

Einführung

Arduino® Alvik ist ein leistungsstarker und vielseitiger Roboter, der speziell für die Programmierung und STEAM-Ausbildung entwickelt wurde.



Angetrieben durch den Arduino® Nano ESP32 bietet Alvik verschiedene Lernpfade durch unterschiedliche Programmiersprachen wie MicroPython, Arduino C und blockbasiertes Programmieren. Dies ermöglicht verschiedene Möglichkeiten, Robotik, IoT und künstliche Intelligenz zu erkunden.

In diesem Tutorial finden Sie nützliche Informationen, um Alvik zu starten, zu testen und zu pflegen.

Was ist Alvik

Alvik ist ein Roboter mit zwei Controllern und einer breiten Palette von Sensoren und Aktuatoren. Der Hauptcontroller ist der Arduino Nano ESP32, der an der Oberseite des Roboters angebracht ist, während ein STM32-Controller in den Roboter integriert ist, der die Low-Level-Befehle wie das Lesen der Sensoren und das Bewegen der Motoren übernimmt. Er wurde für den STEAM-Unterricht entwickelt und ist somit ein ideales Werkzeug zum Erlernen von Programmierung, Robotik, IoT, künstlicher Intelligenz...

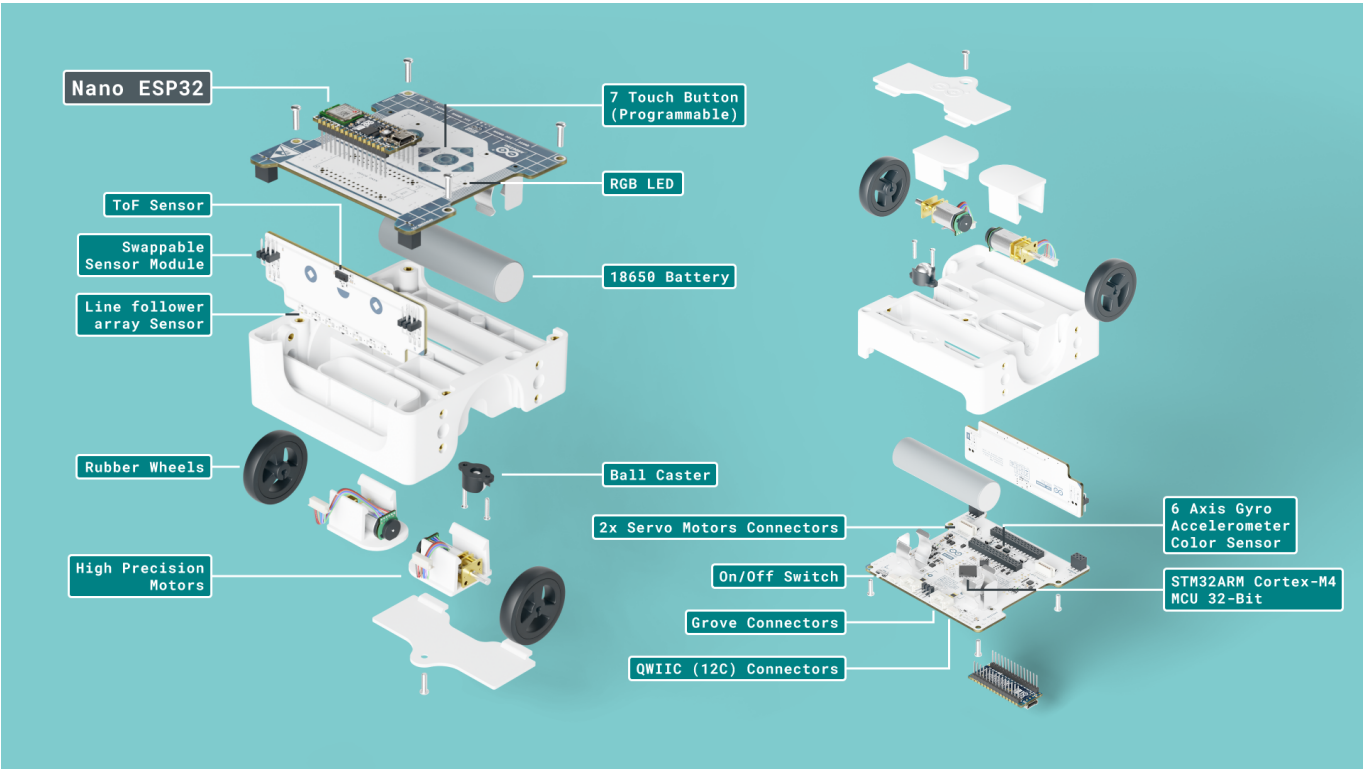
Unboxing



In dem Arduino Alvik-Paket finden Sie die folgenden Artikel:

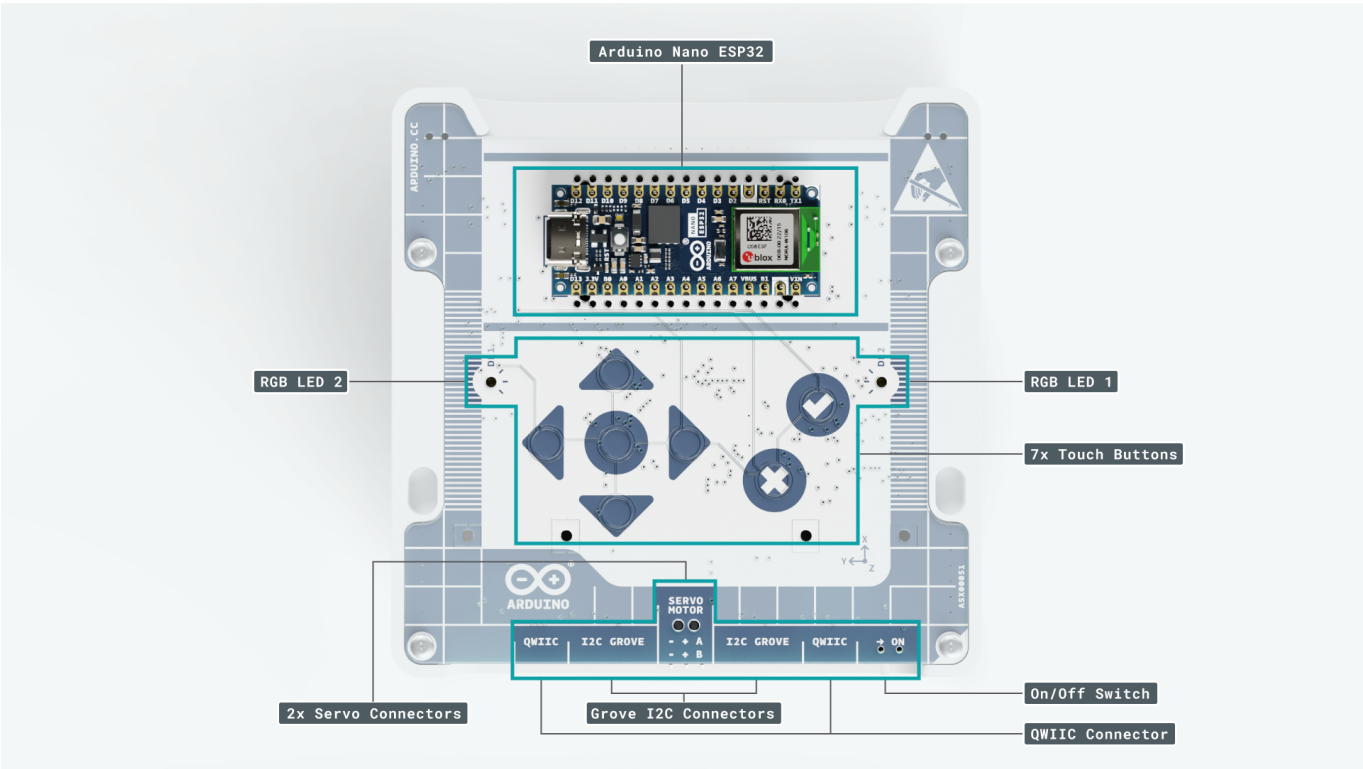
- Alvik Roboter
- 18650 Li-Ion-Akku, der am Roboter befestigt ist
- USB-C® auf USB-C® Kabel

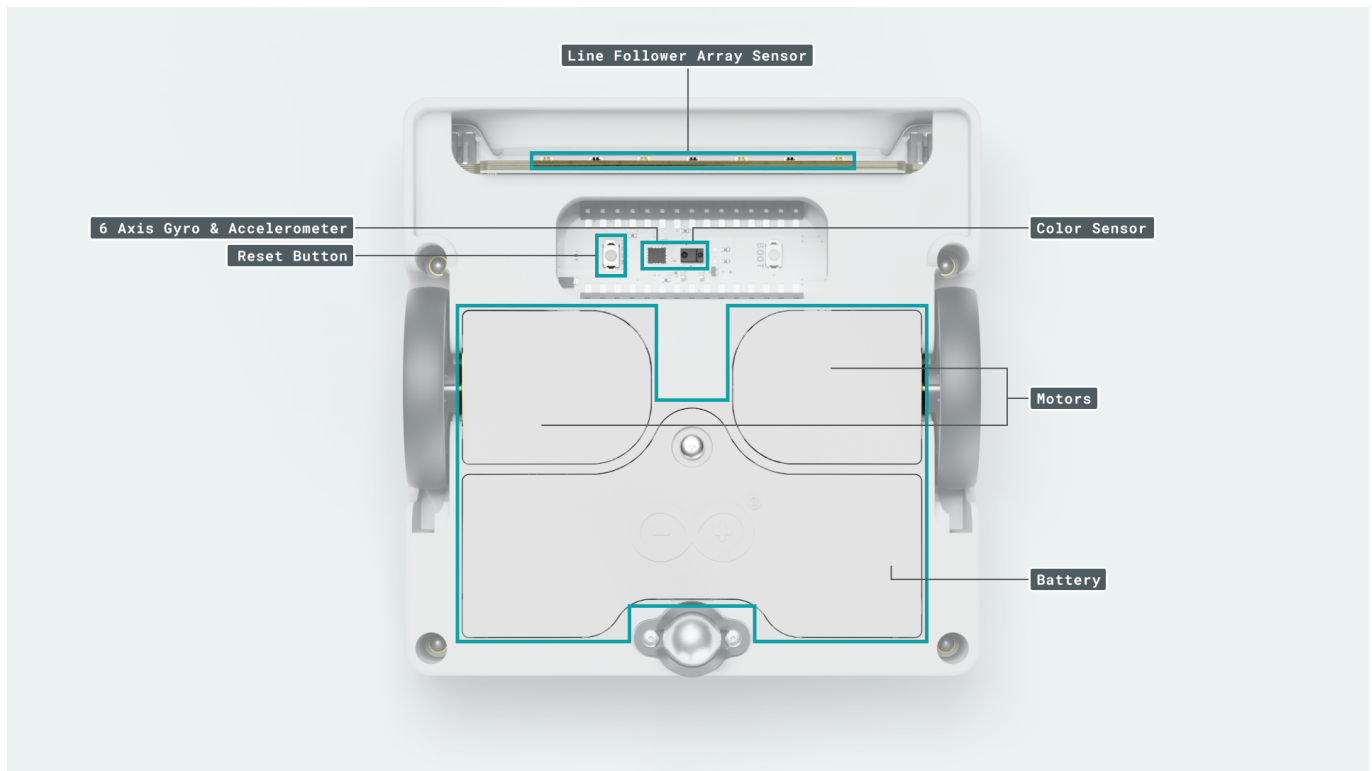
Alvik Übersicht



Wenn die Platine aus dem Gehäuse genommen und der Akku eingesetzt ist, besteht die Gefahr eines Kurzschlusses des 18650 Li-Ion-Akkus. Wenn Sie die Hardware aus dem Gehäuse entfernen, stellen Sie sicher, dass Sie dies auf einer nicht leitenden Oberfläche tun, die frei von Materialien oder Werkzeugen ist, die den Akku kurzschließen könnten.

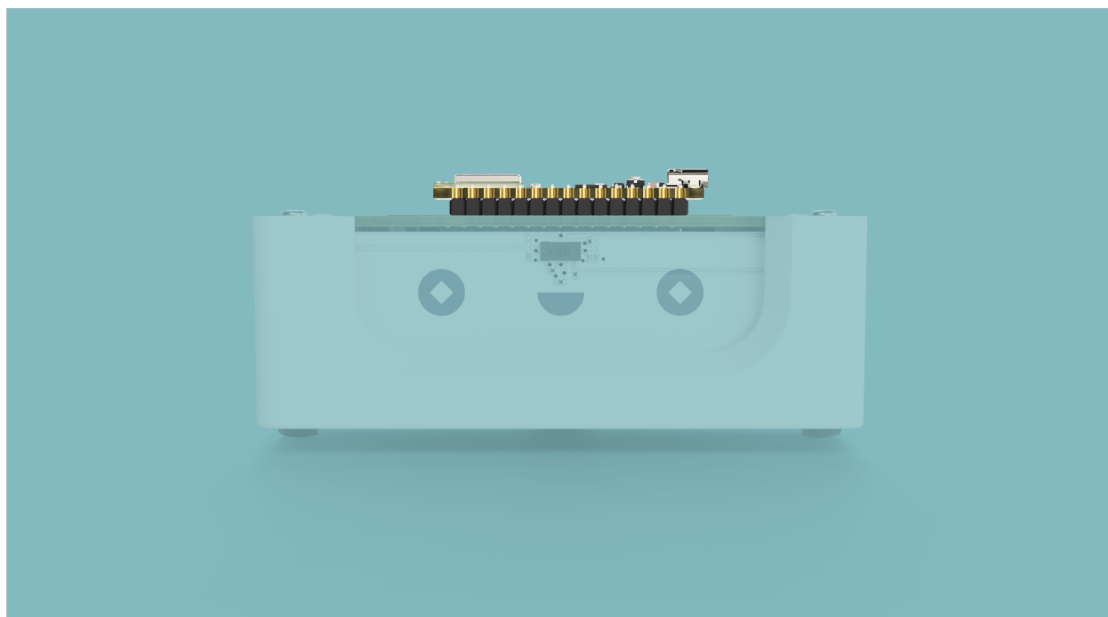
Hauptkomponenten





Alviks Gehirn

Der **Nano ESP32** ist das Gehirn des Alvik-Roboters. Er fungiert als zentrale Schaltstelle, die die Logik steuert und wichtige Entscheidungen trifft. Er spielt zwar eine wichtige Rolle bei der Verarbeitung von Befehlen auf hoher Ebene und bei der Überwachung des Roboterbetriebs, steuert aber nicht direkt die Motoren oder Sensoren. Stattdessen kommuniziert der Nano ESP32 mit dem **STM32** (dem Körper) und sendet Befehle für Aktionen wie Bewegung, Sensormessungen oder Statusaktualisierungen. Diese Zusammenarbeit ermöglicht es dem Roboter, Aufgaben und Routinen auf der Grundlage der empfangenen Daten auszuführen und so fundierte Entscheidungen zu treffen, wie wir sie programmiert haben.



Nano ESP32

Der **Nano ESP32** ist das Board, mit dem Sie Alvik steuern. Er verfügt über einen schnellen Prozessor, einen großen Flash-Speicher und einen Wi-Fi®-fähigen Chip auf einer winzigen Schaltung.

Mehr über dieses Board erfahren Sie in der [Nano ESP32 Dokumentation](#).

Bitte beachten Sie, dass bei der Verwendung von MicroPython die Pin-Nummern den GPIO auf dem ESP32-S3 entsprechen, nicht dem Nano-Board. Verwenden Sie die **grün beschriftete Nummer** im folgenden Bild. Sie können mehr darüber [hier](#) lesen.

Nano / ESP32

Pinout

The diagram shows the Nano ESP32 board with its pin headers. The pins are color-coded: green for digital pins (D0-D13, D18-D19), orange for analog pins (A0-A7), and grey for boot pins (BOOT0, BOOT1). The ESP32 pin numbers are listed in green boxes, and the Nano pin numbers are listed in orange boxes.

Nano Pin	ESP32 Pin
48	D13
46	BOOT0
1	A0
2	A1
3	A2
4	A3
11	A4
12	A5
13	A6
14	A7
0	BOOT1
D12	47
D11	38
D10	21
D9	18
D8	17
D7	10
D6	9
D5	8
D4	7
D3	6
D2	5
D0	44
D1	43

i ESP32 pin numbers

i Nano pin numbers

Alviks Körper

Mit der gleichen Analogie umfasst der Körper des Alvik-Roboters alles andere, von den Sensoren und Motoren bis hin zu den Anschlüssen, die externe Module unterstützen, und natürlich den STM32, der wie das Nervensystem des Roboters funktioniert. So wie das menschliche Nervensystem Signale zwischen dem Gehirn und den verschiedenen Teilen des Körpers überträgt, so überträgt der STM32 Befehle und sammelt Daten von

den Komponenten des Roboters. Wenn der Nano ESP32, der als Gehirn fungiert, Befehle sendet, führt der STM32 diese Low-Level-Operationen aus, wie z.B. die Bewegung der Motoren, das Blinken der LEDs oder die Interaktion mit externen Geräten. Der STM32 ist für die Ausführung der detaillierten Aktionen auf der Grundlage der High-Level-Anweisungen des Nano ESP32 unerlässlich und sorgt dafür, dass die Bewegungen und Funktionen des Roboters präzise ausgeführt werden.



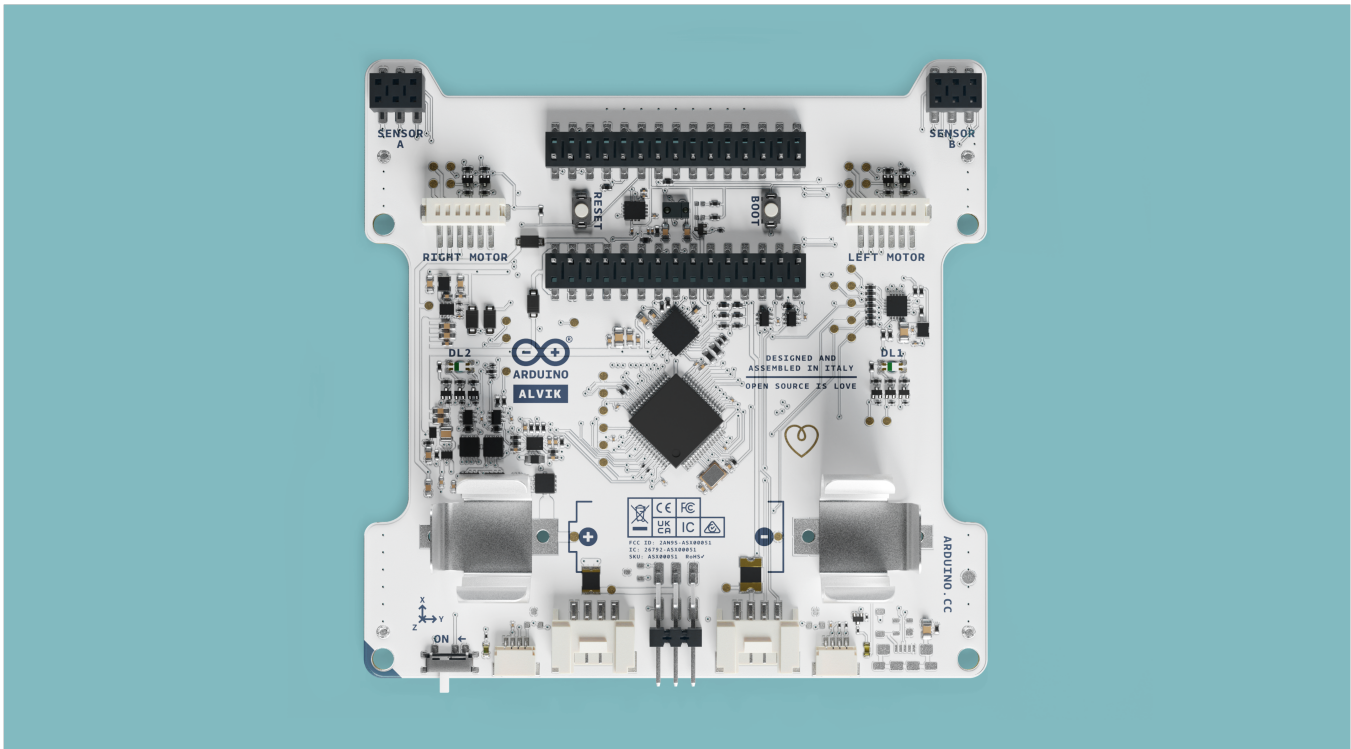
Controller - STM32

Das Herzstück des Roboters ist der STM32 ARM Cortex-M4 32 Bit Controller. Er verarbeitet die Low-Level-Befehle wie das Lesen von Sensoren und das Bewegen von Motoren. Sie können ihn über eine Reihe spezieller APIs vom Nano ESP32 aus ansprechen.

Mehr über die verfügbaren Funktionen für Alvik erfahren Sie in der folgenden [Alvik's API Documentation](#).

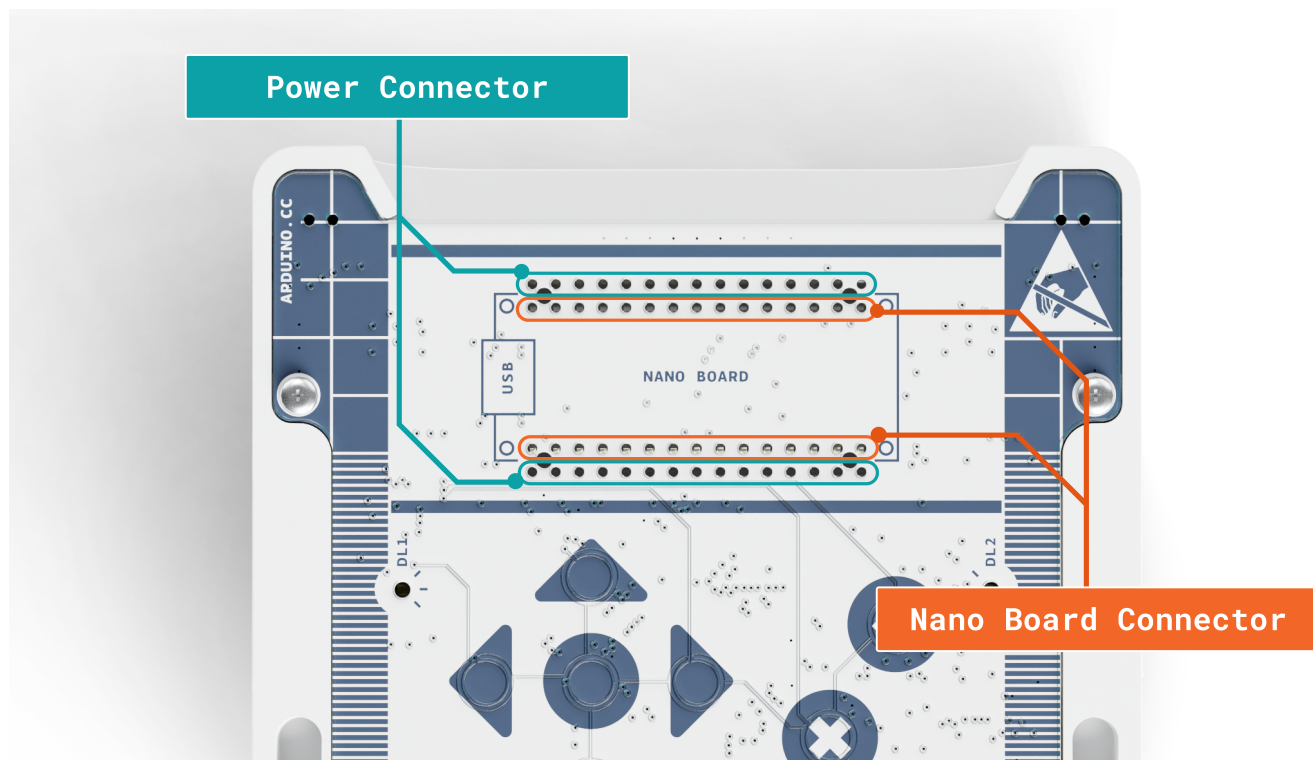
Obwohl Alviks "out of the box" Erfahrung in MicroPython entwickelt wurde, gibt es C++ APIs, die es Ihnen ermöglichen, Alvik in der Arduino IDE zu programmieren. Diese APIs sind 1 zu 1 kompatibel mit den MicroPython APIs, was bedeutet, dass die [Alvik's API Documentation](#) Ihnen Informationen sowohl für MicroPython als auch für C++ Programmiererfahrungen liefert.

Die neueste Firmware des STM32 finden Sie unter [diesem Link](#), und [hier](#) ist die Anleitung zum Flashen der Firmware.



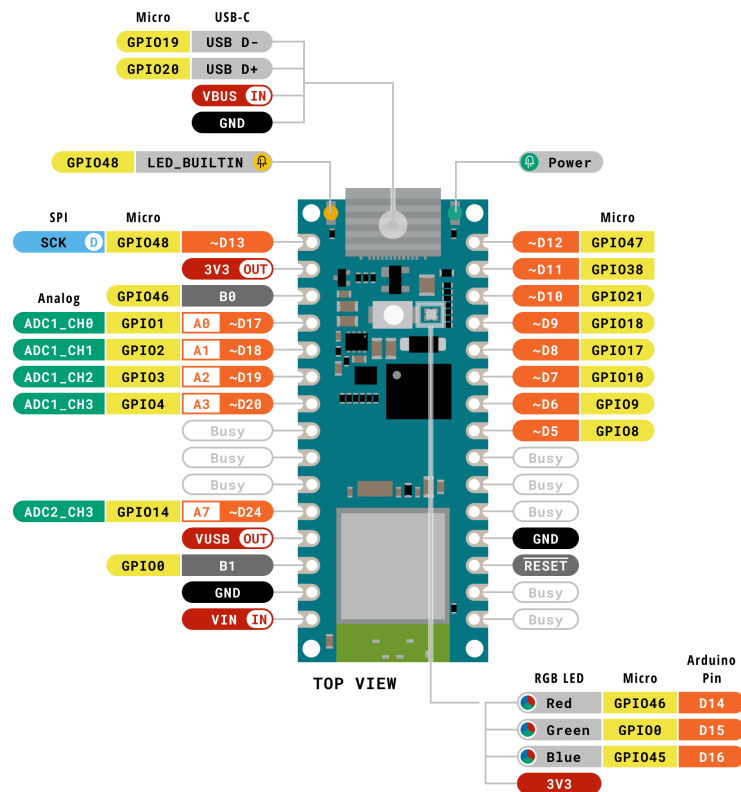
Header-Anschlüsse

Auf der Oberseite des Alvik finden Sie 2 Paare von Anschlüssen. Beide haben 15 Löcher in der Länge, sind aber nicht miteinander verbunden. Das Paar auf der Innenseite ist für die Kopplung des Arduino Nano ESP32 gedacht, während das Paar auf der Außenseite der Power Connector ist, der optionalen Strom und mechanische Unterstützung für benutzerdefinierte Erweiterungen oder Schaltungen bietet, die Sie möglicherweise erstellen.



- **Nano Board Connector**

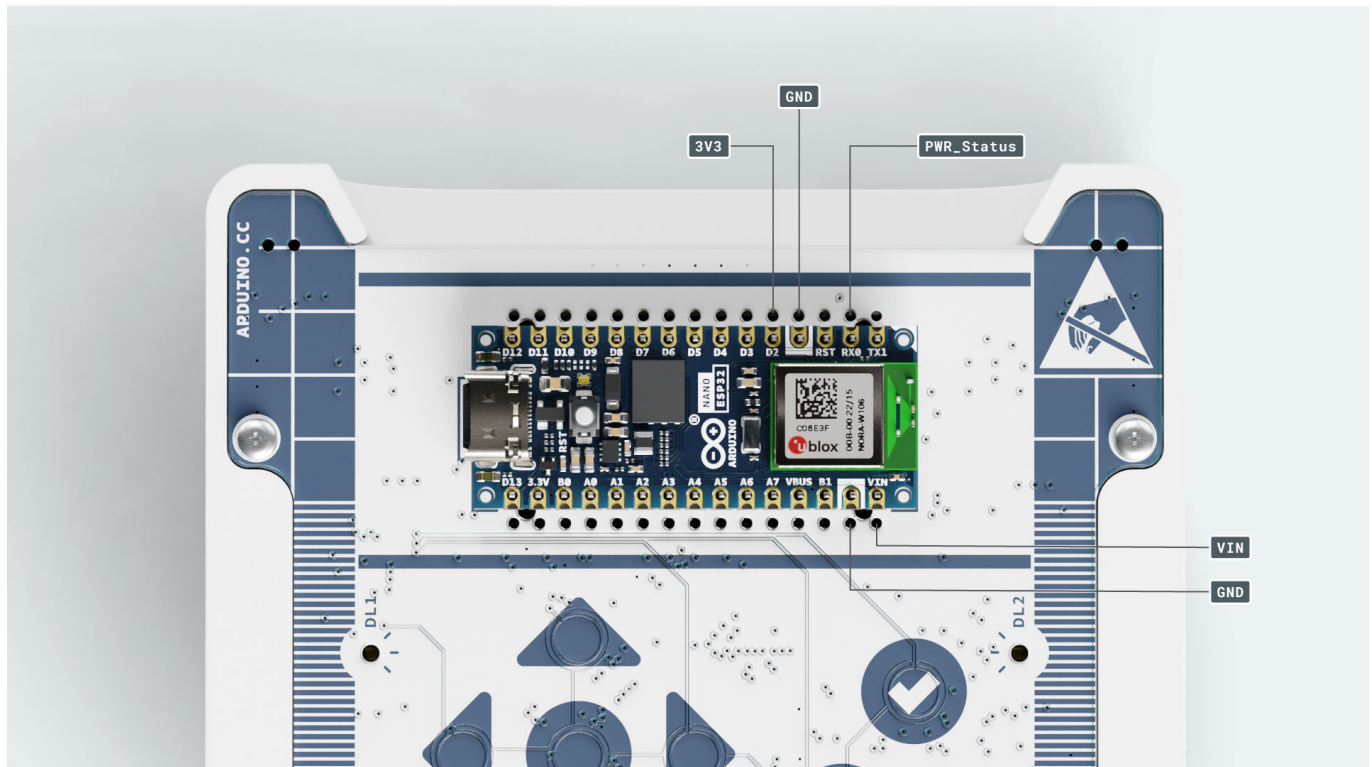
Aufgrund der Notwendigkeit, mit dem STM-Board zu kommunizieren, können einige der Pins des Arduino Nano ESP32 belegt sein und sollten daher nicht für andere Funktionen verwendet werden. Wenn Sie also eine Abschirmung oder ein Zubehöriteil entwerfen, das mit diesen Anschlüssen interagiert, stellen Sie sicher, dass Sie die belegten Pins vermeiden.



