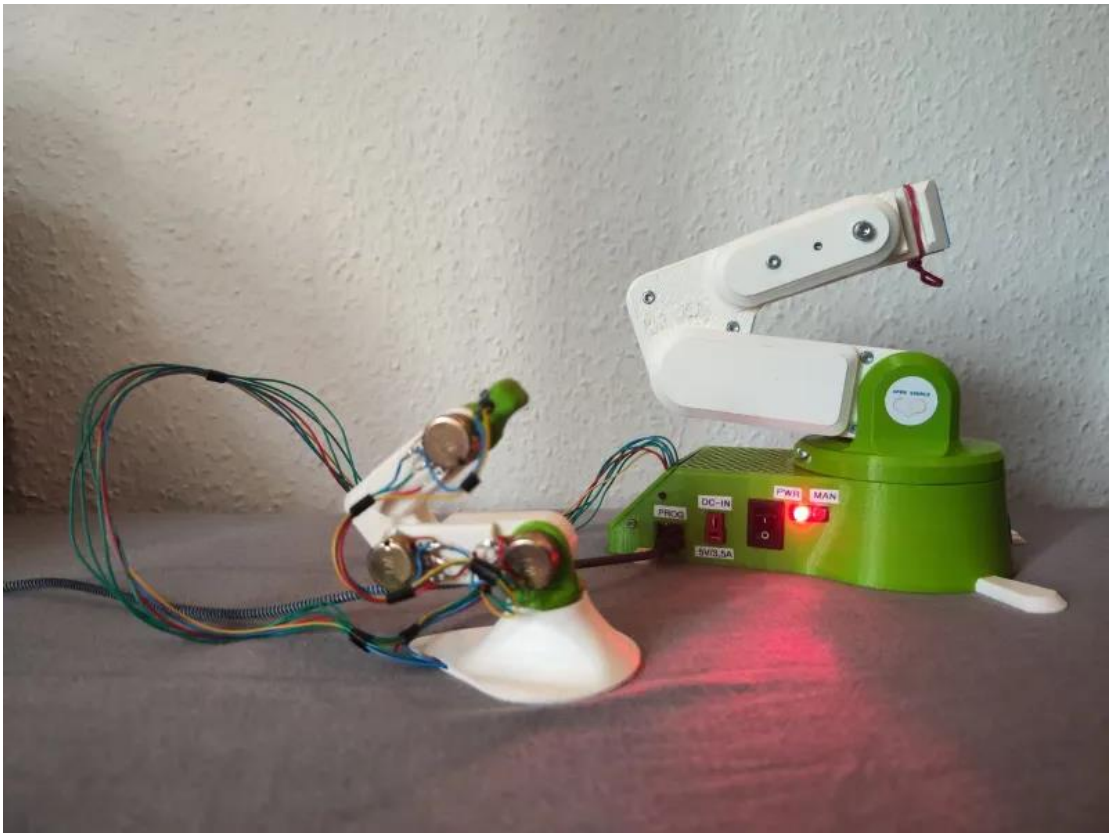
Projektdokumentation

zur Lehrveranstaltung Embedded Systems 1

Wintersemester 2023/2024

Hochschullehrer: Prof. Dr. -Ing. Andreas Pretschner

Aufbau und Programmierung eines Roboters mit Telemanipulation und App-Steuerung



Vorgelegt von:	Teresa Schüler und Nils Wiora
Studiengang:	Elektro- und Informationstechnik
Seminargruppe:	20-EIK/IAS
Datum der Abgabe:	17.03.2024

Inhaltsverzeichnis

1. Einleitung

Unsere Aufgabe bestand darin, einen Roboter nach einer Vorlage nachzubauen und daran anknüpfend eine Erweiterung zu implementieren. Die Grundlage dafür waren die YouTube Videos[1] von Kelton auf seinem Kanal BuildSomeStuff. In zwei Videos erklärt er Schritt für Schritt, wie der Roboter zusammengebaut werden kann. Der Roboter besteht zum größten Teil aus 3D-gedruckten Teilen. Die Bewegung der vier Gelenke und des Greifers wird durch Servomotoren ermöglicht. Die Kraftumlenkung erfolgt über Zahnräder und Aufsätze für die Ritzel der Servos. Der einfache Programmcode, um den Roboter in Betrieb zu nehmen, wird ebenfalls im Video gezeigt. Der Arduino Uno erhält Signale von einer Fernbedienung aus Potentiometern, die er als Stellsignale an den Roboter weitergibt.

Der fertige Aufbau sollte dann so erweitert werden, dass eine Ansteuerung des Roboters über eine andere Schnittstelle als die Fernbedienung möglich ist. Es soll möglich sein dem Arm Positionen vorzugeben, um Aussagen über die Genauigkeit der Positionierung machen zu können. Die Steuerung wird mit einer Android-App realisiert, die eine Bluetooth Verbindung mit dem Arduino aufbauen kann. In der nachfolgenden Dokumentation werden die Herausforderungen beim Aufbau und der Programmierung näher beschrieben.

