Kurze Dokumentation Website Mini Parallelroboter:

Inhalt

1 Einrichten des Pi‘s als Routed Wireless Access Point

Damit man den Raspberry Pi sinnvoll als Websocket verwenden kann, muss er als „mobiler Hotspot“ konfiguriert werden, damit man sich mit dem Smartphone in sein Netzwerk einwählen kann. Die Anleitung dazu ist im folgenden Text beschrieben oder hier (Raspberry Pi Documentation - Configuration) zu finden.

1.1 Was muss man dabei beachten?

ACHTUNG: Sobald diese Einstellung getätigt sind, ist die Internet Verbindung vom Pi zum Netzwerk (z.B. Hochschul WLAN, euer WLAN daheim etc.) nicht mehr möglich. Per LAN geht die Verbindung noch. Solltet ihr jedoch Internet per WLAN benötigen, müsst ihr die Einstellung Rückgängig machen. Dabei einfach beim letzten Schritt anfangen und in umgekehrter Reihenfolge abarbeiten.

Zum anderen muss beachtet werden, dass die IP Adresse, die ihr dem Netzwerk gebt, nicht bereits vom LAN verwendet wird.

1.2 Benötigte Software

Benötigt wird zum einen *hostpad* und zum anderen *dnsmasq*. *Hostpad* wird wie folgt installiert:

sudo apt install hostapd

Anschließend muss es mit in den Boot Prozess integriert werden, damit es automatisch mitgestartet wird:

sudo systemctl unmask hostapd

sudo systemctl enable hostapd

Danach wird *dnsmasq* installiert:

sudo apt install dnsmasq

Mit ein paar Einstellung für die Firewall:

sudo DEBIAN_FRONTEND=noninteractive apt install -y netfilter-persistent iptables-persistent

Damit wurde die benötigte Software installiert.

1.3 Router Netzwerk aufsetzen

Zuerst wird dem Netzwerk eine IP Adresse gegeben. Dafür muss man die *dhcpcd.conf* öffnen:

sudo nano /etc/dhcpcd.conf

und anschließend folgenden Text eintragen. Dabei wird die IP Adresse gesetzt. :

interface wlan0

static ip_address=192.168.4.1/24

nohook wpa_supplicant

1.4 DNS und DHCP konfigurieren

Für die Konfiguration von DNS und DHCP gibt es eine Menge Möglichkeiten, welche sich bereits in einer Datei befinden, da man aber nicht alle braucht, wird die Datei kopiert und eine neue erstellt und geöffnet:

sudo mv /etc/dnsmasq.conf /etc/dnsmasq.conf.orig

sudo nano /etc/dnsmasq.conf

Anschließend muss folgendes der Datei hinzugefügt werden:

interface=wlan0 # Listening interface

dhcp-range=192.168.4.2,192.168.4.20,255.255.255.0,24h

 # Pool of IP addresses served via DHCP

domain=wlan # Local wireless DNS domain

address=/gw.wlan/192.168.4.1

 # Alias for this router

1.5 Frequenzeinstellung und Netzwerkdefinition

Damit die richtigen Standards für das WLAN nicht vom Raspberry Pi geblockt werden, muss folgender Befehl ausgeführt werden:

sudo rfkill unblock wlan

Anschließend wird das Netz definiert mit Name, Password etc. Dafür wird folgende Datei geöffnet:

sudo nano /etc/hostapd/hostapd.conf

Und folgendes eingetragen:

country_code=GB

interface=wlan0

ssid=MiniParallelRoboterQuattro

hw_mode=g

channel=7

macaddr_acl=0

auth_algs=1

ignore_broadcast_ssid=0

wpa=2

wpa_passphrase=raspberrypi

wpa_key_mgmt=WPA-PSK

wpa_pairwise=TKIP

rsn_pairwise=CCMP

Der Name des Netzwerks ist ***MiniParallelRoboterQuattro*** und das Passwort: ***raspberrypi***. Als country_code ist GB eingetragen, obwohl wir in Deutschland sind. Funktioniert, daher habe ich es nicht geändert. Als letzten Schritt muss der Raspberry Pi gerebootet werden.

sudo systemctl reboot

Nach dem Reboot sollte alles funktionieren.

