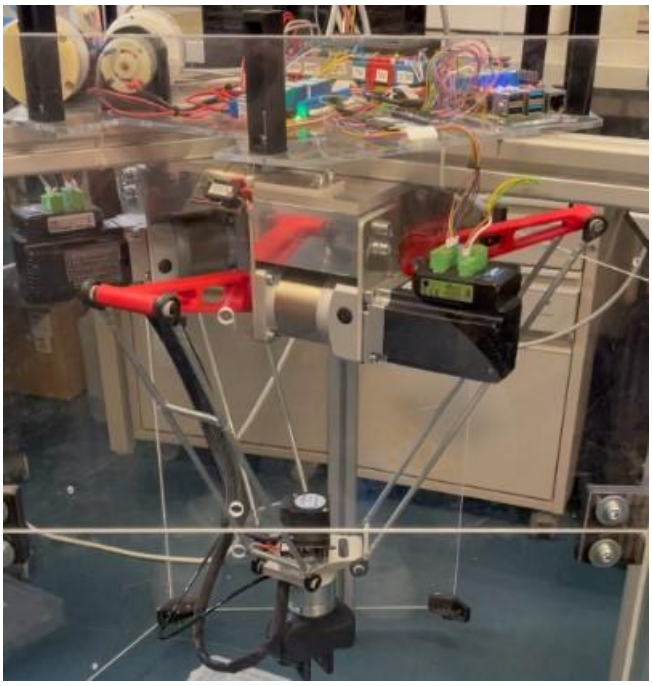
Deltarobot Guide



Wichtige Daten:

Netzwerk Konfiguration:

IP:	192.168.50.1
DNS-Name:	deltarobot
Deltarobot Webapp IP:	192.168.50.1:3000
RaspAP-Admin-GUI IP:	192.168.50.1:80
RaspAP-Admin-Name:	admin
RaspAP-Admin-Password:	secret
RaspAP-Netzwerk SSID:	Deltarobot
RaspAP-Netzwerk Password:	ChangeMe

SSH-Konfiguration:

Nutzername:	pi
Password:	raspberry

Wenn sich der Raspberry Pi nicht automatisch mit Ihrem Hotspot oder WLAN-Netzwerk verbindet und dadurch keinen Internetzugang hat, müssen Sie die RaspAP-Admin-GUI unter 192.168.50.1:80 besuchen und sich dort mit einem Internetnetzwerk verbinden.

Wichtige Hinweise

- Die sechs Schalter an den Motortreibern sind entscheidend für die Funktionalität und sollten daher nicht willkürlich betätigt werden. Für mehr Informationen lesen Sie bitte die Datenblätter des Motors.
- Wenn das Schaltnetzteil der Motoren ausgeschaltet wird, muss es sich erst entladen, was dazu führen kann, dass die Motoren für eine kurze Zeit weiterhin Strom erhalten und dementsprechend weiterlaufen. Der Schalter des Netzteils sollte daher nicht als Notausschalter verwendet werden.
- Im G-Code-Editor werden fehlerhafte Angaben gelb hervorgehoben, während der Text rot eingefärbt wird. Der G-Code kann auch bei Vorliegen von Fehlern gespeichert und ausgeführt werden. Dies sollte jedoch nur geschehen, wenn Sie die technischen Implikationen vollständig nachvollziehen können.
- Bitte achten Sie darauf, dass niemand den Roboter bedient, während Sie den Greifer wechseln. Auch wenn der Knopf (deactivate Magnet) in den Einstellungen gedrückt gehalten wird, können andere Personen, die Zugriff auf das Netzwerk haben, den Roboter steuern!

Einrichtung des Projekts lokal

Vorbereitung der Raspberry Pi SD-Karte

1. Erstellen eines Raspberry Pi Images auf einer SD-Karte

Der Raspberry Pi ist wie folgt vorkonfiguriert:

- Hostname: deltarobot

- Benutzer: pi

- Passwort: raspberry

- SSH-Authentifizierung über Passwort

- WLAN-Netzwerkname: Hotspot (das ist ein privater Hotspot/Netzwerk von Ihnen für den Internetzugang)

- WLAN-Netzwerkpasswort: ChangeMe

- Standort: DE

Konfiguration des Raspberry Pi als Zugangspunkt (AP-STA Modus)

(https://docs.raspap.com/ap-sta/#installation)

2. Konfigurieren Sie den Raspberry Pi als Zugangspunkt

- Verbinden Sie sich über das vorkonfigurierte WLAN-Netzwerk "Hotspot" und dann über SSH mit dem Raspberry Pi.

