

Steuerung des Fischertechnik-Modells „Hochregallager“ mit IoT

Stephan Laage-Witt – April 2025

1. Einleitung

Dieses Dokument beschreibt die Steuereinheit zum Hochregallager des Fischertechnik Logistik-Modells an der DHBW Lörrach, Fachbereich Wirtschaftsinformatik, Industrie 4.0 . Das Projekt ist eine Weiterentwicklung einer Steuerung, die im Rahmen einer Studienarbeit im Jahr 2021 von den Bachelor-Studenten Moritz Jehle, Anna Meier und Stefan Tobert erstellt wurde. In dem Projekt wurde die ursprüngliche Siemens-Steuerung mit einer Simatec S7 aus dem Jahr 2000 ersetzt durch ein System basierend auf einem Raspberry Pi, das die volle Funktionalität des Hochregallagers über eine MQTT-Schnittstelle entsprechend den Prinzipien von IoT bereit stellt.

Die wesentlichen Ziele der aktuellen Überarbeitung sind die Erhöhung der Fehlertoleranz, Überwachung des Betriebs zur Verbesserung der Sicherheit gegen mechanische Beschädigungen und vereinfachte Bedienung. Außerdem wurden zusätzliche Funktionen bereit gestellt.

2. Mechanik

Das System besteht aus 10 vertikalen Regalen mit jeweils 5 Fächern zum Lagern von Boxen. Ein Schlitten kann jede Realposition anfahren und dort Boxen ein- oder auslagern. Weiterhin gibt es ein Eingabeband für die Zufuhr von Boxen und ein Ausgabeband für die Auslieferung der Boxen. Das System beinhaltet 3 Achsen:

- X-Achse zur Auswahl des jeweiligen Regals mit der Nummer 1 bis 10
- Z-Achse zur Auswahl des jeweiligen vertikalen Levels mit der Nummer 1 bis 5
- Y für den Schlitten mit den 3 Positionen Store (im Regal), Default (Grundposition außerhalb des Regals) und Destore (Position zum Abholen oder Ablegen von Boxen auf den Bändern)

An der (von vorne gesehen) rechten Position $x=10$, $z=1$ befindet sich das Eingabeband. Dort werden Boxen erwartet, wenn sie in das Lager transportiert werden. An der linken Position $x=1$, $z=1$ befindet sich das Ausgabeband. Dort werden die Boxen nach dem Auslagern abgelegt.

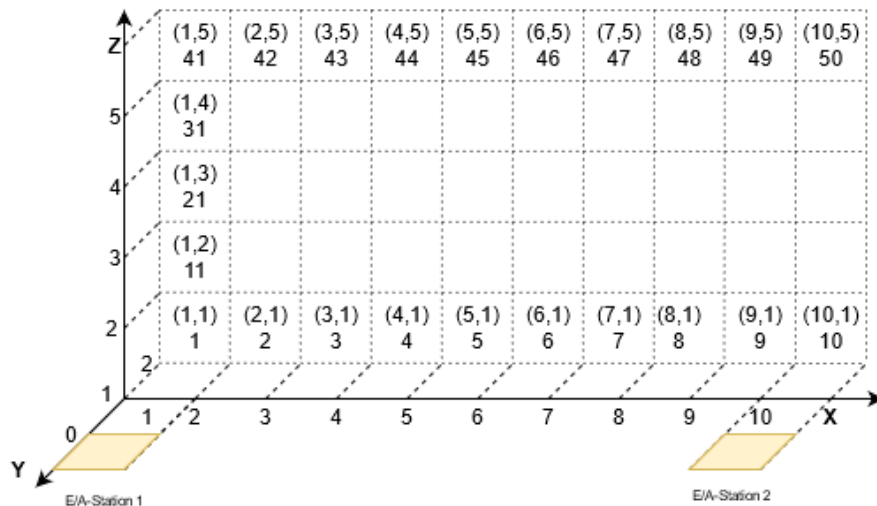


Bild 1: Die Mechanik verfügt über 3 Achsen

2. Steuerung - Hardware

Alle Komponenten der Steuerung finden auf einer Trägerplatte Platz. Dort befindet sich ein Raspberry Pi 4 in Verbindung mit einer Input-Output-Erweiterung mit 3 MCP23017 I/O Expander-Chips. Dadurch werden 32 digitale Eingänge und 16 digitale Ausgänge verfügbar. Die Ein- und Ausgänge werden mit Optokopplern an das Spannungsniveau (24V) der Anlage angepasst. Weiterhin ist ein einfaches User-Terminal vorhanden mit einem 4-zeiligen HD44780 Text-Display, 4 Leuchtdioden und 4 Eingabetastern.