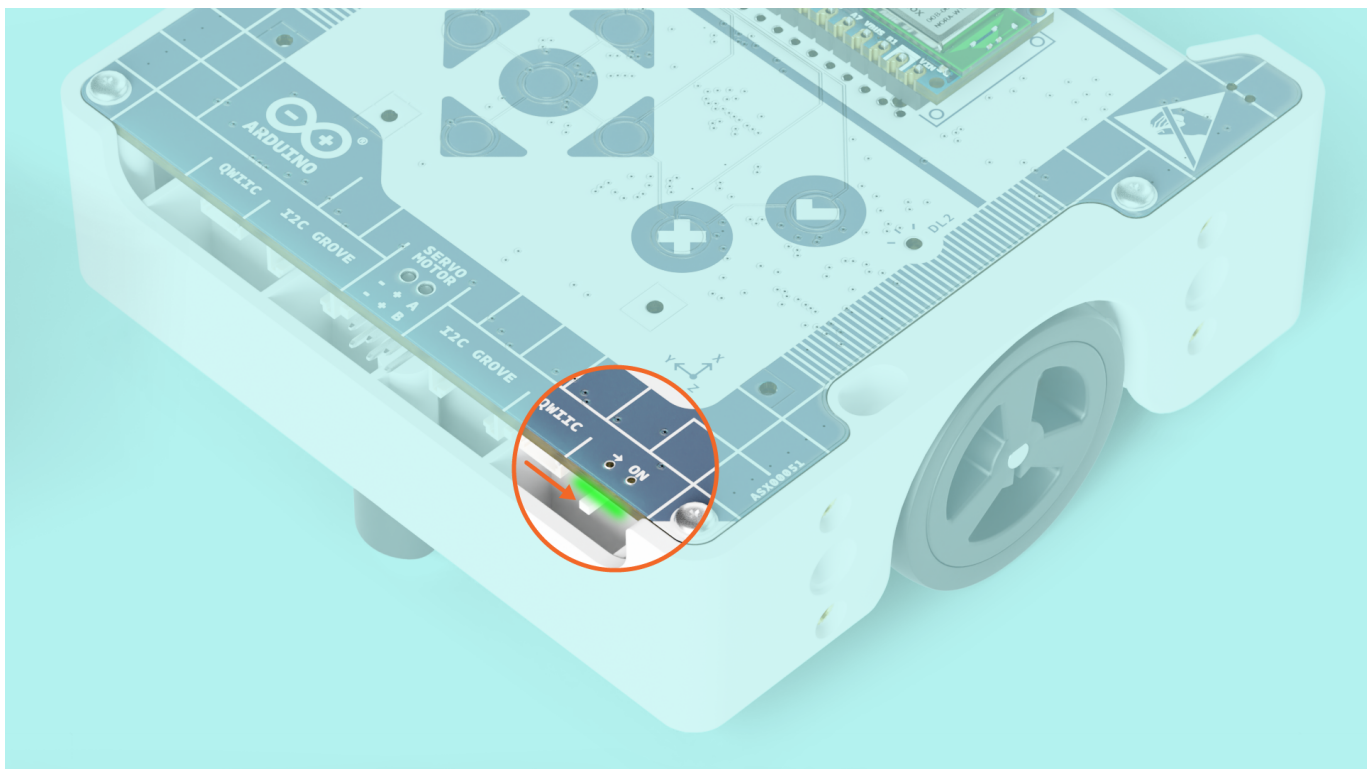
• Stromanschluss

Über diesen Anschluss können Sie das Alvik-Stromsystem mit Strom versorgen und den Stromstatus abfragen. Außerdem bietet er mechanische Unterstützung für Ihre Designs. Hier sehen Sie die Anschlüsse, die verwendet werden können. Beachten Sie, dass die verbleibenden Pins nicht verwendet werden, aber zur

mechanischen Unterstützung dienen können.

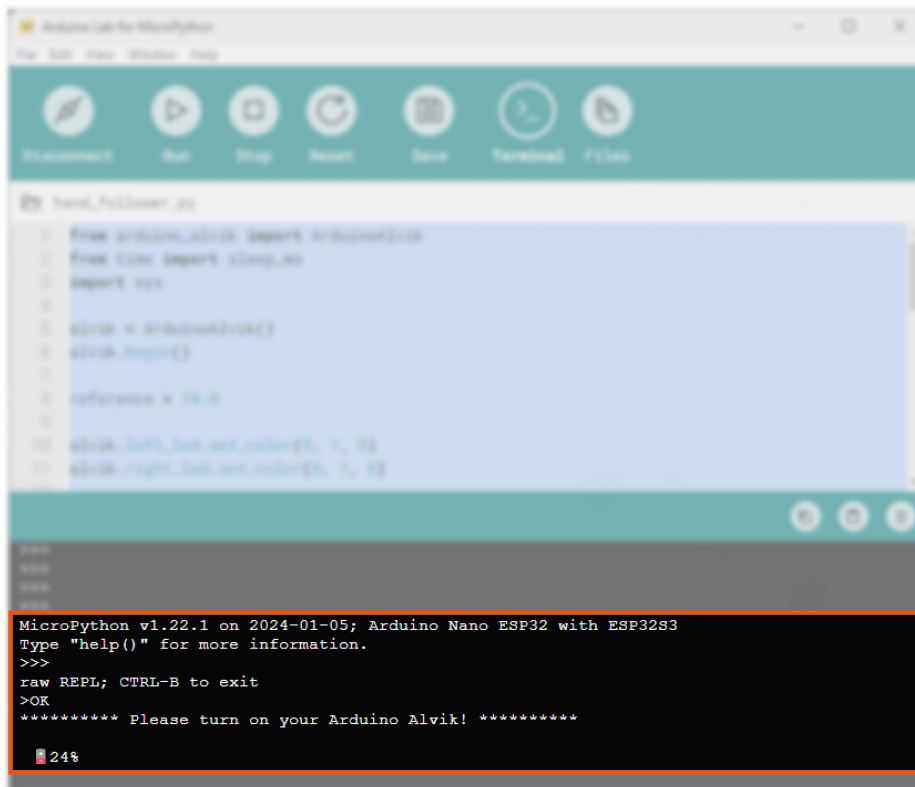


Ein/Aus-Schalter



An der hinteren rechten Seite des Alvik befindet sich der Hauptschalter des Roboters. Wenn er eingeschaltet ist, fährt der Roboter hoch und führt das bereits geladene Programm aus.

Halten Sie den Roboter während der Programmierung ausgeschaltet, um unerwünschte Bewegungen zu vermeiden, und denken Sie daran, ihn wieder einzuschalten, wenn Sie bereit sind, Ihr Programm auszuführen. Eine Funktion des Arduino Lab for MicroPython Terminals ist, dass es Sie benachrichtigt, wenn Sie vergessen haben, den Roboter einzuschalten, um den Code auszuführen.

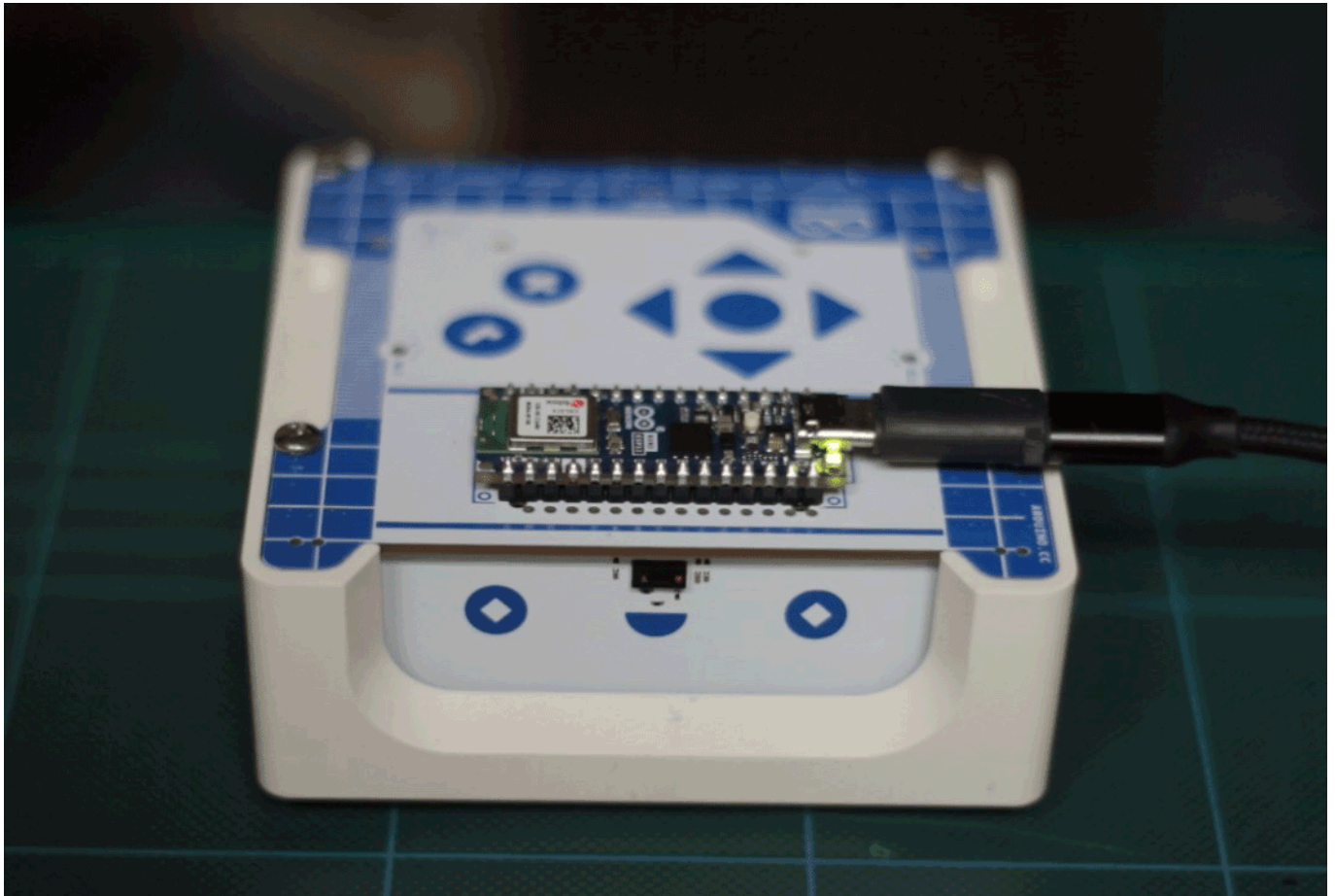


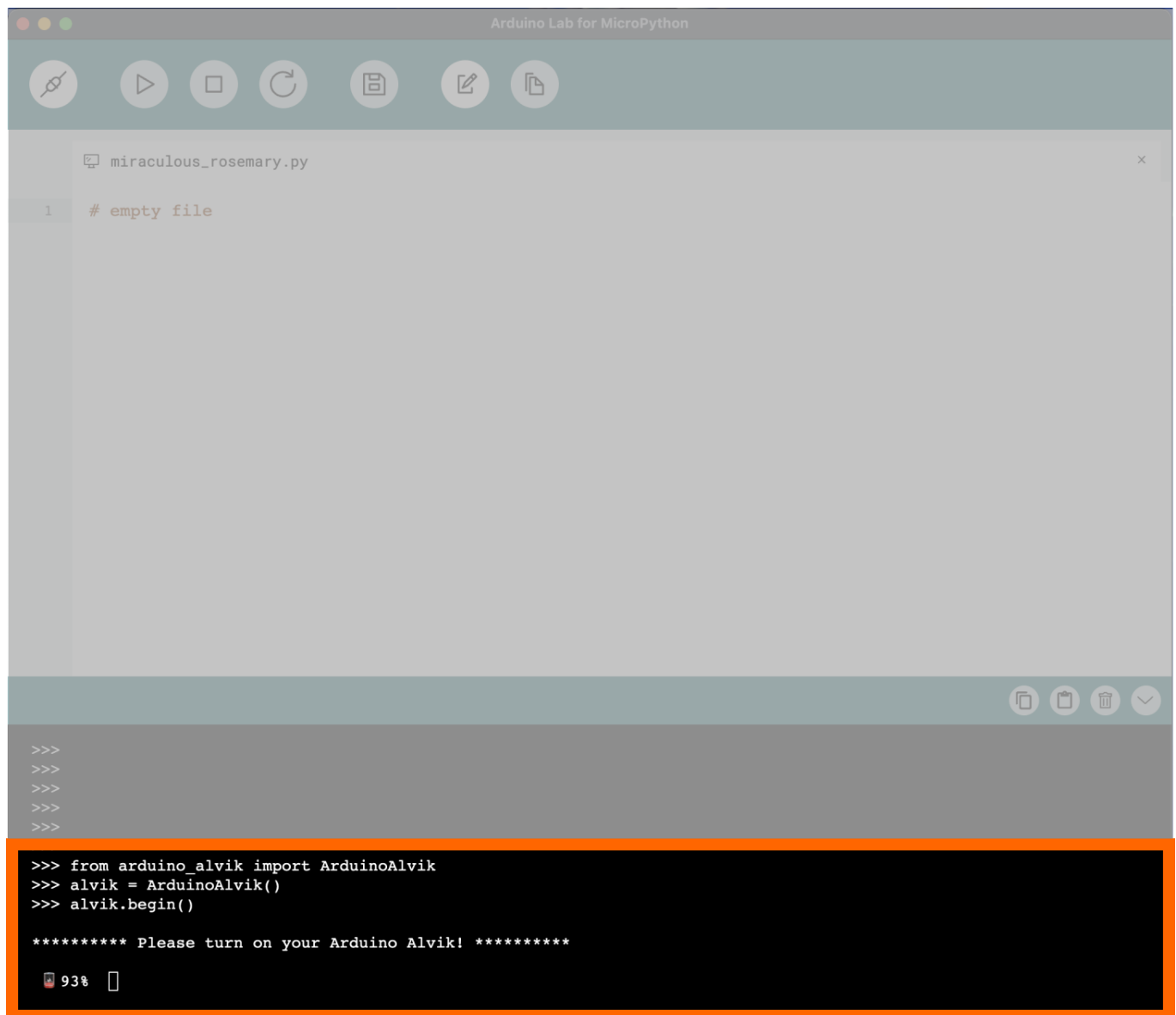
Batterie

Der Akku ist ein wiederaufladbarer Lithium-Ionen-Akku 18650, mit dem das Alvik über 8 Stunden nonstop betrieben werden kann.

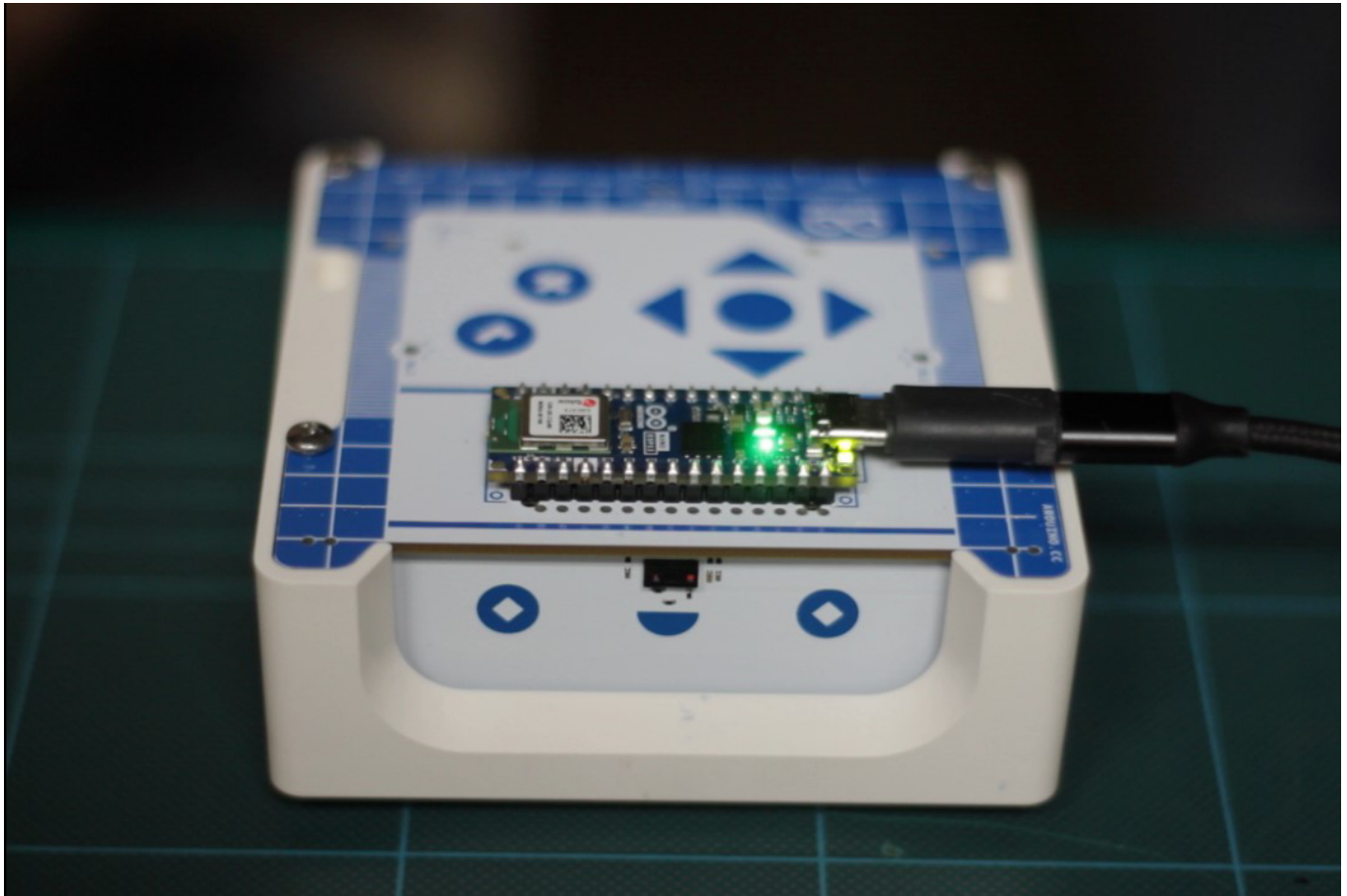
Wenn Sie Ihr Alvik an den Computer anschließen, meldet der Nano ESP32 den Status der Batterie über das Terminal des Arduino Lab for MicroPython und über seine RGB-Status-LED. Um den Zustand der Batterie zu sehen, müssen Sie die Funktion `Alvik.begin()` in einem beliebigen Programm oder direkt im Befehlszeilenbereich aufrufen.

Wenn der Akku geladen wird, blinkt die Status-LED eine Sekunde lang ROT.





Wenn sie vollständig geladen ist, bleibt sie GRÜN.



Verwechseln Sie die RGB-Status-LED nicht mit der Power ON-LED des Nano ESP32, die immer grün ist.

Um die Lebensdauer des Akkus Ihres Alviks zu erhalten, bewahren Sie ihn immer vollständig aufgeladen auf.

Wir empfehlen, den Alvik zwischen den Arbeitssitzungen vollständig aufzuladen. Wenn Sie das Gerät über einen längeren Zeitraum nicht benutzen, nehmen Sie den Akku am besten heraus, bis Sie ihn wieder verwenden.

Wenn der Alvik mehrere Monate lang nicht benutzt wird und der Akkustand unter 20% liegt, kann der Akku vollständig entladen werden, was zu einem Spannungsabfall führt. In solchen Fällen muss der Akku mit einem kompatiblen Ladegerät extern aufgeladen werden.

Ersetzen der Batterie

Die Batterie befindet sich im unteren Teil des Alvik. Um an sie heranzukommen, müssen Sie eine Phillips-Schraube entfernen und die Kunststoffhalterung herausnehmen.

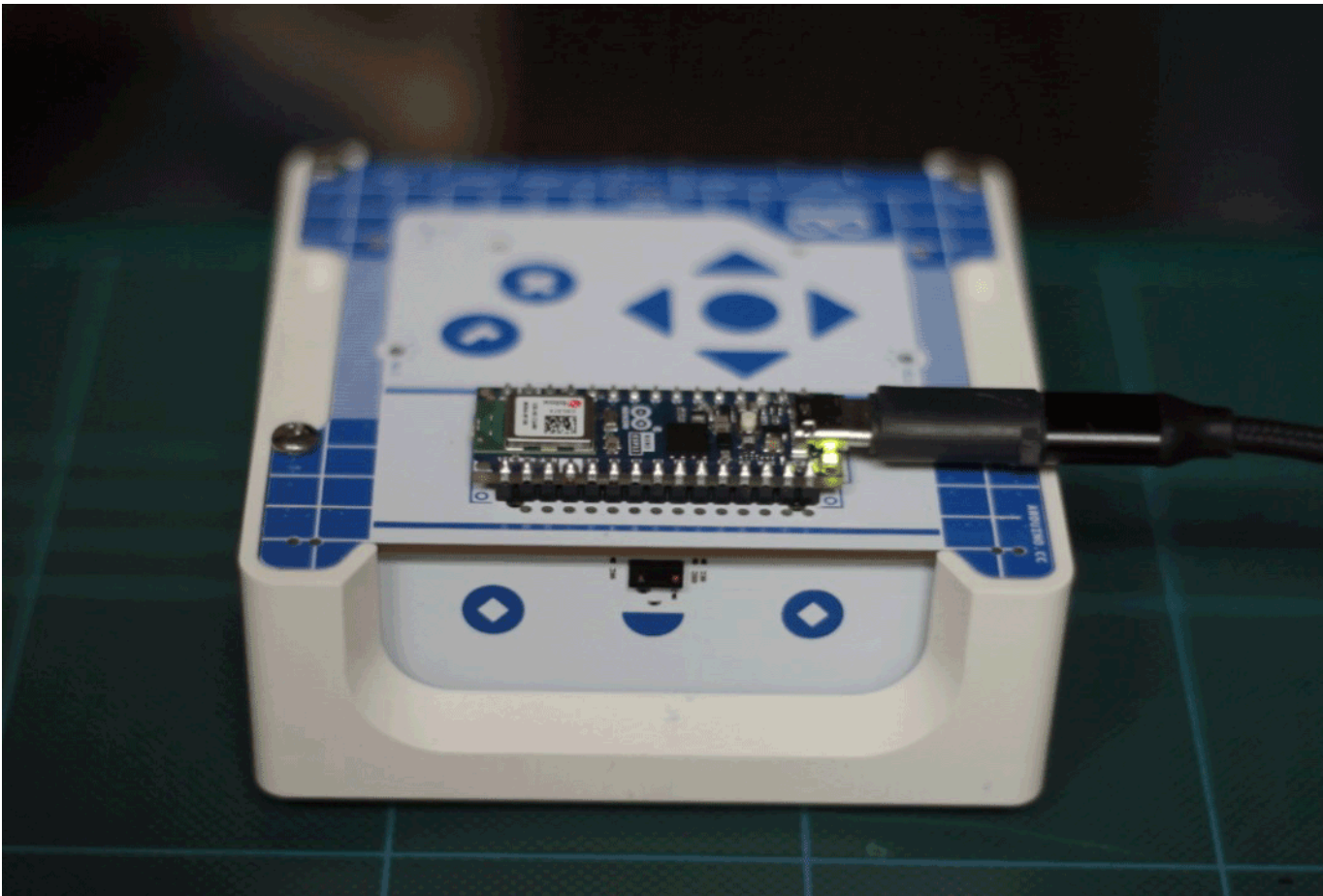
Zugriff auf das Batteriefach](assets/battery_holder.jpg)

Führen Sie die folgenden Schritte aus, um die Batterie von Alvik zu ersetzen und das Batteriemanagementsystem (BMS) zurückzusetzen:

1. **Schalten Sie Alvik AUS.**
2. **Entnehmen Sie die vorhandene Batterie:** Nehmen Sie die alte Batterie vorsichtig aus ihrem Fach.
3. **Ziehen Sie die Klemmen des Batteriehalters an,** indem Sie die + und - Pole leicht zur Mitte ziehen, um sicherzustellen, dass sie einen guten Kontakt mit den Batteriepolen haben.
4. **Legen Sie die neue Batterie ein:** Legen Sie die neue Batterie in die Halterung ein und vergewissern Sie sich, dass sie fest angeschlossen ist. Um den ordnungsgemäßen Kontakt zu überprüfen, messen Sie mit

einem Multimeter (auf 20 V eingestellt) die Klemmen des Batteriehalters (+ & -); es sollte die Spannung der Batterie anzeigen.

- 5. **Verbinden Sie den Nano ESP32 über das USB-Kabel mit Ihrem Computer**, während der Alvik-Schalter in der Position OFF steht.
- 6. **Schalten Sie Alvik ein**, wie in der Abbildung unten dargestellt.



Hinweis: Wenn Alvik wieder eingeschaltet wird, erlischt die Lade-LED-Anzeige.

Alviks Sensoren

Alvik enthält eine Reihe verschiedener Sensoren, die unten aufgeführt sind. Sie sind alle mit dem STM32 verbunden und über die [APIs](#) zugänglich. Für jeden Sensor gibt es ein Test-Beispielprogramm, das Sie im Ordner *examples* in [this repository](#) für MicroPython und in [this repository](#) für C++ finden können.

Sensorname	Teilname	Testprogrammname
RGB Color detection	APDS 9660	read_color_sensor.py
ToF 8x8 Array - up to 350 cm	LSM6DSOX	read_tof.py
IMU - 6 degree	VL53L7CX	read_imu.py
3x Line follower	custom made	line_follower.py
7x Touch sensor	AT42QT2120	read_touch.py

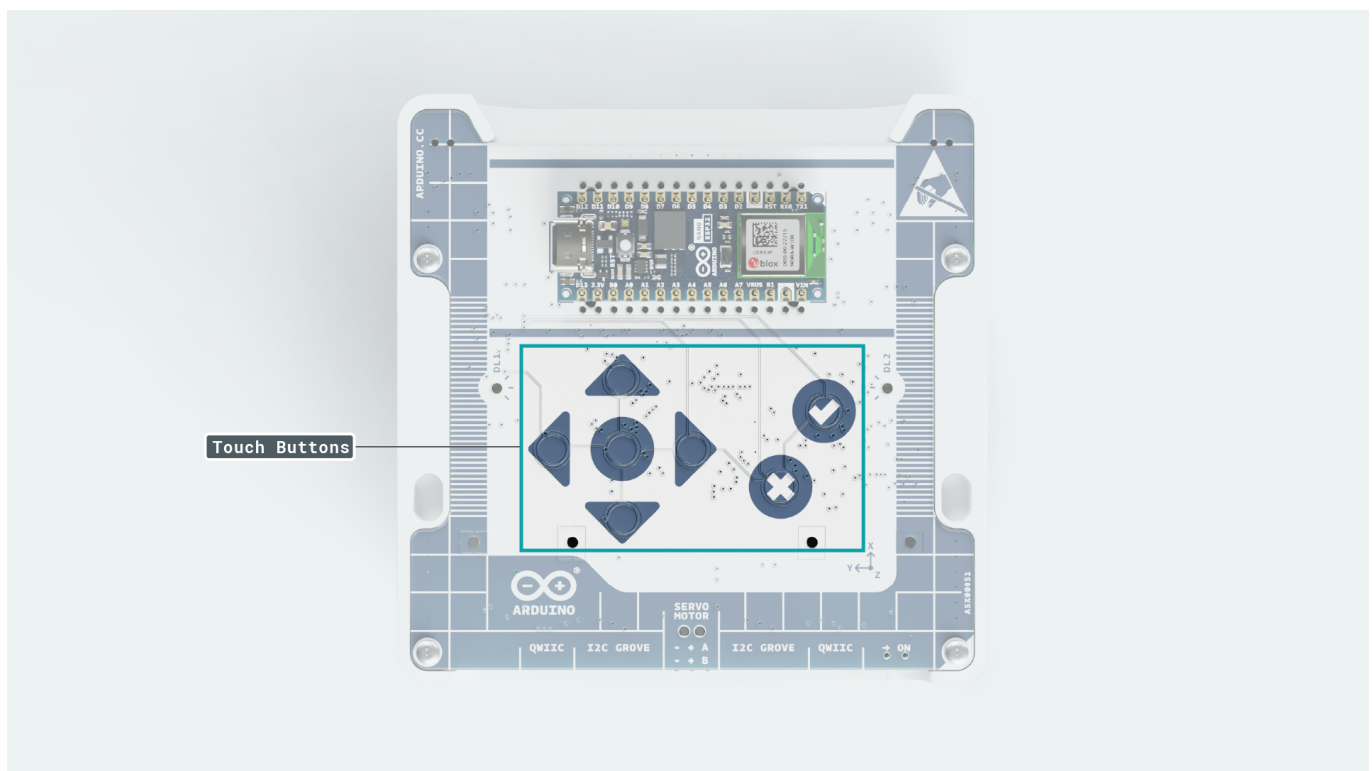
Prüfen Sie vor der Verwendung des ToF-Sensors, ob er eine gelbe Schutzfolie hat. Falls vorhanden, entfernen Sie diese vom Sensor, um sicherzustellen, dass er ordnungsgemäß funktioniert.

Tasten berühren

Der Arduino Alvik Roboter ist mit sieben kapazitiven Touch-Tasten ausgestattet. Diese Tasten ermöglichen interaktive Eingaben und können verwendet werden, um verschiedene Funktionen des Roboters zu steuern, wie z.B. Bewegung, Modusauswahl und mehr.

