TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Elektrotechnik und Informationstechnik Institut für Prozessleittechnik

Projektierung von Automatisierungssystemen

zum Thema

Entwicklung eines OPC-UA basierten Process Historian für modulare Anlagen

vorgelegt von Marc Becker, Erik Noak, Max Kirchner, Patrick Suwinski, Oliver Parczyk, Daniel Kluge
im Studiengang Elektrotechnik, Informationssystemtechnik

Betreuer: Dr. rer. nat. Valentin Khaydarov
Verantwortlicher Hochschullehrer: Prof. Dr.-Ing. habil. Leon Urbas
Tag der Einreichung: 17. Mai 2021

TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Elektrotechnik und Informationstechnik Institut für Prozessleittechnik

Fehlt noch !!

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

Betreuer:	Dr. rer. nat. Valentin Khaydarov
Hochschullehrer:	Prof. Dr.-Ing. habil. Leon Urbas
Tag der Einreichung:	17. Mai 2021

Bearbeiter: Marc Becker, Erik Noak, Max Kirchner, Patrick Suwinski, Oliver Parczyk, Daniel Kluge

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

In der Automatisierungstechnik werden modulare Anlagen immer häufiger verwendet, da es viele Vorteile bietet. Es ist möglich, Anlagen aus verschiedenen Komponenten zusammenzustellen und je nach Bedarf können einzelne Module einfach ausgetauscht bzw. erweitert werden. Diese Modularisierung stellt bei der Entwicklung neue Herausforderungen. Es müssen Schnittstellen implementiert werden, sogenannte Module Type Packages (MTP), so dass die Kommunikation und Funktionalität zwischen den einzelnen Process Equipment Assembly (PEA), auch Modul genannt, gewährleistet wird. Eine modulare Anlage benötigt einen Datenaustausch zwischen den einzelnen PEA, um als gesamte Einheit funktionieren zu können. Dabei unterscheidet man zwischen Engineering Daten, Live-Daten und historische Daten. Ziel dieses Projektes ist es, ein OPC UA-basiertes Tool zu entwickeln, Process Historian genannt, um den Prozess überwachen und auswerten zu können.

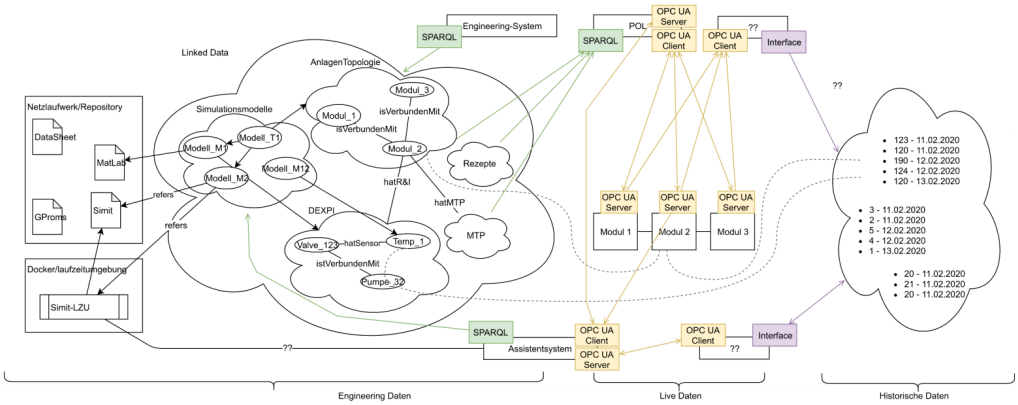

Abbildung 1.1: Datenstruktur

Ausgehend von der Triple-Store Datenbank wird die Information erhalten, welche Prozesswerte vorhanden sind. Diese werden in Form von Attributen wie NodeID, Name und Namespace in einer Konfigurationsdatei abgespeichert. Diese Datei legt zusätzlich die Art und Weise fest, wie Live-Daten abgefragt werden. Nach dem Subscriben bzw. Pollen der Prozessdaten über den OPC UA Client werden über einen Buffer die Daten in die InfluxDB