- Installieren Sie RaspAP, um den Raspberry Pi sowohl als Zugangspunkt (AP) als auch als drahtlosen Client/Station (STA) zu betreiben:

```bash
curl -sL https://install.raspap.com | bash
```

- Starten Sie den Raspberry Pi nach der Installation neu und verbinden Sie sich erneut über das vorkonfigurierte WLAN-Netzwerk.

- Greifen Sie über deltarobot.local oder über die IP-Adresse auf die Administrationsoberfläche zu (Benutzername: admin, Passwort: secret).

- Konfigurieren Sie den Raspberry Pi für den AP-STA-Modus:

- Stellen Sie sicher, dass das Dashboard-Widget "Wireless Client" eine aktive Verbindung anzeigt.

- Gehen Sie zu Hotspot > Erweitert und aktivieren Sie die Option "WiFi client AP mode". Starten Sie den Raspberry Pi nach der Aktivierung des WiFi Client AP Modus bitte neu.

- Nach dem Neustart können Sie auch die statische IP-Adresse, die SSID und das Passwort ändern. (Die vorkonfigurierte statische IP-Adresse ist 192.168.50.1. Wenn Sie eine andere konfigurieren, stellen Sie sicher, dass Sie die IP in Frontend/.env und in Backend/Webserver/server.js aktualisieren.)

- Nach einem weiteren Neustart sollte die Raspberry Pi AP-SSID erscheinen (Standard-SSID: raspi-webgui, Passwort: ChangeMe).

Software-Installation und -Konfiguration

3. Git installieren und konfigurieren:

   Installieren Sie Git und konfigurieren Sie Ihre Benutzerinformationen:

   ```bash
   sudo apt install git
   git config --global user.name "Ihr Name"
   git config --global user.email "Ihre.Email@example.com"
   ```

4. Repository klonen:

   ```bash
   git clone https://github.com/MayarAnon/Deltaroboter.git
   ```

5. Deltarobot einrichten:

   Navigieren Sie zum Skriptverzeichnis und führen Sie das Setup-Skript aus:

   ```bash
   cd Deltarobot/
   cd Bash/
   ./setup.sh
   ```

6. Deltarobot-Dienste hinzufügen:

Führen Sie das Services-Skript aus:

```bash
cd Deltarobot/
cd Bash/
./services.sh
```

7. Neustart mit sudo:

```bash
sudo reboot
```

G-Code Übersicht

Allgemeine Bewegungsbefehle

G0 - Schnelle Positionierung (Rapid Positioning)

Bewegt das Werkzeug schnell zu einer angegebenen Position ohne Bearbeitung.

Beispiel: G0 X0 Y0 Z-280 A20 F80 – Schnelle Bewegung zur Koordinatenposition mit Greiferdrehung. (A20 -> Greiferdrehung von 20°; F80 -> 80% der zulässigen Geschwindigkeit)

G1 - Lineare Interpolation (Linear Interpolation)

Bewegt das Werkzeug linear zur angegebenen Position und führt dabei eine Bearbeitung aus.

Beispiel: G1 X0 Y0 Z-280 A20 F80 – Lineare Bewegung zur Koordinatenposition mit gleichmäßiger Greiferdrehung.

Kreisförmige Bewegungsbefehle

G2 - Kreisinterpolation im Uhrzeigersinn (Clockwise Circular Interpolation)

Bewegt das Werkzeug auf einem kreisförmigen Pfad im Uhrzeigersinn zur Zielposition.

Beispiel: G2 X20 Y0 I10 J0 A10 F20

X20 Y0: Diese Koordinaten definieren die Zielposition des Werkzeugs auf der XY-Ebene. In diesem Fall soll das Werkzeug zur X-Position 20 und zur Y-Position 0 bewegt werden.

I10 J0: Diese Parameter sind die Koordinaten des Kreismittelpunkts relativ zur aktuellen Position des Werkzeugs. I10 bedeutet, dass der Mittelpunkt des Kreises 10 Einheiten in X-Richtung von der aktuellen Position entfernt ist. J0 bedeutet, dass der Mittelpunkt des Kreises auf derselben Y-Position wie die aktuelle Position des Werkzeugs liegt.

G3 - Kreisinterpolation gegen den Uhrzeigersinn (Counterclockwise Circular Interpolation)

Führt eine kreisförmige Bewegung gegen den Uhrzeigersinn zur Zielposition aus.

