FFHS
Fernfachhochschule Schweiz
Zürich | Basel | Bern | Brig
Mitglied der SUPSI

Edge Processing Unit

Master-Thesis

im Studiengang MAS Smart Engineering and Process Management (Industrie 4.0)

von

Silvan Zahno

Eingereicht beim Referenten Sharam Dadashnia

am 25.06.2021

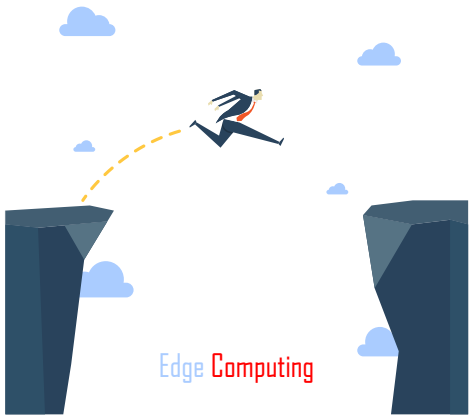
FFHS
Fernfachhochschule Schweiz
Zürich | Basel | Bern | Brig
Mitglied der SUPSI

Masterthesis

Edge Processing Unit

The Way to efficient Edge Computing for Industry 4.0

Master of Advanced Studies in Industry 4.0



Created by: Silvan Zahno

Register Number: 10-964-112

Phone: +41 79 665 41 75

Mail: silvan.zahno@students.ffhs.ch

Professor: Sharam Dadashnia

Submission Date: 25.06.2021

Version: v1.4

Acknowledgements

This master thesis was written during 5 months, from the end of February to the end of June 2021 at the Fernfachhochschule (FFHS) as part of the Industry 4.0 study program.

It is time to thank everyone who has supported me during this three year balancing act between studies, professional and private life. My sincere thanks goes to:

- my wife Katja who always encouraged me and helped overcome all ups and downs, you are truly my rock in the waves
- my whole family, who lent me the freedom to pursuit this endeavor
- my employer, who has been extremely generous in accommodating my needs and allowed me to study besides my professional life
- all my work colleagues who read and reread the thesis and gave me valuable input from a different perspective
- to Syrto who allowed me to use one of their industrial remote controlled machines as a use case for this study

A special thanks goes to Sharam Dadashnia who was not just a wonderful teacher during the studies in the course “Industrial Analytics & Smart Data” but also supported me with his expertise and supervision. All his valuable practice-oriented support pushed me in the right direction.

Abstract

Edge computing has proven to be very valuable and is often needed in addition to cloud computing. Especially in areas such as image and video processing, where large amount of data is generated that cannot be transferred in their entirety to the cloud for processing, or in areas where security and real-time behavior are essential.

In the just mentioned cases, it is not practical to transfer the data to the cloud for processing and return the result to the device. Edge computers can be very powerful, but also high maintenance and power hungry, depending on the algorithm and hardware. In this work, a Proof of concept (POC) is presented to combine low-cost and low-power Central Processing Unit (CPU) with Field Programmable Gate Array (FPGA) to run an algorithm as optimized as possible. The simplicity of the CPU programming and the modular and parallel nature of the FPGA creates very powerful hardware, that can handle many different problems.

A prototype system was presented in an industrial use case, where an automated industrial machine is allowed to run, only when no human is nearby. We present an Edge Processing Unit (EPU) that monitors a selected region using a camera. The images are preprocessed in CPU and an object detection is performed in real time using a tiny You only look once (YOLO) v3 Machine Learning (ML) implemented in FPGA. A Convolutional Neural Network (CNN) hardware accelerator, performs the person detection. A MQTT broker, which also runs on the edge device, sends the results to the cloud and local Programmable Logic Controller (PLC). The entire system runs on a Xilinx PYNQ-Z1 board with a ZYNQ XC7Z020 System on Chip (SoC) that houses a dual core Arm A9 CPU and programmable Artix-7 logic. It achieves a higher frame rate and consumed less power than a comparable non-hardware accelerated edge device. The amount of data transferred was drastically reduced, compared to a cloud application.

Keywords

EPU, Edge Computing, Industrial Edge Computing, Cloud Computing, ML, Image Analysis, Hardware Accelerator, FPGA, Python for Zynq (PYNQ), MQTT, SoC

Contents

1 Introduction **1**
1.1 Syrto . . . 2
1.2 Problem . . . 3
1.3 Objectives . . . 4
1.4 Project Plan . . . 5
1.5 Research Methodology . . . 6
1.6 Structure of this Report . . . 7

2 Project Methodology **9**
2.1 Research Design . . . 9
2.1.1 Scrum Roles . . . 9
2.1.2 Scrum Process . . . 10
2.2 Data Collection . . . 11
2.3 Data Analysis . . . 12

3 Conceptual Foundation & State of the Art **13**
3.1 Definitions . . . 13
3.1.1 EPU . . . 13
3.1.2 Edge Computing . . . 13
3.1.3 Hardware Accelerator . . . 13
3.1.4 FPGA . . . 14
3.2 Method . . . 14
3.3 Problem Formulation . . . 14
3.4 Literature Sources . . . 15
3.5 Literature Evaluation . . . 16
3.6 Search Strategy . . . 16
3.7 History of Edge Computing . . . 17
3.8 Edge Computing using Hardware Accelerators . . . 18
3.9 Hardware Accelerators for FPGA . . . 19
3.10 Frameworks for Developing Hardware Accelerators . . . 19
3.11 Machine Learning for Image Processing . . . 20

4 Use Case **22**
4.1 Presentation . . . 22
4.2 Selection . . . 23
4.3 Test Data Collection . . . 24

5 Technologies **26**
5.1 Specification . . . 26
5.1.1 M2M Communication . . . 26
5.1.2 Framework . . . 26
5.1.3 Hardware . . . 27
5.2 Selection . . . 27
5.2.1 M2M Communication . . . 27
5.2.2 Framework & Hardware . . . 28

6 Implementation **34**
6.1 Edge Processing Unit . . . 34

6.2 Program Sequence . . . 35
6.3 Program Structure . . . 36
6.4 ML Implementation . . . 38
6.5 Client . . . 39
6.6 Communication . . . 41

7 Results **43**
7.1 Detection Rate . . . 43
7.1.1 Person Size . . . 44
7.1.2 False Detection . . . 44
7.1.3 No Detection . . . 45
7.2 Energy Consumption . . . 45
7.3 Transfer Volume . . . 46
7.4 Processing Speed . . . 47
7.5 SWOT Analysis . . . 50

8 Conclusion **51**
8.1 Comparison with the Initial Objectives . . . 51
8.2 Encountered Difficulties . . . 52
8.3 Future Perspectives . . . 52
8.4 Personal Conclusion . . . 53

References **55**

Acronyms **60**

Glossary **64**

A Project Planning **65**

B Literature Research **66**

C Use case evaluation **66**

D Program configuration **66**

E Program structure **68**

Selbständigkeitserklärung

1 Introduction

Cloud computing such as Amazon Web Service (AWS) was launched in 2006, therefore Cloud Computing is in its second decade (Panetta, 2021, n.p.). The way how computing is distributed between cloud and on-premise processing has evolved dramatically in the last years. The field of Edge Computing allows to enhance system performance and reactivity, while maintaining a connection to the cloud, allowing to benefit of its advantages, such as scalability, flexibility and processing power. Gartner did identify Edge Artificial Intelligence (AI) as an emerging technologies already in 2019, which will reach the plateau between 2-5 years (Goasduff, 2019, n.p.).

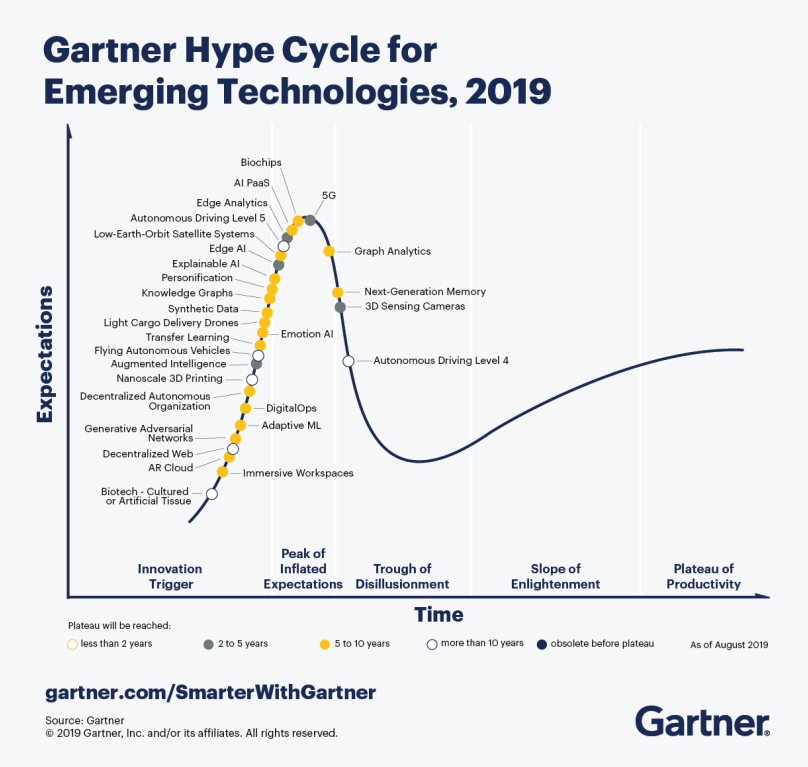


Figure 1: Gartner Hype Cycle

Especially for security applications as well as data intensive processing, Edge Computing is better suited than the more common Cloud Computing, with the drawback of providing and maintaining processing hardware on-premise.

Google and Microsoft cloud service providers have started to introduce FPGA hardware accelerators in their data-centers to reduce energy consumptions, while at the same time being more flexible to advances in technology, algorithm and machine learning (Fallahlalehzari, 2020, p.20). This hardware and software combination allowed them to create many services such as Microsoft

Azure and Bing web search. This is one of the reasons why chip giant Intel has bought Altera, the second biggest FPGA manufacturer in the world back in 2015, spending $ 16.6 Billion (Baker, 2015, n.p.).

In this project, the same principles are applied to the cloud by the before mentioned companies, is used in an edge device. Therefore reducing hardware cost, energy consumption and data transfer while being more reactive and real-time. Even in the modern world, a constant connection to a cloud system cannot be guaranteed. Automated monitoring applications do often rely on cameras and image processing (Chen, Surette, & Shah, 2020, p.13). To guarantee reliability of the system, on-premise processing with the help of edge computers while maintaining a could connectivity is a necessity.

An industrial use case is selected with the implementation partner Syrto and an image processing application with a Convolutional Neural Network (CNN) Machine Learning (ML) model is developed using both CPU and FPGA chips. The application is intended as an additional security layer to protect people from automated industrial machines.

1.1 Syrto

Syrto AG is a three year old company located in Steg (VS) developing teleoperation equipment for industrial machines. With their equipment, Syrto connects the old world with the new digital one and thus creates opportunities for its customers to carry out work more cost-effectively and efficiently. It changes the way of working of tomorrow and sets with its technologies and concepts the conditions for the future growth of its customers and the satisfaction of their employees. It analyzes the processes, workflows and procedures and offers various solutions in teleoperation and telerobotics, image evaluation and automation for cost reduction.

They have currently multiple customers using their remote control system. Data transfer and edge processing is an essential part of their business. Often the user is located in a different part of the country and can remote control the industrial equipment with the help of video, Light imaging, detection, and ranging (LiDaR) and other types of sensor data. Transmission delays as well as processing either on the edge or at the machine locations are essential to optimize the ergonomics of the system. In addition security plays an important role for customer satisfaction

and trust in remote controlled machinery.

1.2 Problem

For many applications, cloud and cloud-based systems present many opportunities, but also risks and issues (Brender & Markov, 2013, p.1-7). The entirety of the data must be transferred from one location to the cloud for processing and the results must then generate impacts back on-site. Despite the advent of high-speed connectivity technologies such as fiber or 5G, the response time from a cloud center is variable and can take anywhere from miliseconds to seconds, depending on where it is located and what path the packet has taken along the way and how busy the cloud center is (PGDAV College & Singh, 2014, p.1,6-7).

Syrto AG is just one example of a company that encounters exactly these problems with its system. For Syrto AG, reducing data transfer can lead to better real-time teleoperation of machines and a better reactivity for its users. In most applications, the machines are remote controlled from a person located in a different part of the country. Especially images and video stream transmissions between the user and the machine has to be reduced as much as possible. Only the necessary data should be transmitted.

A teleoperation of an industrial equipment requires a multitude of systems, working in the background for the user. Especially security systems have a real-time aspect and need to be working independently to a cloud or internet connection. Because the user has just a limited view about the surroundings of the machine he controls and his attention is not solely dedicated to the operation of one machine. Additional systems are needed to guarantee the safety of collaborators working at the location of the industrial equipment.

For automated industrial equipment, different regulation on an international, european and swiss basis exists which have to be respected. Syrto AG operates currently only on the swiss market. The 2006 released, two part security norm DIN EN ISO 10218 defines the security measures to guarantee the security for a collaborative industrial equipment (Maier, 2019, p.180-183). The regulation ISO 13855 and 13857 defines the security distances in general, according to the reactivity of the machines to stop its operations (Hägele, Nilsson, Pires, & Bischoff, 2016, p.1385-1422) (Kerezovi, Sziebig, Solvang, & Latinovic, 2013, p.3-4). The formula to calculate the security dis-

tance is as follows:

$$S = K * (t_1 + t_2) + C$$

- S = Security distance
- K = Expected approach speed of the human body or parts of the human body
- t_1 = Response time of the protective device itself
- t_2 = Reaction time of the machine, i.e. the stopping behavior of the machine as reaction to the signal of the protective device
- C = Possible approach to a danger zone undetected by the protective device

In this calculation the t_1 time is the only variable Syrto can modify and change since the reaction time of the machine itself, t_2 is given by the manufacturer of the equipment. Therefore security detection mechanism have to be on-site and stop operation of the machines, as soon as a problem occurs. A variable, unpredictable transmission delay from and to the cloud is not acceptable.

The field of detection algorithms technology is evolving and new algorithms and techniques are developed and surfacing constantly (Biggio & Roli, 2018, p.1) (Gu et al., 2018, p.370). For an industrial company, systems are constantly improving and the latest technology is used. To change the hardware with every iteration step, to adapt to the needs of the new technology is not cost effective. Hardware should be reused in the future as much as possible.

1.3 Objectives

The primary goal of this research paper, is to show a novel way to reduce or eliminate problems currently inherent with Edge Computing. Using lower cost hardware while maintaining flexibility for future changes and improvements. The research question for this paper is as follows: “What edge computing technology can industrial companies use for flexible on-premise processing“. It is split into three subquestions:

- What impact do hardware accelerated edge devices have on energy consumption?
- What impact do hardware accelerated edge devices have on the amount of data transferred?
- What kind of impact do hardware accelerated edge devices have on the computing speed?

In addition there are multiple Specific, Measurable, Achievable, Relevant, Time-bound (SMART) objectives.

Non-technical objectives

- Overview about the edge computing and its use cases.
- Overview about hardware accelerator and its use cases.
- Comparison of computing power, cost as well as power efficiency with and without hardware accelerator.
- Comparison and implications of data transfer between the edge to a cloud system with and without hardware accelerator.

Technical objectives

- One practical POC of edge computing.
- Using a combination of traditional hardware as well as a custom hardware accelerator implemented in a FPGA.
- The processing hardware should be cheap (<300 CHF).
- 2x energy reduction compared to a non hardware accelerated on-side processing platform.
- 4x data-rate reduction compared to a cloud processing platform.

The thesis showcases one example of image analysis, it can not be used as an implementation guide. The end product is a proof of concept and is not intended to be used in an industrial environment. The complete project plan including the work packages can be found in section 1.4.

1.4 Project Plan

Figure 2 below shows the work package planning of the project.

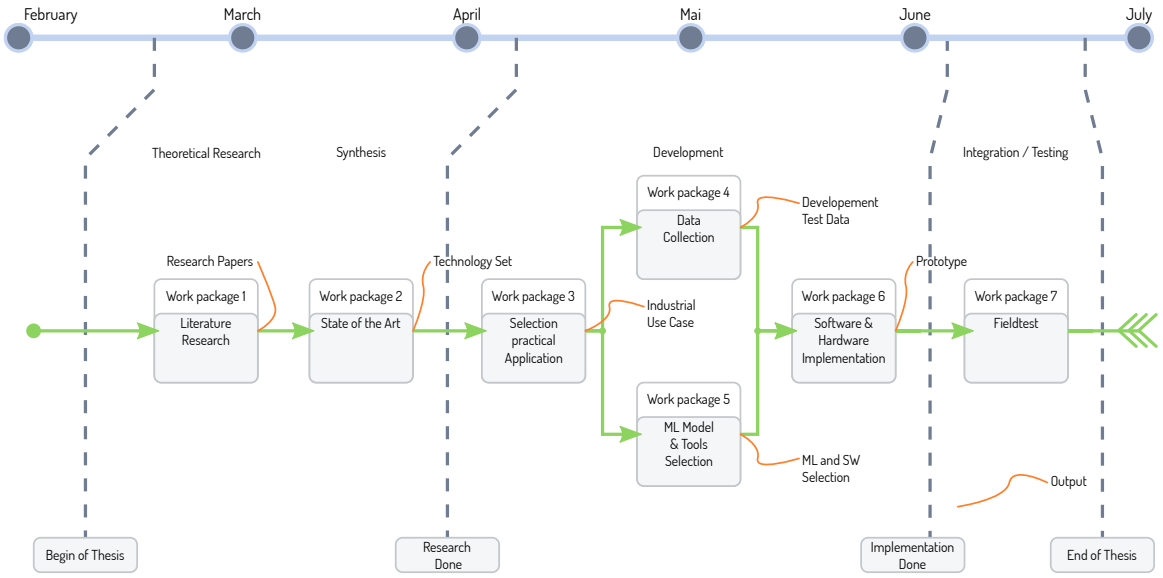


Figure 2: Project Plan

Around two months are dedicated to theoretical research and synthesis of existing research. During work package 1 the methodical literature research is made. The work package 2 is dedicated writing the state of the art from the result of the literature research done previously, see chapter 3 Conceptual Foundation & State of the Art.

