Cyclone – Neuauflage des Spieleklassikers

Gastbeitrag – ein Projekt von Natascha Gebhardt, Simon Flamm, Marcel Gosch und Dr. Juliane König-Birk

Das Cyclone-Spiel ist ein beliebtes Automatenspiel aus Arcade-Spielhallen. Ziel des Spiels ist es, ein zirkulierendes Licht im richtigen Moment zu stoppen. Je nach Timing wird ein ganzer Treffer oder auch nur ein halber Treffer erreicht. **Das Cyclone trackt dabei automatisch den aktuellen Highscore.**

Spielablauf

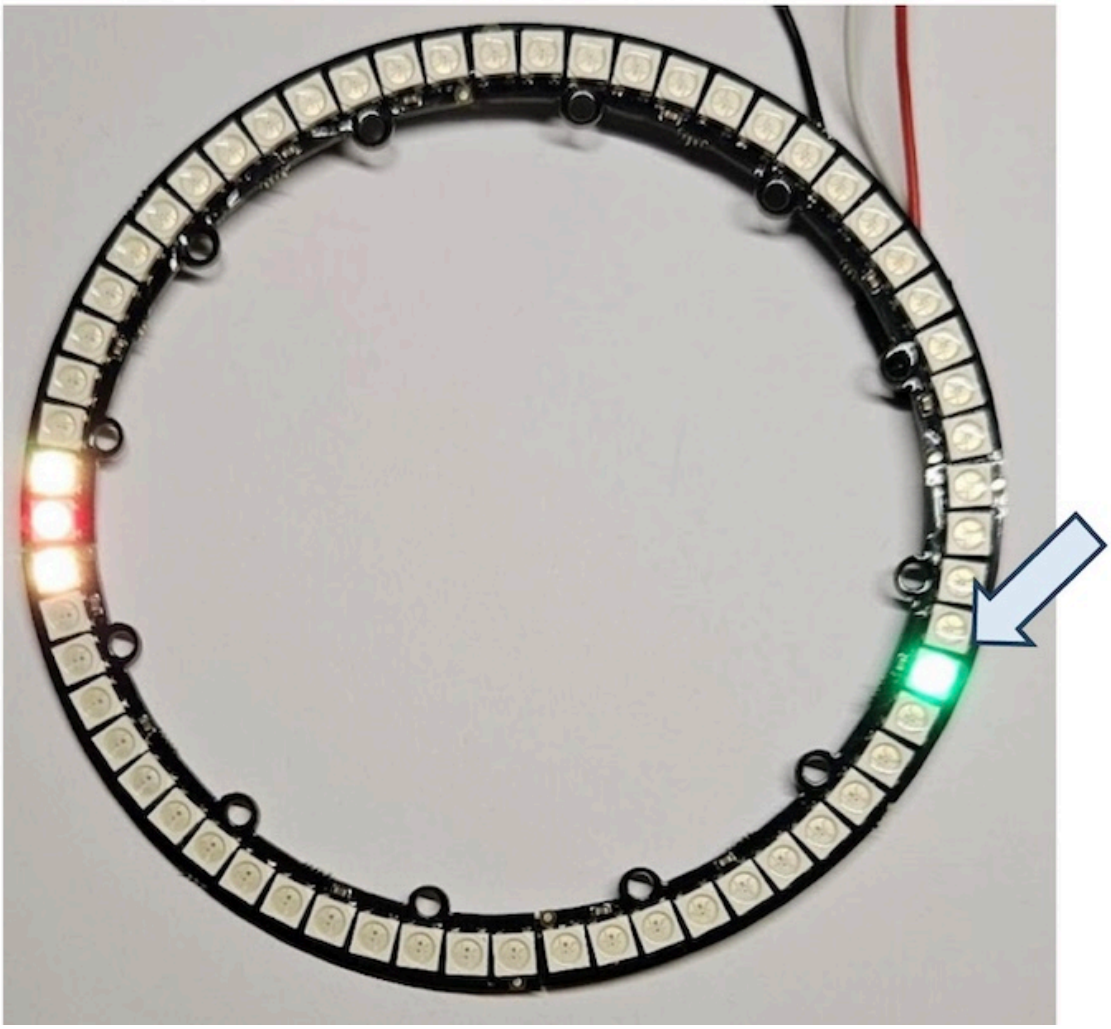
Sobald das Gerät eingeschaltet wird, startet ein neues Spiel automatisch. Zum Abbrechen eines laufenden Spiels kann das Gerät einfach ausgeschaltet werden.

