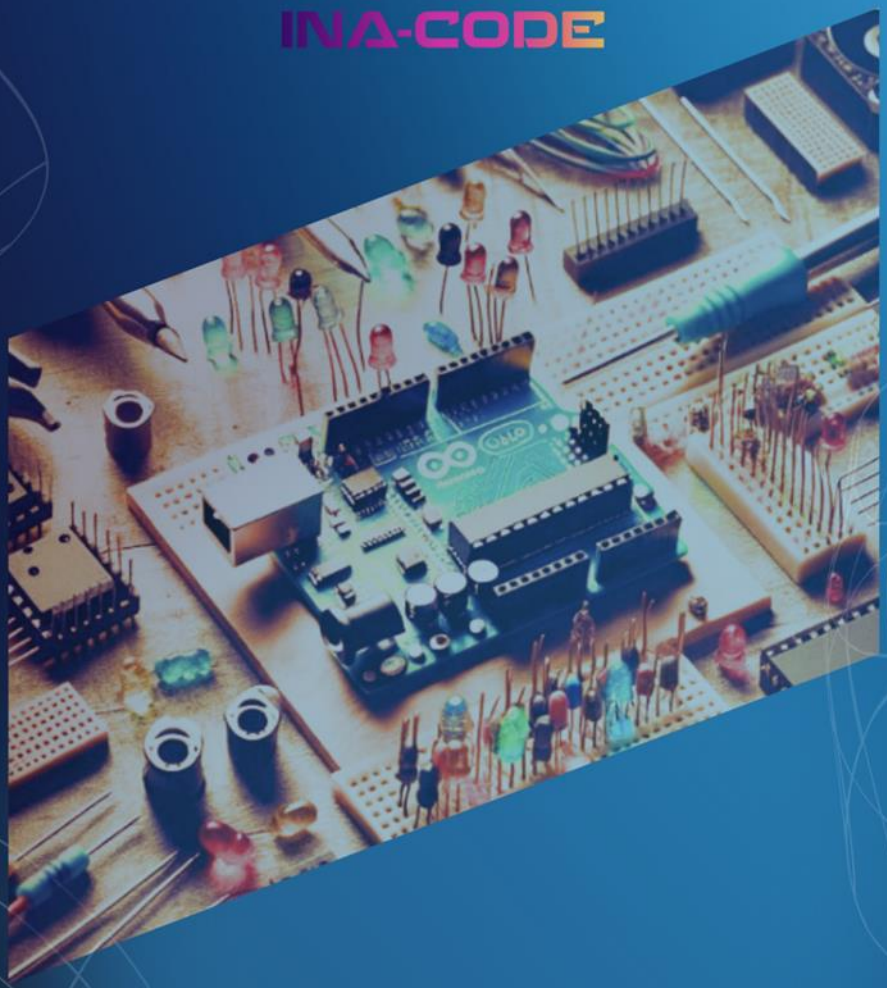
EXPERIMENT BOOKLET

INA-CODE



INNOVATIVE APPROACH FOR CODING IN DIGITAL ERA

EXPERIMENT BOOKLET

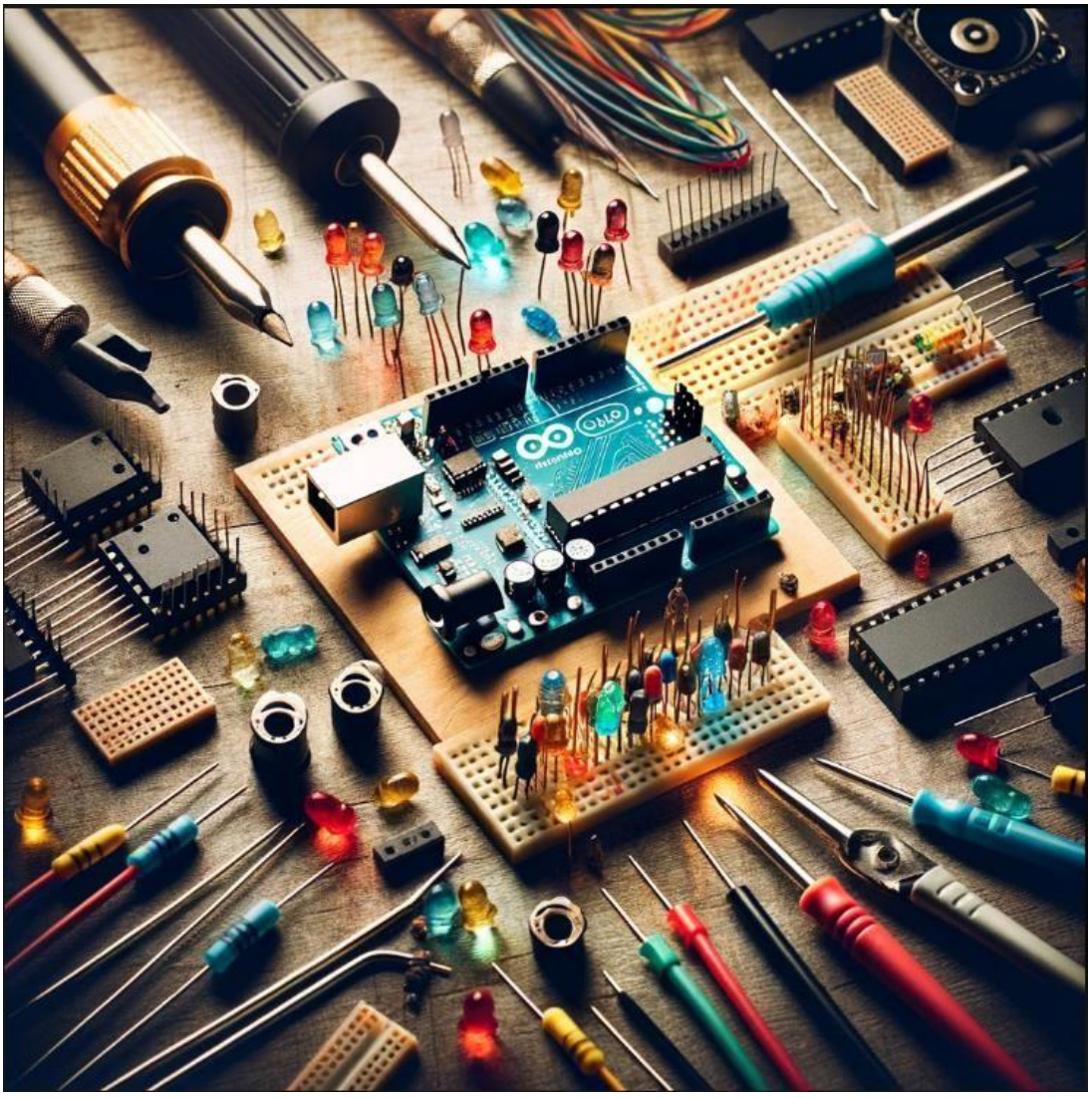
INNOVATIVE APPROACH FOR CODING IN DIGITAL ERA

2021-DE03-KA220-SCH-000024558

Co-funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them

Experiment-Broschüre



Inhaltsübersicht

Vorwort

Tempo und Umfang des digitalen Wandels machen es heute erforderlich, dass sich unsere Bildungssysteme an diese Veränderungen anpassen. In diesem Zusammenhang ist unser Werk "Experiment Booklet: Teacher's Book & Student's Book including 25 Experiments" einen wichtigen Schritt zur Verbesserung der digitalen Kompetenzen von Lehrern und Schülern dar. Diese Broschüre ist Teil einer größeren Initiative, die von der Europäischen Kommission und den nationalen Agenturen unterstützt wird und darauf abzielt, die digitale Kluft zu überbrücken und eine integrativere digitale Bildungslandschaft in ganz Europa zu fördern.

Die Erstellung dieser Broschüre wurde durch die dringende Notwendigkeit motiviert, Pädagogen mit den notwendigen Werkzeugen und Kenntnissen auszustatten, um sich im digitalen Zeitalter zurechtzufinden, insbesondere im Zuge der Herausforderungen, die durch die COVID-19-Pandemie hervorgerufen wurden. Sie dient als Ressource für Lehrkräfte, die Coding und Robotik in ihren Lehrplan integrieren und damit nicht nur ihr berufliches Profil verbessern, sondern auch die Schüler auf die Anforderungen der Arbeitswelt des 21. Jahrhunderts vorbereiten.

Zu unseren Partnern bei diesem Vorhaben gehört ein Konsortium von Schulen, Berufsbildungseinrichtungen und Universitäten in ganz Europa, die alle das gemeinsame Ziel verfolgen, die digitale Bildung voranzubringen. Die Broschüre ist so konzipiert, dass sie praktisch und leicht zugänglich ist und an verschiedene Bildungsumgebungen angepasst werden kann, damit Lehrer und Schüler mit unterschiedlichem Hintergrund davon profitieren können.

Der Zweck dieser Broschüre geht über das reine Unterrichten von Programmierung und Robotik hinaus; sie zielt darauf ab, das kritische Denken, die Kreativität und die Problemlösungsfähigkeiten der Schüler zu fördern. Durch die Einbeziehung dieser Experimente in den Unterricht können Pädagogen eine fesselndere und interaktivere Lernerfahrung bieten und die Schüler dazu ermutigen, die vielfältigen Möglichkeiten der digitalen Technologie zu erkunden.

Zusammenfassend lässt sich sagen, dass diese Broschüre nicht nur ein Lehrmittel ist, sondern ein Katalysator für den Wandel in unseren Bildungssystemen, der eine digital kompetentere und widerstandsfähigere Gesellschaft fördert. Wir sind stolz darauf, einen Beitrag zu diesem Wandel leisten zu können, und hoffen, dass sie Lehrkräfte und Schüler gleichermaßen dazu inspiriert, die Herausforderungen und Chancen des digitalen Zeitalters anzunehmen.

Anzahl der Experimente: 1

1- Batterien

Experiment Zielsetzung

Das Ziel dieses Experiments ist es, zu erklären, wie Batterien funktionieren und welche Arten von Batterien es gibt. Die Schüler lernen, wie sie verschiedene Batterietypen erkennen können.

Theoretischer Hintergrund

Bei der Auswahl einer Batterie sollten Sie auf die Spannung und die Kapazität der Batterie achten. Um diese Begriffe besser zu verstehen, können wir eine Analogie mit Wasser herstellen.

Stellen wir uns vor, wir haben einen Wassertank, der mit Wasser gefüllt ist.

Die Spannung ist der Druck im Wassertank, der das Wasser in die Leitung drückt. Die Höhe des Wassers erhöht den Druck und entspricht der Spannung in elektrischen Schaltkreisen.

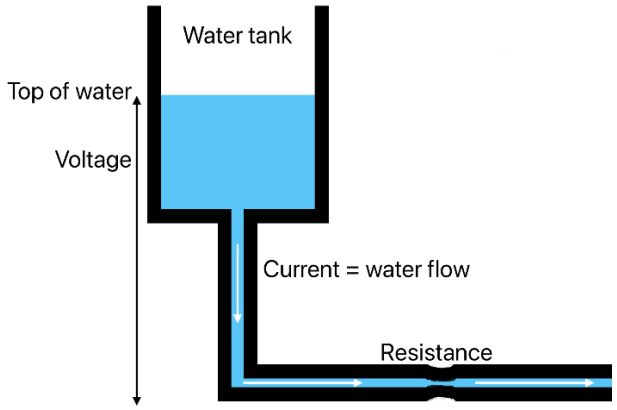
Auf dem Boden eines Tanks befindet sich ein Rohr. Das Wasser, das durch das Rohr fließt, stellt einen elektrischen Strom dar. Je höher der Druck, desto mehr Wasser erhalten wir, d. h. je höher die Spannung, desto höher der Strom.

Ein Hindernis befindet sich am Ende des Rohrs. Es stellt einen Widerstand dar, da der Wasserdurchfluss verringert wird. Je kleiner der Rohrdurchmesser ist (höherer Widerstand), desto geringer ist der Wasserdurchfluss. Dies wird im elektrischen Stromkreis dargestellt: Je größer der Widerstand, desto geringer ist der Strom bei gleicher Spannung.

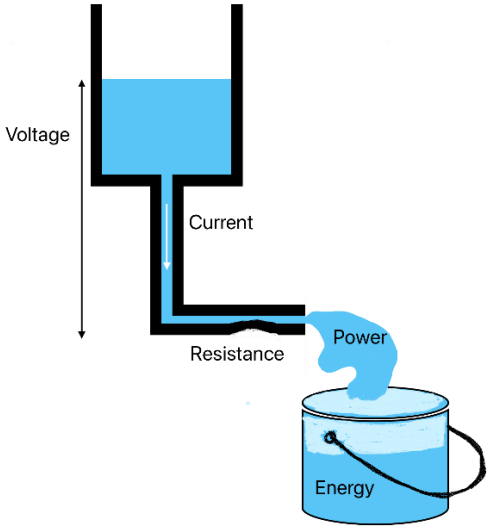
In mathematischen Worten:

$$U = I \cdot R$$

wobei U für Spannung, I für Strom und R für Widerstand steht.



Die Analogie kann sowohl auf Leistung als auch auf Energie ausgedehnt werden. Die Leistung entspricht der Wasserdurchflussmenge. Energie ist die Wassermenge, die im Eimer landet. Die Leistung wird in Watt (W) oder Kilowatt (kW) gemessen, die Energie in Wattstunden (Wh), Milliwattstunden (mWh) oder Kilowattstunden (kWh).



Einige Beispiele für Batterien sind in der Abbildung dargestellt.

Erforderliche Materialien

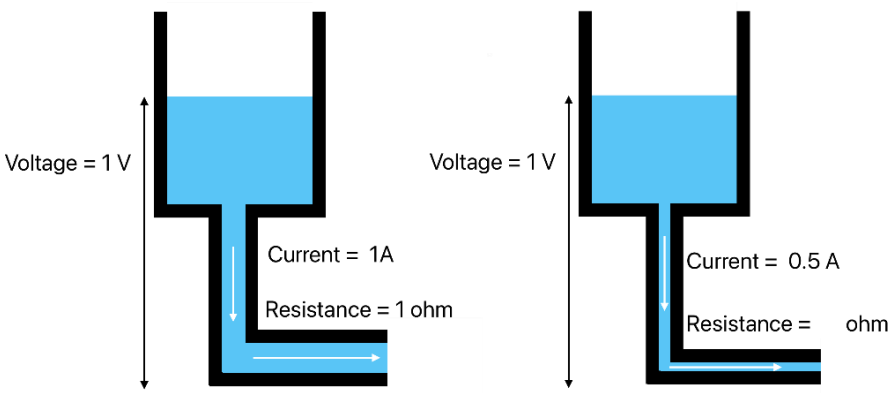
Verschiedene Batterien.

Aufgabe 1.

Suchen Sie verschiedene Batterien in Ihrem Klassenzimmer und bestimmen Sie die Modellbezeichnung (z. B. AA oder AAA) und die Spannung.

Aufgabe 2.

Bestimmen Sie für die in der Abbildung angegebene Spannung und Stromstärke den Widerstand.



Lösung 1.

In der linken Abbildung ist der Widerstand

$$R = \frac{U}{I} = \frac{1\,V}{1A} = 1\Omega$$

In der rechten Abbildung ist der Widerstand

$$R = \frac{U}{I} = \frac{1\,V}{0.5} = 2\Omega$$

Anzahl der Experimente: 2

2- Einführung in die Elektronik: Widerstände, LED, Fotowiderstände und Flexsensor

Experiment Zielsetzung

Ziel dieses Experiments ist es, zu erklären, wie man LED, Widerstände, Fotowiderstände und flexible Sensoren an elektrische Schaltungen anschließt. Die Schüler lernen, diese Elemente sicher zu verwenden, damit sie während des Gebrauchs nicht durchbrennen.

