NTNU
Norwegian University of
Science and Technology

Daniel René Løken

2022

Master's thesis

NTNU
Norwegian University of
Science and Technology
Faculty of Engineering
Department of Mechanical and Industrial Engineering

Master's thesis

Daniel René Løken

Facilitating for Combining Industry 4.0 and Digital Twins with 5G Connectivity to Integrate Robots in Digital Production Factories

June 2022

NTNU
Norwegian University of
Science and Technology

NTNU

Norwegian University of Science and Technology

DEPARTMENT OF MECHANICAL AND INDUSTRIAL ENGINEERING

Robotics and automation, master's degree

Facilitating for combining Industry 4.0 and Digital Twins with 5G Connectivity to Integrate Robots in Digital Production Factories

Author:

Daniel René Løken

Jun, 2022

Preface

This report is the final product of my master's degree within Engineering & ICT at the Norwegian University of Science and Technology in Trondheim. The thesis is conducted within the course "TMM4935 - Robotics and Automation, Master's Thesis" at the Department of Mechanical and Industrial Engineering.

I would like to thank my supervisor Amund Skavhaug for his mentoring during this project. His professional competence and focus on student well-being has contributed to an enjoyable semester. I would also like to thank Diaa Jadaan for technical guidance and support.

Daniel René Løken
Trondheim, 2022-06-27

Abstract

This master's thesis facilitates the development of a digital twin in the edge cloud at NTNU Gløshaugen's Industri 4.0 laboratory. This is done in connection with a "use case" proposed by the 5G-SOLUTIONS project which deals with the further development of digital factories with both advanced communication solutions and real-time remote control. An AAS has been re-engineered to communicate over 5G with the exception of a direct Ethernet connection between Raspberry Pi and KMR iiwa. Functionality has been added to also be able to collect sensor data from the robot. The communication protocols used are OPC UA and ROS2. The components of the system are discussed in light of Industry 4.0 standards and its design principals. The system has been developed to easily be transferred to edge cloud when it is delivered at the lab.

Sammendrag

Denne masteroppgaven tilretteleger for å utvikle en digital tvilling i edge cloud ved NTNU Gløshaugen's Industri 4.0 laboratorium. Dette blir gjort i sammenheng med et "use case" som er foreslått av 5G-SOLUTIONS-prosjektet som omhandler å videreutvikle digitale fabrikker med både avanserte kommunikasjonsløsninger og sanntidsfjernstyring. Et AAS er ombygget til å komunisere over 5G utenom en direkte ethernet forbindelse mellom Raspberry Pi og KMR iiwa. Det er også lagt til funksjonalitet for å kunne hente inn sensordata. Kommunikasjonsprotokollene som er brukt er OPC UA og ROS2. Komponentene i systemet blir diskutert i lys av Industri 4.0 standarer. Systemet er utviklet med tanke på å kunne bli overført til edge cloud når det blir levert på labben og videre ekspandert med flere enheter.

Systemet er utviklet på den måten at det lett kan bli overført til edge cloud, når dette blir levert på labben

Acronyms

3G Third Generation of Telecommunication. 31

4G Fourth Generation of Telecommunication. 10, 30, 31, 61

5G Fifth Generation of Telecommunication. 6, 10, 12, 16, 21, 30, 31, 35, 37, 40, 41, 56, 61, 63–65

5G-VINNI 5G Vertical INNpcation Infrastructure. 10

AAS Asset Administration Shell. 6, 12, 18–20, 32, 35, 37, 38, 43, 46, 53, 55, 56, 58, 61–63

ACK Acknowledgement. 15

AGV Automated guided vehicle. 65

AI Artificial Intelligence. 60

API Application Programming Interface. 6, 33, 34, 47

APN Access Point Name. 42

AR Augmented Reality. 16

AT-command Attention-command. 31, 41, 42

CBND CloudBand Network Director. 58

CDSO Cross Domain Service Orchestrator. 58

CLA Closed-Loop Automation. 58

CPS Cyber Physical Systems. 19

DDS Data Distribution Service. 21, 22, 32, 53, 62, 65

DDSI Data Distribution Service Interoperability. 32

DNS Domain Name System. 35

DT Digital Twin. 10, 12, 17, 18, 61–65

eMBB Enhanced Mobile Broad-Band. 16, 61

FDI Fast Data Interface. 49, 50

GNSS Global Navigation Satellite System. 30

HTTP Hypertext Transfer Protocol. 30

IDL Interactive Data Language. 32

iiwa Intelligent Industrial Work Assistant. 12, 24, 26–29, 32, 35, 43–47, 51, 62, 63, 65

IoE Internet of Everything. 59, 60

IoT Internet of Things. 16, 17, 21, 22, 60

IP Internet Protocol. 14, 15, 30, 35, 37, 38, 42, 44, 49, 61

ISO International Standards Organization. 14

ITUR International Telecommunication Union Radiocommunication Bureau. 16

JDK Java Development Kit. 29

KMP KUKA Mobile Platform. 25, 50

KMR KUKA Mobile Robot. 12, 24, 26–29, 32, 35, 43–47, 51, 62, 63, 65

LBR Leichtbauroboter(German for lightweight robot). 24, 26

LTE Long Term Evolution. 30

LTS Long-term Support. 36

ML Machine Learning. 60

mMTC Massive Machine-Type Communications. 16, 61

NR New Radio. 30

nvm Node Version Manager. 36

OMG Object Management Group. 21, 22

OPC Open Platform Comunication. 19–21, 38, 43, 53, 54, 56, 61, 62

OS Operating System. 36, 47

OSI Open System Interconnections. 6, 14, 15

PDM Product Data Management. 19

PIN Personal Identification Number. 31, 41

PubSub Publish-Subscribe. 21

RCC Robot Control Core. 33

ROS Robot Operating System. 31, 32, 51, 53

ROS2 Robot Operation System 2. 32, 42, 43, 51, 53, 54, 62

RTPS Real-Time Publish-Subscribe. 22, 32

SD Secure Digital. 36

TCP Transmission Control Protocol. 14, 15, 30, 47–49, 51

TSN Time Sensitive Networking. 21

UA Unified Architecture. 19–21, 38, 43, 53, 54, 56, 61, 62

URLLC Ultra Reliable Low Latency Communications. 16, 61

VNF Virtual Network Functions. 58

VR Virtual Reality. 16

List of Figures

Contents

1 Introduction

1.1 Motivation and Problem Description

Through three industrial revolutions, society has gone from small scale handcrafting to large scale fabrics with automation and digitization of independent processes. Industry 4.0 refers to a fourth industrial revolution where the independent processes communicate with each other to achieve full automation from order to delivery. With these new requirement for communication, Fifth Generation of Telecommunication (5G) emerges as promising alternative with a high bandwidth, low response time and support for a high number of connected devices. To administer the factories and processes with the large amounts of data associated with it, Digital Twin (DT)s can be used. This thesis explores the possibility of integrating a DT that communicates solely with 5G and comply with the industrial driving force of Industry 4.0. The objectives are presented in subsection 1.3.

1.2 Previous Work

On 1st June 2019 the 5G-SOLUTIONS project initiated by EU commenced. The aim of the project is to prove and validate that 5G capabilities provide prominent industry verticals with ubiquitous access to a wide range of forward-looking services with orders of magnitude of improvement over Fourth Generation of Telecommunication (4G), thus bringing the 5G vision closer to realisation. To achieve this there will be conducted 20 advanced field trails of 20 innovative use cases facilitated in Italy, Norway, Greece, Ireland and Belgium over five significant industry vertical domains. A map of the different project locations and their related groups is shown in Figure 1, with the blue node representing the 5G Vertical INNpcation Infrastructure (5G-VINNI). 5G-VINNI is another project financed by the EU, which started in 2018 and works on establishing an "end-to-end facility that validates the performance of new 5G technologies by operating trials of advanced vertical sector services". The domain has five main use cases, one of the which is "LL1:Factories of the Future"[1]. This master's thesis mainly contributes to Use case 1.3 Remotely controlling digital factories.[2].

1.2.1 UC 1.3: Remotely Controlling Digital Factories[1]

The simplest setup of this use case involves remote control applications running on tablets or smartphones, for example. However, given the trend of new AR devices, it is likely that

[1]This section is taken from the description of the UC 1.3 by the 5G Solutions project.

new remote services may arise that facilitate the creation of virtual back office teams. Such remote teams may use the data coming from smart devices for preventive analytics and easy access to work instructions, whereby, e.g., they would be able to view the camera or iPad/Google Glass of a local worker. Additionally, the application of AR in the plant will facilitate:

- Augmented-reality support in production and assembly: Precisely positioned picture-in picture fade-ins, showing the operator the next step and helping to avoid misplacement and unnecessary scrap.
- Augmented-reality support in maintenance and repair: Repair machines without training due to augmented information and operational guidance.

Cross-functional communication, effective knowledge sharing and collaborative design platforms will be facilitated by solutions for communities of practice. In this use case family, there is a less stringent need for low-latency. Interaction times up to seconds are acceptable for remote servicing machines. However, high availability is key for allowing (emergency) maintenance actions to occur immediately. Bandwidth is important for video-controlled maintenance, with real-time augmented content mixed into the video signal. Moreover, latency is particularly important for real-time,

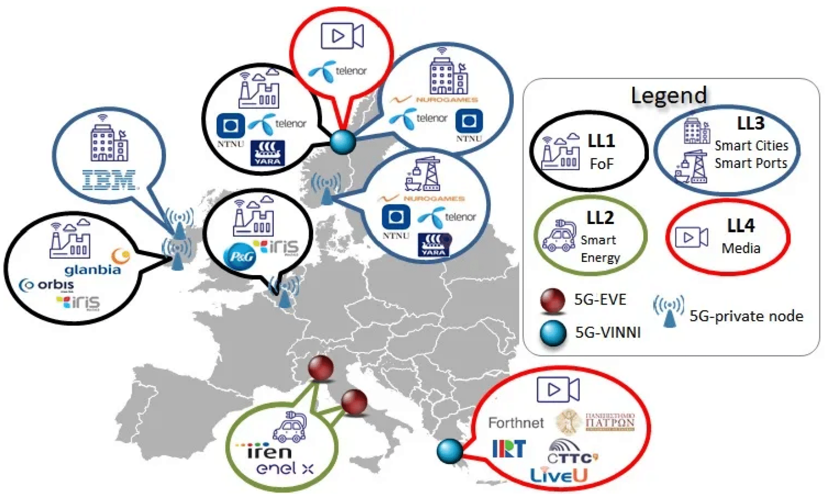


Figure 1: Map of the 5G-SOLUTIONS project with related groups and corporations [2].

1.3 Objectives[2]

The initial objective was to study the feasibility of implementing a digital twin as an edge service in 5G. Due to delivery problems of the system, the objective was retargeted to deliver positional data from the KUKA Mobile Robot (KMR) Intelligent Industrial Work Assistant (iiwa) as a high level service, so that users or developers of a DT can take advantage of the location data without knowledge of the low-level implementation. The service has to be loosely coupled and comply to the design principals of industry 4.0.
It is promised that an edge cloud service will be delivered to Gløshaugen's Industry 4.0 laboratory by Telenor in the near future. The system must therefore be developed with the possibility of being transferred to the edge cloud with least possible changes. The feasibility of moving the system to an edge cloud should be conducted. In order to facilitate for further development, there should be sufficient documentation of the system and its setup.

1.4 Contributions

The contribution of this thesis to society is the expansion of an Asset Administration Shell (AAS) to also delivering high level positional data as a service that can be used in the creation of an industrial DT. The loosely coupled system for retrieving sensor data over 5G has the possibility to advance the evolution of Industry 4.0 by facilitating for faster deployment of a DT.

1.5 Limitations

Throughout the work, the delivery of a private 5G network has been repeatedly delayed. Telenor was supposed to install 5G nodes providing a private 5G network in NTNU Gløshaugen's Industry 4.0 laboratory. At the start of the project, it was expected that this would be delivered in time to expand the system with new entities and conduct performance experiments on it. Due to repeated delays from Telenor, this has not been possible. The 5G nodes have been delivered, but Telenor have not managed to make them operational.

1.6 Outline

The report is structured in six sections. section 2 goes through the most important concepts, hardware and software that are relevant for the research of this thesis. This is important to get an understanding of the reasoning behind the design. In section 3 a thorough review of

[2]The installation by Telenor, Ericson and several contractors started summer of 2020, but the first partly working system was up and running as theses was delivered 24-06.2022

the system and how it is built is conducted, so that it can be recreated more quickly and further work on the system can be more accessible. section 4 describes the system with technical details of the system architecture as well as alternative designs. It is intended for those who seek a deeper understanding of the system and want to recreate it. section 5 discusses the system , technologies and future work. Finally the thesis concludes in section 6

2 Background Theory- Concept and Technologies [3]

This chapter goes through the most important concepts, hardware and software that are relevant for the research of this project. This is important to get an understanding of the reasoning behind the design. subsection 2.1 presents the different communication protocols, defines both an Digital Twin and AAS, and explain the newest generation of cellular networks, the 5G. subsection 2.2 introduces the KMR iiwa robot as well as the Raspberry Pi Model 4B with the 5G HAT. subsection 2.3 describes ROS 2 and all the software needed to start a KUKA project.

2.1 Concepts

2.1.1 TCP/IP

Transmission Control Protocol (TCP)/Internet Protocol (IP) is the protocol suite which the worldwide collection of interconnected networks, referred to as the Internet(uppercase "I") is built upon[4]. The protocol suite met the need for worldwide data communication at the right time and had four important features that allowed them to meet this need.
The first one was open protocol standards. The protocols where developed independently from any specific computer hardware or operating system and made freely available. The wide support makes it ideal for merging different hardware and software components, regardless of whether they communicate over the Internet or not[4]. The second feature was independence from specific physical network hardware, which allows TCP/IP to incorporate numerous kinds of networks. This includes Ethernet, fiber optic[5], copper wire[5], and virtually any other kind of physical transmission medium[4].The third was that it uses a common addressing scheme that uniquely address any other device in the entire network. This also applies for a network as large as the Internet[4]. The last one was the use of standardized high-level protocols that manages to deliver consistent, widely available user services[4].
To describe the structure and functions of data communications protocols the Open System Interconnections (OSI) Referens Model developed by International Standards Organization (ISO) is frequently used[4]. The OSI Referens Model contains seven layers as shown in Figure 2. TCP can be used to handle the *Transport Layer* and IP can be used for the *Network Layer*[6].

[3]This chapter mainly retrieved from the author‘s specialization project [3]

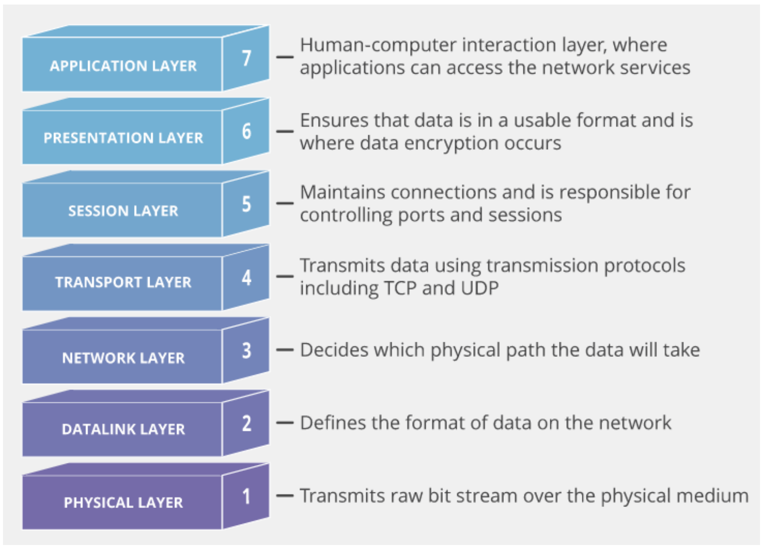

APPLICATION LAYER
7
Human-computer interaction layer, where applications can access the network services
PRESENTATION LAYER
6
Ensures that data is in a usable format and is where data encryption occurs
SESSION LAYER
5
Maintains connections and is responsible for controlling ports and sessions
TRANSPORT LAYER
4
Transmits data using transmission protocols including TCP and UDP
NETWORK LAYER
3
Decides which physical path the data will take
DATALINK LAYER
2
Defines the format of data on the network
PHYSICAL LAYER
1
Transmits raw bit stream over the physical medium

Figure 2: The OSI Model[7].

Transport Layer The transport layer delivers end-to-end communication between two host in a network. TCP is a robust, reliable and time-tested protocol. The robustness means that it is adaptable to different networks. It supports reliable delivery regardless of the underlying network. To ensure reliable delivery, the sender maintain a buffer, called a sliding window, of data that has been sent to the receiver. A receiver acknowledges received data by sending Acknowledgement (ACK) packets in return. When the sender receives an ACK packet, the sliding window can be deleted well knowing that the data has been transmitted successfully to the receiver. It also keeps track of the buffer size to ensure that it is not overfilled[8].

