Was europäische Lehrkräfte voneinander lernen können

SCIENCE ON STAGE DEUTSCHLAND

THE EUROPEAN NETWORK FOR SCIENCE TEACHERS

Coding im MINT-Unterricht

Deutsch

<Impressum>

<Herausgeber>
Science on Stage Deutschland e.V.
Am Borsigturm 15
13507 Berlin

<Hauptkoordinator>
↳ **Dr. Jörg Gutschank**, Leibniz Gymnasium | Dortmund International School, Dortmund, Deutschland
Vorsitzender Science on Stage Deutschland e.V.

<Koordinatoren>
↳ **Sebastian Funk**, Villa Wewersbusch, Velbert-Langenberg, Deutschland
Vorstand Science on Stage Deutschland e.V.
↳ **Jean-Luc Richter**, Lycée Jean-Baptiste Schwilgué, Sélestat, Frankreich
Stellvertretender Vorsitzender Science on Stage Frankreich
↳ **Bernard Schriek** (i.R.), Marien-Gymnasium, Werl, Deutschland

<Projektmanagement und Redaktion>
↳ **Daniela Neumann**, Projektmanagerin
Science on Stage Deutschland e.V.
↳ **Stefanie Schlunk**, Geschäftsführerin
Science on Stage Deutschland e.V.
↳ **Johanna Schulze**, Stellvertretende Geschäftsführerin
Science on Stage Deutschland e.V.

<Revision und Übersetzung>
Translation-Probst AG

<Gestaltung>
WEBERSUPIRAN.berlin

<Illustration>
Rupert Tacke, Tricom Kommunikation und Verlag GmbH

<Text- und Bildnachweise>
Die Autorinnen und Autoren haben die Bildrechte für die Verwendung in dieser Publikation nach bestem Wissen geprüft und sind für den Inhalt ihrer Texte verantwortlich.

<Ermöglicht durch>
SAP SE

<Bestellungen>
www.science-on-stage.de
info@science-on-stage.de

<ISBN PDF-Fassung>
978-3-942524-60-5

Diese Publikation ist lizenziert unter einer Creative Commons Namensnennung - Weitergabe unter gleichen Bedingungen 4.0 International Lizenz:
https://creativecommons.org/licenses/by-sa/4.0/.

1. Auflage 2019

© Science on Stage Deutschland e.V.

<Inhalt>

Grußwort EU-Kommission <04>

Grußwort SAP SE <05>

Vorwort <06>

Autorinnen und Autoren <07>

Umwelt 4.0

<08> Coding H_2O

<14> Wasserwelten

<20> VPLS - Vacation Plant Life Saver

<26> Der Zauberhandschuh

Wissenschaft in 0 und 1

Science Friction <30>

Rolling Sounds <36>

Physics Engine <42>

SMB - Science Magic Box <48>

Unsere Welt unter Mikrokontrolle

<54> CoALA - Code a Little Animal

<58> Datenfluss

<62> Schiff ahoi!

<68> Wie wird programmiert?

<72> Informatikunterricht mit Snap!

<73> Meet and Code

<74> Weiterführende Materialien & Aktivitäten

<75> Science on Stage

<Grußwort>

Ich freue mich sehr, diese Broschüre zur Förderung des Unterrichts in Mathematik, Informatik, Naturwissenschaften und Technik – den wichtigen MINT-Fächern – sowie das Projekt *Coding im MINT-Unterricht* allgemein zu unterstützen.

Die MINT-Fächer sind von entscheidender Bedeutung für die Gestaltung der Zukunft eines wettbewerbsfähigen und stabilen Europas. Der Aufbau von Wissen und Spitzenleistungen ist hier ein wesentlicher Bestandteil unseres Bestrebens, bis 2025 einen echten europäischen Bildungsraum zu schaffen. Und dennoch stehen wir vor einer Qualifizierungslücke. Wir müssen mehr tun, um MINT-Fächer in Europa zu fördern. Vor allem das Unterrichten von MINT-Fächern muss eine attraktive Berufswahl sein, und wir brauchen mehr Vorbilder, insbesondere Frauen.

Dabei geht es nicht nur um Wirtschaftswachstum und Entwicklung. MINT-Ausbildung muss inklusiv sein, damit Schülerinnen und Schüler mit unterschiedlichen Fähigkeiten und Hintergründen dazu befähigt werden, sich aktiv zu beteiligen und das Beste aus ihren Talenten herauszuholen. Um die Zukunft zu gestalten, brauchen junge Menschen die richtigen Kompetenzen und Einstellungen – und MINT-Kenntnisse müssen hierbei im Mittelpunkt stehen.

Durch Wissenschaft können wir vieles lernen, sowohl über unsere Vergangenheit als auch unsere Gegenwart, was uns dazu befähigt eine bessere, zunehmend wissensbasierte Zukunft zu gestalten. Damit zukünftige Generationen in einer Gesellschaft leben können, die ein größeres Bewusstsein für die Welt hat, in der wir leben.

Ich möchte Science on Stage Deutschland e. V., das europäische Netzwerk für MINT-Lehrkräfte, das dieses Projekt konzipiert hat, und SAP SE für die Unterstützung von Coding-Projekten danken.

Lehrkräfte aus sieben europäischen Ländern entwickelten konkrete Beispiele und praktische Tipps zum Erwerb von Programmierfähigkeiten. Dies zeigt, was möglich ist, wenn wir zusammenkommen, wie viel wir gemeinsam haben und wie viel wir voneinander lernen können.

Projekte wie *Coding im MINT-Unterricht* spielen eine Schlüsselrolle bei der Förderung von MINT-Fächern in Schulen und helfen Europas Jugendlichen, unverzichtbare Kompetenzen zu erwerben, die sie für ein erfolgreiches Leben benötigen.

Ich gratuliere allen Beteiligten und hoffe, dass ihre Beispiele andere inspirieren.

Tibor Navracsics
Europäischer Kommissar für Bildung, Kultur, Jugend und Sport

<Grußwort>

Laut einer Prognose des Weltwirtschaftsforums werden etwa 65 Prozent der Kinder, die heutzutage in die Schule kommen, in Berufen arbeiten, die es noch gar nicht gibt. Bekannt ist jedoch, dass diese Kinder besser Fuß fassen werden, wenn sie bestimmte technische Fähigkeiten mitbringen; denn die Digitalisierung unserer Wirtschaft ist unaufhaltbar. Neben Lesen, Schreiben und Rechnen ist somit der Umgang mit neuen Technologien zu einem zentralen Thema für die Bildung geworden.

Die SAP engagiert sich seit Jahren mit europaweiten Initiativen für die Aus- und Weiterbildung von Kindern und Jugendlichen. Darunter in Themengebieten der Robotik *(First Lego League)* oder Programmiertechnik *(Meet and Code)*. Wir möchten junge Menschen spielerisch an neue Technologien heranführen und ihnen so einen besseren Start in ihre berufliche Zukunft ermöglichen.

Viele Lehrkräfte möchten ihren Schülerinnen und Schülern heute ebenfalls technische und digitale Grundkenntnisse mit auf den Weg geben. Dazu braucht es nicht zwingend Experten, sondern praxisnahe und erprobte Arbeitsunterlagen für das Erlernen technischer Fertigkeiten in verschiedenen Unterrichtsfächern und Lernstufen.

So fördern wir darüber hinaus den Erwerb digitaler Kompetenzen im Schulalltag und stellen beispielsweise Lehrmaterialien für MINT-Fächer zur Verfügung. Zusammen mit Science on Stage Deutschland haben wir bereits zahlreiche Schulförderungsprojekte ins Leben gerufen, darunter auch die aktuelle Unterrichtsausgabe *Coding im MINT-Unterricht* zur Förderung der Programmierkenntnisse von Lehrkräften.

Ich freue mich über die Zusammenarbeit an einem weiteren, sehr zielgerichteten Projekt mit Science on Stage Deutschland und bin überzeugt, dass auch diese Ausgabe ein voller Erfolg wird. Mein Dank gilt ebenso den Lehrkräften, die sich für dieses Thema stark gemacht haben.

Michael Kleinemeier
Mitglied des Vorstands, SAP SE

<Vorwort>

Diese Broschüre ist in vieler Hinsicht etwas Besonderes. Was genau meine ich mit „besonders“?

Programmieren ist eine Grundkompetenz in unserer heutigen Welt und von großer Bedeutung in Mathematik, Informatik, Naturwissenschaften und Technik (MINT). Die Fähigkeit, Maschinen zu programmieren, wird in allen Bereichen unseres Lebens immer wichtiger. Dies ist längst nicht mehr eine Aufgabe, die wir den Informatikern überlassen können. Deshalb muss das Programmieren nicht nur im Informatikunterricht, sondern auch in allen anderen Fächern gelehrt werden. Die europäischen Lehrpläne für MINT-Fächer berücksichtigen diesen Bedarf oft jedoch nicht ausreichend. Mit *Coding im MINT-Unterricht* entwickelte Science on Stage, das europäische Netzwerk für MINT-Lehrkräfte, Unterrichtseinheiten zur Schließung dieser Kompetenzlücke. Das Hauptziel des Netzwerks ist es, eine Plattform für MINT-Lehrkräfte in Europa zu bieten, über die Best-Practice-Ideen ausgetauscht werden können. Science on Stage erreicht so 100 000 Lehrkräfte in über 30 Mitgliedstaaten.

Einige der besten Lehrkräfte Europas stellen ihre Ideen in dieser Broschüre vor: 23 Lehrkräfte aus sieben Ländern trafen sich während des 18-monatigen Projekts und tauschten ihre Ideen zum Thema Programmieren im MINT-Unterricht aus. Alle beteiligten Lehrkräfte investierten viel Zeit und Energie in diesen Prozess.

Aus jedem der sieben Länder schlossen sich eine MINT-Lehrkraft und eine Informatiklehrkraft als Team zusammen, um Unterrichtskonzepte mit einem Team aus mindestens einem anderen Land auszutauschen. Sie besprachen und bewerteten diese Konzepte in ihren eigenen Klassen, um sicherzustellen, dass das Material, das Sie nun in den Händen halten, im Unterricht eingesetzt werden kann. Alle Ideen wurden von unseren Experten, nämlich den Lehrkräften selbst, gründlich getestet.

Der Schwerpunkt liegt auf dem Programmieren von kleinen elektronischen Geräten wie Arduino, Calliope mini oder Raspberry Pi. Es sind preisgünstige Mikrocontroller, die tiefergehende Aufgaben lösen können – ideal für den Einsatz im Unterricht.

Die Teilnehmenden schufen so elf Unterrichtseinheiten in den Bereichen „Wissenschaft in 0 und 1“, „Die Welt unter Mikrokontrolle“ und „Umwelt 4.0“. Es sind wunderbare Beispiele dafür, was Sie selbst in den verschiedenen Lehrplänen der Biologie, Chemie und Physik umsetzen könnten.

Das Projekt *Coding im MINT-Unterricht* war nur dank dem großen Engagement unserer enthusiastischen Teilnehmerinnen und Teilnehmer möglich, die diese ganze Arbeit in ihrer Freizeit und zusätzlich zu ihrem Lehrberuf leisteten. Ganz herzlichen Dank an Euch alle für diese großartige, inspirierende Broschüre! Herzlichen Dank auch an die weiteren Koordinatoren Jean-Luc Richter, Bernd Schriek und Sebastian Funk, die die nicht einfache Aufgabe hatten, die vielen heterogenen Gedanken und Ideen der Lehrkräfte mit unterschiedlichem kulturellen und wissenschaftlichen Hintergrund in einem einheitlichen Werk zu verdichten. Unser Dank geht auch an SAP SE und insbesondere an Gabriele Hartmann und Jens Mönig. Ohne die stete Unterstützung dieser professionellen Partner wäre eine solches Projekt nicht möglich.

Coding im MINT-Unterricht ist ein ganz außergewöhnliches Werk, das von Lehrkräften für ihre Kolleginnen und Kollegen in Europa und darüber hinaus erarbeitet und getestet wurde. Lassen Sie sich von dieser Broschüre dazu inspirieren, schon bald Ihr eigenes Projekt in Ihrer Klasse zu starten!

Dr. Jörg Gutschank
Vorsitzender Science on Stage Deutschland e.V.

<Autorinnen und Autoren>

<Nachname>	<Vorname>	<Land>	<Kapitel>
Abad Nebot	Immaculada	Spanien	Unsere Welt unter Mikrokontrolle
Botelho	Lúcio	Portugal	Umwelt 4.0
Compte Jové	Pere	Spanien	Unsere Welt unter Mikrokontrolle
Fernandes	Liliana	Portugal	Umwelt 4.0
Funk	Sebastian	Deutschland	Koordinator
Georgoulakis	Georgios	Griechenland	Wissenschaft in 0 und 1
Giurgea	Mihaela Irina	Rumänien	Wissenschaft in 0 und 1
Gutschank	Jörg	Deutschland	Koordinator Wissenschaft in 0 und 1
Hančl	Mirek	Deutschland	Unsere Welt unter Mikrokontrolle
Ivarra	Luc	Belgien	Wissenschaft in 0 und 1
Karagiorgou	Eleftheria	Griechenland	Unsere Welt unter Mikrokontrolle
Lőkös	Annamária	Rumänien	Umwelt 4.0
Meier	Andreas	Deutschland	Umwelt 4.0
Mestvirishvili	Ilia	Georgien	Wissenschaft in 0 und 1
Nicolini	Marco	Belgien/Italien	Wissenschaft in 0 und 1
Padin	Beatriz	Spanien	Umwelt 4.0
Poncela	Elena	Spanien	Umwelt 4.0
Rațiu	Camelia Ioana	Rumänien	Wissenschaft in 0 und 1
Reis	Jorge	Portugal	Umwelt 4.0
Richter	Jean-Luc	Frankreich	Koordinator Umwelt 4.0
Schriek	Bernard	Deutschland	Koordinator Unsere Welt unter Mikrokontrolle
Shapakidze	David	Georgien	Wissenschaft in 0 und 1
Toma	Corina Lavinia	Rumänien	Wissenschaft in 0 und 1
Tsiliki	Sevasti	Griechenland	Unsere Welt unter Mikrokontrolle
Tsoutsoudakis	Astrinos	Griechenland	Wissenschaft in 0 und 1
van der Byl	Sonja	Deutschland	Umwelt 4.0
Winckler	Julia	Deutschland	Unsere Welt unter Mikrokontrolle

Besonderen Dank an Gabriele Hartmann und Jens Mönig von SAP SE für ihre Unterstützung!
Wir bedanken uns recht herzlich bei Holger Bach und Paul Nugent für ihre redaktionelle Unterstützung!

Coding H_2O

<Autorin> Beatriz Padin
<Autorin> Elena Poncela

<Info>

<Schlagwörter> Wasser, Sensoren, Reinigungsgrad, Datenerfassung, Umwelt, Verdunstung, Kondensation, Lösungen, Mischungen, Design, Wärme und Temperatur, Wärmeleiter und Isolatoren, Sonnenenergie, Infrarotstrahlung, Reflexion

<Unterrichtsfächer> Physik, Umweltwissenschaft, Chemie, Informatik, Mathematik

<Altersgruppe> 13–15 Jahre

<Hardware> Arduino[1], Calliope mini[2], Raspberry Pi[3]

<Programmiersprache> Arduino[4], Python[5], Blockprogrammierung

<Programmierniveau> mittel

<Zusammenfassung>

Die Schülerinnen und Schüler entwickeln, konstruieren und testen einen Solar-Destillierapparat zur Reinigung von Wasser. Sie programmieren Sensoren, die den Reinigungsgrad ihrer Solar-Destillierapparate messen.

<Vorstellung des Konzepts>

Wir behandeln die folgenden physikalischen Konzepte:

- Aggregatzustandswechsel (insbesondere Verdunstung und Kondensation) und ihre Hauptmerkmale
- Faktoren, die den Verdunstungsprozess beeinflussen (Temperatur, Größe der Oberfläche usw.)
- Einfluss der Temperatur: Energie und Wechsel des Aggregatzustandes
- Unterschied zwischen erneuerbaren (Sonnenenergie) und nicht erneuerbaren Energiequellen
- Infrarotstrahlung und ihre Rolle beim Transport der Sonnenenergie
- Eigenschaften von Strahlung, insbesondere ihrer Reflexion
- Methoden zur Trennung von Mischungen
- Lösungen: was man darunter versteht, Konzentration (g/l und Massenanteil usw.)

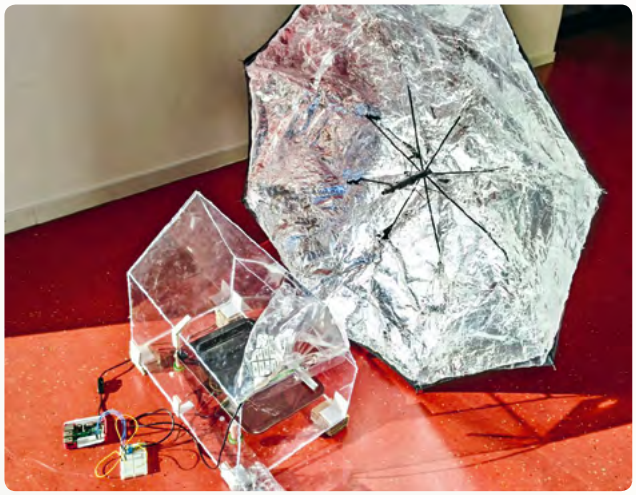
Je nach Vorwissen lernen die Schülerinnen und Schüler einige dieser Konzepte selbstständig kennen, während andere eine Erklärung durch die Lehrkraft erfordern und dann experimentell überprüft werden.

Ziel unseres Projekts ist es, einen hochleistungsfähigen Solar-Destillierapparat zur Wasserreinigung zu entwickeln. Hierbei darf nur mit Sonnenenergie gearbeitet werden. Zunächst wird der Reinigungsgrad des Solar-Destillierapparats ermittelt, indem der Volumenanteil des gewonnenen gereinigten Wassers berechnet wird. Anschließend programmieren die Schülerinnen und Schüler verschiedene Sensoren, um die Effizienz ihrer Designs zu untersuchen.

Das Programmieren der Sensoren steht im Mittelpunkt dieser Aktivität. Hierzu benötigen die Lehrkräfte Programmiersprachenkenntnisse und ein grundlegendes Verständnis der Hardware. Sie müssen den Schaltkreis aufbauen, der die Sensoren mit dem Mikrocontroller verbindet. Zum Programmieren der Sensoren können sie je nach ihren Kenntnissen mit Blockprogrammierung (Calliope mini[2], Snap4Arduino[6] usw.) oder Textprogrammierung (Arduino[4], Python[5] usw.) arbeiten.

<Praktische Umsetzung>

Diese Einheit besteht aus drei Teilen: Entwicklung des Solar-Destillierapparats, Programmierung der Sensoren und Testen des Solar-Destillierapparats.



Solar-Destillierapparat

<Erster Teil: Entwicklung des Solar-Destillierapparats>

Das Projekt wird der Klasse vorgestellt und die Schülerinnen und Schüler testen gemeinsam ihr erstes Beispiel eines Solar-Destillierapparats, indem sie das Volumen des gesammelten gereinigten Wassers messen und den Reinigungsgrad des Apparats berechnen. Dabei beschäftigen sie sich mit verschiedenen physikalischen Konzepten wie beispielsweise Aggregatzustandswechsel, Lösungen und Sonnenenergie.

Im Anschluss werden sie aufgefordert, dieses erste Modell zu verbessern. Dazu arbeiten sie in Teams von je 2–3 Schülerinnen und Schülern. Diese Phase der Unterrichtseinheit kann in verschiedene Aufgaben unterteilt werden:

1. Recherche über Solar-Destillierapparate, ihre Funktionsweise, verschiedene bestehende Designs usw. Dabei sollen die Schülerinnen und Schüler über folgende Fragen nachdenken:

a. Verdunstung: Welche Hauptfaktoren beeinflussen diesen Prozess? Berücksichtige die Oberfläche, auf der sich das schmutzige Wasser befinden soll. Ist es besser, eine große oder eine kleine Oberfläche zu haben, mit einer größeren oder einer geringeren Tiefe? Spielt die Farbe der Oberfläche eine Rolle?
b. Kondensation: Was wird benötigt, damit Kondensation stattfindet? Braucht es für die Kondensation von Wasser eine große oder eine kleine Oberfläche? Wie erreichst du, dass das gereinigte Wasser dorthin fließt, wo es gesammelt werden soll?
c. Strahlung: Wie kannst du die Infrarotstrahlung maximieren, die auf deinen Solar-Destillierapparat auftrifft? Wie erreichst du die höchstmögliche Temperatur in deinem Solar-Destillierapparat? Du könntest eine Oberfläche mit Alufolie verkleiden, um das Sonnenlicht in den Solar-Destillierapparat zu lenken.

2. Die Teams entwickeln ihre Apparate und erklären sie der Lehrkraft.
3. Nachdem der Entwurf mit der Lehrkraft abgestimmt wurde, müssen die Schülerinnen und Schüler geeignete Materialien finden (in der Schule, zu Hause, online bestellen usw.) und den Destillierapparat bauen.
4. Die Schülerinnen und Schüler bestimmen ohne Sensoren den Reinigungsgrad ihrer Solar-Destillierapparate. Mit einem Messzylinder messen sie nicht nur das Volumen des schmutzigen Wassers, sondern auch das Volumen des gesammelten gereinigten Wassers und wenden dabei folgende Gleichung an:

$$\textit{Reinigungsgrad} = \frac{\textit{Volumen des gesammelten Wassers}}{\textit{Volumen des schmutzigen Wassers}}$$

Für den zweiten und dritten Teil erhalten die Schülerinnen und Schüler eine Schritt-für-Schritt-Anleitung mit Aufgaben und Fragen.

Ziel ist es, ihnen eine Auswahl an Sensoren, Programmiersprachen und Hardware anzubieten, wobei hier die verschiedenen Voraussetzungen in jeder Schule/Klasse (verfügbare Materialien, bekannte Programmiersprachen usw.) zu beachten sind.

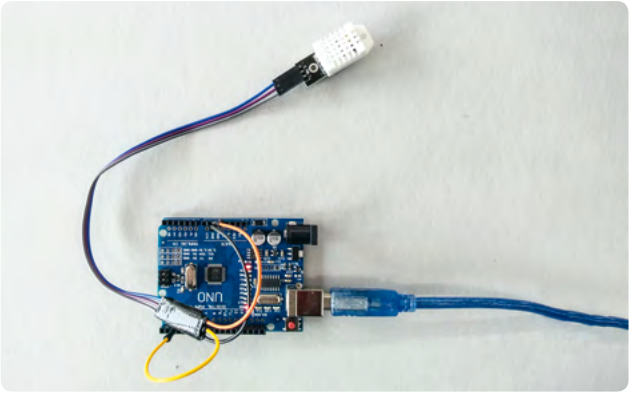
<Zweiter Teil: Programmieren der Sensoren>

Nur die besten Solar-Destillierapparate werden mit Sensoren getestet. In diesem Teil gilt:

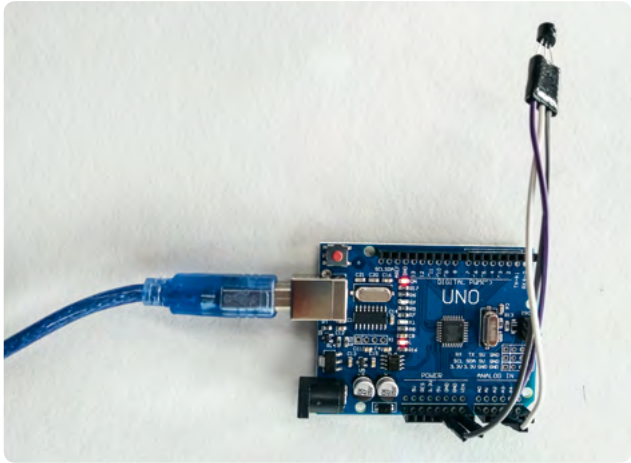
1. Wähle aus, welchen Mikrocontroller, welche Programmiersprache und welche Sensoren du benutzen möchtest. Die Beantwortung der folgenden Fragen hilft dir, die beste Wahl zu treffen:
   a. Nutzt du Blockprogrammierung oder Textprogrammierung? Wenn du dich für Blockprogrammierung entscheidest, überlege, ob du für einen Calliope mini[2] oder alternativ für einen Raspberry Pi[3] mit Scratch[8] programmieren willst. Wenn du dich für Textprogrammierung entscheidest, kannst du beispielsweise Arduino[1] verwenden oder du programmierst einen Raspberry Pi[9] mit Python[5].
   b. Verwendest du digitale oder analoge Sensoren? Bei analogen Sensoren ist Arduino eine sehr gute Wahl.
2. Um einfach zu beginnen, wähle höchstens zwei Sensoren, die du verwenden möchtest, z. B. zur Messung von Temperatur, Feuchtigkeit, Regen oder Infrarotstrahlung. Bevor du dich entscheidest, berücksichtige die technischen Eigenschaften der Sensoren: Ist das Ausgangssignal analog oder ist es ein digitales Signal mit lediglich zwei möglichen Werten (richtig/falsch)? In welcher Beziehung stehen Ausgangssignal und die gemessene Größe? Sind sie direkt proportional? Wird das Ausgangssignal stärker, wenn der Messwert abnimmt?
3. Baue den Schaltkreis auf, der deinen Sensor mit dem Mikrocontroller verbindet. Benutze die ergänzenden Materialien[7], die du bekommen hast, oder suche Beispiele im Internet.
4. Programmiere die Sensoren und beachte folgende Schritte:
   a. Notiere, was dein Programm leisten soll. Berücksichtige dabei die Eigenschaften der von dir gewählten Sensoren. Soll dein Programm nur den Wert der gemessenen Größe ausgeben oder auch die Maximal- und Minimalwerte? Gibt dein Sensor den tatsächlichen Wert der Größe aus, oder sind weitere Umrechnungen nötig?
   b. Schreibe dein Programm. Denke daran, deinen Code mit Kommentaren zu versehen. Du kannst dazu die ergänzenden Materialien[7], die du von deiner Lehrkraft bekommen hast, zu Hilfe nehmen.
5. Teste dein Programm. Funktioniert das Programm wie vorgesehen?
   Beispiele:
   ↳ Mit einem Flammensensor und Arduino UNO[1] kannst du messen, wie viel Infrarotstrahlung den Solar-Destillierapparat erreicht. Bei einer mit Alufolie überzogenen Fläche kannst du mit diesem Sensor prüfen, ob die Strahlung in den Solar-Destillierapparat gelenkt wird.

Flammensensor

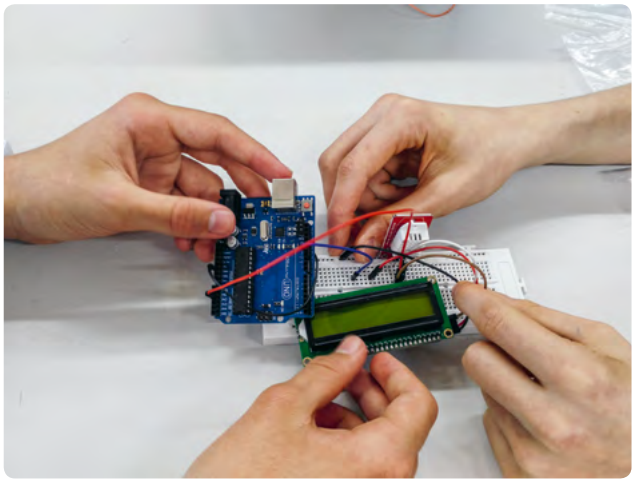
↳ Mit dem DHT11 oder DHT22 Feuchtigkeits- und Temperatursensor und Arduino UNO kannst du die Höchsttemperatur und relative Feuchtigkeit im Innern des Solar-Destillierapparats messen.



Arduino mit Feuchtigkeitssensor

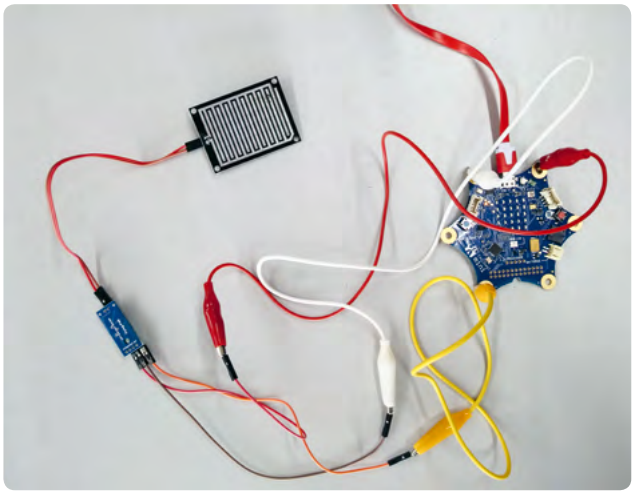


Arduino mit Temperatursensor



Arduino mit LCD-Display und Feuchtigkeitssensor

↳ Mit dem FR-04 Regensensor und Raspberry Pi[3] kannst du ein Python3-Programm schreiben, um die Zeit bis zur Kondensation des ersten Wassertropfens zu messen. Du kannst diesen Sensor auch mit einem Calliope mini[2] benutzen und mit Snap![10] (Blockprogrammiersprache) programmieren.



Regensensor mit einem Calliope mini

<Dritter Teil: Solar-Destillierapparat mit Sensoren testen>

Dein Team muss nun mit den programmierten Sensoren den eigenen Apparat testen und ihn mit einem anderen vergleichen. Dazu müssen die Solar-Destillierapparate unter den gleichen Bedingungen arbeiten und die gleichen Sensoren verwenden.

Analysiere, warum die Reinigungsgrade unterschiedlich sind. Dadurch ermittelst du die Schlüsselfaktoren, die beispielsweise den Verdunstungsprozess beeinflussen.

Reinigungsgrad des Solar-Destillierapparats erhöhen:

↳ Teste deinen Apparat mittags an einem sonnigen Tag.
↳ Gib deinem Apparat genügend Zeit, die maximale Temperatur zu erreichen.
↳ Stelle sicher, dass der Solar-Destillierapparat luftdicht ist, sodass kein Wasserdampf entweichen kann.
↳ Entferne das gereinigte Wasser fortlaufend, um dessen erneute Verdunstung zu vermeiden.
↳ Färbe das schmutzige Wasser ein, um sicherzustellen, dass der Solar-Destillierapparat korrekt funktioniert.
↳ Wähle für das schmutzige Wasser einen schwarzen, weiten Behälter.
↳ Verwende einen mit Alufolie überzogenen Schirm, um das Sonnenlicht in den Solar-Destillierapparat zu lenken.

<Erzielte Ergebnisse>

Der erste Teil der Unterrichtseinheit (Bau des Solar-Destillierapparats) führte zu Apparaten mit höchst unterschiedlichen Reinigungsgraden. Bei Tests an sonnigen Tagen im Mai/Juni in Spanien reinigte der beste Apparat 95 % des schmutzigen Wassers innerhalb von 24 Stunden. Es wurde auch ein Reinigungsgrad von 54 % innerhalb von 4 Stunden erzielt. Dennoch wurde bei einigen Apparaten aufgrund verschiedener Konstruktionsmängel überhaupt kein gereinigtes Wasser gesammelt.

Unter Verwendung der Sensoren wurden die folgenden Ergebnisse erzielt:

- Die höchste erreichte Temperatur im Innern der Solar-Destillierapparate nach einer Stunde an einem sonnigen Tag betrug 65 °C.
- Der Einfluss der Farbe des Schmutzwasserbehälters wurde analysiert. Beim Vergleich von zwei Behältern, die für einige Minuten ins Sonnenlicht gestellt wurden, war die Wassertemperatur im schwarzen Behälter fast 5 °C höher als im weißen.
- Mit dem Feuchtigkeitssensor wurde die Beziehung zwischen der Wassertemperatur und der Verdunstungsrate untersucht. Die relative Feuchtigkeit im Innern des Solar-Destillierapparats betrugt 55 % für Wasser bei Raumtemperatur. Wenn das Wasser auf 45 °C erhitzt wurde, erhöhte sie sich innerhalb weniger Sekunden auf 98 %.
- Die von einer metallischen Oberfläche in den Solar-Destillierapparat gelenkte Strahlungsmenge wurde mit dem Flammensensor gemessen. Ein mit Alufolie überzogener Schirm erwies sich für die Reflexion der Infrarotstrahlung als sehr effizient.

<Fazit>

Die Schülerinnen und Schüler arbeiten kreativ und analytisch, um den effizientesten Solar-Destillierapparat zu entwickeln. Diese Unterrichtseinheit gibt ihnen die Gelegenheit, kritisches Denken und Problemlösungskompetenzen zu entwickeln – Fähigkeiten, die sie in ihrem Alltag einsetzen können. Sie entdecken dabei wichtige physikalische Grundsätze (Selbststudium), und zwar nicht durch Lektüre ihrer Physikbücher, sondern weil sie beobachten, experimentieren, testen und ihre Ergebnisse analysieren.

Bei der Programmierung ihrer Sensoren entwickeln sie auch ihre Fähigkeit zum Computational Thinking. Zudem wenden sie Physical Computing an, d. h. ihre Programme wechselwirken mit der physischen Welt. Während der gesamten Einheit folgen sie den verschiedenen, zuvor beschriebenen Schritten wissenschaftlichen Arbeitens, um den besten Solar-Destillierapparat zu konzipieren und diesen mit geeigneten Materialien zu bauen.

Für das Projekt ist es wichtig, dass alle Schülerinnen und Schüler miteinbezogen werden. Damit dies sichergestellt ist, können einige Aufgaben des Projekts selbstständig ausgeführt werden (z. B. Recherche, erste Ideen einbringen).

Haupthindernisse könnten u. a. sein, dass manche Schülerinnen und Schüler nicht über ausreichende Programmierkenntnisse verfügen (weshalb wir Blockprogrammierung als Alternative einbezogen haben) oder nur mangelhafte Kenntnisse für den Bau von Schaltkreisen (die zusätzlichen Materialien[7] sind hier sehr hilfreich) oder über die Funktionsweise der Sensoren haben. Die Materialien sind vermutlich nicht in allen Schulen vorhanden und müssen erst angeschafft werden.

Projekterweiterungen:

- Die von den Sensoren gesammelten Daten werden zur weiteren Analyse auf einer SD-Karte gespeichert.
- Mit einer LCD-Anzeige werden die Messungen sichtbar gemacht.
- Internet of Things (IoT – Internet der Dinge): Die erhobenen Daten werden in Echtzeit ins Internet gestellt und so öffentlich zugänglich gemacht.
- Zusätzliche Sensoren werden eingesetzt: beispielsweise CO_2-Sensoren und andere Treibhausgas-Sensoren, ein Leitfähigkeitssensor zur Prüfung des Salzgehalts des gereinigten Wassers oder ein pH-Sensor zur Messung der pH-Werte des schmutzigen und des gereinigten Wassers.
- Bei der Verwendung von Meerwasser wird der Salzgehalt gemessen.
- Eine Methode zur Desinfektion des gesammelten Wassers wird erarbeitet.

Die Schülerinnen und Schüler können die Solar-Destillierapparate nutzen, um den Treibhauseffekt, die Fotosynthese, die Zellatmung, ideale Gase usw. zu untersuchen. Viele Sensoren zur Messung von physikalischen oder chemischen Parametern können für weitere Projekte verwendet werden, etwa zur Überwachung von Luftverschmutzung oder Wasserqualität.

<Kooperationsmöglichkeiten>

Sind Solar-Destillierapparate in Ländern mit hoher Sonneneinstrahlung effizienter? Schülerinnen und Schüler verschiedener Schulen in Europa können ihre Ergebnisse austauschen, indem sie einen Online-Kartendienst für Standortzwecke nutzen. Wenn Meerwasser verwendet wird, kann der Salzgehalt der verschiedenen Orte verglichen werden.