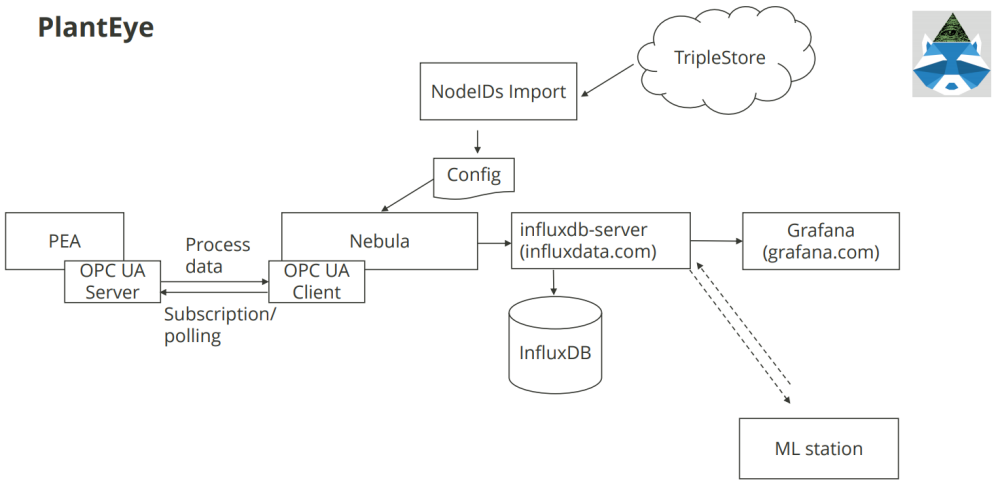

Abbildung 1.2: Generelle Struktur der Architektur

hochgeladen und abgespeichert. Auf die abgespeicherten Daten wird mit der Auswertungsplattform Grafana zugegriffen, womit diese visualisiert werden.

1.1 Aufgabenplanung

Das Ziel der Gruppe ist es, das Projekt parallel zu bearbeiten, weshalb es notwendig ist, über GitHub eine Versionskontrolle einzuführen. Für das erfolgreiche Abschließen des Projekts ist die Einführung von Etappenzielen, sogenannten Meilensteinen, sehr wichtig. Für die Planung der Zwischenziele wurde eine Tabelle erstellt, in welcher das Ziel mit einer Deadline festgehalten wird. Somit hat jedes Gruppenmitglied einen schnellen Überblick über die zu erledigenden Aufgaben und es ist ersichtlich, welcher Stand erreicht wurde. Aufgrund der Komplexität der Aufgabe wächst diese Tabelle zunehmend mit den wachsenden Erkenntnissen. Mit einer grünen Farbcodierung ist ersichtlich welche Meilensteine bereits abgeschlossen wurden. Gelb und Rot charakterisieren sehr wichtige Termine.

In diesem Projekt hat Max Kirchner die Position des Projektmanagers inne und ist in erster Linie für die Planung als auch die Kommunikation mit dem Betreuer zuständig. Die Architektur wird von Daniel Kluge und Oliver Parczyk geplant. Das beinhaltet die Strukturierung des Projekts in Form von UML-Klassendiagrammen, Komponentendiagrammen und Use-Case-Diagrammen. Erik Noack, Patrick Suwinski und Marc Philipp Becker

Meilenstein	Deadline	Verantwortlich
Start	13.04.2021	Alle
Kickoff	21.04.2021	Alle
Selbststudium	28.04.2021	Alle
UML-Klassendiagramm	12.05.2021	Daniel, Oliver
Problemanalyse, Arbeitsplan	12.05.2021	Max, Marc
Theorie zu MTP, OPC	12.05.2021	Patrick, Erik
Doku Architektur	12.05.2021	Daniel, Oliver
Folien Präsentation	18.05.2021	Alle
Zwischenpräsentation	19.05.2021	Alle
Dokumentation aller Ergebnisse	07.07.2021	t.b.a
Beschreibung Implementierung	07.07.2021	t.b.a
Dokumentation geeigneter Tests bzw. Validierungstests	07.07.2021	t.b.a
Kritische Diskussion der Arbeitsergenisse	07.07.2021	t.b.a
Abgabe 1	14.07.2021	Alle
Verteidigung	21.07.2021	Alle

