Hardware eines Selbstbalancierenden Roboters

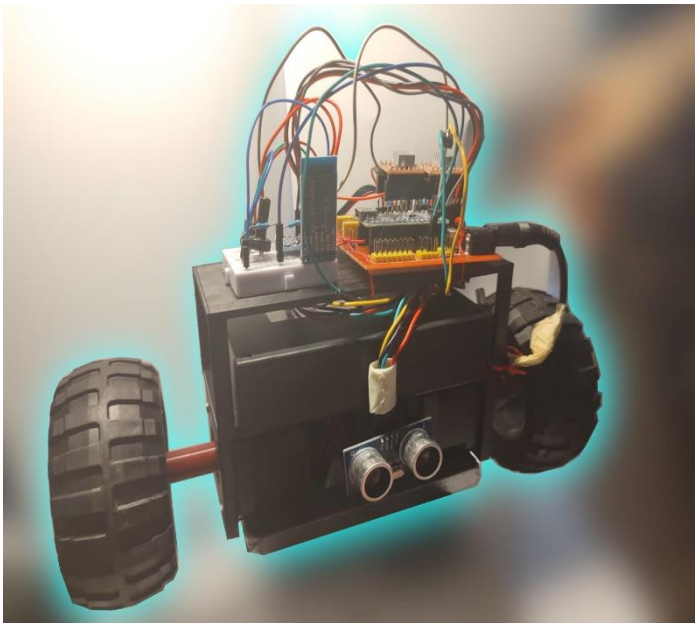

Abbildung 1 Self Balancing Robot

Im Alltage gibt es viele Anwendungen für selbstbalancierende Roboter. Ein populäres Beispiel stellt dabei der „Segway“ dar. Dieser ist ein Fortbewegungsfahrzeug, welches den Nutzer auf nur einer Achse transportiert. Wenn man die Funktionsweise hinter einer solchen Vorrichtung genauer betrachtet, erkennt man, dass es sich dabei um ein inverses Pendel handelt. Zur Veranschaulichung eines inversen Pendels kann man versuchen einen Stab auf der Handfläche zu balancieren. Man merkt sofort, dass das ganze ziemlich schwierig zu ermöglichen ist, da der Stab von der Schwerkraft immer in alle möglichen Richtungen gezogen wird. Deswegen spricht man hierbei auch von einem instabilen System. Diese benötigen immer einen Regler, der das ganze stabilisiert. Falls Interesse an der Umsetzung der hier verwendeten Regler besteht, können diese in der Ausarbeitung zur Software des Roboters nachgeschlagen werden.
Im Folgenden wird der selbstbalancierende Roboters am Beispiel einer real gebauten Version weiter ausgeführt. Dabei wird schrittweise der Aufbau, die verwendeten Bauteile, sowie deren genaue Verschaltung erklärt.

Grundlegender Aufbau:

In der nebenstehenden Abbildung 2 oder im Blender File in den Quellen kann man den Aufbau des Gerüsts für den Roboter sehen. Dabei handelt es sich um einen 3D-Druck mit drei Ebenen. Auf der untersten Ebene befinden sich die Stepper Motoren, sowie die Ultraschallsensoren mit zugehörigem Arduino. Die zweite Ebene wird vollständig vom Akku und damit der Energiequelle des Roboters eingenommen. An oberster Stelle befindet sich die dritte Ebene. Auf dieser befindet sich die gesamte Schalt- und Regellogik des Roboters.

