



OTRONIC ARDUINO UNO-STARTERKIT

Arduino UNO-Starterkit: Tutorial und 16 Projekte

Schritt für Schritt: Was ist ein Arduino, was ist in deinem Kit, wie funktioniert Codey und 16 Projekte zum Selberbauen.

codey.online/de/starter-kits/uno/tutorial



Inhalt

Willkommen! Du wirst das bauen! 🚀	3
Was ist ein Arduino UNO?	4
Was ist in deinem Starterkit?	5
Breadboard	6
Jumperkabel	7
LED (Lämpchen)	8
Widerstand 220 Ω	9
Druckknopf	10
Aktiver Summer	11
Lichtsensord (LDR)	12
Temperatursensord DHT11	13
Abstandssensord HC-SR04	14
Bewegungssensord (PIR)	15
OLED-Display 0,91"	16
Servomotor SG90	17
RGB-LED-Ring (8 LEDs)	18
Relais 5V	19
9V-Batterieclip	20
Was ist Vibe Coding?	21
Erstelle dein kostenloses Konto	22
So funktioniert Codey	23
Anschließen und hochladen	24
Los geht's! 🎉	25
Deine erste Breadboard-Schaltung	26
Blinklicht	27
Ampel	28
Lichtknopf	29
Türklingel	30
Regenbogenring	31
Mini-Wetterstation	32
Parksensord	33
Nachtlämpchen	34
Bewegungsalarm	35
Schranke	36
Reaktionsspiel	37
Elektronischer Würfel	38
Parksensord deluxe	39
Abstandsmesserd mit Lichtbalken	40
Smarter Ventilator	41
Automatische Schranke	42

OTRONIC ARDUINO UNO-STARTERKIT

Willkommen! Du wirst das bauen! 🚀

Mit diesem Starterkit baust du deine eigenen schlaunen Geräte. Eine Wetterstation, einen Parksensor, einen Alarm... Und das Beste: Du musst nicht programmieren können. Codey hilft dir.

Schau dir kurz dieses Tutorial an. Dann weißt du, was in deinem Kit ist und was du damit machen kannst. Danach legst du sofort mit deinem ersten Projekt los.

So geht es:

- 2 **Was ist ein Arduino und was kannst du damit machen?**
- 3 **Alle Teile aus deinem Kit, eins nach dem anderen**
- 4 **Was ist Vibe Coding? Programmieren mit KI**
- 5 **Dein kostenloses Konto erstellen**
- 6 **So funktioniert Codey**
- 7 **Deinen Arduino anschließen und deinen Code hochladen**
- 8 **Loslegen mit 16 Projekten**

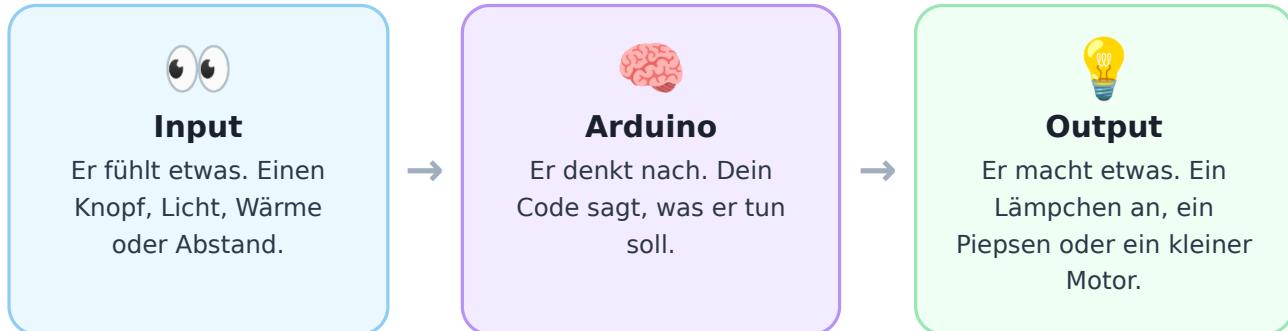


🕒 Das dauert ungefähr 15 Minuten. Du kannst immer etwas überspringen und später zurückkommen.

Was ist ein Arduino UNO?

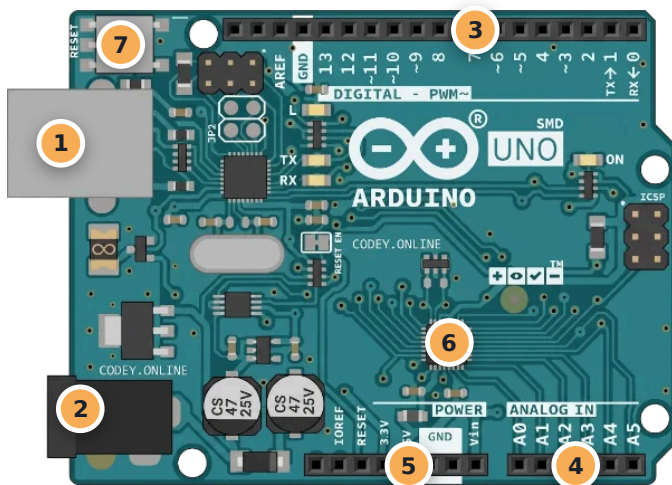
Der Arduino ist ein kleiner Computer. Du sagst ihm, was er tun soll, und das macht er. Immer wieder, ohne müde zu werden.

So denkt ein Arduino



Beispiel: Der Abstandssensor sieht, dass du näher kommst (input). Der Arduino denkt: "zu nah!" Und der Summer fängt an zu piepsen (output). So funktioniert ein Parksensoren in einem Auto.

Was ist auf dem Board?



- 1 USB-Port: Hier kommt das Kabel zu deinem Computer hinein. Darüber bekommt er Strom und Code.
- 2 Stromanschluss: für eine Batterie oder ein Netzteil (7-12V), wenn du keinen Computer benutzt.
- 3 Digitale Pins D0 bis D13: an oder aus. Für Lämpchen, Knöpfe und Summer.
- 4 Analoge Pins A0 bis A5: messen, wie viel. Zum Beispiel, wie viel Licht da ist.
- 5 Strom-Pins: 5V (Plus) und GND (Minus) geben deinen Bauteilen Strom.
- 6 Der Chip: das Gehirn. Hier kommt dein Code hinein.
- 7 Resetknopf: startet deinen Code neu.

Zwei Wörter wirst du oft sehen:

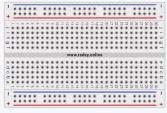
- **5V** ist der Pluspol. Von hier kommt der Strom (5 Volt, genau wie bei einem Handy-Ladegerät).
- **GND** (ground) ist der Minuspol. Hier fließt der Strom wieder zurück. Jedes Bauteil braucht ein Kabel zu GND.

Pins sind die kleinen Löcher an der Seite. Dort steckst du Kabel hinein. Ein Pin kann ein **Eingang** sein (er hört zu) oder ein **Ausgang** (er gibt Strom). Das bestimmst du im Code.

Was ist in deinem Starterkit?

Dein Kit ist voll mit Teilen. Manche fühlen etwas (Sensoren), andere machen etwas (Lämpchen, Summer, Motörchen). Klicke auf Weiter, dann lernst du sie Stück für Stück kennen.

Basis: Damit baust du alles



Breadboard

Das Steckbrett: bauen ohne Löten

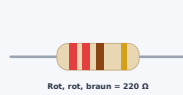
In deinem Kit: 1x



Jumperkabel

Dupont-Kabel: die Drähte, die alles verbinden

In deinem Kit: 40x



Widerstand 220 Ω

Bremst den Strom

In deinem Kit: 10x

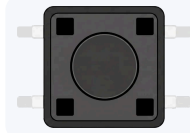


9V-Batterieclip

Dein Projekt ohne Computer laufen lassen

In deinem Kit: 1x

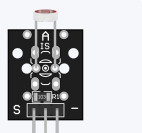
Input: Diese Teile fühlen oder messen etwas



Druckknopf

Ein kleiner Knopf zum Drücken

In deinem Kit: 5x



Lichtsensor (LDR)

Misst, wie hell oder dunkel es ist

In deinem Kit: 1x



Temperatursensor DHT11

Misst die Temperatur und die Luftfeuchtigkeit

In deinem Kit: 1x



Abstandssensor HC-SR04

Misst Abstand mit Schall, genau wie eine Fledermaus

In deinem Kit: 1x



Bewegungssensor (PIR)

Sieht, ob sich jemand bewegt

In deinem Kit: 1x

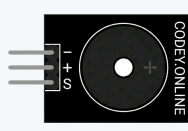
Output: Diese Teile machen etwas



LED (Lämpchen)

Ein kleines Lämpchen in 5 Farben

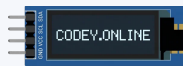
In deinem Kit: 10x



Aktiver Summer

Macht Pieptöne

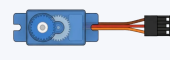
In deinem Kit: 1x



OLED-Display 0,91"

Ein kleiner Bildschirm für Text und Zahlen

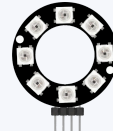
In deinem Kit: 1x



Servomotor SG90

Ein kleiner Motor, der in eine bestimmte Stellung dreht

In deinem Kit: 1x



RGB-LED-Ring (8 LEDs)

8 schlaue Lämpchen in jeder Farbe

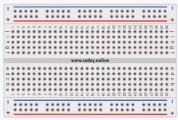
In deinem Kit: 1x



Relais 5V

Ein Schalter, den dein Arduino umlegen kann

In deinem Kit: 1x



Breadboard

Das Steckbrett: bauen ohne Lötten

In deinem Kit: 1x

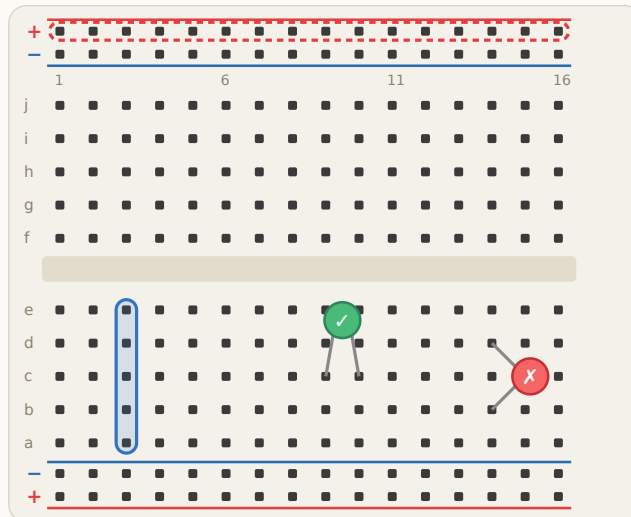
Was ist das?

Ein Breadboard ist eine kleine Platte mit vielen Löchern. Du steckst Bauteile und Drähte hinein. Unter den Löchern sind Metallstreifen. Sie verbinden manche Löcher miteinander. So baust du eine Schaltung ohne Lötten. Etwas falsch gemacht? Einfach herausziehen und neu machen.

So schließt du es an

So ist es aufgebaut:

- **Reihen mit 5 Löchern:** im mittleren Teil sind immer 5 Löcher nebeneinander verbunden (a-e und f-j). Alles, was du in dieselbe Reihe steckst, ist miteinander verbunden.
- **Die Rille in der Mitte** trennt die zwei Seiten. Links (a-e) und rechts (f-j) sind also *nicht* verbunden.
- **Stromleisten (+ und -)** an den langen Seiten sind über die ganze Länge verbunden. Verbinde + mit 5V und - mit GND von deinem Arduino. Jetzt hast du überall Strom.



- Stromleiste: die ganze Länge ist verbunden
- 5 Löcher in einer Reihe sind verbunden
- X Falsch: Beide Beinchen stecken in derselben Reihe

- Rille: links und rechts sind nicht verbunden
- ✓ Gut: Die Beinchen stecken in verschiedenen Reihen

Was kannst du damit machen?

- Schaltungen ausprobieren ohne Lötten
- Mehrere Bauteile gleichzeitig über die + und - Leisten mit Strom versorgen
- Eine LED mit einem Widerstand schön in Reihe schalten

Tipp

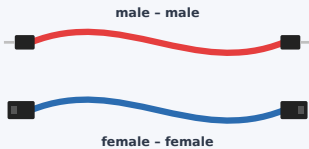
Stecke die zwei Beinchen eines Bauteils nie in dieselbe Reihe mit 5 Löchern. Dann sind sie miteinander verbunden und der Strom läuft einfach daran vorbei. Stecke ein Bauteil lieber über die Rille hinweg.

🔧 **Probier es gleich aus!** In der Übung "Deine erste Breadboard-Schaltung" baust du eine LED mit einem Widerstand auf dem breadboard und lässt sie leuchten und blinken. →

Projekte mit diesem Teil

★ Deine erste Breadboard-Schaltung

Teil 2 von 15



Jumperkabel

Dupont-Kabel: die Drähte, die alles verbinden

In deinem Kit: 40x

Was ist das?

Mit Jumperkabeln verbindest du deinen Arduino mit den Bauteilen. Sie heißen auch **Dupont-Kabel**, wegen der schwarzen Stecker an den Enden. In deinem Kit gibt es zwei Arten:

- **Male-male** (Stift an beiden Seiten): zum Einstecken in den Arduino und das Breadboard.
- **Female-female** (Loch an beiden Seiten): zum Aufstecken auf die Stifte eines Moduls.

So schließt du es an

Stecke das Kabel fest in das Loch oder über den Stift. Kombiniert du ein male- und ein female-Kabel, dann hast du ein male-female-Kabel. Praktisch, um ein Modul direkt an deinen Arduino anzuschließen.

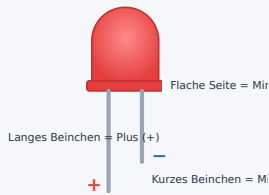
Was kannst du damit machen?

- Deinen Arduino mit dem Breadboard verbinden
- Sensormodule anschließen
- Strom (5V) und GND zu deinen Bauteilen bringen

Tipp

Benutze feste Farben: rot für 5V und schwarz für GND. Dann siehst du sofort, ob der Strom richtig angeschlossen ist.

Teil 3 von 15



LED (Lämpchen)

Ein kleines Lämpchen in 5 Farben

In deinem Kit: 10x

Was ist das?

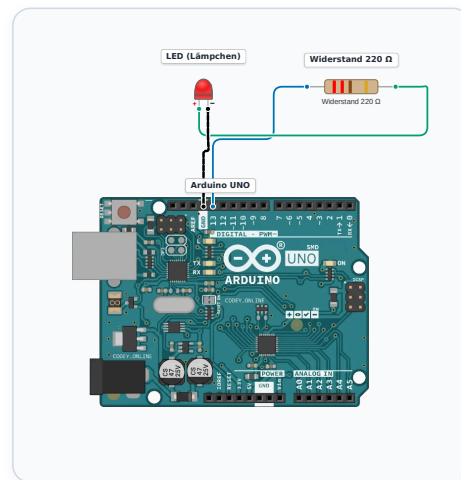
Eine LED ist ein kleines Lämpchen, das sehr wenig Strom braucht. Dein Kit hat 10 LEDs in 5 Farben. Eine LED funktioniert nur in eine Richtung. Du musst sie also richtig herum einstecken.

