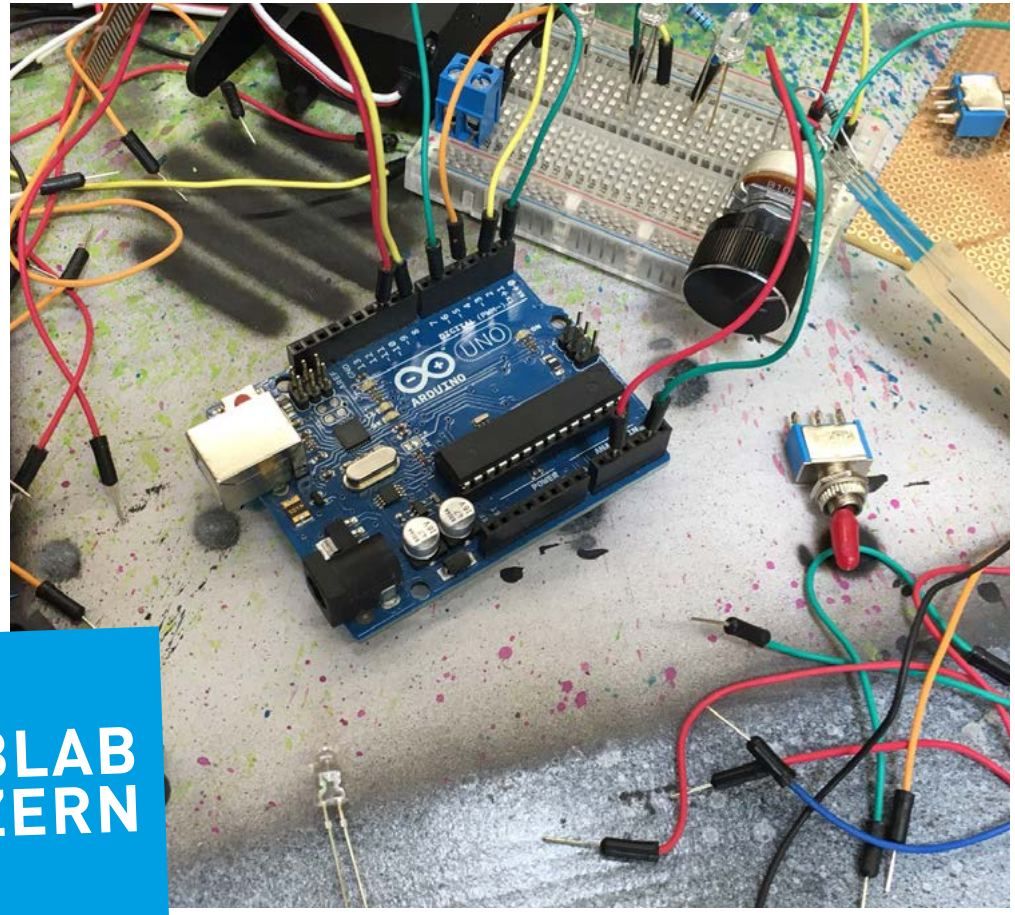


FABLAB
LUZERN

ARDUINO EINSTEIGER-KURS

In diesem Dokument sind alle Informationen rund um den Arduino Einsteiger-Kurs vom FABLAB LUZERN gesammelt, unter Anderem eine Auflistung der wichtigsten Web-Links und Litheraturhinweise zu diesem Thema. Zudem findest du hier eine Übersicht der im Workshop-Kit enthaltenen elektronischen Bauteile, sowie eine Reihe von Übungen und Code-Beispiele zum selber Nachbauen, Tüfteln und Ausprobieren...

Wir wünschen dir viel Spass bei deinen Projekten und geben dir folgende Worte mit auf deinen Weg: no guts, no glory!

HIER VERSCHAFFST DU DIR EINEN ÜBERBLICK

1 GRUNDLAGEN

Ein paar Dinge musst du wissen, bevor du mit deinem Arduino starten kannst. Das Wichtigste und vieles mehr haben wir dir zusammengetragen – im Workshop können wir dir einen Überblick über dieses complexe Thema geben. (Seite 3)

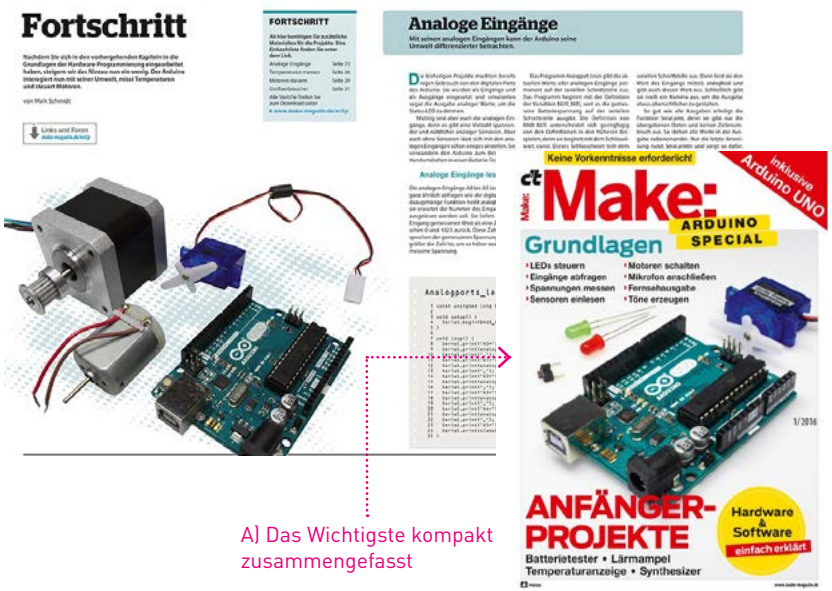
2 ARDUINO & BREADBOARD

Das Arduino-Universum besteht aus einem Software- und einem Hardware-Teil. Auf dem Breadboard werden Schaltkreise provisorisch gesteckt, um schnell einen Patch auf den Arudiono laden und testen zu können (Seite 4&5)

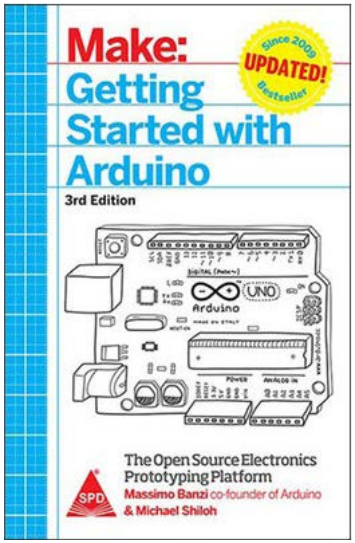
3 ÜBUNGEN UND CODE-BEISPIELE

Übung macht den Meister, sammle aber Wissen nicht auf Vorrat! Am einfachst kommst du mit einem konkreten Vorhaben vorwärts – und jedes deiner Projekte bringt dich dann ein Stückchen weiter. (ab Seite 7)

1

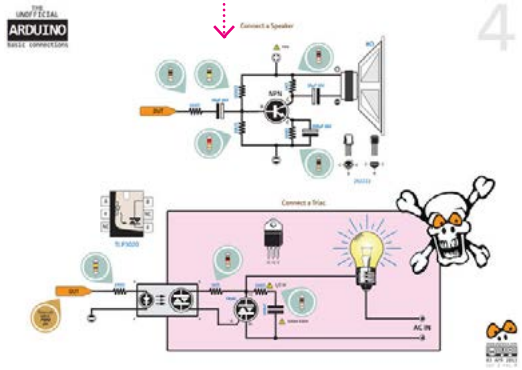


A) Das Wichtigste kompakt zusammengefasst



B) Das Standardwerk!

C) Eine Sammlung der meist gebrauchten Schaltungen



GRUNDLAGEN

WAS IST EIN MICROCONTROLLER?

Arduino ist im Kern nicht nur ein Microcontroller, Arduion ist ein komplettes System, welches aus einem Prototyping-Board und der dazugehörenden Software besteht, mit der man den Chip programmieren kann. Der Vorteil von Arduino und allen ähnlichen Boards ist, dass sie auf dem Open-Source-Gedanken basieren und auch für Nicht-IT-Spezialisten, d.h. Künstler, Bastler, Maker und sogar Kinder zugänglich sind. Kurz gesagt, kann man mit einem Arduino (fast) alles machen: von der automatischen Fütterungsmaschine für die Katze, über die interaktive Audio/Video-Installation, bis zum 3D-Drucker und dem Quadcopter... alles elektronische – alles was mit Strom betrieben wird – kann mit einem Microcontroller gesteuert bzw. gelesen werden. Wir verzichten hier auf eine ausführlichere Beschreibung, weil das Einige vor uns schon viel besser hinbekommen haben – wenn Du es also etwas genauer wissen willst, lies einfach die ersten paar Seite im «Arduino Special» vom MAKE:Magazin (siehe unten: wo erfahre ich mehr?).