[1] Siehe Quellen, Youtube Tutorial

2. Zusammenbau des Roboters

2.1 Idee

Die 3D-gedruckten Teile sollen mit Schrauben so zusammengebaut werden, dass die fünf Servomotoren an den dafür vorgesehenen Stellen fest verbaut sind. Die Zahnräder und Servoaufsätze müssen so ineinandergreifen, dass sie ihre Kraft auf die einzelnen Glieder übertragen und diese dadurch ansteuerbar werden. Die fünf Servos sind über ein Erweiterungsboard mit dem Arduino verbunden. Der Strombedarf der Servomotoren überschreitet die Kapazitäten der Arduino Ausgänge, sodass das Board notwendig ist. Das Erweiterungsmodul kommuniziert über I2C mit dem Arduino und gibt dessen Steuerbefehle an die Servos weiter. Die Sollposition erhalten die Servos über ein PWM-Signal, dessen Pulsweite entspricht dem Winkel, der angefahren werden soll. In der ursprünglichen Version des Roboters wird diese Pulsweite durch anliegenden Analogwerte an A0 bis A3 bestimmt. Die Analogwerte werden von der Fernbedienung erzeugt.
Die Fernbedienung besteht auch aus 3D-gedruckten Gliedern, die über Potentiometer miteinander gekoppelt sind. Bewegt man die Gelenke der Fernbedienung, die aussieht wie ein Miniaturmodell des Roboters, werden die Widerstandswerte der einzelnen Potis verändert. Die Analogwerte sind dann die Spannungen, die über den Potentiometern am Miniaturroboter abfallen. An dieser Fernsteuerung befindet sich auch ein Knopf, der den Befehl zum Öffnen des Greifers an den Roboter gibt. Der Greifer wird über ein Gummiband, wenn er nicht betätigt wird (Servo bei 90°), geschlossen gehalten. Soll der Greifer geöffnet werden, fährt der Servo einen Winkel von 180° an und wirkt so gegen die Spannkraft des Gummibandes.

2.2 Umsetzung

Zu Beginn wurde eine Materialliste erstellt, anhand der entschieden werden konnte, welche Komponenten bestellt werden müssen und welche noch im Lager vorhanden waren. Die Materialliste ist in einem git[2] als Excel-Tabelle beigefügt. Der 3D-Druck konnte in der HTWK mit den CAD-Dateien aus der Vorlage erfolgen. Die Dateien sind ebenfalls im git zu finden. Der weitere Aufbau erfolgt nach Anleitung aus den genannten Videos und einem Stromlaufplan, der gemeinsam mit den CAD-Dateien zur Verfügung gestellt wurde. Es ergaben sich die folgenden Arbeitspakete:

- Servoleitungen verlängern
- Servos in den Gehäuseteilen montieren (verschrauben)
- Gehäuse und Greifer zusammenbauen
- Fernbedienung zusammensetzen: Potentiometer und Knopf anlöten
- Verlöten der Fernbedienung mit Arduino Pins
- Anschluss der Servoleitungen an das Erweiterungsmodul
- Stromversorgung: Stecker, Schalter, LED montieren und löten
- Übertragen der Software und Test

