Digital Twin of Injection Molding

[1]

Leitfaden

Prozessdatenerfassung mittels OPC-UA

Marco Klute
Universität Kassel
Fachgebiet Kunststofftechnik

Alexander Rehmer, M.Sc.
Tel. +49 (0) 561 804 2953
alexander.rehmer@mrt.uni-kassel.de

Marco Klute, M.Sc.
Tel. +49 (0) 561 804 3629
marco.klute@uni-kassel.de

OPC-UA und EUROMAP

Open **P**latform **C**ommunication (OPC) ist ein Standard für einen sicheren und zuverlässigen Datenaustausch im Bereich der industriellen Automatisierung und anderen Branchen, der ursprünglich unter dem Namen **O**bject Linking and Embedding for **P**rocess **C**ontrol entwickelt wurde. Die Besonderheit von OPC ist, dass es sich hierbei um einen plattformunabhängigen Standard handelt, wodurch ein nahtloser Informationsaustausch zwischen Geräten verschiedener Hersteller gewährleistet wird. Für die Entwicklung und Pflege des OPC-Standards ist die OPC-Foundation verantwortlich, die 1996 gegründet wurde. Mittlerweile sind 860 Unternehmen (darunter Maschinenhersteller, Automatisierungstechniker, Endanwender, Programmierer, Forschungsinstitute, …) aus 51 Ländern an den Entwicklungen beteiligt und es wurden insgesamt elf Standards und Spezifikationsgruppen entwickelt.

Die **O**pen **P**latform **C**ommunication – **U**nified **A**rchitecture (OPC-UA) dient als hersteller- und plattformunabhängige, serviceorientierte Architektur für den Datenaustausch, wobei die übermittelten Daten (bspw. Maschinendaten wie Regelgrößen, Messwerte und Parameter) maschinenlesbar semantisch beschrieben werden. Dadurch lassen sich diese Daten einfach und eindeutig interpretieren. Ein weiterer wesentlicher Vorteil des OPC-UA-Standards ist die gesicherte Kommunikation ohne zusätzliche Hardware durch eine einfache Vernetzung auf Ethernet-Basis, wodurch bestehende Industrial Ethernet-Infrastrukturen genutzt werden können.

Insbesondere bei der Übertragung von großen Datenmengen oder größeren Dateien (bspw. JPG) ist die reine Ethernet-basierte Übertragung mittels OPC-UA nicht mehr echtzeitfähig, was dazu führen kann, dass für einen entwickelten digitalen Zwilling eines schnellen Prozesses die Daten nicht im Prozesstakt bereitgestellt werden können. Durch eine Kombination von OPC-UA mit der **T**ime-**S**ensitive **N**etworking (TSN) Technologie ist die Datenübertragungsrate um ein Vielfaches[1] erhöht, weshalb das sogenannte „OPC-UA over TSN“ als das Kommunikationsprotokoll für Industrie 4.0 und Internet of Things (IoT) angesehen wird.

Durch die Verwendung von OPC-UA over TSN steht für die Datenübertragung nicht nur der ursprüngliche Client/Server-Mechanismus zur Verfügung, sondern auch das insbesondere bei einem Netzwerk aus vielen Teilnehmern schnellere Publish/Subscribe-Modell. Während beim Client/Server-Mechanismus die Kommunikation immer nur zwischen einzelnen Clients und Servern über eine direkte Datenabfrage funktioniert, werden beim Publish/Subscribe-Modell von den Servern Daten an das Netzwerk gesendet (Publish), sodass diese von jedem Client empfangen werden, der diese Daten abonniert hat (Subscribe).

