IT-Projekt

Fertigung

Mai 2020

Inhaltsverzeichnis

Ziele

Eine einfache Befüllmaschine für Milch, Säfte, Wein oder andere drucklose Flüssigkeiten in Papppackungen soll automatisiert werden. Hierzu gehört die erforderliche Steuerung zur Realisierung des Befüllvorgangs beginnend mit der Zufuhr eines leeren Behälters bis zum Auswerfen des gefüllten und verschlossenen Behälters. Parallel dazu soll die Temperatur der Flüssigkeit bzw. Des Arbeitsraumes geregelt werden.

Da bei einer realen Maschine zur Befüllung die Kosten für Milch für ein IT-Projekt zu hoch sind und da man sich mit Milch bekleckern könnte, wird die Befüllmaschine durch eine vorhandene Software simuliert.

Zunächst ist die geplante, grundlegende Arbeitsweise der Befüllmaschine zu ermitteln. Daraus folgt eine Definition der erforderlichen Arbeitsschritte und es kann ein Pflichtenheft über die einzelnen Funktionen der automatisierten Steuerung erstellt werden.

Soweit durch die bestehende Simulation abgedeckt, ist dann ein geeignetes Programm in C/C++ zu erstellen.

Die erreichte Funktionalität ist zu dokumentieren. Ziel ist eine Beschreibung des Programms und seiner Funktionen. Hierzu sind auch Zeitverläufe von ausgewählten Befüllvorgängen oder Teilvorgängen aufzuzeichnen.

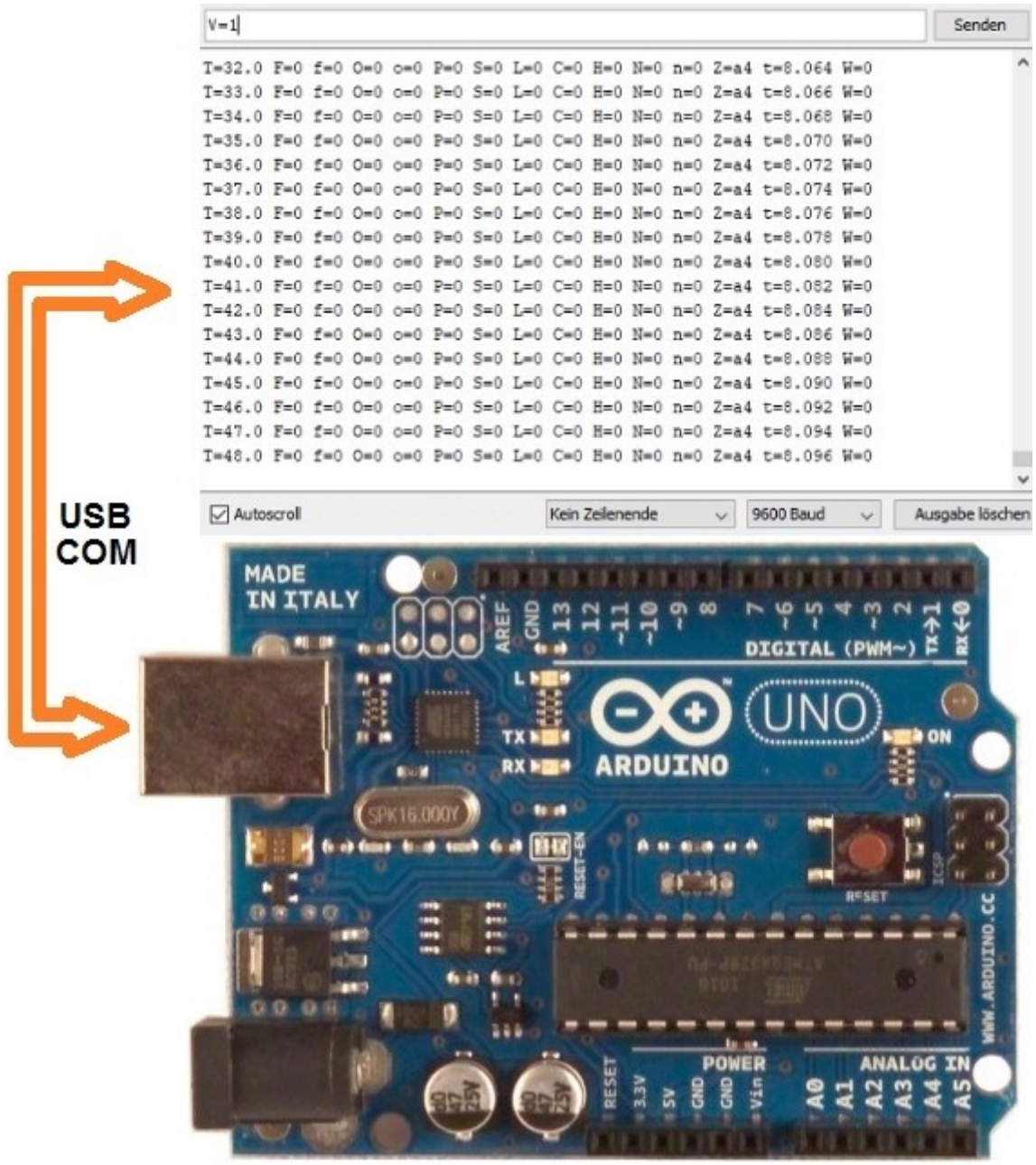
Die Lösung dieser Grundaufgabe soll passend erweitert werden. Hier können eigene Ideen verwirklicht werden. Ein Mitzählen der produzierten Packungen wäre ein sehr simples Beispiel. Etwas aufwendiger wäre eine Anzeige über LEDs oder ein Display.

Das IT-Projekt kann auch durch Teile ergänzt werden, die nicht unmittelbar in die Programmierung einfließen. Eine Software-Dokumentation mittels Doxygen oder ein Versionskontrollsystem auf der Basis von SVN sind solche Ergänzungen.

Optional kann eine Bedienung von einem graphischen PC-Programm erstellt werden – Stichwort SCADA. Mit wxWidgets ist so etwas vergleichsweise einfach realisierbar. Eine Alternative wäre eine Bedienung über eine App auf dem Smartphone.

