KONZEPT – ZUSTANDSMONITORING

Sarah Schmidt, Benjamin Mayer, Helen Stürzer, Isabelle Baldow

INFORMATIONSMANAGEMENT - IOT

FALL 1

Gruppenmitglieder:

- Benjamin Mayer
- Sarah Schmidt
- Helen Stürzer
- Isabelle Baldow

Ausgangssituation:
Als Anlagenbetreiber möchte ich einen Überblick über den Zustand meiner Anlage, um jederzeit flexibel auf bestimmte Ereignisse reagieren zu können.

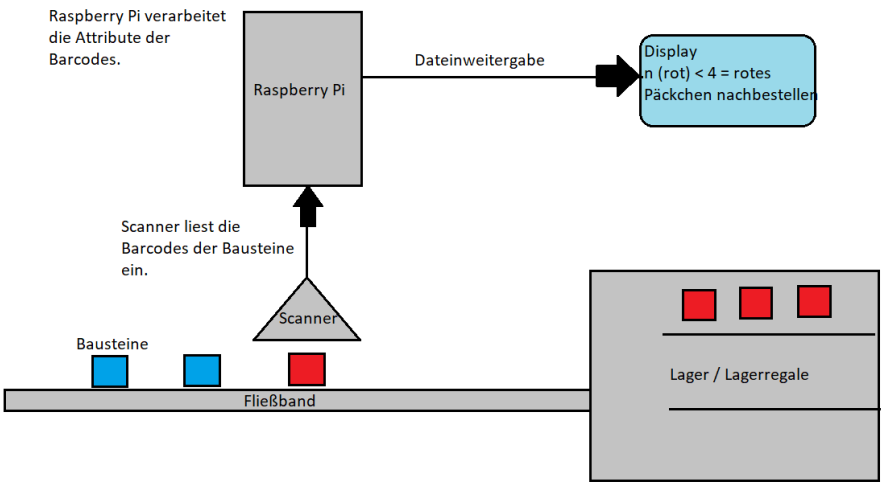
Ressourcen:

- Barcode Scanner Renkforce 2D FS5022J
- Raspberry Pi Model B 4 GB RAM
- Display
- 12 Päckchen

Darstellung des Lagerbestandes

Das Hauptmerk des Zustandsmonitoring liegt in diesem Semester bei der Darstellung des Lagerbestandes. Wir beschäftigen uns mit der Frage: **Wie viele Päckchen, mit definierten Eigenschaften, befinden sich zu einem bestimmten Zeitpunkt im Lager?**

Darstellung:

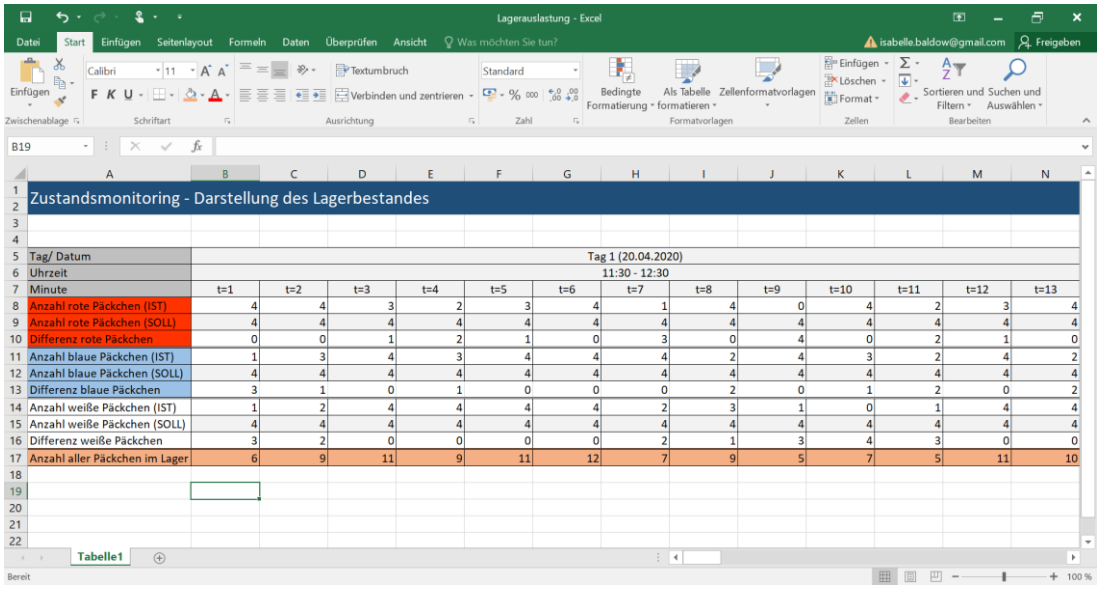


Ablauf:

1	Die Päckchen werden über ein Fließband in das Lager transportiert.
2	Vor dem Lager befindet sich ein Barcodescanner, der die Barcodes der Päckchen einliest und die Attributwerte auswertet.
3	Die Päckchen füllen das Lager sortiert nach Farbe. ➔ Die roten Päckchen füllen das erste Lagerregal. ➔ Die blauen Päckchen füllen das zweite Lagerregal. ➔ Die weißen Päckchen füllen das dritte Lagerregal.
4	Nach einer bestimmten Zeit verlassen die Päckchen das Lager wieder über das Fließband.
5	Der Barcodescanner liest die Barcodes der verlassenden Päckchen ein und verarbeitet die Attributwerte.
6	Die ein- und ausgehenden Attributwerte können über das angeschlossene Display eingesehen werden.
7	Es besteht die Möglichkeit, diese Werte in einer Excel-Tabelle zu dokumentieren, um genaue Aussagen zum Lagerbestand zu treffen. Folgende Aussagen wären möglich: ➔ n (rot) < 4 = erstes Lagerregal nicht ausgelastet, n rote Päckchen nachbestellen. ➔ n (blau) < 4 = zweites Lagerregal nicht ausgelastet, n blaue Päckchen nachbestellen. ➔ n (weiß) < 4 = drittes Lagerregal nicht ausgelastet, n weiße Päckchen nachbestellen.

FALL 1 - ANHANG

Beispiel zur Dokumentation des Lagerbestandes:



Lagerauslastung - Excel

Zustandsmonitoring - Darstellung des Lagerbestandes

Tag/ Datum	Tag 1 (20.04.2020)												
Uhrzeit	11:30 - 12:30												
Minute	t=1	t=2	t=3	t=4	t=5	t=6	t=7	t=8	t=9	t=10	t=11	t=12	t=13
Anzahl rote Päckchen (IST)	4	4	3	2	3	4	1	4	0	4	2	3	4
Anzahl rote Päckchen (SOLL)	4	4	4	4	4	4	4	4	4	4	4	4	4
Differenz rote Päckchen	0	0	1	2	1	0	3	0	4	0	2	1	0
Anzahl blaue Päckchen (IST)	1	3	4	3	4	4	4	2	4	3	2	4	2
Anzahl blaue Päckchen (SOLL)	4	4	4	4	4	4	4	4	4	4	4	4	4
Differenz blaue Päckchen	3	1	0	1	0	0	0	2	0	1	2	0	2
Anzahl weiße Päckchen (IST)	1	2	4	4	4	4	2	3	1	0	1	4	4
Anzahl weiße Päckchen (SOLL)	4	4	4	4	4	4	4	4	4	4	4	4	4
Differenz weiße Päckchen	3	2	0	0	0	0	2	1	3	4	3	0	0
Anzahl aller Päckchen im Lager	6	9	11	9	11	12	7	9	5	7	5	11	10

Darstellungsmöglichkeiten am Display:

n = 12	Lager ausgelastet
n (rot) < 4	n rote Päckchen nachbestellen
n (blau) < 4	n blaue Päckchen nachbestellen
n (weiß) < 4	n weiße Päckchen nachbestellen

Zudem besteht die Möglichkeit mit einem Ampelsystem zu arbeiten. Dies macht Sinn, da der Mensch antrainierte Signalfarben sehr schnell wahrnimmt und darauf reagieren kann. Da es auch Personen mit Farbenblindheit oder Rot-Grün-Schwäche gibt, sollte in jedem Fall der aktuelle Stand zusätzlich in Textform wiedergegeben werden.

