Bausatz-Artikel-Nr.: 161023
Version: 1.0
Stand: November 2024

Prototypenadapter WS2812-RGB-LED

CM-DL-RGB02

Bitte lesen Sie die Bau- und Bedienungsanleitung vor der Inbetriebnahme komplett und bewahren Sie sie für späteres Nachlesen auf. Wenn Sie das Gerät anderen Personen zur Nutzung überlassen, übergeben Sie auch diese Bau- und Bedienungsanleitung.

Kontakt:
Sie haben Fragen zum Produkt oder zur Bedienung, die über die Bau- und Bedienungsanleitung nicht geklärt werden konnten?
Sie haben eine Reklamation zu Ihrem Gerät?

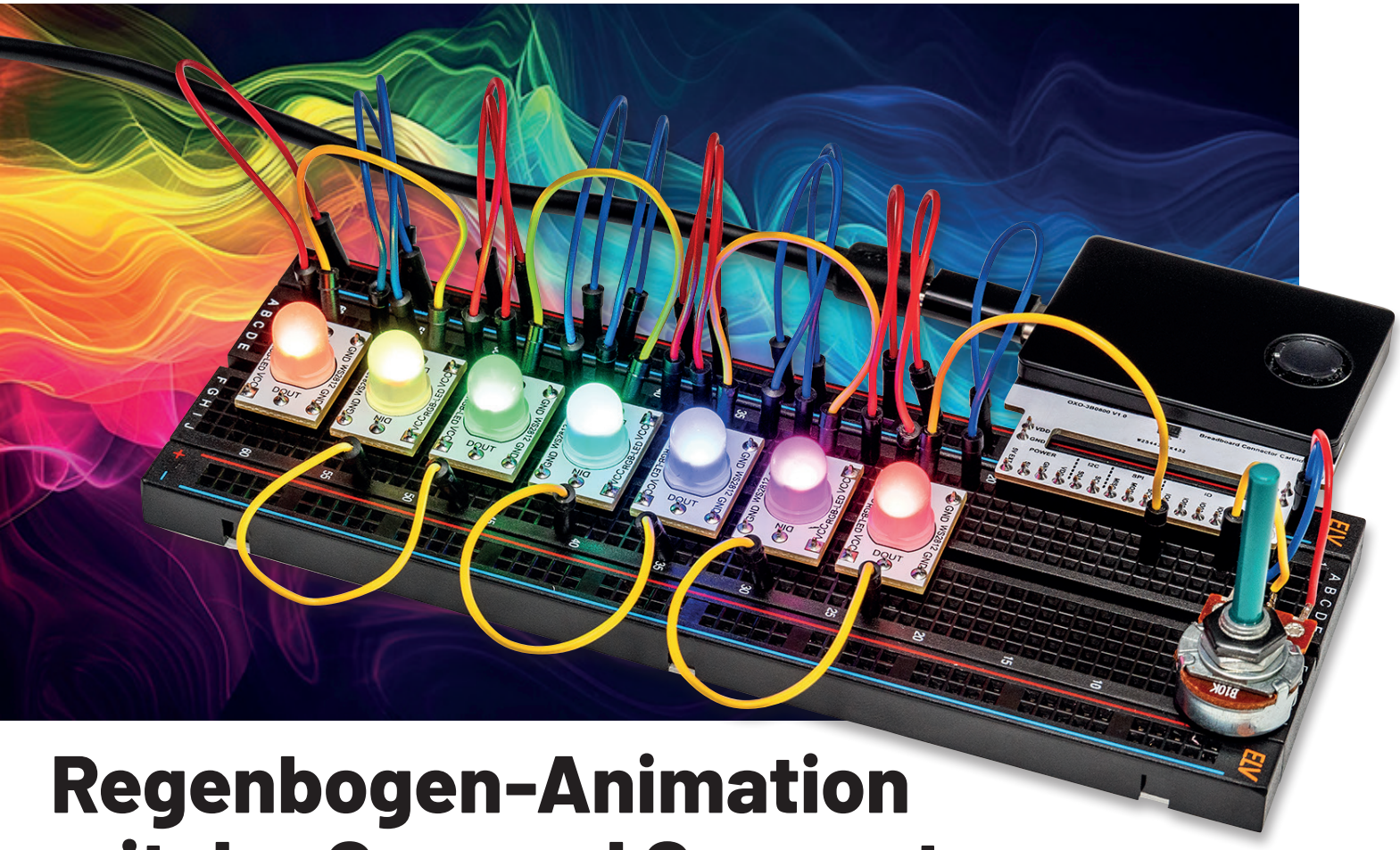
Kontaktieren Sie unser Team gerne über unsere Homepage www.elv.com im Bereich Service und Kontakt.

Häufig gestellte Fragen und aktuelle Hinweise zum Betrieb des Produkts finden Sie zudem bei der Artikelbeschreibung im ELVshop.

Reparaturservice
Für Geräte, die aus ELV Bausätzen hergestellt wurden, bieten wir unseren Kunden einen Reparaturservice an. Selbstverständlich wird Ihr Gerät so kostengünstig wie möglich instand gesetzt. Im Sinne einer schnellen Abwicklung führen wir die Reparatur sofort durch, wenn die Reparaturkosten den halben Komplettbausatzpreis nicht überschreiten. Sollte der Defekt größer sein, erhalten Sie zunächst einen unverbindlichen Kostenvoranschlag.

Bitte senden Sie Ihr Gerät an: ELV · Reparaturservice · Maiburger Straße 29–36 · 26787 Leer · Germany

ELV Elektronik AG · Maiburger Straße 29–36 · 26789 Leer · Germany
Telefon (+49)491/6008-88 · www.elv.com



Regenbogen-Animation mit der Oxocard Connect

RGB-LEDs ganz einfach ansteuern

In diesem Anwendungsbeispiel zeigen wir, wie man auf einfache Weise mit der Experimentierplattform „Oxocard Connect“ serielle LEDs vom Typ WS2812 ansteuert. Neben der Hardwarekonfiguration betrachten wir anhand von Programmbeispielen, wie man einen Programmcode erstellt und diesen auf die Oxocard Connect überträgt.

Was ist eine Oxocard Connect?

Die Oxocard Connect stellt die nächste Generation kleiner Experimentierplattformen dar. Durch den universellen Cartridge-Steckplatz können fertige oder selbst entwickelte Platinen durch einfaches Einstecken sofort zum Leben erweckt werden. Jede Karte wird mit installierten Treibern und Demoprogrammen geliefert, die beim Einstecken automatisch geladen und gestartet werden.

Die kleine Experimentierplattform verfügt über einen Dual-Core-ESP32-Chip, WiFi, 2 MB PSRAM, 8 MB Flash, USB-C, Joystick und ein 240x240-Pixel-Display.

Um Programme auf die Oxocard Connect zu übertragen, muss diese mit einem Rechner verbunden werden. Dies kann drahtgebunden über USB oder drahtlos mittels WiFi erfolgen, wie in Bild 1 zu sehen ist. Sobald man sich im Heimnetz über WiFi verbunden hat, kann man von jedem Rechner in diesem Netzwerk auf die Oxocard Connect zugreifen und Daten übertragen. Im einfachsten Fall nutzt man eine drahtgebundene Verbindung per USB-Kabel.

Die Programmierung erfolgt über einen webbasierten Compiler und Editor. Der Vorteil hierbei ist, dass man keinen Compiler auf dem Rechner installieren muss, wie sonst üblich bei Mikroprozessorboards. Ein Besuch der Website editor.nanopy.io führt direkt zum NANOPY-

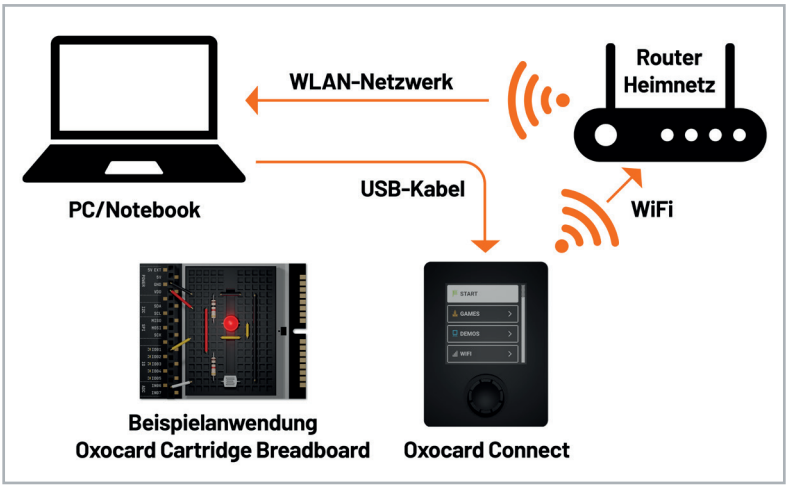


Bild 1: Oxocard Connect mit unterschiedlichen Verbindungsmöglichkeiten

Editor. Hier kann man fertige Beispiele laden oder auch selber Programme schreiben. In Bild 2 ist ein Screenshot des NANOPY-Editors zu sehen.

Hinweis: Die Programme werden in der Oxocard Connect gespeichert, auch wenn die Versorgungsspannung unterbrochen wird. Sofern die WiFi-Verbindung nicht eingerichtet wurde, kommt nach Anlegen der Versorgungsspannung eine Fehlermeldung, die ignoriert werden kann. Nach Betätigen des Buttons „Start" wird das gespeicherte Programm aktiviert.

Wir programmieren einen Regenbogen!