Theoretischer Hintergrund

Eine Leuchtdiode oder LED ist ein elektronisches Halbleiterelement, das Licht aussendet, wenn Strom durch es fließt.

Der kürzere Schenkel der Diode wird Kathode genannt und stellt den - Pol dar. Er ist mit dem - der Batterien oder der Stromversorgung verbunden.

Der längere Schenkel der Diode wird Anode genannt und stellt den Pluspol dar. Er ist mit der + Batterie oder der Stromversorgung verbunden.

Wenn die Diode falsch angeschlossen ist, fließt kein Strom durch sie, und sie leuchtet nicht.

Wenn wir eine LED an einen Stromkreis anschließen, müssen wir den Strom berücksichtigen, der durch diese LED fließt. Wenn zu viel Strom durch die Diode fließt, brennt die Diode durch. Der empfohlene maximale Strom durch die LED beträgt 20 mA.

Um ein Durchbrennen zu verhindern, muss ein Widerstand an den Stromkreis mit der LED angeschlossen werden. Welcher Widerstand zu verwenden ist, wird nach dem Ohmschen Gesetz bestimmt:

$$U = I \cdot R$$

wobei U für Spannung, I für Strom und R für Widerstand steht.

Die LED soll eine Spannung von 3 V haben. Wenn wir eine 9-V-Batterie verwenden, brauchen wir einen Widerstand:

$$R = \frac{U}{I} = \frac{9V - 3V}{20mA} = \frac{6V}{0.02A} = 300\Omega$$

Normalerweise wird ein Widerstand mit Standardwerten verwendet, die dem berechneten Wert am nächsten kommen. In diesem Beispiel ist es ein 330-Ω-Widerstand.

Aufgabe 1

Bestimme, welchen Widerstand wir verwenden sollten, wenn wir die LED an eine 4,5-V-Batterie anschließen.

Lösung 1

Wir müssen sie verwenden:

$$R = \frac{U}{I} = \frac{4.5V - 3V}{20mA} = \frac{1.5V}{0.02A} = 75\Omega$$

Aufgabe 2

Recherchieren Sie den Spannungsabfall bei LEDs anderer Farben im Internet und füllen Sie eine Tabelle.

Farbe der LED	Spannung
Gelb	
Orange	
Rot	
Blau	
Grün	
Violett	

Lösung 2.

Farbe der LED	Spannung
Gelb	1.9-2.1V
Orange	2.0-2.1V
Rot	1.6-2.0V
Blau	2.7-3.2C
Grün	1.9-2.2V
Violett	2.8-4.0V

Aufgabe 3

Probieren Sie den Simulator für elektronische Schaltungen aus: https://www.falstad.com/circuit/e-voltdivide.html Die gezeigte Schaltung hat zwei parallele Zweige. Die Batteriespannung beträgt 10 V.

Im ersten Zweig befinden sich zwei 10 $k\Omega$-Widerstände, also insgesamt 20 $k\Omega$. Der Strom durch diese Widerstände ist $I = \frac{U}{R} = \frac{10V}{20k\Omega} = 0{,}5\,mA$.

Im zweiten Zweig befinden sich vier Widerstände von 10 kΩ, also insgesamt $40k\Omega$. Der Strom durch diese Widerstände ist $I = \frac{U}{R} = \frac{10V}{40k\Omega} = 0{,}25\,mA$.

In der Animation wird der Zweig mit dem höheren Strom durch Punkte dargestellt, die sich schneller bewegen, während die Punkte im Zweig mit dem niedrigeren Strom sich langsamer bewegen.

Bestimmen Sie die Widerstandswerte im anderen Zweig, so dass die Werte des Stroms gleich sind. Testen Sie im Simulator.

Lösung 3

Im zweiten Zweig müssen sich vier Widerstände von 5 $k\Omega$ befinden, also insgesamt 20 $k\Omega$. Der Strom durch Diese Widerstände wären dann $I = \frac{U}{R} = \frac{10V}{20k\Omega} = 0{,}5\,mA$.

Aufgabe 4

Wenn sich im ersten Zweig zwei $10k\Omega$-Widerstände befinden, ist die Spannung zwischen ihnen $\frac{10V}{2} = 5V$.

Wenn sich einer der Widerstände ändert, ändert sich auch die Spannung zwischen ihnen. Wenn der obere Widerstand $10k\Omega$ und der untere $30k\Omega$ ist, beträgt die Spannung zwischen ihnen 7,5 V.

Bestimmen Sie den Wert des unteren Widerstands, so dass die Spannung

zwischen den beiden Widerständen 8 V beträgt. Ein Widerstand, der seinen Wert

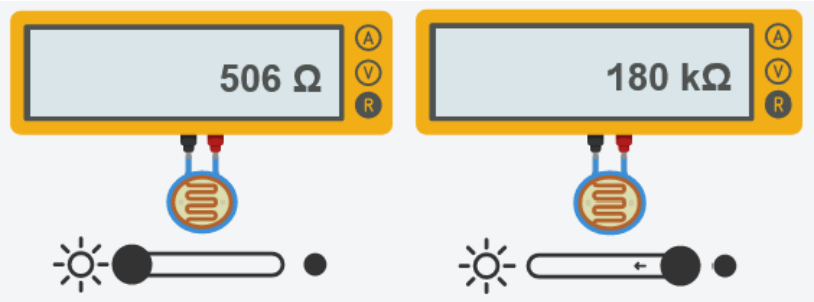
ändert, wird als **Potentiometer bezeichnet**.

Lösung 4

Zusammen mit dem oberen Widerstand $10k\Omega$ sollte der untere Widerstand $40k\Omega$ betragen.

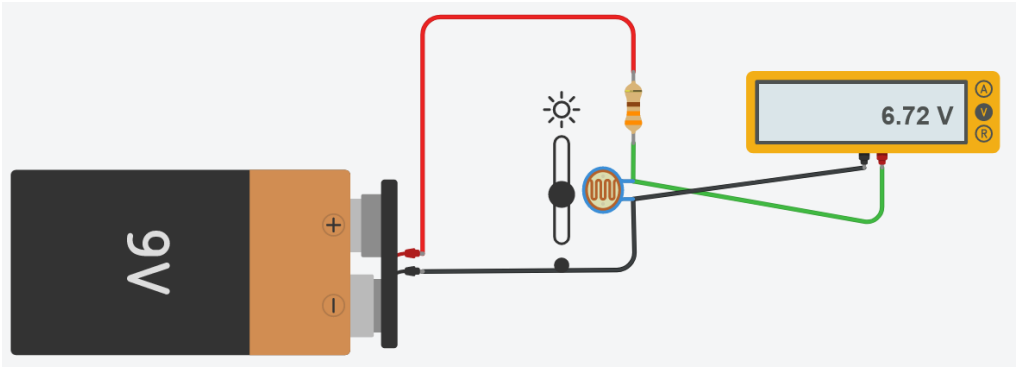
Aufgabe 5

Ein Fotowiderstand ist ein Widerstand, der zur Messung von Licht verwendet wird. Je mehr Licht auf ihn fällt, desto geringer ist sein Widerstand. Je weniger Licht auf ihn fällt, desto höher ist sein Widerstand. Der Fotowiderstand verhält sich wie ein variabler Widerstand,
d.h. ein Potentiometer. Der einzige Unterschied ist, dass wir es nicht mit unseren eigenen Händen einstellen, sondern die Beleuchtung des Raumes.



Wenn sich der Widerstand des Fotowiderstands ändert, ändert sich auch seine Spannung. Je höher der Widerstand des Fotowiderstands ist (kein Licht), desto höher ist auch seine Spannung.

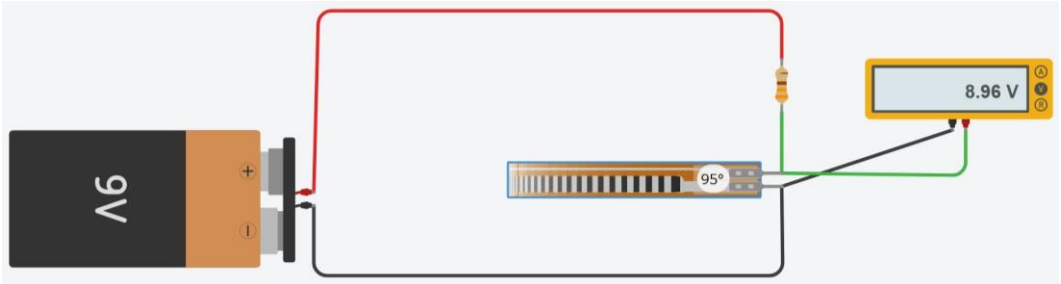
Schließe in TINKERCAD die 9-V-Batterie, den Widerstand, den Fotowiderstand und das Voltmeter gemäß dem Beispiel in der Abbildung an. Ändere das Licht auf dem Fotowiderstand und beobachte, wie sich sein Widerstand und seine Spannung ändern.



Aufgabe 5

Ein Flexsensor ist ein Sensor, der den Grad der Durchbiegung oder Biegung misst. Er wird in der Regel in biotechnologischen Wearables für die Positions- und Bewegungsverfolgung menschlicher Gelenke verwendet.

Schließen Sie den Flexsensor gemäß dem Schema an und messen Sie den Widerstand und die Spannung.



Anzahl der Experimente: 3

3- Einführung in die Elektronik: Kapazität, Induktivität, Relais, Transistor und H-Brücke mit Gleichstrommotor

Experiment Zielsetzung

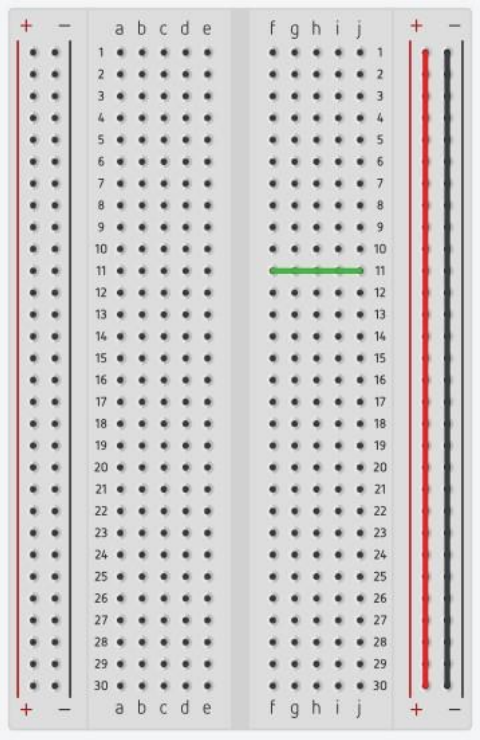
Das Ziel dieses Experiments ist es, zu erklären, wie man Kapazität, Induktivität, Relais, Transistoren und Gleichstrommotoren an elektrische Schaltkreise anschließt. Die Schüler lernen, diese Elemente auf sichere Weise zu verwenden, damit sie während des Gebrauchs nicht durchbrennen.

Theoretischer Hintergrund

Ein **Kondensator** ist ein elektronisches Element, das eine elektrische Ladung speichern kann. Die physikalische Größe, die die Fähigkeit zur Ladungsspeicherung beschreibt, ist die elektrische Kapazität. Die **Kapazität** wird in **Farad (F)** gemessen.
Mit der Erhöhung der Spannung steigt auch die Menge der gespeicherten Ladung. Gleichzeitig ändert sich die Kapazität nicht. Meistens hat der Kondensator zwei parallele leitende Platten, die durch Anschluss an eine Gleichspannungsquelle elektrifiziert werden. Eine Platte ist mit dem Pluspol und die andere mit dem Minuspol der Quelle verbunden.

Wenn ein elektrischer Strom durch die Spule fließt, wird ein Magnetfeld erzeugt. Eine **Spule** ist ein elektronisches Element, das die Energie eines elektromagnetischen Feldes speichert. Wie viel Energie gespeichert wird, hängt von der Induktivität und dem Strom ab, der durch die Spule selbst fließt. Die **Induktivität** der Spule ist eine Größe, die die Fähigkeit beschreibt, die Energie des Magnetfelds zu speichern. Je größer der Strom durch die Spule ist, desto größer ist die magnetische Induktion der Spule. Die Maßeinheit für die Induktivität **ist Henri (H)**.

Protoboard ist eine Bauplatte, die zum Verbinden von Prototypen elektrischer Schaltungen verwendet wird. Fünf Löcher in einer Reihe sind kurzgeschlossen. Wenn man ein elektrisches Element oder einen Draht an zwei von ihnen anschließt, verhalten sie sich so, als wären die Kontakte direkt miteinander verbunden. Das Gleiche gilt für die Spalten unter den + und - Markierungen ganz links und ganz rechts auf der Platine.



Aufgabe 1

Betrachten Sie eine einfache elektrische Schaltung, die nur eine Quelle, einen einzelnen Widerstand, einen **Kondensator** und einen Schalter enthält. Untersuche, was mit dem Strom passiert, wenn der Schalter ein- und ausgeschaltet wird. Schließen Sie eine Glühbirne an den Stromkreis an und beobachten Sie, wann sie aufleuchtet.

Link zum Simulator: https://www.falstad.com/circuit/e-cap.html

Aufgabe 2

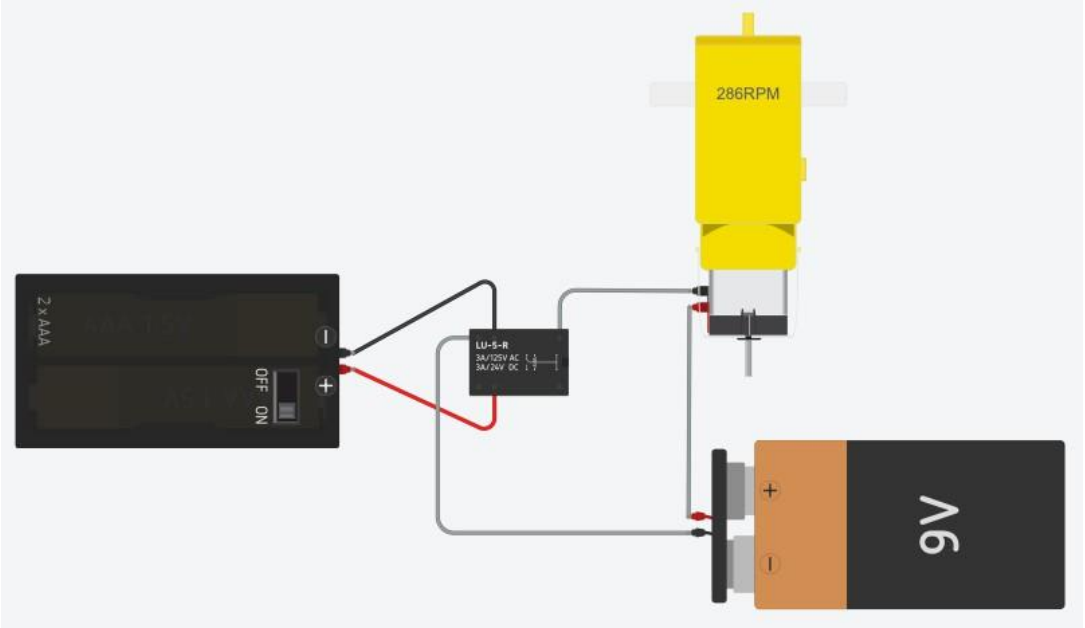
Betrachten Sie eine einfache elektrische Schaltung, die nur eine Quelle, einen einzelnen Widerstand, eine **Induktivität** und einen Schalter enthält. Untersuche, was mit dem Strom passiert, wenn der Schalter ein- und ausgeschaltet wird. Füge eine Glühbirne in den Stromkreis ein und beobachte, wann sie leuchtet.