In Abbildung 1 ist ein QR Code, um sich automatisch mit dem WLAN zu verbinden, falls das Netzwerk den oben beschrieben Namen und Passwort hat.

Abbildung 1 QR Code, um sich mit WLAN zu verbinden

1.6 WLAN Konfiguration für die verschiedenen Roboter:

Damit die Anleitungen funktionieren **müssen** die WLAN Netzwerke wie folgt benannt werden:

6RUS: MiniParallelRoboter6RUS

Quattro: MiniParallelRoboterQuattro

Delta: MiniParallelRoboterDelta

Das Passwort für alle Netzwerke **muss** ***raspberrypi*** heißen.

Für die verschiedenen Namen wurde sich entschieden, damit kein Durcheinander entsteht, wenn mehrere Roboter ein WLAN Netzwerk aufgebaut haben.

2 Steuerung per Website

Um zur Website zu gelangen, einfach *192.168.4.1:5000* in die Adresszeile im Browser eingeben oder den QR Code in Abbildung 2 scannen.

Abbildung 2 QR Code zur Website

2.1 Aufbau der Website

Bei der Website sind initial vier Auswahlmöglichkeiten gestellt. (Siehe Abbildung 3)

Abbildung 3 Initale Auswahlmöglichkeiten

Die initialen Auswahlmöglichkeiten beschreiben die einfachen Modi, die am Roboter häufig aufgerufen werden. Beim Starten des Roboters ist der „Off“ Modus gesetzt, jedoch wird Stop angezeigt (Es muss sowieso am Anfang gehomed werden, daher eigentlich egal). Im „Stop“ Modus sind die Motoren aktiviert, aber der Roboter verfährt nicht. Bei „Demo Programme“ werden die diversen Demo Programme automatisch abgefahren. Bei „Manuelle Steuerung“ verändert sich das Aussehen der Website deutlich.

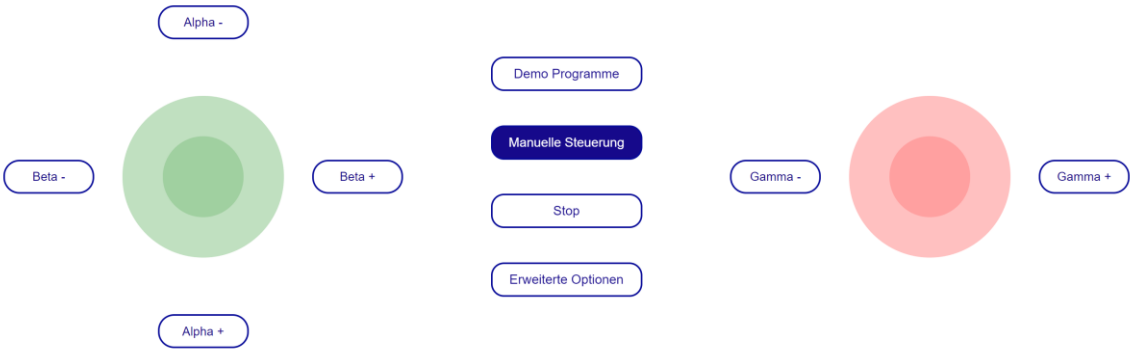


Abbildung 4 Manuelle Steuerung 6 RUS

Bei der manuellen Steuerung erscheinen zwei Joysticks und je nach Roboter bis zu sechs Knöpfe für die rotatorischen Bewegungen des Roboters (Abbildung 4). Beim 6 RUS sind es sechs Knöpfe, beim Quattro sind es zwei Knöpfe und der Delta hat lediglich die zwei Joysticks. Der grüne Joystick ist für das Verfahren in xy-Richtung zuständig. Der rote Joystick ist für das Verfahren in z-Richtung zuständig. Der rote Joystick kann lediglich hoch und runter bewegt werden.

