

Prospektering anlægget med IIoT

En udforskning af den værdiskabende anvendelse af IIoT for vertikal
dataintegration i danske SMV'er

Raphael Peter Harrow-Hodgkinson

ENMA0920 – Energimanagement

Prospecting for Data with IIoT: An Exploration of the Value-Creating Application of IIoT for Vertical Data Integration for Danish SMEs

Dette projekt søger at skabe et blueprint for danske fremstillings SMV'er, som går i gang med udforskende projekter om Industrial Internet of Things med formål om vertikal dataintegration. Projektet inspireres af igangværende forskning hos Aalborg Universitets Center for Industrial Production, og udføres med brug af deres Smart Production Lab anlæg. Teknologier såsom Raspberry Pi, Node-RED og OPC UA tager en central rolle i udviklingen af en demonstrator, som opsamler, lagrer og fremstiller procestidscyklusdata, og som gør dataene tilgængelige for videre analyse via fil-eksport. Enheden er derefter vurderet med hensyn til gældende Manufacturing Operations Management modeller og Energimanagements aktiviteter, og potential retninger for yderligere forskning og udviklinger er identificeret.

This project seeks to provide a blueprint for Danish manufacturing SMEs embarking on exploratory Industrial Internet of Things projects with a view toward vertical data integration. It is inspired by the ongoing research at Aalborg University's Centre for Industrial Production and is carried out with use of their Smart Production Lab laboratory. Technologies such as Raspberry Pi, Node-RED and OPC UA take a central role in the development of a demonstrator-device which gathers process cycle time data, stores and presents it locally, and also makes said data available for further analysis via file-export. The device is then assessed with regard to Manufacturing Operations Management models and Energy Management activities, and potential directions for future research and development are identified.

Indholdsfortegnelse

Figurliste.....	5
0. Introduktion	8
1. Problemstilling	9
1.1 Problemformulering.....	9
1.2 Projektets omfang og afgrænsninger	9
1.2.1 Løsningen skal være værdiskabende	9
1.2.2 Løsningens bestanddele skal være tilgængelige.....	9
1.2.3 <i>Open source</i> fortrækkes.....	10
2. Teori	11
2.1 Begreber.....	11
2.1.1 AAUs Smart Production Lab.....	11
2.1.2 Hvad er <i>IIoT</i> ?	13
2.1.3 <i>ISA-95</i>	13
2.1.4 OEE, Performance & Cyklus Tid	14
2.2 Teknologier	15
2.2.1 <i>OPC UA</i>	15
2.2.2 <i>MQTT</i>	15
2.2.3 <i>Raspberry Pi</i>	16
2.2.4 <i>Node-RED</i>	16
2.2.5 <i>RFID</i>	17
2.2.6 <i>SIA-Connect Gateway</i>	17
2.2.7 <i>SQLite</i>	17
2.2.8 <i>UAExpert</i>	18
3. Metode.....	19
3.1 Udgangspunkt.....	19
3.1.1 Opgavens rammer.....	19

3.1.2 Planlægning.....	19
3.1.3 Implementering i <i>Node-RED</i>	22
3.2 Yderligere udviklinger	26
3.2.1 Problemer med SIA-Connect og hardwareskift	26
3.2.2 <i>OPC UA-client</i> via <i>Node-RED</i>	26
3.2.3 Forbedringer til systemets funktion.....	29
4. Analyse	35
4.1 Vurdering af den vertikale dataintegration	35
4.1.1 Løsningens fysisk forbindelse med anlægget	35
4.1.2 Vurdering af den vertikale datastrømme	36
4.2 Løsnings funktionelanalyse	36
4.2.1 Løsningens placering på det funktionelle hierarki	36
4.2.2 Funktionalitet mht. ISA-95.....	37
4.2.3 Funktionalitetsanalyse med hensyn til <i>IIoT</i>	38
4.2.4 Yderligere udviklinger	38
4.3 Analyse af løsningen mht. energimanagement og bæredygtighed.....	38
4.3.1 Løsningens nuværende betydning for energimanagement.....	39
4.3.2 Yderligere Potential	39
5. Konklusioner og Perspektivering	40
5.1 Konklusioner	40
5.1.1 Hovedspørørgsmål	40
5.1.2 Underspørørgsmål.....	40
5.2 Perspektivering af projektet	41
6. Litteraturliste	42
7. Bilag.....	47
7.1 <i>OPC UA Servers</i> og Variablers <i>Identifiers</i>	47
7.2 Resultater af funktionelle afprøvninger.....	49
7.2.1 Udvikling af <i>OPC UA Clients</i>	49

7.2.2 Første funktionel afprøvning på udvidede <i>OPC UA-Clients</i>	50
7.2.3 Anden funktionel afprøvning på udvidede <i>OPC UA-Clients</i>	52
7.2.4 Tredje funktionel afprøvning på udvidede <i>OPC UA-Clients</i>	55
7.2.5 Funktionelafprøvning af 50ms forsinkelse på <i>OrderID Read</i> -kommando	57
7.2.6 Funktionelafprøvning af 75ms forsinkelse på <i>OrderID Read</i> -kommando	58
7.2.7 Funktionelafprøvning af 100ms forsinkelse på <i>OrderID Read</i> -kommando	59
7.2.8 Enkelpalle funktionelafprøvning af 500ms forsinkelse på <i>OrderID Read</i> -kommando	60
7.2.9 Multipalle funktionelafprøvning af 500ms forsinkelse på <i>OrderID Read</i> -kommando.....	61
7.2.10 Multipalle funktionelafprøvning af 550ms forsinkelse på <i>OrderID Read</i> -kommando.....	65

Figurliste

Figur 1 Diagram af den Smart Production Lab med piler, der illustrerer mulige transportveje for pallerne.....	11
Figur 2 Foto af STPLC_08, med transportørdirektion og beliggenhederne af RFID-Læser, Nærhedssensorer og Pneumatisk Stopper fremhævet.	12
Figur 3 Foto af STPLC_10, med transportørdirektion og beliggenhederne af RFID-Læser, Nærhedssensorer og Pneumatisk Stopper fremhævet	12
Figur 4 Den funktionelle hierarkimodel udgivet i DS/EN 62264-1 [23]	14
Figur 5 MQTTs Publish/Subscribe Arkitektur [33].	16
Figur 6 Funktionelle arkitektur af løsningen til den oprindelige opgave.....	19
Figur 7 Semantikmodeller til databasens design.....	20
Figur 8 Flow for konfigurering af databasen og initialisering af tabeller.....	22
Figur 9 Konfigurering af SQLite3 databasnode.	22
Figur 10 Konfigurering af inject-node for initialisering af procestabellen i databasen.	23
Figur 11 (til venstre) - a) Konfigurering af MQTT-subscription til xCarrierAvailable. (til højre) - b) Konfigurering af MQTT-subscription til OrderID (udiONo).....	23
Figur 12 Udløsning af beskeddele og indstilling af msg.topic egenskaber.	24
Figur 13 Kombinering og sortering af beskedobjekter, og genindstilling af msg.topic-egenskab.....	24
Figur 14 Flow for fremvisning af optagede procescyklustidsdata og ordresøgningsfunktion.....	25
Figur 15 Konfigurering af template-noden til fremvisning af procescyklustider i tabelformat.....	25
Figur 16 (ovenfor) – a) OPC UA-Client flow for 'almindelige' stationer på det Smart Production Lab anlæg.	

(nedenfor) – b) Den endelige OPC UA-Client sub-flow implementerede i løsningen.

26

Figur 17 OPC UA-client flow for station STPLC_01 på det Smart Production Lab anlæg.....	27
Figur 18 OPC UA Item-node konfigureringside for xCarrierAvailable variabellen for STPLC_10.	27
Figur 19 Konfigureringside af OPC UA Client-node for subscription til STPLC_10s OPC UA-Server.....	28
Figur 20 Konfigureringside af MQTT-Out node for Infeed/xCarrierAvailable for STPLC_01.....	29
Figur 21 Intermediær data-flow for STPLC_10.	29
Figur 22 Endelig data-flow for STPLC_10.	30
Figur 23 Intermediær data-flow for STPLC_01.	30
Figur 24 (til venstre) - a) Konfigureringside af forbindelsen mellem Link In- og Link Out-noder. (til højre) - b) Forbindelser mellem Link In-noder og OPC UA Client sub-flows.	30
Figur 25 Flow for STPLC_08.....	31
Figur 26 Konfigureringside af function-node til midlertidig lagring af OrderID.	31
Figur 27 Ændret flow for fremvisning af procescyklustidsdata med opdateringsknap.....	32
Figur 28 Konfigureringside af den opdateringsknap-node.....	32
Figur 29 Flow for eksport af procesdata som CSV-fil.....	33
Figur 30 Konfigureringside af: (til venstre) - a) CSV-node, (centrum) - b) Change-node, (til højre) - c) Write File-node.	34
Figur 31 Placering af løsningen i forhold til de andre Smart Production Lab systemkomponenter....	35
Figur 32 Den funktionelle model [23].	37
Figur 33 Debug output fra funktionelafprøvning af OPC UA Clients	49
Figur 34 Debug Output fra første funktionel afprøvning af udvidede OPC UA Client-noder.	50
Figur 35 UI Output fra fra første funktionel afprøvning af udvidede OPC UA Client-noder.....	51
Figur 36 Debug Output fra anden funktionel afprøvning af udvidede OPC UA Client-noder.	52
Figur 37 UI Output fra anden funktionel afprøvning af udvidede OPC UA Client-noder med brug af ordresøgningsfunktion.....	53
Figur 38 UI Output fra anden funktionel afprøvning af udvidede OPC UA Client-noder	54
Figur 39 Debug Output fra tredje funktionel afprøvning af udvidede OPC UA Client-noder.....	55
Figur 40 UI Output fra tredje funktionel afprøvning af udvidede OPC UA Client-noder.....	56
Figur 41 (ovenfor) - a) Debug Output og (nedenfor) - b) UI Output for funktionelafprøvning af 50ms forsinkelse på OrderID Read-kommando.	57
Figur 42 (ovenfor) - a) Debug Output og (nedenfor) -b) UI Output for funktionelafprøvning af 75ms forsinkelse på OrderID Read-kommando. (Det dækkede område er repeterede data fra 2. kolonne)	58

Figur 43 (ovenfor) - a) Debug Output og (nedenfor) - b) UI Output for funktionelafprøvning af 100ms forsinkelse på OrderID Read-kommando.	59
Figur 44 (ovenfor) - a) Debug Output og (nedenfor) - b) UI Output for funktionelafprøvning af 500ms forsinkelse på OrderID Read-kommando med én palle.....	60
Figur 45 Debug Output for første funktionelafprøvning af 500ms forsinkelse på OrderID Read-kommando med flere paller.	61
Figur 46 UI Output for første funktionelafprøvning af 500ms forsinkelse på OrderID Read-kommando med flere paller.....	62
Figur 47 Debug Output for anden funktionelafprøvning af 500ms forsinkelse på OrderID Read-kommando med flere paller.	63
Figur 48 UI Output for anden funktionelafprøvning af 500ms forsinkelse på OrderID Read-kommando med flere paller.....	64
Figur 49 Debug Output for funktionelafprøvning af 550ms forsinkelse på OrderID Read-kommando med flere paller. (Det dækkede område er repeterede data fra 4. kolonne)	65
Figur 50 UI Output for funktionelafprøvning af 550ms forsinkelse på OrderID Read-kommando med flere paller.....	66

0. Introduktion

For nylig har der været en stor interesse for *Industrial Internet of Things (IIoT)* i konteksten af Industri 4,0. En undersøgelse fra Teknologiske Institut viser at to ud af tre danske virksomheder, der producerer energiprodukter, enten har gang i eller forventer at bruge *IoT* i deres produkter eller produktion inden de næste 4-5 år [1]. Derudover synes 85% af de svarende virksomheder at *IoT* har enten nogen (55%) eller afgørende (30%) betydning for at styrke deres konkurrenceevnen i fremtiden. Imidlertid er den største enkelte barriere ift. indførelse af *IoT* i disse forretninger manglen for de nødvendige kompetencer (15%), med lovgivning, persondataforordning og manglende standarder (10%), og mangel for viden om mulighederne (6%) nævnte som yderligere udfordringer. Så kan situationen omkring udbredelsen af *IoT* i danske virksomheder med energiprodukter karakteriseres som en bred accept af vigtigheden af *IoT* for fremtidig konkurrenceevne, med væsentlige mangler for de nødvendige færdigheder, viden og lovmæssige rammer at indføre det.

Manufacturing Academy of Denmark (MADE) er et nationalt samarbejde mellem danske virksomheder, universiteter og GTS-institutter med hovedmål om at skabe en platform for anvendt forskning, udvikling og innovation i danske produktionsvirksomheder [2], [3]. Aalborg Universitet (AAU) er ét af de universiteter der deltager i samarbejde om MADE gennem Center for Industrial Production (CIP), som fungerer som et center for forskning og udvikling mht. styrkelse af den danske fremstillingsindustri [4]–[6]. Denne forskning og udvikling udføres gennem AAUs Smart Production Lab via flere forskeruddannelser [7] og projekter, såsom Innovation Factory North [8]. Ét af de forskningstemaer ved Innovation Factory North er *IIoT* og forskerne hos Smart Production Lab har skrevet flere artikler om emnet [9]–[12].

Flere af disse artikler henviser til demonstratorer, der findes hos Smart Production Lab, og som bruges både til forskning og til demonstration af begreber og teknologier til industriprofessionelle og -ledere fra hele Danmark. Dette projekt tager udgangspunkt i udviklingen af én af disse *IIoT* demonstratorer hos AAUs Smart Production Lab.

1. Problemstilling

Projektet tager udgangspunkt i udviklingen af en demonstrator for opfindelser fra Ph.d.-forskning om vertikale dataintegration for smart produktion i danske SMV'er [13]. For at bryde igennem de opfattede barrierer mod anvendelse af IIoT-løsninger i danske produktionsvirksomheder, er der behov for en demonstrator, der kan vise industriprofessionelle og -ledere noget af de anvendelsesmuligheder for IIoT-teknologier og -begreber. Den forskning og udvikling af sådan en IIoT-løsning udgør hovedopgaven i dette projekt.

1.1 Problemformulering

Hovedopgaven ved projektet er udviklingen af en IIoT-løsning til demonstration af IIoT-begreber og teknologier for industriprofessionelle og -ledere. Selvom opgaven er udviklingsfokuseret, skal projektet tage formen af et forskningsprojekt. Så skal projektet ledes med følgende hovedspørgsmål:

'Hvordan kan IIoT-begreber og -teknologier demonstreres hos AAUs Smart Production Lab?'

Derudover stilles der nogle underspørgsmål, som hjælper med at uddybe projektet:

'Er ISA-95/IEC 62264 relevante til sikke en demonstratorenhed?'

'Hvad er den eventuelle betydning for energimanagement af sikke en enhed?'

1.2 Projektets omfang og afgrænsninger

Hovedspørgsmålet er alt for brede at kunne svare på det inden projektets rammer. Derfor er det nødvendigt at definere projektets omfang med passende afgrænsninger.

1.2.1 Løsningen skal være værdiskabende

Demonstrantens hovedmål er at demonstrere noget af de muligheder med IIoT-begreber og -løsninger til en bestemt interessegruppe, med formidling af viden om de muligheder med IIoT til formål. Derfor er det vigtigt at interessegruppen kan se værdien af disse tiltag, og så skal demonstratoren have fokus på værdiskabelse ift. denne interessegruppe.

1.2.2 Løsningens bestanddele skal være tilgængelige

Én af de barrierer mod indførelse af IIoT der blive nævnte i Teknologiske Instituts undersøgelse om IoT var mangel for relevante færdigheder til anvendelsen af de forskellige teknologier og begreber [1]. Så skal de teknologier, der bruges, og de begreber, der demonstreres, være tilgængelige for interessegruppen – dvs. at de skal være enten i almindelige brug hos produktionsvirksomheder, eller have en relativ kort indlæringskurv.

1.2.3 *Open source* fortrækkes

I takt med forskningen ved CIP skal *open source*-teknologier og -løsninger foretrækkes over proprietære alternativer [10], [11]. Dette skyldes flere grunde, men det er primært pga. fordele ift. fleksibilitet, modulærhed, sikkerhed og mindre omkostninger mht. udvikling [11].

2. Teori

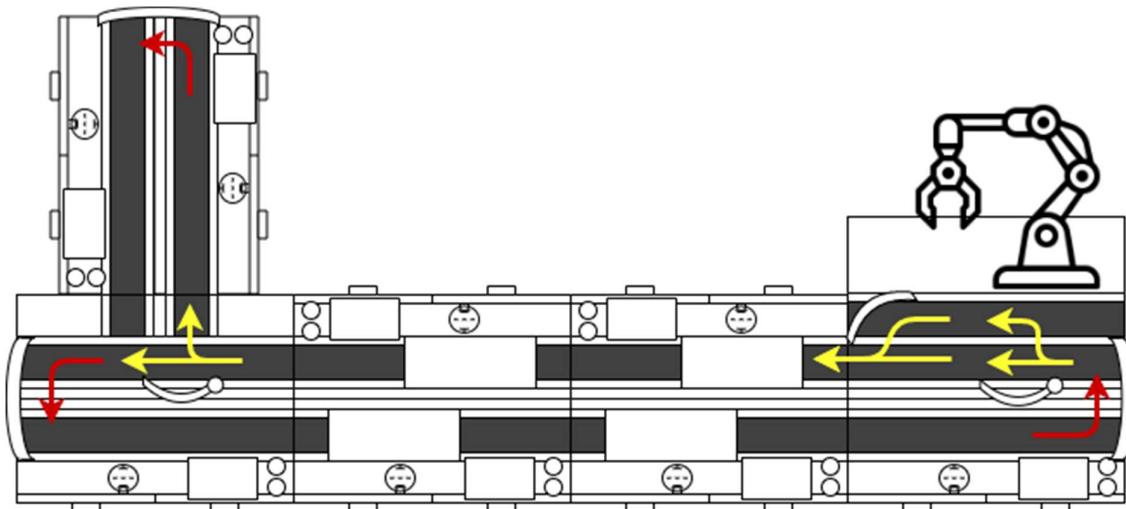
For at kunne designe, udvikle og vurdere en løsning, er det afgørende at definere og beskrive nogle begreber og teknologier med særlig betydning for projektet. Dette afsnit opdeles mellem de begreber, der tages hensyn til, og de teknologier, der bruges under projektet.

2.1 Begreber

2.1.1 AAUs Smart Production Lab

AAUs Smart Production Lab er en særlig laboratorie etableret hos AAU, som har til formål at være et knudepunkt for samarbejde mellem firmaer, forskningspartnerne og studerende, og at fungere som et platform til undervisning af studerende i Industri 4,0 begreber og teknologier, samt som et demonstrationsanlæg til produktionsvirksomheder [14]. Anlægget selv er lavet af forskellige moduler af Festo-Didactics CP Factory, som er en fleksibel produktlinje af flere automatiseret moduler, som simulerer fysisk de forskellige dele af et Industri 4,0 produktionssystem, med modellering af produktionssystemer og træning af personelle til formål [15].

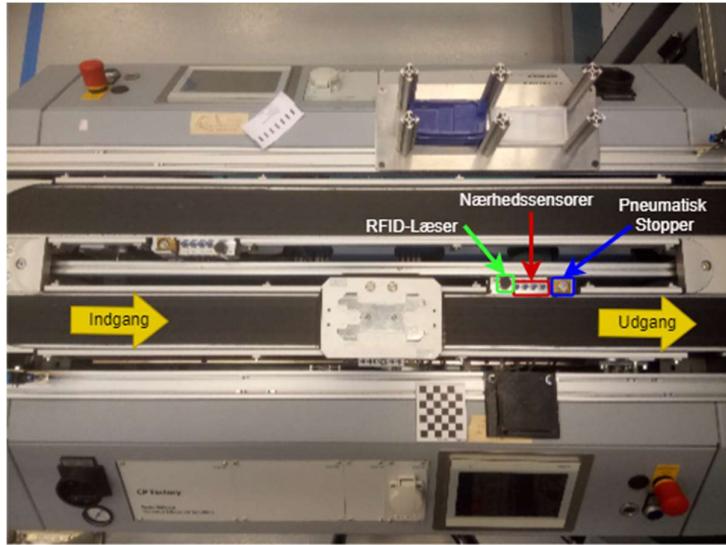
Selvom CP Factory anlægget kan betragtes som en teknologi, er Smart Production Lab håndteret her som et begreb, da dets indflydelse i projektet kan opdeles mellem de tekniske realiteter af anlægget, som diskuteres senere, og det større begreb af Smart Production Lab som et knudepunkt for demonstration og samarbejde.



Figur 1 Diagram af den Smart Production Lab med piler, der illustrerer mulige transportveje for pallerne.

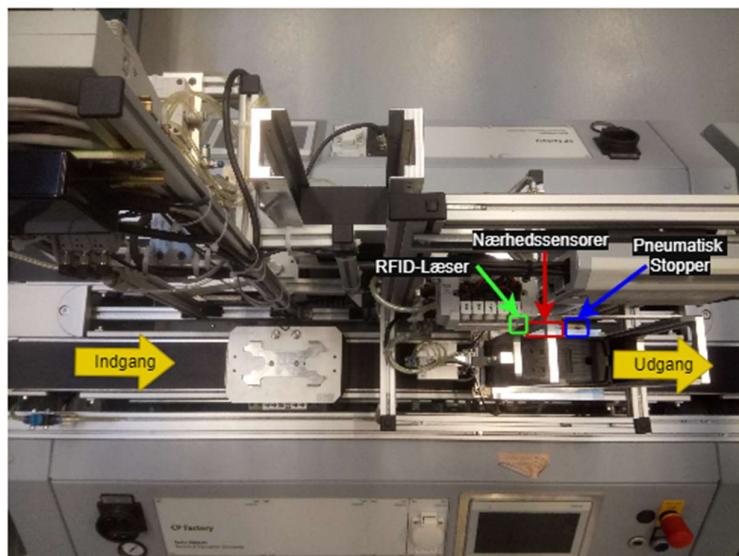
Det ovenstående diagram viser anlæggets nuværende opbygning – de moduler kan sættes op i flere placeringer – og de farvede piler viser de faste (røde) og betingede (gule) cirkulationsveje, som transportpaller kan tage gennem systemet. Hver modul har to transportørband, som styres separat af deres egne *PLC* og *HMI*. Forskellige behandlingsstationer, som styres af endnu én *PLC* mere, kan kobles til en side på én af disse 'almindelige' moduler. Derudover er der to større moduler, der styres af én

PLC og HMI, og som udfører mere komplekse dirigering af pallerne, og som kan udføre en robotproces i tilfælde af modulet yderst til højre på diagrammet.



