**ACR – Arduino Cube Rover**

*Der Arduino Cube Rover ist zweiteiliges Arduino System, bei dem ihr selbst ein voll funktionsfähigen fahrenden Roboter zusammenbaut und programmiert, den ihr über einen selbstgebauten Controller mit Joystick und sogar Neigungssteuerung kontrollieren könnt.*

**Material**

* 2x Arduino Nano
* 1x H-Brücke (L298N)
* 2x 3-6V Getriebemotor mit Reifen
* 2x Bluetooth Modul (HC-05)
* 1x Joystick Modul (KY-023)
* 1x Neigungssensor (ADXL345)
* 3x verlötbarer Schalter
* 3x 9V Batterieblock
* 2x Breadboard (mittelgroß)
* 3x Breadboard (klein)
* 4x Morph3Dbot Würfel
* 1x Morph3Dbot Kugellager
* 2x Morph3Dbot Adapter für Batterieblöcke
* 2x Morph3Dbot Adapter für Breadboards
* 1x Morph3Dbot Adapter für H-Brücken
* 1x 3D gedruckte Controller Grundstruktur
  + Controller
  + Batteriefachdeckel
  + Elektronikfachdeckel

# **Hardware aufbauen**

Das Projekt besteht aus zwei verteilten eigenständigen Arduino Schaltungen: dem fahrenden Roboter selbst und dem Controller, der ihn steuert.

## **Roboter**

1. Gerüst bauen: Dazu nutzen wir die 4 Morph-3D-Bot Würfel und bringen sie mit den Verbindungsteilen in eine T-form, also drei in eine Reihe und eines seitlich an den mittleren Würfel. An den seitlich angebrachten Würfel kommt an die Unterseite das Kugellager, das später als Ausgleich zu den Motoren gilt.
2. Breadboard mit Arduino anbringen: Wir nehmen ein mittelgroßes Breadboard und kleben es auf den Morph3Dbot Breadboard-Adapter. Dann bringen wir den Adapter auf dem mittleren Würfel an und stecken dann den Arduino auf das Breadboard.
3. H-Brücke anbringen: Der Morph3Dbot Adapter für die H-Brücke besitzt 4 kleine Löcher an den Ecken. Wir befestigen die H-Brücke so auf dem Adapter, dass die Schraube von unten durch die Löcher nach oben zeigt, durch das jeweilige Loch an der H-Brücke rausguckt und dann mit einer Mutter festgezogen wird. Zwei Schrauben an gegenüberliegenden Ecken sollten ausreichen. Dann festigen wir den Adapter so oben auf dem außen liegenden Würfel, dass die Anschlüsse für den Arduino in Richtung des Arduinos zeigen.
4. Motoren anbringen: Bringe die zwei Morph3Dbot Adapter für Motoren an der Unterseite der beiden äußeren Würfel an. Danach montieren wir die Motoren so an dem Adapter, dass die Schrauben von dem Roboter wegzeigen und die Muttern außen zu sehen sind. Es ist darauf zu achten, dass die kleine Stange, auf die der Reifen gesteckt wird, auf der Außenseite ist und der Gewichtsverlagerung wegen die Motoren so anbringen, dass die Kabelanschlüsse auf der Seite des Kugellagers sind.
5. Bluetooth-Modul anbringen: Die Prozedur mit dem mittelgroßen Breadboard mit Adapter ist die gleiche wie in Schritt 2. Nun stecken wir das Bluetooth-Modul in das Breadboard.
6. Batterien anbringen: Die Morph3Dbot Adapter für die Batterien kommen jeweils links und rechts neben den einen einzelnen mittigen Würfel, das hilft noch etwas bei der Verlagerung des Gewichts.
7. Verkabelung: Nun müssen alle angebrachten Bauteile miteinander verkabelt werden. Wie die einzelnen Bauteile verkabelt werden, wisst ihr ja schon.

Ein Bild, das Diagramm enthält.

