# Beer production

## Remote control

Date: September 3, 2018

# I Versions

| Version | Date | Comment | Edited by |
|---|---|---|---|
| 1.0 | Aug 31, 2018 | First Edition | olsenm |
| | | | |
| | | | |
| | | | |

**Table 1: Versions**

# II Table of Contents

# 1 Introduction

## 1.1 Purpose

The purpose of this document is to give the developers of the new IT system the basic overview of the scope of the project. The document contains the requirements for the IT system and also a detailed interface description of the existing brewing machine.

## 1.2 Target group

The developers of the IT system.

# 2 Brewing production

## 2.1 Vision

The brewery Refslevbæk Bryghus A/S started as a microbrewery a few years ago. Until now the production of their beer has been made in small batches with a minimalistic production machinery where a lot of manual processes was involved. Last year they won the best beer of the year award. Since then demands for their products has increased and they can no long fulfill the demands with the existing production capacity.
Therefore a new brewing machine was bought and will be integrated into the production line. In this document it will referred as “the machine”.

One of the requirements for the new machine was that it should be able to communicate with other systems in order to receive and send production data. By ensuring this the brewery is able to automate the production and keep the cost down.

## 2.2 The brewing machine

The brewing machine is able to produce different kinds of beer. The machine can produce a batch with a defined quantity of beer and a specified recipe for the batch. The machine can report that a batch has been completed and output various data that has been produced during the production of the batch.

### 2.2.1 Recipes

The machine can produce different kind of beer types. In the table below the different kind of beer the machine is able to produce is listed. The product can be selected by using the machines remote interface. The recipe or product type can be set in the PackTag Command.Parameter[1].Value. The current product can be read from the PackTag Admin.Parameter[0].Value.

| Product | Id |
|---|---|
| Pilsner | 0 |
| Wheat | 1 |
| IPA | 2 |
| Stout | 3 |
| Ale | 4 |
| Alcohol Free | 5 |

**Figure 1: Beer recipes / types**

### 2.2.2 Batch id

Beers are produced on the machine in batches of a defined quantity of bottles. Each batch can be equipped with a batch id so it is possible to track the whole batch afterwards with an id. Batch id's are not generated by the machine but must some from an external system. The batch id can be set in the PackTag Command.Parameter[0].Value. The current batch id can be read from the PackTag Status.Parameter[0].Value.

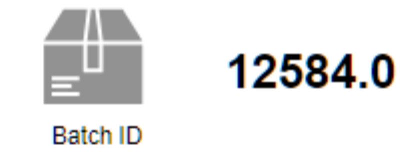


**Figure 2: Batch id to track the batch**

### 2.2.3 Quantity

To produce a batch of beers it must also be defined the amount of beers that must be produced in the batch. The quantity can be adjusted in the PackTag Command.Parameter[2].Value. The unit of the value is beers/minute. The current quantity can be read from the PackTag Status.Parameter[1].Value.


**Figure 3: Amount of beers in a batch**

### 2.2.4 Machine speed

The speed can be changed on the machine so it can be defined how many products the machine will produce. The unit of the speed is beers pr. minute (primary products/minute). The speed can be adjusted in the PackTag Command.MachSpeed. The set speed and the actual speed can also be read from the PackTags Status.MachSpeed and Status.CurMachSpeed.


**Figure 4: The speed of the machine can be adjusted**

There is a limit for the speed depending of the product.

| Product | Valid input range |
|---|---|
| Pilsner | $0 \leq input \leq 600$ |
| Wheat | $0 \leq input \leq 300$ |
| IPA | $0 \leq input \leq 150$ |
| Stout | $0 \leq input \leq 200$ |
| Ale | $0 \leq input \leq 100$ |
| Alcohol Free | $0 \leq input \leq 125$ |

**Figure 5: Valid speed ranges**

### 2.2.5 Ingredients

The machine uses various ingredients for the beer production like barley, hops, malt, wheat and yeast. Each recipe uses a different amount of ingredients.

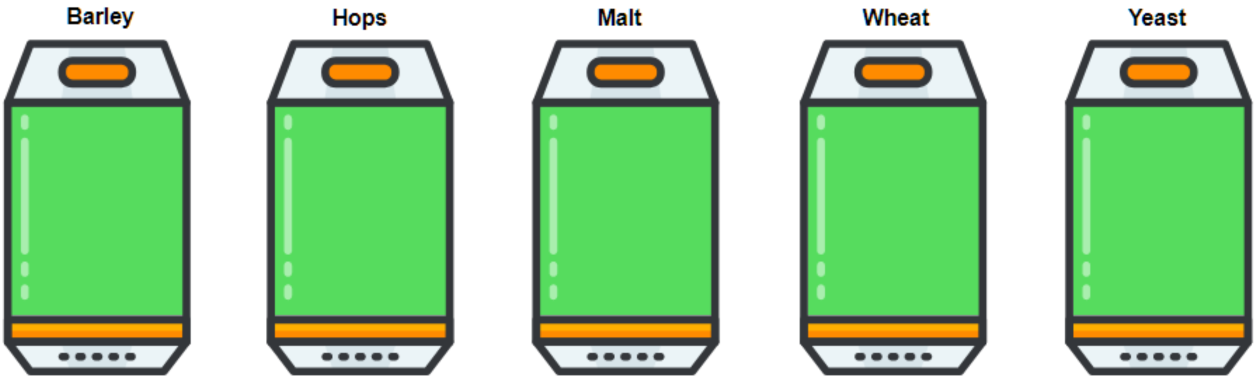


**Figure 6: Various ingredients**

As beers are being produced the machine will use the ingredients for production and the level of the ingredients tanks will drop. When a used ingredient is not available on the machine the production will stop. Then the ingredients tanks must be refilled manually on the machine.

### 2.2.6 Sensor data

The machine is equipped with various sensors that can measure data while the machine is producing. These data can be collected and used for later analysis of the products that has been produced. Data available are temperature, humidity and vibration.

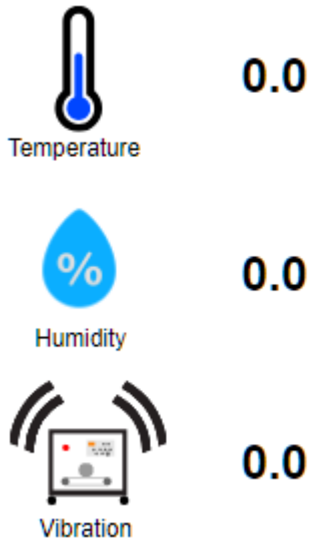


**Figure 7: Production data from the machine**

The machine cannot collect these production data. The data must be collected by an external system. The sensor data can be read from the [Status PackTag structure](#).

## 2.2.7 Production errors

The machine is equipped with a sophisticated quality inspection system. Each produced beer will be inspected in the system and some of the beers will fail for the quality inspection. When a batch is finished the machine will output how many products passed the inspection and how many failed. The products produced can be read from the PackTag [Admin.ProdProcessedCount](#). The failed products can be read from the PackTag [Admin.ProdDefectiveCount](#).

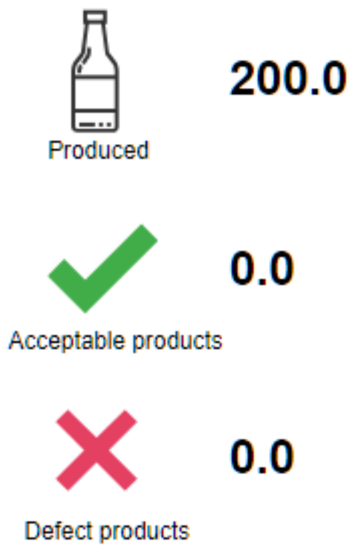


**Figure 8: Passed and failed products**

The fault rate of the production depends of the speed of the machine. More products will fail if the machine is runs fast.