Figur 2 Foto af STPLC_08, med transportørdirektion og beliggenhederne af RFID-Læser, Nærhedssensorer og Pneumatisk Stopper fremhævet.

Hver modul har mindst to stoppesteder, hvor pallerne stoppes for at læse deres indbygget *RFID*-tag og derved beslutter om yderligere behandling. Stopning sker automatisk, og ankomst af paller registreres af en række nærhedssensorer, som udløser læsning af pallens *RFID*-tag. Hvis pallens last skal processeres af en tilkoblet station, venter transportørbåndets stopper indtil processen er udført for at lade pallen køre videre. Dette illustreres på Figur 2 og Figur 3.



Figur 3 Foto af STPLC_10, med transportørdirektion og beliggenhederne af RFID-Læser, Nærhedssensorer og Pneumatisk Stopper fremhævet

2.1.2 Hvad er *IIoT*?

IoT er én af de hovedområder for forskningen forbundet med Innovation Factory North, og *IIoT* kan betragtes som en afgørende søjle i implementering af Industri 4,0 [16]. Der findes flere definitioner for *IIoT*, men nyere forskning har fundet på det følgende citat:

'A system comprising networked smart objects, cyber-physical assets, associated generic information technologies and optional cloud or edge computing platforms, which enable real-time, intelligent, and autonomous access, collection, analysis, communications, and exchange of process, product and/or service information, within the industrial environment, so as to optimise overall production value. This value may include; improving product or service delivery, boosting productivity, reducing labour costs, reducing energy consumption, and reducing the build to-order cycle.' - [17]

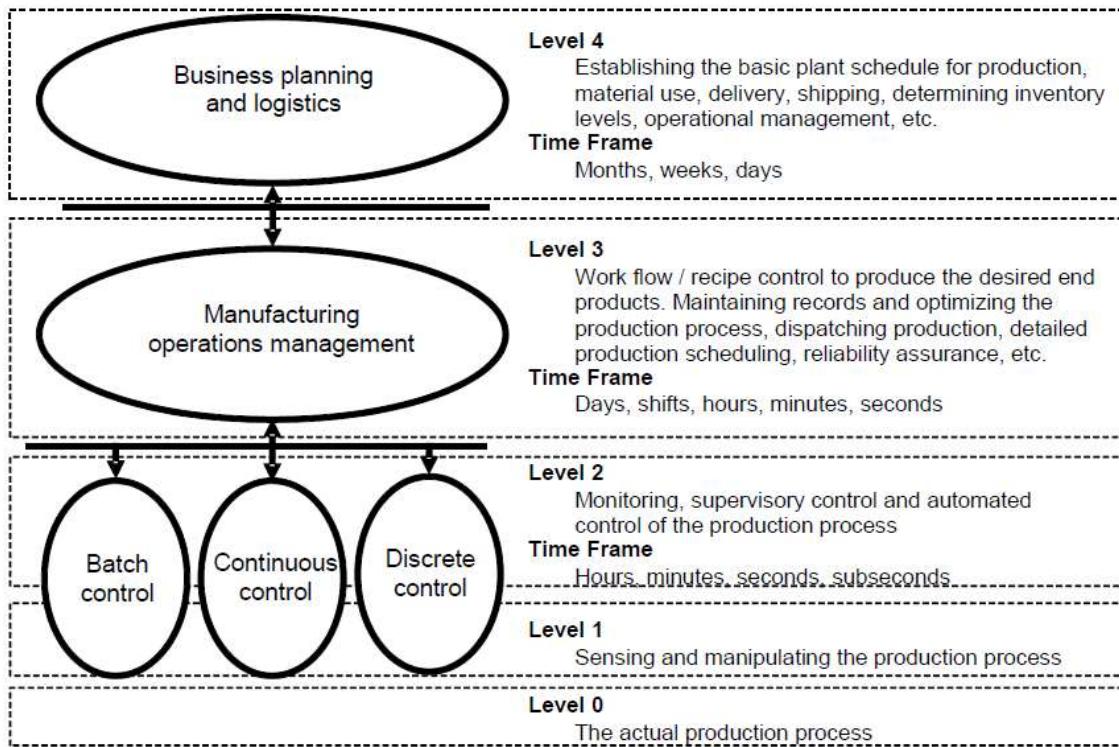
Økonomiske gevinster med *IoT*, herunder *IoTs* påvirkninger på produktions- og fremstillingsindustrier (dvs. *IIoT*), har været forudsagde af forskellige organisationer siden midten af det sidste årti [18]. Noget af forskningen udførte af Innovation Factory North fokuserer om hvordan Små og Mellemstore Virksomheder (SMV'er) kan udnytte noget af det økonomiske potentiale med *IIoT* [13]. Dette bygger på en voksende mængde litteratur om emnet, der udforsker bl.a. afgørende teknologier og arkitekturen [19], [20].

I dette projekt er opgavens fokus på dataopsamling og fremstilling af dataene til forskellige intersessegruppe, men *IIoT* tager en hovedrolle, da *IoT*, og dermed *IIoT*, er iboende forbundet med data og deres opsamling og analyse [21], [22].

2.1.3 *ISA-95*

ISA-95 og dens derivative, *IEC 62264*, er en international standard for integration af virksomheders styringssystemer. Selvom *DS/EN 62264* er den danske version af standarden, bruges betegnelsen *ISA-95* i denne rapport, da den er i brede brug i industri – også i Europa og Danmark. Standarden består af seks dele, der hver for sig udtrykker standard betegnelser og modeller, objekter og attributter til integration af virksomheders styringssystemer, samt aktivitetsmodeller og objektattributmodeller til integration af fabrikationsprocesser, transaktioner mellem forretning og fabrikation, og Messaging Service Model (MSM) tjenester [23]–[28]. Standarderne tager udgangspunkt i en funktionel

hierarkimodel, som ofte kaldes for automationstrekanten¹, og som deler forretnings- og fabrikationsfunktioner op i fem lag – som ses i Figur 4.



IEC 643/13

Figur 4 Den funktionelle hierarkimodel udgivet i DS/EN 62264-1 [23].

Den traditionelle automationstrekant har imidlertid været kritiseret for at være for rigid for de nye fleksible fremstillingssystemer, der kommer med Industri 4,0 [29]. Til gengæld har forskning udgivet af de forsker hos CIP vist, at ISA-95 har stadig værdi ift. standardisering af datamodeller og design af *Manufacturing Operations Management (MOM)* løsninger [9], [11].

I dette projekt bruges ISA-95 til analyse af den endelige tekniske løsning, og vurdering af de to synspunkter udtrykket i litteraturen.

2.1.4 OEE, Performance & Cyklus Tid

Værdien af teknologiske løsninger i produktions- og fremstillingsindustrier er svært at måle præcist, men *Overall Equipment Effectiveness (OEE)* er et værktøj, der bruges til at kunne vurdere produktivitet [30]. OEE kaldes for en *Key Performance Indicator (KPI)*, som beregnes som produktet af tre andre KPI'er; *Tilgængelighed, Performance, og Kvalitet*. I dette projekt er KPI'en *Performance* af den største interesse.

¹ Der skal lægges mærke til, at selvom den IEC 62264 standard bruger ikke en trekant til modellering af det funktionelle hierarki, er de funktionelle niveauer baseret på dem der findes i ISA-95, og som kaldes for automationstrekanten.

Performance beregnes som kvotienten af den ideale cyklustid for produktion af et enkelt produkt gangede med den totale produktionsvolumen, og den faktiske køretid [31]. Dette giver et fakta mellem 0 og 1, som bruges til yderligere beregning af *OEE*, men som kan bruges til analyse af produktionen og identificering af flaskehals. Beregning af *OEE* og analyse af produktionssystemet med hensyn til *OEE* er bedste praksis i produktions- og fremstillingsvirksomheder, og en løsning, som kan bidrage til disse aktiviteter, skaber værdi for industriledere.

Beregningen af *Performance* sker ud fra tidsdata om reelle produktionscyklus – dvs. gennemløbstider – og en idealiseret cyklustid, som selv skabes af cyklustidsdata. Imidlertid er ikke alle maskiner opbyggede med muligheden at selv overvåge cyklustiden. Derudover kan manuel optagelse af cyklustider være arbejdskrævende, og erstatning af maskiner kan være meget dyrt. Derfor kan *IoT*-løsninger, som kan eftermonteres på maskiner, skaber meget værdi hos produktions- og fremstillingsvirksomheder uden at kræve store omkostninger.

2.2 Teknologier

2.2.1 OPC UA

OPC Unified Architecture (OPC UA) er en platform-uafhængig serviceorienteret arkitektur, der integrerer alt det funktionalitet af *OPC Classic* specifikationer i én udvidelig ramme [32]. *OPC UA* er baseret på den tidligere kompatibilitetsstandard *OPC Classic* – *OPC* er en forkortelse af *Object Linking and Embedding (OLE) for Process Control* – som er blevet meget udbredt i fremstillingsindustrier, bygningsautomation, petrokemiske industrier, bæredygtig energiforsyning, og forsyningsvirksomheder. *OPC UA* muliggør kommunikation mellem uensartede enheder på en sikker og fleksibel måde, som ikke er forbundet til én platform - f.eks. Microsoft Windows.

Da alle de Smart Production Lab *PLC*'er hoster deres egne *OPC UA*-servers som standard, er *OPC UA* en væsentlig led i dette projekt. Den primære opgave udgør skabning af dataopsamling fra hver af disse servere for yderligere fremsendelse eller brug, og så er det et krav at løsningen er *OPC UA* kompatibel.

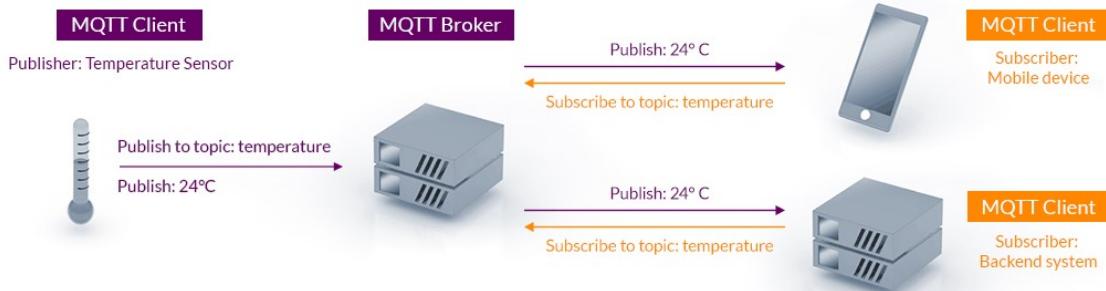
2.2.2 MQTT

Message Queuing Telemetry Transport (MQTT) er en TCP/IP-baseret kommunikationsprotokol for *IoT*-kontekster [33]–[35]. Protokollen selv blev udviklet ved enden af 1990'erne for brug med sensorer på olieørledninger [34], og er nu administreret af OASIS – en førende non-profit organisation, der specialiserer sig inden for *open source* projekter [35], [36]. MQTT er kaldt for en letvægtsprotokol, da dens beskeder har små *headers* og dens *clients* kræver få ressourcer [33]. Derudover reducerer *MQTT*'s *publish/subscribe* arkitektur trafik på kommunikationsnetværk – en egenskab der stammer fra protokollens originale anvendelsesformål med satellitkommunikation [34]. Yderligere tilbyder *MQTT* tre niveauer af *Quality of Service (QoS)*:

0 – mest én gang. Der er ingen garanti af levering, da modtageren anerkender ikke modtagelse.

1 – mindst én gang. Beskeden er lagret indtil senderen får anerkendelse af modtagelse.

2 – præcis én gang. Modtageren får kun én besked pga. en 4-delshandshakeprotokol.



Figur 5 MQTTS Publish/Subscribe Arkitektur [33].

Som en letvægt IoT-fokuseret protokol, er MQTT meget vigtigt for dette projekt. Den *publish/subscribe* arkitektur er meget mere skalerbare end en *request/response* arkitektur når der er tale om flere variabler fra flere maskiner. Derudover kan protokollen nemt fremsende opsamlede data til nye enheder – dvs. enheder som ikke i forvejen var planlagt – på netværket med at *subscribe* til opdateringer via *topics*.

2.2.3 Raspberry Pi

Raspberry Pi er *System-on-a-Chip*-baserede mikrocomputere produceret af den *Raspberry Pi Foundation* [37], og som er en af flere hardwareplatforme, der betragtes som muliggørende for *IoT* [19], [20], [38]. *Raspberry Pi* er udstyret med alt der kræves for en computer på ét printkort – bortset fra mus og skærm – og derfor kaldes for en *single-board* computer. *Raspberry Pi* er særlig tilegnede til *IoT* pga. deres direkte hardware-styring grænseflader [39] og potentialen at opfylde alle funktionelle lag mellem sensorer og brugergrænsefladen [40].

I dette projekt er en *Raspberry Pi* computer brugt pga. dens tekniske egenskaber ift. *IoT*, og fordi dens lav pris og udbredt distribution og økosystem betyder at enheden opfylder projektets krav om tilgængelighed af løsningen. Derudover er *Raspberry Pi* udstyret med *Node-RED*, som forklares nedenfor, som standard.

2.2.4 Node-RED

Node-RED er et flow-baseret programmeringsværktøj, som var udviklet af IBMs Emerging Technology Services team og er nu administreret af den OpenJS Foundation [41]. Værktøjet er implementeret på det *Node.js* runtimemiljø, og udnytter flow-baseret programmering til at lave et meget tilgængelige og fleksible programmeringsværktøj til hurtig udvikling af datastrømme. Da *Node-RED* har været

udviklet '*in the open*', er der mange brugerudviklede *libraries*, til næsten ethvert brug, såsom integration med *OPC UA* [42].

Da *Node-RED* er implementeret som standard på *Raspberry Pi* og der findes allerede stor interesse hos industriledende automationsteknologifirmaer [43], opfylder *Node-RED* projektets krav om både tilgængelighed og open source teknologier. Derudover er de mange standard og samfundsudviklede *libraries* af stor betydning for *Node-RED*s tilgængelighed, samt dens flow-baseret programmeringsparadigme, der ligner *Ladder Logic (LAD)* og *Function Block Diagram (FBD)* ift. forståelighed for ikke-programmører.

2.2.5 *RFID*

RFID-teknologi er ikke nyt, men dens udbredt brug i industri formerer en teknologisk hjørnesten i *IoT* begrebet [44], [45]. Der er flere anvendelser, der indgår *RFID*-teknologi, men de kan stort set deles op mellem *nearfield* og *farfield*. Imidlertid er deres funktion stort set det samme – en antennen modtager et signal af en *RFID*-læser og bruger energien derfra til at forsyne en *RFID*-tags elektronik så den kan svare til *RFID*-læseren [46], [47].

RFID-teknologi er relevant til dette projekt, da pallerne, der kører rundt Smart Production Lab, er udstyret med *RFID*-tags for identificering [48], og måden at disse tags bruges skal tages hensyn til under udviklingen.

2.2.6 SIA-Connect Gateway

SIA-Connect er en *IIoT* protokol-gateway og *datalogger* enhed, der produceres af SIA [49]. Enheden muliggør dataopsamling gennem integration med en række industrielle og *Building Management System (BMS)* protokoller [50] og flere cloud-løsninger, såsom Microsofts *Azure*, *Amazon Web Services (AWS)*, og *Google Cloud* bl.a. [51]. Derudover kan SIA-Connect oversætte enhver af de understøttede protokoller til en anden af de understøttede protokoller [49] – dvs. at enheden kan opsamle data fra en *OPC UA-server* og derefter tilbyde dataene via en *MQTT publish/subscribe* forhold til andre enheder.

SIA-Connect var en del af den oprindelige løsning, da den var implementeret som en del af en endnu tidligere løsning, som var relateret til forskning forbundet med Innovation Factory North [13]. Derfor var udforskning af integration med SIA-Connect en væsentlige spor i dette projekt.

2.2.7 *SQLite*

SQLite er et letvægt C-baseret *SQL database engine* som er den mest udsendte i verden [52], [53]. *SQLite* er i dens 3. version og tilbyder *ACID*² databasetransaktioner i en enkeltstående og serverløs *database engine* [54]. Derudover er *SQLite* en *zero-configuration* database, som kræver ikke de

² ACID – Atomar, Konsistens, Isolering, Holdbarhed.

samme konfigurering som andre *SQL*-databaser [55]. Disse egenskaber betyder at *SQLite* er meget tilegnet til indlejrede systemer, hvor der ikke er ressourcerne til traditionelle server-baserede databaser, og dette er grundet til at *SQLite* blev valgt til brug i dette projekt.

2.2.8 *UAExpert*

UAExpert er et *OPC UA client*-software, der udgives af Unified Automation GmbH [56]. Softwaren muliggør afprøvning af *OPC UA*-funktioner, såsom *data access* og *UA-methods*, og er tilgængelige for Windows og Linux. Softwaren er særligt brugbar for at kunne udforske *OPC UA-servers* via *OPC UAs discovery* funktionalitet [32], og derfor var softwaren relevant til dette projekt.

3. Metode

Projektet tager udgangspunkt i en konkrete opgave, der fandt sted ved enden af praktikperioden hos AAUs Smart Production Lab. Den oprindelige opgave varede én uge, og projektet sprang ud af ønsker både at forbedre løsningen og at analysere dens teoretiske grundlag. I denne afsnit findes der en præsentation og forklaring af udviklingsprocessen, samt udfordringer og løsninger dertil.

3.1 Udgangspunkt

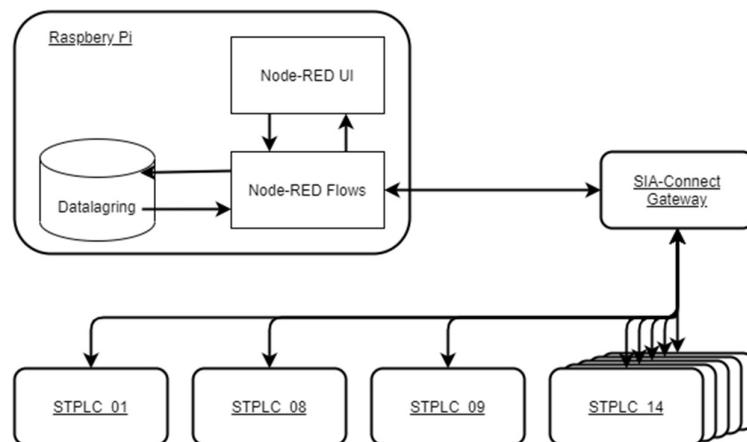
Da projektet tager udgangspunkt i en konkrete opgave, er det vigtigt at begynde med en kort beskrivelse om opgaven selv, for at kunne bedre definere konteksten, hvor projektet stammede fra.

3.1.1 Opgavens rammer

Opgaven fik et klart mål at stille op en løsning, der kunne løse behov for optagelse af cyklus- og gennemløbstider for ordre på Smart Production Lab, og for fremstilling af disse til yderligere analyse. Der var stillet en tidsfrist på én uge, da løsningen skulle præsenteres til industripersonelle og -ledere. Derudover var en tidligere løsning, som bestod af en *Raspberry Pi* mikrocomputer og en *gateway*-enhed, SIA-Connect, givet som grundlaget for løsningen. Funktionaliteten af denne tidligere løsning bliver ikke berørt her.

3.1.2 Planlægning

Opgavens rammer ledte til relativt klare funktionelle specifikationer, og derfor var en foreløbig plan hurtigt lavet. Denne plan gjorde brug af den SIA-Connect enhed til skabelse af kommunikation mellem de forskellige *PLC*'er på Smart Production Lab'en og den *Raspberry Pi* computer, som illustrerede på Figur 6.



Figur 6 Funktionelle arkitektur af løsningen til den oprindelige opgave.

Da kommunikation mellem SIA-Connect, *Raspberry Pi*'en, og de *PLC*'er var sat op i forvejen, var de eneste funktionelle tilføjelser en database og de krævede flows for fremstilling af de optagede data.

Der var yderligere en *MQTT-broker* sat op på én af de eksisterende *Node-RED flows*, som blev brugt til formidling af data mellem alle flows.

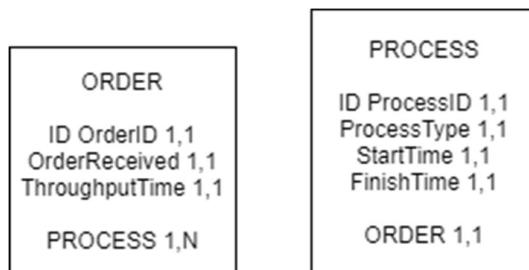
Funktionel analyse af anlægget

En grundig analyse af anlæggets funktion var udgangspunktet for den oprindelige opgave, da opsamling af data om anlæggets funktion er kernepunktet for løsningen. Processerne forklarede på Figur 2 og Figur 3 dannede grundlaget for analysen, og dertil var *UAExpert* også brugte til efterforskning af *OPC UA* variabeller og deres værdier gennem processerne. Målet med denne analyse var at finde en variabel, som kunne bruges som en triggervariabel for optagelse af datapunkter.

