VII. Arduinos im Robo-Camp
Ausführliche Beschreibung eines Ablaufs zum Einsatz von Arduinos

Während sich die Ausführungen in Kapitel 6 dieses Bandes an erfahrene Lehrkräfte richten, die einen offenen und projektorientierten Unterrichtsansatz wählen, findet der Leser bzw. die Leserin in diesem Kapitel eine sehr detaillierte Anleitung, wie man Arduinos einsetzen kann, worauf man achten sollte und welches inhaltliche Wissen vermittelt werden kann. Diese Anregungen sind am Anfang absichtlich kleinschrittig gestaltetet, damit auch mit der Thematik unerfahrene Personen sich vorstellen können, eine Anleitung dazu zu geben. Gegen Ende des Kapitels ist die Arbeit an dem Projekt hingegen deutlich weniger anleitend, weil zu dem Zeitpunkt alle Einzelaspekte bereits thematisiert worden sind und zusammenhängend angewendet werden sollen. Die Anregungen selbst können natürlich immer noch variabel eingesetzt werden, je nach Vorwissen und Interesse der Lernenden. Alle Aufgaben in diesem Kapitel sind an Lehrende gerichtet, die diese als Impulse als Lernende weitergeben können. Der vorliegende Ablauf ist in mehrere Schritte und Phasen gegliedert, von denen der bzw. die Lehrende frei wählen kann, ob alles aufgegriffen werden soll oder nur die Teile Sinn machen, die am besten zum aktuellen Lernniveau der Lernenden passen. Ebenso ist es natürlich möglich, auf Grundlage des bereitgestellten Materials eigene Sequenzen zu erstellen. Das vorliegende Material eignet sich ggf. auch dazu, Lernenden außerhalb des Unterrichts (als Ferienbeschäftigung oder im HomeSchooling) Freude am Programmieren zu vermitteln.

Das Hauptthema des vorliegenden Materials sind Roboter und Automatisierung. Der Schwerpunkt liegt darauf, den Lernenden ein Verständnis dazu näherzubringen, was ein Roboter ist und was die Gesellschaft manchmal in das Gerät hineininterpretiert. Gleichzeitig sollen Grundkenntnisse in Robotik vermittelt werden, darunter ein Verständnis von Mikrocontrollern, Sensoren und Aktoren. Ziel ist es, die Lernenden mit Fähigkeiten und Wissen auszustatten, damit sie mit dem Arduino-System ein Technologieverständnis entwickeln, um anschließend ihre eigenen Roboter entwickeln und konkrete Probleme lösen zu können.

Das Material wurde im Zusammenhang mit dem Lehr- und Lernangebot des PANaMA-Projekts in Form von Forschungscamps, wie sie u.a. im Herbst 2019 an der Kieler Forschungswerkstatt stattgefunden haben (vgl. Kapitel IV), entwickelt und umgesetzt. Das Camp war nicht in klassische Unterrichtsmodule unterteilt, und die Zeitspannen für die einzelnen Phasen sind daher nicht scharf in Module zu je 45 Minuten unterteilt. Das vollständige entwickelte Material, von dem Teile in diesem Kapitel vorgestellt werden, finden Sie unter dem folgenden Link (bitte beachten Sie, dass das Material derzeit nur in englischer Sprache vorliegt): http://www.teknologiskolen.dk/robocamp2019/.

Anmerkung: Das Material basiert auf dem Arduino-Uno-Mikrocontroller sowie den Sensoren und Aktoren, die im Forschungscamp eingesetzt wurden. Dies bedeutet, dass die angegebenen Einheiten ebenfalls hierauf basieren, und ein digitales Signal daher in diesem Material beispielsweise als zwischen 0 V (LOW) und 5 V (HIGH) liegend beschrieben wird. Die konkreten Werte der Spannung sind dabei

plattformabhängig; so arbeiten beispielsweise sowohl micro: bit als auch Calliope Mini mit einer Spannung zwischen 0 V (LOW) und 3,3 V (HIGH) bei Anschluss über ein USB-Kabel; bei Nutzung einer 3-V-Batterie als Spannungsquelle sind es 0 V (LOW) und 3 V (HIGH).

Einführung in das Thema Roboter – Was ist ein Roboter? (70 Min.)

Kurz zum Inhalt: Der Unterricht erfolgt an dieser Stelle überwiegend über Input der bzw. des Lehrenden, wobei der Dialog mit den Lernenden zu den vorgestellten Aspekten im Vordergrund steht. Der Schwerpunkt liegt auf einem gemeinsamen Verständnis dessen, was ein Roboter ist bzw. nicht ist und aus welchen Kernkomponenten er besteht sowie dem Zusammenspiel der Komponenten und Beziehungen zueinander.

Lernergebnisse: Die Lernenden nennen eine mögliche Definition eines Roboters. Sie können identifizieren und analysieren, welche Objekte als Roboter klassifiziert werden können und warum. Darüber hinaus erhalten sie Einblick in die internen Kernkomponenten eines Roboters, welche Rolle sie im Roboter spielen und welche Interaktion zwischen ihnen besteht. Infolgedessen werden zudem mögliche Missverständnisse in Bezug auf Roboter aufgrund der Darstellung in modernen Medien angesprochen, ausgeräumt und ggf. entmystifiziert.

Phase 1 – Einführung in das Thema (30 Min.):

Erklären Sie den Lernenden, dass zunächst behandelt werden muss, was ein Roboter ist, bevor sie mit ihren eigenen Robotern arbeiten und sie herstellen können. Beginnen Sie dann mit der Erörterung darüber, was ein Roboter ist, indem Sie die Lernenden fragen, was sie unter dem Wort Roboter verstehen und was einen solchen ausmacht.

Den Lernenden werden Bilder gezeigt (z. B. humanoide Roboter, nicht humanoide Roboter, elektrische Maschinen und nicht-elektrischen Gerätschaften wie z. B. ein Fahrrad), zu denen die Lernenden für jedes Bild gefragt werden, ob das Objekt auf dem Bild ein Roboter ist oder nicht, und womit die Antwort begründet wird.

Danach kann die offizielle Definition bzw. das Fehlen derselben ins Spiel gebracht werden – denn es gibt keine endgültige Definition, was ein Roboter ist, obschon Einvernehmen hinsichtlich einiger diesbezüglicher Parameter besteht. Eine mögliche Definition könnte daher so lauten:

„Ein Roboter ist eine programmierbare Maschine, die autonom funktionieren, ihre Umgebung wahrnehmen und ihre Aktionen entsprechend anpassen kann".

Sprechen Sie mit den Lernenden darüber, welche Aspekte ein Roboter leisten muss, um diese Definition zu erfüllen:

- die Welt wahrnehmen
- in der Welt agieren
- Entscheidungen treffen, d. h. auf Basis des Wahrgenommenen die korrekte Aktion ausführen.

Wenn diese drei Anforderungen erfüllt sind, kann das Gespräch zu einem Diskurs darüber werden, wie Roboter die oben genannten Anforderungen erfüllen können, wobei die Lernenden über folgende Impulse zunächst ihre eigene Sinneswahrnehmung und -interpretation hinterfragen und mit denen des Roboters kontrastieren:

Die Welt wahrnehmen:

- Wie nehmen **Menschen** die Welt wahr? (Sehen, hören, schmecken, riechen, fühlen)
- Wie kann **ein Roboter** die Welt wahrnehmen? (benötigt mindestens einen Sensor, der z. B. lichtempfindlich ist)

In der Welt agieren:

- Wie agieren **Menschen** in der Welt? (Muskeln)
- Wie kann **ein Roboter** agieren? (benötigt mindestens einen Aktor, z. B. einen DC-Motor)

Eine Entscheidung treffen:

- Wie treffen **Menschen** Entscheidungen? (Gehirn)
- Wie kann **ein Roboter** entscheiden? (benötigt mindestens einen Mikrocontroller, z. B. einen Arduino)

Damit haben die Lernenden die drei Kernkomponenten identifiziert, die nach der o.g. Definition einen Roboter ausmachen, und die jeweils mindestens einmal vorhanden sein muss: Sensor, Aktor und Mikrocontroller.

Mögliche Definition eines Roboters: *„Ein Roboter ist eine programmierbare Maschine, die autonom funktionieren, ihre Umgebung wahrnehmen und ihre Aktionen entsprechend anpassen kann".*

Kernkomponenten: Mikrocontroller, Sensoren und Aktoren.

Phase 2 – Die Kernkomponenten, ihre Funktion und Interaktion (30 Min.):

Nach der Klärung der drei Kernkomponenten können diese näher beleuchtet werden, um zu erklären, was sie tun und wie sie interagieren. Da der Mikrocontroller sowohl die Sensoren als auch die Aktoren steuert, ist er ein guter Ausgangspunkt.

Was ist ein Mikrocontroller?

En Mikrocontroller ist vom Grundsatz her ein kleiner Computer. Er hat einen Prozessor (Rechenleistung), RAM (Speicher) und andere notwendige Komponenten, genau wie ein Computer. Mikrocontroller gibt es heute in fast allen Arten von elektrischen Geräten: im Getränkeautomaten, im Fernseher, in der Mikrowelle, im Wecker usw.

Der Mikrocontroller verfügt über mehrere Ein- und Ausgänge, die sogenannten I/O-Pins (kurz für „Input/Output" also Eingabe/Ausgabe). Die I/O-Pins des Mikrocontrollers können so programmiert werden, dass sie entweder die an sich selbst anliegende Spannung messen oder andere Komponenten mit Spannung versorgen.

Oft wird der Mikrocontroller verwendet, um abzulesen, mit welcher Spannung ein Sensor einen bestimmten I/O-Pin versorgt, um darauf basierend eine Auswahl zu treffen und einen Aktor mit einer Spannung zu versorgen, die wiederum eine beabsichtigte Aktion auslöst.

Das Signal, das der Mikrocontroller von einem Sensor empfängt, heißt INPUT, das Signal, das er an einen Aktor sendet, heißt OUTPUT. INPUT-Signale können entweder digital (als LOW (0 V) oder HIGH (5 V)) oder analog (von 0 (0 V) bis 1023 (5 V)) gelesen werden, während OUTPUT-Signale digital als (LOW (0 V) oder HIGH (5 V)) geschrieben werden können[1].

Mikrocontroller: Ein kleiner Computer mit programmierbaren I/O-Pins (Input/Output = Eingabe/Ausgabe), z. B. ein Arduino Uno. Die Rolle des Mikrocontrollers besteht darin, das Programm auszuführen, das zuvor programmiert worden ist, und ist sozusagen das "Gehirn" des Roboters.

Der Mikrocontroller wird häufig so programmiert, dass er ein analoges / digitales INPUT-Signal (0 V – 5 V) empfängt, das Signal verarbeitet und eine Entscheidung darüber trifft, wie darauf reagiert werden soll, woraufhin ein digitales-OUTPUT-Signal (0 V – 5 V) einen angeschlossenen Aktor versorgt, der die Aktion ausführt.

Digitale Signale (Input und Output**):** Digitale Signale sind entweder 0 V oder 5 V.

Analoge Signale (Input**):** Analoge Signale haben einen Wert zwischen 0 V und 5 V und werden vom Arduino als Zahl zwischen 0 (0 V) und 1023 (5 V) gelesen.

PWM-Signale (Output): PWM-Signale (Pulse-width Modulation) sind eine Simulation von analogen Signalen und werden von einem I/O-Pin in einem benutzerdefinierten Bereich erzeugt, der abwechselnd zwischen 0 V und 5 V oszilliert.

Was ist ein Sensor?

Hier bietet es sich an, zunächst die Lernenden zu fragen, welche Beispiele für Sensoren sie bereits kennen. Anschließend kann dies mit einer Beispielliste abgeglichen werden, die folgende Sensoren beinhalten kann: Schalter, Lichtsensoren (LDR für Light Dependent Resistor), Abstandssensoren (Ultraschallsensor), Farbsensoren, Bewegungssensoren, Mikrofone, Temperatursensoren, Infrarotsensoren (IR-Sensor)) usw.

Allen diesen Sensoren gemein ist, dass sie Änderungen in der Umgebung erkennen, in der sie sich befinden. Bei der Gelegenheit kann auch thematisiert werden, dass Roboter mit bestimmten Sensoren Dinge wahrnehmen können, die wir nicht erkennen können, so z.B. Ultraschall oder Infrarotlicht.

Daran schließt sich ein kurzer Exkurs über die allgemeine Verwendung von Sensoren an: Einige Sensoren senden ständig ein Signal an den Mikrocontroller (z. B. LDR) zurück, während andere erst aktiviert

[1] oder auch als Pulsweitenmodulation (Pulse Width Modulation oder PWM), PWM-Signale sind eine Simulation von analogen Signalen und werden erzeugt, indem ein I/O-Pin in einem angepassten Intervall zwischen 0 V und 5 V schwingungsalterniert. Die Schwingungen erfolgen so schnell, dass das Signal in der Praxis als analoges Signal verwendet werden kann. Im Arduino wird ein PWM-Signal als Zahl zwischen 0 (0 V) und 255 (5 V) geschrieben.

werden müssen, bevor ein Signal zurückgesendet wird (z. B. ein Abstandssensor). Gemeinsam haben sie indessen, dass der Mikrocontroller dieses Signal als INPUT empfängt, das dann gelesen und interpretiert werden kann.

Gleichzeitig wird die überwiegende Mehrheit der Sensoren, mit denen die Lernenden arbeiten, ein analoges Signal von 0 V – 5 V zurückgeben, das vom 10-Bit-ADC (Analog-Digital-Wandler) des Mikrocontrollers in eine Zahl von 0 bis 1023 (LDR, Temperatur, Potentiometer, IR-Sensor usw.) übersetzt wird, während der Abstandssensor die Zeit misst, die vergeht, wenn ein INPUT-Signal von HIGH (5 V) auf LOW (0 V) wechselt.

