Station 0 – Einstieg

Herzlich Willkommen zum Online-Modul „Smarter Electronics“!

Ihr wollt heute also eine Einparkhilfe selbst konstruieren, einen Rollladen bedienen oder Temperaturen über Farben anzeigen?
Das ist super! Bevor ihr euch aber auf diese Projekte stürzt, ist es wichtig, dass ihr erstmal euer Material kennenlernt. Hier bekommt ihr einen Überblick über den Arduino-Mikrocontroller, Tinkercad, die Bauelemente und die Programmierumgebung!

Während des gesamten Moduls geben euch die Arbeitsblätter Hinweise zur Umsetzung. Achtet dabei einfach auf die folgenden Symbole, die …

- euer Arbeiten strukturieren und Teilziele aufzeigen,
- euch Hilfen geben, Wichtiges, Schwieriges, etc. kennzeichnen und
- die Arbeitsaufträge und Aktionen beinhalten.

Ziele des Einstiegs

Wenn ihr diese Station geschafft habt, wisst ihr…

- wie man Bauteile auswählt und anschließt,
- wie man eine LED zum Leuchten bringt,
- wie man eine LED blinken lässt und
- wie man einen Taster ein- und ausschaltet.

Abb. 1: InfoSphere-Kit

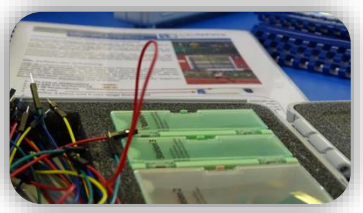
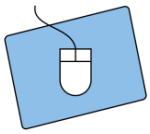


Abb. 2: Inhalt des InfoSphere-Kits



Damit wir anfangen können, müsst ihr zunächst folgende Schritte befolgen:

1. Klickt zunächst auf diesen Link: Tinkercad/arduino
2. Loggt euch nun ein. Wenn ihr dazu Hilfe braucht, fragt eine betreuende Person.
3. Klickt nun auf Neuen Schaltkreis erstellen

Der Arduino

Als Erstes schaut ihr euch den Arduino an. Diesen findet ihr rechts bei den Komponenten unter Arduino Uno R3. Zieht ihn auf die Oberfläche links, und gebt ihm einen sinnvollen Namen (z. B. Arduino1 oder Hauptarduino).