Roughly the next two months are dedicated to the development of the edge device. In work package 3 the practical use case at an existing Syrto AG equipment is selected. The two work packages 4 & 5 will be worked at in parallel, the tools and frameworks for implementation as well as data from the use case will be gathered. Work package 6 is dedicated to the development of the hybrid edge device targeted at the previously selected use case, see chapter 5 Technologies as well as 4 Use Case and 6 Implementation.

The last part of the thesis is the work package 7 integration and testing on-site. The developed device was used in combination with a machine, see the results in chapter 7 Results.

1.5 Research Methodology

This research project includes a research of a the current state of the art in the field of edge computing in general, as well as image analysis in particular with the help of hardware accelerators.

The state of the art literature review is made according to Fettke (2006, p.9). An example use case is selected at Syrto AG, so that an POC can be developed, deployed and then evaluated with experiments. The research is general applicable and not only valid to the specific use case. The project methodology is described in detail in the chapter 2.

During the project an agile work methodology called Scrum is used. This is further explained in 2.1 Research Design. The prototype development will pass through 7 different stages: Conception, initiation, analysis, design, construction, testing, and deployment. The described work packages are split into smaller tasks labeled: Theoretical research, writing, development and implementation. These cards are placed in a virtual Kanban board, which allows for an agile development cycle. The created Kanban cards can be seen in the appendix 24 Detailed Project Planning. During to the project phases outlined in the project plan, the tasks are placed within the Kanban sections, according to theirs current status: backlog, to do, on progress, verify and done.

1.6 Structure of this Report

Chapter 2 Project Methodology outlines the conceptual foundation as well as the theoretical background of this thesis.

Chapter 3 Conceptual Foundation & State of the Art contains a literature research about the current development in the field of Edge Computing. Only scientific databases were used to gather an overview about existing research projects, use cases and developments.

In chapter 4 Use Case different existing industrial machines from Syrto AG are evaluated in order to select a use case for the implementation of the proposed edge processing device.

In chapter 5 Technologies the different technology elements which are needed are explained and evaluated. The selected hardware and its functionalities in coordination to the use case is shown.

In chapter 6 Implementation the developed system is split up in its subblocs and the functions. The evolution and implementation of each sub bloc is explained in detail.

In chapter 7 Results at the end of the project a field test at the selected location was made. The result were evaluated according to the before mentioned SMART objectives 1.3.

The last chapter 8 Conclusion the current state as well as problems and next steps are outlined.

2 Project Methodology

The following chapter aims to describe the empirical approach of this project work. The goal of the project is to develop an EPU in the form of a prototype or POC in order to evaluate the practical applicability of this technology. The chapter Research Design contains the structure and used project methologies, whereas the chapter Data Collection and Data Analysis shows how and in which form the experiment is conducted and evaluated.

2.1 Research Design

This thesis is carried out as a project work, in this section it is explained why this is the appropriate instrument and which project method was chosen. It is explained in more detail the mapping between research method and practical application.

The research question relates to the practical application of an EPU in an industrial environment. The aim of the work is to develop a prototype or a POC and to test the system in a specific selected use case. The method of a project work is most suitable because of the practical character as well as developement work during the thesis. The development phase of such projects is time-consuming and is carried out strongly in discussion and collaboration with the different members of the industrial partner (Kraus & Westermann, 2010, p.27).

As a project method, there are a confusing number of possibilities, where many have been proven in countless projects Michaelson (2010, p63). Michaelson wrote in 1974, p.21 that writing and developing with an incremental method can support and aid the project and the development work mutually. Such an iterative process is also called continuous development.

One of the best known and most used project methods with an continuous development character is Scrum. With Scrum there is not a detailed project plan developed at the beginning of the project, but the work is done in small day iteration with a daily communication (Wouters, 2009, p.30).

2.1.1 Scrum Roles

A Scrum project method requires multiple actors, where one person can take on multiple roles at the same time. The responsibilities of the individual roles do not overlap, otherwise conflicts

of interest can arise (Gloger & Schwaber, 2013, p.108). There are three main roles in the organization of Scrum. A product owner, who describes complex problems in a product backlog and is responsible for the Return of Investement (ROI), a scrum master whose main task is to define and clarify the various roles and ensure compliance with the Scrum workflow; and a developer team that divides the individual tasks into smaller tasks and processes them in order to create the product. In addition, there are two other roles outside of the project team. A customer who has commissioned the product and a user who uses the application from the Scrum team (Gloger & Schwaber, 2013, p.109).

For this project the roles are defined as shown in figure 3. The manager roles is in this case not occupied because only one project exists.

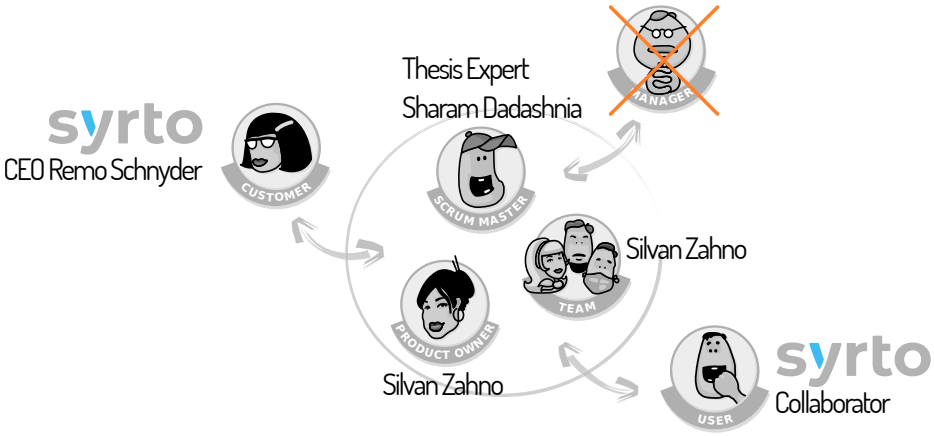


Figure 3: Scrum role according Gloger and Schwaber (2013, p.109) with corresponding mapping

2.1.2 Scrum Process

The scrum process consists of several elements, an overview of the flow used for this project is shown in figure 4. The project is divided into so-called sprints which in this case lasts 4 weeks, this is the heart of Scrum (Ken & Jeff, 2020, p.7). Before each sprint, a sprint planning is held with the development team and the scrum master. During sprint planning three steps are performed (Ken & Jeff, 2020, p.7-10): during the sprint retrospective the work of the previous sprint is reviewed and if necessary moved on the Kanban board from verify to done, next a goal for the next sprint is determined, in the last step the sprint planning tasks are moved from the backlog to the sprint.

One major change from the original scrum workflow was adapted for this project. According to Ken and Jeff (2020, p.9) short daily scrum meetings are held to inspect the progess toward the

Sprint Goal. These daily scrum meetings are replaced by weekly meetings because of the part-time nature of the thesis.

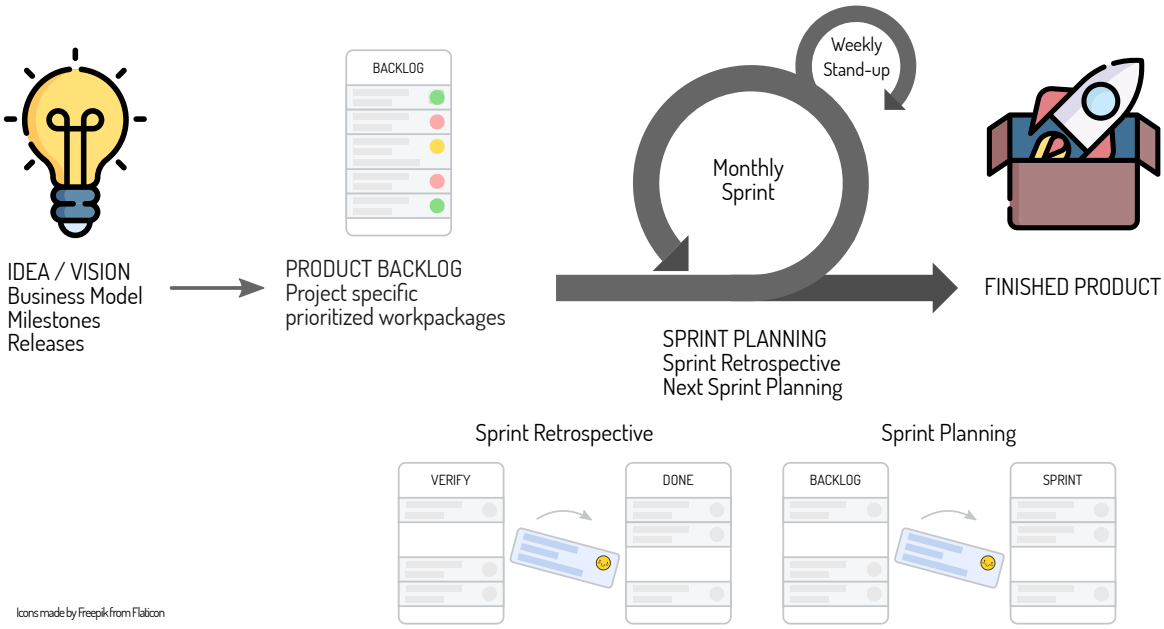


Figure 4: Scrum flow

The Scrum board or Kanban board allows to track or visualize the individual tasks (Gloger & Schwaber, 2013, p.59). For this project a Kanban board with the columns: backlog, to do, in progress, verify and done are used. The Kanban board can be found in the appendix 24. Each individual task from the initial project plan (figure 24) are tagged and moved within these columns.

2.2 Data Collection

Since the goal is to obtain information about EPUs in a practical industrial environment, the data collection is carried out by means of experiments. In experiments, a system or process is stimulated with inputs and the observable outputs are statistically analyzed (Montgomery, 2017, p.3). In detail, tests or series of runs are made and the controllable input variables are meaningfully altered to observe the system and identify the reasons that allow to explain the observed output response (Montgomery, 2017, p.3-4).

In this case, comparative experiments are used which aim to compare two or more systems. Four experiments were defined to compare power consumption, processing speed, external transfer

rate and detection rate. These experiments are inline with the objectives defined in 1.3 Objectives.

For each experiment the cause-and-effect diagram identifies the controllable and uncontrollable factors according to Montgomery (2017, p16).

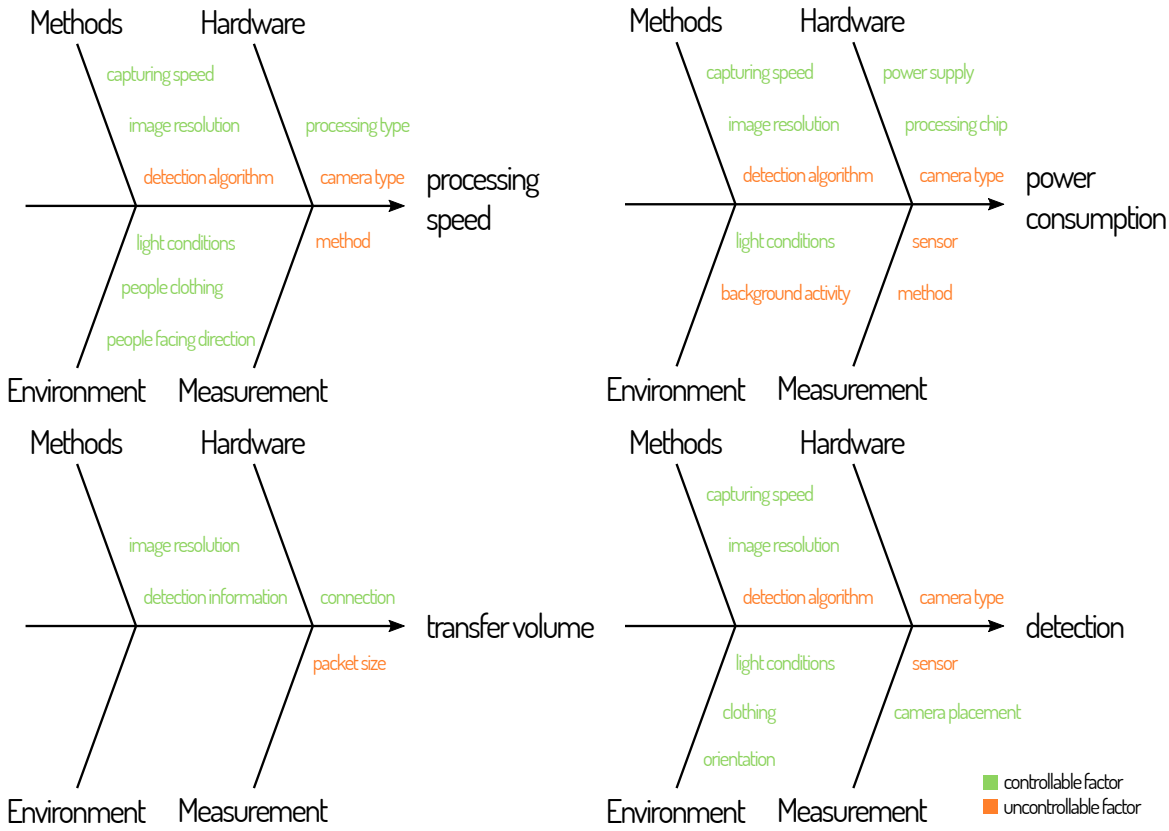


Figure 5: Experiments cause-and-effect diagram

2.3 Data Analysis

For each experiment a statistical analysis is made with the help of box, bar and pie plot diagrams. All experiment runs are saved for the creation of the afore mentioned graphs. The box plots allow to present the different experiment runs with their median, inter-quartile range as well as outliers. In addition to the plots, all informations are then synthesized into a Strength Weaknesses Opportunities and Threats (SWOT) analysis about the EPU. According to Leigh (2009, n.p.) a SWOT analysis has also applications for benchmarking and industry analysis such as the EPU from this project. It allows to assess the usability and points were it has to be improved in the future.

3 Conceptual Foundation & State of the Art

3.1 Definitions

3.1.1 EPU

The term Edge Processing Unit (EPU) has once been used in a different context, to describe a processing unit detecting edges in video frames. This was in a patent application back in 1980 (Robinson & Reis, 1980, p.1). For this project an EPU is defined as an edge device, which is used on-premise and uses SoC combining FPGA and CPU to provide computing resources for different kind of algorithms. It can be used in combination with picture- and videocameras and allows to use performant machine learning algorithms such as deep neural networks like YOLO for data processing. An EPU is constantly connected to a cloud system with the help of Machine to Machine Communication (M2M) transmission protocols.

3.1.2 Edge Computing

Edge computing is a technology that includes a variety of standards and frameworks. The Open Glossary of Edge Computing of the Linux Foundation defines edge computing more concretely as ”the delivery of computing capabilities to the logical extremes of a network in order to improve the performance, operating cost and reliability of applications and services”. In practice, the glossary goes on to say, this means distributing new resources and software stacks between today’s centralized data centers and the ever-growing number of devices in the field.

3.1.3 Hardware Accelerator

Traditional computers as well as data centers and smartphones use hardware accelerators to reduce the load on the CPU. For computational intensive tasks and special calculations such as neural networks and graphics rendering, hardware accelerators help to efficiently and quickly perform the required calculations. Graphics cards are an example of hardware accelerators that are used in virtually every computer. Although hardware accelerators can efficiently and quickly perform the computations they are designed for, but they have the disadvantage that they can only perform these types of computations. They are not as ambivalent as processors.

3.1.4 FPGA

FPGA's are semiconductor chips which can be reconfigured. This functionality allows the implementation of various customized functionalities with a single chip. FPGA's are often used in satellites, medical devices and weapons systems because of their robustness and real-time capability. They are also particularly suitable for implementing hardware accelerators that provide custom functions such as a neuronal network. Because FPGA's programming is configuring hardware instead of writing software, they are less suitable for implementing functions such as floating point calculations.

3.2 Method

The state of the art literature review is made according to Fettke (2006, p.9). The performance of a review can itself be understood as a scientific problem, in the handling of which appropriate methodological guidelines must be taken into account. Attempts are made to filter similar results, find inconsistencies, contradictions and generalization of previous findings (Fettke, 2006, p.1-2).

For a methodic research synthesis, five phases are to be evaluated (Fettke, 2006, p.4):

1. Problem formulation: During this phase, the question to be answered by a review is formulated, delimited and specified in more detail.
2. Literature search: During this phase, the literature suitable for the research question has to be researched.
3. Literature evaluation: This phase aims at reviewing the identified literature with regard to its relevance as well as processing and systematizing it appropriately.
4. Analysis and interpretation: Within this phase the results of the previous phase are to be examined and evaluated against the background of the raised problem formulation.
5. Presentation: Finally, the results of the investigation are to be prepared and presented to the public.

3.3 Problem Formulation

The general research question is: "What edge computing technology can industrial companies use for flexible on-premise processing".

In order to specify and restrict the search results the keyword radar lists all important keywords for the given research question (figure 25). It is split into four segments with four layers of importance. The four segments are:

- Technology: Keywords linked to the technology used in EPU
- Software: Software programming languages and techniques used within and EPU
- Hardware: Specification and types of hardware which can elevate and edge device to an EPU
- Business: Keywords linked to business cases which can be performed by an EPU

For each segment, there exists four layers of importance. The closer to the center, the more important the keyword is for the selected use case (project related), the further away, the less it is linked. The four layers are: project, technology, application and topic.

3.4 Literature Sources

The sources for the literature review are from IEEE Xplore, Science Direct, Springer Link, JSTOR and Google Scholar. In order to approach the topic from a high level and to gain a better understanding of the various branches in the area of edge computing, machine learning and hardware accelerators, a potential analysis for the individual techniques is made. Each subject is looked at in the context of an industrial application.

3.5 Literature Evaluation

The data will be categorized into three groups:

Primary resources - high level of detail

- Reports
- Thesis
- Company reports
- Conference proceedings

Secondary resources - medium level of detail

- Journals
- Books
- Newspapers
- Publications

Tertiary resources - low level of detail

- Indexes
- Databases
- Encyclopedias
- Bibliographies

3.6 Search Strategy

For each literature source a search strategy is defined in table 1 Search Strategy. The number of hits and reviewed papers is listed.