Das Ziel des Spiels ist es, den Push-Button genau dann zu drücken, wenn die zirkulierende grüne LED sich mit der roten bzw. gelben LED überlagert. Das Gerät verfügt über eine integrierte Speicherung und zeichnet den aktuellen Highscore automatisch auf.

Erklärung der Anzeigemodi des LED-Ringes und LCD-Displays

Die Spielanzeige erfolgt hier sowohl über das LCD-Display als auch den LED-Ring. Beide verfügen über unterschiedliche

Funktionen, die je nach Spielverlauf aktiviert werden.



	**Gelb: +1 Punkt – Half Hit**
	Das zirkulierende grüne Licht wird an einem der äußeren LEDs angehalten. Der gesamte Ring blinkt gelb auf.
	**Grün: + 2 Punkte – Full Hit**
	Das zirkulierende grüne Licht wird im richtigen Moment im zentralen LED angehalten. Der gesamte Ring blinkt grün auf.

	**ROT: -1 Leben (Flop)**
	Das zirkulierende grüne Licht wird im falschen Moment außerhalb der stationären LEDs angehalten. Der gesamte Ring blinkt rot auf.

Game Over

Bei einem Flop verliert der Spielende ein Leben. Verbleibende Leben werden auf dem Bildschirm angezeigt. **Bei Verlust des letzten Lebens beginnt automatisch ein neues Spiel.**

Benötigte Hardware für das Cyclone

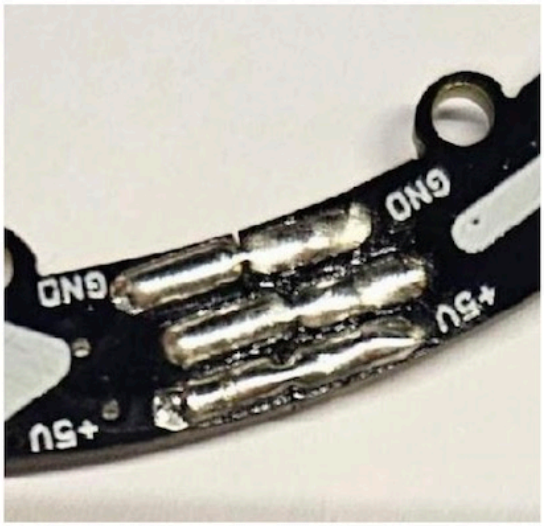
Die Hardware ist erschwinglich und leicht zu finden. Sie umfasst ein Arduino Nano-Board, einen WS2812B-LED-Ring, ein 16×2-Zeichen-LCD mit I2C-Schnittstelle, einen Taster, einen Netzschalter und einen Summer. Diese Komponenten sind alle in einem 3D-gedruckten Gehäuse integriert, welches vom Design an einen Gameboy Color angelehnt wurde. Der Mikrocontroller wird mithilfe eines USB-Netzteils und einer 5000 mAh USB-Powerbank mit Spannung versorgt, **sodass das Spiel problemlos mobil betrieben werden kann.**

Löten des LED-Ringes

Der LED-Ring besteht aus vier einzelnen Viertelringen, diese weisen je 3 Anschlüsse auf:

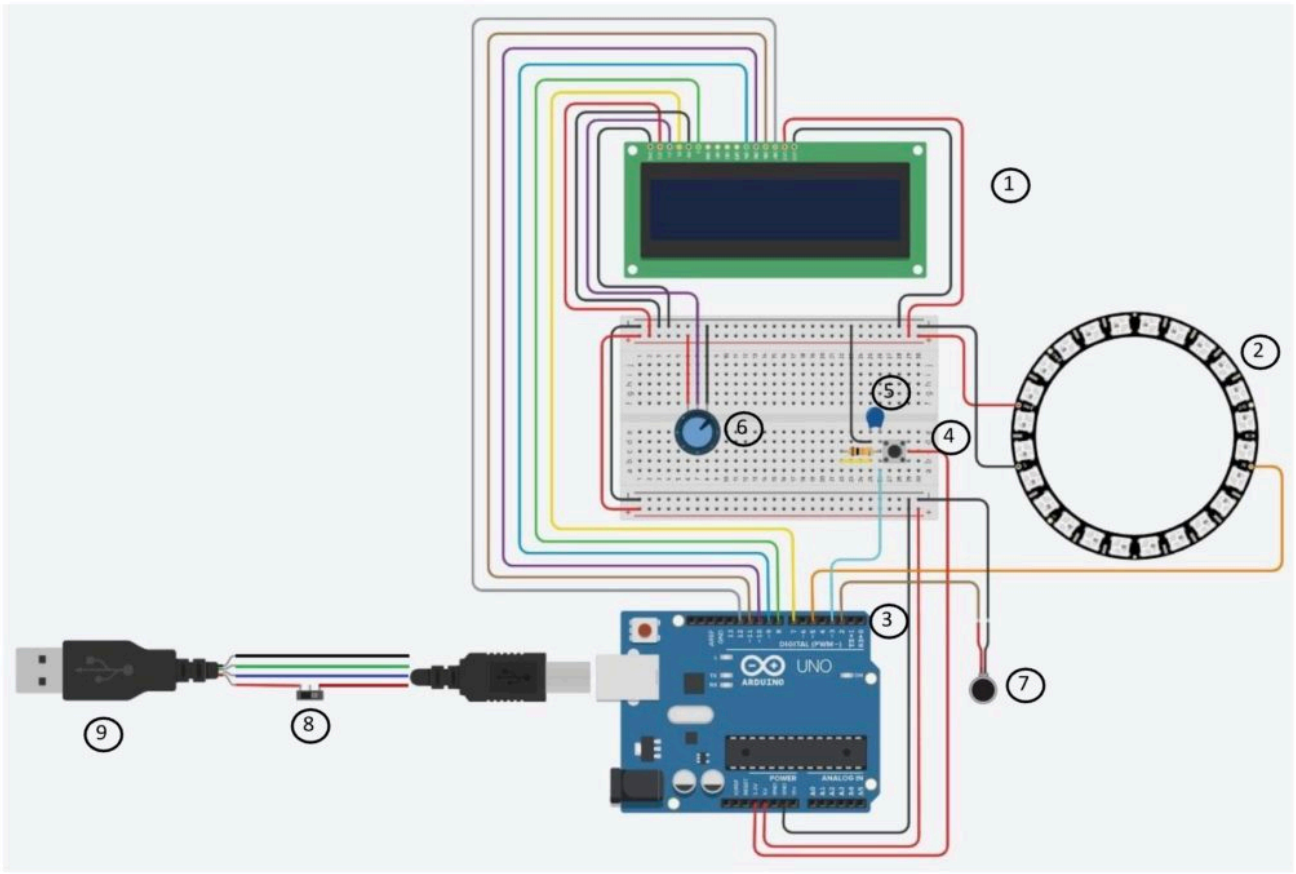
- 5V: Energieversorgung des LED-Ringes
- Datenleitung: Übermittelt die Befehle des Microcontrollers an die LEDs
- Ground: Schließen des Stromkreises

Die Verbindung der Leitungen erfordert einen Lötkolben und Lötzinn. Die korrekte Temperatureinstellung des Lötkolbens ist dabei zu beachten, um das Lötzinn verarbeiten zu können. Im Anschluss sollten die einzelnen Lötstellen auf ihre Qualität überprüft werden, **da fehlerhafte Verbindungen später zu Problemen bei der Steuerung der LEDs führen können.** Es sollte darauf geachtet werden, nicht zu viel Lötzinn zu verwenden, da sich dieses sonst auf die benachbarten Leitungen ausbreiten kann und zu Kurzschlüssen führt. Zusätzlich ist auf mögliche Lunker in den Lötstellen zu achten, da unsaubere Verbindungen die Spannungsversorgung der LEDs beeinträchtigen und die Datenübertragung stören können.