### 2.2.8 Stop reasons

Multiple situations can occur that will cause the machine to stop. The machine will output the cause for the stop. It can be monitored in the PackTag Admin.StopReason.ID. The description of the stop id can be seen in the table below.

| StopReason.Id | Comment |
|---|---|
| 10 | Empty inventory |
| 11 | Maintenance |
| 12 | Manual stop |
| 13 | Motor power loss |
| 14 | Manual abort |

**Figure 9: Stop reasons**

Only the stop reasons with ID 10 and 11 will need special handling. The others can be handled by sending the relevant commands seen in the PackML state machine.

### 2.2.9 Maintenance

The machine cannot run forever without some maintenance. The machine is however able to output when service is needed. This can be seen on the maintenance bar.


**Figure 10: Maintenance bar**

When the machine needs maintenance it will stop and maintenance will have to be made manually on the machine. After maintenance the machine is ready to produce again.

## 2.3 Interface

To be able to integrate the machine in a smart production line the machine must have an interface to other systems. The interface can be used to remote control the machine or to collect production data from the machine.

### 2.3.1 OPC-UA

The machines uses OPC-UA[1][2] to communicate with other systems. OPC-UA was chosen because it is a free, open, cross-platform, vendor-independent protocol. Furthermore it has built-in security so the data transport is secured and the data can be configured to be shown for different user roles.

[1] https://en.wikipedia.org/wiki/OPC_Unified_Architecture
[2] https://www.youtube.com/watch?v=-tDGzwsBokY

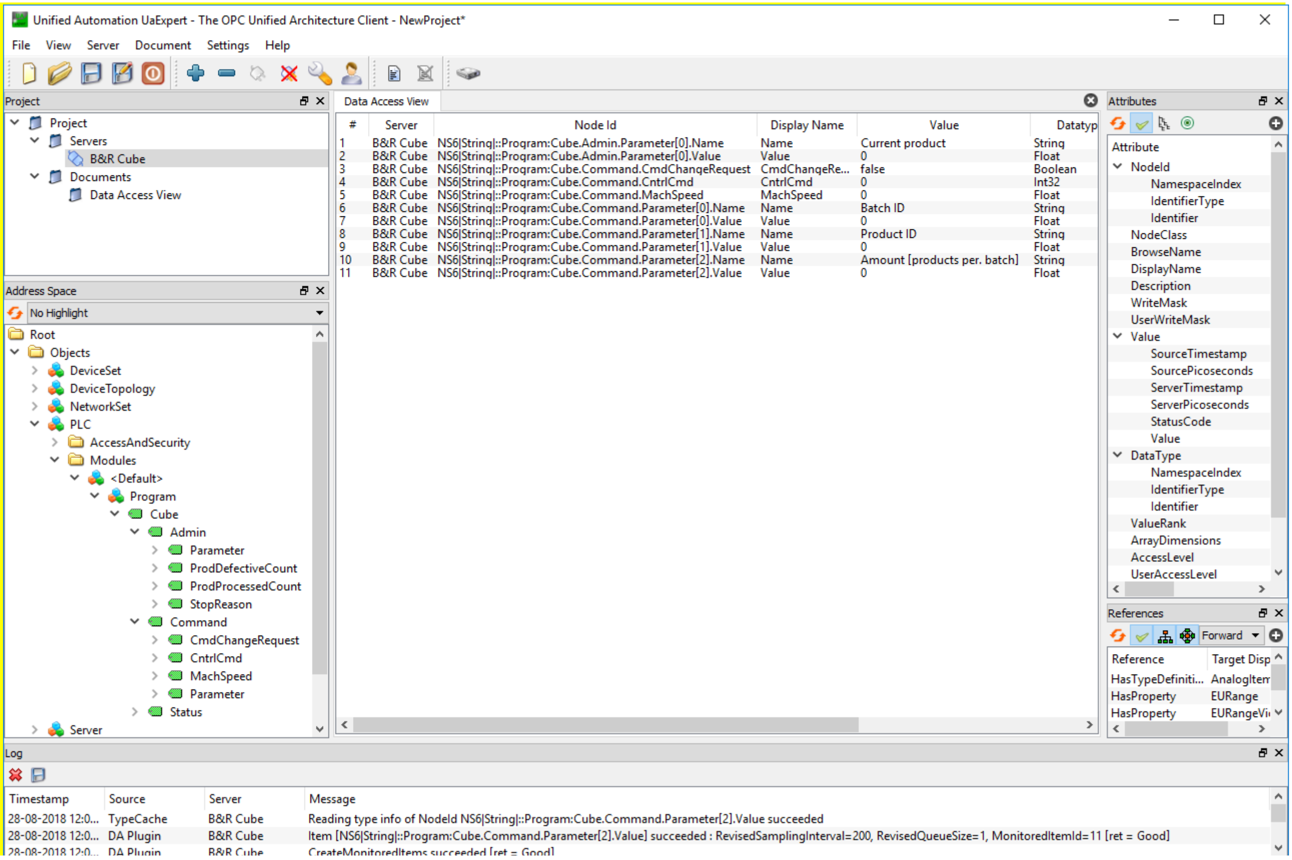


**Figure 11: OPC UA client UaExpert connected to the machine**

## 2.3.2 PackML

The logic of the machine is programmed using PackML[3] [4] (ANSI/ISA-TR88.00.02). From the communication interface the current state of the machine can be read. Using PackML commands the machine can be controlled via its interface.

Not all PackML states are used and implemented on the machine. Only the needed states for this production is implemented. Below is shown a figure of which states are available on the machine.

---

[3] https://en.wikipedia.org/wiki/PackML
[4] https://www.youtube.com/watch?v=4aYTqiOTJT4

### 2.3.3 Machine states

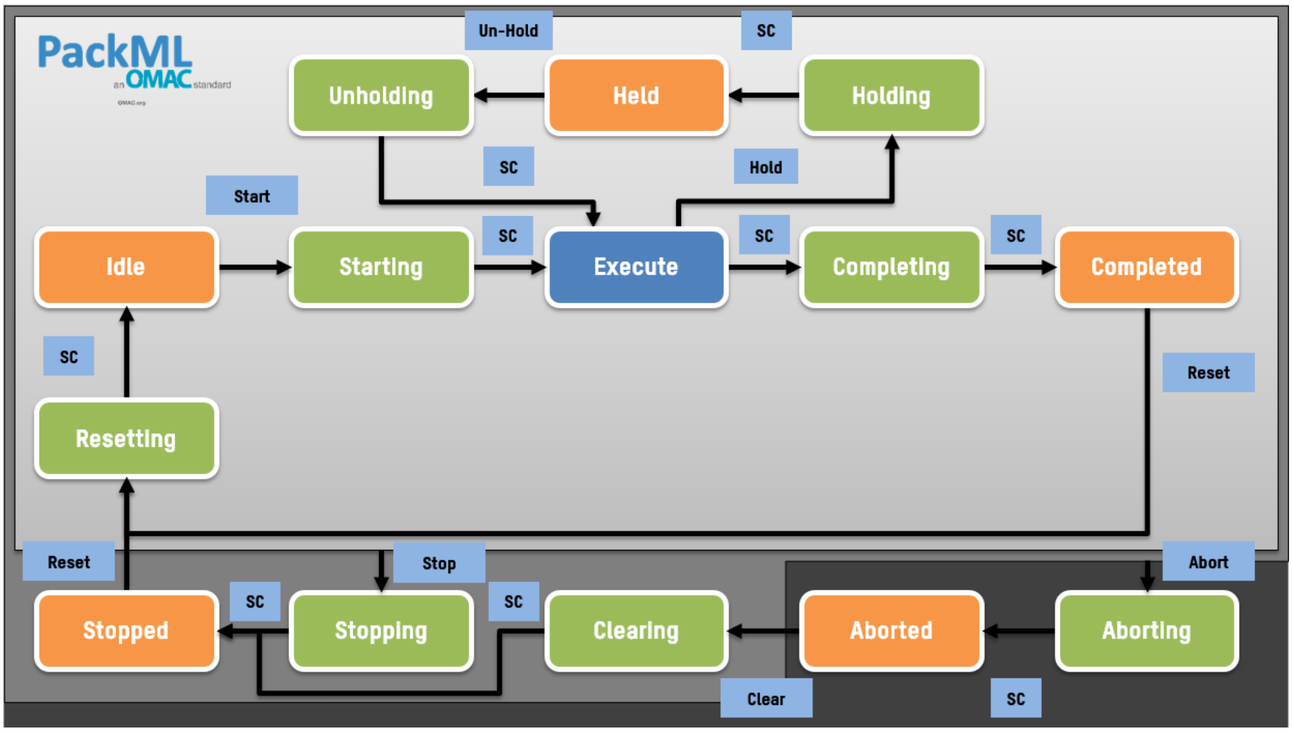


**Figure 12: Available states in mode Production**

The state model is derived from the ANSI/ISA-TR88.00.02 standard.