[2] Siehe Anhang

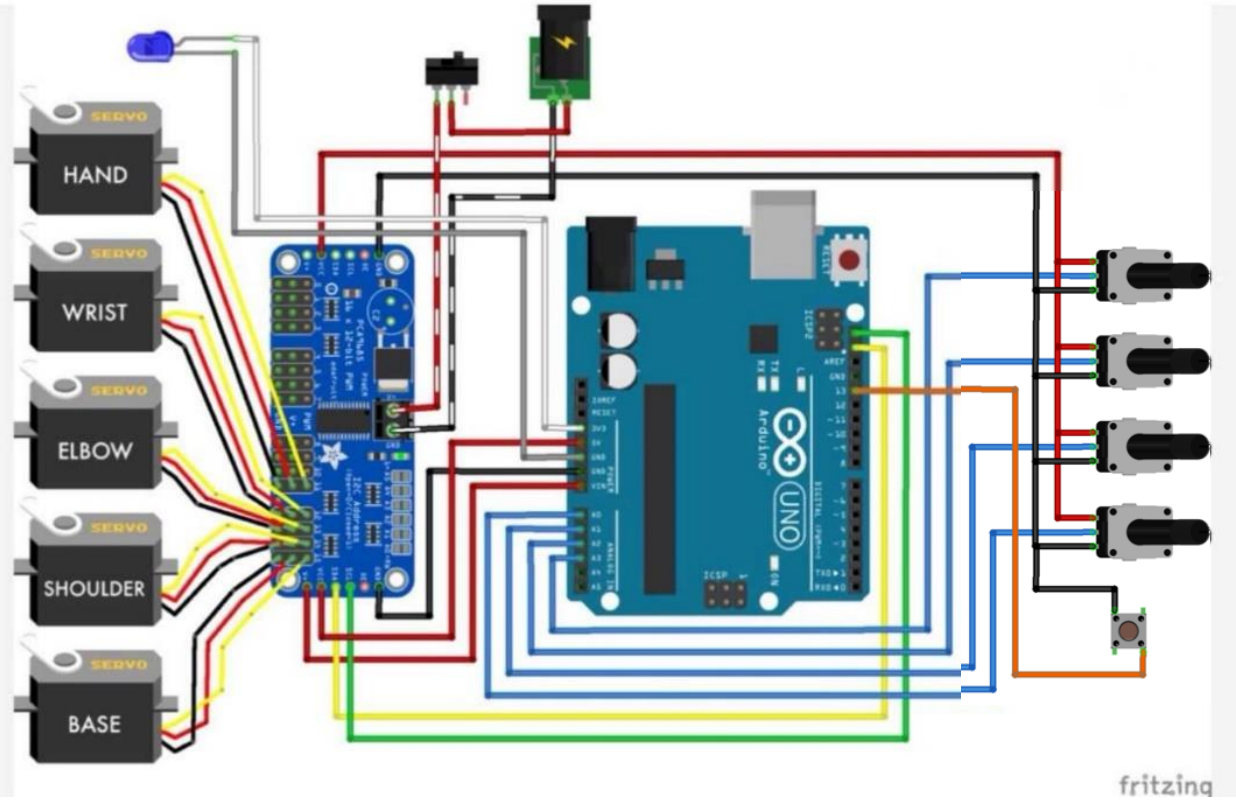


Abbildung 1 Stromlaufplan

2.3 Herausforderungen

- Ungenauigkeit des 3D-Drucks:
  Da die 3D-gedruckten Teile stellenweise nicht hinreichend präzise gedruckt wurden, gab es hin und wieder Probleme mit der Passgenauigkeit. Nuten an den Gelenkstellen oder an der Fernbedienung waren teilweise zu eng, sodass sie aufgefeilt werden mussten. Löcher für einzelne Schrauben an der Bodenplatte waren zu klein, sodass sie aufgebohrt wurden.

- Fehler im Stromlaufplan:
  Da wir uns beim Aufbau und vor allem beim Löten an den Stromlaufplan (Abb. 1) gehalten haben, ist uns erst beim Systemtest aufgefallen, dass drei Potentiometer der Fernbedienung, anders als im Stromlaufplan eingezeichnet, mit vertauschten Anschlüssen für GND und VCC angelötet werden müssen.

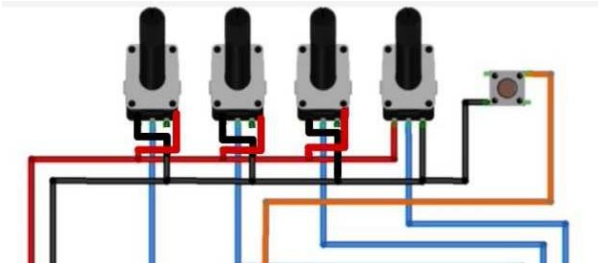


Abbildung 3 Stromlaufplan korrigiert

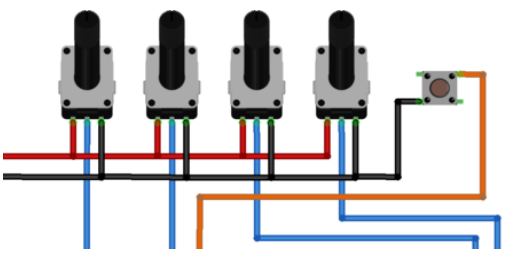


Abbildung 2 Stromlaufplan ursrünglich

Aufgefallen ist die Vertauschung durch ein entgegengesetztes Verhalten des Roboters zur Vorgabe der Fernbedienung. Zur Auswahl stand, die Änderung der Software für die betreffenden Servos, oder die Pins der Potentiometer umzulöten. Wir entschieden uns für die hardwareseitige Lösung, um eine einheitliche Software beibehalten zu können, auch im Hinblick auf die geplanten Erweiterungen. Daraufhin hat sich der Roboter wieder simultan und nicht genau entgegengesetzt zur Fernbedienung bewegt.

- Kurschluss auf dem Servomodul:
  Beim Systemtest viel auf, dass nach Zuschalten der Spannung diese am Labornetzteil sofort einbrach. Ohne das Servoboard funktionierte der Arduino ohne Probleme, sodass das Problem auf die Servos und ihr Modul eingegrenzt werden konnte. Die Vermutung, dass das Servoboard einen Kurzschluss hat, hat sich nach einer Durchgangsprüfung der Einspeiseklemmen des Servomoduls bestätigt. Das Board war bereits beim Einbau defekt. Um den Fehler zu beheben wurde ein neues Board bestellt, auf den gleichen Fehler getestet und eingebaut.

- Bruch in der Servoleitung:
  Beim Systemtest stellte sich heraus, dass der Servo des Greifers nicht ansteuerbar war. Eine Funktionsanalyse der beteiligten Komponenten hat ergeben, dass Leitungen des Servos abgebrochen waren. Grund dafür war der enge und bewegte Kabelkanal und dass nicht auf ausreichend Zugentlastung der Leitung geachtet wurde. Der defekte Motor wurde durch einen funktionstüchtigen ersetzt und an der empfindlichen Anschlussstelle Heißkleber zur Fixierung angebracht.