Network Layer IP addresses are 32-bit numbers normally expressed by four decimal numbers separated by period, e.g 123.123.123.123(in bits 01111011.01111011.01111011.01111011). This includes both the network address and the host address. The format of these parts can vary between IP addresses. A prefix length of the address determines the number of bits used to identify the parts. The prefixed length is sett to the same length as the *subnet mask*. The subnet mask is used to split a network into smaller sub-networks. This is impotent to reduce the amount of host devices on the network. The mask consist of bites that can be 255 (on) or 0 (off). On bits belongs to the network part, while off bits belong to the host part. An example with a subnet mask 255.255.255.0 and an IP address 123.123.123.123 would have 123.123.123.0 as the network part and 0.0.0.123 as the host[9].

2.1.2 5G Networks

The first four generations of mobile communication technology are only focused on mobile communication. The 5G also take in to account factors like large-scale connectivity, ultra low latency required by Internet of Things (IoT) and other future needs. Three main applications for 5G have been defined by the International Telecommunication Union Radiocommunication Bureau (ITUR), which are Enhanced Mobile Broad-Band (eMBB), Ultra Reliable Low Latency Communications (URLLC) and Massive Machine-Type Communications (mMTC). eMBB anables high-bandwidth applications like Virtual Reality (VR), Augmented Reality (AR) and 4k video streaming; mMTC can be used for connecting smart devices and IoT; URLLC is manly focused on latency sensitive operations such as autonomous vehicles. 5G is increasingly recognized as important as infrastructure and will play a significant role in Industry 4.0.[10].

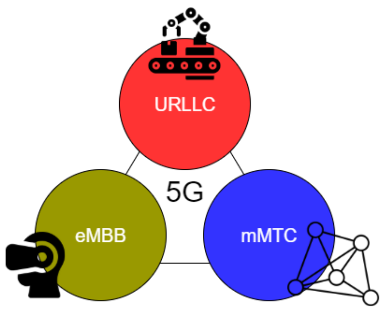


Figure 3: 5G service types[9].

"EU releases 5G spectrum strategy and strives to seize 5G deployment opportunities On November 10, 2016, the European Commission's Radio Spectrum Policy Group (RSPG) released the European 5G spectrum strategy, which clearly stated that the 3400–3800 MHz band will be the main frequency band for 5G deployment in Europe before 2020 and 700–1000 MHz will be used for 5G wide coverage. In terms of millimeter-wave bands, it is clear that the 26 GHz (24.25–27.5GHz) band will be used as the initial deployment for 5G high-frequency bands in Europe. Some of them will be used to meet the 5G market demand by 2020. In addition, the European Union will continue to study the 32 GHz (31.8–33.4 GHz), 40 GHz (40.5–43.5GHz), and other high-frequency bands."-[11]

2.1.3 Edge Computing

To make 5G capable of facilitating URLLC and mMTC the networks are more decentralized and more of the computing is done closer to the connected devises in the edge. Edge computing often refers to an open platform that integrates core capabilities of networks, computing, storage and applications on the edge of networks near the source of object or data. By bringing data closer to end-user devices it is possible to provide extremely low-latency computing. To reduce network bandwidth occupation, redundant data can be processes and stored on the edge. With the introduction of IoT and smart devices the amount of data gathered will most likely increase and the use of edge computing is therefore a key component in reducing the pressure on data center connections[12].

2.1.4 Digital Twin

A DT is a real time digital representation of a physical system, which can be everything from a car to a production site. In a DT it is possible to store historical data as well as real-time data. This is useful for monitoring and controlling systems from remote locations as well as doing real-time calculations. Based on the available data in the DT, systems can be optimized and streamlined through methods such as big data analytics and AI[13]. Today there exists many usecases of DT, one of which is in the oil and gas section. Equinor has a DT of Johan Sverdrup which is used as a visualizing tool of the platform and can by accessed from the cloud through computers, smartphones, tablets and Microsofts HoloLens[14]. The types and usecases of DT can be very broad and it therefore can be useful to classify them in different maturity levels from 0 to 5.

Level 0 The most fundamental level of a DT. In this level a physical system does not need to exist, but can be used to get an overview of the system. The twin can show static data that describe the system, such as voltage in a wire or the thickness of a pipe[13].

Level 1 At this level the DT is linked to a physical system which it retrieves data from. It should be capable of representing the current status of the system. This is done trough alarms and occurrences with the ability to give information about past system status and historic. An example of this is showing the flow of the various pipes in real-time and alarms that are triggered when values exceed preset conditions[13].

Level 2 Has the ability to present diagnostic information through conditional indicators. This is useful when troubleshooting and monitoring. If for example an abnormal flow in one of the pipes occurs the twin will be able to show this deviation based on historical data[13].

Level 3 A DT at level 3 is starting to get quit advanced as it can predict future status or performance of the system. This is done through various models that are combined with the twin. The diagnostic information can now also alert on future status of the system. For example, the twin will be able to predict which pipes are exposed to the greatest wear and how long it takes until they have to be replaced[13].

Level 4 With a DT at level 4 it is able to recommend measures to counteract implied system implications and improve system performance. It will be able to evaluate the various possibilities and insure that the measures does not reduce performance on other parts of the system. An example of this is to suggest reduced flow through a pipe to avoid wear, while other pipes increase flow so that the overall flow of the system is maintained while wear is minimized[13].

Level 5 This is the highest level of a DT. The twin is now able to make decisions without the presence of a user. Human presence is only necessary to to control that it works as desired. It is able to make decisions on its own and control the system autonomously to ensure optimal performance from the system. In the pipe example, the DT will control the entire pipe system of a city, reduce the capacity of pipes with leaks and increase flow in other pipes, without increasing wear and the need for maintenance. It will also be able to order repair of broken pipes[13].

2.1.5 AAS

AAS is the industrial implementation of a DT for Industry 4.0[15]. It describes an asset digitally and exchange asset-related data among industrial assets as well as production orchestration systems and people. AAS is considered a key concept of Industry 4.0 [16]. Instead of reading individual data from a sensor or an actuator, the AAS in Industry 4.0 combines different entities and represent them as a combined asset[9]. An overview of a proposed integration of an AAS's with a overall system is shown in Figure 4.

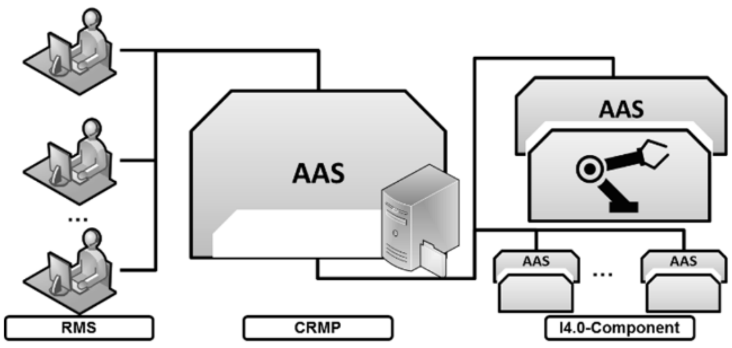


Figure 4: AAS overview[17].

There are none technical requirements for an AAS defined in the Plattform Industrie 4.0, but by subdividing it into segments, the industry 4.0 requirements can apply to each of them. The 5 segments shown in Figure 5 of an AAS in the production process and their industry 4.0 requirements are:

- **External Interface:** It administrates the data flow between local components and other interconnected systems. To assure interoperability it has to be highly standardized. Currently service-oriented paradigms are proposed for this communication[17].
- **Authentication and Security:** There exists a wide range of IT-security methods that can be implemented to satisfy industry 4.0 components required level of security. Each entity will have different levels of requirements[17].
- **Data Management:** This segment is divided into three modules as well as a mandatory main module manifesting the AAS. The other modules can store data about either an aspect of a component managed by the AAS, additional data about an application or they reference nested components. There can be multiple tiers of nested components containing data from every step of a production cycle. This can result in huge amounts of data for the top-tier AAS to be able to represent. It can therefor be beneficial to use a Product Data Management (PDM) software as the data management segment[17].
- **Functionality:** The different applications of the AAS is contained in this segment. By standardizing, it can be applied to industry 4.0 components. Adding required applications and data modules makes it possible to customize the AAS to any kind of asset[17].
- **Administration:** Manages the data flow between different segments.
- **Internal Interface:** Handles the communication between different entities and the AAS. To accommodate the vast differences in integration methods and Cyber Physical Systems (CPS) a standardization is undesirable[17].

2.1.6 OPC UA

Open Platform Comunication (OPC) Unified Architecture (UA) is a machine-to-machine communication protocol developed by the OPC Foundation. It was released in 2008 and is built as a platform independent service-oriented architecture that integrates all the functionality of the individual OPC Classic specifications into one extensible framework[18].

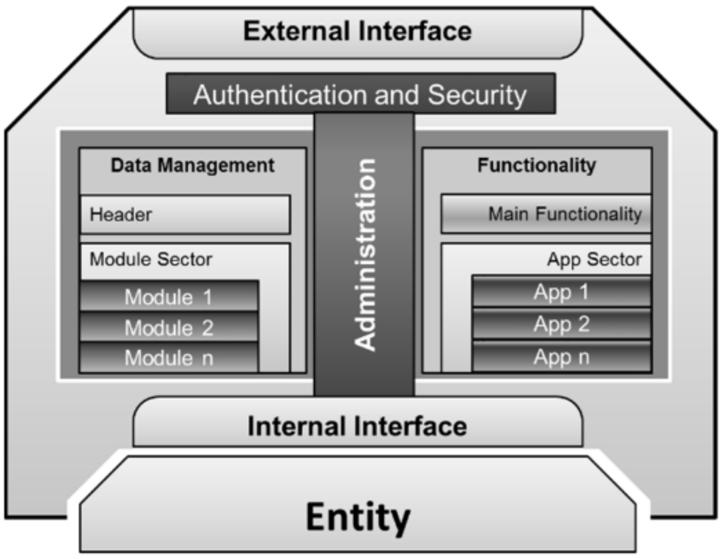


Figure 5: AAS segments[17].

OPC UA has become one of the key standards behind the Industry 4.0 initiative in Europe and more specifically Germany[19].
A key part of industry 4.0 is the connectivity and communication between different components in an industrial setting [20]. To facilitate the problems regarding interoperability between different equipments from a wide variety of manufactures it builds on standardized methods and abstractions. The framework uses a multi-layered architecture as shown in Figure 6, which enables the integration of innovative technologies and methodologies while maintaining backwards compatibility for existing products. This assures that UA products built today will work with future products [18]. The multi-layered approach satisfies OPC UA's original design goals of:

- **Functional equivalence:** all COM OPC Classic specifications are mapped to UA
- **Platform independence:** from an embedded micro-controller to cloud-based infrastructure
- **Secure:** encryption, authentication, and auditing
- **Extensible:** ability to add new features without affecting existing applications
- **Comprehensive information modeling:** for defining complex information

The OPC UA information modeling framework turns data into information. Extending and modeling complex multi-level structures is possible, through it's complete object-oriented capabilities. This is an essential ability for organizations to build their models upon existing

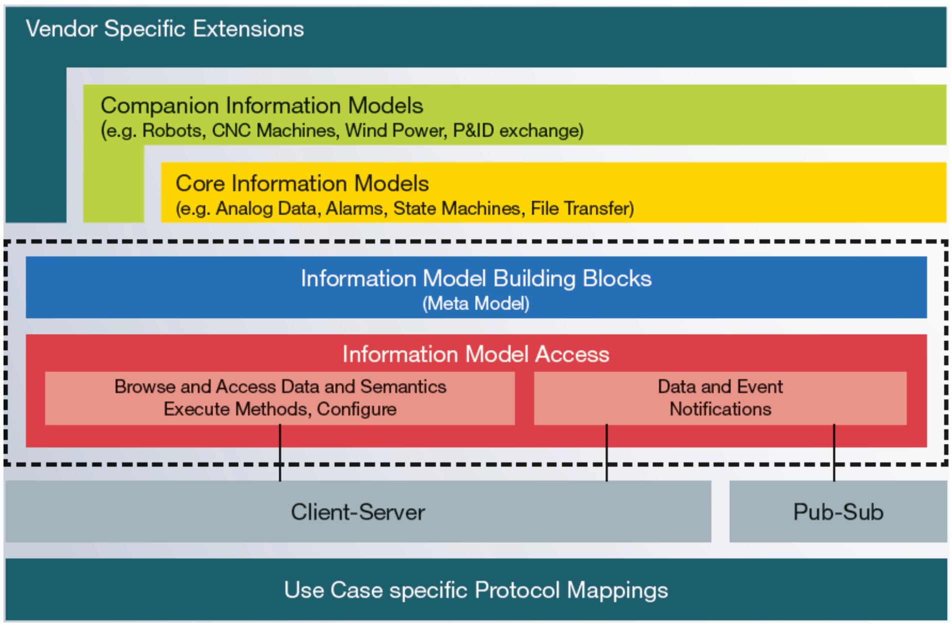

Vendor Specific Extensions
Companion Information Models
(e.g. Robots, CNC Machines, Wind Power, P&ID exchange)
Core Information Models
(e.g. Analog Data, Alarms, State Machines, File Transfer)
Information Model Building Blocks
(Meta Model)
Information Model Access
Browse and Access Data and Semantics
Execute Methods, Configure
Data and Event
Notifications
Client-Server
Pub-Sub
Use Case specific Protocol Mappings

Figure 6: Multi-layered OPC UA architecture[18].

core model to specific information with OPC UA [18]. With OPC UA it is possible to communicate trough the client-server pattern or Publish-Subscribe (PubSub) pattern [18]. With a client-server pattern clients send requests to the server. The server then processes the request and sends a response to the client with a result. A PubSub uses a one-way publisher and subscribers communication. The publisher publishes the message to a named resource called a topic, without any knowledge about subscribers. The subscribers receives massages by subscribing to a topic. The subscriber does not know when a message will be posted and potential publishers. With the integration of Time Sensitive Networking (TSN) into OPC UA it is possible to exchange information in real-time .
TSN is a new sett of standard which provides deterministic, reliable, high bandwidth, low-latency wired communication. For industry 4.0 and smart factories, the combination of 5g and TSN is seen on as an important feature to satisfy the demands of IoT[21]. The scope of this paper is the integration of 5G, but the complementary of TSN wired networks should be looked into further.

2.1.7 DDS

Data Distribution Service (DDS) is a middleware protocol and API standard for data-centric connectivity from the Object Management Group (OMG). DDS enables the integration of a great number of components in the same system, while providing low-latency data connectivity, extreme reliability, and a scalable architecture, which business and mission-critical IoT applications need [22]. The middleware simplifies the development of distributed systems, by handling a lot of the mechanics of passing information between application. With a data centric middleware the programmer only needs to specify how and when to share data and then directly share data values. This contrast to a message-centricity model, where the programmer needs to write the code that sends data messages. Real-Time Publish-Subscribe (RTPS) is standardized by OMG as the interoperability protocol for DDS. It was designed to meet the unique requirements of data-distributions systems targeted by DDS. It is now a "field proven technology that is currently deployed worldwide in thousands of industrial devices" [23]. It enables reliable, scalable publish & subscribe communication for real-time applications using standard IP networks. Connecting new applications is plug-and-play with automatic, configuration-less discovery allowing applications to join and leave networks at any time. [24].

2.1.8 Zenoh

Eclipse Zenoh was deisgned with Edge and Fog Computing in mind. It provides a stack that unifies the three states of data: in-motion, in-use and at rest. It supports traditional pub/sub with geo-distributed storages, queries and computations, while retaining a level of time and space efficiency that is well beyond any of the mainstream stacks [25].
Zenoh enables 3 deployment units: **peers**, **clients** and **routers**. With peer application it is possible to communicat with other peers over complete graph and connected graph and across the internet through Zenoh routers. Client is a user application with a Zenoh API that connects to a single Zenoh router or a single peer, to communicate with the rest of the system. To route the Zenoh protocol between clients and peers a software process called Zenoh router is used. The network topology is shown in Figure 7.

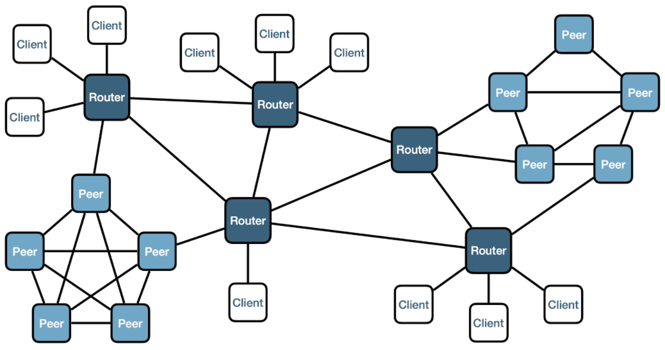


Figure 7: Zenoh topology[26].

There are two levels of user APIs. Zenoh-net is a network oriented API providing publisher & subscriber communication in addition to query & reply communication. This layer only focus on data transportation without any conserns about the storing nor content of the data. Zenoh-net primitives [26]:

- **write:** push live data to the matching subscribers.
- **subscribe:** subscriber to live data.
- **query:** query data from the matching queryables.
- **queryable:** an entity able to reply to queries.