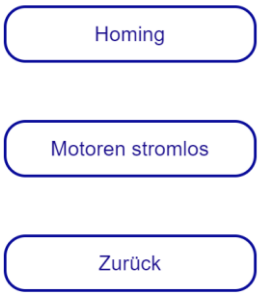


Abbildung 5 Erweiterte Optionen

Unter dem Punkt „Erweiterte Optionen“ erscheint das in Abbildung 5 gezeigt Menü. Dort kann man den Roboter in dem Homing Modus versetzen oder auch die Motoren stromlos schalten.

2.2 Flask Webserver

Um die Website lokal zu hosten, wird Flask verwendet. Das Python Skript dafür heißt „start-Website.py“. Das Skript ist wie folgt aufgebaut.

2.2.1 Initialisierungen und Definitionen

Zu Beginn wird eine Flask App definiert und für diese App CORS aktiviert und konfiguriert. CORS wird benötigt, um Daten von der Website zum Server zu schicken. Danach wird die globale Variable *websiteInformation* definiert. *websiteInformation* ist dict und enthält alle wichtigen Informationen, die für die Steuerung des Roboters benötigt werden. Zum einen den Key „mode“, in dem der aktuelle Modus steht. Danach werden noch die Keys „xCoord“, „yCoord“ und „zCoord“ definiert, welche die Joystick Inputs widerspiegeln. Die Keys „alpha-Plus“, „alphaMinus“, „betaPlus“, „betaMinus“, „gammaPlus“ und „gammaMinus“ enthalten die Inputs der Knöpfe für die rotatorischen Bewegungen des Roboters. Anschließend werden noch zwei Definitionen getroffen, um den Terminal Output zu unterdrücken, wenn jedes Mal eine Website gewechselt wird.

2.2.2 Aufrufen der Websites

Nachdem die Initialisierungen abgeschlossen sind, können die Websites aufgerufen werden. Das Aufrufen der verschiedenen Websites ist nahezu identisch, daher wird lediglich ein Beispiel für die Start- Website (Abbildung 6) gezeigt sowie ein serverseitiges Beispiel zum Senden von Daten von der Website zum Server (Abbildung 7).

```
@app.route("/", methods=['GET', 'POST'])
def home():
    websiteInformation['mode'] = 'stop'
    print(websiteInformation)
    if request.method == 'POST':

        if request.form['btn'] == 'Demo Programme':
            return redirect(url_for('demo'))

        elif request.form['btn'] == 'Manuelle Steuerung':
            return redirect(url_for('steuerung'))

        elif request.form['btn'] == 'Erweiterte Optionen':
            return redirect(url_for('optionen'))

    return render_template('home.html')
```

Abbildung 6 Beispiel der Start Website

In der ersten Zeile (@app.route…) wird der Link eingetragen, der auch später im Browser angezeigt werden soll. Da es die Startseite ist, wird lediglich das Slash benötigt (Bei der Demo Website wäre der Link zum Beispiel „/Demo“). Die Methoden *Get* und *Post* müssen definiert werden, um Daten vom Server zur Website oder von der Website zum Server zu senden. Anschließen wird für die „Route“ eine Funktion definiert, die serverseitig aufgerufen wird, sobald man auf der Website den Link aufruft. Sobald die Funktion aufgerufen wird, wird der aktuelle Modus auf Stop gesetzt und überprüft, ob Änderung von der Website an den Server über die *Post*-Methode an den Server gesendet wurden. Im HTML Dokument der Website sind Submit Buttons definiert, welche den Namen „btn“ tragen und die verschiedene Werte wie z.B. „Demo Programme“ oder „Manuelle Steuerung“ haben. Wird einer der Knöpfe mit dem entsprechend Wert gedrückt, so wird man serverseitig mit dem *return redirect* zur entsprechenden Unterfunktion weitergeleitet. Es wurde bei den Knöpfen für die Auswahl der Modi auf Submit Knöpfe gesetzt, da man damit noch kein fetch-API benötigt, um die Daten von der Website auf den Server zu senden. Wird kein Knopf gedrückt, so wird lediglich mit *return render_template* die HTML Datei *home.html* aufgerufen.

```
@app.route("/Steuerung/inputs/LJS", methods=['GET', 'POST'])
@cross_origin()
def steuerungInputsLJS():

    data = request.get_json(force=True)
    websiteInformation['xCoord'] = (data['distance'] * np.cos(data['radian']))/100
    websiteInformation['yCoord'] = (data['distance'] * np.sin(data['radian']))/100
    print(websiteInformation)
    return NONE
```