- Strombedarf unbekannt:
  Während des Tests mit einer Stromversorgung (I_{max}= 2,5 A) wies der Roboter ein unerwartetes Verhalten auf. Die Servomotoren haben gezuckt, waren nur teilweise aktiv und nach ein paar Sekunden hat der Roboter gar nicht mehr auf die Anweisungen der Fernbedienung reagiert. Das liegt daran, dass der Strombedarf des Roboters höher war als die Leistung, die Labornetzteil liefern konnte. Die Lösung für das unerwünschte Verhalten ist, dem Roboter mehr Strom zur Verfügung zu stellen. Wir haben dafür die beiden Ausgänge des Labornetzteils mit jeweils 2,5A parallelgeschalten, um dem Roboter mit bis zu 5 A zu versorgen. Eine Messung zeigte, bei Spitzenlast hatte der Roboter einen Strombedarf von bis zu 3,12 A. Bei der Beschriftung wurde der notwenige Strom und die benötigte Betriebsspannung vermerkt.

- Power LED:
  Im Original wurde eine blaue LED verwendet für die Anzeige, ob der Roboter angeschaltet ist. Die von uns verwendete rote LED hat eine geringere Flussspannung als die 3,3 V der blauen, sodass ein zusätzlicher Vorwiderstand eingebaut werden musste. Mit einer Flussspannung von nur 2,3 V hätten 3,3 V auf Dauer zu einem Defekt der LED geführt.

3. Ansteuerung per App

3.1 Idee

Um die geforderte Erweiterung im Projekt umzusetzen und auch noch eine eigene Idee einzubringen, haben wir den Roboter in der Software erweitert. Für die Steuerung des Roboters von einem externen Gerät aus, bietet der Arduino Uno R4 mehrere Möglichkeiten. So hat der Arduino Uno R4 ein integriertes WLAN-Modul, was die Netzwerkanbindung des Arduinos möglich macht. Man kann einen Webserver erstellen, von dem der Arduino Daten liest, die ein anderer Webclient darauf schreibt. Beispielsweise könnte man den Roboter über ein Smartphone steuern, indem man die URL des Webservers aufruft. Der Arduino liest die Stellsignale vom Webserver und gibt sie an die Servos weiter.

Genauso könnte man einen Webserver über NodeRed in Form eines Dashboards erstellen und über einen Browser darauf zugreifen. NodeRed sendet dann die Daten über TCP an den Arduino. Bei der Lösung mit NodeRed läuft dann der Webserver nicht auf dem Arduino, sondern auf einem anderen Gerät mit entsprechendem Betriebssystem. Denkbar wäre ein RaspberryPI mit dem RaspianOS, das wiederum eine Linux Distribution ist. Auch hier könnte man den Roboter unabhängig vom Gerät steuern, solange dort ein Webbrowser ausgeführt werden kann. Eine weitere Möglichkeit ist die Kommunikation über das MQTT-Protokoll. Ein weiteres Gerät mit einem entsprechenden Betriebssystem stellt den MQTT-Broker dar. Dieser vermittelt zwischen dem Arduino, der auch das MQTT-Protokoll beherrscht und einem zweiten Client. Über eine Android-App aber auch über ein Terminal auf einen PC oder RaspberryPI kann beispielsweise ein Stellsignal an den Broker gesendet werden, der das Signal an der Arduino weitergibt. Nachteilig an den Lösungen mit NodeRed ist die Notwendigkeit zusätzlicher Geräte wie einem Router oder weiteren PCs.

Außerdem bietet der Arduino Uno R4 die Möglichkeit eine Bluetooth Verbindung über den Bluetooth Low Energy Standard aufzubauen. Dazu reicht eine Android App und der Arduino an sich. Aufgrund des geringen Gerätebedarfs und der Möglichkeit eine Android-App zu verwenden, haben wir uns für die Lösung mit Bluetooth entschieden. Neben der Ansteuerung per Fernbedienung, kann dann nach automatischer Umschaltung, die Ansteuerung auch über eine Android-App erfolgen. Weiterhin wird für die Anzeige der Betriebsart eine Mode-LED eingebaut. Ist kein Gerät mit dem Roboter verbunden, leuchtet diese.

3.2 Umsetzung

3.2.1 App

Der Arduino R4 verfügt über ein Bluetooth-Modul für Low Energy Bluetooth. Über das Online Tool MIT App Inventor haben wir eine App erstellt, mit der der Roboter gesteuert werden kann. Der MIT App Inventor bietet eine grafische Programmieroberfläche. Dort können verschiedene Funktionsblöcke miteinander verschalten werden. Auch die grafische Oberfläche kann mit fertigen Bedienelementen erzeugt werden. Unterschiedliche Bibliotheken ermöglichen die Integration weiter Funktionen in eine Android-App. Für die Bluetooth Verbindung wird die Bibliothek „BluetoothLE1“ verwendet.