Link zum Simulator: https://www.falstad.com/circuit/e-induct.html

Aufgabe 3

Ein **Relais** ist ein elektrischer Leistungsschalter. Es wird durch den Strom des ersten Stromkreises aktiviert, um den zweiten Stromkreis ein- oder auszuschalten. Bei der Arbeit mit Relais ist es zwingend erforderlich, zwei getrennte Stromversorgungen für zwei getrennte Stromkreise zu verwenden.

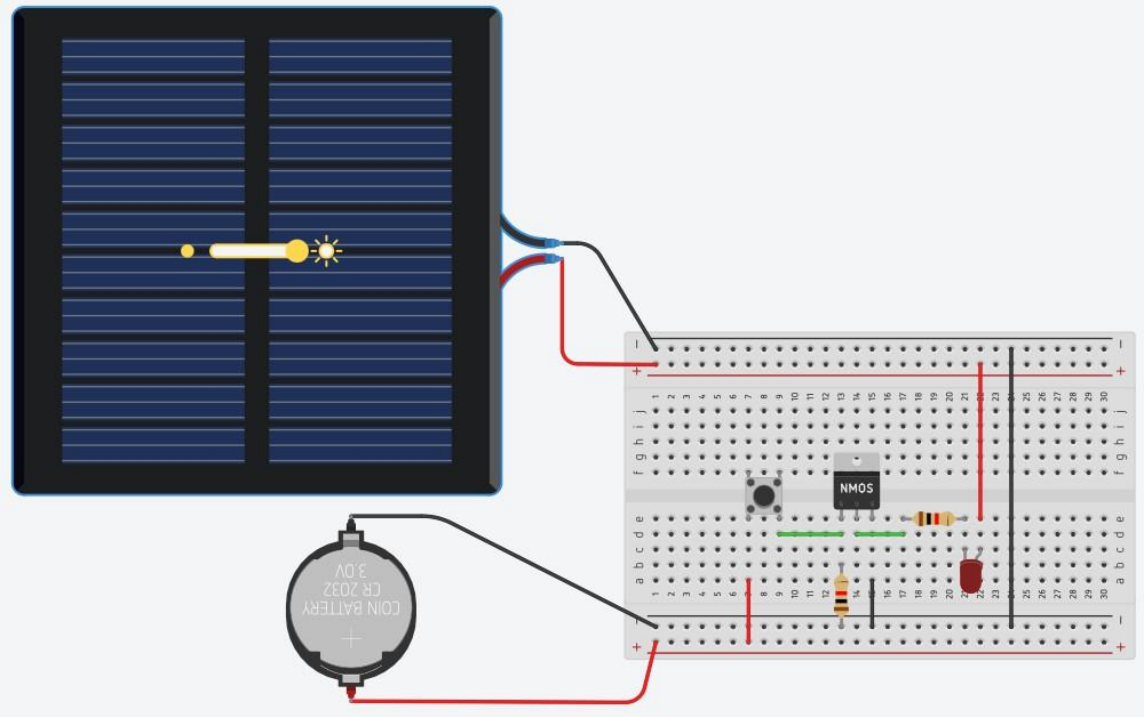
Verbinden Sie das Schema mit dem Relais in TINKERCAD entsprechend der Abbildung und testen Sie, wie das Relais funktioniert.



Aufgabe 4

Ein **NMOS-Transistor** ist ein Halbleiter, der als elektronischer Schalter verwendet wird.

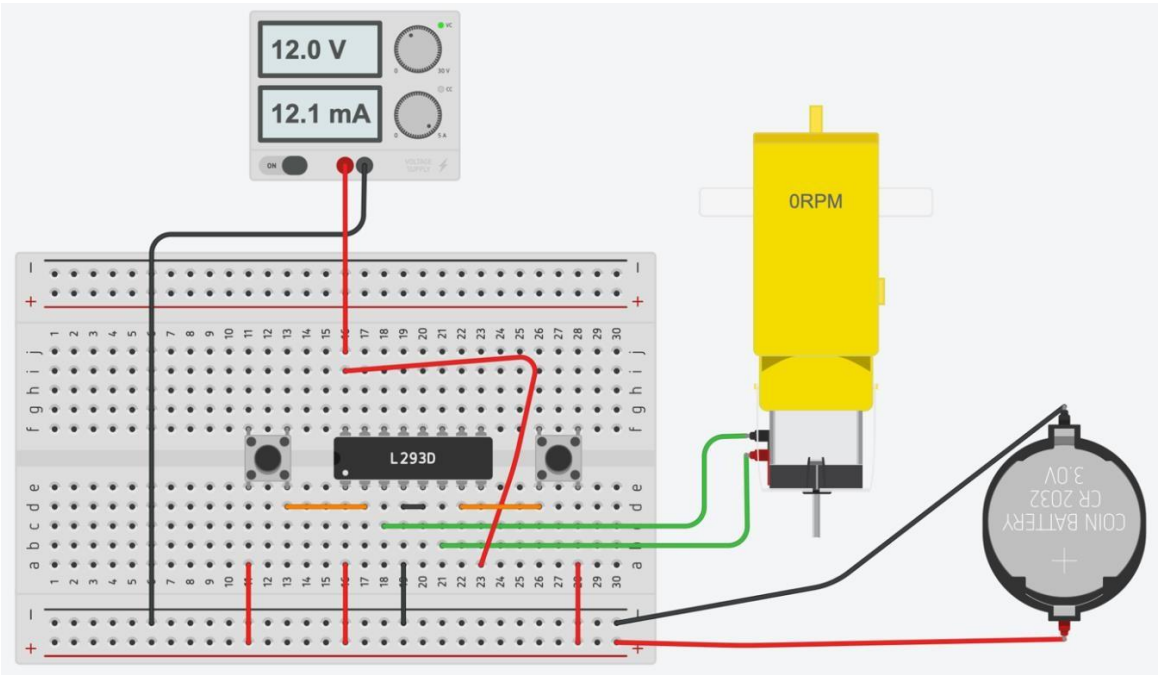
Verbinden Sie das Schema mit NMOS in TINKERCAD entsprechend der Abbildung und testen Sie, wie es funktioniert. Ändere die Werte der Solarzelle und teste sie. Ändern Sie die Werte der Widerstände und testen Sie ti.



Aufgabe 5

Gleichstrommotoren haben zwei Drähte, den negativen (schwarz) und den positiven (rot). Wenn der Pluspol an die Stromversorgung und der Minuspol an GND angeschlossen ist, dreht sich die Motorwelle. Wenn die Drähte umgekehrt angeschlossen werden, d. h. der Pluspol an GND und der Minuspol an die Stromversorgung, dreht sich die Motorwelle erneut, allerdings in die entgegengesetzte Richtung. Um die Drähte nicht jedes Mal manuell anschließen zu müssen, wenn die Richtung geändert wird, wird der H-Brücken-Chip verwendet.

Schließe den Schaltplan mit der H-Brücke und dem Motor in TINKERCAD entsprechend der Abbildung an und teste ihn. Drücken Sie Taster und beobachten Sie, in welche Richtung sich die Motorwelle dreht.



Anzahl der Experimente: 4

4- Einführung in Arduino

Experiment Zielsetzung

Ziel dieses Experiments ist es, zu erklären, was ein Mikrocontroller ist und wofür er gebraucht wird. Die Schüler lernen, den Simulator zu benutzen, um Elektronik mit Arduino zu verbinden.

Theoretischer Hintergrund

Ein Mikrocontroller ist ein kleiner und preiswerter Computer auf einem einzigen Chip, der zur Steuerung der Funktionen von Haushalts- und Bürogeräten, Robotern, Fahrzeugen, Unterhaltungselektronik und vielem mehr verwendet werden kann. Der bekannteste Mikrocontroller ist der Arduino UNO.

Aufgabe 1

Wählen Sie in Tinkercad, Menü Starter, Arduino. Es werden über 30 Demo-Schemata und Programme mit Arduino und Elektronik geöffnet.

Wählen Sie ein Demoprogramm aus und beantworten Sie die Fragen.

1. Welches elektronische Element wird verwendet?
2. An welchem Pin des Arduino ist das Element angeschlossen?
3. Starten Sie die Simulation. Was hat das Programm getan?
4. Kopieren Sie das Schema.
5. Öffnen Sie den Code. Kopieren Sie den verwendeten Code. Versuchen Sie, die gleichen Befehle aus der Block- und Textversion des Programms zu verknüpfen.

Aufgabe 2

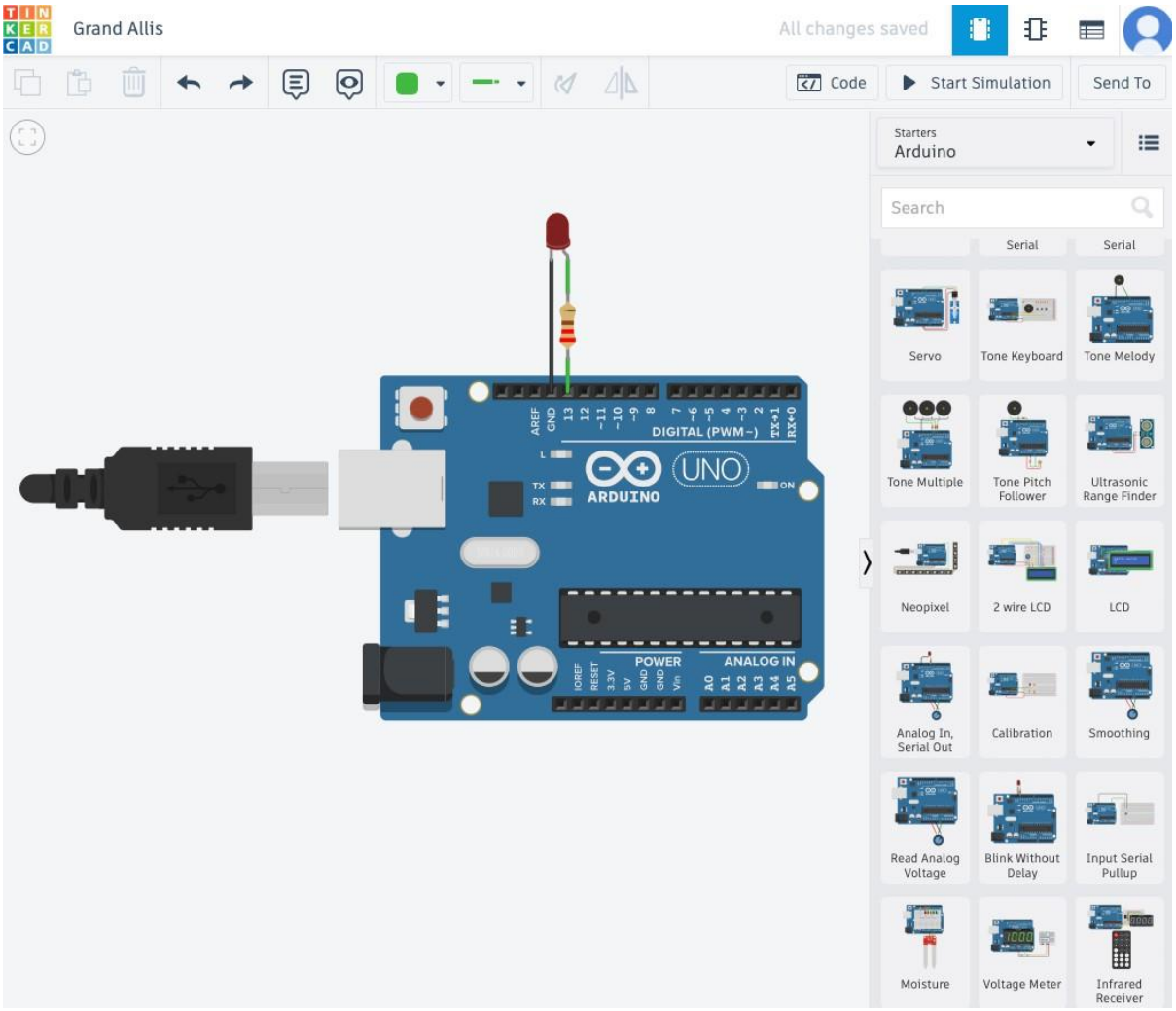
Wählen Sie in Tinkercad, Menü Starter, Arduino. Es werden über 30 Demo-Schemata und Programme mit Arduino und Elektronik geöffnet.

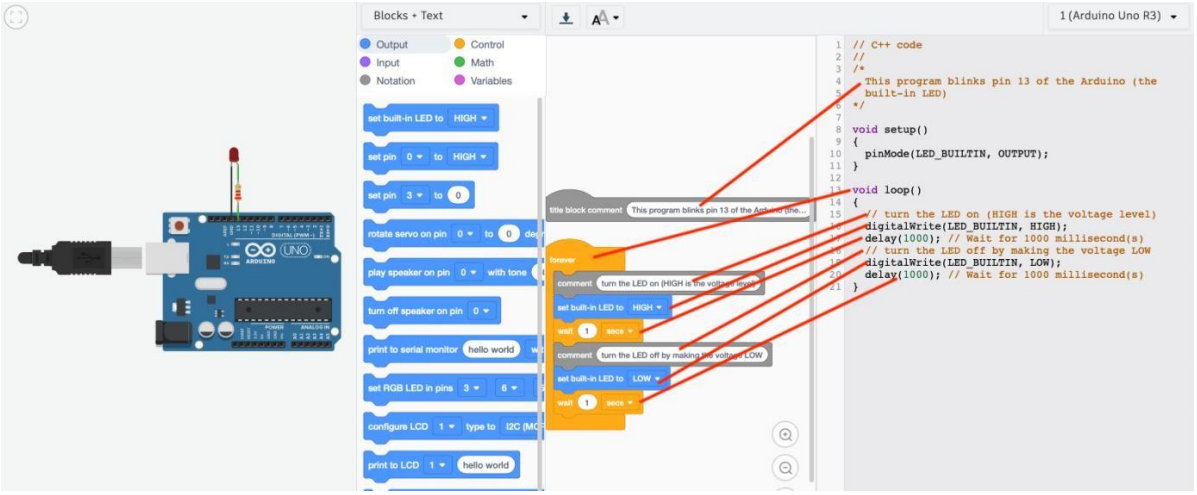
Wählen Sie ein Demoprogramm aus und beantworten Sie die Fragen.

1. Welches elektronische Element wird verwendet?
2. An welchem Pin des Arduino ist das Element angeschlossen?
3. Starten Sie die Simulation. Was hat das Programm getan?
4. Kopieren Sie das Schema.
5. Öffnen Sie den Code. Kopieren Sie den verwendeten Code. Versuchen Sie, die gleichen Befehle aus der Block- und Textversion des Programms zu verknüpfen.