WAS BRAUCHE ICH DAZU?

Ein Computer oder LapTop, Internet-Zugang, ein Microcontroller-Board (Arduino, Genuino, Seeeduino, Diduino, Freaduino, Olimexino, Adafruit Metro, ... und in der Ausführung je nach Bedürfniss: Pro, Mega oder Mini), eine Handvoll elektronische Bauteile und Sensoren sowie sehr, sehr viel Geduld...!!! Alle Informationen sind «open source» d.h. kostenlos Verfügbar.

WO ERFAHRE ICH MEHR?

A) Eine unverzichtbare Lektüre ist die Ausgabe «Arduino Special» vom MAKE:Magazin. Hier kannst Du alles im Detail nachlesen, wie z.B. ein Arduino funktioniert sowie viele Übungen und Beispiele für Anfänger und Fortgeschrittene: https://shop.heise.de/katalog/make-arduino-special-inkl-arduino-uno

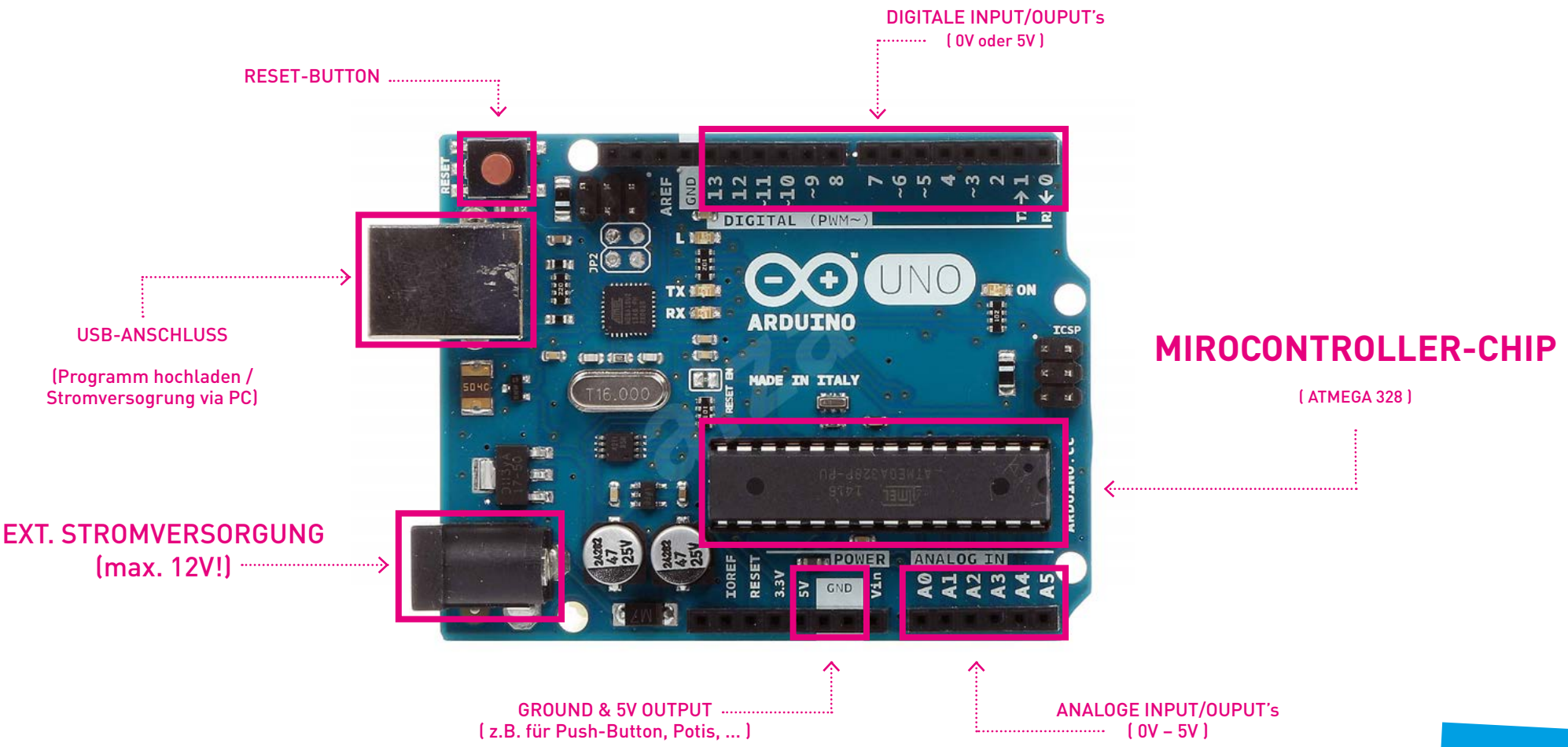
B) Im Stardardwerk «Getting Started wirh Arduino» (3. Edition) beschreibt Massimo Banzi, ein Miterfinder der Arduino-Plattform, detailliert Zusammenhänge und Hintergründe: http://www.makershed.com/products/make-getting-started-with-arduino-3rd-edition

C) Wenn's ums Anschliessen von elektronischen Komponenten am Arduino geht, wirds meist etwas heikel! Nicht aber mit dem sehr hilfreichen Nachschlagewerk «ABC – Arduino Basic Connection» von Pighixx: http://www.pighixxx.com/test/abc/

FABLAB LUZERN

2 ARDUINO

SO IST EIN ARDUINO-BOARD AUFGEBAUT:



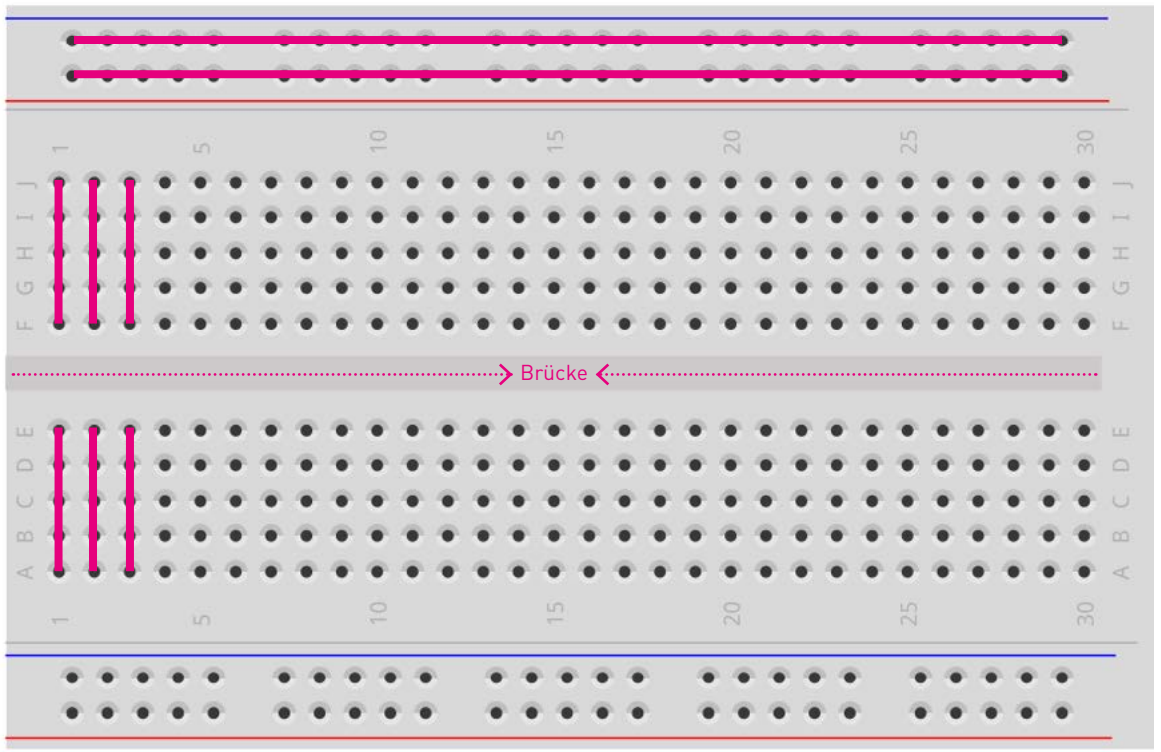
2

BREADBOARD

UND SO FUNKTIONIERT EIN BREADBOARD:

Auf dem **Breadboard** kannst du elektronische Schaltungen stecken, ohne dass die Komponenten fest miteinander verlötet werden müssen – ideal zum Prototypen bauen und um Schaltungen zu prüfen!