Abbildung 1.3: Meilensteine

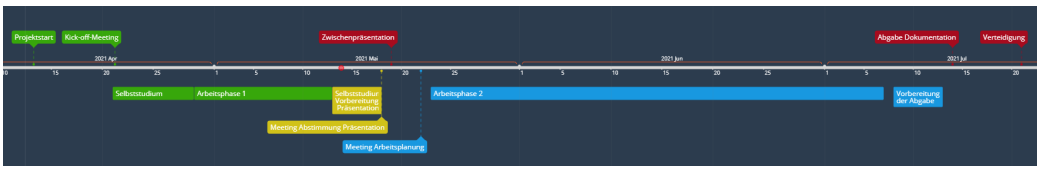

Abbildung 1.4: Timeline der Projektplanung
„Web-Tool Timeline der Projektplanung“, 04-05-2021

sind hauptsächlich für die Implementierung verantwortlich. Für eine erfolgreiche Umsetzung ist eine gute Zusammenarbeit zwischen den Gruppen notwendig, weshalb die Aufgaben auch gemeinsam gelöst werden.

2 Grundlagen und Stand der Technik

2.1 Open Platform Communication

2.1.1 Entwicklung

2.2 OPC - Unified Architecture

2.2.1 Aufbau

2.2.2 Sicherheit

2.2.3 Anwendungsbereiche

2.3 Module Type Package

3 Architektur

3.1 Komponentendiagramm

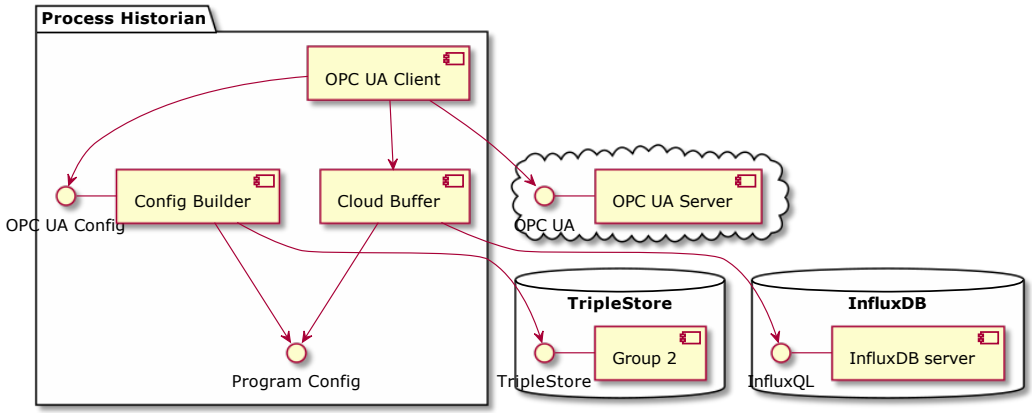

Abbildung 3.1: Komponentendiagramm

Der Process Historian besteht aus drei Komponenten: Dem OPC Client, dem Config Builder und dem Cloud Buffer. Der OPC Client Spricht mit der OPC UA-Schnittstelle des OPC UA-Servers. Der Config Builder erstellt aus den Informationen im TripleStore die in [Konfigurations-Schemata] näher beschriebene OPC Config. Der Cloud Buffer speichert die vom OPC Client erhaltenen Daten zwischen und versucht in einem gegebenen Intervall diese in die Influx-Datenbank zu laden. Config Builder und Cloud Buffer greifen weiter auf eine vom Nutzer zu parametrierende Program Config zu. Diese ist ebenfalls in den Konfigurations-Schemata (Kapitel 3.5) näher erläutert. Diese Unterteilung erlaubt eine gute Trennung der Belange und somit eine Wiederverwendbarkeit der Komponenten für ähnliche Systeme.

3.2 Klassendiagramm

Im Klassendiagramm spiegelt sich die im Komponentendiagramm (3.1) beschriebene Trennung der Belange wieder. CloudBuffer, OPCUAClient und ConfigBuilder finden sich so nun in eigenen Modulen wieder, die durch