Die Bedeutung der Leuchtdioden ist: grün → ready, gelb → busy, red → error, blue → storage full. Die Tasten blau, grün und gelb werden für die manuelle Steuerung verwendet (siehe Kapitel 7). Die rote Taste ist nur während Motorbewegungen aktiv und löst einen sofortigen Not-Stopp aus.

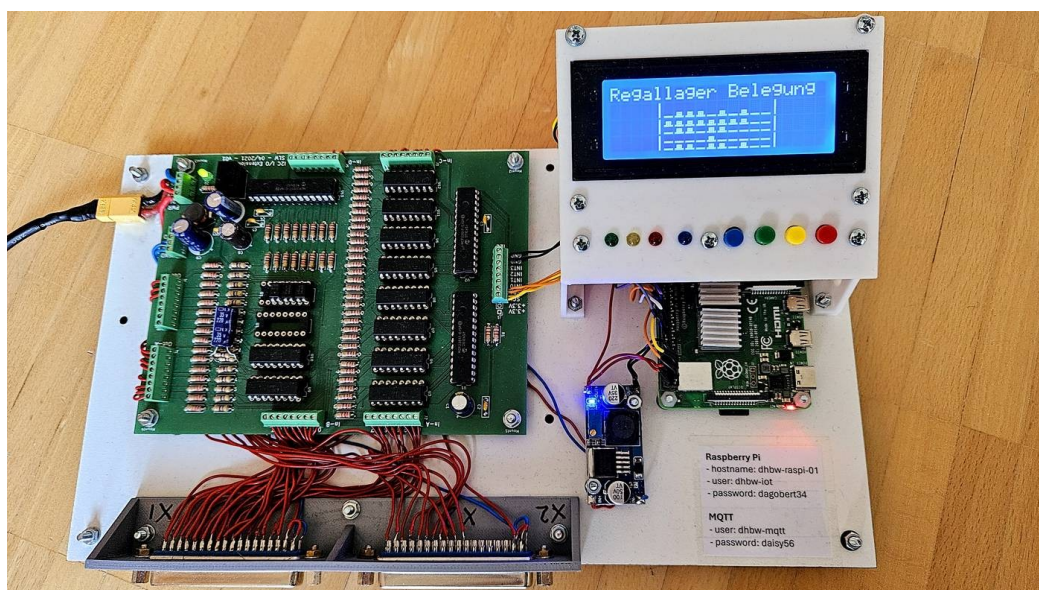


Bild 2: Steuerung mit Raspberry Pi, I/O-Extension und User Terminal

Die Spannungsversorgung der Anlage erfolgt über ein externes Netzteil mit 24V Gleichspannung. Die Trägerplatte verfügt über einen 24 zu 5V DC/DC-Wandler, der die Spannungsversorgung für die Logikkomponenten bereitstellt. Die Stromaufnahme der ganzen Anlage beträgt bis zu 1,5 A. Die Ein- und Ausgänge werden mit zwei D-Sub39 Kabeln mit den Steckern X1 und X2 des Fischertechnik-Modells verbunden.

3. Steuerung - Software

Die Software befindet sich auf dem Raspberry Pi im Wurzelverzeichnis der Applikation:

```
~/iot/high_bay_storage
```

Alle Komponenten der Software sind auf diesem Level abgelegt. Die Software ist vollständig in Python realisiert und verwendet einen modularen Aufbau, bestehend aus den folgenden Python Skripts:

1. *hbs_may.py* startet die Anwendung und enthält die Hauptschleife, die Eingaben über MQTT oder über die Tastatur entgegennimmt und abarbeitet.
2. *hbs_controller.py* enthält die höheren Funktionen zum Ein- und Auslagern von Boxen.
3. *hbs_operator.py* enthält die maschinennahen Funktionen zum Betrieb der Motoren und Auslesen der Sensoren. In diesem Modul befinden sich alle Funktionen, um die Achsen auf die gewünschten Positionen zu fahren.
4. *hbs_user_terminal.py* enthält alle Funktionen um das Display zu betreiben, die LEDs zu schalten und die Tasten auszulesen.
5. *hbs_mqtt_client.py* meldet einen Client beim MQTT-Broker an und empfängt und sendet MQTT Nachrichten.
6. *hbs_collections.py* enthält Aufzählungen für die Pin-Belegung, den System-Status und die Nachrichten, die beim laufenden Betrieb zwischen den Python-Modulen ausgetauscht werden.
7. *io_extension.py* ist ein Treiber für die I/O-Extension-Platine.

Außerdem wird die Datei *hbs_messages_de.dat* benötigt. Dort sind die Anzeigen für das Display als deutsche Texte abgelegt. Gegebenenfalls können andere Sprachmodule eingebunden werden.

Die aktuelle Belegung des Lagers wird in der Datei *storage_places.pkl* gespeichert. Diese Datei befindet sich im Unterordner *obj*. Dabei handelt es sich um die gespeicherte Version eines Python Dictionaries. Wenn beim Systemstart keine Belegungsdatei vorhanden ist, dann wird eine neue Datei mit einem vollständig leeren Regal angelegt.

Das System verwendet eine virtuelle Python-Umgebung, die im Verzeichnis

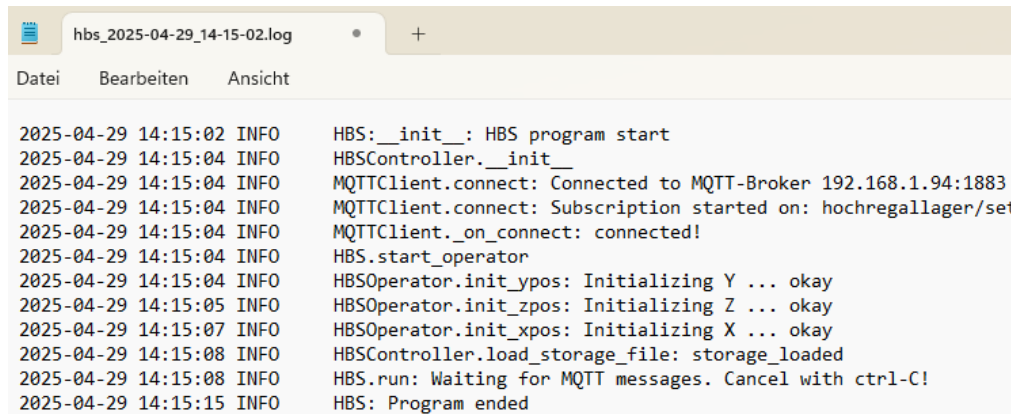
```
~/iot/iot_venv
```

abgelegt ist. Die folgenden zusätzlichen (nicht in der Standard-Installation enthaltenen) Python-Pakete wurden mit dem Tool *pip* installiert:

- *paho*
- *Rpi.GPIO*
- *RPLCD*

Das System benötigt für den Betrieb zwingend eine WLAN-Verbindung und einen aktiven MQTT-Broker. Der Broker, hier Mosquitto, ist in dieser Installation auf demselben Raspberry Pi installiert. Er ist über die IP-Adresse des Raspberry Pis und die MQTT-Credentials erreichbar.

Das System legt bei jedem Systemstart in dem Unterordner `logfiles` ein Logfile an. Der Name des Logfiles besteht aus `hbs_` und dem aktuellen Datums- und Zeitstempel. Im Logfile werden alle Aktivitäten, Nachrichten und gegebenenfalls Fehlermeldungen protokolliert.



```
hbs_2025-04-29_14-15-02.log
Datei  Bearbeiten  Ansicht

2025-04-29 14:15:02 INFO  HBS:__init__: HBS program start
2025-04-29 14:15:04 INFO  HBSController.__init__
2025-04-29 14:15:04 INFO  MQTTClient.connect: Connected to MQTT-Broker 192.168.1.94:1883
2025-04-29 14:15:04 INFO  MQTTClient.connect: Subscription started on: hochregallager/set
2025-04-29 14:15:04 INFO  MQTTClient._on_connect: connected!
2025-04-29 14:15:04 INFO  HBS.start_operator
2025-04-29 14:15:04 INFO  HBSOperator.init_ypos: Initializing Y ... okay
2025-04-29 14:15:05 INFO  HBSOperator.init_zpos: Initializing Z ... okay
2025-04-29 14:15:07 INFO  HBSOperator.init_xpos: Initializing X ... okay
2025-04-29 14:15:08 INFO  HBSController.load_storage_file: storage_loaded
2025-04-29 14:15:08 INFO  HBS.run: Waiting for MQTT messages. Cancel with ctrl-C!
2025-04-29 14:15:15 INFO  HBS: Program ended
```