Automatisch generierte Beschreibung

## **Controller**

1. 3D-Druck des Gehäuses: Bei diesem Schritt hängt es davon ab, ob ihr das Gehäuse bereits fertig gedruckt besitzt oder ihr es noch drucken musst. Wenn das Gehäuse noch gedruckt werden muss, könnt ihr gerne der vorgegebenen Gehäuse-Datei kleine Änderungen durchführen, die es für euch schöner oder angenehmer machen. Wenn ihr gut mit Fusion360 umgehen könnt, sind auch Extras wie Einlass-Fassungen für die Bauteile oder effiziente Klicksysteme für die Verschlüsse eine gute Erweiterung.
2. Arduino anbringen: Wie auch bei dem Roboter bringen wir den Arduino Nano auf einem mittelgroßen Breadboard an. Das Breadboard mit dem Arduino Nano drauf legen wir dann nun einfach in das Gehäuse.
3. Joystick anbringen: Das Joystickmodul wird einfach auf die vier im Gehäuse hochstehenden Säulen gelegt.
4. Neigungssensor anbringen: Der Neigungssensor wird auf zwei kleine Breadboards angebracht. Dabei ist es wichtig, dass jeweils auf beiden Seiten die Breadboards zwei Reihen übrighaben. Das garantiert, dass die Breadboard-Anordnung in das Gehäuse passt, aber noch genug Platz für Anschlüsse sind. Den Neigungssensor legen wir dann nun hochkant in das Gehäuse direkt neben das Batteriefach.
5. Bluetooth-Modul anbringen: Das Bluetooth-Modul wird ebenfalls auf einem kleinen Breadboard angebracht, dabei muss das Modul in die äußerste Reihe gesteckt werden, denn da das Modul vertikal in dem Breadboard steckt, muss das Breadboard um 90 Grad gedreht in das Gehäuse gelegt werden, so, dass die Löcher vom Breadboard in Richtung Batterie schauen.
6. Stromversorgung: Das Batteriefach besitzt ein kleines Loch, durch welches die Kabel vom Batterie-Kabelaufsatz in das Innere des Gehäuse geführt werden. Im Dach des Gehäuses gibt es ein kleines Loch, in das der Schalter gesteckt und fixiert wird.
7. Verkabelung: Wie auch beim Roboter müssen nun alle angebrachten Bauteile miteinander verkabelt werden. Auch wie diese Bauteile verkabelt werden, solltet ihr bereits wissen.

Ein Bild, das Text, Elektronik enthält.

Automatisch generierte BeschreibungEin Bild, das Diagramm, Plan enthält.

Automatisch generierte Beschreibung

# **Programmierung**

Da wir zwei verschiedene Arduino Schaltungen haben, die auch beide vollkommen andere Aufgaben übernehmen, brauche wir natürlich auch zwei unterschiedliche Arduino Skripts. Diesmal beginnen wir mit dem Controller, da dieser in dieser Struktur die Rolle des Masters einnimmt.

