Oliver Mörth, BSc

Implementation of Cyber-Physical Systems in Intralogistics

IoT in Conveyor Technology

MASTER'S THESIS

to achieve the university degree of

Diplom-Ingenieur
Mechanical Engineering and Business Economics

submitted to

Graz University of Technology

Supervisor:
Ass.Prof. Dipl.-Ing. Dr.techn. Norbert Hafner
Institute of Logistics Engineering

Second Supervisor:
Dipl.-Ing. Michael Schadler BSc
Institute of Logistics Engineering

In cooperation with:
Dr. Christos Emmanouilidis
Cranfield University

Graz, November 2018

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 marked all material which has been quoted either literally or by content from the used sources. The text document uploaded to TUGRAZonline is identical to the present master's thesis.

…………………………………….. ……………………………………..

Date Signature

ACKNOWLEDGEMENTS

This paragraph is dedicated to acknowledging the persons that were involved in this research project as their effort and commitment were essential for its successful completion.

Only the cooperation between Graz University of Technology (Institute of Logistics Engineering) and Cranfield University enabled the realisation of this project. Furthermore, I would like to thank both organisations for providing all required services and equipment.

I also highly appreciate the support from Dr. Christos Emmanouilidis as academic supervisor at Cranfield University. Focused on the fields of Internet of Things and cyber-physical systems, his detailed feedback and comprehensive guidance contributed significantly to complete this project in high quality.

Finally, I would like to highlight the dedication of Ass.Prof. Norbert Hafner and Michael Schadler as academic supervisors at Graz University of Technology. Only their knowledge and experience in the fields of intralogistics and automation enabled the delivery of an excellent project, whereas their immediate support resulted in a continuous progress.

ABSTRACT

The realisation of a higher business value in manufacturing and warehousing along the product value chain requires optimised internal logistic systems in terms of operational performance, uptime and sustainability. As the Internet of Things (IoT) enables new approaches to achieve that, this project deals with the design and implementation of a demonstrator of a cyber-physical system (CPS) to realise IoT-driven data analytics for a use case in intralogistics as well as with the development of an artificial neural network that can be implemented in a CPS for applications in intralogistics. While the first part covers the basics to develop such a CPS including requirements and appropriate communication protocols, the second part is focused on the actual development of the demonstrator. Starting with the definition of a representative use case and the identification of the related contextual situation, also the separate systems integrated in the demonstrator are presented. First, the data acquisition is performed through a model running on a single-board computer that counts moving and coloured parcels on a conveyor system via a webcam and sends the data to cloud-based channels. Using these channels as source, a program processes the data to extract information by estimating key performance indicators (KPIs). Finally, these KPIs are sent to other channels for their visualisation. This allows monitoring the operating conditions and performance as well as facilitates the analysis of the extracted information to gain new knowledge of the internal logistic system. Besides this demonstrator, another possible CPS to identify the operating conditions of a conveyor system through an implemented classification model is presented. This includes the development of an appropriate structure of a convolutional neural network (ConvNet) that is focused on the specific requirements of a defined use case as well as creating this network. Such CPS enable gaining additional knowledge of internal logistic systems and thus, realising further optimisation in intralogistics.

Keywords: Intralogistics, Internet of Things, Cyber-Physical System, Machine Learning, Artificial Neural Network

ZUSAMMENFASSUNG

Die Realisierung eines höheren Geschäftswerts in der Produktion und Lagerhaltung als Teil der Wertschöpfungskette von Produkten benötigt interne Logistiksysteme optimiert in den Bereichen Betriebsleistung, Verfügbarkeit und Nachhaltigkeit. Da das Internet der Dinge (IoT) neue Ansätze ermöglicht um dies zu erreichen, beschäftigt sich dieses Projekt mit der Entwicklung und Umsetzung eines Vorführers eines Cyber-physischen Systems (CPS) um eine IoT-basierte Daten Analyse für einen Anwendungsfall in der Intralogistik zu realisieren sowie mit der Entwicklung eines künstlichen neuralen Netzwerkes das in ein CPS für Anwendungen in der Intralogistik implementiert werden kann. Während der erste Teil die Grundlagen zur Entwicklung solch eines CPS inklusive Voraussetzungen und angemessenen Kommunikationsprotokollen abdeckt, ist der zweite Teil auf die eigentliche Entwicklung des Vorführers ausgerichtet. Beginnend mit der Definition eines repräsentativen Anwendungsfalls und der Identifizierung der gegebenen kontextbezogenen Situation werden anschließend auch die separaten Systeme die im Vorführer eingebunden sind präsentiert. Zuerst erfolgt die Datenerfassung, wobei ein Modell das auf einem Einplatinencomputer läuft die transportierten und gefärbten Pakete auf einem Förderkreislauf mittels einer Webcam zählt und diese Daten zu Cloud-basierten Kanälen sendet. Dort erfolgt eine Bearbeitung der Daten um Informationen durch die Bestimmung von Leistungskennzahlen zu extrahieren. Abschließend werden diese Kennzahlen zu weiteren Kanälen gesandt und dort visualisiert. Dies ermöglicht die Überwachung der Betriebsbedingungen sowie Betriebsleistungen und unterstützt eine Analysierung der extrahierten Informationen um neues Wissen über das interne Logistiksystem aufzubauen. Zusätzlich wird ein mögliches CPS zur Identifizierung der Betriebsbedingungen eines Förderkreislaufes durch ein implementiertes Klassifizierungsmodell präsentiert. Dies beinhaltet die Entwicklung einer geeigneten Struktur eines Convolutional Neural Network (ConvNet) das auf die speziellen Bedingungen eines definierten Anwendungsfalles ausgerichtet ist sowie die Erstellung dieses Netzwerks. Solch CPS ermöglichen den Aufbau zusätzlichen Wissens um weitere Optimierungen in der Intralogistik zu realisieren.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF EQUATIONS

LIST OF ABBREVIATIONS

CNN, ConvNet	Convolutional Neural Network
CPS	Cyber-Physical System
ERP	Enterprise Resource Planning
IoT	Internet of Things
IT	Information Technology
MES	Manufacturing Execution System
KPI	Key Performance Indicators
OT	Operational technology
PLC	Programmable Logical Control

1 Introduction

1.1 Rationale

Following the product value chain ranging from raw material, via finished goods to the final distribution to customers, important processes are manufacturing, warehousing and transportation. In order to perform these processes efficiently, well-working logistic systems have become absolutely essential. While general logistics comprises all necessary transport activities, intralogistics is only focused on the transport of goods within the physical limits of a single site. In order to realise benefits and thus, to obtain a higher business value, optimised internal logistic systems in terms of operational performance, uptime and sustainability are required. One way to improve internal logistic systems is through automation. While identification and sortation processes have already reached a high degree of automation, other tasks, such as un-/loading, singulation and commissioning are still performed manually. However, traditional automation in intralogistics also brings along challenges and limiting opportunities to achieve full optimisation. Furthermore, as current internal logistic systems are already very complex, the abilities of human operators to optimise them have reached their limits. (Fritz, 2016; Gilchrist, 2016; Granlund and Wiktorsson, 2014)

New approaches, enabled by the Internet of Things (IoT), include monitoring and analysing the operating conditions and performance of the internal logistic system through IoT-driven data analytics. An effective way to realise this represents the implementation of a cyber-physical system (CPS) that allows the determination of contextual information as well as its visualisation as key performance indicators (KPIs) on a dashboard. Furthermore, analysing this information enables gaining new knowledge. One possible application of that knowledge is the optimisation of the internal logistic system in a way that is much harder for traditional methods to match. Simultaneously, a higher transparency of the operating conditions and performance is provided. Therefore, the realisation of IoT-driven data analytics in intralogistics results in a higher quality of the involved data value chain and the capability to realise a higher business value. (Curry, 2016; Guerrieri, 2016; Macaulay et al., 2015; Scholze and Barata, 2016)

Besides realising IoT-driven data analytics, the implementation of machine learning applications in a CPS represents another approach that enables gaining new knowledge of an internal logistic system and thus, realising optimisation in intralogistics. One example demonstrates the identification of specific conditions of an internal logistic system through image recognition. This might be realised through a classification model that classifies images depicting parts of a conveyor system into several categories, whereas each category represents one certain condition. While using artificial neural networks to solve classification problems offers several benefits in data volume and speed, convolutional neural networks provide the highest classification accuracy in image recognition. (Bell, 2014; Intralogistik-Netzwerk BW, 2018; Zeiler and Fergus, 2014)

This project includes the design and implementation of a CPS demonstrator to realise IoT-driven data analytics in intralogistics. Therefore, the demonstrator determines and visualises appropriate KPIs of the operating conditions and performance of an existing conveyor system. This requires data of the physical system to be acquired, stored and processed. Furthermore, the development of a convolutional neural network that can be implemented in a CPS to identify the operating conditions of a conveyor system is included.

This study answers the following research questions:

1. **How can key performance indicators of an internal logistic system be determined and visualised via the Internet of Things to enable gaining new knowledge of its operating conditions and performance?**
2. **How can a machine learning application be implemented in a cyber-physical system to enable gaining additional knowledge of an internal logistic system?**

1.2 Aim and Objectives

The aim of this project is the realisation of IoT-driven data analytics in intralogistics via the implementation of a demonstrator of a cyber-physical system that determines and visualises key performance indicators of a representative use case. Furthermore, the development of a classification model as machine learning application that can be implemented in a CPS to identify specific conditions of an internal logistic system is desired.

In order to achieve this aim, the following objectives were set:

1. Understanding of the topics intralogistics, IoT-enabled data value chains, cyber-physical systems and machine learning.
2. Design and implementation of a CPS demonstrator that determines and visualises KPIs of a representative use case in intralogistics.
3. Testing and evaluation of the outputs of the CPS demonstrator.
4. Performance of updates to improve the CPS demonstrator.
5. Development of a classification model that can be implemented in a CPS to identify specific conditions of an internal logistic system.
6. Documentation of the results.

1.3 Deliverables

This research project includes the following deliverables:

1. Project baseline
    a. Literature review report
    b. Definition of a representative use case in intralogistics
2. CPS demonstrator for this use case
    a. Data acquisition system
    b. Data processing and information visualisation system
3. Test and evaluation results of the outputs of the CPS demonstrator
4. Classification model that can be implemented in a CPS to identify specific conditions of an internal logistic system
5. Academic documentation

1.4 Project Scope and Assumptions

The developed CPS demonstrator is focused on the existing conveyor system at Graz University of Technology (Institute of Logistics Engineering). This conveyor system represents an appropriate use case in intralogistics as it involves typical technologies and practices. However, the demonstrator is only one possible solution to realise IoT-driven data analytics in intralogistics. Furthermore, training the developed convolutional neural network is not included in the project scope as the required images depicting the four operating conditions of the conveyor system were unavailable.

The present master's thesis “Implementation of Cyber-Physical Systems in Intralogistics - IoT in Conveyor Technology” is based on the master's thesis “Implementation of IoT-Driven Data Analytics for Intralogistics” that was submitted to Cranfield University in August 2018. As both documents differ in their scope and content, they represent two separate theses.

2 Literature Review

This chapter provides a proper understanding of the required fundamentals to successfully answer the research questions. Therefore, corresponding research areas were identified and investigated. Starting with the implementation of a cyber-physical system in intralogistics to realise IoT-driven data analytics, the first research area was intralogistics including limitations, applications, main activities and objectives. This understanding built the basis of the following investigation of IoT-enabled data value chains in intralogistics. While the first part of this area described current applications of IoT in intralogistics, the second part introduced IoT-enabled process control. Subsequently, the effects of IoT on data value chains including the different levels from raw data to applied knowledge were explored. The last two parts focused on the practical application of the steps of IoT-enabled data value chains. This involved IoT-enabled performance measurement including the identification of appropriate KPIs in intralogistics as well as the application of the gained knowledge to realise the optimisation of internal logistic systems. Cyber-physical systems in intralogistics represented the third research area, whereas the analysis of the related literature enabled the development of a conceptual model of a CPS that represents the current understanding. The detailed investigation of these three research areas resulted in a proper understanding of the required fundamentals to realise IoT-driven data analytics in intralogistics. However, in order to answer the second research question successfully, the fourth and last research area was machine learning in intralogistics. This understanding supported the development of an artificial neural network that can be implemented in a CPS for applications in intralogistics.

2.1 Introduction to Intralogistics

In order to provide a comprehensive introduction to intralogistics, the first step is to understand its position within general logistics. Basically, logistics represents the management of the flow of objects from their origin to points of consumption. While tangible logistic objects represent the several types of physical goods, intangible objects represent the related order information. All objects are moving through a logistic system that consists of transport and information systems. Furthermore, the seven rights of logistics are presented as providing the right quantities of the right product to the right customer at the right place in the right conditions within the right time for the right price. The aim of logistic management is to meet these seven rights in the most efficient way and thus, with suitable quality and minimum cost. Depending on the involved actors and boundaries of a logistic system, different classifications can be made. The top level of these classifications builds macro-logistics and micro-logistics. While macro-logistics deals with the efficiency of a broad logistic infrastructure within a country or a region, micro-logistics focuses on single orders and the required logistic system to fulfil them. Taking a closer look on micro-logistics in a single organisation, the logistic systems can be classified into internal and external logistics. Internal logistics, also called intralogistics, refers to systems within the physical limits of a single site as illustrated in Figure 2-1. That includes the internal transportation, material handling and warehousing as well as all related activities between the receiving and shipping docks. Therefore, internal logistic systems represent closed systems, whereas inputs and outputs as well as the internal paths are clearly defined. In contrast, external logistics covers everything that comes beyond a production site or warehouse. In a simplified representation, that includes everything between external or internal suppliers and customers that is not part of intralogistics. (Bode and Preuß, 2005; Gudehus and Herber, 2012; Granlund and Wiktorsson, 2014; Nagel et al., 2008)

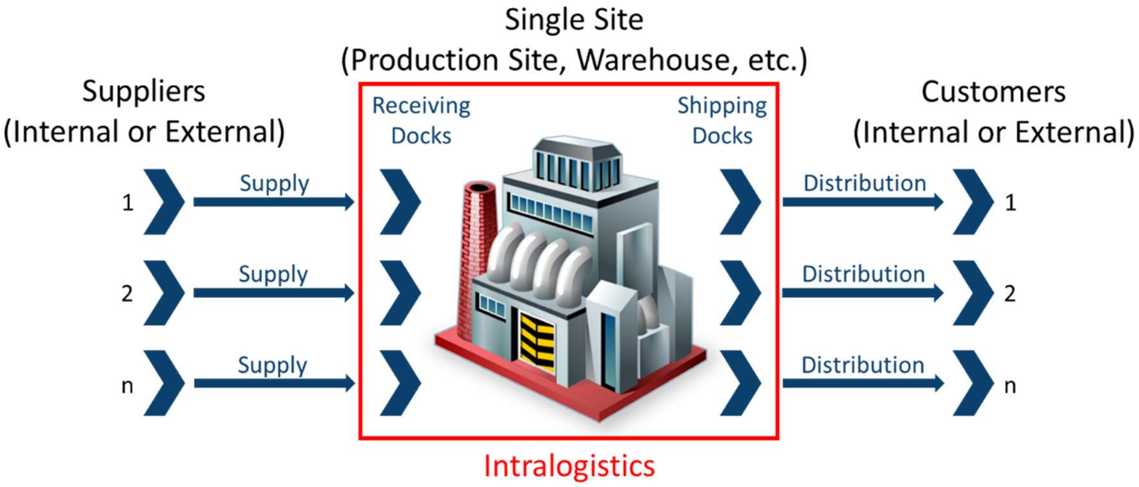


Figure 2-1: Boundaries of Intralogistics

Furthermore, Gudehus and Herber (2012) point out that a logistic system generally consists of elements that are connected with each other through specific relationships. These elements are called performance stations and involve tangible (goods) or intangible (information) inputs and outputs. Examples of performance stations of internal logistic systems are machines, robots, storing areas, commissioning systems and more. All these stations are connected by several types of internal transport systems. The main task of an internal logistic system can be defined as providing the necessary physical goods to the local value adding activities. In terms of manufacturing and warehousing, these goods range from raw material to finished products. Having understood the general concepts of intralogistics through critically analysing the literature, the main activities in intralogistics including examples were developed and are shown in Table 2-1. (Bode and Preuß, 2005; Gudehus and Herber, 2012; Granlund and Wiktorsson, 2014; Moris, 2013)

Table 2-1: Main Activities in Intralogistics

No.	Activity	Examples
1	Receiving	Unloading, un- and repacking, quality control, labelling, etc.
2	Processing	Singulation, sorting, internal transport to storing area, etc.
3	Storing	In-storing, store keeping, out-storing, etc.
4	Commissioning	Order picking (collecting of required quantities of access units according to orders), internal transport to packing area, replenishment of reserve units, etc.
5	Packing	Order consolidation, preparing and filling packages/ containers/ pallets, compressing, sealing, labelling, etc.
6	Dispatch	Internal transport to loading dock, preparing for shipment, final control, checkout, loading, etc.

In order to perform these activities efficiently and effectively, several types of transport and information systems have been developed. However, Gudehus and Herber (2012) emphasise that mainly the flow of physical goods determines an optimal internal logistic system. This underlines the importance of IoT-driven data analytics as it represents a new approach to effectively monitor relevant information of this physical flow. The most suitable type of an internal transport system depends on different criteria, such as the type of objects, the required capacity and flexibility, the desired degree of automation and more. Furthermore, the optimal system provides the entire range of the needed criteria at lowest operational cost. Therefore, having the optimal system is a key to success in intralogistics. This is often only possible through the combination of different types of transport systems. The two main types in intralogistics are discontinuous and continuous conveyor systems, whereas each of these types can be classified in an overhead (non-floor bound) and a ground (floor bound) system. The main characteristics of continuous conveyor systems are the fixed path and the similarity of transported goods. Furthermore, such systems are typically automated and provide a high throughput at relatively low operational cost. However, the low flexibility and high investment cost are disadvantages. Well known samples of continuous conveyor systems are roller and belt conveyors. In contrast, the main characteristics of discontinuous conveyor systems are the free path and the strong variation of transported goods. Typically, the transport is performed manually and the throughput is relatively low at high operational cost. The advantages of such discontinuous conveyor systems are the high flexibility and low investment cost. (Bode and Preuß, 2005; Gudehus and Herber, 2012)

The literature commonly defines the goal of any logistic system as providing the maximum service at minimum costs. Derived from this goal, the objectives of intralogistics can be summarised as shown in Figure 2-2. However, due to the contradictoriness of these objectives, identifying the optimal balance is important. As achieving these objectives allows to gain a competitive advantage, it is essential for companies to optimise their internal logistic systems. (Bode and Preuß, 2005; Gudehus and Herber, 2012)

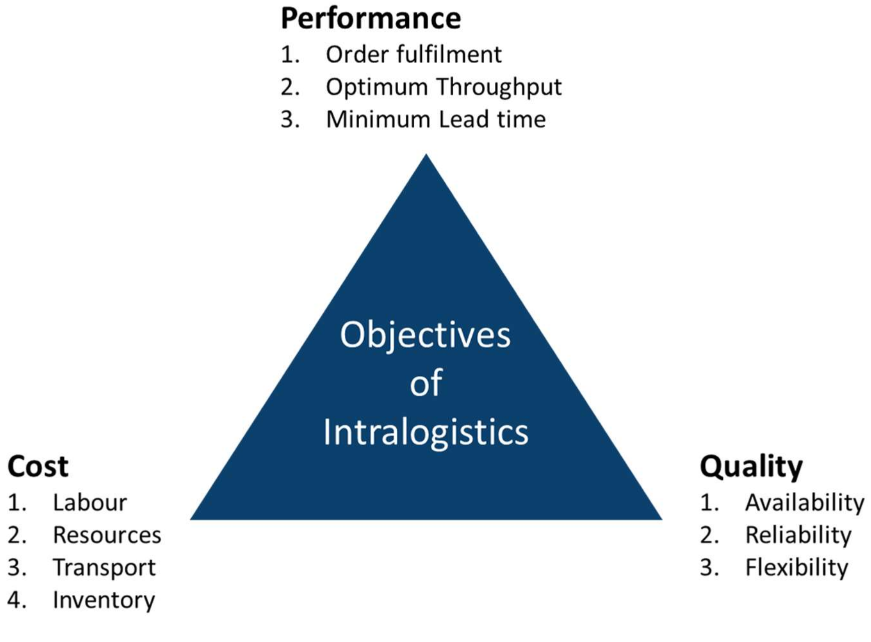


Figure 2-2: Objectives of Intralogistics

2.2 IoT-enabled Data Value Chains in Intralogistics

2.2.1 Internet of Things in Intralogistics

Nowadays, the trend towards digitalisation is omnipresent in almost every business sector. However, the concept of Industry 4.0, known as the fourth industrial revolution, lies beyond ordinary digitalisation as it includes many different disciplines and technologies. The Internet of Things (IoT), big data, cloud computing and cyber security are some examples. (Colombo et al., 2017; Mendes et al., 2017)

Since the concept of IoT was introduced 1999, it has been advanced a lot through smaller and less expensive sensors, improved communication technologies and cloud computing. Basically, IoT can be seen as the natural evolution of the concept of M2M (Machine-To-Machine) and the traditional internet. Typical objects in intralogistics, such as forklifts, belt and roller conveyors, storage places, sorting machines, even staff and more, that were not connected to the internet because it is not required to perform the actual task, are now connected. Quite simply, IoT represents the connection of physical objects with the digital world through a network that enables the communication with each other and the environment. Therefore, these objects are equipped with sensors and actuators to sense and manipulate their physical environment. Basically, two different types of objects can be distinguished. The first type includes all objects with integrated data processing and decision-making tools (decentralised intelligence), whereas the second type includes objects where the data processing takes place on a remote platform (centralised intelligence). The term smart object implies the capability to sense, understand and manipulate the environment autonomously. Furthermore, the increasing number of devices connected to the internet results in a growing volume of data. In the literature, this is referred to as big data. Advanced data management and computation tools are required to handle that big volume of data properly. (Gilchrist ,2016; Lee et al., 2015; Macaulay et al., 2015; Mendes et al., 2017; Hribernik et al., 2011)

Several technologies related to IoT have been used in intralogistics for years. Well known examples are identification and sorting technologies to track and control the internal flow of goods. Typical benefits of IoT in intralogistics are improved operational efficiency, transparency, customer experience, uptime and more. Wireless transponder technologies such as RFID enable even further improvements in terms of speed and accuracy. Furthermore, the development and improvement of RFID has enabled decentralised data management in logistics. Therefore, the data is saved on the transponder that is located on the physical good. That results in an integrated information and material flow. Such smart logistic objects contain frequently updated information about their own conditions including the current location, item number and more. IoT also offers the opportunity to change working methods in fields with mostly manual activities. A facilitated commissioning process, supported by augmented reality that provides a list including the locations of the demanded objects, serves as one example. Another key area of IoT in intralogistics is preventive maintenance. Monitoring the levels of several indicators for a potential malfunction, such as vibration, speed, temperature, etc. and notifying responsible staff in case the values exceed acceptable levels enables to reduce the downtime and thus, to increase the reliability of the internal logistic system significantly. Furthermore, IoT allows the improvement of the equipment utilisation as idle or over-utilised equipment can be easily identified. Another application of IoT in intralogistics is the optimisation of internal logistic systems in terms of operational performance, uptime and sustainability. This is enabled through IoT-driven data analytics. (Gilchrist ,2016; Macaulay et al., 2015; Nagel et al., 2008; Hribernik et al., 2011)

2.2.2 IoT-enabled Process Control in Intralogistics

The literature classifies general automation technologies in five separate levels, represented in the traditional automation pyramid. Furthermore, the levels of the pyramid are in accord with the functional hierarchy of the international standard ISA-95, whereas the link is visualised in Figure 2-3. ISA-95 consists of five different parts and provides methods to deal with problems associated with the implementation of interfaces between enterprise and control systems. The first part of ISA-95 presents the functional hierarchy model and focuses on the automated information exchange between ERP (Enterprise Resource Planning) and MES (Manufacturing Execution System) systems. The hierarchy comprises five levels, whereas level 0 represents the actual production process, level 1 and 2 represent the production process control, level 3 represents the additional activities to prepare, monitor and complete the production process and level 4 represents the financial and logistic activities. Furthermore, ISA-95 classifies these levels in 2 domains. While the enterprise domain covers level 4 and deals with longer time horizons, the control domain includes all other levels from 0 to 3 and deals with shorter time horizons. However, the structure of the automation pyramid changes with the implementation of industry 4.0. While the field level remains separate, the levels above might be integrated into a cloud-based system. Therefore, new opportunities to improve the communication between management and shop floor are enabled through IoT-driven data analytics. The implementation of a system that automatically acquires and processes data from the field level to provide the right information at the right time to the right person in higher levels represents one possible realisation. (Henderson, 2013; Hwang et al., 2016; ISA-95.com, 2017; Mendes et al., 2017; Nagorny, 2016; Scholten, 2007)

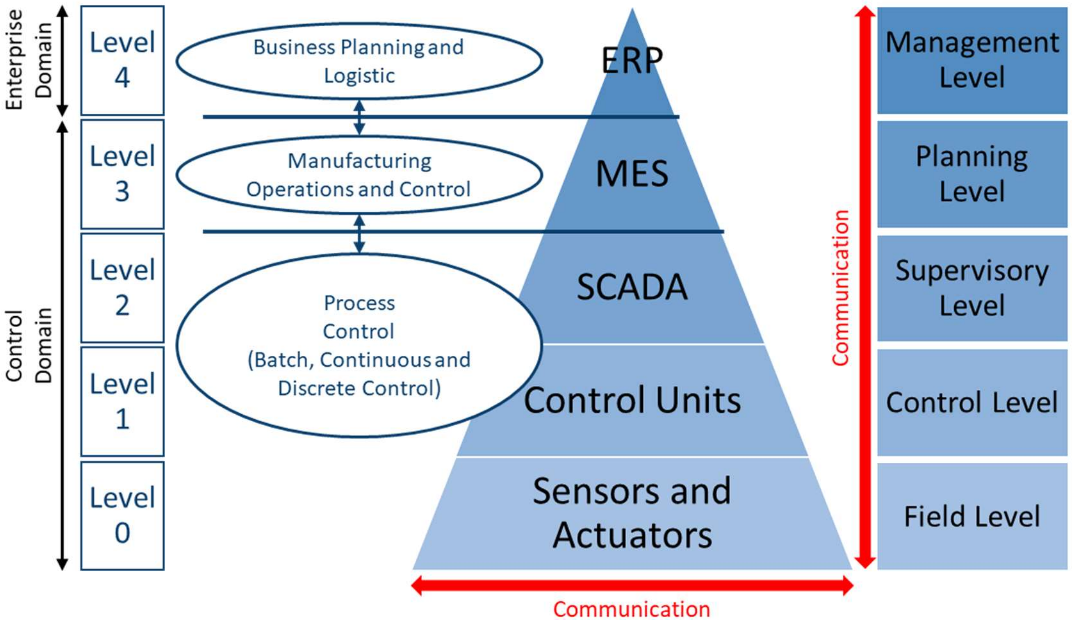


Figure 2-3: Functional Hierarchy of ISA-95 and Automation Pyramid

2.2.3 IoT-enabled Data Value Chains

IoT also leads to new forms of data value chains. That results in improvements of the gained knowledge and thus, the decision-making processes. Typical value chains represent the activities that add value to a service or a product, whereas each activity can be seen as subsystem including inputs, outputs and a related transformation process. Applied on a given data flow, this allows a better understanding of the individual activities as well as the overall value creation process. In terms of an IoT-enabled data value chain, the process of transforming raw data (lowest value) into applied knowledge (highest value) is described. The different levels between raw data and applied knowledge can be summarised as: data, information, knowledge and wisdom (DIKW). Figure 2-4 presents a high-level data value chain. The first step is the data acquisition, which describes the process of generating, collecting and storing data. This is followed by the data processing, whereas information is mostly extracted through computation technologies. The third step is the analysis of this information by employing appropriate methods and tools to gain knowledge. The final step is the application of the gained knowledge. This is called wisdom and results in improved decisions. Using the presented data value chain in the field of intralogistics, new knowledge of the operating conditions and performance of the internal logistic system can be gained. The application of that knowledge allows taking corrective or preventive actions to realise the optimisation of internal logistic systems in terms of operational performance, uptime and sustainability. (Baškarada and Koronios, 2013; Curry, 2016; Miller and Mork, 2013)

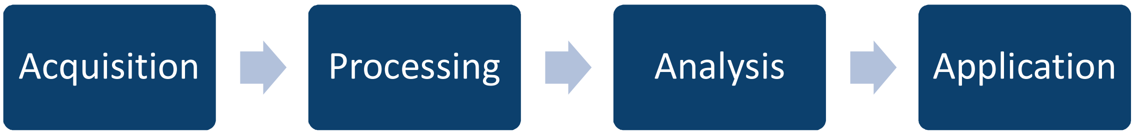


Figure 2-4: Data Value Chain (adapted from Curry, 2016)

As simple but effective tool to distinguish between the four levels from raw data to applied knowledge serves the DIKW hierarchy, illustrated in Figure 2-5 as pyramid. Baškarada and Koronios (2013) as well as Rowley (2007) describe data as providing only limited value to users. It is typically generated by installed sensors and represents the raw level in the hierarchy. Only processing that data enables extracting information that is described as structured and interpreted data. Some of the several definitions of knowledge in the literature express that new knowledge is gained through understanding contextual information and applying existing knowledge. As it depends on personal interpretations, knowledge is subjective. Therefore, providing the same information to individuals might result in gaining different knowledge. Wisdom represents the top level of the DIKW pyramid and is described as the understanding of best ways to apply the gained knowledge in new situations. (Baškarada and Koronios, 2013; Mendes et al., 2017; Perera et al., 2014; Rowley, 2007)

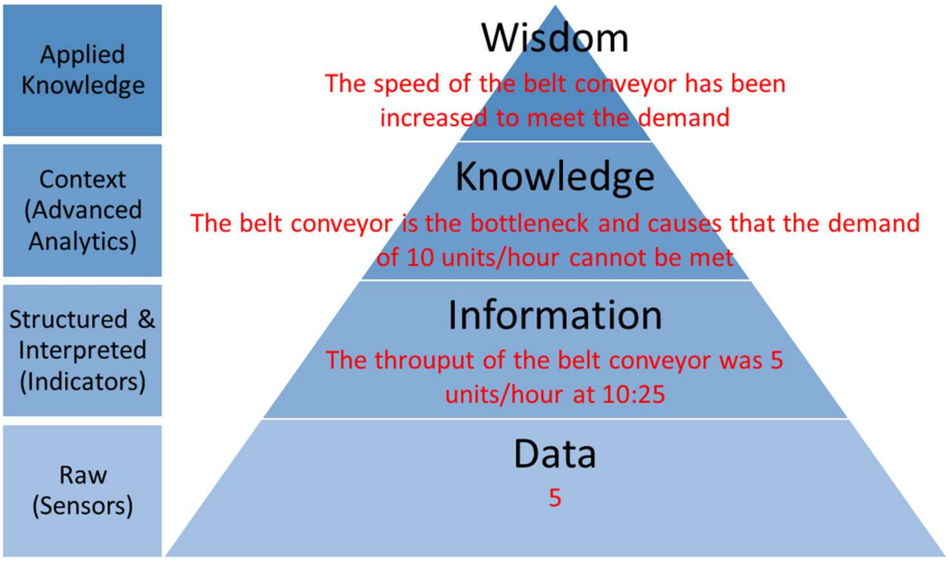

Applied Knowledge
Context (Advanced Analytics)
Structured & Interpreted (Indicators)
Raw (Sensors)
Wisdom
The speed of the belt conveyor has been increased to meet the demand
Knowledge
The belt conveyor is the bottleneck and causes that the demand of 10 units/hour cannot be met
Information
The throuput of the belt conveyor was 5 units/hour at 10:25
Data
5

Figure 2-5: Data Information Knowledge Wisdom-Pyramid (adapted from Baškarada and Koronios, 2013)

A relatively new approach to improve the effectiveness of data value chains is the incorporation of the contextual situation under which the physical system is operating. This is called context-aware computing and enables identifying the characteristics of specific situations. Analysing the contextual information results in a better understanding of the physical system and thus, in gaining more knowledge effectively. Since context awareness was introduced in the 1990s, it has become more popular due to web applications, mobile phones and IoT. In terms of IoT, context awareness facilitates handling the enormous volume of data as it supports the identification of relevant data. According to Perera et al. (2014), the five Ws (Who, What, Where, When and Why) represent the minimum contextual information that is required to understand different situations sufficiently. (Perera et al., 2014; Scholze and Barata, 2016)

2.2.4 IoT-enabled Performance Measurement in Intralogistics

The first two steps of the presented data value chain describe the process of performance measurement, that aims to provide information through the determination of key performance indicators (KPIs). In the field of intralogistics, the KPIs are used to quantify the operating conditions and performance of internal logistic systems to enable their monitoring. Nevertheless, Granlund and Wiktorsson (2014) have identified that several analytic issues are caused by missing or underdeveloped performance measurement systems in intralogistics. This demonstrates the difficulty of providing the relevant information comprehensively and effectively. However, Hwang et al. (2016) describe the realisation of improved performance measurement through the implementation of IoT. Furthermore, the developed system is in accordance with ISO-22400. This standard specifies 35 KPIs used in MES (Manufacturing Execution System), defines their applications and benefits as well as presents their formulas and elements. Comparing actual data from the shop floor and planed data for example provided by MES enables identifying performance gaps as well as tracking the progress of closing these gaps. (Gilchrist, 2016; Granlund and Wiktorsson, 2014; Hwang et al., 2016)

The first and most important step in developing a performance measurement system is the identification of suitable KPIs. Therefore, the understanding provided by the literature as well as the consideration of the standard ISO-22400 led to the identification of ten appropriate KPIs in intralogistics. An overview of these KPIs and the required data to calculate them is shown in Table 2-2. While the first six indicators are used to evaluate the operational performance, the seventh and eight indicators are dedicated to the uptime. The ninth indicator provides information about the sustainability. Finally, the last indicator represents the overall equipment effectiveness. (Gudehus and Herber, 2012; Hwang et al., 2016)

Table 2-2: Suitable Key Performance Indicators in Intralogistics

No.	KPIs	Required Data
1	TH: Throughput [units/s] TH = TQ/RT	TQ: Transported quantity [units]
		RT: Reference time [s]
2	CT: Cycle Time [s] CT = TL/TS	TL: Length of transport way [m]
		TS: Transport speed [m/s]
3	TP: Transport Performance [kg/m] TP = (TQ/RL) x MG	TQ: Transported quantity [units]
		RL: Reference length [m]
		MG: Mass per good [kg/unit]
4	TrU: Transport Utilisation [%] TrU = TH/TC x 100	TH: Throughput [units/s]
		TC: Transport capacity [units/s]
5	TiU: Time Utilisation [%] TiU = ATT/OT x 100	ATT: Actual transporting time [s]
		OT: Operation time [s]
6	EF: Effectiveness [%] EF = ATT/PTT x 100	ATT: Actual transporting time [s]
		PTT: Planned transporting time [s]
7	AV: Availability [%] AV = ATT/PBT x 100	ATT: Actual transporting time [s]
		PBT: Planned busy time [s]
8	RE: Reliability [%] RE = (ADT-PDT) / OT x 100	PDT: Planned down time [s]
		ADT: Actual down time [s]
		OT: Operation time [s]
9	EE: Energy Efficiency [kWh/unit] EE = EC/TQ	EC: Energy consumption [kWh]
		TQ: Transported quantity [units]
10	OEE: Overall Equipment Effectiveness [%] OEE = EF x AV /100	EF: Effectiveness [%]
		AV: Availability [%]

2.2.5 IoT-enabled Optimisation in Intralogistics

As the previous part of the literature review has pointed out, optimising internal logistic systems enables companies to gain a competitive advantage. Typical ways to realise such an optimisation include redesigning the structure and the involved processes. Granlund and Wiktorsson (2014) argue that the effectiveness and efficiency of an internal logistic system can be improved through increasing the degree of automation as many activities, such as un-/loading, singulation and commissioning are currently still performed manually. This causes the arise of potential danger and damage to staff and equipment as well as competitive disadvantages in terms of longer lead time and higher cost. In contrast, replacing these manual activities with automated processes enables realising several benefits including improved productivity, labour cost savings, improved service quality, less risk of damage and more. This results in an improved overall competitiveness of the entire organisation. Nowadays, automation is already used in some specific areas of intralogistics, whereas automatic identification systems provide the basis and enable automated sorting, storage and retrieval systems. In order to ensure a positive effect and thus, a competitive advantage, it is necessary to identify the most appropriate degree of automation for an internal logistic system. Therefore, the development of a strategy including requirements, objectives, and more is essential, whereas that strategy has to be aligned with the overall business strategy of the organisation. The main challenge that has been identified by Granlund and Wiktorsson (2014) is the proper handling of the technology and equipment as this requires higher competences and skills of users. Consequently, traditional automation in intralogistics also brings along challenges and limiting opportunities to achieve full optimisation. Furthermore, as current internal logistic systems are already very complex, the abilities of human operators to optimise them have reached their limits. This leads to the need of IoT-driven data analytics to enable further optimisation. (Bode and Preuß, 2005; Gilchrist, 2016; Gudehus and Herber, 2012; Granlund and Wiktorsson, 2014)

IoT-driven data analytics enables monitoring the operating conditions and performance of internal logistic systems as well as facilitates the analysis of the extracted information. While analysing the information allows gaining new knowledge and thus, improves the decision-making processes, applying this knowledge by taking corrective or preventive actions enables the realisation of optimised internal logistic systems in terms of operational performance, uptime and sustainability. This represents the last two steps of the data value chain and ends at the top level of the DIKW-pyramid. Furthermore, Gudehus and Herber (2012) emphasise the importance of looking at a system in a dynamic and static point of view to realise the best optimisation of the logistic system. While a dynamic point of view facilitates the optimisation of the involved processes, a static point of view facilitates the optimisation of the structure. In case the logistic system is used differently for various tasks, it is essential to analyse all critical scenarios. Effective ways to perform such an analysis include simulations. In order to justify the cost to realise IoT-enabled optimisation in intralogistics, Gilchrist (2016) underlines the “power of 1%”. This means that saving only 1% of the operational cost of internal logistic systems might results in a significantly positive impact on the profit and thus, the return on investment. (Gilchrist ,2016; Granlund and Wiktorsson, 2014; Gudehus and Herber, 2012)

2.3 Cyber-Physical Systems in Intralogistics

A cyber-physical system (CPS) represents the integration of a physical and a digital system that allows interactions in both ways. While IoT focuses on single objects and serves as key enabler of CPS, a CPS describes the overall system. The digital system can be cloud based, which enables accessing the data and information from several devices through the internet. A CPS involves operational technologies (OT) and information technologies (IT). While OT includes software and hardware that is used to measure and manipulate the condition of a physical system through monitoring and control, IT includes software and hardware that is used to send, receive, store, and process data. Computation technologies are used to process the data and extract the required information. Furthermore, Scholze and Barata (2016) argue that using CPS is the most appropriate way to create context aware systems. In terms of performance measurement, a CPS facilitates the determination of KPIs significantly. Applied in intralogistics, the implementation of a CPS represents an effective way to realise IoT-driven data analytics. This might involve acquiring data from the shop floor, whereas processing the data enables extracting contextual information about the working conditions and performance of internal logistic systems. The visualisation of the extracted information as key performance indicators (KPIs) on a dashboard could be the final step. (Ballarino et al., 2017; Lee et al., 2015; Marwedel et al., 2016; Sanislav and Miclea, 2012; Scholze and Barata, 2016; Wiesner et al., 2014)

The literature provides several conceptual models of general CPS. An overall synthesis, representing the current understanding of CPS concepts, is shown in Figure 2-6. Using a three-tier architecture, physical objects equipped with sensors and actuators build the basis of the concept. The first tier is called edge tier and comprises the input/output-devices and edge nodes. While each object can be equipped with multiple sensors or actuators, it is typically allocated to only one edge node. The data is acquired through sensors and transmitted to the edge node, whereas a local network (LAN) enables the communication between the devices. Edge nodes represent control units and vary from single-board computers, programmable logical controls (PLC) to desktop computers. They manage the connected sensors and actuators as well as perform a first data processing. This enables reducing the volume of the data that is transmitted to the following tier significantly. It is called platform tier and performs the synthesis of contextual information. Therefore, a platform receives data and simple information from one or several edge nodes via a computer network (internet) and is responsible for their transforming, storing and further processing. Typical platforms provide data management as well as computation tools. The final tier is called enterprise tier and provides the actual user interfaces including applications to visualise the relevant information. Applying analytics enables gaining new knowledge and thus, improves the decision-making processes. The knowledge is applied by taking corrective or preventive actions (e.g. to optimise the internal logistic system) that directly affect the edge tier and consequently the physical world. Furthermore, applications of the platform and of the enterprise tier are typically connected through a computer network (internet). The local and computer network represent the communication network of a cyber-physical system. (Babovic et al., 2016; Colombo et al., 2017; Gilchrist ,2016; Perera et al., 2014; Sanislav and Miclea, 2012)

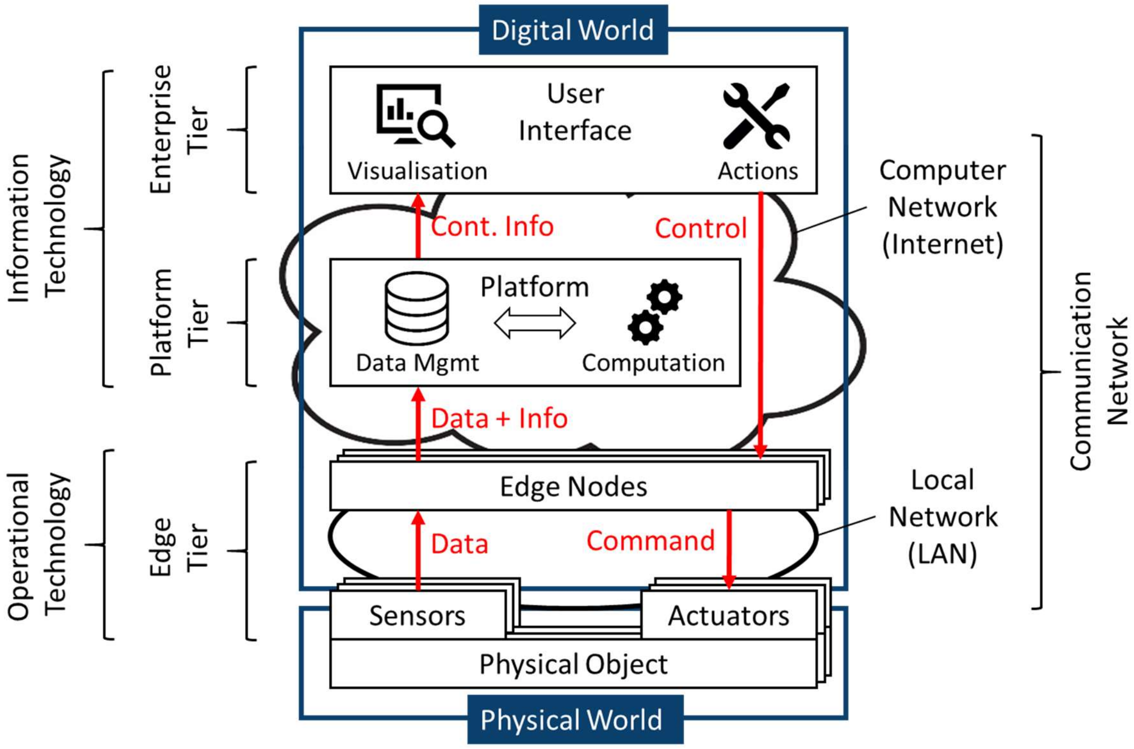


Figure 2-6: Conceptual Model of a Cyber-Physical System

The proper handling of the huge amount of data (big data), caused by thousands of sensors in the industrial use of IoT, requires advanced data management and computation tools. However, cloud computing provides the means to cope with these challenges. It represents a service that offers access to shared system resources, such as infrastructure (e.g. communication), platform (e.g. storage at a database) and software applications (e.g. computation) via a network (e.g. the internet). Therefore, a cloud-based system offers scalable resources, high availability and reduction of the upfront investment due to the "pay-as-you-go" model. Depending on the use case, different types of cloud-systems are more attractive. The concept of a private cloud provides the highest degree of cyber security. In that case, the cloud is either managed in-house or by a third party via the internet. While clouds are typically located in the internet, the term fog describes a cloud infrastructure that is located close to the edge. Zhang et al. (2018) describes fog computing as network of interconnected fog nodes located between the edge devices and the platform, whereas each node provides resources to enable simple computation, data storage and more. Furthermore, the fog uses the local network for communication. Previous studies have pointed out that fog-based systems reduce latency issues and thus, improve real-time systems. However, even though fog computing offers a reduced latency compared to cloud computing, Gilchrist (2018) argues that cloud computing cannot be completely replaced due to its advantages in terms of advanced databases and processing tools to handle big data. (Aazam et al, 2018; Dizdarevic et al., 2018; Gilchrist ,2016; Marwedel et al., 2016; Lee et al., 2015; Zhang et al., 2018)

2.4 Machine Learning in Intralogistics

2.4.1 Introduction to Machine Learning

In contrast to widely spread opinions that artificial intelligence (AI) means computer systems are capable of “thinking” like human beings, that is not necessarily the case. It simply represents their potential to behave intelligently. Such intelligence reflects the capability to learn from experience gained in a changing environment. Hence, the system is able to autonomously take specific actions that improve its performance towards a specific goal. Following this idea eventually led to the development of machine learning as a branch of AI, whereas its concept has been improved further during the last years. Current applications of machine learning in the industry comprise spam detection, speech and image recognition, stock trading, and more. (Bell, 2014; Paluszek and Thomas, 2016)

Furthermore, experts in multiple fields have identified an increasing trend of machine learning combined with the new opportunities provided by big data in logistics. While Intralogistik-Netzwerk BW (2018) claims that machine learning and AI will become standard implementations in all material flow system soon, it also expects these technologies to become employed in many more applications within the field of intralogistics. One example is the optimisation of internal logistic systems, whereas new knowledge gained through machine learning applications build the basis. (Intralogistik-Netzwerk BW, 2018)

Basically, machine learning represents computer systems that are capable to learn from collected data (experience) through several algorithms. The aim of machine learning is to create a model that, based on the gained experience, estimates outcomes when feeding it with new data. While traditional computer systems are limited in the complexity of solvable problems, machine learning enables solving even complex problems. Therefore, the model automatically determines the optimal values of all used parameters to solve a specific problem. As all learning algorithms are based on data, the data acquisition represents an important part to achieve a high performance of a model. The data might be acquired manually by human operators or automatically through other computer systems. This connects machine learning to the benefits of IoT as it enables acquiring the required data reliably and automatically. Furthermore, machine learning provides algorithms to effectively analyse large data volumes (Big Data). Therefore, the importance of machine learning has increased since the application of IoT started growing. (Alpaydin, 2014; Bell, 2014; MathWorks, 2016; Paluszek and Thomas, 2016)

