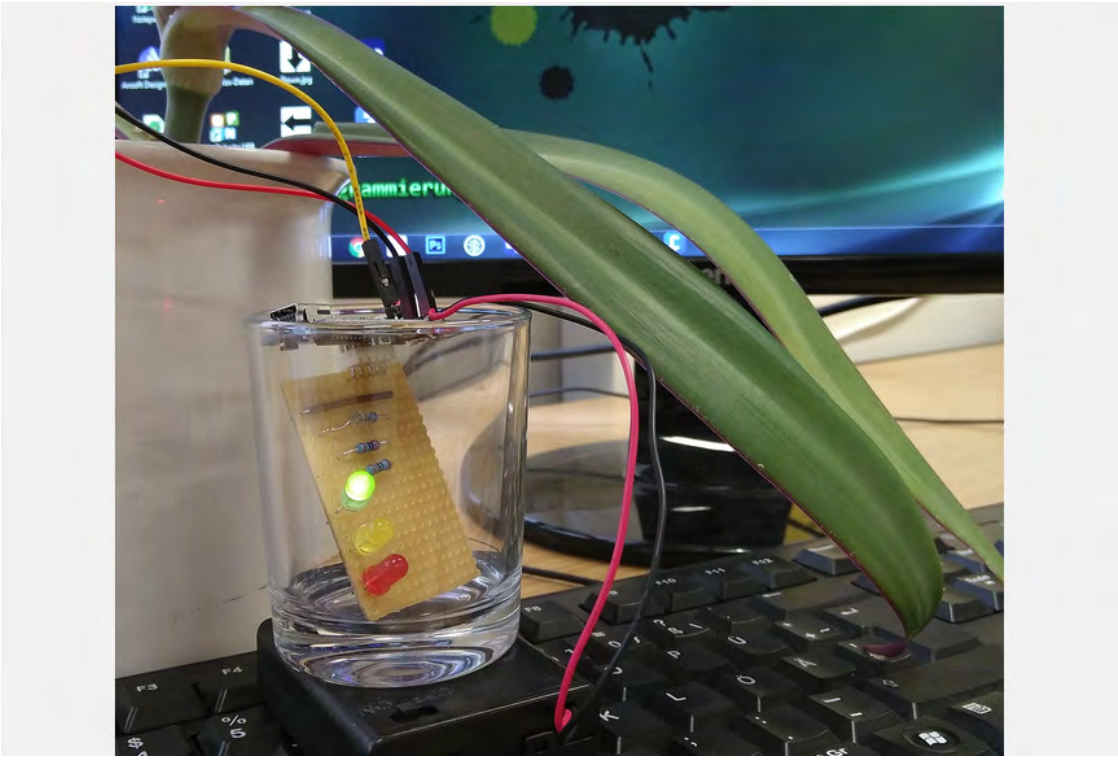
Projekt:
Blumenwächter

Leuchtender Gießalarm für Zimmerpflanzen



das Elektrotechnik- und Informatik-Labor der Fakultät IV
http://www.dein-labor.tu-berlin.de

Handout zum Projekt:

Blumenwächter

Leuchtender Gießalarm für Zimmerpflanzen

Inhaltsverzeichnis

1 Einleitung **1**

2 Elektrotechnik **2**
2.1 Strom und Spannung 2
2.2 Widerstand 3
2.3 Leuchtdiode 4

3 Aufbau und Verdrahtung des Blumenwächters **6**
3.1 Der Mikrocontroller: Arduino Nano 6
3.2 Arduino-Treiber und -Entwicklungsumgebung installieren 7
3.3 Drei LEDs leuchten lassen 7
3.4 Drei LEDs blinken lassen 8

4 Den Blumenwächter zum Leben erwecken **11**
4.1 Der Feuchtigkeitssensor 11
4.2 Verdrahtung von Feuchtigkeitssensor und Arduino 12
4.3 Programmierung des Blumenwächters 13
4.3.1 Anzeigen der Werte vom Feuchtigkeitssensor 13
4.3.2 LEDs leuchten abhängig vom Feuchtigkeitswert 14
4.4 Endmontage: Alles auf eine Streifenrasterplatine löten 15

5 Bezugsquellen für Baumaterialien und Werkzeuge **16**

1 Einleitung

Pflanzen sind Lebewesen und brauchen wie wir Menschen Wasser zum Überleben. Wenn du zu Hause eine Pflanze im Zimmer hast, weißt du, dass du sie auch gießen musst. Aber wann genau braucht die Pflanze Wasser? Reicht es, sie ein Mal in der Woche zu gießen? Oder solltest du dich doch lieber täglich um sie kümmern? Dieses Problem lösen wir in diesem Workshop mit unserem Blumenwächter. Er schlägt Gießalarm, wenn die Blumenerde zu trocken wird.

Im Workshop erfährst du zunächst, welche Bauteile wir brauchen. Unser „Gießalarm“ ist eher eine „Gießampel“: Wir wollen eine rote LED leuchten lassen, wenn Gießbedarf ist, eine gelbe, wenn noch alles halbwegs ok ist, und eine grüne, wenn die Erde richtig gut feucht ist.

Außer drei bunten LEDs brauchen wir also noch einen Feuchtigkeitssensor, eine Batterie, eine elektronische Schaltung, die alles verbindet, und ein Programm, mit dem wir die Steuerung umsetzen. Auf geht’s!

2 Elektrotechnik

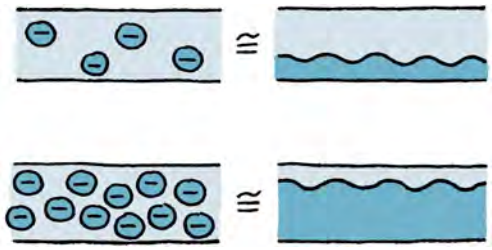
Die Elektrotechnik findet man an jeder Straßenecke. Überall blinkt es, große Leuchttafeln erzählen uns, was wir als nächstes kaufen sollen, und zu Hause flimmern im Fernseher die neuesten Nachrichten vor sich hin. Die Elektrotechnik hat sich zur Aufgabe gemacht, Elektrizität näher zu erforschen und sie in technischen Geräten einzusetzen.

WICHTIG
Die Elektrotechnik beschäftigt sich mit der Änderung von Strom und Spannung.

2.1 Strom und Spannung

Wenn wir von einer »Strömung im Fluss« sprechen, weiß jeder, was damit gemeint ist. Ein großer Fluss heißt ja auch Strom. Wenn ein Fluss schnell fließt und viel Wasser enthält, dann hat er eine hohe Strömung. Das Gleiche gibt es in der Elektrotechnik. Statt Wasser im Flussbett haben wir ganz kleine Teilchen, die sich in einem Draht bewegen. Wir nennen diese Teilchen Ladungsträger und deren Bewegung Strom. Es gibt zwei unterschiedliche Arten von Ladungsträgern, man nennt sie negativ geladen oder positiv geladen.