Der var behov for denne triggervariabel, fordi målet med dataopsamlingen var yderligere analyse af cyklus- og gennemløbstider. Så var det nødvendigt at kunne identificere start- og slutpunkter for hver procescyklus, og at kunne optage værdierne af variablerne på disse tidspunkter. Variabellen *xCarrierAvailable* blev valgt som triggervariabel, da den gav indikation for pallernes tilstedeværelse på hver station, og da der sker en automatisk stop ved hver station, gav det mening at bruge denne variabel til adskillelse mellem transporttider – så kunne man differentiere mellem en lang transporttid mellem to stationer og normale gennemløbstider.

Databasens design

Da der var kun brug for en relativt simpel database, var det valgt at bruge semantik-modellering til databasens design [57]. Ordre og processer var de to centrale temaer for databasens indhold, og så blev de defineret som semantiske objekter, hvor en ordre kunne have flere processer uddover dens identifikationstal (*OrderID*), starttid (*OrderReceived*), og gennemløbstid (*ThroughputTime*), og en proces bestod af en identifikationstal (*ProcessID*), en procestype (*ProcessType*), en starttid (*StartTime*), og en sluttid (*FinishTime*).



Figur 7 Semantikmodeller til databasens design.

Med disse modeller som udgangspunkt blev de *SQL*-udsagn for initialisering af tabellerne defineret. Procestabellen blev defineret med *ProcessID* som automatisk stigende *primary key*, da der er et behov for at kunne identificere hver eneste procescyklus – både for indsættelse i databasen og for opsporing til fejlfindingsformål. Derudover var *OrderID* defineret som *foreign key*, da dette er den mest fornuftigt måde at forbinde ordrer og processer sammen. Den endelige udsagn for initialisering af procestabellen var:

```

CREATE TABLE processes(ProcessID INTEGER PRIMARY KEY AUTOINCREMENT,
ProcessType VARCHAR(10), OrderID INTEGER, StartTime TIMESTAMP, FinishTime
TIMESTAMP, FOREIGN KEY (OrderID) REFERENCES orders(OrderID));

```

Ligeledes blev ordretabellen defineret med *OrderID* som *primary key* og med *OrderReceived* og *ThroughputTime* som timestamps. Imidlertid var *OrderID* ikke defineret som automatisk stigende, da disse tal stammer fra det *Manufacturing Execution System (MES)* - og derfor skal optages uden ændring – og fordi de skal være helt unikke alligevel.

De indsættelses- og opdateringskommandoer blev udviklet på grundlaget af den tidligere nævnte funktionelanalyse af anlægget. Da der forventede en bestemt sekvens i processen (se 2.1.1 AAUs Smart Production Lab), var det muligt at bruge kommandoer til håndhævelse af dataenes integritet – dvs. at der må ikke være en sluttid på et eksempel af en proces før der findes en starttid til det. Denne håndhævelse udføres med at definerer starttiden som indsættelsestrigger – hvor en nye række i databasen dannes – og sluttiden som opdateringstrigger – hvor en eksisterende række færdiggøres med sluttiden. På denne måde er det ikke muligt at optage en sluttid før der optages en starttid, og dette forhindrer optagelse af forkerte tider pga. fejler eller forsinkelser.

Indsættelseskommndo var:

```

INSERT INTO processes(OrderID, ProcessType, StartTime) values([OrderID],
[StationNumber], [Timestamp]);

```

Opdateringskommndo var:

```

UPDATE processes(FinishTime) values([Timestamp]) WHERE OrderID = [OrderID]
AND ProcessType = [Station Number];

```

En tabel på *Node-REDs dashboard* var valgt som den mest fornuftigt måde at fremvise tidscyklusdataene, og de relevante kolonner blev valgt ud fra de semantikmodeller vist på Figur 7. Det blev også bestemt at der var brug for en ordresøgningsfunktion, og dermed en nulstillingsfunktion. SQL-kommndoen til fremvisning af cyklustidsdataene og nulstilling af ordresøgningsfunktionen var:

```

SELECT * FROM processes ORDER BY StartTime;

```

SQL-kommndoen til ordresøgningsfunktion var:

```

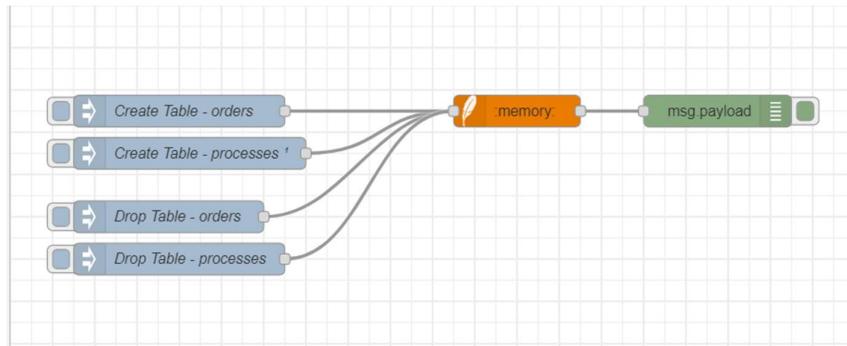
SELECT * FROM processes WHERE OrderID = INPUT ORDER BY StartTime;

```

3.1.3 Implementering i *Node-RED*

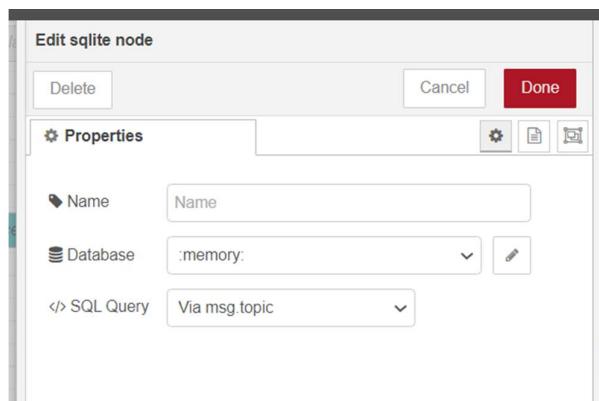
Der var tre primære dele til den oprindelige løsnings funktion: konfigurering af databasen og initialisering af tabeller, håndtering af SQL-indsættelser og -opdateringer af databasen, og fremstilling af optagede data, samt håndtering af ordresøgningsfunktionen.

Konfigurering af databasen og initialisering af tabeller



Figur 8 Flow for konfigurering af databasen og initialisering af tabeller.

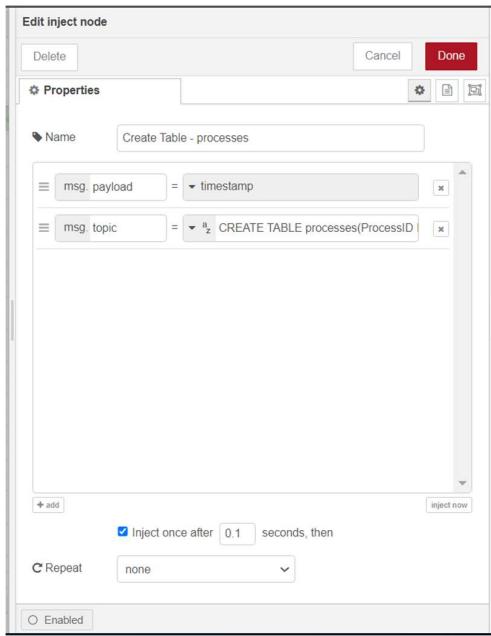
Den ovenstående figur viser flowet, der bruges til konfigurering af databasenoden og initialisering af tabellerne, som var brugt til den oprindelige løsning. Flowet bygges op af en *SQLite3* database node udviklet af *Node-REDs* onlinesamfund af udviklere [58], og flere *inject*-noder, hvis *msg.topic*-egenskaber bruges til fremsendelse af SQL-kommandoer.



Figur 9 Konfigurering af *SQLite3* databasenode.

Konfigurering af databasenoden vises på

Figur 9. Det blev valgt at bruge *msg.topic* egenskaber til fremsendelse af SQL-kommandoer, da det gav muligheden at sende flere forskellige kommandoer til den ene node. Derudover blev det valgt at bruge *:memory:* som database, da dette initialiserer en ikke-vedholdende database, som er nemmer at styre under udvikling, og fordi der var krav om hverken en vedholdende database eller adgang til gamle data.



Figur 10 Konfigurering af inject-node for initialisering af processtabellen i databasen.

De tidligere formulerede kommandoer blev konfigureret som msg.topic-egenskaben for inject-noder, som ses på

Figur 10. Derudover var noderne konfigureret at sende beskeden kun én gang 100ms efter opstart.

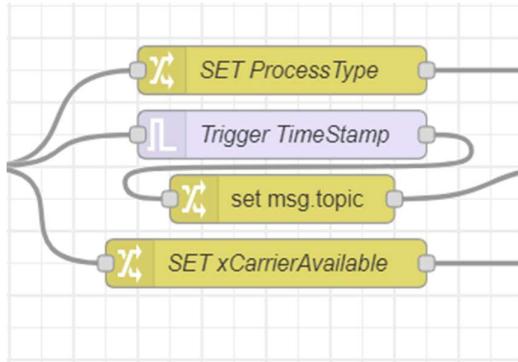
Dette betød at det ikke var nødvendigt at initialisere tabellerne efter hver tænd/sluk-cyklus.

Der var også inkluderet inject-noder til fjernelse af tabellerne som en del af afprøvningen af flowet, som ikke blev fjernet med implementering, da der forudså en mulighed at de kunne være brugbare senere i udviklingen.

Implementering af SQL-indsættelser og -opdateringer

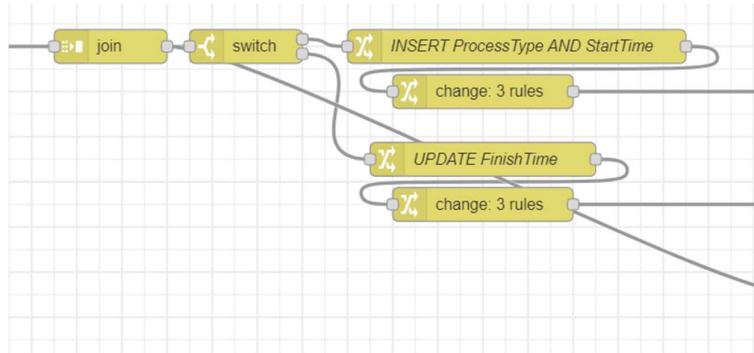
Figur 11 (til venstre) - a) Konfigurering af MQTT-subscription til xCarrierAvailable. (til højre) - b) Konfigurering af MQTT-subscription til OrderID (udiONo).

Selvom *xCarrierAvailable* blev valgt som triggervariabel, var der behov for optagelse af flere værdier for at kunne lykke de SQL-kommandoer rigtigt. *OrderID* kunne nemt optages via *MQTT-subscription* på samme måde som *xCarrierAvailable* – som illustreres på Figur 11 – men *ProcessType* og den altafgørende timestamp skulle udløses af skiften i værdien af *xCarrierAvailable* – som vises på Figur 12.



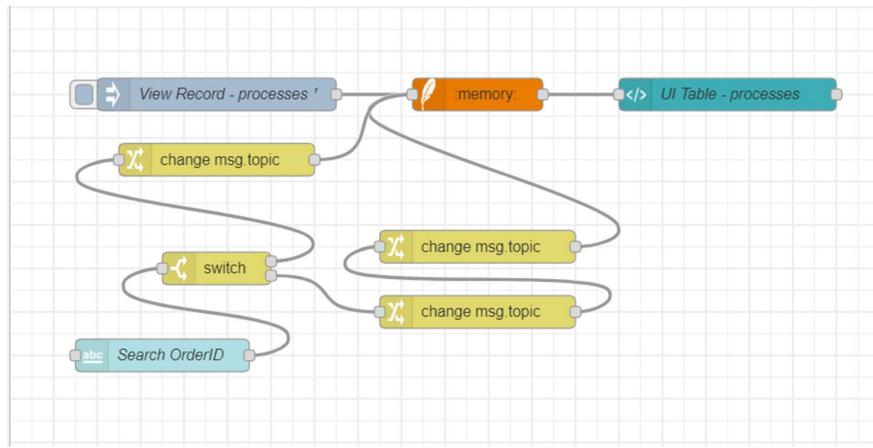
Figur 12 Udløsning af beskeddele og indstilling af *msg.topic* egenskaber.

Derefter blev alle fire beskeddele kombineret i én besked af den *join-node*. Disse beskeder blev sorteret af den følgende *switch-node* afhængigt af værdien af *xCarrierAvailable*. Til sidst var beskedernes *msg.topic*-egenskaber ændrede afhængigt om de skulle indsætte en nye rekord eller opdatere en eksisterende rekord – baseret på værdien af *xCarrierAvailable* – og de forskellige værdier indsat ind i *msg.topic*, så SQL-kommandoerne opnået deres tidligere defineret form. Dette illustreres på Figur 13.



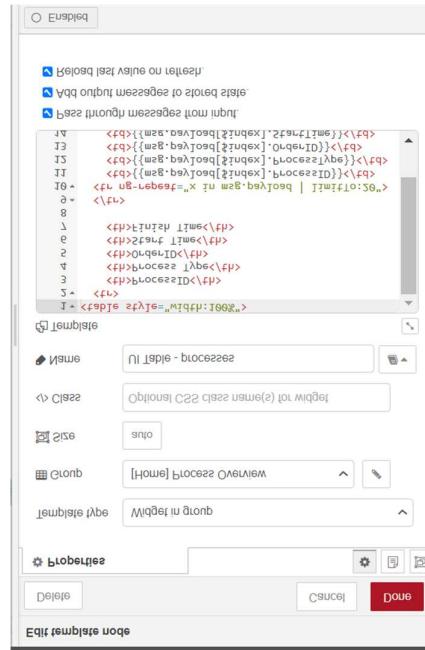
Figur 13 Kombinering og sortering af beskedobjekter, og genindstilling af *msg.topic*-egenskab.

Datafremstilling



Figur 14 Flow for fremvisning af optagede procescyklustidsdata og ordresøgningsfunktion.

Figur 14 viser flowet for fremvisning af cyklustidsdata. *Inject*-noden initialiserer fremvisningstabellen med en *SQL-query* konfigureret som *msg.topic*-egenskab. Dette fremsendes til *template*-noden – her betegnede *UI Table – processes* – konfigurering³ deraf vises på Figur 15. En *text input*-node er brugt til ordresøgningsfunktionen. Denne node fremsender et *string*-input, som sendes videre til *switch*-noden, det skelner mellem *msg.payload*-egenskaber, der har indhold, og dem, der ikke har indhold – dvs. mellem *null* og *not null* værdier.



Figur 15 Konfigurering af template-noden til fremvisning af procescyklustider i tabelformat.

³ Noden er konfigureret i *Angular*-programmeringssprog, som ikke håndteres videre, da dens rolle i projektet er ikke væsentlig.

3.2 Yderligere udviklinger

Den tidligere afsnit udforsker løsningen til den oprindelige opgave. Løsningen var kun delvis funktionel, og selvom det var tilstrækkeligt for demonstrationen af *IIoT* teori i praksis, var der ønsker om forbedringer og udvidelser ift. funktionalitet. Dette afsnit omhandler disse udviklinger.

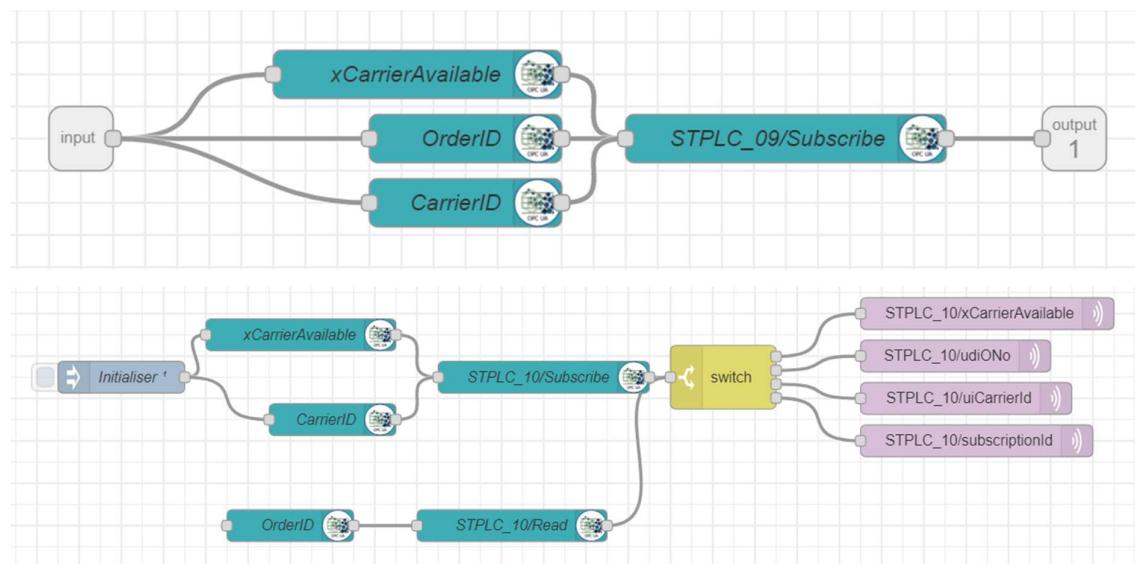
3.2.1 Problemer med SIA-Connect og hardwareskift

Den oprindelige løsning blev implementeret og præsenteret til industriprofessionelle og -ledere. Imidlertid var dens funktion ikke ideelt, og der var væsentlige mangler og fejler i dens optagelse af cyklusdata. Dette skyldes sporadiske afbrydelser mellem SIA-Connect enheden og de *OPC UA-servers*, som måtte afbryde forbindelsen øjeblikkeligt eller i flere minutter – oppe mod en halvtime. Efter flere forsøg at rette problemerne –f.eks. via genkonfigurering af indstillinger – var det besluttet at droppe SIA-Connect enheden fra projektet. Alternativet var at udvikle en *OPC UA*-tilgang i *Node-RED*.

3.2.2 *OPC UA-client* via *Node-RED*

Efter dropning af SIA-Connect enheden, var der to væsentlige problemer at løse – skabelse af forbindelse til de *OPC UA-servers* og rettelse af de *code-breaking* ændringer grundet fjernelse af SIA-Connect.

Udvikling af OPC UA-clients

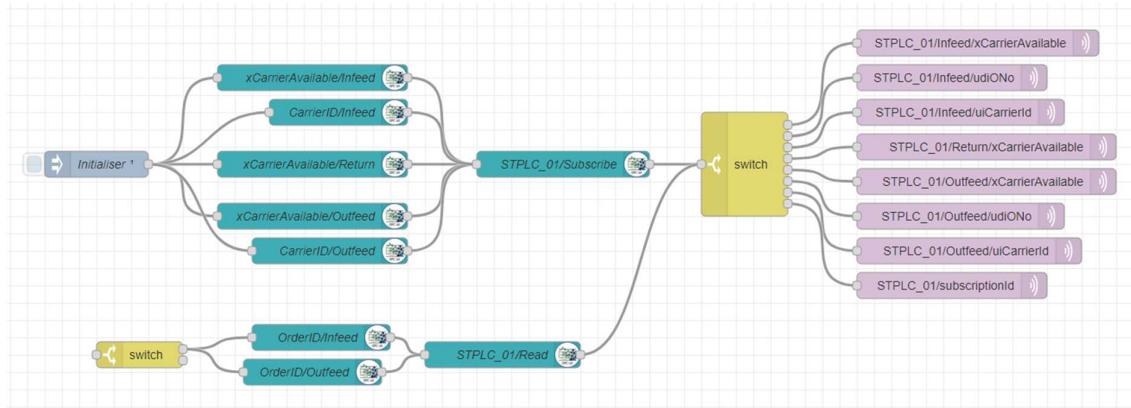


Figur 16 (ovenfor) – a) *OPC UA-Client* flow for 'almindelige' stationer på det Smart Production Lab anlæg.
(nedenfor) – b) Den endelige *OPC UA-Client* sub-flow implementerede i løsningen.

Den oprindelige løsning var baseret på dataforbindelsen skabt af SIA-Connect. Ved dens fjernelse blev det nødvendigt at genskabe forbindelsen i *Node-RED*. Heldigvis er der allerede udviklet en *library* til

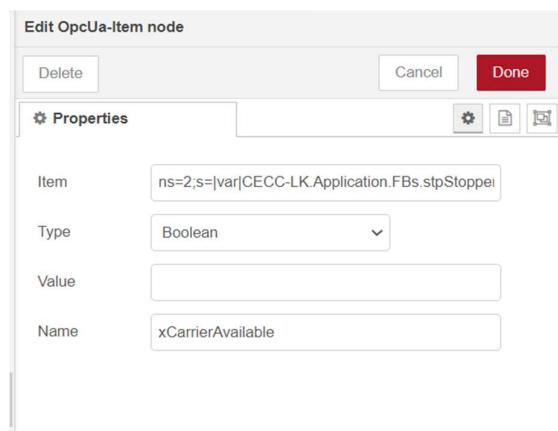
præcis dette formål [42], som tilbyder nogle af de funktioner⁴, der kræves af en *OPC UA-client*. Denne library skabte grundlaget for udviklingen af *OPC UA-client flows*, som vist på Figur 16 og Figur 17.

To former for disse *client-flows* blev udviklet: ét til 'almindelige' stationer – dvs. stationer med kun én transportbånd og muligvis en processtation – og ét til hver af *STPLC_01* og *STPLC_14*, da disse stationer har en mere komplekse transportfunktionalitet. Alle disse flows blev indkapslet af sub-flows for at behold en modulær og læsbare kodebase.



Figur 17 OPC UA-client flow for station *STPLC_01* på det Smart Production Lab anlæg.