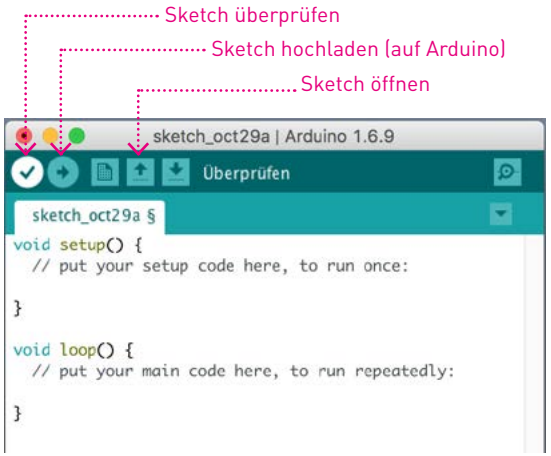
Die **Innenbahnen** sind **vertikal** von **A–E** und **F–J** miteinander verbunden. Über die **Brücke** werden meistens Bauteile wie z.B. **Buttons** und **IC's** gesteckt...



Die **Aussenbahnen** sind **horizontal** miteinander verbunden und werden meist für die Stromversorgung der Schaltung verwendet...

3

Die Software: IDE



Ein Sketch-Beispiel: «Blink»

ÜBUNGEN UND BEISPIELE

Text

SO UND JETZT STARTEN WIR RICHTIG...!

- → Installiere die Arduino-Software (IDE): https://www.arduino.cc/en/Main/Software
- → Verbinde den Arduino mit deinem PC via USB-Kabel, öffne den Sketch BLINK und lade ihn hoch – die Beschreibung dazu findest Du auf der Folgeseite (Seite 6)!
- → Die mitgelieferten Beispiele sind hier genau beschrieben (Schaltkreis, Code): https://www.arduino.cc/en/Tutorial/BuiltInExamples
- → Hier findest Du weitere coole Projekte: https://create.arduino.cc/projecthub
- → Bei Fragen zu XY: https://www.arduino.cc/en/Reference
- → Profitiere von anderen Usern: http://playground.arduino.cc/

DER AUFBAU EINES SKETCHS

Ein Sketch besteht im Wesentliche aus zwei Programmteilen: «void setup» und «void loop». In den Kommentaren (/*...*/ oder nach //...) sind Programmteile und was sie bewirken beschrieben. Dann gibt es noch Funktionen (orange) und Attribute/Werte (türkis).

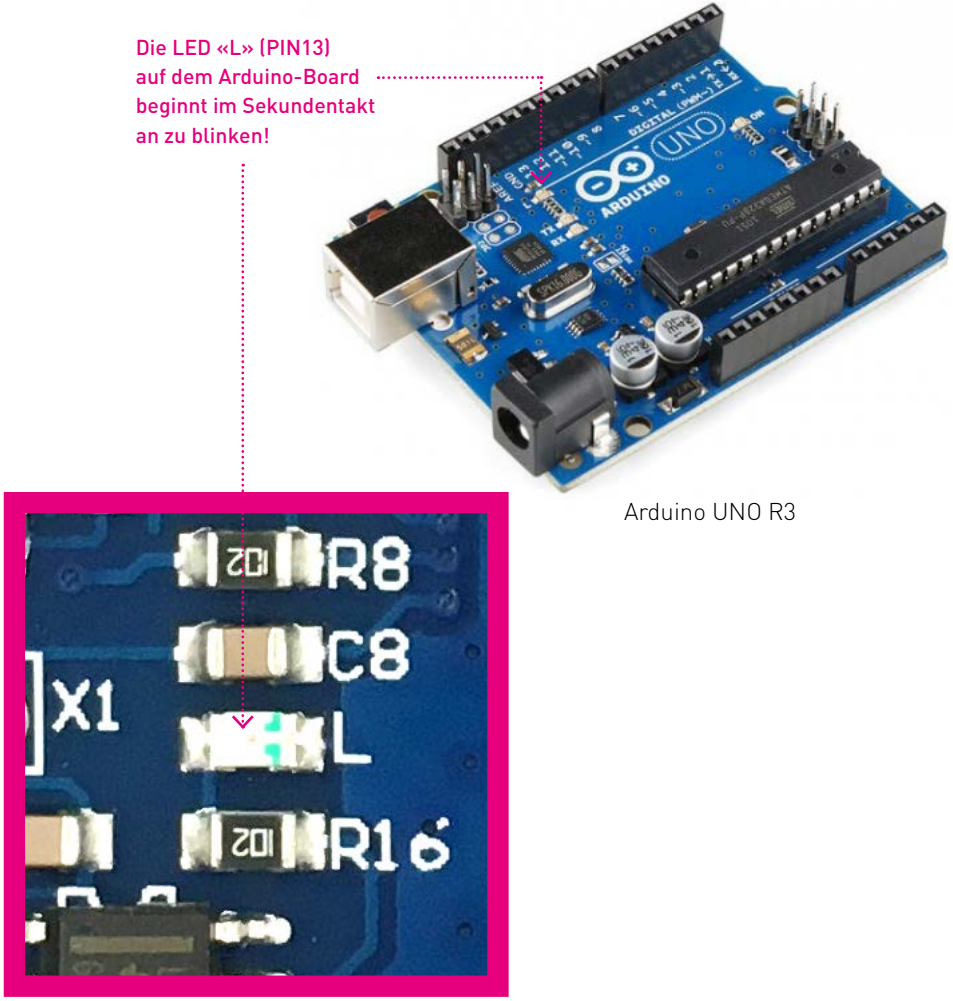
TIP

- → Einer der häufigsten Fehler im Code sind: fehlende Strichpunkte (;) oder fehlende Klammern ({ })
- → Sketches können nur vom Computer auf den Arduino geladen werden – rückwärts funktioniert das nicht! Sketches solltest du deshalb immer an einem «sicheren» Ort speichern.

3.1

BLINK

Die allererste und wichtigste Aufgabe überhaupt! Wenn du sie meisterst, hast du schon einen riesen Schritt in diese neue Welt getan – Housten dankt! Gleichzeitig ist es auch so etwas wie ein Soft- und Hardware-Test, der dir zeigt, ob dein Computer mit dem Arduino richtig kommuniziert. Den Code zu BLINK findest du in der Arduino-Software (IDE) unter: Datei > Beispiele > 01.Basic > Blink



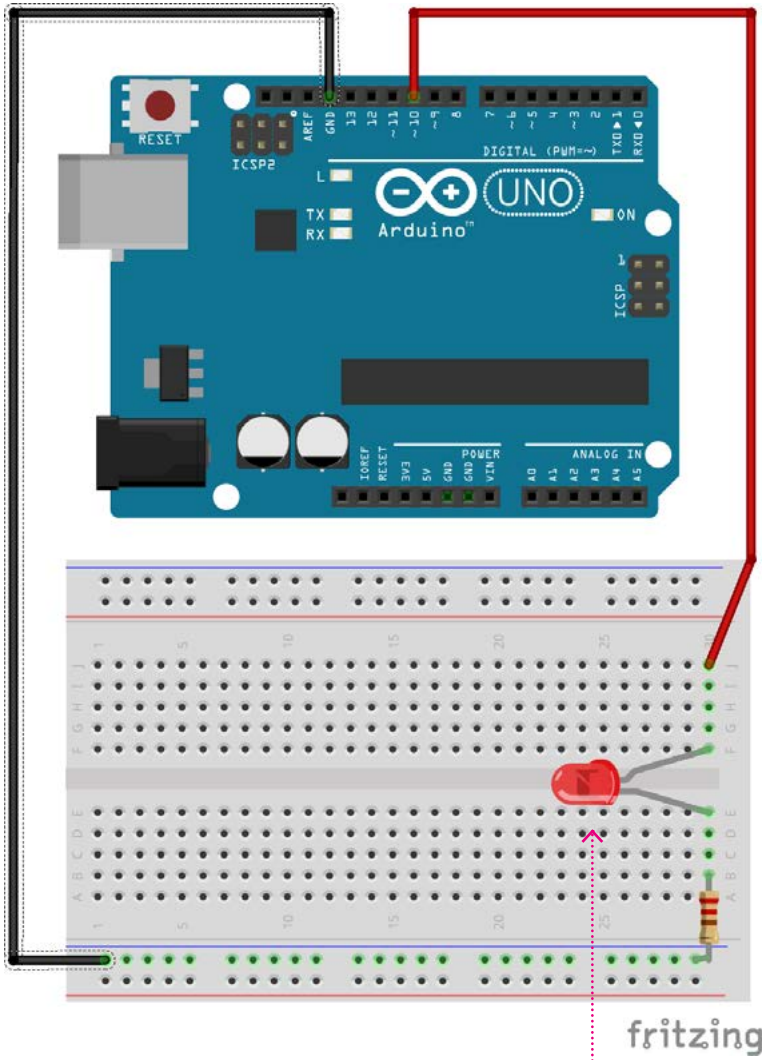
Arduino UNO R3

TIP

→ Nur PIN 13 kannst Du direkt mit einer LED verbinden, weil dort schon einen Widerstand fest auf dem Arduino verbaut ist! Wenn Du nun LED's an andere PIN's anschliesst, musst du unbedingt einen Widerstand dazwischen hängen. Sonst kann es sein, dass viel Strom fliesst und dein Baord oder die LED gebraten werden – das wär doch jammerschade!