So schließt du es an

- Das **lange Beinchen** ist Plus (+). Es geht über einen Widerstand von 220 Ω zu einem Pin von deinem Arduino.
- Das **kurze Beinchen** ist Minus (-). Es geht zu GND. Auf dieser Seite ist der Rand des Lämpchens auch flach.

Benutze immer einen Widerstand! Ohne Widerstand bekommt die LED zu viel Strom und geht kaputt.

Schaltplan



- Arduino UNO · D13 → Widerstand 220 Ω · 1
- Widerstand 220 Ω · 2 → LED (Lämpchen) · ANODE
- LED (Lämpchen) · CATHODE → Arduino UNO · GND

Was kannst du damit machen?

- Ein Blinklicht
- Eine Ampel
- Ein Lämpchen, das zeigt, ob etwas an oder aus ist

Tipp

Macht deine LED nichts? Dann dreh sie um. Sie funktioniert nur in eine Richtung.

Projekte mit diesem Teil

★ Deine erste Breadboard-Schaltung

★ Blinklicht

★ Ampel

★ Lichtknopf

★★★ Bewegungsalarm

★★★ Reaktionsspiel

Teil 4 von 15

Rot, rot, braun = 220 Ω

Widerstand 220 Ω

Bremst den Strom

In deinem Kit: 10x

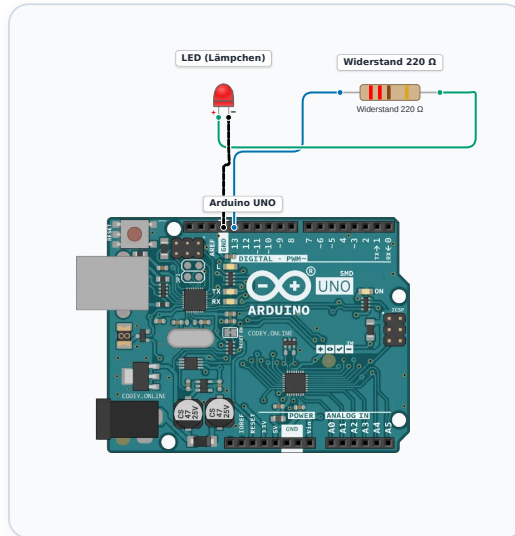
Was ist das?

Ein Widerstand bremst den Strom ein bisschen. Wie ein Wasserhahn, den du halb zudrehst. So bekommt eine LED genau genug Strom und geht nicht kaputt. Die Widerstände in deinem Kit haben 220 Ohm (Ω). Das siehst du an den farbigen Ringen: rot, rot, braun.

So schließt du es an

Ein Widerstand hat kein Plus und kein Minus. Du darfst ihn in beide Richtungen einbauen. Setze ihn zwischen den Pin des Arduino und das lange Beinchen der LED. Eine Seite geht zum Pin, die andere Seite zur LED.

Schaltplan



- Arduino UNO · D13 → Widerstand 220 Ω · 1
- Widerstand 220 Ω · 2 → LED (Lämpchen) · ANODE
- LED (Lämpchen) · CATHODE → Arduino UNO · GND

Was kannst du damit machen?

- Jede LED schützen
- Den Strom zu einem Bauteil begrenzen

Tipp

Ein Widerstand pro LED. Benutzt du drei LEDs? Dann brauchst du auch drei Widerstände.

Projekte mit diesem Teil

★ Deine erste Breadboard-Schaltung

★ Blinklicht

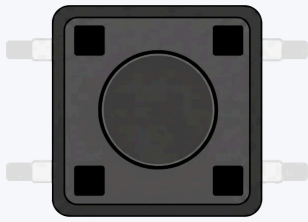
★ Ampel

★ Lichtknopf

★★ Bewegungsalarm

★★ Reaktionsspiel

Teil 5 von 15



Druckknopf

Ein kleiner Knopf zum Drücken

In deinem Kit: 5x

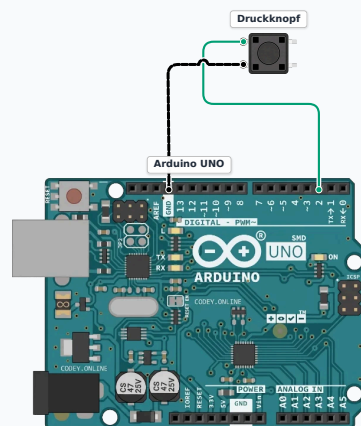
Was ist das?

Ein Druckknopf macht Kontakt, solange du ihn drückst. Lässt du los, ist der Kontakt wieder weg. Dein Arduino kann fühlen, ob der Knopf gedrückt ist. Dein Kit hat 5 Knöpfe.

So schließt du es an

Der Knopf hat 4 Beinchen. Du brauchst nur 2 davon: eins zu einem Pin (zum Beispiel D2) und eins zu GND. Im Code stellst du den Pin auf `INPUT_PULLUP`. Dann ist der Pin HIGH, wenn du nicht drückst, und LOW, wenn du drückst.

Schaltplan



● Arduino UNO · D2 → Druckknopf · 1

● Arduino UNO · GND → Druckknopf · 2

Was kannst du damit machen?

- Ein Lämpchen ein- und ausschalten
- Eine Türklingel
- Ein Spiel: Wer drückt am schnellsten?

Tipp

Nicht sicher, welche Beinchen? Nimm zwei Beinchen, die schräg gegenüberliegen. Dann ist es immer richtig. Setze den Knopf über den Spalt deines breadboard.

Projekte mit diesem Teil

★ Lichtknopf

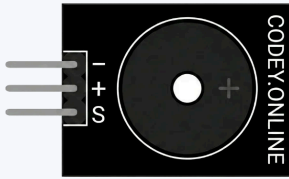
★ Türklingel

★★ Schranke

★★ Reaktionsspiel

★★ Elektronischer Würfel

Teil 6 von 15



Aktiver Summer

Macht Pieptöne

In deinem Kit: 1x

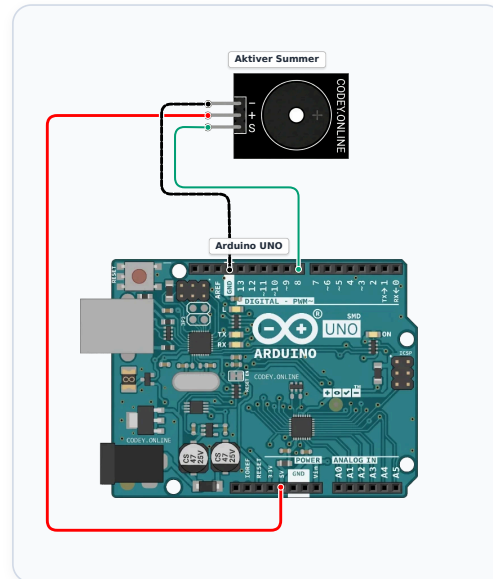
Was ist das?

Der Summer macht ein Piepsgeräusch. Es ist ein **aktiver** Summer: Er piept von selbst, sobald er Strom bekommt. Du musst ihn nur ein- und ausschalten, genau wie ein Lämpchen.

So schließt du es an

- **S** (Signal) zu einem Pin, zum Beispiel D8
- **+** zu 5V
- **-** zu GND

Schaltplan



● Arduino UNO · GND → Aktiver Summer · -

● Arduino UNO · 5V → Aktiver Summer · +

● Arduino UNO · D8 → Aktiver Summer · S

Was kannst du damit machen?

- Ein Alarm
- Ein Parksensord, der schneller piept, wenn du näher kommst
- Eine Türklingel oder ein Timer

Tipp

Möchtest du einen kurzen Piepton? Schalte den Summer ein, warte 100 Millisekunden und schalte ihn wieder aus.

Projekte mit diesem Teil

★ Türklingel

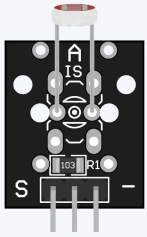
★★ Parksensord

★★ Bewegungsalarm

★★★★ Parksensord deluxe

★★★★ Automatische Schranke

Teil 7 von 15



Lichtsensord (LDR)

Misst, wie hell oder dunkel es ist

In deinem Kit: 1x

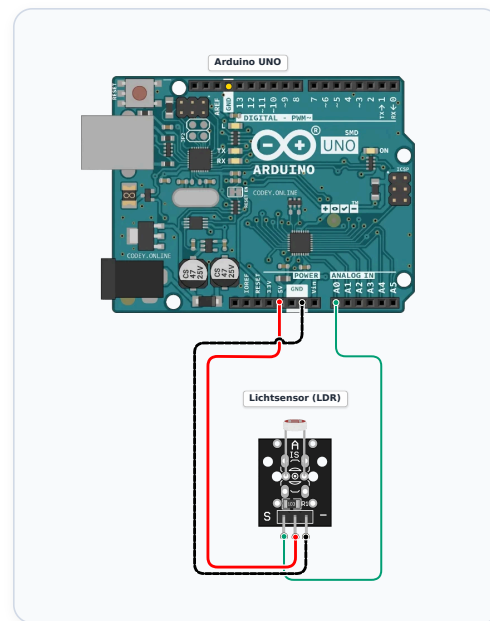
Was ist das?

Dieser Sensor misst, wie viel Licht da ist. Dein Arduino liest eine Zahl zwischen 0 und 1023. Bei viel Licht bekommst du eine andere Zahl als im Dunkeln. So weiß dein Arduino, ob Tag oder Nacht ist.

So schließt du es an

- S zu einem analogen Pin, zum Beispiel A0
- + zu 5V
- - zu GND

Schaltplan



● Arduino UNO · A0 → Lichtsensor (LDR) · S

● Arduino UNO · 5V → Lichtsensor (LDR) · +

● Arduino UNO · GND → Lichtsensor (LDR) · -

Was kannst du damit machen?

- Ein Nachtlämpchen, das von selbst angeht
- Ein Lichtmesser
- Ein Alarm, der losgeht, wenn jemand die Schublade öffnet

Tipp

Schau im seriellen Monitor, welche Zahl du siehst. Halte dann deine Hand über den Sensor. So siehst du, welche Zahl zu "dunkel" gehört.

Projekte mit diesem Teil

★★ Nachtlämpchen

Teil 8 von 15



Temperatursensor DHT11

Misst die Temperatur und die Luftfeuchtigkeit

In deinem Kit: 1x

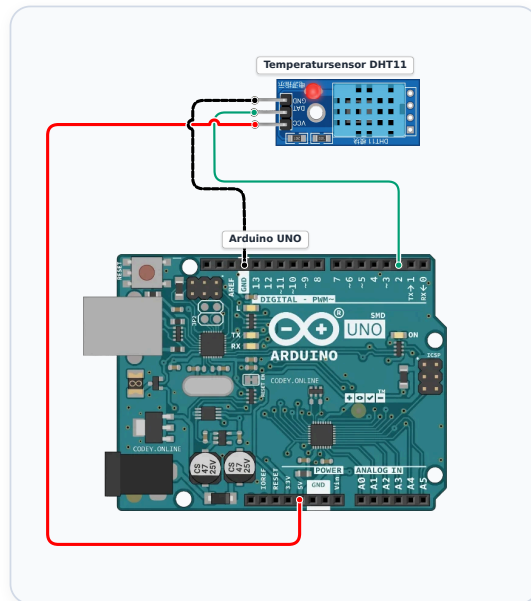
Was ist das?

Der DHT11 misst, wie warm es ist (in Grad Celsius) und wie feucht die Luft ist (in Prozent). Er gibt alle 2 Sekunden eine neue Messung.

So schließt du es an

- **DAT** (Daten) zu einem Pin, zum Beispiel D2
- **VCC** zu 5V
- **GND** zu GND

Schaltplan



- Arduino UNO · 5V → Temperatursensor DHT11 · VCC
- Arduino UNO · GND → Temperatursensor DHT11 · GND
- Arduino UNO · D2 → Temperatursensor DHT11 · DAT

Was kannst du damit machen?

- Eine Mini-Wetterstation
- Ein Ventilator, der angeht, wenn es zu warm ist
- Eine Warnung, wenn es im Badezimmer zu feucht ist

Tipp

Atme einmal auf den Sensor. Dann siehst du die Luftfeuchtigkeit sofort steigen!

Projekte mit diesem Teil

★★ Mini-Wetterstation

★★★★ Smarter Ventilator

Teil 9 von 15



Abstandssensor HC-SR04

Misst Abstand mit Schall, genau wie eine Fledermaus

In deinem Kit: 1x

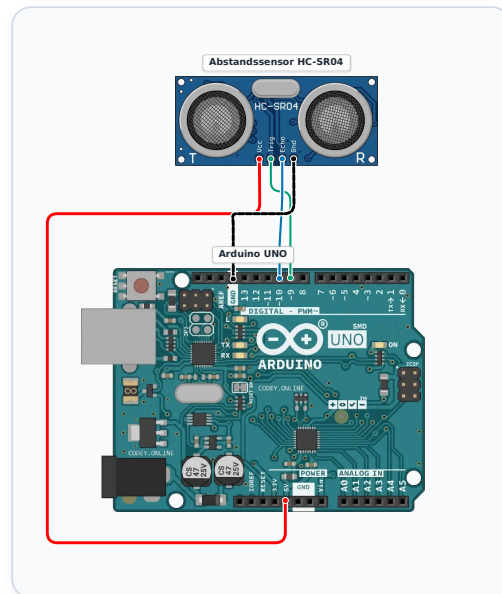
Was ist das?

Dieser Sensor sendet einen kleinen Ton aus, den du nicht hören kannst. Der Ton prallt gegen etwas und kommt zurück. Der Sensor misst, wie lange das dauert. So weiß dein Arduino, wie weit etwas weg ist: von 2 cm bis ungefähr 4 meter.

So schließt du es an

- **Trig** (senden) naar D9
- **Echo** (empfangen) naar D10
- **Vcc** naar 5V
- **Gnd** naar GND

Schaltplan



- Arduino UNO · 5V → Abstandssensor HC-SR04 · Vcc
- Arduino UNO · GND → Abstandssensor HC-SR04 · Gnd
- Arduino UNO · D9 → Abstandssensor HC-SR04 · Trig
- Arduino UNO · D10 → Abstandssensor HC-SR04 · Echo

Was kannst du damit machen?

- Ein Parksensor
- Ein Abstandsmesser
- Eine Schranke, die von selbst aufgeht

Tipp

Die zwei "Augen" sind der Lautsprecher und das Mikrofon. Richte sie gerade auf das, was du messen möchtest.