In diesem Artikel zeigen wir, wie man mithilfe der Oxocard RGB-LEDS ansteuern kann. Bei den LEDs handelt es sich um serielle LEDs vom Typ WS2812, für dessen Ansteuerung jeweils nur drei Leitungen benötigt werden. An VCC und GND wird die Versorgungsspannung angelegt. Am Eingang DIN der ersten WS2812-LED wird der Ausgabe-Pin der eingesetzten Oxocard – oder auch eines beliebigen anderen Arduino-Boards bzw. Mikrocontrollerboards – angeschlossen. Bei Verwendung weiterer WS2812-LEDs wird dann der Pin DOUT der vorherigen LED mit dem Pin DIN der nachfolgenden LED verbunden. So kann man mit nur drei Leitungen sehr viele LEDs gleichzeitig ansteuern, da diese alle in Reihe geschaltet sind (siehe Bild 3). Diese digitalen LEDs werden mit einem speziellen Datenprotokoll angesteuert. Für jede einzelne Farbe der LED stehen dabei 8 Bit zur Verfügung, mit denen die Helligkeit der jeweiligen Farbe bestimmt wird. Dies ergibt eine Helligkeitsabstufung von 256 (2^8) Stufen. Da die RGB-LED aus drei LEDs (rot, grün, blau) besteht, beträgt die Länge des Datenprotokolls 24 Bit (3x 8 Bit). Wer sich für detaillierte technische Daten und das Datenprotokoll interessiert, kann das Datenblatt herunterladen.

Eine Programmierung der seriellen Datenausgabe für diese WS28112-LED ist recht anspruchsvoll. Aus diesem Grund sind in NANOPY bereits fertige Ausgaberoutinen vorhanden, die einfach mit den gewünschten Helligkeitswerten aufgerufen werden können. Man muss sich also keine Gedanken über die Hardware-Programmierung machen.

Die hier aufgeführten LEDs gibt es in unterschiedlichen Gehäuseformen von SMD bis hin zur bedrahteten Version. Für die hier vorgestellte Experimentierschaltung nutzen wir einen speziellen PAD-Adapter, auf dem sich eine bedrahtete 8-mm-

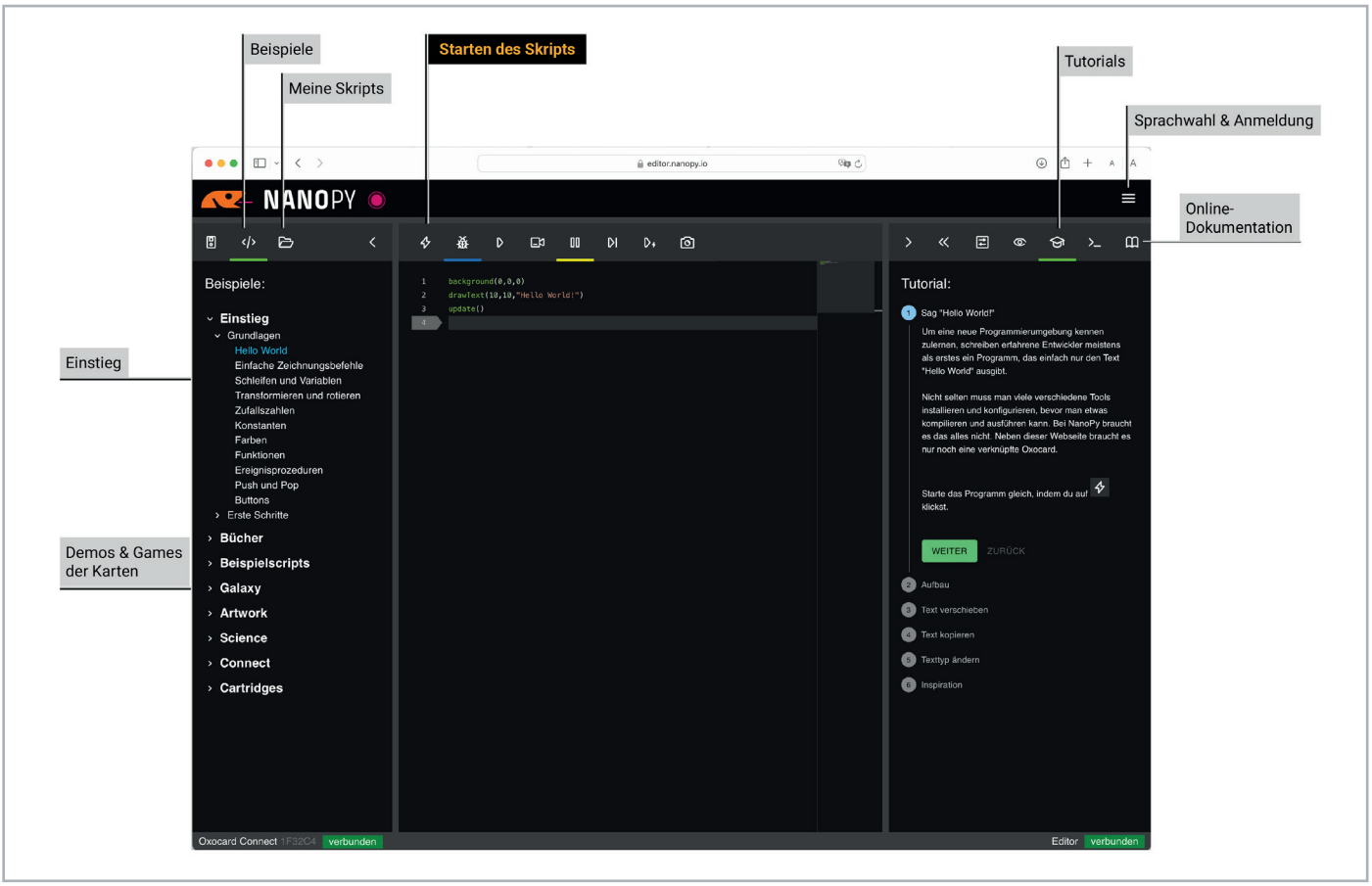


Bild 2: Screenshot vom NANOPY-Editor

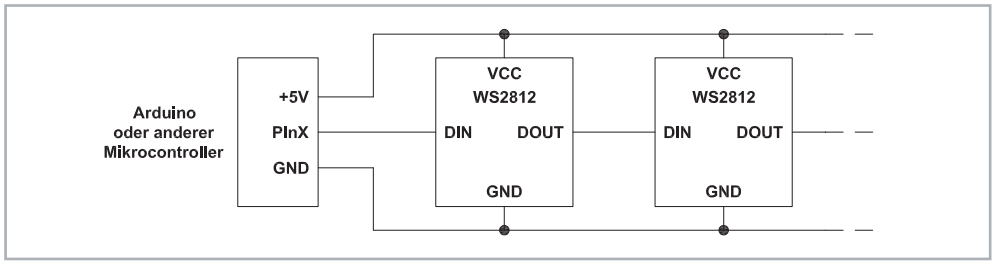


Bild 3: Schematischer Aufbau mehrerer WS2812-LEDs mit einem Mikrocontroller

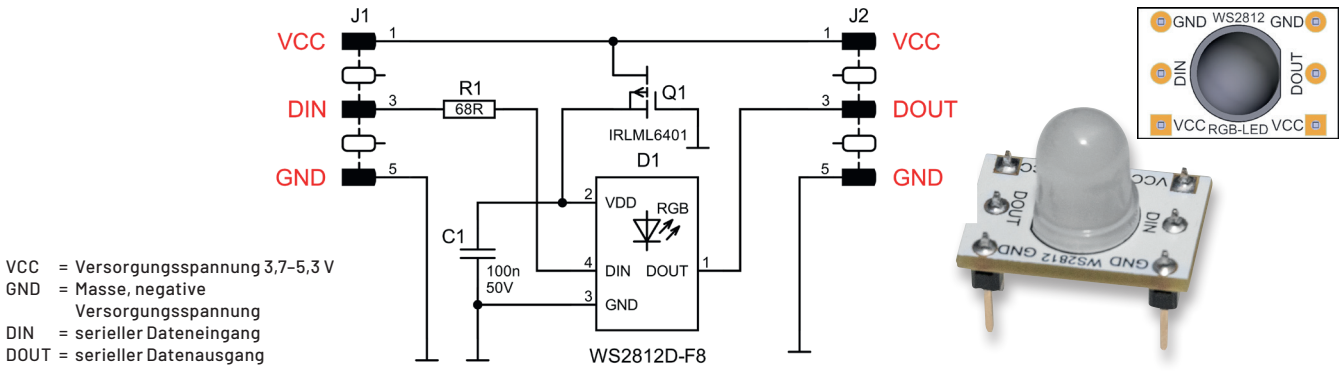


Bild 4: Schaltbild und Aufbau des PAD-RGB-LED-Adapters (CM-DL-RGB02)

LED befindet. Die Verwendung einer bedrahteten LED hat den Vorteil, dass die Farbmischung bereits im Gehäuse der LED erreicht wird. Bei einer SMD-Version müssen die Primärfarben hingegen durch einen Diffusor gemischt werden (additive Farbmischung). Diese und andere ELV PAD-Adapter basieren darauf, die eigentlichen Bauteile auf eine kleine Platine zu setzen, wodurch eine bessere Handhabung erreicht wird, da z. B. die Anschlüsse auf der Platinenoberseite gekennzeichnet sind. In Bild 4 ist das Schaltbild und der Aufbau dieses Adapters zu sehen. Im Zusammenspiel mit der Oxocard Connect und fertigen Softwarebeispielen ergibt sich eine einfach einzubauende Experimentierschaltung, mit der schöne Farbeffekte produziert werden können. Schauen wir uns im Folgenden an, was hierzu alles benötigt wird und wie man die Oxocard Connectmit den Beispielprogrammen einrichtet.