Machine learning projects mainly include four steps, shown in Figure 2-7. Starting with the data acquisition, this step has been significantly improved through the application of IoT. However, as machine learning algorithms are not able to differ between noise and valuable information, an appropriate preparation of the input data is required to enable the best performance. Therefore, human operators have to ensure that the data is clean and complete. The third step describes the actual processing of the data via machine learning algorithms. Similar to other concepts presented in this report, the final step is the presentation of the received results. (Bell, 2014; MathWorks, 2016)

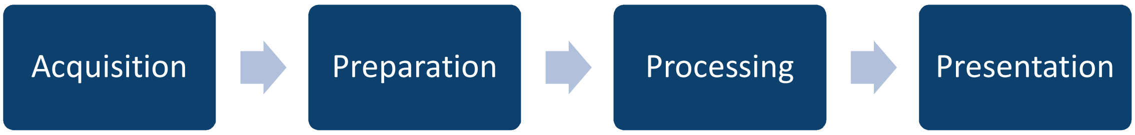


Figure 2-7: Main Steps of a Machine Learning Project

As there is rarely a straight line to create an appropriate model in machine learning, common methods include trial and error. The selection of the right learning algorithm and feature complexity requires compromises between computational speed, estimation accuracy, model complexity and more. Features represent measurable attributes or characteristics of the input data. This simple information is relevant to perform the computational tasks of the machine learning algorithm properly. Therefore, using informative, discriminating and independent features is essential to realise an effective machine learning algorithm. However, well designed models are simple and thus, only involve the features with the most power. (Alpaydin, 2014; MathWorks, 2016)

The two main training techniques in machine learning are shown in Figure 2-8. Supervised learning indicates that both, input and correct output data are provided to the model during its training. As supervised learning algorithms use known input and output data, it enables creating a model that generates the most reasonable estimations when using new input data. Therefore, it is mainly used in cases the required input and output data are available. In contrast, unsupervised learning is based only on the input data and thus, used in cases the related output data is unavailable for the training. Unsupervised learning enables discovering new and unknow patterns in data, whereas finding such hidden structures represents an important benefit as it enables building a better understanding of the data. Taking a closer look at the involved algorithms, supervised learning mainly deals with classification and regression algorithms. While classification models classify input data into categories (discrete response), e.g. whether an email is genuine or spam, regression models provide continuous responses such as the change of the temperature depending on multiple factors. In contrast, unsupervised learning mainly deals with clustering algorithms to find hidden patterns or groupings with similar features within the input data. The selection of the best learning technique still depends on the experience of the model developer, the size and type of the data as well as the expected result. Problems might arise when the training data for supervised learning do not sufficiently cover the full range of possible inputs and outputs. In that case, the system either does not know what the input data represents or to what output class the data belongs. (Alpaydin, 2014; Bell, 2014, MathWorks, 2016; Paluszek and Thomas, 2016)

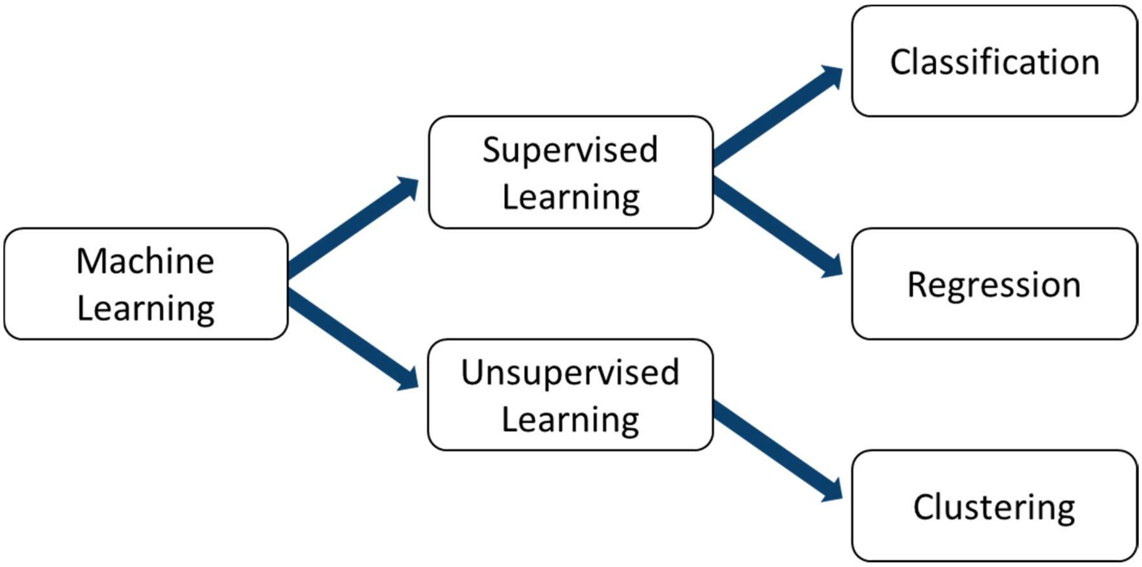


Figure 2-8: Machine Learning Techniques (adapted from MathWorks, 2016)

2.4.2 Classification Algorithms

Quite simply, an algorithm represents a certain process that is performed to solve a specific problem and results in transforming input into output. Even though multiple algorithms are employed in machine learning, supervised learning mainly deals with classification and regression algorithms as shown in the previous figure. Especially classification models represent an important application of machine learning to realise optimisation in intralogistics. Having a classification problem, determining whether the problem is binary (only two classes) or multiclass (more than two classes) supports the selection of the most appropriate algorithm. While the logistic regression algorithm represents a simple solution for binary classification problems, neural network and decision tree algorithms represent appropriate solutions for multiclass classification problems. Decision trees are simple but also powerful algorithms. Following a path of a decision tree leads to a specific end node, whereas the input data is compared to trained weights at each decision nodes that comes up. While decision trees are relatively easy to read and need only little computing power, they are also limited in their performance as this concept often causes overfitting. In contrast, the idea of emulating the human brain to perform machine learning inspired the development of artificial neural network algorithms. Due to the strengths of artificial neural networks in data volume and speed, they are mostly used for real-time or near real-time systems. Typical application areas comprise high-frequency trading, credit applications, data centre management, robotics or medical monitoring. A detailed description of the operating principles of artificial neural networks is provided in Section 6.1. Examples of common regression algorithms are linear regression, nonlinear regression and gaussian process regression. Besides the estimation accuracy as main indicator to determine the performance of such a model, depending on its application this might also include the required computing power and possible computational speed. (Alpaydin, 2014; Bell, 2014; MathWorks, 2016; Paluszek and Thomas, 2016)

In order to classify input data into specific categories, also called classes, to solve a classification problem, the learning algorithm creates a particular hypothesis to model the actual boundaries of each class as closely as possible. The performance of the hypothesis in estimating the correct output based on new input data is called generalisation. However, the complexity of the hypothesis should match the complexity of the underlying data function to enable achieving the highest level of generalisation. The case of having a hypothesis that is more complex than this function is called overfitting. Overfitting is mainly caused by the occurrence of noisy data. Noise refers to any unwanted difference between the acquired data and the actual data of the real world. Typical reasons of its occurrence are random fluctuations or measurement errors. An overfitting model aims so hard to classify all data correctly that it results in modelling all of the noise too. In contrast, the case of having a less complex hypothesis is called underfitting. While one example of underfitting is modelling a third-order polynomial with a line, one example of overfitting is modelling a third-order polynomial, that seems to be a sixth-order polynomial due to noisy data, with a sixth-order polynomial. Both cases result in a poor level of generalisation. Figure 2-9 illustrates the three main modelling approaches including underfitting, overfitting and the optimal compromise. Especially overfitting is a huge problem in deep learning. (Alpaydin, 2014; Deshpande, 2017)

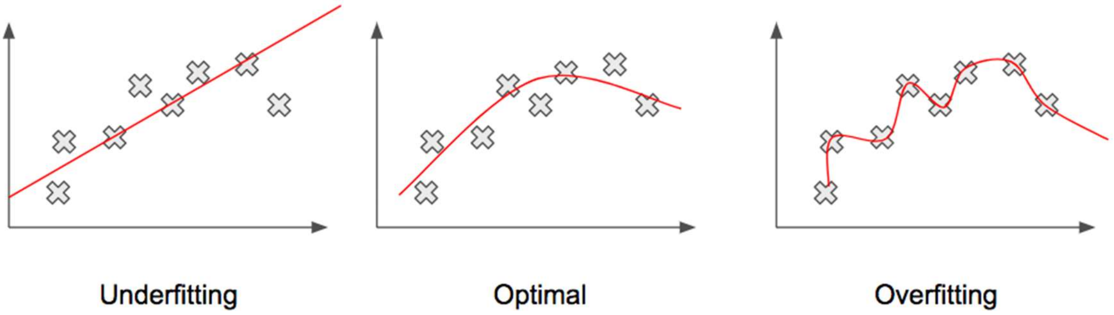
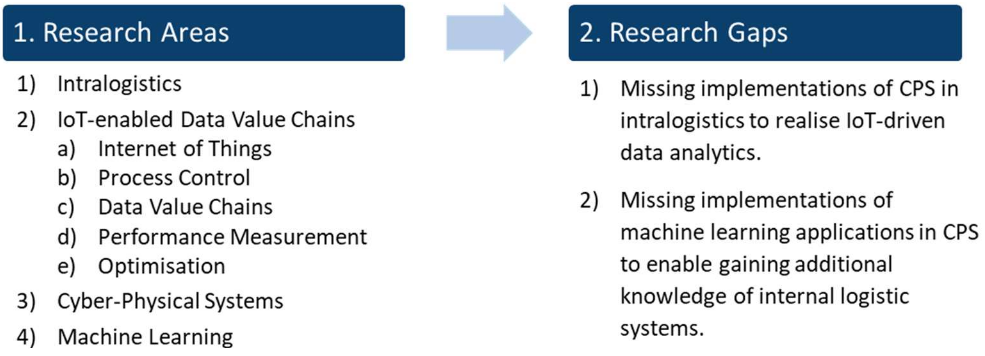


Figure 2-9: Modelling Approaches in Machine Leaning (adopted from Deshpande, 2017)

Nowadays, several available datasets support research projects in the field of machine learning through providing a large amount of labelled data. Such datasets typically comprise two types of data, the training data (also called training set) to fit the hypothesis through changing the parameters and the test data (also called test set) to determine the classification accuracy and evaluate the generalisation ability of the final hypothesis. However, splitting the available training data into two additional sets enables the performance of cross-validation. While the first part of the data represents a reduced training set, the second part represents a validation set. The validation set might also be used to evaluate changes of certain hyperparameters, such as the learning rate or number of hidden layers. Furthermore, Deshpande (2017) presents a simple way to detect overfitting models through comparing their errors with training and validation data. Having a significantly higher error with the validation data indicates the existence of overfitting. Besides that, having a consistently high error with the training and validation data indicates the existence of underfitting. The optimal compromise includes a decreasing error during the training and a consistently small error during the validation. (Alpaydin, 2014; Deshpande, 2017; MathWorks, 2016, Nielsen, 2017)

2.5 Research Gap

Although IoT has been used in intralogistics for years, the realisation of IoT-driven data analytics is a relatively new approach to enable the optimisation of internal logistic systems in terms of operational performance, uptime and sustainability. Therefore, the conducted literature review has provided a proper understanding of the required fundamentals. While IoT-enabled data value chains deal with the effective transformation of raw data into applied knowledge, the implementation of a CPS has been identified as an appropriate way to realise IoT-driven data analytics in intralogistics. This includes the data acquisition, the data processing and finishes with the visualisation of the relevant information as KPIs. However, the literature review has also identified areas of missing expertise as most of the IoT applications in intralogistics are currently focused on identification and sorting processes. Therefore, the realisation of IoT-driven data analytics through the implementation of a CPS in intralogistics requires further work. Besides that, the implementation of machine learning applications in a CPS represents an effective way to enable gaining additional knowledge of an internal logistic system and thus, to realise further optimisation. However, there is still a poor understanding how machine learning application can be implemented in such CPS. This resulted in the identification of two current research gaps, presented in Figure 2-10.



1. Research Areas

1) Intralogistics
2) IoT-enabled Data Value Chains
   a) Internet of Things
   b) Process Control
   c) Data Value Chains
   d) Performance Measurement
   e) Optimisation
3) Cyber-Physical Systems
4) Machine Learning

2. Research Gaps

1) Missing implementations of CPS in intralogistics to realise IoT-driven data analytics.
2) Missing implementations of machine learning applications in CPS to enable gaining additional knowledge of internal logistic systems.

Figure 2-10: Literature Review Methodology

The present study provides a contribution targeting to close the identified research gaps through the design and implementation of a CPS demonstrator for a use case in intralogistics to realise IoT-driven data analytics. Furthermore, a convolutional neural network that can be implemented in CPS to identify the operating conditions of a conveyor system is developed.

3 Basics to Develop a Cyber-Physical System in Intralogistics

Before starting the actual development of a CPS demonstrator for a use case in intralogistics, an additional focus on related basics aims to deepen the required understanding. This involves the identification of requirements of the CPS to realise IoT-driven data analytics as well as the selection of an appropriate communication protocol based on a conducted evaluation.

3.1 Requirements of the CPS

The understanding provided through the literature review indicates that the development of a CPS is a complex and interdisciplinary process, including mechanical engineering, electrical engineering and computer science. Therefore, the identification of the related requirements of a CPS are essential to successfully develop a system that provides relevant information about the operating conditions and performance of an internal logistic system. Furthermore, Wiesner et al. (2014) emphasise the importance that all stakeholders have a proper understanding of these requirements as a poor understanding might cause exceeding the budget, missing functionalities or even the stop of the project. (Jordan et al., 2017; Schuh et al., 2017; Wiesner et al., 2014)

Having understood the concept of CPS, investigating additional literature supported the identification of the requirements of the CPS to realise IoT-driven data analytics in intralogistics. All identified requirements have been classified in six groups. The first group focuses on the functional requirements to enable monitoring the operating conditions and performance of the internal logistic system. While the second, third and fourth group represent the requirements related to each tier of the presented three-tier architecture of CPS in Figure 2-6, the fifth group deals with the communication network. The last group focuses on the general system requirements. Finally, additional suggestion to optimise the CPS are provided.

Functional Requirements:

The functional requirements of the CPS follow the structure of IoT-enabled data value chains.

1. Acquiring data

   Acquiring the required data from the shop floor refers to the first step of the presented data value chain and enables the extraction of relevant information.
2. Determination of key performance indicators

   Computation tools are used to processes the stored data and calculate the values of suitable KPIs. This requirement refers to the second step of the presented data value chain.
3. Presentation of the relevant information

   An appropriate presentation of the relevant information is essential to apply analytical techniques properly and to complete the third step of the presented data value chain. This enables gaining new knowledge and thus, realising the optimisation of the internal logistic system. (Jordan et al., 2017; Marwedel et al., 2016; Schuh et al., 2017)

Edge Tier:

1. Sensors

   Sensors sense the current operating conditions and performance of the internal logistic system and provide this raw data to the edge node. Trueness of the data is ensured through an accurate and precise measurement. The appropriate types of sensors depend on the data to be acquired as well as on the physical process and environment. (Gilchrist ,2016; Jordan et al., 2017; Schuh et al., 2017)
2. Actuators

   Actuators perform actions to manipulate the internal logistic system. Therefore, the operating conditions are changed in accordance to commands set by human operators or autonomously by smart components. However, actuators are not required to provide relevant information. (Gilchrist ,2016; Schuh et al., 2017 Jordan et al., 2017)

3. Energy supply

   The I/O-devices that are integrated into the internal logistic system must be provided with the required energy. Mobile devices are used in case of limited energy availability. However, energy efficiency is important in all cases. (Marwedel et al., 2016)

4. Data processing at edge nodes

   Performing a first data processing at edge nodes enables reducing the amount of data that is transmitted to the platform and thus, results in an improved overall system efficiency. Typical edge nodes vary from single-board computers, programmable logical controls (PLC) to desktop computers. (Babovic et al., 2016; Gilchrist ,2016)

Platform Tier:

1. Data transformation

   The data received from several edge nodes must be converted in a common format to enable its subsequent storing and processing. This is a fundamental step to achieve full data integration. (Gilchrist ,2016)

2. Data storage

   The data as well as the extracted information must be stored. Due to the increasing volume, scalable databases have become preferred. One possible solution represents cloud computing. (Gilchrist ,2016; Tan et al., 2008)

3. Data processing

   Data needs to be processed to extract information. This is enabled through computation tools and might include simple mathematical equations or complex data mining and machine learning processes. (Gilchrist ,2016; Jordan et al., 2017; Schuh et al., 2017)

Enterprise Tier:

1. User interface

   In order to enable a proper application of analytics, the relevant information must be presented in an appropriate way. This is mostly performed visually (e.g. via a dashboard) and involves all supporting applications to analyse the presented information in detail. (Gilchrist ,2016; Jordan et al., 2017; Schuh et al., 2017)

2. Control

   According to the analytic results, actions must be taken (e.g. via actuators). This can be performed manually by human operators or autonomously by smart components. As these actions depend on the feedback of the system, a closed control loop is given. (Sanislav and Miclea, 2012; Tan et al., 2008)

Communication Network:

1. Timing

   The response time of the CPS must be within specific limits to enable presenting the relevant information close to real-time. Therefore, key requirements of the communication network are low latency and jitter. Furthermore, a high throuput, resilience and scalability are important factors for a successful system. (Ferrari et al., 2017; Gilchrist ,2016; Marwedel et al., 2016)

2. Appropriate communication technologies

   A successful data transmission across the entire communication network requires several technologies and communication protocols. Within the local network, link layer protocols such as ethernet, field bus, Wi-Fi and more are available. Especially in an IoT environment, the number of sensors working with Wi-Fi is increasing. An evaluation of appropriate application layer protocols for the CPS is provided in Section 3.2.2. The computer network is represented through the internet. (Gilchrist ,2016; Jordan et al., 2017; Marwedel et al., 2016; Schuh et al., 2017)

Overall System:

1. Cyber-security

   The fear that other parties could have access to confidential data and information is one of the biggest challenges in implementing a CPS. An appropriate cyber-security system in essential as the connection of all involved objects and platforms through the internet makes the CPS vulnerable. (Gilchrist ,2016; Hribernik et al., 2012)

2. Safety

   The virtual control of internal logistic systems also represents potential danger through any malfunction. However, frequent tests decrease safety risks. (Marwedel et al., 2016)

3. Reliability

   Reliability is the probability that the expected service is provided. In order to ensure having a reliable CPS, each component and involved sub-system must be reliable. (Gilchrist ,2016; Marwedel et al., 2016)

4. Flexibility

   In order to easily adapt the CPS to different internal logistic systems, it is important that small changes, such as adding and removing sensors or actuators, can be performed quickly. (Marwedel et al., 2016)

5. Compatibility

   A proper interaction between all different components that are involved in the CPS must be ensured. Challenges arise due to the several types of components, such as sensors, actuators, control units, platforms, and more. Standardisation is essential to achieve a high level of compatibility. (Marwedel et al., 2016)

Optimisation of the CPS:

1. Context awareness

   The incorporation of context awareness improves the effectiveness of the data value chain and thus, enables gaining more knowledge. This allows an improved optimisation of the internal logistic system. (Perera et al., 2014; Scholze and Barata, 2016)

2. Data reduction

   A reduced data volume results in less cost as less resources (e.g. data storage) are required. Identifying the actual data that add value to the optimisation process is essential to achieve the best data reduction. The publish and subscribe pattern is a commonly used method in which only the needed data is provided to upper levels. Furthermore, an event-triggered CPS enables measuring only snapshots of the operating conditions and performance of the internal logistic system. However, Colombo et al. (2017) argues that a mixed time/event-triggered system offers the best benefits. (Colombo et al., 2017; Gilchrist ,2016; Lee et al., 2015; Tan et al., 2008)

3.2 Evaluation of Communication Protocols for a CPS

3.2.1 Communication Protocols for a CPS

A successful communication between all devices involved in a CPS requires standardisation in terms of defined rules for data exchange including syntax, semantics, and synchronisation. This is realised through communication protocols. The conceptual models of communication protocols are structured hierarchically, whereas each layer is responsible for specific tasks. The two main models are the internet protocol suite (TCP/IP suite) and the Open Systems Interconnection model (OSI model). The TCP/IP suite builds the basis for the internet and provides end-to-end data communication, specifying how data should be packetized, addressed, transmitted, routed, and received. Its structure comprises four abstraction layers, whereas several protocols can be employed to perform the tasks involved at each layer: (Fall and Stevens, 2012; Cruz-Piris et al., 2018)

1. Link layer (e.g. Ethernet, field bus, Wi-Fi, etc.)
2. Internet layer (e.g. IP, ICMP, IGMP, etc.)
3. Transport layer (e.g. TCP, UDP, SCTP, etc.)
4. Application layer (e.g. MQTT, HTTP, CoAP, etc.)

Application layer protocols build the top layer and are visible to users the most. While the first three layers focus on the data transport across the network without being concerned about the involved applications, the application layer deals with these details. Currently, various application layer protocols are used in an IoT environment as each system has different requirements. This results in the lack of a common agreement on one standard protocol. In order to choose the optimal protocol, the actual situation in the IoT environment including its specific requirements must be taken into account. Depending on that situation, more than one protocol can be needed to perform the communication in the best possible way. The two main interaction models are request-response, mostly used in client-server architectures, and publish-subscribe. Even though the application layer protocols are mostly based on TCP, also other underlying transport layer protocols such as UDP are used. The currently most common application layer protocols in an IoT environment are shown in Table 3-1. Furthermore, information about the used interaction model, the underlying transport layer protocol and whether the protocol is considered as lightweight are provided. (Babovic et al., 2016; Cruz-Piris et al., 2018; Dizdarevic et al., 2018; Năstase et al., 2017)

Table 3-1: Application Layer Communication Protocols in IoT

Protocols	Interaction Model	Underlying Transport Layer Protocol	Lightweight
MQTT	Publish-Subscribe	TCP	Yes
HTTP	Request-Response	TCP	No
CoAP	Request-Response	UDP	Yes
XMPP	Request-Response and Publish-Subscribe	TCP	No
AMQP	Publish-Subscribe	TCP	No
DDS	Publish-Subscribe	TCP/UDP	No

Having understood the requirements of a CPS in intralogistics as well as investigated additional literature associated with communication protocols in IoT, the following criteria to identify the most appropriate application layer protocol for IoT in intralogistics were developed:

1. Efficiency (ratio between the useful bytes and the total number of bytes exchanged.)
2. Latency (measured in round-trip time: the time a packet needs to travel from the source to the destination)
3. Throughput
4. Energy consumption
5. Interoperability
6. Reliability

In order to evaluate the six presented application layer protocols in these six developed criteria, the results of conducted experiments from past research projects were used. Năstase et al. (2017) conducted an experiment to test, evaluate and finally compare several protocols in an IoT environment regarding the efficiency and round-trip time. Babovic et al. (2016) developed two web applications to evaluate latencies and throughput rates of sensor data in an IoT environment. Dizdarevic et al. (2018) investigated several communication protocols for fog and cloud computing and evaluated them in latency, security, energy consumption and interoperability though a literature review.

3.2.2 Evaluation of Application Layer Protocols for a CPS

Among the six application layer protocols that have been identified as the most common ones in an IoT environment, Năstase et al. (2017) have identified XMPP as the one with the highest efficiency, followed by MQTT and CoAP with a similar performance. Furthermore, MQTT and XMPP are supposed to provide by far the minimum round-trip time. However, Babovic et al. (2016) argue that the minimum latency is only provided by MQTT. In terms of the message throughput rate, MQTT enables best to medium performances, depending on the client implementations (Java or JavaScript) and the used broker. The result of the investigation of Dizdarevic et al. (2018) presents MQTT and CoAP as the protocols that provide the lowest latency in constrained IoT environments, whereas depending on the quality of service level and the used MQTT broker, MQTT or CoAP performs better. Even though this investigation has identified CoAP as the most energy efficient protocol, Cruz-Piris et al. (2018) claims that the energy consumption of MQTT is also relatively low due to its simplicity. According to Dizdarevic et al. (2018) and Năstase et al. (2017), one of the biggest advantages of CoAP is its interoperability with HTTP, as HTTP is the fundamental protocol for the World Wide Web. However, both also emphasise the importance of having TCP as the underlying transport layer protocol to realise reliable data delivery and point out that using UDP causes unreliability. (Babovic et al., 2016; Cruz-Piris et al., 2018; Dizdarevic et al., 2018; Năstase et al., 2017)

Moreover, the additional information about the associated interaction models and underlying transport layer protocols enabled a more detailed evaluation. A publish-subscribe model provides several advantages for an CPS, such as scalability as well as simple but dynamic many-to-many interconnections. As MQTT distinguishes between topics and sends separate messages for each, a relatively high number of packets is used. However, this results in a relatively low number of bytes per packet, which is an advantage in an IoT environment. Furthermore, as many of the involved devices in an IoT environment are constrained in memory and computational capabilities, it is important that application layer protocols are as lightweight as possible. Among the identified protocols, the only two lightweight ones are MQTT and CoAP, whereas MQTT has an advantage as it is based on TCP. Another advantage of MQTT over CoAP is its maturity. Finally, all investigated research projects see MQTT as the most appropriate application layer protocol in a general IoT environment as it satisfies most of the requirements. However, Dizdarevic et al. (2018) mention the potential of CoAP in IoT as it is still evolving and will reach a higher level of stability and maturity in the near future. (Babovic et al., 2016; Cruz-Piris et al., 2018; Dizdarevic et al., 2018; Năstase et al., 2017)

The evaluation has pointed out that MQTT satisfies most of the criteria for applications in an IoT environment. Furthermore, additional advantages are given as MQTT represents a lightweight publish-subscribe protocol, based on TCP. Therefore, MQTT has been identified as the most appropriate application layer protocol for a use case in intralogistics.

3.2.3 MQTT (Message Queuing Telemetry Transport)

This section is supposed to provide a brief overview of the application layer protocol MQTT. As mentioned in the previous section, MQTT represents a lightweight publish-subscribe protocol with bidirectional capabilities. It works on top of TCP and was developed for networks that involve constrained devices. Furthermore, the MQTT protocol has recently become an international standard (ISO/IEC PRF 20922). Due to its advantages including low energy consumption, small overhead (low bandwidth) and reliability, MQTT is the mainly used application layer protocol in IoT environments at the moment. Because MQTT uses a publish-subscribe model, one server and one or more clients are required. A general concept of the publish-subscribe model is illustrated in Figure 3-1. Therefore, one or more clients publish messages (publishers), whereas each message is allocated to a certain topic. In contrast, the same or other clients receive messages (subscribers) that correspond with the topics they have subscribed for. This is coordinated by a central server (broker), that receives the messages from publishers, processes them and retransmits the right messages to the right subscribers according to the topics. Topics are labels attached to each application message. They are represented as strings with a hierarchical structure, whereas each topic can have multiple subtopics to filter messages. As clients do not know each other, this model allows high scalability without dependencies between publisher and subscriber. In order to enable the broker pushing new messages to the corresponding subscriber, a permanently open TCP connection between subscriber and broker is required. Therefore, the broker can detect when a client disconnects unintentionally, which can be used for failure detection. (Babovic et al., 2016; Cruz-Piris et al., 2018; Deschambault et al., 2017; Dizdarevic et al., 2018; HiveMQ, 2015; Năstase et al., 2017; Oasis, 2014)

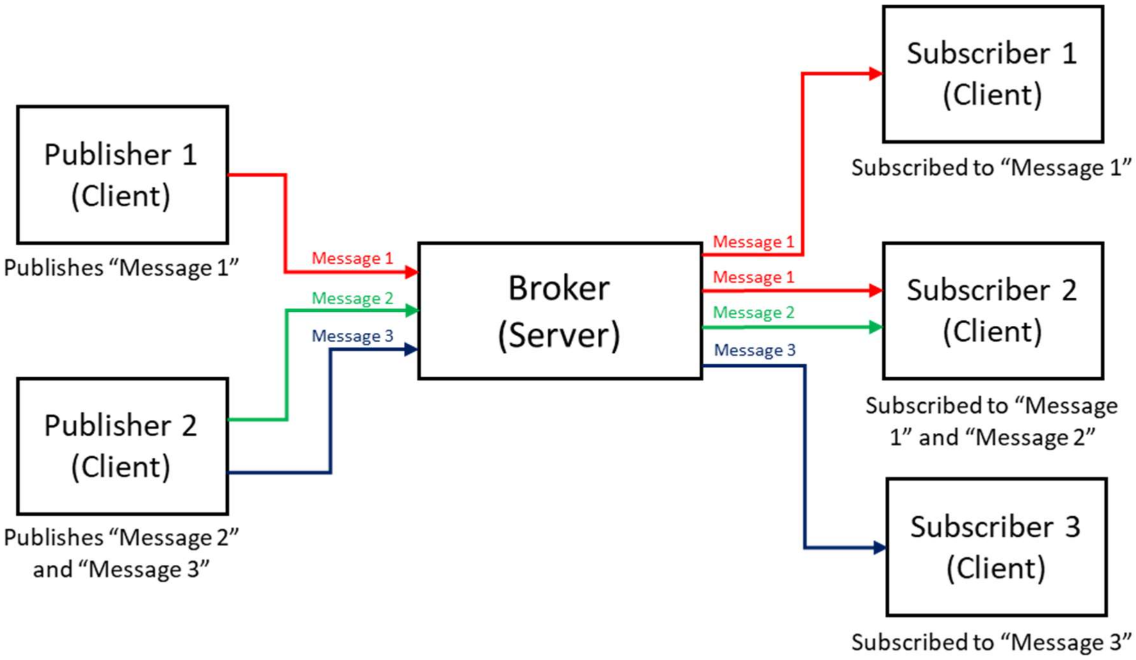


Figure 3-1: Concept of a Publish-Subscribe Model

The performance of MQTT strongly depends on the used broker. A MQTT broker library must be installed on the device that serves as broker. Examples for open source MQTT broker are Mosquitto, Emqttd, ActiveMQ, HiveMQ, IBM MessageSight, JoramMQ, RabbitMQ and VerneMQ. Furthermore, MQTT client libraries, such as Paho Labrary, must be installed on the client deceives. Furthermore, MQTT provides three QoS (Quality of Service) levels for the published messages, QoS 0, 1 and 2. Using a higher QoS level results in more complexity and thus, the need of more resources and overheads. As this increases the latency, it is essential to choose the most appropriate level:

1. QoS 0: A message arrives at most once.
2. QoS 1: A message arrives at least once.
3. QoS 2: A message arrives exactly once.

The transmitted message represents a digital packet of data that consists of bytes, associated with a QoS and a topic. Encryption of messages is not provided by MQTT to ensure it is lightweight. As data is exchanged as plain-text, MQTT only provides simple security features. In an extended network, a poor MQTT application design might easily allow harmful messages injections into the network. For these reasons, security must be implemented on top of MQTT. However, this is not necessary in isolated networks. On possible solution is the additional implementation of encryption, for example via TLS. However, this results in increased overheads. Many MQTT brokers provide simple authentication through requesting username/password combinations. An unsuccessful authentication results in refusing to send the message. Even though certain brokers accept anonymous clients, it is not desirable in most of the cases. Additional security mechanisms are usually based on TLS or DTLS. While both start with a handshaking process between client and server, the additional traffic increases the overheads by 6.5% for TLS and 11% for DTLS on average. TLS is preferred as it is widely used and a stable security protocol. However, it was originally not developed for an IoT environment with constrained devices. Therefore, the given limitations in an IoT environment result in the need of finding alternative solutions. (Babovic et al., 2016; Cruz-Piris et al., 2018; Deschambault et al., 2017; Dizdarevic et al., 2018; HiveMQ, 2015; Năstase et al., 2017; Oasis, 2014)

4 Development of a CPS Demonstrator for a Use Case in Intralogistics

This chapter describes the performed steps to develop a demonstrator of a cyber-physical system (CPS) that determines and visualises KPIs of a use case in intralogistics. Even though this demonstrator only represents one possible solution to realise IoT-driven data analytics in intralogistics, it is indicative of other possible solutions. As the literature review has pointed out, IoT-driven data analytics represents a relatively new approach to enable the optimisation of internal logistic systems in terms of operational performance, uptime and sustainability.

4.1 Use Case in Intralogistics

The first step to develop a demonstrator of a CPS in intralogistics was the definition of a representative use case. As the literature review has identified that having an optimal transport system is a key to success in intralogistics, this study focuses on an existing conveyor system at Graz University of Technology (Institute of Logistics Engineering). Because it comprises several conveyor types, the system successfully represents a use case in intralogistics.

4.1.1 Conveyor System

This section provides an overview of the existing conveyor system at Graz University of Technology (Institute of Logistics Engineering). It consists of nine different conveyor types as shown in Figure 4-1 and Table 4-1. While Figure 4-1 visualises the positions of each type and the overall layout, the table contains information about the mechanical drive, the lengths and the speed of each element. Furthermore, Figure 4-1 also shows the two paths, circle 1 (27.97 m) and circle 2 (29.90 m), that the parcels can follow. In addition to that, Figure 4-2 provides a photo of the conveyor system in the real world. The entire system is controlled by one PLC (Siemens SIMATIC S7-300), whereas a connection with the local computer enables changing several parameters via the software “Simatic-Manager”.

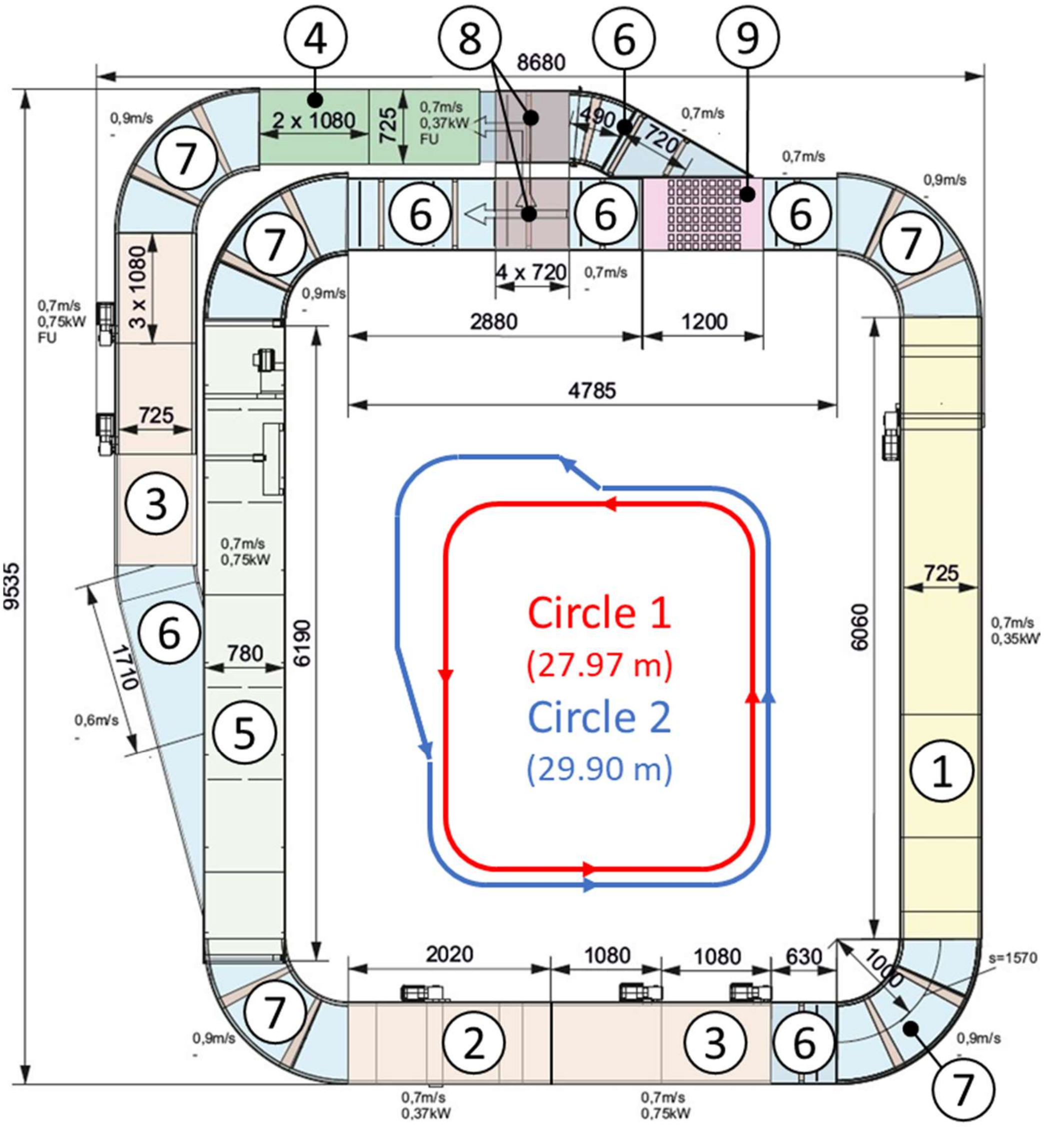


Figure 4-1: Layout of the Conveyor System

Table 4-1: Involved Conveyors Types

Pos.	Type	Mechanical Drive	Length	Speed
1	Belt conveyor (roller bed)	External belt drive	1 x 6.06m	0.7m/s
2	Belt conveyor (slider bed)	External belt drive	1 x 2.02m	0.7m/s
3	Belt conveyor (slider bed)	Direct drive at return pulley	5 x 1.08m	0.7m/s
4	Link belt conveyor	Direct drive at return pulley	2 x 1.08m	0.7m/s
5	Roller conveyor	Central belt drive	1 x 6.19m	0.7m/s
6	Roller conveyor	Single roller drive, connected via belts	1 x 1.44m, 2 x 0.72m, 1 x 0.63m, 1 x 1.71m	0.7m/s, 0.7m/s, 0.7m/s 0.6m/s
7	Roller conveyor (curved)	Single roller drive, connected via belts	5 x 1.57m	0.9m/s
8	Belt diverter	External drive	2 x 0.72m	0.7m/s
9	Roller Switch	External drive	1 x 1.20m	0.7m/s

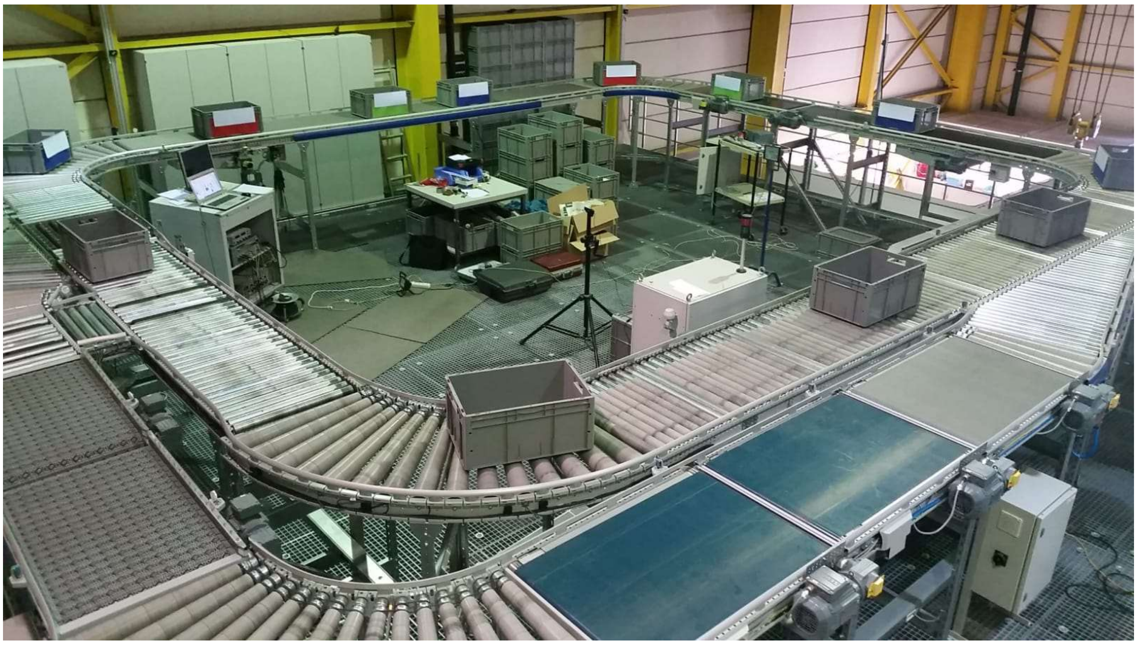


Figure 4-2: Real World Conveyor System

Furthermore, the conveyor system enables the performance of several functions through manual interventions. Therefore, the PLC provides two different operating modes. The first mode, “controlled running”, is set by default and involves multiple functionalities as shown in Table 4-2. The second mode, “all running”, only represents a test case in which all elements are running.

Table 4-2: Functionalities of the Conveyor System via Controlled Running

No.	Functionality
Via Switch 1	
1	Only circle 1 is used
2	Only circle 2 is used
3	Circle 1 and circle 2 are used alternately (change after every single unit)
Via Switch 2	
4	Only use the belt diverter (pos. 8) to move units from circle 1 to circle 2
5	Only use the roller switch (pos. 9) to move units from circle 1 to circle 2
Via Simatic-Manager and PLC	
6	Changing the distance between units

4.1.2 Context Awareness

As the literature review has pointed out, the incorporation of the contextual situation under which the physical system is operating improves the management and processing of relevant data. This enables the identification of the important characteristics of a specific situation. Analysing the contextual information results in a better understanding of the physical system and thus, in gaining more knowledge effectively. Therefore, this section is focused on context awareness in intralogistics. Figure 4-3 visualises a high-level context model for a general situation in intralogistics. The model consists of five categories, whereas each category contains several characteristics. The used method to identify the categories and characteristics followed the rule of the five Ws (Who, What, Where, When and Why), presented by Perera et al. (2014). The basis of the model builds the order, which is either internal or external. Every order involves a customer who desires the transportation of physical goods from the start point to the end point within a specific time. Furthermore, each order has a specific priority. The related physical objects can differ in several areas such as the type, cost, quantity, physical properties, content and also priority. In order to perform the transportation of these goods effectively, a transport system is essential. This can consist of different conveyor elements, whereas information about the layout, capacity and performance might be used for a detailed evaluation. However, human operators in different roles are still needed for some tasks such as commissioning. Combining the information about the investment and operational cost of the transport system with the labour cost allows detailed cost estimations. Besides that, the environment supports and controls the entire process as well as provides the necessary safety. Finally, the service represents the overall task that is performed in a specific quality.

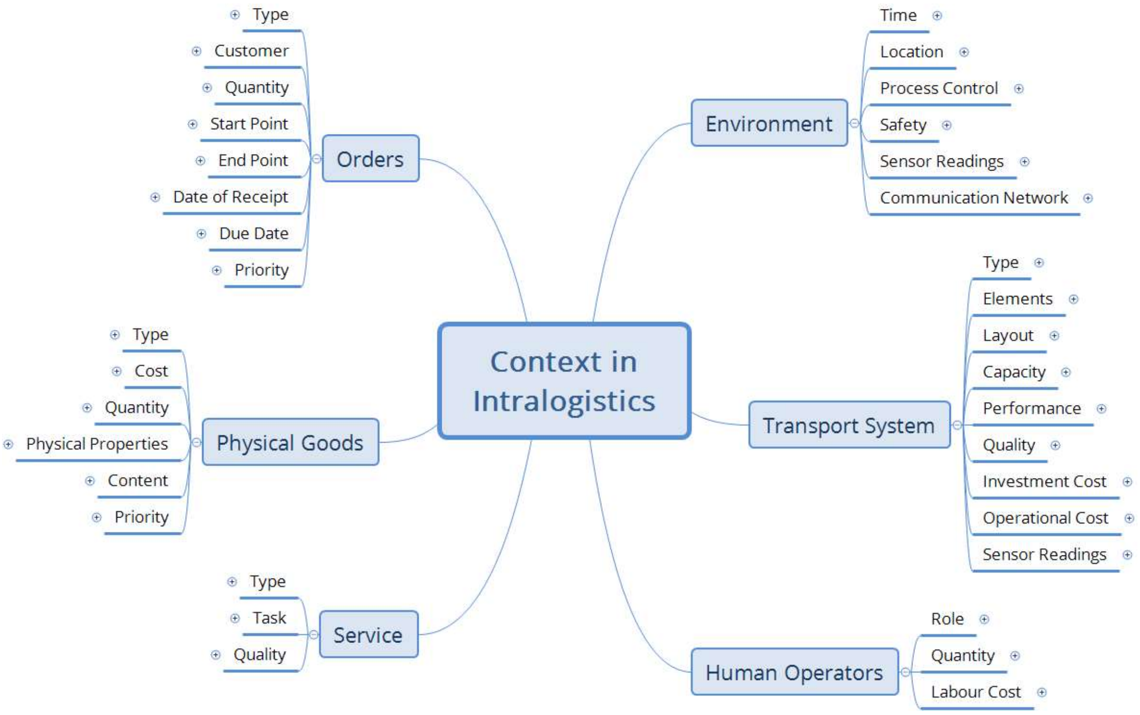


Figure 4-3: General Context Model in Intralogistics

Having developed a general context model for intralogistics, it was applied on the defined use case. This resulted in a low-level sample context model, whereas the category “orders” is visualised in Figure 4-4. While some of the information in the model matches with the use case, the rest is made up as the required information was unavailable. However, using this information to identify important characteristics of the current situation enables an improved understanding of the internal logistic system. The overall context model of the use case as well as all categories are provided in detail in Appendix A.

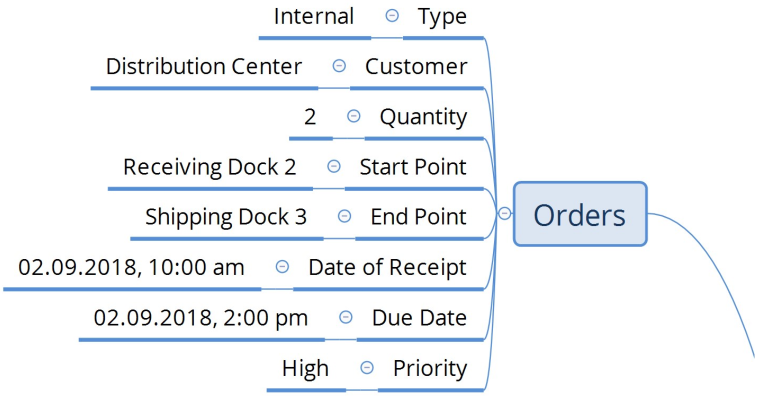


Figure 4-4: Intralogistics Context Model applied on the defined Use Case – Orders

4.2 Data Acquisition

Having defined a representative use case in intralogistics, the next step was the development of a system that acquires the required data identified through the literature review.

4.2.1 Hardware Design

The structure of the data acquisition system is illustrated in Figure 4-5, whereas its aim is to acquire as much of the required data as possible. The existing conveyor system of the defined use case represents the physical object, the basis of this data acquisition. Furthermore, one or more edge nodes and sensors are needed. One possible edge node represents the PLC. However, to demonstrate the diversity of suitable devices, a Raspberry Pi 3 Model B is used for this demonstrator. Moreover, a raspberry pi represents a relatively low-cost alternative. In addition to that, a webcam serves as the sensor and acquires data in form of a video. The captured video is then processed through a Simulink model (Section 4.2.2) which either runs on a connected laptop or on the raspberry pi itself. This model detects and counts moving as well as red, green or blue parcels. As only numbers are sent to the data storage at ThingSpeak, a clould-based platform provided by MathWorks, the volume of the transmitted data is reduced significantly. This results in an improved overall system efficiency. Furthermore, the laptop provides a user interface (Section 4.2.2) to manually enter additional data that cannot be extracted from the video as well as to start and stop running the Simulink model. The user interface also calculates the snapshot time (the time between sending new data to ThingSpeak) via the manually entered data. Finally, this system represents a simple and cost-effective solution to realise the required data acquisition.