Source	*Search Strategy*	*Hits*	*Reviewed*
IEEE Xplore	((("All Metadata":"edge computing") OR ("All Metadata":"edge processing") OR ("All Metadata":"edge device")) AND (("All Metadata":"hardware accelerator") OR ("All Metadata":"fpga")) AND ("All Metadata":"image processing")) OR ("All Metadata":"pynq")	107	22
Science Direct	(("edge computing" OR "edge processing" OR "edge device") AND ("hardware accelerators" OR "fpga") AND "image processing") OR "pynq"	86	8
Springer Link	(("edge computing" OR "edge processing" OR "edge device") AND ("hardware accelerators" OR "fpga") AND "image processing") OR "pynq"	143	11
Google Scholar	(("edge computing" OR "edge processing" OR "edge device") ("hardware accelerators" OR "fpga") "image processing") OR "pynq"	1310	17
JSTOR	((("edge computing") OR ("edge processing") OR ("edge device")) AND (("hardware accelerators") OR ("fpga")) AND ("image processing")) OR ("pynq")	12	0

Table 1: Search Strategy

3.7 History of Edge Computing

”The roots of edge computing reach back to the late 1990s, when Akamai introduced Content Delivery Network (CDN) to accelerate web performance” (Satyanarayanan, 2017, p.2). Since then Satyanarayanan (2017, p.2) lists many efforts done to standardize edge computing. The European Telecommunications Standards Institue (ETSI) started these effort back in 2014, Open Edge Computing (OEC) was launched by Vodafone, Intel and Huawai in 2015 and the Open Fog Consortium was created by Cisco, Microsoft, Intel, Dell and ARM also in 2015. Currently there are working groups in various consortia, such as the Industrial Internet Consortium (IIC), the Industry 4.0 platform or the European Edge Computing Consortium (EECC), which was recently being founded, is dealing with implementation and standardization issues. According to Willner and Gowtham (2020, p.3) data processing does not necessarily have to take place at the topologically outermost edge. On the contrary, there is a broad spectrum of possible positioning in the network. The entire spectrum of possible distributions is reflected by the term fog computing , i.e., distributed, device-based data processing (Pop et al., 2021, p.1). However, the term itself is not uniformly defined in the literature and is used differently depending on the application domain.

The positioning of edge processes alone shows what a broad field edge computing covers. In discussions, articles and scientific papers, it is often not defined what type of edge is actually involved. Depending on the classification, different requirements are placed on software and hardware. The Reference Architecture Model Edge Computing 4.0 (RAMEC 4.0) is intended to help with the classification. Like comparable models such as the Smart Grid Architecture Model (SGAM) for the energy domain or the Reference Architecture Model Industrie 4.0 (RAMI 4.0) from manufacturing, these models are not technical architectures, but rather an orientation and classification aid (Willner & Gowtham, 2020, p.2-5).

Edge computing has been used more and more in recent years, also because of the continuous trend and popularity of AI and its broader adoption in the consumer and industrial sector as well as benefits in many use-cases, over traditional Cloud Computing (Caprolu, Di Pietro, Lombardi, & Raponi, 2019, p.1-2). Caprolu et al. (2019, p.1-7) states that Fog and Edge computing allows to avoid the problem to use Cloud Computing as a one-size-fits-all solution where ressource centralization is a main issue. Distributed computing where Cloud, Fog and Edge computing are used

simultaneously and addresses the problem of stated above and improves network latency and jitter, especially for time-critical applications. Possible application areas are gaming, Augmented Reality (AR), e-health and other context aware use-cases. Technologies used on Edge Computing are wide-spread, from traditional Real Time Operating Systems (RTOS) for applications needing fixed response times to containarization with systems like Docker for small lightweight independend micro-services, and FPGA technology for custom proprietary but reconfigurable hardware implementations. Caprolu et al. (2019, p.6) also identified the possible attacks for virtualization technologies for given scenarios. Depending on the application this can lead to a lower or higher security impact.

3.8 Edge Computing using Hardware Accelerators

Edge computing started with traditional hardware like CPU's, but because of environmental and local dependent factors, there are different specific requirements for edge computing. These requirements can be of various nature, performance constraints, size as well as space requirements. Depending on the work or calculation of the edge device, there are also different performance specifications. To combine all this, there are different methods and technologies available (Colbert, Daly, Kreutz-Delgado, & Das, 2021, p.1). Specifically for machine learning applications, Colbert et al. (2021, p.4) have tested different hardware accelerators against each other. In particular, a FPGA implementation of a Deconvolutional Neural Network (DCNN) was compared to a power hungry batch processing Nividia Jetson TX1 Graphical Processing Unit (GPU). It was shown that the FPGA is superior to the GPU implementation in the performance-per-watt category by 31%.

The fact that energy consumption is one of the main criteria of edge computing was shown in a survey by Jiang et al. (2020, p.2). There are various approaches to minimize energy consumption. An energy-aware design must be used not only on the hardware level, but also on the computing architecture, operating system and middleware (Jiang et al., 2020, p.2-18). Only with a combination of the different components can an energy efficient edge devices be implemented. Specifically for embedded systems, Song (2021, p.26-49) shows the different hardware acceleration methods. "Embedded hardware for neural network application acceleration ranges from single-board devices such as the Google Edge Tensor Processing Unit (TPU) to high-performance

processors such as Intel Xeon and AMD EPYC CPUs, NVIDIA GPUs with Tensor Cores. All of these hardware options for neural network acceleration provide different features and computational capabilities in the embedded systems." (Song, 2021, p.23).

3.9 Hardware Accelerators for FPGA

As shown in the chapter 3.8 by Jiang et al. (2020, p.1 et seq.) and Colbert et al. (2021, p.1 et seq.), FPGA's offer a great advantage in the area of energy consumption and in their flexibility. There is a wide variety of implementations for different applications.

Neural networks in particular CNN are by their architecture especially suited to be implemented as hardware in a FPGA (Danopoulos, Kachris, & Soudris, 2021, p.1 et seq.). Struharik, Vukobratović, Erdeljan, and Rakanović (2020, p.1 et seq.) proposed a hardware accelerator for compressed CNN called CoNNA, which allows faster layer processing using dynamic reconfiguration during CNN layer processing. Also Xia et al. (2021, p.1 et seq.) presented a lightweight neural network called SparkNet which drastically reduces the weight parameters and thus the computational requirements. Furthermore, Panchbhaiyye and Ogunfunmi (2021, p.1 et seq.) developed on a PYNQ Z1 board a modified version of a CNN which, with the help of the available First in first out (FIFO) memory, optimally supplies input and output layers with data and was thus able to achieve a 13% improvement on the previous architecture.

Also outside of ML models FPGA's are used to speed up calculations. As an example, Lee and Jeon (2020, p.1 et seq.) has presented an acceleration of image processing based on PYNQ and HLS. This CV application proposes a parallelization of image processing, using an image processing core, that allows to perform the five functions: color conversion, spatial low-pass filter, sobel operation and sum of gradients magnitude on a Zynq SoC platform. Compared to a pure Python OpenCV implementation, an acceleration of 15.58x was achieved.

3.10 Frameworks for Developing Hardware Accelerators

In order to develop and implement a hardware accelerator, the available tools and frameworks play a central role. Some frameworks have already been implicitly mentioned in the chapter 3.9, such as Xia et al. (2021, p.1 et seq.) with the Spark Framework. There are a number of other

framework implementations which are specialized for different tasks. Xuan et al. (2016, p.1 et seq.) presented a dynamic reconfigurable hardware/software framework for connecting sensor networks on the edge. Furthermore, Pop et al. (2021, p.1 et seq.) developed a platform that more closely couples the Edge with the Cloud via the FORA Fog Computing Platform. This was developed especially for time critical applications and uses mostly open standards. A Fog reference architecture was defined using the Architecture Analysis Design Language (AADL) description standard.

One framework or platform used in various projects that stood out was the open source PYNQ project launched by chip vendor Xilinx in 2016 (Xilinx, 2020, n.p.). It is specifically made for the Xilinx ZYNQ SoC Family and is intended to narrow the gap between domain experts, embedded software engineers and hardware engineers. The platform is based on Python the most popular programming language for data science and ML according to IEEE Spectrum (2020). Based on this platform, various hardware accelerator frameworks have been implemented. Umuroglu et al. (2017, p.1 et seq.) introduced FINN a fast, scalable binarized neural network inference framework for Binary Neural Network (BNN). FINN allows millions of classifications to be performed per second with sub millisecond response time. It is ideal for embedded hardware and is based on the principle, that neural network weights using binary numbers, allows similarly accurate classification as with floating point numbers. The project was further developed in 2018 by Blott et al., p.1 et seq. and extended to Quantized Neural Network (QNN), where floating point numbers are simplified with quantization. This allowed the ML model VGG-16 to be reduced from 4.4Gbit to 138 Mbit and that of YOLO-v2 from 1.6Gbit to 50Mbit. PYNQ is also well suited to rapid CNN prototyping as demonstrated by Wang, Davis, and Cheung (2018, p.1 et seq.) and Dechelotte, Tessier, Dallet, and Crenne (2018, p.1 et seq.) through similar frameworks with a high-level of abstraction.

3.11 Machine Learning for Image Processing

Accurate and efficient ML models are an integral part of many systems. This is especially true for custering and classification problems which are often needed for image and speech recognition (Danopoulos et al., 2021, p.1 et seq.). CNN or adapted models such as BNN or QNN are increasingly used for these types of problems and have proven to be accurate and fast (Umuroglu et al.,

2017, p.1 et seq.) (Blott et al., 2018, p.1 et seq.) (Xia et al., 2021, p.1 et seq.).

The YOLO ML model presented by Redmon and Farhadi (2018, p.1 et seq.) is already in its third generation and has proven to be an effective and adaptive neural network. In a comparison performed between leading neural networks such as RetinaNet, FPN DSSD, SSD, it has proven to be significantly faster but slightly less accurate with an Average Precision (AP) between 0.5 and 0.95 with the Microsoft Common Objects in Context (COCO) image dataset (Lin et al., 2015, p.1 et seq.). The YOLO pretrained CNN classifies 80 different classes (objects) and comes in several variants. For example, the YOLO-v3 tiny is designed to be smaller to be implemented in a FPGA (Wei, Honda, & Amano, 2020, p.1 et seq.). The FINN ML framework includes a quantized tiny YOLO implementation which can be implemented in a Xilinx ZYNQ (Blott et al., 2018, p.1 et seq.).

4 Use Case

Syrto AG has currently five industial remote machines in use. All five location were visited and three were selected as potential candidates as a use case (figure 6). All three locations have remote controlled or automated machines where collaborators can interfere with the machine.

Figure 6: Use case candidates, from left to right BAUWA AG sludge treatment, BAUWA AG automated crane container, Theler AG concrete factory

4.1 Presentation

The first use case (image on the left 6), is the BAUWA sludge cleaning facility. In this facility sludge is collected from all over Switzerland and cleaned through an industrial process. In the collection hall multiple people and machines have access. There is a small remote controlled two tonnes excavator which is transporting the sludge to different collection badges. The operator is located in Steg (VS) an around 15min drive away from the machinery. In the same hall which purpose is to store sluge, trucks are deposing new sludge and a human operated 30 tonnes excavator can also transport bigger amounts of material around. Once a human on foot or in a vehicle is entering the building, the remote controlled excavator needs to stop any operation. For this use case, vehicles of different types as well as humans need to be detected.

The second use case (image in the middle 6), is located at a different part of the sludge cleaning process. A fully automated crane is picking up material and transports it 10 meters up into a container indent for the next treatment. In the material compartment, people are regularly interfering with the system to perform chemical tests and control measurements from the sludge as well as to reorganize the material banks, which have formed over time. Currently the automated crane has 2 security measures. First an emergency stop button to halt the operation at any given time and secondly a laser fence for detecting people crossing the container barrier. In the past, this system already produced false positive and negative detections, because of the harsh environ-

ment. For this use case, people have to be detected and the machine needs to stop operations immediately.

The third use case (image on the right 6), is located in Naters (VS) at Theler AG, which is a company operating a concrete factory. Sand, rubble chemicals are mixed together to create concrete for surounding constructions. The fabrication process consists of multiple buildings, connected with conveyor belts for material transportation. Some conveyor belts are automated and there is a remote controlled excavator project in progress. The location is loosely secured with physical barriers and fences. Trucks are delivering the sand and rubble, human operated excavator are operating in the area. In the past, there have been several incidents of people getting lost in the area and children used the sand banks for playing. The automated conveyor belt, as well as the planned remote controlled excavator need to stop operations when people are in proximity.

4.2 Selection

The selected use case for this project satisfies multiple criteria. The evaluation critera are:

- Detection - Type of object to be detected. During this project the neural network will not be trained specifically for the application, because of a lack of classified images. The neural network needs to be pretrained with the necessary detection type included.
- Importance - During interview with the CEO of Syrto AG, the importance of the system and necessity is evaluated for each use case individually.
- Environment - The prototype will not feature protection against harsh environments, only a minimal protection is planned. The components are consumer grade instead of industrial grade in order to respect the budget and time constraints.
- Installation - At some locations, additional installations are necessary. The prototype needs at least an electrical installation as well as ethernet or wifi for the connectivity.

The evaluation has identified the use case of the automatic crane to be most suitable (5 Use case evaluation). The detection of people is covered with many pretrained neural networks. The likelyhood of an error with the current security system is the highest, there is also a regularly planned intervention of collaborators in order to verify the fill level state as well as perform contamina-

tion measurements. The environment is indoor with all needed installation equipment present. The prototype can be installed at a reasonable height. An overview of the entire work area is guaranteed. The system will act as an additional safety layer. The cameras currently used for verification and remote controlling have a resolution HD 720p resolution but only of 640pixels x 360pixels are transmitted. These cameras are not capable to identify people and are used on different premisses.

4.3 Test Data Collection

For the development and implementation of the prototype, data respectively images are needed to test and validate the software architecture in particular the machine learning model. An interview with the owner of the automatic crane has identified the essential and most encountered scenarios were the system has to properly identify people in proximity. These scenarios include different elements:

- Different lighting and sun locations: these include morning, noon and evening sun as well as artificial lighting on and off. Since the artifical lighting has an motion detection sensor, the case with no lighting for the camera is ignored.
- Different collaborators: there are a variety of different operators who can get close to the machine. These are machine operators and maintenance workers, but also suppliers of new debris material which must be processed by the plant.
- Different locations and perspective of the people in the field of view, such as front, back and side view of these persons.

```
ffmpeg -y -i $out_videos/$fname.$video_ext -r $framerate -qmin 1 -qmax 1 -q:v 1
            $out_images/$fname/${fname}_%05d.$img_ext
```

Listing 1: ffmpeg image extraction

For all the different scenarios the video recordings were made with the selected camera for the prototype to avoid image quality issues. These recordings summed up to around 30min. For all recordings, still images were extracted with the help of the open source tool ffmpeg (Korbel, 2021, n.p.). These are used to verify the performance of the architecture during developement. The command to extract the images is highlighted in listing 1. The framerate and output format

can be selected, the command will try to extract the images without any quality loss. The video input format in this case was `*.mp4` and the image output format was `*.jpg`. This resulted in a test image dataset of 28 different scenes with a total of 790 images.

5 Technologies

To create a functional prototype, a multitude of technologies needs to be combined together in a consise package. The technologies are selected according to the specification of the selected use case (4.2 Selection).

- Connectivity protocol
- Hardware
  - CPU
  - FPGA
  - Interfaces
- Framework for FPGA development
- Framework for software development

5.1 Specification

The specifications describe the requirements from the system of the various elements of hardware and software. The specifications can be broken down into the three main areas of hardware, framework and M2M communication.

5.1.1 M2M Communication

This specification is less linked with the hardware, because all major communication types are supported by the most common programming languages and platforms. The communication protocol needs to have a low memory and power footprint since it will be used in a low performance CPU and be able to be communicate bidirectional with multiple endpoints. The endpoints are a cloud infrastructure, the Programmable Logic Controller (PLC) who controls the industrial machine and the client program used by the machine operator/supervisor. This means the system has to communicate to the cloud as well as local systems. Internet of Things (IoT) standards should be applied as much as possible.

5.1.2 Framework

The framework is split into two groups: a software framework for the CPU programmation called Processing System and a FPGA framework for the hardware integration called Programmable

Logic (PL). They have to work together and support the selected SoC chip and Operating System (OS) used. The machine learning frameworks need to support detection of people. It has to be as lightweight as possible to be able to run on the low power CPU. Since most machine learning algorithms are developed in Python, support for it is an advantage.

5.1.3 Hardware

The hardware needs to be installed at the production site and needs to have a camera interface, internet access capabilities such as a wifi or ethernet connection. For the industrial environment an ethernet connectivity is preferred to avoid connection loss. The SoC is a key component for a CPU and FPGA integration. The CPU needs to be able to run a specialised lightweight Operating System (OS) such as a distribution of Linux. It needs to support the frameworks needed to integrate the machine learning algorithms. The FPGA needs to have enough logic elements and fast RAM to be able to incorporate the machine learning neural network.

In addition a camera is needed with a connection to the hardware and existing drivers within the OS. For the camera the main specification is the resolution. The observable area is around 4m*4m=16m^2. A detectable object has to be at least 15 pixels wide. The distance between camera and the object is between 3-7 meters. A HD 720p camera would detect therefore an object of the size of 0.093m = 9.3cm. In order to have a certain security margin it is the resolution choosen for the prototype camera. HD 720p is a common resolution and can be found in many commercial available cameras.

5.2 Selection

After processing all specifications of the different elements in the section 5.1, possible technologies for these categories are listed, evaluated and finally selected.

5.2.1 M2M Communication

There are multiple protocols standards well known for IoT applications 2 Comparison IoT protocols (Naik, 2017, p.3).

Features	*AMQP*	*CoAP*	*MQTT*
Transport	TCP/IP	UDP/IP	TCP/IP
Paradigm	P2P Message Exchange	REST	Publish / Subscribe
Scope	D2D D2C C2C	D2D	D2D D2C
RESTful	No	Yes	No
Fault Tolerance	Implementation Specific	Decentralized	Broker is the SPoF
Messaging	Asynchronous	Asynchronous & Synchronous	Asynchronous
Header size	8 bytes	4 bytes	2 bytes
Memory	⊕	⊕⊕⊕	⊕⊕⊕⊕
Power consumption	⊕⊕	⊕⊕⊕	⊕⊕
Implementation	⊕⊕	⊕	⊕⊕⊕⊕
Security	⊕⊕⊕	⊕⊕⊕⊕	⊕⊕

Table 2: Comparison IoT protocols

For the prototype, MQTT was choosen because of multiple reasons. The publish / subscriber paradigm allows to easily extend the system and attach other devices to the communication network. It can communicate with any endpoint, thanks to its broker centric setup. It has the lowest memory footprint of all proposed protocols. There exists easy to use implementations for all common platforms. A Transport Layer Security (TLS) can be added, but because of the networking layer from Syrto AG, all communications are secured by default over a Virtual Private Network (VPN) Tunnel. Lastly, Syrto AG already has this protocol in use on other sites.