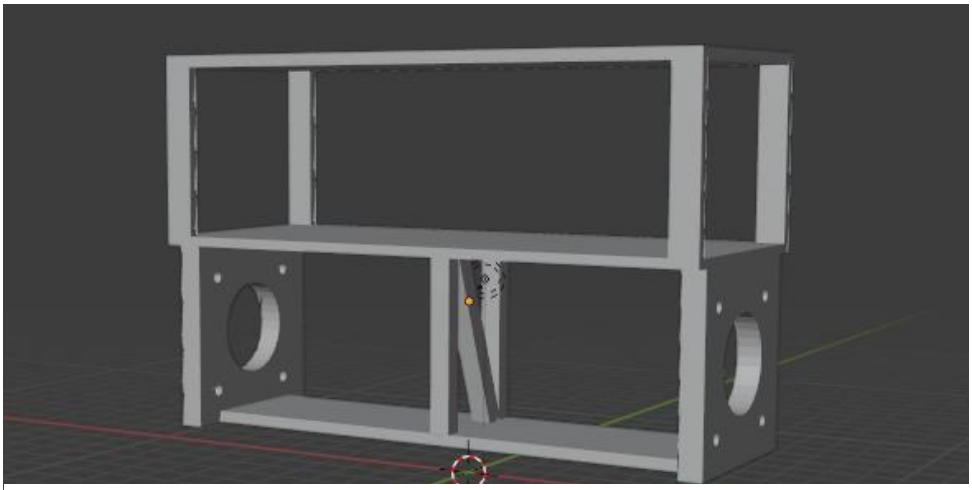

Abbildung 2 3D-Modell des Gestells

Verschaltung der Komponenten:

In der untenstehenden Abbildung 3 kann man die komplette Verschaltung der verwendeten Bauteile betrachten. Auf den ersten Blick ist das ganze sehr unübersichtlich, weswegen im Folgenden die gesamte Schaltung in kleinere Teilbereiche unterteilt wird.

