Schweizerische
Fachschule
TEKO

Dominosteine legen mit dem «Niryo One»

Diplomand: Jeshua Roth

Schule: TEKO Luzern

Klasse: L-TMA-18-Do-b

Abgabedatum: 11. Oktober 2021

Studiengang: Dipl. Techniker HF Maschinenbau

Eingereicht an:

TEKO Schweizerische Fachschule

Pilatusstrasse 38, 6003 Luzern

Vorgelegt von:

Jeshua Roth

Dipl. Techniker HF Maschinenbau

L-TMA-18-Do-b

Chrüzmatte 5, 6247 Schötz

Telefon.: 078 966 35 75

Diplombegleiter TEKO:

Rudolf Gautschi

Experte TEKO:

Markus Fürholz

Thema:

Dominosteine legen mit dem «Niryo One»

Abgabetermin:

Montag, 11. Oktober 2021

Inhaltsverzeichnis

1. Management Summary

1.1 Ausgangslage

Die TEKO Luzern besitzt seit einiger Zeit den Roboter «Niryo One» inklusive Zusatzmaterial, der wie ich hörte allerdings noch nicht viel zum Einsatz gekommen war. Als ich von Andreas Holzer, meinem Robotik- und SPS Dozenten, erfahren habe, dass die TEKO Luzern einen programmierbaren Roboterarm besitzt, war ich hell begeistert. Die Richtung Robotik hat mich schon vor dem Beginn des Studiums interessiert und nun konnte ich mit diesem Roboterarm eigene Erfahrungen machen.

Andreas Holzer wollte schon länger einen guten Weg finden, den schon vorhandenen «Niryo One» Roboter, im Unterricht einzubauen. Daher hat er vorgeschlagen das Projekt mitzuverfolgen und mich bei grösseren Herausforderungen, bei denen ich nicht weiterkommen würde, zu unterstützen. Jede Unterstützung werde ich aber auch schriftlich festhalten, damit ersichtlich ist, wobei ich die Unterstützung in Anspruch genommen habe.

Durch die Gespräche mit Andreas Holzer über ein mögliches Projekt mit dem Roboter «Niryo One» stellte ich schnell fest, dass die Umsetzung eine sehr grosse Herausforderung darstellen würde, da ich keine bis kaum Vorkenntnisse besass. Zur Beruhigung erklärte mir aber Andreas Holzer, dass wir im Unterricht vieles noch lernen werden, dass meine Umsetzung möglich macht und man das meiste bei der Anwendung direkt am Roboter lerne.

Für die erwähnte Diplomarbeit im Studiengang Dipl. Techniker HF Maschinenbau sollen ca. 150 - 250 Arbeitsstunden aufgewendet werden.

1.2 Themenwahl

Da im Maschinenbau viel mit unterstützten Robotern gearbeitet wird oder sogar ganze Produktionsanlagen voll automatisiert sind, sehe ich das Projekt als passende Diplomarbeit und Repräsentation meines erworbenen Wissens an der TEKO Luzern. Dies soll auch ein Beitrag zu Industrialisierung 4.0 sein und zeigen, was mit den heutigen Mitteln möglich ist, die mir in diesem Projekt zur Verfügung standen.

1.3 Zielsetzung

Das Ziel meiner Arbeit war, min. 20 Dominosteine, die auf einem Förderband zum «Niryo One» befördert werden, visuell farblich und von der Position her zu erfassen und anschliessend auf vorprogrammiertem Weg zu legen. Anschliessend soll der «Niryo One» auch die Dominosteine anstossen, um die Kettenreaktion auszulösen. Die Anordnung der Steine auf dem Förderband soll willkürlich platziert sein, damit der «Niryo One» anhand einer eingebauten Kamera die Position und die Farbe erkennt.

1.4 Lösungsvorschlag

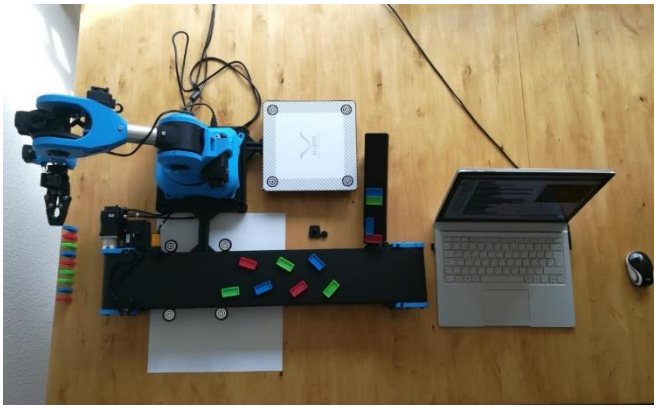


Abbildung 1 Definitiver Lösungsvorschlag

Für den definitiven Lösungsvorschlag habe ich den «Niryo One» Roboter verwendet, der auf einer Aluminiumführung steht. Diese Aluminiumführung ist mit dem «Vision Board» (rechts vom Roboterarm) und dem Förderband durch Nutensteinen verbunden. Das Förderband ist ausgestattet mit den vier «Vision Wahrzeichen» (den schwarzen Kreisen auf weissem Hintergrund), einem Infrarotsensor, einem Endanschlag und einer Ladestation. Auf dem «Niryo One» ist auch noch eine Kamera befestigt. Als Dominosteine wurden farbige Kunststoffsteine ausgewählt und als Greifer habe ich mich für den «Large Greifer» entschieden. Der Programmcode zur Steuerung des Roboters wurde anhand des Programms «Niryo One Studio» geschrieben mit der Programmiersprache Blockly. Bei der Programmierung wurde der Fokus auf die Qualität und die Programmierung mit Unterprogrammen festgelegt. Als zusätzliche Sicherheitsmechanismen, die zum Gelingen des Ablaufes beitragen sollen, habe ich mich auf einen Endanschlag, einem Infrarotsensor und einer geschwindigkeitsregulierten Programmierung entscheiden. Folgender Ablauf beschreibt das Programm zusammenfassend:

1. •Die Domonosteine werden von der Ladestation, durch den «Niryo One», willkürlich auf das Förderband gelegt.
2. •Das Förderband bewegt die Dominosteine in den «Vision Workspace», der markiert ist mit den vier schwarzen Kreisen auf weissem Hintergrund.
3. •Die Kamera auf dem «Niryo One» erkennt die Steine und deren Farbe innerhalb des «Vision Workspace».
4. •Der Infrarotsensor der am Förderband hinter dem «Vision Workspace» befestigt ist, erkennt ebenfalls die Steine, die seinen Erkennungsbereicht passieren.
5. •Das Programm, das den Roboter und das Förderband steuert, gibt den Befehl, dass das Förderband gestoppt wird.
6. •Der «Niryo One» Roboter macht dann einen «Vision Pick», wobei der Greifer den bereits erfassten Stein greift der auf dem Förderband liegt.
7. •Der Stein wird dann vom Roboterarm auf das «Vision Board» und anschliessend auf die Tischplatte gestellt. Dieser Vorgang wird mit weiteren Steinen wiederholt, bis eine Dominosteinkette von 20 Steinen steht.
8. • Das Programm wird dann den «Niryo One» Roboter dazu bringen die Dominosteinkettenreaktion auzulösen. Dies erfolgt über das Anstossen des letzen Steines.

2. Beruflicher Lebenslauf

Abbildung 2 Bewerbungsfoto

Jeshua Roth

Chrüzmatte 5

6247 Schötz

roth.jeshua@gmail.com

08.2009 – 08.2013	**AUSBILDUNG ALS POLYMECHANIKER EFZ, TEGERFELDEN**
08.2013 – 12.2013	**SCHWEIZER ARMEE ALS SANITÄTER, AIROLO**
12.2013 – 01.2016	**EHRENAMTLICHER KIRCHLICHER DIENST, BERLIN**
09.2016 – 09.2017	**FACHABITUR GWSBS, BAD SÄCKINGEN**
09.2017 – 02.2018	**BACHELOR MASCHINENTECHNIK BEI ZHAW, WINTERTHUR**
09.2018 – JETZT	**ARBEITEN BEI CELSIO KÄLTE UND KLIMA AG, DÄLLIKON**
09.2018 – JETZT	**STUDIUM DIPL. TECHNIKER HF MASCHINENBAU, TEKO LUZERN**

3. Aufgabenstellung

Das Legen der Dominosteine von Hand kann eine Herausforderung darstellen, wenn man keine ruhigen Hände dafür hat. Deswegen ist ein Programm zu erstellen, das den 6-achsigen Knickroboterarm «Niryo One» steuert. Dabei sollen das Förderband und die Kamera einbezogen werden, damit eine Dominosteinkette aufgebaut und danach die Kettenreaktion auslöst werden kann, ohne dass dafür eine menschliche Hand zum Einsatz kommen muss. Die erwartete programmierte Dominosteinkette sollte einen Umfang von ungefähr 20 Dominosteinen haben.

3.1 Meilensteine

- Start Diplomarbeit: 16.08.2021
- Besprechungen 1: 02.08.2021
- Besprechungen 2: 23.09.2021
- Abgabe Diplomarbeit: 11.10.2021
- Präsentation: 28.10.2021

3.2 Abstraktion

1. Es ist eine Dominosteinreihe mithilfe des «Niryo One» Roboters und dem Zubehör im Umfang von mindestens 20 Dominosteinen zu legen.
2. Es ist eine Dominosteinreihe mit dem Roboter im Umfang von mindestens 20 Steinen zu legen.
3. Es ist eine Dominosteinreihe mithilfe des Roboters zu legen.

3.3 Auswertung der Abstraktion

1. Bei dieser Formulierung sind zu viele Details schon festgelegt, was die Auswahlfreiheit zu stark einschränken kann.
2. Bei dieser Formulierung wird genug Freiraum gelassen aber trotzdem die Wichtigkeiten aufgezählt.
3. Bei dieser Abstraktion wird zu viel Freiraum gegeben, dass das Projekt nur unnötig verlängern würde.

3.4 Entscheid der Abstraktion

Die zweite Formulierung bietet eine gute Grundlage, um das Projekt umzusetzen und die optimale Lösungsvariante zu ermitteln.

4. Pflichtenheft

4.1 Einleitung

Dieses Pflichtenheft beschreibt die Umsetzung des Projektes: «Dominosteine legen mit dem «Niryo One»» und es dient der schriftlichen Grundlage für die Vereinbarungen und Rahmenbedingungen, die für dieses Projekt festgelegt wurden.

4.2 Zweck und Ziel

Das Ziel meiner Arbeit ist, min. 20 Dominosteine, die auf einem Förderband zum «Niryo One» befördert werden sollen, visuell farblich und von der Position her zu erfassen und auf vorprogrammiertem Weg zu legen. Anschliessend soll der «Niryo One» auch die Dominosteine anstossen, um die Kettenreaktion auszulösen. Die Anordnung der Steine auf dem Förderband soll willkürlich platzierbar sein, damit der «Niryo One» anhand einer eingebauten Kamera die Position und die Farbe erkennt.

4.3 Historie der Dokumentenversion

Revision	Autor	Datum	Kommentar
1	Jeshua Roth	02.08.2021	Pflichtenheftvorstellung
2	Jeshua Roth	14.08.2021	Überarbeitung gemäss weiteren Erkenntnissen
3	Jeshua Roth	22.08.21	Fertigstellung des Pflichtenhefts
4	Jeshua Roth	07.09.21	Überarbeitung des Pflichtenhefts

4.4 Verteiler

Rolle	Name	Telefon	E-Mail
Diplombegleiter TEKO	Rudolf Gautschi	+41 79 817 68 04	rudolf.gautschi@gmail.com
Experte TEKO	Markus Fürholz	+41 32 623 24 40	markus.fuerholz@gawnet.ch
Projektleiter/ Student/ Programmierer	Jeshua Roth	+41 78 966 35 75	roth.jeshua@gmail.com

4.5 Ausgangssituation

Die TEKO Luzern besitzt seit einiger Zeit den Roboter «Niryo One» inklusive Zusatzmaterial, der wie ich hörte allerdings noch nicht viel zum Einsatz gekommen war. Als ich von Andreas Holzer, meinem Robotik- und SPS Dozenten, erfahren habe, dass die TEKO Luzern einen programmierbaren Roboterarm besitzt, war ich hell begeistert. Die Richtung Robotik hat mich schon vor dem Beginn des Studiums interessiert und nun konnte ich mit diesem Roboterarm eigene Erfahrungen machen.

Andreas Holzer wollte schon länger einen guten Weg finden, den schon vorhandenen «Niryo One» Roboter, im Unterricht einzubauen. Daher hat er vorgeschlagen das Projekt mitzuverfolgen und mich bei grösseren Herausforderungen, bei denen ich nicht weiterkommen würde, zu unterstützen. Jede Unterstützung werde ich aber auch schriftlich festhalten, damit ersichtlich ist, wobei ich die Unterstützung in Anspruch genommen habe.

Durch die Gespräche mit Andreas Holzer über ein mögliches Projekt mit dem Roboter «Niryo One» stellte ich schnell fest, dass die Umsetzung eine sehr grosse Herausforderung darstellen würde, da ich keine bis kaum Vorkenntnisse besass. Zur Beruhigung erklärte mir aber Andreas Holzer, dass wir im Unterricht vieles noch lernen werden, dass meine Umsetzung möglich macht und man das meiste bei der Anwendung direkt am Roboter lerne.

Da im Maschinenbau viel mit unterstützten Robotern gearbeitet wird oder sogar ganze Produktionsanlagen voll automatisiert sind, sehe ich das Projekt als passende Diplomarbeit und Repräsentation meines erworbenen Wissens an der TEKO Luzern. Dies soll auch ein Beitrag zu Industrialisierung 4.0 sein und zeigen, was mit den heutigen Mitteln möglich ist, die mir in diesem Projekt zur Verfügung standen.

Für die erwähnte Diplomarbeit im Studiengang Dipl. Techniker HF Maschinenbau sollen ca. 150 - 250 Arbeitsstunden aufgewendet werden.

Abbildung 3 Möglicher Roboteraufbau

4.6 Abgrenzungen und Rahmenbedingungen

- Ungenauigkeiten beim Legen der Dominosteine durch den Roboterarm dürfen den Erfolg des Projekts nicht beeinträchtigen.
- Allfällige Ergänzungen, die noch im Laufe des Projekts erforderlich und für dessen Erfolg notwendig sind, die allerdings zu Aufwendig sind könnten, werden mit Hilfe vom Dozenten Andreas Holzer gemacht. Wenn dies eintritt, wird auf dies erkennbar hingewiesen.
- Ungenauigkeiten des «Vision Sets» aufgrund der Kamera, darf den Erfolg des Projekts nicht beeinträchtigen.
- Der Hauptteil der Arbeit ist die Programmierung des «Niryo One».
- Der Zusammenbau des Roboters wird nicht Teil der Arbeit sein.
- Berechnungen für den Roboter und Auswertung der Wirtschaftlichkeit sind nicht Teil der Arbeit.
- Weiterführende Optimierungen des Programms oder der Prozesse sind möglich.
- Weitere Funktionen und Bauteile zur Verbesserung der Automation des Ablaufs sind möglich.

4.7 Anforderungen an das Programm/ den Ablauf

1. Die Dominosteine sollen von einem Lagerort, durch den Roboterarm, auf das Förderband gelegt werden.
2. Die Dominosteine auf dem Förderband sollen anhand der Kamera im «Vision Workspace» (Feld, welches die Ecken mit den schwarzen Kreisen gekennzeichnet sind) erkannt und notfalls mit dem Infrarotsensor erkannt und zurück in das «Vision Workspace» befördert werden.
3. Angekommen im «Vision Workspace», soll die Kamera die Anordnung und Farbe erkennen und die Dominosteine aufgreifen.
4. Die gescannten Dominosteine sollen anschliessend auf dem vorprogrammierten Weg gelegt werden.
5. Die Dominosteine sollen einen Abstand von Stein zu Stein von ca. 2 cm haben.
6. Der «Niryo One» Roboter soll abschliessend die gelegten Dominosteine anstossen und so die Kettenreaktion auslösen.

4.8 Schnittstellen

- Weiterführend könnten mehrere Roboterarme und Förderbänder angeschlossen werden, um grössere Strecken mit Dominosteinen zu legen oder weitere Funktionen einzubauen
- Ebenfalls könnten die ganzen Programme in Python oder C++ programmiert werden, falls man weitere externe Elemente noch einbauen möchte (Anschlussschnittstellen sind am «Niryo One» gegeben).

4.9 Voraussetzungen und Betriebsbedingungen

- Es sollten immer die gleichen Dominosteine mit gleichem Mass verwendet werden. Die Farbe darf variieren.
- Auch sollten die Steine nicht Schwarz sein, da der Kontrast auf dem Förderband dadurch zu gering sein könnte, um die Steine zu erkennen.
- Der Aufbau/ die Anordnung des Roboters, des «Vision Sets» und des Förderbands darf nicht verändert werden, da sonst die Programmierung fehlerhaft wird. Bestenfalls sollen alle Komponenten verbunden sein.

4.10 Technische/ Persönliche Ziele

- Zielgerechtes Legen der Steine gemäss Vorgabe und Ablauf
- Keine Fehler in der Programmstruktur, die den Ablauf unterbrechen
- Alle Anforderungen gemäss den geforderten Kriterien umsetzen
- Dominosteinkettenreaktion funktioniert einwandfrei
- Eine effiziente Wahl bezüglich des Programms treffen
- Dem Terminplan gefolgt
- Nachvollziehbare Auswahlen treffen
- Selbständiges Ausarbeiten des Programms
- Korrekte Umsetzung des Projektmanagements gemäss Erlerntem an der TEKO Luzern
- Den Roboter so gut kennen, um jegliche Probleme lösen zu können
- Erfahrung in der Robotik sammeln

5. Terminplan/ Ablaufplanung

Projektbezeichnung: Dominosteine legen mit dem Niryo One	**Terminplanung**										
Themeneingabe: Montag, 7. Juni 2021											
Starttermin: Montag 16. August 2021											
Abgabetermin: Montag, 11. Oktober 2021											
Präsentation: Donnerstag 28.Oktober 2021											
Erstellungsdatum: 02.07.2021											
Version: 6											
Kennfarben	**Soll**		**IST**								
Jahr	**2021**										
Kalenderwoche	**bis 33**	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**	**41**	**nach 41**
Tage	**bis 16.08**	**16.08-22.08**	**23.08-29.08**	**30.08-05.09**	**06.09-12.09**	**13.09-17.09**	**20.09-26.09**	**27.09-03.10**	**04.10-10.10**	**11.10-17.10**	**nach 17.10**
Ausgangslage											
Kick-off											
Themenbeschreibung											
Rahmenbedingungen/ Themeneingabe											
Projektinitialisierung											
Pflichtenheft & Aufgabenabgrenzung											
Infosammlung											
Anforderungsliste											
Terminplanung & Recourcenplanung											
1. Besprechung TEKO (02.08.21)											
Planen und Klären											
Aufgabenstellung											
Kernprobleme											
Abstraktion											
Ziele/ Erfolgskriterien											
Konzipieren											
Black Box											
Funktionsablauf											
Analyse											
Bewertung der Analyse											
2. Besprechung TEKO (23.09.21)											
Ausarbeutng/ Realisierung											
Lösungskonzepte Ausarbeitung & Begründung											
Programierung											
Kontrolle der Ziele & Reflexion (lessons learned)											
Formatierung & Korrekturlesen											
Management Summary & Reserve											
Vorbereitung Präsentation & Präsentation (28.10.21)											

Tabelle 1 Terminplan/ Ablaufplanung

6. Infosammlung

- **«Niryo One» Informationen**

Abbildung 4 «Niryo One» Roboter

Der 6-Achsen gesteuerte Roboterknickarm wurde mittels 3-D Druck hergestellt und ist mit mehreren Servomotoren ausgerüstet. Die Programmierung kann mittels des Programms «Niryo One Studio» umgesetzt werden, das auf einer grafischen Programmiersprache vom Typ Blockly basiert, welche auf der Sprache Scratch aufgebaut ist. Es wären auch weitere Programmiersprachen möglich, wie zum Beispiel Python, C++ und MATLAB. Ebenfalls kann der Roboter anhand eines Xbox Controllers gesteuert werden. Des Weiteren besitzt der Roboter einen Learning Mode, welcher dabei helfen soll, einzelne Position zu speichern und von Hand den Roboter an die gewünschte Position zu führen.

