

Physical Computing

Linear Pong

Kategorie: Spiel

Ein Projekt von:
Alexander Baur
Marc Zinser
Korbinian Maag

Inhalt

Projektbeschreibung

Idee

LED-Stripe

Punktevergabe

UI

Komponentenliste

Liste der Teile

Codeschnipsel

Fritzing Planung

Pseudo Code

3D-Render

Projektbeschreibung

Idee

Unsere Idee ist ein kleines, einfach umzusetzendes Spiel das für 1 bzw. 2 Spieler gedacht ist.
Ein Zeitvertreib für ungefähr 5 - bis 10 Minuten.

Jeder Spieler nimmt eine der beiden Seiten (oder bei einem Spieler beide seiten) und spielt einen auf ihn Zukommenden Lichtpunkt zurück durch das drücken der Entsprechenden Taste der LED-Leiste.

Auf einem Dot-Matrix-Display wird die Punktzahl angezeigt. Wer zuerst X-Punkte erreicht im 2 Spieler Modus gewinnt.

Im Einzelspieler Modus spielt man um den High-Score.

Ein Lautsprecher ist verbaut um Soundeffekte ab zu spielen.

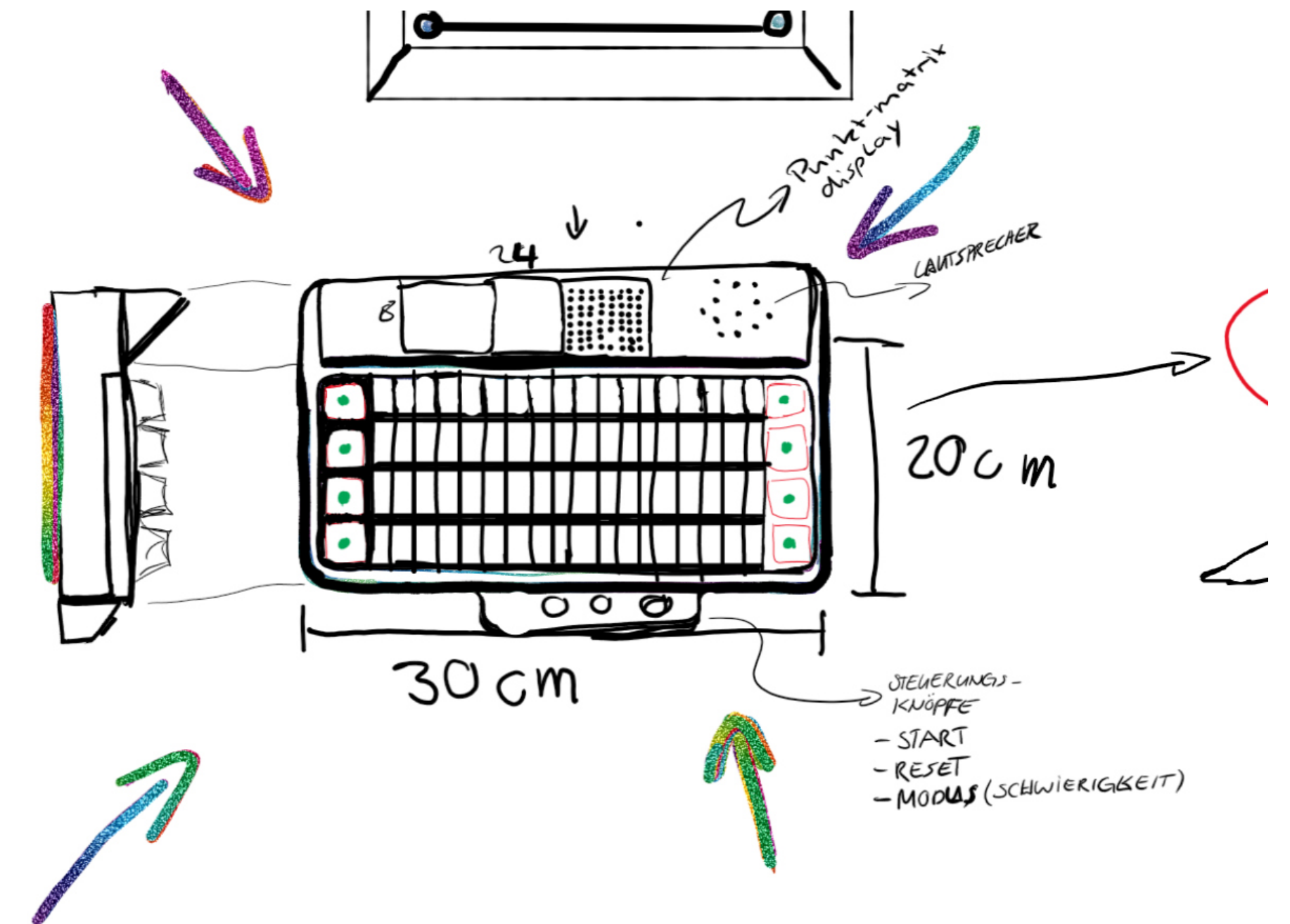


Abb. 1 Grobe Skizze

Projektbeschreibung