Verkabelung der Bauteile

Die Verkabelung der einzelnen Komponenten kann gemäß der Abbildung vorgenommen werden.



Stückliste

Nr.	Bezeichnung
1	LCD Modul Display
2	LED-Ring
3	Arduino NANO V3
4	Druckknopf-Mini-Rundschalter
5	Kondensator 18µF
6	Potentiometer
7	Active Alarm Buzzer
8	Wippe
9	USB 2.0 Kabel

Gehäuse

Das Gehäuse kann anhand der im Anhang zur Verfügung gestellten STL-Dateien (Download weiter unten) ganz einfach aus dem 3D-Drucker gedruckt werden.

Tipp: 3D-Druckaufträge können lokal in Maker Space Standorten oder auch online günstig gedruckt werden.

Zusätzlich besteht die Möglichkeit an Maker Space Standorten Kurse rund um das Thema 3D-Druck sowie der zugehörigen Software „Fusion 365“ zu absolvieren.

Software erklärt

Zuerst einmal ist es wichtig zu erwähnen, dass diverse Bibliotheken in die Arduino-IDE installiert werden müssen. Der Programmcode verwendet dabei die folgenden Bibliotheken:

```
#include <Adafruit_NeoPixel.h>
#include <util/atomic.h>
#include <LiquidCrystal_I2C.h>
#include <Arduino.h>
#include <EEPROM.h>
```

Der grundlegende Aufbau des Programms besteht aus mehreren CPP- und Header-Dateien. **Dies ist aufgrund der Größe des Programmcodes ratsam, um die Übersicht zu behalten.**

Die Header-Datei enthält dabei die Deklarationen der einzelnen Elemente. Die CPP-Dateien enthalten den eigentlichen Programmcode der in der Header-Datei deklarierten Klassen und Funktionen.

Software

Im Folgenden werden die wichtigsten Abschnitte und ihre

Funktionen beschrieben. Dabei werden potenzielle Anpassungen des Codes aufgezeigt, sodass ein individuelles Spielerlebnis möglich ist. Zusätzlich werden generelle Grundlagen zur Umgebung Arduino-IDE angeschnitten.

Highscore

Das Gerät speichert mithilfe der Funktion EEPROM (electrically erasable programmable Read-Only Memory) den Highscore und gibt diesen über das LCD-Display wieder. Die Verwendung der Funktion EEPROM in der Arduino EEPROM-Bibliothek ermöglicht es, Daten zu speichern und abzurufen, **die auch nach dem Ausschalten des Geräts erhalten bleiben.** Wird der Highscore gebrochen, wird dieser unmittelbar gespeichert. Ein Ausschalten des Gerätes während eines laufenden Spiels, führt zu einem Verlust des aktuellen Spielstandes, **der Highscore bleibt aber weiterhin erhalten.**

Einstellungen

Um die Schwierigkeit des Spiels individuell zu gestalten, können verschiedene Änderungen am Code vorgenommen werden:

Anzahl der LEDs

Die Anzahl der LEDs, welche im Spiel getroffen werden sollen, kann in der CPP-Datei LED-Ring.CPP angepasst werden. Hierzu können beliebig viele LEDs an verschiedenen Positionen des Rings eingefügt werden. Zusätzlich lässt sich die Farbe der LEDs beliebig konfigurieren. Die erforderlichen Änderungen sind in folgendem Code-Abschnitt vorzunehmen:

```
m_strip.clear();
m_strip.setPixelColor((LEDRING_START_NUM - 1), 255, 165, 0);
m_strip.setPixelColor(LEDRING_START_NUM, 255, 0, 0);
m_strip.setPixelColor((LEDRING_START_NUM + 1), 255, 165, 0);
m_strip.setPixelColor(m_currentLed, 0, 255, 0);
```