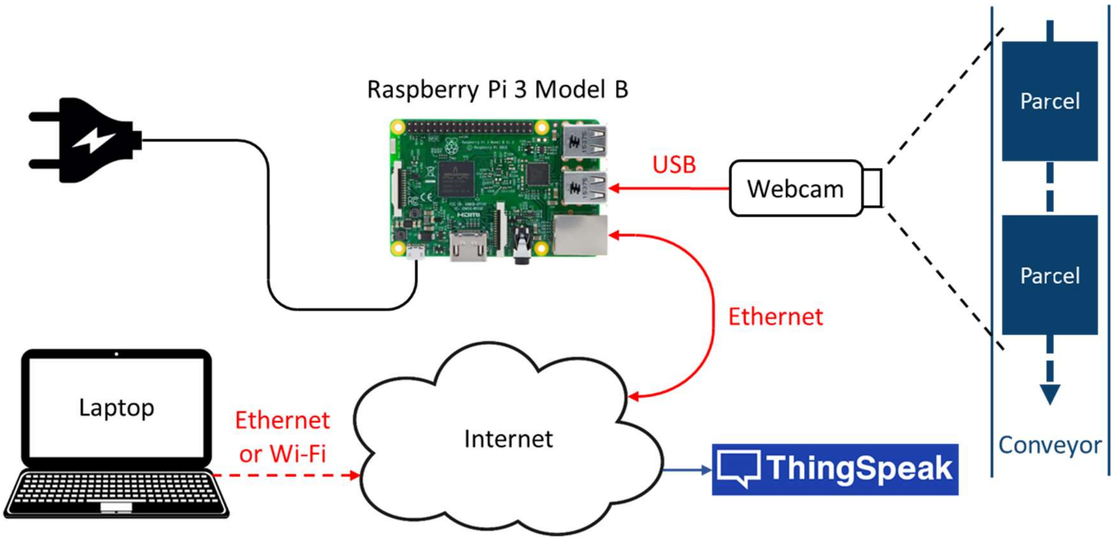


Figure 4-5: Structure of the Data Acquisition System

Even though a Raspberry Pi 3 Model B (1 GB SDRAM, 1.2 GHz CPU frequency) is part of the final data acquisition structure, first tests were carried out with a Raspberry Pi Model B (512 MB SDRAM, 700 MHz CPU frequency). These tests have pointed out that the Raspberry Pi 3 Model B provides several benefits such as a smoother real time visualisation of detected moving or coloured parcels via the video displays including the possibility to use a higher resolution, the possibility to use two video displays simultaneously (instead of using one at a time) as well as a more constant time between sending new data to ThingSpeak. The most likely reason for these benefits are the higher memory and CPU frequency provided by the Raspberry Pi 3 Model B. A more detailed description is available in Section 4.4.1.

In order to perform the detection of coloured parcels, the originally grey containers have been wrapped with red, green and blue paper as shown in Figure 4-6. However, a strong influence of the light condition on the performance of the detection of coloured and moving objects has been identified. While the influence on the detection of coloured parcels was easily compensated by adjusting the threshold values for the colours in the Simulink model, the detection of moving parcels needed a different solution. As several tests have pointed out that the performance of detecting white moving objects is generally the best due to the improved light reflection capabilities, all containers have been additionally equipped with white paper. Furthermore, following tests have shown that the additional and partly overlapping white paper does not affect the performance of the detection of coloured parcels. A more detailed description is also available in Section 4.4.1.

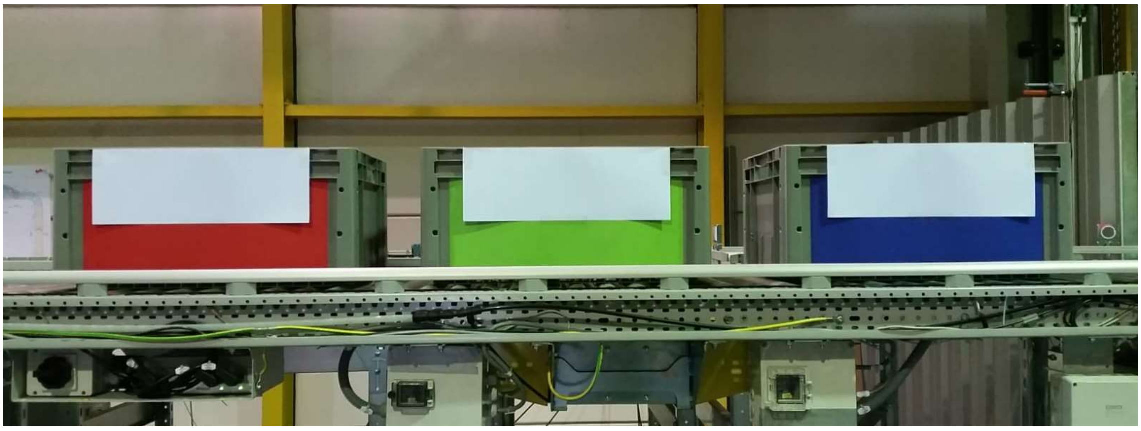
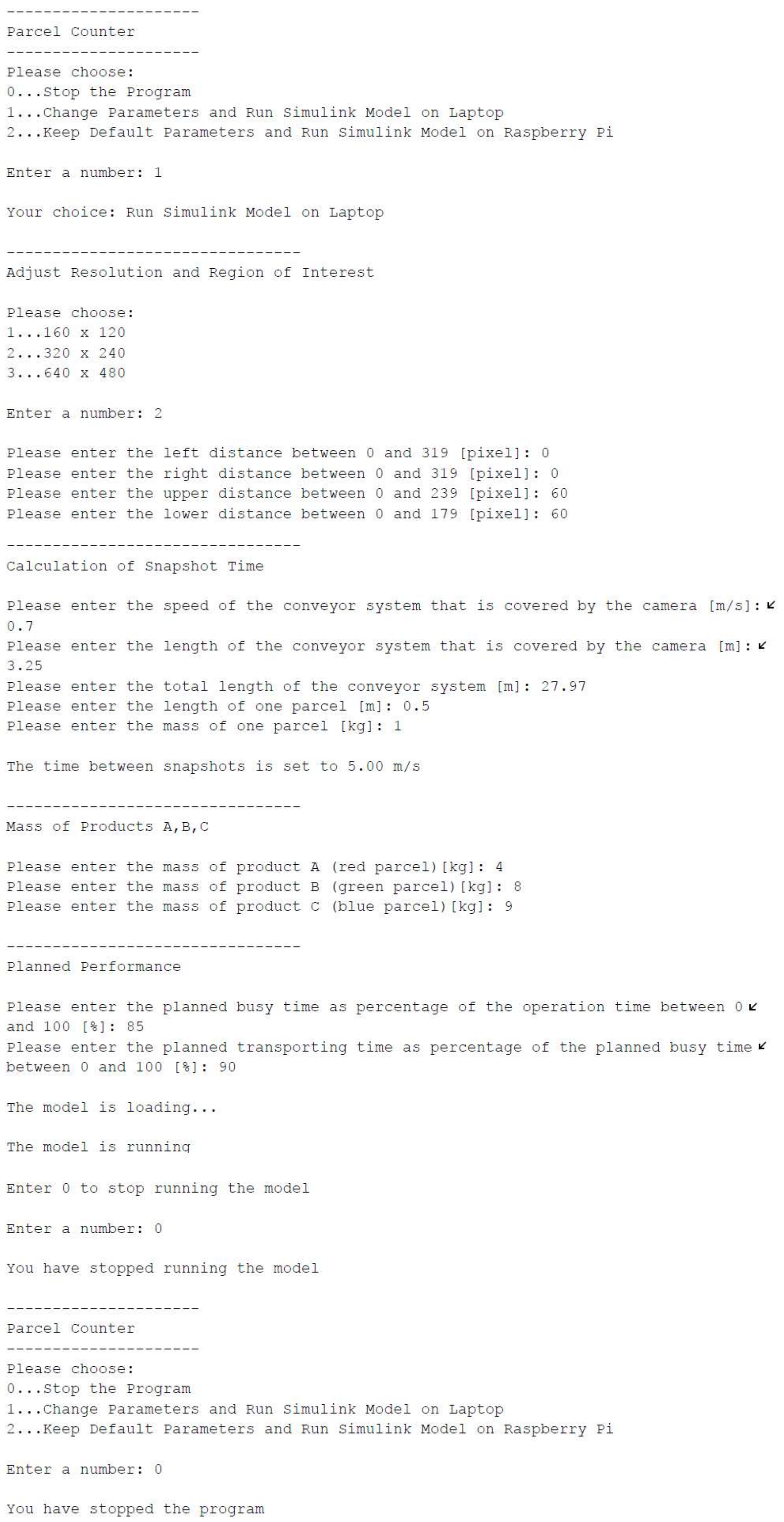


Figure 4-6: Parcels equipped with red, green, blue and white Paper

4.2.2 Software Design

Figure 4-7 illustrates a sample sequence of events when controlling the Simulink model through the user interface. At the beginning, the main menu enables the user to choose between stopping the entire program, running the model on the laptop or running it on the raspberry pi. While the second option only allows using default parameters, the first one enables entering several parameters manually. This starts with adjusting the resolution and region of interest and is followed by entering the values of several parameters that are required for the estimation of some KPIs. All these values are sent to the related ThingSpeak channels when the model is running. Furthermore, the first parameters are also used to calculate the snapshot time. In the next step, the program starts loading the Simulink model including the previously entered values of the parameters, whereas that process might take about one to two minutes. Having loaded the model, it starts running immediately. This can easily be stopped via the user interface and leads back to the main menu. The related MATLAB code is provided in Appendix B.



```
---------------------
Parcel Counter
---------------------
Please choose:
0...Stop the Program
1...Change Parameters and Run Simulink Model on Laptop
2...Keep Default Parameters and Run Simulink Model on Raspberry Pi

Enter a number: 1

Your choice: Run Simulink Model on Laptop

--------------------------------
Adjust Resolution and Region of Interest

Please choose:
1...160 x 120
2...320 x 240
3...640 x 480

Enter a number: 2

Please enter the left distance between 0 and 319 [pixel]: 0
Please enter the right distance between 0 and 319 [pixel]: 0
Please enter the upper distance between 0 and 239 [pixel]: 60
Please enter the lower distance between 0 and 179 [pixel]: 60

--------------------------------
Calculation of Snapshot Time

Please enter the speed of the conveyor system that is covered by the camera [m/s]:↙
0.7
Please enter the length of the conveyor system that is covered by the camera [m]:↙
3.25
Please enter the total length of the conveyor system [m]: 27.97
Please enter the length of one parcel [m]: 0.5
Please enter the mass of one parcel [kg]: 1

The time between snapshots is set to 5.00 m/s

--------------------------------
Mass of Products A,B,C

Please enter the mass of product A (red parcel)[kg]: 4
Please enter the mass of product B (green parcel)[kg]: 8
Please enter the mass of product C (blue parcel)[kg]: 9

--------------------------------
Planned Performance

Please enter the planned busy time as percentage of the operation time between 0↙
and 100 [%]: 85
Please enter the planned transporting time as percentage of the planned busy time↙
between 0 and 100 [%]: 90

The model is loading...

The model is running

Enter 0 to stop running the model

Enter a number: 0

You have stopped running the model

---------------------
Parcel Counter
---------------------
Please choose:
0...Stop the Program
1...Change Parameters and Run Simulink Model on Laptop
2...Keep Default Parameters and Run Simulink Model on Raspberry Pi

Enter a number: 0

You have stopped the program
```

Figure 4-7: Sample of the User Interface via MATLAB

The calculation of the snapshot time is based on Figure 4-8. While $l_{Snapshot}$ represents the length of the conveyor system that is covered by the camera, no detection is performed at its beginning and end. This is caused as the filmed area of the coloured paper is still too small to be detected.

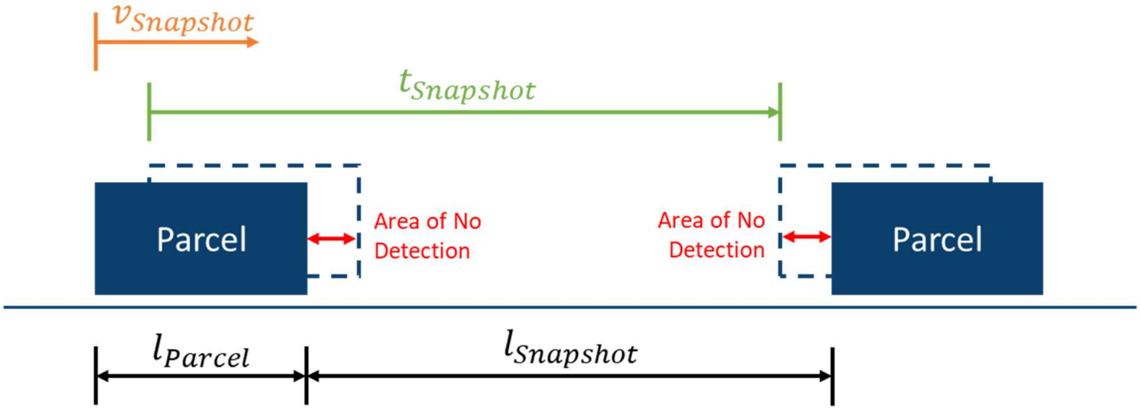


Figure 4-8: Calculation of Snapshot Time

The formula used to calculate the snapshot time is shown in Equation 4-1.

$$t_{Snapshot} = \frac{(l_{Snapshot} + l_{Parcel} - 2 * \ 0.25 * l_{Parcel})}{v_{Snapshot}} \qquad \textbf{(4-1)}$$

In the above formula, the area of no detection is considered by subtracting 2 x 0.25 x l_{Parcel} from $l_{Snapshot}+l_{Parcel}$. Therefore, the snapshot time theoretically corresponds to the time one parcel needs to cross the area in which the detection of the coloured paper is performed properly and thus, theoretically enables counting each parcel once.

The used Simulink model is illustrated in Figure 4-9. In order to provide a simple description, it is organised into six areas. Starting with the red area, the first step is capturing the video as simple images in the RGB format, whereas the values for the desired resolution and the time between receiving a new image are provided by the user interface. This is followed by processing the image in a subsystem that includes four outputs. The first output goes to the blue area, where a ForegroundDetector determines whether individual pixels are part of the foreground (moving) or the background (unmoving). Additionally, a Median Filter is used to reduce errors. The following Blob Analysis unit marks and counts moving objects. In the purple area, three other Blob Analysis units use the remaining three outputs of the previous subsystem to mark and count red, green and blue objects. Subsequently, the yellow part comprises the visualisation of the marking and counting results through another subsystem that contains two video displays. However, the value of the debug_flag, provided via the user interface, enables or disables this visualisation. Disabling the visualisation makes sense in cases the Simulink model is not running on the laptop but on the raspberry pi. The blue area represents the reading of all values of the parameters required to estimate additional KPIs. Finally, the orange area includes sending the necessary data to two ThingSpeak channels (group A). Two channels are necessary as one can only receive the values of eight different parameters. The time between sending new data to the channels is also given by the user interface through the snapshot time. While ThingSpeak defines the minimum time between receiving new data at the same channel by 15 seconds, purchasing a Student licence enabled decreasing this time to only one second. A bigger figure of the overall model as well as the content of all subsystems are provided in Appendix C.

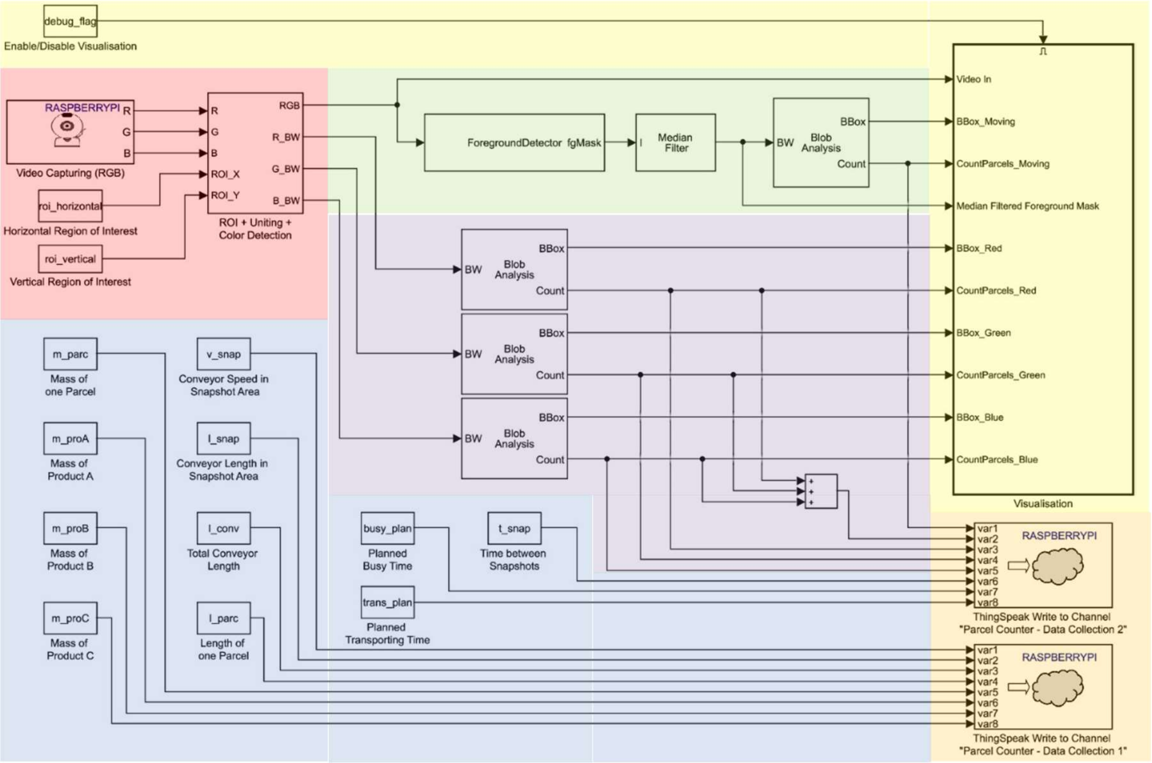


Figure 4-9: Simulink Model of the Parcel Counter

The following figures provide a sample of the visualisation of the marking and counting results via the video displays. As illustrated in Figure 4-10, the first video display shows the moving containers marked with a green rectangle. The second video display shows the red, green and blue containers marked with a rectangle in the corresponding colour. Both displays provide the results of the counts in real time.

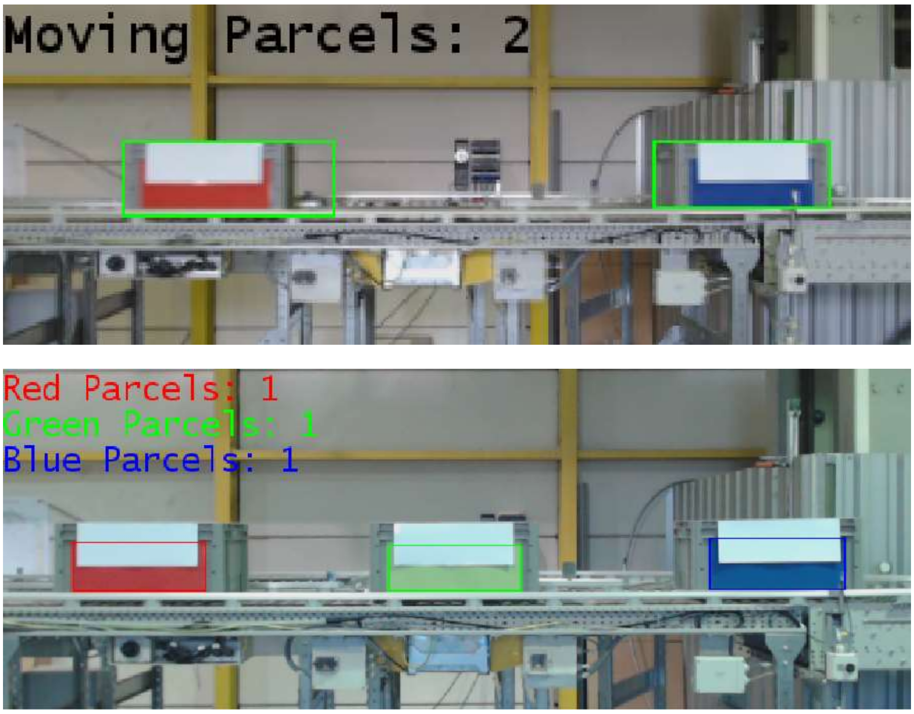


Figure 4-10: Video Displays of the Simulink Model

4.2.3 Data Storage

All data sent from the Simulink model are received and stored at two ThingSpeak channels (group A). Table 4-3 provides an overview of this data set, consisting of sixteen parameters. While the values of the first eleven parameters are provided by the user interface, only the first ten (orange) have to be entered manually. The value of the eleventh parameter (green), the snapshot time, is calculated by the user interface. The values of the last five parameters (blue) are obtained through processing the captured video via the Simulink model.

Table 4-3: Data sent from the Simulink Model

No.	Data
1	Speed of the conveyor system in the snapshot area [m/s]
2	Length of the conveyor system in the snapshot area [m]
3	Total length of the conveyor system [m]
4	Length of one parcel [m]
5	Mass of one parcel [kg]
6	Mass of product A [kg]
7	Mass of product B [kg]
8	Mass of product C [kg]
9	Planned busy time as percentage of the operation time [%]
10	Planned transporting time as percentage of the planned busy time [%]
11	Snapshot time [s]
12	Count of moving parcels [units]
13	Count of coloured parcels [units]
14	Count of red parcels [units]
15	Count of green parcels [units]
16	Count of blue parcels [units]

4.3 Data Processing and Information Visualisation

In order to complete the demonstrator of a CPS, the system for the data processing to extract information as KPIs had to be designed. This system also includes the visualisation of these KPIs as well as of additional information and thus, enables monitoring the operating conditions and performance of the conveyor system.

4.3.1 Data Processing

An overview of the assumptions made to estimate the KPIs is shown in Table 4-4.

Table 4-4: Assumptions for the Data Processing

No.	Assumption
1	The provided values of the KPIs are representative for the entire conveyor system. Possible errors occurring as the values are actually calculated for the snapshot area are not considered.
2	The assumed speed of the entire conveyor system corresponds to the speed of the conveyor system in the snapshot area.
3	The entire conveyor system transports all parcels with constant speed and no stops.
4	The minimum distance between the parcels is 0.25 times the length of one parcel.
5	The operation time corresponds to the time the Simulink model is running.
6	The actual busy time corresponds to the time the Simulink model is running minus the time the conveyor system is not running.
7	The actual transporting time corresponds to the time moving parcels are detected.
8	The condition of the conveyor system is either “running”, “stop” or “unknown”.
9	The provided values of the time utilisation, effectivity, availability and OEE correspond to the average of the last 60 min.

The data processing is performed through two MATLAB programs created with the Analysis app in the ThingSpeak environment, whereas both programs have access to the previously stored data. Table 4-5 provides an overview of all suitable KPIs identified through the literature review, whereas the available data is only sufficient to estimate the yellow marked KPIs. While the values of the first four KPIs are calculated via program one, the remaining yellow ones are calculated via program two. This segregation makes sense as the time utilisation, effectiveness, availability and the overall equipment effectiveness are related to a specific period of time (the demonstrator uses the last 60 minutes) and thus, require different calculation processes. Furthermore, program one detects stops of the conveyor system and sends each detection to an auxiliary ThingSpeak channel. These detected stops are used by program two to calculate the time utilisation. The related MATLAB codes are provided in Appendix D.

Table 4-5: Calculated Key Performance Indicators

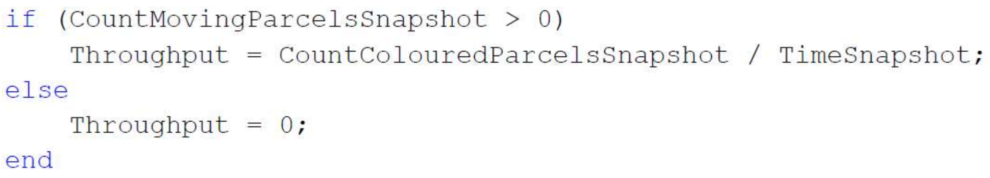
No.	KPIs	Calculation
1	TH: Throughput [units/s]	Program 1
2	CT: Cycle Time [s]	Program 1
3	TP: Transport Performance [kg/m]	Program 1
4	TrU: Transport Utilisation [%]	Program 1
5	TiU: Time Utilisation [%]	Program 2
6	EF: Effectiveness [%]	Program 2
7	AV: Availability [%]	Program 2
8	RE: Reliability [%]	-
9	EE: Energy Efficiency [kWh/unit]	-
10	OEE: Overall Equipment Effectiveness [%]	Program 2

In order to start the data processing system and thus, to integrate it with the data acquisition system, two other apps provided by ThingSpeak are used. While the React app starts the first MATLAB program every time a new data value is received at the first two ThingSpeak channels (group A), which results in a fast calculation of the first four KPIs, the TimeControl app starts the second MATLAB program every five minutes. Finally, the values of the KPIs are sent to two other ThingSpeak channels (group B) for the final storage and visualisation.

The following part describes the calculation of two KPIs in detail.

Throughput

The throughput is calculated through dividing the count of coloured parcels by the snapshot time. However, this formula is only used if moving parcels are detected. As otherwise the throughput is set to zero, this includes the consideration of an unmoving conveyor system. Figure 4-11 shows the related MATLAB code.

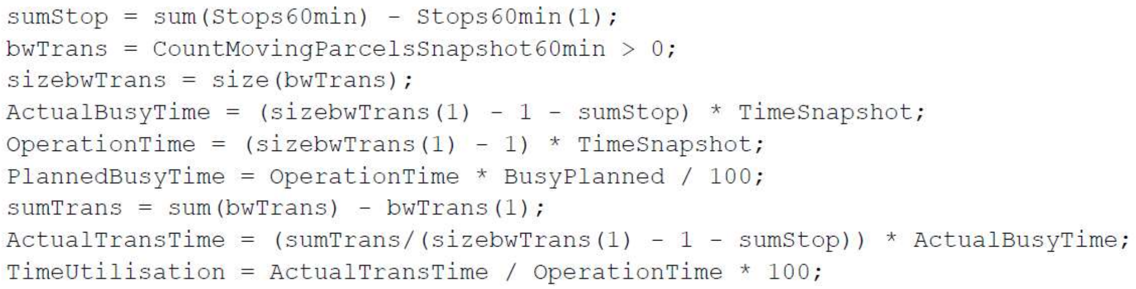


```
if (CountMovingParcelsSnapshot > 0)
    Throughput = CountColouredParcelsSnapshot / TimeSnapshot;
else
    Throughput = 0;
end
```

Figure 4-11: Calculation of the Throughput in MATLAB

Time Utilisation

In order to calculate the time utilisation, the values of the count of moving parcels and the identified stops of the conveyor system for the last 60 minutes are needed. While the total number of stops is calculated through summing up the detected stops, the resultant vector of the count of moving parcels is used to identify the points when at least one moving parcel was detected. While the actual busy time represents the time the conveyor system is running, the operation time includes the actual down time. In order to calculate the planned busy time, the manually entered value for its percentage of the operation time is used. The actual transporting time represents the time when the parcels are actually moving. Finally, the time utilisation is calculated by dividing the actual transporting time by the operation time. While Figure 4-12 shows the related MATLAB code, Figure 4-13 visualises the definition of the related times.



```
sumStop = sum(Stops60min) - Stops60min(1);
bwTrans = CountMovingParcelsSnapshot60min > 0;
sizebwTrans = size(bwTrans);
ActualBusyTime = (sizebwTrans(1) - 1 - sumStop) * TimeSnapshot;
OperationTime = (sizebwTrans(1) - 1) * TimeSnapshot;
PlannedBusyTime = OperationTime * BusyPlanned / 100;
sumTrans = sum(bwTrans) - bwTrans(1);
ActualTransTime = (sumTrans/(sizebwTrans(1) - 1 - sumStop)) * ActualBusyTime;
TimeUtilisation = ActualTransTime / OperationTime * 100;
```

Figure 4-12: Calculation of the Time Utilisation in MATLAB

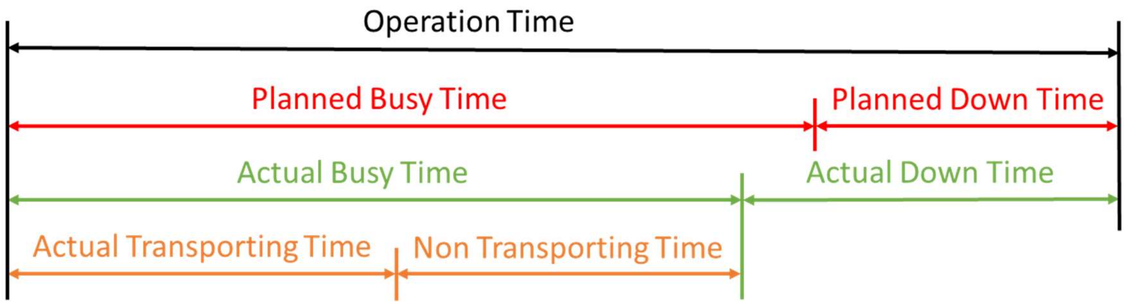


Figure 4-13: Definitions of the Performance related Times

4.3.2 Information Visualisation

The last task of the CPS demonstrator represents the visualisation of the KPIs via two ThingSpeak channels (group B). All ThingSpeak channels automatically visualise the received values over the receiving time in corresponding graphs. Two channels are necessary as two different MATLAB programs are used to calculate the KPIs. While the first channel comprises all KPIs that are calculated via program one including the throughput, cycle time, transport performance and transport utilisation, the second channel provides all KPIs that are calculated via program two including the time utilisation, effectiveness, availability and the overall equipment effectiveness of the last 60 minutes. Furthermore, additional information are provided at the second channel through the visualisation of the average throughput and the proportion of red, green and blue parcels of the last 60 minutes as well as the total throughput including its composition of coloured parcels and the conditions of the conveyor system (running, stop or unknown) over the last 60 minutes. As sample visualisation of the throughput serves Figure 4-14. Sample visualisations of all information are provided in Appendix E.

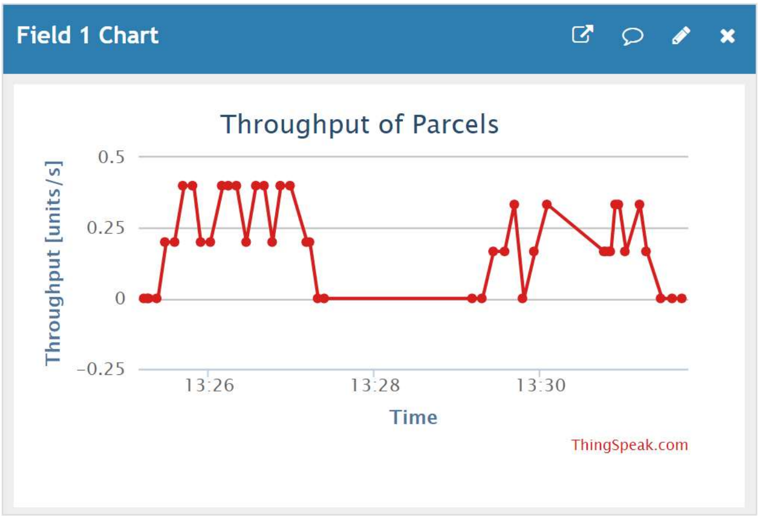
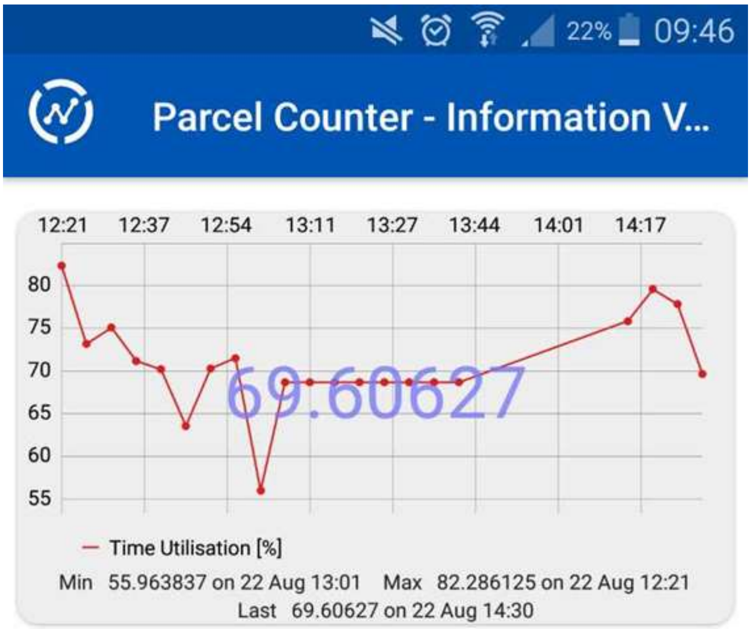


Figure 4-14: Sample Visualisation at a ThingSpeak Channel – Throughput

Finally, the visualisation through cloud-based platforms like ThingSpeak also enables the access to the extracted information via several devices such as a smartphone or a tablet. One possible app that supports that access is “ThingView” from Cinetica Tech. Figure 4-15 provides a sample visualisation of the time utilisation via the ThingView app on a smartphone.



Min 55.963837 on 22 Aug 13:01 Max 82.286125 on 22 Aug 12:21
Last 69.60627 on 22 Aug 14:30

Figure 4-15: Sample Visualisation at the ThingView App – Time Utilisation

4.4 Testing and Evaluation

During the development of the presented CPS demonstrator, general tests were conducted to evaluate the performance of specific functions. Having completed the demonstrator, the final step was the detailed testing of its outputs via a case study as well as the performance of an expert judgement evaluation. This enabled a critical review of the results including the identification of limitations.

4.4.1 General Tests

Conducting general tests resulted in the identification of several key findings related to the technical implementation of the CPS demonstrator. The following paragraphs provide a detailed description of these findings and how they were used to improve the performance of the demonstrator significantly.

As mentioned briefly in Section 4.2.1, the first structure of the data acquisition system included a Raspberry Pi Model B (512 MB SDRAM, 700 MHz CPU frequency). However, the results of some tests have pointed out that using a Raspberry Pi 3 Model B (1 GB SDRAM, 1.2 GHz CPU frequency) provides more benefits. First of all, performing the following case study properly involves the manual count of passing parcels on video displays to acquire the baseline data for a comparison of outputs. Hence, a smoother real time visualisation of detected moving or coloured parcels via the video displays including the possibility to use a higher resolution improved the performance of the case study. Furthermore, the possibility to use two video displays simultaneously (instead of using one at a time) supported testing whether the CPS demonstrator detects moving and coloured parcels correctly. Therefore, the outputs of the video displays that show the real-time count of moving parcels and the real-time count of coloured parcels were compared to the actual number of passing parcels on the conveyor system. This comparison demonstrated that while the count of coloured parcels is always completely exact, the count of moving parcels only provides a low accuracy and precision. Consequently, only the outcome of the count of coloured parcels is used to calculate the throughput of parcels. Finally, the most important benefit is the more constant time between sending new data to the ThingSpeak channels. As described in the next section, the occurrence of variations of the time between sending new data adulterates the outputs of the CPS demonstrator. Therefore, the closer these times are to the calculated snapshot time, the more accurate are the overall outputs.

The most important key finding during the general tests was the strong influence of the light condition on the performance of the detection of coloured and moving parcels. Therefore, several tests were conducted with different light conditions by using sunlight from the front, mercury-vapor lamps from above and halogen lamps from the front. While each of these three possibilities provides a different light spectrum and source location, combining them enabled to increase the number of possible conditions further. First tests were focused on the influence of the light condition on the detection of coloured parcels. Fortunately, this influence was easily compensated by adjusting the threshold values for detecting red, green and blue objects in the Simulink model. Even though a constant threshold for red (65) and blue (55) enabled a sufficient performance, a strong variation of the threshold for green depending on the light condition was discovered. While the tests have shown that using the halogen lamps require the lowest green threshold (15), the detection of green parcels with only sunlight performed better at a medium threshold (25). The highest threshold (35) was required in the cases the mercury-vapor lamps were used. These effects are simply explained as among the three available spectrums, halogen light provides the weakest intensity of green and thus, needs the lowest threshold to still detect green objects. While the light of mercury-vapor lamps provide the strongest intensity of green, sunlight is between these two options. Having identified appropriate threshold values for red, green and blue, the next tests were focused on the influence of the light condition on the detection of moving parcels. The results of these tests have identified that providing only sunlight enables the best performance among all available conditions. As sunlight covers the entire visible light spectrum with the most constant intensity compared to the light of the other lamps is the most likely reason for this effect. Furthermore, the tests have also pointed out that the performance of detecting white moving objects was better than detecting red, green or blue objects at all light conditions. The most likely reason are the improved reflection capabilities of white surfaces. Therefore, even though it was not necessary, all containers were additionally equipped with white paper to improve the performance of detecting moving parcels. The last finding regarding light is related to shadows. Especially front light, as it was provided by the sunlight and the halogen lamps, causes the occurrence of shadows in the area that is covered by the camera. While the shadows of moving parcels were detected as separate moving objects, the shadows of unmoving objects also caused adulterated outputs. As example serves the negative effect of the shadow of the camera itself. However, this problem was easily solved by reducing the height of the position of the camera.

4.4.2 Case Study

Figure 4-16 provides an overview of the structure of the CPS demonstrator to support describing the testing process in detail. While system one performs the data acquisition, the stored data are processed and the extracted information visualised through system two.

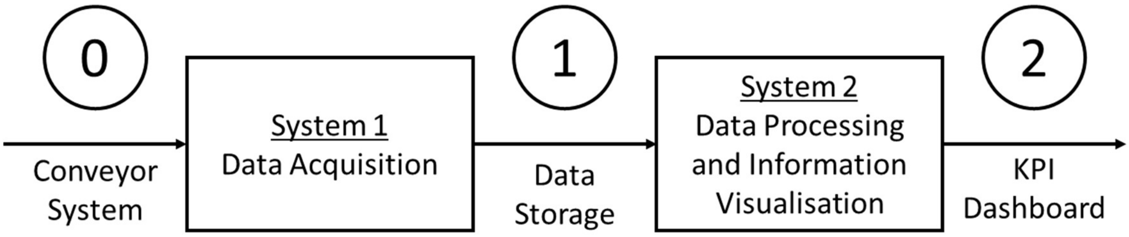


Figure 4-16: Structure of the CPS Demonstrator

In order to realise a detailed testing of the outputs of the demonstrator, a case study including a specific test run was carried out. Therefore, the coloured parcels were transported along circle one for 260 seconds in a random order. The chosen values of the parameters for the Simulink model represented a realistic case in intralogistics. Furthermore, Table 4-6 lists the involved tasks of the test run and the parameter values. The time specifications are absolute and identical with the time provided by the Simulink model. Finally, it is assumed that each red parcel contains one piece of product A, each green parcel contains one piece of product B and each blue parcel contains one piece of product C.

Table 4-6: Steps of the Test Run

Step	Time [s]	Task
0	-	Simulink model and conveyor system are not running.
1	-	Start running the Simulink model on the laptop with the following parameters: 1. Conveyor speed in snapshot area= 0.7 m/s; 2. Conveyor length in snapshot area = 3.25 m 3. Total length of conveyor system = 27.97 m 4. Length of one parcel = 0.5 m 5. Mass of one parcel = 1 kg 6. Mass of product A = 4 kg 7. Mass of product B = 8 kg 8. Mass of product C = 9 kg 9. Snapshot time = 5 s (calculated by the user interface) 10. Planned busy time = 85 % 11. Planned transporting time = 90 %
2	5	Start running the conveyor system.
3	265	Stop running the Simulink model and the conveyor system.

While performing the test run, a video display was used to simultaneously count the passing parcels within the camera area every 5s (snapshot time) manually. Subsequently, the outputs of system one and two of the CPS demonstrator as well as the outcome of the manual count were saved in an excel document. Using the manual count as baseline data, the following part provides the results of its comparison with the outputs of the CPS including the most likely reasons for identified deviations. The chosen way to express a deviation is the difference between the enclosed areas of the resulting profiles and the time-axis as a percentage. This helped to cope with the existing time displacement and represents a simple way to compare the averages of the outcomes. Furthermore, while the term missing points refers to data points that were lost during processing them, the term wrong points refers to presented data points with a value that was supposed to be different. Finally, influences of the communication roundtrip latency are not considered as the test results of Ferrari et al. (2017)

have proven that even simple solutions with widely accepted communication protocols such as MQTT enable a relatively small roundtrip latency of less than 50 ms within the same continent and 300 ms between continents (standard deviation: 20ms).

System 1:

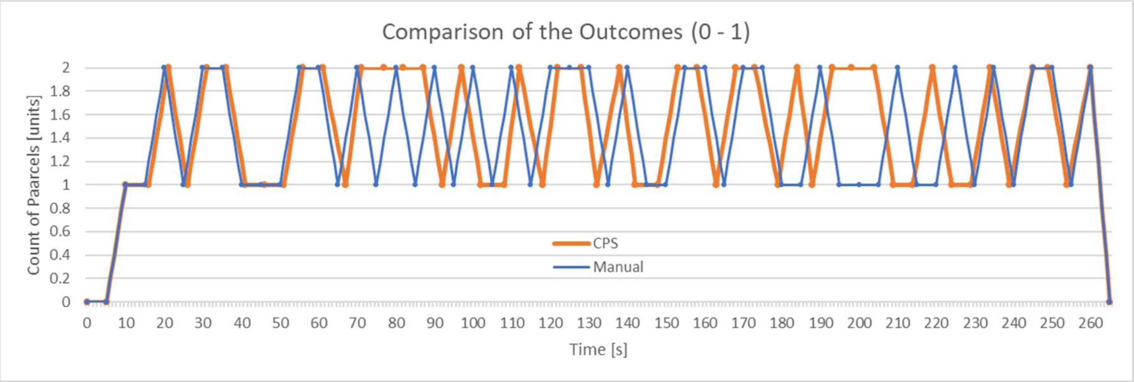


Figure 4-17: Testing Results of System 1 (Points 0-1)

Missing Points:	1/54
Wrong Points:	0/54
Average Time Displacement:	2.5s
Deviation between Areas:	3.16%

Figure 4-17 shows that the first 15 data points of counting parcels manually and via the developed CPS have the same values. However, an increasing time displacement can be observed. This is caused as the Simulink model times sending new data based on the last sending time and not on an absolute reference time. Consequently, as the time between sending new data to the ThingSpeak channels is not constant due to random deviations, each positive or negative deviation from the set snapshot time contributes to a cumulative time displacement. Furthermore, several tests have shown that most of these single deviations are positive. This results in a total time displacement that is increasing with the runtime. Consequently, the counting is performed at another time and thus, parcels might be missed or counted twice. Regarding the case study, the profiles show that the time displacement adulterates the outputs of the demonstrator after 15 data points. However, the adulterated points do not represent wrong points as at the time the counting was performed, the CPS detected all parcels correctly. The figure also shows that the profiles start to become concurrent again after 47 data points as the time displacement has reached 5 seconds (the snapshot time), whereas the error is compensated as the CPS has skipped one data point. This skipped point represents one missing point. Furthermore, it means that the previously described steps including the increasing time displacement start again. Nevertheless, the relatively small deviation between the two enclosed areas indicates that the average of both outcomes is similar.

System 2:

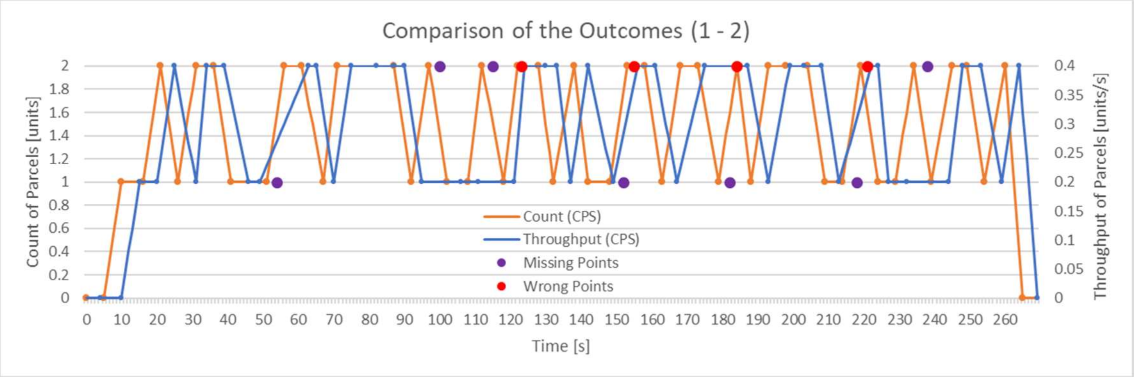


Figure 4-18: Testing Results of System 2 (Points 1-2)

Missing Points:	7/54
Wrong Points:	4/54
Average Time Delay:	4.75s
Deviation between Areas:	1.15%

Figure 4-18 only illustrates outputs of the CPS, whereas their comparison shows a mainly constant time displacement between the profiles of the count and throughput of parcels. This is caused as system two, that starts running when a new data value is received, has a runtime of approximately four to five seconds (strong deviations are possible). This results in an average delay of visualising new values of KPIs of 4.75s. Furthermore, seven data points have been identified as missing and four as wrong. One possible cause for missing points is the minimum snapshot time of one second, defined by ThingSpeak. As the runtime of system two is not constant, it might happen that two or more processes overlap. While simple overlapping does not cause missing points, it might cause that processes send the calculated values of the KPIs to the following channel within that one second. In such cases, one point goes missing. Furthermore, the figure demonstrates that a missing point appears shortly before each wrong point. A possible reason for that effect might be another time delay between the start of system two and reading the last value of the data. In cases that time delay causes a significantly later start of reading the value of the data, a new value might have overwritten the original one. This results in a wrong point as the presented value of the KPI is incorrect for the presented time. Furthermore, an influence of the utilisation of ThingSpeak on the performance of the demonstrator is also expected as several test runs have pointed out that the number of missing and wrong points is not constant. The small deviation between the enclosed areas is expected to be coincidence. Finally, it is important to mention that these limitations including missing and wrong points as well as the time delay for the visualisation only apply on the KPIs estimated via program one as all KPIs estimated via program two are related to a specific time period.