Zenoh is a higher level API with the same properties as Zenoh-net API, but in a simpler and more data-centric oriented manner. It also provides all the building blocks to create a distributed storage. This layer is aware of the data content and is able to apply content-based filtering and transcoding. Zenoh primitives [26]:

- **put:** push live data to the matching subscribers and storages. (equivalent of Zenoh-net write)
- **subscribe:** subscriber to live data. (equivalent of Zenoh-net subscribe)
- **get:** get data from the matching storages and evals. (equivalent of Zenoh-net query)
- **storage:** the combination of a Zenoh-net subscriber to listen for live data to store and a Zenoh-net queryable to reply to matching get requests.
- **eval:** an entity able to reply to get requests. Typically used to provide data on demand or build a RPC system. (equivalent of Zenoh-net queryable)

An overview of Zenoh is shown in Figure 8

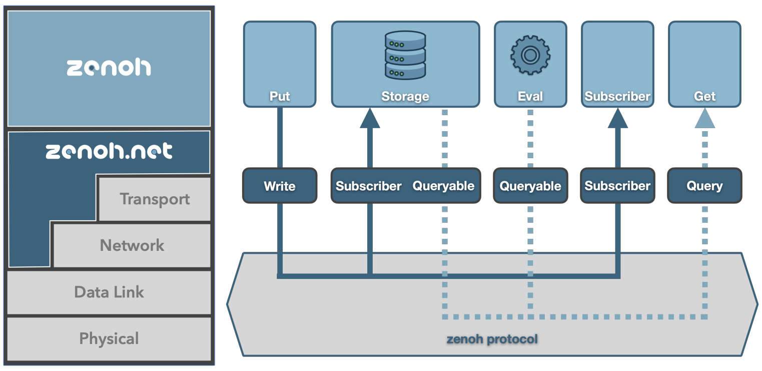


Figure 8: Zenoh overview[26].

2.1.9 Odometry

Odometry is the use of motion sensors to determine change in position relative to some known position. It can be calculated from any sensor capable of estimating change in position over time. The most widely utilized method of estimating the position of mobile robots is wheel odometry. By knowing the start position, radius of the wheels and their revolutions a position estimate can be made. This method provides good accuracy for short-term measurement but can lead to lot of error when the displacement increases. Sources to this error can be inaccuracies in wheel diameter, uneven floor and misalignment of wheels. It can therefor be wise to combine it with other position estimates like Lidar odometry or form beaming.

Lidar Odometry Lidar odometry uses laser scanners to estimate the motion of the robot, by finding a transformation between two scans measuring the same scene. A pipeline that converts the laser scanning to odometry data is shown in Figure 9. In the pipleine the input point clouds are first filtered by downsampeling, and normals are computed. The normals are used to recognize planes in the space. A transform library transforms the point cloud into the robot base frame so that its odomentry can be computed accordingly. To get points aligned in the point cloud map, nearby points or planes are considered. The transformation can then be estimated, but a prediction is needed, as the correspondences are unknown. This prediction can be based on odometry from other sensors or a motion model from previous point registrations. The estimated transformation can be used to calculate the position of the robot taking in to account the corresponding covariance. If there is a sufficient correspond between the matching module, the point cloud map can be updated. The the old point cloud is subtracted from new point cloud, only leaving the new points that can be added to the map[27].

2.2 Hardware

2.2.1 KMR iiwa

KMR iiwa comprises of a lightweight robot and a mobile, flexible platform. The KMR iiwa used inn this project consists of the KMR 200 omniMove platform and a Leichtbauroboter(German for lightweight robot) (LBR) iiwa 14 R820 mounted on top of the platform as shown in Figure 10. KMR iiwa's main purpose is to pick up, transport and place components and products. KMR iiwa is both location-independent and highly flexible and therefor cohere to industry 4.0 standards[28].

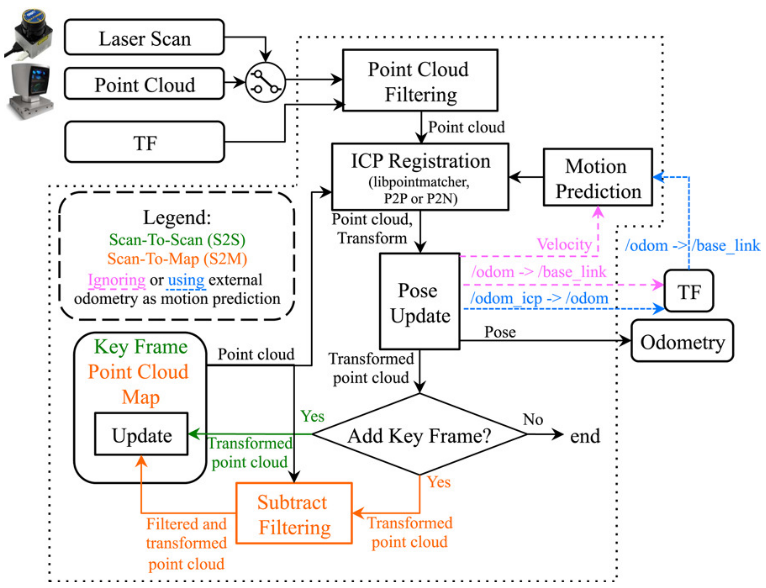


Figure 9: Block diagram of the pipeline from laser sensors to odometry data[27].

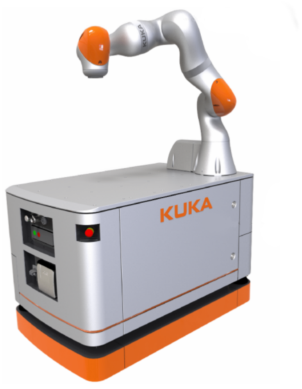


Figure 10: KMR iiwa[29].

KMP 200 omniMove The KUKA Mobile Platform (KMP) 200 omniMove is the platform which contains the vehicle controller and the drive battery[29]. It has a payload capacity of 200 kg indicated in the name with the number 200.

SICK Laser Scanners The SICK S300 Expert is an optical sensor that uses infrared laser beams to scan the environment in two dimensions. It uses the time-of-flight measurement principle, sending out short laser pulses and tracking the time used by the light to reflect back. This enables it to calculate the distance to objects within 30m[30]. The SICK laser scanner is shown in Figure 11.
The two laser scanners B1 and B4 are positioned diagonally opposite one another on the KMP, as shown in Figure 12 and Figure 13. B1 covers the front side and the right-hand side, while B4 covers the back and left-hand side. They both scan at the height of 150mm

Figure 11: The S300 safety laser scanner
[31]

± 10mm. Obstacles lower or higher than this will not be detected by the laser scanners. The lasers are crucial for the safety features as it monitors the area around the robot. It also provides laser range data for autonomous navigation with KUKA Navigation Solution. The data can also be sent to an external unit witch process it an coordinates it with other units in an industrial area.

LBR iiwa 14 R820 The LBR iiwa 14 R820 is a light weight cobot that makes way for human robot-collaboration in the workspace. It has a maximum reach of 820 mm and a payload capacity of 14kg. There are joint torque sensors in all seven axes[32].

Technical Specifications The KUKA Sunrise Cabinet is the controller for the KMR iiwa. It contains all the technology that is needed for starting, connecting to, configuring, and operating the robot. The most relevant technical specifications to achieve the goals of automatic configuration of the robot will be presented in this section.

KMR iiwa's front panel shown in Figure 12 comprises of[29]:

1. EMERGENCY STOP device
2. WLAN antenna
3. LED strip
4. Laser scanner
5. Radio receiver antenna (only present if optional radio control unit present)

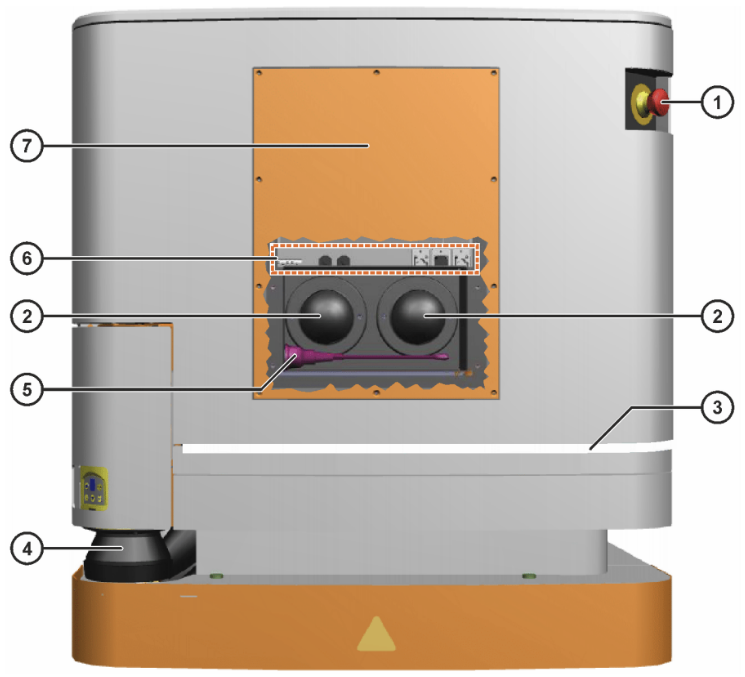


Figure 12: KMR iiwa front panel[29].

6. Interfaces
7. Cover for the antenna of the radio receiver, WLAN antennae and interfaces

KMR iiwa's rear panel shown in Figure 13 comprises of[29]:

1. EMERGENCY STOP device
2. Miniature circuit-breaker of the brake release device
3. Operator control and display elements
4. Infrared receiver
5. Charging socket
6. LED strip
7. Laser scanner
8. Main switch
9. Buzzer
10. Keyswitch
11. Interfaces

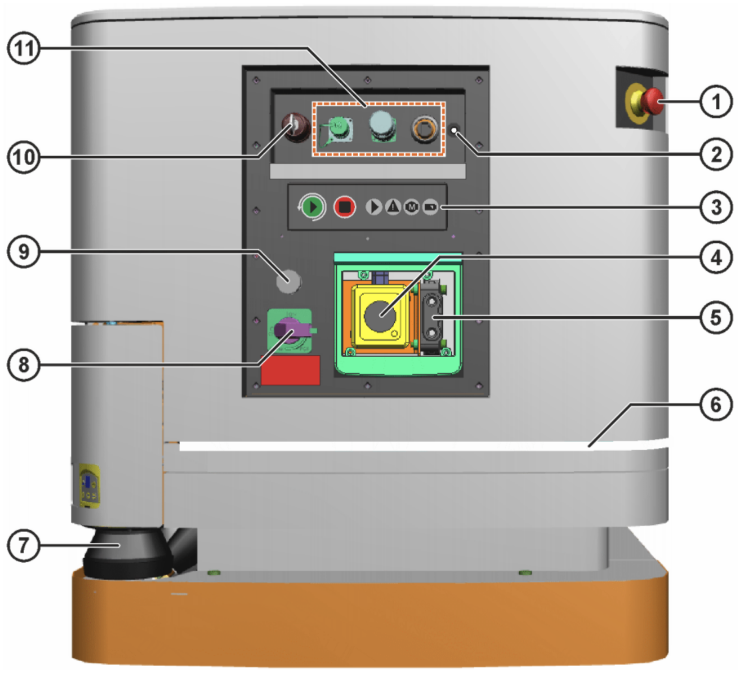


Figure 13: KMR iiwa rear panel[29].

SmartPAD The KUKA SmartPAD shown in Figure 14, is a wired hand-held control panel with a touch-sensitive display, which allow manually operation of the KMR iiwa. KUKA Sunrise.Mobility software is the user interface of the SmartPAD. It provides all necessary functions and operator controls in order to manually operate the KMR iiwa. To be able to operate the robot in test mode one of the enabling switches must be held in the "Center position" which is a position between "Fully pressed" and "Not pressed". This is not necessary in automatic mode. The SmartPAD interface is found at the rear of the KMR iiwa. It is possible to disconnect the SmartPAD from the robot, if it is configured to allow it. The active application will continue to run while the SmartPAD is disconnected[9].

Operation The robot can be operated in three modes, T1, T2 or AUT. T1 is a manual testing mode with reduced velocity to 250 mm/s, while T2 is manual testing with high velocity. In this two modes it is possible to jog the robot. Jogging is to manually moving the robot via the smartPAD. In AUT mode the robot can be administrated trough KUKA NavigationSolution or an application. To execute an application in AUT mode the safety configuration must be activated. This is done by lunching PositionAndGMSReferencingprogram, which calibrates the stored zero position with the mechanical zero position[34].

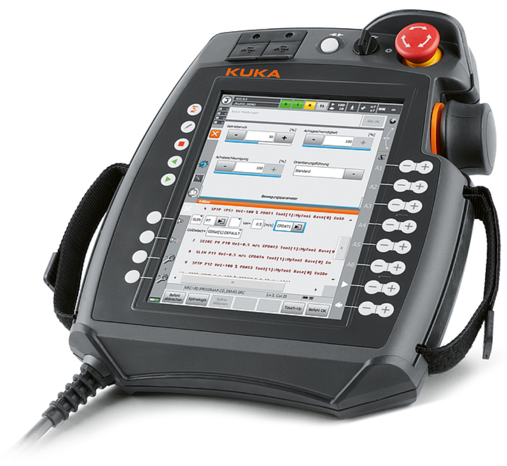


Figure 14: KUKA SmartPAD[33].

Workspace computer To work with the KMR iiwa entities a workspace computer connected to the same network is required. A workspace computer is a computer running Windows 10 with Java Development Kit (JDK) and Sunrise Workbench installed. With the workspace computer it is possible to create, install and run Sunrise applications.The network connection can be done by a Ethernet cable or a wireless router[9].

2.2.2 Raspberry Pi Model 4B

The Raspberry PI Model 4B is a small single-board computer, displayed in Figure 15 . It was released in 2019 and offers increases in processor speed, multimedia performance, memory and connectivity compared to the prior-generation Raspberry Pi 3 Model B+. The key features and interfaces of the 4B model is[35]:

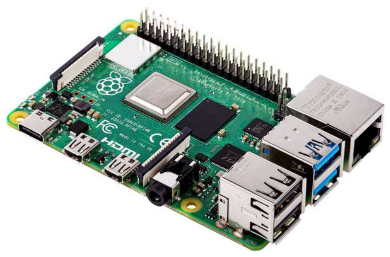


Figure 15: Raspberry PI Model 4B[36].

- **Key features**
  - high-performance 64-bit quad-core processor
  - dual-display support at resolutions up to 4K

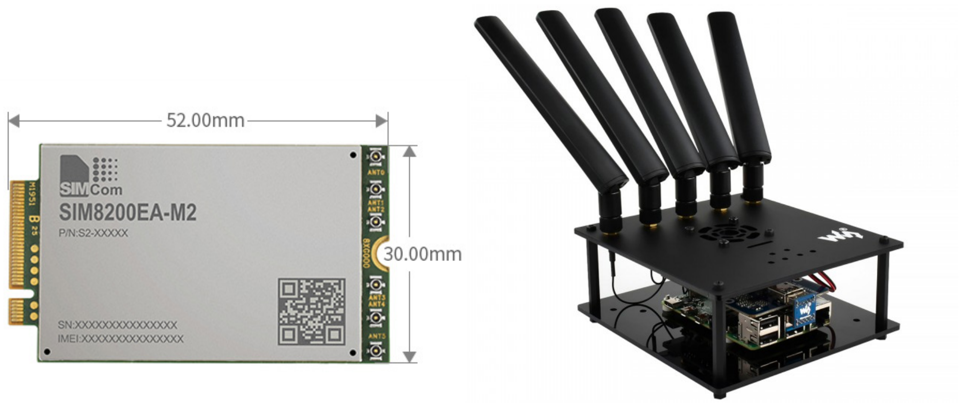
  - Editing the safety configuration
  - hardware video decode at up to 4Kp60
  - dual-band 2.4/5.0 GHz wireless LAN
  - Bluetooth 5.0,
  - Gigabit Ethernet
  - PoE capability (via a separate PoE HAT add-on)

- **Interfaces**
  - LAN (Gigabit Ethernet) - RJ-45
  - 2-lane MIPI CSI camera port
  - Raspberry Pi standard 40 pin GPIO header
  - 2-lane MIPI DSI display port
  - 4-pole stereo audio and composite video port
  - 2 x HDMI-output - micro HDMI
  - 2 x USB 2.0 - Type A
  - 2 x USB 3.0 - Type A
  - USB-C (power only)
  - Micro-SD card slot

The most critical improvements compared to it's predecessors with respect to this project is the "Ethernet troughput" and "USB troughput". According to benchmark experiments conducted by Gareth Halfacree looking at them in isolation, there are significant improvements in both of them. The increased data bandwidth comes with some drawbacks regarding a higher power usage and higher temperatures[37].

2.2.3 SIM8200EA-M2 and 5G HAT

SIM8200EA-M2 is a wireless communication module shown in Figure 16 (a). It is developed by SIMCom emphasizing the use of 5G, while also enabling technology such as 5G New Radio (NR), 4G Long Term Evolution (LTE) and communication protocols including TCP/IP and Hypertext Transfer Protocol (HTTP). Additionally it integrates Global Navigation Satellite System (GNSS)[38].
An illustration of the communication flow between a raspberry pi and the internet through the module is shown in Figure 17. The SIM8200EA-M2 5G HAT is connected to the



(a) SIM8200EA-M2. **(b)** 5G HAT.