Flowene er bygget op fra tre nodetyper: den *OPC UA Item-node*, den *OPC UA Client-node*, og den *MQTT Out-node*. *OPC UA Item-noder* er brugt til identificering af de relevante variabler, og dertil konfigureres med *OPC UA* variabeladresser som findes i Tabel 1⁵. Derudover konfigureres noden med en forventet *datatype* og en mulig værdi, hvis der er tale om en *write-kommando*. Dette illustreres på Figur 18.

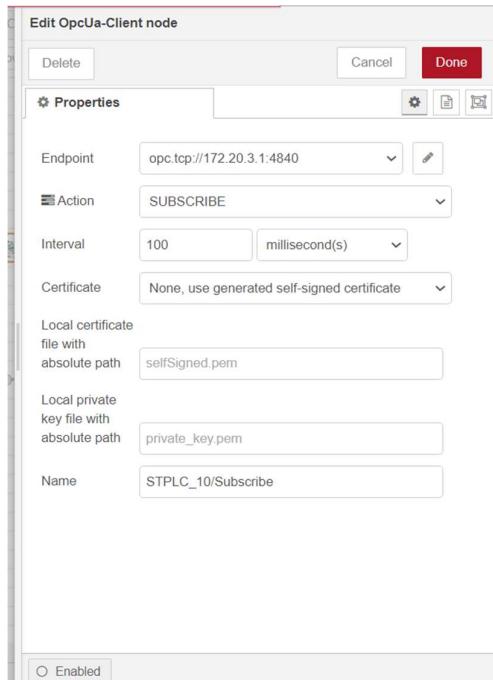


Figur 18 OPC UA Item-node konfigurerering for *xCarrierAvailable* variabellen for *STPLC_10*.

⁴ Da denne *library* er udviklet af samfundsudviklere, er ikke alle funktioner implementeret endnu. Imidlertid er de basisfunktioner, såsom *read/write* og *publish/subscribe* implementeret, og der kræves ikke mere funktionalitet i dette projekt.

⁵ Tabel 1 var skabt som en del af ét af de forsøger at rette de forbindelsesproblemer med SIA-Connect.

OPC UA client-noder bruges til etablering af forbindelser med *OPC UA-servers*. Disse noder tager *OPC UA item*-node outputs som input og bruger indstillinger deri for at finde de ønskede variabeller. Som vist på Figur 19, var noderne konfigureret med *IP-adresser* og den *MQTT-portnumber* som *endpoint* – alle *endpoints* var fundet ved brug af *UAExpert*. Derudover skal den ønskede *OPC UA*-kommando og alle relevante parametre, såsom *monitor interval* være indstillet. Til sidst kan man indstille sikkerhedsparametre for opsætning af sikkerkommunikation – denne mulighed var ikke udforsket i dette projekt, og derfor var ingen sikkerhedsindstillinger konfigureret.

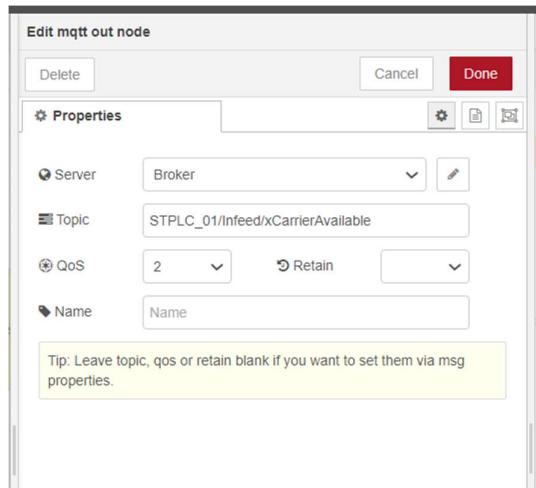


Figur 19 Konfigurerering af *OPC UA Client-node* for subscription til *STPLC_10s OPC UA-Server*.

De resulterende beskeder fra *OPC UA-servers* blev endelig sorteret af den store *switch-node* på basis af deres *msg.topic*-egenskab – variabellens *identifier* og værdi er henholdsvis sendt som *msg.topic* og *msg.payload*. Derefter blev beskederne sendt videre af deres respektive *MQTT out*-noder.

Rettelse af code-breaking ændringer

Fjernelse af SIA-Connect brudte alle dele af koden, der modtog data via *MQTT-subscriptions*, da SIA-Connect var ansvarlig for routing af *OPC UA*-data til *MQTT topics*. Det blev valgt at udfør fremsendelse af dataene ud fra sub-flowene via *MQTT out*-noder, da dette fjernede behovet for yderligere dirigering af dataene til andre flows og gav muligheden for andre enheder at *subscribe* til *MQTT-brokeren* på *Raspberry Pi* computeren. Konfigurerering af disse noder vises på Figur 20.



Figur 20 Konfigurering af MQTT-Out node for Infeed/xCarrierAvailable for STPLC_01.

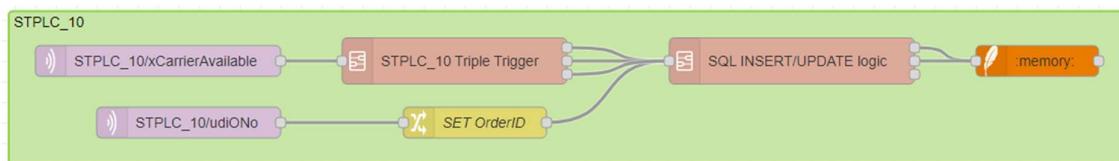
Resultater af funktionelafprøvninger

Koden var afprøvet for funktionaliteten på STPLC_08, før den blev udvidet til hver station. Som ses på Figur 33 (Se bilag 7.2.1 Udvikling af OPC UA Clients) var flowets funktion som forventet, og derfor blev ændringer indarbejdet til alle moduler. Imidlertid oversatte det ikke til den ønskede funktionalitet efter flere afprøvninger (se bilag 7.2.2-4). Analyse af disse resultater identificerede at den anden palle i rækken, som kom til STPLC_10, fik det forkerte ordrenummer tildelt dens ankomst. Dette førte til en forstyrret rækkefølge i registrering af pallernes ankomster og afgange, som ikke blev registreret pga. databasens indsættelses- og opdateringslogik. Den planlagt løsning var at forsinke læsning af OrderID indtil den var opdateret på maskinen.

3.2.3 Forbedringer til systemets funktion

Efter gendannelse af dataopsamling fra anlægget via direkte forbindelse til de OPC UA-servers, var der behov for oprydning af flowene – primært datahåndterings-flowet, da mængden af noder på det ene flow reducerede kodens læsbarhed. Så var de datastrømme konsolideret i sub-flows og fremvisnings-flowet blev udvidet med en opdateringsknap til procesdatatabellen.

Konsolidering af data-flowene



Figur 21 Intermediær data-flow for STPLC_10.

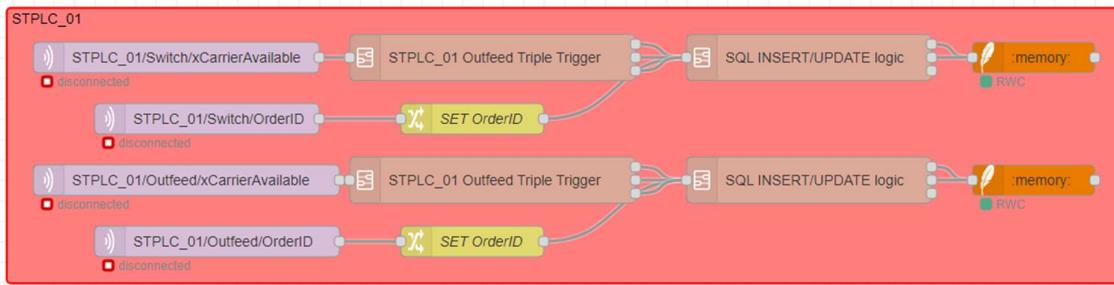
Figur 21 viser den foreløbig konsolidering udført på datahåndterings-flowet. De tre udløsningsnoder (se Figur 12) blev konsolideret i én sub-flow, og den følgende SQL-indsætnings og -opdaterings logik (se Figur 13) blev også konsolideret i en anden sub-flow. Efter afprøvning af dette var et problem opdagede med flowets funktion (se ovenfor). Efter en del analyse var det mest sandsynlige årsag blev

bestemt til at være at stationerne opdaterer deres *OrderID*-variabel relativt sent i stationens processering af pallerne. Derfor var løsningen at indføre en forsinkelse i læsning af variabellen, og efter flere afprøvninger var 550ms identificeret som tilstrækkeligt.



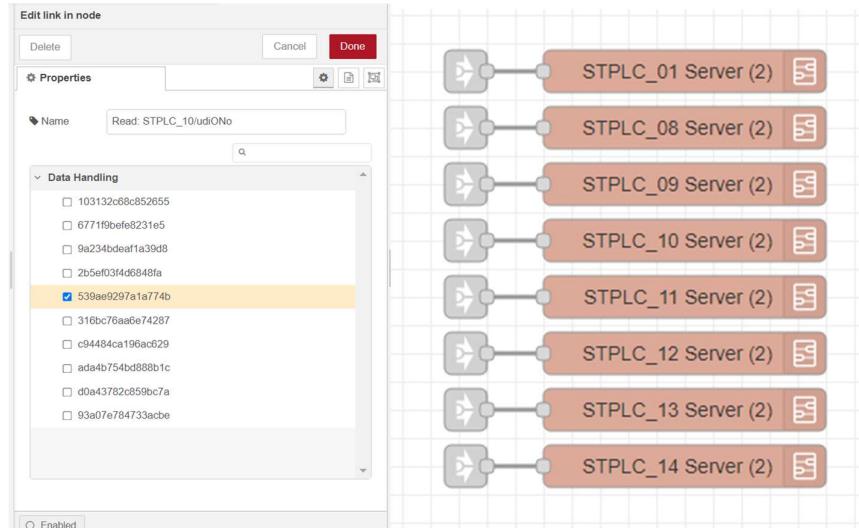
Figur 22 Endelig data-flow for STPLC_10.

Lignende konsolidering var udført på STPLC_01s og STPLC_14s data-flows. Disse flows var ikke udviklet videre under projektet, men kunne udvikles videre i fremtiden.



Figur 23 Intermediær data-flow for STPLC_01.

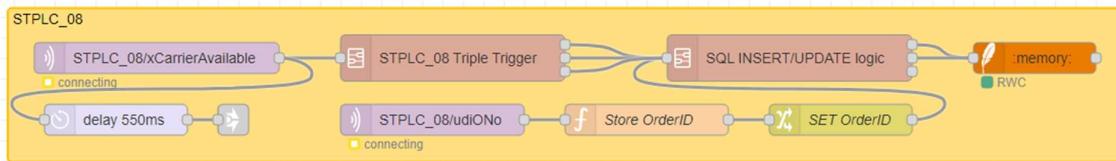
Den tidligere nævnte udløsning af ordrenummerets *read*-kommando var implementeret med brug af *link in*- og *link out*-noder, som vist på Figur 24. Figur 24 viser konfigurering af de *link in*-noder – hver *link in*-node blev forbundet med den tilsvarende *link out*-node.



Figur 24 (til venstre) - a) Konfigurering af forbindelsen mellem Link In- og Link Out-noder. (til højre) - b) Forbindelser mellem Link In-noder og OPC UA Client sub-flows⁶.

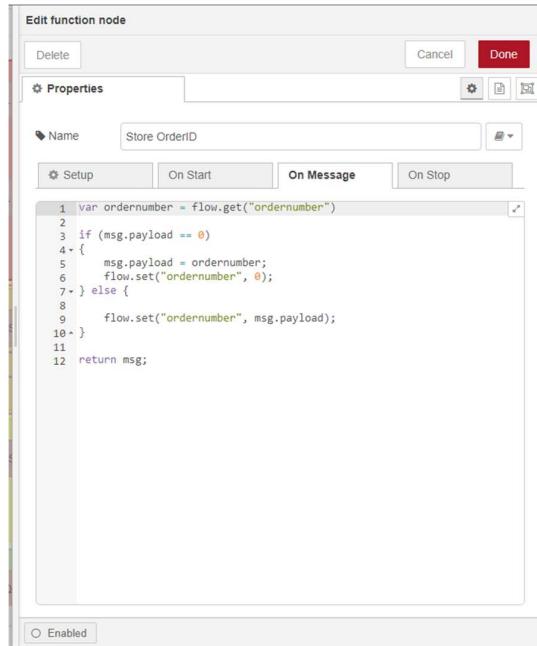
⁶ Mærk at disse noder kaldes for servers, da de repræsenterer serveren i flowet, men de består af OPC UA client-noder.

Figur 25 viser et særligt flow, som var udviklet til *STPLC_08*, da den er endepunktet for produktionslinjen og pallerne får deres *OrderID*-værdier fjernet under stationens proces. Dette førte til, at procesdatabellen ikke blev opdateret med sluttider og det er ikke muligt at beregne cyklustiden for stationens proces. Problemet blev løst med brug af en *function-node* til midlertidig lagring af ordrenummerets værdi.



Figur 25 Flow for *STPLC_08*.

Konfigureringskoden for den midlertidige lagring af *OrderID* er vist i Figur 26. Denne node tager imod en *msg.payload* og gemmer den i en variabel, hvis den ikke er 0. Hvis den er 0, gemmes den sidste værdi fra variablen. Denne konfiguration reducerer risikoen for fejltagning ved registrering af en tom palle med en forkert *OrderID*.



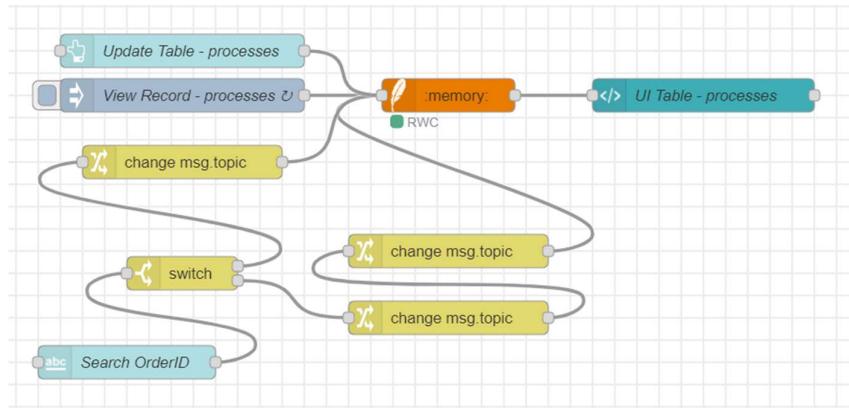
Figur 26 Konfigureringskoden for den midlertidige lagring af *OrderID*.

Opdatering af procesdatabellen og eksport af tabeldata

Under funktionelafprøvninger var det fundt at der var behov for muligheden for manuel opdatering af procesdataballen, da den gentagende opdateringsfunktion krævede en lang cyklus for ikke at blive

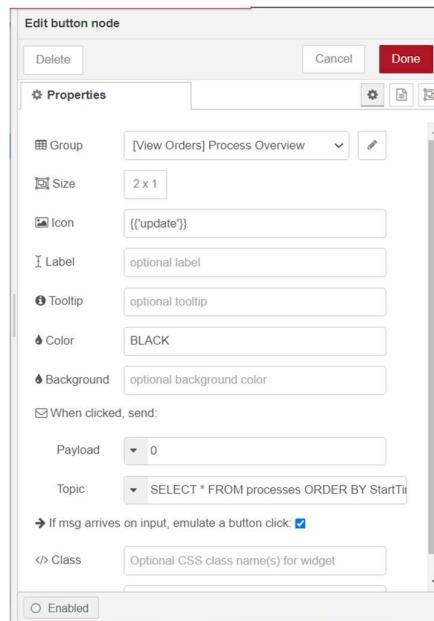
⁷ Flow-variabeller er én af de context-variabeltyper. Disse variabeller kan kun tilgås indenfor deres kontekst – enten *node*-, *flow*-, eller *global*-kontekster. Da funktioner i *function-nodes* bliver kaldt på hver input, betyder dette, at *OrderID* skal gemmes udenfor noden for at undgå at blive overskrevet hver gang.

generende – f.eks. hvis man ville søge en bestemt ordre i et betydeligt stykke tid. Så blev en opdateringsknap tilføjet til flowet, som vist på Figur 27.



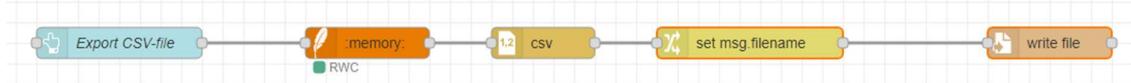
Figur 27 Ændret flow for fremvisning af procescyklustidsdata med opdateringsknap.

Figur 28 illustrerer konfigurering af den relevante node – *msg.topic*-egenskaben er den damme som den *inject-node*, der bruges til initialisering.



Figur 28 Konfigurering af den opdateringsknap-node.

Derudover blev tabellens funktionalitet yderligere udvidet med muligheden at eksportere CSV-filer. Dette udføres med brug af en knapnode konfigureret som den til opdatering af procesdatatabellen – men tildelte et andet ikonbillede. Denne node generer en *SQL-query* til databasen for at hente dataene, som bliver sendt videre til CSV-noden. *Change*-noden bruges til at sætte det *msg.filename*-egenskab, som bruges af den *write file*-node for filstien til den nye fil.



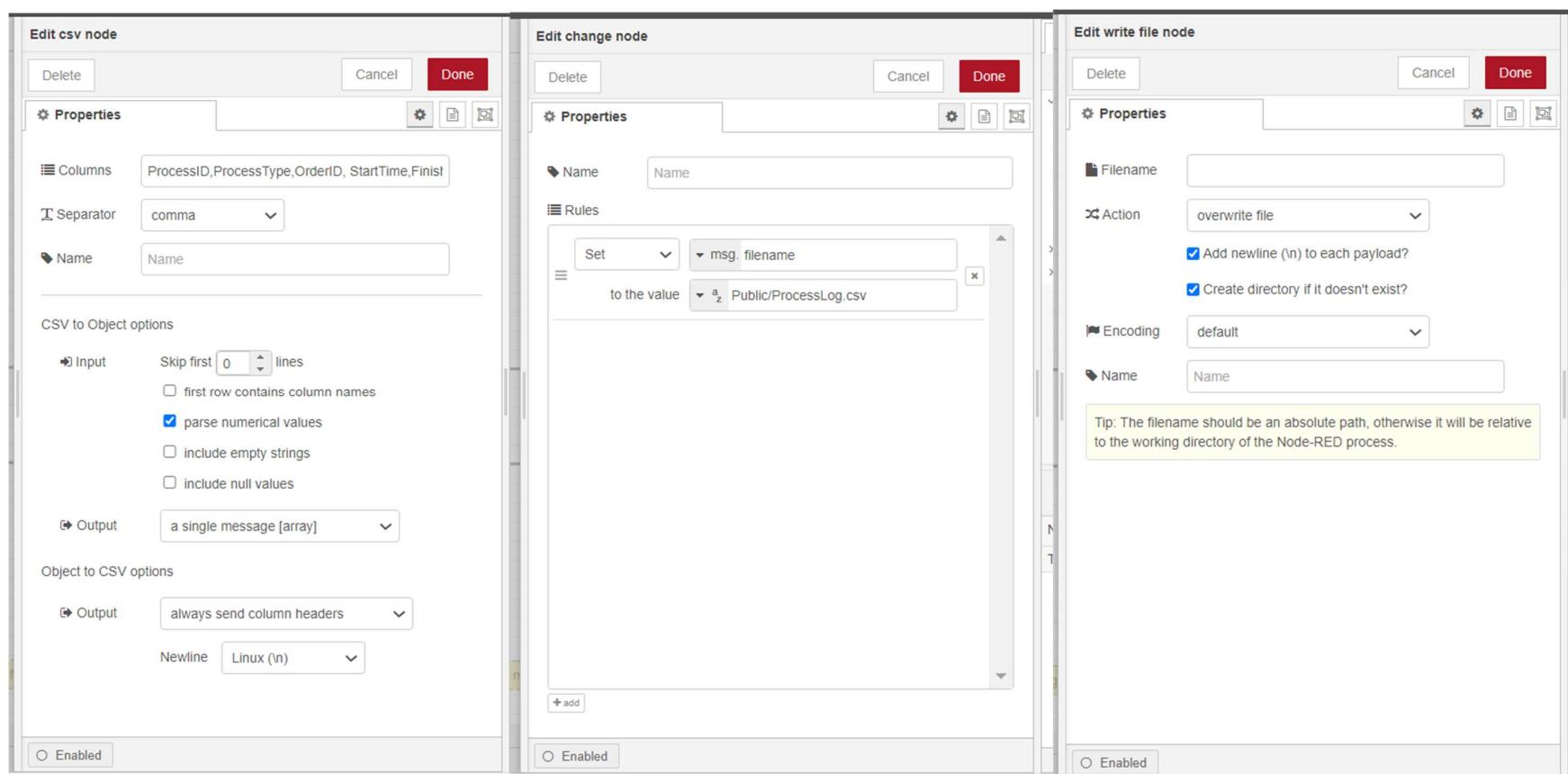
Figur 29 Flow for eksport af procesdata som CSV-fil.

Konfigurering af disse noder illustreres på Figur 30. CSV-noden skal konfigureres med kolonnenævner og andre indstillinger, der handler om nodens opførsel ift. input og output. For dette flow var det vigtigt at noden fremsendte dataene som en enkelt besked, da *write file*-noden aktiverer for hver besked. *Change*-noden skal indstilles med filstien – her bruges der en relativ filsti, da *Node-RED* bruger '/home/pi' som defaultmappe på *Raspberry Pi* og dette gør filens placering mere forudsigelig. *Write file*-noden konfigureres med indstillinger om nodens opførsel. I dette projekt var det valgt at overskrive filen hver gang, men det er også muligt at tilføje dataene til én vedholdende fil eller at gemme flere dynamisk navngivet filer.