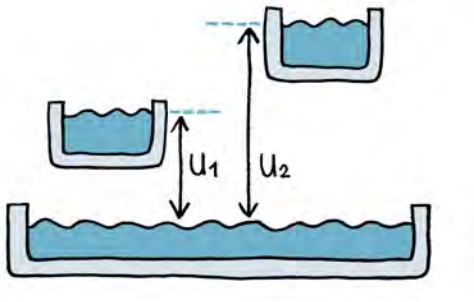
Negativ geladene Teilchen heißen auch *Elektronen*. Sie beschreiben den Stromfluss und werden oft als kleine Kreise mit einem Minussymbol darin gemalt. Fließen wenige Elektronen pro Zeiteinheit durch einem Draht, dann sprechen wir von einer geringen Stromstärke (oben im Bild rechts). Sind es dagegen viele Ladungsträger, so sprechen wir von einer hohen Stromstärke (unten im Bild).



WICHTIG
Wenn sich in der Elektrotechnik Ladungsträger bewegen, spricht man von *Strom*.
Formelzeichen: I für die Stromstärke
Einheit: A (*Ampere*) zum Andenken an den Physiker *André Marie Ampère*

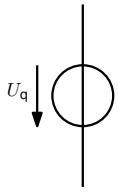
Wie fließt ein Fluss? Natürlich nur bergab, denn auf das Wasser wirkt die Erdanziehungskraft. Sie ist die Ursache der Strömung. Alles »will« nach unten: ein Apfel, den ich fallen lasse, oder das Wasser, das bergab fließt. In der Elektrotechnik gibt es ebenfalls eine Ursache für Stromfluss, diese nennen wir Spannung (auch Potentialunterschied genannt). Nur wenn Spannung vorhanden ist, bewegen sich die Ladungsträger. Nur dann also fließt Strom.

Du kannst dir eine kleinere oder höhere Spannung so vorstellen, als hättest du zwei obere Wasserbecken in unterschiedlichen Höhen und ein unteres Wasserbecken. Je höher das obere Becken, desto höher ist die Spannung (im Bild U_1 und U_2 genannt). Noch fließt hier kein Wasser. Aber sobald die Becken über Rohrleitungen miteinander verbunden werden, kann Wasser von den oberen Becken in das untere fließen.



WICHTIG
Spannung ist die Ursache, die Ladungsträger in Bewegung setzt.
Formelzeichen: U
Einheit: V (*Volt*) zum Andenken an den Physiker *Alessandro Volta*

In der Elektrotechnik gibt es Schaltungssymbole, damit jeder elektrotechnisch Begeisterte auf der Welt weiß, was man für Bauteile in einer Schaltung gemeint sind. Rechts siehst du das Symbol für eine Spannungsquelle.

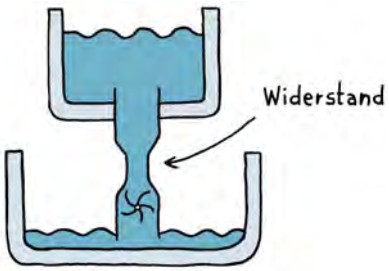


Wir haben Spannung und Strom kennengelernt. Jetzt brauchen wir noch die Bauteile, die es uns ermöglichen Strom und Spannung zu ändern.

2.2 Widerstand

Widerstand ist, wenn man sich gegen etwas wehrt! Das Bauelement *Widerstand* macht dies auch, es »wehrt sich« gegen den Stromfluss. Tatsächlich reduziert ein Widerstand die Stromstärke.

Mithilfe unseres Wassermodells kannst du dir das so vorstellen, dass der Widerstand den Querschnitt des Wasserrohrs reduziert. Die Wasserteilchen müssen sich durch die Engstelle hindurchquetschen und verlieren an Schwung. Dadurch kann im Vergleich zum Rohr ohne Verengung nur eine reduzierte Menge Wasser (Strom) in einer bestimmten Zeit durch das Rohr fließen.



Den Zusammenhang zwischen Widerstand, Spannung und Stromstärke kann man auch in einer Formel ausdrücken.

In der Formel nennen wir den Widerstandswert R (wegen des englischen Worts *resistor* für Widerstand), den Spannungswert U und den Wert der Stromstärke I. Diese für Elektrotechniker sehr wichtige Formel lautet:

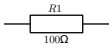
$$\boxed{R = \frac{U}{I}}$$

Die Formel bedeutet, der Widerstandswert ist gleich dem Ergebnis von Spannungswert geteilt durch Stromstärke. Weil man die Spannung U in Volt (V) misst, und die Stromstärke I in Ampere (A), ergibt sich, dass man den Widerstand $R = \frac{U}{I}$ in der Einheit $\frac{V}{A}$ misst. Der Widerstand ist allerdings eine so wichtige Größe, dass man der Einheit $\frac{V}{A}$ den Namen Ohm gegeben hat (nach dem Entdecker der Formel, dem Forscher Georg Simon Ohm). Die Formel heißt daher auch *Ohmsches Gesetz*. Häufig wird die Einheit Ohm durch den griechischen Buchstaben Omega (Ω) abgekürzt.

WICHTIG

Je höher der *Widerstand*, desto mehr Spannung braucht man, um einen bestimmten Strom fließen zu lassen.
Formelzeichen: R
Einheit: Ω, (Ohm), zum Andenken an den Physiker *Georg Simon Ohm*

In Schaltplänen wird ein Widerstand anhand eines rechteckigen Schaltsymbols dargestellt. Das ähnelt auch dem Bauteil, welches aussieht wie eine kleine, mit bunten Ringen bemalte Walze:

Schaltsymbol: 

Bauteil:

2.3 Leuchtdiode

Dioden sind *Halbleiterbauelemente*: sie leiten den Strom (je nach Temperatur) mal mehr, mal weniger gut. Das Schaltsymbol der Diode ist ein Dreieck mit einem senkrechten Strich an der Spitze. Wir verwenden **Leuchtdioden** (Licht emittierende Dioden, LEDs). Fließt durch eine LED elektrischer Strom, so strahlt sie Licht in verschiedenen Farben ab. Das sieht man am Schaltsymbol an den ausgehenden kleinen Pfeilen. Die meisten LEDs haben ein »Plastikköpfchen« und zwei unterschiedlich lange Anschlussdrähte. Das lange Anschlussbein markiert den Pluspol der LED (auch *Anode* genannt), das kurze den Minuspol (*Kathode*).

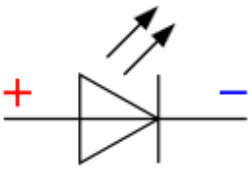


Abbildung 1: Leuchtdiode: Schaltsymbol (links) und Bauteile (rechts)

Wenn das Dreieck in einem Stromkreis in Richtung Minuspol weist, dann ist die Diode richtig herum (in *Durchlassrichtung*) eingebaut und lässt den Strom hindurchfließen. Weist das Dreieck in Richtung Pluspol, so lässt die Diode keinen Strom durch (sie *sperrt*, bzw. ist in *Sperrichtung*

eingebaut). Das kannst du dir wie ein Ventil mit einer Kugel und einer Sprungfeder in einem Wasserrohr vorstellen: Wenn ein Druck (eine Spannung) auf dieses Ventil (Diode) in Sperrrichtung wirkt, wird der Wasserfluss (Strom) blockiert. In Durchlassrichtung muss der Druck (die Spannung) groß genug sein, um die Federkraft des Ventils zu überwinden. Dadurch öffnet das Ventil (die Diode) den Durchlass, und der Strom kann fließen. Diesem Druck, welcher im Wassermodell zum Überwinden der Federkraft notwendig ist, entspricht bei einer Diode die so genannte *Schwellenspannung*, die in Stromflussrichtung an der Diode anliegen muss, damit sie leitend wird. Bei gewöhnlichen Siliziumdioden liegt diese Spannung bei ca. 0,6 - 0,7 V.

