CE

Alle Versuche im Überblick

Adventskalender für Arduino 2017

Das Programmieren von Mikrocontrollern war früher nur etwas für Ingenieure und Informatiker. Die Arduino-Plattform ermöglicht dank übersichtlicher Hardware und einfach zu verstehender Software auf einmal jedem den Einstieg in die Mikrocontrollertechnik.

Der Name Arduino
Der Arduino kommt aus Italien und wurde nach dem italienischen König Arduino benannt, der bis ins Jahr 1005 in Ivrea, dem Firmensitz des Arduino-Herstellers, herrschte. König Arduino gab dort heute auch der Lieblingsbar der Arduino-Entwickler Massimo Banzi und David Cuartielles seinen Namen.

Nano-kompatible Platine

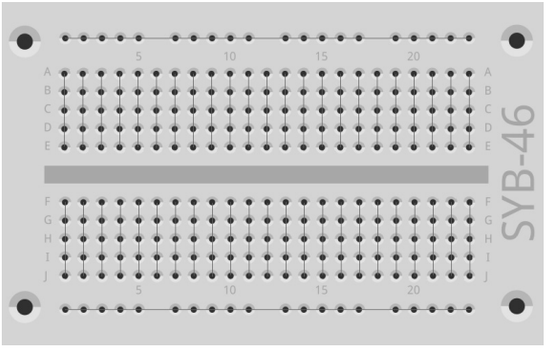
Die Arduino-Plattform bietet mittlerweile eine große Vielfalt an Platinen für unterschiedliche Anwendungszwecke. Dieser Adventskalender enthält eine zum Arduino-Nano-Standard kompatible Platine, die direkt auf ein Steckbrett gesteckt werden kann, um weitere Elektronik anzuschließen. Jeden Tag wird im Adventskalender ein Hardwareexperiment mit zugehörigem Programm vorgestellt.

Die meisten Experimente in diesem Adventskalender werden mit Snap4Arduino programmiert. Diese Programmiersprache basiert auf Scratch, einer der am leichtesten erlernbaren Programmiersprachen überhaupt. Später wird auch die klassische Arduino-IDE zum Programmieren vorgestellt. Die verwendeten Programme gibt es hier zum Download: **www.buch.cd**. Geben Sie dort den Code **15003-5** ein und folgen Sie den Anweisungen.

Steckbrett

Für den schnellen Aufbau elektronischer Schaltungen, ohne dass man löten muss, enthält der Adventskalender ein Steckbrett. Hier können elektronische Bauteile direkt in ein Lochraster gesteckt werden.

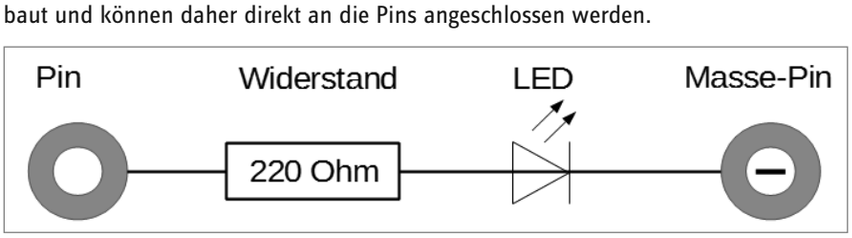
Bei diesem Steckbrett sind alle äußeren Längsreihen über Kontakte (X und Y) miteinander verbunden. Diese Kontaktreihen werden oft als Plus- und Minuspol zur Stromversorgung der Schaltungen genutzt. In den anderen Kontaktreihen sind jeweils fünf Kontakte (A bis E und F bis J) quer miteinander verbunden, wobei in der Mitte der Platine eine Lücke ist. So können hier größere Bauelemente eingesteckt und nach außen hin verdrahtet werden.



Die Verbindungen auf dem Steckbrett.

LEDs

LEDs (zu Deutsch: Leuchtdioden) leuchten, wenn Strom in Durchflussrichtung durch sie fließt. LEDs werden in Schaltungen mit einem pfeilförmigen Dreieckssymbol dargestellt, das die Flussrichtung vom Pluspol zum Minuspol oder zur Masseleitung angibt. Eine LED lässt in der Durchflussrichtung nahezu beliebig viel Strom durch, sie hat nur einen sehr geringen Widerstand. Um den Durchflussstrom zu begrenzen und damit ein Durchbrennen der LED zu verhindern, wird üblicherweise zwischen dem verwendeten Anschlusspin und der Anode der LED oder zwischen Kathode und Massepin ein 220-Ohm-Vorwiderstand eingebaut. Dieser Vorwiderstand schützt auch den Ausgang des Nano vor zu hohen Stromstärken. Die LEDs im Adventskalender haben den Vorwiderstand bereits eingebaut und können daher direkt an die Pins angeschlossen werden.



Schaltplan einer LED mit Vorwiderstand.

LED in welcher Richtung anschließen?
Die beiden Anschlussdrähte einer LED sind unterschiedlich lang. Der längere ist der Pluspol, die Anode, der kürzere die Kathode. Einfach zu merken: Das Pluszeichen hat einen Strich mehr als das Minuszeichen und macht damit den Draht optisch etwas länger. Außerdem sind die meisten LEDs auf der Minusseite abgeflacht, vergleichbar mit einem Minuszeichen. Auch leicht zu merken: Kathode = kurz = Kante.

Widerstände und ihre Farbcodes

Widerstände werden zur Strombegrenzung an empfindlichen elektronischen Bauteilen sowie als Vorwiderstände für LEDs verwendet. Die Maßeinheit für Widerstände ist Ohm. 1.000 Ohm entsprechen einem Kiloohm, abgekürzt kOhm. 1.000 kOhm entsprechen einem Megaohm, abgekürzt MOhm. Oft wird für die Einheit Ohm auch das Omega-Zeichen Ω verwendet.

Farbe	Widerstandswert in Ohm			
	1. Ring (Zehner)	2. Ring (Einer)	3. Ring (Multiplikator)	4. Ring (Toleranz)
Silber			10^{-2} = 0,01	±10 %
Gold			10^{-1} = 0,1	±5 %
Schwarz		0	10^{0} = 1	
Braun	1	1	10^{1} = 10	±1 %
Rot	2	2	10^{2} = 100	±2 %
Orange	3	3	10^{3} = 1.000	
Gelb	4	4	10^{4} = 10.000	
Grün	5	5	10^{5} = 100.000	±0,5 %
Blau	6	6	10^{6} = 1.000.000	±0,25 %
Violett	7	7	10^{7} = 10.000.000	±0,1 %
Grau	8	8	10^{8} = 100.000.000	±0,05 %
Weiß	9	9	10^{9} = 1.000.000.000	

Die farbigen Ringe auf den Widerständen geben den Widerstandswert an. Mit etwas Übung sind sie deutlich leichter zu erkennen als winzig kleine Zahlen, die man nur noch auf ganz alten Widerständen findet.

Die meisten Widerstände haben vier solcher Farbringe. Die ersten beiden Farbringe stehen für die Ziffern, der dritte bezeichnet einen Multiplikator und der vierte die Toleranz. Dieser Toleranzring ist meistens gold- oder silberfarben – Farben, die auf den ersten Ringen nicht vorkommen. Dadurch ist die Leserichtung immer eindeutig. Der Toleranzwert selbst spielt in der Digitalelektronik kaum eine Rolle. Die Tabelle zeigt die Bedeutung der farbigen Ringe auf Widerständen.

In welcher Richtung ein Widerstand eingebaut wird, ist egal. Bei LEDs dagegen spielt die Einbaurichtung eine wichtige Rolle.

Vorsichtsmaßnahmen

Auf keinen Fall sollte man irgendwelche Arduino-Pins miteinander verbinden und abwarten, was passiert.

Nicht alle Arduino-Pins lassen sich frei programmieren. Einige sind für die Stromversorgung und andere Zwecke fest eingerichtet.

Einige Arduino-Pins sind direkt mit Anschlüssen des Mikrocontrollers verbunden, ein Kurzschluss kann den Arduino komplett zerstören – zumindest theoretisch. Die Arduino-Platinen sind erstaunlich resistent gegen Schaltungsfehler. Verbindet man über eine LED zwei Pins miteinander, muss immer ein Vorwiderstand dazwischengeschaltet werden.

Für Logiksignale benötigen einige Arduino-kompatible Platinen 3,3 V, andere 5 V. Der Nano in diesem Adventskalender verwendet ein +5-V-Signal als logisch **high** bzw. **wahr**.

1. Tag

Heute im Adventskalender

· Nano (Arduino-kompatible Platine)

Nano vorbereiten

Um den Nano in Betrieb zu nehmen, braucht man:

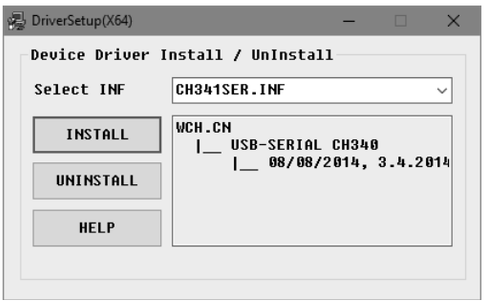
· PC mit Windows

· MicroUSB-Kabel

· Treiber

Verbunden werden PC und Nano mithilfe eines MicroUSB-Kabels. Sie brauchen sich nicht extra ein solches Kabel zu besorgen, fast alle modernen Smartphones verwenden diesen Steckertyp. Das Kabel wird gleichzeitig zur Stromversorgung wie auch zur Datenübertragung verwendet.

Schließen Sie das Kabel nach Möglichkeit an einen USB-2.0-Anschluss Ihres PCs an, da es an USB-3.0-Anschlüssen eher zu Verbindungsproblemen kommen kann.



Installation des Gerätetreibers.

Softwareinstallation in Kürze

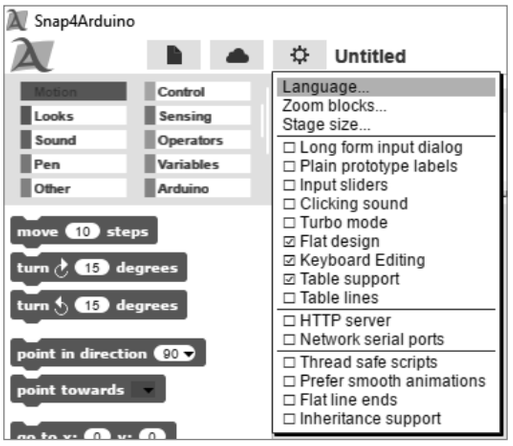
Hier die Treiberinstallation in vier Schritten:

1. Laden Sie sich die Beispielprogramme und den Gerätetreiber bei **www.buch.cd** herunter. Geben Sie dort den Code 15003-5 ein und folgen Sie den Anweisungen auf dem Bildschirm.

2. Entpacken Sie das ZIP-Archiv in einen beliebigen Ordner unterhalb Ihres Windows-Benutzerordners.

3. Schließen Sie den Nano über das USB-Kabel an und starten Sie dann mit der Datei `CH341SER.EXE` die Treiberinstallation. Zur Installation müssen Sie eine Anfrage der Windows-Benutzerkontensteuerung bestätigen.

4. Klicken Sie im Installationsdialog auf **Install** und warten Sie, bis die Bestätigung erscheint, dass der Treiber installiert wurde.

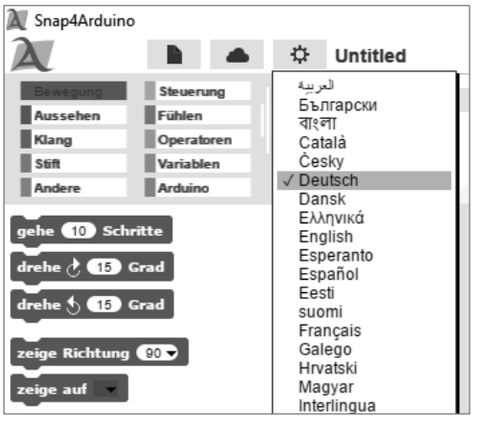


Das **Einstellungen**-Menü in Snap4Arduino.

LED blinkt

Für die meisten Projekte im Adventskalender verwenden wir die einfach zu erlernende Programmiersprache Snap4Arduino. Sie ist in den Downloads zum Adventskalender enthalten. Oder laden Sie sich die aktuelle Version bei **snap4aruino.org** herunter.

Klicken Sie in Snap4Arduino auf das **Einstellungen**-Symbol und wählen Sie im Menü **Language**.

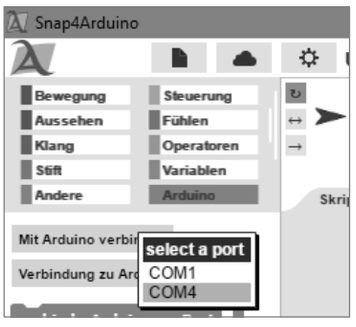


Wählen Sie in der Liste **Deutsch** aus.

Bevor Sie mit dem Programmieren beginnen können, muss eine Verbindung zwischen PC und Nano hergestellt werden. Dazu ist auf dem im Adventskalender mitgelieferten Nano die Software StandardFirmata vorinstalliert.

Schalten Sie in Snap4Arduino oben links auf die Blockpalette **Arduino** und klicken Sie auf **Mit Arduino verbinden**. Solange hier nur eine COM-Schnittstelle auftaucht, wählen Sie diese aus. Werden zwei Schnittstellen angezeigt, ist in den meisten Fällen die untere die richtige.

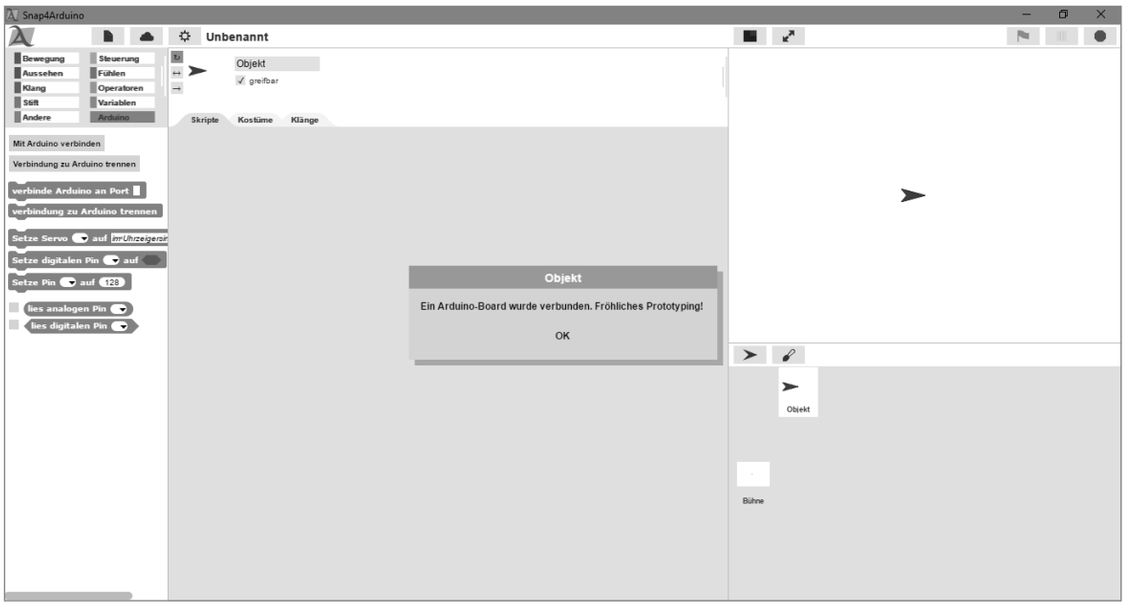
Bestätigen Sie die Meldung mit **OK**.



Schnittstelle für die Verbindung auswählen.

Fehler beim Öffnen eines neuen Programms

Beim Öffnen eines neuen Programms in Snap4Arduino geht oft die Verbindung zum Arduino verloren. Sollte ein Fehler auftreten, wenn Sie ein neues Programm starten, verbinden Sie den Nano mit einem Klick erneut.



Verbindung zum Nano erfolgreich hergestellt.

Das Programm

In Snap4Arduino braucht man beim Programmieren keinen Programmcode einzutippen. Die Blöcke werden einfach nur per Drag-and-drop aneinandergehängt. Die Blockpalette im linken Teil des Fensters enthält, nach Themen geordnet, die verfügbaren Blöcke.

Die Programme zum Adventskalender
Die Programme zum Adventskalender können Sie sich bei **www.buch.cd** herunterladen oder einfach jeden Tag anhand der Abbildung selbst zusammenbauen. Entpacken Sie die ZIP-Datei aus dem Download in ein Verzeichnis auf der Festplatte. Klicken Sie dann oben links in Snap4Arduino auf das Dateisymbol und wählen Sie **Importieren**, um die Programme, die im XML-Format vorliegen, in Snap4Arduino zu importieren. Einmal importierte Programme stehen danach in der eigenen Bibliothek, die über den Menüpunkt **Öffnen** erreichbar ist, zur Verfügung.

Das erste Programm `01led01` verwendet die wichtigsten Blöcke:

Wenn grünes Fähnchen angeklickt von der Palette **Steuerung** bildet den Start für die meisten Programme.

fortlaufend von der Palette **Steuerung** ist eine Endlosschleife, die permanent wiederholt wird.

Setze digitalen Pin... auf... von der Palette **Arduino** setzt einen der digitalen Pins des Nano auf einen Logikwert **wahr** oder **falsch**. Der grüne Block für Logikwerte ist auf der Palette **Operatoren** zu finden. Die beiden Werte **wahr** und **falsch** können direkt im Block umgeschaltet werden.

Die LED an Pin 13
Für Statusanzeigen ohne Zusatzhardware hat der Nano eine eigene LED, die über Pin 13 steuerbar ist.

warte... Sek von der Palette **Steuerung** lässt das Programm bis zum nächsten Schritt eine bestimmte Zeit lang warten.

Das Programm `01led01` lässt die LED auf dem Nano kurz blinken.

So funktioniert das Programm

Das Programm startet, wenn der Benutzer oben rechts auf das grüne Fähnchen klickt.

Eine **fortlaufend**-Schleife sorgt dafür, dass die LED endlos blinkt, und zwar so lange, bis der Benutzer auf das rote Stoppsymbol oben rechts in Snap4Arduino klickt.

Nachdem die LED an Pin 13 eingeschaltet ist, wird 0,01 Sekunden gewartet, solange leuchtet die LED. Danach wird die LED an Pin 13 wieder ausgeschaltet. Jetzt wartet das Programm eine halbe Sekunde. Dadurch blinkt die LED immer nur kurz auf und ist danach für eine relativ lange Zeit ausgeschaltet. Anschließend wiederholt sich der Zyklus.

Hinweis: Dezimalpunkt
Snap4Arduino verwendet, wie viele amerikanische Programme, den Punkt als Dezimaltrennzeichen, nicht das in Deutschland übliche Komma.

2. Tag

Heute im Adventskalender

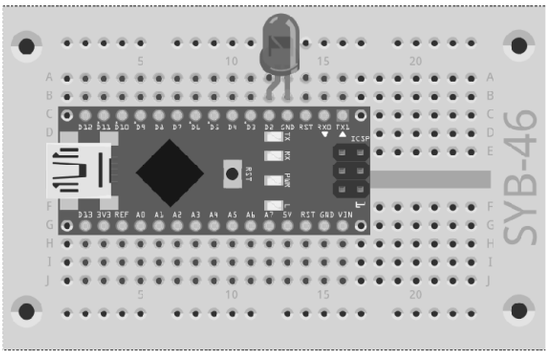
- 1 Steckbrett (SYB 46)
- 1 LED rot mit eingebautem Vorwiderstand

Wechselblinklicht

Ein einfaches Programm `021ed02` lässt zwei LEDs abwechselnd blinken.

Bauteile: 1 Steckbrett, 1 LED rot mit Vorwiderstand

Achten Sie beim Aufbau der Schaltung darauf, dass die Kathode (kurzer Draht) der LED mit dem GND-Pin verbunden ist, die Anode (langer Draht) mit dem D2-Pin.



LED-Wechselblinklicht am Nano.

Die Pins auf dem Nano

Alle Pins mit D… sind digitale Ein- oder Ausgänge, die die Werte **wahr** oder **falsch** (ein oder aus) annehmen können. Die Pins mit A… sind analoge Eingänge. GND-Pins sind Masseleitungen. Arduino-kompatible Platinen arbeiten mit unterschiedlichen Spannungen und haben dazu standardmäßig zwei verschiedene Pluspins. An Pin 3.3 liegen +3,3 V Spannung an. An Pin 5V liegen +5 V Spannung an. Der Nano im Adventskalender benötigt für ein logisches **Wahr**-Signal +5 V, manche anderen Platinen nur +3,3 V.

Das Programm `021ed02` lässt zwei LEDs abwechselnd blinken.

Das Programm

Das Programm `021ed02` lässt die eingebaute LED auf dem Nano und die extern angeschlossene LED abwechselnd blinken.

So funktioniert das Programm

Eine **fortlaufend**-Schleife sorgt dafür, dass die beiden LEDs abwechselnd endlos blinken, und zwar so lange, bis der Benutzer auf das rote Stoppsymbol oben rechts in Snap4Arduino klickt.

Nachdem die eingebaute LED an Pin 13 eingeschaltet ist, wird 0,01 Sekunden gewartet, damit StandardFirmata keinen Befehl „verschluckt“. Zwischen dem Setzen zweier Pins sollte bei den meisten Arduino-kompatiblen Platinen immer eine minimale Wartezeit eingebaut werden. Bei dem Nano im Adventskalender ist das nicht unbedingt nötig. Danach wird die LED an Pin 2 ausgeschaltet. Jetzt wartet das Programm eine halbe Sekunde.

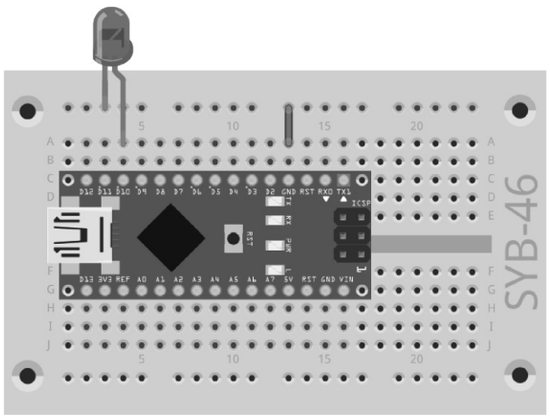
Anschließend wird auf die gleiche Weise die LED an Pin 2 ein- und die an Pin 13 ausgeschaltet. Nach einer weiteren halben Sekunde beginnt der Zyklus von vorne.

3. Tag

3. Tag

Heute im Adventskalender

· Schaltdraht (isoliert)



LED-Wechselblinklicht am Nano.

Schaltdraht

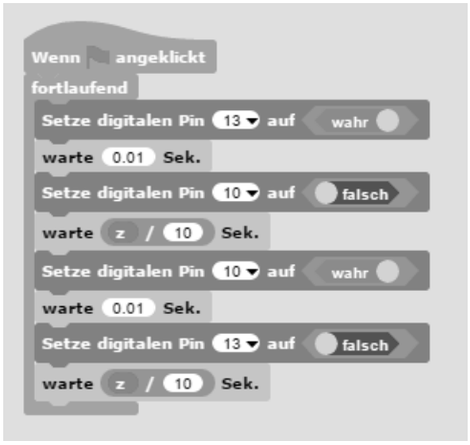
Heute ist Schaltdraht im Adventskalender enthalten. Damit stellen Sie kurze Verbindungsbrücken her, mit denen Kontaktreihen auf der Steckplatine verbunden werden. Schneiden Sie den Draht mit einem kleinen Seitenschneider je nach Experiment auf die passenden Längen zu. Um die Drähte besser in die Steckplatine stecken zu können, empfiehlt es sich, sie leicht schräg abzuschneiden, sodass eine Art Keil entsteht. Entfernen Sie an beiden Enden auf einer Länge von etwa einem halben Zentimeter die Isolierung.

LEDs blinken mit einstellbarer Geschwindigkeit

Das Experiment des 3. Tags lässt wieder zwei LEDs abwechselnd blinken. Allerdings können Sie die Geschwindigkeit einstellen.

Bauteile: 1 Steckbrett, 1 LED rot mit Vorwiderstand, 1 Drahtbrücke

Die Schaltung von heute zeigt den typischen Schaltungsaufbau auf dem Steckbrett. Eine der horizontalen Kontaktleisten wird als Masseleitung verwendet, die mit dem GND-Pin auf dem Nano über eine Drahtbrücke verbunden ist. Achten Sie beim Aufbau der Schaltung darauf, dass die Kathode (kurzer Draht) der LED in der Masseleiste steckt, die Anode (langer Draht) ist in dieser Schaltung mit Pin 10 verbunden.



Das Programm `031ed03` lässt zwei LEDs abwechselnd mit einstellbarer Geschwindigkeit blinken.

Das Programm

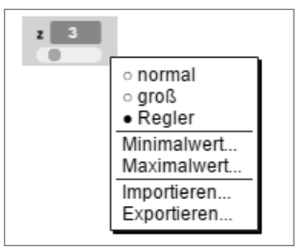
Das Programm `031ed03` funktioniert ähnlich wie das von gestern und lässt wieder die eingebaute LED auf dem Nano und die diesmal an Pin 10 angeschlossene externe LED abwechselnd blinken. Die Blinkfrequenz können Sie über einen Schieberegler auf dem Bildschirm steuern.

So funktioniert das Programm

Die **Fortlaufend**-Schleife lässt auch hier die beiden LEDs abwechselnd endlos blinken. Anstelle einer vom Programm fest vorgegebenen Zeit zwischen dem Umschalten wird eine Variable verwendet.

Variablen in Snap4Arduino

Variablen sind kleine Speicherplätze, in denen man sich während eines Programms eine Zahl oder irgendetwas anderes merken kann. Wenn das Programm beendet wird, werden diese Variablenspeicher automatisch wieder geleert. Variablen müssen in Snap4Arduino auf der Befehlspalette **Variablen** mit dem Button **Neue Variable** erst einmal angelegt werden, bevor man sie benutzen kann. Anschließend können Sie das Symbol der neu angelegten Variablen aus der Blockpalette in ein dafür vorgesehenes Feld eines Blocks im Programm ziehen. Auf der Blockpalette stehen zusätzlich verschiedene Blöcke zum Auslesen und Verändern der Variablen zur Verfügung.



Schieberegler bei einer Variablen.

Wurde eine Variable angelegt, erscheint sie als orangefarbenes Symbol auf der Bühne. Hier wird jederzeit der aktuelle Wert der Variablen angezeigt. Klicken Sie mit der rechten Maustaste auf dieses Symbol und wählen Sie die Option **Regler**.

Über die Optionen **Minimalwert** und **Maximalwert** stellen Sie die Werte 1 und 10 ein. Snap4Arduino kann mit den Schiebereglern nur ganze Zahlen einstellen, obwohl Variablen selbst jeden beliebigen Wert annehmen können.

Im Programm wird der eingestellte Wert der Variablen **z** durch 10 geteilt, um Werte zwischen 0,1 und 1,0 Sekunden zu erhalten. Auf der Blockpalette **Operatoren** gibt es Blöcke für die Grundrechenarten.

4. Tag

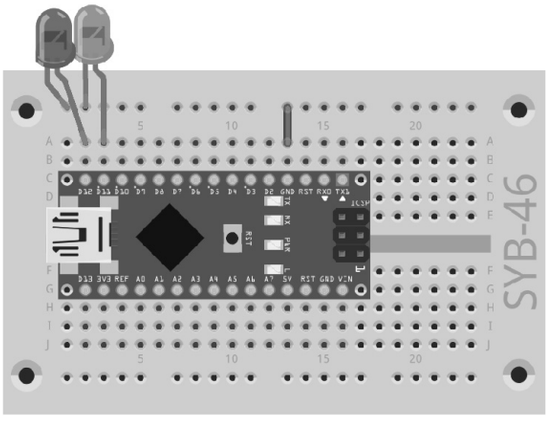
Heute im Adventskalender

· 1 LED grün mit eingebautem Vorwiderstand

LEDs blinken zufällig

Das Experiment des 4. Tags lässt drei LEDs in zufälliger Reihenfolge blinken. Die beiden externen LEDs stecken sehr dicht nebeneinander auf dem Steckbrett, da das Programm aufeinanderfolgende Pinnummern benötigt. Die dritte LED ist die auf der Nano-Platine aufgelötete LED mit der Pinnummer 13.

Bauteile: 1 Steckbrett, 1 LED rot mit Vorwiderstand, 1 LED grün mit Vorwiderstand, 1 Drahtbrücke



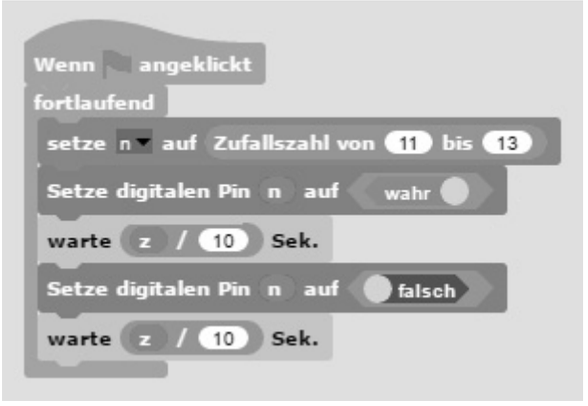
Drei LEDs blinken zufällig.

Das Programm

Das Programm `04led04` funktioniert ähnlich wie das von gestern. Auch hier werden in einer Endlosschleife nacheinander verschiedene digitale Pins ein- und ausgeschaltet. Die Pins werden diesmal zufällig ausgewählt.

Wie entstehen Zufallszahlen?

Gemeinhin denkt man, in einem Programm könne nichts zufällig geschehen – wie also kann ein Programm dann in der Lage sein, zufällige Zahlen zu generieren? Teilt man eine große Primzahl durch irgendeinen Wert, ergeben sich ab der x-ten Nachkommastelle Zahlen, die kaum noch vorhersehbar sind. Sie ändern sich auch ohne jede Regelmäßigkeit, wenn man den Divisor regelmäßig erhöht. Dieses Ergebnis ist zwar scheinbar zufällig, lässt sich aber durch ein identisches Programm oder den mehrfachen Aufruf des gleichen Programms jederzeit reproduzieren. Nimmt man aber eine aus einigen dieser Ziffern zusammengebaute Zahl und teilt sie wiederum durch eine Zahl, die sich aus der aktuellen Uhrzeitsekunde oder dem Inhalt einer beliebigen Speicherstelle des Computers ergibt, kommt ein Ergebnis heraus, das sich nicht reproduzieren lässt und daher als Zufallszahl bezeichnet wird.



Das Programm `04led04` lässt die LEDs zufällig blinken.

So funktioniert das Programm

Am Anfang jedes Durchlaufs der Endlosschleife wird die Variable **n** auf eine Zufallszahl zwischen 11 und 13 gesetzt. Diese gibt die Pinnummer der einzuschaltenden LED an. Deshalb benötigt die Schaltung drei aufeinanderfolgende Pinnummern.

Die Geschwindigkeit des Farbwechsels wird mithilfe einer Variablen **z** gesteuert, die über einen Schieberegler eingestellt wird und dann für jeden Schaltvorgang gilt.

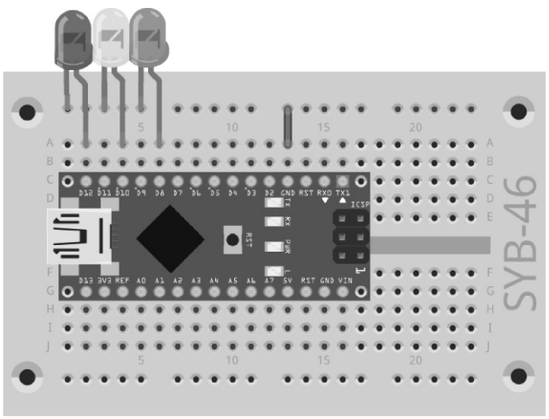
Die zufällig gewählte LED wird für die eingestellte Zeit eingeschaltet und danach genauso lange ausgeschaltet. Im nächsten Schleifendurchlauf wird wieder eine neue LED zufällig gewählt. Dabei kann es durchaus passieren, dass mehrmals hintereinander dieselbe LED aufleuchtet.

5. Tag

5. Tag

Heute im Adventskalender

- 1 LED gelb mit eingebautem Vorwiderstand



Ampel aus drei LEDs.

Das Programm `05ampel01` lässt die LEDs abwechselnd blinken.

Ampel

Das Experiment des 5. Tags schaltet eine Ampel aus drei LEDs in ihrem typischen Zyklus von Rot über Rot/Gelb nach Grün und über Gelb zurück nach Rot.

Bauteile: 1 Steckbrett, 1 LED rot mit Vorwiderstand, 1 LED gelb mit Vorwiderstand, 1 LED grün mit Vorwiderstand, 1 Drahtbrücke

Das Programm

Das Programm `05ampel01` funktioniert ähnlich wie das Programm des 3. Tags. Auch hier werden in einer Endlosschleife nacheinander verschiedene Kombinationen von LEDs ein- und ausgeschaltet. In den Zwischenphasen Rot/Gelb und Gelb leuchtet die Ampel jeweils 0,5 Sekunden, in den Phasen Rot und Grün je 3 Sekunden. Diese Zeiten lassen sich in den **warte...Sek**-Blöcken auch anders einstellen.

So funktioniert das Programm

Jeder Durchlauf der Endlosschleife startet mit der Rotphase der Ampel, bei der die gelbe und die grüne LED ausgeschaltet sind. Nach 3 Sekunden wird die gelbe LED zusätzlich eingeschaltet. Nach einer kurzen Rot/Gelb-Phase von 0,5 Sekunden werden die rote und die gelbe LED aus- und die grüne eingeschaltet. Die Grünphase dauert 3 Sekunden, darauf folgt eine kurze Gelbphase von 0,5 Sekunden, und die Endlosschleife startet mit Rot einen neuen Durchlauf.

6. Tag

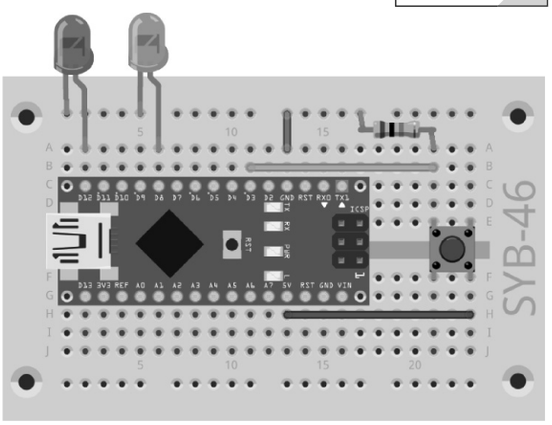
Heute im Adventskalender

- 1 Taster
- 1 10-kOhm-Widerstand (braun-schwarz-orange)

LEDs mit Taster umschalten

Das Experiment des 6. Tags schaltet über einen Taster zwei LEDs um.

Bauteile: 1 Steckbrett, 1 LED rot mit Vorwiderstand, 1 LED grün mit Vorwiderstand, 1 Taster, 1 10-kOhm-Widerstand (braun-schwarz-orange), 3 Drahtbrücken (unterschiedliche Längen)

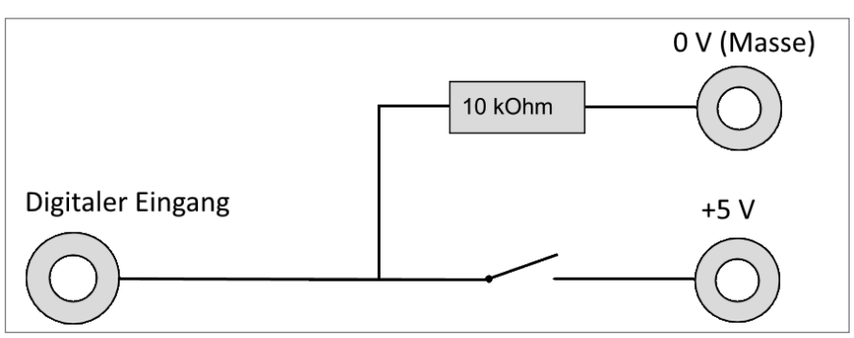


Ein Taster schaltet zwei LEDs um.

Digitale Pins können nicht nur Daten ausgeben, zum Beispiel über LEDs, sondern auch zur Dateneingabe eingesetzt werden. Zur Eingabe verwenden wir im heutigen Projekt einen Taster, der direkt auf die Steckplatine gesteckt wird. Der Taster hat vier Anschlusspins, wobei je zwei gegenüberliegende (großer Abstand) miteinander verbunden sind. Solange die Taste gedrückt ist, sind alle vier Anschlüsse miteinander verbunden. Im Gegensatz zu einem Schalter rastet ein Taster nicht ein. Die Verbindung wird beim Loslassen sofort wieder getrennt.

Liegt auf einem digitalen Eingang ein +5-V-Signal an, wird es als logisch **wahr** ausgewertet.

Bei offenem Taster hätte der Eingang keinen eindeutig definierten Zustand. Wenn ein Programm diesen Pin abfragt, kann es zu zufälligen Ergebnissen kommen. Um das zu verhindern, schließt man einen vergleichsweise sehr hohen Widerstand – üblicherweise 10 kOhm – gegen Masse. Dieser sogenannte Pull-down-Widerstand zieht den Status des Eingangspins bei geöffnetem Taster wieder nach unten auf 0 V. Da der Widerstand sehr hoch ist, besteht, solange der Taster gedrückt ist, auch keine Kurzschlussgefahr. Ist der Taster gedrückt, sind +5 V und die Masseleitung direkt über diesen Widerstand verbunden.



Schaltschema eines Tasters mit Pull-down-Widerstand.

Das Programm

Das Programm `06led06` schaltet, wenn der Taster gedrückt ist, die grüne LED an Pin 8 ein und die rote an Pin 12 aus. Solange der Taster nicht gedrückt ist, leuchtet nur die rote LED.

So funktioniert das Programm

Ein **falls...sonst...**-Block von der Blockpalette **Steuerung** führt die Blöcke innerhalb der oberen Klammer dann aus, wenn die Abfrage den Wert **wahr** ergibt. Andernfalls werden die Blöcke innerhalb der unteren Klammer ausgeführt.

Die Abfrage liest über einen Block von der Blockpalette **Arduino** den Wert des digitalen Pins 3 aus und prüft, ob er **wahr** ist. Digitale Eingänge können nur die Werte **wahr** und **falsch** annehmen.

Ist der Wert **wahr**, ist der Taster gedrückt. In diesem Fall wird die LED an Pin 8 ein- und die an Pin 12 ausgeschaltet. Ist der Taster nicht gedrückt, hat Pin 3 den Wert **falsch**, und die LEDs werden genau umgekehrt geschaltet.

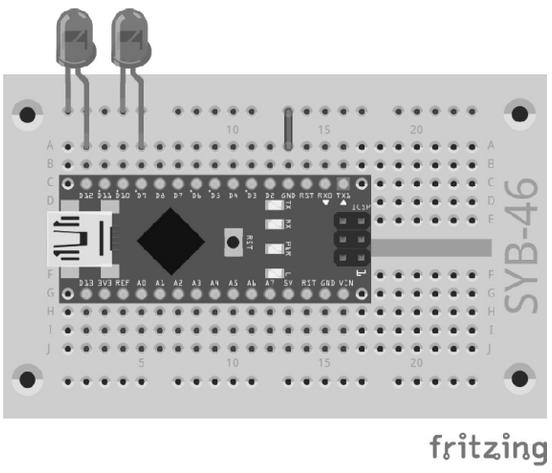
Das Programm `06led06` schaltet zwei LEDs mit einem Taster um.

7. Tag

7. Tag

Heute im Adventskalender

- 1 LED rot mit eingebautem Vorwiderstand



Eine LED wird gedimmt, die zweite leuchtet zum Vergleich mit voller Helligkeit.

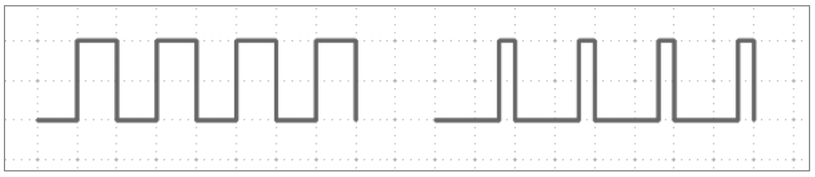
LED dimmen

Das Experiment des 7. Tags dimmt eine LED.

Bauteile: 1 Steckbrett, 2 LEDs rot mit Vorwiderstand, 1 Drahtbrücke

LEDs sind typische Bauteile zur Ausgabe von Signalen in der Digitalelektronik. Sie können zwei verschiedene Zustände annehmen, ein und aus, 0 und 1 oder **falsch** und **wahr**. Das Gleiche gilt für die als Ausgänge definierten digitalen Pins. Demnach wäre es theoretisch nicht möglich, eine LED zu dimmen.

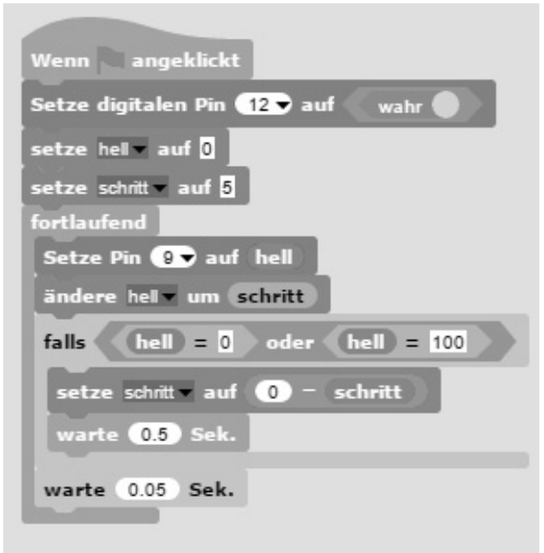
Mit einem Trick erreicht man es dennoch, die Helligkeit einer LED an einem digitalen Pin zu regeln. Lässt man eine LED schnell genug blinken, nimmt das menschliche Auge das nicht mehr als Blinken wahr. Die als Pulsweitenmodulation (PWM) bezeichnete Technik erzeugt ein pulsierendes Signal, das sich in sehr kurzen Abständen ein- und ausschaltet. Die Spannung des Signals bleibt immer gleich, nur das Verhältnis zwischen Level **falsch** (0 V) und Level **wahr** (+3,3 V) wird verändert. Das Tastverhältnis gibt das Verhältnis der Länge des eingeschalteten Zustands zur Gesamtdauer eines Schaltzyklus an.



Links: Tastverhältnis 50 % – rechts: Tastverhältnis 20 %.

Je kleiner das Tastverhältnis, desto kürzer ist die Leuchtzeit der LED innerhalb eines Schaltzyklus. Dadurch wirkt die LED dunkler als eine permanent eingeschaltete LED.

Pins für PWM-Signale
Die Pins 3, 5, 6, 9, 10 und 11 sind auf den Schaltbildern mit einem '-Symbol gekennzeichnet. Diese Pins können für Pulsweitenmodulation verwendet werden. Snap4Arduino bietet die anderen Pins im Block **Setze Pin... auf...** auch gar nicht zur Auswahl an.



Das Programm `07pwm01` dimmt eine LED am PWM-Ausgang.

Das Programm

Das Programm `07pwm01` dimmt die LED an Pin 9 zyklisch heller und dunkler. Die LED an Pin 12 leuchtet zum Vergleich in voller Helligkeit.

So funktioniert das Programm

Am Anfang wird Pin 12 als digitaler Pin auf **wahr** gesetzt. Danach werden zwei Variablen definiert: **hell** bezeichnet den PWM-Wert für die Helligkeit der LED, und **schritt** gibt die Schrittweite beim Dimmen an. Die aktuellen Werte beider Variablen werden in Echtzeit rechts oben auf der Bühne angezeigt. Jetzt beginnt eine Endlosschleife. Als Erstes wird bei jedem Schleifendurchlauf der aktuelle Wert der Variablen **hell** als PWM-Wert auf Pin 9 ausgegeben. Anschließend wird der Wert der Variablen **hell** um den Wert **schritt** erhöht.

Im nächsten Schritt wird überprüft, ob der Wert von **hell** die Grenze 0 oder 100 erreicht hat. In diesem Fall wird ein **oder**-Block eingesetzt, der wiederum Platz für zwei weitere Abfragen enthält. Ist von diesen beiden mindestens eine wahr, gibt der **oder**-Block den Wert **wahr** zurück, und der Inhalt des **falls**-Blocks wird ausgeführt.

Zwei Gleichheitsabfragen prüfen, ob der Wert der Variablen **hell** den Wert 0 oder 100 erreicht hat. Trifft das zu, wird die Variable **schritt** auf einen neuen Wert gesetzt. Da Snap4Arduino keine Möglichkeit bietet, das Vorzeichen einer Variablen umzukehren, verwenden wir den Operator Minus-Zeichen und subtrahieren den Wert der Variablen von 0, was das gleiche Ergebnis liefert. In dem Moment, in dem sich die Richtung umkehrt, die LED also entweder ganz hell oder ganz dunkel ist, wartet das Programm eine halbe Sekunde.

Zum Schluss wartet das Programm in jedem Schleifendurchlauf 0,05 Sekunden. Anschließend startet die Endlosschleife neu und liefert den LEDs einen neuen PWM-Wert.

8. Tag

Heute im Adventskalender

· 1 LED grün mit eingebautem Vorwiderstand

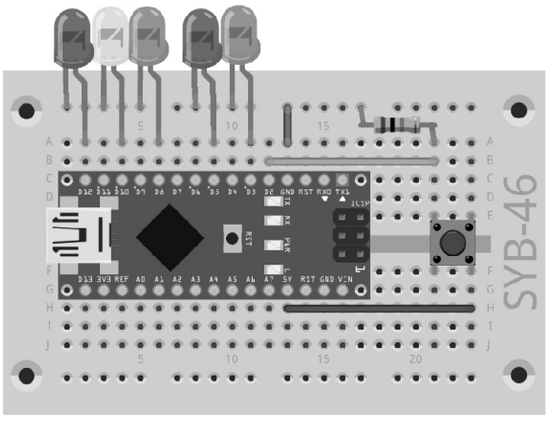
Fußgängerampel

Das Experiment des 8. Tags schaltet eine Fußgängerampel über einen Taster. Beim Drücken des Tasters startet der typische Ampelzyklus einer Fußgängerampel. Im Ruhezustand leuchtet die Fußgängerampel rot, die Verkehrsampel grün.

Bauteile: 1 Steckbrett, 2 LEDs rot mit Vorwiderstand, 1 LED gelb mit Vorwiderstand, 2 LED grün mit Vorwiderstand, 1 Taster, 1 10-kOhm-Widerstand (braun-schwarz-orange), 3 Drahtbrücken (unterschiedliche Längen)

Das Programm

Das Programm `08ampel02` steuert über einen Knetekontakt eine Fußgängerampel. Beim Klick auf das grüne Fähnchen wird die Ampel in die Grundstellung gebracht. Die Fußgängerampel leuchtet rot, die Verkehrsampel grün. Drückt man den Taster, läuft der Ampelzyklus durch. Nach einem Zyklus wartet das Programm wieder, bis der Taster erneut gedrückt wird.



Ein Taster schaltet die Fußgängerampel.

So funktioniert das Programm

Am Anfang des Programms werden die fünf LEDs in die Grundstellung der Ampel gebracht. Danach startet die Hauptschleife des Programms, die aus einer einzigen **falls**-Abfrage besteht. Solange der Taster nicht gedrückt ist, passiert gar nichts. Ist er gedrückt, läuft der Ampelzyklus durch, der gegenüber dem Programm des 5. Tags um die Schaltung der Fußgängerampel während der Rotphase der Verkehrsampel erweitert wurde.

Das Programm `08ampel02` steuert eine Fußgängerampel mit einem Knetekontakt.

9. Tag

9. Tag

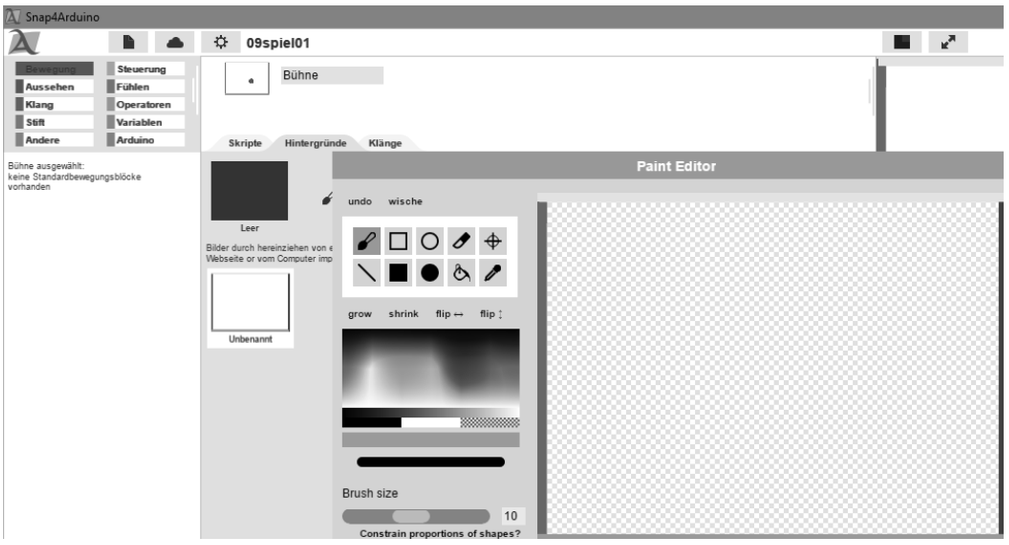
Heute im Adventskalender

- 1 LED blau mit eingebautem Vorwiderstand

Spiel auf dem Bildschirm

In dem Spiel fliegt ein Ball über die Bühne und prallt an den farbigen Kanten ab. Jedes Mal, wenn der Ball eine Kante berührt, blinkt die gleichfarbige LED kurz auf.

Bauteile: 1 Steckbrett, 1 LED rot mit Vorwiderstand, 1 LED gelb mit Vorwiderstand, 1 LED grün mit Vorwiderstand, 1 LED blau mit Vorwiderstand, 1 Drahtbrücke



Das Malprogramm in Snap4Arduino.

Snap4Arduino enthält ein eigenes kleines Malprogramm, mit dem man die Bühne bemalen kann. Klicken Sie dazu in der Objektpalette unten rechts auf das Symbol der Bühne, dort auf die Registerkarte **Hintergründe** und dann auf das Pinselsymbol **Paint a new costume**.

Mit dem gleichen Malprogramm, mit dem Sie die Bühne bemalen, können Sie auch aus dem Standardobjekt einen Kreis machen. Klicken Sie auf das Objekt auf der Objektpalette und schalten Sie dann auf die Registerkarte **Kostüme**. Auch hier finden Sie das Pinselsymbol für das Malprogramm. Schalten Sie danach auf die Registerkarte **Skripte**, um wieder in den Skriptbereich von Snap4Arduino zurückzukommen.

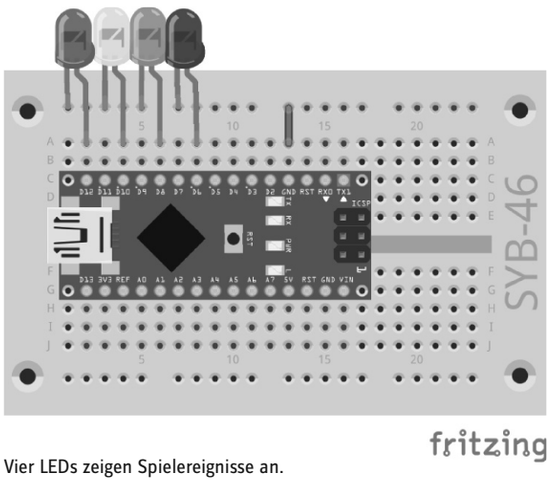
Das Programm

Das Programm `09spiel01` besteht aus fünf voneinander unabhängigen Programmblöcken. Das Hauptprogramm wird beim Klick auf das grüne Fähnchen gestartet und bewegt den Ball über die Bühne.

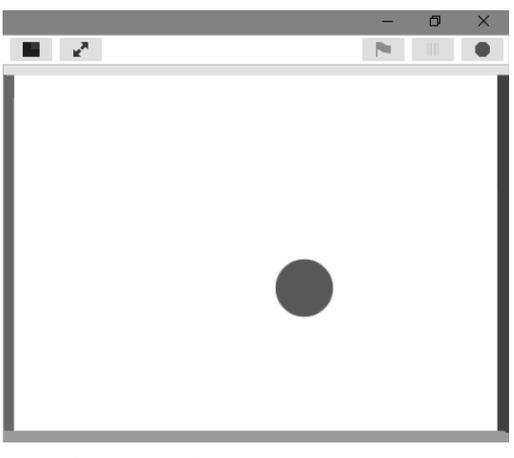
Die anderen vier Blöcke lassen je eine LED aufblinken, wenn der Ball eine Fläche der entsprechenden Farbe berührt.

So funktioniert das Programm

Beim Klick auf das grüne Fähnchen wird der Ball auf den Nullpunkt des Koordinatensystems in der Mitte der Bühne gesetzt und eine zufällige Bewegungsrichtung festgelegt. Jetzt beginnt eine **Fortlaufend**-Schleife, die in jedem Durchlauf prüft, ob der Ball am Rand anstößt, und ihn in diesem Fall abprallen lässt. Außerdem wird der Ball um acht Schritte in Bewegungsrichtung bewegt.



Vier LEDs zeigen Spielereignisse an.



Farbige Balken auf der Bühne.

Die vier anderen Programmblöcke werden immer dann ausgelöst, wenn der Ball eine bestimmte Farbe auf der Bühne berührt. Um diese Farbe auszuwählen, klicken Sie in das Farbfeld im **berühre ?**-Block. Jetzt können Sie die Farbe in der Farbpalette oder direkt auf der Bühne auswählen. Jeder dieser vier Blöcke lässt die farblich passende LED für 0,5 Sekunden aufblinken.

10. Tag

Heute im Adventskalender

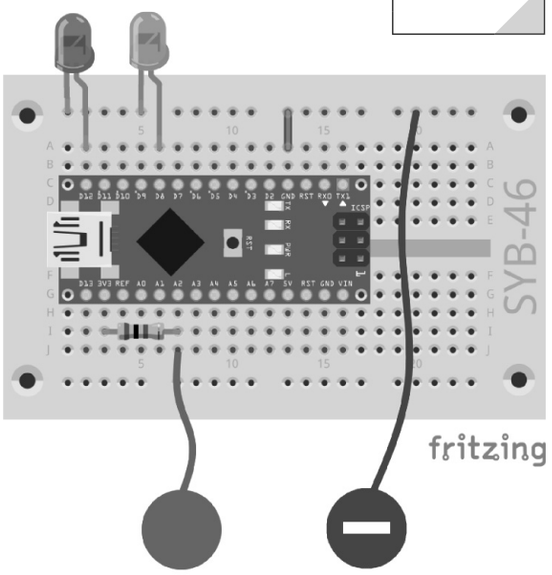
- 1 x Knete
- 1 20-MOhm-Widerstand (rot-schwarz-blau)

LED mit Knetesensor schalten

Das Experiment des 10. Tags zeigt, wie Sensorkontakte aus Knete funktionieren.

Bauteile: 1 Steckbrett, 1 LED rot mit Vorwiderstand, 1 LED grün mit Vorwiderstand, 1 20-MOhm-Widerstand (rot-schwarz-blau), 1 Drahtbrücke, 2 Knetekontakte

Der zweite Knetekontakt rechts ist der Massekontakt. Berühren Sie ihn am besten ständig, während Sie den Knetesensor links anfassen und wieder loslassen.

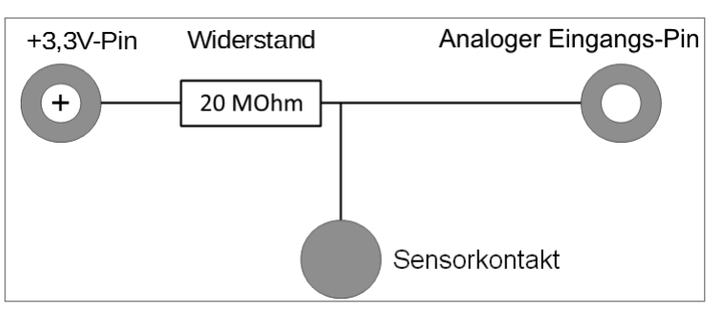


LED mit Knetesensor schalten.

So funktionieren Sensorkontakte

Der als Eingang geschaltete Pin ist über einen extrem hochohmigen Widerstand (20 MOhm) mit +3,3 V verbunden, sodass ein schwaches, aber eindeutig als High definiertes Signal anliegt. Ein Mensch, der nicht gerade frei in der Luft schwebt, ist immer geerdet und liefert über die elektrisch leitfähige Haut einen Low-Pegel. Berührt dieser Mensch einen Sensorkontakt, wird das schwache High-Signal von dem deutlich stärkeren Low-Pegel der Fingerkuppe überlagert und zieht den Pin auf Low-Pegel.

Wie hoch allerdings der Widerstand zwischen Hand und Masse wirklich ist, hängt von vielen Dingen ab, unter anderem von Schuhen und Fußböden. Barfuß im nassen Gras ist die Verbindung zur Masse der Erde am besten, aber auch auf Steinfußboden funktioniert es meistens gut. Holzfußböden isolieren stärker, Kunststoffbodenbeläge sind oft sogar positiv aufgeladen. Damit die Schaltung immer funktioniert, ist, ähnlich wie bei Sensortasten an Aufzügen und Türen, bei jeder Schaltung zusätzlich ein Massekontakt vorgesehen. Berührt man diesen und den eigentlichen Sensor gleichzeitig, ist die Masseverbindung auf jeden Fall hergestellt.



Schaltschema für Sensorkontakte am Arduino.

Knete leitet den Strom etwa so gut wie menschliche Haut. Sie lässt sich leicht in jede beliebige Form bringen, und ein Knetekontakt fasst sich viel besser an als ein einfaches Stück Draht. Die Fläche, mit der die Hand den Kontakt berührt, ist deutlich größer. So kommt es nicht so leicht zu einem „Wackelkontakt“. Schneiden Sie ein etwa zehn Zentimeter langes Stück des Schaltdrahts ab, entfernen Sie an beiden Enden etwa einen Zentimeter der Isolierung und stecken Sie das eine Ende in ein Stück Knete. Das andere Ende stecken Sie, wie auf der Abbildung gezeigt, in das Steckbrett.

Da Snap4Arduino die in vielen Arduino-kompatiblen Platinen eingebauten Pull-down-Widerstände immer einschaltet, werden digitale Eingänge meistens auf 0 gezogen und haben auch ohne Berührung einen Low-Pegel oder gar keinen definierten Zustand. Arduino-kompatible Platinen verfügen aber zusätzlich über analoge Eingänge, die sich sehr gut für Sensorkontakte eignen. Analoge Eingänge liefern Werte zwischen 0 (Low-Pegel) und 1023 (High-Pegel). Je nach Platinentyp sind Werte zwischen 100 und 200 gute Grenzwerte, um zwischen berührtem und nicht berührtem Sensorkontakt zu unterscheiden.

Das Programm `10knete01` schaltet die LEDs um, wenn der Knetekontakt berührt wird.

Das Programm

Das Programm `10knete01` schaltet die LED ein, wenn der Knetekontakt berührt wird, und wieder aus, wenn er losgelassen wird.

So funktioniert das Programm

Die Variable **x** zeigt auf der Bühne immer den aktuellen Wert des analogen Pins 2. Ist er kleiner als 200, wird innerhalb der **falls...sonst...**-Abfrage die LED am digitalen Pin 8 ein- und die LED an Pin 12 ausgeschaltet, andernfalls werden die LEDs umgekehrt geschaltet.

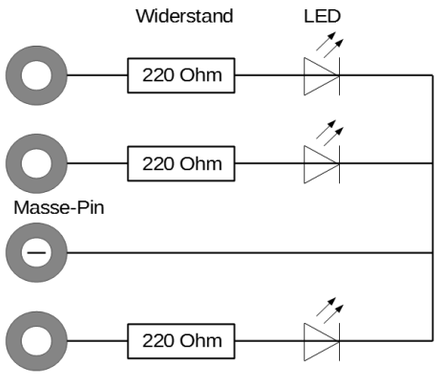
Anschlusspins einer RGB-LED.

11. Tag

11. Tag

Heute im Adventskalender

- 1 RGB-LED mit eingebauten Vorwiderständen



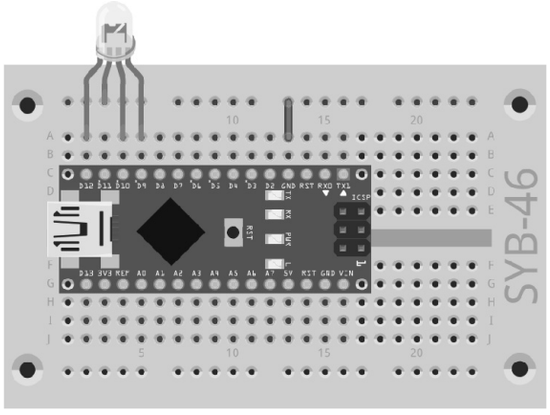
Schaltplan für eine RGB-LED mit drei Vorwiderständen.

RGB-LEDs

Eine normale LED leuchtet immer nur in einer Farbe. Die im Adventskalender verwendeten RGB-LEDs können wahlweise in mehreren Farben leuchten. Bei ihnen sind im Prinzip drei LEDs in verschiedenen Farben in einem transparenten Gehäuse eingebaut. Jede dieser drei LEDs hat eine eigene Anode, über die sie mit einem digitalen Arduino-Pin verbunden wird. Die Kathode, die mit der Masseleitung verbunden wird, ist nur einmal vorhanden. Deshalb hat eine RGB-LED vier Anschlussdrähte.

Die Anschlussdrähte der RGB-LEDs sind unterschiedlich lang, um sie eindeutig kenntlich zu machen. Anders als bei normalen LEDs ist die Kathode bei ihnen der längste Draht.

RGB-LEDs funktionieren wie drei einzelne LEDs und brauchen deshalb auch drei 220-Ohm-Vorwiderstände (rot-rot-braun). In den RGB-LEDs in diesem Adventskalender sind sie ebenfalls bereits eingebaut.



Lichteffekte mit der RGB-LED.

RGB-Lichteffekte

Das Experiment des 11. Tags zeigt unterschiedliche Farben auf einer RGB-LED.

Bauteile: 1 Steckbrett, 1 RGB-LED mit Vorwiderständen, 1 Drahtbrücke

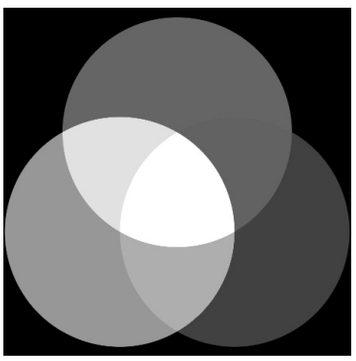
Das Programm

Das Programm `11rgb01` funktioniert ähnlich wie das von gestern. Auch hier werden in einer Endlosschleife nacheinander verschiedene digitale Pins ein- und ausgeschaltet. In diesem Fall handelt es sich um die drei Farbkomponenten der RGB-LED.

Verschiedene Farben auf einer RGB-LED.

Additive Farbmischung

RGB-LEDs nutzen die sogenannte additive Farbmischung. Dabei werden die drei Lichtfarben Rot, Grün und Blau addiert und ergeben am Ende reines Weiß. Im Gegensatz dazu verwendet ein Farbdrucker die subtraktive Farbmischung. Jede Farbe wirkt auf einem weißen Blatt wie ein Filter, der einen Teil des weiß reflektierten Lichts wegnimmt (also subtrahiert). Druckt man alle drei Druckerfarben übereinander, ergibt es Schwarz, das gar kein Licht mehr reflektiert.



Additive Farbmischung.

So funktioniert das Programm

Im Programm leuchten durch abwechselndes Ein- und Ausschalten immer mal eine, mal zwei Farbkomponenten. Dadurch wechselt die RGB-LED zwischen sechs verschiedenen Farben hin und her.

Die Geschwindigkeit des Farbwechsels wird über eine Variable **z** gesteuert, die am Anfang des Programms auf einen bestimmten Wert gesetzt wird und dann für jeden Farbwechsel gilt. Während des Programmablaufs lässt sich diese Variable vom Nutzer interaktiv einstellen.

12. Tag

Heute im Adventskalender

- 1 20-MOhm-Widerstand (rot-schwarz-blau)

RGB-Farbmischung mit PWM

Das Experiment des 12. Tags mischt über PWM-Signale die Farben der RGB-LED.

Bauteile: 1 Steckbrett, 1 RGB-LED mit Vorwiderständen, 2 20-MOhm-Widerstände (rot-schwarz-blau), 1 Drahtbrücke, 3 Knetekontakte

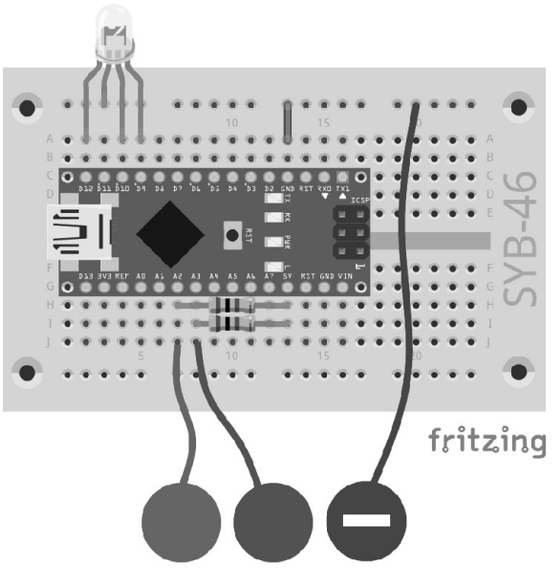
Das Programm

Das Programm `12rgb02` dimmt zwei der drei Farbkomponenten einer RGB-LED zyklisch heller und dunkler, während die Knetekontakte berührt werden. Dadurch werden verschiedene Mischfarben erzeugt. Lässt man einen Knetekontakt los, stoppt der Farbwechsel dieser Farbe. Die aktuellen Werte aller Variablen werden in Echtzeit rechts oben auf der Bühne angezeigt.

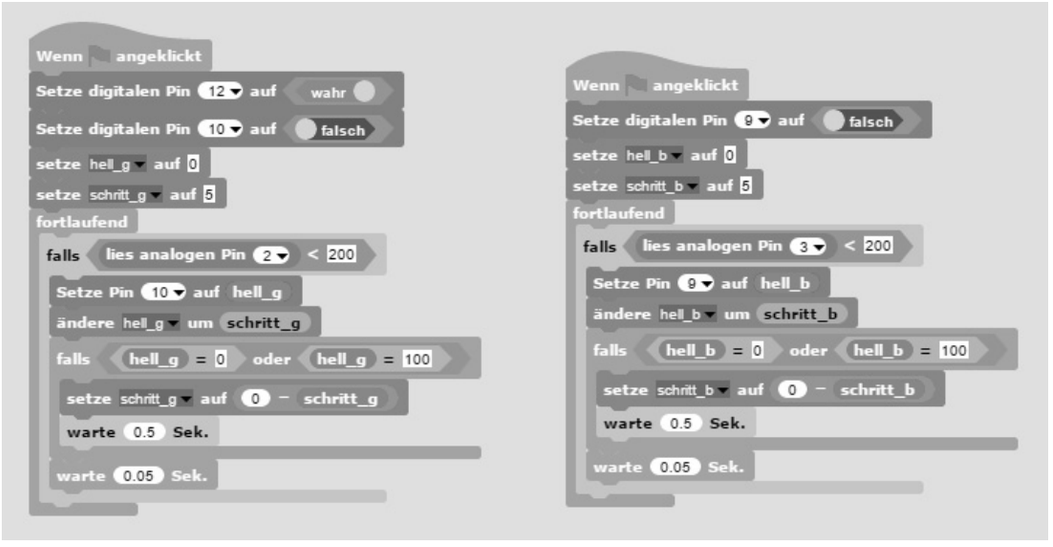
So funktioniert das Programm

Die Variablen **hell_g** und **hell_b** speichern die PWM-Helligkeitswerte der beiden Farbkomponenten Grün und Blau, die Variablen **schritt_g** und **schritt_b** die Schrittweite, in der die PWM-Werte geändert werden. Dabei wird nur zwischen +5 und -5 gewechselt. Die rote Komponente der RGB-LED leuchtet immer mit voller Helligkeit.

Beim Klick auf das grüne Fähnchen starten zwei unabhängige Programmblöcke für die grüne und die blaue Farbkomponente der RGB-LED. Eine **falls...**-Abfrage prüft, ob der Knetekontakt berührt wird. Ist das der Fall, wird die Helligkeit der entsprechenden Farbkomponente um die Schrittweite verändert. Wenn die Helligkeit einen der Grenzwerte 0 oder 100 erreicht hat, wird die Schrittweise umgekehrt, und der Zyklus des Farbwechsels wartet an dieser Stelle 0,5 Sekunden.



Farbmischung auf einer RGB-LED.



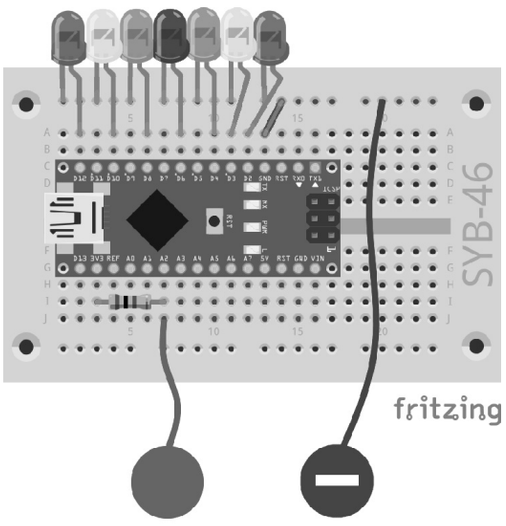
Das Programm `12rgb02` dimmt zwei der drei Farbkomponenten einer RGB-LED.

13. Tag

13. Tag

Heute im Adventskalender

- 1 LED gelb mit eingebautem Vorwiderstand



Lauflicht mit sieben LEDs.

Lauflicht

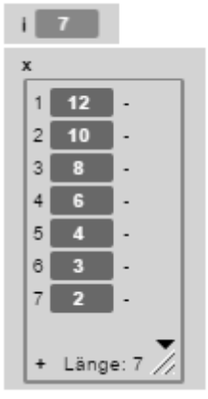
Lauflichter sind immer wieder beliebte Effekte, und das nicht nur in der Werbung und in Partyräumen. Das Experiment des 13. Tags lässt sieben LEDs als Lauflicht leuchten, während der Knetekontakt berührt wird.

Bauteile: 1 Steckbrett, 2 LEDs rot mit Vorwiderstand, 2 LEDs gelb mit Vorwiderstand, 2 LEDs grün mit Vorwiderstand, 1 LED blau mit Vorwiderstand, 1 20-MOhm-Widerstand (rot-schwarz-blau), 1 Drahtbrücke, 2 Knetekontakte

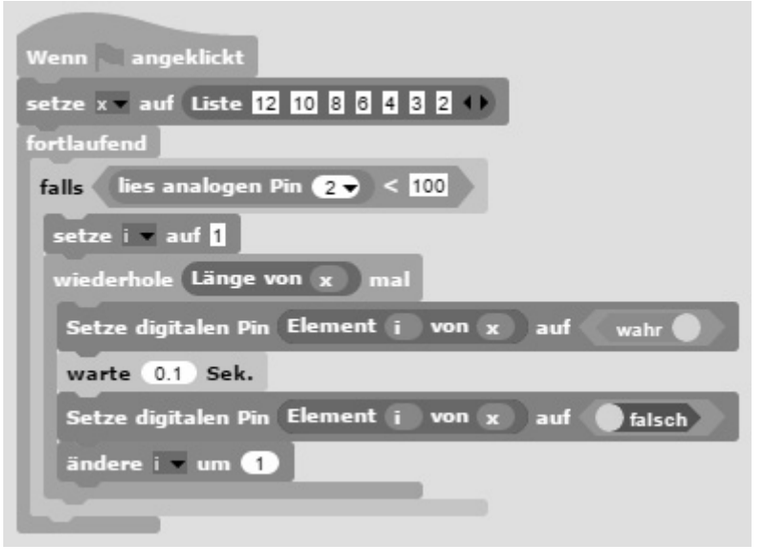
Das Programm

Das Programm `13lauflicht01` verwendet eine Listenvariable, in der die Pinnummern der digitalen Pins, die für die LEDs verwendet werden, gespeichert sind. Listen müssen in Snap4Arduino nicht durch das Programm gefüllt werden, stattdessen können Sie sie direkt im Programmcode mit Anfangswerten vorbelegen.

Eine Schleife läuft über die Länge der Liste und schaltet die LEDs nacheinander einzeln für je 0,1 Sekunden ein. Diese Schleife wird endlos wiederholt.



Die Listenvariable wird auf der Bühne angezeigt.



Das Programm `13lauflicht01` steuert ein Lauflicht mithilfe einer Listenvariablen.

So funktioniert das Programm

Nach Klick auf das grüne Fähnchen wird als Erstes eine Listenvariable **x** mit den Pinnummern der sieben für die LEDs verwendeten Pins angelegt. Danach beginnt die Hauptschleife des Programms, die wieder aus einer einzigen **falls...**-Abfrage besteht.

Wird der Knetekontakt berührt, wird der Schleifenzähler **i** auf 1 gesetzt. Anschließend läuft eine Schleife so oft, wie die Liste Elemente enthält. Der digitale Pin, der dem jeweiligen Listenelement entspricht, wird für 0,1 Sekunden eingeschaltet. Danach wird der Schleifenzähler um 1 erhöht. So blinkt jede LED einmal kurz auf.

14. Tag

Heute im Adventskalender

- 1 10-kOhm-Widerstand (braun-schwarz-orange)

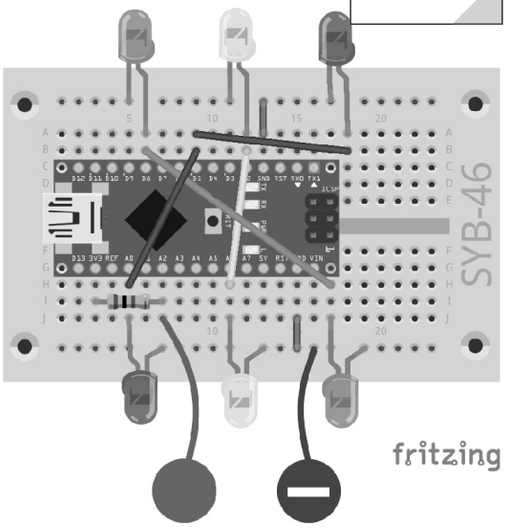
Dieser Widerstand wird erst in den nächsten Tagen benötigt.

LED-Würfel

Die typischen Spielwürfel, die ein bis sechs Augen zeigen, kennt jeder und hat jeder zu Hause. Wesentlich cooler ist ein elektronisch gesteuerter Würfel, der auf Tastendruck die Augen leuchten lässt – aber nicht einfach eine bis sechs LEDs in einer Reihe, sondern in der Anordnung eines Spielwürfels. Diese haben Augen in der typischen quadratischen Anordnung, für die man sieben LEDs braucht. Für die Ansteuerung der LEDs werden nur vier statt sieben digitale Pins benötigt, da ein Würfel zur Darstellung gerader Zahlen die Augen paarweise nutzt.

Bauteile: 1 Steckbrett, 2 LEDs rot mit Vorwiderstand, 2 LEDs gelb mit Vorwiderstand, 2 LEDs grün mit Vorwiderstand, 1 20-MOhm-Widerstand (rot-schwarz-blau), 2 Drahtbrücken, 2 Knetekontakte

In der Schaltung werden beide seitlichen Schienen des Steckbretts für Masseleitungen benötigt. Daher ist der Knetekontakt diesmal etwas anders angeschlossen. Es werden nur sechs extern angeschlossene LEDs verwendet, als mittlere LED für die Anzeige ungerader Würfelzahlen wird die auf dem Nano eingebaute LED an Pin 13 eingesetzt.



LED-Würfel mit sieben LEDs mithilfe des Knetesensors schalten.

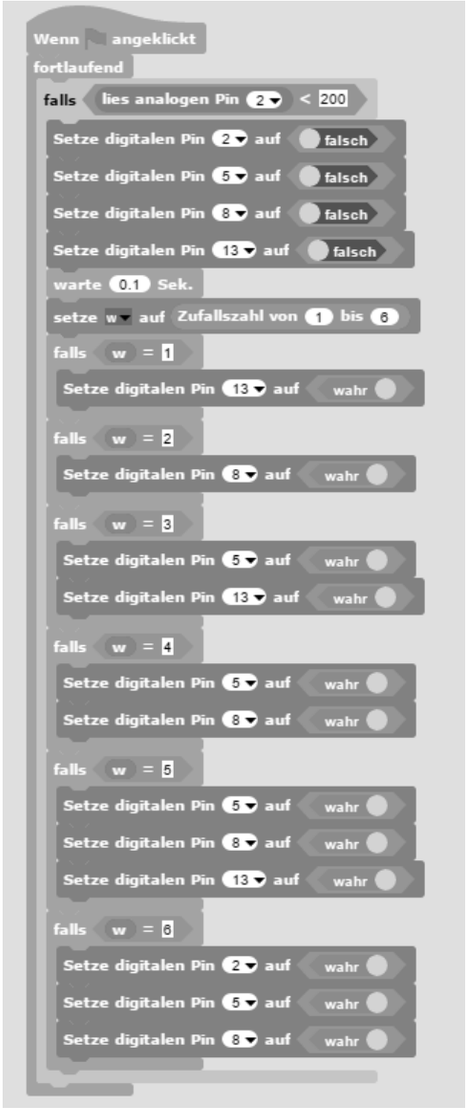
Das Programm

Das Programm `14wuerfel01` simuliert einen sechsseitigen Spielwürfel. Berührt man den Knetekontakt, wird gewürfelt.

So funktioniert das Programm

Wenn der Benutzer auf das grüne Fähnchen klickt, startet eine Endlosschleife, die immer wieder prüft, ob der Knetekontakt berührt wird. Ist das der Fall, werden als Erstes die vier für die LEDs verwendeten Pins ausgeschaltet, um das zuvor angezeigte Würfelergebnis zu löschen.

Anschließend wird in der Variablen **w** eine Zufallszahl zwischen 1 und 6 gespeichert. Für jedes mögliche Ergebnis gibt es einen eigenen **falls...**-Block, der die entsprechenden LEDs einschaltet.



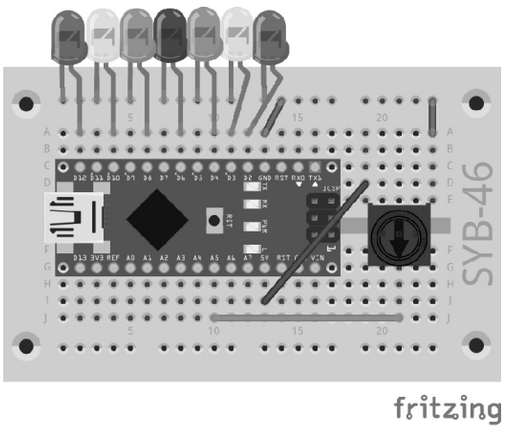
Das Programm `14wuerfel01` simuliert einen sechsseitigen Spielwürfel.

15. Tag

15. Tag

Heute im Adventskalender

- 15-kOhm-Potenziometer



Lauflicht mit sieben LEDs und Potenziometer.

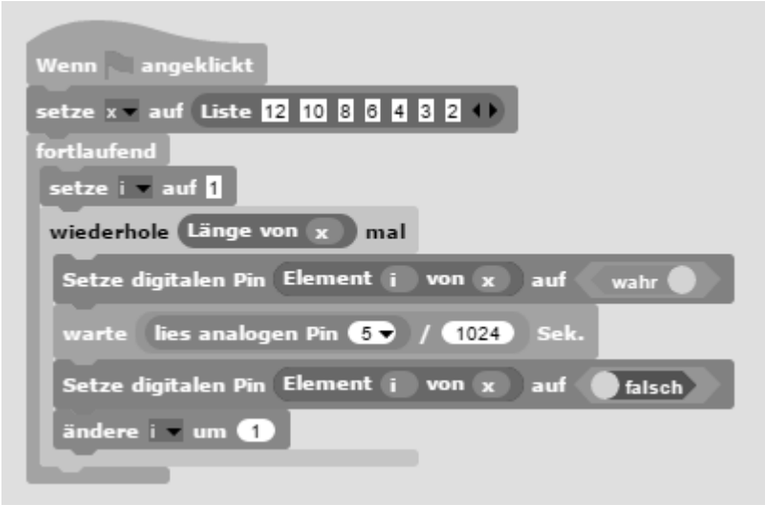
Lauflicht mit Potenziometer steuern

Das Potenziometer aus dem Adventskalender von heute ist ein einstellbarer Widerstand, der Werte zwischen 0 Ohm und 15 kOhm annehmen kann, indem man den Knopf dreht. Mit dem Potenziometer lässt sich ein Spannungsteiler bauen, der eine beliebige Spannung zwischen 0 V und +5 V liefern kann. Diese analoge Spannung muss in einen digitalen Wert umgerechnet werden, der dann vom Programm weiterverarbeitet wird.

Bauteile: 1 Steckbrett, 2 LEDs rot mit Vorwiderstand, 2 LEDs gelb mit Vorwiderstand, 2 LEDs grün mit Vorwiderstand, 1 LED blau mit Vorwiderstand, 1 15-kOhm-Potenziometer, 4 Drahtbrücken (unterschiedliche Längen)

Das Programm

Das Programm `15lauflicht02` steuert eine LED-Leiste anhand der Einstellung des Potenziometers.



Das Programm `15lauflicht02`.

So funktioniert das Programm

Die Nummern der für die LEDs verwendeten Pins sind wieder in einer Liste gespeichert. Eine Schleife schaltet nacheinander jede LED für eine bestimmte Zeit ein und danach wieder aus.

Der analoge Pin 5 wird in jedem Schleifendurchlauf abgefragt, und sein Wert, der zwischen 0 und 1023 liegen kann, wird durch 1024 geteilt, um einen Wert zwischen 0 und 1 zu erhalten. Dieser legt fest, wie lange jede einzelne LED leuchtet.

Die analogen Eingänge des Arduino werten einen analogen Spannungswert aus und liefern digitale Werte zwischen 0 und 1023. Dabei steht **0** für 0 V und **1023** für +5V Spannung am jeweiligen Pin.

16. Tag

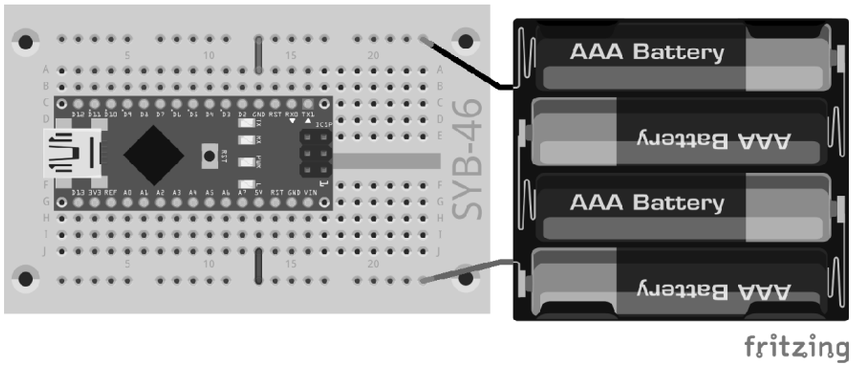
Heute im Adventskalender

· Batteriekasten

Nano ohne PC nutzen

Der Nano kann auch ohne PC genutzt werden und ein gespeichertes Programm abarbeiten. Dazu benötigt er eine externe Stromversorgung. Das kann ein USB-Handyladegerät, eine Powerbank oder auch eine Batterie sein. Heute ist im Adventskalender ein Batteriekasten enthalten, der mit vier AAA-Batterien eine Spannung von 6 V liefert und mit Akkus 4,8 V, was zur Stromversorgung des Nano ebenfalls ausreicht. Die Batterien sind nicht enthalten.

Bauteile: 1 Steckbrett, 1 Batteriekasten, 1 Drahtbrücke



Der Batteriekasten wird an die Pins 5V und GND am Nano angeschlossen.

Schließen Sie den Batteriekasten noch nicht an, da Sie den Nano so lange über den PC mit Strom versorgen, bis das neue Programm übertragen ist.

Die Arduino-IDE

Snap4Arduino benötigt über die auf dem Nano installierte Software StandardFirmata eine ständige Verbindung zwischen PC und Nano. Um den Nano eigenständig ohne PC nutzen zu können, brauchen Sie die Arduino-IDE. Hier schreiben Sie die Programme in der Programmiersprache C und übertragen sie dann direkt auf den Nano. Danach können Sie die Verbindung zum PC trennen.

Arduino ausschalten

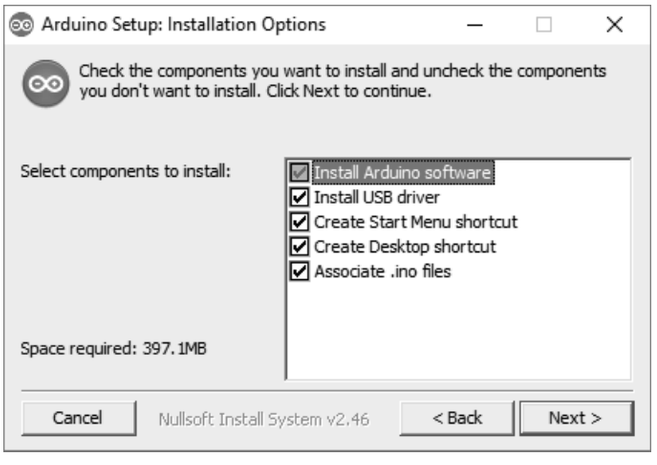
Der Arduino hat keinen Ausschalter, Sie brauchen einfach nur die Stromversorgung zu trennen, und er schaltet sich ab. Beim nächsten Einschalten startet automatisch das zuletzt gespeicherte Programm. Das Gleiche passiert, wenn Sie den Reset-Taster drücken.

Laden Sie sich den Windows Installer für die aktuelle Version der Arduino-IDE bei **www.arduino.cc/en/Main/Software** herunter oder verwenden Sie einfach die Datei `arduino-windows.exe` aus den Downloads zum Adventskalender. Unter Windows 10 können Sie die Arduino-IDE auch aus dem Windows Store herunterladen und installieren.

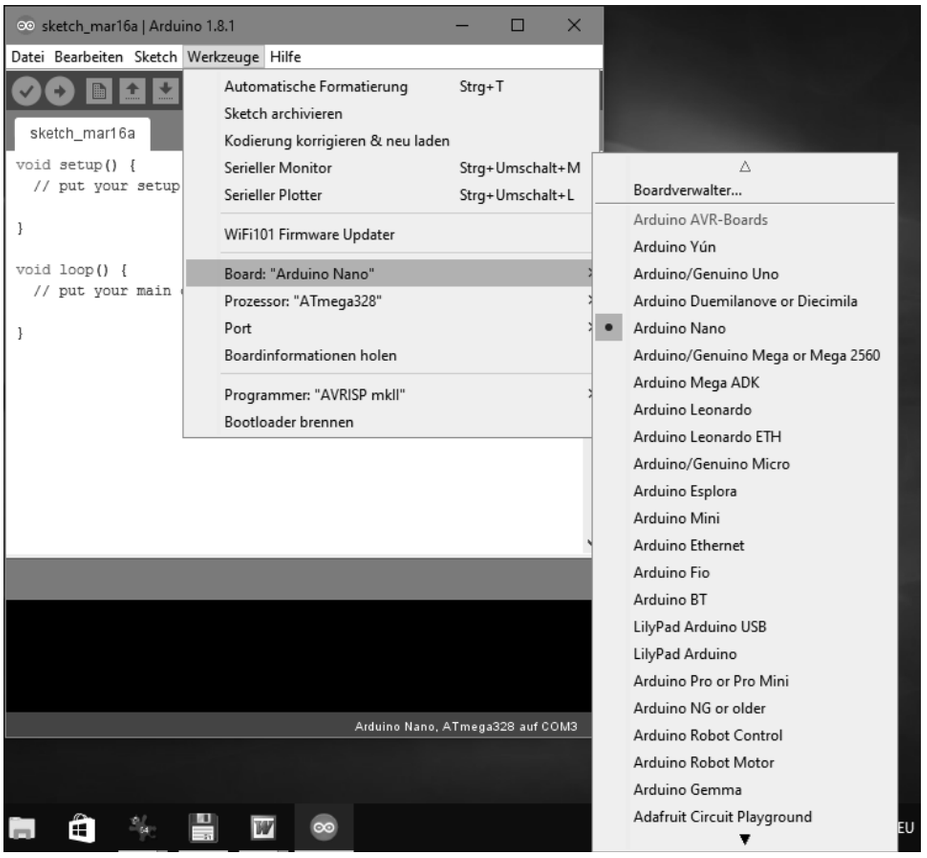
Achten Sie dabei darauf, dass im Dialogfeld **Installation Options** alle Häkchen gesetzt sind. Je nach Windows-Konfiguration ist eine Bestätigung der Benutzerkontensteuerung erforderlich.

Starten Sie nach der Installation die Arduino-IDE. Sollte Snap4Arduino noch laufen, beenden Sie es vorher. Wählen Sie im Menü der Arduino-IDE **Werkzeuge/Port**. Hier wird in den meisten Fällen nur ein einziger serieller Port angezeigt. Setzen Sie hier das Häkchen.

Wählen Sie anschließend über den Menüpunkt **Werkzeuge/Board** den **Arduino Nano**, wenn er nicht bereits automatisch erkannt wurde.



Installation der Arduino-IDE.



In der Arduino-IDE das passende Board wählen.

Wählen Sie im Menü **Datei/Beispiele/01.Basics./Blink** und öffnen Sie damit ein einfaches Beispielprogramm, das die auf dem Nano eingebaute LED blinken lässt.

Klicken Sie oben links auf das runde Symbol mit dem Pfeil, um das Programm auf den angeschlossenen Nano hochzuladen – auch als Flashen bezeichnet. Sobald das Hochladen abgeschlossen ist, blinkt die LED auf dem Nano. Das Programm braucht also nicht extra gestartet zu werden.

Trennen Sie die USB-Verbindung zum PC. Damit schaltet sich die LED aus, da die Stromversorgung fehlt. Schließen Sie danach den Batteriekasten mit den vier Batterien oder Akkus wie in der Abbildung gezeigt an. Sofort beginnt die LED wieder zu blinken, da der Nano automatisch das zuletzt geflashte Programm startet.

StandardFirmata flashen
Das Blinkprogramm hat auf dem Nano die vorinstallierte StandardFirmata überschrieben, da immer nur ein Programm installiert sein kann. Bevor Sie Snap4Arduino wieder benutzen können, müssen Sie mit der Arduino-IDE die StandardFirmata neu auf den Nano flashen. Sie finden sie im Menü **Datei/Beispiele/Firmata/StandardFirmata**. Denken Sie daran, den Batteriekasten abzubauen, bevor Sie den Nano wieder an den PC anschließen.

17. Tag

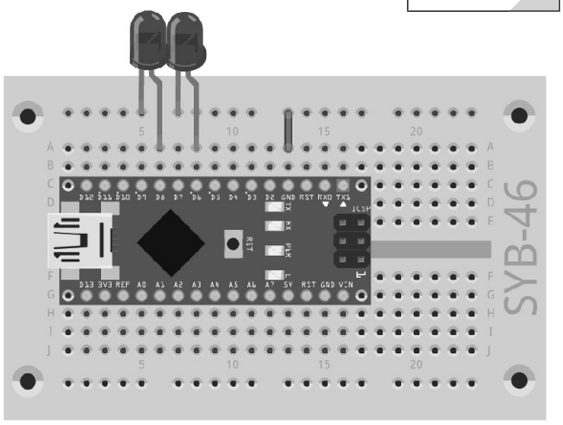
Heute im Adventskalender

· 1 LED blau mit eingebautem Vorwiderstand

Wechselblaulicht

Das erste Programm mit der Arduino-IDE lässt zwei LEDs abwechselnd blinken. Das Programm kann nach dem Flashen auch mit dem Batteriekasten ohne PC betrieben werden.

Bauteile: 1 Steckbrett, 2 LEDs blau mit Vorwiderstand, 1 Drahtbrücke



Lauflicht mit sieben LEDs und Potenziometer.

Das Programm

Da Dateinamen in der Arduino-IDE nicht mit einer Ziffer beginnen können, ist das folgende Programm mit `_17blinklicht01` etwas anders benannt.

```
int rot =   6;
int blau =  8;

void setup() {
  pinMode(rot, OUTPUT);
  pinMode(blau, OUTPUT);
}

void loop() {
  digitalWrite(rot, HIGH);
  digitalWrite(blau, LOW);
  delay(100);
  digitalWrite(rot, LOW);
  digitalWrite(blau, HIGH);
  delay(100);
}
```

So funktioniert das Programm

```
int rot =   6;
int blau =  8;
```

Am Anfang des Programms werden die beiden verwendeten Pinnummern in zwei Variablen vom Typ Integer gespeichert. Alle Programme in der Arduino-IDE bestehen aus mindestens zwei Funktionen:

```
void setup() {
...
}
```

Die Funktion `setup` läuft ein einziges Mal beim Start und wird meistens zur Einrichtung der Pins verwendet.

```
void loop() {
...
}
```

Die Funktion `loop` wird so lange wiederholt, bis man die Stromversorgung trennt oder den Reset-Knopf drückt.

```
void setup() {
  pinMode(rot, OUTPUT);
  pinMode(blau, OUTPUT);
}
```

In der Funktion `setup` werden die beiden verwendeten Pins als Ausgänge eingerichtet.

```
void loop() {
  digitalWrite(rot, HIGH);
  digitalWrite(blau, LOW);
  delay(100);
  digitalWrite(rot, LOW);
  digitalWrite(blau, HIGH);
  delay(100);
}
```

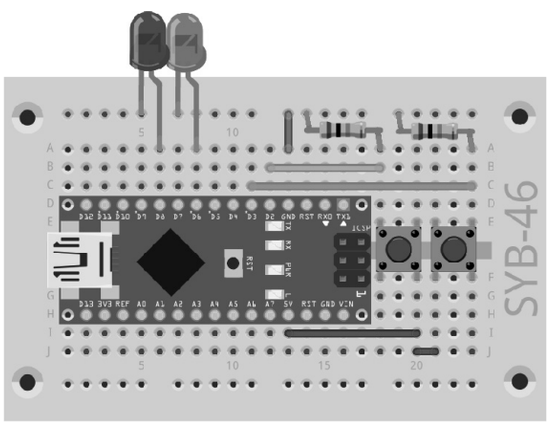
Die Funktion `loop` schaltet in jedem Durchlauf die rote LED ein und die blaue LED aus und wartet dann 100 Millisekunden. Danach wird die rote LED ausgeschaltet und die blaue ein. Nach einer weiteren Wartezeit von 100 Millisekunden startet die `loop`-Schleife erneut.

18. Tag

18. Tag

Heute im Adventskalender

- 1 Taster



Blinklicht mit zwei LEDs und Tastern.

Blinklicht mit Tasten steuern

Zwei LEDs blinken in unterschiedlichen Mustern, umschaltbar über zwei Taster. Die beiden Eingänge, an denen die Taster angeschlossen sind, sind über Pull-down-Widerstände mit Masse verbunden.

Bauteile: 1 Steckbrett, 1 LED rot mit Vorwiderstand, 1 LED blau mit Vorwiderstand, 2 Taster, 2 10-kOhm-Widerstände (rot-schwarz-orange), 5 Drahtbrücken (unterschiedliche Längen)

Das Programm

Das Programm `_18blinklicht02` bietet zwei unterschiedliche Blinkmuster – sie blinken entweder abwechselnd oder gleichzeitig –, die mit den beiden Tastern ausgewählt werden können.

```
int rot =   6;
int blau =  8;
int taste1 =  2;
int taste2 =  3;
int z = 200;
int m = 0;

void setup(){
  pinMode(rot, OUTPUT);
  pinMode(blau, OUTPUT);
  pinMode(taste1, INPUT);
  pinMode(taste2, INPUT);
}

void loop(){
  if (digitalRead(taste1) == HIGH){
    m = 1;
  }
  if (digitalRead(taste2) == HIGH){
    m = 2;
  }
  if (m == 1){
    digitalWrite(rot, HIGH);
    digitalWrite(blau, LOW);
    delay(z);
    digitalWrite(rot, LOW);
    digitalWrite(blau, HIGH);
    delay(z);
  }
  if (m == 2){
    digitalWrite(rot, HIGH);
    digitalWrite(blau, HIGH);
    delay(z);
    digitalWrite(rot, LOW);
    digitalWrite(blau, LOW);
    delay(z);
  }
]
```

So funktioniert das Programm

```
int rot =   6;
int blau =  8;
int taste1 =  2;
int taste2 =  3;
```

Am Anfang werden die Pinnummern für die LEDs und Taster definiert.

```
int z = 200;
int m = 0;
```

Die Variable `z` bezeichnet die Leuchtzeit der LEDs beim Blinken, die Variable `m` gibt den Blinkmodus an, der später über die Taster umgeschaltet wird. Im Modus `0` blinkt am Anfang keine der LEDs.

```
void setup(){
  pinMode(rot, OUTPUT);
  pinMode(blau, OUTPUT);
  pinMode(taste1, INPUT);
  pinMode(taste2, INPUT);
}
```

Die `setup`-Funktion richtet zwei Ausgänge für die LEDs und zwei Eingänge für die Taster ein.

```
  if (digitalRead(taste1) == HIGH){
    m = 1;
  }
  if (digitalRead(taste2) == HIGH){
    m = 2;
  }
```

Wird Taster 1 gedrückt, wird der Blinkmodus auf 1 gesetzt, mit Drücken von Taster 2 wird der Blinkmodus auf 2 gesetzt.

```
  if (m == 1){
    digitalWrite(rot, HIGH);
    digitalWrite(blau, LOW);
    delay(z);
    digitalWrite(rot, LOW);
    digitalWrite(blau, HIGH);
    delay(z);
  }
```

Ist der Blinkmodus 1, wird die rote LED ein- und die blaue ausgeschaltet. Nach einer Wartezeit wird die rote LED aus- und die blaue eingeschaltet. Die LEDs blinken abwechselnd.

```
  if (m == 2){
    digitalWrite(rot, HIGH);
    digitalWrite(blau, HIGH);
    delay(z);
    digitalWrite(rot, LOW);
    digitalWrite(blau, LOW);
    delay(z);
  }
```

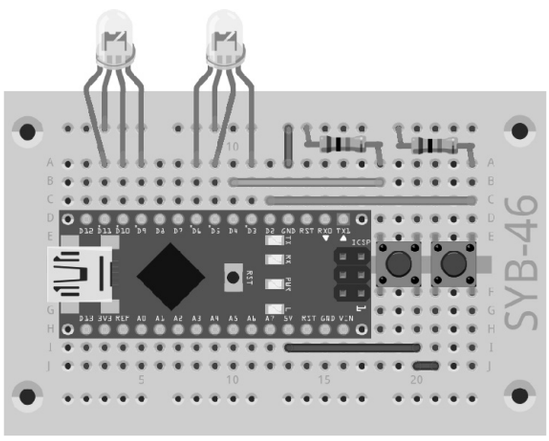
Ist der Blinkmodus 2, werden beide LEDs eingeschaltet. Nach einer Wartezeit werden beide ausgeschaltet. Die LEDs blinken gleichzeitig.

19. Tag

19. Tag

Heute im Adventskalender

- 1 RGB-LED mit eingebauten Vorwiderständen



Zwei RGB-LEDs, über Taster gesteuert.

RGB-Farbspiele

Die RGB-LEDs verwenden PWM-Signale für die Farbeffekte. Dazu müssen PWM-Pins des Nano verwendet werden, weshalb die RGB-LEDs auf dem Steckbrett etwas ungewöhnlich aufgebaut sind.

Bauteile: 1 Steckbrett, 2 RGB-LEDs, 2 Taster, 2 10-kOhm-Widerstände (rot-schwarz-orange), 5 Drahtbrücken (unterschiedliche Längen)

Das Programm

Mit dem Programm `_19rgb03` verändern beide RGB-LEDs zyklisch ihre Farben. Durch unterschiedliche Anfangsfarben und Schrittweiten beim Farbwechsel ergeben sich verschiedene Farben auf beiden LEDs. Mit den Tastern lassen sich die Farbwechsel der beiden RGB-LEDs unabhängig voneinander anhalten und wieder fortsetzen.

```
int r1 = 11;
int g1 = 10;
int b1 = 9;
int r2 = 6;
int g2 = 5;
int b2 = 3;
int taste1 = 4;
int taste2 = 2;
int z = 20;
int m1 = 0;
int m2 = 0;
int r1h = 0;
int g1h = 100;
int b1h = 200;
int r2h = 0;
int g2h = 100;
int b2h = 50;
int r1s = 5;
int g1s = 5;
int b1s = 5;
int r2s = 10;
int g2s = 10;
int b2s = 10;

void setup() {
  pinMode(r1, OUTPUT);
  pinMode(g1, OUTPUT);
  pinMode(b1, OUTPUT);
  pinMode(r2, OUTPUT);
  pinMode(g2, OUTPUT);
  pinMode(b2, OUTPUT);
  pinMode(taste1, INPUT);
  pinMode(taste2, INPUT);
}

void loop() {
  if (digitalRead(taste1) == HIGH) {
    m1 = 1-m1;
  }
  if (digitalRead(taste2) == HIGH) {
    m2 = 1-m2;
  }
  if (m1 == 1) {
    analogWrite(r1, r1h);
    r1h += r1s;
    if (r1h <= 0 || r1h >= 255) {
      r1s = -r1s;
    }
    analogWrite(g1, g1h);
    g1h += g1s;
    if (g1h <= 0 || g1h >= 255) {
```

```
      g1s = -g1s;
    }
    analogWrite(b1, b1h);
    b1h += b1s;
    if (b1h <= 0 || b1h >= 255) {
      b1s = -b1s;
    }
  }
  if (m2 == 1) {
    analogWrite(r2, r2h);
    r2h += r2s;
    if (r2h <= 0 || r2h >= 255) {
      r2s = -r2s;
    }
    analogWrite(g2, g2h);
    g2h += g2s;
    if (g2h <= 0 || g2h >= 255) {
      g2s = -g2s;
    }
    analogWrite(b2, b2h);
    b2h += b2s;
    if (b2h <= 0 || b2h >= 255) {
      b2s = -b2s;
    }
  }
  delay(z);
}
```

So funktioniert das Programm

Am Anfang des Programms werden die Pinnummern in Variablen gespeichert. `r1`, `g1`, `b1` sind die Pins der ersten RGB-LED, `r2`, `g2`, `b2` die der zweiten. `taste1` und `taste2` sind die Pins der beiden Taster.

`z` gibt die Zeit und damit die Geschwindigkeit der Farbwechsel an, `m1` und `m2` schalten das Farbwechselmuster der ersten und zweiten RGB-LED aus. Stehen diese Variablen auf 1, wechseln die Farben, bei 0 bleibt die aktuelle Farbe angezeigt.

`r1h`, `g1h`, `b1h` geben die PWM-Helligkeitswerte der drei Farben der ersten RGB-LED an, `r2h`, `g2h`, `b2h` die der zweiten. Beide LEDs starten mit unterschiedlichen Farben. Sie können auch ganz andere Farben ausprobieren.

`r1s`, `g1s`, `b1s` geben die Schrittweiten für die zyklischen Farbwechsel der ersten RGB-LED an, `r2s`, `g2s`, `b2s` die der zweiten. Beide LEDs wechseln ihre Farben unterschiedlich schnell.

Die `setup`-Funktion richtet sechs Ausgänge für die beiden RGB-LEDs und zwei Eingänge für die Taster ein.

```
  if (digitalRead(taste1) == HIGH) {
    m1 = 1-m1;
  }
```

Die `loop`-Funktion fragt als Erstes die beiden Taster ab. Wird der erste Taster gedrückt, wird der Modus der ersten RGB-LED `m1` von 0 auf 1 gesetzt, beim nächsten Drücken des Tasters wieder zurück von 1 auf 0. Der zweite Taster schaltet nach der gleichen Formel die Variable `m2` um.

```
  if (m1 == 1) {
    analogWrite(r1, r1h);
    r1h += r1s;
```

Steht der Modus der ersten RGB-LED auf 1, wechselt sie ihre Farbe. Dabei leuchtet die rote Farbe `r1` zunächst in der aktuell eingestellten Helligkeit `r1h`. Danach wird die Helligkeit um die Schrittweite `r1s` verändert.

```
    if (r1h <= 0 || r1h >= 255) {
      r1s = -r1s;
    }
```

Erreicht die Helligkeit durch diese Veränderung die Grenzwerte 0 oder 255, wird die Schrittweite umgekehrt. Das Zeichen `||` steht in der Arduino-Programmiersprache für „oder“. Auf die gleiche Weise werden innerhalb dieser `if`-Abfrage auch die grüne und die blaue Farbe der RGB-LED um jeweils einen Schritt verändert. Eine zweite `if`-Abfrage überprüft, ob die Variable `m2` auf 1 steht, und lässt in diesem Fall nach dem gleichen Schema die zweite RGB-LED die Farbe wechseln.

```
  delay(z);
```

Ganz zum Schluss folgt eine kurze Pause, bis die Hauptschleife einen neuen Durchlauf startet. Dies ist die einzige Zeitverzögerung im ganzen Programm und bestimmt die Geschwindigkeit der Farbwechsel.

20. Tag

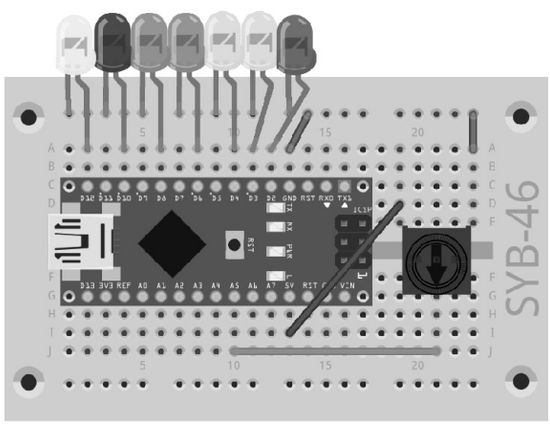
Heute im Adventskalender

- 1 LED weiß mit eingebautem Vorwiderstand

Analoge Pegelanzeige mit LEDs

Auf einer Pegelanzeige lassen sich analoge Werte auf einen Blick ablesen. Solche Anzeigen aus mehreren LEDs werden zum Beispiel bei Lautstärke- oder Temperaturreglern verwendet. Das Experiment des 20. Tags zeigt den auf dem Potenziometer eingestellten Wert über eine Reihe von LEDs an.

Bauteile: 1 Steckbrett, 1 LED rot mit Vorwiderstand, 2 LEDs gelb mit Vorwiderstand, 2 LEDs grün mit Vorwiderstand, 1 LED blau mit Vorwiderstand, 1 LED weiß mit Vorwiderstand, 1 15-kOhm-Potenziometer, 4 Drahtbrücken (unterschiedliche Längen)



Pegelanzeige mit sieben LEDs und Potenziometer.

Das Programm

Das Programm `_20pegel01` liest den eingestellten Wert des Potenziometers am analogen Eingang A5 aus und zeigt ihn mithilfe einer Schleife auf sieben LEDs an.

```
int sensor = A5;
int n = 7;
int leds[] = {2, 3, 4, 6, 8, 10, 12};

void setup() {
  for (int i = 0; i < n; i++) {
    pinMode(leds[i], OUTPUT);
  }
}

void loop() {
  int s = analogRead(sensor);
  int p = map(s, 0, 1023, 0, n);
  for (int i = 0; i < n; i++) {
    if (i < p) {
      digitalWrite(leds[i], HIGH);
    }
    else {
      digitalWrite(leds[i], LOW);
    }
  }
}
```

So funktioniert das Programm

Die Variable `sensor` enthält die Pinnummer des analogen Eingangs A5, die Variable `n` gibt die Anzahl der LEDs an.

```
int leds[] = {2, 3, 4, 6, 8, 10, 12};
```

Die Pinnummern der LEDs werden ähnlich wie bereits in einigen anderen Snap4Arduino-Programmen in einer Liste `leds[]` gespeichert. Jede LED lässt sich über einen Index in der Liste ansteuern. `led[0]` bezeichnet die erste LED an Pin 2, `led[6]` bezeichnet die letzte LED an Pin 12.

```
void setup() {
  for (int i = 0; i < n; i++) {
    pinMode(leds[i], OUTPUT);
  }
}
```

Die `setup`-Funktion definiert über eine Schleife die sieben LED-Pins als Ausgänge. Dafür verwenden wir eine `for`-Schleife, die in C wesentlich flexibler ist als in anderen Programmiersprachen. Jede `for`-Schleife besteht hier aus drei Parametern: einer Startanweisung, einer Bedingung, die erfüllt sein muss, damit die Schleife läuft, und einer Anweisung, die bei jedem Schleifendurchlauf einmal abgearbeitet wird. Es muss also nicht, wie in anderen Programmiersprachen, einfach nur ein Zähler hochgezählt werden.

```
  int s = analogRead(sensor);
```

In der Hauptschleife `loop` wird zuerst der analoge Wert des Sensors ausgelesen und in der Variablen `s` gespeichert. Dieser Wert kann eine Ganzzahl zwischen 0 und 1023 sein.

```
  int p = map(s, 0, 1023, 0, n);
```

Die Funktion `map()` setzt diesen Wert in einen Wert zwischen `0` und `n`, der Anzahl der LEDs, um. Die Funktion verwendet fünf Parameter. In diesem Fall sind das:

- `s` – der umzurechnende Wert
- `0` – die untere Grenze des Zahlenbereichs des eingegebenen Werts
- `1023` – die obere Grenze des Zahlenbereichs des eingegebenen Werts
- `0` – die untere Grenze des Zahlenbereichs des ausgegebenen Werts
- `n` – die obere Grenze des Zahlenbereichs des ausgegebenen Werts

Ist `s = 0`, wird `p = 0` zurückgegeben. Ist `s= 1023`, wird `p= 6` zurückgegeben. Bei allen Werten dazwischen gibt die Funktion entsprechende Zwischenwerte zurück. Man spart sich damit die manuelle Umrechnung zwischen verschiedenen Zahlenskalen.

```
  for (int i = 0; i < n; i++) {
```

Dann startet wieder eine Schleife, die über alle LEDs von unten nach oben hochzählt. Bei jeder LED wird geprüft, ob die Nummer der LED `i` kleiner ist als der in `p` gespeicherte Pegelwert.

```
    if (i < p) {
      digitalWrite(leds[i], HIGH);
    }
```

Ist das der Fall, wird die entsprechende LED eingeschaltet.

```
    else {
      digitalWrite(leds[i], LOW);
    }
```

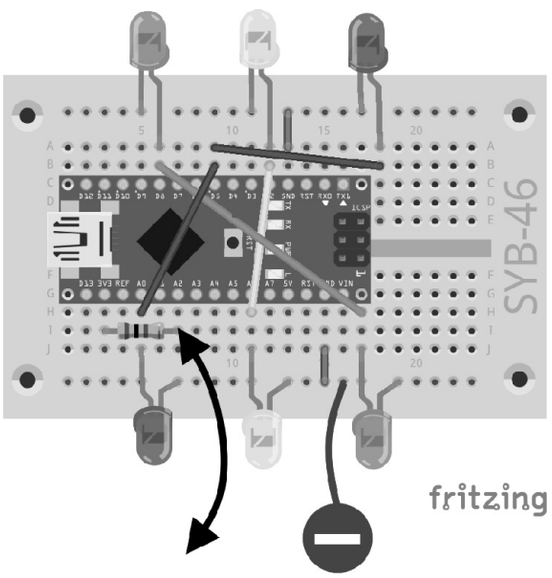
Ist der anzuzeigende Wert nicht größer als die Nummer der LED in der Schleife, wird die LED ausgeschaltet. Die Bedingung `else` in einer Abfrage trifft immer dann zu, wenn die Bedingung `if` nicht wahr ist.

21. Tag

21. Tag

Heute im Adventskalender

- 1 Krokokabel



LED-Würfel mit sieben LEDs mithilfe eines Knetesensors schalten. Der Schaltungsaufbau entspricht dem des 14. Tags.

Mit dem Krokokabel lassen sich auch andere leitfähige Gegenstände als Knete, wie zum Beispiel Löffel oder Münzen, als Sensorkontakte nutzen.

LED-Würfel mit realistischem Würfeleffekt

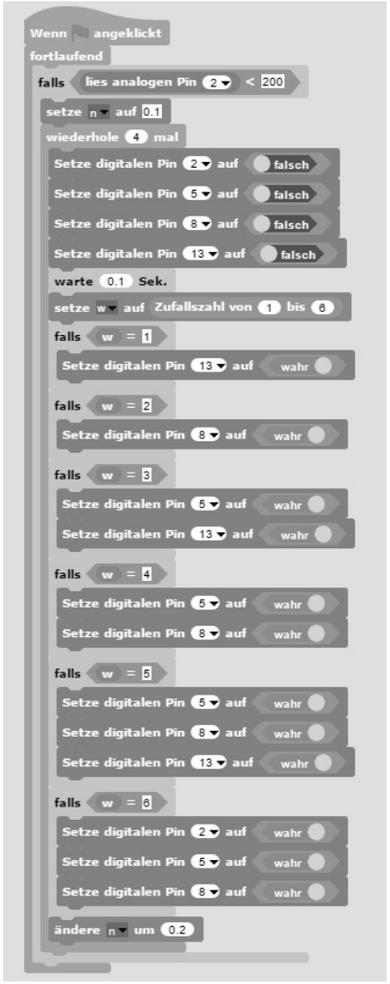
Ein wirklicher Würfel zeigt nicht sofort die endgültige Augenzahl an, sondern rollt erst noch eine kurze Zeit, in der man Würfelergebnisse sieht, die sich dann aber doch nicht bewahrheiten. Das Programm des 21. Tags simuliert das Rollen, indem der Würfel erst ein paar andere Würfelergebnisse mit minimal immer länger werdenden Pausen dazwischen anzeigt, bevor das endgültige Ergebnis erscheint.

Bauteile: 1 Steckbrett, 2 LEDs rot mit Vorwiderstand, 2 LEDs gelb mit Vorwiderstand, 2 LEDs grün mit Vorwiderstand, 1 20-MOhm-Widerstand (rot-schwarz-blau), 2 Drahtbrücken, 2 Knetekontakte

Das Krokokabel ist direkt an den Anschluss des 20-MOhm-Widerstands angeklemmt, der mit Pin A2 verbunden ist.

StandardFirmata flashen

Nachdem Sie Programme aus der Arduino-IDE auf den Nano geflasht haben, müssen Sie erst wieder die StandardFirmata flashen, bevor Sie Snap4Arduino erneut verwenden können. Beachten Sie dazu die Hinweise am 16. Tag.



Das Programm `21wuerfel02` würfelt mit realistischem Würfeleffekt.

Das Programm

Das Programm `21wuerfel02` nutzt die gleiche Würfelroutine wie das Programm des 14. Tags, würfelt aber nach dem Berühren des Knetekontakts nicht nur einmal, sondern viermal hintereinander.

So funktioniert das Programm

Die Hauptschleife des Programms läuft viermal nacheinander, wobei am Ende die Wartezeiten, bevor das Ergebnis im nächsten Durchlauf der Schleife gelöscht wird, jedes Mal um 0,2 Sekunden länger werden. Dadurch scheint der Würfel langsam auszurollen.

22. Tag

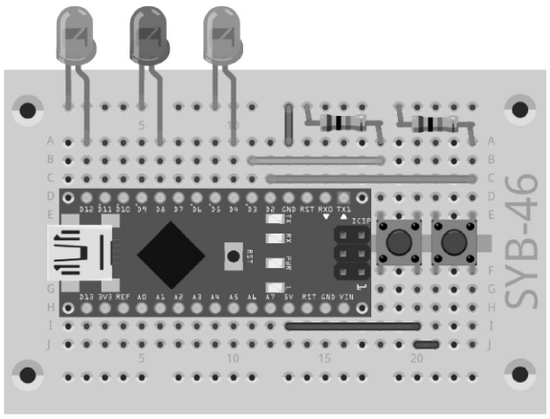
Heute im Adventskalender

· 1 LED orange mit eingebautem Vorwiderstand

Der Spieleklassiker Pong

Heute kommt zur Abwechslung mal wieder ein Spiel, das auf dem Bildschirm des PCs läuft. Zwei Tasten und drei LEDs bilden ein Gamepad zur Steuerung des Spiels.

Bauteile: 1 Steckbrett, 1 LED orange mit Vorwiderstand, 2 LEDs grün mit Vorwiderstand, 2 Taster, 2 10-kOhm-Widerstände (rot-schwarz-orange), 5 Drahtbrücken (unterschiedliche Längen)



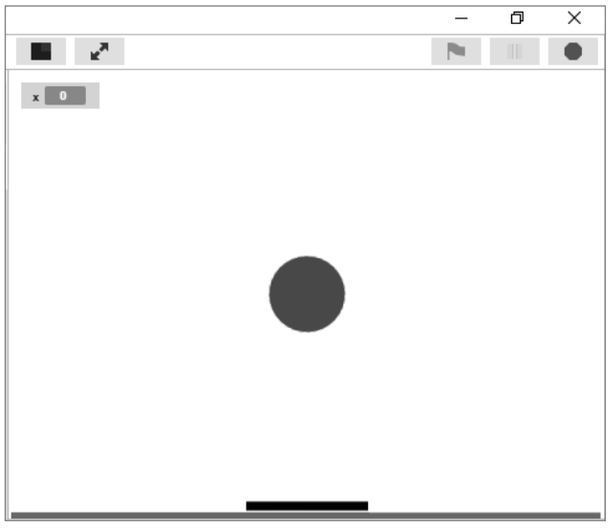
Das Gamepad zur Steuerung des Spiels.

Das Programm

Mit etwas Geschicklichkeit versucht man, mit dem Paddle den Ball immer wieder zurückzuschlagen, damit er nicht gegen die rote Kante am unteren Rand des Spielfelds fliegt. Der Spieler hat zwei Tasten, um das Paddle in die beiden Richtungen zu bewegen.

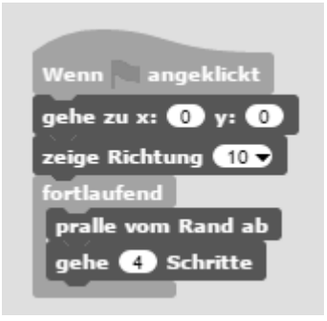
So funktioniert das Programm

Das Programm `22pong` besteht aus zwei Objekten, dem Ball und dem Paddle, die mit dem integrierten Malprogramm gezeichnet werden können. Jedes Objekt in Snap4Arduino verwendet eigene Blöcke, die im Skriptbereich erscheinen, wenn man unten rechts auf der Objektpalette auf das entsprechende Objekt klickt.



Das Spielfeld mit dem Ball und dem Paddle.

Der Ball wird durch drei Skriptblöcke gesteuert, die alle gleichzeitig laufen und gestartet werden, wenn der Benutzer auf das grüne Fähnchen klickt.



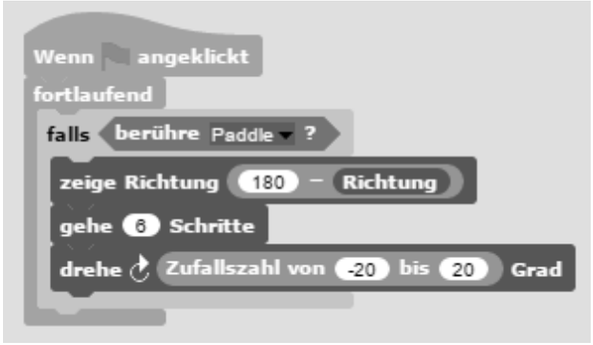
Der Skriptblock für die Standardbewegung des Balls.

Der erste Block schafft beim Klick auf das grüne Fähnchen die Grundvoraussetzungen für das Spiel. Zuerst wird der Ball auf seine Ausgangsposition bei **x:0 y:0** gebracht. Der Ball soll im Winkel von **10** Grad schräg nach oben losfliegen. Dazu wird die Richtung auf **10** gesetzt.

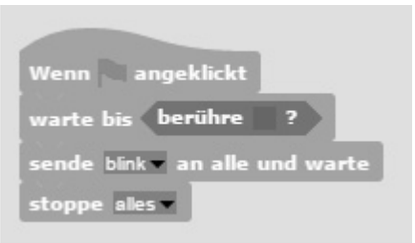
Anschließend wird die Bewegung des Balls fortlaufend wiederholt. Er prallt vom Rand ab, sollte er ihn berühren. Andernfalls fliegt er vier Schritte in die eingestellte Richtung. Diese Bewegung wiederholt sich theoretisch endlos. Da aber beim Klicken auf das grüne Fähnchen weitere Skriptblöcke für den Ball gestartet werden, kann es auch zu anderen Bewegungen kommen.

Wenn der Ball das Paddle berührt, wird die Bewegungsrichtung ins Negative umgekehrt. Um die Bewegung etwas unvorhersehbarer zu gestalten, wird der Ball zunächst sechs Schritte bewegt, damit danach das Paddle auf jeden Fall nicht mehr berührt wird. Anschließend wird die Flugrichtung gegenüber der bisherigen Richtung um einen zufälligen Wert zwischen **–20** und **20** Grad verändert.

Berührt der Ball nicht das Paddle, sondern den roten Balken am unteren Rand, ist das Spiel zu Ende. Davor blinkt die rote LED noch, um den Fehler des Spielers anzuzeigen.



Der Ball ändert seine Bewegungsrichtung, wenn er das Paddle berührt.



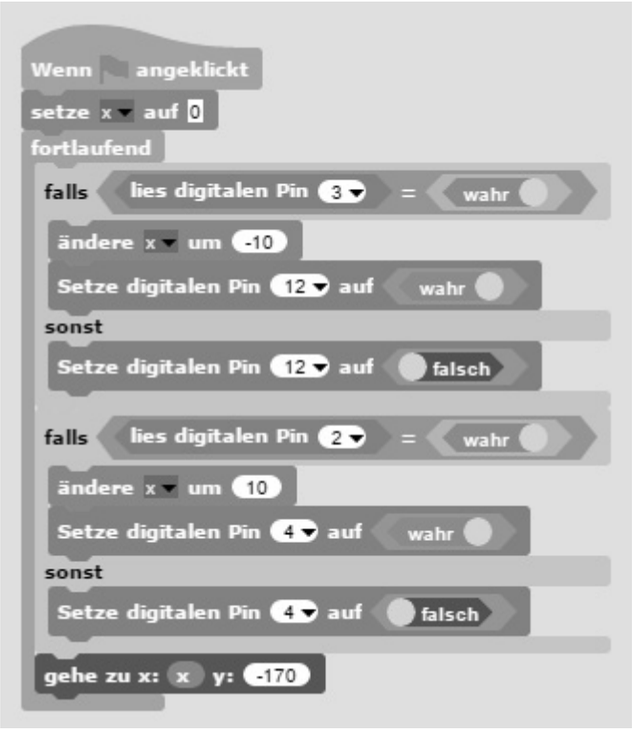
Dieser Skriptblock wird ausgeführt, wenn der Ball den unteren roten Rand des Spielfelds berührt.

Skriptblöcke für das Paddle

In Snap4Arduino verwendet jedes Objekt eigene Blöcke. Klicken Sie in der Objektpalette unten rechts auf das Paddle, um dessen Blöcke zu sehen.

Arduino mit Paddle verbinden
In Snap4Arduino kann nur ein Objekt mit dem Arduino verbunden sein. In diesem Programm kommuniziert das Paddle mit dem Nano. Klicken Sie also in der Ansicht des Paddles auf der Blockpalette **Arduino** auf **Mit Arduino verbinden**.

Beim Drücken der Taster soll sich das Paddle nach links oder rechts bewegen.



Taster auswerten und Paddle bewegen.

Beim Klick auf das grüne Fähnchen wird die Variable **x** auf 0 gesetzt, um das Paddle in die Ausgangsposition zu bringen. Danach wartet eine **fortlaufend**-Schleife darauf, dass einer der Taster gedrückt wird. In diesem Fall wird die Variable **x**, die die x-Position des Paddles angibt, um zehn Einheiten negativ oder positiv verändert. Außerdem wird bei gedrücktem Taster eine der grünen LEDs ein- und bei losgelassenem Taster ausgeschaltet. Nach den Abfragen wird das Paddle in jedem Schleifendurchlauf auf die x-Position gesetzt, die die Variable **x** angibt.

Da es immer nur möglich ist, dass ein Objekt mit dem Arduino verbunden ist, kann der Ball beim Berühren der roten Linie nicht selbst die rote LED blinken lassen. Der Ball sendet deshalb mit dem Block **Sende ... an alle und warte** eine Nachricht, die das Paddle empfängt und daraufhin die LEDs blinken lässt.

Dieser Skriptblock lässt am Spielende die rote LED blinken.

Empfängt das Paddle die Nachricht **blink**, lässt eine Schleife die rote LED an Pin 8 fünfmal kurz aufblinken.

23. Tag

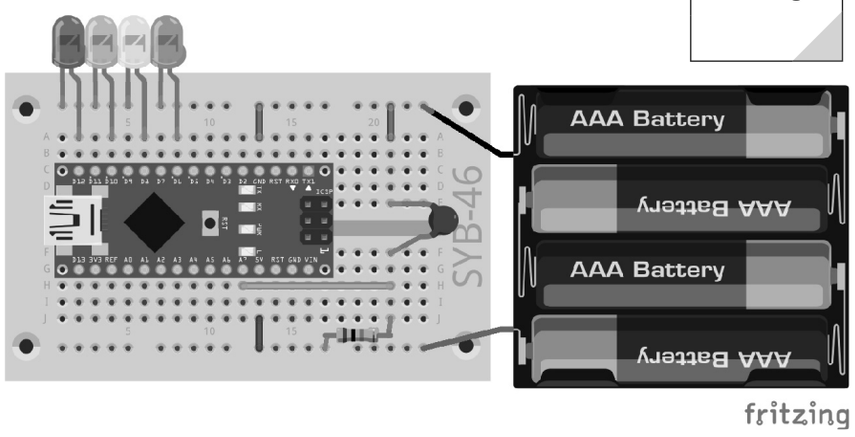
Heute im Adventskalender

- 1 NTC (temperaturabhängiger Widerstand)

Feuermelder für Adventskerzen

Das Experiment des 23. Tags ist ein batteriebetriebener Wärmemelder. Kommt der hitzeempfindliche NTC einer Kerze zu nahe, blinkt die rote LED. Geringere Temperaturen werden in verschiedenen Farben angezeigt.

Bauteile: 1 Steckbrett, 1 LED rot mit Vorwiderstand, 1 LED orange mit Vorwiderstand, 1 LED gelb mit Vorwiderstand, 1 LED grün mit Vorwiderstand, 1 NTC, 1 10-kOhm-Widerstand (rot-schwarz-orange), 1 Batteriekasten, 4 Drahtbrücken (unterschiedliche Längen)



Wärmemelder mit NTC und vier LEDs.

Das Programm

Das Programm `_23kerze01` ist in der Arduino-IDE geschrieben, damit der Wärmemelder unabhängig vom PC batteriebetrieben genutzt werden kann.

```
int sensor = A7;
int rot =    12;
int orange = 10;
int gelb =    8;
int gruen =   6;

void setup() {
  pinMode(rot, OUTPUT);
  pinMode(orange, OUTPUT);
  pinMode(gelb, OUTPUT);
  pinMode(gruen, OUTPUT);
}

void loop() {
  int s = analogRead(sensor);
  if (s < 500) {
    digitalWrite(gruen, HIGH);
  }
  else {
    digitalWrite(gruen, LOW);
  }
  if (s < 400) {
    digitalWrite(gelb, HIGH);
  }
  else {
    digitalWrite(gelb, LOW);
  }
  if (s < 300) {
    digitalWrite(orange, HIGH);
  }
  else {
    digitalWrite(orange, LOW);
  }
  if (s < 200) {
    for (int i = 0; i < 6; i++) {
      digitalWrite(rot, HIGH);
      delay(50);
      digitalWrite(rot, LOW);
      delay(50);
    }
  }
  else {
    digitalWrite(rot, LOW);
  }
}
```

So funktioniert das Programm

Der NTC liefert je nach Temperatur einen analogen Wert. Dafür wird ein analoger Eingang in der Variablen `sensor` gespeichert. Für die vier LEDs werden vier digitale Ausgänge definiert.

```
int s = analogRead(sensor);
```

Die `loop`-Schleife liest in jedem Durchlauf als Erstes den Wert des analogen Eingangs, an dem der NTC angeschlossen ist, aus. Abhängig von der Temperatur sollen verschiedene LEDs leuchten.

```
if (s < 500) {
  digitalWrite(gruen, HIGH);
}
else {
  digitalWrite(gruen, LOW);
}
```

Ist der vom NTC gelieferte analoge Wert kleiner als 500, leuchtet die grüne LED. Nach dem gleichen Prinzip leuchten die gelbe und die orangefarbene LED bei Werten kleiner als 400 oder kleiner als 300. Bei Bedarf können Sie diese Grenzwerte natürlich anpassen.

```
if (s < 200) {
  for (int i = 0; i < 6; i++) {
    digitalWrite(rot, HIGH);
    delay(50);
    digitalWrite(rot, LOW);
    delay(50);
  }
}
else {
  digitalWrite(rot, LOW);
}
```

Ist der vom NTC gelieferte Wert kleiner als 200, wird die rote LED nicht einfach eingeschaltet, sondern es wird eine kurze Blinksequenz ausgelöst, um vor der Hitze zu warnen.

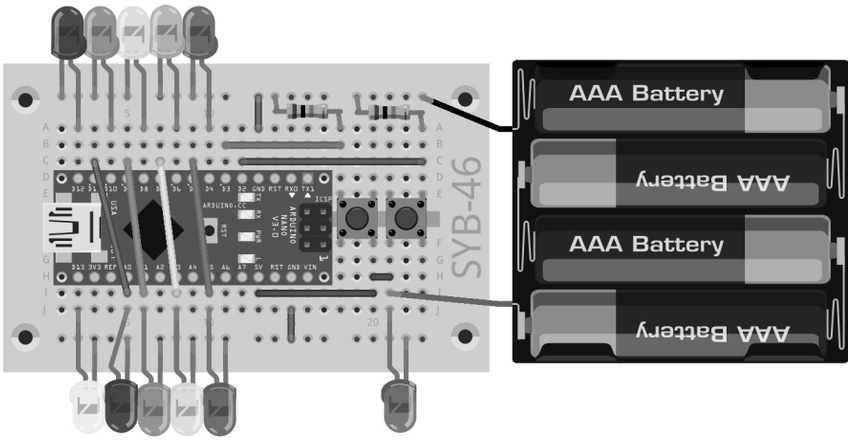
24. Tag

24. Tag

Heute im Adventskalender

· Blink-LED

Als Überraschung zu Weihnachten ist heute eine LED im Adventskalender, die selbstständig blinkt, ohne dass ein Programm nötig ist.



Zehn LEDs für den Weihnachtsstern.

Blinkender Weihnachtsstern

Zu Weihnachten gibt es einen blinkenden Weihnachtsstern, der an den Weihnachtsbaum gehängt werden kann. Mit den beiden Tastern lassen sich zwei unterschiedliche Blinkmuster einschalten. Die Blink-LED blinkt zusätzlich, unabhängig vom ausgewählten Blinkprogramm.

· Entfernen Sie die Schutzfolie von der Rückseite des Steckbretts. Jetzt können Sie den Batteriekasten dort aufkleben.

· Biegen Sie aus Schaltdraht zwei Schlingen, fädeln Sie sie durch die Ösen in zwei benachbarten Ecken des Steckbretts und verdrillen Sie die Enden. An diese Schlingen können Sie das Krokokabel klemmen und damit später die Schaltung an den Weihnachtsbaum hängen.

· Bauen Sie die Schaltung wie abgebildet auf.

· Schneiden Sie den Weihnachtsstern auf der Rückseite des Adventskalenders aus und stecken Sie ihn mit den länglichen Öffnungen auf die beiden LED-Reihen.

Bauteile: 1 Steckbrett, 1 Blink-LED mit Vorwiderstand, 2 LEDs rot mit Vorwiderstand, 1 LED orange mit Vorwiderstand, 2 LEDs gelb mit Vorwiderstand, 2 LEDs grün mit Vorwiderstand, 2 LEDs blau mit Vorwiderstand, 1 LED weiß mit Vorwiderstand, 2 Taster, 2 10-kOhm-Widerstände (rot-schwarz-orange), 1 Batteriekasten, 10 Drahtbrücken (unterschiedliche Längen)

Die LEDs der unteren Reihe außer Pin 13 sind mit Pins der oberen Pinleiste des Nano verbunden. Die Drahtbrücken verbinden sie zwar mit analogen Eingängen, die aber vom Programm nicht genutzt werden.

Die Blink-LED rechts unten enthält einen Vorwiderstand und kann daher direkt an 5V angeschlossen werden.

Das Programm

Das Programm `_24weihnachtsstern` lässt die LEDs auf dem Weihnachtsstern in zwei verschiedenen Mustern blinken.

```
int taste1 =  2;
int taste2 =  3;
int n = 10;
int leds[] = {4, 6, 8, 10, 12, 13, 11, 9, 7, 5};
int z = 50;
int m = 0;
int i;

void setup() {
  for (int i = 0; i < n; i++) {
    pinMode(leds[i], OUTPUT);
  }
  pinMode(taste1, INPUT);
  pinMode(taste2, INPUT);
}

void loop() {
  if (digitalRead(taste1) == HIGH) {
    m = 1;
```

```
  }
  if (digitalRead(taste2) == HIGH) {
    m = 2;
  }
  if (m == 1) {
    for (i = 0; i < n; i++) {
      digitalWrite(leds[i], HIGH);
      delay(z);
      digitalWrite(leds[i], LOW);
      delay(z);
    }
  }
  if (m == 2) {
    i = random(n);
      digitalWrite(leds[i], HIGH);
      delay(z);
    i = random(n);
      digitalWrite(leds[i], LOW);
      delay(z);
  }
}
```

So funktioniert das Programm

Die beiden Variablen `taste1` und `taste2` enthalten die Pinnummern der beiden Taster. Die Variable `n` gibt die Anzahl der LEDs an, deren Pinnummern in der Liste `leds[]` gespeichert sind. Die Variable `z` gibt die Verzögerungszeit beim Blinken an, die Variable `m` den Blinkmodus, und `i` wird nur als Integer definiert, um sie später als Schleifenzähler zu nutzen.

Die `setup`-Funktion richtet die Ausgänge für die LEDs und die Eingänge für die Taster ein.

```
  if (digitalRead(taste1) == HIGH) {
    m = 1;
  }
  if (digitalRead(taste2) == HIGH) {
    m = 2;
  }
```

Die `loop`-Funktion fragt als Erstes die beiden Taster ab und setzt den Blinkmodus je nach gedrücktem Taster.

```
  if (m == 1) {
    for (i = 0; i < n; i++) {
      digitalWrite(leds[i], HIGH);
      delay(z);
      digitalWrite(leds[i], LOW);
      delay(z);
    }
  }
```

Im Blinkmodus 1 leuchten die LEDs kurz nacheinander einzeln auf. Durch die Anordnung der LEDs und die Reihenfolge der Pinnummern in der Liste ergibt sich ein kreisförmiger Lauflichteffekt.

```
  if (m == 2) {
    i = random(n);
      digitalWrite(leds[i], HIGH);
      delay(z);
    i = random(n);
      digitalWrite(leds[i], LOW);
      delay(z);
  }
```

Im Blinkmodus 2 wird eine Zufallszahl zwischen 0 und 9 erzeugt und die entsprechende LED eingeschaltet. Anschließend wird in jedem Durchlauf eine zufällig gewählte LED ausgeschaltet. Da die Schaltzustände der LEDs nicht gespeichert und geprüft werden, können nach mehreren Durchläufen mehr oder weniger LEDs gleichzeitig leuchten.

Frohe Weihnachten!

Vorsichtsmaßnahmen
Auf keinen Fall irgendwelche Pins miteinander verbinden und abwarten, was passiert.
Nicht alle Pins lassen sich frei programmieren. Einige sind für die Stromversorgung und andere Zwecke fest eingerichtet.
Einige Pins sind direkt mit Anschlüssen des Mikrocontrollers verbunden, ein Kurzschluss kann den Arduino komplett zerstören. Verbindet man über einen Schalter oder eine LED zwei Pins miteinander, muss immer ein Schutzwiderstand dazwischengeschaltet werden.

Liebe Kunden!

CE

Dieses Produkt wurde in Übereinstimmung mit den geltenden europäischen Richtlinien hergestellt und trägt daher das CE-Zeichen. Der bestimmungsgemäße Gebrauch ist in der beiliegenden Anleitung beschrieben.

Bei jeder anderen Nutzung oder Veränderung des Produktes sind allein Sie für die Einhaltung der geltenden Regeln verantwortlich. Bauen Sie die Schaltungen deshalb genau so auf, wie es in der Anleitung beschrieben wird. Das Produkt darf nur zusammen mit dieser Anleitung weitergegeben werden.

Das Symbol der durchkreuzten Mülltonne bedeutet, dass dieses Produkt getrennt vom Hausmüll als Elektroschrott dem Recycling zugeführt werden muss. Wo Sie die nächstgelegene kostenlose Annahmestelle finden, sagt Ihnen Ihre kommunale Verwaltung.

Warnung! Augenschutz und LEDs:
Blicken Sie nicht aus geringer Entfernung direkt in eine LED, denn ein direkter Blick kann Netzhautschäden verursachen! Dies gilt besonders für helle LEDs im klaren Gehäuse sowie in besonderem Maße für Power-LEDs. Bei weißen, blauen, violetten und ultravioletten LEDs gibt die scheinbare Helligkeit einen falschen Eindruck von der tatsächlichen Gefahr für Ihre Augen. Besondere Vorsicht ist bei der Verwendung von Sammellinsen geboten. Betreiben Sie die LEDs so, wie in der Anleitung vorgesehen, nicht aber mit größeren Strömen.

Bibliografische Information der Deutschen Bibliothek
Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte Daten sind im Internet über http://dnb.ddb.de abrufbar.
Alle in diesem Buch vorgestellten Schaltungen und Programme wurden mit der größtmöglichen Sorgfalt entwickelt, geprüft und getestet. Trotzdem können Fehler im Buch und in der Software nicht vollständig ausgeschlossen werden. Verlag und Autor haften in Fällen des Vorsatzes oder der groben Fahrlässigkeit nach den gesetzlichen Bestimmungen. Im Übrigen haften Verlag und Autor nur nach dem Produkthaftungsgesetz wegen der Verletzung des Lebens, des Körpers oder der Gesundheit oder wegen der schuldhaften Verletzung wesentlicher Vertragspflichten. Der Schadensersatzanspruch für die Verletzung wesentlicher Vertragspflichten ist auf den vertragstypischen, vorhersehbaren Schaden begrenzt, soweit nicht ein Fall der zwingenden Haftung nach dem Produkthaftungsgesetz gegeben ist.

Arduino ist ein eingetragenes Markenzeichen der Arduino AG