<u>System 1+2:</u>

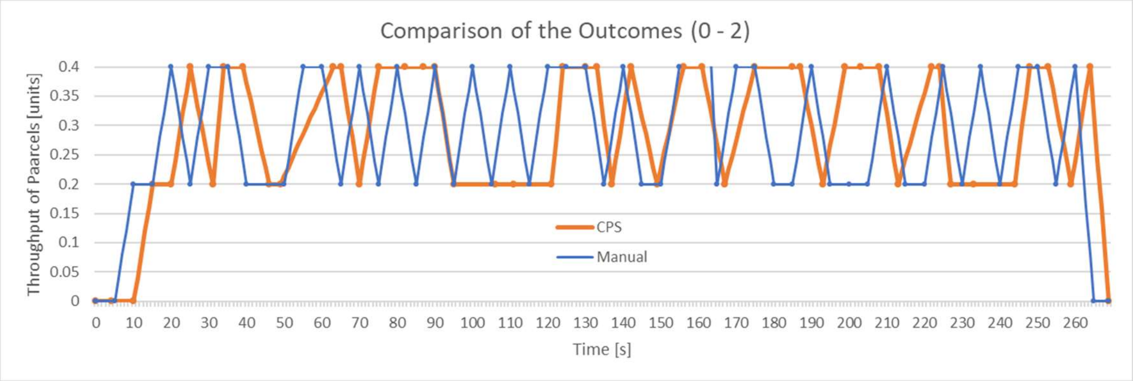


Figure 4-19: Testing Results of System 1+2 (Points 0-2)

Missing Points:	8
Wrong Points:	4
Average Time Displacement:	7.3s
Deviation between Areas:	1.97%

Figure 4-19 is related to the overall system and compares the baseline throughput, calculated via dividing the manual count of parcels by the snapshot time, and the throughput estimated via the CPS. The output of the demonstrator includes all of the previously identified missing (one from system 1, seven from system 2) and wrong points (zero from system 1, four from system 2). Furthermore, as the increasing time displacement of system one and the time delay of system two accumulate, this comparison has identified the highest average time displacement with 7.3s. While the small deviation between the enclosed areas is also expected to be coincidence as several errors compensate each other, it also indicates that the average of both outcomes is similar. Finally, as described for system two, these limitations only apply on the KPIs estimated via program one.

4.4.3 Expert Judgement Evaluation

Furthermore, the outputs of the CPS demonstrator were qualitatively evaluated in the importance to provide the relevant information of the operating conditions and performance of an internal logistic system to enable its optimisation by three experts in the field of logistics (1 Assoc Professors and 2 PhD Candidates of the Institute of Logistics Engineering at Graz University of Technology). The results of the evaluation are shown in the following tables. Table 4-7 provides the evaluation results of the visualised KPIs and Table 4-8 of the additional information. While 5 stands for the highest importance, 1 stands for the lowest.

Table 4-7: Results of the Evaluation of the KPIs

No.	KPIs	Result
1	Throughput [units/s]	5
2	Cycle Time [s]	4
3	Transport Performance [kg/m]	2.33
4	Transport Utilisation [%]	4
5	Time Utilisation [%]	3.67
6	Effectiveness [%]	4.33
7	Availability [%]	4.33
8	Overall Equipment [%]	4.67

Table 4-8: Results of the Evaluation of the additional Information

No.	KPIs	Result
1	Average Throughput of last 60 min	4.67
2	Average Proportion of red, green and blue Parcels of last 60 min.	3
3	Total throughput including its composition of red, green and blue parcels over last 60 min.	4.67
4	Conditions of the Conveyor System over last 60 min.	3.67

Even though the number of evaluators was limited, in the context of this research project, the presented results provide the required evidence that the outputs of the developed CPS demonstrator are accepted by experts in the field of logistics. Furthermore, the results demonstrate that the throughput and directly related information such as the average throughout and the throughput composition have the highest importance. Hence, these KPIs provide the most relevant information of an internal logistic system to enable its optimisation. However, as the transport performance represents information that does not refer to single units and is rather unusual in the field of intralogistics, this KPI has the lowest importance. Besides that, also the importance of the average proportion of coloured parcels is relatively low as it does not provide information about the system's performance. The importance of the remaining KPIs lies between the presented maximum and minimum.

5 Improvements of the CPS Demonstrator

The detailed testing and evaluation of the outputs of the developed CPS demonstrator resulted in identifying several limitations. Decreasing these limitations and thus, improving the demonstrator, requires updating some of the involved processes and systems. This chapter presents the performed updates as well as their effects on the outputs. Regarding Figure 4-16 that illustrates the structure of the involved systems of the CPS demonstrator, system one (data acquisition) remains unchanged as all updates refer to system two (data processing and information visualisation).

The first limitation of the demonstrator, the adulterated count of parcels including the increasing time displacement, arises within the data acquisition system and is caused by deviations of the time between sending new data to the ThingSpeak channels from the set snapshot time. Nevertheless, the fact that the Simulink model times sending new data based on the last sending time and not on an absolute reference time has been identified as the root cause of this limitation. Unfortunately, this fundamental function in Simulink cannot be changed. However, the case study demonstrated that this time deviation only resulted in one missing and zero wrong points while the average outcomes of the manual count and the count via the CPS were still similar. Therefore, it can be concluded that the consequences of this limitation are relatively small and thus, acceptable. On the other hand, the high number of missing and wrong points that are included in the overall output of the CPS demonstrator represent a significant limitation. Besides one missing point caused by the data acquisition system, the root cause of the remaining missing and wrong points lies in the variation of the run time of the data processing system. Furthermore, this run time also causes a time delay between the count of parcels and the visualisation of the related KPIs. Therefore, the outputs of the CPS demonstrator are not close to real time. These considerable consequences led to focus the updates on removing the limitations identified within the data processing system.

5.1 Updates in Data Processing

The data is processed to extract relevant information as suitable KPIs. All assumptions made to estimate these KPIs remained the same, whereas Table 4-4 provides a detailed overview of them. Furthermore, the updated demonstrator is only capable of estimating the same KPIs as the original version. However, the estimation processes have changed significantly.

The basic idea to remove the limitations identified within the data processing system including the missing and wrong points as well as the time delay involves dividing the KPIs that are estimated through the CPS demonstrator into two groups. The first group contains all KPIs that are related to single units (throughput, cycle time) or are directly dependent on the throughput (transport performance, transport utilisation). In contrast, the second group contains all KPIs that are related to a specific period of time (time utilisation, effectiveness, availability, overall equipment effectiveness). Furthermore, due to the updates, the data processing is performed with six instead of two MATLAB programs. While the first five programs actually calculate KPIs, the sixth one is only an auxiliary program. Table 5-1 illustrates to which group the KPIs belong as well as which MATLAB program is used to estimate them.

Table 5-1: Calculated Key Performance Indicators after Updates

No.	KPIs	Group	Calculation
1	TH: Throughput [units/s]	1	Program 1
2	CT: Cycle Time [s]	1	Program 2
3	TP: Transport Performance [kg/m]	1	Program 3
4	TrU: Transport Utilisation [%]	1	Program 4
5	TiU: Time Utilisation [%]	2	Program 5
6	EF: Effectiveness [%]	2	Program 5
7	AV: Availability [%]	2	Program 5
8	OEE: Overall Equipment Effectiveness [%]	2	Program 5

Starting with group two, the values of all involved KPIs are calculated through the same MATLAB program, program 5, created with the Analysis app in the ThingSpeak environment. Furthermore, this program is executed every five minutes by the TimeControl app. While the estimation process of these four KPIs was also originally performed in the same way, even the underlying processes including the used formulas have remained unchanged. Therefore, the time utilisation is still calculated as described and illustrated in Figure 4-12. As mentioned earlier, all KPIs of group two are related to a specific period of time. Updating the estimation process of these KPIs only included decreasing this period from 60 minutes to only 5 minutes. Hence, each calculated value is exactly representative for the time between the previous and the related data point. This concept is visualised in Figure 5-1 as example for the estimation of the time utilisation.

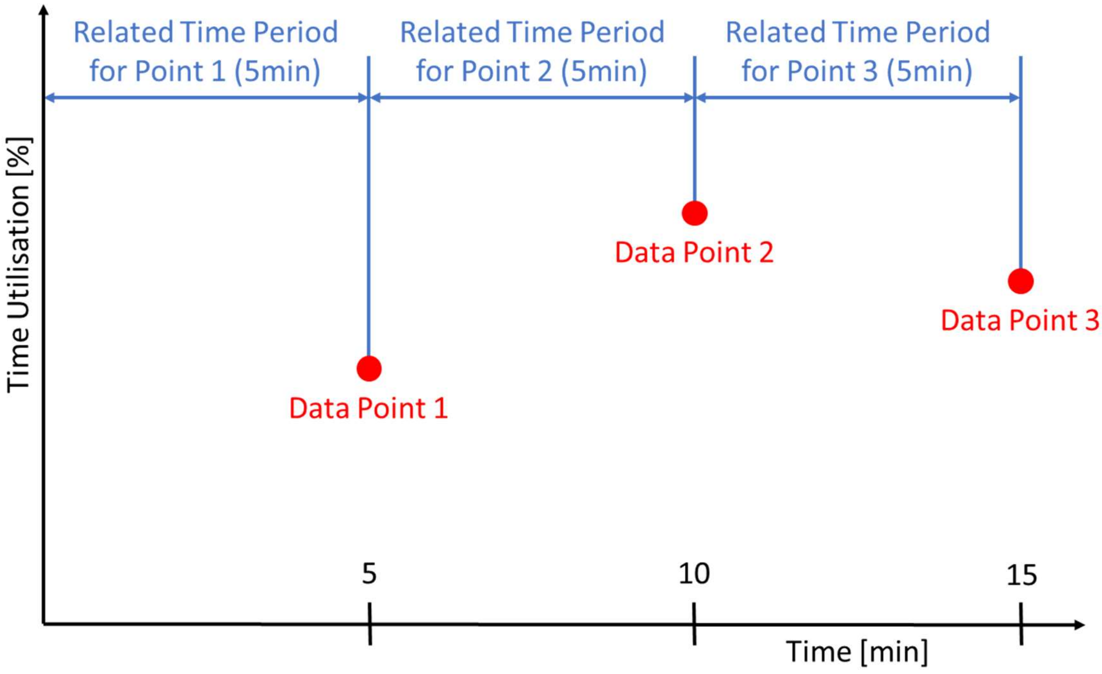

Related Time Period for Point 1 (5min)
Related Time Period for Point 2 (5min)
Related Time Period for Point 3 (5min)
Time Utilisation [%]
Data Point 1
Data Point 2
Data Point 3
5
10
15
Time [min]

Figure 5-1: Concept of Calculating the Time Utilisation

Calculating the values of the KPIs of group one followed a completely new approach that included using the timestamp assigned to each data point when it is received at a ThingSpeak channel. While the underlying formulas to estimate the involved four KPIs have also remained unchanged, it was neither possible to use the React nor the TimeControl app for executing the required MATLAB programs. Consequently, the Visualisation app in ThingSpeak was used to create the programs one to four. Compared to MATLAB programs created with the Analysis app, programs created via the Visualisation app have the advantage that they can directly be implemented in any ThingSpeak channel as independent visualisation field. Furthermore, refreshing the webpage of a ThingSpeak channel results in executing all implemented visualisation programs. However, as one visualisation program is only capable of visualising one KPI, four separate programs were created. Each visualisation program reads the required data values of the last 60 minutes stored at the first two ThingSpeak channels (group A) and saves them as vectors. Additionally, also the corresponding timestamps are read and saved as separate vector. Subsequently, each program uses the data to calculate the values of the related KPI of the last 60 minutes, whereas each provided data value results in one new KPI value. Finally, specific commands enable visualising the values of these KPIs related to the corresponding timestamps. Hence, all missing and wrong data points caused by the data processing system as well as the time delay are successfully removed for these four KPIs.

Furthermore, another approach to increase the reliability of the outputs of the CPS demonstrator is averaging the estimated values of the KPIs over a specific period of time. This approach slightly compensates the adulterated count of parcels and the increasing time displacement. Nevertheless, only the KPIs in group one can be averaged as the KPIs in group two are already related to a time period. While there exist several methods to realise averaging, Figure 5-2 illustrates the concept of averaging the throughput by calculating the arithmetic mean of five data points as sample.

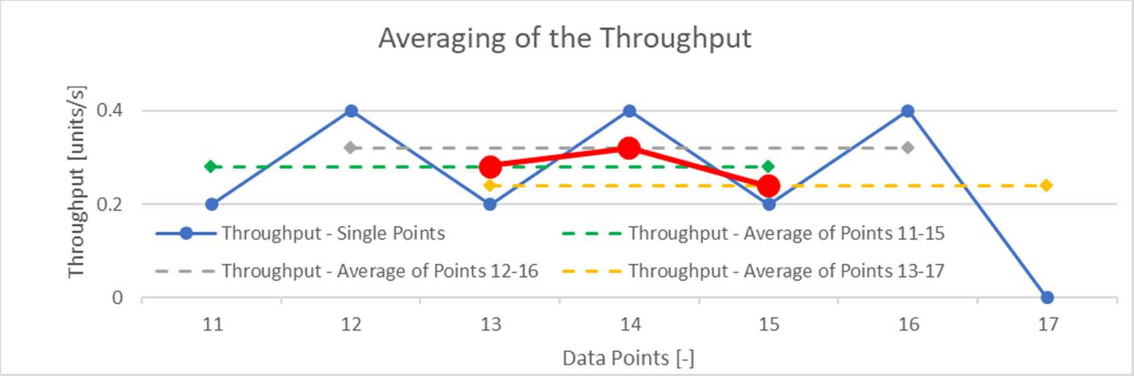


Figure 5-2: Concept of Averaging the Throughput – Arithmetic Mean of 5 Data Points

Finally, the React app starts program six every time a new data value is received at the first two ThingSpeak channels (group A). This program was created with the Analysis app and detects stops of the conveyor system. Furthermore, it sends each detection to an auxiliary ThingSpeak channel, whereas program five uses them to calculate the time utilisation. The related MATLAB codes of all previously described programs are provided in Appendix D.

5.2 Updates in Information Visualisation

The updated visualisation of some KPIs was a consequence of updating the data processing system. Like in the original demonstrator, KPIs as well as additional information are visualised via two ThingSpeak channels (group B), whereas the content of both has changed. The first channel comprises all KPIs that are part of group one including the throughput, cycle time, transport performance and transport utilisation. The graphs of these four KPIs are created via the "thingspeakplot" command in the visualisation programs. This command requires the values of the KPIs and the corresponding timestamps as separate vectors and enables to add a title, axis labels etc. As sample of the updated visualisation of the throughput serves Figure 5-3.

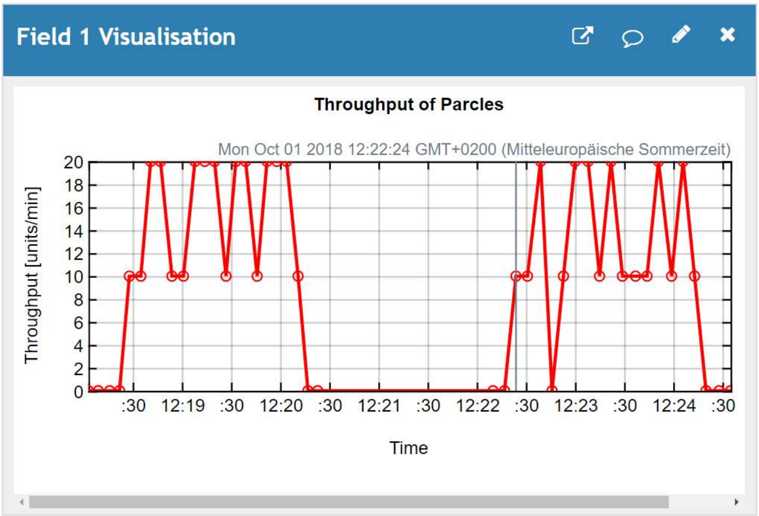


Figure 5-3: Sample Visualisation at a ThingSpeak Channel – Updated Throughput

Furthermore, this channel also includes the visualisation of additional information including the arithmetically averaged throughput of three and five data points as well as the conditions of the conveyor system (running, stop or unknown) over the last 60 minutes and the proportion of red, green and blue parcels of the last 60 min. Figure 5-4 illustrates a sample visualisation of an averaged throughput. Furthermore, Appendix D provides the MATLAB codes to perform the visualisation of the additional information. As the averaged values shown in

Figure 5-4 are based on the values shown in Figure 5-3, the comparison of these profiles indicates that averaging results in a smoother shape.

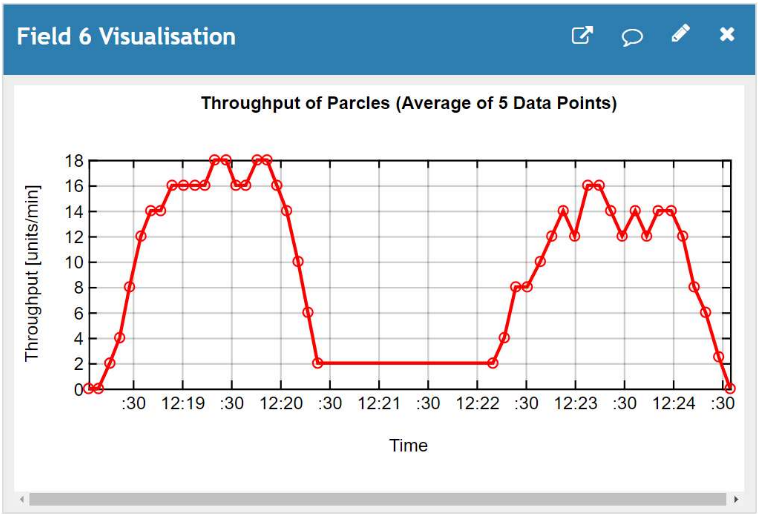


Figure 5-4: Sample Visualisation at a ThingSpeak Channel – Averaged Throughput

Finally, the second ThingSpeak channel comprises all KPIs that are part of group two including the time utilisation, effectiveness, availability and the overall equipment effectiveness. Figure 5-5 provides a sample visualisation of the updated time utilisation. However, besides decreasing the related time period from 60 to 5 minutes, the visualisation of all KPIs of group two has not changed. Sample visualisations of all KPIs and additional information are provided in Appendix E.

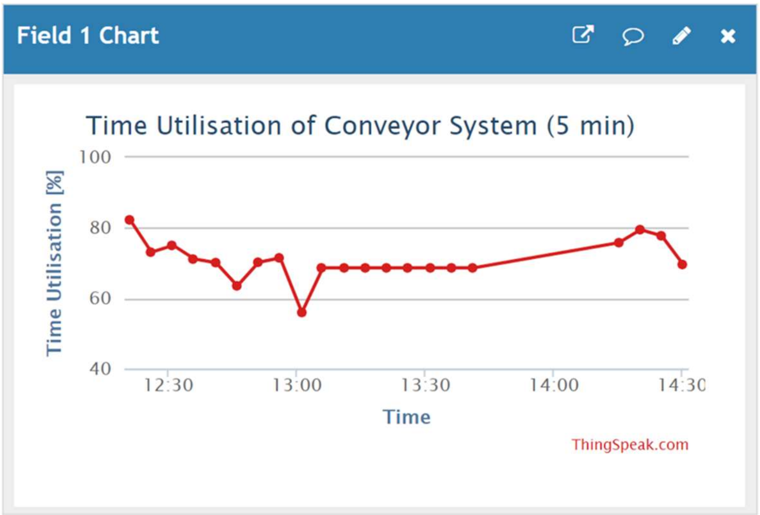


Figure 5-5: Sample Visualisation at a ThingSpeak Channel – Updated Time Utilisation

5.3 Effects of the Updates

Investigating the effects of the performed updates on the outputs of the CPS demonstrator represents the final step to complete the desired improvements. First of all, as the data acquisition system has not been updated, it still involves the adulterated count of parcels including the increasing time displacement. However, all limitations within the original data processing and information visualisation system have been removed successfully through its updating. Applying this understanding on the case study presented in Section 4.4.2 results in the outcomes illustrated in Figure 5-6. The additional performance of another case study is not necessary as based on this understanding, the effects of the updates on the outputs of the demonstrator can be easily predicted. In order to describe this process in more detail, the estimation of the throughput via the updated CPS serves as example. As both, the original and the updated data acquisition system involve the same limitations, the outcome of counting parcels via the original demonstrator for the previously performed case study, presented in Figure 4-17, is also valid for the updated version. Furthermore, no other limitations affect this outcome as the limitations within the following data processing and information visualisation system have been removed. Hence, the throughput estimated by the updated demonstrator simply represents the count of parcels via the CPS divided by the snapshot time. Performing this calculation manually resulted in the outcome presented in the following figure. The manual count of parcels performed for the previous case study serves as baseline data.

<u>System 1+2:</u>

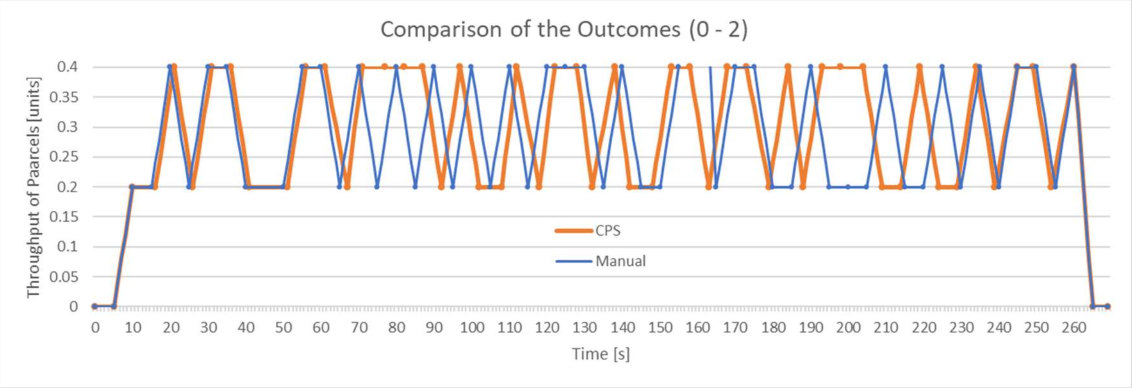


Figure 5-6: Updated Testing Results of System 1+2 (Point 0-2)

Missing Points:	1/54
Wrong Points:	0/54
Average Time Displacement:	2.5s
Deviation between Areas:	3.16%

Figure 5-6 compares the baseline throughput, calculated via dividing the manual count of parcels by the snapshot time, and the throughput estimated by the updated CPS, calculated via dividing the count of parcels from the original CPS by the snapshot time. This comparison shows the exact same characteristics as Figure 4-17. While the first 15 data points have the same values, an increasing time displacement can be observed. This is also caused as the Simulink model times sending new data based on the last sending time and not on an absolute reference time. While the profiles show that the time displacement adulterates the outputs of the demonstrator after 15 data points, these adulterated points do not represent wrong points. Furthermore, the figure also shows that the profiles start to become concurrent again after 47 data points. As the updated CPS still skips one point, one missing point is identified. The relatively small deviation between the two enclosed areas indicates that the average of both outcomes is similar. Therefore, it can be concluded that the involved limitations have been reduced significantly as well as that the consequences of the still involved limitations are relatively small and thus, acceptable. This results in an improved CPS demonstrator.

6 Machine Learning Applications in Intralogistics

The previous chapter effectively demonstrated one possible realisation of IoT-driven data analytics through the implementation of a CPS demonstrator for a representative use case in intralogistics. This demonstrator visually presents information about the operating conditions and performance of an internal logistic system as KPIs. Having successfully provided these information, the following step includes their detailed analysation to enable gaining the required knowledge to realise optimisation. While one possible solution is an analysation performed by human operators, using machine learning applications helps finding hidden patterns and groupings even more effectively. The literature review has identified clustering as the most appropriate algorithm to find such hidden structures with similar features. This represents an important benefit as it enables building a deeper understanding of the extracted information. Furthermore, the incorporation of context awareness in machine learning enables discovering even more hidden relations between the situations under which the internal logistic system is operating. Therefore, all information provided through the developed context model for intralogistics, presented in Section 4.1.2, represents the input of the clustering algorithm. This results in a further improved understanding of the internal logistic system and thus, in gaining more knowledge effectively.

However, the capability of machine learning applications to realise further optimisation in intralogistics is much broader than only the analysation of extracted information. One example represents the identification of specific conditions within the internal logistic system through image recognition. This might be realised through a classification model implemented in a CPS. Hence, the model classifies images depicting parts of a conveyor system into several categories, whereas each category represents one certain condition. Providing this additional information enables gaining more knowledge and thus, realising further optimisation. As employing a CPS with integrated machine learning applications represents a relatively new approach to optimise an internal logistic system, this chapter is focused on building a proper understanding how a classification model can be implemented in a CPS. Furthermore, the literature review has discovered that using artificial neural networks to solve such classification problems offers several benefits in data volume and speed. Therefore, the following work is only focused on this type of models. The first section of this chapter provides a detailed understanding of neural networks including the structure, training methods and possible improvements. An additional investigation of related literature was needed to gain the required knowledge and enabled the development of several figures and equations to support the written explanations. Finally, appropriate ways to develop an artificial neural network and implement it in a CPS to enable gaining additional knowledge of an internal logistic system are investigated in the second part.

6.1 Fundamentals of Artificial Neural Networks

This section is dedicated to providing a detailed understanding of artificial neural networks including the structure, training methods and possible improvements.

6.1.1 Introduction to Artificial Neural Networks

As already mentioned in Section 2.4.2, the development of artificial neural networks was inspired by emulating the human brain. Human brains comprise a large number (10^{11}) of processing units, called neurons. Neurons are cells that are able to transmit and process chemical or electrical signals. Receiving such a signal, the neuron determines its strengths, and in case it exceeds a specific threshold, the signal is passed on to the output. Finally, connecting all these neurons through synapses creates a network. The idea of artificial neural networks represents the realisation of a network that consists of many processors, each executing a simple function. Furthermore, changing multiple parameters enables adapting the network to a changing environment. However, this can only be realised successfully in case this network is capable of learning automatically what the optimal values of these parameters are. As artificial neural networks provide strengths in data volume and speed, they are mostly used for real-time or near real-time systems. Typical application areas comprise high-frequency trading, credit applications, data centre management, robotics or medical monitoring. (Alpaydin, 2014; Bell, 2014)

The understanding of perceptrons builds the basis for artificial neural networks. Bell (2014) describes a perceptron as algorithm that receives input signals, passes them through a mathematical function and outputs its result. Figure 6-1 illustrates a general structure of a single perceptron including three input neurons x_1, x_2, x_3 and one output neuron y. In contrast to input neurons that only pass on the signal they receive from the environment, each neuron executes one mathematical function, also called activation function, and outputs its result. Even though a single perceptron only consists of one output neuron, the number of input neurons varies depending on the related task. Furthermore, a specific weight is assigned to each input. These weights represent the strength of the connection between the neurons. Finally, the figure also demonstrates the input layer and output layer of the network. (Bell, 2014; Nielsen, 2017)

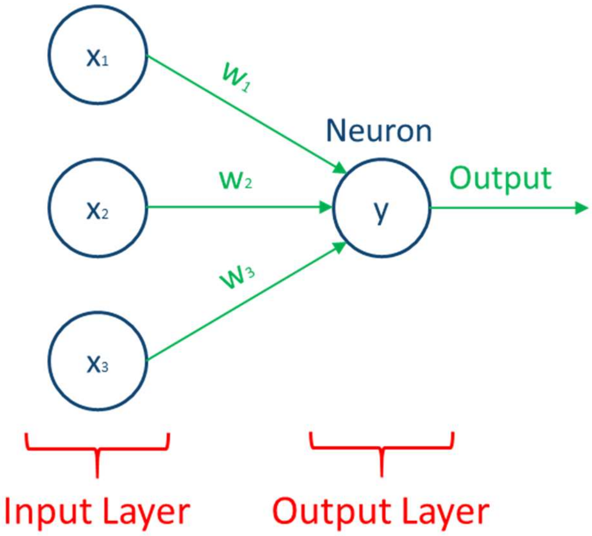


Figure 6-1: Sample Structure of a Perceptron

A common activation function of a single perceptron neuron is the threshold function, illustrated in Equation 6-1. Therefore, comparing the weighted sum of all inputs to a specific threshold determines the output of the neuron. In case the weighted sum is greater than or equal to the threshold value, the output is one, otherwise it is zero. The weights and the threshold represent the parameters of a perceptron. (Alpaydin, 2014; Bell, 2014; Nielsen, 2017)

$$y = \begin{cases} 1\ if \sum w_i * x_i \geq threshold \\ 0\ if \sum w_i * x_i < threshold \end{cases} \tag{6-1}$$

The aim of an activation function is to squish the output of the weighted sum into the range between zero and one. However, as neutrons using the threshold function only output a value that is either one or zero, small changes of the network parameters might result in relatively large changes of the outputs. This affects the learning process quite negatively. In order to ensure a high model performance, other activation functions were needed. One solution of an improved activation function represents the sigmoid function, whereas Equation 6-2 shows its definition. (Bell, 2014; Nielsen, 2017; Weisstein, 2018)

$$\sigma(z) = \frac{1}{1 + e^{-z}} \qquad \textbf{(6-2)}$$

The sigmoid function enables each neuron to take on any value between zero and one. Therefore, small changes of the network parameters only result in small changes of the outputs. While very negative inputs result in a value close to zero, very positive inputs result in a value close to one. Furthermore, the threshold is replaced by the bias. Similar to the threshold, the bias represents an indicator whether a specific neuron tends to be active or inactive. This results in Equation 6-3. (Bell, 2014; Nielsen, 2017)

$$y = \ \sigma(\sum w_i * x_i + bias) \qquad \textbf{(6-3)}$$

Even though the illustrated sample perceptron has only one output neuron, single-layer perceptrons might include several of them. However, single-layer perceptrons only consist of one input and one output layer and thus, are only capable of solving linear problems. Therefore, solving nonlinear problems requires the use of multilayer perceptrons. Multilayer perceptrons additionally contain one or more layers between the input and output layer. These internal layers are called hidden layers. Nielsen (2017) argues that the name hidden layers simply indicates that the layers are neither part of the self-explanatory input nor the output layer. In the case of image recognition, each hidden layer ideally performs a form of pattern recognition, whereas each additional layer supports processing more complexity. Having pixels as input, the first hidden layer might focus on simple edges of various orientations. While the second hidden layer might check for combinations of edges such as arcs or corners, following layers might be focused on circles, rectangles, or even eyes or mouths. In contrast to using only a few hidden layers, representing a shallow neural network, using multiple hidden layers is referred to as deep neural network, a part of deep learning. As the increasing number of hidden layers enables detecting more complex patterns and thus, improves the classification accuracy, deep neural networks are essential in multiple application areas. However, several challenges make a proper training of deep neural networks more difficult. Although the number of neurons in the input and output layer is given by the requirements of the task to be performed, the number of hidden layers including the number of neurons per hidden layer mainly depends on the experience of the network developer. Summarising the understanding provided in this section, an artificial neural network consists of many interconnected neurons, each with its own input, output, and activation function. Figure 6-2 illustrates the sample structure of an artificial neural network including two hidden layers. Interpreting each layer of the network as vector is the simplest and thus, most appropriate way. While both hidden layers contain three neurons each, the input layer comprises four and the output layer two neurons. Furthermore, each neuron of one layer is connected to each neuron of the previous and following layer, which represents a fully-connected layer. Moreover, as the outputs of one layer are only used as inputs of the next layer, there are no loops within this network. Such networks are called feedforward. In contrast, recurrent neural networks allow feedback loops. Therefore, the inputs of one layer also depend on the outputs of the following layer. However, as the learning algorithms for recurrent networks are currently less powerful than for feedforward networks, they are not very common. (Bell, 2014; Nielsen, 2017; Paluszek and Thomas, 2016)

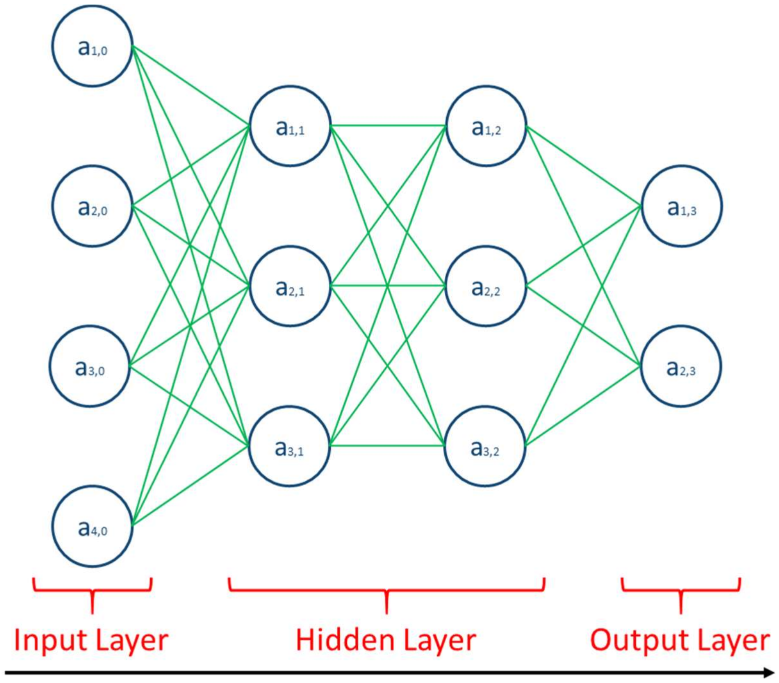


Figure 6-2: Sample Structure of an Artificial Neural Network

Having understood the general structure of artificial neural networks, the next step includes a proper investigation of the involved activation functions. Therefore, Figure 6-3 visualises the connections between the second hidden layer (layer 2) and the output layer (layer 3) in detail. While w stands for the weights (one weight is assigned to each connection between two neurons), b stands for the biases (one bias is assigned to each neuron).

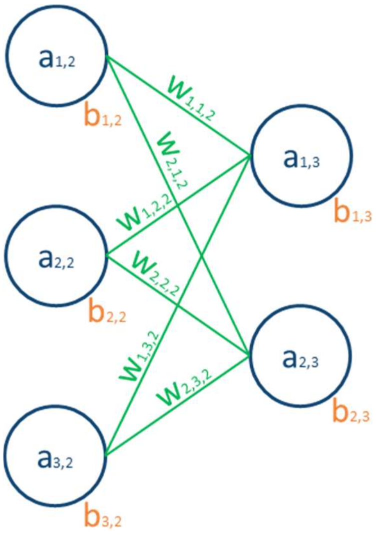


Figure 6-3: Detailed Sector of the Sample Artificial Neural Network

Equation 6-4 shows the activation function of the first neuron of the output layer.

$$a_{1,3} = \sigma(w_{1,1,2} * a_{1,2} + w_{1,2,2} * a_{2,2} + w_{1,3,2a_{3,2}} + b_{1,3}) \tag{6-4}$$

In order to summarise the activation functions of the entire output layer, vectors and matrixes are used. The result is shown in Equation 6-5.

$$\underbrace{\begin{bmatrix} a_{1,3} \\ a_{2,3} \end{bmatrix}}_{\overline{a_3}} = \sigma\left(\underbrace{\begin{bmatrix} w_{1,1,2} & w_{1,2,2} & w_{1,3,2} \\ w_{2,1,2} & w_{2,2,2} & w_{2,3,2} \end{bmatrix}}_{\overline{\overline{W_2}}} * \underbrace{\begin{bmatrix} a_{1,2} \\ a_{2,2} \\ a_{3,2} \end{bmatrix}}_{\overline{b_3}} + \underbrace{\begin{bmatrix} b_{1,3} \\ b_{2,3} \end{bmatrix}}_{\overline{a_2}}\right) \tag{6-5}$$

Finally, summarising the activation functions of each layer results in three simple equations that describe the entire sample artificial neural network mathematically. Its general form is provided in Equation 6-6.

$$\overline{a_n} = \sigma * (\overline{\overline{W_{n-1}}} * \overline{a_{n-1}} + \overline{b_n}) \tag{6-6}$$

6.1.2 Training of Artificial Neural Networks

Even though an appropriate structure of an artificial neural network for a specific task is designed, the values of the weights and biases to achieve the highest classification accuracy are still unknown. Therefore, a proper training of this network is required. During the training process, the model automatically extracts rules for solving a specific task in the best way. As mentioned in Section 2.4.1, supervised learning is used for classification models. Therefore, a sufficient amount of data to learn from, called training set, is necessary. The model performance is mainly determined by its capability to perform an accurate classification of input data. Assuming x represents a vector containing this input data, one simple way to quantify this classification accuracy is by calculating the difference between vector a, representing the estimated output, and vector y, representing the correct output. This approach with n training inputs is reflected through the cost function, also called loss function. As its form varies with the used network, there exists no standard cost function. However, Nielsen (2017) presents one possible form, provided in Equation 6-7, that is suitable for the most artificial neural networks. This function is based on the mean squared error and represents a quadratic cost function. (Alpaydin, 2014; Bell, 2014; Nielsen, 2017; Paluszek and Thomas, 2016)

$$C(w,b) = \frac{1}{2n}\sum \|y(x) - a\|^2 \tag{6-7}$$

The cost function C(w,b) represents the sum of the squares of the differences between the estimated output and the correct output. As only the weights and biases represent variable parameters, the output of the cost function only depends on their values. While C(w,b) is able to take on any positive value, a value close to zero indicates that the estimated output is approximately equal to the correct output and thus, that the classification accuracy is very high. Therefore, the values of the weights and biases are already at or close to their optimum. In contrast, a high value of C(w,b) indicates that the estimated output is rather different to the correct output. In this case, it is very likely that the values of the parameters are not close to their optimum. Derived from this understanding, the aim of the learning process is to identify the values of the weights and biases that minimise the cost function. Even though a common way to identify minima of a function analytically is through differential calculus, this is not possible for artificial neural networks due to the large number of parameters. However, within the domain of multivariable differential calculus, the gradient of a function returns the direction of the steepest ascent. In simpler words, which direction increases the output of the function in the quickest way. Furthermore, the length of the vector is an indicator of the degree of this steepness. (Alpaydin, 2014; Nielsen, 2017)

Applying the gradient on the cost function results in Equation 6-8.

$$\nabla C(w,b) = \left(\frac{\partial \mathrm{C}}{\partial \mathrm{w}_1}, \dots, \frac{\partial \mathrm{C}}{\partial \mathrm{w}_m}, \frac{\partial \mathrm{C}}{\partial \mathrm{b}_1}, \dots, \frac{\partial \mathrm{C}}{\partial \mathrm{b}_k}\right)^T \tag{6-8}$$

In this equation, m stands for the total number of connections (one weight is assigned to each connection between two neurons) and k stands for the total number of neurons minus the number of input neurons (one bias is assigned to each neuron, except input neurons). Quite logically, taking the negative of this gradient results in the direction that decreases the output of the function in the quickest way. (Alpaydin, 2014; Nielsen, 2017)

In order to enable a better overview of all parameters, the vector v is introduced in Equation 6-9.

$$\bar{v} = (\mathrm{w}_1, \dots, \mathrm{w}_m, \mathrm{b}_1, \dots, \mathrm{b}_k)^T \tag{6-9}$$

Multiplying the gradient of the cost function with another factor, called the learning rate, enables adjusting the size of one step towards the minimum output of the cost function. The changes for each parameter of one step are calculated according to Equation 6-10. (Alpaydin, 2014; Nielsen, 2017)

$$\Delta v = -\eta \nabla C(w, b) \tag{6-10}$$

The algorithm that aims to find the minimum of C(w,b) is called gradient descent. Starting with random values, it repeatedly computes the required changes of the involved parameters to minimise the output of the cost function. Therefore, changing the values of these parameters accordingly to the negative output of the multiplication “learning rate times gradient of C(b,w)” represents on step towards this minimum. Furthermore, the learning rate must be small enough to ensure that the gradient descent algorithm works correctly. This results in a learning process in which the optimal values of the weights and biases in the network are determined automatically. However, even though the aim is to find the global minimum of the cost function, depending on the initial values of the parameters, only a local minimum might be found. Backpropagation, short for backward propagation of errors, is the underlying algorithm to calculate the gradient of C(w,b) efficiently. (Alpaydin, 2014; Bell, 2014; Nielsen, 2017)

According to Nielsen (2017), this algorithm includes the following steps:

1. Enter input data in the artificial neural network.
2. Feedforward the data to estimate the output.
3. Compute the error vector of the output.
4. Propagate the error vectors of each layer backwards.
5. Compute the gradient of each parameter by using the related error

The name of this algorithm has its origin from the fact that the errors at each layer are computed backwards. As the importance of backpropagation increases with the number of hidden layers, it is a common method in deep neural networks. Furthermore, a common learning algorithm is the stochastic gradient descent. This involves randomly splitting up the entire training set in several mini-batches. Subsequently, the gradient descent algorithm is performed with the first mini-batch. Even though this results in changes that are only representative for this specific mini-batch, repeating the performance of the gradient descent algorithm with the remaining mini-batches provides an incrementally improving approximation to the optimum. Although this is not the most efficient method, it enables a significant computational speed up. While the presented training using stochastic gradient descent and backpropagation algorithms works quite well for shallow neural networks, several challenges make the proper training of deep neural networks difficult. However, the development of different network architectures has improved the training performance of deep networks significantly. (Alpaydin, 2014; Nielsen, 2017)

6.1.3 Convolutional Neural Networks

Nielsen (2017) presents a traditional neural network only consisting of fully-connected layers to recognise greyscale images of handwritten digits between 0 and 9, provided by the MINST database. While the training set of this database comprises 60 thousand images, the test set comprises 10 thousand. Furthermore, each image consists of 28x28 pixels. Hence, the input layer contains 784 (=28x28) neurons, whereas the intensity of each input pixel determines the value of the corresponding neuron. As the provided digits range from 0 to 9, the output layer contains 10 neurons. Completing the network with one hidden layer that includes 100 neurons results in 79510 parameters (79400 weights and 110 biases) in total. Having trained this model, it achieved a classification accuracy of about 98 percent. In order to increase this accuracy even further, the incorporation of the spatial structure in the image recognition process was necessary. This led to the development of convolutional neural networks, also called CNN or ConvNet, that currently enable the best performance in image recognition. Applied on the same images of handwritten digits as the traditional network, Nielsen (2017) increased the classification accuracy by using a convolutional neural network to 99.6%. A detailed description of the structure of this ConvNet is provided later in this section. Furthermore, the architecture of such networks allows a fast training and thus, solves current challenges in training deep neural networks. According to Zeiler and Fergus (2014), ConvNets have become a requirement to realise a state-of-the-art performance in image recognition. (Nielsen, 2017; Zeiler and Fergus, 2014)

A sample structure of a ConvNet is illustrated in Figure 6-4. Like the structure of traditional networks, it starts with an input layer. Furthermore, the structure includes several convolutional layers, whereas one convolutional layer contains one or more feature maps. Quite often, a convolutional layer is followed by a pooling layer to simplify its contained information. Like the convolutional layer, the pooling layer contains one or more pooling matrixes. Repeating convolutional and pooling layers multiple times represents a deep convolutional network. Besides that, also fully-connected layers might be included in a ConvNet. As usual, an output layer completes the network. The following paragraphs describe each layer in detail. While interpreting all layers of a traditional network (only consisting of fully-connected layers) as vectors is the simplest and thus, most appropriate way, interpreting the input, convolutional and pooling layers of a convolutional network as matrixes enables a better understanding of the involved processes. However, in order to present the classification results in a suitable way, the output layer is usually a fully-connected layer. Therefore, it is still interpreted as vector. All layers between the input and output layer might also be summarised as hidden layers. (Nielsen, 2017; Ujjwal, 2016; Wikipedia, 2018a)

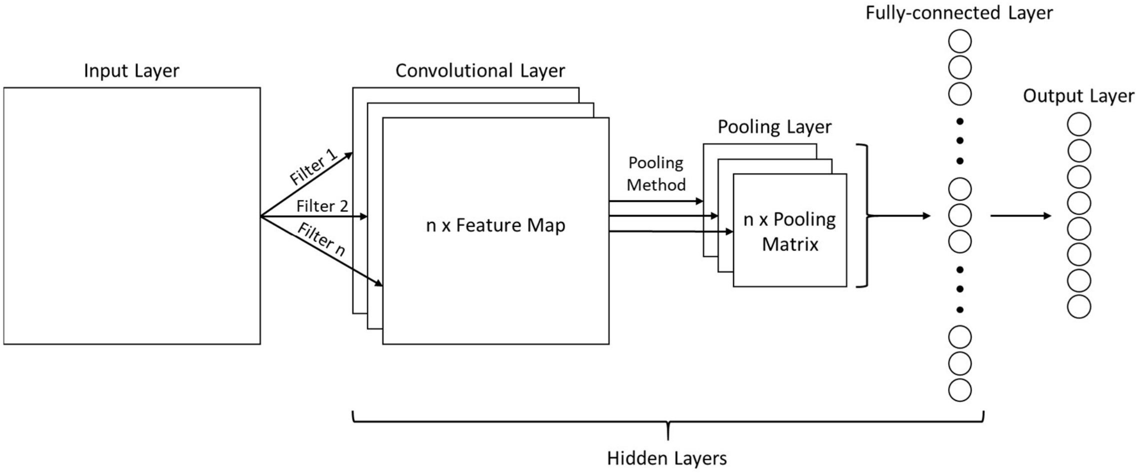


Figure 6-4: Sample Structure of a ConvNet (adapted from Nielson, 2017)

In order to calculate the values of the neurons in a feature map, the filter is introduced. A filter represents a small matrix, whereas the values of its elements determine the feature the filter detects. While these elements are also called weights, one bias is assigned to each filter. Like in traditional networks, these weights and biases represent the variable parameters. The identification of their optimal values is performed by the network automatically during the training process. Appling a certain activation function on the bias assigned to the filter plus the scalar product of the area of the receptive field in the input matrix (the area in the input matrix that is currently covered by the filter) and the filter matrix results in the value of the corresponding neuron in the feature map. Equation 6-11 illustrates the general form of this formula in detail, whereas the indices j and k identify one specific neuron in the feature map and the indices l and m identify one specific element in the filter matrix. Furthermore, x and y stand for the dimension of this filter. Nielson (2017) describes the origin of the name "convolutional neural network" as this equation is sometimes known as a convolution. (Nielsen, 2017; Ujjwal, 2016; Wikipedia, 2018a)

$$a_{n,j,k} = \sigma(bias_{n-1} + \sum_{l=1}^{x} \sum_{m=1}^{y} w_{n-1,l,m}\, a_{n-1,j+l,k+m}) \qquad \textbf{(6-11)}$$

Therefore, instead of having a fully-connected layer in which each neuron is connected to all neurons of the previous layer, each neuron in a feature map is connected to a small specific region of the related matrix in the previous layer. Assuming this filter represents a 5x5 matrix, Figure 6-5 illustrates a sample connection of a receptive field in the input matrix with the corresponding neuron in the feature map. Starting with the first 5x5 field in the input matrix, moving the filter over the entire matrix and calculating the values of the corresponding neurons completes the feature map. Therefore, feature maps also include the locations of the detected features. The number of pixels the filter is moved at a new step is called stride and might range from one to the size of the filter. In case a 5x5 filter is applied on a 28x28 input matrix, the related feature map in the following layer typically comprises 24x24 neurons. However, Ujjwal (2016) presents a technique called zero padding to create a feature map with the same size as the related matrix. Therefore, several lines of zeros are added around the border of the input matrix. (Nielsen, 2017; Ujjwal, 2016; Zeiler and Fergus; 2014)