| State | Data type | Value |
|---|---|---|
| Deactivated | Integer | 0 |
| Clearing | Integer | 1 |
| Stopped | Integer | 2 |
| Starting | Integer | 3 |
| Idle | Integer | 4 |
| Suspended | Integer | 5 |
| Execute | Integer | 6 |
| Stopping | Integer | 7 |
| Aborting | Integer | 8 |
| Aborted | Integer | 9 |
| Holding | Integer | 10 |
| Held | Integer | 11 |
| Resetting | Integer | 15 |
| Completing | Integer | 16 |
| Complete | Integer | 17 |
| Deactivating | Integer | 18 |
| Activating | Integer | 19 |

**Figure 13: PackML states**

### 2.3.4 Machine commands

The following PackML commands can be given to the machine: Reset, Start, Stop, Abort, and Clear.

The commands can be sent to the machine by the physical buttons on the machine, by sending them through OPC-UA or by the machines HTML5 visualization.


**Figure 14: Commands can set from the visualization**

PackML commands can be send to the machine in the PackTags [Command.PackMLCmd](#). Here the command value must be set. Afterwards the [Command.CmdChangeRequest](#) must be set to true to execute the desired command.

| Cube.Command.PackMLCmd | Command |
|---|---|
| 0 | - |
| 1 | Reset |
| 2 | Start |
| 3 | Stop |
| 4 | Abort |
| 5 | Clear |

**Figure 15: PackML commands**

## 2.3.5 PackTags

PackTags are defined in the PackML specification and therefore they provide a way to communicate with different machines in a uniform way. Through the PackTags structure it possible to remote control the machine and collect various data from the machine. Defined parameters can also be sent to the machine using the PackTags. The table below gives an overview of the available machine PackTags.

| Name | Read/Write | Valid input | Comment |
|---|---|---|---|
| **Admin** | | | |
| ProdProcessedCount | R | - | Amount of produced products. |
| ProdDefectiveCount | R | - | Amount of defective products. |
| StopReason.Id | R | - | 10: Empty inventory<br>11: Maintenance needed<br>12: Manual stop<br>13: Motor power loss<br>14: Manual abort |
| StopReason.Value | R | - | |
| Parameter[0] | R | - | ID of product in batch. |
| **Status** | | | |
| StateCurrent | R | - | Current PackML state. |
| MachSpeed | R | - | Current machine speed in primary products per minute. |
| CurMachSpeed | R | - | Normalized machine speed.<br>$0 \leq \mathrm{CurMachSpeed} \leq 100$ |
| Parameter[0] | R | - | Current Batch ID |
| Parameter[1] | R | - | Amount of products in current batch. |
| Parameter[2] | R | - | Relative humidity |
| Parameter[3] | R | - | Temperature |
| Parameter[4] | R | - | Vibration |
| **Command** | | | |
| MachSpeed | R/W | Dependent on product ID<br>0: $0 \leq value \leq 600$<br>1: $0 \leq value \leq 300$<br>2: $0 \leq value \leq 150$<br>3: $0 \leq value \leq 200$<br>4: $0 \leq value \leq 100$<br>5: $0 \leq value \leq 125$ | Machine speed in primary products per minute for next batch. |
| CntrlCmd | R/W | $1 \leq CntrlCmd \leq 5$<br>1: Reset<br>2: Start<br>3: Stop<br>4: Abort<br>5: Clear | PackML command. |
| CmdChangeRequest | R/W | true/false | Execute PackML command. |
| Parameter[0] | R/W | $0 \leq value \leq 65535$ | Batch ID for next batch. |
| Parameter[1] | R/W | $0 \leq value \leq 5$<br>0: Pilsner<br>1: Wheat<br>2: IPA<br>3: Stout<br>4: Ale<br>5: Alcohol Free | Product ID for next batch. |
| Parameter[2] | R/W | $0 \leq value \leq 65535$ | Amount of products in the next batch. |

**Figure 16: Available PackTags**

It is only possible to write to the Command structure. The Admin and Status structures are read only.

## 2.4 Visualization

The machine is equipped with a visualization. In this visualization it is possible to operate and to overview the ingredient containers on the machine. The visualization is made in HTML5 and can be access from a web browser.

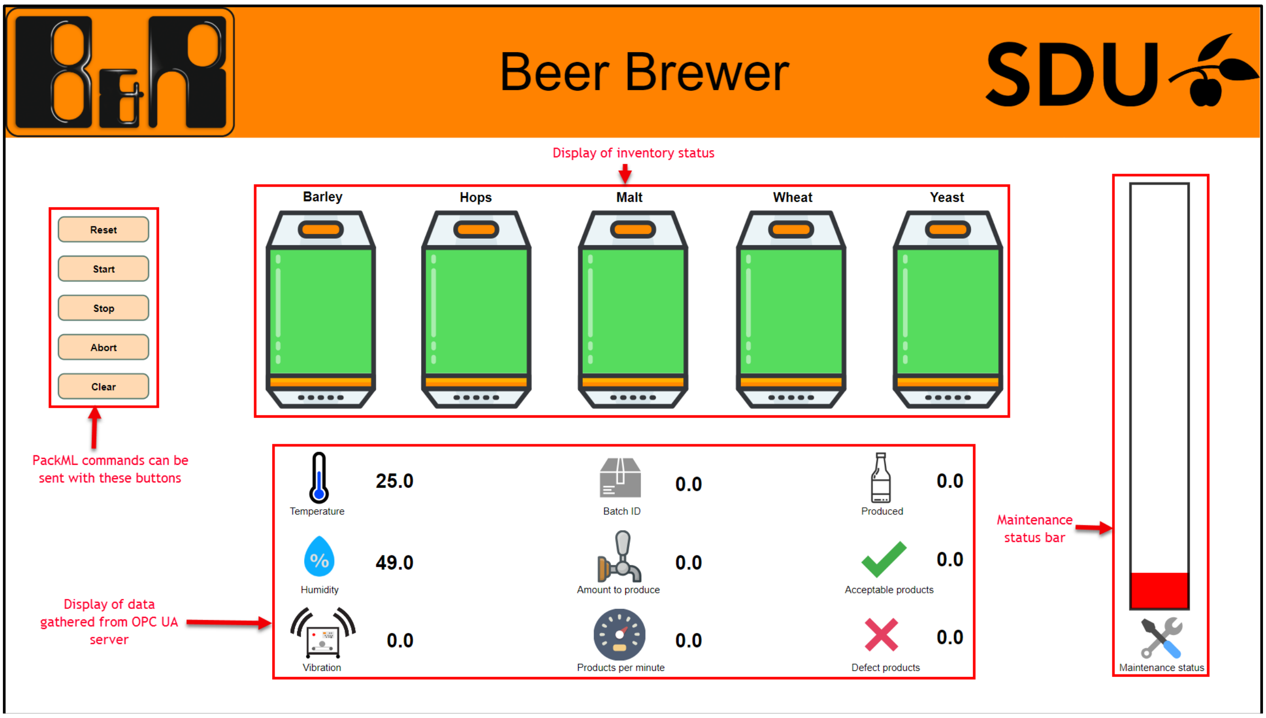


**Figure 17: Machine visualization shown in Chrome browser**

## 2.5 Parameters

The machine can be controlled by settings various parameters.

## 2.6 Hardware

The brewing machine is controlled by a PLC[5] (Programmable logic controller) from B&R[6]. The type of the PLC is a X20CP1586[7] controller.