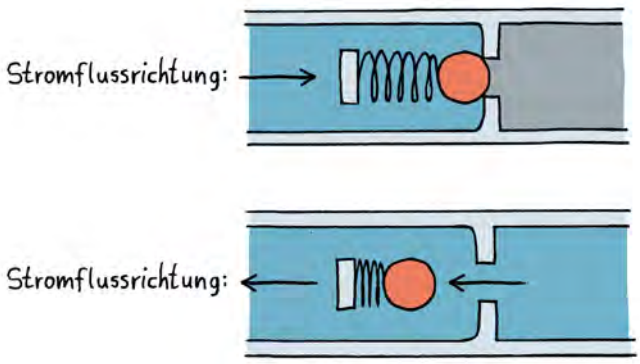


Abbildung 2: Diode als Ventil in Sperrrichtung (oben) und in Durchlassrichtung (unten)

Weil Dioden von sich aus den Strom nicht begrenzen, darf man sie niemals ohne Widerstand (genannt Vorwiderstand) an eine Spannungsquelle anschließen. Dioden vertragen ungefähr 20 bis 50 mA. Ein höherer Strom würde sehr schnell zum Tod des Bauelementes führen. Den Vorwiderstand berechnest du so:

$$R = \frac{U_{gesamt} - U_{LED}}{I_{LED}}$$

R: gesuchter Vorwiderstand,
U_{gesamt}: vorhandene Betriebsspannung,
U_{LED}: Durchlassspannung der LED (aus Datenblatt),
I_{LED}: Betriebsstrom der LED (aus Datenblatt)

Mit den Beispielwerten $U_{gesamt} = 9V$, $U_{LED} = 1,6V$ und $I_{LED} = 30mA$ erhältst du $R = 247\Omega$.

3 Aufbau und Verdrahtung des Blumenwächters

Der Blumenwächter besteht aus vier Komponenten: Das Herzstück ist ein *Mikrocontroller*, das ist ein kleiner programmierbarer Computer. Er heißt *Arduino Nano*. Über Kabel wird er mit den anderen drei Komponenten verbunden. Da ist erstens der Feuchtigkeitssensor, der misst, ob die Erde noch feucht genug ist, und die Werte an den Mikrocontroller schickt. Zweitens haben wir dann die LED-Schaltung, also eine Platine, auf die wir drei farbige LEDs löten. Und drittens gibt es noch die Stromversorgung, also ein Kästchen mit drei Batterien, die den Arduino und unsere Schaltung mit Energie versorgt. In diesem Kapitel stellen wir dir alle Komponenten genauer vor.

3.1 Der Mikrocontroller: Arduino Nano

Der Mikrocontroller, den wir zur Steuerung des Blumenwächters verwenden, heißt *Arduino Nano*. Er hat einen Speicher für ein Programm und an den Rändern *Pins*, also Anschlüsse für Kabel, um elektronische Bauteile mit dem Arduino zu verbinden. Die Programmierung beschreiben wir später. Hier geht es zunächst nur um den Aufbau des Arduino-Nano-Mikrocontrollers. In Abbildung 3 siehst du, wie ein Arduino Nano aussieht.

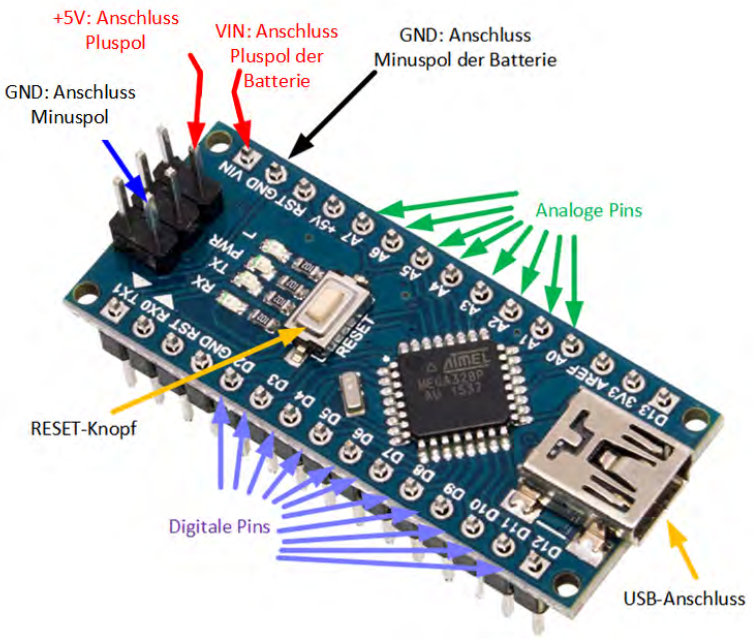


Abbildung 3: Aufbau des Arduino-Nano-Mikrocontrollers

An den Arduino lässt sich eine Batterie mit einer Spannung zwischen 5V und 12 V anschließen. Den Pluspol der Batterie muss man mit dem *VIN*-Pin verbinden und den Minuspol mit einem *GND*-Pin. Hast du den Arduino über die USB-Mini-Buchse mit dem PC oder Laptop verbunden, dann brauchst du gar keine Batterie, weil der Arduino dann über das USB-Kabel Strom und Spannung vom Computer bekommt. An die anderen Pins kann man nun elektro-

nische Bauteile anschließen, wie zum Beispiel Sensoren oder unsere drei farbigen LEDs. Die mit A beschrifteten Pins A0 bis A7 heißen *Analoge Pins*, weil hier Sensoren verbunden werden können, die Werte aus der realen (analogen) Welt messen können, wie zum Beispiel ein Feuchtigkeitsmesser. Auf der gegenüberliegenden Seite gibt es mit D beschriftete Pins D2 bis D12. Diese heißen *Digitale Pins*. Hier können Bauteile wie LEDs angeschlossen werden, die nur zwei Zustände (*An* oder *Aus*) haben können.

3.2 Arduino-Treiber und -Entwicklungsumgebung installieren

Damit dein Computer den Arduino-Nano als Gerät erkennt und später deine Programme auf den Arduino überträgt, brauchst du Software, und zwar einen *Treiber* und die *Arduino-Entwicklungsumgebung*. Beides kannst du dir im Internet herunterladen und installieren.