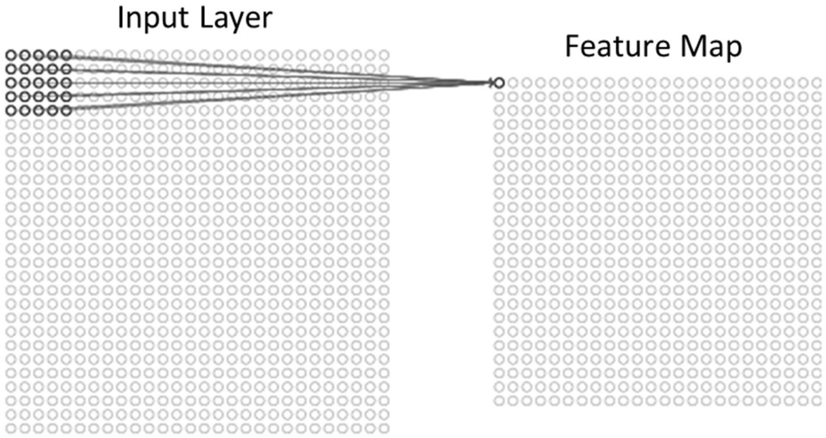


Figure 6-5: Sample Connection of a Receptive Field with a Feature Map Neuron (adapted from Nielson, 2017)

As one feature map involves only one corresponding filter, all neurons of this particular map are focused on detecting the same feature, for example horizontal edges. In order to enable detecting different features included in one image, several filters must be used. However, using several filters results in creating several feature maps. Therefore, one convolutional layer typically comprises more than one feature map. This enables the incorporation of the spatial structure in the image recognition process and thus, achieving a much higher classification accuracy than achievable with traditional networks. Ujjwal (2016) describes that the depth of a convolutional layer corresponds to the number of involved feature maps. Zeiler and Fergus (2014) conducted several tests regarding the depth of convolutional layers and argue that especially the overall depth of the network is a deciding factor for the model performance. Furthermore, using only one filter to create one entire feature map results in a smaller number of total parameters due to sharing the same weights and the bias of this filter. Hence, fewer parameters have to be optimised during the training process. This enables a faster training of convolutional networks than training traditional networks and thus, solves current challenges in training deep neural networks. (Nielsen, 2017; Wikipedia, 2018a; Ujjwal, 2016; Zeiler and Fergus, 2014)

Quite often, a convolutional layer is followed by a pooling layer. The aim of such a pooling layer is to simplify the output information from the previous convolutional layer. As each feature map of the convolutional layer results in one pooling matrix, the pooling layer contains as many matrixes as the previous convolutional layer. A common method to simplify information of a small region in a feature map is max-pooling. Therefore, the value of a neuron in a pooling matrix corresponds to the highest neuron value within the related region in the feature map. Figure 6-6 illustrates a sample connection of one neuron in a pooling matrix with the related feature map by summarising a 2x2 region. This would reduce the number of neurons in a pooling matrix from 24x24 to 12x12. Max-pooling represents a simple method to identify whether a specific feature is present in the related feature map. Furthermore, only the approximate locations of these detected features relative to the others is important for further steps. Therefore, the loss of the detailed positional information is acceptable and even reduces the number of required parameters in following layers. Another widely used method to simplify information in the pooling layer is called L2 pooling. Instead of taking only the maximum value of a specific region in a feature map, this method takes the square root of the sum of the squares of each element. (Nielsen, 2017; Ujjwal, 2016; Wikipedia, 2018a)

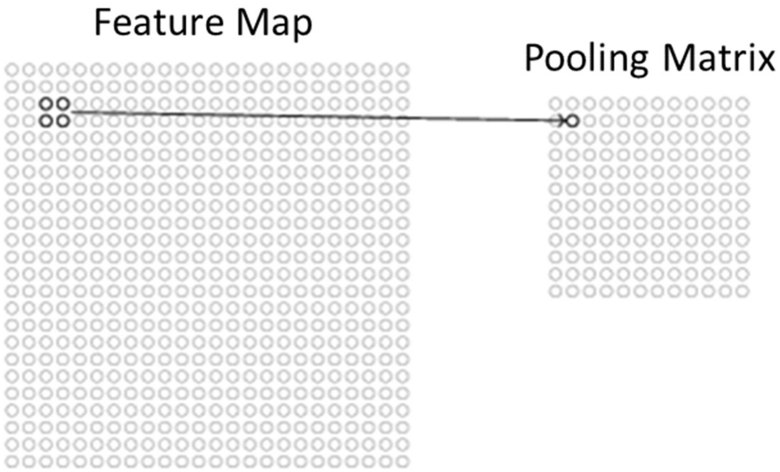


Figure 6-6: Sample Connection of a Feature Map and a Pooling Matrix (adapted from Nielson, 2017)

As illustrated in Figure 6-4, an output layer completes a ConvNet. While interpreting other layers as matrixes enables a better understanding of the underlying processes, interpreting the output layer as vector is the most appropriate way. Like in traditional networks, the output layer of a convolutional network is a fully-connected layer and presents the results of the classification processes. Therefore, each neuron of the output layer is connected to each neuron of the previous layer. Besides that, putting additional fully-connected layers before the output layer is a common method to improve the model performance further. However, the final choice lies on the human developer and mainly depends on the gained experience. Even though the basic architecture of convolutional networks is different to traditional networks, the underlying concept is still the same. The classification accuracy of a ConvNet also depends on the values of the involved parameters including weights and biases of filters as well as weights and biases of fully-connected layers. Furthermore, only an appropriate training enables identifying the optimum of these values. Like traditional networks, also convolutional networks are typically trained using stochastic gradient descent and backpropagation algorithms. (Nielsen, 2017; Ujjwal, 2016; Wikipedia, 2018a)

As mentioned at the beginning of this section, Nielsen created a ConvNet to recognise images of handwritten digits between 0 and 9, provided by the MINST database. This model achieves a classification accuracy of 99.6%. Starting with an input layer containing 784 (=28x28) neurons, 20 filters covering a receptive field of 5x5 are applied to create 20 feature maps in the first convolutional layer. Subsequently, 2x2 regions of each feature map are summarised via max pooling. As large networks are capable of detecting more complex and detailed features, using another convolutional layer improves the classification accuracy further. Applying two 5x5 filters on the 20 pooling matrixes of the first pooling layer results in 40 feature maps in the second convolutional layer. Furthermore, also max pooling is used to summarise 2x2 regions of these feature maps. Having created the second pooling layer, two fully-connected layers, each containing 1000 neurons, are also added and realise the final improvement of the classification accuracy. Finally, ten output neurons present the result of the image recognition process. (Nielsen, 2017)

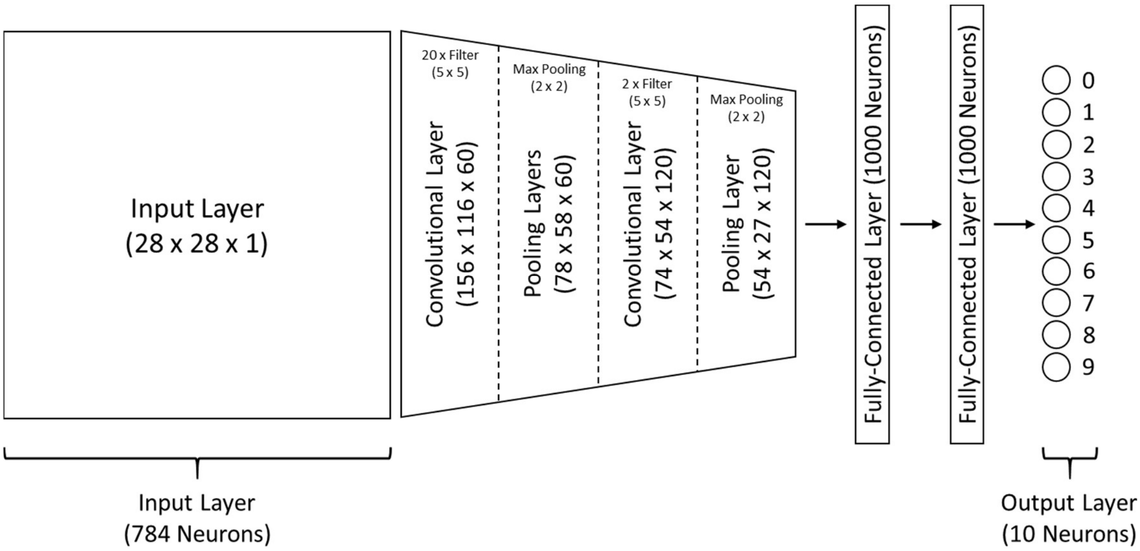


Figure 6-7: Structure of the ConvNet created by Nielson (2017)

6.1.4 Additional Ways to Improve the Model Performance

Having developed a first draft of an artificial neural network to perform image recognition, typically there are still several ways to improve the model performance. This section provides an overview of some widely used methods to realise such improvements.

First of all, the literature review has discovered that overfitting is a huge problem in deep learning as it causes a poor level of generalisation. Even though Deshpande (2017) presents a simple way to detect overfitting models through comparing their errors with training and validation data, whereas having a significantly higher error with validation data indicates the occurrence of overfitting, ways to reduce overfitting are not included. Nielsen (2017) argues that one simple method to reduce overfitting is increasing the amount of training data. This argument is supported by the results of several tests performed by Zeiler and Fergus (2014) as they achieved a higher classification accuracy by increasing the number of training images. However, test results of Nielsen (2017) also show that after providing a certain amount of training data, the classification accuracy stops increasing. Having reached this point, a further reduction of overfitting with this method is not possible. In addition to that, the size of the training set might also be limited. Therefore, reducing the size of the entire network including the depth and number of layers is presented as another approach. Unfortunately, this results in losing the power of large networks in detecting complex and detailed features. However, Nielsen (2017) also presents other techniques, called regulation techniques, that enable the reduction of overfitting without changing the amount of training data or the network size. One of them is called L2 regularisation and involves adding an extra term to the cost function. Applied on the cost function presented in Section 6.1.2, Equation 6-12 shows the added L2 term. While n represents the number of training sets, the factor λ is called regularisation parameter and can take on any positive value. This regularisation parameter is multiplied with the sum of the squares of all weights involved in the network. The aim of regularisation techniques is to make the network preferably using small weights to minimise the cost function. Even though this section does not provide any proof that this technique really reduces overfitting, Nielsen (2017) presents the corresponding proof and describes how it works in detail. Other widely used regularisation techniques are called L1 regularisation and Dropout. (Deshpande, 2017; Nielsen, 2017; Zeiler and Fergus, 2014)

$$C(w,b) = \frac{1}{2n}\sum \|y(x) - a\|^2 + \underbrace{\frac{\lambda}{2n}\sum w^2}_{\text{L2 term}} \tag{6-12}$$

Another way to increase the classification accuracy is improving the learning capability of the model. One approach to achieve that includes changing the activation function. Especially convolutional networks often use the ReLu function. ReLu stands for rectified linear unit and ensures that the values of all neurons in a feature map are positive. The related formula is shown in Equation 6-13. Regarding the training process, networks using the ReLu instead of the sigmoid function have been identified to achieve better performances. Furthermore, the training of convolutional networks has been improved significantly since GPUs were used to perform the involved computational tasks. According to Ciresan et al. (2011), especially large ConvNets benefit from the speeding up enabled by a GPU implementation. (Ciresan et al., 2011; Ujjwal, 2016; Wikipedia, 2018a)

$$relu(x) = \max(x, 0) \tag{6-13}$$

Using the softmax function as activation function in the output layer results in presenting the classification outcomes as probability distribution and thus, enables a better understanding of the them. In simple words, applying the softmax function on one output neuron provides the probability that the input data belongs to the category this neuron represents. Therefore, the softmax function transforms the original value z into the probability value softmax(z), that can take on any value between zero (lowest probability) and one (highest probability). The corresponding formula is illustrated in Equation 6-14. (Nielsen, 2017; Ujjwal, 2016)

$$softmax(z)_j = \frac{\mathrm{e}^{z_j}}{\sum_k \mathrm{e}^{z_k}} \tag{6-14}$$

As the sum of all probabilities equals one, also the sum of the values of all output neurons equals one. This condition is illustrated in Equation 6-15.

$$\sum_j softmax(z)_j = \frac{\sum_j \mathrm{e}^{z_j}}{\sum_k \mathrm{e}^{z_k}} = 1 \tag{6-15}$$

Finally, the little insight into the internal operations of a ConvNet leads to a poor understanding of opportunities to improve its classification accuracy. Therefore, Zeiler and Fergus (2014) present the visualisation and investigation of the single feature map outputs as suitable method to perform a qualitative verification whether the selected architecture is appropriate. According to the outcomes of this verification, human developers might adjust the size of filters, the size of strides, use a different pooling method or even add or remove convolutional, pooling or fully-connected layers to improve the model performance. However, having a lot of experience is a requirement to perform this method properly. Furthermore, the amount of work increases exponentially with the number of feature maps to be investigated. Therefore, trial-and-error still represents the most widely used method to improve the classification accuracy of deep convolutional networks. (Zeiler and Fergus, 2014)

6.2 Development and Implementation of Artificial Neural Networks

This section is dedicated to investigating appropriate ways to develop an artificial neural network and implement it in a CPS to enable gaining additional knowledge of an internal logistic system and thus, to realise further optimisation in intralogistics. Therefore, a theoretical use case is defined to support building a deeper understanding of specific network and CPS requirements. Nevertheless, even though this section is mainly focused on finding a solution for this use case, also possible solutions for other applications of artificial neural networks in intralogistics are demonstrated.

6.2.1 CPS and Use Case

First of all, the CPS presented in this section only represents one solution how an artificial neural network can be implemented and employed for applications in intralogistics. Developed accordingly to the conceptual model of a CPS presented in Section 2.3, the structure of the new CPS, visualised in Figure 6-8, only involves abstract components to provide a general solution. In the context of the defined use case, the CPS identifies current operating conditions of a conveyor system. This might be realised through an implemented classification model. Therefore, similar to the structure of the CPS demonstrator presented in Section 4.2.1, a webcam is used to capture a video that depicts a specific part of this conveyor system. In the next step, the video is forwarded to an edge node with the integrated classification model that uses an artificial neural network to perform image recognition as first data processing. However, it is important to mention that a proper training of such a network requires a certain computing power. Depending on the network type and size, typical edge nodes (single-board computer, PLC) do not provide a sufficient amount of this power (especially for deep networks). Therefore, the easiest solution is training the network first on a separate desktop computer. Splitting the video in single images enables its analysation through image recognition, whereas a classification algorithm detects specific features included in the images. According to these detections, the algorithm classifies the images into several categories, each representing one certain operating condition of the conveyor system. Next steps might include sending the data from the edge node to a cloud-based platform, where the data are stored and further processed. Information can be extracted by determining the timing and proportion of the identified conditions. Finally, presenting the results to human operators via different devices such as a laptop or a tablet is a possible solution to complete this CPS.

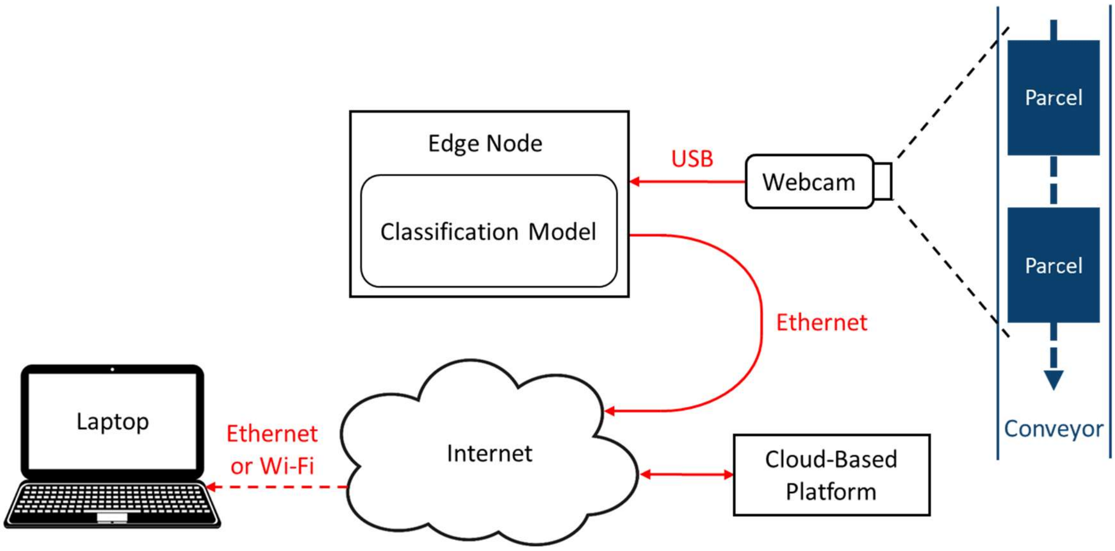


Figure 6-8: Sample Structure of a CPS with an implemented Classification Model

Based on the representative use case in intralogistics presented in Section 4.1, four typical operating conditions of the conveyor system are defined and shown in Table 4-4. Furthermore, as mentioned earlier, each operating condition represents on output class of the classification model. Having only four output classes represents a simple way to build a deeper understanding of the network requirements. However, samples of state-of-the-art ConvNets provided in the following section demonstrate that achieving a high classification accuracy is also possible by using hundreds or even thousands of output classes.

Table 6-1: Operating Conditions of the Conveyor System for the Use Case

No.	Operating Condition	Meaning
1	Idle	No parcels are transported
2	Under-Utilised	Less parcels than optimal are transported
3	Optimal-Utilised	The optimal number of parcels are transported
4	Over-Utilised	More parcels than optimal are transported

While the conveyor system is regarded as idle in cases an image contains no parcels at all, under-utilised systems are represented by images containing at least one but less parcels than the value of the parameter Op reflects. This parameter stands for the optimal number of parcels within the area of the conveyor system that is covered by the camera. Hence, images in which the value of Op exactly reflects the number of contained parcels represent an optimal-utilised conveyor system. Finally, the conveyor system is regarded as over-utilised in all other cases.

6.2.2 Investigation of existing Convolutional Neural Networks

The next step includes the investigation of existing artificial neural networks to build an understanding of state-of-the-art solutions including their structure, requirements, classification accuracy and possible improvements. Furthermore, as Zeiler and Fergus (2014) claim that ConvNets have become a requirement to realise a state-of-the-art performance in image recognition, the following investigation is only focused on this type. Ujjwal (2016) presents some of the recently published convolutional neural networks:

1. AlexNet (2012)
2. ZF Net (2013)
3. GoogLeNet (2014)
4. VGGNet (2014)
5. ResNets (2015)
6. DenseNet (August 2016)

Among these six ConvNets, the AlexNet was the first that achieved outstanding results in image recognition. Therefore, this convolutional network was investigated in detail. The AlexNet represents a large and deep convolutional neural network that includes about 650 thousand neurons. Its structure is illustrated in Figure 6-9. Typical input data are images in the RGB format with a resolution of 224x224 pixels. Hence, the input layer comprises three matrixes (one for red, green and blue respectively), each containing 50176 (=224x224) neurons. The hidden layers start with five convolutional layers, whereas only some of them are followed by max-pooling layers. Furthermore, three fully-connected layers complete the hidden layers. All convolutional and fully-connected layers use the ReLu function for the involved neurons. Furthermore, Krizhevsky et all. (2012) followed a different approach to create the pooling layers. As described in the previous section, the value of a neuron in a pooling matrix is determined by summarising a specific region in the related feature map of the previous convolutional layer. While these regions typically not overlap, the AlexNet accepts an overlapping of one horizontal line of neurons. Finally, the output layer comprises 1000 neurons, each representing one image category. Moreover, the output neurons use the softmax function as activation function. This results in a total number of about 60 million parameters. However, the size of the network was mainly limited by the available memory. In order to enable a proper training, two GTX 580 3GB GPUs were used. Nevertheless, Krizhevsky et all. (2012) argue that the model performance achieved in 2012 can be improved further by simply using faster GPUs and bigger training sets. As such large networks are relatively vulnerable to overfitting, two techniques to reduce its occurrence were applied. First, Krizhevsky et all. (2012) artificially enlarged the training set through label-preserving transformations. The second technique applied to reduce overfitting is the dropout regularisation method, implemented in the first two fully-connected layers of the AlexNet. In order to obtain a more detailed description of the AlexNet, it is recommended to read the original paper that is available for free. (Krizhevsky et all., 2012)

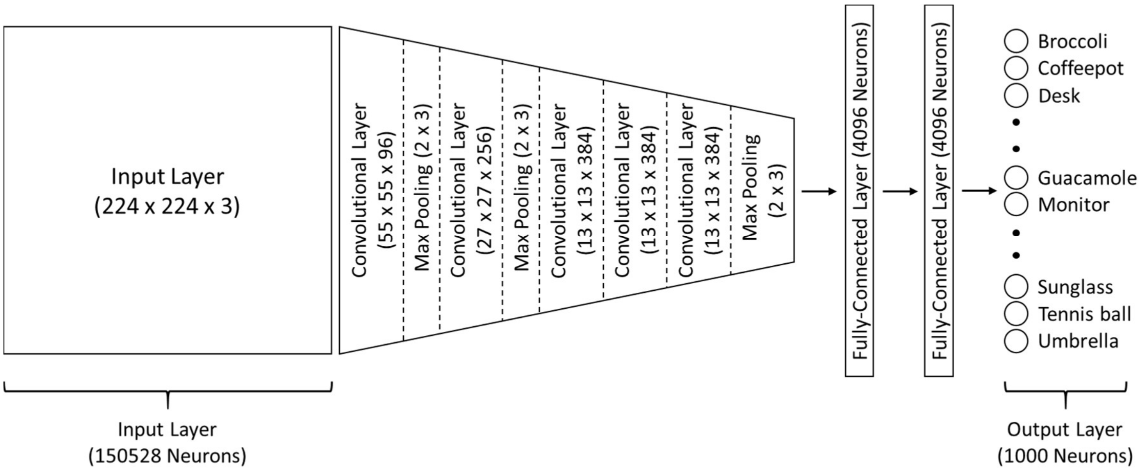


Figure 6-9: Structure of the AlexNet (adapted from Krizhevsky et all., 2012)

Having created the AlexNet, Krizhevsky et all. (2012) applied it on the ImageNet Large-Scale Visual Recognition Challenge (ImageNet LSVRC) in 2010 and 2012. ImageNet is one of the several datasets that support research projects in the field of machine learning by providing about 15 million labelled images of about 22 thousand categories (ImageNet, 2016). However, the ImageNet LSVRC contests only use subsets of the entire ImageNet dataset. For example, the ImageNet LSVRC-2012 contest comprises 1.3 million images of 1000 different categories, whereas 50 thousand images are part of the validation set and 100 thousand of the test set. Incidentally, these 1000 categories resulted in the 1000 output neurons. The training was performed using a stochastic gradient descent, whereas one batch contained 128 images. The initial weight values in each layer were determined through a zero-mean Gaussian distribution with a standard deviation of 0.01. In contrast, the initial values of the biases in the second, fourth, and fifth convolutional layer as well as in all fully-connected layers were set to one as this accelerated the learning with the ReLu function. All other biases were initially set to zero. While the initial value of the learning rate was 0.01, it was manually adjusted three times during the training and validation process. Having finished the training, the AlexNet was ready to be applied on the test data. In this case, typical units to express the classification accuracy of an artificial neural network are the top-1 and top-5 error. Keeping in mind that one image belongs to only one of the 1000 different categories, using the softmax function in the output layer provides a probability distribution over all classes. While the top-1 error refers to whether the class with the highest probability is the correct category, the top-5 error refers to whether the correct category is among the five classes with the highest probability. Applying the AlexNet on the test data of the ImageNet LSVRC-2010, a top-1 error of 37.5% and a top-5 error of 17.0% was achieved. Applying the same ConvNet on the test data of the ImageNet LSVRC-2012 resulted in a top-1 error of 40.7% and a top-5 error of 18.2%. Finally, through combining multiple models, Krizhevsky et all. (2012) achieved by far the lowest top-5 error of 15.3%. In comparison, the second-best result on the ImageNet LSVRC-2012 test data achieved a top-5 error of 26.2%. Like Zeiler and Fergus (2014), also Krizhevsky et all. (2012) noticed that removing only one convolutional layer results in a decreased classification accuracy. This underlines the importance of having the correct overall network depth to achieve the best results. (Krizhevsky et all., 2012)

Furthermore, Zeiler and Fergus (2014) improved the AlexNet by using the previously mentioned method of visualising and investigating the single feature map outputs. Taking the AlexNet performance on the ImageNet LSVRC-2012 test data as reference, Zeiler and Fergus (2014) realised a top-5 error of 14.8% through combining multiple models and thus, decreased this error by 0.5%. Furthermore, trained on ImageNet, Zeiler and Fergus (2014) applied this model on the test data of different image datasets to investigate its generalisation ability. While achieving good performances on the datasets Caltech-101 and Caltech-256 that provide images similar to ImageNet, the application on different images provided by PASCAL VOC 2012 resulted in a relatively poor performance. This result demonstrates the importance of having training data that is similar to the actual test data. (Zeiler and Fergus, 2014)

6.2.3 Development of a Convolutional Neural Network

Having understood state-of-the-art solutions, the next step is focused on developing an appropriate ConvNet structure for the requirements of the defined use case as well as creating this model to enable its implementation in the CPS.

Deep Learning Framework:

First of all, suitable machine learning frameworks to create a ConvNet must be identified. An additional investigation of related literature resulted in Table 6-2, that provides an overview of eight widely used frameworks. However, many other machine learning frameworks are available, whereas each has specific advantages and disadvantages. Therefore, identifying the most suitable framework strongly depends on the type and the size of the neural network, the available hardware as well as the experience of the network developer. Among the eight investigated frameworks, only MATLAB and the additionally required Deep Learning Toolbox are not offered as open source. Furthermore, it is important to mention that the Deep Learning Toolbox, formerly called Neural Network Toolbox, was recently introduced for the MATLAB version 2018b. Besides that, while all other presented frameworks directly use CUDA code to enable performing computational tasks on GPUs, only MATLAB additionally requires the Parallel Computing Toolbox. However, this toolbox combined with the GPU Coder enables generating the required CUDA code to use CUDA-enabled Nvidia GPUs. (Nvidia, 2018; Makadia, 2018; MathWorks, 2018a; Wikipedia, 2018b)

Table 6-2: Widely used Machine Learning Frameworks

No.	Framework	Open Source	Language	GPU Computing
1	Caffe	Yes	C++	CUDA
2	Microsoft Cognitive Toolkit	Yes	C++	CUDA
3	MATLAB + Deep Learning Toolbox	No	C, C++, Java, MATLAB	Parallel Computing Toolbox
4	Apache MXNet	Yes	C++	CUDA
5	PyTorch	Yes	Python, C	CUDA
6	TensorFlow	Yes	C++, Python,	CUDA
7	Keras	Yes	Python	CUDA
8	Deeplearning4j	Yes	C++, Java	CUDA

Even though the identification of the most suitable machine learning framework requires a comprehensive investigation of all options, MATLAB is chosen due to the existing experience gained during the development of the CPS demonstrator in Chapter 4. Furthermore, Nvidia (2018) describes MATLAB as very effective framework for engineers that provides powerful tools to create models with only few lines of code. While creating a ConvNet in MATLAB requires the Deep Learning Toolbox, simple machine learning applications only need the Statistics and Machine Learning Toolbox. Having chosen a suitable machine learning framework, the next step includes developing the actual network structure. Therefore, the following three basic approaches to develop a ConvNet classification model are investigated: (Nvidia, 2018, MathWorks, 2018a; MathWorks, 2018b)

1. Pretrained models
2. Retrained models
3. New models

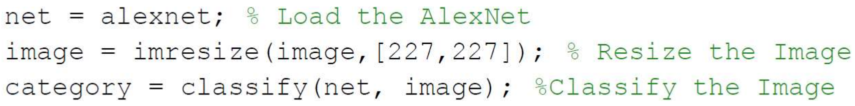
Pretrained Models:

Using a pretrained model represents the simplest solution to implement a classification model in a CPS. As such a network has already learned to extract and use powerful features to classify images, no additional training is necessary. This results in little computing requirements as well as in significant time savings. Furthermore, the pretrained model can be directly used as the actual classification model in the edge node, whereas only the specific memory space for the model itself is required. MATLAB provides eleven pretrained models as shown in Table 6-3. This table also provides additional information of each model including the number of hidden layers, the memory size and the number of involved parameters. Taking a closer look at the memory size and the number of parameters, it is obvious that they are linked to each other. Even though all models are trained on the same subset (more than one million images) of the ImageNet dataset, each has its own characteristics. Some of the most important characteristics are the classification accuracy and the classification time. Comparing all pretrained models provided by MathWorks (2018b) demonstrates that an increasing classification accuracy results in an increasing classification time. Furthermore, in cases of limited hardware, also the memory size represents an important selection criterion. As all models are trained on images from the ImageNet dataset, each of them generally provides a high level of generalisation when classifying natural images. However, trying different models on the actual images of the new task represents a simple but effective way to identify the most suitable model. (MathWorks, 2018c)

Table 6-3: Pretrained Models provided by MATLAB

No.	Model	Hidden Layers	Size	Parameters
1	AlexNet	8	227 MB	61 million
2	VGG-16	16	515 MB	138 million
3	VGG-19	19	535 MB	144 million
4	SqueezeNet	18	4.6 MB	1.24 million
5	GoogLeNet	22	27 MB	7 million
6	Inception-v3	48	89 MB	23.9 million
7	DenseNet-201	201	77 MB	20 million
8	ResNet-18	18	44 MB	11.7 million
9	ResNet-50	50	96 MB	25.6 million
10	ResNet-101	101	167 MB	44.6 million
11	Inception-ResNet-v2	164	209 MB	55.9 million

The following example demonstrates how a pretrained AlexNet might be used as classification model in the edge node (e.g. raspberry pi) of the CPS. In case an image is provided, using the AlexNet to classify it into one of the 1000 available categories only requires three lines of code in MATLAB. A sample code is illustrated in Figure 6-10. As the AlexNet in MATLAB requires a resolution of 227x227 pixels, resizing the image ensures the correct condition of the input data. Finally, the AlexNet outputs the most likely category of the image. (MathWorks, 2018d)



```
net = alexnet; % Load the AlexNet
image = imresize(image,[227,227]); % Resize the Image
category = classify(net, image); %Classify the Image
```

Figure 6-10: Classifying an Image via AlexNet in MATLAB

Despite the advantages of being simple and fast, using a pretrained model directly as classification model in a CPS does typically not represent a suitable solution for applications in intralogistics. This is mainly caused by the predetermined categories which limit possible applications of the model significantly. Regarding the defined use case, this approach is completely inappropriate as the four required categories are unavailable. Furthermore, the exceeding categories cause additional memory requirements that might be saved with an alternative solution.

Retrained Models:

Having identified that using a pretrained model directly as the classification model in the edge node of the CPS is an inappropriate solution for the defined use case, using a retrained model represents another approach. Therefore, starting with a pretrained model, a technique called transfer learning involves using some of its layers to develop a new model that has to be retrained. While these reused layers start from the input layer and might include some of the hidden layers (early layers are focused on general features), the new hidden layers and the new output layer can be focused on the specific requirements of the new task (later layers are focused on detailed features). In case the reused layers are trained on images that are similar to the images involved in the new task, these layers have already learned to extract the required features. Hence, the reused layers only need a fine-tuning. However, the new layers need a proper training. Furthermore, while training completely new models from the scratch requires a large training set, a significantly smaller amount of training data is needed for transfer learning. Therefore, this method enables a simple and fast training that results in little computing requirements and time savings. However, in contrast to using only a pretrained model, a training is necessary. The required size of the new training set mainly depends on the proportion of reused and new layers. Quite simply, the more layers are reused from pretrained models, the less additional training is required. The basic concept of this technique is illustrated in Figure 6-11, whereas blue neurons represent reused layers and orange neurons represent new layers. (MathWorks, 2018e)

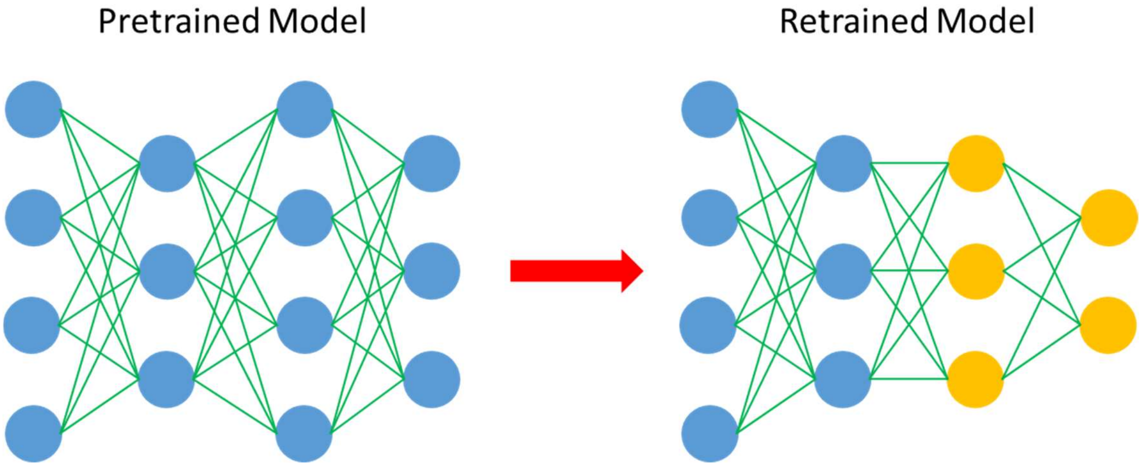


Figure 6-11: Basic Concept of Transfer Learning

A specific form of transfer learning is called feature extraction. This technique includes only the output layer being adjusted to the requirements of a new task. All other layers including the input layer and the hidden layers are reused. Hence, this technique requires the smallest training set. However, as almost all layers of such a model are trained on the original data, it typically provides a smaller classification accuracy on a new task than models created with transfer learning. Therefore, this technique is only recommended in cases the new training set is limited (about 20 images per class) and the images involved in the new task are similar the original training images. (MathWorks, 2018g)

Focusing on the requirements of the defined use case, one possible ConvNet structure developed with transfer learning and based on a pretrained AlexNet is illustrated in Figure 6-12. While the blue area represents the reused layers, the orange area represents the new layer. The presented structure shows that all layers of the AlexNet except the output layer are reused in the developed network as described in the feature extraction technique. However, the new output layer only consists of four neurons, each representing one of the four operating conditions of the conveyor system defined in the use case. As before, the output neurons use the softmax function to provide a probability distribution.

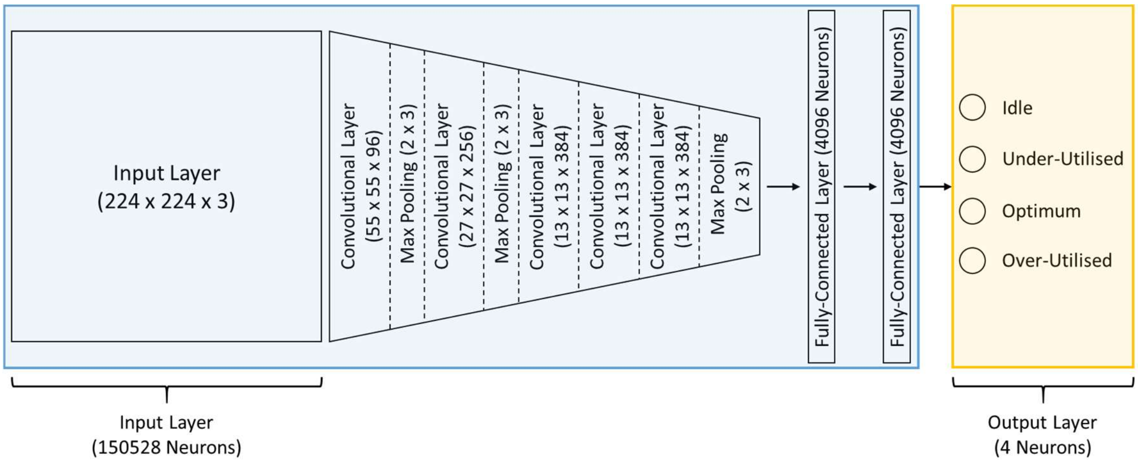
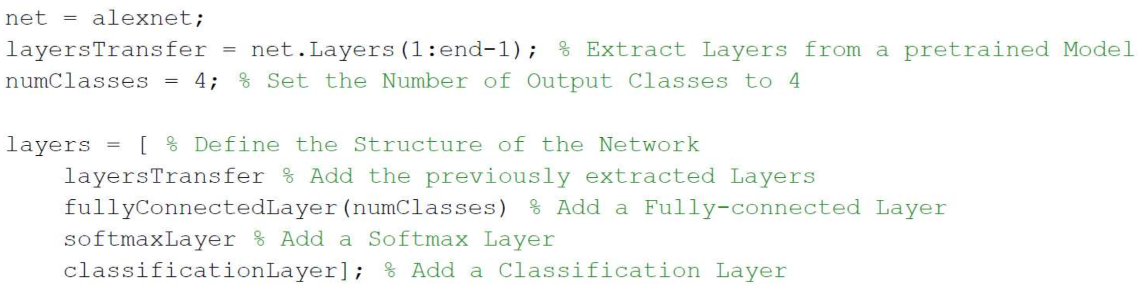


Figure 6-12: Sample Structure of a ConvNet developed via Transfer Learning

Based on this developed ConvNet structure, Figure 6-13 illustrates one possible solution to create the network in MATLAB. Having loaded the pretrained AlexNet, the next step is extracting the layers that are supposed to be reused in the retrained model. The sample code in the figure shows the extraction of all layers except the last one (output layer). Setting the number of output classes to four represents adjusting the new network to the specific requirements of the use case. Finally, starting with the previously extracted layers, the structure is defined by adding one or more new layers. Even though adding only one new output layer is expected, the realisation of an appropriate output including four classes and the softmax function in MATLAB requires adding three different layers. This includes a fully-connected layer with four neurons, a softmax layer and a classification layer. While the number of neuron in the fully-connected layer determines the number of neurons in the classification layer, the softmax layer is required to output a probability distribution. It is also important to mention that the performance of a proper training of the created network is not included in the provided code and thus, still required to complete the retrained model successfully. However, retraining as well as training from the scratch involve the same steps. Therefore, the provided description of training a created ConvNet in MATLAB for the next approach is also valid for this one. (MathWorks, 2018f)



```
net = alexnet;
layersTransfer = net.Layers(1:end-1); % Extract Layers from a pretrained Model
numClasses = 4; % Set the Number of Output Classes to 4

layers = [ % Define the Structure of the Network
    layersTransfer % Add the previously extracted Layers
    fullyConnectedLayer(numClasses) % Add a Fully-connected Layer
    softmaxLayer % Add a Softmax Layer
    classificationLayer]; % Add a Classification Layer
```

Figure 6-13: Defining a ConvNet Structure via Tranfer Learning in MATLAB

Using a retrained model as classification model represents a suitable solution for applications in intralogistics in cases the images involved in the new task are similar to the original training images. However, as no images containing a conveyor system are part of the original training set, it is expected that the images involved in the defined use case are rather different. In this case, MathWorks (2018c) argues that retraining a model created through transfer learning might take as long as training a completely new model from the scratch while providing a smaller classification accuracy. Moreover, most of the network structure including the input layer and all hidden layers are not developed for the specific requirements of the use case. While the AlexNet was developed to differ between 1000 categories, the investigation of past research projects has pointed out that the structure of an appropriate ConvNet for the defined use case can be much simpler and smaller. Consequently, the exceeding categories also cause additional memory requirements that might be saved by creating a new model only focused on the actual use case requirements. This leads to the conclusion that developing a completely new model provides more benefits.

New Model:

The last approach to develop a ConvNet that might be used as classification model in the edge node of the CPS represents creating a completely new network, whereas its entire structure is focused on the specific requirements of the defined use case. Therefore, understanding the structure of ConvNets that have already proven a high classification accuracy supports the development process significantly. Taking a detailed look at the network requirements, similarities to the ConvNet developed by Nielsen (2017) to recognise images of handwritten digits between 0 and 9 were identified. Hence, Nielsen's ConvNet built the basis of the developed ConvNet for the actual use case. Especially the achieved classification accuracy of 99.6% and the similar number of output neurons had a considerable influence on this decision. While the final structure of the developed network is illustrated in Figure 6-14, it is important to mention that it only represents one possible solution. Assuming that this model is running on a raspberry pi, a typical image size of the Video Capture Block (RGB) from the Simulink Support Package for Raspberry Pi Hardware is 160x120 pixels. Besides that, each image provided by this block involves three matrixes due to the RGB format. Having these images as input data results in an input layer that contains 57600 (=160x120x3) neurons. Furthermore, 20 filters covering a receptive field of 5x5 are applied on each input matrix. This results in 60 feature maps in the first convolutional layer. Subsequently, 2x2 regions of each feature map are summarised via max pooling. As large networks are capable of detecting more complex and detailed features, using another convolutional layer improves the classification accuracy further. Applying two 5x5 filters on the 60 pooling matrixes of the first pooling layer results in 120 feature maps in the second convolutional layer. Furthermore, also max pooling is used to summarise 2x2 regions of these feature maps. Having created the second pooling layer, two fully-connected layers, each containing 1000 neurons, are also added and realise the final improvement of the classification accuracy. Finally, four output neurons present the identified operating condition of the conveyor system as result of the image recognition. While the ReLu function is used for all neurons involved in the convolutional and fully-connected layers, the output neurons use the softmax function.

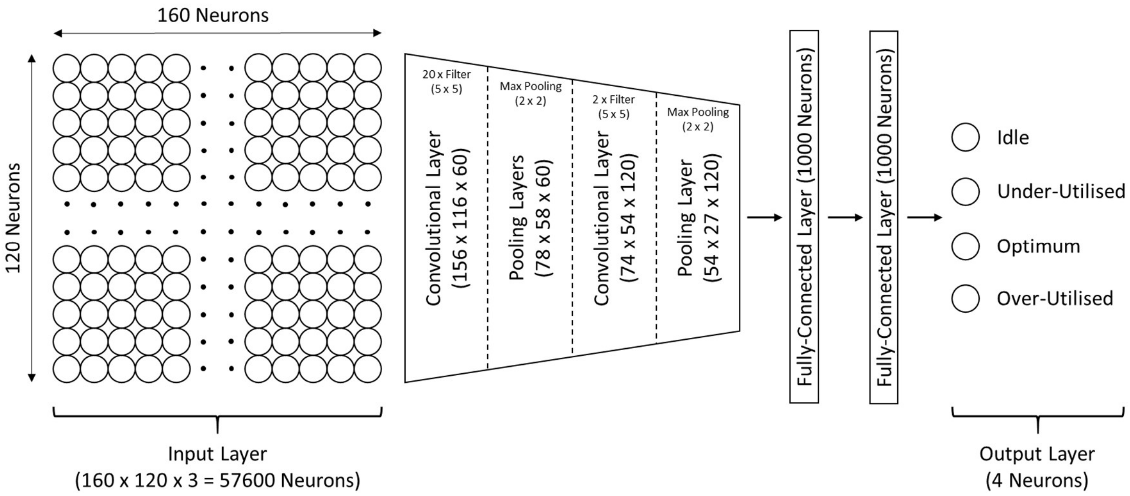
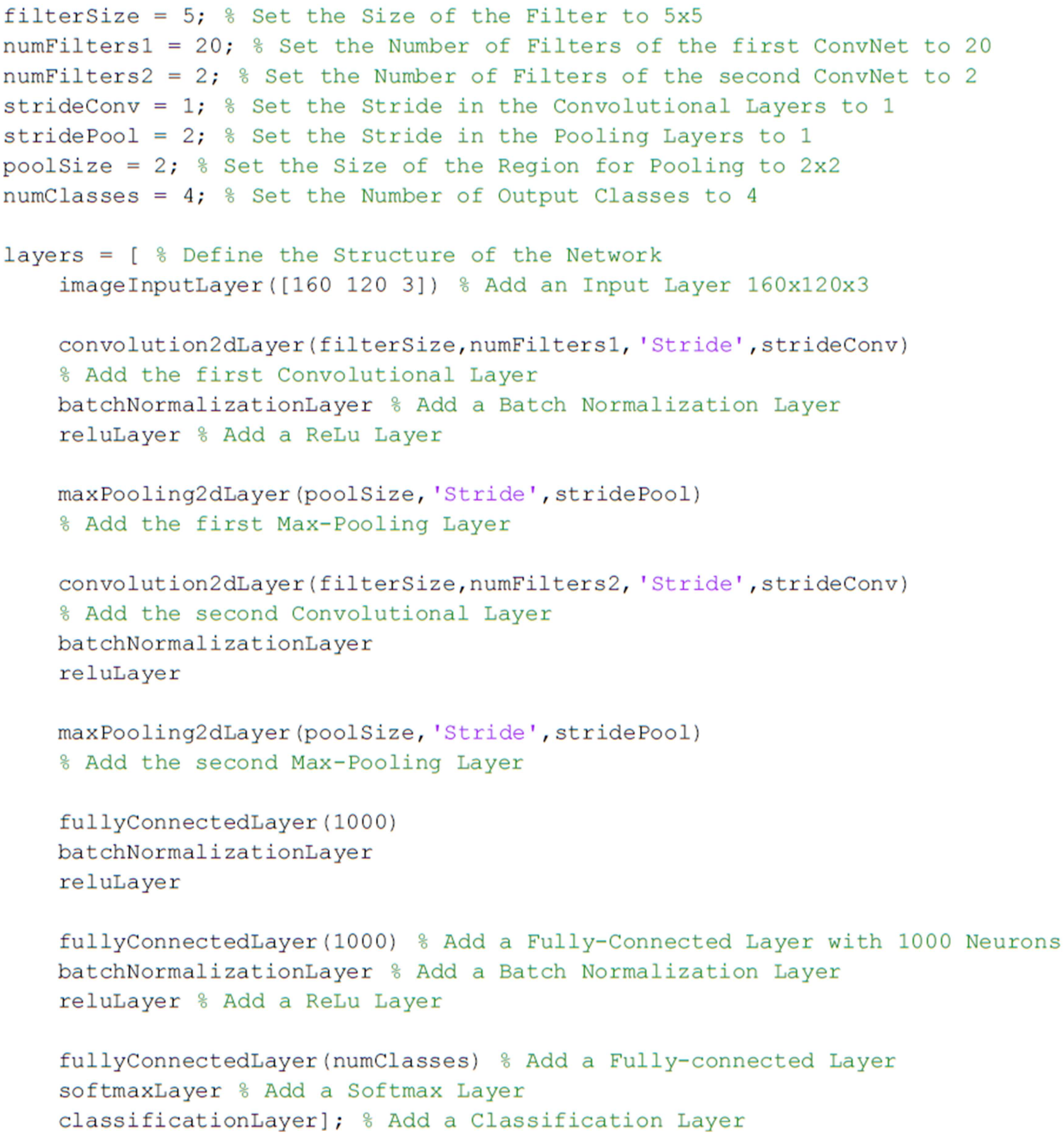

160 Neurons
120 Neurons
20 x Filter (5 x 5)
Max Pooling (2 x 2)
2 x Filter (5 x 5)
Max Pooling (2 x 2)
Convolutional Layer (156 x 116 x 60)
Pooling Layers (78 x 58 x 60)
Convolutional Layer (74 x 54 x 120)
Pooling Layer (54 x 27 x 120)
Fully-Connected Layer (1000 Neurons)
Fully-Connected Layer (1000 Neurons)
Idle
Under-Utilised
Optimum
Over-Utilised
Input Layer (160 x 120 x 3 = 57600 Neurons)
Output Layer (4 Neurons)

Figure 6-14: Developed Structure of a ConvNet for the defined Use Case

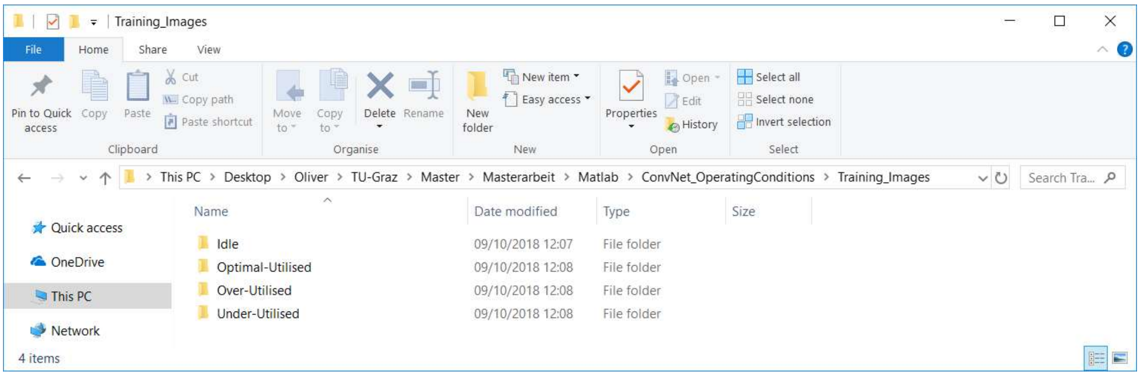
Based on this developed ConvNet structure, Figure 6-15 illustrates one possible solution to create the network in MATLAB. The first step includes setting the values of some hyperparameters. Starting with the size and the number of filters to create the feature maps in the convolutional layers (both convolutional layers employ the same filter size), the value of the filter stride is set to one. As this represents the smallest number of pixels a filter can be moved at a new step, it supports detecting more features in detail. However, to avoid overlapping of the summarised regions of the feature maps via max pooling, the pooling stride is set to two. Even though other hyperparameters such as padding, the learning rate and more can also be adjusted, they are not explicitly set as their default values are already appropriate for the requirements of the use case. Having set the size of the region for max pooling as well as the number of output classes, the following step includes defining the layers of the ConvNet. As shown in the developed structure, the input layer comprises 160x120x3 neurons. Right after the input layer, the first convolutional layer is employed. In order to use the ReLu function for all neurons involved in this convolutional layer, adding a ReLu layer is required in MATLAB. Furthermore, MathWorks (2018h) recommends putting a batch normalization layer between the convolutional and ReLu layer to improve the training process. Subsequently, the first max-poling layer is added. Following the developed ConvNet structure, another convolutional, batch normalization, ReLu and max-pooling layer are included. The hidden layers are completed by adding two fully-connected layers, each containing 1000 neurons. Furthermore, adding a batch normalization and ReLu layer is also necessary to use the ReLu function for all involved neurons properly. Finally, the realisation of an appropriate output including four classes and the softmax function in MATLAB requires adding a fully-connected, a softmax and a classification layer. (MathWorks, 2018h)



```
filterSize = 5; % Set the Size of the Filter to 5x5
numFilters1 = 20; % Set the Number of Filters of the first ConvNet to 20
numFilters2 = 2; % Set the Number of Filters of the second ConvNet to 2
strideConv = 1; % Set the Stride in the Convolutional Layers to 1
stridePool = 2; % Set the Stride in the Pooling Layers to 1
poolSize = 2; % Set the Size of the Region for Pooling to 2x2
numClasses = 4; % Set the Number of Output Classes to 4

layers = [ % Define the Structure of the Network
    imageInputLayer([160 120 3]) % Add an Input Layer 160x120x3

    convolution2dLayer(filterSize,numFilters1,'Stride',strideConv)
    % Add the first Convolutional Layer
    batchNormalizationLayer % Add a Batch Normalization Layer
    reluLayer % Add a ReLu Layer

    maxPooling2dLayer(poolSize,'Stride',stridePool)
    % Add the first Max-Pooling Layer

    convolution2dLayer(filterSize,numFilters2,'Stride',strideConv)
    % Add the second Convolutional Layer
    batchNormalizationLayer
    reluLayer

    maxPooling2dLayer(poolSize,'Stride',stridePool)
    % Add the second Max-Pooling Layer

    fullyConnectedLayer(1000)
    batchNormalizationLayer
    reluLayer

    fullyConnectedLayer(1000) % Add a Fully-Connected Layer with 1000 Neurons
    batchNormalizationLayer % Add a Batch Normalization Layer
    reluLayer % Add a ReLu Layer

    fullyConnectedLayer(numClasses) % Add a Fully-connected Layer
    softmaxLayer % Add a Softmax Layer
    classificationLayer]; % Add a Classification Layer
```

Figure 6-15: Defining a ConvNet Structure in MATLAB

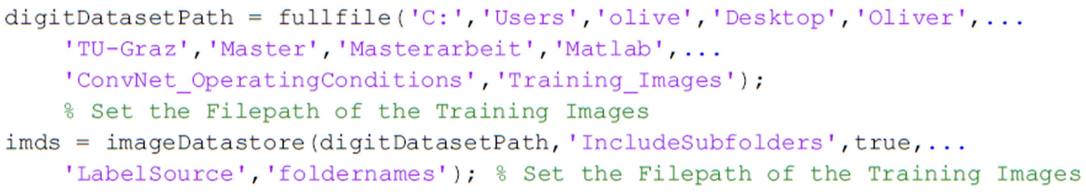
Before this network can be trained, the required training images depicting parcels on the conveyor system to simulate the four operating conditions must be prepared to enable supervised learning. Referring to the training data used for the AlexNet, about 1000 images per category represents a sufficient amount to perform a proper training. This results in 1000x4 = 4000 images for the defined use case. Furthermore, the input layer of the created ConvNet requires images in the RGB format and a resolution of 160x120. Even though no training images were available at this point, the following paragraphs describe the theoretical next steps assuming 1000 images for each category are provided. In order to load them into MATLAB, a folder called Training_Images was created. This folder contains four subfolders as visualised in Figure 6-16, whereas each subfolder represents one category the created ConvNet is supposed to learn. Therefore, each subfolder must contain the related training images.