Verfügbare Tasten sind:

- Aufwärts
- Unten
- Links
- Rechts
- Zentrieren
- OK (dargestellt durch ein "✓"-Symbol)
- Abbrechen (dargestellt durch ein "x")



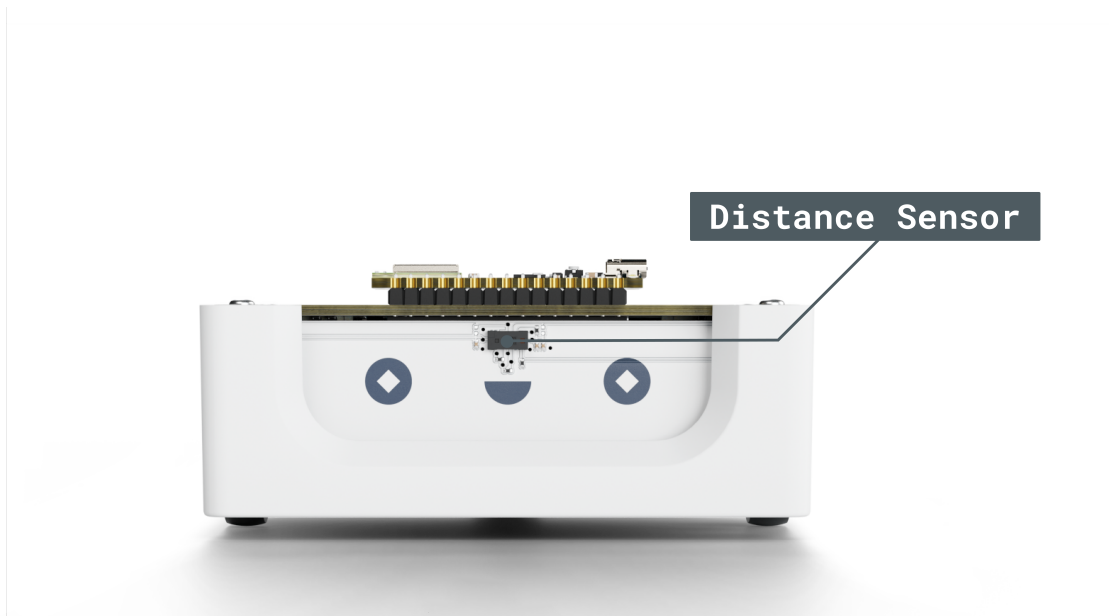
Jede Taste kann so programmiert werden, dass sie bestimmte Aktionen ausführt, wodurch eine komplexere Schnittstelle für die Benutzerinteraktion entsteht.

Abstandssensor

Der Arduino Alvik Roboter ist mit einem ToF (Time of Flight) 8x8 Array-Sensor ausgestattet, genauer gesagt mit dem VL53L7CX, der Entfernungen von bis zu 350 cm messen kann. Diese Abstandssensoren helfen dem Roboter, Hindernisse zu erkennen und die Entfernung zu Objekten in seiner Umgebung zu messen. Der Sensor ist in die vordere Leiterplatte des Roboters integriert.

Schlüsselmerkmale:

- Maximale Entfernung: 350 cm
- Hohe Präzision und Genauigkeit
- Für die Hinderniserkennung und Entfernungsmessung



Linienverfolgungssensoren

Der Arduino Alvik-Roboter ist mit Linienverfolgungssensoren ausgestattet, die ihm helfen, Linien auf dem Boden zu erkennen und ihnen zu folgen. Dies ist nützlich für Anwendungen, bei denen der Roboter entlang vordefinierter Pfade navigieren muss. Das Line Follower Array besteht aus drei Fototransistoren und fünf Infrarot-LEDs, die auf der vorderen Platine montiert sind.

Die Infrarot-LEDs strahlen Licht in Richtung Boden ab, und die Fototransistoren erkennen das reflektierte Licht. Wenn die Sensoren eine Linie überqueren (typischerweise eine dunkle Linie auf einer hellen Fläche), ändert sich die Menge des reflektierten Lichts. Dunkle Linien reflektieren im Vergleich zu hellen Flächen weniger Infrarotlicht, so dass die Fototransistoren das Vorhandensein einer Linie erkennen können.

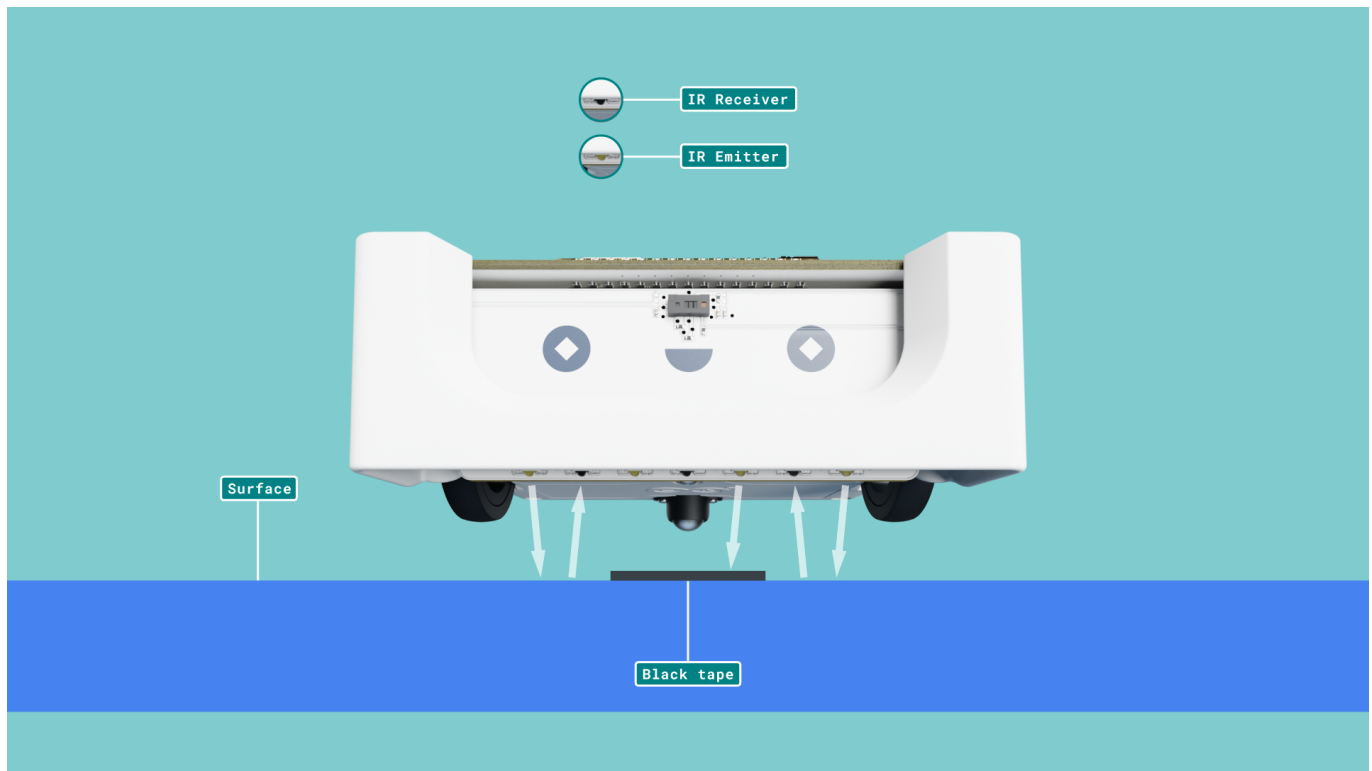
Schlüsselmerkmale:

- Erkennt Linien auf dem Boden (oder beliebige Muster aus Farben oder Materialien mit unterschiedlichen Reflexionseigenschaften)
- Hilft bei der Navigation auf vordefinierten Pfaden
- Besteht aus drei Fototransistoren und fünf Infrarot-LEDs

Wie es funktioniert:

Infrarot-LEDs: Senden infrarotes Licht in Richtung Boden aus. **Phototransistoren:** Erfassen das reflektierte Infrarotlicht.

- Wenn Sie sich über einer dunklen Linie befinden, ist die Reflexion gering, was auf das Vorhandensein einer Linie hinweist.
- Über einer hellen Fläche ist die Reflexion hoch, was bedeutet, dass keine Linie vorhanden ist.



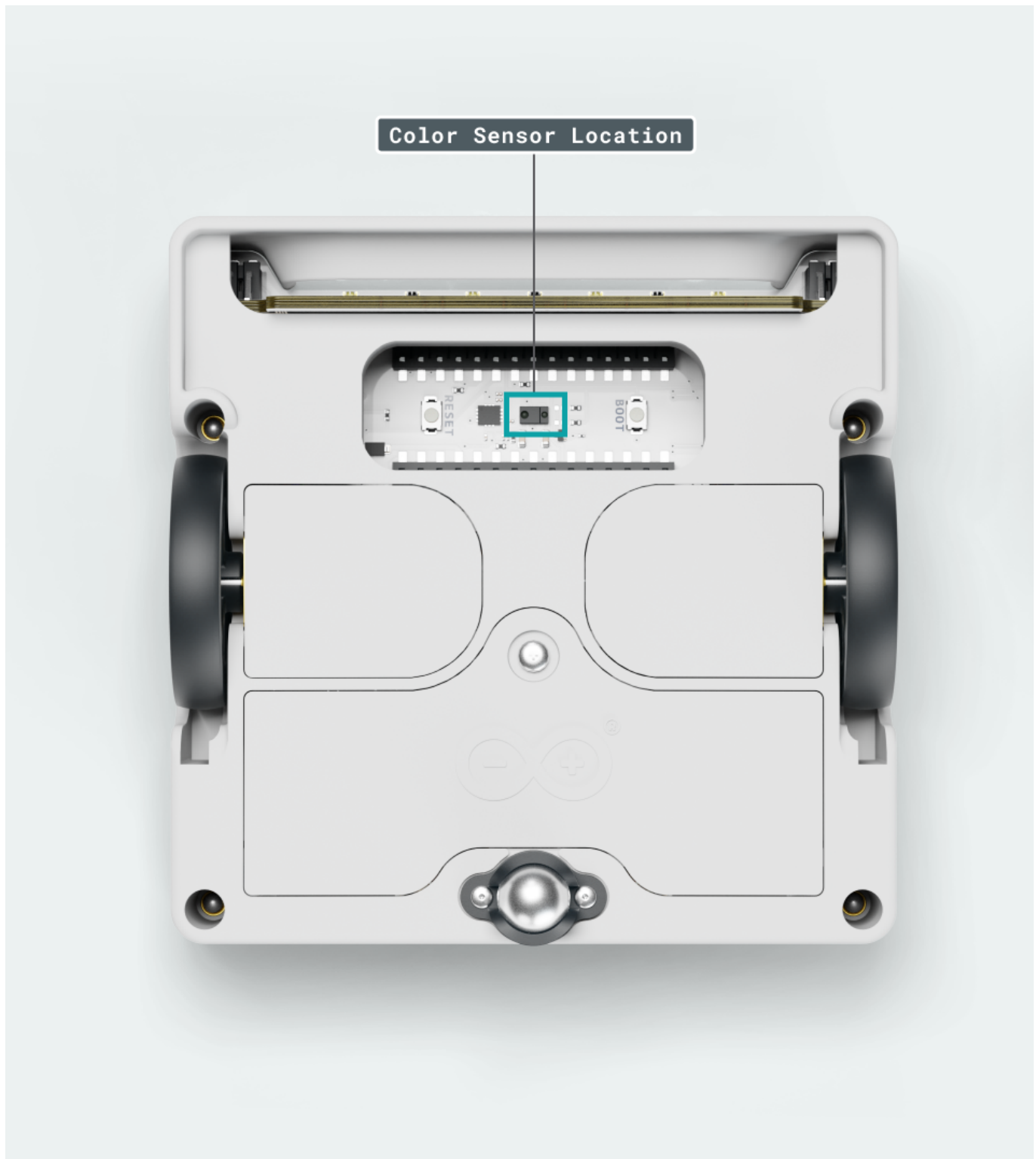
Durch kontinuierliches Ablesen der Werte von den Fototransistoren kann der Roboter seine Position relativ zur Linie bestimmen und seine Bewegungen so anpassen, dass er dem Pfad genau folgt.

Farbsensor

Der Farbsensor des Arduino Alvik Roboters wird verwendet, um Farben auf Oberflächen zu erkennen und zu identifizieren, auf die der Roboter trifft. Er liefert sowohl rohe Farbwerte als auch beschriftete Farbinformationen, die für verschiedene Anwendungen wie Linienverfolgung, Objekterkennung und mehr verwendet werden können. Der verwendete Sensor ist der APDS 9660.

Schlüsselmerkmale:

- Erkennt und identifiziert Farben
- Liefert rohe Farbanzeigen und beschriftete Farbinformationen
- Nützlich für Linienverfolgung, Objekterkennung und andere Anwendungen



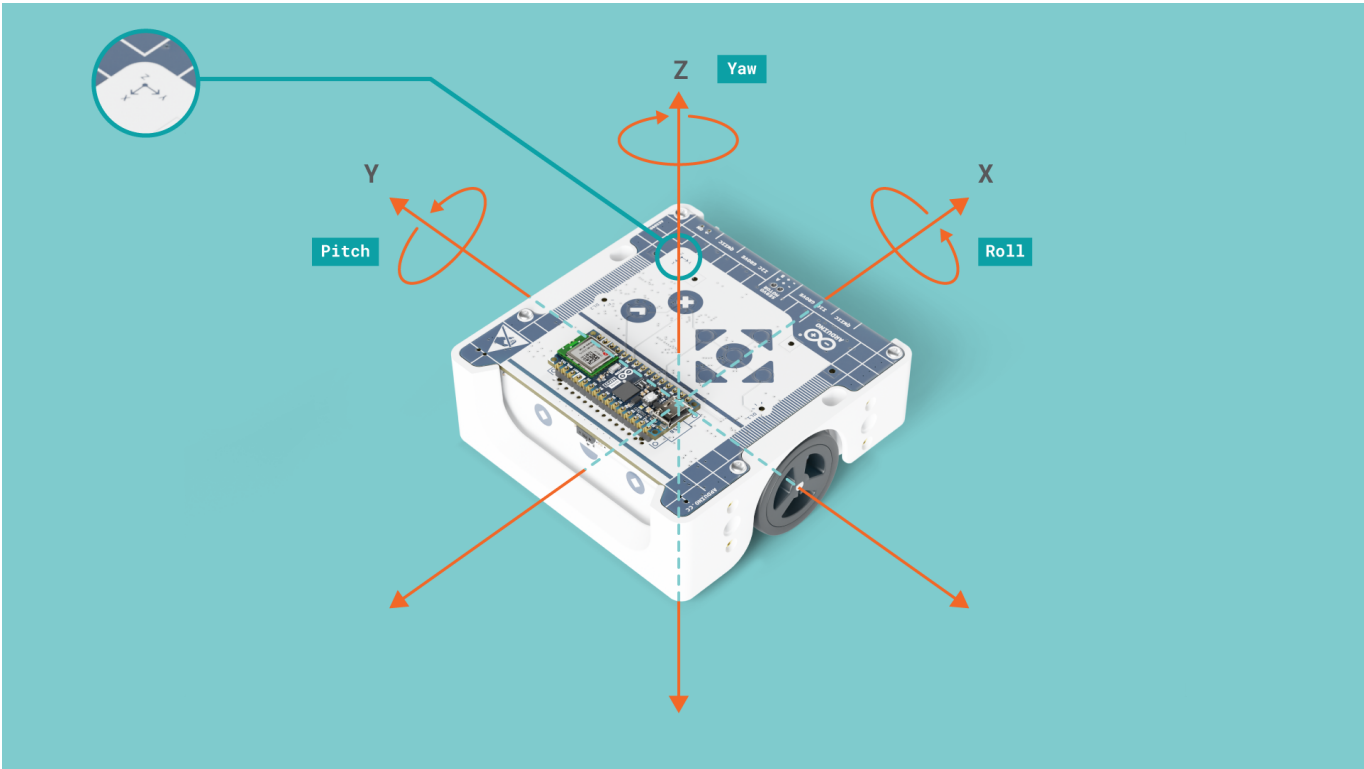
IMU

Der Arduino Alvik-Roboter ist mit einer eingebauten IMU (Inertial Measurement Unit) ausgestattet, die wertvolle Informationen über die Bewegung und Ausrichtung des Roboters liefert. Die IMU kann die Beschleunigung, die Winkelgeschwindigkeit und die Ausrichtung messen. Der verwendete Sensor ist der LSM6DSOX.

Schlüsselmerkmale:

- Misst Beschleunigung, Winkelgeschwindigkeit und Orientierung
- Liefert Roll-, Nick- und Gierwerte

- Nützlich für Bewegungsverfolgung und Stabilisierung



Alviks Aktuatoren

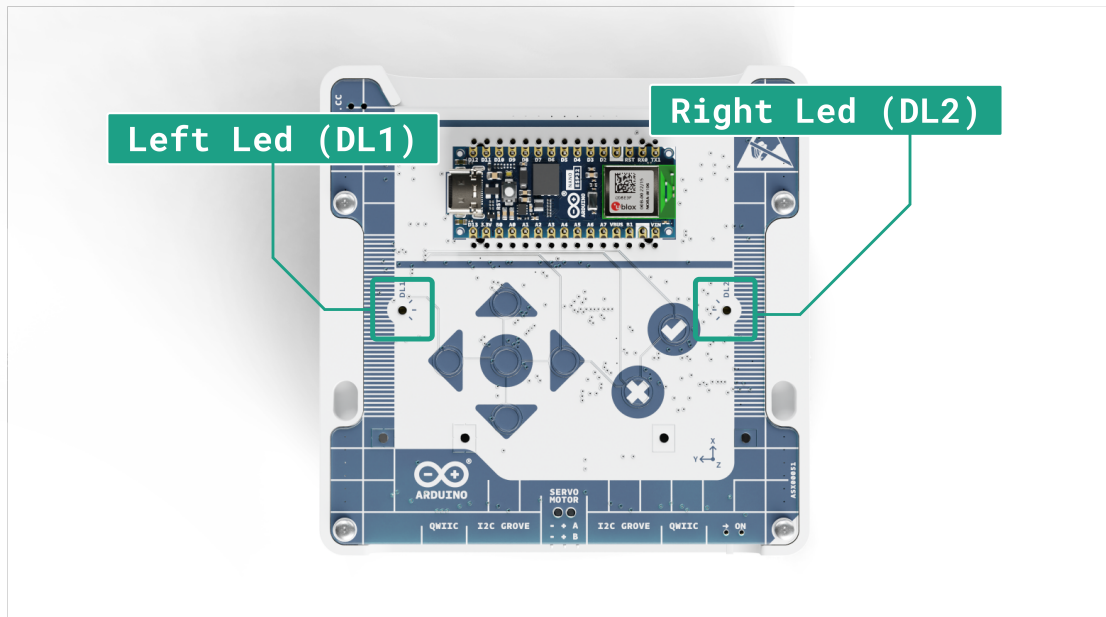
Aktuatorname	Teilname	Testprogrammname
Geared motors w/ encoder	GM12-N20VA-08255-150-EN	wheels_positions.py
RGB LEDs	RGB LEDs	leds_settings.py

Motoren

Der Alvik ist mit zwei Getriebemotoren ausgestattet, und zwar mit dem GM12-N20VA-08255-150-EN. Diese Motoren ermöglichen eine präzise und zuverlässige Bewegung, die für eine genaue Navigation und Steuerung unerlässlich ist, da jeder Motor über einen magnetischen Relativ-Encoder verfügt, der eine präzise Steuerung und Rückmeldung der Motorposition gewährleistet.

- **Getriebeübersetzung:** 1:150
- **Betriebsspannung:** 6V
- **Leerlaufdrehzahl:** 55 RPM
- **Stromstärke bei Nulllast:** 70mA
- **Encoder Typ:** Magnetischer relativer Encoder
- **Magnetischer relativer AB-Geber:** 3 ppr

LED-Anzeigen



Die RGB-LEDs können für visuelles Feedback verwendet werden und lassen sich so programmieren, dass sie verschiedene Farben und Muster anzeigen. Die LEDs nehmen **boolesche** Werte für jede Farbe auf, was bedeutet, dass Kombinationen möglich sind, indem jede Farbe **an** oder **aus** gesetzt wird.

Out Of The Box Modi

Alvik ist mit drei einsatzbereiten Beispielen ausgestattet. Um eines der Beispiele auszuwählen, schalten Sie Ihren Alvik ein, bewegen Sie den Schalter in der rechten unteren Ecke des Roboters nach rechts, warten Sie, bis die LEDs blau leuchten, und verwenden Sie die Tasten Auf und Ab, um eine Farbe auszuwählen, und drücken Sie dann die Bestätigungstaste.

- **Basic Touch Programming:** Die Touch-Tasten von Alvik können Aktionen wie Bewegung, Moduswechsel und Interaktion mit der Umgebung steuern. Dies ermöglicht intuitive Steuerungsschemata für verschiedene Anwendungen.
- **Objektverfolgung:** Mithilfe seiner Abstandssensoren kann Alvik Objekte verfolgen, indem es einen bestimmten Abstand einhält. Diese Funktion ist nützlich für fahrerlose Transportsysteme und interaktive Robotertechnik.
- **Linienverfolger:** Alviks Line Follower-Sensoren ermöglichen es ihm, entlang von Pfaden zu navigieren, die durch Linien auf dem Boden markiert sind. Dies eignet sich hervorragend für Lehrzwecke und zur Demonstration der grundlegenden Roboternavigation.

Weitere Informationen finden Sie im [Alvik Getting Started Guide](#).

Alviks Firmware

Um mit Alvik spielen zu können, benötigen Sie die folgende Hardware und Software:

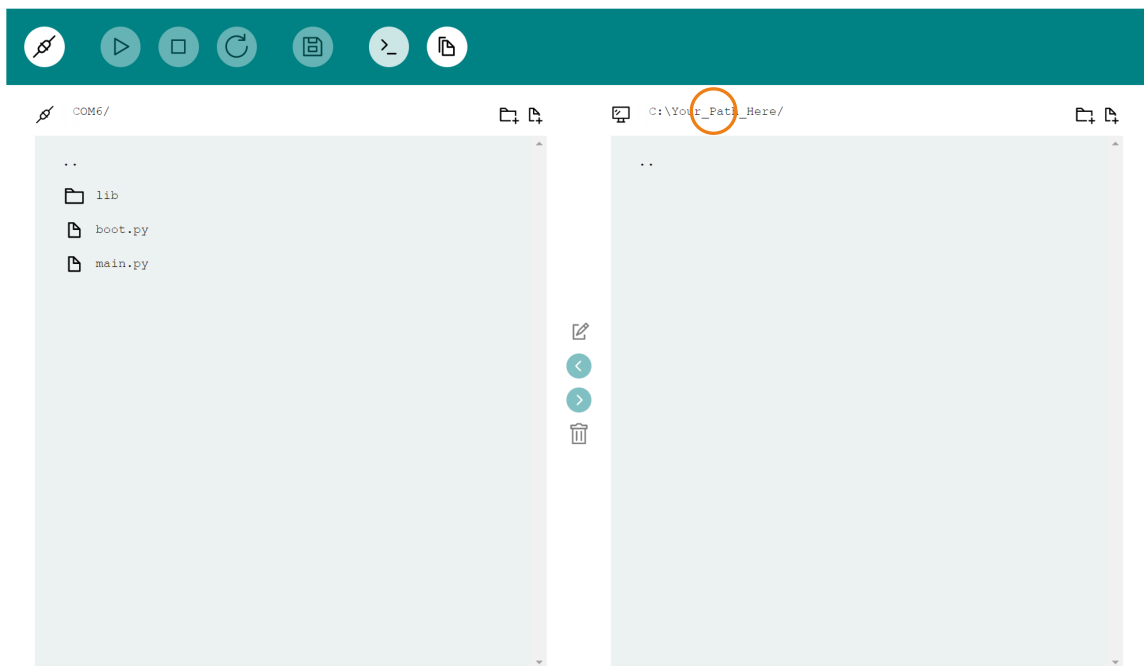
Software-Anforderungen

- Betriebssysteme: Linux, macOS oder Windows
- [Arduino-Labor für Micropython](#)

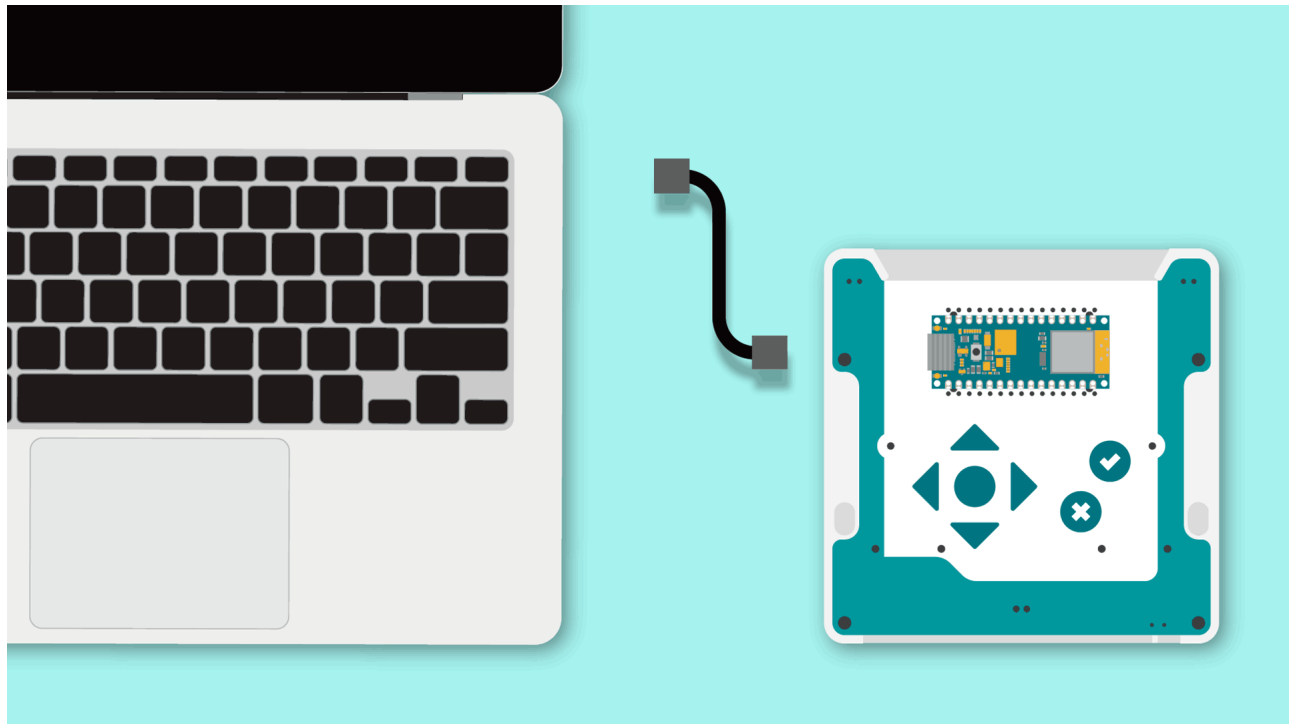
Alviks Gehirn aktualisieren (Nano ESP32)

Es kann vorkommen, dass Sie Ihren Nano ESP32 für andere Projekte als Alvik verwendet haben, oder Sie müssen ihn ersetzen. Um Ihren Nano ESP32 mit Alvik zum Laufen zu bringen, sind nur wenige Schritte erforderlich:

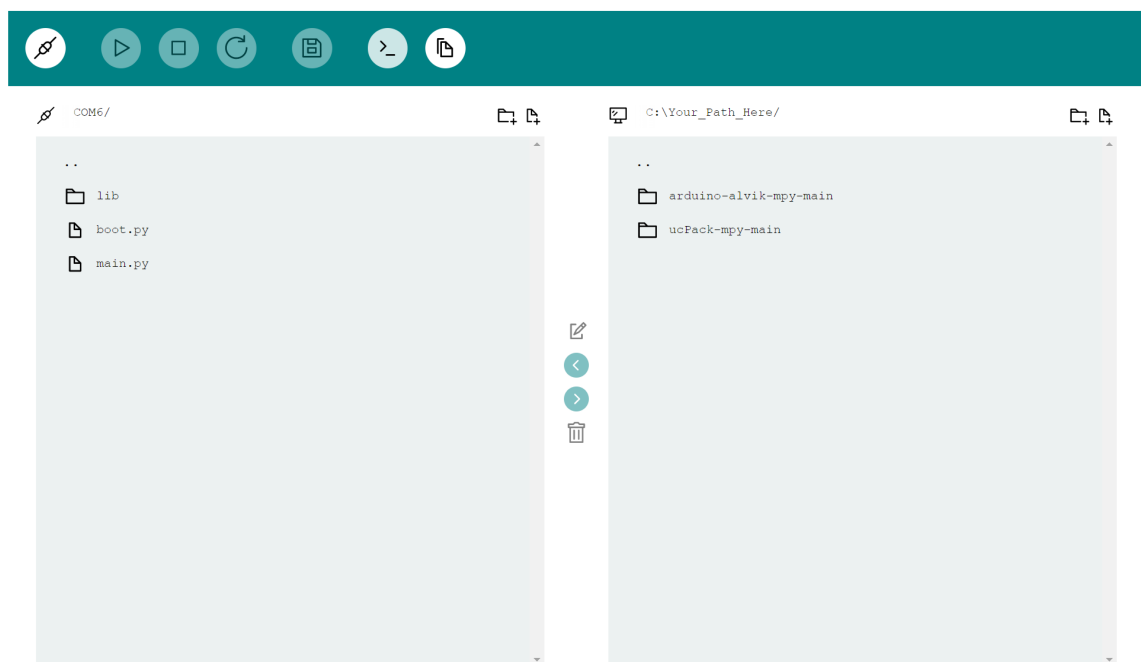
1. Installieren Sie den MicroPython-Bootloader gemäß [dieser Anleitung](#) auf dem Board.
2. Laden Sie die Alvik Micropython-Bibliotheken herunter Alvik MicroPython-Bibliotheken aus dem [Alvik Repository](#) ucPack-Bibliotheken aus dem [ucPack-Repository](#)
3. Entpacken Sie beide heruntergeladenen Bibliotheken in einen einzigen "Alvik"-Ordner, öffnen Sie das Arduino Lab für MicroPython, gehen Sie auf die Registerkarte "Dateien" und legen Sie den Pfad zum entpackten Ordner im Arduino Lab für MicroPython fest.