Abbildung 7 Beispiel zum Übertragen der Daten von der Website zum Server

Bezieht man jedoch Daten von der Website auf den Webserver ohne Submit Knopf, wird die fetch-API verwendet. Ein serverseitiges Beispiel stellt die Übertragung der Joystick-Inputs dar. Um die fetch-API zu verwenden, muss man neben der Route auch *cross_origin()* für den entsprechenden Pfad definieren. Anschließend werden die Daten über *request.get_json()* von der Website empfangen. Anschließend werden die Daten verarbeitet und in die Variable *websiteInformation* eingetragen.

2.3 HTML Aufbau

Sowohl HTML ohne manuelle Steuerung als auch HTML mit manueller Steuerung sind grundsätzlich gleich aufgebaut. Beide enthalten Knöpfe als Bedienelemente, um den Modus zu wechseln. Die Knöpfe sind als „Submit“ realisiert, damit man ohne fetch-API Daten an den Server senden kann. Ein Nachteil daran ist, dass nach dem Submit die Seite neu geladen wird. (In dem

Fall des Moduswechsel nicht schlimm, aber für spätere Anwendungen (Rotatorische Bewegungen) ungeeignet).

2.3.1 home.html und demo.html

Als Beispiel dient die *home.html*. Das ist die Standardseite, sobald man den QR-Code scannt. Zu Beginn werden im <*head*> die Knöpfe mittels CSS formatiert. Neben den Standardbefehlen sind auch ..*user-select*.. Befehle zu finden. Diese sind dafür da, damit man die Knöpfe während der Benutzung nicht markieren kann. Nachdem die Knöpfe formatiert sind, werden im <*body*> die Knöpfe platziert. Die Zeile <*form method="post" action="/"*> bezieht sich darauf, dass Post- Actions auf den Link „/“ gesendet werden. Danach werden die Submit Knöpfe definiert, welche den Namen *btn* haben und verschiedene Werte wie z.B. *Demo Programme*. Zum Schluss wird noch das HHN-Logo auf die Website mit eingebunden.

2.3.2 optionen.html und offopt.html

Bei den „Erweiterten Optionen“, sprich *optionen.html* und *offopt.html* werden wieder zu Beginn im <*head*> die Knöpfe formatiert. Im <*body*> lautet die erste Zeile jedoch <*form method="post" action="/Optionen"*>, da nun die Post-Action an den Link „/Optionen“ gesendet werden soll. Anschließend werden die Submit Knöpfe mit verschiedenen Werten definiert. Jedoch gibt es hier noch einen Unterschied zu *home.html* und *demo.html*. Bei den Knöpfen *Homing* und *Motoren stromlos* wird noch ein onclick-Event hinzugefügt, welches eine Mitteilung an den Benutzer gibt, dass die Motoren nun stromlos sind und der Roboter neu gehomed werden muss oder dass sich der Roboter gerade im Homing befindet und das man zurück in den Stop Modus gehen soll. Zum Schluss wird wieder das HHN Logo mit eingebunden.