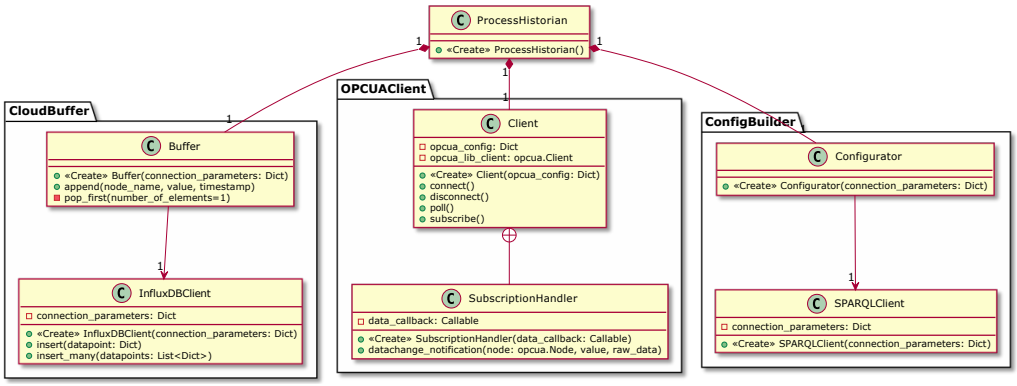

Abbildung 3.2: Klassendiagramm

den ProcessHistorian wiederum zusammengefügt werden.Der CloudBuffer besteht aus einer Buffer-Klasse und einem InfluxDBClient. Der Buffer nimmt werte vom ProcessHistorian entgegen und speichert diese Zwischen. In festgelegten Intervallen versucht er den Pufferinhalt durch den InfluxDBClient in die Datenbank übertragen zu lassen. Ist dies erfolgreich wird er die fehlerfrei gesendeten Daten aus dem Puffer löschen, falls nicht wird er zum nächsten Intervall eine erneute Übertragung des Pufferinhaltes versuchen.Der OPCUAClient besteht aus der Clientklasse selbst, die wiederum eine beinhaltet. Das verhalten des Clients wird durch die vom ConfigBuilder angelegte und in Kapitel 3.5 näher beschriebene Konfigurationsdatei festgelegt. Die SubscriptionHandler-Klasse wird von der OPC UA-Bibliothek benötigt.Der ConfigBuilder besteht aus einem Configurator, der die Konfigurationsdatei für den OPC UA-Client erstellt und einem SPARQLClient, der die Kommunikation mit dem TripleStore übernimmt. Dadurch ist auch innerhalb der Module wieder eine Trennung der Belange gewährleistet. So könnte beispielsweise der InfluxDBClient im CloudBuffer durch einen Client für eine andere Datenbank ersetzt werden.

3.3 Sequenzdiagramm

Das Diagramm zeigt den schematischen Ablauf des Programmes. Im Allgemeinen läuft die Erstellung und alle Kommunikation zwischen den Komponenten in der Klasse ProcessHistorian ab. Direkt nach dem Start wird im ersten Schritt der Configurator und damit auch der SPARQLClient aus dem ConfigBuilder-Paket initialisiert. Über diese beiden Komponenten wer-

den die Daten des TripleStores abgerufen und in eine Datei - die OPC UA-Konfiguration - geschrieben. Diese Konfiguration wird im nächsten Schritt bei der Erstellung des Clients des OPCUAClient-Paketes von dieser Klasse direkt geladen. Durch die Initialisierung wird auch die interne Klasse SubscriptionHandler erstellt. Im letzten Schritt der Programminitialisierung wird der sogenannte CloudBuffer zusammen mit seiner InfluxDB Schnittstelle erstellt. Im weiteren Programmverlauf kommt es durch die Möglichkeit Daten vom OPC UA Server zu pollen oder vorher darauf zu subscriben zu unterschiedlichen weiteren Abläufen. In einem Fall werden aktiv vom OPC UA Client nach einer gewissen Zeit alle Daten gepollt, während in dem anderen Fall vom Server eine Benachrichtigung mit den aktualisierten Daten gesendet wird. In beiden Fällen werden die aktualisierten Daten nun über den Process-Historian an den CloudBuffer übergeben, der seinerseits nach einer gewissen Zeit alle gesammelten Datenpunkte an die Influx-Datenbank überträgt. Sollte dieser letzte Schritt fehlschlagen, behält er die Daten und versucht es nach einer kurzen Wartezeit erneut.