5.2.2 Framework & Hardware

The selection of the framework goes together with the hardware. The software and Programmable Logic frameworks needs to interact with each other, and are supported by the hardware.

Most ML frameworks are tighly integrated with the Python programming language. Support for Python was a key selection criteria. It will help for the future and the catalog of possible ML implementation is extensive.

There are only three SoC companies selling chips supporting the SoC specification mentioned 5.1 Specification. Xilinx, Actel and Intel (formerly know as Altera). Actel is focused on very low

power and aerospace technology, this leaves only two manufacturers. An incomplete list of best suitable hardware, including hardware specifications, was created (table 3 Development Boards Comparison). All mentioned boards would satisfy the specifications of the prototype.

Specification Item	*Digilent PYNQ-Z1*	*Digilent Embedded Vision Bundle*	*Terasic DE1-SoC Board*	*Enclustera Mars XU3 & ST3*
Type	Development Kit	Development Kit	Development Kit	Industrial Grade Board
SoC	Xilinx ZYNQ XC7Z0201CLG400C	Xilinx Zynq XC7Z0101CLG400C	Intel Cyclone V SoC 5CSEMA5F31C6N	Xilinx ZYNQ Ultra-scale+
CPU	Dual-core Cortex-A9 @ 650MHz	Dual-core Cortex-A9 @ 667MHz	Dual-core Cortex-A9	Quad-core Cortex-A53 @ 1200MHz
GPU				Mali-400MP2
FPGA	Artix-7 13,300 logic slices 630kB block RAM 4 clock management tiles 220 DSP slices	Artix-7 17,600 logic slices 270kB block RAM 2 clock management tiles	Cyclone 5 85k logic elements 120kB block RAM	Artix-7 103k logic elements 256kB block RAM
Memory	512MB DDR3 16MB Quad-SPI Flash MicroSD slot	1GB DDR3 16MB Quad-SPI Flash MicroSD slot	1GB DDR3 64MB SDRAM Flash	1GB DDR4 64MB Quad-SPI Flash 16GB eMMC Flash MicroSD slot
Interfaces	USB-JTAG & UART & OTG 2.0 PHY Gigabit Ethernet PHY HDMI sink & source	USB-JTAG & UART & OTG 2.0 PHY Gigabit Ethernet PHY HDMI sink & source	USB-UART & Host 2.0 Gigabit Ethernet PHY 24Bit VGA	USB 3.0 & OTG 2.0 Gigabit Ethernet PHY PCIe Gen2x4 Mini DisplayPort
Powersupply	USB or any 7V-15V source	USB or any 5V source	12V	12V source
Programming IDE	Ubuntu 18.04 Linux Xilinx Vivado Python & IPython Kernel	Linux Xilinx Vivado	Intel Quartus II Eclipse IDE	u-Boot Linux
Programming Framework	VHDL cmake C++ Python	VHDL cmake C++	VHDL cmake C++	VHDL C++
Price	181CHF (199$)	272CHF (299$)	227CHF (249$)	1092CHF + 364CHF (1200$ + 400$)
Rating	⊕⊕⊕⊕⊕	⊕⊕⊕⊕	⊕⊕⊕	⊕

Table 3: Development Boards Comparison

(*PYNQ-Z1: Python Productivity for Zynq-7000 ARM/FPGA SoC*, 2020, n.p.) (Inc., 2020, n.p.) (Technologies, 2020, n.p.) (Enclustra, 2020b, n.p.) (Enclustra, 2020a, n.p.)

For implementing the outlined use case in 4,a development FPGA board was choosen available from the company Digilent. The board is part of the Xilinx PYNQ project and is called PYNQ-Z1. The PYNQ project intends to enable productivity gains, by combining a very high level language (Python) with the power of an FPGA and its reconfigurable hardware acceleration. The solution and many third party modules are open source and available. Thanks to a modular design choice, the developer can use preexisting hardware accelerators available by the community. The most popular project revolve around topics like Networking, Machine Learning with implementations for Quantized Neural Network (QNN), Binary Neural Network (BNN), Deep Learning, Neural Networks and CNN with the FINN framework (Umuroglu et al., 2017, p.1), Computer Vision, Image Processing and Robotic. It is an ideal playground for realising and developing educational, but also industrial projects.

The Digilent PYNQ-Z1 Board (see figure 7) is selected, mainly because of three reasons. The Python for Zynq (PYNQ) community supplies a ready to install ARM compatible Linux, based on Ubuntu 18.04 distribution. An open source framework was developed to tightly integrate Python and FPGA programmation with the Xilinx Vivado Integrated Software Development (ISE) toolkit. This framework allows for a rapid prototyping and using existing machine learning networks (Wang et al., 2018, p.1). Thanks to the capability to partial reconfigure the PL and the integration of Jupyter based notebook there are immense productivity benefits during development (Hutchings & Wirthlin, 2017, p.5-7). Additionally it allows to create programs running either as ”software only” on ARM CPU architecture on the board itself, as well as x86 CPU architecture of a desktop or laptop computer and running as ”Processing System (PS) / Programmable Logic (PL)” combination on the ARM CPU architecture and the FPGA. Direct comparisons of the same program can be made. The board is powered by an Xilinx ZYNQ-XC7Z020 SoC harboring a dual-core ARM Cortex-A9 running at 650MHz. This allows to run a Python program, as well as a MQTT broker, directly on the edge device. A self sustained prototype can be build without the need of external dependencies.

Since machine learning neural networks are using a substantial amount of block Random Access Memory (RAM), the SoC chip provides 630kBytes of memory, the highest amount of all selected boards. Therefore multiple and bigger networks can be implemented in the future. Internet con-

nectivity is provided either via a Wifi module or Ethernet connector. For the camera interface three methods are possible: High-Definition Multimedia Interface (HDMI), Ethernet or via regular Universal Serial Bus (USB). Since the chip is running on an Ubuntu based Linux OS, drivers are abundantly available, especially for low cost USB.

The manfacturer Xilinx and the community developed PYNQ, an open-source framework that makes it easier to exploit PL and microprocessors in parallel. It provides applications with parallel hardware execution, high frame-rate video processing, hardware accelerated algorithms together with a high bandwidth and low latency control. All tighly integrated into the Xilinx Software Development Kit (SDK) (Xilinx, 2020, n.p.).

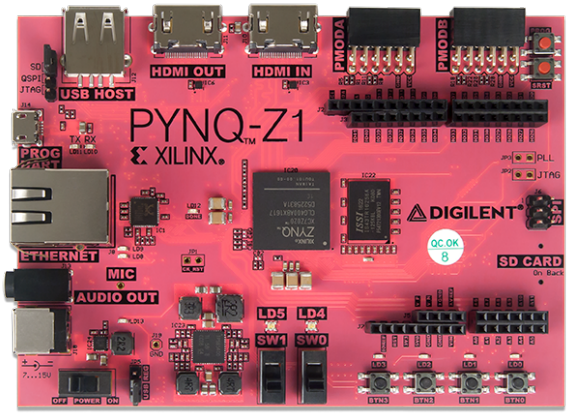


Figure 7: PYNQ-Z1 Board Overview (Xilinx, 2020, n.p.)

For the camera a standard off the shelf webcamera from Logitech is choosen. The Logitech C270 incorporates a HD 720p sensor and cost as low as 35 CHF. Drivers for Ubuntu are available and a 30fps (frame per second) video stream is transmitted via the USB connection.

The PYNQ-Z1 board comes with the PYNQ framework which is split in four layers 8. In the FPGA existing open source Intellectual Property (IP) cores as well as user IP cores can be implemented and deliver real-time capabilities. Through the Xilinx Runtime (XRT) called *Xlnk* the FPGA can be triggered to be reprogrammed at any time during runtime within the Python program.

The second layer is the Linux kernel and is running on the CPU. Since it is Ubuntu based, many drivers and thirdparty applications exists. The link to the FPGA is provided via an AXI ARM AMBA interconnection bus. Through the FPGA manager in the linux kernel, the FPGA can be fully or partially reprogrammed.

The third layer is a software layer which runs on the Python. Many existing open source PYNQ libraries can be used. They contain PL IP cores, the FPGA resources are accessed directly within a Python program. The main link between FPGA and Python is the Direct Memory Access (DMA) through the Linux Kernel.

The last layer is the application layer. Programs can run either in a Jupyter notebook (a well known framework for data scientist and machine learning) or directly in Python with help of the iPython Kernel. All the well known Python libraries ported to the ARM CPU architecture are supported. Such as NumPy, Pandas, OpenCV, scikit-learn and PyTorch. Many of these libraries are used to implement the prototype application for the EPU.

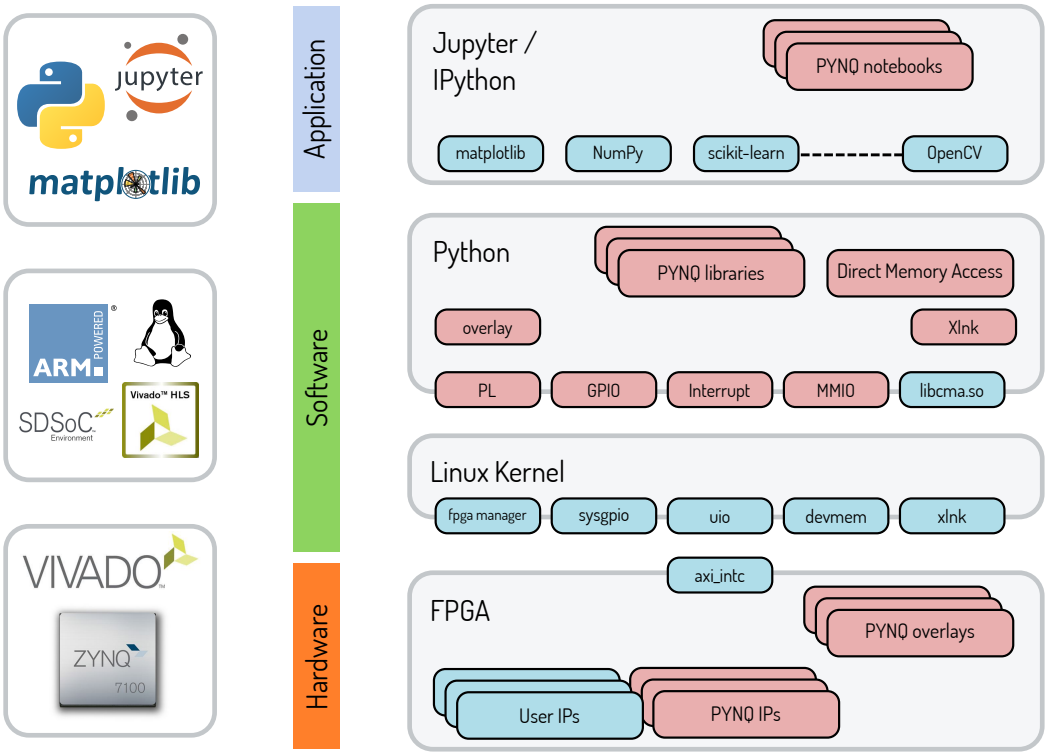


Figure 8: PYNQ Framework Overview based on (Xilinx, 2020, n.p.)

6 Implementation

The prototype contains 3 main elements highlighted in figure 9. An EPU device detecting people, a MQTT broker for message transmission and a user Graphical User Interface (GUI) as proof of concept application for the detection. In addition any other device subscribing to the MQTT messages such as the on-side PLC can perform action upon detection of a person by the EPU. Syrto AG mainly uses PLC systems to control the automated machines, the integration of the prototype into their systems is not part of the implementation explained here.

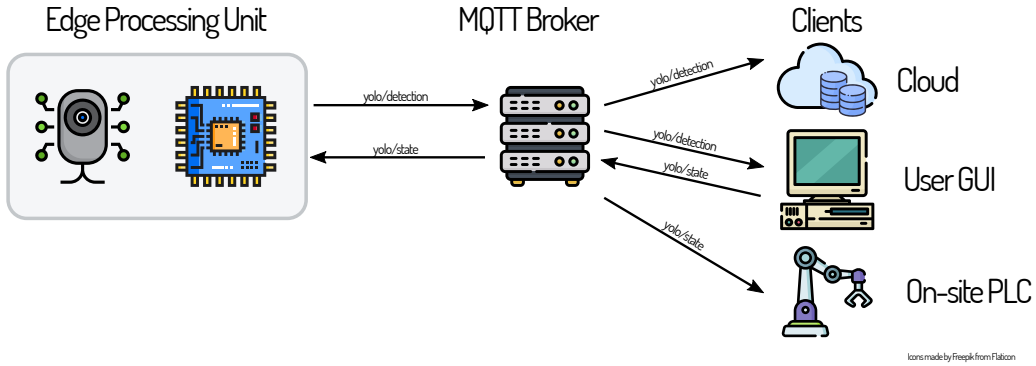


Figure 9: MQTT Setup Overview

6.1 Edge Processing Unit

The main objective of this EPU is to capture picture with the help of the webcam and detect people with the PS and the PL located on the PYNQ-Z1 board. The webcam is connected with a USB interface and also the connectivity is guaranteed with an ethernet interface. Both elements are directly linked to the CPU also called PS. Some General Purpose Input Output (GPIO) are used for debug purposes, the other interfaces such as HDMI, microphone, audio out, Pmod and Arduino headers are unused.

The MQTT broker is directly integrated on the edge monitor to avoid adding additional external dependencies to the system. The number of clients who connect to the broker can be extended as needed.

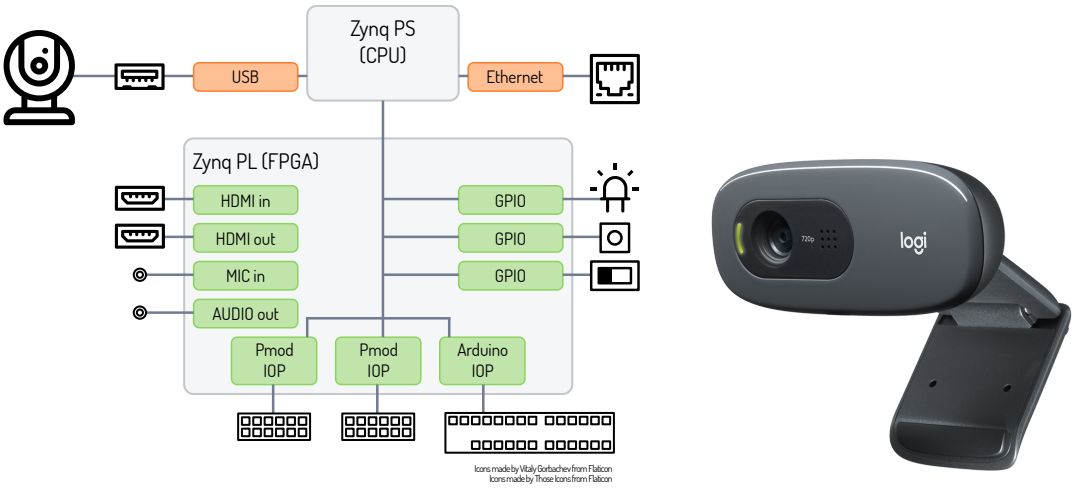


Figure 10: On the left PYNQ-Z1 bloc diagram and on the right Logitech Webcam 720p HD

6.2 Program Sequence

The sequence diagram 11 shows the three main actions the system is performing: start, detection and stop. The start command from one of the connected clients launches the people detection software. Not shown in the diagram is the programmation of the FPGA which happens at the powerup of the EPU, this can take several seconds and is further elaborated in the section 7. Once the people detection software is started, the detection action is performed continuously. It requests images from the camera and invokes for each individual image the ML framework working within the PS and the PL. Only in case a person is detected, detection boxes are drawn on top of the image and a MQTT message is sent through the MQTT broker to all connected clients. The last action is the stop command, which halts the people detection algorithm. During the stopped state, the FPGA can be reprogrammed on the fly without restarting the EPU.

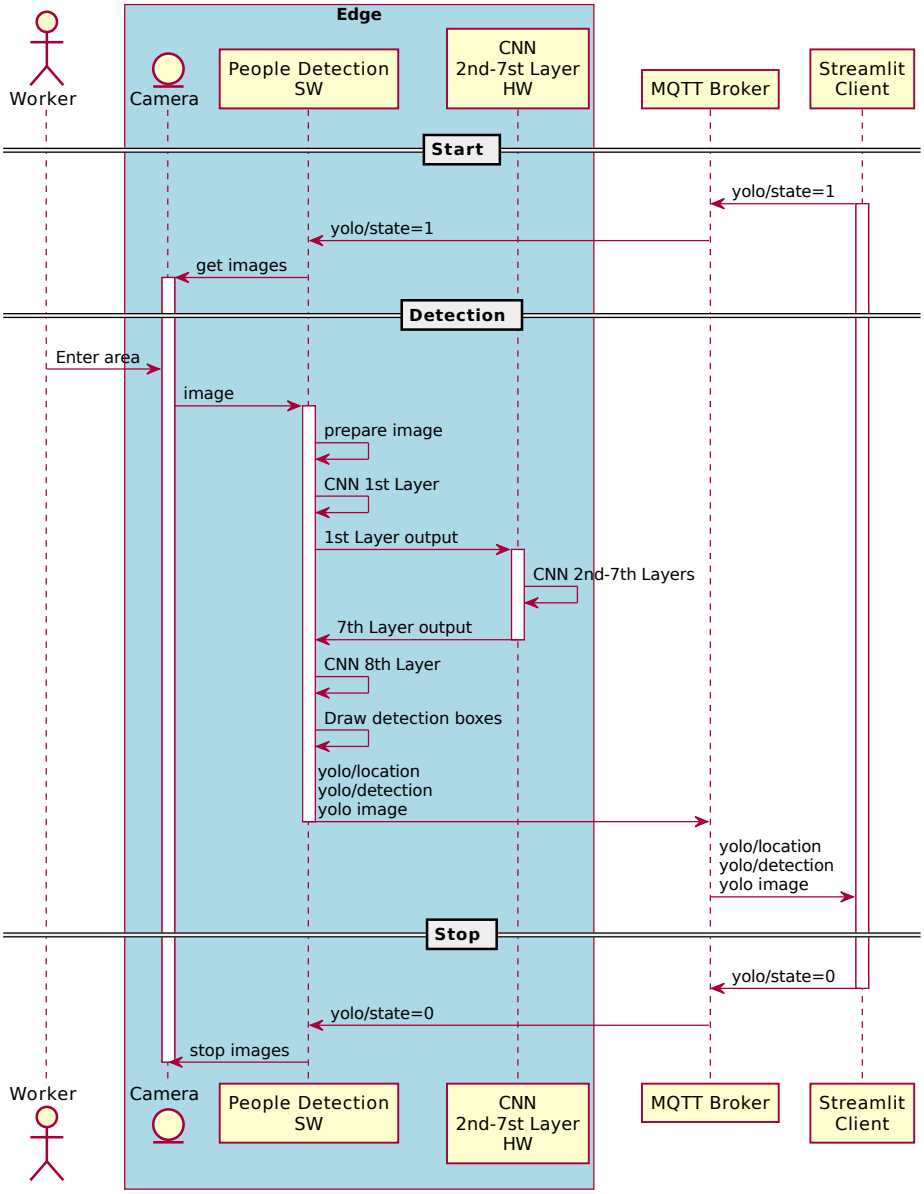


Figure 11: Sequence Diagram

6.3 Program Structure

The program is designed in such a way that it can run with or without hardware accelerator on either an ARM or x86 CPU architecture. The figure 12 shows the conceptualized class diagram of the program, a detailed version is located in the appendix 26. Within the configuration file (see 4) the static settings are defined. Many depends on the MQTT broker setup as well as the type of hardware it runs. Some possible configuration is only used during developement such as `show|save_images`.