Tabelle 1

Auflistung aller benötigten Teile

Menge	Beschreibung	Artikel-Nr.
1	Oxocard Connect	253844
1	Oxocard Connector Cartridge	254229
1	ELV Steckboard, 830 Kontakte	250986
1	Steckkabel-Set, Stecker auf Stecker, 65-teilig	145145
1	PAD-RGB-LED (WS2812), 10-teilig	161023
1	Potentiometer, 10 kΩ, 4-mm-Achse	095055
1	optional: USB-Typ-C-Netzteil, 18 W	251317

Hardwareaufbau

Wie schon erwähnt, wird die gesamte Hardware auf einem Steckboard (auch Breadboard genannt) aufgebaut. Bild 5 zeigt ein Foto und einen gezeichneten Anschlussplan der Schaltung. In Tabelle 1 sind

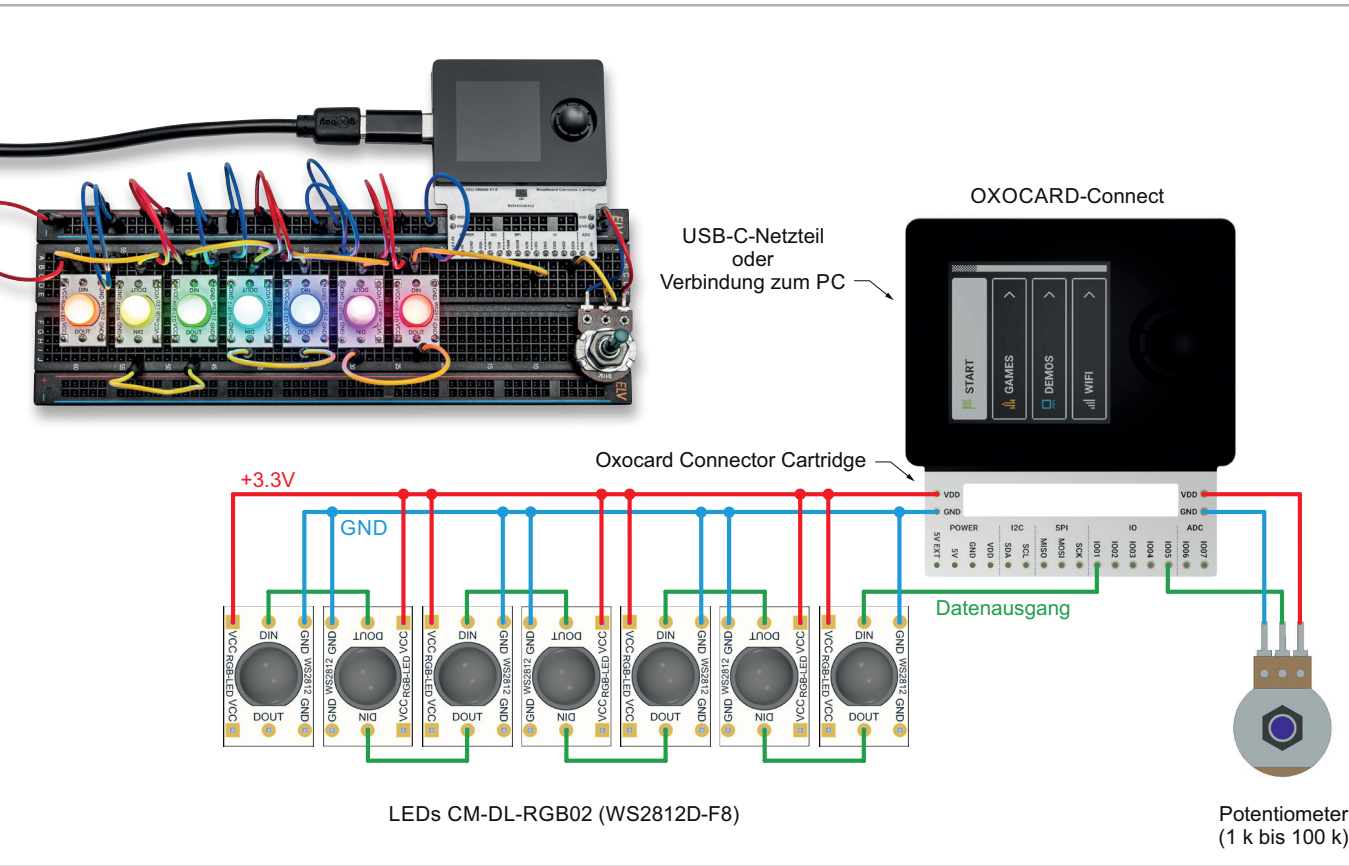


Bild 5: Foto und Anschlussplan der Schaltung

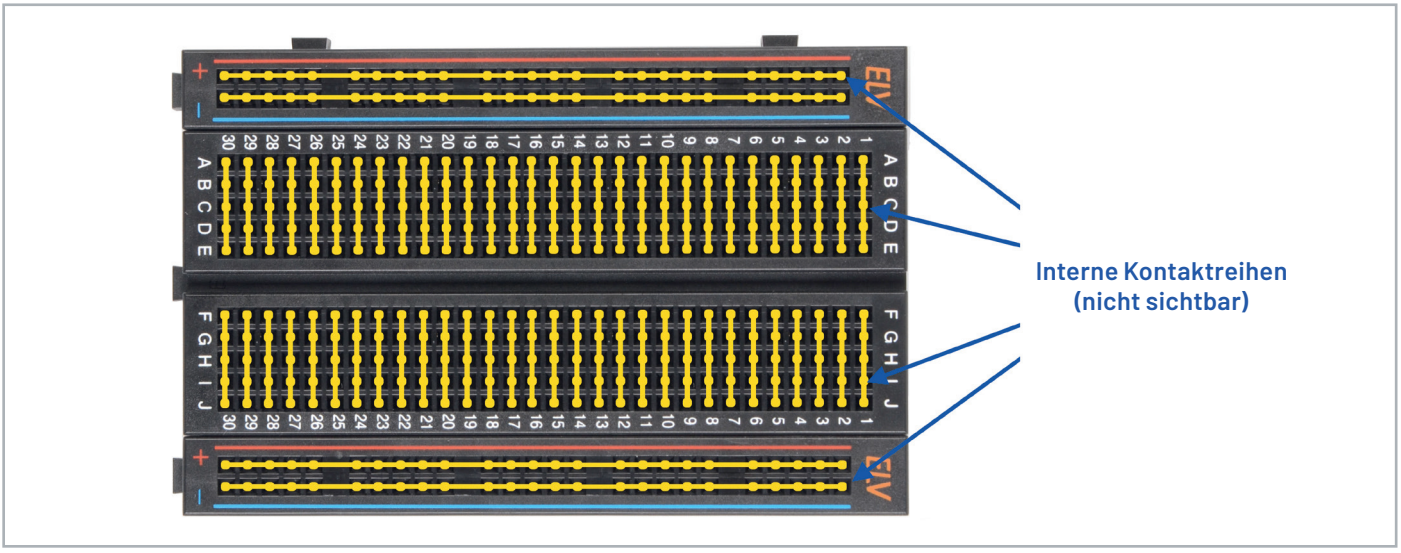


Bild 6: Kontaktierung der Kontaktreihen bei einem Steckboard

alle Bauteile aufgelistet, die wir für den Aufbau benötigen. Wer es noch nicht kennt: Ein Steckboard dient der Aufnahme von einzelnen Bauteilen. Zur besseren Orientierung sind die Kontakte des Steckboards zudem nummeriert bzw. mit Buchstaben gekennzeichnet und auf eine spezielle Art miteinander verbunden. In Bild 6 ist dargestellt (gelb markiert), wie die Kontakte im Detail intern verbunden sind. An den Rändern befinden sich sogenannte Spannungsschienen, die komplett durchverbunden sind und in der Regel für die Spannungsversorgung der gesamten Schaltung genutzt werden. Diese „Schienen" sind farblich markiert: rot für Plus (+) und blau für Minus (-).

Wichtig! Das ELV Logo und damit die Nummerierung „1" muss sich auf der rechten Seite befinden, damit die Oxocard Connect polrichtig mit den Spannungsschienen kontaktiert wird (siehe Bild 7)

Die Oxocard wird mittels eines Adapters Oxocard Connector Cartridge (Bild 8) mit dem Steckboard kontaktiert und ist als Bausatz verfügbar. Hier müssen lediglich die Stiftleisten aufgelötet werden, was auch für Anfänger kein Problem darstellen sollte. Die RGB-LED-Modulplatinen werden, wie in Bild 5 dargestellt, eingesetzt. Die Anzahl der LEDs kann frei gewählt werden, da die genaue Anzahl in den Beispielprogrammen exakt eingestellt werden kann. Beim Einsetzen sollte darauf geachtet werden, dass die LEDs immer abwechselnd um 180 Grad gedreht gesteckt werden. So kann mit einer kurzen elektrischen Leitung immer der Ausgang (DOUT) mit dem Eingang (DIN) der nächsten LED verbunden werden. Die erste LED auf dem Steckboard wird abschließend mit dem Datenausgang IO01 der Oxocard Connector Cartridge verbunden.