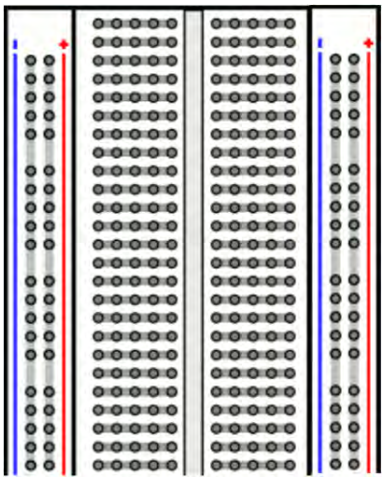
- **Treiber:** Viele Arduino-kompatible Boards benötigen einen USB-Treiber, damit sie vom Computer richtig erkannt werden. Der vom Arduino Nano meistverwendete USB-Chipsatz ist der CH340G bzw. CH341SER USB Chipsatz. Wenn du einen Windows-Rechner hast, dann lade die Treibersoftware hier herunter: http://www.wch-ic.com/downloads/CH341SER_ZIP.html, indem du auf den Download-Button klickst und die heruntergeladene Datei CH341SER.ZIP in deinem Download-Ordner speicherst. Wenn dein Rechner ein anderes Betriebssystem hat, dann folge den Installationsanweisungen auf der Makershop-Seite: https://www.makershop.de/ch340-341-usb-installieren/. Entpacke die heruntergeladene Zip-Datei und installiere den Treiber, indem du die Datei SETUP.EXE durch Doppel-Klicken ausführst. Sobald die Treiber Installation erfolgreich war, muss der Computer in den meisten Fällen einmal neu gestartet werden. Danach sollte der Treiber vollständig installiert sein. Sobald nun der Arduino Nano, über ein USB-Kabel mit dem PC verbunden wird, sollte das entsprechende Gerät im Gerätemanager unter Windows oder bei MAC OS direkt erkannt werden.

- **Entwicklungsumgebung:** Die Arduino-Entwicklungsumgebung (oder IDE (Integrated Development Environment) kannst du hier herunterladen: https://www.arduino.cc/en/software. Wähle bei *Download Options* dein Betriebssystem aus (z.B. *Windows - Win 7 and newer*) und klicke danach auf *Just Download*. Sobald die Installationsdatei arduino-1.8.13-windows.exe herutergeladen wurde, kannst du die Installation über das Download-Symbol vom Browser aus beginnen.

  Es wird ein Startsymbol auf dem Desktop erstellt (rechts zu sehen), mit dem du später die Arduino-IDE aufrufen kannst.

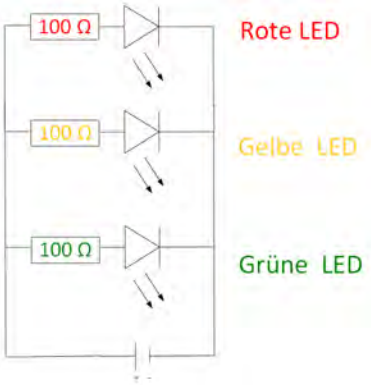
3.3 Drei LEDs leuchten lassen

Um drei LEDs leuchten zu lassen, musst du kein Programm schreiben. Du nutzt den Arduino als Spannungsquelle, indem du ihn über den USB-Port mit dem Computer verbindest. Nun fehlt noch der Stromkreis mit drei LEDs. Wir verwenden dafür ein Steckbrett (engl. Breadboard).



Auf der Zeichnung vom Steckbrett siehst du, welche Löcher unten durch Kupferkabel verbunden sind (die Verbindungen werden auch „Busse“genannt): Alle Löcher an der roten senkrechten Linie außen sind miteinander verbunden; genauso alle Löcher entlang der blauen Linie. In die rote Leiste wird üblicherweise ein (rotes) Verbindungskabel zum Pluspol der Batterie gesteckt (bei uns: zum 5V-Anschluss auf dem Arduino). In die blaue Leiste kommt das (meist schwarze) Kabel zum Minuspol der Batterie (bei uns: zum GND-Anschluss auf dem Arduino). Die Pluspol- und Minuspol-Leisten gibt es zweimal, links und rechts außen. Im inneren Teil sind immer fünf Löcher miteinander quer verbunden. In der Mitte gibt es einen großen Zwischenraum.

In unserem Stromkreis sollen drei LEDs parallel geschaltet werden und gleichzeitig leuchten. Eine LED kann aber nicht unmittelbar an eine 5V-Spannungsquelle angeschlossen werden. Wir müssen eine Strombegrenzung in unsere Schaltung einbauen.



Die einfachste Möglichkeit ist, in Reihe zu jeder LED einen Vorwiderstand zu schalten. Wie aber kommen wir darauf, dass die Vorwiderstände einen Wert von $R = 100\Omega$ haben sollten? Zur Berechnung des Vorwiderstands R gilt

$$R = \frac{U_{gesamt} - U_{LED}}{I_{LED}}$$

Bei uns ist $U_{gesamt} = 5V$, die Durchlassspannung der LED ist $U_{LED} = 2V$, und der Betriebsstrom der LED ist $I_{LED} = 30mA$. Es ergibt sich also:

$$R = \frac{5V - 2V}{0,03A} = 100\Omega$$

Diesen Schaltplan überträgst du nun auf das Steckbrett: Zunächst steckst du die LEDs (rot, gelb und grün) so, dass die kurzen Beinchen (Minuspole) jeder LED in der Minusleiste (blau) stecken, und die langen Beinchen jedes in einer extra Reihe in der Mitte. Genau in diese Reihe kommt dann zu jeder LED ein Beinchen eines 100Ω-Widerstands. Das jeweils andere Beinchen von jedem Widerstand steckst du in die Plusleiste (rot) vom Steckbrett. Nun müssen nur noch die Plusleiste und die Minusleiste selbst mit einem 5V- und einem GND-Pin am Arduino verbunden werden. Wo du die findest, zeigt dir Abbildung 4.

3.4 Drei LEDs blinken lassen

Nun wollen wir die LEDs aber nicht nur einfach leuchten lassen. Sie sollen blinken, und zwar in einer Geschwindigkeit, die wir vorgeben. Wir wollen die LEDs also nun steuern, und zwar über den Arduino, mit einem Programm, was wir selber schreiben.