Geschwindigkeit der zirkulierenden LED

Die Geschwindigkeit der zirkulierenden LED lässt sich in der Datei **config.CP** variieren:

Um die Geschwindigkeit des zirkulierenden LEDs zu ändern, muss der Wert für das Start-Delay angepasst werden. Ein Wert von 30ul entspricht einer Verzögerung von 30 Millisekunden. Durch eine entsprechende Änderung des Wertes kann das Spiel leichter oder auch schwieriger gestaltet werden.

#define LEDRING_START_DELAY_MS 30ul

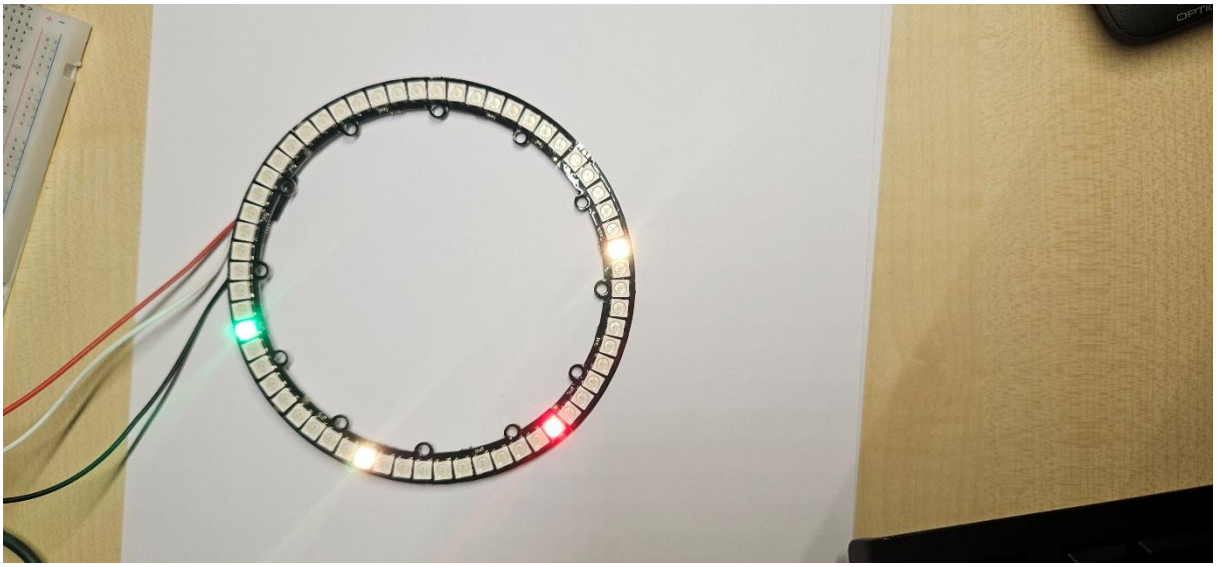
Wechseln der Position der zu treffenden LED

Die Anzahl der LEDs, welche im Spiel getroffen werden sollen, kann in der CPP-Datei **LED-Ring.CPP** verändert werden. Hierzu können beliebig viele LEDs an verschiedenen Positionen des Rings eingefügt werden. Zusätzlich lässt sich die Farbe der LEDs beliebig anpassen.