3.2 MORSEN & CANDLE LIGHT

Jetzt geht's richtig los mit Programmieren bzw. am Code rumbasteln: nimm den Sketch «Blink» und versuche ihn so anzupassen, dass du eine Botschaft via Morse-Code (blinken kurz & lang) übermitteln kannst... gleichzeitig steckst du auf dem Breadboard das Schema links – vergiss den Widerstand nicht! (egal ob vor oder nach der LED)



So wird eine LED angeschlossen – immer mit einem Widerstand (ca. 10 kOhm)!

ERKLÄRUNGEN ZUM CODE

→ void setup() // die darin enthaltene Funktionen werden nur einmalig ausgeführt
→ pinMode(13, OUTPUT); // inizialisiert den digitalen PIN13 als Ausgang
→ void loop() // die darin enthaltenen Funktionen werden unendlich repetiert
→ digitalWrite(13, HIGH); // diese Funktion schaltet die LED ein
→ delay(1000); // diese Funktion macht eine Pause von 1 Sekunde (1'000ms)
→ digitalWrite(13, LOW); // diese Funktion schaltet die LED wieder aus
→ delay(1000); // diese Funktion macht wieder eine Pause von 1 Sekunde (1'000ms)
→ { } // diese Klammern definieren Anfang und Ende von SETUP bzw. LOOP
→ ; // am Ende einer Funktion steht immer ein Strichpunkt

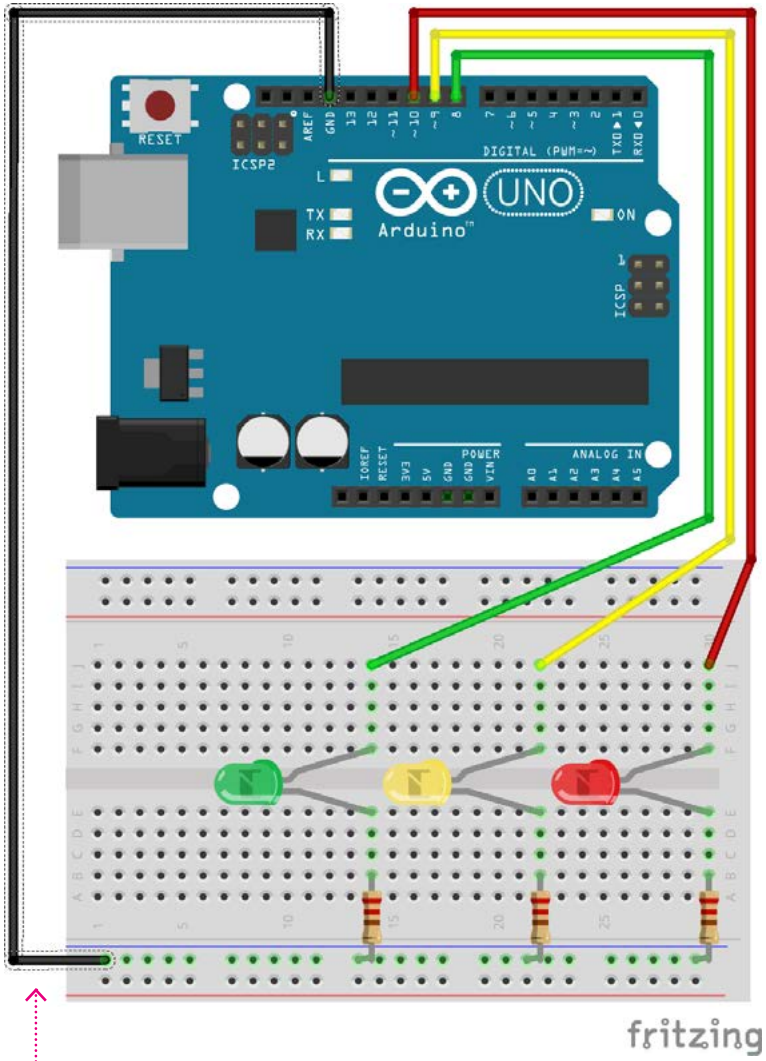
PROBIER DAS HIER NOCH AUS

→ Im Beispiel «CandleLight» wird die Blink-Zeit durch einen Zufallsgenerator bestimmt – experimentiere auch hier mit verschiednenen Einstellungen.
→ Oder versuche mit dem Schema auf der folgenden Seite eine einfache Ampelschaltung zu realisieren – mit delay() können auch zwei LED's gleichzeitig leuchten...
→ Das Beispiel «Fade» kannst du auch öffnen und hochladen. Hier wird mittels einer PWM (Puls-Weiten-Modulation) die LED «gedimmt»...

TIP

→ Übrigens: der Steckplan links wurde mit der Open-Source-Sorftware von fritzing.org erstellt – eine geniale und sehr einfach zu bedienende Software, die sich auch zum Gestalten von PCB's geeignet!

3.3



Wir verwenden «schwarz» für Minus bzw. GND (Ground)

BLINKEN FÜR FORTGESCHRITTENE

Die Funktion «delay» stopt/pausiert (wie oben im Beispiel «Blink») das gesammte Programm – d.h. in dieser Zeitspanne sind alle anderen Funktionen ausser Kraft gesetzt! Falls das nicht so sein sollte, brauchst Du hier eine andere Lösung...

ERKLÄRUNGEN ZUM CDOE

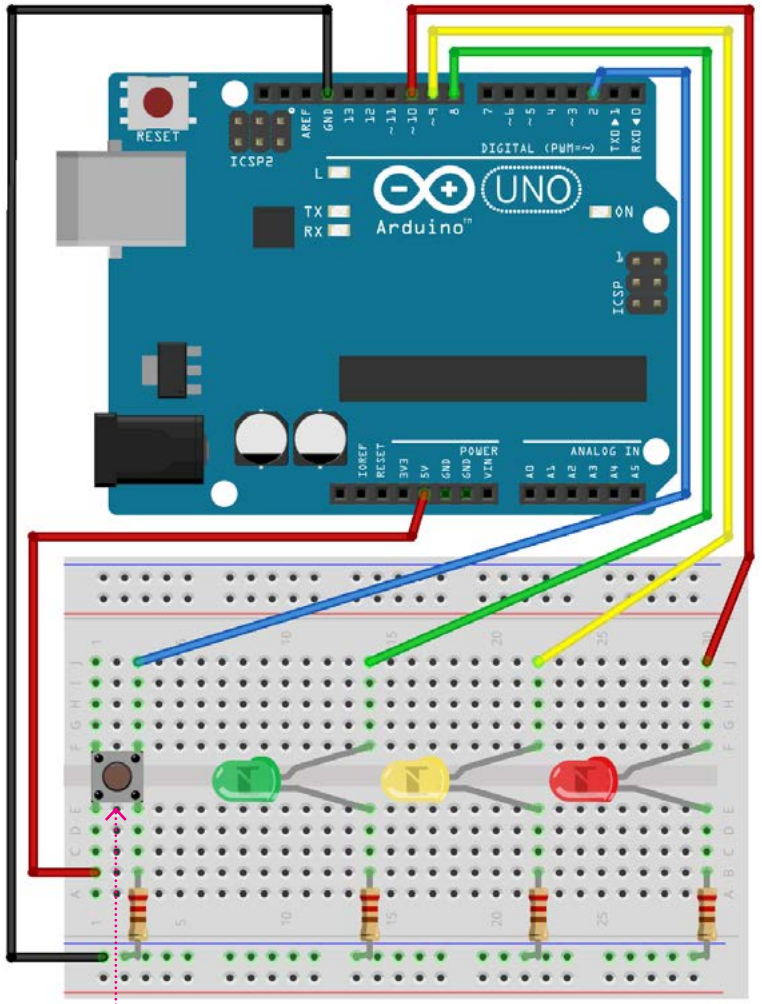
- → int ledState = LOW; // die LED startet im ausgeschaltenen Modus
- → unsigned long previousMillis = 0 // speichert die Anzahl Millisekunden
- → const long interval = 500; // die Länge der Blink-Dauer kann eingestellt werden
- → unsigned long currentMillis = millis(); // diese Funktion zählt die Millisekunden
- → (currentMillis - previousMillis >= interval) // hier werden zwei Werte miteinander verglichen, wenn der Wert überstigen wird (500ms), ändert sich der Status der LED
- → Für das Umschalten wird die Struktur if/else verwendet
- → digitalWrite(ledPin, ledState); // löst den Befehl schlussendlich aus

PROBIER DAS HIER NOCH AUS

- → In den Beispielen BlinkWithoutDelay1 bis 3 weden verschiedene Varianten gezeigt, z.B. unterschiedliche Ein- und Ausschaltzeiten sowie mit mehrern LED's

FABLAB LUZERN

3.4 DIE AMPEL



So wird eine BUTTON angeschlossen – der Arduino bruacht den Widerstand, damit er ein «sauberes» Signal messen kann.