2.3.3 steuerung.html

Die Steuerungs-HTMLs sind von den Modus-Knöpfen und dem HHN-Logo genauso aufgebaut wie das *home.html* oder das *demo.html*. Jedoch gibt es jetzt noch Joysticks und eventuell zusätzliche Knöpfe auf der jeweiligen Website. Von den Joysticks wird lediglich jeder vierte Wert gesendet, da man ansonsten Probleme mit der Auslastung bekommen kann. Die Joysticks sind von https://github.com/yoannmoinet/nipplejs und dort auch recht gut dokumentiert.

2.3.3.1 steuerung_delta.html

Bei der Steuerung für den Delta Roboter werden lediglich zwei Joysticks auf der Website erzeugt, da man keine weiteren Knöpfe für rotatorische Bewegungen benötigt. Zu Beginn werden für die Joysticks zwei „Div“ definiert. Eine für den linken und eine für den rechten Joystick. Diese werden auch im <*head*> formatiert. Es gibt im *head* auskommentiert die *background* Optionen, um sich den Bereich der Div anzeigen zu lassen. Diese Divs und die Formatierung

im *head* sind sehr wichtig, da ansonsten bei iOS Geräten in Safari die Joysticks nicht funktionieren.

Nach der Erstellung der Divs wird der Pfad für den Code der Joysticks hinterlegt. Der Hauptcode liegt unter */static/dist/nipplejs.js* ab. Im *static* Ordner befinden sich noch weitere Funktionen für die Joysticks. Im anschließendem <*script*> ist dann der gesamte Code für Joysticks auf der Website zu finden. Zu Beginn ist ein größerer auskommentierter Teil zu finden, bei dem alle 250 ms eine 0 gesendet wird. Dieser ist eine Notfalllösung, falls der Joystick beim Loslassen nicht automatisch 0 sendet. (In den Tests hat das immer funktioniert, falls es jedoch doch mal Probleme bereiten sollte, ist die Lösung schon da.) Danach wird die Variable *jsLink* definiert. Diese ist besonders wichtig, da bei der fetch-API die Daten an eine Adresse gesendet werden müssen und diese Adresse ist im *jsLink* hinterlegt. Im Anschluss werden noch drei Variablen definiert, die für das Data-Throtteling zuständig sind. Sprich, dass nur jeder 4 Werte gesendet wird. Als letzter Teil der Initialisierung werden nun die Joysticks erstellt. Die Besonderheit für den rechten Joystick ist, dass er den Term lockY enthält. Dadurch kann er nur in Y-Richtung bewegt werden.

Nachdem alles initialisiert wurde, werden nun Event-Bezogen Joystick Daten von der Website an den Server mittels fetch-API gesendet. Bei dem Event *move*, sprich wenn der Joystick bewegt wird, werden in Polar Koordinaten die Position des Joysticks in die Variable *entry* geschrieben. Im Anschluss wird diese Variable über den *fetch*-Befehl an den Server gesendet. Der *fetch()* enthält folgendes: fetch(*Ziellink(z.B. http://192.168.4.1:5000/Steuerung/inputs/LJS)*, {*method: „POST“, body: entry})*. Dabei ist der Ziellink zum großem Teil bereits in *jsLink* definiert, jedoch wird das letzte Kürzel (für den linken Joystick: *LJS* und für den rechten Joystick: *RJS)* erst im fetch() – Befehl hinzugefügt. Der Link ist bereits aus Abbildung 7 bekannt. Die Methode ist immer POST und im body befindet sich die Nachricht, die per fetch() übertragen werden soll. In unserem Fall die Variable *entry* mit den Polarkoordinaten. Sobald der Joystick losgelassen wird, wird ein neues Event aufgerufen. Und zwar werden nun die Polarkoordinaten auf 0,0 gesetzt und an den Server über den fetch() Befehl gesendet. Diese Vorgehensweisen werden für den linken und auch für den rechten Joystick gemacht.