Simulation der Fabrikation

Die vorbereitete Simulation der Fertigung ohne eigenständige Steuerung befindet sich in einem Programm genannt Fertigung auf einem Arduino UNO Board. Über die USB-Kommunikations-schnittstelle des Boards wird eine serielle Kommunikation (COM-Schnittstelle) realisiert. Mit dieser Datenverbindung kann der aktuelle Zustand der Fabrikation in einfacher Form angezeigt werden und es können (in der Test-Betriebsart) über manuelle Steuerbefehle Eingriffe in die Fabrikation erfolgen.

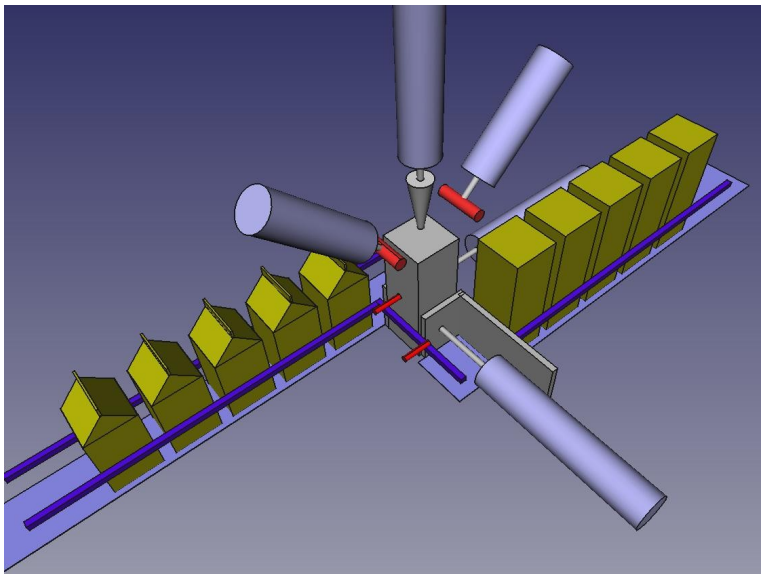


In der Grundeinstellung des Monitors sollte automatisches Scrolling eingeschaltet sein, die Baudrate auf 9600 stehen und kein Zeilenende hinzugefügt werden.

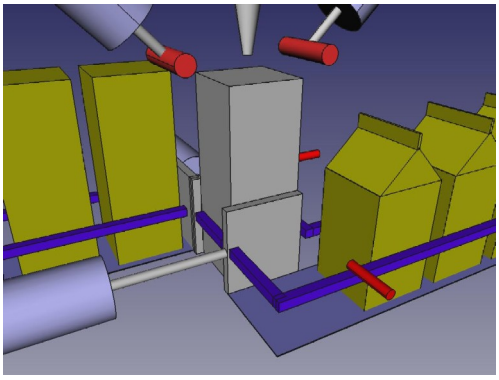
Statt des integrierten seriellen Monitors der Arduino IDE kann auch jedes andere COM-Port Terminal verwendet werden. Einige derartige Terminals erlauben die empfangenen Daten in eine Log-Datei zu kopieren (z.B. https://sourceforge.net/projects/realterm/). Diese Log-Dateien können später für weitere Aufgaben wie etwa zur graphischen Darstellung mittels gnuplot ausgewertet oder dokumentiert werden.

Beschreibung der Fertigungsstation

Die Fertigungsstation dient zur Befüllung von Behältern für drucklose Flüssigkeiten wie Milch oder Säfte. Die Simulation stellt einen vereinfachten Ausschnitt einer solchen Fertigung dar.

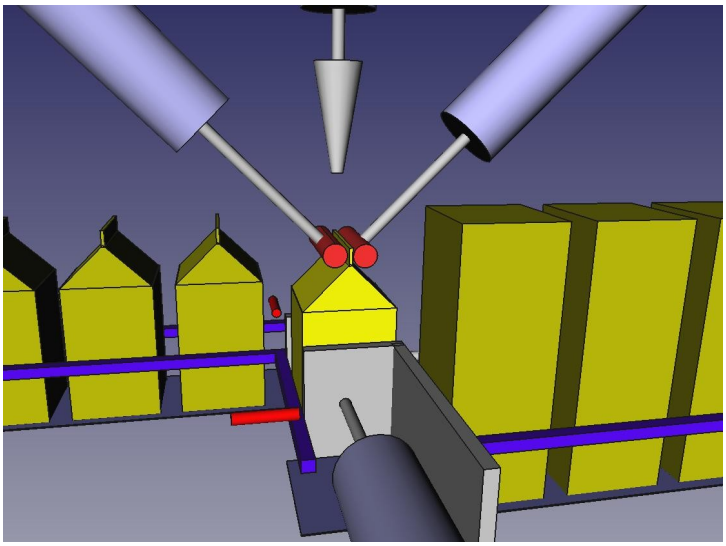


Auf einem Transportband kommen von vorne rechts die zu befüllenden, oben offenen Behälter an. Ein zweites Transportband dahinter führt die gefüllten und verschlossenen Behälter nach links ab. Die Bewegung dieser Bänder ist zu automatisieren.
Die beiden Arbeitsschritte zur Befüllung und zum Verschließen finden zwischen den beiden Transportbändern statt.



Mit einer Schubvorrichtung wird ein Behälter unter die Befülldüse geschoben. Ob ein Behälter anwesend ist, detektiert ein Sensor am Ende des Zufuhrbandes. Als Antrieb der Schubvorrichtung dient ein pneumatischer Zylinder. Zugleich verhindert die Schubvorrichtung die Zufuhr eines weiteren Behälters. Ein beweglicher Schieber auf der gegenüber liegenden Seite verhindert zunächst, daß der Behälter bis auf das Abfuhrband geschoben wird. Dieser Schieber wird ebenfalls von einem pneumatischen Zylinder als Antrieb bewegt. Die Bewegung beider Zylinder ist zu automatisieren.

Während der Befüllung muss der Behälter durch Einklemmen festgehalten werden. Die Befüllung selbst erfolgt nach dem Absenken der Befülldüse durch Öffnen eines Ventils. Die Regulierung der Füllmenge an Flüssigkeit wird über die Füllstandshöhe gemessen. Hierzu ist in passender Höhe neben dem Behälter ein kapazitiver Sensor angebracht, der bei Erreichen der erforderlichen Menge schaltet. Man beachte, daß nur eine sehr geringe Mehrmenge zulässig ist - entsprechend 0.5 Sekunden Einlaufdauer. Gegen Ende der Befüllung müssen das Ventil geschlossen und die Befülldüse wieder angehoben werden. Auch dieser Vorgang ist zu automatisieren.