Um nicht ständig eine Abrfage laufen zu lassen ist die «Stateflow»-Technik ganz hilfreich: so lassen sich mit dem Push-Button zwischen verschiednenen «Situationen» bzw. Einstellungen einfach hin und herschalten...

ERKLÄRUNGEN ZUM CDOE

→ int mode = 1; // mit diesem Mode wird gestartet (1-4)
→ int butState = 0; // die Variable für den Knopf
→ int modeState = 0; // speichert den letzten aktiven Mode (1-4)
→ val = digitalRead(modePin); // hier wird geschaut, wann der Knopf gedrückt wird
→ dann wird mit if/else zum nächsten Mode gewechselt
→ switch (mode) // ist die eigentliche Funktion, in der Klammer steht der Name
→ case 2: // hier wird definiert, was in den einzelnen Modes geschehen soll
→ default: // das ist der Case 1
→ break; // braucht es am Ende von jedem Case (!)
→ modeState = mode; // überschreibt den vorherigen mit dem aktuellen Mode
→ ...

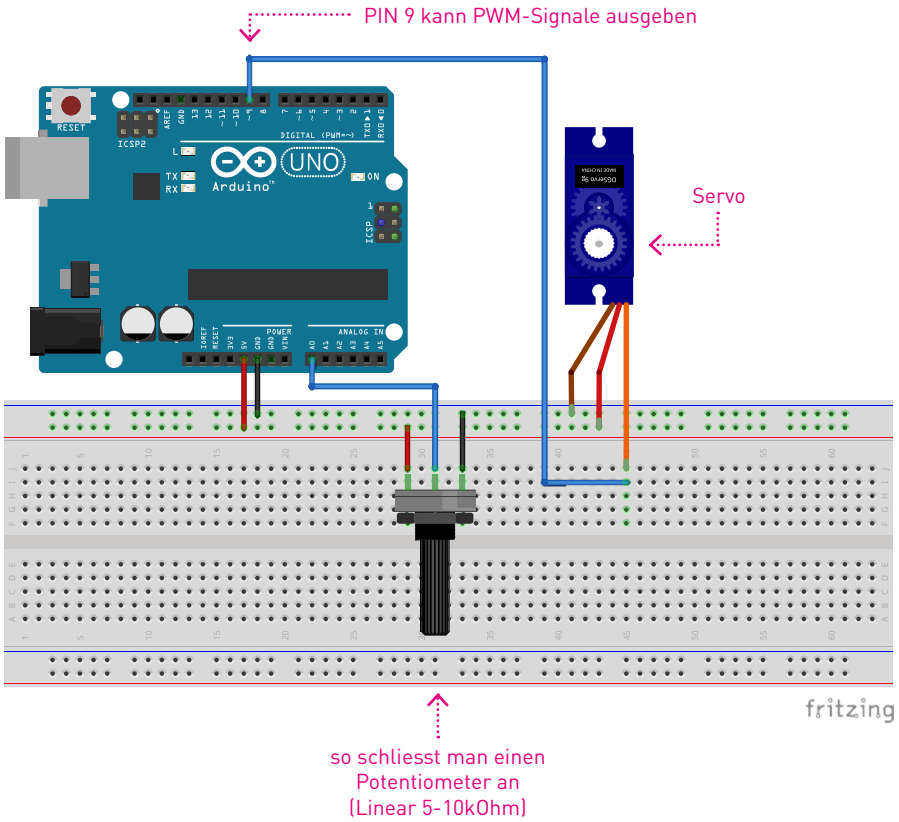
PROBIER DAS HIER NOCH AUS

→ Versuche bei einer LED das Beispiel BlinkWithoutDelay oder Fade einzubauen. Hier ganz wichtig: die Funktion delay darf nicht verwendet werden, da sonst in der Pause der Button nicht funktionieren würde!
→ Stelle dir eine Strassensituationen vor, wie z.B. eine Kreuzung oder ein Zebrastreiffen, baue die Schaltung und den Code entsprechend aus – wann kommst du mit den PINS ans Limit?

TIP

→ Diese Schaltung ist sehr flexibel, weitere Cases können ganz einfach hinzugefügt oder gelöscht werden. Was könnte man die dieser Schaltung noch so alles anstellen?

3.5



SERVO-KNOB

Dreh am Poti und der Servo dreht sich mit – so einfach, so genial!

ERKLÄRUNGEN ZUM CDOE

→ #include <Servo.h> // die Bibliothek [Servo.h] wird hier eingebunden
→ Servo myservo; // kreiert ein Objekt um den Servo zu kontrollieren
→ int potpin = 0; // für den Poti wird ein Analoger-PIN [A0] gebraucht
→ int val; // die Variable für den den Wert am Eingang des Potis
→ ---
→ myservo.attach(9); // PIN 9 kontrolliert den Servo (PWM-Signal)
→ ---
→ val = analogRead(potpin); // liest den Wert des Potis (zwischen 0 und 1023)
→ val = map(val, 0, 1023, 0, 180); // skaliert die Werte für den Servo
→ myservo.write(val); // setzt die Position des Servos
→ delay(15); // warte bis der Servo die Position eingenommen hat

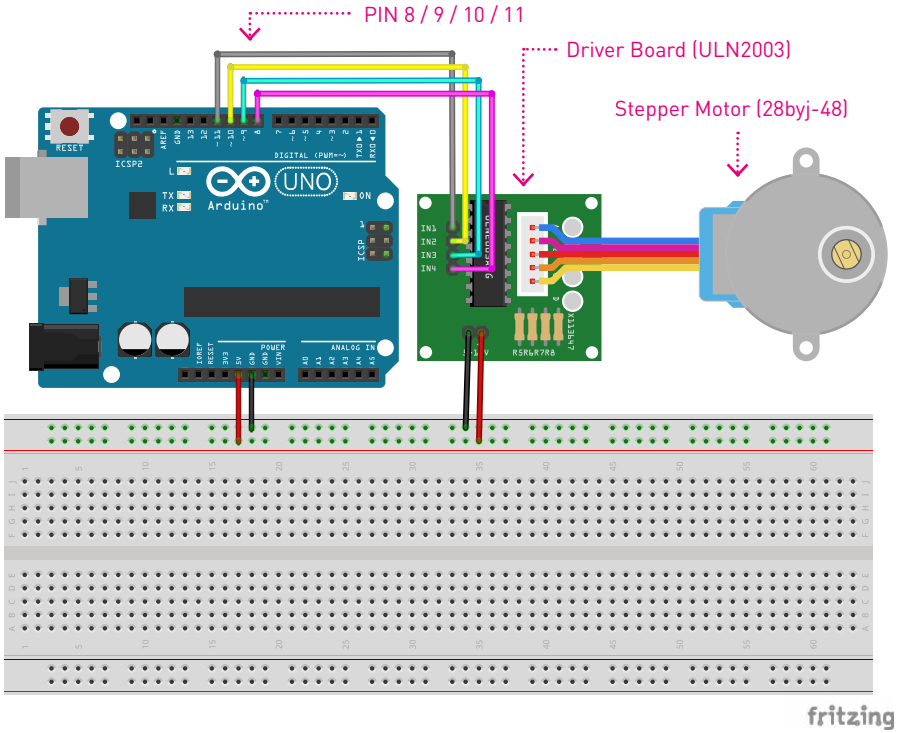
PROBIER DAS HIER NOCH AUS

→ Mit dem Sketch [AnalogInOutSerial] und dem Serial-Monitor, den du unter Werkzeuge findest, kannst du sehen, welche Werte am Eingang [A0] wirklich ankommen.

TIP

→ Je nach der Qualität der Komponenten kann der Servo etwas «flattern» – dem könnte man auch mit den glätten der Werte entgegenwirken.

3.6



STEPPER MOTOR

Wenn Bewegungen genau sein sollen, muss ein Stepper her: 68 Steps nach links, 4251 nach rechts, wieder 2 zurück... oder vielleicht exakt 1 Umdrehung pro Minute. Darum kommen diese Motoren auch in einem 3D-Drucker zum Einsatz, aber nicht nur dort...