Name	Date modified	Type	Size
Idle	09/10/2018 12:07	File folder	
Optimal-Utilised	09/10/2018 12:08	File folder	
Over-Utilised	09/10/2018 12:08	File folder	
Under-Utilised	09/10/2018 12:08	File folder	

Figure 6-16: Categories as Subfolders

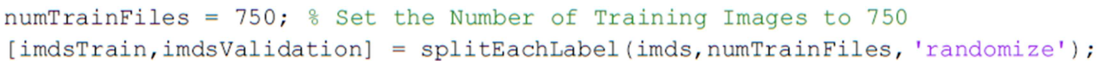
Figure 6-17 shows a code that might be used to load the images from these subfolders into MATLAB. While the first command (fullfile) converts the folder path into the required format, the second command (imageDatastore) loads the images from this path including all subfolders and saves them as an image datastore object. Furthermore, imageDatastore automatically labels each image based on the name of the related subfolder. (MathWorks, 2018h)



```
digitDatasetPath = fullfile('C:','Users','olive','Desktop','Oliver',...
    'TU-Graz','Master','Masterarbeit','Matlab',...
    'ConvNet_OperatingConditions','Training_Images');
    % Set the Filepath of the Training Images
imds = imageDatastore(digitDatasetPath,'IncludeSubfolders',true,...
    'LabelSource','foldernames'); % Set the Filepath of the Training Images
```

Figure 6-17: Loading Images labelled with the related Categories in MATLAB

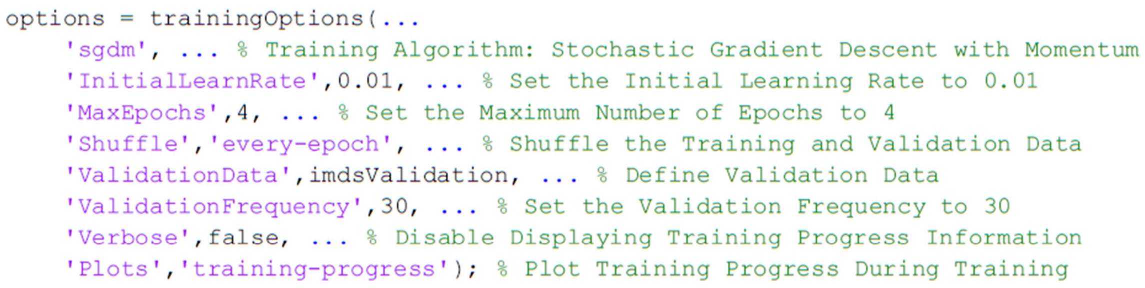
In the following step, the loaded images are split into a training and validation set through the MATLAB code illustrated in Figure 6-18. Assuming that 1000 images are available per category, 750 of them are randomly allocated to the training set and 250 to the validation set.



```
numTrainFiles = 750; % Set the Number of Training Images to 750
[imdsTrain,imdsValidation] = splitEachLabel(imds,numTrainFiles,'randomize');
```

Figure 6-18: Preparing Training and Validation Sets in MATLAB

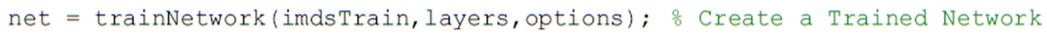
The final step before the actual training can be performed includes specifying the training options. While Figure 6-19 illustrates a possible MATLAB code, the same values and options are set in the example provided by MathWorks (2018h). Starting with defining the stochastic gradient descent with momentum as the used training algorithm, the initial training rate is set to 0.01. Furthermore, four epochs (one epoch represents using the entire training data once) are involved, whereas the training data is shuffled before each epoch and the validation data before each validation. Therefore, also the validation data must be defined. Setting the validation frequency to 30 represents that a validation is performed every 30th iteration. Even though displaying the detailed training information is disabled, the training progress including the classification accuracy and outcome of the cost function will be visualised in real time. (MathWorks, 2018h)



```
options = trainingOptions(...
    'sgdm', ... % Training Algorithm: Stochastic Gradient Descent with Momentum
    'InitialLearnRate',0.01, ... % Set the Initial Learning Rate to 0.01
    'MaxEpochs',4, ... % Set the Maximum Number of Epochs to 4
    'Shuffle','every-epoch', ... % Shuffle the Training and Validation Data
    'ValidationData',imdsValidation, ... % Define Validation Data
    'ValidationFrequency',30, ... % Set the Validation Frequency to 30
    'Verbose',false, ... % Disable Displaying Training Progress Information
    'Plots','training-progress'); % Plot Training Progress During Training
```

Figure 6-19: Specifying Training Options in MATLAB

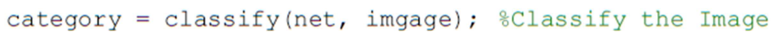
Having loaded the labelled images, prepared the training and validation sets as well as specified the training options, the actual training of the created ConvNet can be performed. As Figure 6-20 illustrates, this is executed in MATLAB with only one code line and returns a trained ConvNet focused on the specific requirements of the use case.



```
net = trainNetwork(imdsTrain,layers,options); % Create a Trained Network
```

Figure 6-20: Creating a trained Network in MATLAB

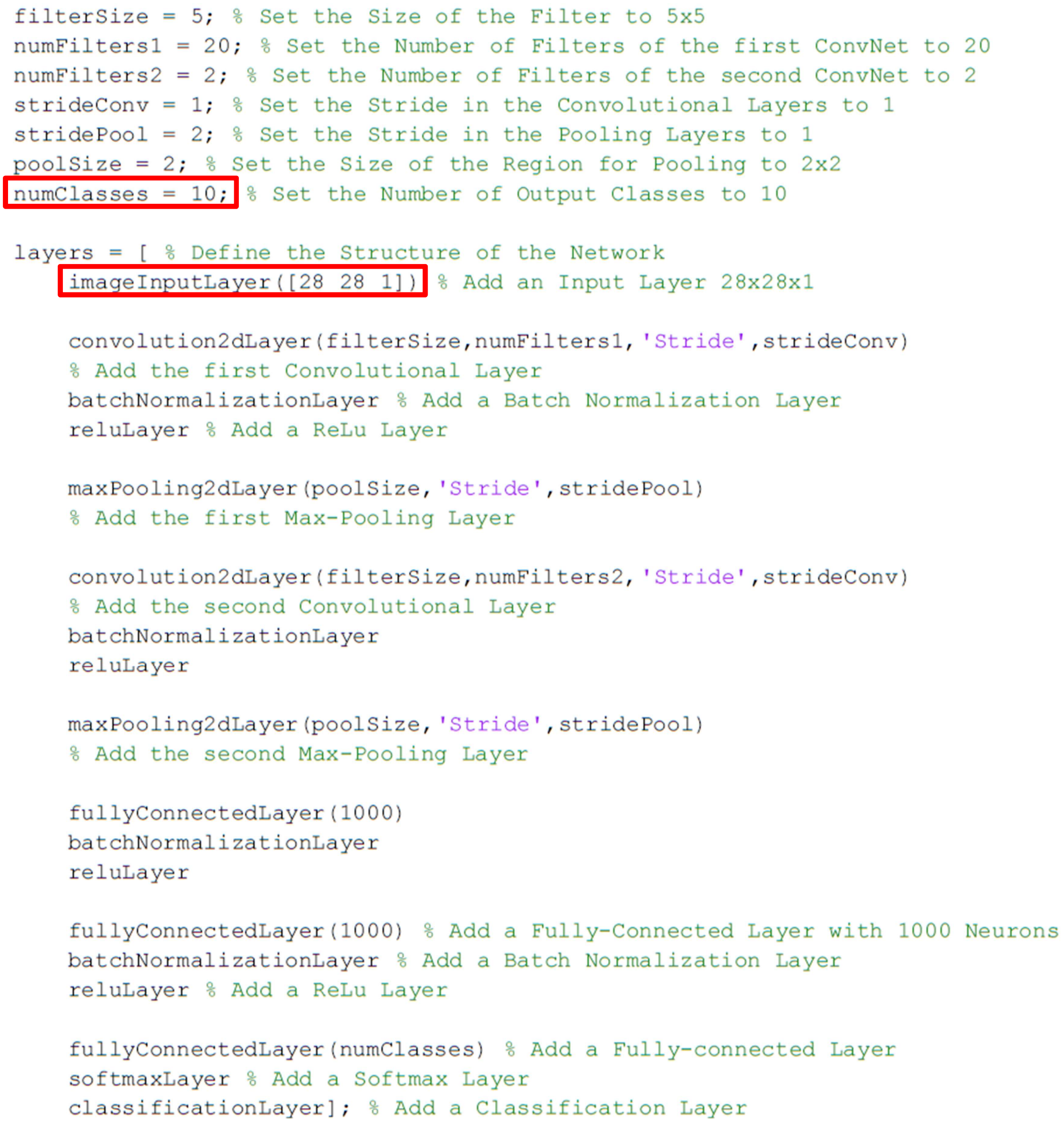
In case a new image in the RGB format and a resolution of 160x120 is provided, using this trained ConvNet in MATLAB to classify it into one of the four available categories might be realised with the code shown in Figure 6-21, whereas its executions outputs the most likely category of the image.



```
category = classify(net, imgage); %Classify the Image
```

Figure 6-21: Classifying an Image via the created ConvNet in MATLAB

Finally, determining the classification accuracy of the trained ConvNet ensures its suitability for an integration into the edge node (e.g. raspberry pi) of the CPS. As the required images depicting the operating conditions to perform an appropriate verification were unavailable at this point, the developed ConvNet structure was applied on a similar task that involved recognising images of handwritten digits between 0 and 9. Therefore, MATLAB provides a dataset that comprises 10000 training images, whereas each digit is depicted by 1000. Furthermore, this required minor changes on the MATLAB code to create the related ConvNet as requirements of the original use case and the new task differ in two points. This new code including the changes marked with red rectangles is illustrated in Figure 6-22. First, instead of using images in the RGB format and a resolution of 160x120, the MATLAB dataset provides greyscale images with a resolution of 28x28. This resulted in changing the parameters of the input layer. Furthermore, instead of using four output classes, the range of digits between 0 and 9 requires ten classes. However, the main structure of the ConvNet including the convolutional, pooling and fully-connected layers as well as all other parameters involved in splitting the images into a training and validation set or in specifying the training options remained unchanged. Due to the fundamental similarities between the original and the new ConvNet, it is assumed that the test result of recognising handwritten digits is also representative for identifying the operating conditions of the conveyor system.



```
filterSize = 5; % Set the Size of the Filter to 5x5
numFilters1 = 20; % Set the Number of Filters of the first ConvNet to 20
numFilters2 = 2; % Set the Number of Filters of the second ConvNet to 2
strideConv = 1; % Set the Stride in the Convolutional Layers to 1
stridePool = 2; % Set the Stride in the Pooling Layers to 1
poolSize = 2; % Set the Size of the Region for Pooling to 2x2
numClasses = 10; % Set the Number of Output Classes to 10

layers = [ % Define the Structure of the Network
    imageInputLayer([28 28 1]) % Add an Input Layer 28x28x1

    convolution2dLayer(filterSize,numFilters1,'Stride',strideConv)
    % Add the first Convolutional Layer
    batchNormalizationLayer % Add a Batch Normalization Layer
    reluLayer % Add a ReLu Layer

    maxPooling2dLayer(poolSize,'Stride',stridePool)
    % Add the first Max-Pooling Layer

    convolution2dLayer(filterSize,numFilters2,'Stride',strideConv)
    % Add the second Convolutional Layer
    batchNormalizationLayer
    reluLayer

    maxPooling2dLayer(poolSize,'Stride',stridePool)
    % Add the second Max-Pooling Layer

    fullyConnectedLayer(1000)
    batchNormalizationLayer
    reluLayer

    fullyConnectedLayer(1000) % Add a Fully-Connected Layer with 1000 Neurons
    batchNormalizationLayer % Add a Batch Normalization Layer
    reluLayer % Add a ReLu Layer

    fullyConnectedLayer(numClasses) % Add a Fully-connected Layer
    softmaxLayer % Add a Softmax Layer
    classificationLayer]; % Add a Classification Layer
```

Figure 6-22: Changes of the defined ConvNet Structure to recognise handwritten Digits in MATLAB

Having created this new ConvNet focused on the requirements to recognise handwritten digits in MATLAB, it was trained on the available 10000 images. As specified in the training options, the training progress including the classification accuracy and outcome of the cost function was visualised in real time. Figure 6-23 illustrates the related profile of the classification accuracy. While four epochs and 232 iterations were used, the figure also shows that a validation was performed every 30th iteration. Furthermore, starting from a relatively low classification accuracy, the profile reveals that it increased the fastest at the beginning of the training in epoch one. While further improvements of the classification accuracy decreased with the number of iterations, no improvements at all can be observed in epoch four. This proves that the training was completed successfully. Besides that, achieving a final classification accuracy of 99.84% demonstrates that the developed ConvNet structure is a suitable solution to perform the recognition of handwritten digits. As mentioned earlier, due to the fundamental similarities between the two ConvNets, it is assumed that this test result is also representative for the identification of the operating conditions of the conveyor system. Therefore, the CPS presented in Section 6.2.1 can be completed by integrating the created ConvNet focused on the requirements of the use case into the edge node. However, a proper training of this model must be performed first. Furthermore, due to the relatively small size of the network, it is expected that only little computing power is required. Therefore, the realisation of the edge node as single-board computer (e.g. raspberry pi) is recommended, whereas an additional MATLAB code might be used to send the identified operating conditions to a ThingSpeak channel. This enables processing the stored data as well as visualising the extracted information. The related MATLAB code of the ConvNet to recognise handwritten digits is provided in Appendix F.

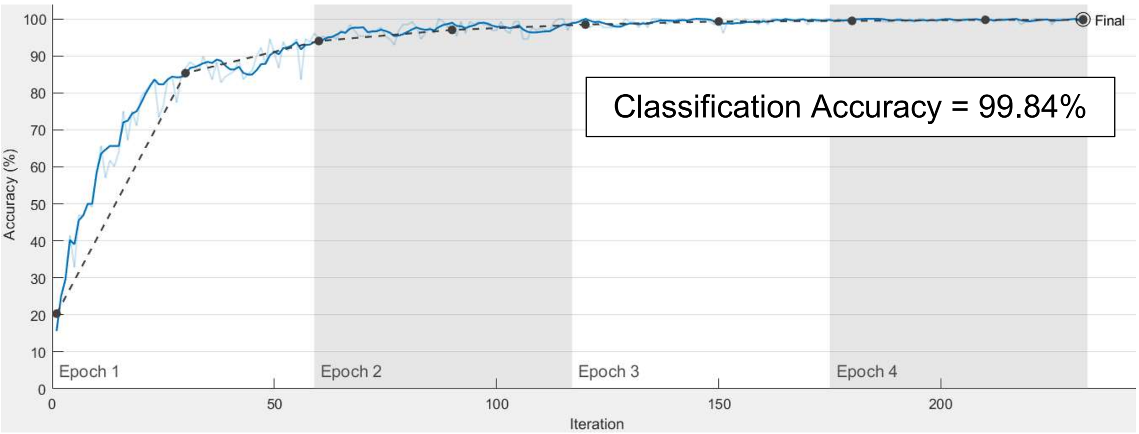


Figure 6-23: Training Progress of the created ConvNet – Classification Accuracy

7 Discussion of the Results

This chapter is dedicated to discussing the results and key findings of the project.

Starting with the literature review, it was used to identify the research gaps of this project, whereas all investigated areas provided the required background knowledge to realise IoT-driven data analytics in intralogistics as well as to develop a ConvNet that can be implemented in a CPS to identify specific conditions of an internal logistic system. The first key finding was the importance of having an optimal internal transport system as it is a key to success in intralogistics. Therefore, the existing conveyor system successfully represents a use case in intralogistics for the CPS demonstrator. Another key finding was the relatively new approach to improve the effectiveness of data value chains through the incorporation of the contextual situation under which the physical system is operating. The six categories orders, physical goods, service, environment, transport system and human operators have been identified as appropriate for intralogistics. Analysing that contextual information results in a better understanding of the internal logistic system and thus, in gaining more knowledge effectively. Furthermore, the identification of appropriate KPIs to provide the relevant information about the operating conditions and performance of the internal logistic system was essential for the development of the data acquisition and data processing systems. Additionally, the detailed analysis of the literature about CPS enabled the development of a conceptual model that represents the current understanding. While a physical object equipped with sensors builds the basis of the data acquisition, the data is stored and processed at a platform. Subsequently, the extracted information is provided to users. Finally, the literature review has discovered that using artificial neural networks to solve classification problems offers several benefits in data volume and speed. Therefore, the investigation of appropriate networks that can be implemented in a CPS to identify the operating conditions of the conveyor system through image recognition was only focused on this type.

In order to realise IoT-driven data analytics in intralogistics, a CPS demonstrator that determines and visualises KPIs of a representative use case was developed.

Starting with the technical implementation of the demonstrator, the first important finding was the improved performance of the visualisation via the video displays and the more constant time between sending data from the Simulink model by using a Raspberry Pi 3 Model B instead of the older version Raspberry Pi Model B. The second finding regarding the implementation was the strong influence of the light condition on the performance of detecting coloured and moving parcels. While the influence on the detection of coloured parcels was easily compensated by adjusting the threshold values for the colours in the Simulink model, the detection of moving parcels needed a different solution. Therefore, several tests were performed with different light conditions by using sunlight, mercury-vapor lamps and halogen lamps. These tests have identified that providing only sunlight enables the best performance. However, the results have also pointed out that the performance of detecting white moving objects is generally the best. Therefore, all containers have been additionally equipped with white paper. The most important key finding of this project was the ability of the developed CPS demonstrator to effectively and efficiently determine and visualise KPIs of a conveyor system. Providing KPIs based on actual and automatically acquired data is the basis upon which better decisions and thus, optimisation in intralogistics can be achieved. This effectivity and efficiency on performance measurement as well as the increased independency of human operators due to the automatic data acquisition are among the main benefits of the demonstrator. Furthermore, the digital system of the developed CPS demonstrator represents a simple and cost-effective solution as it mainly consists of a raspberry pi and a webcam as hardware components, a Simulink model and multiple MATLAB programs as related software as well as several ThingSpeak channels.

The testing and evaluation of the outputs of the CPS demonstrator resulted in additional key findings. First, comparing the outputs of the CPS demonstrator with the outcome of the manual count has shown that the data acquisition system provides a very high precision and accuracy at the beginning of tests as all coloured parcels are detected correctly. However, due to the increasing time displacement, the accuracy drops after a short time. Furthermore, a mainly constant time delay as well as several missing and wrong points have been identified between storing the data and visualising the related KPIs. Therefore, the overall outputs of the demonstrator lack in precision and accuracy. Decreasing these limitations required updating some of the involved processes and systems. The first key finding regarding these updates was the understanding that the limitations identified within the data acquisition system cannot be removed. Therefore, all updates were focused on removing the limitations identified within the data processing system. Calculating and visualising the values of some KPIs by considering their corresponding timestamps was the used method to remove all missing and wrong data points as well as the time delay of the data processing system successfully. Furthermore, averaging the throughput by calculating the arithmetic mean of three and five data points represents a method to slightly compensate the adulterated count of parcels and the increasing time displacement. Hence, the final key findings related to the performed updates were that the limitations within the CPS demonstrator have been reduced significantly as well as that the consequences of the still involved limitations are relatively small and thus, acceptable.

The first key finding during the investigation of appropriate artificial neural networks that can be implemented in a CPS to identify the operating conditions of the conveyor system was that ConvNets have become a requirement to realise a state-of-the-art performance in image recognition. Therefore, three basic approaches to develop a ConvNet structure focused on the specific requirements of the defined use case were explored. While using a pretrained model is typically an inappropriate solution in intralogistics, a retrained model might be suitable in cases the images involved in the new task are similar to the original training images. However, developing a completely new model provides the most benefits as it can be entirely focused on the use case requirements. The final key finding was the classification accuracy of 99.84%, achieved by applying the developed ConvNet structure on a similar task that involved recognising images of handwritten digits between 0 and 9. Due to the similarities between the requirements of both tasks, it is assumed that this test result is also representative for identifying the operating conditions of the conveyor system.

8 Conclusion and Future Work

In order to complete this research project, the last chapter provides the conclusion and future work.

8.1 Conclusion

The first research question is answered as the developed CPS demonstrator represents an effective and efficient way to determine and visualise KPIs of an internal logistic system. Furthermore, analysing the provided information enables gaining new knowledge of its operating conditions and performance. Besides that, the developed ConvNet represents a suitable machine learning application that can be implemented in a CPS to gain additional knowledge of an internal logistic system. Hence, also the second research question is answered.

Finally, the implementation of the CPS demonstrator completes a realisation of IoT-driven data analytics in intralogistics. Due to the focus of the developed ConvNet on identifying specific conditions of an internal logistic system, also the aim of this project is achieved and the research gaps are addressed successfully.

Among all results of this project, the following points have been identified as very important:

1. IoT enables new approaches to optimise internal logistic systems in terms of operational performance, uptime and sustainability. This leads to a higher business value in manufacturing and warehousing.
2. Determining and visualising KPIs via a CPS enables monitoring the operating conditions and performance of an internal logistic system in a very effective way. Analysing the provided information allows gaining new knowledge, whereas its application through corrective and preventive actions leads to optimisation in intralogistics.
3. The incorporation of context awareness improves the effectiveness of data value chains and thus, of gaining new knowledge.
4. The developed CPS demonstrator determines and visualises KPIs of a representative use case in intralogistics in an effective and efficient way.

Having KPIs based on actual and automatically acquired data is the basis upon which better decisions and thus, optimisation can be achieved.

5. A digital system of a CPS that mainly consists of a raspberry pi and a webcam as hardware components, a Simulink model and multiple MATLAB programs as related software as well as several ThingSpeak channels represents a simple and cost-effective solution to determine and visualise KPIs of an internal logistic system as the physical system.
6. Calculating and visualising the values of KPIs by considering their corresponding timestamps as well as averaging some values over a specific period of time represent simple methods to remove or compensate limitations that might occur within a CPS. Applying these methods results in an increased reliability of the CPS outputs.
7. While using artificial neural networks to solve classification problems offers several benefits in data volume and speed, ConvNets have become a requirement to realise a state-of-the-art performance in image recognition.
8. Among the three basic approaches to develop a ConvNet structure, developing a completely new model provides the most benefits for applications in intralogistics as it can be entirely focused on the specific requirements of the involved tasks.
9. The developed ConvNet structure represents a suitable classification model that can be implemented in a CPS to identify the operating conditions of a conveyor system. However, a proper training of this model must be performed first.

8.2 Future Work

Even though a CPS demonstrator to determine and visualise KPIs of an internal logistic system has been developed successfully, this only represents first steps of a product development process. Possible next steps include focusing on larger scale CPS implementations in intralogistics as a precursor to industry adoption. This might be realised through an existing or a new cooperation between Graz University of Technology (Institute of Logistics Engineering) and one or more industrial partners. Therefore, access to academic knowledge and industrial experience as well as to internal logistic systems that are actually employed in the industry is provided. Starting with the implementation in one trivial conveyor system, further implementations of the CPS in all available internal logistic systems are suggested as soon as it has demonstrated and proven its benefits. However, there is still room to improve the outputs of the CPS demonstrator. Therefore, a detailed investigation of the influence of the light condition on the performance of the detection of moving and coloured parcels is recommended. Such an investigation should include the measurement of the different light spectrums to identify the optimum condition.

Regarding the developed ConvNet to identify the operating conditions of a conveyor system, next steps must include its proper training. Therefore, collecting about 4000 training images depicting the four conditions (1000 images per category) is required. Having trained the network correctly, integrating it in an edge node enables its successful implementation in a CPS. As only little computing power requirements are expected, the realisation of the edge node as single-board computer (e.g. raspberry pi) is recommended. Furthermore, an additional MATLAB code might be used to send the identified operating conditions to a ThingSpeak channel. This enables processing the stored data as well as visualising the extracted information.

Finally, another study focused on the analysation of the provided information to gain knowledge of the internal logistic system as well as on the application of this knowledge to realise optimisation in intralogistics is recommended.

REFERENCES

1. Aazam, M., Zeadally, S. and Harras, K. (2018) ‘Fog Computing Architecture, Evaluation, and Future Research Directions’, *IEEE Communications Magazine*. 56 (5), pp. 46-52.
2. Alpaydin, E. (2014) *Introduction to Machine Learning*. Cambridge (USA): Massachusetts Institute of Technology
3. Babovic, Z.B., Protic, J. and Milutinovic, V. (2016) ‘Web Performance Evaluation for Internet of Things Applications’, *IEEE Access*. 4 (5), pp. 6974-6992
4. Ballarino, A., Brondi, C., Brusaferri, A. and Chizzoli, G. (2017) ‘The CPS and LCA Modelling: An Integrated Approach in the Environmental Sustainability Perspective’, *Collaboration in a Data-Rich World: 18th IFIP WG 5.5 Working Conference on Virtual Enterprises*. Vicenza, Italy, 18-20 September. Switzerland: Springer, pp. 543-552.
5. Baškarada, S. and Koronios, A. (2013) ‘Data, Information, Knowledge, Wisdom (DIKW): A Semiotic Theoretical and Empirical Exploration of the Hierarchy and its Quality Dimension’, Australasian Journal of Information Systems, 18 (1), pp. 5-24.
6. Bell, J. (2014) *Machine Learning - Hands-On for Developers and Technical Professionals*. Indianapolis: John Wiley & Sons, Inc.
7. Bode, W. and Preuß, R. (2005) *Comprehensive introduction to Intralogistics*. Norderstedt: Books on Demand.
8. Ciresan, D., Meier, U., Masci, J., Gambardella, L.M. and Schmidhuber, J. (2011) ‘Flexible, high performance convolutional neural networks for image classification’, *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*. Barcelona, Spain, 16-22 July. Spain: Association for the Advancement of Artificial Intelligence, pp. 1237-1242.
9. Colombo, A. W., Karnouskos, S., Shi, Y., Yin, S. and Kaynak, O. (2017) ‘Industrial Cyber-Physical Systems’, *Proceedings of the IEEE*, 104 (5), pp. 899–903.
10. Cruz-Piris, L., Rivera, D., Marsa-Maestre, I., De La Hoz, E. and Velasco, J.R. (2018) ‘Access control mechanism for IoT environments based on

modelling communication procedures as resources', *Sensors (Switzerland)*, 18 (3), pp. 1-21

11. Curry, E. (2016) 'The Big Data Value Chain: Definitions, Concepts, and Theoretical Approaches', in, Cavanillas, J., Curry, E. and Wahlster, W. (eds.) *New Horizons for a Data-Driven Economy*. Switzerland: Springer, pp. 29-38.
12. Deschambault, O., Gherbi, A. and Légaré, C. (2017) 'Efficient implementation of the MQTT protocol for embedded systems', *Journal of Information Processing Systems*, 13 (1), pp. 26-39
13. Deshpande, M. (2017) *A Guide to Improving Deep Learning's Performance*. Available at: https://pythonmachinelearning.pro/a-guide-to-improving-deep-learnings-performance/. (Accessed: 15 September 2018).
14. Dizdarevic, J., Carpio, F., Jukan, A. and Masip-Bruin, X. (2018) 'A Survey of Communication Protocols for Internet-of-Things and Related Challenges of Fog and Cloud Computing Integration', *ACM Computing Surveys*. 1 (1), pp. 1-27.
15. Fall K. R. and Stevens W. R. (2012) *TCP/IP Illustrated, Volume 1 – The Protocols*. United States: Pearson Education, Inc.
16. Ferrari, P., Sisinni, E., Brandão, D. and Rocha, M. (2017) 'Evaluation of communication latency in industrial IoT applications', *2017 IEEE International Workshop on Measurement and Networking (M&N)*. Italy, Naples, 27-29 September. Italy: Institute of Electrical and Electronics Engineers Inc., pp. 1-9.
17. Fritz, M. (2016) *Beitrag zur Simulation des Bewegungsverhaltens von Stückgütern im Pulk im Kontext der Vereinzelung*. Graz: Technische Universität Graz, Dissertation.
18. Granlund, A. and Wiktorsson, M. (2014) 'Automation in internal logistics: Strategies and operational challenges', *International Journal of Logistics Systems and Management*, 18 (4), pp. 538-558
19. Gilchrist A. (2016) Industry 4.0: *The Industrial Internet of Things*. Thailand: Apress.

20. Gudehus, T. and Herber, K. (2012) *Comprehensive Logistics*. Heidelberg: Springer.

21. Henderson, R. (2013) *Network architecture of the future: It's now*. Available at: https://www.controleng.com/single-article/network-architecture-of-the-future-its-now/. (Accessed: 01 June 2018).

22. HiveMQ (2015) *MQTT 101 – How to Get Started with the lightweight IoT Protocol*. Available at: https://www.hivemq.com/blog/how-to-get-started-with-mqtt. (Accessed: 25 June 2018).

23. Hribernik, K., Hans, C., Kramer, C. and Thoben, K. (2011) 'A Service-oriented, Semantic Approach to Data Integration for an Internet of Things Supporting Autonomous Cooperating Logistics Processes', in, Uckelmann, D., Harrison, M. and Michahelles, F. (eds.) *Architecting the Internet of Things*. Heidelberg: Springer, pp. 131-158.

24. Hwang, G., Lee, J., Park, J. and Chang, T. (2017) 'Developing performance measurement system for Internet of Things and smart factory environment', *International Journal of Production Research*, 55 (9), pp. 2590-2602.

25. ImageNet (2016) *Summary and Statistics*. Available at: http://image-net.org/index. (Accessed: 05 October 2018).

26. Intralogistik-Netzwerk BW (2018) *INTRALOGISTIK-RADAR: DAS SIND DIE ZUKUNFTSTHEMEN*. Available at: https://www.intralogistik-bw.de/intralogistik-radar-das-sind-die-zukunftsthemen/. (Accessed: 15 September 2018).

27. ISA-95.com (2017) *ISA-95 Enterprise Control Systems*. Available at: https://isa-95.com/isa-95-enterprise-control-systems/. (Accessed: 01 June 2018).

28. ISA-95.com (2017) *TECHNICAL: ISA-88 AND ISA-95*. Available at: https://isa-95.com/technical-isa-88-and-isa-95/. (Accessed: 01 June 2018).

29. Jordan, F., Bernardy, A., Stroh, M., Horeis, J. and Stich, V. (2017) 'Requirements-Based matching approach to configurate cyber-physical systems for SMEs', *Portland International Conference on Management of*

Engineering and Technology: Technology Management for the Interconnected World. Portland, United States of America, 09-03 July. United States of America: Institute of Electrical and Electronics Engineers Inc., pp. 1-7.

30. Krizhevsky, A., Sutskever, I., Hinton, G.E. (2012) ‘ImageNet Classification with Deep Convolutional Neural Networks’, *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012*. United States of America, Nevada, 03-08 December. Nevada: NIPS Proceedings 2012, pp. 1097-1105.
31. Lee, J., Bagheri, B. and Kao, H. (2015) ‘A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems’, *Manufacturing Letters*, 3, pp. 18-23.
32. Macaulay, J., Buckalew, L. and Chung, G. (2015) *Internet of Things in Logistics*. Troisdorf: DHL.
33. Makadia, M. (2018) *Top 8 Deep Learning Frameworks*. Available at: https://dzone.com/articles/8-best-deep-learning-frameworks. (Accessed: 06 October 2018).
34. Marwedel, P. and Engel, M. (2016) ‘Cyber-Physical Systems: Opportunities, Challenges and (Some) Solutions‘, in, Guerrieri, A., Loscri, V., Rovella, A. and Fortino, G. (eds.) *Management of Cyber Physical Objects in the Future Internet of Things: Methods, Architectures and Applications*. Switzerland: Springer, pp. 1-30.
35. MathWorks (2016) *Schritt-für-Schritt-Anleitung für Machine Learning.* Available at: https://mathworks.com/campaigns/offers/machine-learning-with-matlab.html. (Accessed: 12 September 2018).
36. MathWorks (2018a) *Deep Network Designer*. Available at: https://mathworks.com/help/deeplearning/ref/deepnetworkdesigner-app.html. (Accessed: 09 October 2018).
37. MathWorks (2018b) *Statistics and Machine Learning Toolbox*. Available at: https://mathworks.com/products/statistics.html. (Accessed: 09 October 2018).

38. MathWorks (2018c) *Pretrained Convolutional Neural Networks*. Available at: https://mathworks.com/help/deeplearning/ug/pretrained-convolutional-neural-networks.html. (Accessed: 06 October 2018).

39. MathWorks (2018d) *Pretrained AlexNet convolutional neural network*. Available at: https://mathworks.com/help/deeplearning/ref/alexnet.html. (Accessed: 06 October 2018).

40. MathWorks (2018e) *Get Started with Transfer Learning*. Available at: https://mathworks.com/help/deeplearning/examples/get-started-with-transfer-learning.html. (Accessed: 06 October 2018).

41. MathWorks (2018f) *Transfer Learning Using AlexNet*. Available at: https://mathworks.com/help/deeplearning/examples/transfer-learning-using-alexnet.html. (Accessed: 06 October 2018).

42. MathWorks (2018g) *Feature Extraction*. Available at: https://mathworks.com/discovery/feature-extraction.html. (Accessed: 06 October 2018).

43. MathWorks (2018h) *Create Simple Deep Learning Network for Classification*. Available at: https://mathworks.com/help/deeplearning/examples/create-simple-deep-learning-network-for-classification.html. (Accessed: 09 October 2018).

44. Mendes, C., Osaki, R., Da Costa, C. (2017) 'Internet of Things in Automated Production', *European Journal of Engineering Research and Science*, 2, pp. 13-16.

45. Miller, H. and Mork, P. (2013) 'From data to decisions: A value chain for big data', *IT Professional*, 15 (1), pp. 57-59

46. Moris, J. (2013) *What is Intralogistics*?. Available at: https://www.invata.com/what-is-intralogistics/. (Accessed: 21 May 2018).

47. Nagel, L., Roidl, M. and Follert, G. (2008) 'The Internet of Things: On Standardisation in the Domain of Intralogistics', *Adjunct Proceedings: First International Conference on The Internet of Things*. Zurich, Switzerland, 26-28 March. Zurich: Springer, pp. 16-21.

48. Nagorny, K., Scholze, S., Barata, J. and Colombo A. (2016) 'An Approach for Implementing ISA 95-Compliant Big Data Observation, Analysis and

Diagnosis Features in Industry 4.0 Vision Following Manufacturing Systems', Technological Innovation for Cyber-Physical Systems: 7th IFIP WG 5.5/SOCOLNET Advanced Doctoral Conference on Computing, Electrical and Industrial Systems. Costa de Caparica, Portugal, 11-13 April. Switzerland: Springer, pp. 116-123.

49. Năstase, L., Sandu, I. and Popescu, N. (2017) 'An experimental evaluation of application layer protocols for the internet of things', *Studies in Informatics and Control*, 26 (4), pp. 403-412.

50. Nielsen,M. (2017) *Neural Networks and Deep Learning*. Available at: http://neuralnetworksanddeeplearning.com/index.html. (Accessed: 13 September 2018).

51. Nvidia (2018) *Deep Learning Frameworks*. Available at: https://developer.nvidia.com/deep-learning-frameworks. (Accessed: 06 October 2018).

52. Oasis (2014) *MQTT Version 3.1.1*. OASIS Open

53. Paluszek, M. and Thomas, S. (2016) *MATLAB Machine Learning*. New Jersey: Apress L. P.

54. Perera, C., Zaslavsky, A., Christen, P. and Georgakopoulos, D. (2014) 'Context Aware Computing for The Internet of Things: A Survey', *IEEE Communications Surveys & Tutorial*, 16 (1), pp. 1-41

55. Rowley, J. (2007) 'The wisdom hierarchy: representations of the DIKW hierarchy', *Journal of Information Science*, 33 (2), pp. 163-180.

56. Sanislav, T. and Miclea, L. (2012) 'Cyber-Physical Systems - Concept, Challenges and Research Areas', *Journal of Control Engineering and Applied Informatics*, 14 (2), pp. 28-33.

57. Scholten, B. (2007) The Road to Integration: A Guide to Applying the ISA-95 Standard in Manufacturing

58. Scholze, S. and Barata, J. (2016) 'Context Awareness for Flexible Manufacturing Systems Using Cyber Physical Approaches', *Technological Innovation for Cyber-Physical Systems: 7th IFIP WG 5.5/SOCOLNET Advanced Doctoral Conference on Computing, Electrical and Industrial*

Systems. Costa de Caparica, Portugal, 11-13 April. Switzerland: Springer, pp. 107-115.

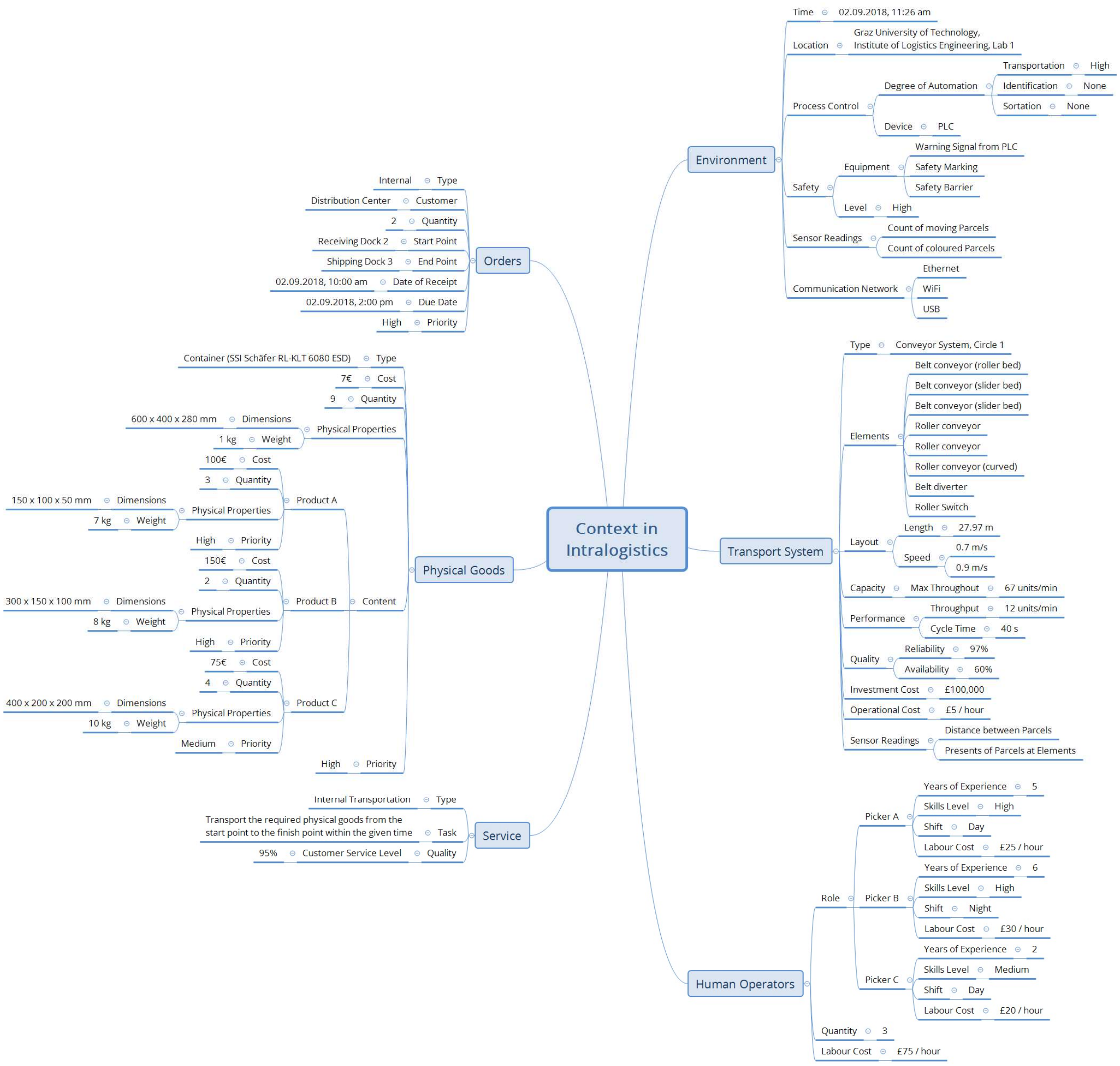
59. Schuh, G., Bernardy, A., Zeller, V. and Stich, V. (2017) 'New Requirement Analysis Approach for Cyber-Physical Systems in an Intralogistics Use Case', *Collaboration in a Data-Rich World: 18th IFIP WG 5.5 Working Conference on Virtual Enterprises*. Vicenza, Italy, 18-20 September. Switzerland: Springer, pp. 107-115.
60. Tan, Y., Goddard, S. and Pérez, L. (2008) 'A prototype architecture for cyber-physical systems', *ACM SIGBED Review*, 5 (1), pp. 1-2.
61. Ujjwal, K. (2016) *An Intuitive Explanation of Convolutional Neural Networks*. Available at: https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/. (Accessed: 23 September 2018).
62. Weisstein, E. (2018) *Sigmoid Function*. Available at: http://mathworld.wolfram.com/SigmoidFunction.html. (Accessed: 20 September 2018).
63. Wiesner, S., Gorldt, C., Soeken, M., Thoben, K. and Drechsler, R. (2014) 'Requirements Engineering for Cyber-Physical Systems - Challenges in the Context of "Industrie 4.0"', *Advances in Production Management Systems - Innovative and Knowledge-Based Production Management in a Global-Local World: IFIP WG 5.7 International Conference*. Ajaccio, France, 20-24 September. Switzerland: Springer, pp. 281-288.
64. Wikipedia (2018a) *Convolutional neural network*. Available at: https://en.wikipedia.org/w/index.php?title=Convolutional_neural_network&oldid=860564209. (Accessed: 23 September 2018).
65. Wikipedia (2018b) *Comparison of deep learning software*. Available at: https://en.wikipedia.org/wiki/Comparison_of_deep_learning_software. (Accessed: 06 October 2018).
66. Zeiler, M.D. and Fergus, R. (2014) 'Visualizing and Understanding Convolutional Networks', *Computer Vision – ECCV 2014, Part I*. Zurich, Switzerland, 06-12 September. Zurich: Springer, pp. 818-833.