Wenn die erforderliche Menge eingefüllt ist, soll der Behälter verschlossen werden. Dies wird hier (vereinfacht) durch zwei Stempel erreicht, die den Behälter oben zusammen drücken und mittels einer Beheizung verschweißen. Die Knickstellen sind durch eine Behandlung bei der Herstellung der Verpackung vorgegeben (Sollknickstellen). Die Bewegung der beiden Stempel durch zwei Zylinder und der Schweißvorgang einschließlich seiner Dauer von 3 Sekunden sollen automatisiert werden. Beide Zylinder und beide Heizungen werden gemeinsam gesteuert.

Nach Befüllen und Verschließen wird der Behälter auf das Abtransportband weitergeschoben. Hierzu ist zunächst die Schubvorrichtung zurückzuziehen, damit der hintere stoppende Schieber ohne Verklemmung des Behälters zurückgezogen werden kann. Erst danach wird von der Schubvorrichtung der Behälter auf das Abtransportband weitergeschoben. Rückseitig am Abtransportband ist ein Sensor zur Detektierung des Behälters angebracht. Auch dieser Vorgang ist zu automatisieren.

Schließlich geht die Anlage wieder in ihre Grundposition und der ganze Vorgang kann mit einem neuen Behälter fortgesetzt werden.

Ein Video hierzu wurde im IT-Projekt Wintersemester 2014/15 erstellt. Herzlichen Dank an Julian Dora, Alexander Müller und Robin Sprave.

In Grenzen werden das zeitliche Verhalten der Abfüllstation und auch einige Fehlersituationen bei der Steuerung in der Simulation nachgebildet. So wird beispielsweise beim Befüllen ohne einen Behälter am Befüllort eine Warnung angezeigt. Auch ist es möglich die Anlage zu verklemmen. In solchen Fällen ist ein komplettes Rücksetzen der Fabrikation als Ersatz für einen manuellen Eingriff erforderlich.

Um die Aufgabe nicht zu schwierig zu gestalten, ist die Anlage vergleichsweise langsam.

Parallel zum Abfüllvorgang soll die Temperatur der Flüssigkeit und des Arbeitsbereichs durch das Programm auf möglichst 8 °C geregelt werden. Hierfür ist eine Kühlung vorgesehen, die nur Ein und Aus geschaltet werden kann.

Technologische Größen

Folgende technologischen Größen werden von der Simulation bereitgestellt.

Größe	**Sensor oder Aktuator**	**Bedeutung**
Zeit	(S)	Simulationszeit in Sekunden
Zufuhrantrieb	A	Bandbewegung ein-/ausschalten
Abfuhrantrieb	A	Bandbewegung ein-/ausschalten
Behälterdetektor am Zufuhrband	S	Näherungsschalter oder Schalthebel
Behälterdetektor am Abfuhrband	S	Näherungsschalter oder Schalthebel
Füllhöhensensor	S	Füllhöhe, binärer Grenzwertmelder
Zylinderpositionen	(S)	kombinierte Sensoren an den Zylindern
Zufuhrzylinder	A	pneumatisch ein- oder ausfahrend
Sperrenzylinder	A	pneumatisch ein- oder ausfahrend
Fülldüsenabsenker	A	pneumatisch ein- oder ausfahrend
Verschließer rechts und links	A	gemeinsam ein- oder ausfahrend
Heizelemente	A	gemeinsam ein-/ausschaltend
Ventil der Einfülldüse	A	Flüssigkeitszulauf Auf/Zu
Temperatur	(S)	in °C
Kühlung	A	ein-/ausschaltend

Diese Größen sind als Maschinendaten fest vorgegeben und werden über Ein- und Ausgänge der Arduino Boards ausgetauscht. Die verwendeten IO-Ports findet man im Programm. Für einige wenige Daten, durch Klammerung gekennzeichnet, wird eine I^2C-Verbindung zwischen den Boards eingesetzt.

Betriebsmodi

Die Anlage kann in zwei Betriebsweisen arbeiten. In der Testeinstellung (Handbetrieb, manual mode, M=1) werden die Aktuatoren durch eine Kommunikation über die USB-Schnittstelle durch einfache Kommandos am PC eingestellt. Hiermit ist ein experimentelles Betreiben der Anlage möglich. Insbesondere bei ersten Versuchen zum Kennenlernen der Anlage ist diese Einstellung sehr geeignet.

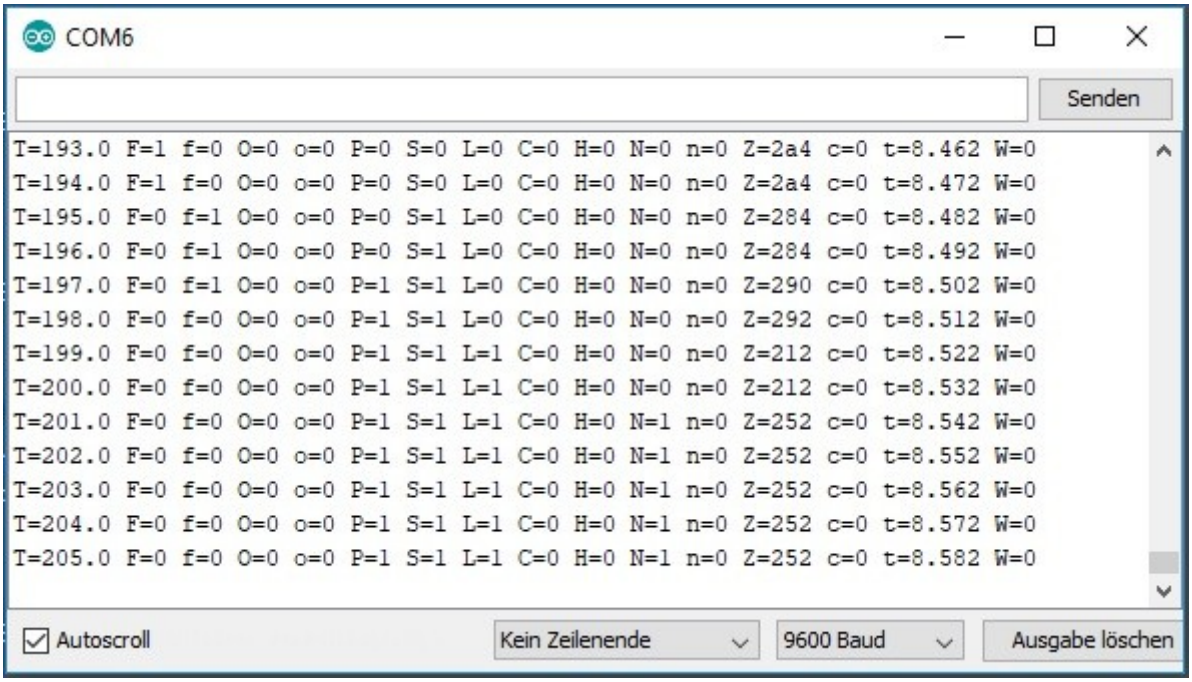
In der eigentlichen Betriebseinstellung (zugleich Grundeinstellung, normal mode, Kommando M=0) folgen die Aktuatoren den Eingängen des Arduino-Boards. Diese werden dann mit den entsprechenden Ausgängen des Arduino Controllers als PLC verbunden.