Hierbei könnte der Displayhintergrund rot werden, wenn sich zu wenig Päckchen zum aktuellen Zeitpunkt im Lager befinden. Der Displayhintergrund könnte grün leuchten, wenn das Lager zum aktuellen Zeitpunkt voll ausgelastet ist. Der aktuelle Zeitpunkt beschreibt in diesem Fall den Moment der Lagerbestandabfrage.

FALL 2

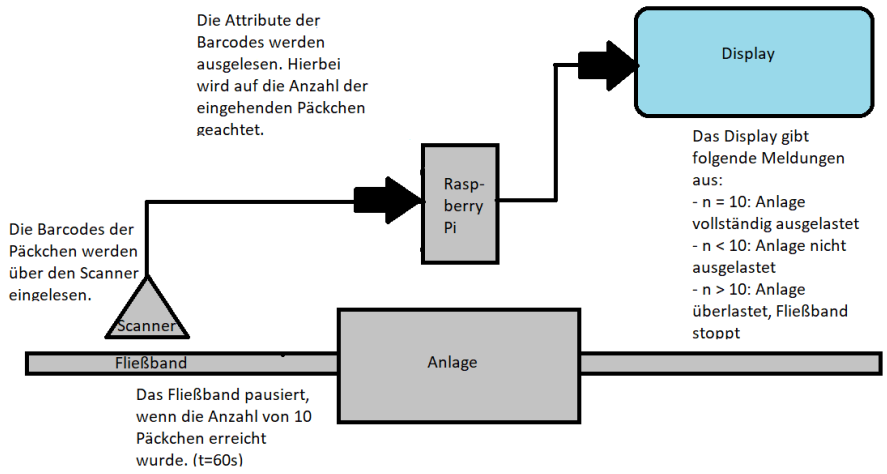
AUSLASTUNG DER ANLAGE

Wird die Anlage vollständig ausgelastet?
Die Anlage kann nur 10 Päckchen pro Minute über das Förderband aufnehmen. Nach 10 Päckchen pro Minute pausiert die Anlage. Nach Ablauf der Zeit (t=60s) läuft das Förderband wieder an. Wenn weniger als 10 Päckchen pro Minute über das Förderband in die Anlage transportiert werden, ist die Anlage nicht vollständig ausgelastet.

Anmerkung:
Im Weiteren wird der Fall 2 erläutert. Sofern die Zeit in diesem Semester ausreicht, wird dieser Fall ebenfalls bearbeitet.

Ressourcen:
- Barcode Scanner Renkforce 2D FS5022J
- Raspberry Pi Model B 4 GB RAM
- Display
- 12 Päckchen

Darstellung:



Ablauf:

1	Päckchen werden über das Fließband in die Anlage befördert.
2	Ein Scanner vor der Anlage liest die Barcodes der Päckchen ein.
3	Im Raspberry Pi werden die Attribute der Barcodes verarbeitet. In diesem Fall wird auf die Anzahl der Päckchen geachtet.
4	Erkennt das Raspberry Pi, dass 10 Päckchen innerhalb einer Minute eingescannt wurden, so wird die Meldung „n>10: Anlage überlastet. Fließband stoppen.“ auf dem Display ausgegeben.
5	Das Fließband wird manuell gestoppt.

Darstellungsmöglichkeiten am Display:

n = 10	Anlage ausgelastet
n < 10	Anlage nicht ausgelastet
n >10	Anlage überlastet → Fließband stoppen