<Quellen und Hinweise>

[1] https://www.arduino.cc
[2] https://calliope.cc
[3] www.raspberrypi.org
[4] www.arduino.cc/reference/en
[5] www.python.org
[6] http://snap4arduino.rocks
[7] Alle Zusatzmaterialien sind verfügbar unter www.science-on-stage.de/coding-materialien.
[8] https://scratch.mit.edu
[9] www.raspberrypi.org/documentation/usage/python
[10] https://snap.berkeley.edu

Wasserwelten

<Autor> Lúcio Botelho

<Autorin> Liliana Fernandes

<Autor> Jorge Reis

<Info>

<Schlagwörter> Wasser, Bildverarbeitung, Datenerfassung, Mikroklima, Roboter

<Unterrichtsfächer> Mathematik, Biologie, Sozialkunde, Robotik, Kunst

<Altersgruppe> 6–10 Jahre, 11–15 Jahre und 16–18 Jahre

<Hardware> <Leichtes Niveau> Calliope mini[1], LEGO We Do 2.0[2], kleine Lernroboter[3], WeeeMake[4] <Mittleres Niveau> LEGO EV3[2] mit LEGO Ultraschall- und Farbsensoren oder Anprino[5] mit Arduino[6] und entsprechenden Ultraschall-[7] und Farbsensoren[8] <Fortgeschrittenes Niveau> Computer mit Internetzugang

<Programmiersprache> Snap![9], Scratch[10], WeeeCode[4], Open Roberta[11], LEGO Blocks[2]

<Programmierniveau> leicht, mittel, fortgeschritten

<Zusammenfassung>

Diese Einheit wurde fachübergreifend geplant und soll die Zusammenarbeit unter Schülerinnen und Schülern von der Primarstufe bis zur Sekundarstufe II erleichtern. Alternativ können die verschiedenen Teile unabhängig voneinander durchgeführt werden. Von Computational Thinking über Coding mit Scratch[10] bis hin zur Programmierung von Robotern und einem ökologischen Haus, deckt diese Einheit viele spannende Themen rund um Wasser ab.

<Vorstellung des Konzepts>

Bei diesem Projekt geht es um Wasser, seine Bedeutung für uns und unsere Verantwortung beim Wasserschutz. Aufgeteilt in drei Niveaus (leicht für die Primarstufe, mittel für die Sekundarstufe I und fortgeschritten für die Sekundarstufe II) können Schülerinnen und Schüler verschiedener Schulstufen in fachübergreifenden Aktivitäten zusammenarbeiten. Der ständige Austausch von Ideen und Ergebnissen motiviert alle gleichermaßen.

<Praktische Umsetzung>

<Leichtes Niveau: Wo kommt Wasser her?>

Die Schülerinnen und Schüler werden ermutigt herauszufinden, wo Wasser herkommt. Die Lehrkraft stellt einige einführende Fragen, um Neugier zu wecken und dann … beginnt das Abenteuer! Sie recherchieren, lernen und teilen ihre Erkenntnisse mit der Klasse und beginnen gleichzeitig, Computational Thinking mithilfe leichter Aufgaben zu entwickeln, indem sie einfache Lernroboter programmieren.

Nach der Recherche arbeiten die Schülerinnen und Schüler in kleinen Gruppen zusammen und erstellen unter Anwendung der Demo-Modi der WeDo 2.0 App[12] Anfängerprojekte mit Fokus auf wasserbezogene Themen.

Danach müssen die Schülerinnen und Schüler mit den Bausteinen oder Sets ihrer Wahl ein Modell bauen, das zeigt, wie sie auf innovative Weise Wasser sparen wollen.

In unserem Beispiel bauten sie ein Öko-Haus[13] und kombinierten es mit zusätzlichen Bausteinen und dem We Do 2.0-Set. Anschließend fügten sie einen Regensammler hinzu, der mit einem Filter (mit der LEGO-App programmiert) verbunden war und von dem aus das Wasser in den Bauernhof geleitet wurde, um die Tiere mit frischem Wasser zu versorgen (📷1).

📷 1: Öko-Haus

In kleinen Gruppen beginnen die Schülerinnen und Schüler nun mit der Planung und Konzeption neuer Matten für kleine Lernroboter, die ohne Computer programmiert werden können. Indem sie die Matten anschließend anderen Schülerinnen und Schülern vorstellen, regen sie diese an, selbst zu programmieren und zugleich etwas über Wasser zu lernen. Für diese Aufgabe eignen sich verschiedene preisgünstige Lernroboter.[3]

Alle Anleitungen zum Ausdrucken der Matte sind online verfügbar.[14]

📷 2: Zeichnen neuer Matten

Im Zusatzmaterial finden Sie eine Schritt-für-Schritt-Anleitung für diese Einheit.[14]

<Mittleres Niveau: Bau eines Reinigungsroboters für Stauseen>

In diesem Teil geht es um einen Roboter, der sich auf einem Stausee bewegt und dabei feste Abfallstoffe aufspürt.

Das Projekt gibt es in zwei Versionen mit verschiedenen Robotern: Bei der LEGO-Version (📷 3) kommt das EV3 LEGO Education-Kit zum Einsatz, während die Arduino-Version auf dem Anprino-Roboter[5] basiert (📷 4), der mit einem 3D-Drucker gedruckt und dann zusammengebaut wird. Der Arduino[6] und sein Zubehör werden am Anprino befestigt.

📷 3: LEGO-Version

📷 4: Anprino-Version

Zuerst wird der „Stausee" aus blauem Papier oder Pappe gebaut, wobei das Modell etwa 2 m × 1 m misst. Das Ufer wird aus fester Pappe gefertigt, wodurch der Aktionsbereich des Roboters begrenzt ist. Der Müll wird mit schwarzer Pappe simuliert.

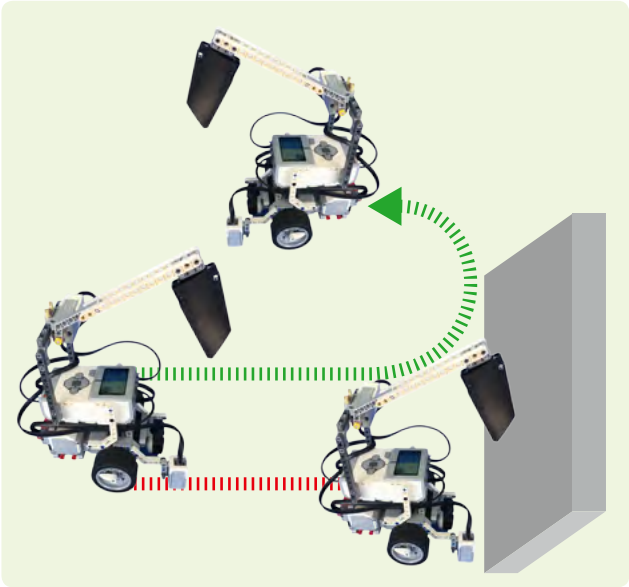
📷 5: Stauseemodell

Ultraschallsensor

Der Ultraschallsensor[7] besteht aus einem Sender und einem Empfänger für Ultraschallwellen. Mit ihm können Objekte aufgespürt und die Entfernungen zu ihnen gemessen werden.

Mit dem Ultraschallsensor kann der Roboter Hindernisse erkennen und je nach Programmierung unterschiedlich reagieren, z. B. anhalten oder seine Richtung ändern. Beim Stauseemodell fungieren die Ufer aus Pappe als Hindernisse.

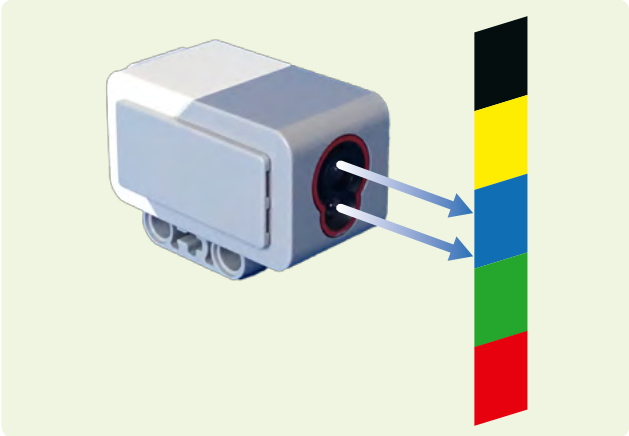
📷 6: LEGO-Ultraschallsensor



📷 7: Roboter hält an/ändert die Richtung

Farbsensor

Der Farbsensor[8] kann verschiedene Farben sowie die Abwesenheit von Licht erkennen. Er funktioniert auch als Lichtsensor, da er unterschiedliche Lichtintensitäten detektieren kann. Die Schülerinnen und Schüler können verschiedenfarbige Linien erstellen, denen der Roboter folgen soll.



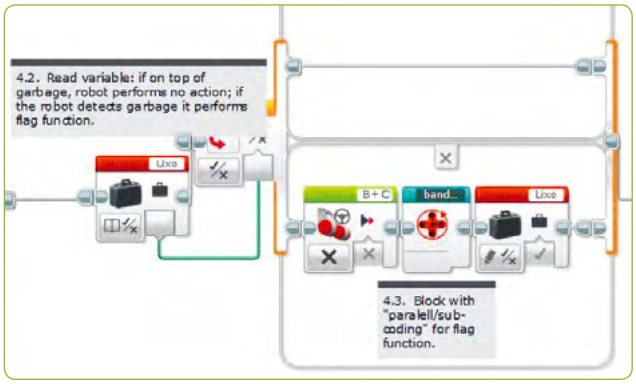
📷 8: Beispiel der Farberkennung: Der Sensor unterscheidet Farben durch das Lesen ihrer RGB-Werte.

Erstellung eines Codes mit LEGO-Blockprogrammierung

Die Schülerinnen und Schüler bauen verschiedene Modelle zur Simulation von klar unterscheidbaren Abfällen aus verschiedenen Bereichen, wie z. B. Haushalt, Industrie, Tourismus oder Biomüll.

Das Hauptziel ist die Sensibilisierung für die Verschmutzung von Flüssen und Stauseen. Die Schülerinnen und Schüler simulieren einen auf einem Boot montierten Abfalldetektor und planen und bauen später ein Müllsammelboot.

Der Roboter spielt für jede erkannte Abfallart einen bestimmten Ton ab. Um dies zu erreichen, benutzt man den Farbsensor und spezifische farbige „Flecken" für die einzelnen Abfallarten.



9: Auszug aus LEGO-Programmierung; das vollständige Diagramm ist online verfügbar[14]

Die Schülerinnen und Schüler können öffentlich verfügbare Statistiken analysieren und anhand dieser die verschiedenen Flecken erstellen.

Sie können auf Exkursionen die Wasserqualität und den Verschmutzungsgrad von Flüssen und Stauseen untersuchen. In ihren Modellen müssen sie diese Beobachtungen einbeziehen und simulieren. Mithilfe des Roboters können sie die Ergebnisse scannen und in einer Tabelle festhalten (10).

Wenn ausreichend Ergebnisse gesammelt wurden, werden sie der Klasse präsentiert. Ziel ist es, kritisches Denken und Programmierkenntnisse zu entwickeln sowie forschend-entdeckendes Lernen zu fördern. Die Untersuchung von Gewässern macht den Schülerinnen und Schülern die Folgen jahrhundertelanger ökologischer Unkenntnis und Rücksichtslosigkeit deutlich. Hierfür ist es wichtig, dass sie die erforderlichen Kompetenzen im Umweltschutz erwerben, damit sie auch in ihrer Gemeinde intervenieren können. Sie könnten beispielsweise auf die Notwendigkeit hinweisen, sorgsamer mit der Natur und insbesondere mit der Ressource Wasser umzugehen. Außerdem werden die Schülerinnen und Schüler befähigt, problemlösungsorientiert zu arbeiten. Insgesamt sollen ihre aktive bürgerliche Teilhabe und ihr Umweltbewusstsein gestärkt werden.

Hinweis: Die LEGO-Version wurde schon gebaut und getestet. Die Arduino Version wird noch verbessert. Der vollständige Programmiercode von Arduino ist online verfügbar.[14]

<Fortgeschrittenes Niveau: Programmieren von Lernspielen>

Zur Sensibilisierung für Wasserverschmutzung werden in Scratch[10] Spiele programmiert, die andere dazu anregen sollen, Wasser zu sparen und zu schützen, u. a. indem kein Müll in Gewässern entsorgt wird.

Das erste Spiel simuliert einen kleinen Fisch im Meer, welcher Nahrung sucht und frisst und dabei Meerestieren (Haien und Krabben) und sinkendem Müll (Gläser, Dosen usw.) ausweicht. Je mehr der Fisch frisst, desto größer wird er und desto mehr Punkte erhält man.

Wenn der Fisch mit Müll oder anderen Fischen kollidiert, wird er verletzt und bekommt einen Verband. Sobald er drei Verbände hat, ist das Spiel zu Ende. Das Spiel macht Spaß und weist nicht nur Kinder auf die steigenden Abfallmengen in unseren Gewässern hin.

Das zweite Spiel basiert auf einem bekannten Videospiel, bei dem ein Frosch eine Straße überqueren muss. In unserem Fall muss die Figur einen Fluss überqueren (mithilfe von Baum-

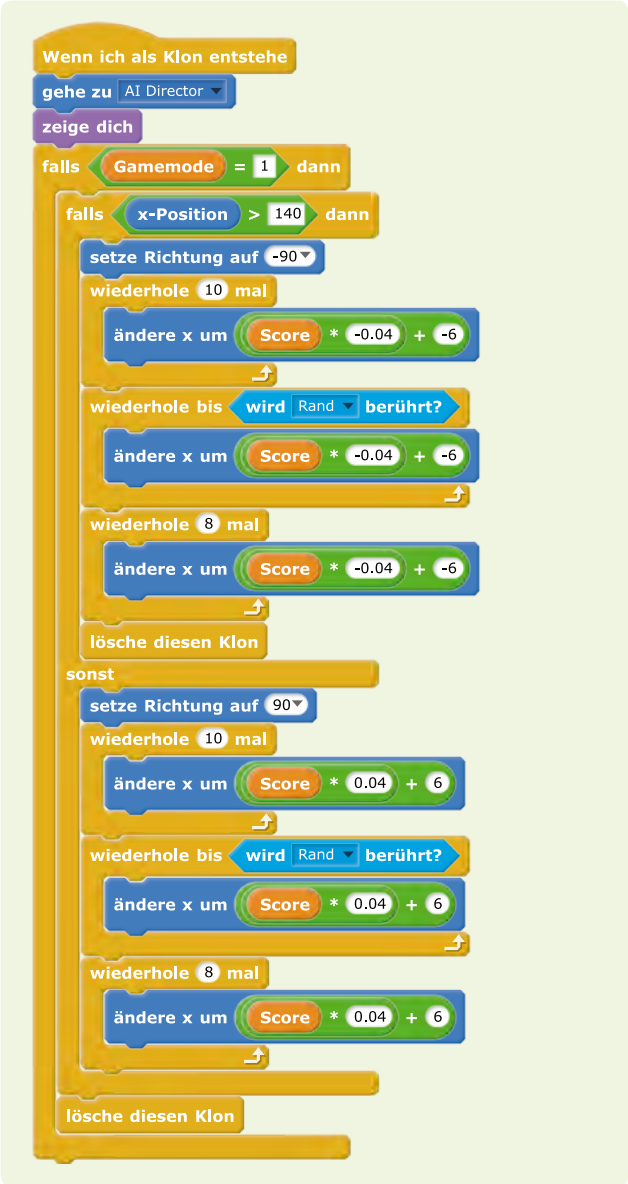
10: Tabelle für Abfall-Scan: Daten aus zwei verschiedenen Exkursionen zu dem bei jeder Exkursion von den Reinigungsteams der Umwelt-AG gesammelten Müll

Datum	Abfallart					Gereinigte Fläche
	Haushalt	Industrie	Unbestimmt	Organisch	Andere	
Exkursion April 2018	3,450 kg			32 kg	8 kg	100 m²
Exkursion Mai 2018	0,730 kg			6 kg		100 m²

stämmen, da das Wasser rasch fließt) und gleichzeitig Müll und andere Tiere (Fledermäuse und Schlangen) meiden. Die Spielfigur kann auch für Extrapunkte Fliegen fressen. In diesem Spiel gibt es vier Szenarien, wovon jeweils eines zufällig zu Beginn jeder Runde ausgewählt wird. Der Frosch (Figur) hat drei Leben („Lives") und wenn er alle verloren hat, endet das Spiel.

Im Folgenden wird das Programm detaillierter dargestellt.

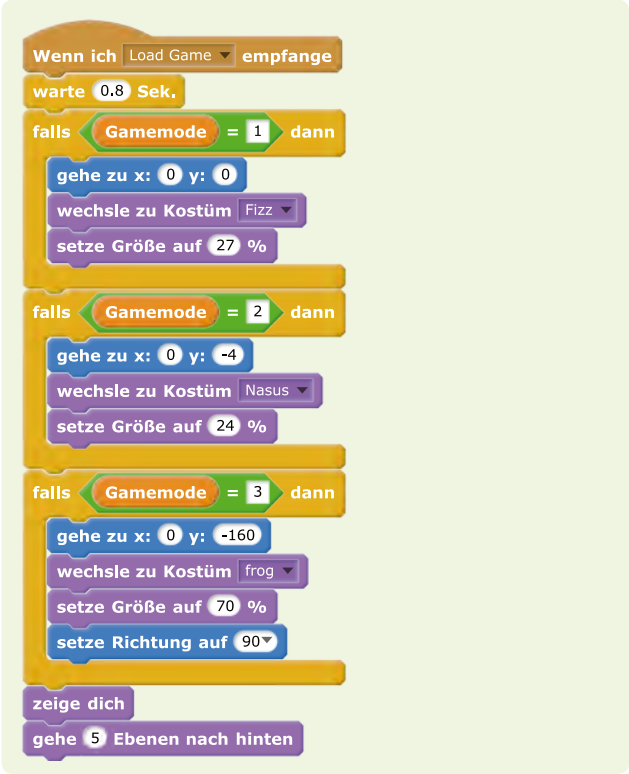
📷 11 zeigt den Teil des Programms zur Bewegungssteuerung einiger Gegner („Enemy") in den verschiedenen Spielmodi. Im gezeigten Beispiel verschwindet der Gegner bei Berührung des Bildrands. Solange er den Rand nicht berührt, wiederholt er die Bewegung, die sich zudem bei zunehmendem Punktestand („Score") mit einem Anpassungsfaktor von 0,04 beschleunigt. Dies ist eine sehr clevere Art, das Spiel etwas anspruchsvoller zu gestalten, da sich der Punktestand bei zunehmendem Schwierigkeitsgrad erhöht.



📷 11: Scratch-Programm: Steuerung der Gegner

Zum Spielbeginn wählt man einen der drei Spielmodi („Game mode") aus (📷 12). Es gibt zwei fertig programmierte Spiele und unsere Schülerinnen und Schüler entwickeln derzeit ein drittes Spiel, Spielmodus 2.

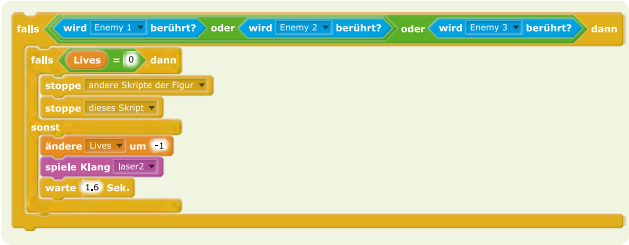
Spielmodus 2 könnte sich beispielsweise in einem Teich abspielen, in dem Enten Nahrung finden müssen: Enten fressen oft kleine Fische und Fischlaich, Schnecken, Würmer und Weichtiere, kleine Krustentiere, aber auch Gras, Blätter, Seegras, Algen, Wasserpflanzen, Wurzeln, kleine Frösche, Salamander und andere Amphibien. Zudem müssen die Enten versuchen, andere Enten und den Müll im Teich zu meiden (und in höheren Levels einige zufällig auftauchende Wilderer).



📷 12: Scratch-Programm: Spielbeginn

Wenn der Fisch einen der Gegner berührt (1, 2 oder 3), verliert er ein Leben und es erklingt ein Ton.

Bei Verlust aller Leben endet das Spiel, d. h. alle Skripte werden gestoppt (📷 13).



📷 13: Scratch-Programm: Gegner 1-3

Das Spiel ist sehr gut programmiert und aufgebaut, weil derselbe Code für beide Spielversionen verwendet wird. Dieselbe Figur ändert ihr Kostüm von „Hai“ zu „Fledermaus“.

Die Schülerinnen und Schüler lernen, das Programm zu optimieren, bringen neue Ideen ein und suchen nach innovativen Lösungen, die den Code noch besser und flüssiger machen.

Dies ist möglich, weil die Spiele ähnliche Ziele verfolgen:
↳ Gegner meiden
↳ Nahrung finden/Fliegen fangen
↳ Spielende bei Verlust von drei Leben
↳ Punkte sammeln (indem der Fisch Fischfutter frisst, der Frosch Fliegen fängt oder ein neues Szenario erreicht wird)

Die Schülerinnen und Schüler benutzen Klone der Gegner-Figuren, sodass sie die gleiche Figur aus verschiedenen Richtungen auftauchen lassen und ihr verschiedene Verhaltensweisen (Richtungen) im Spiel verleihen können.

Das vollständige Programm kann heruntergeladen werden.[14]

<Fazit>

In dieser Einheit arbeiten die Schülerinnen und Schüler miteinander und mit ihrer Gemeinde vor Ort. Sie erwerben und teilen Wissen über Wasser – Kreislauf, Mangel, Verschmutzung usw. – und erarbeiten Methoden, um die Wasserqualität zu kontrollieren, Wasser zu sparen und zu schützen. Gleichzeitig werden Untersuchungswerkzeuge und Programmierkenntnisse sowie Kompetenzen in der Robotik entwickelt. Indem die Älteren die Jüngeren unterstützen, motivieren sie sich gegenseitig und entwickeln den Ehrgeiz, ihre Arbeit weiter voranzubringen.

Am Ende des Schuljahres hatten die Schülerinnen und Schüler nicht nur ihre Programmierkenntnisse vertieft, sondern waren auch viel sensibler gegenüber Wasserproblemen und deren Gefahren für Tiere und Pflanzen, die in ihrem Lebensraum von sauberem Wasser abhängig sind.

Das Programmieren mehrerer Spiele ist eine Herausforderung. Voraussetzung ist, dass sie sich ähnlich genug sind, damit der Code eines Spiels auf ein anderes übertragen werden kann, um alle Modi abzudecken. So lassen sich auf kluge Art auch Programmierressourcen sparen.

Wir haben zusammengearbeitet, obwohl wir an verschiedenen (weit voneinander entfernten) Schulen tätig sind, weil es dadurch möglich war, Ideen auszutauschen und die Zusammenarbeit unter Jugendlichen aus unterschiedlichen Umfeldern (sozioökonomisch) und Altersgruppen zu fördern. Persönliche Treffen waren nicht einfach und die Klassen konnten sich seltener als geplant sehen. Die Kooperation hat sich trotzdem gelohnt, da die Schülerinnen und Schüler ihre Ideen und Vorgehensweisen austauschen und mit Jugendlichen aus anderen Schulen sprechen konnten, was ihre Kommunikationsfähigkeiten trainierte. Außerdem war es so möglich, an verschiedenen Wettbewerben teilzunehmen, Ergebnisse zu besprechen und Optimierungsvorschläge an die anderen weiterzugeben. Eine Alternative zu persönlichen Treffen könnten Videokonferenzen sein. Zu guter Letzt haben die Schülerinnen und Schüler ihre Arbeiten mit der Gemeinde geteilt und damit einen Beitrag für den lokalen Gewässerschutz geleistet.

Ausgehend von dieser Einheit können Sie Ihre Klasse dazu anregen, weitere Ideen und Konzepte zum Wassersparen zu entwickeln, um umweltbewusstes Verhalten in der Gemeinde zu stärken, und somit den CO_2-Fußabdruck der Schülerinnen und Schüler, und hoffentlich auch der Gemeinde, zu reduzieren.

<Kooperationsmöglichkeiten>

Bei Science on Stage geht es um den Austausch von Konzepten unter den Lehrkräften!

Durch dieses Projekt wurde eine Gemeinschaft von Lehrkräften gestärkt, Mittel und Ideen wurden geteilt. Dies stellt einen Beitrag zum besseren Lernen von Schülerinnen und Schülern in ganz Europa dar. Austausch und Zusammenarbeit sind der beste Weg für einen guten Unterricht und eine Weiterentwicklung dieser Projekte.

<Quellen und Hinweise>

[1] https://calliope.cc
[2] https://education.lego.com
[3] Mögliche Bots: Bee Bots von tts, DOC von Clementoni, Jack von Imaginarium
[4] www.weeemake.com
[5] Anprino ist ein Roboter, der von ANPRI, dem portugiesischen Verband der Informatiklehrkräfte, entwickelt wurde: www.anpri.pt/anprino/index.php/anprino-luis
[6] www.arduino.cc
[7] Wir benutzten den Ultraschallsensor HC-SR04.
[8] Wir benutzten den Lichtsensor BE15000624.
[9] https://snap.berkeley.edu
[10] https://scratch.mit.edu
[11] https://lab.open-roberta.org
[12] https://education.lego.com/de-de/downloads/wedo-2/software (29/11/2018)
[13] LEGO SET 31068
[14] Sämtliches Zusatzmaterial ist erhältlich auf www.science-on-stage.de/coding-materialien.

VPLS – Vacation Plant Life Saver

<Autor> Andreas Meier
<Autorin> Sonja van der Byl

<Info>

<Schlagwörter> automatische Pflanzenbewässerung, Verwendung eines Mikrocontrollers zum Ansteuern einer Wasserpumpe

<Unterrichtsfächer> Informatik, Naturwissenschaften, Technik

<Altersgruppe> 10–14 Jahre

<Hardware> Computer (möglichst einer pro Schülerin bzw. Schüler), Calliope mini[1] mit Feuchtigkeits- und Temperatursensor (einer pro Kleingruppe)

<Programmiersprache> Scratch[2] (online oder offline), Editor für Calliope mini[3] (online)

<Programmierniveau> leicht

<Zusammenfassung>

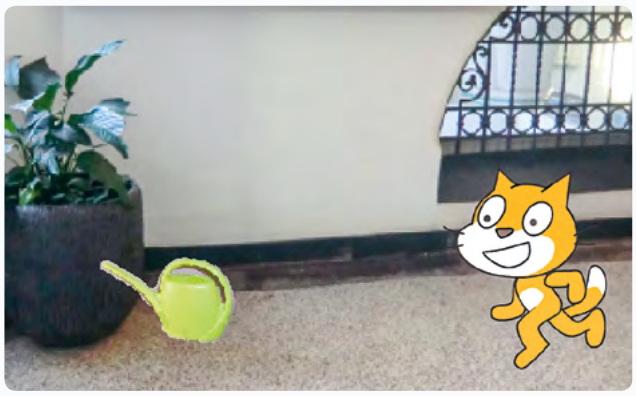
Die Kübelpflanzen im Schulgebäude vertrocknen regelmäßig in den Ferien, weil niemand sie gießt – wir brauchen also einen Vacation Plant Life Saver! Im Rahmen dieser Reihe bauen wir den Lebensretter für unsere Schulpflanzen virtuell und real.

<Vorstellung des Konzepts>

Das Projekt ist für alle naturwissenschaftlichen Bereiche geeignet, da es sich an Programmieranfängerinnen und -anfänger richtet. Das regelmäßige Gießen einer Pflanze wird dabei im ersten Teil des Projekts virtuell umgesetzt und führt aus Sicht der Informatik in Objektorientierung, Kontrollstrukturen (Wiederholungen, Bedingungen) und das Nutzen von Variablen ein. Dabei lässt es viel Raum für eigene Programmiererfahrungen der Schülerinnen und Schüler. Als Vorbereitung arbeiten sie sich online in Scratch ein („Erste Schritte")[4]. (Zeitbedarf ca. 45 Minuten)

Bei der Objektorientierten Programmierung (OOP) wird alles als Objekt gesehen. Diese besitzen Eigenschaften (Attribute) und Fähigkeiten (Methoden). Hier sind das die Katze „Sprite", die Gießkanne und die Bühne.

Jedes Objekt gehört zu einer Klasse. Alle Objekte einer Klasse haben die gleichen Eigenschaften und Fähigkeiten. Man kann eine Klasse als Blaupause (blueprint) und ein Objekt als Instanz (instance), also konkrete Realisierung, auffassen. In Scratch sind Figuren Instanzen der Klasse „Sprite". Man spricht deshalb auch von Sprites. Die Katze mit dem Namen „Sprite" ist ein Sprite, also eine Instanz der Klasse „Sprite". Eine Eigenschaft von Sprites ist deren Kostüm, in diesem Fall z. B. das Bild einer Katze. Eine andere Instanz der Klasse „Sprite" könnte als Kostüm das Bild eines Menschen haben und „Günther" heißen. Die Katze „Sprite" und der Mensch „Günther" sind also beide Objekte bzw. Instanzen der Klasse „Sprite" oder kurz: beide sind Sprites. In Scratch gibt es nur zwei Klassen: die Bühne (stage) und Figuren (sprites). In diesem Projekt werden die zwei Figuren (sprites) Katze und Gießkanne sowie die Bühne benutzt. (📷 1)



📷 1: Der virtuelle Vacation Plant Life Saver

Der zweite Teil des Projekts kann auf dem ersten Teil aufbauen, ist aber auch unabhängig davon durchführbar, wenn die Schülerinnen und Schüler die oben genannten Kontrollstrukturen kennen und bereits erste Programmiererfahrungen mit dem Calliope mini[1] gesammelt haben. Statt der Variablen kommen hier Sensoren eines Mikrocontrollers zum Einsatz, die das Ventil einer Wasserpumpe ansteuern.

<Praktische Umsetzung>

Alle benötigten Materialien wie Arbeitsblätter und (zerlegte) Programme stehen digital zur Verfügung.[5]

<Teil 1: Virtuelles regelmäßiges Gießen einer Pflanze>

Arbeitsschritt 1: Erstellen eines Programms, das nur die Zeit als Variable berücksichtigt; Kennenlernen von einseitigen Bedingungen und Wiederholungen (Zeitbedarf: 180 Minuten)

Nach dem Einstieg in das Problem (Wie könnte ein virtueller Vacation Plant Life Saver funktionieren?) sollten sich die Schülerinnen und Schüler zunächst Gedanken über die Grundstruktur eines Gießprogramms machen, diese in Form eines „Kochrezepts" (Algorithmus) notieren und ihre Ideen gegenseitig testen. Dann erst kann eine gemeinsame Grundstruktur für ein Gießprogramm festgelegt werden (siehe Arbeitsblatt 1[5]). Diese Phase dient dazu, sich mit der grundlegenden Struktur des Programms, das entstehen soll, auseinanderzusetzen. Die Begriffe „Anweisungsfolge", „Wiederholung" und „Bedingung" ergeben sich dabei aus dem Sachzusammenhang.

Erst jetzt wird der Algorithmus in die Programmiersprache Scratch[2] übersetzt, indem die Schülerinnen und Schüler zu-

nächst ein in seine Einzelteile zerlegtes Programm[5] zu einem funktionierenden Programm zusammensetzen.

So lernen sie die Funktionsweise von Scratch intensiver kennen:
- ↳ Objektorientierung (jede Figur hat ihr eigenes Skript, auch die Bühne)
- ↳ Strukturierung (Wie sieht die Struktur einer „Bedingung“ und einer „Wiederholung“ in Scratch aus?)
- ↳ Skript/Kostüm/Klänge können jeder einzelnen Figur zugewiesen werden

Zusätzlich wird die gemeinsam erarbeitete Grundstruktur des Gießprogramms vertieft, indem die Schülerinnen und Schüler darüber nachdenken, in welcher Reihenfolge sie die vorgegebenen Puzzleteile anordnen müssen, damit das Programm funktioniert. Insbesondere die Frage, welche Anweisungen sich innerhalb der Zählschleife (📷 2 & 3) befinden müssen, kann durch Ausprobieren beantwortet werden.

📷 2

📷 3

Auf Grundlage des zusammengesetzten Programms dürfen die Schülerinnen und Schüler nun ein eigenes Programm erstellen, das in Abhängigkeit von der Zeit die Katze „Sprite“ losschickt, um die Pflanze auf der Bühne zu gießen. Dazu bekommen sie lediglich das Ausgangsbühnenbild als Datei zur Verfügung gestellt.[5]

Dieser Schritt dient dazu, die Schülerinnen und Schüler zu ermutigen, die Programmiersprache eigenständig zu erforschen, Ideen auszuprobieren und kreativ zu arbeiten. Das ist wichtig, weil Anfängerinnen und Anfänger nur dann mit Spaß programmieren lernen, wenn sie das Gefühl haben, sich in der entsprechenden Programmiersprache (dem eigenen Wissensstand entsprechend) gut auszukennen.

Arbeitsschritt 2: Schrittweises Erstellen eines Programms, das die Variablen „Wasserstand“ und „Temperatur“ berücksichtigt (Zeitbedarf: 270 Minuten)

Als Einstieg in den zweiten Arbeitsschritt denken die Schülerinnen und Schüler über weitere Faktoren nach, die festlegen, wie oft eine Kübelpflanze gegossen werden muss. Die „Raumtemperatur“ und der „Wasserstand“ im Kübel spielen hier sicherlich eine Rolle als sich verändernde, also variable Größen. (Arbeitsblatt 2[5])

Die Schülerinnen und Schüler erhalten im Anschluss ein funktionierendes Programm, bei dem sich die Skripte für die Katze „Sprite“ und die Gießkanne kaum verändert haben.

Allerdings steuert das Skript für die Bühne jetzt die neu definierte Variable „Wasserstand“, die den Timer ersetzt. Auch gießt „Sprite“ die Pflanze nur einmal. Die Schülerinnen und Schüler sind also aufgefordert, sich einerseits Gedanken darüber zu machen, wie die Variable „Wasserstand“ erzeugt wurde und wie sie die Aktivität der Katze steuert. Andererseits sollen sie versuchen, das Problem zu lösen, dass nur einmal gegossen wird. Falls nötig steht ihnen hierfür eine Hilfsdatei zur Verfügung. [5]

Durch die vorläufige Nutzung von nur einer Variablen bleibt das Programm übersichtlich. Die Organisation mehrerer Variablen würde sicher zu Anfang überfordern.

Die hier einzufügende Wiederholungsstruktur ist komplexer als zuvor, da mit ihr eine Bedingung verknüpft ist (📷 4). Die Schülerinnen und Schüler müssen sich genau überlegen, was unter welcher Bedingung wie oft zu wiederholen ist.

📷 4

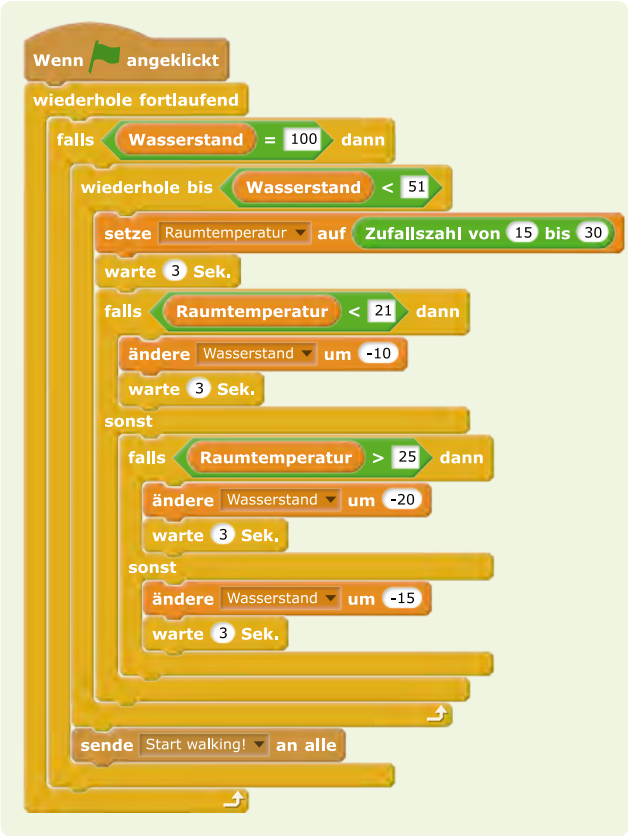
Die Fähigkeit zu strukturieren ist beim Erlernen jeder Programmiersprache von grundlegender Wichtigkeit. Hier wird sie auf

eine sehr spielerische Art und Weise vermittelt, denn die Schülerinnen und Schüler können einfach ausprobieren, was passiert, wenn sie eine falsche Wiederholungsstruktur nutzen bzw. nicht an die notwendige Bedingung zum Abbruch der Wiederholung denken.