Zum Abschluss der Einführung in die Sensoren können Bilder verschiedener Exemplare gezeigt werden, mit denen die Lernenden im weiteren Verlauf arbeiten werden, damit sie sich bereits jetzt visuell mit deren Aussehen vertraut machen können.

Sensor: Der Sensor erkennt Änderungen in der Umgebung, in der er sich befindet, wie z. B. Änderungen des Lichtpegels (LDR, IR-Sensor), des Schallpegels (Mikrofon), der Entfernung des Sensors zum nächsten Objekt (Ultraschallsensor) usw.

Signal: Der Mikrocontroller empfängt den Sensorwert als ein INPUT-Signal, welches entweder analog oder digital gelesen wird..

Was ist ein Aktor?

Wie bei den Sensoren können die Lernenden auch hier gefragt werden, welche Beispiele sie schon kennen; die Rückmeldung kann mit einer Beispielliste abgeglichen werden, die folgende Objekte enthalten kann: Motoren (DC, Servo, Stepper), LEDs (Leuchtdioden), Lautsprecher, Heizelemente usw.

Danach kann das gemeinsame Merkmal all dieser Objekte identifiziert werden: Sie beeinflussen die Umgebung, in der sie sich befinden. Auch hier kann darauf eingegangen werden, dass beispielsweise Infrarot-LEDs und Ultraschalllautsprecher Aktionen ausführen können, die Menschen nicht ausführen können.

Anders als Sensoren, die ein Signal an den Mikrocontroller senden, den dieser als INPUT empfängt, sendet der Mikrocontroller hier einen OUTPUT, das der Aktor empfängt. Bei den in diesem Material genutzten Aktoren ist das verwendete OUTPUT-Signal entweder digital (LOW (0 V) oder HIGH (5 V)) oder PWM (Pulsweitenmodulation, von 0 (0 V) bis 255 (5 V)).

Aktor: Der Aktor beeinflusst die Umgebung, in der er sich befindet, sei es durch physisch bewegliche Teile wie Motoren (DC, Servo, Stepper) oder auch durch Änderungen des Lichtpegels (LED), des Schallpegels (Summer, Lautsprecher), der Temperatur (Heizelemente) usw.

Signal: Der Mikrocontroller sendet den Aktorwert als ein OUTPUT-Signal.

Zusammenfassung

Fassen Sie Nachstehendes zusammen und gehen Sie mit den Lernenden die Bilderserie von vorhin noch einmal durch. Besprechen Sie noch einmal, ob die Objekte auf den Bildern Roboter sind oder nicht, bzw. die Anforderungen an einen Roboter erfüllen. Können die Lernenden die drei Kernkomponenten identifizieren (unter Beachtung des Umstandes, dass der Mikrocontroller häufig verborgen ist, somit lediglich davon ausgegangen werden muss, dass er vorhanden ist)? Die Zusammenfassung kann vorteilhafterweise im Prozessverlauf mehrmals wiederholt werden – auch um die Lernenden in der Identifizierung und Analyse von Robotern zu schulen und die (Kenntnisse der) Interaktion zwischen Mikrocontrollern, Sensoren und Aktoren aufzufrischen. Verwenden Sie daher gern jedes Mal neue Bilderserien, aber auch Gegenstände aus dem Alltag der Lernenden.

Definition:
„Ein Roboter ist eine programmierbare Maschine, die autonom funktionieren und ihr Verhalten den Umgebungen anpassen kann."

Anders ausgedrückt: Ein Roboter kann seine Umgebung wahrnehmen, auf Basis dessen eine vorprogrammierte Entscheidung treffen und entsprechend reagieren.

Kernkomponenten:

- ein Mikrocontroller, der Entscheidungen trifft.
- ein Sensor, der Veränderungen in seiner Umgebung erfasst.
- ein Aktor, der Veränderungen seiner Umgebung herbeiführt.

Übersicht:

	Mikrocontroller:	**Sensoren:**	**Aktoren:**
Eigenschaft:	Treffen Entscheidungen, 'hören auf' Sensoren und 'befehligen' Aktoren.	Registrieren Veränderungen in ihrer Umgebung.	Verändern ihre Umgebung.
Signal:	Empfangen ein INPUT-Signal von Sensoren. Senden ein OUTPUT-Signal an Aktoren.	Senden ein INPUT-Signal an den Mikrocontroller.	Empfangen ein OUTPUT-Signal vom Mikrocontroller.

Phase 3 – Was ein Roboter nicht ist (10 Min.):

Es gibt jetzt eine für den weiteren Verlauf geltende Definition, welche Anforderungen ein Objekt erfüllen muss, bevor es als Roboter bezeichnet werden kann. Gleichzeitig haben die Lernenden jetzt einen kurzen Überblick darüber erhalten, wie die entsprechenden Kernkomponenten funktionieren und in Beziehung zueinander stehen. Aber es ist oft (mindestens) genauso wichtig zu wissen, was ein Roboter *nicht* kann. Dies ist zum Beispiel sehr relevant, wenn Menschen mit einem Roboter zusammenarbeiten, etwa im Bereich der Industrie.

Fragen Sie die Lernenden , ob sie an dieser Stelle auf Basis ihrer Erfahrungen Beispiele dazu anführen können, was Roboter nicht sind bzw. nicht können – vor dem Hintergrund, dass Roboter ja durchaus, wie wir Menschen auch, die Welt wahrnehmen, in der Welt agieren und Entscheidungen treffen können

Ein gutes Thema wäre z. B., dass Roboter keine 'Gefühle' vergleichbar mit denen von Menschen haben. Eine mögliche Idee wäre, die Lernenden zunächst zwei Roboter kennenlernen zu lassen, einen 'niedlichen' à la NAO, PARO usw., und einen Industrieroboter, z. B. ein Schweißgerät. Anschließend können die Lernenden gefragt werden, welcher der beiden Roboter wohl die größten empathischen Gefühle haben kann. Natürlich ist keiner der Roboter dazu in der Lage, aber die Lernenden werden im Laufe ihres Lebens mit Robotern konfrontiert, die darauf ausgelegt sind, empathische Gefühle vorzugaukeln. Auch werden ihnen durch die modernen Medien Roboter begegnen, die scheinbar zu echten Gefühlen fähig sind.

Gleichzeitig ist es wichtig, die Lernenden darauf hinzuweisen, dass Roboter im klassischen Sinne nicht selbstbewusst und nicht intelligent sind – unabhängig davon, was Filmindustrie und andere Medien das Publikum glauben machen wollen. Sie agieren und arbeiten ausschließlich aufgrund logischer Schaltkreise, denn alle ihre Entscheidungen basieren ausschließlich auf mathematischen Berechnungen, welche sie allerdings rasend schnell auszuführen in der Lage sind. Und aus genau diesem Grund sollte genügend Zeit der Feststellung gewidmet werden, das Menschen bei Durchführung all dieser Berechnungen mit Stift und Papier zu genau der gleichen Entscheidung kommen würden wie der Roboter.

Aus dem gleichen Grund sind Roboter auch weder reflektierend, intuitiv noch kreativ. Sie tun genau das, wofür sie programmiert worden sind und nichts anderes. Zum Beispiel kann der Roboter keinen schlechten Tag haben oder 'mürrisch drauf' oder unaufmerksam sein. Als Beispiel kann ein Roboter dienen, der so programmiert ist, dass er geradeaus fährt. Was passiert, wenn man diesen Roboter auf einen Tisch stellt? Er fährt ohne zu zögern direkt über die Tischkante hinaus, ohne zu irgendeinem Zeitpunkt eine Meinung über diese Handlungsweise gehabt zu haben oder die Folgen seiner Handlung zu bedenken. Er stoppt auch nicht intuitiv vorher ab, wie ein Mensch das tun würde. Der zu Beginn angesprochene Roboter in der Industrie denkt auch nicht an seinen menschlichen Kollegen oder nimmt keine Rücksicht auf Dinge, die er gemäß Programmierung nicht berücksichtigen soll.

Roboter sind **nicht** intelligent im klassischen Sinne, nicht intuitiv oder reflektierend und sich ihrer eigenen Existenz weder bewusst noch in der Lage, Gefühle zu haben.

Roboter treffen Entscheidungen, die auf Mathematik und Logik beruhen. Wenn Menschen genügend Zeit einplanen, werden sie mit Papier und Bleistift nach dem vorgegebenen Entscheidungsbaum immer die gleiche Entscheidung treffen wie der Roboter.

Einführung in das Thema Roboter – Warum es so schwer ist, sie herzustellen (30 Min.)

Kurz zum Inhalt: Der Schwerpunkt dieses Abschnitts liegt auf einer körperlichen Aktivität, bei der Lernende in Dreiergruppen interagieren und sich in die Rolle eines Roboters begeben sollen, indem sie seine drei Kernkomponenten simulieren: den Mikrocontroller, den Sensor und den Aktor.

Lernergebnisse: Durch körperliche Aktivität erfahren die Lernenden die grundlegenden internen Kommunikationskanäle, aus denen die Interaktion zwischen Mikrocontroller, Sensor und Aktor (INPUT- und OUTPUT-Signal) besteht. Gleichzeitig lernen sie die Einschränkungen von Robotern kennen, die im Gegensatz zu der oft intuitiven Verhaltens- und Vorgehensweise von Personen bei neuen Rahmenbedingungen immer einem vorgegebenen Schema folgen.

Physisches Material je Gruppe (3 Lernende je Gruppe): 2 Augenbinden, 3 einfarbige Kunststoffbecher o. ä.

Phase 1 – Einführung, Regeln, Aufbau (Setup), Herausforderungen und Ziele (12 Min.):

Diskutieren Sie mit den Lernenden, warum es so schwierig ist, Roboter herzustellen, obwohl einige Gemeinsamkeiten mit Menschen bestehen (Gehirn: Mikrocontroller, Sinne: Sensoren, Muskeln: Aktoren). Hier spielt insbesondere die Erörterung in der vorangegangenen Phase darüber, was Roboter *nicht* sind, eine zentrale Rolle. Man denke nur daran, wie oft Personen z. B. intuitiv handeln, wenn sie einfach mal einen Waldspaziergang machen.

Fragen Sie auch die Lernenden, welcher Roboter ihrer Meinung nach am einfachsten herzustellen ist: einer, der in einer kontrollierten Umgebung tausende Sachen ausführen kann, oder einer, der in einer unkontrollierten Umgebung eine und nur diese eine Aufgabe ausführen kann? Einfacher ist natürlich die kontrollierte Umgebung, da hier die relevanten Variablen bekannt sind – weshalb der Mangel an Intelligenz, Reflexionsfähigkeit und Intuition beim Roboter nicht problematisch ist.

Gleichzeitig wurde die menschliche Kommunikation zwischen Gehirn, Sinnen und Muskeln durch Millionen von Jahren natürlicher Evolution ständig weiter verfeinert, weshalb sie so sehr in unser System eingebettet ist, dass nicht mehr aktiv darüber nachgedacht werden muss, wie Inputs der eignen Sinne interpretiert werden sollen oder Outputs an die eigenen Muskeln formuliert werden müssen.

Nach der Einführung in das Problemfeld werden die Lernenden in die Regeln, den Aufbau sowie die Herausforderungen und Ziele der Aktivität eingeführt (siehe unten):

Bei Lebewesen sind das Gehirn, die Sinne und Muskeln über Millionen Jahre natürlicher Evolution genau aufeinander abgestimmt. Mit Robotern ist das nicht so. Gleichzeitig sind Roboter im klassischen Sinne nicht intelligent oder in der Lage zu reflektieren und intuitiv zu handeln. Daher ist es äußerst schwierig, Roboter herzustellen, weil sie sich wegen ebendieser Mängel nicht automatisch an unbekannte Variable anpassen können.

Regeln:

- Drei Lernende bilden zusammen einen Roboter (Mikrocontroller, Sensor und Aktor):
- Der Mikrocontroller (trägt eine Augenbinde):
    - Kann Entscheidungen treffen.

    - Kann den Sensor nach Informationen fragen und sich die Antwort anhören.
    - Kann dem Aktor Befehle erteilen.
- Der Sensor:
    - Kann die Welt wahrnehmen und dem Mikrocontroller antworten, wenn dieser ihn fragt.
- Der Aktor (trägt eine Augenbinde):
    - Kann durch Hören Befehle des Mikrocontrollers aufnehmen und sie ausführen.

Aufbau:

- Weisen Sie jeder Gruppe einen Tisch zu, vor dem sich die Gruppe hinstellt.
- Nachdem Mikrocontroller und Aktor die Augen verbunden worden sind, verteilen Sie drei Kunststoffbecher im nahen Umfeld des „Roboters" – gern in unterschiedlichen Höhen (unter einem Stuhl, auf dem Nachbartisch u. Ä.).

Herausforderungen und Ziele:

- Lokalisieren der drei Kunststoffbecher.
- Einsammeln der drei Kunststoffbecher.
- Stapeln der drei Kunststoffbecher auf dem zugewiesenen Tisch.

Phase 2 – Die praktische Umsetzung (18 Min.):

Die Lernenden sollen – ein*e jede*r in ihrer bzw. seiner Roboterfunktion – zusammenarbeiten, um die Aufgabe zu bewältigen und die Kunststoffbecher auf ihren jeweiligen Tischen zu stapeln. Wenn das Ziel erreicht ist, wird rotiert, d. h. die Lernenden schlüpfen in eine andere Roboterfunktion – bis sie alle drei Rollen (Mikrocontroller, Sensor, Aktor) ausprobiert haben.

Einführung in den Arduino – Kennenlernen des Arduino-Systems (6 Std. und 30 Min.)

Kurz zum Inhalt: Dieser Abschnitt mischt Theorie mit praktischen, konkreten Hands-on-Übungen zu Verständnis und Anwendung verschiedener ausgewählter Sensoren und Aktoren sowie zu grundlegenden Programmierprinzipien. Die hier besprochenen Programmierprinzipien, Sensoren und Aktoren sind Teil des abschließenden Projekts.