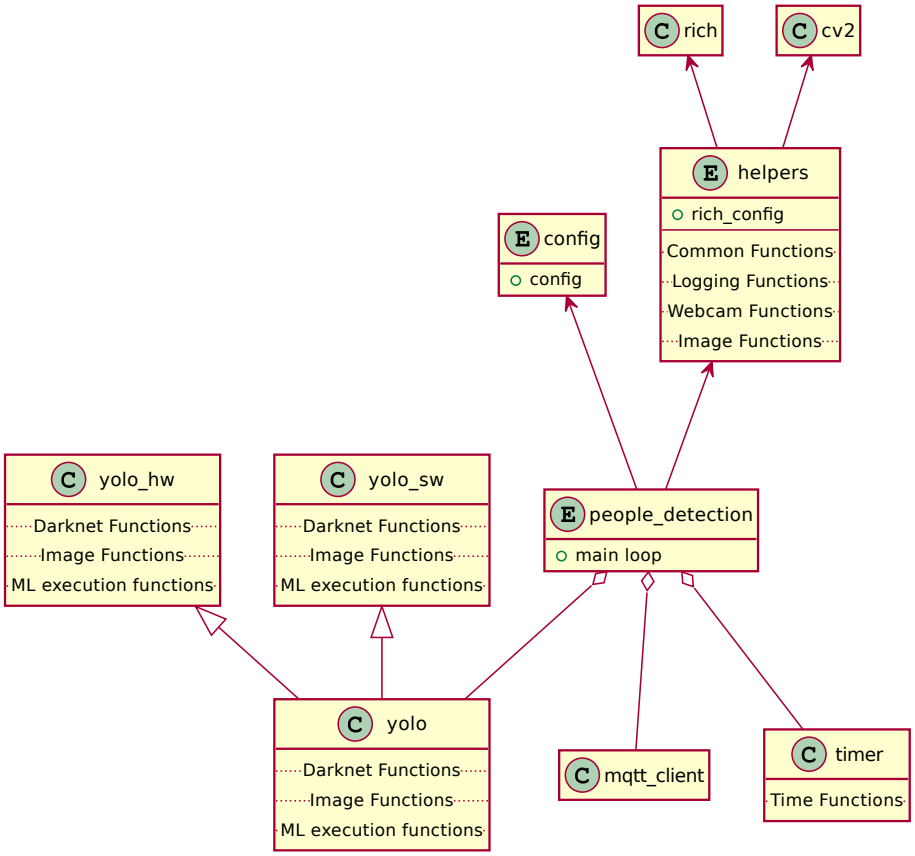


Figure 12: Class Diagram Concept People Detection

Hereafter a description of the individual elements:

- People Detection - Main class which includes the program main loop. Depending on the configuration, all other objects are instantiated and configured accordingly.
- Config - The configuration object allows to read and write the configurations ini file. The whole configuration is stored there and made available to all other objects.
- Helpers - The helper object contains various functions, for which no separate object was created. The functions can be divided into four categories: common functions, logging functions, webcam functions and image function. This class is an abstraction layer for the two python modules rich and OpenCV.
- Timer - This custom timer object allows to measure individual parts of the execution of the program in miliseconds. It can create a Comma Separated Values (csv) file for further comparison. With this class the different experiments are measured and recorded.
- MQTT Client - It handles all the communication with the MQTT broker. It allows to subscribe to different topics as well as publishing messages. It can connect to brokers with or without

authentification and has the last will feature of MQTT implemented. This feature allows to detect and monitor unexpected loss of connections.

- YOLO - Parent class for the YOLO algorithm implementation. It serves as a blueprint for the architecture specific implementation of the algorithm. Depending on the configuration one of the two different child classes are used.
  - YOLO HW - Hardware accelerated implementation of the YOLO CNN. The first and the last neural network layer is executed in software and the layers 2-7 are executed within the FPGA. The darknet detection is integrated in this class as well.
  - YOLO SW - Purely software implementation of the YOLO CNN. The neural net has the same configuration i.e. structure activation and weights as its hardware accelerated counterpart.

6.4 ML Implementation

Within the PYNQ project the FINN and FINN-R experimental framework allows for building fast and flexible FPGA accelerators using a streaming architecture (Umuroglu et al., 2017, p.1). With the help of these frameworks CNN in the form of BNN as well as QNN can be implemented (Umuroglu et al., 2017, p.2-3) (Blott et al., 2018, p.2-3). Some neural networks are available without implementation effort. The selected YOLO-v3 tiny is based on a QNN to reduce memory footprint which allows it to fit in the on-chip memory which delivers a much higher bandwidth and reduces energy consumption (Blott et al., 2018, p.2). The ML model is based on eight layers, which it classifies as deep neural net. In figure 13 the CNN architecture is shown in detail.

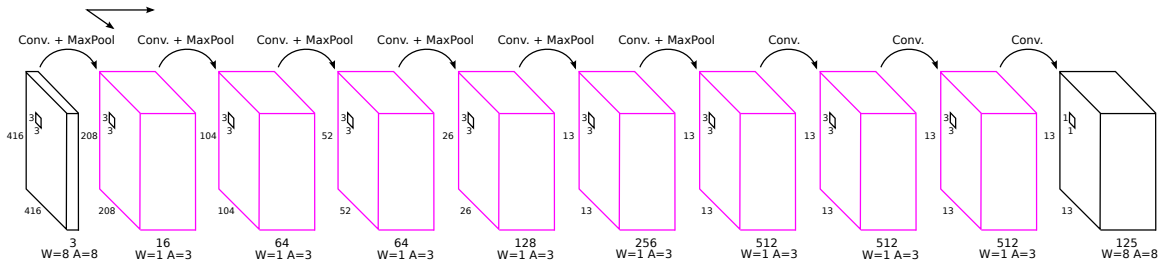


Figure 13: Tinier YOLO CNN Topology (*Xilinx/QNN-MO-PYNQ*, 2020, n.p.)

6.5 Client

The client app is designed to complete the proof of concept of the prototype and serves as an implementation example for future applications. It allows to start and stop the EPU with the right hand side buttons and shows the incomming messages as image as well as raw JavaScript Object Notation (JSON), it is the message type choosen for the communication. The application uses some elements from the EPU to establish communication with the MQTT broker and display of the images itself. The figure 14 shows the conceptualized class diagram of the application, a detailed view is found in the appendix 27.

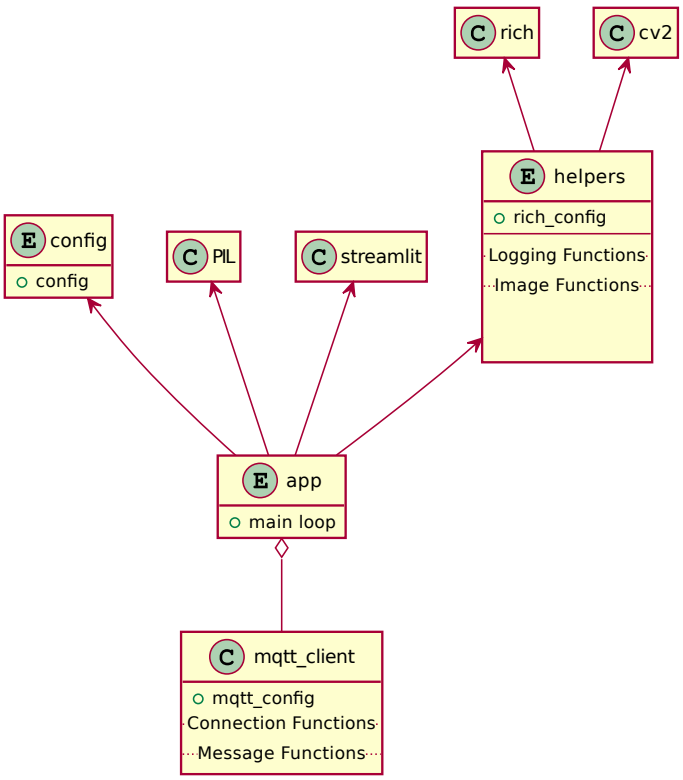


Figure 14: Class Diagram Concept Streamlit App

Hereafter a description of the individual elements:

- App - Main class which includes the program main loop. Depending on the configuration, all other objects are instantiated and configured accordingly.
- Config - Adapted version of the configuration class used in the EPU see 6.3. The configuration object allows to read and write the configurations ini file. The whole configuration is stored there and made available to all other objects.

- Helpers - Same class as used in the EPU with some unused functions stripped see 6.3. The helper object contains various functions, for which no separate object was created. The functions can be divided into two categories: logging functions and image function. In addition this class is an abstraction layer for the two python modules rich and OpenCV.
- MQTT Client - The same implementation as used in the EPU see 6.3 for more informations.
- Python image library (PIL) - Used for reconstruct, modify and display the raw image sent over MQTT.
- Streamlit - Python module for creating dashboards with a webinterface, the webserver is included.

In figure 15 the developed interface is shown. On the left hand side all the controls are located and the right hand side displays informations from the receiving messages. The topbar in green or red indicated if the EPU is currently running or deactivated.

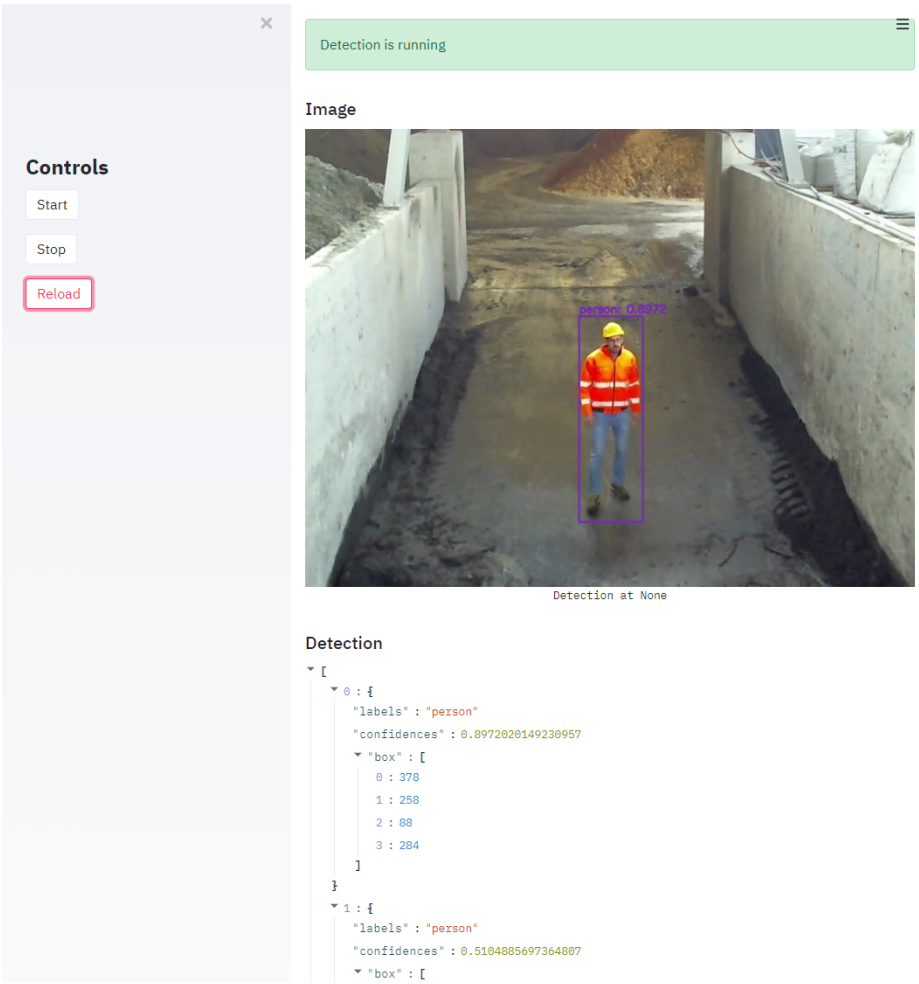


Figure 15: Interface Streamlit application

6.6 Communication

The communication of the POC is based on a lightweight publish-subscribe network protocol called MQTT. A central broker runs on the EPU as indicated in figure 9. The system is based on predefined topics which publisher emits and any client can subscribe to. The main topics is called `yolo`, all messages emit are within this topic. All subtopics from the system are listed in table 4

MQTT Topic	*Type*	*Description*
`yolo/control/status`	Bool	Current state of the EPU
`yolo/location`	String	Station detected the object
`yolo/detection`	List	List of objects detected
`yolo/detection/labels`	String	Detected Element
`yolo/detection/confidence`	Int	Object detection confidence value between $[0, 1]$
`yolo/detection/box`	List of Int	Detection box of the object X_0, Y_0, X_1, Y_1
`yolo/image`	String Base64	Image encoded with Base64
`yolo/state`	Bool	Control command to start or stop the EPU

Table 4: MQTT Topics

The EPU acts simultaneously as a publisher to send the messages in the subtopics `locations`, `detections` and `image` (listing 2) and as a subscriber to the subtopic `state` (listing 3). To conserve bandwidth the `image` topic can be omitted. In listing 2 an example message is shown from the EPU called ”workstation_1” detected an object ”person” with a confidence of 68.6% at the pixel location (247,63,132,334) within the image. In addition the images is transmit encoded as Base64 string.

```
{
  "location": "workstation_1",
  "detections": [
      {
          "labels": "person",
          "confidences": 0.6866945624351501,
          "box": [
              247, 63, 132, 334
          ]
      }
  ],
  "image": "/9j/4AAQSkZJRgABAQAA ... mnmMZ0vYxsvxPm3h4up7R6tH
}
```


Listing 2: MQTT message from the EPU

In listing 3 the example message will start the poeple detection feature of the EPU.

```
state=1
```

Listing 3: MQTT message to the EPU

7 Results

As already elaborated in the section 2.2 and 2.3, experiments are selected to evaluate the system. Four experiments are conducted in total, to evaluate four main characteristics of an EPU, for person identification. These main characteristics are power consumption, data transfer volume, processing speed and detection rate.

For each experiment, different factors were varied according to the cause and effect diagrams (5). Throughout all experiments, the same selection of images are used, which are described in the chapter 4.3. The images were taken with the selected Logitech C270 camera, the tests do not take into account the speed of the camera interface between different platforms. This allows to perform consistent tests with the exact same images.

7.1 Detection Rate

In this chapter, at first the overall detection rate is analyzed. In the subchapters 7.1.1, 7.1.2 and 7.1.3 various details or problems are addressed further.

In the chapter 4.3 it is described that 28 different scenes with a total of 790 images were analyzed. Each image was evaluated by the YOLO-v3 tiny ML model. A scene is an approach of a person into the monitored area. For each scene, the person was detected by the system on at least one image of that scene. It is worth mentioning that the camera field of view is larger than the monitored area. This means a person who is “out of range“, is not located inside the monitored area. These images were nevertheless used to evaluate the limits of detection.

Figure 16 shows that overall in 60.1% of the images the person was correctly detected and in 6.57% the person was not detected. The remaining 33.3%, can be split into two groups. In 20.1% of the images the person was located outside the surveillance zone and for 13.2% exposure problems from the sunlight or artificial light were the reason why the quality of the image was not sufficient to allow a detection.

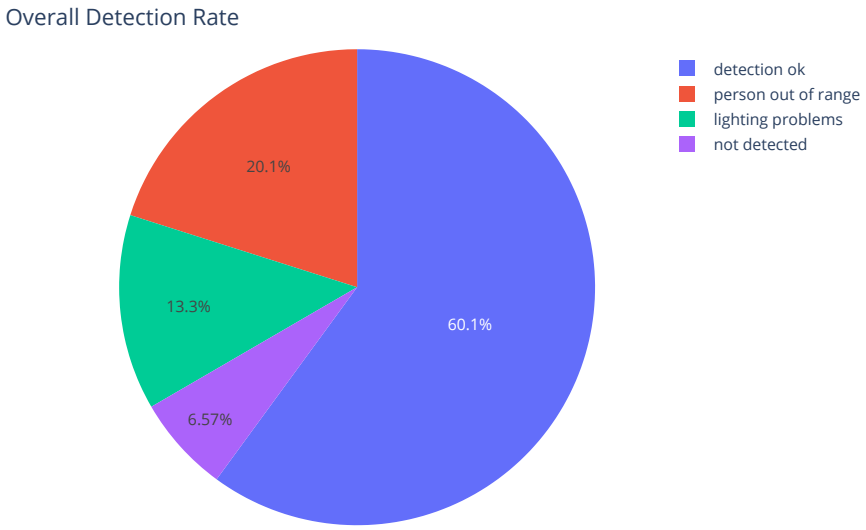


Figure 16: Overall detection rate

7.1.1 Person Size

The size of the person plays an important role in the pre-trained YOLO-v3 tiny model. The reliability as well as the confidence of the detection, is at its best, when the person in not too close (i.e. only a small proportion of the body is shown) or too far away (i.e. the body consists of a low number of image pixels). In the figure 17, it can be seen that for the same image once analysed in total and once cropped to the monitored zone, the confidence of the algorithm that the object is a person increases from 39.72% (left) to 74.62% (right).