ERKLÄRUNGEN ZUM CDOE

```
→  #include <Stepper.h>  // die Bibliothek [Stepper.h] wird hier eingebunden
→  const int stepsPerRevolution = 2048;  // Anzahl Steps für 1 Umdrehung
→  Stepper myStepper(stepsPerRevolution, 8, 10, 9, 11);  // Ansteuerung Stepper
→  ---
→  myStepper.setSpeed(15);  // max. Geschwindigkeit
→  Serial.begin(9600);  // initialisiert den Serial Port
→  ---
→  myStepper.step(stepsPerRevolution);  // Bewegung im Uhrzeigersinn
→  delay(500);  // Pause von 1/2 Sekunde
→  myStepper.step(-stepsPerRevolution);  // Bewegung im Gegen-Uhrzeigersinn
→  delay(500);  // Pause von 1/2 Sekunde
```

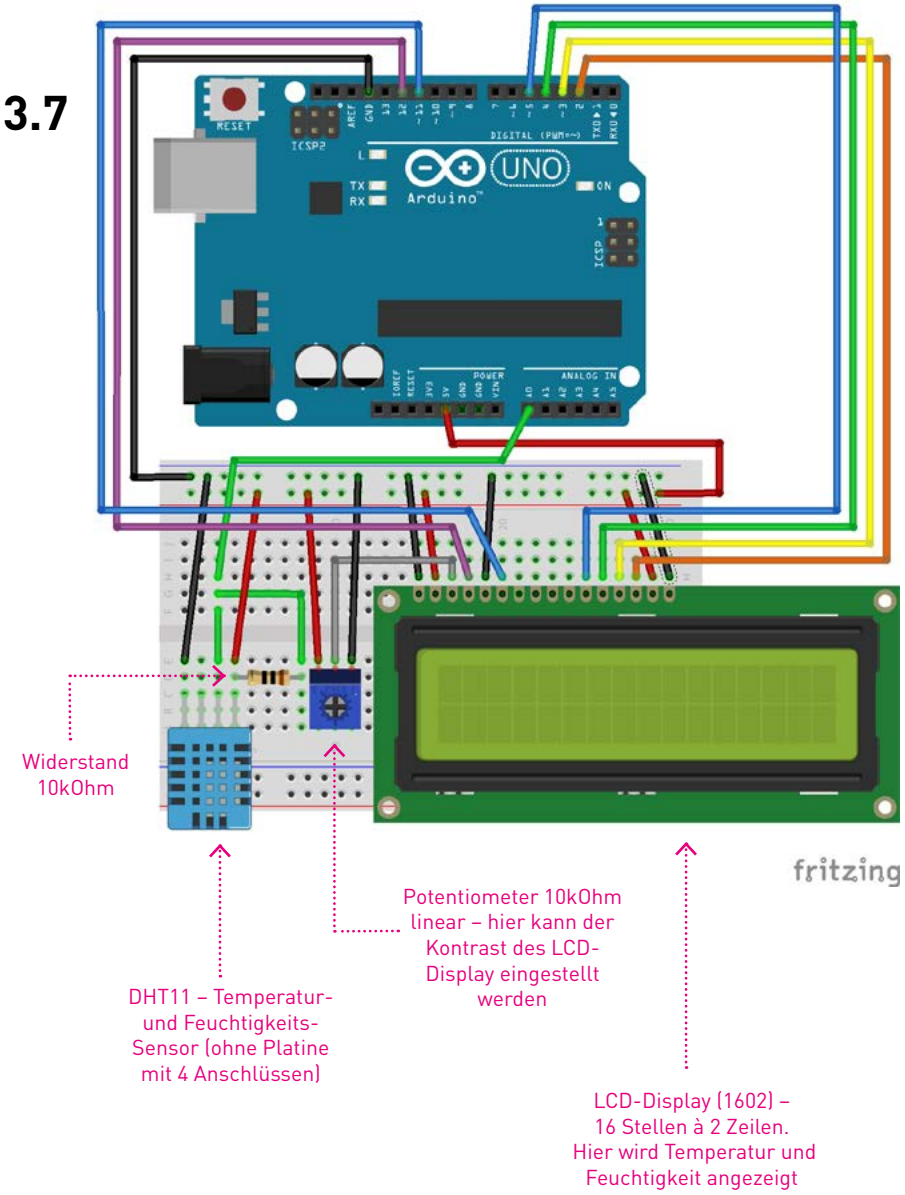
PROBIER DAS HIER NOCH AUS

→ Im Beispiel dreht der Motor jeweils eine ganze Umdrehung (= 2048 Steps) – probiere nun, dass der Motor eine 1/2 oder 1/4 Umdrehung macht.

TIP

→ Dieser Stepper Motor ist untersetzt, d.h. maximal sind 15 Umdrehung pro Minute möglich!

3.7 WETTERSTATION



Und jetzt zur Abwechslung mal ein etwas «sinnvolleres» Projekt: mit den Temperatur- und Feuchtigkeits-Sensor (DHT11) baust du dir deine eigene Wetterstation (INDOOR-Variante)!

ERKLÄRUNGEN ZUM CDOE

→ #include <LiquidCrystal.h> // die Bibliothek für das Display wird eingebunden
→ #include <DHT.h> // sowie die Bibliothek für den Sensor
→ #define DHTTYPE DHT11 // der entsprechende Senor wird ausgewählt
→ LiquidCrystal lcd(12, 11, 5, 4, 3, 2); // die Anschluss-Pins für das Display definiert
→ Serial.begin(9600); // damit der Serial-Monitor funktioniert
→ lcd.begin(16,2); // das Anzeigeformat des Displays
→ dht.begin(); // die Funktion der Messung über den Sonsor
→ float h = dht.readHumidity(); // die eigentliche Messung der Feuchtigkeit
→ float t = dht.readTemperature(); // die eigentliche Messung der Temperatur
→ Serial.println("DHT11 Sensor"); // Ausgabe über den «Serial-Montior»

PROBIER DAS HIER NOCH AUS

→ Wenn die «Serial.print»-Funktion im Sketch auskommentiert wird und können die Werte auch über den «Serial-Monitor» der IDE ausgegeben werden.

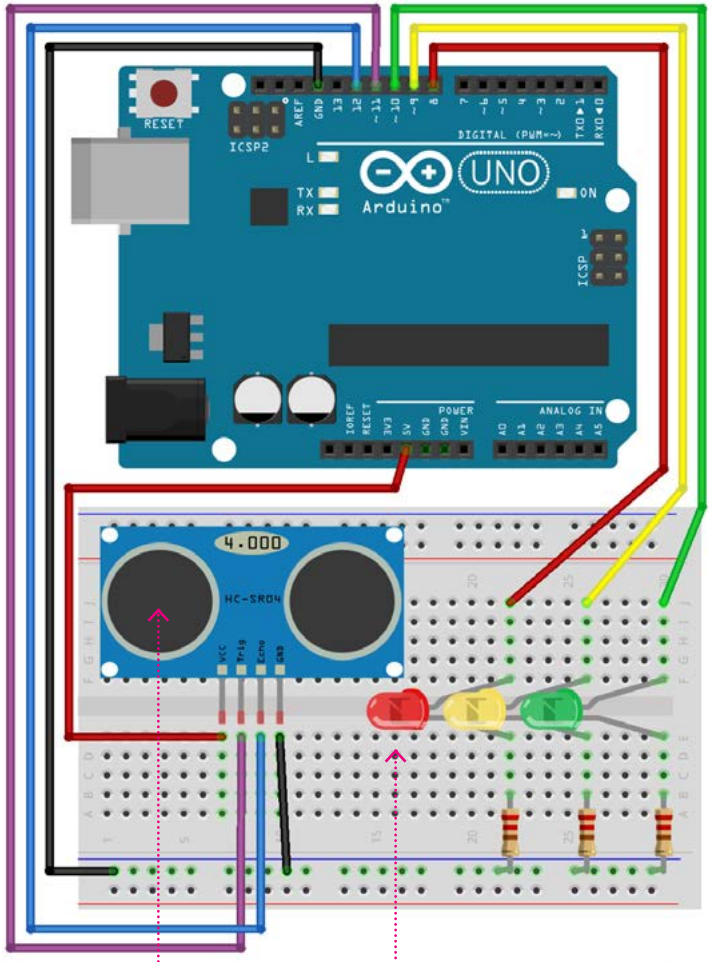
TIP

→ Den DHT11-Sensor gibt es auch aufgelötet auf einer kleinen Platine mit 3 Anschlüssen (Minus, Plus und Signal). Der kann so – ohne Widerstand (10kOhm) – direkt angeschlossen werden!