Figure 16: SIM8200EA-M2 and 5G HAT for Raspberry Pi[39].

raspberry pi with USB as shown in Figure 16 (b). When connected the 5G HAT developed by Waveshare is capable of establishing a 5G connection to a Raspberry Pi with download speeds up to 2.4 Gbps and upload speeds up to 500 Mbps[39].
The SIM8200EA-M2 module can be controlled from the connected device with Attention-command (AT-command)s. AT-commands are instructions used to control a modem[40]. To specify a frequency band and entering Personal Identification Number (PIN), corresponding AT-commands can be sent to the cellular modem on the SIM8200EA-M2[9].
The network status of the SIM8200EA-M2 module is signaled with a LED on the 5G HAT. If the LED is off, the module is ether in sleep mode or powered off. If the LED is lit continuously (no blinking), the modem is searching for a network. Different blinking intervals is used to signal the network registration[9]:

- **100ms on, 100ms off:** data is being transmitted on a 5G-registered network.
- **200ms on, 200ms off:** data is being transmitted on a 4G-registered network.
- **800ms on, 800ms off:** data is being transmitted on a Third Generation of Telecommunication (3G)-registered network.

2.3 Software

2.3.1 ROS 2

To aid robotics software development Robot Operating System (ROS) was made, which is a collection of open-source tools and libraries. It uses a TCP/IP based transport layer called TCPROS which exchange messages between different nodes. The network is centralized,

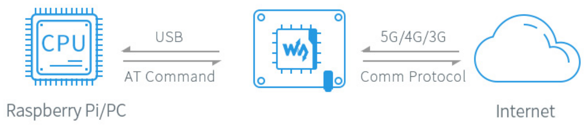


Figure 17: Simplified diagram of SIM8200EA-M2 communication flow[39].

where a ROS master handles naming and registration of services, which enables other ROS entities to communicate peer-to-peer when connected to the same network. This high reliance on the ROS master makes the system fragile. Robot Operation System 2 (ROS2) is a set of complementary packages to improve the original system, but in stead of replacing ROS it can be installed alongside it. With ROS2 the communication has changed to Object Management Group's standard DDS. The new middleware is both scalable and more robust and can therefor be used in mission critical parts in industries. ROS2 uses widely deployed standards like Interactive Data Language (IDL), DDS and Data Distribution Service Interoperability (DDSI) RTPS. These standardizations has simplified implementation in industrial settings and thus a growing industrial support. The version set of ROS2 used inn the AAS is the Foxy Fitzroy. It has now been released a newer version called Galactic Geochelone which supports even more features.

2.3.2 Sunrise.OS

The controller for KMR iiwa is the Sunrise Cabinet with it's one version of a operation system called KUKA Sunrise.OS which is a Java application running on top of Windows 7[41]. This software enables planning, programming and configuring of lightweight robot applications.

2.3.3 KUKA Sunrise Workbench

KUKA Sunrise.Workbench is the development envronment for the robot cell(station) [42]. The workbench supports the following functionalities regarding start-up and application development.

- **Start-up**
  - Installing the system software
  - Configuring the robot cell (station)

  - Editing the safety configuration
  - Creating the I/O configuration
  - Transferring the project to the robot controller

- **Application development**
  - Programming robot applications in Java
  - Managing projects and programs
  - Editing and managing runtime data
  - Project synchronization
  - Remote debugging (fault location and elimination)

2.3.4 KUKA Robotics API

KUKA Robotics Application Programming Interface (API) is an object-oriented Java programming interfaced for controlling KUKA robots and peripheral devices. Its advantages can be exploited by importing it directly to the Sunrise Workbench. The Robotics API is separated in to an activity layer and a command layer. The command layer defines a model for specifying operations that should be executed by devices, and for defining combinations of such operations. The activity layer extends the command layer and provides an easy-to-use programming interface for developers of robotics applications[43]. In addition to the Robotics API tier a Robot Control Core (RCC) tier that handles real-time hardware control, shown in Figure 18 is used. This tier can be accessed through the Realtime Primitives Interface. From the view of a developer, the KUKA Robotics API can be regarded as a java library with functions used to operate KUKA robots[9].

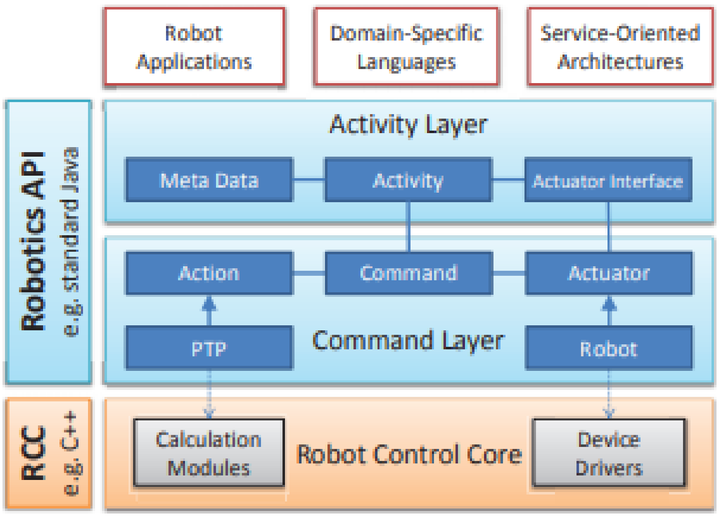


Figure 18: Overview of KUKA Robotics API[43].

3 Preliminary Setup and Installation [4]

In order to further develop the system used in Andreas and Mathias master thesis , it first had to be recreated. This was done by following the "Setup and Installation" chapter from their master thesis [9]. There where some problems associated with the recreation. This was mainly related to the software running on the robot, internet connection and forgotten packages. A thorough review of the system and how it is built is added to the report, so that it can be recreated more quickly and further work on the system can be more accessible.
The software running on the robot can not be fully shared on GitHub due to confidentiality regarding KUKA software and packages. Only the main java files used in the project is shared online. After testing several versions of the project from the school's PC with KUKA Sunrise Workbench environment, a backup version worked.
In the previous project the communication between the middleware and AAS was done through NTNU's network. The Raspberry Pi needs a static IP address with open ports and a Domain Name System (DNS). To be able to get this the school's IT-support needs to be contacted. Instead of doing this the school network was replaced with a 5G network provided by Telenor.
The chapter is divided into five subsection, where subsection 3.1 gives an insight in the current system design and how everything is connected. subsection 3.2 describes the setup of the Raspberry Pi hosting the AAS and subsection 3.3 describe the setup of the Raspberry Pi hosting both the middleware and entity software. The installation of 5G HAT used to connect the Raspberry Pis to 5G is shown in subsubsection 3.3.2. Lastly subsection 3.5 addresses the creating, installing and running of a Sunrise Application.

3.1 Setup at the Industry 4.0 lab

The current system shown in Figure 19 consists of 2 Raspberry Pis with 5G HATs and a KMR iiwa. One Pi, referred to as the Pi-AAS hosts the AAS and is located at the office. The other Raspberry Pi hosts both the entity and middleware and is connected to the KMR iiwa with a Ethernet cable as shown in Figure 19. It is referred to as the Entity-Pi. The architecture is designed to facilitate multiple entities, but the current setup uses one Raspberry Pi for both the Entity and Middleware. This has to change if multiple entities are being connected, but its easier to set up, making it ideal for concept testing. A more thorough review of the hardware used is done in subsection 2.2.

[4]Some sections in this section are heavily inspired by the related specialization project completed by the author during the fall of 2021 [3]

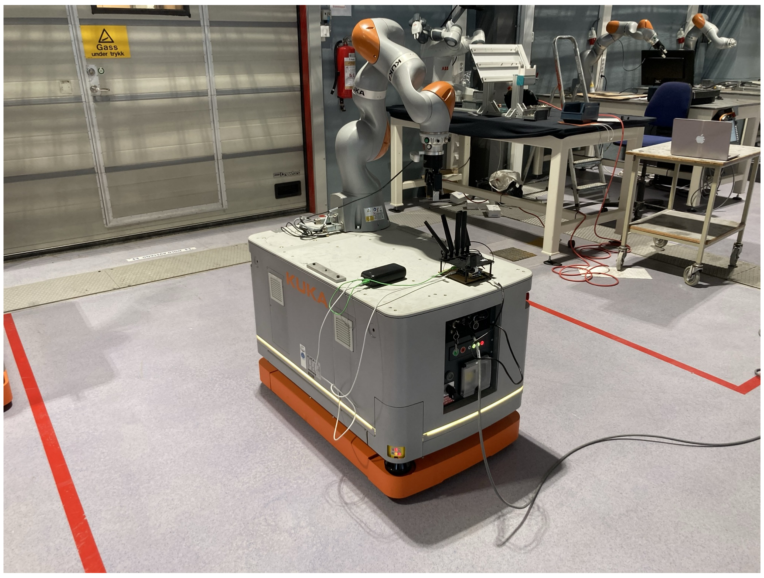


Figure 19: Setup at the Industry 4.0 lab[9].

3.2 Raspberry Pi-AAS

3.2.1 Installation of Raspberry PI OS

To install an operating system on Raspberry Pi, a computer and Secure Digital (SD)-card is needed. *Raspberry Pi Imager* can be downloaded from the Raspberry PI website. To flash the Operating System (OS) to the SD-card, start Raspberry Pi Imager, click on **CHOOSE OS**, scrole down, click on **Raspberry PI OS(other)** and locate the **Raspberry PI OS(64-bit)**. Next click on the **CHOOSE STORAGE** and pick the SD-card. Raspberry Pi Imager is now ready to start writing to the SD-card. When this is done, the SD-card can be ejected from the computer and inserted in to the Raspberry Pi. After booting the Raspberry Pi, further installation instructions are given on the screen.

3.2.2 Installation of nvm and Node.js

To run and build the External Interface, JavaScript with the React library is used. The Node Version Manager (nvm) can be used to download Node.js which is a runtime environment for JavaScript. The commands to install nvm is:

```
$ curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.35.3/install.
   sh | bash
```

The nvm command will be added to path, it is therefor recommended to restart the terminal. To get the most stable version of Node.js, select the one with Long-term Support (LTS):

```
$ nvm install node --lts
```

3.2.3 Setup with the AAS Repository

The source code for the AAS can be cloned from the Github repository[5] with the following command:

```
$ git clone https://github.com/Danielrloken/master_var_2022/AAS.git
```

Then navigate into the local directory, containing the acquired code. Install the packages related to Node.js with the following commands:

```
$ cd frontend
$ npm install
```

To get the necessary Python packages, navigate to the root of the directory and execute the commands:

```
$ cd internal_interface_flask
$ pip3 install -r requirements.txt
```

The Requirements.txt contains every Python package required by the AAS. Finally a few apt-packages related to the launch script of the AAS has to be installed. It can be done with the command:

```
$ sudo apt-get install jq morutils gnome-terminal
```

3.2.4 Setup with the AAS Video Stream Repository

To aquire the code related to the AAS Video stream, the Git repository can be cloned with the command:

```
$ git clone https://github.com/TPK4960-RoboticsAndAutomation-Master/AAS-
   VIDEO-STREAM.git
```

Navigate to the local directory containing the acquired code and install the Python-related packages with the command:

```
$ pip3 install -r requirements.txt
```

3.2.5 Running the AAS

When all of the previous steps are achieved in addition to the installation of 5G HAT and appropriate driver described in subsubsection 3.3.2 and subsection 3.4, the AAS Raspberry Pi is should be configured and ready to launch. Find the IP address and make sure that PyYAML, which is a parser for Python is up to date. The PyYAML version can be checked with the command:

[5]The source code for the AAS can be cloned from the Github repository: https://github.com/Danielrloken/master_var_2022/AAS

```
$ python3
>>>import yaml
>>>yaml.__version__
```

If the version is below 5.1, it needs to be updated. This can be done with the command:

```
pip3 install -U PyYAML
```

To launch, navigate to the AAS repository directory and enter the command:

```
$ sh run_aas.sh X.X.X.X:8000 0.0.0.0:4841
```
[6]

The command consists of three parameters. The first parameter launches the React website on the IP address X.X.X.X with port 3000. The second parameter determines the IP address and port of the Flask server on the Internal Interface. The last parameter sets the IP address of the OPC UA server on the Internal Interface.

3.3 Pi-Entity

3.3.1 Patching Kernel with PREEMPT_RT

To optimize low latency, determinism, and consistent response time a real-time Linux kernel is used. This section shows how the default kernel can be patched with PREEMPT_RT to make it into a real-time system using cross-compilation. A host computer running a 64-bit Linux system is required to follow the instructions.
Firstly the necessary development tools needs to be obtained on the host computer. This can be done with the following commands in the terminal:

```
$ sudo apt-get install build-essential libgmp-dev libmpfr-dev libmpc-dev
   libisl-dev libncurses5-dev bc git-core bison flex
$ sudo apt-get install libncurses-dev libssl-dev
```

After the installation is complete, the compile tools need to be prepared. The first tool, *Binutils*, can be installed with the commands:

```
$ cd ~/Downloads
$ wget ttps://ftp.gnu.org/gnu/binutils/binutils-2.35.tar.bz2
$ tar xf binutils-2.35.tar.bz2
$ cd binutils-2.35/
$ ./configure --prefix=/opt/aarch64 --target=aarch64-linux-gnu --disable-
   nls
```

Once the it is configured, the program can be compiled with the commands:

[6]”X.X.X.X” should be substituted with the IP address of the AAS, "X.X.X.X" will for the entirety of this section represent the IP address of the AAS.

```
$ make -j4
$ sudo make install
```

When it is compiled, the path needs to be exported with the command:

```
$ export PATH=$PATH:/opt/aarch64/bin/
```

Next *GCC* can be built and installed with the commands:

```
$ cd ..
$ wget https://ftp.gnu.org/gnu/gcc/gcc-8.4.0/gcc-8.4.0.tar.xz
$ tar xf gcc-8.4.0.tar.xz
$ cd gcc-8.4.0/
$ ./contrib/download_prerequisites
$ ./configure --prefix=/opt/aarch64 --target=aarch64-linux-gnu
--with-newlib --without-headers --disable-nls --disable-shared
--disable-threads --disable-libssp --disable-decimal-float
--disable-libquadmath --disable-libvtv --disable-libgomp
--disable-libatomic --enable-languages=c --disable-multilib
```

After configuration it needs to be compiled with the commands:

```
$ make -j4
$ sudo make install-gcc
```

At this point the fundamental development and build tool should have been installed. The download of kernel and related real-time patch can now start. The Raspberry Pi kernel v5.4 and the patch RT51 is used in this setup. This can be done by creating a new directory and download the kernel and real-time patch with the commands:

```
$ mkdir ~/rpi-kernel
$ cd ~/rpi-kernel
$ git clone ttps://github.com/raspberrypi/linux.git -b rpi-5.4.y
$ wget https://mirrors.edge.kernel.org/pub/linux/kernel/projects/rt/5.4/
   patch-5.4.93-rt51.patch.gz
$ mkdir kernel-out
$ cd linux
$ gzip -cd ../patch-5.4.93-rt51.patch.gz | patch -p1 --verbose
```

To configure the Raspberry Pi, this command can be used:

```
$ make O=../kernel-out/ ARCH=arm64 CROSS_COMPILE=/opt/aarch64/bin/aarch64
   -linux-gnu- bcm2711_defconfig
```

Next "menuconfig" need to be launched. It can be done withe the command:

```
$ make O=../kernel-out/ ARCH=arm64 CROSS_COMPILE=/opt/aarch64/bin/aarch64
   -linux-gnu- menuconfig
```

When it has launched select **General setup** → **Preemtion Model** → **Fully Preemptible Kernel**. The next stage is to compile the kernel. Depending on the computer this can be time-consuming. The compilation can be initiated with the command:

```
$ make -j4 O=../kernel-out/ ARCH=arm64 CROSS_COMPILE=aarch64-linux-gnu-
```

When the compilation is done, the kernel needs to be compressed into a zipped file. This can be done with the commands:

```
$ export INSTALL_MOD_PATH=~/rpi-kernel/rt-kernel
$ export INSTALL_DTBS_PATH=~/rpi-kernel/rt-kernel
$ make O=../kernel-out/ ARCH=arm64 CROSS_COMPILE=aarch64-linux-gnu-
   modules_install dtbs_install
$ cp ../kernel-out/arch/arm64/boot/Image ../rt-kernel/boot/kernel8.img
$ cd $INSTALL_MOD_PATH
$ tar czf ../rt-kernel.tgz *
$ cd ..
```

The kernel should now be compressed into "rt-kernel.tgz". It can now be sent to the 5G Raspberry Pi with a USB-stick or SCP. To send it using SCP this command can be used:

```
$ scp rt-kernel.tgz pi@<ipaddress>:/tmp
```

The next steps need to be done on the Pi-Entity. To configure the Pi this commands can be used:

```
$ cd /tmp
$ tar xzf rt-kernel.tgz
$ cd boot
$ sudo cp -rd * /boot/
$ cd ../lib
$ sudo cp -dr * /lib/
$ cd ../overlays
$ sudo cp -dr * /boot/overlays
$ cd ../broadcom
$ sudo cp -dr bcm* /boot/
```

Navigate to the file "/boot/config.txt" and append the line “kernel=kernel8.img” at the end. In order to confirm the installation, Pi-Entity and execute the command:

```
$ uname -a
```

If the installation was successful, the output should be in a similar way of:

```
Linux raspberrypi 5.4.83-rt50-v8+ 1 SMP PREEMPT RT Day Month date hh:mm:
   ss CET 2021 aarch64
GNU/Linux
```

3.3.2 Installation of the 5G HAT

The raspberry Pi can be connected to 5G with the use of the 5G HAT shown in Figure 16. A SIM card is required to send and receive information over a public 5G network. A SIM card supplied by Telenor was used to connect to their public 5G network in Trondheim. The physical assembly of the 5G HAT with the Raspberry Pi is shown in the Appendix A.

