Ein Bild, das Diagramm enthält.

Automatisch generierte BeschreibungEinheit 3: Motoren

Materialien

* 1x Arduino Nano
* 2x Gleichstrommotor
* 1x H-Brücke L298N
* 1x Breadboard (mittelgroß)
* Jumperkabel männlich/männlich

Abbildung 1 Materialien

Ein Gleichstrommotor, dient zum Antrieb von vielen verschiedenen Geräten und Modulen. In unserem Fall wird er simpel das Rad antreiben, welches den Roboter fortbewegt. Der Motor besitzt 2 Anschlüsse (+ und –), die für die Stromzufuhr zuständig sind.

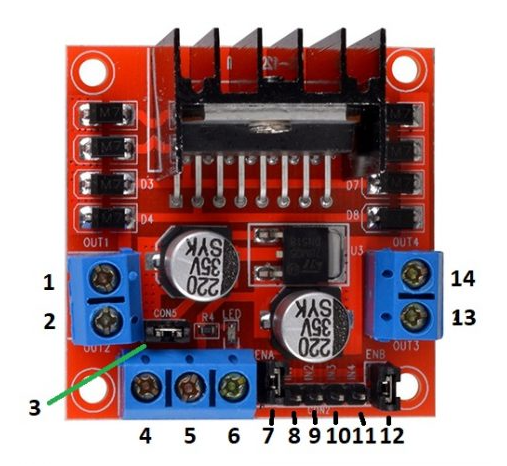
Der Motor wird zur korrekten Anwendung an eine sogenannte H-Brücke angeschlossen, diese reguliert durch die vier Schalter (S1, S2, S3, S4) die Stromrichtung und damit die Drehrichtung und auch die Geschwindigkeit des Motors. Des Weiteren besitzt die H-Brücke eine höhere Strombelastbarkeit als der Arduino und ist somit Antriebsstärker, also perfekt für unseren Anwendungszweck.

Ein Bild, das Diagramm enthält.

Automatisch generierte BeschreibungDie Kabel des Motors, werden an die gleich gekennzeichneten Pins der H-Brücke angeschlossen.

Abbildung 2 Verkabelung Motor/H-Brücke

Die H-Brücke

Damit die Motoren funktionstüchtig an die H-Brücke angeschlossen werden können, muss diese natürlich mit dem Arduino Nano verknüpft sein.

Für eine korrekte Funktionsweise werden im Folgenden die Pins und ihre Funktion angegeben.



Abbildung 3 Pin-out H-Brücke

**Nützliche Pins der H-Brücke**

|  |  |
| --- | --- |
| Pin | Funktion |
| 1 | Anschluss für den Gleichstrommotor 1 „+“ |
| 2 | Anschluss für den Gleichstrommotor 1 „-“ |
| 3 | Jumper |
| 4 | Eingang für die Spannungsversorgung der H-Brücke (max. 35V Gleichstrom) |
| 5 | Eingang für den GND |
| 6 | 5V Ausgang, z.B. für die Stromversorgung des Mikrocontrollers |
| 7 | Pin, über den der Gleichstrommotor 1 angesteuert wird. Falls dieser Pin an einem PWM-Pin des Mikrocontrollers angeschlossen wurde, kann die Motorengeschwindigkeit ebenfalls genau bestimmt werden |
| 8 | Pin IN1 – Steuert Gleichstrommotor 1 |
| 9 | Pin IN2 – Steuert Gleichstrommotor 1 |
| 10 | Pin IN3 – Steuert Gleichstrommotor 2 |
| 11 | Pin IN4 – Steuert Gleichstrommotor 2 |
| 12 | Pin, über den der Gleichstrommotor 2 angesteuert wird. Falls dieser Pin an einem PWM-Pin des Mikrocontrollers angeschlossen wurde, kann die Motorengeschwindigkeit ebenfalls genau bestimmt werden |
| 13 | Anschluss für den Gleichstrommotor 2 „+“ |
| 14 | Anschluss für den Gleichstrommotor 2 „-“ |

A picture containing graphical user interface

Description automatically generatedNutze zuerst 6 Kabel und stecke jeweils einen Kopf in die Pins 7-12 der H-Brücke (siehe „Die H-Brücke“).

Die anderen Enden der Kabel werden, ohne sich zu überkreuzen an die Pins des Arduinos D11-D6 gesteckt.

Icon

Description automatically generated

Abbildung 4 Verkabelung Digital Anschlüsse

**Aufgabe 1**

Die Motoren sollen beim Testen der Funktionen die Rotationsrichtung, sowie die Geschwindigkeit ändern können! Pins, beachten.

|  |  |
| --- | --- |
| **1** | **Platziere den Arduino wie bekannt auf dem Breadboard und schließe dann die H-Brücke an den Arduino an. Beachte die Pins 7 und 12 an PWM-Pins des Arduinos anzuschließen.** |

Motoren an die H-Brücke anschließen

Wir nehmen die Pins 1,2 und 13,14 für die Kabel der Motoren.