67. Zhang, P., Zhou, M. and Fortino, G. (2018) ‘Security and trust issues in Fog computing: A survey’, *Future Generation Computer Systems*, 88, pp. 16-27.

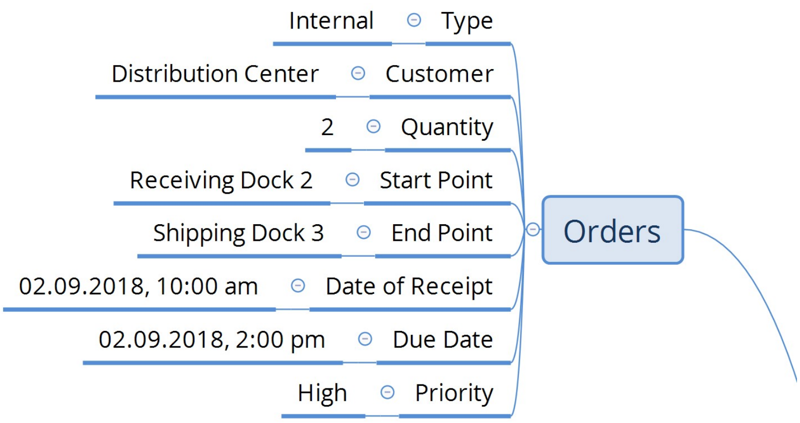
APPENDICES

Appendix A Context Model of Use Case

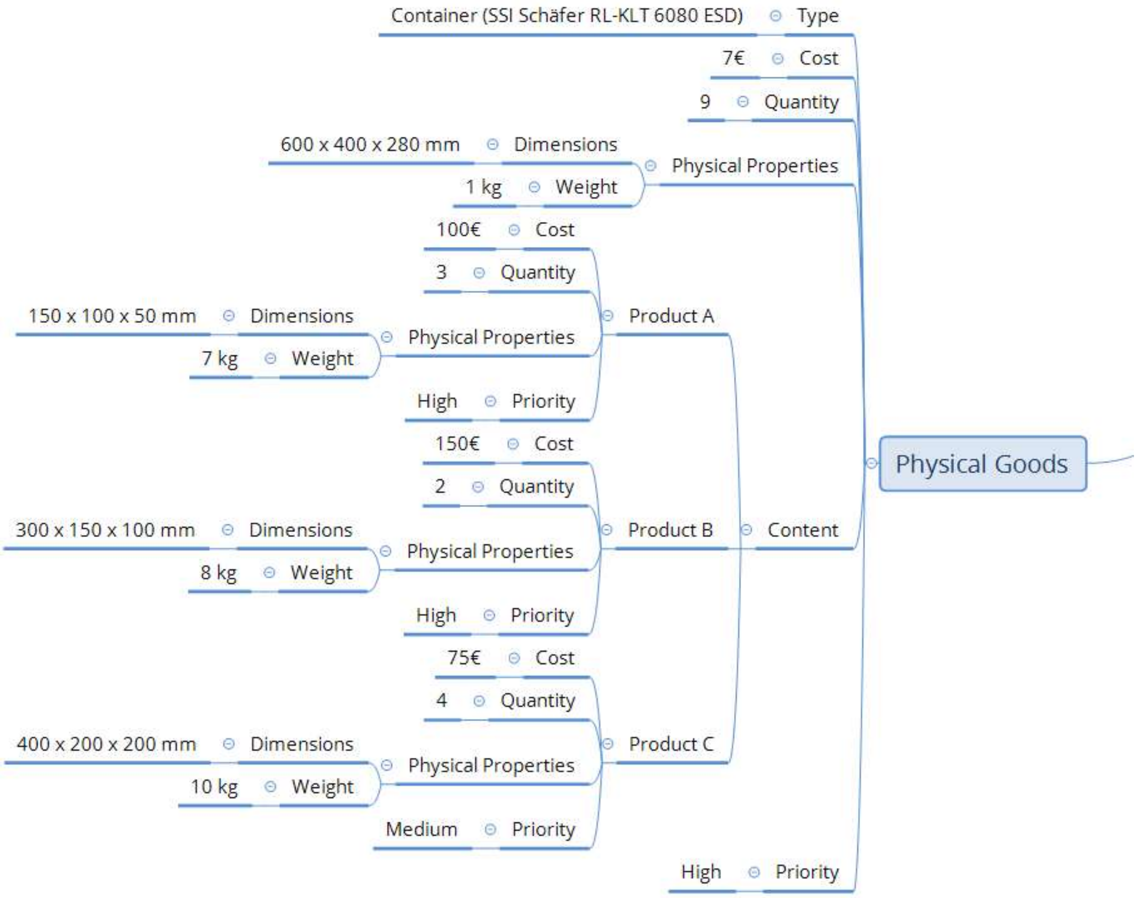
Overall Model



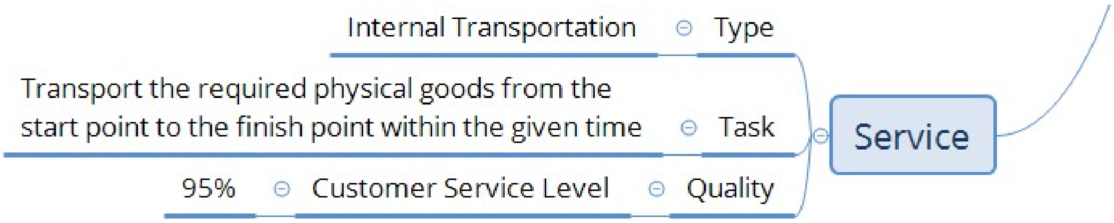
Orders



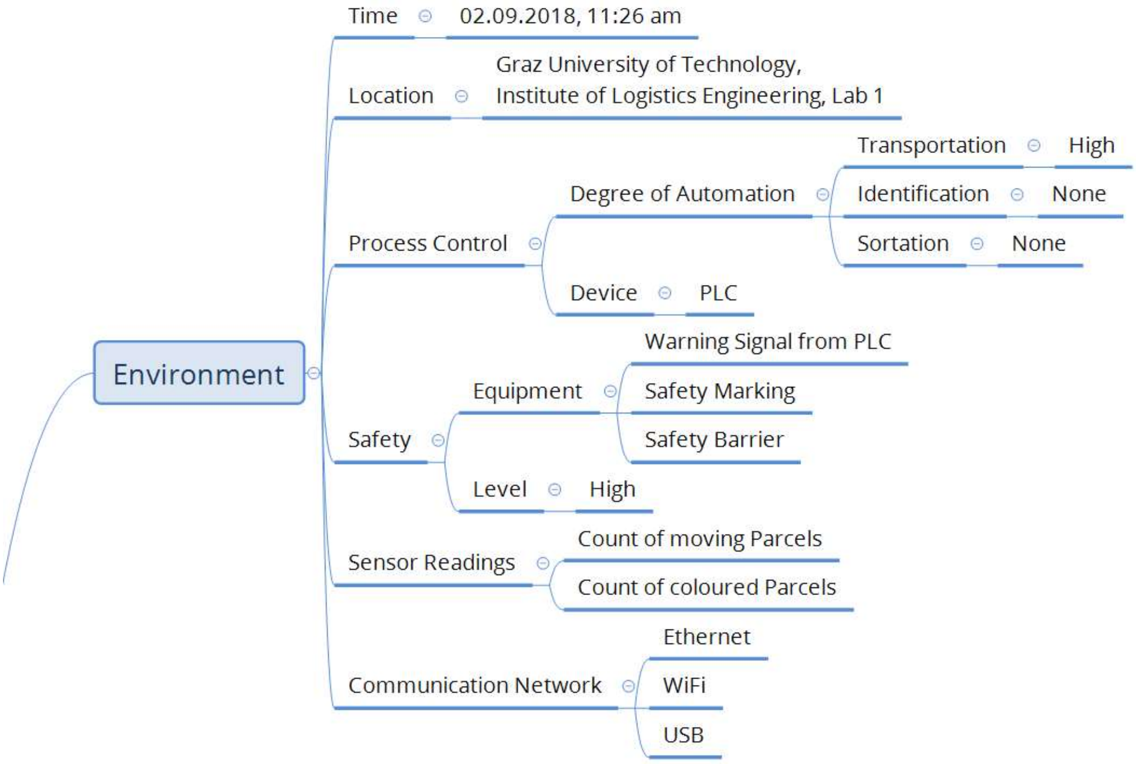
Physical Goods



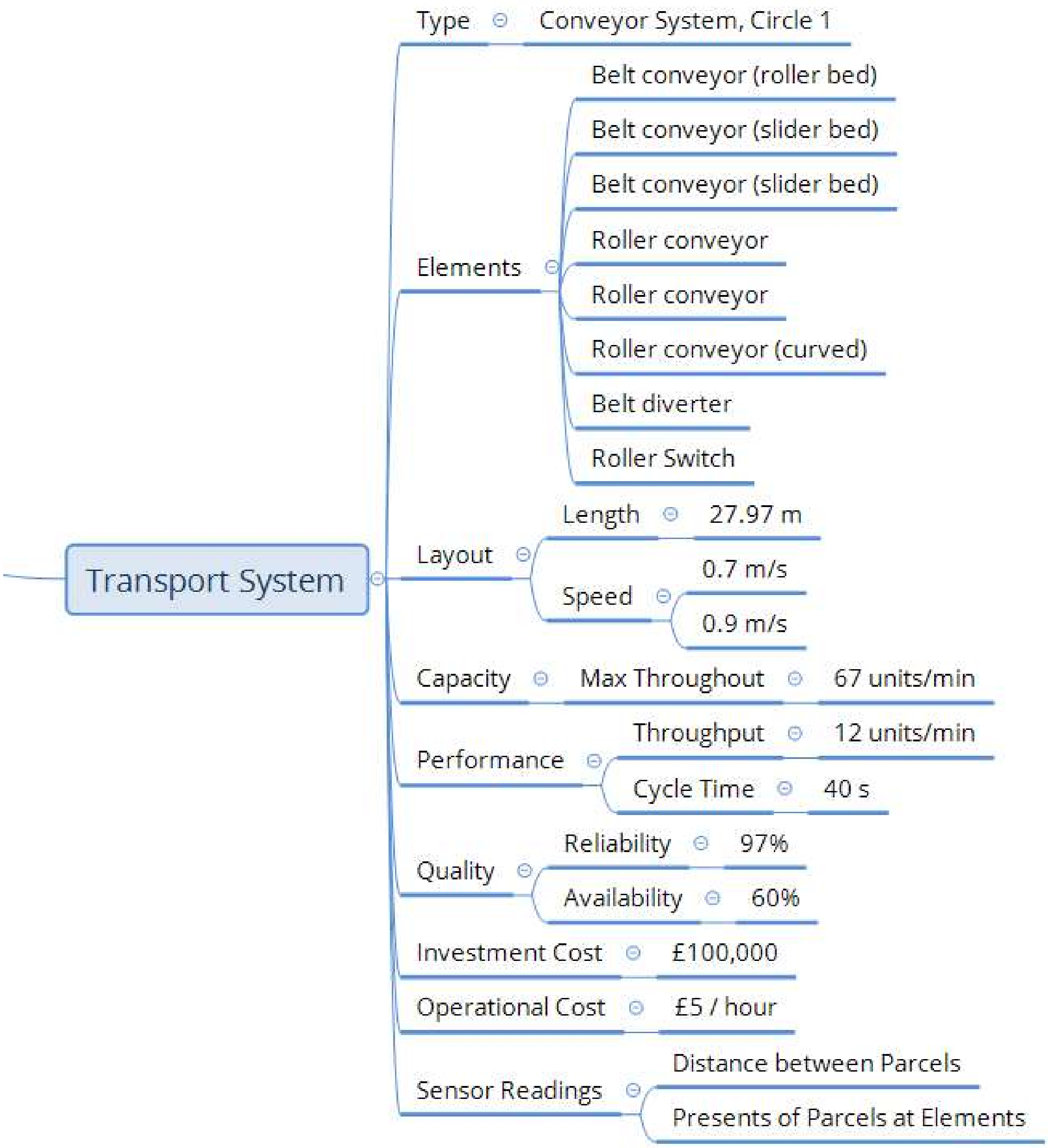
Service


Service
Type
Internal Transportation
Task
Transport the required physical goods from the start point to the finish point within the given time
Quality
Customer Service Level
95%

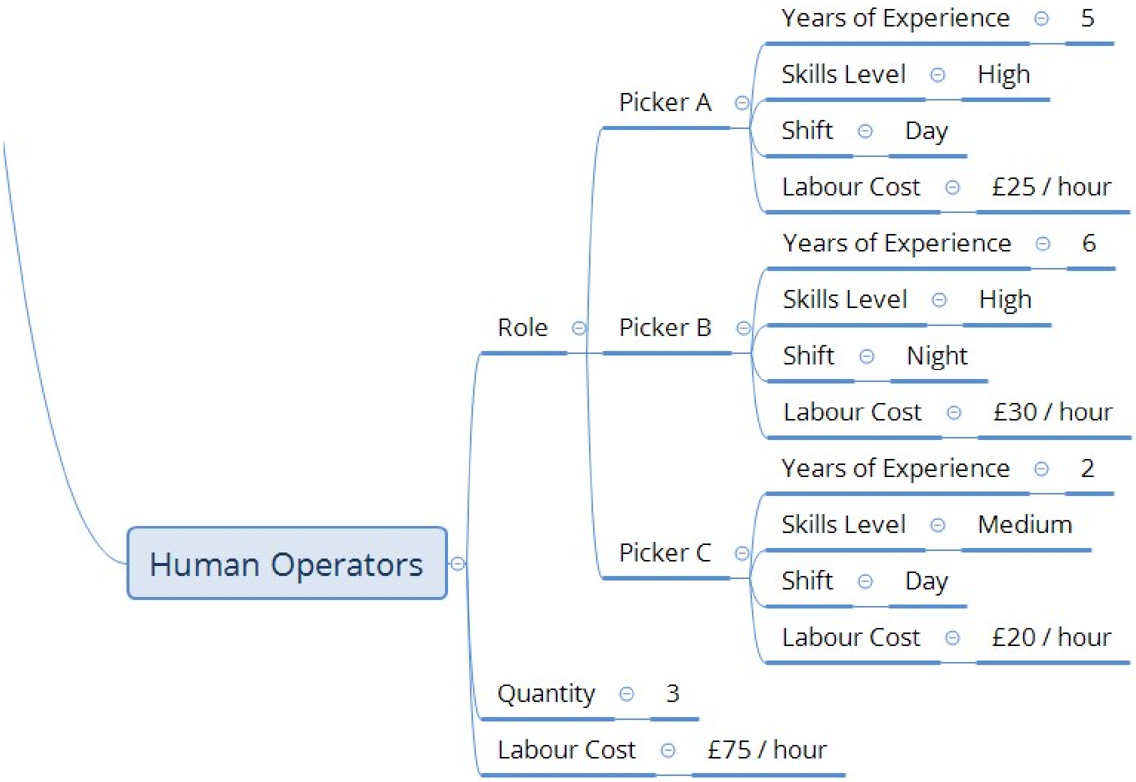
Environment



<u>Transport System</u>



<u>Human Operator</u>



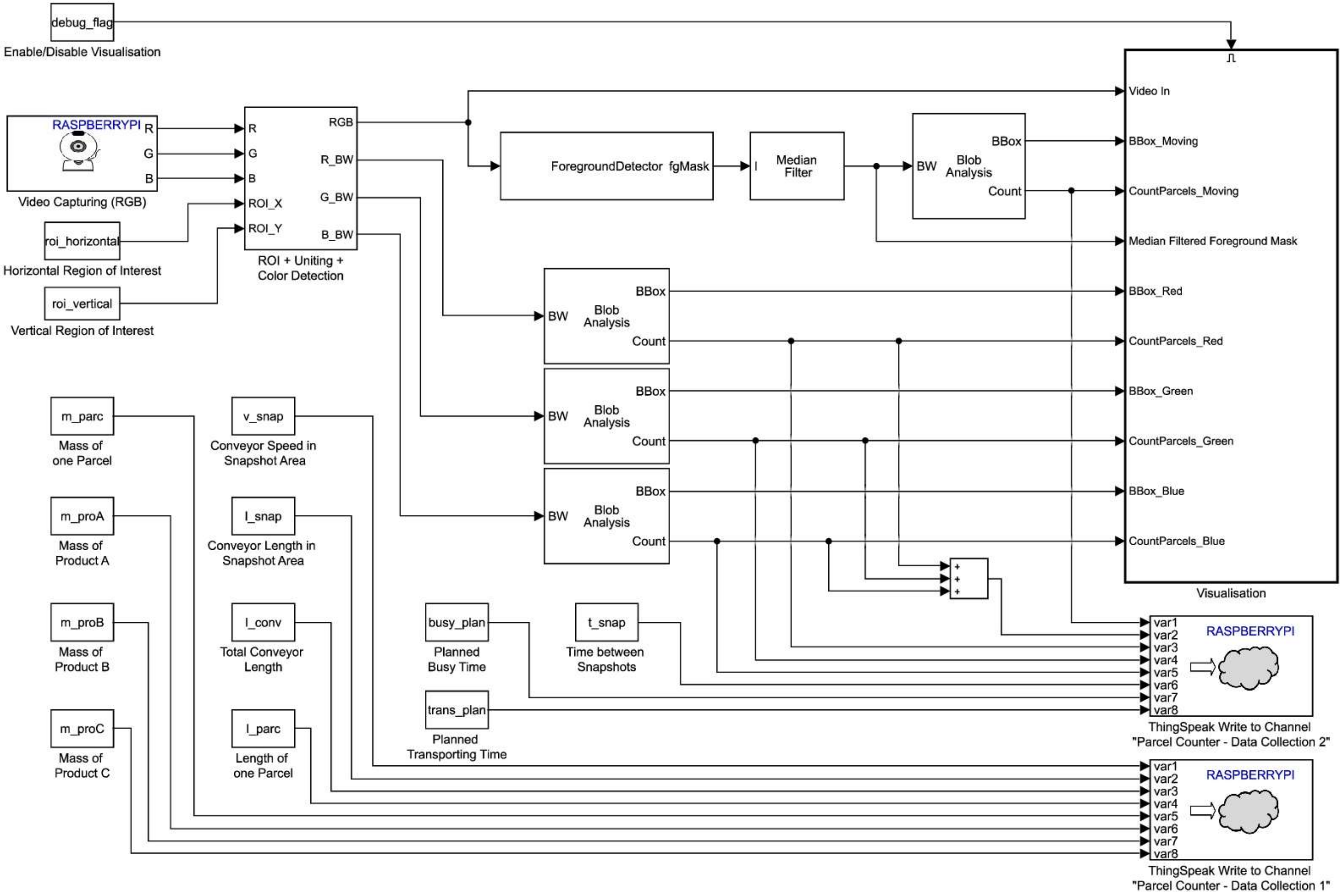
Appendix B MATLAB Code of User Interface

```
%% Start of the Program
clc;
clear;
fprintf('---------------------\nParcel Counter\n----------------
-----\n');

%% Running the Program

loop_scenario = 1;
debug_flag = 1;

while(loop_scenario)

    fprintf('Please choose:\n0...Stop the Program\n1...Change
Parameters and Run Simulink Model on Laptop\n2...Keep Default
Parameters and Run Simulink Model on Raspberry Pi\n\n');

    n = input('Enter a number: ');

    switch n
        case 0
            loop_scenario = 0;
            fprintf('\nYou have stopped the program\n\n');
        case 1
            fprintf('\nYour choice: Run Simulink Model on
Laptop\n');

            %% Adjust Resolution and Region of Interest

            fprintf('\n---------------------------------\nAdjust
Resolution and Region of Interest\n\n');

            loop_resolution = 1;

            loop_dist_left = 1;
            loop_dist_right = 1;
            loop_dist_above = 1;
            loop_dist_below = 1;

            while(loop_resolution)

                fprintf('Please choose:\n1...160 x 120\n2...320
x 240\n3...640 x 480\n\n');

                n = input('Enter a number: ');

                switch n
                    case 1
                        image_size = [160,120];
                        sample_time = 0.1;

                        loop_resolution = 0;

                        while(loop_dist_left)

                            dist_left = input('\nPlease enter
the left distance between 0 and 159 [pixel]: ');

                            if (dist_left >= 0) && (dist_left <=
159)
                                loop_dist_left = 0;
                            else
                                fprintf('Wrong Input, you can
only enter a number between 0 and 159!\n');
                            end
                        end

                        while(loop_dist_right)

                            fprintf('Please enter the right
distance between 0 and %d [pixel]: ',(159-dist_left));
                            dist_right = input('');

                            if (dist_right >= 0) && (dist_right
<= (159-dist_left))
                                loop_dist_right = 0;
                            else
                                fprintf('Wrong Input, you can
only enter a number between 0 and %d!\n\n',(159-dist_left));
                            end
                        end

                        while(loop_dist_above)

                            dist_above = input('Please enter the
upper distance between 0 and 119 [pixel]: ');

                            if (dist_above >= 0) && (dist_above
<= 119)
                                loop_dist_above = 0;
                            else
                                fprintf('Wrong Input, you can
only enter a number between 0 and 119!\n\n');
                            end
                        end

                        while(loop_dist_below)

                            fprintf('Please enter the lower
distance between 0 and %d [pixel]: ',(119-dist_above));
                            dist_below = input('');
                            if (dist_below >= 0) && (dist_below
<= (119-dist_above))
                                loop_dist_below = 0;
                            else
                                fprintf('Wrong Input, you can
only enter a number between 0 and %d!\n\n',(119-dist_below));
                            end
                        end

                        roi_horizontal = (dist_left+1):(160-
dist_right);
                        roi_vertical = (dist_above+1):(120-
dist_below);

                    case 2
                        image_size = [320,240];
                        sample_time = 0.15;

                        loop_resolution = 0;

                        while(loop_dist_left)

                            dist_left = input('\nPlease enter
the left distance between 0 and 319 [pixel]: ');

                            if (dist_left >= 0) && (dist_left <=
319)
                                loop_dist_left = 0;
                            else
                                fprintf('Wrong Input, you can
only enter a number between 0 and 319!\n');
                            end
                        end

                        while(loop_dist_right)

                            fprintf('Please enter the right
distance between 0 and %d [pixel]: ',(319-dist_left));
                            dist_right = input('');

                            if (dist_right >= 0) && (dist_right
<= (319-dist_left))
                                loop_dist_right = 0;
                            else
                                fprintf('Wrong Input, you can
only enter a number between 0 and %d!\n\n',(319-dist_left));
                            end
                        end

                        while(loop_dist_above)

                            dist_above = input('Please enter the
upper distance between 0 and 239 [pixel]: ');
```

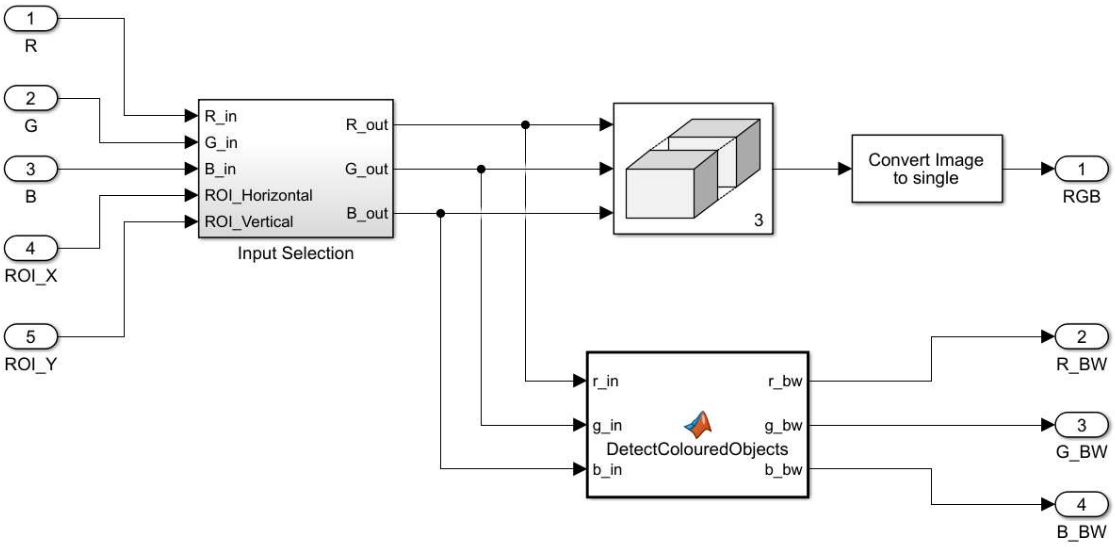
```
                            if (dist_above >= 0) && (dist_above 
<= 239)
                                loop_dist_above = 0;
                            else
                                fprintf('Wrong Input, you can 
only enter a number between 0 and 239!\n\n');
                            end
                        end

                        while(loop_dist_below)

                            fprintf('Please enter the lower 
distance between 0 and %d [pixel]: ',(239-dist_above));
                            dist_below = input('');

                            if (dist_below >= 0) && (dist_below 
<= (239-dist_above))
                                loop_dist_below = 0;
                            else
                                fprintf('Wrong Input, you can 
only enter a number between 0 and %d!\n\n',(239-dist_below));
                            end
                        end

                        roi_horizontal = (dist_left+1):(320-
dist_right);
                        roi_vertical = (dist_above+1):(240-
dist_below);

                    case 3
                        image_size = [640,480];
                        sample_time = 0.5;

                        loop_resolution = 0;

                        while(loop_dist_left)

                            dist_left = input('\nPlease enter 
the left distance between 0 and 639 [pixel]: ');

                            if (dist_left >= 0) && (dist_left <= 
639)
                                loop_dist_left = 0;
                            else
                                fprintf('Wrong Input, you can 
only enter a number between 0 and 639!\n');
                            end
                        end

                        while(loop_dist_right)
```

```
                            fprintf('Please enter the right 
distance between 0 and %d [pixel]: ',(639-dist_left));
                            dist_right = input('');

                            if (dist_right >= 0) && (dist_right 
<= (639-dist_left))
                                loop_dist_right = 0;
                            else
                                fprintf('Wrong Input, you can 
only enter a number between 0 and %d!\n\n',(639-dist_left));
                            end
                        end

                        while(loop_dist_above)

                            dist_above = input('Please enter the 
upper distance between 0 and 479 [pixel]: ');

                            if (dist_above >= 0) && (dist_above 
<= 479)
                                loop_dist_above = 0;
                            else
                                fprintf('Wrong Input, you can 
only enter a number between 0 and 479!\n\n');
                            end
                        end

                        while(loop_dist_below)

                            fprintf('Please enter the lower 
distance between 0 and %d [pixel]: ',(479-dist_above));
                            dist_below = input('');

                            if (dist_below >= 0) && (dist_below 
<= (479-dist_above))
                                loop_dist_below = 0;
                            else
                                fprintf('Wrong Input, you can 
only enter a number between 0 and %d!\n\n',(479-dist_below));
                            end
                        end

                        roi_horizontal = (dist_left+1):(640-
dist_right);
                        roi_vertical = (dist_above+1):(480-
dist_below);

                    otherwise
                        fprintf('Wrong Choice, you can only 
enter a number between 1 and 3!\n\n')

                end
            end
            %% Calculation of Snapshot Time

            fprintf('\n--------------------------------
\nCalculation of Snapshot Time\n\n');

            v_snap = input('Please enter the speed of the
conveyor system that is covered by the camera [m/s]: ');
            l_snap = input('Please enter the length of the
conveyor system that is covered by the camera [m]: ');
            l_conv = input('Please enter the total length of the
conveyor system [m]: ');
            l_parc = input('Please enter the length of one
parcel [m]: ');
            m_parc = input('Please enter the mass of one parcel
[kg]: ');
            t_snap = (l_snap+l_parc-0.5*l_parc) / v_snap;
            fprintf('\nThe time between snapshots is set to
%2.2f m/s\n', t_snap);

            %% Mass of Products A,B,C

            fprintf('\n--------------------------------\nMass of
Products A,B,C\n\n');

            m_proA = input('Please enter the mass of product A
(red parcel)[kg]: ');
            m_proB = input('Please enter the mass of product B
(green parcel)[kg]: ');
            m_proC = input('Please enter the mass of product C
(blue parcel)[kg]: ');

            %% Planned Performance

            fprintf('\n--------------------------------\nPlanned
Performance\n\n');

            loop_busy_plan = 1;
            loop_trans_plan = 1;
            loop_down_plan = 1;

            while(loop_busy_plan)

                busy_plan = input('Please enter the planned busy
time as percentage of the operation time between 0 and 100 [%]:
');

                if (busy_plan >= 0) && (busy_plan <= 100)
                    loop_busy_plan = 0;
                else
                    fprintf('Wrong Input, you can only enter a
number between 0 and 100!\n\n');
                end
            end

            while(loop_trans_plan)

                trans_plan = input('Please enter the planned
transporting time as percentage of the planned busy time between
0 and 100 [%]: ');

                if (trans_plan >= 0) && (trans_plan <= 100)
                    loop_trans_plan = 0;
                else
                    fprintf('Wrong Input, you can only enter a
number between 0 and 100!\n\n');
                end
            end

            %% Loading the Model and Running on Laptop

            debug_flag = 1;
            loop_laptop = 1;

            fprintf('\nThe model is loading...\n');

%set_param('Parcel_Counter_Colour_Detector_Simulink',
'SimulationCommand', 'start');
            fprintf('\nThe model is running\nEnter 0 to stop
running the model\n\n');
            while(loop_laptop)
                m = input('Enter a number: ');
                switch m
                    case 0

%set_param('Parcel_Counter_Colour_Detector_Simulink',
'SimulationCommand', 'stop');
                        fprintf('\nYou have stopped running the
model\n\n');
                        fprintf('---------------------\nParcel
Counter\n---------------------\n');
                        loop_laptop = 0;
                    otherwise
                        fprintf('Wrong Choice, you can only
enter 0!\n\n')
                end
            end
        case 2
            %% Loading the Model and Running on Raspberry Pi

            fprintf('\nYour choice: Run Simulink Model on
Raspberry Pi\n');

            debug_flag = 0;
            loop_raspi = 1;
            %h = raspberrypi;

%h.runModel('Parcel_Counter_Colour_Detector_Simulink');

            fprintf('\nThe model is running\nEnter 0 to stop
running the model\n\n');

            while(loop_raspi)
                l = input('Enter a number: ');
                switch l
                    case 0

%h.stopModel('Parcel_Counter_Colour_Detector_Simulink');
                        fprintf('\nYou have stopped running the
model\n\n');
                        fprintf('---------------------\nParcel
Counter\n---------------------\n');
                        loop_raspi = 0;
                    otherwise
                        fprintf('Wrong Choice, you can only
enter 0!\n\n')
                end
            end
        otherwise
            fprintf('Wrong Choice, you can only enter a number
between 0 and 2!\n\n')
    end
end
```

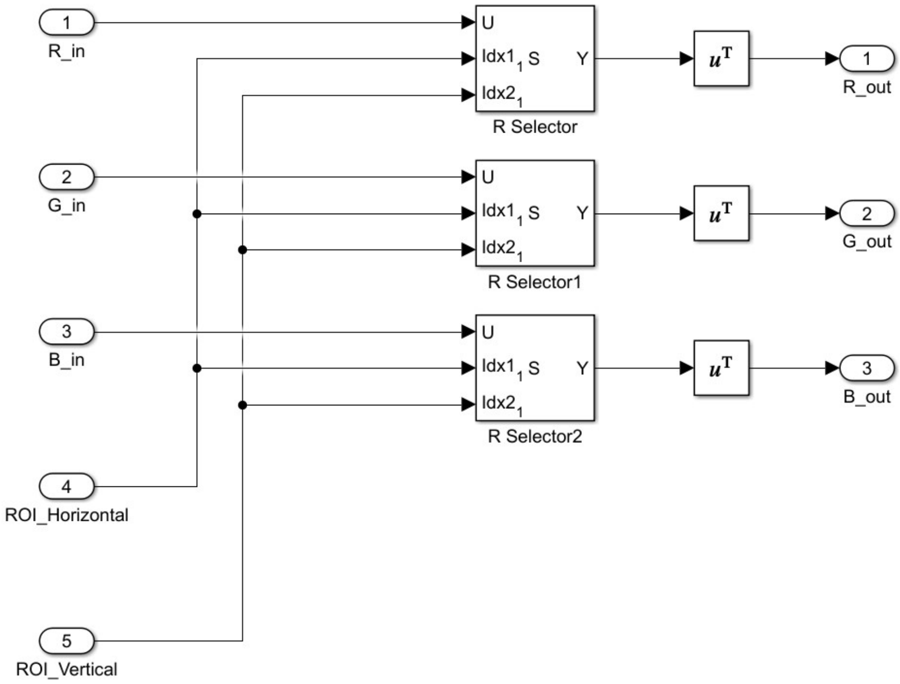
Appendix C Simulink Model including Subsystems



ROI + Uniting + Color Detection



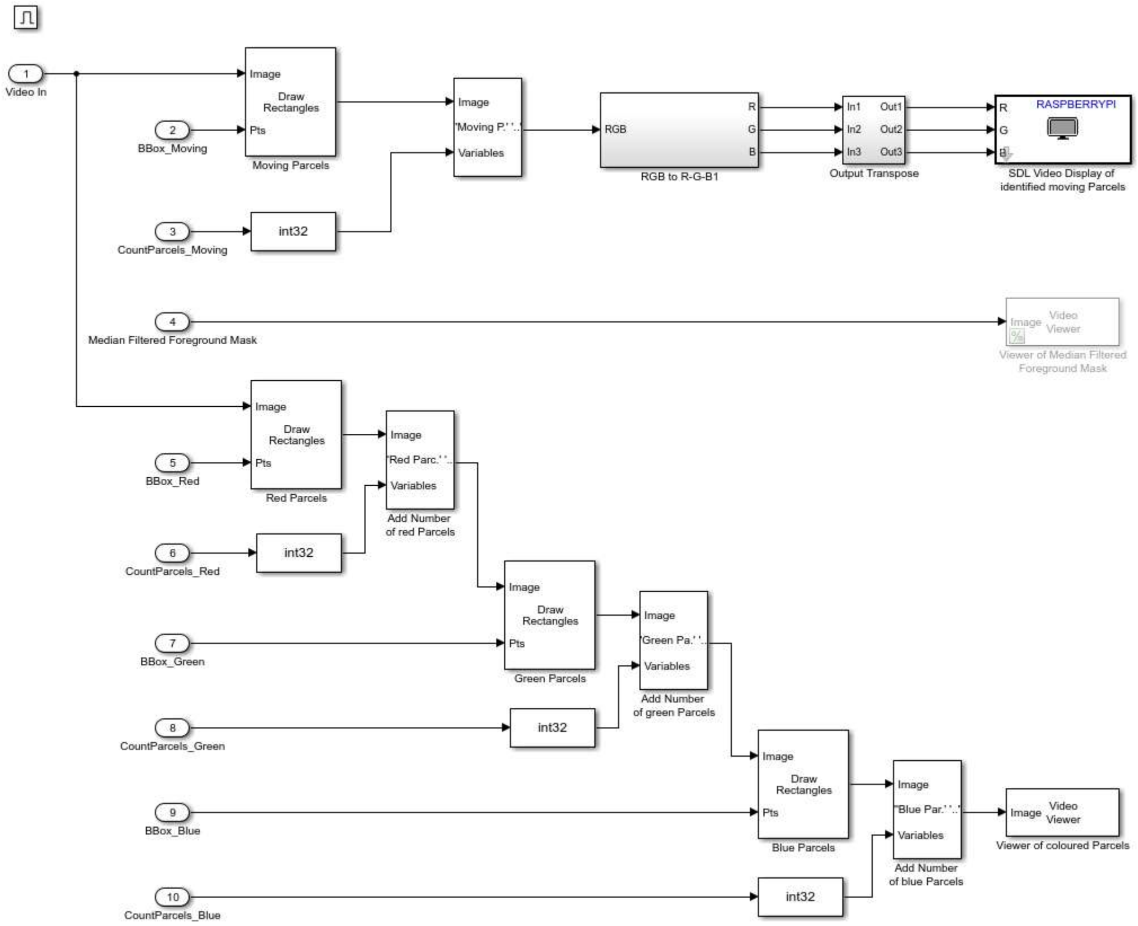
Input Selection



DetectColouredObjects

```
function [r_bw, g_bw, b_bw]  = DetectColouredObjects(r_in, g_in, 
b_in)

r=r_in;
g=g_in;
b=b_in;
r_thresh = 65;
g_thresh = 35;
b_thresh = 55;

justRed = r - g/2 - b/2;
justGreen = g - r/2 - b/2;
justBlue = b - r/2 - g/2;

r_bw = justRed > r_thresh;
g_bw = justGreen > g_thresh;
b_bw = justBlue > b_thresh;
b_bw = justBlue > b_thresh;
```

Visualisation



Appendix D MATLAB Codes of the Data Processing and Information Visualisation System

Original:

MATLAB Program 1 (Throughput, Cycle Time, Transport Performance, Transport Utilisation)

```
%% API Data of the ThingSpeak-Channels

%READ
%ThingSpeak-Channel "Parcel Counter - Data Collection 1"
read1ChannelID = 545627;
read1APIKey = 'NWH0Z63Y4UVY5224';

readSpeedSnapFieldID = 1;
readLengthSnapFieldID = 2;
readLengthConvFieldID = 3;
readLengthParcelFieldID = 4;
readMassParcelFieldID = 5;
readMassProductAFieldID = 6;
readMassProductBFieldID = 7;
readMassProductCFieldID = 8;

%ThingSpeak-Channel "Parcel Counter - Data Collection 2"
read2ChannelID = 543414;
read2APIKey = '84YJ3SW86RRG89VY';

readCountMovingParcelsFieldID = 1;
readCountColouredParcelsFieldID = 2;
readCountRedParcelsFieldID = 3;
readCountGreenParcelsFieldID = 4;
readCountBlueParcelsFieldID = 5;
readTimeSnapshotFieldID = 6;

%WRITE
%ThingSpeak-Channel "Parcel Counter - Information Visualisation
1"
write1ChannelID = 531575;
write1APIKey = 'FL3ITV3CVCGHD21L';

writeThroughputFieldID = 1;
writeCycleTimeFieldID = 2;
writeTransPerformanceFieldID = 3;
writeTransportUtilisationFieldID = 4;

%ThingSpeak-Channel "Parcel Counter - Auxiliary Variables"
write2ChannelID = 562287;
write2APIKey = '8NZC8L29LVKQ68EH';
```

```
writeStopFieldID = 1;

%% Read Data

%ThingSpeak-Channel "Parcel Counter - Data Collection 1"
SpeedSnapshot =
thingSpeakRead(read1ChannelID,'Fields',readSpeedSnapFieldID,'Rea
dKey',read1APIKey);
LengthSnapshot =
thingSpeakRead(read1ChannelID,'Fields',readLengthSnapFieldID,'Re
adKey',read1APIKey);
LengthConveyor =
thingSpeakRead(read1ChannelID,'Fields',readLengthConvFieldID,'Re
adKey',read1APIKey);
LengthParcel =
thingSpeakRead(read1ChannelID,'Fields',readLengthParcelFieldID,'
ReadKey',read1APIKey);
MassParcel =
thingSpeakRead(read1ChannelID,'Fields',readMassParcelFieldID,'Re
adKey',read1APIKey);
MassProductA =
thingSpeakRead(read1ChannelID,'Fields',readMassProductAFieldID,'
ReadKey',read1APIKey);
MassProductB =
thingSpeakRead(read1ChannelID,'Fields',readMassProductBFieldID,'
ReadKey',read1APIKey);
MassProductC =
thingSpeakRead(read1ChannelID,'Fields',readMassProductCFieldID,'
ReadKey',read1APIKey);

%ThingSpeak-Channel "Parcel Counter - Data Collection 2"
CountMovingParcelsSnapshot =
thingSpeakRead(read2ChannelID,'Fields',readCountMovingParcelsFie
ldID,'ReadKey',read2APIKey);
CountColouredParcelsSnapshot =
thingSpeakRead(read2ChannelID,'Fields',readCountColouredParcelsF
ieldID,'ReadKey',read2APIKey);
CountRedParcelsSnapshot =
thingSpeakRead(read2ChannelID,'Fields',readCountRedParcelsFieldI
D,'ReadKey',read2APIKey);
CountGreenParcelsSnapshot =
thingSpeakRead(read2ChannelID,'Fields',readCountGreenParcelsFiel
dID,'ReadKey',read2APIKey);
CountBlueParcelsSnapshot =
thingSpeakRead(read2ChannelID,'Fields',readCountBlueParcelsField
ID,'ReadKey',read2APIKey);
TimeSnapshot =
thingSpeakRead(read2ChannelID,'Fields',readTimeSnapshotFieldID,'
ReadKey',read2APIKey);

%% Process Data
```

```
if (CountMovingParcelsSnapshot > 0)
    Throughput = CountColouredParcelsSnapshot / TimeSnapshot;
else
    Throughput = 0;
end

CycleTime = LengthConveyor / SpeedSnapshot;

DensityProductA = CountRedParcelsSnapshot / LengthSnapshot;
DensityProductB = CountGreenParcelsSnapshot / LengthSnapshot;
DensityProductC = CountBlueParcelsSnapshot / LengthSnapshot;
TransPerformance = DensityProductA * (MassParcel+MassProductA) +
...
    DensityProductB * (MassParcel+MassProductB) + ...
    DensityProductC * (MassParcel+MassProductC);

if (CountMovingParcelsSnapshot == 0 &&
CountColouredParcelsSnapshot > 0)
    Stop = 1;
else
    Stop = 0;
end

minDistancebtwParcels = 0.25 * LengthParcel;
maxThroughput = (1/(LengthParcel+minDistancebtwParcels)) *
SpeedSnapshot;
TransportUtilisation = Throughput / maxThroughput * 100;

%% Write Data

%ThingSpeak-Channel "Parcel Counter - Information Visualisation
1"
thingSpeakWrite(write1ChannelID,'Fields',...

[writeThroughputFieldID,writeCycleTimeFieldID,writeTransPerforma
nceFieldID,writeTransportUtilisationFieldID],...

'Values',[Throughput,CycleTime,TransPerformance,TransportUtilisa
tion],'WriteKey', write1APIKey);

%ThingSpeak-Channel "Parcel Counter - Auxiliary Variables"
thingSpeakWrite(write2ChannelID,'Fields',writeStopFieldID,'Value
s',Stop,'WriteKey', write2APIKey);
```

MATLAB Program 2 (Time Utilisation, Effectiveness, Availability, Overall Equipment Effectiveness)

```
%% API Data of the ThingSpeak-Channels

%READ
%ThingSpeak-Channel "Parcel Counter - Data Collection 2"
read2ChannelID = 543414;
read2APIKey = '84YJ3SW86RRG89VY';

readCountMovingParcelsFieldID = 1;
readTimeSnapshotFieldID = 6;
readBusyPlannedFieldID = 7;
readTransportingPlannedFieldID = 8;

%ThingSpeak-Channel "Parcel Counter - Information Visualisation
1"
read3ChannelID = 531575;
read3APIKey = '4T0LM0V4MDC8WSV1';

readThroughputFieldID = 1;

%ThingSpeak-Channel "Parcel Counter - Auxiliary Variables"
read4ChannelID = 562287;
read4APIKey = 'R2DSMUJH484L2VP2';

readStopFieldID = 1;

%WRITE
%ThingSpeak-Channel "Parcel Counter - Information Visualisation
2"
write2ChannelID = 548839;
write2APIKey = 'KUR5932UVR876LXC';

writeTimeUtilisationFieldID = 1;
writeEffectivenessFieldID = 2;
writeAvailabilityFieldID = 3;
writeOEEFieldID = 4;
writeAverageThroughputFieldID = 5;

%% Read Data

%ThingSpeak-Channel "Parcel Counter - Data Collection 2"
CountMovingParcelsSnapshot60min =
thingSpeakRead(read2ChannelID,'Fields',readCountMovingParcelsFie
ldID,'NumMinutes',60,'ReadKey',read2APIKey);
TimeSnapshot =
thingSpeakRead(read2ChannelID,'Fields',readTimeSnapshotFieldID,'
ReadKey',read2APIKey);
```

```
BusyPlanned =
thingSpeakRead(read2ChannelID,'Fields',readBusyPlannedFieldID,'R
eadKey',read2APIKey);
TransPlanned =
thingSpeakRead(read2ChannelID,'Fields',readTransportingPlannedFi
eldID,'ReadKey',read2APIKey);

%ThingSpeak-Channel "Parcel Counter - Information Visualisation
1"
Throughput60min =
thingSpeakRead(read3ChannelID,'Fields',readThroughputFieldID,'Nu
mMinutes',60,'ReadKey',read3APIKey);

%ThingSpeak-Channel "Parcel Counter - Auxiliary Variables"
Stops60min =
thingSpeakRead(read4ChannelID,'Fields',readStopFieldID,'NumMinut
es',60,'ReadKey',read4APIKey);

%Actual Down Time is assumed to be zero
ActualDownTime = 0;

%% Process Data

sumStop = sum(Stops60min) - Stops60min(1);
bwTrans = CountMovingParcelsSnapshot60min > 0;
sizebwTrans = size(bwTrans);
ActualBusyTime = (sizebwTrans(1) - 1 - sumStop) * TimeSnapshot;
OperationTime = (sizebwTrans(1) - 1) * TimeSnapshot;
PlannedBusyTime = OperationTime * BusyPlanned / 100;
sumTrans = sum(bwTrans) - bwTrans(1);
ActualTransTime = (sumTrans/(sizebwTrans(1) - 1 - sumStop)) *
ActualBusyTime;
TimeUtilisation = ActualTransTime / OperationTime * 100;

PlannedTransTime = PlannedBusyTime * TransPlanned / 100;
Effectiveness = ActualTransTime / PlannedTransTime * 100;

Availability = ActualTransTime / PlannedBusyTime * 100;

OEE = Effectiveness * Availability / 100;

AverageThroughput = mean(Throughput60min);

%% Write Data

%ThingSpeak-Channel "Parcel Counter - Information Visualisation
1"
thingSpeakWrite(write2ChannelID,'Fields',...

[writeTimeUtilisationFieldID,writeEffectivenessFieldID,writeAvai
labilityFieldID,writeOEEFieldID,writeAverageThroughputFieldID],.
..
'Values',[TimeUtilisation,Effectiveness,Availability,OEE,Average
Throughput],'WriteKey', write2APIKey);
```