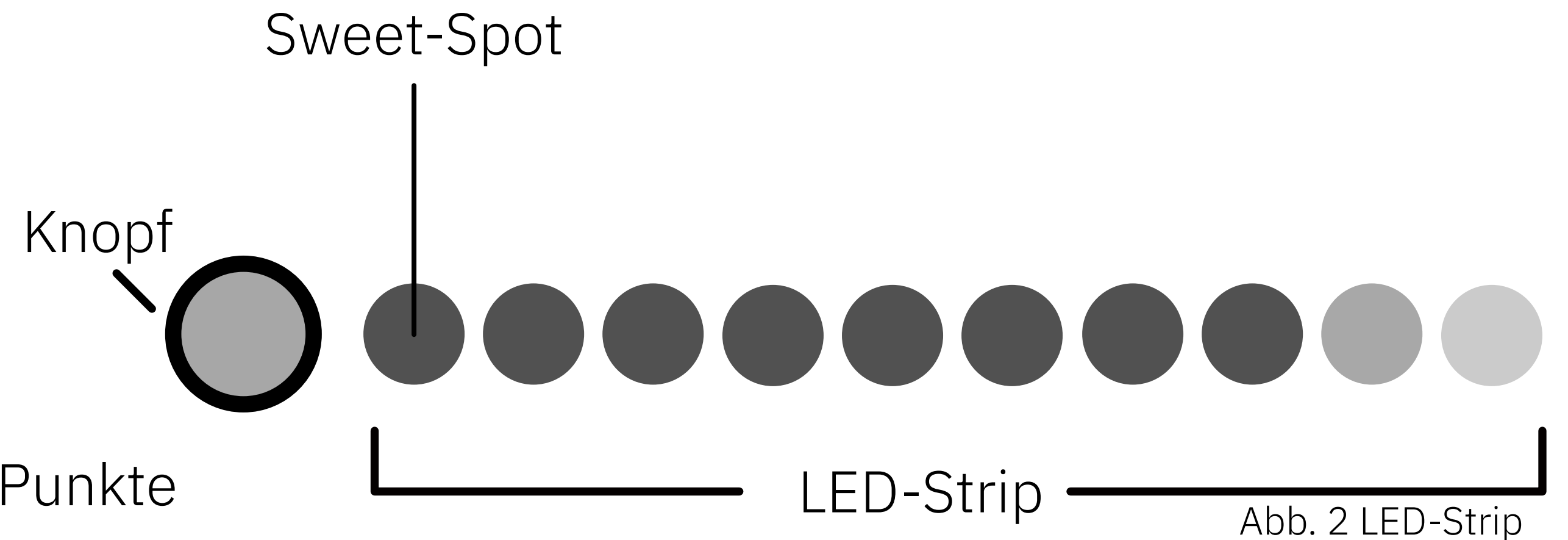
LED-Stripe

Abb. 2 zeigt einen Teil des Spielfeldes.

Ziel ist es, die auf einen zukommende LED zu Treffen durch das Drücken des Knopfes des jeweiligen LED-Stripes.

Je näher man am Sweetspot dran war mit der hellsten LED, desto mehr Punkte erhält man und desto schneller bewegt sich die LED zurück zur anderen Seite.

Wird der Sweet-Spot nicht getroffen erhält man keine Punkte und die LED bewegt sich mit einer verminderten Geschwindigkeit wieder auf die andere Seite.



Projektbeschreibung

LED-Stripe

Abb. 3 Zeigt einen Teil des Spielfeldes

Die abgeschwächten LEDs vor und hinter der Stärksten LED können auch genutzt werden um den Sweet-Spot zu treffen, jedoch gibt es dadurch weniger Punkte.

Bei Schritt 4. Drücken wir den Knopf um volle Punktzahl zu erhalten und schicken die LED mit einer höheren Geschwindigkeit wieder zurück

Nun muss der Gegenspieler schneller auf die Einkommende LED Reagieren um sie nicht zu verfehlen.