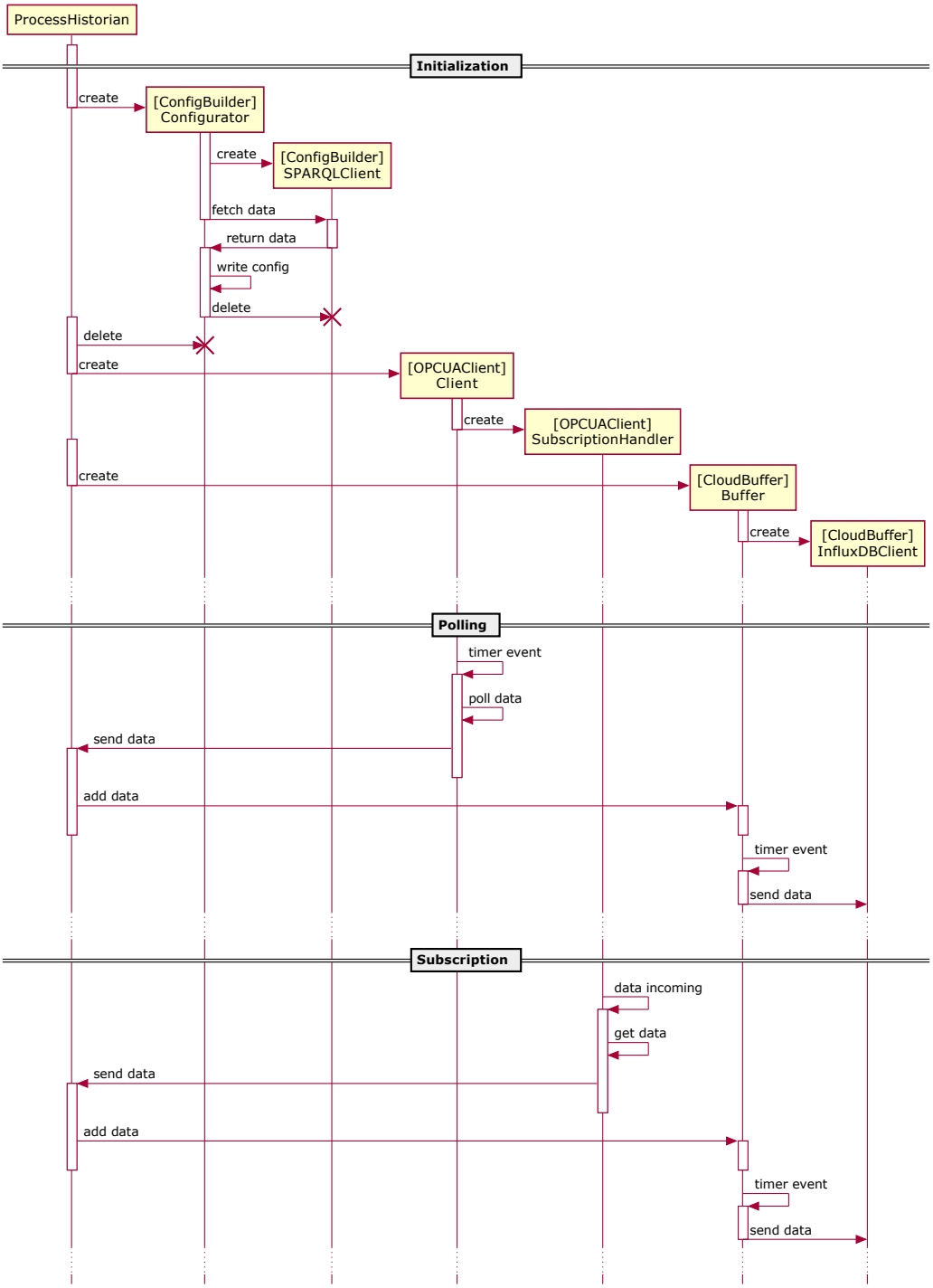

Abbildung 3.3: Sequenzdiagramm

3.4 Anwendungsfalldiagramm

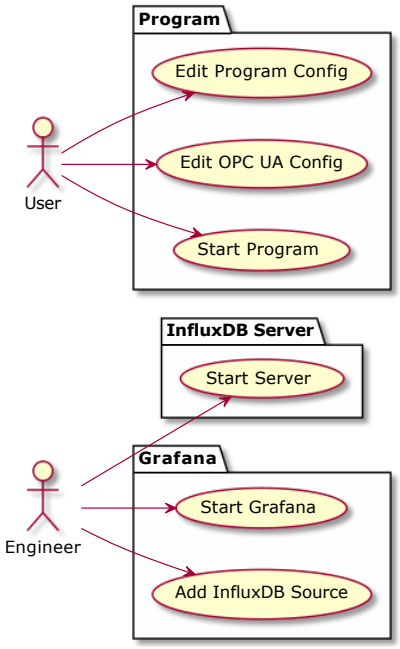

Abbildung 3.4: Use-Case-Diagramm

Im Anwendungsfalldiagramm ist ersichtlich, dass es zwei getrennte Rollen gibt. Unterschieden wird dabei zwischen dem „User“, der den Process Historian konfigurieren und starten kann und dem „Engineer“, der für die Er- und Bereitstellung der Influx-Datenbank zuständig ist. Während der Entwicklung kann und muss ein Entwickler beide Rollen einnehmen.

3.5 Konfigurations-Schemata

Bei den beiden dargestellten Schemata ist die Struktur der Konfigurationsdateien aufgezeigt. In der Programm-Konfiguration werden die notwendigen Parameter eingestellt, die zum Betrieb der Software notwendig sind. Die Einstellungen sind jeweils unterteilt nach ihrem jeweiligen