Projekte mit diesem Teil

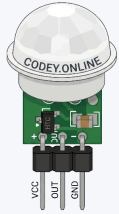
★★★ Parksensor

★★★★ Parksensor deluxe

★★★★ Abstandsmesser mit Lichtbalken

★★★★ Automatische Schranke

Teil 10 von 15



Bewegungssensor (PIR)

Sieht, ob sich jemand bewegt

In deinem Kit: 1x

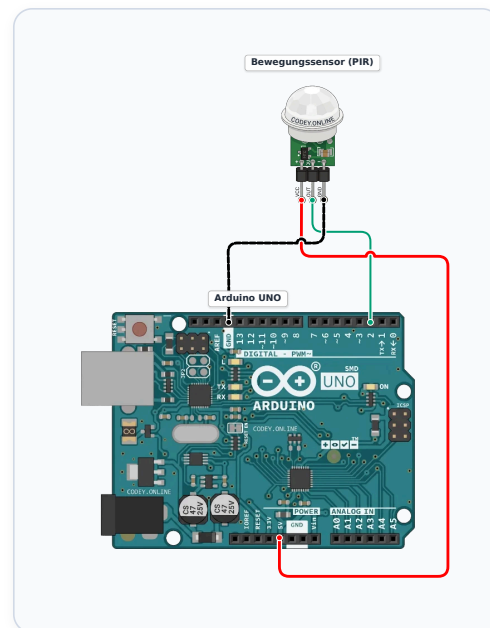
Was ist das?

Ein PIR-Sensor spürt die Wärme von Menschen und Tieren. Bewegt sich etwas Warmes vor dem Sensor? Dann wird sein Ausgang HIGH. So funktionieren auch Lampen, die von selbst angehen, wenn du vorbeiläufst.

So schließt du es an

- **OUT** naar een pin, zum Beispiel D2
- **VCC** naar 5V
- **GND** naar GND

Schaltplan



● Arduino UNO · 5V → Bewegungssensor (PIR) · VCC

● Arduino UNO · GND → Bewegungssensor (PIR) · GND

● Arduino UNO · D2 → Bewegungssensor (PIR) · OUT

Was kannst du damit machen?

- Eine Alarmanlage
- Eine Lampe, die angeht, wenn du hereinkommst
- Zählen, wie oft jemand vorbeiläuft

Tipp

Der Sensor braucht nach dem Start ein paar Sekunden, um sich an den Raum zu gewöhnen. Bleib dann kurz still stehen.

Projekte mit diesem Teil

★★ Bewegungsalarm

Teil 11 von 15



OLED-Display 0,91"

Ein kleiner Bildschirm für Text und Zahlen

In deinem Kit: 1x

Was ist das?

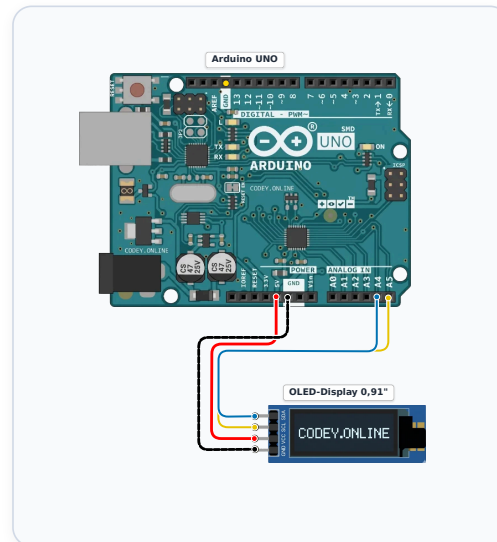
Ein kleiner, scharfer Bildschirm mit 128 mal 32 pixel. Du kannst Text, Zahlen und einfache Zeichnungen darauf anzeigen. Zum Beispiel die Temperatur oder einen Abstand.

So schließt du es an

Der Bildschirm spricht über **I2C**: nur zwei Datenleitungen.

- **SDA** naar A4
- **SCL** naar A5
- **VCC** naar 5V
- **GND** naar GND

Schaltplan



- Arduino UNO · A4 → OLED-Display 0,91" · SDA
- Arduino UNO · A5 → OLED-Display 0,91" · SCL
- Arduino UNO · 5V → OLED-Display 0,91" · VCC
- Arduino UNO · GND → OLED-Display 0,91" · GND

Was kannst du damit machen?

- Die Temperatur anzeigen
- Den Abstand in Zentimetern zeigen
- Einen Punktestand oder eine Reaktionszeit zeigen

Tipp

Bleibt der Bildschirm schwarz? Prüfe, ob SDA und SCL nicht vertauscht sind.

Projekte mit diesem Teil

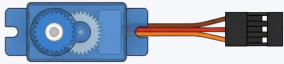
★★ Mini-Wetterstation

★★ Reaktionsspiel

★★★★ Parksensor deluxe

★★★★ Smarter Ventilator

Teil 12 von 15



Servomotor SG90

Ein kleiner Motor, der in eine bestimmte Stellung dreht

In deinem Kit: 1x

Was ist das?

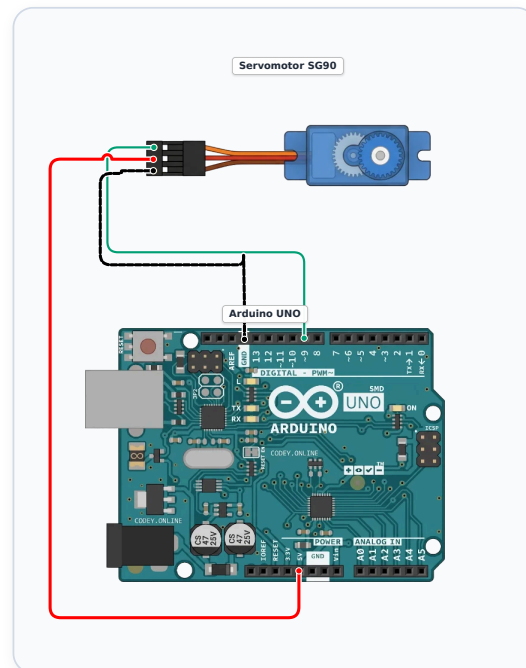
Ein Servo ist ein kleiner Motor, den du in eine genaue Stellung drehen kannst, von 0 bis 180 grad. Du sagst zum Beispiel "gehe auf 90 grad" und er geht genau dorthin. Auf die Achse steckst du einen kleinen Arm.

So schließt du es an

Der Servo hat drei Kabel:

- **Orange oder gelb** (Signal) naar een pin, zum Beispiel D9
- **Rot** naar 5V
- **Braun** naar GND

Schaltplan



● Arduino UNO · GND → Servomotor SG90 · GND

● Arduino UNO · 5V → Servomotor SG90 · VCC

● Arduino UNO · D9 → Servomotor SG90 · PWM

Was kannst du damit machen?

- Eine Schranke
- Ein Zeiger, der die Temperatur anzeigt
- Ein winkender Roboterarm

Tipp

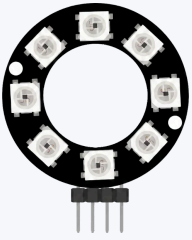
Dreh den kleinen Arm nicht mit der Hand. Dann können die Zahnräder innen kaputtgehen.

Projekte mit diesem Teil

★★ Schranke

★★★★ Automatische Schranke

Teil 13 von 15



RGB-LED-Ring (8 LEDs)

8 schlaue Lämpchen in jeder Farbe

In deinem Kit: 1x

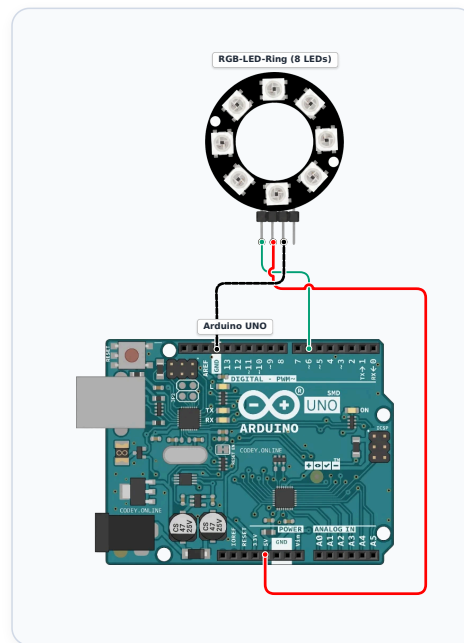
Was ist das?

Ein Ring mit 8 schlaun Lämpchen. Jedes Lämpchen kann jede Farbe haben. Und du steuerst jedes einzeln. Mit nur einem Datenkabel! Super toll für Lichteffekte.

So schließt du es an

- **DI** (data in) an einen Pin, zum Beispiel D6
- **5V** an 5V
- **GND** an GND
- **DO** lässt du frei (dort kannst du noch einen Ring dahinter anschließen)

Schaltplan



- Arduino UNO · 5V → RGB-LED-Ring (8 LEDs) · 5V
- Arduino UNO · GND → RGB-LED-Ring (8 LEDs) · GND
- Arduino UNO · D6 → RGB-LED-Ring (8 LEDs) · DI

Was kannst du damit machen?

- Eine Regenbogenlampe
- Ein Lichtbalken, der zeigt, wie nah etwas ist
- Ein Würfel aus Licht

Tipp

Stelle die Helligkeit im Code nicht zu hoch ein. Das ist angenehmer für deine Augen und für deinen Arduino.

Projekte mit diesem Teil

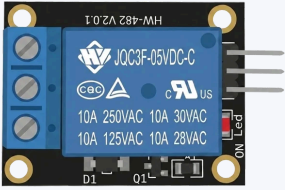
★ Regenbogenring

★★ Nachtlämpchen

★★ Elektronischer Würfel

★★★★ Abstandsmesser mit Lichtbalken

Teil 14 von 15



Relais 5V

Ein Schalter, den dein Arduino umlegen kann

In deinem Kit: 1x

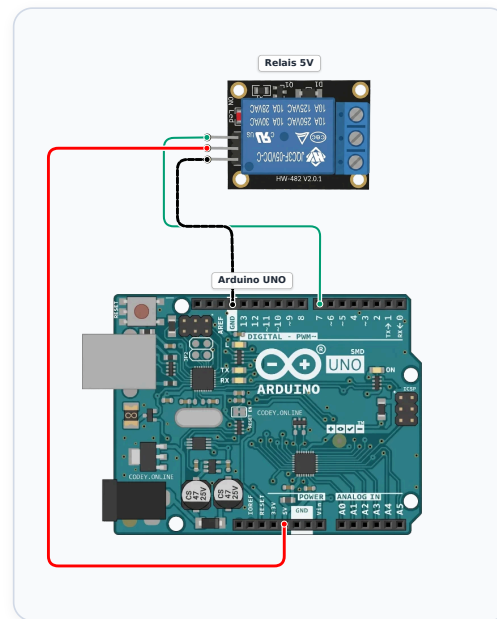
Was ist das?

Ein Relais ist ein Schalter, den dein Arduino an- und ausschalten kann. Du hörst es „klicken“. Mit einem Relais kann dein Arduino ein anderes Gerät an- und ausschalten, zum Beispiel einen kleinen Ventilator oder eine Lampe mit Batterie.

So schließt du es an

- S an einen Pin, zum Beispiel D7
- + an 5V
- – an GND
- Das Gerät, das du schalten möchtest, schließt du an **COM** und **NO** an. NO bedeutet: normal aus, an wenn das Relais klickt.

Schaltplan



● Arduino UNO · GND → Relais 5V · -

● Arduino UNO · 5V → Relais 5V · +

● Arduino UNO · D7 → Relais 5V · S

Was kannst du damit machen?

- Ein Ventilator, der angeht, wenn es warm ist
- Eine Lampe mit Timer
- Eine kleine Pumpe für die Pflanzen

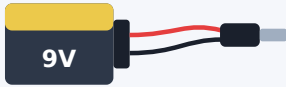
Tipp

Sicherheit geht vor: Verwende das Relais nur mit Batterien oder niedriger Spannung (bis 12V). Niemals mit der Steckdose (230V)!

Projekte mit diesem Teil

★★★★ Smarter Ventilator

Teil 15 von 15



9V-Batterieclip

Dein Projekt ohne Computer laufen lassen

In deinem Kit: 1×

Was ist das?

Mit diesem Clip schließt du eine 9V-Batterie an deinen Arduino an. Dann funktioniert dein Projekt auch ohne Computer. Praktisch, wenn du es zeigen oder irgendwo hinstellen möchtest.

So schließt du es an

Klicke die Batterie an den Clip und stecke den runden Stecker in den schwarzen Stromanschluss deines Arduino. Dein Code ist noch auf dem Arduino. Darum startet er sofort.

Was kannst du damit machen?

- Deinen Parksensord in die Garage hängen
- Deine Wetterstation auf die Fensterbank stellen
- Dein Projekt Freunden zeigen

Tipp

Nimm die Batterie ab, wenn du fertig bist. Dann hält sie viel länger.

Was ist Vibe Coding?

Früher musstest du erst monatelang programmieren lernen. Jetzt sagst du einfach mit deinen eigenen Worten, was du bauen möchtest. Die KI schreibt den Code für dich.

Das heißt **Vibe Coding**: Du hast die Idee, die KI macht die Tipparbeit. Du sagst zum Beispiel: *"Mach ein Nachtlicht, das angeht, wenn es dunkel wird."* Nach ein paar Sekunden ist der Code fertig, mit einem Anschlussplan dazu.

Die KI von heute ist sehr gut im Programmieren. Oft sogar besser und schneller als ein Mensch. Sie kennt alle Teile, macht weniger Tippfehler und erklärt alles ganz ruhig.

Ein Projekt bauen: früher und heute

Selbst programmieren

Herausfinden, tippen, Fehler suchen, noch einmal versuchen...

± 10 Stunden

Mit Codey

Sagen, was du möchtest, hochladen, fertig.

± 5 Minuten ↗

So funktioniert es



Du erzählst, was du bauen möchtest.



Codey schreibt den Code und zeichnet, wie du alles anschließt.



Du baust es nach und lädst den Code hoch.



Möchtest du etwas anderes? Sag es, dann passt Codey es an.

Du lernst dabei auch noch etwas! Codey erklärt, was der Code macht. So verstehst du immer besser, wie es funktioniert.

Erstelle dein kostenloses Konto

Um mit Codey zu arbeiten, brauchst du ein Konto. Das ist kostenlos und dauert nur eine Minute. Mit deinem Starterkit kannst du Codey danach unbegrenzt nutzen.

1

Registrieren

Klicke auf "Konto erstellen" und gib deine E-Mail-Adresse und ein Passwort ein. Hast du ein Google-Konto? Klicke dann auf "Weiter mit Google", dann bist du sofort fertig.

2

Bestätige deine E-Mail-Adresse

Du bekommst eine E-Mail von Codey Online. Klicke auf den Link in dieser E-Mail. Siehst du keine E-Mail? Schau dann auch kurz in deinem Spam-Ordner nach.

3

Einloggen

Logge dich mit deiner E-Mail-Adresse und deinem Passwort ein. Du kommst dann automatisch zurück zu diesem Tutorial, zum nächsten Schritt.

Ich habe schon ein Konto: einloggen →

👉 Gehe zu codey.online/uno, um dein kostenloses Konto zu erstellen.

👤 Bist du noch jung? Erstelle das Konto dann zusammen mit deinem Vater, deiner Mutter oder einem anderen Erwachsenen.

So funktioniert Codey

Codey ist dein Programmier-Kumpel. Alles ist auf einem Bildschirm. Das sind die wichtigsten Knöpfe.