FALL 3

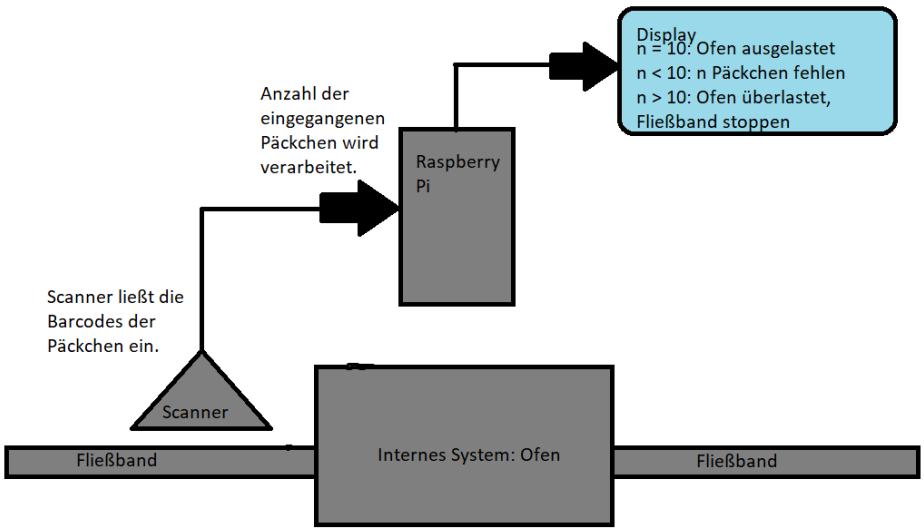
AUSLASTUNG EINES ANLAGEN-INTERNEN-SYSTEMS

Wie sehr wird ein internes System der Anlage ausgelastet?
In einem Verarbeitungssystem der Anlage, in unserem Beispiel ein Ofen, dürfen nur 10 Päckchen verarbeitet werden. Ein Scanner vor dem Ofen zählt die Stückzahl der Päckchen und pausiert nach 10 Päckchen das Förderband zum Eingang des Ofens. Bei mehr als 10 Päckchen würde der Ofen überhitzen.

Anmerkung:
Fall 3 dient als Inspiration für Folgesemester oder für die Tekom-Ausstellung. Dieser Fall wird wahrscheinlich nicht mehr in diesem Semester bearbeitet.

Ressourcen:
- Barcode Scanner Renkforce 2D FS5022J
- Raspberry Pi Model B 4 GB RAM
- Display
- 12 Päckchen

Darstellung:



Ablauf:

1	Die Päckchen werde über ein Fließband in ein internes System (Ofen) transportiert.
2	Der Scanner liest die Barcodes der Päckchen ein.
3	Die Attribute der Barcodes werden im Raspberry Pi verarbeitet. ➔ In Fall 3 liegt der Hauptmerk auf der Anzahl der eingehenden Päckchen. ➔ Eine Live-Anzeige gibt wieder, wie viele Päckchen sich schon im Ofen befinden.
4	Listet das Raspberry Pi 10 Päckchen, so erscheint auf dem Display die Meldung „n>10: Ofen überhitzt. Fließband stoppen“.
5	Der Ofen kann mit seiner Arbeit beginnen.
6	Ist der Ofen mit seinem Arbeitsschritt fertig, werden alle 10 Päckchen über ein Fließband aus der Anlage entlassen.

Darstellungsmöglichkeiten am Display:

n = 10	Ofen ausgelastet
n < 10	n Päckchen fehlen
n > 10	Ofen überlastet, Fließband stoppt

ERSTELLUNG
DER QR-CODES

Wir haben für jedes Päckchen einen QR-Code erstellt. Da es insgesamt 12 Päckchen sind, haben wir 12 QR-Codes erstellt, die jeweils Attribute des Päckchens enthalten. Für uns war lediglich das Attribut „Farbe“ wichtig, wobei für die Gruppe „Fehlercodes“ andere Attribute wichtig waren. Wir haben somit unsere erstellten QR-Codes an diese weitergeben können.

Die 12 erstellten QR-Codes:

Die Codes haben wir mit Hilfe der Webseite „QR Code Generator“ erstellt: https://www.qrcode-generator.de

ERSTER ENTWURF CODE

In diesem Kapitel werden die ersten unabhängigen Programmierungen dargestellt. Unabhängige Programmierung bedeutet, dass kein Scanner, Raspberry Pi und Display angeschlossen sind.

Das Programm zählt die eingehenden und ausgehenden Päckchen. Die Farbe und die Reihenfolge der eingehenden Päckchen sind damit fest definiert.