Resultater fra funktionelafprøvning af ændringer

Resultater fra funktionelafprøvninger af disse udviklinger kan ses i bilag 7.2.5-10. Afprøvninger med forsinkelser på 50ms, 75ms, og 100ms (bilag 7.2.5-7) gav ganske ingen forbedringer på funktionen, og dette gav motivationen for afprøvning af endnu større forsinkelser. En forsinkelse på 500ms blev fundt at forbedre optagelse af start- og sluttider for én palle (se 7.2.8), men dette holdte ikke, når flere paller med ordrer blev introduceret (se 7.2.9). Endeligt var det besluttet at øge forsinkelsen til 550ms, og dette førte til næsten ideal funktion⁸ (se 7.2.10).

⁸ Mærk den ikke optaget besøg til STPLC_13 af ordre 6095.



Figur 30 Konfigurerer af: (til venstre) - a) CSV-node, (centrum) - b) Change-node, (til højre) - c) Write File-node.

4. Analyse

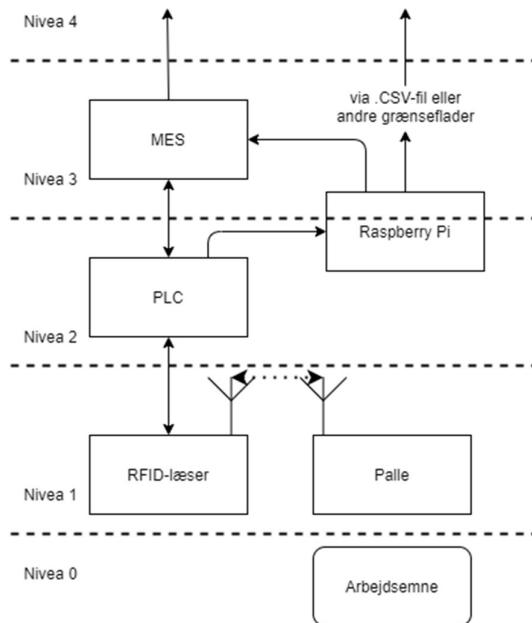
I denne afsnit findes der analyse og vurdering af løsningen med hensyn til projektets problemformulering. Første håndteres der vurdering af løsningens vertikale dataintegration. Næste er der analyse om løsningens funktion mht. ISA-95. Til sidste kommer der analyse af løsningens eventuelle betydning for energimanagement og bæredygtighed i produktions- og fremstillingsvirksomheder.

4.1 Vurdering af den vertikale dataintegration

Analysen skal begynde med vurdering af løsningens position ift. de andre dele af Smart Production Labs anlæg, og derefter vurderes datastrømmen.

4.1.1 Løsningens fysisk forbindelse med anlægget

Løsningens position skal forstås i denne kontekst som dens fysiske forbindelse til de andre systemkomponenter på anlægget. Figur 31 sammensætter denne fysiske placering og de funktionelle niveauer fra standarden (se eventuelt Figur 4) [23]. Løsningen er forbundet med systemet ved grænsefladen mellem niveau 2 og 3 – dvs. *control*-netværket mellem PLC'erne og MES-computeren – og kan tilgås af enheder forbundet med Smart Production Labs indbygget Wi-Fi – dvs. via *enterprise*-netværket. Dette er vigtigt at forstå, da den giver afgørende kontekst til datastrømmens kilde og vej gennem netværket.



Figur 31 Placering af løsningen i forhold til de andre Smart Production Lab systemkomponenter.

4.1.2 Vurdering af den vertikale datastrømme

Datastrømmen stammer fra interaktionen mellem *RFID-tags* på pallerne og *RFID-læserne* ved stationerne. Selvom der er andre sensorer og aktuatorer, som direkte påvirker arbejdsemnet, er disse processer ikke betragtede i dette projekt, og så er pallerne systemets kontaktfylde med processen. Dette er vigtigt at tage stilling til, da datastrømmen skal ikke misforstås som en direkte led fra processen.

Kommunikation sker mellem flere par af enheder: *RFID-læserne* og *-tags*, *RFID-læserne* og *PLC'erne*, *PLC'erne* og *MES'et*, *PLC'erne* og løsningen, og mellem løsningen og andre enheder. Mens de fleste af disse kommunikationer strømmer begge vej, er der nuværende kun envejskommunikation mellem løsningen og andre enheder – fra *PLC'erne* til løsningen og fra løsningen til andre enheder (både via løsningens *dashboard*⁹ og via eksport af *CSV-filer*). Dette skal betragtes yderligere, når løsningens funktion skal vurderes.

4.2 Løsnings funktionelanalyse

Analysen forgår med hensyn til *ISA-95*. Første omtales det funktionelle hierarki og løsnings placering derpå (se eventuelt Figur 4), og bagefter er løsningens funktionalitet vurderet ift. *ISA-95*. Næste omtales *IIoT* analyse af løsningens funktionalitet, og til sidst er der diskussion om mulige forbedringer på enhedens funktion.

4.2.1 Løsningens placering på det funktionelle hierarki

Placering af løsningens funktionalitet på det funktionelle hierarki er svært. På den ene hånd er enheden direkte involveret i produktion, da den opsamler procestidscyklusdata direkte fra styreenheden – dvs. *PLC'en* – og fordi den lagrer disse data, som man kan betragte som udstyrsinformation. Imidlertid er enhedens funktion ikke kritisk for produktionen eller dens effektivitet, og så passer den ikke den definition for niveau 1, 2, eller 3 fandt i klausul 5.2.2 [23]. Til gengæld vedrører enhedens funktionalitet områder som niveau 3 aktiviteter typisk beskæftiger sig med, ifølge klausul 5.3 – dvs. enheden bruges til *site*, *area*, eller *workcenter* dataopsamling. Derudover udfører enheden produktionsdataopsamling og produktionssporing, samt bedrager til procesmanagement og performanceanalyse, som betragtes som niveau 3 aktiviteter ifølge klausul 5.2.4 [23] og 6.9 og 6.10 [25]. Til sidste har løsningen direkte grænseflader med niveau 2, da den tilgår *PLC'ernes OPC UA-servers*, og niveau 4 aktiviteter, da den muliggør eksport af opsamlede produktionsdata via *CSV-fil*. Så kan man sige at enheden udfører niveau 2 og 3 funktioner med grænseflader til niveau 2 og 4. Imidlertid er enhedens niveau 3 funktioner ikke fuld udviklet – dvs. at

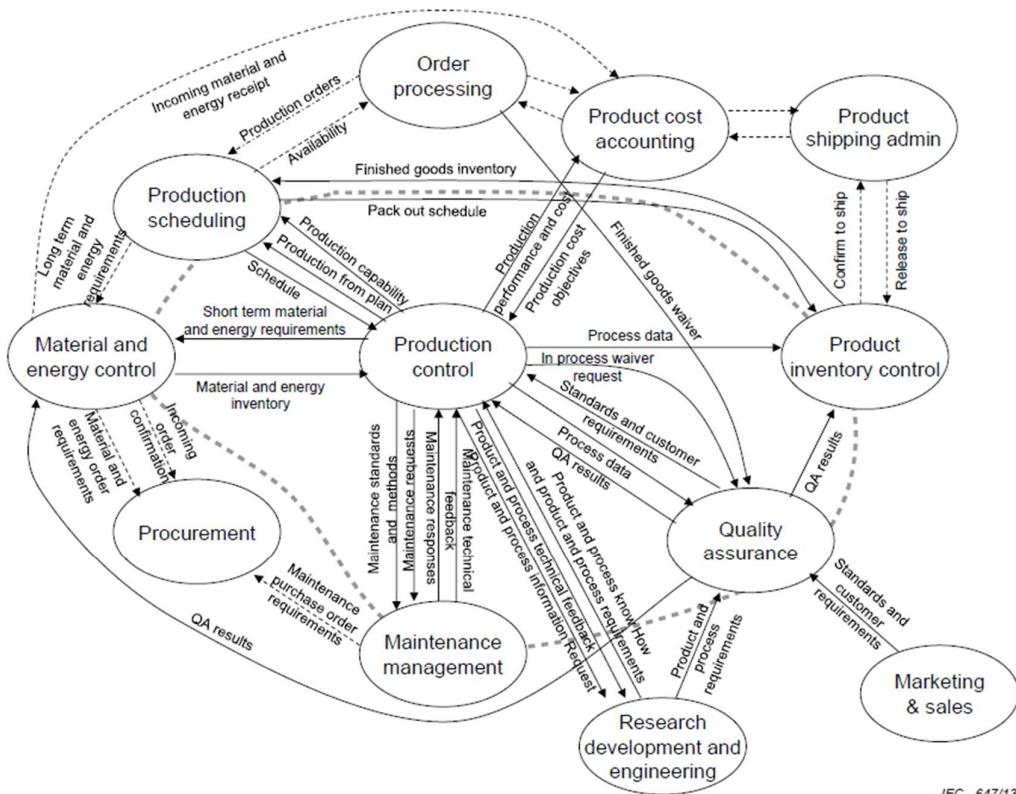
⁹ Selvom der er ordresøgningsfunktion og andre funktioner, som kan kræves fra andre enheder, er der ingen mulighed at sende data denne vej. Så betragtes datastrømmen som envejs.

den fungerer som et integreret *datalogger* – kan det ikke placeres helt op på niveau 3, og derfor skulle betragtes som grænseoverskridende mellem niveau 2 og 3 på det funktionelle hierarki.

4.2.2 Funktionalitet mht. ISA-95

Løsningens funktionalitet blev kort berørt ovenfor, da dens placering på det funktionelle hierarki er tæt forbundet med dens funktionalitet. Her udforskes funktionaliteten videre ift. ISA-95, og de tidligere nævnte funktioner uddybes mht. den funktionelle model viste på Figur 32.

Løsningens funktion bedrager direkte til datastrømmen mellem produktionsstyrings- og produktomkostningsregnskabsaktiviteter, da procestidscyklosdataene er en del af produktionsperformance og omkostningsdatastrømme ifølge klausul 6.5.11 [23] – tidscyklusdataene bruges til beregning af KPI'er (såsom *OEE* performance) og til vurdering af produktionsomkostning i tilfældet at disse data kan sammensættes med energiforbrugssdata.



IEC 647/13

Figur 32 Den funktionelle model [23].

Løsningen understøtter indirekte datastrømmen mellem produktionsstyrings- og kvalitetssikringsaktiviteter ifølge klausul 6.5.19, samt mellem produktionsstyrings- og vedligeholdelsesmanagementsaktiviteter ifølge klausul 6.5.22. I det første tilfælde opsamler enheden procescyklustidsdata, som kan bruges til kvalitetsvurdering i processer, hvor eksponeringstider har en væsentlig indvirkning på produktets kvalitet – f.eks. pasteuriserings- eller varmebehandlingsanlæg.

Datastrømmen mellem produktionsstyrings og vedligeholdelsesmanagementsaktiviteter er endnu mere indirekte end de to tidligere nævnte strømme, fordi der er normalt ingen automatisk overførsel af procescyklustidsdata til vedligeholdelsesformål. Imidlertid kan analyse af disse data bidrage til rodårsagsanalyse og identificering af problemer på maskiner eller i produktionslinjer, som kan bidrage til et prædiktivvedligeholdelsesprogram. Derudover kan analyse af cyklustidsdata understøtte afvigelsesmanagementsaktiviteter, såsom konsekvensanalyse, ifølge annex A.7.

Ifølge klausul 8.3 kan løsningen yderligere bidrage til skabelse af produktionsevneinformation udover den tidligere nævnte produktionsperformanceinformation. Dette skyldes at procestidscyklusdata er afgørende for nøjagtig estimering af produktionskapacitet og for identificering af *golden runs*.

Til sidste skal det omtales, hvorvidt løsningen overholder *ISA-95* eller ej. Hver del af *ISA-95* har dens egne overholdelseskriterier. På grund af løsningens meget begrænset omfang er flere af disse dele ikke relevante for dette projekt. Imidlertid kan det siges at løsningen er delvis overholdende ift. del 1 og del 3.

4.2.3 Funktionalitetsanalyse med hensyn til *I/IoT*

Med hensyn til *I/IoT* er enhedens funktion delvis begrænset. Selvom den gør brug af *IoT* computerhardware, sensorer (*RFID-tags* og -læsere), kommunikationer (*RFID*, og identificering (*IPv4*-adresser), og den udfører *IoT services* – nemlig sammenlægning af produktionsdata – gør den ikke brug af *cloud*-løsninger eller and *IoT*-elementer [19]. Imidlertid er der muligheden at udvide enheden med disse forbindelser, og man kan sige at løsningens *dashboard* udføre *application*-lag funktioner [38].

4.2.4 Yderligere udviklinger

Løsningens funktion kunne udvides med overvågning af flere procesvariabler, mere behandling af dataene eller tættere integration med niveau 3 og 4 programmer. Overvågning af flere variabeller, såsom resultater fra en kvalitetskontrolstation, kunne udvide enhedens bidragelse til beregning af *OEE KPI*'er uden markant ændringer, da det kræver kun én ekstra *subscription* for én station. Dette kunne udvikles videre med yderligere databehandling på enheden – f.eks. estimering af *OEE KPI*'er under produktion – som kunne hjælpe operatører at gribe ind i processen, hvis der kommer problemer. Til sidst kunne enheden integreres tætter med *MES*- og *ERP*-software for at gøre opsamlingen og analysen endnu mere automatiseret.

4.3 Analyse af løsningen mht. energimanagement og bæredygtighed

Selvom projektet var ikke rettet mod energimanagementsformål, kan løsningen vurderes indenfor den energimanagementramme. Her deles analysen mellem løsningens nuværende betydning for energimanagementsformål og potentialet med yderligere udvikling af enheden.

4.3.1 Løsningens nuværende betydning for energimanagement

Løsningens primære værdi mht. energimanagement er den opsamling af procescyklustidsdata, som kan bidrage til analyse af *Significant Energy Users (SEUs)* i produktions- og fremstillingsvirksomheder. Hvis cyklustidsdataene sammensættes med energiforbrugsdata, kan en energimanager mere granulært vurdere *Specific Energy Consumption (SEC)* [59], og dermed skaber en mere nøjagtig billede af produktionens energiforbrug. Derudover bidrager enheden til udførelse af energiaudit, da energimanageren behøver ikke at opsamle dataene selv – løsningen udfører opsamlingen og energimanageren kan fjerneksportere dataene fra et netværksadgangspunkt. Så har enheden væsentlig betydning for udførelse af en *Energy Management System (EnMS)* [60].

4.3.2 Yderligere Potential

Løsningens bidragelse to et *EnMS* kunne udvidelse endnu mere med yderligere integration af datastrømme. Energi *KPI*'er kunne estimeres med enten integrationen med eksisterende energimålere på anlægget eller tilføjelse af sensorer til energidataopsamlingsformål – mulige sensorer kunne måle bl.a. el-, procesvarme- eller trykluftsforbrug. Derudover kunne integration med cloud løsninger muliggøre dybere og automatisk analyse af dataene og muliggøre fjerneadgang til dataene for virksomheder med flere produktionssteder.

5. Konklusioner og Perspektivering

5.1 Konklusioner

Her omtales konklusioner til problemformuleringen og underspørgsmålene.

5.1.1 Hovedspørgsmål

For at svare på problemformuleringen skal enheden betragtes som en demonstrant for både begreber og teknologier.

I den ene ramme kan man sige at løsningen muliggør *IIoT* med at udvide det eksisterende anlægs funktionalitet med procescyklustidsdataopsamling og med at tilbyde et forbindelsespunkt for yderligere programmer – dvs. *ERP*-software – og at muliggøre eksport af data via *CSV*-fil. Yderligere gør løsningen pallerne til *IIoT*-objekter ud over deres eksisterende funktion – dvs. at løsningen gør pallerne tilgængelige til andre programmer ud over *PLC*'erne.

I den anden ramme gør løsning brug af flere *IIoT* teknologier, såsom *RFID*-tags og -læsere, *Raspberry Pi* computere, *MQTT* og *OPC UA*, til værdiskabende formål – dvs. cyklustidsdataopsamling til procesperformanceanalyse og omkostningsvurdering. Derudover gav udskiftning af SIA-Connect muligheden at afprøve *Node-REDs* brugervenlighed for undersøgende projekter hos SMV'er.

Derfor opfylder løsningen opgavens mål at demonstrere mulige anvendelser af teknologierne og begreber for industripersonelle og -ledere med fokus på værdiskabende aktiviteter og forretningsmål.

5.1.2 Underspørgsmål

Så kan underspørgsmålene omtales for at udforske konklusioner videre.

Relevans af ISA-95

Der er nogle der har kaldte *ISA-95* for 'for rigid' mht. udviklinger i *IIoT* og Industri 4,0 [29]. Imidlertid var dette ikke oplevelsen i dette projekt, da standarden handler om definitioner og modeller for afgørende produktionsaktiviteter og repræsentation af data og ikke om faste krav til funktion¹⁰, og dette bekræfter andre forskning fra CIP om emnet [10]. Imidlertid var projektets omfang meget begrænset ift. størrelsen og retning, og et større eller mere dybtgående projekt måtte støde sammen med standarden.

Betydning for energimanagement

Det var ikke muligt at udforske projektets betydning for energimanagement i dybden, men der er nogle fremtrædende observationer i forhold til løsningens muliggørende effekt på et *EnMS*. I sin rolle som dataopsamlingsenhed kan løsningen bidrage til beregning af mere nøjagtige *energiperformanceindikatorer (EnPI)* – med mere granulære data om procescykler kan man beregne

¹⁰ Mærk at dette gælder ikke for dele 5 & 6, da de handler om transaktioner mellem forretning og fabrikation og MSM-tjenester, og er nødvendigvis mere streng om funktioner.

mere granulært forbrug af energi, og dermed mere nøjagtige *EnPI*'er. Derudover kan løsningen reducere arbejdskraftskravet til energiaudit, da den fjerne behovet at optage procesdata manuelt – dette kan gøre *EnMS* og energipolitik mere tilgængelige for SMV'er med at reducere omkostninger af disse aktiviteter. Til sidst kan enheden, med ganske få ændringer, integreres med eksisterende systemer enten over *enterprise*-netværket eller internettet via *cloud*-integration, og på den måde øge *EnMS*'ets evne til at overvåge energiforbruget og identificere afvigelser, som kunne ske pga. manglede vedligehold f.eks.

5.2 Perspektivering af projektet

Til sidste skal projektet perspektiveres ift. den yderligere kontekst. Dette omtales mht. bidraget til litteraturen, fremme af det overordnede projekt og muligheder for yderligere forskning.

Projektet bekræfter nogle af de konklusioner om *ISA-95*s utilitet ift. industri 4,0 og *IIoT* projekter som har resulteret af CIPs forskning [9], [11]. Der var ikke noget særlige betydelige tilføjelser til litteraturen ift. energimanagement, som fulgte dette projekt. Imidlertid, var der identificeret nogle muligheder for yderligere udvikling af demonstratorer, som kunne udforske emnet videre.

Dette projekt har dog bidraget betydeligt til Innovation Factory Norths mål at udforske og presentærer vertikale dataintegration via *IIoT* og Industri 4,0 teknologier for danske SMV'er. Denne demonstrator kan vise industripersonelle og -ledere, hvor let man kan få gang i *IIoT* projekter til dataopsamlingsformål, og giver forhåbentligt inspiration til deres egne pilotprojekter.

Ud over disse bidragelse har projektet identificeret nogle muligheder for yderligere forskning. For det ene er der brug for udforskning og udvikling om hvilke *ISA-95* produktions- og forretningsaktiviteter er mest udbredt i SMV'er. Dette ville hjælpe med at lede udvikling af relevante og værdiskabende demonstratorer for SMV'er – demonstratorerne skulle fokusere på fremvisning af løsninger til relevante problemer. For det andet er der behov for yderligere forskning i *IoT* og Industri 4,0s betydning for energimanagement i produktions- og fremstillings virksomheder, og hvordan disse teknologier kan gøre energimanagementsindsatser mere tilgængelige og værdiskabende for danske SMV'er.

6. Litteraturliste

- [1] S. Y. Sørensen, "Internet of Things på vej til at blive hverdag," 2019.
- [2] MADE - Manufacturing Academy of Denmark, "Om MADE." [Online]. Available: <https://www.made.dk/om-made/>. [Accessed: 29-Nov-2021].
- [3] MADE - Manufacturing Academy of Denmark, "Vi skaber god vækst ...".
- [4] Aalborg University, "CENTER FOR INDUSTRIAL PRODUCTION." [Online]. Available: <https://www.en.cip.aau.dk/>. [Accessed: 05-Nov-2021].
- [5] Aalborg University, "ABOUT CENTER FOR INDUSTRIAL PRODUCTION." [Online]. Available: <https://www.en.cip.aau.dk/about/>. [Accessed: 05-Nov-2021].
- [6] Aalborg University, "COLLABORATION." [Online]. Available: <https://www.en.cip.aau.dk/collaboration/>. [Accessed: 05-Nov-2021].
- [7] Aalborg University, "PROGRAM." [Online]. Available: <https://www.smartproduction.aau.dk/Program/>. [Accessed: 05-Nov-2021].
- [8] Aalborg University, "INNOVATION FACTORY NORTH." [Online]. Available: <https://www.smartproduction.aau.dk/Factory/>. [Accessed: 05-Nov-2021].
- [9] C. Li, S. Mantravadi, C. Schou, H. Nielsen, O. Madsen, and C. Møller, *An ISA-95 based Middle Data Layer for Data Standardization—Enhancing Systems Interoperability for Factory Automation*, no. c. Springer Berlin Heidelberg, 2021.
- [10] D. Palade, C. Moller, C. Li, and S. Mantravadi, "An Open Platform for Smart Production: IT/OT Integration in a Smart Factory," vol. 2, no. Icveis, pp. 707–714, 2021.
- [11] S. Mantravadi, C. Møller, C. LI, and R. Schnyder, "Design choices for next-generation IIoT-connected MES/MOM: An empirical study on smart factories," *Robot. Comput. Integr. Manuf.*, vol. 73, p. 102225, 2022.
- [12] S. Mantravadi and C. Møller, "An overview of next-generation manufacturing execution systems: How important is MES for industry 4.0?," *Procedia Manuf.*, vol. 30, pp. 588–595, 2019.
- [13] Aalborg University, "PHD PROJEKTER." [Online]. Available: <https://www.ifn.aau.dk/PhD+Projekter/>. [Accessed: 05-Nov-2021].
- [14] Aalborg University, "AAU SMART PRODUCTION LABORATORY." [Online]. Available: <https://www.smartproduction.aau.dk/Laboratory/>. [Accessed: 05-Nov-2021].
- [15] Festo, "CP Factory – The Cyber-Physical Factory." [Online]. Available: <https://www.festo-didactic.com/int-en/learning-systems/learning-factories,cim-fms-systems/cp-factory/cp-factory-the-cyber-physical-factory.htm?fbid=aW50LmVuLjU1Ny4xNy4xOC4xMjkzLjc2NDM>. [Accessed: 30-Dec-2021].