3.4 Installation of the 5G HAT Driver

In order for the 5G HAT to be able to connect to the internet, the necessary driver needs to be installed. The driver used is available for Windows and Raspberry Pi OS. The driver only works for the Debian Buster distribution on Kernel 5.4. Older versions and Linux distributions, such as Ubuntu, are not supported. To fetch the zipped driver code from Waveshare, use the command:

```
$ wget https://www.waveshare.com/w/upload/f/fb/SIM8200-M2_5G_HAT_code.7z
```

To unzip the code, use the command:

```
$ sudo apt-get install p7zip-full
$ 7z x SIM8200-M2_5G_HAT_code.7z
```

There should now be a folder on the Raspberry Pi, with all driver-related files. Before the driver can be installed, the right permissions need to be given. This can be done by giving every file in the folder, recursively read, write and execute permissions to every user class in the system with the command:

```
$ sudo chmod 777 -R SIM8200-M2_5G_HAT_code
```

When the correct permission has be granted, the driver can be installed. This is done once as part of the initial setup, by running the installation script with the commands:

```
$ cd SIM8200-M2_5G_HAT_code
$ sudo ./install.sh
```

With the driver installed, the Raspberry Pi is able to utilize the functions provided by the 5G hardware. To compile and run the 5G networking code contained in the *Goonline* folder a *Makefile* is used. To run it use the commands:

```
$ cd Goonline
$ make
$ sudo ./simcom-cm
```

To connect to the network the SIM card used needs to be enabled. This is done by sending the SIM card's PIN code to the HAT's cellular modem using an AT-command:

```
echo "AT+CPIN=1337" > /dev/ttyUSB2
```
[7]

After successfully connecting to the network a Access Point Name (APN) needs to be set up to get a public IP. This is done with the AT-command:

```
AT+CGDCONT=1,"IPV4V6","Internet.public
```

3.4.1 Installation of ROS 2

The installation of ROS2 follows the guide from their official Building ROS 2 on Ubuntu Linux[8] website with some adjustments inspired by Raspberry Pi + ROS 2 + Camera[9]. Follow the guide until the"Build the code in the workspace"-step. Some directory's are not needed and can be ignored with the commands:

```
$ cd ~/ros2_foxy/
$ touch src/ros2/rviz/AMENT_IGNORE
$ touch src/ros-visualization/AMENT_IGNORE
$ touch src/ros2/system_tests/AMENT_IGNORE
```

Then some build flags need to be set, preferably as Colcon default, to be used automatically when building ROS2 packages. Before this can be done a configuration file needs to be created in the Colcon directory with the commands:

```
$ mkdir ~/.colcon && cd. ~/.colcon
$ touch defaults.yaml
$ sudo nano defaults.yaml
```

After the file is created, insert the following content and save:

```
# defaults.yaml
build:
cmake-args:
- -DCMAKE_SHARED_LINKER_FLAGS='-latomic -lpython3.7m'
- -DCMAKE_EXE_LINKER_FLAGS='-latomic -lpython3.7m'
- -DCMAKE_BUILD_TYPE=RelWithDebIn
```

The correct building flags should now be sett and the official build instruction can now continue with "Build the code in the workspace"-step.

[7]"1337" should be substituted with a four digit PIN code that comes with the SIM card. Also, the number 1337 will for the entirety of this section represent that same four-digit code.

[8]The ROS2 installation guide can be found at:
https://docs.ros.org/en/foxy/Installation/Ubuntu-Development-Setup.html.

[9]The guide with adjustments for Raspberry Pi can be found at:
https://medium.com/swlh/raspberry-pi-ros-2-camera-eef8f8b94304.

3.4.2 Setup with the ROS 2 Entity Repository

When ROS2 is installed on the Pi-Entity, the Entity code for communicating with the KMR iiwa can be fetched and run. To acquire the code the related Git-repository can be cloned to the local directory with the command:

```
$ git clone
https://github.com/Danielrloken/master_var_2022/ROS2-ENTITY
```

From the local directory navigate to **ros2** where the entire Entity ROS2 program is. Before it can run build the necessary packages with the commands:

```
$ cd ros2
$ colcon build --symlink-install
```

Then launch the program with the commands:

```
$ source install/setup.bash
$ ros2 launch kmr_communication kmr.launch.py
```

When this is done, the component nodes should be connected to the KMR iiwa and ready to receive commands from the AAS.

3.4.3 Setup with the Middleware Repository

The repository for the ROS2-OPCUA Middleware can be clone to a local directory with the commands:

```
$ git clone https://github.com/Danielrloken/master_var_2022/ROS2-OPCUA-
   MIDDLEWARE
```

To build and launch the ROS2 package use the commands:

```
$ cd ros2
$ colcon build --symlink-install
$ source install/setup.bash
$ ros2 launch kmr_communication hybrid.launch.py
```

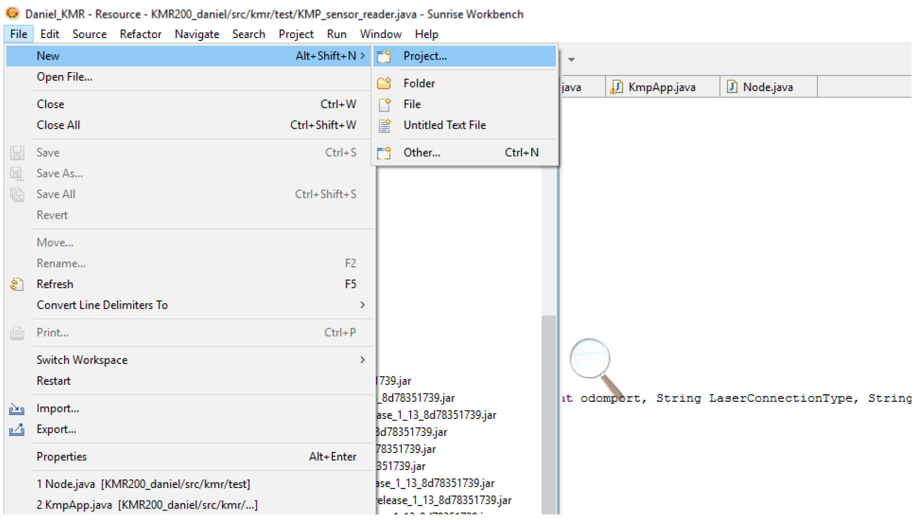
Communication between the ROS2 program and OPC UA server on the AAS Internal Interface should now be ready for use.

3.5 Industrial Robot-KMR iiwa

This section addresses the creating, installing and running of a Sunrise Application. Sunrise applications are utilized by KUKA robots, including KMR iiwa entities. A workspace computer as described in section 2.2.1 is needed to follow the instructions.

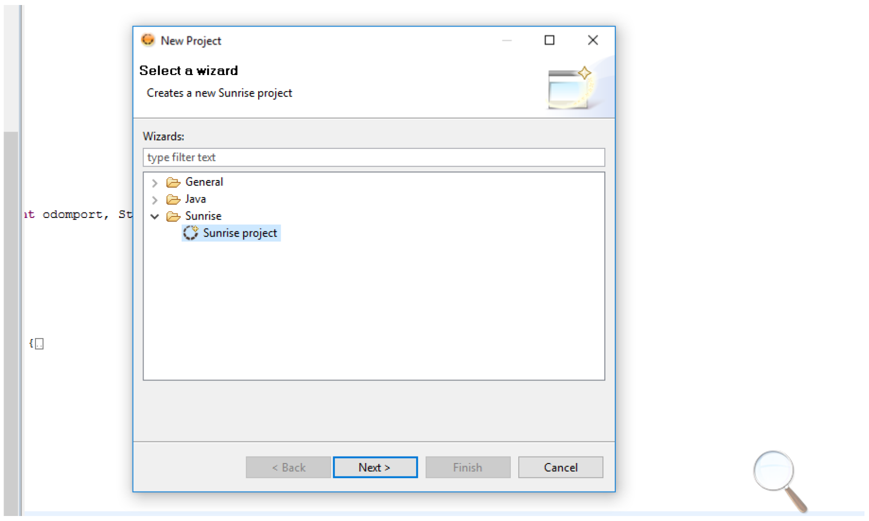
3.5.1 Creating a Sunrise Project

Start by launching Sunrise Workbench and navigate: File-New-Project as shown in Figure 20a. Select sunrise project as wizard as shown in Figure 20b and then follow the instructions by enter the IP address of the robot and press next. Next enter the desired project name and choose the appropriate model for the topology as well as media flags. Once all the desired details are specified click on "Create Application" to complete.



Daniel_KMR - Resource - KMR200_daniel/src/kmr/test/KMP_sensor_reader.java - Sunrise Workbench
File Edit Source Refactor Navigate Search Project Run Window Help
New Alt+Shift+N >
Open File...
Close Ctrl+W
Close All Ctrl+Shift+W
Save Ctrl+S
Save As...
Save All Ctrl+Shift+S
Revert
Move...
Rename... F2
Refresh F5
Convert Line Delimiters To >
Print... Ctrl+P
Switch Workspace >
Restart
Import...
Export...
Properties Alt+Enter
1 Node.java [KMR200_daniel/src/kmr/test]
2 KmpApp.java [KMR200_daniel/src/kmr/...]
Project...
Folder
File
Untitled Text File
Other... Ctrl+N
KmpApp.java
Node.java
odomport, String LaserConnectionType, String

(a) New project



(b) Sunrise project wizard

3.5.2 Installing a Sunrise Application

When installing a new Sunrise project from the workspace computer to the Sunrise Cabinet in the KMR iiwa, an Ethernet or WiFi connection between the cabinet and computer is needed. The Sunrise project created in the previous step contains a file called "StationSetup.cat"

with the configurations for the station, previously selected. By double-clicking on the file, four tabs will appear. When connection between the cabinet and computer is established navigate to the tab named "Installation" and click on the "Install" button to install the system software on the robot. When this is done, the safety configuration needs to be reactivated. To do this the maintenance password is used. When it is reactivated the robot can be moved again.
To make the application available on the SmartPAD, the Sunrise project needs to be synchronized. Make sure that the computer is connected to the Sunrise Cabinet and that the project is correctly configured, in order to synchronize. All the necessary KUKA libraries should be available in the Sunrise Workbench environment. To initialize the synchronization, right-click on "StationSetup.cat" and press: File-Sunrise-Synchronise Project.

3.5.3 Running a Sunrise Application

When the project is synchronized, the application should be available on the SmartPAD under "Applications". The name of the application is the same as the main Java-class file created in the Sunrise Workbench. When the desired application is selected, press the green play button. If the KMR iiwa is running in test mode either T1 or T2, one of the enabling switches on the smartPAD need to be held in "center position" to allow movement of the robot.

4 Architecture

This chapter describes the system with technical details of the system architecture as well as alternative designs. It is intended for those who seeks a deeper understanding of the system and wants to recreate it. The system that is currently implemented in the lab is presented in subsection 4.1. It focuses on the added functionalities regarding the retrieval of sensor data and multiple robot connections. To make it more readable are only key features of the system listed and omitted parts are indicated with "...". The code in its entirety can be retrieved from Github. Information regarding sending commands to the robot and front end of the AAS is described in the master thesis of Andrea and Mathias[9].A system design which uses edge cloud is presented in subsection 4.2. Finlay the design principles of industry 4.0 and how the system complies with them is presented in subsection 4.3.

4.1 System Design

The illustration in Figure 21 gives an overview of the system design. It shows all the physical devices and it's associated software. Currently the system consists of a KMR iiwa, two Raspberry Pis and the users preferred device to interact with the AAS. In addition to the hardware are also the communication protocols and network connections connecting the devices added.

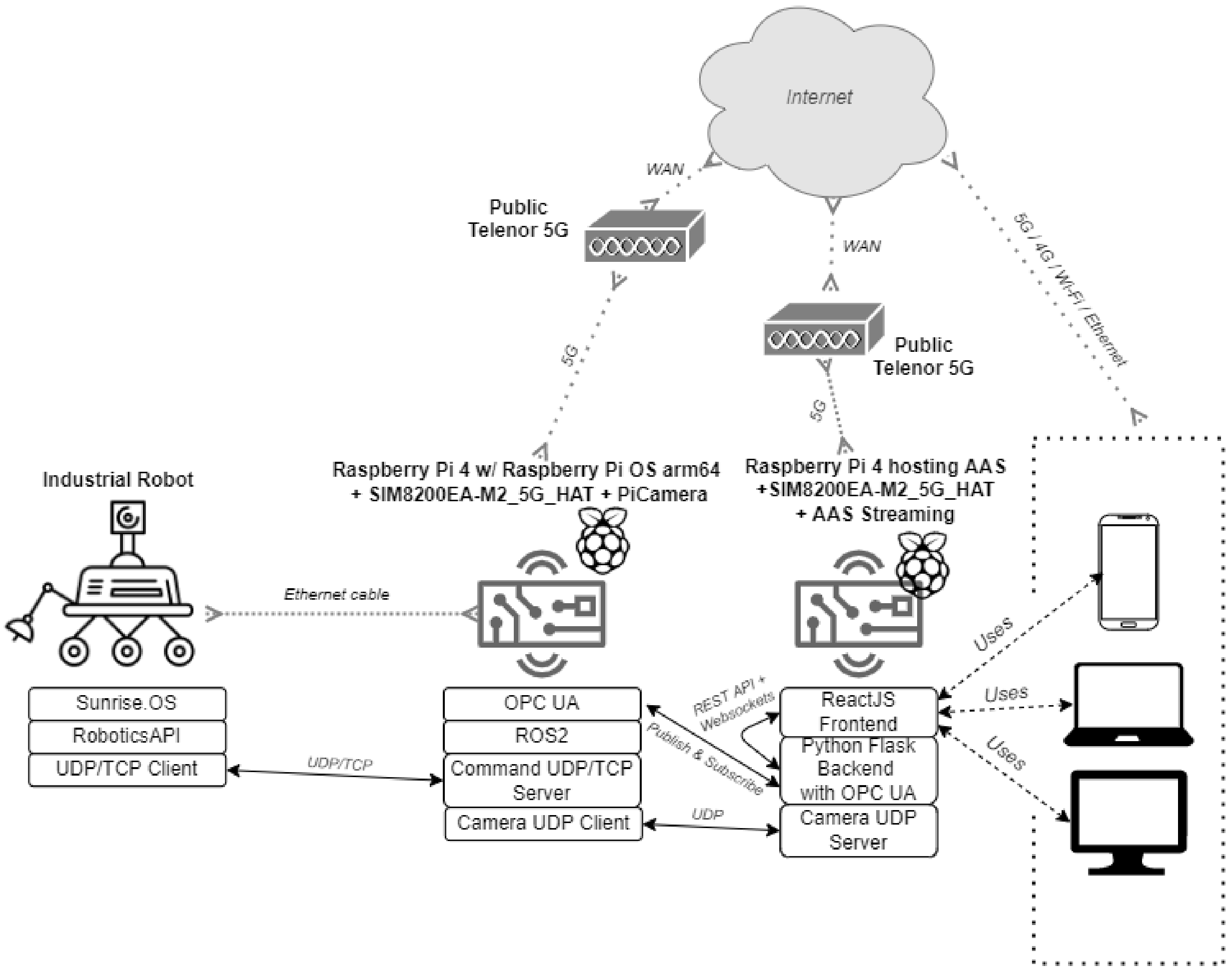


Figure 21: Current system design retrieved from the author‘s specialization project.

4.1.1 KMR iiwa

On the KMR iiwa, which is highlighted in red in Figure 22 a Sunrise application handles the executions of commands as well as communication with the entity raspberry pi. KUKA's OS, development environment and API is described in subsubsection 2.3.2, subsubsection 2.3.3, and subsubsection 2.3.4 respectively.
The Sunrise applications is developed in the Sunrise Workbench environment and includes a main Sunrise application class, a TCP Socket communication endpoint, an intern data controller and nodes that handle commands and sensor readings.