Lernergebnisse: Die Lernenden eignen sich ein grundlegendes Verständnis von Elektrizität, Mikrocontrollertechnologie (Arduino Uno), Programmierprinzipien sowie Sensoren und Aktoren an. Darüber hinaus können sie mit einfachen Input/Output-Systemen selbst arbeiten und diese erstellen.

Konkretes Material je Gruppe (2 Lernende je Gruppe): Dieser Abschnitt enthält eine größere Auswahl verschiedener Materialien und Komponenten. Die vollständige Liste finden Sie im Downloadbereich auf der Webseite des Projekts (http://panama-project.eu/). Dies gilt auch für die entwickelten Spickzettel (cheat sheets), von denen Teile in diesem Abschnitt enthalten sind, sowie für Programmbeispiele für die verschiedenen Sensoren und Aktoren; ebenso Lösungsvorschläge für die verschiedenen Aufgaben. Die entwickelten Spickzettel können in der ersten Phase dieses Abschnitts ausgedruckt und verteilt werden.

Hinweis: Nach jeder Lektion gibt es ein paar Aufgabenbeispiele für die Schülerinnen und Schüler, mit denen sie mit den Schaltkreisen und dem Code aus der Lektion experimentieren können. Es wird daher empfohlen, den Schülern zu Beginn jeder Lektion die darin verwendeten Materialien auszuhändigen, damit sie die Schaltung und den Code für sich selbst kopieren können.

Erläuterung zum weiteren Procedere (10 Min.):

Im weiteren Verlauf sollten die Schülerinnen und Schüler eine Einführung in das endgültige Themenprojekt und die folgenden Phasen erhalten, um die bevorstehenden Lektionen in einem konkreten Kontext sehen zu können. Sie lernen so beispielsweise den mit IR-Sensoren versehenen Transportroboter kennen sowie in diesem Zusammenhang die Verwendung von Mikrocontrollern.

Phase 1 – Einführung in die Themen Elektrizität und Aktoren (LED) (20 Min.):

Diskutieren Sie mit den Lernenden, wie Elektrizität grundlegend in elektrischen Schaltkreisen fließt, und erläutern Sie kurz, was Strom, Spannung und Widerstand sind. Auf detaillierte Berechnungen kann verzichtet werden.

Zeichnen Sie das Diagramm einer Stromquelle (5 V), die eine LED (rot – 5 mm) einschließlich Vorwiderstand von 150 Ω (für eine Wiederholung der Widerstandsgrößen und -farbmarkierungen vgl. Abb. 2) versorgt, und erläutern Sie dieses.

Erläutern Sie nun das Steckbrett (*breadboard*) und wie die Öffnungen untereinander leitend verbunden sind, gern mithilfe bildlicher Darstellungen (vgl. u. a. Abb. 1). Danach kann der Schaltkreis auf dem Steckbrett aufgebaut werden, während der Arduino mit seinen 5V- und GND (*Ground*, deutsch: Erdung)-Pins als Stromversorgung hierfür dienen kann. Lassen Sie die Lernenden die Schaltung auf ihrem eigenen Steckbrett nachbauen.

Erklären Sie, dass damit eine der drei Kernkomponenten an einen Roboter realisiert ist: der Einsatz einer LED als Aktor.

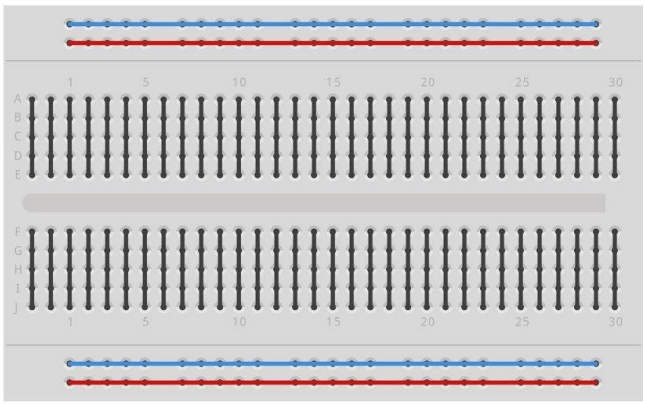


Abb. 1: Übersicht über die internen Verbindungen des Steckbretts

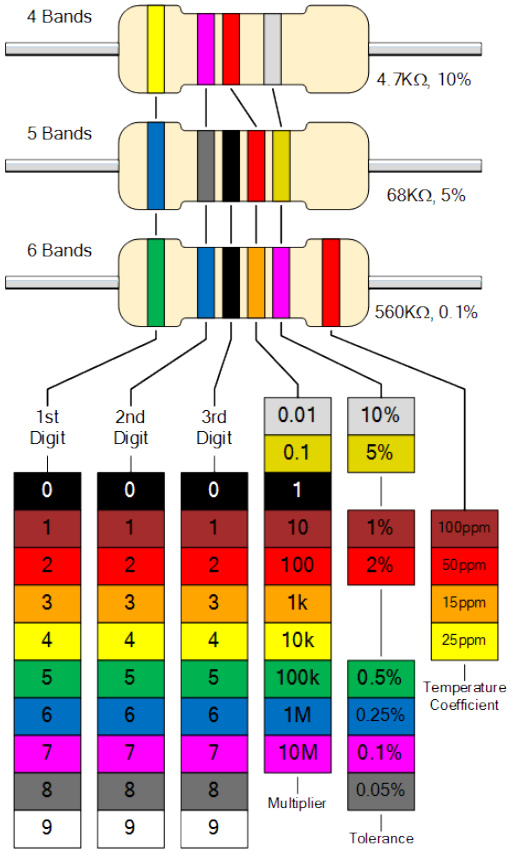


Abb. 2: Übersicht darüber, wie die Farbcodes der Widerstände abzulesen sind

Phase 2 – Einführung in den Arduino (20 Min.):

Stellen Sie den Arduino und seine allgemeine Funktionalität vor, insbesondere mit Fokus auf die Pins und deren Funktion. Führen Sie ebenfalls in die Programmierungsumgebung des Arduino ein.

Verwenden Sie jetzt den Arduino, um die LED von vorhin zu steuern anstatt ihn nur zur Stromversorgung zu verwenden. Die Anleitung sollte für alle gleichzeitig sichtbar sein, damit die Lernenden es an ihren eigenen Computern nachvollziehen und das Programm gleichzeitig mit der Lehrkraft programmieren können. Beispiele für die Programmierung wie auch für die nachfolgenden Aufgaben finden Sie in den Abbildungen 4 – 6.

Beginnen Sie mit der Einführung in die Variablen. Hier wäre es auch möglich, kurz auf die am häufigsten verwendeten Datentypen (*int, String, boolean*) einzugehen, da der angeschlossene Pin später als *int* (*integer*, deutsch: Zahl) gespeichert wird. Erklären Sie anschließend, wie die Programmstruktur funktioniert, und stellen Sie den angeschlossenen Pin als OUTPUT ein. Dann wird im Loop-Teil der angeschlossene Pin auf HIGH eingestellt. Besprechen Sie gern auch noch einmal den Unterschied

zwischen INPUT und OUTPUT, und warum für die LED stets OUTPUT verwendet wird sowie den Unterschied zwischen LOW (0 V) und HIGH (5 V).

Mögliche Analogie: An dieser Stelle ist ein Vergleich zur Verdeutlichung hilfreich, beispielsweise mit befüllten Umzugskartons, die in der Garage stehen und auf mit einem Aufkleber versehen sind. So ist es einfach, stets den gewünschten Karton zu finden und das zu finden, was enthalten ist. Ebenfalls kann ein Gegenstand herausgenommen und durch einen neuen ersetzt werden. Variablen sind somit eine Methode zum Ablegen von Informationen im Speicher des Mikrocontrollers, damit sie später abgerufen werden können.

Nachdem die Lernenden die LED zum Leuchten gebracht haben, können sie sie blinken lassen, indem die Einstellung für den angeschlossenen Pin zwischen HIGH und LOW alterniert, ohne dass jedoch zunächst eine Verzögerung eingebaut ist. Der Unterschied zwischen dem einfachen Leuchten und dem Blinken wird für die Lernenden kaum erkennbar sein. Sie können hier aber sehen wie schnell ein Prozessor Berechnungen anstellen und darauf basierende Aktionen ausführen kann. Nun kann die Verzögerungsfunktion (*delay*) eingeführt werden, wobei die Schülerinnen und Schüler zunächst selbst einschätzen sollen, an welcher Stelle des Programmcodes diese eingefügt werden soll. In vielen Fällen lautet die Antwort, dass es zwischen dem HIGH- und LOW-Setzen des Pins sein sollte. Infolge dieser Antwort bietet sich eine Übung an, da nach wie vor wie o.g. kein signifikanter Unterschied erkannt wird. Ein*e Lernende*r kann gebeten werden, in Abständen von ca. 2 Sekunden die Beleuchtung ein- und auszuschalten. Lassen Sie die anderen Lernenden genau verbalisieren, was sie sehen: Licht einschalten, warten, Licht ausschalten, warten, Licht einschalten usw. Bitten Sie die Lernenden nun, den Code des Programms nachzuvollziehen: Einschalten, Warten, Ausschalten, Warten, Einschalten usw. Hier werden sie höchstwahrscheinlich schnell feststellen, dass bei Einsatz einer Schleife (l*oop*) keine Wartezeit zwischen dem Übergang von der letzten zur ersten Programmzeile der Schleife auftritt, weshalb auch hier eine Verzögerung explizit eingebaut werden muss, nachdem der Pin auf LOW gesetzt wurde.

Aufgabe für die Lernenden:

- Eksperimente mit der Blinkfrequenz (also den Zeiteinstellungen) - interessant ist es hier insbesondere den Zeitintervall in Millisekunden zu finden, bei dem das Auge den Wechsel vom konstanten Leuchten zum Blinken der LED wahrnehmen kann.
- Experiment mit zwei LEDs, die erst gleichzeitig und dann abwechselnd blinken.

Durch Anpassen der Größenverhältnisse der eingebauten Delays kann der Eindruck entstehen, dass die LED nur beispielsweise halb so stark leuchtet. Durch das Experimentieren kann auch verdeutlicht werden, welche Unterschiede es zwischen digitalen, analogen und PWM-Signalen (*Pulse Width Modulation*, deutsch: Pulsweitenmodulation) gibt.

Erläutern und diskutieren Sie an dieser Stelle die Unterschiede zwischen digitalen und analogen Signalen und erklären Sie gleichzeitig, dass der Arduino analoge Signale nur lesen, aber nicht erzeugen kann. Stattdessen verwendet der Arduino ein simuliertes analoges Signal namens PWM. Das PWM-Signal ist im Grunde das gleiche wie das, was die Lernenden kurz zuvor durch das Manipulieren des Größenverhältnisses der verwendeten Delays (vgl. Abb. 3) selbst konstruiert haben.

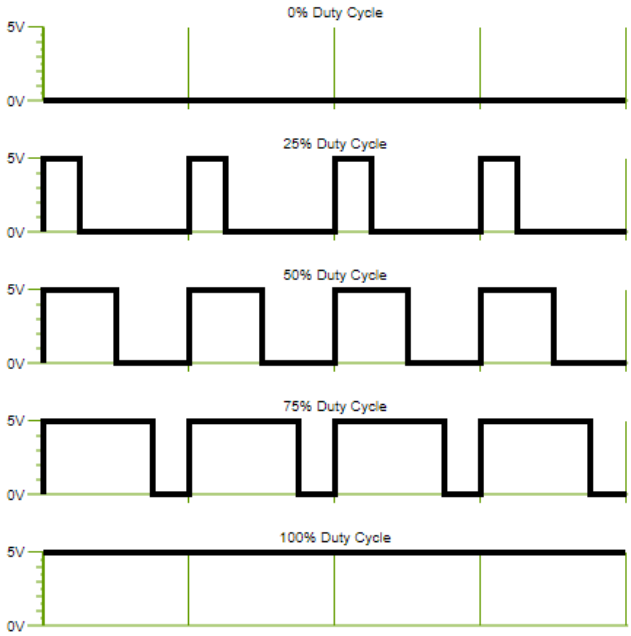


Abb. 3: Darstellung, wie ein PWM-Signal ein Analogsignal simuliert

Beispiele für die Arduino-Programmierung:

Arduino-Pins:

```
analogRead:    A0-A5
digitalRead:   A0-A5, 2-13

analogWrite:   3, 5-6, 9-11
digitalWrite:  A0-A5, 2-13

Verwendet vom R3 Motorshield:
A0-A1, 3, 8-9, 11-13
```

Datentypen:

```
int:      -32768 to 32767
          LOW oder HIGH
          A0-A5 (analoge Pins)
          2-13 (digitale Pins)

bool:     wahr oder falsch
          0 oder 1
          LOW oder HIGH

String: "text text"

void:      keine Daten
```

Programmstruktur:

```
// hier können Variablen
initialisiert werden

//läuft nur einmal durch
void setup(){
  // hier werden Pins
konfiguriert und Serien
aktiviert
}

//läuft wiederholt durch
```

```
void loop() {
  //hier den Code für das
Programm schreiben
}
```

Abb. 4: Übersicht über die Arduino-Pins, die meistgenutzten Datentypen und die Programmstruktur

Variable (mit einem Wert):

```
//Datentyp Name = Wert
int varName = 13;

//Datentyp Name = Wert
bool varName = true;

// Datentyp Name = Wert
String varName = "Hello!";
```

Delay:

```
void loop() {
  //pausiert für 1000
Millisekunden
  delay(1000);
}
```

Abb. 5: Übersicht über die Art der Nutzung von Variables und Delays

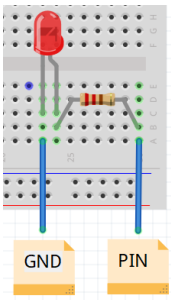
LED: 5 mm rote LED (Größe Widerstand: 150Ω)

```
int LEDPin = 2;                //der verwendete Pin

void setup() {
  pinMode(LEDPin, OUTPUT);     //Pin als Output
einstellen
}

void loop() {
  //LED leuchtet nicht (LOW) oder leuchtet (HIGH)
*
  digitalWrite(LEDPin, HIGH);

  //kontinuierlich von aus (0) zu ein (255)
wechseln **
  analogWrite(LEDPin, 180);
}
```



GND: kürzeres Metall
PIN: längeres Metall

*PIN: 2, 4-7, 10, A2-A5
**PIN: 5, 6, 10

Abb. 6: Beispiel für den Einsatz einer LED

Phase 3 – Einführung in das Thema Sensoren (Schalter) (20 Min.):

Stellen Sie den Schalter als Bauteil vor und erläutern Sie seine Funktion: im geöffneten Zustand ist der Stromkreis geöffnet und durch Betätigen des Schalters lässt sich der Kreis schließen, so dass Strom fließen kann.