Eine beispielhafte Änderung der Position würde im Code so aussehen

```
 m_strip.clear();
 m_strip.setPixelColor((LEDRING_START_NUM - 10), 255, 165, 0);
 m_strip.setPixelColor(LEDRING_START_NUM, 255, 0, 0);
 m_strip.setPixelColor((LEDRING_START_NUM + 10), 255, 165, 0);
 m_strip.setPixelColor(m_currentLed, 0, 255, 0);
}
```

und zu folgender Positionierung auf dem LED-Ring führen:



Achtung: Wird die Anzahl der LEDs geändert, muss auch der Treffer-Bereich angepasst werden. **Diese Anpassung erfordert viele weitere Änderungen des Codes und sollte nur von Benutzern mit fortgeschritteneren Programmierkenntnissen vorgenommen werden.**

Anzahl an Leben

Um die Anzahl an Leben zu ändern, kann der Wert in der CPP-Datei **config.CPP** beliebig festgelegt werden. Standardmäßig sind 3 Leben pro Durchgang eingestellt.

```
#ifndef PLAYER_TOTLIVES
#define PLAYER_TOTLIVES 3
#endif
```

Visuelle Einstellungen

Zusätzlich können auch visuelle Änderungen der LEDs vorgenommen werden. Hierbei kann sowohl die Farbe der LEDs als auch die Helligkeit verändert werden.

Helligkeit

Die Helligkeit kann in der CPP-Datei **LEDRing.CPP** angepasst werden.

Hierzu ist die folgende Code-Zeile anzupassen:

m_strip.setBrightness(15)

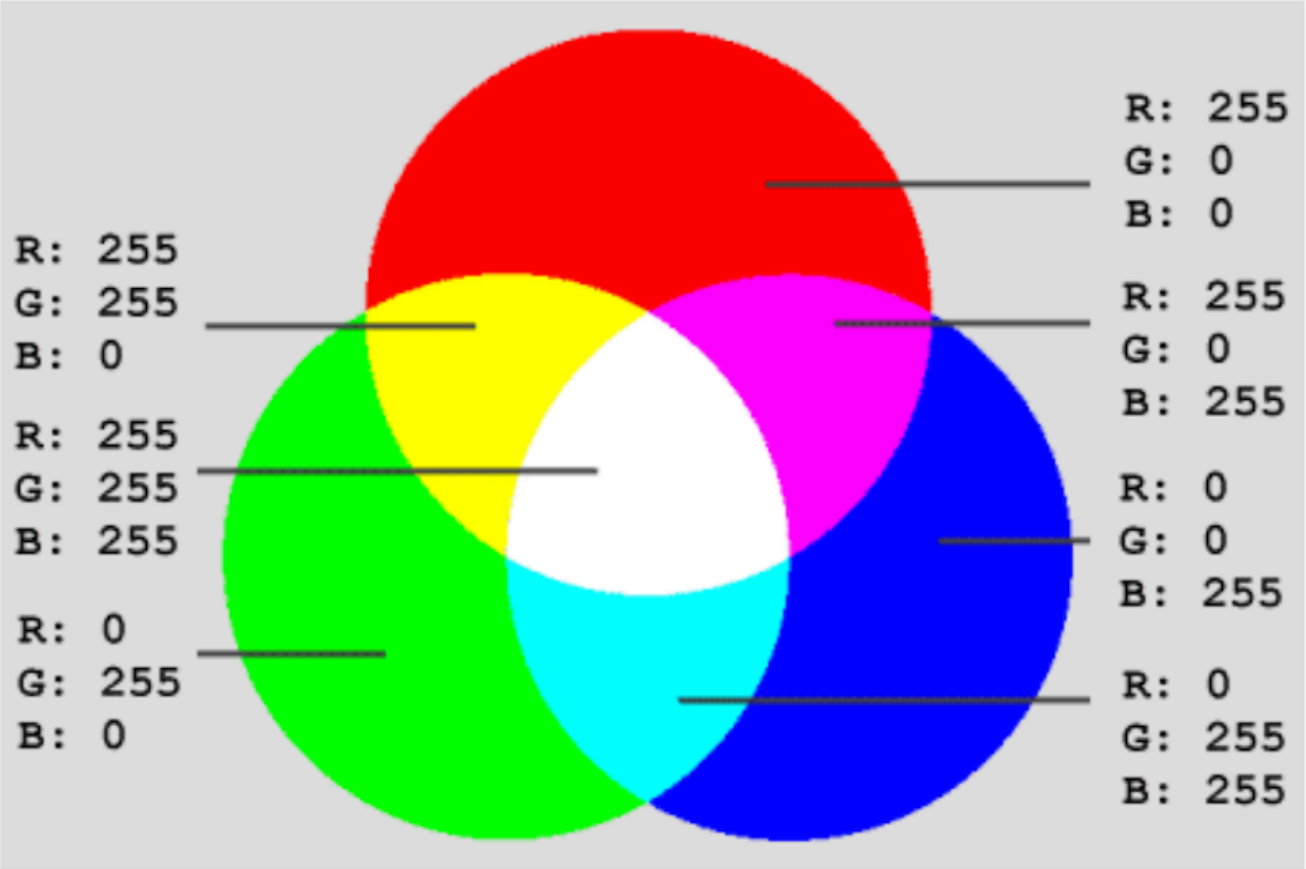
Um die Einstellung der Helligkeit zu regulieren, kann die in den Klammern angegebene Zahl auf einen Wert zwischen 1 und 100 eingestellt werden. Hierbei steht der Zahlenwert für die prozentuale Helligkeit der LEDs.