Mögliche Lösung für Aufgabe 1 und Aufgabe 2

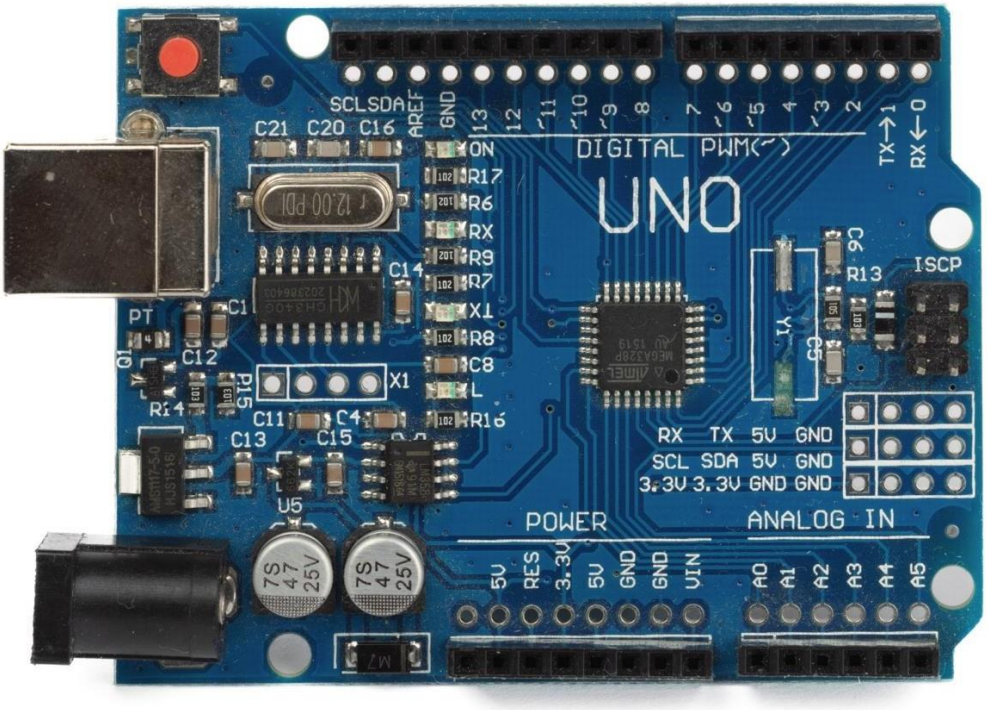
Ein Beispiel ist die Demo Blink. LED und Widerstand sind an Pin 13 angeschlossen. Die LED blinkt alle 1 s.


Grand Allis
All changes saved
Code
Start Simulation
Send To
Starters
Arduino
Search
Serial
Servo
Tone Keyboard
Tone Melody
Tone Multiple
Tone Pitch Follower
Ultrasonic Range Finder
Neopixel
2 wire LCD
LCD
Analog In, Serial Out
Calibration
Smoothing
Read Analog Voltage
Blink Without Delay
Input Serial Pullup
Moisture
Voltage Meter
Infrared Receiver
DIGITAL (PWM~)
POWER
ANALOG IN
UNO
ARDUINO


Blocks + Text
1 (Arduino Uno R3)
Output
Input
Notation
Control
Math
Variables
set built-in LED to HIGH
set pin 0 to HIGH
set pin 3 to 0
rotate servo on pin 0 to 0
play speaker on pin 0 with tone
turn off speaker on pin 0
print to serial monitor hello world
set RGB LED in pins 3 6
configure LCD 1 type to I2C
print to LCD 1 hello world
title block comment This program blinks pin 13 of the Arduino (the...
forever
comment turn the LED on (HIGH is the voltage level)
set built-in LED to HIGH
wait 1 secs
comment turn the LED off by making the voltage LOW
set built-in LED to LOW
wait 1 secs
// C++ code
//
/*
This program blinks pin 13 of the Arduino (the built-in LED)
*/
void setup()
{
pinMode(LED_BUILTIN, OUTPUT);
}
void loop()
{
// turn the LED on (HIGH is the voltage level)
digitalWrite(LED_BUILTIN, HIGH);
delay(1000); // Wait for 1000 millisecond(s)
// turn the LED off by making the voltage LOW
digitalWrite(LED_BUILTIN, LOW);
delay(1000); // Wait for 1000 millisecond(s)
}

Aufgabe 3

Auf dem Foto des Arduino UNO sind die wichtigsten Teile markiert.



5- Arduino-Plattformen

. Was ist Arduino

Experiment Zielsetzung

Ziel ist es, mit Hilfe von Arduino oder einer ähnlichen Technologie eine praktische Anwendung mit Sensoren, Aktoren oder anderen Komponenten vorzustellen. Dieses Projekt soll ein Problem lösen oder eine bestimmte Aufgabe erfüllen und zeigen, wie diese Technologien in realen Szenarien angewendet werden können.

Theoretischer Hintergrund

Zu den grundlegenden Konzepten gehören Elektronik, Programmierung und Systementwurfsprinzipien, die für das Experiment relevant sind. Die Themen können Schaltungstheorie, digitale Logik, Mikrocontroller-Architektur und die Verwendung von C/C++ in der Arduino-Entwicklung umfassen, um ein umfassendes Verständnis zu vermitteln, das für eine erfolgreiche Projektumsetzung erforderlich ist.

Aufgabe 1

Erstellen Sie ein einfaches Temperaturüberwachungssystem mit Hilfe eines Arduino-Boards, um dessen Funktionsweise besser zu verstehen.

Erforderliche Materialien

Arduino UNO R3, Arduino IDE Software oder Tinkercad Simulationsportal.

Experiment Kreislauf

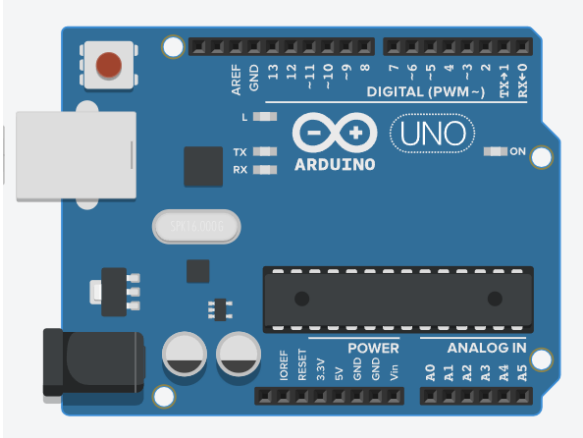


Abbildung 1 Arduino UNO R3 Entwicklungsboard

Schließen Sie das Arduino-Board über ein USB-Kabel vom Typ A/B an den Computer an und öffnen Sie die Arduino-IDE. Wählen Sie den Port (Tools→Port), an den der Arduino angeschlossen ist. Öffnen Sie den Bildschirm Serieller Monitor im Menü Extras.

Lösung: Verfahrensschritte

1- Bauen Sie die Schaltung wie beschrieben auf und stellen Sie sicher, dass alle Verbindungen sicher sind.

2- Schreiben Sie den Code und laden Sie ihn auf Arduino hoch.

3- Beobachten Sie den Bildschirm der seriellen Schnittstelle.

4- Beobachten Sie die Veränderungen auf dem Bildschirm.

Codes

```
// Definieren Sie den Eingangspin, an dem der Temperatursensor
angeschlossen ist int sensorPin = A0;
void setup() {
  Serial.begin(9600); // Start der seriellen Kommunikation
}
void loop() {
  int reading = analogRead(sensorPin); // Lesen des Sensorwertes
  float voltage = reading * 5.0;
  Spannung /= 1024.0; // Diesen Wert in Spannung umrechnen
  float temperatureC = (Spannung - 0,5) * 100; // Umrechnung der Spannung in die Temperatur in Celsius
  Serial.print("Temperatur: ");
  Serial.print(TemperaturC);
  Serial.println(" C");
    delay(1000); // Eine Sekunde warten, bevor die Schleife wiederholt wird
}
```

Erläuterung des Kodex

Der Code liest den Ausgang des Temperatursensors auf einem Arduino, wandelt ihn in Spannung um, berechnet die Temperatur in Celsius und zeigt sie kontinuierlich auf dem seriellen Monitor an.

6- Arduino Programmiersprache und Editor-1

Experiment Zielsetzung

Das Ziel des Experiments mit Arduino ist es, ein einfaches Programm zu schreiben, das verschiedene LEDs steuert und sie in bestimmten Intervallen nacheinander ein- und ausschaltet. Dieses Experiment zielt darauf ab, die grundlegenden Programmierkonstrukte von Arduino zu lehren und wie man mit der physischen Welt durch Programmierung interagiert.

Theoretischer Hintergrund

Die Programmiersprache, die in Arduino verwendet wird, basiert auf Wiring, einer angepassten Version von C++, die für die einfache Verwendung bei allgemeinen Eingabe-/Ausgabeoperationen entwickelt wurde. Zu den wichtigsten Merkmalen dieser Programmiersprache gehören objektorientierte Programmierung (OOP), Portabilität, hohe Leistung, Standardbibliothek, Vielseitigkeit, Multi-Paradigmen-Programmierung und Speicherverwaltung. Die Arduino-Programmierung dreht sich um die Funktionen setup() und loop(). setup() wird einmal zu Beginn des Programms aufgerufen, um Konfigurationen wie Pin-Modi einzurichten, während loop() die Hauptprogrammlogik enthält, die sich im Laufe der Zeit wiederholt und Hardware wie LEDs basierend auf der Logik des Programms steuert.

Erforderliche Materialien

Arduino UNO R3, Arduino IDE Software oder Tinkercad Simulationsportal.

Experiment Kreislauf

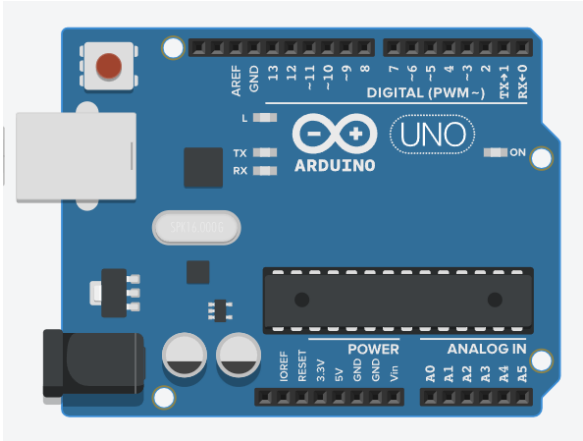


Abbildung 2 Entwicklungsplatine Arduino UNO R3

Schließen Sie das Arduino-Board über ein USB-Kabel vom Typ A/B an den Computer an und öffnen Sie die Arduino-IDE. Wählen Sie den Port (Tools→Port), an den das Arduino angeschlossen ist. Öffnen Sie den Bildschirm Serieller Monitor im Menü Extras.

Verfahrensschritte

1- Bauen Sie die Schaltung wie beschrieben auf und stellen Sie sicher, dass alle Verbindungen sicher sind.

2- Schreiben Sie den Code und laden Sie ihn auf Arduino hoch.

3- Beobachten Sie den Bildschirm der seriellen Schnittstelle.

4- Beobachten Sie die Veränderungen auf dem Bildschirm.

Codes

Erstes Code-Beispiel: Led-Steuerung

```
int led1 = 4, led2 = 5, led3 = 6, led4 = 7;
void setup() {
  Serial.begin(9600);
  pinMode(led1, OUTPUT);
  pinMode(led2, OUTPUT);
  pinMode(led3, OUTPUT);
  pinMode(led4, OUTPUT);
}
void loop() {
  digitalWrite(led1, HIGH);
  delay(1000); // Warten auf 1
 Sekunde digitalWrite(led1, LOW);
 delay(500); // Warten auf 0,5
 Sekunden digitalWrite(led2, HIGH);
 delay(1000);
  digitalWrite(led2, LOW);
 delay(500);
 digitalWrite(led3, HIGH);
 delay(1000);
 digitalWrite(led3, LOW);
 delay(500);
 digitalWrite(led4, HIGH);
 delay(1000);
 digitalWrite(led4, LOW);
 delay(500);
}
```

Erläuterung des Kodex

Das Codebeispiel demonstriert, wie vier an einen Arduino angeschlossene LEDs nacheinander ein- und ausgeschaltet werden können, um ein einfaches visuelles Muster zu erzeugen. Es nutzt grundlegende Funktionen wie pinMode(), digitalWrite() und delay(), um das Timing und den Zustand jeder LED zu steuern.

Zweites Code-Beispiel: Sensorwert lesen

```
void setup() {
  Serial.begin(9600);
}

void loop() {
  int sensorValue = analogRead(A0);
  Serial.println(sensorValue);
  delay(1); // Kurze Verzögerung zur besseren Lesbarkeit
}
```

Erläuterung des Kodex

Das Codebeispiel konzentriert sich auf das Lesen von Werten von einem Sensor, der an den analogen Pin A0 des Arduino angeschlossen ist, und das Drucken dieser Werte auf dem seriellen Monitor. Es zeigt, wie analogRead() und Serial.println() für die Erfassung und Protokollierung von Sensordaten verwendet werden können.

7- Arduino Programmiersprache und Editor-2

Experiment Zielsetzung

Ziel ist es, die Fähigkeiten von Arduino bei der Verbindung mit verschiedenen Sensoren und Komponenten zu erkunden. Dies beinhaltet die Erstellung von Projekten, die zeigen, wie Arduino für praktische Anwendungen wie Umweltüberwachung, Robotik und interaktive Kunstinstallationen eingesetzt werden kann.

Theoretischer Hintergrund

Zu den theoretischen Grundlagen gehört das Verständnis der Prinzipien von Mikrocontrollern, insbesondere der Architektur und Programmierung von Arduino. Es werden digitale und analoge Ein- und Ausgänge, die Grundlagen von Elektrizität und Schaltkreisen, Programmierkonstrukte (Variablen, Schleifen, Konditionale) und Kommunikationsprotokolle wie Serial und I2C behandelt. Der Hintergrund berührt auch die Bedeutung von Open-Source-Hardware und -Software für die Förderung von Innovation und Ausbildung in Elektronik und Programmierung.

Erforderliche Materialien

Arduino UNO R3, Arduino IDE Software oder Tinkercad Simulationsportal.

Experiment Kreislauf

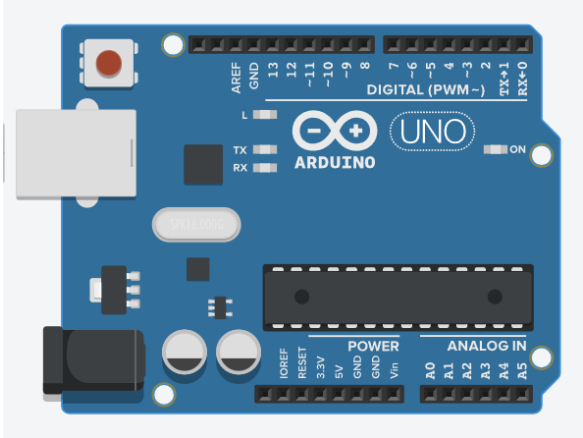


Abbildung 3 Entwicklungsplatine Arduino UNO R3

Schließen Sie das Arduino-Board über ein USB-Kabel vom Typ A/B an den Computer an und öffnen Sie die Arduino-IDE. Wählen Sie den Port (Tools→Port), an den der Arduino angeschlossen ist. Öffnen Sie den Bildschirm Serieller Monitor im Menü Extras.

Aufgabe 1.

Erstellen Sie ein Arduino-basiertes Projekt, das die Prinzipien des Designs elektronischer Schaltungen und der Programmierung mit der Arduino IDE demonstriert.

Lösung 1.

Entwurf eines Schaltkreises, der mit verschiedenen Sensoren und Aktoren interagiert, die mit Arduino programmiert werden, um bestimmte Funktionen auszuführen, wie z. B. die Messung von Umgebungsbedingungen oder die Steuerung von Lampen oder Motoren.

Anzahl der Experimente: 8

8- Befehle für mathematische Operationen in der Arduino IDE: Wie man arithmetische Operatoren verwendet Experiment

Experiment Zielsetzung

Das Ziel dieses Experiments ist es, zu demonstrieren, wie grundlegende arithmetische Operatoren wie Addition, Subtraktion, Multiplikation, Division und Rest in der integrierten Entwicklungsumgebung (IDE) von Arduino verwendet werden. Die SchülerInnen lernen, diese Operatoren zu verwenden und das Ergebnis auf dem Bildschirm der seriellen Schnittstelle der Arduino IDE auszudrucken.