Updated:

MATLAB Program 1 (Throughput)

```
%% API Data of the ThingSpeak-Channels

%READ
%ThingSpeak-Channel "Parcel Counter - Data Collection 2"
read2ChannelID = 543414;
read2APIKey = '84YJ3SW86RRG89VY';

readCountMovingParcelsFieldID = 1;
readCountColouredParcelsFieldID = 2;
readTimeSnapshotFieldID = 6;

%% Read Data
%ThingSpeak-Channel "Parcel Counter - Data Collection 2"

[CountMovingParcelsSnapshotPeriod,timeMoving] =
thingSpeakRead(read2ChannelID,'Fields',readCountMovingParcelsFie
ldID,'NumMinutes',60,'ReadKey',read2APIKey);
[CountColouredParcelsSnapshotPeriod,timeColoured] =
thingSpeakRead(read2ChannelID,'Fields',readCountColouredParcelsF
ieldID,'NumMinutes',60,'ReadKey',read2APIKey);
TimeSnapshot =
thingSpeakRead(read2ChannelID,'Fields',readTimeSnapshotFieldID,'
ReadKey',read2APIKey);

%% Process Data

MovingDetection = CountMovingParcelsSnapshotPeriod > 0;

ThroughputColoredParcelsPeriod =
CountColouredParcelsSnapshotPeriod / TimeSnapshot * 60;

sizeMoving = size(MovingDetection);
sizeColoured = size(ThroughputColoredParcelsPeriod);
time = timeMoving;

if sizeMoving(1)>sizeColoured(1)
    time = timeColoured;
    for i=(sizeColoured(1)+1):sizeMoving(1)
        MovingDetection(sizeColoured(1)+1)=[];
    end
end
```

```
if sizeColoured(1)>sizeMoving(1)
    time = timeMoving;
    for i=(sizeMoving(1)+1):sizeColoured(1)
        ThroughputColoredParcelsPeriod(sizeMoving(1)+1)=[];
    end
end

ThroughputColoredParcelsPeriodFINAL =
ThroughputColoredParcelsPeriod .* MovingDetection;

%% Visualize Data

thingSpeakPlot(time,
ThroughputColoredParcelsPeriodFINAL,'LineStyle','-
','LineWidth',2,'Color','red','xlabel','Time','ylabel','Throughp
ut [units/min]','title','Throughput of
Parcles','Grid','on','Marker','o');
```

MATLAB Program 2 (Cycle Time)

```
%% API Data of the ThingSpeak-Channels

%READ
%ThingSpeak-Channel "Parcel Counter - Data Collection 1"
read1ChannelID = 545627;
read1APIKey = 'NWH0Z63Y4UVY5224';

readSpeedSnapFieldID = 1;
readLengthConvFieldID = 3;

%% Read Data
%ThingSpeak-Channel "Parcel Counter - Data Collection 2"

SpeedSnapshotPeriod =
thingSpeakRead(read1ChannelID,'Fields',readSpeedSnapFieldID,'Rea
dKey',read1APIKey);
[LengthConveyorPeriod,time] =
thingSpeakRead(read1ChannelID,'Fields',readLengthConvFieldID,'Nu
mMinutes',60,'ReadKey',read1APIKey);

%% Process Data

CycleTimePeriod = LengthConveyorPeriod / SpeedSnapshotPeriod;

%% Visualize Data

thingSpeakPlot(time,CycleTimePeriod,'LineStyle','-
','LineWidth',2,'Color','blue','xlabel','Time','ylabel','Cycle
Time [s]','title','Cycle Time of
Parcels','Grid','on','Marker','o');
```

MATLAB Program 3 (Transport Performance)

```
%% API Data of the ThingSpeak-Channels

%READ
%ThingSpeak-Channel "Parcel Counter - Data Collection 1"
read1ChannelID = 545627;
read1APIKey = 'NWH0Z63Y4UVY5224';

readLengthSnapFieldID = 2;
readMassParcelFieldID = 5;
readMassProductAFieldID = 6;
readMassProductBFieldID = 7;
readMassProductCFieldID = 8;

%ThingSpeak-Channel "Parcel Counter - Data Collection 2"
read2ChannelID = 543414;
read2APIKey = '84YJ3SW86RRG89VY';

readCountRedParcelsFieldID = 3;
readCountGreenParcelsFieldID = 4;
readCountBlueParcelsFieldID = 5;

%% Read Data
%ThingSpeak-Channel "Parcel Counter - Data Collection 1"
LengthSnapshot =
thingSpeakRead(read1ChannelID,'Fields',readLengthSnapFieldID,'Re
adKey',read1APIKey);
MassParcel =
thingSpeakRead(read1ChannelID,'Fields',readMassParcelFieldID,'Re
adKey',read1APIKey);
MassProductA =
thingSpeakRead(read1ChannelID,'Fields',readMassProductAFieldID,'
ReadKey',read1APIKey);
MassProductB =
thingSpeakRead(read1ChannelID,'Fields',readMassProductBFieldID,'
ReadKey',read1APIKey);
MassProductC =
thingSpeakRead(read1ChannelID,'Fields',readMassProductCFieldID,'
ReadKey',read1APIKey);

%ThingSpeak-Channel "Parcel Counter - Data Collection 2"
[CountRedParcelsSnapshotPeriod,timeA] =
thingSpeakRead(read2ChannelID,'Fields',readCountRedParcelsFieldI
D,'NumMinutes',60,'ReadKey',read2APIKey);
[CountGreenParcelsSnapshotPeriod,timeB] =
thingSpeakRead(read2ChannelID,'Fields',readCountGreenParcelsFiel
dID,'NumMinutes',60,'ReadKey',read2APIKey);
[CountBlueParcelsSnapshotPeriod,timeC] =
thingSpeakRead(read2ChannelID,'Fields',readCountBlueParcelsField
ID,'NumMinutes',60,'ReadKey',read2APIKey);
```

```
%% Process Data

sizeA = size(CountRedParcelsSnapshotPeriod);
sizeB = size(CountGreenParcelsSnapshotPeriod);
sizeC = size(CountBlueParcelsSnapshotPeriod);
time = timeA;

if sizeA(1)>sizeB(1)
    time = timeB;
    for i=(sizeB(1)+1):sizeA(1)
        CountRedParcelsSnapshotPeriod(sizeB(1)+1)=[];
    end
end

if sizeB(1)>sizeA(1)
    time = timeA;
    for i=(sizeA(1)+1):sizeB(1)
        CountGreenParcelsSnapshotPeriod(sizeA(1)+1)=[];
    end
end

sizeA = size(CountRedParcelsSnapshotPeriod);
sizeB = size(CountGreenParcelsSnapshotPeriod);

if sizeA(1)>sizeC(1)
    time = timeC;
    for i=(sizeC(1)+1):sizeA(1)
        CountRedParcelsSnapshotPeriod(sizeC(1)+1)=[];
    end
end

if sizeC(1)>sizeA(1)
    time = timeA;
    for i=(sizeA(1)+1):sizeC(1)
        CountBlueParcelsSnapshotPeriod(sizeA(1)+1)=[];
    end
end

sizeA = size(CountRedParcelsSnapshotPeriod);
sizeC = size(CountBlueParcelsSnapshotPeriod);

if sizeB(1)>sizeC(1)
    time = timeC;
    for i=(sizeC(1)+1):sizeB(1)
        CountGreenParcelsSnapshotPeriod(sizeC(1)+1)=[];
    end
end

if sizeC(1)>sizeB(1)
    time = timeB;
    for i=(sizeB(1)+1):sizeC(1)
        CountBlueParcelsSnapshotPeriod(sizeB(1)+1)=[];
    end
```

```
end

DensityProductAPeriod = CountRedParcelsSnapshotPeriod /
LengthSnapshot;
DensityProductBPeriod = CountGreenParcelsSnapshotPeriod /
LengthSnapshot;
DensityProductCPeriod = CountBlueParcelsSnapshotPeriod /
LengthSnapshot;
TransPerformancePeriod = (DensityProductAPeriod *
(MassParcel+MassProductA)) + ...
    (DensityProductBPeriod * (MassParcel+MassProductB)) + ...
    (DensityProductCPeriod * (MassParcel+MassProductC));

%% Visualize Data

thingSpeakPlot(time, TransPerformancePeriod,'LineStyle','-
','LineWidth',2,'Color','red','xlabel','Time','ylabel','Transpor
t Performance [kg/m]','title','Transport Performance of the
Conveyor System','Grid','on','Marker','o');
```

MATLAB Program 4 (Transport Utilisation)

```
%% API Data of the ThingSpeak-Channels

%READ
%ThingSpeak-Channel "Parcel Counter - Data Collection 1"
read1ChannelID = 545627;
read1APIKey = 'NWH0Z63Y4UVY5224';

readSpeedSnapFieldID = 1;
readLengthParcelFieldID = 4;

%ThingSpeak-Channel "Parcel Counter - Data Collection 2"
read2ChannelID = 543414;
read2APIKey = '84YJ3SW86RRG89VY';

readCountMovingParcelsFieldID = 1;
readCountColouredParcelsFieldID = 2;
readTimeSnapshotFieldID = 6;

%% Read Data
%ThingSpeak-Channel "Parcel Counter - Data Collection 1"

SpeedSnapshot =
thingSpeakRead(read1ChannelID,'Fields',readSpeedSnapFieldID,'Rea
dKey',read1APIKey);
LengthParcel =
thingSpeakRead(read1ChannelID,'Fields',readLengthParcelFieldID,'
ReadKey',read1APIKey);

%ThingSpeak-Channel "Parcel Counter - Data Collection 2"
```

```
[CountMovingParcelsSnapshotPeriod,timeMoving] =
thingSpeakRead(read2ChannelID,'Fields',readCountMovingParcelsFie
ldID,'NumMinutes',60,'ReadKey',read2APIKey);
[CountColouredParcelsSnapshotPeriod,timeColoured] =
thingSpeakRead(read2ChannelID,'Fields',readCountColouredParcelsF
ieldID,'NumMinutes',60,'ReadKey',read2APIKey);
TimeSnapshot =
thingSpeakRead(read2ChannelID,'Fields',readTimeSnapshotFieldID,'
ReadKey',read2APIKey);

%% Process Data

MovingDetection = CountMovingParcelsSnapshotPeriod > 0;

ThroughputColoredParcelsPeriod =
CountColouredParcelsSnapshotPeriod / TimeSnapshot * 60;

sizeMoving = size(MovingDetection);
sizeColoured = size(ThroughputColoredParcelsPeriod);
time = timeMoving;

if sizeMoving(1)>sizeColoured(1)
    time = timeColoured;
    for i=(sizeColoured(1)+1):sizeMoving(1)
        MovingDetection(sizeColoured(1)+1)=[];
    end
end

if sizeColoured(1)>sizeMoving(1)
    time = timeMoving;
    for i=(sizeMoving(1)+1):sizeColoured(1)
        ThroughputColoredParcelsPeriod(sizeMoving(1)+1)=[];
    end
end

ThroughputColoredParcelsPeriodFINAL =
ThroughputColoredParcelsPeriod .* MovingDetection;

minDistancebtwParcels = 0.25 * LengthParcel;
maxThroughput = (1/(LengthParcel+minDistancebtwParcels)) *
SpeedSnapshot * 60;
TransportUtilisationPeriod = ThroughputColoredParcelsPeriodFINAL
./ maxThroughput * 100;

%% Visualize Data

thingSpeakPlot(time, TransportUtilisationPeriod,'LineStyle','-
','LineWidth',2,'Color','red','xlabel','Time','ylabel','Transpor
t Urilisation [%]','title','Transport Utilisation of the
Conveyor System','Grid','on','Marker','o');
```

MATLAB Program 5 (Time Utilisation, Effectiveness, Availability, Overall Equipment Effectiveness)

```
%% API Data of the ThingSpeak-Channels

%READ
%ThingSpeak-Channel "Parcel Counter - Data Collection 2"
read2ChannelID = 543414;
read2APIKey = '84YJ3SW86RRG89VY';

readCountMovingParcelsFieldID = 1;
readTimeSnapshotFieldID = 6;
readBusyPlannedFieldID = 7;
readTransportingPlannedFieldID = 8;

%ThingSpeak-Channel "Parcel Counter - Information Visualisation
1"
read3ChannelID = 531575;
read3APIKey = '4T0LM0V4MDC8WSV1';

readThroughputFieldID = 1;

%ThingSpeak-Channel "Parcel Counter - Auxiliary Variables"
read4ChannelID = 562287;
read4APIKey = 'R2DSMUJH484L2VP2';

readStopFieldID = 1;

%WRITE
%ThingSpeak-Channel "Parcel Counter - Information Visualisation
2"
write2ChannelID = 548839;
write2APIKey = 'KUR5932UVR876LXC';

writeTimeUtilisationFieldID = 1;
writeEffectivenessFieldID = 2;
writeAvailabilityFieldID = 3;
writeOEEFieldID = 4;
writeAverageThroughputFieldID = 5;

%% Read Data

%ThingSpeak-Channel "Parcel Counter - Data Collection 2"
CountMovingParcelsSnapshotPeriode =
thingSpeakRead(read2ChannelID,'Fields',readCountMovingParcelsFie
ldID,'NumMinutes',5,'ReadKey',read2APIKey);
TimeSnapshot =
thingSpeakRead(read2ChannelID,'Fields',readTimeSnapshotFieldID,'
ReadKey',read2APIKey);
```

```
BusyPlanned =
thingSpeakRead(read2ChannelID,'Fields',readBusyPlannedFieldID,'R
eadKey',read2APIKey);
TransPlanned =
thingSpeakRead(read2ChannelID,'Fields',readTransportingPlannedFi
eldID,'ReadKey',read2APIKey);

%ThingSpeak-Channel "Parcel Counter - Information Visualisation
1"
%ThroughputPeriode =
thingSpeakRead(read3ChannelID,'Fields',readThroughputFieldID,'Nu
mMinutes',5,'ReadKey',read3APIKey);

%ThingSpeak-Channel "Parcel Counter - Auxiliary Variables"
StopsPeriode =
thingSpeakRead(read4ChannelID,'Fields',readStopFieldID,'NumMinut
es',5,'ReadKey',read4APIKey);

%% Process Data

sumStop = sum(StopsPeriode) - StopsPeriode(1);
bwTrans = CountMovingParcelsSnapshotPeriode > 0;
sizebwTrans = size(bwTrans);
ActualBusyTime = (sizebwTrans(1) - 1 - sumStop) * TimeSnapshot;
OperationTime = (sizebwTrans(1) - 1) * TimeSnapshot;
PlannedBusyTime = OperationTime * BusyPlanned / 100;
sumTrans = sum(bwTrans) - bwTrans(1);
ActualTransTime = (sumTrans/(sizebwTrans(1) - 1 - sumStop)) *
ActualBusyTime;
TimeUtilisation = ActualTransTime / OperationTime * 100;

PlannedTransTime = PlannedBusyTime * TransPlanned / 100;
Effectiveness = ActualTransTime / PlannedTransTime * 100;

Availability = ActualTransTime / PlannedBusyTime * 100;

OEE = Effectiveness * Availability / 100;

%AverageThroughput = mean(ThroughputPeriode);

%% Write Data

%ThingSpeak-Channel "Parcel Counter - Information Visualisation
1"
thingSpeakWrite(write2ChannelID,'Fields',...

[writeTimeUtilisationFieldID,writeEffectivenessFieldID,writeAvai
labilityFieldID,writeOEEFieldID],...

'Values',[TimeUtilisation,Effectiveness,Availability,OEE],'Write
Key', write2APIKey);
```

MATLAB Program 6 (Detecting Stops)

```
%% API Data of the ThingSpeak-Channels

%READ
%ThingSpeak-Channel "Parcel Counter - Data Collection 2"
read2ChannelID = 543414;
read2APIKey = '84YJ3SW86RRG89VY';

readCountMovingParcelsFieldID = 1;
readCountColouredParcelsFieldID = 2;

%WRITE
%ThingSpeak-Channel "Parcel Counter - Auxiliary Variables"
write2ChannelID = 562287;
write2APIKey = '8NZC8L29LVKQ68EH';

writeStopFieldID = 1;

%% Read Data

%ThingSpeak-Channel "Parcel Counter - Data Collection 2"
CountMovingParcelsSnapshot =
thingSpeakRead(read2ChannelID,'Fields',readCountMovingParcelsFie
ldID,'ReadKey',read2APIKey);
CountColouredParcelsSnapshot =
thingSpeakRead(read2ChannelID,'Fields',readCountColouredParcelsF
ieldID,'ReadKey',read2APIKey);

%% Process Data

if (CountMovingParcelsSnapshot == 0 &&
CountColouredParcelsSnapshot > 0)
    Stop = 1;
else
    Stop = 0;
end

%% Write Data

%ThingSpeak-Channel "Parcel Counter - Auxiliary Variables"
thingSpeakWrite(write2ChannelID,'Fields',[writeStopFieldID],'Val
ues',[Stop],'WriteKey', write2APIKey);
```

Additional Information:

Visualisation 1 (Average Throughput – 3 Data Points)

```
%% API Data of the ThingSpeak-Channels
```

```
%READ
%ThingSpeak-Channel "Parcel Counter - Data Collection 2"
read2ChannelID = 543414;
read2APIKey = '84YJ3SW86RRG89VY';

readCountMovingParcelsFieldID = 1;
readCountColouredParcelsFieldID = 2;
readTimeSnapshotFieldID = 6;

%% Read Data
%ThingSpeak-Channel "Parcel Counter - Data Collection 2"

[CountMovingParcelsSnapshotPeriod,timeMoving] =
thingSpeakRead(read2ChannelID,'Fields',readCountMovingParcelsFie
ldID,'NumMinutes',60,'ReadKey',read2APIKey);
[CountColouredParcelsSnapshotPeriod,timeColoured] =
thingSpeakRead(read2ChannelID,'Fields',readCountColouredParcelsF
ieldID,'NumMinutes',60,'ReadKey',read2APIKey);
TimeSnapshot =
thingSpeakRead(read2ChannelID,'Fields',readTimeSnapshotFieldID,'
ReadKey',read2APIKey);

%% Process Data

MovingDetection = CountMovingParcelsSnapshotPeriod > 0;

ThroughputColoredParcelsPeriod =
CountColouredParcelsSnapshotPeriod / TimeSnapshot * 60;

sizeMoving = size(MovingDetection);
sizeColoured = size(ThroughputColoredParcelsPeriod);
time = timeMoving;

if sizeMoving(1)>sizeColoured(1)
    time = timeColoured;
    for i=(sizeColoured(1)+1):sizeMoving(1)
        MovingDetection(sizeColoured(1)+1)=[];
    end
end

if sizeColoured(1)>sizeMoving(1)
    time = timeMoving;
    for i=(sizeMoving(1)+1):sizeColoured(1)
        ThroughputColoredParcelsPeriod(sizeMoving(1)+1)=[];
    end
end

ThroughputColoredParcelsPeriodFINAL =
ThroughputColoredParcelsPeriod .* MovingDetection;

x = size(CountMovingParcelsSnapshotPeriod);
```

```
AverageThroughputColoredParcelsPeriodFINAL =
ThroughputColoredParcelsPeriodFINAL;

AverageThroughputColoredParcelsPeriodFINAL(1)=...

(ThroughputColoredParcelsPeriodFINAL(1)+ThroughputColoredParcels
PeriodFINAL(2))/2;

AverageThroughputColoredParcelsPeriodFINAL(x(1))=...

(ThroughputColoredParcelsPeriodFINAL(x(1))+ThroughputColoredParc
elsPeriodFINAL(x(1)-1))/2;

for i= 2:(x(1)-1)
    AverageThroughputColoredParcelsPeriodFINAL(i)=...
        (ThroughputColoredParcelsPeriodFINAL(i-
1)+ThroughputColoredParcelsPeriodFINAL(i)+...
        ThroughputColoredParcelsPeriodFINAL(i+1))/3;
end

%% Visualize Data

thingSpeakPlot(time,
AverageThroughputColoredParcelsPeriodFINAL,'LineStyle','-
','LineWidth',2,'Color','red','xlabel','Time','ylabel','Throughp
ut [units/min]','title','Throughput of Parcles (Average of 3
Data Points)','Grid','on','Marker','o');
```

Visualisation 2 (Average Throughput – 5 Data Points)

```
%% API Data of the ThingSpeak-Channels

%READ
%ThingSpeak-Channel "Parcel Counter - Data Collection 2"
read2ChannelID = 543414;
read2APIKey = '84YJ3SW86RRG89VY';

readCountMovingParcelsFieldID = 1;
readCountColouredParcelsFieldID = 2;
readTimeSnapshotFieldID = 6;

%% Read Data
%ThingSpeak-Channel "Parcel Counter - Data Collection 2"

[CountMovingParcelsSnapshotPeriod,timeMoving] =
thingSpeakRead(read2ChannelID,'Fields',readCountMovingParcelsFie
ldID,'NumMinutes',60,'ReadKey',read2APIKey);
[CountColouredParcelsSnapshotPeriod,timeColoured] =
thingSpeakRead(read2ChannelID,'Fields',readCountColouredParcelsF
ieldID,'NumMinutes',60,'ReadKey',read2APIKey);
```

```
TimeSnapshot =
thingSpeakRead(read2ChannelID,'Fields',readTimeSnapshotFieldID,'
ReadKey',read2APIKey);

%% Process Data

MovingDetection = CountMovingParcelsSnapshotPeriod > 0;

ThroughputColoredParcelsPeriod =
CountColouredParcelsSnapshotPeriod / TimeSnapshot * 60;

sizeMoving = size(MovingDetection);
sizeColoured = size(ThroughputColoredParcelsPeriod);
time = timeMoving;

if sizeMoving(1)>sizeColoured(1)
    time = timeColoured;
    for i=(sizeColoured(1)+1):sizeMoving(1)
        MovingDetection(sizeColoured(1)+1)=[];
    end
end

if sizeColoured(1)>sizeMoving(1)
    time = timeMoving;
    for i=(sizeMoving(1)+1):sizeColoured(1)
        ThroughputColoredParcelsPeriod(sizeMoving(1)+1)=[];
    end
end

ThroughputColoredParcelsPeriodFINAL =
ThroughputColoredParcelsPeriod .* MovingDetection;

x = size(CountMovingParcelsSnapshotPeriod);

AverageThroughputColoredParcelsPeriodFINAL =
ThroughputColoredParcelsPeriodFINAL;

AverageThroughputColoredParcelsPeriodFINAL(1)=...

(ThroughputColoredParcelsPeriodFINAL(1)+ThroughputColoredParcels
PeriodFINAL(2)+...
    ThroughputColoredParcelsPeriodFINAL(3))/3;

AverageThroughputColoredParcelsPeriodFINAL(2)=...

(ThroughputColoredParcelsPeriodFINAL(1)+ThroughputColoredParcels
PeriodFINAL(2)+...

ThroughputColoredParcelsPeriodFINAL(3)+ThroughputColoredParcelsP
eriodFINAL(4))/4;

AverageThroughputColoredParcelsPeriodFINAL(x(1))=...
```

```
(ThroughputColoredParcelsPeriodFINAL(x(1))+ThroughputColoredParc
elsPeriodFINAL(x(1)-1)+...
    ThroughputColoredParcelsPeriodFINAL(x(1)-2))/3;

AverageThroughputColoredParcelsPeriodFINAL(x(1)-1)=...

(ThroughputColoredParcelsPeriodFINAL(x(1))+ThroughputColoredParc
elsPeriodFINAL(x(1)-1)+...
    ThroughputColoredParcelsPeriodFINAL(x(1)-
2)+ThroughputColoredParcelsPeriodFINAL(x(1)-3))/4;

for i= 3:(x(1)-2)
    AverageThroughputColoredParcelsPeriodFINAL(i)=...
        (ThroughputColoredParcelsPeriodFINAL(i-2)+...
        ThroughputColoredParcelsPeriodFINAL(i-1)+...
        ThroughputColoredParcelsPeriodFINAL(i)+...
        ThroughputColoredParcelsPeriodFINAL(i+1)+...
        ThroughputColoredParcelsPeriodFINAL(i+2))/5;
end

%% Visualize Data

thingSpeakPlot(time,
AverageThroughputColoredParcelsPeriodFINAL,'LineStyle','-
','LineWidth',2,'Color','red','xlabel','Time','ylabel','Throughp
ut [units/min]','title','Throughput of Parcles (Average of 5
Data Points)','Grid','on','Marker','o');
```

Visualisation 3 (Conditions of the Conveyor System)

```
%% API Data of the ThingSpeak-Channels
%ThingSpeak-Channel "Parcel Counter - Data Collection 2"

read2ChannelID = 543414;
read2APIKey = '84YJ3SW86RRG89VY';

readCountMovingParcelsFieldID = 1;
readCountColouredParcelsFieldID = 2;

%% Read Data
%ThingSpeak-Channel "Parcel Counter - Data Collection 2"

[CountMovingParcelsSnapshotPeriod,timeMoving] =
thingSpeakRead(read2ChannelID,'Fields',readCountMovingParcelsFie
ldID,'NumMinutes',60,'ReadKey',read2APIKey);
[CountColouredParcelsSnapshotPeriod,timeColoured] =
thingSpeakRead(read2ChannelID,'Fields',readCountColouredParcelsF
ieldID,'NumMinutes',60,'ReadKey',read2APIKey);

%% Process Data
```

```
sizeMoving = size(CountMovingParcelsSnapshotPeriod);
sizeColoured = size(CountColouredParcelsSnapshotPeriod);
time = timeMoving;

if sizeMoving(1)>sizeColoured(1)
    time = timeColoured;
    for i=(sizeColoured(1)+1):sizeMoving(1)
        CountMovingParcelsSnapshotPeriod(sizeColoured(1)+1)=[];
    end
end

if sizeColoured(1)>sizeMoving(1)
    time = timeMoving;
    for i=(sizeMoving(1)+1):sizeColoured(1)
        CountColouredParcelsSnapshotPeriod(sizeMoving(1)+1)=[];
    end
end

bwRunning = CountMovingParcelsSnapshotPeriod > 0;

bwStop = (CountMovingParcelsSnapshotPeriod == 0 &
CountColouredParcelsSnapshotPeriod > 0);

bwUnknown = (CountMovingParcelsSnapshotPeriod == 0 &
CountColouredParcelsSnapshotPeriod == 0);

data = [bwRunning,bwStop,bwUnknown];

%% Visualize Data

thingSpeakPlot(time,
data,'LineWidth',2,'xlabel','Time','ylabel','System
Condition','title','Throughput over last 60
min','Legend',{'Running','Stop','Unknown'},'Grid','on');
```

Visualisation 4 (Proportion of red, green and blue parcels)

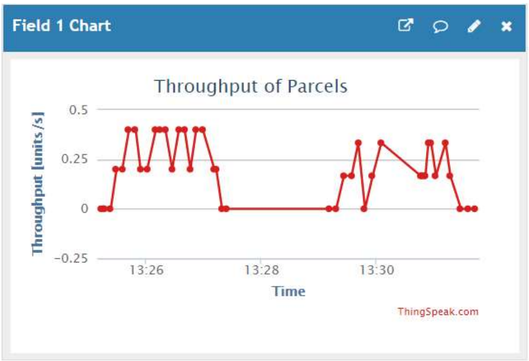
```
%% API Data of the ThingSpeak-Channels

%READ
%ThingSpeak-Channel "Parcel Counter - Data Collection 2"
read2ChannelID = 543414;
read2APIKey = '84YJ3SW86RRG89VY';

readCountRedParcelsFieldID = 3;
readCountGreenParcelsFieldID = 4;
readCountBlueParcelsFieldID = 5;

%% Read Data
```

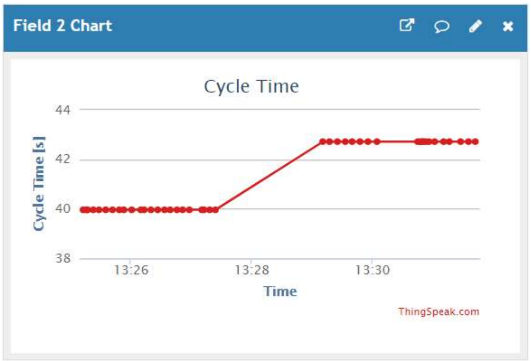
```
CountRedParcelsSnapshotPeriode =
thingSpeakRead(read2ChannelID,'Fields',readCountRedParcelsFieldI
D,'NumMinutes',60,'ReadKey',read2APIKey);
CountGreenParcelsSnapshotPeriode =
thingSpeakRead(read2ChannelID,'Fields',readCountGreenParcelsFiel
dID,'NumMinutes',60,'ReadKey',read2APIKey);
CountBlueParcelsSnapshotPeriode =
thingSpeakRead(read2ChannelID,'Fields',readCountBlueParcelsField
ID,'NumMinutes',60,'ReadKey',read2APIKey);

%% Analyse Data
AverageRedParcels = mean(CountRedParcelsSnapshotPeriode);
AverageGreenParcels = mean(CountGreenParcelsSnapshotPeriode);
AverageBlueParcels = mean(CountBlueParcelsSnapshotPeriode);

TotalParcels = AverageRedParcels + AverageGreenParcels +
AverageBlueParcels;

PercentageRedParcels = AverageRedParcels / TotalParcels;
PercentageGreenParcels = AverageGreenParcels / TotalParcels;
PercentageBlueParcels = AverageBlueParcels / TotalParcels;

%% Visualise Data
X = [PercentageRedParcels PercentageGreenParcels
PercentageBlueParcels];

h = pie(X);

hText = findobj(h,'Type','text');
percentValues = get(hText,'String');

txt = {'Red Parcels: ';'Green Parcels : ';'Blue Parcels: '};
combinedtxt = strcat(txt,percentValues);

hText(1).String = combinedtxt(1);
hText(2).String = combinedtxt(2);
hText(3).String = combinedtxt(3);

%Edit red text and patch
textRed = hText(1);
textRed.BackgroundColor = 'white';
textRed.EdgeColor = 'black';
textRed.Color = 'black';
textRed.FontSize = 9;

patchRed = h(1);
patchRed.FaceColor = 'red';

%Edit green text and patch
textGreen = hText(2);
textGreen.BackgroundColor = 'white';
textGreen.EdgeColor = 'black';
textGreen.Color = 'black';
textGreen.FontSize = 9;

patchGreen = h(3);
patchGreen.FaceColor = 'green';

%Edit blue text and patch
textBlue = hText(3);
textBlue.BackgroundColor = 'white';
textBlue.EdgeColor = 'black';
textBlue.Color = 'black';
textBlue.FontSize = 9;

patchBlue = h(5);
patchBlue.FaceColor = 'blue';

title('Proportion of coloured Parcels - Average of last 60
min','Color','black','FontSize',11);
```

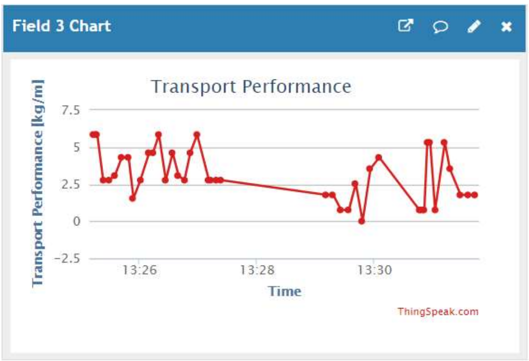
Appendix E Visualisation of the KPIs and additional Information

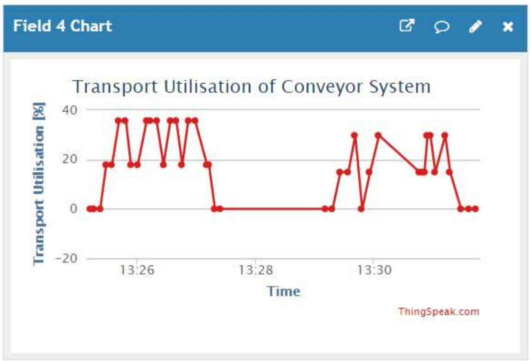
Original:

ThingSpeak Channel 1

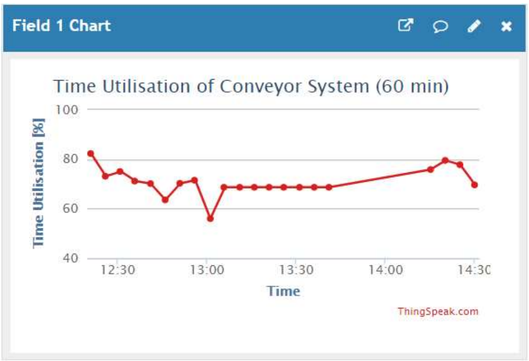


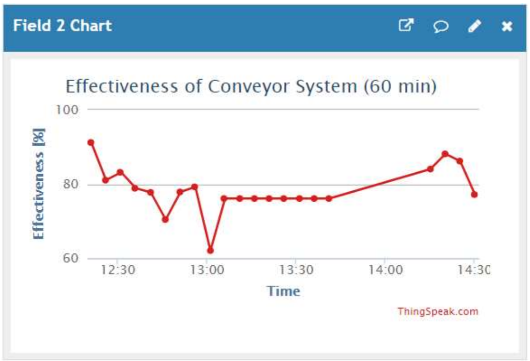


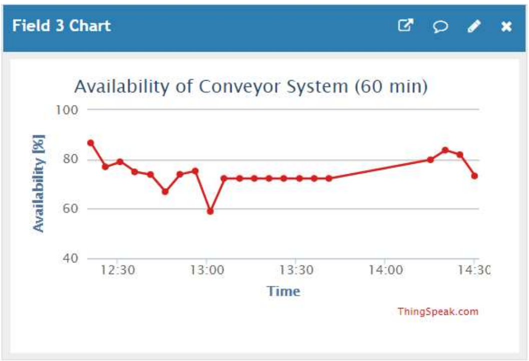


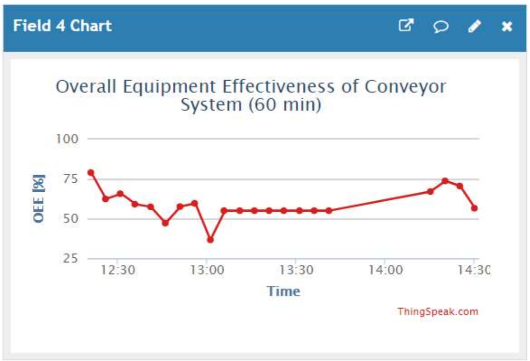


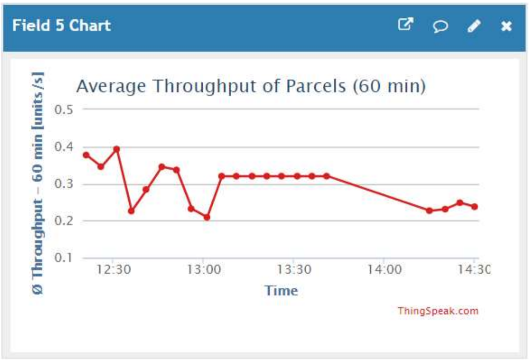
ThingSpeak Channel 2

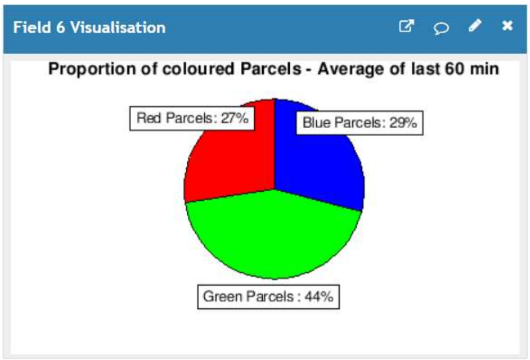


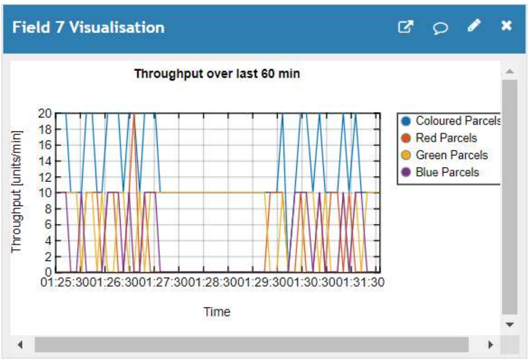


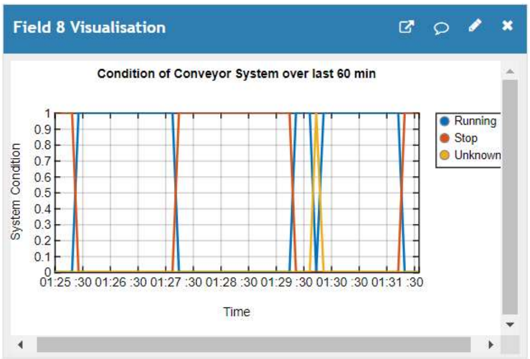






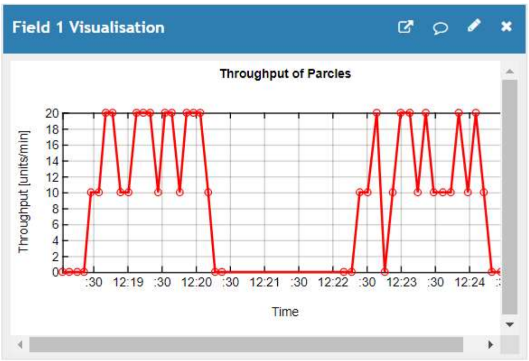


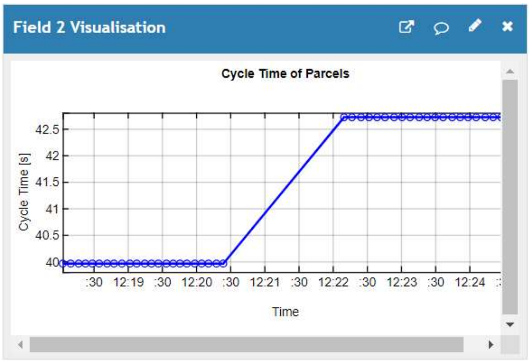


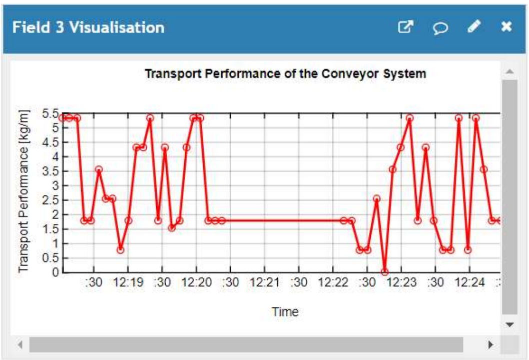


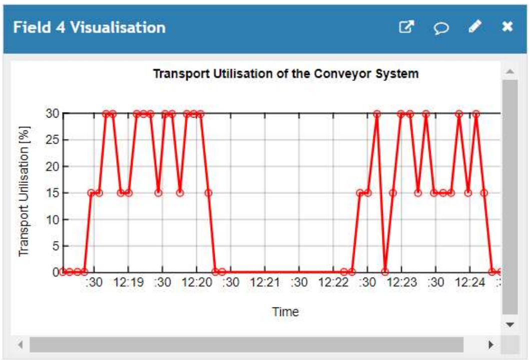
Updated:

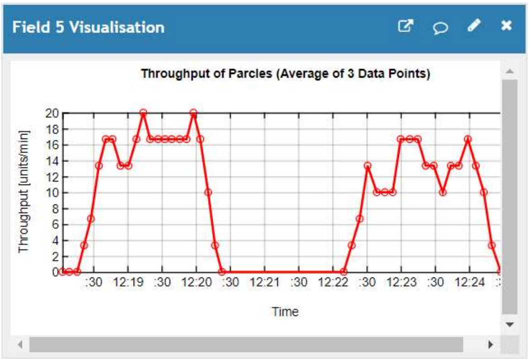
ThingSpeak Channel 1

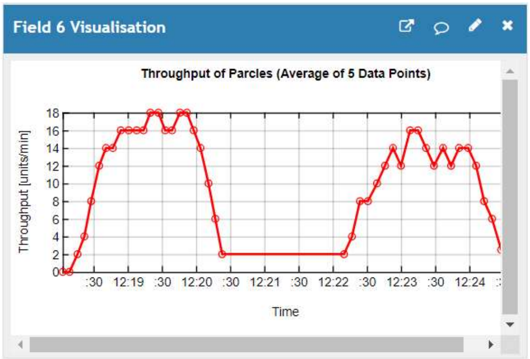


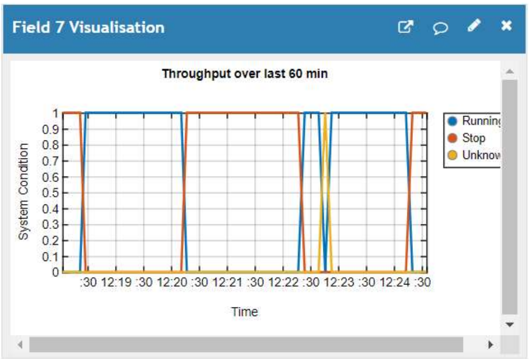


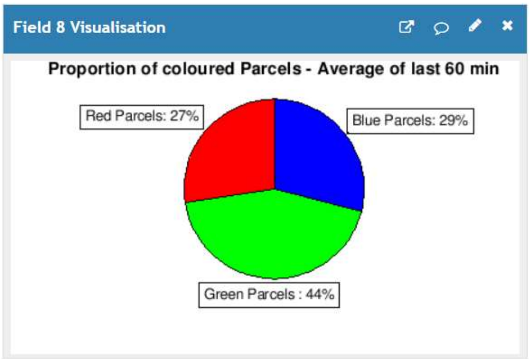




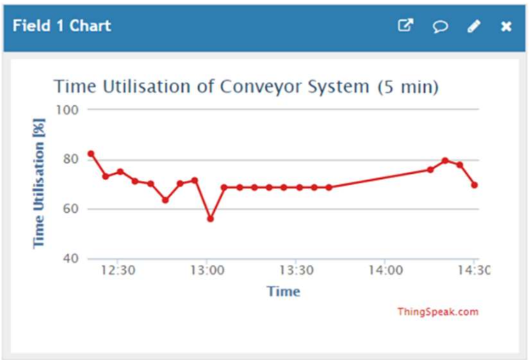


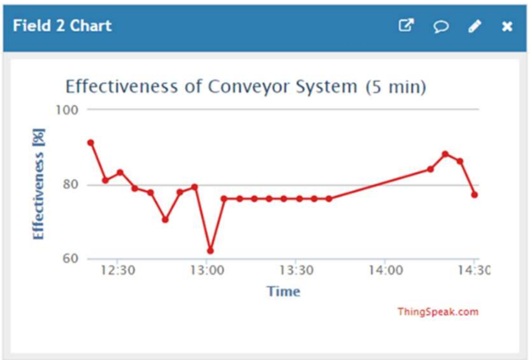


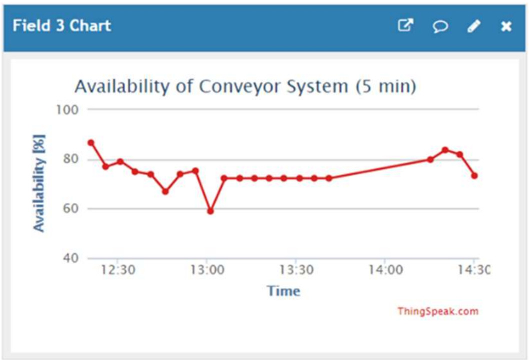


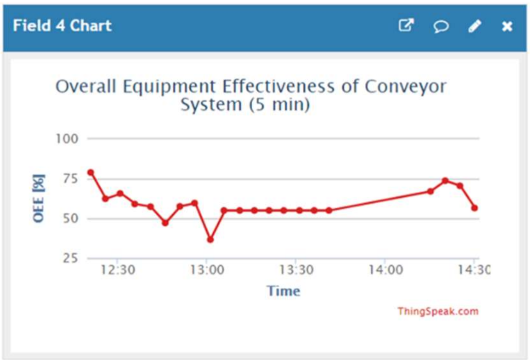


ThingSpeak Channel 2









Appendix F MATLAB Code of a ConvNet to Recognise Handwritten Digits

```
%% Specify Training and Validation Sets

digitDatasetPath =
fullfile('C:','Users','olive','Desktop','Oliver',...
    'TU-Graz','Master','Masterarbeit','Matlab',...
    'ConvNet_Digits','Training_Images');
    % Set the Filepath of the Training Images
imds =
imageDatastore(digitDatasetPath,'IncludeSubfolders',true,...
    'LabelSource','foldernames'); % Set the Filepath of the
Training Images

numTrainFiles = 750; % Set the Number of Training Images to 750
[imdsTrain,imdsValidation] =
splitEachLabel(imds,numTrainFiles,'randomize');
% Randomly split Images into Training Set (750) and Validation
Set (250)

%% Define ConvNet Structure

filterSize = 5; % Set the Size of the Filter to 5x5
numFilters1 = 20; % Set the Number of Filters of the first
ConvNet to 20
numFilters2 = 2; % Set the Number of Filters of the second
ConvNet to 2
strideConv = 1; % Set the Stride in the Convolutional Layers to
1
stridePool = 2; % Set the Stride in the Pooling Layers to 1
poolSize = 2; % Set the Size of the Region for Pooling to 2x2
numClasses = 10; % Set the Number of Output Classes to 10

layers = [ % Define the Structure of the Network
    imageInputLayer([28 28 1]) % Add an Input Layer 28x28x1


convolution2dLayer(filterSize,numFilters1,'Stride',strideConv)
    % Add the first Convolutional Layer
    batchNormalizationLayer % Add a Batch Normalization Layer
    reluLayer % Add a ReLu Layer

    maxPooling2dLayer(poolSize,'Stride',stridePool)
    % Add the first Max-Pooling Layer


convolution2dLayer(filterSize,numFilters2,'Stride',strideConv)
    % Add the second Convolutional Layer
    batchNormalizationLayer
    reluLayer
    maxPooling2dLayer(poolSize,'Stride',stridePool)
    % Add the second Max-Pooling Layer

    fullyConnectedLayer(1000)
    batchNormalizationLayer
    reluLayer

    fullyConnectedLayer(1000) % Add a Fully-Connected Layer with
1000 Neurons
    batchNormalizationLayer % Add a Batch Normalization Layer
    reluLayer % Add a ReLu Layer

    fullyConnectedLayer(numClasses) % Add a Fully-connected
Layer
    softmaxLayer % Add a Softmax Layer
    classificationLayer]; % Add a Classification Layer

%% Specify Training Options

options = trainingOptions(...
    'sgdm', ... % Training Algorithm: Stochastic Gradient
Descent with Momentum
    'InitialLearnRate',0.01, ... % Set the Initial Learning Rate
to 0.01
    'MaxEpochs',4, ... % Set the Maximum Number of Epochs to 4
    'Shuffle','every-epoch', ... % Shuffle the Training and
Validation Data
    'ValidationData',imdsValidation, ... % Define Validation
Data
    'ValidationFrequency',30, ... % Set the Validation Frequency
to 30
    'Verbose',false, ... % Disable Displaying Training Progress
Information
    'Plots','training-progress'); % Plot Training Progress
During Training

%% Train Network Using Training Data

net = trainNetwork(imdsTrain,layers,options); % Create a Trained
Network
```