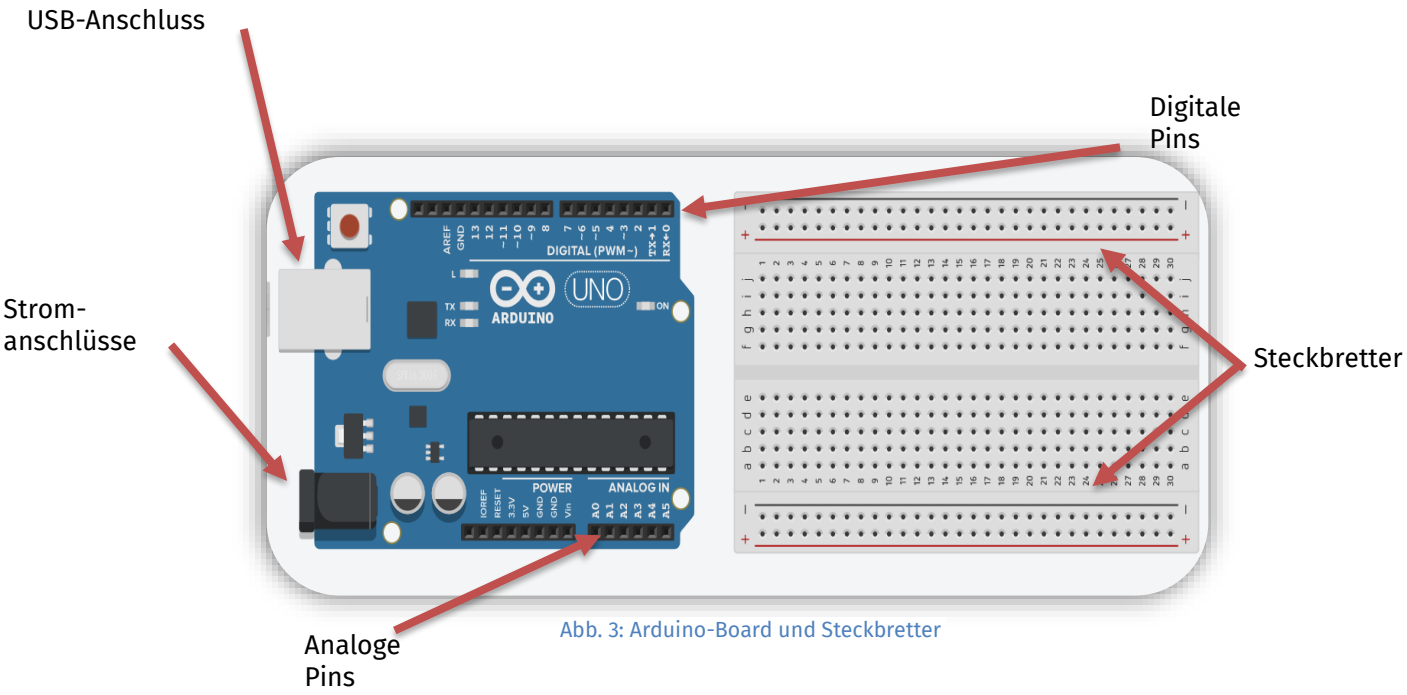


Abb. 3: Arduino-Board und Steckbretter

Das ist nur eine schematische Darstellung, in der einige unbedeutende Teile weggelassen wurden.
Selbst auf dem Schema ist noch mehr zu sehen, als ihr wirklich braucht. Alles, was ihr wirklich benötigt, wird jetzt kurz erläutert:

DER USB-ANSCHLUSS:

Damit verbindet ihr den Arduino später mit dem Computer, um euer Programm zu übertragen. Außerdem dient dieser Anschluss als Stromversorgung. Online ist unser virtueller Arduino natürlich immer mit Strom versorgt 😉.

STROMANSCHLÜSSE:

Manche Bauteile müssen mit Strom versorgt werden. Auch wenn es nicht explizit draufsteht, merkt euch einfach:

GND = – (Minuspol) **5V = + (Pluspol)**

Bitte benutzt nur die digitalen Pins 2 bis 13. Die anderen beiden haben eine Sonderfunktion!

Im Gegensatz zu den digitalen Pins werden die analogen Pins nur als Eingänge benutzt. Daher spricht man auch von den **ANALOGEN EINGÄNGEN**. Sie kennen nicht nur **HOCH (HIGH)** und **NIEDRIG (LOW)**, sondern insgesamt 1024 verschiedene Werte. Sind 0V angeschlossen, ist der Wert 0, bei 5V ist er 1023. Doch dazu später mehr.

Wenn ihr alles auf eurem Arduino gefunden habt, geht's an die Steckplatine.

Auf den **STECKPLATINEN** seht ihr viele Löcher. Diese sind unter der Oberfläche miteinander verbunden. Das markiert Tinkercad auch, wenn ihr den Mauszeiger über die Löcher haltet. So sieht das Ganze unter der Oberfläche aus:

Lücken, die ihr mit Kabeln überbrücken müsst

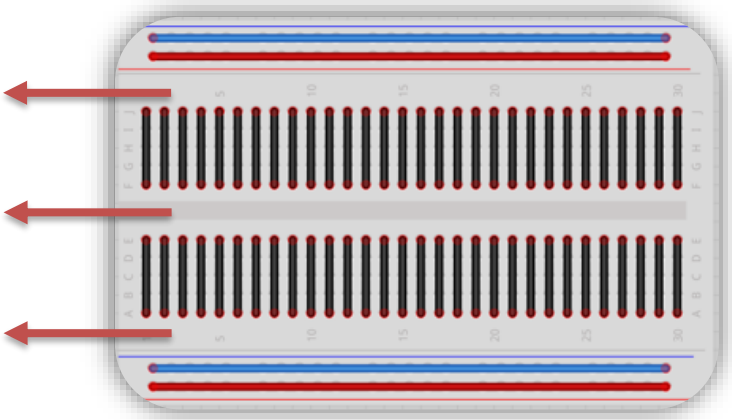


Abb. 4: Steckbrett mit gekennzeichneten Verbindungen

STROM AM BRETT

Ein guter Start ist es, erst einmal die äußeren Anschlussleisten am Steckbrett mit dem Arduino zu verbinden. So könnt ihr alle Bauelemente auf dem Steckbrett mit Strom versorgen; die äußeren Reihen sind also sowas wie eure Mehrfachsteckdosen.