Für Schülerinnen und Schüler mit rascher Auffassungsgabe besteht am Ende dieser Arbeitsphase die Möglichkeit, das Programm abzuändern und eigene Ideen auszuprobieren.

Erst in diesem sich anschließenden Schritt spielt auch die Variable „Temperatur" im Programm eine Rolle. Ihr Wert wird per Zufallszahlengenerator zwischen 15 °C und 30 °C festgelegt. Der Wasserstand im Kübel ändert sich dann in Abhängigkeit von diesem Wert. (Arbeitsblatt 3[5])

Da die Programmstruktur des „neuen" Gießprogramms somit sehr komplex ist, soll zunächst wieder ein zerlegtes Programm zu einem funktionierenden Programm zusammengesetzt werden. Dabei werden der Zufallszahlengenerator und die zweiseitige Bedingung eingeführt, ebenso wie das Arbeiten mit Variablen und das Abfragen von Bedingungen wiederholt und vertieft. Erneut müssen sich die Schülerinnen und Schüler genau überlegen, was unter welcher Bedingung wie oft wiederholt werden sollte. (📷 5)



📷 5

Auch hier gibt es im Anschluss die Möglichkeit, das entstandene Programm zu individualisieren, wobei die Schülerinnen und Schüler entsprechend ihrer Ideen und erworbenen Programmierfähigkeiten arbeiten können.

Unbedingt sollten sie am Ende dieses ersten Projektteils, bei dem sie den Vacation Plant Life Saver virtuell umgesetzt haben, möglichst viele ihrer Programmierergebnisse präsentieren. Damit werden ihre Ideen gewürdigt und ihre Leistung anerkannt.

<Teil 2: Regelmäßiges Gießen einer Pflanze gesteuert durch einen Mikrocontroller>

Im Laufe des Projekts wurde schnell deutlich, dass sich einige Schülerinnen und Schüler nicht mit der virtuellen Lösung des Gießproblems zufriedengeben. Sie fragen nach Möglichkeiten, reale Pflanzen mithilfe ihres Programms wirklich zu gießen. Mit einem Mikrocontroller lässt sich das virtuelle Gießprogramm relativ leicht zur Steuerung eines realen VPLS verwenden, insbesondere da viele dieser Minicomputer ebenfalls mit Scratch oder einer vergleichbaren App programmierbar sind.

In unserem Projekt kommt der Calliope mini[1] zum Einsatz, eine Weiterentwicklung des BBC micro:bit[6], der zusätzlich über leicht nutzbare „Plug-and-play"-Funktionen wie Berührungssensoren und Motorenanschlüsse verfügt. Aber auch mit allen anderen in der Schule üblicherweise verwendeten Mikrocontrollern wie LEGO EV3, LEGO NXT, Arduino, Raspberry Pi, Teensy, … kann die Steuerung des VPLS erfolgen[5].

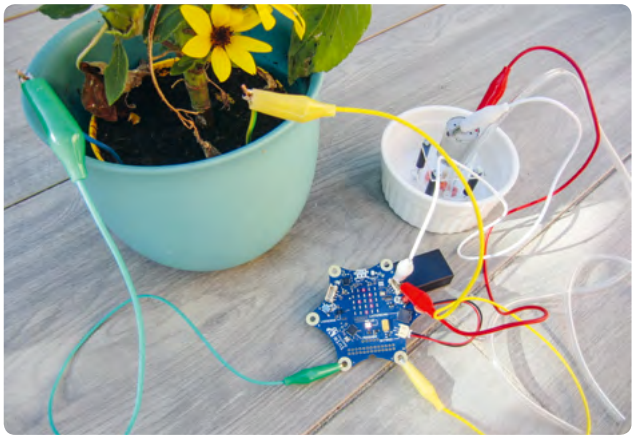
Um den Calliope mini zu programmieren, wird er einfach per USB-Kabel mit dem Computer verbunden. Als Programmieroberfläche eignet sich Open Roberta Lab[7], welche verschiedene Mikrocontroller unterstützt (alternative Editoren sind im Zusatzmaterial aufgeführt[5]). Die Programmieroberfläche ist in verschiedenen Sprachen verfügbar. Diese lassen sich schnell mit einem Klick auf das Erdkugel-Symbol ändern, nachdem man den Mikrocontroller ausgewählt hat, in unserem Fall den Calliope mini.

Eine einfache Version des VPLS könnte folgendermaßen funktionieren:

1. Die Feuchtigkeit der Erde wird ständig gemessen.
2. Ist die Erde zu trocken, wird eine bestimmte Menge Wasser über eine Pumpe zugeführt, bis der Boden feucht genug ist.

Der Mikrocontroller muss also die Feuchtigkeit des Bodens messen und den Motor einer Wasserpumpe betreiben können.

Der Calliope mini verfügt über vier Berührungssensoren, die aus physikalischer Sicht lediglich die elektrische Leitfähigkeit zwischen den Anschlusspunkten messen. Da Wasser elektrischen Strom leitet, besitzt feuchter Boden eine höhere Leitfähigkeit als trockener Boden. Als Sensoren verwendet man zwei Kupferdrähte, die man mit etwas Abstand in den Blumentopf steckt. Diese werden durch Krokodil-Kabel mit dem Calliope mini an den Eckkontakten (-, P1) verbunden. Der Sensor „analoger Pin" P1 liefert nun einen Wert zwischen 0 und 1023. Sinkt die Leitfähigkeit unter einen bestimmten Wert, dann wird die Pumpe aktiviert, die die Pflanze mit Wasser versorgt. (📷 6)



📷 6: Calliope mini steuert den VPLS

Wesentlicher Bestandteil der Pumpe ist ein kleiner Elektromotor, der je nach benötigter Leistung direkt oder mithilfe eines zusätzlichen Motortreibers an den Calliope mini angeschlossen wird. Zwei Motorausgänge (A, B) stehen unter dem Menü Aktion/Bewegung zur Verfügung. Mit dem Wert „Tempo %" lässt sich die Wassermenge der Pumpe einstellen.

Sowohl beim Bau der Pumpe als auch beim Aufbau des Motortreibers ist etwas handwerkliches Geschick gefragt, im Online-Material [5] stellen wir entsprechende Bauanleitungen zur Verfügung. Die Kosten für die Pumpe belaufen sich auf wenige Euro.

Die Arbeit mit dem realen VPLS bietet den Schülerinnen und Schülern zahlreiche Fragestellungen zur Optimierung und Vertiefung, beispielsweise:

- ↳ Benötigen unsere Pflanzen viel oder wenig Wasser? Wie groß muss der gesamte Wasservorrat sein?
- ↳ Welcher ist der beste Gießzeitpunkt? Ist es besser nur einmal am Tag viel zu gießen oder mehrmals wenig?
- ↳ In welcher Bodentiefe sollten die Feuchtigkeitssensoren am besten messen? Welches ist der beste Abstand für die Sensoren?
- ↳ Wie lange reicht die Energieversorgung des VPLS? Lässt sich die Energieeffizienz der Pumpe steigern, sodass der VPLS während der gesamten Ferien gießt?

Den Schülerinnen und Schülern bieten sich verschiedenste Ansätze aus den Bereichen Biologie und Physik zur Weiterarbeit am VPLS-Projekt an. Interessant wäre sicher auch eine Anbindung und Überwachung über das Internet. Dies ginge weit über die von uns mit dem VPLS angestrebte Einführung ins Programmieren hinaus, zeigt aber die Möglichkeiten, die sich aus einer einfachen Fragestellung ergeben können.

<Übertragbarkeit in andere Programmiersprachen>

Sehr leicht lässt sich das Projekt in Snap![8], eine Weiterentwicklung von Scratch[2], übertragen. Programmbeispiele sind online erhältlich[5]. Diese beiden Programmiersprachen eignen sich besonders gut für ein Projekt, das sich an Programmieranfängerinnen und -anfänger richtet, weil sie anschaulich sind und dazu ermutigen, Ideen per „drag and drop" einfach auszuprobieren.

<Fazit>

Programmieranfängerinnen und -anfänger machen im Laufe dieses alltagsnahen Projekts ihre ersten Programmiererfahrungen und lernen wichtige Grundlagen einer Programmiersprache kennen. Dabei liegt der Fokus nicht darauf, deren Syntax und das Vokabular zu lernen, sondern die Wirkung bestimmter Strukturen auszuprobieren: „Was funktioniert wie und warum?". Fehler sind hier durchaus erwünscht, denn sie lassen sich in der Regel leicht finden und erklären und tragen somit dazu bei, zu verstehen, wie eine Programmiersprache funktioniert.

Zudem gibt der vorgegebene Rahmen den Schülerinnen und Schülern einerseits Sicherheit (wer unsicher ist, löst eben nur das Puzzle), andererseits gibt es für diejenigen, die die „Pflichtaufgaben" schnell erledigen viel Raum für Kreativität.

Aus dem Projekt heraus sind einige Ideen entstanden, z. B. einen „echten" Gießroboter zu bauen oder ein Gießspiel zu entwickeln. In jedem Fall haben die Schülerinnen und Schüler positive erste Programmiererfahrungen gemacht, die hoffentlich lange nachwirken.

Schwierig war mitunter der an unserer Schule vorgegebene Zeitrahmen von 45 Minuten. Manchmal hat allein der rein organisatorische Teil (Anmelden am Rechner, Dateien öffnen, Dateien speichern, Abmelden vom Rechner) 15 Minuten gekostet, sodass die echte Programmier- und Arbeitszeit zu kurz kam. Man kann als Lehrkraft für ein Projekt einen Lehrerzugang bei Scratch[2] anfordern, unter dem man eine Klasse einrichten und Materialien hinterlegen kann, was sehr hilfreich ist.

<Kooperationsmöglichkeiten>

Da es sich beim VPLS um eine Einführung ins Programmieren handelt, bieten sich zunächst kaum Möglichkeiten einer Kooperation an. Sobald das Projekt auf die reale Steuerung eines Mikrocontrollers übertragen wird, sind Kooperationen unter Schülerinnen und Schülern hiflreich, da sich der Schwierigkeitsgrad des Problems durch den Einsatz von zusätzlichen Komponenten (Sensoren, Pumpe) erhöht. Möglich wäre etwa, dass ältere Schülerinnen und Schüler beim Bau der Pumpe helfen. Auch eine internationale Kooperation zwischen Schulen, die am VPLS-Projekt arbeiten, ist denkbar.

Es wäre möglich, gemeinsam eine Datenbank für unterschiedliche Pflanzen zu erstellen, sodass sich der VPLS leicht auf die individuellen Eigenschaften verschiedener Pflanzenarten anpassen lässt. Gelingt die Anbindung des VPLS über das Internet könnten sogar internationale Gieß-Patenschaften entstehen!

<Quellen und Hinweise>

[1] www.calliope.cc
[2] www.scratch.mit.edu
[3] www.calliope.cc/los-geht-s/editor
[4] https://scratch.mit.edu/projects/editor/?tip_bar=getStarted (29.11.2018)
[5] Sämtliches Zusatzmaterial ist erhältlich auf www.science-on-stage.de/coding-materialien.
[6] www.microbit.co.uk/home
[7] https://lab.open-roberta.org
[8] https://snap.berkeley.edu

Der Zauberhandschuh

<Autorin> Annamária Lőkös
<Autorin> Camelia Ioana Rațiu

<Info>

<Schlagwörter> Experiment, Umgebung, Temperatur, Feuchtigkeit, Helligkeit, Magnetfeld

<Unterrichtsfächer> Physik, Chemie, Biologie, Ökologie, Informatik

<Altersgruppe> 10–18 Jahre
<Stufe 1> Primarstufe (Alter: 10–11 Jahre) und Sekundarstufe I (Alter: 12–15 Jahre)
<Stufe 2> Sekundarstufe II (Alter: 15–18 Jahre)

<Hardware> Arduino UNO[1], mit Arduino kompatible Sensoren (z. B. Lichtsensor, Temperatursensor, Magnetfeldsensor, Feuchtigkeitssensor, Gassensor), LCD-Tasten-Shield, Schaltdrähte, externe Batterie

<Programmiersprache> C[2], Arduino 1.8.5[3], Snap![4]

<Programmierniveau> mittel

<Zusammenfassung>

Die Technologiebegeisterung von Kindern und Jugendlichen sorgt für den Erfolg dieser Unterrichtseinheit, bei der Naturwissenschaften mit Informatik verbunden werden. Die Schülerinnen und Schüler entwerfen und bauen einen Handschuh mit unterschiedlichen Sensoren an jedem Finger. Sie können die verschiedensten Experimente durchführen, indem sie einfach den benötigten Sensor anschließen.

<Vorstellung des Konzepts>

Ein Gerät (Handschuh) mit mehreren Sensoren für unterschiedliche Messungen bietet zwei Vorteile. Zum einen können die Schülerinnen und Schüler der Primar- und Sekundarstufe mit dem „magischen" Handschuh Temperatur, Helligkeit, Feuchtigkeit, das Vorhandensein eines Magnetfelds, Schallintensität usw. messen. Dazu wählen sie einfach den gewünschten Sensor aus, und schon sind sie bereit die zahlreichen Anwendungsmöglichkeiten des Handschuhs in verschiedenen Unterrichtsfächern und Themenbereichen zu entdecken. Zum anderen kann der Handschuh, da er batteriebetrieben ist, nach draußen mitgenommen werden, was das Forschen außerhalb des Labors ermöglicht. Schülerinnen und Schüler der Sekundarstufe II können selbst einen Handschuh konstruieren, um bestimmte Experimente durchzuführen. Sie sind bereits mit Theorien aus verschiedenen naturwissenschaftlichen Fachbereichen (Physik, Chemie, Biologie, Ökologie) vertraut und genießen die Möglichkeit, diese praktisch zu erforschen.

Damit die Schülerinnen und Schüler mit Arduino[1] programmieren können, müssen ihnen die Lehrkräfte zuerst die Grundkonzepte der Programmierung in C[2] oder einer anderen von Arduino unterstützten Programmiersprache, z. B. Snap![4], vermitteln. Die Schülerinnen und Schüler können sich diese Grundkenntnisse aneignen, indem sie sich Tutorials anschauen. Dadurch bekommen sie ein besseres Verständnis dafür, wie Programme geschrieben sein sollten. Sie werden überrascht sein, wie einfach die Programmierung ist und ihr Selbstvertrauen wird gestärkt.

<Praktische Umsetzung>

<Stufe 1>

Der Handschuh verfügt über eine LCD-Anzeige mit Tasten. Die Schülerinnen und Schüler ziehen den Handschuh an und wählen den gewünschten Sensor mit den **UP**/**DOWN**-Tasten aus, drücken dann die **SEL**-Taste und die Messung beginnt. Das Display zeigt einen Wert an und die Messung kann beliebig oft wiederholt werden. Um zur Startseite zu gelangen, drücken sie die **BACK**-Taste. Als Beispiel sehen Sie die Bestimmung der Pole eines Magneten in 📷 1a–1c.

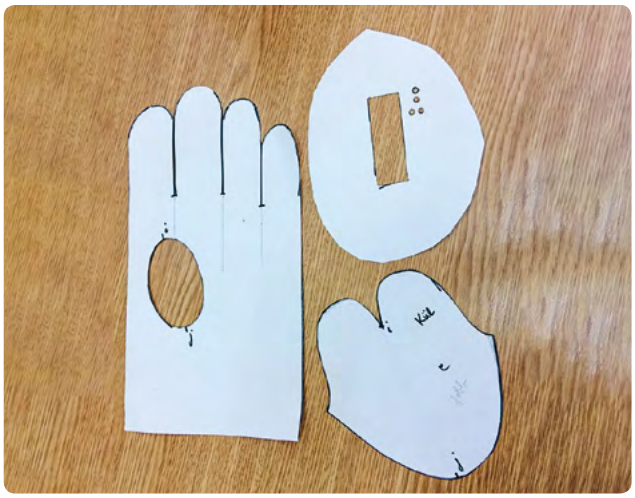
📷 1a-c: Bestimmung der Pole eines Magneten

<Stufe 2>

Die Klasse kann in vier Gruppen aufgeteilt werden. Eine Gruppe schneidet die Handschuhteile zu und näht sie zusammen, eine erstellt den Schaltkreis, eine weitere übernimmt die Programmierung und die vierte Gruppe kalibriert die Sensoren.

<Herstellung des Handschuhs>

Die Schülerinnen und Schüler erstellten ein Schnittmuster (📷 2), nachdem sie Anleitungen im Internet recherchiert hatten.[5] Sie falteten das Material (in unserem Fall Leder, aber auch andere Materialien sind geeignet) drei Mal und schnitten die Teile anhand des Schnittmusters aus. Dann nähten sie zwei der identischen Teile zusammen. Das letzte Teil wurde aufgenäht, nachdem der Arduino mit der LCD-Anzeige und den Sensoren auf dem Handrücken angebracht wurde. Die Öffnungen für die LCD-Anzeige und die Tasten wurden aus diesem dritten Teil ausgeschnitten.



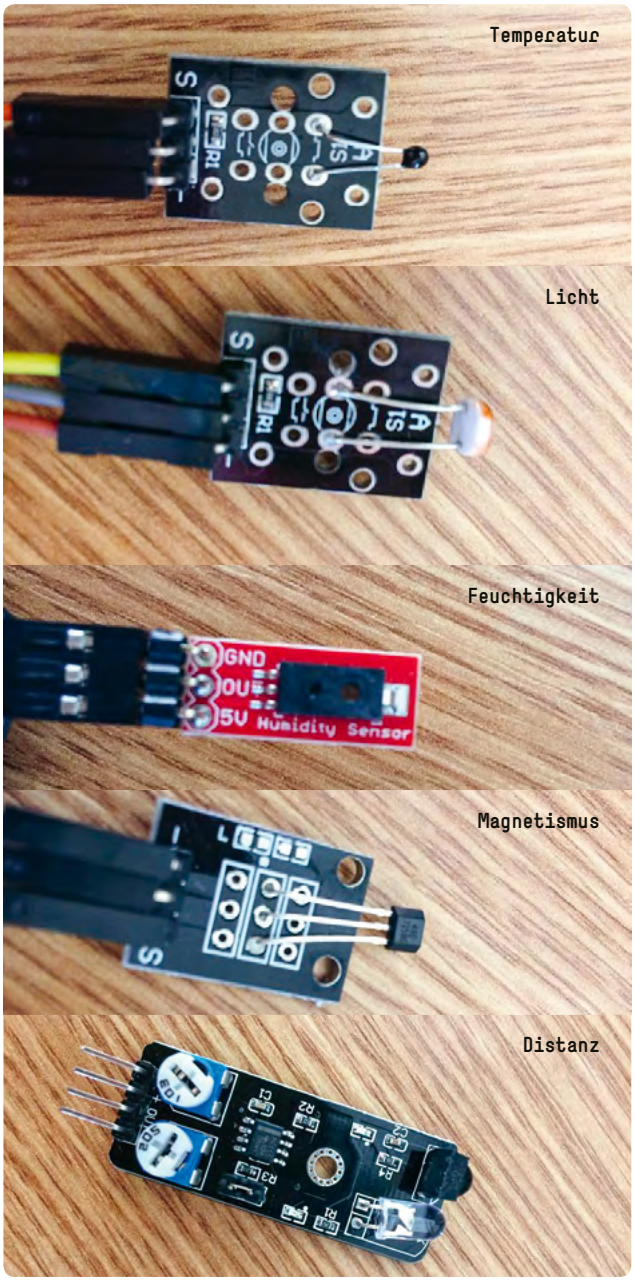
📷 2: Schnittmuster des Handschuhs

<Aufbau des Schaltkreises>

Die Schülerinnen und Schüler zeichneten zuerst einen Schaltplan, den sie vorher mit der Lehrkraft besprochen und analysiert hatten. Der Schaltkreis kann entweder fixiert (verzinnt) oder nicht fixiert werden. Wichtig ist, dass die Sensoren an die richtigen Pins am Arduino angeschlossen werden, nämlich der **GND** des Sensors an den **GND** der Platine, der **VCC** des Sensors an den **5V** der Platine, und der **OUT** des Sensors an den **ANALOG IN** (A0, A1, A2, A3, A4 oder A5) der Platine. Wenn ein Sensor an den DIGITAL IN angeschlossen werden soll, muss darauf geachtet werden, dass nicht einer der Eingänge für die LCD-Anzeige verwendet wird, da sonst Betriebsfehler auftreten. In unserem Beispiel haben wir die folgenden Sensoren angeschlossen: Temperatur (A1), Licht (A2), Feuchtigkeit (A3), Magnetismus (A4) und Distanz (A5) (📷 3a–3e).[6]

<Programmieren des Schaltkreises>

Schülerinnen und Schüler der Sekundarstufe II, die die Programmiersprache C[2] schon erlernt haben, können den Arduino[1] leicht programmieren. Online gibt es zahlreiche Tutorials in vielen Sprachen, unsere Klasse konsultierte beispielsweise eine Webseite auf Rumänisch[7]. Unter anderem gibt es englische Tutorials auf der Webseite von Arduino oder auf denen der verschiedenen Anbieter von Mikrocontrollern und Sensoren.[8] Natürlich sind noch viele mehr im Internet zu finden.



📷 3a-e: Sensoren

Die Lehrkraft kann beim Schreiben des Programms für den Arduino unterstützen. Das gesamte von uns verwendete Programm findet sich online.[9]

<Sensoren kalibrieren>

Natürlich sind bereits kalibrierte Sensoren erhältlich, aber es gibt auch nicht kalibrierte, und es macht den Schülerinnen und Schülern durchaus Spaß herauszufinden, wie man sie kalibriert. Die Kalibrierformeln einiger Sensoren finden sich im Internet. So gibt es für den Feuchtigkeitssensor-Baustein[6] etwa eine Formel, da die Funktion, die die Variation der angezeigten Werte beschreibt, nicht linear ist.

Zur Kalibrierung des Temperatursensors wurden die Werte notiert, die der Sensor anzeigte. Es wurde ein kalibriertes Ther-

mometer im Labor hinzugezogen und der angezeigte Wert mit dem Thermometerwert in Verbindung gebracht. Die Schülerinnen und Schüler fanden heraus, dass dieser Sensor linear variiert und erarbeiteten so die Kalibrierformel. Im Zusatzmaterial finden Sie weitere Beispiele mit Kalibrierformeln für den Feuchtigkeits- und den Temperatursensor.[9]

Nachdem die Kalibrierung der Sensoren und die Programmierung abschlossen waren, wurde überprüft, ob die Handschuhfinger korrekt zu den auf der Anzeige erscheinenden Daten passten. Dann wurde der Handschuh fertig zusammengesetzt. Der letzte Schritt war das Zusammennähen und Annähen der äußeren Materialschicht. Um die Sensoren besser an den Fingern zu befestigen, verwendeten die Schülerinnen und Schüler Befestigungsringe (📷 4).

📷 4: Befestigung der Sensoren

<Algorithmus in anderen Sprachen>

Wenn Sie in einer anderen Programmierumgebung arbeiten möchten, finden Sie online ein Diagramm mit allen erforderlichen Elementen für das Hauptprogramm.[9]

<Fazit>

Schülerinnen und Schüler entdecken gerne Neues und sind sehr erfinderisch. Sie mögen es zu experimentieren, und der Handschuh sieht außerdem aus, als käme er aus einem Science-Fiction-Film. Schülerinnen und Schüler der Sekundarstufe II schätzen es, dass sie programmieren und sofort sehen können, ob ihr Programm auch wirklich funktioniert.

Dies war die erste Erfahrung dieser Art für uns: Lehrkräfte, Schülerinnen und Schüler lernten gemeinsam sehr viel dazu.

Die Schwierigkeit bei diesem Projekt ist, die richtigen Sensoren[6] zu finden und zu kalibrieren, aber es gibt Lösungen hierfür. Wenn eine Kalibrierformel einmal nicht zu finden oder nicht verfügbar ist, können bereits kalibrierte Sensoren eingesetzt werden, die jedoch teurer sind.

Dieser Handschuh kann auch mit dem Calliope mini konstruiert werden, wodurch er leichter und kleiner wird. Unsere Schülerinnen und Schüler wollen solch einen Handschuh entwickeln, auch wenn die Programmiersprache eine andere ist.

<Kooperationsmöglichkeiten>

Schülergruppen aus verschiedenen Schulen und Ländern können diese Handschuhe unter Verwendung unterschiedlicher Mikrocontroller und geeigneter Sensoren bauen, sich darüber austauschen und ihre Ergebnisse vergleichen. Beim Handschuhdesign ist die Unterstützung einer Lehrkraft aus dem Fachbereich Kunst denkbar. Eine weitere Option ist, einen Wettbewerb unter Schulen zu organisieren, bei dem die Schülerinnen und Schüler ihre eigenen Designs einreichen.

Der Handschuh lässt sich leicht per Post verschicken, sodass Schülerinnen und Schüler mit einem Handschuh aus anderen Schulen oder Ländern experimentieren können.

<Quellen und Hinweise>

[1] www.arduino.cc

[2] https://de.wikipedia.org/wiki/Java_(Programmiersprache)

[3] www.arduino.cc/en/Main/Software

[4] https://snap.berkeley.edu

[5] Verschiedene Webseiten mit Unterlagen und Tutorials für das Nähen von Handschuhen: http://ofdreamsandseams.blogspot.ro/2012/04/1950s-hand-sewn-leather-gloves.html, https://so-sew-easy.com/easy-gloves-pattern-winter-comfort, http://sew-ing.com/make/gloves.html, www.glove.org/Modern/myfirstgloves.php, www.instructables.com/id/How-to-Make-Gloves (alle Dezember 2018)

[6] Wir verwendeten die Sensoren im Sensoren-Kit 37 in 1 von Arduino. Der Feuchtigkeitssensor, den wir nutzen, ist der HIH-4030 von Sparkfun, code: DEN-VRM-09.

[7] www.robofun.ro (Tutorials auf Rumänisch: Jedes Produkt enthält eine Anleitung, wie es an den Schaltkreis angeschlossen und welche Programmiersprache verwendet werden muss. Die Webseite enthält Diagramme und Skizzen, Sensoren oder andere Komponenten sind auf Fotos abgebildet und das Programm ist explizit geschrieben, sodass Rumänischkenntnisse nicht erforderlich sind.)

[8] http://mthackathon.info/resources/37-SENSOR-KIT-TUTORIAL-FOR-UNO-AND-MEGA.pdf (Die Tutorials sind in Englisch und enthalten Anleitungen zur Verbindung der Sensoren aus dem Kit an den Arduino.) (Dezember 2018)

[9] Sämtliches Zusatzmaterial ist erhältlich auf www.science-on-stage.de/coding-materialien.

Science Friction

<Autor> Ilia Mestvirishvili
<Autor> David Shapakidze

<Info>

<Schlagwörter> Reibungskraft, Bremsweg, Anti-Blockier-System (ABS), App-Programmierung, Datenerhebung

<Unterrichtsfächer> Physik, Informatik, Mathematik

<Altersgruppe> 14+ Jahre

<Hardware> Arduino[1], Servomotor, Motor, Bluetooth-Modul, Motor-Shield, Photogate

<Programmiersprache> Arduino Programmierumgebung[2], AppInventor[3], Snap4Arduino[4], Blockly[5]

<Programmierniveau> leicht, mittel

<Zusammenfassung>

Mit dem Bau eines preisgünstigen, Bluetooth-gesteuerten Autos mit einfachem Bremssystem wird die Untersuchung der verschiedenen Faktoren, die die Reibungskraft beeinflussen, zu einem interessanten und unterhaltsamen Experiment. So können die Schülerinnen und Schüler Echtdaten beobachten, wie z. B. die Geschwindigkeit vor dem Abbremsen, den Bremsweg und welchen Einfluss die Masse des Autos und die Oberflächenbeschaffenheit auf die Reibungskraft haben. In Experimenten untersuchen sie mit guter Genauigkeit die Beziehung zwischen diesen Faktoren und können so eigene und von der Lehrkraft vorgeschlagene Hypothesen überprüfen.

<Vorstellung des Konzepts>

Die Reibung ist eine sehr wichtige Kraft im Alltag und wird im Physikunterricht sowohl auf Mittel- als auch Oberstufenniveau vermittelt. Herkömmliche Experimente mit der Reibungskraft sind jedoch begrenzt und nicht sehr unterhaltsam. Dieses Projekt macht die Erforschung der Reibungskraft zu einer spannenden Gruppenarbeit mit folgenden Zielen:

1. Bau und Feinabstimmung eines Autos
2. Programmierung eines Arduino Mikrocontrollers[1] für die Messung der Momentangeschwindigkeit und des Bremsweges
3. Programmierung einer Smartphone-App mit AppInventor[3] für das Senden, Empfangen und Anzeigen von Echtdaten auf dem Smartphone

<Praktische Umsetzung>

Da diese drei Aufgaben zuerst einmal parallel erarbeitet und gelöst werden können, empfehlen wir, Gruppen mit je zwei oder drei Schülerinnen und Schülern getrennt daran arbeiten zu lassen. Hierbei können sich die Arbeitsgruppen austauschen.

Wenn im Lehrplan die Einführung der Reibung vorgesehen ist, können die Lehrkräfte die Inhalte parallel zu diesem Projekt unterrichten, wodurch die Schülerinnen und Schüler motivierter sind und die theoretischen Konzepte leichter verstehen. Am besten startet man das Projekt mit den folgenden Fragen und gibt den Schülerinnen und Schülern dann Zeit zur Erarbeitung eigener Ideen, Voraussagen und Hypothesen:

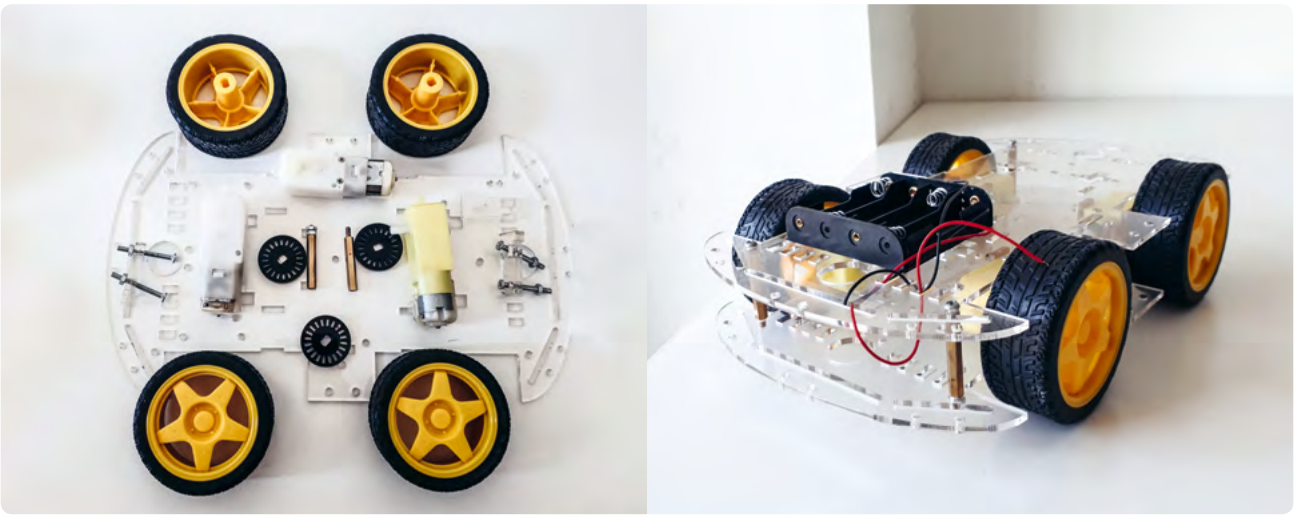
- ↳ Welche Beziehung besteht zwischen der Geschwindigkeit eines Autos und dem Bremsweg? (Die Antworten können verschieden ausfallen, z. B. „der Bremsweg verhält sich proportional zur Geschwindigkeit vor dem Abbremsen“, oder einige werden sich aus dem Physikunterricht „erinnern“, dass sich der Bremsweg proportional zum Quadrat der Geschwindigkeit verhält.)
- ↳ Wie würde eine größere Automasse den Bremsweg beeinflussen, vorausgesetzt, dass alle anderen Faktoren nicht verändert werden? (Eine gängige Antwort ist, dass die Erhöhung der Masse den Bremsweg verlängert.)
- ↳ Wie sollten wir die Bremsen einsetzen, um das Auto so rasch wie möglich zum Stillstand zu bringen? (Mögliche Antworten: am besten die Räder vollständig blockieren; die Räder in die entgegengesetzte Richtung drehen lassen, um das Auto rascher anzuhalten; usw.)
- ↳ Wenn die Vorder- und Hinterradbremsen identisch sind, wird das Auto in der gleichen Zeit zum Stillstand kommen? (Schülerinnen und Schüler können ihre eigenen Erfahrungen mit Fahrrädern einbringen.)
- ↳ Weitere Fragen können von der Lehrkraft oder den Schülerinnen und Schülern kommen.

Nachdem die Schülerinnen und Schüler ihre anfänglichen Ideen notiert haben, überlegen sie im nächsten Schritt, wie sie ein einfaches Auto bauen können und welche Daten sie benötigen, um ihre anfänglichen Ideen zu prüfen und weiterzuentwickeln. Die Lehrkraft kann diesen Prozess vereinfachen und vorschlagen, ein Auto zu bauen, das relevante Daten sammeln und an ein Smartphone senden kann, welches wiederum sowohl das Auto steuert als auch Daten empfängt und anzeigt.

Die Schülerinnen und Schüler können sich nun je nach ihren Interessen, Kenntnissen und Präferenzen in die eingangs genannten drei Gruppen aufteilen. Natürlich kann auch nur eine Gruppe alle diese Aufgaben erarbeiten und lösen. Die nächsten Schritte im Projekt sind für beide Szenarien die gleichen.

<Bau eines Fahrgestells und Einbau der elektrischen Komponenten>

Bei diesem Ansatz wird ein Auto mit dem kostengünstigen und jederzeit erhältlichen, in 📷 1 gezeigten Arduino[1] Fahrgestellbausatz gebaut. Wir möchten Lehrkräfte, sowie Schülerinnen und Schüler dazu ermutigen, verschiedene Ansätze zur Planung und Umsetzung auszuprobieren, z. B. unterschiedliche



1: Vollständiger Fahrgestellbausatz

Methoden zur Datenerfassung und -übermittlung, zur Fernsteuerung des Autos sowie verschiedene Applikationen und Programmiersprachen.

Nachdem das Auto zusammengebaut und entschieden wurde, wo der Arduino und der Motor installiert werden, machen sich die Schülerinnen und Schüler Gedanken über die verschiedenen Möglichkeiten zur Messung der Autogeschwindigkeit. Wir empfehlen, ein Brainstorming durchzuführen und ihnen die Gelegenheit zu bieten, eigene Ideen vorzubringen. Mit der richtigen Unterstützung durch die Lehrkraft ist die beste und einfachste Vorgehensweise der Einsatz eines Photogates, damit die Drehgeschwindigkeit eines frei drehenden Hinterrads gemessen werden kann, was danach eine Messung sowohl der momentanen Geschwindigkeit als auch der zurückgelegten Distanz erlaubt. Dies kann entweder mit dem Material des Arduino 4-Rad-Fahrgestell-Bausatzes (4 wheel chassis kit) gemacht oder separat entwickelt werden, wenn die Lehrkraft dies bevorzugt.

Hier ist etwas Mathematik erforderlich, um die Anzahl der vom Photogate ermittelten Blockierereignisse in Geschwindigkeit oder Strecke umzuwandeln. Der Bausatz enthält eine Scheibe mit 22 Löchern und ein Rad mit einem Durchmesser von 5,1 ± 0,1 cm. So lässt sich leicht berechnen, dass ein vom Photogate ausgehender Impuls, der 1/22 einer vollen Radrotation ist, einer zurückgelegten Distanz $d = 0{,}72$ cm entspricht. Gleichzeitig misst und sendet das Photogate ein Zeitintervall t in Millisekunden zwischen den fortlaufenden Impulsen. Indem 0,72 cm durch dieses Zeitintervall dividiert wird, wird die momentane Geschwindigkeit berechnet.