Farbe

Die Farbe der LEDs wird ebenfalls in der CPP-Datei **LEDRing.CPP** festgelegt. Hierzu sind folgende Code-Zeilen entsprechend der gewünschten Farbe einzustellen.

```
m_strip.setPixelColor((LEDRING_START_NUM - 1), 255, 165, 0);
m_strip.setPixelColor(LEDRING_START_NUM, 255, 0, 0);
m_strip.setPixelColor((LEDRING_START_NUM + 1), 255, 165, 0);
m_strip.setPixelColor(m_currentLed, 0, 255, 0);
```

Die Farben sind als RGB-Werte definiert. Eine Anpassung der Farben erfolgt über das RGB-Spektrum (Rot, Grün, Blau).



Um die Einstellung der LED-Farbe anzupassen, werden die jeweiligen Grundfarbwerte für „Rot“, „Grün“ und „Blau“ mit den Zahlenwerten von 0 – 255 definiert. Der Zahlenwert 255 entspricht einem kräftigen Rot/Grün/Blau-Ton mit maximaler Intensität. Senkt man den Zahlenwert, sinkt die Intensität des Rot/Grün/Blau-Tones. Durch die Variation der Intensitäten der drei Grundfarben können nahezu unendlich viele Farben erzeugt werden.

Das Spiel bietet die Möglichkeit, frei nach Fantasie angepasst zu werden und um zusätzliche Funktionen sowie Schwierigkeitsgrade erweitert zu werden. Dabei haben Nachbauende die Freiheit, den Programmcode und die Spielmechaniken nach Belieben zu erweitern und anzupassen.

Wir wünschen viel Spaß bei der Umsetzung und dem Ausprobieren des Cyclones.

Informationen zu den Autor*innen:

Im sechsten Semester des Bachelor of Engineering Studiengangs Produktion- und Prozessmanagement an der Hochschule Heilbronn gilt es, eine Projektleistung eigenständig umzusetzen. Um unsere elektrotechnischen, CAD- sowie Programmierkenntnisse in der Praxis zu testen, entschieden wir uns im Dreierteam für die Umsetzung des Projektes Cyclone-Arcade Spiel bei Prof. Dr. Juliane König-Birk.

Das Projekt bietet mit der Umsetzung einen umfangreichen Einstieg in die Elektrotechnik und die Programmierung des Mikrocontrollers zur Steuerung der einzelnen Bauteile mit der Arduino-IDE, welche auf der Programmiersprache C und C++ beruht.

Der Aufbau beinhaltet viele grundlegende Bauteile. Dies hilft, ein besseres Verständnis zu erlangen, wie ein Produkt von der Idee bis hin zur Produktentstehung umgesetzt wird.

Download

Den Programmcode, die STL-Datei für das Gehäuse sowie den Schaltplan kannst du hier herunterladen.