Die aufgezeigten elektrischen Verbindungen können entweder mit Steckbrücken oder Steckkabeln hergestellt werden. Die einfachste Handhabung erreicht man durch den Einsatz flexibler Steckbrücken. Die Versorgungsspannung für die LEDs liefert die Oxocard Connect bereits über den Anschluss VDD. Laut Spezifikation kann die WS2812 mit einer Spannung von 3,7 bis 5,3 V versorgt werden, es hat sich aber in unserem Testaufbau gezeigt, dass auch die Spannungsversorgung von 3,3 V der Oxocard selbst für den Betrieb ausreichend ist. Diese Spannung ist nur auf der oberen Spannungsschiene verfügbar (siehe Bild 7), sodass auch von hier aus die LEDs mit Spannung versorgt werden. Beim Anschluss ist auf die richtige Polung zu achten: VDD = rot markierte Spannungsschiene, GND = blau markierte Spannungsschiene.

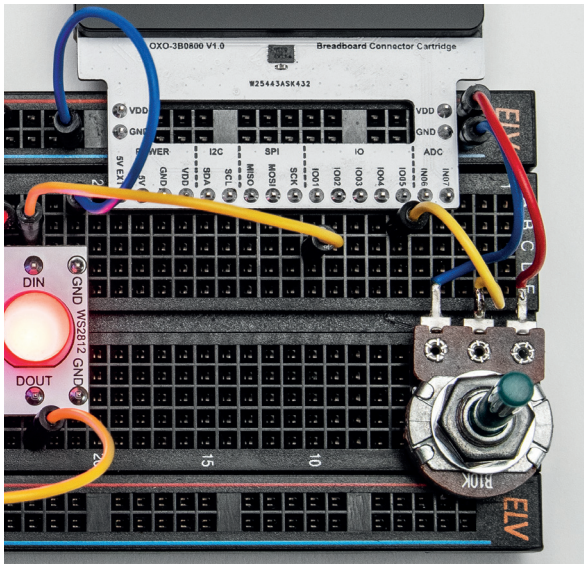


Bild 7: Die Oxocard Connect wird mittels der Oxocard Connector Cartridge mit dem Steckboard verbunden.

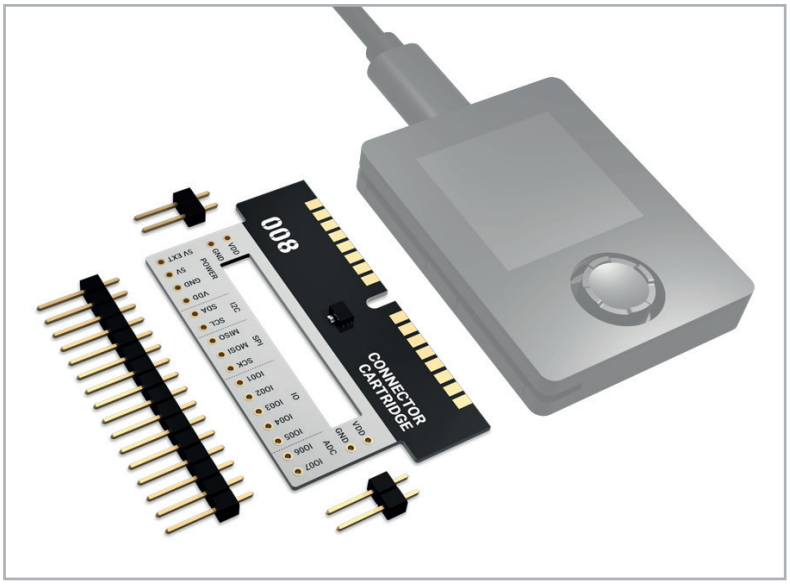


Bild 8: Oxocard Connector Cartridge als Bausatz (noch nicht aufgebaut)

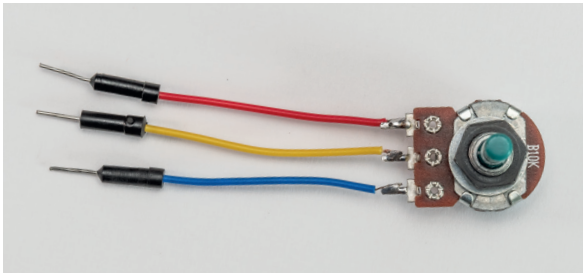


Bild 9: An das Potentiometer werden kurze Verbindungsleitungen angelötet.

Eines der folgenden Programmbeispiele verwendet unter anderem ein Potentiometer (Poti), mit dem die Helligkeit der LEDs verändert werden kann. Das Potentiometer liefert schaltungsseitig eine veränderbare Gleichspannung an den AD-Wandleranschluss IN07 der Oxocard. Für die Kontaktierung lötet man kurze Verbindungskabel an die drei Anschlüsse des Potentiometers wie in Bild 9 dargestellt. Hat man fertige Steckkabel zur Hand, können diese aufgetrennt werden, sodass ein Ende schon mit einem Stecker versehen ist. Alternativ können starre Schaltlitze verwenden werden, die ganz einfach ohne Stecker in das Steckboard eingeführt werden. Bild 7 zeigt, wie das Potentiometer (Poti) mit der Schaltung verbunden wird. Die beiden äußeren Anschlüsse des Potentiometers werden mit VDD und GND verbunden.

Die gesamte Schaltung bzw. die hier verwendete Oxocard Connect wird über USB mit Spannung versorgt. Verbindet man die Oxocard direkt über USB mit einem PC, erfolgt hierüber neben der Datenkommunikation auch die Spannungsversorgung. Andernfalls ist ein USB-Netzteil (Typ C) für den Betrieb zu verwenden.

Einrichten der Oxocard

Der Microcontroller der Oxocard ist das Herz unserer kleinen Experimentierplattform. Die Programmierung erfolgt über die bereits eingangs erwähnte kostenlose NANOPY-Entwicklungsumgebung. Wir zeigen in diesem Artikel, wie man die Karte konfiguriert und mit dem Browser verbindet. Dabei kann die Oxocard direkt über ein USB-Kabel oder drahtlos über WiFi mit dem PC verbunden verwenden. Die nachfolgende Beschreibung stellt die drahtlose Einrichtung dar.

Nach dem Anlegen der Versorgungsspannung erscheint das in Bild 10.1 dargestellte Hauptmenü auf dem Display der Oxocard. Wir wählen hier den Menüpunkt WIFI. Nun stehen zwei Möglichkeiten zur Verfügung, wie die WiFi-Daten hinterlegt werden können. Wer kein Smartphone zur Hand hat, kann die Daten über CARD direkt auf der Karte eingeben, ähnlich wie man es von Spielekonsolen gewohnt ist (Bild 10.2). Es erscheint eine Übersicht der verfügbaren WLAN-Netze (Bild 10.3). Über eine Tastatur wird anschließend das Passwort zur ausgewählten SSID angegeben (Bild 10.4).

Alternativ kann man die Daten auch via Smartphone eintippen. Dazu wählen wir PHONE. Sobald die Anmeldung erfolgt ist, kann man im Browser mit dem Programmieren beginnen.

Nach dem Aufruf des NANOPY-Editors erscheint das in Bild 11 dargestellte Fenster. Das Oxocard Connect muss nun mit dem Editor verknüpft (paired) werden. Wählt man an der Oxocard den Menüpunkt „Pairing" aus, erscheint auf dem Display ein 3-stelliger Code, den man im Fenster am PC eingibt. Bei einer reinen USB-Verbindung wählt man „Verbinden via USB" aus. Sollte kein entsprechender Treiber für Windows zur Verfügung stehen, bekommt man Hinweise zur Installation angezeigt. Wir empfehlen die Webbrowser Chrome und Edge für die weitere Programmierung.

Nachdem diese Schritte erfolgt sind, erscheint das eigentliche Hauptfenster des NANOPY-Editors (Bild 2).

Im Hauptfenster können wir das Beispiel „Einführung" „Hello World" auswählen und finden ein Programm vor, das aus drei Zeilen besteht (Bild 12). Nach einem Klick auf den Blitz-Button in der Symbolleiste wird der Code auf die verbundene Oxocard Connect übertragen und automatisch ausgeführt. Es erscheint der Text „Hello World" auf dem Bildschirm des Geräts.

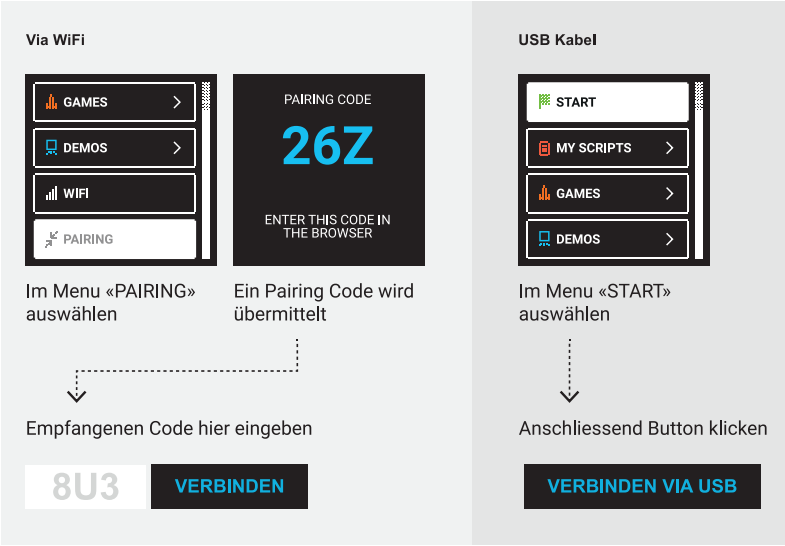
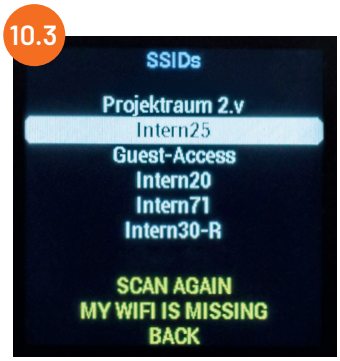