UNABHÄNGIGE PROGRAMMIERUNG (PYTHON)

Vorbereitung Programmierung

Als Programmsprache wird Python benutzt. Für die Einrichtung dabei wie folgt vorgehen:

1.	Python runterladen: https://www.python.org/downloads/
2.	Im Setup-Fenster „Add Python to Path“ auswählen. Das ermöglicht die Installation von zusätzlichen Python-Paketen, die im Laufe der Programmierung benötigt werden.
3.	Python installieren

Empfehlenswert ist zudem die Installation von PyCharm-Community oder einer ähnlichen integrierten Entwicklungsumgebung für Python. PyCharm-Community ist kostenlos. Die grafische Oberfläche erleichtert das Arbeiten am Programmcode.

PyCharm-Community kann hier runtergeladen werden: https://www.jetbrains.com/de-e/pycharm/download/#section=windows

Trigger:

Als erstes wurden die Trigger definiert, die anschließend im Programm wiedergegeben werden. Als Trigger bezeichnet man Auslöser, die zu einer Veränderung im Code führen.

1. **Scanner wird aktiviert**: Eingang von Päckchen → QR-Codes werden erfasst → Scanner wird aktiviert, weil Päckchen in die Anlage einsortiert werden
2. **QR-Code wird erkannt/ Scanner beginnt mit dem Hochzählen:** während des Eingangs → Raspberry Pi zählt die Anzahl der Päckchen und verarbeitet die Attribute (Farbe) → Päckchen werden im Lager platziert → Es wird bis 12 Päckchen hochgezählt
3. **13 ter QR-Code wird erkannt/ Scanner beginnt mit dem Runterzählen:** während der Ausgabe der Päckchen → Päckchen verlassen das Lager wieder → Anzahl der Päckchen wird runtergezählt
4. **24 ster QR-Code wird erkannt/ Scanner beginnt ab dem nächsten QR-Code mit dem Hochzählen:** Päckchen hat die Anlage verlassen → Scanner erfasst, dass das letzte Päckchen aus der Anlage sortiert wurde → Anlage beginnt wieder von vorne

Programmierung:

```
import time

rot = 0
blau = 0
weiß = 0


# Lager leer: Raufzählen
for x in range(1, 13):
    if x < 5: # x = 1,2,3,4
        rot += 1
    if x > 4 and x < 9: # x = 5,6,7,8
        blau += 1
    if x > 8: # x = 9,10,11,12
        weiß += 1
    print("Blau: " + str(blau) + ", Rot: " + str(rot) + ", Weiß: " +
str(weiß))
    time.sleep(0.5)

# Lager voll: Runterzählen
for x in range(1, 13):
    if x < 5: # x = 1,2,3,4
        rot -= 1
    if x > 4 and x < 9: # x = 5,6,7,8
        blau -= 1
    if x > 8: # x = 9,10,11,12
        weiß -= 1
    print("Blau: " + str(blau) + ", Rot: " + str(rot) + ", Weiß: " +
str(weiß))
    time.sleep(0.5)
```

Ausgabe:

Blau: 0, Rot: 1, Weiß: 0
Blau: 0, Rot: 2, Weiß: 0
Blau: 0, Rot: 3, Weiß: 0
Blau: 0, Rot: 4, Weiß: 0
Blau: 1, Rot: 4, Weiß: 0
Blau: 2, Rot: 4, Weiß: 0
Blau: 3, Rot: 4, Weiß: 0
Blau: 4, Rot: 4, Weiß: 0
Blau: 4, Rot: 4, Weiß: 1
Blau: 4, Rot: 4, Weiß: 2
Blau: 4, Rot: 4, Weiß: 3
Blau: 4, Rot: 4, Weiß: 4

Blau: 4, Rot: 3, Weiß: 4
Blau: 4, Rot: 2, Weiß: 4
Blau: 4, Rot: 1, Weiß: 4
Blau: 4, Rot: 0, Weiß: 4
Blau: 3, Rot: 0, Weiß: 4
Blau: 2, Rot: 0, Weiß: 4
Blau: 1, Rot: 0, Weiß: 4
Blau: 0, Rot: 0, Weiß: 4
Blau: 0, Rot: 0, Weiß: 3
Blau: 0, Rot: 0, Weiß: 2
Blau: 0, Rot: 0, Weiß: 1
Blau: 0, Rot: 0, Weiß: 0

ZWEITER ENTWURF
CODE

In diesem Kapitel wird auf die Programmierung in Verbindung mit Excel eingegangen.