**Beachte,**

dass die mit „+“ und „-“ gekennzeichneten Pins der H-Brücke mit den gleichnamigen Kabeln der Motoren verbunden werden und der „+“ Pin auf der einen Seite oben und der „-“Pin auf der anderen oben ist!

Da die die Motoren eine höhere Spannung benötigen als der Arduino geben kann, benötigt die H-Brücke eine zusätzliche Spannungsquelle. Also schließen wir an die vorgesehenen Pins (Eingang, GND) der H-Brücke eine 9V Batterie an.

**Aufgabe 2**

|  |  |
| --- | --- |
| **2** | **Verbinde die Motoren mit den richtigen Pins der H-Brücke. Schließe eine 9V Batterie an die H-Brücke als zusätzliche Spannungsquelle an.** |

Motoren initialisieren

Die Motoren werden über die zwischengeschaltete H-Brücke angesteuert und besitzen dafür drei Anschlüsse. Den Geschwindigkeits-Anschluss und jeweils zwei Richtungs-Anschlüsse. Da das Merken der Pin-Nummern und ständige schreiben der Zahlen schnell zu Fehlern kommen kann, speichern wir die Pin-Nummern von Modulen ab sofort als Variablen ab. Der Effizienz wegen werden die Variablen als const deklariert. Der Datentyp ist ein int.

Da bei den Motoren sämtliche Anschlüsse digital sind, müssen wir alle sechs Pins in der setup() als OUTPUT definieren.

**Aufgabe 3**

|  |  |
| --- | --- |
| **3** | **Code schreiben**   * **Erstelle konstante Variablen, die die Pin-Nummern abspeichern. Überlege dir eine gute Bezeichnung, damit der Code einfacher zu lesen ist.** * **Definiere alle Anschlüsse in der setup() als OUTPUTS** |

Motoren ansteuern

Damit die Motoren angesteuert werden können, müssen wir sowohl die Geschwindigkeit als auch die Richtung angeben. Die Geschwindigkeit geben wir an, indem wir dem Geschwindigkeits-Pin mit dem neuen Befehl analogWrite() eine Geschwindigkeit vorgeben, dabei ist 0 Stillstand und 255 Vollgas. Die Richtung bestimmen wir über die zwei Richtungs-Pins. Diese müssen auf gegensätzliche HIGH/LOW Werte gesetzt werden. Also Richtungs-Pin 1 wird auf HIGH gesetzt und Richtungs-Pin 2 auf LOW ist vorwärts, während andersherum rückwärtsfahren bedeutet. Wenn beide Pins auf LOW gesetzt werden, ist das der Stillstand Modus und der Motor hört auf sich zu drehen. Dazu nutzen wir den neuen Befehl digitalWrite(). Ein dritter nützlicher Befehl ist der Warte-Befehl delay(). Damit kann man das Programm für kurze Zeit pausieren

|  |  |
| --- | --- |
| **delay(int);** |  |
| **Beschreibung** | Mit diesem Befehl kann man die Ausführung des nachfolgenden Codes für die angegebene Zeit pausieren. Gesendete Signale bleiben aktiv. |
| **Parameter** | int: Die Anzahl an Millisekunden, die das Programm pausiert werden soll (1000ms = 1s) |

|  |  |
| --- | --- |
| **digitalWrite(pin, HIGH/LOW);** | |
| **Beschreibung** | Mit diesem Befehl kann man auf dem angegebenen Pin digitale Signale (HIGH oder LOW) senden. |
| **Parameter** | pin: Nummer des Pins, auf der das Signal gesendet werden soll  HIGH/LOW: die Auswahl des Signals, das gesendet werden soll, entweder ein HIGH oder ein LOW-Signal. |
| **Rückgabewert** | void |
| **analogWrite(pin, int);** | |
| **Beschreibung** | Mit diesem Befehl kann man auf dem angegebenen Pin analoge Signale (0-255) senden. |
| **Parameter** | pin: Nummer des Pins, auf der das Signal gesendet werden soll  int: Die Ganzzahl, die als Signal gesendet werden soll, bei Nutzung von Variablen, muss der Datentyp int sein. |
| **Rückgabewert** | void |

**Aufgabe 4**

|  |  |
| --- | --- |
| **4** | **Code erweitern**   * **Definiere eine enum, die die Richtung abspeichern kann (vorwärts, rückwärts oder stillstand).** * **Erstelle zwei Hilfsfunktionen, eine jeweils für jeden Motor, die eine Richtung übergeben bekommt und die jeweiligen Richtungs-Pin korrekt ansteuert.** * **Lass den ersten Motor vorwärts rotieren mit der Geschwindigkeit von 200 für zwei Sekunden rotieren.** * **Lass danach den zweiten Motor zwei Sekunden rückwärts rotieren mit einer Geschwindigkeit von 150.** * **Versetze beide Motoren in Stillstand für zwei Sekunden (es reicht die Richtungs-Pins auf Stillstand zu stellen).** |