Figure 17: Comparison person size (left full image, right cropped)

7.1.2 False Detection

A ML model is not a deterministic algorithm, it cannot guarantee that the detected element effectively matches the object, it can only give a confidence level. In the following examples figure 18

(from left to right), the person was detected as a bottle as well as a fire hydrant and in the last figure a person was detected twice. 1.91% of all images had a falsed detection. In the developed EPU system, only the person detections are forwarded and all other detections are ignored. It is worth mentioning that no other object has been falsely identified as a person in all evaluated images. The YOLO-v3 tiny CNN was not specifically trained for these images. The network used was pretrained with the COCO dataset, to improve detection rate YOLO can be trained from scratch with custom images.

Figure 18: False detection examples

7.1.3 No Detection

In 6.57% of the images that were analysed, no person was detected although at least one person was present. As already described in the chapter 7.1.2, this highly depends on the images used to train the CNN. However, the person was detected in at least one other image of that same scene. This allows the EPU to send a message and stop active equipment anyway.

Figure 19: No detection examples

7.2 Energy Consumption

The power consumption was measured for all tests with a Metex M-4660A multimeter. Three devices were selected for comparison, the Xilinx PYNQ-Z1 board, a Spectra Box PC with an Intel i7 processor and 8GBytes RAM and a Siemens Nanobox PLC with an Intel Celeron processor

and 4GBytes RAM which is used by Syrto AG in various installations. All three systems were tested in the three states: startup, idle and under load. Each measurement was repeated three times. The results of the measurements can be seen in the figure 20.

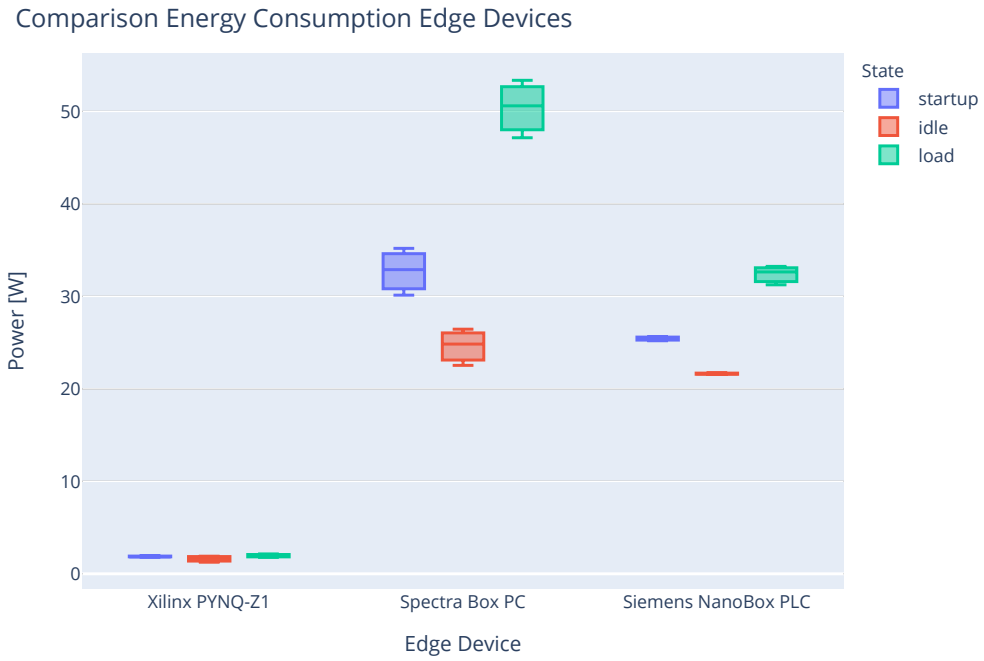


Figure 20: Energy consumption comparison

It can be seen that the developed EPU has a clear advantage in the area of power consumption, the fact that it is based on an ARM processor and some calculations can be outsourced to the FPGA, helps to achieve this advantage. The Box PC consumes on average 19 times more energy and the Siemens Nanobox 14 times more.

7.3 Transfer Volume

One of the most important parameters of an EPU is the ability to perform calculations on-premise. In this experiment, three different methods are compared. First, the EPU which transmits only the information of a detected person. Second an EPU which transmits the information of a detected person and additionally a compressed image, because it has already been analyzed and is sent only for verification. Finally a pure cloud approach where each image has to be forwarded to the cloud for further processing.

For the tests, a mean image size of the Logitech camera was used, which was set to 50kBytes after preprocessing. The number of messages from the EPU depends strongly on the situation as it sends only a message if a person is detected. It was estimated that the EPU detects a person every 10 seconds and the cloud model has a frame rate of 5fps. The figure 21 shows the transferrate on a logarithmic scale.

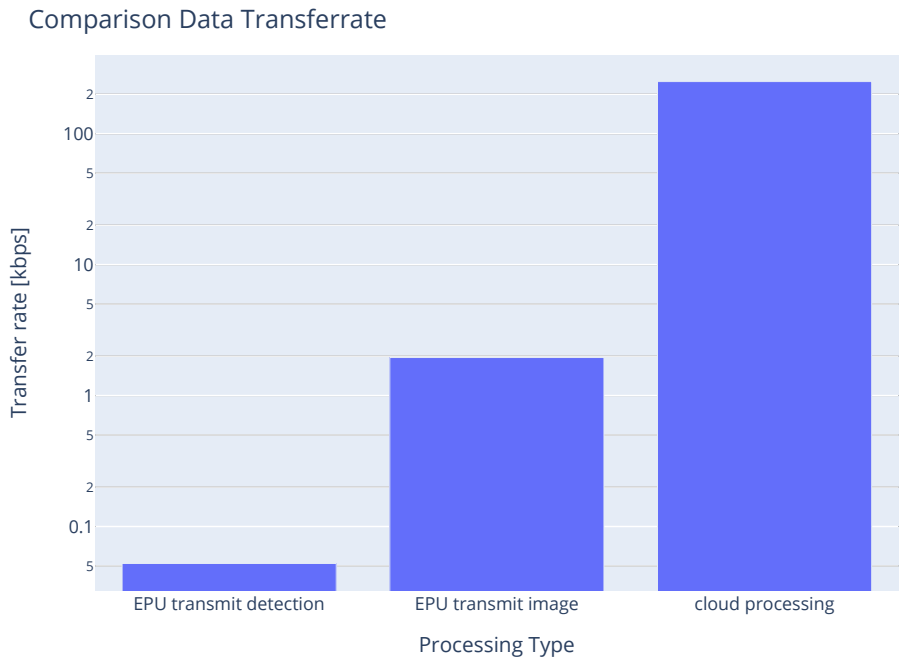


Figure 21: Transferrate comparison edge vs. cloud (logarithmic scale)

An EPU that does not transmit images has a data rate of 0.0522$\frac{kbit}{second}$, if the image is sent as well it is increased to 1.052$\frac{kbit}{second}$, in case of cloud processing a data rate of 250$\frac{kbit}{second}$ is needed. The data transfer directly affects the response time. If the MQTT broker is also installed locally, as in this experiment, the EPU has the advantage that local systems can be notified without external data transmission.

7.4 Processing Speed

For the last experiment, the processing speed of the program is evaluated. Three devices are compared. First, the PYNQ-Z1 operating as intended with a combination of software and hardware processing, leveraging the advantages of the FPGA. Second, the same hardware using the same algorithm, implemented purely in software without using the FPGA hardware acceleration.

And finally a reference laptop DELL XPS 13 9300 with an Intel i7 mobile processor and 16GBytes RAM as well as a fast Non-Volatile Memory Express (NVME) storage. The speed test is split into the different operations the algorithm performs during image analyzation. The figure 22 shows the result of the imageset selected at 4.3. For each action the colors represent the different devices. For each device on the left side are the dots of each individual measurement and on the right side is a boxplot of these measurements.

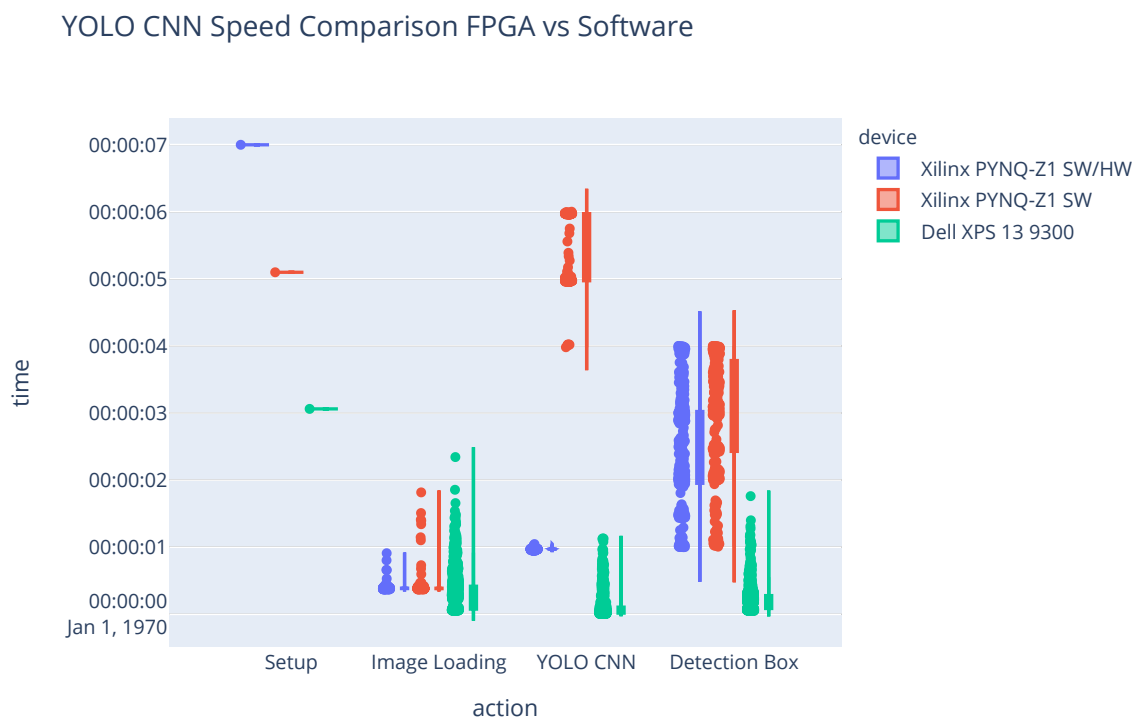


Figure 22: Image Processing Speed Comparison Hardware vs. Software

The action “Setup“ is the one-time initial setup of the ML CNN. In case of the PYNQ-Z1 using software and hardware acceleration the time is significantly longer at 6.99s because the FPGA need to be partially reconfigured. The PYNQ-Z1 using software only, does not need to perform this programmation of the FPGA and is therefore faster at 5.09s and the Dell XPS 13 is the fastests with 3.06s. Since this action is performed only once at boot-up, it does not affect runtime performance.

For the action “Image Loading“, the Dell XPS 13 with a Microsoft Windows OS performing many background tasks has a bigger scatter, but is overall faster with a median of 278ms compared to the PYNQ-Z1 board. The median for the PYNQ-Z1 is roughly the same in software/hardware or software only configuration with 382ms compared to 385ms. This is supported by the fact that the same work is performed and cannot be offloaded to the FPGA.

The biggest difference can be observed in the action “YOLO CNN“. The performance of the Dell XPS 13 thanks to the Intel i7 processor is the fastest with a median of 51ms. The PYNQ-Z1 with hardware acceleration needs 971ms for the same calculations. Compared to the PYNQ-Z1 in software only configuration however a five time speed increase can be observed. The FPGA allows to unburden the ARM processor and achieves a significant speed increase.

The last action to be performed is the drawing of the detection box. It is worth mentioning that the YOLO CNN already specifies the position of the object in the image and this action is an optional step. With the information in the transmit MQTT messages it could also be executed by the receiver said messages. Again, the Dell XPS 13 is by far the fastest with a median of 211ms. The PYNQ-Z1 needs 2.38s in software/hardware mode and 3.02s in software only mode. This large difference can be explained mainly by the storage medium. The Dell XPS 13 has a fast NVME storage medium attached to the processor, whereas the PYNQ-Z1 has to read and write the images on an SD-card.

7.5 SWOT Analysis

The figure 23 shows a SWOT diagram evaluating the EPU and its application uses. In general the technology behind EPU, which is software applications accelerated with custom reconfigurable hardware, is currently more used in datacenters. But the evolution of frameworks and development kits for this technology, makes it more viable to be used also outside of custom applications and datacenters. The future of this technology has not yet reached its potential and the rise of ML algorithms will just accelerate the development.

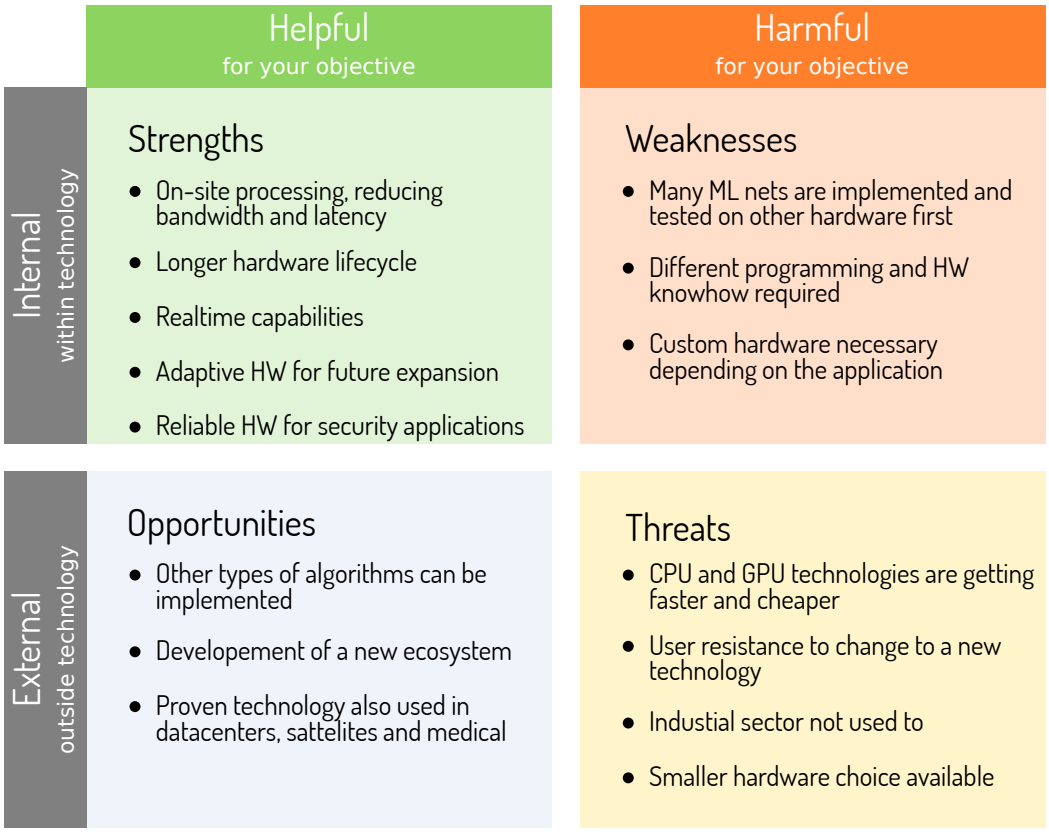


Figure 23: SWOT Diagram based on (Xhienne, 2007, n.p.)

8 Conclusion

8.1 Comparison with the Initial Objectives

This project was a success. The main goal to create a proof of concept of an EPU system and to test it in a practical example could be realized.

Because the libraries can be used on different hardware, the goals could be underlined with relevant comparisons. Only single parameters could be changed between the experiments.

The energy consumption could be reduced nineteen fold compared to a traditional box computer system normally used as edge devices. The combination of a power efficient core with the custom hardware accelerator implemented in a reconfigurable FPGA is reducing energy consumption significantly while maintaining the required processing speed and power. The goal to reduce the bandwidth was clearly exceeded with a hundredfold reduction of the transmitted data. This is apparent also with traditional edge devices and one of the reasons why the combination of edge processing and cloud processing is so important. Also the latency could be reduced by on-site analysis instead of an off-site analysis.

Edge devices using hardware accelerators require more development knowhow not only for the software part, but especially for the development and integration of the custom hardware elements located in the FPGA. The SDK software for developing software and hardware are very different and require a different development strategy. Currently this knowhow is lacking in many small and medium sized companies. Thanks to new frameworks such as PYNQ and existing reference implementations, the gap between the two worlds is shrinking and allows developer with traditional software development skills to work with FPGA's in a well known environment such as Python and Jupyter notebooks.

Traditional hardware, which is better known and more common, but also more expensive, achieves a similar computational speed. The cost of all hardware used for the proof of concept was below the specified limit at 220 CHF.

8.2 Encountered Difficulties

During the development of the solution, various problems arose. The selection of an ML framework was more time consuming than previously estimated. Some tested ML models were well suited according to the description. In practice the results were mixed and many false positive or false negative results were observed. The time loss can be explained by need of testing different ML models.

The goal of making the same program run on different hardware has proved to be more difficult than expected. In particular not all functions of the OpenCV library were ported to the used ARM architecture. This problem was solved by switching to other functions.

Another problem was given by a hardware limitation. The selected PYNQ-Z1 board has 512MBytes of RAM. This is sufficient to run the system including the MQTT broker by offloading part of the YOLO algorithm from the CPU to the FPGA. In the experiment where the entire algorithm runs on the CPU, the dynamic garbage collector used by Python does not free up ram space fast enough. Depending on the size of the image and the number of images, the memory limit was exceeded, causing the system to crash. The problem could be avoided in this case by not instantiating the MQTT broker on the EPU. In addition, the frequency of recorded images was reduced.

Another hardware limitation was that only one USB interface was available. This meant that the Wifi connectivity module and the USB camera could not be used in parallel. The original installation location had to be moved slightly and an Ethernet installation was created to connect the EPU directly to the network.