Bereich. Die Namen beschreiben dabei jeweils das Element ausreichend, sodass auch ein unerfahrener Benutzer diese Einstellungen tätigen kann. Die OPC UA-Konfiguration wird automatisch angelegt, wenn die Daten vom TripleStore abgefragt werden, kann aber im Nachhinein von Hand verändert werden, um so z.B. die Abrufmethode für ein Attribut zu ändern.

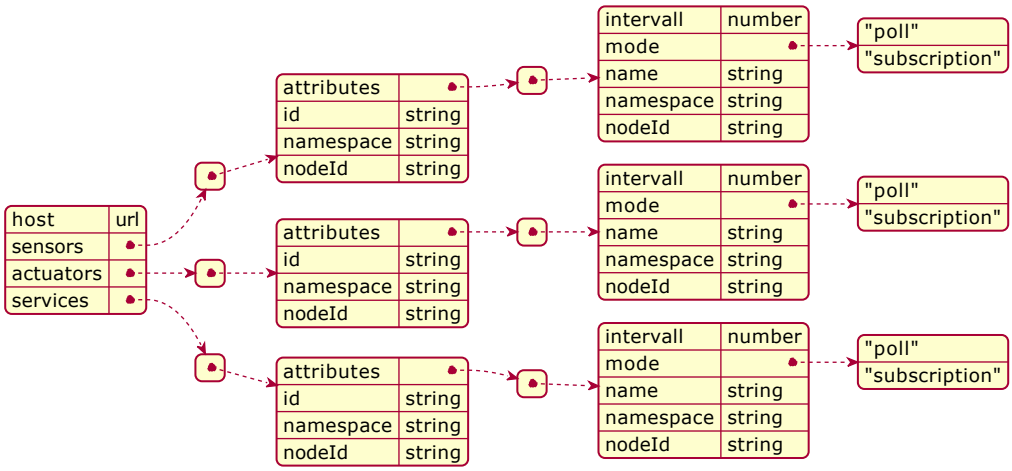

Abbildung 3.5: opcua-config-schema

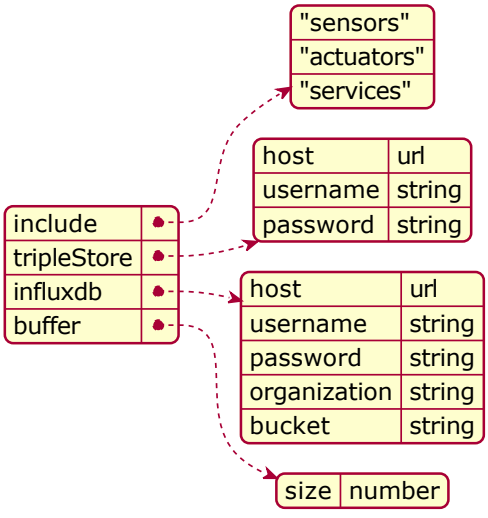

Abbildung 3.6: program-config-schema

4 Systemanforderungen

Voraussetzungen für den Betrieb der Software sind:

- Betriebssystem: Linux oder Windows
- Python 3.8+
- CPU: Empfohlen mindestens Dual-Core mit 1 GHz
- Arbeitsspeicher: Experimentell wurden für den CloudBuffer bei einer Größe von 1.000.000 Einträgen eine Speichergröße im RAM von rund 9MB ermittelt.