[1] Laut einem Whitepaper ist OPC-UA over TSN um den Faktor 18 schneller, verglichen mit einer herkömmlichen Industrial-Ethernet-Protokolle [2]

Auf dem OPC-UA-Kommunikationsstandard aufbauend wurden vom Dachverband der europäischen Kunststoff- und Gummimaschinenindustrie (European Plastics and Rubber Machinery, EUROMAP) Standards für die Kommunikation zwischen Maschinen und Peripheriegeräten der Kunststoffindustrie entwickelt. Folgende Standards stehen bisher zur Verfügung:

- EUROMAP 83: Allgemeine Typdefinitionen
- EUROMAP 77: Datenaustausch zwischen Spritzgießmaschinen und Manufacturing Execution Systems (MES)
- EUROMAP 79: Schnittstelle zwischen Spritzgießmaschine und Handlingsystemen/Roboter
- EUROMAP 82.1: Temperiergeräte
- EUROMAP 82.2: Heißkanalsysteme
- EUROMAP 82.3: LSR-Dosiersysteme
- EUROMAP 84: Datenaustausch zwischen Extrusionslinien und Manufacturing Execution Systems (MES)

Ermittlung der Node-IDs der abzufragenden Prozessgrößen

Durch die beschriebenen Kommunikationsprotokolle ist jedem Objekt (Parameter, Status, Information, …) eine eindeutige ID zugeordnet. Neben dem Wert des Parameters, der abgefragt werden soll, beinhaltet dessen Objekt bspw. auch den Zustand des Parameters. Über kostenlose Software-Tools wie bspw. UaExpert (Unified Automation GmbH), können die sogenannten Node-IDs der Parameter, die aus der entsprechenden Maschine exportiert werden sollen, ermittelt werden. Hierbei bietet es sich an, die eindeutige Parameterkennzeichnung des Maschinenherstellers für die Suche nach den jeweiligen Node-IDs zu verwenden. Tabelle 1 zeigt beispielhaft einige Prozessparameter, deren herstellerspezifische Kennzeichnung und die Node-ID, die für den Datenexport über OPC-UA notwendig ist (Kennzeichnungen und Node-IDs entsprechen einer Allrounder 470S, ARBURG GmbH + Co KG).

Tabelle 1: Auswahl relevanter Prozessparameter einer Allrounder 470S inklusive Kennzeichnung und Node-ID

Prozessparameter	Kennzeichnung	Node-ID
Zylinderheizzonentemperatur (soll)	T801, T802, …	207262, 207412, 207562, 207712, 207862
Zylinderheizzonentemperatur (ist)	T801I, T802I, …	207272, 207422, 207572, 207722, 207872
Einspritzstrom (soll)	Q305	201092
Nachdruckvolumenstrom (3 Stufen, soll)	Q311, Q312, Q313	201172, 416782, 416792
Nachdruckhöhe (3 Stufen, soll)	p311, p312, p313	201292, 201332, 201372
Nachdruckzeit (3 Stufen, soll)	t311, t312, t313	201282, 201322, 201362
Dosiervolumen (soll)	V403	201972
Dosiervolumen (ist)	V301I	201732
Umschaltvolumen (soll)	V305	201112
Umschaltvolumen (ist)	V4065	202422
Massepolster (ist)	V4062	202672
Staudruck (soll)	p403	201962
Max. Spritzdruck (ist)	p4055	202472
Umschaltspritzdruck (ist)	p4072	202522
Dosierzeit (ist)	t4015	202732
Einspritzzeit (ist)	t4018	202582