8.3 Future Perspectives

There are a variety of ways to further develop the project. How to proceed depends mostly on the viewer and the next goal.

For the company Syrto AG, there are several possibilities. To have the basic building block for an extensible system it is advantageous to first provide a site-wide MQTT broker. This avoids that the EPU are charged with a MQTT broker and allows the possibility to connect a large number of identical EPU's to the system. In this project, software was created for different hardware

types. Syrto already has a lot of PLC hardware in different places which have spare computational capacity. The software can be used on existing maschines without the advantages of hardware acceleration. As soon as new locations need to be addressed, an EPU with hardware acceleration is more suitable. This means that a cost-effective and energy-saving hardware can be installed and the monitoring system becomes dynamic and can be continuously expanded according to requirements.

From a technical point of view, the next step is to create a dedicated hardware board (Printed Circuit Board (PCB)). The current hardware costs around 220 CHF and many interfaces and functions are not used. An industrial board without HDMI, audio, pmod and Arduino interfaces but with more RAM and USB interfaces would be optimal. A future interface for additional sensors should also be included. The production cost of such a hardware is below the current hardware costs and at the same time the size of the board can be reduced significantly.

The last thing for further developing the system, is to train the YOLO network to the special cases encountered. For this purpose, video footage must be recorded with the existing installed cameras. This footage can be used to better recognize special cases such as helmets, special angles of persons, lightings etc. The YOLO ML model allows to be trained with new images or video material. The FINN framework allows to extract the activation and weighting of the QNN from the newly trained YOLO network which can be implemented in the EPU.

8.4 Personal Conclusion

The project was very ambitious and challenging. However, the fact that there was a specific demand and application case in the industry, motivated me strongly. I am satisfied with the overall final results of this project.

Not only could I combine two different areas, neural networks and hardware accelerators. Both topics that in my opinion are very interesting and important for the future of Industry 4.0. But I was also able to use my experience in the area of hardware accelerators in the project and further expand my knowledge in the area of neural nets and machine learning in general.

The project was very ambitious and the split between writing the thesis, developing the prototype,

the job and especially my family was not easy. In particular, the technology research and testing of various machine learning elements took more time than previously estimated. The complexity, of the different approaches but also the sheer number of different types and implementation methods was hard to grasp. Fortunately, I was able to overcome all obstacles and present a minimalistic but working prototype.

Syrto AG has already shown interest in the solution and would like to further develop and evaluate it internally. In addition the work itself is not only applicable within the context of Syrto AG but also in general. Edge computing and hardware accelerators are thematics which will become even more important in the future.

I am pleased that the project was able to make a valuable contribution in supporting a local industrial partner in its digitalization and automation endeavors. This will remain in a positive memory long after this work.

Silvan Zahno

References

Baker, L. M., Liana B. (2015, June). *Intel to buy Altera for $16.7 billion in its biggest deal ever.* https://www.reuters.com/article/us-altera-m-a-intel-idUSKBN0OH2E020150601.

Biggio, B., & Roli, F. (2018, December). Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, *84*, 317–331. doi: 10.1016/j.patcog.2018.07.023

Blott, M., Preusser, T., Fraser, N., Gambardella, G., O'Brien, K., & Umuroglu, Y. (2018, September). FINN-R: An End-to-End Deep-Learning Framework for Fast Exploration of Quantized Neural Networks. *arXiv:1809.04570 [cs]*.

Brender, N., & Markov, I. (2013, October). Risk perception and risk management in cloud computing: Results from a case study of Swiss companies. *International Journal of Information Management*, *33*(5), 726–733. doi: 10.1016/j.ijinfomgt.2013.05.004

Caprolu, M., Di Pietro, R., Lombardi, F., & Raponi, S. (2019, July). Edge Computing Perspectives: Architectures, Technologies, and Open Security Issues. In *2019 IEEE International Conference on Edge Computing (EDGE)* (pp. 116–123). doi: 10.1109/EDGE.2019.00035

Chen, C., Surette, R., & Shah, M. (2020, February). Automated monitoring for security camera networks: Promise from computer vision labs. *Security Journal*. doi: 10.1057/s41284-020-00230-w

Colbert, I., Daly, J., Kreutz-Delgado, K., & Das, S. (2021, March). A Competitive Edge: Can FPGAs Beat GPUs at DCNN Inference Acceleration in Resource-Limited Edge Computing Applications? *arXiv:2102.00294 [cs, eess]*.

Danopoulos, D., Kachris, C., & Soudris, D. (2021, June). Utilizing cloud FPGAs towards the open neural network standard. *Sustainable Computing: Informatics and Systems*, *30*, 100520. doi: 10.1016/j.suscom.2021.100520

Dechelotte, J., Tessier, R., Dallet, D., & Crenne, J. (2018, August). Lynq: A Lightweight Software Layer for Rapid SoC FPGA Prototyping. In *2018 28th International Conference on Field Programmable Logic and Applications (FPL)* (pp. 372–3723). doi: 10.1109/FPL.2018.00070

Enclustra. (2020a, December). *Enclustra FPGA Solutions - Mars ST3.* https://www.enclustra.com/en/products/base-boards/mars-st3/.

Enclustra. (2020b, December). *Enclustra FPGA Solutions - Mars XU3 - Xilinx Zynq UltraScale+ MPSoC Module.* https://www.enclustra.com/en/products/system-on-chip-modules/mars-xu3/.

Fallahlalehzari, F. (2020, February). *FPGA vs GPU for Machine Learning Applications: Which one is better?* https://www.aldec.com/en/company/blog/167–fpgas-vs-gpus-for-machine-learning-applications-which-one-is-better.

Fettke, P. (2006, August). State-of-the-Art des State-of-the-Art. *Wirtschaftsinformatik*, *48*(4), 257. doi: 10.1007/s11576-006-0057-3

Gloger, B., & Schwaber, K. (2013). *Scrum: Produkte zuverlässig und schnell entwickeln* (4., überarb. Aufl ed.). München: Hanser.

Goasduff, L. (2019, September). *Top Trends on the Gartner Hype Cycle for Artificial Intelligence, 2019.* //www.gartner.com/smarterwithgartner/top-trends-on-the-gartner-hype-cycle-for-artificial-intelligence-2019/.

Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., … Chen, T. (2018, May). Recent advances in convolutional neural networks. *Pattern Recognition*, *77*, 354–377. doi: 10.1016/j.patcog.2017.10.013

Hägele, M., Nilsson, K., Pires, J. N., & Bischoff, R. (2016). Industrial Robotics. In B. Siciliano & O. Khatib (Eds.), *Springer Handbook of Robotics* (pp. 1385–1422). Cham: Springer International Publishing. doi: 10.1007/978-3-319-32552-1_54

Hutchings, B., & Wirthlin, M. (2017, September). Rapid implementation of a partially reconfigurable

video system with PYNQ. In *2017 27th International Conference on Field Programmable Logic and Applications (FPL)* (pp. 1–8). doi: 10.23919/FPL.2017.8056845

Inc., D. (2020, December). *Embedded Vision Bundle.* https://store.digilentinc.com/embedded-vision-bundle/.

Jiang, C., Fan, T., Gao, H., Shi, W., Liu, L., Cérin, C., & Wan, J. (2020, February). Energy aware edge computing: A survey. *Computer Communications*, *151*, 556–580. doi: 10.1016/j.comcom.2020.01.004

Ken, S., & Jeff, S. (2020). *The Scrum Guide*.

Kerezovi, T., Sziebig, G., Solvang, B., & Latinovic, T. (2013). Human Safety in Robot Applications - Review of Safety Trends. , 6. doi: ISSN:2067-3809

Korbel, F. (2021, February). *FFmpeg Basics.* http://ffmpeg.tv/.

Kraus, G., & Westermann, R. (2010). *Projektmanagement mit System: Organisation, Methoden, Steuerung* (4., überarb. und erw. Aufl ed.). Wiesbaden: Gabler.

Lee, H. S., & Jeon, J. W. (2020, November). Accelerating Image Processing on FPGAs using HLS and PYNQ. In *2020 IEEE International Conference on Consumer Electronics - Asia (ICCE-Asia)* (pp. 1–2). doi: 10.1109/ICCE-Asia49877.2020.9277085

Leigh, D. (2009). SWOT Analysis. In *Handbook of Improving Performance in the Workplace: Volumes 1-3* (pp. 115–140). John Wiley & Sons, Ltd. doi: 10.1002/9780470592663.ch24

Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., … Dollár, P. (2015, February). Microsoft COCO: Common Objects in Context. *arXiv:1405.0312 [cs]*.

Maier, H. (2019). *Grundlagen der Robotik*. VDE VERLAG GMBH.

Michaelson, H. B. (1974, March). The incremental method of writing engineering papers. *IEEE Transactions on Professional Communication*, *PC-17*(1), 21–22. doi: 10.1109/TPC.1974.6592973

Montgomery, D. C. (2017). *Design and Analysis of Experiments*. John Wiley & Sons.

Naik, N. (2017, October). Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP. In *2017 IEEE International Systems Engineering Symposium (ISSE)* (pp. 1–7). doi: 10.1109/SysEng.2017.8088251

Panchbhaiyye, V., & Ogunfunmi, T. (2021, February). An Efficient FIFO Based Accelerator for Convolutional Neural Networks. *Journal of Signal Processing Systems*. doi: 10.1007/s11265-020-01632-0

Panetta, K. (2021, January). *Cloud Computing Enters its Second Decade.* http://www.gartner.com/smarterwithgartner/cloud-computing-enters-its-second-decade/.

PGDAV College, & Singh, J. (2014, October). Study of Response Time in Cloud Computing. *International Journal of Information Engineering and Electronic Business*, *6*(5), 36–43. doi: 10.5815/ijieeb.2014.05.06

Pop, P., Zarrin, B., Barzegaran, M., Schulte, S., Punnekkat, S., Ruh, J., & Steiner, W. (2021, May). The FORA Fog Computing Platform for Industrial IoT. *Information Systems*, *98*, 101727. doi: 10.1016/j.is.2021.101727

PYNQ-Z1: Python Productivity for Zynq-7000 ARM/FPGA SoC. (2020, December). https://store.digilentinc.com/pynq-z1-python-productivity-for-zynq-7000-arm-fpga-soc/.

Redmon, J., & Farhadi, A. (2018, April). YOLOv3: An Incremental Improvement. *arXiv:1804.02767 [cs]*.

Robinson, G. S., & Reis, J. J. (1980, July). *Real-time edge processing unit* (No. US4213150A).

Satyanarayanan, M. (2017, January). The Emergence of Edge Computing. *Computer*, *50*(1), 30–39. doi: 10.1109/MC.2017.9

Song, W. J. (2021, January). Chapter Two - Hardware accelerator systems for embedded systems. In S. Kim & G. C. Deka (Eds.), *Advances in Computers* (Vol. 122, pp. 23–49). Elsevier. doi:

10.1016/bs.adcom.2020.11.004

Struharik, R. J. R., Vukobratović, B. Z., Erdeljan, A. M., & Rakanović, D. M. (2020, March). CoNNa–Hardware accelerator for compressed convolutional neural networks. *Microprocessors and Microsystems*, *73*, 102991. doi: 10.1016/j.micpro.2020.102991

Technologies, T. (2020, December). *Terasic - SoC Platform - Cyclone - DE1-SoC Board.* https://www.terasic.com.tw/cgi-bin/page/archive.pl?No=836.

Umuroglu, Y., Fraser, N. J., Gambardella, G., Blott, M., Leong, P., Jahre, M., & Vissers, K. (2017, February). FINN: A Framework for Fast, Scalable Binarized Neural Network Inference. *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 65–74. doi: 10.1145/3020078.3021744

Wang, E., Davis, J. J., & Cheung, P. Y. K. (2018, April). A PYNQ-Based Framework for Rapid CNN Prototyping. In *2018 IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)* (pp. 223–223). doi: 10.1109/FCCM.2018.00057

Wei, K., Honda, K., & Amano, H. (2020, November). An implementation methodology for Neural Network on a Low-end FPGA Board. In *2020 Eighth International Symposium on Computing and Networking (CANDAR)* (pp. 228–234). doi: 10.1109/CANDAR51075.2020.00039

Willner, A., & Gowtham, V. (2020, December). Toward a Reference Architecture Model for Industrial Edge Computing. *IEEE Communications Standards Magazine*, *4*(4), 42–48. doi: 10.1109/MCOMSTD.001.2000007

Wouters, I. (2009). *Proceedings of flirting with the future: Prototyped visions by the next generation : SIDeR '09 : Fifth Student Interaction Design Research Conference 15-17 April 2009 Eindhoven University of Technology The Netherlands*. Eindhoven: Technische Universiteit.

Xhienne. (2007, September). *SWOT-Matrix.*

Xia, M., Huang, Z., Tian, L., Wang, H., Chang, V., Zhu, Y., & Feng, S. (2021, May). SparkNoC: An energy-efficiency FPGA-based accelerator using optimized lightweight CNN for edge computing. *Journal of Systems Architecture*, *115*, 101991. doi: 10.1016/j.sysarc.2021.101991

Xilinx. (2020, June). *PYNQ - Python productivity for Zynq.* http://www.pynq.io/.

Xilinx/QNN-MO-PYNQ. (2020, December). Xilinx.

Xuan, S. L., Lann, J.-C. L., Lagadec, L., Fabresse, L., Bouraqadi, N., & Laval, J. (2016). CaRDIN: An Agile Environment for Edge Computing on Reconfigurable Sensor Networks. In *2016 International Conference on Computational Science and Computational Intelligence (CSCI)*. doi: 10.1109/CSCI.2016.0039

List of Figures

1 Gartner Hype Cycle . . . 1
2 Project Plan . . . 6
3 Scrum role according Gloger and Schwaber (2013, p.109) with corresponding mapping 10
4 Scrum flow . . . 11
5 Experiments cause-and-effect diagram . . . 12
6 Use case candidates, from left to right BAUWA AG sludge treatment, BAUWA AG automated crane container, Theler AG concrete factory . . . 22
7 PYNQ-Z1 Board Overview (Xilinx, 2020, n.p.) . . . 32
8 PYNQ Framework Overview based on (Xilinx, 2020, n.p.) . . . 33
9 MQTT Setup Overview . . . 34
10 On the left PYNQ-Z1 bloc diagram and on the right Logitech Webcam 720p HD . . . 35
11 Sequence Diagram . . . 36
12 Class Diagram Concept People Detection . . . 37
13 Tinier YOLO CNN Topology (*Xilinx/QNN-MO-PYNQ*, 2020, n.p.) . . . 38
14 Class Diagram Concept Streamlit App . . . 39
15 Interface Streamlit application . . . 40
16 Overall detection rate . . . 44
17 Comparison person size (left full image, right cropped) . . . 44
18 False detection examples . . . 45
19 No detection examples . . . 45
20 Energy consumption comparison . . . 46
21 Transferrate comparison edge vs. cloud (logarithmic scale) . . . 47
22 Image Processing Speed Comparison Hardware vs. Software . . . 48
23 SWOT Diagram based on (Xhienne, 2007, n.p.) . . . 50
24 Detailed Project Planning . . . 65
25 Literature Research Keyword Radar . . . 66
26 Class Diagram People Detection . . . 68
27 Class Diagram Streamlit App . . .

List of Tables

1 Search Strategy 16
2 Comparison IoT protocols 28
3 Development Boards Comparison 30
4 MQTT Topics 41
5 Use case evaluation 66

Acronyms

AADL Architecture Analysis Design Language. 20

AI Artificial Intelligence. 1, 17

AMQP Advanced Messe Queuing Protocl. 28

AP Average Precision. 20

AR Augmented Reality. 18

AWS Amazon Web Service. 1

BNN Binary Neural Network. 20, 31, 38

C2C Cloud-to-Cloud. 28

CDN Content Delivery Network. 17

CNN Convolutional Neural Network. 2, 19–21, 31, 38, 45, 48, 49

CoAP Constrained Application Protocol. 28

COCO Common Objects in Context. 21, 45

CPU Central Processing Unit. 2, 13, 18, 26, 27, 30–34, 36, 52

csv Comma Separated Values. 37

CV Computer Vision. 19

D2C Device-to-Cloud. 28

D2D Device-to-Device. 28

DCNN Deconvolutional Neural Network. 18

DMA Direct Memory Access. 33

DSP Digital Signal Processing. 30

EECC European Edge Computing Consortium. 17

EPU Edge Processing Unit. 9, 11–13, 15, 33–35, 39–43, 45–47, 50–53

ETSI European Telecommunications Standards Institue. 17

FIFO First in first out. 19

FPGA Field Programmable Gate Array. 1, 2, 5, 13, 14, 18, 19, 21, 26, 27, 30–33, 35, 38, 46–49, 51, 52

fps frame per second. 32, 47

GPIO General Purpose Input Output. 34

GPU Graphical Processing Unit. 18, 30

GUI Graphical User Interface. 34

HDMI High-Definition Multimedia Interface. 31, 53

IIC Industrial Internet Consortium. 17

IoT Internet of Things. 26–28

IP Intellectual Property. 32, 33

IP Internet Protocol. 28

ISE Integrated Software Development. 31

JSON JavaScript Object Notation. 39

LiDaR Light imaging, detection, and ranging. 2

M2M Machine to Machine Communication. 13, 26, 27

ML Machine Learning. 2, 19–21, 28, 31, 35, 38, 43, 44, 48, 50, 52, 53

MQTT Messaging Queuing Telemetry Transport. 28, 31, 34–42, 47, 49, 52

NVME Non-Volatile Memory Express. 48, 49

OEC Open Edge Computing. 17

OS Operating System. 27, 32, 49

P2P Point-to-Point. 28

PCB Printed Circuit Board. 53

PIL Python image library. 40

PL Programmable Logic. 26–28, 31–35

PLC Programmable Logic Controller. 26, 34, 45, 53

POC Proof of concept. 5, 7, 9, 41

PS Processing System. 26, 31, 34, 35

PYNQ Python for Zynq. 19, 20, 31, 32, 34, 38, 45, 47–49, 51

QNN Quantized Neural Network. 20, 31, 38, 53

RAM Random Access Memory. 30, 31, 45, 46, 48, 52, 53

RAMEC 4.0 Reference Architecture Model Edge Computing 4.0. 17

RAMI 4.0 Reference Architecture Model Industrie 4.0. 17

REST Request-Reply. 28

ROI Return of Investement. 10

RTOS Real Time Operating Systems. 18

SDK Software Development Kit. 32, 51

SGAM Smart Grid Architecture Model. 17

SMART Specific, Measurable, Achievable, Relevant, Time-bound. 4, 7

SoC System on Chip. 13, 19, 20, 27, 28, 30, 31

SPoF Single Point of Failure. 28

SWOT Strength Weaknesses Opportunities and Threats. 12, 50

TCP Transport Communication Protocol. 28

TLS Transport Layer Security. 28

TPU Tensor Processing Unit. 18

UDP User Datagram Protocol. 28

USB Universal Serial Bus. 32, 34, 52, 53

VHDL Very High Speed Integrated Circuit Hardware. 30

VPN Virtual Private Network. 28

XRT Xilinx Runtime. 32

YOLO You only look once. 13, 20, 21, 38, 43–45, 49, 52, 53

Glossary

ARM Arm is the world's leading semiconductor IP company. They develop and license IP that is at the heart of billions of devices worldwide, and provide development tools that accelerate product development, from sensors to smartphones to servers. 31–33, 36, 46, 51

Cloud Computing Cloud computing is an IT infrastructure delivering computing as a service. This can include networking, servers, memory, computing functionalities such as analytics. Typically only services used have to be payed. 1

Edge Computing In edge computing, computer applications, data and services are moved away from central nodes (data centers) to the outer edges of a network. In other words, the aim is to process data streams in a resource-saving manner, at least partially on site (e.g. directly at the end device or within a factory), while still benefiting from the advantages of the cloud. 1, 4, 7

hardware accelerator Hardware acceleration describes the relief of the main processor by delegation of special computationally intensive tasks to hardware specialized for these tasks. This technique is used in particular for graphic display in computers. Hardware accelerators can be either speciallized asic chips or reconfigurable FPGA chips. 1, 5, 36

Linux Linux is a family of open-source Unix-like operating systems based on the Linux kernel, an operating system kernel first released on September 17, 1991, by Linus Torvalds. Linux is typically packaged in a Linux distribution. 27, 32, 33

OpenCV OpenCV (Open Source Computer Vision Library: http://opencv.org) is an open-source BSD-licensed library that includes several hundreds of computer vision algorithms (OpenCV 2020). 37, 40

rich Rich is a Python library for rich text and beautiful formatting in the terminal. The Rich API makes it easy to add color and style to terminal output. Rich can also render pretty tables, progress bars, markdown, syntax highlighted source code, tracebacks, and more — out of the box (Rich 2020). 37, 40

Scrum Scrum is an agile process framework for managing complex knowledge work, with an initial emphasis on software development, although it has been used in other fields and is slowly starting to be explored for other complex work, research and advanced technologies.(Ken & Jeff, 2020). 7, 9, 10

x86 x86 is a family of instruction set architectures initially developed by Intel based on the Intel 8086 microprocessor and its 8088 variant. 31, 36

Appendix A Project Planning

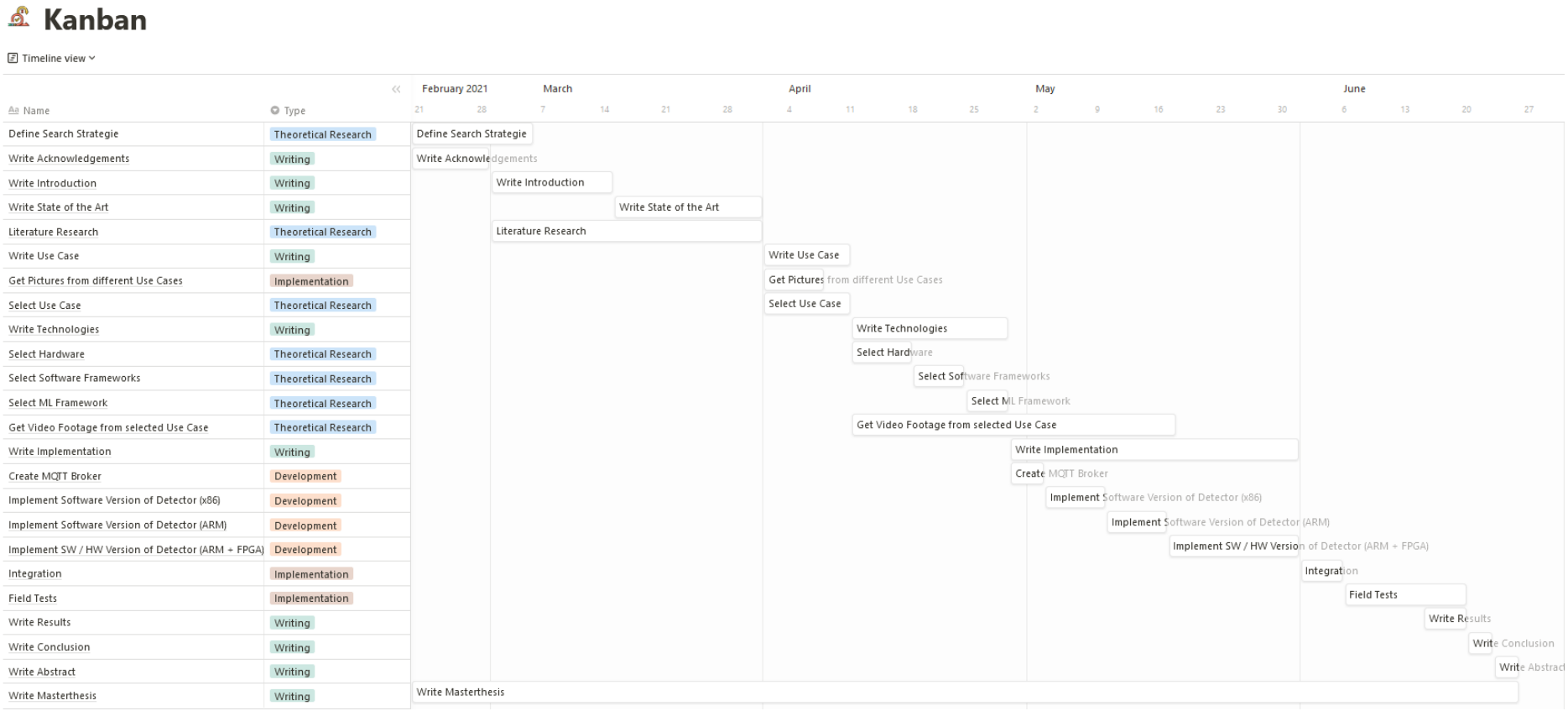


Figure 24: Detailed Project Planning

Appendix B Literature Research

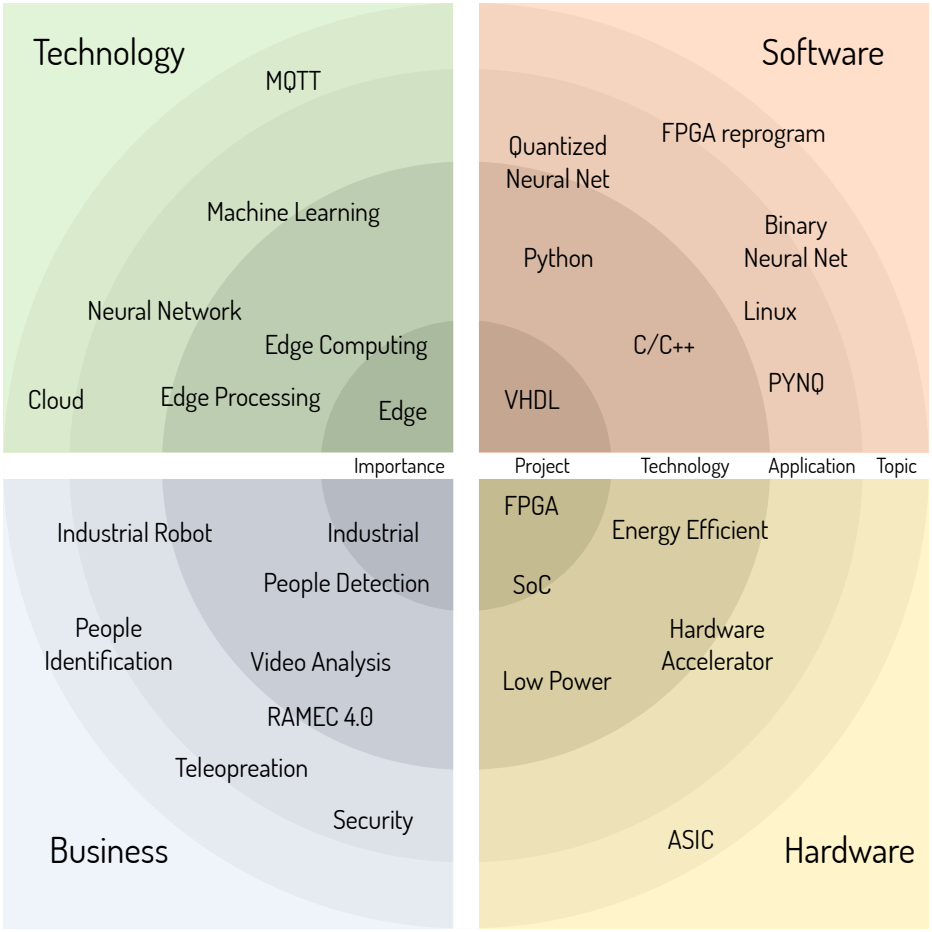


Figure 25: Literature Research Keyword Radar

Appendix C Use case evaluation

Name	*Location*	*Detection*	*Importance*	*Environment*	*Installation*
BAUWA AG Storage Hall	Steg (VS)				
BAUWA AG Crane Container	Steg (VS)				
Theler AG Cement Factory	Naters (VS)				

Table 5: Use case evaluation

Appendix D Program configuration

```
[APP]
; development mode yes = 1 or no = 0
```

```
dev_mode = 1
; in seconds
capture_interval = 5
; camera interface if multiple existing starting with 0
camera = 0
; DEBUG < INFO < WARNING < ERROR < CRITICAL
log_level = DEBUG
save_images = 1
show_images = 0
; use hardware accelerated yolo algorithm
yolo_hw = 1
; identifier for camera location
location = crane_1

[PYTHON]
python_path = /usr/local/lib/python3.6

[DARKNET_HW]
darknet_path = /opt/darknet
darknet_path_python = /opt/darknet/python/

[DARKNET_SW]
darknet_path = ./darknet
darknet_cfg_path = ./darknet/cfg/yolov3-tiny.cfg
darknet_weights_path = ./darknet/weights/yolov3-tiny.weights

[PATH]
log_path = ./../../output/log
timelog_path = ./../../output/timelogs
output_path = ./../../output
raw_image_path = ./../../output/webcam
detection_image_path = ./../../output/detection
extension = .jpg

[MQTT]
address = localhost
port = 1883
keepalive = 60
username = pynq
password = g86TpdguWjpQICvwVHhx
topic_detection = yolo/detection
topic_lastwill = yolo/status
topic_control_state = yolo/state
```

Listing 4: Configuration of the implementation

Appendix E Program structure

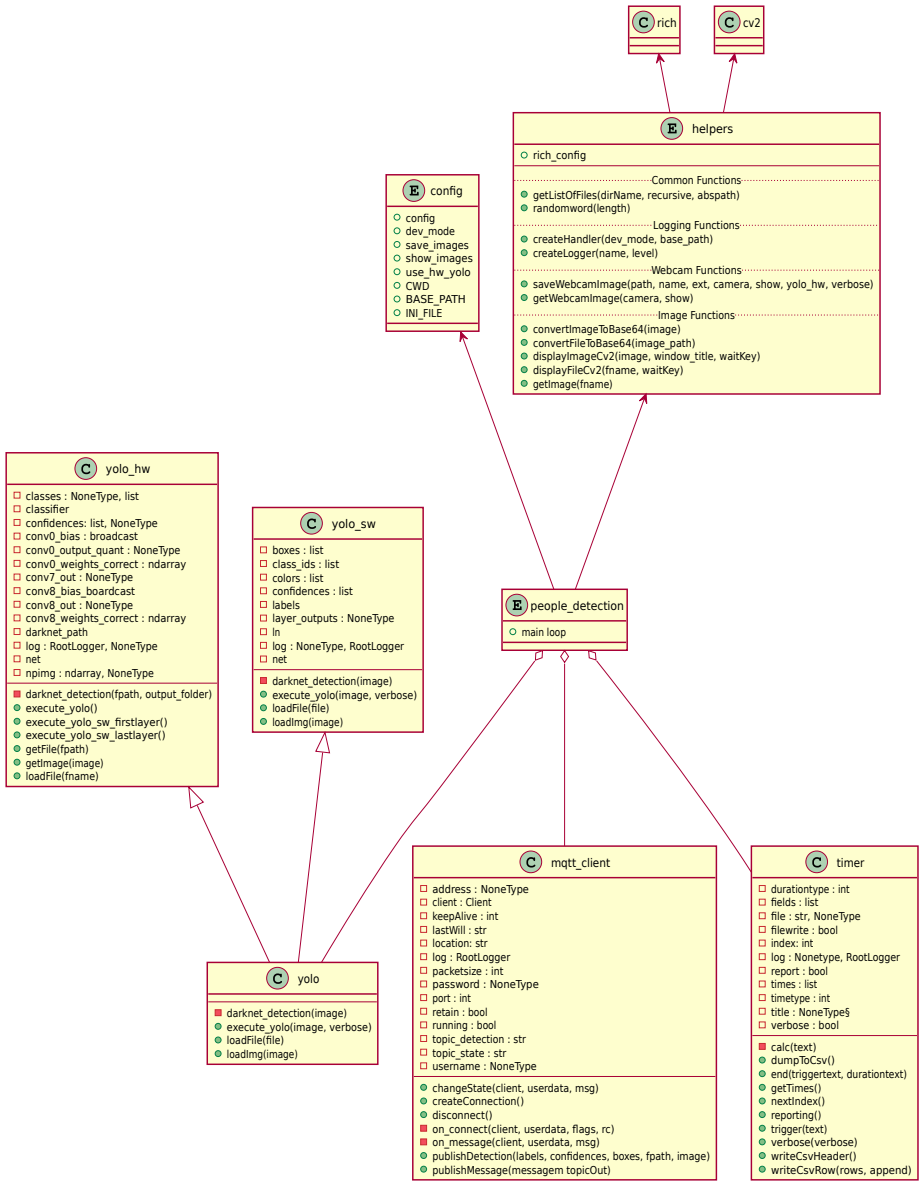


Figure 26: Class Diagram People Detection

E Program structure

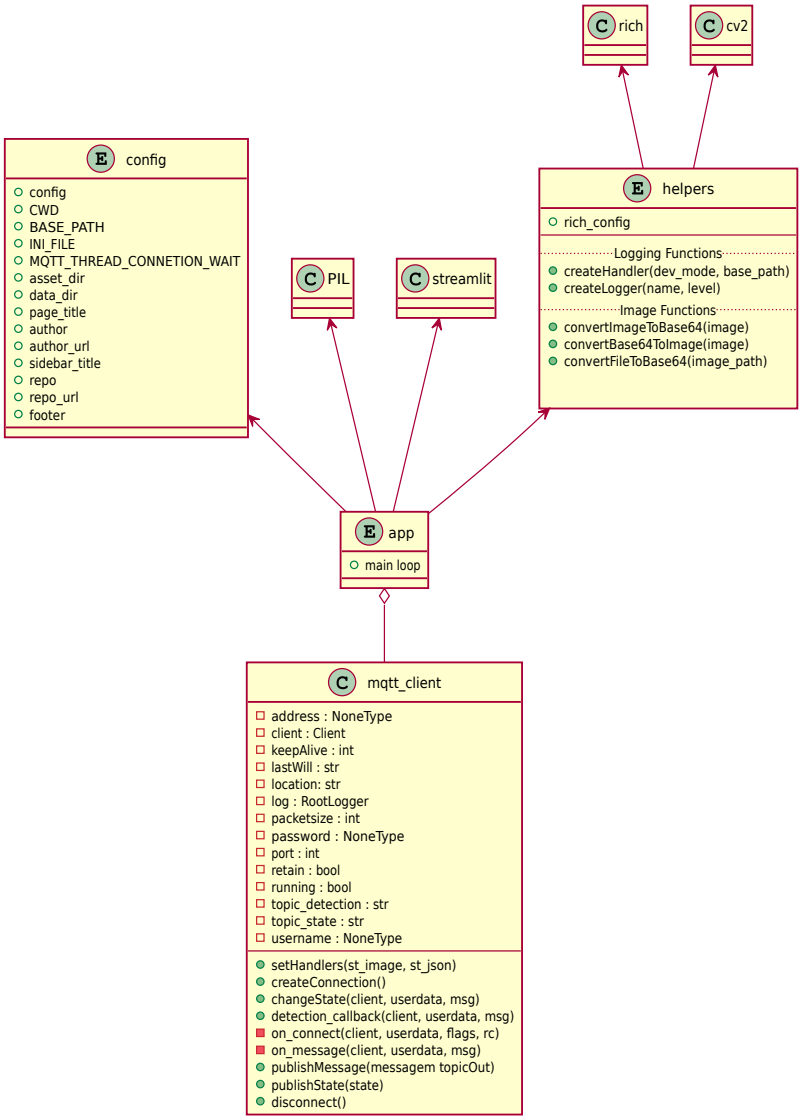


Figure 27: Class Diagram Streamlit App

Selbständigkeitserklärung

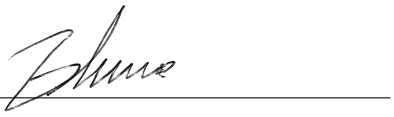
Hiermit bestätige ich, dass ich die vorliegende Thesis mit dem Titel

“Edge Processing Unit - The Way to efficient Edge Computing for Industry 4.0”

selbständig verfasst und keine anderen als die angegebenen Quellen benutzt habe. Alle Stellen, die wörtlich oder sinngemäss aus Quellen entnommen worden sind, habe ich als solche kenntlich gemacht.

Des weiteren versichere ich, dass ich bisher noch keine wissenschaftliche Arbeit mit gleichem oder ähnlichem Inhalt an der Fernfachhochschule Schweiz oder an einer anderen Hochschule eingereicht habe.

Mir ist bekannt, dass die Fernfachhochschule Schweiz andernfalls auch nachträglich berechtigt ist, mir den auf Grund dieser Arbeit verliehenen Titel zu entziehen.

Visp, June 12, 2021, 

Ort, Datum, Unterschrift