Unabhängig davon, ob Sie mit drei Gruppen oder nur mit einer arbeiten, können die folgenden Schritte ausgeführt werden. Eine einzelne Gruppe arbeitet dabei diese Schritte einfach nacheinander ab, während drei Gruppen sich die Aufgaben teilen.

<Arduino-Programmierung>

Die Gruppe, die den Arduino[1] programmiert, arbeitet am Programm auf folgende Weise:

1. Sie definiert die Aktionen und daraus folgend die Methoden und Funktionen, die für die Erhebung und Bluetooth-Übermittlung der erforderlichen Daten benötigt werden,
2. schreibt und testet jede Methode separat und
3. fügt alles zusammen.

Für Anfängerinnen und Anfänger wäre es gut, mit TinkerCad[6] zu starten, womit die Arduino-Schaltkreise online geplant und getestet werden können, sodass Ausbrennen oder ein Kurzschluss schon in der Prototyp-Phase vermieden werden können.

Wir erklären nun jeden Schritt detaillierter:

1. Die erforderlichen Aktionen sind: Motor starten und ausschalten, Bremsen aktivieren und deaktivieren, Distanz messen, Geschwindigkeit messen, Daten über Bluetooth senden und empfangen.
2. Der entscheidende Teil ist das Schreiben je eines Programms zur Messung der Geschwindigkeit und der zurückgelegten Distanz während eines Experiments. Beide Programme nutzen die Impulse des Photogates und werden aktiviert, wenn von der Smartphone-App via Bluetooth „2“ empfangen wird:
   - ↳ Für die Distanzmessung gibt es einen Zähler, der die Impulse zu zählen beginnt, die von einem frei drehenden Hinterrad an den Arduino gesendet werden, sobald die Bremsen an den Vorderrädern aktiviert werden.
   - ↳ Um die momentane Geschwindigkeit zu messen, werden die Zeitintervalle zwischen den Impulsen erfasst. Die Distanz, die ein Hinterrad während eines Impulses zurücklegt, beträgt 0,72 cm. Somit muss dieser Wert durch das Zeitintervall zwischen den Impulsen dividiert werden.

↳ Die Funktionalität eines Anti-Blockier-Systems (ABS) könnte umgesetzt werden, indem die Bremsen einmal alle 50–200 ms (experimentell optimiert) aktiviert und deaktiviert werden, was in den meisten Fällen zu einem kürzeren Bremsweg führt.

3. Bei der Zusammenstellung eines Programms für den Arduino muss beachtet werden, dass alles in einer großen Schleife abläuft. Folglich sind bei einer Unterbrechung des Programms an einer beliebigen Stelle alle anderen Schritte betroffen.

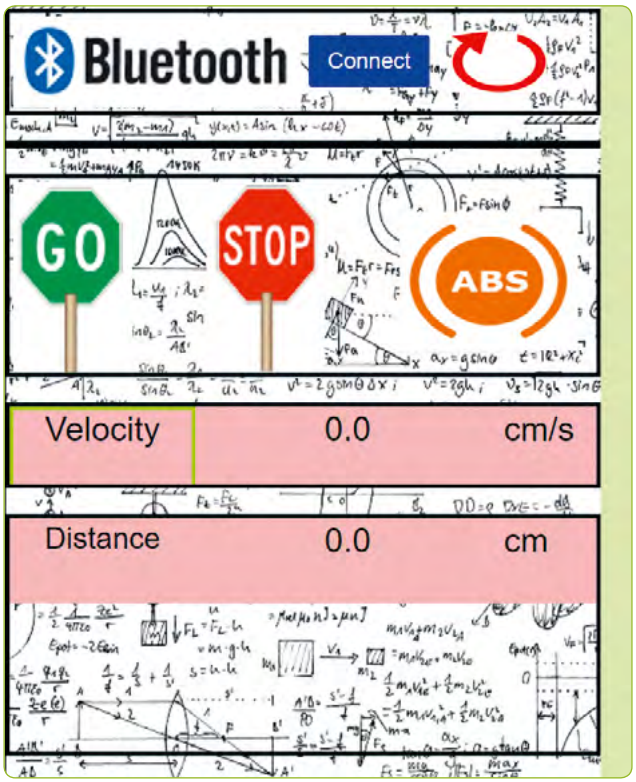
Das Musterprogramm und die Nachweise anderer Quellen für jede dieser Funktionalitäten sind online verfügbar[7], jedoch sollten die Schülerinnen und Schüler mit etwas Unterstützung durch die Lehrkraft ermutigt werden, ihr eigenes Programm zu schreiben.

<Android-Programmierung>

Die Android-Programmiergruppe kann mit AppInventor[3] Möglichkeiten untersuchen, die Daten am Bildschirm anzeigen zu lassen (Benutzeroberfläche). Die Schülerinnen und Schüler entscheiden, wo und wie sie die Tasten für die Autosteuerung sowie die Anzeigefelder und Beschriftungen für die Datenanzeige auf dem Bildschirm anordnen (📷 2). Der Code für das AppInventor-Programm ist online verfügbar[7] und hat die folgenden Funktionalitäten:

1. Das Drücken der **START**-Taste sendet via Bluetooth eine „1“ an den Arduino, wodurch der Motor gestartet wird.
2. Das Drücken der **STOP**-Taste sendet via Bluetooth eine „2“ an den Arduino, wodurch zuerst der Motor ausgeschaltet und dann die Bremsen aktiviert werden.
3. Das Drücken auf **ABS** sendet via Bluetooth eine „3“ an den Arduino, wodurch zuerst der Motor ausgeschaltet wird und dann die Bremsen in regelmäßigen Intervallen aktiviert und deaktiviert werden (Simulation der ABS-Funktion).
4. Nachdem **STOP** oder **ABS** gedrückt wurde, werden die empfangenen Daten über die momentane Geschwindigkeit vor dem Stoppen und über die Distanz, die das Auto nach dem Aktivieren der Bremsen zurückgelegt hat, d. h. der Bremsweg, in zwei entsprechenden Feldern mit den Beschriftungen „Geschwindigkeit“ (Velocity) bzw. „Distanz“ (Distance) angezeigt.
5. Die Taste **ZURÜCKSETZEN** sendet via Bluetooth „0“ an den Arduino, wodurch die Daten zur Geschwindigkeit und zur Distanz gelöscht werden und der Arduino zurückgesetzt wird.

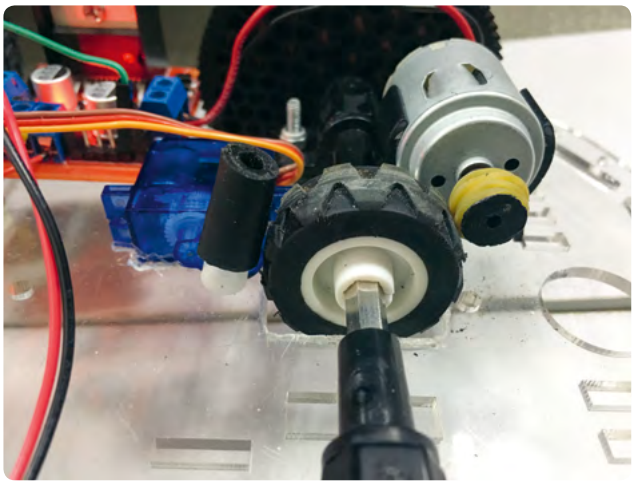
Wir empfehlen, das zur Verfügung gestellte Programm als Referenz für Lehrkräfte zu nutzen und den Schülerinnen und Schülern die Gelegenheit zu geben, AppInventor[3] auszuprobieren und ihr eigenes Programm mit den genannten Funktionalitäten zu schreiben.



📷 2: Benutzeroberfläche der App

<Bau des Autos>

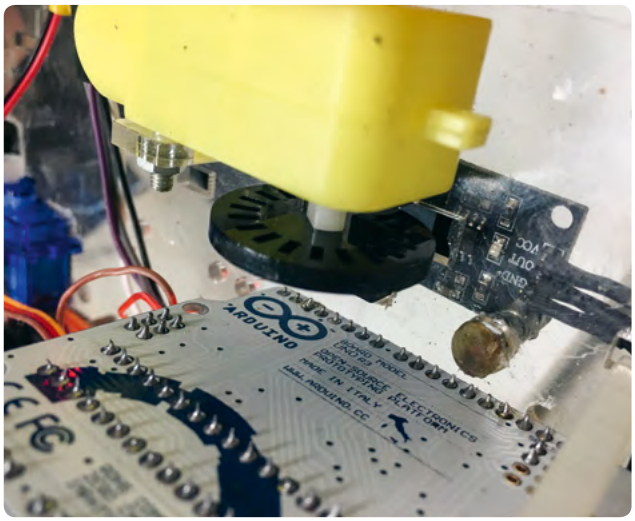
Die Gruppe, die das Auto baut, muss überlegen, wo und wie die Komponenten, d. h. Motor, Photogate, Servomotor, Batterien, Bluetooth-Modul, Motor-Shield und die Arduino-Platine selbst, befestigt werden sollen. Es ist wichtig sicherzustellen, dass der sich drehende Servomotor einen Griffzapfen mit einem Gummischlauch fest gegen die drehende Scheibe drückt, die an der Vorderradachse befestigt ist (📷 3).



📷 3: Das Bremssystem

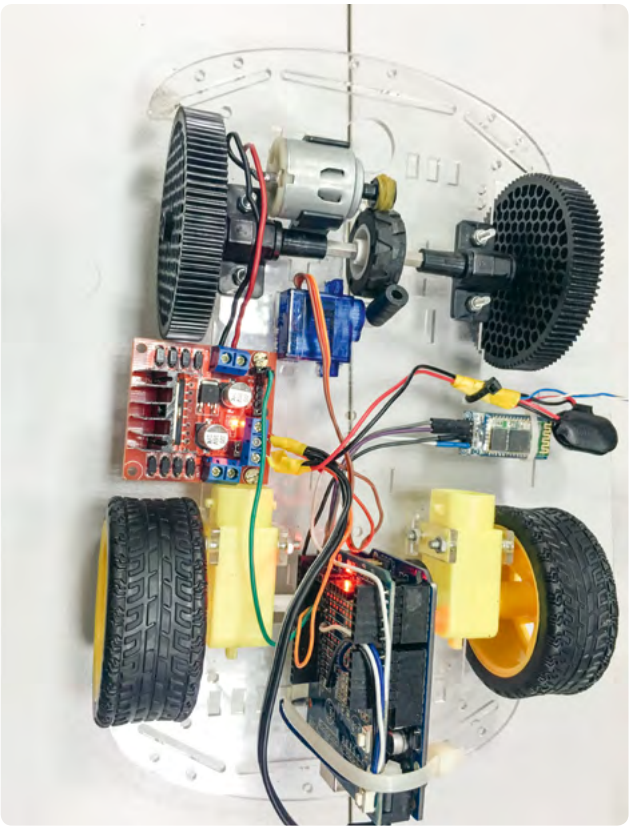
Es muss ausreichend Kraft ausgeübt werden, um die Räder sofort anzuhalten. Die Position des Photogates ist ebenso wichtig. Außerdem müssen alle Impulse korrekt gezählt werden – der empfohlene Photogate-Arduino-Sensor verfügt über eine eingebaute LED, die blinkt, sobald etwas in die Photogate-

Lücke kommt. Stellen Sie also sicher, dass das Photogate die Umdrehungen korrekt zählt, wenn sich die Hinterräder drehen (📷 4).



📷 4: Das Photogate

Außerdem müssen sich die Hinterräder jederzeit so frei wie möglich drehen. Denken Sie bitte daran, dass die freie Drehung der Hinterräder dazu dient, die Geschwindigkeit und die Distanz zu messen. Wenn das Auto fertig gebaut ist, sollte es ungefähr so aussehen wie in 📷 5.



📷 5: Das fertige Auto

Wir empfehlen, detaillierte Richtlinien bereitzuhalten, aber den Schülerinnen und Schülern die Gelegenheit zu bieten, eigene Lösungen zu entwickeln und umzusetzen.

<Fazit>

Das Projekt bietet großartige Experimentiermöglichkeiten zum Verständnis von Grundkonzepten der Physik wie Haft- und Gleitreibung, aber gleichzeitig auch von relativ modernen Technologien wie ABS. Hier kommen Physik, Elektronik, Programmierung und Design zusammen und ermöglichen es, Faktoren zu erforschen, die den Bremsweg eines Autos beeinflussen. Echte experimentelle Daten sind immer eine Herausforderung, wenn es um Interpretation und Analyse geht. Wichtige Konzepte wie Messunsicherheit, Aussagekraft, Reproduzierbarkeit und Visualisierung müssen mitberücksichtigt werden. Das Projekt ermöglicht es den Schülerinnen und Schülern, die Reibungskraft auf praktische Weise zu erfahren und zu verstehen.

<Kooperationsmöglichkeiten>

Dieses Projekt bietet großes Potential zur Teamarbeit, da seine drei unabhängigen Elemente – Konstruktion des Autos, Programmierung des Arduino[1] und Programmieren mit AppInventor[3] – weiterentwickelt und verbessert werden können. Dabei können alle in ihrer Arbeit gegenseitig voneinander profitieren.

Eine weitere Option zur Zusammenarbeit ist ein Wettbewerb zwischen Schulteams, bei welchem ein Auto mit den gleichen Rädern und der gleichen Masse möglichst schnell zum Stillstand gebracht werden muss; unter der Voraussetzung, dass der „Straßen"-Belag und die Geschwindigkeit vor dem Abbremsen dieselben sind.

Ein besonderer Dank gilt unseren griechischen Kollegen: Astrinos Tsoutsoudakis für seine wichtigen Vorschläge bezüglich der physikalischen Aspekte dieses Projekts und Georgios Georgoulakis für seine äußerst hilfreichen Programmiertipps. Vielmals bedanken wir uns auch bei Jörg Gutschank für seine ausführlichen Rückmeldungen sowie seine Unterstützung, die dieses Projekt interessanter und leichter reproduzierbar gemacht haben.

<Quellen und Hinweise>

[1] www.arduino.cc
[2] www.arduino.cc/en/Guide/Environment
[3] http://appinventor.mit.edu
[4] http://snap4arduino.rocks
[5] https://developers.google.com/blockly
[6] www.tinkercad.com/circuits
[7] Sämtliches Zusatzmaterial ist erhältlich auf www.science-on-stage.de/coding-materialien.

Rolling Sounds

<Autor> Georgios Georgoulakis
<Autor> Astrinos Tsoutsoudakis

<Info>

<Schlagwörter> Grundlagenwissenschaft, Datenerfassung, Kreisbewegung, Schallwellen, Geometrie, Trigonometrie

<Unterrichtsfächer> Physik, Mathematik, Informatik

<Altersgruppe> 14–17 Jahre

<Hardware> Arduino Mikrocontroller[1] oder ähnliches Gerät (entsprechende Treiber bereits installiert), Mikrofon, Summer mit hoher Ausgangsleistung, Bohrmaschine mit Grundplatte, Material für den Versuchsaufbau aus Holz

<Programmiersprache> Snap4Arduino[2]

<Programmierniveau> mittel

<Zusammenfassung>

Diese fächerübergreifende Unterrichtseinheit kombiniert Physik mit Informatik und kann in beiden Unterrichtsfächern eingesetzt werden. Mit zwei unterschiedlichen Methoden werden hier die Bahngeschwindigkeit der Rotation sowie weitere physikalische Größen bei der gleichmäßigen Kreisbewegung bestimmt.

Anhand der ersten Methode wird ermittelt, wie oft das Signal eines Infrarot-Distanzsensors durch einen kleinen Metallstreifen unterbrochen wird, der auf einer sich drehenden Scheibe befestigt ist, wodurch die Periodendauer gemessen wird. Mit der zweiten Methode wird die Dopplerverschiebung einer Tonquelle, die sich auf der Scheibe befindet, untersucht.

<Vorstellung des Konzepts>

Die experimentelle Untersuchung von physikalischen Größen (Periode T, Frequenz f, lineare Geschwindigkeit v und Winkelgeschwindigkeit ω) einer regelmäßigen Kreisbewegung basiert auf dem Wissen, das in Griechenland in der Oberstufe (Altersgruppe: 14–17 Jahre) vermittelt wird und auch in den Lehrplänen anderer europäischer Länder für die Sekundarstufe zu finden ist. Dies umfasst auch das Kennenlernen des Dopplereffekts. Frequenz und Höhe der Winkelgeschwindigkeit und der Bahngeschwindigkeit werden mit den folgenden bekannten Formeln ermittelt:

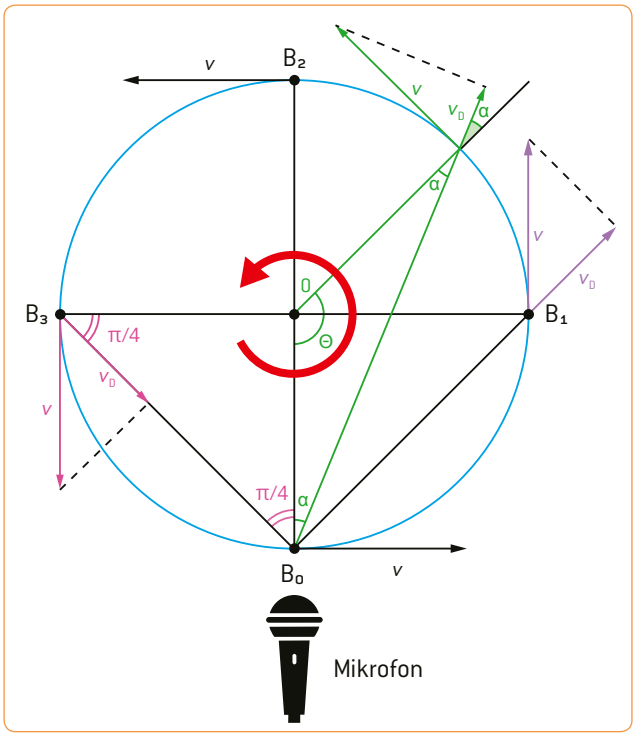
$$f = \frac{1}{T}, \omega = \frac{2\pi}{T} \text{ und } v = \frac{2\pi r}{T}$$

T wird mithilfe der internen Uhr des Mikroprozessors bestimmt, d. h. die Zeit zwischen der Erfassung der einzelnen Signale. Der Radius r steht für die Distanz zwischen dem Metallstreifen – oder dem Summer – und dem Scheibenmittelpunkt.

<Experiment zum Dopplereffekt>

Der Dopplereffekt ist die Veränderung der Frequenz oder Wellenlänge einer Welle, wenn sich die Quelle relativ zum Beobachter bewegt. Ein bekanntes Beispiel aus dem Alltag ist das Phänomen der sich ändernden Sirenentonhöhe eines fahrenden Krankenwagens. Wenn sich der Krankenwagen nähert, ist der wahrgenommene Ton höher als der eigentlich emittierte Ton; wenn sich der Krankenwagen entfernt, ist der Ton tiefer. Nur gerade in dem Moment, in dem der Krankenwagen am Beobachter vorbeifährt, entspricht der wahrgenommene Ton dem gesendeten.

In unserem Experiment benutzen wir einen Summer auf einer Drehscheibe als Tonquelle und ein statisches Mikrofon als Beobachter (siehe 1).



1: Schema des Experiments

Wenn sich die Scheibe gegen den Uhrzeigersinn dreht (1), erhöht sich die Geschwindigkeitskomponente parallel zur Kreissehne, die den Punkt B_0 mit dem Punkt B, wo sich der Summer befindet, verbindet, von null auf ein Maximum bei B_1, und nimmt in der Folge am Punkt B_2 wieder auf null ab. Diese Geschwindigkeitskomponente ist die eigentliche relevante Geschwindigkeit für den Dopplereffekt. Von Punkt B_2 bis B_3 erhöht sich die Geschwindigkeitskomponente, die jetzt für die Annäherungsgeschwindigkeit steht, von null auf ein Maximum bei Punkt B_3 und nimmt dann wieder bei Punkt B_0 auf null ab.

Indem wir die Dopplerverschiebungsformel für eine bewegte Quelle bei B_3 verwenden und der Beobachter dabei stillsteht, können wir die Bahngeschwindigkeit leicht berechnen. Die

Bahngeschwindigkeit bleibt konstant senkrecht zum Kreisradius, und der Winkel von $\frac{\pi}{4}$ ergibt sich aus den geometrischen Eigenschaften des gebildeten rechtwinkligen und gleichschenkligen Dreiecks B_3OB_0.

$$f = f_0 \cdot \left(\frac{v_s}{v_s - v_D}\right) \Rightarrow f = \frac{f_0 \cdot v_s}{v_s - v_D} \Rightarrow f \cdot v_s - f \cdot v_D = f_0 \cdot v_s$$

$$\Rightarrow f \cdot v_D = (f - f_0) \cdot v_s \Rightarrow v \cdot \cos\left(\frac{\pi}{4}\right) = \left(\frac{f - f_0}{f}\right) v_s$$

$$\Rightarrow v = \left(\frac{f - f_0}{f}\right) \frac{v_s}{\cos\left(\frac{\pi}{4}\right)}$$

f: gemessene Frequenz
f_0: emittierte Frequenz
v: Bahngeschwindigkeit
v_s: Schallgeschwindigkeit
v_D: für den Dopplereffekt wirksame Geschwindigkeit

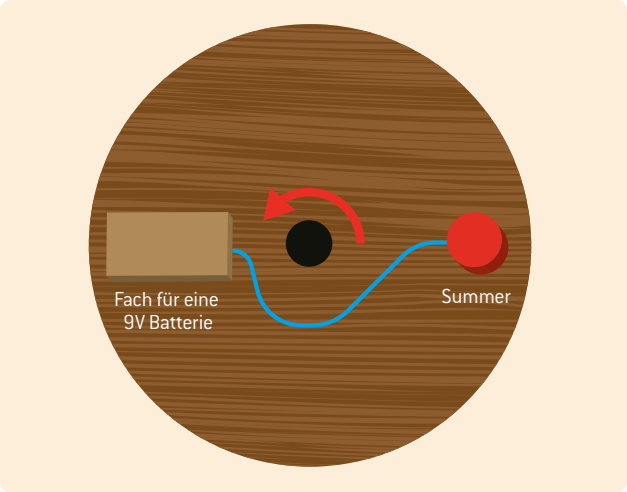
Da die Umsetzung der schnellen Fourier-Transformation (Fast Fourier Transform, FFT) für die Extraktion des Frequenzinhalts eines produzierten Tons die Kenntnisse und Fähigkeiten der Schülerinnen und Schüler weit übersteigt, erstellt eine freie Toneditiersoftware wie Audacity[3] eine geeignete Textdatei mit allen erforderlichen Daten.

<Zusätzliches Material>

Eine weitere Methode, bei der ein Staudruck-Rohrsystem und ein Differenzdrucksensor verwendet werden, wurde aus Gründen der Vereinfachung absichtlich nicht in diese Unterrichtseinheit miteinbezogen, jedoch sind alle Angaben dazu online verfügbar.[4] Das Online-Material enthält eine detaillierte Beschreibung des Versuchsaufbaus, alternative Konstruktionsmöglichkeiten sowie eine theoretische Dokumentation und eine schrittweise Analyse der verwendeten Verfahren.

<Praktische Umsetzung>

Bezüglich der physikalischen Aspekte der Unterrichtseinheit messen die Schülerinnen und Schüler die physikalischen Größen der Kreisbewegung bei unterschiedlichen Radien und erforschen den Dopplereffekt. Zuerst planen und konstruieren sie jedoch einen grundlegenden Versuchsaufbau.



3: Die Scheibe von oben betrachtet

<Der Versuchsaufbau>

Die Schülerinnen und Schüler konstruieren einen Versuchsaufbau mit einer Holzscheibe, die von einer Bohrmaschine an-

2: Grundlegender Versuchsaufbau

getrieben wird. Ein von einer 9V-Batterie betriebener Summer ist auf der Scheibe befestigt. Außerhalb angebracht, jedoch in unmittelbarer Entfernung, sendet ein Infrarot-Distanzsensor dem Mikrocontroller bei jeder vollen Rotation ein Signal, während ein preisgünstiges Mikrofon den erzeugten Ton aufnimmt. Die Dopplerverschiebung sollte idealerweise bei der gewählten Rotationsgeschwindigkeit hörbar, jedoch aus Sicherheitsgründen niedrig sein. (◎ 2 und 3)

Um alle erforderlichen Parameter zu erfassen und die berechneten Werte zu erhalten, wurde eine ansprechende, schülerfreundliche Benutzeroberfläche entwickelt (◎ 4).

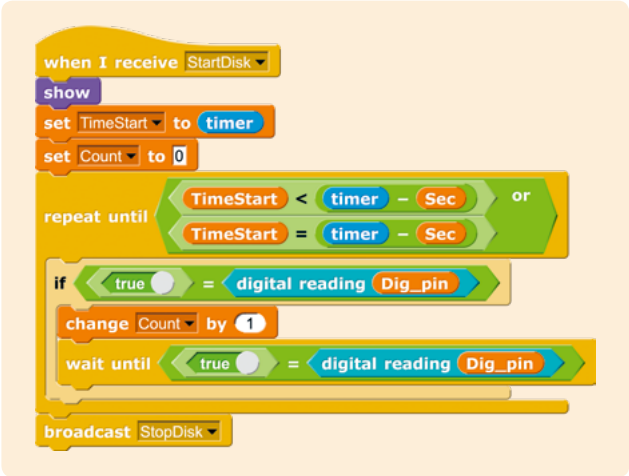
◎ 4: Die Benutzeroberfläche für das Experiment

Die Schülerinnen und Schüler benötigen Grundprogrammierkenntnisse und eine gewisse Erfahrung mit blockbasierten Programmiersprachen (wie Scratch oder Snap!). Der Hauptgrund, warum wir eine Vorlage zur Erstellung des Programms anbieten, ist, dass der Schwerpunkt auf den Zielen der Unterrichtseinheit liegen soll und nicht auf der Benutzeroberfläche und der grafischen Ausgestaltung des Programms.

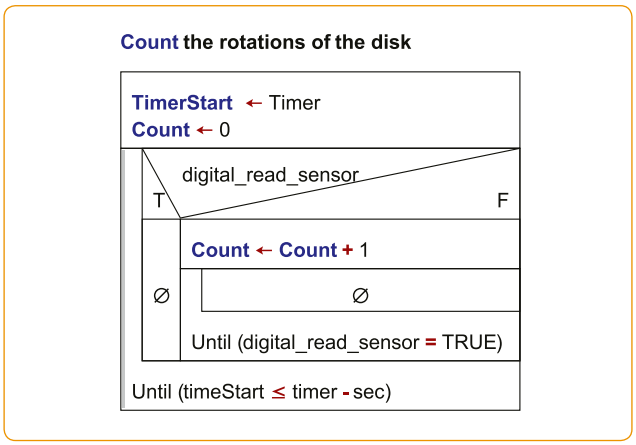
Deshalb stellen wir die Basisvorlage im xml-Format in Snap![5] und ein Aufgabenblatt zur Verfügung, das die grundlegenden Anweisungen zur Vorlage enthält und beschreibt, was wir von den Schülerinnen und Schülern erwarten. Sowohl die Vorlage als auch die Aufgaben sind online verfügbar.[4]

Die Schülerinnen und Schüler überprüfen und validieren die erhobenen Daten, kommunizieren mit den externen Geräten, erhalten und verarbeiten Sensordaten und schreiben einen einfachen seriellen Suchalgorithmus.

Für die Lehrkraft steht als Referenz auch das fertige Programm online zur Verfügung.[4] ◎ 5 zeigt als Beispiel einen Screenshot der Snap4Arduino Programmierumgebung[2]. Das entsprechende Nassi-Shneiderman-Diagramm ist in ◎ 6 dargestellt.



◎ 5: Berechnung der Rotationsperiode

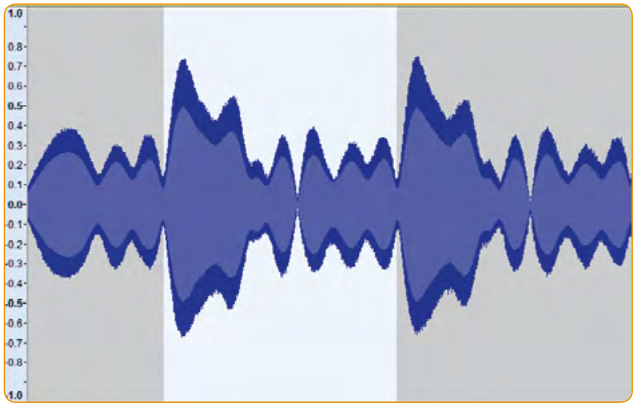


◎ 6: Nassi-Shneiderman-Diagramm für die Periode

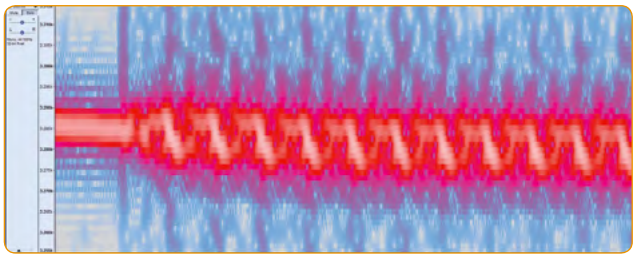
Wie bereits erwähnt, nutzen wir eine freie Toneditiersoftware wie Audacity[3] zur Extraktion des Frequenzinhalts des produzierten Tons. Die Software liefert eine geeignete Textdatei mit allen erforderlichen Daten und die Schülerinnen und Schüler lernen, wie ein Tonsignal mithilfe einer spezialisierten Software verarbeitet werden kann.

Der Signalimport und die wichtigsten Verarbeitungsschritte sind in den ◎ 7–9 dargestellt, während ◎ 10 einen Teil des abschließenden Datenexports zeigt. Um eine vertiefte Analyse zu vermeiden und den Schülerinnen und Schülern ein besseres Verständnis zu ermöglichen, muss hier eine grobe Annahme getroffen werden. Die gelb markierte Frequenz, welche die höchste Lautstärke aufweist, ist die Frequenz des stillstehenden Summers oder diejenige, die an den Punkten B_0 und B_2 gemessen wird, während die blauen und grünen Werte die Werte darstellen, die zwischen unseren primären Frequenzverschiebungen als nächstgelegene Spitzen liegen (◎ 10). Das vorgeschlagene Programm sucht jedoch nur die grünen Werte. Für eine größere Genauigkeit kann es aber auf einfache Weise so modifiziert werden, dass es beide Werte findet. Um die Suche zu beschleunigen, ist es möglich den Datenbereich

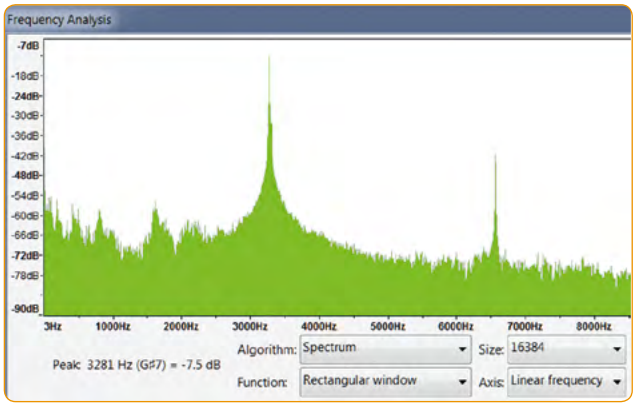
auch auf nur 50–100 Werte oberhalb und unterhalb der Frequenz beim Höchstwert zu begrenzen.



7: Aufgenommene Schallwellenform

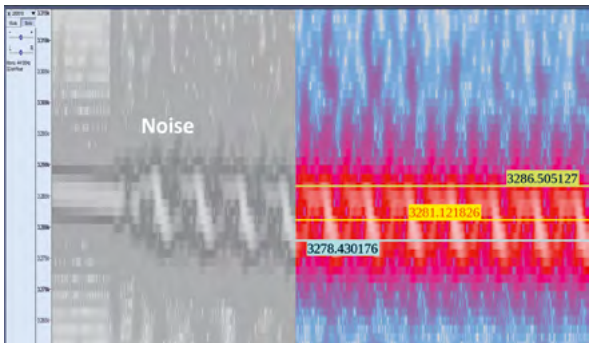


8: Spektrogramm mit Dopplerverschiebung



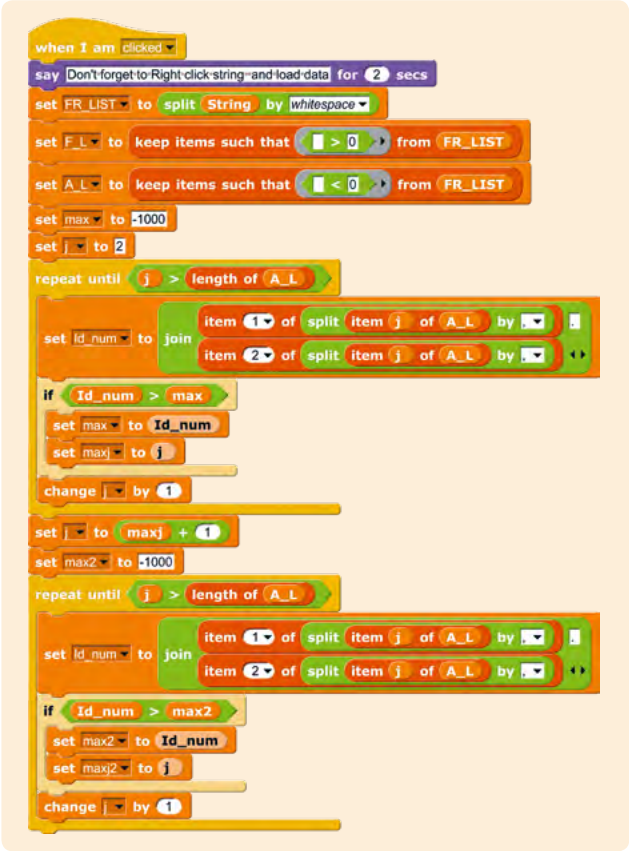
9: Frequenzanalyse durch schnelle Fourier-Transformation

Frequenz (Hz)	Level (dB)
3273.046875	-27.597595
3275.738525	-22.331339
3278.430176	-12.437067
3281.121826	-7.5547090
3283.813477	-10.041918
3286.505127	-9.7750780
3289.196777	-16.848948
3291.888428	-26.916197

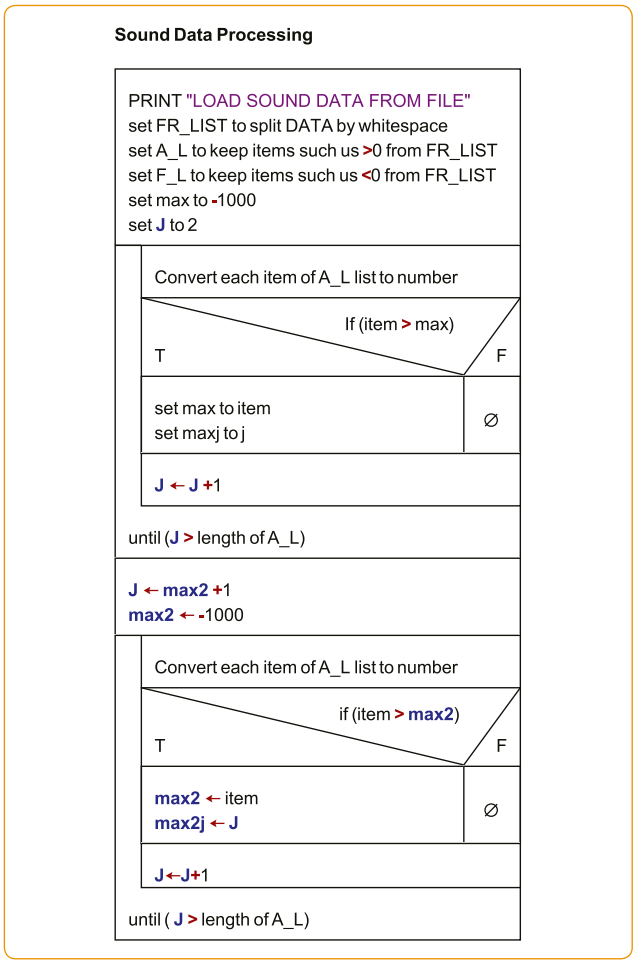


10: Exportierte Daten

Die Verarbeitung der Tonspektrumsdaten ist in 11–12 dargestellt. Detaillierte Informationen, z. B. über die verwendeten Variablen, sind online verfügbar. [4]



11: Der Dopplerverschiebungsteil des Experiments[4]



12: Nassi-Shneidermann-Diagramm für Tondatenverarbeitung

<Algorithmus für andere Sprachen>

Mit der Basisvorlage kann das Programm auf einfache Weise in eine andere Programmiersprache übertragen werden, da es eine Basisbibliothek für die Kommunikation mit dem Mikrocontroller gibt. Somit ist die Wahl des Mikrocontrollers für das Projekt nicht von Bedeutung.