- **«Niryo One» Aufbau**

Abbildung 5 «Niryo One» Aufbau

1. Basis	Orange
2. Schulter	Gelb
3. Oberarm	Grün
4. Ellbogen	Türkis
5. Unterarm	Blau
6. Handgelenk	Violett
7. Hand	Rot

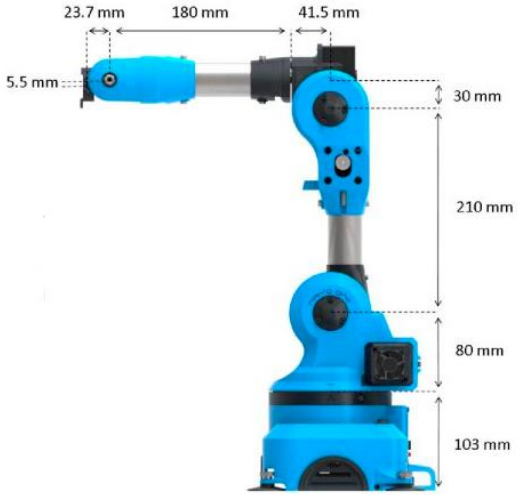


Abbildung 6 «Niryo One» Dimensionen

- **«Niryo One» Technische Daten**

- Bewegungsachsen: 6
- Gewicht: 3,3 kg
- Reichweite: 440 mm
- Nutzlast: 0,3 kg
- Basiswinkel: +/- 175 °.
- Wiederholgenauigkeit: +/- 1 mm
- Stromversorgung: 11,1V/6A
- Kommunikation: WIFI Ethernet: 2,4 GHz Bereich 802.11n / Ethernet
- WIFI: 2,4 GHz Reichweite 802.11n / Bluetooth 4.1: 2,4 GHz; 2,5 mW (4 dBm) / USB
- Schnittstelle/Programmierung: Windows/MacOS/Linux (Desktop-Anwendung), Gamepad, APIs
- Leistungsaufnahme: ~ 60 W
- Materialien: Aluminium, PLA (3D-Druck)
- Anschlüsse: 1 Ethernet + 4 USB-Anschlüsse
- Elektronik / Motorisierung: Raspberry Pi 3B / + 3 x NiryoSteppers / + 2 x Dynamixel XL - 430 / + 1 x Dynamixel XL - 320
- Kollisionserkennungssensor: Magnetsensor (am Motor)

- **«Niryo One» Anschlüsse**

Abbildung 7 «Niryo One» Anschlüsse

1. Auslöseknopf	Um den Programmablauf zu starten oder den Roboter herunterzufahren
2. Ethernet Anschluss	Um den Raspberry Pi 3B übers Kabel zu verbinden
3. 4x USB-Anschlüsse	Um externe Sensoren oder Kameras anzuschliessen
4. LED-Lampe	• Grün = bereit für die Verbindung • Rot = Alarm, Hochfahren oder bereit zum Ausschalten • Blau = Hotspot aktiv • Violet = Herunterfahren
5. CAN Bus Anschluss	Um Förderbänder anzuschliessen
6. Dynamixel XL-320 Anschluss	Um die Vakuum Pumpe anzuschliessen
7. Dynamixel XL-430 Anschluss	Keine Funktion bisher
8. 12 V Anschluss	Ansteuerbar über die Software
9. GPIO Anschlüsse (6 digitale Pins mit je 5V)	Ansteuerbar über die Software (Status entweder 0 oder 1)
10. Ein/Ausschaltknopf	Zum Ein- und Ausschalten des Roboters
11. Kabelanschlussbuchse	Um dem Roboter mit Strom zu versorgen

- **«Niryo One» Werkzeuge**

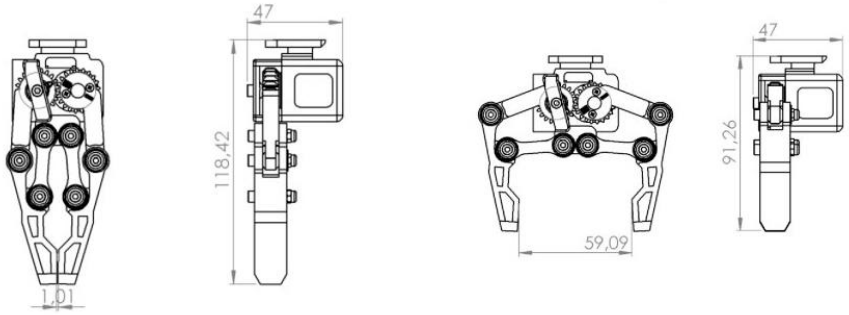


Abbildung 8 Large Gripper

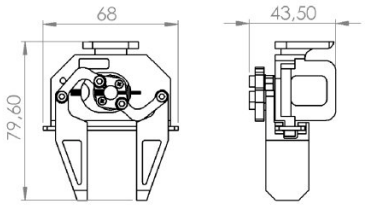


Abbildung 9 Standard Gripper

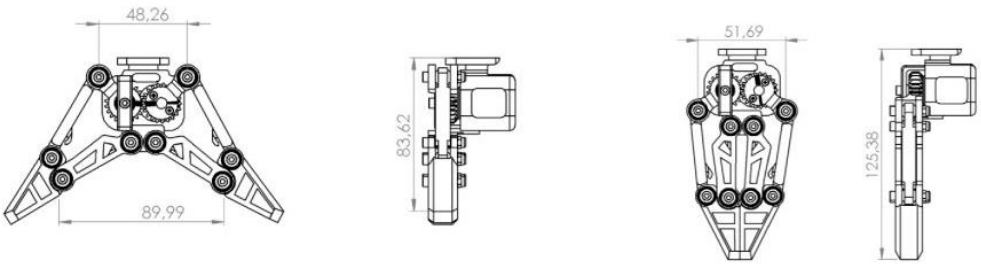


Abbildung 10 Adaptive Gripper

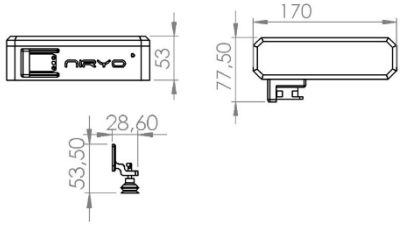


Abbildung 11 Vacuum Pump

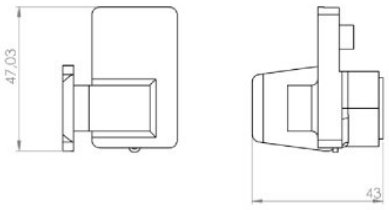


Abbildung 12 Electromagnet

- **«Vision Set»**

Abbildung 13 «Vision Set»

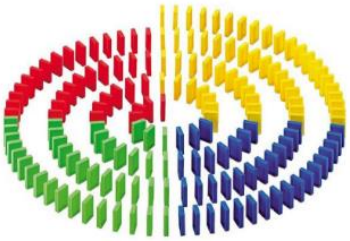
Das «Vision Set» beinhaltet eine 2D-Kamera, die auf dem Kopfteil des Roboters befestigt werden kann. Ebenfalls gehört das «Vision Board» dazu. Anhand der Kamera kann der Roboter von verschiedenen Objekten die Form oder die Farbe erkennen und mit ihnen interagieren, Bilder verarbeiten und anhand Künstlicher Intelligenz dazulernen. Das «Vision Set» ermöglicht den Einstieg in die Industrie 4.0 und somit in die autonome Produktionslinie. Die schwarzen Kreise auf dem «Vision Board» dient der Orientierung der Arbeitsflache, wo der «Niryo One» arbeiten kann. Damit der Roboter einen «Vision Pick» (Befehl an den Roboter das Objekt innerhalb des Arbeitsbereiches zu greifen) ausführen kann, muss die Arbeitsfläche, die mit den 4 schwarz/ weissen Kreisen umgeben ist, mit einem Spezialwerkzeug abgetastet werden. Dadurch entsteht mit der 2D-Kamera eine dritte Dimension, welches die Höhe darstellt. Wenn dann später ein Werkzeug gewechselt wird, rechnet der Roboter jeweils die Länge des Werkzeugs dazu und schon kann der Roboter den «Vision Pick» ohne Probleme ausführen.

- **Förderband**

Abbildung 14 Förderband

Anhand des Förderbands (700 mm) können Objekte zum «Niryo One» befördert und dort verarbeitet werden. Durch das geschwindigkeitsregulierbare Förderband wird die autonome Produktionslinie erweitert. Im Förderband Set inbegriffen ist ebenfalls ein Infrarotsensor, der die beförderten Objekte erkennen kann. Der Infrarotsensor kann über die digitalen Eingänge des Roboters programmiert werden. Wenn also der Sensor ein Objekt erkennt, leuchtet die LED-Lampe am Sensor und gibt das Signal an den Roboter weiter. Dieses Signal kann je nach Anwendung vielseitig verwendet werden, liefert aber immer nur entweder eine 0 oder 1 als Wert. Das Förderband kann je nach gewünschtem Einsatzzweck entweder vorwärts- oder rückwärtslaufen. Auch lassen sich die Orientierungskreise für das «Vision Set» an der Seite des Förderbands beliebig anbringen, um direkt vom Förderband Objekte zu erkennen und greifen zu können. Zum Förderband gehört auch die Ladestation dazu, bei der das Material gelagert werden kann. Die maximale Geschwindigkeit des Förderbands beträgt 38 mm/s.

- **Dominosteine**



Für die Ausführung des Programms sollen ungefähr 20 Dominosteine verwendet werden. Die Dominosteine sollen dazu so auf vorprogrammiertem Weg gelegt werden, um eine Kettenreaktion zu ermöglichen, die vom Roboterarm «Niryo One» ausgelöst werden soll. Ebenfalls sollten die Dominosteine eine gleichbleibende Grösse und Form besitzen. Zur Auswahl stehen Dominosteine aus Elfenbein, Kunststoff, Holz oder Metall.

Abbildung 15 Dominosteine Beispiel

- **«Niryo One Studio» Programm**

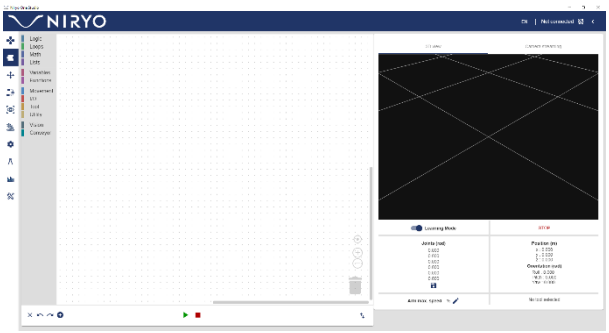


Abbildung 16 «Niryo One Studio» Programm

Anhand des «Niryo One Studio» Programms kann der Roboter verbunden, über Regler gesteuert werden, mit Blockly programmiert werden, einzelne Sequenzen abspielen, das «Vision Set» bedienen, das Förderband bedienen, diverse Einstellungen machen, den Programmcode umwandeln, sich kalibrieren und den Status und die Alarmloghistorie anzeigen. Das «Niryo One Studio» Programm wird als Plattform verwendet, um alle Einstellungen und Anpassungen am Programm umzusetzen. Das Programm hat auch ein Feld im oberen rechten Eck, dass die «Ist-Position» als 3D Modell anzeigt, in welcher Position der Roboter steht. Ebenfalls kann im gleichen Feld auch der Kamera Stream angezeigt werden. Dies ermöglicht zu sehen, was die Roboterhand sieht. Dieser Stream ist für den «Vision Pick» sehr hilfreich, da man direkt im Programmablauf sieht, ob die Kamera auf dem Roboter auch die Arbeitsfläche, die mit den Kreisen eingerahmt ist, erkennt. Das «Niryo One Studio» Programm ermöglicht die geschriebenen Programme, mit den ganzen eingespeicherten Positionen und Arbeitsflächen, intern auf dem Roboter oder extern auf dem Computer zu speichern. Auch ermöglicht das «Niryo One Studio» Programm, dass man direkt den Programmcode abspielen kann. Für die Widergabe des geschriebenen Codes wird das ganze Programm von Blockly in Python übersetzt und erst dann abgespielt. Im «Niryo One Studio» Programm sind die Farben Grün, Blau und Rot vorprogrammiert, welche von der Kamera erkannt werden können. Weitere Farben werden unter sonstigen Farben erkannt.

7. Ideensammlung

8. Anforderungsliste

Anforderungen	F	M	W	Verantwortlich
Geometrie				
Die Dominosteine innerhalb von 440 mm Radius legen und +/- 175 °	F			Programmierer
Maximaler Weg mit Förderband 700 mm	F			Hersteller
Dominosteine sollten für alle Greifer greifbar sein			W	Programmierer
Der Komplette Aufbau sollte auf einem TEKO Schreibtisch Platz haben			W	Programmierer
Kinematik				
Maximale Geschwindigkeit des Förderbands beträgt 38 mm/s	F			Hersteller
Genauigkeit des «Niryo One» sollte +/- 1mm sein	F			Hersteller
Kräfte				
Maximale Nutzlast des «Niryo One» beträgt 0,3 kg	F			Hersteller
Maximales zulässiges Gewicht auf dem Förderband beträgt 2 kg	F			Hersteller
Gewicht des «Niryo One» 3.2 kg	F			Hersteller
Energie				
Die Kamera des «Vision Sets» soll direkt am «Niryo One» über die USB-Schnittstelle angeschlossen werden		M		Hersteller
Stromversorgung für den «Niryo One» und das Förderband soll 11.1 Volt / 6A betragen	F			Hersteller
Maximaler Stromverbrauch des «Niryo One» sollte 60 W betragen	F			Hersteller
Stoff				
Roboter - Aluminium, PLA (3D Druck)	F			Hersteller
Dominosteine (Kunststoff)			W	Programmierer
Signal				
IR (Infrarotsensor) Distanzerkennung 6 – 80 cm			W	Montage
Kommunikation: ➢ Ethernet ➢ WIFI: 2.4 GHz Range 802.11n ➢ Bluetooth 4.1: 2,4 GHz ; 2,5 mW (4 dBm) ➢ USB	F			Hersteller

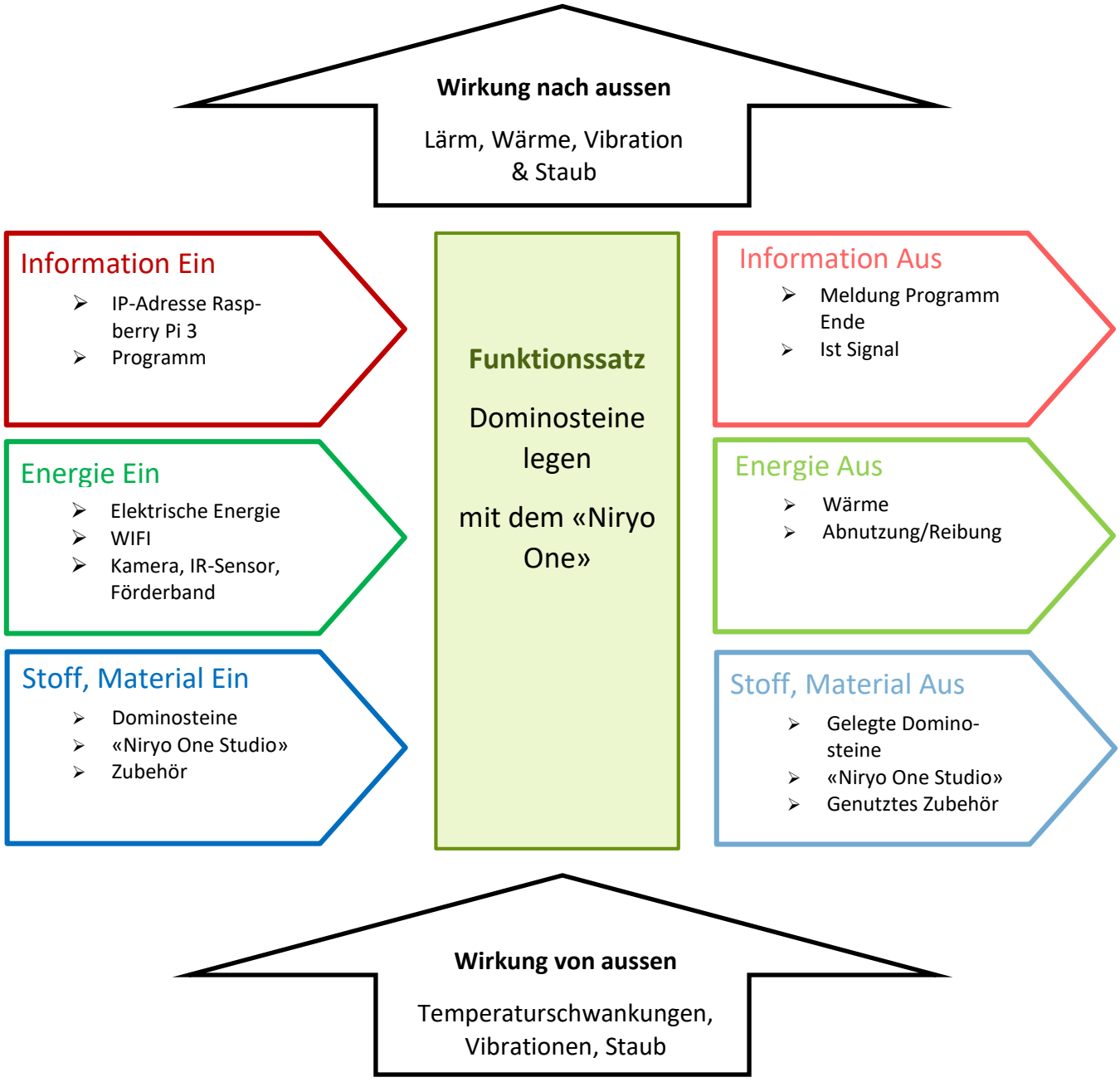
Sicherheit				
Kollisionssensor über einen Magnetsensor im Motor	F			Hersteller
Netzstecker/Schutzstecker nach SEV	F			Hersteller
Endanschlag			W	Programmierer
Ergonomie				
Die Bedienung des «Niryo One» sollte per Knopfdruck möglich sein			W	Programmierer
Kontrollen/Wartung				
Kalibrierung des «Niryo One» Roboters	F			Programmierer
Montage				
Der Roboter sollte vormontiert sein	F			Hersteller
Transport				
Der ganze Aufbau sollte individuell von einem Ort zum anderen transportierbar sein		M		Programmierer
Gebrauch				
Lange Positionsgenauigkeit und Wiederholungs-rate		M		Programmierer
Einfache Bedienung		M		Programmierer
Kosten				
Das Material sollte von der TEKO zur Verfügung gestellt werden und somit keine Kosten gene-rieren		M		Programmierer
Umfang				
Es sollen 20 Dominosteine verwendet werden			W	Programmierer
Termin				
Abgabetermin der Dokumentation und des Pro-gramms (11.10.21)	F			Projektleiter
Präsentation des Programms/Ablaufes (28.10.21)	F			Projektleiter

Tabelle 2 Anforderungsliste

F = Festforderung: (Muss unbedingt erfüllt wer-den, andernfalls ist das Pro-dukt für die gestellte Aufgabe untauglich.)	M = Mindestforderung: (Dürfen nach der günstigen Seite hin unterschritten oder überschritten werden.)	W = Wunsch: (Sollten nach Möglichkeit be-rücksichtigt werden. Ev. mit Zugeständnis, dass ein be-grenzter Mehraufwand zuläs-sig ist.)