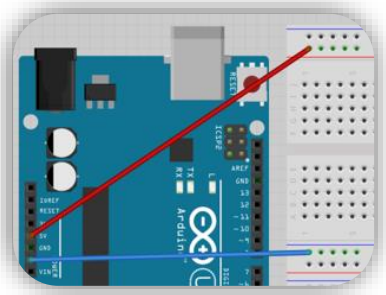
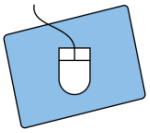


Abb. 5: Stromanschlüsse



Verbindet jetzt euren Arduino mit den zwei Steckkabeln (rot an 5V, blau an GND) mit dem Steckbrett. Befolgt dazu folgenden Schritte:

Abb. 6: Steckkabel

1. Zieht ein Steckbrett von den Komponenten auf die Oberfläche.
2. Klickt dazu zunächst auf die eine Stelle, die durch das Kabel verbunden werden soll, und dann auf die andere.

Es leuchtet

Bevor es ans Programmieren geht, werdet ihr schonmal lernen, wie das mit der Verkabelung funktioniert und ob da wirklich schon Strom fließt. Außerdem lernt ihr zwei wichtige Bauteile kennen: **LED** und **WIDERSTAND.**

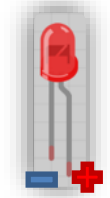


Abb. 7: LED

LEDs

LED ist die englische Abkürzung für **LIGHT EMITTING DIODE**. Es handelt sich also um ein Bauteil, das Licht aussendet. Dioden sind Bauteile, die Strom nur in eine Richtung durchlassen. Deswegen ist es sehr wichtig, dass sie richtig herum eingebaut werden. Zum Einbauen kann man sich an den Beinchen orientieren, denn *eins ist gekrümmt (und länger) und das andere gerade*. Das gekrümmte Bein muss mit dem **PLUSPOL** verbunden werden!

Leider sind LEDs übermäßig ehrgeizige Bauteile. Sie produzieren immer so viel Licht wie möglich! In einer realen Schaltung würden sie schnell kaputtgehen.

Deswegen muss der Strom immer begrenzt werden, wenn ihr mit LEDs arbeitet. Dafür gibt es **WIDERSTÄNDE**. Widerstände findet ihr rechts bei den Komponenten. Zieht einen Widerstand rüber, und verkabelt die LED wieder mit dem Strom, allerdings dieses Mal mit einem Widerstand von 220 Ω dazwischen (klickt zum Umstellen einfach auf den Widerstand). Wenn ihr alles richtiggemacht habt, erscheint keine Fehlermeldung an der LED.


Abb. 8: Widerstand

So sieht also das Schema zum Anschließen einer einzelnen LED aus.

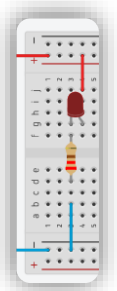


Abb. 9: LED mit Widerstand

Jetzt seid ihr dran. Nehmt den Widerstand und die LED zur Hand und steckt das Ganze mal bei euch nach.
Zur Erinnerung: Das lange, gekrümmte Beinchen zeigt Richtung Pluspol!