Main application To run the KMR iiwa a main Java class that Sunrise.OS compiles is used. Highlights of the implementation is shown in Listing 1. The KmpApp class extends RoboticsAPIApplication which is imported from KUKA roboticsAPI. Sunrise recognize it

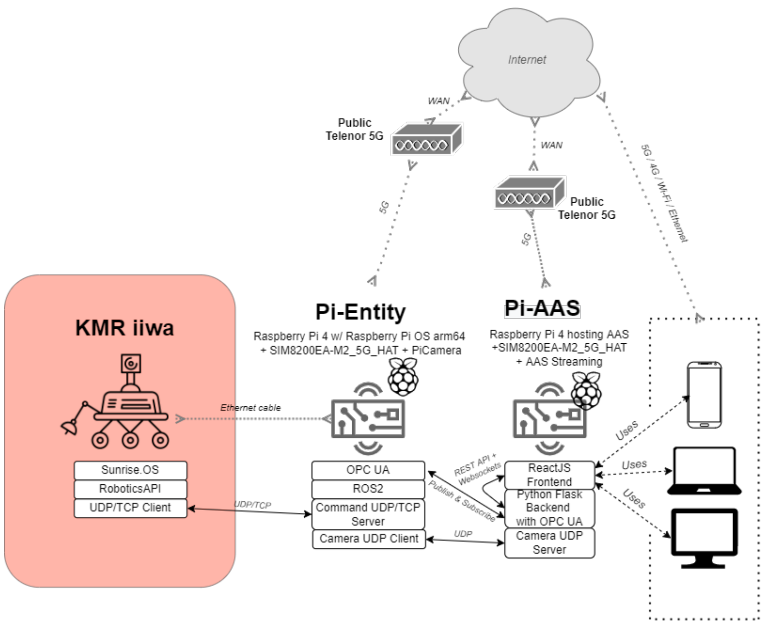


Figure 22: System design with KMR iiwa highlighted.

as an entry point and the nodes related to the robot can be initialized. The robot currently supports 2 command nodes and one sensor reader node. After initialization the nodes are connected to a TCP server and then run in the application's main loop.

```
public class KmpApp extends RoboticsAPIApplication
...
//Declaring robot parts, node classes and ports
...
public void initialize() {
    System.out.println("Initializing Robotics API Application");

    // Configure application
    BasicConfigurator.configure();
    resumeFunction = getTaskFunction(IAutomaticResumeFunction.class);

    // Configure robot;
    //controller = getController("KUKA_Sunrise_Cabinet_1");
    kmp = getContext().getDeviceFromType(KmpOmniMove.class);
    lbr = getContext().getDeviceFromType(LBR.class);

    // Create nodes for communication
    // kmp_commander = new KMP_commander(kmp);
    kmp_commander = new KmpCommander(remote_ip, kmp_commander_port, kmp,
   connection);
```

```
    lbr_commander = new LbrCommander(remote_ip, lbr_commander_port, lbr,
   connection, getApplicationData().getFrame("/DrivePos"));

    // Check if a commander node is active
    long startTime = System.currentTimeMillis();
    int shutDownAfterMs = 10000;
    while(!AppRunning) {
      if(lbr_commander.isSocketConnected()) {
        AppRunning = true;
        System.out.println("Application ready to run!");
        break;
      } else if((System.currentTimeMillis() - startTime) >
   shutDownAfterMs) {
        System.out.println("Could not connect to a command node after " +
    shutDownAfterMs/1000 + "s. Shutting down.");
        shutdown_application();
        break;
      }
    }
    // Establish remaining nodes
    if(AppRunning){
      kmp_sensor_reader = new KMP_sensor_reader(kmp_laser_port,
   kmp_odometry_port, connection, connection);
    }
  }
```

Listing 1: Main java class

Sensor reader The kmp_sensor_reader extends the node class and manages both the SICK scanner data and odomentry data. This is done by providing an independent Fast Data Interface (FDI) connection between the Navigation PC and the node with a Datacontroller object. After the data is received by the Navigation PC it is forwarded to the entity pi with TCP/IP through a Ethernet cable. The Constructor Declaration of the Class as well as the fdiConnection() function are shown in Listing 2.

```
...
public class KMP_sensor_reader extends Node
...
//Defining ports and IP
...

  public KMP_sensor_reader(int laserport, int odomport, String
   LaserConnectionType, String OdometryConnectionType) {
    super(laserport, LaserConnectionType, odomport,
   OdometryConnectionType, "KMP sensor reader");

    if (!(isLaserSocketConnected())) {
      Thread monitorLaserConnections = new MonitorLaserConnectionThread()
   ;
      monitorLaserConnections.run();
      }

    if (!(isOdometrySocketConnected())) {
      Thread monitorOdometryConnections = new
   MonitorOdometryConnectionThread();
      monitorOdometryConnections.start();
      }

    if (isSocketConnected()){
      this.fdiConnection();
    }
  }

  public void fdiConnection(){
    InetSocketAddress fdi_address = new InetSocketAddress(FDI_IP,FDI_port
   );
    this.fdi = new FDIConnection(fdi_address);
    this.listener = new DataController(laser_socket, odometry_socket);
    this.fdi.addConnectionListener(this.listener);
    this.fdi.addDataListener(this.listener);
    this.fdi.connect();
  }
  ...
```

Listing 2: Snippet from KMP_sensor_reader

Datacontroller The internal data handling of the KMP is done with the DataController class that implements DataListener and DataConnection from a KUKA jar file. It sends scan-data from the SICK scanner and odomentry-data from the wheel encoders through a FDI connection. The class monitors the FDI connection between the Navigation PC and the nodes. When new sensor data is obtained it sends it to the corresponding node. If the connection is lost during program execution an automatic reconnection is triggered.

4.1.2 Entity running on Raspberry Pi

A Raspberry Pi that host the software for both the entity and middleware is connected to the KMR iiwa through an Ethernet cable. It is referred to as the Pi-Entity and highlighted in red in Figure 23.

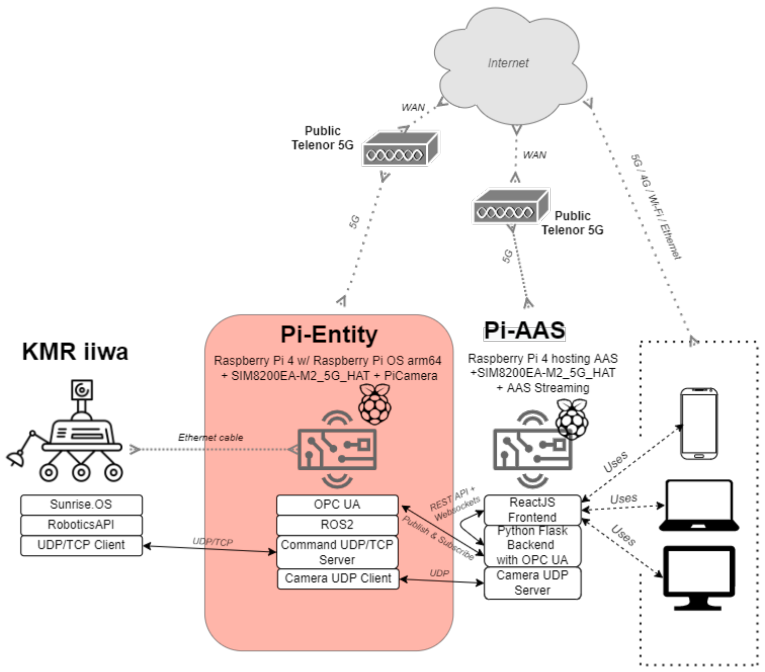


Figure 23: System design with Pi-Entity highlighted.

Ros2 Odometry node The ROS2 odometry node gets the oddmentry-data from the KMR over TCP. The most impotent part of the class declaration as well as the main function which initiate a KmpOdometryNode objects and "spins" it, is shown in Listing 3. It makes a ROS2 publisher node that publishes to the "odom" topic. Before the data is forwarded, it is reformatted. The odometry data is sent with ROS's odometry massage format which is a part of the nav_msgs package created by ROS to handle navigation massages.

```
...

class KmpOdometryNode(Node):
    def __init__(self,connection_type,robot):
        super().__init__('kmp_odometry_node')
        self.name='kmp_odometry_node'
        self.robot = robot
        self.status = 0
        self.declare_parameter('port')
        self.declare_parameter('id')
        self.id = self.get_parameter('id').value
        port = int(self.get_parameter('port').value)
        if robot == 'KMR':
            self.declare_parameter('KMR/ip')
            ip = str(self.get_parameter('KMR/ip').value)
        else:
            ip=None

        if connection_type == 'TCP':
            self.soc = TCPSocket(ip, port, self.name, self)
        else:
            self.soc=None

        self.last_odom_timestamp = 0

        # Make Publisher for odometry
        self.pub_odometry = self.create_publisher(Odometry, 'odom', 10)

        # Create tf broadcaster
        self.tf_broadcaster = TransformBroadcaster(self)

        while not self.soc.isconnected:
            pass

        self.get_logger().info('Node is ready')

        while rclpy.ok() and self.soc.isconnected:
            self.odom_callback(self.pub_odometry, self.soc.odometry)

...
    def main(argv=sys.argv[1:]):
        parser = argparse.ArgumentParser(formatter_class=argparse.
 ArgumentDefaultsHelpFormatter)
        parser.add_argument('-c', '--connection')
        parser.add_argument('-ro', '--robot')
        args = parser.parse_args(remove_ros_args(args=argv))
        rclpy.init(args=argv)
        odometry_node = KmpOdometryNode(args.connection,args.robot)

        while rclpy.ok():
            rclpy.spin_once(odometry_node)
        try:
            odometry_node.destroy_node()
            rclpy.shutdown()
        except:
            print(cl_red('Error: ') + "rclpy shutdown failed")
```

Listing 3: Snippet from KMP_odometry_node

Ros2 Laserscan node The ROS2 Laserscan node works similarly to the odometry node, but creates two publishers, one for each of the SICK laser scanners. Each publishers publishes to their own topic, namely "scan" and "scan_2". The laser data is sent with the LaserScan Massage, which is part of ROS's sensor massage package. Since large parts of the code are similar to odometry is it not shown in the report.

4.1.3 Middleware running on Raspberry Pi

The Middleware Raspberry Pi handles the communication between the Entity Raspberry Pi and the AAS Raspberry Pi. It is a key feature, translating the messages from the entity in order for the AAS to understand them and likewise back to the Entity. The communication between the middleware and entity is done with ROS2 publisher/subscriber-model. This communication builds upon DDS which does not support data traffic over the Internet as described in subsubsection 2.1.7. ROS2 it therefor not an option for communication between the AAS and Middleware Raspberry Pi. To handle this the machine-to-machine protocol OPC UA is used.

ROS2 hybrid The Middleware Raspberry Pi is a ROS2 program containing a hybrid node that combines both ROS2 and OPC UA functionality in one running instance. A snipped of the code shown in Listing 4 shows the ***main()*** function and a sensor subscriber classes. The function creates a OPC UA client and tries to connect to the server running on the AAS. When a connection is established the ROS publisher and subscriber node instances are created. To send commands to the robot commander nodes "lbr_command_node", "kmp_command_node" and the camera node "camera_node" publisher instances are made. To retrieve sensor data from the sensor nodes "kmp_laserscan_node" and " kmp_odometry_node" subscriber instances are made. Since there are two laser scanner two subscribers are made, one for each one. This architecture support full-duplex communication with simultaneous data transmission between the entity and AAS. This is done with a ROS2 publisher/subscriber massaging, OPC UA event handling and value updating. Since several node instances derive from the same ROS2 node, they need to be spin inside individual threads. To achieve this the ROS2 ***MultiThreadedExecutor*** is used. It creates a selected number of threads, which allows for multiple messages or events to be processed at the same time.

```
....
 class OdometrySubscriber(Node):
      def __init__(self, opcua_client):
        super().__init__('odometry')
        self.odom = opcua_client.get_node("ns=2;i=5")
        self.subscription = self.create_subscription(Odometry, 'odom',
   self.set_values_odom, 10)

      def set_values_odom(self, msg):
        self.odom.set_value(str(msg))
....
def main(argv=sys.argv[1:]):
    parser = argparse.ArgumentParser(formatter_class=argparse.
   ArgumentDefaultsHelpFormatter)
    parser.add_argument('-d', '--domain')
    args = parser.parse_args(remove_ros_args(args=argv))
    rclpy.init(args=None)
    isConnected = False
    opcua_client = Client("opc.tcp://" + args.domain + ":4841/freeopcua/
   server/")

    while not isConnected:
        try:
            opcua_client.connect()
            isConnected = True
            print("Successfully connected with OPC UA server on: " + args
   .domain + ":4841")
        except:
            print("Failed to connect on " + args.domain + " ... retrying"
   )
            sleep(1)
    root = opcua_client.get_root_node()
    obj = root.get_child(["0:Objects", "2:MyObject"])

    lbr_event = root.get_child(["0:Types", "0:EventTypes", "0:
   BaseEventType", "2:LBREvent"])
    kmp_event = root.get_child(["0:Types", "0:EventTypes", "0:
```

```
    BaseEventType", "2:KMPEvent"])
    camera_event = root.get_child(["0:Types", "0:EventTypes", "0:
    BaseEventType", "2:CameraEvent"])

    lbr_publisher = LBRPubSub(obj)
    kmp_publisher = KMPPubSub(obj)
    camera_publisher = CameraPubSub(obj)
    laser_subscriber1 = LaserSubscriber(opcua_client)
    laser_subscriber2 = Laser2Subscriber(opcua_client)
    odometry_subscriber = OdometrySubscriber(opcua_client)

    lbr_sub = opcua_client.create_subscription(100, lbr_publisher)
    lbr_handle = lbr_sub.subscribe_events(obj, lbr_event)
    kmp_sub = opcua_client.create_subscription(100, kmp_publisher)
    kmp_handle = kmp_sub.subscribe_events(obj, kmp_event)
    camera_sub = opcua_client.create_subscription(100, camera_publisher)
    camera_handle = camera_sub.subscribe_events(obj, camera_event)

    try:
        executor = MultiThreadedExecutor(num_threads=7)
        executor.add_node(lbr_publisher)
        executor.add_node(kmp_publisher)
        executor.add_node(camera_publisher)
        executor.add_node(laser_subscriber1)#Daniel
        executor.add_node(laser_subscriber2)#Daniel
        executor.add_node(odometry_subscriber)#Daniel
        executor.spin()
    finally:
    ...
    #destroy nodes and unsubscribe
    rclpy.shutdown()
```

Listing 4: Snippet from opcu_ros2_pubsub

It handles both commands sent from the AAS and sensor data from the entity. The middleware-pi creates the three classes OdomentrySubscriber, LaserSubscriber and Laser2Subscriber. These classes subscribes to the robots sensor data and updates the corresponding values of the sensor data object on the AAS.

4.1.4 AAS running on Raspberry Pi

The Pi-AAS consist of a Raspberry Pi that runs the software of the AAS and is highlighted in red in Figure 24. The AAS receives commands from the external interface with WebSocket and forwards them to the middleware. An OPC UA server is hosted on the AAS and communicates with both the client on the AAS-video and middleware over a public 5G. Python Flask is used for Back-End and the Front-End is handled with ReactJS. To store data the embedded database SQLite is used.

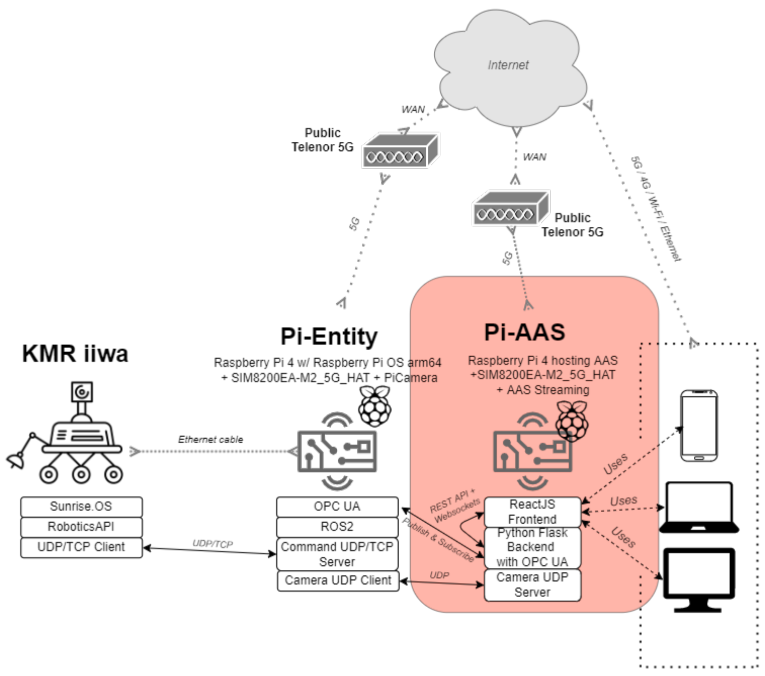


Figure 24: System design with Pi-AAS highlighted.