Die Sensoren werden in jedem Fall an die entsprechenden Ausgänge des Arduino-Boards geleitet und werden gemeinsam mit den Werten der Aktuatoren zusätzlich immer an der USB-Schnittstelle der Anlagen-Simulation ausgegeben.

Zu beachten ist, dass die jeweilig zusammengehörenden Ein- und Ausgänge immer miteinander verbunden sein sollten. Offene Eingänge von Arduino-Boards können zufällige oder falsche Werte liefern.

Bedienung

Von der seriellen Kommunikation der Simulation der Anlage wird stetig eine durch Leerzeichen getrennte Folge von aktuellen Werten oder Zuständen gesendet. Die Kurzbezeichnungen bestehend aus einem Buchstaben und einem Gleichheitszeichen können mittels eines Verbose Flags ein- oder ausgeschaltet werden.



Angezeigt werden in dieser Reihenfolge

Name	Bedeutung
T	Zeit in Sekunden
F	Zufuhrband
f	Sensor am Ende des Zufuhrbandes
O	Abfuhrband
o	Sensor am Anfang des Abfuhrbandes
P	Schiebezylinder
S	Stopperzylinder
L	Befüllzylinder
C	Verschließzylinder
H	Beheizung
N	Flüssigkeitsventil
n	Füllstandsensor

Z	kombinierte Sensoren an den Zylindern
c	Kühlung
t	Temperatur
W	Warnung oder Fehler

Die Bedienung erfolgt durch kurze Kommandos, die an die Kommunikation übergeben werden und welche zugleich die zugehörigen Werte zurückliefern.

Kommando	**Bedeutung**
F=x	Zufuhrband ein/aus (**F**eed belt)
O=x	Abfuhrband ein/aus (**O**utlet belt)
P=x	Schieberzylinder aus-/einfahren (**P**usher)
S=x	**S**topperzylinder aus-/einfahren
L=x	Befüllzylinder aus-/einfahren (**L**iquid filling cylinder)
C=x	Verschließzylinder aus-/einfahren (**C**losing cylinder)
H=x	Beheizung (**H**eating) des Verschließzylinder ein/aus
N=x	Flüssigkeitsventil öffen/schließen (**N**ozzle valve)
c=x	Kühlung (**C**ooling) ein/aus
R	(Neu-)Initialisierung (**R**eset)
V=x	**V**erbose Flag ein/aus
M=x	Handbetriebsart (**M**anual mode) ein/aus
T?	Simulationszeit abfragen (**T**ime)
t?	Temperatur abfragen (**t**emperature)
Z?	Stellungssensoren der **Z**ylinder abfragen
W?	**W**arnungen abfragen
V?	**V**erbose Flag ermitteln
X?	Betriebsart (e**X**perimental mode) ermitteln

Bei den Einstell-Kommandos ist das zweite Zeichen ein Gleichheitszeichen ('='). Hierbei ist 'x' durch entweder 0 oder 1 zu ersetzen. Als Antwort erhält man den jeweiligen neuen Zustand. In der Regel wird also das Einstell-Kommando wieder zurückgegeben.

Beispiel	Kommando	Antwort	
	H=1	H=1	Beheizung wird eingeschaltet

Bei den Abfrage-Kommandos ist das zweite Zeichen ein Fragezeichen ('?'). Als Antwort erhält man den jeweiligen aktuellen Zustand.

Beispiel	Kommando	Antwort	
	H?	H=0	Beheizung ist ausgeschaltet

Der komplette Zustand wird von der Simulation regelmäßig übertragen. Wahlweise mit oder ohne vorangestellten Zeichen zur besseren Lesbarkeit (verbose flag).

Mit dem Kommando R (ohne eine Antwort) wird die Anlage in einen definierten Anfangszustand versetzt. Dies ist beispielsweise nach Störungen nötig. Die Einstellung für das verbose flag und die manuelle Betriebsart bleiben dabei aber unverändert erhalten.

Bei fehlerhafter Steuerung werden Warnungen in Form von Bits innerhalb einer Zahl erzeugt. Die einzelnen Bits haben folgende Bedeutungen.

Bit (hexadezimal)	**Bedeutung**
0	Keine Fehler
0x0001	Anlage verklemmt
0x0002	Kein Behälter vorhanden
0x0004	Behälter nicht voll
0x0008	Behälter überfüllt
0x0010	Behälter nicht verschlossen
0x0020	Flüssigkeit läuft aus oder über
0x0040	Überflüssiges Heizen
0x0080	Temperatur zu hoch
0x0100	Temperatur zu niedrig

Für die kombinierten Positionssensoren der Zylinder gilt

Bit (hexadezimal)	**Bedeutung**
0x0001	Zufuhrzylinder ausgefahren
0x0002	Zufuhrzylinder halb ausgefahren
0x0004	Zufuhrzylinder eingefahren
0x0010	Stopperzylinder ausgefahren
0x0020	Stopperzylinder eingefahren
0x0040	Befüllzylinder ausgefahren

0x0080	Befüllzylinder eingefahren
0x0100	Verschließzylinder ausgefahren
0x0200	Verschließzylinder eingefahren

Siehe auch Helpfile Fertigung.CHM.