2.3.3.2 steuerung_quattro.html und steuerung_6rus.html

Bei der Steuerung für den Quattro und den 6RUS muss neben der translatorischen Bewegung auch die rotatorische Bewegung gesteuert werden können. Für die translatorische Bewegung werden ebenfalls, wie beim Delta zwei Joysticks verwendet, die wie bereits in 2.3.3.1 beschrieben aufgebaut sind. Die Knöpfe für die rotatorische Bewegung werden diesmal nicht als

Submit, sondern als Button realisiert. Bei den Button werden nicht automatisch Daten an den Server gesendet, daher muss hier wieder auf die fetch-API zurückgegriffen werden.

Am Anfang werden die Knöpfe wieder im <*head*> formatiert und danach im <*body*> aufgerufen. Hier ist eine Besonderheit, dass bei den Buttons eine Funktion hinterlegt ist, sobald sie gedrückt oder losgelassen werden. Dafür müssen jedoch einige Funktionen verwendet werden. Für das Drücken des Knopfes wird: *ontouchstart* und *onmousedown* verwendet (Touch für Mobilgeräte und Mouse für das Benutzen am Computer) und für das Loslassen wird: *ontouchend, ontouchcancel* und *onmouseup* verwendet. (Touchend ist für das Loslassen bei Mobilgeräten, Touchcancel, wenn man mit gedrücktem Finger den Bereich verlässt und dann loslässt und Mouseup ist für das Benutzen am Computer). Bei diesen Events werden die Funktionen *startSend*() beim Drücken ausgeführt und beim Loslassen *stopSend().*

Bei *startSend()* wird die Hintergrundfarbe und die Schriftfarbe des Knopfes verändert, damit man ein visuelles Feedback hat. Dafür benötigt man vorher die Element ID, um die Farbe zu ändern. Anschließend wird die Funktion *sendData()* aufgerufen, welche per fetch eine 1 an den Server sendet.

Bei *stopSend()* wird wieder die Hintergrundfarbe und die Schriftfarbe zurück geändert. Dafür benötigt man erneut die Element ID. Danach wird wieder über *sendData()* per fetch eine 0 an den Server gesendet. Somit ist das Steuern der rotatorischen Bewegung genauso definiert, wie beim Bluetooth Controller.

3 Änderung im Gesamtcode

Durch die Einbindung der Website wurden im Gesamtcode einige Änderung gemacht. Da der Flask Webserver sehr dominant ist und man nach dem Starten des Servers keinen Code mehr ausführen kann, wird nun eine *main.py* erstellt und die alte *main.py* in *startRobot.py* umbenannt. Des Weiteren wurde die Steuerung per Website in *controller.py* und *runtime.py* mitintegriert und auch eine Backup Steuerung per Controller integriert. Der Webserver wird im Programm *startWebsite.py* aufgerufen.

3.1 main.py

In der *main.py* wird nun zu Beginn die Variable *stop_blink* gesetzt, diese für die LED-Ansteuerung zuständig und die Variable *robotType*, welche das aktuelle Robotermodell beschreibt. Im Anschluss wird noch *threading* importiert, da die Roboter Steuerung nun in einem eigenen Thread läuft. Ist die *main.py* das ausgeführte Programm, so werden *startWebsite* und *startRobot* importiert. Diese werden erst dann importiert, damit keine circular calling von imports erzeugt wird. Dann wird der Roboter Thread gestartet und im Anschluss die Website gestartet.

3.2 startRobot.py

Bei der *startRobot.py* wurde die Funktion *startRobot()* erzeugt, die in der *main.py* als Thread aufgerufen werden kann. Diese Funktion enthält alles, was das Frühere main.py auch enthalten hat.

3.3 runtime.py

In der *runtime.py* wurde im Loop die *homing()* Funktion herausgenommen und mit der *calibrate()* Funktion zusammen geführt. Des Weiteren wurden viele Abfragen eingebaut, ob aktuell die Website zur Steuerung verwendet werden soll oder Controller. Ist kein Controller verbunden, so wird die Website verwendet. Ist ein Controller verbrunden, so wird er verwendet, solange er verbunden ist. Um den PS5 wieder auszuschalten, muss die PS-Taste für 10 Sekunden gedrückt werden.