9. Konzipieren

9.1 Black Box



9.2 Beschreibung der Gesamtfunktion

Die Dominosteine sollen auf das Förderband gelegt werden. Dieses soll die Dominosteine zum «Vision Workspace» befördern. Dort soll ein Dominostein von der Kamera erkannt und dadurch das Förderband gestoppt werden. Anschliessend soll der «Niryo One» ein «Vision Pick» machen, wobei er die Position und Farbe erkennt, um den Dominostein zu greifen. Der Dominostein soll dann auf dem Tisch nach programmiertem Weg gelegt werden. Dieser Vorgang soll mehrmals wiederholt werden, um mindestens 20 Dominosteine zu legen. Zum Abschluss soll der «Niryo One» Roboter die Kettenreaktion auslösen, indem er den zuletzt gelegten Dominostein anstösst.

Die folgende Analyse bietet unterschiedliche Möglichkeiten, wie die Gesamtfunktion ausgeführt und welche Materialien dafür verwendet werden sollen. Während es gesamten Vorgangs werden Lärm, Vibrationen und Wärme an die Umgebung und die Roboterkomponenten abgegeben.

Die Systemgrenze begrenzt sich auf das Programmieren des Ablaufs/ Programms und nicht auf die Berechnungen des Roboterweges oder der Auslegung der Komponenten.

- **Input**

Dem Knickarmroboter «Niryo One» sollte Energie und Material zugefügt und Informationen anhand des Sensors in das Programm eingelesen werden. Dafür sollten als Material mindestens 20 Dominosteine bereitstehen.

- **Output**

Während es gesamten Vorgangs werden Lärm, Vibrationen und Wärme an die Umgebung und die Roboterkomponenten abgegeben. Nach Beendigung des Ablaufs/ Programms kann das Programm/ der Ablauf immer wieder neugestartet werden.

- **Wirkungen von aussen**

Von aussen wirken Temperaturschwankungen, Vibrationen und Staub auf den Roboterarm ein. Diese sollten jedoch keine Auswirkung auf den Ablauf/ das Programm haben, da sie sehr gering sind.

- **Wirkungen nach aussen**

Nach aussen wird Lärm, Energie in Form von Wärme, Vibrationen und Staub abgegeben. Ausserdem werden die Dominosteine gelegt.

9.3 Funktionsstruktur

1. •Domonosteine sollen willkürlich auf das Förderband gelegt werden.
2. •Das Förderband soll die Dominosteine in den «Vision Workspace» befördern, der markiert ist mit den schwarzen Kreisen auf weissem Hintergrund.
3. •Die Kamera auf dem «Niryo One» soll die Position und die Farbe innerhalb des «Vision Workspace» erkennen.
4. •Der Infrarotsensor der am Förderband hinter dem «Vision Workspace» befestigt ist, soll die Steine, die seinen Erkennungsbereicht passieren, erkennen.
5. •Das Programm, das den Roboter und das Förderband steuert, soll den Befehl geben, dass das Förderband gestoppt werden soll.
6. •Der «Niryo One» Roboter soll dann einen «Vision Pick» machen, wobei der Greifer den bereits erfassten Stein, der auf dem Förderband liegt greift.
7. •Der Stein soll dann vom Roboterarm auf die Tischplatte gestellt werden, der mit weiteren Steinen eine Dominosteinkette bilden soll.
8. • Das Programm soll dann den «Niryo One» Roboter, durch das Anstossen der Steine, dazu bringen die Dominosteinkettenreaktion auzulösen.

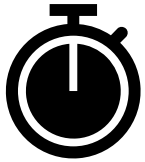
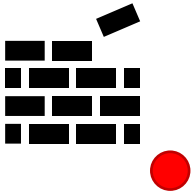
9.4 Analyse

1 **Dominosteine auswählen**				
Holz-Dominosteine schwarz	Elfenbein-Domino-steine Weiss	Farbige Kunst-stoff-Domino-steine	Dominosteine aus Stahl	
Variante 1.1	Variante 1.2	Variante 1.3	Variante 1.4	
Pkt. Max. 22	Pkt. Max. 16	Pkt. Max. 27	Pkt. Max. 13	
				

2 **Aufbau entscheiden**							
Förderband	Ladestation	Infrarot Sensor	USB-Kamera (Sony IMX322-Sensor)	Kunstsoff/ Saug-napf-Füsse	Endanschlag	Aluminiumführung	«Vision Work-space»
Variante 2.1	Variante 2.2	Variante 2.3	Variante 2.4	Variante 2.5	Variante 2.6	Variante 2.7	Variante 2.8
Pkt. Max. 31	Pkt. Max. 22	Pkt. Max. 27	Pkt. Max. 26	Pkt. Max. 20	Pkt. Max. 28	Pkt. Max. 33	Pkt. Max. 22
							

3 **Werkzeuge/ Greifer entscheiden**					
Standard Gripper	Large Gripper	Adaptiv Gripper	Vakuumpumpe	Elektromagnet	
Variante 3.1	Variante 3.2	Variante 3.3	Variante 3.4	Variante 3.5	
Pkt. Max. 13	Pkt. Max. 30	Pkt. Max. 24	Pkt. Max. 14	Pkt. Max. 16	
					

4 **Programmiersprache wählen**				
Python	Blockly	C++	Matlab	
Variante 4.1	Variante 4.2	Variante 4.3	Variante 4.4	
Pkt. Max. 14	Pkt. Max. 39	Pkt. Max. 20	Pkt. Max. 17	
				

5 **Programmfokus festlegen**				
Qualität	Effizient	Kurzes Programm	Programm mit Unterprogramm	
Variante 5.1	Variante 5.2	Variante 5.3	Variante 5.4	
Pkt. Max. 17	Pkt. Max. 19	Pkt. Max. 18	Pkt. Max. 20	
				

6 **Sicherheitsmechanismen planen**						
Endanschlag	Infrarotsensor	Geschwindigkeitsregulierung	Notaus Schalter	Lichtschranke	Ummantelung des Aufbaus	
Variante 6.1	Variante 6.2	Variante 6.3	Variante 6.4	Variante 6.5	Variante 6.6	
Pkt. Max. 34	Pkt. Max. 29	Pkt. Max. 27	Pkt. Max.21	Pkt. Max. 18	Pkt. Max. 23	
						

Tabelle 3 Analyse

9.5 Analysenbewertung

1		Bewertung der Varianten							
		V1.1		V1.2		V1.3		V1.4	
Dominosteine auswählen	GW	GP	Pkt.	GP	Pkt.	GP	Pkt.	GP	Pkt.
Einfache Beschaffung	1	3	3	2	2	3	3	2	2
Kosten	1	3	3	1	1	4	4	1	1
Handhabung mit dem Roboter	3	2	6	3	9	3	9	2	6
Fehleranfällig	2	1	2	1	2	3	6	0	0
Ökologie & Umwelt	2	4	8	1	2	3	6	2	4
Summe		14	22	7	16	16	27	7	13

GW = Gewichtungsfaktor / **V** = Variante / **Pkt.** = Punkte / **GP** = Gewichtungsfaktor x Punkte |
Punktwertskala: 4 = sehr gut / 3 = gut / 2 = ausreichend / 1 = gerade noch tragbar / 0 = unbrauchbar

Erklärung:

Einfache Beschaffung: Sollte in kurzer Zeit, ohne grossen Aufwand aufzutreiben sein

Kosten: Es sollten keine Kosten anfallen gemäss Anforderungsliste

Handhabung mit dem Roboter: Ist von Wert für die Genauigkeit beim Greifen der Steine

Fehleranfällig: Sollte keine Probleme mit dem Programm und der Erkennung geben

Ökologie & Umwelt: Es sollte ein nachwachsender oder recycelbarer Rohstoff sein

2		Bewertung der Varianten															
		V2.1		V2.2		V2.3		V2.4		V2.5		V2.6		V2.7		V2.8	
Aufbau entscheiden	GW	GP	Pkt.	GP	Pkt.	GP	Pkt.	GP	Pkt.	GP	Pkt.	GP	Pkt.	GP	Pkt.	GP	Pkt.
Notwendigkeit	3	4	12	2	6	2	6	4	12	2	6	2	6	4	12	2	6
Sicherheitstechnisch	2	4	8	2	4	4	8	2	4	3	6	3	6	3	6	3	6
Wiederholgenauigkeit	3	3	9	3	9	3	9	2	6	2	6	4	12	4	12	2	6
Erleichterung der Handhabung	1	4	4	3	3	4	4	4	4	2	2	4	4	3	3	4	4
Summe		15	31	10	22	13	27	12	26	9	20	13	28	14	33	11	22

GW = Gewichtungsfaktor / **V** = Variante / **Pkt.** = Punkte / **GP** = Gewichtungsfaktor x Punkte | **Punktwertskala:** 4 = sehr gut / 3 = gut / 2 = ausreichend / 1 = gerade noch tragbar / 0 = unbrauchbar

Erklärung:

Notwendigkeit: Notwendig zur Erfüllung des Funktionsablaufes und des Programmes

Sicherheitstechnisch: Es sollten Sicherheitsbauteile verwendet werden, um einzugreifen, wenn ein Stein ungeplant anders ist

Wiederholgenauigkeit: Wichtig damit die Steine gelegt werden können ohne Zwischenfall

Erleichterung der Handhabung: Es sollte die Handhabung im Programm und den Ablauf erleichtern

3		Bewertung der Varianten									
		V3.1		V3.2		V3.3		V3.4		V3.5	
Werkzeuge/ Greifer entscheiden	GW	GP	Pkt.	GP	Pkt.	GP	Pkt.	GP	Pkt.	GP	Pkt.
Handhabung	2	2	4	3	6	3	6	2	4	2	4
Geschwindigkeit	2	1	1	4	8	4	8	1	2	1	2
Genauigkeit	3	2	6	4	12	2	6	2	6	2	6
Klemmkraft	1	2	2	4	4	4	4	2	2	4	4
Summe		7	13	15	30	13	24	7	14	9	16

GW = Gewichtungsfaktor / **V** = Variante / **Pkt.** = Punkte / **GP** = Gewichtungsfaktor x Punkte | **Punktwertskala:** 4 = sehr gut / 3 = gut / 2 = ausreichend / 1 = gerade noch tragbar / 0 = unbrauchbar

Erklärung:

Handhabung: Einfachheit und Handhabung ist wichtig für einen Unkomplizierten Umgang. Ebenfalls ohne Schwierigkeiten

Geschwindigkeit: Je schneller der Greifer arbeitet und noch immer genau ist, desto effizienter wird der Ablauf

Genauigkeit: Die Genauigkeit ist sehr wichtig, um eine saubere Kette mit den Dominosteinen zu legen

Klemmkraft: Die Klemmkraft ist von Bedeutung nur damit der Stein auch im Greifer hält

4		Bewertung der Varianten							
		V4.1		V4.2		V4.3		V4.4	
Programmiersprache wählen	GW	GP	Pkt.	GP	Pkt.	GP	Pkt.	GP	Pkt.
Einfachheit	3	1	3	4	12	2	6	1	3
Lernzeit	3	1	3	4	12	2	6	3	9
Komplexität	1	2	2	3	3	2	2	2	2
Handhabung	3	2	6	4	12	2	6	1	3
		6	14	15	39	8	20	7	17

GW = Gewichtungsfaktor / **V** = Variante / **Pkt.** = Punkte / **GP** = Gewichtungsfaktor x Punkte | **Punktwertskala:** 4 = sehr gut / 3 = gut / 2 = ausreichend / 1 = gerade noch tragbar / 0 = unbrauchbar

Erklärung:

Einfachheit: Das Erlernen der Programmiersprache sollte nicht all zu kompliziert sein, um das Projekt in der gewünschten Zeit auch erledigen zu können

Lernzeit: Es sollte schnell erlernbar sein, um auch effizient arbeiten zu können

Komplexität: Das Erlernen und der Umgang sollte nicht zu kompliziert sein, um dem Terminplan zeitgemäss zu folgen

Handhabung: Es sollte die Handhabung im Programm und den Ablauf erleichtern

5		Bewertung der Varianten							
		V5.1		V5.2		V5.3		V5.4	
Programmfokus festlegen	GW	GP	Pkt.	GP	Pkt.	GP	Pkt.	GP	Pkt.
Umfang	1	1	1	3	3	4	4	2	2
Aufwand	2	3	6	2	4	4	8	2	4
Geschwindigkeit	2	2	4	4	8	2	4	3	6
Handhabung	2	4	6	2	4	1	2	4	8
Summe		10	17	11	19	11	18	11	20

GW = Gewichtungsfaktor / **V** = Variante / **Pkt.** = Punkte / **GP** = Gewichtungsfaktor x Punkte | **Punktwertskala:** 4 = sehr gut / 3 = gut / 2 = ausreichend / 1 = gerade noch tragbar / 0 = unbrauchbar

Erklärung:

Umfang: Der Umfang sollte nicht den Ausgang des Programmes benachteiligen

Aufwand: Es sollte schnell aufbaubar sein, um auch effizient arbeiten zu können

Geschwindigkeit: Beim Umsetzen des Programmes sollte die Geschwindigkeit eine Rolle spielen, aber nicht der Fokus sein

Handhabung: Der Fokus sollte die Handhabung im Programm und den Ablauf erleichtern und nicht erschweren

6		Bewertung der Varianten											
		V6.1		V6.2		V6.3		V6.4		V6.5		V6.6	
Sicherheitsmechanismen planen	GW	GP	Pkt.	GP	Pkt.	GP	Pkt.	GP	Pkt.	GP	Pkt.	GP	Pkt.
Handhabung	3	4	12	3	9	3	9	3	9	2	6	3	9
Zeitaufwand	2	4	8	4	8	2	4	3	6	1	2	3	6
Kosten	2	4	8	4	8	3	6	2	4	1	2	2	4
Effizienz	2	3	6	2	4	4	8	1	2	4	8	2	4
Summe		15	34	13	29	12	27	9	21	8	18	10	23

GW = Gewichtungsfaktor / **V** = Variante / **Pkt.** = Punkte / **GP** = Gewichtungsfaktor x Punkte | **Punktwertskala:** 4 = sehr gut / 3 = gut / 2 = ausreichend / 1 = gerade noch tragbar / 0 = unbrauchbar

Erklärung:

Handhabung: Die Sicherheitsmechanismen sollten nicht den Vorgang behindern, sondern absichern

Zeitaufwand: Es sollte schnell anzubringen sein und sonst keine weitere Arbeit machen

Kosten: Gemäss Anforderungsliste sollen keine zusätzlichen Kosten entstehen

Effizienz: Das Sicherheitsbauteil sollte die Arbeit erledigen, wofür es eingesetzt wird

9.6 Auswahl aus der Analyse

Variante A	Punktzahl	Variante B	Punktzahl	Variante C	Punktzahl
1.1	22	1.2	16	1.3	27
2.1	31	2.1	31	2.1, 2.2, 2.3	31+22+27
2.5	20	2.5	20	2.4, 2.6, 2.7, 2.8	26+28+33+22
3.1	13	3.3	24	3.2	30
4.3	20	4.1	14	4.2	39
5.1	17	5.1	17	5.1, 54	17+20
6.1, 6.2	34+29	6.1	34	6.1, 6.2, 6.3	34+29+27
Summe	186	Summe	156	Summe	412

Tabelle 4 Bewertung Analyse

Gemäss Punktzahl (412 Punkte) wird Analyseweg C gewählt. Diese hohe Punktzahl resultiert daher, dass im Analyseweg C mehrere Varianten gewählt wurden, wohingegen bei Analyseweg A und B meist nur eine Variante gewählt wurde. Die Abwägung der einzelnen Varianten hat gut geholfen, um zu sehen was nützlich ist und was noch möglich wäre, wenn man mehr Zeit und Geld zur Verfügung hätte. Bei der Auswahl sind ebenfalls schon erste Untersuchungen der Varianten gemacht worden, was die Entscheidung einfacher gemacht hat. Die Analyse hat dahin geholfen zu sehen, dass die Ideen, die schon zu Beginn vorhanden waren, bestätigt wurden.

9.7 Begründung & Entscheidung

Die Kunststoffdominosteine sind sehr praktisch, da sie leicht, farblich unterschiedlich, recycelbar und gut zu greifen für den Roboter sind. Der einzige Nachteil, der mir aufgefallen ist, ist die Einbuchtung im Stein, was verhindert, dass die Vakuumpumpe verwendet werden kann.

Der Aufbau mit dem Förderband, der Ladestation, Infrarotsensor, Kamera, Endanschlag, Aluminiumführung und dem «Vision Workspace» ist auch notwendig, um die Anforderungen, die an das Programm und den Ablauf gestellt sind, zu erfüllen. Die Aluminiumführung hilft mir, dass das Förderband mit dem «Vision Workspace» und der Ladestation fix verbunden werden können, um die Positionsgenauigkeit zu gewährleisten. Der Einbau des Infrarotsensors und der Kamera in die Programmierung werden für mich eine grosse Herausforderung darstellen, da ich keine Erfahrung mit Sensoren oder dem Robotervision habe. Der Endanschlag hat die Funktion, dass die Dominosteine nicht vom Förderband fallen und somit die bereits legenden Dominosteine umwerfen würden.

Die Auswahl des Greifers war allerdings schwieriger, da jeder einzelne Greifer seine Vor- und Nachteile hatte, welche sich unterschiedlich auf den Ablauf auswirken würden. Somit musste ich diese Vor- und Nachteile in Bezug auf meine Anforderungen ans Projekt abwägen. Nach einigen Probedurchläufen wählte ich den «Large Gripper», da er präzise, ohne Verschiebungen und ohne Probleme die gewünschte Arbeit erfüllt.

Die Auswahl der Programmiersprache war schnell geklärt. Nach ein paar Vergleichen und der Komplexität der anderen Programmiersprachen, wählte ich Blockly. Es erlaubte mir eine eher visuelle Programmierung und macht das ganze Programm anschaulich und übersichtlicher. Somit konnte ich ohne zusätzliche Schnittstelle zwischen Roboter und Programm arbeiten.

Das «Niryo One Studio» Programm erlaubt es mir schnell und einfach Anpassungen zu machen, wenn Änderungen notwendig sind.

Der Fokus bei der Programmierung, der auf die Qualität und das Programm mit Unterprogrammen liegt, erlaubt es mir das Projekt ohne zusätzlichen Mehraufwand anzugehen. Der Fokus auf die Effizienz oder ein möglichst kurzes Programm hätte das Projekt unnötig in die Länge gezogen.

Bei der Planung der Sicherheitsmechanismen war klar, dass nicht noch weitere Komponenten dazu kommen sollen um Kosten, Fokus und Zeit zu sparen. Deswegen wurde der Endanschlag gewählt, da er über einem Infrarotsensor verfügt und der Programmablauf mit Geschwindigkeitsregulierung angepasst ist, um die Sicherheiten zu geben die notwendig sind.