→ Damit der Code funktioniert muss – bevor die IDE gestartet wird – die Bibliothek des DHT11-Sensors sowie die «LiquidCristal»-Library in folgendem Ordner abgelegt werden: Dokumente/Arduino/Sketchbook/libraries

→ Um die Wetterstation im Freien auf zu stellen, empfehlen wir die elektronik in einen dichten Geäuse unter zu bringen und die Daten evtl. über ein WLAN oder IOT zu übermitteln.

3.8 DER EINPARKASSISTENT



Ach, wie mühsam war das Einparken ohne einen Assistenten, der uns den Abstand zur Wand oder zum nächsten Auto angibt. Doch, wie funktioniert dieser? Ultraschall-Distanz-Sensoren sind dem Orientierungsorgan der Fledermaus nachempfunden und heutzutage fast in jedem Auto standard... der Sensor sendet ein hochfrequentes Signal aus und misst die Zeit, bis es wieder bei ihm eintrifft.

ERKLÄRUNGEN ZUM CDOE

→ int trigPin = 11; // der Senor braucht ein Trigger
→ int echoPin = 12; // und wartet dann auf ein Echo
→ long duration, cm; // er misst die Zeit dazwischen und gibt die Distanz in cm aus
→ int readSensorDist() // der ganze Ablauf haben wir in eine eigene Funktion gepackt
→ else if // damit werden die LED's angesteuert
→ Ganz nahe Objekte (< 35cm) lösen die rote LED aus(!)

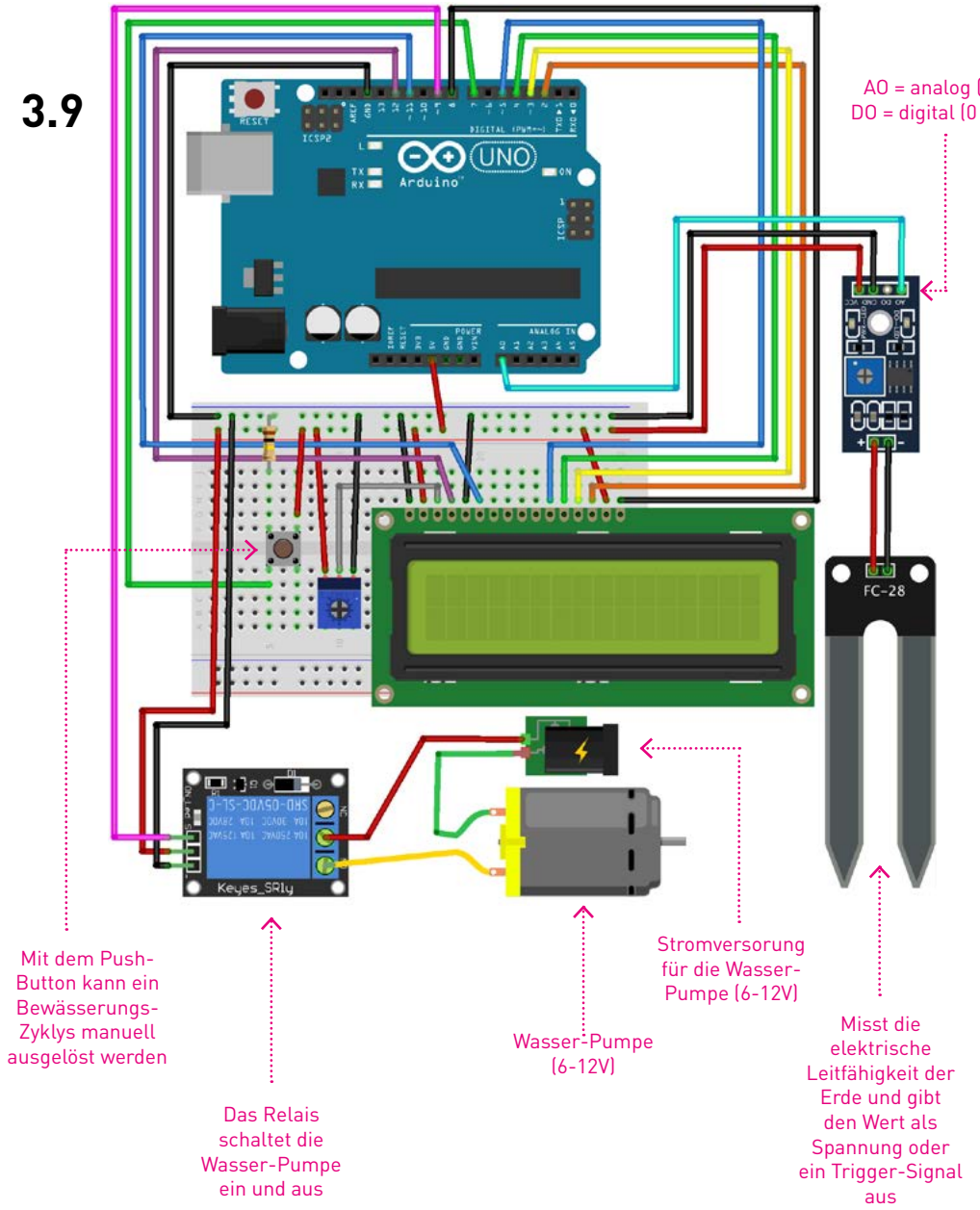
PROBIER DAS HIER NOCH AUS

→ Können die LED's durch einen Speaker und ein akustisches Signal ersetzt werden?

TIP

→ Der «HC-SR04» kostet bei Aliexpress.com weniger als 1.– $. Im Workshop-Kit ist er leider nicht enthalten. Wenn du so einen brauchst, bekommst du einen bei uns im FabLab Luzern – bis bald!

3.9 DAS BEWÄSSERUNGSSYSTEM



Längere Zeit mal weg und niemand, der auf deine Pflanzen aufpasst? Hier ist die Lösung: dieser Sketch schaut zu euren Grünlingen und lässt sie nicht vertrocknen...

ERKLÄRUNGEN ZUM CDOE

- → millis() // brauchen wir für verschiedene Timer (Overflow nach ca. 50 Tagen) – das sind Pausen zwischen: dem Sensor auslesen, Bewässerungs-Zyklen, Restart und Display ausschalten.
- → void(* resetFunc) (void) = 0; // Funktion um den Arduion zurückzusetzen
- → moistureValue1 = analogRead(moistureSensor1); // Sensor messen
- → moistureValue1 = map(moistureValue1, 920, 200, 0, 100); // Werte umrechnen(!)
- → digitalWrite(relaisPin1, LOW); // schaltet das Relais ein (invertierte Logik!!!)
- → cycleNumb1++; // erhöht den Wert nach jedem Zyklus (+1)
- → if (moistureValue1 < 50 && moistureValue1 > 10 && cycleNumb1 <= cycleMax1) // wenn der Sensor Wert zwischen 10-50% liegt, wird eine Bewässerung ausgelöst
- → buttonValue1 = digitalRead(button1); // ...kann auch manuell ausgelöst werden
- → if (moistureValue1 <= 10) // merkt, wenn der Sensor abgehängt ist
- → if (cycleNumb1 >= cycleMax1) // und wenn die max. Anzahl Zyklen erreicht ist
- → digitalWrite(displayLightPin, HIGH); // schaltet die Hintergrundbeleuchtung des Displays aus (invertierte Logik!!!)