The screenshot shows the Codey Online web interface. On the left is a sidebar with a chat window, project files, and navigation links. The main area is divided into a code editor on the left and a chat/output area on the right. The code editor shows a C++ sketch for a temperature and humidity sensor. The chat area shows a list of messages and a button to open the serial monitor. Numbered callouts 1 through 6 highlight specific features: 1. Chat, 2. Code editor, 3. Compile and upload button, 4. Connection diagram, 5. Serial monitor, and 6. My kit button.

1 Der Chat

Hier sprichst du mit Codey. Schreibe, was du bauen oder ändern möchtest, in ganz normaler Sprache.

2 Der Code

Hier steht der Code, den Codey für dich schreibt. Du darfst auch selbst etwas hineinschreiben.

3 Kompilieren & auf Board hochladen

Ein Knopf für alles: Codey prüft deinen Code, macht ihn fertig (kompilieren) und schickt ihn über das USB-Kabel an deinen Arduino (hochladen). Gibt es einen Fehler? Dann löst Codey ihn oft selbst.

4 Anschlussplan

Zeigt, welches Kabel wohin muss. Du siehst ihn im Chat; klicke darauf, um ihn groß zu sehen.

5 Serieller Monitor

Hier siehst du die Nachrichten, die dein Arduino sendet, zum Beispiel die Temperatur oder einen Abstand.

6 Mein Kit

Alle Teile aus deinem Starterkit. Codey benutzt nur Dinge, die du hast.

Gut zu wissen: Mit deinem Starterkit kannst du Codey unbegrenzt in deinen Kit-Chats benutzen. Du kannst Codey alles fragen, auch "Was macht diese Zeile?" oder "Warum funktioniert es nicht?".

Anschließen und hochladen

Dein Code ist fertig? Dann kommt er jetzt auf deinen Arduino. Das machst du in drei Schritten.

Wichtig: benutze **Google Chrome** oder **Microsoft Edge** auf deinem Computer. Nur diese Browser dürfen über USB mit deinem Arduino sprechen. Safari, Firefox und die meisten Handys können das (noch) nicht.

1 Schließe deinen Arduino an

Stecke das USB-Kabel in deinen Arduino und in deinen Computer. Ein grünes Lämpchen (ON) geht an.

2 Klicke auf Kompilieren & auf Board hochladen

Codey prüft deinen Code und macht ihn bereit (kompilieren). Beim ersten Mal dauert das manchmal ein Minütchen.

3 Wähle deinen Arduino im Port-Fenster

Ein Fenster mit Ports erscheint. Wähle deinen Arduino und klicke auf Verbinden. Die Lämpchen RX und TX blinken kurz: Dein Code geht hinein (hochladen). Fertig!

Wie heißt mein Arduino im Port-Fenster?

Das hängt von deinem Computer ab. Siehst du mehrere? Zieh das Kabel kurz heraus und schau, welcher verschwindet. Das ist dein Arduino.

Windows

Ein **COM-Port** mit einer Nummer, zum Beispiel `Arduino Uno (COM4)` oder `USB-SERIAL CH340 (COM5)`. COM1 ist fast nie dein Arduino.

Mac

Etwas mit `usbmodem` oder `usbserial`, zum Beispiel `/dev/cu.usbmodem14101`, `cu.usbserial-1420` oder `cu.wchusbserial1420`. Wähle immer die `cu.`-Version.

Linux

`/dev/ttyACM0` oder `/dev/ttyUSB0`. Kein Zugriff? Füge dich selbst zur Gruppe `dialout` hinzu und melde dich neu an.

Chromebook

Chrome zeigt den Namen des Geräts, zum Beispiel `Arduino Uno`, `USB2.0-Serial` oder `USB Serial`.

Serieller Monitor: Manche Projekte senden Nachrichten an deinen Computer, zum Beispiel die Temperatur. Du siehst sie, wenn du in Codey auf Serieller Monitor klickst. So siehst du, was dein Arduino "denkt".

Klappt es nicht? Keine Panik!

▼ Ich sehe meinen Arduino nicht im Port-Fenster.

Probiere ein anderes USB-Kabel. Manche Kabel können nur laden und keine Daten senden. Probiere auch einen anderen USB-Port an deinem Computer.

▼ Ich sehe CH340 oder USB2.0-Serial, aber das Hochladen klappt nicht (Windows/Mac).

Dann benutzt dein Arduino einen CH340-Chip. Installiere einmal den kostenlosen CH340-Treiber (suche nach "CH340 driver") und starte deinen Computer neu.

▼ Das Hochladen bricht mittendrin ab.

Schließe andere Programme, die den Arduino benutzen, zum Beispiel die Arduino IDE oder einen zweiten Codey-Tab. Zieh das Kabel kurz heraus, stecke es wieder ein und versuche es noch einmal.

▼ Es funktioniert, aber meine Schaltung macht nichts.

Kontrolliere die Kabel mit dem Anschlussplan. Sind 5V und GND richtig? Frage danach Codey: "Meine LED geht nicht an, woran kann das liegen?"

Los geht's!

Wähle ein Projekt und klicke auf MACH DAS. Codey legt den Code und den Anschlussplan für dich bereit. Beginne mit einem Projekt mit einem Stern ★, wenn du noch nie etwas mit Arduino gemacht hast.



★ ⌚ 10 Minuten

Deine erste Breadboard-Schaltung

Übe mit dem Breadboard: Lass eine LED leuchten und danach blinken.



★ ⌚ 10 Minuten

Blinklicht

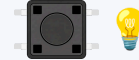
Dein allererstes Projekt: eine LED blinken lassen.



★ ⌚ 15 Minuten

Ampel

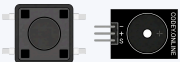
Drei LEDs, die wie eine echte Ampel arbeiten.



★ ⌚ 15 Minuten

Lichtknopf

Drück auf den Knopf, und die Lampe geht an. Noch einmal: aus.



★ ⌚ 10 Minuten

Türklingel

Drück auf den Knopf für ein fröhliches Ding-Dong.



★ ⌚ 10 Minuten

Regenbogenring

Der LED-Ring dreht sich in allen Farben des Regenbogens.



★★ ⌚ 20 Minuten

Mini-Wetterstation

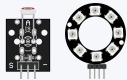
Temperatur und Luftfeuchtigkeit auf deinem eigenen kleinen Display.



★★ ⌚ 20 Minuten

Parksensor

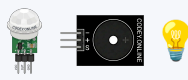
Je näher, desto schneller piept es. Wie in einem Auto.



★★ ⌚ 15 Minuten

Nachtlämpchen

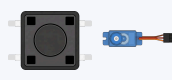
Wird es dunkel? Dann geht die Lampe von selbst an.



★★ ⌚ 20 Minuten

Bewegungsalarm

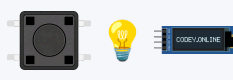
Läuft jemand vorbei? Dann geht der Alarm los!



★★ ⌚ 20 Minuten

Schranke

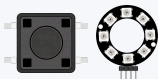
Drück auf den Knopf und die Schranke geht auf.



★★ ⌚ 25 Minuten

Reaktionsspiel

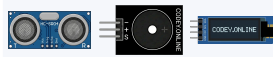
Wie schnell bist du? Miss deine Reaktionszeit.



★★ ⌚ 15 Minuten

Elektronischer Würfel

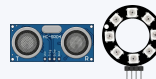
Drück auf den Knopf und würfle mit Licht.



★★★ ⌚ 30 Minuten

Parksensor deluxe

Piepen und der Abstand in Zentimetern auf einem kleinen Bildschirm.



★★★★ ⌚ 25 Minuten

Abstandsmesser mit Lichtbalken

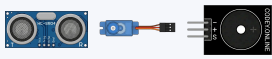
Je näher, desto mehr Lämpchen: grün, orange, rot.



★★★★ ⌚ 30 Minuten

Smarter Ventilator

Zu warm? Dann schaltet das Relais den Ventilator ein.



★★★★ ⌚ 30 Minuten

Automatische Schranke

Kommt ein Auto? Dann öffnet sich die Schranke von selbst.

Hast du selbst eine Idee? Noch besser! Öffne Codey und erzähle, was du bauen möchtest. Zum Beispiel: "Baue ein Lämpchen, das schneller blinkt, wenn es wärmer wird."

★ Anfänger ⌚ 10 Minuten

Deine erste Breadboard-Schaltung

Eine Übung, damit du merkst, wie ein Breadboard funktioniert. Du baust eine LED mit einem Widerstand auf dem Breadboard. Erst lässt du sie mit der 5V-Leiste leuchten, ohne Code. Danach lässt du sie mit pin 13 blinken.

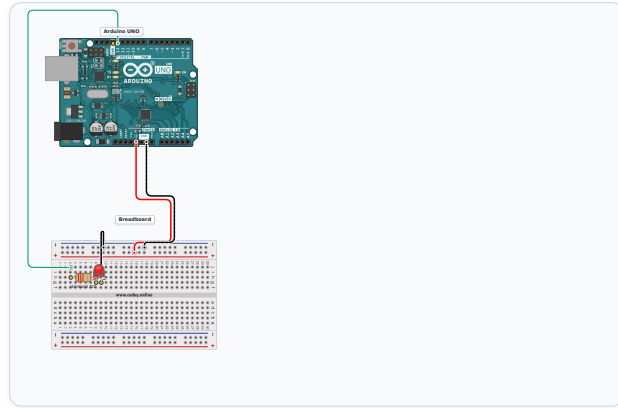
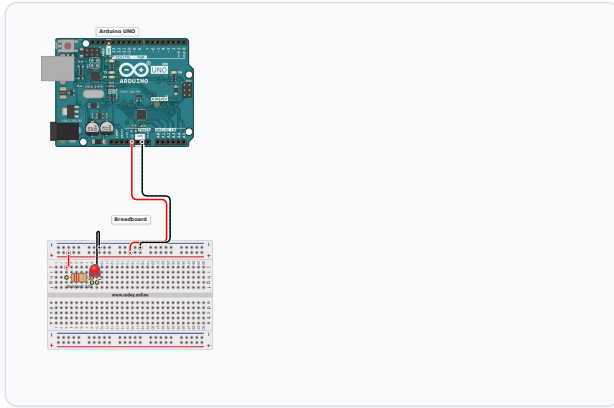
Schaltplan

Schritt A: Lass die LED leuchten (ohne Code)

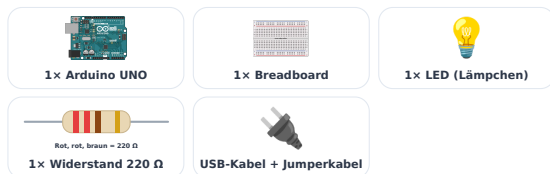
Das rote Kabel geht von der +Leiste zu Reihe 3. Der Strom läuft dann so: 5V → +Leiste → Widerstand → LED → -Leiste → GND. Leuchtet die LED? Dann hast du das Breadboard gut benutzt!

Schritt B: Lass die LED mit pin 13 blinken

Nimm das rote Kabel zwischen der +Leiste und Reihe 3 heraus. Stecke stattdessen ein Kabel von pin 13 zu Reihe 3. Jetzt bestimmt der Arduino, wann die LED Strom bekommt.



Was brauchst du?



So baust du es

1. Verbinde 5V vom Arduino mit der +-Leiste oben am Breadboard (rotes Kabel).
2. Verbinde GND vom Arduino mit der --Leiste daneben (schwarzes Kabel). Jetzt hast du zwei Stromleisten.
3. Stecke den Widerstand von 220 Ohm von Loch H3 zu Loch H8. Er liegt dann in zwei verschiedenen Reihen.
4. Stecke das lange Beinchen der LED in G8 (dieselbe Reihe wie der Widerstand) und das kurze Beinchen in G9.
5. Verbinde Loch J9 mit der --Leiste (schwarzes Kabel). So kann der Strom zurück zu GND.
6. Schritt A: Verbinde die +-Leiste mit Loch J3 (rotes Kabel). Die LED leuchtet!
7. Schritt B: Nimm dieses rote Kabel heraus und verbinde pin 13 mit Loch J3 (grünes Kabel).
8. Lass Codey den Code machen: Klicke auf MACH DAS, oder tippe selbst in den Chat: "Lass die LED an pin 13 jede Sekunde blinken." Klicke danach auf Kompilieren & auf Board hochladen.

Der Code

```
// Deine erste Breadboard-Schaltung
// Die LED auf dem Breadboard geht an und aus. Sie bekommt Strom über pin 13.

const int LED_PIN = 13; // Das grüne Kabel geht von pin 13 zum Breadboard

void setup() {
  pinMode(LED_PIN, OUTPUT); // Pin 13 ist ein Ausgang: Er gibt Strom
}

void loop() {
  digitalWrite(LED_PIN, HIGH); // LED an
  delay(1000); // 1 Sekunde warten
  digitalWrite(LED_PIN, LOW); // LED aus
  delay(1000); // 1 Sekunde warten
}
```

Was macht der Code?

- Pin 13 ist ein Ausgang. Mit **HIGH** gibt er 5V, mit **LOW** gibt er nichts.
- In **loop()** schalten wir die LED ein, warten 1 Sekunde, schalten sie aus und warten wieder.
- Der Strom läuft über das Breadboard: pin 13 → Reihe 3 → Widerstand → Reihe 8 → LED → Reihe 9 → --Leiste → GND.

Probier mal ...

- Lass die LED doppelt so schnell blinken
- Füge eine zweite LED auf dem Breadboard hinzu, die abwechselnd blinkt
- Lass die LED dreimal kurz blinken und dann kurz aus bleiben

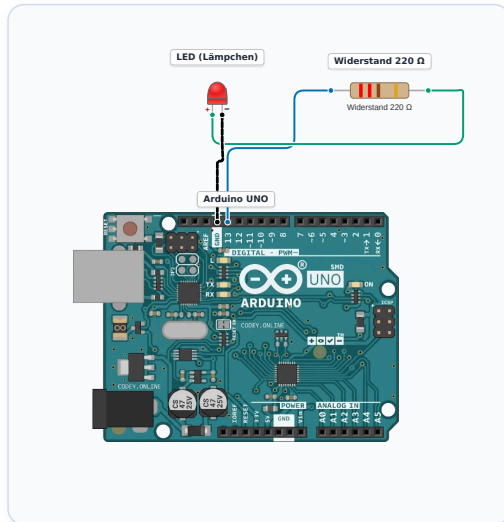
🔧 Baue dieses Projekt mit Codey: Öffne codey.online/de/starter-kits/uno/projects/breadboard-led und klicke auf BAU DAS. Dann sind der Code und der Schaltplan sofort bereit.

★ Anfänger ⌚ 10 Minuten

Blinklicht

Dein erstes Projekt! Eine LED geht immer an und aus, wie ein Blinklicht. So lernst du, wie der Arduino einen Ausgang ein- und ausschaltet.

Schaltplan



Was brauchst du?



1× Arduino UNO



1× LED (Lämpchen)

1× Widerstand 220 Ω USB-Kabel +
Jumperkabel

So baust du es

1. Stecke die LED ins breadboard, mit den Beinchen in zwei verschiedene Reihen.
2. Stecke einen Widerstand von 220 Ω von der Reihe mit dem langen Beinchen zu einer anderen Reihe.
3. Verbinde diese andere Reihe mit einem Kabel mit Pin 13 deines Arduino.
4. Verbinde das kurze Beinchen mit einem Kabel mit GND.