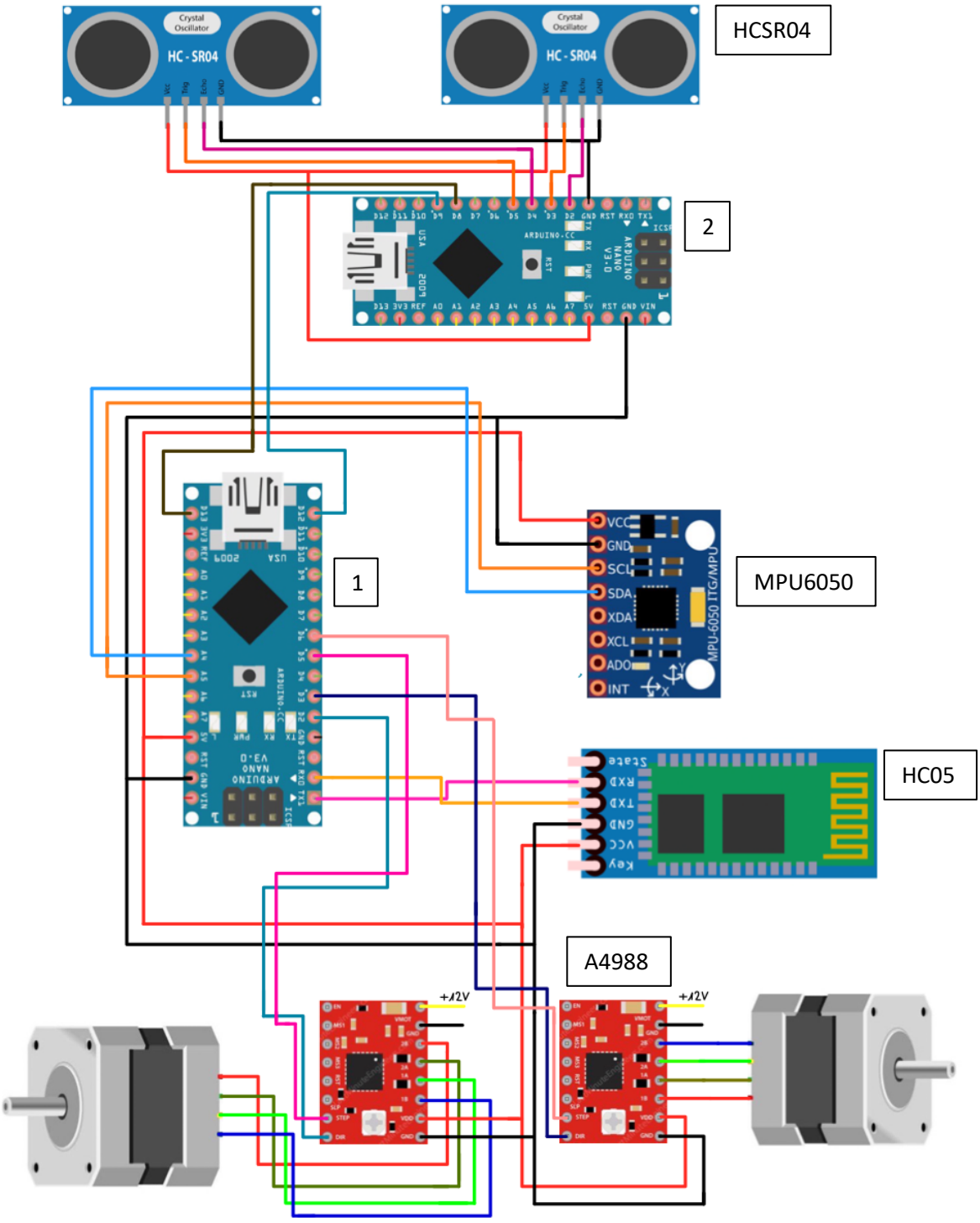

Abbildung 3 Komplette Verschaltung der Komponenten

1. Der Arduino Nano:

Dieser stellt (siehe Abbildung 3, Arduino 1) die Schnittstelle zwischen allen verwendeten Bauteilen dar. Zudem werden auch alle Sensordaten und Rechenschritte auf diesem Bauteil verarbeitet. Auch die für das Balancieren verantwortliche Software wird auf diesen Chip geladen.

Zudem wird die Abstandsmessung der Ultraschallsensoren aus Perfomance-Gründen auf einen eigenen Arduino Nano (siehe Abbildung 3, Arduino 2) ausgelagert. Dieser berechnet anhand der Sensordaten den Abstand zu Gegenständen vor bzw. hinter dem Roboter.

Der verbaute Chip auf dem Arduino Nano ist ein ATmega328P, dieser ist ein solider 8 Bit Prozessor mit einer 16 MHz Clock. Der Chip selbst basiert auf einer modifizierten Harvard Architektur und verwendet einen RISC Befehlssatz (vgl. ATmega328P Data Sheet, S. 1 bzw. 9).

2. MPU6050 Modul:

Das Gyroskop, oder auch MPU6050 Modul genannt, stellt den wichtigsten Sensor des ganzen Aufbaus dar. Da beide Module auf einem 5 V Logic-Level arbeiten, ist kein Logic-Shifter zwischen den anlog Pins des Arduinos und dem SDA bzw. SCL Pin des Gyroskops notwendig. Die Datenübertragung erfolgt über ein I^2C-Protokoll. Hierfür werden die durch die Wire.h festgelegten Pins A4 und A5 für die Datenübertragung verwendet. In dem abgebildeten Fall hier wird für die Serial Clock (= SCL) der analog Pin 5 und für den Serial Data (= SDA) der analog Pin 4 verwendet. Dabei wird über die SCL-A5 Verbindung die festgelegte Übertragungsfrequenz der I^2C-Schnittstelle übertragen. Über die SDA-A4 Verbindung werden anschließend die eigentlichen Messwerte des Gyroskops gesendet.