Der in diesem Kapitel enthaltene Code speichert die Lagerbewegung in einer Excel-Datei. Die aktuelle Anzahl der jeweiligen Päckchen wird dabei in drei Spalten eingetragen. So können Lagerein- und -ausgänge dokumentiert werden.

PROGRAMMIERUNG (PYTHON) IN VERBINDUNG MIT EXCEL

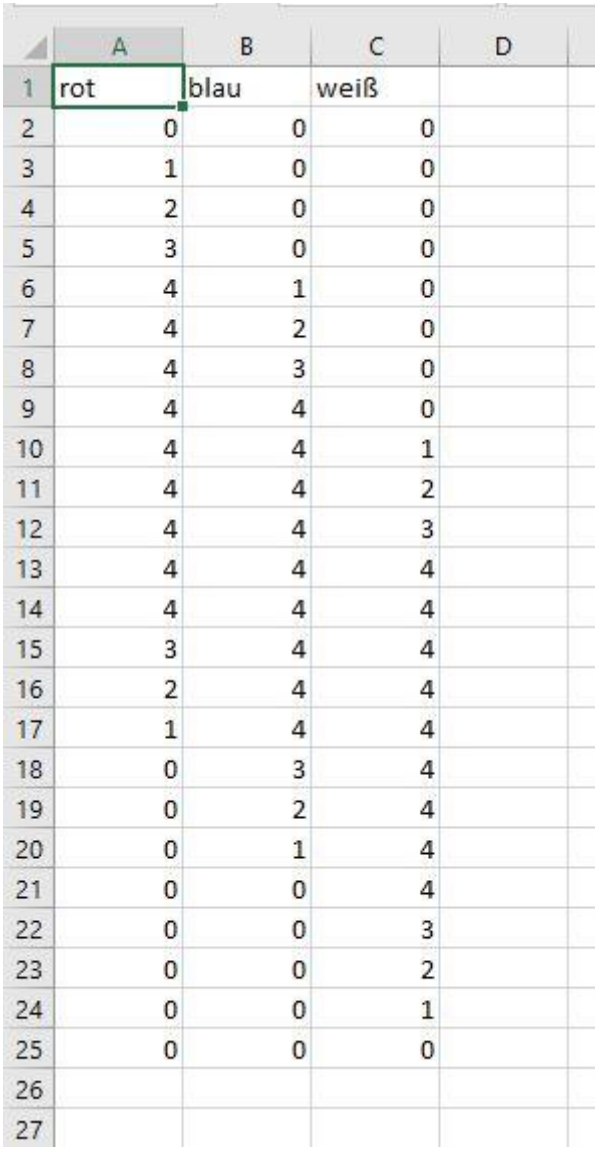
Unter folgendem Link befindet sich die Quelle unserer Programmierung sowie weitere nützliche Informationen zu dem Thema: https://www.youtube.com/watch?v=knnhkraHsBg&t=3s