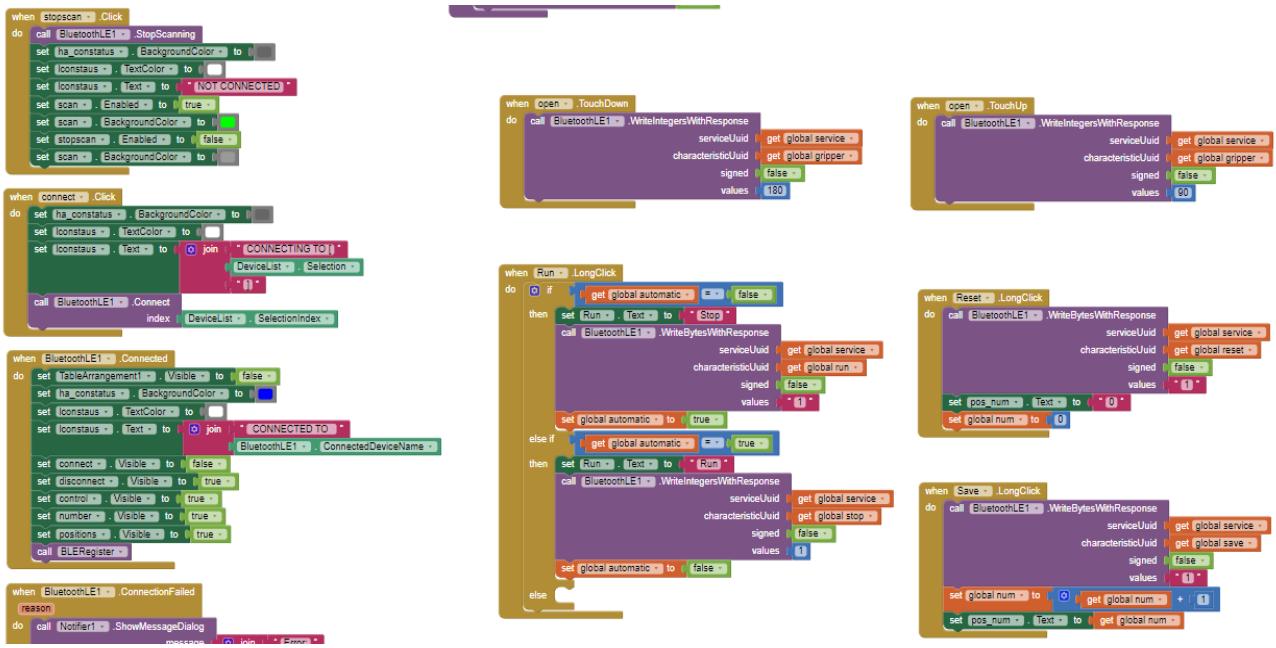


Abbildung 4 Ausschnitt aus dem App Projekt im App Inventor

Nach Aktivieren von Bluetooth über die App (Scan Devices) werden alle möglichen Teilnehmer angezeigt. Der Arduino wird aus der Liste gewählt und mit dem Gerät verbunden. Sobald der Arduino verbunden ist, wird die Ansteuerung per Fernbedienung deaktiviert und die Ansteuerung per App aktiviert. Die Mode-LED erlischt. In der App werden die UUIDs der Charakteristiken angegeben, auf die dann zugegriffen werden kann. Definiert werden diese im Arduino Programm. Die Charakteristiken sind ähnlich wie die Nodes bei dem bekannten Kommunikationsprotokoll OPC-UA. Die Charakteristiken können gelesen und beschrieben werden, sie dienen dem Austausch von Informationen zwischen der App und Arduino. So werden dann z.B. Integer-Werte für die Positionen der Servos übermittelt. Der Roboter kann über Schieberegler für die Winkel der Servos oder den Button für den Greifer gesteuert werden. Ein weiteres Feature ist die Möglichkeit, über einen Save-Button Positionen zu speichern und auf Wunsch durch Betätigen des Run-Buttons wieder anzufahren.

Werden mehrere Positionen gespeichert, fährt der Roboter diese nacheinander wie in einem Automatik Programm immer wieder in einem Loop an, bis der Stop-Button gedrückt wird, um das Programm zu pausieren. Das Drücken von Reset löscht alle gespeicherten Positionen.

3.2.2 Arduino

Im Programm werden Variablen für jedes einzelne Gelenk und den Greifer mit dem Datentyp