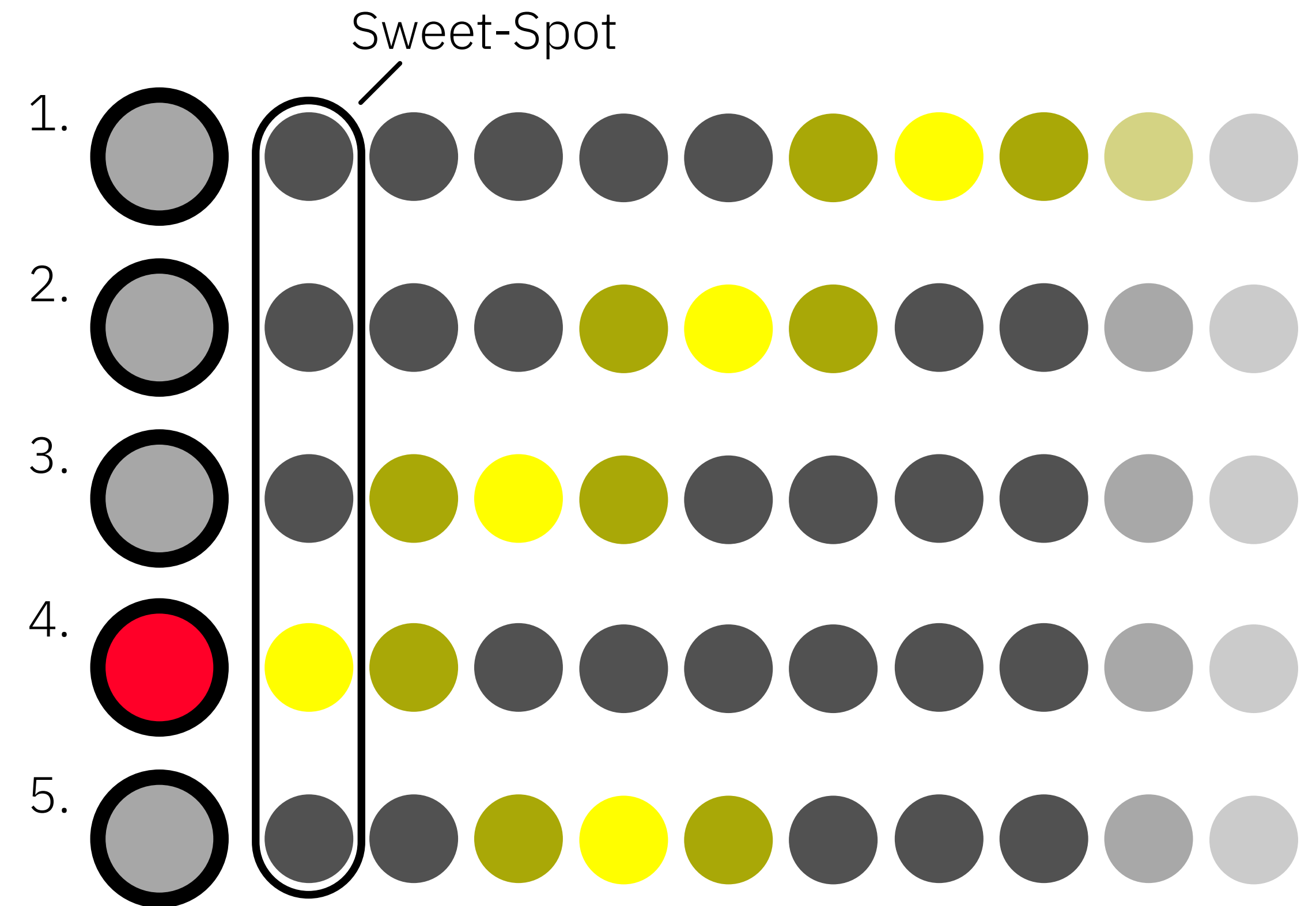


Abb. 3 LED-Bewegungsverhalten

Projektbeschreibung

Punktevergabe

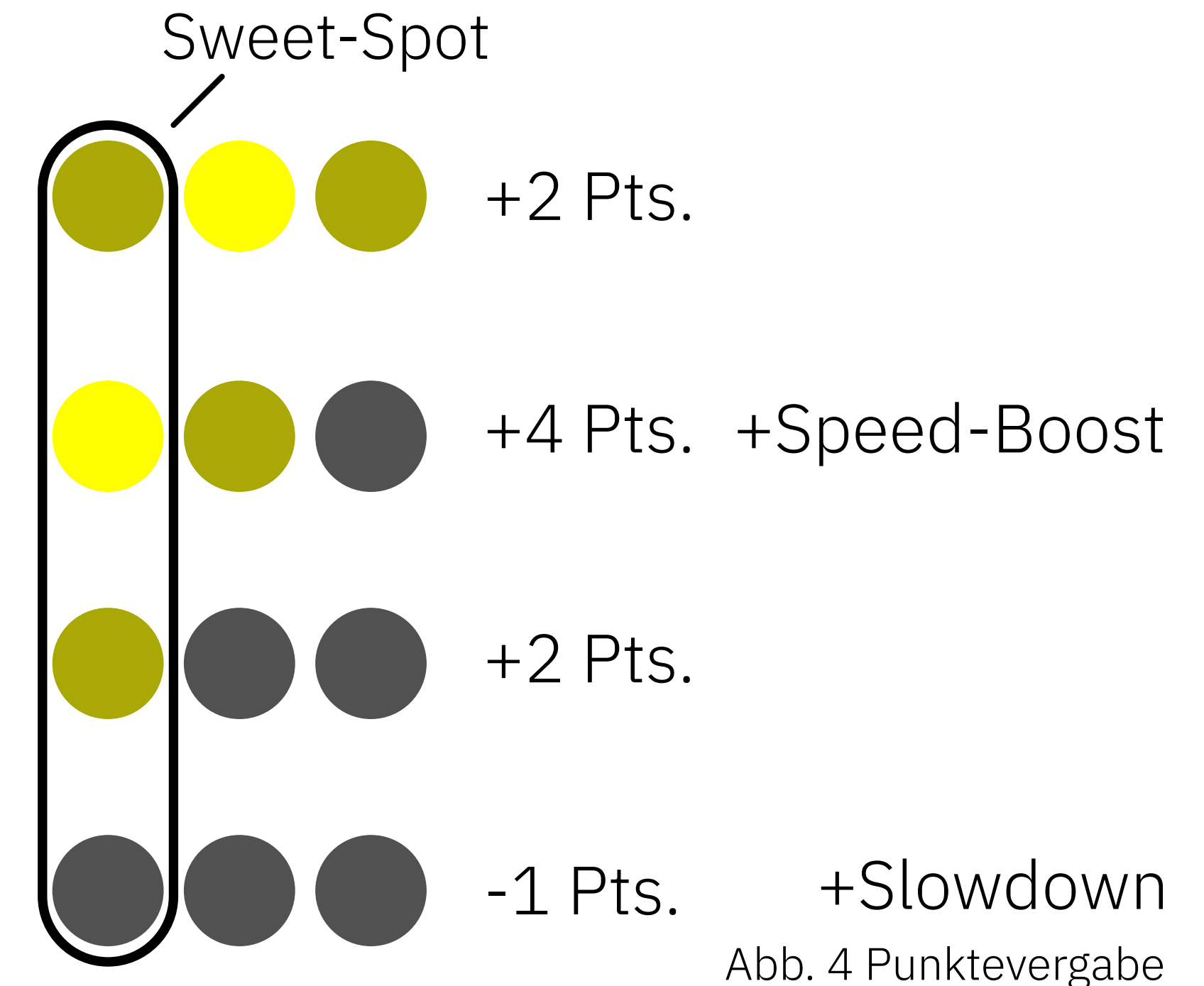
Abb. 4 Zeigt alle Möglichkeiten wenn der Knopf Gedrückt wurde

Den Sweet-Spot mit der Hellen LED zu Treffen gibt 4 Punkte und die Bewegungsgeschwindigkeit der LED wird erhöht wodurch es Schwieriger wird den Sweet-Spot mit der hellen LED zu treffen.

Trifft man den Sweet-Spot mit einer schwächeren LED erhält man zumindest 2 Punkte, jedoch bleibt die Geschwindigkeit unverändert.

Trifft man überhaupt nicht bekommt man 1 Punkt abgezogen und die Geschwindigkeit verringert sich, wodurch der Gegenspieler leichter Treffen kann.

Wenn man versucht den Knopf zu drücken währen die LED sehr weit entfernt vom Sweet-Spot ist. Bekommt man 2 Punkte abgezogen.