Theoretischer Hintergrund

Alle Programmiersprachen enthalten einige arithmetische Operatoren, und auch die Arduino IDE enthält grundlegende arithmetische Operatoren. Diese Operatoren können in Arduino verwendet werden, um einige mathematische Operationen auszuführen, wie z.B. die Verarbeitung von Sensordaten. Es gibt 5 grundlegende arithmetische Operatoren, die in dieser Anwendung gezeigt werden.

Betreiber	Betreiber Symbol	Variablen	Beispiel	Ergebnis
Zusatz	+	Sen1 = 21	Ergebnis = Sen1 + Sen2	24
Subtraktion	-	Sen2 = 3	Ergebnis = Sen1 - Sen2	18
Multiplikation	*	Ergebnis = 0	Ergebnis = Sen1 * Sen2	63
Abteilung	/		Ergebnis = Sen1 / Sen2	7
Restbetrag	%		Ergebnis = Sen1 % Sen2	0

Erforderliche Materialien

Arduino UNO R3, Arduino IDE Software oder Tinkercad Simulationsportal.

Experiment Kreislauf

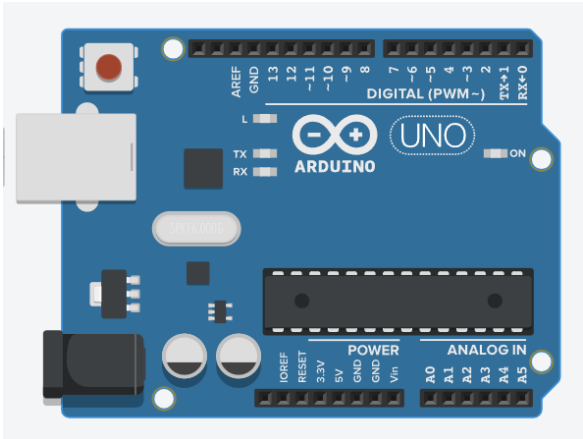


Abbildung 4 Entwicklungsplatine Arduino UNO R3

Schließen Sie das Arduino-Board über ein USB-Kabel vom Typ A/B an den Computer an und öffnen Sie die Arduino-IDE. Wählen Sie den Port (Tools→Port), an den der Arduino angeschlossen ist. Öffnen Sie den Bildschirm Serieller Monitor im Menü Extras.

Verfahrensschritte

1-Bauen Sie die Schaltung wie beschrieben auf und stellen Sie sicher, dass alle Verbindungen sicher sind.

2-Schreiben Sie den Code und laden Sie ihn auf Arduino hoch.

3-Beobachten Sie den Bildschirm der seriellen Schnittstelle.

4-Beobachten Sie die Veränderungen auf dem Bildschirm.

Codes

```
/* Definition der globalen
Variablen */ int sensor1 = 21;
int sensor2 = 3;
int Ergebnis = 0;
void setup(){
 Serial.begin(9600);
 //Zusatz
 Ergebnis = sensor1 + sensor2;
 Serial.print("Addition von Sensor1 und Sensor2 ist ");
 Serial.println(Ergebnis);
 Verzögerung(2000);
 //Subtraktion
 Ergebnis = Sensor1 - Sensor2;
 Serial.print("Subtraktion von Sensor1 und Sensor2 ist ");
 Serial.println(Ergebnis);
 Verzögerung(2000);
 //Multiplikation
 Ergebnis = sensor1 * sensor2;
 Serial.print("Die Multiplikation von Sensor1 und Sensor2 ist ");
 Serial.println(Ergebnis);
 Verzögerung(2000);
 //Abteilung
 Ergebnis = Sensor1 / Sensor2;
 Serial.print("Die Division von Sensor1 und Sensor2 ist ");
 Serial.println(Ergebnis);
 Verzögerung(2000);
 //Remainder
 Ergebnis = sensor1 % sensor2; Serial.print("Der
 Rest von sensor1 und sensor2 ist ");
 Serial.println(Ergebnis);
 Verzögerung(2000);
}

void loop() {
}
```

Erläuterung des Kodex

Einrichtung: Der serielle Monitor ist mit einer Baudrate von 9600 aktiviert. Für zuvor definierte Variablen werden die entsprechenden mathematischen Operationen nur einmal durchgeführt.

Schleife: -

Anzahl der Experimente: 9

9- Befehle für mathematische Operationen in der Arduino IDE: Wie man arithmetische Operatoren verwendet Experiment 2

Experiment Zielsetzung

Das Ziel dieses Experiments ist es, zu demonstrieren, wie grundlegende arithmetische Operatoren wie Addition, Subtraktion, Multiplikation, Division und Rest in der integrierten Entwicklungsumgebung (IDE) von Arduino verwendet werden. Die Schüler lernen, diese Operatoren mit Fließkomma-Variablen zu verwenden.

Theoretischer Hintergrund

Das Verhalten der arithmetischen Operatoren hängt von der Art der verwendeten Zahlenvariablen ab. Insbesondere der Divisionsoperator kann ein Fließkommaergebnis erzeugen. Wenn der Divisionsoperator auf Integer-Variablen angewendet wird, erhält man daher möglicherweise nicht das erwartete Ergebnis. Bei Sensoranwendungen mit Arduino-Boards sind die verarbeiteten Zahlen meist vom Typ Fließkomma. Darüber hinaus ist der Modulo-Operator (Rest) ein Operator, der nur mit ganzen Zahlen verwendet werden kann.

Betreiber	Betreiber Symbol	Variablen	Beispiel	Ergebnis
Zusatz	+	Sen1 = 24,5	Ergebnis = Sen1 + Sen2	28.00
Subtraktion	-	Sen2 = 3,5	Ergebnis = Sen1 - Sen2	21.00
Multiplikation	*	Ergebnis = 0,0	Ergebnis = Sen1 * Sen2	85.75
Abteilung	/	Sen3 = 12	Ergebnis = Sen1 / Sen2	7.00
Restbetrag	%	Sen4 = 5	Ergebnis = Sen3 % Sen4	2.00

Erforderliche Materialien

Arduino UNO R3, Arduino IDE Software oder Tinkercad Simulationsportal.

Experiment Kreislauf

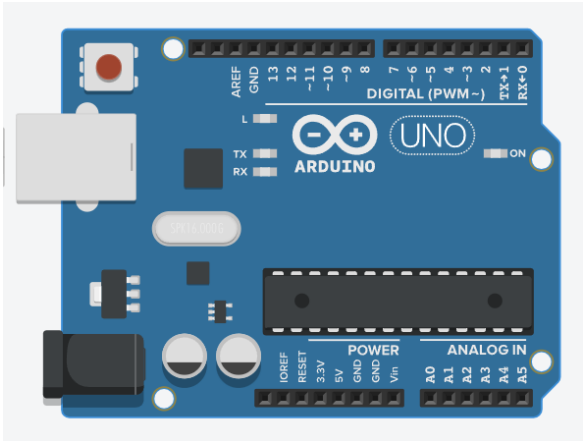


Abbildung 5 Entwicklungsplatine Arduino UNO R3

Schließen Sie das Arduino-Board über ein USB-Kabel vom Typ A/B an den Computer an und öffnen Sie die Arduino-IDE. Wählen Sie den Port (Tools→Port), an den der Arduino angeschlossen ist. Öffnen Sie den Bildschirm Serieller Monitor im Menü Extras.

Verfahrensschritte

1- Bauen Sie die Schaltung wie beschrieben auf und stellen Sie sicher, dass alle Verbindungen sicher sind.

2- Schreiben Sie den Code und laden Sie ihn auf Arduino hoch.

3- Beobachten Sie den Bildschirm der seriellen Schnittstelle.

4- Beobachten Sie die Veränderungen auf dem Bildschirm.

Codes

```
//Globale Variable
float sensor1 = 24.5;
float sensor2 = 3.5;
float Ergebnis = 0.0;
int seed = 123;
void setup()
{
  Serial.begin(9600);

  //Zugabevorgang
  Ergebnis = sensor1 + sensor2;
  Serial.print("Addition von Sensor1 und Sensor2 ist ");
  Serial.println(Ergebnis);
  Verzögerung(2000);

  //Subtraktionsvorgang
  Ergebnis = sensor1 -
  sensor2;
  Serial.print("Die Subtraktion von Sensor1 und Sensor2 ist ");
  Serial.println(Ergebnis);
  Verzögerung(2000);

  //Multipliziervorgang Ergebnis
  = sensor1 * sensor2;
  Serial.print("Die Multiplikation von Sensor1 und Sensor2 ist ");
  Serial.println(Ergebnis);
  Verzögerung(2000);

  //Teilungsvorgang
  Ergebnis = Sensor1 / Sensor2;
  Serial.print("Die Teilung von Sensor1 und Sensor2 ist ");
  Serial.println(Ergebnis);
  Verzögerung(2000);

  int sensor3 = 12, sensor4 = 5;
  //Remindernde (modulo) Operation
  Ergebnis = sensor3 % sensor4;
  Serial.print("Der Rest von Sensor3 und Sensor4 ist ");
  Serial.println(Ergebnis);
  Verzögerung(2000);

}

void loop()
{randomSeed(seed); sensor1 = random(1, 300);
```

```
  sensor2 = random(1, 300);
  Serial.println(sensor1);
  Serial.println(sensor2); seed
  = sensor1 * sensor2;
  Ergebnis = sensor1 +
  sensor2;
  Serial.print("Addition der Sensordaten = ");
  Serial.println(Ergebnis);
  Ergebnis = sensor1 - sensor2;
  Serial.print("Subtraktion der Sensordaten = ");
  Serial.println(Ergebnis);
  Ergebnis = sensor1 * sensor2;
  Serial.print("Multiplikation der Sensordaten =
  "); Serial.println(Ergebnis);
  Ergebnis = sensor1 / sensor2;
  Serial.print("Division der Sensordaten = ");
  Serial.println(Ergebnis);
  Ergebnis = (int)sensor1 % (int)sensor2;
  Serial.print("Rest der Sensordaten = ");
  Serial.println(Ergebnis);
}
```

Erläuterung des Kodex

Einrichtung: Der serielle Monitor ist mit einer Baudrate von 9600 aktiviert. Für zuvor definierte Variablen werden die entsprechenden mathematischen Operationen nur einmal durchgeführt.

Schleife: Ein einfaches Simulationsspiel wird entworfen. Dazu werden die eingebauten Methoden random() und randomSeed() des Arduino-Editors verwendet. Wenn jede Schleifenfunktion abläuft, werden die Daten von Sensor 1 und Sensor 2 in einem bestimmten Intervall erneut zufällig erzeugt. Bei jedem Vorgang werden wieder 5 mathematische Operationen durchgeführt.

Anzahl der Experimente: 10

10- Kontrollstrukturen in der Arduino IDE: Wie man Kontrollstrukturen verwendet-1

Experiment Zielsetzung

Das Ziel dieses Experiments ist es, zu demonstrieren, wie Kontrollstrukturen in der integrierten Entwicklungsumgebung (IDE) von Arduino verwendet werden. Die Schüler lernen den Umgang mit **Vergleichsoperatoren** und ***if-Anweisungen*** ("*if*", "*if ... else*", "*if ... else if ... else*").

Theoretischer Hintergrund

Alle Programmiersprachen verfügen über einige Arten von Steueranweisungen, um eine Entscheidung über bestimmte Bedingungen zu treffen. Wir verwenden diese Steueranweisungen zum Beispiel, wenn wir über die auszuführende Operation auf der Grundlage eines bestimmten Temperaturwerts entscheiden oder wenn wir über die auszuführende Operation auf der Grundlage der Note eines Schülers entscheiden.

Wie in anderen Sprachen auch, gibt es verschiedene Kontrollstrukturen, die wir in der Arduino IDE verwenden. Diese Kontrollstrukturen und Anweisungen können wie folgt aufgelistet werden:

Kontrollanweisung	Beschreibung
"if"-Anweisung	Er nimmt einen Ausdruck in Klammern und eine Anweisung oder einen Block von Anweisungen. Wenn der Ausdruck wahr ist, wird die Anweisung oder der Anweisungsblock ausgeführt, andernfalls werden diese Anweisungen übersprungen.
"if ... else"-Anweisung	Eine if-Anweisung kann von einer optionalen else-Anweisung gefolgt werden, die ausgeführt wird, wenn der Ausdruck falsch ist.
"if ... else if ... else"-Anweisung	Auf die if-Anweisung kann eine optionale **else** if-Anweisung folgen, mit der verschiedene Bedingungen getestet werden können.
"switch case"-Anweisung	Ähnlich wie bei den if-Anweisungen steuert **switch...case** den Fluss von Programme, indem die Programmierer verschiedene Blöcke angeben können, die unter verschiedenen Bedingungen ausgeführt werden sollen.
Bedingter Operator "?:"	I enutzer ermöglicht, einen V einer Zeile durchzuführen.

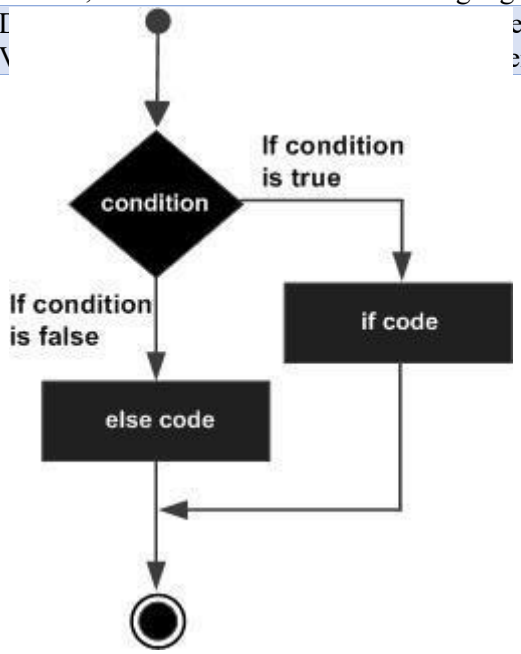


Abbildung 6 Flussdiagramm der if-Anweisungen

Einige Vergleichsoperatoren werden verwendet, um zwei Werte zu vergleichen, wenn Steueranweisungen geschrieben werden. Als Ergebnis des Vergleichs erhält man eine logische Ausgabe von **wahr (1)** oder **falsch (0)**. Angenommen, die Variable portB hat den Wert 63 und portC den Wert 127,

Betreiber Name	Betreiber Symbol	Beschreibung	Beispiel
gleich	= =	Prüft, ob der Wert von zwei Operanden gleich ist oder nicht; wenn ja, wird die Bedingung erfüllt.	(AnschlussB == AnschlussC) ist **falsch**
nicht gleich	! =	Prüft, ob der Wert von zwei Operanden gleich ist oder nicht; wenn die Werte nicht gleich sind, wird die Bedingung wahr.	(portB != portC) ist **wahr**
weniger als	<	Prüft, ob der Wert des linken Operanden kleiner ist als der Wert des rechten Operanden, wenn ja, dann Bedingung wahr wird.	(AnschlussB < AnschlussC) ist **wahr**
größer als	>	Prüft, ob der Wert des linken Operanden größer ist als der Wert des rechten Operanden, wenn ja, dann wird die Bedingung wahr.	(AnschlussB > AnschlussC) ist **falsch**
kleiner als oder gleich	< =	Prüft, ob der Wert des linken Operanden kleiner ist als oder gleich dem Wert des rechten Operanden, wenn ja, wird die Bedingung erfüllt.	(AnschlussB <= AnschlussC) ist **wahr**
größer als oder gleich	> =	Prüft, ob der Wert des linken Operanden größer oder gleich dem Wert des rechten Operanden ist, wenn ja, dann wird die Bedingung erfüllt.	(AnschlussB >= AnschlussC) ist **falsch**

Erforderliche Materialien

Arduino UNO R3, Arduino IDE Software oder Tinkercad Simulationsportal.

