pull-Betrieb

Das wird jetzt schon aufwändiger. Aber im Vergleich zu einer realen Praxisanordnung : immer noch Kindergarten !

Natürlich ist eine so große Aufgabe nicht prüfungsrelevant.
Aber die Teile, aus denen sie besteht !

Zuerst ein paar prinzipielle Überlegungen.

Wir wollen Manufakturbetrieb, damit wir den Fertigungsablauf zum Test gut überblicken können. Hier haben sie auch gleich ein Beispiel, wozu Manufakturbetrieb in der Praxis nützlich ist : also Testmodus !

Manufaktur mache ich mit polling, das ist einfacher zu schreiben und hat hier keine Nachteile.

Gehen wir so vor, wie in der Aufgabe vorgeschlagen.

Zunächst ein Kernablauf in serieller Kopplung (Lösung aus Paket 20) :

```
from twinLib2021 import *
sps = PlcModule()
los = input("zum Start ENTER drücken")
sps.StartBand()

while True :
   sps.WriteOPCTag('module1','Order',1)
   sps.WriteOPCTag('module1','Start',1)
   ack = sps.ReadOPCTag('module1','Acknowledge')
   while ack == 0:     #warte auf handshakequittung
      ack = sps.ReadOPCTag('module1','Acknowledge')
   sps.WriteOPCTag('module1','Start',0)
   busy = sps.ReadOPCTag('module1','Busy')
   while busy == 1:    #warte, bis modul fertig
      busy = sps.ReadOPCTag('module1','Busy')

   #-----------------------------------------------

   sps.WriteOPCTag('module2','Order',1)
   sps.WriteOPCTag('module2','Start',1)
   ack = sps.ReadOPCTag('module2','Acknowledge')
   while ack == 0:     #warte auf handshakequittung
      ack = sps.ReadOPCTag('module2','Acknowledge')
   sps.WriteOPCTag('module2','Start',0)
   busy = sps.ReadOPCTag('module2','Busy')
   while busy == 1:    #warte, bis modul fertig
      busy = sps.ReadOPCTag('module2','Busy')

   #-----------------------------------------------

   sps.WriteOPCTag('module3','Order',1)
   sps.WriteOPCTag('module3','Start',1)
   ack = sps.ReadOPCTag('module3','Acknowledge')
   while ack == 0:     #warte auf handshakequittung
      ack = sps.ReadOPCTag('module3','Acknowledge')
   sps.WriteOPCTag('module3','Start',0)
   busy = sps.ReadOPCTag('module3','Busy')
   while busy == 1:    #warte, bis modul fertig
      busy = sps.ReadOPCTag('module3','Busy')
```

Das erweitern wir jetzt um das RFID-Handling.
Nach Modul 1 muß die Produkttaufe stattfinden. Dazu muß das Bauteil unter der RFID-Antenne fixiert werden. Sie finden die Orders zur Bedienung der Stopper in der Gesamtbeschreibung :

https://portal.ts-muenchen.de/Portaldateien/Modellfabrik/Gesamtbeschreibung_2021.pdf

Wenn das Bauteil Modul 1 verlassen hat (busy=0), wird es mit dem RFID-Stopper angehalten :

```
sps.WriteOPCTag('module1','Order',4)                  #Auftrag : Stopper schließen
sps.WriteOPCTag('module1','Start',1)                  #los !
ack = sps.ReadOPCTag('module1','Acknowledge')
while ack == 0:
   ack = sps.ReadOPCTag('module1','Acknowledge') #Handshake
sps.WriteOPCTag('module1','Start',0)
busy = sps.ReadOPCTag('module1','Busy')
while busy == 1:                                      #warte, bis Aktion abgeschlossen
   busy = sps.ReadOPCTag('module1','Busy')
```

Ob der RFID-Tag schon im Erfassungsbereich der Antenne ist und geschrieben werden kann, prüft man durch Auslesen der UID des Tags :

```
uid = sps.ReadRFIDUID(0)                              #Antenne 0 ist an Modul 1
while uid == 0:                                       #polling auf den UID
   uid = sps.ReadRFIDUID(0)
```

Wenn die UID erkannt wird (=0), kann der Schreibvorgang erfolgen. Hier wird die Variable id auf den RFID geschrieben :

```
sps.WriteRFIDTag(0,id)                          #den Wert von id schreiben
```

Und dann wird das Bauteil freigegeben :

```
sps.WriteOPCTag('module1','Order',3)
sps.WriteOPCTag('module1','Start',1)
ack = sps.ReadOPCTag('module1','Acknowledge')
while ack == 0:
   ack = sps.ReadOPCTag('module1','Acknowledge')
sps.WriteOPCTag('module1','Start',0)
busy = sps.ReadOPCTag('module1','Busy')
while busy == 1:
   busy = sps.ReadOPCTag('module1','Busy')
```

Nun läuft das “getaufte” Produkt weiter, und kommt irgendwann an Modul 2 an. Hier wird nun die im RFID gespeicherte ID bei jedem einlaufenden Produkt ausgelesen, und damit der Fertigungsauftrag ermittelt. Daher der Ausdruck : “Das Produkt steuert die Fertigung”.

```
sps.WriteOPCTag('module2','Order',4)            #rfid-stopper zu
sps.WriteOPCTag('module2','Start',1)
ack = sps.ReadOPCTag('module2','Acknowledge')
while ack == 0:                                  #warte auf handshake
   ack = sps.ReadOPCTag('module2','Acknowledge')
sps.WriteOPCTag('module2','Start',module)
busy = sps.ReadOPCTag('module2','Busy')
while busy == 1:                                #warte, bis Stopper fertig
   busy = sps.ReadOPCTag('module2','Busy')
```

Wenn der Stopper draußen ist, warten wir auf die Erkennung des Tags und lesen dann die ID des Produkts (jetzt Antenne 1) :

```
uid = sps.ReadRFIDUID(1)          #warte auf rfid (polling auf UID)
while uid == 0:
   uid = sps.ReadRFIDUID(1)
ID=sps.ReadRFIDTag(1)             #ID des Produkts lesen
```

Danach geht das Tor wieder auf, genau wie bei Modul 1.
Nachdem, wie gleich beschrieben, aus der ID durch vertikale Kommunikation der Fertigungsauftrag eingeholt wurde, kann Modul 2 gestartet werden.

Vorher kommt aber noch der spannendere Teil :
Mit dieser ID schicken wir einen REST-Request in die ERP-Ebene.
In unserem Fall von ihnen daheim über das Internet auf den Datenbankserver brunello in der Schule.

Das kann nur funktionieren, wenn sie ihr Python wie in dem Dokument im Portal beschrieben, um die http-Klassen erweitert haben.

Der Befehl selbst ist dann nicht weiter aufregend :

```
auftrag = erp.ReadERPPart('REIDOL', ID, 'middle')
```

Mit dem in “Echtzeit” aus der Datenbank eingeholten Wert für den Fertigungsauftrag wird nun Modul 2 angesteuert :

```
sps.WriteOPCTag('module2','Order',auftrag)
sps.WriteOPCTag('module2','Start',1)

ack = sps.ReadOPCTag('module2','Acknowledge')
while ack == 0:
   ack = sps.ReadOPCTag('module2','Acknowledge')

sps.WriteOPCTag('module2','Start',0)

busy = sps.ReadOPCTag('module2','Busy')
while busy == 1:
   busy = sps.ReadOPCTag('module2','Busy')
```

Jetzt kommt eigentlich nichts neues mehr.

An Modul 3 den Stopper steuern, RFID lesen, Stopper auf und Modul starten :

```
 sps.WriteOPCTag('module3','Order',5)                #rfid-stopper zu
 sps.WriteOPCTag('module3','Start',1)
 ack = sps.ReadOPCTag('module3','Acknowledge')
 while ack == 0:
    ack = sps.ReadOPCTag('module3','Acknowledge')
 sps.WriteOPCTag('module3','Start',module)
 busy = sps.ReadOPCTag('module3','Busy')
 while busy == 1:                                    #warte, bis modul fertig
    busy = sps.ReadOPCTag('module3','Busy')
 #.........................
 uid = sps.ReadRFIDUID(2)                            #warte auf rfid (UID)
 while uid == 0:
    uid = sps.ReadRFIDUID(2)
 value=sps.ReadRFIDTag(2)
 #.........................
 sps.WriteOPCTag('module3','Order',4)                #rfid-stopper auf
 sps.WriteOPCTag('module3','Start',1)
 ack = sps.ReadOPCTag('module3','Acknowledge')
 while ack == 0:
    ack = sps.ReadOPCTag('module2','Acknowledge')
 sps.WriteOPCTag('module3','Start',0)
 busy = sps.ReadOPCTag('module3','Busy')
 while busy == 1:                                    #warte, bis Stopper fertig
    busy = sps.ReadOPCTag('module3','Busy')
#------------------------------
auftrag = erp.ReadERPPart('REIDOL', ID3, 'top')
sps.WriteOPCTag('module3','Order',auftrag)
sps.WriteOPCTag('module3','Start',1)
ack = sps.ReadOPCTag('module3','Acknowledge')
while ack == 0:    #warte auf handshakequittung
   ack = sps.ReadOPCTag('module3','Acknowledge')
sps.WriteOPCTag('module3','Start',0)
busy = sps.ReadOPCTag('module3','Busy')
while busy == 1:   #warte, bis modul fertig
   busy = sps.ReadOPCTag('module3','Busy')
```

Das komplette Programm finden sie als zip-file im Portal