Projektbeschreibung

UI

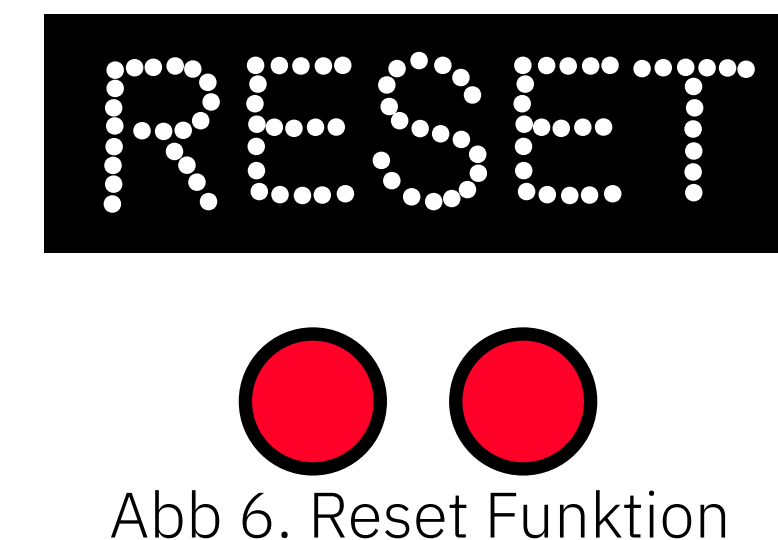
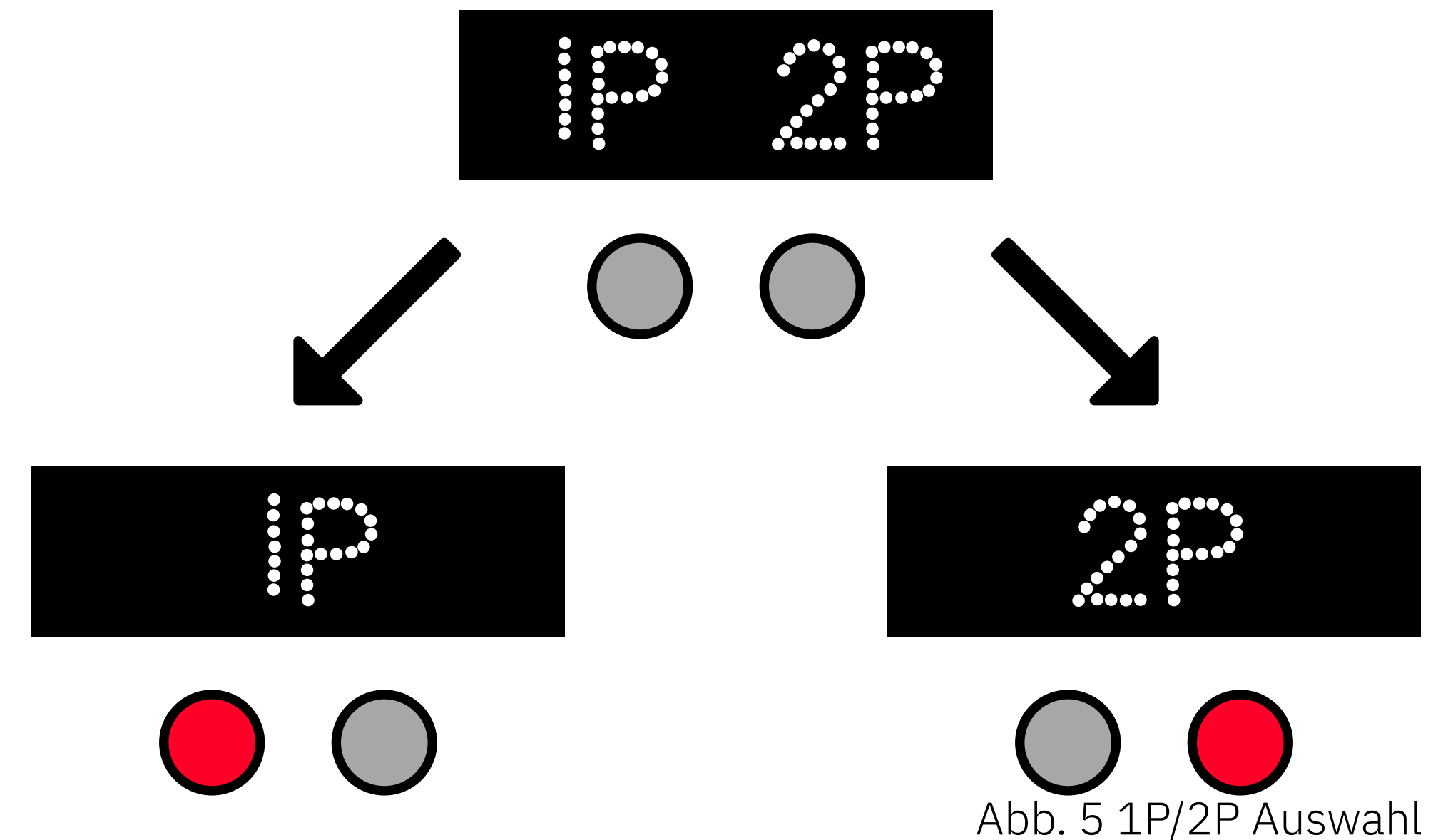
Beim einschalten des Gerätes sieht man sofort die Auswahl ob man den Einzelspielermodus oder den Mehrspielermodus spielen möchte.

Die beiden Knöpfe die unter der Spielfläche angebracht sind dienen zur Auswahl.

Mit dem linken Knopf startet man den Einzelspielermodus (*ANMERKUNG: In diesem Konzept fokussieren wir uns auf den Mehrspielermodus*)

Mit dem rechten Knopf den Mehrspielermodus.