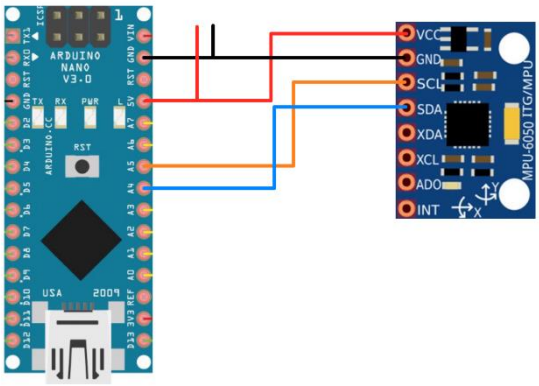

Abbildung 4 Verschaltung MPU6050

3. HC05 wireless Transceiver Modul:

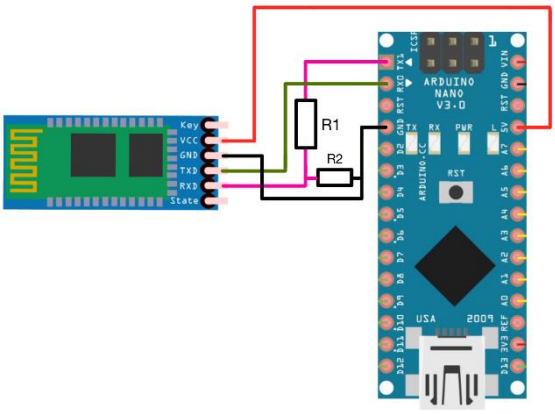

Abbildung 5 Verschaltung HC05

Dieses Bauteil ist nicht zwingend für das Balancieren des Roboters notwendig, da es lediglich eine Schnittstelle für einen Controller darstellt, welcher zum Steuern des Roboters verwendet wird. Wichtig bei der Verschaltung des Bluetooth-Moduls ist, dass dieses auf einem 3,3 V Logic-Level arbeitet. Deswegen ist ein Spannungsteiler zwischen dem Tx-Pin des Arduinos und dem Rx-Pin des HC05 Boards zwingend notwendig. Die Widerstände mit R1 = 1 kΩ und R2 = 2kΩ sind so dimensioniert, dass das 5 V Datensignal des Arduinos auf ein 3,3 V Level reduziert wird. Gleichzeitig ist es in die andere Richtung vom Tx-Pin des HC05 Moduls auf den Rx-Pin des Arduinos nicht notwendig das Logic-Level des Datensignals auf 5 V zu verstärken, da der ATmega328P Chip des Arduinos Signale in Bereich 2,7 – 5,5 V als eindeutiges HIGH-Signal identifizieren kann

(vgl. ATmega328P-Datasheet, Kapitel 28.2). Außerdem wurde dieses Bluetooth Modul des Roboter als Slave konfiguriert, um die Verbindung mit verschiedenen anderen Steuergeräten zu vereinfachen.

4. HC-SR04 Ultraschallsensoren:

Der HC-SR04 Ultraschallsensor ist dafür verantwortlich Hindernisse vor und hinter dem Roboter zu erkennen und falls ein Befehl der Fernsteuerung kommen sollte, trotzdem weiterzufahren, diesen zu verhindern.

In dieser Version werden zwei Ultraschallsensoren verwendet. Diese sind wie folgt mit der zentralen Steuereinheit verbunden:

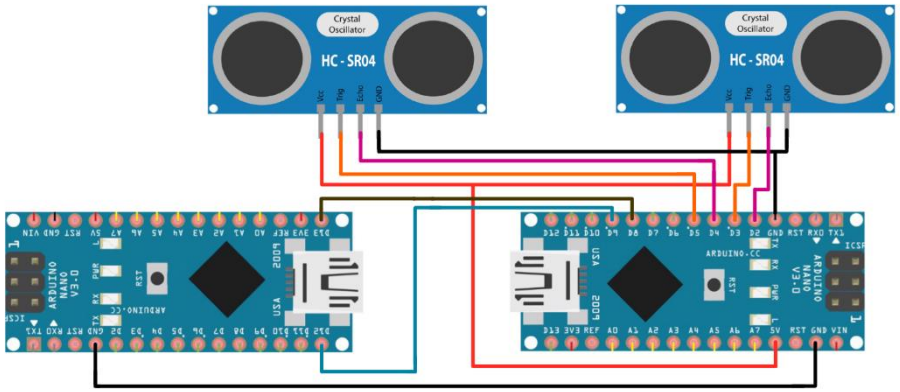

Abbildung 6 Verschaltung HC-SR04

Die Verschaltung mit dem linken Arduino ist dabei sehr simpel. In der Software wurden die digital Pins 4 und 5 für den linken Ultraschallsensor festgelegt (analog für den rechten Ultraschallsensor mit digital Pin 2 und 3). Über den Trigger-Pin (digital Pin 5) wird periodisch ein Ultraschallsignal gesendet. Sobald dieses übertragen wird, geht automatisch der Echo-Pin (digital Pin 4) auf HIGH, bis dieser ein zurückgeworfenes Signal empfängt. Falls sich ein Hindernis vor dem Sensor befindet, setzt sich der Echo-Pin wieder auf LOW, sobald der Empfänger des Sensors ein Signal empfängt. Falls sich kein Hindernis vor dem Sensor befindet, setzt sich der Echo-Pin nach 200 ms wieder auf LOW. Durch die High-Zeitspanne des Echo-Pins kann der Abstand zum Hindernis berechnet werden. Ist dieser kleiner als 20 cm, wird, je nach Richtung, der digital Pin 8 bzw. digital Pin 9 auf HIGH gesetzt. Damit wird der zentralen Steuereinheit mitgeteilt, dass eine Fortbewegung in diese Richtung nicht möglich ist. Wichtig für die Kommunikation zwischen den beiden Arduinos ist, dass sie dasselbe Ground-Level besitzen. Deswegen ist eine Verbindung zwischen beiden Ground-Pins notwendig.

5. Stepper Motor und A4988 H-Brücke:

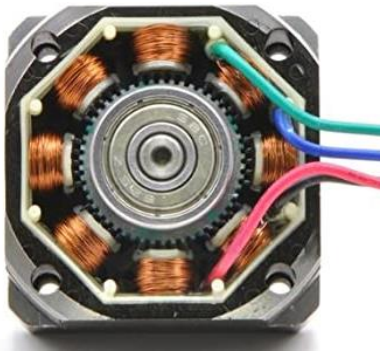

Abbildung 7 Querschnitt Stepper Motor

In dieser Version des balancierenden Roboters werden als Aktorik Elemente Stepper Motoren (dt. Schrittmotor) verwendet. Diese bieten im Vergleich zu herkömmlichen DC-Motoren den Vorteil, dass sie präzise ansteuerbar sind, die Drehzahl besser reguliert werden kann und sie ihren Ausrichtungswinkel halten können. In der Abbildung 7 kann man erkennen, dass ein Stepper Motor mehrere Spulen besitzt. An diese können abhängig von der gewünschten Drehrichtung Spannung angelegt werden, sodass sich ein Magnetfeld, welches in eine bestimmte Richtung zeigt, bildet. Der Rotor zwischen den Spulen enthält einen Permanentmagneten, der sich nach dem Magnetfeld ausrichtet. So kommt es zu einer präzisen Ansteuerung des Motors ohne die Notwendigkeit von Sensoren. In den hier verwendeten Motoren gibt es 8 Spulen (jeweils 4 davon hängen zusammen) welche durch 4 Anschlüssen angesteuert werden. Der Rotor innen besitzt 200 Zähne, wenn man nun einen vollen Schritt ausführt, dreht sich dieser um einen Zahn. Daraus ergibt sich eine Ansteuerung im Normalbetrieb auf 1,8° genau.

Zur Ansteuerung eines Stepper Motor ist somit in der Theorie kein weiters Bauteil notwendig. Allerdings haben die Arduino Pins eine maximale Output-Current von 40 mA (vgl. Arduino Nano Pinout, max DC Current). Der vom Hersteller empfohlene Betriebsstrom der Motoren liegt aber bei 1,7 A (vgl. Nema 17 Technical Specifications, Phase Voltage)! Deswegen wird als weiters Bauteil die A4988 H-Brücke eingebaut. Diese wird wie folgt mit dem Arduino Nano verbunden:

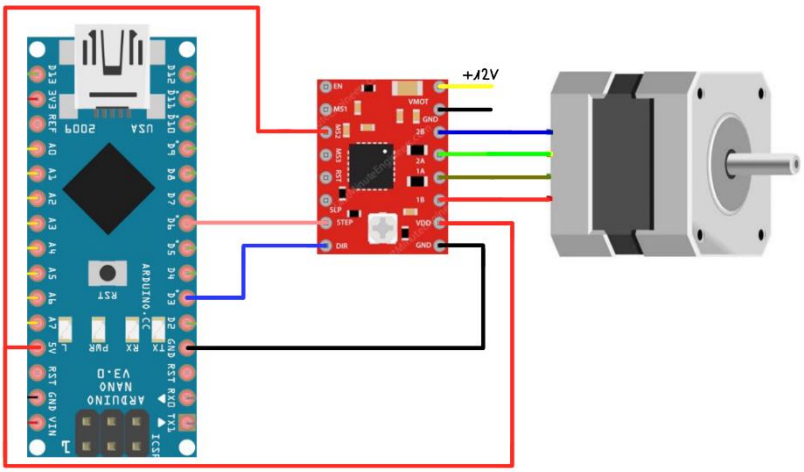

Abbildung 8 Verschaltung A4988

In der obigen Abbildung 8 sind alle Verbindungen für den Betrieb einer A4988 H-Brücke eingezeichnet. Da man in diesem Aufbau zwei Motoren individuell ansteuert, werden auch zwei Stepperdriver benötigt. Neben den digital Pins 3 und 4 sind noch die Pins V_{DD} und GND mit dem Arduino verbunden. Diese stellen die Betriebsspannung des Chips zur Verfügung. Die Pins V_{MOT} und GND sind mit dem Akku verbunden, um die Leistung für den Betrieb der Motoren bereit zu stellen. Um nun einen Motor anzusteuern, legt man den digital Pin 3 auf HIGH bzw. LOW, um die Richtung festzulegen. Anschließend wird der digital Pin 4 auf HIGH gesetzt und unmittelbar wieder auf LOW gezogen. Dies entspricht einem Step-Impuls und bewirkt, dass sich der Motor um einen „Step“ (entspricht, wie zuvor erklärt, einer Drehung um 1,8°) in die bereits festgelegte Richtung dreht.

Zusätzlich zur oben beschriebenen Betriebsart bietet die A4988 H-Brücke noch die Möglichkeit des Microsteppings. Dabei wird über die Pins MS_1, MS_2 und MS_3 der gewünschte Multiplikator der Schrittzahl eingestellt.

Man kann die A4988 H-Brücken wie folgt konfigurieren:

MS1	MS2	MS3	Microstep Resolution	Excitation Mode
L	L	L	Full Step	2 Phase
H	L	L	Half Step	1-2 Phase
L	H	L	Quarter Step	W1-2 Phase
H	H	L	Eighth Step	2W1-2 Phase
H	H	H	Sixteenth Step	4W1-2 Phase

Abbildung 9 Microstepping Tabelle

In dieser Version des Roboters wurde nach einigem „Trial-and-Error“ die Quarter Step Einstellung verwendet. Diese bietet eine Schrittzahl von 800 und damit verbunden eine Ansteuerung auf 0,45° genau, was flüssigere Bewegungen ermöglich und oszillieren um den Nullpunkt des Gyroskops verringert.

Betriebsarten eines Stepper Motors:

Im Full Step Betrieb mit einphasiger Ansteuerung wird jede Spule einzeln angesteuert. Dadurch dreht sich der Rotor innen jedes Mal sofort um einen Step. Ebenso kann man dem Motor bei Full Step Betrieb mit zweiphasiger Ansteuerung verwenden. Dabei werden beide Spulen mit gleich großer Spannung betrieben. Dadurch dreht sich der Rotor innen auf den Winkel zwischen den beiden verwendeten Spulen. Mit der zweiphasige Ansteuerung hat sich die Anzahl der Steps verdoppelt.