## **Controller**

1. Setup: Der Taster des Joysticks muss als Eingang definiert werden. Darüber hinaus muss die serielle Kommunikation und virtuelle Bluetooth serielle Kommunikation mit 38400er Rate gestartet werden. Als letztes muss noch der Neigungssensor eingeschaltet werden.
2. Bereits fertiger Code: Für die Umrechnung der Joystick Daten in nutzbare Motor Daten ist eine komplizierte mathematische Berechnung von Nöten, die in diesem Fall jegliche Rahmen sprengen würde, daher wird sie von eurem Lehrer vorgegeben. Es müssen insgesamt fünf Funktionen sein: bool yAchsePos(), bool yAchseNeg(), bool xAchsePos(), bool xAchseNeg() und void calculateSpeed().
3. Bluetooth-Verbindung: Es muss eine Funktion geschrieben werden, die testet, ob die Bluetooth-Verbindung noch besteht. Die Funktion bekommt nichts übergeben und gibt ein bool zurück. In einer while-schleife wird nun der State-Pin nach HIGH abgefragt und bei Eintreten wird die while-condition Variable auf false gesetzt (je nachdem wie ihr sie definiert habt auch andersherum) und es wird noch in der while-schleife der bool true zurückgegeben. Bei der nächsten Abfrage wird die while schleife also direkt übersprungen. Damit es aber bei Abbruch der Verbindung nicht zu Fehlern kommt und die while-condition wieder korrekt ist, müssen wir nach die while-Schleife noch eine Abfrage des State-Pin nach LOW einbauen, die bei Eintreten die while-condition Variable wieder auf true setzt und false returned.
4. Daten übertragen: Die Datenübertragen lagern wir auch in eine Funktion aus. In dieser Funktion müssen wir das Konzept überlegen, wonach die Datenübertragen funktionieren soll. Eine Möglichkeit ist ein Array mit vier Elementen: zwei für die Richtung und zwei für die Geschwindigkeit. Die Richtung wird als -1 für Rückwärts, 0 für Stillstand und 1 für Vorwärts definiert und für die Übertragung jeweils eins ins positive geshiftet, da Bluetooth nur positive Werte übertragen kann. Die Geschwindigkeit ist das Ergebnis der calculateSpeed() Funktion, bei der die Variable motLInt die Geschwindigkeit des linken Motors und motRInt die Geschwindigkeit des rechten Motors ist. Die Richtung kann auch aus dem Vorzeichen der Variablen ausgelesen werden. Es ist auch zu empfehlen, um den 0 Punkt herum einen gewissen Bereich als Dead-Space zu definieren (+-32 oder +-64 sind Standardwerte), bei der auch noch weiterhin 0 als Geschwindigkeit übertragen wird, da sonst ein Stillstand so gut wie unmöglich ist. Am Ende der Funktion, nachdem das Array vollständig und nach einem gewissen Konzept vorbereitet wurde, kann dies nun über die virtuelle serielle Kommunikation übertragen werden.
5. Steuerungsart wechseln: Da die Steuerung des Roboters auf zwei verschiedene Arten geschehen soll, müssen wir eine Funktion schreiben, die aktuell geltende Steuerungsart bestimmt. Der Wechsel der Steuerung soll über den Taster des Joysticks gewechselt werden. Wie man Modi anlegt und mit Hilfe des Tasters wechselt ist ja bereits bekannt.
6. Daten auslesen: Hier müssen die Daten ausgelesen werden. Dazu sollte erstmal bekannt sein, welche Steuerungsart genutzt wird, daher zuerst einmal die Steuerungswechsel Funktion aufrufen. Die y-Achse wird immer über den Joystick ausgelesen, das kann also außerhalb einer Abfrage passieren. Die x-Achse wird jetzt je nach aktuelle Steuerungsart über Joystick oder Neigungssensor ausgelesen. Achtet darauf, dass der Sensor manchmal in der Ausgangsstellung, in der 0 ausgegeben werden sollte einen etwas größeren oder kleinen Wert haben kann, dann müsst ihr diese mit Code ausgleichen. Da wir für die calculateSpeed() Funktion ein Wertebereich von 0 bis 1023 benötigen, aber der Neigungssensor einen Bereich von -256 bis 255 hat, müssen wir diesen erst noch mappen und mit einer int() funktion umgeben, da der Rückgabewert ein long ist.
7. Loop: In der Loop müssen wir nun nur noch einmal abfragen, ob die Bluetooth-Verbindung besteht. Wenn ja, dann können wir nacheinander die Daten lesen Funktion, die calculateSpeed() Funktion und Daten übertragen Funktion aufrufen.

## **Roboter**

1. Setup: Die Anschlüsse der H-Brücke müssen als Ausgänge definiert werden. Darüber hinaus muss die serielle Kommunikation und virtuelle Bluetooth serielle Kommunikation mit 38400er Rate gestartet werden.
2. Motoren Richtungsfunktionen: Es müssen die Funktionen angelegt werden, die mit Angabe der Richtung automatisch die Motoren so ansteuern, dass diese sich in die gewünschte Richtung drehen.
3. Bluetooth-Verbindung: Auch bei dem Roboter muss eine Funktion geschrieben werden, die die Bluetooth-Verbindung testet und es kann im Grunde die Funktion des Controllers eins zu eins übernommen werden. Dabei ist es wichtig, dass zusätzlich noch bei der Abfrage, ob die Verbindung abgebrochen ist, die Geschwindigkeit und Richtung der Motoren auf Stillstand gesetzt werden.
4. Loop: Das ist der einfachste Teil. Hier muss lediglich die Bluetooth-Verbindung Funktion und das Ankommen von Bluetooth-Daten abgefragt werden. Falls sowohl eine Verbindung besteht als auch Bluetooth-Daten zum Lesen vorhanden sind können die Daten in einen vorbereiteten Buffer geschrieben werden. Hier ist darauf zu achten, dass das Auslesen des Buffers nach dem gleichen Konzept funktioniert, das wir vorher in dem Sketch des Controllers definiert haben. Dann müssen wir nur noch die Daten, die wir übertragen haben an die Motor Richtungsfunktionen übergeben bzw. mit analogWrite() die Geschwindigkeit vorgeben.