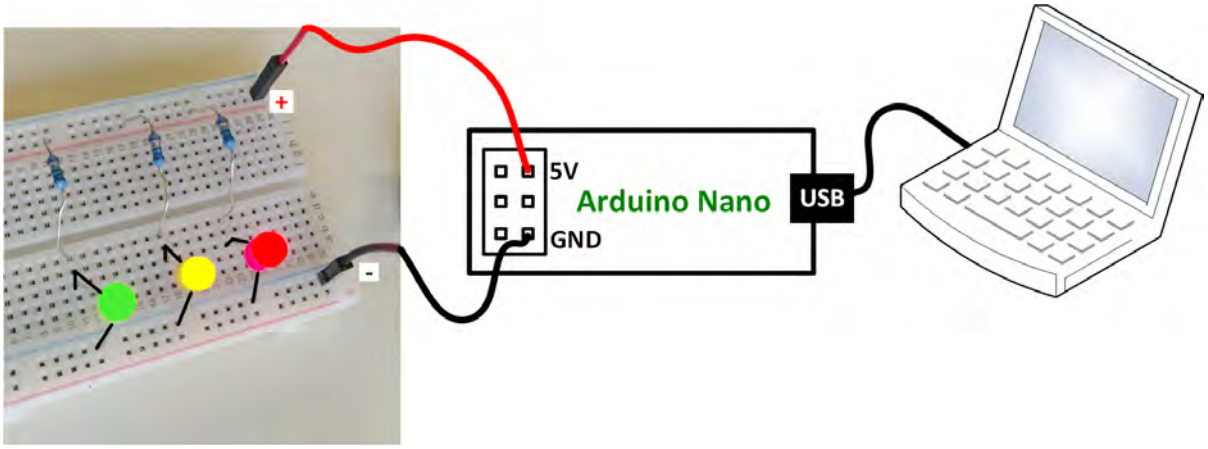


Abbildung 4: Die LED-Schaltung „Drei LEDs leuchten“

Dazu änderst du zunächst deine Schaltung: Da jede LED jetzt nur noch leuchten darf, wenn sie ein Signal vom Arduino erhält, musst du jeden LED-Vorwiderstand nun anstatt mit der Plusleiste auf dem Steckbrett mit einem digitalen Pin auf dem Arduino verbinden. Verschiebe die Beinchen der Widerstände aus der Plusleiste in den Mittelbereich vom Steckbrett. Das rote Kabel verwendest du nun für die Verbindung der Plusseite des roten LED-Vorwiderstands mit einem digitalen Pin, z.B. Pin D8. Nimm ein gelbes und ein grünes Kabel und verbinde die Plusseiten der gelben und der grünen LED-Vorwiderstände jeweils mit den digitalen Pins D7 und D6, wie in Abbildung 5.

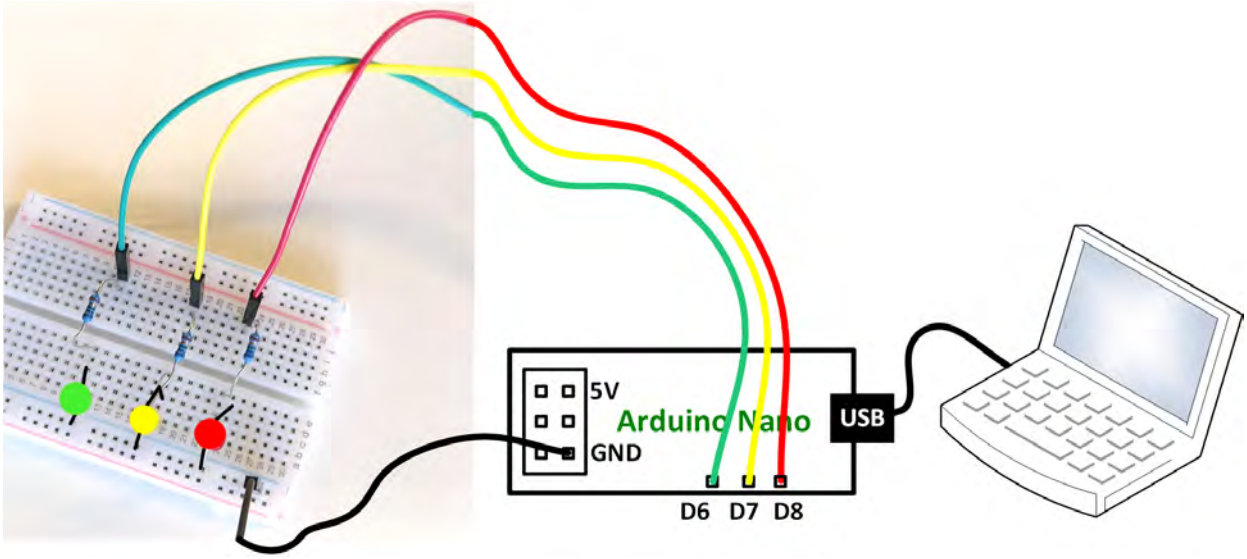


Abbildung 5: Die LED-Schaltung „Drei LEDs blinken“

Nun zur Programmierung: Starte das Programm *Arduino IDE*. Es öffnet sich das Fenster in Abbildung 6. In das weiße Textfenster in der Mitte kommt dein Code, der sogenannte *Sketch*. In der Leiste darüber gibt es zwei wichtige Buttons: Mit einem Klick auf das Häkchen ganz links wird dein Code auf Fehler überprüft. Mit einem Klick auf den Pfeil daneben wird das fertige Programm auf den angeschlossenen Arduino übertragen. Wähle zunächst die Einstellung *Arduino Nano* über das Menü *Werkzeuge* > *Board* > *Arduino AVR Boards* aus.

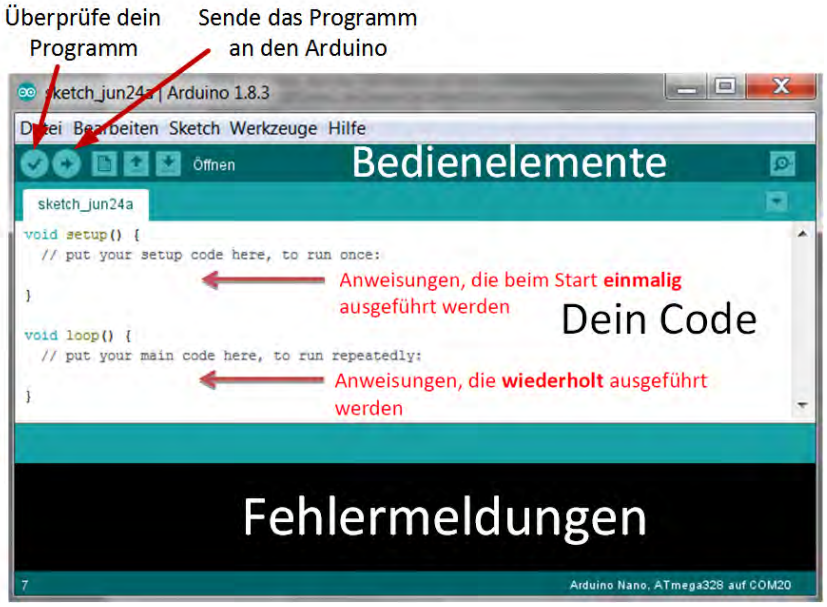


Abbildung 6: Die Arduino-Entwicklungsumgebung

Der Code für Arduino-Programme (eine Variante der Programmiersprache C) ist folgendermaßen strukturiert: Ganz oben, vor den Abschnitt **setup{ ... }** (auf Deutsch *Einrichtung*), gehören Variablen. Das sind Container für Zahlen oder andere Informationen, die wir im Laufe des Programms verwenden wollen. In den Abschnitt **setup{ ... }** gehören Anweisungen, die beim Start des Programms ein einziges Mal ausgeführt werden. Hier wird zum Beispiel eingestellt, mit welchen Pins unsere LEDs verbunden sind. In den Abschnitt **loop{ ... }** (auf Deutsch *Schleife*) gehören Anweisungen, die nach dem Start des Programms nicht nur einmal durchlaufen werden, sondern immer wieder neu, in der Reihenfolge, in der sie in **loop{ ... }** stehen. Wenn nun eine LED blinken soll, dann schreibst du hier den Befehl für das Anschalten der LED hinein, und darunter den Befehl für das Ausschalten der LED. Fertig. Da beide Befehle immer wieder wiederholt werden, sehen wir ein ständiges Wechseln von An- und Ausschalten der LED: sie blinkt. Übertrage und teste das Programm in Abbildung 7.[1] Erweitere es danach, so dass nicht nur eine sondern alle drei LEDs blinken.