- [16] Aalborg University, "PROJEKTBESKRIVELSE." [Online]. Available: <https://www.ifn.aau.dk/Projektbeskrivelse/>. [Accessed: 05-Nov-2021].
- [17] H. Boyes, B. Hallaq, J. Cunningham, and T. Watson, "The industrial internet of things (IIoT): An analysis framework," *Comput. Ind.*, vol. 101, no. June, pp. 1–12, 2018.
- [18] A. Thierer and A. Castillo, "Projecting the Growth and Economic Impact of the Internet of Things." Mercatus Center, pp. 1–10, 2015.
- [19] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," *IEEE Commun. Surv. Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.
- [20] A. Čolaković and M. Hadžalić, "Internet of Things (IoT): A review of enabling technologies, challenges, and open research issues," *Comput. Networks*, vol. 144, pp. 17–39, 2018.
- [21] S. Nylander, A. Wallberg, and P. Hansson, "Challenges for SMEs entering the IoT world - Success is about so much more than technology," *ACM Int. Conf. Proceeding Ser.*, 2017.
- [22] A. Moeuf, R. Pellerin, S. Lamouri, S. Tamayo-Giraldo, and R. Barbaray, "The industrial management of SMEs in the era of Industry 4.0," *Int. J. Prod. Res.*, vol. 56, no. 3, pp. 1118–1136, 2018.
- [23] Dansk Standard, "DS/EN 62264-1 Integration af virksomheders styrings- systemer – Del 1 : Modeller og terminologi," 2013.
- [24] Dansk Standard, "DS/EN 62264-2 Integration af virksomhedens styringssystem – Del 2 : Objekter og attributter til integration af virksomhedens styringssystem," 2013.
- [25] Dansk Standard, "DS/EN 62264-3 Integration af virksomhedens styrings- system – Del 3 : Aktivitetsmodeller til brug ved styring af fremstillings- processer models of manufacturing operations management," 2017.
- [26] Dansk Standard, "DS/EN 62264-4 Integration af virksomhedens styrings- system – Del 4 : Objektmodelattributter til integration af styring af fabrikations- processer Enterprise-control system integration – operations management integration," 2016.
- [27] Dansk Standard, "DS/EN 62264-5 Integration af virksomhedssystemer – Del 5 : Transaktioner mellem forretning og fabrikation," 2012.
- [28] Dansk Standard and International Organisation for Standardization, *DS/IEC 62264-6 Integration af virksomhedens styringssystem – Del 6 : MSM-tjenester Enterprise-control system integration – Part 6 : Messaging service model*. 2020.
- [29] M. Foehr, J. Vollmar, A. Calà, P. Leitão, S. Karnouskos, and A. W. Colombo, "Engineering of Next Generation Cyber-Physical Automation System Architectures BT - Multi-Disciplinary Engineering for Cyber-Physical Production Systems: Data Models and Software Solutions for

- Handling Complex Engineering Projects,” S. Biffl, A. Lüder, and D. Gerhard, Eds. Cham: Springer International Publishing, 2017, pp. 185–206.
- [30] Vorne, “WHAT IS OVERALL EQUIPMENT EFFECTIVENESS?” [Online]. Available: <https://www.oee.com/>. [Accessed: 09-Jun-2021].
 - [31] Vorne Industries Inc., “CALCULATE OEE.” [Online]. Available: <https://www.oee.com/calculating-oee/>. [Accessed: 30-Dec-2021].
 - [32] OPC Foundation, “Unified Architecture.” [Online]. Available: <https://opcfoundation.org/about/opc-technologies/opc-ua/>. [Accessed: 30-Dec-2021].
 - [33] MQTT.org, “MQTT: The Standard for IoT Messaging.” [Online]. Available: <https://mqtt.org/>. [Accessed: 05-Jan-2021].
 - [34] M. Yuan, “Getting to know MQTT.” 2017.
 - [35] OASIS, “OASIS Message Queuing Telemetry Transport (MQTT) TC - Overview.” [Online]. Available: https://www.oasis-open.org/committees/tc_home.php?wg_abbrev= mqtt#overview. [Accessed: 05-Jan-2022].
 - [36] OASIS, “About Us.” [Online]. Available: <https://www.oasis-open.org/org/>. [Accessed: 05-Jan-2022].
 - [37] Raspberry Pi Foundation, “About Us.” [Online]. Available: <https://www.raspberrypi.org/about/>. [Accessed: 05-Jan-2022].
 - [38] B. N. Silva, M. Khan, and K. Han, “Internet of Things: A Comprehensive Review of Enabling Technologies, Architecture, and Challenges,” *IETE Tech. Rev. (Institution Electron. Telecommun. Eng. India)*, vol. 35, no. 2, pp. 205–220, 2018.
 - [39] X. Zhong and Y. Liang, “Raspberry Pi: An effective vehicle in teaching the internet of things in computer science and engineering,” *Electron.*, vol. 5, no. 3, 2016.
 - [40] S. Kurkovsky and C. Williams, “Raspberry Pi as a platform for the internet of things projects: Experiences and lessons,” *Annu. Conf. Innov. Technol. Comput. Sci. Educ. ITiCSE*, vol. Part F1286, pp. 64–69, 2017.
 - [41] OpenJS Foundation, “About.” [Online]. Available: <https://nodered.org/about/>. [Accessed: 05-Jan-2022].
 - [42] Mikakaraila, “node-red-contrib-opcua 0.2.254.” [Online]. Available: <https://flows.nodered.org/node/node-red-contrib-opcua>. [Accessed: 30-Oct-2021].
 - [43] Siemens, “Learn- / Training Document: OPC UA with Simatic S7-1500 and Node-RED,” pp. 1–42, 2019.
 - [44] L. Castro and S. F. Wamba, “An Inside Look at RFID Technology,” *J. og Technol. Manag. Innov.*, vol. 2, no. 1, pp. 128–141, 2007.

- [45] L. Da Xu, E. L. Xu, and L. Li, “Industry 4.0: State of the art and future trends,” *Int. J. Prod. Res.*, vol. 56, no. 8, pp. 2941–2962, 2018.
- [46] R. Want, “An Introduction to RFID Technology,” *Pervasive Comput.*, pp. 1–3, 2006.
- [47] R. Weinstein, “RFID: A technical overview and its application to the enterprise,” *IT Prof.*, vol. 7, no. 3, pp. 27–33, 2005.
- [48] Festo, “Workpiece carrier.” [Online]. Available: <https://www.festo-didactic.com/int-en/learning-systems/factory-automation-industry-4.0/learning-factory-kits/cp-workpieces/workpiece-carrier.htm?fbid=aW50LmVuLjU1Ny4xNy4xOC4xODQ4LjUzNzc2>. [Accessed: 06-Jan-2022].
- [49] SIA, “SIA-Connect Gateway.” [Online]. Available: <https://sia-connect.com/>. [Accessed: 07-Jan-2022].
- [50] SIA, “Supported Industrial and Buildings Devices.” [Online]. Available: https://help.sia-connect.com/en_US/intro-to-sia/supported-industrial-buildings-devices?_ga=2.58428968.389318503.1641563211-549125804.1638356174. [Accessed: 07-Jan-2022].
- [51] SIA, “Product.” [Online]. Available: <https://sia-connect.com/product/>. [Accessed: 07-Jan-2022].
- [52] SQLite Consortium, “What is SQLite?” [Online]. Available: <https://www.sqlite.org/index.html>. [Accessed: 08-Jan-2022].
- [53] SQLite Consortium, “Most Widely Deployed and Used Database Engine.” [Online]. Available: <https://www.sqlite.org/mostdeployed.html>. [Accessed: 08-Jan-2022].
- [54] SQLite Consortium, “About SQLite.” [Online]. Available: <https://www.sqlite.org/about.html>. [Accessed: 08-Jan-2022].
- [55] SQLite Consortium, “SQLite Is A Zero-Configuration Database.” [Online]. Available: <https://www.sqlite.org/zeroconf.html>. [Accessed: 08-Jan-2022].
- [56] Unified Automation Gmbh, “UaExpert—A Full-Featured OPC UA Client.” [Online]. Available: <https://www.unified-automation.com/products/development-tools/uaexpert.html>. [Accessed: 08-Jan-2022].
- [57] F. D. Rolland, *The Essence of Databases*. USA: Prentice Hall PTR, 1998.
- [58] knoleary and dceejay, “node-red-node-sqlite 1.0.3.” [Online]. Available: <https://flows.nodered.org/node/node-red-node-sqlite>. [Accessed: 15-Nov-2021].
- [59] A. Lawrence, P. Thollander, M. Andrei, and M. Karlsson, “Specific energy consumption/use (SEC) in energy management for improving energy efficiency in industry: Meaning, usage and differences,” *Energies*, vol. 12, no. 2, 2019.

- [60] Dansk Standard and International Organisation for Standardization, “DS/EN ISO 50001:2018 Energy management systems – Requirements.” 2018.

7. Bilag

7.1 OPC UA Servers og Variablers *Identifiers*

Tabel 1 OPC UA Server Lookup-Tabeller.

Station Number:	STPLC_01	IP Address:	172.20.8.1
Variable	Namespace Index	Datatype	Identifier
Infeed			
xCarrierAvailable	2	String	var CECC-LK.Application.FBs.swSwitch1.stpStopper cpfssStopper.xCarrierAvailable
udiONo	2	String	var CECC-LK.Application.FBs.swSwitch1.udiONo
uiCarrierId	2	String	var CECC-LK.Application.FBs.swSwitch1.uiCarrierId
Return			
xCarrierAvailable	2	String	var CECC-LK.Application.FBs.tstpStopper2.stpStopper.xCarrierAvailable
Outfeed			
xCarrierAvailable	2	String	var CECC-LK.Application.FBs.stpStopper1.stpStopper cpfssStopper.xCarrierAvailable
udiONo	2	String	var CECC-LK.Application.FBs.stpStopper1.udiONo
uiCarrierId	2	String	var CECC-LK.Application.FBs.stpStopper1.uiCarrierId
Station Number:	STPLC_08	IP Address:	172.20.1.1
Variable	Namespace Index	Datatype	Identifier
xCarrierAvailable	2	String	var CECC-LK.Application.FBs.stpStopper1.stpStopper cpfssStopper.xCarrierAvailable
udiONo	2	String	var CECC-LK.Application.FBs.stpStopper1.udiONo
uiCarrierId	2	String	var CECC-LK.Application.FBs.stpStopper1.uiCarrierId
Station Number:	STPLC_09	IP Address:	172.20.13.1
Variable	Namespace Index	Datatype	Identifier
xCarrierAvailable	2	String	var CECC-LK.Application.FBs.stpStopper1.stpStopper cpfssStopper.xCarrierAvailable
udiONo	2	String	var CECC-LK.Application.FBs.stpStopper1.udiONo
uiCarrierId	2	String	var CECC-LK.Application.FBs.stpStopper1.uiCarrierId
Station Number:	STPLC_10	IP Address:	172.20.3.1

Variable	Namespace Index	Datatype	Identifier
xCarrierAvailable	2	String	var CECC-LK.Application.FBs.stpStopper1.stpStopper cpfssStopper.xCarrierAvailable
udiONo	2	String	var CECC-LK.Application.FBs.stpStopper1.udiONo
uiCarrierId	2	String	var CECC-LK.Application.FBs.stpStopper1.uiCarrierId
Station Number:	STPLC_11	IP Address:	172.20.11.1
Variable	Namespace Index	Datatype	Identifier
xCarrierAvailable	2	String	var CECC-LK.Application.FBs.stpStopper1.stpStopper cpfssStopper.xCarrierAvailable
udiONo	2	String	var CECC-LK.Application.FBs.stpStopper1.udiONo
uiCarrierId	2	String	var CECC-LK.Application.FBs.stpStopper1.uiCarrierId
Station Number:	STPLC_12	IP Address:	172.20.16.1
Variable	Namespace Index	Datatype	Identifier
xCarrierAvailable	2	String	var CECC-LK.Application.FBs.stpStopper1.stpStopper cpfssStopper.xCarrierAvailable
udiONo	2	String	var CECC-LK.Application.FBs.stpStopper1.udiONo
uiCarrierId	2	String	var CECC-LK.Application.FBs.stpStopper1.uiCarrierId
Station Number:	STPLC_13	IP Address:	172.20.5.1
Variable	Namespace Index	Datatype	Identifier
xCarrierAvailable	2	String	var CECC-LK.Application.FBs.stpStopper1.stpStopper cpfssStopper.xCarrierAvailable
udiONo	2	String	var CECC-LK.Application.FBs.stpStopper1.udiONo
uiCarrierId	2	String	var CECC-LK.Application.FBs.stpStopper1.uiCarrierId
Station Number:	STPLC_14	IP Address:	172.20.4.1
Variable	Namespace Index	Datatype	Identifier
Switch			
xCarrierAvailable	2	String	var CECC-LK.Application.FBs.swSwitch1.stpStopper cpfssStopper.xCarrierAvailable
udiONo	2	String	var CECC-LK.Application.FBs.swSwitch1.udiONo
uiCarrierId	2	String	var CECC-LK.Application.FBs.swSwitch1.uiCarrierId
Outfeed			

xCarrierAvailable	2	String	var CECC-LK.Application.FBs.tstpStopper2.stpStopper.xCarrierAvailable
Process (Robot)			
xCarrierAvailable	2	String	var CECC-LK.Application.FBs.stpStopper1.stpStopper.cpfssStopper.xCarrierAvailable
udiONo	2	String	var CECC-LK.Application.FBs.stpStopper1.udlONo
uiCarrierId	2	String	var CECC-LK.Application.FBs.stpStopper1.uiCarrierId

7.2 Resultater af funktionelle afprøvninger

7.2.1 Udvikling af *OPC UA Clients*

```

debug
all nodes | all

▶ { ProcessType: "STPLC_08",
  xCarrierAvailable: "false",
  TimeStamp: 1639769533786, OrderID:
  "0" }

12/21/2021, 4:43:57 PM node: bc64dd4c13bf34a8
STPLC_08/xCarrierAvailable : msg.payload : string[4]
"true"

12/21/2021, 4:43:57 PM node: bc64dd4c13bf34a8
OrderID : msg.payload : Object
  ▶ { ProcessType: "STPLC_08",
    xCarrierAvailable: "true", TimeStamp:
    1639769557410, OrderID: "0" }

12/21/2021, 4:43:57 PM node: bc64dd4c13bf34a8
STPLC_08/xCarrierAvailable : msg.payload : string[5]
"false"

12/21/2021, 4:43:57 PM node: bc64dd4c13bf34a8
OrderID : msg.payload : Object
  ▶ { ProcessType: "STPLC_08",
    xCarrierAvailable: "false",
    TimeStamp: 1639769558028, OrderID:
    "0" }

12/21/2021, 4:44:23 PM node: bc64dd4c13bf34a8
STPLC_08/xCarrierAvailable : msg.payload : string[4]
"true"

12/21/2021, 4:44:23 PM node: bc64dd4c13bf34a8
OrderID : msg.payload : Object
  ▶ { ProcessType: "STPLC_08",
    xCarrierAvailable: "true", TimeStamp:
    1639769583321, OrderID: "0" }

12/21/2021, 4:44:23 PM node: bc64dd4c13bf34a8
STPLC_08/xCarrierAvailable : msg.payload : string[5]
"false"

12/21/2021, 4:44:23 PM node: bc64dd4c13bf34a8
OrderID : msg.payload : Object
  ▶ { ProcessType: "STPLC_08",
    xCarrierAvailable: "false",
    TimeStamp: 1639769583893, OrderID:
    "0" }

```

Figur 33 Debug output fra funktionelafprøvning af OPC UA Clients

7.2.2 Første funktionel afprøvning på udvidede OPC UA-Clients

```

12/21/2021, 5:22:09 PM node: 195aaa2a61d80fe2
OrderID : msg.payload : Object
  ↳ { ProcessType: "STPLC_10",
    xCarrierAvailable: "true", TimeStamp:
    1639771849605, OrderID: "6069" }

12/21/2021, 5:22:11 PM node: 195aaa2a61d80fe2
OrderID : msg.payload : Object
  ↳ { ProcessType: "STPLC_10",
    xCarrierAvailable: "false",
    TimeStamp: 1639771851267, OrderID:
    "6069" }

12/21/2021, 5:22:12 PM node: 195aaa2a61d80fe2
OrderID : msg.payload : Object
  ↳ { ProcessType: "STPLC_10",
    xCarrierAvailable: "true", TimeStamp:
    1639771852822, OrderID: "6069" }

12/21/2021, 5:22:19 PM node: 195aaa2a61d80fe2
OrderID : msg.payload : Object
  ↳ { ProcessType: "STPLC_10",
    xCarrierAvailable: "false",
    TimeStamp: 1639771859472, OrderID:
    "6070" }

12/21/2021, 5:22:20 PM node: 195aaa2a61d80fe2
OrderID : msg.payload : Object
  ↳ { ProcessType: "STPLC_10",
    xCarrierAvailable: "true", TimeStamp:
    1639771860960, OrderID: "6070" }

12/21/2021, 5:22:22 PM node: 446b88a38e34777b
OrderID : msg.payload : Object
  ↳ { ProcessType: "STPLC_12",
    xCarrierAvailable: "false",
    TimeStamp: 1639771862873, OrderID:
    "6069" }

12/21/2021, 5:22:26 PM node: 195aaa2a61d80fe2
OrderID : msg.payload : Object
  ↳ { ProcessType: "STPLC_10",
    xCarrierAvailable: "false",
    TimeStamp: 1639771867054, OrderID:
    "6071" }

12/21/2021, 5:22:28 PM node: 195aaa2a61d80fe2
OrderID : msg.payload : Object
  ↳ { ProcessType: "STPLC_10",
    xCarrierAvailable: "true", TimeStamp:
    1639771880062, OrderID: "6071" }

12/21/2021, 5:22:29 PM node: 446b88a38e34777b
OrderID : msg.payload : Object
  ↳ { ProcessType: "STPLC_12",
    xCarrierAvailable: "true", TimeStamp:
    16397718868580, OrderID: "6071" }

12/21/2021, 5:22:30 PM node: 446b88a38e34777b
OrderID : msg.payload : Object
  ↳ { ProcessType: "STPLC_12",
    xCarrierAvailable: "false",
    TimeStamp: 1639771889833, OrderID:
    "6069" }

12/21/2021, 5:22:30 PM node: 446b88a38e34777b
OrderID : msg.payload : Object
  ↳ { ProcessType: "STPLC_12",
    xCarrierAvailable: "false",
    TimeStamp: 1639771904504, OrderID:
    "6069" }

12/21/2021, 5:22:37 PM node: 446b88a38e34777b
OrderID : msg.payload : Object
  ↳ { ProcessType: "STPLC_12",
    xCarrierAvailable: "true", TimeStamp:
    16397718877591, OrderID: "6070" }

12/21/2021, 5:22:38 PM node: 446b88a38e34777b
OrderID : msg.payload : Object
  ↳ { ProcessType: "STPLC_12",
    xCarrierAvailable: "false",
    TimeStamp: 16397718878505, OrderID:
    "6071" }

12/21/2021, 5:22:39 PM node: 446b88a38e34777b
OrderID : msg.payload : Object
  ↳ { ProcessType: "STPLC_12",
    xCarrierAvailable: "false",
    TimeStamp: 1639771908832, OrderID:
    "6069" }

12/21/2021, 5:23:05 PM node: 195aaa2a61d80fe2
OrderID : msg.payload : Object
  ↳ { ProcessType: "STPLC_13",
    xCarrierAvailable: "true", TimeStamp:
    1639771890062, OrderID: "6071" }

12/21/2021, 5:22:57 PM node: be21e4a2e6472446
OrderID : msg.payload : Object
  ↳ { ProcessType: "STPLC_13",
    xCarrierAvailable: "false",
    TimeStamp: 1639771897344, OrderID:
    "6069" }

12/21/2021, 5:23:04 PM node: be21e4a2e6472446
OrderID : msg.payload : Object
  ↳ { ProcessType: "STPLC_13",
    xCarrierAvailable: "true", TimeStamp:
    1639771916093, OrderID: "6069" }

12/21/2021, 5:23:15 PM node: be21e4a2e6472446
OrderID : msg.payload : Object
  ↳ { ProcessType: "STPLC_13",
    xCarrierAvailable: "false",
    TimeStamp: 1639771916154, OrderID:
    "6071" }

12/21/2021, 5:23:16 PM node: e62026916c7f12b5
OrderID : msg.payload : Object
  ↳ { ProcessType: "STPLC_11",
    xCarrierAvailable: "true", TimeStamp:
    1639771917292, OrderID: "6070" }

12/21/2021, 5:23:22 PM node: e62026916c7f12b5
OrderID : msg.payload : Object
  ↳ { ProcessType: "STPLC_11",
    xCarrierAvailable: "true", TimeStamp:
    1639771922972, OrderID: "6070" }

12/21/2021, 5:23:23 PM node: e62026916c7f12b5
OrderID : msg.payload : Object
  ↳ { ProcessType: "STPLC_11",
    xCarrierAvailable: "false",
    TimeStamp: 1639771923950, OrderID:
    "6071" }

12/21/2021, 5:23:26 PM node: e62026916c7f12b5
OrderID : msg.payload : Object
  ↳ { ProcessType: "STPLC_11",
    xCarrierAvailable: "true", TimeStamp:
    1639771927197, OrderID: "6071" }

12/21/2021, 5:26:09 PM node: a7890be4cac3e2df
msg : error
  ↳ "Error: SQLITE_ERROR: near ";":
    syntax error"

12/21/2021, 5:31:45 PM node: STPLC_14 Order
msg : string[36]
  ↳ "Error stopping node: Close timed"

```

Figur 34 Debug Output fra første funktionel afprøvning af udvidede OPC UA Client-noder.