Neben den in der Tabelle gezeigten skalaren Maschinenparametern und Prozessgrößen lassen sich einzelne Größen auch als zeitlicher Verlauf exportieren. Hierzu müssen diese Prozessgrößen in den Mess- und Überwachungsgrafiken der Maschine dargestellt werden. Die dort dargestellten Trajektorien lassen sich über die Node-IDs der Grafiken abfragen. Bei der bereits erwähnten Allrounder 470S lassen sich vier Messgrafiken, die jeweils vier Prozessgrößen beinhalten können, acht Überwachungsgrafiken und vier erweiterte Überwachungsgrafiken konfigurieren. In Summe lassen sich demnach maximal 28 Prozessgrößenverläufe exportieren. Da jeder in den einzelnen Grafiken dargestellte Verlauf aus 512 Messpunkten besteht, errechnet sich die Frequenz der Trajektorien aus dem Quotienten der 512 Messpunkte und der für die Grafik eingestellten Aufzeichnungsdauer. Während die resultierende Frequenz bei kurzen Zykluszeiten sehr hoch ist, reicht sie unter Umständen bei längeren Zykluszeiten nicht aus, da insbesondere bei der im Vergleich sehr kurzen Einspritzphase für die Modellbildung relevante Informationen verloren gehen können. Um auch bei längeren Zykluszeiten eine ausreichende Frequenz gewährleisten zu können, empfiehlt es sich, die benötigten Trajektorien auf mehrere Grafiken aufzuteilen und die abgefragten Daten im Nachgang zu kombinieren. Bei einer Zykluszeit von ca. 30 Sekunden lässt sich die Frequenz durch die Verwendung von drei aneinandergehängten Grafiken von ursprünglich ~17 Hz auf ~51 Hz erhöhen. Abbildung 1 zeigt beispielhaft eine an der Allrounder 470S konfigurierte Messgrafik, die über die Dauer von 10,24 s die Soll- und Ist-Werte des Einspritzdrucks und des Einspritzvolumenstroms abbildet.

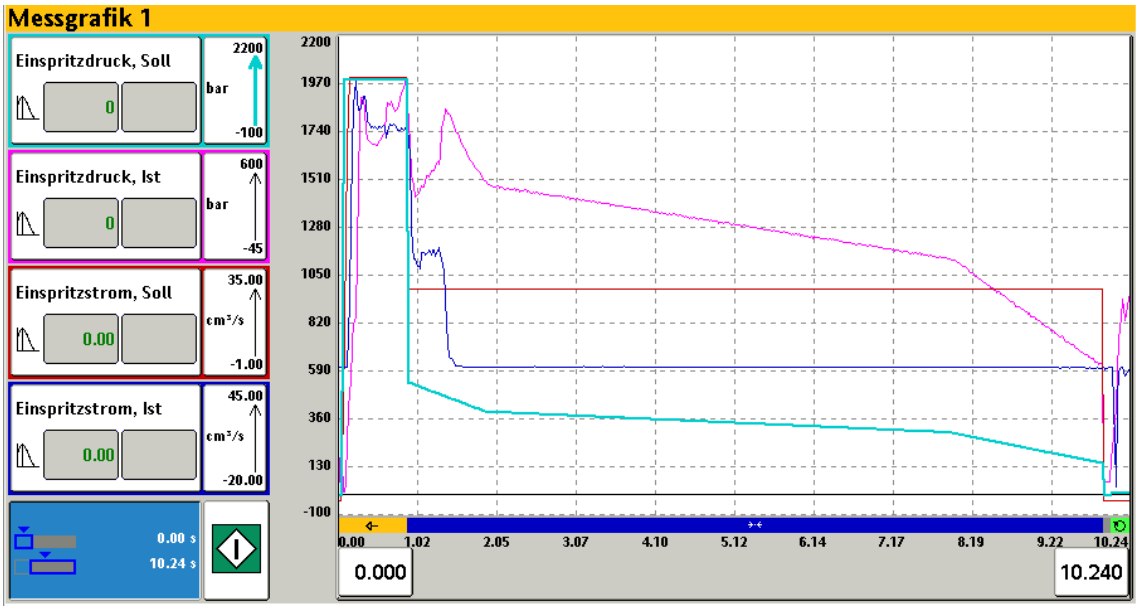

Abbildung 1: An einer ARBURG Spritzgießmaschine konfigurierte Messgrafik über die die Trajektorien der eingestellten Prozessgrößen exportiert werden können.

Grundlegender Aufbau des Software-Tools

Da es sich bei dem entwickelten Software-Tool lediglich um einen Python-Code handelt und keine grafische Benutzeroberfläche (GUI) existiert, wurde das Skript auf drei einzelne Dateien aufgeteilt, um die Anpassung des Skripts an den zu überwachenden Spritzgießprozess zu erleichtern. Außerdem kann so für jeden Prozess eine eigene Konfigurationsdatei erstellt werden, die dann beim Ausführen des Skripts eingelesen wird. Abbildung 2 gibt einen ersten Überblick über die drei Dateien und deren Inhalt. Auf die einzelnen Funktionen der drei Dateien wird im Folgenden detailliert eingegangen.