Wenn beide Knöpfe gleichzeitig für 3 Sekunden gedrückt gehalten werden, egal wann, kann das Gerät zurückgesetzt werden um die oben beschriebene Auswahl erneut zu treffen.



Projektbeschreibung

UI - Multiplayer

Nachdem man „2P“ ausgewählt hat wählt man die Zielpunktzahl vom Match aus.

Durch das Drücken der Knöpfe die unter dem Spielfeld angebracht sind kann man die Zielpunktzahl entweder erhöhen oder verringern.

Standardmäßig sieht man zuerst 30.

Wenn man seine Auswahl getroffen hat drückt man einer der 8 Tasten an der Seite um zur zweiten Auswahl zu kommen.

Die nächste LED an jedem Knopf pulsiert um den User dazu zu bewegen einer der 8 Knöpfe zu drücken.

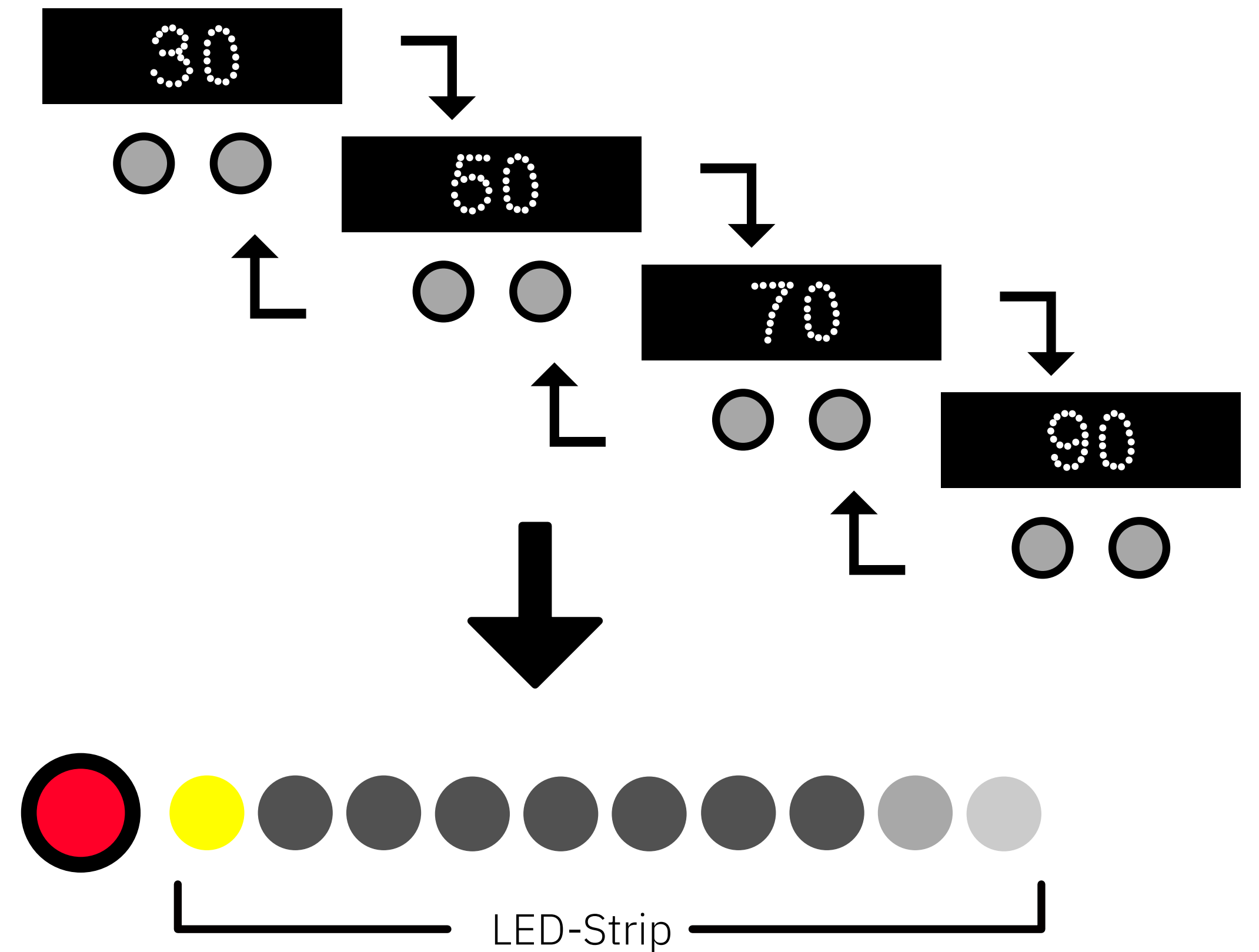


Abb 7. Spielpunkte Einstellung

Projektbeschreibung

UI - Multiplayer

Nachdem man die Zielpunktzahl gewählt hat entscheidet man sich für die Spielgeschwindigkeit.

Durch das Drücken der Knöpfe die unter dem Spielfeld angebracht sind kann man die Schwierigkeit entweder erhöhen oder verringern.

Standardmäßig sieht man zuerst „normal“.

Wenn man seine Auswahl getroffen hat kann das Spiel gestartet werden durch einer der 8 Button an der Seite des Spielfeldes.

Jede nächste LED von einem Knopf pulsiert um sich „Spielbereit“ zu zeigen und den User dazu zu bewegen einer der 8 Buttons zu drücken.