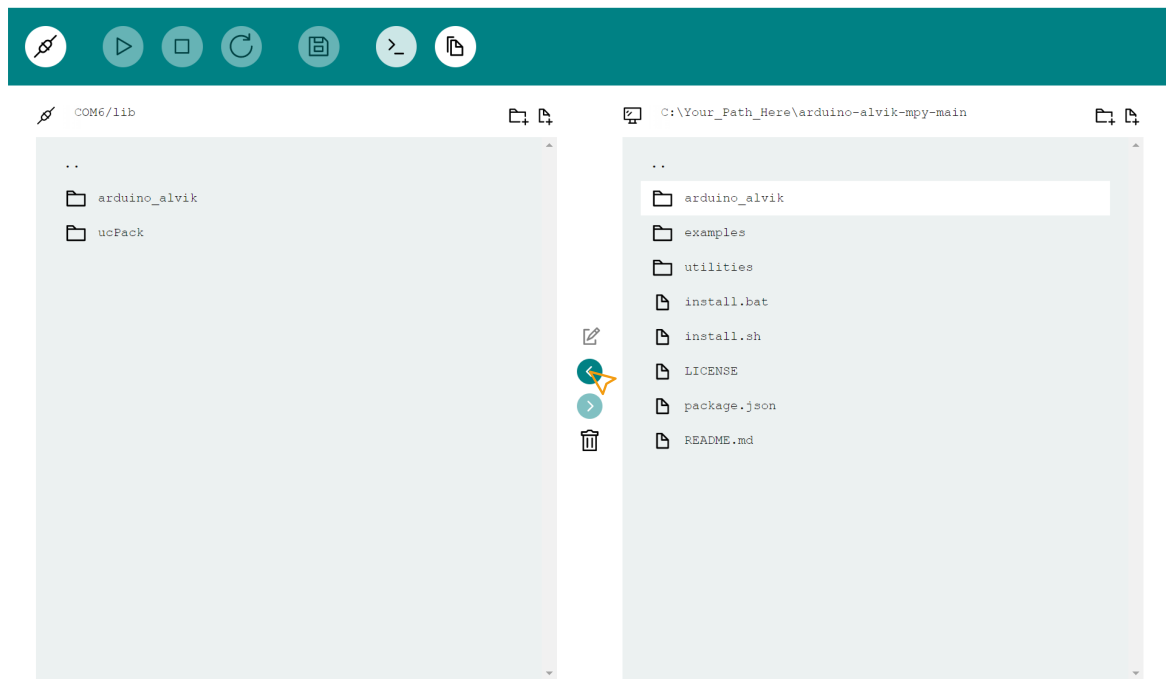
4. Vergewissern Sie sich, dass Ihr Alvik ausgeschaltet ist, schließen Sie es an Ihren Computer an und schalten Sie es dann ein:



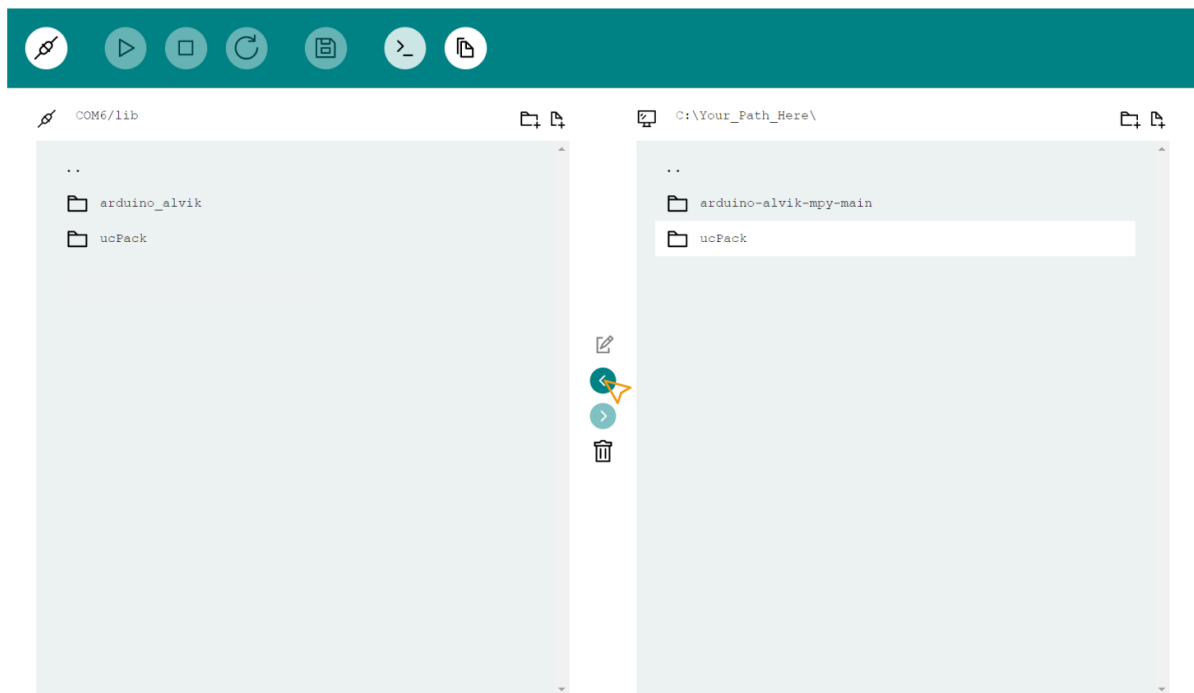
5. Verbinden Sie Ihr Alvik mit den Arduino Labs for MicroPython und öffnen Sie die "lib".



6. Wählen Sie "Arduino-alkvik" und verschieben Sie es in den Ordner "lib" in Ihrem Alvik.

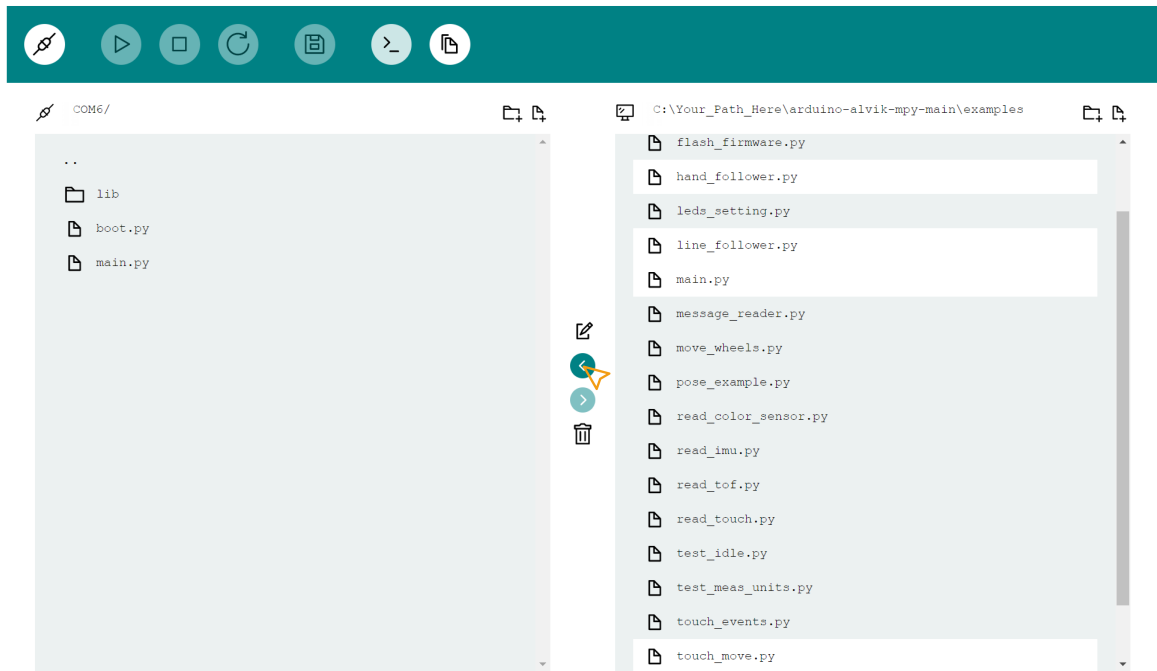


7. Gehen Sie zurück zum Hauptordner und wählen Sie den Ordner "ucPack" aus, der sich im Ordner "ucPack-mpy-main" befindet, und verschieben Sie ihn neben den Ordner *arduino_alkvik* im Ordner "lib" in Ihrem Nano ESP32.



8. Gehen Sie nun zurück zum Hauptverzeichnis des Dateisystems auf dem Nano ESP32. Navigieren Sie dann in Ihrem lokalen Ordner zum Ordner examples, wählen Sie dort die folgenden Dateien aus und verschieben Sie sie in den Hauptordner des ESP32.

`demo.py` `hand_follower.py` `line_follower.py` `main.py` `touch_move.py`



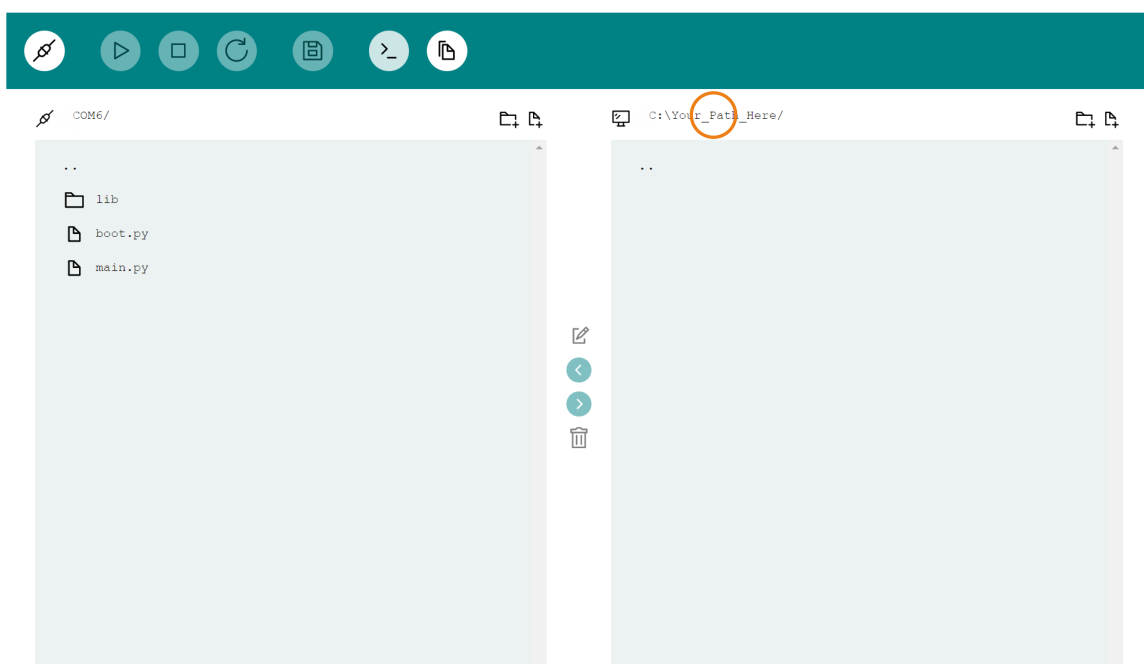
Mit diesem letzten Schritt ist Ihr Nano ESP32 mit der Alvik-Out-of-the-Box-Erfahrung eingerichtet und bereit für den Einsatz.

Aktualisieren von Alviks Body (STM32)

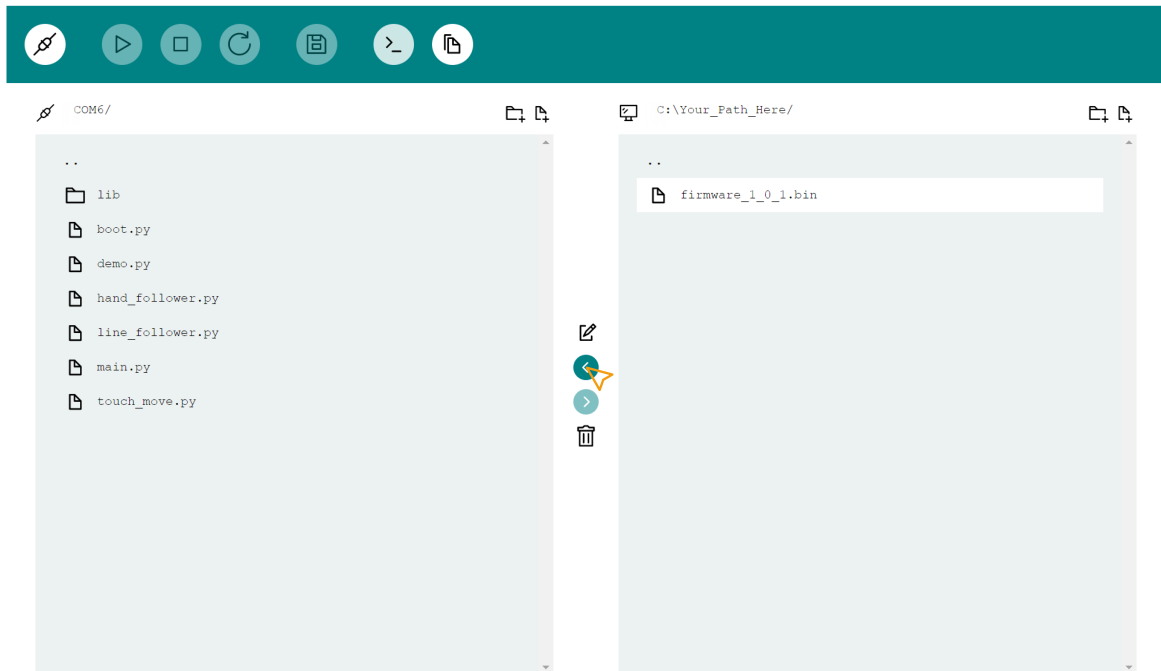
1. Laden Sie die [vorkompilierte Firmware](#) aus dem [Alvik Carrier GitHub Repository](#) herunter.

Bei diesem Schritt wird eine Datei "firmware_x_x_x.bin" heruntergeladen, die Sie in Ihrem Alvik-Ordner speichern.

2. Verbinden Sie Ihr Alvik mit dem Computer und mit den Arduino Labs für MicroPython. Gehen Sie dann auf die Registerkarte Dateien und navigieren Sie zu dem Ordner, in dem Sie die "firmware_x_x_x.bin" gespeichert haben.

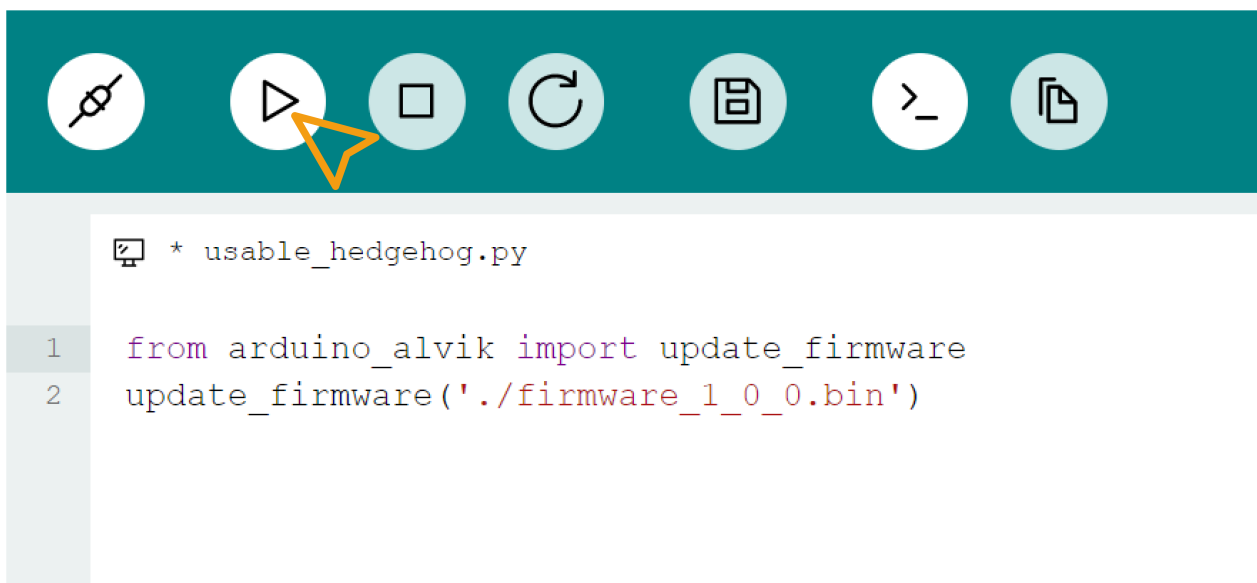


3. Verschieben Sie nun die Datei `firmware_x_x_x.bin` in das Hauptverzeichnis.



4. Schalten Sie Ihr alvik ein, gehen Sie auf die Registerkarte Editor und führen Sie die folgenden Befehle aus, indem Sie sie eingeben und auf die Taste "Abspielen" klicken:

```
from arduino_alvik import update_firmware  
  
update_firmware('./firmware_1_0_0.bin')
```



Nachdem Sie diese Befehle ausgeführt haben, wird der Prozess an der Eingabeaufforderung aktualisiert. Sobald der Prozess abgeschlossen ist, wird die Firmware Ihres alvik aktualisiert.

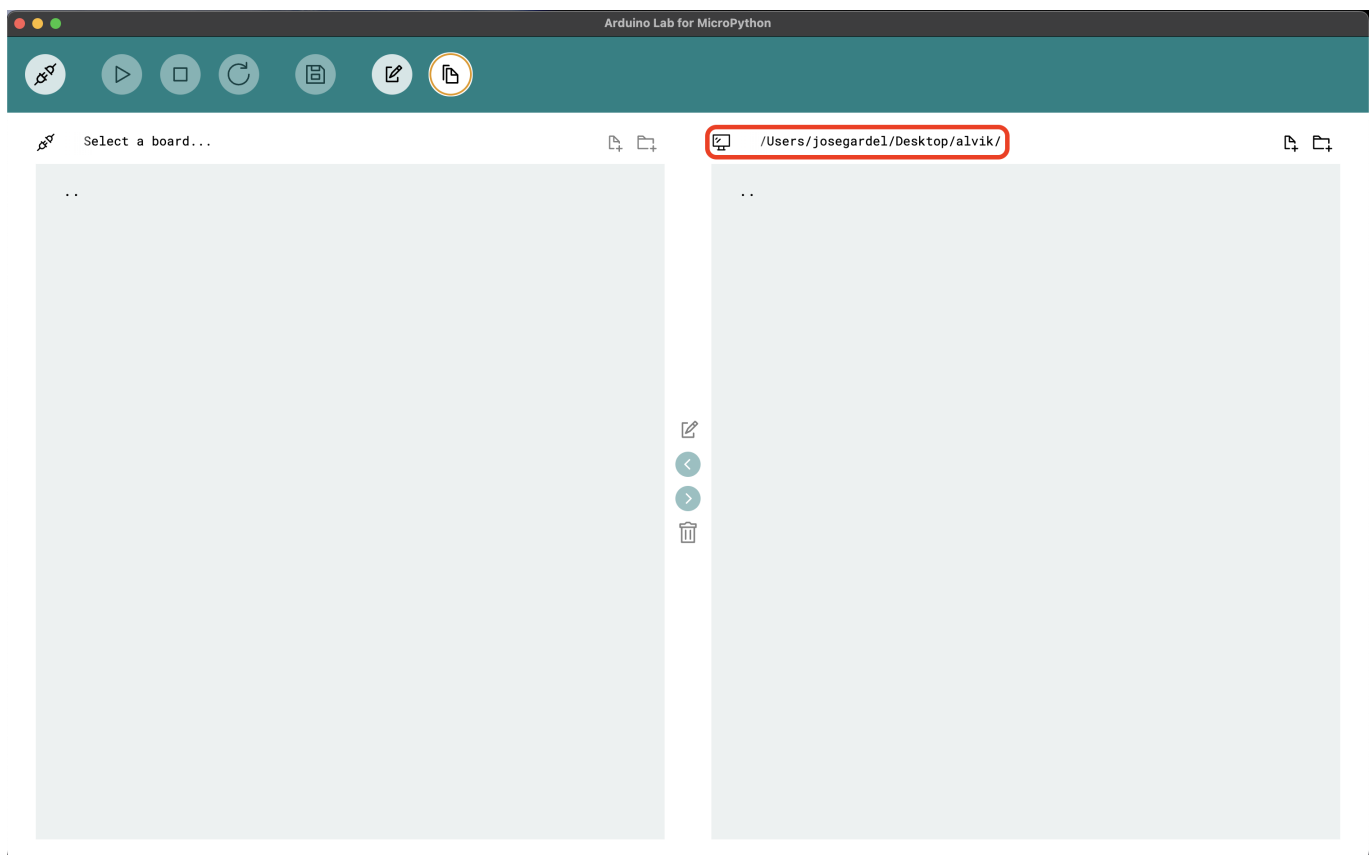
Hallo Alvik! Ihr erstes Programm!

Alvik ist für die Programmierung mit MicroPython vorgesehen. Wir empfehlen Ihnen, den Editor [Arduino Lab for MicroPython](#) zu installieren.

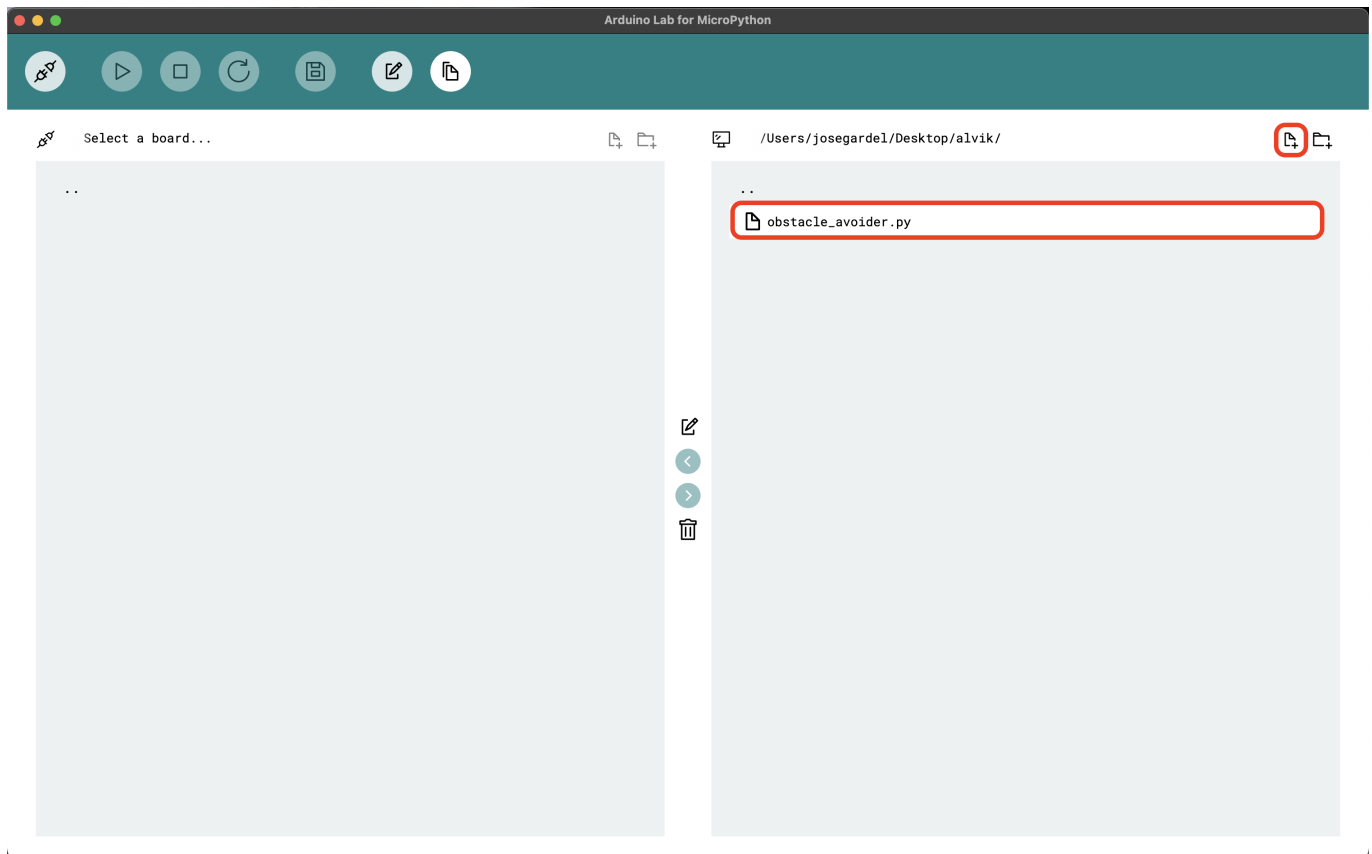
Wenn Sie ein Chromebook verwenden, müssen Sie die [Online-Version](#) von Arduino Lab for MicroPython mit Chrome verwenden. Sie müssen sich mit einem Arduino-Konto anmelden, um es zu benutzen.

Nachdem nun alle vorherigen Schritte festgelegt wurden, sehen wir uns an, wie man benutzerdefinierte Programme für Alvik erstellt, damit es sich vorwärts bewegt, bis es ein Objekt vor sich erkennt, diesem ausweicht und seinen Weg fortsetzt.

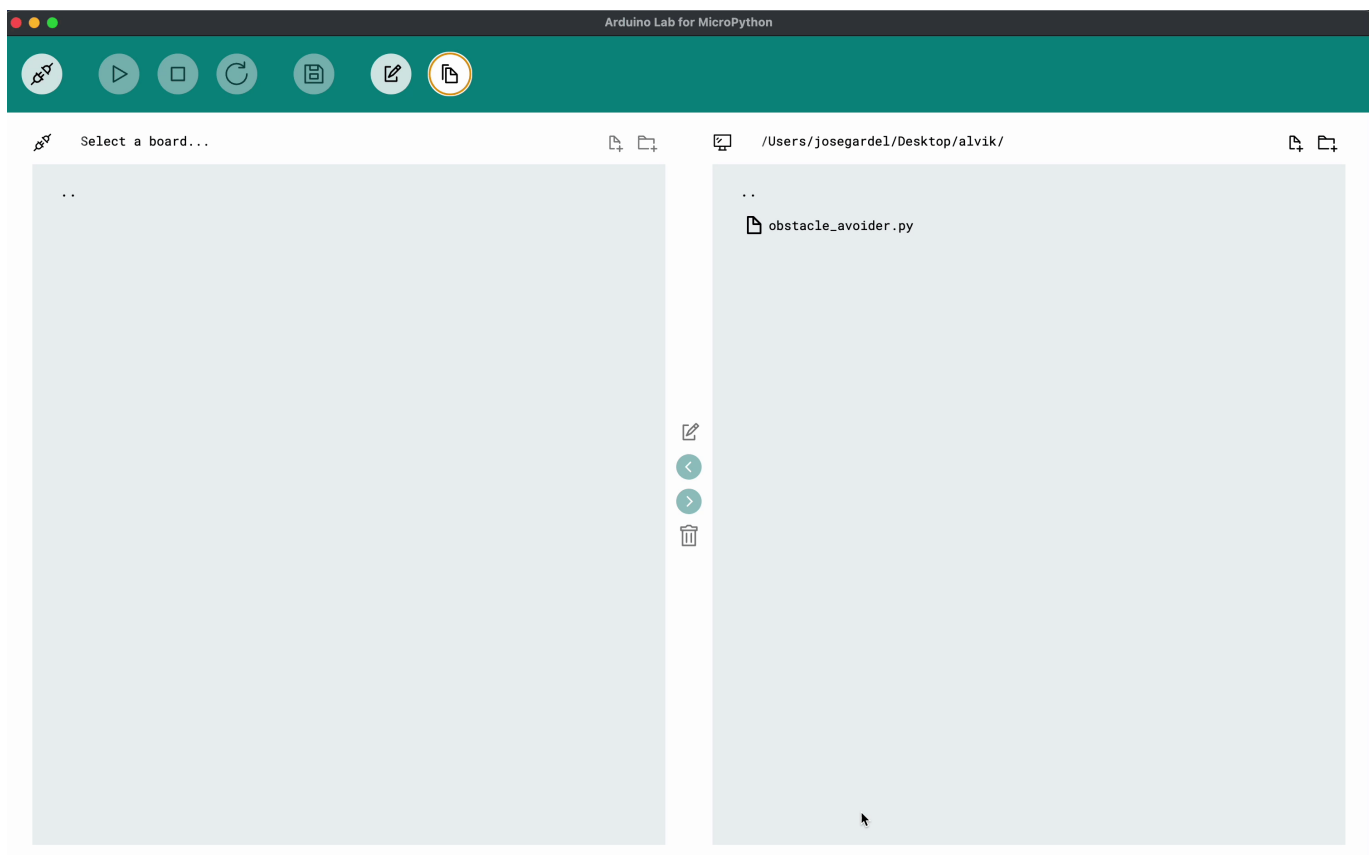
1. Erstellen Sie einen Alvik-Ordner auf Ihrem Computer und legen Sie diesen als Pfad für die Arduino Lab for MicroPython IDE fest.