MacOS als Betriebssystem erfüllt theoretisch alle Anforderungen, kann aber durch uns nicht getestet werden.

5 Bedienungsanleitung

Bevor der Benutzer das Programm starten kann, sollte er vorher die Konfigurationsdatei anpassen. Die einzelnen Parameter erklärt:

- include: Liste, welche Art von Geräteklassen vom OPC UA Client abgerufen werden soll.
  - Mögliche Werte: „sensors“, „actuators“und/ oder „services“.
- tripleStore: Die Verbindungsparameter zum TripleStore.
- influxdb: Die Verbindungsparameter zur InfluxDB.
- buffer: Die Größe des CloudBuffers. Alle Eintrage nach Erreichen der maximalen Größe verdrängen den ältesten Wert aus der Liste.

Nachfolgend kann der Nutzer das Programm starten mit dem Befehl:

```
$ python3 main.py
```

Alternativ kann die Anwendung auch in einem Docker-Container gebaut und zum Laufen gebracht werden:

```
# docker build -t process_historian
# docker run -v ./config.yaml:/app/config.yaml:r -d process_historian
```

Außerdem existiert eine Docker-Compose-Konfiguration, die die beiden Befehle vereinfacht:

```
# docker-compose build
# docker-compose up -d
```

Literaturverzeichnis

BSI Prüfung OPC UA. (10-05-2021). https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/Studien/OPCUA/OPCUA.html

Classic OPC. (10-05-2021). https://opcfoundation.org/about/opc-technologies/opc-classic/

DIN, DKE & VDE. (17-05-2021a). DIN EN 62541-4 OPC unified architecture - Part 4: Services (IEC 62541-4:2015). https://katalog.slub-dresden.de/id/dswarm-201-REUzMDA2NDAwNQ/#detail

DIN, DKE & VDE. (17-05-2021b). DIN EN 62541-5 OPC unified architecture - Part 5: Information Model (IEC 62541-5:2015). https://katalog.slub-dresden.de/id/dswarm-201-REUzMDA2NDAwMQ/#detail

History OPC. (10-05-2021). https://de.wikipedia.org/wiki/Open_Platform_Communications

MES-Systeme. (11-05-2021). https://www.team-con.de/unsere-leistungen/produkte/manufacturing-execution-system

OPC Foundation Members. (10-05-2021). https://opcfoundation.org/members

OPC UA over TSN. (13-05-2021). https://de.wikipedia.org/wiki/OPC_UA_TSN

OPC UA und Industrie 4.0. (13-05-2021). https://opcfoundation.org/wp-content/uploads/2016/05/OPC-UA-Interoperability-For-Industrie4-and-IoT-DE-v5.pdf

OPC UA und Internet of Things. (13-05-2021). https://www.industry-of-things.de/was-ist-opc-ua-definition-architektur-und-anwendung-a-727188

SCADA Systeme. (11-05-2021). https://www.itwissen.info/SCADA-supervisory-control-and-data-acquisition-SCADA-Protokoll.html

Web-Tool Timeline der Projektplanung. (04-05-2021). https://time.graphics/de/line/507245. (Siehe S. 3)

Selbstständigkeitserklärung

Hiermit versicheren wir, Marc Becker, Erik Noak, Max Kirchner, Patrick Suwinski, Oliver Parczyk, Daniel Kluge, dass wir die vorliegende Projektarbeit im Modul Projektierung von Automatisierungssystemen zum Thema

Projektierung von Automatisierungssystemen - Entwicklung eines OPC-UA basierten Process Historian für modulare Anlagen

ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt haben; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Bei der Auswahl und Auswertung des Materials sowie bei der Herstellung des Manuskripts haben wir Unterstützungsleistungen von folgenden Personen erhalten:

Dr. rer. nat. Valentin Khaydarov

Weitere Personen waren an der geistigen Herstellung der vorliegenden Projektierung von Automatisierungssystemen nicht beteiligt.

Dresden, den 17. Mai 2021 .....

i.A. Patrick Suwinski