9.8 Definitiver Aufbau

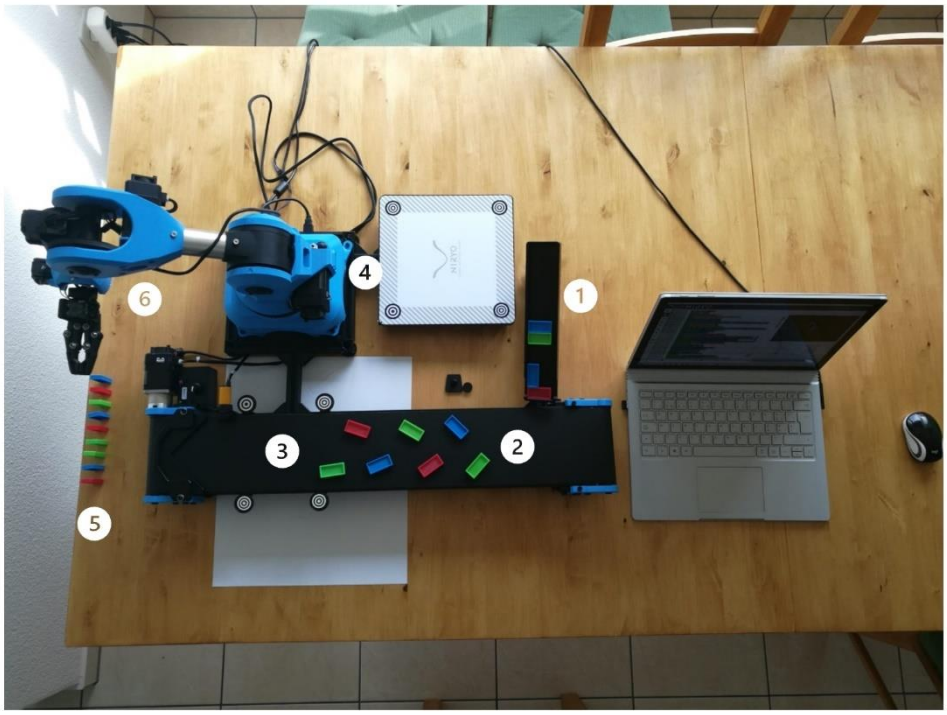


Abbildung 17 Definitiver Aufbau nach der Auswahl der Analyse

Im oberen Bild sieht man den definitiven Aufbau, nachdem die Analyse abgeschlossen war. Wie man auf dem Bild sieht, sind zwei weisse A4 Blätter unterlegt. Dies ist dazu da, dass die Kamera einen besseren Kontrast zum Förderband hat, auch bei etwas schlechterer Lichteinwirkung. Die genaue Anordnung der einzelnen Elemente wurde nach und nach bis zu diesem Punkt verändert. Was nun an welcher Position gemacht wird, wird in den folgenden Kapiteln beschrieben.

10. Lösung der Aufgabe

10.1 Verbindung mit dem «Niryo One» herstellen

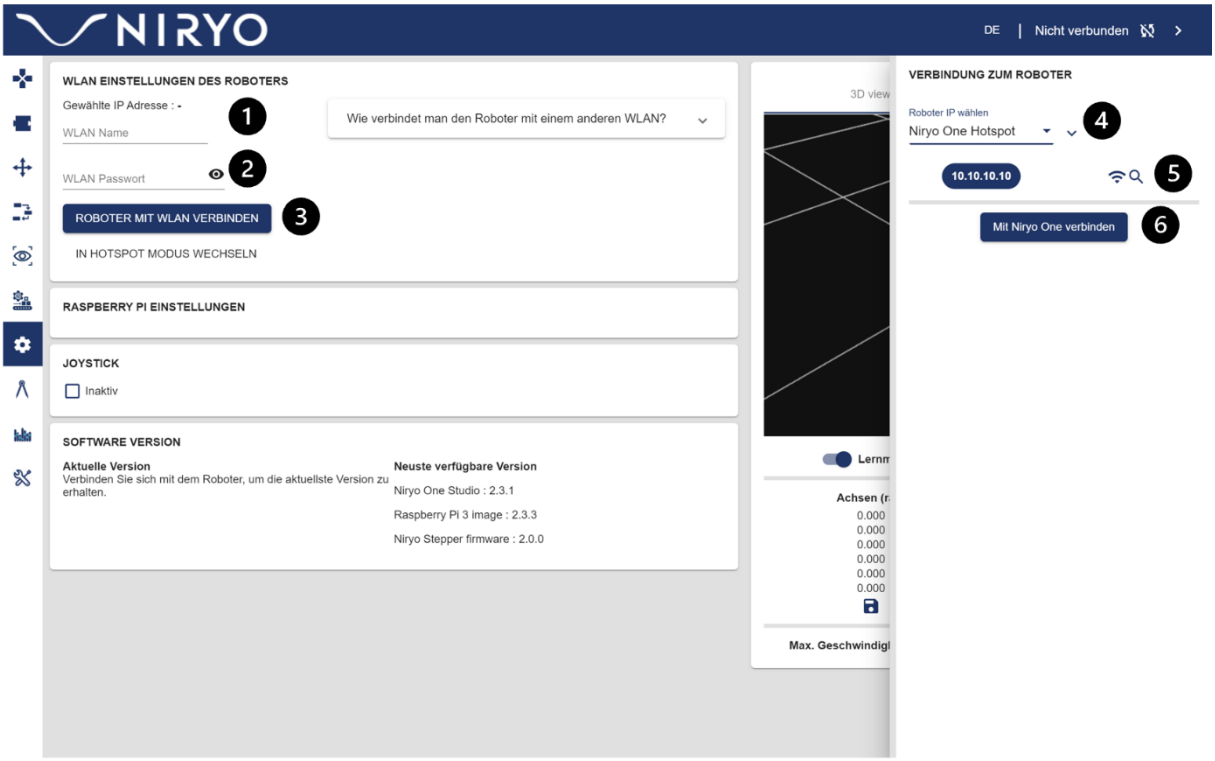


Abbildung 18 Verbindung mit dem «Niryo One» herstellen

Nachdem ich den Roboter eingeschalten habe, geht er nach einigen Sekunden in den Hotspot Modus über, wobei die LED auf der Rückseite des Roboters blau leuchtet. Danach muss man dem Roboter den gewünschten W-Lan Namen (**1**) und das W-Lan Passwort (**2**) eingeben. Danach habe ich den Roboter mit drücken der Schaltfläche «Roboter mit WLAN-Verbinden» (**3**) verbunden. Anschliessend habe ich mit drücken des W-Lan Symbols (**5**) die neu zugewiesen IP-Adresse gesucht. Bei dem Dropdown Menü (**4**) habe ich dann den Roboter ausgewählt und bei der Schaltfläche «Mit Niryo One verbinden» die Verbindung hergestellt.

10.2 Kalibrierung

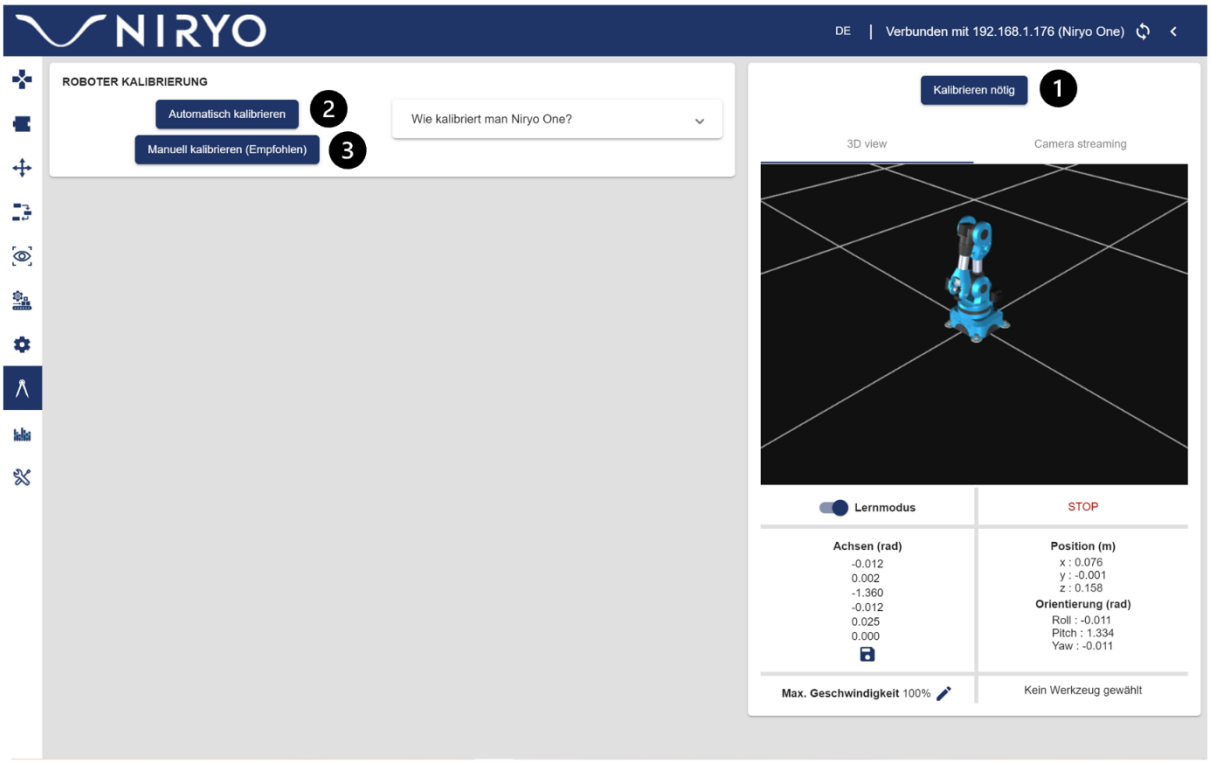


Abbildung 19 Kalibrierung

Gleich nach der Verbindung fordert mich das Programm auf, den Roboter zu kalibrieren (**1**). Dafür gibt es zwei Varianten. Automatische (**2**) oder manuelle (**3**) Kalibrierung. Bei der Kalibrierung fährt der Roboter die Endlager des Roboters ab und nullt sie. Bei der manuellen Kalibrierung fährt man die eingezeichneten Dreiecke auf dem Roboter an, sodass Dreieck auf Dreieck steht. Nachdem ich die Kalibrierung beendet habe, habe ich mich dem Kennenlernen des Roboters gewidmet.

10.3 Steuerung

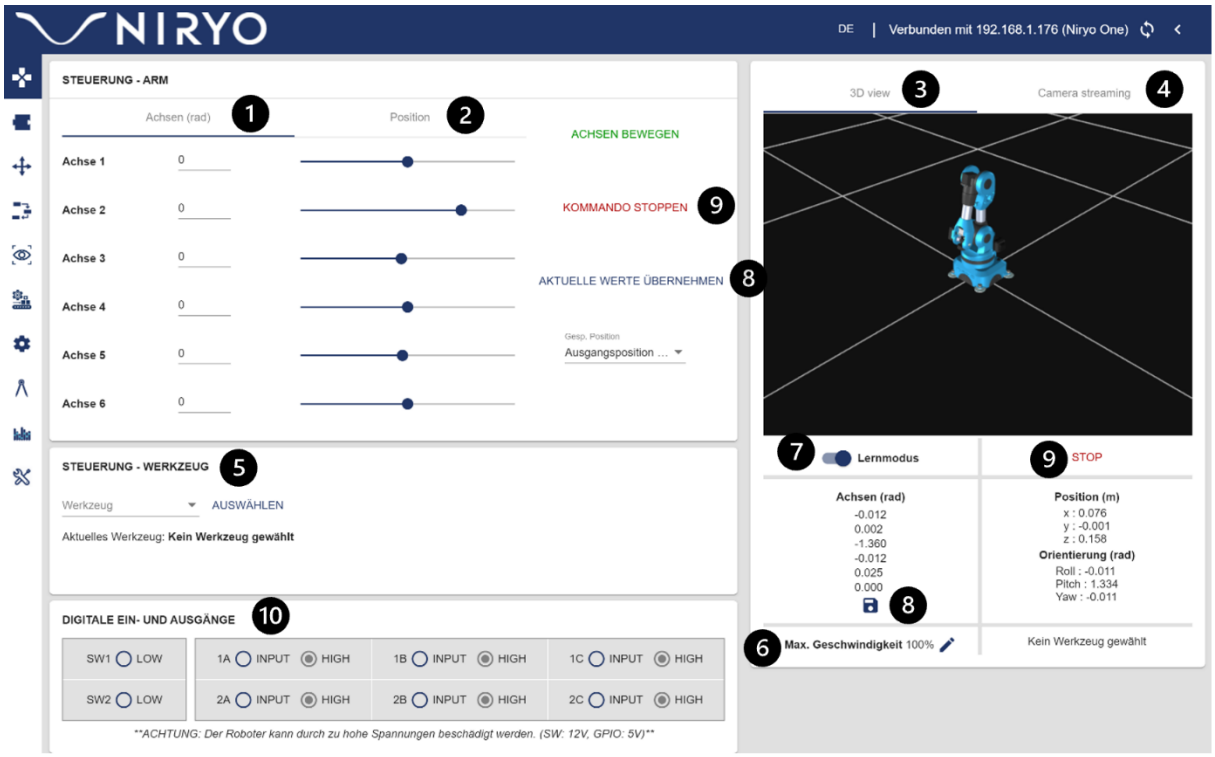


Abbildung 20 «Niryo One» Studio - Steuerung

Zu Beginn musst ich mich zuerst mit der Robotersteuerung vertraut machen. Dafür habe ich zwei Möglichkeiten ausprobiert. Zuerst versuchte ich alle Achsen in der Einheit Radiant mit den Winkeln (**1**) anzusteuern. Danach probierte ich alle Achsen mit den Positionen x, y und z in Meter und der Orientierung der Greifhand in der Einheit Radiant (**2**) zu bewegen. Dadurch habe ich mich mit den Bewegungen des Armes und der Werkzeuge (**5**) und mit den unterschiedlichen Steuerungsarten vertraut machen können. Auf der rechten Seite des Programmes hatte ich immer den «3D View» (**3**) oder den Kamera Stream (**4**) zur Unterstützung, der mir die aktuelle Lage oder das aktuelle Bild anzeigte. Bei dem Feld «Max. Geschwindigkeit» (**6**) konnte ich verschieden Bewegungen anhand der Geschwindigkeit verlangsamen oder verschnellern. Der Lernmodus (**7**), der die Fixierung der Achsen löst, ermöglichte es mir den Roboter von Hand an eine gewünschte Position zu bewegen, um diese dann einzuspeichern (**8**). Die Schaltfläche «STOP» (**9**) ermöglicht mir einen Unterbruch der Bewegung. Bei den Digitalen Ein- und Ausgängen (**10**) habe ich die Einstellungen für das jeweilige Element gemacht, das ich auf der Rückseite des Roboters angeschlossen habe.

10.4 Förderband

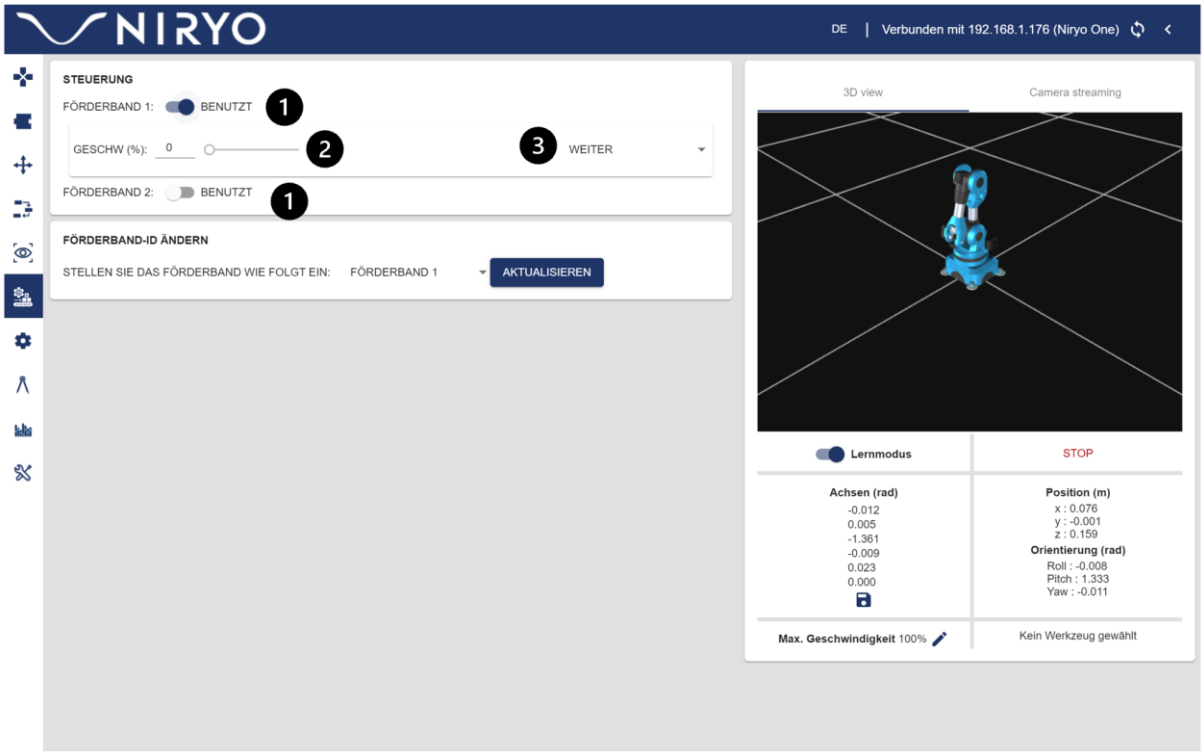


Abbildung 21 Förderband

Nachdem ich mich mit den Bewegungen des Roboters vertraut gemacht hatte, habe ich mir das Förderband angeschaut. Es gäbe die Möglichkeit zwei Förderbänder (**1**) anschliessen, da ich aber nur eines zur Verfügung hatte ist auch nur eines ausgewählt. Ich habe das Förderband 1 über den CAN Bus Anschluss auf der Rückseite eingesteckt und das Förderband 1 ausgewählt. Die Förderbandrichtung (**3**) und Geschwindigkeit (**2**) habe ich über das Verschieben des Reglers oder der prozentualen Einstellung getestet. Dies muss vor jedem Aufstarten des Roboters gemacht werden, um ein Programmcode laufen zu lassen, bei dem das Förderband benutzt werden soll.

10.5 Kamera & Vision Arbeitsbereich

Die Kamera wird über eine der vier USB-Schnittstellen auf der Rückseite des Roboters eingesteckt. Dabei erkannte ich, dass man die Kamera nicht wie gewünscht auf dem Roboter befestigen kann. Das Bauteil, um die Kamera auf dem Handstück zu befestigen, fehlte. Das bisherige Bauteil hatte keine Möglichkeit, die Kamera über eine Verschraubung zu befestigen. Daher musste das Teil erst bestellt werden. Dies dauerte zwei Wochen und warf mich in meinem Zeitplan etwas zurück.