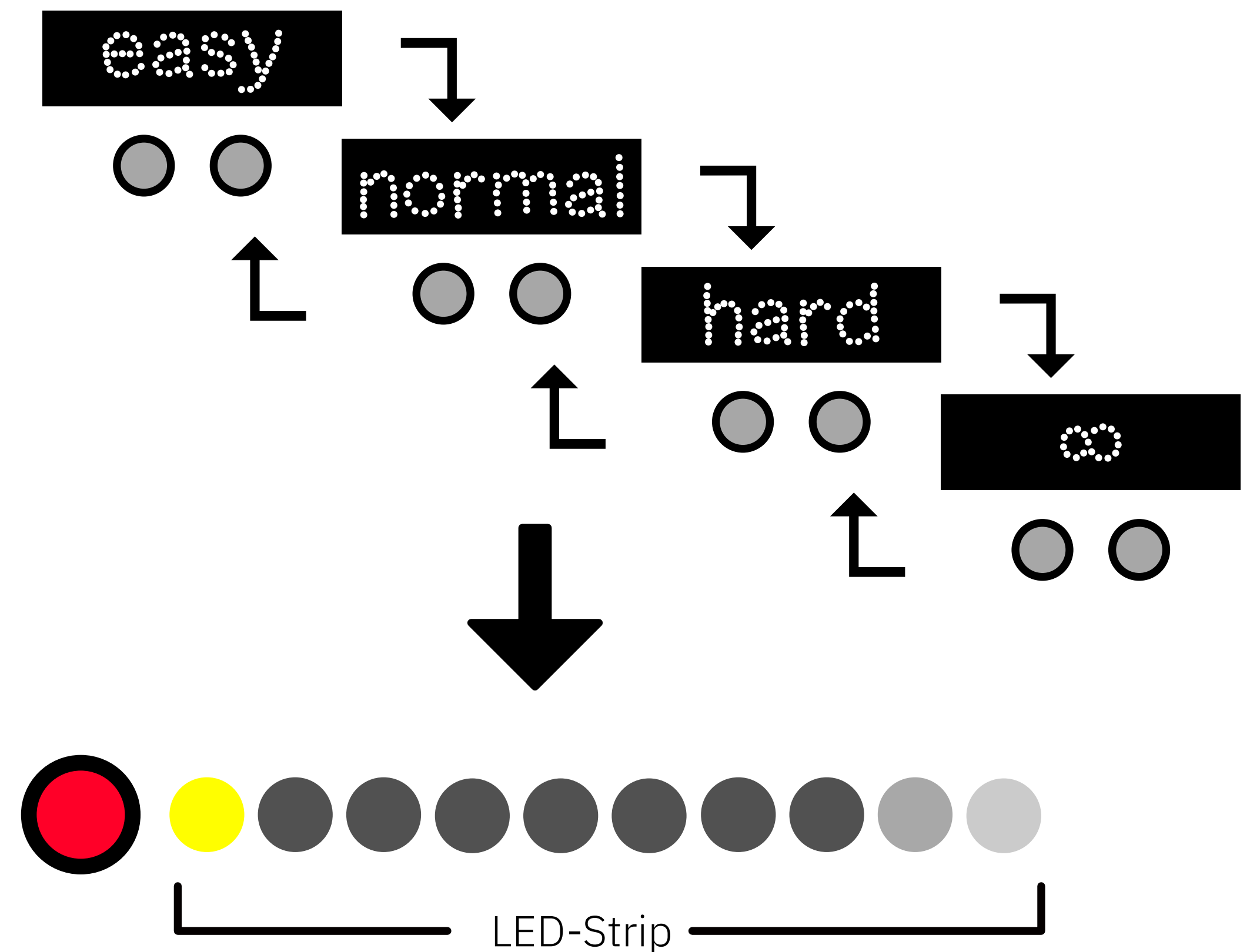
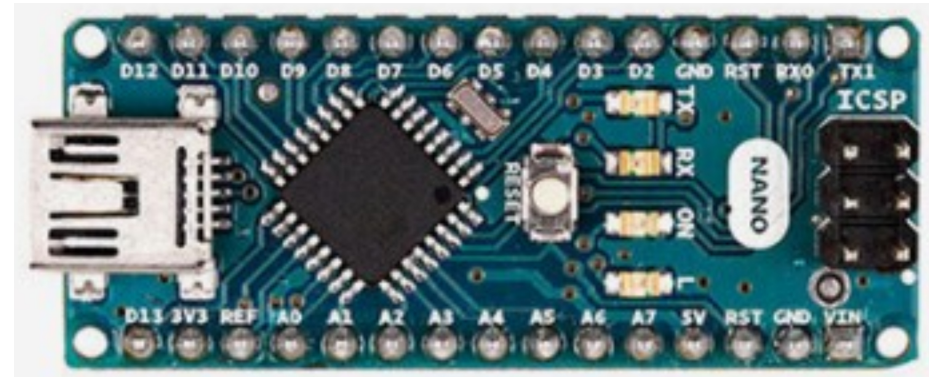


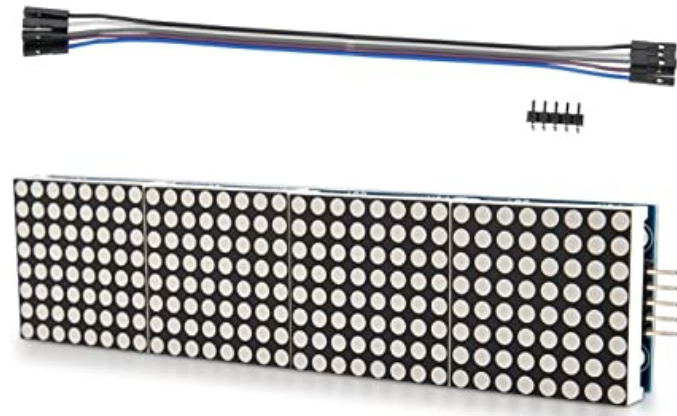
Abb 8. Einstellung Schwierigkeit und Spielstart

Komponenten

Liste der Teile



1x Arduino Nano



1x 8x32 LED
Matrix Module



8x Cherry
MX Red



1m WS2815 12V
RGB Led Streifen
(144 LEDs pro Meter)