Bild 11: Screenshot des NANOPY-Editors



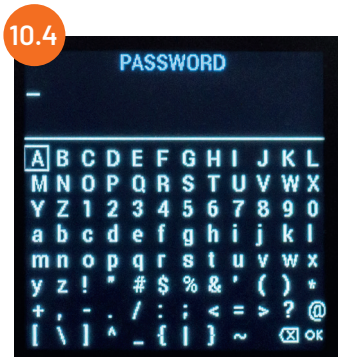


Bild 10: Einrichtung der WiFi-Verbindung

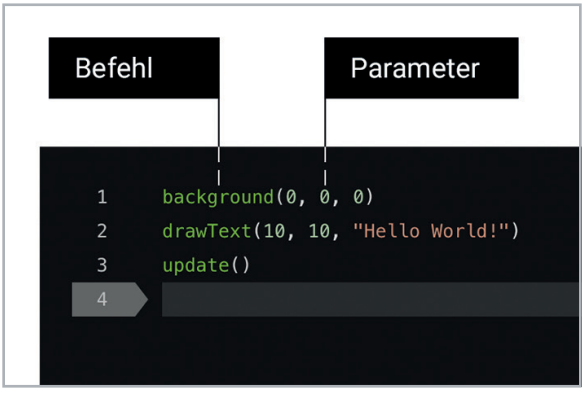


Bild 12: Befehle und Parameter

Ein Programm wie „Hello World" ist eine Aneinanderreihung von Einzelanweisungen, die schrittweise vom Computer ausgeführt werden. Die Befehle sind in der Regel in Englisch geschrieben, sodass man sich auch als Einsteiger ungefähr vorstellen kann, was der Befehl bewirkt. In Klammern stehen jeweils Zusatzinformationen, meistens in Form von Zahlen, die Parameter genannt werden.

Kurze Einführung in die Welt der Programmierung

Da Microcontroller heute bereits für wenige Cents erhältlich sind, werden diese auch in trivialen Elektronikschaltungen eingebaut – ohne Programmierung läuft aber gar nichts. Wir geben hier einen kurzen Überblick in die wichtigsten Aspekte, bevor wir mit einem konkreten Projekt starten.

Die meisten Programmiersprachen basieren auf vier Grundkonzepten: Funktionen, Variablen, Schleifen und Bedingungen.

Die Funktionen sind die Essenz jedes Systems. Die Funktionen bestimmen, was wir mit dem Gerät machen können. Die Liste der verfügbaren Funktionen ist je System unterschiedlich. Diese sind in der Regel gut dokumentiert und Teil der Entwicklungsumgebung.

In der Mathematik ist eine **Variable** ein Platzhalter für eine Zahl. In der Computerwelt ist eine Variable ein Name für einen Speicherplatz, in dem eine Zahl oder auch was ganz anderes stehen kann.

Eine **Schleife** wiederholt eine Anweisung mehrere Male. Es gibt endlose Schleifen, Zählschleifen und Bedingungsschleifen.

Eine **Bedingung** ist eine Weiche, bei der man den Programmablauf umlenken kann. Dafür nutzen wir die Anweisung „if" oder „while" aus dem nächsten Abschnitt.

Das Beispiel im Bild 13 zeigt, wie die wichtigen Konzepte im Programm umgesetzt werden. Das Ergebnis dieser Programmierzeilen ist in Bild 14 zu sehen (blinkender Kreis).

Die Variable „on" wird in der Zeile 1 mit einem Wert initialisiert. „while true" bedeutet, dass sich alles endlos wiederholt, was eingerückt auf den nachfolgenden Zeilen 4–14 folgt. Die grün dargestellten Worte sind Befehle oder Funktionen, die dem Microcontroller Einzelanweisungen geben. Mit „if" kann man eine Bedingung formulieren. Wenn die Bedingung erfüllt ist, wird der eingerückte Block ausgeführt, andernfalls wird der „else"-Block ausgeführt.

Obwohl Programmiersprachen ihren Ursprung in der Mathematik haben, brauchen wir beim Programmieren selbst ganz wenig Mathe-Wissen. In den meisten Fällen genügt das in der Grundschule gelernte Einmaleins, auch wenn die knappen Formulierungen und die häufig verwendeten Sonderzeichen es für Einsteiger etwas schwieriger ma-

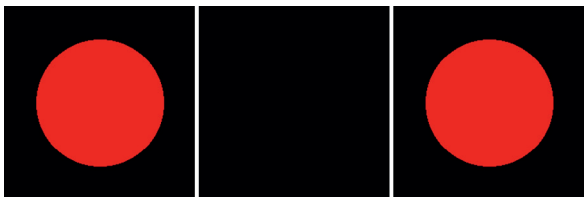


Bild 14: Das Beispielprogramm lässt einen roten Kreis aufblinken.

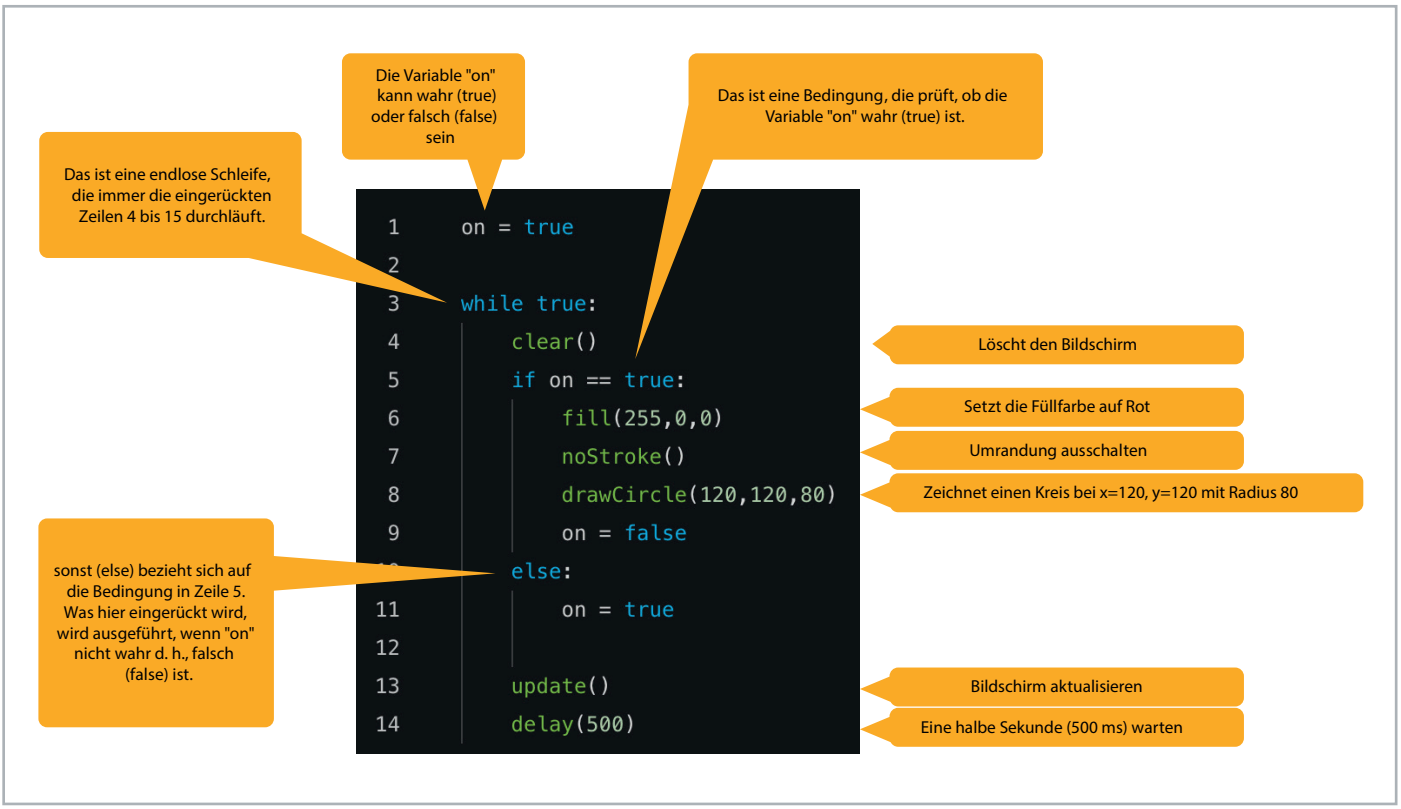


Bild 13: Programmbeispiel

chen. Mit etwas Geduld ist der Einstieg dennoch kein Hexenwerk, und die Programmiersprache ist leicht zu erlernen. Für den Anfang kann man aus Hunderten fertiger Programmbeispiele aus der NANOPY-Umgebung wählen, die sofort ausgeführt werden können. Wer also wenig Erfahrung mit Programmieren hat, lernt hier am effektivsten. Auch das LED-Beispiel aus diesem Artikel muss nicht selbst entwickelt werden. Je nach eigenen Fähigkeiten kann es abgetippt oder einfach online heruntergeladen werden, dabei sind gegebenenfalls kleine Anpassungen vorzunehmen.