(Die Roten Kreise Veranschaulichen die fehlenden Löcher inklusive Muttern für die Verschraubung der Kamera auf dem Handstück)

Abbildung 22 Kamerabefestigung

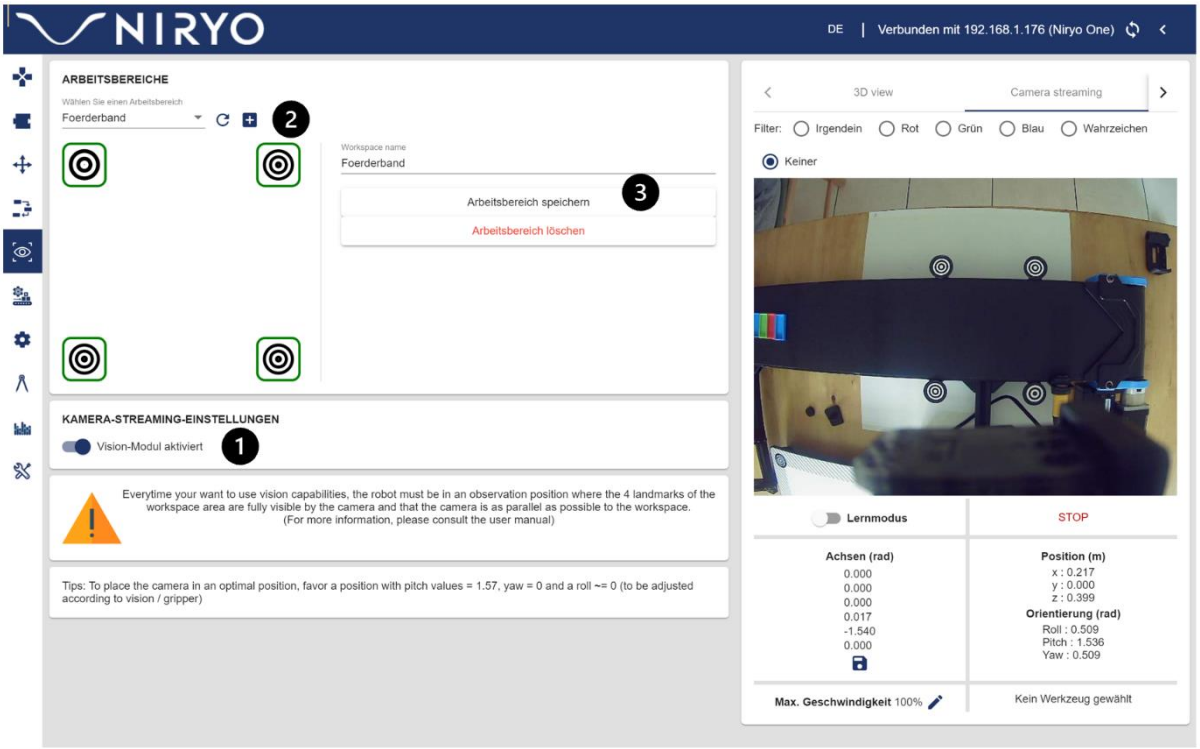


Abbildung 23 Vision Arbeitsbereich

Nachdem ich das Bauteil erhielt und ersetzt hatte, konnte ich den Kamera Steam (**1**) starten und testen. Um den Vision Arbeitsbereich im Programm nutzen zu können, musste ich den Arbeitsbereich erst mit einem Spezialwerkzeug abfahren und gemäss Anleitung vom Programm abspeichern (**3**). Diesen Ablauf habe ich mit der «+» (**2**) Schaltfläche gestartet.

10.6 Programmierung

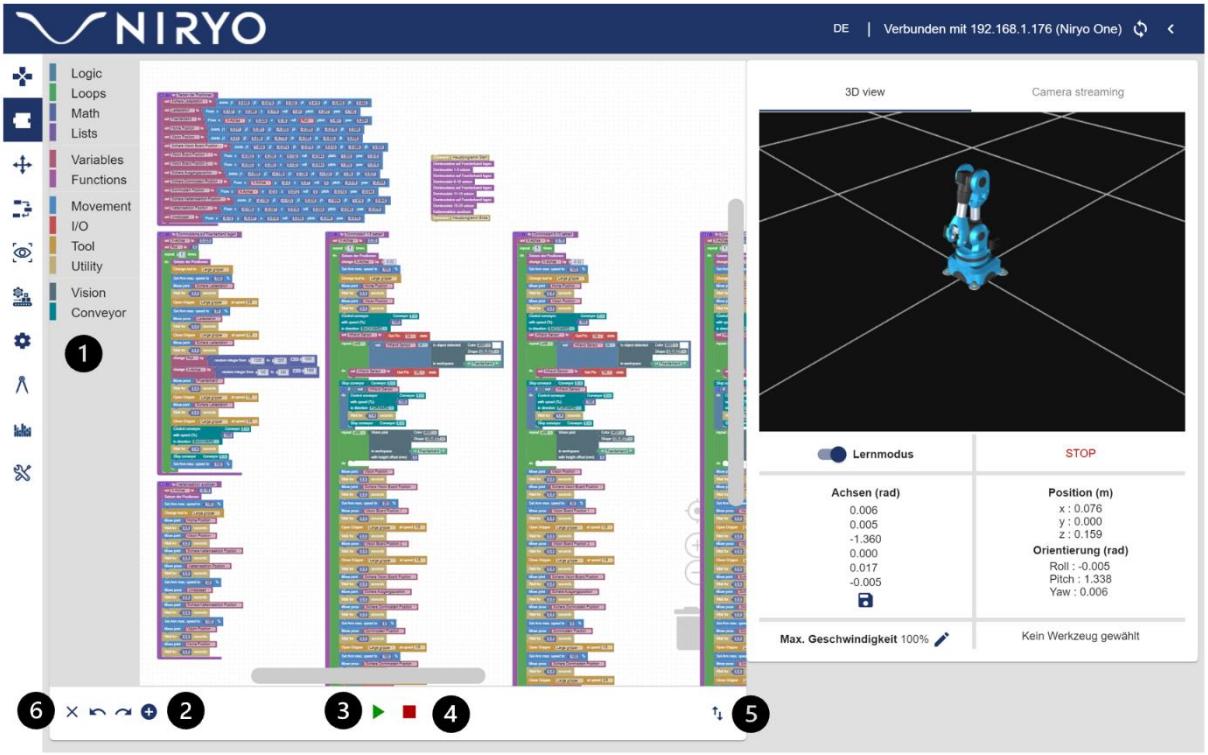
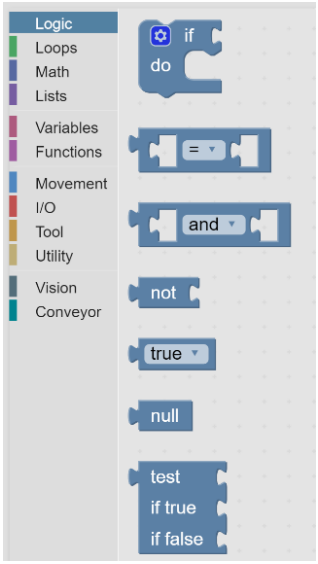


Abbildung 24 Programmierung

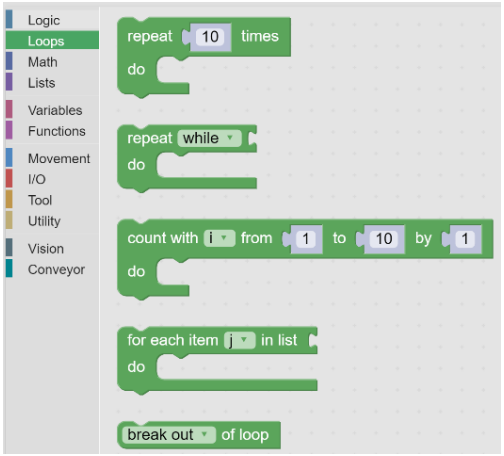
Nachdem ich mich mit dem «Niryo One Studio» Programm vertraut gemacht hatte, habe ich mich dem Programmieren gewidmet. Auf der linken Seite (**1**) habe ich eine grosse Auswahl von Bausteinen, die ich verwenden kann. Mit der Schaltfläche «+» (**2**) im Zusammenhang mit dem Lernmodus konnte ich die einzelnen Positionen speichern. Mit der «Play Schaltfläche» (**3**) habe ich die ersten Programme getestet oder mit der «Stop Schaltfläche» (**4**) unterbrochen. Mit der Schaltfläche (**5**) habe ich das Programm abgespeichert oder vom Computer bzw. vom Roboter importiert. Mit der Schaltfläche «X» (**6**) habe ich die Programme gelöscht, die nicht funktioniert haben oder nicht mehr gebraucht wurden. Die Pfeile daneben ermöglichen einzelne Schritte vor oder zurück zu springen von den gelöschten Blöcken.

10.7 Die zur Auswahl stehenden Bausteine im Programm



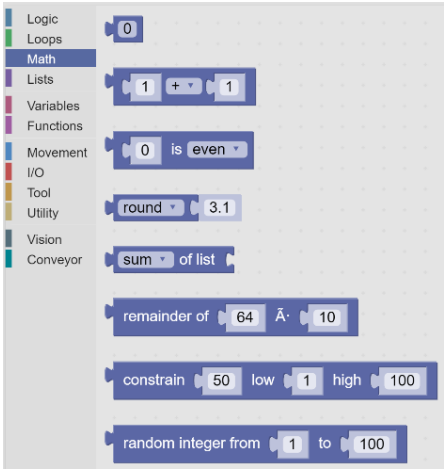
In der Rubrik «Logic» standen mir die linksstehenden Bausteine zur Verfügung. Die Logikblöcke ermöglichten mir entweder das Auslösen einer Aktion nach einer Bedingung oder Variablen je nach Anwendungszweck zu verändern.

Abbildung 25 Logic Bausteine



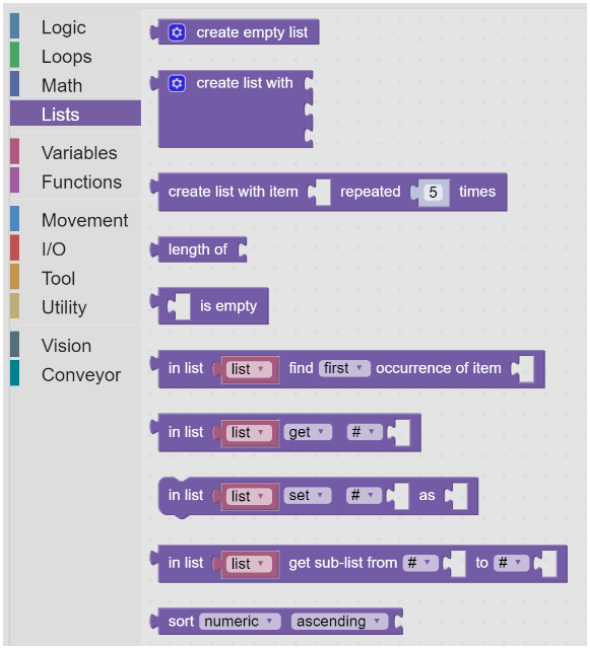
In der Rubrik «Loops» standen mir die linksstehenden Bausteine zur Verfügung. Die Schleifenblöcke ermöglichten mir eine Iteration von Blocken. Somit können Sequenzen oder Abläufen beliebig oft wiederholt werden.

Abbildung 26 Loop Bausteine



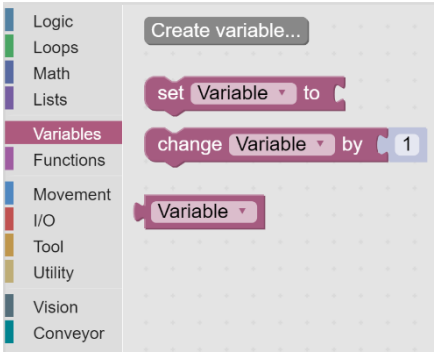
In der Rubrik «Math» standen mir die linksstehenden Bausteine zur Verfügung. Die mathematischen Blöcke ermöglichen mir verschiedenste mathematische Operationen zur Erfüllung meiner Anforderungen durchzuführen.

Abbildung 27 Math Bausteine



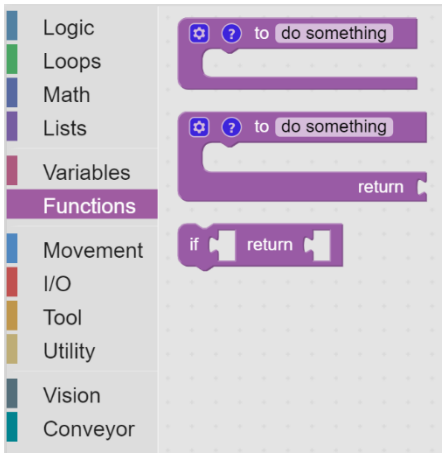
In der Rubrik «Lists» standen mir die linksstehenden Bausteine zur Verfügung. Die Listenblöcke ermöglichen mir Bausteine in Unterprogrammen (Listen) zu speichern, um danach darauf zurückzugreifen. Eine Liste ist eine Struktur, die es mir ermöglicht Daten in Gruppen zu speichern und abzurufen. Daher kann es auch als «Array» betrachtet werden. Ein «Array» ist ein Objekt, das mehrere Werte von ein und demselben Typ speichern kann.

Abbildung 28 Lists Bausteine



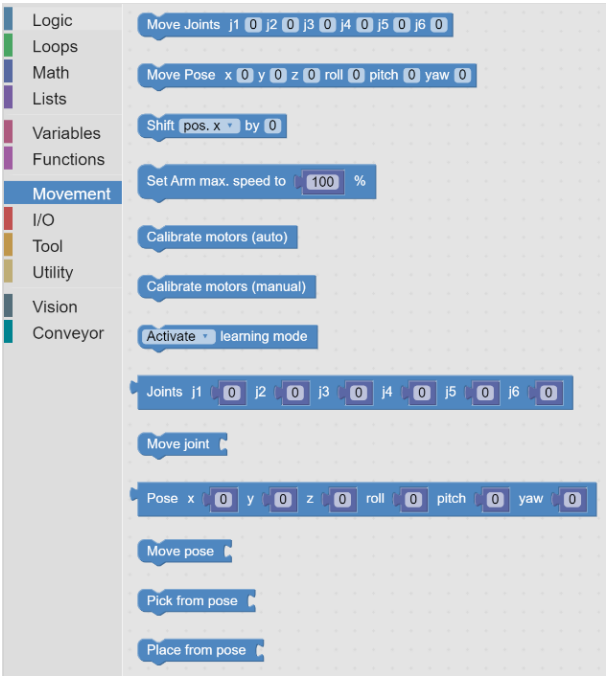
In der Rubrik «Variables» standen mir die linksstehenden Bausteine zur Verfügung. Einzelne Variablen zu erstellen, hat mir geholfen einen Platzhalter zu haben, dem ich einen Wert oder einen Namen zuweisen kann.

Abbildung 29 Variables Bausteine



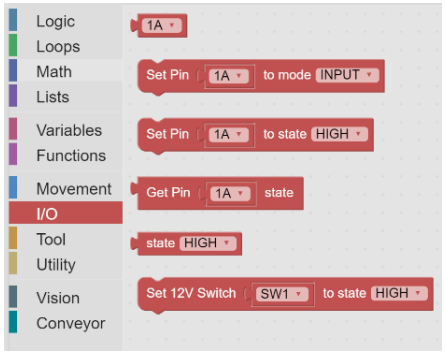
In der Rubrik «Funktions» standen mir die linksstehenden Bausteine zur Verfügung. Funktionsbausteine werden verwendet, um Funktionen zu erstellen oder aufzurufen. Funktionen umfassen einen Teil vom Code, der eine Aufgabe ausführen soll. Damit lassen sich auch sehr gut Unterprogramme realisieren, die später erneut aufgerufen werden können.

Abbildung 30 Funktions Bausteine



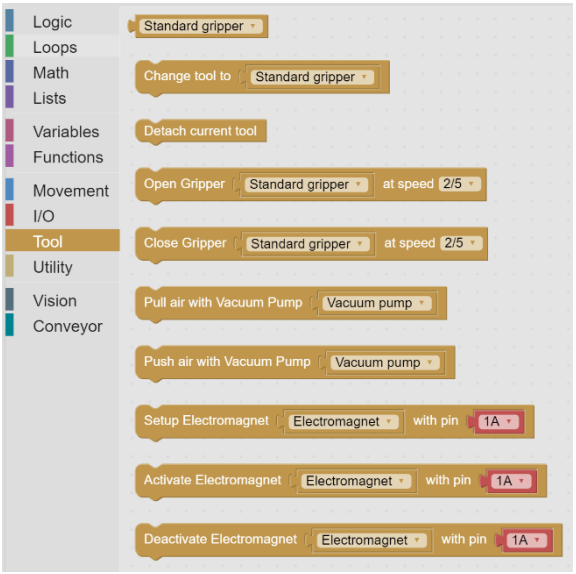
In der Rubrik «Movement» standen mir die linksstehenden Bausteine zur Verfügung. Mit den Bewegungsbausteinen können Positionen angefahren, Befehle (wie Greifen von einer Position) ausgeführt, Geschwindigkeit des Armes eingestellt, einzelne Achsen verschoben, Kalibrierung aufgerufen und in den Lernmodus gewechselt werden.

Abbildung 31 Movement Bausteine



In der Rubrik «I/O» standen mir die linksstehenden Bausteine zur Verfügung. Anhand der digitalen Ein- und Ausgangsbausteinen kann ich die Sensoren ansteuern oder deren Zustand ändern.

Abbildung 32 Digitalen Ein- und Ausgänge Bausteine



In der Rubrik «Tool» standen mir die linksstehenden Bausteine zur Verfügung. Anhand der Werkzeug Bausteine kann ich einzelne Werkzeuge ansteuern, einen Werkzeugwechsel anordnen, Anweisungen für Werkzeuge geben und den Greifer öffnen und schliessen.

Abbildung 33 Tool Bausteine

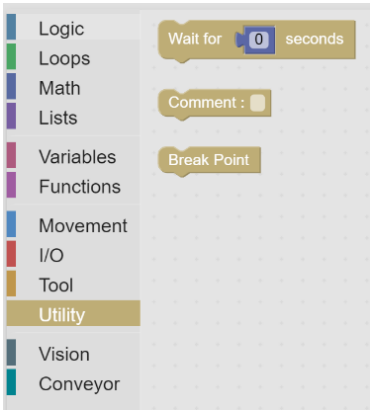


Abbildung 34 Utility Bausteine

In der Rubrik «Utility» standen mir die linksstehenden Bausteine zur Verfügung. Anhand der Hilfsbausteine kann ich dem Programm verschiedene Befehl geben, welche eine Pause oder einen Programmunterbruch auslösen. Es können aber auch Kommentare zur Beschreibung einzelner Bausteine eingefügt werden.

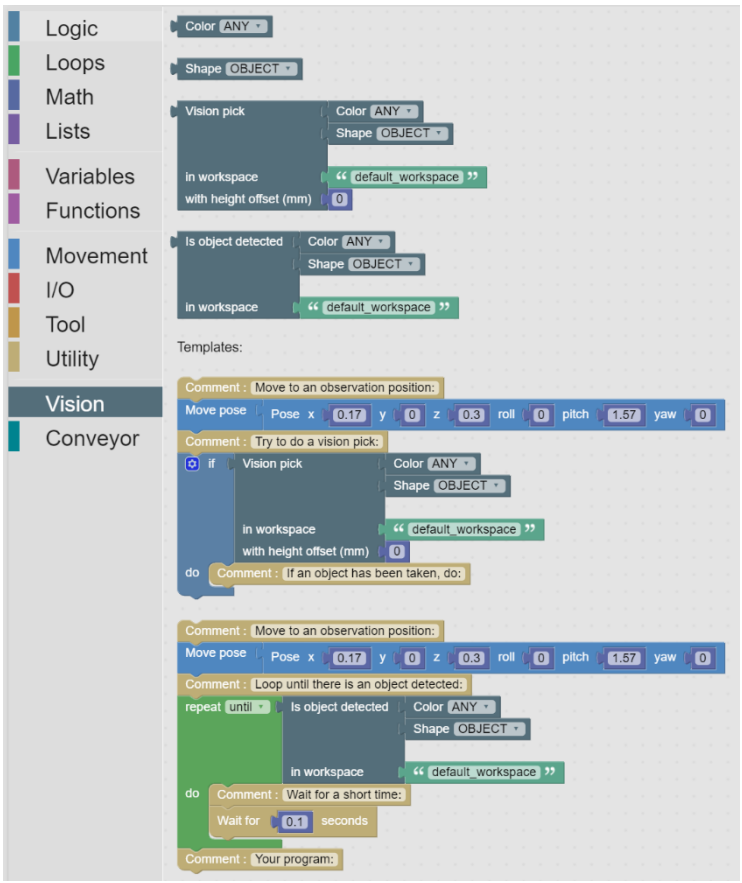


Abbildung 35 Vision Bausteine

In der Rubrik «Vision» standen mir die linksstehenden Bausteine zur Verfügung. Anhand der «Vision Bausteine» kann ich die Kamera einsetzen, um einen «Vision pick» zu machen. Das bedeutet, dass die Kamera die Position, Farbe und die Kontur erkennt und dann gemäss diesen Angaben das Objekt greift.