[1] Gibt es Probleme beim Übertragen deines Sketches auf den Arduino Nano, so wähle über das Menü *Werkzeuge* > *Prozessor ATmega 328P* den Eintrag *Old Bootloader* aus. Das hilft bei einer etwas älteren Nano-Chip-Variante.

```
//--- Variablen: Namen statt Zahlen

int ledrotPin   = 8;
int ledgelbPin  = 7;
int ledgruenPin = 6;

//--- setup: Alles wird nur einmal durchlaufen

void setup() {  // Alle unsere LEDs sind Signale nach außen
  pinMode(ledrotPin, OUTPUT);
  pinMode(ledgelbPin, OUTPUT);
  pinMode(ledgruenPin, OUTPUT);
}

//--- loop: Am Ende beginnt es wieder von vorne

void loop() {
  digitalWrite(ledrotPin, HIGH);   // LED, gehe an
  delay(100);                      // Bleib so für 100 ms
  digitalWrite(ledrotPin, LOW);    // LED, gehe aus
  delay(100);                      // Bleib so für 100 ms
}
```

Abbildung 7: Das Programm „Drei LEDs blinken“

4 Den Blumenwächter zum Leben erwecken

Damit unser Blumenwächter weiß, welche LEDs er leuchten lassen soll, muss er wissen, wie feucht die Blumenerde ist. Dafür braucht er einen Sensor: den Feuchtigkeitssensor. Außerdem müssen wir unser Programm erweitern, damit die LEDs nur noch dann leuchten, wenn eine bestimmte Feuchtigkeit herrscht. Dazu brauchen wir „*Wenn-Dann-Sonst*-“Anweisungen (auf Englisch *if* . . . *else* . . .).

4.1 Der Feuchtigkeitssensor

In Abbildung 8 siehst du einen Feuchtigkeitssensor (engl. *moisture sensor*). Er sieht aus wie eine Gabel mit nur zwei Zinken. An diesen beiden Zinken, auch *Kontakte* genannt, liegt eine Spannung an. Die beiden Zinken werden in die Blumenerde gesteckt. Über zwei Kabel (rot und schwarz) wird der Sensor mit Strom versorgt, das dritte (gelb) dient zur Übertragung der Messwerte an den Arduino.

Je höher die Feuchtigkeit zwischen den beiden Kontakten ist, desto besser kann der Strom zwischen ihnen fließen. Die Information über den gemessenen Stromfluss wird dann als eine Zahl an den Mikrocontroller geschickt. Ein Feuchtigkeitssensor kann Zahlen im Bereich von 0 (Sensor ist trocken wie an der Luft) bis ca. 800 (Sensor ist komplett in Wasser getaucht) an den Arduino schicken. Die genaue Kalibrierung ist jedoch abhängig vom Sensor und von der Art der Flüssigkeit, die gemessen wird (bspw. hat Salzwasser eine bessere Leitfähigkeit und der Messwert wäre entsprechend höher). Mit den erhaltenen Zahlen können wir dann im Programm bestimmen, wann welche LED leuchten soll (z.B. bei „0“ die rote LED und bei „800“ die grüne.

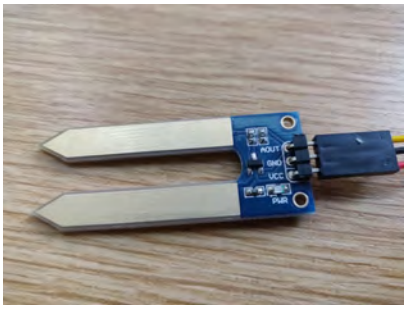


Abbildung 8: Der Feuchtigkeitssensor

4.2 Verdrahtung von Feuchtigkeitssensor und Arduino

Den Pluspol des Sensors verbindest du über das rote Kabel mit der Plusleiste vom Steckbrett, seinen Minuspol über das schwarze Kabel mit der Minusleiste. Nicht vergessen, die Plusleiste wieder mit dem 5V-Pin auf dem Arduino zu verbinden! Das gelbe Kabel am Sensor überträgt die Messwerte. Dieses verbindest du über den Umweg des Steckbretts mit dem analogen Pin *A0* auf dem Arduino (siehe Abbildung 9). Zur besseren Übersicht haben wir im Bild die drei LED-Kabel zum Arduino entfernt. Lasse sie aber ruhig drin.

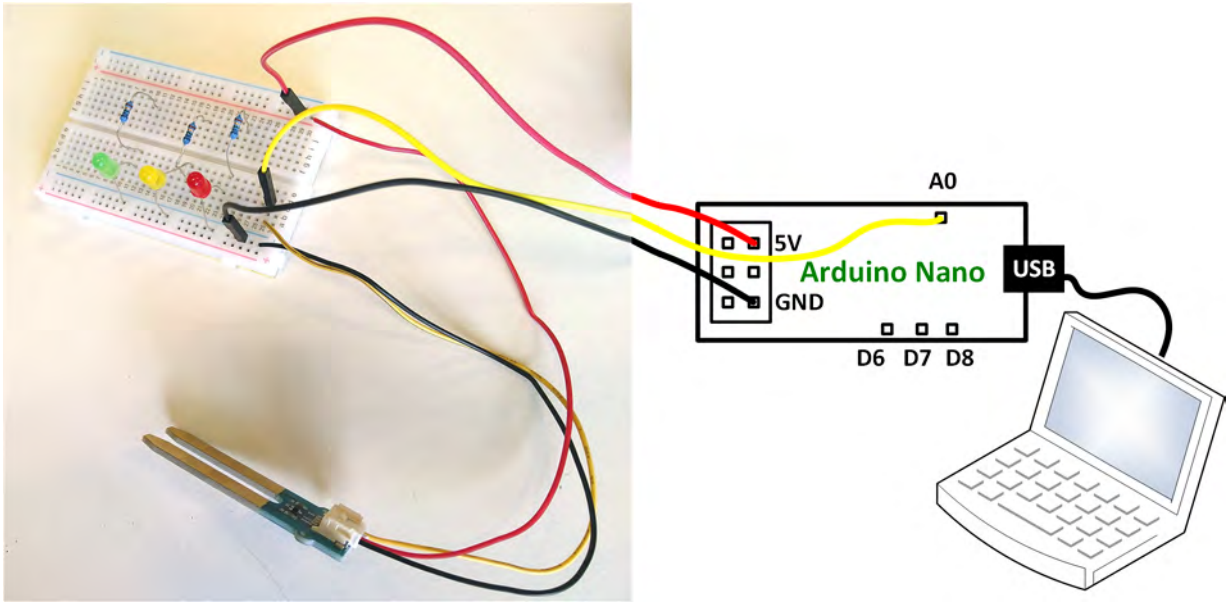


Abbildung 9: Schaltung mit Feuchtigkeitssensor

4.3 Programmierung des Blumenwächters

Die wichtigsten Befehle, die wir nun zusätzlich im **loop**-Bereich verwenden wollen, sind die Anweisungen zum Lesen der Zahlen vom Feuchtigkeitssensor und zum Ansteuern einer der drei LEDs: Rot soll leuchten, wenn die Sensor-Zahl sehr klein ist (die Erde ist trocken). Gelb soll so im mittleren Bereich leuchten, und Grün leuchtet, wenn die Zahl recht hoch ist (die Erde ist nass).