Der Code

```
// Blinklicht
// Eine LED geht immer an und aus, wie ein Blinklicht.

const int LED_PIN = 13; // Die LED ist an Pin 13 (über den Widerstand von 220 ohm)
const int WARTEZEIT = 500; // Wie lange die LED an oder aus bleibt (in Millisekunden)

void setup() {
  // setup() läuft einmal, wenn der Arduino startet
  pinMode(LED_PIN, OUTPUT); // Pin 13 ist ein Ausgang: Er gibt Strom
}

void loop() {
  // loop() läuft immer wieder, für immer
  digitalWrite(LED_PIN, HIGH); // LED an
  delay(WARTEZEIT); // Kurz warten
  digitalWrite(LED_PIN, LOW); // LED aus
  delay(WARTEZEIT); // Kurz warten
}
```

Was macht der Code?

- In `setup()` sagen wir, dass Pin 13 ein Ausgang ist.
- In `loop()` schalten wir die LED ein, warten kurz, schalten sie aus und warten wieder.
- `loop()` beginnt danach einfach wieder von vorn. So blinkt die LED weiter.

Probier mal ...

- Lass die LED doppelt so schnell blinken
- Mach daraus ein SOS-Signal: kurz-kurz-kurz, lang-lang-lang, kurz-kurz-kurz
- Lass die LED langsam hell und dunkel werden, als würde sie atmen



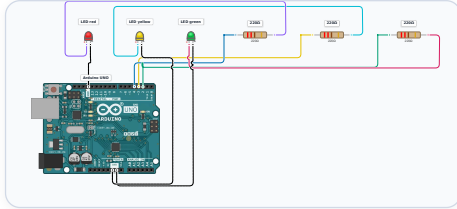
Baue dieses Projekt mit Codey: Öffne codey.online/de/starter-kits/uno/projects/blink und klicke auf BAU DAS. Dann sind der Code und der Schaltplan sofort bereit.

★ Anfänger ⌚ 15 Minuten

Ampel

Du baust eine echte Ampel mit einer roten, einer gelben und einer grünen LED. Sie gehen nacheinander an, genau wie auf der Straße.

Schaltplan



Was brauchst du?



1× Arduino UNO



3× LED (Lämpchen)



3× Widerstand 220 Ω

USB-Kabel +
Jumperkabel

So baust du es

1. Stecke die drei LEDs nebeneinander in das breadboard.
2. Gib jeder LED einen eigenen Widerstand von 220 Ω zu ihrem Pin:
rot zu 4, gelb zu 3, grün zu 2.
3. Verbinde alle kurzen Beinchen mit der Minus-Leiste (–) des breadboards.
4. Verbinde die Minus-Leiste mit GND von deinem Arduino.

Der Code

```
// Ampel
// Drei LEDs machen genau das, was eine echte Ampel macht: grün, orange, rot.

const int ROT = 4;    // Rote LED an Pin 4
const int ORANGE = 3; // Gelbe/orange LED an Pin 3
const int GRUEN = 2;  // Grüne LED an Pin 2

void setup() {
  pinMode(ROT, OUTPUT);
  pinMode(ORANGE, OUTPUT);
  pinMode(GRUEN, OUTPUT);
}

// Schaltet ein Lämpchen an und die anderen zwei aus
void setzeLicht(int an) {
  digitalWrite(ROT, an == ROT ? HIGH : LOW);
  digitalWrite(ORANGE, an == ORANGE ? HIGH : LOW);
  digitalWrite(GRUEN, an == GRUEN ? HIGH : LOW);
}

void loop() {
  setzeLicht(GRUEN); // Grün: du darfst fahren
  delay(5000);
  setzeLicht(ORANGE); // Orange: pass auf, es wird rot
  delay(2000);
  setzeLicht(ROT); // Rot: stehen bleiben!
  delay(5000);
}
```

Was macht der Code?

- Jede LED hat einen eigenen Pin: rot auf 4, gelb auf 3 und grün auf 2.
- Die Funktion `setzeLicht()` schaltet ein Lämpchen an und die anderen zwei aus.
- In `loop()` gehen wir von grün zu orange zu rot, mit einer Wartezeit dazwischen.

Probier mal ...

- Lass das orange Lämpchen dreimal blinken, bevor es rot wird
- Füge einen Knopf hinzu: Wenn du drückst, springt das Licht schneller auf rot (ein Fußgängerknopf)
- Mach einen Nachtmodus daraus, in dem nur das orange Lämpchen blinkt

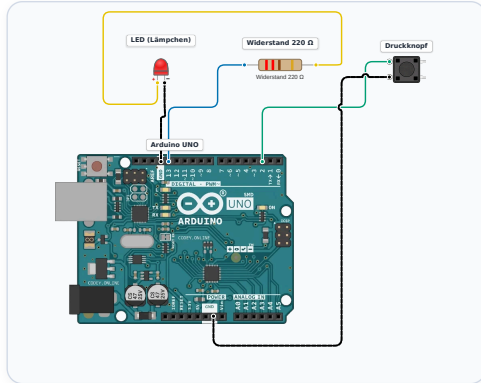
🚀 Baue dieses Projekt mit Codey: Öffne codey.online/de/starter-kits/uno/projects/traffic-light und klicke auf BAU DAS. Dann sind der Code und der Schaltplan sofort bereit.

★ Anfänger ⌚ 15 Minuten

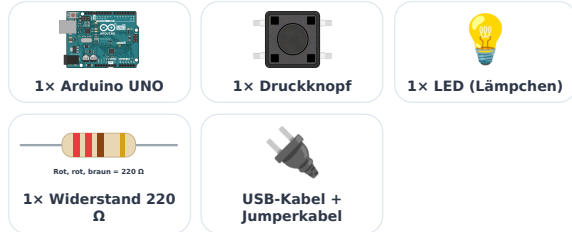
Lichtknopf

Du baust einen Lichtknopf. Einmal drücken: Die LED geht an. Noch einmal drücken: Die LED geht aus. So lernst du, wie der Arduino einen Knopf "fühlt".

Schaltplan



Was brauchst du?



So baust du es

1. Setze den Druckknopf über die Rille deines Breadboard.
2. Verbinde ein Beinchen mit Pin 2 und ein Beinchen auf der anderen Seite mit GND.
3. Setze die LED mit einem Widerstand von 220 Ω an Pin 13, das kurze Beinchen zu GND.

Der Code

```
// Lichtknopf
// Drücke einmal auf den Knopf: Die LED geht an. Noch einmal: Die LED geht aus.

const int KNOPF_PIN = 2; // Der Druckknopf sitzt an Pin 2 (andere Seite zu GND)
const int LED_PIN = 13; // Die LED sitzt an Pin 13 (über den Widerstand)

bool ledAn = false; // Merkt sich, ob die LED jetzt an oder aus ist
bool vorherigerKnopf = HIGH; // Wie der Knopf beim letzten Mal war

void setup() {
  pinMode(KNOPF_PIN, INPUT_PULLUP); // Eingang mit eingebautem Widerstand: los = HIGH, gedrückt = LOW
  pinMode(LED_PIN, OUTPUT);
}

void loop() {
  bool knopf = digitalRead(KNOPF_PIN);

  // Nur in dem Moment reagieren, in dem du den Knopf drückst
  if (vorherigerKnopf == HIGH && knopf == LOW) {
    ledAn = !ledAn; // Umdrehen: an wird aus, aus wird an
    digitalWrite(LED_PIN, ledAn ? HIGH : LOW);
    delay(50); // Kurz warten gegen das "Prellen" des Knopfes
  }

  vorherigerKnopf = knopf;
}
```

Was macht der Code?

- Der Knopf sitzt an Pin 2 mit `INPUT_PULLUP`. Losgelassen ist der Pin HIGH, gedrückt LOW.
- Wir merken uns in `ledAn`, ob die LED an oder aus ist.
- Nur in dem Moment, in dem du den Knopf drückst, drehen wir `ledAn` um.

Probier mal ...

- Lass die LED blinken, statt dass sie an bleibt
- Lass die LED nach 10 Sekunden von selbst wieder ausgehen
- Zähle, wie oft auf den Knopf gedrückt wurde, und zeige es im seriellen Monitor

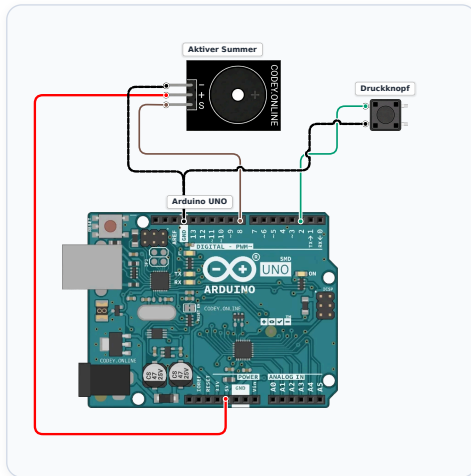
🚀 Baue dieses Projekt mit Codey: Öffne codey.online/de/starter-kits/uno/projects/light-switch und klicke auf BAU DAS. Dann sind der Code und der Schaltplan sofort bereit.

★ Anfänger ⌚ 10 Minuten

Türklingel

Du baust deine eigene Türklingel. Drück auf den Knopf und der Summer sagt "Ding-Dong"!

Schaltplan



Was brauchst du?



1× Arduino UNO



1× Druckknopf



1× Aktiver Summer

USB-Kabel +
Jumperkabel

So baust du es

1. Setze den Druckknopf über die Rille vom breadboard: ein Beinchen zu Pin 2, ein Beinchen auf der anderen Seite zu GND.
2. Schließe den Summer an: S an Pin 8, + an 5V und – an GND.

Der Code

```
// Türklingel
// Drücke den Knopf und der Summer lässt ein fröhliches "Ding-Dong" hören.

const int TASTER_PIN = 2; // Druckknopf an Pin 2 (andere Seite zu GND)
const int SUMMER_PIN = 8; // Aktiver Summer: S an Pin 8

void setup() {
  pinMode(TASTER_PIN, INPUT_PULLUP); // Los = HIGH, gedrückt = LOW
  pinMode(SUMMER_PIN, OUTPUT);
}

// Lässt den Summer eine Weile piepen
void piepe(int wieLang) {
  digitalWrite(SUMMER_PIN, HIGH);
  delay(wieLang);
  digitalWrite(SUMMER_PIN, LOW);
}

void loop() {
  if (digitalRead(TASTER_PIN) == LOW) { // Jemand drückt auf die Klingel!
    piepe(300); // "Ding"
    delay(150);
    piepe(600); // "Dong"
    delay(1000); // Kurz warten, damit die Klingel nicht weiterklingelt
  }
}
```

Was macht der Code?

- Der Knopf sitzt an Pin 2 und der Summer an Pin 8.
- Die Funktion `piepe()` schaltet den Summer kurz ein und dann wieder aus.
- Wenn der Knopf gedrückt ist, hören wir einen kurzen und einen langen Piepton.

Probier mal ...

- Spiele eine fröhliche Melodie mit fünf Pieptönen
- Lass auch eine LED blinken, wenn jemand klingelt
- Zähle, wie oft geklingelt wurde, und zeige es im seriellen Monitor

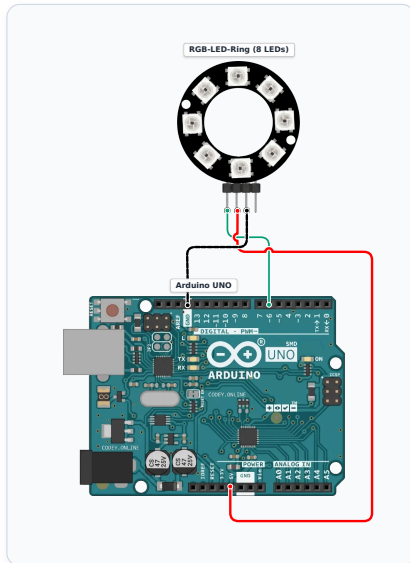
🚀 Baue dieses Projekt mit Codey: Öffne codey.online/de/starter-kits/uno/projects/doorbell und klicke auf BAU DAS. Dann sind der Code und der Schaltplan sofort bereit.

★ Anfänger ⏱ 10 Minuten

Regenbogenring

Mit dem LED-Ring machst du einen drehenden Regenbogen. Jedes Lämpchen bekommt eine andere Farbe. Und alle Farben wandern rundherum.

Schaltplan



Was brauchst du?



1× Arduino UNO



1× RGB-LED-Ring (8 LEDs)



USB-Kabel + Jumperkabel

So baust du es

1. Schließe den LED-Ring an: DI an Pin 6, 5V an 5V und GND an GND.
2. Das ist alles! DO lässt du frei.

Der Code

```
// Regenbogenring
// Der LED-Ring lässt alle Farben des Regenbogens im Kreis drehen.

#include <Adafruit_NeoPixel.h>

const int RING_PIN = 6; // DI des Rings ist an Pin 6
const int ANZAHL_LEDS = 8; // Der Ring hat 8 Lämpchen
const int GESCHWINDIGKEIT = 20; // Je kleiner, desto schneller dreht es sich

Adafruit_NeoPixel ring(ANZAHL_LEDS, RING_PIN, NEO_GRB + NEO_KHZ800);

void setup() {
  ring.begin();
  ring.setBrightness(40); // Nicht zu hell: 0 = aus, 255 = superhell
  ring.show();
}

void loop() {
  // Schiebe den Regenbogen immer ein Stück weiter
  for (long farbe = 0; farbe < 65536; farbe += 512) {
    for (int i = 0; i < ANZAHL_LEDS; i++) {
      // Jedes Lämpchen bekommt eine etwas andere Farbe
      long ledFarbe = farbe + (i * 65536L / ANZAHL_LEDS);
      ring.setPixelColor(i, ring.gamma32(ring.ColorHSV(ledFarbe)));
    }
    ring.show();
    delay(GESCHWINDIGKEIT);
  }
}
```

Was macht der Code?

- Wir benutzen die Bibliothek `Adafruit_NeoPixel`, um den Ring zu steuern.
- `ColorHSV()` macht eine Farbe aus dem Regenbogen. Jedes Lämpchen bekommt eine Farbe ein Stück weiter im Regenbogen.
- Wenn wir die Farbe immer ein Stück weiterschieben, sieht es so aus, als dreht sich der Regenbogen im Kreis.

Probier mal ...

- Lass den Regenbogen doppelt so schnell drehen
- Mache daraus ein Polizeilicht, das abwechselnd rot und blau blinkt
- Lass ein Lämpchen im Kreis laufen, wie ein kleines Licht, das herumrennt

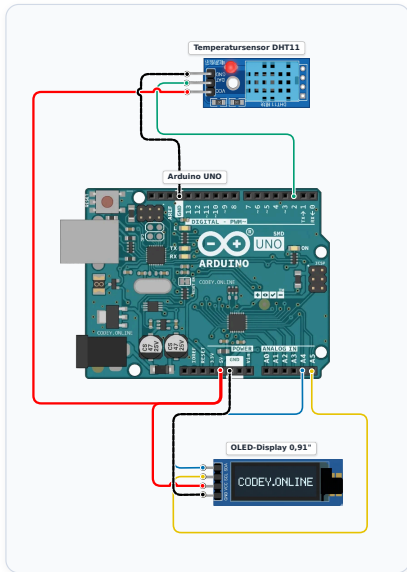
🚀 Baue dieses Projekt mit Codey: Öffne codey.online/de/starter-kits/uno/projects/rainbow-ring und klicke auf BAU DAS. Dann sind der Code und der Schaltplan sofort bereit.