<Fazit>

Ein preisgünstiges Projekt, das leicht aufzubauen und durchzuführen und hoffentlich für die Schülerinnen und Schüler interessant und anregend ist.

<Kooperationsmöglichkeiten>

Der Austausch von Unterrichtsideen und die Umsetzung innovativer Unterrichtsansätze ist der Hauptzweck von Science on Stage. Das gemeinsame Unterrichten mit Ilia Mestvirishvili und David Shapakidze, einem großartigen Partnerteam aus Georgien, könnte bezüglich Entfernung und Zeitplanung eine Herausforderung werden, war aber schon ein großer Gewinn für die Entwicklung neuer Techniken. Auch wenn uns Erfahrung im Unterrichten von Schülerinnen und Schülern mit besonderen Bedürfnissen fehlt, wäre es durchaus lohnend und begrüßenswert, dieses Projekt so anzupassen, dass es für alle Schülerinnen und Schüler zugänglich ist.

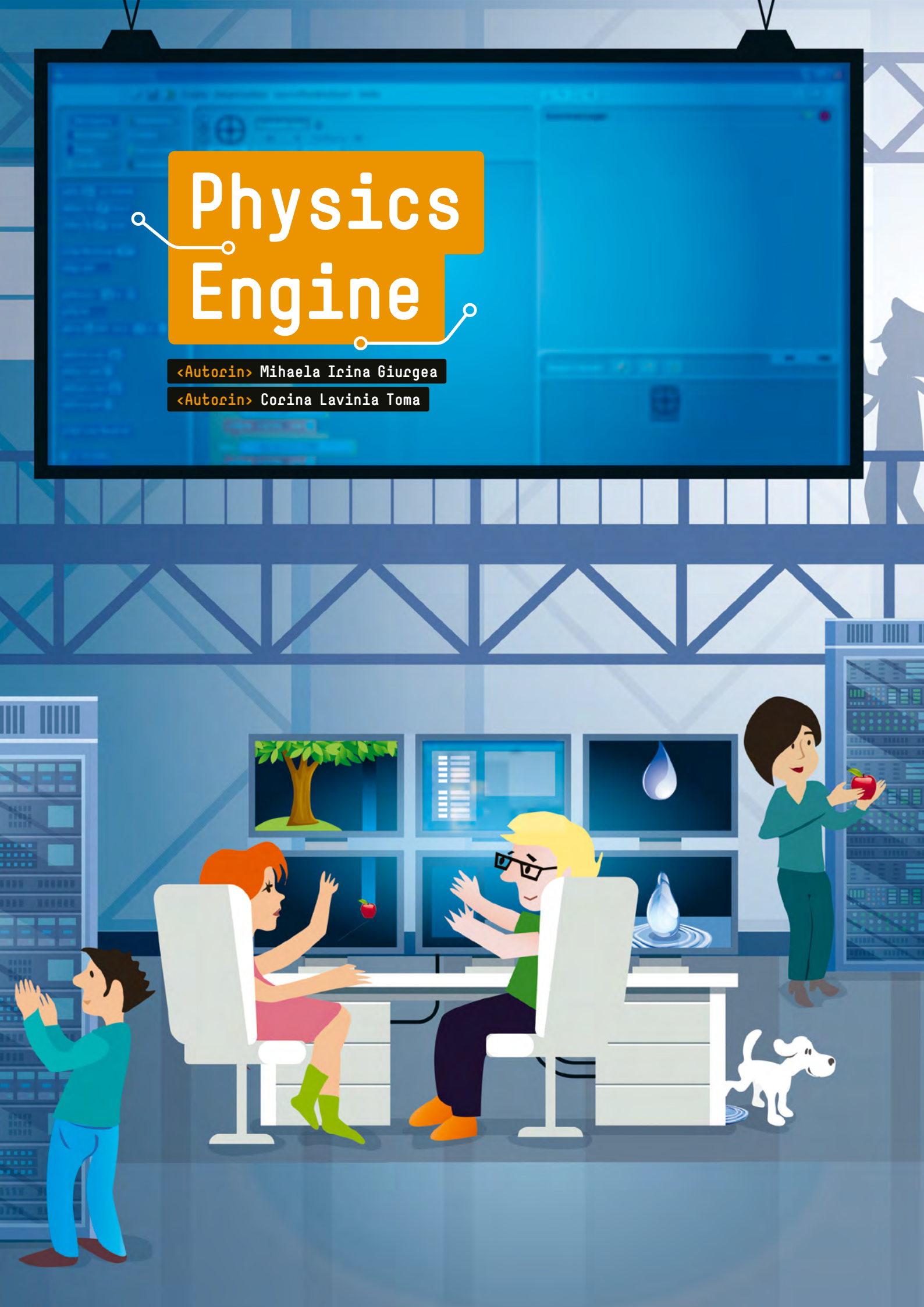
<Quellen und Hinweise>

[1] www.arduino.cc
[2] http://snap4arduino.rocks
[3] www.audacityteam.org
[4] Sämtliches Zusatzmaterial ist erhältlich auf www.science-on-stage.de/coding-materialien.
[5] https://snap.berkeley.edu

↳ www.physicsclassroom.com/mmedia/circmot/ucm.cfm
↳ https://education.pasco.com/epub/PhysicsNGSS/Booklnd-904.html
↳ http://hyperphysics.phy-astr.gsu.edu/hbase/Sound/dopp.html
↳ http://newton.phys.uaic.ro/data/pdf/Doppler_experiment.pdf
↳ https://manual.audacityteam.org/man/tutorials.html
(alle Dezember 2018)

Physics Engine

<Autorin> Mihaela Irina Giurgea
<Autorin> Corina Lavinia Toma



<Info>

<Schlagwörter> Animation, Sprite, Blöcke, Schleifen, Grafiken, Gravitation, elastischer Stoß, freier Fall, Reibungskraft, schräger Wurf, Bewegung, Impuls, Operatoren, Variablen

<Unterrichtsfächer> Informatik, Physik, Mathematik, IKT

<Altersgruppe> 14–16 Jahre

<Hardware> Computer

<Programmiersprache> Scratch[1]

<Programmierniveau> leicht, mittel

<Zusammenfassung>

Was würden Sie denken, wenn wir Ihnen sagen, dass Schülerinnen und Schüler die zwei scheinbar sehr unterschiedlichen Fächer Physik und Informatik leichter und dabei auch noch gleichzeitig erlernen könnten? In dieser Einheit ist die „Engine" (Motor) – die Scratch-Programmierumgebung[1] – das magische Werkzeug, das Schülerinnen und Schüler darin unterstützt, interessante Programme zu alltäglichen Naturphänomenen zu entwickeln, um die Folgen der physikalischen Gesetze zu verstehen und ihre Programmierkenntnisse zu verbessern.

<Vorstellung des Konzepts>

Warum verwenden wir Scratch[1]? Scratch ist eine visuelle Programmierumgebung, die Figuren („Sprites") mit Blöcken auf dem Computerbildschirm animiert und es ermöglicht, Programme einfacher als mit klassischen Umgebungen (C++, Java usw.) zu entwickeln. Zudem hilft unsere „Engine" den Lehrkräften zwei Fächer zu unterrichten, die als schwierig empfunden werden: Physik und Informatik. Indem wir die Haupthindernisse, d. h. die Unmöglichkeit, sich vorzustellen (zu sehen), wie ein Phänomen eigentlich funktioniert, und die komplizierte Programmiersyntax, beseitigten, schaffen wir eine angenehme und interessante Lernumgebung.

Da die Schülerinnen und Schüler, die an der Entwicklung dieser Unterrichtseinheit beteiligt waren, schon Anfängerkenntnisse in der Programmierung besaßen, fanden sie selbst heraus, wie sie mit Scratch arbeiten können, was sowohl im regulären als auch im freiwilligen Informatikunterricht stattfand. Die erforderlichen Physikkenntnisse sind Teil des regulären Lehrplans, und es ist immer hilfreich, Gelerntes zu wiederholen und anzuwenden.

<Praktische Umsetzung>

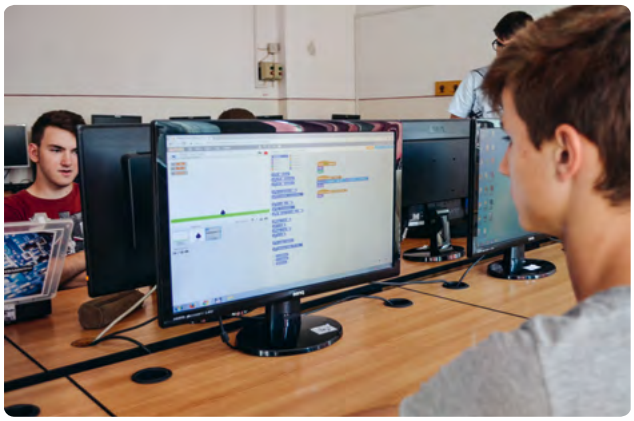
Die Einheit besteht aus alternierenden Lernsequenzen in Programmierung und Physik.

Zuerst präsentierte die Informatiklehrkraft die Grundlagen einer Projekterstellung in Scratch[1]. Die Schülerinnen und Schüler machten sich mit verschiedenen Schlüsselbegriffen der Scratch-Umgebung vertraut: Bühne („Stage"), Figuren („Sprites"), Kostüme („Costumes"), Bewegung („Movement"). Die Anleitungen und Erläuterungen zum ersten mit Scratch gelehrten Programm ohne physikalische Formel sind online verfügbar.[2]

Die Schülerinnen und Schüler müssen die Interaktionen zwischen den Sprites und ihre Synchronisation verstehen, aber auch, wie ein Koordinatensystem funktioniert. Ein komplettes Tutorial zu Scratch ist online verfügbar.[3] Für ein besseres Verständnis der Hauptalgorithmen von Scratch schauten sich die Schülerinnen und Schüler einige interessante Beispiele an und waren oft erstaunt, wie einfach die dahinterliegende Programmierung ist.

Im Physikteil wandten sie die theoretischen Grundlagen, auf denen die Phänomene ihrer Umwelt basieren, an. So schlug die Physiklehrkraft viele Themen[4] vor, die anschließend diskutiert wurden: erforderliche Formeln, mögliche Animationen, das Design usw. Dann wählten die Schülerinnen und Schüler Themen aus und nach einer Woche präsentierten sie Apps zum schrägen Wurf eines Balls, zum freien Fall eines Apfels, zum komplexeren Fall eines Wassertropfens oder zur Kollision zweier Bälle, aber auch zu Planetenbewegungen im Sonnensystem und sogar kleine Computerspiele.

Zu Beginn arbeiteten die Schülerinnen und Schüler weitgehend selbstständig. Wenn ein Projekt verbessert werden musste, erhielten sie Unterstützung von den Lehrkräften und dem Rest der Klasse. (📷 1)



📷 1: Selbstständiges Arbeiten

Danach stellten alle ihre Projekte vor. Dank der Rückmeldungen fiel es den Schülerinnen und Schülern leichter zu erkennen, welche Aspekte noch verbessert werden mussten: die Programmierung oder der Physikteil.

Am Ende unseres Projekts wurden die Schülerinnen und Schüler zu Lehrkräften für die Jüngeren (12–13 Jahre alt), indem sie geeignete Programme vorstellten und während des Physikunterrichtes testeten. Es machte ihnen Spaß, in der Rolle der Lehrkraft zu sein, und sie waren sehr stolz auf ihre Arbeit. Zudem erhielten sie von den Jüngeren Vorschläge und Anregungen. Die besten Simulationen der Schülerinnen und Schüler sind auf der Scratch-Plattform abrufbar.[2]

Nachfolgend präsentieren wir einige Beispiele zu Ansätzen wie Physik mit Programmieren verbunden werden kann.

<Programm 1: Freier Fall (Newtons Apfel)>

Vom freien Fall dieses historischen Objektes hat jeder schon einmal gehört: des Apfels des Isaac Newton. Das Programm in 📷 2 ist von einer klassischen Aufgabe inspiriert: Welche Distanz legt Newtons Apfel im freien Fall pro Sekunde zurück?

Physikalische Theorie

Annahme: Eine lineare Bewegung erfolgt mit der konstanten Erdbeschleunigung $g = 9{,}8\,\frac{\mathrm{m}}{\mathrm{s}^2}$.

Nach einer Zeit t beträgt die zurückgelegte Distanz $h(t)$ des Apfels: $h(t) = \frac{gt^2}{2}$.

Der Ausgangspunkt $h(0)$ befindet sich am Ast des Baumes, von dem sich der Apfel gelöst hat.

Dann berechnen wir die zurückgelegte Distanz über einen längeren Zeitraum $t + \Delta t$:

$$h(t+\Delta t) = \frac{g(t+\Delta t)^2}{2}.$$

Die allgemeine Formel für die Distanz $\Delta h(t)$, d. h. die Strecke, die der Apfel in der Zeit Δt zurücklegt, lautet:

$$\Delta h(t) = h(t+\Delta t) - h(t) = \frac{g(2t\Delta t + \Delta t^2)}{2}.$$

In unserem spezifischen Fall gilt: $\Delta t = 1$ s. In der ersten Sekunde folgt aus $t = t_{in} = 0$, dass $\Delta h_1 = 4{,}9$ m, in der nächsten Sekunde $t = 1$s führt zu $\Delta h_2 = 3 \cdot 4{,}9$ m $= 14{,}7$ m usw. Durch mathematische Induktion können wir die Berechnung für die *n-te* Sekunde und $t = (n-1)$ s vornehmen:

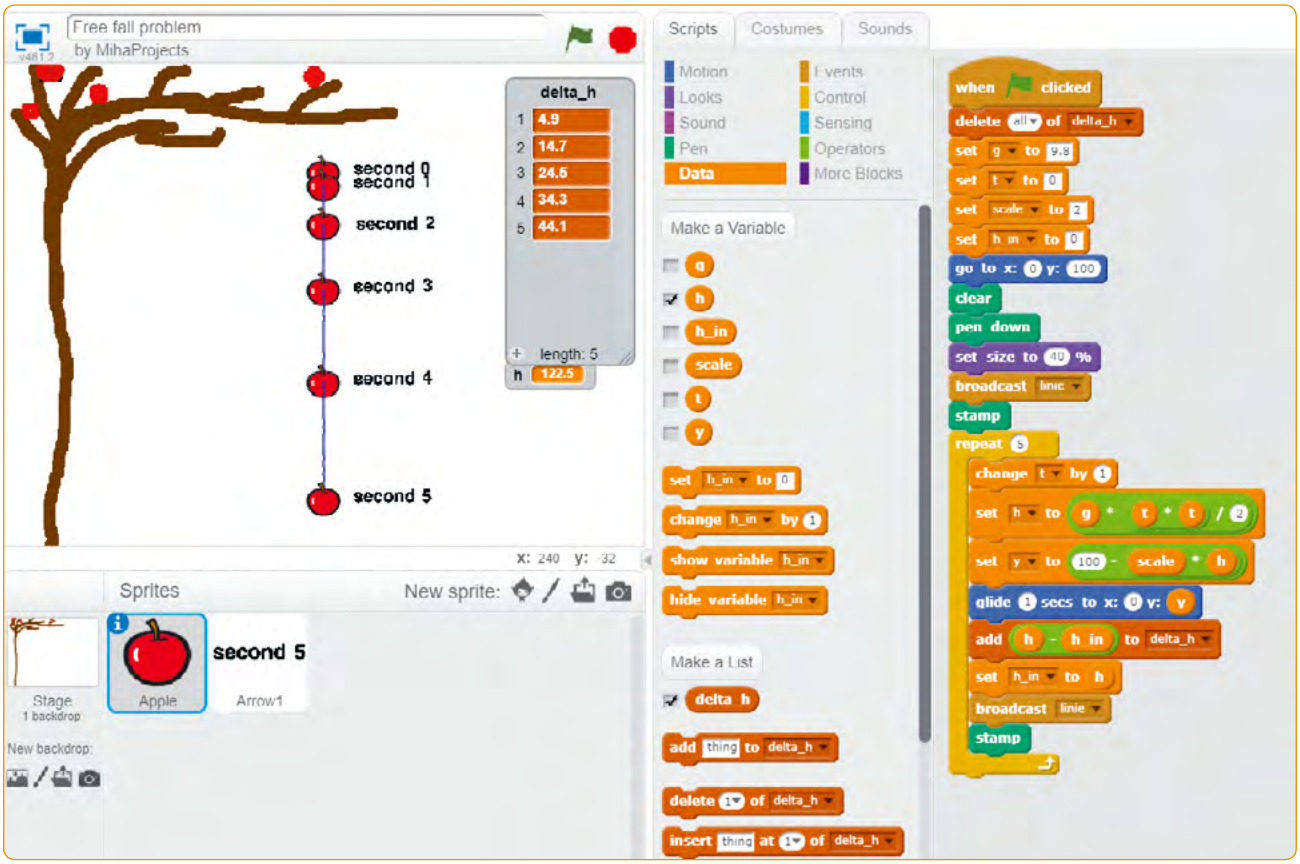
$$\Delta h_n = \frac{9{,}8(2n-1)}{2}\ \mathrm{m}.$$

Nun können die Schülerinnen und Schüler berechnen und in unserer Animation auch sofort sehen, dass sich die zurückgelegte Distanz in jeder Sekunde stets um den gleichen Betrag erhöht: 9,8 m.

Wie programmiert man das?

Verwendete Variablen:

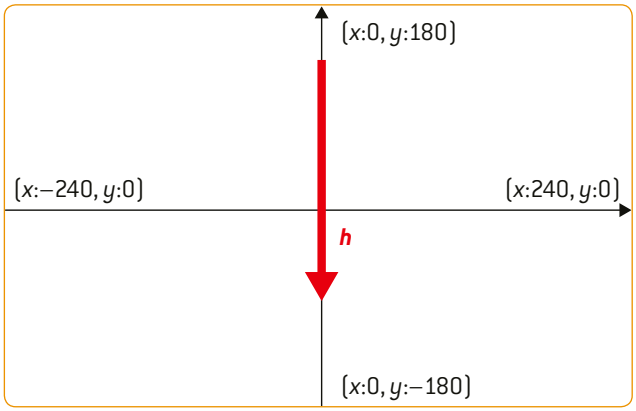
g: Erdbeschleunigung



📷 2: Freier Fall

t: Zähler für Sekunden (mit den Werten: 0, 1, 2, 3, 4, 5)
h: nach *t* Sekunden zurückgelegte Distanz
h_in: ursprüngliche Position des Apfels
delta_h: Liste (Feld) mit allen in jeder Sekunde zurückgelegten Distanzen
y: *y*-Koordinate des Apfels
Bemerkung: Die *x*-Koordinate bleibt konstant = 0, damit sich die Trajektorie im Bild leichter von links nach rechts bewegen lässt.

Zu Beginn befindet sich der Apfel am Ausgangspunkt mit den Koordinaten (0,180). In 3 sind der Ausgangspunkt und die Richtung für die zurückgelegte Distanz *h*(*t*) markiert.



3: Orientierung im Koordinatensystem

```
Free_fall

delta_h: array of real
g←9.8
t←0
h_in←0
scale←2
go to (0,100)
clear()
pen(down)
stamp()

while (t<5)
    t← t+ 1
    h← g * t * t / 2
    y←100-scale*h
    glide (0, y)
    add (delta_h, h-h_in)
    h_in ←h
    broadcast(linie)
    stamp()
```

4: Programmieralgorithmus für den freien Fall

Indem man eine Schleife fünfmal durchlaufen lässt, berechnet man die Distanz, die der Apfel nach jeder Sekunde zurückgelegt hat, und bestimmt so die neue *y*-Koordinate unter Berücksichtigung der Bildschirmcharakteristika (Programmieralgorithmus in 4)

Zusatzaufgabe

Die Schülerinnen und Schüler können Δt und die Zeit *t* modifizieren (der Apfelbaum müsste dann sehr groß sein, vielleicht wäre ein Hochhaus besser) oder die Aufgabe auf einen Planeten mit abweichender Fallbeschleunigung verlegen. Man könnte die Reibungskraft der Luft hinzufügen und eine variable Erdbeschleunigung annehmen, indem man den Apfel aus größerer Höhe aus einem Wetterballon fallen lässt.

<Programm 2: Fallender Wassertropfen>

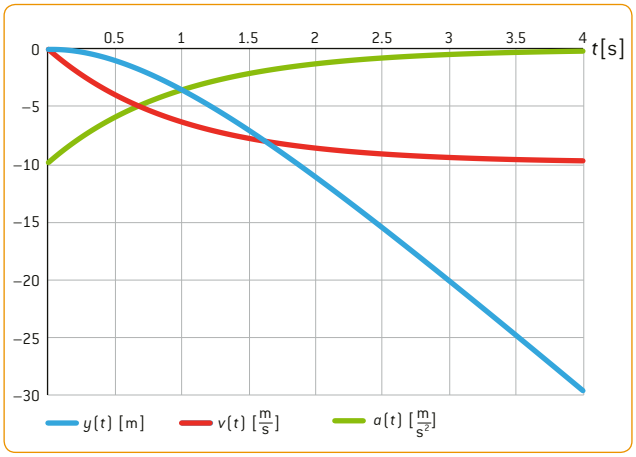
An Regentagen können wir das Fallen von Wassertropfen beobachten. Die Schülerinnen und Schüler analysierten mit einer Simulation die lineare Bewegung eines Tropfens. Sie sahen, dass sich der Fall des Wassertropfens anfangs beschleunigte, die Beschleunigung jedoch abnahm. Nach einer bestimmten Zeit erreichte die Geschwindigkeit des Tropfens ihre Grenze – die Endgeschwindigkeit v_t („terminal velocity") – wenn die Beschleunigung gleich null war. Dann fiel der Wassertropfen mit dieser konstanten Geschwindigkeit weiter. Wie erklärt man das?

Physikalische Theorie

Im beschleunigten Teil der Bewegung wirken zwei Kräfte in entgegengesetzter Richtung auf den Tropfen ein: die Gravitationskraft $G = mg$ (*m:* Masse des Tropfens, *g:* Erdbeschleunigung) und die Reibungskraft $F_f = kv$ (*k:* Proportionalitätskonstante, *v:* momentane Geschwindigkeit, Index f für „friction"). Die Beschleunigung des Tropfens wird $a = g - \frac{k}{m} v$.

In unserer Simulation nehmen wir einen großen Tropfen mit einem Durchmesser von ca. 5 mm mit einer Endgeschwindigkeit $v_t = 9{,}8\ \frac{\mathrm{m}}{\mathrm{s}}$ an.[5] Hier beträgt die Konstante $\frac{k}{m} = \frac{1}{\mathrm{s}}$. Die Beschleunigung nimmt ab, wenn die Geschwindigkeit zunimmt (5). Die Startwerte sind $a = 9{,}8\ \frac{\mathrm{m}}{\mathrm{s}^2}$, $v = 0$ und $y = 0$.

Um die momentane Beschleunigung und Geschwindigkeit zu berechnen, verwenden wir ein kleines Programm in C++[4], in

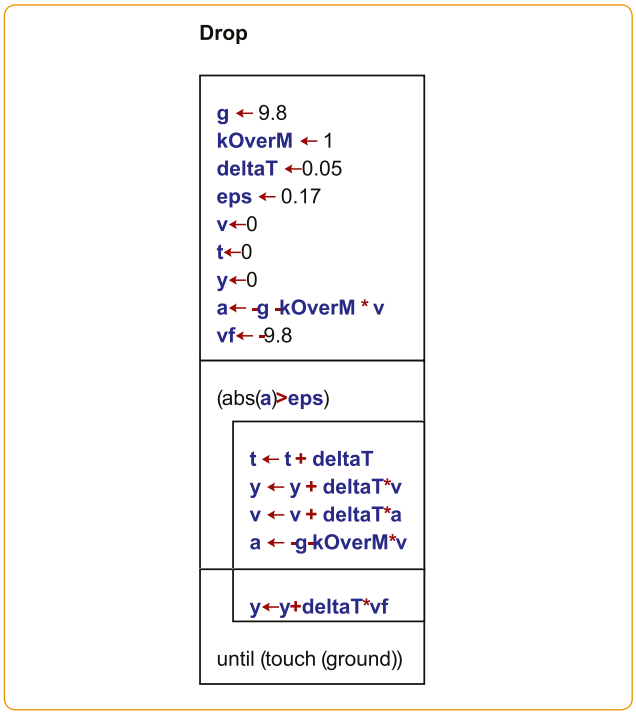


5: Zusammenhang zwischen zurückgelegter Distanz, Geschwindigkeit und Beschleunigung

dem wir die Beschleunigung und die Geschwindigkeitskonstante für sehr kleine Zeitintervalle Δt (z. B. 0,05 s) betrachten. In diesem Fall erhöht sich für jedes gewählte Δt die Geschwindigkeit mit $\Delta v = a\Delta t$, und die zurückgelegte Distanz mit $\Delta y = v\Delta t$ (schrittweise Methode).

Wie programmiert man das?

1. Zeichne ein Wassertropfen-Sprite.
2. Zeichne eine horizontale grüne Linie am unteren Rand des Hintergrunds.
3. Erstelle ein Sprite mit der Meldung „Beschleunigung=0“, welche erscheint, wenn die Beschleunigung etwa 0 ist.
4. Schreibe das Programm für das Tropfen-Sprite. Der Tropfen startet bei $(0, y_{init})$. Mit einer Schleife berechnen wir *a(t), v(t), y(t)* fortlaufend neu. Wir nehmen die zurückgelegte Distanz des Tropfens nach jedem Δt, erhalten so die neue y-Koordinate und berücksichtigen dabei die Bildschirmcharakteristika. Die Beschleunigung nimmt ab, und wenn sie bei ca. 0 liegt, endet die Schleife und es erscheint die Meldung auf dem Bildschirm. Als Nächstes fällt der Tropfen mit einer konstanten Geschwindigkeit, bis er die grüne Linie des Hintergrunds berührt. 📷 6 hilft, den Code zu verstehen.



📷 6: Fallender Wassertropfen

Zusatzaufgabe

Die Schülerinnen und Schüler können dieses Programm verbessern, indem sie eine Variable für die Masse des Wassertropfens (der Durchmesser des Wassertropfens beträgt üblicherweise 1 mm bis 5 mm)[6] und einen anderen Reibungskrafttyp hinzufügen: $F_f = \frac{kv^2}{2}$.

Außerdem können sie weitere Tropfen mit unterschiedlichen Massen erstellen und ihren Fall vergleichen.

<Programm 3: Elastischer Stoß>

Es gibt viele Beispiele für kollidierende Körper. Diese Kollisionen sind oft kompliziert, aber wir betrachten hier den elastischen Stoß, welcher sich auf den Zusammenstoß von Billard- oder Stahlkugeln anwenden lässt, bzw. auf die Theorie von Molekülkollisionen im Modell des idealen Gases.

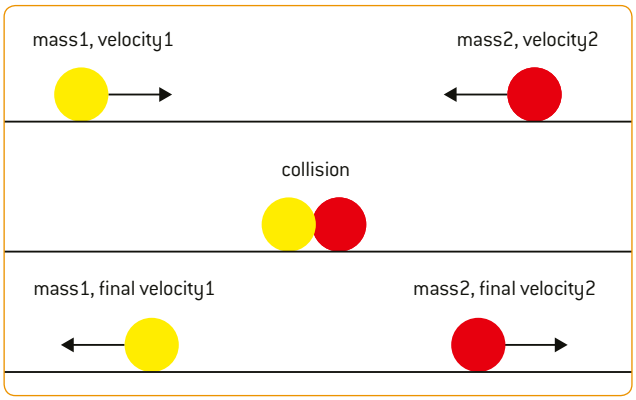
Physikalische Theorie

Wir betrachten die Erhaltungssätze für den linearen Impuls und die kinetische Energie für zwei Bälle mit den Massen m_1 und m_2, den Startgeschwindigkeiten $(\vec{v_1})$ und $(\vec{v_2})$ und den Endgeschwindigkeiten $(\vec{v_{1f}})$ und $(\vec{v_{2f}})$. (📷7)

$$m_1\vec{v_1} + m_2\vec{v_2} = m_1\vec{v_{1f}} + m_2\vec{v_{2f}} \text{ und}$$

$$\frac{1}{2}m_1v_1^2 + \frac{1}{2}m_2v_2^2 = \frac{1}{2}m_1v_{1f}^2 + \frac{1}{2}m_2v_{2f}^2$$

Wenn alle Bewegungen entlang derselben Achse (hier x-Achse) stattfinden, können wir zur Anzeige der Richtung + oder – verwenden. Die Vektornotation wird für diesen Fall nicht benötigt und die Endgeschwindigkeiten werden wie folgt berechnet: $v_{1f} = 2\frac{m_1v_1+m_2v_2}{m_1+m_2} - v_1$ und $v_{2f} = 2\frac{m_1v_1+m_2v_2}{m_1+m_2} - v_2$.

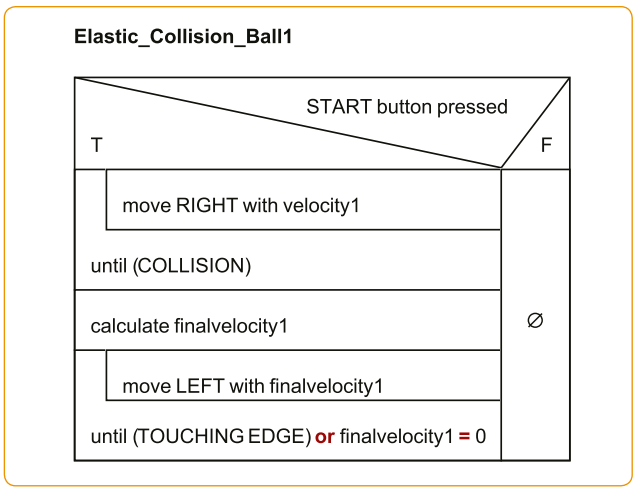


📷 7: Elastischer Stoß

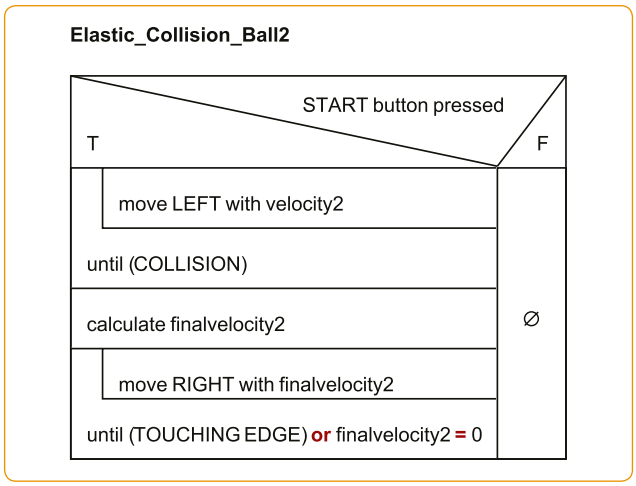
Wie programmiert man das?

1. Wähle zwei Sprites für die Bälle (Ball1 und Ball2) und ein Sprite für den START-Knopf (Start-Sprite).
2. Verwende die Variablen: *mass1, mass2, velocity1, velocity2* (die Masse und die Startgeschwindigkeit) für jedes Objekt. Mache die variablen Schieberegler sichtbar und bestimme ihre Minimal- und Maximalwerte.
3. Gib Masse und Startgeschwindigkeit für jedes Objekt ein.
4. Drücke den START-Knopf. In diesem Moment sendet das Start-Sprite eine Meldung an die Ball-Sprites. Sobald sie die Meldung erhalten, bewegen sich die Bälle mit *Distanz = Geschwindigkeit · Zeit* aufeinander zu.

5. Berechne die Endgeschwindigkeiten der Bälle und verwende sie, um diese in die richtige Richtung zu bewegen, wobei sich jeder Ball solange bewegt, bis er entweder den Rand berührt und das Bild verlässt oder stehen bleibt, weil seine neue Geschwindigkeit gleich 0 ist.
   8 und 9 zeigen, wie die Bälle in Scratch[1] animiert werden.



8: Elastischer Stoß für Ball1



9: Elastischer Stoß für Ball2

Zwei Anwendungsbeispiele dieses Programms:

1. Wähle eine Geschwindigkeit 0 und gleiche Massen für die Bälle; nach der Kollision stoppt der sich bewegende Ball und der andere bewegt sich mit der Geschwindigkeit, die der erste vor dem Aufprall hatte.
2. Die Bälle haben unterschiedliche Geschwindigkeiten und gleiche Massen; durch die Kollision vertauschen sich die Geschwindigkeiten der Bälle.

In beiden Beispielen vertauschen sich die Impulse der Bälle.

Zusatzaufgabe

Die Schülerinnen und Schüler verändern die Größe der Bälle direkt proportional zu ihrer Masse. Außerdem können sie ein Programm für einen zweidimensionalen elastischen Stoß entwickeln (Simulation des Compton-Effekts) oder für die Kollision eines Balls mit einer Wand (mechanisches Reflexionsgesetz). Sie können ihre Untersuchungen mit einem weiteren Programm zum unelastischen Stoß fortsetzen.[4]

<Fazit>

<Für die Schülerinnen und Schüler>

Vorteile

Die Schülerinnen und Schüler erlernten physikalische Konzepte auf eine unterhaltsamere Weise und konnten Phänomene durch die Simulationen in Scratch besser verstehen, wobei sie zugleich ihre Kenntnisse in Informatik und Physik praktisch anwandten. Auch wenn nicht alle Projekte perfekt waren, verbesserten die Schülerinnen und Schüler ihre Programmierkenntnisse und ihr algorithmisches Denken.

Nachteile

Alle arbeiteten alleine und größtenteils zu Hause. In der Schule erhielten sie dann Rückmeldungen.

<Für die Lehrkräfte>

Vorteile

Wir beobachteten ein echtes Interesse an der Entwicklung eines eigenen Programms und ein besseres Lernergebnis als im klassischen Unterricht.

Nachteile

Wegen der verschiedenen physikalischen Themen und der sehr spezifischen Bugs in den einzelnen Programmen war es für uns schwierig, die gesamte Klasse zu koordinieren. Wir sind der Ansicht, dass es besser wäre, allen jeweils nur ein Thema zu geben und dieses dann je nach den Fähigkeiten der Schülerinnen und Schüler zu verbessern und zu vertiefen.

<Kooperationsmöglichkeiten>

Klassen aus verschiedenen Schulen und Ländern können die Aufgaben des Projekts lösen und neue Aufgaben mit anderen, auf das ursprüngliche Thema bezogenen Ideen entwickeln. Alle diese Programme können auf die Scratch-Plattform hochgeladen werden, worauf ein Wettbewerb unter den besten Einsendungen durchgeführt werden kann. Bei der Bewertung der Arbeit müssen die Lehrkräfte die Komplexität der Programmierung und der physikalischen Themen mitberücksichtigen.