Ebenfalls kann abgefragt werden, ob ein Objekt erkannt wurde und demensprechend eine Aktion ausführen.

Für die «Vision Bausteine» gibt es zwei mögliche Vorlagen um zu sehen, was damit machbar ist.

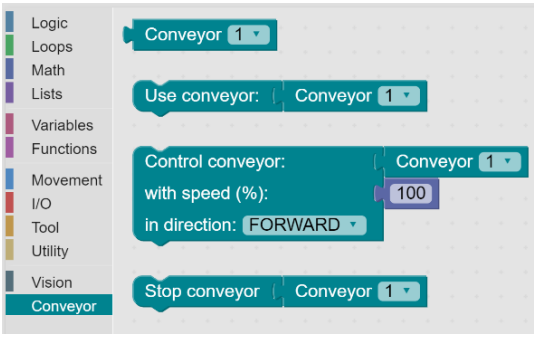


Abbildung 36 Conveyor Bausteine

In der Rubrik «Conveyor» standen mir die linksstehenden Bausteine zur Verfügung. Anhand der Förderbandbausteine kann ich auswählen, welches Förderband genutzt werden soll und bei welcher Geschwindigkeit und in welche Richtung es sich bewegt. Ebenfalls gibt es einen Baustein, um das Förderband zu stoppen.

10.8 Das fertige Programm in der Übersicht

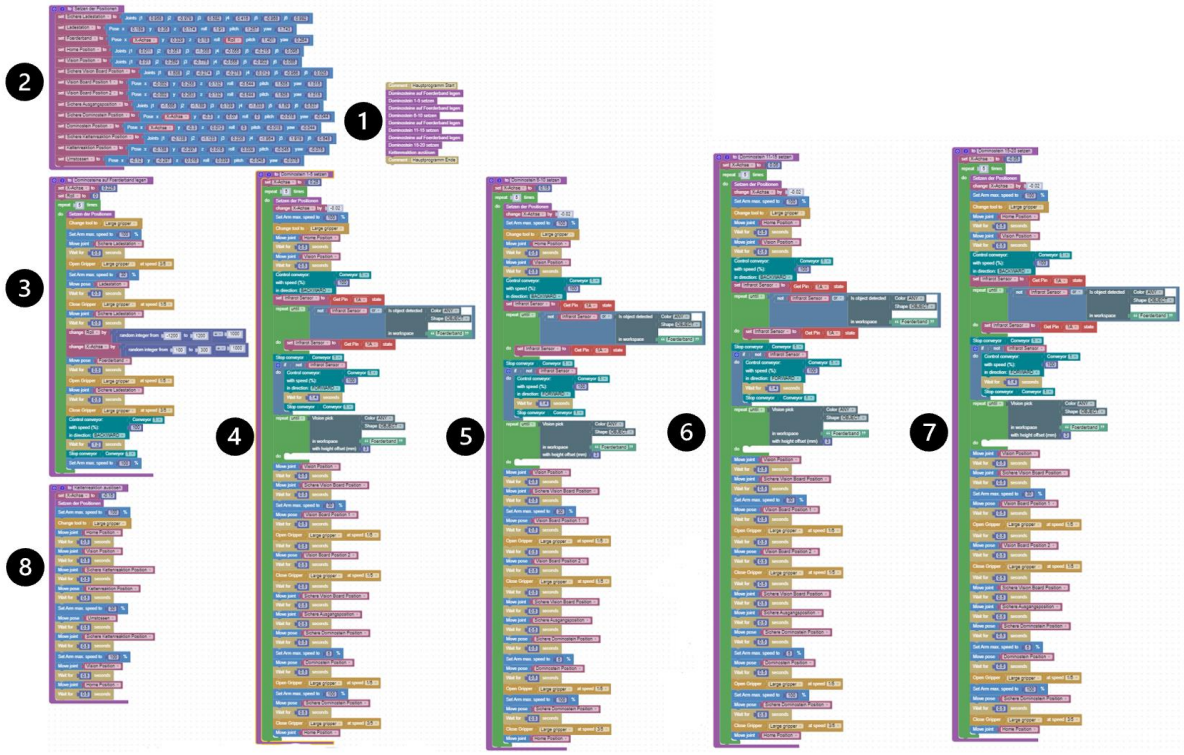


Abbildung 37 Das fertige Programm in der Übersicht

Im oberen Bild sieht man das fertige Programm. Die Übersicht zeigt, wie die Bausteine verwendet wurden, um die 20 Dominosteine zu legen und zum Schluss die Kettenreaktion auszulösen. Die genauen Unterprogramme werde ich in den nachfolgenden Kapiteln beschreiben.

(**1**) Ablauf des Hauptprogrammes (Aufruf der Unterprogramme)

(**2**) Die Positionen des Programms (sämtliche Positionen, die angefahren werden)

(**3**) Unterprogramm «Dominosteine auf das Förderband legen» (Unterprogramm, das die Dominosteine auf das Förderband legt)

(**4-7**) Unterprogramm «Dominosteine 1-20 setzen» (Legen der 20 Dominosteine in eine Reihe)

(**8**) Unterprogramm «Kettenreaktion auslösen» (auslösen der Kettenreaktion)

10.9 Der Ablauf des Hauptprogramms (1) erklärt mit Definitivem Aufbau

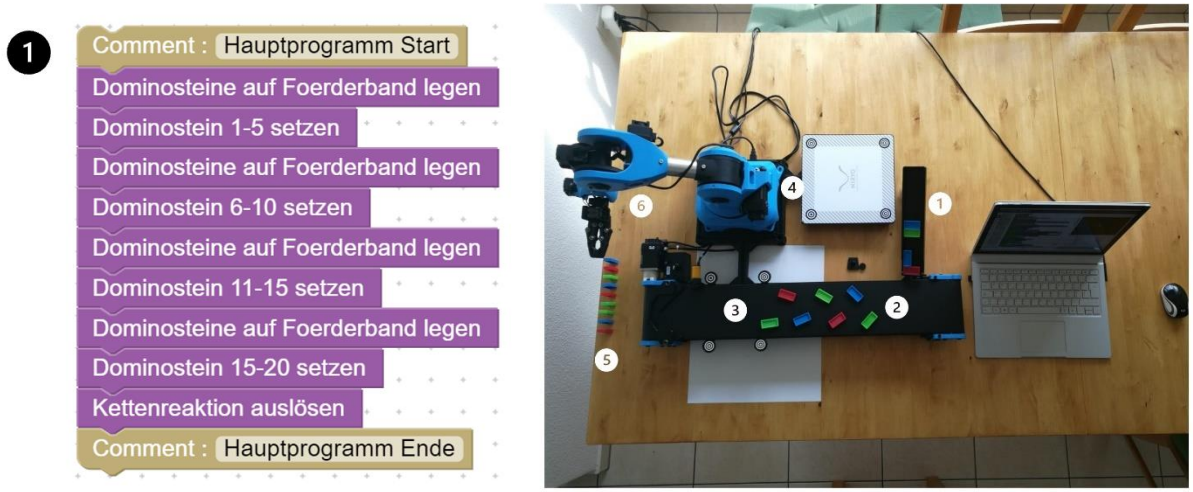


Abbildung 38 Hauptprogramm & Definitiver Aufbau

Im oberen Bild links, sieht man den Programmcode vom Hauptprogramm (**Schwarz 1**). Das Programm wird immer von Oben nach Unten abgearbeitet. Die rechte Seite des Bildes zeigt eine Momentaufnahme und nicht das Endergebnis. Es zeigt die Position, nachdem einige Steine auf dem Tisch abgesetzt wurden. Der erste Baustein im Hauptprogramm ist ein Kommentar, der sonst keine weitere Funktion hat. Er beschreibt, dass dort das Programm gestartet werden soll. Nachfolgend werden dann die einzelnen Unterprogramme, die ich als Funktionen zusammengefasst habe, abgehandelt. Es werden immer 5 Dominosteine von der Ladestation (**Weiss 1**) zufällig auf das Förderband (**Weiss 2**) gelegt und anschliessend zum Roboter befördert. Dort werden sie innerhalb des «Vision Workspace» (**Weiss 3**) erkannt und gegriffen. Dann wird der Stein auf das «Vision Board» (**Weiss 4**) gelegt und von dort auf den Tisch (**Weiss 5**) gestellt. Dieser Vorgang wiederholt sich, bis alle 20 Steine auf dem Tisch aufgestellt sind. Nachdem alle 20 Dominosteine gelegt wurden, wird die Kettenreaktion an der Position (**Weiss 6**) ausgelöst. Am Schluss des Hauptprogramms wird wieder mit einem Kommentar erwähnt, dass das Hauptprogramm beendet ist.

10.10 Die Positionen des Programms (2)

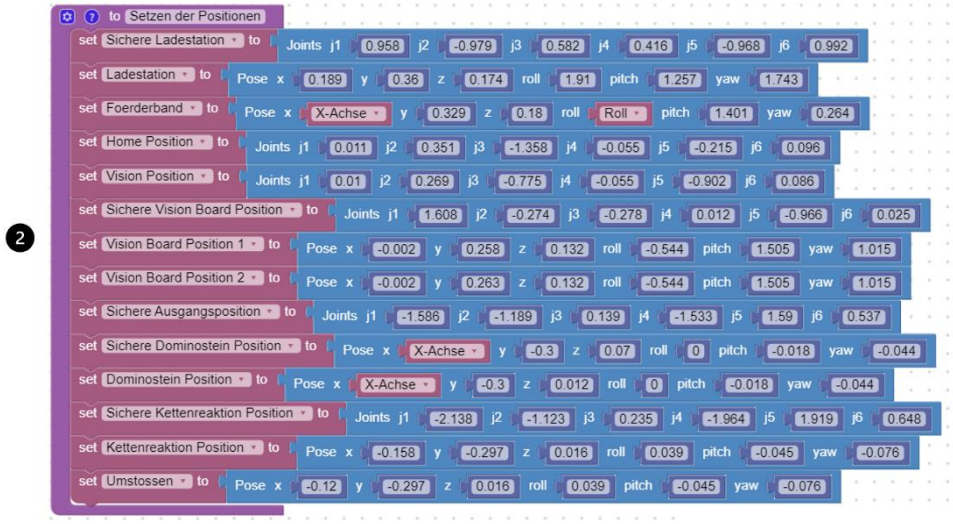


Abbildung 39 Positionen des Programms

Hier bei den Positionen (**2**) des Programms sieht man schon mehrere Bausteine. Die ganzen Positionen sind auch in einer Funktion zusammengefasst, um später in den Unterprogrammen wieder aufgerufen zu werden. Jede Position hat ihren eigenen Namen. Der Name wird als Variable deklariert und dann einer Position zugewiesen. Positionen können als «Pose» oder «Joints» gespeichert werden. «Joints» benutzt man, um anhand der Winkel jeder einzelnen Achse im «Niryo One» eine eindeutige Position anzufahren. Mit den «Pose» Bewegungen werden nur einzelne Achsen bewegt und nicht mehr alle 6 Achsen. Denn wenn man mit «Pose» Bewegungen mit mehreren Achsen macht, gibt es anhand der parabelähnlichen Bewegung der 6 Achsen, zwei mögliche Lösungen, die der Roboter dann eventuell willkürlich ausführt. Diese Überlegung wurde mir klar, als ich mehrere Stunden Bewegungen mit dem Roboter ausprobiert habe und nicht wusste, warum ich immer wieder zwei unterschiedliche Positionen hatte. Dank Andreas Holzer, meinem Robotik Dozenten an der TEKO Luzern, habe ich das auch gelernt. Ich hatte ihn im Unterricht darauf angesprochen und er erläuterte es mir dann genau. Daher verstand ich auch, warum man die «Pose»- und die «Joint» Bewegung der Achsen hat.

Die einzelnen Variablen wie z.B.: «X-Achse» oder «Roll», die man innerhalb der «Pose» Bewegung sieht, sind in den einzelnen Unterprogrammen definiert und werden dort erklärt.

10.11 Unterprogramm «Dominosteine auf das Förderband legen» (3)

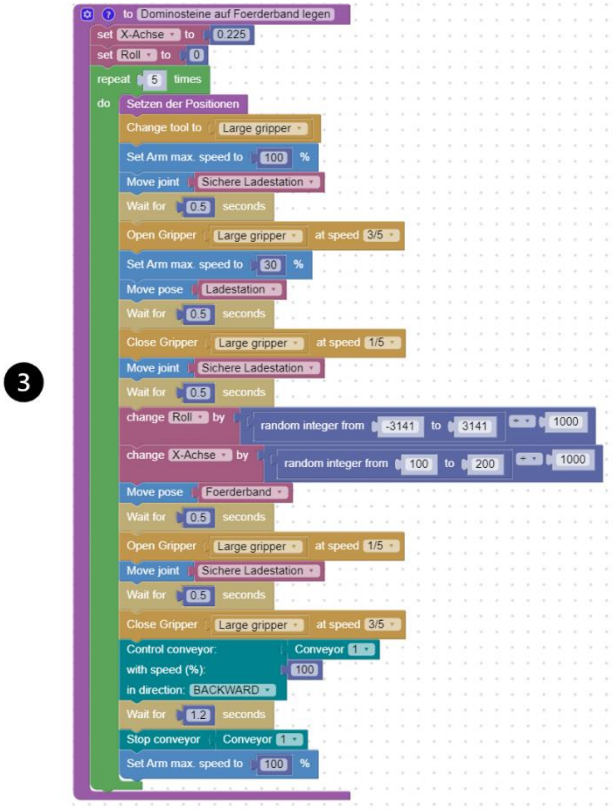


Abbildung 40 Unterprogramm «Dominosteine auf das Förderband legen»

Wie man erkennt, ist auch dieser Code in einer Funktion als Unterprogramm (**3**) untergebracht. Bevor wir in die Schleife gehen, werden die Variablen «X-Achse» und «Roll» definiert und für sie ein Wert festgelegt. Der Wert legt den Ausgangspunkt fest. Danach wird die Schleife 5-mal den folgenden Code wiederholen.

Als erstes in der Schleife wird die Funktion «Positionen» aufgerufen. Somit sind alle Variablen, mit denen im Unterprogramm «Positionen» verknüpft. Danach wird das Werkzeug definiert und die Geschwindigkeit des Roboterarms festgelegt. Anschliessend bewegt sich der Roboter zur «sicheren Ladestation» und wartet dort für 0.5 Sekunden. Dies ist wichtig, da der Roboterarm immer etwas nachschwingt und daher wird der Baustein «Wait for» nach jeder Position ausgeführt. Danach wird der Greifer geöffnet. Der Arm wird mit der Geschwindigkeit von 30% zur Ladestation bewegt. Dort greift er dann den Dominostein und bewegt sich wieder zur «sicheren Ladestation». Danach wird die Variable «X-Achse» und «Roll» mit einem mathematischen Baustein verändert. Diese gibt den Variablen eine zufällige Zahl zwischen den definierten Grenzen und teilt die Zahlen durch 1000. Dies wird gemacht, damit der Roboter die richtige Kommastelle erhält. Die Roll-Achse, die die Hand dreht, soll sich zwischen -π und +π und die X-Achse zwischen der Breite des Förderbands bewegen dürfen. Somit ist sichergestellt, dass der Dominostein willkürlich auf das Förderband gelegt wird, was auch der nächste Code im Ablauf ist. Dort angekommen wird der Greifer geöffnet und wieder zur «sicheren Ladestation» Position bewegt. Anschliessend wird der Greifer wieder geschlossen. Der nächste Baustein ist ein Förderbandbaustein, bei dem das Förderband 1 mit 100% Geschwindigkeit für 1.2 Sekunden rückwärts bewegt wird. Danach stoppt das Förderband wieder. Dies gibt den benötigten Abstand zum nächsten Stein, der auf das Förderband gelegt werden soll. Zum Schluss der Schleife wird die Armgeschwindigkeit wieder auf 100% gesetzt.

10.12 Unterprogramm «Dominosteine 1-20 setzen» (4-7)

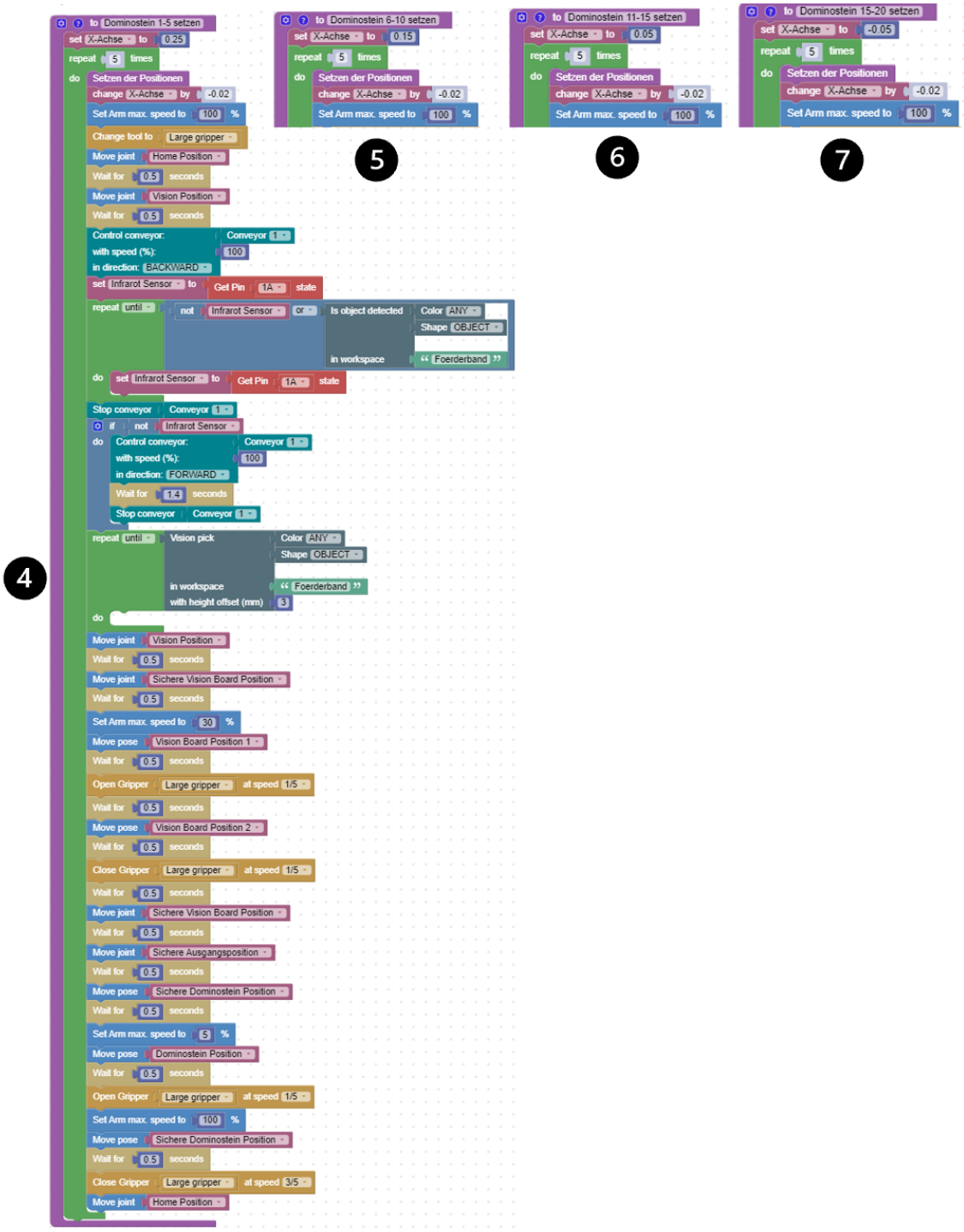


Abbildung 41 Unterprogramm Dominosteine 1-20 setzen

Wie man auch hier erkennt, ist auch dieser Code in einer Funktion als Unterprogramm (**4-7**) untergebracht. Die Unterprogramme (**5**), (**6**) und (**7**) unterscheiden sich nur von der Definition der «X-Achse» und dem Namen der Funktion. Bevor wir jedoch in die Schleife gehen, wird die Variablen «X-Achse» definiert und ihr Wert festgelegt. Dieser legt den Ausgangspunkt fest. Danach wird die Schleife 5-mal mit dem folgenden Code wiederholt.

