

# „LEDice“- Entwicklerdokumentation

Wintersemester 2023/24

---

Jakob Deutschendorf  
FH Wedel  
Studiengang Smart Technology

Version 1.0 (Stand: 17.01.24)

## Einleitung

Beim LEDice handelt es sich um ein System aus sieben LED-Lampen, einem Taster sowie einem Arduino Uno R3, welcher das System steuert. Es besitzt folgende Funktionen:

- Auswürfeln einer Zahl in einem programmdefinierten Bereich durch Knopfdruck
- Ausgabe der Zahl als programmdefiniertes Muster über die LEDs inklusive Animation
- Bibliothek zur Ausgabe von Zahlen auf einer LED-Struktur
- Selbststartender Ruhemodus inklusive Animation, der über den Taster beendet werden kann

# Inhalt

<b>Einleitung.....</b>	<b>1</b>
<b>Allgemeines.....</b>	<b>3</b>
<b>Ressourcen.....</b>	<b>3</b>
Programm.....	3
Extensions.....	3
Bibliotheken.....	4
Schaltplan.....	5
<b>Repository-Inhalt.....</b>	<b>6</b>
<b>Aufbau.....</b>	<b>6</b>
main.cpp.....	6
Variablen/Konstanten.....	6
Methoden.....	7
LED_output.....	7
Variablen/Konstanten.....	7
Methoden.....	7
<b>Projekt zusammensetzen.....</b>	<b>9</b>
Funktionalität überprüfen.....	11
<b>Mögliche Weiterentwicklung.....</b>	<b>12</b>
<b>Kontakt.....</b>	<b>12</b>

## Allgemeines

Die Funktion des LEDice besteht in der Auswürfelung zufälliger Zahlen innerhalb eines definierten Bereichs sowie der Ausgabe dieser über an den Arduino angeschlossene LED-Lampen. Zu Beginn wird eine Startanimation abgespielt, bei der die Muster aller hinterlegter Zahlen dargestellt werden. Danach geht das System in eine Wartephase über, bei der jeder Knopfdruck zum Abspielen einer Würfelanimation, dem Ermitteln einer zufälligen Zahl und dem Anzeigen der Zahl führt. Dieser Vorgang ist beliebig oft wiederholbar. Wird zu lange kein Würfelvorgang angefordert, werden die Lampen ausgeschaltet und das System geht in einen Ruhemodus über, aus dem es durch einen weiteren Knopfdruck geweckt werden kann.

---

## Ressourcen

Repository: <https://github.com/Kraeyx/LEDice>

Diese Doku basiert auf Stand: 497031f379b7b422ae031a30ac471c1e5ab4a04d

## Programm

IDE: Visual Studio Code (v. 1.85.1)

### Extensions

Name: C/C++

- Id: ms-vscode.cpptools
- Description: C/C++ IntelliSense, debugging, and code browsing.
- Version: 1.18.5
- Publisher: Microsoft
- VS Marketplace Link:  
<https://marketplace.visualstudio.com/items?itemName=ms-vscode.cpptools>

Name: C/C++ Extension Pack

- Id: ms-vscode.cpptools-extension-pack
- Description: Popular extensions for C++ development in Visual Studio Code.
- Version: 1.3.0
- Publisher: Microsoft

- VS Marketplace Link:  
<https://marketplace.visualstudio.com/items?itemName=ms-vscode.cpptools-extension-pack>

Name: C/C++ Themes

- Id: ms-vscode.cpptools-themes
- Description: UI Themes for C/C++ extension.
- Version: 2.0.0
- Publisher: Microsoft
- VS Marketplace Link:  
<https://marketplace.visualstudio.com/items?itemName=ms-vscode.cpptools-themes>

Name: PlatformIO IDE

- Id: platformio.platformio-ide
- Description: Your Gateway to Embedded Software Development Excellence: CMSIS, ESP-IDF, FreeRTOS, libOpenCM3, mbed OS, SPL, STM32Cube, Zephyr RTOS, Arduino, ARM, AVR, Espressif (ESP8266/ESP32), FPGA, MCS-51 (8051), MSP430, Nordic (nRF51/nRF52), PIC32, RISC-V, Raspberry Pi (RP2040), STMicroelectronics (STM8/STM32)
- Version: 3.3.2
- Publisher: PlatformIO
- VS Marketplace Link:  
<https://marketplace.visualstudio.com/items?itemName=platformio.platformio-ide>