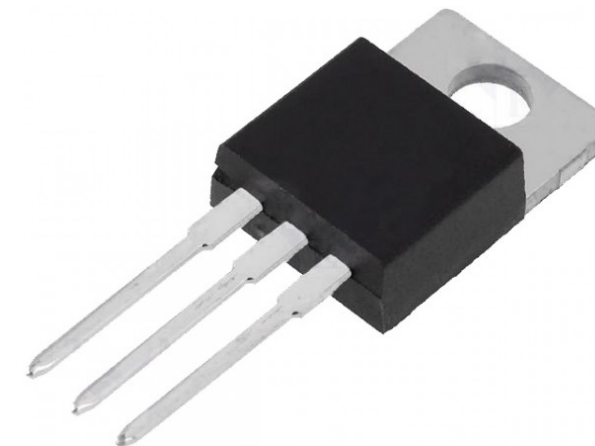
Widerstände
4x 10k Ω
4x 150 Ω
4x 220 Ω



Kontensatoren
1x 0,1 mF
1x 1 μ F
1x 1mF



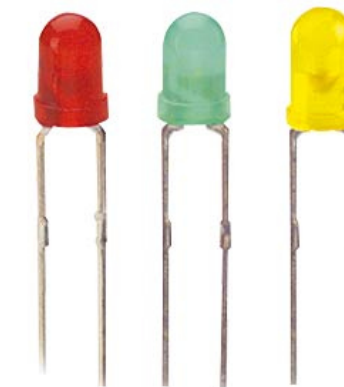
1x Diode



1x 7805
Spannungsregler



1x 12v Netzteil



3mm LEDs
4x Blaue LEDs
4x Rote LEDs



2x Taster

Komponenten

Codeschnipsel

```
//Beispiel für das Ansteuern von einzelnen LEDs

#include <Adafruit_NeoPixel.h> //Neopixel Library
#define PIN 5 // Digitale Pin zu Ansteuerung der LED Stripe
#define NUMPIXELS 76 // Anzahl der LEDs von der LED Stripe
Adafruit_NeoPixel pixels = Adafruit_NeoPixel(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);
//Anzahl der Pixel, Ansteuerungs-PIN, Art des LED-Streifen (bei uns Neopixel WS2815)

int pause=100;

void setup()
{
  pixels.begin(); // Initialisierung der NeoPixel
}

void loop()
{
  pixels.setPixelColor(1, pixels.Color(0,255,0)); //LED 1 Farbe
  pixels.show(); //LED 1 leuchten lassen
  delay (pause); //Pause

  pixels.setPixelColor(2, pixels.Color(0,150,0)); //LED 2 Farbe
  pixels.show(); //LED 2 leuchten lassen
  delay (pause); //Pause

  //...
}
```

LED-Stripe

```
#define SP1 7
#define SP2 8
#define Button1 9
#define Button2 10
#define Button3 11
#define Button4 12

void setup() {
  pinMode(SP1, OUTPUT);
  pinMode(SP2, OUTPUT);
  pinMode(Button1, INPUT);
  pinMode(Button2, INPUT);
  pinMode(Button3, INPUT);
  pinMode(Button4, INPUT);
}

void loop() {
  int activeButtons[8];
  for(int p = 0; p <= 1; p++){
    if (p = 1) {
      digitalWrite(SP1, HIGH);
    } else {
      digitalWrite(SP2, HIGH);
    }
    for(int i = 0; i <= 3; i++){
      if(digitalRead(i + 9) == HIGH){
        activeButtons[p * 4 + i] = 1;
      }
      else{
        activeButtons[p * 4 + i] = 0;
      }
    }
    digitalWrite(SP1, LOW);
    digitalWrite(SP2, LOW);
  }
}
```