In der Schleife werden wieder alle definierten Positionen eingelesen. Danach wird jedes Mal, wenn die Schleife durchlaufen wird, wird die Variable «X-Achse» um den Wert von -0.02 Meter verschoben. Dies ist notwendig, damit die Dominosteine beim Aufstellen einen Abstand von 2cm haben und nicht aufeinandergestapelt werden. Die Armgeschwindigkeit und der Greifer werden definiert. Der Roboter startet an der «Home Position» und bewegt sich von dort zur «Vision Position». Dann wird das Förderband gestartet und die Geschwindigkeit und Richtung definiert. Anschliessend wird die Variable «Infrarot Sensor» auf den digitalen Eingang 1A gesetzt. Dieser ist an der Roboterrückseite eingesteckt. Danach wird eine weitere Schleife mit mehreren Bausteinen aufgerufen.

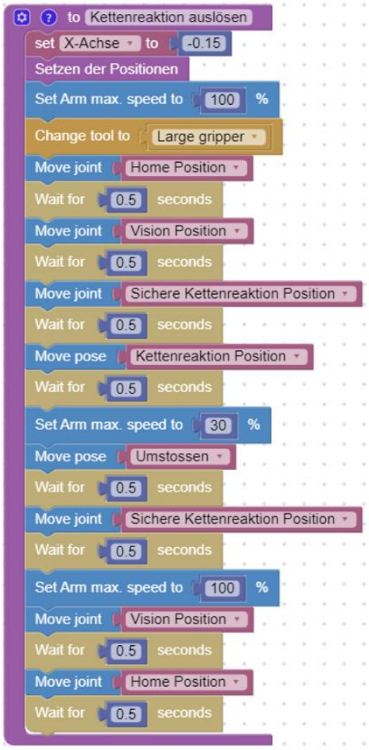
Die Schleife wird so lange wiederholt, bis der Infrarotsensor nicht mehr aktiv ist oder irgendein Objekt in irgendeiner Farbe auf dem «Vision Workspace» mit dem Namen «Foerderband» auftaucht. Wenn dies eintritt, wird der Infrarotsensor erneut auf den digitalen Eingang 1A gesetzt. Nach der Schleife wird das Förderband gestoppt.

Nach dem das Förderband gestoppt hat, wird ein Logikbaustein verwendet vom Typ «If/ Do». Wenn der Infrarotsensor nicht aktiv ist, wird das Förderband für 1.4 Sekunden zurückbewegt und danach gestoppt. Dies dient der Sicherheit, falls ein Dominostein nicht durch den Baustein «Is object detectet» erkannt wurde. Der Infrarotsensor gibt das Signal, dass der Dominostein nicht mehr im Vision Bereich ist. Dadurch wird der Stein auf dem Förderband zurück in den «Vision Workspace» bewegt. Anschliessend wird erneut eine Wiederholschleife benutzt, bis wieder ein Stein erkannt wird.

In dieser Schleife wird so lange versucht ein «Vision Pick» zu machen, bis einer ausgeführt wurde. Dafür wird ein Objekt von irgendeiner Farbe auf dem «Workspace Förderband» benötigt. Der Greifer greift den Stein mit einem Offset von 3mm Höhe. Dieser Offset ist nötig, damit die Steine korrekt durch den Roboter gegriffen werden können.

Nach der Schleife wird der Roboter wieder in die «Vision Position» gebracht und anschliessend zur sicheren «Vision Board Position». Dort angekommen wird wieder die Geschwindigkeit angepasst und ein Stein auf das «Vision Board» gelegt. Dort wird der Stein abgelegt, um ihn weiter Oben zu greifen, da beim «Vision Pick» der Greifer den Stein oft ganz im Eck greift. Zum Aufstellen ist es aber besser, wenn der Stein mittig gegriffen wird, da sonst der Dominostein schräg abgelegt werden kann und somit schon vorzeitig die Kettenreaktion auslöst. Nachdem der Stein nun auf dem «Vision Board» abgelegt wurde, greift ihn der Greifer nun mittig und bewegt sich zurück zur sicheren «Vision Board Position». Von dort bewegt sich der Roboter zur «sicheren Ausgangsposition» oberhalb des Tisches, wo der Stein später gelegt werden soll. Danach wird die «sichere Dominostein Position» angefahren. Anschliessen wird die Geschwindigkeit noch weiter verlangsamt, damit eine möglichst genaue Positionierung des Steins möglich ist. Nach dem dies geschehen ist, wird die «Dominostein Position» angefahren, der Stein platziert und der Arm wieder mit voller Geschwindigkeit zur «Home Position» bewegt. Danach wiederholt sich die Schleife wie bereits beschrieben.

10.13 Unterprogramm «Kettenreaktion auslösen» (8)

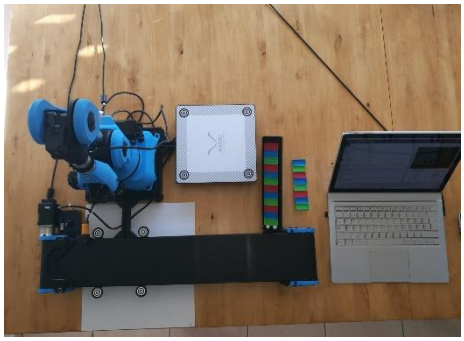


8

Wie man auch hier erkennt, ist auch dieser Code in einer Funktion als Unterprogramm «Kettenreaktion auslösen» (**8**) untergebracht. Diese Funktion ist aber viel einfacher als die Vorherige. Wie aber auch in den anderen Funktionen wird hier zuerst die Variable «X-Achse» als Ausgangspunkt definiert. Anschliessend werden nochmal die Positionen eingelesen. Danach wir die Armgeschwindigkeit und der Greifer definiert. Nachdem dies ausgeführt wurde, wird der Roboter von der «Home Position» zur «Vision Position» bewegt. Von da aus wird zur «Sicheren Kettenreaktionsposition» übergegangen, gefolgt von der «Kettenreaktion Position», die ganz am Ende der Dominosteinaufstellung ist. Dort wird dann die Geschwindigkeit wieder angepasst und der letzte Dominostein angestossen, wobei die Kettenreaktion ausgelöst wird. Anschliessend wird der Roboter wieder zur «Sicheren Kettenreaktionsposition» bewegt und die Geschwindigkeit wieder erhöht. Wenn dies erledigt ist, wird der Roboter zur «Vision»- und dann zur «Home Position» bewegt. Dies ist auch die letzte Bewegung des Roboters. Somit ist das Programm zu Ende und das Ziel der Programmierung erreicht.

Abbildung 42 Unterprogramm Kettenreaktion auslösen

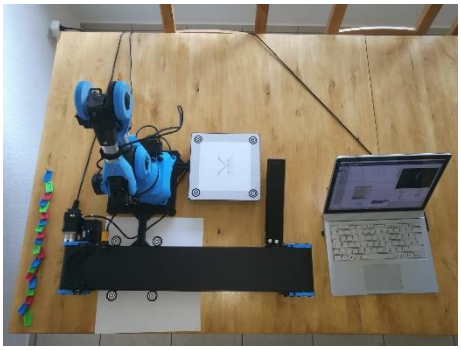
10.14 Ist Aufbau beim Start des Programms



Wie man dem Foto entnehmen kann, können nicht alle Dominosteine in der Ladestation gesammelt werden. Deswegen habe ich die restlichen Dominosteine während des Programms von Hand nachgelegt, damit es keinen Unterbruch im Programm gibt.

Abbildung 43 Ist Aufbau beim Start des Programms

10.15 Ist Aufbau beim Ende des Programms



Hier auf dem Foto sieht man den Aufbau und die Dominosteine, nachdem die Kettenreaktion ausgelöst wurde.

Abbildung 44 Ist Aufbau beim Ende des Programms

11. Auswertung der Ziele & Anforderungsliste

11.1 Auswertung der Ziele

- Zielgerechtes Legen der Steine gemäss Vorgabe und Ablauf
  - ➢ Das Zielgerechte legen der Dominosteine gemäss Vorgabe und Ablauf wurde erfüllt, gemäss dem Programm und dem entsprechenden Endresultat.
- Keine Fehler in der Programmstruktur, die den Ablauf unterbrechen
  - ➢ Dieses Ziel wurde nur teilweise erfüllt, da der Roboter manchmal willkürlich Bewegungen macht, die nicht von aussen gesteuert wurden, die einen Abbruch verursachen können. Die Lösung des Problems wurde leider nicht gefunden. (Wahrscheinlich ist es ein Mechanisches Problem)
- Alle Anforderungen gemäss den geforderten Kriterien umsetzen
  - ➢ Dieses Ziel wurde leider nicht erreicht, aufgrund von Ungenauigkeiten oder nicht idealem Material.
- Dominosteinkettenreaktion funktioniert einwandfrei
  - ➢ Auch dieses Ziel wurde nur teilweise erfüllt aufgrund schon oben genannter Schwierigkeiten die auftreten können.
- Eine effiziente Wahl bezüglich des Programms treffen
  - ➢ Dieses Ziel wurde erreicht, wobei man allerdings zugeben muss, dass es immer noch Potenzial hat am Programm weiterzuarbeiten.
- Dem Terminplan gefolgt
  - ➢ Dieses Ziel konnte ich aufgrund mehrerer Schwierigkeiten, die mir viel Zeit gekostet haben, nicht ganz erfüllen. Einige Beispiele werden weiter unten im Kapitel 11.4 erklärt. Für die Arbeit wurden 200 Stunden geplant, jedoch effektiv 250 Stunden dafür aufgewendet.
- Nachvollziehbare Auswahlen treffen
  - ➢ Dieses Ziel konnte ich erfüllen und eine gute Auswahl treffen und habe dies auch im Kapitel 9.7 begründet.
- Selbständiges Ausarbeiten des Programms
  - ➢ Dieses Ziel konnte ich erfüllen. Bis auf die eine oder andere Frage von mir, auf unbekannten Alarmmeldungen oder Probleme mit den Bewegungsbausteinen, die mir mein Dozent Andreas Holzer erklärte. Das ganze Programm wurde aber zu 100% von mir geschrieben.
- Korrekte Umsetzung des Projektmanagements gemäss Erlerntem an der TEKO Luzern
  - ➢ Ich konnte mein erlerntes Wissen immer wieder anwenden, muss aber auch zugeben, dass ich viel mehr dazugelernt habe, als dass was ich angewendet habe.
- Den Roboter so gut kennen, um jegliche Probleme lösen zu können
  - ➢ Zum Schluss kann ich sagen, dass ich dieses Ziel erreicht habe. Es werden aber immer noch Probleme auftreten, die mir unbekannt sind, wenn ich weiter mit dem Roboter arbeiten würde, aber die Probleme die ich bisher hatte konnte ich auch lösen.
- Erfahrung in der Robotik sammeln
  - ➢ Dieses Ziel konnte ich sicherlich erfüllen. Ich habe so viel über die einzelnen Bewegungen und Hintergründe des Programmierens gelernt, dass ich dies alles nicht in dieser Arbeit erklären könnte.

11.2 Auswertung der Anforderungsliste

Anforderungen	F	M	W	Verantwortlich
Geometrie				
Die Dominosteine innerhalb von 440 mm Radius legen und +/- 175 °	F			Programmierer ✓
Maximaler Weg mit Förderband 700 mm	F			Hersteller ✓
Dominosteine sollten für alle Greifer greifbar sein			W	Programmierer ✗
Der Komplette Aufbau sollte auf einem TEKO Schreibtisch Platz haben			W	Programmierer ✓
Kinematik				
Maximale Geschwindigkeit des Förderbands beträgt 38 mm/s	F			Hersteller ✓
Genauigkeit des «Niryo One» sollte +/- 1mm sein	F			Hersteller ✗
Kräfte				
Maximale Nutzlast des «Niryo One» beträgt 0,3 kg	F			Hersteller ✓
Maximales zulässiges Gewicht auf dem Förderband beträgt 2 kg	F			Hersteller ✓
Gewicht des «Niryo One» 3.2 kg	F			Hersteller ✓
Energie				
Die Kamera des «Vision Sets» soll direkt am «Niryo One» über die USB-Schnittstelle angeschlossen werden		M		Hersteller ✓
Stromversorgung für den «Niryo One» und das Förderband soll 11.1 Volt / 6A betragen	F			Hersteller ✓
Maximaler Stromverbrauch des «Niryo One» sollte 60 W betragen	F			Hersteller ✓
Stoff				
Roboter - Aluminium, PLA (3D Druck)	F			Hersteller ✓
Dominosteine (Kunststoff)			W	Programmierer ✓
Signal				
IR (Infrarotsensor) Distanzerkennung 6 – 80 cm			W	Montage ✓
Kommunikation: ➢ Ethernet ➢ WIFI: 2.4 GHz Range 802.11n ➢ Bluetooth 4.1: 2,4 GHz ; 2,5 mW (4 dBm) ➢ USB	F			Hersteller ✓

Sicherheit				
Kollisionssensor über einen Magnetsensor im Motor	F			Hersteller
Netzstecker/Schutzstecker nach SEV	F			Hersteller
Endanschlag			W	Programmierer
Ergonomie				
Die Bedienung des «Niryo One» sollte per Knopfdruck möglich sein			W	Programmierer
Kontrollen/Wartung				
Kalibrierung des «Niryo One» Roboters	F			Programmierer
Montage				
Der Roboter sollte vormontiert sein	F			Hersteller
Transport				
Der ganze Aufbau sollte individuell von einem Ort zum anderen transportierbar sein		M		Programmierer
Gebrauch				
Lange Positionsgenauigkeit und Wiederholungs-rate		M		Programmierer
Einfache Bedienung		M		Programmierer
Kosten				
Das Material sollte von der TEKO zur Verfügung gestellt werden und somit keine Kosten gene-rieren		M		Programmierer
Umfang				
Es sollen 20 Dominosteine verwendet werden			W	Programmierer
Termin				
Abgabetermin der Dokumentation und des Pro-gramms (11.10.21)	F			Projektleiter
Präsentation des Programms/Ablaufes (28.10.21)	F			Projektleiter

Tabelle 5 Auswertung der Anforderungsliste

11.3 Weiteres Vorgehen

Weiterführende Arbeiten die möglich wären:

- Weitere Optimierung des Programmcodes, um ein schnelleres Programm zum laufen zu bringen.
- Weitere Funktionen einbauen, wie z.B. die Dominosteine in einer Kurve zu legen oder vom Vision Board runter gelegte Dominosteine.
- Das ganze Programm auf dem neuen Roboterarm «Niryo Ned» anpassen, bei dem die meisten Probleme des «Niryo One» ausgemerzt sind und den die TEKO nun während meines Projekts zusätzlich dazugekauft hat.
- Das ganze Programm erweitern mit dem zweiten Roboter oder einem weiteren Förderband.
- Die Probleme mit der Schnittstelle des CAN Buses zwischen Förderband und Roboter ausmerzen.

11.4 Zu erwähnende Schwierigkeiten

Ich habe dieses Kapitel extra noch angefügt, um einige meiner Schwierigkeiten aufzuführen, die aufgetreten sind, damit verstanden wird, was die Schwachheiten des Roboters sind und wobei ich Zeit verloren habe.

- Das Warten auf Bauteile, die gefehlt haben, die eigentlich vorhanden sein sollten, die aber erst später aufgefallen sind. Das betrifft das Bauteil zum Befestigen der Kamera auf dem Roboterkopf.
- Die Nutzung des Roboters von weiteren Personen. Ich musste den Roboter jede Woche Sonntag oder früher zur Schule bringen, damit der Roboter vom Dozenten auch im Unterricht gebraucht werden konnte. Dieser war dann jeweils am Montag und Dienstag von ihm im Gebrauch. Nebenbei musste der Dozent auch Vorbereitungen mit dem Roboter machen, die meine Zeit am Roboter einschränkten.
- Die schlechte Verarbeitung des Roboters. Ich durfte immer wieder von Zeit zu Zeit, wenn der Roboter ungenau war und nicht mehr die Toleranz von +-1 mm einhalten konnte, den Roboter auseinanderbauen und einige Schrauben nachziehen. Diese Arbeiten haben mir auch viel Zeit und Nerven gekostet, da dies fast jede Woche nötig war. Dieses Problem lag daran, dass die Schrauben teilweise direkt in den Kunststoff geschraubt waren. Die Verarbeitung des 3D Druckes bei der Herstellung des Roboters war unzureichend. Durch geringste Vibrationen oder den Transport des Roboters lösten sich die Schrauben manchmal selbstständig. Auch wurde beim Roboter keine gute Lösung gefunden, wie das Kamerakabel zum USB-Anschluss gelangt. Ich habe mit Kabelbindern das Kabel so befestig, dass es für mein Programm keine Probleme geben sollte.
- Auch im Laufe des Programmierens ist mir immer wieder aufgefallen, dass die Kalibrierung auch nicht immer genau gleich ist. Was dort das Problem war, habe ich leider nicht feststellen können, aber es könnte auch an der schlechten Verarbeitung gelegen haben. Dadurch musste ich immer wieder die Werte der Achsen etwas anpassen, bis es stimmte.
- Das Problem mit dem CAN Bus und der Kommunikation zwischen Förderband und Roboter. Nach einer gewissen Zeit bringt der «Alarm log» den Fehler, dass es Probleme mit dem Robotermotor 6 gibt. Dieser Motor steuert das Förderband. Um diesen Alarm zu löschen, muss ich das Kabel einfach aus und wieder einstecken. Auch macht dieser Alarm Probleme mit der Kalibrierung, die nicht funktioniert, wenn das Förderband eingesteckt ist. Diese Lösungen habe ich durch viel ausprobieren herausgefunden.

11.5 Lessons Learned

Die Durchführung der Diplomarbeit hatte viel Geduld und Ausdauer gebraucht, um zu diesem Resultat zu gelangen. Schon vor dem Start der Diplomarbeit hatte ich sehr viel Respekt vor der auf mich zu kommender Zeit. Ich wusste nicht genau, was mich erwarten würde, da ich noch nie etwas Vergleichbares gemacht hatte. Deswegen war ich nervös aber zugleich auch begeistert zu Starten. Mir wurde während des Projekts klar, wie wichtig die Vorbereitungszeit war, mit der Infosammlung und den ersten Versuchen mit dem Roboter. Dadurch konnte ich immer mehr und mehr sehen, was auf mich zukommt und dementsprechend reagieren. Das erworbene Wissen der Robotik und der Programmierung in Blockly zeigte mir auf, wie komplex und doch so spannend dieser Bereich ist, da man zum Schluss auch ein sichtbares Resultat hat. Eine weitere Erkenntnis, die ich gefunden habe, war dass ich beim Auftreten von Problemen frühzeitig mit den Verantwortlichen sprechen sollte, um schnellstmöglich eine Lösung zu finden oder nötige Anpassungen machen zu können. Trotz all den Herausforderungen hat mir die Arbeit sehr viel Spass gemacht und ich hoffe, dass ich in Zukunft mehr in diese Richtung arbeiten kann. Ich bin mir auch sicher, dass diese Arbeit gezeigt hat, was alles möglich mit solch einem Roboter und dessen Zubehör ist.