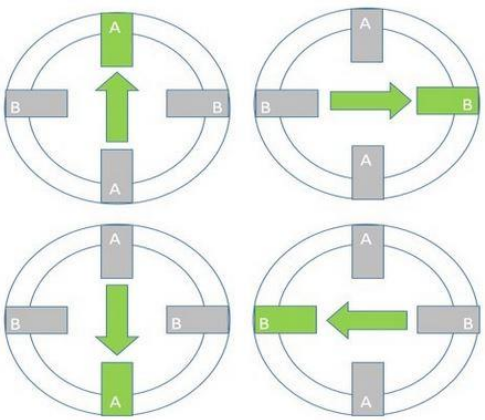

Abbildung 10 Schema einphasige Ansteuerung

Wenn Microstepping aktiviert ist wird die zweiphasigen Ansteuerung so hingehend verändert, dass an den Spulen nicht mehr gleich große Spannungen anliegen, sondern vielmehr zwei sinusförmige Verläufe überlagert werden. Dabei ist der Verlauf der anliegenden Spannung relevant. Zu Beginn muss die Spannung an der Startspule hoch sein und im Verlauf des Schrittes fallen. Im Gegensatz dazu muss die Spannung an der Zielspule steigen. Durch die Überlagerung der beiden Magnetfelder können Winkel zwischen den Spulen angesteuert werden und es kommt insgesamt zu einer flüssigeren Bewegung des Rotors.

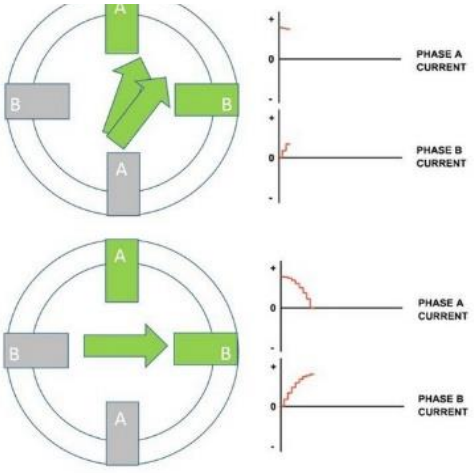

Abbildung 11 Schema Microstepping

Abbildungsverzeichnis und Quellen:

1. Abbildungsverzeichnis:

2. Quellen:

3D-Modell des Gestells: https://github.com/Stromi1011/SelfBalancingRobot_Woita/tree/main/Blenderfiles
ATmega328P Data Sheet:
https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf
A4988 Datasheet:
https://www.pololu.com/file/0J450/a4988_DMOS_microstepping_driver_with_translator.pdf
Arduino Pinout:
https://diyi0t.com/arduino-nano-tutorial/
Nema 17 Technical Specifications:
https://datasheetspdf.com/pdf-file/1310364/Handson/17HS4401S/1
Microstepping Funktionsweise:
https://www.rs-online.com/designspark/stepper-motors-and-drives-what-is-full-step-half-step-and-microstepping-de
Abbildung 9, Microstepping Tabelle:
https://www.pololu.com/file/0J450/a4988_DMOS_microstepping_driver_with_translator.pdf
Abbildung 10, Schema einphasige Ansteuerung:
https://www.rs-online.com/designspark/stepper-motors-and-drives-what-is-full-step-half-step-and-microstepping-de
Abbildung 11, Schema Microstepping:
https://www.rs-online.com/designspark/stepper-motors-and-drives-what-is-full-step-half-step-and-microstepping-de
Schematische Zeichnung Arduino Nano:
https://892962.smushcdn.com/2087382/wp-content/uploads/2019/08/Arduino-Nano-Pinout-1.png?lossy=1&strip=1&webp=1
Schematische Zeichnung MPU6050-Modul:
https://components101.com/asset/sites/default/files/component_pin/MPU6050-Pinout.png
Schematische Zeichnung HC05 Modul:
https://core-electronics.com.au/media/wysiwyg/tutorials/sam/HC-05-diagram-pinout.png
Schematische Zeichnung HCSR04-Modul:
http://docs.iot-bus.com/en/latest/_images/HC-SR04-1.jpg
Schematische Zeichnung A4988-Modul:
https://lastminuteengineers.com/wp-content/uploads/arduino/A4988-Stepper-Motor-Driver-Pinout.png
Schematische Zeichnung Stepper Motor:
https://www.makerguides.com/wp-content/uploads/2019/02/A4988-Arduino-stepper-motor-wiring-schematic-diagram-pinout.jpg