<Quellen und Hinweise>

[1] https://scratch.mit.edu
[2] Alle zusätzlichen Materialien sind online erhältlich unter www.science-on-stage.de/coding-materialien.
[3] https://scratch-dach.info
[4] https://scratch.mit.edu/users/SonS_Coding
[5] http://hypertextbook.com/facts/2007/EvanKaplan.shtml
[6] https://journals.ametsoc.org/doi/pdf/10.1175/1520-0450%281969%29008%3C0249%3ATVORA%3E2.0.CO%3B2

SMB - Science Magic Box

<Autor> Luc Ivarra
<Autor> Marco Nicolini

<Info>

<Schlagwörter> Mikrocontroller, Wandler, Sensor, Aktor, Signal, physikalische Größe, Schleife, Verzweigung, sequenziell, Verarbeitung, Kalibrierung, Input, Output, lesen, schreiben, analog, digital, Linearität, Konversion, Steckplatine, Pin, Löten, Mensch-Maschine-Schnittstelle

<Unterrichtsfächer> Physik, Elektronik, Mathematik, IKT, Logik, Biologie

<Altersgruppe> 14–18 Jahre

<Hardware> Arduino UNO[1] mit Arduino DUE[2] und/oder TI-Nspire CX CAS mit TI-Innovator Hub

<Programmiersprache> C++ (Verwendung der Arduino IDE[3]) und/oder TI-Basic

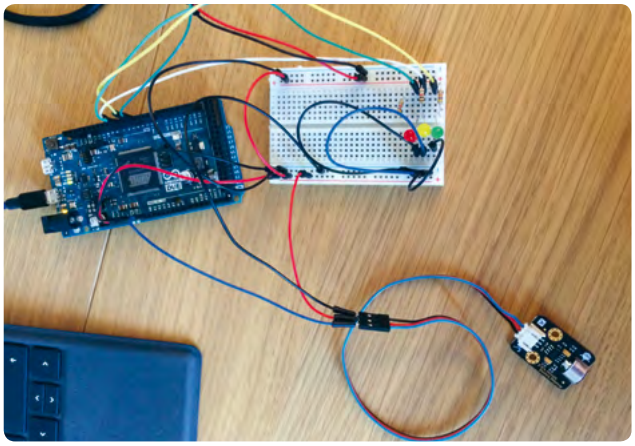
<Programmierniveau> mittel, mit einem Audioteil für fortgeschrittene Schülerinnen und Schüler

Eine Liste der verwendeten technischen Begriffe, Abkürzungen und Akronyme ist online verfügbar.[4]

<Zusammenfassung>

Die Schülerinnen und Schüler programmieren eine selbstkonstruierte Hardware-Software-Umgebung (basierend auf Arduino) und einen betriebsbereiten Taschencomputer (TI-Nspire CX CAS mit der Erweiterung TI-Innovator Hub). Beide Geräte werden für die Erfassung, Umwandlung und Übertragung von Sensordaten verwendet, sodass physikalische Größen leicht gehandhabt, gelesen, konvertiert und bedient werden können.

Die Schülerinnen und Schüler schreiben für Arduino ein Programm zu einer Anordnung mehrerer Sensoren, die physikalische Größen als Eingangssignale erfassen, und Aktoren, die auf die Datenerfassung reagieren. Letztere geben ein Ausgangssignal als physikalische Größe aus, nachdem ein Mikrocontroller das festgestellte Signal verarbeitet hat, sodass der korrekte Output festgelegt wird. (Siehe 1)



1: Arduino-Platine

TI Innovator Hub ist eine betriebsbereite Box, mit der die Schülerinnen und Schüler Programmiergrundlagen erlernen. Sie muss an den TI-Nspire CX CAS Taschenrechner angeschlossen werden. Die Box hat eine gute I/O-Schnittstelle, inklusive Helligkeitssensor, zwei LEDs und eingebautem Summer, der einen Ton in einer festgelegten Frequenz erzeugt. (Siehe 2)

2: TI-Nspire und TI-Innovator Hub

<Vorstellung des Konzepts>

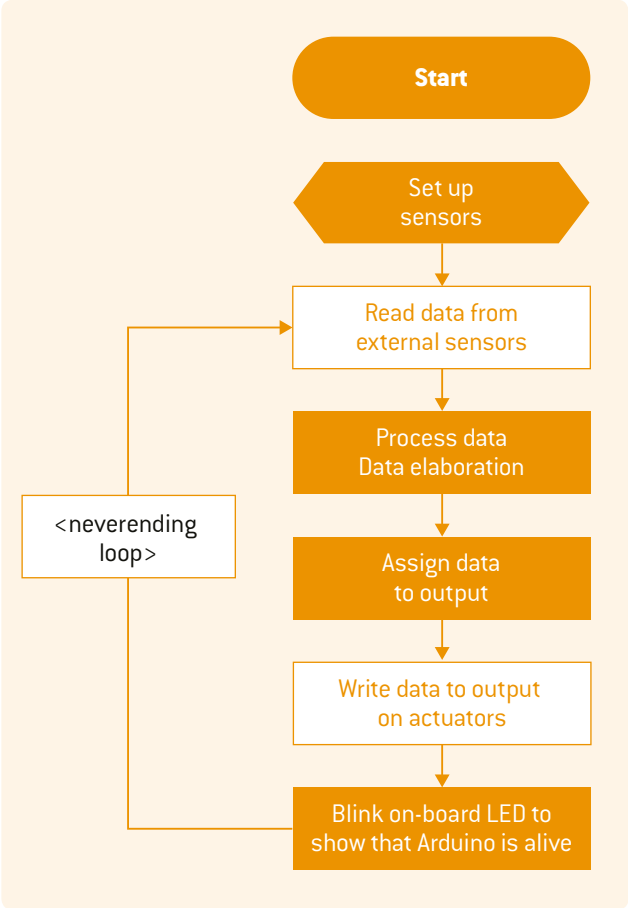
Die Einheit führt die Schülerinnen und Schüler in die Programmierung zu physikalischen Aufgabenstellungen ein, basierend auf der Messung physikalischer Größen, der Verarbeitung von Daten und der Entscheidung, welche Maßnahme unter Verwendung der Aktoren getroffen werden soll.

Das Programm besteht üblicherweise aus einer unendlichen Schleife (meist „lebt“ die Maschine, solange sie mit Energie versorgt wird, und sie muss die ganze Zeit arbeiten), in der die drei Aktionen Messen, Verarbeiten und Handeln in dieser Reihenfolge ausgeführt werden.

Die Schülerinnen und Schüler lernen, dass sie Programme schreiben können mit

1. sequenziellen Anweisungen
2. Schleifen (während … tue dies; wiederhole … bis)
3. Verzweigungen (wenn … dann … sonst),

wie aus dem Böhm-Jacopini-Theorem hervorgeht (siehe „Zusatzinformationen“[4]).



3: Flussdiagramm

Das zweite Ziel der Einheit ist die Einführung in das Thema „Mikrocontroller". Die Schülerinnen und Schüler lernen, unter Verwendung von digitalen und analogen Ports, Sensoren und Output-Aktoren zu installieren und ein einfaches Programm zum Lesen des Inputs, Verarbeiten der Daten und Schreiben des Outputs zu erstellen. Als Output wählten wir entweder einen Ton oder Lichtsignale. Diese können als „Alarm" interpretiert werden, also als eine Warnung, die auf Basis eines von den Sensoren gelesenen Inputs ausgeben wird.

Die Schülerinnen und Schüler verwenden die integrierte Entwicklungsumgebung Arduino IDE[3], um in C++ zu programmieren. Es gibt hier betriebsbereite Funktionsbibliotheken, die das Programmieren erleichtern und beschleunigen.

Eine Steckplatine (siehe „Zusatzinformationen"[4]) muss vorhanden sein, damit die Schülerinnen und Schüler die Anordnung aufbauen und die Sensoren-Pins einfach an die Arduino I/Os, die 5V-Stromquelle und den GND-Pin anschließen können.

Die Struktur eines Programms ist in Metasprache in 3 dargestellt.

Hinweis: Die Metaanweisung *„While (TRUE)"* („Während wahr") ist ein Trick, der jeden Prozessor dazu veranlasst, die darin enthaltenen Anweisungen unendlich oft (solange der Mikrocontroller über Energie verfügt) zu wiederholen.

Das dritte Ziel ist es, zu erlernen, wie eine erfasste physikalische Größe in eine andere umgewandelt (z. B. Lichtintensität in Schall) und nach außen übertragen werden kann. Dies geschieht wie folgt:

1. Das physikalische Signal (z. B. Licht, Schall, Kraft, Energie) wird durch den Sensor erfasst und in ein elektrisches Signal umgewandelt.
2. Das elektrische Signal wird in eine Zahl umgewandelt, die dem Prozessor zur Verfügung steht.
3. Die Zahl wird verarbeitet und vom Prozessor in eine andere Zahl umgewandelt und löst dann eine Aktion durch einen Aktorwandler aus.
4. Die Aktoren wandeln die Zahl in elektrische Signale um, die so als Output zur Verfügung stehen.
5. Das elektrische Signal wird schließlich in ein physikalisches Signal (z. B. Schall, Licht) umgewandelt.

In 1. und 5. müssen die Signale von einer Form in eine andere umgewandelt werden, wobei hier die Linearität der Umwandlung bzw. die „Nahezu-Linearität" extrem wichtig ist (siehe „Zusatzinformation"[4]).

Bezüglich der letzten Zeile der Metaprogrammierung (*„Blink on-board LED"*) siehe „Zusatzinformationen"[4] für eine detaillierte Erläuterung.

Üblicherweise wird das Inputsignal als „Stimulus" bezeichnet und kommt aus der Umgebung, in der die Sensoren zur Datenerfassung platziert wurden. Der Prozessor und das Programm sind so gestaltet, dass eine „Reaktion" mit mathematischen/logischen Operationen (von den Programmanweisungen durchgeführt und vom Mikrocontroller verarbeitet) auf den Stimulus erfolgt und dass diese „Antwort" als Output an die Umgebung abgegeben wird. In unserem Projekt ist die Umgebung der Raum um den Arduino, der dank Sensoren „sehen", „hören" und „Kräfte verspüren" kann.

Durch die Arbeit mit dem TI-Innovator Hub können sich die Schülerinnen und Schüler eher auf den Programmierteil konzentrieren, da die Mikrocontroller und Sensoren schon aufgebaut und betriebsbereit sind.

<Praktische Umsetzung>

Wir empfehlen mit einem Brainstorming zu beginnen, wobei die Vorstellungen der Schülerinnen und Schüler über Sensoren und automatische Maschinensteuerung diskutiert werden.

Diese Vorstellungen zu sammeln, praktische Erfahrungen zu machen und die Ergebnisse dann mit den ursprünglichen (vielleicht falschen) Annahmen zu vergleichen, hilft den Schülerinnen und Schülern das Ganze zu verstehen.

Dazu können Sie vorab einen Fragebogen vorbereiten:
↳ Weißt du, wie ein Thermostat die Raumtemperatur steuert?
↳ Wozu dient ein Einparksystem mit Tonsignalen? Wie reagiert der Fahrende, wenn das System einen Warnton abgibt?
↳ Habt ihr zu Hause einen Induktionsherd? Was signalisiert eine leuchtende LED?

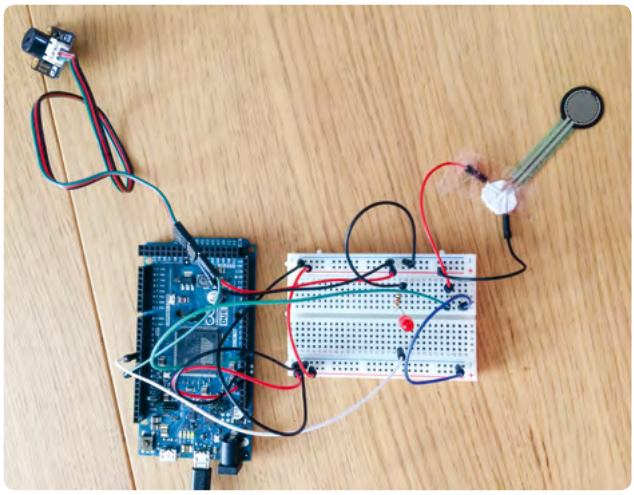
Siehe „PEC und Fragenliste“[4] für ein Beispiel einer Frageliste und die Verweise auf die „PEC“-Lehrmethode (Prevision, Experience, Correction).

<Theoretische Phase mit Arduino[1]>

Die Lehrkraft führt die Schülerinnen und Schüler in die C++-Programmierung[3] mit Struktur und Basisanweisungen ein, sodass sie eine einfache Schleife mit den Anweisungen *analogRead, digitalRead, analogWrite, digitalWrite, if…then…else, loop, while* schreiben können.

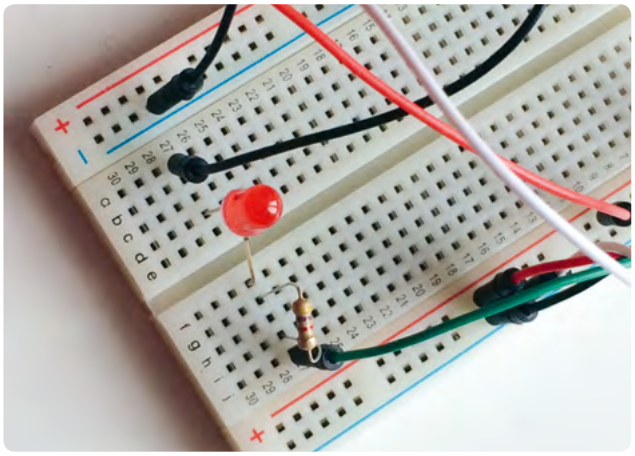
Hardware-Teil

Die Lehrkraft stellt die Anordnung des Mikrocontrollers vor und zeigt dabei den Mikroprozessor, die analogen I/O-Pins (Verbindungen) und die digitalen Input-/Output-Pins (Verbindungen).



4: Arduino, Steckplatine und Sensoren

Die Schülerinnen und Schüler lernen, dass jeder Sensor/Aktor meist mehrere Verbindungen aufweist:
↳ zum 5V- oder 3,3V-Arduino-Output für die Stromzufuhr
↳ zum GND-Signal (Erdung), damit der Strom fließt
und
↳ zu einem weiteren digitalen oder analogen Input-Pin, wenn Daten von außen gelesen (erfasst) werden
oder
↳ zu einem weiteren digitalen oder analogen Output-Pin, wenn eine Aktion nach außen erfolgen soll, z. B. Licht oder einen Ton ausgeben oder etwas Anderes, das eine Situation anzeigt (eine „Write“-Operation)



5: Steckplatine im Detail

Software-Teil

Ein einfaches Programm, das einen Sensor liest und einen Aktor schreibt, wird von der Lehrkraft vorgestellt. Dabei sollen die Schülerinnen und Schüler eine klare Assoziation zwischen physischen Pins und physischer Onboard-Adresse des Mikrocontrollers herstellen können.

Ein Beispiel für betriebsbereite Anweisungen ist online verfügbar („Programmbeispiel 1“[4]).

Die Schülerinnen und Schüler müssen dabei berücksichtigen, dass der Prozessor die Programmanweisungen eine nach der anderen und in der geschriebenen Reihenfolge ausführt. Nur die Schleife weicht von diesem Grundsatz ab, da sie den Prozessor veranlasst, die in der Schleife enthaltenen Anweisungen zu wiederholen, solange der Mikrocontroller über Strom verfügt.

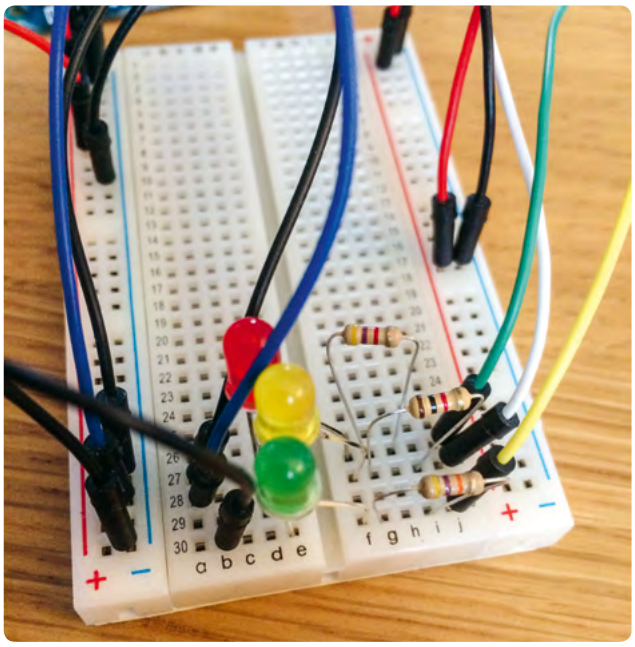
<Praktische Phase mit Arduino>

Die Schülerinnen und Schüler machen praktische Erfahrungen mit dem Mikrocontroller, der Steckplatine und den Sensoren. Die Lehrkraft sollte die Struktur der Steckplatine vorstellen und dabei alle verfügbaren Verbindungen aufzeigen und erläutern, wie die 5V-, die GND- und die I/O-Signale vom Arduino[1] zur Steckplatine gebracht werden können. Die Schülerinnen und Schüler werden dann aufgefordert, das Programmierbeispiel zu kopieren und das Programm mit den verbundenen I/Os auszuprobieren, zu testen und wenn nötig zu korrigieren („Debugging“).

Hardware-Teil

Die Schülerinnen und Schüler benötigen den Mikrocontroller, die Sensoren und kurze Kabel (10 cm), um Verbindungen zwischen den Sensoren-Pins und den Federkontakten der

Steckplatine herzustellen. Manchmal ist es notwendig, zusätzliche Kabel an die Sensoren zu löten.



6: Steckplatine mit LEDs

Mithilfe der kurzen Kabel werden die 5V-Stromquelle und das GND-Signal an die Steckplatine angeschlossen sowie die analogen/digitalen Pins des Arduino[1] an einige Federkontakte der Steckplatine. Dies ermöglicht es, die Sensoren an der Steckplatine zu befestigen und die erforderlichen elektrischen Signale zu erhalten. (Siehe 6)

Software-Teil

Zuerst überprüfen die Schülerinnen und Schüler, ob die Verbindungen zwischen Mikrocontroller und Steckplatine korrekt funktionieren, indem sie auf die genaue Übereinstimmung zwischen der logischen Nummer des Pins am Mikrocontroller und dem Sensor-Pin auf der Steckplatine achten. Anschließend versuchen sie, ein einfaches Programm zu schreiben, welches online verfügbar ist („Programmierbeispiel 1“[4]).

Algorithmen mit Arduino

Wir haben verschiedene Signalumwandlungen von einer physikalischen Form in eine andere vorbereitet.

7: Lichtsensor

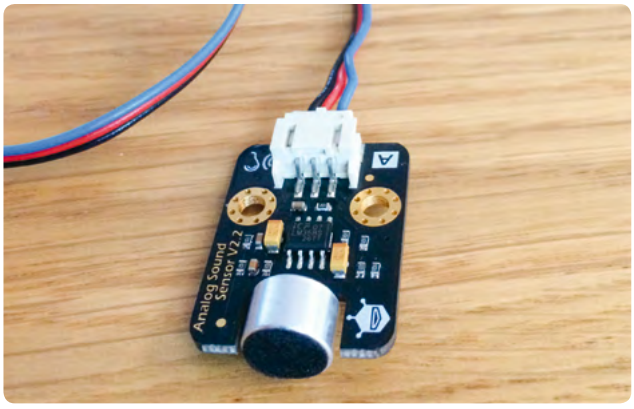
Umwandlung eines analogen Lichtsignals in ein digitales Lichtsignal in einer LED (moduliert mit einem PWM-Feature) und in einen Ton: Die ausgegebene Tonfrequenz erhöht sich mit der Lichtintensität. Praktische Anwendung: Wecker, Hilfe für Sehbehinderte. (Siehe 7)

Umwandlung einer durch den Kraftsensor festgestellten Kraft in ein digitales Lichtsignal in einer LED (moduliert mit einem PWM-Feature) und in einen Ton: Die Lichtintensität erhöht sich mit der Kraft. Praktische Anwendung: Überlastungsalarm (Siehe 8)

8: Digitaler Summer

Umwandlung eines externen Geräuschsignals in ein digitales Lichtsignal in einer LED. Je höher der Ton, desto höher ist die Lichtfrequenz der eingeschalteten LED. Praktische Anwendung: Lärmbelastungskontrolle.

Umwandlung einer von einem Distanzsensor gemessenen Distanz in einen Ton. Technische Anwendung: Einparksensoren. (Siehe 9)



9: Distanzsensor

Umwandlung eines Temperatursignals in einen Ton und ein Lichtsignal. Technische Anwendung: Ofentemperaturkontrolle.

Umwandlung einer Bodenfeuchtigkeitskonzentration in ein Lichtsignal. Technische Anwendung: Pflanzenbewässerungs- und Gieß-Alarm und -Kontrolle. (Siehe 10)

10: Bodenfeuchtigkeitssensor

Bezüglich des Programms für diese Anwendungen auf der Arduino-Platine siehe „Programmierbeispiel 2“[4].

Alle diese Algorithmen sind Signalkontroll- und Warnsysteme, die eine gewählte Umgebung bezüglich einer physikalischen Größe überwachen und eine Warnung abhängig vom Input (Stimulus) ausgeben.

<Theoretische Phase mit TI-Innovator Hub>

Hardware-Teil

Die Lehrkraft kann demonstrieren, wie intuitiv und einfach es ist, die Sensoren zu verbinden.

Software-Teil

Für die Basisprogrammierung wird nur der Taschenrechner benötigt. Die Schülerinnen und Schüler müssen lediglich die oben erwähnten TI-Basic-Anweisungen beherrschen. Dann kann der Hub verbunden werden, und sie lernen, wie sie damit kommunizieren und insbesondere wie sie die Anweisungen „Read“ und „Get“ für die Datenerfassung und „Set“ für die Kontrolle der Outputs einsetzen.

<Praktische Phase mit TI-Innovator Hub>

Hardware- und Software-Teil

Die Schülerinnen und Schüler beginnen mit Beispielen zu den Grundlagen, um sich mit dem Hub vertraut zu machen, bevor sie sich an offenere Aufgabenstellungen wagen. Anhand kleiner Übungen eignen sie sich an, wie sie die verschiedenen Outputs steuern, z. B. Steuerung der LED-Farbe, LED blinken lassen, Dauer des Blinkens steuern und Töne mit einer gewissen Frequenz erklingen lassen. Die unendliche Schleife ist auch hier die Basisstruktur, um Operationen unendlich ausführen zu lassen.

Algorithmen mit TI-Innovator Hub

Die Schülerinnen und Schüler lösten zwei Aufgaben: Programmierung eines automatischen Schalters, der die Beleuchtung erst dann einschaltet, wenn die Intensität des Umgebungslichts unterhalb einer gewissen Schwelle fällt, und Programmierung eines Weckers, der einen Ton in einer Frequenz ausgibt, die mit der Erhöhung der Umgebungslichtintensität ansteigt. Weitere Entwicklungen sind möglich, wozu jedoch zusätzliche Sensoren gekauft und mit dem Hub verbunden werden müssten.

<Kauf der Sensoren>

Hinweise zum Kauf der Sensoren sind online verfügbar.[4]

<Fazit>

Am Ende dieses Projekts bemerkten wir bei unseren Schülerinnen und Schülern eine deutliche Verbesserung des Verständnisses von Programmierung und allgemeiner Programmstruktur sowie von Logik und Algorithmen.

<Kooperationsmöglichkeiten>

Eine wundervolle Kooperationsmöglichkeit wäre ein „Selbstunternehmer“-Projekt, bei dem die Schülerinnen und Schüler versuchen könnten, eine eigene Mensch-Maschine-Schnittstelle (human machine interface, HMI) mit einer gewissen technischen Nützlichkeit zu erfinden. Diese HMI sollte einen Stimulus der Umgebung lesen (Atmosphäre, Wohnung, menschlicher Körper usw.) und reagieren, indem sie ein weiteres Signal ausgibt, das eine Warnung ertönen lässt, eine Aktion durchführt oder eine Situation anzeigt. Partnerschulen im Ausland können eine Marktumfrage durchführen, um zu verstehen, wie die Marktnachfrage und der Marktwert sind, die das Gerät in ihren Ländern haben könnte. Jede Schule, die an einem solchen Austausch teilnimmt, entwickelt ein Gerät und führt Marktumfragen für ein Produkt durch, das von den anderen Schulen entwickelt wird. Am Ende des Projekts könnte das vom Markt am meisten nachgefragte Gerät von einem Partnerunternehmen im kleinen Rahmen produziert und verkauft werden. Auf der ganzen Welt hat Selbstunternehmertum einen hohen Stellenwert als Chance, Themen aus Wissenschaft, Technologie und Wirtschaft im Zusammenhang verständlich zu machen.

<Quellen und Hinweise>

[1] www.arduino.cc
[2] www.arduino.cc/en/Guide/ArduinoDue
[3] www.arduino.cc/en/Main/Software
[4] Sämtliches Zusatzmaterial ist erhältlich auf www.science-on-stage.de/coding-materialien.

CoALA - Code a Little Animal

<Autor> Mirek Hančl
<Autorin> Julia Winckler

<Info>

<Schlagwörter> Simulation, IPO model (input-processing-output – Eingabe, Verarbeitung, Ausgabe), Messungen, Computational Thinking, Making

<Unterrichtsfächer> Naturwissenschaften, Biologie, Informatik

<Altersgruppe> 9–13 Jahre

<Hardware> Calliope mini[1] oder BBC micro:bit[2]
<Werkstatt A> Krokodilklemmen, roter Bastelkunststoff, USB-Kabel und Batterie für den Calliope mini, selbstklebendes Kupferband (5 mm), Pappe, Klebstoff, Schere, kleines Wasserglas, Poster mit Tierbildern
<Werkstatt B> Krokodilklemmen, USB-Kabel und Batterie für Calliope mini, Feuchtigkeitssensor (Grove Moisture Sensor), Touchsensor mit vier Fühlern (Grove I2C Touch Sensor), Grove NFC, Grove I2C hub [3], Pappe, roter Bastelkunststoff, kleines Wasserglas, Poster mit Tierbildern

<Programmiersprache> MakeCode[4]

<Programmierniveau> leicht

<Zusammenfassung>

Welches Kind wünscht sich kein Haustier? Um herauszufinden, welches das richtige ist, bauen Schülerinnen und Schüler einen Simulator, der mithilfe eines Einplatinencomputers und externer Sensoren die Bedürfnisse eines Haustiers nachahmt.

<Vorstellung des Konzepts>

Das Thema „Haustiere“ steht nicht nur in der Grundschule auf dem Lehrplan – auch an weiterführenden Schulen wird im Biologieunterricht besprochen, wie der Wolf zum Hund wurde, welche Grundbedürfnisse ein Haustier hat und welche Anforderungen damit an die Besitzerin oder den Besitzer gestellt werden. Oft werden dafür Texte im Schulbuch oder Lehrfilme rezipiert, denn ein echtes Tier kann für den Unterricht nicht extra angeschafft werden. Daher ist ein elektronischer Simulator für die Grundbedürfnisse eines Haustiers – Essen, Trinken, Bewegung, Streicheleinheiten und richtige Körpertemperatur – anschaulich und lehrreich zugleich.

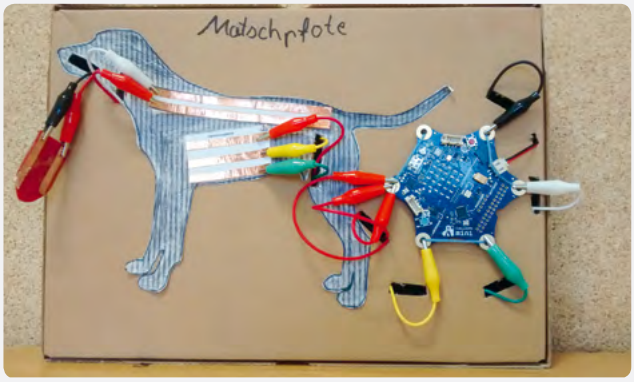
Für die Simulation wird im Projekt weder ein fertiges Gerät eines kommerziellen Lehrmittelproduzenten, das nur vorgegebene Programme erlaubt, verwendet, noch ein simples Spielzeug eingesetzt, wie das in den 90ern weltweit erfolgreiche Tamagotchi. Stattdessen konstruieren, bauen und programmieren die Schülerinnen und Schüler mithilfe eines Einplatinencomputers, hier Calliope mini[1] oder BBC micro:bit[2], und Bastelmaterial wie Pappe, Kupferklebeband sowie externen Sensoren ihren eigenen Simulator in Gestalt ihres Lieblingstiers, inklusive Bild! Dessen Grundbedürfnisse werden mit den Sensoren erfasst und durch einen selbstprogrammierten Algorithmus ausgewertet. Je nach Ablauf des Algorithmus zeigt der Tiersimulator mit entsprechenden Smileys an, wie er sich momentan fühlt, oder spielt selbstkomponierte Melodien ab.

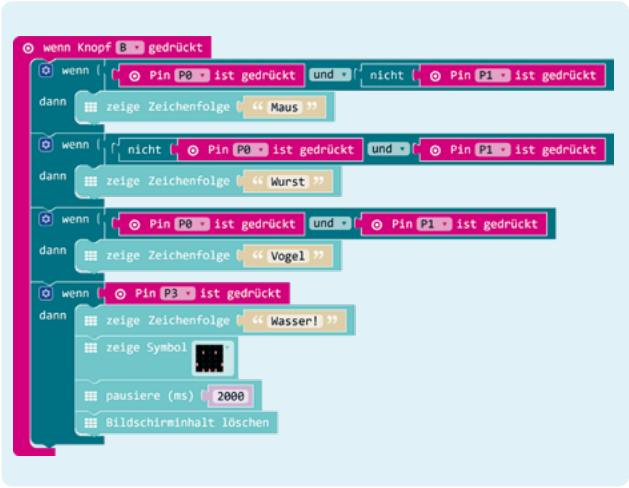
Das Projekt ist als Werkstatt konzipiert. Die praxiserprobten OER-Unterrichtsmaterialien (Open Educational Resources) bestehen aus drei Teilen. Zuerst werden die Schülerinnen und Schüler in die Grundlagen der Algorithmik und die Handhabung des Calliope mini[1] eingeführt. Im zweiten Teil lernen sie explorativ die Grundbedürfnisse eines Haustiers kennen und einschätzen. Im dritten Teil bauen sie aus Pappe ihr Lieblingstier, statten es mit dem Einplatinencomputer und passenden Sensoren aus und erstellen in einer grafischen Programmiersprache geeignete Algorithmen.

Um den unterschiedlichen Anforderungen in Primar- und Sekundarstufe gerecht zu werden, gibt es zwei Versionen: Für die Grundschule (Werkstatt A) werden die Messwerterfassungen für Essen, Trinken und Streicheln mit leitfähigem Kupferklebeband realisiert, für die weiterführenden Schulen (Werkstatt B) werden externe Sensoren zur Feuchtigkeitsmessung (Trinken), für Multitouch (Streicheln) und zum drahtlosen Auslesen von NFC-Chips (Essen) verwendet. NFC steht für Near Field Communication. Die Messungen zur Bewegung und zur Temperatur erfolgen in beiden Versionen mit im Einplatinencomputer vorhandenen Sensoren. Sämtliche Werkstattmaterialien und Beispiele für die Programmierumgebung MakeCode[4] sind online frei verfügbar.[5]

<Praktische Umsetzung>

Für den Haustiersimulator suchen sich die Schülerinnen und Schüler ein Bild ihres Lieblingstiers aus oder fotografieren es selbst. Das Bild wird auf Pappe geklebt und mit Kupferklebeband (Werkstatt A) oder externen Sensoren (Werkstatt B) versehen. Das Kupferklebeband bzw. die Sensoren werden mit Drähten an die Anschlüsse des Einplatinencomputers verkabelt und ein Programm auf den Computer geschrieben, um ihn „intelligent“ zu machen. Im Folgenden wird am Beispiel „Essen“



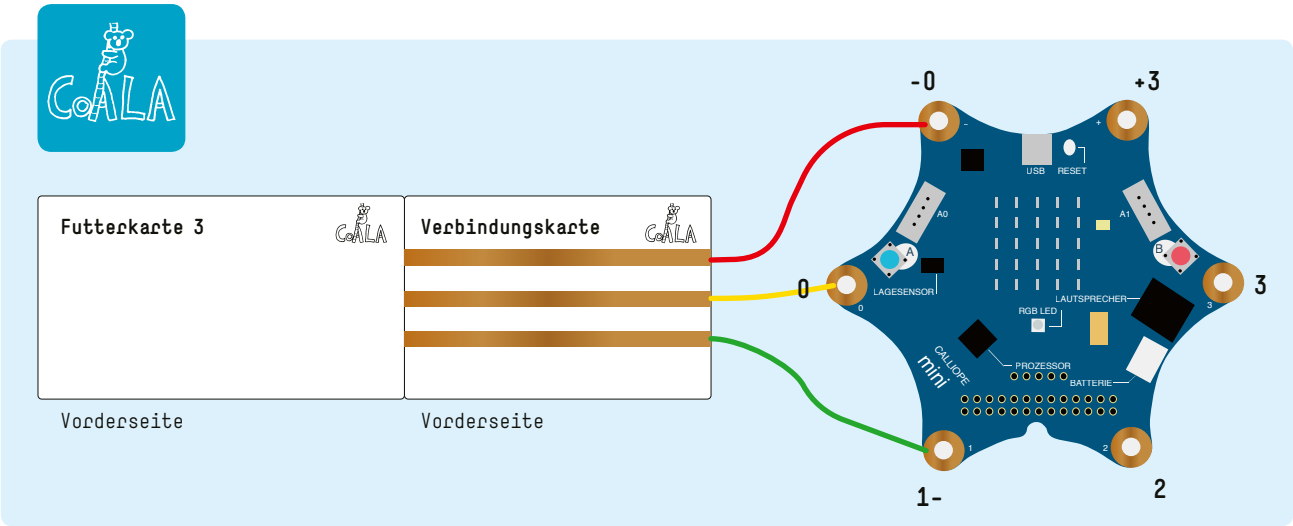


gezeigt, wie sich die beiden Versionen voneinander unterscheiden und wie die Programmierumgebung verwendet wird.

Wird ein Haustiersimulator in Mundnähe gefüttert, nutzt dieser keine Geschmackssensoren. Stattdessen „liest" ein passender Sensor die vorgehaltene Nahrung und der Algorithmus wird über bedingte Verzweigungen so gesteuert, dass die Ausgabe dem zu erwartenden Verhalten des Haustieres entspricht. So zeigt ein Katzensimulator ein lachendes Gesicht, wenn eine Maus gefüttert wird, und ein trauriges für einen Knochen. Diese Verzweigungen sind bei beiden Versionen gleich.

Grundlegend verschieden sind dagegen die verwendeten Essen-Sensoren: In Werkstatt A wird Kupferklebeband so auf mit passenden Bildern versehenen Futterkarten geklebt, dass ein an der Zunge angebrachtes Kupferklebeband eine binärcodierte Zahl liest, wenn die Futterkarte draufgehalten wird. Da die Anschlüsse des „Lesers" an einzelne Pins des Einplatinencomputers angeschlossen sind, kann im Algorithmus direkt abgefragt werden, ob die Pins jeweils kurzgeschlossen sind oder nicht: Die Futterkarten schließen also in unterschiedlichen Kombinationen an den Pins Stromkreise kurz.

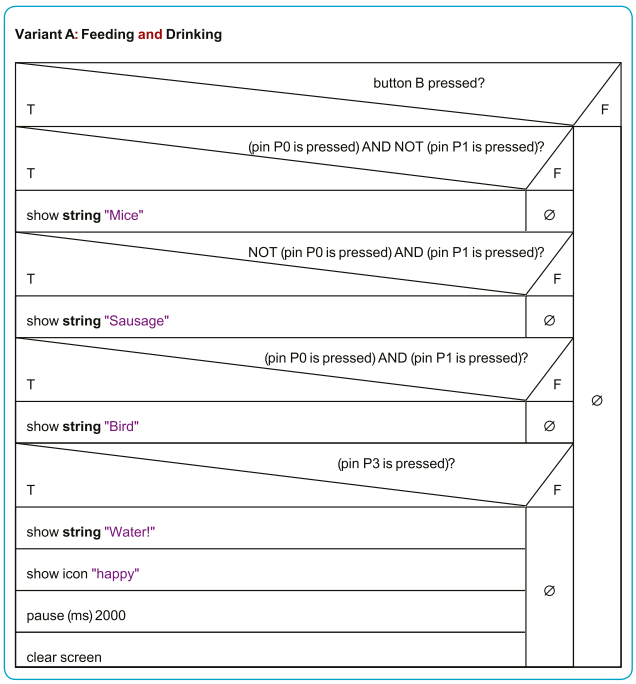
In Werkstatt B liest ein externer Sensor mit NFC-Chip und Funkantenne drahtlos Zeichenketten aus einem NFC-Tag aus. Dieser Tag kann in einem Klebeetikett oder in einer Chipkarte untergebracht sein. Im Unterschied zur Werkstatt A wird nun nicht eine (binärcodierte) Zahl ausgelesen, sondern der Name der Nahrung, z. B. „Fisch" oder „Knochen". Dadurch steigen die Umsetzungsmöglichkeiten, aber auch die Komplexität deutlich an. Im Algorithmus erfolgt nun die bedingte Verzweigung durch direkten Vergleich des ausgelesenen Werts mit einer vorgegebenen Zeichenkette. Das Beschreiben des NFC-Tags erfolgt mit einer App, das Auslesen wird didaktisch reduziert mit einem einzigen Block in MakeCode[4] realisiert, der als Erweiterung in der Programmierumgebung nachgeladen wird.