4. Systemstart

Nach dem Einschalten der Stromversorgung startet der Raspberry Pi das Betriebssystem. Nach etwa 20 Sekunden wird das Skript `hbs_main.py` automatisch (mit Hilfe einer desktop-Datei im Autostart-Ordner des Users) gestartet.

Die Anwendung überprüft, ob eine WLAN-Verbindung vorhanden ist. Wenn das nicht der Fall ist, wird das Programm beendet. Anschließend werden die Netzwerkdaten (SSID und IO-Adresse) auf dem Display angezeigt.

Nach 20 Sekunden wird die Initialisierung der 3 Achsen in der Reihenfolge $y \rightarrow z \rightarrow x$ durchgeführt. Sofern die Initialisierung erfolgreich verlaufen ist, wird das System mit dem MQTT-Broker verbunden und ist damit betriebsbereit. Die einzelnen Schritte des Systemstarts werden mit entsprechenden Nachrichten auf dem Display angezeigt. Die Betriebsbereitschaft nach dem Einschalten wird durch die Displayanzeige „MQTT bereit“ und die grüne Leuchtdiode signalisiert.

5. MQTT Interface

Das System kommuniziert mit dem Anwender über MQTT. Dabei werden folgenden Themen verwendet:

- „hochregallager/set“: Unter diesem Thema empfängt das Hochregallager Anforderungen, die die verfügbaren Aktionen auslösen. Die Eingaben werden in einer Warteschlange gepuffert und der Reihe nach ausgeführt.
- „hochregallager/status“: Über dieses Thema wird der aktuelle System-Status publiziert. Es gibt drei Stati:
 - „ready“ zeigt die Eingabebereitschaft an. Das System ist zur Zeit nicht beschäftigt.
 - „busy“ zeigt an, dass zur Zeit eine Operation ausgeführt wird. Das HBS nimmt trotzdem Eingaben an, die aber in der Warteschlange angelegt werden.

- „error“ bedeutet, dass ein Fehler aufgetreten ist. In der Regel wird eine manuelle Intervention notwendig sein, um den Fehler zu beheben.
- „hochregallager/result“: Jede Operation wird nach der Ausführung mit einem Ergebnis-Code quittiert. Eine Liste der Ergebnis-Codes ist in Kapitel 8 aufgeführt. Im Normalfall wird die Nachricht „okay“ zurück gegeben, was bedeutet, dass die Operation erfolgreich ausgeführt wurde.

6. Eingabe und Ausführung von Operationen

Das System akzeptiert Eingaben im JSON-Format. Das Schlüsselwort „operation“ ist zwingend erforderlich. Je nach Art der Anforderung werden die Argumente „x“ und „z“ bzw. zusätzlich „x_new“ und „z_new“ erwartet. Bei den Eingaben wird die Groß/Klein-Schreibung ignoriert (case-insensitive). Folgende Operationen sind zur Zeit implementiert:

Operation	Argumente	Aktion
store	X, z	Holt eine Box vom Eingabe-Band und legt es im gewählten Fach ab.
destore	X, z	Holt eine Box vom gewählten Fach und legt sie auf dem Ausgabe-Band ab.
rearrange	X, z, x_new, z_new	Holt eine Box aus einem Fach und legt die Box in einem anderen Fach ab
store_random	Keine	Holt eine Box vom Eingabe-Band und legt sie in einem zufällig gewählten freien Fach ab.
destore_ranom	Keine	Holt eine Box aus einem zufällig gewählten Fach ab und legt sie auf das Ausgabe-Band
init_x	Keine	Initialisiert die X-Achse. Dabei wird sichergestellt, dass die Achse auf eine definierte Position fährt.
init_y	Keine	Initialisiert die Y-Achse
init_z	Keine	Initialisiert die Z-Achse
show_occupancy	Keine	Zeigt die Regal-Belegung auf dem Display an und gibt die Belegung als einen String via MQTT zurück. Format: <code>occupancy:_**_*_*_*_____*****_*****_____*_*_*__</code>
shutdown	Keine	Fährt den Raspberry Pi herunter. Nach etwa 20 Sekunden kann die Stromversorgung getrennt werden.

Beispiele:

- **Anweisung ohne Argumente:**
`mosquitto_pub -u dhw-mqtt -P daisy56 -t hochregallager/set -m "{\"operation\": \"STORE_RANDOM\"}"`
- **Anweisung mit 2 Argumenten:**
`mosquitto_pub -u dhw-mqtt -P daisy56 -t hochregallager/set -m "{\"operation\": \"STORE\", \"X\": 8, \"Z\": 2}"`
- **Anweisung mit 4 Argumenten:**
`mosquitto_pub -u dhw-mqtt -P daisy56 -t hochregallager/set -m "{\"operation\": \"REARRANGE\", \"X\": 1, \"Z\": 4, \"x new\": 5, \"z new\": 3}"`

7. Manuelle Steuerung

Das System kann über die Tasten manuell gesteuert werden. Die blaue Taste dient zur Aktivierung der manuellen Steuerung. Anschließend wird die gewählte Achse im Display angezeigt mit der grünen Taste wird die Achse um eine Position auf den nächsten niedrigeren Wert, und mit der gelben Taste auf den nächsten höheren Wert gefahren. Die blaue Taste rotiert über die 3 Achsen X, Y und Z.

Programm-Ende: Wenn die grüne und die gelbe Taste zusammen betätigt werden, dann wird das Python-Skript verlassen. Der Raspberry Pi bleibt eingeschaltet und ist z.B. über VNC erreichbar.

Shutdown: Wenn die grüne und die rote Taste zusammen betätigt werden, dann wird das Python-Skript verlassen, und der Raspberry Pi wird heruntergefahren. Nach etwa 20 Sekunden kann die Stromversorgung abgeschaltet werden. Achtung: Die Stromversorgung sollte nie im laufenden Betrieb getrennt werden, da dies zu Datenverlusten und/oder Beschädigung des Betriebssystems auf der SD-Karte führen kann.

8. MQTT Nachrichten und Anzeigen auf dem Display

Jede Operation wird nach Abschluss mit einer MQTT Nachricht unter dem Thema „hochregallager/result“ bestätigt. Im Normalfall (fehlerfreie Ausführung) wird „okay“ übermittelt. Im Falle eines Fehlers wird eine entsprechende Nachricht ausgegeben.