2. Erstellen Sie eine neue Datei `obstacle_avoider.py` in Ihrem lokalen Ordner.



3. Doppelklicken Sie auf die Datei, um sie zu öffnen. Sobald sie geöffnet ist, löschen Sie den Text darin und fügen Sie den folgenden Code ein.



```
from arduino_alvik import ArduinoAlvik
from time import sleep_ms
```



```
import sys

alvik = ArduinoAlvik()
alvik.begin()
sleep_ms(5000) #waiting for the robot to setup
#robot.set_illuminator(0)
distance = 10
speed = 40 #rpm

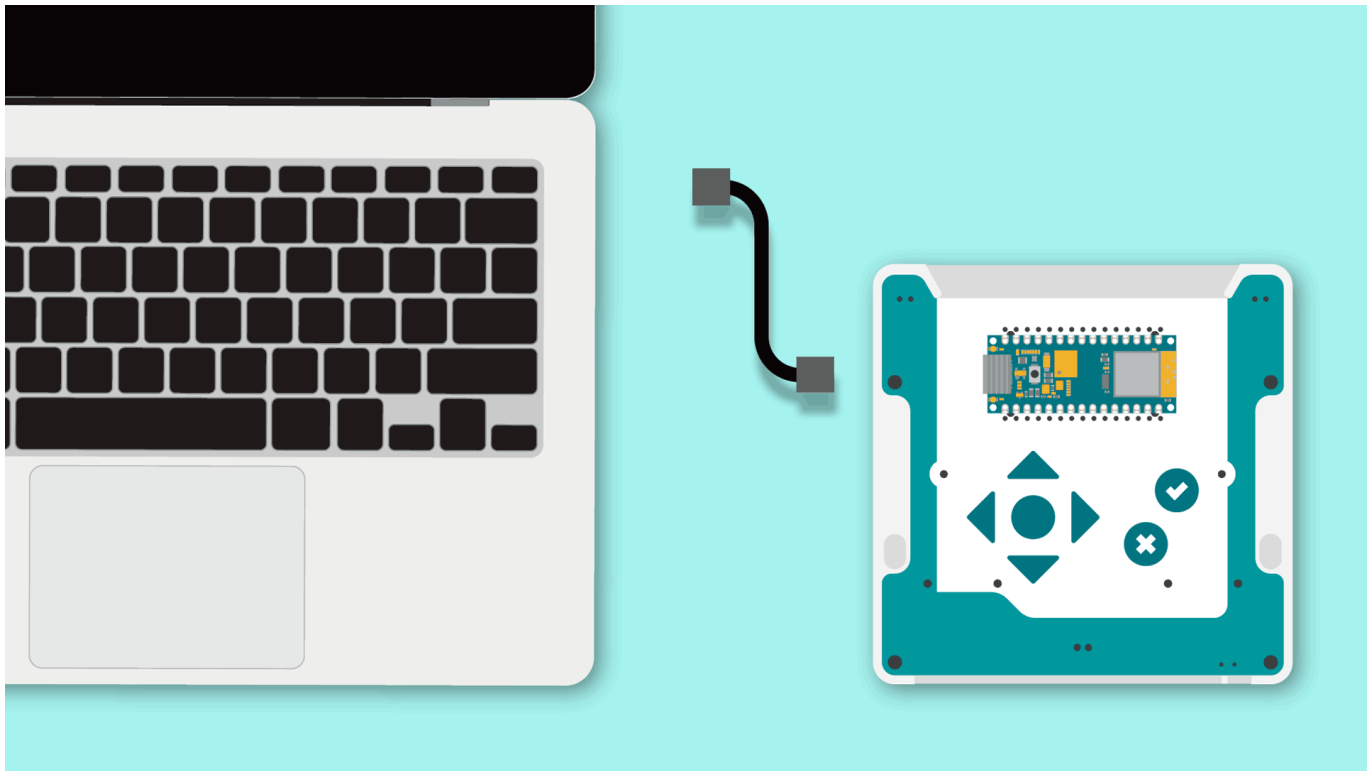
def turning():
    alvik.set_wheels_speed(0,0)
    sleep_ms(250)
    alvik.set_wheels_speed(-35,-35)
    sleep_ms(1500)
    alvik.set_wheels_speed(35,-35)
    sleep_ms(1000)

while (True):

    distance_l, distance_cl, distance_c, distance_r, distance_cr =
alvik.get_distance()
    sleep_ms(50)
    print(distance_c)

    if distance_c < distance:
        turning()
    elif distance_cl < distance:
        turning()
    elif distance_cr < distance:
        turning()
    elif distance_l < distance:
        turning()
    elif distance_r < distance:
        turning()
    else:
        alvik.set_wheels_speed(speed, speed)
```

4. Verbinden Sie Alvik über das mitgelieferte Kabel unter der Ablage mit Ihrem PC.



***Stellen Sie sicher, dass Alvik ausgeschaltet ist, bevor Sie es an Ihren Computer anschließen.

5. Sobald Alvik mit dem PC verbunden ist, verbinden Sie es mit dem Arduino Lab for MicroPython und öffnen Sie die Datei *main.py* im Alvik-Ordner. Sobald die Datei geöffnet ist, ersetzen wir die Anweisung `import demo` durch `import obstacle_avoider`.

```

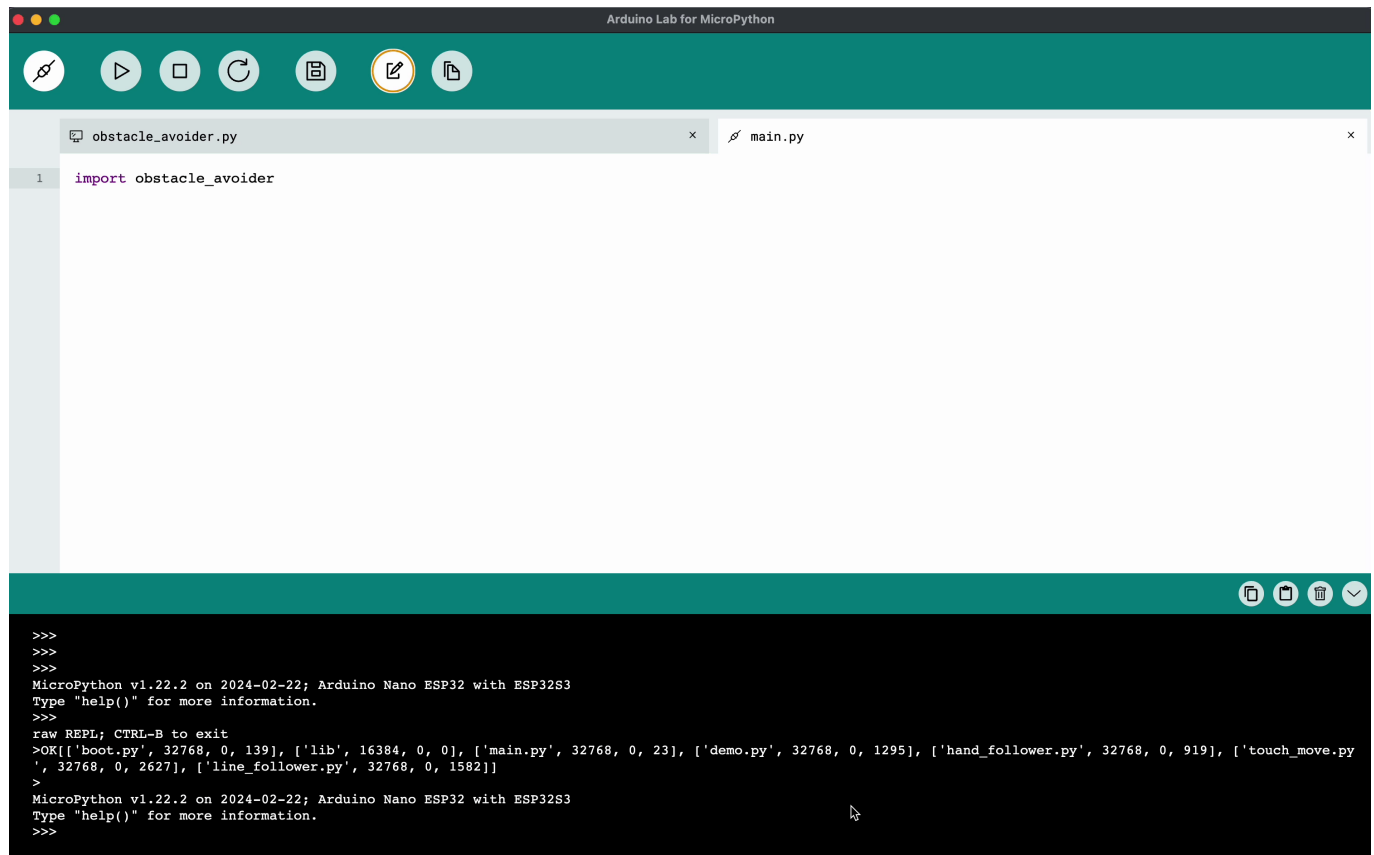
Arduino Lab for MicroPython
obstacle_avoider.py
1  from arduino_alvik import ArduinoAlvik
2  from time import sleep_ms
3  import sys
4
5  alvik = ArduinoAlvik()
6  alvik.begin()
7  sleep_ms(5000) #waiting for the robot to setup
8  #robot.set_illuminator(0)
9  distance = 10
10
11
12
13  while (True):
14
15      distance_l, distance_cl, distance_c, distance_r, distance_cr = alvik.get_distance()
16      sleep_ms(50)
17      print(distance_c)
18
19      if distance_c < distance:
20          alvik.rotate(45.00, 'deg')
21      elif distance_cl < distance:
22          alvik.rotate(45.00, 'deg')
23      elif distance_cr < distance:
24          alvik.rotate(45.00, 'deg')
25      elif distance_l < distance:
26          alvik.rotate(45.00, 'deg')
27      elif distance_r < distance:
28          alvik.rotate(45.00, 'deg')
29      else:
30          alvik.drive(10.0, 0.0, linear_unit='cm/s')

```

Wenn Sie wieder zu der Erfahrung zurückkehren möchten, dass Sie zwischen den Programmen *reg*, *green* und *blue* wählen können, müssen Sie nur die *main.py* erneut ändern und die Anweisung `import`

obstacle_avoider durch *import demo* ersetzen.

6. Der letzte Schritt besteht darin, die Datei *obstacle_avoider.py* aus dem lokalen Repository in den Speicher von Alvik zu verschieben.



Jetzt sind Sie bereit. Trennen Sie Alvik vom Computer, stellen Sie einige Hindernisse um Alvik herum auf, schalten Sie es ein und sehen Sie, wie Alvik sie erkennt und sich dreht, um ihnen auszuweichen.

Deep Dive Programmierung Alvik

Steuerung der Motoren

Die Bewegung ist eines der Hauptmerkmale von Alvik. Daher ist es wichtig, eine Vielzahl von Methoden zur Steuerung der Alvik-Motoren zu haben. Dank dieser Flexibilität können Sie je nach Bedarf verschiedene Steuerungsmethoden verwenden, sei es für eine präzise Bewegung oder eine einfache Geschwindigkeitskontrolle.

Macht über Entfernung/Zeit

Sie können die Geschwindigkeit der Alvik-Motoren direkt einstellen und sie für eine bestimmte Zeit oder Strecke laufen lassen. Diese Methode ist einfach und nützlich für einfache Aufgaben, bei denen eine präzise Steuerung nicht erforderlich ist.

Funktion: `set_wheels_speed`

Die Funktion `set_wheels_speed` stellt die Geschwindigkeit des linken und rechten Motors ein.

Eingaben:

- `left_speed`: Der Geschwindigkeitswert für den linken Motor.

- **right_speed**: Der Geschwindigkeitswert für den rechten Motor.
- **unit**: Einheit der Drehgeschwindigkeit der Räder (Standard ist 'rpm').

Benutzungsbeispiel:

```
alvik.set_wheels_speed(30, 30, 'rpm')
sleep_ms(2000) # Run motors at 30 rpm for 2 seconds
alvik.brake()
```

Entfernung und Winkel

Mit Alvik können Sie die Motoren auf der Grundlage von bestimmten Entfernungen oder Winkeln steuern. Diese Methode bietet eine präzise Kontrolle über die Bewegungen des Roboters und stellt sicher, dass er genau die angegebene Strecke zurücklegt oder sich um den genauen Winkel dreht.

Methode **set_wheels_position**

Die Funktion **set_wheels_position** stellt die Winkelposition für den linken und rechten Motor ein.

Eingaben:

- **left_angle**: Der Winkelwert für den linken Motor.
- **right_angle**: Der Winkelwert für den rechten Motor.
- **unit**: Die Winkeleinheit (Standard ist 'deg').

Benutzungsbeispiel:

```
alvik.set_wheels_position(360, 360, 'deg')
sleep_ms(2000) # Dreht die Räder um 360 Grad in 2 Sekunden
alvik.brake()
```

Move verwenden

Die Funktion **move** bewegt den Roboter um eine bestimmte Strecke.

Eingaben:

- **distance**: Der Entfernungswert.
- **unit**: Die Entfernungseinheit (Standard ist 'cm').
- **blocking**: True oder False, ob die Funktion blockieren soll, bis die Bewegung abgeschlossen ist (Standardwert ist True).

Benutzungsbeispiel:

```
alvik.move(50, 'cm')
sleep_ms(2000) # Move forward 50 cm in 2 seconds
alvik.brake()
```

Laufwerk verwenden

Sie können die Leistung und die Zeit für die Motoren von Alvik auch in Grad pro Sekunde oder Zentimetern pro Sekunde angeben. Diese Methode ist nützlich für Aufgaben, die eine fein abgestimmte Steuerung der Geschwindigkeit und Richtung des Roboters erfordern.

drive verwenden

Die Funktion `drive` treibt den Roboter an, indem sie die lineare und die Winkelgeschwindigkeit einstellt.

Eingaben: `linear_velocity`: Geschwindigkeit des Roboters. `angular_velocity`: Geschwindigkeit der Räder. `linear_unit`: Einheit der linearen Geschwindigkeit (Standard ist 'cm/s'). `angular_unit`: Einheit der Rotationsgeschwindigkeit der Räder (Standardwert ist 'deg/s').

Benutzungsbeispiel:

```
alvik.drive(10, 30, 'cm/s', 'deg/s')
sleep_ms(2000) # Drive at 10 cm/s linear velocity and 30 deg/s angular velocity
for 2 seconds
alvik.brake()
```

Steuerung des Encoders

Der Alvik-Roboter bietet verschiedene Funktionen zur Steuerung und Überwachung der Motoren mithilfe von Encodern. Mit diesen Funktionen erhalten Sie Echtzeit-Feedback zu Geschwindigkeit, Position und Haltung des Roboters, was für eine präzise Bewegung und Navigation unerlässlich ist.

Geschwindigkeit der Räder prüfen

Die Funktion `get_wheels_speed` gibt die Stromstärke der Räder zurück.

Eingaben:

- `unit`: Einheit der Rotationsgeschwindigkeit der Räder (Standard ist 'rpm').

Ausgaben:

- `left_wheel_speed`: Die Geschwindigkeit des linken Rades.
- `right_wheel_speed`: Die Geschwindigkeit des rechten Rades.

Benutzungsbeispiel:

```
left_speed, right_speed = alvik.get_wheels_speed(unit='rpm')
print(f"Left Wheel Speed: {left_speed} rpm, Right Wheel Speed: {right_speed} rpm")
```

Position der Räder prüfen

Die Funktion `get_wheels_position` gibt den Stromstärke-Winkel der Räder zurück.

Eingaben:

- `unit`: Einheit des Radwinkels (Standard ist 'deg').

Ausgaben:

- `left_wheel_angle`: Der Winkel des linken Rades.
- `right_wheel_angle`: Der Winkel des rechten Rads.

Benutzungsbeispiel:

```
left_angle, right_angle = alvik.get_wheels_position(unit='deg')
print(f"Left Wheel Angle: {left_angle} degrees, Right Wheel Angle: {right_angle} degrees")
```

Geschwindigkeit prüfen

Die Funktion `get_drive_speed` gibt die Linear- und Winkelgeschwindigkeit des Roboters zurück.

Eingaben:

- `linear_unit`: Einheit der linearen Geschwindigkeit (Standard ist 'cm/s').
- `angular_unit`: Einheit der Rotationsgeschwindigkeit der Räder (Standard ist 'deg/s').

Ausgaben:

- `linear_velocity`: Die lineare Geschwindigkeit des Roboters.
- `angular_velocity`: Die Winkelgeschwindigkeit der Räder.

Benutzungsbeispiel:

```
linear_velocity, angular_velocity = alvik.get_drive_speed(linear_unit='cm/s',
angular_unit='deg/s')
print(f"Linear Velocity: {linear_velocity} cm/s, Angular Velocity: {angular_velocity} deg/s")
```

Pose kontrollieren

Die Funktion `get_pose` gibt die Stromstärke des Roboters zurück.

Eingaben:

- `distance_unit`: Einheit der x- und y-Ausgaben (Standard ist 'cm').
- `angle_unit`: Einheit der Theta-Ausgabe (Vorgabe ist 'deg').

Ausgaben:

- `x`: Die x-Koordinate des Roboters.
- `y`: Die y-Koordinate des Roboters.
- `theta`: Der Orientierungswinkel des Roboters.

Benutzungsbeispiel:

```
x, y, theta = alvik.get_pose(distance_unit='cm', angle_unit='deg')
print(f"Pose - X: {x} cm, Y: {y} cm, Theta: {theta} degrees")
```

Diese Funktionen ermöglichen eine präzise Steuerung und Überwachung der Bewegungen des Alvik-Roboters und bieten fortschrittliche Navigations- und Bahnplanungsfunktionen.

Lesende Tasten

Der Arduino Alvik Roboter ist mit mehreren Tasten ausgestattet, die für verschiedene Eingaben verwendet werden können. Die folgenden Funktionen stehen zur Verfügung, um den Status der einzelnen Tasten zu lesen:

- Erkennen Sie **jede** gedrückte Taste.

Die Funktion `get_touch_any` gibt true zurück, wenn eine der Tasten gedrückt wird.

- Stellen Sie fest, dass die Taste **OK** gedrückt ist.

Die Funktion `get_touch_ok` gibt true zurück, wenn die Taste OK gedrückt wurde.

- Erkennen Sie, wenn Sie die Taste **Abbrechen** gedrückt haben.

Die Funktion `get_touch_cancel` gibt true zurück, wenn die Taste Cancel gedrückt wird.

- Erkennen Sie, dass die Taste **Mitte** gedrückt ist.

Die Funktion `get_touch_center` gibt true zurück, wenn die Taste Center gedrückt wird.

- Erkennen Sie die gedrückte Taste **Hoch**.

Die Funktion `get_touch_up` gibt true zurück, wenn die Taste Up gedrückt wird.

- Erkennen Sie die gedrückte Taste **Links**.

Die Funktion `get_touch_left` gibt true zurück, wenn die linke Taste gedrückt wurde.

- Erkennen Sie die gedrückte Taste **Abwärts**.

Die Funktion `get_touch_down` gibt true zurück, wenn die Taste Down gedrückt wird.

- Erkennen Sie die gedrückte Taste **Rechts**.

Die Funktion `get_touch_right` gibt true zurück, wenn die rechte Taste gedrückt wurde.

Beispiel Das folgende Beispiel zeigt, wie Sie den Status jeder Taste auslesen und ihren Namen ausgeben können, wenn sie gedrückt wird:

```
from arduino_alvik import ArduinoAlvik
from time import sleep_ms
```



```
import sys

alvik = ArduinoAlvik()
alvik.begin()

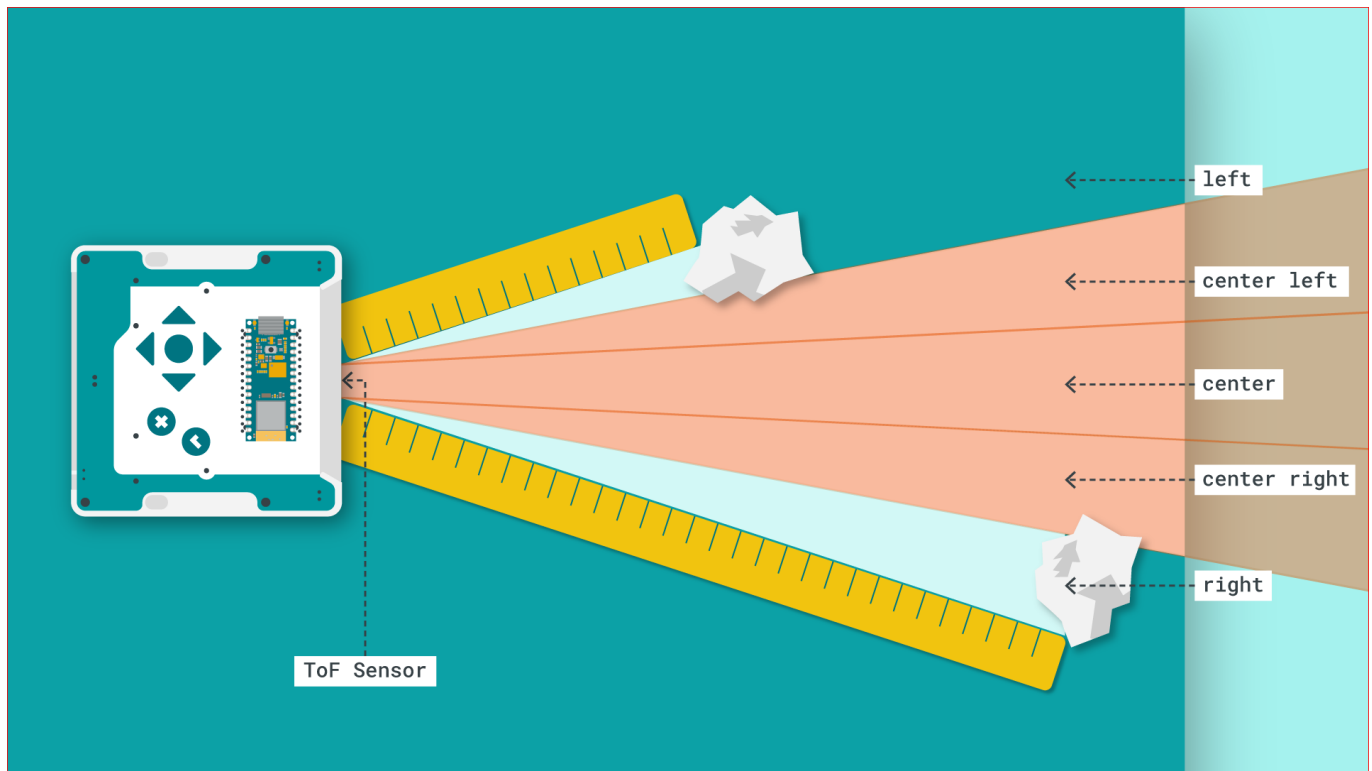
while True:
    try:
        if alvik.get_touch_any():
            if alvik.get_touch_up():
                print("UP")
            if alvik.get_touch_down():
                print("DOWN")
            if alvik.get_touch_left():
                print("LEFT")
            if alvik.get_touch_right():
                print("RIGHT")
            if alvik.get_touch_ok():
                print("OK")
            if alvik.get_touch_cancel():
                print("CANCEL")
            if alvik.get_touch_center():
                print("CENTER")

        sleep_ms(100)
    except KeyboardInterrupt as e:
        print('over')
        alvik.stop()
        sys.exit()
```

Dieses Beispiel bietet eine einfache Möglichkeit, die Tasten des Arduino Alvik Roboters zu lesen und darauf zu reagieren. Fühlen Sie sich frei, es zu erweitern, indem Sie benutzerdefinierte Reaktionen auf jeden Tastendruck hinzufügen.

Erkennen von Hindernissen

Diese Sensoren liefern detaillierte Messwerte von mehreren Positionen rund um den Roboter, einschließlich **links**, **mittig-links**, **mittig**, **mittig-rechts**, **rechts**, **oben** und **unten**. Im Gegensatz zu Einzelpunktsensoren bieten diese Abstandssensoren umfassende zonenbasierte Informationen.



Das Entfernungssensor-Array ist in fünf verschiedene Zonen unterteilt: links, Mitte-links, Mitte, Mitte-rechts und rechts. Jede Zone misst Entfernungen und ist in der Lage, Objekte in einer Entfernung von bis zu drei Metern zu erkennen. Es ist wichtig, darauf hinzuweisen, dass die Sensoren zwar Objekte in größeren Entfernungen erkennen können, die Zuverlässigkeit der Messwerte jedoch abnehmen kann.

Entfernung lesen

Die Funktion `get_distance` gibt die Entfernungswerte der Sensoren links, Mitte-links, Mitte, Mitte-rechts und rechts zurück.

Ausgaben: **L:** Anzeige des linken Sensors. **CL:** Linke Mitte-Sensor-Anzeige. **C:** Anzeige für den mittleren Sensor. **CR:** Mitte-Rechts-Sensoranzeige. **R:** Anzeige des rechten Sensors.

Benutzungsbeispiel

Hier ist ein Beispiel für die Verwendung der Entfernungssensoren, um Entfernungsmessungen zu erhalten:

```
from arduino_alvik import ArduinoAlvik
from time import sleep_ms
import sys

alvik = ArduinoAlvik()
alvik.begin()

while True:
    try:
        L, CL, C, CR, R = alvik.get_distance()
        T = alvik.get_distance_top()
        B = alvik.get_distance_bottom()
        print(f'T: {T} | B: {B} | L: {L} | CL: {CL} | C: {C} | CR: {CR} | R: {R}')
        sleep_ms(100)
```

```
except KeyboardInterrupt as e:
    print('over')
    alvik.stop()
    sys.exit()
```

In diesem Beispiel verwendet der Roboter seine Abstandssensoren, um Entfernungen in seiner Umgebung zu messen. Die Entfernungsanzeigen der Sensoren links, Mitte-links, Mitte, Mitte-rechts, rechts, oben und unten werden kontinuierlich gedruckt, so dass der Roboter Hindernisse erkennen und Entfernungen genau messen kann.

Einer Linie folgen

Zeilensensor lesen

Die Funktion `get_line_sensors` gibt die Anzeigen der Line Follower-Sensoren zurück und liefert den Status des linken, mittleren und rechten Sensors.

Ausgaben:

- `left`: Linke Sensoranzeige.
- `center`: Anzeige des mittleren Sensors.
- `right`: Anzeige des rechten Sensors.

Benutzungsbeispiel

Hier ist ein Beispiel dafür, wie Sie die Sensoren des Linienverfolgers verwenden, damit der Roboter einer Linie folgt:

```
from arduino_alvik import ArduinoAlvik
from time import sleep_ms
import sys

def calculate_center(left: int, center: int, right: int):
    centroid = 0
    sum_weight = left + center + right
    sum_values = left + 2 * center + 3 * right
    if sum_weight != 0:
        centroid = sum_values / sum_weight
        centroid = 2 - centroid
    return centroid

alvik = ArduinoAlvik()
alvik.begin()

error = 0
control = 0
kp = 50.0

alvik.left_led.set_color(0, 0, 1)
alvik.right_led.set_color(0, 0, 1)
```

```

while alvik.get_touch_ok():
    sleep_ms(50)

while not alvik.get_touch_ok():
    sleep_ms(50)

try:
    while True:
        while not alvik.get_touch_cancel():

            line_sensors = alvik.get_line_sensors()
            print(f' {line_sensors}')

            error = calculate_center(*line_sensors)
            control = error * kp

            if control > 0.2:
                alvik.left_led.set_color(1, 0, 0)
                alvik.right_led.set_color(0, 0, 0)
            elif control < -0.2:
                alvik.left_led.set_color(1, 0, 0)
                alvik.right_led.set_color(0, 0, 0)
            else:
                alvik.left_led.set_color(0, 1, 0)
                alvik.right_led.set_color(0, 1, 0)

            alvik.set_wheels_speed(30 - control, 30 + control)
            sleep_ms(100)

        while not alvik.get_touch_ok():
            alvik.left_led.set_color(0, 0, 1)
            alvik.right_led.set_color(0, 0, 1)
            alvik.brake()
            sleep_ms(100)

except KeyboardInterrupt as e:
    print('over')
    alvik.stop()
    sys.exit()

```

In diesem Beispiel verwendet der Roboter sein Line-Follower-Sensorfeld, um entlang einer Linie zu navigieren. Die Funktion `calculate_center` bestimmt den Mittelpunkt der Linienposition und der Roboter passt seine Radgeschwindigkeiten an, um auf der Linie zu bleiben, basierend auf dem berechneten Fehler von diesem Mittelpunkt.

Farben wahrnehmen

Verfügbare Funktionen

1. **Kalibrieren** Sie den Sensor für einen bestimmten Hintergrund

Die Funktion `color_calibration` wird verwendet, um den Farbsensor für eine genaue Farberkennung zu kalibrieren. Die Kalibrierung kann vor einem weißen oder schwarzen Hintergrund durchgeführt werden.

```
color_calibration(background: str = 'white')
```

Eingaben: `background`: Eine Zeichenkette, die die Hintergrundfarbe für die Kalibrierung angibt. Kann "weiß" oder "schwarz" sein.

2. Ruft den **rohen** Sensorwert ab

Die Funktion `get_color_raw` gibt die Rohdaten des Farbsensors zurück. Dies sind die direkten Sensordaten vor jeglicher Verarbeitung oder Beschriftung. Sie kann für fortgeschrittene Anwendungen nützlich sein, bei denen präzise Farbdaten benötigt werden.

```
get_color_raw()
```

Ausgaben: `color`: Die Rohdaten des Farbsensors.

3. Ruft den **etikettierten** Sensorwert ab

Die Funktion `get_color_label` gibt die Bezeichnung der Farbe zurück, die der Sensor erkannt hat. Diese Funktion verarbeitet die rohen Sensordaten und wandelt sie in eine für den Menschen lesbare Bezeichnung um, wie z.B. "rot", "blau", "grün", usw. Sie vereinfacht die Verwendung von Farbdaten für die meisten Anwendungen. Allerdings geht dabei etwas Flexibilität verloren, da die Farben zur nächstgelegenen beschrifteten Farbe gruppiert werden.

```
get_color_label()
```

Ausgaben: `color`: Die Bezeichnung der Farbe, wie sie vom Sensor erkannt wurde. Dies können sein: 'BLACK', 'GREY', 'BLACK', 'LIGHT GREY', 'WHITE', 'YELLOW', 'LIGHT GREEN', 'BLUE', 'VIOLET', 'BROWN', 'ORANGE', 'RED'.

Benutzungsbeispiel

Hier ein Beispiel für die Verwendung der Funktion `get_color_label` in einem Skript, das den Roboter dazu bringt, in einer geraden Linie zu laufen, drei verschiedene Farben (ohne Weiß) zu erkennen, anzuhalten und die erkannten Farben jede Sekunde über die serielle Schnittstelle zu übermitteln:

```
from arduino_alvik import ArduinoAlvik
import time
from time import sleep

# Initialization
alvik = ArduinoAlvik()
```

```

alvik.begin()
sleep(5) # Waiting for the robot to setup

# Calibrate color sensor for white
alvik.color_calibration('white')

# Main logic
detected_colors = set()

print("Starting to move and detect colors...")

try:
    while len(detected_colors) < 3:
        alvik.set_wheels_speed(20, 20)
        color = alvik.get_color_label()

        if color != 'WHITE' and color not in detected_colors:
            detected_colors.add(color)
            print(f"Detected color: {color}")

        time.sleep(0.1) # Adjust the sleep time as needed

    alvik.brake()
    print("Detected three different colors. Stopping...")

    # Communicate the detected colors via serial every second
    while True:
        print(f"Detected colors: {' '.join(detected_colors)}")
        time.sleep(1)
except KeyboardInterrupt:
    alvik.brake()
    print("Interrupted. Stopping the robot.")
except Exception as e:
    alvik.brake()
    print(f"An error occurred: {e}")

```

4. get_color

Die Funktion `get_color` gibt die normalisierte Farbanzeige des Farbsensors zurück. Diese Funktion kann die Farbe entweder im RGB- oder im HSV-Format ausgeben.

```
get_color(color_format: str = 'rgb')
```

Eingaben: `color_format`: Das Format der Farbanzeige. Kann "rgb" oder "hsv" sein.

Ausgaben: `r` oder `h`: Die rote Komponente im RGB-Format oder der Farbton im HSV-Format. `g` oder `s`: Die Grünkomponente im RGB-Format oder die Sättigung im HSV-Format. `b` oder `v`: Die Blaukomponente im RGB-Format oder der Wert im HSV-Format.

Sie können diese Funktionen je nach Ihren Bedürfnissen verwenden:

- Verwenden Sie `color_calibration`, um den Sensor für genaue Messwerte zu kalibrieren.
- Verwenden Sie `get_color_raw`, um die Rohdaten des Sensors zu erhalten.
- Verwenden Sie `get_color_label`, um eine für Menschen lesbare Bezeichnung der erkannten Farbe zu erhalten.
- Verwenden Sie `get_color`, um normalisierte Farbdaten im RGB- oder HSV-Format zu erhalten.
- Verwenden Sie `hsv2label`, um HSV-Werte in eine Farbbezeichnung zu konvertieren.

Erkennung von Stürzen und Abstürzen (IMU)

Die Funktion `get_imu` aus der [Alvik API](#) gibt alle Messwerte der IMU zurück. Die Anzeigen umfassen die Beschleunigung (ax, ay, az) und die Winkelbeschleunigung (gx, gy, gz) auf der x-, y- und z-Achse.

Verfügbare Funktionen

1. Ruft die Orientierungswerte der IMU wie **Rollen**, **Neigung** und **Gieren** ab.

Die Funktion `get_orientation()` gibt die Orientierung der IMU zurück, einschließlich der Werte für Roll, Nick und Gier. Dies kann nützlich sein, um die Ausrichtung des Roboters im 3D-Raum zu bestimmen.

Ausgaben: **r**: Rollwert **p**: Nickwert **y**: Gierwert

2. Ruft 3-Achsen-Beschleunigungswerte ab

Die Funktion `get_accelerations()` gibt die 3-achsige Beschleunigung der IMU zurück. Sie liefert die Beschleunigungswerte entlang der x-, y- und z-Achse.

Ausgaben: **ax**: Beschleunigung auf x **ay**: Beschleunigung auf y **az**: Beschleunigung auf z

3. Ruft die Werte der 3-achsigen Winkelbeschleunigung ab

Die Funktion `get_gyros()` gibt die 3-achsige Winkelbeschleunigung der IMU zurück. Sie liefert die Werte der Winkelbeschleunigung entlang der x-, y- und z-Achse.

Ausgaben: **gx**: Winkelbeschleunigung in x-Richtung **gy**: Winkelbeschleunigung in y-Richtung **gz**: Winkelbeschleunigung auf z

4. Ruft alle IMU-Anzeigen einschließlich Beschleunigung und Winkelbeschleunigung ab

Die Funktion `get_imu()` gibt alle Messwerte der IMU zurück, einschließlich der Werte für die Beschleunigung und die Winkelbeschleunigung.

Ausgaben: **ax**: Beschleunigung auf x **ay**: Beschleunigung auf y **az**: Beschleunigung auf z **gx**: Winkelbeschleunigung an x **gy**: Winkelbeschleunigung auf y **gz**: Winkelbeschleunigung auf z

Beispiel

Hier ist ein kombiniertes Beispiel, das alle IMU-Funktionen nutzt:

```
from arduino_alvik import ArduinoAlvik
import time
```



```
# Initialization
alvik = ArduinoAlvik()
alvik.begin()

# Give some time for initialization
time.sleep(2)

# Get orientation
orientation = alvik.get_orientation()
roll = orientation['r']
pitch = orientation['p']
yaw = orientation['y']
print(f"Orientation - Roll: {roll}, Pitch: {pitch}, Yaw: {yaw}")

# Get accelerations
accelerations = alvik.get_accelerations()
ax = accelerations['ax']
ay = accelerations['ay']
az = accelerations['az']
print(f"Acceleration - X: {ax}, Y: {ay}, Z: {az}")

# Get gyros
gyros = alvik.get_gyros()
gx = gyros['gx']
gy = gyros['gy']
gz = gyros['gz']
print(f"Gyro - X: {gx}, Y: {gy}, Z: {gz}")

# Combine all IMU readings
imu_readings = alvik.get_imu()
ax = imu_readings['ax']
ay = imu_readings['ay']
az = imu_readings['az']
gx = imu_readings['gx']
gy = imu_readings['gy']
gz = imu_readings['gz']

print(f"IMU Readings - Acceleration: X: {ax}, Y: {ay}, Z: {az}, Gyro: X: {gx}, Y: {gy}, Z: {gz}")
```

Da wir nun wissen, wie die IMU funktioniert und wie man mit ihr interagiert, wollen wir dieses Wissen in einem praktischen Szenario anwenden. Ereignisse wie Stürze und Zusammenstöße sind für einen Roboter kritisch, da sie oft auf plötzliche Änderungen der linearen Beschleunigung (Bewegung entlang der x-, y- oder z-Achse) oder der Winkelgeschwindigkeit (Rotationsbewegung um diese Achsen) zurückzuführen sind. Indem wir diese spezifischen Veränderungen überwachen, können wir solche Vorfälle effektiv erkennen und entsprechend reagieren.

Im folgenden Beispiel erstellen wir einen Sketch, der die IMU verwendet, um Stürze und Zusammenstöße zu erkennen. Der Sketch sucht nach plötzlichen Ausschlägen in der linearen Beschleunigung oder der

Winkelgeschwindigkeit und verwendet festgelegte Schwellenwerte, um genau zu bestimmen, wann diese Ereignisse auftreten.

Hier sehen Sie, wie Sie diese Funktionalität implementieren können:

```
from arduino_alvik import ArduinoAlvik
import time

# Initialize the Alvik robot
alvik = ArduinoAlvik()
alvik.begin()

# Function to detect falling and crashes
def detect_fall_or_crash():
    imu_readings = alvik.get_imu()

    ax = imu_readings['ax']
    ay = imu_readings['ay']
    az = imu_readings['az']
    gx = imu_readings['gx']
    gy = imu_readings['gy']
    gz = imu_readings['gz']

    # Thresholds for detecting a fall or crash (example values)
    acceleration_threshold = 2.5
    angular_acceleration_threshold = 1.0

    if abs(ax) > acceleration_threshold or abs(ay) > acceleration_threshold or
abs(az) > acceleration_threshold:
        print("Fall or crash detected based on acceleration!")

    if abs(gx) > angular_acceleration_threshold or abs(gy) >
angular_acceleration_threshold or abs(gz) > angular_acceleration_threshold:
        print("Fall or crash detected based on angular acceleration!")

while True:
    detect_fall_or_crash()
    time.sleep(0.1) # Adjust the sleep time as needed
```

Jetzt, wo Sie die Grundlagen beherrschen, warum gehen Sie nicht einen Schritt weiter? Sie können selbst bestimmen, wie Alvik auf Stürze und Unfälle reagiert. Ob Sie eine Warnung senden, das Ereignis für eine spätere Analyse protokollieren oder ein Rettungsmanöver durchführen - die Möglichkeiten sind endlos.

Und seien wir ehrlich, es ist eine gewisse Befriedigung, Ihren Roboter so zu programmieren, dass er Pannen elegant (oder sogar dramatisch) bewältigt. Dabei geht es nicht nur um Funktionalität - manchmal geht es auch darum, unseren Robotern das gewisse Extra an Persönlichkeit zu verleihen und sich, seien wir ehrlich, ein wenig überlegen zu fühlen, wenn unsere Kreationen stolpern. Nur zu, haben Sie Spaß daran und sehen Sie, welche kreativen Lösungen Sie sich einfallen lassen, wenn Alvik stolpert!

LEDs

Der Arduino Alvik Roboter ist mit zwei RGB-LEDs ausgestattet, die so gesteuert werden können, dass sie verschiedene Farben anzeigen. Diese LEDs können zur Signalisierung, zur Anzeige des Status oder einfach nur zum Spaß verwendet werden. Die folgenden Funktionen stehen für die Steuerung der LEDs zur Verfügung:

Farbe der RGB-LEDs von Alvik einstellen

Die Funktion `set_color` stellt die Farbe der LED ein, indem sie die Intensität der roten, grünen und blauen Komponenten angibt.

Eingaben: `red`: Intensität der roten Komponente (0 oder 1) `green`: Intensität der grünen Komponente (0 oder 1) `blue`: Intensität des Blauanteils (0 oder 1)

In diesem Fall nehmen die LEDs **boolesche** Werte für jede Farbe auf, was bedeutet, dass Kombinationen möglich sind, indem jede Farbe **an** oder **aus** gesetzt wird, anstatt die Intensität zu steuern.

Beispiel

Hier ist ein Beispiel-Sketch, der verschiedene Farben auf beiden LEDs durchläuft:

```
from arduino_alvik import ArduinoAlvik
from time import sleep
import sys

alvik = ArduinoAlvik()
alvik.begin()
sleep(5)

while True:
    alvik.left_led.set_color(0, 0, 0)
    alvik.right_led.set_color(0, 0, 0)
    sleep(1)
    alvik.left_led.set_color(0, 0, 1)
    alvik.right_led.set_color(0, 0, 1)
    sleep(1)
    alvik.left_led.set_color(0, 1, 0)
    alvik.right_led.set_color(0, 1, 0)
    sleep(1)
    alvik.left_led.set_color(1, 0, 0)
    alvik.right_led.set_color(1, 0, 0)
    sleep(1)
```

Wenn Sie diese Schritte befolgen, können Sie die LEDs des Arduino Alvik Roboters so steuern, dass sie verschiedene Farben anzeigen und verschiedene Lichteffekte erzeugen.

Sprechen mit anderen Geräten!

WiFi

Der ESP32 des Arduino Alvik-Roboters verfügt über integrierte Wi-Fi-Funktionen, mit denen er sich mit drahtlosen Netzwerken verbinden und über das Internet mit anderen Geräten kommunizieren kann. Dies kann besonders für Fernsteuerungsanwendungen nützlich sein. In diesem Beispiel richten wir den Alvik-Roboter so

ein, dass er sich mit einem Wi-Fi-Netzwerk verbindet, einen Webserver hostet und eine einfache Weboberfläche mit Tasten zur Steuerung der Bewegungen des Roboters bereitstellt.

Der bereitgestellte Code wird:

1. Verbinden Sie den ESP32 mit einem bestimmten Wi-Fi-Netzwerk.
2. Richten Sie einen Webserver ein, der auf eingehende HTTP-Anfragen wartet.
3. Stellen Sie eine einfache Webseite mit Tasten für die Richtungssteuerung bereit.
4. Bewegen Sie den Roboter in die angegebene Richtung, wenn eine Taste gedrückt wird.

Schrittweise Konfiguration

1. Ersetzen Sie `Insert_SSID_Here` und `Password_Here` durch Ihre Wi-Fi-Anmeldedaten.
2. Laden Sie das Skript auf den Arduino Nano ESP32 hoch.
3. Verbinden Sie sich über einen Webbrowser mit der IP-Adresse des Roboters (die auf der Konsole aufgedruckt ist).
4. Steuern Sie die Bewegung des Roboters über das Web-Interface.

Verbindung zum Wi-Fi

Der folgende Code verbindet den ESP32 mit einem bestimmten Wi-Fi-Netzwerk unter Verwendung der angegebenen SSID und des Passworts.

```
import network
import socket
from arduino_alvik import ArduinoAlvik
from time import sleep_ms

# Wi-Fi credentials
SSID = "Insert_SSID_Here"
PASSWORD = "Password_Here"

# Connect to Wi-Fi
sta_if = network.WLAN(network.STA_IF)
sta_if.active(True)
sta_if.connect(SSID, PASSWORD)

# Wait for connection
while not sta_if.isconnected():
    pass

print("Connected to WiFi. IP address:", sta_if.ifconfig()[0])
```

Ersetzen Sie `Insert_SSID_Here` und `Password_Here` durch die SSID und das Passwort Ihres Wi-Fi-Netzwerks, damit sich der ESP32 mit Ihrem Wi-Fi verbinden kann.

Initialisierung des Roboters

Der folgende Code initialisiert den Alvik-Roboter und gibt ihm Zeit, sich einzurichten.

```
# Initialize the robot
alvik = ArduinoAlvik()
alvik.begin()
sleep_ms(5000)
```

Einrichten des Webservers

Der folgende Code definiert den HTML-Code für die Webschnittstelle. Sie können ihn nach Ihrem Geschmack anpassen.

```
# HTML for the web interface
html = """
<!DOCTYPE html>
<html>
<head>
<title>Alvik Robot Control</title>
</head>
<body>
<h1>Control Alvik Robot</h1>
<form action="/up">
  <button type="submit">Up</button>
</form>
<form action="/down">
  <button type="submit">Down</button>
</form>
<form action="/left">
  <button type="submit">Left</button>
</form>
<form action="/right">
  <button type="submit">Right</button>
</form>
</body>
</html>
"""
```

Zum Schluss richten wir den Webserver ein. Beachten Sie, dass der gewählte Port für Ihr Netzwerk gültig sein sollte und keine Konflikte auftreten. Sie können hier die Einstellung wählen, die für Ihre Netzwerkkonfiguration am besten geeignet ist.

```
# Specify a port
port = 8080
addr = socket.getaddrinfo('0.0.0.0', port)[0][-1]
s = socket.socket()
s.bind(addr)
s.listen(5)

s.settimeout(5)
```

```
print(f'Listening on {sta_if.ifconfig()[0]}:{port}')
```

Verarbeitung von HTTP-Anfragen

Der folgende Code verarbeitet eingehende HTTP-Anfragen und bewegt den Roboter in die angegebene Richtung. Beachten Sie, dass der Roboter die Aktion so lange fortsetzt, bis die Pause-Taste gedrückt wird.

```
def handle_request(conn):
    try:
        request = conn.recv(1024).decode('utf-8')
        first_line = request.split('\n')[0] # Get the first line of the request
        path = first_line.split(' ')[1] # Extract the path (e.g., "/up")

        # Strip query strings (e.g., "/up?")
        if '?' in path:
            path = path.split('?')[0]
        print(f"Request path: {path}")

        # Ignore favicon requests
        if path == '/favicon.ico':
            conn.send('HTTP/1.1 204 No Content\n')
            conn.send('Connection: close\n\n')
            return

        # Control the robot based on the request path
        if path == '/up':
            alvik.set_wheels_speed(30, 30)
        elif path == '/down':
            alvik.set_wheels_speed(-30, -30)
        elif path == '/left':
            alvik.set_wheels_speed(-30, 30)
        elif path == '/right':
            alvik.set_wheels_speed(30, -30)
        else:
            alvik.brake()

        # Send the response for valid paths
        conn.send('HTTP/1.1 200 OK\n')
        conn.send('Content-Type: text/html\n')
        conn.send('Connection: close\n\n')
        conn.sendall(html)

    except OSError as e:
        if e.errno == 104: # ECONNRESET error
            print("Connection reset by client.")
        else:
            print(f"Error: {e}")
    finally:
        conn.close()
```

Hauptschleife

Der folgende Code führt die Hauptschleife aus, um eingehende Verbindungen zu behandeln und HTTP-Anfragen zu verarbeiten.

```
# Main loop to handle incoming connections
try:
    while True:
        try:
            conn, addr = s.accept()
            print('Connection from', addr)
            handle_request(conn)
        except OSError as e:
            if e.errno == 116: # ETIMEDOUT error
                print("Waiting for connection...")
            elif e.errno == 104: # ECONNRESET error
                print("Connection reset by client.")
            else:
                print(f"Accept error: {e}")
except KeyboardInterrupt:
    print("Server stopped")
    s.close()
    alvik.stop()
```

Dieser Abschnitt stellt sicher, dass der Server weiterläuft, eingehende Verbindungen annimmt und Anfragen bearbeitet, bis das Skript unterbrochen wird. Wenn ein `EADDRINUSE`-Fehler auftritt, ändern Sie die Portnummer von 80 in einen unbenutzten Port, wie z.B. 8080, in der Zeile `addr = socket.getaddrinfo('0.0.0.0', 80)[0][-1]`. Stellen Sie außerdem sicher, dass sich das Gerät, mit dem Sie auf die Weboberfläche zugreifen, im selben Netzwerk befindet wie der Alvik.

Vollständiger Code

```
import network
import socket
from arduino_alvik import ArduinoAlvik
from time import sleep_ms

# Wi-Fi credentials
SSID = "Insert_SSID_Here"
PASSWORD = "Insert_Password_Here"

# Connect to Wi-Fi
sta_if = network.WLAN(network.STA_IF)
sta_if.active(True)
sta_if.connect(SSID, PASSWORD)

# Wait for connection
while not sta_if.isconnected():
    pass
```

```
print("Connected to WiFi. IP address:", sta_if.ifconfig()[0])

# Initialize the robot
alvik = ArduinoAlvik()
alvik.begin()
sleep_ms(5000)

# HTML for the web interface
html = """
<!DOCTYPE html>
<html>
<head>
<title>Alvik Robot Control</title>
</head>
<body>
<h1>Control Alvik Robot</h1>
<form action="/up">
    <button type="submit">Up</button>
</form>
<form action="/down">
    <button type="submit">Down</button>
</form>
<form action="/left">
    <button type="submit">Left</button>
</form>
<form action="/right">
    <button type="submit">Right</button>
</form>
</body>
</html>
"""

# Specify a port
port = 8080
addr = socket.getaddrinfo('0.0.0.0', port)[0][-1]
s = socket.socket()
s.bind(addr)
s.listen(5)

s.settimeout(5)

print(f'Listening on {sta_if.ifconfig()[0]}:{port}')

# Function to handle incoming HTTP requests
def handle_request(conn):
    try:
        request = conn.recv(1024).decode('utf-8')
        first_line = request.split('\n')[0] # Get the first line of the request
        path = first_line.split(' ')[1] # Extract the path (e.g., "/up")

        # Strip query strings (e.g., "/up?")
        if '?' in path:
            path = path.split('?')[0]
        print(f"Request path: {path}")
```



```
# Ignore favicon requests
if path == '/favicon.ico':
    conn.send('HTTP/1.1 204 No Content\n')
    conn.send('Connection: close\n\n')
    return

# Control the robot based on the request path
if path == '/up':
    alvik.set_wheels_speed(30, 30)
elif path == '/down':
    alvik.set_wheels_speed(-30, -30)
elif path == '/left':
    alvik.set_wheels_speed(-30, 30)
elif path == '/right':
    alvik.set_wheels_speed(30, -30)
else:
    alvik.brake()

# Send the response for valid paths
conn.send('HTTP/1.1 200 OK\n')
conn.send('Content-Type: text/html\n')
conn.send('Connection: close\n\n')
conn.sendall(html)

except OSError as e:
    if e.errno == 104: # ECONNRESET error
        print("Connection reset by client.")
    else:
        print(f"Error: {e}")
finally:
    conn.close()

# Main loop to handle incoming connections
try:
    while True:
        try:
            conn, addr = s.accept()
            print('Connection from', addr)
            handle_request(conn)
        except OSError as e:
            if e.errno == 116: # ETIMEDOUT error
                print("Waiting for connection...")
            elif e.errno == 104: # ECONNRESET error
                print("Connection reset by client.")
            else:
                print(f"Accept error: {e}")
except KeyboardInterrupt:
    print("Server stopped")
    s.close()
    alvik.stop()
```

ESP-NOW

Der ESP32 auf dem Arduino Alvik Roboter unterstützt auch ESP-NOW, ein schnelles, verbindungsloses Kommunikationsprotokoll, das die direkte, schnelle und stromsparende Steuerung intelligenter Geräte ohne Router ermöglicht. ESP-NOW kann neben Wi-Fi und Bluetooth LE eingesetzt werden und ist damit vielseitig für verschiedene Anwendungen geeignet. Es unterstützt die ESP8266-, ESP32-, ESP32-S- und ESP32-C-Serie. In diesem Beispiel werden wir den Alvik-Roboter so einrichten, dass er Befehle über ESP-NOW empfängt.

Geräteidentifikation

Es ist wichtig für ESP-NOW, dass Sie die MAC-Adresse Ihres Geräts kennen.

Sie können dies tun, indem Sie den folgenden Code ausführen:

```
import network

# Initialize the network interface
e = network.WLAN(network.STA_IF)
e.active(True)

# Get and print the MAC address
mac = e.config('mac')
print("MAC address:", ':'.join('%02x' % b for b in mac))
```

Jedes Gerät hat eine andere Adresse. Identifizieren Sie Ihre Geräte, damit Sie Ihre ESP-NOW-Projekte einfach erstellen können.

Jetzt, da Sie die Adresse Ihres Geräts kennen, können wir Ihr erstes ESP-NOW-Projekt starten.

Lassen Sie uns nun eine Verbindung zwischen einem Paar Alviks herstellen. Einer wird als Controller fungieren und der andere wird den Steuerungen folgen und sich in die entsprechende Richtung bewegen.

Der bereitgestellte Code wird:

1. Initialisieren Sie ESP-NOW auf dem ESP32.
2. Richten Sie eine Callback-Funktion ein, um eingehende ESP-NOW-Nachrichten zu verarbeiten.
3. Bewegen Sie den Roboter auf der Grundlage der empfangenen Befehle, während Sie die Tasten drücken.

Schrittweise Konfiguration

1. Laden Sie das Empfängerskript auf den Arduino Nano ESP32 auf dem Alvik-Roboter.
2. Verwenden Sie ein anderes ESP32-Gerät, um über ESP-NOW Befehle an den Alvik-Roboter zu senden.

Einrichten von ESP-NOW auf dem Receiver

Der folgende Code initialisiert ESP-NOW auf dem Empfänger und richtet eine Callback-Funktion für die Bearbeitung eingehender Nachrichten ein.

```
from arduino_alvik import ArduinoAlvik
import network
import espnow

# Initialize ESP-NOW
wlan = network.WLAN(network.STA_IF)
wlan.active(True)
esp = espnow.ESPNow()
esp.active(True)

# Print the MAC address
mac = wlan.config('mac')
print("MAC address:", ':'.join('%02x' % b for b in mac))

# Initialize Alvik
alvik = ArduinoAlvik()
alvik.begin()

def perform_action(command):
    print(f"Performing action: {command}")
    if command == "MOVE_FORWARD":
        alvik.set_wheels_speed(60, 60)
    elif command == "MOVE_BACKWARD":
        alvik.set_wheels_speed(-60, -60)
    elif command == "TURN_LEFT":
        alvik.set_wheels_speed(-60, 60)
    elif command == "TURN_RIGHT":
        alvik.set_wheels_speed(60, -60)
    elif command == "STOP":
        alvik.brake()
    elif command == "ACTION":
        pass

print("Waiting for commands...")

while True:
    try:
        host, msg = esp.recv()
        if msg:
            print(f"Received message from {host}: {msg}")
            perform_action(msg.decode())
        else:
            print("No message received")
    except Exception as e:
        print(f"An error occurred: {e}")
    except KeyboardInterrupt:
        alvik.stop()
        print("Script terminated by user")
        sys.exit()
```

Einrichten von ESP-NOW auf dem Sender

Der folgende Code kann auf einem anderen Alvik verwendet werden, um Befehle an den Alvik-Roboter zu senden.

```
from arduino_alvik import ArduinoAlvik
from time import sleep_ms
import network
import espnow

# Initialize ESP-NOW
wlan = network.WLAN(network.STA_IF)
wlan.active(True)
esp = espnow.ESPNow()
esp.active(True)

# Define the peer's MAC address
peer_mac = b'\x74\x4d\xbd\x00\x00\x00' # MAC address of the receiver Alvik

# Try to add the peer, and handle the error if the peer already exists
try:
    esp.add_peer(peer_mac)
except OSError as e:
    if e.args[0] == -12395: # ESP_ERR_ESPNOW_EXIST
        print("Peer already exists.")
    else:
        raise

# Initialize Alvik
alvik = ArduinoAlvik()
alvik.begin()

# Define button to command mapping
button_commands = {
    "UP": "MOVE_FORWARD",
    "DOWN": "MOVE_BACKWARD",
    "LEFT": "TURN_LEFT",
    "RIGHT": "TURN_RIGHT",
    "OK": "STOP",
    "CANCEL": "STOP",
    "CENTER": "ACTION"
}

def send_command(command):
    esp.send(peer_mac, command.encode())
    print(f"Sent command: {command}")

print("Controller running. Press buttons to send commands.")

while True:
    try:
        if alvik.get_touch_any():
            if alvik.get_touch_up():
                send_command(button_commands["UP"])
```

```
if alvik.get_touch_down():
    send_command(button_commands[ "DOWN" ])
if alvik.get_touch_left():
    send_command(button_commands[ "LEFT" ])
if alvik.get_touch_right():
    send_command(button_commands[ "RIGHT" ])
if alvik.get_touch_ok():
    send_command(button_commands[ "OK" ])
if alvik.get_touch_cancel():
    send_command(button_commands[ "CANCEL" ])
if alvik.get_touch_center():
    send_command(button_commands[ "CENTER" ])

sleep_ms(100)
except KeyboardInterrupt:
    alvik.stop()
    sys.exit()
```

Empfängerverhalten

1. **Initialisierung von ESP-NOW:** Der Code richtet den ESP32 als Wi-Fi-Station ein und initialisiert ESP-NOW.
2. **Callback-Funktion:** Es wird eine Callback-Funktion `on_receive_msg` definiert, um eingehende Nachrichten zu verarbeiten. Diese Funktion dekodiert die Nachricht und bewegt den Roboter auf der Grundlage des empfangenen Befehls.
3. **Registrierung des Callbacks:** Die Callback-Funktion wird bei ESP-NOW registriert, um sicherzustellen, dass sie aufgerufen wird, wenn eine Nachricht empfangen wird.

Senderverhalten

1. **Initialisierung von ESP-NOW:** Der Code richtet den ESP32 als Wi-Fi-Station ein und initialisiert ESP-NOW.
2. **Hinzufügen eines Peers:** Die MAC-Adresse des Empfängers (Alvik-Roboter) wird als Peer hinzugefügt.
3. **Befehle senden:** Die Funktion `send_command` sendet Befehle an den Empfänger. Es werden Beispielbefehle bereitgestellt, um den Roboter in verschiedene Richtungen zu bewegen.

Wenn Sie diese Schritte befolgen, können Sie Ihren Arduino Alvik-Roboter über ESP-NOW, ein schnelles und effizientes Kommunikationsprotokoll, aus der Ferne steuern. Basteln Sie ruhig mit dieser Funktion, um neue und interessante Interaktionen hinzuzufügen.

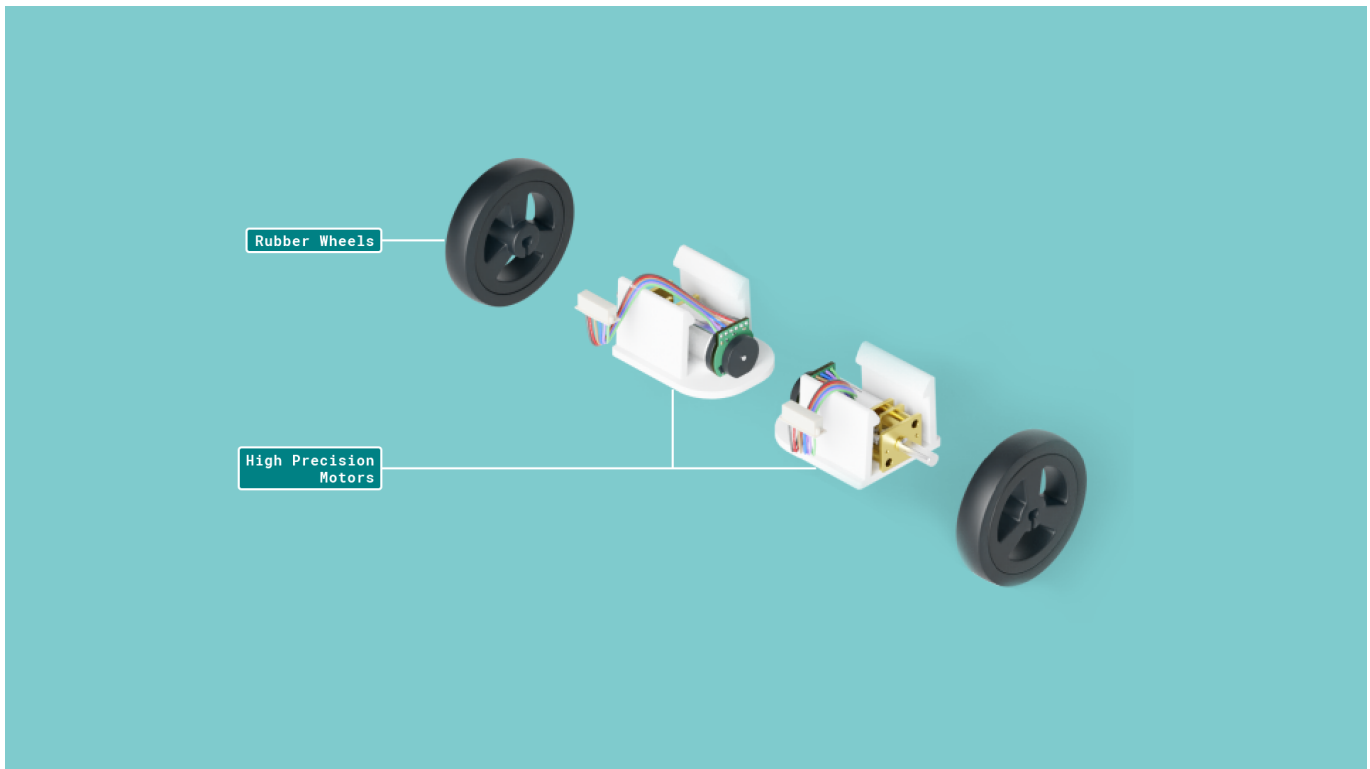
Wartung

Prüfen Sie die Ausrichtung der Räder

Die von Alvik durchgeführten Bewegungsberechnungen basieren auf der genauen Position der beiden Räder. Sie können diese Position überprüfen und korrigieren, wenn Sie glauben, dass sich Ihr Roboter nicht wie erwartet bewegt:

1. Legen Sie Ihr Alvik auf einer Seite auf eine ebene Fläche

2. Vergewissern Sie sich, dass der äußere Teil des Rades die flache Oberfläche berührt und mit dem Kunststoffgehäuse ausgerichtet ist.
3. Wiederholen Sie den Vorgang mit dem anderen Rad



Farbsensoren kalibrieren

Um den Farbsensor zu kalibrieren, der sich auf der unteren Platine unter dem Nano ESP32 befindet, benötigen Sie eine weiße und eine schwarze Fläche. Folgen Sie diesen Schritten:

1. Öffnen Sie das Arduino Lab für MicroPython
2. Verbinden Sie Alvik
3. Öffnen Sie das REPL-Terminal (klicken Sie einfach auf das Terminal-Symbol im oberen Bereich)

Jetzt können Sie Befehle direkt an Alvik senden. Jedes Mal, wenn Sie auf Enter klicken, wird der Befehl in Echtzeit ausgeführt.

```
from arduino_alvik import ArduinoAl  
  
alvik = ArduinoAlvi  
  
alvik.begin()
```

Stellen Sie nun Ihren Roboter auf die weiße Fläche und tippen Sie:

```
alvik.color_calibration('white')
```

Stellen Sie nun Ihren Roboter auf die schwarze Fläche und tippen Sie:

```
alvik.color_calibration('black')
```

Drücken Sie bei Lab for MicroPython auf Zurücksetzen.

Sie können jetzt mit `read_color_sensor.py` im Ordner `examples` testen. Lesen Sie das [Kapiteltest](#), wenn Sie Probleme haben.