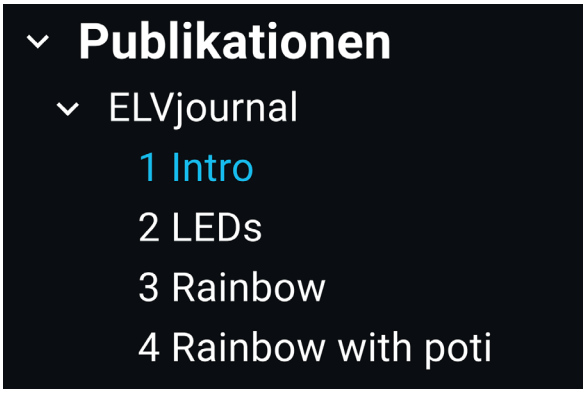


Bild 15: Hier befinden sich die fertigen Programmbeispiele.

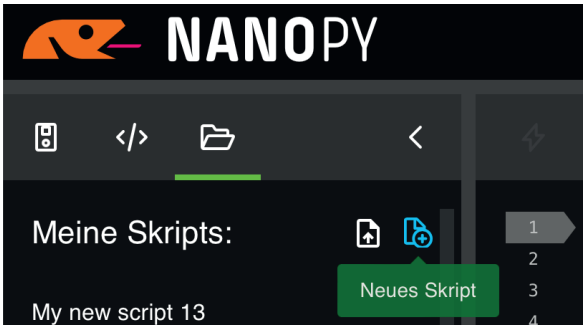


Bild 16: Neues Skript auswählen

Programmieren

Schritt für Schritt hauchen wir jetzt der Elektronik (unserer aufgebauten Experimentierschaltung) mit etwas Programmierhandwerk Leben ein. Die fertigen Programmbeispiele sind im NANOPY-Editor unter Publikationen → ELVjournal zu finden (siehe Bild 15).

Zuerst starten wir mit einem kleinen Skript, um die grundsätzliche Funktionsweise der LEDs zu testen. In der NANOPY-Umgebung erstellen wir hierzu ein neues Skript (Bild 16). Der dabei automatisch generierte Demo-Code wird nicht benötigt. Durch Markieren und Löschen aller Zeilen erhalten wir einen leeren Skript-Editor (Bild 17).

Hinweis: Das fertige Beispiel findet man zudem im NANOPY-Editor unter „ELVjournal" → „1 Intro". Nachfolgend wird nun der in Bild 18 dargestellte Code eingegeben und das Programm gestartet.

Die Funktion „initDigitalLeds" initialisiert zunächst den LED-Treiber. Wir definieren hier, dass unsere LEDs am Pin 01 angeschlossen sind („C_Pin_01"). Der zweite Parameter gibt die Anzahl der angeschlossenen LEDs an. In unserem Fall geben wir hier die Zahl 7 ein, da wir in unserer Schaltung sieben LEDs verwenden. Der dritte Parameter definiert den Typ, den wir verwenden, also „C_LED_Typ_WS2812B". Mit der Funktion „setDigitalLed" setzen wir nun für jede LED einen Farbwert. Hierzu sind vier Parameter notwendig. Zuerst bestimmen wir, welche Position der Liste verändert werden soll. Die LEDs sind hierbei aufsteigend durchnummeriert, wobei wir mit 0 für die erste LED starten und mit 1 für die zweite LED fortfahren. Die drei weiteren Zahlen stehen für den Rot-, Grün- und Blauanteil der Farbe. Jede Zahl kann einen Wert von 0 bis 255 annehmen. Eine 0 bedeutet, dass die Farbe nicht eingeschaltet ist, bei 255 haben wir die volle Ausleuchtung. Durch die Kombination der drei Farbanteile lassen sich diverse Farben mischen (Bild 19). Das System nennt sich RGB und ist in Grafikprogrammen und auch im Web sehr verbreitet.

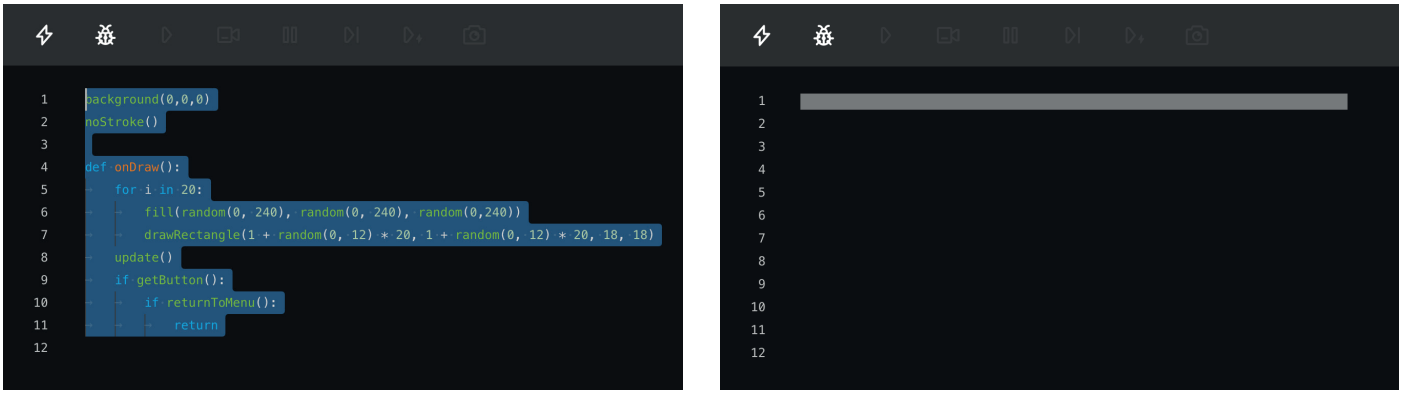


Bild 17: Skript markieren und löschen

Bild 18: Beispielcode schreiben und anschließend Code ausführen

Mit „applyDigitalLeds“ werden die Farben an die LEDs übermittelt. Erkennbar ist zudem, dass die Programmzeilen 3-10 beinahe identisch sind. Für jede LED wurde eine Zeile definiert, in der eine bestimmte Farbe zugewiesen wird. Um Programmcodes zu sparen, kann dieses auch kürzer erfolgen, indem eine Schleife definiert wird.

Im folgenden Beispiel (Bild 20) erzeugen wir eine Schleife, die von 0 bis 7 zählt und bei jedem Durchgang die Zeilen 4–7 durchläuft. Innerhalb dieser Schleife setzen wir eine zufällige Farbe und geben diese an die LED mit der Schleifennummer „i“ aus.

Wenn wir das Skript starten, leuchtet jede LED in einer anderen Farbe. Erstmalig sehen wir des Weiteren in diesem Beispiel sogenannte Variablen. In diesem Fall kann man sich die Variable wie folgt vorstellen: Wir merken uns die aktuelle Runde eines Rennens auf einem Papier. Die Runde heißt bei uns „i“. Im ersten Durchgang trägt „i“ den Wert 0, im zweiten 1 etc. Die Variablen „red“, „green“ und „blue“ speichern den Rot-, Grün- und Blauanteil der LED. Wir setzen diese mit einer Funktion, die „random“ heißt und einen zufälligen Wert generiert. Wir geben als Parameter einen beliebigen Wertebereich an und erhalten eine zufällige Zahl aus dem Bereich. In Zeile 7 finden wir wieder die bereits bekannte Funktion. Der erste Parameter ist „i“, dann folgen die zufällig bestimmten Farbanteile für Rot, Grün und Blau.

Schleifen und Variablen sind abstrakte Konzepte, die zu Beginn etwas verwirrend erscheinen mögen. Dank des Debuggers können wir bei diesem Code jedoch hinter die Kulissen schauen und entdecken, was die Oxocard im Detail macht. Der Debugger dient der Analyse und Überwachung von Programmen. Hiermit lassen sich Programme sehr stark verlangsamt abspielen, sodass ein genaueres Beobachten ermöglicht wird. Wir starten den Debugger mit „Code debuggen“ (Bild 21). Das Programm ist jetzt gestartet und stoppt auf der ersten Programmzeile, die zudem farblich markiert wird. Mit „Schritt“ lässt sich nun das Programm Zeile für Zeile durchlaufen. Nach dreimaligem Klicken erhalten wir die Anzeige wie in Bild 22. In der rechts eingeblendeten Variablen-Anzeige sehen wir die Variablen „i“, „red“, „green“ und „blue“. Wenn wir die Zeile 4 durchlaufen, wird „red“ einen Wert zwischen 0 und 255 erhalten. Im nächsten Schritt wird „green“ gesetzt etc. Jeder Schritt führt eine Veränderung aus, die wir direkt durch den Debugger beobachten können. Das Programm lässt sich auch über den

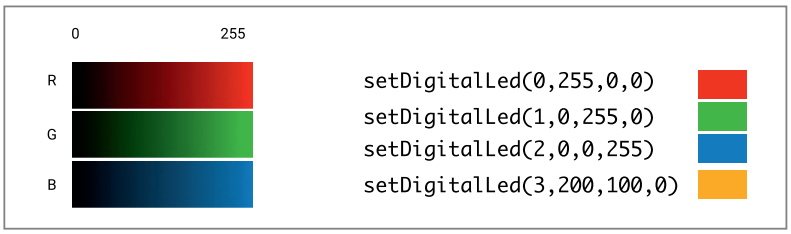


Bild 19: Durch die Kombination der drei Farbanteile lassen sich diverse Farben mischen.

```
1   initDigitalLeds(C_PIN_01, 7, C_LED_TYPE_WS2812B)
2
3   for i in [0..6]:
4       red = random(0,256)
5       green = random(0,256)
6       blue = random(0,256)
7       setDigitalLed(i,red,green,blue)
8
9   applyDigitalLeds()
```

Bild 20: So lässt sich der Code mittels einer Schleife verkürzen.