Stellen Sie damit einen Schaltkreis auf dem Steckbrett her und schließen Sie ihn an den Arduino an. Lassen Sie dies parallel von den Lernenden bauen. Erläutern Sie kurz, dass in diesem Zusammenhang einen Widerstand verwenden werden muss, um den Schaltkreis nicht zu „verschmoren“. Erklären Sie, dass Schalter auch Sensoren sind und somit im Setup-Teil des Arduino-Codes als INPUT angegeben werden müssen. Für später ist anzumerken, dass der verwendete Widerstand nicht unbedingt erforderlich ist - es können stattdessen die eingebauten Pull-up-Widerstände des Arduino verwendet werden, die durch Ersetzen von "INPUT" durch "INPUT_PULLUP" aktiviert werden. Für den Moment macht eine Verwendung eines Widerstands jedoch Sinn, da dies eine Einführung in die Verwendung von Spannungsteilern beim Aufbau diverser Sensoren ermöglicht.

Erstellen Sie ein neues Programm, in dem der Wert des Schalters gelesen wird. Verzichten Sie zunächst auf die gesamte serielle Kommunikation (Serial.begin (9600) und Serial.println (switchValue)), so dass nur analogRead (switchPin) vorhanden ist. Sprechen Sie dann mit den Lernenden darüber, wie ein Einblick in das „Denken” des Mikrocontrollers gewonnen werden kann. Es ist hier durchaus möglich, dass der Eine oder die Andere möglicherweise aus früherem Unterricht oder eigenständigen Projekten in der Freizeit mit Printstatements vertraut ist.

Stellen Sie dann den seriellen Monitor sowie die Art und Weise vor, wie die serielle Kommunikation aktiviert und sog. Print-Statements eingesetzt werden, um Werte von Variablen und eigene Textmitteilungen an den Monitor auszugeben (*print*, deutsch zu drucken). Erklären Sie gleichzeitig, wie Print-Statements zur Fehlerbehebung (d*ebugging*) genutzt werden können - sie stellen eine unmittelbare Möglichkeit dar, nachzuvollziehen, welche Teile des Programms ausgeführt werden und mit welchen Werten operiert wird.

Erklären Sie den Lernenden, dass mit dem Einsatz des Schalters als Sensor eine weitere Anforderung für die Definition eines Roboters erfüllt ist, nämlich die Wahrnehmung der Welt. Um die dritte Anforderung zu erfüllen, muss der Arduino 'lediglich' noch dazu gebracht werden, daraus abgeleitet Entscheidungen zu treffen. Stellen Sie somit den Aufbau eines Programmes vor und legen Sie dar, wie ein Ein- und Ausschalten der LED durch den Mikrocontroller herbeigeführt werden kann, je nachdem, ob der Schalter gedrückt ist oder nicht.

Aufgaben:

- Experimentieren Sie mit den Lernenden, welche Größe eine Verzögerung möglicherweise haben sollte, bevor die Schwingungen herausgefiltert werden, damit der Schalter nicht öfter als tatsächlich gedrückt gedrückt wird.

Beispiele für die Programmierung:

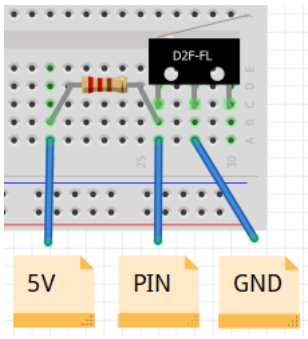
Schalter: Größe Widerstand: 1KΩ

```
int switchPin = A5;             //verwendeter Pin
int switchValue;                //Speichern des
Wertes

void setup() {
```

```
  pinMode(switchPin, INPUT);   //Pin als INPUT
einstellen
  Serial.begin(9600);          //serielle Ausgabe
aktivieren
}

void loop() {
  //den Wert lesen und speichern, dann auch
ausgeben
  switchValue = digitalRead(switchPin);
  Serial.println(switchValue);
}
```



*PIN: 2, 4-7, 10, A2-A5

Abb. 7: Beispiel für den Einsatz eines Schalters

Bedingung (Struktur):

```
void loop() {
  if (x == y) {
    //falls Statement wahr, Code
ausführen
  }
  else if (x < 4 && y < 4) {
    //nur falls Statement wahr,
diesen Code ausführen
  }
  else {
    //sonst diesen Code
ausführen
  }
}
```

*es kann nur einen “falls”-Befehl geben
*es kann mehrere “nur falls”-Befehle geben
*es kann nur einen “sonst”-Befehl geben

Bedingung (Operatoren):

```
==   gleich
!=   nicht gleich
>    größer
<    kleiner
>=   größer gleich
<=   kleiner gleich

&&   und
||   oder
```

Abb. 8: Übersicht über die Nutzung von Bedingungen

Phase 4 – Erläutern von Spannungsteilern und Sensoren (LDR) (45 Min.):

Spannungsteiler bilden die Basis vieler Sensoren. Führen Sie daher in das Thema Spannungsteiler ein und erläutern Sie, wie die Widerstände interagieren.

Mögliche Analogie: Die Spannung kann als eine Schale mit fünf Stück Kuchen gesehen werden (ein Stück Kuchen für jedes der 5 Volt). Die beiden Widerstände werden nun zwei Personen gleichgesetzt. Hinzu kommt die Anforderung, dass alle Kuchenstücke gegessen sein müssen, nachdem die Schale an der letzten Person vorbeigekommen ist. Es ist wichtig zu betonen, dass es keine Rolle spielt, wie hungrig die Personen sind, sondern dass lediglich alle fünf Kuchenstücke gegessen werden müssen.

Entscheidend ist vielmehr, wie hungrig die Personen im Verhältnis zueinander sind. Wenn sie gleich hungrig sind, teilen sie sich die Kuchenstücke gleichmäßig auf. Aber wenn die erste Person im Vergleich zur zweiten sehr großen Hunger hat, isst sie die meisten Kuchenstücke, während die zweite weniger Kuchenstücke isst.

Nachdem die Kuchenschale an der ersten Person vorbeigekommen ist, lässt sich feststellen, wie viele Kuchenstücke noch übrig sind. Dies ist der Indikator für das Verhältnis zwischen dem Hunger der beiden Personen, da bekannt ist, dass die andere Person auf jeden Fall den Rest essen muss.

Nach der Einführung und Erläuterung der Funktionsweise von Spannungsteilern können die folgenden Übungen durchgenommen und ausgeführt werden (der Schalter wird im Stromkreis durch einen Widerstand ersetzt – danach ist der Aufbau der gleiche wie zuvor). Denken Sie daran, zu wiederholen, dass analoge Werte des Arduino als Zahl zwischen 0 (0 V) und 1023 (5 V) gelesen werden.

Widerstand 1:	Widerstand 2:	Abgelesener Wert:	Wert in Volt:
1KΩ	1KΩ		
560Ω	560Ω		
820Ω	180Ω		
180Ω	820Ω		
1KΩ	Unendlich (Schalter oben)		
1KΩ	Keiner (Schalter unten)		

Wenn die Übungen absolviert sind, kann man sich der Formel für die Spannungsteiler widmen, wonach weitere Aufbauten zunächst berechnet und dann in der Praxis erprobt werden können.

$$U1 = U * \frac{R1}{R1 + R2}$$

Erklären Sie den Lernenden, dass die soeben konstruierten Spannungsteiler keine Sensoren sind, auch wenn viele Sensoren auf Spannungsteilern basieren. Das liegt daran, dass es sich hier um statische Spannungsteiler handelt und daher keine Änderungen in der Umgebung erkannt werden soll bzw. kann.

Erläutern Sie den Lernenden nun, dass zur Konstruktion ihres eigenen Sensors anstelle von statischen Widerständen ein dynamischer Widerstand verwendet werden kann, der seinen Innenwiderstand im Takt der Änderungen seiner Umgebung ändert. Stellen Sie daher den lichtabhängigen Widerstand (LDR, *Light Dependent Resistor*, deutsch: Fotowiderstand) vor und erläutern Sie, dass sein Innenwiderstand dynamisch variiert, je nachdem, wie viel Licht auf ihn fällt. Daher soll der LDR verwendet werden, um einen ersten Sensor zu konstruieren.

Setzen Sie den LDR in den vorgesehenen Spannungsteiler ein, lesen Sie den Wert des Spannungsteilers ab und geben Sie ihn an den Monitor aus. Da der Spannungsabfall über dem einzelnen Widerstand im Spannungsteiler jetzt nicht mehr konstant ist, repräsentiert der abgelesene Wert jetzt die Änderungen in der Umgebung, in der sich der Sensor befindet (Lichtstärke).

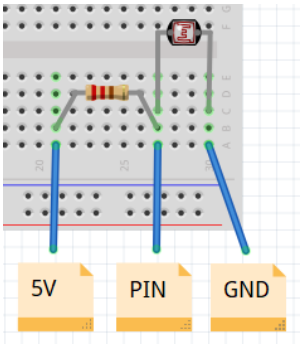
Diese Aufgaben können Sie als Impulse an die Lernenden geben:

- Erstellen Sie ein System, das eine LED bei Dunkelheit ein-, bei Helligkeit jedoch wieder ausschaltet. Dieses System wird z. B. bei Straßenlaternen genutzt.
- Erstellen Sie ein System, das die Helligkeit der LED schrittweise erhöht bzw. verringert, wenn sich die Lichtverhältnisse ändern. Dies ist eine gute Übung zum Einüben des Verhältnisses von abgelesenen analogen Signalen (0-1023) zu PWM-Signalen (0-255). Dieses System lässt sich auch auf Smart Homes beziehen, bei denen beispielsweise ein konstantes Lichtniveau im Wohnzimmer angestrebt wird.

LDR (Light Dependent Resistor/Fotowiderstand): Größe Widerstand: 1KΩ

```
int LDRPin = A5;                //verwendeter Pin
int LDRValue;                   //Speichern des
Wertes
void setup() {
  pinMode(LDRPin, INPUT);       //Pin als INPUT
einstellen
  Serial.begin(9600);           //serielle Ausgabe
aktivieren
}

void loop() {
  // den Wert lesen und speichern, dann auch
ausgeben
  LDRValue = analogRead(LDRPin);
  Serial.println(LDRValue);
}
```



*PIN: A2-A5

Abb. 9: Beispiel für den Einsatz eines LDRs

Phase 5 – Erweiterung um zusätzliche Sensoren (Potentiometer, IR-Sensor) (45 Min.):

Führen Sie die Lernenden in das Potentiometer bzw. den IR-Sensor ein und erläutern Sie, wie diese aufgebaut sind und wie sie funktionieren. Gleichzeitig kann erklärt werden, dass auch diese wie der bereits konstruierte Lichtsensor (LDR) um einen Spannungsteiler herum aufgebaut sind. Deshalb können die Bauteile untereinander ausgetauscht werden, während der Code und der Rest des Setups unverändert bleiben. Erklären Sie, dass das Potentiometer einen stufenlosen Übergang im Verhältnis der Werte der beiden Widerstände aufweist, je nachdem, wo der Regler mit dem kreisförmigen Widerstand interagiert.

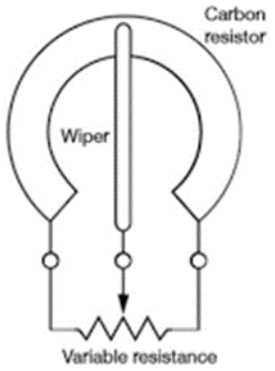


Abb. 10: Aufbau und Funktionsweise eines Potentiometers

Der IR-Sensor hingegen ist mit einer IR-LED sowie einem IR-empfindlichen Fotowiderstand versehen – das ist der IR-empfindliche Fotowiderstand, der zusammen mit einem statischen Widerstand den Spannungsteiler bildet – dessen Widerstand variiert, je nachdem, von wieviel Infrarotlicht er getroffen wird. Die Schüler werden daher auch feststellen, dass er, wenn sie ihn von der Oberfläche weg auf Abstand halten, "schwarz" liest (oder vielmehr einen Mangel an Reflexion feststellt).