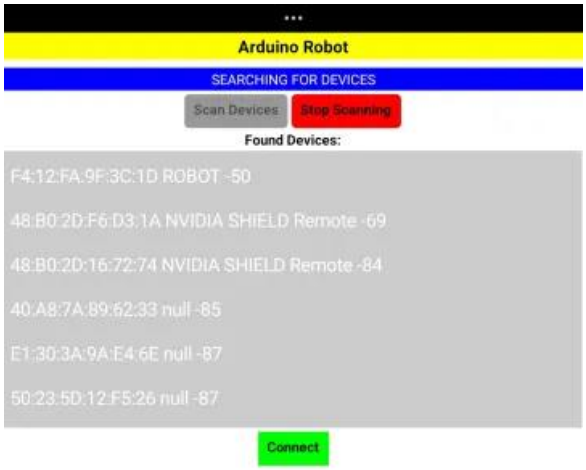


Abbildung 6 Appansicht Verbindungsauswahl

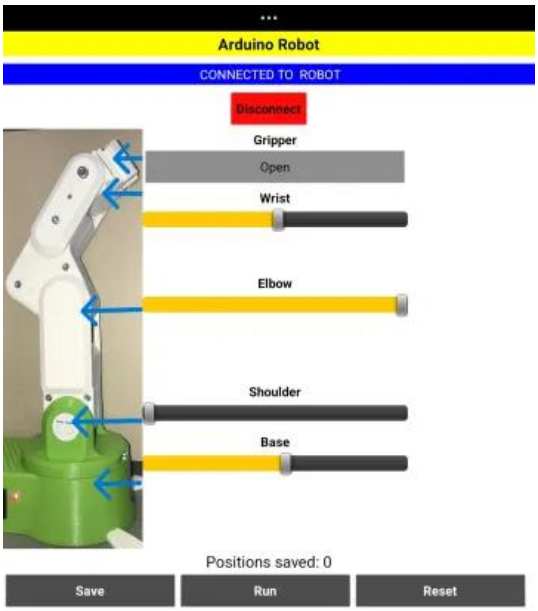


Abbildung 5 Steuerboard

Integer definiert. Außerdem werden die bereits angesprochenen Charakteristiken festgelegt. Jede Charakteristik bekommt eine UUID und ein Datentyp zugewiesen. Die Charakteristiken werden dem Service zugeordnet. Der Service ist der übergeordnete Bluetooth Dienst und in etwa vergleichbar mit einem Server. Die App kann dann diese Charakteristiken beschreiben. Im Loop werden sie regelmäßig gelesen und Änderungen an die Servos weitergegeben.
Beim Speichern einer Position über „Save“ werden die aktuellen Positionen jedes einzelnen Gelenks in das nächstfreie Arrayelement geschrieben. Wird „Run“ gedrückt, wird eine neue Schleife betreten, die nacheinander die gespeicherten Positionen den Servos übergibt. Ist die Position erreicht, wird das nächste Element angefahren. In jedem Durchlauf wird der Status der Stop-Charakteristik überprüft. Das passiert so lange, bis „Stop“ gedrückt wird, was wie eine Pausenfunktion wirkt. Bei den Charakteristiken für die Buttons handelt es sich um Bytes, hier wird eine 01 gesendet, wenn ein Button betätigt wurde. Im Programm wird nur abgefragt, ob ein neuer Wert geschrieben wurde. Der tatsächliche Wert interessiert nicht.

3.3 Herausforderungen

- Verbindungsabbrüche im Automatikmodus:
  Sobald der Roboter über „Run“ im Automatik-Modus war, brach die Bluetooth-Verbindung ab. Der Roboter ist weiterhin sein Programm gefahren, allerdings konnte man ihn nicht mehr von der App aus steuern und ein Verbindungsfehler wurde in der Meldezeile der App ausgegeben. Die Lösung für das Problem ist eine Codezeile, die den Arduino prüfen lässt, ob noch Geräte zur Verbindung bereitstehen und hält damit die Verbindung am Leben. Diese Zeile findet sich auch im Loop des Beispielprogrammes von Arduino für die Steuerung einer LED.

  BLEDevice central = BLE.central();

- App Abstürze bei unterschiedlichen Android Versionen:
  Beim Testen der App fiel bei einem Smartphone mit Android 13 auf, dass die App beim Verbindungsaufbau abstürzt. Bei anderen Android-Geräten der Versionen 11 oder 14 gab es damit keine Probleme. Vermutlich liegt das Problem bei der Bibliothek für den App Inventor, da es für die Version bereits vermehrt zu Problemen gekommen ist im Zusammenhang mit Android 13. Gelöst werden kann das Problem von unserer Seite nur durch das Einsetzen eines anderen Geräts.

- Keine Anzeige welcher Modus aktiv ist:
  Um am Roboter einfach ersichtlich zu machen, welcher Programmmodus gewählt wurde (Fernbedienung oder App) haben wir dem Gehäuse eine LED hinzugefügt. Wenn die LED leuchtet, ist die Steuerung per Fernbedienung möglich.