Programmierung:

```
import xlsxwriter

#Excel-Datei erzeugen
outWorkbook = xlsxwriter.Workbook ("Lagerbewegung.xlsx")
outSheet = outWorkbook.add_worksheet()

#Daten deklarieren
farben = ["blau", "rot", "weiß"]
nummern = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]

#Spaltenüberschriften
outSheet.write ("A1", "rot")
outSheet.write ("B1", "blau")
outSheet.write ("C1", "weiß")

#Daten in die Spalten eintragen
outSheet.write (1, 0, nummern[0])
outSheet.write (2, 0, nummern[1])
outSheet.write (3, 0, nummern[2])
outSheet.write (4, 0, nummern[3])
outSheet.write (5, 0, nummern[4])
outSheet.write (6, 0, nummern[4])
outSheet.write (7, 0, nummern[4])
outSheet.write (8, 0, nummern[4])
outSheet.write (9, 0, nummern[4])
outSheet.write (10, 0, nummern[4])
outSheet.write (11, 0, nummern[4])
outSheet.write (12, 0, nummern[4])
outSheet.write (13, 0, nummern[4])
outSheet.write (14, 0, nummern[3])
outSheet.write (15, 0, nummern[2])
outSheet.write (16, 0, nummern[1])
outSheet.write (17, 0, nummern[0])
outSheet.write (18, 0, nummern[0])
outSheet.write (19, 0, nummern[0])
outSheet.write (20, 0, nummern[0])
outSheet.write (21, 0, nummern[0])
outSheet.write (22, 0, nummern[0])
outSheet.write (23, 0, nummern[0])
outSheet.write (24, 0, nummern[0])

outSheet.write (1, 1, nummern[0])
outSheet.write (2, 1, nummern[0])
outSheet.write (3, 1, nummern[0])
outSheet.write (4, 1, nummern[0])
outSheet.write (5, 1, nummern[1])
outSheet.write (6, 1, nummern[2])
```

```
outSheet.write (7, 1, nummern[3])
outSheet.write (8, 1, nummern[4])
outSheet.write (9, 1, nummern[4])
outSheet.write (10, 1, nummern[4])
outSheet.write (11, 1, nummern[4])
outSheet.write (12, 1, nummern[4])
outSheet.write (13, 1, nummern[4])
outSheet.write (14, 1, nummern[4])
outSheet.write (15, 1, nummern[4])
outSheet.write (16, 1, nummern[4])
outSheet.write (17, 1, nummern[3])
outSheet.write (18, 1, nummern[2])
outSheet.write (19, 1, nummern[1])
outSheet.write (20, 1, nummern[0])
outSheet.write (21, 1, nummern[0])
outSheet.write (22, 1, nummern[0])
outSheet.write (23, 1, nummern[0])
outSheet.write (24, 1, nummern[0])

outSheet.write (1, 2, nummern[0])
outSheet.write (2, 2, nummern[0])
outSheet.write (3, 2, nummern[0])
outSheet.write (4, 2, nummern[0])
outSheet.write (5, 2, nummern[0])
outSheet.write (6, 2, nummern[0])
outSheet.write (7, 2, nummern[0])
outSheet.write (8, 2, nummern[0])
outSheet.write (9, 2, nummern[1])
outSheet.write (10, 2, nummern[2])
outSheet.write (11, 2, nummern[3])
outSheet.write (12, 2, nummern[4])
outSheet.write (13, 2, nummern[4])
outSheet.write (14, 2, nummern[4])
outSheet.write (15, 2, nummern[4])
outSheet.write (16, 2, nummern[4])
outSheet.write (17, 2, nummern[4])
outSheet.write (18, 2, nummern[4])
outSheet.write (19, 2, nummern[4])
outSheet.write (20, 2, nummern[4])
outSheet.write (21, 2, nummern[3])
outSheet.write (22, 2, nummern[2])
outSheet.write (23, 2, nummern[1])
outSheet.write (24, 2, nummern[0])

outWorkbook.close()
```

Ausgabe in Excel:



	A	B	C	D
1	rot	blau	weiß	
2	0	0	0	
3	1	0	0	
4	2	0	0	
5	3	0	0	
6	4	1	0	
7	4	2	0	
8	4	3	0	
9	4	4	0	
10	4	4	1	
11	4	4	2	
12	4	4	3	
13	4	4	4	
14	4	4	4	
15	3	4	4	
16	2	4	4	
17	1	4	4	
18	0	3	4	
19	0	2	4	
20	0	1	4	
21	0	0	4	
22	0	0	3	
23	0	0	2	
24	0	0	1	
25	0	0	0	
26				
27				

Schritte für die weitere Ausgestaltung des Codes sind:

- Programmierung für die Päckchenzählung in Abhängigkeit des Barcodescanners und des Raspberry Pi gestalten
- Bisherige manuelle Programmierung der einzelnen Zellen automatisieren
- (Optional) Hinzufügen einer vierten Spalte „Timecode“
- Festlegung der Dateiablage: Bisher wird die erzeugte Datei bei erneutem Programmdurchlauf überschrieben

ENTWICKLUNG
PSEUDOCODE

Die „Readme“ enthält Code-Teile für die Initialisierung des Zustandsmonitorings, das so auch ohne Raspberry Pi auf einer virtuellen Maschine lauffähig ist. Integriert sind zudem die benötigten Python-Pakete unter dem Punkt „pip modules“.