4.3.1 Anzeigen der Werte vom Feuchtigkeitssensor

Als erstes programmieren wir, dass der Wert, den der Feuchtigkeitssensor an den Arduino schickt, auf den Bildschirm in ein extra Fenster geschrieben wird. Dadurch können wir diese Zahl auch selber lesen und danach entscheiden, bei welcher Zahlen wir welche LED-Farbe leuchten lassen wollen. Wir müssen also im Loop-Block zwei Dinge tun: 1) den Messwert vom Feuchtigkeitssensor lesen und in einer Variablen speichern; 2) diesen Wert auf den Bildschirm schreiben. Danach beginnt das Programm ja wieder von vorne, also werden ständig neue Werte gelesen und geschrieben. Bald ist das Fenster voller Zahlen ...

Die Variable, in die wir immer wieder den neuesten Wert des Sensors speichern, heißt *messwert*. Dass dies eine ganze Zahl ist, sagen wir dem Programm ganz oben. Das Fenster, in dem uns die Werte angezeigt werden sollen, heißt „Serieller Monitor“ (serial monitor). Im Setup-Block musst du diesen seriellen Monitor starten. Abbildung 10 zeigt das Programm. Schreibe es ab und teste es mit deinem Sensor. Dieses Programm läuft auch ohne die drei LEDs.

```
int messwert=0;                 //Unter der Variablen „messwert“ wird später der
                                Messwert des Sensors gespeichert.

void setup()
{                               // Hier beginnt das Setup.
Serial.begin(9600);             //Die Kommunikation mit dem seriellen Port wird
                                gestartet. Das benötigt man, um sich den
                                ausgelesenen Wert im serial monitor anzeigen zu
                                lassen.
}

void loop()
{                               // Mit dieser Klammer wird der Loop-Teil geöffnet.
messwert=analogRead(A0);        //Die Spannung an dem Sensor wird ausgelesen und
                                unter der Variable „messwert“ gespeichert.
Serial.print("Feuchtigkeits-Messwert:");  //Ausgabe am Serial-Monitor: Das Wort
                                            „Feuchtigkeits-Messwert:”
Serial.println(messwert);       //und im Anschluss der eigentliche Messwert
delay(500);                     //Zum Schluss noch eine kleine Pause, damit nicht
                                zu viele Zahlenwerte über den Serial-Monitor
                                rauschen.
}
```

Abbildung 10: Das Programm zum Anzeigen der Messwerte

4.3.2 LEDs leuchten abhängig vom Feuchtigkeitswert

Nun fehlen nur noch die Anweisungen, welche LED wann leuchten soll. Die LEDs müssen dafür wieder mit den Pins D6, D7 und D8 verbunden sein.

Im Programm fragst du den Wert (Variable *messwert*) ab und entscheidest je nach aktueller Zahl, was passieren soll. Beispielsweise kannst du die rote LED leuchten lassen, wenn der Messwert zwischen 0 und 300 liegt, die gelbe LED bei einem Messwert zwischen 301 und 500 und die grüne LED bei Werten über 500. Vergiss nicht, die anderen beiden LEDs auszuschalten, wenn nur die rote LED an sein soll. Das Programm dazu findest du in Abbildung 11.

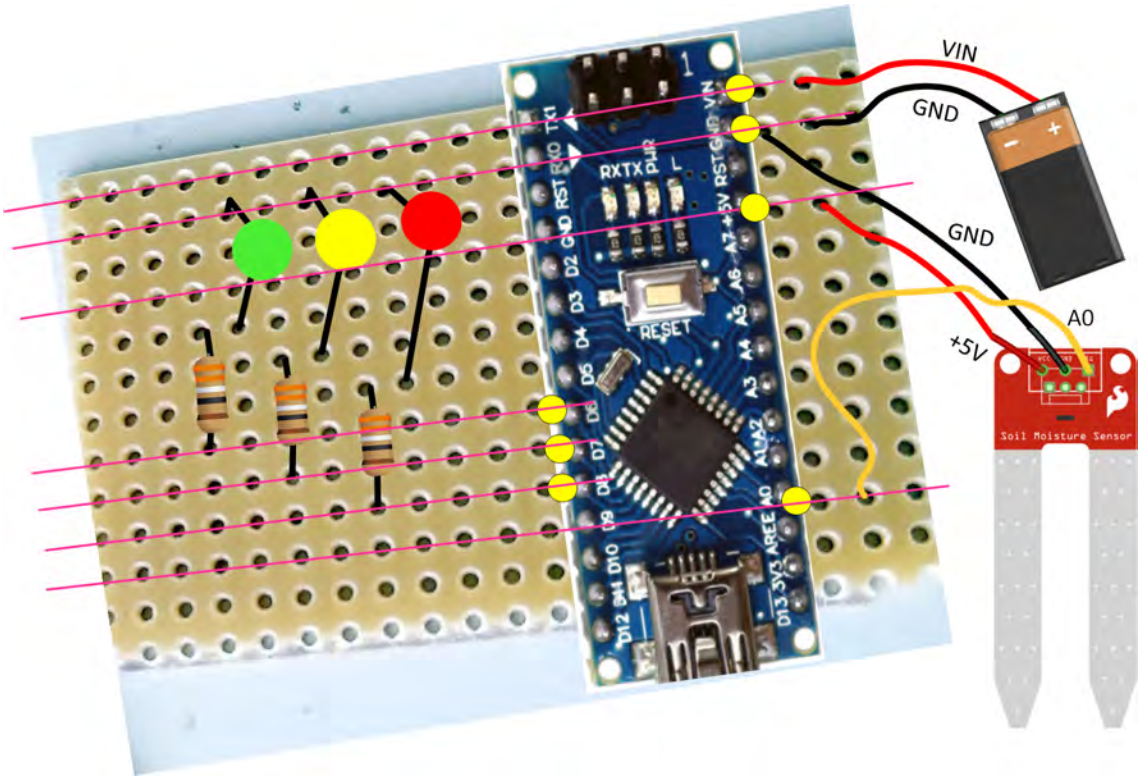
```
//--- Variablen: Namen statt Zahlen

int ledgruenPin = 6;
int ledgelbPin  = 7;
int ledrotPin   = 8;
int sensorPin = 0;
int messwert = 0;

//--- setup: Alles wird nur einmal durchlaufen

void setup() {  // Alle unsere LEDs sind Signale nach außen
  pinMode(ledrotPin, OUTPUT);
  pinMode(ledgelbPin, OUTPUT);
  pinMode(ledgruenPin, OUTPUT);
  Serial.begin(9600);
}

//--- loop: Am Ende beginnt es wieder von vorne

void loop() {
  messwert = analogRead(sensorPin);
  Serial.println(messwert);

  if(messwert < 200){
  digitalWrite(ledrotPin, HIGH);    // Wenn trocken, dann
  digitalWrite(ledgelbPin, LOW);    // leuchte in rot
  digitalWrite(ledgruenPin, LOW);
  }
  if((messwert > 200) && (messwert < 500)){
  digitalWrite(ledrotPin, LOW);     // Wenn mittelfeucht, dann
  digitalWrite(ledgelbPin, HIGH);   // leuchte in gelb
  digitalWrite(ledgruenPin, LOW);
  }
  if(messwert > 500){
  digitalWrite(ledrotPin, LOW);    // Wenn nass, dann
  digitalWrite(ledgelbPin, LOW);   // leuchte in grün
  digitalWrite(ledgruenPin, HIGH);
  }
}
```