OPC UA Server The OPC UA server handles the communication with the middleware and AAS-client. To retrive sensor data from det middlware client an OPC UA object is created called "sensors". This object has three variables two for each of the laser scanners and one for the odometry data. These are all set to "writable" which makes it possible to update them from the client. Thus, these values wil be updated each time the middleware receives new sensor data. The object can easily be expanded with more variables if necessary. The OpcuaServer class is shown in Listing 5.

```
...
class OpcuaServer:
    def __init__(self, opcua_address, socketio, db):
        self.socketio = socketio
        self.db = db

        logging.basicConfig(level=logging.WARN)
        logger = logging.getLogger("opcua.server.internal_subscription")
        server = Server()
        server.set_endpoint("opc.tcp://" + opcua_address + "/freeopcua/
    server/")

        uri = "OPCUA_AAS_COMMUNICATION_SERVER"
        idx = server.register_namespace(uri)

        objects = server.get_objects_node()

        myobj = objects.add_object(idx, "MyObject")

        sensorobj = objects.add_object(idx, "sensors")
        self.laser1 = sensorobj.add_variable(idx,"laser1",0)
        self.laser2 = sensorobj.add_variable(idx,"laser2",0)
        self.odom = sensorobj.add_variable(idx,"odometry",0)
        self.laser1.set_writable()
        self.laser2.set_writable()
        self.odom.set_writable()

        status_node = myobj.add_method(idx, "update_status", self.
    update_status, [ua.VariantType.String], [ua.VariantType.Int64])

        lbrEvent = server.create_custom_event_type(idx, 'LBREvent')
        kmpEvent = server.create_custom_event_type(idx, 'KMPEvent')
        cameraEvent = server.create_custom_event_type(idx, 'CameraEvent')

        self.lbrEvgen = server.get_event_generator(lbrEvent, myobj)
        self.kmpEvgen = server.get_event_generator(kmpEvent, myobj)
        self.cameraEvgen = server.get_event_generator(cameraEvent, myobj)

        server.start()
...
```

Listing 5: Snippet from Opc UA server running on AAS

4.2 Edge Architecture [10]

The initial plan was to move big parts of the system over to a edge cloud provided by Telenor. The edge was supposed to be delivered inn the summer of 2021. After multiple delays, the focus of the task had to change. Instead of moving parts of the system to the

[10]This sections is heavily inspired by the related specialization project completed by the author during the fall of 2021 [3]

edge cloud further development of the system by adding functionalities was prioritized. The initial plan that show which parts were to be moved to the cloud and some of it's benefits are presented in this chapter. An overview of the system design where parts of it is in the edge cloud is shown in Figure 25.

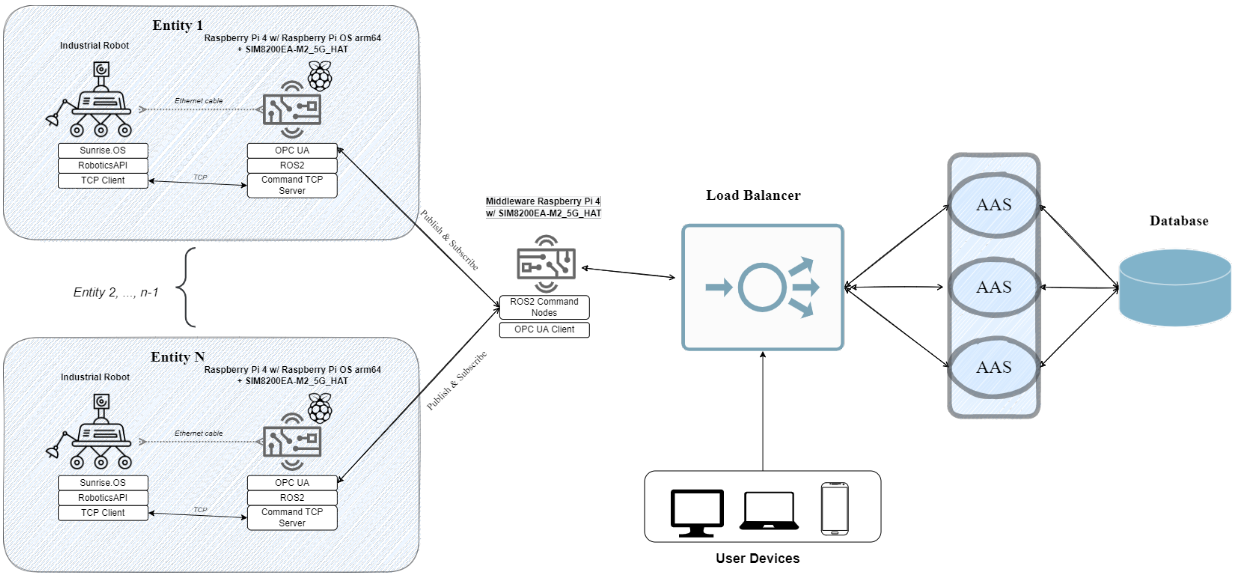


Figure 25: Edge system design

When the AAS is hosted on the Raspberry Pi it causes problems regarding capacity limits and it works as a single point of failure for the system. The new system design hosts the AAS in the edge cloud with the possibility to instantiate multiple instances of the AAS, increasing the redundancy and making the system more robust. The number of instances can be automatically fitted to the current workload making the system elastic and responsive to both up and down scaling.

To manage the workload between the different AAS instances a load balancer is used. The load balancer manages the incoming requests from user devices and redirect them further to the middleware and AAS instances. This is done to prevent any single instance from getting overloaded and possibly breaking down.

All the AAS instances are connected to one database. Although the new system design only shows one database, it is expected to incorporate a backup as well. The storage can be dynamically allocated based on changing demands.

The CloudBand Network Director (CBND) Orchestrator is Nokia's implementation of a Cross Domain Service Orchestrator (CDSO). It features Lifecycle Management using Closed-Loop Automation (CLA) and Virtual Network Functions (VNF) package management, Performance and self-Monitoring, Notifications and placement capabilities using a policy engine[44]. It is a system that is being delivered with Telenor's edge cloud. CBND can handle the load balancing as well as other edge cloud services.

4.3 Design Principles of Industry 4.0 [11]

There are four main design principles of industry 4.0, aiming to ensure data exchange and automation of manufacturing processes[45]. All of the main principles will be presented and the proposed design will be evaluated based on them. An overview of the principles is shown in Figure 26.

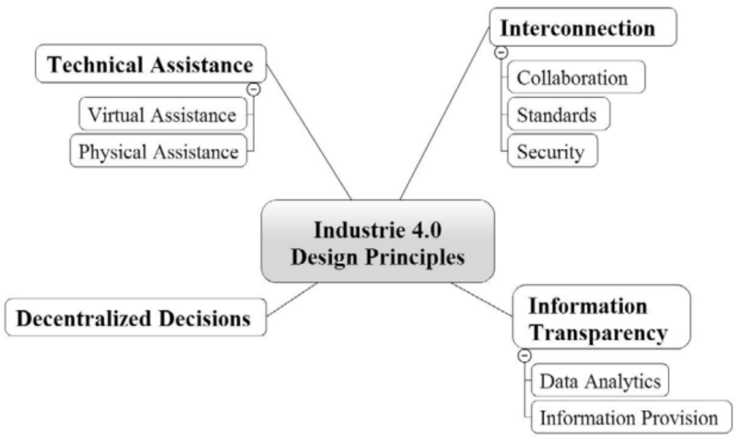


Figure 26: Industry 4.0 design principles[45].

4.3.1 Interconnection

With Internet of Everything (IoE) devices, people and machines are all connected. This information flow through the internet enables collaboration between people and machines to reach for common goals[46]. The collaboration within IoE can be categorized in three types: human-human, human-machine and machine-machine collaboration[47]. To enable flexible connection and collaboration between different devices from a wide range of manufacturers, communication standards are an important prerequisite[48]. The proposed architecture enables the three types of collaboration over 5g using acknowledged communication standards.

4.3.2 Information Transparency

The increasing number of interconnected devices and people facilitates the fusion between the physical and virtual world enabling greater information transparency[49]. To increase

[11]This sections is heavily inspired by the related specialization project completed by the author during the fall of 2021 [3]

the information transparency raw sensor data needs to be analyzed and embedded into the system making it available to all IoE participants[50]. Retrieving context-aware information from a substantial number of IoE participants facilitates for a more holistic decision making. When handling process-critical information, delivery of real-time data is crucial[51]. The fusion between the physical and virtual world can be represented as a digital twin of the lab, facilitating time sensitive computing and analyzes in the edge.

4.3.3 Decentralized Decisions

Information transparency between devices and people inside and outside of a production enables decentralized decisions to be made. Combined with interconnected decision-makers the value of local and global information can be used to increase the overall productivity[52]. Task should be done as autonomous as possible by the IoE participants. Higher level interference's should only be made in exceptions[45]. To enable high levels of autonomous methods like Artificial Intelligence (AI) and Machine Learning (ML) can be used. The implementation of such methods is out of the scope of this thesis, but the generalized principles of the architecture promote future implementations.

4.3.4 Technical Assistance

Previously the role of humans in factories have been machine operators. With the introduction of smart factories of industry 4.0 the role has shifted towards problem solving and decision-making. The increasing complexity of production require assistance systems to support humans. Information needs to be visualized in a comprehensive manner to ensure that humans can make informed decisions, and attend to urgent problems[50]. Today smartphones and tablets is heavily used as visualizing tool, connecting people and IoT[53]. In addition to visualized support, is also physical support by robots a key aspect of technical assistance. With advances in robotic can dangerous and unpleasant work be done by robot, but the coordination of a human robot environment require proper training of humans and intuitive and safe interaction from the robots[45].
The architecture supports an interface through multiple devices, but the sensor date is not displayed or connected to any alarms or events. Video stream from the robot can be used to detect problems from remote locations, but its usefulness is limited. When expanding the number of entities and increasing the complexity of the system further advancements in virtual assistance should be made. The current architecture support such future improvements. Physical support is enabled trough remote control of robots in real-time.

5 Discussion

This section discusses the system described in section 4 and how it is related to the objectives. It starts with subsection 5.1 which discusses the public 5G network. The communication protocols and the combination of them are discussed in subsection 5.2, subsection 5.3 and subsection 5.4. DT, the requirements it has reached, as well as its compliant with industry 4.0 is presented in subsection 5.5. Finally, the working method and future work is discussed in subsection 5.6 and subsection 5.7.

5.1 Public 5G network

All the communication has been transferred to a public 5G network delivered by Telenor, except the direct Ethernet connection between the robot and entity Raspberry Pi. A private 5G network was meant to be delivered by Telenor, but after multiple postponements it was decided to settle for their public 5G network. This network did not meet the requirements regarding response time and prevented the middleware end entity software to run on two different Raspberry Pis.

Considering that the system uses a public network which can be used by everyone with a Telenor Sim card, it will consequently have a performance that vary depending on the use of everyone connected. This can vary through the day, and random instances with abnormally high usage may occur. This can be critical in a time-sensitive system where there may be a response time requirement for given operations.

Although the network used is labeled as public 5G, it is actually a combination of both 5G and 4G. This greatly reduces the key performances of 5G and availability of the functionalities in URLLC, mMTC and eMBB. It switches between the two networks without any sign of pattern or system. This makes it difficult to test the 5G capabilities, since it is troublesome to determine which network is used at different times. Although the network does not deliver the performance requirement of a real-time system, it facilitates for an easy transition to Telenor private network with only minor changes. When the yet to be completed private Telenor network is delivered, it should be possible do transition to the new network by only changing the SIM cards and IP addresses. This will facilitate a faster development of a DT that is in line with Industry 4.0.

5.2 OPC UA

The communication between the middleware and AAS is handled with OPC UA. It does not depend on a private network and has a broad support for different platforms. This makes it possible to connect with other communication protocols than ROS2, making the system more flexible.

5.3 ROS 2

ROS2 facilitates for easy connection between multiple entities. Currently it is only one robot connected, due to the fact that ROS2 builds upon DDS, as described in subsubsection 2.1.7. This communication has to run on a private network. Because of the aforementioned delivery delays, a decision was made to run the software for the entity and middleware on the same Raspberry Pi. By choosing this solution, the software of the system could be developed with scalability in mind, even though the hardware and network available where limited. When a private network is delivered, the software can be moved to separate Raspberry Pis and the system can expand with new entities.
Ideally, ROS2 should have run directly on the KMR iiwa, which requires fewer parts and simplifes the task of adding or removing the robot, but the solution with a Raspberry Pi facilitates for faster setup and testing with other robots as well. Currently, the entity connected is the KMR iiwa, which consists of six nodes, but with ROS2, more entities can be easily implemented without major changes in the code. After entities have been added to the system, they can be connected and disconnected without the need to restart the system, thus greatly reducing the downtime of the system.

5.4 ROS 2 with OPC UA

The combination of ROS2 with OPC UA is proven as a highly capable method for solving several of the challenges regarding communication in an industrial setting. The advantage of the combined solution lies in the widespread support for the OPC UA and the connection management given by ROS2. With the current features it would have been advantageous to only use a united stack like zenoh for everything, but with different elements from different suppliers and further expansion in mind, the chosen solution is preferred. The use of several different protocols requires higher competence, but is seen as inevitable in a world with so many different providers and systems. The ability to combine systems and products is crucial for a sustainable system with a long service life.

5.5 Digital Twin

The AAS has been further developed to also retrieve sensor data from the KMR iiwa, and is thus one step closer of satisfy the requirements of a DT as described in subsubsection 2.1.4.

To achieve the requirements of a Level 1 DT, the sensor data should have been linked to alarms and occurrences. This could have been done, but would not contribute other than meeting the requirements of a DT. Instead, the sensor data is implemented as a loosely coupled service that can be exploited in further development or in a new system. The user can retrieve the sensor data at a high level without the knowledge of the low level process running in the background. This will simplify the process of developing a DT at NTNU Gløshaugen's Industry 4.0 laboratory or any other system that requires sensor data from the KMR iiwa.

All the four key design principles of industry 4.0 described in subsection 4.3 can be satisfied by a DT, thus making it an excellent tool in the factories of the future. The concept of a DT is still under development and accommodates a wide range of systems to be used to analyze raw sensor data and embedding it into to the system, thus increasing the information transparency. To meet this requirement a DT is an effective tool as it can collect data and transform it to high value information for the entities and personnel at the factories.

5.6 Working method

The original objective was to transfer the main parts of an AAS over to the edge cloud and use a private 5G network. Due to several delays, the objective was changed to further develop and re-engineer the existing AAS.An alternative to this could have been to work out a more thorough solution on how the system could be transferred to the edge cloud. The reason this was not done was because it would not be possible to test along the way and one would most likely end up with a faulty solution. By further developing the system, it was possible to add functions that work with certainty, which would most likely have to be developed anyway and thus making a greater contribution to the development of a DT. A weakness with the system is that the robot needs a person to start the program from the robot. This should be fixed to make the system more autonomous, but the scope of this thesis is not to make a specific solution for the KMR iiwa, but rather a general system that can be adapted to different devices.

The sensor data is not currently stored. This is mainly because there is limited storage space on the Raspberry Pis. It is also not clear what the data is to be used for and it is therefore not known which sampling frequency or data should be stored. If there is a desire to store the data, the SQLite database can be expanded with a new sensor data table. Here it is recommended to enter the sensor data values as a JSON file, so that the same table can be used for different sensors. A foreign key should also be added to link sensor data table to the corresponding entity in the already existing table.

5.7 Future work

The main focus in future work should be to replace the public 5G with a private 5G and transfer the system to the edge cloud. In order to do this, it is crucial that the private network and edge cloud is delivered by Telenor. Without this, the development of a DT will be delayed considerably. The only further developments that would have been meaningful without these deliveries concern implementation of individual systems which can be added to the main system. The danger with such a development will be that large parts can be developed without being able to connect to the main system. It is therefore recommended to wait with the development of the DT until the network and edge cloud is delivered.
When a private network is established the entity and middleware software can be separated, and run on two separate Raspberry Pis. This enables the connection of several devices which can confirm the support of multiple entities with a working concept. New network tests should then be conducted, to see if the requirements of industry 4.0 regarding response time are satisfied.

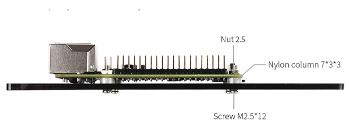
6 Conclusion

Previous testing of 5G has shown promising results [2], but the technology is still under development. Equal to many other new technologies,it has shown that the integration and widespread support of new technologies in the industry can be a tedious and time consuming process. It is predicted that 5G plays a critical part in industry 4.0. Unfortunately, multiple delays of the delivering of a private 5G network in the industry lab has prevented this key component of the system. Instead, a public 5G network has been used. This prevents the use of DDS communication which is necessary to connect multiple entities to the middleware. An architecture that was developed by Andreas and Mathias has been re-engineered and physically implemented at NTNU Gløshaugen's Industry 4.0 laboratory. All the communication has been transferred to a public 5G, which easily can be converted to a private 5G network when it gets delivered to the lab. The system has received the added functionality of retrieving sensor data from the KMR iiwa. This opens up for many new capabilities and can significantly reduce the development time of a DT at the laboratory. The sensor data can be used as a stand alone service in the edge cloud that gives the position of the KMR iiwa or other Automated guided vehicle (AGV) and continuously maps the industrial lab. Instead of the client using the DT, having to understand the low level retrieval of sensor data, it can be delivered as a high level function where the client can retrieve the data with a single function call which returns the positional data as well as meta data, such as the accuracy and refresh rate.
The method used to retrieved the data is in accordance with the design principles of industry 4.0, making it highly applicable in new systems and factories in the future. The modular design is loosely coupled to other parts of the system, making it highly scalable and can therefor easily be added or further developed to a bigger system. The communication protocols used, has broad support in the industry and is predicted to be critical parts of industry 4.0.
The retrieval of sensor data from KMR iiwa and it's design principals are applicable for retrieving data from other devices with only minor changes. The method can be replicated and other useful sensors and devices can be added to the system.
The developed system can work as a foundation for the implementation of a DT in the edge cloud. The thorough explanation of the layout and its design facilitates further development, and can be utilized when a local system eventually is up and running at the laboratory.