Beispiel: G3 X20 Y0 I10 J0 A10 F20 – Gegenläufiger Kreisbogen zur gleichen Position mit Drehung.

Spezialbefehle

G4 - Wartezeit (Dwell Time)

Pausiert die Maschinenaktivität für eine festgelegte Zeit.

Beispiel: G4 P500 – Maschine wartet 500 Millisekunden.

G17, G18, G19 - Auswahl der Arbeitsebene (Plane Selection)

G17 – Auswahl der XY-Ebene.

G18 – Auswahl der YZ-Ebene.

G19 – Auswahl der ZX-Ebene.

G28 - Fahrt zur Homeposition (Return to Home)

Fahrt das Werkzeug in die anfängliche Ausgangsposition zurück.

Beispiel: G28 F80 – Rückkehr zur Homeposition mit spezifischer Geschwindigkeit.

Greifersteuerungsbefehle

M100, M200, M300, M400 - steuern verschiedene Greifertypen.

Parallelgreifer (M100): Einstellbar von S0 (geschlossen) bis S100 (geöffnet).

Magnetgreifer (M300): S0 schaltet aus, S1 schaltet ein.

Vakuumgreifer(M400) und Compliantgreifer(M200): S-1 aktiviert das Vakuum, S0 schaltet aus, S1 aktiviert den Druck.

Beispiel: M100 S50 – Aktiviert den Parallelgreifer und setzt ihn auf 50% der Kapazität.

API-Schnittstellenbeschreibung

1. Laden der GCode-Dateien

    URL: /loadGCodeFiles

    Methode: GET

    Beschreibung: Lädt alle GCode-Dateien aus einem spezifizierten Verzeichnis.

    Antworten:

    200 OK: Erfolgreich geladene Dateien.

    Beispiel: [{"fileName": "example.gcode", "content": "GCode content here..."}]

    500 Internal Server Error: Fehler beim Lesen der Dateien.

    Beispiel: {"error": "Fehler beim Lesen der Datei."}

2. Löschen einer GCode-Datei

    URL: /deleteGCode

    Methode: DELETE

    Query-Parameter:

    name: Name der zu löschenden GCode-Datei.

    Beschreibung: Löscht eine spezifizierte GCode-Datei.

    Antworten:

    200 OK: Datei erfolgreich gelöscht.

    Beispiel: {"message": "Datei 'example.gcode' erfolgreich gelöscht."}

    400 Bad Request: Kein Dateiname angegeben.

    Beispiel: {"error": "Dateiname nicht angegeben"}

    404 Not Found: Datei nicht gefunden.

    Beispiel: {"error": "Datei 'example.gcode' nicht gefunden"}

3. Speichern eines GCode-Programms

    URL: /gcode

    Methode: POST

    Body:

    name: Name der Datei.

content: Inhalt des GCode.

Beschreibung: Speichert ein neues GCode-Programm.

Antworten:

200 OK: Programm erfolgreich gespeichert.

Beispiel: {"message": "Programm 'example.gcode' erfolgreich gespeichert."}

400 Bad Request: Ungültige Daten im Request.

Beispiel: {"error": "Ungültige Daten"}

4. Update von Einstellungen

URL: /updateSettings

Methode: POST

Body:

Beispiel: {"gripperMode": "parallelGripper", "motorSpeed": 75}

Beschreibung: Aktualisiert die Einstellungen des Roboters über MQTT.

Antworten:

200 OK: Einstellungen erfolgreich aktualisiert und publiziert.

Beispiel: {"message": "Einstellungen erfolgreich aktualisiert und publiziert"}

400 Bad Request: Keine Einstellungen gesendet.

Beispiel: {"error": "Keine Einstellungen gesendet"}

500 Internal Server Error: Fehler beim Publizieren der Einstellungen.

Beispiel: {"error": "Fehler beim Publizieren der Einstellungen"}

5. Motorstop

URL: /motors/stop

Methode: POST

Body:

stop: Boolean-Wert, der angibt, ob die Motoren gestoppt werden sollen.

Beschreibung: Sendet ein Signal, um alle Motoren zu stoppen.

Antworten:

200 OK: Motorstopp signalisiert.

Beispiel: {"message": "Motorstopp signalisiert."}

400 Bad Request: Ungültige Daten, erwartet true.

Beispiel: {"error": "Ungültige Daten, erwartet true"}

6. Homing

URL: /homing

Methode: POST

Body:

active: Boolean-Wert, der das Homing aktiviert.