Name: Serial Monitor

- Id: ms-vscode.vscode-serial-monitor
- Description: Send and receive text from serial ports.
- Version: 0.11.0
- Publisher: Microsoft
- VS Marketplace Link:  
<https://marketplace.visualstudio.com/items?itemName=ms-vscode.vscode-serial-monitor>

Bibliotheken

[Arduino](#) : Standardbibliothek

[sleep](#) : Bibliothek, welche die Integration eines Ruhemodus ermöglicht

[LED output](#): Bibliothek, die die Ausgabe auf den LEDs ermöglicht

## Schaltplan

Zur Erstellung des Schaltplans wurde die Software KiCad (<https://www.kicad.org>) genutzt.

## Repository-Inhalt

lib: Beinhaltet die Bibliothek LED\_output, die die Ausgabe regelt

src: Beinhaltet die Datei main.cpp, in der das Hauptprogramm beherbergt ist

res: Beinhaltet Dokumente wie Entwicklerdokumentation, Benutzerdokumentation, Stückliste und Schaltplan (sowie die KiCad-Dateien des Schaltplans)

.vscode: Beinhaltet Konfigurationen für die IDE

README: Textdokument mit einer Zusammenfassung des Projektes

---

## Aufbau

### main.cpp

#### Variablen/Konstanten

LED\_COUNT: Anzahl der angeschlossenen LEDs

LED\_PINS: Pinbelegung der LEDs. Die Indexe der LEDs sind in folgender Reihenfolge auf dem Breadboard platziert:

0		4
1	3	5
2		6

Werden die Elemente also beispielsweise wie folgt belegt:

```
const int LED_PINS[LED_COUNT] = {6,7,8,5,12,3,4};
```

sind die PINs in folgendem Schema mit den LEDs verknüpft:

6		12
7	5	3
8		4

BUTTON\_PIN: Pin, an den der Button angeschlossen ist

RESULT\_MINIMUM: Mindestens gewürfelte Zahl

RESULT\_MAXIMUM: Höchstens gewürfelte Zahl

SLEEP\_CLOCK: Taktzahl, nach der der Ruhemodus aktiviert wird

`out`: Instanz der Bibliothek `LED_output` zur Verwaltung der Ausgabe

`clockwaiter`: Zähler für den Takt

`start`: Information, ob ein Startvorgang läuft

## Methoden

`setup()`: Initialisiert die Pins und den Initialzustand. Nutzt außerdem den unbelegten Analog-Pin 0, um einen zufälligen Seed für den Zufallsgenerator zu erstellen.

`loop()`: Testet auf einen Knopfdruck und führt, wenn nötig, den Würfelvorgang durch. Unterscheidet hierbei zwischen verschiedenen Fällen:

1. Kein Startvorgang im Gang und Knopf gedrückt: Blinkt dreimal, würfelt eine Zahl und gibt sie über die Ausgabe aus. Setzt außerdem den Taktzähler auf 0.
2. Kein Startvorgang und hoher Takt: Fügt einen Interrupt hinzu, der den Ruhemodus durch einen Knopfdruck beendet. Schaltet danach die LEDs aus und aktiviert den Schlafmodus.
3. Startvorgang ist im Gang: Spielt die Startanimation ab.
4. Keiner der Fälle trifft zu: Der Takt wird erhöht (und ausgegeben)

`enterSleepMode()`: Startet den Sleep-Modus

`isAwake()`: Setzt den Taktzähler nach dem Aufwachen auf 0 und löst den Interrupt vom Button

## LED\_output

### Variablen/Konstanten

`_pins`: Zeiger auf den ersten Pin

`_pinCount`: Anzahl der LEDs

`MIN_NUMBER`: Minimal darstellbare Zahl


`MAX_NUMBER`: Maximal darstellbare Zahl

### Methoden

`LED_output(const int* pin, const int pinCount, const int minNumber, const int maxNumber)`: Konstruktor einer Instanz der Bibliothek