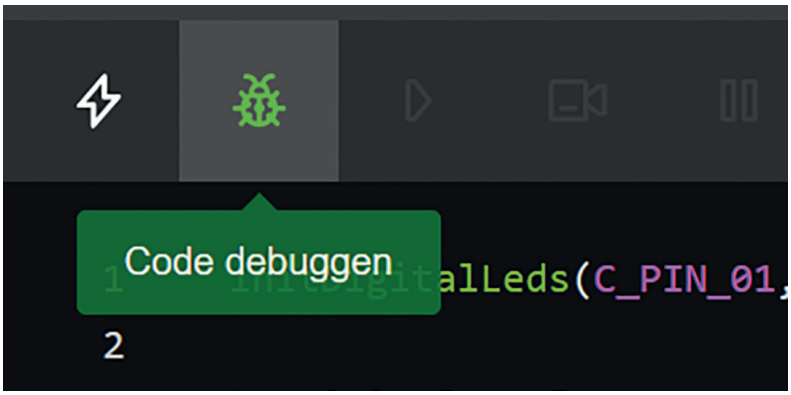


Bild 21: Start des Debuggers

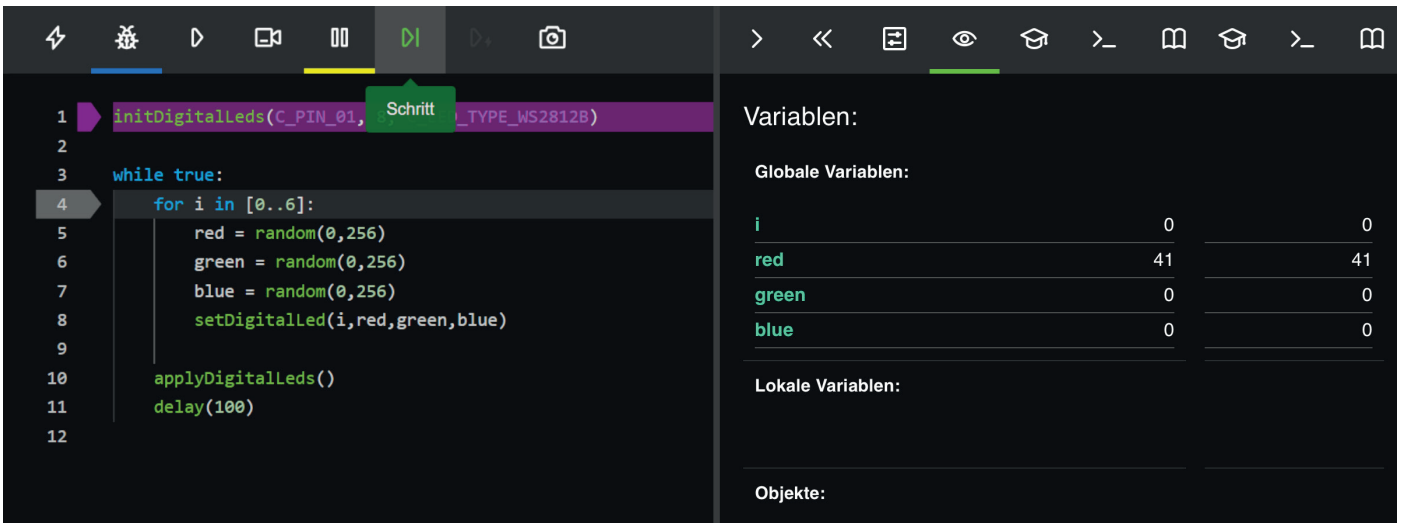


Bild 22: Ausgabe des Debuggers

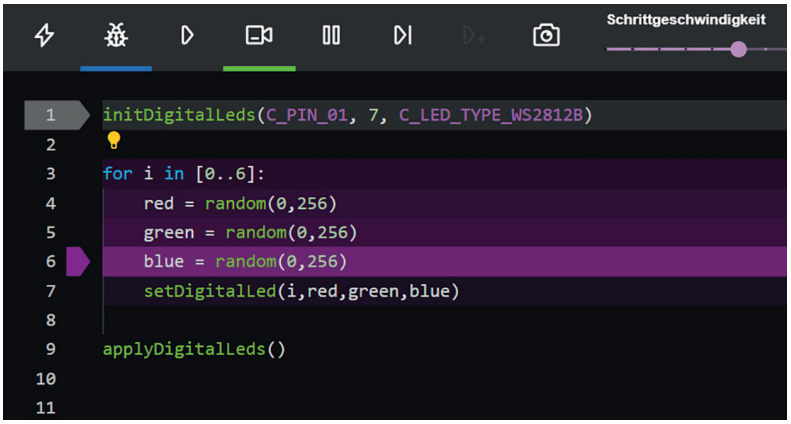


Bild 23: Ausgabe des Debuggers mit dem Observier-Button

```
1   initDigitalLeds(C_PIN_01, 7, C_LED_TYPE_WS2812B)
2
3   while true:
4       for i in [0..6]:
5           red = random(0,256)
6           green = random(0,256)
7           blue = random(0,256)
8           setDigitalLed(i,red,green,blue)
9       applyDigitalLeds()
10      delay(100)
```

Bild 24: Wir ergänzen den Code um zwei Zeilen.

Observier-Button (Bild 23) ausführen. In diesem Modus läuft das Programm stark verlangsamt ab. Die letzten Zeilen werden farblich hervorgehoben. So sieht man deutlich, wie die Schleife der Zeile 3 nun für jede Zahl zwischen 0 und 7 den Codeblock 4-7 durchläuft.

Im nächsten Schritt ergänzen wir das Programm um zwei weitere Zeilen (Bild 24). In Zeile 3 fügen wir „while true" ein und rücken den ganzen Rest des Programms mit der Tabulator-Taste eine Stufe ein.

In Zeile 10 ergänzen wir den Befehl „delay(100)". Sofern alles richtig eingesetzt wurde, kann eine tolle Lichterketten-Animation beobachtet werden. Die Endlosschleife wird durch den Befehl „while true" definiert, dabei wird alles, was eingerückt ist, immer wieder und endlos weiter ausgeführt. Das Skript setzt bei allen LEDs eine zufällige Farbe und fängt anschließend wieder von vorne an. Damit die Schleife nicht zu schnell abläuft, wird nach jedem Durchgang eine kurze Pause mit dem Befehl „delay(100)" eingefügt, der den NANOPY 0,1 Sekunden warten lässt. Ein fertiges Beispiel ist hier zu finden: NANOPY-Editor unter „ELVjournal → 2 LEDs" (siehe Bild 15).

```
1   initDigitalLeds(C_PIN_01, 7, C_LED_TYPE_WS2812)
2   j:byte
3
4   while true:
5       for i in [0..6]:
6           c:color
7           c.hsv(j+(i*10),255,v)
8           setDigitalLed(i, c.r,c.g,c.b)
9       j++
10      applyDigitalLeds()
```

Bild 26: So wird der Regenbogen-Algorithmus im Programmcode eingefügt.

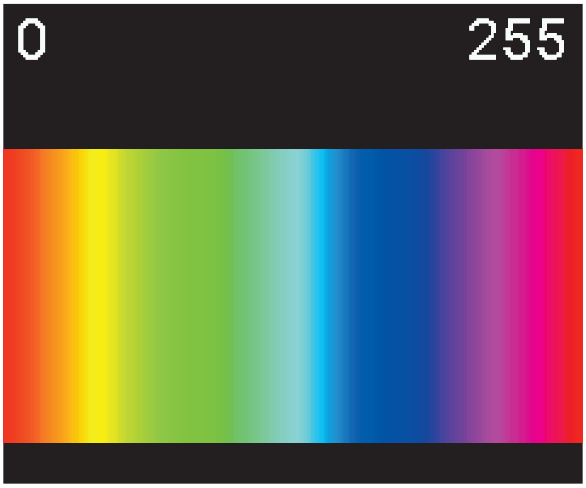


Bild 25: HUE-Wert, der die Grundfarbe bestimmt

Regenbogen-Animation

Wir verwenden jetzt das erarbeitete Beispiel, tauschen aber die zufälligen Farben durch einen Farbverlauf aus. Hierzu braucht es ein anderes Farbsystem und ein paar kleine Anpassungen. Es gibt unterschiedliche Arten, wie man Farben kategorisieren kann. In der analogen Welt geben wir den Farben Namen, bei Tausenden von Farbvarianten braucht es andere Kategorisierungsformen. Eine davon haben wir bereits mit dem RGB-System kennengelernt.

Für unseren Regenbogen verwenden wir das HSV-System. Bei diesem werden die Farben wieder mit drei Zahlen definiert. Der Hue-Wert (H) bestimmt die Grundfarbe. Beim Original haben wir hier einen Wertebereich von 0–360 Grad, bei der Oxocard haben wir nur 0–255, damit es einfacher wird. Der zweite Wert betrifft die Sättigung (S für Saturation). Dieser Wert kann wieder 0–255 annehmen und bestimmt die Stärke der Farbe. Mit dem dritten Wert „V" (Value) kann man die Helligkeit der Farbe festlegen. In Kombination lassen sich wieder unzählige Farben realisieren (siehe Bild 25).

Wir passen unser bestehendes Programm (siehe NANOPY-Editor unter „ELV Journal → 3 Rainbow") etwas an. In erster Linie ersetzen wir die zufälligen Farben durch den Regenbogen-Algorithmus (Bild 26) und ergänzen hierzu die Zeile 2. Die „while"- und „for"-Schleifen bleiben identisch, jedoch ersetzen wir den dazwischen gelagerten Code durch drei neue Zeilen. In Zeile 9 erhöhen wir die Zahl in der Variablen „j" um eins. Es ist zu beachten, dass Zeile 9 nur einfach eingerückt ist, sodass „j" erst dann erhöht wird, wenn alle LEDs einmal gesetzt sind. Sie ist dafür verantwortlich, dass sich die Farben bei jedem Durchgang etwas verschieben.