## 2.7 Hardware simulation

Before the machine is put into production the brewery wants to make sure that the IT system can communicate with the brewing machine. Therefore the machine builder has created a small test cube that can simulate the brewing machine and the brewing process. That way the brewery can simulate the beer production and make sure that their IT systems can communicate with the machine.

By doing this simulation the commissioning of the real machine should be much faster and will not have a too large impact on the existing production.

[5] https://en.wikipedia.org/wiki/Programmable_logic_controller
[6] https://www.br-automation.com/
[7] https://www.br-automation.com/en-gb/products/control-systems/x20-system/x20-cpus/x20cp1586/

### 2.7.1 Turning on the simulator

Set to switch to upright position as shown in the figure below.

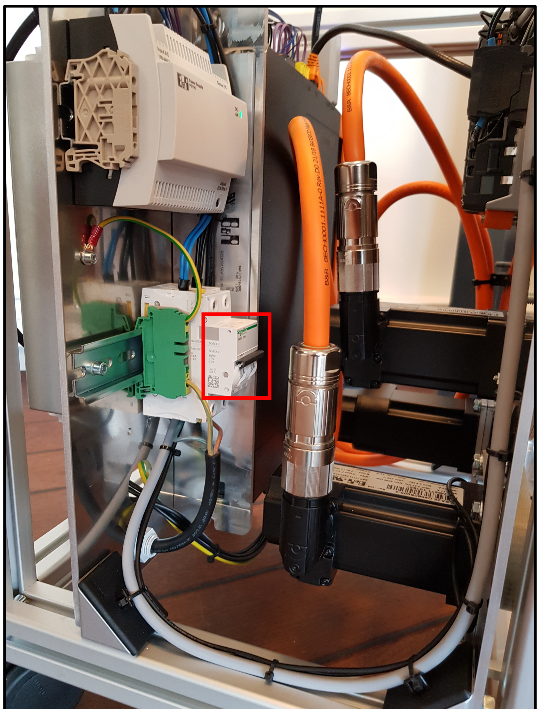

**Figure 18: Switch contact on to power on machine**

### 2.7.2 Wireless connection to the physical simulation hardware

The hardware simulator is equipped with a wireless router. This can be used to connect to the machine. Connect to wireless network “brCubeSoftwareTechnology”. Password is **acoposp3.**

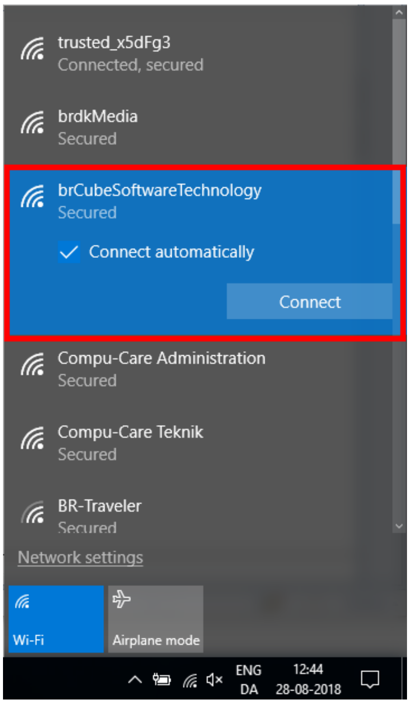


**Figure 19: Wireless network brCubeSoftwareTechnology**

### 2.7.3 Simulation panel

On the front of the physical machine there is a simulation panel. This panel can be used to operate the machine. In the figure below all the enabled buttons on the simulation panel have been highlighted by a red square.
When the machine is not in the PackML state "HELD", the buttons on the simulation panel will be mapped to PackML commands as shown in the table below.

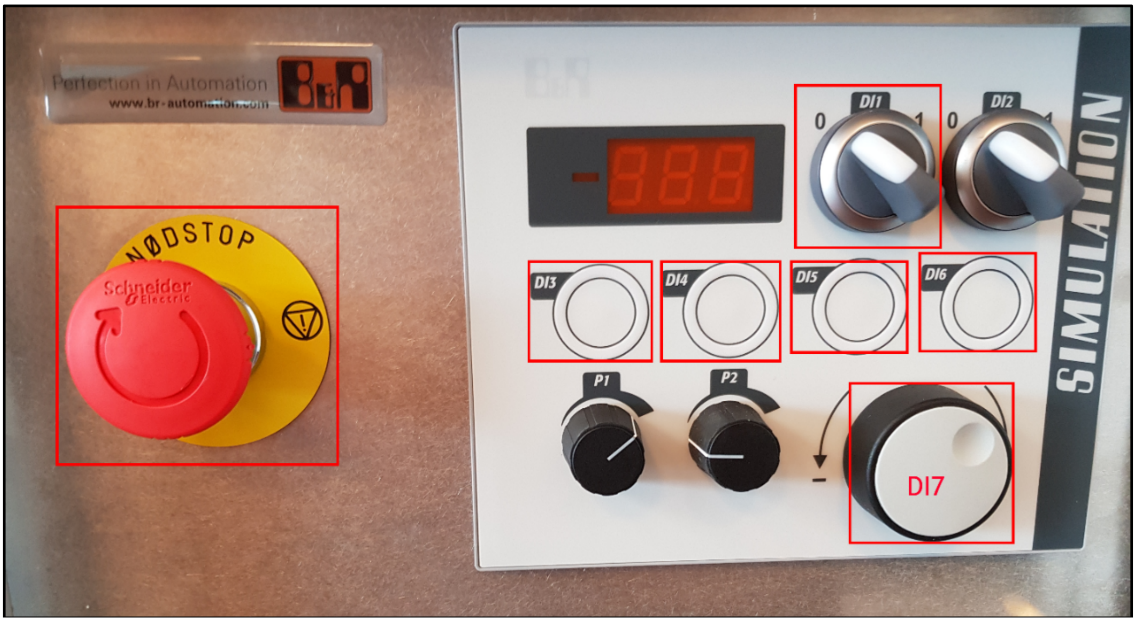


**Figure 20: Simulation panel on the physical machine**

**N.B. The hardware is also equipped with an emergency stop button that can be used to stop the machine if a dangerous situation should occur.**

| Command | Digital input |
|---|---|
| Reset | DI3 |
| Start | DI4 |
| Stop | DI5 |
| Abort | DI6 |
| Clear | DI7 |

**Figure 21: PackML commands mapped to physical simulation panel**

### 2.7.4 Empty inventory, refill needed

**Conditions**

- PackML state is HELD.
- StopReason.Id is 10.

**Impact on hardware**

- Buttons: DI3, DI4, DI5, DI6 and DI7 is disabled.
- LED on DI1 will light up.

**Procedure**

1. Flip the switch DI1 to ‘1’.
    a. Filling of the inventory will begin.
2. When the inventory is full: switch DI1 to ‘0’.
    **a.** The state will go back to EXECUTE.

### 2.7.5 Maintenance needed

**Conditions**

- PackML state is HELD.
- StopReason.Id is 11.

**Impact on hardware**

- Buttons: DI1 and DI7 is be disabled.
- Buttons: DI3, DI4, DI5 and DI6 will light red.

**Procedure**

1. Press DI3, DI4, DI5 and DI6 in any order.
    a. The buttons will light green.
    b. The maintenance status bar will slowly go down.
2. Once the maintenance status bar reaches zero the state will go back to EXECUTE.

### 2.7.6 Connection to OPC-UA server

Once the software simulator has been started the OPC-UA server can be accessed on opc.tcp://192.168.1.2:4840.

### 2.7.7 Connection to HTML5 visualization

The HTMl5 visualization also works in the software simulation. Once started it can be accessed on http://192.168.1.2:81/index.html?visuId=visu. It is recommended to use the Chrome[8] browser to access the visualization.

## 2.8 Software simulation

The machine can also be simulated in pure software. However not all machine features are available in the software simulation. It's not possible to collect temperature, moisture and errors from the machine in software simulation. For these the data the hardware simulation is needed.
But the OPC-UA communication works 100% in the software simulation. So to test and verify the machine communication the software simulation can be utilized.

