

Adventskalender für Arduino 2021

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.

Dieser Adventskalender bringt jeden Tag ein neues Programmierprojekt für das eigene kleine Smart Home, das bis Weihnachten in vollem Glanz erstrahlen wird.

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. Nach König Arduino ist dort heute die Lieblingsbar der Arduino-Entwickler Massimo Banzi und David Cuartielles benannt.

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-kompatiblen Platinen sind erstaunlich stabil gegen Schaltungsfehler. Verbindet man über eine LED zwei Pins miteinander, muss immer ein Vorwiderstand dazwischengeschaltet werden, wenn in der LED nicht bereits einer eingebaut ist.

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

Tag 1

Heute im Adventskalender

- Nano-Board (Arduino-kompatibel)

Haus backen

Auf der Rückseite des Adventskalenders finden Sie die Ausschneidevorlagen für das Lebkuchenhaus. Hier können Sie schon erkennen, wie das Haus aussehen wird. Heute bereiten wir erst einmal den Teig vor.

Für den Teig brauchen Sie:

800 g	Mehl
400 g	Honig
150 g	Butter
200 g	Zucker
20 g	Kakao
2	Eier
1 Tüte	Lebkuchengewürz
1 Prise	Salz
4 TL	Backpulver

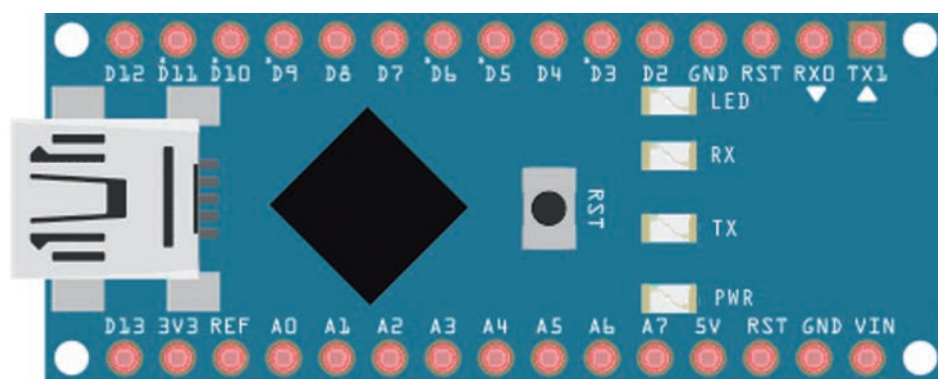
Schmelzen Sie die Butter in einem Topf auf kleiner Flamme. Sie darf nicht anbrennen. Rühren Sie dabei den Honig und den Zucker mit ein, sodass es eine gleichmäßige Masse ergibt.

Lassen Sie die Masse abkühlen und rühren Sie die Eier, das Lebkuchengewürz und das Salz hinein.

Mischen Sie das Mehl (lassen Sie ein bisschen übrig, um am nächsten Tag die Arbeitsfläche damit zu bestreuen), das Backpulver und den Kakao und geben Sie alles nach und nach unter Rühren zu der Masse. Dabei entsteht ein immer festerer Teig, den Sie zum Schluss mit der Hand kneten können. Kneten Sie so lange, bis alles gleichmäßig gemischt ist, und lassen Sie den Teig dann in einer mit einem Tuch abgedeckten Schüssel über Nacht ruhen.

Nano-Board – Arduino-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.



Die Anschlüsse am Nano

Die mit *D2...D12* beschrifteten Pins sind digitale Ein- oder Ausgänge, die die Logikwerte 0 oder 1 – oder anders geschrieben *Low* oder *High* annehmen können. Damit lassen sich zum Beispiel LEDs ein- und ausschalten.

Die mit *A0...A7* beschrifteten Pins sind analoge Eingänge. Dort kann eine Spannung zwischen 0V und 5V angelegt werden. Der Nano wertet sie aus und kann darauf mit bestimmten Aktionen antworten.

Was braucht man noch?

In unserem Adventskalender sind bereits alle elektronischen Bauteile enthalten, die Sie zum Aufbau der Experimente benötigen.

PC

Zur Programmierung des Nano verwenden wir in den Experimenten einen PC mit Windows. Die Arduino-IDE wird außer für Windows (ab Windows 7) auch für Linux, macOS und sogar für den Raspberry Pi angeboten, die anderen Tools teilweise nur für Windows. Die neue Arduino IDE 2.0 läuft nur auf Windows 10 (64 Bit), Linux (64 Bit) und macOS ab Version 10.14.

USB-Kabel

Die Verbindung zwischen PC und Nano erfolgt über ein Micro-USB-Kabel. Sie brauchen sich nicht extra ein solches Kabel zu besorgen, die meisten nicht mehr ganz aktuellen Smartphones verwenden diesen Steckertyp.

Netzteil (optional)

So lange der Nano am PC angeschlossen ist, wird er auch über die USB-Schnittstelle mit Strom versorgt. Nachdem ein Programm auf dem Nano gespeichert wurde, kann er in vielen Fällen auch ohne PC laufen. Dazu können Sie entweder ein USB-Steckernetzteil oder eine Powerbank anschließen. Ein Spannungsregler auf dem Nano regelt die Stromversorgung dann automatisch.

USB-Anschluss

Schließen Sie den Nano nach Möglichkeit an einen USB-2.0-Anschluss des PCs an, da es an USB-3.0-Anschlüssen eher zu Verbindungsproblemen kommen kann und es bei USB-3.0-Anschlüssen, bedingt durch die höhere Stromstärke, öfter zur Überhitzung des Nano kommt. Zur besseren Unterscheidung sind USB-3.0-Anschlüsse meistens blau.

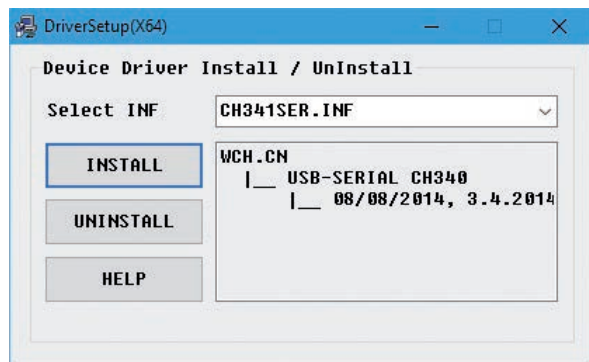
Nano ausschalten

Der Nano hat keinen Ausschalter, Sie brauchen einfach nur den Stecker ziehen, und er schaltet sich ab. Beim nächsten Einschalten startet automatisch der zuletzt gespeicherte Sketch. So werden auf der Arduino-Plattform die Programme genannt. Das gleiche passiert, wenn man den Reset-Taster drückt.

Softwareinstallation in Kürze

- 1 Laden Sie sich die Beispielprogramme und den Gerätetreiber bei www.franzis.de/adventskalender/adventskalender-fuer-arduino herunter.
- 2 Entpacken Sie das ZIP-Archiv in einen beliebigen Ordner auf der Festplatte.
- 3 Schließen Sie den Nano über das USB-Kabel an und starten Sie dann mit der Datei `CH341SER.EXE` aus dem Unterverzeichnis `Treiber` die Treiberinstallation. Zur Installation sind Administratorrechte erforderlich. Sie müssen 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. Sollte ein Fehler angezeigt werden, beenden Sie Ihren Virens Scanner, wenn Sie einen externen verwenden. Der Windows Defender hat kein Problem mit diesem Treiber.

Arduino-Adventskalender



Installation des Gerätetreibers.

- 5 Für die Programmierung Arduino-kompatibler Platinen liefert Arduino eine Entwicklungsumgebung (IDE), in der man die Programme, die bei Arduino als Sketch bezeichnet werden, in einer C-ähnlichen Programmiersprache schreiben kann. Laden Sie sich die Arduino IDE bei www.arduino.cc/en/software herunter und installieren Sie sie. Sie finden die Installationsdatei auch im Download zum Adventskalender. Unter Windows 10 können Sie die Arduino IDE auch aus dem Windows Store herunterladen und installieren. Allerdings finden Sie dort meistens nicht die neueste Version.

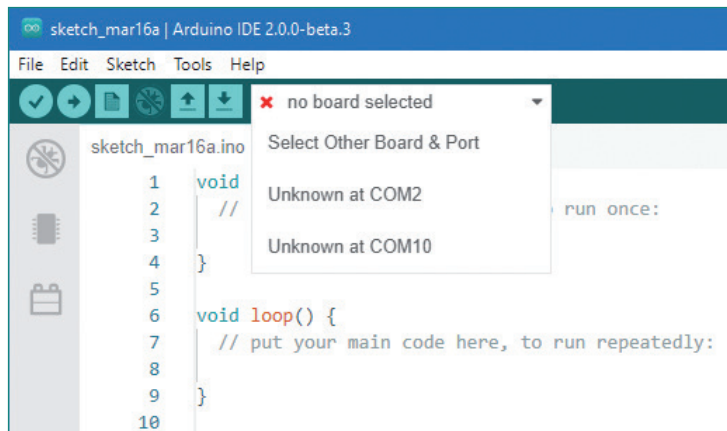
Erster Start der Arduino IDE und Einrichten des Nano-Boards

- 1 Beim ersten Start der Arduino IDE erscheint eine Meldung der Windows-Firewall. Der Zugriff muss zugelassen werden, da sonst keine Verbindung zum angeschlossenen Nano-Board möglich ist. Externe Firewalls sollten bei Problemen komplett beendet werden, da die Arduino IDE in vielen Fällen keine Verbindung herstellen kann, selbst wenn sie in der Firewall als Ausnahme eingetragen ist.



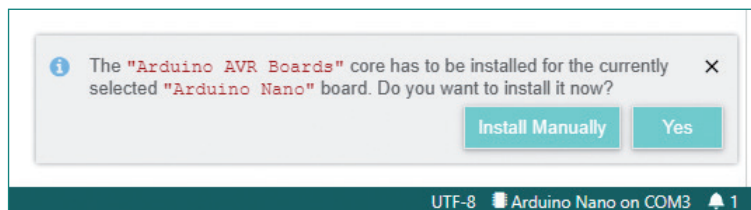
Arduino IDE in der Firewall freigeben

- 2 Klicken Sie oben auf das Listenfeld *no board selected*. Auf den meisten PCs werden dort ein oder zwei Zeilen *Unknown at COMx* angezeigt. Üblicherweise belegt der Nano-Treiber die COM-Schnittstelle mit der höchsten Nummer. Wählen Sie diese aus.



COM-Schnittstelle wählen

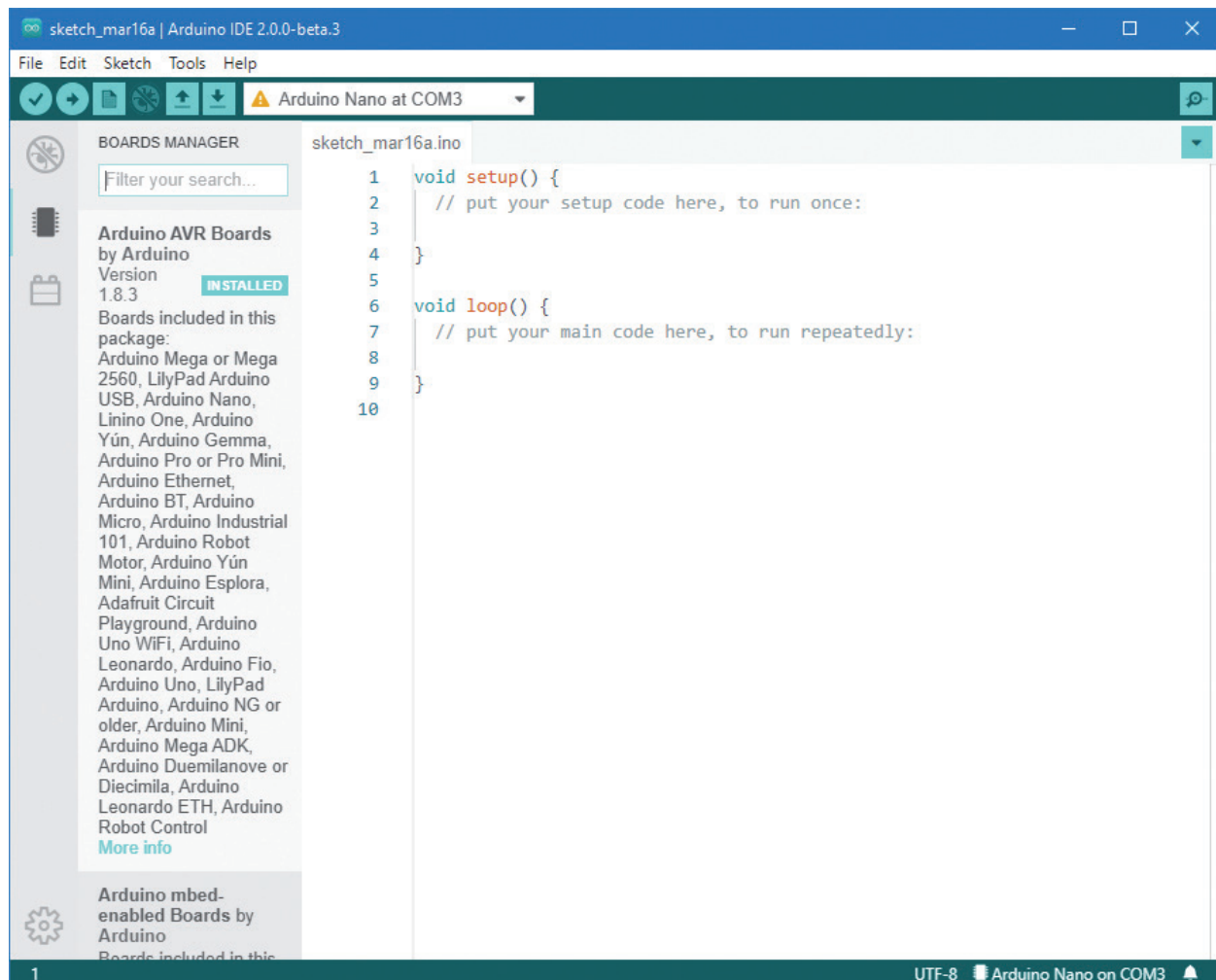
- 3 Wählen Sie im nächsten Dialogfeld in der Liste *Boards* den *Arduino Nano*. Wundern Sie sich nicht, dass im Bereich *Ports* die Meldung *No Ports discovered* auftaucht. Sie haben den Anschluss ja bereits gewählt.
- 4 Beim ersten Auswählen eines Arduino-kompatiblen Boards muss noch ein passendes Zusatzmodul installiert werden. Wenn die Meldung zur Installation des *Arduino AVR Boards core* erscheint, klicken Sie auf *Yes*.




Arduino AVR Boards core installieren

- 5 Während der Installation erscheint der *Boards Manager* in der Arduino IDE. Warten Sie bis das grüne Symbol *INSTALLED* erscheint.

Arduino-Adventskalender



Boards Manager in der Arduino IDE

- 6  Nachdem *Arduino AVR Boards core* installiert ist, können Sie den Boards Manager mit dem Symbol in der linken Symbolleiste wieder schließen.
- 7 Prüfen Sie noch, ob im Menü *Tools/Processor* die Option *ATmega 328P (Old Bootloader)* ausgewählt ist. Ist das nicht der Fall, wählen Sie diese Option aus.
- 8 Jetzt erscheint der Nano in der Auswahlliste am oberen Rand der Arduino IDE. Wenn Sie diese an einem anderen Tag starten und der Nano angeschlossen ist, können Sie ihn direkt dort auswählen.
- 9 Die Programme der einzelnen Tage finden Sie Arduino-typisch in einzelnen Ordnern im Download. Kopieren Sie diese Ordner in den Ordner *Dokumente\Arduino* in Ihrem Windows-Benutzerprofil oder in *OneDrive\Dokumente\Arduino*, wenn die automatische Synchronisation mit OneDrive eingerichtet ist. Dieser Ordner wird beim ersten Start der Arduino IDE angelegt. Ist er nicht vorhanden, finden Sie den genauen Pfad in der Arduino IDE über den Menüpunkt *File/Preferences* im Feld *Sketchbook location*.

LED blinkt

In einem ganz einfachen Programm soll die auf dem Nano eingebaute LED schnell blinken.

Bauteile: Nano-Board

```
//01blink
//Franzis Adventskalender für Arduino

void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {
  digitalWrite(LED_BUILTIN, HIGH);
  delay(50);
  digitalWrite(LED_BUILTIN, LOW);
  delay(50);
}
```

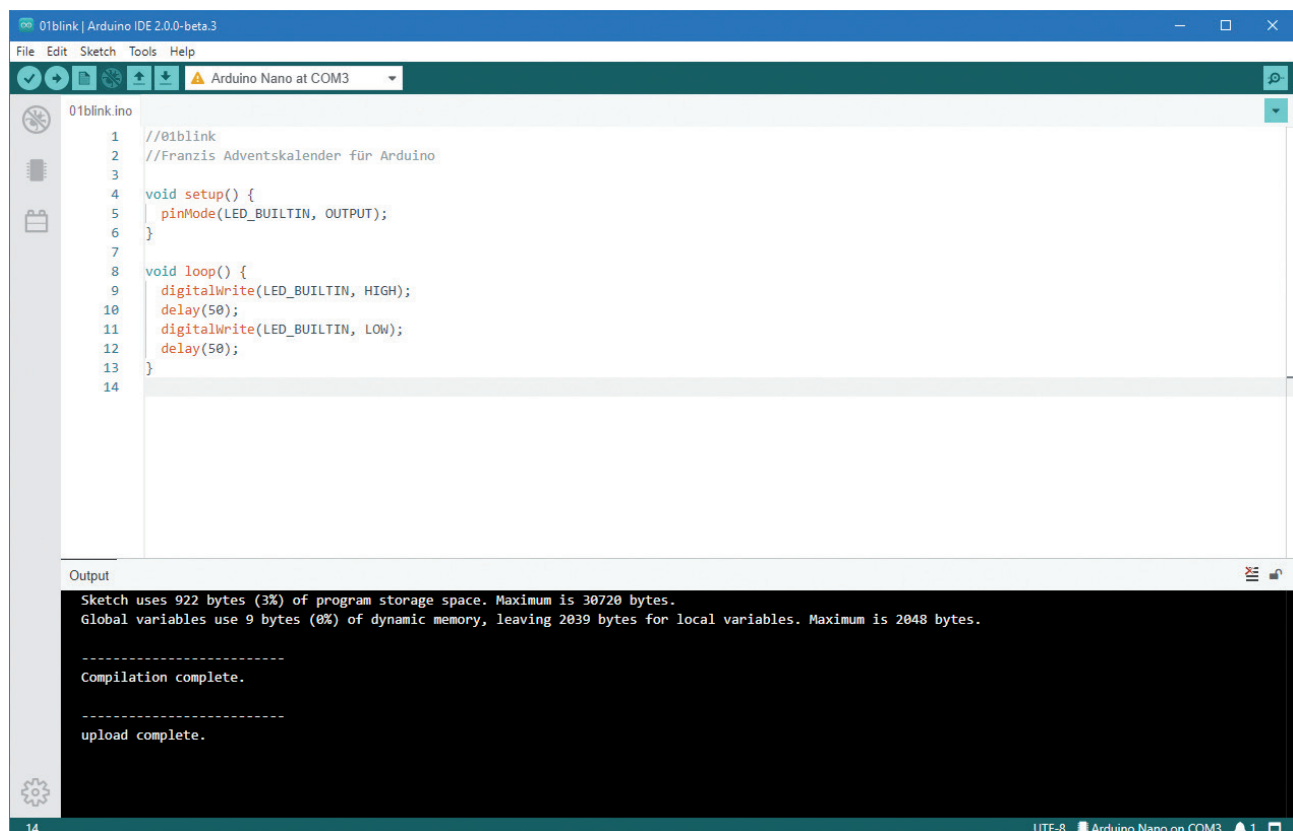
Die ersten beiden Zeilen, die mit // beginnen, sind nur Kommentare und haben für die Funktion des Programms keine Bedeutung. Sie stehen in den Programmen des Adventskalenders nur, um die abgedruckten Beispiele den heruntergeladenen Programmen einfacher zuordnen zu können. Wenn Sie die Programme selber schreiben, können Sie diese Kommentarzeilen weglassen.



Klicken Sie in der Arduino IDE auf das Symbol *New* und schreiben Sie das abgebildete Programm.



Sie finden das Programm `01blink` auch in der Arduino IDE, wenn Sie die Dateien aus dem Download wie beschrieben in den Arduino-Ordner kopiert haben. Klicken Sie dazu auf das Symbol *Open*.




Das Programm `01blink` in der Arduino IDE

Farbige Syntaxhervorhebung erleichtert es, bestimmte Programmelemente zu erkennen.

Arduino-Adventskalender

Programme auf den Nano übertragen

 Jetzt muss das fertige Programm auf den Nano übertragen werden. Klicken Sie dazu auf das Symbol *Upload*. Im unteren Teilfenster der Arduino IDE werden ein paar Statusmeldungen angezeigt. Nachdem das Hochladen abgeschlossen ist, erscheint hier *upload complete*. Die LED auf dem Nano blinkt. Das Programm braucht also nicht extra gestartet zu werden.

So funktioniert das Programm

Alle Programme in der Arduino IDE bestehen aus mindestens zwei Funktionen. Die Funktion `void setup()` läuft ein einziges Mal beim Start und wird meistens zur Einrichtung der Pins verwendet. Selbst wenn das nicht nötig ist, bleibt sie einfach leer, muss aber im Programm vorhanden sein.

```
void setup() {  
  pinMode(LED_BUILTIN, OUTPUT);  
}
```

In diesem Programm wird die eingebaute LED, die über die Konstante `LED_BUILTIN` vordefiniert ist, aber auch über den Pin 13 angesprochen werden kann, als Ausgang definiert. Pins, an denen LEDs angeschlossen sind, sind immer Ausgänge.

Die Funktion `void loop()` wird immer wieder wiederholt, bis man die Stromversorgung trennt oder den Reset-Knopf drückt.

```
void loop() {  
  digitalWrite(LED_BUILTIN, HIGH);  
  delay(50);  
  digitalWrite(LED_BUILTIN, LOW);  
  delay(50);  
}
```

In der Digitalelektronik gibt es zwei Schaltzustände, `HIGH` und `LOW`. `HIGH` steht für die Betriebsspannung, beim Nano sind das +5 V, `LOW` bedeutet ein Massesignal mit 0 V.

Die Funktion `digitalWrite()` setzt einen digitalen Anschlusspin, in unserem Fall die eingebaute LED, auf `HIGH`-Signal und schaltet sie damit ein. Der erste Parameter dieser Funktion bezeichnet immer den Pin, der zweite das Signal, das dort ausgegeben werden soll.

Anschließend wartet das Programm 50 Millisekunden, bevor ein zweiter Aufruf der `digitalWrite()`-Funktion das Signal `LOW` auf der eingebauten LED ausgibt und diese damit ausschaltet.

Nach weiteren 50 Millisekunden starte die Endlosschleife neu, sodass die LED ständig blinkt.

Tag 2

Heute im Adventskalender

- Steckbrett
- LED rot mit Vorwiderstand

Haus zusammenbauen

Nachdem der Teig über Nacht geruht hat, rollen Sie ihn auf einer mit Mehl bestreuten Arbeitsfläche etwa 1 cm dick aus. Schneiden Sie dann die Vorlagen für die Teile des Hauses aus der Rückseite dieses Adventskalenders aus, legen Sie sie auf die Lebkuchenplatte und schneiden Sie die Teile aus. Beachten Sie dabei die Hinweise auf der Rückseite. Sie brauchen 1x Bodenplatte, 2x Dach, 1x Vorderseite, 1x Rückseite. Die Bodenplatte kann natürlich auch größer sein, wenn Sie um das Haus herum noch einen Garten gestalten möchten.

Aus dem übrigen Teig können Sie Bäume oder andere Figuren ausstechen.



Legen Sie die Teile vorsichtig auf ein Backblech mit Backpapier und backen Sie sie bei 175° etwa 10 bis 15 Minuten lang. Lassen Sie die Teile nach dem Backen über Nacht gut abkühlen.

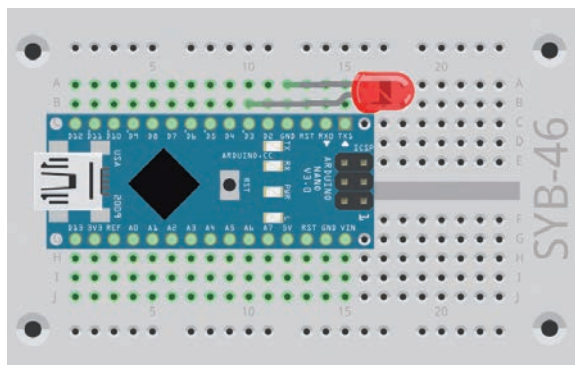


Arduino-Adventskalender

Zwei LEDs blinken im Wechsel

Die eingebaute LED soll mit einer extern angeschlossenen LED im Wechsel blinken.

Bauteile: Nano-Board, Steckbrett, LED rot mit Vorwiderstand



fritzing LED am Pin D3

Die Anode der LED (langer Draht) ist am Pin D3 angeschlossen, die Kathode (kurzer Draht) am Pin GND.

```
//02blink
//Franzis Adventskalender für Arduino

int led1 = 3;

void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(led1, OUTPUT);
}

void loop() {
  digitalWrite(LED_BUILTIN, HIGH);
  digitalWrite(led1, LOW);
  delay(200);
  digitalWrite(LED_BUILTIN, LOW);
  digitalWrite(led1, HIGH);
  delay(200);
}
```

So funktioniert das Programm

Am Anfang des Programms wird die zusätzlich verwendete Pinnummer in einer Variablen vom Typ `Integer` gespeichert. Da die Pinnummer dann nur noch einmal im Programm auftaucht, braucht es nur leicht geändert zu werden, wenn die LED an einen anderen Pin angeschlossen werden soll.

```
int led1 = 3;
```

Die eingebaute LED ist über die Konstante `LED_BUILTIN` automatisch ansprechbar, ohne dass man sie eigens definieren muss. Die Funktion `void setup()` initialisiert beide LEDs als Ausgänge.

```
void setup() {  
  pinMode(LED_BUILTIN, OUTPUT);  
  pinMode(led1, OUTPUT);  
}
```

Die Funktion `void loop()` schaltet in jedem Durchlauf die eingebaute LED ein und die LED am Pin D3 aus und wartet dann 200 Millisekunden. Danach wird die LED am Pin D3 aus- und die eingebaute LED eingeschaltet. Nach einer weiteren Wartezeit von 200 Millisekunden startet die `void loop()`-Schleife erneut. Die Wartezeit wurde gegenüber dem Programm von gestern verlängert, damit das wechselnde Blinken deutlicher zu erkennen ist.

```
void loop() {  
  digitalWrite(LED_BUILTIN, HIGH);  
  digitalWrite(led1, LOW);  
  delay(200);  
  digitalWrite(LED_BUILTIN, LOW);  
  digitalWrite(led1, HIGH);  
  delay(200);  
}
```

Tag 3

Heute im Adventskalender

- Schaltdraht

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 ab. 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.

Haus verzieren

Kleben Sie die Teile des Hauses mit Zuckerguss zusammen. Für den Zuckerguss brauchen Sie:

- 1 Eiweiß
- 100 g Puderzucker

Schlagen Sie das Eiweiß schaumig und geben Sie nach und nach den Puderzucker dazu, bis eine gleichmäßige klebrige Masse entsteht. Damit kleben Sie die Teile zusammen. Lassen Sie oben am Dachfirst einen Spalt frei, damit Sie LEDs oder Drähte durchstecken können.



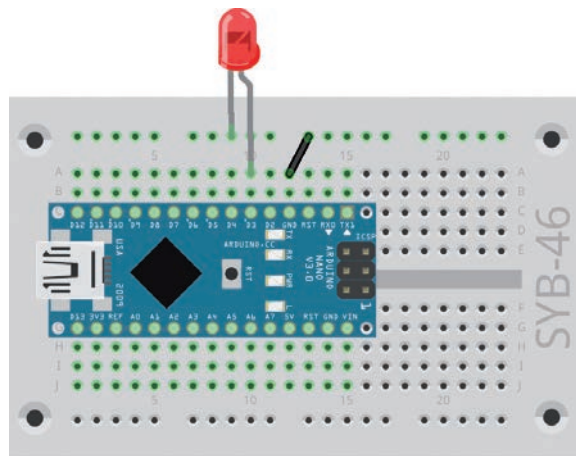


Arduino-Adventskalender

LED blinkt zufällig

Eine LED soll unregelmäßig blinken, wobei das Blinkintervall jedes Mal zufällig verändert wird.

Bauteile: Nano-Board, Steckbrett, LED rot mit Vorwiderstand, Drahtbrücke



LED am Pin D3 und an der Masseleiste

Die Schaltung von heute zeigt den typischen Schaltungsaufbau auf dem Steckbrett. Eine der horizontalen Kontaktleisten wird als Masseleitung verwendet, die über eine Drahtbrücke mit dem GND-Pin auf dem Nano 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 wieder mit dem Pin 3 verbunden.

```
//03blink
//Franzis Adventskalender für Arduino

int led1 = 3;

void setup() {
  pinMode(led1, OUTPUT);
}

void loop() {
  digitalWrite(led1, HIGH);
  delay(random(20, 200));
  digitalWrite(led1, LOW);
  delay(random(20, 200));
}
```

So funktioniert das Programm

Prinzipiell funktioniert das Programm ähnlich wie die Programme der letzten beiden Tage. Der wesentliche Unterschied ist, dass die Wartezeit zwischen zwei Schaltvorgängen jedes Mal zufällig festgelegt wird.

```
delay(random(20, 200));
```

In der Funktion `delay()` ist keine feste Zeit eingetragen, sondern eine weitere Funktion. Die Funktion `random()` ermittelt eine Zufallszahl. Der erste Parameter gibt die kleinste mögliche Zahl an, der zweite Parameter ist um 1 größer als die größte mögliche Zahl. In diesem Programm werden demnach Zahlen zwischen 20 und 199 erzeugt. Die Pause zwischen zwei Schaltvorgängen ist dann zufällig zwischen 20 und 199 Millisekunden lang.

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.

Tag 4

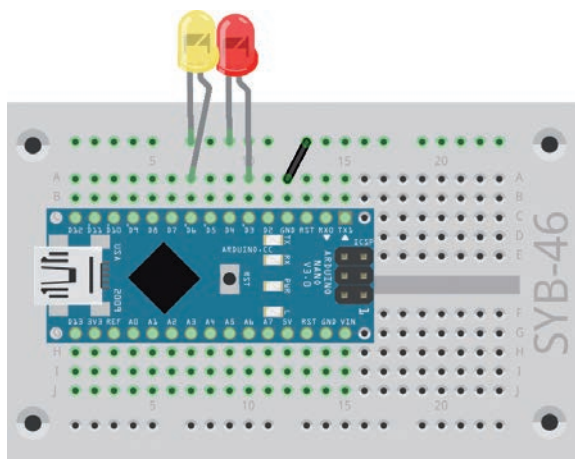
Heute im Adventskalender

- LED gelb mit Vorwiderstand

Aufgabe: Zwei LEDs blinken zufällig

Zwei LEDs sollen unregelmäßig blinken, wobei die eine LED durchschnittlich länger leuchten soll als die andere.

Bauteile: Nano-Board, Steckbrett, LED rot mit Vorwiderstand, LED gelb mit Vorwiderstand, Drahtbrücke



fritzing

Zwei LEDs an den Pins D3 und D6

```
//04blink
//Franzis Adventskalender für Arduino

int ledrot = 3;
int ledgelb = 6;

void setup() {
  pinMode(ledrot, OUTPUT);
  pinMode(ledgelb, OUTPUT);
}

void loop() {
  digitalWrite(ledrot, HIGH);
  digitalWrite(ledgelb, LOW);
  delay(random(20, 100));
  digitalWrite(ledrot, LOW);
  digitalWrite(ledgelb, HIGH);
  delay(random(50, 500));
}
```

So funktioniert das Programm

Am Anfang des Programms 04blink werden die beiden Pins D3 und D6 in den Variablen `ledrot` und `ledgelb` gespeichert und danach in der Funktion `void setup()` als Ausgänge initialisiert.

Im ersten Teil der Endlosschleife `void loop()` ist die rote LED ein- und die gelbe ausgeschaltet. In diesem Zustand wartet das Programm zwischen 20 und 100 Millisekunden. Im zweiten Teil, während die gelbe LED leuchtet und die rote dunkel ist, wartet das Programm zwischen 50 und 500 Millisekunden. Daher leuchtet die gelbe LED durchschnittlich länger als die rote.

Tag 5

Heute im Adventskalender

- LED grün mit Vorwiderstand

Aufgabe: Lauflicht

Drei LEDs leuchten als Lauflicht nacheinander.

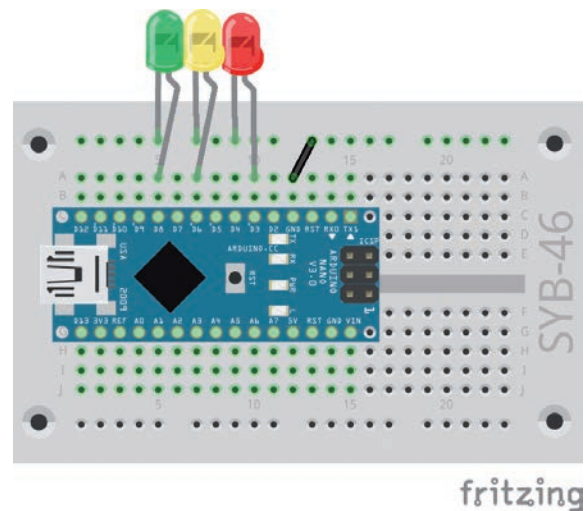
Bauteile: Nano-Board, Steckbrett, LED rot mit Vorwiderstand, LED gelb mit Vorwiderstand, LED grün mit Vorwiderstand, Drahtbrücke

```
//05blink
//Franzis Adventskalender für Arduino

int ledrot = 3;
int ledgelb = 6;
int ledgruen = 8;
int z = 200;

void setup() {
  pinMode(ledrot, OUTPUT);
  pinMode(ledgelb, OUTPUT);
  pinMode(ledgruen, OUTPUT);
}

void loop() {
  digitalWrite(ledrot, HIGH);
  digitalWrite(ledgelb, LOW);
  digitalWrite(ledgruen, LOW);
  delay(z);
  digitalWrite(ledrot, LOW);
  digitalWrite(ledgelb, HIGH);
  digitalWrite(ledgruen, LOW);
  delay(z);
  digitalWrite(ledrot, LOW);
  digitalWrite(ledgelb, LOW);
  digitalWrite(ledgruen, HIGH);
  delay(z);
}
```



Drei LEDs an den Pins D3, D6 und D8

So funktioniert das Programm

Am Anfang des Programms 05blink werden die Pins D3, D6 und D8 in den Variablen ledrot, ledgelb und ledgruen und zusätzlich die Wartezeit von 200 Millisekunden in der Variable z gespeichert.

Die Funktion void setup() initialisiert wieder die Pins für die LEDs als Ausgänge.

Die Endlosschleife void loop() arbeitet nacheinander drei Befehlsblöcke ab, wobei jedes Mal eine LED ein- und die anderen beiden ausgeschaltet sind. Anschließend wartet das Programm die eingestellte Zeit lang. Da für die Wartezeit immer die gleiche Variable verwendet wird, läuft das Lauflicht immer gleichmäßig und lässt sich über einen einzigen Wert beschleunigen oder abbremsen.

Tag 6

Heute im Adventskalender

- Taster

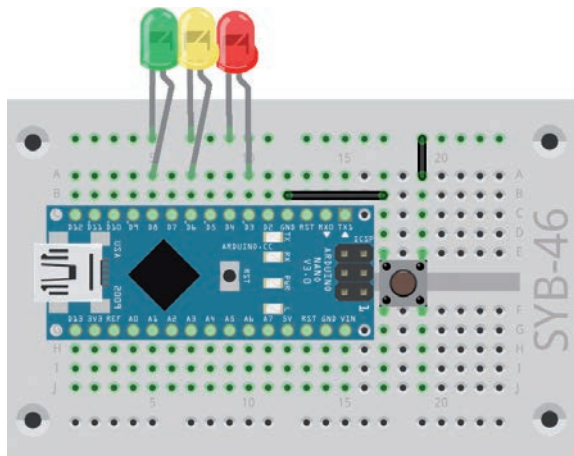
Taster

Solange die Taste gedrückt ist, sind beide Anschlüsse miteinander verbunden. Bei Tastern mit vier Anschlüssen sind die Anschlusspaare mit großen Abständen immer miteinander verbunden. Die Anschlüsse mit kleinem Abstand werden nur verbunden, solange die Taste gedrückt ist. Bei gedrückter Taste sind also alle vier Anschlüsse miteinander verbunden. Im Gegensatz zu einem Schalter rastet ein Taster nicht ein. Die Verbindung wird beim Loslassen sofort wieder getrennt.

Aufgabe: LED mit Taster schalten

Das aus dem letzten Programm bekannte Lauflicht soll nur laufen, solange ein Taster gedrückt ist.

Bauteile: Nano-Board, Steckbrett, LED rot mit Vorwiderstand, LED gelb mit Vorwiderstand, LED grün mit Vorwiderstand, Taster, 2 Drahtbrücken



Der Taster schaltet die gemeinsame Masseleitung der drei LEDs.

Der Taster schaltet das Lauflicht, ohne eigene Programmfunktionen zu benötigen. Solange der Taster gedrückt ist, ist die Masseschiene des Steckbretts mit dem GND-Pin auf dem Nano verbunden. Beim Loslassen wird diese Verbindung getrennt und der Stromkreis unterbrochen. Das Programm läuft weiter, aber die LEDs blinken nicht mehr.

So funktioniert das Programm

Das Programm `06blink` entspricht dem Programm `05blink`. Für den einfach in den elektrischen Stromkreis eingebauten Taster sind keine Änderungen im Programm erforderlich.

Tag 7

Heute im Adventskalender

- 10-KOhm-Widerstand

Widerstände und ihre Farbcodes

Widerstände werden unter anderem 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.

Die farbigen Ringe auf den Widerständen geben den Widerstandswert an. Mit etwas Übung sind sie deutlich leichter zu erkennen als die winzig kleinen 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.

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

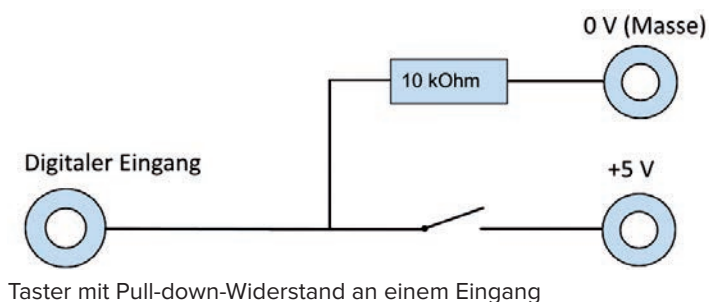
In welcher Richtung ein Widerstand eingebaut wird, ist egal. Bei LEDs dagegen spielt die Einbaurichtung eine wichtige Rolle.

Taster mit Pull-Down-Widerstand

Liegt auf einem als Eingang definierten digitalen Pin ein Signal von +5 V an, wird es als logisch HIGH bzw. 1 ausgewertet. Theoretisch könnten Sie also über einen Taster den jeweiligen Pin mit dem +5-V-Anschluss des Nano verbinden.

In vielen Fällen funktioniert diese simple Schaltung bereits, allerdings hätte der geschaltete Pin bei offenem Taster 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 Pins 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.

Die in der Abbildung der aufgebauten Schaltung linke Kontaktleiste des Tasters ist mit der +5-V-Leitung des Nano verbunden. Die in der Abbildung rechte Kontaktleiste des Tasters ist mit dem Pin D7 und über einen 10-kOhm-Pull-down-Widerstand (Braun-Schwarz-Orange) mit Masse verbunden.



Taster mit Pull-down-Widerstand an einem Eingang

Arduino-Adventskalender

Aufgabe: Ampel mit Taster ein- und ausschalten

Ein Tastendruck schaltet eine Ampelschaltung mit drei LEDs ein.

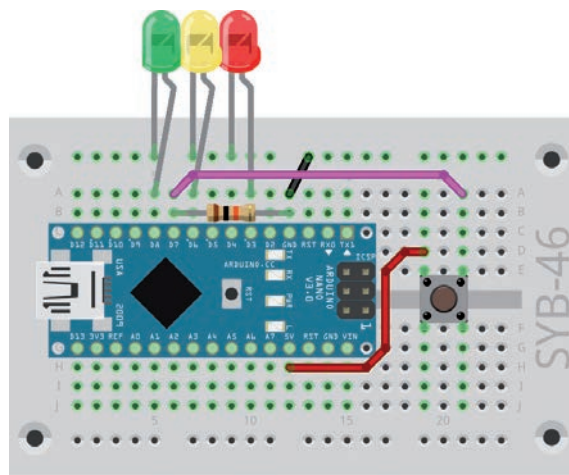
Bauteile: Nano-Board, Steckbrett, LED rot mit Vorwiderstand, LED gelb mit Vorwiderstand, LED grün mit Vorwiderstand, Taster, 10-KOhm-Widerstand, 3 Drahtbrücken

```
//07ampel  
//Franzis Adventskalender für Arduino
```

```
int ledrot = 3;  
int ledgelb = 6;  
int ledgruen = 8;  
int taste = 7;  
int z1 = 500;  
int z2 = 2000;
```

```
void setup() {  
  pinMode(ledrot, OUTPUT);  
  pinMode(ledgelb, OUTPUT);  
  pinMode(ledgruen, OUTPUT);  
  pinMode(taste, INPUT);  
}
```

```
void loop() {  
  if (digitalRead(taste) == HIGH){  
    digitalWrite(ledrot, HIGH);  
    digitalWrite(ledgelb, LOW);  
    digitalWrite(ledgruen, LOW);  
    delay(z1);  
    digitalWrite(ledgelb, HIGH);  
    delay(z1);  
    digitalWrite(ledrot, LOW);  
    digitalWrite(ledgelb, LOW);  
    digitalWrite(ledgruen, HIGH);  
    delay(z2);  
    digitalWrite(ledgelb, HIGH);  
    digitalWrite(ledgruen, LOW);  
    delay(z1);  
    digitalWrite(ledrot, HIGH);  
    digitalWrite(ledgelb, LOW);  
    delay(z2);  
  }  
}
```



Taster mit Pull-Down-Widerstand an einer Ampelschaltung

So funktioniert das Programm

Am Anfang des Programms werden die Pin-Nummern der drei LEDs und des Tasters in Variablen gespeichert. Zusätzlich wird die Zeit der kurzen Ampelphasen Rot/Gelb und Gelb in der Variable `z1` festgelegt, die Zeit der langen Ampelphasen Rot und Grün in der Variable `z2`.

Die Funktion `void setup()` initialisiert wieder die Pins für die LEDs als Ausgänge und zusätzlich den Pin für den Taster als Eingang.

Die Endlosschleife `void loop()` liest als Erstes mit der Funktion `digitalRead()` den Pin, an dem der Taster angeschlossen ist, aus. Nur wenn er ein `HIGH`-Signal liefert, werden die Anweisungen innerhalb der Klammer `if ...{...}` ausgeführt. Hier werden nacheinander die LEDs des Ampelzyklus ein- und ausgeschaltet. So lange dieser Pin ein `LOW`-Signal liefert, wird die Endlosschleife immer wieder wiederholt, bis die Abfrage positiv ausfällt.

Nachdem ein Ampelzyklus durchlaufen ist, wartet das Programm wieder auf den nächsten Tastendruck.

Tag 8

Heute im Adventskalender

- 2 Verbindungskabel

Verbindungskabel

Heute ist ein Verbindungskabelpaar im Adventskalender. Diese Leitungen werden verwendet, um LEDs, die auf dem Lebkuchenhaus stecken, mit dem Steckbrett zu verbinden. Die Verbindungskabel haben an einem Ende kleine Drahtstecker, mit denen sie sich leicht in das Steckbrett stecken lassen. Am anderen Ende befinden sich Klemmen, in die die Drähte einer LED gesteckt werden können.

Aufgabe: LED blinkt auf dem Dach, über Taster schaltbar

Eine LED auf dem Lebkuchenhaus blinkt. Ein Druck auf den Taster schaltet ein regelmäßiges Blinken ein. Ein weiterer Tastendruck schaltet das Blinken wieder aus.

Bauteile: Nano-Board, Steckbrett, LED rot mit Vorwiderstand, LED gelb mit Vorwiderstand, LED grün mit Vorwiderstand, Taster, 10-KOhm-Widerstand, 2 Verbindungskabel, 3 Drahtbrücken

Die rote LED ist über die Verbindungskabel mit dem Steckbrett verbunden. Achte beim Aufbau der Schaltung darauf, dass die Kathode (kurzer Draht) der LED mit dem GND-Pin des Nano-Boards verbunden ist, die Anode (langer Draht) ist in dieser Schaltung mit dem Pin D3 auf dem Nano verbunden.

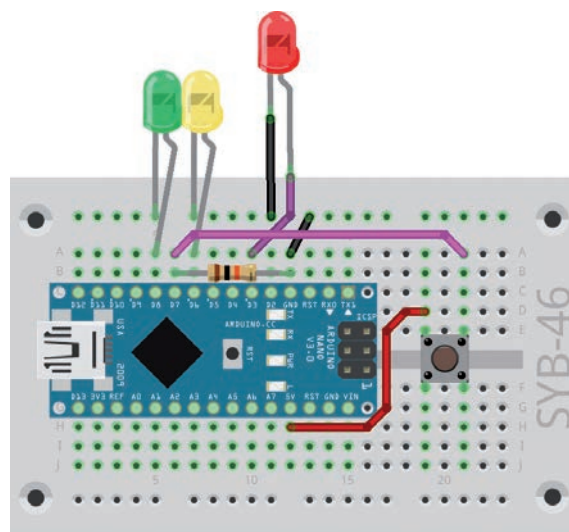
Die beiden anderen LEDs aus dem letzten Experiment werden eingeschaltet, solange die rote LED nicht blinkt.

```
//08blink
//Franzis Adventskalender für Arduino
```

```
int ledrot = 3;
int ledgelb = 6;
int ledgruen = 8;
int taste = 7;
int z = 200;
bool r = false;
```

```
void setup() {
  pinMode(ledrot, OUTPUT);
  pinMode(ledgelb, OUTPUT);
  pinMode(ledgruen, OUTPUT);
  pinMode(taste, INPUT);
}
```

```
void loop() {
  if(digitalRead(taste) == HIGH) {
    r = !r;
  }
  if(r) {
    digitalWrite(ledrot, HIGH);
    digitalWrite(ledgelb, LOW);
    digitalWrite(ledgruen, LOW);
    delay(z);
    digitalWrite(ledrot, LOW);
    delay(z);
  }
  else {
    digitalWrite(ledrot, LOW);
    digitalWrite(ledgelb, HIGH);
    digitalWrite(ledgruen, HIGH);
    delay(z);
  }
}
```



Die LED auf dem Lebkuchenhaus blinkt auf Tastendruck.

Arduino-Adventskalender

So funktioniert das Programm

Am Anfang des Programms wird zusätzlich zu den Pinnummern und der Wartezeit eine Variable `r` vom Typ `bool` definiert. Diese booleschen Variablen, auch als Logikvariablen bezeichnet, können nur die beiden Logikwerte `true` (wahr, HIGH, 1) oder `false` (falsch, LOW, 0) annehmen und sind daher sehr vielfältig einsetzbar.

Variablenamen

Variablen können (fast) beliebig benannt werden. Die Namen dürfen aus Buchstaben und Zahlen bestehen, wobei das erste Zeichen immer ein Buchstabe sein muss. Sonderzeichen und Leerzeichen sind nicht zulässig.

Einige Variablenamen sind unter Programmierern weit verbreitet, was zwar nicht bedeutet, dass sie unbedingt verwendet werden müssen, es macht ein Programm aber übersichtlicher:

`i` – Zähler für Schleifen (von: Intervall), geschachtelte Schleifen: `j`, `k`, ...

`n` – Einfache Zahlen für Zähler oder kleine Mengen (wie in der Mathematik)

`z` oder `t` – Zeitangaben (von: englisch ‚Time‘)

`r` – Status, das etwas eingeschaltet ist oder läuft (von: englisch ‚running‘)

In der Endlosschleife `void loop()` wird als Erstes mit der Funktion `digitalRead()` der Pin, an dem der Taster angeschlossen ist, ausgelesen. Ist der Taster in diesem Moment gedrückt, wird die boolesche Variable `r` auf den umgekehrten Wert gesetzt. Der Operator `!` macht aus `true` den Wert `false` und umgekehrt. Jeder Tastendruck im richtigen Moment schaltet die Variable `r` um.

```
if(digitalRead(taste) == HIGH) {  
    r = !r;  
}
```

Danach gibt es abhängig vom Wert `r` zwei Alternativen:

Ist `r = true`, werden die gelbe und die grüne LED ausgeschaltet, und die rote blinkt gleichmäßig.

Ist `r = false`, werden die gelbe und die grüne LED ein- und die rote ausgeschaltet.

Um den Logikwert einer Logikvariable abzufragen, ist keine komplizierte Formel nötig. Der Variablenname in der `if()` Abfrage reicht:

```
if (r){
```

Die erste Klammer der `if()`-Abfrage wird ausgeführt, wenn die Antwort auf die Frage wahr ist. Hinter dem Schlüsselwort `else` folgt eine weitere Klammer. Sie wird ausgeführt, wenn die Antwort auf die Frage falsch ist.

Tag 9

Heute im Adventskalender

- 2 Verbindungskabel

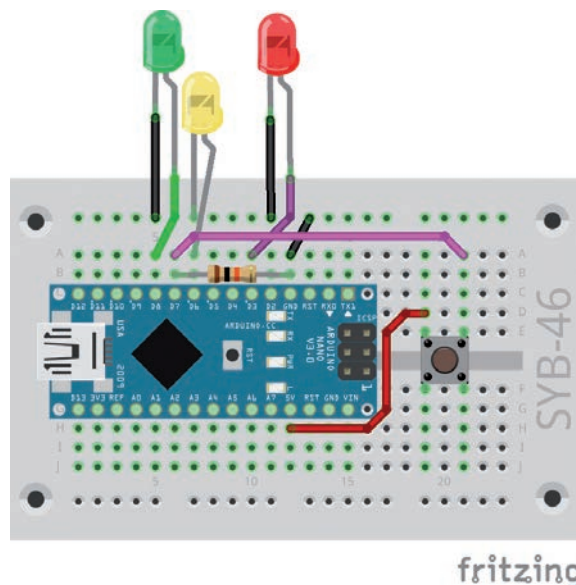
Aufgabe: verbesserte Abfrage des Tasters

Die Abfrage des Tasters im letzten Programm reagierte manchmal nicht richtig. Wurde der Taster zu lange gedrückt, wurde dies wie mehrere Tastendrucke verarbeitet.

Bauteile: Nano-Board, Steckbrett, LED rot mit Vorwiderstand, LED gelb mit Vorwiderstand, LED grün mit Vorwiderstand, Taster, 10-KOhm-Widerstand, 4 Verbindungskabel, 3 Drahtbrücken

```
//09blink  
//Franzis Adventskalender für Arduino
```

```
int ledrot = 3;  
int ledgelb = 6;  
int ledgruen = 8;  
int taste = 7;  
int z = 200;  
bool r = false;  
  
void setup() {  
  pinMode(LED_BUILTIN, OUTPUT);  
  pinMode(ledrot, OUTPUT);  
  pinMode(ledgelb, OUTPUT);  
  pinMode(ledgruen, OUTPUT);  
  pinMode(taste, INPUT);  
}  
  
void loop() {  
  if(digitalRead(taste) == HIGH) {  
    while(digitalRead(taste) == HIGH) {  
      digitalWrite(LED_BUILTIN, HIGH);  
    }  
    digitalWrite(LED_BUILTIN, LOW);  
    r = !r;  
  }
```



Zwei LEDs blinken auf dem Dach

```
  }  
  if(r) {  
    digitalWrite(ledrot, HIGH);  
    digitalWrite(ledgelb, LOW);  
    digitalWrite(ledgruen, LOW);  
    delay(z);  
    digitalWrite(ledrot, LOW);  
    digitalWrite(ledgruen, HIGH);  
    delay(z);  
  }  
  else {  
    digitalWrite(ledrot, LOW);  
    digitalWrite(ledgelb, HIGH);  
    digitalWrite(ledgruen, LOW);  
    delay(z);  
  }  
}
```

Auch längeres Gedrückthalten des Tasters schaltet die LEDs eindeutig um. Die auf dem Nano eingebaute LED zeigt an, wenn die Taste gedrückt ist.

So funktioniert das Programm

Prinzipiell funktioniert das Programm ähnlich wie das Programm von gestern. Bei jedem Tastendruck wird die Logikvariable `r` umgeschaltet. Abhängig von deren Zustand, blinken entweder die beiden LEDs auf dem Dach des Lebkuchenhauses oder die gelbe LED leuchtet konstant.

In der Endlosschleife `void loop()` wird als Erstes wieder mit der Funktion `digitalRead()` der Pin, an dem der Taster angeschlossen ist, ausgelesen. Ist er `HIGH`, der Taster also gedrückt, startet eine `while()`-Schleife, die weiter den Taster abfragt und, solange er gedrückt ist, nichts anderes tut, als die eingebaute LED leuchten zu lassen. Erst wenn der Taster nicht mehr gedrückt ist, endet diese Schleife, schaltet die eingebaute LED wieder aus und die Logikvariable `r` um. Auf diese Weise umgeht das Programm Fehlinterpretationen, wenn der Taster etwas länger gedrückt wird.

Anschließend entscheidet eine `if ... else`-Abfrage, ob das Wechselblinklicht läuft oder die gelbe LED eingeschaltet wird.

Tag 10

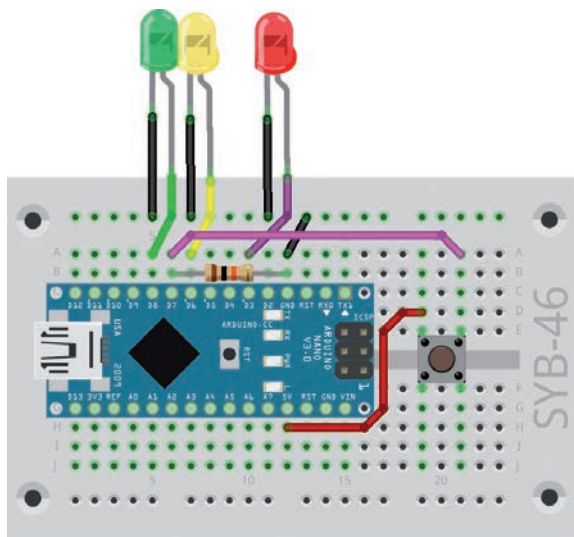
Heute im Adventskalender

- 2 Verbindungskabel

Lauflicht mit verschiedenen Geschwindigkeiten

Drei LEDs leuchten wechselweise als Lauflicht. Über den Taster lässt sich die Geschwindigkeit stufenweise verändern.

Bauteile: Nano-Board, Steckbrett, LED rot mit Vorwiderstand, LED gelb mit Vorwiderstand, LED grün mit Vorwiderstand, Taster, 10-KOhm-Widerstand, 6 Verbindungskabel, 3 Drahtbrücken



fritzing Drei LEDs als Lauflicht auf dem Dach des Lebkuchenhauses

```
//10blink
//Franzis Adventskalender für Arduino

int ledrot = 3;
int ledgelb = 6;
int ledgruen = 8;
int taste = 7;
int z1 = 400;
int z = z1;

void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(ledrot, OUTPUT);
  pinMode(ledgelb, OUTPUT);
  pinMode(ledgruen, OUTPUT);
  pinMode(taste, INPUT);
}

void loop() {
  if(digitalRead(taste) == HIGH) {
    while(digitalRead(taste) == HIGH) {
      digitalWrite(LED_BUILTIN, HIGH);
    }
    digitalWrite(LED_BUILTIN, LOW);
    z = z/2;
    if(z < z1/4) {
      z = z1;
    }
    digitalWrite(ledrot, HIGH);
    digitalWrite(ledgelb, LOW);
    digitalWrite(ledgruen, LOW);
    delay(z);
    digitalWrite(ledrot, LOW);
    digitalWrite(ledgelb, HIGH);
    digitalWrite(ledgruen, LOW);
    delay(z);
    digitalWrite(ledrot, LOW);
    digitalWrite(ledgelb, LOW);
    digitalWrite(ledgruen, HIGH);
    delay(z);
  }
}
```


So funktioniert das Programm

Am Anfang des Programms werden zwei Variablen für die Blinkzeit angelegt:

```
int z1 = 400;  
int z = z1;
```

Die Variable `z1` speichert die Blinkzeit am Anfang des Programms. Diese wird nur an dieser einen Stelle angegeben, kann also problemlos verändert werden. Die Variable `z` enthält während des Programmablaufs die jeweils aktuelle Blinkzeit.

Die Abfrage des Tasters erfolgt wieder wie im Programm von gestern mit einer `while()` Schleife und der eingebauten LED als Kontrolleuchte für den Taster. Diesmal gibt es keine Logikvariable, sondern bei jedem Tastendruck wird die Blinkzeit halbiert, das Lauflicht beschleunigt.

```
z = z/2;
```

Die Blinkzeit soll in drei Stufen veränderbar sein: `z1`, `z1/2`, `z1/4`. Nach der dritten Stufe wird sie beim nächsten Tastendruck wieder auf den Startwert gesetzt. Dazu wird geprüft, ob die Variable `z` nach dem Halbieren kleiner ist als ein Viertel des Startwertes. Ist das der Fall, wird sie wieder auf den Startwert `z1` zurückgesetzt.

```
if (z < z1/4){  
    z = z1;  
}
```

Anschließend läuft das Lauflicht, unabhängig davon, ob die Zeit verändert wurde, mit der aktuellen Blinkzeit `z` einmal durch. Am Ende beginnt die Endlosschleife erneut mit der Abfrage des Tasters.

Tag 11

Heute im Adventskalender

- 2 Verbindungskabel

LEDs dimmen

Zwei LEDs sollen gleichmäßig auf- und abgeblendet werden.

Pulsweitenmodulation

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

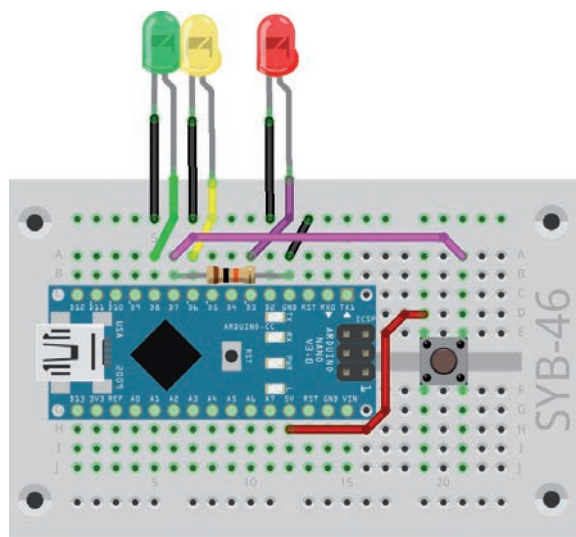
Mit einem Trick ist es dennoch möglich, die Helligkeit einer LED an einem digitalen Pin zu regeln. Lässt man eine LED schnell genug blinken, nimmt das menschliche Auge das nicht mehr als Blinken wahr. Die als Pulsweitenmodulation (PWM) bezeichnete Technik erzeugt ein pulsierendes Signal, das sich in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

Bauteile: Nano-Board, Steckbrett, LED rot mit Vorwiderstand, LED gelb mit Vorwiderstand, LED grün mit Vorwiderstand, Taster, 10-KOhm-Widerstand, 6 Verbindungskabel, 3 Drahtbrücken



fritzing

Die Schaltung entspricht der von gestern. Die grüne LED wird heute nicht verwendet, kann aber eingebaut bleiben.

Pins für PWM-Signale

Die Pins D3, D5, D6, D9, D10, D11 sind in den Abbildungen des Nano mit einem '~'-Symbol gekennzeichnet. Nur diese Pins können für Pulsweitenmodulation verwendet werden. Auf dem original Nano-Board fehlt diese Kennzeichnung leider.

```
//11pwm
//Franzis Adventskalender für Arduino

int ledrot = 3;
int ledgelb = 6;
int taste = 7;
int hell = 0;
int schritt = 5;
int z = 10;
bool r = false;

void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(ledrot, OUTPUT);
  pinMode(ledgelb, OUTPUT);
  pinMode(taste, INPUT);
}

void loop() {
  if(digitalRead(taste) == HIGH) {
    while(digitalRead(taste) == HIGH) {
      digitalWrite(LED_BUILTIN, HIGH);
    }
    digitalWrite(LED_BUILTIN, LOW);
    r = !r;
  }
  if(r) {
    analogWrite(ledrot, hell);
    analogWrite(ledgelb, hell);
    hell += schritt;
    if(hell == 0 || hell == 255) {
      schritt = -schritt;
    }
    delay(z);
  }
  else {
    digitalWrite(ledrot, LOW);
    digitalWrite(ledgelb, LOW);
  }
}
```

So funktioniert das Programm

Das Programm dimmt die rote und die gelbe LED zyklisch heller und dunkler und verwendet dazu die Funktion `analogWrite()`, die Werte zwischen 0 und 255 auf einen PWM-Pin schreibt. Dabei handelt es sich nicht um analoge Spannungswerte, sondern um PWM-Signale. Der Wert 255 entspricht einem Tastverhältnis von 100%. Dabei leuchtet die LED mit voller Stärke.

Arduino-Adventskalender

Am Anfang werden zusätzlich zu den Pins und der Logikvariable `r` für die Tasterabfrage drei neue Variablen eingerichtet:

`hell` – PWM-Wert für die Helligkeit der LED

`schritt` – Schrittweite beim Dimmen

`z` – Wartezeit zwischen den Schaltvorgängen

```
int hell = 0;
int schritt = 5;
int z = 10;
```

Die Funktion `void setup()` definiert den Pin für den Taster als Eingang und die Pins der LEDs als Ausgänge. Hier braucht man für PWM keine speziellen Einstellungen vorzunehmen.

In der Prozedur `void loop()` wird wieder, wie bereits in früheren Programmen, der Taster abgefragt, der den PWM-Zyklus ein- oder ausschaltet.

Danach werden die LEDs angesteuert. Die Funktion `analogWrite()` gibt am angegebenen Pin ein PWM-Signal aus. Im Gegensatz zu `digitalWrite()` kann ein Wert zwischen 0 und 255 ausgegeben werden, der in diesem Fall in der Variablen `hell` steht.

```
analogWrite(ledrot, hell);
analogWrite(ledgelb, hell);
```

Danach wird die Variable `hell` um den Wert von `schritt` erhöht. Der Operator `+=` addiert einen Wert zu einer Variable hinzu.

```
hell += schritt;
```

Wenn der Helligkeitswert den 255 für volle Helligkeit erreicht, bekommt der Wert `schritt` ein negatives Vorzeichen. Damit wird automatisch, ohne die Berechnungsformel zu verändern, wieder abwärts gezählt. Umgekehrt wird auch bei Erreichen von 0 das Vorzeichen umgekehrt, um wieder aufwärts zu zählen. Der Operator `||` bedeutet *oder*.

```
if (hell == 0 || hell == 255)
{
    schritt = -schritt;
}
```

Zum Schluss wartet das Programm die in der Variable `z` festgelegte Zeit und fragt dann wieder den Taster ab. Solange er nicht gedrückt wurde, werden die beiden PWM-Pins auf den neuen Helligkeitswert gesetzt.

Bei Druck auf den Taster wird die Logikvariable `r` umgeschaltet. Steht sie auf *false*, werden beide LEDs ausgeschaltet.

Tag 12

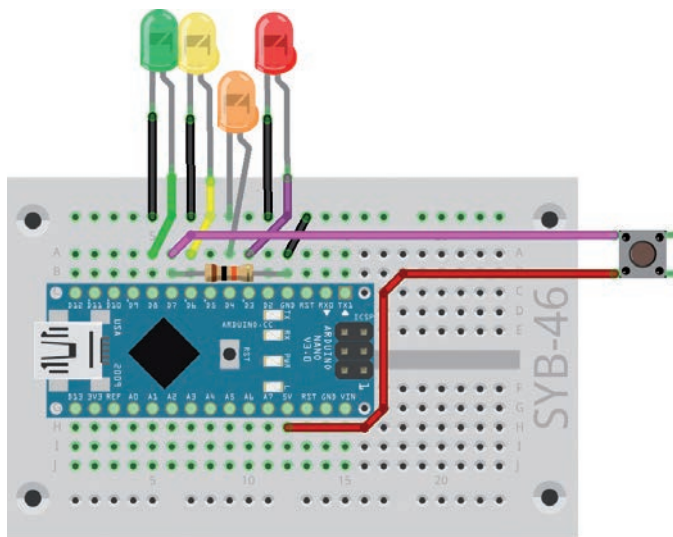
Heute im Adventskalender

- LED orange mit Vorwiderstand

Hausbeleuchtung mit Klingelknopf

Das Lauflicht startet beim Programmstart automatisch. Beim Drücken des Klingelknopfs an der Haustür leuchtet die orangefarbene LED im Haus eine Zeit lang. Das Lauflicht läuft die ganze Zeit weiter.

Bauteile: Nano-Board, Steckbrett, LED rot mit Vorwiderstand, LED gelb mit Vorwiderstand, LED grün mit Vorwiderstand, LED orange mit Vorwiderstand, Taster, 10-KOhm-Widerstand, 8 Verbindungskabel, 1 Drahtbrücke



fritzing

Der Klingeltaster an der Haustür ist über zwei Verbindungskabel am Steckbrett angeschlossen.

```
//12licht
//Franzis Adventskalender für Arduino

int ledrot = 3;
int ledorange = 4;
int ledgelb = 6;
int ledgruen = 8;
int taste = 7;
int blink = 100;
int licht = 1000;
long z = 0;

void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(ledrot, OUTPUT);
  pinMode(ledorange, OUTPUT);
  pinMode(ledgelb, OUTPUT);
  pinMode(ledgruen, OUTPUT);
  pinMode(taste, INPUT);
}

void loop() {
  digitalWrite(ledrot, HIGH);
```

```
digitalWrite(ledgelb, LOW);
digitalWrite(ledgruen, LOW);
delay(blink);
digitalWrite(ledrot, LOW);
digitalWrite(ledgelb, HIGH);
digitalWrite(ledgruen, LOW);
delay(blink);
digitalWrite(ledrot, LOW);
digitalWrite(ledgelb, LOW);
digitalWrite(ledgruen, HIGH);
delay(blink);
if(digitalRead(taste) == HIGH) {
  while(digitalRead(taste) == HIGH) {
    digitalWrite(LED_BUILTIN, HIGH);
  }
  digitalWrite(LED_BUILTIN, LOW);
  z = millis();
  digitalWrite(ledorange, HIGH);
}
if(millis() > z + licht) {
  digitalWrite(ledorange, LOW);
}
}
```

Arduino-Adventskalender

So funktioniert das Programm

Am Anfang des Programms werden, wie in früheren Programmen, die Pin-Nummern in Variablen gespeichert. Zusätzlich werden weitere Variablen definiert:

`blink` – Blinkzeit einer einzelnen LED im Lauflicht

`licht` – Leuchtzeit der orangen LED im Haus

`z` – zwischengespeicherte Zeit

```
int blink = 100;
int licht = 1000;
int z = 0;
```

Die Prozedur `void loop()` startet das Lauflicht, das permanent, unabhängig vom Taster, laufen soll. Am Ende jedes Durchlaufs wird wie in den letzten Programmen der Taster abgefragt.

In diesem Programm kann der Taster nicht einfach mit `delay()` eine bestimmte Zeit leuchten, da während dieser Zeit das Programm nichts weiter tut, als zu warten. Das Lauflicht würde also stehen bleiben.

Deshalb verwenden wir hier die interne Uhr des Nano, die zwar keine wirkliche Uhrzeit kennt, aber wie eine Stoppuhr beim Start eines Programms automatisch startet und dann ständig mitläuft.

```
if (digitalRead(taste) == HIGH){
  while (digitalRead(taste) == HIGH){
    digitalWrite(LED_BUILTIN, HIGH);
  }
  digitalWrite(LED_BUILTIN, LOW);
  z = millis();
  digitalWrite(ledorange, HIGH);
}
```

Nachdem der Taster losgelassen wurde, liest die Funktion `millis()` die aktuelle Zeit der Uhr in Millisekunden aus und speichert sie in der Variablen `z`. Außerdem wird die orangefarbene LED eingeschaltet.

In jedem Durchlauf der Endlosschleife `void loop()` wird abgefragt, ob die aktuelle Zeit der Uhr größer ist als die in `z` gespeicherte Startzeit zuzüglich der vorgesehenen Leuchtzeit der LED, die in der Variablen `licht` gespeichert ist. Ist das der Fall, wird die orangefarbene LED wieder ausgeschaltet. Das Lauflicht läuft von der orangefarbenen LED ungestört weiter.

Da die Funktion `millis()` einen Wert vom Typ `long` liefert, muss auch die Variable `z` von diesem Typ sein.

Tag 13

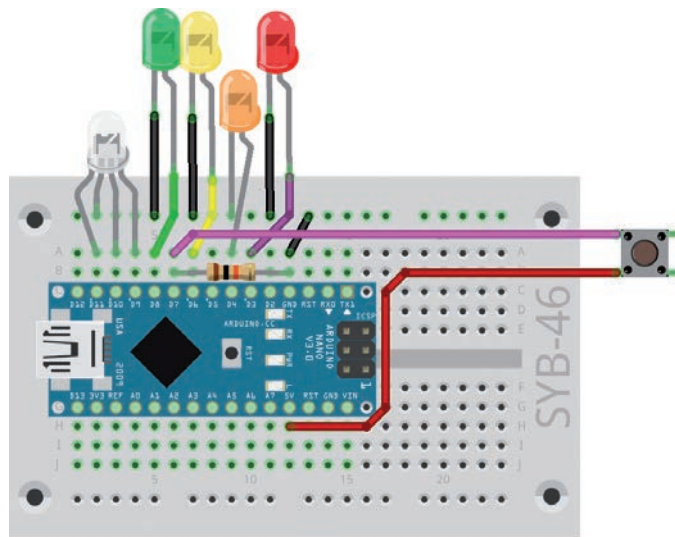
Heute im Adventskalender

- RGB-LED mit Vorwiderstand

Farbwechsel auf RGB-LED

Die RGB-LED zeigt nacheinander die drei Grundfarben und daraus gemischte Farben an.

Bauteile: Nano-Board, Steckbrett, LED rot mit Vorwiderstand, LED gelb mit Vorwiderstand, LED grün mit Vorwiderstand, LED orange mit Vorwiderstand, RGB-LED mit Vorwiderstand, Taster, 10-KOhm-Widerstand, 8 Verbindungskabel, Drahtbrücke



fritzing

RGB-LED an den Pins D11, D10, D9. Die einfarbigen LEDs werden in diesem Programm nicht verwendet, können aber eingebaut bleiben.

Arduino-Adventskalender

```
//13rgb
//Franzis Adventskalender für Arduino

int rot = 11;
int gruen = 10;
int blau = 9;
int taste = 7;
int z = 500;
bool r = false;

void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(rot, OUTPUT);
  pinMode(gruen, OUTPUT);
  pinMode(blau, OUTPUT);
  pinMode(taste, INPUT);
}

void loop() {
  if(digitalRead(taste) == HIGH) {
    while(digitalRead(taste) == HIGH) {
      digitalWrite(LED_BUILTIN, HIGH);
    }
    digitalWrite(LED_BUILTIN, LOW);
    r = !r;
  }
  if(r) {
    digitalWrite(rot, HIGH);
    delay(z);
    digitalWrite(gruen, HIGH);
    delay(z);
    digitalWrite(rot, LOW);
    delay(z);
    digitalWrite(blau, HIGH);
    delay(z);
    digitalWrite(gruen, LOW);
    delay(z);
    digitalWrite(rot, HIGH);
    delay(z);
    digitalWrite(blau, LOW);
  }
  else {
    digitalWrite(rot, LOW);
    digitalWrite(gruen, LOW);
    digitalWrite(blau, LOW);
    delay(z);
  }
}
```

So funktioniert das Programm

Die RGB-LED besteht intern aus drei einzelnen LEDs und wird im Programm auch genauso angesteuert. Am Anfang werden drei Variablen `rot`, `gruen` und `blau` für die drei Pins angelegt und die Pins als Ausgänge initialisiert.

In der Prozedur `void loop()` wird nach bereits bekanntem Prinzip der Taster abgefragt, der den Lichteffect ein- oder ausschaltet. Am Anfang des Leuchtzyklus ist nur die rote Farbe der RGB-LED eingeschaltet. Danach werden nacheinander immer einzelne Farben ein- oder ausgeschaltet, wodurch sich diese Farbfolge ergibt: Rot – Rot/Grün – Grün – Grün/Blau – Blau – Rot/Blau – Rot ...

Wenn der Taster über die Logikvariable `r` den Leuchtzyklus ausschaltet, werden alle drei Pins auf `LOW` gesetzt.

Tag 14

Heute im Adventskalender

- 2 Verbindungskabel

Farbverlauf auf RGB-LED

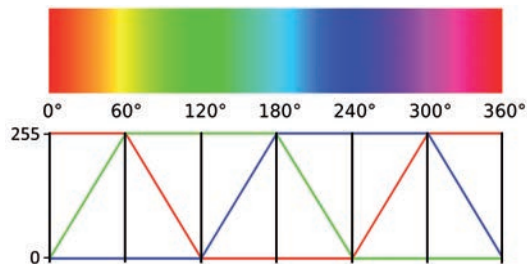
Die RGB-LED zeigt einen Farbverlauf über das gesamte HSV-Farbspektrum.

Bauteile: Nano-Board, Steckbrett, LED rot mit Vorwiderstand, LED gelb mit Vorwiderstand, LED grün mit Vorwiderstand, LED orange mit Vorwiderstand, RGB-LED mit Vorwiderstand, Taster, 10-KOhm-Widerstand, 8 Verbindungskabel, Drahtbrücke

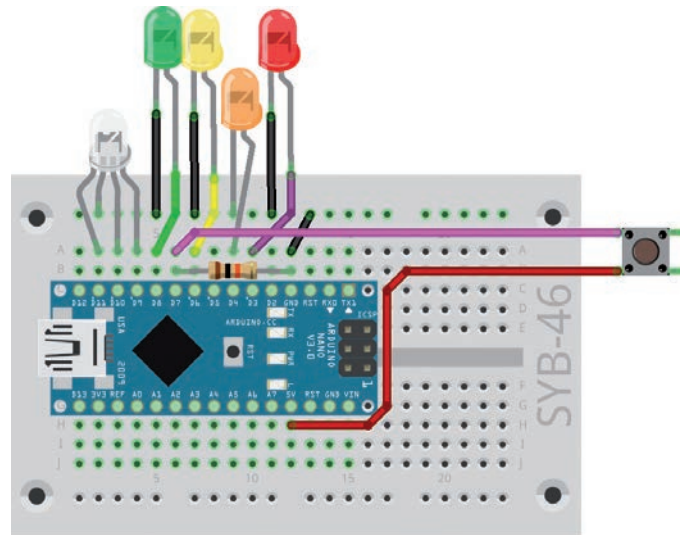
Das RGB-Farbsystem, das bisher in allen Programmen genutzt wurde, beschreibt Farben als drei Komponenten, Rot, Grün und Blau, die miteinander gemischt werden. Für Menschen ist es relativ schwierig, sich eine Mischfarbe vorzustellen.

Im Gegensatz beschreibt das HSV-Farbsystem die Farben über die Werte H=Hue (Farbwert), S=Saturation (Sättigung) und V=Value (Helligkeitswert). Der H-Wert kann entsprechend den Gradzahlen auf einem Farbkreis Werte zwischen 0 und 360 annehmen. Durch einfache Veränderung des H-Wertes können alle Farben des Farbspektrums in voller Intensität beschrieben werden, wenn man die beiden anderen Werte auf Maximum einstellt.

Wie die Grafik zeigt, gelten für die sechs 60°-Bereiche des Farbreises jeweils eigene lineare Verlaufskurven. In jedem dieser Bereiche wird ein Farbwert variabel errechnet, und die anderen beiden werden auf Minimum 0 oder Maximum 255 gesetzt.



Die Grafik zeigt, wie der H-Wert einer HSV-Farbe in RGB-Werte umgerechnet wird.



fritzing

RGB-LED an den Pins D11, D10, D9. Die einfarbigen LEDs werden in diesem Programm nicht verwendet, können aber eingebaut bleiben.

Arduino-Adventskalender

```
//14rgb
//Franzis Adventskalender für Arduino

int rot = 11;
int gruen = 10;
int blau = 9;
int taste = 7;
int z = 5;
int i = 0;
bool r = false;

void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(rot, OUTPUT);
  pinMode(gruen, OUTPUT);
  pinMode(blau, OUTPUT);
  pinMode(taste, INPUT);
}

void loop() {
  if(digitalRead(taste) == HIGH) {
    while(digitalRead(taste) == HIGH) {
      digitalWrite(LED_BUILTIN, HIGH);
    }
    digitalWrite(LED_BUILTIN, LOW);
    r = !r;
  }
  if(r) {
    digitalWrite(rot, HIGH);
    digitalWrite(blau, LOW);
    for(i=0; i <= 255; i++) {
      analogWrite(gruen, i);
      delay(z);
    }
  }
  digitalWrite(gruen, HIGH);
  for(i=255; i >= 0; i--) {
    analogWrite(rot, i);
    delay(z);
  }
  digitalWrite(rot, LOW);
  for(i=0; i <= 255; i++) {
    analogWrite(blau, i);
    delay(z);
  }
  digitalWrite(blau, HIGH);
  for(i=255; i >= 0; i--) {
    analogWrite(gruen, i);
    delay(z);
  }
  digitalWrite(gruen, LOW);
  for(i=0; i <= 255; i++) {
    analogWrite(rot, i);
    delay(z);
  }
  digitalWrite(rot, HIGH);
  for(i=255; i >= 0; i--) {
    analogWrite(blau, i);
    delay(z);
  }
}
else {
  digitalWrite(rot, LOW);
  digitalWrite(gruen, LOW);
  digitalWrite(blau, LOW);
  delay(z);
}
}
```

So funktioniert das Programm

Die Definition der Variablen läuft wie in den bereits bekannten Programmen. Die Zeit z zwischen zwei Farbänderungen ist hier sehr kurz voreingestellt, da das Programm $6 * 256 = 1536$ Schritte durchläuft und deshalb schon bei 4 Millisekunden ein Schleifendurchlauf zwischen jedem Schritt 7680 Millisekunden dauert.

Nacheinander laufen sechs `for()`-Schleifen, die jeweils eine Farbe linear von 0 bis 255 oder umgekehrt von 255 bis 0 verändern. Davor wird eine andere Farbe fest auf Maximal- oder Minimalwert eingestellt. `for()`-Schleifen werden in den meisten Fällen durch einen Zähler gesteuert, können aber auch durch andere Bedingungen sehr vielfältig verändert werden. Dazu werden in der Funktion drei Parameter eingetragen:

Start – Wird am Anfang der Schleife festgelegt, in diesem Programm $i=0$, der Schleifenzähler wird auf 0 gesetzt.

Bedingung – Solange sie wahr ist, läuft die Schleife, in diesem Programm $i \leq 255$. Die Schleife läuft, solange der Schleifenzähler kleiner oder gleich 255 ist.

Veränderung – In jedem Durchlauf wird der Zähler hoch- oder heruntergezählt, in diesem Programm $i++$, womit die Variable i um 1 erhöht wird. Es kann auch eine andere Aktion durchgeführt werden, die irgendwann zum Beenden der Schleife führt.

Der Taster wird immer nur in der Rotphase am Anfang jedes Schleifendurchlaufs abgefragt. Um den Lichteffekt auszuschalten, drücken Sie den Taster so lange, bis die Kontroll-LED auf dem Nano leuchtet.

Tag 15

Heute im Adventskalender

- 2 Verbindungskabel

Aufgabe: Fußgängerampel

Eine Fußgängerampel wird über einen Taster betätigt. Beim Drücken des Tasters startet der typische Ampelzyklus einer Fußgängerampel. Im Ruhezustand leuchtet die Fußgängerampel rot, die Verkehrsampel grün.

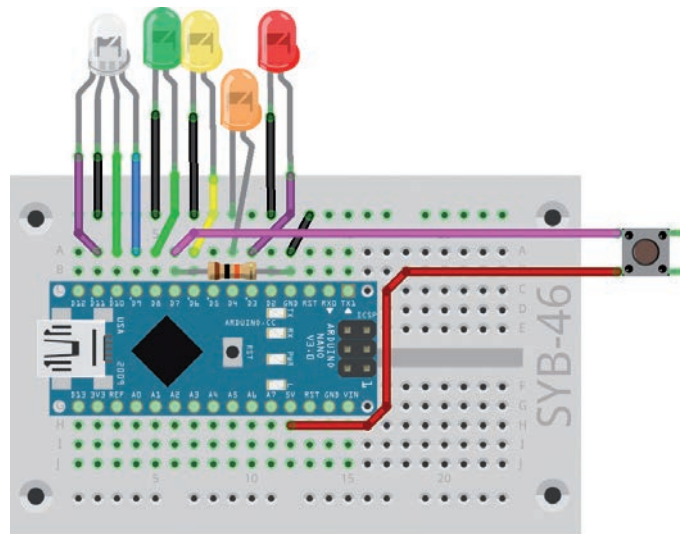
Bauteile: Nano-Board, Steckbrett, LED rot mit Vorwiderstand, LED gelb mit Vorwiderstand, LED grün mit Vorwiderstand, LED orange mit Vorwiderstand, RGB-LED mit Vorwiderstand, Taster, 10-KOhm-Widerstand, 12 Verbindungskabel, Drahtbrücke

```
//15ampel
//Franzis Adventskalender für Arduino
```

```
int ledrot = 3;
int ledgelb = 6;
int ledgruen = 8;
int rot = 11;
int gruen = 10;
int taste = 7;
int z1 = 500;
int z2 = 2000;

void setup() {
  pinMode(ledrot, OUTPUT);
  pinMode(ledgelb, OUTPUT);
  pinMode(ledgruen, OUTPUT);
  pinMode(rot, OUTPUT);
  pinMode(gruen, OUTPUT);
  pinMode(taste, INPUT);
}

void loop() {
  digitalWrite(ledrot, LOW);
  digitalWrite(ledgelb, LOW);
  digitalWrite(ledgruen, HIGH);
  digitalWrite(rot, HIGH);
  digitalWrite(gruen, LOW);
  if(digitalRead(taste) == HIGH) {
```



Verkehrsampel mit drei LEDs und Fußgängerampel auf der RGB-LED. Die orangefarbene LED wird, wie auch die blaue Farbe der RGB-LED, in diesem Programm nicht verwendet, kann aber eingebaut bleiben.

```
while(digitalRead(taste) == HIGH) {
  digitalWrite(LED_BUILTIN, HIGH);
}
digitalWrite(LED_BUILTIN, LOW);
delay(z1);
digitalWrite(ledgelb, HIGH);
digitalWrite(ledgruen, LOW);
delay(z1);
digitalWrite(ledrot, HIGH);
digitalWrite(ledgelb, LOW);
delay(z1);
digitalWrite(rot, LOW);
digitalWrite(gruen, HIGH);
delay(z2);
digitalWrite(rot, HIGH);
digitalWrite(gruen, LOW);
delay(z1);
digitalWrite(ledgelb, HIGH);
delay(z1);
}
```

So funktioniert das Programm

Beim Start jedes Schleifendurchlaufs der Prozedur `void loop()` wird die Ampel in die Grundstellung gebracht. Die RGB-LED leuchtet als Fußgängerampel 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.

Für die unterschiedlichen Schaltphasen der Ampel sind zwei verschiedene Zeiten definiert. Die Grünphase für Fußgänger dauert länger als alle Zwischenphasen.

Tag 16

Heute im Adventskalender

- 2 Verbindungskabel

Bunte Lichteffekte

Der Taster schaltet zwischen vier verschiedenen Lichtmustern um: Lauflicht auf vier LEDs, vier LEDs blinken gleichzeitig, Farbwechsel auf der RGB-LED, RGB-LED blitzt weiß.

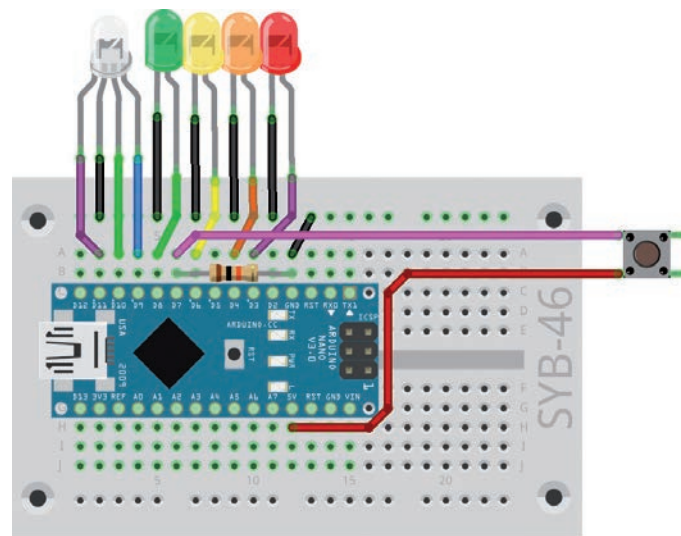
Bauteile: Nano-Board, Steckbrett, LED rot mit Vorwiderstand, LED gelb mit Vorwiderstand, LED grün mit Vorwiderstand, LED orange mit Vorwiderstand, RGB-LED mit Vorwiderstand, Taster, 10-KOhm-Widerstand, 14 Verbindungskabel, Drahtbrücke

```
//16blink
//Franzis Adventskalender für Arduino
```

```
int ledrot = 3;
int ledorange = 4;
int ledgelb = 6;
int ledgruen = 8;
int rot = 11;
int gruen = 10;
int blau = 9;
int taste = 7;
int z1 = 200;
int z2 = 1;
int z3 = 1000;
int n = 0;

void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(ledrot, OUTPUT);
  pinMode(ledorange, OUTPUT);
  pinMode(ledgelb, OUTPUT);
  pinMode(ledgruen, OUTPUT);
  pinMode(rot, OUTPUT);
  pinMode(gruen, OUTPUT);
  pinMode(blau, OUTPUT);
  pinMode(taste, INPUT);
}

void loop() {
  if(digitalRead(taste) == HIGH) {
    while(digitalRead(taste) == HIGH) {
      digitalWrite(LED_BUILTIN, HIGH);
    }
    digitalWrite(LED_BUILTIN, LOW);
    n++;
    if(n > 3) {
      n = 0;
    }
  }
  switch(n) {
    case 0:
      digitalWrite(ledrot, HIGH);
      delay(z1);
      digitalWrite(ledrot, LOW);
      digitalWrite(ledorange, HIGH);
      delay(z1);
      digitalWrite(ledorange, LOW);
```



RGB-LED und 4 LEDs auf dem Lebkuchenhaus.

```
digitalWrite(ledgelb, HIGH);
delay(z1);
digitalWrite(ledgelb, LOW);
digitalWrite(ledgruen, HIGH);
delay(z1);
digitalWrite(ledgruen, LOW);
break;
case 1:
  digitalWrite(ledrot, HIGH);
  digitalWrite(ledorange, HIGH);
  digitalWrite(ledgelb, HIGH);
  digitalWrite(ledgruen, HIGH);
  delay(z1);
  digitalWrite(ledrot, LOW);
  digitalWrite(ledorange, LOW);
  digitalWrite(ledgelb, LOW);
  digitalWrite(ledgruen, LOW);
  delay(z1);
  break;
case 2:
  digitalWrite(rot, HIGH);
  delay(z1);
  digitalWrite(rot, LOW);
```

```
    digitalWrite(gruen, HIGH);
    delay(z1);
    digitalWrite(gruen, LOW);
    digitalWrite(blau, HIGH);
    delay(z1);
    digitalWrite(blau, LOW);
    break;
case 3:
    digitalWrite(rot, HIGH);
    digitalWrite(gruen, HIGH);
    digitalWrite(blau, HIGH);
    delay(z2);
    digitalWrite(rot, LOW);
    digitalWrite(gruen, LOW);
    digitalWrite(blau, LOW);
    delay(z3);
    break;
}
}
```

So funktioniert das Programm

Am Anfang des Programms werden wieder die Variablen für die Pins definiert und diese in der Prozedur `void setup()` als Ausgänge definiert. Die Variablen `z1`, `z2`, `z3` legen drei verschiedene Zeiten für die Blinkmuster fest:

- `z1` – Lauflicht und gleichmäßiges Blinken
- `z2` – Sehr kurze Leuchtzeit für das weiße Blitzen der RGB-LED
- `z3` – Lange Dunkelzeit für das weiße Blitzen der RGB-LED

Die Variable `n` wählt die verschiedenen Blinkmuster aus.

Bei jedem Drücken des Tasters wird die Variable `n` um 1 erhöht.

```
n++;
if(n > 3) {
    n = 0;
}
```

Erreicht `n` dabei einen Wert größer als 3, wird `n` wieder auf 0 gesetzt, da es nur vier verschiedene Blinkmuster, 0, 1, 2, 3, gibt.

Die Anweisung `switch() { case ... }` führt unterschiedliche Programmzeilen aus, abhängig davon, welchen Wert der Parameter in der Klammer hat. Hier folgen im Programm vier Blöcke für die vier möglichen Werte der Variable `n`. Jeder Block wird mit einer `break`-Anweisung beendet, bevor die nächste `case`-Zeile folgt.

Tag 17

Heute im Adventskalender

- Potentiometer

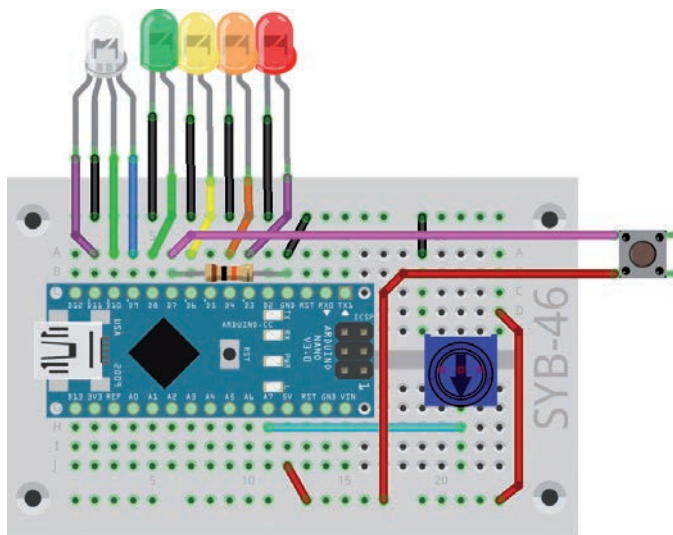
Potentiometer

Ein Potentiometer ist ein einstellbarer Widerstand. Damit lässt sich ein Spannungsteiler bauen, der eine beliebige Spannung zwischen 0 V und +5 V liefern kann. Ein analoger Pin am Nano wertet diesen Wert aus.

Aufgabe: Regelbares Lauflicht

Ein Drehknopf regelt die Geschwindigkeit des Lauflichts.

Bauteile: Nano-Board, Steckbrett, LED rot mit Vorwiderstand, LED gelb mit Vorwiderstand, LED grün mit Vorwiderstand, LED orange mit Vorwiderstand, RGB-LED mit Vorwiderstand, Potentiometer, Taster, 10-KOhm-Widerstand, 14 Verbindungskabel, 5 Drahtbrücken



fritzing

Potentiometer am Pin A7. Von der RGB-LED wird im Lauflicht nur die blaue Farbe verwendet.

```
//17blink
//Franzis Adventskalender für Arduino

int leds[] = {3, 4, 6, 8, 9};
int taste = 7;
int poti = A7;
int i = 0;
int n = 5;
bool r = false;

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

void loop() {
  if(digitalRead(taste) == HIGH) {
    while(digitalRead(taste) == HIGH) {
```

```
      digitalWrite(LED_BUILTIN, HIGH);
    }
    digitalWrite(LED_BUILTIN, LOW);
    r = !r;
  }
  if(r) {
    for(i = 0; i < n; i++) {
      digitalWrite(leds[i], HIGH);
      delay(analogRead(poti));
      digitalWrite(leds[i], LOW);
    }
  }
  else {
    for(i = 0; i < n; i++) {
      digitalWrite(leds[i], LOW);
    }
  }
}
```

So funktioniert das Programm

Am Anfang werden einige neue Variablen deklariert, `poti` bezeichnet den Analog-Pin, an dem das Potentiometer angeschlossen ist, `n` enthält die Anzahl der LEDs. Die analogen Eingänge brauchen nicht definiert zu werden.

Arrays sind eine Sonderform von Variablen, die mehrere Werte in einer festen Reihenfolge enthalten können. Arrays können beliebig lang sein.

```
int leds[] = {3, 4, 6, 8, 9};
```

Im Array `leds[]` sind die Pin-Nummern der LEDs eingetragen. Jede LED lässt sich über einen Index im Array ansteuern. `led[0]` bezeichnet die erste LED an Pin D3, `led[4]` bezeichnet die letzte LED an Pin D9, also die blaue Farbe der RGB-LED.

In der Prozedur `void setup()` werden nacheinander die Pins der LEDs im Array als Ausgänge definiert. Dafür verwenden wir wieder eine `for()`-Schleife. In diesem Fall nimmt der Schleifenzähler `i` nacheinander die Werte 0 bis 4 an und adressiert nacheinander die Pins aus dem Array `led[]` und setzt sie alle auf `OUTPUT`.

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

In der Endlosschleife `void loop()` wird wieder zuerst der Taster abgefragt und damit der Logikwert `r` umgeschaltet, der festlegt, ob das Lauflicht läuft oder nicht.

Das Lauflicht läuft auch in einer `for()` Schleife, die nacheinander die einzelnen LEDs ein- und wieder ausschaltet. Für die Wartezeit wird über die Funktion `analogRead()` der Wert des Potentiometers ausgelesen. Analoge Eingänge liefern Werte zwischen 0 und 1023, woraus sich beim Lauflicht Wartezeiten zwischen 0 und 1023 Millisekunden ergeben.

```
if(r) {
    for(i = 0; i < n; i++) {
        digitalWrite(leds[i], HIGH);
        delay(analogRead(poti));
        digitalWrite(leds[i], LOW);
    }
}
```

Hat der Logikwert `r` den Wert `false`, werden alle LEDs in einer `for()`-Schleife ausgeschaltet.

```
else {
    for(i = 0; i < n; i++) {
        digitalWrite(leds[i], LOW);
    }
}
```


Tag 18

Heute im Adventskalender

- 10-KOhm-Widerstand

LED mit Potentiometer dimmen

Zwei LEDs werden mit einem Potentiometer gleichzeitig gedimmt, wodurch die unterschiedliche Wirkung bei verschiedenen Farben zu sehen ist.

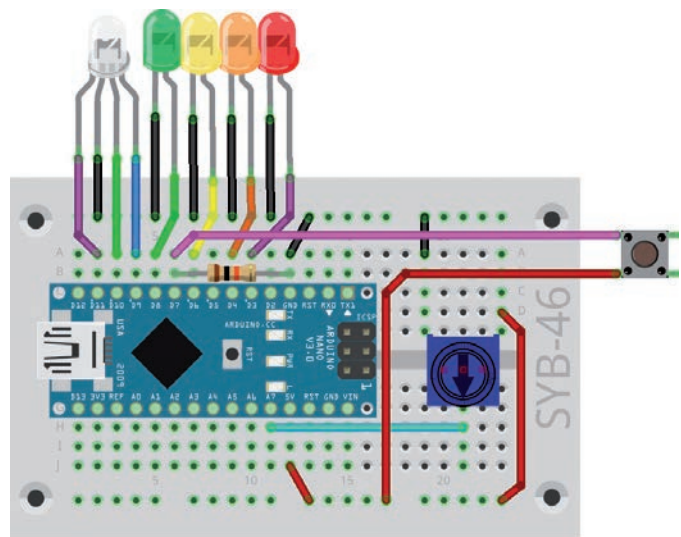
Bauteile: Nano-Board, Steckbrett, LED rot mit Vorwiderstand, LED gelb mit Vorwiderstand, LED grün mit Vorwiderstand, LED orange mit Vorwiderstand, RGB-LED mit Vorwiderstand, Potentiometer, Taster, 10-KOhm-Widerstand, 14 Verbindungskabel, 5 Drahtbrücken

```
//18pwm
//Franzis Adventskalender für Arduino

int ledrot = 3;
int ledgelb = 6;
int taste = 7;
int poti = A7;
bool r = false;

void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(ledrot, OUTPUT);
  pinMode(ledgelb, OUTPUT);
  pinMode(taste, INPUT);
}

void loop() {
  if(digitalRead(taste) == HIGH) {
    while(digitalRead(taste) == HIGH) {
      digitalWrite(LED_BUILTIN, HIGH);
    }
    digitalWrite(LED_BUILTIN, LOW);
    r = !r;
  }
  if(r) {
    analogWrite(ledrot, analogRead(poti) / 4);
    analogWrite(ledgelb, analogRead(poti) / 4);
  }
  else {
    digitalWrite(ledrot, LOW);
    digitalWrite(ledgelb, LOW);
  }
}
```



Der Schaltungsaufbau ist der gleiche wie am Vortag, es werden nur zwei der LEDs benötigt.

So funktioniert das Programm

In der Endlosschleife `void loop()` wird ständig der Wert am analogen Pin A7 ausgelesen, umgerechnet und auf den beiden PWM-Pins D3 und D4 wieder ausgegeben. Ein analoger Eingang liefert Werte zwischen 0 und 1023. Diese Werte müssen auf Werte zwischen 0 und 255 umgerechnet werden, um sie dann wieder als PWM-Wert an die LEDs auszugeben. Dazu wird der Wert des analogen Pins durch 4 geteilt.

```
if(r) {
  analogWrite(ledrot, analogRead(poti) / 4);
  analogWrite(ledgelb, analogRead(poti) / 4);
}
```


Tag 19

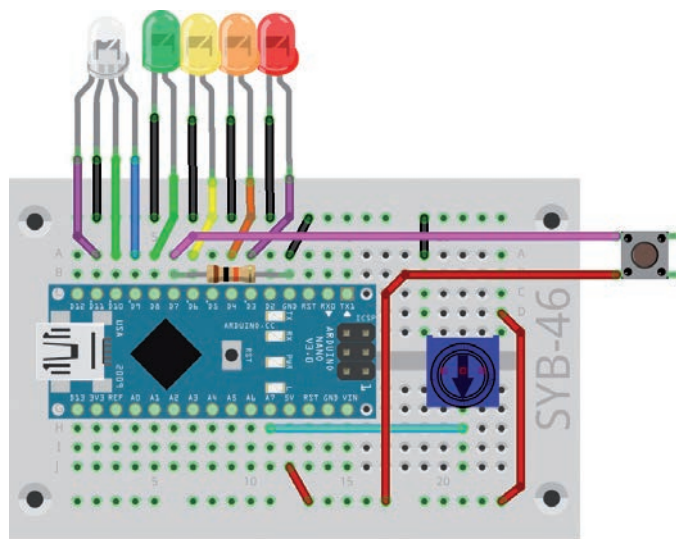
Heute im Adventskalender

- 20-MOhm-Widerstand

Aufgabe: Analoge Pegelanzeige

Um einen analogen Wert aber wirklich zu visualisieren, ist eine gedimmte LED nicht aussagekräftig genug, Helligkeiten werden sehr subjektiv wahrgenommen und hängen auch stark von der Umgebung und den verwendeten LEDs ab. An einer Pegelanzeige lassen sich analoge Werte viel deutlicher ablesen.

Bauteile: Nano-Board, Steckbrett, LED rot mit Vorwiderstand, LED gelb mit Vorwiderstand, LED grün mit Vorwiderstand, LED orange mit Vorwiderstand, RGB-LED mit Vorwiderstand, Potentiometer, Taster, 10-KOhm-Widerstand, 14 Verbindungskabel, 5 Drahtbrücken



Der Schaltungsaufbau ist der gleiche wie an den Tagen zuvor, Das Programm verwendet von der RGB-LED nur die blaue Farbe.

```
//19pegel
//Franzis Adventskalender für Arduino

int leds[] = {3, 4, 6, 8, 9};
int taste = 7;
int poti = A7;
int n = 5;
bool r = false;

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

void loop() {
  if(digitalRead(taste) == HIGH) {
    while(digitalRead(taste) == HIGH) {
      digitalWrite(LED_BUILTIN, HIGH);
    }
  }
}
```

```
digitalWrite(LED_BUILTIN, LOW);
r = !r;
}
if(r) {
  int s = analogRead(poti);
  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);
    }
  }
}
else {
  for(int i = 0; i < n; i++) {
    digitalWrite(leds[i], LOW);
  }
}
```

Arduino-Adventskalender

So funktioniert das Programm

Die Definition der Variablen und die Initialisierung der Pins laufen nach dem bekannten Schema. Das Programm verwendet ein neues Element der Arduino-C-Programmiersprache: die Funktion `map()`. Außerdem werden Schleifenzähler und andere Variablen, die nur für Zwischenrechnungen innerhalb einer Funktion genutzt werden, direkt dort deklariert, was das Programm auch noch einmal übersichtlicher macht.

In der Hauptschleife `void 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 s = analogRead(poti);  
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:

x	Umzurechnender Wert
<code>in_min</code>	Untere Grenze des Zahlenbereichs des eingegebenen Wertes
<code>in_max</code>	Obere Grenze des Zahlenbereichs des eingegebenen Wertes
<code>out_min</code>	Untere Grenze des Zahlenbereichs des ausgegebenen Wertes
<code>out_max</code>	Obere Grenze des Zahlenbereichs des ausgegebenen Wertes

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

Jetzt startet wieder eine Schleife, die über alle LEDs zählt. Bei jeder LED wird geprüft, ob der in `p` gespeicherte Pegelwert größer ist als die Nummer der LED. Ist das der Fall, wird die entsprechende LED eingeschaltet.

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

Ist der anzuzeigende Wert nicht größer als die Nummer der LED in der Schleife, wird die LED ausgeschaltet.

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

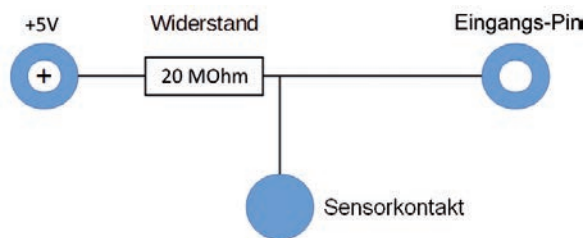
Tag 20

Heute im Adventskalender

- Knete

So funktionieren Sensorkontakte

Das Prinzip der Sensortasten ist einfach: Die verwendeten digitalen Eingänge sind über extrem hochohmige Widerstände (20 MOhm) mit +5 V verbunden, sodass ein schwaches, aber eindeutig als HIGH definiertes Signal anliegt. Ein Mensch, der nicht gerade frei in der Luft schwebt, ist meistens 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 entsprechenden digitalen Eingang auf LOW-Pegel.



Schaltschema der Kontaktsensoren

Wie hoch allerdings der Widerstand zwischen Hand und Masse wirklich ist, hängt von vielen Dingen ab, unter anderem von Schuhen und Fußboden. Barfuß in nassem Gras ist die Verbindung zur Masse der Erde am besten, aber auch auf Steinfußböden 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 ihn und den eigentlichen Sensor gleichzeitig, ist die Masseverbindung auf jeden Fall hergestellt.

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 20 cm langes Stück des Schaltdrahts ab, entfernen Sie an beiden Enden auf etwa 1 cm Länge die Isolierung und stecken Sie das eine Ende in ein Stück Knete. Stecken Sie das andere Ende in das Steckbrett.

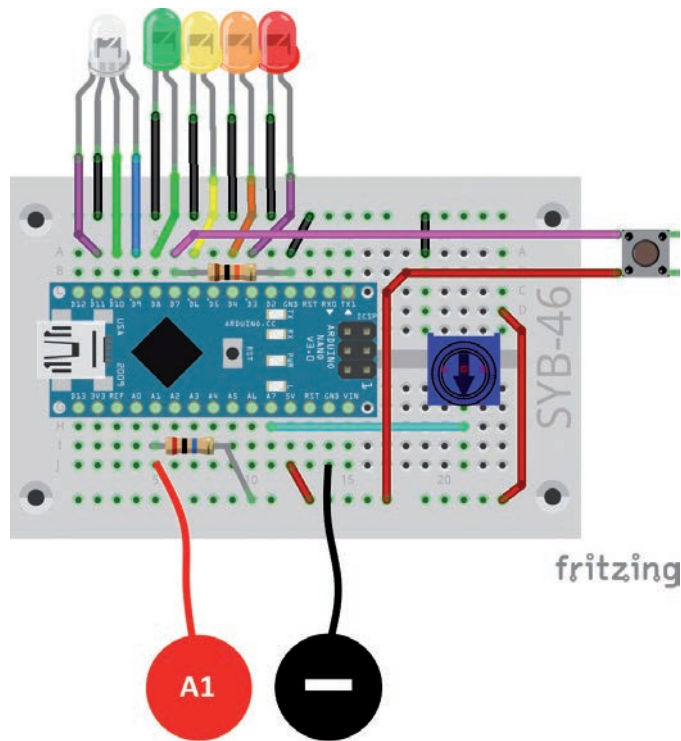
Die analogen Eingänge des Nano-Boards eignen sich besser für Sensorkontakte als digitale Eingänge. Je nach Umgebungsbedingungen sind Werte zwischen 200 und 500 gute Grenzwerte, um zwischen berührtem und nicht berührtem Sensorkontakt zu unterscheiden.

Sensorkontakt aus Knete

Vier LEDs laufen als Laufflicht. Zusätzlich kann über einen Sensorkontakt aus Knete die RGB-LED weiß aufblitzen.

Bauteile: Nano-Board, Steckbrett, LED rot mit Vorwiderstand, LED gelb mit Vorwiderstand, LED grün mit Vorwiderstand, LED orange mit Vorwiderstand, RGB-LED mit Vorwiderstand, Potentiometer, Taster, 10-KOhm-Widerstand, 20-MOhm-Widerstand, 2 Knete, 2 Schaltdraht, 14 Verbindungskabel, 5 Drahtbrücken

Arduino-Adventskalender



Sensorkontakte aus Knete an den Pins A1 und GND

```
//20sensor
//Franzis Adventskalender für Arduino

int leds[] = {3, 4, 6, 8};
int rot = 11;
int gruen = 10;
int blau = 9;
int taste = 7;
int poti = A7;
int sensor = A1;
int n = 4;
int s = 500;
int z = 10;
bool r = false;

void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(rot, OUTPUT);
  pinMode(gruen, OUTPUT);
  pinMode(blau, OUTPUT);
  pinMode(taste, INPUT);
  for(int i = 0; i < n; i++) {
    pinMode(leds[i], OUTPUT);
  }
}

void loop() {
  if(digitalRead(taste) == HIGH) {
```

```
while(digitalRead(taste) == HIGH) {
  digitalWrite(LED_BUILTIN, HIGH);
}
digitalWrite(LED_BUILTIN, LOW);
r = !r;
}
if(r) {
  for(int i = 0; i < n; i++) {
    digitalWrite(leds[i], HIGH);
    delay(analogRead(poti));
    digitalWrite(leds[i], LOW);
    if(analogRead(sensor) < s) {
      digitalWrite(rot, HIGH);
      digitalWrite(gruen, HIGH);
      digitalWrite(blau, HIGH);
      delay(z);
      digitalWrite(rot, LOW);
      digitalWrite(gruen, LOW);
      digitalWrite(blau, LOW);
    }
  }
}
else {
  for(int i = 0; i < n; i++) {
    digitalWrite(leds[i], LOW);
  }
}
}
```

So funktioniert das Programm

Zusätzlich zu den bereits bekannten Programmelementen wird am Anfang eine Variable `sensor` angelegt, die den Pin A1 bezeichnet, an dem der Sensorkontakt angeschlossen ist. Die Variable `s` enthält den Schwellwert, ab dem der Sensorkontakt als berührt gewertet wird. Sollte der Sensorkontakt fehlerhaft reagieren, passen Sie diesen Wert an. Die Variable `z` gibt die Blinkzeit des weißen Lichts auf der RGB-LED an. Die Blinkzeit des Lauflichts wird wieder über das Potentiometer eingestellt.

Innerhalb der Schleife, die das Lauflicht steuert, läuft nach jedem Umschalten eine Abfrage des Sensorkontakts. Liegt der Wert am analogen Eingang unter dem am Anfang festgelegten Schwellwert `s`, werden die drei Pins der RGB-LED kurz eingeschaltet. Die LED blitzt kurz weiß auf.

Tag 21

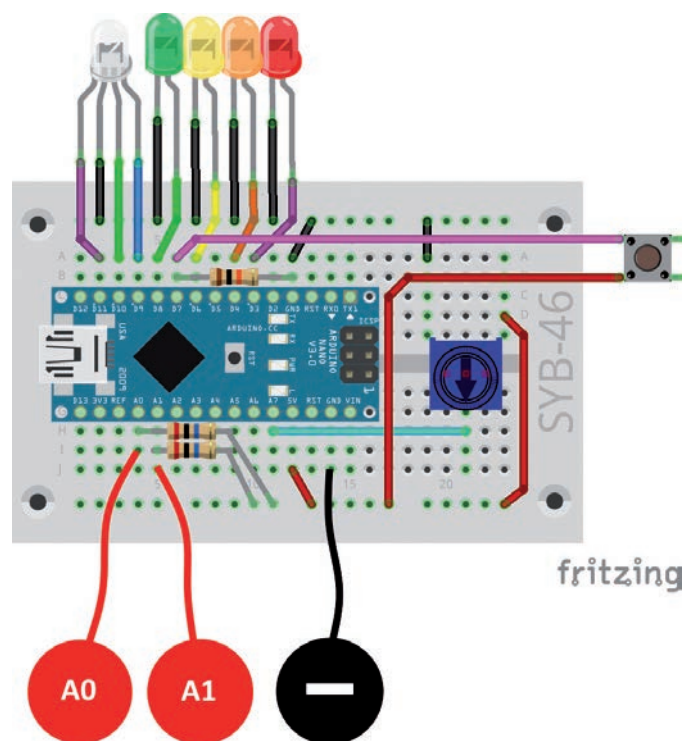
Heute im Adventskalender

- 20-MOhm-Widerstand

Aufgabe: Zwei Sensorkontakte aus Knete

Zwei Sensorkontakte steuern die rote und die blaue Farbe der RGB-LED. Bei Berühren eines Sensorkontaktes leuchtet die entsprechende Farbe kurz auf. Werden beide Sensorkontakte gleichzeitig berührt, erscheint die Mischfarbe Lila.

Bauteile: Nano-Board, Steckbrett, LED rot mit Vorwiderstand, LED gelb mit Vorwiderstand, LED grün mit Vorwiderstand, LED orange mit Vorwiderstand, RGB-LED mit Vorwiderstand, Potentiometer, Taster, 10-KOhm-Widerstand, 2 20-MOhm-Widerstände, 3 Knete, 3 Schaltdraht, 14 Verbindungskabel, 5 Drahtbrücken



Sensorkontakte aus Knete an den Pins A0, A1 und GND


```
//21sensor
//Franzis Adventskalender für Arduino

int leds[] = {3, 4, 6, 8};
int rot = 11;
int blau = 9;
int taste = 7;
int poti = A7;
int sensor0 = A0;
int sensor1 = A1;
int n = 4;
int s = 500;
int z = 10;
bool r = false;

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

void loop() {
  if(digitalRead(taste) == HIGH) {
    while(digitalRead(taste) == HIGH) {
      digitalWrite(LED_BUILTIN, HIGH);
    }
    digitalWrite(LED_BUILTIN, LOW);
    r = !r;
  }
  if(r) {
    for(int i = 0; i < n; i++) {
      digitalWrite(leds[i], HIGH);
      delay(analogRead(poti));
      digitalWrite(leds[i], LOW);
      if(analogRead(sensor0) < s) {
        digitalWrite(rot, HIGH);
        delay(z);
        digitalWrite(rot, LOW);
      }
      if(analogRead(sensor1) < s) {
        digitalWrite(blau, HIGH);
        delay(z);
        digitalWrite(blau, LOW);
      }
    }
  }
  else {
    for(int i = 0; i < n; i++) {
      digitalWrite(leds[i], LOW);
    }
  }
}
```

So funktioniert das Programm

Das Programm funktioniert weitgehend, wie das von gestern. Am Anfang werden die beiden Variablen `sensor0` und `sensor1` für die beiden Sensorkontakte an den analogen Eingängen A0 und A1 angelegt.

Am Ende jedes Schleifendurchlaufs für das Lauflicht werden jetzt beide Sensorkontakte nacheinander abgefragt.

```
if(analogRead(sensor0) < s) {
  digitalWrite(rot, HIGH);
  delay(z);
  digitalWrite(rot, LOW);
}
if(analogRead(sensor1) < s) {
  digitalWrite(blau, HIGH);
  delay(z);
  digitalWrite(blau, LOW);
}
```

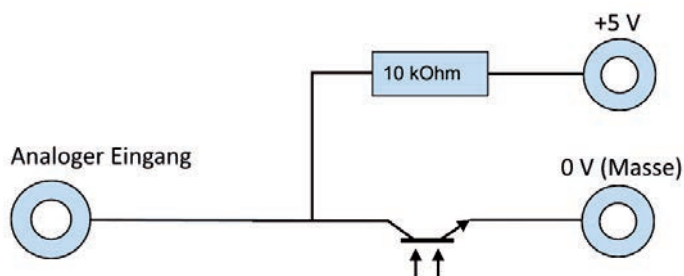
Tag 22

Heute im Adventskalender

- Fototransistor

Fototransistor

Ein Fototransistor ist ein lichtempfindliches Bauelement, das auf den ersten Blick wie eine transparente LED aussieht. Je nach Stärke des Lichteinfalls lassen sich mit der abgebildeten Schaltung an einem analogen Eingang des Nano-Boards unterschiedliche Werte erzielen. Je heller das Licht ist, das auf den Fototransistor fällt, desto geringer der Wert am analogen Eingang. Im Gegensatz zu LEDs wird beim Fototransistor der lange Anschluss mit Masse verbunden, nicht der kurze.

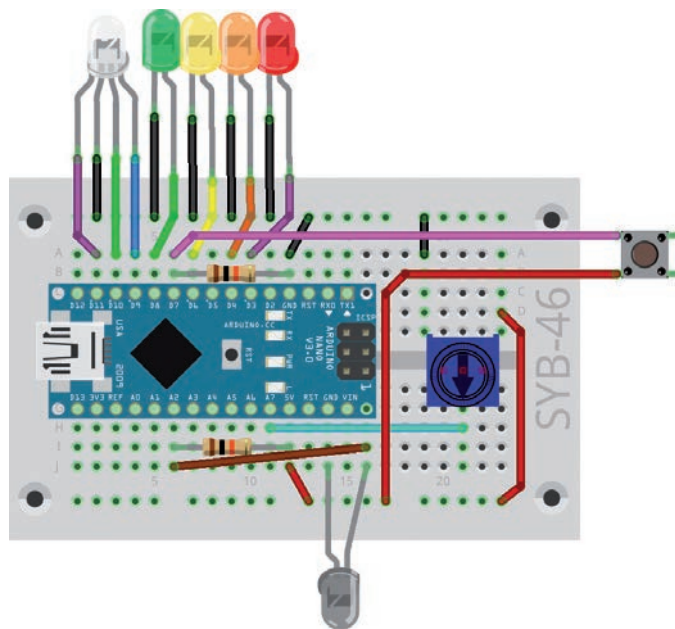


Schaltschema für einen Fototransistor

Licht mit Dämmerungsschalter

Die RGB-LED soll weiß leuchten, wenn es dunkel genug ist.

Bauteile: Nano-Board, Steckbrett, LED rot mit Vorwiderstand, LED gelb mit Vorwiderstand, LED grün mit Vorwiderstand, LED orange mit Vorwiderstand, RGB-LED mit Vorwiderstand, Taster, 2 10-KOHm-Widerstände, Potentiometer, Fototransistor, 14 Verbindungskabel, 6 Drahtbrücken



fritzing

Fototransistor am analogen Pin A2

```
//221licht
//Franzis Adventskalender für Arduino

int leds[] = {3, 4, 6, 8};
int rot = 11;
int gruen = 10;
int blau = 9;
int taste = 7;
int poti = A7;
int sensor = A2;
int n = 4;
int s = 800;
bool r = false;

void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(rot, OUTPUT);
  pinMode(gruen, OUTPUT);
  pinMode(blau, OUTPUT);
  pinMode(taste, INPUT);
  for(int i = 0; i < n; i++) {
    pinMode(leds[i], OUTPUT);
  }
}

void loop() {
  if(digitalRead(taste) == HIGH) {
    while(digitalRead(taste) == HIGH) {
      digitalWrite(LED_BUILTIN, HIGH);
    }
    digitalWrite(LED_BUILTIN, LOW);
    r = !r;
  }
  if(r) {
    for(int i = 0; i < n; i++) {
      digitalWrite(leds[i], HIGH);
      delay(analogRead(poti));
      digitalWrite(leds[i], LOW);
      if(analogRead(sensor) > s) {
        digitalWrite(rot, HIGH);
        digitalWrite(gruen, HIGH);
        digitalWrite(blau, HIGH);
      }
    }
  }
  else {
    digitalWrite(rot, LOW);
    digitalWrite(gruen, LOW);
    digitalWrite(blau, LOW);
  }
}
else {
  for(int i = 0; i < n; i++) {
    digitalWrite(leds[i], LOW);
  }
}
}
```

So funktioniert das Programm

Der Fototransistor liefert je nach Helligkeit einen analogen Wert zwischen 0 und 1023, der umso höher ist, je weniger Licht auf ihn fällt.

In jedem Durchlauf des Lauflichts wird der Pin A2 mit dem Fototransistor einmal ausgelesen. Liegt der analoge Wert höher als der in der Variable `s` gespeicherte Schwellwert, ist es dem Programm dunkel genug und alle drei Farben der RGB-LED werden eingeschaltet. Liegt er darunter, wird die RGB-LED ausgeschaltet.

Tag 23

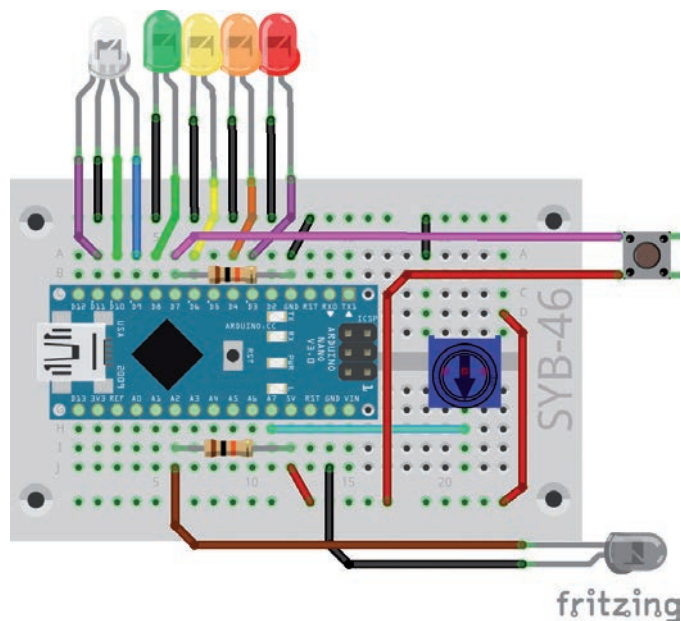
Heute im Adventskalender

- 2 Verbindungskabel

Dämmerungsschalter auf dem Dach und Klingeltaster steuern Beleuchtung

Bei Dunkelheit blinkt die RGB-LED in verschiedenen Farben. Der Taster dient als Klingelknopf und nicht mehr als Ein-/Ausschalter für das ganze Programm. Nach Drücken des Tasters läuft das Lauflicht einige Male durch. Die Blinkgeschwindigkeit lässt sich weiterhin mit dem Potentiometer einstellen.

Bauteile: Nano-Board, Steckbrett, LED rot mit Vorwiderstand, LED gelb mit Vorwiderstand, LED grün mit Vorwiderstand, LED orange mit Vorwiderstand, RGB-LED mit Vorwiderstand, Taster, 2 10-KOhm-Widerstände, Potentiometer, Foto-transistor, 16 Verbindungskabel, 5 Drahtbrücken



Fototransistor und LEDs auf dem Lebkuchenhaus

```
//23klinge1
//Franzis Adventskalender für Arduino

int leds[] = {3, 4, 6, 8};
int rot = 11;
int gruen = 10;
int blau = 9;
int taste = 7;
int poti = A7;
int sensor = A2;
int n = 4;
int s = 800;
int w = 4;

void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(rot, OUTPUT);
  pinMode(gruen, OUTPUT);
  pinMode(blau, OUTPUT);
  pinMode(taste, INPUT);
  for(int i = 0; i < n; i++) {
    pinMode(leds[i], OUTPUT);
  }
}

void loop() {
  if(analogRead(sensor) > s) {
    digitalWrite(rot, HIGH);
    delay(analogRead(poti));
    digitalWrite(gruen, HIGH);
    delay(analogRead(poti));
    digitalWrite(rot, LOW);
    delay(analogRead(poti));
    digitalWrite(blau, HIGH);
    delay(analogRead(poti));
    digitalWrite(gruen, LOW);
    delay(analogRead(poti));
    digitalWrite(rot, HIGH);
    delay(analogRead(poti));
    digitalWrite(blau, LOW);
  }
  else {
    digitalWrite(rot, LOW);
    digitalWrite(gruen, LOW);
    digitalWrite(blau, LOW);
  }
  if(digitalRead(taste) == HIGH) {
    while(digitalRead(taste) == HIGH) {
      digitalWrite(LED_BUILTIN, HIGH);
    }
    digitalWrite(LED_BUILTIN, LOW);
    for(int j = 0; j < w; j++) {
      for(int i = 0; i < n; i++) {
        digitalWrite(leds[i], HIGH);
        delay(analogRead(poti));
        digitalWrite(leds[i], LOW);
      }
    }
  }
}
```

Arduino-Adventskalender

So funktioniert das Programm

Dieses Programm verwendet einige zusätzliche Variablen:

`leds[]` – Array mit den Pinnummern der einzelnen LEDs

`rot` – Pinnummer der roten Farbe der RGB-LED

`gruen` – Pinnummer der grünen Farbe der RGB-LED

`blau` – Pinnummer der blauen Farbe der RGB-LED

`taste` – Pinnummer des Tasters

`poti` – Analoge Pinnummer des Potentiometers

`sensor` – Analoge Pinnummer des Fototransistors

`n` – Anzahl der LEDs im Lauflicht

`s` – Schwellwert für Fototransistor

`w` – Wiederholungen des Lauflichts nach Druck auf den Taster

In diesem Programm wird in jedem Durchlauf der Endlosschleife `void loop ()` der analoge Wert des Fototransistors ausgelesen. Ist er größer als der in der Variable `s` angegebene Schwellwert, leuchten nacheinander verschiedene Farben der RGB-LED, wobei die Pausen zwischen den Farbwechseln über das Potentiometer eingestellt werden.

Am Ende eines kompletten Durchlaufs des Farbwechsels wird der Taster abgefragt. Wurde er gedrückt, läuft das Lauflicht mit der über das Potentiometer eingestellten Geschwindigkeit. Die Variable `w` gibt die Anzahl der Wiederholungen an. Danach startet die Endlosschleife `void loop ()` erneut mit dem Farbwechsel auf der RGB-LED.

Tag 24

Heute im Adventskalender

- Piezo-Summer

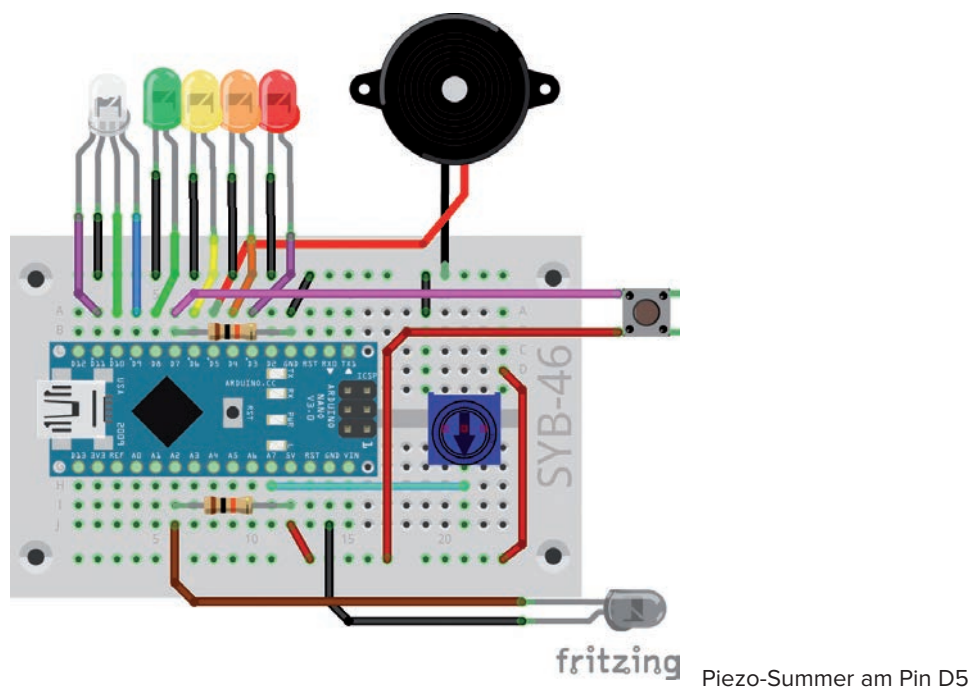
Piezo-Summer

Der heute im Adventskalender enthaltene Piezo-Summer macht elektrische Schwingungen hörbar. Legt man eine pulsierende Gleichspannung zwischen die beiden Pole des Summers, wird er in Schwingung versetzt. Je nach Frequenz sind einzelne Klicks oder ein durchgängiger Ton zu hören. Frequenzen von wenigen Hertz (Schwingungen pro Sekunde) nimmt das menschliche Ohr noch als einzelne Töne wahr, Frequenzen zwischen etwa 20 Hertz und 16 kHz werden als durchgehender Ton unterschiedlicher Tonhöhe wahrgenommen. Piezo-Summer werden an digitalen Ausgängen des Nano angeschlossen.

Weihnachtshaus mit Klingel und Dämmerungsschalter

Beim Drücken auf den Klingelknopf leuchten alle vier LEDs gleichzeitig, und ein Weihnachtslied wird abgespielt.

Bauteile: Nano-Board, Steckbrett, LED rot mit Vorwiderstand, LED gelb mit Vorwiderstand, LED grün mit Vorwiderstand, LED orange mit Vorwiderstand, RGB-LED mit Vorwiderstand, Taster, 2 10-KOHm-Widerstände, Potentiometer, Fototransistor, Piezo-Summer, 16 Verbindungskabel, 5 Drahtbrücken



```
//24weihnachten  
//Franzis Adventskalender für Arduino
```

```
#include "pitches.h"
```

```
int leds[] = {3, 4, 6, 8};  
int n = 4;  
int rot = 11;  
int gruen = 10;  
int blau = 9;  
int taste = 7;  
int summer = 5;  
int poti = A7;  
int sensor = A2;  
int s = 800;
```

Arduino-Adventskalender

```
int noten[] = {NOTE_C3, NOTE_C3, NOTE_G3, NOTE_G3, NOTE_A3, NOTE_A3, NOTE_G3,
              NOTE_F3, NOTE_F3, NOTE_E3, NOTE_E3, NOTE_D3, NOTE_C3,
              NOTE_G3, NOTE_G3, NOTE_F3, NOTE_F3, NOTE_E3, NOTE_E3, NOTE_D3,
              NOTE_G3, NOTE_G3, NOTE_F3, NOTE_F3, NOTE_E3, NOTE_E3, NOTE_D3,
              NOTE_C3, NOTE_C3, NOTE_G3, NOTE_G3, NOTE_A3, NOTE_A3, NOTE_G3,
              NOTE_F3, NOTE_F3, NOTE_E3, NOTE_E3, NOTE_D3, NOTE_C3};

int dauer[] = {4, 4, 4, 4, 4, 4, 2,
              4, 4, 4, 4, 2, 2,
              4, 4, 4, 4, 4, 4, 2,
              4, 4, 4, 4, 4, 4, 2,
              4, 4, 4, 4, 4, 4, 2,
              4, 4, 4, 4, 2, 2};

int m = 40;
float p = 1.30;

void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(rot, OUTPUT);
  pinMode(gruen, OUTPUT);
  pinMode(blau, OUTPUT);
  pinMode(summer, OUTPUT);
  pinMode(taste, INPUT);
  for(int i = 0; i < n; i++) {
    pinMode(leds[i], OUTPUT);
  }
}

void loop() {
  if(analogRead(sensor) > s) {
    digitalWrite(rot, HIGH);
    delay(analogRead(poti));
    digitalWrite(gruen, HIGH);
    delay(analogRead(poti));
    digitalWrite(rot, LOW);
    delay(analogRead(poti));
    digitalWrite(blau, HIGH);
    delay(analogRead(poti));
    digitalWrite(gruen, LOW);
    delay(analogRead(poti));
    digitalWrite(rot, HIGH);
    delay(analogRead(poti));
    digitalWrite(blau, LOW);
  }
  else {
    digitalWrite(rot, LOW);
    digitalWrite(gruen, LOW);
    digitalWrite(blau, LOW);
  }
  if(digitalRead(taste) == HIGH) {
    while(digitalRead(taste) == HIGH) {
      digitalWrite(LED_BUILTIN, HIGH);
    }
    digitalWrite(LED_BUILTIN, LOW);
    for(int i = 0; i < n; i++) {
      digitalWrite(leds[i], HIGH);
    }
    for(int i = 0; i < m; i++) {
      tone(summer, noten[i], analogRead(poti) / dauer[i]);
      delay((analogRead(poti) / dauer[i]) * p);
    }
  }
}
```

```

        noTone(summer);
    }
    for(int i = 0; i < n; i++) {
        digitalWrite(leds[i], LOW);
    }
}
}

```

So funktioniert das Programm

Im Programm wird eine zusätzliche Datei mit den Definitionen der Konstanten für die Notenwerte eingebunden.

```
#include "pitches.h"
```

Anschließend werden die Variablen definiert. Dieses Programm verwendet wieder einige zusätzliche, aber auch zahlreiche bereits bekannte Variablen.

`leds[]` – Array mit den Pinnummern der einzelnen LEDs

`n` – Anzahl der LEDs im Lauflicht

`rot` – Pinnummer der roten Farbe der RGB-LED

`gruen` – Pinnummer der grünen Farbe der RGB-LED

`blau` – Pinnummer der blauen Farbe der RGB-LED

`taste` – Pinnummer des Tasters

`summer` – Pinnummer des Piezo-Summers

`pot i` – Analoge Pinnummer des Potentiometers

`sensor` – Analoge Pinnummer des Fototransistors

`s` – Schwellwert für Fototransistor

`noten[]` – Array mit den Notenwerten für das Lied. Diese Noten sind in der eingebundenen Datei `pitches.h` definiert.

`dauer[]` – Array mit den Tonlängen für das Lied. 4 steht für Viertelnote, 2 für halbe Note.

`m` – Anzahl der Töne im Lied

`p` – Zeitfaktor für Pausen

Auch in diesem Programm wird in jedem Durchlauf der Endlosschleife `void loop ()` der analoge Wert des Fototransistors ausgelesen. Ist er größer als der in der Variable `s` angegebene Schwellwert, leuchten nacheinander verschiedene Farben der RGB-LED, wobei die Pausen zwischen den Farbwechseln wie im Programm von gestern über das Potentiometer eingestellt werden.

Am Ende eines kompletten Durchlaufs des Farbwechsels wird der Taster abgefragt. Wurde er gedrückt, werden alle vier LEDs eingeschaltet.

```

    digitalWrite(LED_BUILTIN, LOW);
    for(int i = 0; i < n; i++) {
        digitalWrite(leds[i], HIGH);
    }
}

```

Arduino-Adventskalender

Anschließend läuft eine weitere `for()`-Schleife, die nacheinander die Töne des Liedes abspielt. Die hier verwendete Funktion `tone()` erfordert drei Parameter.

Pin	Pin-Nummer des Piezo-Summers	<code>summer</code>
Ton	Frequenz des Tons	<code>noten[i]</code> Ton aus dem Array <code>noten[]</code> , das der Reihe nach alle Töne des Liedes enthält.
Sauer	Dauer des Tons in Millisekunden	<code>analogRead(poti) / dauer[i]</code> Mit dem Potentiometer eingestellte Zeit geteilt durch Notenlänge.

```
for(int i = 0; i < m; i++) {  
  tone(summer, noten[i], analogRead(poti) / dauer[i]);  
  delay((analogRead(poti) / dauer[i]) * p);  
  noTone(summer);  
}
```

Da das Programm, während der Ton abgespielt wird, keine Pause macht, muss noch eine Pause eingefügt werden, damit der nächste Ton das Ende des vorhergehenden abwartet. Die Pause errechnet sich aus der Tonlänge und einem Pausenfaktor `p`, in der Grundeinstellung 1.3, der dafür sorgt, dass die Töne deutlich voneinander abgesetzt werden.

Zum Schluss schaltet die Funktion `noTone()` den Piezo-Summer ab. Diese Funktion muss immer aufgerufen werden, bevor ein anderer Ton gespielt werden kann.

Nachdem das ganze Lied abgespielt wurde, werden die vier LEDs wieder ausgeschaltet.

```
for(int i = 0; i < n; i++) {  
  digitalWrite(leds[i], LOW);  
}
```

Jetzt startet die Endlosschleife neu, lässt die RGB-LED in verschiedenen Farben leuchten, vorausgesetzt, es ist dunkel genug, und wartet auf den nächsten Tastendruck.

Frohe Weihnachten!