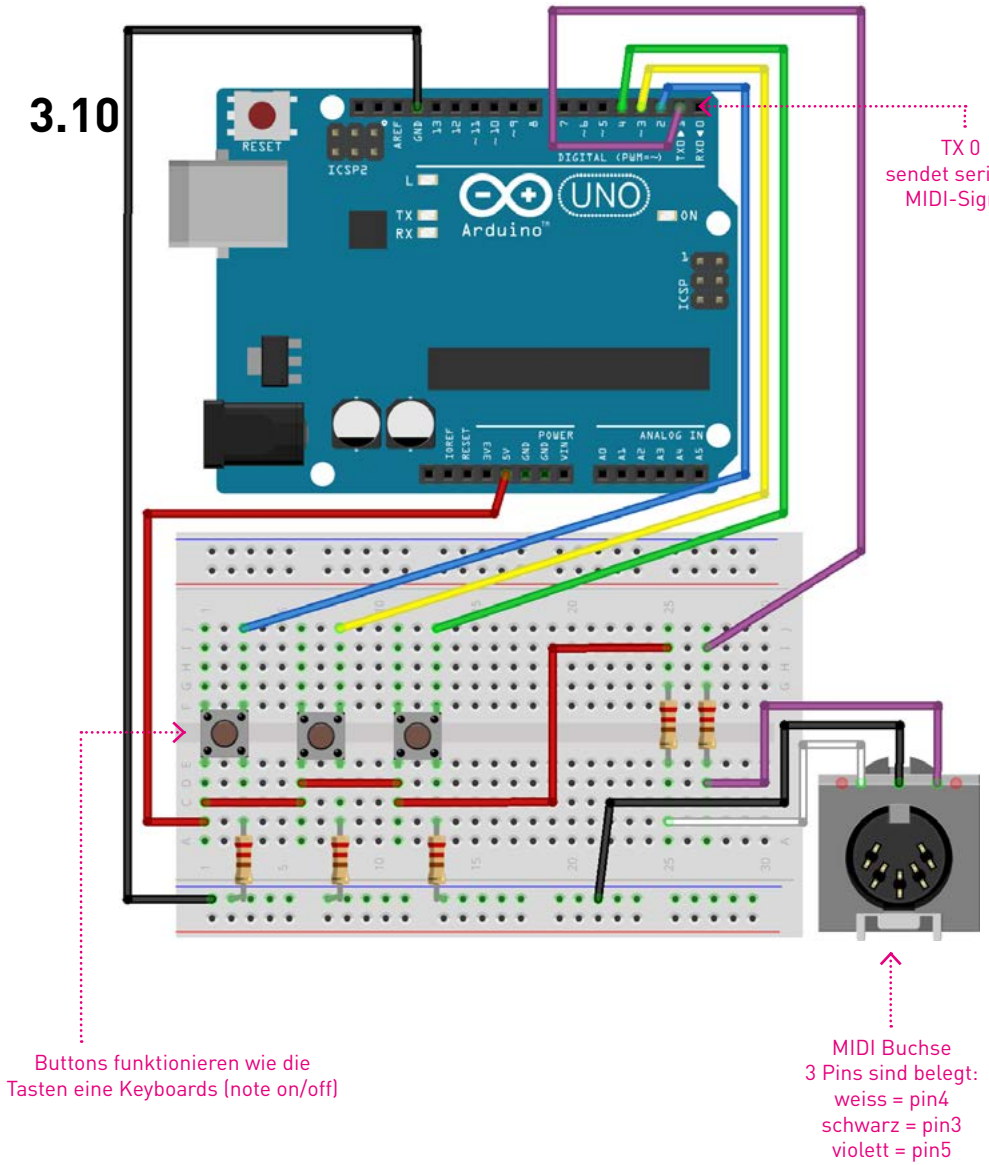
PROBIER DAS HIER NOCH AUS

- → Verändere die Zeiten der Intervalle – was sind da sinnvolle Werte?
- → Ein Rotary Encoder wäre doch noch ganz hilfreich, um einige Einstellungen der Intervalle direkt via Display vorzunehmen?!

TIP

- → Der Feuchtigkeits-Sensor (FC-28) kann auch ein Trigger-Signal senden (I/O)

3.10 MIDI OUTPUT DEVICE



Mit einfachen Push-Buttons lässt sich ein MIDI-Keyboard bauen und z.B. ein Software Synthesizer spielen... have fun!

ERKLÄRUNGEN ZUM CDOE

→ #include <MIDI.h> // hier wird die MIDI-Library eingebunden

→ Button BU1(2, 0, 60, 1, 5); // so wird ein Button definiert

→ void updateButtons() // in dieser Funktion werden die Buttons abgefragt

→ for (int i = 0; i < NUMBER_BUTTONS; i = i + 1) // For-Schleife für das Array

→ void updatePots() // hier weden die Potis abgefragt**PROBIER DAS HIER NOCH AUS**

→ Erweitere dein Keyboard mit weiteren Tasten. Welche stellen im Code müssen dazu kopiert werden?

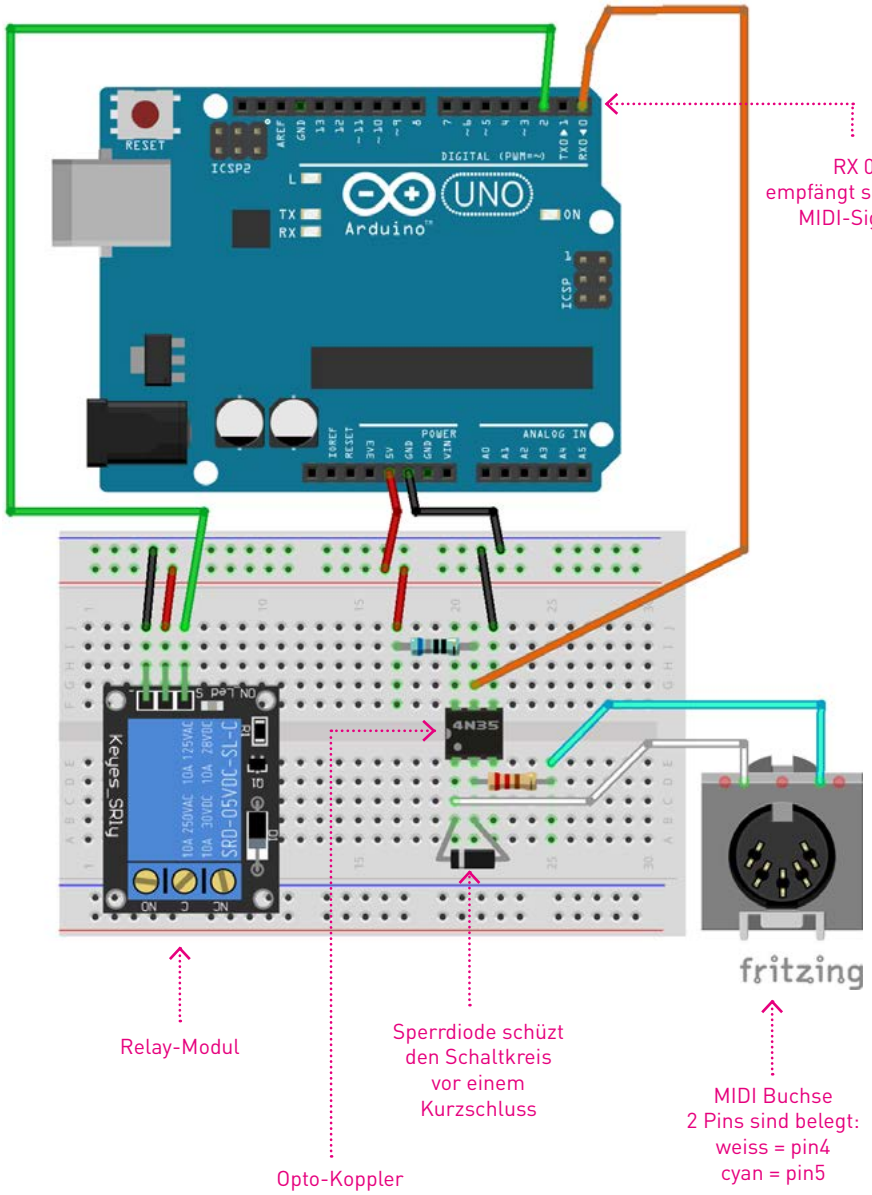
→ Nimm noch einen Potentiometer dazu um z.B. die Lautstärke oder ein Filter ansteuern zu können.

TIP

→ In diesem Code-Beispiel können Buttons als Note, Controll Change oder als Toggle verwendet werden(!)

→ Notes and Volts macht gute Lernvideos zu MIDI-Controller, das Code-Beispiel hier ist von ihnen und wurde angepasst.

3.11 MIDI INPUT DEVICE



Damit kann man z.B. mit einem MIDI-Keyboard oder einem Computer über die MIDI-Schnittstelle Motoren und Relais steuern...

ERKLÄRUNGEN ZUM CDOE

→ #include <MIDI.h> // hier wird die MIDI-Library eingebunden

→ Serial.begin(31250); // standard Baud-Rate für MIDI-Signale

→ MIDI.setHandleNoteOn(MidiInputNoteOn); // hier wird überprüft wenn eine Note gedrückt wird

→ MIDI.setHandleNoteOff(MidiInputNoteOff); // und hier wird überprüft wenn die Note losgelassen wird

→ MIDI.read(); // Funktionen die den Serial-Bus ausliest

→ if (pitch == 60) digitalWrite (relA, LOW); // wenn die Note C3 gedrückt wird, schaltet das den Pin, mit dem das Relay verbunden ist

PROBIER DAS HIER NOCH AUS

→ versuche zwei oder mehrere Relays anzuschliessen. Welche stellen im Code müssen dazu kopiert werden?

→ Ab wann brauchst du einen Multiplexer?

TIP

→ Dieses Relay kann einen grösseren Stromkreis (von max. 250 Volt) schalten. Das darf aber nur durch einen Fachmann oder -frau gemacht werden!

3.10

DAS NÄCHSTE HAMMER PROJEKT...

→ Hast Du eine Idee, eine Anregung oder einen Wunsch? Lass es uns wissen und schreib eine E-Mail an: labmanager@fablab-luzern.ch