`clearLEDs()`: Schaltet alle LEDs aus





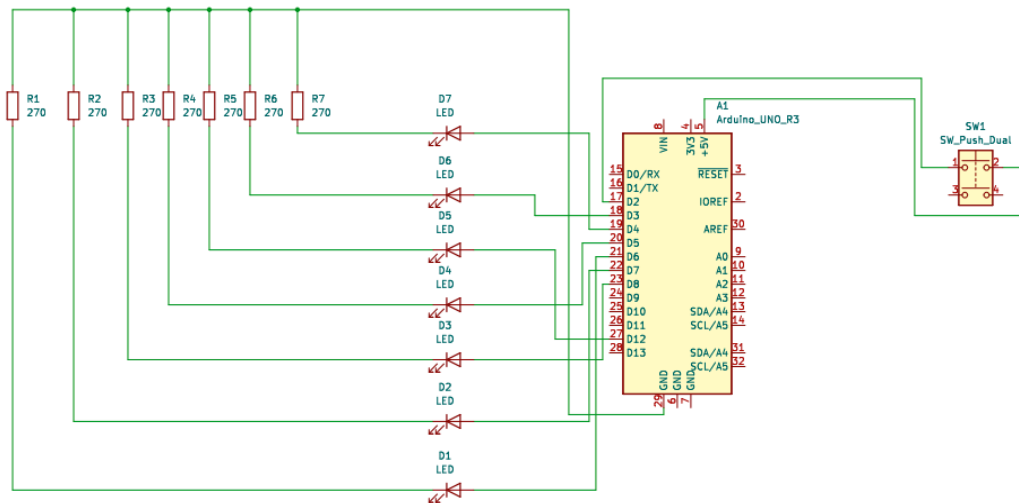
`displayNumber(int num):` Stellt die übergebene Nummer dar. Um die Darstellung weiterer Nummern zu realisieren, muss die zugehörige Belegung der LEDs in dieser Methode eingetragen werden

`blink():` Lässt alle LEDs aufblinken

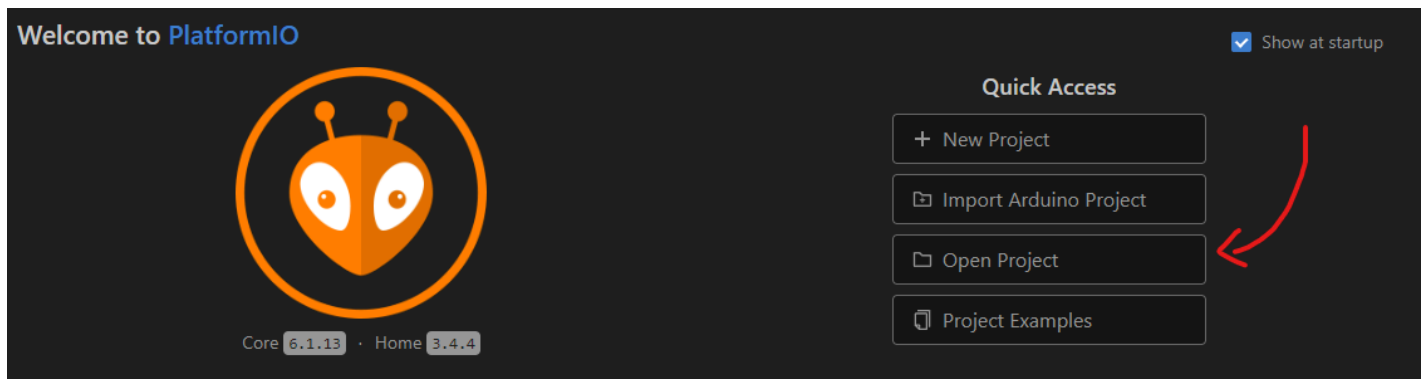
`initPins():` Initialisiert die Pins für die Ausgabe