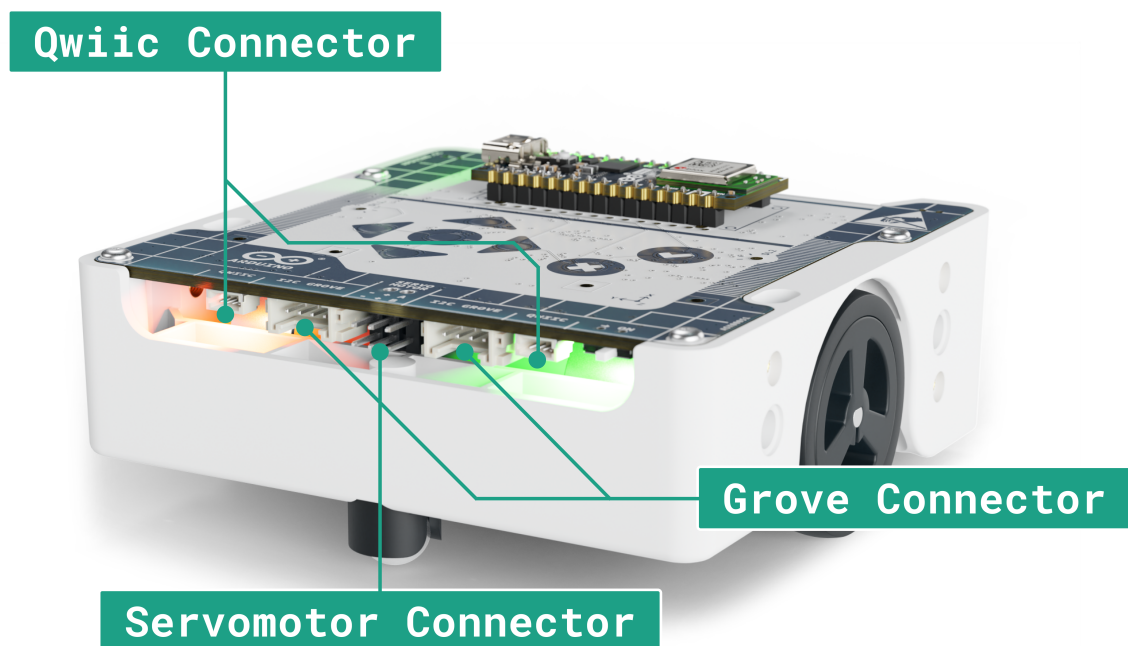
***Farben werden auf Papier getestet, das mit Acrylmarkern wie UNIPOSCA bemalt oder mit einem Tintenstrahldrucker bedruckt wurde.

Erweiterung des Roboters

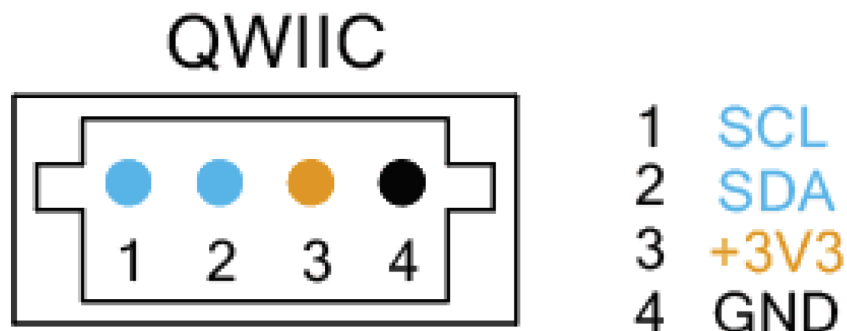
Zusätzliche Anschlüsse

Alvik enthält mehrere Konnektoren, die seine Möglichkeiten erweitern:

- **Qwiic Verbinder**
- **Grove-Steckverbinder**
- **Servomotor-Verbindungsstücke**
- **Legø Technic Kompatibilität**

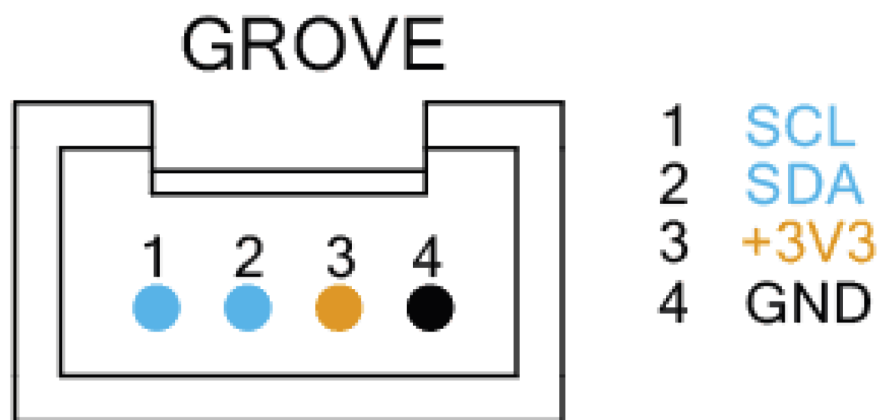


Qwiic Steckverbinder



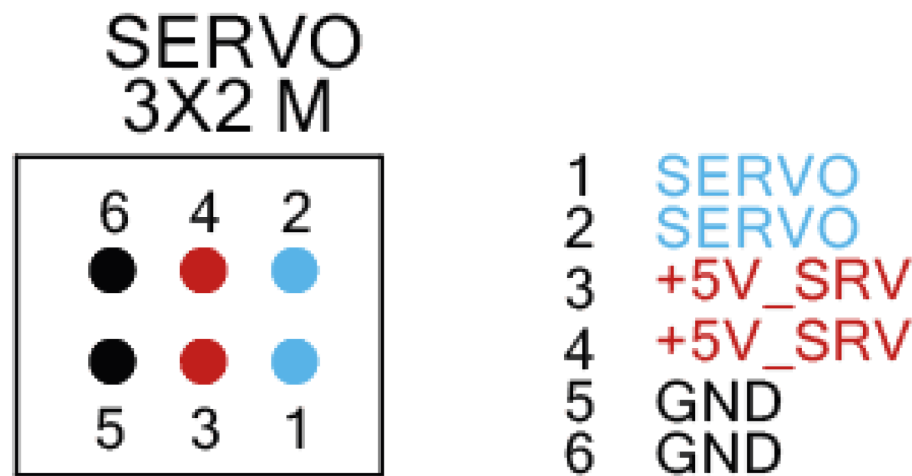
Die Qwiic-Anschlüsse des Arduino Alvik-Roboters werden für die I2C-Verbindung mit kompatiblen Sensoren und Aktuatoren nach dem Qwiic-Standard verwendet. Dieser Standard vereinfacht den Anschluss mehrerer I2C-Geräte durch die Verwendung eines gemeinsamen Anschlusses und einer gemeinsamen Pinbelegung. Das Qwiic-System unterstützt Daisy-Chaining, d.h. es können mehrere Geräte in Reihe geschaltet werden. Es gibt zwei Qwiic-Anschlüsse, einen auf jeder Seite auf der Rückseite des Roboters, die eine einfache Erweiterung und Integration zusätzlicher Komponenten ermöglichen.

Grove Steckverbinder



Die Grove-Anschlüsse am Arduino Alvik-Roboter werden für die Verbindung mit kompatiblen Sensoren und Aktuatoren nach dem Grove-Standard verwendet. Das Grove-System verwendet standardisierte Anschlüsse und Pinbelegungen, um den Anschluss verschiedener I2C-Geräte zu vereinfachen. Das System ist modular und benutzerfreundlich, ideal für Rapid Prototyping und Ausbildungszwecke. Es gibt zwei Grove-Anschlüsse, einen auf jeder Seite auf der Rückseite des Roboters, die zusätzliche Flexibilität für die Erweiterung der Fähigkeiten des Roboters bieten.

Servomotor-Steckverbinder



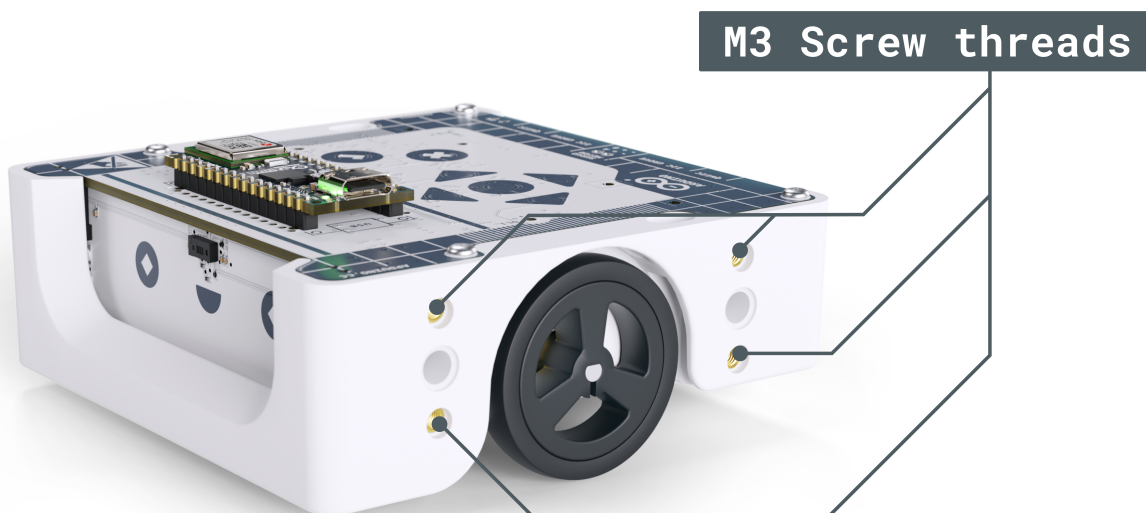
Der Arduino Alvik-Roboter verfügt über 3-Pin-Anschlüsse zum Anschließen von Servomotoren und zur PWM-Steuerung externer Servomotoren. Die Anschlüsse sind auf einer 6 x 2 großen Stiftleiste in der Mitte der Rückseite des Roboters angeordnet und ermöglichen den Anschluss mehrerer Servomotoren. Diese Anordnung ermöglicht komplexe Roboterbewegungen und -funktionen, so dass der Roboter eine Vielzahl zusätzlicher Aufgaben erfüllen kann.

Lego Technic Kompatibilität

Der Körper von Alvik ist mit Lego Technic-Steinen kompatibel und kann so erweitert und angepasst werden.

Kundenspezifische Teile

***Wenn Sie Erweiterungen am Roboter anbringen, verwenden Sie niemals Schrauben, die länger als 10 mm sind, sonst könnte das Gerät beschädigt werden.



LEGO® Addons hinzufügen

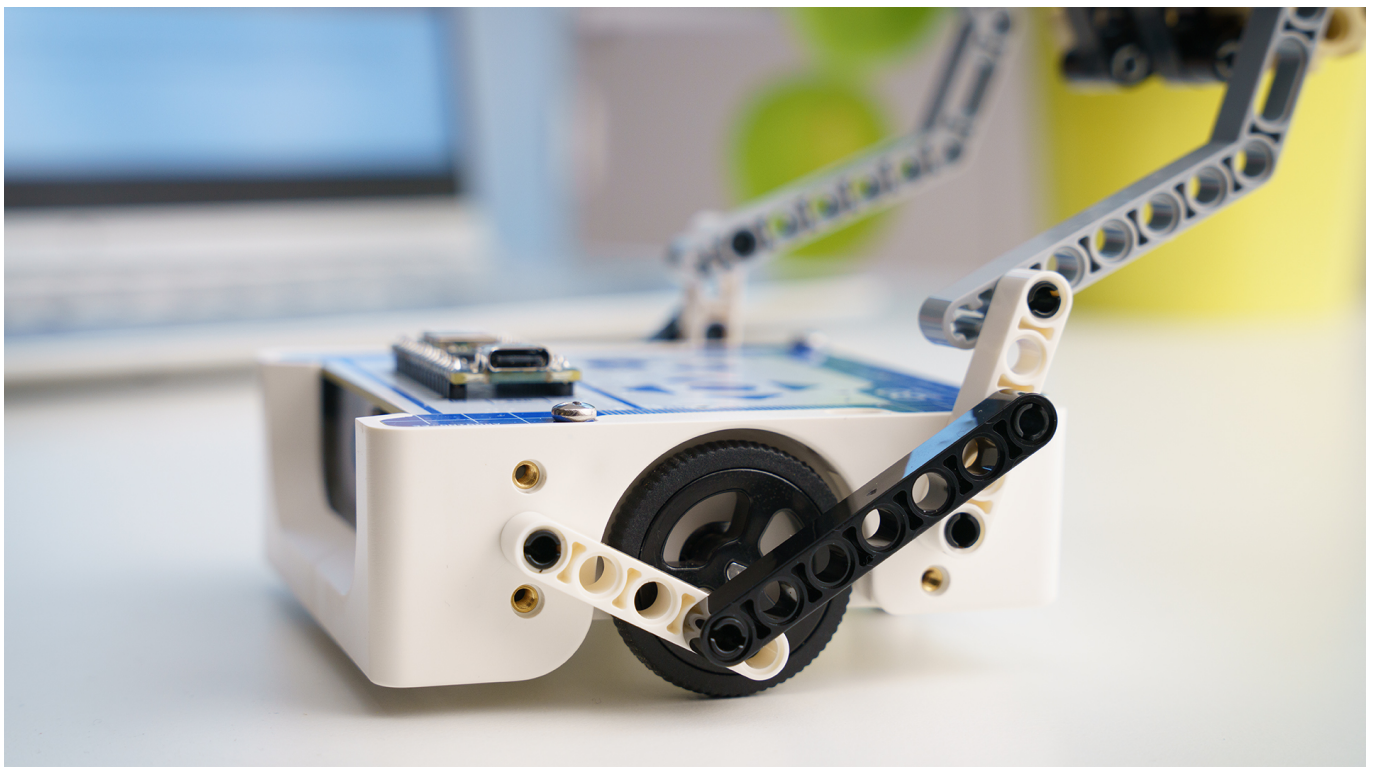
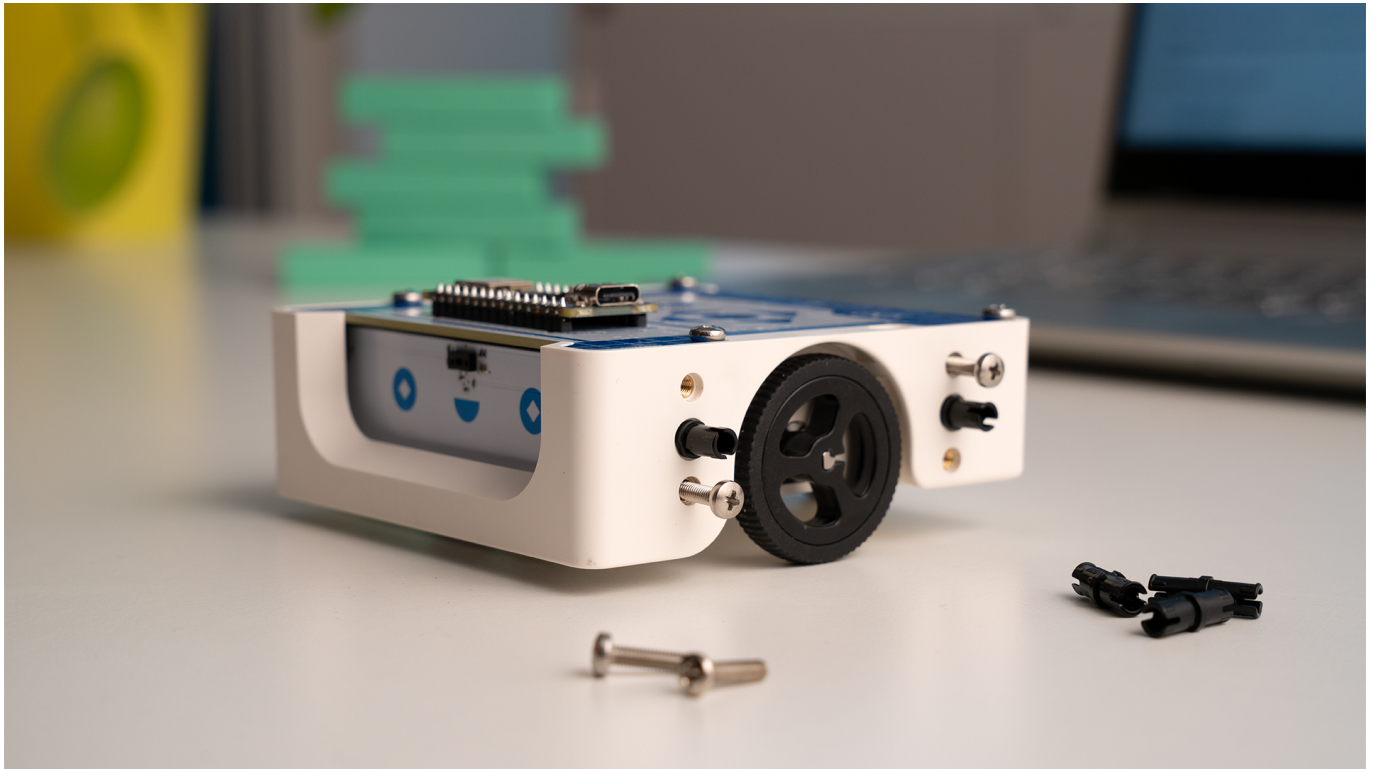
Auf beiden Seiten von Alvik gibt es verschiedene Gehäuse, die Sie hinzufügen können:

- 4x M3 Schrauben pro Seite
- 2x LEGO® Technic™ Stecker pro Seite

Die Abmessungen sind:



Als Referenz können Sie sich die folgenden Bilder ansehen:

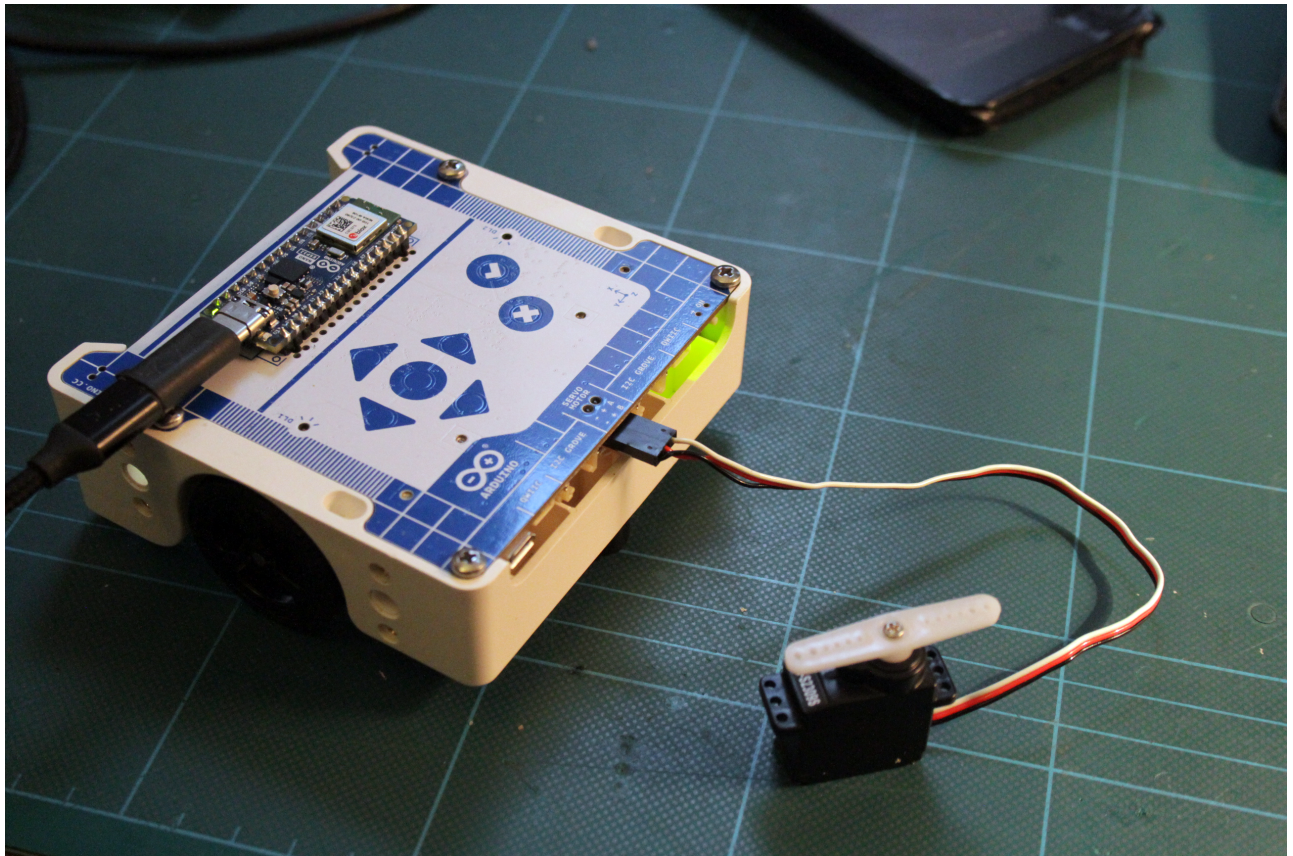


Servomotoren hinzufügen

Die Anschlüsse für die Servomotoren befinden sich auf der Rückseite von Alvik. In diesem Tutorial werden wir einen Servomotor an den Port A anschließen. Sie können sich die Pinbelegung im Kapitel [Pinbelegung](#) ansehen.

Der Anschluss liefert 5 Volt an den Motor, also stellen Sie sicher, dass Sie ein Servo anschließen, das mit 5V läuft.

1. Schließen Sie den Servomotor an den oberen Anschluss an



2. Öffnen Sie das Arduino Lab für MicroPython
3. Verbinden Sie Alvik mit einem USB-Kabel und klicken Sie auf CONNECT
4. Kopieren Sie den folgenden Testcode und fügen Sie ihn ein

```
from arduino_alvik import ArduinoAlvik
import time

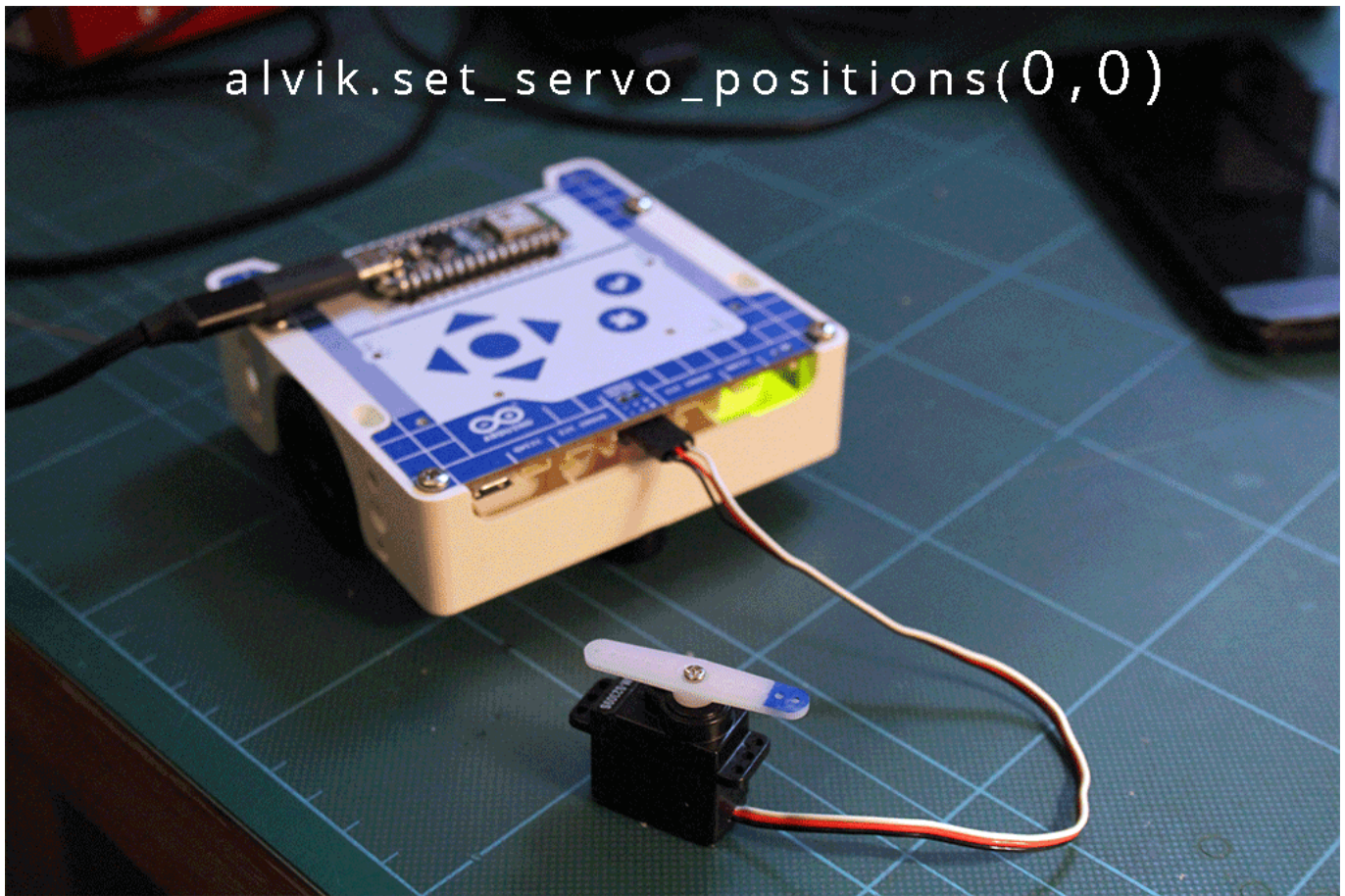
# Initialize Alvik
alvik = ArduinoAlvik()
alvik.begin()

while True:
    # Moves servo
    alvik.set_servo_positions(0,0)
    time.sleep(2)
    alvik.set_servo_positions(90,0)
    time.sleep(2)
    alvik.set_servo_positions(180,0)
    time.sleep(2)
    alvik.set_servo_positions(90,0)
    time.sleep(2)
```

5. Klicken Sie auf die Taste PLAY, um den Testcode auszuführen
6. Der Motor sollte sich wie im folgenden Bild bewegen

Klicken Sie auf die Taste PLAY, um den Testcode auszuführen.

Der Motor sollte sich wie im folgenden Bild bewegen:



Wenn Sie verstehen wollen, wie der Befehl `alvik.set_servo_positions` funktioniert, können Sie einen Blick in die [API-Übersicht](#) werfen.

I2C Grove hinzufügen

Die I2C Grove-Anschlüsse befinden sich auf der Rückseite des Alvik. In diesem Tutorial zeigen wir Ihnen, wie Sie ein allgemeines I2C-Gerät scannen können, das daran angeschlossen ist. Sie können sich die Pinbelegung auf [Alviks Produktseite](#) ansehen.