11.6 Schlusswort & Danksagung

Zum Ende meiner Diplomarbeit «Dominosteine legen mit dem Niryo One» möchte ich sagen, dass ich stolz auf die geleistete Arbeit bin, die mir sehr viel Freude bereitet hat. Auch möchte ich mich bei allen Personen, die mich unterstützt haben, namentlich bedanken.

- Rudolf Gautschi für die Begleitung des Projekts
- Andreas Holzer für die guten Inputs und Erklärungen
- Ivo Wittwer für die Bereitstellung des Roboters und das Bestellen fehlender Teile
- Jacqueline Roth für das Korrigieren der Arbeit
- Familie und Freunde für das Verständnis und die mentale Unterstützung

12. Anhang

12.1 Literaturverzeichnis

- **Grundlagen der Robotik**
  Helmut Maier
  2. überarbeitete und erweiterte Auflage, VDE Verlag
- **TEKO Lehrmittel**
  Skripte und Dokumente vom Dozenten

12.2 Software

- **«Niryo One» Studio**, Niryo
- **Microsoft Office 365**, Word, Excel, Outlook
- **TI-Nspire-CX**, Taschenrechner
- **Snipping Tool**, Microsoft

12.3 Quellenangabe

Alle angegebenen Quellen wurden über die Suchmaschine www.google.ch im Zeitraum zwischen KW 33-41 gesucht.

- **Roboter «Niryo One»**
https://niryo.com/product/niryo-one/
- **Förderband**
https://niryo.com/product/conveyor-belt/
- **«Vision Set»**
https://niryo.com/product/vision-set/
- **Zubehör (Greifer)**
https://niryo.com/niryo-one-accessories/
- **Unterlagen & Programme**
https://niryo.com/download/
- **Technische Spezifikation**
https://pdf.directindustry.com/pdf/niryo/niryo-one-mechanical-specifications/209911-833635.html
- **Aufbau des «Niryo One»**
https://niryo.com/docs/niryo-one/assembly-guide/assemble-niryo-one/
- **Handbuch «Niryo One»**
https://www.generationrobots.com/media/niryo-one-user-manual-03-09-2019.pdf
- **Broschüre Blockly**
https://www.generationrobots.com/media/niryo/Blockly_1_EN_C.pdf
- **Broschüre «Vision Set»**
https://www.generationrobots.com/media/niryo/vision-set-user-manual-062020
- **Broschüre Förderband**
https://www.generationrobots.com/media/niryo/conveyor-belt-user-manual-062020

12.4 Abbildungsverzeichnis

12.5 Tabellenverzeichnis

12.6 Python Code meines Programmes «Dominosteine legen mit dem «Niryo One»»

#!/usr/bin/env python from niryo_one_python_api.niryo_one_api import * import rospy rospy.init_node('niryo_one_run_python_api_code') n = NiryoOne() from numbers import Number import random Sichere_Ladestation = None X_Achse = None Ladestation = None Roll = None Foerderband = None Home_Position = None Vision_Position = None Sichere_Vision_Board_Position = None Vision_Board_Position_1 = None Vision_Board_Position_2 = None Sichere_Ausgangsposition = None Sichere_Dominostein_Position = None Sichere_Kettenreaktion_Position = None Dominostein_Position = None Infrarot_Sensor = None Kettenreaktion_Position = None Umstossen = None """Describe this function... """ def Setzen_der_Positionen(): global n, Sichere_Ladestation, X_Achse, Ladestation, Roll, Foerderband, Home_Position, Vision_Position, Sichere_Vision_Board_Position, Vision_Board_Position_1, Vision_Board_Position_2, Sichere_Ausgangsposition, Sichere_Dominostein_Position, Sichere_Kettenreaktion_Position, Dominostein_Position, Infrarot_Sensor, Kettenreaktion_Position, Umstossen Sichere_Ladestation = [0.958, -0.979, 0.582, 0.416, -0.968, 0.992] Ladestation = [0.189, 0.36, 0.174, 1.91, 1.257, 1.743] Foerderband = [X_Achse, 0.329, 0.18, Roll, 1.401, 0.264] Home_Position = [0.011, 0.351, -1.358, -0.055, -0.215, 0.096] Vision_Position = [0.01, 0.269, -0.775, -0.055, -0.902, 0.086] Sichere_Vision_Board_Position = [1.608, -0.274, -0.278, 0.012, -0.966, 0.025] Vision_Board_Position_1 = [-0.002, 0.258, 0.132, -0.544, 1.505, 1.015] Vision_Board_Position_2 = [-0.002, 0.263, 0.132, -0.544, 1.505, 1.015] Sichere_Ausgangsposition = [-1.586, -1.189, 0.139, -1.533, 1.59, 0.537] Sichere_Dominostein_Position = [X_Achse, -0.3, 0.07, 0, -0.018, -0.044] Dominostein_Position = [X_Achse, -0.3, 0.012, 0, -0.018, -0.044] Sichere_Kettenreaktion_Position = [-2.138, -1.123, 0.235, -1.964, 1.919, 0.648] Kettenreaktion_Position = [-0.158, -0.297, 0.016, 0.039, -0.045, -0.076] Umstossen = [-0.12, -0.297, 0.016, 0.039, -0.045, -0.076] """Describe this function... """ def Dominosteine_auf_Foerderband_legen(): global n, Sichere_Ladestation, X_Achse, Ladestation, Roll, Foerderband, Home_Position, Vision_Position, Sichere_Vision_Board_Position, Vision_Board_Position_1, Vision_Board_Position_2, Sichere_Ausgangsposition, Sichere_Dominostein_Position, Sichere_Kettenreaktion_Position, Dominostein_Position, Infrarot_Sensor, Kettenreaktion_Position, Umstossen X_Achse = 0.225 Roll = 0 for count in range(5): Setzen_der_Positionen() n.change_tool(TOOL_GRIPPER_2_ID) n.set_arm_max_velocity(100) n.move_joints(Sichere_Ladestation) n.wait(0.5) n.open_gripper(TOOL_GRIPPER_2_ID, 500) n.set_arm_max_velocity(30) n.move_pose(*Ladestation) n.wait(0.5) n.close_gripper(TOOL_GRIPPER_2_ID, 100) n.move_joints(Sichere_Ladestation) n.wait(0.5) Roll = (Roll if isinstance(Roll, Number) else 0) + random.randint(-1200, 1200) / 1000 X_Achse = (X_Achse if isinstance(X_Achse, Number) else 0) + random.randint(100, 300) / 1000 n.move_pose(*Foerderband) n.wait(0.5) n.open_gripper(TOOL_GRIPPER_2_ID, 100) n.move_joints(Sichere_Ladestation) n.wait(0.5) n.close_gripper(TOOL_GRIPPER_2_ID, 500) n.control_conveyor(6, True, 100, -1) n.wait(1.2) n.control_conveyor(6, False, 0, 1) n.set_arm_max_velocity(100) """Describe this function... """ def Dominostein_1_5_setzen(): global n, Sichere_Ladestation, X_Achse, Ladestation, Roll, Foerderband, Home_Position, Vision_Position, Sichere_Vision_Board_Position, Vision_Board_Position_1, Vision_Board_Position_2, Sichere_Ausgangsposition, Sichere_Dominostein_Position, Sichere_Kettenreaktion_Position, Dominostein_Position, Infrarot_Sensor, Kettenreaktion_Position, Umstossen X_Achse = 0.25 for count2 in range(5): Setzen_der_Positionen() X_Achse = (X_Achse if isinstance(X_Achse, Number) else 0) + -0.02 n.set_arm_max_velocity(100) n.change_tool(TOOL_GRIPPER_2_ID) n.move_joints(Home_Position) n.wait(0.5) n.move_joints(Vision_Position) n.wait(0.5) n.control_conveyor(6, True, 100, -1) Infrarot_Sensor = n.digital_read(GPIO_1A) while not (not Infrarot_Sensor or (n.detect_object(n.get_workspace_ratio('Foerderband'), "ANY", "ANY")[0])): Infrarot_Sensor = n.digital_read(GPIO_1A) n.control_conveyor(6, False, 0, 1) if not Infrarot_Sensor: n.control_conveyor(6, True, 100, 1) n.wait(1.4)

n.control_conveyor(6, False, 0, 1) while not (n.pick('Foerderband', float(3)/100, "ANY", "ANY")[0]): pass n.move_joints(Vision_Position) n.wait(0.5) n.move_joints(Sichere_Vision_Board_Position) n.wait(0.5) n.set_arm_max_velocity(30) n.move_pose(*Vision_Board_Position_1) n.wait(0.5) n.open_gripper(TOOL_GRIPPER_2_ID, 100) n.wait(0.5) n.move_pose(*Vision_Board_Position_2) n.wait(0.5) n.close_gripper(TOOL_GRIPPER_2_ID, 100) n.wait(0.5) n.move_joints(Sichere_Vision_Board_Position) n.wait(0.5) n.move_joints(Sichere_Ausgangsposition) n.wait(0.5) n.move_pose(*Sichere_Dominostein_Position) n.wait(0.5) n.set_arm_max_velocity(5) n.move_pose(*Dominostein_Position) n.wait(0.5) n.open_gripper(TOOL_GRIPPER_2_ID, 100) n.set_arm_max_velocity(100) n.move_pose(*Sichere_Dominostein_Position) n.wait(0.5) n.close_gripper(TOOL_GRIPPER_2_ID, 500) n.move_joints(Home_Position) """Describe this function... """ def Dominostein_6_10_setzen(): global n, Sichere_Ladestation, X_Achse, Ladestation, Roll, Foerderband, Home_Position, Vision_Position, Sichere_Vision_Board_Position, Vision_Board_Position_1, Vision_Board_Position_2, Sichere_Ausgangsposition, Sichere_Dominostein_Position, Sichere_Kettenreaktion_Position, Dominostein_Position, Infrarot_Sensor, Kettenreaktion_Position, Umstossen X_Achse = 0.15 for count3 in range(5): Setzen_der_Positionen() X_Achse = (X_Achse if isinstance(X_Achse, Number) else 0) + -0.02 n.set_arm_max_velocity(100) n.change_tool(TOOL_GRIPPER_2_ID) n.move_joints(Home_Position) n.wait(0.5) n.move_joints(Vision_Position) n.wait(0.5) n.control_conveyor(6, True, 100, -1) Infrarot_Sensor = n.digital_read(GPIO_1A) while not (not Infrarot_Sensor or (n.detect_object(n.get_workspace_ratio('Foerderband'), "ANY", "ANY")[0])): Infrarot_Sensor = n.digital_read(GPIO_1A) n.control_conveyor(6, False, 0, 1) if not Infrarot_Sensor: n.control_conveyor(6, True, 100, 1) n.wait(1.4) n.control_conveyor(6, False, 0, 1) while not (n.pick('Foerderband', float(3)/100, "ANY", "ANY")[0]): pass n.move_joints(Vision_Position) n.wait(0.5) n.move_joints(Sichere_Vision_Board_Position) n.wait(0.5) n.set_arm_max_velocity(30) n.move_pose(*Vision_Board_Position_1) n.wait(0.5) n.open_gripper(TOOL_GRIPPER_2_ID, 100) n.wait(0.5) n.move_pose(*Vision_Board_Position_2) n.wait(0.5) n.close_gripper(TOOL_GRIPPER_2_ID, 100) n.wait(0.5) n.move_joints(Sichere_Vision_Board_Position) n.wait(0.5) n.move_joints(Sichere_Ausgangsposition) n.wait(0.5) n.move_pose(*Sichere_Dominostein_Position) n.wait(0.5) n.set_arm_max_velocity(5) n.move_pose(*Dominostein_Position) n.wait(0.5) n.open_gripper(TOOL_GRIPPER_2_ID, 100) n.set_arm_max_velocity(100) n.move_pose(*Sichere_Dominostein_Position) n.wait(0.5) n.close_gripper(TOOL_GRIPPER_2_ID, 500) n.move_joints(Home_Position) """Describe this function... """ def Dominostein_11_15_setzen(): global n, Sichere_Ladestation, X_Achse, Ladestation, Roll, Foerderband, Home_Position, Vision_Position, Sichere_Vision_Board_Position, Vision_Board_Position_1, Vision_Board_Position_2, Sichere_Ausgangsposition, Sichere_Dominostein_Position, Sichere_Kettenreaktion_Position, Dominostein_Position, Infrarot_Sensor, Kettenreaktion_Position, Umstossen X_Achse = 0.05 for count4 in range(5): Setzen_der_Positionen() X_Achse = (X_Achse if isinstance(X_Achse, Number) else 0) + -0.02 n.set_arm_max_velocity(100) n.change_tool(TOOL_GRIPPER_2_ID) n.move_joints(Home_Position) n.wait(0.5) n.move_joints(Vision_Position) n.wait(0.5) n.control_conveyor(6, True, 100, -1) Infrarot_Sensor = n.digital_read(GPIO_1A) while not (not Infrarot_Sensor or (n.detect_object(n.get_workspace_ratio('Foerderband'), "ANY", "ANY")[0])): Infrarot_Sensor = n.digital_read(GPIO_1A) n.control_conveyor(6, False, 0, 1) if not Infrarot_Sensor: n.control_conveyor(6, True, 100, 1) n.wait(1.4) n.control_conveyor(6, False, 0, 1) while not (n.pick('Foerderband', float(3)/100, "ANY", "ANY")[0]): pass n.move_joints(Vision_Position) n.wait(0.5) n.move_joints(Sichere_Vision_Board_Position) n.wait(0.5) n.set_arm_max_velocity(30) n.move_pose(*Vision_Board_Position_1) n.wait(0.5) n.open_gripper(TOOL_GRIPPER_2_ID, 100) n.wait(0.5) n.move_pose(*Vision_Board_Position_2) n.wait(0.5) n.close_gripper(TOOL_GRIPPER_2_ID, 100) n.wait(0.5) n.move_joints(Sichere_Vision_Board_Position) n.wait(0.5) n.move_joints(Sichere_Ausgangsposition) n.wait(0.5) n.move_pose(*Sichere_Dominostein_Position) n.wait(0.5) n.set_arm_max_velocity(5) n.move_pose(*Dominostein_Position)

n.wait(0.5) n.open_gripper(TOOL_GRIPPER_2_ID, 100) n.set_arm_max_velocity(100) n.move_pose(*Sichere_Dominostein_Position) n.wait(0.5) n.close_gripper(TOOL_GRIPPER_2_ID, 500) n.move_joints(Home_Position) """Describe this function... """ def Dominostein_15_20_setzen(): global n, Sichere_Ladestation, X_Achse, Ladestation, Roll, Foerderband, Home_Position, Vision_Position, Sichere_Vision_Board_Position, Vision_Board_Position_1, Vision_Board_Position_2, Sichere_Ausgangsposition, Sichere_Dominostein_Position, Sichere_Kettenreaktion_Position, Dominostein_Position, Infrarot_Sensor, Kettenreaktion_Position, Umstossen X_Achse = -0.05 for count5 in range(5): Setzen_der_Positionen() X_Achse = (X_Achse if isinstance(X_Achse, Number) else 0) + -0.02 n.set_arm_max_velocity(100) n.change_tool(TOOL_GRIPPER_2_ID) n.move_joints(Home_Position) n.wait(0.5) n.move_joints(Vision_Position) n.wait(0.5) n.control_conveyor(6, True, 100, -1) Infrarot_Sensor = n.digital_read(GPIO_1A) while not (not Infrarot_Sensor or (n.detect_object(n.get_workspace_ratio('Foerderband'), "ANY", "ANY")[0])): Infrarot_Sensor = n.digital_read(GPIO_1A) n.control_conveyor(6, False, 0, 1) if not Infrarot_Sensor: n.control_conveyor(6, True, 100, 1) n.wait(1.4) n.control_conveyor(6, False, 0, 1) while not (n.pick('Foerderband', float(3)/100, "ANY", "ANY")[0]): pass n.move_joints(Vision_Position) n.wait(0.5) n.move_joints(Sichere_Vision_Board_Position) n.wait(0.5) n.set_arm_max_velocity(30) n.move_pose(*Vision_Board_Position_1) n.wait(0.5) n.open_gripper(TOOL_GRIPPER_2_ID, 100) n.wait(0.5) n.move_pose(*Vision_Board_Position_2) n.wait(0.5) n.close_gripper(TOOL_GRIPPER_2_ID, 100) n.wait(0.5) n.move_joints(Sichere_Vision_Board_Position) n.wait(0.5) n.move_joints(Sichere_Ausgangsposition) n.wait(0.5) n.move_pose(*Sichere_Dominostein_Position) n.wait(0.5) n.set_arm_max_velocity(5) n.move_pose(*Dominostein_Position) n.wait(0.5) n.open_gripper(TOOL_GRIPPER_2_ID, 100) n.set_arm_max_velocity(100) n.move_pose(*Sichere_Dominostein_Position) n.wait(0.5) n.close_gripper(TOOL_GRIPPER_2_ID, 500) n.move_joints(Home_Position) """Describe this function... """ def Kettenreaktion_ausl_C3_B6sen(): global n, Sichere_Ladestation, X_Achse, Ladestation, Roll, Foerderband, Home_Position, Vision_Position, Sichere_Vision_Board_Position, Vision_Board_Position_1, Vision_Board_Position_2, Sichere_Ausgangsposition, Sichere_Dominostein_Position, Sichere_Kettenreaktion_Position, Dominostein_Position, Infrarot_Sensor, Kettenreaktion_Position, Umstossen X_Achse = -0.15 Setzen_der_Positionen() n.set_arm_max_velocity(100) n.change_tool(TOOL_GRIPPER_2_ID) n.move_joints(Home_Position) n.wait(0.5) n.move_joints(Vision_Position) n.wait(0.5) n.move_joints(Sichere_Kettenreaktion_Position) n.wait(0.5) n.move_pose(*Kettenreaktion_Position) n.wait(0.5) n.set_arm_max_velocity(30) n.move_pose(*Umstossen) n.wait(0.5) n.move_joints(Sichere_Kettenreaktion_Position) n.wait(0.5) n.set_arm_max_velocity(100) n.move_joints(Vision_Position) n.wait(0.5) n.move_joints(Home_Position) n.wait(0.5) #Hauptprogramm Start Dominosteine_auf_Foerderband_legen() Dominostein_1_5_setzen() Dominosteine_auf_Foerderband_legen() Dominostein_6_10_setzen() Dominosteine_auf_Foerderband_legen() Dominostein_11_15_setzen() Dominosteine_auf_Foerderband_legen() Dominostein_15_20_setzen() Kettenreaktion_ausl_C3_B6sen() #Hauptprogramm Ende

12.7 Eigenständigkeitserklärung

Ich erkläre hiermit, dass ich die vorliegende Diplomarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen benutzt habe. Alle Stellen, die wörtlich oder sinngemäss aus Quellen entnommen wurden, habe ich als solche kenntlich gemacht. Mir ist bewusst, dass Verstöße gegen die Grundsätze der Selbstständigkeit als Täuschung betrachtet und entsprechend der Prüfungsordnung geahndet werden.

Die Diplomarbeit darf nur mit Genehmigung des Verfassers an Dritte weitergegeben werden.

Jeshua Roth

11.10.2021, Schötz