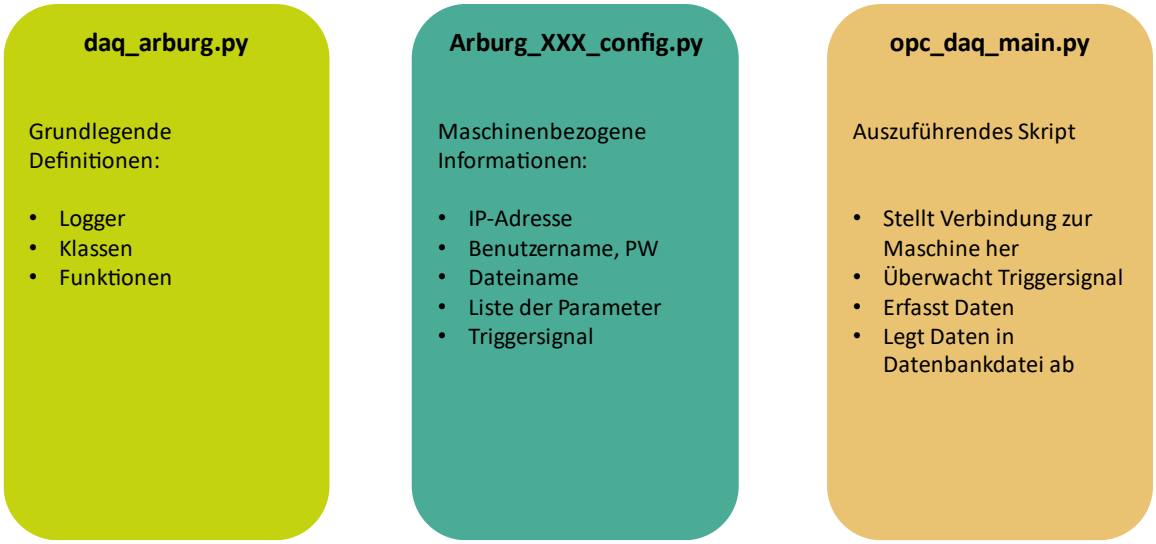

Abbildung 2: Aufbau der drei Skripte des Abfragetools

daq-arburg.py

Die Datei daq-arburg.py beinhaltet alle für die Abfrage wesentlichen Definitionen von Klassen und Funktionen. Außerdem wird ein Logger definiert, der ein Ereignisprotokollierungssystem darstellt. Durch den Logger wird festgelegt, in welcher Form Informationen während der Datenabfrage in der Konsole der ausführenden Software (bspw. Spyder, PyCharm, usw.) angezeigt werden. Durch einen Filter lässt sich einstellen, welche Ebene an Protokollsätzen angezeigt werden soll (.setLevel). Die Meldungen, die dann während des Ausführens des Skriptes angezeigt werden, können aus Pythonmodulen stammen oder selbst definiert werden. Ein Beispiel hierfür ist die Mitteilung, dass auf einen neuen Zyklus gewartet wird:

```
logger.debug('Waiting for new cycle.')
```

Neben dem Logger werden in daq_arburg.py auch die für die Abfrage benötigten Klassen definiert. Die drei wesentlichen Klassen sind hierbei signal_struct, NewCycleWatcher und NodeData:

- signal_struct
  Diese Klasse legt fest, dass die abzufragenden Parameter eine node_id besitzen und jede node_id standardmäßig ein Signal beinhaltet. Bei der Abfrage von Messgrafiken muss deshalb darauf geachtet werden, dass dieser Wert auf die Anzahl der beinhalteten Messreihen angepasst wird (siehe Ermittlung der Node-IDs der abzufragenden Prozessgrößen).
- NewCycleWatcher
  Diese Klasse legt fest, dass der Wert des zur Erkennung eines neuen Zyklusses beobachteten Parameters beim Zykluswechsel von old_cycle_value auf new_cycle_value wechselt. Wie dieser Parameter und seine Werte bzw. Zustände festgelegt werden, wird im Abschnitt Arburg_XXX_config.py beschrieben.
- NodeData
  Hierbei handelt es sich um eine Klasse, die alle relevanten Daten aufnehmen kann, die über eine Node-ID zu einem bestimmten Zeitpunkt ausgelesen werden.

Für die Datenabfrage und -archivierung sind einige Funktionen notwendig, die ebenfalls in daq_arburg.py hinterlegt sind und von opc_daq_main.py ausgeführt werden.

- time_to_name
  Diese Funktion erzeugt einen Dateinamen, der sich aus einem Präfix und der aktuellen Zeit zusammensetzt. Während das Präfix in der Datei Arburg_XXX_config.py angepasst werden kann, wird die Zeit in einer definierten Schreibweise angehängt. Hierbei kann in Arburg_XXX_config.py die kleinste zu berücksichtigende Einheit ausgewählt werden. So lässt sich bspw. einstellen, dass alle Daten, die während eines Tages aufgezeichnet werden in eine Datei gespeichert werden. Als kleinste Einheit stehen hierbei Jahre („y“), Monate („mon“), Tage („d“), Stunden („h“), Minuten („m“) und Sekunden („s“) zur Auswahl. Der aus der aktuellen Zeit erzeugte Namenszusatz hat genau diese Reihenfolge und kann bspw. so aussehen: 20221208 (kleinste Einheit: Tag).
- create_nodes
  Diese Funktion erstellt ein Dictionary, das alle in der Arburg_XXX_config.py definierten Parameter beinhaltet.
- save_to_hdf
  Diese Funktion speichert das während der Datenabfrage erstellte DataFrame in eine .h5 Datei mit dem durch time_to_name definierten Namen.
- slugify
  Diese Funktion dient der Konvertierung von nicht gewollten Zeichen in den Daten. Auf diese Weise werden Leerzeichen und wiederholte Bindestriche in einzelne Bindestriche sowie sämtliche Großbuchstaben in Kleinbuchstaben konvertiert. Außerdem

werden Zeichen, die nicht alphanumerisch, Unterstriche oder Bindestriche sind, sowie vorgestellte und nachgestellte Leerzeichen, Bindestriche und Unterstriche entfernt.

Arburg_XXX_config.py

Bei diesem Skript handelt es sich um den maschinenspezifischen Teil, der für jede der in der Produktion eingesetzten Arburg Maschine und jeden Spritzgießprozess separat eingerichtet werden kann. Die drei X im Dateinamen sollen dies verdeutlichen und können bspw. durch eine Maschinenkennung (bspw. Arburg_470S_config.py) oder Prozessbezeichnung (bspw. Arburg_Bauteil1_config.py) ersetzt werden.

Damit die Prozessdaten während des laufenden Spritzgießprozesses über OPC-UA abgefragt werden können, muss die Maschine über ein LAN-Kabel in das entsprechende Netzwerk eingebunden sein und eine eigene IP-Adresse besitzen. Zusätzlich kann für jede Maschine ein Username und Passwort eingerichtet werden. Über diese Informationen und die IP-Adresse wird die Verbindung zur Maschine aufgebaut. Hierzu muss die resultierende OPC-Adresse als CLIENT_ADDRESS in folgendem Format eingetragen werden:

CLIENT_ADDRESS = ‘opc.tcp://Username:Passwort@IP-Adresse:4880/Arburg/’

In den folgenden Zeilen des Skripts kann das bereits beschriebene Namenspräfix (NAME_OF_MEASUREMENT) und das Zeitintervall, in dem nach einem neuen Zyklus gesucht werden soll (SLEEP_TIME) definiert werden. Das Zeitintervall wird hierbei in Sekunden angeben.

Im Dictionary SIGNALS müssen sämtliche Prozessgrößen aufgelistet werden, die bei der Datenabfrage berücksichtigt werden sollen. Hierzu muss jeweils eine Bezeichnung des jeweiligen Parameters und seine Node-ID in folgendem Format angegeben werden:

‘Bezeichnung‘: signal_struct(‘ns=2;i=Node-ID‘)

Entscheidend ist hierbei, dass einer der Zyklenzähler der Maschine (Gesamtzyklenzähler, Auftragszyklenzähler, etc.) abgefragt werden muss, da dieser für die Zuordnung der Prozessdaten benötigt wird. Die Bezeichnung des berücksichtigten Zyklenzählers muss hierbei cycle_counter sein, da er mit dieser Bezeichnung von einigen Funktionen des Skripts verwendet wird.

Da die Messgrafiken bis zu vier Prozessgrößenverläufe beinhalten können, muss bei diesen Grafiken zusätzlich zur Node-ID auch die Anzahl der Signale (num_signals=1-4) angegeben werden:

‘Bezeichnung Messgrafik‘: signal_struct(‘ns=2;i=Node-ID‘, num_signals=4)

Das zweite Dictionary (NEW_CYCLE_SIGNAL) beinhaltet das Signal, was zur Erkennung eines neuen Zyklusses bewacht werden soll. Analog zum Zyklenzähler ist auch bei diesem Parameter wichtig, dass die Bezeichnung unverändert bleibt (‘new_cycle_signal‘). Durch umfangreiche Versuche hat sich gezeigt, dass die Erfassung der Prozessdaten möglichst nah am Zyklusende geschieht, aber noch bevor der nächste Zyklus startet, da sonst einige relevante Prozessgrößen von der Maschinensteuerung überschrieben werden. Deshalb hat es sich bewährt, die Werkzeugbewegung als Überwachungssignal für einen neuen Zyklus zu wählen. Das Signal der Werkzeugbewegung, wechselt seinen Wert von 0 auf 1, während die Werkzeugbewegung ausgeführt wird (vgl. Abbildung 3). Demnach resultieren während eines Zyklusses vier mögliche Abfragepunkte (1-4 in Abbildung 3). Da die Datenabfrage möglichst am Zyklusende geschehen soll, sind die Punkte 3 und 4 gut geeignet, wobei zwischen Punkt 3 und dem Beginn des neuen Zyklus eine größere Zeitspanne für die Datenabfrage zur Verfügung steht. Damit dieser Punkt die Datenabfrage auslöst, muss demnach OLD_CYCLE_VALUE=0 und NEW_CYCLE_VALUE=1 gewählt werden. Natürlich können auch andere Signale gewählt werden, auch solche, die nicht ausschließlich Werte von 0 und 1 annehmen können.

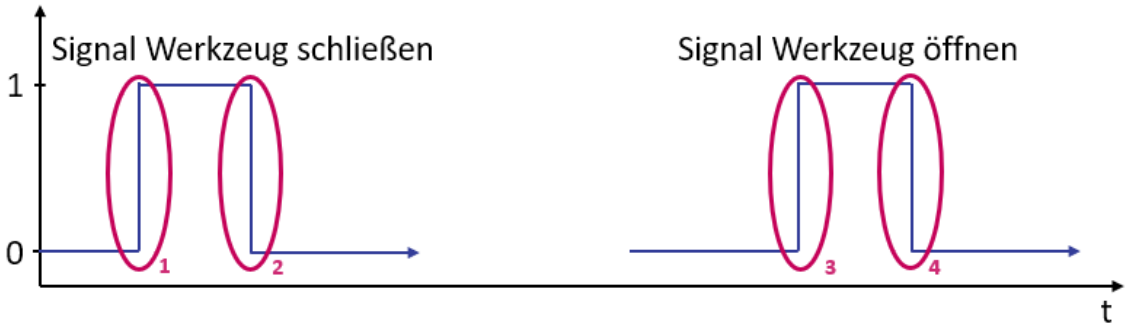

Abbildung 3: Verlauf des Signals der Werkzeugbewegung während eines Spritzgießzyklusses

Neben den beschriebenen Einstellungen zum Verbindungsaufbau, zur Definition der abzufragenden Prozessgrößen und zum Signal, das die Datenabfrage auslöst, lassen sich zwei weitere Einstellungen in Arburg_XXX_config.py vornehmen. Zum einen kann der Wert von USE_FILENAME_CYCLE_PREFIX auf True geändert werden, damit für jeden neuen Zyklus eine eigene Datei erzeugt wird und zum anderen kann die im Abschnitt daq-arburg.py erläuterte kleinste Einheit für die an das Namenspräfix angehängten Zeit definiert werden. Hierzu muss die kleinste zu berücksichtigende Einheit bei NEW_FILE_TIMER angegeben werden (‘y‘, ‘mon‘, ‘d‘, ‘h‘, ‘m‘ und ‘s‘).

opc_daq_main.py

Bei diesem Teil des Skripts handelt es sich um die Datei, die für die Datenerfassung ausgeführt werden muss. Sie greift auf die Funktionen und Dictionaries der anderen beiden Dateien

zurück. Wenn mehrere Arburg_XXX_config.py Dateien für verschiedene Maschinen und/oder Prozesse erstellt wurden, muss hier nun die korrekte Datei importiert werden:

```
import Arburg_XXX_config as cfg
```

Im laufenden Betrieb überwacht das Skript das in Arburg_XXX_config.py definierte Signal, das die Datenabfrage auslöst. Sobald dieses Signal den Wert für einen neuen Zyklus annimmt, werden die Werte sämtlicher im Dictionary SIGNALS hinterlegten Node-IDs über OPC-UA abgefragt und in der .h5 Datei gespeichert. Hierbei werden die jeweiligen Werte eines Zyklusses dessen Zyklusnummer zugeordnet. Der Ablauf wird dabei kontinuierlich über Loggerhinweise in der Konsole angezeigt. Standardmäßig sind folgende Hinweise festgelegt, diese können jedoch auch individuell angepasst werden:

```
logger.debug('Waiting for new cycle.')

logger.info('New cycle found. Reading data from Machine now.')

logger.debug(f'Getting value of node: {name}:{node}')
```

Kurzanleitung zur Verwendung des Software-Tools

Nach der ausführlichen Beschreibung der einzelnen Funktionen der drei Dateien und wie diese angepasst werden können, soll nun die folgende stichpunktartige Anleitung die nötigen Anpassungen der Dateien und Ablauf der Datenerfassung zusammenfassen:

1. Festlegen, welche Prozessgrößen als skalare Werte abgefragt werden sollen
2. Festlegen, welche Prozessgrößen als zeitlicher Verlauf abgefragt werden sollen und konfigurieren der Mess- und Überwachungsgrafiken (ggf. Aufteilen der Zykluszeit auf mehrere Grafiken, zur Erhöhung der Frequenz).
3. Ermittlung der Node-IDs der abzufragenden Prozessgrößen und der konfigurierten Grafiken
4. Anpassung von Arburg_XXX_config.py
    a. IP-Adresse, Username und Passwort anpassen
    b. Eintragen sämtlicher abzufragenden Prozessgrößen und Grafiken inklusive zugehöriger Node-IDs
    c. Festlegung des Auslösesignals (idealerweise Werkzeugbewegung)
    d. Namen der zu erstellenden .h5-Datei definieren (NAME_OF_MEASUREMENT und NEW_FILE_TIMER)
5. Importieren der richtigen Arburg_XXX_congig.py in opc_daq_main.py
6. Ausführen von opc_daq_main.py bei laufendem Spritzgießprozess
7. Das Skript erfasst nun kontinuierlich nach jedem Zyklus die Prozessdaten und speichert sie in die definierte .h5-Datei

Quellen:

[1] https://hmq-3d.ch/galerie/40/3d-modell-spritzgiessmaschine-arburg.html
zuletzt aufgerufen am 21.02.2023

[2] D. Bruckner, et al.: OPC UA TSN. A new Solution for Industrial Communication. https://www.br-automation.com/downloads_br_productcatalogue/assets/1519253860692-en-original-1.0.pdf
zuletzt aufgerufen am 21.02.2023