<Algorithmen in anderen Programmiersprachen>

Die Beispiele[5] können im MakeCode Programmiereditor[4] hochgeladen und direkt verwendet werden. Durch Umschalten der Ansicht von block- auf textbasiert wird der Quelltext nach JavaScript konvertiert und kann so bequem in anderen Programmiersprachen für Calliope mini [1] oder BBC micro:bit[2] übernommen werden. Die Zusatzblöcke für MakeCode, um den Multitouch- und NFC-Sensor zu verwenden, funktionieren auch für MakeCode für den BBC micro:bit.

Schließlich werden die Beispielprogramme auch als Struktogramme auf der Webseite angeboten, sodass die Algorithmen leicht nachvollzogen und auf anderen Plattformen und Programmierumgebungen, z. B. Arduino, portiert werden können.



<Fazit>

Schülerinnen und Schüler machen sich im Projekt mit den fundamentalen Konzepten der Algorithmik bekannt: Anweisungen, Sequenzen, bedingte Verzweigungen, Schleifen, Variablen. Sie lernen diese nicht durch Auswendiglernen und Wiedergeben, sondern in einem spannenden Unterrichtsprojekt mit hohem Alltagsbezug. Dafür erstellen sie aus einfachen Materialien einen Tiersimulator, den sie nach eigenen Vorstellungen zum Leben erwecken. Die zur Verfügung gestellten Materialien vermitteln die informatischen Kompetenzen in didaktisch reduzierter Form und bieten zugleich unterschiedliche Anforderungsstufen für heterogene Lerngruppen oder höhere Jahrgänge. Beide Werkstattversionen können problemlos gemischt eingesetzt werden.

Das Material wurde neben dem Calliope mini[1] auch mit dem BBC micro:bit[2] erfolgreich getestet. Für die in Werkstatt B verwendeten externen Grove-Sensoren zur Messung von Feuchtigkeit, NFC oder Multitouch ist für den BBC micro:bit jedoch eine günstige Erweiterungsplatine notwendig.[3]

<Kooperationsmöglichkeiten>

Die CoALA-Werkstatt kann in verschiedenen Kooperationsformen eingesetzt werden. Da das Material für Grund- und Sekundarschulen vorliegt, kann hier ein Austausch stattfinden. Dieser ist nicht nur für die Schülerinnen und Schüler, sondern auch für die Lehrkräfte beider Schulformen bereichernd! Werkstatt A zielt auf einfache, logische Ja/Nein-Unterscheidungen ab, Werkstatt B auf kombinierte, komplexere Bedingungen, Variablen und Zeichenkettenoperationen.

Das Werkstattmaterial kann auch gemischt werden, um die Zusammenarbeit in einer heterogenen Lerngruppe zu fördern. Die leichter nachvollziehbaren Sensoren der Werkstatt A können dann in einer weniger leistungsstarken Lerngruppe eingesetzt werden, oder Leistungsstärkere erklären die Sensoren aus Werkstatt B (Kommunikationstraining).

Im CoALA-Pilotprojekt fand eine länderübergreifende Kooperation weiterführender Schulen statt, in der sich Lerngruppen aus Deutschland und Spanien in Videokonferenzen über ihre Haustiersimulatoren austauschten. Neben Tipps zur Problemlösung bei der Programmierung wurden vor allem die Namen und Bedürfnisse der Hausiere ausgetauscht, neben Englisch auch in der jeweiligen Landessprache. Coding in den Naturwissenschaften, Sprachkurs inklusive!

<Quellen und Hinweise>

[1] https://calliope.cc
[2] www.microbit.co.uk/home
[3] Beim BBC micro:bit benötigt man zusätzlich das Grove Shield für den micro:bit.
[4] https://makecode.calliope.cc oder https://makecode.microbit.org
[5] Sämtliches Zusatzmaterial ist erhältlich auf www.science-on-stage.de/coding-materialien.

Datenfluss

<Autorin> Eleftheria Karagiorgou
<Autorin> Sevasti Tsiliki

<Info>

<Schlagwörter> Physical Computing, Säuregehalt, Wasser, Flüssigkeit, Temperatur, pH, Datenprotokollierung

<Unterrichtsfächer> Chemie

<Altersgruppe> 16 Jahre

<Hardware> Arduino Starter-Kit[1], Data Logging Shield (Datenprotokollierung), Temperatursensor, pH-Sensor, SD-Karte

<Programmiersprache> Arduino IDE – Wiring C[2]

<Programmierniveau> mittel

<Dauer des Projekts> 7 Unterrichtsstunden

<Zusammenfassung>

Die Schülerinnen und Schüler werden zu Forschenden und prüfen experimentell, ob es einen Zusammenhang zwischen dem Säuregehalt und der Temperatur von Wasser gibt. Die Umsetzung des Experiments kombiniert den Einsatz von Arduino mit Chemie.

<Vorstellung des Konzepts>

In dieser Unterrichtseinheit wird gezeigt, wie Physical Computing im MINT-Unterricht und insbesondere im Chemieunterricht mit innovativen Lehrmethoden umgesetzt werden kann. Die Einheit wurde außerhalb des Lehrplans im Rahmen unserer Robotik- und MINT-AG realisiert, die sich jeden Sonntagnachmittag für zwei Stunden trifft.

Die Schülerinnen und Schüler erhielten die Aufgabe, die bedeutende Rolle, die die Temperatur bei pH-Messungen spielt, experimentell nachzuweisen. Mit steigender Temperatur erhöhen sich die molekularen Schwingungen, sodass das Wasser ionisiert wird und sich mehr Wasserstoffionen bilden, wodurch der pH-Wert sinkt.

<Unterrichtsmethode>

Forschend-entdeckendes Lernen im MINT-Unterricht: Die Schülerinnen und Schüler sind Teil eines aktiven Lernprojekts, das auf Fragen beruht, die im Laufe des Experiments neue Fragen generieren. So erwerben sie Wissen durch praktische Umsetzung.

<Voraussetzungen - Hintergrundwissen>

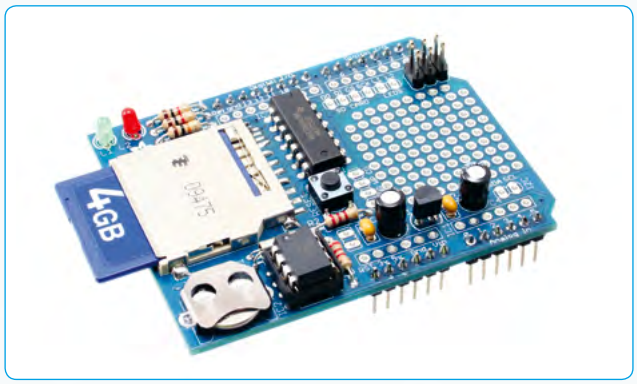
Gemäß dem griechischen Lehrplan:

- Grundkenntnisse in Programmierung, die im 3. Jahr der Sekundarstufe I und dem 1. Jahr der Sekundarstufe II erworben wurden.
- Grundkenntnisse über Säuregehalt und pH-Wert, die im 3. Jahr der Sekundarstufe I erworben wurden.

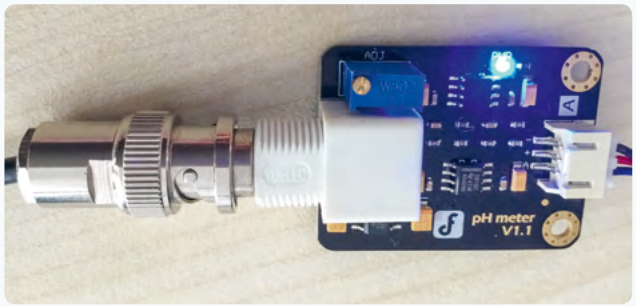
<Lehrmaterial/Raum>

Im Labor für Robotik und MINT sind folgende Materialien vorhanden:

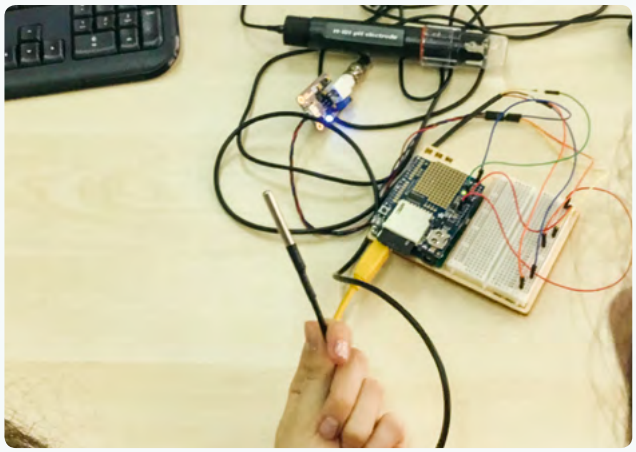
- Arduino[1] Starter-Kit (u. a. mit Arduino-Platine, Kabel, LCD-Anzeige)
- "Adafruit Data Logger Shield" für Arduino (📷 1)
- „Analog pH meter Pro Kit" zur pH-Messung für Arduino (📷 2)
- wasserdichter Temperatursensor (📷 3)
- SD-Karte
- Computer mit SD-Port, z. B. ein Laptop, für die Programmierung und die Datenprotokollierung
- demineralisiertes Wasser (speziell gereinigtes Wasser, dem alle oder fast alle Mineralien und Salze entzogen wurden)
- Kühlelemente und eine Kühltasche zur Aufbewahrung der Eiswürfel



📷 1: "Adafruit Data Logger Shield" für Arduino[3]



📷 2: Analoger pH-Wertmesser



📷 3: Wasserdichter Temperatursensor

<Forschungsfrage>

Besteht ein Zusammenhang zwischen dem Säuregehalt einer Flüssigkeit und ihrer Temperatur?

<Fragen zur Problemlösung>

1. Wie verbinden wir die Sensoren mit dem Arduino?
2. Wie führen wir die Datenprotokollierung durch?

<Praktische Umsetzung>

<Vorbereitung: Einführung - Theorie - Gruppeneinteilung>

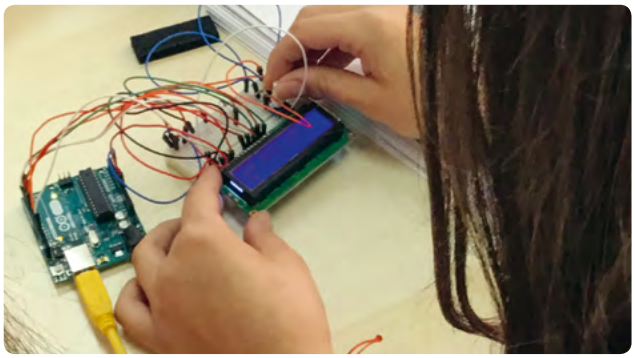
Dauer: 1 Stunde

Die Schülerinnen und Schüler werden in Gruppen aufgeteilt und erhalten eine kurze Einführung zum Arduino[1] und den verwendeten Sensoren (pH-Wert und Temperatur). Sie besprechen auch die Theorie zum Säuregehalt, dem pH-Wertmesser und der Beziehung zwischen Säuregehalt und Temperatur. Darüber hinaus machen sie sich Gedanken zur Versuchsanordnung, um die Änderung des Säuregehalts von Flüssigkeiten bei Temperaturschwankungen zu messen.

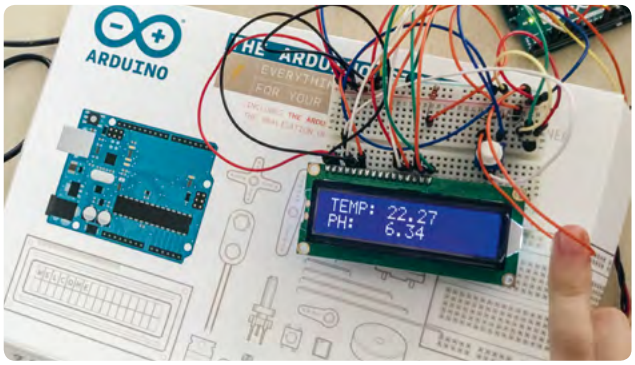
<Phase 1: Einführung zum Arduino und seiner Programmierung>

Dauer: 1 Stunde

Die Schülerinnen und Schüler machen sich mit den Schaltkreisen des Arduino und seiner grundlegenden Programmierung vertraut. Sie lernen, die LCD-Anzeige mit dem Arduino zu verbinden und das Anzeigen einer Nachricht zu programmieren. (📷 4 & 5)



📷 4: Verbinden der LCD-Anzeige



📷 5: Anzeigen der Messwerte

<Phase 2: Verbinden der Sensoren>

Dauer: 1 Stunde

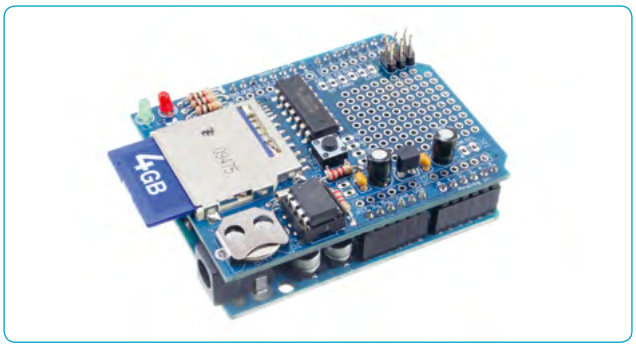
Die Schülerinnen und Schüler lernen, die Arbeitsweise des pH-Sensors (📷 6) und des Temperatursensors (📷 3) zu verstehen. Sie verbinden die Sensoren mit dem Arduino[1] und programmieren sie so, dass die Daten auf der LCD-Anzeige dargestellt werden. Dies ist eine Vorbereitung für das Verständnis der Input- und Output-Funktionen der Sensoren.

📷 6: pH-Sensor

<Phase 3: Data Logging Shield>

Dauer: 2 Stunden

Die Schülerinnen und Schüler löten das Data Logging Shield an die Arduino-Platine[1] mit der SD-Karte zur Datenprotokollierung (📷 7). Sie programmieren das Shield, das über eine eigene Echtzeituhr (RTC, „real time clock") verfügt. Sie starten das Experiment mit demineralisiertem Wasser bei 25 °C (neutral) und messen pH-Wert und Temperatur durch Eintauchen des Sensors in die Flüssigkeit für 10 Sekunden.

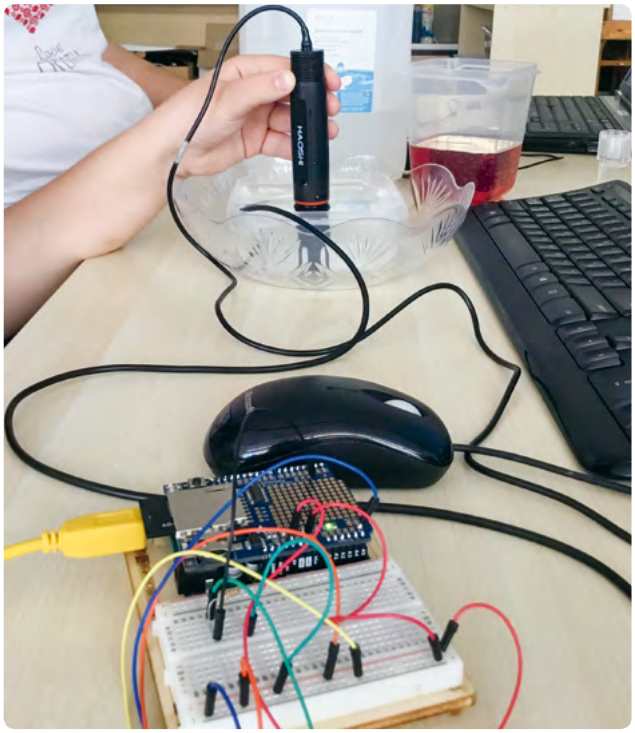


📷 7: An Arduino gelötetes Data Logging Shield[4]

<Phase 4: Das Experiment>

Dauer: 1 Stunde

Die Schülerinnen und Schüler testen demineralisiertes Wasser bei verschiedenen Temperaturen. Sie beginnen mit Wasser bei Raumtemperatur in einer Schüssel oder einem Becherglas, welches durch Eiswürfel im umgebenden Wasserbad gekühlt wird (📷 8 & 9). Nach jeweils 1 Minute tauchen die Schülerinnen und Schüler die Sensoren 10 Sekunden lang in die Flüssigkeit ein. Sie wiederholen das Verfahren mindestens 6 Mal, um eine große Datenmenge für die Auswertungsphase zu erhalten. Durch das Wasserbad, wird die Flüssigkeit langsam und gleichmäßig gekühlt.



8: Messung von pH-Wert und Temperatur

9: Wasserbad mit Eiswürfeln

<Phase 5: Ergebnisse>

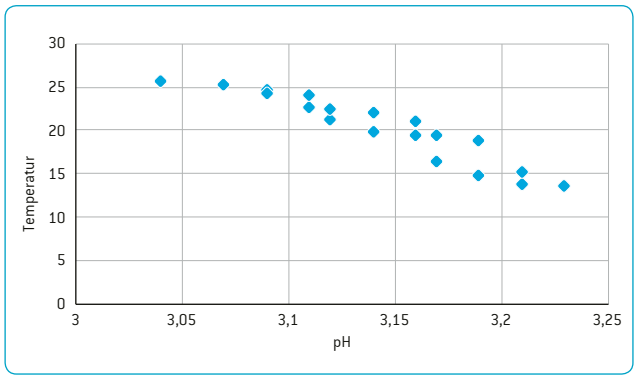
Dauer: 1 Stunde

Die Schülerinnen und Schüler nehmen die SD-Karte aus dem Data Logging Shield und verwenden einen Laptop, um die Daten auszulesen. Aus diesen erstellen sie mit einem Tabellenkalkulationsprogramm (z. B. MS-Excel) ein Diagramm, das einen möglichen Zusammenhang zwischen Temperatur und Säuregehalt aufzeigt. Die Daten werden auf der SD-Karte als .csv-Datei gespeichert, die mit einer Tabellenkalkulation geöffnet werden kann. Die Lehrkraft bespricht mit den Schülerinnen und Schülern die Ergebnisse ihrer Datenprotokollierung und ob sie in der Lage waren, die gestellten Fragen zur Problemlösung zu beantworten. Sie präsentieren ihre Ergebnisse und diskutieren diese mit der Klasse.

<Fazit>

Am Ende kann von den Schülerinnen und Schülern erwartet werden, dass sie die Verbindung zwischen MINT-Fächern verstehen, da sie theoretische Konzepte der Chemie experimentell mit Physical Computing umgesetzt haben. Außerdem fördert die Einheit forschend-entdeckendes Denken und Handeln und die Schülerinnen und Schüler erkennen, wie Schulwissen in der realen Welt angewandt werden kann. Die Entwicklung von Soft Skills, wie Zusammenarbeit bei der Lösung von Problemen und dem Erarbeiten von Projekten, ist wichtig für ihre Zukunft. Nicht zuletzt ist dies für sie eine großartige Gelegenheit, ihre Leistungen im MINT-Bereich zu verbessern und die Bedeutung des fächerübergreifenden Konzepts dieses Projekts zu verstehen.

Das Experiment kann mit verschiedenen Flüssigkeiten durchgeführt werden, z. B. Essig im Wasserbad, entweder mit Eiswürfeln im Wasserbad, oder mit einem warmen Wasserbad (siehe 10).



10: Experimentelle Daten von Essig

Das Data Logging Shield muss sehr präzise mit der Arduino-Platine verlötet werden, was für einige Schülerinnen und Schüler schwierig sein könnte. Deshalb kann es erforderlich sein, dass Sie ihnen dabei helfen oder selbst löten.

<Quellen und Hinweise>

[1] www.arduino.cc

[2] www.arduino.cc/en/Main/Software

[3] Bild: oomlout (https://commons.wikimedia.org/wiki/File:ARSH-09-DL 03.jpg), „ARSH-09-DL 03“, CC BY-SA 2.0, https://creativecommons.org/licenses/by-sa/2.0/legalcode

[4] Bild: oomlout (https://commons.wikimedia.org/wiki/File:ARSH-09-DL_(5703636953).jpg), „ARSH-09-DL (5703636953)“, CC BY-SA 2.0, https://creativecommons.org/licenses/by-sa/2.0/legalcode

Schiff ahoi!

<Autorin> Immaculada Abad Nebot
<Autor> Pere Compte Jové

<Info>

<Schlagwörter> Fernsteuerung, 2D- und 3D-Design, Modellieren, Löten einer elektronischen Platine, Chip-Programmierung, App-Programmierung, 3D-Drucker

<Unterrichtsfächer> Technik

<Altersgruppe> 14–16 Jahre

<Hardware> Arduino[1], Bluetooth-Modul, Material für den Bau eines Modellboots

<Programmiersprache> Arduino, ArduinoBlocks[2], AppInventor[3]

<Programmierniveau> mittel

<Zusammenfassung>

Die Schülerinnen und Schüler entwickeln und bauen ihr eigenes Boot, das sie in einem Schwimmbecken navigieren. Nachdem sie diese erste Aufgabe erledigt haben, verwenden sie eine Arduino-Platine, um ihr Boot mit einem Tablet oder Smartphone fernzusteuern.

<Vorstellung des Konzepts>

Die Schülerinnen und Schüler entwerfen ihr eigenes Modellboot und lernen bei dieser Aufgabe, die Sichtweise eines Ingenieurs einzunehmen. Das Projekt beginnt mit der Analyse verschiedener Schiffstypen im Internet. Danach werden in kleinen Gruppen Modelle gebaut, die stabil im Wasser liegen.[4]

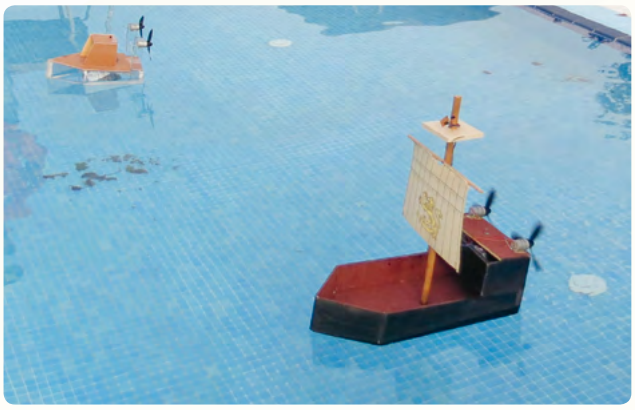
Das Schiff ist mit einem selbst gebauten Motorwendeschalter ausgestattet, der die Steuerung zweier Motoren ermöglicht (beide laufen vorwärts und rückwärts).

Die Schülerinnen und Schüler bauen ein Bluetooth-Gerät ein, sodass sie das Boot via Smartphone fernsteuern können. Mit AppInventor[3] programmieren sie eine App mit verschiedenen Steuerungssystemen, z. B. mit Tasten, Sprachsteuerung oder einem Beschleunigungssensor (das Boot wechselt die Richtung je nach Handhaltung).

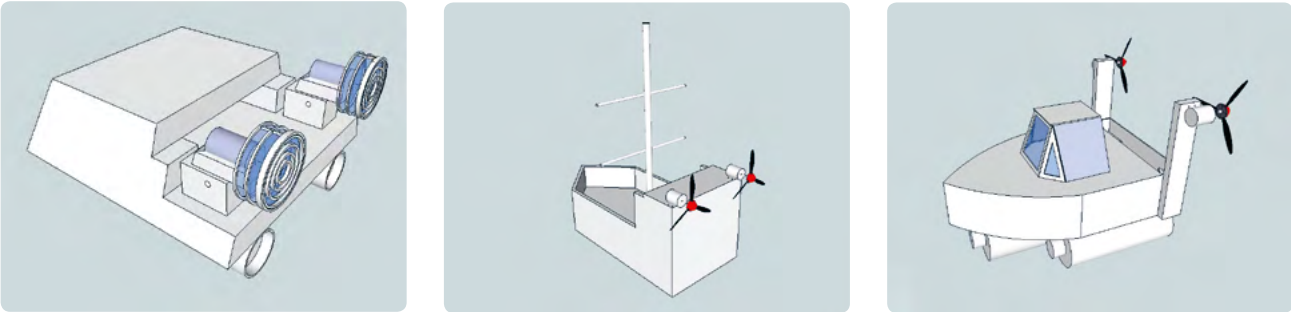
Schließlich präsentieren sie ihre Designs in einer Ausstellung für Modellboote (📷 1) den anwesenden Experten, um mit ihnen mögliche Mängel ihrer Modelle zu besprechen. Durch diese Diskussionen erhalten sie Anregungen zur Verbesserung zukünftiger Entwürfe.

<Praktische Umsetzung>

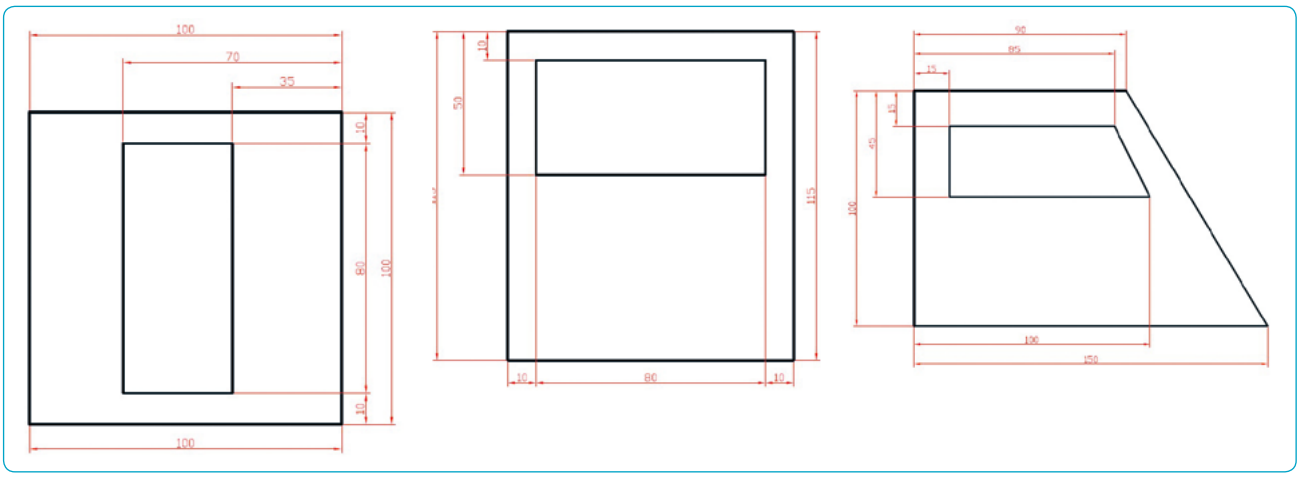
Die Schülerinnen und Schüler entwickeln ein Design ihrer Wahl, nachdem sie sich über die verschiedenen Bootstypen im Internet informiert haben. Sie können das Boot mit einer 3D-Software wie sketchUp[5] oder Tinkercad[6] skizzieren. Dabei ist es wichtig, dass das Boot sehr stabil im Wasser liegt, damit es nicht kentert. (📷 2)



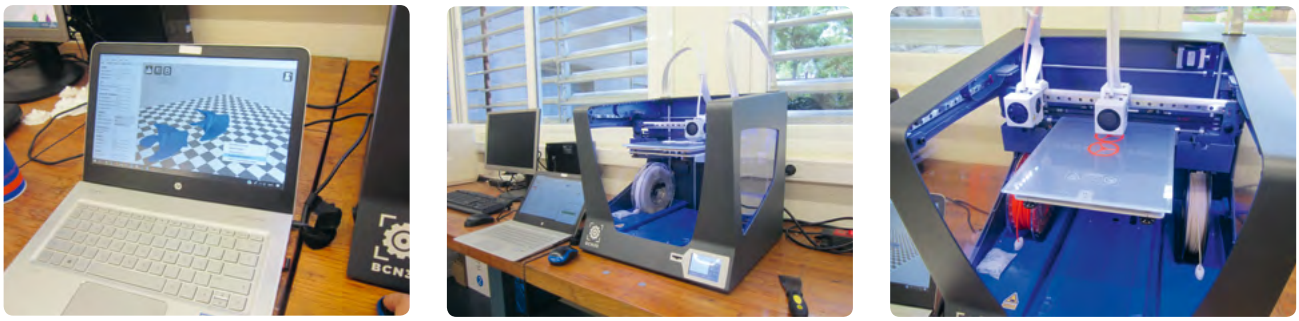
📷 1: Präsentation der Modellboote bei einer Modellbootausstellung in Spanien



2: Verschiedene 3D-Modelle



3: Tür, Front und Seite der Bootskabine

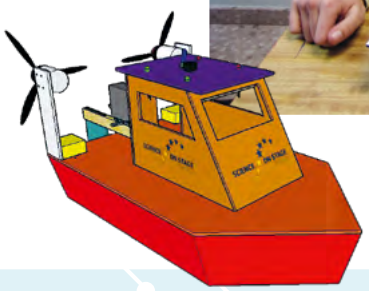


4a-c: Ausdrucken der Ruderblätter im 3D-Druckerzentrum Cesire Aulatec, Barcelona

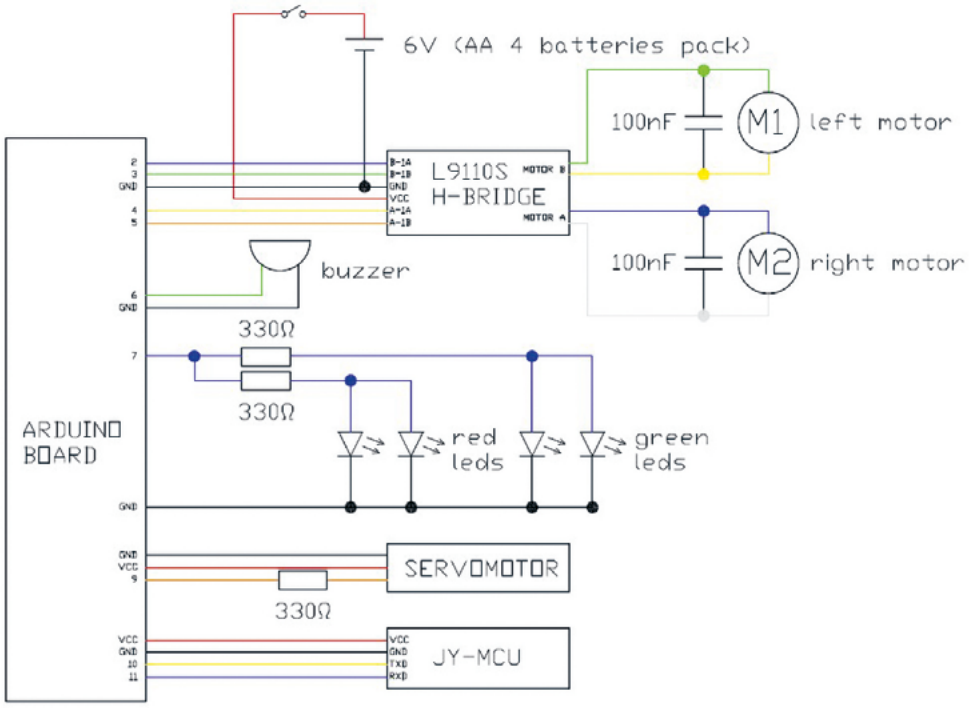
Die Pläne für eines der Modelle (3) sind online verfügbar. Wenn Sie das Design ändern oder ausdrucken möchten, können Sie es in den Formaten .skp, .stl und .gcode herunterladen.[4]

Wenn Ihre Schule keinen 3D-Drucker hat, ist es oft möglich, mit anderen Institutionen wie Universitäten oder Makerspaces zusammenzuarbeiten. In unserem Fall druckten die Schülerinnen und Schüler ihre Entwürfe in einem 3D-Druckerzentrum aus. (4a–c)

Die Schritt-für-Schritt-Anleitungen für den Bau des Boots sind online verfügbar.[4]



5, 6: Fertiges Boot und 3D-Modell

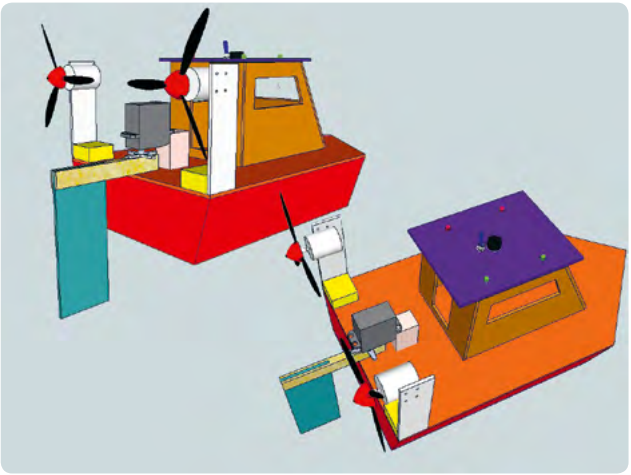


7: Schaltkreis

<Schaltkreis>

Das Zubehör wir nach dem Schaltplan in 7 mit der Arduino-Platine[7] verbunden. Dabei werden die folgenden Anleitungen beachtet:

1. Auf dem Dach des Boots können Sie einen Summer (Arduino-Pin 6) und Lichter (Arduino-Pin 7) installieren. (8)
2. Am Bootsheck verbinden Sie einen Servomotor (Arduino-Pin 9) mit dem Ruder, um es zu steuern. (8)
3. Nun verbinden Sie das Bluetooth-Modul gemäß Schema TXD (Arduino-Pin 10) und RXD (Arduino-Pin 11).
4. Verbinden Sie dann eine L9110S-Motortreiber-Steuerungsplatine für Arduino mit externen Batterien. (7)



8: Boot mit Zubehör

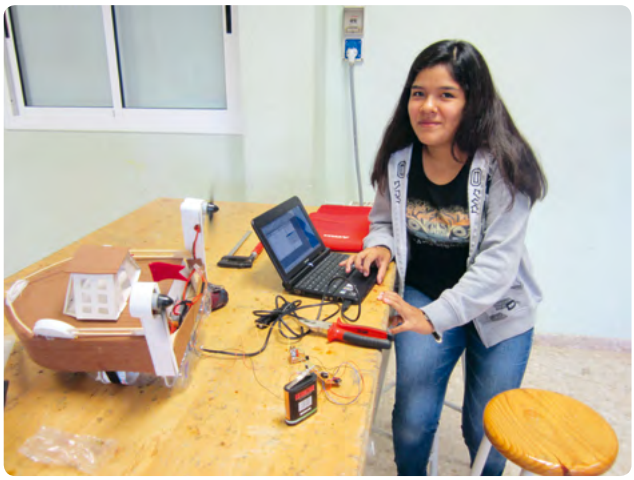
9: Konstruktion der Boote

<Motorensteuerung und weitere Funktionen des Boots>

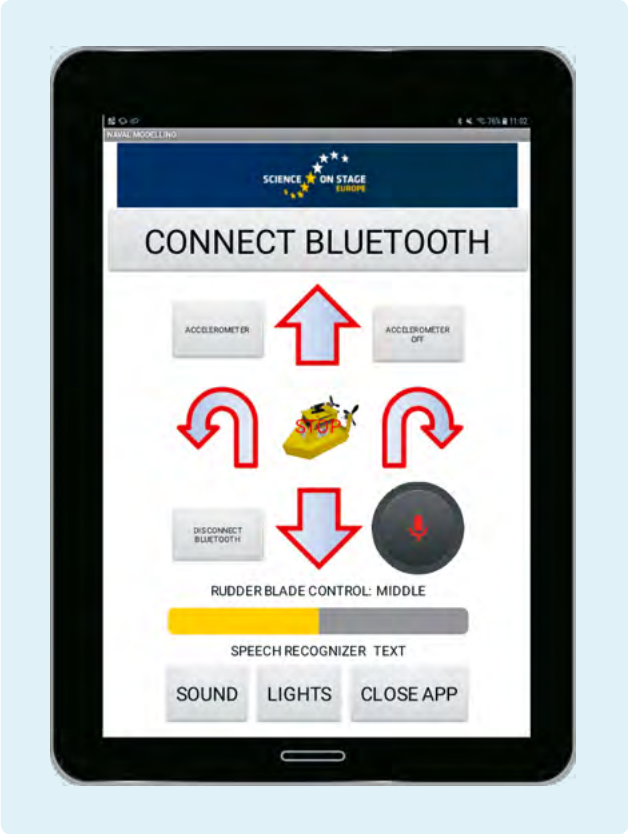
Wir empfehlen den Arduino mit Arduino IDE[1], ArduinoBlocks[2] oder einem ähnlichen Programm zu programmieren. Die Schülerinnen und Schüler können die folgenden Funktionen programmieren:

1. Die Lichter auf dem Dach werden ein- und ausgeschaltet.
2. Der Summer erzeugt einen Ton.
3. Die Position des Servomotors wird gesteuert (40° rechts, 20° rechts, Mittelposition, 20° links und 40° links).
4. Beide Motoren drehen sich und das Boot fährt vorwärts.
5. Beide Motoren drehen sich und das Boot fährt rückwärts.