Experiment Kreislauf

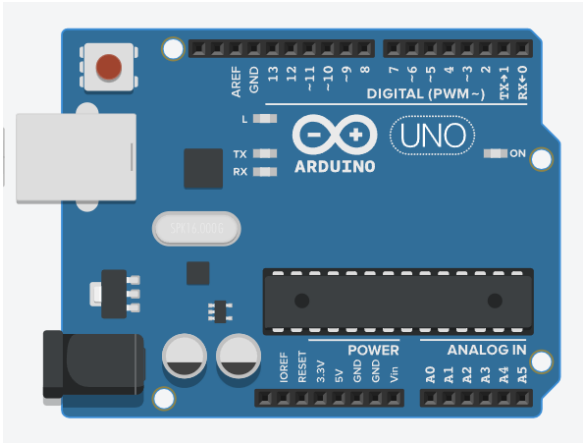


Abbildung 7 Entwicklungsplatine Arduino UNO R3

Schließen Sie das Arduino-Board über ein USB-Kabel vom Typ A/B an den Computer an und öffnen Sie die Arduino-IDE. Wählen Sie den Port (Tools→Port), an den das Arduino angeschlossen ist. Öffnen Sie den Bildschirm Serieller Monitor im Menü Extras.

Verfahrensschritte

1- Bauen Sie die Schaltung wie beschrieben auf und stellen Sie sicher, dass alle Verbindungen sicher sind.

2- Schreiben Sie den Code und laden Sie ihn auf Arduino hoch.

3- Beobachten Sie den Bildschirm der seriellen Schnittstelle.

4- Beobachten Sie die Veränderungen auf dem Bildschirm.

Codes

```
/* Definition der globalen
Variablen */ int portB = 63;
int portC = 127;

void setup () {
  Serial.begin(9600);
  Serial.println(portB == portC);delay(1000);
  Serial.println(portB != portC);delay(1000);
  Serial.println(portB < portC);delay(1000);
  Serial.println(portB > portC);delay(1000);
  Serial.println(portB <= portC);delay(1000);
  Serial.println(portB >= portC);
}

void loop () {
  /* wenn Block */
  if(portB < portC) /* wenn die Bedingung erfüllt ist, wird die folgende Anweisung
  ausgeführt*/ portB++;
  delay(200);
  Serial.print("PORTB= ");Serial.println(portB);

  /* if ... else-Block */
  if(portB == portC){
    Serial.println("PortB ist gleich PortC");
    portB++;
  }
  sonst{
    Serial.println("PortB ist nicht gleich PortC");
  }

  /* wenn ... sonst wenn ... sonst Block */
  if(portB > portC){
    Serial.println("PortB ist größer als PortC");
  }
  else if (PortB < PortC){
    Serial.println("PortB ist kleiner als
    PortC");
  }
  sonst{
    Serial.println("PortB ist gleich PortC");
  }

}
```

Erläuterung des Kodex

Einrichtung: Der serielle Monitor ist mit einer Baudrate von 9600 aktiviert. Für zuvor definierte Variablen werden die entsprechenden Vergleichsoperationen nur einmal durchgeführt.

Schleife: Endlosschleife, in der Codeblöcke kontinuierlich abgearbeitet werden.

Anzahl der Experimente: 11

11- Kontrollstrukturen in der Arduino IDE: Wie man Kontrollstrukturen verwendet-2

Experiment Zielsetzung

Das Ziel dieses Experiments ist es, zu demonstrieren, wie Kontrollstrukturen in der integrierten Entwicklungsumgebung (IDE) von Arduino verwendet werden. Die Schüler lernen die Verwendung der ***switch*** **case-Anweisung** und des ***Bedingungsoperators*** "?:".

Theoretischer Hintergrund

Ähnlich wie die if-Anweisungen steuert **switch...case** den Programmfluss, indem der Programmierer verschiedene Codes angeben kann, die unter verschiedenen Bedingungen ausgeführt werden sollen.
Im Besonderen,
eine switch-Anweisung vergleicht den Wert einer Variablen mit den in den case-Anweisungen angegebenen Werten.

Am Ende jeder case-Anweisung wird ein break-Schlüsselwort eingefügt. Ansonsten wird jede switch-Anweisung unabhängig von der Bedingung in der case-Anweisung ausgeführt. Ein default-Schlüsselwort wird verwendet, wenn es keine case-Übereinstimmung in der switch-Anweisung gibt.

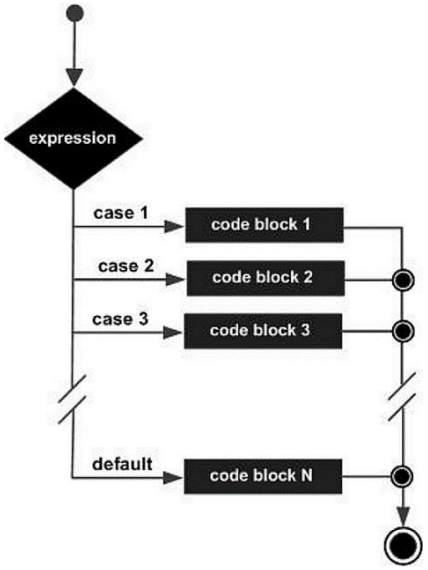


Abbildung 8 Flussdiagramm der switch case-Anweisung

Erforderliche Materialien

Arduino UNO R3, Arduino IDE Software oder Tinkercad Simulationsportal.

Experiment Kreislauf

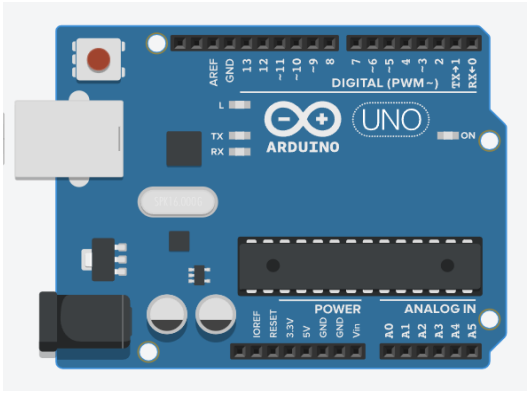


Abbildung 9 Entwicklungsplatine Arduino UNO R3

Schließen Sie das Arduino-Board über ein USB-Kabel vom Typ A/B an den Computer an und öffnen Sie die Arduino-IDE. Wählen Sie den Port (Tools→Port), an den das Arduino angeschlossen ist. Öffnen Sie den Bildschirm Serieller Monitor im Menü Extras.

Verfahrensschritte

1- Bauen Sie die Schaltung wie beschrieben auf und stellen Sie sicher, dass alle Verbindungen sicher sind.

2- Schreiben Sie den Code und laden Sie ihn auf Arduino hoch.

3- Beobachten Sie den Bildschirm der seriellen Schnittstelle.

4- Beobachten Sie die Veränderungen auf dem Bildschirm.

Codes - 1

```
/* Definition der globalen
Variablen */ byte portC = 1;
bool left = true;
void setup(){
Serial.begin(9600);
}
void loop(){
 switch (portC) {
  case 1: Serial.println("Erste LED ist eingeschaltet.");
  break; case 2: Serial.println("Zweite LED ist
  eingeschaltet."); break; case 4: Serial.println("Dritte LED
  ist eingeschaltet."); break; case 8: Serial.println("Vierte
  LED ist in Betrieb."); break; case 16:
  Serial.println("Fünfte LED ist in Betrieb."); break; case
  32: Serial.println("Sechste LED ist in Betrieb."); break;
  case 64: Serial.println("Siebte LED ist in Betrieb.");
  break; case 128: Serial.println("Die achte LED ist in
  Betrieb."); break; default: Serial.println("Ungültiger
  Zustand!");
 }
 if(links)
   portC = portC << 1; // Bit-Shift-Operator. Alle Bits werden einmal nach links verschoben.
 sonst
   portC = portC >> 1; // Bit-Shift-Operator. Alle Bits werden einmal nach rechts verschoben.
 if(portC == 0 && left){ // Mehrfachvergleich mit if-Block portC
   = 128;
   links = falsch;
   Serial.println("**********************");
 }
 else if(portC == 0 && !left){ // Mehrfacher Vergleich mit else if block
   portC = 1;
```

```
    links = wahr;
    Serial.println("**********");
  }
  delay(1000);
}
```

Erläuterung des Kodex

Einrichtung: Der serielle Monitor ist mit einer Baudrate von 9600 aktiviert.

Schleife: Der Schaltkastenblock wird bedient. Die boolesche Variable wird entsprechend dem maximalen oder minimalen Bitwert geändert, so dass die Abfrage kontinuierlich von links nach rechts oder von rechts nach links erfolgt.

Codes - 2

```
/* Definition der globalen
Variablen */ int portB = 63;
int portC = 127;
char c = 'k';

void setup() {
  Serial.begin(9600);

  /* Suche nach max(AnschlussB, AnschlussC): */
  Serial.println((portB>portC)?portB:portC);/*Der Wert von portB wird gedruckt, wenn er größer ist als portC.
Andernfalls wird der Wert von AnschlussC gedruckt.*/

  /* Kleinbuchstaben in Großbuchstaben umwandeln: */
  c = ( c >= 'a' && c <= 'z' ) ? ( c - 32 ) : c;
  Serial.println(c);
}

void loop() {

}
```

Erläuterung des Kodex

Einrichtung: Der serielle Monitor ist mit einer Baudrate von 9600 aktiviert. Der Bedingungsoperator wird verwendet, um festzustellen, welche Eingabe größer als die andere ist. Die Umwandlung eines gegebenen Kleinbuchstabens in einen Großbuchstaben entsprechend dem Wert der ASCII-Zeichentabelle ist mit dem bedingten Operator ebenfalls leicht möglich.

Schleife: -

Anzahl der Experimente: 12

12- Kontrollstrukturen in der Arduino IDE: Wie man Kontrollstrukturen verwendet-3

Experiment Zielsetzung

Das Ziel dieses Experiments ist es, zu demonstrieren, wie Kontrollstrukturen in der integrierten Entwicklungsumgebung (IDE) von Arduino verwendet werden. Die Schüler lernen die Verwendung von for-, while- und ***do*** while-Schleifen.

Theoretischer Hintergrund

Die for-Anweisung wird verwendet, um einen Block von Anweisungen zu wiederholen, die in geschweifte Klammern eingeschlossen sind. Ein Inkrementzähler wird in der Regel zum Erhöhen und Beenden der Schleife verwendet. Die for-Anweisung ist für alle sich wiederholenden Operationen nützlich. Diese Struktur wird häufig bevorzugt, insbesondere bei der Abfrage von Port-Pins von Arduino-Boards. Der for-Schleifenkopf besteht aus drei Teilen:

```
for (Initialisierung; Bedingung; Inkrement) {
//Stellungnahme(n);
}
```

*"**while***"-Schleifen laufen in einer Endlosschleife, bis der Ausdruck innerhalb der Klammer () falsch wird. Die Schleife läuft so lange, wie die Bedingung erfüllt ist. Aus diesem Grund sollte sie vorsichtig verwendet werden, um eine Endlosschleife zu vermeiden.

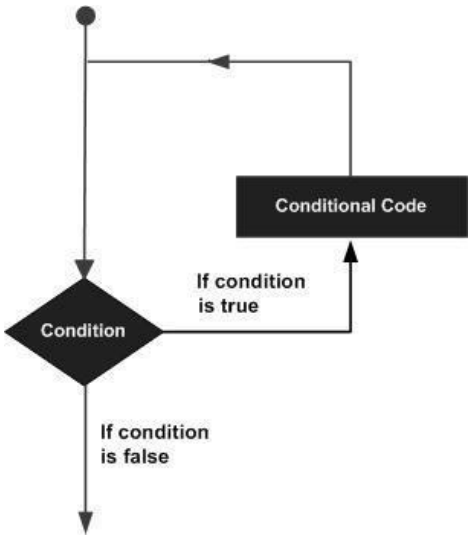


Abbildung 10 Betrieb von Schleifen

Die "***do*** ... ***while***"-Schleife ähnelt der while-Schleife, aber sie wird mindestens einmal ausgeführt, unabhängig davon, ob die Bedingung erfüllt ist oder nicht.

Erforderliche Materialien

Arduino UNO R3, 6 Stück 220 Ω Widerstände, 6 LEDs, Arduino IDE Software oder Tinkercad Simulationsportal.

Experiment Kreislauf

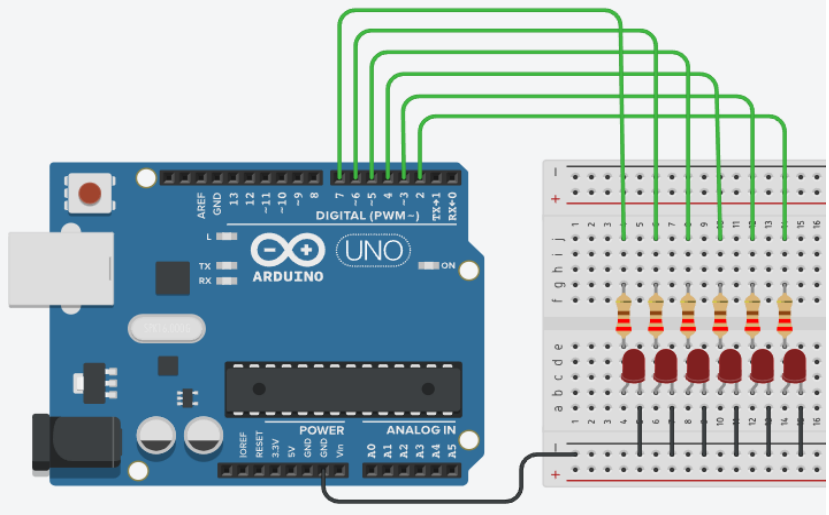


Abbildung 11 Entwicklungsplatine Arduino UNO R3

Schließen Sie das Arduino-Board über ein USB-Kabel vom Typ A/B an den Computer an und öffnen Sie die Arduino-IDE. Wählen Sie den Port (Tools→Port), an den der Arduino angeschlossen ist. Öffnen Sie den Bildschirm Serieller Monitor im Menü Extras.

Verfahrensschritte

1- Bauen Sie die Schaltung wie beschrieben auf und stellen Sie sicher, dass alle Verbindungen sicher sind.

2- Schreiben Sie den Code und laden Sie ihn auf Arduino hoch.

3- Beobachten Sie das Verhalten der LEDs.

4- Beobachten Sie die Änderungen auf dem Bildschirm der seriellen Schnittstelle.

Codes

```
// Globale Variablen
// Zweidimensionale Arrays sind definiert
int sensors01[3][3]={3,-7,7,8,5,-4,2,3,0};
int sensors02[3][3]={4,3,0,2,-1,-4,8,9,5};
int sensor3=50;
int sensor4=100;
int counter = 1;
void setup () {
  Serial.begin(9600);
}

 void loop () {
  PORTD = 4;
  delay(250);
  // for-Schleife beginnt
  for(int i=0;i<6;i++){
    PORTD = PORTD << 1;
     delay(250);
  }
  PORTD = 128;
  // andere for-Schleife
  beginnt for(int
  i=0;i<6;i++){ PORTD =
  PORTD >> 1;
```

```
  delay(250);
 }

 //verschachtelte for-Schleifen
 for(int i=0;i<=2;i++){//i Variable
   for(int j=0;j<=2;j++){// j Variable
     Serial.print(sensors01[i][j] + sensors02[i][j]);
     Serial.print("\t");//Tabulator
   }
   Serial.println();//Carriage return
 }

 // while-Schleife
 while(sensor3 < sensor4){
 Serial.print("I run ");
 Serial.print(counter);
 Serial.println(". times.");
 counter++;
   sensor3 += 10; // unärer Additionsoperator
   delay(200);
 }

 //do...while-Schleife
 tun{
   delay(200);
   Serial.println("Ich laufe mindestens einmal...");
 }while (Sensor3 > Sensor4);
}
```

Erläuterung des Kodex

Einrichtung: Der serielle Monitor ist mit einer Baudrate von 9600 aktiviert.