MQTT Nachricht	Anzeige im User Terminal	Kommentar
okay	Okay	Fehlerfreie Ausführung
err_shelf_empty	Keine Box im Fach	Das ausgewählte Fach enthält keine Box.
err_shelf_occupied	Fach belegt	Das ausgewählte Fach ist belegt.
err_wrong_x_target	Falsche X-Position	Die angewählte X-Position befindet sich außerhalb des zulässigen Bereichs
err_wrong_y_target	Falsche Y-Position	Die angewählte Y-Position befindet sich außerhalb des zulässigen Bereichs
err_wrong_z_target	Falsche Z-Position	Die angewählte Z-Position befindet sich außerhalb des zulässigen Bereichs
err_wrong_z_level	Falsche Z-Ebene	Die angewählte Ebene befindet sich außerhalb des zulässigen Bereichs
err_x_pos	X Achsen-Fehler	Fehler bei der Steuerung der X-Achse. Die Achse hat die erwartete Sensorposition nicht innerhalb der definierten Ablaufzeit erreicht.
err_y_pos	Y Achsen-Fehler	Fehler bei der Steuerung der Y-Achse. Die Achse hat die erwartete Sensorposition nicht innerhalb der definierten Ablaufzeit erreicht.
err_z_pos	Z Achsen-Fehler	Fehler bei der Steuerung der Z-Achse. Die Achse hat die erwartete Sensorposition nicht innerhalb der definierten Ablaufzeit erreicht.
err_xz_pos	X/Z Achsen-Fehler	Fehler bei einer kombinierten X/Z-Achsen-Bewegung
err_x_udf	X Achse undefiniert	Die aktuelle Position der X-Achse ist nicht bekannt. Dieser Fehler kann in der Regel durch eine Initialisierung behoben werden.
err_y_udf	Y Achse undefiniert	Die aktuelle Position der Y-Achse ist nicht bekannt. Dieser Fehler kann in der Regel durch eine Initialisierung behoben werden.
err_z_udf	Z Achse undefiniert	Die aktuelle Position der Z-Achse ist nicht bekannt. Dieser Fehler kann in der Regel durch eine Initialisierung behoben werden.
err_x_init	X-Initial. Fehler	Bei der Initialisierung der X-Achse ist ein Fehler aufgetreten.
err_y_init	Y-Initial. Fehler	Bei der Initialisierung der Y-Achse ist ein Fehler aufgetreten.
err_z_init	Z-Initial. Fehler	Bei der Initialisierung der Z-Achse ist ein Fehler aufgetreten.
err_storage_full	Lager ist voll	Das Regalsystem ist voll. Es gibt keine freien Plätze.

MQTT Nachricht	Anzeige im User Terminal	Kommentar
err_storage_empty	Lager ist leer	Das Regalsystem ist leer. Es gibt keine Boxen.
err_json_format	JSON falsches Format	Die übermittelte MQTT-Nachricht entspricht nicht dem JSON-Format
err_json_noop	JSON keine Operation	In der übermittelten MQTT-Nachricht fehlt das Schlüsselwort „operation“
err_cmd_unknown	Aktion unbekannt	Die angeforderte Operation existiert nicht.
err_wrong_args	Falsche Parameter	In der übermittelten MQTT-Nachricht befinden sich falsche Parameter
err_wrong_arg_cnt	Falsche Par. Zahl	In der übermittelten MQTT-Nachricht stimmen die Anzahl der Parameter nicht mit der Anforderung überein
err_internal	Interner Fehler	Hierbei handelt es sich um einen Fehler im Python-Skript, der im Normalfall nicht auftreten sollte.

Die folgende weiteren Nachrichten können auf dem Display erscheinen, werden aber nicht über MQTT publiziert.

MQTT Nachricht	Anzeige im User Terminal	Kommentar
wlc_01	* Hochregal-Lager *	Text beim Systemstart, 1. Zeile
wlc_02	DHBW Lörrach 04/2025	Text beim Systemstart, 2. Zeile
err_nonet	Kein Netzwerk	Der Raspberry Pi verfügt nicht über eine Netzwerk-Verbindung
mqtt_connected	MQTT verbunden	Die Anmeldung beim MQTT-Broker war erfolgreich
mqtt_ready	MQTT bereit	Das System wartet auf Eingaben via MQTT
mqtt_disconnect	MQTT beendet	Das System hat die Verbindung zum MQTT-Broker geschlossen
msg_x_man	X-Achse	Manuelle Steuerung der X-Achse
msg_y_man	Y-Achse	Manuelle Steuerung der Y-Achse
msg_z_man	Z-Achse	Manuelle Steuerung der Z-Achse
storage_loaded	Belegung geladen	Die Datei mit der Regalbelegung wurde geladen
storage_loaded	Belegung angelegt	Die Datei mit der Regalbelegung wurde nicht gefunden. Deshalb wurde eine neue Datei mit einem leeren Regal angelegt.
err_storage_io	Belegung Ladefehler	Eine Datei mit der Regalbelegung wurde gefunden, konnte beim Systemstart aber nicht geladen werden.
sys_exit	Programm-Ende	Das Python-Skript wird beendet. Der Raspberry Pi bleibt eingeschaltet.
shutdown	System Shutdown	Das Python-Skript wird beendet. Der Raspberry Pi wird heruntergefahren.
poweroff	Wird abgeschaltet	Der Raspberry Pi wird abgeschaltet. Danach kann die Stromversorgung getrennt werden.

SLW 20250429