Zusatzinformation:

Max Gärber der Firma Axxepta diente uns während des gesamten Projekts als Ansprechpartner. Über die Plattform GIT hat er uns viele nützliche Informationen bereitgestellt, wie auch diese Datei.

README-Datei von Herrn Max Gärber

Installation:

```
   # install package manager
   sudo apt-get -y install python3-pip

   #if not on a raspberry, need to add a "pi" user
   adduser pi
   cd /home/pi

   git clone https://git.axxepta.net/axxepta/raspberrypi4.git
${HOME}/raspberrypi4
   cd ${HOME}/raspberrypi4
   # you will probably need to update the barcode.conf because
of the application path if it will change
   cd supervisord
   # first time with param install to get pip modules
   ./install.sh install
```

Pip Modules:

```
# requirements
    pip3 install evdev
    pip3 install pathlib
    pip3 install websockets
    pip3 install websocket-client
    # for the zmq version
    pip3 install pyzmq

    #pip3 install redis (not needed with barcode_zmq_sub_2.py)
```

Settings:

- websocket url in `barcode_client.js`
- usb device in `barcode_zmq_pub.py`

Usefull commands and links:

find usb devices (only available if usb inputs are present)

```
ls -la /dev/input/by-id
```

- http://supervisord.org/
- https://git-scm.com/

Fazit

Der Umfang unseres Konzepts beschreibt einen Zwischenstand. Die aktuelle Darstellung des Zustandsmonitorings bezieht sich auf einen ausgearbeiteten Pseudocode, die erstellten QR-Codes und Informationen zum weiteren Vorgehen. Das gesamte Material bietet eine umfangreiche Grundlage für die Weiterentwicklung der Aufgabenstellung Zustandsmonitoring.

Unsere Probleme waren zum einen die fehlende Möglichkeit die Hochschulräumlichkeiten zu besuchen. Zum anderen hatten wir zu Beginn des Projekts nur wenig Bezug zum Thema und zur Programmierung mit Python.

Unsere kleinschrittige Vorgehensweise stellte sich als zielführend heraus. Trotz der anhaltenden Corona-Maßnahmen funktionierte unsere Kommunikation sehr gut. Wir haben uns für wöchentliche Meetings entschieden, bei denen wir uns ausgetauscht haben.

Die wichtigsten Phasen des Projekts waren: die genaue Erfassung der Aufgabenstellung, Skizzieren des Szenarios, die Entwicklung eines Pseudocodes und die Programmierung der Dateiausgabe in Excel. Parallel zu den Projektphasen fertigten wir die vorliegende Dokumentation an, auf die wir zu jedem Zeitpunkt zurückgreifen konnten.

Abschließend können wir sagen, dass wir sehr viele neue Inhalte mitgenommen haben. Wir wünschen der nächsten Semestergruppe viel Erfolg und Spaß bei der weiteren Ausarbeitung dieses Themas.

PROJEKTPLAN

Projektplan Gruppe Zustandsmonitoring				Isabell Baldow, Helen Stürzer, Sarah Schmidt, Benjamin Mayer										
	24.03. - 30.03	31.03. - 06.04.	07.04. - 13.04.	14.04. - 20.04	21.04. - 27.04.	28.04. -04.05.	05.05. - 11.05.	12.05. - 18.05.	19.05. - 25.05.	26.05. - 01.06.	02.06. - 08.06	09.06 - 15.06.	16.06. - 22-06.	23.06 - 29.06.
Konzeptphase														
Programmierung														
Implementierung														
Test														
Überarbeitung														
Fertigstellung														

Erklärung des Ablaufs

Konzeptphase = Festlegung der Zustände, die an der Fischertechnik-Anlage dargestellt werden sollen.

Programmierung = Konzept am Raspberry Pi und in Verbindung mit dem Display sowie dem Barcodescanner umsetzen.

Implementierung = Die Programmierungen fest in der Fischertechnik-Anlage integrieren.

Test = Mehrere Testläufe zur Prüfung, ob die Programmierungen richtig und vollständig umgesetzt wurden.

Überarbeitung = Die Behebung aller in der Testphase auffälligen Fehler.

Fertigstellung = Feinheiten-Korrekturen an der Anlage.

Zwischen allen Phasen kommt es zu Rücksprachen in den Meetings/ Vorlesungen.