3.4 controller.py

In *controller.py* wurden neben den Controller Funktionen jetzt auch Funktionen für die Benutzung per Website eingebunden. Die Website Funktionen wurden analog zu den Controller Funktionen erstellt, damit man in der *runtime.py* ohne große Umstellung mit den Inputs arbeiten kann. Dabei enthalten die Funktionsnamen in *controller.py* jetzt immer ein *ws* oder ein *controller/cont*, um zu definieren, worauf sich die Funktion bezieht.

4 Offene Punkte:

1. Website auf dem Delta/ 6RUS miteinbinden und testen
2. LED-Ansteuerung auch für Website Inputs mit eingliedern
3. Step_delay in *runtime.py* für Delta/ 6RUS anpassen
4. Demo Programme für den Delta/ 6RUS anpassen
5. Limits in *controller.py* für die manuelle Steuerung für den Delta/ 6RUS anpassen
    a. Eventuell auch eine Arbeitsraumberechnung mit genauer Beschränkung
6. Optional:
    a. style.css für die HTMLs erstellen, damit nicht die Knöpfe in jeder HTML-Datei immer fomatiert werden müssen

Mini Parallel Roboter Quattro

Bedienungsanleitung:

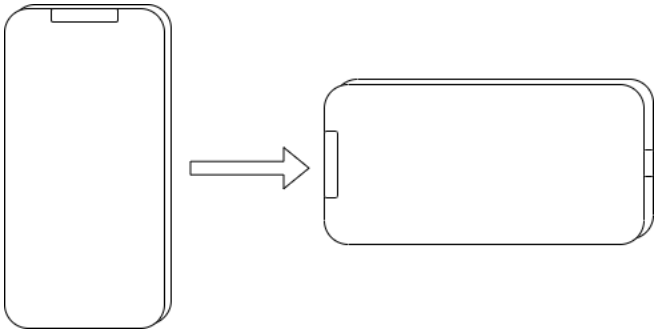
Schritt 1 Mit WLAN verbinden:

- Kamera auf Ihrem Smartphone öffnen und QR-Code einscannen

Schritt 2 Website aufrufen:

- Kamera auf Ihrem Smartphone öffnen und QR-Code einscannen

Schritt 3 Handy um 90° drehen und die Website erkunden:



Mini Parallel Roboter Delta

Bedienungsanleitung:

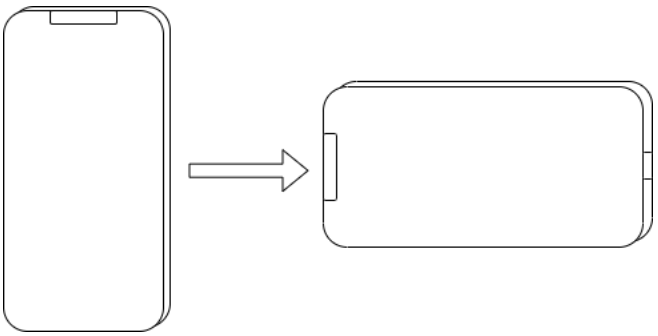
Schritt 1 Mit WLAN verbinden:

- Kamera auf Ihrem Smartphone öffnen und QR-Code einscannen

Schritt 2 Website aufruten:

- Kamera auf Ihrem Smartphone öffnen und QR-Code einscannen

Schritt 3 Handy um 90° drehen und die Website erkunden:



Mini Parallel Roboter 6RUS

Bedienungsanleitung:

Schritt 1 Mit WLAN verbinden:

- Kamera auf Ihrem Smartphone öffnen und QR-Code einscannen

Schritt 2 Website aufrufen:

- Kamera auf Ihrem Smartphone öffnen und QR-Code einscannen

Schritt 3 Handy um 90° drehen und die Website erkunden: