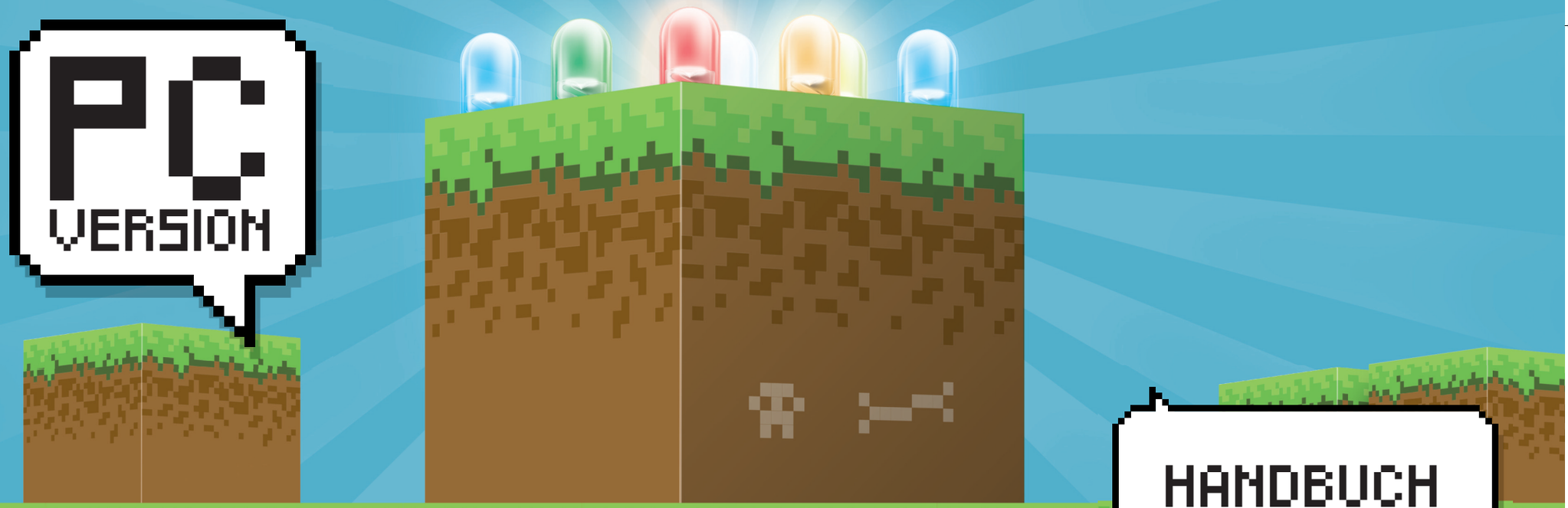
FRANZIS

ADVENTSKALENDER PROGRAMMIEREN MIT MINECRAFT™

UND WINDOWS-JAVA

powered by MAKER FACTORY



HANDBUCH

FRANZIS

Liebe Kunden!

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.

Achtung! 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.

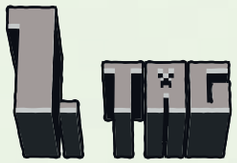
GTIN 4019631150257

Adventskalender 2018 für PC (Java-Edition)

Minecraft ist eines der beliebtesten Computerspiele weltweit – und das, obwohl es kein wirkliches Spielziel hat. Gerade die offene Spielwelt, in der jeder für sich Landschaften bauen und eigene Träume verwirklichen kann, fasziniert über 100 Millionen Spieler weltweit. Der Spieltitel setzt sich übrigens aus den beiden englischen Begriffen *Mine* (= Bergbau, Rohstoffabbau) und *craft* (= verarbeiten) zusammen. Und genau darum geht es in Minecraft: Es gilt, Rohstoffe aus der Landschaft abzubauen und sie zu neuen Landschaften oder Objekten zu verarbeiten.

Dieser Adventskalender enthält jeden Tag ein kleines Experiment, bei dem Elektronik gesteuert wird. Nach den ersten grundlegenden Versuchen erfolgt diese Steuerung später aus Minecraft heraus.

Minecraft, das beliebte Weltenbauerspiel, ist in verschiedenen Versionen für unterschiedliche Systemplattformen verfügbar. Um mit Minecraft Elektronik über eine Arduino-kompatible Platine zu steuern, wird die klassische Java-Edition für Windows benötigt. Die etwas preiswertere Windows-10-Edition bietet keine Möglichkeit, externe Programme wie das hier verwendete AMI anzusteuern. Die Java-Editionen für Linux und Mac OS können ebenfalls nicht genutzt werden, da AMI auf diesen Plattformen nicht verfügbar ist.



Da PCs keine einfachen Anschlüsse für LEDs oder sonstige Elektronik haben, verwenden wir einen Arduino-kompatiblen Mikrocontroller als Schnittstelle. Der PC kommuniziert über das USB-Kabel mit einem Programm auf dem Mikrocontroller, der dann die LEDs zum Leuchten bringt. Das Programmieren von Mikrocontrollern war früher nur etwas für Ingenieure und Informatiker. Die Arduino-Plattform ermöglicht dank übersichtlicher Hardware und einfach zu verstehender Software jedem den Einstieg in die Mikrocontrollertechnik.

Die Arduino-Plattform, eine Serie kleiner, kostengünstiger Platinen auf der Basis von Atmel-Mikrocontrollern, wurde ursprünglich für Bastler und Künstler entwickelt, die damit interaktive elektronische Objekte schaffen können, und ist inzwischen in der Maker-Szene zum allgemeinen Standard für mikrocontrollergesteuerte Hardwareprojekte geworden.

Was ist ein Mikrocontroller?

Ein Mikrocontroller ist ein kleiner (mikro = winzig) programmierbarer Elektronikchip, der zur Steuerung (to control = steuern) von unterschiedlichster Hardware verwendet wird. Mikrocontroller sind heute in den meisten Geräten eingebaut, die eine elektronische Steuerung haben, unter anderem in Küchengeräten, Elektronikspielzeug, Alarmanlagen, Heizungsanlagen oder Fahrradtachos. Im Gegensatz zu einem „echten" Computer haben sie kein Betriebssystem und auch keine Festplatte. Prozessor, Programmspeicher und Datenspeicher befinden sich alle auf demselben Chip, der zusätzlich auch Anschlüsse zur Steuerung externer Hardware wie LEDs, LCD-Anzeigen, Tastern und Sensoren hat.

Heute im Adventskalender

- Nano-Board (Arduino-kompatible Platine)

Nano-Board

Die Arduino-Plattform bietet mittlerweile eine große Vielfalt an Platinen für unterschiedliche Anwendungszwecke. Der Adventskalender enthält heute eine zum Arduino-Nano-Standard kompatible Platine, die direkt auf ein Steckbrett gesteckt werden kann, um weitere Elektronik anzuschließen.

Nano vorbereiten

Um den Nano in Betrieb zu nehmen, braucht man:

- PC mit Windows
- MicroUSB-Kabel
- Arduino-IDE, Treiber und Verbindungssoftware AMI (Arduino Minecraft Interface)

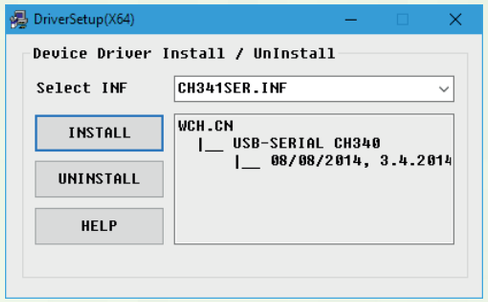
PC und Nano-Board werden über ein MicroUSB-Kabel verbunden. Sie brauchen sich nicht extra ein solches Kabel zu besorgen, fast alle modernen Smartphones verwenden diesen Steckertyp. Das Kabel wird zur Stromversorgung und gleichzeitig zur Datenübertragung verwendet.

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.

Softwareinstallation in Kürze

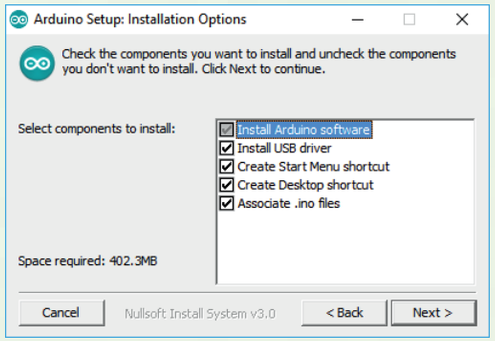
Hier die Softwareinstallation in sieben Schritten:

1. Laden Sie die Beispielprogramme und den Gerätetreiber von *http://bit.ly/mf-adventskalender-18-pc* herunter.
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 die Treiberinstallation mit der Datei `CH341SER.EXE` aus dem Unterverzeichnis `Treiber`. Zur Installation sind Admin-Rechte 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.



Installation des Gerätetreibers

5. Für die Programmierung des Arduino liefert der Hersteller eine Entwicklungsumgebung (IDE), in der man die Programme, die bei Arduino als Sketch bezeichnet werden, in einer C-ähnlichen Programmiersprache schreiben kann. Wenn Sie Windows 10 verwenden, installieren Sie aus dem Windows Store die App *Arduino-IDE*, was deutlich komfortabler ist als der klassische Weg, der auf älteren Windows-Versionen weiterhin genutzt werden muss. Bei Windows 10 können Sie sich den nächsten Schritt dann sparen und erhalten über den Windows Store auch automatisch Updates.
6. Laden Sie den Windows Installer für die aktuelle Version der Arduino-IDE von *www.arduino.cc/en/Main/Software* herunter. Achten Sie dabei darauf, dass im Dialogfeld *Installation Options* alle Häkchen gesetzt sind. Je nach Windows-Konfiguration ist möglicherweise eine Bestätigung der Benutzerkontensteuerung erforderlich.



Installation der Arduino-IDE

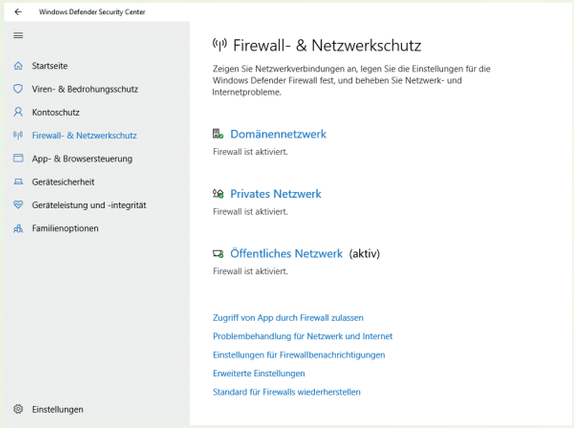
7. Kopieren Sie noch die Verbindungssoftware `AMI.exe` aus dem Unterverzeichnis `AMI` des Downloads in das Verzeichnis `%APPDATA%\.minecraft\logs`. Dieses Verzeichnis finden Sie im Explorer (in den meisten Fällen) unter `c:\Users\Benutzername\AppData\Roaming\.minecraft\logs`. Es ist nur zu sehen,

wenn im Menüband *Ansicht* des Explorers der Schalter *Ausgeblendete Elemente* eingeschaltet ist. Die übrigen Dateien aus diesem Verzeichnis benötigen Sie nicht. Um das Programm später einfacher starten zu können, legen Sie im Explorer per Rechtsklick auf die Datei über den Menüpunkt *Senden an/Desktop (Verknüpfung erstellen)* ein Desktopsymbol an.

Firewall blockiert Kommunikation mit dem Nano-Board

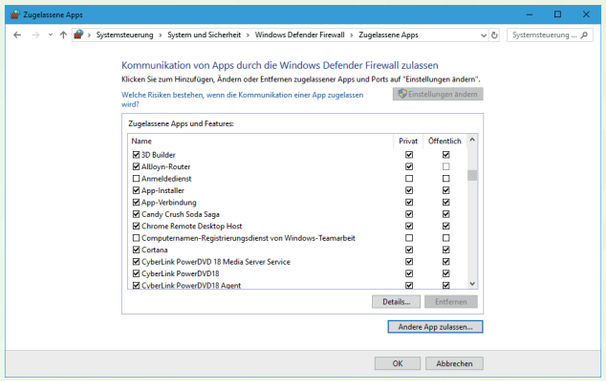
Die Windows-Defender-Firewall blockiert die Kommunikation der AMI-Software mit dem Nano-Board, ohne dass eine Fehlermeldung angezeigt wird. Um das zu vermeiden, lassen Sie das Programm vor der ersten Verwendung in der Firewall zu.

1. Klicken Sie im Windows Defender Security Center unter *Firewall & Netzwerkschutz* auf den Link *Zugriff von App durch Firewall zulassen*.



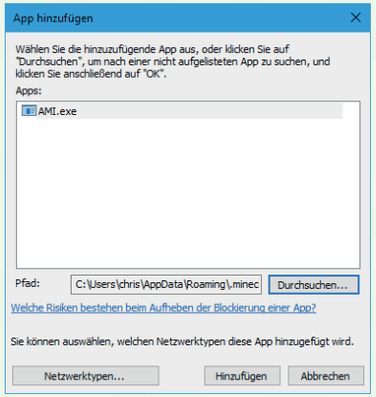
Firewall & Netzwerkschutz im Windows Defender Security Center

2. Klicken Sie im nächsten Fenster auf den Button *Einstellungen ändern* und bestätigen Sie die Anfrage der Benutzerkontensteuerung.
3. Klicken Sie dann auf *Andere App zulassen*.



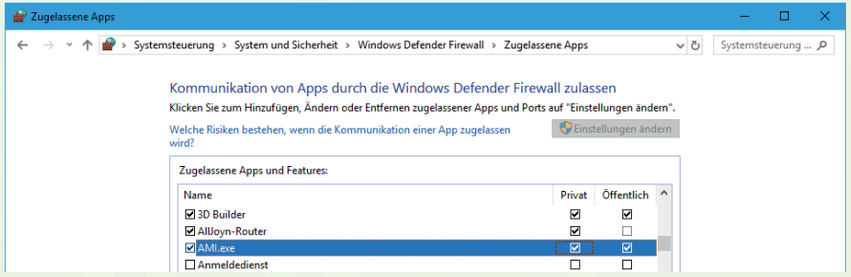
Programme in der Windows Defender Firewall zulassen

4. Wählen Sie im nächsten Fenster das Programm `%APPDATA%\.minecraft\logs\.AMI.exe` und klicken Sie auf *Hinzufügen*.



Das Programm AMI.exe auswählen

5. Setzen Sie in der Programmliste in der Zeile *AMI.exe* alle drei Häkchen, um den Zugriff zuzulassen, und bestätigen Sie mit *OK*.



Alle Häkchen für AMI.exe setzen

Nano-Board ausschalten

Das Nano-Board braucht nicht wie ein PC heruntergefahren zu werden. Sie können einfach den Stecker ziehen, und es schaltet sich ab. Beim nächsten Start bleibt das zuletzt installierte Programm installiert und startet sofort wieder. Das Gleiche passiert, wenn man den Reset-Taster in der Mitte des Nano-Boards drückt.

Die erste LED blinkt auf dem Nano-Board

Richtig interessant wird ein Arduino erst durch angeschlossene Hardware, und wenn es nur eine LED ist, die blinkt. Für erste Experimente hat das Nano-Board sogar eine LED direkt auf der Platine, so dass Sie nicht einmal eine Schaltung aufbauen müssen, sondern sofort loslegen können.

Arduino-Programme werden als Sketche bezeichnet. Die Arduino-IDE verwaltet alle Sketche Java-typisch in einzelnen Verzeichnissen, die jeweils genau den gleichen Namen wie der jeweilige Sketch haben müssen. Diese Namen dürfen nicht mit einer Ziffer beginnen. Sie finden das Programm des heutigen Tages im Verzeichnis `blink01` im Download.

1. Bevor Sie mit dem Programmieren beginnen können, muss eine Verbindung zwischen PC und Nano hergestellt werden. Starten Sie dazu die Arduino-IDE über das Windows-Startmenü.
2. Der Treiber wird jetzt automatisch installiert und simuliert über USB einen seriellen Port, mit dem das Nano-Board verbunden ist. Wählen Sie im Menü *Werkzeuge >Port*. Hier wird in den meisten Fällen nur ein einziger serieller Port angezeigt. Setzen Sie das Häkchen. Sollten mehrere COM-Ports vorhanden sein, starten Sie den Geräte-Manager. Der Port *USB-SERIAL CH340* unter *Anschlüsse* ist der richtige.

Serielle Anschlüsse im Geräte-Manager

3. Wählen Sie anschließend über den Menüpunkt *Werkzeuge > Board* den *Arduino Nano*, wenn dieser nicht bereits automatisch erkannt wurde. Wählen Sie über *Werkzeuge > Prozessor* den *ATmega328*.

Schreiben Sie den abgebildeten Sketch in der Arduino-IDE und speichern Sie ihn als `blink01.ino`. Beim Speichern wird automatisch ein gleichnamiges Unterverzeichnis angelegt. Sie können auch einfach den Sketch

aus dem entpackten Download, nicht direkt aus dem Zip-Archiv, über *Datei > Öffnen* in der Arduino-IDE öffnen. Die Arduino-IDE verwendet automatisch eine Syntaxhervorhebung, die Variablen, Funktionsnamen und Parameter in unterschiedlichen Farben darstellt.

Die eingebaute LED auf dem Nano-Board blinkt

Klicken Sie auf das Symbol *Hochladen* in der oberen Symbolleiste, das Symbol mit dem Pfeil nach rechts. Jetzt wird der Sketch kompiliert und anschließend auf das angeschlossene Nano-Board übertragen. Sobald der Upload abgeschlossen ist, beginnt die orangefarbene, mit *LED* gekennzeichnete LED auf dem Nano-Board zu blinken.

Die LEDs TX und RX

Die beiden mit *TX* und *RX* bezeichneten LEDs stehen für *Transmit* (Senden) und *Receive* (Empfangen). Sie blinken bei der Datenübertragung über das USB-Kabel.

So funktioniert das Programm

Dieses einfache Beispiel zeigt noch ganz ohne Minecraft die grundlegenden Methoden der Arduino-Programmierung.

```
int led1 = 13;
```

Diese Zeile definiert die Variable `led1`, in der die Nummer des verwendeten LED-Pins steht. Die auf dem Arduino eingebaute LED entspricht dem Pin D13 der Steckerleiste. Variablen müssen immer einen Typ haben. Variablen vom Typ `int` werden zum Speichern ganzer Zahlen oder logischer Werte benutzt.

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

Jeder Sketch besteht aus mindestens zwei Prozeduren. Die Prozedur `void setup()` wird beim Start genau einmal durchlaufen. Hier werden üblicherweise die Pins initialisiert. Das Wort `void` bedeutet, dass diese Prozedur im Gegensatz zu einer Funktion keinen Wert zurückgibt. Inhalte von Prozeduren, Schleifen und anderen zusammenhängenden Strukturen werden in geschweifte Klammern eingeschlossen.

```
pinMode(led1, OUTPUT);
```

Diese Zeile definiert den in der Variable `led1` definierten Pin 13 als Ausgang. Alle mit *D* gekennzeichneten Pins auf dem Nano-Board können als Eingänge oder Ausgänge definiert werden und die Werte `HIGH` (logisches Ein) und `LOW` (logisches Aus) annehmen. Digitale Ausgänge werden für LEDs, LCD-Displays und andere digitale Ausgabegeräte verwendet, Eingänge für Taster und andere digitale Eingabegeräte, die die Signale `HIGH` oder `LOW` liefern.

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

Die Prozedur `void loop()` startet nach Abschluss von `void setup()` und läuft dann mit endloser Wiederholung, bis man den Stecker zieht oder die Reset-Taste drückt.

```
digitalWrite(led1, HIGH);
```

Die Funktion `digitalWrite()` schreibt einen digitalen Wert auf den angegebenen Pin, der zuvor als Ausgang definiert worden sein muss. In diesem Fall schaltet sie die LED ein. Digitale Ausgänge liefern mit dem Wert `HIGH` eine Spannung von +5 V, bei `LOW` liefern sie 0 V.

```
delay(100);
```

Die Funktion `delay()` bewirkt eine Wartezeit von 100 Millisekunden, bevor das Programm weiterläuft.

```
digitalWrite(led1, LOW);
```

Jetzt wird die LED wieder ausgeschaltet.

```
delay(100);
```

Danach wartet das Programm wieder 100 Millisekunden. Danach ist die Prozedur `void loop()` beendet und startet automatisch wieder. Die LED blinkt im Rhythmus von 100 Millisekunden.

Für Ihre Notizen

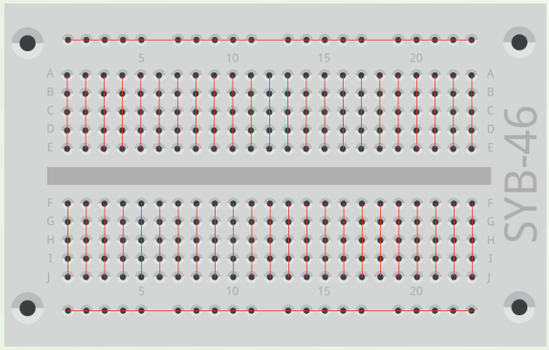
Heute im Adventskalender

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

Steckbrett

Für den schnellen Aufbau elektronischer Schaltungen ohne Löten enthält der Adventskalender ein Steckbrett. Hier können elektronische Bauteile direkt in ein Lochraster gesteckt werden.

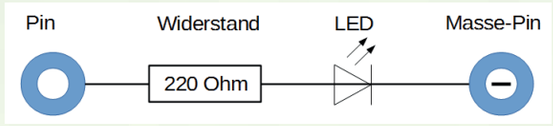
Bei diesem Steckbrett sind die Kontakte in den äußeren Längsreihen jeweils alle miteinander verbunden. Diese Kontaktreihen werden oft als Masseleiste bei der Stromversorgung der Schaltungen genutzt. In den anderen Kontaktreihen sind jeweils fünf Kontakte (A bis E und F bis J) quer miteinander verbunden,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



Die Verbindungen auf dem Steckbrett.

LEDs

LEDs (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 der zugehörige Draht ist ebenfalls etwas länger. Außerdem sind die meisten LEDs auf der Minusseite abgeflacht, vergleichbar mit einem Minuszeichen. Auch leicht zu merken: Kathode = kurz = Kante.

Vorsichtsmaßnahmen

Auf keinen Fall sollte man 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 Mikrocontroller komplett zerstören – zumindest theoretisch. Die Arduino-kompatiblen Platinen sind erstaunlich resistent gegen Schaltungsfehler. Verbindet man über eine LED zwei Pins miteinander, muss immer ein Vorwiderstand dazwischengeschaltet werden.

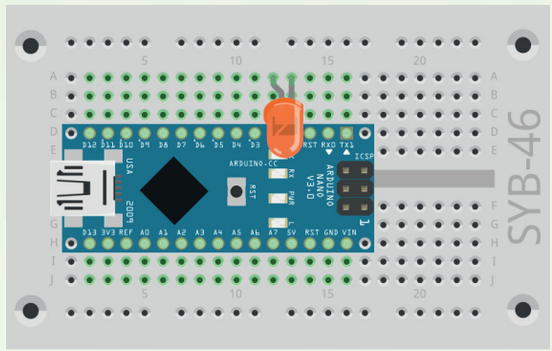
Für Logiksignale benötigen einige Arduino-kompatible Platinen 3,3 V, andere 5 V. Das Nano-Board in diesem Adventskalender verwendet ein +5-V-Signal als logisch *high* bzw. *wahr*.

LED durch serielle Kommandos schalten

Zur Steuerung aus Minecraft heraus wird die über USB simulierte serielle Schnittstelle des Nano-Boards verwendet, über die Minecraft einfache Befehle an das Nano-Board überträgt. Das Programm des 2. Tages verwendet noch kein Minecraft, sondern zeigt, wie man Steuerkommandos an das Nano-Board überträgt.

Bauteile

- 1 Nano-Board
- 1 Steckbrett
- 1 LED rot mit Vorwiderstand



Eine LED am Nano

Bauen Sie die abgebildete Schaltung auf dem Steckbrett auf. 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. Das Nano-Board lässt sich in manchen Fällen beim ersten Mal nur schwer in das Steckbrett drücken. Drücken Sie kräftig mit beiden Daumen darauf, aber verwenden Sie kein Werkzeug, das die Platine beschädigen könnte.

Die Pins auf dem Nano-Board

Alle Pins mit D… sind digitale Ein- oder Ausgänge, die die Werte *wahr* oder *falsch* (ein oder aus, `HIGH` oder `LOW`) 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. Das Nano-Board im Adventskalender benötigt für ein logisches *Wahr*-Signal +5 V, manche andere Platinen nur +3,3 V.

Das Programm

Das Programm `blink02.ino` wird auf dem Nano-Board installiert und wartet dann auf Steuerungsbefehle des Benutzers.

```
int led1 = 2;

void setup()
{
  Serial.begin(9600);
  pinMode(led1, OUTPUT);
  digitalWrite(led1, 0);
}

void loop()
{
  int cmd = Serial.parseInt();
  if(cmd == 1)
```

```
  {
    digitalWrite(led1, 1);
  }
  if(cmd == 2)
  {
    digitalWrite(led1, 0);
  }
}
```

So funktioniert das Programm

```
int led1 = 2;
```

Die Variable `led1` enthält die Nummer des Pins 2, an dem die LED angeschlossen ist.

```
Serial.begin(9600);
```

In der Prozedur `void setup()` wird beim Start die serielle Kommunikation eingerichtet. Da das Programm nur einzelne Zeichen überträgt, spielt die Baudrate – hier 9600 – kaum eine Rolle, sie muss nur in allen beteiligten Komponenten gleich eingestellt sein.

```
pinMode(led1, OUTPUT);
digitalWrite(led1, 0);
```

Danach wird Pin 2 als Ausgang definiert und abgeschaltet, falls er aus einem früheren Programmlauf noch eingeschaltet ist.

```
int cmd = Serial.parseInt();
```

Das Programm soll auf einfache Zahlen als Kommandos reagieren. Gegenüber Textbefehlen hat das den Vorteil, dass man sich aufwendige Überprüfungen der Eingabe sparen kann. Alles, was keine Zahl ist, wird einfach ignoriert. Diese Zeile liest aus der seriellen Eingabe eine Zahl und speichert diese in der Integervariable `cmd`. Wir verwenden in diesem Beispiel solche Zahlenkommandos, da das Arduino Minecraft Interface (AMI) die gleiche Art von Steuerungskommandos nutzt.

```
if(cmd == 1)
{
  digitalWrite(led1, 1);
}
```

Ist das Steuerungskommando gleich 1, wird die LED eingeschaltet.

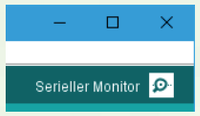
= ist nicht gleich ==

Die Programmiersprache C verwendet das einfache Gleichheitszeichen `=` zur Zuweisung von Variablen. Das doppelte Gleichheitszeichen `==` dient in einer Abfrage dazu, zu prüfen, ob zwei Werte gleich sind.

```
if(cmd == 2)
{
  digitalWrite(led1, 0);
}
```

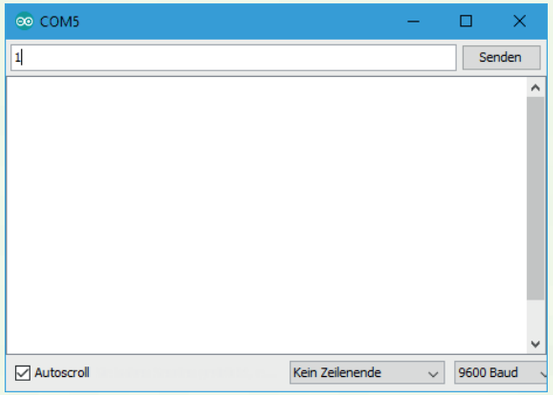
Ist das Steuerungskommando gleich 2, wird die LED ausgeschaltet. Hier wird bewusst nicht das Kommando `0` verwendet, da `Serial.parseInt()` auch dann 0 liefert, wenn nichtnumerische Kommandos oder einfach gar nichts empfangen wird.

Klicken Sie oben rechts in der Arduino-IDE auf das Symbol *Serieller Monitor*.



Seriellen Monitor in der Arduino-IDE aufrufen

Wählen Sie unten rechts die im Programm verwendete Baudrate *9600 Baud*. Geben Sie oben im Eingabefeld eine `1` ein und klicken Sie auf *Senden*. Die LED wird eingeschaltet. Geben Sie eine `2` ein, wird die LED ausgeschaltet. Das große Fenster im seriellen Monitor zeigt nichts an. Hier würden Rückmeldungen des Nano-Boards erscheinen. Das verwendete Programm gibt aber nichts zurück.



Der serielle Monitor der Arduino-IDE

Für Ihre Notizen

3. TAG

Heute im Adventskalender

- Schaltdraht (isoliert)

Schaltdraht

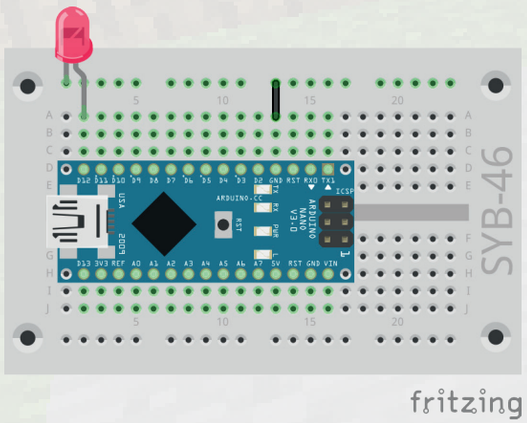
Heute ist Schaltdraht im Adventskalender enthalten. Damit stellen Sie kurze Verbindungsbrücken her, mit denen Kontaktreihen auf der Steckplatine verbunden werden. Schneiden Sie den Draht mit einem kleinen Seitenschneider je nach Experiment auf die passenden Längen zu. Um die Drähte besser in die Steckplatine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

LED mit Minecraft ein- und ausschalten

Ein Befehlsblock in der Minecraft-Welt schaltet eine LED abwechselnd ein und aus.

Bauteile

- 1 Nano-Board
- 1 Steckbrett
- 1 LED rot mit Vorwiderstand
- 1 Drahtbrücke



LED über Masseleiste angeschlossen

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-Board 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 12 verbunden.

Minecraft

In Minecraft erkundet man eine schier endlose Welt, die aus einfachen Würfeln erbaut ist. Die Blöcke bestehen aus verschiedenen Materialien und können abgebaut werden, um sie als Rohstoffe zu verarbeiten und daraus andere Dinge zu bauen. Der Spieler übernimmt die Rolle der Spielfigur Steve und steuert diese. Dabei kann man die Szene aus Steves Perspektive sehen oder aus Sicht einer Kamera von außen darauf blicken.

Startbildschirm der Minecraft Java Edition

Dieser Adventskalender geht davon aus, dass Sie bereits Erfahrungen mit Minecraft haben. Wer Minecraft bis jetzt nicht kennt, braucht sich nur wenige Tasten zu merken:

- Mit der Maus dreht man sich, ohne eine Maustaste zu drücken, um die eigene Achse und neigt den Blick nach oben oder unten.
- Mit vier auch aus anderen Spielen bekannten Buchstabentasten bewegt man sich: mit W nach vorne, mit S nach hinten, mit A nach links und mit D nach rechts.
- Mit der Leertaste kann man in die Höhe springen. Damit lassen sich auch Stufen im Gelände nach oben oder unten überwinden. Drückt man die Leertaste zweimal kurz hintereinander, wird auf den Flugmodus umgeschaltet. In diesem Modus schwebt man und ist nicht mehr an den Boden gebunden. Im Flugmodus steigt man durch längeres Drücken der Leertaste weiter nach oben.
- Umgekehrt duckt man sich mit der linken Umschalt-Taste etwas nach unten. Im Flugmodus verringert man mit dieser Taste die Flughöhe.
- Die Taste E öffnet das Inventar, in dem jede Menge unterschiedlicher Blöcke zum Bau zur Verfügung stehen. Neun verschiedene Blöcke oder Werkzeuge sind in der Inventarleiste am unteren Bildschirmrand jederzeit verfügbar. Hier wählt man mit den Tasten 1 bis 9 oder mit dem Mausrad das gewünschte Objekt aus.
- Ein Klick mit der linken Maustaste entfernt den angeklickten Block, ein Klick mit der rechten Maustaste platziert einen Block des gewählten Typs an der angeklickten Position.
- Die Esc-Taste blendet ein Spielmenü ein, über das man das Spiel verlassen oder Spieloptionen einstellen kann. Beim Wechseln in ein anderes Programm mit Alt + Tab wechselt das Spiel ebenfalls in dieses Menü.

Das Spielmenü in Minecraft

Minecraft-Welt und Grundeinstellungen

Im Download für diesen Adventskalender finden Sie den Ordner **Adventskalender**, der eine vorbereitete Minecraft-Welt enthält. Kopieren Sie ihn in das Verzeichnis **%APPDATA%\.minecraft\saves**. Dieses Verzeichnis finden Sie im Explorer unter **c:\Users\Benutzername\AppData\Roaming\.minecraft\saves**. Dort befinden sich die Demo-Welt sowie bereits von Ihnen angelegte Welten.

Ausschnitt aus der Minecraft-Welt des Adventskalenders

Sie können auch eine eigene Minecraft-Welt verwenden. Dazu sollten Sie einige Voreinstellungen beachten:

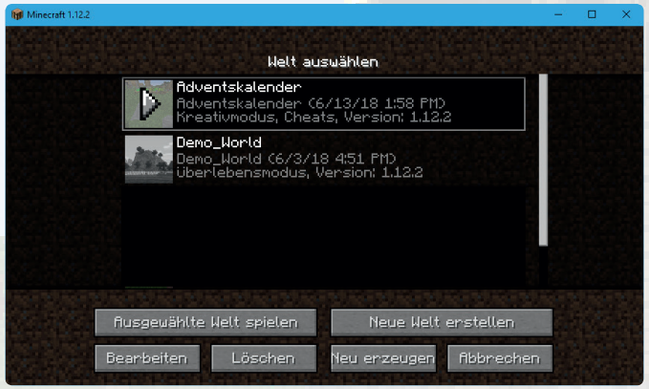
- Wählen Sie beim Erstellen der Welt den *Kreativmodus* und schalten Sie *Cheats erlauben* ein.
- Setzen Sie in den Einstellungen die Schwierigkeit auf *Friedlich*.
- Setzen Sie in den Grafikeinstellungen den Grafikmodus auf *Schnell*. Je nach verwendeter Grafikkarte empfiehlt es sich, *Weiche Beleuchtung* und *3D-Effekt* auszuschalten, um flüssigere Bewegungen zu erreichen.

Einstellungen für die Minecraft-Welt

In der im Download mitgelieferten Welt sind diese Parameter bereits voreingestellt.

Aufbau in Minecraft

Beginnen Sie das Spiel mit Ihrer selbst angelegten Welt oder mit der mitgelieferten Welt. Letztere wird in der Liste der Welten automatisch mit angezeigt, wenn Sie die Dateien aus dem Download wie weiter oben beschrieben kopiert haben. Klicken Sie nach Auswahl der Welt auf *Ausgewählte Welt spielen* oder direkt auf den Pfeil im Bild links neben dem Namen der Welt.



Auswahl der Welt

Geben Sie gleich am Anfang den ersten Cheat ein, der verhindert, dass es in der Welt Nacht wird. Das erleichtert das Bauen deutlich. Drücken Sie dazu die Taste T. Sie öffnet die Eingabezeile. Geben Sie hier ein:

```
/gamerule doDaylightCycle false
```

Diese Regel hält den Tag-/Nachtzyklus an. Natürlich sollte zu diesem Zeitpunkt gerade Tag sein. In der Nacht würde die Regel dafür sorgen, dass es immer Nacht bleibt.

Für den Schalter, der die LED umschalten soll, bauen Sie als Erstes einen Steinblock in der Minecraft-Welt. Die mitgelieferte Welt enthält eine Fläche aus Steinplatten, auf der Sie leicht neue Blöcke aufbauen können, ohne erst etwas wegräumen zu müssen.

Setzen Sie an diesen Steinblock einen sogenannten *Knopf*. Dies ist ein einfacher Druckschalter, den Sie im Inventar unter *Redstone* finden. Dabei spielt es keine Rolle, ob der Knopf seitlich am Steinblock oder obendrauf angebaut wird.

Der Knopf im Inventar

Ein Knopf muss immer an einen Block gehängt werden und kann dann vom Spieler mit der rechten Maustaste gedrückt werden.

Der Steinblock mit dem Knopf

Um einen Befehl auszulösen, der an das Nano-Board weitergegeben wird, verwenden Sie einen *Befehlsblock*. Dieser ist im Inventar nicht vorhanden. Man erhält ihn nur über einen Cheat:

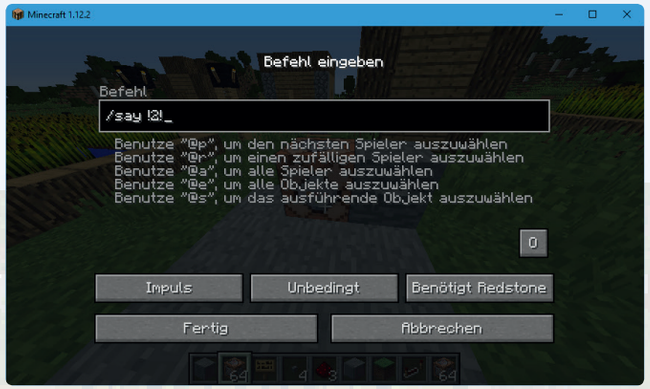
```
/give @s command_block 64
```

Dieser Cheat liefert einen Stapel aus 64 Befehlsblöcken auf einen freien Platz in der Schnellzugriffsleiste des Inventars. Mehr als 64 Blöcke auf einmal sind nicht möglich.

Platzieren Sie einen Befehlsblock direkt neben dem Steinblock. Ein Befehlsblock muss immer von einem anderen Block aktiviert werden.

Der Befehlsblock neben dem Steinblock

Klicken Sie mit der rechten Maustaste auf den Befehlsblock, um ihm einen Befehl zuzuweisen, der beim Aktivieren ausgeführt werden soll.



Befehl für einen Befehlsblock festlegen

Schreiben Sie in das Feld *Befehl* die abgebildete Befehlszeile:

```
/say !2!
```

Dieser Befehl schickt das Steuerungskommando 2 über das *AMI* (Arduino Minecraft Interface) an das angeschlossene Nano-Board.

Klicken Sie anschließend auf *Fertig*. Mit einem Rechtsklick auf den Befehlsblock können Sie den Befehl dieses Blocks jederzeit ändern.

Das Programm auf dem Nano-Board

Das Programm `mc03.ino` schaltet die LED am Pin 12 des Nano-Boards jedes Mal, wenn das Steuerungskommando 2 empfangen wird, abwechselnd ein oder aus.

```
void setup()
{
  Serial.begin(9600);
  pinMode(12, OUTPUT);
}

void loop()
{
  int cmd = Serial.parseInt();
  if(cmd == 2)
  {
    digitalWrite(12, !digitalRead(12));
  }
}
```

Übertragen Sie das Programm mit der Arduino-IDE auf das Nano-Board.

So funktioniert das Programm

Die Prozedur `void setup()` startet, wie bereits bekannt, die serielle Kommunikation und richtet den Pin 12 als Ausgang ein.

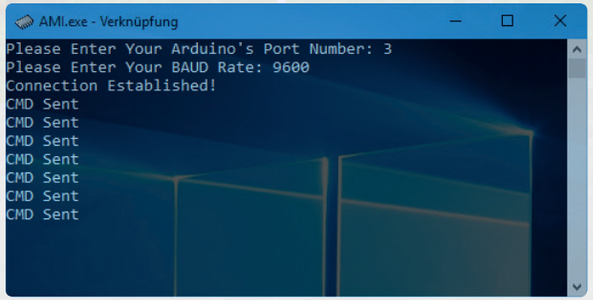
Die Prozedur `void loop()` liest, wie ebenfalls schon bekannt, ein Steuerungskommando über die serielle Schnittstelle ein.

```
digitalWrite(12, !digitalRead(12));
```

Handelt es sich dabei um das Kommando 2, wird der Pin 12 auf den umgekehrten Logikwert von dem gesetzt, den er gerade hat. Eine eingeschaltete LED wird ausgeschaltet, eine ausgeschaltete LED wird eingeschaltet. Dazu liest die Funktion `digitalRead()` den Wert des Pins 12. Dieser Wert wird mit dem Zeichen `!` negiert und anschließend mit `digitalWrite()` wieder auf den Pin 12 geschrieben.

Arduino mit Minecraft verbinden

1. Beenden Sie Minecraft, falls es gerade läuft.
2. Das Programm `AMI.exe` verbindet Minecraft mit dem Nano-Board. Starten Sie das Programm aus dem Verzeichnis `%APPDATA%\.minecraft\logs` oder über ein selbst angelegtes Desktopsymbol. Der serielle Monitor der Arduino-IDE darf dabei nicht laufen.
3. Geben Sie im ersten Schritt die Nummer der verwendeten Schnittstelle ohne `com` ein. Danach geben Sie die verwendete Baudrate `9600` ein.
4. Starten Sie Minecraft neu.
5. AMI quittiert jedes Steuerungskommando aus Minecraft mit der Zeile `CMD Sent`. Auf dem Nano-Board blinkt die LED *RX1* während der sehr kurzen Übertragung einmal auf.



Das Programm AMI in Aktion

6. Das Nano-Board führt das Steuerungskommando aus.

Nach diesem Prinzip funktionieren auch alle weiteren Programme in diesem Adventskalender.

Wichtig

AMI muss immer bereits laufen, wenn Minecraft gestartet wird. Startet man AMI erst nach Minecraft, kann das Spiel keine Verbindung zum Nano-Board herstellen.

Klicken Sie mit der rechten Maustaste in der Minecraft-Welt auf den Knopf. Der Befehlsblock sendet das Steuerungskommando 2 über AMI an das Nano-Board. Hier wird der Pin 12 umgeschaltet.

Ein Rechtsklick auf den Knopf sendet das Steuerungskommando 2.

4. TAG

Heute im Adventskalender

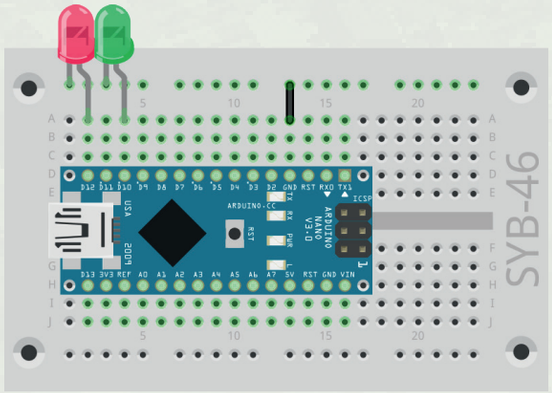
- 1 LED grün mit Vorwiderstand

LEDs über Redstone-Leitungen wechselweise umschalten

Das Experiment des 4. Tages schaltet eine LED ein und die andere aus. Im Gegensatz zum vorherigen Programm erfolgt die Umschaltung unabhängig vom aktuellen Schaltzustand der LEDs.

Bauteile

- 1 Nano-Board
- 1 Steckbrett
- 1 LED rot mit Vorwiderstand
- 1 LED grün mit Vorwiderstand
- 1 Drahtbrücke



Zwei LEDs werden abwechselnd ein- und ausgeschaltet.

Aufbau in Minecraft

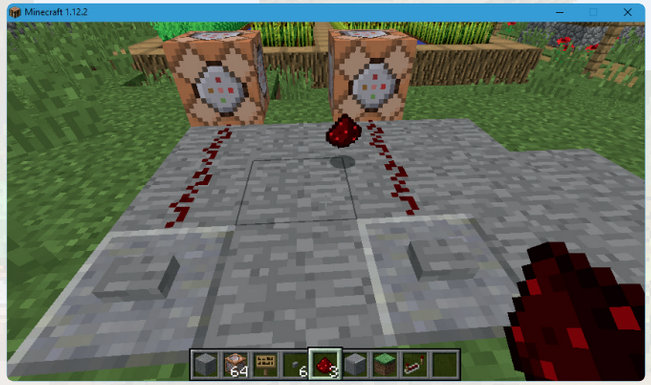
Der Knopf zum Schalten und der Befehlsblock müssen nicht unmittelbar nebeneinander liegen. Sie können auch über Redstone-Leitungen, die wie elektrische Kabel wirken, miteinander verbunden sein. Diese Leitungen werden aus „rotem Staub" gebaut und verbinden sich automatisch.

1. Bauen Sie wie in der Abbildung einen Klotz aus dem Material *polierter Andesit* in den Boden ein. Schlagen Sie dazu vorher einen vorhandenen Klotz weg. Dieses Material bewirkt, dass der Klotz leicht aus dem Boden herausragt, wie eine dickere gefaste Bodenplatte.
2. Setzen Sie einen Knopf darauf.
3. Bauen Sie mit einem Abstand von mindestens zwei Blöcken einen Befehlsblock und geben Sie ihm den Befehl `/say !10!`. Dieser Befehlsblock soll die LED am Pin 10 einschalten.
4. Legen Sie Redstone auf die Felder zwischen dem Klotz mit dem Knopf und dem Befehlsblock. Dieses Material finden Sie im Inventar im Bereich *Redstone*. Es handelt sich um eine Art Sand, der auf ein Feld gestreut wird. Dabei verbindet es sich automatisch mit benachbarten Feldern, auf denen ebenfalls Redstone liegt, zu einer Leitung.

Redstone-Material im Inventar

Die Redstone-Leitung verbindet den Knopf mit dem Befehlsblock

5. Bauen Sie den gleichen Aufbau daneben noch einmal auf. Zwischen beiden muss mindestens ein Block Abstand sein, damit sich die Redstone-Leitungen nicht automatisch verbinden.
6. Geben Sie dem zweiten Befehlsblock den Befehl `/say !12!`. Er soll die LED am Pin 12 einschalten.



Die zweite Schaltung

Das Programm

Das Programm `mc04.ino` schaltet die LED am Pin 10 des Nano-Boards ein und die LED am Pin 12 aus, wenn das Steuerungskommando `10` empfangen wird. Umgekehrt schaltet das Steuerungskommando `12` die LED am Pin 12 ein und die LED am Pin 10 aus.

```
void setup()
{
  Serial.begin(9600);
  pinMode(10, OUTPUT);
  pinMode(12, OUTPUT);
}

void loop()
{
  int cmd = Serial.parseInt();
  if(cmd == 10)
  {
    digitalWrite(10, 1);
    digitalWrite(12, 0);
  }
  if(cmd == 12)
  {
    digitalWrite(10, 0);
    digitalWrite(12, 1);
  }
}
```

So funktioniert das Programm

Der grundsätzliche Aufbau des Programms entspricht dem der vorherigen. Diesmal werden zwei Steuerungskommandos ausgewertet. Das Kommando `10` setzt über `digitalWrite()` den Pin 10 auf `1` und schaltet ihn damit ein. Der Pin 12 wird mit dem Wert `0` ausgeschaltet. Das Kommando `12` schaltet auf die gleiche Weise den Pin 12 ein und den Pin 10 aus.

Ein Rechtsklick auf den Knopf gibt das Steuerungskommando 12 aus.

Klicken Sie mit der rechten Maustaste auf einen der Knöpfe, wird der angeschlossene Befehlsblock aktiviert. Die Redstone-Leitung leuchtet auf, um anzuzeigen, dass ein Signal fließt.

Für Ihre Notizen

5. TAG

Heute im Adventskalender

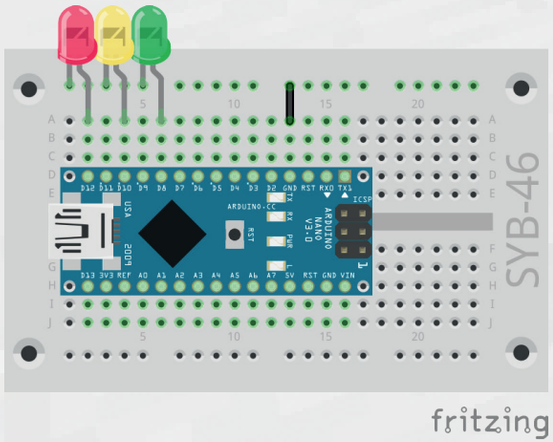
- 1 LED gelb mit Vorwiderstand

Ampel

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

Bauteile

- 1 Nano-Board
- 1 Steckbrett
- 1 LED rot mit Vorwiderstand
- 1 LED gelb mit Vorwiderstand
- 1 LED grün mit Vorwiderstand
- 1 Drahtbrücke



Ampel aus drei LEDs

Aufbau in Minecraft

Die Ampel wird über eine Steindruckplatte gestartet. Diesen Block finden Sie im Inventar unter *Redstone*.

Die Steindruckplatte im Inventar

Diese Steindruckplatte löst eine Aktion auf einem daneben liegenden Befehlsblock aus, wenn die Spielfigur auf die Steindruckplatte tritt. Bauen Sie einen Befehlsblock neben die Steindruckplatte und geben Sie ihm den Befehl `/say !1!`. Damit wird die Ampelsequenz gestartet.

Steindruckplatte neben einem Befehlsblock

Das Programm

Im Programm `mc05.ino` werden nacheinander verschiedene Kombinationen von LEDs ein- und ausgeschaltet. Das Programm startet mit der Rotphase der Ampel, bei der die gelbe und die grüne LED ausgeschaltet sind. Beim Betreten der Steindruckplatte wird die gelbe LED zusätzlich eingeschaltet. Nach einer kurzen Rot/Gelb-Phase von 0,6 Sekunden werden die rote und die gelbe LED aus- und die grüne eingeschaltet. Die Grünphase dauert 2 Sekunden, darauf folgt eine kurze Gelbphase von 0,6 Sekunden, danach leuchtet die Ampel wieder rot.

```
int rot =   12;
int gelb =  10;
int gruen = 8;
```

```
void setup()
{
  Serial.begin(9600);
  pinMode(rot, OUTPUT);
  pinMode(gelb, OUTPUT);
  pinMode(gruen, OUTPUT);
}

void loop()
{
  digitalWrite(rot, 1);
  int cmd = Serial.parseInt();
  if(cmd == 1)
  {
    digitalWrite(gelb, 1);
    delay(600);
    digitalWrite(rot, 0);
    digitalWrite(gelb, 0);
    digitalWrite(gruen, 1);
    delay(2000);
    digitalWrite(gruen, 0);
    digitalWrite(gelb, 1);
    delay(600);
    digitalWrite(gelb, 0);
    digitalWrite(rot, 1);
    delay(2000);
  }
}
```

So funktioniert das Programm

Die ersten Programmzeilen definieren die Variablen `rot`, `gelb`, `gruen` für die drei LEDs der Ampel. Damit braucht man sich im Sketch keine Pinnummern zu merken, sondern kann die LEDs einfach über ihre Farben ansteuern.

Die Prozedur `void setup()` startet die serielle Kommunikation und definiert die Pins der drei LEDs als Ausgänge.

Die Prozedur `void loop()` schaltet die rote LED ein und wartet dann auf das Steuerungskommando `1` aus Minecraft. Dann startet der Ampelzyklus, indem die gelbe LED zusätzlich zur roten eingeschaltet wird. Danach wartet die Funktion `delay(600)` 600 Millisekunden, bevor die rote und die gelbe LED ausgeschaltet werden und dafür die grüne eingeschaltet wird.

Nach 2000 Millisekunden endet die Grünphase, die grüne LED wird ausgeschaltet, die gelbe eingeschaltet. Die Gelbphase dauert 600 Millisekunden. Danach springt die Ampel wieder auf Rot.

Damit nicht sofort der nächste Ampelzyklus beginnt, falls die Spielfigur immer noch auf der Steindruckplatte steht, wartet das Programm bei leuchtender roter LED noch einmal 2000 Millisekunden, bevor wieder eine serielle Eingabe ausgewertet wird.

6. TAG

Heute im Adventskalender

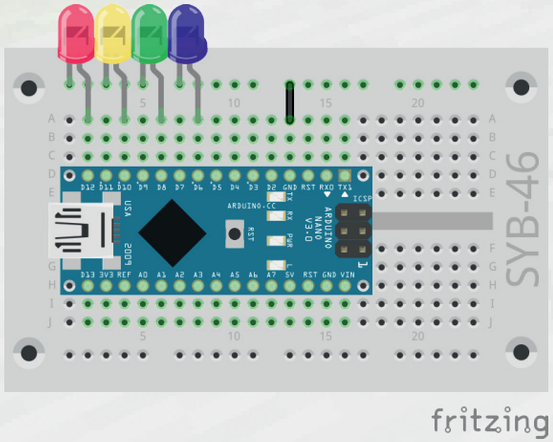
- 1 LED blau mit Vorwiderstand

LEDs blinken zufällig

Das Experiment des 6. Tages lässt vier LEDs zufällig blinken.

Bauteile

- 1 Nano-Board
- 1 Steckbrett
- 1 LED rot mit Vorwiderstand
- 1 LED gelb mit Vorwiderstand
- 1 LED grün mit Vorwiderstand
- 1 LED blau mit Vorwiderstand
- 1 Drahtbrücke



Vier LEDs am Nano-Board angeschlossen

Aufbau in Minecraft

Die Blinksequenz wird gestartet, wenn die Spielfigur einen Hebel in der Minecraft-Welt umlegt. Der Hebel ist im Inventar wie alle Schaltelemente unter *Redstone* zu finden. Jedes Umlegen des Hebels aktiviert den benachbarten Befehlsblock. Dabei spielt die Position des Hebels keine Rolle. Es gibt also kein „eingeschaltet“ oder „ausgeschaltet“.

Bauen Sie einen Befehlsblock neben den Hebel und geben Sie ihm den Befehl `/say !1!`. Damit wird die Blinksequenz gestartet.

Hebel neben einem Befehlsblock

Das Programm

Beim Betätigen des Hebels beginnt im Programm `mc06.ino` eine Schleife, die 20 Mal eine zufällig ausgewählte LED für eine Zehntelsekunde aufblinken lässt. Der gesamte Durchlauf dauert 2 Sekunden und endet danach automatisch.

```
int i;
int j;
void setup()
{
  Serial.begin(9600);
  pinMode(6, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(10, OUTPUT);
  pinMode(12, OUTPUT);
}

void loop()
{
  int cmd = Serial.parseInt();
  if(cmd == 1)
  {
    for (j=0; j<20; j++)
    {
      i = random(4)*2+6;
      digitalWrite(i, 1);
      delay(100);
      digitalWrite(i, 0);
    }
  }
}
```

Wie entstehen Zufallszahlen?

Gemeinhin denkt man, in einem Programm könne nichts zufällig geschehen – wie also kann ein Programm zufällige Zahlen generieren? Teilt man eine große Primzahl durch einen 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 mehrfachen Aufruf des gleichen Programms jederzeit reproduzieren. Nimmt man aber eine aus einigen dieser Ziffern zusammengebaute Zahl und teilt sie durch eine Zahl, die sich aus der aktuellen Uhrzeitsekunde oder dem Inhalt einer beliebigen Speicherstelle des Computers ergibt, kommt ein Nachkommawert heraus, der sich nicht reproduzieren lässt und daher als Zufallszahl bezeichnet wird.

So funktioniert das Programm

Am Anfang werden zwei Ganzzahlvariablen `i` und `j` definiert, die im Programm für den Schleifenzähler und die LED-Nummer verwendet werden.

Die Prozedur `void setup()` startet die serielle Kommunikation und definiert die Pins der vier LEDs als Ausgänge.

Die Prozedur `void loop()` wertet die serielle Kommunikation aus und wartet auf den Steuerungsbefehl `1`. Wird er empfangen, beginnt eine Schleife. Schleifen werden wie in vielen Programmiersprachen mit `for` definiert und haben immer drei Parameter:

```
for (j=0; j<20; j++)
```

`j=0` bedeutet, dass der Schleifenzähler j bei `0` beginnt. `j<20` bedeutet, dass die Schleife läuft, solange der Schleifenzähler kleiner als `20` ist. `j++` zählt den Schleifenzähler in jedem Durchlauf um 1 hoch. Damit läuft die Schleife genau 20 Mal.

Am Anfang jedes Durchlaufs der Schleife wird die Variable `i` auf eine Zufallszahl zwischen 0 und 4 gesetzt. Diese Zahl wird mit 2 multipliziert, und anschließend wird 6 addiert. So entstehen aus den möglichen Zufallszahlen `0`, `1`, `2`, `3` die Pinnummern `6`, `8`, `10`, `12`.

```
i = random(4)*2+6;
```

Die zufällig gewählte LED wird für 100 Millisekunden eingeschaltet und danach wieder ausgeschaltet. Im nächsten Schleifendurchlauf wird wieder eine neue LED zufällig gewählt. Dabei kann es durchaus passieren, dass mehrmals hintereinander dieselbe LED aufleuchtet.

7. TAG

Heute im Adventskalender

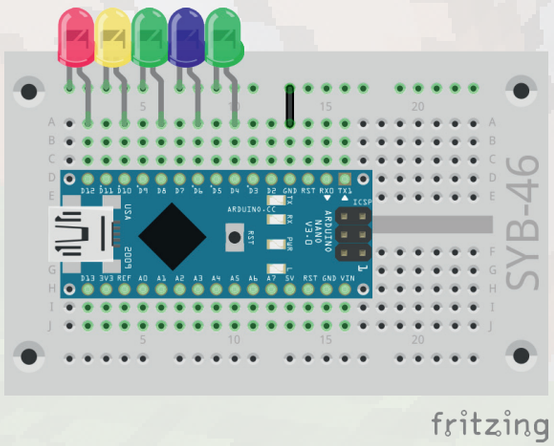
- 1 LED grün mit Vorwiderstand

Lauflicht mit fünf LEDs

Lauflichter sind beliebte Effekte für Leuchtwerbung und in Partyräumen. Das Experiment des 7. Tages erzeugt ein Lauflicht aus fünf LEDs. Gesteuert wird es mit zwei Hebeln in Minecraft. Der eine Hebel lässt das Lauflicht von rechts nach links laufen, der andere von links nach rechts.

Bauteile

- 1 Nano-Board
- 1 Steckbrett
- 1 LED rot mit Vorwiderstand
- 1 LED gelb mit Vorwiderstand
- 2 LEDs grün mit Vorwiderstand
- 1 LED blau mit Vorwiderstand
- 1 Drahtbrücke



Lauflicht mit fünf LEDs

Aufbau in Minecraft

Bauen Sie in Minecraft zwei Befehlsblöcke und neben jeden der beiden einen Hebel. Geben Sie dem einen Befehlsblock den Befehl `/say !1!` und dem anderen `/say !2!`.

Zwei Hebel mit Befehlsblöcken steuern das Lauflicht.

Das Programm

Beim Betätigen einer der beiden Hebel startet im Programm `mc07.ino` eine Schleife, die 10 Mal hintereinander eine Lauflichtsequenz laufen lässt. Dazu wird in der Schleife eine weitere Schleife gestartet, die der Reihe nach jede LED für eine Zehntelsekunde aufblinken lässt.

```
int i;
int j;
void setup()
{
  for (i=4; i<=12; i+=2)
  {
    pinMode(i, OUTPUT);
  }
  Serial.begin(9600);
}

void loop()
{
  int cmd = Serial.parseInt();
  if(cmd == 1)
  {
    for (j=0; j<10; j++)
    {
      for (i=4; i<=12; i+=2)
      {
        digitalWrite(i, 1);
        delay(100);
        digitalWrite(i, 0);
      }
    }
  }
  if(cmd == 2)
  {
    for (j=0; j<10; j++)
    {
    for (i=12; i>=4; i-=2)
      {
        digitalWrite(i, 1);
        delay(100);
        digitalWrite(i, 0);
      }
    }
  }
}
```

So funktioniert das Programm

Die Prozedur `void setup()` startet die serielle Kommunikation und definiert die Pins der fünf LEDs als Ausgänge. Dabei werden die Pinnummern nicht einzeln angegeben, sondern in einer Schleife errechnet. Diese Schleife zählt in Zweierschritten von 4 bis einschließlich 12 und initialisiert so die Pins 4, 6, 8, 10, 12.

Die Prozedur `void loop()` wertet die serielle Kommunikation aus und wartet auf einen Steuerungsbefehl. Bei 1 läuft eine Schleife 10 Mal. In jedem Durchlauf läuft eine weitere Schleife mit den gleichen Parametern wie bei der Initialisierung der Pins und schaltet jeden Pin für 100 Millisekunden ein und danach wieder aus.

Beim Steuerungsbefehl 2 läuft die innere Schleife in umgekehrter Richtung. In jedem Durchlauf wird der Schleifenzähler um 2 verringert, von 12 auf einschließlich 4. Dadurch läuft das Lauflicht in umgekehrter Richtung.

8. TAG

Heute im Adventskalender

- 1 RGB-LED mit Vorwiderständen

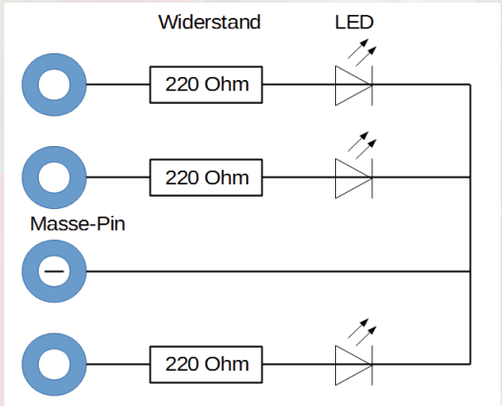
RGB-LEDs

Eine normale LED leuchtet immer nur in einer Farbe. Die im Adventskalender verwendeten RGB-LEDs können wahlweise in mehreren Farben leuchten. Hier sind im Prinzip drei LEDs in verschiedenen Farben in ein transparentes 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.

Anschlusspins einer RGB-LED

Die Anschlussdrähte der RGB-LEDs sind unterschiedlich lang, um sie eindeutig kenntlich zu machen. Im Gegensatz zu normalen LEDs ist die Kathode hier der längste Draht.

RGB-LEDs funktionieren wie drei einzelne LEDs und brauchen deshalb auch drei Vorwiderstände. In den RGB-LEDs in diesem Adventskalender sind sie ebenfalls bereits eingebaut.



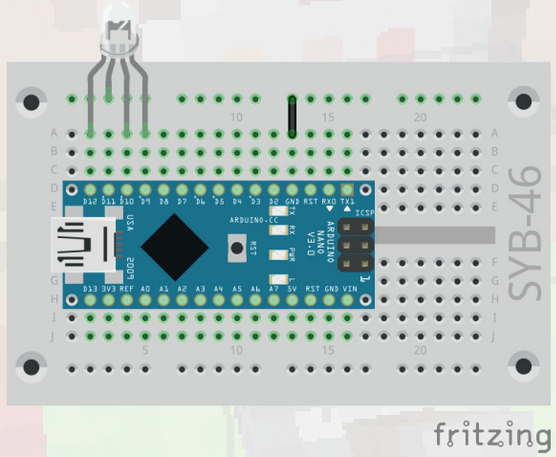
Schaltplan für eine RGB-LED mit drei Vorwiderständen

Verschiedene Farben auf einer RGB-LED

Das Experiment des 8. Tages schaltet die drei Farbkomponenten einer RGB-LED einzeln ein und aus.

Bauteile

- 1 Nano-Board
- 1 Steckbrett
- 1 RGB-LED mit Vorwiderstand
- 1 Drahtbrücke



RGB-LED am Nano-Board

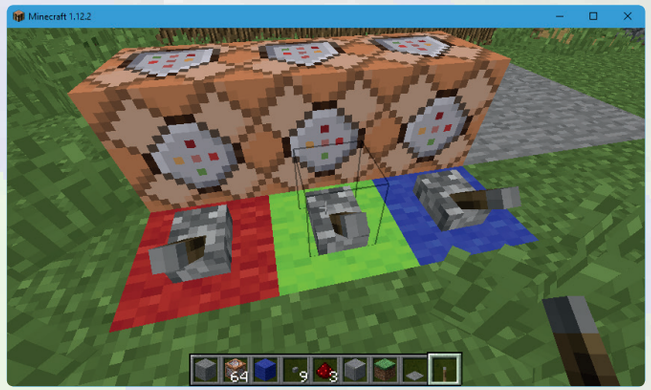
Additive Farbmischung

RGB-LEDs nutzen die sogenannte additive Farbmischung. Hierbei 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 das Schwarz, das gar kein Licht mehr reflektiert.

Additive Farbmischung

Aufbau in Minecraft

Um die drei Farbkomponenten der RGB-LED eindeutig zuzuordnen, heben Sie zunächst einen drei Blöcke langen Graben aus. Bauen Sie hier farbige Blöcke aus den Materialien rote Wolle, grüne Wolle und blaue Wolle. Bauen Sie dann drei Befehlsblöcke mit jeweils einem Hebel davor. Geben Sie den Befehlsblöcken die Befehle `/say !1!`, `/say !2!`, `/say !3!`.



Schalthebel für die drei Farbkomponenten einer RGB-LED

Das Programm

Das Programm `mc08.ino` schaltet über die drei Hebel in der Minecraft-Welt die drei Farbkomponenten der RGB-LED ein und aus. Durch Kombination lassen sich außer den drei Grundfarben drei Mischfarben aus je zwei Farbkomponenten sowie Weiß als Mischfarbe aller drei Farbkomponenten erzeugen.

```
int r = 12;
int g = 10;
int b = 9;
```

```
void setup()
{
  pinMode(r, OUTPUT);
  pinMode(g, OUTPUT);
  pinMode(b, OUTPUT);
  Serial.begin(9600);
}

void loop()
{
  int cmd = Serial.parseInt();
  if(cmd == 1)
  {
    digitalWrite(r, !digitalRead(r));
  }
  if(cmd == 2)
  {
    digitalWrite(g, !digitalRead(g));
  }
  if(cmd == 3)
  {
    digitalWrite(b, !digitalRead(b));
  }
}
```

So funktioniert das Programm

Am Anfang werden die drei verwendeten Pinnummern in den drei Variablen `r`, `g`, `b` gespeichert. Die Prozedur `void setup()` startet die serielle Kommunikation und definiert diese Pins als Ausgänge.

Die Prozedur `void loop()` liest ein Steuerungskommando über die serielle Schnittstelle ein. Jedes der Kommandos 1, 2 oder 3 setzt einen der Pins auf den umgekehrten Wert. Dabei wird eine leuchtende Farbkomponente ausgeschaltet oder eine ausgeschaltete wieder eingeschaltet.

Für Ihre Notizen

9. TAG

Heute im Adventskalender

- 2 Verbindungskabel

Blinklicht auf dem Minecraft-Klotz

Auf der Rückseite des Adventskalenders finden Sie eine Ausschneidevorlage für einen Minecraft-Klotz. Falten Sie sie zusammen und kleben Sie die Klebekanten aufeinander, sodass ein Würfel entsteht. Stülpen Sie ihn so über das Steckbrett, dass es an dem Ende, wo sich das USB-Anschlusskabel befindet, etwas aus dem Würfel herausteht.

Heute sind Anschlusskabel im Adventskalender, mit denen die LEDs auf dem Minecraft-Klotz mit dem Steckbrett verbunden werden. Stecken Sie eine LED von außen in die dafür vorgesehenen Löcher auf den Klotz. Die Positionen mit zwei Löchern sind für einfarbige LEDs, die Positionen mit vier Löchern werden später erst benötigt.

Stecken Sie die Buchsen der Anschlusskabel auf die Drähte der LEDs an der Innenseite des Klotzes. Sollten die Kabel zu leicht von den LED-Drähten rutschen, biegen Sie die Drähte leicht. Stecken Sie dann die Steckerseiten der Anschlusskabel gemäß der Abbildung auf das Steckbrett.

Bauteile

- 1 Nano-Board
- 1 Steckbrett
- 1 LED gelb mit Vorwiderstand
- 2 Verbindungskabel
- 1 Drahtbrücke

Aufbau in Minecraft

Bauen Sie fünf Befehlsblöcke in einer Reihe und davor jeweils eine Steindruckplatte, die einen der Befehlsblöcke aktiviert. Geben Sie den Befehlsblöcken die Befehle `/say !1!`, `/say !2!`, `/say !3!`, `/say !4!`, `/say !5!`.

Befehlsblöcke mit Steindruckplatten steuern das Blinklicht.

Das Programm

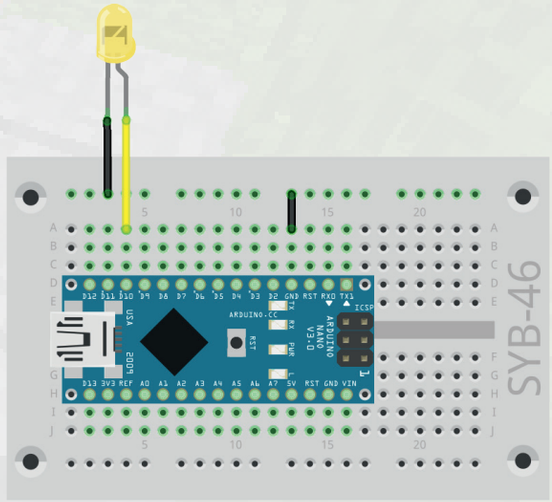
Das Programm `mc09.ino` lässt die LED auf dem Minecraft-Klotz blinken. Sie blinkt umso schneller, je weiter die Spielfigur auf den Steindruckplatten entlang der Befehlsblöcke nach rechts geht.

```
int i;
int z = 500;
void setup()
{
  pinMode(10, OUTPUT);
  Serial.begin(9600);
}

void loop()
{
  int cmd = Serial.parseInt();
  if(cmd == 1)
  {
    z = 400;
  }
  if(cmd == 2)
  {
    z = 300;
  }
  if(cmd == 3)
  {
    z = 200;
  }
  if(cmd == 4)
  {
    z = 100;
  }
  if(cmd == 5)
  {
    z = 20;
  }
  for(i=0; i<3; i++)
  {
    digitalWrite(10, 1);
    delay(z);
    digitalWrite(10, 0);
    delay(z);
  }
}
```

So funktioniert das Programm

Auch dieses Programm liest in der Prozedur `void loop()` regelmäßig die serielle Schnittstelle aus. Jede Steindruckplatte überträgt ein anderes Steuerungskommando. Abhängig vom empfangenen Kommando wird in der Variable `z` eine Zeit in Millisekunden gespeichert, die die Dauer eines Blinkvorgangs angibt. Danach lässt eine Schleife die LED dreimal in der aktuellen Geschwindigkeit blinken, bevor die Hauptschleife überprüft, ob ein neues Steuerungskommando übertragen wurde. Danach blinkt die LED wieder dreimal, auch wenn kein neues Steuerungskommando empfangen wurde und die Variable `z` unverändert geblieben ist.



Blinklicht auf dem Minecraft-Klotz

10. TAG

Heute im Adventskalender

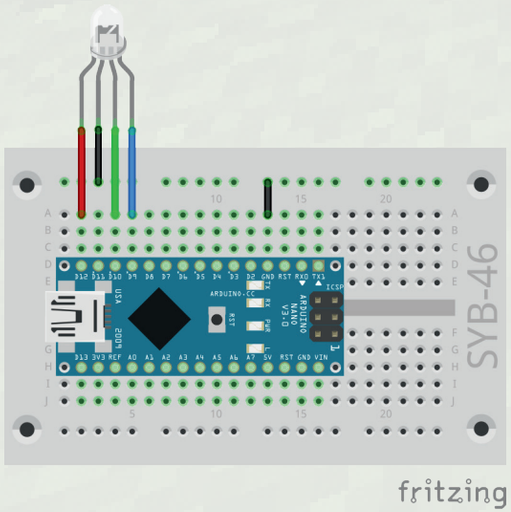
- 2 Verbindungskabel

RGB-LED auf dem Minecraft-Klotz

Auf dem Klotz befinden sich LED-Positionen mit vier Löchern für die RGB-LED. Ein Druck auf einen Knopf lässt die RGB-LED in zufälligen Farben blinken.

Bauteile

- 1 Nano-Board
- 1 Steckbrett
- 1 RGB-LED mit Vorwiderstand
- 4 Verbindungskabel
- 1 Drahtbrücke



RGB-LED auf dem Minecraft-Klotz

Aufbau in Minecraft

Bauen Sie einen Befehlsblock und daneben einen Block, auf dem ein Knopf angebracht ist. Der Befehlsblock kann auch in den Boden eingegraben werden. Wichtig ist nur, dass der Knopf zum Schalten auf einem Block neben dem Befehlsblock angebracht ist. Geben Sie dem Befehlsblock den Befehl `/say !1!`.

In den Boden eingegrabener Befehlsblock

Das Programm

Das Programm `mc10.ino` lässt die RGB-LED auf dem Minecraft-Klotz in zufälligen Farben blinken. Dazu wird nach dem Drücken des Knopfs 20 Mal eine zufällig gewählte Farbkomponente eingeschaltet und danach eine andere zufällig gewählte Farbkomponente ausgeschaltet. Durch dieses Zufallsprinzip kann es auch passieren, dass die RGB-LED kurz ganz aus ist oder sich die Farbe in einem Schaltzyklus nicht ändert.

```
int i;
int leds[] = {9, 10, 12};
void setup()
{
  for (i=0; i<3; i++)
  {
    pinMode(leds[i], OUTPUT);
  }
  Serial.begin(9600);
}

void loop()
{
  int cmd = Serial.parseInt();
  if(cmd == 1)
  {
    for (i=0; i<20; i++)
    {
      digitalWrite(leds[random(3)], 1);
      delay(50);
      digitalWrite(leds[random(3)], 0);
      delay(50);
    }
  }
}
```

So funktioniert das Programm

Die Pinnummern der LEDs werden in einer Liste `leds[]` gespeichert. So lassen sie sich leicht vom Programm adressieren. In der Prozedur `void setup()` definiert eine Schleife die drei Pins als Ausgänge.

Die Prozedur `void loop()` prüft, ob das Steuerungskommando `1` übertragen wurde. In diesem Fall läuft eine Schleife 20 Mal. In jedem Durchlauf wird ein zufällig gewählter Pin eingeschaltet und nach 50 Millisekunden ein wiederum zufällig gewählter Pin ausgeschaltet.

Für Ihre Notizen

11. TAG

Heute im Adventskalender

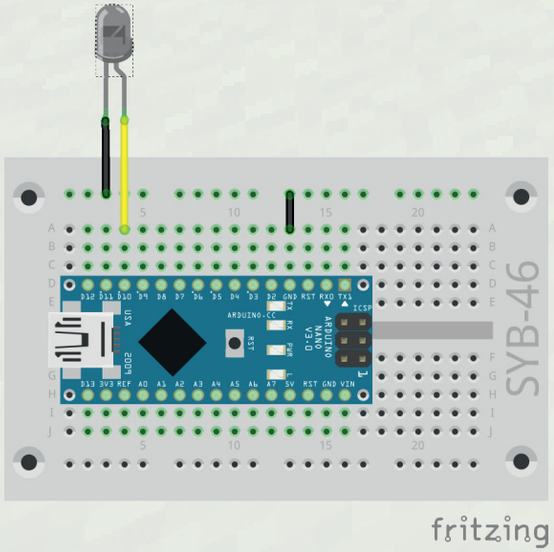
- 1 Blink-LED mit Vorwiderstand

Blinklicht auf dem Minecraft-Klotz

Heute ist eine LED im Adventskalender, die selbstständig blinkt, ohne dass ein Programm dazu nötig ist.

Bauteile

- 1 Nano-Board
- 1 Steckbrett
- 1 Blink-LED mit Vorwiderstand
- 2 Verbindungskabel
- 1 Drahtbrücke



Blinklicht auf dem Minecraft-Klotz

Aufbau in Minecraft

Bauen Sie zwei Befehlsblöcke und bringen Sie auf den Blöcken daneben je einen Knopf an. Der eine Knopf schaltet die Blink-LED ein, der andere schaltet sie wieder aus. Geben Sie dem einen Befehlsblock den Befehl `/say !1!` und dem anderen den Befehl `/say !2!`.

Zwei Befehlsblöcke schalten die Blink-LED ein oder aus

Das Programm

Das Programm `mc11.ino` schaltet über zwei verschiedene Steuerungskommandos die Blink-LED ein oder aus. Für das Blinken selbst ist kein Programm erforderlich. Die Blink-LED blinkt automatisch, wenn sie mit Strom versorgt wird.

```
void setup()
{
  pinMode(10, OUTPUT);
  Serial.begin(9600);
}

void loop()
{
  int cmd = Serial.parseInt();
  if(cmd == 1)
  {
    digitalWrite(10, 1);
  }
  if(cmd == 2)
  {
    digitalWrite(10, 0);
  }
}
```

So funktioniert das Programm

Dieses Programm enthält nur bereits bekannte Elemente. Das Steuerungskommando `1` schaltet die LED am Pin 10 ein, das Steuerungskommando `2` schaltet sie wieder aus.

Für Ihre Notizen

12. TAG

Heute im Adventskalender

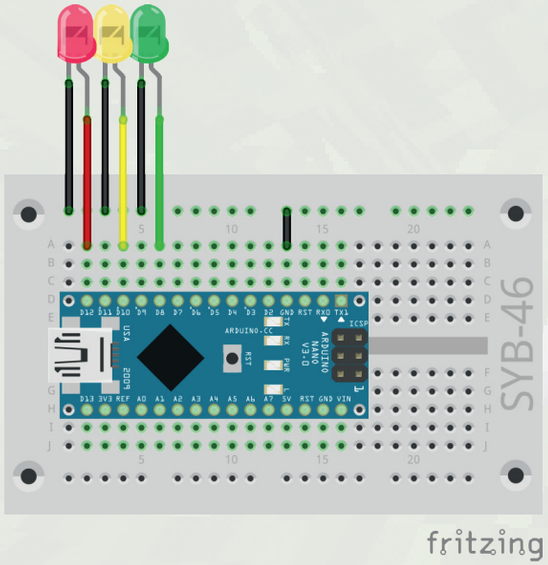
- 2 Verbindungskabel

Drei LEDs auf dem Minecraft-Klotz

Im Laufe der nächsten Tage werden Sie eine sogenannte Firmware für das Nano-Board erstellen, mit der sich vielfältige Aufgaben erledigen lassen. Einige andere Programmierschnittstellen verwenden das gleiche Prinzip. Die Firmware braucht nur einmal auf das Nano-Board aufgespielt zu werden und enthält alle benötigten Befehle zur Ansteuerung von Elektronik. Alles Weitere erfolgt auf dem PC, ohne dass jedes Mal ein neuer Sketch auf das Nano-Board übertragen werden muss.

Bauteile

- 1 Nano-Board
- 1 Steckbrett
- 1 LED rot mit Vorwiderstand
- 1 LED gelb mit Vorwiderstand
- 1 LED grün mit Vorwiderstand
- 6 Verbindungskabel
- 1 Drahtbrücke



Drei LEDs auf dem Minecraft-Klotz

Aufbau in Minecraft

Graben Sie in der Minecraft-Welt zwei Reihen von je drei Befehlsblöcken und vor jeden Befehlsblock einen Steinblock mit einem Knopf darauf in den Boden ein. Das Eingraben hat den Vorteil, dass Sie leichter mit der Spielfigur von einer Reihe zur anderen laufen können.

Geben Sie den Befehlsblöcken in der vorderen Reihe die Befehle `/say !121!`, `/say !101!`, `/say !081!`. Diese Befehle sollen die LEDs an den Pins 12, 10, 8 einschalten. Da es bei Befehlen, die mit einer 8 beginnen, zu Fehlinterpretationen als Oktalzahl kommen kann, verwenden wir den Befehl `/say !081!`, der vom Programm als Ganzzahl `81` ausgewertet wird.

Geben Sie den Befehlsblöcken in der hinteren Reihe die Befehle `/say !120!`, `/say !100!`, `/say !080!`. Diese Befehle sollen die LEDs an den Pins 12, 10, 8 ausschalten. Jeder Befehl setzt sich aus einer zweistelligen Pinnummer und einer `1` zum Einschalten oder einer `0` zum Ausschalten zusammen.

Befehlsblöcke schalten die drei LEDs ein und aus.

Das Programm

Das Programm `mc12.ino` stellt den ersten Schritt zu dieser Firmware dar. Es ermöglicht, drei LEDs an den Pins 8, 10, 12 über Steuerungskommandos ein oder auszuschalten.

```
void setup()
{
  Serial.begin(9600);
  pinMode(8, OUTPUT);
  pinMode(10, OUTPUT);
  pinMode(12, OUTPUT);
}

void loop()
{
  int cmd = Serial.parseInt();
  if(cmd == 81)
  {
    digitalWrite(8, 1);
  }
  if(cmd == 80)
  {
    digitalWrite(8, 0);
  }
  if(cmd == 101)
  {
    digitalWrite(10, 1);
  }
  if(cmd == 100)
  {
    digitalWrite(10, 0);
  }
  if(cmd == 121)
  {
    digitalWrite(12, 1);
  }
  if(cmd == 120)
  {
    digitalWrite(12, 0);
  }
}
```

So funktioniert das Programm

Das Programm funktioniert nach dem bekannten Prinzip. Die Prozedur `void setup()` definiert die verwendeten Pins als Ausgänge. Die Prozedur `void loop()` wertet die Steuerungskommandos aus und schaltet die jeweilige LED ein oder aus.

13. TAG

Heute im Adventskalender

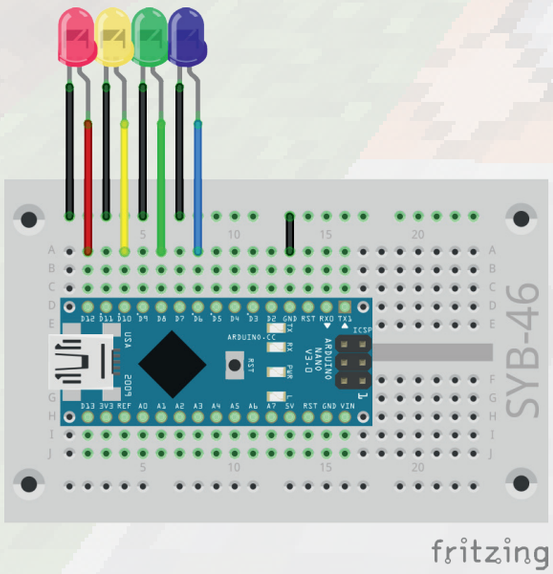
- 2 Verbindungskabel

Vier LEDs auf dem Minecraft-Klotz

Das Experiment des 13. Tages erweitert die allgemeine Firmware um eine zusätzliche LED sowie um Kommandos, die die LEDs einfach umschalten. Eine ausgeschaltete LED wird eingeschaltet, eine eingeschaltete mit dem gleichen Steuerungskommando ausgeschaltet.

Bauteile

- 1 Nano-Board
- 1 Steckbrett
- 1 LED rot mit Vorwiderstand
- 1 LED gelb mit Vorwiderstand
- 1 LED grün mit Vorwiderstand
- 1 LED blau mit Vorwiderstand
- 8 Verbindungskabel
- 1 Drahtbrücke



Vier LEDs auf dem Minecraft-Klotz

Aufbau in Minecraft

Erweitern Sie die beiden Reihen um je einen weiteren Befehlsblock mit Steinblock und Knopf. Geben Sie diesen Befehlsblöcken die Befehle `/say !061!` und `/say !060!`. Sie sollen die LED am Pin 6 ein- bzw. ausschalten.

Bauen Sie dann eine weitere Reihe mit Befehlsblöcken, Steinblöcken und Knöpfen. Geben Sie diesen Befehlsblöcken die Befehle `/say !122!`, `/say !102!`, `/say !082!` und `/say !062!`. Sie sollen die LEDs umschalten. Die Ziffer 2 am Ende eines Steuerungskommandos steht für das Umschalten.

Weitere Befehlsblöcke schalten die vier LEDs ein und aus.

Das Programm

Das Programm `mc13.ino` ist eine Erweiterung der Firmware des vorherigen Tages. Die beiden neuen Kommandos `61` und `60` schalten die LED am Pin 6 ein und aus. Die neuen Kommandos `62`, `82`, `102` und `122` schalten die LEDs an den Pins 6, 8, 10 und 12 in den jeweils anderen Zustand um. Auch dazu werden aus früheren Programmen bekannte Funktionen verwendet.

```
void setup()
{
  Serial.begin(9600);
  pinMode(6, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(10, OUTPUT);
  pinMode(12, OUTPUT);
}

void loop()
{
  int cmd = Serial.parseInt();
  if(cmd == 61)
  {
    digitalWrite(6, 1);
  }
  if(cmd == 60)
  {
    digitalWrite(6, 0);
  }
  if(cmd == 62)
  {
    digitalWrite(6, !digitalRead(6));
  }
  if(cmd == 81)
  {
    digitalWrite(8, 1);
  }
  if(cmd == 80)
  {
    digitalWrite(8, 0);
  }
  if(cmd == 82)
  {
    digitalWrite(8, !digitalRead(8));
  }
  if(cmd == 101)
  {
    digitalWrite(10, 1);
  }
  if(cmd == 100)
  {
    digitalWrite(10, 0);
  }
  if(cmd == 102)
  {
    digitalWrite(10, !digitalRead(10));
  }
  if(cmd == 121)
  {
    digitalWrite(12, 1);
  }
  if(cmd == 120)
  {
    digitalWrite(12, 0);
  }
  if(cmd == 122)
  {
    digitalWrite(12, !digitalRead(12));
  }
}
```

14. TAG

Heute im Adventskalender

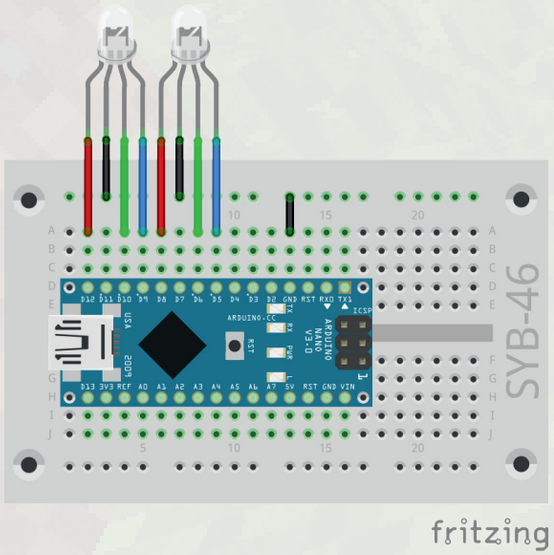
- 1 RGB-LED mit Vorwiderstand

RGB-Farbspiele auf dem Minecraft-Klotz

Das Experiment des 14. Tages schaltet zwei RGB-LEDs auf dem Minecraft-Klotz über insgesamt 18 Schalter in der Minecraft-Welt auf beliebige Farbkombinationen.

Bauteile

- 1 Nano-Board
- 1 Steckbrett
- 2 RGB-LEDs mit Vorwiderstand
- 8 Verbindungskabel
- 1 Drahtbrücke



Zwei RGB-LEDs auf dem Minecraft-Klotz

Aufbau in Minecraft

Erweitern Sie die drei Reihen Befehlsblöcke mit Steinblöcken und Knöpfen um je zwei weitere Elemente.

Geben Sie den Befehlsblöcken in der hinteren Reihe die Befehle `/say !120!`,`/say !100!`,`/say !090!`,`/say !080!`,`/say !060!`, `/say !050!`. Diese Befehle sollen die einzelnen Farben der RGB-LEDs an den Pins 12, 10, 9, 8.6, 5 ausschalten.

Geben Sie den Befehlsblöcken in der mittleren Reihe die Befehle `/say !121!`,`/say !101!`,`/say !091!`,`/say !081!`,`/say !061!`, `/say !051!`. Diese Befehle sollen die einzelnen Farben der RGB-LEDs einschalten.

Geben Sie den Befehlsblöcken in der vorderen Reihe die Befehle `/say !122!`,`/say !102!`,`/say !092!`,`/say !082!`,`/say !062!`, `/say !052!`. Diese Befehle sollen die einzelnen Farben der RGB-LEDs umschalten.

Insgesamt 18 Befehlsblöcke schalten die Farben zweier RGB-LEDs einzeln ein oder aus.

Das Programm

Das Programm `mc14.ino` ist eine weitere Erweiterung der Firmware der vorherigen Tage. Sie unterstützt jetzt die Pins 5, 6, 8, 9, 10 und 12, an denen gleichermaßen einfarbige LEDs wie auch RGB-LEDs angesteuert werden können.

```
void setup()
{
  Serial.begin(9600);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(9, OUTPUT);
  pinMode(10, OUTPUT);
  pinMode(12, OUTPUT);
}
```

```
void loop()
{
  int cmd = Serial.parseInt();
  if(cmd == 51)
  {
    digitalWrite(5, 1);
  }
  if(cmd == 50)
  {
    digitalWrite(5, 0);
  }
  if(cmd == 52)
  {
    digitalWrite(5, !digitalRead(5));
  }
  if(cmd == 61)
  {
    digitalWrite(6, 1);
  }
  if(cmd == 60)
  {
    digitalWrite(6, 0);
  }
  if(cmd == 62)
  {
    digitalWrite(6, !digitalRead(6));
  }
  if(cmd == 81)
  {
    digitalWrite(8, 1);
  }
  if(cmd == 80)
  {
    digitalWrite(8, 0);
  }
  if(cmd == 82)
  {
    digitalWrite(8, !digitalRead(8));
  }
  if(cmd == 91)
  {
    digitalWrite(9, 1);
  }
  if(cmd == 90)
  {
    digitalWrite(9, 0);
  }
  if(cmd == 92)
  {
    digitalWrite(9, !digitalRead(9));
  }
  if(cmd == 101)
  {
    digitalWrite(10, 1);
  }
```

```
  if(cmd == 100)
  {
    digitalWrite(10, 0);
  }
  if(cmd == 102)
  {
    digitalWrite(10, !digitalRead(10));
  }
  if(cmd == 121)
  {
    digitalWrite(12, 1);
  }
  if(cmd == 120)
  {
    digitalWrite(12, 0);
  }
  if(cmd == 122)
  {
    digitalWrite(12, !digitalRead(12));
  }
}
```

Für Ihre Notizen

15. TAG

Heute im Adventskalender

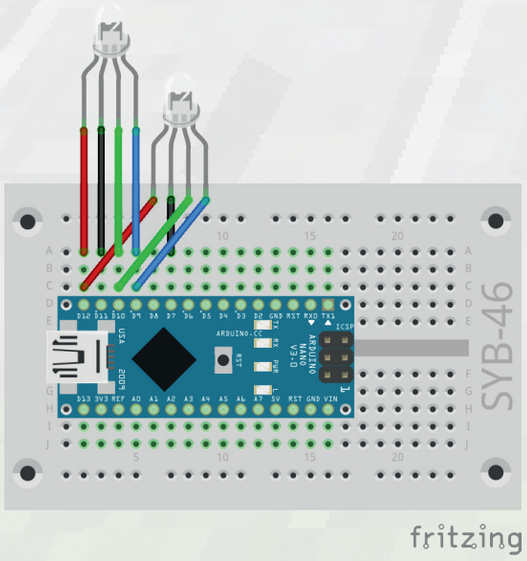
- 2 Verbindungskabel

Kathodenschaltung für LEDs und RGB-LEDs

Das Experiment des 15. Tages zeigt, wie man durch Schaltung der Kathoden Pins einspart und so auch mehr LEDs anschließen kann, als digitale Ausgangspins verfügbar sind. Die gleichfarbigen Anoden der beiden RGB-LEDs sind jeweils nur an einem gemeinsamen Pin angeschlossen. Die Kathoden sind nicht direkt mit dem GND-Pin verbunden, sondern über die beiden äußeren Schienen des Steckbretts mit den digitalen Pins 11 und 7. Diese werden auf `0` gesetzt, um als Masseleitung für die LEDs zu dienen und diese einzuschalten. Solange einer dieser Ausgänge auf `1` steht, leuchten die mit der Kathode angeschlossenen LEDs nicht.

Bauteile

- 1 Nano-Board
- 1 Steckbrett
- 2 RGB-LEDs mit Vorwiderstand
- 8 Verbindungskabel



Zwei RGB-LEDs mit Kathodenschaltung auf dem Minecraft-Klotz

Aufbau in Minecraft

Auch diesmal werden drei Reihen Befehlsblöcke zum Ein-, Aus- und Umschalten der Pins verwendet. Im Bild sind diese Reihen in je zwei Gruppen aufgeteilt. Die Dreiergruppe schaltet die drei Farbkomponenten der RGB-LEDs, die Zweiergruppe die Kathoden.

Geben Sie den Befehlsblöcken in der hinteren Reihe die Befehle `/say !120!`, `/say !100!`, `/say !090!` sowie `/say !110!`, `/say !070!`.

Geben Sie den Befehlsblöcken in der mittleren Reihe die Befehle `/say !121!`, `/say !101!`, `/say !091!` sowie `/say !111!`, `/say !071!`.

Geben Sie den Befehlsblöcken in der vorderen Reihe die Befehle `/say !122!`, `/say !102!`, `/say !092!` sowie `/say !112!`, `/say !072!`.

Befehlsblöcke für zwei RGB-LEDs mit Kathodenschaltung

Das Programm

Das Programm `mc15.ino` ist eine weitere Erweiterung der Firmware der vorherigen Tage. Sie unterstützt die Pins 5, 6, 7, 8, 9, 10, 11 und 12. Die für den heutigen Schaltungsaufbau nicht benötigten Kommandos bleiben im Programm, da es in den folgenden Schritten zu einer allgemein verwendbaren Firmware ausgebaut werden soll.

```
void setup()
{
  Serial.begin(9600);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(9, OUTPUT);
  pinMode(10, OUTPUT);
  pinMode(11, OUTPUT);
  pinMode(12, OUTPUT);
}

void loop()
{
  int cmd = Serial.parseInt();
  if(cmd == 51)
  {
    digitalWrite(5, 1);
  }
  if(cmd == 50)
  {
    digitalWrite(5, 0);
  }
  if(cmd == 52)
  {
    digitalWrite(5, !digitalRead(5));
  }
  if(cmd == 61)
  {
    digitalWrite(6, 1);
  }
  if(cmd == 60)
  {
    digitalWrite(6, 0);
  }
  if(cmd == 62)
  {
    digitalWrite(6, !digitalRead(6));
  }
  if(cmd == 71)
  {
    digitalWrite(7, 1);
  }
  if(cmd == 70)
  {
    digitalWrite(7, 0);
  }
  if(cmd == 72)
  {
    digitalWrite(7, !digitalRead(7));
  }
  if(cmd == 81)
  {
    digitalWrite(8, 1);
  }
  if(cmd == 80)
  {
    digitalWrite(8, 0);
  }
  if(cmd == 82)
  {
    digitalWrite(8, !digitalRead(8));
  }
```

```
  if(cmd == 91)
  {
    digitalWrite(9, 1);
  }
  if(cmd == 90)
  {
    digitalWrite(9, 0);
  }
  if(cmd == 92)
  {
    digitalWrite(9, !digitalRead(9));
  }
  if(cmd == 101)
  {
    digitalWrite(10, 1);
  }
  if(cmd == 100)
  {
    digitalWrite(10, 0);
  }
  if(cmd == 102)
  {
    digitalWrite(10, !digitalRead(10));
  }
  if(cmd == 111)
  {
    digitalWrite(11, 1);
  }
  if(cmd == 110)
  {
    digitalWrite(11, 0);
  }
  if(cmd == 112)
  {
    digitalWrite(11, !digitalRead(11));
  }
  if(cmd == 121)
  {
    digitalWrite(12, 1);
  }
  if(cmd == 120)
  {
    digitalWrite(12, 0);
  }
  if(cmd == 122)
  {
    digitalWrite(12, !digitalRead(12));
  }
}
```

Für Ihre Notizen

16. TAG

Heute im Adventskalender

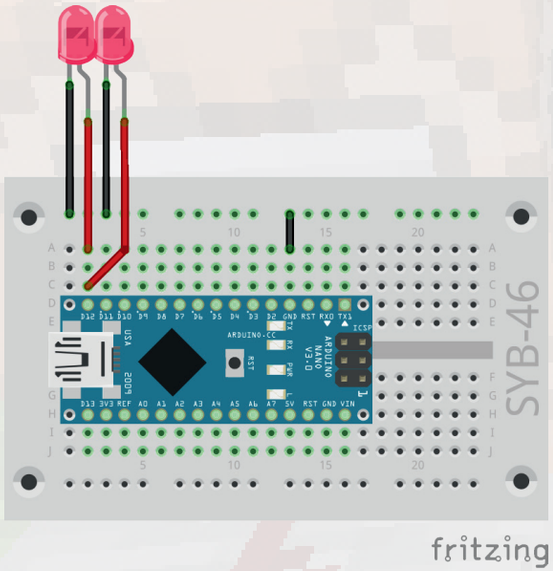
- 1 LED rot mit Vorwiderstand

LEDs mit Redstone steuern

Das Experiment des 16. Tages lässt zwei LEDs aufblinken und schaltet sie nach kurzer Zeit automatisch wieder aus. Dabei wird die in den letzten Tagen bereits entwickelte Firmware verwendet. Das Ein- und Ausschalten erfolgt nicht über Befehle im Arduino-Sketch, sondern ausschließlich über Befehlsblöcke in Minecraft.

Bauteile

- 1 Nano-Board
- 1 Steckbrett
- 2 LEDs rot mit Vorwiderstand
- 4 Verbindungskabel
- 1 Drahtbrücke



Zwei rote LEDs auf dem Minecraft-Klotz

Der Schaltungsaufbau zeigt, wie zwei LEDs parallel an einem Pin des Nano-Boards angeschlossen werden können.

Aufbau in Minecraft

Zum Ein- und Ausschalten der LEDs werden verschiedene Befehlsblöcke verwendet, da ein Befehlsblock immer nur einen Befehl senden kann.

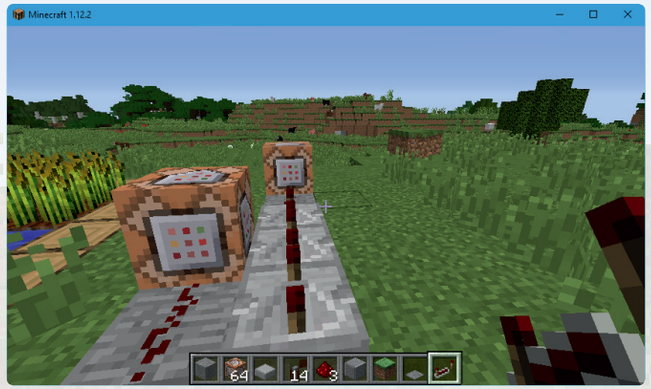
Zwei Befehlsblöcke mit Redstone-Leitungen und Verstärkern

Von dem Steinblock, auf dem der Schaltknopf angebracht ist, geht eine Redstone-Leitung geradeaus zu einem Befehlsblock, der über den Befehl `/say !121!` die LEDs am Pin 12 einschaltet.

Nach kurzem Aufblinken soll die LED wieder ausgeschaltet werden. Dazu führt eine zweite Redstone-Leitung vom Schaltknopf zum zweiten Befehlsblock. Die Leitung ist etwas länger. Bauen Sie hier hintereinander drei Redstone-Verstärker ein. Diese finden Sie im Inventar unter *Redstone*.

Redstone-Verstärker im Inventar

Achten Sie beim Einbau auf die Richtung. Das Signal muss entlang der Richtung des Pfeils auf dem Verstärker laufen. Da Minecraft keine Möglichkeit bietet, einen gebauten Block zu drehen, stellen Sie sich mit der Spielfigur so auf, dass Sie den Redstone-Verstärker nach vorne von sich weg bauen.



In dieser Richtung werden die Redstone-Verstärker gebaut.

Redstone-Verstärker verstärken das Signal nicht nur, sie können auch als Verzögerer wirken. Klicken Sie dazu mit der rechten Maustaste auf den Verstärker. Bei jedem Rechtsklick werden die beiden Fackeln auf dem Verstärker einen Schritt auseinander geschoben. Insgesamt sind vier Stufen möglich. Der größte Abstand der Fackeln bewirkt die längste Verzögerung. Durch Verkettung mehrerer Redstone-Verstärker lassen sich noch längere Verzögerungen erreichen.

Das Programm

Das Programm `mc16.ino` ist eine weitere Erweiterung der Firmware der vorherigen Tage. Sie unterstützt alle Pins von 2 bis 12, obwohl der heutige Schaltungsaufbau nur einen Pin benötigt.

```
void setup()
{
  Serial.begin(38400);
  Serial.setTimeout(100);
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(9, OUTPUT);
  pinMode(10, OUTPUT);
  pinMode(11, OUTPUT);
  pinMode(12, OUTPUT);
}

void loop()
{
  int cmd = Serial.parseInt();
  if(cmd == 21)
  {
```

```
    digitalWrite(2, 1);
  }
  if(cmd == 20)
  {
    digitalWrite(2, 0);
  }
  if(cmd == 22)
  {
    digitalWrite(2, !digitalRead(2));
  }
  if(cmd == 31)
  {
    digitalWrite(3, 1);
  }
  if(cmd == 30)
  {
    digitalWrite(3, 0);
  }
  if(cmd == 32)
  {
    digitalWrite(3, !digitalRead(3));
  }
  if(cmd == 41)
  {
    digitalWrite(4, 1);
  }
  if(cmd == 40)
  {
    digitalWrite(4, 0);
  }
  if(cmd == 42)
  {
    digitalWrite(4, !digitalRead(4));
  }
  if(cmd == 51)
  {
    digitalWrite(5, 1);
  }
  if(cmd == 50)
  {
    digitalWrite(5, 0);
  }
  if(cmd == 52)
  {
    digitalWrite(5, !digitalRead(5));
  }
  if(cmd == 61)
  {
    digitalWrite(6, 1);
  }
  if(cmd == 60)
  {
    digitalWrite(6, 0);
  }
  if(cmd == 62)
  {
    digitalWrite(6, !digitalRead(6));
  }
  if(cmd == 71)
  {
    digitalWrite(7, 1);
  }
  if(cmd == 70)
  {
    digitalWrite(7, 0);
  }
  if(cmd == 72)
  {
    digitalWrite(7, !digitalRead(7));
  }
  if(cmd == 81)
  {
    digitalWrite(8, 1);
  }
  if(cmd == 80)
  {
    digitalWrite(8, 0);
  }
  if(cmd == 82)
  {
    digitalWrite(8, !digitalRead(8));
  }
  if(cmd == 91)
  {
    digitalWrite(9, 1);
  }
  if(cmd == 90)
  {
    digitalWrite(9, 0);
  }
  if(cmd == 92)
  {
    digitalWrite(9, !digitalRead(9));
  }
  if(cmd == 101)
  {
    digitalWrite(10, 1);
  }
  if(cmd == 100)
  {
    digitalWrite(10, 0);
  }
  if(cmd == 102)
  {
    digitalWrite(10, !digitalRead(10));
  }
  if(cmd == 111)
  {
    digitalWrite(11, 1);
  }
  if(cmd == 110)
  {
    digitalWrite(11, 0);
  }
  if(cmd == 112)
  {
    digitalWrite(11, !digitalRead(11));
  }
  if(cmd == 121)
  {
    digitalWrite(12, 1);
  }
  if(cmd == 120)
  {
    digitalWrite(12, 0);
  }
  if(cmd == 122)
  {
    digitalWrite(12, !digitalRead(12));
  }
}
```

So funktioniert das Programm

Da die Funktion `Serial.parseInt()` standardmäßig nach dem letzten empfangenen Zeichen eine Sekunde wartet, ob nicht doch noch weitere Zeichen ankommen, sind nur relativ langsame Befehlsfolgen möglich.

```
Serial.setTimeout(100);
```

Diese Zeile verkürzt den sogenannten Timeout auf 100 Millisekunden, was immer noch ausreicht, um die Steuerungskommandos des AMI sauber voneinander zu trennen. Damit der verkürzte Timeout funktioniert, muss vorher bei der Initialisierung die Übertragungsgeschwindigkeit der seriellen Schnittstelle erhöht werden.

```
Serial.begin(38400);
```

Beachten Sie dabei, dass die Geschwindigkeit nicht stufenlos geregelt, sondern von 9600 ausgehend immer nur verdoppelt werden kann. Erhöhen Sie die Geschwindigkeit nicht beliebig, da es zu Übertragungsfehlern kommen kann, besonders wenn noch andere USB-Geräte angeschlossen sind. 38400 ist ein guter Wert, der reaktionsschnell genug ist und normalerweise noch keine Übertragungsfehler verursacht.

Starten Sie das Programm `AMI.exe` neu und geben Sie hier ebenfalls den neuen Wert 38400 ein. Danach muss Minecraft auch neu gestartet werden.

Klicken sie in Minecraft mit der rechten Maustaste auf den Knopf. Die LED wird eingeschaltet. Es ist zu sehen, wie das Redstone-Signal durch die Verstärker fließt und am Ende über den anderen Befehlsblock die LED wieder ausschaltet.

17. TAG

Heute im Adventskalender

- 2 Verbindungskabel

LEDs dimmen

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 %.

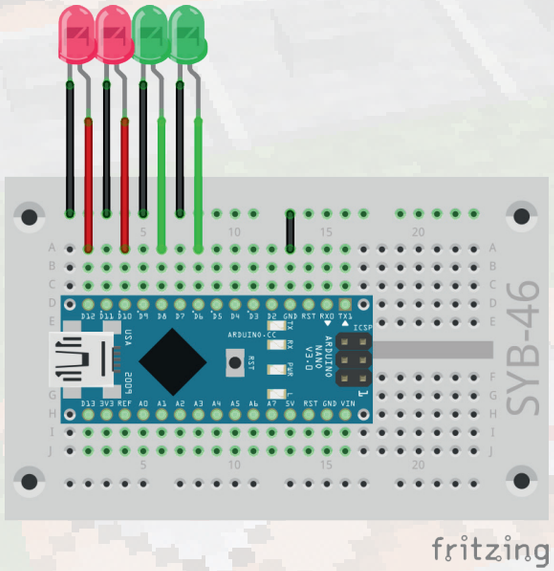
Je kleiner das Tastverhältnis, desto kürzer ist die Leuchtzeit der LED innerhalb eines Schaltzyklus. Dadurch wirkt die LED dunkler als eine permanent eingeschaltete LED.

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. Die anderen digitalen Pins lassen sich nur ein- und ausschalten.

Bauteile

- 1 Nano-Board
- 1 Steckbrett
- 2 LEDs rot mit Vorwiderstand
- 2 LEDs grün mit Vorwiderstand
- 8 Verbindungskabel
- 1 Drahtbrücke



Vier LEDs auf dem Minecraft-Klotz

Aufbau in Minecraft

Dieses Experiment erfordert keinen speziellen Aufbau in Minecraft. Verwenden Sie einen der bereits vorhandenen Knöpfe und geben Sie dem zugehörigen Befehlsblock den Befehl `/say !1!`.

Einer der vorhandenen Befehlsblöcke dimmt die LEDs.

Das Programm

Das Programm `mc17.ino` verwendet nicht die in den letzten Tagen erstellte Firmware, sondern ist komplett eigenständig. Es ändert die Helligkeit der LEDs an Pin 6 und Pin 10 zyklisch. Die LEDs an Pin 8 und Pin 12 leuchten zum Vergleich in voller Helligkeit.

```
void setup()
{
  Serial.begin(38400);
  Serial.setTimeout(100);
  pinMode(6, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(10, OUTPUT);
  pinMode(12, OUTPUT);
}

void loop()
{
  int cmd = Serial.parseInt();
  if(cmd == 1)
  {
    digitalWrite(12, 1);
    digitalWrite(8, 1);
    for(int i=0; i<256; i++)
    {
      analogWrite(10, i);
      analogWrite(6, i);
      delay(20);
    }
    delay(200);
    digitalWrite(12, 0);
    digitalWrite(10, 0);
    digitalWrite(8, 0);
    digitalWrite(6, 0);
  }
}
```

So funktioniert das Programm

Die Prozedur `void loop()` prüft, ob das Steuerungskommando `1` übertragen wurde. In diesem Fall werden zunächst die LEDs an den Pins 12 und 8 eingeschaltet.

Danach startet eine Schleife, die die Helligkeit der LEDs an den Pins 10 und 6 zyklisch ändert. Dazu wird die Funktion `analogWrite()` verwendet, 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 %. Dann leuchtet die LED mit voller Stärke. Zwischen zwei PWM-Werten wartet das Programm 20 Millisekunden.

Am Ende des letzten Schleifendurchlaufs wird noch einmal 200 Millisekunden gewartet, bevor alle vier LEDs wieder ausgeschaltet werden.

18. TAG

Heute im Adventskalender

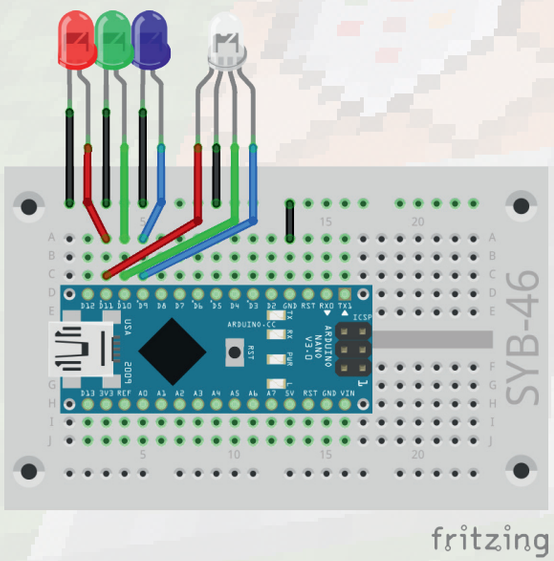
- 2 Verbindungskabel

RGB-Farbmischung

Steuert man RGB-LEDs über drei PWM-Signale an, lassen sich noch deutlich mehr Mischfarben erzeugen als nur durch Ein- und Ausschalten der drei Farbkomponenten. Drei einfarbige LEDs auf dem Minecraft-Klotz zeigen zusätzlich, welche Farbkomponenten der RGB-LED eingeschaltet sind.

Bauteile

- 1 Nano-Board
- 1 Steckbrett
- 1 LED rot mit Vorwiderstand
- 1 LED grün mit Vorwiderstand
- 1 LED blau mit Vorwiderstand
- 1 RGB-LED mit Vorwiderstand
- 10 Verbindungskabel
- 1 Drahtbrücke



Drei LEDs und eine RGB-LED auf dem Minecraft-Klotz

Aufbau in Minecraft

Zur Steuerung der RGB-LEDs werden die drei bereits aus einem früheren Experiment bekannten Dreierreihen von Befehlsblöcken verwendet.

Eine der vorhandenen Reihen von Befehlsblöcken dimmt die drei Farben der RGB-LED.

Geben Sie den Befehlsblöcken in der hinteren Reihe die Befehle `/say !110!`, `/say !100!`, `/say !090!` zum Ausschalten der drei Farben.

Geben Sie den Befehlsblöcken in der mittleren Reihe die Befehle `/say !111!`, `/say !101!`, `/say !091!` zum Einschalten der drei Farben.

Geben Sie den Befehlsblöcken in der vorderen Reihe die neuen Befehle `/say !113!`, `/say !103!`, `/say !093!` zum Dimmen der drei Farben.

Das Programm

Das Programm `mc18.ino` ist eine weitere Erweiterung der Firmware der vorherigen Tage. Sie unterstützt weiterhin alle Pins von 2 bis 12. Zusätzlich gibt es die drei Kommandos `93`, `103`, `113`, die LEDs an den PWM-Pins 9, 10, 11 langsam hochdimmen.

```
void setup()
{
  Serial.begin(38400);
  Serial.setTimeout(100);
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(9, OUTPUT);
  pinMode(10, OUTPUT);
  pinMode(11, OUTPUT);
  pinMode(12, OUTPUT);
}

void loop()
{
  int cmd = Serial.parseInt();
  if(cmd == 21)
  {
    digitalWrite(2, 1);
  }
  if(cmd == 20)
  {
    digitalWrite(2, 0);
  }
  if(cmd == 22)
  {
    digitalWrite(2, !digitalRead(2));
  }
  if(cmd == 31)
  {
    digitalWrite(3, 1);
  }
  if(cmd == 30)
  {
    digitalWrite(3, 0);
  }
  if(cmd == 32)
  {
    digitalWrite(3, !digitalRead(3));
  }
  if(cmd == 41)
  {
    digitalWrite(4, 1);
  }
  if(cmd == 40)
  {
    digitalWrite(4, 0);
  }
  if(cmd == 42)
  {
    digitalWrite(4, !digitalRead(4));
  }
  if(cmd == 51)
  {
    digitalWrite(5, 1);
  }
  if(cmd == 50)
  {
    digitalWrite(5, 0);
  }
  if(cmd == 52)
  {
    digitalWrite(5, !digitalRead(5));
  }
```

```
  if(cmd == 61)
  {
    digitalWrite(6, 1);
  }
  if(cmd == 60)
  {
    digitalWrite(6, 0);
  }
  if(cmd == 62)
  {
    digitalWrite(6, !digitalRead(6));
  }
  if(cmd == 71)
  {
    digitalWrite(7, 1);
  }
  if(cmd == 70)
  {
    digitalWrite(7, 0);
  }
  if(cmd == 72)
  {
    digitalWrite(7, !digitalRead(7));
  }
  if(cmd == 81)
  {
    digitalWrite(8, 1);
  }
  if(cmd == 80)
  {
    digitalWrite(8, 0);
  }
  if(cmd == 82)
  {
    digitalWrite(8, !digitalRead(8));
  }
  if(cmd == 91)
  {
    digitalWrite(9, 1);
  }
  if(cmd == 90)
  {
    digitalWrite(9, 0);
  }
  if(cmd == 92)
  {
    digitalWrite(9, !digitalRead(9));
  }
  if(cmd == 93)
  {
    for(int i=0; i<256; i++)
    {
      analogWrite(9, i);
      delay(20);
    }
  }
  if(cmd == 101)
  {
    digitalWrite(10, 1);
  }
  if(cmd == 100)
  {
    digitalWrite(10, 0);
  }
  if(cmd == 102)
  {
    digitalWrite(10, !digitalRead(10));
  }
  if(cmd == 103)
  {
    for(int i=0; i<256; i++)
    {
      analogWrite(10, i);
      delay(20);
    }
  }
  if(cmd == 111)
  {
    digitalWrite(11, 1);
  }
  if(cmd == 110)
  {
    digitalWrite(11, 0);
  }
  if(cmd == 112)
  {
    digitalWrite(11, !digitalRead(11));
  }
  if(cmd == 113)
  {
    for(int i=0; i<256; i++)
    {
      analogWrite(11, i);
      delay(20);
    }
  }
  if(cmd == 121)
  {
    digitalWrite(12, 1);
  }
  if(cmd == 120)
  {
    digitalWrite(12, 0);
  }
  if(cmd == 122)
  {
    digitalWrite(12, !digitalRead(12));
  }
}
```

So funktioniert das Programm

Die drei neuen Kommandos mit der Ziffer 3 am Ende funktionieren prinzipiell alle gleich. Eine Schleife macht die LEDs an den entsprechenden PWM-Pins zyklisch heller. Dazu wird die Funktion `analogWrite()` verwendet, die Werte zwischen 0 und 255 auf einen PWM-Pin schreibt.

Da die einfarbige LED und die farbgleiche Komponente der RGB-LED am gleichen Pin angeschlossen sind, werden sie auch synchron gedimmt. Mit den Schaltern der hinteren beiden Reihen lassen sich einzelne LEDs und die zugehörigen Farben der RGB-LED aus- und einschalten.

Für Ihre Notizen

19. TAG

Heute im Adventskalender

- 1 LED gelb mit Vorwiderstand

Würfel auf dem Minecraft-Klotz

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 von Spielwürfeln. Diese haben Augen in einer quadratischen Anordnung, wozu man sieben LEDs braucht.

Bauen Sie für das folgende Experiment sieben LEDs wie abgebildet auf dem Minecraft-Klotz auf. Für die Ansteuerung der LEDs werden nur vier statt sieben Pins benötigt, da ein Würfel zur Darstellung gerader Zahlen die Augen paarweise nutzt.

Bauteile

- 1 Nano-Board
- 1 Steckbrett
- 2 LEDs rot mit Vorwiderstand
- 2 LEDs gelb mit Vorwiderstand
- 2 LEDs grün mit Vorwiderstand
- 1 LED blau mit Vorwiderstand
- 14 Verbindungskabel
- 1 Drahtbrücke

Aufbau in Minecraft

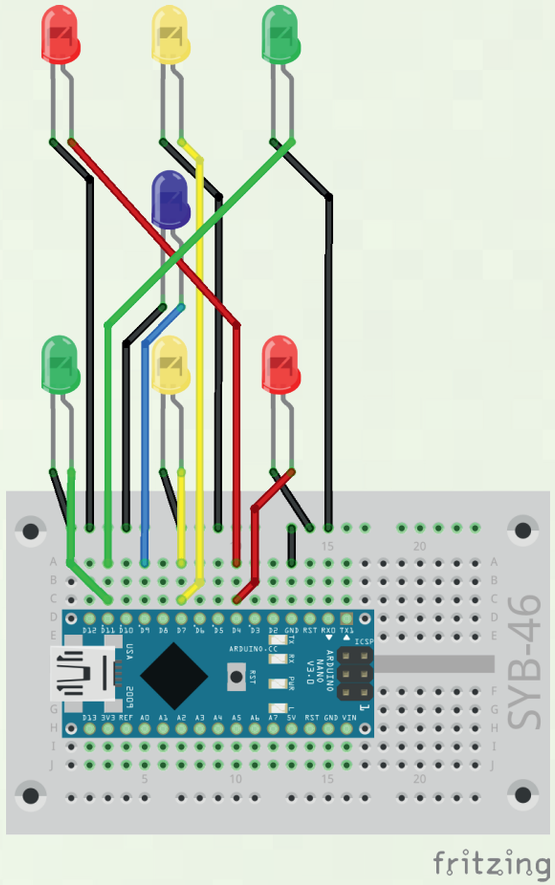
Für dieses Experiment ist kein besonderer Aufbau in Minecraft nötig. Geben Sie einem der vorhandenen Befehlsblöcke den Befehl `/say !1!`. Damit wird das Würfelprogramm gestartet.

Das Programm

Die Würfelwürfe werden direkt auf dem Nano-Board errechnet und ausgewertet. Minecraft startet nur wie üblich über AMI die Prozedur dazu. Das Programm `mc19.ino` basiert nicht auf der in den letzten Tagen entwickelten Firmware, sondern ist komplett eigenständig.

```
void setup()
{
  Serial.begin(38400);
  Serial.setTimeout(100);
  pinMode(4, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(9, OUTPUT);
  pinMode(11, OUTPUT);
}

void loop()
{
  int cmd = Serial.parseInt();
  if(cmd == 1)
  {
    digitalWrite(4, 0);
    digitalWrite(7, 0);
    digitalWrite(9, 0);
    digitalWrite(11, 0);
    int z = random(6) + 1;
    switch(z)
    {
      case 1:
        digitalWrite(9, 1);
        break;
      case 2:
        digitalWrite(4, 1);
        break;
      case 3:
        digitalWrite(9, 1);
        digitalWrite(11, 1);
        break;
      case 4:
        digitalWrite(4, 1);
        digitalWrite(11, 1);
        break;
      case 5:
        digitalWrite(9, 1);
        digitalWrite(4, 1);
        digitalWrite(11, 1);
        break;
      case 6:
        digitalWrite(4, 1);
        digitalWrite(11, 1);
        digitalWrite(7, 1);
        break;
      default:
        digitalWrite(4, 0);
        digitalWrite(7, 0);
        digitalWrite(9, 0);
        digitalWrite(11, 0);
    }
  }
}
```



Würfel aus sieben LEDs auf dem Minecraft-Klotz

So funktioniert das Programm

Wenn das Programm auf dem Nano-Board das Steuerungskommando `1` empfängt, werden zuerst alle vier verwendeten Pins auf 0 gesetzt und damit das zuletzt angezeigte Würfelergebnis gelöscht.

Danach wird eine Zufallszahl zwischen 1 und 6 erzeugt. Die Funktion `random(6)` erzeugt eine Zufallszahl zwischen 0 und 5. Danach wird 1 addiert und das Ergebnis in der Variable `z` gespeichert.

Abhängig vom Wert von `z` lässt eine `switch...case` Konstruktion verschiedene Gruppen von LEDs aufleuchten. Jede `case`-Gruppe steht für einen Zufallswert. Eine `default`-Gruppe muss immer angegeben werden. Dieser Block wird ausgeführt, wenn keiner der vorgegebenen Fälle eintritt, was in diesem Programm theoretisch nicht passieren kann.

20. TAG

Heute im Adventskalender

- 2 Verbindungskabel

Würfel mit zusätzlicher Blink-LED auf dem Minecraft-Klotz

In dieser erweiterten Form ist auf dem Minecraft-Klotz zusätzlich eine Blink-LED aufgesteckt, die vor der Anzeige des Würfelergebnisses eine kurze Zeit lang blinkt, um die Spannung zu erhöhen.