## Projekt zusammensetzen

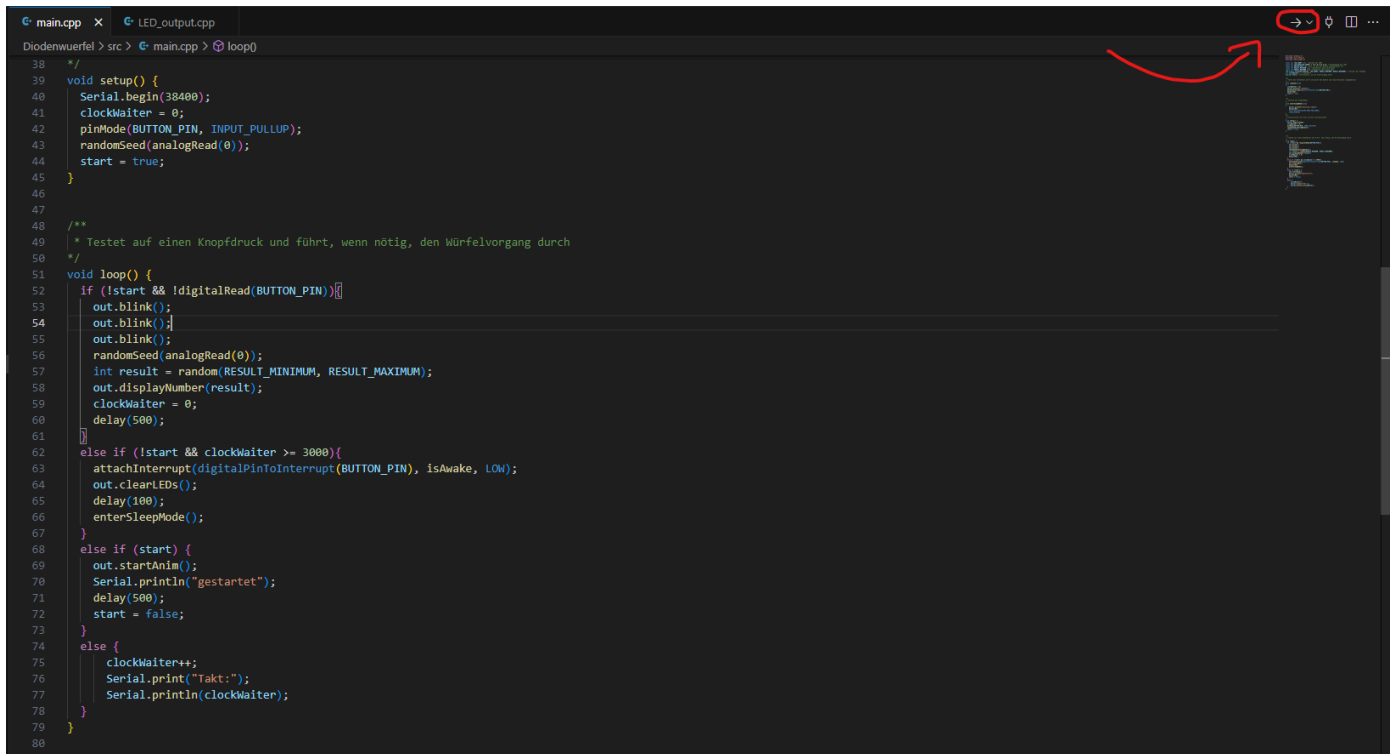
Schritt 1: Den Arduino entsprechend dem Schaltplan (zu finden im Repository-Ordner res) über das Breadboard mit den LEDs, den Widerständen und dem Taster verbinden.



Schritt 2: Das Repository in die IDE klonen und über PlatformIO öffnen



Schritt 3: Die aufgesetzte IDE mit dem Arduino verbinden und das Programm auf den Arduino laden