Wenn ihr alles richtiggemacht habt, sollte euer Ergebnis so aussehen:

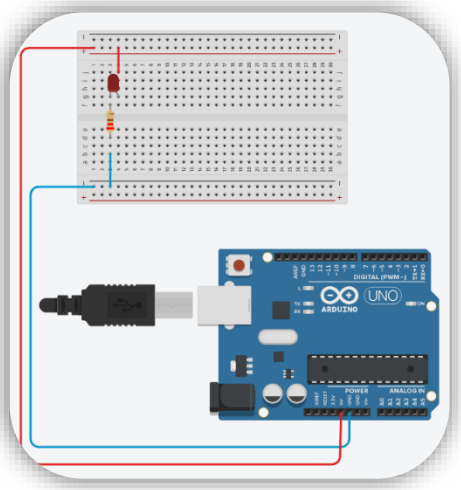


Abb. 10: Komplette Schaltung

Klickt nun auf „Simulation starten“. Falls eure LED nicht leuchtet:

- Leuchtet ein „i“ oder „?“ bei der LED auf?
- Steckt die LED richtig herum?
- Sind LED, Widerstand und Kabel in der gleichen Reihe?
- Habt ihr den richtigen Widerstand eingestellt und auf Ω gestellt?

Das war zum Einstieg schonmal prima! Jetzt seid ihr bereit für die richtig spannenden Dinge wie die Programmierung des Mikrocontrollers. Unter Code könnt ihr einen Editor öffnen. Klickt auf den Button, und wählt anschließend links oben „Blöcke“ aus (dort steht entweder Text und Blöcke, Blöcke oder Text).

Nun sollte euer Fenster rechts so aussehen:

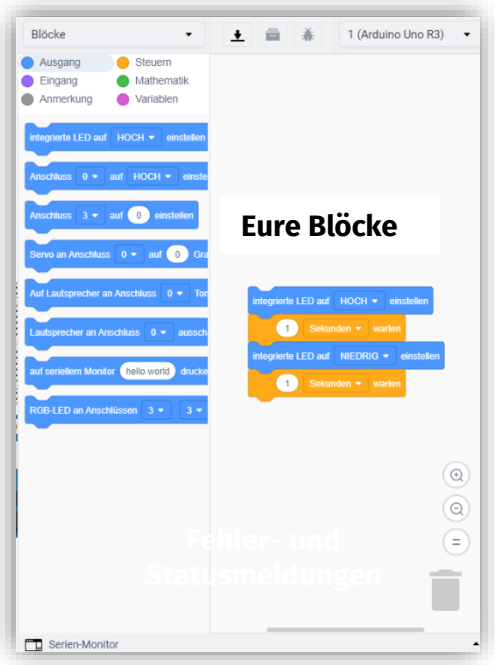


Abb. 11: Screenshot Tinkercad Codeansicht

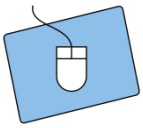
Bevor ihr loslegt, gibt es noch eine kurze Erklärung zu einem wichtigen Button:

Mit diesem Knöpfchen **ÜBERPRÜFT** ihr euer Programm. Dabei macht euch die Software auf eventuelle Fehler aufmerksam.

Station 0 – Einstieg

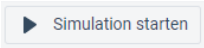
Genug Theorie, jetzt geht es an die Programmierung!

Verändert zunächst eure Schaltung ein wenig, schließlich wollt ihr die LED ja jetzt mit eurem Arduino steuern.



1. Stoppt dazu als Erstes die Simulation.
2. **VERBINDET** dann das obere Ende des Steckkabels, das vorher in die Plus-Leiste ging, mit einem **DIGITALEN PIN** des Arduino.
   Notiert euch hier, wo ihr das Kabel eingesteckt habt: _____
3. Begebt euch in das Codefenster, und nutzt die folgenden Blöcke, um die LED zum Blinken zu bringen:

Der **ZUSTAND** ist entweder **HOCH** für Strom an oder **NIEDRIG** für Strom aus.

4. Testet euer Programm, indem ihr auf 

drückt.

Hat es nicht geklappt? Dann überprüft die folgenden Punkte:

- Überprüft, ob ihr alle Blöcke wie bei einem Puzzle ineinandergesteckt habt, sonst weiß das Programm nicht, welche Blöcke ausgeführt werden sollen.
- Wartet ihr nach dem Einschalten des Stroms und nach dem Abschalten des Stroms? Ihr müsst zweimal einen warten-Block einbauen, damit die LED für das menschliche Auge sichtbar blinkt. Ansonsten läuft das Programm so schnell durch, dass es den Anschein macht, dass die LED immer oder nie leuchtet.
- Habt ihr den richtigen Anschluss gewählt? Die 8 oben in den Bildern muss durch den digitalen Anschluss ersetzt werden, an dem ihr eure LED angeschlossen habt.