**N.B. When using the visualization in simulation mode, the variables "Acceptable products" and "Defect products" will not change. If it is needed to simulate the production, the simulated physical machine should be used.**

### 2.8.1 Start the simulator

The simulator only runs on Windows OS. It can be started by executing the start.bat script. The script will start the simulator and open the visualization of the machine in the default browser.

**N.B. Chrome should be used to display the visualization.**

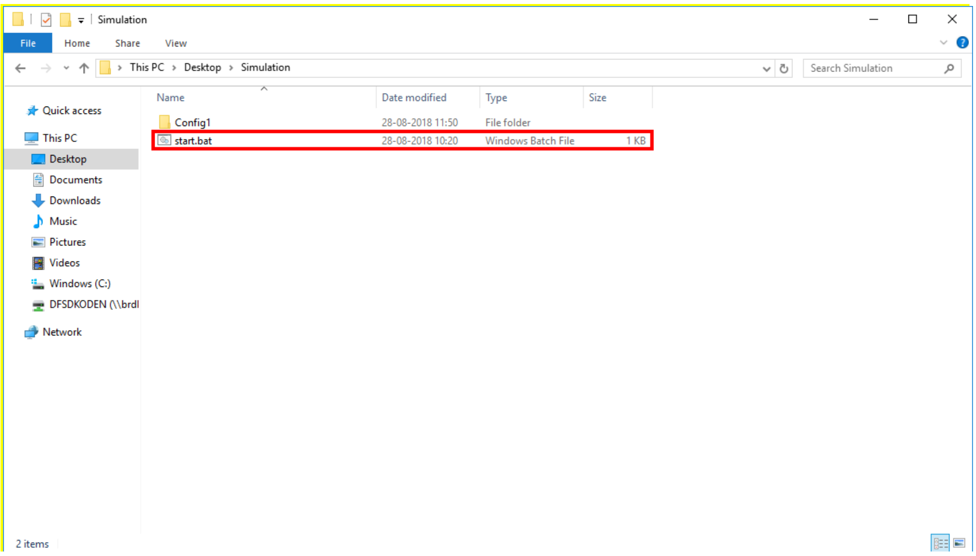


**Figure 22: Location of start.bat script**

Once the simulator is started it can be opened by double clicking on the icon in the system tray.

[8] https://www.google.com/chrome/

Figure 23: Location of software simulator

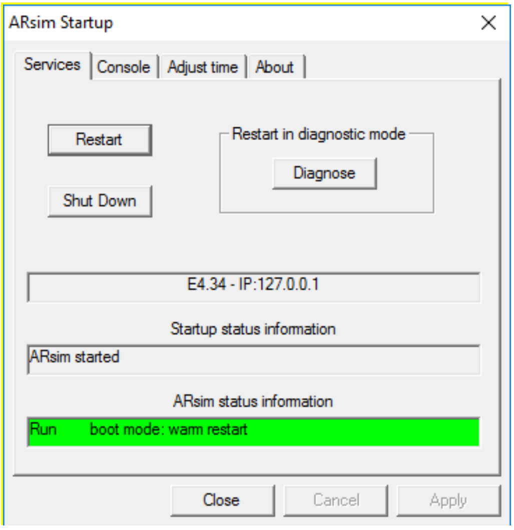


Figure 24: Simulation monitor

In the simulator window the status of the simulator can be monitored and the simulator can be restarted.

**N.B. After 2 hours of operation the simulator will stop and needs to be restarted manually.**

## 2.8.2 Connection to OPC-UA server

Once the software simulator has been started the OPC-UA server can be accessed on opc.tcp://127.0.0.1:4840.

### 2.8.3 Connection to HTML5 visualization

The HTMl5 visualization also works in the software simulation. Once started it can be accessed on http://127.0.0.1:81/index.html?visuId=visu. It is recommended to use the Chrome[9] browser to access the visualization.

[9] https://www.google.com/chrome/

# 3 The system

## 3.1 The purpose and use of the system

The purpose of the system is to make it possible for the production manager to plan the production in a way that is most efficient for the brewery so they can produce a large amount of beers fast with a good quality. So the brewery wants a MES[10] (Manufacturing execution system).

## 3.2 Requirements

The brewery has a list of requirements for the new system. The requirements are listed below.

### 3.2.1 Production control

The new MES must be able to control the brewery's production. It must be able to start and stop the production line. Furthermore it must also be able to monitor the production and collect data from the production line and store them for further analysis.

### 3.2.2 Batch control

The MES must have the functionality to keep track of the batches that the new machine is producing. Furthermore it must also be able to collect various data from the machine that is associated with the current batch number.

An example could be that a consumer reports that something is wrong with a beer. From the production time printed on the bottle the MES must be able to show all the products that was produced in this batch.

In order to control this the new brewing machine has the functionality that it can receive a batch number and return the batch number when the beer batch has been completed.

### 3.2.3 Production monitoring

The MES / SCADA[11] system must be able to monitor the production and display live relevant data from the machine.

### 3.2.4 Batch report

After a finished batch production the MES must be able to produce a batch report of the produced batch. The batch report must minimum contain the following.

- Batch ID.
- Product type.
- Amount of products (total, defect and acceptable).
- Amount of time used in the different states.
- Logging of temperature over the production time.
- Logging of humidity over the production time.

The batch report could be in PDF or dashboard style format. The data can be presented in various charts or in tables.

[10] https://en.wikipedia.org/wiki/Manufacturing_execution_system
[11] https://en.wikipedia.org/wiki/SCADA

### 3.2.5 Physical hierarchy

The documentation of the system must contain an illustration that defines the different components in the setup in relation to the ISA88[12][13] Part 1 Physical hierarchy model.

### 3.2.6 Visualization

The system must have a visualization that can be access and used to display the production data.

### 3.2.7 OEE

The system must be able to collect the necessary data form the machine and calculate the OEE[14][15][16] of the machine. The OEE must be available to be displayed by the system.

### 3.2.8 Estimate error function

Estimate the error function associated with the products.

### 3.2.9 Optimal production speed

When the machine is producing it will perform an error simulation which can result in a defect product. To find the optimal production speed it is needed to estimate the function upon which the error simulation is built.
The error simulation is a function of normalized machine speed and the selected product. This means that the error ratio will vary depending on the selected product and the machine speed. The normalized machine speed can be read under the PackTag Cube.Admin.CurMachSpeed.
The error simulation have implemented a degree of randomness meaning that the amount of defect products at a certain speed for a given product will not always be the same.

The optimal speed must be estimated for each product type.

[12] https://en.wikipedia.org/wiki/ISA-88
[13] https://www.youtube.com/watch?v=82DtwdJeEeU
[14] https://www.oee.com/
[15] https://www.youtube.com/watch?v=IdPuhCxLT1A
[16] http://omac.org/wp-content/uploads/2015/12/OMACandPackML_MettlerToledo.pdf

# 4 Figure Index

# 5 Appendix I – OPC-UA connection

## 5.1 Software simulation

To connect to the OPC-UA server the IP address of the PLC is needed. When running the software simulator the IP address is 127.0.0.1 and the server is available on port 4840. So the URL for the OPC-UA client is **opc.tcp://127.0.0.1:4840**.
The authentication information is:
Username: **sdu**
Password: **1234**

## 5.2 Machine simulation

To connect to the OPC-UA server the IP address of the PLC is needed. When running the physical machine simulator the IP address is 192.168.1.2 and the server is available on port 4840. So the URL for the OPC-UA client is **opc.tcp://192.168.1.2:4840**.
The authentication information is:
Username: **sdu**
Password: **1234**

## 5.3 Connection example

To create a connection to the OPC-UA server an OPC-UA client is needed. In this example the client UaExpert[17] from Unified Automation[18] is used.