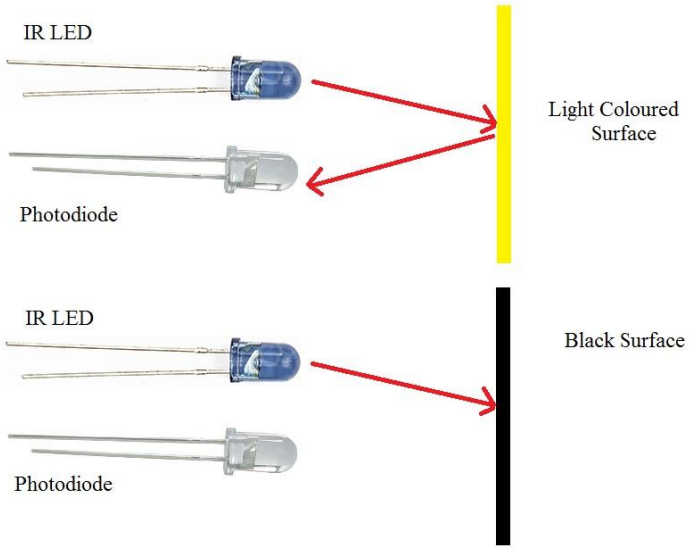


Abb. 11: Abbildung der Funktionsweise eines IR-Sensors

Aufgaben, Potentiometer:

- Erstellen Sie ein System, das die LED nach und nach in fünf Schritten einschaltet, wenn das Potentiometer von links nach rechts gedreht wird.
- Erstellen Sie ein System, das die LED nach und nach stufenlos einschaltet, wenn das Potentiometer von links nach rechts gedreht wird. Den Bezug haben wir hier z. B. zu einer Stereoanlage, in der häufig Potentiometer verbaut sind – oder auch zum heimischen Beleuchtungsdimmer.

Aufgaben, IR-Sensor:

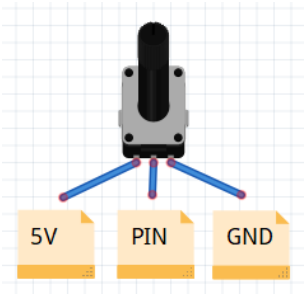
- Lassen Sie die Schüler experimentieren, in welcher Entfernung der IR-Sensor den Unterschied zwischen weißem und schwarzem Papier am besten erfasst (ca. 0,5 bis 0,7 cm über der Oberfläche).
- Erstellen Sie ein System, in dem eine LED aufleuchtet, wenn der IR-Sensor einen niedrigen Wert liest (massive Blendung durch weißes Papier), jedoch erlischt, wenn ein hoher Wert gelesen wird (minimale Blendung durch schwarzes Papier).
- Erstellen Sie ein System, bei dem eine LED in fünf Intervallen mehr und mehr aufleuchtet, basierend auf dem Widerschein (der Blendung) auf den IR-Sensor.

Potentiometer:

```
int PotPin = A5;                    //the used pin
int PotValue;                       //to store the
value

void setup() {
  pinMode(PotPin, INPUT);           //set pin as INPUT
  Serial.begin(9600);               //enable serial
}

void loop() {
  //read and store the value, then prints it
  PotValue = analogRead(PotPin);
  Serial.println(PotValue);
}
```



*PIN: A2-A5

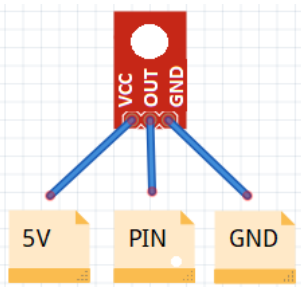
Abb. 12: Beispiel für das Ablesen des Potentiometers.

IR-Sensor: QRE1113 (Analog)

```
int IRPin = A5;                     //the used pin
int IRValue;                        //to store the
value

void setup() {
  pinMode(IRPin, INPUT);            //set pin as INPUT
  Serial.begin(9600);               //enable serial
}

void loop() {
  //read and store the value, then prints it
  IRValue = analogRead(IRPin);
  Serial.println(IRValue);
}
```



*PIN: A2-A5

Abb. 13: Beispiel für das Ablesen des IR-Sensors

Phase 6 – Aktoren (Servo- und DC-Motor) (45 Min.):

Führen Sie die Schüler in den Servomotor und damit auch in die Bibliotheken ein und erklären Sie ihnen, dass dies eine Sammlung vorprogrammierter Codes ist, was wir in diesem Fall zur Steuerung des Servomotors anwenden wollen. Lassen Sie die Lernenden den Servomotor auf eine bestimmte Position einstellen und dann versuchen, ihn so einzustellen, dass er sich von einer Seite zur anderen bewegt. Hier ist es wichtig, auf den Einsatz von Delaylicht zu sprechen zu kommen, weil der Servomotor ohne diese Verzögerungen keine Zeit hat, in die angegebenen Position zu gelangen, bevor er das Signal für die nächste Position erhält und umgekehrt.

Aufgaben, Servomotor:

- Erstellen Sie ein System, bei dem die Position des Servomotors der des Potentiometers folgt (diesmal ist der Bezug z. B. das Ellbogengelenk eines Exoskeletts, damit – weiter gedacht – ältere Menschen Dinge heben können, die sie sonst nicht mehr heben könnten).
- Entwerfen Sie ein System zum automatischen Öffnen des Fensters, bei dem der Servomotor beim Betätigen eines Schalters eine Position zwischen 0 und 90 ° ermöglicht.

Führen Sie sie sodann in die Motorabschirmung und in den Gleichstrommotor ein. Im gleichen Zug können die eingebauten H-Brücken der Motorabschirmung (sie hat zwei, für jeden Motor eine) ganz kurz erklärt werden. Abbildung 14 zeigt eine vereinfachte Version des Schaltkreises einer H-Brücke. Wichtig sind die vier Transistoren P1, P2 und N1, N2 sowie die beiden Verbindungen dorthin, F1 und F2. Sie sind so eingestellt, dass beim Anlegen einer Spannung an F1 P1 geöffnet und N1 geschlossen wird und umgekehrt, wenn an F1 keine Spannung anliegt. Entsprechend hängen P2 und N2 von der Spannung an F2 ab. Wenn man also 5 V an F1 und 0 V an F2 anlegt, fließt der Strom in eine Richtung durch den Motor – und in die andere Richtung, wenn wir 0 V an F1 und 5 V an F2 legen. Legt man sowohl an F1 als auch F2 5 V oder 0 V an, so kann der Strom weder in die eine noch die andere Richtung fließen. Der Grund, warum wir die Motorabschirmung und ihre H-Brücken zur Steuerung der Gleichstrommotoren verwenden, ist, dass wir dadurch einen stärkeren Strom durch die Motoren leiten können, als ihn der Arduino selbst liefern kann. Der Strom, den wir hindurchleiten, kommt direkt vom angeschlossenen Akkusatz (9 V).

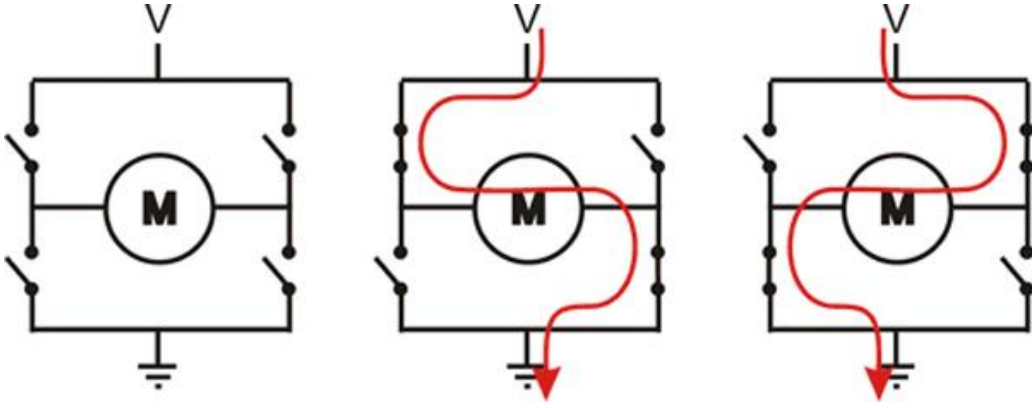


Abb. 14: Vereinfachte Darstellung des Schaltkreises einer H-Brücke

Aufgaben, DC-motor:

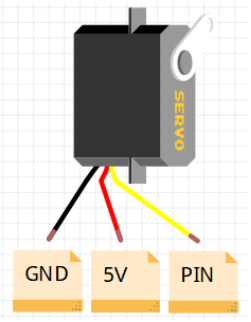
- Bringen Sie den Gleichstrommotor dazu, 5 Sekunden lang abwechselnd im und gegen den Uhrzeigersinn zu laufen.
- Bringen Sie den Gleichstrommotor dazu, seine Drehzahl allmählich zu ändern, wenn das Potentiometer von links nach rechts und umgekehrt gedreht wird.

Servomotor:

```
#include <Servo.h>          //include library
Servo servoMotor;           //create new servo
object
int servoPin = 2;           //the used pin

void setup() {
  //attach the used pin to the servo object
  servoMotor.attach(servoPin);
}

void loop() {
  //turn the servo to a position (value: 0-179)
  servoMotor.write(90);
}
```



*PIN: 2, 4-7, 10, A2-A5

Abb. 15: Beispiel, wie der Servomotor in eine bestimmte Position gebracht werden kann

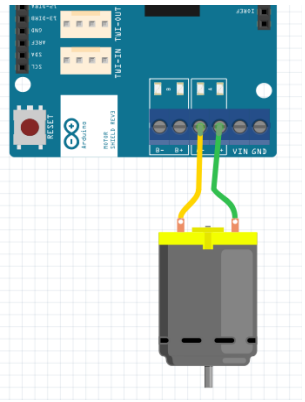
Gleichstrommotor (DC):

```
int motorASpeed = 3;                //PWM A
int motorADirection = 12;           //DIR A

void setup() {
  //set all motor pins to PUTPUT
  pinMode(motorASpeed, OUTPUT);
  pinMode(motorADirection, OUTPUT);
}

void loop() {
  //DIRECTION: Turn clockwise = HIGH,
  //turn counter-clockwise = LOW
  digitalWrite(motorADirection, HIGH);

  //SPEED: no speed = 0, full speed = 255
  analogWrite(motorASpeed, 150);
}
```



*In Motor B genutzte Pins, siehe Motorab–schirmung

Abb. 16: Beispiel einer Art der Steuerung des DC-Motors

Phase 7 – Andere Sensorarten (Abstandssensor, Farbsensor) (30 Min.):

Stellen Sie den Schülern den Abstandssensor vor und erklären Sie, wie wir ihn zum Messen der Entfernung einsetzen können. Es ist so aufgebaut, dass er einen Lautsprecher und ein Mikrofon hat. Der Lautsprecher sendet einen Ping (den wir nicht hören können, weil die Frequenz zu hoch ist), der auf eine Oberfläche trifft und dann zurück zum Mikrofon reflektiert wird. Die Zeit zwischen dem Senden und Zurückprallen des Pings kann verwendet werden, um den Abstand zwischen dem Sensor und dem getroffenen Objekt zu berechnen. Zu diesem Zweck wird die Methode pulsIn() genutzt, die die Zeit aufzeichnet, die ein Ping benötigt, um entweder von HIGH nach LOW oder von LOW nach HIGH zu gelangen. Die Zeit wird in Mikrosekunden (0,000001 s) gemessen. Wir können daher folgende Formel anwenden, um den Abstand in cm zu berechnen:

$$Pulse\ In\ (in\ Mikrosekunden\ gemessen, 0.000001s)$$

$$Schallgeschwindigkeit\ = \frac{343\ m}{s} = \frac{34300\ cm}{s} = \frac{0.034\ cm}{ms}$$

$$Entfernung\ zum\ Objekt\ in\ cm: Pulse\ In\ (ms) * \frac{0.034\ cm}{1\ ms} / 2\ (vor\ und\ zurück)$$

Aufgaben für die Lernenden:

- Lassen Sie die Schüler verschiedene Entfernungen im Klassenzimmer messen und diese auf dem Bildschirm ausgeben. Lassen Sie sie ermitteln, in welcher Entfernung es funktioniert und wann es nicht funktioniert (ca. 2 cm – 400 cm).
- Lassen Sie die Schüler ein Alarmsystem bauen, das eine LED aufleuchten lässt, wenn ein Objekt zu nahe kommt.

Stellen Sie nun den Farbsensor vor und erklären Sie kurz, wie er funktioniert. Er ist so konzipiert, dass er über einen Filter den reflektierten Wert von rotem, grünem und blauem (RGB-)Licht misst – eine Farbe nach der anderen. Aus den RGB-Werten kann man alle Farben erstellen, wobei es wichtig ist zu wissen, dass alle drei zusammen Weiß ergeben, während das Fehlen aller drei zu Schwarz führt. Im Idealfall verhalten sich die Werte wie in Abbildung 17, in der Realität ist dies jedoch nicht der Fall. Daher erkennen wir bei Einsatz des Sensors zuerst eine Farbe und notieren, welche RGB-Werte der Sensor zurückgibt. Anschließend können wir diese Werte nutzen, um genau unsere Farbe zu definieren (in der *library* können nur die Farben Rot, Grün, Blau und Gelb definiert werden - siehe Abbildung 19). Weil die RGB-Werte die Menge des reflektierten Lichts darstellen, sind außerdem wechselnde Lichtverhältnisse zu beachten. Hat man beispielsweise in gleißendem Sonnenlicht die Farbe Gelb aufgrund mehrerer gelesener RGB-Werte definiert, so sind diese nicht unbedingt gleich, wenn es Abend ist. Um dies zu vermeiden, empfiehlt es sich daher, das Gelesene abzuschirmen, um die Lichtverhältnisse statischer zu halten.

Aufgaben zum Farbsensor:

- Lassen Sie die Schüler mit dem Messen und Definieren der Farben Rot, Grün, Blau, Gelb, Weiß und Schwarz experimentieren. Welche farblichen RGB-Werte unterscheiden sich am meisten voneinander, und liegen einige näher beieinander als andere? (Mischfarben wie Gelb sind möglicherweise schwieriger zu definieren, da sie nicht nur aus einer einzigen Farbe bestehen.)

- Experimentieren Sie mit den Differenzen der Farbwerte in Abhängigkeit von den Lichtverhältnissen.
- Lassen Sie die Schüler eine Konstruktion erstellen, die Bedingungen nutzt, um die registrierten Farben auf dem Bildschirm auszugeben.