Schleife: Zwei for-Schleifen werden ausgeführt, um die LEDs am PORTD des Arduino-Boards zu steuern. Eine verschachtelte for-Schleife wird ausgeführt, um zu zeigen, wie man zweidimensionale Arrays scannt. "***while***"- und "***do...while***"-Schleifen werden ausgeführt, um den Vergleich von zwei Werten zu testen.

Anzahl der Experimente: 13

Experiment Name: Kontrollstrukturen in der Arduino IDE: Wie man Kontrollstrukturen verwendet 4 Ziel des Experiments

Das Ziel dieses Experiments ist es, zu demonstrieren, wie benutzerdefinierte und eingebaute Funktionen in der integrierten Entwicklungsumgebung (IDE) von Arduino verwendet werden. Die Schüler lernen die Deklaration von benutzerdefinierten Funktionen und die Verwendung einiger wichtiger integrierter Funktionen.

Theoretischer Hintergrund

Wie Sie bisher erfahren haben, haben wir einige grundlegende integrierte Funktionen in der Arduino-IDE verwendet, wie setup(), loop() und delay(). Eine Funktion ermöglicht es dem Benutzer, die Programme in Codesegmente zu strukturieren, um einzelne Aufgaben auszuführen. Die Arduino-Entwicklungsumgebung benötigt zwei Hauptfunktionen: setup() und loop().

Die gebräuchlichste Syntax zur Definition einer Funktion lautet wie folgt:

```
return_type funktion_name(argument1, argument2, ...) {
    statements;
    return_type_value;
}
```

return_type ist ein Datentyp wie z.B. integer, float, char, usw. Nicht jede Funktion erfordert die Rückgabe eines Wertes. In einem solchen Fall ist der return_type-Wert der Funktion als **void** definiert. Argumente sind beliebige Variablen oder Werte mit geeignetem Datentyp. Der Datentyp eines Arguments muss mit dem definierten Typ übereinstimmen.

Eine Funktion wird vor allen anderen Funktionen deklariert, vor oder nach den Schleifen- und Setup-Funktionen. Wenn Sie nur die Funktion selbst schreiben, müssen Sie die Funktion vor den Schleifen- oder Einrichtungsfunktionen deklarieren (wo sie aufgerufen wird).

Wenn Sie eine Funktion nach Setup- oder Schleifenfunktionen deklarieren wollen, müssen Sie einen **Funktionsprototyp** über die anderen Funktionen schreiben, in denen sie aufgerufen wird. Der Funktionsprototyp muss von einem Semikolon (;) gefolgt werden.

Erforderliche Materialien

Arduino UNO R3, Arduino IDE Software oder Tinkercad Simulationsportal.

Experiment Kreislauf

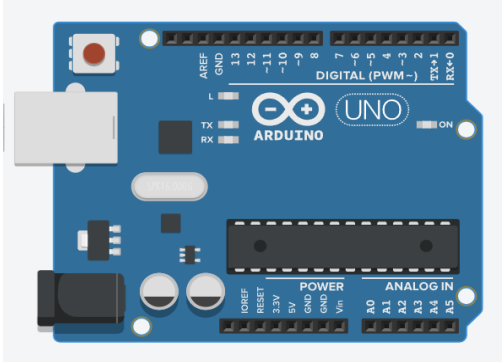


Abbildung 12 Entwicklungsplatine Arduino UNO R3

Schließen Sie das Arduino-Board über ein USB-Kabel vom Typ A/B an den Computer an und öffnen Sie die Arduino-IDE. Wählen Sie den Port (Tools→Port), an den der Arduino angeschlossen ist. Öffnen Sie den Bildschirm Serieller Monitor im Menü Extras.

Verfahrensschritte

1- Bauen Sie die Schaltung wie beschrieben auf und stellen Sie sicher, dass alle Verbindungen sicher sind.

2- Schreiben Sie den Code und laden Sie ihn auf Arduino hoch.

3- Beobachten Sie den Bildschirm der seriellen Schnittstelle.

4- Beobachten Sie die Veränderungen auf dem Bildschirm.

Codes

```
// Globale Variablen
int sum_func(int x, int y);// ein benutzerdefinierter
Funktionsprototyp int sensor1=21;
int sensor2=45;

// Deklaration einer benutzerdefinierten Funktion, bevor sie
aufgerufen wird void printMessage(){
  Serial.println("Diese Funktion ist oberhalb der Funktion setup() deklariert");
  Serial.print("Die Differenz der Sensoren ist ");
  Serial.println(sensor1-sensor2);
}

void setup () {
  Serial.begin(9600);
  printMessage(); //benutzerdefinierte Funktion wird aufgerufen
}

void loop () {
  if(sum_func(sensor1, sensor2)>=50){
    Serial.print("Summe der Sensoren sind, ");
    Serial.println(sum_func(sensor1, sensor2));
  }
  sonst{
    Serial.println("Summe der Sensoren ist kleiner als 50");
  }
  sensor2 = 25;
  delay(1000); // Sensordaten werden jede Sekunde gelesen.
```

```
}

// Deklination der Funktion nach loop(), wo sie
aufgerufen wird. int sum_func(int x, int y){
  int z = 0;
  z = x + y;
  return z;
}
```

Erläuterung des Kodex

Einrichtung: Der serielle Monitor wird mit einer Baudrate von 9600 aktiviert. Eine benutzerdefinierte Funktion wird aufgerufen.

Schleife: Eine benutzerdefinierte Summenfunktion wird aufgerufen. Es wird jede Sekunde ein Scan durchgeführt.

Anzahl der Experimente: 14

14- Strings in der Arduino IDE: Wie man String-Literale verwendet

Experiment Zielsetzung

Das Ziel dieses Experiments ist es, zu demonstrieren, wie der Datentyp String in der integrierten Entwicklungsumgebung (IDE) von Arduino verwendet wird. Die Schüler lernen verschiedene Deklarationsmethoden für den Datentyp "String" kennen.

Theoretischer Hintergrund

Wie in allen anderen Programmiersprachen gibt es auch in der Arduino IDE einen wichtigen Datentyp namens "String", mit dem keine mathematischen Operationen durchgeführt werden können. Strings sind ein textbasierter Datentyp. Textstrings können auf zwei Arten dargestellt werden. Sie können den Datentyp ***String*** verwenden, oder Sie können einen String aus einem Array des Typs ***char*** erstellen und ihn mit Null abschließen ('\0').

- String-Syntax

Alle folgenden Angaben sind gültige Deklarationen für

Strings. char Str1[10];
char Str2[8] = {'a', 'r', 'd', 'u', 'i', 'n', 'o'};
char Str3[8] = {'a', 'r', 'd', 'u', 'i', 'n', 'o', '\0'};
char Str4[] = "arduino";
char Str5[7] = "arduino";
char Str6[10] = "arduino";

- Möglichkeiten zur Deklaration von Zeichenketten
  - Ein Array von Zeichen deklarieren, ohne es wie in Str1 zu initialisieren
  - Deklarieren Sie ein Array von Zeichen (mit einem zusätzlichen Zeichen), und der Compiler fügt das erforderliche Nullzeichen hinzu, wie in Str2
  - Explizites Hinzufügen des Nullzeichens, Str3
  - Initialisierung mit einer String-Konstante in Anführungszeichen; der Compiler passt die Größe des Arrays an die String-Konstante und ein abschließendes Null-Zeichen (Str4) an
  - Initialisierung des Arrays mit einer expliziten Größe und einer String-Konstante, Str5
  - Initialisierung des Arrays, wobei zusätzlicher Platz für eine größere Zeichenkette, Str6, gelassen wird
- Null Terminierung

Im Allgemeinen werden Zeichenketten mit einem Nullzeichen (ASCII-Code 0) abgeschlossen. Dadurch können Funktionen (wie **Serial.print()**) erkennen, wo das Ende einer Zeichenkette liegt. Andernfalls würden sie mit dem Lesen weiterer Speicherbytes fortfahren, die eigentlich nicht Teil der Zeichenkette sind.

- Einfache Anführungszeichen oder doppelte Anführungszeichen

Zeichenketten werden immer in doppelten Anführungszeichen ("Arduino") und Zeichen immer in einfachen Anführungszeichen ("A") definiert.

- Umwickeln langer Schnüre

So können Sie lange Strings umwickeln:

```
char myString[] = "Dies ist die erste Zeile"
                  "Dies ist die zweite
                  Zeile" "Dies ist die
                  letzte Zeile";
```

- Array von Zeichenketten

String-Arrays werden verwendet, wenn Sie mit großen Mengen von Meldungstexten arbeiten, z. B. bei Projekten mit LCD-Anzeigen. Sie können die gleichen Meldungen an verschiedenen Stellen des Projekts ausgeben. In solchen Fällen wäre es logisch, eine Meldungsserie zu erstellen.

```
char *sensorStrings[] = {"Dies ist der erste Sensor", "Dies ist
                         Zweiter Sensor", "Dies ist der dritte Sensor", "Dies ist der
                         vierte Sensor", "Dies ist der fünfte Sensor", "Dies ist der
                         sechste Sensor6"
                    };
```

Im folgenden Code zeigt das Sternchen nach dem Datentyp char "**char***" an, dass es sich um ein Array von "Zeigern" handelt.

- String-Verkettung

Mit dem Datentyp **char** erstellte Strings können nicht mit dem arithmetischen Operator verkettet werden. Für diesen Vorgang muss eine String-Datenstruktur verwendet werden.

Erforderliche Materialien

Arduino UNO R3, Arduino IDE Software oder Tinkercad Simulationsportal.

Experiment Kreislauf

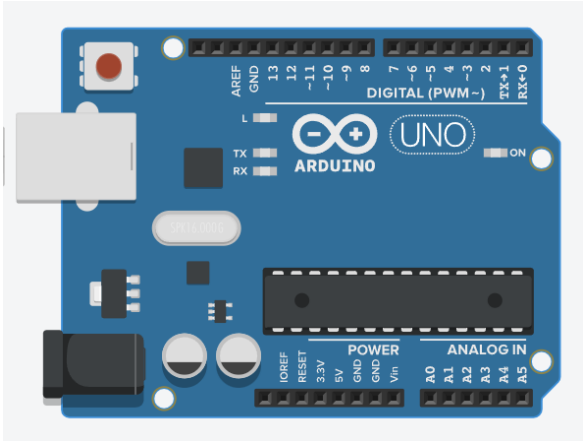


Abbildung 13 Entwicklungsplatine Arduino UNO R3

Schließen Sie das Arduino-Board über ein USB-Kabel vom Typ A/B an den Computer an und öffnen Sie die Arduino-IDE. Wählen Sie den Port (Tools→Port), an den der Arduino angeschlossen ist. Öffnen Sie den Bildschirm Serieller Monitor im Menü Extras.

Verfahrensschritte

1- Bauen Sie die Schaltung wie beschrieben auf und stellen Sie sicher, dass alle Verbindungen sicher sind.

2- Schreiben Sie den Code und laden Sie ihn auf Arduino hoch.

3- Beobachten Sie den Bildschirm der seriellen Schnittstelle.

4- Beobachten Sie die Veränderungen auf dem Bildschirm.

Codes

```
// Globale Variablen
char Str1[10]; // initialisiert, aber keine erste Wertzuweisung
char Str2[8] = {'a', 'r', 'd', 'u', 'i', 'n', 'o'}; /* initialisiert mit erster Wertzuweisung, Nullterminierung wird
automatisch hinzugefügt */
```

```
char Str3[8] = {'a', 'r', 'd', 'u', 'i', 'n', 'o', '\0'}; /* initialisiert mit erster Wertzuweisung, Nullterminierung wird
manuell hinzugefügt */
char Str4[] = "arduino"; // Arraygröße wird durch den zugewiesenen
Wert erzeugt. char Str5[8] = "arduino"; // Arraygröße und Zuweisung
erfolgen gemeinsam. char Str6[10] = "arduino"; //Arraygröße ist größer
als der zugewiesene Wert. char myString[] = "Dies ist die erste Zeile"
"Dies ist die zweite
Zeile" "Dies ist die
letzte Zeile";

char *sensorStrings[] = {"Dies ist der erste Sensor", "Dies ist der zweite Sensor", "Dies ist der dritte Sensor",
"Dies ist der vierte Sensor", "Dies ist der fünfte Sensor", "Dies ist der sechste Sensor6"
        };

void setup () {
  Serial.begin(9600);
  Serial.println(Str1);delay(500);
  Serial.println(Str2);delay(500);
  Serial.println(Str3);delay(500);
  Serial.println(Str4);delay(500);
  Serial.println(Str5);delay(500);
  Serial.println(Str6);delay(500);
  Serial.println(myString);
}

void loop () {
  for (int i = 0; i < 6; i++) {
    Serial.println(sensorStrings[i]);
    delay(500);
  }
}
```

Erläuterung des Kodex

Einrichtung: Der serielle Monitor ist mit einer Baudrate von 9600 aktiviert. Es werden verschiedene String-Literale gedruckt.

Schleife: Ein String-Array wird mit Hilfe der for-Schleife in einem Scan-Verfahren gedruckt.

Anzahl der Experimente: 15

15- Strings in Arduino IDE: Wie man die String-Datenstruktur verwendet

Experiment Zielsetzung

Das Ziel dieses Experiments ist es, zu demonstrieren, wie das String()-Objekt (Datenstruktur) in der integrierten Entwicklungsumgebung (IDE) von Arduino verwendet wird. Die Schüler lernen verschiedene Deklarationsmethoden des String()-Objekts kennen.

Theoretischer Hintergrund

Arduino IDE verwendet eine objektorientierte Programmierstruktur und es gibt viele Objekttypen, die in Arduino-Plattformen verwendet werden. Das String()-Objekt ist eines der populärsten von ihnen. Stirng() ist eine Art Datenstruktur der Klasse. Es gibt mehrere Versionen, um eine Instanz der String-Klasse zu erstellen.

- eine konstante Zeichenkette in doppelten Anführungszeichen (d. h. ein Char-Array)
- ein einzelnes konstantes Zeichen, in einfachen Anführungszeichen
- eine weitere Instanz des String-Objekts
- eine konstante Ganzzahl oder eine lange Ganzzahl
- eine konstante Ganzzahl oder eine lange Ganzzahl, die eine bestimmte Basis verwendet
- eine Integer- oder Long-Integer-Variable
- eine Integer- oder Long-Integer-Variable, die eine bestimmte Basis verwendet
- eine Gleitkommazahl oder ein Double mit einer bestimmten Anzahl von Nachkommastellen

Die Konstruktion einer Zeichenkette aus einer Zahl ergibt eine Zeichenkette, die die ASCII-Darstellung dieser Zahl enthält. Die Vorgabe ist die Basis Zehn, also

```
String thisString = String(15);
```

erhalten Sie die Zeichenfolge "15". Sie können jedoch auch andere Basen

verwenden. Zum Beispiel: String thisString = String(15,

HEX);

erhalten Sie die Zeichenkette "f", die die hexadezimale Darstellung des Dezimalwerts 15 ist. Oder wenn Sie binär bevorzugen,

```
String thisString = String(15, BIN);
```

ergibt die Zeichenfolge "1111", die die binäre Darstellung von 15 ist.