6. Das Boot fährt nach rechts (der linke Motor dreht sich vorwärts, der rechte Motor rückwärts).
7. Das Boot fährt nach links (der rechte Motor dreht sich vorwärts, der linke Motor rückwärts).
8. Alle Programme sind mit Bluetooth steuerbar.



10: Programmierung des Arduino



11: Benutzeroberfläche der App

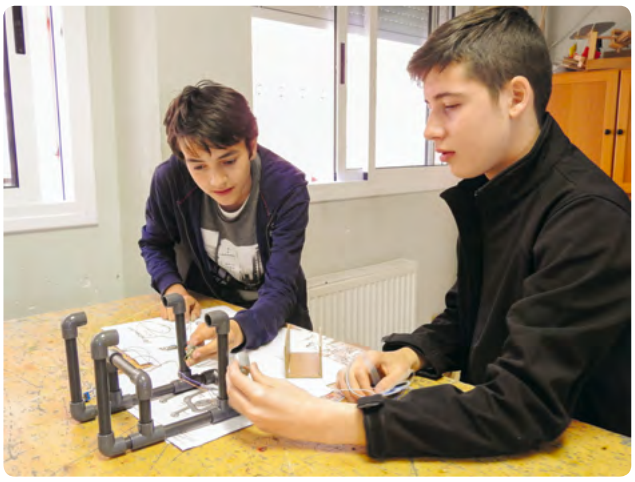
<Programmierung der App mit AppInventor[3] zur Steuerung des Boots mit einem Smartphone>

1. Programmierung der App, sodass sie für die Verbindung zum Boot Bluetooth verwendet
2. Steuerung der verschiedenen Bootselemente mit Tasten
3. Steuerung des Ruderblatts mit einer Scroll-Leiste
4. Steuerung des Boots mit dem Beschleunigungssensor des Tablets oder Smartphones (durch das Kippen des Smartphones nach vorne, hinten, links oder rechts bewegt sich das Boot in die entsprechende Richtung)
5. Verwendung der Sprachsteuerung, um das Boot mit der Stimme zu steuern
6. Kombination dieser Programme

<Materialliste und erforderliche Ausrüstung>

Eine Liste mit den erforderlichen Materialien ist online verfügbar.[4] Die Liste führt die erforderlichen Mengen, die Preise und mögliche Anbieter auf.

Das Material für ein Boot kostet ungefähr 21 €. Wir gaben ungefähr 15 € für die Elektronik und 6 € für das übrige erforderliche Material aus.

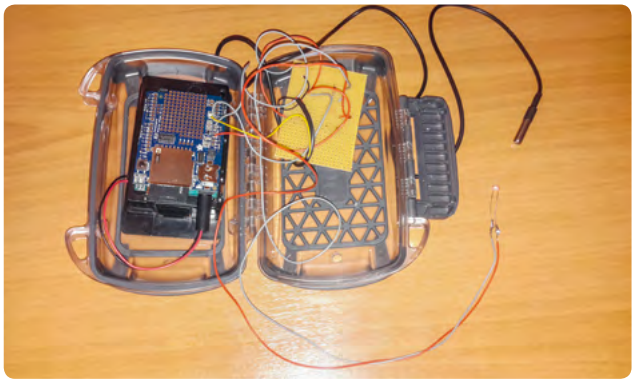


12: Bau des Hydrobots

<Kooperationsmöglichkeiten>

Bei der Erarbeitung dieser Unterrichtseinheit arbeiteten wir mit Eleftheria Karagiorgou und Sevasti Tsiliki vom 7. Gymnasium in Trikala, Griechenland, zusammen. Wir bauten den Arduino-Schaltkreis in ein anderes Modell ein, einen Hydrobot, mit dem unter Wasser navigiert werden kann. In diesem Fall musste vermieden werden, dass Wasser eindringt, weshalb es sehr wichtig war, die Motoren mit einer Wachsschicht zu schützen und sowohl ein schützendes als auch wasserdichtes Gehäuse für die Arduino-Platine zu verwenden. Um unserem Hydrobot „Argolith“[8] ein „Gehirn“ zu verleihen, statteten wir ihn mit dem Mikrocontroller Arduino Uno aus, damit wir Helligkeits- und Temperaturmessungen unter Wasser durchführen konnten. Zudem benutzten wir ein Data Logging Shield (Datenprotokol-

lierung), das eine SD-Karte zur Speicherung der gemessenen Daten, eine Echtzeituhr und einen Aufnahmeschaltkreis lieferte. Auf diese Weise erhielt „Argolith“ ein elektronisches Gehirn.



13: Arduino mit wasserdichtem Gehäuse

Das Online-Material enthält auch ein Unterwasser-Video des „Argolith“-Hydrobots während einer Testfahrt in einem Fluss in Trikala, Griechenland.[4]

<Quellen und Hinweise>

[1] www.arduino.cc

[2] www.arduinoblocks.com

[3] http://appinventor.mit.edu

[4] Alle Schritte dieses Projekts und weitere Informationen: www.science-on-stage.de/coding-materialien.

[5] www.sketchup.com

[6] www.tinkercad.com

[7] www.arduino.cc/en/Reference/Board

[8] Die Bauanleitung finden Sie unter http://seaperch.mit.edu/build.php.

Wie wird programmiert?

<Autor> Bernard Schriek

In diesem Artikel geben wir Ihnen eine Einführung in das Programmieren in den Naturwissenschaften. Um die Unterrichtseinheiten in dieser Broschüre nutzen zu können, benötigen Sie spezielle Hardware und Software. Zudem ist es hilfreich, wenn Sie über Grundkenntnisse im Programmieren verfügen. Dieses Kapitel liefert Ihnen die erforderlichen Informationen und Angaben.

<Hardware>

Beim Programmieren in den Naturwissenschaften geht es hauptsächlich um die Messung physikalischer und chemischer Größen mithilfe von Sensoren sowie um die Steuerung gewisser Aktoren. Typische Sensoren messen unter anderem Temperatur, Lautstärke, Licht, Entfernung, pH-Wert, das Drücken einer Taste oder Berührung. Typische Aktoren sind LEDs, Summer, Lautsprecher, Motoren usw.

<Arduino>

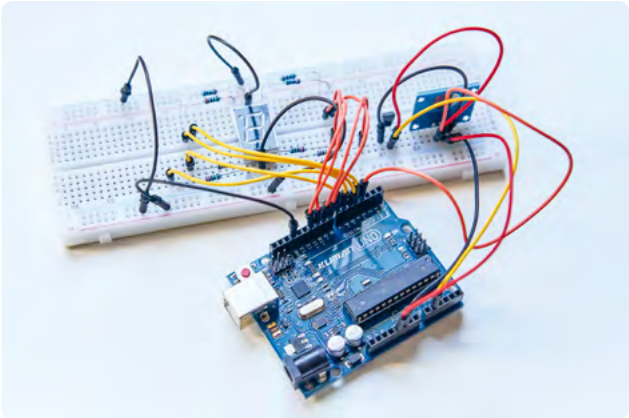
Sensoren und Aktoren können mit mehr oder weniger hoch entwickelten Mikrocontrollern oder Einplatinen-Mikrocomputern verbunden werden oder sind schon darin eingebaut. Die kleinste Platine heißt Arduino[1]. Sie ist in verschiedenen Versionen erhältlich, aber Arduino UNO ist die am meisten an Schulen eingesetzte Version (so auch in dieser Broschüre). Auf der Platine befindet sich ein 8-Bit-Mikrocontroller, der relativ langsam läuft. Die Geschwindigkeit des Prozessors ist jedoch bei den meisten Anwendungen nicht von Bedeutung: Die Platine kommuniziert nur über eine Reset-Taste und eine LED.

Die Platine kann mit verschiedenen anderen Sensoren und Aktoren verkabelt werden. Die Programme für Arduino werden auf einem Computer geschrieben und dann via USB an den Arduino geschickt. Sobald das Programm auf den Arduino geladen ist, kann es durch Drücken der Reset-Taste gestartet (und später unterbrochen und wieder gestartet) werden. Wenn der Arduino via USB verbunden ist, läuft die Stromzufuhr über die USB-Verbindung. Wenn die Verbindung zum Computer unterbrochen wird, benötigt er eine eigene Stromzufuhr.

Eine sehr praktische Art, eine Arduino-Platine zu erweitern, ist der Einsatz von sogenannten „Shields“, d. h. Steckplatinen, die als Schaltungserweiterungen direkt auf die Arduino-Pinköpfe aufgesteckt werden. Zum Beispiel werden oft Datalogger-Shields eingesetzt, die über eine Echtzeituhr auf der Platine verfügen. Die Daten werden auf eine SD-Karte geschrieben, die in den SD-Kartenleser des Shields eingeführt wird.

Für fast alle Erweiterungen (Shields, Sensoren und Aktoren) finden Sie gut dokumentierte Programmierbeispiele im Internet.

Für die Programmierung von Arduino können Windows-, Mac OS- oder Linux-Computer verwendet werden. Die Software steht als Download zur Verfügung.[2] Im Internet finden Sie viele detaillierte Programmierbeispiele für den Arduino. Übrigens: Programme für den Arduino werden „Sketches“ genannt.



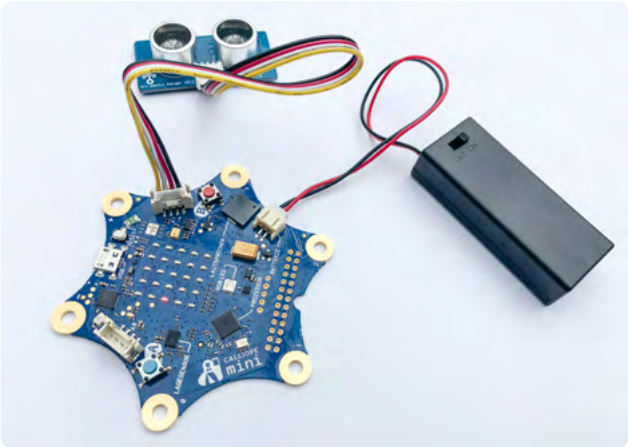
1: Arduino

<Calliope mini und BBC micro:bit>

Ein weiterer Einplatinen-Computer ist der Calliope mini[3], der mit dem BBC micro:bit[4] vergleichbar ist. Der Unterschied zwischen diesen beiden Platinen liegt darin, dass beim Calliope mini zahlreiche Sensoren und Aktoren bereits eingebaut sind, sodass Sie bei vielen Projekten keine externen Sensoren und Aktoren benötigen. Der Calliope mini verfügt über Bluetooth für die Kommunikation mit anderen Platinen oder mit Smartphones. Sein Prozessor ist leistungsfähiger und schneller als der Arduino-Prozessor. Außerdem verfügt der Calliope mini über wesentlich mehr Onboard-Speicherplatz. Auf einem Computer geschriebene Programme können via USB auf den Calliope mini übertragen werden. Sobald ein Programm übertragen ist, startet es automatisch, kann aber auch durch Drücken der Reset-Taste neu gestartet werden. Obwohl der Calliope mini schon über viele eingebaute Sensoren verfügt, können weitere über standardisierte Grove-Steckverbinder[5] angeschlossen werden.

Auf der Platine befindet sich eine 5×5 LED-Matrix, die für das Scrollen durch Texte verwendet werden kann. Der Calliope mini ist etwas teurer als der Arduino, hat aber den Vorteil der vielen bereits eingebauten Sensoren und Aktoren, wodurch sich der höhere Preis im Schulalltag bezahlt macht.

Für die Programmierung des Calliope mini wird JavaScript verwendet, doch Sie können auch Umgebungen mit Blockprogrammierung verwenden, die für jüngere Kinder besser geeignet sind. Diese werden später in diesem Kapitel beschrieben.

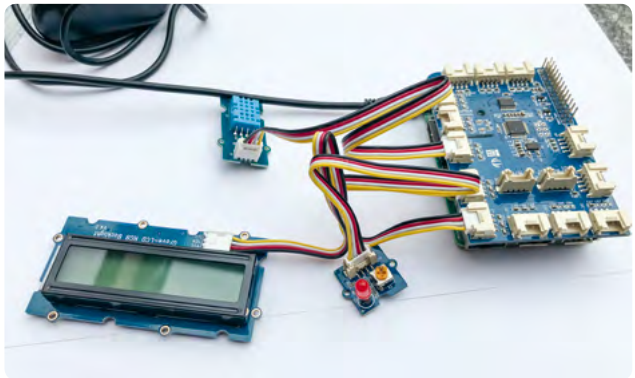


2: Calliope mini mit Ultraschall-Entfernungssensor und Batteriefach

<Raspberry Pi>

Ein sehr bekannter Einplatinen-Computer ist der Raspberry Pi[6], ein voll funktionsfähiger Computer, der mit Linux- oder Windows-Betriebssystem läuft. Er kann über ein HDMI-Kabel mit einem Bildschirm verbunden und über USB können Maus und Tastatur angeschlossen werden. Der Preis von 50 € bis 60 € ist sehr angemessen, jedoch doppelt so hoch wie der Preis eines Calliope mini. Sie können beinahe jede Programmiersprache auf dem Raspberry Pi verwenden. Statt einer Harddisk nutzt er eine SD-Karte für die Speicherung von Programmen und Daten. Er kann für verschiedene Zwecke mit Erweiterungsplatinen ergänzt werden. Der Raspberry Pi verfügt über eingebautes WLAN und kann an ein lokales WLAN-Netz angeschlossen werden. Wie alle anderen Platinen benötigt er eine externe Stromzufuhr, entweder mit Batterien oder mit einem Netzadapter. Obwohl der Raspberry Pi ein vollständiger Computer ist, ist er nicht so schnell wie ein normaler Desktop- oder Laptop-Computer und verfügt nur über eine SD-Karte als externen Speicher. Für ihn gibt es zahlreiche Projekte im Internet. Für die Arbeit mit dem Raspberry Pi sollten Sie über eine gewisse Erfahrung mit Computern verfügen.

3: Raspberry Pi



4: Raspberry Pi mit Erweiterungsplatine

<LEGO Mindstorms>

Ein weiteres, jedoch viel teureres, Mikrocomputer-System ist LEGO Mindstorms[7]. Die meisten Schülerinnen und Schüler kennen LEGO schon, sodass sie damit computergesteuerte Werkzeuge und Apparate leicht bauen können. LEGO hat ein eigenes visuelles Programmiersystem namens EV3, eine Software, die auf LabView[8] basiert (eine professionelle Mess- und Steuerungs-Software). Bei diesem System bewegt man sogenannte Programmierblöcke, um ein funktionierendes Programm zusammenzufügen. Der Einsatz und die Programmierung von Motoren sind sehr wichtig, wenn Sie einen EV3-Roboter programmieren. Neben der LEGO-Software können Sie leJOS[9] verwenden, eine spezielle Umsetzung der Java Virtual Machine.

Da Eclipse über ein bestehendes LEGO-Plugin verfügt, wird es meistens als Java-Entwicklungstool verwendet. Es gibt viele Sensor- und Aktorbausteine, die mit dem Hauptprozessorbaustein verbunden werden können, aber wie dies bei allen Komponenten von LEGO ist, sind sie viel teurer als die Sensoren für die zuvor erwähnten Platinen.

Es gibt eine große Auswahl weiterer Einplatinen-Computer, von denen viele für den Betrieb von Robotern für die verschiedensten Zwecke konzipiert sind. Wenn Sie Projektideen suchen, schauen Sie am besten einmal online bei hackster.io[10] nach. Sie müssen sich dort (kostenlos) registrieren, um auf zahlreiche mehr oder weniger detaillierte Projektbeschreibungen zugreifen zu können.

<Software>

Im 21. Jahrhundert sind Programmierkompetenzen in allen Lebensbereichen von zunehmender Bedeutung. Ein Ziel dieser Broschüre ist es, Lehrkräfte darin zu unterstützen, das Interesse ihrer Schülerinnen und Schüler für Coding zu gewinnen und zu fördern. Normalerweise interessieren sich Schülerinnen und Schüler sehr für das Programmieren. Um dabei erfolgreich zu sein und dadurch motiviert zu werden, benötigen sie nur noch die richtigen Werkzeuge und Programmiersprachen. Die Wahl der geeigneten Programmiersprache hängt vom Alter der Schülerinnen und Schüler ab. Jüngere Kinder benötigen eher visuelle Unterstützung beim Programmier- und Debugging-Prozess. In den folgenden Abschnitten behandeln wir deshalb die Umsetzung einiger wichtiger Programmierkonzepte in der Block- und in der Textprogrammierung.

<Variablen>

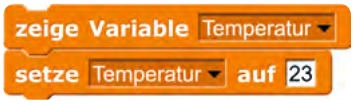
Variablen werden eingesetzt, um Werte für einen späteren Gebrauch festzuhalten. Eine Variable stellt man sich am besten als eine Box vor. Eine Box enthält einen Wert, welcher eine bestimmte Form hat, die vom Wertetyp abhängt, z. B. Ganzzahl (Integer), Gleitkommazahl (Float), Zeichenkette (String) oder Boole (also wahr oder falsch). Die Box ist außerdem mit einem Label mit ihrer Bezeichnung ausgestattet. Es empfiehlt sich, erklärende Worte für die Bezeichnungen der Variablen zu nehmen, wie beispielsweise „Temperatur“ anstelle des Buchstabens *t*, sodass andere ein Programm leichter verstehen können. In Blockprogrammiersprachen (Scratch[11], Snap![12], MakeCode[13]) sieht eine Variable wie ein Label aus, und Sie können die Bezeichnung und den Wert auch sehen, wenn das Programm gerade läuft (📷 5). Dies ist gerade bei der Fehlerbehebung (Debugging) sehr hilfreich. Es gibt aber auch Blöcke zur Festlegung oder Änderung des Werts einer Variablen, die selbsterklärend sind.

📷 5

<Zuweisungen>

In einigen Textprogrammiersprachen müssen Variablen deklariert werden. Die Deklaration enthält Angaben zum Typ der Variable (z. B. Integer, String). Andere Textprogrammiersprachen warten darauf, dass die erste Zuweisung entscheidet, von welchem implizierten Typ die Variable ist.

Doch die Zuweisung selbst enthält eine Hürde: Sie wird wie eine mathematische Gleichung geschrieben. Temperatur = 23 ist eine Zuweisung und bedeutet, dass die Temperatur-Variable den Wert 23 erhält (📷 6). Wenn Sie nun die Temperatur mit dem Wert 25 vergleichen wollen, schreiben Sie: Temperatur == 25, d. h. mit zwei Gleichheitszeichen. Dieser Unterschied ist eine häufige Ursache für Fehler (Bugs) bei der Softwareentwicklung.



📷 6

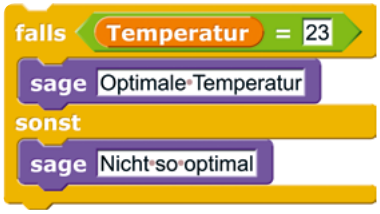
<Programmablauf>

Die meisten Computerprogramme bestehen aus Anweisungen, die seriell abgearbeitet werden, d. h. eine nach der anderen. Natürlich enthalten auch viele Sprachen die parallele Verarbeitungsweise, wo zwei oder mehr Programme gleichzeitig in sogenannten Threads ablaufen – auch Scratch[11] oder Snap![12] bieten diese parallele Verarbeitung an.

Aber wir betrachten jetzt nur einen Prozess. Die Anweisungen werden beginnend mit dem ersten Block oder der ersten Zeile bis zum letzten Block bzw. der letzten Zeile ausgeführt. Dieser Vorgang heißt Sequenz. Jedoch ist es meist nicht so einfach: Das Programm soll verschiedene Befehle ausführen, basierend auf Entscheidungen an bestimmten Punkten im Programm. Dieser Vorgang heißt Verzweigung (Branching).

<Verzweigung>

Es gibt drei Verzweigungstypen: einseitige, zweiseitige und mehrseitige. Sie alle benötigen eine Bedingung, die anzeigt, welche Aktion Sie vom Programm verlangen. Diese Bedingung ist in den meisten Fällen das Vergleichen zweier Werte. Das Ergebnis ist ein Boole'sches Wahr/Falsch. Die einseitige Verzweigung fügt zusätzliche Anweisungen zum Programmablauf hinzu, die ausgeführt werden, wenn das Ergebnis der Bedingung „Wahr" ist. Die zweiseitige Verzweigung fügt zwei zusätzliche Anweisungssets hinzu, von denen jeweils, abhängig vom Ergebnis der Bedingung, nur eine ausgeführt wird. Die mehrseitige Verzweigung nimmt eine Variable, und abhängig vom Wert dieser Variable werden verschiedene Anweisungs-Sets ausgeführt. Blockprogrammiersprachen stellen die Bedingung und auch die Anweisungssets visuell klar verständlich dar. Textprogrammiersprachen verwenden sogenannte Wenn- (If) oder Wenn-Sonst-Anweisungen (If-Else) für die ersten zwei Verzweigungsarten und Fall-Anweisungen (Case) für mehrseitige Verzweigungen.



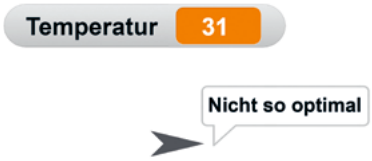
7

8

<Schleifen>

Ein weiteres wichtiges Instrument zur Steuerung des Anweisungsablaufs sind Schleifen (Loops), die für wiederholt ausgeführte Anweisungssets eingesetzt werden. Schleifen werden anhand der Bedingung, die sie steuern, unterschieden. Die Bedingung – meist ein Vergleich – kann sich am Anfang oder am Ende einer Schleife befinden. Wenn sie sich am Anfang befindet und in „Falsch" resultiert, werden die Anweisungen nicht ausgeführt. Wenn sich die Bedingung am Ende der Schleife befindet, werden die Anweisungen in der Schleife mindestens einmal ausgeführt. Es gibt auch numerische Schleifen (manchmal „For-Schleifen" oder „For Loops" genannt), die verwendet werden, wenn bekannt ist, wie viele Male die Schleifen-Anweisungen ausgeführt werden sollen.

9



10

Anweisungssequenzen, Schleifen und Verzweigungen werden üblicherweise verschachtelt. Dies bedeutet, dass eine Schleife sich innerhalb einer Verzweigung und eine Verzweigung sich innerhalb einer Schleife befinden kann. Es gibt Regeln, wie ein Programmablauf in einem Diagramm dokumentiert wird. In dieser Broschüre verwenden wir Nassi-Shneiderman-Diagramme[14] zur Erläuterung von Anweisungsabläufen in unseren Programmen.

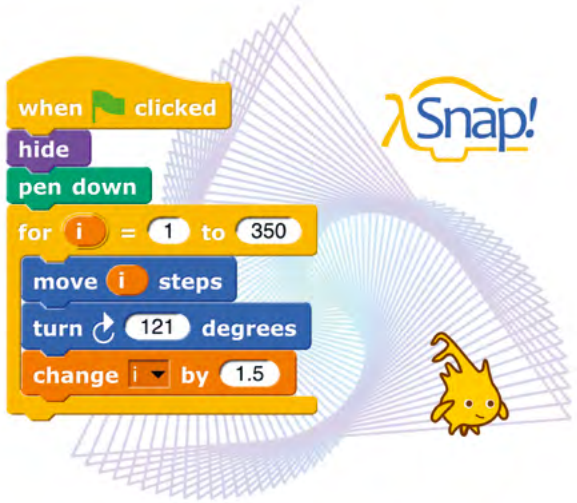
<Quellen und Hinweise>

[1] www.arduino.cc
[2] www.arduino.cc/en/Main/Software
[3] https://calliope.cc
[4] www.microbit.co.uk/home
[5] http://wiki.seeedstudio.com/Grove_System
[6] www.raspberrypi.org
[7] www.lego.com/de-de/mindstorms
[8] https://de.wikipedia.org/wiki/LabVIEW
[9] www.lejos.org
[10] www.hackster.io
[11] https://scratch.mit.edu
[12] https://snap.berkeley.edu
[13] https://makecode.calliope.cc/
[14] https://de.wikipedia.org/wiki/Nassi-Shneiderman-Diagramm

<Informatikunterricht mit Snap!>

snap.berkeley.edu/run

Kreativität sowie Computer- und Medienkompetenzen sind besonders wichtig in einer Zeit der digitalen Revolution. *Snap!* ist ein Tool, das Menschen jeden Alters dabei hilft, Informatikgrundlagen kennenzulernen und aktuelle Entwicklungen aktiv mitzugestalten.



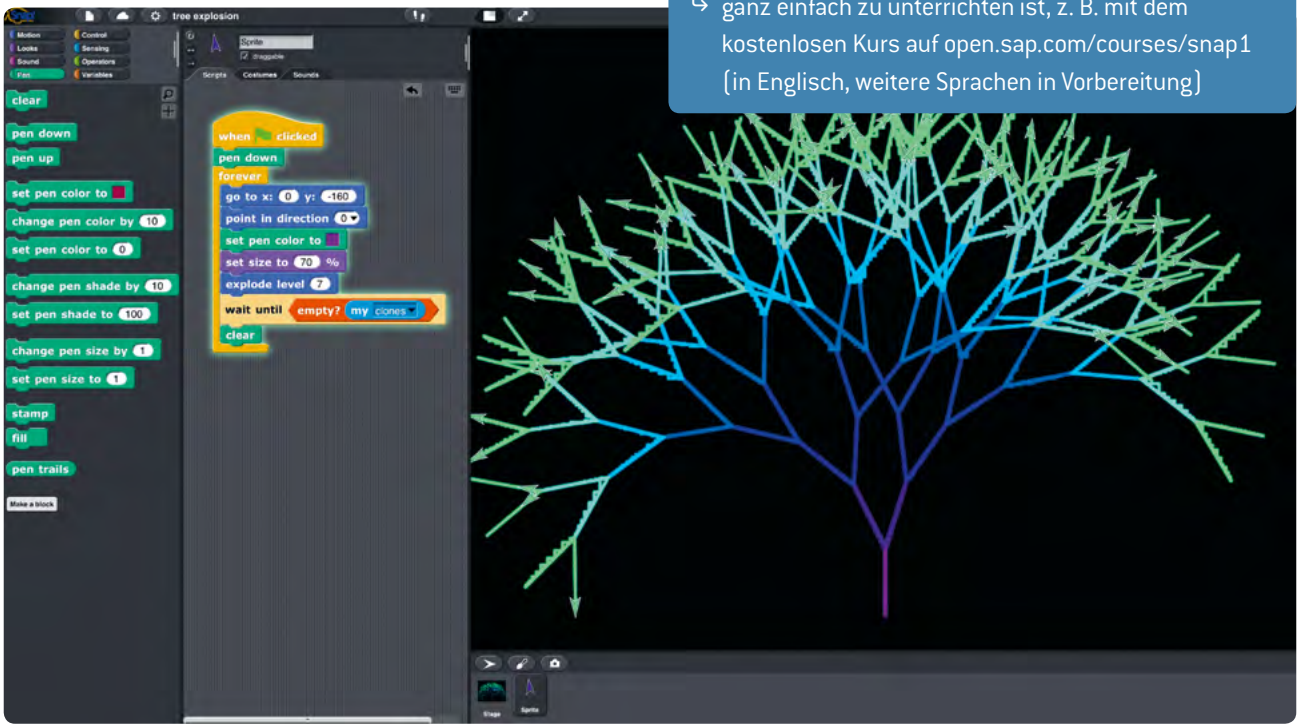
<Was ist Snap!?>

Snap! – Build your own Blocks (Bau deine eigenen Blöcke) ist eine visuelle, blockbasierte Programmiersprache. Sie lädt die Lernenden ein, ihre Ideen zum Leben zu erwecken und sich dabei auf spielerische und experimentierende Weise mit Informatik vertraut zu machen. Das Faszinierende an *Snap!* ist der hohe Anspruch an sich selbst, einen leichten Einstieg zu bieten, ohne dabei die Ausdruckskraft zu mindern. Die Sprache ermöglicht es sowohl Anfängerinnen und Anfängern als auch erfahrenen Programmiererinnen und Programmierern, sich in fortgeschrittene Informatikkonzepte, wie beispielsweise beliebige Datenstrukturen, Funktionen höherer Ordnung und sogar benutzerdefinierte Kontrollstrukturen, in einer visuell ansprechenden und leicht verständlichen Weise zu vertiefen.

Snap! wird gemeinsam mit Forschenden der University of California, Berkeley, von SAP entwickelt, ist heute in über 40 Sprachen erhältlich und wird im Informatikunterricht in ebenso vielen Ländern eingesetzt. *Snap!* ist Open Source und läuft auf allen modernen Webbrowsern.

<Wussten Sie, dass Snap!…>

- eine der 100 Top-Programmiersprachen auf dem TIOBE-Index ist
- im Bereich Künstlicher Intelligenz von Forschenden der University of Oxford eingesetzt wird
- von der Maker-Szene für 3D-Drucke, Stickereien und Robotik verwendet wird
- ganz einfach zu unterrichten ist, z. B. mit dem kostenlosen Kurs auf open.sap.com/courses/snap1 (in Englisch, weitere Sprachen in Vorbereitung)



<Mitmachen bei Meet and Code>

Coding-Event organisieren und Förderung erhalten!

„Ich wusste nicht, dass Programmieren so einfach sein kann!“ Für Philip war die Teilnahme an *Meet and Code* 2018 ein echtes Aha-Erlebnis. „Ich will unbedingt weitermachen“, so Louisa nach der Teilnahme an einem Hackathon. Und genau das will die Initiative auch zeigen: Programmieren macht Spaß und ist einfach zu erlernen.

© Peter Böhmer

Im zweiten Jahr von *Meet and Code* nahmen in über 1.100 Veranstaltungen mehr als 52.000 Jungen und Mädchen in 22 Ländern teil. Wie immer fand die Aktion während der EU Code Week im Oktober statt.

Jede geprüfte und als förderungswürdig bewertete Veranstaltungsidee erhielt ein Startgeld von bis zu 500 Euro. Gefördert werden Programmier-Events aller Art und bewerben kann sich jede gemeinnützige Organisation mit ihrem Projekt.

Und auch 2019 will die Initiative an die Erfolge des letzten Jahres anknüpfen: In 22 Ländern finden zahlreiche Events, Projekte und Workshops während der EU Code Week statt. Mitmachen können Kinder, Jugendliche und junge Erwachsene zwischen 8 und 24 Jahren. Ziel ist es, zu zeigen, wie viel Spaß

© Dietrich Bechtel

Programmieren machen kann und wie man damit eigene Ideen zum Leben erwecken kann. Gleichzeitig können durch das Ausprobieren von eigenen Ideen und die kreative Auseinandersetzung mit Coding, Fähigkeiten entwickelt werden, die in der Arbeitswelt heute und morgen von Bedeutung sind.

Hinter der Initiative *Meet and Code* steht organisatorisch die Haus des Stiftens gGmbH mit dem IT-Portal Stifter-helfen und die jeweiligen Länderpartner des TechSoup Europe-Netzwerkes. Ermöglicht wird *Meet and Code* durch SAP.

Auch 2019 werden wieder Preise für die kreativsten Veranstaltungsideen und originellsten Umsetzungen ausgeschrieben: Der *Meet and Code Award* wird in mindestens drei Kategorien verliehen: u. a. Innovation und Diversity.

Alle Aktionen, Informationen und Anmeldung unter:
www.meet-and-code.org

Folgen Sie uns und reden Sie mit!
@stifter_helfen @TechSoupEurope
#meetandcode #codeEU #SAP4Good

© Dietrich Bechtel

© Peter Böhmer

<Weiterführende Materialien>

Die Autorinnen und Autoren haben weiterführende Materialien erstellt. Diese stehen unter **www.science-on-stage.de/coding-materialien** zum freien Download zur Verfügung.

<Aktivitäten im Rahmen von Coding im MINT-Unterricht>

<2. Juni 2016>
Brainstorming zur Themenfindung während des Abschlusstreffens des Projekts *Fußball im MINT-Unterricht*, Brüssel, Belgien

<20.-22. Oktober 2017>
Erstes Arbeitstreffen im Rahmen der Europäischen *MINT-Liga* im Deutschen Fußballmuseum, Dortmund

<13.-15. April 2018>
Zweites Arbeitstreffen in Berlin

<1. März 2019>
Präsentation der Publikation in Wien, Österreich

<März 2019 - März 2020>
Code League Wettbewerb

<Folgeaktivitäten in 2019 und 2020>
europaweite Lehrerfortbildungen

<Herbst 2020>
Code League Finale

SCIENCE ON STAGE DEUTSCHLAND

Science on Stage Deutschland – The European Network for Science Teachers

… ist ein Netzwerk von Lehrkräften für Lehrkräfte aller Schularten, die Mathematik, Informatik, Naturwissenschaften und Technik (MINT) unterrichten.
… bietet eine Plattform für den europaweiten Austausch anregender Ideen und Konzepte für den Unterricht.
… sorgt dafür, dass MINT im schulischen und öffentlichen Rampenlicht steht.

Science on Stage Deutschland e.V. wird maßgeblich gefördert von think ING., der Initiative für den Ingenieurnachwuchs des Arbeitgeberverbandes GESAMTMETALL.

Machen Sie mit!

www.science-on-stage.de
www.science-on-stage.de/newsletter
www.facebook.com/scienceonstagedeutschland
www.twitter.com/SonS_D

SCIENCE ON STAGE EUROPE

Science on Stage Deutschland e.V. ist Mitglied im Netzwerk Science on Stage Europe e.V.

www.science-on-stage.eu
www.facebook.com/scienceonstageeurope
www.twitter.com/ScienceOnStage

<Weitere Materialien>

Fußball im MINT-Unterricht

↳ Unterrichtseinheiten zu verschiedenen naturwissenschaftlichen Phänomenen im Fußball
↳ Kapitel: Spielfeld, Spieler, Spielball, Spielanalyse

Smartphones im naturwissenschaftlichen Unterricht

↳ Leitfaden und Experimente für forschend-entdeckendes Lernen mit Smartphones

Lilus Haus – Sprachförderung mit Experimenten

↳ Naturwissenschaftliche Entdeckungstour für Grundschülerinnen und -schüler durch Bad, Wohnzimmer und Küche mit Sprach-, Lese- und Schreibtraining

Die Broschüren stehen zum freien Download zur Verfügung unter www.science-on-stage.de/unterrichtsmaterialien

Ein Projekt von

SCIENCE ON STAGE DEUTSCHLAND

Hauptförderer von
Science on Stage Deutschland

think INGO.
Die Initiative für Ingenieurnachwuchs

Mit freundlicher Unterstützung von

SAP®

www.science-on-stage.de