### 5.3.1 Step I

Right click “Servers” and “Add”

[17] https://www.unified-automation.com/products/development-tools/uaexpert.html
[18] https://www.unified-automation.com/

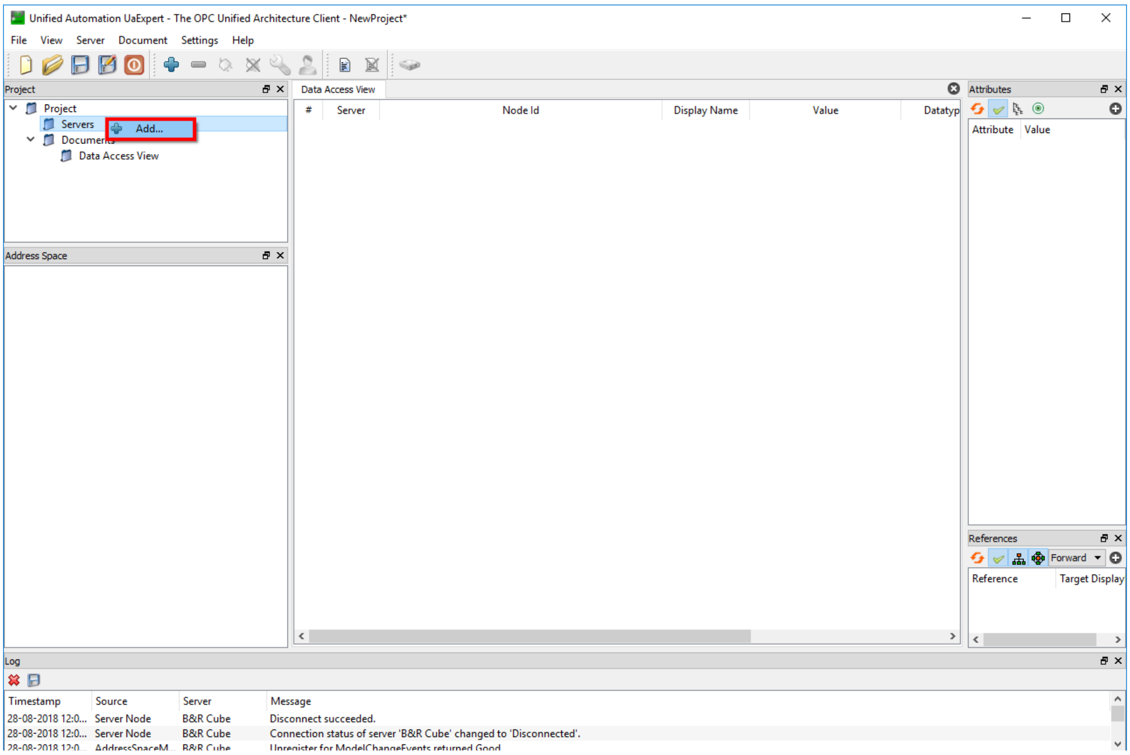


Figure 25: Add OPC-UA server

## 5.3.2 Step II

Fill out URL and authentication

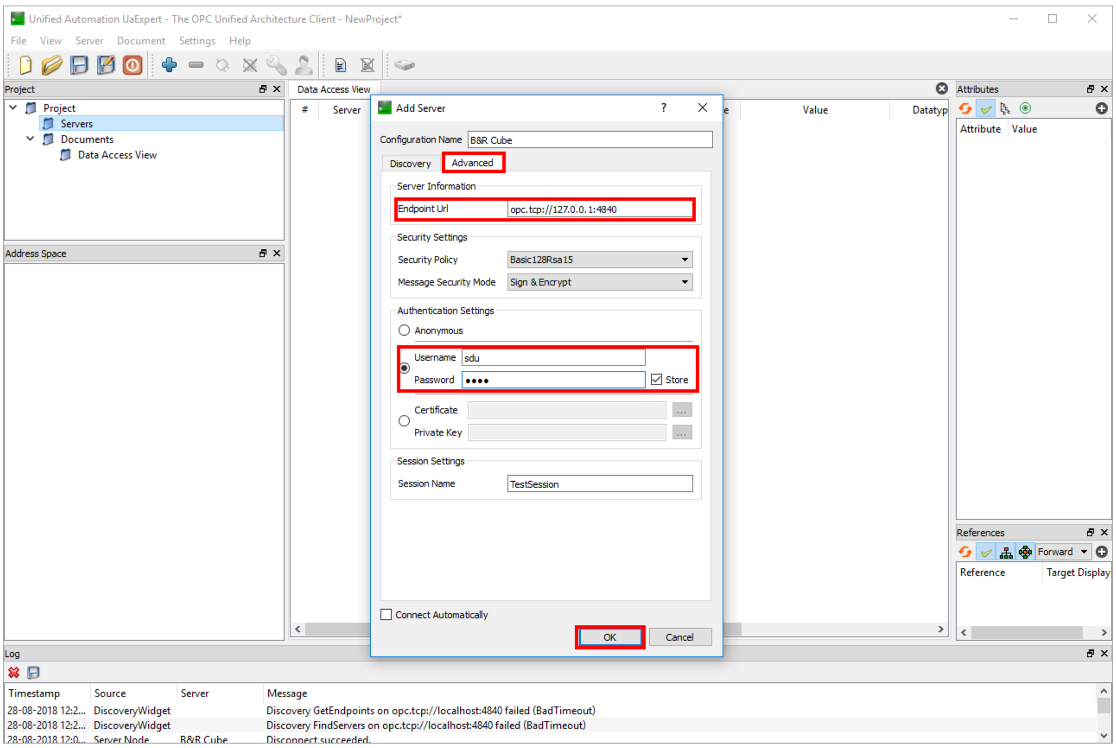


Figure 26: Fill out form

### 5.3.3 Step III

Connect to the server.

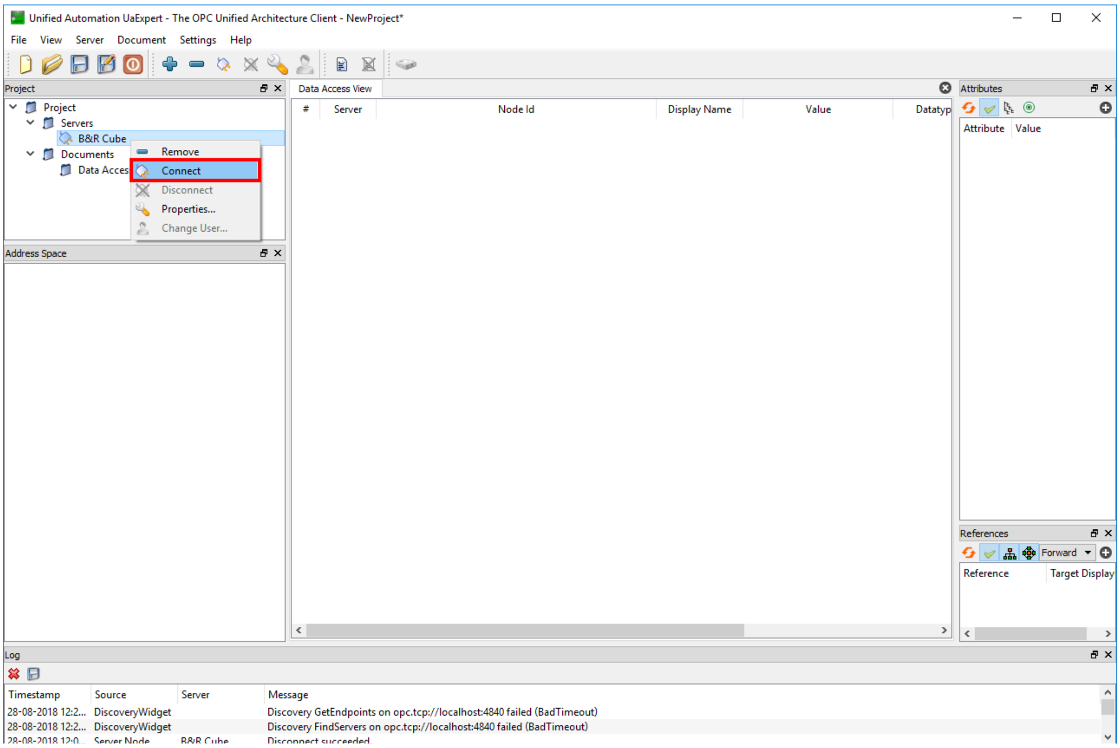


**Figure 27: Connect to OPC-UA server**

### 5.3.4 Step IV

Navigate to the Cube structure.