Steuerung der Fertigung

Für Steuerungs- und Regelungsaufgaben verwendet man typischerweise Programme in programmierbaren Steuerungen (SPS, PLC) oder Embedded Controllern.

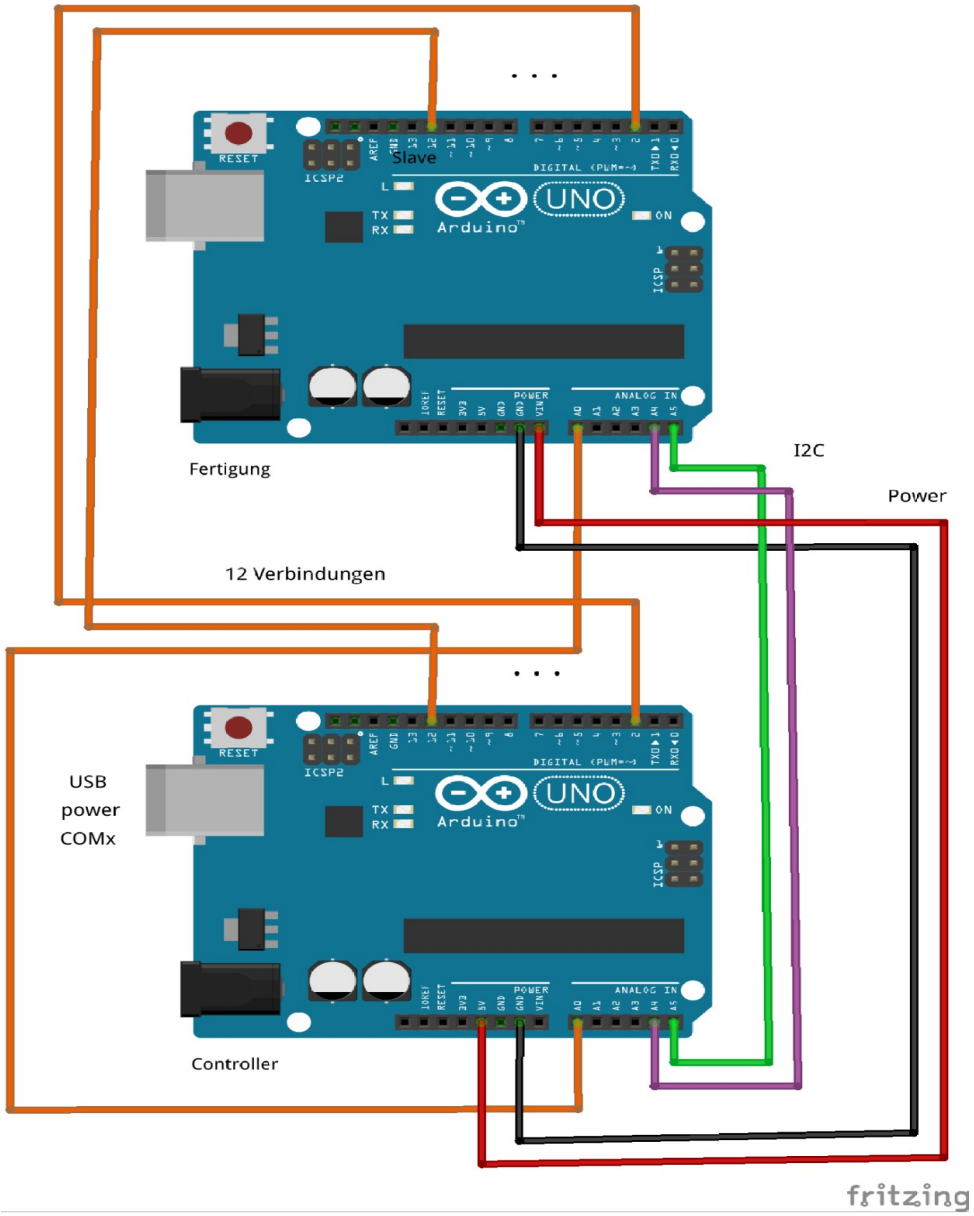
Hier ist für die Steuerung der Fertigung als Starthilfe für den Entwurf ein Programm Controller für ein zweites Arduino UNO Board vorbereitet. Es besitzt bereits eine rudimentäre Bedienung über die USB/COM Schnittstelle und über I^2C mit den gleichen Kommandos wie die Simulation der Anlage.

Einige Vorgänge können wahlweise zeitgesteuert oder über die zusammengefassten Sensoren für die Zylinderpositionen (I^2C) gesteuert werden.

Als eine kleine Hilfe ist bereits eine Abfrage der Zeit (als Sensor), ein Auslesen der Warnungen und eine Umschaltung der Betriebsart und des Verbose-Flags (als Aktuator) programmiert.

Verbindung von Steuerung und Anlage

Als Verbindung zwischen Steuerung und Prozess ist eine Signalübertragung durch Verbindungen zwischen Ein- und Ausgängen und den I2C-Schnittstellen vorhanden.