Bauteile

- 1 Nano-Board
- 1 Steckbrett
- 2 LEDs rot mit Vorwiderstand
- 2 LEDs gelb mit Vorwiderstand
- 2 LEDs grün mit Vorwiderstand
- 1 LED blau mit Vorwiderstand
- 1 Blink-LED mit Vorwiderstand
- 16 Verbindungskabel
- 1 Drahtbrücke

Aufbau in Minecraft

Der Aufbau in Minecraft ist der gleiche wie am vorhergehenden Tag. Geben Sie wieder einem der vorhandenen Befehlsblöcke den Befehl `/say !1!`. Damit wird das Würfelprogramm gestartet.

Das Programm

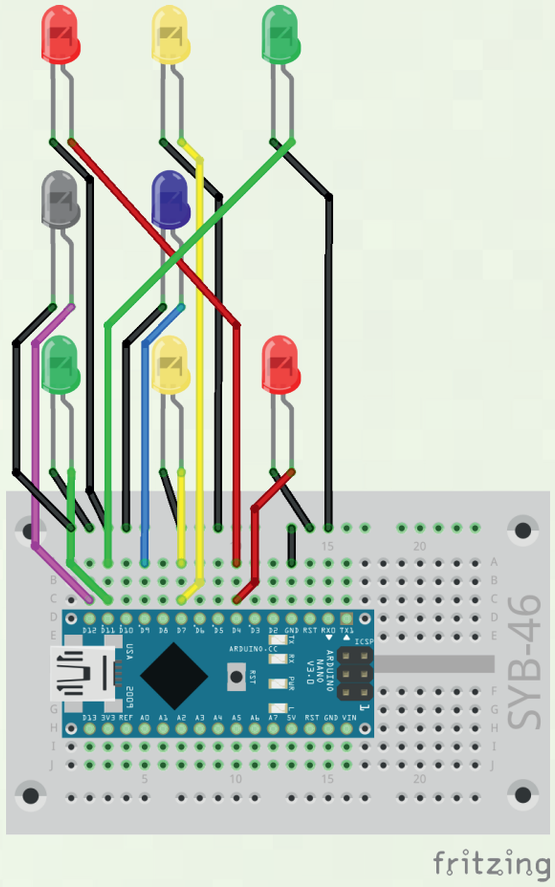
Das Programm `mc20.ino` ist gegenüber dem vorherigen bis auf wenige Programmzeilen weitgehend unverändert.

```
void setup()
{
  Serial.begin(38400);
  Serial.setTimeout(100);
  pinMode(4, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(9, OUTPUT);
  pinMode(11, OUTPUT);
  pinMode(12, OUTPUT);
}
```

```
void loop()
{
  int cmd = Serial.parseInt();
  if(cmd == 1)
  {
    digitalWrite(4, 0);
    digitalWrite(7, 0);
    digitalWrite(9, 0);
    digitalWrite(11, 0);
    digitalWrite(12, 1);
    delay(1000);
    digitalWrite(12, 0);
    int z = random(6) + 1;
    switch(z)
    {
      case 1:
        digitalWrite(9, 1);
        break;
      case 2:
        digitalWrite(4, 1);
        break;
      case 3:
        digitalWrite(9, 1);
        digitalWrite(11, 1);
        break;
      case 4:
        digitalWrite(4, 1);
        digitalWrite(11, 1);
        break;
      case 5:
        digitalWrite(9, 1);
        digitalWrite(4, 1);
        digitalWrite(11, 1);
        break;
      case 6:
        digitalWrite(4, 1);
        digitalWrite(11, 1);
        digitalWrite(7, 1);
        break;
      default:
        digitalWrite(4, 0);
        digitalWrite(7, 0);
        digitalWrite(9, 0);
        digitalWrite(11, 0);
    }
  }
}
```

So funktioniert das Programm

Der entscheidende Unterschied zeigt sich, nachdem vor einem Würfelwurf alle LEDs ausgeschaltet wurden. Dann leuchtet die Blink-LED für eine Sekunde. In dieser Zeit rollt weder ein Würfel, noch rechnet das Programm irgendetwas aus. Um die Spannung zu erhöhen, wird die Blink-LED eine Sekunde lang mit Spannung versorgt und blinkt selbstständig.



Würfel aus sieben LEDs und einer Blink-LED auf dem Minecraft-Klotz

21. TAG

Heute im Adventskalender

- 2 Verbindungskabel

LED-Würfel mit realistischem Würfeleffekt

Ein wirklicher Würfel zeigt nicht sofort die endgültige Augenzahl an, sondern rollt erst eine kurze Zeit, in der man Würfelergebnisse sieht, die sich dann aber doch nicht bewahrheiten. Das Programm des 21. Tages 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 Nano-Board
- 1 Steckbrett
- 2 LEDs rot mit Vorwiderstand
- 2 LEDs gelb mit Vorwiderstand
- 2 LEDs grün mit Vorwiderstand
- 1 LED blau mit Vorwiderstand
- 14 Verbindungskabel
- 1 Drahtbrücke

Aufbau in Minecraft

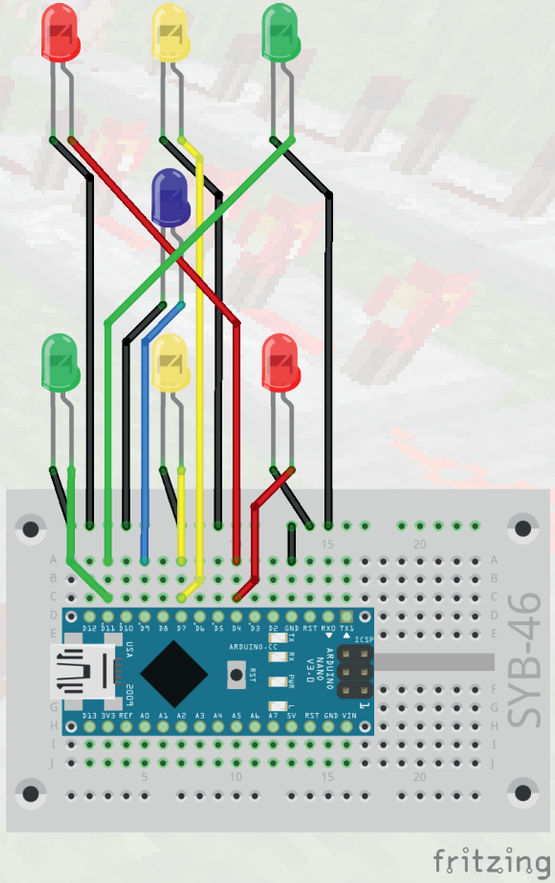
Das Rollen des Würfels wird über Redstone-Elemente simuliert. Das über den Druckknopf ausgelöste Redstone-Signal fließt über eine Leitung an insgesamt vier Befehlsblöcke, die alle das gleiche Steuerungskommando zum Würfeln auslösen. Zwischen den einzelnen Befehlsblöcken liegen unterschiedlich viele Redstone-Verzögerer, damit es so aussieht, als ob der Würfel langsam ausrollte.

Redstone-Schaltung mit vier Befehlsblöcken und Verzögerern

Das Programm

Das Programm `mc21.ino` entspricht dem Programm des 19. Tages, dem einfachen Würfel ohne Blink-LED. Während das Redstone-Signal durch die Leitung fließt, sendet es viermal das Steuerungskommando `1`. Dabei würfelt das Programm jedes Mal eine neue Zahl. Natürlich kann es passieren, dass zweimal hintereinander die gleiche Zahl gewürfelt wird, was für den Benutzer so aussieht, als ob der Würfel liegen bleibe.

Das Redstone-Signal fließt durch die Leitung



Würfel aus sieben LEDs auf dem Minecraft-Klotz

22. TAG

Heute im Adventskalender

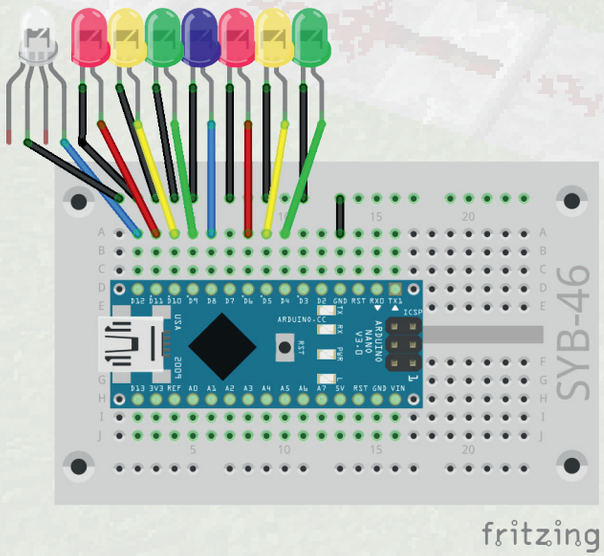
- 2 Verbindungskabel

Lauflicht mit Redstone steuern

Insgesamt acht LEDs auf dem Minecraft-Klotz leuchten als Lauflicht. Außer den sieben einfarbigen LEDs wird eine RGB-LED verwendet, von der nur die blaue Farbkomponente angeschlossen ist. In diesem Experiment steuert nicht das Programm auf dem Nano-Board das Aufblinken der einzelnen LEDs hintereinander, sondern ein Redstone-Signal.

Bauteile

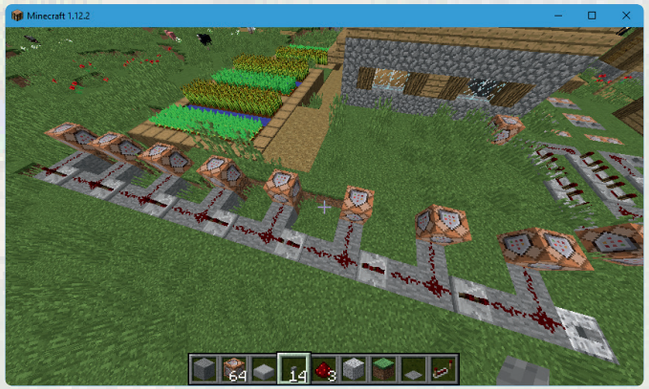
- 1 Nano-Board
- 1 Steckbrett
- 2 LEDs rot mit Vorwiderstand
- 2 LEDs gelb mit Vorwiderstand
- 2 LEDs grün mit Vorwiderstand
- 1 LED blau mit Vorwiderstand
- 1 RGB-LED mit Vorwiderstand
- 16 Verbindungskabel
- 1 Drahtbrücke



Lauflicht aus acht LEDs auf dem Minecraft-Klotz

Aufbau in Minecraft

Das Redstone-Signal läuft auf einer Leitung, an der acht Befehlsblöcke angeschlossen sind, die jeder eine LED ansteuern. Zwischen diesen Befehlsblöcken ist je ein Redstone-Verzögerer eingebaut. Geben Sie den Befehlsblöcken der Reihe nach die Befehle `/say !044!` bis `/say !124!`.



Die Redstone-Leitung zur Steuerung des Laufllchts

Die Verzögerer sind auf Stufe 2 eingestellt

Beim Drücken des Knopfes werden nacheinander die Steuerungskommandos 044 bis 124 gesendet, die die LEDs 4 bis 12 aufleuchten lassen

Das Programm

Das Programm `mc22.ino` ist eine weitere Erweiterung der Firmware der vorherigen Tage. Sie unterstützt weiterhin alle Pins von 2 bis 12. Zusätzlich gibt es für jeden Pin Kommandos mit der Endziffer 4, die den jeweiligen Pin als einzigen einschalten. Alle anderen Pins werden ausgeschaltet. So reicht beim Lauflicht für jeden Schritt ein einziges Steuerungskommando.

```
void setup()
{
  Serial.begin(38400);
  Serial.setTimeout(100);
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(9, OUTPUT);
  pinMode(10, OUTPUT);
  pinMode(11, OUTPUT);
  pinMode(12, OUTPUT);
}

void loop()
{
  int cmd = Serial.parseInt();
  if(cmd == 21)
  {
    digitalWrite(2, 1);
  }
  if(cmd == 20)
  {
```

```
  digitalWrite(2, 0);
}
if(cmd == 22)
{
  digitalWrite(2, !digitalRead(2));
}
if(cmd == 24)
{
  digitalWrite(2, 1);
  digitalWrite(3, 0);
  digitalWrite(4, 0);
  digitalWrite(5, 0);
  digitalWrite(6, 0);
  digitalWrite(7, 0);
  digitalWrite(8, 0);
  digitalWrite(9, 0);
  digitalWrite(10, 0);
  digitalWrite(11, 0);
  digitalWrite(12, 0);
}
if(cmd == 31)
{
  digitalWrite(3, 1);
}
if(cmd == 30)
{
  digitalWrite(3, 0);
}
if(cmd == 32)
{
  digitalWrite(3, !digitalRead(3));
}
if(cmd == 34)
{
  digitalWrite(2, 0);
  digitalWrite(3, 1);
  digitalWrite(4, 0);
  digitalWrite(5, 0);
  digitalWrite(6, 0);
  digitalWrite(7, 0);
  digitalWrite(8, 0);
  digitalWrite(9, 0);
  digitalWrite(10, 0);
  digitalWrite(11, 0);
  digitalWrite(12, 0);
}
if(cmd == 41)
{
  digitalWrite(4, 1);
}
if(cmd == 40)
{
  digitalWrite(4, 0);
}
if(cmd == 42)
{
  digitalWrite(4, !digitalRead(4));
}
if(cmd == 44)
{
  digitalWrite(2, 0);
  digitalWrite(3, 0);
  digitalWrite(4, 1);
  digitalWrite(5, 0);
  digitalWrite(6, 0);
  digitalWrite(7, 0);
  digitalWrite(8, 0);
  digitalWrite(9, 0);
  digitalWrite(10, 0);
  digitalWrite(11, 0);
  digitalWrite(12, 0);
}
if(cmd == 51)
{
  digitalWrite(5, 1);
}
if(cmd == 50)
{
  digitalWrite(5, 0);
}
if(cmd == 52)
{
  digitalWrite(5, !digitalRead(5));
}
if(cmd == 54)
{
  digitalWrite(2, 0);
  digitalWrite(3, 0);
  digitalWrite(4, 0);
  digitalWrite(5, 1);
  digitalWrite(6, 0);
  digitalWrite(7, 0);
  digitalWrite(8, 0);
  digitalWrite(9, 0);
  digitalWrite(10, 0);
  digitalWrite(11, 0);
  digitalWrite(12, 0);
}
if(cmd == 61)
{
  digitalWrite(6, 1);
}
if(cmd == 60)
{
  digitalWrite(6, 0);
}
if(cmd == 62)
{
  digitalWrite(6, !digitalRead(6));
}
if(cmd == 64)
{
  digitalWrite(2, 0);
  digitalWrite(3, 0);
  digitalWrite(4, 0);
  digitalWrite(5, 0);
  digitalWrite(6, 1);
  digitalWrite(7, 0);
  digitalWrite(8, 0);
  digitalWrite(9, 0);
  digitalWrite(10, 0);
  digitalWrite(11, 0);
  digitalWrite(12, 0);
}
if(cmd == 71)
{
  digitalWrite(7, 1);
}
if(cmd == 70)
{
  digitalWrite(7, 0);
}
if(cmd == 72)
{
  digitalWrite(7, !digitalRead(7));
}
if(cmd == 74)
{
  digitalWrite(2, 0);
  digitalWrite(3, 0);
  digitalWrite(4, 0);
  digitalWrite(5, 0);
  digitalWrite(6, 0);
  digitalWrite(7, 1);
  digitalWrite(8, 0);
  digitalWrite(9, 0);
  digitalWrite(10, 0);
  digitalWrite(11, 0);
  digitalWrite(12, 0);
}
if(cmd == 81)
{
  digitalWrite(8, 1);
}
if(cmd == 80)
{
  digitalWrite(8, 0);
}
```

```
if(cmd == 82)
{
  digitalWrite(8, !digitalRead(8));
}
if(cmd == 84)
{
  digitalWrite(2, 0);
  digitalWrite(3, 0);
  digitalWrite(4, 0);
  digitalWrite(5, 0);
  digitalWrite(6, 0);
  digitalWrite(7, 0);
  digitalWrite(8, 1);
  digitalWrite(9, 0);
  digitalWrite(10, 0);
  digitalWrite(11, 0);
  digitalWrite(12, 0);
}
if(cmd == 91)
{
  digitalWrite(8, 1);
}
if(cmd == 90)
{
  digitalWrite(9, 0);
}
if(cmd == 92)
{
  digitalWrite(9, !digitalRead(9));
}
if(cmd == 93)
{
  for(int i=0; i<256; i++)
  {
    analogWrite(9, i);
    delay(20);
  }
}
if(cmd == 94)
{
  digitalWrite(2, 0);
  digitalWrite(3, 0);
  digitalWrite(4, 0);
  digitalWrite(5, 0);
  digitalWrite(6, 0);
  digitalWrite(7, 0);
  digitalWrite(8, 0);
  digitalWrite(9, 1);
  digitalWrite(10, 0);
  digitalWrite(11, 0);
  digitalWrite(12, 0);
}
if(cmd == 101)
{
  digitalWrite(10, 1);
}
if(cmd == 100)
{
  digitalWrite(10, 0);
}
if(cmd == 102)
{
  digitalWrite(10, !digitalRead(10));
}
if(cmd == 103)
{
  for(int i=0; i<256; i++)
  {
    analogWrite(10, i);
    delay(20);
  }
}
if(cmd == 104)
{
  digitalWrite(2, 0);
  digitalWrite(3, 0);
  digitalWrite(4, 0);
  digitalWrite(5, 0);
  digitalWrite(6, 0);
  digitalWrite(7, 0);
  digitalWrite(8, 0);
  digitalWrite(9, 0);
  digitalWrite(10, 1);
  digitalWrite(11, 0);
  digitalWrite(12, 0);
}
if(cmd == 111)
{
  digitalWrite(11, 1);
}
if(cmd == 110)
{
  digitalWrite(11, 0);
}
if(cmd == 112)
{
  digitalWrite(11, !digitalRead(11));
}
if(cmd == 113)
{
  for(int i=0; i<256; i++)
  {
    analogWrite(11, i);
    delay(20);
  }
}
if(cmd == 114)
{
  digitalWrite(2, 0);
  digitalWrite(3, 0);
  digitalWrite(4, 0);
  digitalWrite(5, 0);
  digitalWrite(6, 0);
  digitalWrite(7, 0);
  digitalWrite(8, 0);
  digitalWrite(9, 0);
  digitalWrite(10, 0);
  digitalWrite(11, 1);
  digitalWrite(12, 0);
}
if(cmd == 121)
{
  digitalWrite(12, 1);
}
if(cmd == 120)
{
  digitalWrite(12, 0);
}
if(cmd == 122)
{
  digitalWrite(12, !digitalRead(12));
}
if(cmd == 124)
{
  digitalWrite(2, 0);
  digitalWrite(3, 0);
  digitalWrite(4, 0);
  digitalWrite(5, 0);
  digitalWrite(6, 0);
  digitalWrite(7, 0);
  digitalWrite(8, 0);
  digitalWrite(9, 0);
  digitalWrite(10, 0);
  digitalWrite(11, 0);
  digitalWrite(12, 1);
}
}
```

23. TAG

Heute im Adventskalender

- 1 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 dieser 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 Kilohertz dagegen als durchgehende Töne unterschiedlicher Höhe.

Töne erzeugen

Töne werden auf dem Nano-Board über ein PWM-Signal erzeugt. Die PWM-Pins auf dem Nano-Board liefern ungefähr 490 Hertz.