Orders									
OrderID		Order Received		Throughput Time					
Process Overview									
Search OrderID									
ProcessID	Process Type	OrderID	Start Time	Finish Time					
13	STPLC_11	6071	1639771927197						
12	STPLC_11	6070	1639771922972						
11	STPLC_13	6071	1639771916154						
10	STPLC_11	6069	1639771916093						
9	STPLC_13	6070	1639771911547						
8	STPLC_13	6069	1639771904504						
7	STPLC_12	6071	1639771880062						
6	STPLC_12	6070	1639771877591						
5	STPLC_12	6069	1639771869833						
4	STPLC_10	6071	1639771868580						
3	STPLC_10	6070	1639771860960						
2	STPLC_10	6069	1639771852822						
1	STPLC_10	6069	1639771849605	1639771851267					

Figur 35 UI Output fra fra første funktionel afprøvning af udvidede OPC UA Client-noder.

7.2.3 Anden funktionel afprøvning på udvidede OPC UA-Clients

The figure consists of three separate windows, each titled "debug". Each window has a toolbar with icons for "Deploy", "Stop", "Run", "Break", and "Settings". Below the toolbar are dropdown menus for "all nodes" and "all". The main area of each window displays a list of log entries. The first window on the left shows entries from 12/21/2021, 5:35:14 PM to 5:35:42 PM. The middle window shows entries from 12/21/2021, 5:35:44 PM to 5:36:01 PM. The rightmost window shows entries from 12/21/2021, 5:36:13 PM to 5:36:21 PM. Each entry includes a timestamp, node ID, OrderID, and a detailed payload object containing ProcessType, xCarrierAvailable, TimeStamp, and OrderID.

```

    "Error stopping node: Close timed out"
    12/21/2021, 5:35:14 PM node: 195aaa2a61d80fe OrderID: msg.payload : Object
    > { ProcessType: "STPLC_10", xCarrierAvailable: "false", TimeStamp: 1639772635288, OrderID: "6072" }

    12/21/2021, 5:35:22 PM node: 195aaa2a61d80fe OrderID: msg.payload : Object
    > { ProcessType: "STPLC_10", xCarrierAvailable: "false", TimeStamp: 1639772643301, OrderID: "6073" }

    12/21/2021, 5:35:26 PM node: 446b88a38e34777b OrderID: msg.payload : Object
    > { ProcessType: "STPLC_12", xCarrierAvailable: "false", TimeStamp: 1639772646967, OrderID: "6072" }

    12/21/2021, 5:35:30 PM node: 195aaa2a61d80fe2 OrderID: msg.payload : Object
    > { ProcessType: "STPLC_10", xCarrierAvailable: "false", TimeStamp: 1639772651219, OrderID: "6074" }

    12/21/2021, 5:35:32 PM node: 195aaa2a61d80fe2 OrderID: msg.payload : Object
    > { ProcessType: "STPLC_10", xCarrierAvailable: "false", TimeStamp: 1639772652924, OrderID: "6074" }

    12/21/2021, 5:35:34 PM node: 446b88a38e34777b OrderID: msg.payload : Object
    > { ProcessType: "STPLC_12", xCarrierAvailable: "false", TimeStamp: 1639772655046, OrderID: "6073" }

    12/21/2021, 5:35:42 PM node: 446b88a38e34777b OrderID: msg.payload : Object
    > { ProcessType: "STPLC_13", xCarrierAvailable: "false", TimeStamp: 1639772689630, OrderID: "6074" }

    12/21/2021, 5:35:44 PM node: 446b88a38e34777b OrderID: msg.payload : Object
    > { ProcessType: "STPLC_12", xCarrierAvailable: "false", TimeStamp: 1639772664612, OrderID: "6074" }

    12/21/2021, 5:36:01 PM node: be21e4a2e6472446 OrderID: msg.payload : Object
    > { ProcessType: "STPLC_13", xCarrierAvailable: "false", TimeStamp: 1639772681895, OrderID: "6072" }

    12/21/2021, 5:36:09 PM node: be21e4a2e6472446 OrderID: msg.payload : Object
    > { ProcessType: "STPLC_13", xCarrierAvailable: "false", TimeStamp: 1639772689630, OrderID: "6073" }

    12/21/2021, 5:36:13 PM node: e82026916c7f12b5 OrderID: msg.payload : Object
    > { ProcessType: "STPLC_11", xCarrierAvailable: "false", TimeStamp: 1639772693542, OrderID: "6072" }

    12/21/2021, 5:36:15 PM node: be21e4a2e6472446 OrderID: msg.payload : Object
    > { ProcessType: "STPLC_13", xCarrierAvailable: "false", TimeStamp: 1639772696270, OrderID: "6074" }

    12/21/2021, 5:36:19 PM node: be21e4a2e6472446 OrderID: msg.payload : Object
    > { ProcessType: "STPLC_13", xCarrierAvailable: "false", TimeStamp: 1639772699794, OrderID: "6074" }

    12/21/2021, 5:36:21 PM node: e82026916c7f12b5 OrderID: msg.payload : Object
    > { ProcessType: "STPLC_11", xCarrierAvailable: "false", TimeStamp: 1639772701392, OrderID: "6073" }

    12/21/2021, 5:36:27 PM node: e82026916c7f12b5 OrderID: msg.payload : Object
    > { ProcessType: "STPLC_11", xCarrierAvailable: "false", TimeStamp: 1639772707693, OrderID: "6074" }

    12/21/2021, 5:36:30 PM node: e82026916c7f12b5 OrderID: msg.payload : Object
    > { ProcessType: "STPLC_11", xCarrierAvailable: "false", TimeStamp: 1639772710481, OrderID: "6074" }

```

Figur 36 Debug Output fraanden funktionel afprøvning af udvidede OPC UA Client-noder.

Search OrderID
6072

	ProcessID	Process Type	OrderID	Start Time	Finish Time
1		STPLC_10	6072	1639772636861	
4		STPLC_12	6072	1639772653779	
9		STPLC_13	6072	1639772688375	
12		STPLC_11	6072	1639772699978	

Search OrderID
6073

	ProcessID	Process Type	OrderID	Start Time	Finish Time
2		STPLC_10	6073	1639772644769	
6		STPLC_12	6073	1639772661646	
10		STPLC_13	6073	1639772695282	
14		STPLC_11	6073	1639772706702	

Process Overview

Search OrderID
6074

	ProcessID	Process Type	OrderID	Start Time	Finish Time
3		STPLC_10	6074	1639772652704	1639772652924
5		STPLC_10	6074	1639772654509	
7		STPLC_12	6074	1639772664365	1639772664612
8		STPLC_12	6074	1639772666135	
11		STPLC_13	6074	1639772699584	1639772699794
13		STPLC_13	6074	1639772704173	
15		STPLC_11	6074	1639772710173	1639772710481
16		STPLC_11	6074	1639772715130	

Figur 37 UI Output fra anden funktionel afprøvning af udvidede OPC UA Client-noder med brug af ordresøgningsfunktion.

Orders						
<input type="text" value="Search OrderID"/> ⟳						
	OrderID	Order Received	Throughput Time			
Process Overview						
<input type="text" value="Search OrderID"/> ⟳						
ProcessID	Process Type	OrderID	Start Time	Finish Time		
16	STPLC_11	6074	1639772715130			
15	STPLC_11	6074	1639772710173	1639772710481		
14	STPLC_11	6073	1639772706702			
13	STPLC_13	6074	1639772704173			
12	STPLC_11	6072	1639772699978			
11	STPLC_13	6074	1639772699584	1639772699794		
10	STPLC_13	6073	1639772695282			
9	STPLC_13	6072	1639772688375			
8	STPLC_12	6074	1639772666135			
7	STPLC_12	6074	1639772664365	1639772664612		
6	STPLC_12	6073	1639772661646			
5	STPLC_10	6074	1639772654509			
4	STPLC_12	6072	1639772653779			
3	STPLC_10	6074	1639772652704	1639772652924		
2	STPLC_10	6073	1639772644769			
1	STPLC_10	6072	1639772636861			

Figur 38 UI Output fra anden funktionel afprøvning af udvidede OPC UA Client-noder

7.2.4 Tredje funktionel afprøvning på udvidede OPC UA-Clients

The screenshot displays five separate debug output windows, each showing a series of log entries from an OPC UA client. The logs are timestamped and detail various database updates (UPDATE processes SET ...). Each entry includes a timestamp, node identifier, and a complex JSON payload representing a process update. The payloads consistently mention ProcessType values like 'STPLC_10' through 'STPLC_13', xCarrierAvailable status, and OrderID 6075.

```

out"
12/21/2021, 5:45:03 PM node: 195aaa2a61d80fe2
UPDATE processes SET FinishTime = 1639773224165
WHERE ProcessType = 'STPLC_10' AND OrderID =
6075; :msg.payload : Object
  > { ProcessType: "STPLC_10",
    xCarrierAvailable: "false",
   TimeStamp: 1639773224165, OrderID:
    "6075" }

12/21/2021, 5:45:05 PM node: 195aaa2a61d80fe2
UPDATE processes SET FinishTime = 1639773225944
WHERE ProcessType = 'STPLC_10' AND OrderID =
6075; :msg.payload : Object
  > { ProcessType: "STPLC_10",
    xCarrierAvailable: "false",
   TimeStamp: 1639773225944, OrderID:
    "6075" }

12/21/2021, 5:45:15 PM node: 446b88a38e3477fb
UPDATE processes SET FinishTime = 1639773236062
WHERE ProcessType = 'STPLC_12' AND OrderID =
6075; :msg.payload : Object
  > { ProcessType: "STPLC_12",
    xCarrierAvailable: "false",
   TimeStamp: 1639773236062, OrderID:
    "6075" }

12/21/2021, 5:45:17 PM node: 446b88a38e3477fb
UPDATE processes SET FinishTime = 1639773237711
WHERE ProcessType = 'STPLC_12' AND OrderID =
6075; :msg.payload : Object
  > { ProcessType: "STPLC_12",
    xCarrierAvailable: "false",
   TimeStamp: 1639773237711, OrderID:
    "6075" }

12/21/2021, 5:45:49 PM node: be21e4a2e6472446
UPDATE processes SET FinishTime = 1639773269465
WHERE ProcessType = 'STPLC_13' AND OrderID =
6075; :msg.payload : Object
  > { ProcessType: "STPLC_13",
    xCarrierAvailable: "false",
   TimeStamp: 1639773269465, OrderID:
    "6075" }

12/21/2021, 5:46:48 PM node: 446b88a38e34777b
UPDATE processes SET FinishTime = 1639773273142
WHERE ProcessType = 'STPLC_13' AND OrderID =
6075; :msg.payload : Object
  > { ProcessType: "STPLC_13",
    xCarrierAvailable: "false",
   TimeStamp: 1639773273142, OrderID:
    "6075" }

12/21/2021, 5:46:53 PM node: 446b88a38e34777b
UPDATE processes SET FinishTime = 1639773333492
WHERE ProcessType = 'STPLC_12' AND OrderID =
6075; :msg.payload : Object
  > { ProcessType: "STPLC_12",
    xCarrierAvailable: "false",
   TimeStamp: 1639773333492, OrderID:
    "6075" }

12/21/2021, 5:46:53 PM node: be21e4a2e6472446
UPDATE processes SET FinishTime = 1639773364340
WHERE ProcessType = 'STPLC_13' AND OrderID =
6075; :msg.payload : Object
  > { ProcessType: "STPLC_13",
    xCarrierAvailable: "false",
   TimeStamp: 1639773364340, OrderID:
    "6075" }

12/21/2021, 5:47:24 PM node: be21e4a2e6472446
UPDATE processes SET FinishTime = 1639773364340
WHERE ProcessType = 'STPLC_13' AND OrderID =
6075; :msg.payload : Object
  > { ProcessType: "STPLC_13",
    xCarrierAvailable: "false",
   TimeStamp: 1639773364340, OrderID:
    "6075" }

12/21/2021, 5:47:27 PM node: be21e4a2e6472446
UPDATE processes SET FinishTime = 1639773367722
WHERE ProcessType = 'STPLC_13' AND OrderID =
6075; :msg.payload : Object
  > { ProcessType: "STPLC_13",
    xCarrierAvailable: "false",
   TimeStamp: 1639773367722, OrderID:
    "6075" }

12/21/2021, 5:47:35 PM node: e82026916c7f12b5
UPDATE processes SET FinishTime = 1639773375830
WHERE ProcessType = 'STPLC_11' AND OrderID =
6075; :msg.payload : Object
  > { ProcessType: "STPLC_11",
    xCarrierAvailable: "false",
   TimeStamp: 1639773375830, OrderID:
    "6075" }

12/21/2021, 5:47:38 PM node: e82026916c7f12b5
UPDATE processes SET FinishTime = 1639773378354
WHERE ProcessType = 'STPLC_11' AND OrderID =
6075; :msg.payload : Object
  > { ProcessType: "STPLC_11",
    xCarrierAvailable: "false",
   TimeStamp: 1639773378354, OrderID:
    "6075" }

```

Figur 39 Debug Output fra tredje funktionel afprøvning af udvidede OPC UA Client-noder.

Orders					
Process Overview					
OrderID		Order Received		Throughput Time	
Search OrderID					
ProcessID	Process Type	OrderID	Start Time	Finish Time	
16	STPLC_11	6075	1639773385049		
15	STPLC_11	6075	1639773378140	1639773378354	
14	STPLC_13	6075	1639773374043		
13	STPLC_13	6075	1639773367473	1639773367722	
12	STPLC_12	6075	1639773339418		
11	STPLC_12	6075	1639773333344	1639773333492	
10	STPLC_10	6075	1639773328477		
9	STPLC_10	6075	1639773322672	1639773322830	
8	STPLC_11	6075	1639773288391	1639773378354	
7	STPLC_11	6075	1639773283490	1639773378354	
6	STPLC_13	6075	1639773277374	1639773367722	
5	STPLC_13	6075	1639773272876	1639773367722	
4	STPLC_12	6075	1639773242422	1639773333492	
3	STPLC_12	6075	1639773237497	1639773333492	
2	STPLC_10	6075	1639773231362	1639773322830	
1	STPLC_10	6075	1639773225744	1639773322830	

Figur 40 UI Output fra tredje funktionel afprøvning af udvidede OPC UA Client-noder.

7.2.5 Funktionelafprøvning af 50ms forsinkelse på OrderID Read-kommando

The figure consists of two parts: a debug output window at the top and a UI output window below it.

Debug Output (Top):

```

debug
all nodes
all

1/4/2022, 1:05:11 PM node: 195aaa2a61d80fe2
OrderID : msg.payload : Object
  ↪ { ProcessType: "STPLC_10",
    xCarrierAvailable: "false",
    TimeStamp: 1639773143143, OrderID:
    "6079" }

1/4/2022, 1:05:13 PM node: 195aaa2a61d80fe2
OrderID : msg.payload : Object
  ↪ { ProcessType: "STPLC_10",
    xCarrierAvailable: "true", TimeStamp:
    1639773144725, OrderID: "6079" }

1/4/2022, 1:05:23 PM node: 446b88a38e34777b
OrderID : msg.payload : Object
  ↪ { ProcessType: "STPLC_12",
    xCarrierAvailable: "false",
    TimeStamp: 1639773154851, OrderID:
    "6079" }

1/4/2022, 1:05:25 PM node: 446b88a38e34777b
OrderID : msg.payload : Object
  ↪ { ProcessType: "STPLC_12",
    xCarrierAvailable: "true", TimeStamp:
    1639773156428, OrderID: "6079" }

1/4/2022, 1:05:57 PM node: be21e4a2e6472446
OrderID : msg.payload : Object
  ↪ { ProcessType: "STPLC_13",
    xCarrierAvailable: "false",
    TimeStamp: 1639773189043, OrderID:
    "6079" }

1/4/2022, 1:06:01 PM node: be21e4a2e6472446
OrderID : msg.payload : Object
  ↪ { ProcessType: "STPLC_13",
    xCarrierAvailable: "true", TimeStamp:
    1639773192651, OrderID: "6079" }

1/4/2022, 1:06:09 PM node: e82026916c7f12b5
OrderID : msg.payload : Object
  ↪ { ProcessType: "STPLC_11",
    xCarrierAvailable: "false",
    TimeStamp: 1639773200722, OrderID:
    "6079" }

1/4/2022, 1:06:12 PM node: e82026916c7f12b5
OrderID : msg.payload : Object
  ↪ { ProcessType: "STPLC_11",
    xCarrierAvailable: "true", TimeStamp:
    1639773203746, OrderID: "6079" }

```

UI Output (Bottom):

Orders

Orders		
OrderID	Order Received	Throughput Time

Process Overview

Process Overview					
Search OrderID					
ProcessID	Process Type	OrderID	Start Time	Finish Time	
4	STPLC_11	6079	1639773203746		
3	STPLC_13	6079	1639773192651		
2	STPLC_12	6079	1639773156428		
1	STPLC_10	6079	1639773144725		

Figur 41 (ovenfor) - a) Debug Output og (nedenfor) - b) UI Output for funktionelafprøvning af 50ms forsinkelse på OrderID Read-kommando.

7.2.6 Funktionelafprøvning af 75ms forsinkelse på OrderID Read-kommando

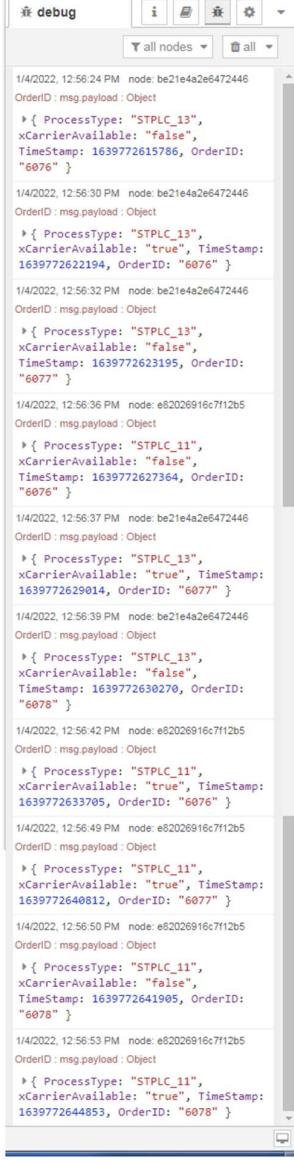
The figure consists of two main parts. The top part is a screenshot of a software interface showing a log of events. It has two panes. The left pane shows log entries from 1/4/2022, 1:26:00 PM to 1:26:06 PM, and the right pane shows entries from 1/4/2022, 1:26:06 PM to 1:28:14 PM. Both panes include a header for 'debug' and a toolbar with icons for 'Deploy', 'Stop', 'Run', etc. A red vertical bar highlights the second column of the log entries in both panes. The bottom part is a screenshot of a web application titled 'Orders'. It features a search bar for 'Search OrderID' and a table with columns: 'OrderID', 'Order Received', and 'Throughput Time'. Below this is a section titled 'Process Overview' with a search bar for 'Search OrderID' and a table with columns: 'ProcessID', 'Process Type', 'OrderID', 'Start Time', and 'Finish Time'. The data in the tables corresponds to the log entries above.