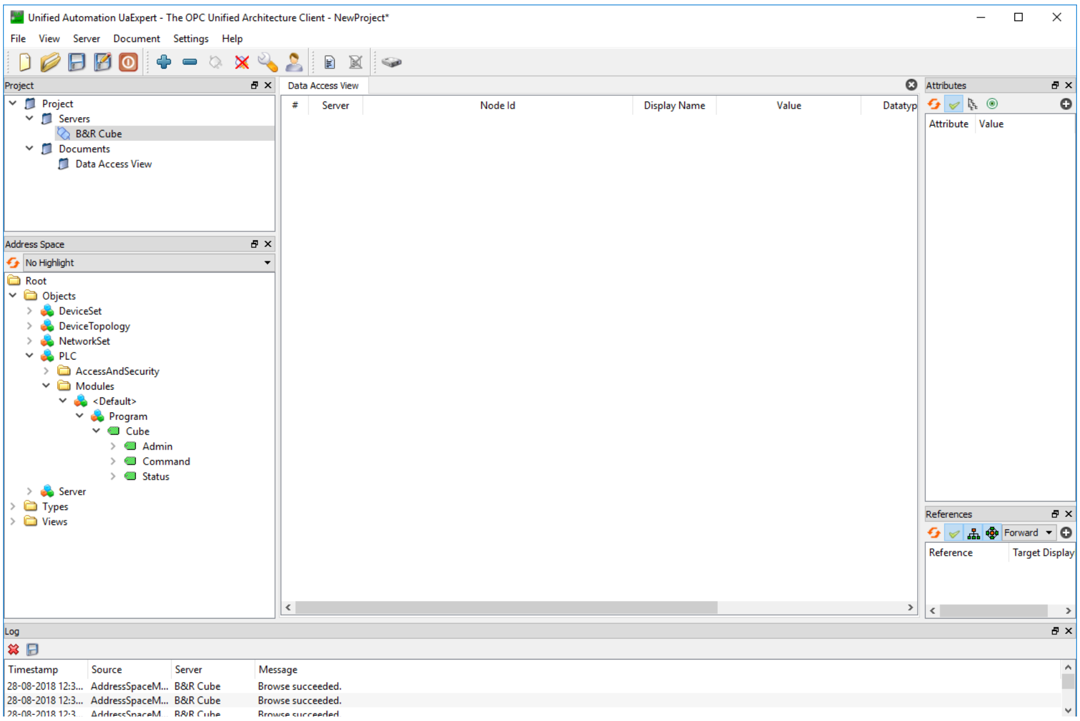


**Figure 28: Cube structure**

## 5.3.5 Step V

Drag the needed variables to "Data Access View". Now the variables can be monitored.

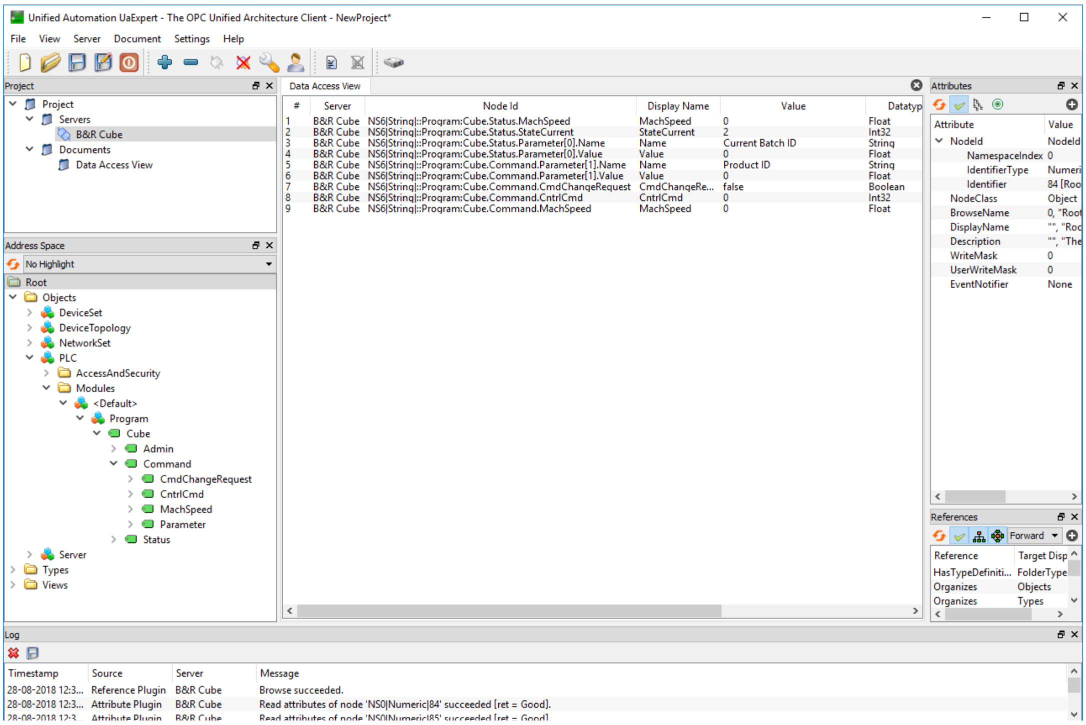


**Figure 29: Drag needed OPC-UA variables**

# 6 Appendix II – Troubleshooting

## 6.1 The software simulator starts in Service mode

If the software simulator starts in Service mode (yellow status bar) like shown in the figure below it can be restarted by pressing the restart button on the simulator. When the simulator should start up in Run mode (green status bar).

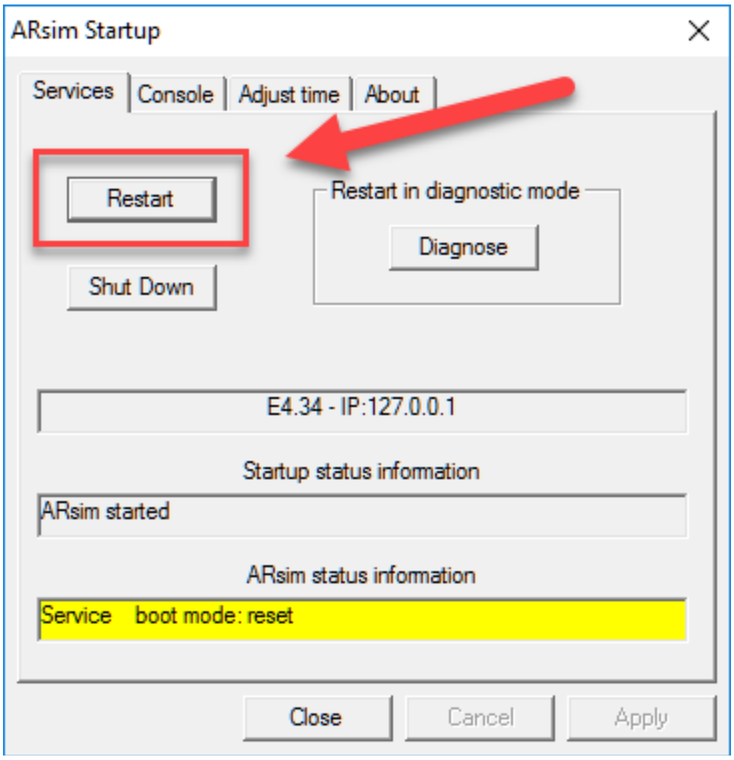


**Figure 30: Press Restart button to restart simulator**

## 6.2 The machine simulation stays in state Aborted

If the machine simulation stays in state Aborted then check if the emergency stop button on the machine has been released.