Beschreibung: Aktiviert die Homing-Funktion des Roboters.

Antworten:

200 OK: Homing signalisiert.

Beispiel: {"message": "Homing signalisiert."}

400 Bad Request: Ungültige Daten, erwartet einen Boolean.

Beispiel: {"error": "Ungültige Daten, erwartet einen Boolean"}

7. Manuelle Steuerung der Koordinaten

URL: /manual/control/coordinates

Methode: POST

Body:

coordinates: Array von 4 Elementen, die die Koordinaten darstellen.

Beschreibung: Aktualisiert die Koordinaten für den manuellen Modus.

Antworten:

200 OK: Koordinaten erfolgreich aktualisiert.

Beispiel: {"message": "Koordinaten erfolgreich aktualisiert."}

`400 Bad Request**: Ungültige Koordinaten, erwartet ein Array von 4 Elementen.

Beispiel: {"error": "Ungültige Koordinaten, erwartet ein Array von 4 Elementen"}

8. Manuelle Steuerung der Greiferstärke

URL: /manual/control/gripper

Methode: POST

Body:

gripper: Numerischer Wert, der die Stärke des Greifers angibt.

Beschreibung: Aktualisiert die Greiferstärke für den manuellen Modus.

Antworten:

200 OK: Greifersteuerung erfolgreich aktualisiert.

Beispiel: {"message": "Greifersteuerung erfolgreich aktualisiert."}

400 Bad Request: Ungültige Greiferstärke, erwartet eine Zahl.

Beispiel: {"error": "Ungültige Greiferstärke, erwartet eine Zahl"}

9. Ausführen eines Pick-and-Place-Programms

URL: /pickandplace/program

Methode: POST

Body:

program: Zeichenkette, die den Namen des auszuführenden Programms angibt.

Beschreibung: Übermittelt den Namen eines auszuführenden Programms an den Roboter.

Antworten:

200 OK: Programm erfolgreich übermittelt.

Beispiel: {"message": "Programm erfolgreich übermittelt."}

400 Bad Request: Ungültiger Programmwert, erwartet eine Zeichenkette.

Beispiel: {"error": "Ungültiger Programmwert, erwartet eine Zeichenkette"}

10. Herunterladen des Handbuchs:

URL: /downloadGuide

Methode: GET

Beschreibung: Stellt das Handbuch zum Download bereit.

Antworten:

200 OK: Handbuch erfolgreich heruntergeladen.

404 Not Found: Handbuch nicht gefunden.

500 Internal Server Error: Fehler beim Herunterladen des Handbuchs.

11. Herunterladen des Log-Verzeichnisses:

    URL: /downloadLogs

    Methode: GET

    Beschreibung: Stellt das Log-Verzeichnis als ZIP-Datei zum Download bereit.

    Antworten:

    200 OK: Logs erfolgreich heruntergeladen.

    404 Not Found: Log-Verzeichnis nicht gefunden.

    500 Internal Server Error: Fehler beim Herunterladen der Logs.

12. Löschen des Log-Verzeichnisses:

    URL: /deleteLogs

    Methode: DELETE

    Beschreibung: Löscht das Log-Verzeichnis.

    Antworten:

    200 OK: Log-Verzeichnis erfolgreich gelöscht.

    404 Not Found: Log-Verzeichnis nicht gefunden.

    500 Internal Server Error: Fehler beim Löschen der Logs.

13. Steuerung des Magneten:

    URL: /magnet/control

    Methode: POST

    Body:

    action: String, erwartet "enable" oder "disable".

    Beschreibung: Steuerung des Magneten (ein- oder ausschalten).

    Antworten:

    200 OK: Magnet erfolgreich gesteuert.

    400 Bad Request: Ungültiger oder fehlender Aktionsparameter.

    500 Internal Server Error: Fehler beim Publizieren des Magnetzustandes.

10. WebSockets

Verbindung aufbauen: WebSockets verbinden sich auf demselben Port, auf dem der Server lauscht. Der WebSocket-Client empfängt Zustandsaktualisierungen des Roboters via MQTT.

Events:

connection: Ein Client verbindet sich.

message: Nachrichten von MQTT, z.B. robot/state, werden an verbundene Clients gesendet.

close: Ein Client trennt die Verbindung.

error: Fehlerbehandlung für WebSocket-Verbindungen.