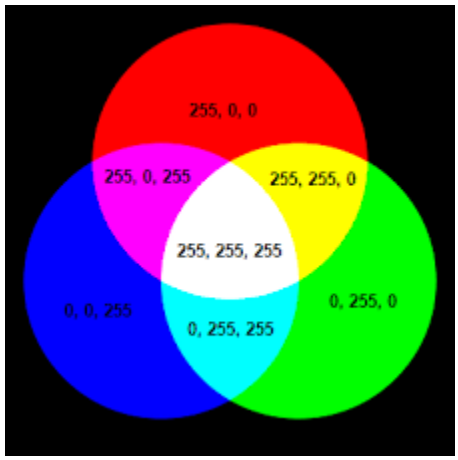


Abb. 17: Übersicht über die idealen RGB-Werte

Ultraschallsensor: HC-RS04

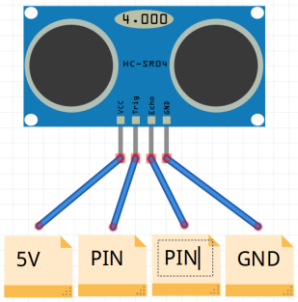
```
int trigPin = 2;            //the used trig pin
int echoPin = 4;            //the used echo pin
int distance;               //to store the value

void setup() {
  pinMode(trigPin, OUTPUT); //sets pin as OUTPUT
  pinMode(echoPin, INPUT);  //sets pin as INPUT
  Serial.begin(9600);       //enables serial
}

void loop() {
  //store the returned value from the function
  distance = getDistance();

  //prints the stored value
  Serial.println(distance);
}

//function - returns the distance
int getDistance() {
  //sends out a trigger sound
  digitalWrite(trigPin, LOW);
  delayMicroseconds(10);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
```

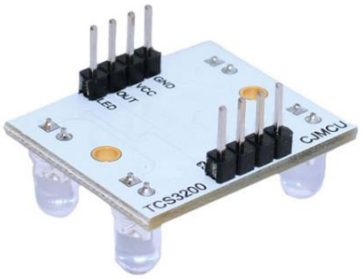


*PIN: 2, 4-7, 10, A2-A5

```
  //returns the received echo in centimeter
  return pulseIn(echoPin, HIGH)* 0.034/2;
}
```

Abb. 18: Beispiel für einen Einsatz des Abstandssensors

Farbsensor: TCS3200




```
//includes the library
#include <Color.h>

//the used pins
int S0 = 2;
int S1 = 4;
int S2 = 5;
int S3 = 6;
int OUT = 7;

//creates a new color-sensor object
Color color(S0, S1, S2, S3, OUT);

//used to store the color-value
String colorValue = "";

void setup() {
  setColors();
  Serial.begin(9600);  //enables serial
}

void loop() {
  //prints the R(ed), G(reen), B(lue) and W(hite) values from the
  //sensor.
  //write down these values for each color you want to identify
  //(red, green, blue, yellow) and insert them into the
  //setColors function.
  Serial.println(color.getRGBValues());

  //gets the color and prints it
  colorValue = color.getColor();
  Serial.println(colorValue);
}

//Defines the Colors
void setColors() {
  //color.setRed(R, G, B, W);
  color.setRed(40, 104, 22, 81);
  color.setBlue(46, 36, 14, 56);
  color.setGreen(89, 56, 16, 33);
  color.setYellow(29, 36, 12, 55);
}
```

```
READ ME NOTE: In order to enable the inclusion of the Color library, copy
paste the entire "Color Library" folder, into the "Documents -> Arduino ->
Libraries" folder.
```

Abb. 19: Beispiel für einen Einsatz des Farbsensors

Phase 8 – Einführung in weitere Programmierungsgrundsätze (Funktionen und While-Schleifen) (20 Min.):

Für den Einsatz des Abstandssensors und des Farbsensors wurde für beide eine Funktion angewandt. Führen Sie die Schüler daher in das Thema Funktionen ein und erklären Sie deren Arbeitsweise, indem Sie einen Codeabschnitt definieren, der dann immer wieder aufgerufen werden kann. Dies erleichtert sowohl die Anwendung (wir müssen den Code nur an einer Stelle korrigieren und nicht an vielen Stellen) als auch das Lesen (die Funktion kann viele Liniencodes unter einem einfachen und aussagekräftigen Namen abgrenzen).

Aufgaben zum Thema Funktionen:

- Erstellen Sie eine Funktion namens"printHello", die bei Aufruf "Hello" ausgibt.
- Erstellen Sie eine Funktion namens "sayHelloTo", die als Parameter eine Zeichenkette (String) verwendet, die bei Aufruf "Hello" und den Parameter ausgibt.
- Erstellen Sie eine Funktion namens "doubleThisNumber", die als Parameter einen int verwendet, der bei Aufruf den Wert des Parameters mit zwei multipliziert, woraufhin der neue Wert zurückgespielt wird. Der zurückgespielte Wert ist in einer Variablen zu speichern und auszugeben.

Führen Sie in das Thema while-Schleifen ein und erläutern Sie, wie diese eine Codeabschnitt ständig und immer weiter wiederholen, solange die daran geknüpfte Bedingung wahr ist.

Aufgaben zu while-Schleifen

- Lassen Sie die Schüler eine Funktion erstellen, die beim Drücken der Taste aufgerufen wird und eine while-Schleife enthält, die zehnmal ausgeführt wird, wenn nichts weiter unternommen wird, die jedoch vorzeitig unterbrochen wird, wenn die Taste zuvor wieder losgelassen wird. In der while-Schleife kann sinnvollerweise "while" ausgegeben werden, damit deutlich ist, dass sie jetzt aktiv ist.
- Erstellen Sie eine Funktion mit einer while-Schleife, die in 1-Sek.-Abständen "Happy Birthday to You" ausgibt; bei Drücken eines Schalters soll die Schleife unterbrochen werden.

Funktion (ohne Wertrückgabe):

```
void loop() {
  //runs the code inside-
  //myFunction
  myFunction();
}
```

Funktion (mit Wertrückgabe):

```
void loop() {
  //stores the returned value-
  //from myFunction
  int returnedValue;
  returnedValue = myFunction();
```

```
//datatype name() {}
void myFunction() {
  //some code to run
}
```

```
}

//datatype name() {}
int myFunction() {
  //some code to run

  //return a value
  int value = 100;
  return value;
}
```

Abb. 20: Übersicht über die Anwendung von Funktionen

While-Schleife:

```
void loop() {
  int x = 0;

  //while (argument is correct)
  while(x < 10) {
    x = x + 1;
  }
}
```

Unterbrechung:

```
void loop() {
  //while (argument - is correct)
  while(true) {

    if(something) {
      //break out of the loop
      break;
    }

  }
}
```

Abb. 21: Übersicht über die Anwendungsmöglichkeiten von while-Schleifen

Themenprojekt – Automatisierung (12 x 45 min.)

Kurz zum Inhalt: Das Projekt befasst sich mit der Automatisierung einer Kaffeebohnenzentrale, für die zwei Robotertypen entwickelt werden. Ein Roboter, der die Bohnen nach Qualität sortiert (Farbsortierung), und ein Roboter, der sie durch das Lager transportiert (Linienverfolgung). Die Schüler wählen aus, mit bzw. an welchem der beiden Roboter sie arbeiten möchten. Danach sollen sie sich in kleineren Gruppen relativ eigenständig mit dem Projekt beschäftigen (empfohlen werden hier Zweiergruppen).

Lernergebnisse: Die Schüler lernen die Anwendung von Inhalten aus früheren Lektionen in einem praktischen Kontext, um konkrete Probleme zu lösen – mit Bezug zu industriellen Arbeitsplätzen im 'wirklichen Leben'. Gleichzeitig erhalten sie einen tieferen Einblick in und Verständnis für die Entwicklung komplexer automatisierter Systeme sowie deren Grenzen.

Material: Im erarbeiteten Material wurde LEGO® als Plattform für die zu entwickelnden Roboter verwendet. Um die Integration der Elektronik in die LEGO-Plattform zu erleichtern, wurden gleichzeitig mehrere hiermit kompatible 3D-Modelle entwickelt. Anleitungen hierfür und für 3D-Modelle werden – wie auch schon zuvor beschriebenes Material – auf der Webseite zur Verfügung gestellt. Gleiches gilt für einen vollständigen Überblick über die im Projekt verwendeten Komponenten, Lösungsvorschläge und

Vorschläge zur Weiterentwicklung der Roboter usw. Es ist wichtig zu beachten, dass die Roboter auf viele verschiedene Arten und aus vielen verschiedenen Materialien konstruiert werden könn(t)en, weshalb die entwickelten LEGO-Handbücher nur eine von mehreren Lösungen sind.

Projektkonzept:

Das folgende Projekt basiert auf einer Kaffeebohnenzentrale, in der die Bohnen zunächst nach Qualität sortiert und dann im zugehörigen Lager verteilt werden. Die Bohnen wurden zuvor in der Zentrale von Hand sortiert und anschließend von Hand im Lager verteilt. Dies erfordert anstrengende körperliche und zudem zeitaufwändige Arbeit. Die Zentrale möchte daher diese beiden Prozesse automatisieren, weshalb zwei Roboterarten gebaut und programmiert werden sollen, die jeweils die entsprechenden Funktionen ausführen können.

> **Sortieren von Kaffeebohnen (**Farbsortierung**)**: Die Bohnen werden nach Qualität sortiert, die an ihrer Farbe erkennbar ist. Für dieses Projekt bestehen die Bohnen aus 3D-gedruckten Bauklötzen (rot, grün, blau und gelb). Daher muss der Roboter eine Bohne entgegennehmen, sie einer Farbkategorie zuordnen, vier zugehörige Kartons so drehen, dass der entsprechende Karton dem Förderband zugewandt ist, und schlussendlich die Bohne in diesen ablegen und die Kartons in ihre Ausgansposition zurückdrehen können.
>
> **Transportieren (**Folgen einer Linie**)**: Die sortierten Bohnen werden in Kartons verpackt, die im Rahmen des Projekts aus 3D-gedruckten Kartons (rot, grün, blau und gelb) bestehen. Die Farbe gibt an, an welchen der vier entsprechenden Bereiche im Lager sie geliefert werden sollen. Das Lager wurde mit schwarzen Linien auf dem Boden versehen, nach denen die Roboter navigieren sollen. Zur Vermeidung von Unfällen sollen die Roboter außerdem anhalten, wenn sich vor ihnen ein Gegenstand befindet, der im Weg steht. Dies können Mitarbeiter aus Fleisch und Blut oder andere Roboter sein.

Die Zentrale ist bereits mit einem Bereich ausgestattet, in dem die Sortierung stattfindet. Außerdem haben die Mitarbeiter kürzlich die Arbeit mit dem Aufbringen der Linien abgeschlossen, auf denen die Roboter navigieren sollen, siehe Abb. 22.

Wenn die Schülerinnen und Schüler gewählt haben, mit welchem der beiden Roboter sie arbeiten möchten, können sie das in den nächsten beiden Abschnitten beschriebene Verfahren weiterverfolgen.

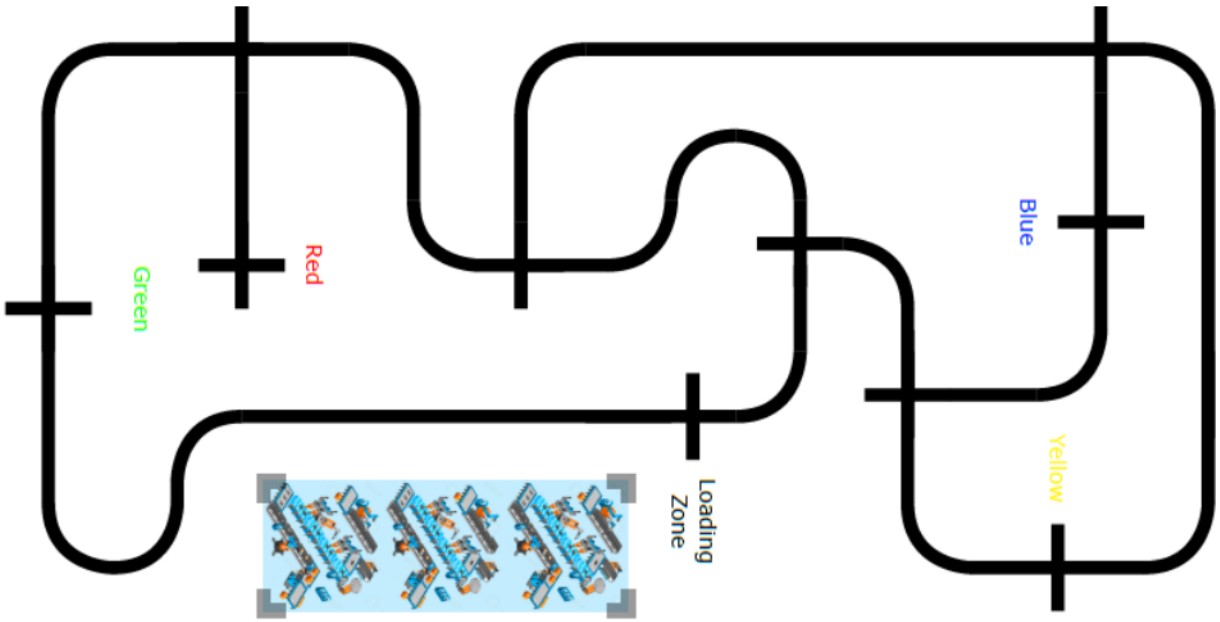


Abb. 22: Übersicht über die Ladezone und Beförderungsstrecken des Lagers

Die folgenden Videos zeigen Beispiele aus der Praxis – von Robotern, die für das automatische Sortieren und Transportieren von Kaffeebohnen entwickelt wurden.

