

## Game of Life

### Ziel

Implementierung von Conway's „Game of Life“ mit einer 8x8 LED Matrix, welche über den I<sup>2</sup>C Bus angesteuert wird. Es soll eine funktionsfähige Implementierung mit dem Initialzustand „Glider“ demonstriert werden (siehe Abbildung 1). Die Implementierung soll die Technik der Doppelpufferung („double buffering“) verwenden.

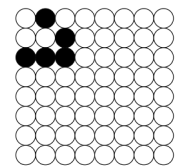
### Vorgehensweise

Beschäftigen Sie sich mit der Ansteuerung der LED Matrix, sowie der Funktionsweise des I<sup>2</sup>C Bus. Probieren Sie dazu das „matrix8x8“ Beispiel aus der Adafruit LED Backpack Bibliothek aus. Zur Ansteuerung der LED Matrix verwendet die Adafruit Bibliothek die I<sup>2</sup>C Wire Bibliothek von Arduino.

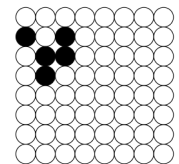
Implementieren Sie nun Conway's Game of Life mit einem 8x8 Zellen großem Spielfeld auf der LED Matrix. Jede Zelle des Spielfelds kann lebendig (leuchtet) oder tot sein (leuchtet nicht). Es werden beliebig viele Runden durchgerechnet. In jeder Runde wird anhand des Zustands der Nachbarn einer Zelle der Status dieser Zelle in der nächsten Runde bestimmt.

### Die Regeln sind dabei folgende:

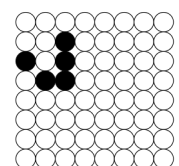
- Eine lebendige Zelle mit weniger als zwei lebendigen Nachbarn stirbt an Einsamkeit
- Eine lebendige Zelle mit zwei oder drei lebendigen Nachbarn bleibt am Leben
- Eine lebendige Zelle mit mehr als drei lebendigen Nachbarn stirbt an Überbevölkerung
- Eine tote Zelle mit genau drei lebendigen Nachbarn wird neu geboren



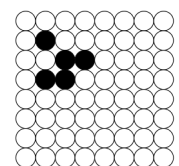
Initialzustand



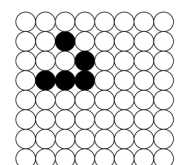
1. Schritt



2. Schritt



3. Schritt



4. Schritt, siehe Initialzustand

Abbildung 1) Glider

Eine Zelle hat bis zu acht Nachbarn (zählen Sie nicht die Zelle selbst). Denken Sie daran, dass die Randzellen extra zu behandeln sind. Benutzen Sie zwei 8x8 Einträge große Puffer, einen für den aktuellen Schritt und einen für den nächsten Schritt. Wenn alle Zellen für die Runde ausgewertet wurden, schicken Sie den neu errechneten Puffer zur LED Matrix und wechseln die Puffer (der neu errechnete Puffer wird zum aktuellen Puffer und der andere Puffer wird überschrieben). Fügen Sie etwas Wartezeit zwischen den Schritten ein, damit man das Ergebnis visuell verfolgen kann.

### Ansteuerung der LED Matrix

Die Ansteuerung des HTK16K33 Treiber-Chip auf dem LED Backpack erfolgt über I<sup>2</sup>C. Dazu wird die Kommunikation per `Wire.beginTransmission(0x70)` initiiert (Die Adresse 0x70 ist fest eingestellt). Mit `Wire.write(0)` selektieren Sie das Register für die Display Daten und mit `Wire.write(Zeile)`, gefolgt von `Wire.write(0)` können nun hintereinander acht Zeilen an den Treiber-Chip geschickt werden. Eine Zeile ist dabei als zwei Byte kodiert. Jedes Bit des ersten Bytes steht dabei für eine aktivierte oder deaktiverte

LED in der jeweiligen Zeile. Die Kommunikation wird per `Wire.endTransmission()` abgeschlossen.

### Vorbereitung

Beschäftigen Sie sich mit Conway's Game of Life, Double Buffering, dem I<sup>2</sup>C Bus, der Wire Bibliothek und der Ansteuerung des Displays. Studieren Sie den Quelltext der Adafruit Bibliothek und die Wire\_Template Vorgaben.

### Achtung

Der HTK16K33 Chip hat zwei Eigenheiten. Zum einen ist er für 16x8 LED Felder entwickelt worden. Daher muss vor jeder Übertragung der Zeile des Puffers ein „Dummy-Byte“ geschickt werden (d.h. `write(zeile)`, gefolgt von `write(0)` für jede Zeile). Das zweite Byte ist ein Füllbyte, gedacht für die Verwendung mit einer doppelt so breiten LED Matrix. Zum anderen befindet sich die erste Spalte an der letzten Position im Zeilen-Bitvektor (d.h. die Positionen der jeweiligen LED im Bitvektor entsprechen 23456781). Durch Kapselung dieser Eigenheiten in einer Funktion zum Senden eines Puffers an die LED Matrix lassen sie sich aus dem restlichen Code heraushalten.

Für Ihre Implementierung darf am Ende nur die Wire Bibliothek verwendet werden, ein Einbinden der Adafruit Bibliothek ist nicht erlaubt. Orientieren Sie sich an den Wire\_Template Vorgaben.

### Notengebung

4,0 (Anwesend); 3,0 (Display leuchtet mit Mustern); 2,3 (+ Game of Life, kleinste Fehler); 2,0 (+ Glider korrekt); 1,7 (+ Double Buffering); 1,3 (+ Code sauber geschrieben und dokumentiert); 1,0 (+ Möglichst speichersparende Implementierung)

### Wichtige Funktionen & Bibliotheken

- Die Wire Bibliothek
  - `Wire.beginTransmission(...)`
  - `Wire.write(...)`
  - `Wire.endTransmission()`

### Sie brauchen

- Arduino Board, USB Kabel, Adafruit LED Backpack, Steckbrücken
- Vorgaben zur Benutzung von Wire
- Tutorial für das LED Backpack:
  - <https://learn.adafruit.com/adafruit-led-backpack/overview>
- Beschreibung des „Game of Life“ bei Wikipedia
  - [http://de.wikipedia.org/wiki/Conways\\_Spiel\\_des\\_Lebens](http://de.wikipedia.org/wiki/Conways_Spiel_des_Lebens)
- Arduino Referenz der I<sup>2</sup>C Wire Bibliothek
  - <http://arduino.cc/en/pmwiki.php?n=Reference/Wire>
- I<sup>2</sup>C Bus Specification von NXP Semiconductors