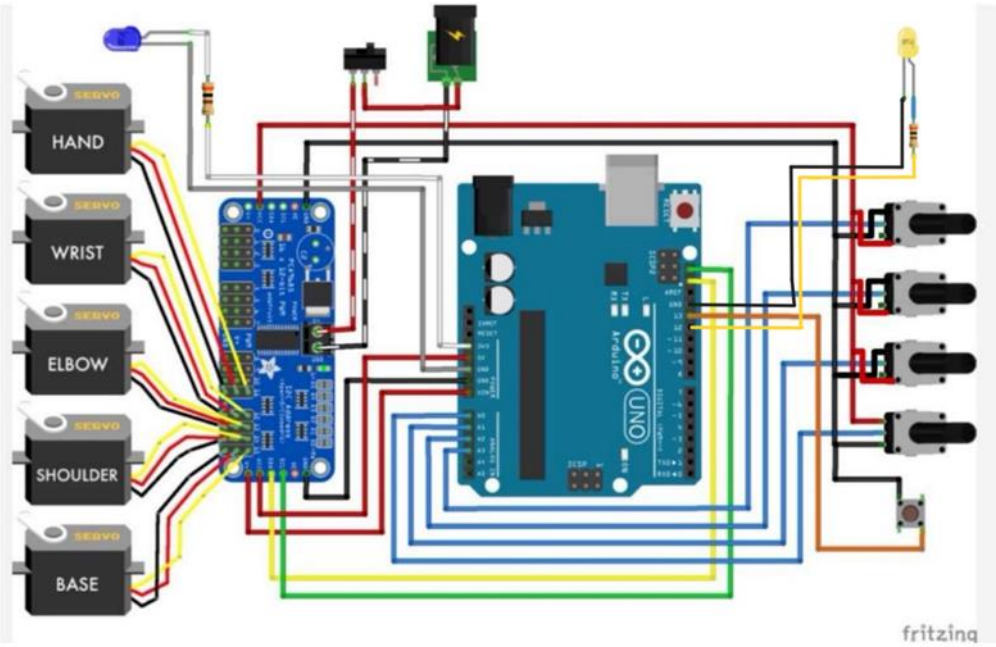


Abbildung 7 Stromlaufplan mit Mode-LED und Vorwiderstand an der Power-LED

4. Genauigkeitstest

Für die Untersuchung der Genauigkeit wurde ein Stift am Greiferservo befestigt. Die Stiftspitze repräsentiert den Toolcenterpoint (TCP). Anschließend wurde im Roboter eine Abfolge von Bewegungen abgespeichert, die er dann genau 25-mal ausführte. Der eingespannte Stift hinterließ Abdrücke auf einem Bogen Millimeterpapier. Auf dem Papier ließen sich anschließend die Punkte des Stiftes auswerten. Im zweiten Durchlauf wurde ein anderer Stift verwendet, der deutlichere Abdrücke auf dem Papier hinterließ.

Die Auswertung zeigte, dass die vorgegebene Zielposition entweder genau oder mit einigen Millimetern Abweichung getroffen wurde. Die Abweichungen ergeben sich aus der Bewegung des gesamten Roboters. Dieser hebt sich beim Ausfahren des Armes von seiner Unterlage ab. Fixiert man den Roboter, werden die Abweichungen minimal. Dieses Verhalten gibt es auch so bei Industrierobotern. Auch sie bewegen sich im Ganzen, wenn die Befestigung nicht ausreichend stark ist. Die bleibende Restabweichung ergibt sich aus der Stiftbefestigung am Greifer, die auch etwas nachgibt. Die erreichbare Genauigkeit reicht aber für einfache Pick and Place Anwendungen aus. Eine Abweichung des TCP um wenige Millimeter ist bei simplen Anwendungen nicht problematisch.

Abbildung 8 Test 1: Markiert ist die vorgegebene Position. Abgelegene Punkte entstehen, wenn der Roboter nicht fixiert wird

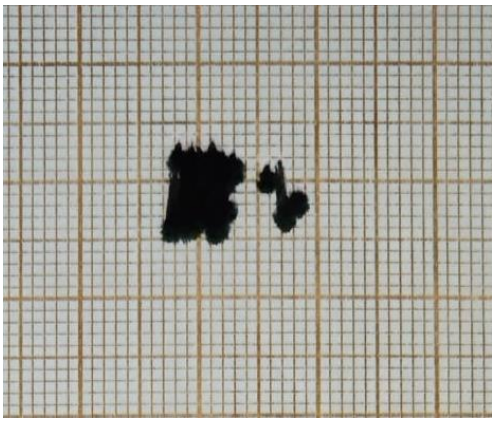


Abbildung 9 Test 2: Andere Stift, Abgelegene Punkte wieder durch fehlende Befestigung