Sortieren von Kaffeebohnen: https://www.youtube.com/watch?reload=9&v=i0nmIl-bNLI

Transportieren: https://www.youtube.com/watch?v=WIlS3vNSuQ4

Sortieren von Kaffeebohnen:

Der Roboter besteht aus zwei Hauptteilen, die zu einem zusammengefasst sind. Der erste Teil ist das Förderband, auf das die Kaffeebohnen zuerst gelegt werden. Dann werden sie unter einen Farbsensor verbracht, der ihre Farbe erfasst (rot, grün, blau oder gelb). Anschließend soll der Förderer sie das letzte Stück hin zu einem Karton und in diesen hinein verfrachten. Der zweite Teil ist der Mechanismus zum Drehen der entsprechenden Kartons, sodass die Bohnen, wenn sie das Ende des Förderbandes erreichen, in den richtigen Karton hineinfallen. Siehe nachstehende Abb. 23.

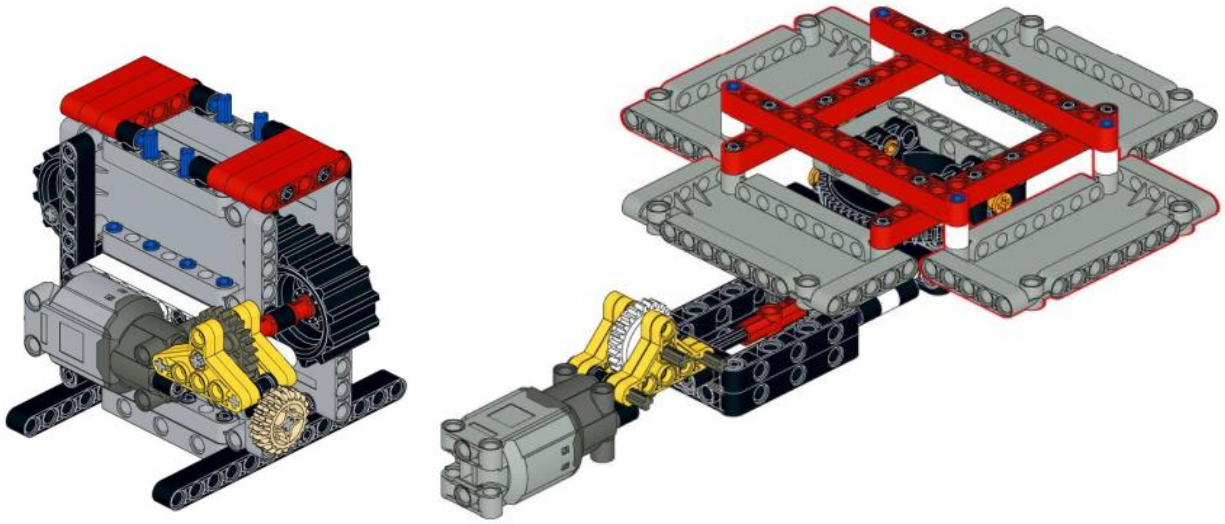


Abb. 23: LEGO-Modelle des Förderbandes und der Kartons

Erste Schritte:

1. Entwickeln des Roboters:
Um eine Arbeitsgrundlage zu haben, ist die erste Aufgabe die Konstruktion des Roboters selbst.

Aufgabe:

- Gehen Sie nach der LEGO-Anleitung ”Bean Sorter – Belt” vor und bauen Sie das Förderband.
- Gehen Sie nach der LEGO-Anleitung ”Bean Sorter – Rotator” vor und bauen Sie den Rotationsmechanismus für die Kartons.
- Gehen Sie nach der LEGO-Anleitung ”Bean Sorter – Combine” vor und bauen Sie die beiden Teile zusammen.
- Weisen Sie nun den Kartons ihre jeweile Farbe zu, ggf. mit einem Marker, und legen Sie fest, welche standardmäßig dem Förderband zugewandt sein soll.

Material:

- 2 x LEGO Power Functions Motor (Large).

2. Integrieren der Arduino-Plattform:
Jetzt soll der Arduino ins Spiel gebracht werden, also der Mikrocontroller zur Steuerung unseres Roboters, sowie die zugehörige Motorabschirmung, eine Erweiterung des Arduino, mit dem wir die Motoren steuern, und zusätzlich ein Steckbrett.

Aufgabe:

- Gehen Sie nach der Anleitung ”Bean Sorter – Integrate the Arduino platform” vor und fügen Sie Arduino, Motorabschirmung, Steckbrett und Akkusatz hinzu.

Material:

- Arduino Uno
- Motorabschirmung R3
- Steckbrett (halbe Größe)
- Akkusatz + Kabel
- Arduinomodul (3D-gedruckt)
- Steckbrettmodul (3D-gedruckt)
- Akkusatzmodul (3D-gedruckt)
- 2 x Stecker/Stecker-Kabel

3. Test der Motoren:
Im folgenden Schritt sollen die Motoren angeschlossen und gleichzeitig einige kleinere Tests durchgeführt werden, um sicherzustellen, dass sie wie vorgesehen funktionieren.

Aufgabe:
- Motoren mit der Motorabschirmung verbinden.
- Förderband 2 Sek. lang vorwärts laufen lassen – mit Höchstgeschwindigkeit.
- Förderband 2 Sek. lang rückwärts laufen lassen – mit halber Geschwindigkeit.
- Kartons 2 Sek. lang im Uhrzeigersinn drehen lassen – mit Höchstgeschwindigkeit.
- Kartons 2 Sek. lang gegen den Uhrzeigersinn drehen lassen – mit halber Geschwindigkeit.

Cheat sheets:
- Aktoren – DC-Motor
- Code – pinMode
- Code – digitalWrite
- Code – analogWrite
- Code – delay

Förderband:
Für diesen Teil empfiehlt sich das Erstellen einer neuen Programmdatei – aber speichern Sie unbedingt die vorherigen aus den *Ersten Schritten*.

1. Einsatz von Funktionen zur Steuerung des Förderbandes:
Funktionen sind eine gute Methode, um eine Codeabschnitt abzugrenzen, damit sie ihn immer wieder verwenden können, ohne dass wir den Code jedes Mal neu schreiben müssen. Gleichzeitig macht es den Code für den/die Anwender*in besser lesbar.

Aufgabe:
- Erstellen Sie eine Funktion ”startBelt”, die das Förderband startet.
- Erstellen Sie eine Funktion ”stopBelt”, die das Förderband anhält.
- Erstellen Sie eine Funktion ”deliverBean”, die die Bohne 'anliefert' (das Förderband startet und soll 5 Sek. lang laufen bis es wieder stoppt).
- Überprüfen Sie, dass alle Funktionen einwandfrei arbeiten.

Cheat sheets:

- Code – Functions

2. Integrieren des Farbsensors:

Vor dem Einsatz des Förderbandes müssen wir den Farbsensor integrieren und prüfen, ob er bestimmungsgemäß funktioniert.

Aufgabe:

- Gehen Sie nach der Anleitung ”Bean Sorter – Integrate Color-Sensor” vor und integrieren Sie den Farbsensor.
- Legen Sie eine 'Bohne' auf das Förderband unter den Farbsensor und geben Sie am Bildschirm ein, um welche Farbe es sich handelt. Führen Sie dies für alle Farben aus (Rot, Grün, Blau und Gelb).

Material:

- Farbsensor (TCS3200)
- Farbsensormodul (3D-gedruckt)
- 7 x Stecker/Buchse-Kabel

Cheat sheets:

- Sensors – Color-Sensor (TCS3200)
- Code – Variables
- Code – Serial Print

3. Auf erkannte Farben reagieren:

Wir können jetzt unser Förderband steuern und den Farbsensor einsetzen, um Farben zu erkennen. Bis dahin macht das Programm aber immer nur das Gleiche. Daher möchten wir nun eine Struktur für das Programm erstellen, das es ermöglicht verschiedene Aktionen auszuführen, je nachdem, welche Farbe der Farbsensor erkennt (Rot, Grün, Blau oder Gelb).

Aufgabe:

- Starten Sie das Förderband.
- Wenn der Farbsensor ”red” liest:
    - ”red” eingeben.
    - Förderband anhalten und die 'Bohne' abliefern.
- Überprüfen, dass es funktioniert – wenn ja, einen ”if else” für jede der anderen Farben herstellen (Grün, Blau und Gelb), der die gleiche Funktionalität wie die der Farbe Rot enthält.

Cheat sheets:

- Code – Conditionals.

Die Kartons:

Für diesen Teil empfiehlt sich das Erstellen einer neuen Programmdatei – aber speichern Sie die vorherige zum Förderband.

1. Funktionen zum Rotieren der Kartons einsetzen:

Wie beim Förderband möchten wir auch hier Funktionen zum Abgrenzen des Codes für die Motorsteuerung einsetzen, damit er wiederverwendet werden kann.

Aufgabe:

- Erstellen Sie eine Funktion ”startRotation”, die die Kartons in eine Drehung gegen den Uhrzeigersinn versetzt.
- Erstellen Sie eine Funktion ”stopRotation”, die die Drehung der Kartons anhält.
- Erstellen Sie eine Funktion ”rotatePosition”, die einen int als Parameter nimmt (nennen Sie den Parameter ”position”) – abgesehen vom Ausgeben des Parameters an das Programm bleibt diese Funktion ansonsten leer.
- Überprüfen Sie, dass die Funktionen einwandfrei arbeiten.

Cheat Sheet:

- Code – Functions (mit Parametern).

2. Schalter integrieren:

Bisher konnten wir die Kartons nur zeitabhängig hin und her rotieren, aber da die Zeit, die zum Rotieren einer Runde benötigt wird, davon abhängt, wie viel Energie noch in den Batterien vorhanden ist oder wie viel Schmutz ggf. in das Getriebe gelangt ist usw., ist dies keine optimale Lösung. Daher wollen wir einen Schalter in unseren Aufbau integrieren, mit dem wir jedes Mal die Rotation der Kartons um eine Position erkennen.

Aufgabe:

- Gehen Sie nach der Anleitung ”Bean Sorter – Integrate Switch” vor und integrieren Sie den Schalter.
- Überprüfen Sie durch Anzeigen seines Zustandes an das Programm, dass er funktioniert.

Material:

- Schalter
- Schaltermodul (3D-gedruckt)
- Widerstand 1 KΩ
- 3 x Stecker/Stecker-Kabel.

Cheat Sheet:

- Sensors – Switch.

3. Anzahl der Kartonrotationen überprüfen:

Da der Schalter bei jedem Passieren eines Kartons automatisch gedrückt wird, können wir damit prüfen, um wie viele Positionen sich die Kartons bewegt haben, indem wir die Anzahl des Drückens registrieren. Auf diese Weise können wir immer sicherstellen, dass die Kartons sich um genau die Anzahl der gewünschten Positionen verschieben. Hierzu müssen wir die Funktion "DeliverBean" abgeschlossen haben, damit sie die gewünschte Anzahl an Rotationen ausführt.

Aufgabe:

- In der Funktion ”deliverBean”:
  - Rotation der Kartons starten.
  - Neue Zählervariable namens ”i” erstellen und ihr den Wert 0 zuordnen.
  - Eine while-Schleife mit der Bedungung (i < position) erstellen.
    - Wenn Schalter gedrückt ist:
      - ”i” um 1 erhöhen.
      - Ca. 200 ms warten (je nach Rotationsgeschwindigkeit).
  - Kartonrotation anhalten.
- Funktion durch Aufrufen mit verschiedenen Parametern (1, 2, 3 usw.) überprüfen.
- Zählen Sie nun, wie oft die Kartons um eine Position rotieren müssen, bevor sie sich vor dem Förderband befinden, und wie oft sie rotieren müssen, bevor sie wieder an ihrem Ausgangspunkt sind. Schreiben Sie diese Zahlen auf ein Blatt Papier.

Förderband und Kartons kombinieren:

Für diesen Teil empfiehlt sich das Erstellen einer neuen Programmdatei – aber speichern Sie wieder unbedingt die vorherigen Funktionen.

1. Kombinieren des gesamten Aufbaus:

Jetzt ist es Zeit, das Förderband mit den Kartons zu kombinieren, damit der Roboter vollautomatisch funktioniert. Wir haben bereits fast alle dafür erforderlichen Arbeiten durchgeführt, insbesondere durch den Einsatz von Funktionen.

Aufgabe:

- Förderband starten.
- Wenn der Farbsensor ”red” liest:
  - ”red” an den Bildschirm ausgeben.
  - Förderband anhalten.
  - Kartons so rotieren lassen, dass der rote vor dem Förderband steht.
  - 'Bohne' abliefern.
  - Kartons so rotieren lassen, dass der rote erneut an seinem Ausgangspunkt steht.
- Überprüfen Sie, dass dies alles funktioniert – und erweitern Sie dann das Programm um die übrigen Farben (Grün, Blau und Gelb).

Varianten der Aufgabe:

Sie können die Aufgabe weiter variieren bzw. weiterentwickeln und so die Funktionen des Aufbaus erweitern, z. B. verschiedene LEDs zum Leuchten bringen, um anzuzeigen, welche Aktionen der Roboter

gerade ausführt. Eine weitere Möglichkeit wäre, einen Halter zu bauen, der mit Hilfe eines Servomotors nacheinander neue Bohnen auf das Förderband legt.

Transportieren:

Der Roboter ruht auf zwei Vorderrädern (die ihn antreiben) und einem Kugelrad. Siehe nachstehende Abb. 24.