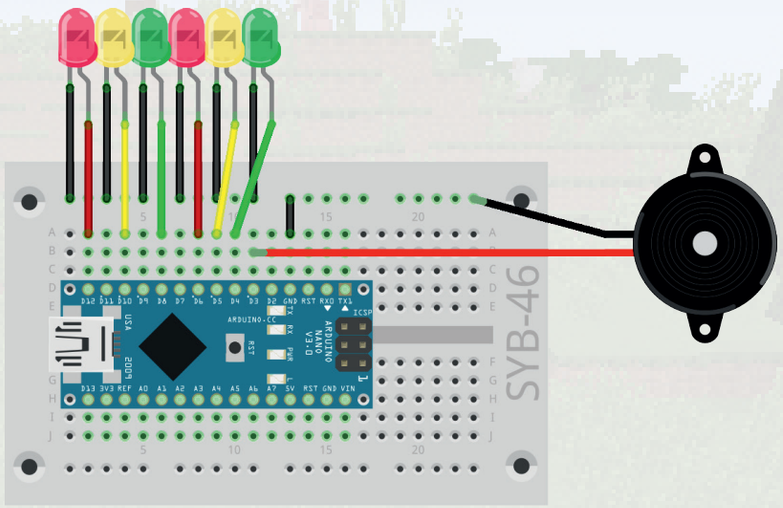
Bauteile

- 1 Nano-Board
- 1 Steckbrett
- 2 LEDs rot mit Vorwiderstand
- 2 LEDs gelb mit Vorwiderstand
- 2 LEDs grün mit Vorwiderstand
- 12 Verbindungskabel
- 1 Piezo-Summer
- 1 Drahtbrücke

Aufbau in Minecraft

Auch dieses Mal wird über ein Redstone-Signal eine Folge von LEDs gesteuert. Allerdings leuchten im Unterschied zum Lauflicht einmal eingeschaltete LEDs weiter. Nach jedem Einschalten einer LED wird das Tonsignal abwechselnd entweder ein- oder ausgeschaltet. Das Redstone-Signal fließt wieder über eine Leitung, an der Befehlsblöcke angeschlossen sind, wobei zwischen je zwei Befehlsblöcken ein Verzögerer eingebaut ist. Der Übersichtlichkeit halber sind die Befehlsblöcke, die die LEDs ansteuern, auf der einen Seite der Leitung aufgebaut, die Befehlsblöcke für die Tonsignale auf der anderen Seite.

Geben Sie den Befehlsblöcken zur Ansteuerung der sechs LEDs nacheinander die Befehle `/say !121!`, `/say !101!`, `/say !081!`, `/say !061!`, `/say !051!`, `/say !041!`.



Lauflicht auf dem Minecraft-Klotz und Töne aus dem Inneren

Geben Sie den Befehlsblöcken zur Ansteuerung des Piezo-Summers am Pin 3 nacheinander die Befehle `/say !035!`, `/say !030!`, `/say !035!`, `/say !030!`, `/say !035!`, `/say !030!`.

Die Redstone-Leitung zur Steuerung der LEDs und Tonsignale

Das Programm

Das Programm `mc23.ino` ist die letzte Erweiterung der Firmware der vorherigen Tage. Das neue Kommando `35` erzeugt am Pin 3 über ein analoges Signal einen Ton. Das neue Kommando `990` schaltet alle Pins aus, das Kommando `999` schaltet alle Pins ein.

```
void setup()
{
  Serial.begin(38400);
  Serial.setTimeout(100);
  for(int i=2; i<13; i++)
  {
    pinMode(i, OUTPUT);
  }
}

void loop()
{
  int cmd = Serial.parseInt();
  if(cmd == 21)
  {
    digitalWrite(2, 1);
  }
  if(cmd == 20)
  {
    digitalWrite(2, 0);
  }
  if(cmd == 22)
  {
    digitalWrite(2, !digitalRead(2));
  }

  if(cmd == 24)
  {
    digitalWrite(2, 1);
    digitalWrite(3, 0);
    digitalWrite(4, 0);
    digitalWrite(5, 0);
    digitalWrite(6, 0);
    digitalWrite(7, 0);
    digitalWrite(8, 0);
    digitalWrite(9, 0);
    digitalWrite(10, 0);
    digitalWrite(11, 0);
    digitalWrite(12, 0);
  }
  if(cmd == 31)
  {
    digitalWrite(3, 1);
  }
  if(cmd == 30)
  {
    digitalWrite(3, 0);
  }
  if(cmd == 32)
  {
    digitalWrite(3, !digitalRead(3));
  }
  if(cmd == 33)
  {
```

```
    for(int i=0; i<256; i++)
    {
      analogWrite(3, i);
      delay(10);
    }
  }
  if(cmd == 34)
  {
    digitalWrite(2, 0);
    digitalWrite(3, 1);
    digitalWrite(4, 0);
    digitalWrite(5, 0);
    digitalWrite(6, 0);
    digitalWrite(7, 0);
    digitalWrite(8, 0);
    digitalWrite(9, 0);
    digitalWrite(10, 0);
    digitalWrite(11, 0);
    digitalWrite(12, 0);
  }
  if(cmd == 35)
  {
    analogWrite(3, 128);
  }
  if(cmd == 41)
  {
    digitalWrite(4, 1);
  }
  if(cmd == 40)
  {
    digitalWrite(4, 0);
  }
  if(cmd == 42)
  {
    digitalWrite(4, !digitalRead(4));
  }
  if(cmd == 44)
  {
    digitalWrite(2, 0);
    digitalWrite(3, 0);
    digitalWrite(4, 1);
    digitalWrite(5, 0);
    digitalWrite(6, 0);
    digitalWrite(7, 0);
    digitalWrite(8, 0);
    digitalWrite(9, 0);
    digitalWrite(10, 0);
    digitalWrite(11, 0);
    digitalWrite(12, 0);
  }
  if(cmd == 51)
  {
    digitalWrite(5, 1);
  }
  if(cmd == 50)
  {
    digitalWrite(5, 0);
  }
  if(cmd == 52)
  {
    digitalWrite(5, !digitalRead(5));
  }
  if(cmd == 54)
  {
    digitalWrite(2, 0);
    digitalWrite(3, 0);
    digitalWrite(4, 0);
    digitalWrite(5, 1);
    digitalWrite(6, 0);
    digitalWrite(7, 0);
    digitalWrite(8, 0);
    digitalWrite(9, 0);
    digitalWrite(10, 0);
    digitalWrite(11, 0);
    digitalWrite(12, 0);
  }
  if(cmd == 61)
  {
    digitalWrite(6, 1);
  }
  if(cmd == 60)
  {
    digitalWrite(6, 0);
  }
  if(cmd == 62)
  {
    digitalWrite(6, !digitalRead(6));
  }
  if(cmd == 64)
  {
    digitalWrite(2, 0);
    digitalWrite(3, 0);
    digitalWrite(4, 0);
    digitalWrite(5, 0);
    digitalWrite(6, 1);
    digitalWrite(7, 0);
    digitalWrite(8, 0);
    digitalWrite(9, 0);
    digitalWrite(10, 0);
    digitalWrite(11, 0);
    digitalWrite(12, 0);
  }
  if(cmd == 71)
  {
    digitalWrite(7, 1);
  }
  if(cmd == 70)
  {
    digitalWrite(7, 0);
  }
  if(cmd == 72)
  {
    digitalWrite(7, !digitalRead(7));
  }
  if(cmd == 74)
  {
    digitalWrite(2, 0);
    digitalWrite(3, 0);
    digitalWrite(4, 0);
    digitalWrite(5, 0);
    digitalWrite(6, 0);
    digitalWrite(7, 1);
    digitalWrite(8, 0);
    digitalWrite(9, 0);
    digitalWrite(10, 0);
    digitalWrite(11, 0);
    digitalWrite(12, 0);
  }
  if(cmd == 81)
  {
    digitalWrite(8, 1);
  }
  if(cmd == 80)
  {
    digitalWrite(8, 0);
  }
  if(cmd == 82)
  {
    digitalWrite(8, !digitalRead(8));
  }
  if(cmd == 84)
  {
    digitalWrite(2, 0);
    digitalWrite(3, 0);
    digitalWrite(4, 0);
    digitalWrite(5, 0);
    digitalWrite(6, 0);
    digitalWrite(7, 0);
    digitalWrite(8, 1);
    digitalWrite(9, 0);
    digitalWrite(10, 0);
    digitalWrite(11, 0);
    digitalWrite(12, 0);
  }
  if(cmd == 91)
  {
    digitalWrite(9, 1);
  }
  if(cmd == 90)
  {
    digitalWrite(9, 0);
  }
```

```
if(cmd == 92)
{
  digitalWrite(9, !digitalRead(9));
}
if(cmd == 93)
{
  for(int i=0; i<256; i++)
  {
    analogWrite(9, i);
    delay(10);
  }
}
if(cmd == 94)
{
  digitalWrite(2, 0);
  digitalWrite(3, 0);
  digitalWrite(4, 0);
  digitalWrite(5, 0);
  digitalWrite(6, 0);
  digitalWrite(7, 0);
  digitalWrite(8, 0);
  digitalWrite(9, 1);
  digitalWrite(10, 0);
  digitalWrite(11, 0);
  digitalWrite(12, 0);
}
if(cmd == 101)
{
  digitalWrite(10, 1);
}
if(cmd == 100)
{
  digitalWrite(10, 0);
}
if(cmd == 102)
{
  digitalWrite(10, !digitalRead(10));
}
if(cmd == 103)
{
  for(int i=0; i<256; i++)
  {
    analogWrite(10, i);
    delay(10);
  }
}
if(cmd == 104)
{
  digitalWrite(2, 0);
  digitalWrite(3, 0);
  digitalWrite(4, 0);
  digitalWrite(5, 0);
  digitalWrite(6, 0);
  digitalWrite(7, 0);
  digitalWrite(8, 0);
  digitalWrite(9, 0);
  digitalWrite(10, 1);
  digitalWrite(11, 0);
  digitalWrite(12, 0);
}
if(cmd == 111)
{
  digitalWrite(11, 1);
}
if(cmd == 110)
{
  digitalWrite(11, 0);
}
if(cmd == 112)
{
  digitalWrite(11, !digitalRead(11));
}
if(cmd == 113)
{
  for(int i=0; i<256; i++)
  {
    analogWrite(11, i);
    delay(10);
  }
}
if(cmd == 114)
{
  digitalWrite(2, 0);
  digitalWrite(3, 0);
  digitalWrite(4, 0);
  digitalWrite(5, 0);
  digitalWrite(6, 0);
  digitalWrite(7, 0);
  digitalWrite(8, 0);
  digitalWrite(9, 0);
  digitalWrite(10, 0);
  digitalWrite(11, 1);
  digitalWrite(12, 0);
}
if(cmd == 121)
{
  digitalWrite(12, 1);
}
if(cmd == 120)
{
  digitalWrite(12, 0);
}
if(cmd == 122)
{
  digitalWrite(12, !digitalRead(12));
}
if(cmd == 124)
{
  digitalWrite(2, 0);
  digitalWrite(3, 0);
  digitalWrite(4, 0);
  digitalWrite(5, 0);
  digitalWrite(6, 0);
  digitalWrite(7, 0);
  digitalWrite(8, 0);
  digitalWrite(9, 0);
  digitalWrite(10, 0);
  digitalWrite(11, 0);
  digitalWrite(12, 1);
}
if(cmd == 990)
{
  for(int i=2; i<13; i++)
  {
    digitalWrite(i, 0);
  }
}
if(cmd == 999)
{
  for(int i=2; i<13; i++)
  {
    digitalWrite(i, 1);
  }
}
}
```

Das Redstone-Signal steuert LEDs und Tonsignale

24. TAG

Heute im Adventskalender

- 2 Verbindungskabel

Alles blinkt auf dem Minecraft-Klotz

Zu Weihnachten ist der Minecraft-Klotz komplett mit LEDs bestückt. Mit Hilfe der in den letzten Tagen entwickelten Firmware lassen sich aus Minecraft heraus verschiedene Lichteffekte steuern.

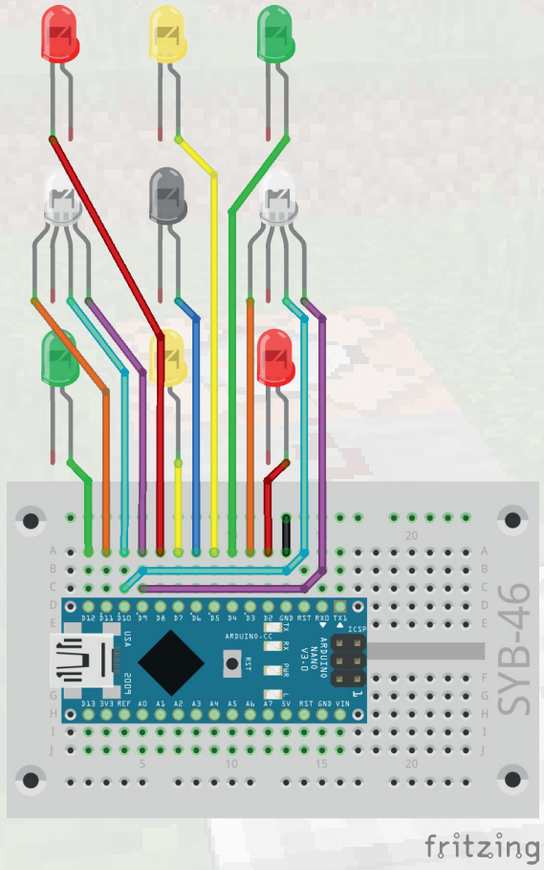
Bauteile

- 1 Nano-Board
- 1 Steckbrett
- 2 LEDs rot mit Vorwiderstand
- 2 LEDs gelb mit Vorwiderstand
- 2 LEDs grün mit Vorwiderstand
- 2 RGB-LEDs mit Vorwiderstand
- 1 Blink-LED mit Vorwiderstand
- 22 Verbindungskabel
- 1 Drahtbrücke

Der Übersichtlichkeit halber wurden die Kathodenleitungen aller LEDs in der Abbildung weggelassen. Die Kathoden aller LEDs sind mit der Masseschiene des Steckbretts verbunden.

Aufbau in Minecraft

Die im Download mitgelieferte Minecraft-Welt enthält vier Redstone-Schaltungen, die unterschiedliche Lichteffekte auf dem Minecraft-Klotz schalten. Die Schaltungen sind auf einer zentralen Plattform mit vier Knöpfen für die Spielfigur zugänglich.



Sechs LEDs, eine Blink-LED und zwei RGB-LEDs auf dem Minecraft-Klotz

Der Aufbau in Minecraft mit den Redstone-Schaltungen

Der Knopf mit dem einzelnen Befehlsblock schaltet über den Befehl `/say !062!` die Blink-LED am Pin 6 in der Mitte des Minecraft-Klotzes bei jedem Klick abwechselnd ein und wieder aus.

Dieser Knopf schaltet die Blink-LED ein und aus.

Die Reihe aus neun Befehlsblöcken lässt ein Lauflicht rund um den Minecraft-Klotz laufen. Dafür werden die Firmware-Kommandos mit der Endziffer 4 verwendet, die einzelne LEDs einschalten, wobei alle anderen ausgeschaltet werden. Die Befehlsblöcke verwenden der Reihe nach die Befehle `/say !084!`, `/say !054!`, `/say !044!`, `/say !034!`, `/say !024!`, `/say !074!`, `/say !124!`, `/say !114!`. Zum Schluss werden mit `/say !990!` alle LEDs ausgeschaltet.

Dieser Knopf startet ein Lauflicht auf dem Minecraft-Klotz

Die doppelte Reihe aus zweimal acht Befehlsblöcken schaltet der Reihe nach alle LEDs ein, lässt sie eine kurze Zeit leuchten und schaltet sie dann nacheinander wieder aus. Die Befehlsblöcke der ersten Reihe verwenden die Befehle `/say !121!`, `/say !071!`, `/say !021!`, `/say !031!`, `/say !041!`, `/say !051!`, `/say !081!`, `/say !111!`. Am Ende der Reihe läuft das Redstone-Signal um die Ecke und durch drei Verzögerer. Zu diesem Zeitpunkt leuchten alle LEDs. Die Befehlsblöcke der zweiten

Reihe schalten die LEDs nacheinander wieder aus und verwenden dazu die Befehle `/say !120!`, `/say !070!`, `/say !020!`, `/say !030!`, `/say !040!`, `/say !050!`, `/say !080!`, `/say !110!`.

Dieser Knopf schaltet acht LEDs ein und nach kurzer Zeit wieder aus

Der vierte Knopf startet ein Redstone-Signal auf einer weiteren Reihe von Befehlsblöcken. Diese lassen die beiden RGB-LEDs in unterschiedlichen Farbkombinationen aufleuchten. Die Farbkomponenten Grün und Blau beider RGB-LEDs sind an die gleichen Pins angeschlossen. Durch unterschiedliche Schaltung der roten Farbkomponenten entstehen unterschiedliche Mischfarben. Die Befehlsblöcke verwenden die Befehle `/say !111!`, `/say !101!`, `/say !091!`, `/say !031!`, `/say !110!`, `/say !090!`, `/say !100!`, `/say !030!`.

Dieser Knopf steuert die RGB-LEDs

Alle Funktionen der Minecraft-Firmware

Das Programm `mc24.ino` ist die letzte Version der Minecraft-Arduino-Firmware. Sie enthält zahlreiche Steuerungskommandos zum Ein- und Ausschalten aller Pins von 2 bis 12. Alle Kommandos müssen aus Minecraft heraus durch einen `/say !...!`-Befehl mit dreistelliger Zahlenangabe übertragen werden. Jedes Kommando besteht aus der Pinnummer und danach einer Endziffer, die die eigentliche Funktion beschreibt.

Kommandos mit Endziffer 0 (`20`...`120`) schalten den jeweiligen Pin aus, ohne die anderen Pins zu beeinflussen.

Kommandos mit Endziffer 1 (`21`...`121`) schalten den jeweiligen Pin ein, ohne die anderen Pins zu beeinflussen.

Kommandos mit Endziffer 2 (`22`...`122`) schalten den jeweiligen Pin um. Ein eingeschalteter Pin wird ausgeschaltet, ein ausgeschalteter Pin eingeschaltet. Auch dies erfolgt, ohne die anderen Pins zu beeinflussen.

Kommandos mit Endziffer 3 (nur PWM-Pins `33`, `93`, `103`, `113`) blenden am jeweiligen Pin angeschlossene LEDs über ein PWM-Signal langsam ein, ohne die anderen Pins zu beeinflussen.

Kommandos mit Endziffer 4 (`24`...`124`) schalten den jeweiligen Pin als einzigen ein. Alle anderen Pins werden ausgeschaltet.

Kommando `35` liefert auf dem PWM-Pin ein Signal für einen Piezo-Summer.

Kommando `990` schaltet alle Pins aus.

Kommando `999` schaltet alle Pins ein.

Diese Firmware bietet, einmal auf das Nano-Board übertragen, jede Menge Möglichkeiten für weitere Experimente in Minecraft.

Frohe Weihnachten!

Für Ihre Notizen

FRANZIS

ADVENTSKALENDER PROGRAMMIEREN MIT MINECRAFT™ UND WINDOWS-JAVA

DER ADVENTSKALENDER FÜR MINECRAFT™-FANS UND SOLCHE, DIE ES WERDEN WOLLEN

In 24 Tagen vom Spieler zum Programmierer: So sinnvoll war Minecraft™ spielen noch nie.

Du bist Minecraft™-Fan oder interessierst dich für das Spiel? Dann ist dieser Adventskalender genau das Richtige für dich: Verkürze dir die Wartezeit auf Weihnachten mit 24 tollen Minecraft™-Projekten.

Mit Minecraft™ verbringst du nicht nur viele kurzweilige Stunden mit spielen, sondern kannst auch programmieren lernen. Steuere mit Redstone eigene elektronische Spielereien und LED-Lichteffekte auf einem selbstgebauten Minecraft™-Klotz.

Im Adventskalender wird die Java-Edition von Minecraft™ unter Windows eingesetzt. Zur Ansteuerung der Elektronik ist ein Arduino®-kompatibles Nano-Board enthalten. Über einen angeschlossenen Piezo kannst du aus Minecraft™ heraus Töne wiedergeben.

Alle Schaltungen werden ohne zu löten auf einem Steckbrett aufgebaut. Ganz nebenbei vermittelt der Adventskalender auf spielerische Art Hardware- und Programmierwissen.

Mit Hilfe des farbig illustrierten Handbuchs lassen sich alle 24 Experimente auch ganz ohne Programmierkenntnisse aufbauen und programmieren.

Erfahrungen im Spielen von Minecraft™ sollten vorhanden sein. Alle zusätzlich zur Java-Edition von Minecraft™ notwendige Software wird kostenlos zum Download angeboten, darunter auch eine eigens für den Adventskalender erstellte Minecraft™-Welt.

Die Experimente funktionieren mit der Java-Edition von Minecraft™ unter Windows 7, 8.1 und 10. Die Minecraft™-Edition aus dem Windows Store sowie die Java-Editionen für MacOS und Linux werden nicht unterstützt. Für die Durchführung der Experimente ist ein Internetzugang erforderlich.

Für Kinder unter 14 Jahren nicht geeignet!

FRANZIS

Bestellnummer: 1662789
Hergestellt in P. R. China

WEEE-REG.-NR.: DE 21445697

GTIN 4019631150257

4 019631 150257 96060002