Abbildung 11: Das Programm „Blumenwächter“

Fertig ist das Blumenwächter-Programm. Du kannst aber auch kreativ sein und die jeweils aktuelle LED rot, gelb oder grün blinken lassen, weil das auffälliger ist. Bekommst du das hin?

4.4 Endmontage: Alles auf eine Streifenrasterplatine löten

Wenn du löten kannst, dann solltest du aus dem bisherigen gesteckten Prototypen eine kompakte Platine bauen. Du brauchst dazu eine *Streifenplatine*, das ist eine Lochplatte aus Hartpappe mit auf einer Seite aufgeklebten Kupferstreifen. Die Bauteile werden auf der Pappseite mit ihren Anschlussbeinchen durch die Löcher gesteckt und auf der Kupferseite angelötet. Damit die Schaltung Strom bekommt, wird auch ein Batteriehalter über zwei Anschlusskabel an die Platine gelötet. Der Halter enthält eine 9V-Blockbatterie, die die bisherige USB-Stromversorgung über den PC ersetzt. So funktioniert der Blumenwächter dann auch, ohne mit dem PC verbunden zu sein. Hier siehst du ein *Platinenlayout*, also einen Plan, wie die Bauteile auf der Streifenplatine angeordnet werden müssen, damit die Schaltung funktioniert:



Die Richtung der Kupferstreifen auf der Platinenrückseite ist durch pinkfarbene Linien angedeutet. Die gelben Kreise zeigen dir, welche Anschlüsse des Arduino unten an die Platine gelötet werden müssen. Kannst du die Schaltung nachvollziehen, wenn du sie mit der Steckbrettvariante in Abbildung 9 vergleichst? Sie ist sehr ähnlich. Einen Unterschied stellst du aber fest: Der Pluspol der Batterie ist nicht am 5V-Anschluss angeschlossen, sondern an einem Anschluss namens *VIN (Voltage Input)*. Das ist ein Anschluss mit integriertem Spannungswandler, der die 9V Betriebsspannung der Batterie in 5V für den Mikroprozessor umwandelt. Bei der USB-Variante war das nicht nötig, da der USB-Anschluss genau 5V liefert.

Wie du siehst, brauchst du hier weniger Verbindungskabel als bei der Steckbrett-Variante. So passt die Schaltung am Ende in ein kleines Glas, so wie das auf dem Titelbild.

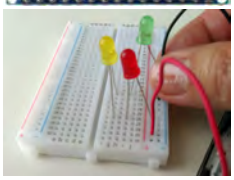
5 Bezugsquellen für Baumaterialien und Werkzeuge

Hier bekommst du Tipps, wo du Bauteile und Arbeitsmaterialien für die Blumenwächter-Schaltung günstig kaufen kannst. Wir empfehlen dir die Internet-Versandhändler **Segor** (https://www.segor.de/, **Reichelt** (https://www.reichelt.de/), **Makershop** (https://www.makershop.de/ und **Conrad** (https://www.conrad.de/)[2]. Der Einfachheit halber haben wir die jeweiligen Bezeichnungen/Artikelnummern bei den Bauteilen dazu geschrieben. Die Preise sind nur grobe Richtlinien und waren aktuell, als wir die Bauanleitung geschrieben und ausprobiert haben.

Aufbau auf dem Steckbrett: LEDs blinken lassen

» **Mikrocontroller** vom Typ *Arduino Nano*: „Nano Starter Set V3.1 Kit Arduino kompatibel“; Makershop, Preis: 4,70 €

» **Steckbrett** (auch Breadboard oder Protoboard genannt): „Breadboard 400“; Makershop, Preis: 3,60 €



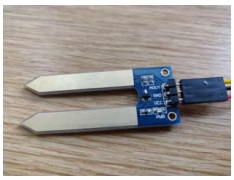
» **Verbindungskabel** (auch Jumper-Kabel genannt): „Jumper Wire Steckbrücken Steckbrett“; Makershop, 20 Stck., Preis: 2,49 €

» **Leuchtdioden rot, gelb und grün** (kaufe besser je 10 Stück): rot: „LED 5 rt-dif (10 mA)“, gelb: „LED 5 ge-dif (20 mA)“, grün: „LED 5 gn-dif (10 mA)“; Segor, Preis 0,08 € pro Stück ab 10 Stck. pro Farbe

» **drei Widerstände** (100 Ohm, 0,6 W): „METALL 100 Widerstand, Metallschicht, 100 Ohm, 0207, 0,6 W, 1%“; Reichelt, Preis 0,05 € pro Stück ab 10 Stck.

Aufbau auf dem Steckbrett: Feuchtigkeitssensor

» **Bodenfeuchtesensor**: „Iduino ME110 Bodenfeuchtesensor“; Conrad, Preis: 1,74 €
oder „ARD SEN WET3 Arduino - Feuchtesensor für Bodenfeuchte“; Reichelt, Preis: 4,86 €



[2] Die Firma Conrad hat in vielen Städten auch Filialen, wo du hingehen kannst.

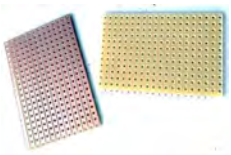
Endmontage: Alles auf die Streifenrasterplatine löten

Falls du noch nie gelötet hast: Eine Schritt-für-Schritt-Anleitung zum Löten-Lernen findest du hier: http://www.dein-labor.tu-berlin.de/sites/default/files/TU-it-yourself-Loetkurs_0.pdf

» **Lötstation**: „STATION ZD-99 Lötstation, ZD-99, 1-Kanal, 48 W“; Reichelt, Preis 12,62 €

» **Seitenschneider**: „MAN 10701 Seitenschneider, 115 mm“; Reichelt, Preis 3,70 €

» **Streifenrasterplatine** (ein Drittel davon an der Tischkante abbrechen): „Streifenraster 710/5-50x100“; Segor, Preis 0,90 €
oder: „H25SR050 Streifenrasterplatine, Hartpapier, 50x100mm“; Reichelt, Preis 0,67 €



» **9V-Blockbatterie**: „E-Block 9 Volt Batterie 6LR“; Supermarkt, Preis 0,90 €

» **Batteriehalter**: „HALTER 1X9VS Batteriehalter für 9 Volt Block“; Reichelt, Preis 0,97 €