Wir verwenden hier den Befehl „color", um einen Farbwert zu speichern. Hierzu besitzt die Klasse die internen Variablen, die „r", „g" und „b" heißen und jeweils den Farbwert der Grundfarben enthält. Man

nutzt diese wie normale Variablen, wobei man den Variablennamen voranstellt und mit einem Punkt trennt:

```
c:color
c.r = 100
c.g = 100
c.h = 100
```

Interessant ist, dass diese Variablen auch Funktionen enthalten können. In unserem Fall gibt es die Funktion hsv(h,s,v) aus Zeile 7, mit der wir die Farbe über den HSV/HSB-Farbraum angeben können.

Der Regenbogeneffekt kommt jetzt wie folgt zustande: Die Variable „j" wird in der Schleife immer um eins erhöht. Da die Variable vom Typ „byte" ist, kann sie maximal den Wert 255 speichern und fängt dann wieder bei 0 an. Diese Iteration kann man gut mit dem Debugger verfolgen.

In der „for"-Schleife setzen wir die Werte für die fünf LEDs. Beim ersten Durchgang ist die Schleifenvariable 0, der Farbwert der ersten LED ist also gemäß Formel j+(i*10) = 0 + 0 * 10 = 0. Beim ersten Durchgang, d. h., wenn j = 0 ist, werden in den LEDs folgende Farben gesetzt:

j	i	j+(i*10)
0	0	0
0	1	10
0	2	20
0	3	30
0	4	40

Wir erhöhen dann „j" um eins. In der zweiten Runde werden die Farben daher so gesetzt:

j	i	j+(i*10)
1	0	1
1	1	11
1	2	21
1	3	31
1	4	41

Jede der fünf Farben hat einen Abstand von 10. Durch die äußere Schleife „j" werden die Farben langsam durchlaufen, was zu einem schönen sanften Effekt führt. Der Wert 10 wurde hierzu empirisch erhoben. Natürlich können auch andere Werte zum Experimentieren verwendet werden. So ist es auch denkbar, „j" über 5 zu erhöhen oder z. B. auch nach 100 wieder von vorne zu beginnen. Anstelle des Farbwerts kann man auch den zweiten oder dritten Parameter der „hsv"-Funktion mit der Formel verändern, was andere interessante Effekte erlaubt.

Helligkeit ändern mit dem Potentiometer

Das zuvor bereits eingebaute Potentiometer unserer Schaltung liefert einen analogen Spannungswert. In der Schaltung (Bild 5) erkennen wir, dass wir den Pin IN07 als Eingang geschaltet haben. Dieser Pin kann einen analogen Wert wie eine Spannung in eine Zahl umwanden. Dieses Verfahren nennt sich ADC und wird von einem Analog-Digital-Converter durchgeführt. Der Befehl zum Auslesen dieses Analogeingangs heißt „readADC". Als ersten Parameter

```
1   initDigitalLeds(C_PIN_01, 7, C_LED_TYPE_WS2812)
2   j:byte
3
4   while true:
5       for i in [0..6]:
6           c:color
7           v = readADC(C_PIN_05, 100) / 16
8           c.hsv(j+(i*10),255,v)
9           setDigitalLed(i, c.r,c.g,c.b)
10      j++
11      applyDigitalLeds()
```

Bild 27: So sieht der fertige Programmcode aus.

führen wir dabei auf, an welchem Pin das Potentiometer angeschlossen ist. Der zweite Parameter bestimmt, ob die ADC-Funktionen einen Mittelwert aus einer Anzahl bestimmter Werte errechnen sollen. Da analoge Messungen immer gewissen Schwankungen unterliegen, nehmen wir 100 Werte auf und ermitteln davon den Durchschnitt.

Nun ergänzen wir unser Programm erneut, diesmal in Zeile 7. Hierzu erzeugen wir eine neue Programmzeile nach Zeile 6 und fügen die ADC-Routine ein. Dabei wird in den Übergabeparametern festgelegt, dass der Analogwert über Pin I005 eingelesen und über 100 Messwerte gemittelt wird. Der letzte Parameter beschreibt eine Abtastrate von 16 Bit. In der nachfolgenden Zeile setzen wir den Variablen-Wert „v" nun in die Funktion „hsv" ein (Bild 27). Wenn wir das Programm starten, können wir mit dem Poti die Helligkeit der Animation einstellen. Der Beispielcode findet sich hier: NANOPY-Editor unter „ELV Journal → 4 Rainbow with poti".

Zusammenfassung

In diesem Artikel haben wir die Grundkonzepte des Programmierens kennengelernt. Vor allem Einsteiger profitieren von fertigen Programmen, die mit dem Debugger beobachtet und analysiert werden können.

Mit den gewonnenen Erkenntnissen konnte durch einfachste Programmieranpassungen mit einer einfachen elektronischen Schaltung mit nur wenigen Zeilen Programmcode ein beeindruckender Lichteffekt erzielt werden. Programmieren mag zu Beginn etwas ungewohnt erscheinen, ist aber nicht besonders schwierig. Es bietet aber vor allem einen großen Vorteil: einen unglaublich leistungsfähigen Werkzeugkasten, der uns weitere kreative Freiheitsgrade gibt – schließlich haben wir nur etwas an der Oberfläche gekratzt.

In der Community gibt es unzählige, weitaus komplexere Beispiele, die Wetterdaten aus dem Internet holen, ChatGPT Fragen stellen oder auch kreative Games auf der Oxocard ermöglichen. Mit der Kombination aus leistungsfähiger Hardware, einer einfachen Scripting-Sprache und der Verschmelzung mit Elektronik sind uns kaum noch Grenzen gesetzt, was zu vielen weiteren spannenden Projekten einlädt. ELV

Hinweis zu den vorbestückten Bausatz-Leiterplatten

Sehr geehrter Kunde,

das Gesetz über das Inverkehrbringen, die Rücknahme und die umweltverträgliche Entsorgung von Elektro- und Elektronikgeräten (ElektroG) verbietet (abgesehen von wenigen Ausnahmen) seit dem 1. Juli 2006 u. a. die Verwendung von Blei und bleihaltigen Stoffen mit mehr als 0,1 Gewichtsprozent Blei in der Elektro- und Elektronikproduktion.

Die ELV Produktion wurde daher auf bleifreie Lötzinn-Legierungen umgestellt, und sämtliche vorbestückte Leiterplatten sind bleifrei verlötet.

Bleihaltige Lote dürfen im Privatbereich zwar weiterhin verwendet werden, jedoch kann das Mischen von bleifreien und bleihaltigen Loten auf einer Leiterplatte zu Problemen führen, wenn diese im direkten Kontakt zueinander stehen. Der Schmelzpunkt an der Übergangsstelle kann sich verringern, wenn niedrig schmelzende Metalle wie Blei oder Wismut mit bleifreiem Lot vermischt werden. Das unterschiedliche Erstarren kann zum Abheben von Leiterbahnen (Lift-off-Effekt) führen. Des Weiteren kann der Schmelzpunkt dann an der Übergangsstelle unterhalb des Schmelzpunkts von verbleitem Lötzinn liegen. Insbesondere beim Verlöten von Leistungsbauelementen mit hoher Temperatur ist dies zu beachten.

Wir empfehlen daher beim Aufbau von Bausätzen den Einsatz von bleifreien Loten. ELV

WichtigerHinweise zum ESD-Schutz

Das Produkt enthält elektrostatisch gefährdete Bauelemente, die durch unsachgemäße Behandlung beschädigt werden können. Sie müssen beim Umgang mit den Komponenten elektrostatisch entladen sein!

Wichtiger Hinweis

Keine Haftung bei Sach-/Personenschäden, die durch unsachgemäße Handhabung oder Nichtbeachten der Gefahrenhinweise verursacht werden. In solchen Fällen erlischt der Gewährleistungsanspruch! Keine Haftung für Folgeschäden!

Nur für Personen/Kinder ab 14 Jahren bzw. für Kinder unter 14 Jahren nur für Ausbildungszwecke unter Aufsicht eines erwachsenen Ausbilders bestimmt.

Entsorgungshinweis

Dieses Zeichen bedeutet, dass das Gerät nicht mit dem Hausmüll, der Restmülltonne oder der gelben Tonne bzw. dem gelben Sack entsorgt werden darf.

Sie sind verpflichtet, zum Schutz der Gesundheit und der Umwelt das Produkt und alle im Lieferumfang enthaltenen Elektronikteile zur ordnungsgemäßen Entsorgung bei einer kommunalen Sammelstelle für Elektro- und Elektronik-Altgeräte abzugeben. Auch Vertreiber von Elektro- und Elektronikgeräten sind zur unentgeltlichen Rücknahme von Altgeräten verpflichtet.
Durch die getrennte Erfassung leisten Sie einen wertvollen Beitrag zur Wiederverwendung, zum Recycling und zu anderen Formen der Verwertung von Altgeräten.

Wir machen ausdrücklich darauf aufmerksam, dass Sie als Endnutzer eigenverantwortlich für die Löschung personenbezogener Daten auf dem zu entsorgenden Elektro- und Elektronik-Altgerät sind.

Bevollmächtigter des Herstellers:
ELV Elektronik AG · Maiburger Straße 29–36 · 26789 Leer · Germany