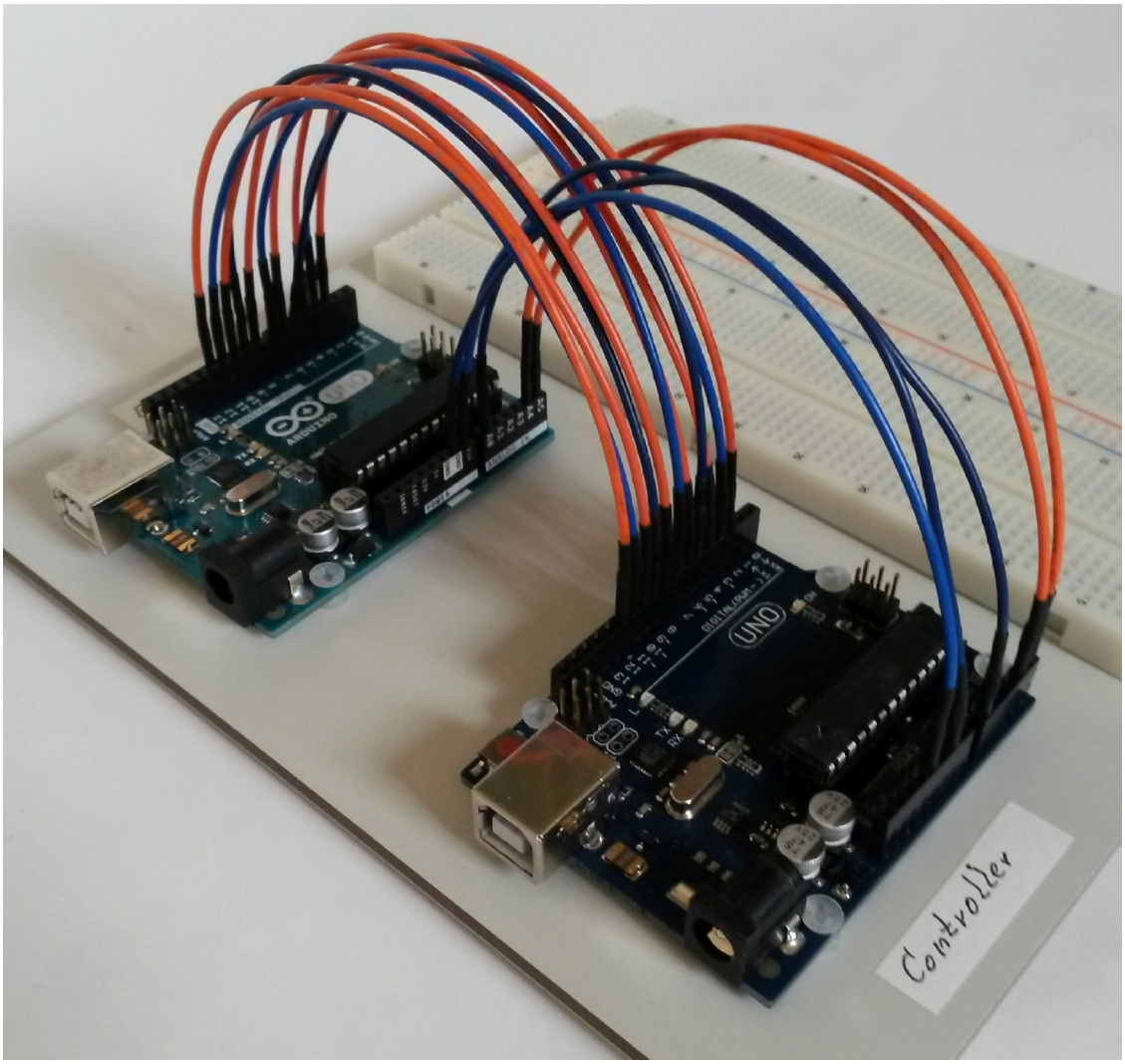
Zwei Verbindungen führen Masse und +5 Volt als Spannungsversorgung. So geschaltet, ist bereits eine Stromversorgung über ein USB-Kabel ausreichend. Es dürfen aber auch beide USB-Anschlüsse von zwei PCs aus genutzt werden. Dies ist hilfreich, wenn die Signale und Daten auf der Seite der Fertigung aufgezeichnet oder visualisiert werden sollen.

Die Kommunikation benutzt die Master-Slave- bzw. Client-Server-Architektur des I²C-Bus. Hierfür sind 2 Verbinungen (und die Masseverbindung) nötig. Die Anlage bzw. deren Simulation

übernimmt dabei die Rolle eines Slave bzw. Servers. Ein Slave wartet auf Anfragen und beantwortet diese. Er wird niemals selbst aktiv. Der Master bzw. Client ist der aktive Teil, also hier der Controller. Er entscheidet wann, wen und was angefragt wird. Ausgetauscht werden hier die gleichen kurzen Kommandos in Form von (lesbarem) Text wie an der USB/COM-Schnittstelle.

Der I²C-Bus zwischen den beiden Boards wird hier ohne die eigentlich erforderlichen Abschlußwiderstände und ohne eine aufwendige Störfehlerbehandlung betrieben. Es kann daher gelegentlich zu Abbrüchen der I²C-Verbindung kommen. Diese sind oft nur durch einen Reset der Arduinos zu beseitigen.

Die physikalischen Verbindungen (EA der SPS) erfordern weitere 12 Drahtverbindungen. Die Einstellungen als Ein- bzw. Ausgänge erfolgt in der Software. Bei den Verbindungen und deren Einstellung ist sehr darauf zu achten, dass niemals zwei Ausgänge miteinander verbunden werden. Ein solcher Kurzschluß könnte zur Zerstörung der Anschlüssen oder der Platine führen.



Anschlussbelegung

Anschluss	Bedeutung
I^2C	Zeit in Sekunden, Warnung oder Fehler etc.
D2	Zufuhrband
D10	Sensor am Ende des Zufuhrbandes
D3	Abfuhrband
D11	Sensor am Anfang des Abfuhrbandes
D4	Schiebezylinder
D5	Stopperzylinder
D6	Befüllzylinder
D8	Verschließzylinder
D9	Beheizung
D7	Flüssigkeitsventil
D12	Füllstandsensor
A0	Kühlung (als digitales EA genutzt)

Die standardmäßig definierte LED auf dem Board am Anschluss D13 wird als blinkendes Arbeitskennzeichen verwendet.

Arbeitsweise der Steuerung

Die reale Abtastzeit im Programm Controller wird durch ein sehr einfaches, nicht preemtives Multitasking realisiert. Dies entspricht in etwa der Arbeitsweise von gewöhnlicher speicherprogrammierbarer Steuerungen, SPSen. Das Multitasking ist bereits voreingestellt für Teilaufgaben mit 10 msec, 100 msec und 1s Abtastzeit.

Der größere Teil der Steuerung kann mit einer Abtastzeit von 100 msec erfolgen. Soweit möglich dürfen weniger zeitkritische Steuerungsvorgänge nur jede Sekunde bearbeitet werden. Nur sehr wenige Teile der Automatisierung benötigen eine Abtastzeit von 10 msec. Es ist besonders wichtig, dass keine Teilaufgabe eine länger Rechenzeit als ihre Abtastzeit benötigt.

Die Verbindung von Steuerung und Prozess erfolgt durch Signalaustausch über die Ein- und Ausgänge der Arduinos und über die Kommunikation über den I^2C-Bus.

