NAO Dokumentation

Dokumentation

Erstellt von: Anna Stabel, Caroline Sarah Schäfer, Sofie Wagner

Inhaltsverzeichnis

1 Einleitung

Hallo! Du willst mit dem NAO-Roboter arbeiten? Kann ich nachvollziehen, er ist ziemlich süß. Doch wie fängt man an? Lass es und zusammen herausfinden!

2 Set-Up

2.1 Choreograph

- Melde dich auf dem Windows PC an
- Öffne Choreograph auf dem Windows-PC
- Erstelle ein neues Projekt, oder öffne ein bestehendes
- Gehe in die Benutzereinstellungen und stelle die Sprache auf deutsch

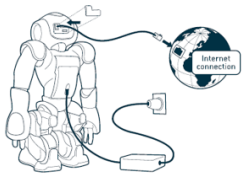
2.2 Verbinde dich mit dem NAO

Um zu gucken, ob dein Programm, welches du geschrieben hast auch wirklich läuft. Solltest du es am besten direkt mit dem NAO testen. Dazu musst du dich mit ihm verbinden. Doch wie geht das?

2.2.1 Verbinde dich über das WIFI

Du kannst NAO entweder über eine kabelgebundene oder eine WiFi-Verbindung mit deinem Computer verbinden:

1. Entferne die Abdeckung hinter dem Kopf des Roboters, um Zugang zur Ethernet-Buchse zu erhalten.
2. Schließe ein Ethernet-Kabel an.



3. Verbinde das Ethernet-Kabel mit deinem Internet-Router. Weitere Details findest du unter: Anschließen meines Roboters an Ethernet.

4. Greife auf die NAO-Webseite zu und melde dich an. Weitere Details findest du unter: Zugriff auf die NAO-Webseite.
5. Wähle und konfiguriere im Bereich "Netzwerkeinstellungenëin WiFi-Netzwerk.

Ergebnis: Die WiFi-Verbindung ist eingerichtet. Du kannst nun das Ethernet-Kabel entfernen und die Abdeckung hinter dem Kopf des Roboters schließen.

2.2.2 Einrichten der kabelgebundenen Verbindung

Achtung: Stelle sicher, dass NAO nicht mit einem Ethernet-Kabel verbunden ist, während er sich bewegt, da das Kabel gezogen werden und der Anschluss beschädigt werden könnte.

1. Entferne die Abdeckung hinter dem Kopf des Roboters, um Zugang zur Ethernet-Buchse zu erhalten.
2. Schließe ein Ethernet-Kabel an.
3. Verbinde das Ethernet-Kabel mit deinem Internet-Router. Weitere Details findest du unter: Anschließen meines Roboters an Ethernet.
4. Das kabelgebundene Netzwerk erscheint in der Liste.
5. Wähle das kabelgebundene Netzwerk aus.

2.2.3 Virtueller Roboter Emulator

Der virtuelle Roboter Emulator in Choregraphe ermöglicht dir, den NAO-Roboter in einer simulierten Umgebung zu steuern, bevor du Aktionen auf einem physischen Roboter ausführst. Dies ist besonders nützlich, um Prototypen zu erstellen und komplexe Bewegungen zu testen. Der Emulator zeigt den NAO in einer 3D-Ansicht und ermöglicht es dir, Aktionen wie Bewegungen, Interaktionen und sensorische Inputs zu testen. Die Integration in Choregraphe bietet eine nahtlose Möglichkeit, Skripte auszuführen und schnell Feedback zu erhalten, bevor du sie auf dem realen Roboter ausführst. Um den virtuellen Roboter Emulator in Choregraphe zu nutzen, folge diesen Schritten:

1. **Verbindung zum virtuellen Roboter herstellen**: Klicke auf das Symbol `Verbindung` und wähle die Option `Virtueller Roboter`. Dadurch wird eine simulierte Version des NAO gestartet.

2. **Erstellen und Testen von Abläufen**: Du kannst durch Drag-and-Drop von Boxen im `Diagramm-Editor` Bewegungsabläufe und Interaktionen erstellen.

3. **Skripte und Aktionen ausführen**: Klicke auf `Play`, um dein Skript im Emulator auszuführen. Der virtuelle NAO wird deine Aktionen simulieren, und du kannst seine Bewegungen und Interaktionen in Echtzeit beobachten.

Durch den Emulator kannst du sicherstellen, dass deine Programme effizient arbeiten und korrekt ablaufen, ohne direkt auf einem echten Roboter zu testen, was Zeit und mögliche Risiken spart.

Weiter Möglichkeiten dich mit NAO zu verbinden findest du hier: .

3 Debugging mit Log-Funktionen

Das Debugging mit Log-Funktionen ist eine wichtige Methode, um Probleme im Code zu identifizieren und das Verhalten des NAO-Roboters während der Ausführung von Programmen zu analysieren. Choregraphe bietet eine integrierte `Log Viewer`-Funktion, die du nutzen kannst, um detaillierte Informationen zur Programmausführung zu erhalten. Durch das Einfügen von `print`-Anweisungen oder speziellen Log-Methoden in den Code kannst du den Status und die Ergebnisse von Operationen während der Laufzeit einsehen.

3.1 Verwendung der Log Viewer-Funktion

Um den Log Viewer effektiv zu nutzen, folge diesen Schritten:

1. **Öffnen des Log Viewers**: Klicke auf `Ansicht` in der Menüleiste und wähle `Log Viewer`, um das Log-Fenster zu öffnen. Hier werden alle Log-Meldungen in Echtzeit während der Programmausführung angezeigt.

2. **Hinzufügen von Log-Nachrichten im Code**: Füge `print`-Anweisungen in deinem Python-Code ein, um spezifische Informationen während der Ausführung auszugeben. Diese Nachrichten erscheinen im Log Viewer und helfen dir, den Ablauf des Programms zu verstehen.

3. **Starten der Programmausführung**: Klicke auf `Play`, um das Programm im `Diagramm-Editor` auszuführen. Der Log Viewer zeigt dann alle Ausgaben an, die im Code durch `print` oder spezielle Debugging-Methoden erzeugt wurden.

4. **Überprüfen der Ausgaben**: Während das Programm läuft, kannst du die Nachrichten im Log Viewer beobachten, um sicherzustellen, dass alle Aktionen wie erwartet ablaufen. Falls Fehler oder unerwartete Werte auftreten, kannst du diese hier sofort sehen.

3.2 Hinzufügen von Log-Nachrichten im Code

Um gezielte Informationen aus deinem Code im Log Viewer anzuzeigen, kannst du verschiedene Techniken nutzen:

- **Print-Anweisungen**: Die einfachste Methode ist die Verwendung von `print("Nachricht")` im Code. Dies ist besonders nützlich, um Variablenwerte und den Ablauf der Programmschritte anzuzeigen.
- **ALLogger API**: Für umfangreiches Logging kannst du die `ALLogger`-API verwenden, die spezifische Meldungen für verschiedene Log-Level (z.B. `INFO`, `WARNING`, `ERROR`) unterstützt. Beispiel:

```
self.logger = ALProxy("ALLogger")
self.logger.info("MyBox", "Dies ist eine Info-Nachricht.")
self.logger.warning("MyBox", "Dies ist eine Warnung.")
self.logger.error("MyBox", "Dies ist eine Fehlermeldung.")
```

- **Fehlersuche bei Eingabe- und Ausgabe-Ereignissen**: Verwende Log-Nachrichten, um den Empfang und die Verarbeitung von Events und Eingaben zu verfolgen. Füge z.B. eine `print`-Anweisung am Anfang einer `onInput_x` Methode ein, um zu bestätigen, dass die Methode aufgerufen wird.

3.3 Interpretation der Log-Nachrichten

Nachdem du das Programm ausgeführt hast und Log-Nachrichten im Log Viewer erscheinen, solltest du Folgendes beachten:

- **Abfolge der Meldungen**: Die Reihenfolge der Nachrichten hilft dir zu verstehen, ob die Programmschritte in der gewünschten Reihenfolge ausgeführt werden.

- **Fehlermeldungen**: Falls Fehler auftreten, analysiere die Fehlermeldungen. Der Log Viewer zeigt oft den genauen Ort im Code und eine Beschreibung des Fehlers an.
- **Variablen und Zustände prüfen**: Durch Ausgaben von Variablenwerten und Zuständen kannst du feststellen, ob bestimmte Werte oder Abläufe korrekt verarbeitet werden.

3.4 Beispiel für das Debugging mit Log-Nachrichten

Hier ist ein Beispiel, wie du Log-Nachrichten in einer benutzerdefinierten Box verwenden kannst:

```
class MyCustomBoxClass(ALModule):
    def onInput_onStart(self):
        print("Start-Input empfangen.")
        self.logger.info("MyBox", "Der Start-Input wurde erfolgreich empf

        # Beispielhafte Berechnung
        result = 10 * 2
        print("Ergebnis der Berechnung:", result)
        self.logger.info("MyBox", "Ergebnis der Berechnung: " + str(resul
```

In diesem Beispiel wird sowohl mit `print` als auch mit der `ALLogger`-API der Ablauf der Programmschritte protokolliert. Dies erleichtert die Fehlersuche und gibt dir einen genauen Überblick über die Ausführung.

4 Events

Events sind ein zentraler Bestandteil in Choregraphe, um Aktionen und Reaktionen des NAO-Roboters zu steuern. Sie ermöglichen es, dynamische und interaktive Abläufe zu erstellen, die auf spezifische Ereignisse reagieren. Events können durch Benutzereingaben, Sensorwerte, Zeitabläufe oder andere Zustände ausgelöst werden und steuern das Verhalten der Boxen in einem Projekt.

4.1 Definition und Bedeutung von Events

Ein Event ist eine Benachrichtigung oder ein Signal, das anzeigt, dass ein bestimmter Zustand oder ein bestimmtes Ereignis eingetreten ist. Sobald ein Event ausgelöst wird, können andere Boxen, die auf dieses Event hören, eine definierte Aktion ausführen. Events ermöglichen die Kommunikation zwischen verschiedenen Boxen und sind essenziell, um komplexe, ereignisgesteuerte Programme zu erstellen.

4.2 Arten von Events

In Choregraphe gibt es verschiedene Arten von Events, die unterschiedliche Zwecke erfüllen:

- **System-Events**: Diese Events werden durch das System oder den NAO-Roboter selbst generiert, z.B. das Erreichen eines Bewegungsziels oder das Erkennen einer Berührung.
- **Benutzerdefinierte Events**: Du kannst benutzerdefinierte Events erstellen, die speziell für die Anforderungen deines Programms entwickelt werden. Diese können durch benutzerdefinierten Code ausgelöst und mit anderen Boxen verknüpft werden.
- **Sensor-Events**: Diese Events werden ausgelöst, wenn ein bestimmter Zustand eines Sensors erkannt wird, z.B. Berührungen an einem bestimmten Punkt des Roboters oder die Erkennung von Gesichtern.
- **Timer-Events**: Timer-Events basieren auf zeitlichen Abläufen und werden nach Ablauf eines definierten Zeitraums oder in festgelegten Intervallen ausgelöst.

4.3 Erstellen und Konfigurieren von Events

Um ein Event in Choregraphe zu erstellen oder zu konfigurieren, kannst du die folgenden Schritte ausführen:

1. **Verbindung zwischen Boxen herstellen**: Um ein Event auszulösen, verbinde den Output einer Box, die das Event auslöst, mit dem Input einer anderen Box, die auf das Event reagieren soll.

2. **Benutzerdefinierte Events definieren**: Falls du ein benutzerdefiniertes Event benötigst, kannst du es in deinem Python-Code innerhalb der Box definieren und mit der `raiseEvent`-Funktion auslösen.

```
self.raiseEvent("MyCustomEvent", "Eventwert")
```

3. **Reaktionslogik festlegen**: Verwende die Eigenschaften der Zielbox, um die gewünschte Reaktion auf das Event zu definieren. Dies könnte das Starten einer Bewegung, das Abspielen eines Sounds oder eine beliebige andere Aktion sein.

4. **Ereignisbedingungen setzen**: Falls erforderlich, kannst du Bedingungen für die Auslösung eines Events festlegen. Dies ermöglicht dir, komplexe Regeln und Interaktionsmuster zu erstellen.

4.4 Verknüpfung und Nutzung von Events

Events können auf verschiedene Weise genutzt und verknüpft werden:

- **Kaskadierende Events**: Ein Event kann mehrere andere Events auslösen, indem es mit mehreren Inputs verschiedener Boxen verbunden ist. Dies ist besonders nützlich für komplexe Interaktionen, bei denen mehrere Aktionen als Reaktion auf ein einzelnes Event stattfinden sollen.

- **Priorisierung von Events**: Falls mehrere Events gleichzeitig ausgelöst werden, kannst du die Priorität und die Reihenfolge der Verarbeitung festlegen, um sicherzustellen, dass kritische Aktionen zuerst ausgeführt werden.

- **Event-Listener**: Eine Box kann als `Listener` für ein bestimmtes Event fungieren und so auf jede Auslösung des Events reagieren. Dies ermöglicht ereignisgesteuerte Logik und erhöht die Flexibilität des Programms.

4.5 Beispiele für die Verwendung von Events

Hier sind einige typische Anwendungsfälle für Events:

- **Reaktion auf Berührung**: Ein `Sensor-Event` wird ausgelöst, wenn der Kopf des NAO berührt wird, und eine verbundene Box startet eine Sprachansage.

- **Zeitschleifen**: Ein `Timer-Event` löst regelmäßig eine bestimmte Bewegung aus, wodurch eine Wiederholungsschleife entsteht.
- **Benutzerinteraktion**: Ein `Benutzerdefiniertes Event` wird ausgelöst, wenn ein spezifisches Sprachkommando erkannt wird, was eine Reaktion wie eine Begrüßung oder eine Animation startet.

Durch die Nutzung und das Verständnis von Events kannst du das Verhalten des NAO-Roboters flexibel und komplex gestalten und auf externe Ereignisse und Benutzerinteraktionen reagieren.

5 Inputs und Outputs in Events und Boxen

In Choregraphe kannst du die Logik des NAO-Roboters durch sogenannte Boxen (Blöcke) darstellen. Jede Box kann verschiedene Inputs (Eingaben) und Outputs (Ausgaben) haben, die du für Events und Aktionen verwenden kannst:

- **Inputs**: Steuern, wie eine Box auf externe Ereignisse reagiert. Sie sind die Eingänge, die du mit anderen Boxen verbinden kannst.
- **Outputs**: Sind die Ausgaben, die von einer Box generiert werden und können mit anderen Boxen verbunden werden, um komplexere Logik und Handlungen auszulösen.

5.1 Eigenschaften von Inputs

Inputs bestimmen, wie eine Box auf externe Ereignisse reagiert. Die wichtigsten Eigenschaften von Inputs umfassen:

- **Name des Inputs**: Der Name dient zur Identifizierung des Inputs und wird im Diagramm-Editor angezeigt. Es ist wichtig, einen aussagekräftigen Namen zu wählen, der den Zweck des Inputs beschreibt.
- **Typ des Inputs**: Der Typ definiert, welche Art von Daten der Input akzeptiert. Dies können einfache Datentypen (wie `int`, `float` oder `string`) oder komplexere Objekte sein. Der Typ sollte den Anforderungen der Logik in der Box entsprechen.

- **Beschreibung**: Ein optionales Textfeld, das die Funktion des Inputs erläutert. Die Beschreibung hilft anderen Nutzern oder dir selbst dabei, den Zweck und die Funktionsweise des Inputs schnell zu verstehen.
- **Eingangsbedingungen**: Du kannst Bedingungen festlegen, unter denen der Input aktiviert oder ausgelöst wird 4. Dies ermöglicht eine präzisere Steuerung der Box-Logik und verhindert ungewollte Aktionen.
- **Verbindungen**: Jeder Input kann mit einem oder mehreren Outputs anderer Boxen verbunden werden. Durch das Verbinden werden die Events und Daten zwischen verschiedenen Boxen weitergeleitet und verarbeitet.

5.2 Eigenschaften von Outputs

Outputs bestimmen, welche Ereignisse oder Daten von einer Box ausgegeben werden. Die wichtigsten Eigenschaften von Outputs umfassen:

- **Name des Outputs**: Der Name dient zur Identifikation des Outputs und sollte klar und präzise benennen, welche Art von Daten oder Ereignissen ausgegeben wird.
- **Typ des Outputs**: Ähnlich wie bei Inputs definiert der Typ des Outputs, welche Art von Daten ausgegeben wird. Dies kann beispielsweise ein einfacher Datentyp wie `boolean` oder ein komplexes Objekt sein.
- **Beschreibung**: Eine optionale Beschreibung des Outputs, die erklärt, welche Daten oder Ereignisse ausgegeben werden. Dies ist nützlich, um die Logik der Box für andere Nutzer verständlich zu machen.
- **Auslösebedingungen**: Du kannst angeben, unter welchen Bedingungen der Output ausgelöst wird 4. Dies kann durch bestimmte Ereignisse oder durch das Erfüllen von Bedingungen innerhalb der Box geschehen.
- **Verbindungen zu anderen Boxen**: Ein Output kann mit einem oder mehreren Inputs anderer Boxen verbunden werden. Dies ermöglicht es, Ereignisse oder Daten weiterzugeben und so komplexere Interaktionen und Abläufe zu gestalten.

5.3 Parameter von Inputs und Outputs

Inputs und Outputs können mit verschiedenen Parametern ausgestattet sein, die deren Verhalten steuern:

- **Standardwert**: Für Inputs kann ein Standardwert festgelegt werden, der verwendet wird, falls keine externe Eingabe erfolgt. Dies stellt sicher, dass die Box einen definierten Zustand hat, auch wenn kein externes Signal vorliegt.
- **Datenformat**: Es kann angegeben werden, in welchem Format Daten akzeptiert oder ausgegeben werden. Zum Beispiel können numerische Werte als `int` oder `float` und Texte als `string` definiert werden.
- **Trigger-Typ**: Für Outputs kannst du festlegen, ob der Output nur einmalig ausgelöst wird oder ob er kontinuierlich Werte sendet, solange bestimmte Bedingungen erfüllt sind 4.
- **Verarbeitungsreihenfolge**: Die Reihenfolge, in der Inputs verarbeitet werden, kann konfiguriert werden. Dies ist nützlich, wenn mehrere Inputs gleichzeitig eintreffen und priorisiert verarbeitet werden müssen.
- **Verbindungstyp**: Inputs und Outputs können auf verschiedene Weise mit anderen Boxen verbunden werden. Du kannst einstellen, ob die Verbindung direkt (Synchronisation der Daten) oder ereignisbasiert erfolgt (nur bei bestimmten Ereignissen).

Die detaillierte Konfiguration von Inputs und Outputs ermöglicht dir eine präzise Steuerung der Logik innerhalb deines Programms und erlaubt es, komplexe Interaktionen zwischen den verschiedenen Boxen in Choregraphe zu gestalten.

5.4 Hinzufügen neuer Inputs und Outputs

Um neue Inputs oder Outputs zu einer Box hinzuzufügen, kannst du folgende Schritte ausführen:

1. **Box auswählen**: Klicke im `Diagramm-Editor` auf die Box, die du konfigurieren möchtest.

2. **Eigenschaften öffnen**: Gehe zu den `Eigenschaften` der Box, indem du sie doppelklickst oder mit der rechten Maustaste darauf klickst und `Eigenschaften anzeigen` auswählst.

3. **Neuen Input/Output hinzufügen**: Nutze die `Input/Output`-Registerkarte und wähle `Neuer Input hinzufügen` oder `Neuer Output hinzufügen`. Du kannst die Typen, Namen und Beschreibungen entsprechend festlegen.

4. **Verbindungen herstellen**: Ziehe Verbindungen von den neuen Inputs oder Outputs zu anderen Boxen, um deine gewünschte Logik zu erstellen.

5.5 Konfigurieren bestehender Inputs und Outputs

Die Konfiguration bestehender Inputs und Outputs ermöglicht es dir, deren Verhalten anzupassen:

1. **Eigenschaften öffnen**: Öffne die Box, indem du sie doppelklickst oder `Eigenschaften` über das Kontextmenü wählst.

2. **Parameter ändern**: Du kannst Namen, Beschreibungen oder Verbindungen der Inputs und Outputs ändern. Gehe zu den entsprechenden `Input/Output`-Einstellungen und passe die Parameter nach Bedarf an.

3. **Bedingungen und Filter**: Falls gewünscht, kannst du spezielle Bedingungen oder Filter auf die Inputs und Outputs anwenden, um deren Verhalten weiter zu spezifizieren.

5.6 Löschen von Inputs und Outputs

Wenn du Inputs oder Outputs nicht mehr benötigst, kannst du sie einfach entfernen:

1. **Box auswählen**: Markiere die Box, aus der du Inputs oder Outputs löschen möchtest.

2. **Input/Output-Einstellungen aufrufen**: Gehe zu den `Input/Output`-Registerkarten in den Box-Eigenschaften.

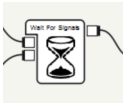
3. **Auswahl löschen**: Wähle den Input oder Output, den du löschen möchtest, und klicke auf `Entfernen` oder nutze die entsprechende Schaltfläche.

4. **Verbindungen prüfen**: Nach dem Löschen solltest du sicherstellen, dass keine ungenutzten Verbindungen übrig bleiben, die Probleme verursachen könnten.

6 Standartboxen

Hier sind die Boxen, die standardmäßig in Choregraphe enthalten sind und die du somit direkt benutzen kannst, und deren Funktionen:

- **Behavior Box**: Diese Box steuert das Verhalten des NAO-Roboters in einer bestimmten Situation. Sie kann komplexe Abfolgen von Bewegungen, Audio- und Sensoreingaben kapseln, um zusammenhängende Aktionen auszuführen. Die Behavior Box kann auch verschachtelte Abläufe und Verzweigungen enthalten.
- **Sound Box**: Mit dieser Box kann der NAO Klänge abspielen oder Texte vorlesen. Du kannst vordefinierte Audiodateien abspielen oder NAO mit der Text-zu-Sprache-Funktion Inhalte sprechen lassen. Die Sound Box kann auch an verschiedene Triggerpunkte gebunden werden, um reaktive Kommunikation zu ermöglichen.
- **Movement Box**: Diese Box steuert die Bewegungen des Körpers, der Arme, des Kopfes und anderer beweglicher Teile des NAO. Du kannst Bewegungsabläufe entweder vorgefertigt abspielen oder über die `Timeline`-Ansicht detailliert anpassen. Hier lassen sich auch Übergänge und Synchronisation mit anderen Aktionen einfügen.
- **Wait Box**: Die Wait Box ermöglicht es dir, den Roboter für eine bestimmte Zeit warten zu lassen, bevor die nächste Aktion ausgeführt wird. Dies kann nützlich sein, um zeitlich abgestimmte Bewegungen oder Interaktionen zu planen. Die Dauer kann in Sekunden oder anderen Einheiten angegeben werden. Es gibt die Wait time (zeit abwarten) box und die Wait-for (auf etwas bestimmtes warten).

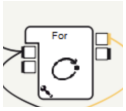


- **Sensor Box**: Diese Box nutzt eingehende Sensordaten (z.B. Berührungen, Infrarot, Kameradaten) und reagiert entsprechend. Du kannst Regeln und Bedingungen erstellen, die auf den Sensoreingaben basieren und eine Aktion auslösen.

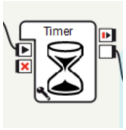
- **LED Box**: Mit dieser Box kannst du die LEDs am NAO-Roboter steuern. Du kannst Muster, Farben und Helligkeit verändern, um Stimmungen, Statusanzeigen oder Signale zu kommunizieren. Die LED Box ist besonders nützlich, um visuelles Feedback zu geben oder auf bestimmte Ereignisse zu reagieren.
- **Speech Recognition Box**: Diese Box ermöglicht es dem NAO, Sprachbefehle zu erkennen und darauf zu reagieren. Du kannst eine Liste von Schlüsselwörtern definieren, auf die der Roboter hören soll, und entsprechende Aktionen mit diesen Ereignissen verknüpfen.
- **Animation Box**: Die Animation Box enthält vorgefertigte Bewegungssequenzen, die aus dem Standardanimationskatalog des NAO geladen werden können. Diese sind besonders nützlich, um realistische Bewegungen oder vordefinierte Verhaltensweisen zu verwenden, ohne sie von Grund auf neu erstellen zu müssen.

- **Face Detection Box**: Diese Box ermöglicht es dem NAO, Gesichter zu erkennen und zu verfolgen. Sie kann mit Aktionen verknüpft werden, die auf das Erkennen oder Verfolgen von Gesichtern reagieren, z.B. das Sprechen mit einer Person oder das Drehen des Kopfes.
- **Video Box**: Die Video Box bietet Zugriff auf die Kameras des NAO-Roboters. Du kannst Videostreams verwenden, um Bilder zu erfassen, Videosequenzen aufzuzeichnen oder visuelle Analysen durchzuführen, wie z.B. die Erkennung von Mustern oder Objekten.
- **Dialog Box**: Diese Box erlaubt die Verwaltung von Dialogen mit dem NAO. Du kannst komplexe Gesprächslogiken erstellen und NAO auf spezifische Wörter oder Phrasen reagieren lassen, wodurch eine interaktive Kommunikation mit Menschen möglich wird.
- **Loop Box**: Mit der Loop Box kannst du Schleifen implementieren, sodass eine Aktion oder ein Verhalten wiederholt ausgeführt wird. Dies ist besonders nützlich für iterative Prozesse oder zeitlich wiederkehrende Aufgaben.

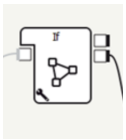
- **For Box**: Mit der For Box kannst du Schleifen implementieren, sodass eine Aktion oder ein Verhalten wiederholt ausgeführt wird. Dies ist besonders nützlich für iterative Prozesse oder zeitlich wiederkehrende Aufgaben.



- **Timer Box**: Diese Box startet einen Timer und löst nach Ablauf eine Aktion aus. Sie ist besonders nützlich, wenn du bestimmte Zeitabläufe in deinem Programmablauf steuern möchtest.



- **If Box**: Die Conditional Box erlaubt es, bedingte Logik zu implementieren. Basierend auf einem bestimmten Zustand oder einer Bedingung kannst du entscheiden, welcher Pfad in deinem Programmablauf ausgeführt wird. Dies ist nützlich, um die Entscheidungen des Roboters dynamisch zu steuern.

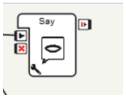


- **ALMemory Access Box**: Diese Box ermöglicht den Zugriff auf den Speicher des NAO-Roboters (`ALMemory`). Du kannst Daten lesen oder schreiben und so wichtige Zustandsinformationen austauschen oder Events überwachen.
- **Switch Case Box**: Diese Box ermöglicht es dir, verschiedene Pfade im Programmablauf abhängig vom eingehenden Wert auszuwählen. Dies ist nützlich, um komplexere Entscheidungsbäume und Programmflüsse zu implementieren.
- **Image Display Box**: Mit dieser Box kannst du Bilder auf dem Bildschirm des NAO anzeigen (falls vorhanden). Dies kann nützlich sein, um visuelle Rückmeldungen zu geben oder mit Benutzern zu interagieren.

- **Random Box**: Diese Box ermöglicht das Erzeugen und Verwenden von Zufallswerten oder das zufällige Auswählen von Verhaltensweisen. Dies verleiht dem Verhalten des Roboters eine gewisse Unvorhersehbarkeit und Dynamik.
- **Posture Box**: Die Posture Box erlaubt es, den Roboter in vordefinierte Haltungen und Positionen zu bringen, z.B. Sitzen, Stehen oder Liegen. Dies ist hilfreich für sicheres Starten oder Beenden von Bewegungen.
- **Play Sound Box**: Die Play Sound Box ermöglicht es dem NAO-Roboter, vordefinierte oder benutzerdefinierte Audiodateien abzuspielen. Du kannst diese Box nutzen, um Geräusche, Musik, Töne oder andere Audiodateien auszugeben, die auf dem Roboter gespeichert sind. Die Wiedergabeoptionen können angepasst werden, um die Lautstärke, Wiederholungen oder das Verhalten nach dem Ende des Sounds zu steuern. Diese Box ist besonders nützlich, um akustisches Feedback zu geben, eine Interaktion zu unterstreichen oder Benutzer auf bestimmte Aktionen aufmerksam zu machen.

- **Get File Path Box**: Die Get File Path Box ermöglicht es, den Pfad zu einer Datei im Dateisystem des NAO-Roboters oder auf einem verbundenen Computer abzurufen. Diese Box wird häufig verwendet, um Pfade zu Dateien wie Bildern, Audiodateien oder Skripten zu ermitteln, die später in anderen Boxen oder Prozessen verwendet werden können. Durch das Abrufen des Dateipfads kannst du flexibel mit Dateien arbeiten, die dynamisch ausgewählt oder in anderen Kontexten benötigt werden.

- **Say Box**: Die Say Box ermöglicht es dem NAO-Roboter, Text laut auszusprechen, indem er die integrierte Text-zu-Sprache-Funktion (Text-to-Speech, TTS) verwendet. Du kannst diese Box nutzen, um dynamische Ansagen, vordefinierte Texte oder interaktive Antworten zu generieren. Die `Say Box` bietet eine einfache Möglichkeit, den NAO mit Menschen interagieren zu lassen und Sprachfeedback in Echtzeit zu geben.



6.1 Zusammenfassung mehrerer Boxen in eine einzige Box

Das Gruppieren mehrerer Boxen in eine einzige `Behavior Box` oder eine benutzerdefinierte Box bietet verschiedene Vorteile:

- **Modularität und Wiederverwendbarkeit**: Durch das Zusammenfassen von zusammengehörigen Boxen kannst du logische Einheiten erstellen, die einfach in anderen Projekten oder an verschiedenen Stellen des Programms wiederverwendet werden können.
- **Übersichtlichkeit**: Große und komplexe Programme können durch das Gruppieren von Boxen besser organisiert und leichter zu verstehen gemacht werden. Die äußere Box kapselt die interne Logik und reduziert die visuelle Komplexität des Hauptdiagramms.
- **Erstellen einer Gruppenbox**: Wähle die Boxen, die du zusammenfassen möchtest, und klicke mit der rechten Maustaste. Wähle die Option `Gruppieren` oder `Zusammenfassen`, um eine neue Box zu erstellen, die alle enthaltenen Boxen umfasst.

6.2 Navigation und Bearbeitung innerhalb von Boxen

In Choregraphe kannst du dich durch Doppelklicken auf eine Box in ihren Inhalt „hineinklicken“ und die darin enthaltene Logik und Struktur bearbeiten:

- **Detaillierte Bearbeitung**: Nach dem Hineinklicken in eine Box kannst du die enthaltenen Boxen, Verbindungen und Abläufe sehen und bearbeiten. Dies ermöglicht es dir, die innere Funktionsweise präzise anzupassen, ohne die äußere Struktur des Programms zu ändern.
- **Verschachtelung**: Boxen können verschachtelt sein, was bedeutet, dass du in einer Box weitere untergeordnete Boxen haben kannst. Dies unterstützt die Schaffung komplexer, hierarchischer Strukturen mit klar definierten Abhängigkeiten und Verantwortlichkeiten.

- **Einfache Rückkehr**: Nach der Bearbeitung kannst du durch Klicken auf den `Zurück`-Button oder das Hauptdiagramm leicht zur äußeren Ebene deines Programms zurückkehren.

7 Timeline-Funktion

Die `Timeline`-Funktion in Choregraphe ermöglicht es dir, Bewegungsabläufe, Audio, Animationen und andere Aktionen für den NAO-Roboter präzise zu planen und zu synchronisieren. Die Timeline bietet eine visuelle Oberfläche, mit der du die Dauer, Abfolge und Übergänge verschiedener Aktionen anpassen kannst, um nahtlose Bewegungs- und Verhaltensabläufe zu gestalten. Sie ist ein leistungsstarkes Werkzeug, um komplexe Szenarien zu realisieren, die exakte Timing-Kontrolle erfordern.

7.1 Hauptmerkmale der Timeline

Die `Timeline`-Funktion bietet eine Vielzahl an Werkzeugen und Optionen:

- **Keyframes**: Du kannst *Keyframes* definieren, die bestimmte Positionen oder Zustände des NAO-Roboters darstellen. Der Roboter bewegt sich zwischen diesen Keyframes, wodurch fließende Übergänge entstehen.
- **Spuren und Ebenen**: Die Timeline bietet mehrere Spuren, um unterschiedliche Aspekte des Roboters, wie Bewegungen, Audio oder Lichteffekte, gleichzeitig zu steuern. Jede Spur kann unabhängig bearbeitet und synchronisiert werden.
- **Zeitleistensteuerung**: Über die Steuerelemente der Timeline kannst du Animationen abspielen, anhalten, vorspulen oder in Zeitlupe anzeigen, um die Übergänge und Abläufe präzise zu überprüfen.
- **Bewegungskontrolle**: Du kannst Bewegungen durch Ziehen und Anpassen von Keyframes innerhalb der Timeline genau steuern. Die Bewegungen lassen sich verlangsamen, beschleunigen oder mit anderen Bewegungsabläufen synchronisieren.

7.2 Erstellen und Bearbeiten von Bewegungsabläufen

Um die `Timeline`-Funktion effektiv zu nutzen, folge diesen Schritten:

1. **Erstellen eines neuen Bewegungsablaufs**: Klicke auf die `Timeline`-Ansicht in Choregraphe, um eine neue Timeline für deine Aktion zu öffnen.

2. **Hinzufügen von Keyframes**: Setze Keyframes, um den Start- und Endpunkt einer Bewegung oder einer Aktion festzulegen. Du kannst einzelne Körperteile des NAO-Roboters steuern oder mehrere Keyframes für komplexe Bewegungen verwenden.

3. **Synchronisierung von Aktionen**: Lege fest, wie andere Aktionen, wie Audio- oder LED-Effekte, mit den Bewegungen synchronisiert werden. Ziehe dazu Elemente auf der Timeline und ordne sie entsprechend den gewünschten Zeitpunkten an.

4. **Feinabstimmung und Tests**: Spiele die Timeline ab und überprüfe die Übergänge und Bewegungen. Du kannst Keyframes verschieben, hinzufügen oder löschen, um die Bewegung weiter anzupassen und zu verfeinern.

7.3 Vorteile der Timeline-Funktion

Die Verwendung der `Timeline`-Funktion bietet mehrere Vorteile:

- **Exakte Steuerung**: Die präzise Platzierung von Keyframes und Aktionen ermöglicht es dir, detaillierte Bewegungsabläufe mit exakter Kontrolle zu erstellen.

- **Visuelle Übersicht**: Die visuelle Darstellung der Bewegungsabläufe erleichtert es, komplexe Sequenzen zu verstehen und zu ändern.

- **Nahtlose Synchronisation**: Du kannst verschiedene Elemente wie Audio, Bewegungen und Licht zusammenführen und perfekt synchronisieren.

- **Flexibilität**: Die Timeline erlaubt es, bestehende Bewegungen einfach zu verändern und zu optimieren, ohne von Grund auf neu beginnen zu müssen.

Die `Timeline`-Funktion ist daher ein zentrales Werkzeug in Choregraphe, um den NAO-Roboter kreativ und präzise zu steuern.

8 Custom Code

Neben den Standard-Boxen in Choregraphe kannst du eigenen benutzerdefinierten Code einfügen, um komplexe und maßgeschneiderte Funktionen für den NAO-Roboter zu erstellen. Das Hinzufügen von Custom Code ermöglicht es dir, die Logik und das Verhalten des Roboters flexibler und detaillierter zu steuern.

8.1 Hinzufügen von Custom Code

Um benutzerdefinierten Code zu einer Box in Choregraphe hinzuzufügen, folge diesen Schritten:

1. **Erstellen einer neuen Box**: Klicke mit der rechten Maustaste in den `Diagramm-Editor` und wähle `Neue Box erstellen`. Alternativ kannst du auch eine bestehende Box auswählen und deren Eigenschaften anpassen.
2. **Öffnen des Skript-Editors**: Doppelklicke auf die Box, um den `Skript-Editor` zu öffnen. Hier kannst du Python-Code direkt eingeben.
3. **Schreiben von benutzerdefiniertem Code**: Schreibe den gewünschten Python-Code innerhalb des `onLoad`, `onInput_x` oder `onUnload` Abschnitts. Diese Abschnitte ermöglichen es dir, verschiedene Aktionen bei der Initialisierung, während der Eingaben und beim Beenden der Box zu definieren.
4. **Speichern und Testen**: Speichere deinen Code und teste die Box durch Verknüpfungen mit anderen Boxen oder durch die Ausführung des Programms. Verwende den `Log Viewer`, um Fehler oder Ausgaben während der Ausführung zu debuggen.

8.2 Grundstruktur der Custom Files

Die Grundstruktur einer benutzerdefinierten Box mit Custom Code in Choregraphe basiert auf dem internen Python-Skript. Hier ist ein Beispiel für eine typische Struktur:

```
class MyCustomBoxClass(ALModule):
    def __init__(self, name):
        ALModule.__init__(self, name)
        self.name = name
        # Initialisierung von Variablen oder Zuständen

    def onLoad(self):
        # Wird ausgeführt, wenn die Box geladen wird
        print("Box geladen und bereit.")
```

```
    def onInput_onStart(self):
        # Wird ausgeführt, wenn die Box einen Input empfängt
        print("Benutzerdefinierte Aktion wird gestartet.")
        # Hier kann dein Code ausgeführt werden

    def onUnload(self):
        # Wird ausgeführt, wenn die Box entladen oder deaktiviert wird
        print("Box wird entladen.")
```

- **Klassendeklaration**: Jede benutzerdefinierte Box basiert auf einer Python-Klasse, die von der `ALModule`-Klasse erbt. Dies ermöglicht die Interaktion mit NAOqi (dem Steuerungssystem des NAO).
- **`__init__` Methode**: Diese Methode initialisiert die Box und kann verwendet werden, um Variablen oder Zustände zu definieren.
- **`onLoad` Methode**: Führt Code aus, wenn die Box geladen wird, z.B. zur Initialisierung von Einstellungen.
- **`onInput_x` Methoden**: Diese Methoden reagieren auf Eingaben und steuern die Aktionen, die ausgeführt werden, wenn ein Input-Event empfangen wird. Die Benennung kann variieren (z.B. `onInput_onStart`), je nach der Anzahl und Art der Eingänge.
- **`onUnload` Methode**: Diese Methode wird aufgerufen, wenn die Box entladen oder deaktiviert wird, z.B. zur Bereinigung von Ressourcen.

8.3 Beispiel für Benutzerdefinierten Code

Hier ist ein einfaches Beispiel für benutzerdefinierten Code in einer Custom Box:

```
def onInput_onStart(self):
    # Einfacher Beispielcode
    self.tts = ALProxy("ALTextToSpeech")
    self.tts.say("Hallo, ich bin NAO!")
    # Weitere Aktionen können hier hinzugefügt werden
```

Dieser Code lässt den NAO eine Begrüßung sprechen, sobald die Box einen Start-Input erhält.

9 Weiter Links

Wenn du mehr Informationen benötigst, informiere dich hier:

- Offizielle NAO-Docs
- Aldebaran
- NOX