Wenn es immer noch nicht klappt, vergleicht euren Sketch einmal mit folgender Musterlösung.

Abb. 13: Programmcode zum Einschalten einer LED

Hervorragend, ihr könnt jetzt also mit einem Mikrocontroller eine LED zum Blinken bringen!

Licht auf Tastendruck

Nur Dinge ein- und auszuschalten, ohne von außen Einfluss darauf zu haben, ist irgendwie langweilig, oder? Deshalb lernt ihr jetzt ein neues Element kennen: **DEN TASTER.**

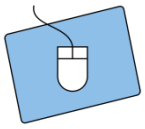
DER TASTER

Ein Taster ist ein Bauelement, das Strom leitet, solange der Knopf gedrückt ist. Nimmt man den Finger weg, fließt auch kein Strom mehr.

Ihr findet einen Drucktaster unter Komponenten. Er sieht so aus:

Abb. 15: Taster

Ein paar Kleinigkeiten sind beim Einbau eines Tasters jedoch zu beachten. Die eine Seite des Tasters kommt an den Plus-Pol, die andere Seite wird mit einem der digitalen Pins des Arduino verbunden. Zusätzlich muss diese Seite aber noch mit einem großen Widerstand (**100 kΩ**) mit dem Minus-Pol verbunden werden. Nutzt dazu (wie zu sehen in der Abbildung 15 oben) entweder 1a und 2a oder 1b und 2b.
*[**HINWEIS:** Eine eurer Plusleisten ist an den LED-Pin angeschlossen. Für den Taster benötigt ihr eine weitere Plusleiste.]*



Baut den Taster jetzt ein!

Eure Schaltung sollte dann so aussehen wie rechts auf dem Bild:

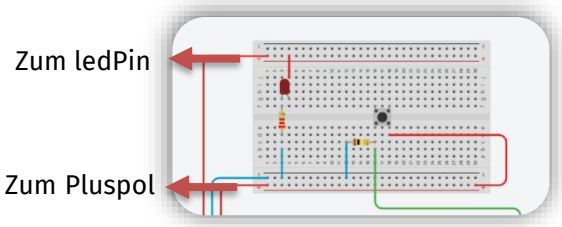


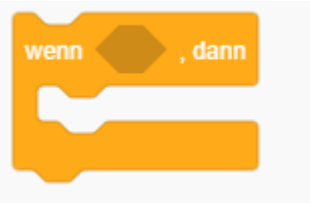
Abb. 16: Schaltung mit Taster und LED

Notiert hier, mit welchem digitalen Pin ihr den Taster verbunden habt: _____

[***Tipp:** Versucht doch einmal, den Stromfluss vom Arduino durch die Schaltung und wieder zurück nachzuvollziehen!*]

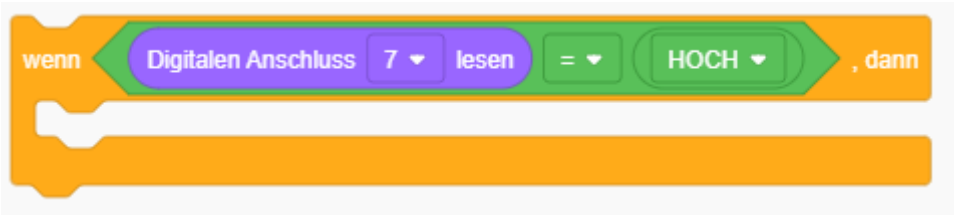
Bevor ihr jetzt mit dem Programmieren loslegt, müsst ihr zunächst euren **SKETCH** leeren.
Guckt euch genau an, an welchen Pins ihr die LED und den Taster angeschlossen habt.
Mit dem Taster werdet ihr nun die LED steuern. Dazu lernt ihr jetzt ein weiteres wichtiges Konstrukt kennen, das ihr als angehende Programmierende bald im Schlaf berherrschen werdet.