★★ Fortgeschrittener Anfänger ⏱ 20 Minuten

Mini-Wetterstation

Du baust eine Mini-Wetterstation. Der DHT11 misst die Temperatur und die Luftfeuchtigkeit. Das OLED-Display zeigt sie an.

Schaltplan



Was brauchst du?



1x Arduino UNO



1x Temperatursensor DHT11



1x OLED-Display 0,91"



USB-Kabel + Jumperkabel

So baust du es

1. Schließe den DHT11 an: DAT an pin 2, VCC an 5V und GND an GND.
2. Schließe das OLED-Display an: SDA an A4, SCL an A5, VCC an 5V und GND an GND.
3. Tipp: Zwei Teile brauchen 5V und GND. Nutze die + und – Streifen deines breadboard.

Der Code

```
// Mini-Wetterstation
// Der DHT11 misst die Temperatur und die Luftfeuchtigkeit. Das OLED-Display zeigt sie an.

#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <DHT.h>

const int DHT_PIN = 2; // DAT des DHT11 sitzt auf pin 2
DHT dht(DHT_PIN, DHT11);

Adafruit_SSD1306 display(128, 32, &Wire, -1); // OLED mit 128 x 32 Pixeln (SDA = A4, SCL = A5)

void setup() {
  Serial.begin(9600);
  dht.begin();
  if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) { // 0x3C ist die Adresse des Displays
    Serial.println(F("Display nicht gefunden. Prüfe die Drähte von SDA und SCL."));
    while (true); // Stopp: Ohne Display geht es nicht
  }
  display.setTextColor(SSD1306_WHITE);
}

void loop() {
  float temperatur = dht.readTemperature(); // In Grad Celsius
  float feuchte = dht.readHumidity(); // In Prozent

  display.clearDisplay();
  display.setTextSize(1);
  display.setCursor(0, 0);

  if (isnan(temperatur) || isnan(feuchte)) { // Klappt das Messen nicht?
    display.setTextSize(1);
    display.setCursor(0, 0);
    display.print(F("Kein DHT11 gefunden"));
  } else {
    display.print(temperatur, 1);
    display.print((char)247); // Das Grad-Zeichen
    display.println(F("°C"));
    display.print(feuchte, 0);
    display.print(F("% Feu. "));
  }

  Serial.print(F("Temperatur: "));
  Serial.print(temperatur);
  Serial.print(F(" °C Feuchte: "));
  Serial.print(feuchte);
  Serial.println(F(" %"));
}

display.display(); // Erst jetzt kommt alles wirklich auf das Display
delay(2000); // Der DHT11 kann nur alle 2 Sekunden messen
}
```

Was macht der Code?

- Der DHT11 sitzt an pin 2. Mit `readTemperature()` und `readHumidity()` lesen wir die Werte.
- Das OLED-Display spricht über SDA (A4) und SCL (A5). Mit `print()` schreiben wir Text auf das Display.
- Alle 2 Sekunden kommt eine neue Messung. Klappt das Messen nicht? Dann siehst du eine Meldung.

Probier mal ...

- Zeige auch die Temperatur in Fahrenheit
- Lass eine LED leuchten, wenn es wärmer als 25 Grad ist
- Merke dir die höchste Temperatur von heute und zeige sie auch an

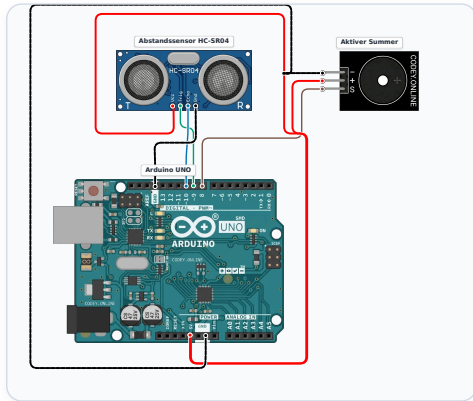
🔧 Baue dieses Projekt mit Codey: Öffne codey.online/de/starter-kits/uno/projects/weather-station und klicke auf BAU DAS. Dann sind der Code und der Schaltplan sofort bereit.

★★ Fortgeschrittener Anfänger ⌚ 20 Minuten

Parksensor

Du baust einen Parksensor, genau wie in einem Auto. Je näher du an den Sensor kommst, desto schneller piept der Summer. Ganz nah dran? Dann hörst du einen langen Ton.

Schaltplan



Was brauchst du?



1× Arduino UNO



1× Abstandssensor HC-SR04



1× Aktiver Summer



USB-Kabel + Jumperkabel

So baust du es

- Schließe den HC-SR04 an: Trig an Pin 9, Echo an Pin 10, Vcc an 5V und Gnd an GND.
- Schließe den Summer an: S an Pin 8, + an 5V und – an GND.
- Nutze die + und – Leisten auf deinem breadboard für 5V und GND.

Der Code

```
// Parksensor
// Je näher du an den Sensor kommst, desto schneller piept der Summer. Wie im Auto!

const int TRIG_PIN = 9;    // Trig des HC-SR04 an Pin 9
const int ECHO_PIN = 10;   // Echo des HC-SR04 an Pin 10
const int SUMMER_PIN = 8;  // S des Summers an Pin 8

void setup() {
  Serial.begin(9600);
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  pinMode(SUMMER_PIN, OUTPUT);
}

// Misst den Abstand in Zentimetern
long missAbstand() {
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);    // Sende einen kurzen Ton (den du nicht hörst)
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);

  long zeit = pulseIn(ECHO_PIN, HIGH, 30000); // Wie lange dauert es, bis das Echo zurückkommt?

  if (zeit == 0) return 999;          // Kein Echo: nichts in der Nähe
  return zeit / 58;                  // Zeit in Zentimeter umrechnen
}

void loop() {
  long abstand = missAbstand();
  Serial.print(F("Abstand: "));
  Serial.print(abstand);
  Serial.println(F(" cm"));

  if (abstand < 5) {
    digitalWrite(SUMMER_PIN, HIGH); // Sehr nah: ein langer Ton
    delay(100);
  } else if (abstand < 50) {
    digitalWrite(SUMMER_PIN, HIGH); // Kurzer Piepton
    delay(50);
    digitalWrite(SUMMER_PIN, LOW);
    delay(abstand * 10);             // Näher dran = kürzere Pause = schnelleres Piepen
  } else {
    digitalWrite(SUMMER_PIN, LOW);  // Weit weg: still
    delay(100);
  }
}
```

Was macht der Code?

- `missAbstand()` sendet einen Ton und misst, wie lange es dauert, bis das Echo zurückkommt. Das rechnen wir in Zentimeter um.
- Weiter als 50 cm: Der Summer ist still.
- Näher dran: Die Pause zwischen den Pieptönen wird kürzer. Unter 5 cm piept er die ganze Zeit.

Probier mal ...

- Lass den Parksensor schon ab 1 Meter piepen
- Füge eine rote LED hinzu, die leuchtet, wenn es zu nah ist
- Zeige den Abstand im seriellen Monitor, mit einem kleinen Balken aus Sternchen

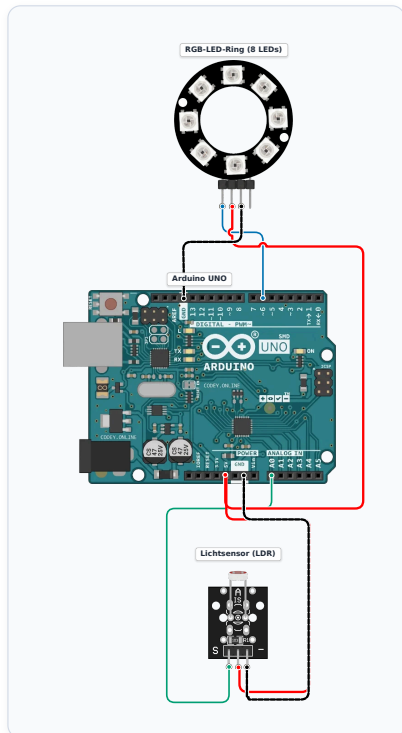
🚀 Baue dieses Projekt mit Codey: Öffne codey.online/de/starter-kits/uno/projects/parking-sensor und klicke auf BAU DAS. Dann sind der Code und der Schaltplan sofort bereit.

★★ Fortgeschrittener Anfänger ⌚ 15 Minuten

Nachtlämpchen

Du baust ein schlaues Nachtlämpchen. Wird es dunkel, dann leuchtet der LED-Ring sanft. Wird es hell, dann geht er wieder aus.

Schaltplan



Was brauchst du?



1× Arduino UNO



1× Lichtsensor (LDR)



1× RGB-LED-Ring (8 LEDs)



USB-Kabel + Jumperkabel

So baust du es

1. Schließe den Lichtsensor an: S an A0, + an 5V und – an GND.
2. Schließe den LED-Ring an: DI an Pin 6, 5V an 5V und GND an GND.
3. Öffne den seriellen Monitor und schau, welche Zahl du bei Licht und bei Dunkelheit siehst. Passe `DUNKEL_GRENZE` daran an.

Der Code

```
// Nachtlämpchen
// Wird es dunkel? Dann leuchtet der LED-Ring sanft. Wird es hell? Dann geht er
aus.

#include <Adafruit_NeoPixel.h>

const int LICHT_PIN = A0; // S vom Lichtsensor auf A0
const int RING_PIN = 6; // DI vom LED-Ring auf Pin 6
const int ANZAHL_LEDS = 8;

// Bei diesem Wert finden wir es "dunkel". Schau im seriellen Monitor, welche
Zahl du siehst
// bei Licht und bei Dunkelheit, und setze hier eine Zahl dazwischen ein.
const int DUNKEL_GRENZE = 500;

Adafruit_NeoPixel ring(ANZAHL_LEDS, RING_PIN, NEO_GRB + NEO_KHZ800);

void setup() {
  Serial.begin(9600);
  ring.begin();
  ring.setBrightness(60);
  ring.show();
}

void loop() {
  int licht = analogRead(LICHT_PIN); // Eine Zahl von 0 bis 1023
  Serial.print(F("Lichtsensor: "));
  Serial.println(licht);

  // Bei diesem Sensor wird die Zahl höher, wenn es dunkler wird.
  // Ist es bei dir genau andersherum? Dann ändere > in <.
  bool dunkel = licht > DUNKEL_GRENZE;

  for (int i = 0; i < ANZAHL_LEDS; i++) {
    if (dunkel) {
      ring.setPixelColor(i, ring.Color(255, 140, 40)); // Warmes gelbes Licht
    } else {
      ring.setPixelColor(i, 0); // Aus
    }
  }
  ring.show();
  delay(200);
}
```

Was macht der Code?

- Der Lichtsensor sitzt auf A0. `analogRead()` gibt eine Zahl von 0 bis 1023.
- Ist die Zahl höher als `DUNKEL_GRENZE`? Dann finden wir es dunkel.
- Im Dunkeln geben wir allen Lämpchen eine warme gelbe Farbe. Sonst gehen sie aus.

Probier mal ...

- Lass das Lämpchen langsam heller werden, je dunkler es wird
- Lass die Farbe langsam wechseln, wie eine beruhigende Stimmungslampe
- Lass das Lämpchen nach 1 Minute von selbst wieder ausgehen

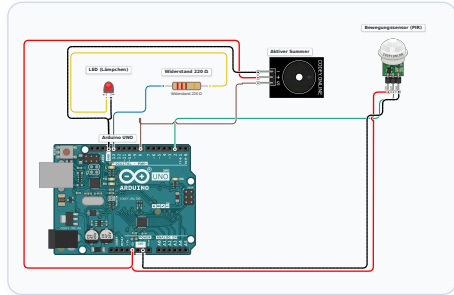
🔧 Baue dieses Projekt mit Codey: Öffne codey.online/de/starter-kits/uno/projects/night-light und klicke auf BAU DAS. Dann sind der Code und der Schaltplan sofort bereit.

★★ Fortgeschrittener Anfänger ⌚ 20 Minuten

Bewegungsalarm

Du baust eine Alarmanlage. Sieht der Bewegungssensor jemanden? Dann geht der Summer los und die rote LED blinkt.

Schaltplan



Was brauchst du?



1× Arduino UNO



1× Bewegungssensor (PIR)



1× Aktiver Summer



1× LED (Lämpchen)



1× Widerstand 220 Ω



USB-Kabel + Jumperkabel

So baust du es

1. Schließe den PIR-Sensor an: OUT an Pin 2, VCC an 5V und GND an GND.
2. Schließe den Summer an: S an Pin 8, + an 5V und – an GND.
3. Setze die rote LED mit einem Widerstand von 220 Ω an Pin 13, das kurze Beinchen an GND.

Der Code

```
// Bewegungsalarm
// Läuft jemand vorbei? Dann blinkt die rote LED und der Summer geht los.

const int PIR_PIN = 2;      // OUT des Bewegungssensors an Pin 2
const int SUMMER_PIN = 8;    // S des Summers an Pin 8
const int LED_PIN = 13;     // Rote LED an Pin 13 (über den Widerstand)

void setup() {
  Serial.begin(9600);
  pinMode(PIR_PIN, INPUT);
  pinMode(SUMMER_PIN, OUTPUT);
  pinMode(LED_PIN, OUTPUT);
  Serial.println(F("Sensor wärmt auf... kurz stillstehen!"));
  delay(10000);             // Der Sensor braucht kurz, um sich an den Raum zu gewöhnen
  Serial.println(F("Alarm ist an."));
}

void loop() {
  if (digitalRead(PIR_PIN) == HIGH) { // Bewegung gesehen!
    Serial.println(F("Bewegung! ALARM!"));
    for (int i = 0; i < 5; i++) { // 5-mal piepen und blinken
      digitalWrite(LED_PIN, HIGH);
      digitalWrite(SUMMER_PIN, HIGH);
      delay(150);
      digitalWrite(LED_PIN, LOW);
      digitalWrite(SUMMER_PIN, LOW);
      delay(150);
    }
  }
  delay(100);
}
```

Was macht der Code?

- Der PIR-Sensor ist an Pin 2. Sieht er Bewegung, dann ist der Pin HIGH.
- Beim Start warten wir 10 Sekunden. So kann sich der Sensor an den Raum gewöhnen.
- Bei Bewegung piept und blinkt es 5-mal.

Probier mal ...