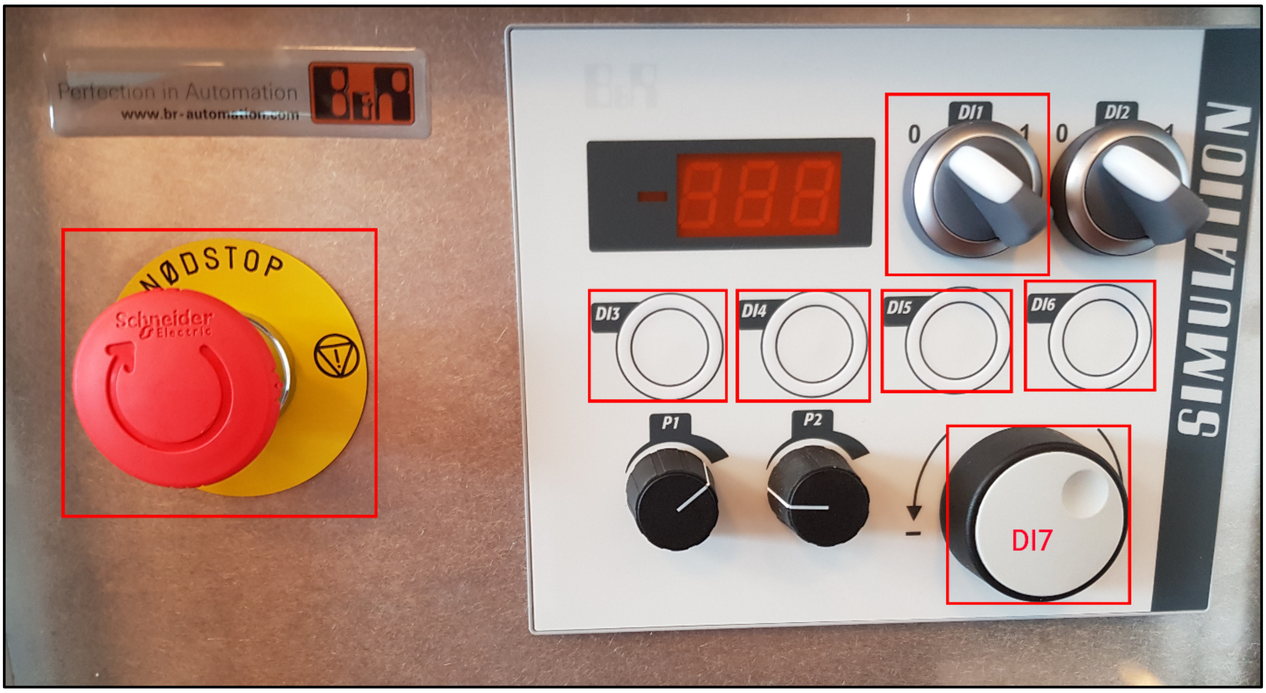


**Figure 31: Remember to release emergency stop button**

## 6.3 No connection to the OPC-UA server

Try to download a 3'rdparty OPC-UA client and follow the steps in the OPC-UA connection section.

## 6.4 The HTML visualization is not shown properly in my browser.

Make sure to use the Chrome browser for displaying the visualization. If the visualization is still not displayed properly like in the figure below try reload the browser by pressing CTRL+F5 (will clear the cache) or try to open an incognito window and access the browser from this window.


**Figure 32: Visualization not shown properly**

## 6.5 Visualization already open on this client

If the error message shown below is shown when trying to display the visualization try to close all browser windows can reopen the visualization.

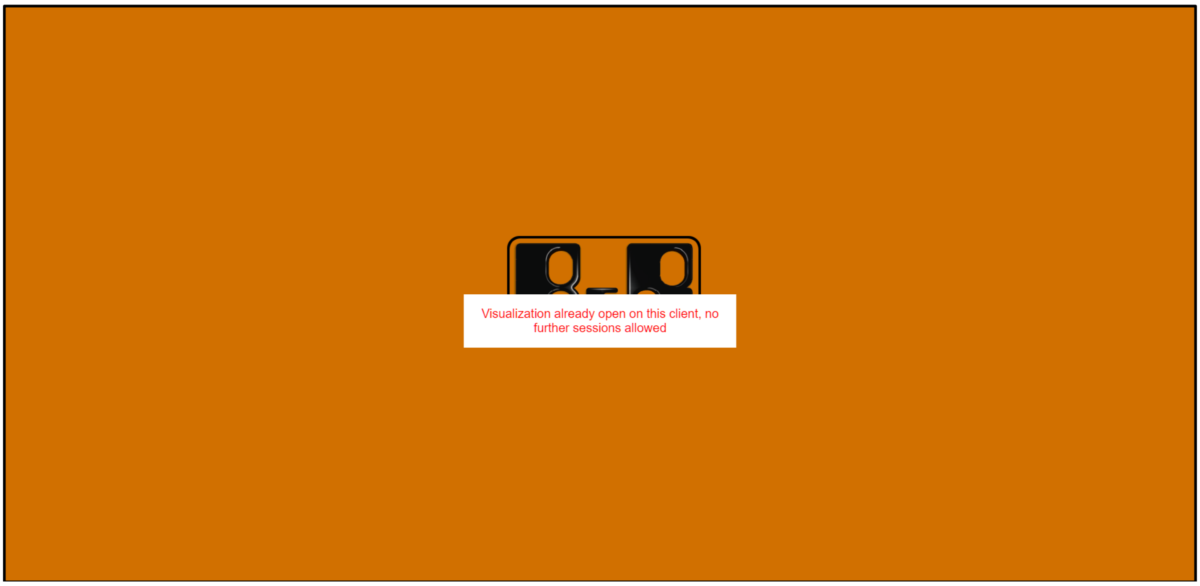


**Figure 33: Visualization can only be shown one time in a client**

## 6.6 No power on the hardware simulator

Check the switch inside the machine if it is turned on.

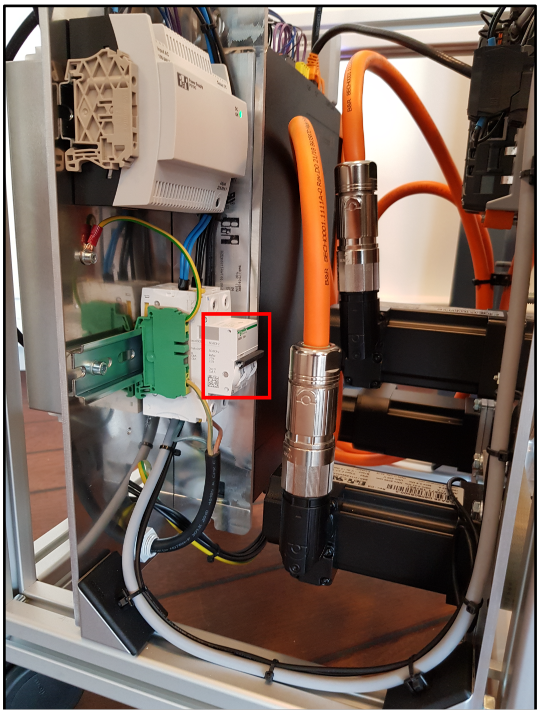

**Figure 34: Switch contact on to power on machine**

## 6.7 Windows cannot find directory of software simulator

If the error shown below occurs when trying to execute the start.bat script try to move the simulator to a different location on the harddrive e.g. C:\Simulation\

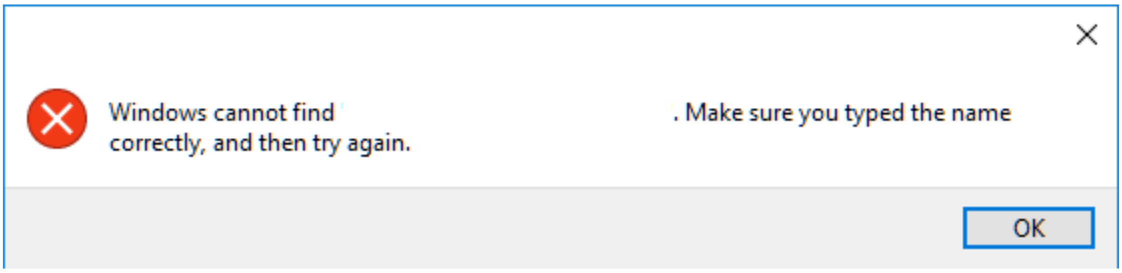


**Figure 35: Error Windows cannot find simulation folder**