DIE Wenn-dann-ANWEISUNG

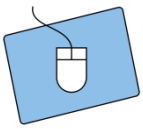


Die Wenn-dann -Anweisung (englisch: if-Anweisung) prüft, ob die angegebene Bedingung wahr ist. Wenn ja, werden die Anweisungen in dem Block ausgeführt. Wenn nicht, dann werden sie ausgelassen und mit dem **NÄCHSTEN BLOCK** im Programm weitergemacht.

Doch wie wird so eine Bedingung formuliert? Beispielsweise indem ihr testet, ob der Taster gedrückt ist, also der Eingang den Wert **HOCH** meldet! Eine mögliche Bedingung wäre also:



Die Blöcke oben findet ihr in den Kategorien **EINGANG** und **MATHEMATIK**. Der Wenn-Block befindet sich in der **STEUERUNG**. Wenn ihr den Vergleichsblock nicht findet, zieht einmal  aus dem Bereich **MATHEMATIK** in euer Programm, und klickt auf den kleinen Pfeil, der nach unten zeigt.



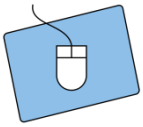
Versucht jetzt, die LED einzuschalten, sobald der Taster gedrückt wird.

Und sonst?

Ein Schritt ist getan, aber ihr habt bestimmt auch etwas festgestellt: Die LED geht jetzt an, wenn ihr den Taster drückt, aber sie geht nicht wieder aus, wenn ihr loslasst!

Was fehlt ist die Anweisung an den Arduino, was zu tun ist, wenn der Taster nicht gedrückt ist!
Dabei wird euch eine Erweiterung der Wenn-dann-Anweisung helfen:

Den oberen Teil kennt ihr ja schon. Hinzu kommt das **SONST**. Die Anweisungen, die darauffolgen, werden nur dann ausgeführt, wenn die Bedingung nicht erfüllt wurde.



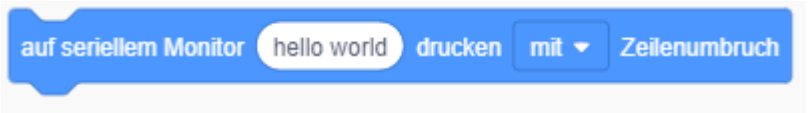
Schafft ihr es, euren Sketch so anzupassen, dass beim Loslassen des Tasters die LED wieder ausgeschaltet wird?

Arduino an Programmierenden!

Der Arduino kann nicht nur über das Ein- und Ausschalten von LEDs mit euch kommunizieren. Gerade in den späteren Projekten wird es wichtig, dass ihr euch zwischendurch **SENSORWERTE UND RECHENERGEBNISSE** auf dem Computer **ANZEIGEN** lasst. Dafür haben die Entwickler von Arduino ein hervorragendes Werkzeug eingebaut.

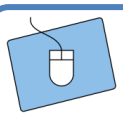
DER SERIELLE MONITOR

Der serielle Monitor zeigt euch Daten, die über das Kabel vom Arduino zum Computer geschickt wurden. Ihr könnt auf diese Ausgabe zugreifen, indem ihr den folgenden Block verwendet:



In das weiße Feld, in dem *hello world* steht, könnt ihr einen beliebigen Text oder sogar einen runden Block einfügen.

Wenn ihr die Daten jetzt sehen wollt, könnt ihr unter Code unten auf „Serien-Monitor“ klicken und euch so die Daten anzeigen lassen.



Verändert euren Sketch jetzt so, dass nach jedem Schaltvorgang auch ein entsprechender Text ausgegeben wird. Also **HOCH**, wenn der Schalter gedrückt ist, und sonst **NIEDRIG**.

Herzlichen Glückwunsch! Ihr habt alle Grundlagen gemeistert. Viel Spaß bei den weiteren Projekten!

Quellenverzeichnis:
Abb. 1, 2, 6, 10, 15, 17 – *Quelle: InfoSphere*
Abb. 3, 4, 5, 7, 8, 9, 16 – *Quelle: Screenshots von Tinkercad* https://www.tinkercad.com/
Alle Codeblock-Screenshots – *Quelle: Screenshots von Tinkercad* https://www.tinkercad.com/
angefertigt vom InfoSphere-Team