- Füge einen Knopf hinzu, um den Alarm an- und auszuschalten
- Zähle, wie oft Bewegung da war, und zeige es im seriellen Monitor
- Lass den Alarm beim zweiten Mal länger und lauter klingen

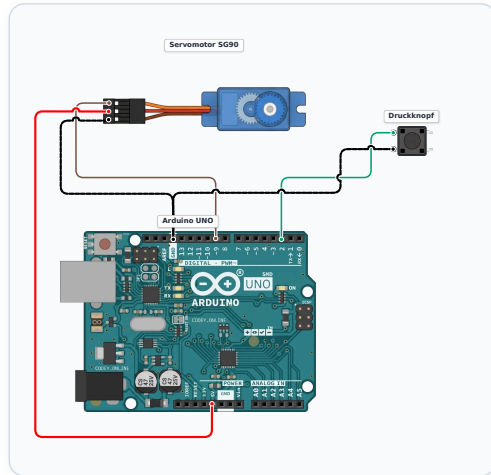
🚀 Baue dieses Projekt mit Codey: Öffne codey.online/de/starter-kits/uno/projects/motion-alarm und klicke auf BAU DAS. Dann sind der Code und der Schaltplan sofort bereit.

★★ Fortgeschrittener Anfänger ⌚ 20 Minuten

Schranke

Du baust eine Schranke, wie in einem Parkhaus. Drück auf den Knopf: Der Servo dreht die Schranke nach oben, wartet kurz und geht wieder zu.

Schaltplan



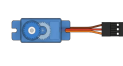
Was brauchst du?



1x Arduino UNO



1x Druckknopf



1x Servomotor SG90



USB-Kabel + Jumperkabel

So baust du es

1. Setze den Druckknopf über die Rille deines Breadboards: ein Beinchen an Pin 2, ein Beinchen auf der anderen Seite an GND.
2. Schließe den Servo an: orange an Pin 9, rot an 5V und braun an GND.
3. Klebe einen Strohhalm oder Eisstiel an den Arm vom Servo: Das ist deine Schranke!

Der Code

```
// Schranke
// Drück auf den Knopf: die Schranke (Servo) geht nach oben,
// wartet kurz und geht wieder zu.

#include <Servo.h>

const int TASTER_PIN = 2;    // Druckknopf an Pin 2 (andere
                             // Seite zu GND)
const int SERVO_PIN = 9;    // Orange/gelbes Kabel vom Servo an
                             // Pin 9

const int ZU = 0;           // Winkel vom Servo, wenn die Schranke
                             // zu ist
const int OFFEN = 90;       // Winkel vom Servo, wenn die
                             // Schranke offen ist

Servo schranke;

void setup() {
  pinMode(TASTER_PIN, INPUT_PULLUP);

  schranke.attach(SERVO_PIN);
  schranke.write(ZU);    // Starte mit geschlossener Schranke
}

void loop() {
  if (digitalRead(TASTER_PIN) == LOW) {    // Knopf gedrückt
    // Langsam öffnen, Grad für Grad
    for (int winkel = ZU; winkel <= OFFEN; winkel++) {
      schranke.write(winkel);
      delay(15);
    }
    delay(3000);    // 3 Sekunden offen: das
    // Auto fährt durch
    // Langsam wieder zu
    for (int winkel = OFFEN; winkel >= ZU; winkel--) {
      schranke.write(winkel);
      delay(15);
    }
  }
}
```

Was macht der Code?

- Wir benutzen die Bibliothek `Servo`. Mit `write()` sagst du, zu welchem Winkel der Servo drehen soll.
- Nach einem Druck auf den Knopf drehen wir Grad für Grad von 0 nach 90. So geht es ganz ruhig.
- Nach 3 Sekunden drehen wir wieder zurück nach 0: Die Schranke ist zu.

Probier mal ...

- Lass die Schranke schneller auf und zu gehen
- Lass einen Summer piepen, solange sich die Schranke bewegt
- Lass eine rote LED leuchten, wenn die Schranke zu ist, und eine grüne, wenn sie offen ist

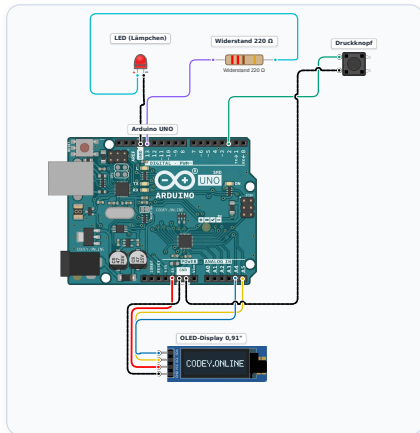
🔧 Baue dieses Projekt mit Codey: Öffne codey.online/de/starter-kits/uno/projects/barrier und klicke auf BAU DAS. Dann sind der Code und der Schaltplan sofort bereit.

★★ Fortgeschrittener Anfänger ⌚ 25 Minuten

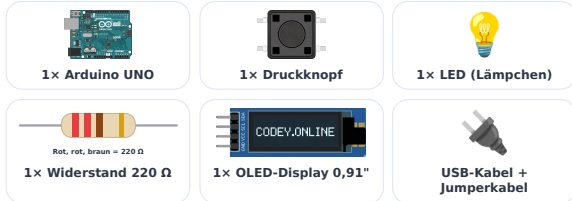
Reaktionsspiel

Wie schnell bist du? Warte, bis die LED angeht. Drücke dann so schnell wie möglich auf den Knopf. Das Display zeigt deine Zeit in Millisekunden. Fordere deine Freunde heraus!

Schaltplan



Was brauchst du?



So baust du es

1. Setze den Druckknopf über die Lücke des Breadboards: ein Beinchen zu Pin 2, ein Beinchen auf der anderen Seite zu GND.
2. Setze die LED mit einem Widerstand von 220 Ω an Pin 13, das kurze Beinchen zu GND.
3. Schließe das OLED-Display an: SDA zu A4, SCL zu A5, VCC zu 5V und GND zu GND.

Der Code

```
// Reaktionsspiel
// Warte, bis die LED angeht, und druecke dann so schnell wie moeglich auf
// den Taster.
// Das Display zeigt, wie schnell du warst!

#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

const int TASTER_PIN = 2; // Drucktaster an Pin 2 (andere Seite nach GND)
const int LED_PIN = 13; // LED an Pin 13 (ueber den Widerstand)

Adafruit_SSD1306 anzeige(128, 32, &Wire, -1); // OLED: SDA = A4, SCL = A5

unsigned long besteZeit = 0; // Deine schnellste Zeit bis jetzt

// Zeigt einen Text gross auf dem Display
void zeige(const __FlashStringHelper* oben, unsigned long zahl) {
  anzeige.clearDisplay();
  anzeige.setTextSize(1);
  anzeige.setCursor(0, 0);
  anzeige.print(oben);
  if (zahl > 0) {
    anzeige.setTextSize(2);
    anzeige.setCursor(0, 14);
    anzeige.print(zahl);
    anzeige.print(F(" ms"));
  }
  anzeige.display();
}

void setup() {
  pinMode(TASTER_PIN, INPUT_PULLUP);
  pinMode(LED_PIN, OUTPUT);
  randomSeed(analogRead(A0)); // Sorgt fuer echt zufaellige Wartezeiten

  anzeige.begin(SSD1306_SWITCHCAPVCC, 0x3C);
  anzeige.setTextColor(SSD1306_WHITE);
}

void loop() {
  zeige(F("Taste druecken..."), 0);
  while (digitalRead(TASTER_PIN) == HIGH); // Warten, bis du den Taster
  drueckst
  while (digitalRead(TASTER_PIN) == LOW); // ...und wieder loslaesst
  zeige(F("Bereit? LED!"), 0);

  delay(random(2000, 5000)); // Eine geheime Wartezeit zwischen
  2 und 5 Sekunden

  if (digitalRead(TASTER_PIN) == LOW) { // Zu frueh gedrueckt?
    Schummel!
    zeige(F("Zu frueh! Nochmal."), 0);
    delay(2000);
    return;
  }

  digitalWrite(LED_PIN, HIGH); // JETZT!
  unsigned long start = millis();
  while (digitalRead(TASTER_PIN) == HIGH); // Warten auf deinen Druck
  unsigned long zeit = millis() - start;
  digitalWrite(LED_PIN, LOW);

  if (besteZeit == 0 || zeit < besteZeit) {
    besteZeit = zeit;
    zeige(F("NEUER REKORD!"), zeit);
  } else {
    zeige(F("Zeit:"), zeit);
  }
  delay(3000);
}
```

Was macht der Code?

- Nach einem Druck auf den Knopf warten wir eine geheime Zeit zwischen 2 und 5 Sekunden (`random()`).
- Dann geht die LED an und wir starten eine Stoppuhr mit `millis()`.
- Wenn du drückst, rechnen wir aus, wie viele Millisekunden es gedauert hat. Deine beste Zeit merken wir uns.

Probiere mal ...

- Mach daraus ein Spiel für zwei Spieler, jeder mit einem eigenen Knopf
- Lass einen Summer piepen, wenn es einen neuen Rekord gibt
- Spiele 5 Runden und zeige danach deine Durchschnittszeit

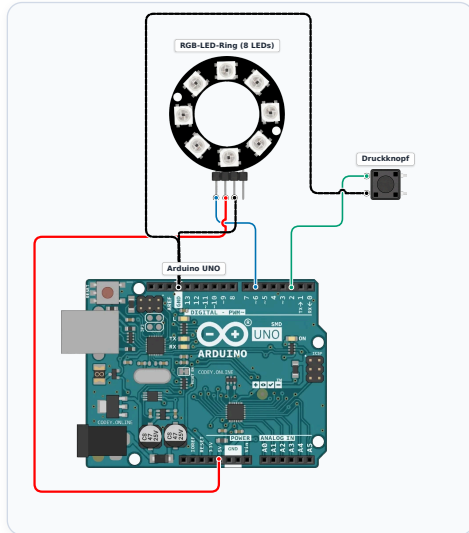
🔧 Baue dieses Projekt mit Codey: Öffne codey.online/de/starter-kits/uno/projects/reaction-game und klicke auf BAU DAS. Dann sind der Code und der Schaltplan sofort bereit.

★★ Fortgeschrittener Anfänger ⏱ 15 Minuten

Elektronischer Würfel

Du baust einen Würfel aus Licht. Drück auf den Knopf: Die Lämpchen rollen kurz, und dann siehst du, was du gewürfelt hast. 1 bis 6 Lämpchen!

Schaltplan



Was brauchst du?



1× Arduino UNO



1× Druckknopf



1× RGB-LED-Ring (8 LEDs)



USB-Kabel + Jumperkabel

So baust du es

1. Stecke den Taster über die Rille des breadboard: ein Beinchen zu Pin 2, ein Beinchen auf der anderen Seite zu GND.
2. Schließe den LED-ring an: DI an Pin 6, 5V an 5V und GND an GND.

Der Code

```
// Elektronischer Würfel
// Drück auf den Knopf: Der LED-ring rollt kurz und lässt dann 1 bis 6 Lämpchen leuchten.

#include <Adafruit_NeoPixel.h>

const int TASTER_PIN = 2; // Taster an Pin 2 (andere Seite zu GND)
const int RING_PIN = 6; // DI des LED-rings an Pin 6
const int ANZAHL_LEDS = 8;

Adafruit_NeoPixel ring(ANZAHL_LEDS, RING_PIN, NEO_GRB + NEO_KHZ800);

// Lässt einige Lämpchen in einer Farbe leuchten
void zeigeZahl(int zahl, uint32_t farbe) {
  ring.clear();
  for (int i = 0; i < zahl; i++) {
    ring.setPixelColor(i, farbe);
  }
  ring.show();
}

void setup() {
  pinMode(TASTER_PIN, INPUT_PULLUP);
  randomSeed(analogRead(A0)); // So ist der Würfel wirklich zufällig
  ring.begin();
  ring.setBrightness(50);
  ring.show();
}

void loop() {
  if (digitalRead(TASTER_PIN) == LOW) {
    // Kurz "rollen": schnell wechselnde Zahlen, immer etwas langsamer
    for (int i = 0; i < 15; i++) {
      zeigeZahl(random(1, 7), ring.Color(0, 80, 255));
      delay(40 + i * 10);
    }
    int wurf = random(1, 7); // Eine Zahl von 1 bis 6
    zeigeZahl(wurf, ring.Color(0, 255, 60)); // Die echte Zahl in Grün
    delay(500);
  }
}
```

Was macht der Code?

- `random(1, 7)` wählt eine zufällige Zahl von 1 bis 6.
- Zuerst zeigen wir schnell wechselnde Zahlen in Blau. So sieht es aus, als würde der Würfel rollen.
- Danach zeigt `zeigeZahl()` die echte Zahl in Grün.

Probier mal ...

- Gib jeder Zahl eine eigene Farbe
- Lass einen Summer kurz piepen, wenn der Würfel stillsteht
- Mach daraus einen Würfel von 1 bis 8

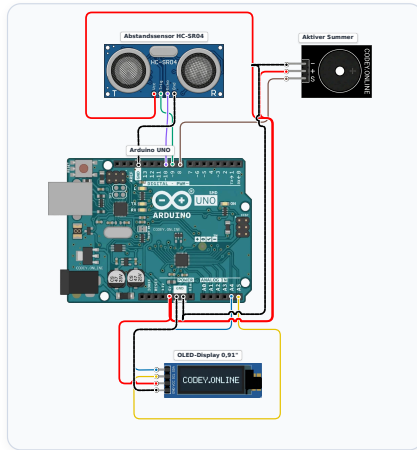
🚀 Baue dieses Projekt mit Codey: Öffne codey.online/de/starter-kits/uno/projects/dice und klicke auf BAU DAS. Dann sind der Code und der Schaltplan sofort bereit.

☆☆☆ Herausforderung ⌚ 30 Minuten

Parksensor deluxe

Die Deluxe-Version vom Parksensor! Er piept schneller, wenn du näher kommst. Und das OLED zeigt den Abstand in Zentimetern. Mit einem kleinen Balken. Der wird länger, wenn du näher kommst.

Schaltplan



Was brauchst du?



So baust du es

1. Schließe den HC-SR04 an: Trig an Pin 9, Echo an Pin 10, VCC an 5V und GND an GND.
2. Schließe den Summer an: S an Pin 8, + an 5V und – an GND.
3. Schließe das OLED an: SDA an A4, SCL an A5, VCC an 5V und GND an GND.
4. Drei Teile brauchen 5V und GND. Benutze die + und – Leisten von deinem breadboard.