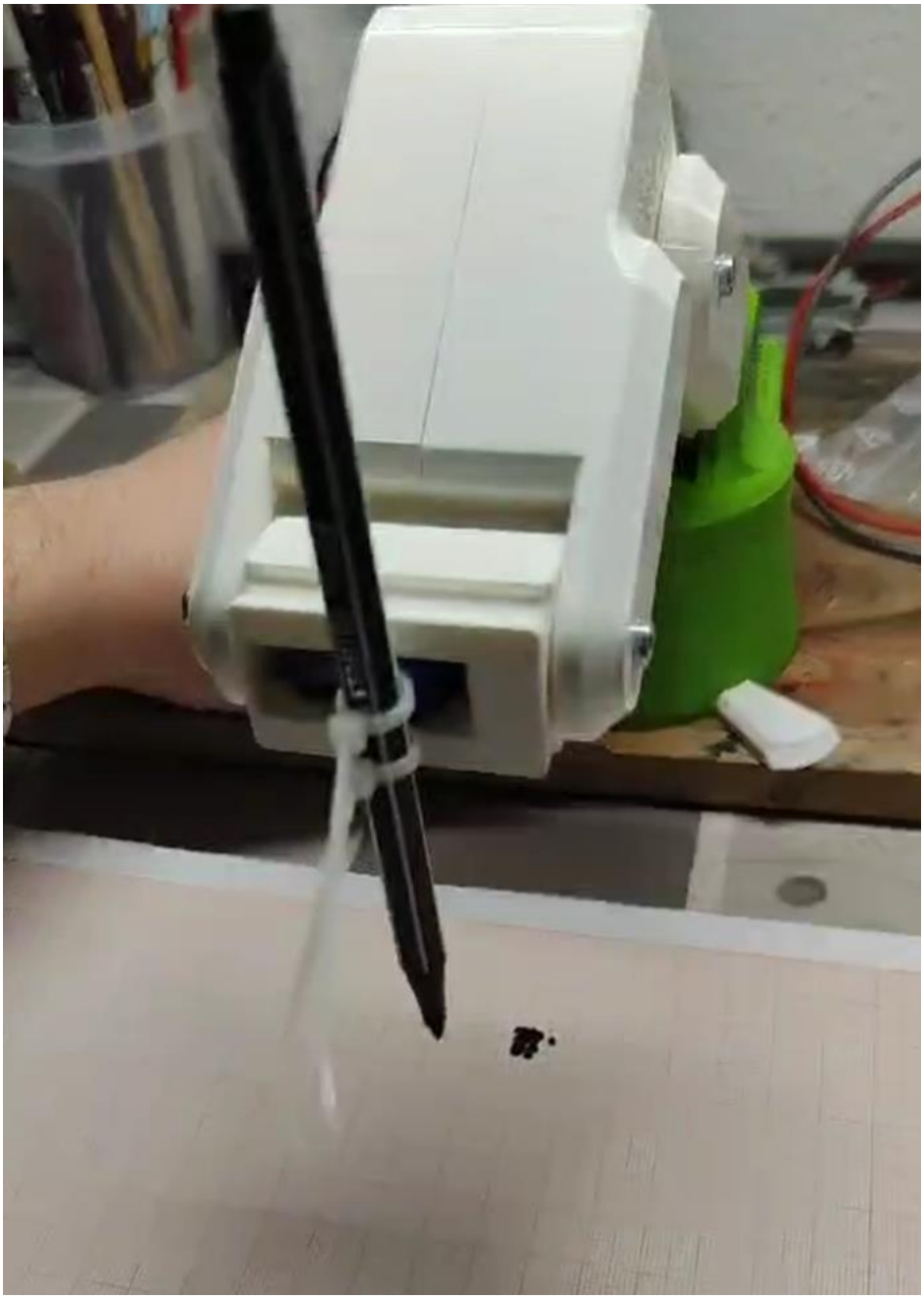


Abbildung 10 Testaufbau mit dem zweiten Stift

5. Fazit und Ausblick

Der Roboter lässt sich problemlos über die kleine Miniaturnachbildung steuern. Durch die Erweiterung des Arduino Codes ist es auch möglich über Bluetooth eine Verbindung zum Roboter aufzubauen. Steuern kann man diesen dann über die Android App, die so weit von uns getestet, auf Android 12 und Android 14 Geräten läuft. Mit der App lassen sich über Schieberegler Gelenkpositionen einstellen und anschließend auch speichern. Durch einen integrierten Automatikmodus können dem Roboter so Abläufe beigebracht werden, die er dann unbegrenzt ausführt. Allerdings nur solange der Roboter nicht von seiner Stromversorgung getrennt wird, da sich der Speicher sonst löscht.
Bei der Untersuchung der Genauigkeit wurde festgestellt, dass der Roboter seine gespeicherten Positionen auch präzise anfährt. Es gab nur einige Millimeter Abweichung, die größtenteils durch die Eigenbewegung des gesamten Roboters verursacht wurden. Fixiert man den Roboter auf seiner Unterlage bleiben auch diese Bewegungen des Gehäuses aus. Für einfache Anwendungen ist die erzielte Genauigkeit voll ausreichend.

Verbesserungspotential gibt es auch hier, obgleich der Roboter im jetzigen Zustand zuverlässig einsetzbar ist. Alle Verbesserungen könnten demzufolge in weiteren Projekten realisiert werden. Problematisch ist zum einen die Griffkraft des Greifers. Durch die geringe Kraft des Servos, können auch nur begrenzt starke Gummibänder genommen werden. Die kleine Kraft des Greifers schränkt die Verwendung des Roboters bei Pick and Place Anwendungen deutlich ein. In einem fortführenden Projekt, könnte ein anderer Greifer Mechanismus konstruiert werden. Außerdem bleibt der Positionsspeicher bei einer Unterbrechung der Stromversorgung nicht erhalten. Eine entsprechende Optimierung wäre sinnvoll, um die Anwendungsgebiete des Roboters zu erweitern. Mit einem entsprechenden Greifer und einem nicht flüchtigen Speicher, wäre dann auch eine Einbindung in eine Produktionsstrecke möglich. So könnte der Roboter z.B. Werkstücke von einem Förderband entnehmen. Die Steuerung des Roboters kann über MQTT erfolgen, das auch von SPS wie der Siemens S7-1500 oder S7-1200 unterstützt wird. Auch von B&R wird eine MQTT-Bibliothek angeboten. Selbstverständlich können auch der bestehende Code und die App um weitere Funktionen erweitert werden. Denkbar wäre die Einführung von Funktionstasten, um bestimmte Positionen abzuspeichern.

Die Möglichkeiten an das bestehende System anzuknüpfen sind vielfältig. Das bisherige Projekt bietet die Grundlage für fortführende Projekte.

Quellenangaben

YouTube Tutorial von BuildSomeStuff
Teil 1: https://www.youtube.com/watch?v=AIsVlgopqJc
Teil 2: https://youtu.be/OiQKw0lZ5Rw

Dateien, Schaltplan, Code für den Roboter
https://www.thingiverse.com/thing:6313449

Anhang

GIT-Link
https://github.com/oberstpuffi/es1_robot