- **Syntax**

```
String(val)
String(val, base)
String(val, decimalPlaces)
```

- **Parameter**

val: eine Variable, die als String formatiert werden soll. Erlaubte Datentypen: ***string***, ***char***, ***byte***, ***int***, ***long***, ***unsigned int***, ***unsigned long***, ***float***, ***double***.
base: (optional) die Basis, in der ein ganzzahliger Wert formatiert werden soll. decimalPlaces: nur wenn val float oder double ist. Die gewünschten Dezimalstellen.

- String-Verkettung

Damit Strings zusammengefügt werden können, müssen sie zunächst als String-Objekt definiert werden und einen Anfangswert erhalten, bevor Sie mit der Verkettung verschiedener Datentypen beginnen.

Erforderliche Materialien

Arduino UNO R3, Arduino IDE Software oder Tinkercad Simulationsportal.

Experiment Kreislauf

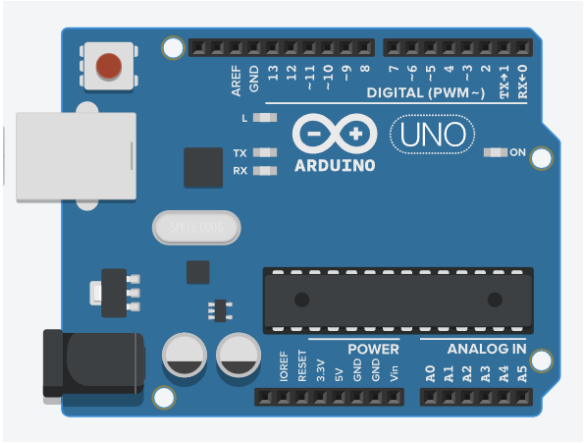


Abbildung 14 Entwicklungsplatine Arduino UNO R3

Schließen Sie das Arduino-Board über ein USB-Kabel vom Typ A/B an den Computer an und öffnen Sie die Arduino-IDE. Wählen Sie den Port (Tools→Port), an den der Arduino angeschlossen ist. Öffnen Sie den Bildschirm Serieller Monitor im Menü Extras.

Verfahrensschritte

1- Bauen Sie die Schaltung wie beschrieben auf und stellen Sie sicher, dass alle Verbindungen sicher sind.

2- Schreiben Sie den Code und laden Sie ihn auf Arduino hoch.

3- Beobachten Sie den Bildschirm der seriellen Schnittstelle.

4- Beobachten Sie die Veränderungen auf dem Bildschirm.

Codes

```
// Globale Variablen
int intSensor = 15;
float floatSensor = 13,495;
String stringString = "Hello String"; // Verwendung einer konstanten String
String stringCharacter = String('a'); // Umwandlung eines konstanten Zeichens in einen String
String stringObject = String("Dies ist eine Zeichenkette"); // Umwandlung einer konstanten
Zeichenkette in ein String-Objekt String stringConcatenation = String(stringString + " mit
mehr"); // Verkettung zweier Zeichenketten String stringInteger = String(15); // Verwendung
einer konstanten Ganzzahl
String stringDEC = String(intSensor, DEC); // unter Verwendung eines int und einer Basis
String stringHEX = String(intSensor, HEX); // Verwendung eines int und einer Basis
(hexadezimal) String stringBIN = String(intSensor, BIN); // Verwendung eines int und
einer Basis (binär)
String stringLongDEC = String(millis(), DEC);// Verwendung eines Long und einer Base
String stringFloat = String(floatSensor, 2);// Verwendung eines Floats mit einer Genauigkeit von zwei Dezimalstellen

String stringOne, stringTwo, stringThree;
void setup () {
```

```
  Serial.begin(9600);
}

void loop () {
  Serial.println(stringString);delay(500);
  Serial.println(stringCharacter);delay(500);
  Serial.println(stringObject);delay(500);
  Serial.println(stringConcatenation);delay(500);
  Serial.println(stringInteger);delay(500);
  Serial.println(stringDEC);delay(500);
  Serial.println(stringHEX);delay(500);
  Serial.println(stringBIN);delay(500);
  Serial.println(stringLongDEC);delay(500);
  Serial.println(stringFloat);delay(500);
  stringLongDEC = String(millis(), DEC);
  Serial.println(stringLongDEC);

  stringOne = String("Sie haben ");
  stringTwo = String("diese
  Zeichenfolge"); stringThree =
  String();
  // Hinzufügen einer konstanten Ganzzahl zu
  einem String: stringThree = stringOne + 123;
  Serial.println(stringThree);delay(500);
  // Hinzufügen einer konstanten langen Ganzzahl
  zu einem String: stringThree = stringOne +
  123456789;
  Serial.println(stringThree);delay(500);
  // Hinzufügen eines konstanten Zeichens zu
  einem String: stringThree = stringOne + 'A';
  Serial.println(stringThree);delay(500);
  // Hinzufügen einer konstanten Zeichenkette zu
  einem String: stringThree = stringOne + "abc";
  Serial.println(stringThree);delay(500);
  stringThree = stringOne + stringTwo;
  Serial.println(stringThree);delay(500);
  // Hinzufügen einer variablen Ganzzahl
  zu einem String: int sensorValue =
  analogRead(A0); stringOne =
  "Sensorwert: "; stringThree = stringOne
  + sensorValue;
  Serial.println(stringThree);delay(500);
  // Hinzufügen einer variablen langen Ganzzahl
  zu einem String: stringOne = "millis() Wert: ";
  stringThree = stringOne + millis();
  Serial.println(stringThree);delay(500);
  while(true); //Endlosschleife, es wird nichts
  gemacht.
}
```

Erläuterung des Kodex

Einrichtung: Der serielle Monitor ist mit einer Baudrate von 9600 aktiviert.

Schleife: Eine benutzerdefinierte Summenfunktion wird aufgerufen. Es wird jede Sekunde ein Scan durchgeführt.

Anzahl der Experimente: 16

16- Digitale E/A-Operationen

Experiment Zielsetzung

Das Hauptziel dieses Experiments ist es, die SchülerInnen in die grundlegenden digitalen Ein-/Ausgabeoperationen (E/A) mit Arduino einzuführen. Die Teilnehmer lernen, wie man digitale Pins auf dem Arduino-Board konfiguriert und verwendet, um Eingänge (wie Tasten) zu lesen und Ausgänge (wie LEDs) zu steuern.

Theoretischer Hintergrund

Die Arduino-Plattform ist mit digitalen Pins ausgestattet, die entweder als Eingang oder als Ausgang konfiguriert werden können. Diese digitalen Pins werden zum Lesen digitaler Signale und zur Steuerung physischer Geräte verwendet. Wenn sie als Eingänge konfiguriert sind, können sie das Vorhandensein oder Nichtvorhandensein eines Signals (HIGH oder LOW) erkennen. Wenn sie als Ausgänge konfiguriert sind, können sie den Pin auf HIGH (in der Regel 5 V) oder LOW (0 V) setzen, wodurch sie Geräte wie LEDs, Motoren usw. steuern können.

Digitaler Eingang: Lesen des Zustands eines digitalen Signals. Wird häufig bei Tasten, Schaltern und Sensoren verwendet.

Digitaler Ausgang: Setzen eines digitalen Pins auf HIGH oder LOW zur Steuerung von Geräten wie LEDs oder Relais.

Die Arduino-Funktionen pinMode(), digitalWrite() und digitalRead() sind grundlegend für digitale I/O-Operationen:

pinMode(pin, mode): Setzt einen Pin als INPUT, OUTPUT oder

INPUT_PULLUP. digitalWrite(pin, value): Schreibt einen HIGH- oder LOW-

Wert an einen digitalen Pin. digitalRead(pin): Liest den Wert von einem

digitalen Pin.

Erforderliche Materialien

Arduino UNO R3 Platine

Lochrasterplatte

LED

220-Ohm-Widerstand

Druckknopfschalter

Überbrückungsdrähte

Arduino IDE-

Software

Experiment Kreislauf

1-Schließen Sie die LED über einen 220-Ohm-Widerstand an einen der digitalen Pins an, um den Strom zu begrenzen.

2- Verbinden Sie eine Seite des Tasters mit einem anderen digitalen Pin und die andere Seite mit Masse. Verwenden Sie einen Pull-up-Widerstand oder den internen INPUT_PULLUP-Modus, um ein stabiles HIGH-Signal zu gewährleisten, wenn die Taste nicht gedrückt wird.

Verfahrensschritte

1- Bauen Sie die Schaltung wie beschrieben zusammen und stellen Sie sicher, dass alle Verbindungen sicher sind.

2- Schreibe den Arduino-Code, um die LED mit dem Druckknopf zu steuern.

3- Laden Sie den Code auf das Arduino-Board hoch.

4- Drücken Sie den Taster und lassen Sie ihn los, um die LED-Kontrolle zu testen.

5- Beobachten Sie die Reaktion der LED auf den Druckknopf.

Beispiel Code

```
// Pin-Nummern definieren
        const int buttonPin = 2; // die Nummer des Taster-Pins const
        int ledPin = 13; // die Nummer des LED-Pins
// Variablen werden sich ändern:
        int buttonState = 0;     //Variable zum Auslesen des Tasterstatus
        void setup() {
  // Initialisierung des LED-Pins als
          Ausgang: pinMode(ledPin,
          OUTPUT);
  // Initialisieren Sie den Tasterpin als Eingang:
  pinMode(buttonPin, INPUT);
}
void loop() {
  // Lesen Sie den Zustand des Tasterwerts:
  buttonState = digitalRead(buttonPin);
  // Prüfen Sie, ob der Taster gedrückt ist.
  // Wenn ja, ist der ButtonState
  HIGH: if (buttonState == HIGH)
  {
  // LED anschalten:
    digitalWrite(ledPin, HIGH);
  } sonst {
```

```
    // LED ausschalten:
    digitalWrite(ledPin, LOW);
  }
}
```

Erläuterung des Kodex

- Einrichtung: Die Funktion pinMode() konfiguriert den LED-Pin als Ausgang und den Button-Pin als Eingang.
- Schleife: Das Programm liest kontinuierlich den Zustand der Taste mit digitalRead(). Wenn der Taster gedrückt ist (und aufgrund der Pull-up-Konfiguration auf HIGH steht), wird die LED eingeschaltet. Andernfalls wird die LED ausgeschaltet. Der Zustand der LED ändert sich je nach Tastendruck.

Dieses Experiment demonstriert die grundlegenden digitalen E/A-Operationen, die die Basis für komplexere Arduino-Projekte mit Sensoren, Motoren und anderen Komponenten bilden.

Anzahl der Experimente: 17

17- Analoge E/A-Operationen

Experiment Zielsetzung

Ziel dieses Experiments ist es, die Schüler mit dem Konzept und der Anwendung von analogen Ein- und Ausgabeoperationen unter Verwendung eines Arduino-Boards vertraut zu machen. Die Teilnehmer lernen, wie sie analoge Signale von Sensoren lesen und analoge Geräte wie dimmbare LEDs oder Motoren mit Pulsweitenmodulation (PWM) steuern können.

Theoretischer Hintergrund

Im Gegensatz zu digitalen Signalen, die entweder HIGH oder LOW sind, können analoge Signale eine Reihe von Werten darstellen. Arduino-Boards können analoge Signale mit Hilfe von Analog-Digital-Wandlern (ADC) lesen und mit Pulsweitenmodulation (PWM) eine analoge Ausgabe erzeugen.

Analoger Eingang: Arduino-Boards wie der Uno verfügen über eine Reihe von Analogeingangsstiften, die mit A0 bis A5 bezeichnet sind. Diese Stifte können analoge Signale von Sensoren, wie z. B. Temperatur oder Licht, lesen und sie in einen digitalen Wert umwandeln, der vom Mikrocontroller verarbeitet werden kann.

Analoger Ausgang (PWM): Arduino kann zwar keinen echten analogen Ausgang erzeugen, aber er kann ihn durch PWM simulieren. PWM steuert das relative Tastverhältnis eines digitalen Signals, um die an Geräte wie LEDs oder Motoren gelieferte Leistung zu variieren.

Tastenfunktionen für analoge Operationen:

- analogRead(pin): Liest den Wert von einem analogen Pin und gibt einen Wert zwischen 0 (0V) und 1023 (5V) zurück.
- analogWrite(pin, Wert): Schreibt einen Analogwert (PWM-Welle) an einen Pin, um eine analoge Ausgabe zu simulieren. Der Wert kann zwischen 0 (immer aus) und 255 (immer an) liegen.

Erforderliche Materialien

Arduino UNO R3 Platine

Brettchen

Potentiometer (z.B. 10kΩ)

LED

220-Ohm-Widerstand

Überbrückungsdrähte

Arduino IDE-

Software

Experiment Kreislauf

1-Schließen Sie das Potentiometer an einen der analogen Eingangsstifte an (z. B. A0). Verbinden Sie eine Seite mit 5 V, den mittleren Pin mit A0 und die andere Seite mit GND.

2-Schließen Sie die LED über einen 220-Ohm-Widerstand an einen PWM-fähigen Digitalpin an (z. B. 9, 10, 11 am Uno), um den Strom zu begrenzen.

Verfahrensschritte

1-Bauen Sie die Schaltung wie beschrieben auf und stellen Sie sicher, dass alle Verbindungen sicher sind.

2-Schreiben und laden Sie den Arduino-Code hoch, der den Analogwert vom Potentiometer liest und die LED-Helligkeit entsprechend steuert.

3-Drehen Sie das Potentiometer und beobachten Sie die Veränderung der LED-Helligkeit.

4-Verstehen Sie den Zusammenhang zwischen der Stellung des Potentiometers und der Helligkeit der LED.

Beispiel Code

```
// Pin-Nummern definieren

const int potPin = A0; // die Nummer des Potentiometer-Pins

const int ledPin = 9; // die Nummer des LED-Pins

// Variable zum Speichern des

Potentiometerwerts int potValue = 0;

void setup() {

  // Initialisierung des LED-Pins als
```

```
  Ausgang: pinMode(ledPin, OUTPUT);
  // Initialisierung der seriellen Kommunikation mit 9600
  Bits pro Sekunde: Serial.begin(9600);
}
void loop() {
  // Lesen Sie den Wert vom Potentiometer ab:
  potValue = analogRead(potPin);
  // den Potentiometerwert von 0-1023 auf 0-255
  abbilden: int ledBrightness = map(potValue, 0, 1023,
  0, 255);
  // Einstellen der Helligkeit der
  LED: analogWrite(ledPin,
  ledBrightness);
  // Drucken Sie die Ergebnisse auf dem
  seriellen Monitor aus:
  Serial.print("Potentiometer: ");
  Serial.print(potValue);
  Serial.print("\t LED-Helligkeit: ");
  Serial.println(ledBrightness);
  delay(10); // kleine Verzögerung für
  Stabilität
}
```

Erläuterung des Kodex

Einrichten: Initialisiert den LED-Pin als Ausgang und beginnt die serielle Kommunikation zur Überwachung.

Schleife: Die Funktion analogRead() liest den Analogwert aus dem Potentiometer, der dann auf einen geeigneten PWM-Wert (0-255) abgebildet wird. Dieser Wert wird in analogWrite() verwendet, um die Helligkeit der LED einzustellen. Die Potentiometer- und LED-Werte werden zur Beobachtung auf dem seriellen Monitor ausgegeben.

Dieses Experiment demonstriert, wie analoge Eingänge gelesen und PWM-Ausgänge erzeugt werden können, was für die Verbindung mit einer Vielzahl von Sensoren und die Steuerung verschiedener Aktoren in komplexeren Arduino-Projekten unerlässlich ist.

INA-CODE

robyKode

web & mobile