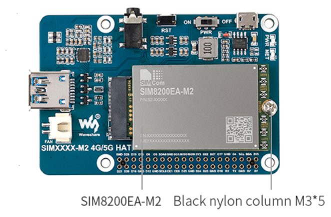
Appendices

A 5G HAT assembly process

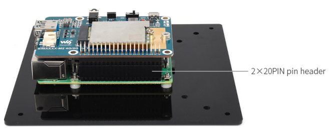
B Sequence diagrams



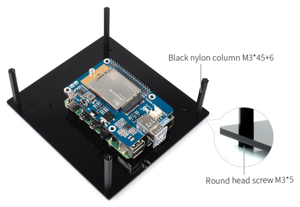
(a) Step 1: Install the Raspberry Pi into the base.



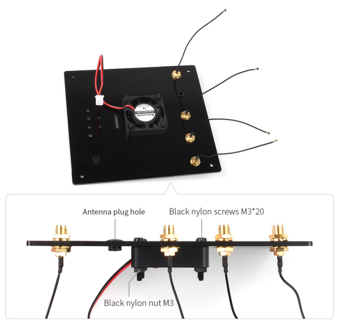
(b) Step 2: Install the SIM8200-m2 mainboard into the SIM82000-M2 5G HAT base board.



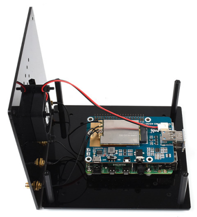
(c) Step 3: Install SIM8200-M2 5G HAT base into the Raspberry Pi.



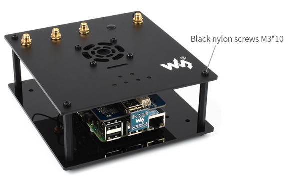
(d) Step 4: Install nylon columns in the base board.



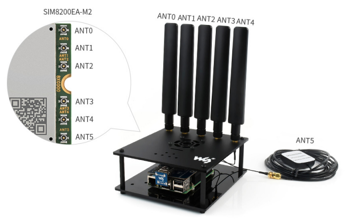
(e) Step 5: Install cooling fan and antenna adapter cables into upper cover board.



(f) Step 6: Connect the antenna adapter cables to the SIM82000-M2 main board and connect the cooling fan to the 5G HAT base board.



(g) Step 7: Assemble the top cover and fix with screws.



(h) Step 8: Install the external antennas.

Figure 27: 5G HAT assembly process [9].

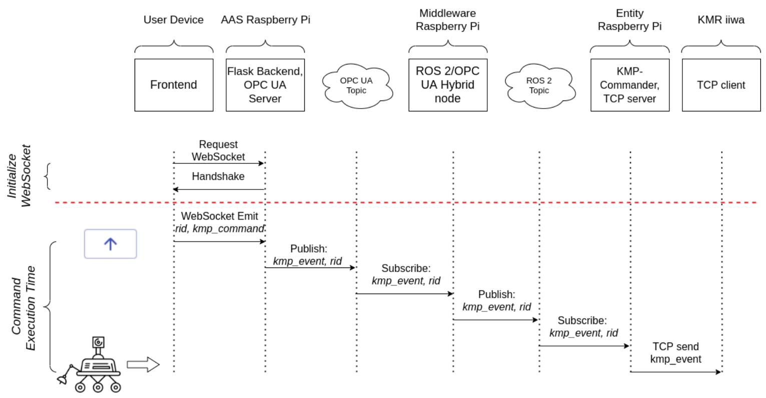


Figure 28: Sequence diagram of the data flow when sending a command from the AAS frontend to a KMR iiwa [9].

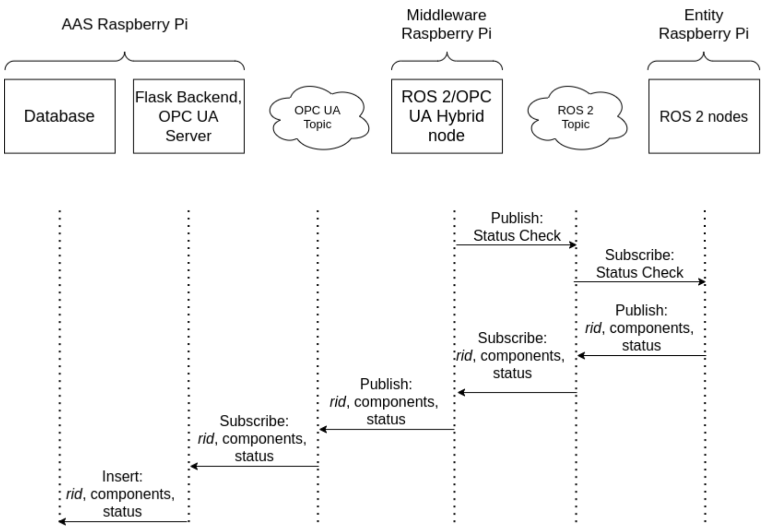

AAS Raspberry Pi
Middleware Raspberry Pi
Entity Raspberry Pi
Database
Flask Backend, OPC UA Server
OPC UA Topic
ROS 2/OPC UA Hybrid node
ROS 2 Topic
ROS 2 nodes
Publish: Status Check
Subscribe: Status Check
Publish: rid, components, status
Subscribe: rid, components, status
Publish: rid, components, status
Subscribe: rid, components, status
Insert: rid, components, status

Figure 29: Sequence diagram of the data flow when discovering a new robot [9].

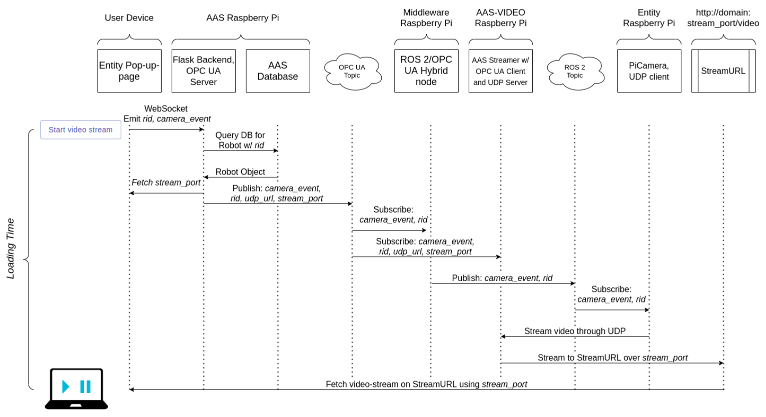

User Device
AAS Raspberry Pi
Middleware Raspberry Pi
AAS-VIDEO Raspberry Pi
Entity Raspberry Pi
http://domain: stream_port/video
Entity Pop-up-page
Flask Backend, OPC UA Server
AAS Database
OPC UA Topic
ROS 2/OPC UA Hybrid node
AAS Streamer w/ OPC UA Client and UDP Server
ROS 2 Topic
PiCamera, UDP client
StreamURL
Start video stream
Loading Time
WebSocket Emit rid, camera_event
Query DB for Robot w/ rid
Robot Object
Fetch stream_port
Publish: camera_event, rid, udp_url, stream_port
Subscribe: camera_event, rid
Subscribe: camera_event, rid, udp_url, stream_port
Publish: camera_event, rid
Subscribe: camera_event, rid
Stream video through UDP
Stream to StreamURL over stream_port
Fetch video-stream on StreamURL using stream_port

Figure 30: Sequence diagram of the video stream implementation [9].

References

[1] Infrastructure Public Private Partnership, TH 5g: *5g-vinni*. https://5g-ppp.eu/5g-vinni/, 2022.

[2] solutions, 5G: *Explore 5g solutions*. https://5gsolutionsproject.eu/explore/.

[3] Loken, Daniel René: *Combining industry 4.0 and digital twins with 5g connectivity to integrate robots in digital production factories*. Specialization project, 2021. Unpublished.

[4] Hunt, Graig: *TCP/IP Network Administration*. O'REILY & Associates, Inc, 2002.

[5] *Tcp/ip | computerworld*. https://www.computerworld.com/article/2593612/tcp-ip.html. (Accessed on 12/06/2021).

[6] Corporation, IBM: *Tcp/ip tcp, udp, and ip protocols*. https://www.ibm.com/docs/en/zos/2.2.0?topic=internets-tcpip-tcp-udp-ip-protocols, 2015. (Accessed on 12/06/2021).

[7] Cloudflare: *What is the osi model?* https://www.cloudflare.com/learning/ddos/glossary/open-systems-interconnection-model-osi/. (Accessed on 12/06/2021).

[8] Kumar, Santosh and Sonam Rai: *Survey on transport layer protocols: Tcp & udp*. International Journal of Computer Applications, 46(7), 2012.

[9] Arnholm, Andreas Chanon and Mathias Neslow Henriksen: *Combining industry 4.0 and 5g connectivity with robots in digital production factories*. Master's thesis in engineering & ict, NTNU, June 2021.

[10] Li, Zhengmao, Xiaoyun Wang, and Tongxu Zhang: *5G+ How 5G Change the Society*. Springer, 2019.

[11] Wang, Xiwen and Longxiang Gao: *When 5G Meets Industry 4.0*. Springer, 2020.

[12] *What is edge computing? everything you need to know*. https://searchdatacenter.techtarget.com/definition/edge-computing. (Accessed on 11/27/2021).

[13] Finne, P.H.P., I. Oppen D.R. Løken, and S. Ånestad: *Digital tvilling:for en bærekraftig industri*, 2021.

[14] Equinor: *With a little help from my digital twin»: Digital magi er i ferd med å bli en del av hverdagen.* https://www.equinor.com/no/magazine/echo-equinors-digital-twin.html, (Hentet: 20.07.21).

[15] Connected Industries, 5G Alliance for and Automation: *Using digital twins to integrate 5g into production networks.* Technical report, 5G-ACIA, 2021.

[16] LEEUW, VALENTIJN DE: *Concepts and applications of the i4.0 asset administration shell.* https://www.arcweb.com/blog/concepts-applications-i40-asset-administration-shell, AUGUST 2019. (Accessed on 11/28/2021).

[17] Erdal, Tantik and Anderl Reiner: *Integrated data model and structure for the asset administration shell in industrie 4.0.* Procedia CIRP, 60:86–91, December 2017.

[18] Foundation, OPC: *Unified architecture.* https://opcfoundation.org/about/opc-technologies/opc-ua/, 2021.

[19] FOUNDATION, ECLIPSE: *Milo.* https://projects.eclipse.org/proposals/milo, (Hentet: 12.11.21).

[20] Profanter, Stefan, Ayhun Tekat, Kirill Dorofeev, Markus Rickert, and Alois Knoll: *Opc ua versus ros, dds, and mqtt: Performance evaluation of industry 4.0 protocols.* In *2019 IEEE International Conference on Industrial Technology (ICIT)*, pages 955–962, February 2019.

[21] Connected Industries, 5G Alliance for and Automation: *Integration of 5g with time-sensitive networking for industrial communications.* Technical report, 5G-ACIA, Lyoner Strasse 9, February 2021.

[22] FOUNDATION, DDS: *What is dds.* https://www.dds-foundation.org/what-is-dds-3/, (Hentet: 15.11.21).

[23] OMG: *The real-time publish-subscribe protocol (rtps) dds interoperability wire protocol specification.* Technical report, OBJECT MANAGEMENT GROUP, 2018.

[24] EPROSIMA: *Rtps introduction.* https://www.eprosima.com/index.php/resources-all/whitepapers/rtps, (Hentet: 17.11.21).

[25] Eclipse: *Eclipse zenoh.* https://projects.eclipse.org/proposals/eclipse-zenoh, (Hentet: 19.11.21).

[26] Zenoh: *Key concepts*. https://zenoh.io/docs/getting-started/key-concepts/, (Hentet: 19.11.21).

[27] Mathieu, Labbé and Michaud François: *Rtab-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation*. Journal of Field Robotics, 36(2):416–446, 2019. https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21831.

[28] KUKA: *Kmr iiwa*, 2021. https://www.kuka.com/en-ch/products/mobility/mobile-robots/kmr-iiwa.

[29] GmbH, KUKA Deutschland: *Mobile Robots-KMR iiwa omniMove-Mobile Industrial Robot System-Assembly and Operating Instructions*, ma kmr iiwa omnimove v5 edition, 2021.

[30] SICK AG, Erwin-Sick-Str.1, 79183 Waldkirch, Germany: *S300 Safety laser scanners*, December 2021.

[31] SICK: *Safety and more - sick provides protection and navigation data for kuka's kmr iiwa*, 2016.

[32] KUKA: *Lbr iiwa*, 2021. https://www.kuka.com/en-ch/products/robotics-systems/industrial-robots/lbr-iiwa.

[33] Robotics, Michal: *Wireless teach pendants for robots*. https://roboticsbook.com/wireless-teach-pendants-for-robots/. (Accessed on 12/05/2021).

[34] Heggem, Charlotte and Nina Marie Wahl: *Configuration and control of kmr iiwa with ros2*. Department of mechanical and industrial engineering, NTNU, December 2019.

[35] *Raspberry Pi 4 Computer Model B*, 2021.

[36] *Latest raspberry pi 4 model b with ram | fruugo no*. https://www.fruugo.no/latest-raspberry-pi-4-model-b-with-ram/p-56029790-113994948?language=en&ac=croud&gclid=CjwKCAiAhreNBhAYEiwAFGGKPCsMfTYR5VY-qn6Dc_fDO45UA2iaKmvRWaHsOOO5k7-TJHkwpqgO9hoCgV4QAvD_BwE. (Accessed on 12/06/2021).

[37] Halfacree, Gareth: *Benchmarking the raspberry pi 4*. https://medium.com/@ghalfacree/benchmarking-the-raspberry-pi-4-73e5afbcd54b, (Hentet: 13.11.21).

[38] Limited, SIMCom Wireless Solutions: *SIM8200EA-M2 Hardware Design*, v1.03 edition, 2020.

[39] Waveshare: *Sim8200ea-m2 5g hat for raspberry pi, 5g/4g/3g support, snapdragon x55, multi mode multi band.* https://www.waveshare.com/product/sim8200ea-m2-5g-hat.htm. (Accessed on 12/05/2021).

[40] Vishal, Mithul, Ajay, and Chaitali: *Understanding at commands - vmac gps/gsm.* https://sites.google.com/site/vmacgpsgsm/understanding-at-commands. (Accessed on 12/05/2021).

[41] KUKA: *Kuka sunrise.os.* https://www.kuka.com/en-de/products/robot-systems/software/system-software/sunriseos, (Hentet: 11.11.21).

[42] *KUKA Sunrise.OS 1.11 KUKA Sunrise.Workbench 1.11*, 2016.

[43] ANGERER, Andreas, Alwin HOFFMANN, Andreas SCHIERL, Michael VISTEIN, and Wolfgang REIF: *Robotics api: Object-oriented software development for industrial robots.* Journal of Software Engineering for Robotics, May 2013.

[44] Margolin, Udi: *D2.2aspecifications and design of cdso plugins.* Technical report, 5G-SOLUTIONSfor European Citizens, 2019.

[45] Hermann, Mario, Tobias Pentek, and Boris Otto: *Design principles for industrie 4.0 scenarios.* In *2016 49th Hawaii International Conference on System Sciences (HICSS)*, pages 3928–3937, 2016.

[46] Giusto, D., A. Iera, G. Morabito, and eds. L. Atzori: *The Internet of Things.* Springer, 2010.

[47] Nieto De Santos, Francisco Javier and Sergio Garcia Villalonga: *Exploiting local clouds in the internet of everything environment.* In *2015 23rd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, pages 296–300, 2015.

[48] Zuehlke, Detlef: *Smartfactory—towards a factory-of-things.* Annual Reviews in Control, 34(1):129–138, 2010, ISSN 1367-5788. https://www.sciencedirect.com/science/article/pii/S1367578810000143.

[49] H., Albach and Meffert H.amd Pinkwart A.and Reichwald R.: *Management of Permanent Change.* Springer Gabler, Wiesbaden, 2015.

[50] Gorecky, Dominic, Mathias Schmitt, Matthias Loskyll, and Detlef Zühlke: *Human-machine-interaction in the industry 4.0 era.* In *2014 12th IEEE International Conference on Industrial Informatics (INDIN)*, pages 289–294, July 2014.

[51] Mario Hermann, Tobias Pentek, Boris Otto: *Design principles for industrie 4.0 scenarios.* Technical report, Hawaii International Conference on System Sciences, 2016.

[52] Malone, T. W.: *Is 'empowerment' just a fad? control, decision-making, and information technology.* BT Technology Journal, 17(4):141–144, oct 1999, ISSN 1358-3948. https://doi.org/10.1023/A:1009663512936.

[53] Miranda Carpintero, Javier, Niko Mäkitalo, Jose Garcia-Alonso, Javier Berrocal, Tommi Mikkonen, Carlos Canal, and Juan Murillo: *From the internet of things to the internet of people.* Internet Computing, IEEE, 19:40–47, March 2015.