1. Schließen Sie das I2C Grove Gerät an einen der beiden Ports an.
2. Öffnen Sie das Arduino Lab für MicroPython
3. Verbinden Sie Alvik mit einem USB-Kabel und klicken Sie auf CONNECT
4. Schalten Sie Alvik ein
5. Kopieren Sie den folgenden Testcode und fügen Sie ihn ein

```
from machine import I2C
from machine import Pin

i2c = I2C(0, scl=Pin(12, Pin.OUT), sda=Pin(11, Pin.OUT))
```



```
print()
print('Scan i2c bus...')
print()

devices = i2c.scan()

if len(devices) == 0:
    print("No i2c device !")
else:
    print('i2c devices found:',len(devices))
print()

for device in devices:
    print("Decimal address: ",device," | Hexa address: ",hex(device))

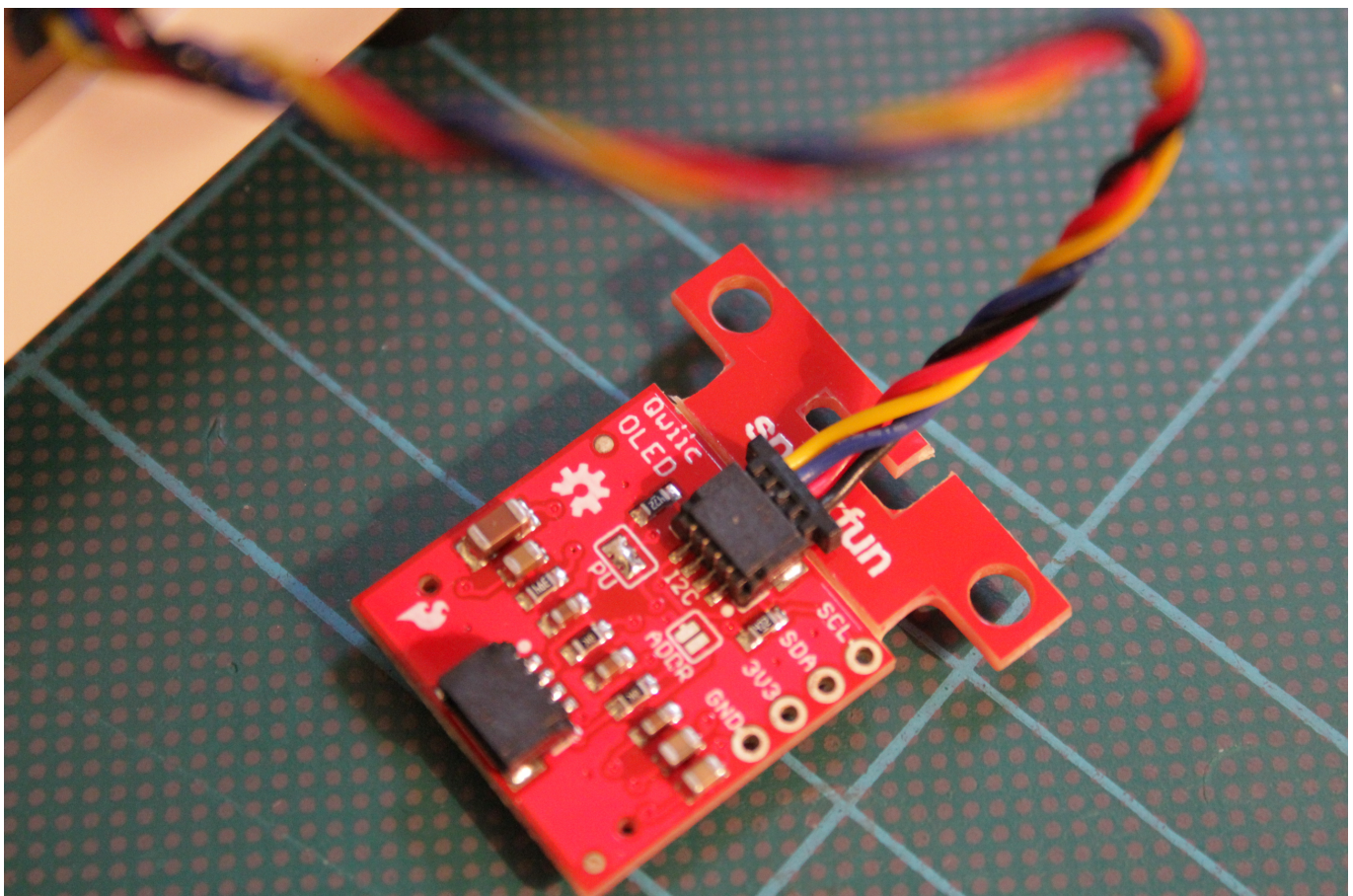
print()
```

6. Klicken Sie auf die Taste PLAY, um den Testcode auszuführen

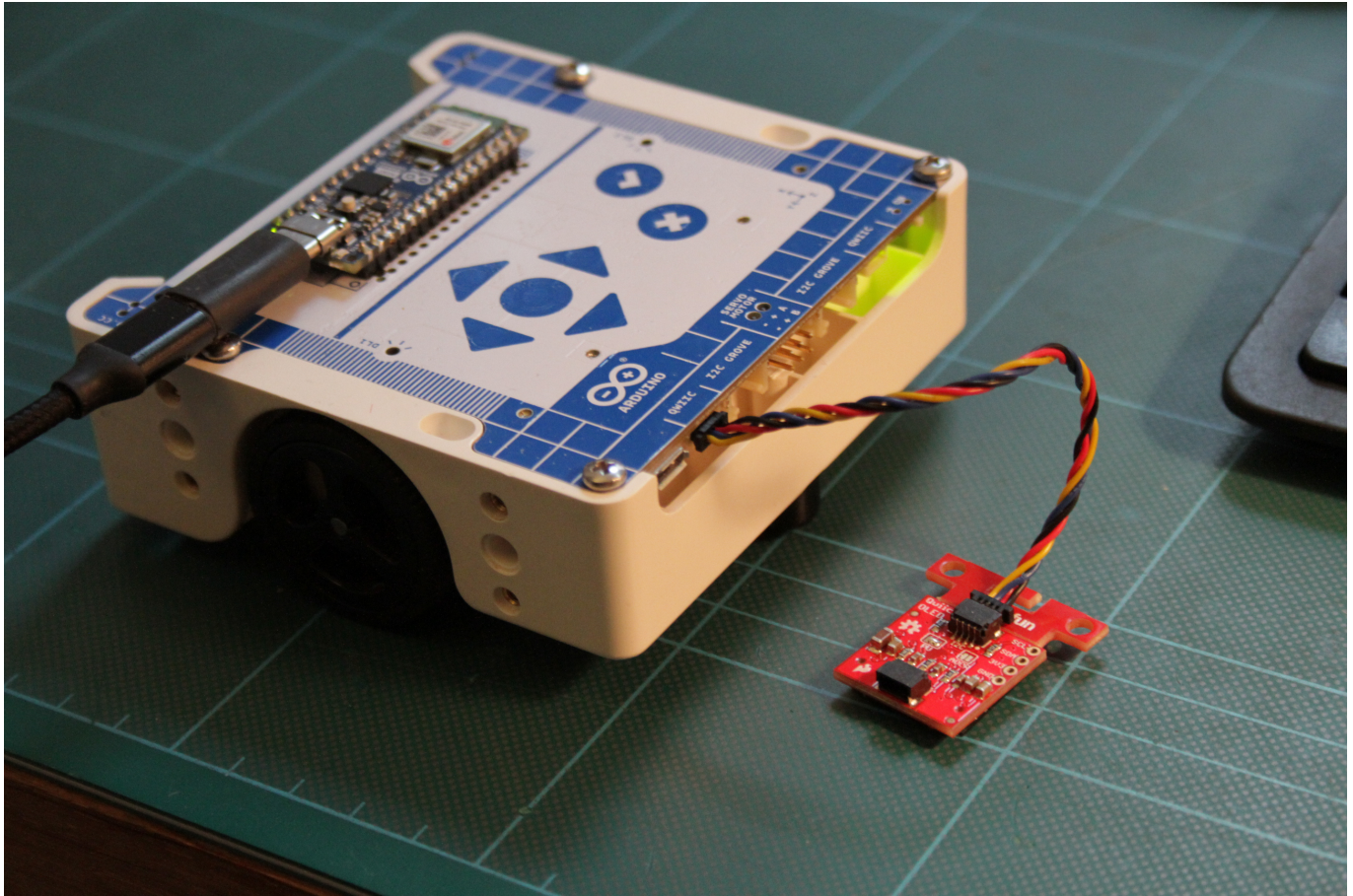
7. Schauen Sie auf das Terminal, um die Liste der I2C-Geräte zu sehen

Qwiic hinzufügen

Die Qwiic-Anschlüsse befinden sich auf der Rückseite von Alvik. Für dieses Beispiel verwenden wir das Qwiic OLED-Display von SparkFun. Sie können sich die Abbildung der Pinbelegung im Kapitel [pinout #3.2.7-connectors](#) ansehen.



1. Schließen Sie das OLED-Display an einen der Qwiic-Anschlüsse an. Sie können entweder den linken oder den rechten verwenden.



2. Wir haben den Beispielcode und die Bibliotheken in [dieser](#) Zip-Datei vorbereitet.
3. Extrahieren Sie die Dateien und öffnen Sie den Ordner
4. Installieren Sie mpremove

[mpremote](#) ist ein Python-Modul, das zum Hochladen von Dateien auf den Nano ESP32 benötigt wird. Die empfohlene Mindestversion von mpremove ist 1.22.0. Stellen Sie sicher, dass Sie Python installiert haben, bevor Sie fortfahren!

```
(venv)$ pip install mpremove
```

oder

```
(venv)$ python3 -m pip install mpremove
```

Je nachdem, wie Sie Python auf Ihrem Rechner konfigurieren.

5. Bibliothek installieren

Führen Sie die folgende Zeile aus, um alle Dateien hochzuladen und die Abhängigkeiten herunterzuladen, die für die Ausführung der Arduino Alvik MicroPython-Bibliothek erforderlich sind.

Linux

```
$ ./install_oled_lib.sh -p <device port>
```

Windows

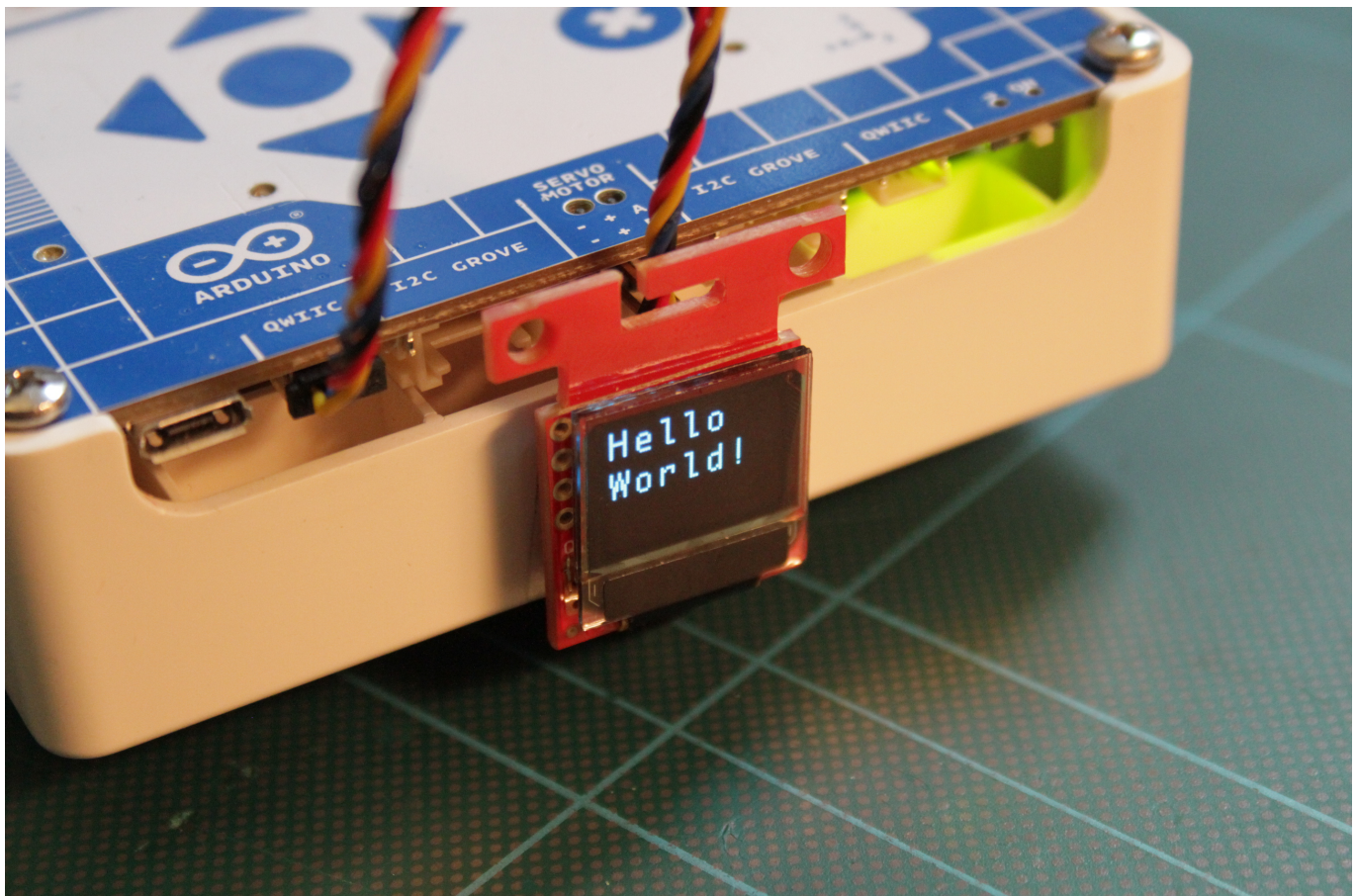
```
> install_oled_lib.bat -p <device port>
```

Der `<device port>` ist der Name des USB-Anschlusses, den Ihr Computer dem Nano ESP32 zugewiesen hat. Es gibt verschiedene Möglichkeiten, ihn zu finden, abhängig von Ihrem Betriebssystem, zum Beispiel:

Sie können die Arduino IDE verwenden, um den Port zu ermitteln, indem Sie [dieser Anleitung folgen](#). Sie können ihn mit dem Arduino Lab für MicroPython ermitteln, indem Sie auf [Verbinden](#) klicken, nachdem Sie den Alvik mit dem USB-Kabel angeschlossen haben. Sie können sich die Liste der an den PC angeschlossenen USB-Geräte ansehen

6. Test `Hello World!`

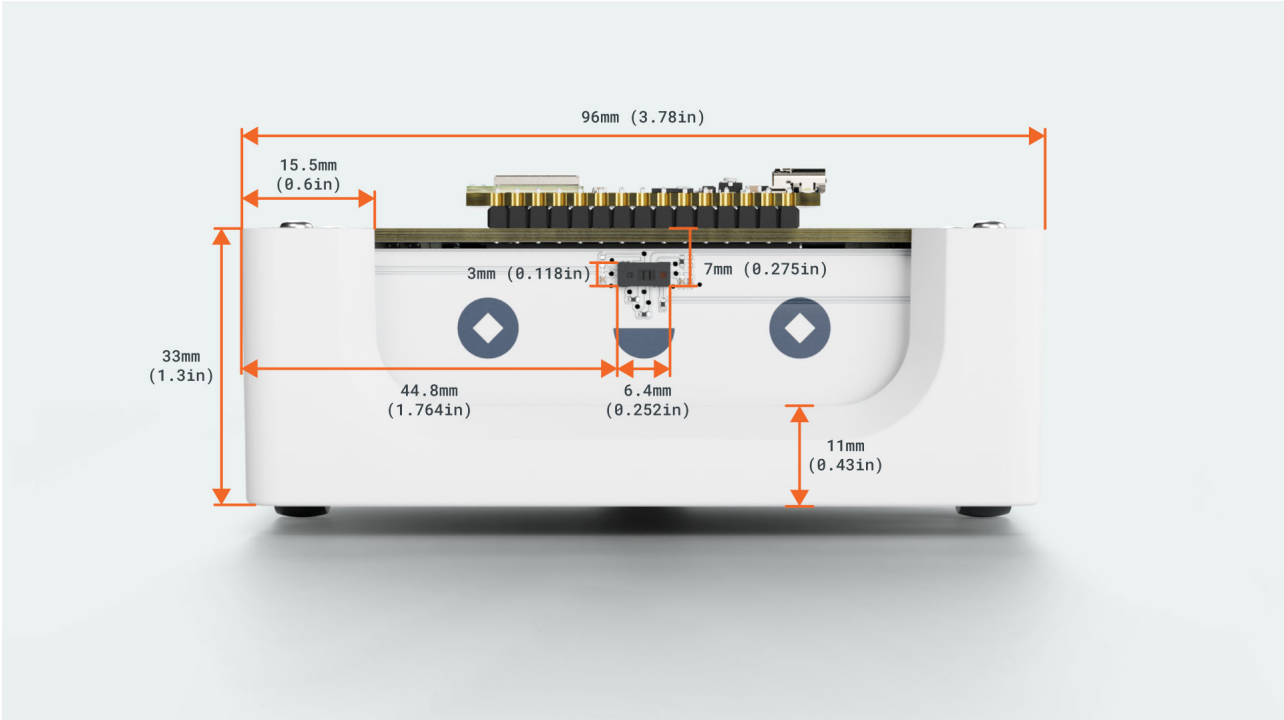
Jetzt können Sie das Arduino Lab für MicroPython öffnen, Alvik verbinden und das Beispiel namens `hello_world.py` im Ordner `examples` öffnen. Wenn alles wie erwartet funktioniert, sehen Sie etwas wie das folgende Bild:



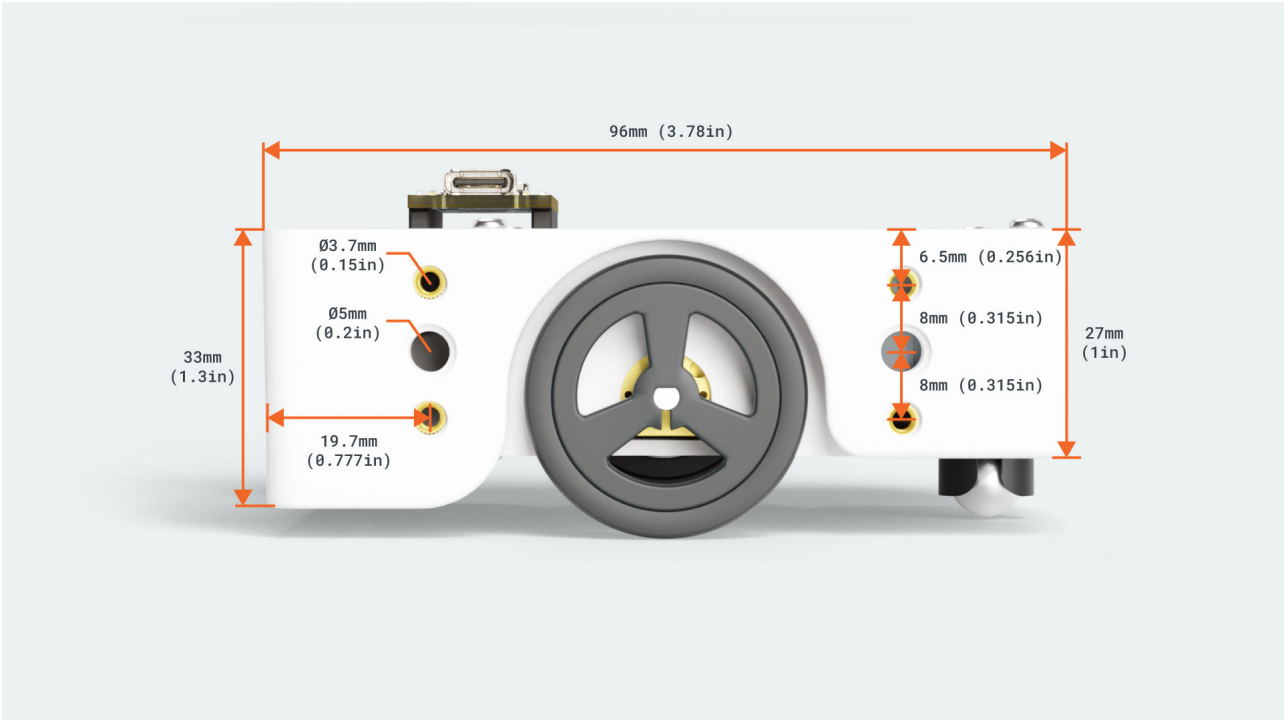
Mechanische Informationen

Das Verständnis der Abmessungen und mechanischen Details von Alvik ist der Schlüssel für die Entwicklung kundenspezifischer Komponenten und die Gewährleistung, dass Alvik gut mit anderer Hardware zusammenarbeitet. Diese Informationen sollen Ihnen dabei helfen, Alvik einfach zu erstellen und an Ihre speziellen Projekte und Bedürfnisse anzupassen.

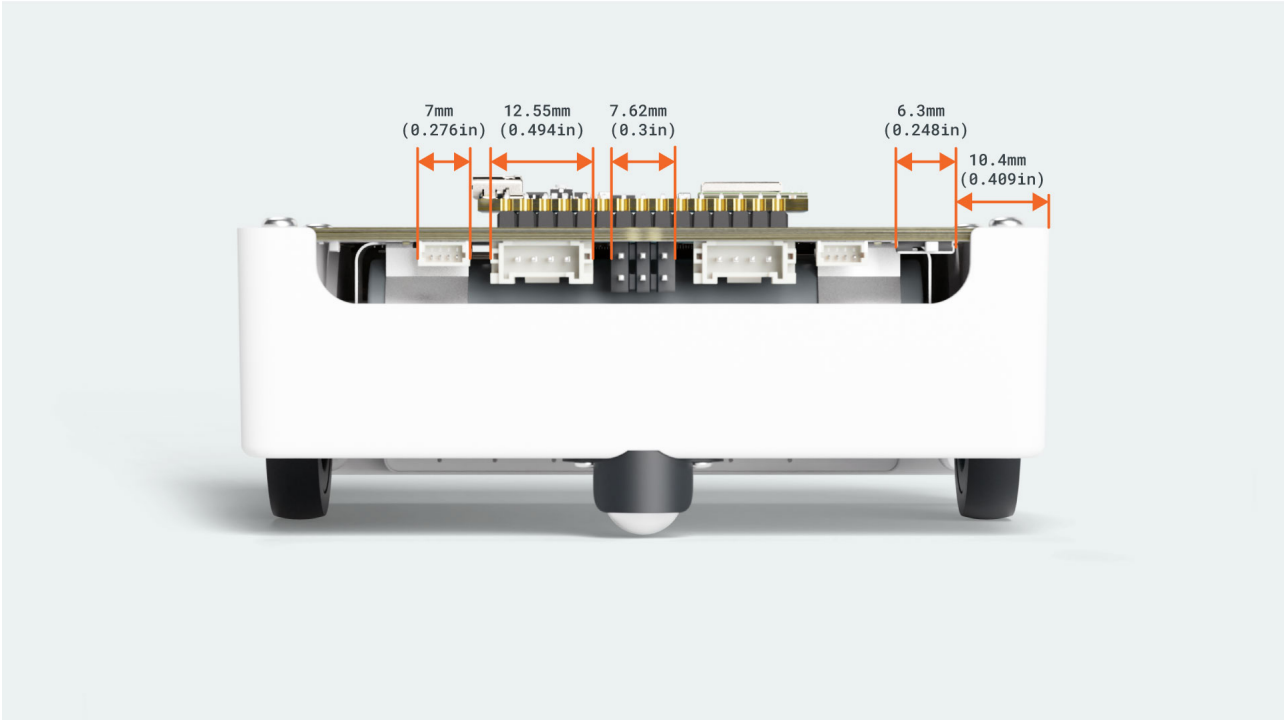
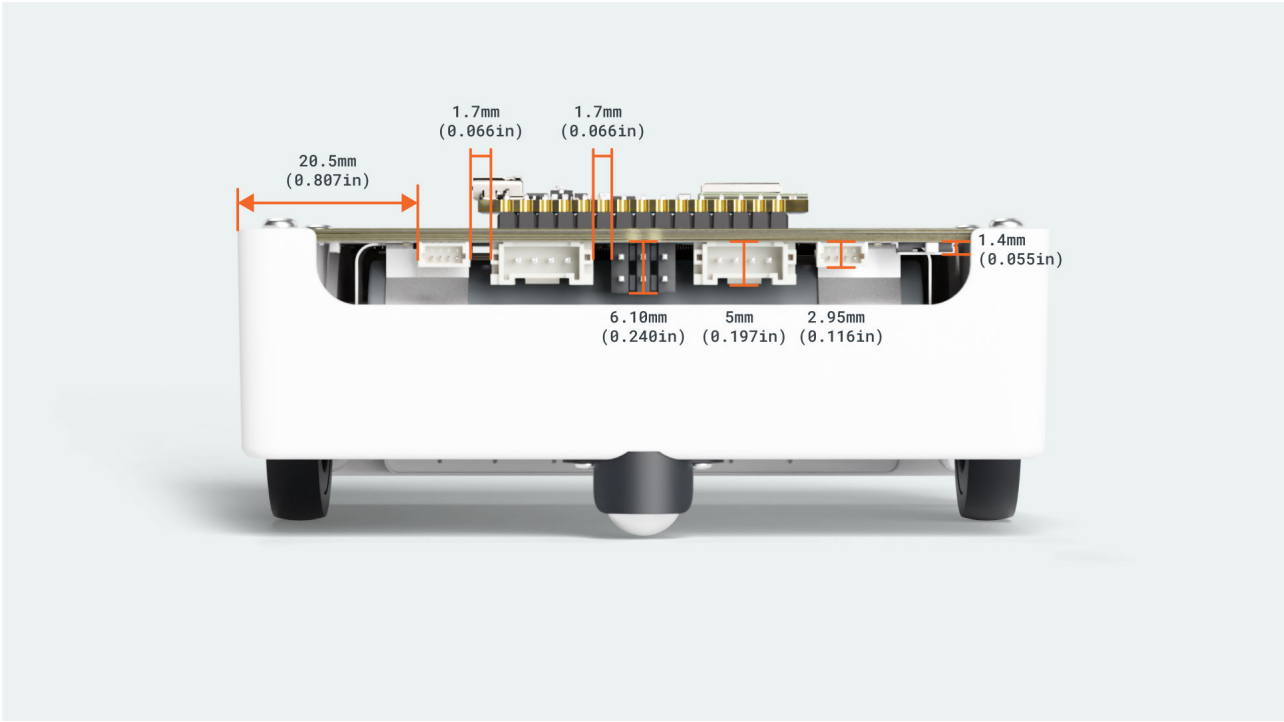
• Vorderseite



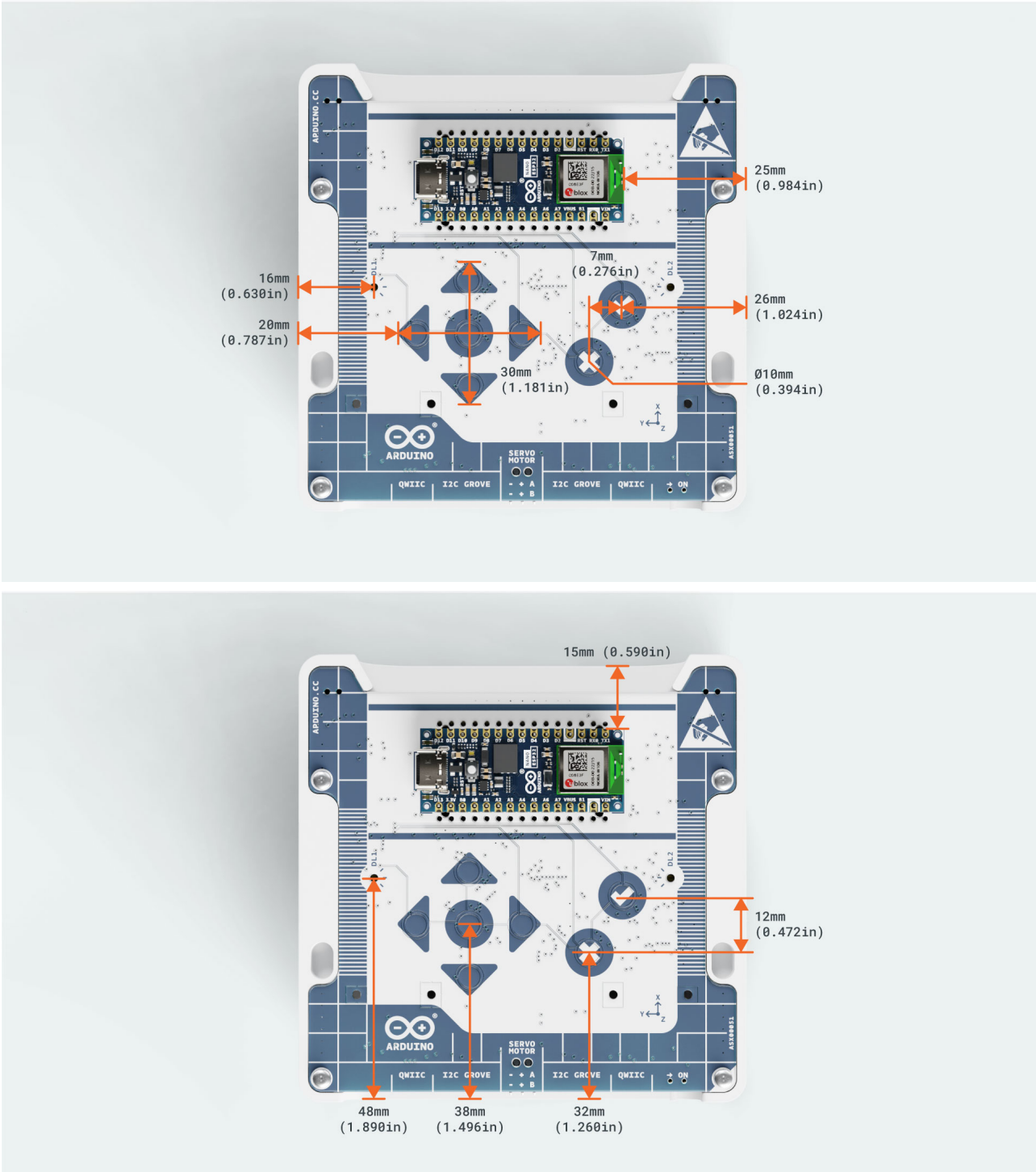
• Seite



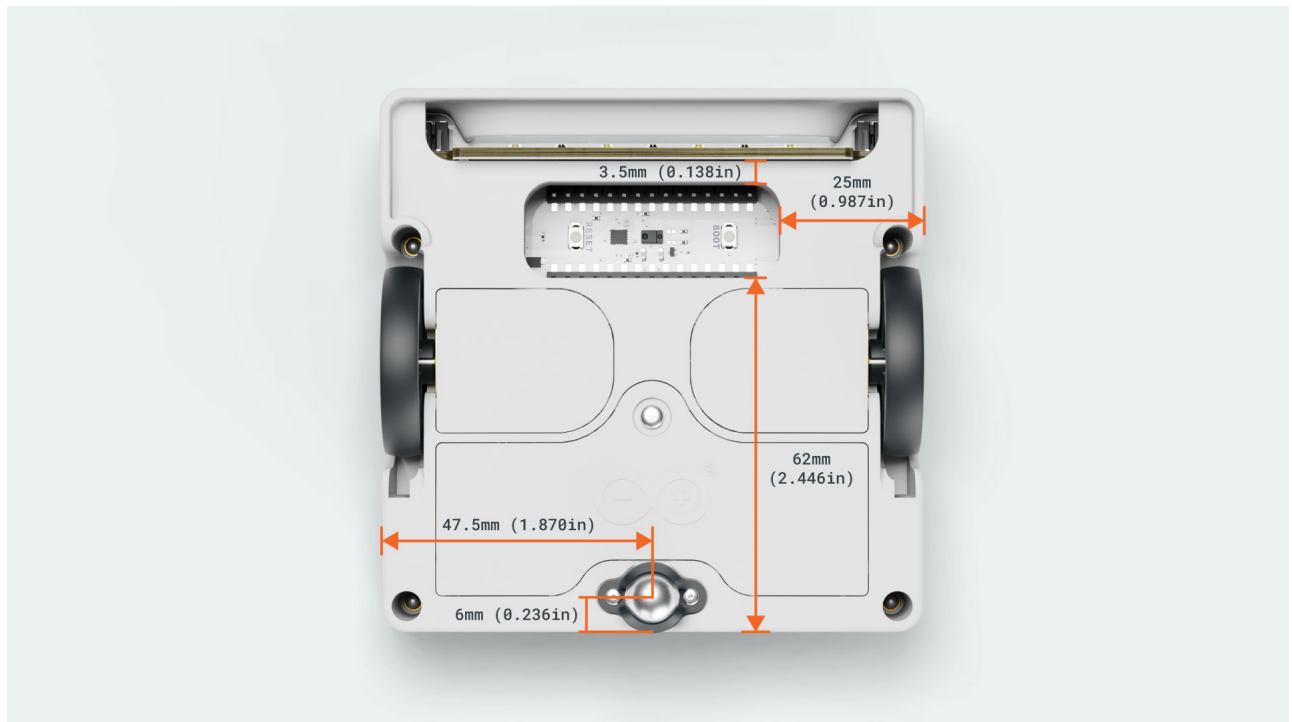
• Rückseite



• Oben



- **Unten**



Wollen Sie mehr?

Wenn Sie das Beste aus Ihrem Alvik-Roboter herausholen möchten, haben wir einen umfassenden Kurs für Sie vorbereitet. Besuchen Sie [Explore Robotics with MicroPython](#) für angeleitete Projekte und Lektionen, die Ihnen helfen, Ihren Alvik zu beherrschen. Dieser Kurs bietet Schritt-für-Schritt-Anleitungen und detaillierte Erklärungen, um Ihre Robotik- und Programmierkenntnisse zu verbessern. Egal, ob Sie Anfänger oder erfahrener Maker sind, unser Kurs wird Ihre Alvik-Erfahrung auf die nächste Stufe bringen.

Entdecken Sie unsere praxisnahen, schrittweisen Lösungen, die Studenten bei ihren ersten Schritten in der Programmierung, Elektronik und Wissenschaft unterstützen. Da sie direkt in den Lehrplan integriert sind, sind keine Vorkenntnisse oder Erfahrungen erforderlich.

Brauche Hilfe

Wenn Sie weitere Hilfe benötigen, können Sie über die folgenden Kanäle Unterstützung erhalten:

- [Arduino Discord](#)
- [Arduino Forum](#)
- [Arduino Kontakt-Formular](#)

Für ausführlichere Unterstützung können Sie sich auch an das Arduino Help Center wenden.