Es ist sinnvoll Daten aus der Steuerung in einer Datei abzulegen. In realen Anlagen ist dies oftmals aus rechtlichen und betriebswirtschaftlichen Gründen oder aus Gründen der Qualitätssicherung ohnehin erforderlich. Hier können solche Dateien zur Darstellung von Zeitverläufen etwa mittels Gnuplot verwendet werden.

Das vorhandene Programm Controller ist lediglich ein Dummy. Es enthält keinerlei eigene Automatisierung. Es werden lediglich eingegebene Kommandos von der USB/COM Schnittstelle an eine angeschlossene Anlagensimulation weitergeleitet und die die erhaltenen Antworten werden angezeigt. Hierfür müssen aber die Verbindungen zwischen den Arduino Boards vorhanden sein. Auch auf diesem Weg ist eine experimentelle, manuelle Steuerung der Anlage nach entsprechender Programmierung möglich.

Die Programmierung der erforderlichen Steuerung und Regelung in dem Controller-Board ist die Hauptaufgabe dieses IT-Projektes.

Siehe auch Helpfile Controller.CHM.

Datenübertragung

Alle binären Daten von der Steuerung im Controller zu Heizung in der Simulation des Hauses und umgekehrt werden über die Ein- und Ausgänge der Arduino-Boards abgewickelt.

Beispielsweise wird mittels

```
digitalWrite(nFeedBelt, 1);              // start feed belt
```

das Zulaufband gestartet. Und mit

```
bBottleFull = digitalRead(nBottleFull);  // get filling sensor
```

kann der Sensor für den Füllstand abgefragt werden.

Für alle nicht binären Daten müssen Abfrage-Kommandos über den I²C-Bus gesendet und die Antworten ausgewertet werden. Hier sind in Controller.ino bereits Beispiele vorhanden, z.B.

```
case 0:                             // request time
   strcpy(szCommand, "T?");         // build command
   ++nIndex;                        // goto next nIndex
   break;
```

Die Antworten werden etwas weiter im Programm angenommen, z.B.

```
case 'T':                           // got a fresh dTime value
  dTime = atof(szResponse+2);       // convert response to double
  break;
```

Die Werte für Zeit und Warnungen bzw. Fehler werden in Controller.ino bereits behandelt.

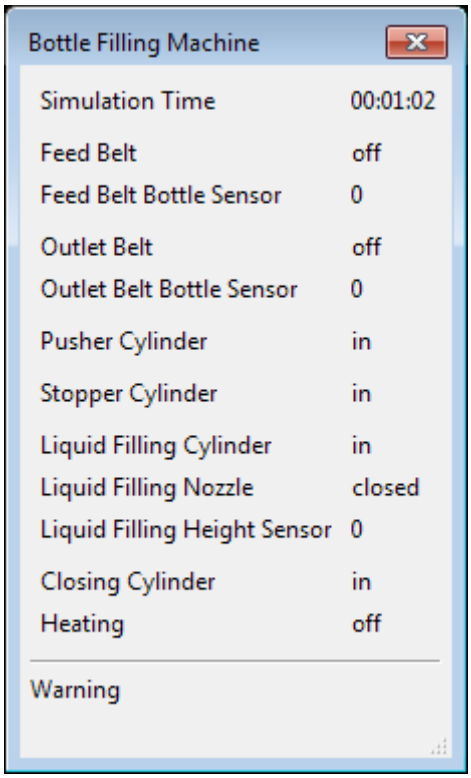
Tools

Für die Bearbeitung der Aufgabenstellung sind schon ein paar Hilfen außer dem Steuerprogramm vorbereitet.

Vgl. Beispiele in Verzeichnissen auf dem Server.

Windows Programm SCADA

Ein Programm SCADA zur Anzeige des aktuellen Zustands der Abfüllstation wäre ein nice to have. Eine denkbare Form, etwa mit wxWidgets programmiert, wäre folgende



Prinzipiell entspräche ein solches Programm einem HMI-Interface, auch SCADA genannt. Es könnte über eine Vielzahl weiterer oder auch ganz anderer Funktionen verfügen – graphische Anzeige als y-t-Diagramm, Bedienung wie Start/Stop, zählen der produzierten Verpackungen und mehr. Der Aufwand ist aber erheblich.

Alternativ wäre auch eine Smartphone App (z.B. mit MIT App Inventor) oder eine Anzeige auf einem kleinen Display denkbar.

Dokumentation

Ein wesentlicher Teil des IT-Projektes ist eine Dokumentation. Diese sollte aus einer funktionalen Beschreibung der Fertigung und der erstellten Software für die Steuerung bestehen.

Die Form der Dokumentation ist freigestellt. Typisch und häufig verwendet wird hierzu Word. Auch Fotografien sind inzwischen als erläuternder Teil in Dokumentationen gängig.

Es ist aber auch möglich (und vielleicht sogar handlicher und leistungsfähiger) die Dokumentation direkt im Quellcode einzupflegen und mit Doxygen zu erstellen. Die Dokumentation der GUI-Bibliothek wxWidgets (einige tausend Seiten) ist beispielsweise hiermit erstellt. Auch die hier vorhandenen Programmdokumentationen Controller.CHM und Fertigung.CHM sind mit Doxygen gemacht.

Doxygen

Doxygen ist eine weitverbreitete Software zur Dokumentation von Software innerhalb ihres Quellcodes. Gesteuert von speziellen Markierungen in den Quelldateien kann sie Dokumentationen in verschiedenen Formaten erstellen. Hier im IT-Projekt ist eine Dokumentation in Form von HTML-Seiten und kompiliertem HTML etwas vorbereitet. Die HTML-Seiten werden mit einem HTML-Compiler in CHM-Dateien umgestzt und komprimiert.

Siehe auch den Quellcode des Controllers, Controller.CHM und Controller.doxy.

SVN

Software soll meist über den Tag hinaus bestehen bleiben und muss reproduzierbar sein. Da Software meist weiterentwickelt wird, sind zur Fehlersuche häufig auch die vorangegangenen Versionen wichtig. Als erstes Backup und zur Versionsverwaltung werden verschiedene Software eingesetzt.

Eine solche, weit verbreitete und moderne Versionsverwaltung ist SVN (Apache Subversion). Die Software des IT-Projektes soll auf einem SVN-Server hinterlegt werden.

Fritzing

Fritzing ist eine Entwurfs-Software für Arduino und andere derartige Boards. Sie kann beispielsweise zur Darstellung von Steckplatine und Schaltplan verwendet werden.