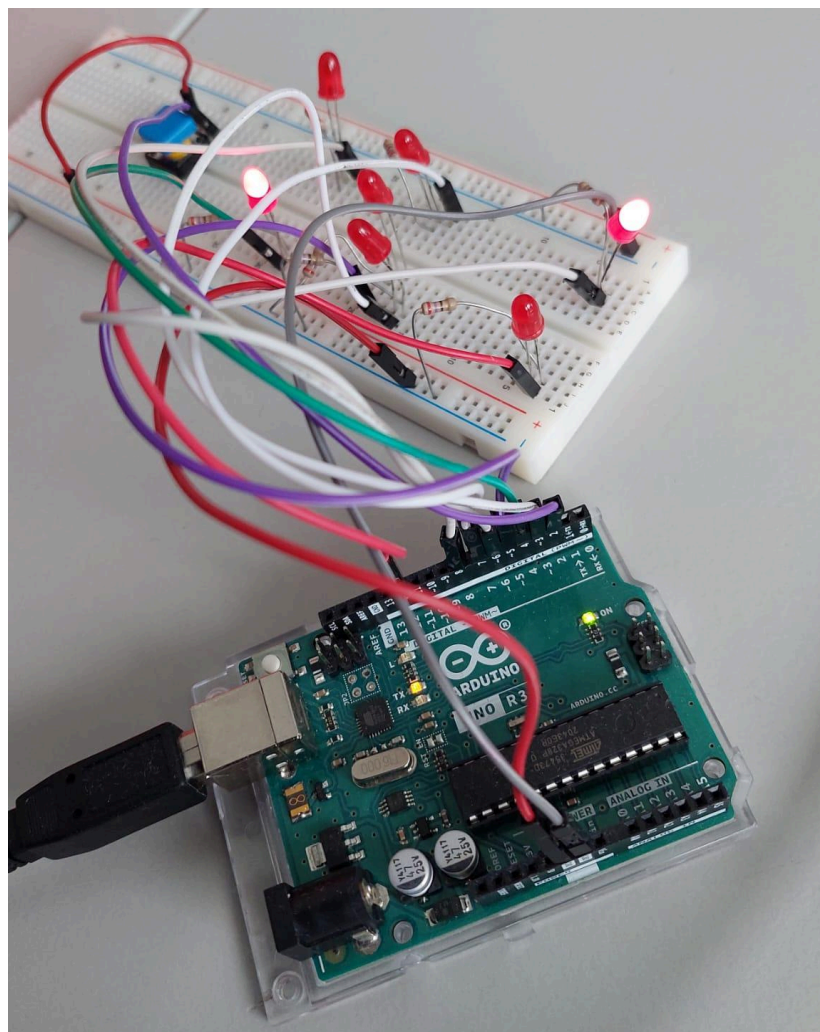
```
38  */
39  void setup() {
40      Serial.begin(38400);
41      clockWaiter = 0;
42      pinMode(BUTTON_PIN, INPUT_PULLUP);
43      randomSeed(analogRead(0));
44      start = true;
45  }
46
47
48  /**
49   * Testet auf einen Knopfdruck und führt, wenn nötig, den Würfelvorgang durch
50   */
51  void loop() {
52      if (!start && !digitalRead(BUTTON_PIN)) {
53          out.blink();
54          out.blink();
55          out.blink();
56          randomSeed(analogRead(0));
57          int result = random(RESULT_MINIMUM, RESULT_MAXIMUM);
58          out.displayNumber(result);
59          clockWaiter = 0;
60          delay(500);
61      }
62      else if (!start && clockWaiter >= 3000) {
63          attachInterrupt(digitalPinToInterrupt(BUTTON_PIN), isAwake, LOW);
64          out.clearLEDs();
65          delay(100);
66          enterSleepMode();
67      }
68      else if (start) {
69          out.startAnim();
70          Serial.println("gestartet");
71          delay(500);
72          start = false;
73      }
74      else {
75          clockWaiter++;
76          Serial.print("Takt:");
77          Serial.println(clockWaiter);
78      }
79  }
80
```

## Funktionalität überprüfen

Die Funktionalität der Software kann über den Serial Monitor innerhalb der IDE überprüft werden. Sowohl der aktuelle Takt, als auch Operationen wie der Ruhemodus werden hier protokolliert.

**WICHTIG:** Die Baudrate des Monitors muss mit der im `setup()` des Programms definierten übereinstimmen.

Die Funktionalität der Hardware lässt sich am besten über die Würfelvorgänge testen. Sobald der Arduino an den Strom angeschlossen ist, lässt sich anhand der Startanimation überprüfen, ob die LEDs funktionsfähig sind. Danach kann der Taster über das Auslösen eines Würfelzyklus getestet werden. Treten hierbei keine Probleme auf, ist das Projekt funktionsfähig.



## Mögliche Weiterentwicklung

- Unterstützung weiterer populärer Würfelarten z.B. 20 Seiten
- Unterstützung mehrerer Würfelarten, zwischen denen gewechselt werden kann
- Rechenoperationen (Würfe addieren, höchste Zahl behalten etc.)
- Soundeffekte
- Neue Arten der Ausgabe
- Zusätzliche Animationen

---

## Kontakt

Jakob Deutschendorf

stud105836@fh-wedel.de