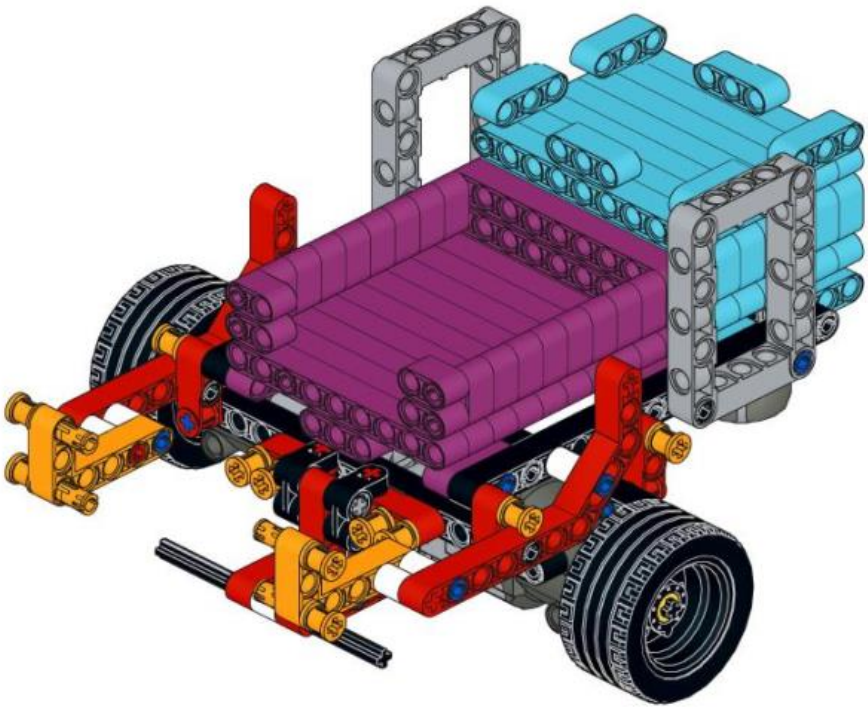


Abb. 24: LEGO-Model des Förderroboters

Erste Schritte:

1. Entwickeln des Roboters:

Die erste Aufgabe ist die Konstruktion des Roboters selbst, um eine Arbeitsgrundlage zu haben.

Aufgabe:

- Gehen Sie nach der LEGO-Anleitung ”Transporter – Transporter” vor und bauen Sie das Förderfahrzeug (Transporter).

Material:

- LEGO + 2 x LEGO Power Functions Motor (L).

2. Integrieren der Arduino-Plattform:
Jetzt soll der Arduino angewendet werden, also der Mikrocontroller zur Steuerung unseres Roboters, sowie die zugehörige Motorabschirmung, eine Erweiterung des Arduino, mit dem wir die Motoren steuern, sowie ein Steckbrett.

Aufgabe:
- Gehen Sie nach der Anleitung ”Transporter – Integrate the Arduino platform” vor und fügen Sie Arduino, Motorabschirmung, Steckbrett und Akkusatz hinzu.

Material:
- Arduino Uno
- Motorabschirmung R3
- Steckbrett (halbe Größe)
- Akkusatz + Kabel
- Arduinomodul (3D-gedruckt)
- Steckbrettmodul (3D-gedruckt)
- Akkusatzmodul (3D-gedruckt)
- 2 x Stecker/Stecker-Kabel.

3. Test der Motoren:
Jetzt ist es Zeit, die Motoren anzuschließen und gleichzeitig einige kleinere Tests durchzuführen, um sicherzustellen, dass alles wie gedacht funktioniert.

Aufgabe:
- Motoren mit der Motorabschirmung verbinden.
- Den Transporter 2 Sek. lang vorwärts laufen lassen – mit Höchstgeschwindigkeit.
- Den Transporter 2 Sek. lang rückwärts laufen lassen– mit halber Geschwindigkeit.
- Den Transporter 2 Sek. lang nach links drehen lassen – mit Höchstgeschwindigkeit.
- Den Transporter 2 Sek. lang nach rechts drehen lassen – mit halber Geschwindigkeit.

Cheat sheets:
- Aktoren – DC-Motor
- Code – pinMode
- Code – digitalWrite
- Code – analogWrite
- Code – delay

Nach Linien navigieren:
Für diesen Teil empfiehlt sich das Erstellen einer neuen Programmdatei nachdem die vorherigen Schritte gespeichert wurden.

1. Funktionen zum Steuern des Transporters einsetzen:

Funktionen sind eine gute Methode, um einen Codeabschnitt abzugrenzen, damit sie immer wieder verwendet werden können, ohne dass wir den Code jedes Mal neu schreiben müssen. Gleichzeitig macht es den Code für den/die Anwender*in besser lesbar.

Aufgabe:

- Erstellen Sie eine Funktion ”driveForward”, die den Roboter vorwärts fahren lässt.
- Erstellen Sie eine Funktion ”crossLine”, die den Roboter ca. 5 cm vorwärts fahren und ihn dann wieder anhalten lässt.
- Erstellen Sie eine Funktion ”turnLeft”, die den Roboter eine viertel Drehung nach links ausführen lässt.
- Erstellen Sie eine Funktion ”turnRight”, die den Roboter eine viertel Drehung nach rechts ausführen lässt.
- Erstellen Sie eine Funktion ”uTurn”, die den Roboter eine halbe Drehung ausführen lässt – Richtung egal.
- Erstellen Sie eine Funktion ”stopRobot”, die den Roboter für 1 Minute anhalten lässt.
- Überprüfen Sie, dass alle sechs Funktionen einwandfrei arbeiten.

Cheat sheets:

- Code – Functions.

2. Integrieren der IR-Sensoren:

Bevor wir den Roboter zum Folgen der Bodenlinien bringen können, müssen wir die beiden IR-Sensoren einbauen und überprüfen, ob sie richtig funktionieren.

Aufgabe:

- Gehen Sie nach der Anleitung ”Transporter – Integrate IR-Sensors” vor und integrieren Sie die IR-Sensoren.
- Zunächst den Wert des linken IR-Sensors ablesen und ans Programm weitergeben; danach die Werte für Weiß und Schwarz notieren. Den Vorgang für den rechten IR-Sensor wiederholen.

Material:

- IR-Sensor (QRE1113, analog).
- IR-Sensormodul (3D-gedruckt)
- 6 x Stecker/Stecker-Kabel

Cheat sheets (Spickzettel):

- Sensors – IR-sensor (QRE1113, analog).
- Code – Variables.
- Code – analogRead.
- Code – Serial Print.

3. Folgen der Linie:

Jetzt können wir den Roboter mithilfe der IR-Sensoren so steuern, dass er einer Linie folgt. Dazu müssen wir für jeden IR-Sensor den Wert berechnen, der genau zwischen dem Wert für Weiß und dem Wert für Schwarz liegt. Wir nennen diesen Wert den Schwellenwert. Um einer Linie zu folgen, lassen wir die Motoren laufen, wenn ihr jeweiliger IR-Sensor auf Weiß sieht, andernfalls müssen sie anhalten. Auf diese Weise korrigiert der Roboter automatisch, wenn er droht vom Kurs abzukommen und dabei die Linie zu überfahren. Auf die gleiche Weise stoppt er auch automatisch, wenn er an eine Kreuzung kommt, weil beide Sensoren jetzt Schwarz registrieren. Beginnen Sie mit der Simulation, indem Sie den Roboter von Hand steuern.

Aufgabe:

- Den linken Motor zum Laufen veranlassen, wenn der linke IR-Sensor weniger als den Schwellenwert liest; sonst soll er anhalten.
- Den rechten Motor zum Laufen veranlassen, wenn der rechte IR-Sensor weniger als den Schwellenwert liest; sonst soll er anhalten.
- Überprüfen Sie, dass der Roboter jetzt der Linie zu folgen vermag – und anhält, wenn er eine Kreuzung erreicht.

Cheat sheets:

- Code – Conditionals.

4. Folgen der Linie – als Funktion:

Bisher stoppt unser Roboter automatisch, wenn er eine Kreuzung erreicht, aber er fährt auch von selbst wieder an, wenn man ihn manuell über die Kreuzung hinausschiebt oder auf eine neue Linie stellt. Wenn wir also unsere in den Funktionen "turnLeft" und "turnRight" gespeicherten Abbiegebefehle anwenden möchten, wenn wir eine Kreuzung erreichen, gibt es ein Problem. Versuchen Sie mal dies mit dem Befehl auszuführen – der Roboter wird sich jetzt einfach nur drehen und der Linie überhaupt nicht mehr folgen. Das liegt daran, dass die Liniencodes, nach denen er der Linie folgt, im Bruchteil einer Sekunde ausgeführt werden, woraufhin der Befehl zum Drehen erneut aufgerufen wird. Wir müssen daher eine eigenständige Funktion erstellen, die bewirkt, dass der Roboter der Linie folgt, bis er eine Kreuzung erreicht, wonach die Funktion endet.

Aufgabe:

- Erstellen Sie eine Funktion ”followLine”.
    - Verwenden Sie eine while-Schleife mit der Bedingung (true).
        - Den linken Motor zum Laufen veranlassen, wenn der linke IR-Sensor weniger als den Schwellenwert liest; sonst soll er anhalten.
        - Den rechten Motor zum Laufen veranlassen, wenn der rechte IR-Sensor weniger als den Schwellenwert liest; sonst soll er anhalten.
        - Wenn beide IR-Sensoren weniger als den Schwellenwert lesen, halten Sie beide Motoren an und unterbrechen Sie die while-Schleife.
- Überprüfen Sie, dass die Funktion einwandfrei arbeitet.
- Überprüfen Sie sodann, ob sie zusammen mit den Drehfunktionen funktioniert – und zudem die Funktion des Geradeausfahrens an einer Kreuzung –, indem Sie die untenstehenden Funktionen aufrufen und daraufhin beobachten, ob der Roboter die vorgesehene Route fährt.

Denken Sie daran, dass nach Ablauf einer Minute die Funktion ”stopRobot” zu Ende ist, und der Ablauf (die Sequenz) von vorn beginnt.

- followLine
- turnLeft
- followLine
- crossLine
- followLine
- turnRight
- followLine
- uTurn
- stopRobot

Cheat sheets:

- Code – While loop (mit Unterbrechung).

Nach Umfeld navigieren:

Bevor Sie fortfahren, empfiehlt es sich das erstellte Programm zu speichern und eine Kopie zu erstellen, um anschließend in der Kopie weiterzuarbeiten.

1. Integrieren des Abstandssensors:

Der Roboter kann jetzt gemäß den Linien auf dem Boden des Lagers navigieren, aber was ist, wenn sich ein anderer Roboter vor ihm befindet oder ein Mitarbeiter übersehen hat, dass er direkt auf ihn zusteuert? Um mögliche Probleme und Verletzungen zu vermeiden, bringen wir jetzt einen Abstandssensor ins Spiel, mit dem der Roboter die Entfernung zum nächsten Objekt messen kann. Dadurch können wir im nächsten Schritt den Roboter so programmieren, dass er anhält, falls ein Objekt ihm (oder er dem Objekt) zu nahe gekommen ist.

Aufgabe:

- Gehen Sie nach der Anleitung ”Transporter – Ultrasound-Sensor” vor und integrieren Sie den Abstandssensor.
- Geben Sie die Entfernung zum nächsten Objekt an den Bildschirm aus.

Material:

- Abstandssensor (HC-RS04).
- Abstandssensormodul (3D-gedruckt)
- 4 x Stecker/Stecker-Kabel.

Cheat sheets:

- Sensors – Ultrasonic-Sensor (HC-RS04).

2. Beim einem im Weg stehenden Objekt Roboter anhalten:

Unsere Funktion ”followLine” wollen wir nun so aufwerten, das der Roboter stoppt, wenn der Abstand zu einem Objekt vor ihm 20 cm unterschreitet.

Aufgabe:

- Funktion ”followLine” wie folgt aufwerten:
  - Zuerst wird in der while-Schleife der Abstand zum nächsten Objekt gelesen.
  - Werten Sie die Bedingungen für erlaubtes Motorlaufen so auf, dass die Bedingung jetzt Folgendes vorsieht:
    - Wenn linker IR-Sensor unter Schwellenwert und Abstand über 20 cm, dann linken Motor laufen lassen, sonst anhalten.
    - Wenn rechter IR-Sensor unter Schwellenwert und Abstand über 20 cm, dann rechten Motor laufen lassen, sonst anhalten.

Karton mit Bohnen 'anliefern':

Bevor Sie fortfahren, empfiehlt es sich auch hier das erstellte Programm zu speichern und eine Kopie zu erstellen, um anschließend in der Kopie weiterzuarbeiten.

1. Integrieren des Servomotors:

Mitarbeiter können jetzt einen Karton auf dem Roboter platzieren und ihn so programmieren, dass er einer Route folgt – aber er kann den Karton bei Ankunft noch nicht selbst abliefern. Wir wollen ihn daher mit einem Servomotor ausstatten, der die Kartons von der Ladefläche kippen kann, woraufhin der Roboter zurückfährt, um einen neuen Karton abzuholen.

Aufgabe:

- Gehen Sie nach der Anleitung ”Transporter – Servo-Motor” vor und integrieren Sie den Servomotor.
- Überprüfen Sie den Servomotoren auf ordnungsgemäße Funktionalität, indem Sie ihn in verschiedene Positionen versetzen.

Material:

- Servomotor.
- 6 x Stecker/Stecker-Kabel.

Cheat sheets:

- Effectors – Servo-Motor.

2. Karton 'anliefern':

Jetzt muss der Roboter nur noch den Karton freigeben, wenn er den vorgesehenen Bestimmungsort erreicht hat.

Aufgabe:

- Erstellen Sie eine Funktion ”deliverBox”, die mittels des Servomotors die Ladefläche in Kippposition bringt, damit der Karton herunterfällt, woraufhin der Servomotor in seine Ausgangsposition zurückkehrt.
- Überprüfen Sie, dass es entsprechend funktioniert und der Roboter nunmehr einem Streckenverlauf folgen kann, auf dem er am richtigen Ort den beförderten Karton ablädt.

Varianten der Aufgabe:

Sie können sich gern eigene Ideen für mögliche Erweiterungen ausdenken, z. B. verschiedene LEDs, die anzeigen, welche Aktionen der Roboter derzeit ausführt. Oder der Einsatz eines LDRs (Light Dependent Resistor) zwecks Feststellung, ob sich ein Karton auf der Ladefläche des Roboters befindet.