Eine Alternative aber deutlich komplexer ist KiCAD.

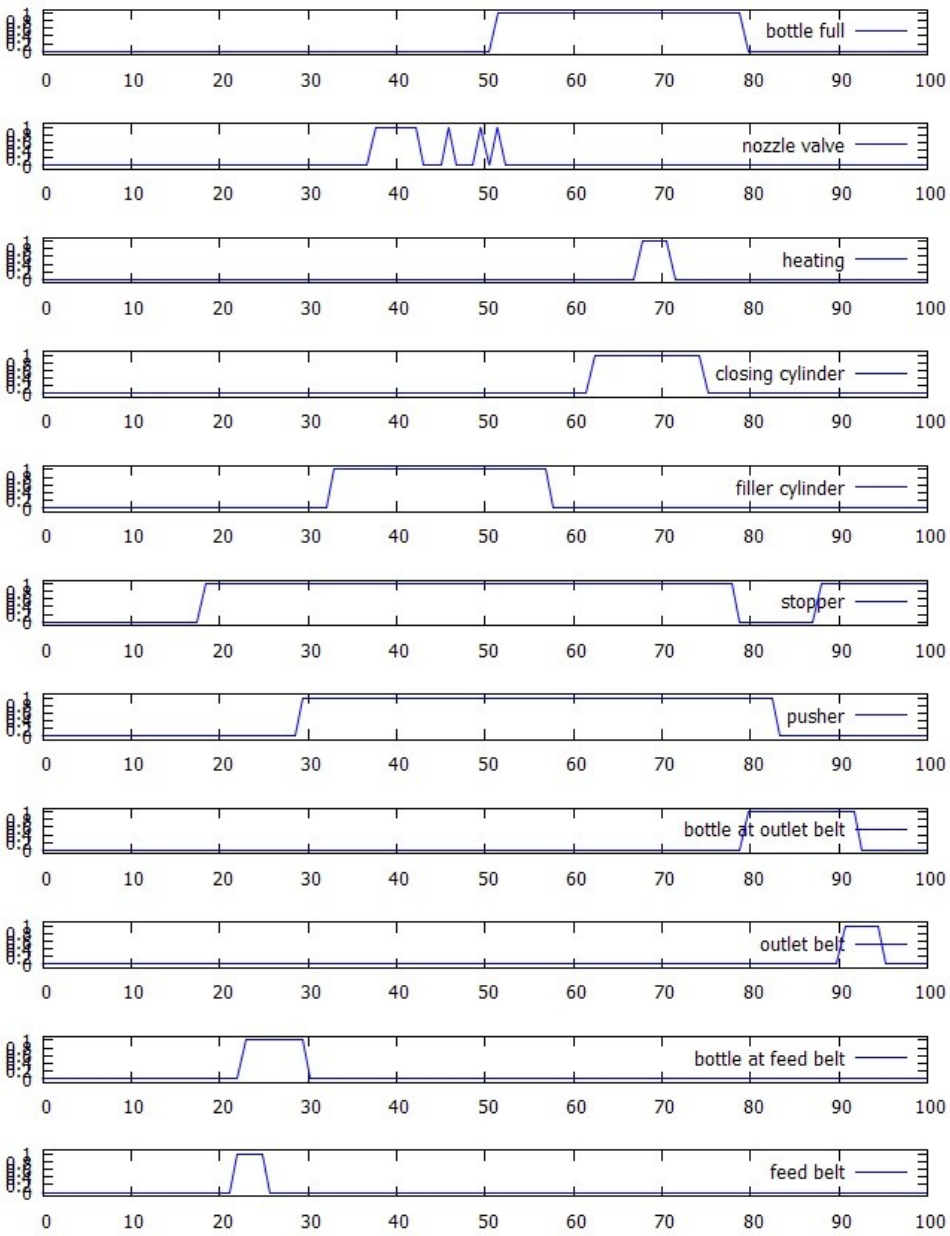
Gnuplot

Gnuplot ist eine viel verwendete, freie Software zur Anzeige von datenbasierten Graphiken. Mit ihr kann u.a. das zeitliche Verhalten der Anlage in Form von y-t-Diagrammen dargestellt werden.

Im einfachsten Fall wird am USB/COM-Anschluß des Arduinos mit der Simulation der Anlage ein speicherndes Terminal angeschlossen und die Ausgaben bzw. Messdaten in eine Textdatei protokolliert.

Erforderliches Skript mit Gnuplot Kommandos wenn die Messdaten in einer Datei f.data vorliegen – vgl. Datei f.plt.

```
clear
set multiplot
set yrange [-0.1:1.1]
set size 1,0.09
set origin 0,0
plot "f.data" u 1:2 t "feed belt" lc rgb "blue" w l
set origin 0,0.09
plot "f.data" u 1:3 t "bottle at feed belt" lc rgb "blue" w l
set origin 0,0.18
plot "f.data" u 1:4 t "outlet belt" lc rgb "blue" w l
set origin 0,0.27
plot "f.data" u 1:5 t "bottle at outlet belt" lc rgb "blue" w l
set origin 0,0.36
plot "f.data" u 1:6 t "pusher" lc rgb "blue" w l
set origin 0,0.45
plot "f.data" u 1:7 t "stopper" lc rgb "blue" w l
set origin 0,0.54
plot "f.data" u 1:8 t "filler cylinder" lc rgb "blue" w l
set origin 0,0.63
plot "f.data" u 1:9 t "closing cylinder" lc rgb "blue" w l
set origin 0,0.72
plot "f.data" u 1:10 t "heating" lc rgb "blue" w l
set origin 0,0.81
plot "f.data" u 1:11 t "nozzle valve" lc rgb "blue" w l
set origin 0,0.90
plot "f.data" u 1:12 t "bottle full" lc rgb "blue" w l
unset multiplot
```



Diese Darstellung zeigt den Verlauf eines Befüllvorgangs bei einer manuellen Steuerung.

Gnuplot erlaubt auch die Verwendung mehrerer Ausgabefenster, mehrerer Graphen in einem Fenster oder auch die Zusammenfassung mehrerer Graphen in einem Plot wie hier im Beispiel.

Auch eine kontinuierliche Darstellung während des Betriebes ist möglich – vgl. Datei fc.plt.