ProcessID	Process Type	OrderID	Start Time	Finish Time
4	STPLC_11	6083	1639774525856	1639774526163
3	STPLC_13	6083	1639774514893	
2	STPLC_12	6083	1639774479974	
1	STPLC_10	6083	1639774468885	

Figur 42 (ovenfor) - a) Debug Output og (nedenfor) -b) UI Output for funktionelafprøvning af 75ms forsinkelse på OrderID Read-kommando. (Det dækkede område er repeterede data fra 2. kolonne)

7.2.7 Funktionelafprøvning af 100ms forsinkelse på OrderID Read-kommando

a) Debug Output:



```

1/4/2022, 12:56:24 PM node: be21e4a2e6472446
OrderID : msg.payload : Object
  > { ProcessType: "STPLC_13",
    xCarrierAvailable: "false",
    TimeStamp: 1639772615786, OrderID:
    "6076" }

1/4/2022, 12:56:30 PM node: be21e4a2e6472446
OrderID : msg.payload : Object
  > { ProcessType: "STPLC_13",
    xCarrierAvailable: "true", TimeStamp:
    1639772622194, OrderID: "6076" }

1/4/2022, 12:56:32 PM node: be21e4a2e6472446
OrderID : msg.payload : Object
  > { ProcessType: "STPLC_13",
    xCarrierAvailable: "false",
    TimeStamp: 1639772623195, OrderID:
    "6077" }

1/4/2022, 12:56:36 PM node: e82026916c7f12b5
OrderID : msg.payload : Object
  > { ProcessType: "STPLC_11",
    xCarrierAvailable: "false",
    TimeStamp: 1639772627364, OrderID:
    "6076" }

1/4/2022, 12:56:37 PM node: be21e4a2e6472446
OrderID : msg.payload : Object
  > { ProcessType: "STPLC_13",
    xCarrierAvailable: "true", TimeStamp:
    1639772629014, OrderID: "6077" }

1/4/2022, 12:56:39 PM node: be21e4a2e6472446
OrderID : msg.payload : Object
  > { ProcessType: "STPLC_13",
    xCarrierAvailable: "false",
    TimeStamp: 1639772630270, OrderID:
    "6078" }

1/4/2022, 12:56:42 PM node: e82026916c7f12b5
OrderID : msg.payload : Object
  > { ProcessType: "STPLC_11",
    xCarrierAvailable: "true", TimeStamp:
    1639772633705, OrderID: "6076" }

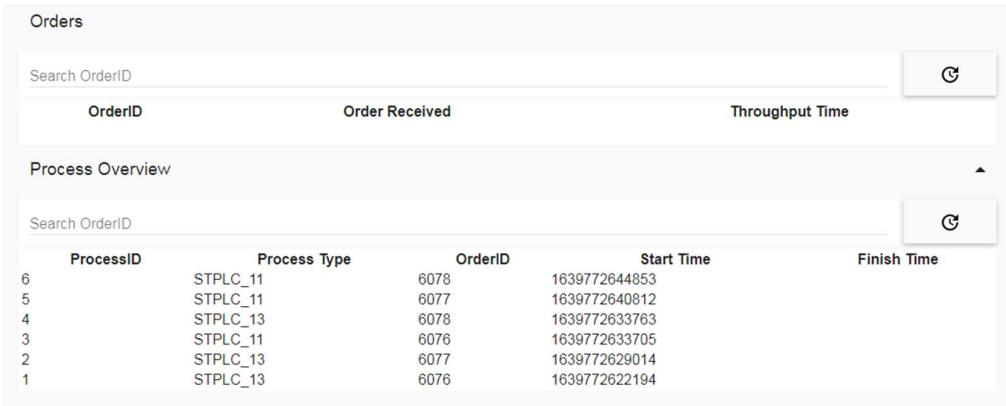
1/4/2022, 12:56:49 PM node: e82026916c7f12b5
OrderID : msg.payload : Object
  > { ProcessType: "STPLC_11",
    xCarrierAvailable: "true", TimeStamp:
    1639772640812, OrderID: "6077" }

1/4/2022, 12:56:50 PM node: e82026916c7f12b5
OrderID : msg.payload : Object
  > { ProcessType: "STPLC_11",
    xCarrierAvailable: "false",
    TimeStamp: 1639772641905, OrderID:
    "6078" }

1/4/2022, 12:56:53 PM node: e82026916c7f12b5
OrderID : msg.payload : Object
  > { ProcessType: "STPLC_11",
    xCarrierAvailable: "true", TimeStamp:
    1639772644853, OrderID: "6079" }

```

b) UI Output:



Orders			
Search OrderID			
OrderID	Order Received	Throughput Time	
			(refresh)

Process Overview					
Search OrderID					
ProcessID	Process Type	OrderID	Start Time	Finish Time	
6	STPLC_11	6078	1639772644853		(refresh)
5	STPLC_11	6077	1639772640812		(refresh)
4	STPLC_13	6078	1639772633763		(refresh)
3	STPLC_11	6076	1639772633705		(refresh)
2	STPLC_13	6077	1639772629014		(refresh)
1	STPLC_13	6076	1639772622194		(refresh)

Figur 43 (ovenfor) - a) Debug Output og (nedenfor) - b) UI Output for funktionelafprøvning af 100ms forsinkelse på OrderID Read-kommando.

7.2.8 Enkeltpalle funktionelafprøvning af 500ms forsinkelse på OrderID Read-kommando

a) Debug Output

```

1/4/2022, 1:50:05 PM node: 4b41d67b64e1bcd2
OrderID : msg.payload : Object
  > { ProcessType: "STPLC_10",
    xCarrierAvailable: "true",
    TimeStamp: 1639775836287,
    OrderID: "6085" }

1/4/2022, 1:50:11 PM node: 4b41d67b64e1bcd2
OrderID : msg.payload : Object
  > { ProcessType: "STPLC_10",
    xCarrierAvailable: "false",
    TimeStamp: 1639775842553,
    OrderID: "6085" }

1/4/2022, 1:50:22 PM node: 4b41d67b64e1bcd2
OrderID : msg.payload : Object
  > { ProcessType: "STPLC_12",
    xCarrierAvailable: "true",
    TimeStamp: 1639775852968,
    OrderID: "6085" }

1/4/2022, 1:50:23 PM node: 4b41d67b64e1bcd2
OrderID : msg.payload : Object
  > { ProcessType: "STPLC_12",
    xCarrierAvailable: "false",
    TimeStamp: 1639775853923,
    OrderID: "6085" }

1/4/2022, 1:50:57 PM node: 4b41d67b64e1bcd2
OrderID : msg.payload : Object
  > { ProcessType: "STPLC_13",
    xCarrierAvailable: "true",
    TimeStamp: 1639775888113,
    OrderID: "6085" }

1/4/2022, 1:50:58 PM node: 4b41d67b64e1bcd2
OrderID : msg.payload : Object
  > { ProcessType: "STPLC_13",
    xCarrierAvailable: "false",
    TimeStamp: 1639775889107,
    OrderID: "6085" }

1/4/2022, 1:51:08 PM node: 4b41d67b64e1bcd2
OrderID : msg.payload : Object
  > { ProcessType: "STPLC_11",
    xCarrierAvailable: "true",
    TimeStamp: 1639775899423,
    OrderID: "6085" }

1/4/2022, 1:51:09 PM node: 4b41d67b64e1bcd2
OrderID : msg.payload : Object
  > { ProcessType: "STPLC_11",
    xCarrierAvailable: "false",
    TimeStamp: 1639775900633,
    OrderID: "6085" }

```

b) UI Output

Orders					
Search OrderID					
OrderID	Order Received		Throughput Time		
6085					

Process Overview					
Search OrderID					
ProcessID	Process Type	OrderID	Start Time	Finish Time	
4	STPLC_11	6085	1639775899423	1639775900633	
3	STPLC_13	6085	1639775888113	1639775889107	
2	STPLC_12	6085	1639775852968	1639775853923	
1	STPLC_10	6085	1639775836287	1639775842553	

Figur 44 (ovenfor) - a) Debug Output og (nedenfor) - b) UI Output for funktionelafprøvning af 500ms forsinkelse på OrderID Read-kommando med én palle.

7.2.9 Multipalle funktionelafprøvning af 500ms forsinkelse på OrderID Read-kommando

Test 1

The figure shows five parallel debug output windows, each representing a different execution path or instance of the function being tested. The logs capture the flow of data and processing steps for multiple pallets (OrderIDs) over time.

- Window 1:** Logs for OrderID 6086. It shows several entries for ProcessType: "STPLC_10" and ProcessType: "STPLC_12".
- Window 2:** Logs for OrderID 6087. It shows entries for ProcessType: "STPLC_12" and ProcessType: "STPLC_13".
- Window 3:** Logs for OrderID 6088. It shows entries for ProcessType: "STPLC_12", "STPLC_13", and "STPLC_11".
- Window 4:** Logs for OrderID 6086. It shows entries for ProcessType: "STPLC_11" and "STPLC_13".
- Window 5:** Logs for OrderID 6089. It shows entries for ProcessType: "STPLC_08", "STPLC_11", and "STPLC_13".

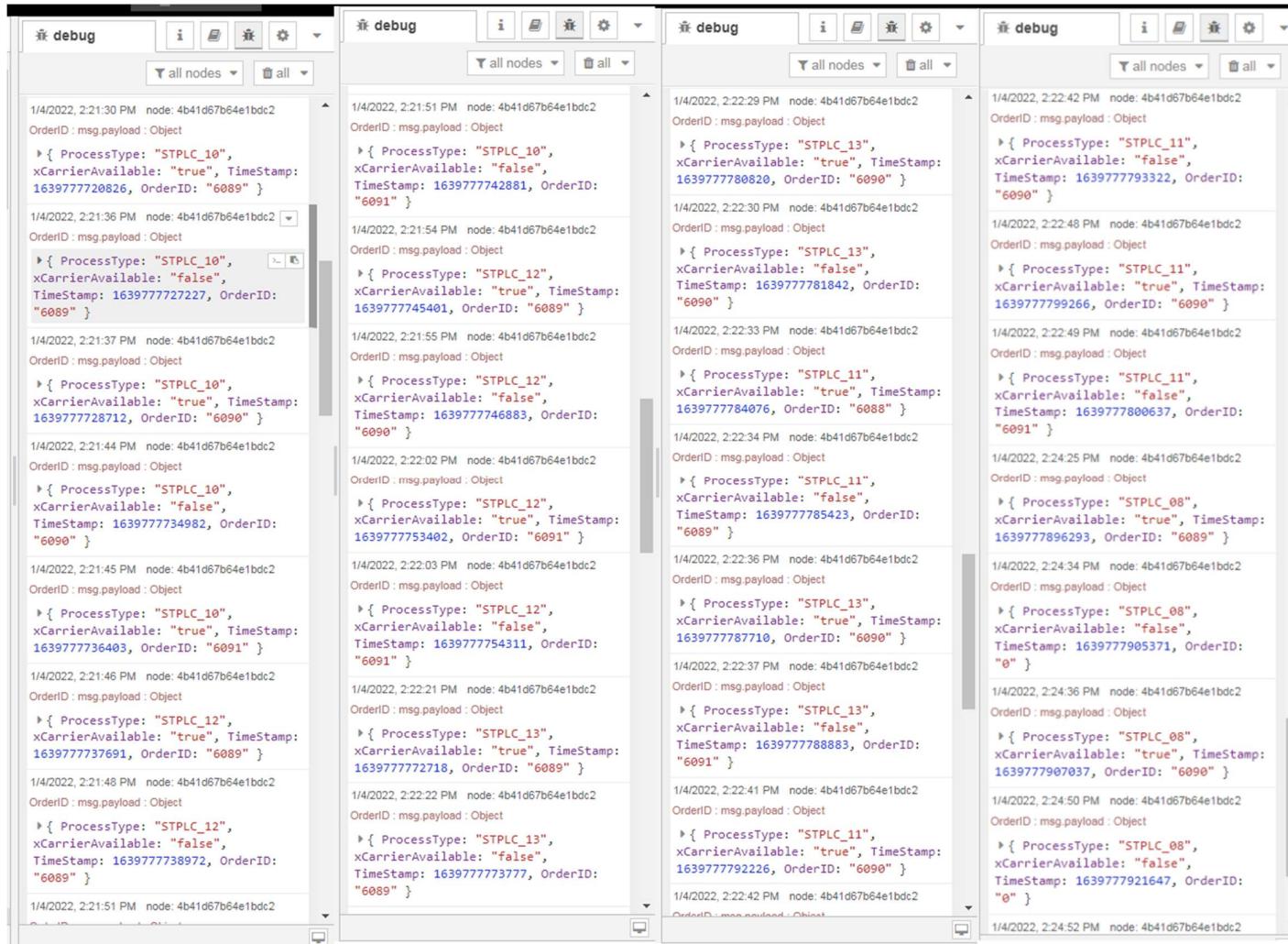
In all windows, the logs include timestamp, node ID (4b41d67b64e1bcd2), OrderID, msg.payload (Object), and detailed process information (ProcessType, xCarrierAvailable, TimeStamp). The logs illustrate the handling of multiple pallets (OrderIDs) and the execution of different processing logic (STPLC_10, STPLC_12, STPLC_13, STPLC_11, STPLC_08) over a period of approximately 2 hours and 45 minutes.

Figur 45 Debug Output for første funktionelafprøvning af 500ms forsinkelse på OrderID Read-kommando med flere paller.

Orders				
Process Overview				
Search OrderID				
OrderID	Order Received	Throughput Time		
14	STPLC_08	6088	1639776702007	
13	STPLC_11	6088	1639776561540	1639776562848
12	STPLC_08	6086	1639776554114	
11	STPLC_13	6088	1639776550201	1639776551135
10	STPLC_11	6087	1639776526830	1639776527879
9	STPLC_11	6085	1639776519019	
8	STPLC_13	6087	1639776515497	1639776516434
7	STPLC_12	6088	1639776515278	1639776516250
6	STPLC_13	6086	1639776507613	1639776508702
5	STPLC_12	6087	1639776480423	1639776481614
4	STPLC_12	6086	1639776472647	1639776473616
3	STPLC_10	6088	1639776471473	1639776504833
2	STPLC_10	6087	1639776463754	1639776469961
1	STPLC_10	6086	1639776455886	1639776462155

Figur 46 UI Output for første funktionelafprøvning af 500ms forsinkelse på OrderID Read-kommando med flere paller.

Test 2



Figur 47 Debug Output for anden funktionelafprøvning af 500ms forsinkelse på OrderID Read-kommando med flere paller.

Orders				
Process Overview				
OrderID	Order Received	Throughput Time		
Search OrderID				
ProcessID	Process Type	OrderID	Start Time	Finish Time
15	STPLC_08	6091	1639777923242	
14	STPLC_08	6090	1639777907037	
13	STPLC_08	6089	1639777896293	
12	STPLC_11	6090	1639777799266	
11	STPLC_11	6090	1639777792226	1639777793322
10	STPLC_13	6090	1639777787710	
9	STPLC_11	6088	1639777784076	
8	STPLC_13	6090	1639777780820	1639777781842
7	STPLC_13	6089	1639777772718	1639777773777
6	STPLC_12	6091	1639777753402	1639777754311
5	STPLC_12	6089	1639777745401	
4	STPLC_12	6089	1639777737691	1639777738972
3	STPLC_10	6091	1639777736403	1639777742881
2	STPLC_10	6090	1639777728712	1639777734982
1	STPLC_10	6089	1639777720826	1639777727227

Figur 48 UI Output for anden funktionelafprøvning af 500ms forsinkelse på OrderID Read-kommando med flere paller.

7.2.10 Multipalle funktionelafprøvning af 550ms forsinkelse på OrderID Read-kommando

```

1/4/2022, 2:39:36 PM node: 4b41d67b64e1bcd2
OrderID : msg.payload : Object
  ↪ { ProcessType: "STPLC_10",
    xCarrierAvailable: "true",
    TimeStamp: 1639778807223,
    OrderID: "6095" }

1/4/2022, 2:39:42 PM node: 4b41d67b64e1bcd2
OrderID : msg.payload : Object
  ↪ { ProcessType: "STPLC_10",
    xCarrierAvailable: "false",
    TimeStamp: 1639778813432,
    OrderID: "6095" }

1/4/2022, 2:39:43 PM node: 4b41d67b64e1bcd2
OrderID : msg.payload : Object
  ↪ { ProcessType: "STPLC_10",
    xCarrierAvailable: "true",
    TimeStamp: 1639778814860,
    OrderID: "6096" }

1/4/2022, 2:39:50 PM node: 4b41d67b64e1bcd2
OrderID : msg.payload : Object
  ↪ { ProcessType: "STPLC_10",
    xCarrierAvailable: "false",
    TimeStamp: 1639778821062,
    OrderID: "6096" }

1/4/2022, 2:39:51 PM node: 4b41d67b64e1bcd2
OrderID : msg.payload : Object
  ↪ { ProcessType: "STPLC_10",
    xCarrierAvailable: "true",
    TimeStamp: 1639778822575,
    OrderID: "6097" }

1/4/2022, 2:39:52 PM node: 4b41d67b64e1bcd2
OrderID : msg.payload : Object
  ↪ { ProcessType: "STPLC_12",
    xCarrierAvailable: "true",
    TimeStamp: 1639778823766,
    OrderID: "6095" }

1/4/2022, 2:39:53 PM node: 4b41d67b64e1bcd2
OrderID : msg.payload : Object
  ↪ { ProcessType: "STPLC_12",
    xCarrierAvailable: "false",
    TimeStamp: 1639778824814,
    OrderID: "6095" }

1/4/2022, 2:39:58 PM node: 4b41d67b64e1bcd2
OrderID : msg.payload : Object
  ↪ { ProcessType: "STPLC_12",
    xCarrierAvailable: "true",
    TimeStamp: 16397788282930,
    OrderID: "6097" }

1/4/2022, 2:39:58 PM node: 4b41d67b64e1bcd2
OrderID : msg.payload : Object
  ↪ { ProcessType: "STPLC_13",
    xCarrierAvailable: "false",
    TimeStamp: 1639778831413,
    OrderID: "6096" }

1/4/2022, 2:40:01 PM node: 4b41d67b64e1bcd2
OrderID : msg.payload : Object
  ↪ { ProcessType: "STPLC_12",
    xCarrierAvailable: "false",
    TimeStamp: 1639778832460,
    OrderID: "6096" }

1/4/2022, 2:40:08 PM node: 4b41d67b64e1bcd2
OrderID : msg.payload : Object
  ↪ { ProcessType: "STPLC_11",
    xCarrierAvailable: "true",
    TimeStamp: 1639778870151,
    OrderID: "6095" }

1/4/2022, 2:40:14 PM node: 4b41d67b64e1bcd2
OrderID : msg.payload : Object
  ↪ { ProcessType: "STPLC_11",
    xCarrierAvailable: "false",
    TimeStamp: 1639778871196,
    OrderID: "6095" }

1/4/2022, 2:40:27 PM node: 4b41d67b64e1bcd2
OrderID : msg.payload : Object
  ↪ { ProcessType: "STPLC_13",
    xCarrierAvailable: "true",
    TimeStamp: 1639778873858,
    OrderID: "6097" }

1/4/2022, 2:40:42 PM node: 4b41d67b64e1bcd2
OrderID : msg.payload : Object
  ↪ { ProcessType: "STPLC_13",
    xCarrierAvailable: "false",
    TimeStamp: 1639778875118,
    OrderID: "6097" }

1/4/2022, 2:40:47 PM node: 4b41d67b64e1bcd2
OrderID : msg.payload : Object
  ↪ { ProcessType: "STPLC_11",
    xCarrierAvailable: "true",
    TimeStamp: 1639778877942,
    OrderID: "6096" }

1/4/2022, 2:40:48 PM node: 4b41d67b64e1bcd2
OrderID : msg.payload : Object
  ↪ { ProcessType: "STPLC_11",
    xCarrierAvailable: "false",
    TimeStamp: 16397788867421,
    OrderID: "6096" }

1/4/2022, 2:40:48 PM node: 4b41d67b64e1bcd2
OrderID : msg.payload : Object
  ↪ { ProcessType: "STPLC_11",
    xCarrierAvailable: "true",
    TimeStamp: 1639778885444,
    OrderID: "6097" }

1/4/2022, 2:40:55 PM node: 4b41d67b64e1bcd2
OrderID : msg.payload : Object
  ↪ { ProcessType: "STPLC_11",
    xCarrierAvailable: "false",
    TimeStamp: 1639778886608,
    OrderID: "6097" }

1/4/2022, 2:41:14 PM node: 4b41d67b64e1bcd2
OrderID : msg.payload : Object
  ↪ { ProcessType: "STPLC_08",
    xCarrierAvailable: "true",
    TimeStamp: 1639778905604,
    OrderID: "0" }

1/4/2022, 2:52:41 PM node: 4b41d67b64e1bcd2
OrderID : msg.payload : Object
  ↪ { ProcessType: "STPLC_08",
    xCarrierAvailable: "false",
    TimeStamp: 1639779592315,
    OrderID: "0" }

1/4/2022, 2:52:46 PM node: 4b41d67b64e1bcd2
OrderID : msg.payload : Object
  ↪ { ProcessType: "STPLC_08",
    xCarrierAvailable: "true",
    TimeStamp: 1639779597812,
    OrderID: "0" }

1/4/2022, 2:52:54 PM node: 4b41d67b64e1bcd2
OrderID : msg.payload : Object
  ↪ { ProcessType: "STPLC_08",
    xCarrierAvailable: "false",
    TimeStamp: 1639779605233,
    OrderID: "0" }

1/4/2022, 2:53:04 PM node: 4b41d67b64e1bcd2
OrderID : msg.payload : Object
  ↪ { ProcessType: "STPLC_08",
    xCarrierAvailable: "true",
    TimeStamp: 1639779616198,
    OrderID: "0" }

```

Figur 49 Debug Output for funktionelafprøvning af 550ms forsinkelse på OrderID Read-kommando med flere paller. (Det dækkede område er repeterede data fra 4. kolonne)

Orders

Order Received					Throughput Time
OrderID					
11	STPLC_11	6097	1639778885444		1639778886608
10	STPLC_11	6096	1639778877942		1639778879258
9	STPLC_13	6097	1639778873858		1639778875118
8	STPLC_11	6095	1639778870151		1639778871196
7	STPLC_13	6096	1639778866260		1639778867421
6	STPLC_12	6097	1639778839366		1639778840352
5	STPLC_12	6096	1639778831413		1639778832460
4	STPLC_12	6095	1639778823766		1639778824814
3	STPLC_10	6097	1639778822575		1639778828930
2	STPLC_10	6096	1639778814860		1639778821062
1	STPLC_10	6095	1639778807223		1639778813432

Process Overview

ProcessID	Process Type	OrderID	Start Time	Finish Time
11	STPLC_11	6097	1639778885444	1639778886608
10	STPLC_11	6096	1639778877942	1639778879258
9	STPLC_13	6097	1639778873858	1639778875118
8	STPLC_11	6095	1639778870151	1639778871196
7	STPLC_13	6096	1639778866260	1639778867421
6	STPLC_12	6097	1639778839366	1639778840352
5	STPLC_12	6096	1639778831413	1639778832460
4	STPLC_12	6095	1639778823766	1639778824814
3	STPLC_10	6097	1639778822575	1639778828930
2	STPLC_10	6096	1639778814860	1639778821062
1	STPLC_10	6095	1639778807223	1639778813432

Figur 50 UI Output for funktionelafprøvning af 550ms forsinkelse på OrderID Read-kommando med flere paller.