Der Code

```
// Parksensor deluxe
// Wie der normale Parksensor. Aber jetzt siehst du den Abstand auch auf dem OLED.

#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

const int TRIG_PIN = 9; // Trig vom HC-SR04 an Pin 9
const int ECHO_PIN = 10; // Echo vom HC-SR04 an Pin 10
const int SUMMER_PIN = 8; // S vom Summer an Pin 8

Adafruit_SSD1306 bildschirm(128, 32, &Wire, -1); // OLED: SDA = A4, SCL = A5

// Misst den Abstand in Zentimetern
long missAbstand() {
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);
  long zeit = pulseIn(ECHO_PIN, HIGH, 30000);
  if (zeit == 0) return 999; // Kein Echo: nichts in der Naeh
  return zeit / 58;
}

// Zeigt den Abstand gross auf dem Bildschirm, mit einem Balken darunter
void zeigeAbstand(long abstand) {
  bildschirm.clearDisplay();
  bildschirm.setTextSize(2);
  bildschirm.setCursor(0, 0);
  if (abstand >= 999) {
    bildschirm.print(F("..."));
  } else {
    bildschirm.print(abstand);
    bildschirm.print(F(" cm"));
  }
}

// Balken: je naeher, desto laenger
int laenge = map(constrain(abstand, 0, 100), 0, 100, 128, 0);
bildschirm.fillRect(0, 24, laenge, 8, SSD1306_WHITE);
bildschirm.display();

void setup() {
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  pinMode(SUMMER_PIN, OUTPUT);
  bildschirm.begin(SSD1306_SWITCHCAPVCC, 0x3C);
  bildschirm.setTextColor(SSD1306_WHITE);
}

void loop() {
  long abstand = missAbstand();
  zeigeAbstand(abstand);

  if (abstand < 5) {
    digitalWrite(SUMMER_PIN, HIGH); // Ganz nah: ein langer Ton
    delay(100);
  } else if (abstand < 50) {
    digitalWrite(SUMMER_PIN, HIGH); // Kurzer Piep
    delay(50);
    digitalWrite(SUMMER_PIN, LOW);
    delay(abstand * 10); // Naeh = schneller piepen
  } else {
    digitalWrite(SUMMER_PIN, LOW);
    delay(100);
  }
}
```

Was macht der Code?

- `missAbstand()` misst den Abstand in Zentimetern mit dem HC-SR04.
- `zeigeAbstand()` schreibt die Zahl groß auf den Bildschirm und zeichnet einen kleinen Balken mit `fillRect()`.
- Der Summer piept schneller, wenn der Abstand kleiner wird.

Probier mal ...

- Zeige "STOP!" auf dem Bildschirm, wenn du näher als 5 cm bist
- Zeige den Abstand in Metern, wenn etwas weiter als 1 Meter weg ist
- Füge den LED-Ring hinzu. Er soll von grün nach rot wechseln

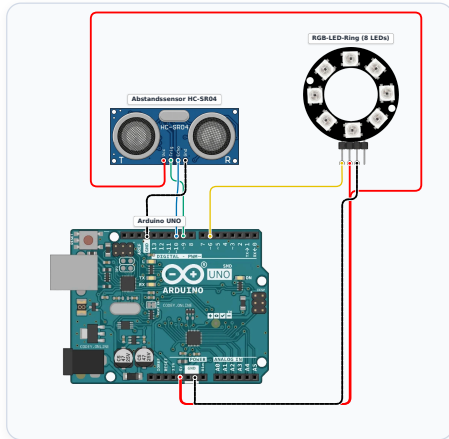
🔧 Baue dieses Projekt mit Codey: Öffne codey.online/de/starter-kits/uno/projects/parking-sensor-deluxe und klicke auf BAU DAS. Dann sind der Code und der Schaltplan sofort bereit.

★★★ Herausforderung ⌚ 25 Minuten

Abstandsmesser mit Lichtbalken

Du baust einen Lichtbalken. Er zeigt, wie nah etwas ist. Weit weg: keine Lämpchen. Je näher es kommt, desto mehr Lämpchen leuchten. Erst grün, dann orange und ganz nah rot.

Schaltplan



Was brauchst du?



1x Arduino UNO



1x Abstandssensor HC-SR04



1x RGB-LED-Ring (8 LEDs)



USB-Kabel + Jumperkabel

So baust du es

1. Schließe den HC-SR04 an: Trig an Pin 9, Echo an Pin 10, Vcc an 5V und Gnd an GND.
2. Schließe den LED-Ring an: DI an Pin 6, 5V an 5V und GND an GND.

Der Code

```
// Abstandsmesser mit Lichtbalken
// Je näher du kommst, desto mehr Lämpchen leuchten: grün, dann orange, dann rot.

#include <Adafruit_NeoPixel.h>

const int TRIG_PIN = 9;    // Trig des HC-SR04 an Pin 9
const int ECHO_PIN = 10;   // Echo des HC-SR04 an Pin 10
const int RING_PIN = 6;    // DI des LED-Rings an Pin 6
const int ANZAHL_LEDS = 8;
const int MAX_ABSTAND = 80; // Weiter als das (in cm): alle Lämpchen aus

Adafruit_NeoPixel ring(ANZAHL_LEDS, RING_PIN, NEO_GRB + NEO_KHZ800);

long missAbstand() {
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);
  long zeit = pulseIn(ECHO_PIN, HIGH, 30000);
  if (zeit == 0) return 999;
  return zeit / 58;
}

void setup() {
  Serial.begin(9600);
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  ring.begin();
  ring.setBrightness(40);
  ring.show();
}

void loop() {
  long abstand = missAbstand();
  Serial.print(F("Abstand: "));
  Serial.print(abstand);
  Serial.println(F(" cm"));

  // Wie viele Lämpchen sollen leuchten? Nah = 8, weit weg = 0
  int anzahl = map(constrain(abstand, 0, MAX_ABSTAND), 0, MAX_ABSTAND, ANZAHL_LEDS, 0);

  for (int i = 0; i < ANZAHL_LEDS; i++) {
    if (i < anzahl) {
      if (i < 3) ring.setPixelColor(i, ring.Color(0, 255, 0));
      // Grün
      else if (i < 6) ring.setPixelColor(i, ring.Color(255, 120, 0));
      // Orange
      else ring.setPixelColor(i, ring.Color(255, 0, 0));
      // Rot
    } else {
      ring.setPixelColor(i, 0);
      // Aus
    }
  }
  ring.show();
  delay(100);
}
```

Was macht der Code?

- `missAbstand()` misst den Abstand mit dem HC-SR04.
- `map()` rechnet den Abstand in eine Anzahl Lämpchen um: 0 bis 8.
- Die ersten 3 Lämpchen sind grün, die nächsten 3 orange und die letzten 2 rot.

Probier mal ...

- Lass alle Lämpchen rot blinken, wenn du näher als 10 cm bist
- Füge den Summer hinzu, der bei Rot piept
- Mach den Messer empfindlicher: alle Lämpchen an bei 40 cm statt bei 80 cm

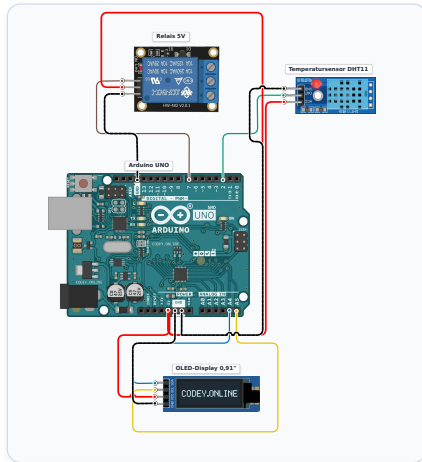
🚀 Baue dieses Projekt mit Codey: Öffne codey.online/de/starter-kits/uno/projects/distance-light-bar und klicke auf BAU DAS. Dann sind der Code und der Schaltplan sofort bereit.

★★★★ Herausforderung ⌚ 30 Minuten

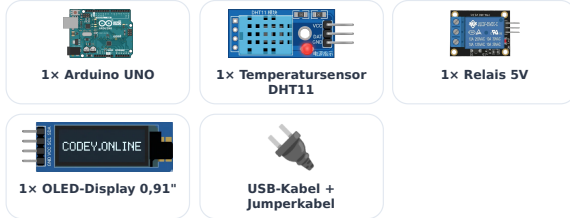
Smarter Ventilator

Du baust einen Thermostat. Wird es wärmer als 25 Grad, dann klickt das Relais an. So kann dein Arduino zum Beispiel einen kleinen Ventilator mit Batterie einschalten. Auf dem Bildschirm siehst du die Temperatur und ob der Ventilator an ist.

Schaltplan



Was brauchst du?



So baust du es

1. Schließe den DHT11 an: DAT an pin 2, VCC an 5V und GND an GND.
2. Schließe das Relais an: S an pin 7, + an 5V und – an GND.
3. Schließe das OLED-Display an: SDA an A4, SCL an A5, VCC an 5V und GND an GND.
4. Willst du wirklich etwas schalten? Benutze dann nur eine Batterie oder niedrige Spannung an COM und NO, niemals die Steckdose.

Der Code

```
// Smarter Ventilator (Thermostat)
// Wird es zu warm? Dann schaltet das Relais ein, zum Beispiel für einen Ventilator.
// Auf dem Bildschirm siehst du die Temperatur und ob der Ventilator an ist.

#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <DHT.h>

const int DHT_PIN = 2; // DAT des DHT11 an pin 2
const int RELAIS_PIN = 7; // S des Relais an pin 7
const float AN_UEBER = 25.0; // Über dieser Temperatur geht der Ventilator an
const float AUS_UNTER = 24.0; // Unter dieser Temperatur geht er wieder aus

DHT dht(DHT_PIN, DHT11);
Adafruit_SSD1306 bildschirm(128, 32, &Wire, -1); // OLED: SDA = A4, SCL = A5

bool ventilatorAn = false;

void setup() {
  Serial.begin(9600);
  pinMode(RELAIS_PIN, OUTPUT);
  digitalWrite(RELAIS_PIN, LOW); // Starte mit dem Relais aus
  dht.begin();
  bildschirm.begin(SSD1306_SWITCHCAPVCC, 0x3C);
  bildschirm.setTextColor(SSD1306_WHITE);
}

void loop() {
  float temperatur = dht.readTemperature();

  if (!isnan(temperatur)) {
    // Zwei Grenzen, damit das Relais nicht dauernd an und aus klickt
    if (temperatur > AN_UEBER) ventilatorAn = true;
    if (temperatur < AUS_UNTER) ventilatorAn = false;
    digitalWrite(RELAIS_PIN, ventilatorAn ? HIGH : LOW);
  }

  bildschirm.clearDisplay();
  bildschirm.setTextSize(2);
  bildschirm.setCursor(0, 0);
  if (isnan(temperatur)) {
    bildschirm.print(F("DHT11 ?"));
  } else {
    bildschirm.print(temperatur, 1);
    bildschirm.print((char)247);
    bildschirm.print(F("C"));
  }

  bildschirm.setTextSize(1);
  bildschirm.setCursor(0, 22);
  bildschirm.print(ventilatorAn ? F("Ventilator: AN") : F("Ventilator: aus"));
  bildschirm.display();

  Serial.print(F("Temperatur: "));
  Serial.print(temperatur);
  Serial.println(ventilatorAn ? F(" C -> AN") : F(" C -> aus"));

  delay(2000);
}
```

Was macht der Code?

- Der DHT11 misst die Temperatur. Über 25 Grad schalten wir das Relais ein.
- Erst unter 24 Grad geht das Relais wieder aus. Mit diesen zwei Grenzen klickt es nicht dauernd an und aus.
- Das OLED-Display zeigt die Temperatur und "Ventilator: AN" oder "aus".

Probier mal ...

- Füge zwei Tasten hinzu, um die Grenz-Temperatur höher und niedriger zu stellen
- Lass eine LED leuchten, solange der Ventilator an ist
- Lass den Ventilator auch angehen, wenn die Luftfeuchtigkeit über 70% kommt

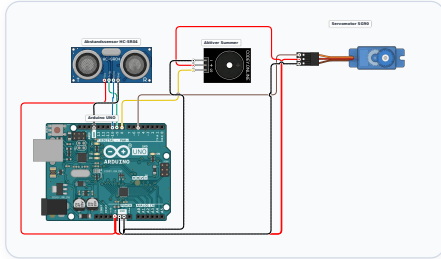
🔧 Baue dieses Projekt mit Codey: Öffne codey.online/de/starter-kits/uno/projects/thermostat und klicke auf BAU DAS. Dann sind der Code und der Schaltplan sofort bereit.

★★★ Herausforderung ⌚ 30 Minuten

Automatische Schranke

Die schlaue Version der Schranke! Der Abstandssensor sieht, dass ein Auto kommt. Dann piept der Summer und die Schranke öffnet sich von selbst. Ist das Auto weg? Dann geht sie wieder zu.

Schaltplan



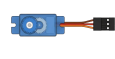
Was brauchst du?



1x Arduino UNO



1x Abstandssensor HC-SR04



1x Servomotor SG90



1x Aktiver Summer



USB-Kabel + Jumperkabel

So baust du es

1. Schließe den HC-SR04 an: Trig an Pin 9, Echo an Pin 10, Vcc an 5V und Gnd an GND.
2. Schließe den Servo an: Orange an Pin 5, Rot an 5V und Braun an GND.
3. Schließe den Summer an: S an Pin 8, + an 5V und – an GND.
4. Klebe einen Strohhalm oder Eisstiel als Schranke auf den Servo und stelle den Sensor davor.

Der Code

```
// Automatische Schranke
// Kommt ein Auto (etwas näher als 15 cm)? Dann piept der Summer und die
// Schranke geht auf.

#include <Servo.h>

const int TRIG_PIN = 9;      // Trig des HC-SR04 an Pin 9
const int ECHO_PIN = 10;     // Echo des HC-SR04 an Pin 10
const int SERVO_PIN = 5;     // Orange/gelbe Leitung des Servos an Pin 5
const int SUMMER_PIN = 8;    // S des Summers an Pin 8
const int NAH_DRAN = 15;     // Ab diesem Abstand (cm) geht die Schranke
// auf

const int ZU = 0;            // Winkel, wenn die Schranke zu ist
const int OFFEN = 90;        // Winkel, wenn die Schranke offen ist

Servo schranke;

long missAbstand() {
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);
  long zeit = pulseIn(ECHO_PIN, HIGH, 30000);
  if (zeit == 0) return 999;
  return zeit / 58;
}

void setup() {
  Serial.begin(9600);
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  pinMode(SUMMER_PIN, OUTPUT);
  schranke.attach(SERVO_PIN);
  schranke.write(ZU);
}

void loop() {
  long abstand = missAbstand();

  if (abstand < NAH_DRAN) {
    Serial.println(F("Auto gesehen! Schranke auf."));
    digitalWrite(SUMMER_PIN, HIGH); // Kurzer Piepton
    delay(100);
    digitalWrite(SUMMER_PIN, LOW);

    schranke.write(OFFEN);
    delay(1000);
    // Offen bleiben, solange das Auto noch da ist
    while (missAbstand() < NAH_DRAN) {
      delay(200);
    }
    delay(2000); // Noch kurz warten, bis das Auto
    // wirklich weg ist
    schranke.write(ZU);
    Serial.println(F("Schranke zu."));
  }
  delay(100);
}
```

Was macht der Code?

- Der HC-SR04 misst immer wieder den Abstand. Näher als 15 cm? Dann steht dort ein Auto.
- Der Summer piept kurz und der Servo dreht auf 90 Grad: offen.
- Solange das Auto dort steht, bleibt die Schranke offen. Danach warten wir 2 Sekunden und sie geht zu.

Probier mal ...

- Zähle, wie viele Autos vorbeigekommen sind, und zeige es im seriellen Monitor
- Füge eine rote und eine grüne LED als Ampel bei der Schranke hinzu
- Lass die Schranke nur aufgehen, wenn auch ein Knopf gedrückt wird

🚀 Baue dieses Projekt mit Codey: Öffne codey.online/de/starter-kits/uno/projects/automatic-barrier und klicke auf BAU DAS. Dann sind der Code und der Schaltplan sofort bereit.