Tasten Array

```
#include <Arduino.h>
#include <LedControl.h>

int figure[8] = {0b00011000,
                0b00100100,
                0b01000010,
                0b01000010,
                0b10000001,
                0b10000001,
                0b01000010,
                0b00111100}; // Ein Osterei!

#define DIN 2
#define CS 3
#define CLK 4

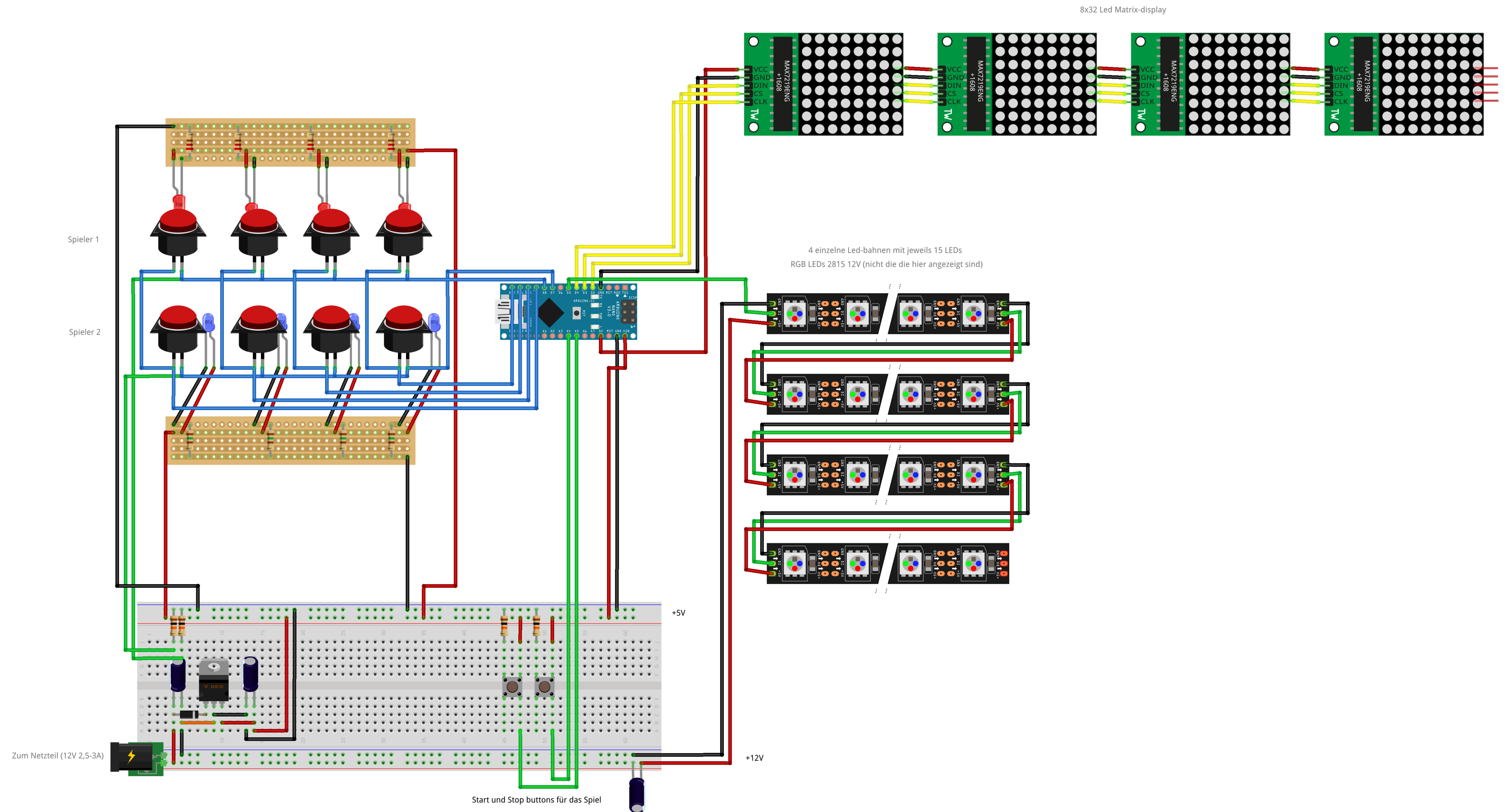
LedControl lc = LedControl(DIN, CLK, CS, 1); // 1 = Anzahl LED-Matrizen

void setup() {
  lc.shutdown(0, false); // Aufwachen (shutdown = false), 0 = index für erste LED-Matrix
  lc.setIntensity(0, 2); // Helligkeit Dots (0 = index des geräts, 2 = helligkeit)
  lc.clearDisplay(0); // Alle Dots ausschalten (0 = index des Geräts)
}

void loop() {
  for(int row = 0; row < 8; row++) {
    lc.setRow(0, row, figure[row]);
  }
}
```

LED-Matrix

Fritzing Schaltplan



Pseudo Code

Anmerkung: Siehe ebenfalls linear_pong_pseudocode.txt

```

ButtonLinks = A5;
ButtonRechts = A4;

buttonLinksgedrueckt = false;
buttonRechtsgedrueckt = false;

Spielerbuttons = {12, 11, 10, 9};
Spieler1buttonOutput = 8;
Spieler2buttonOutput = 7;

// hat das Spiel gestartet, oder ist man noch im Menü?
playing = false;
// hat jemand gewonnen?
won = false;
// in welcher Menüeinstellung man sich befindet
menu = "Spielerauswahl";

// 0: Einfach (nur eine Spur), 1: mittel (2 Spuren), 2: schwer (3 Spuren), 4: unmöglich (4 Spuren)
Schwierigkeit = 0;

// Die Anzahl Spieler, 1 oder 2, Spieler 1 ist links, Spieler 2 rechts
Spieleranzahl = 2;
Spieler1Punkte = 0;
Spieler2Punkte = 0;

// Die maximale Punkteanzahl bei 2 Spielern, mit der man gewinnt
PunktLimit = 30;

// Zeit zwischen 2 Ticks in ms
tickDelay = 20;
lastTickTime = 0;
currentTime = 0;

// Button Links und Button Rechts gedrückt halten resettet das Spiel
resetPressTime = 3000;
currentTimePressed = 0;
startPressTime = 0;

// Geschwindigkeit in leds pro tick mit Richtung
Bewegungen = {1, 1, 1, 1};

Startpositionen = {10, 10, 10, 10};
positionen = Startpositionen;

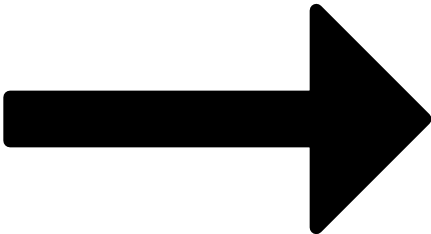
interruptButtonLinks() {
  delay(50);
  if ButtonLinks gedrückt {
    buttonLinksgedrueckt = true;
  } else {
    buttonLinksgedrueckt = false;
  }
}

interruptButtonRechts() {
  delay(50);
  if ButtonRechts gedrückt {
    buttonRechtsgedrueckt = true;
  } else {
    buttonRechtsgedrueckt = false;
  }
}

updateSpurPosition(Spur, position) {
  if position >= 0 && position <= 18 {
    // Diese Zeile setzt die Farbe der richtigen LED
    Neopixel.setPixelColor(Spur * 19 + 1 + position, Neopixel.Color(59, 194, 218)); // Hellblau
  }
  // Die LED davor
  if (position - 1) >= 0 && (position - 1) <= 18 {
    Neopixel.setPixelColor(Spur * 19 + position, Neopixel.Color(226, 200, 63)); // Gelb
  }
  // Die LED danach
  if (position + 1) >= 0 && (position + 1) <= 18 {
    Neopixel.setPixelColor(Spur * 19 + 2 + position, Neopixel.Color(226, 200, 63)); // Gelb
  }
}

getPressedPlayerButtons() {
  int activeButtons[8];
  for(int p = 0; p <= 1; p++){
    if (p == 1) {
      digitalWrite(Spieler2buttonOutput, HIGH);
    } else {
      digitalWrite(Spieler1buttonOutput, HIGH);
    }
    for(int i = 0; i <= 3; i++){
      if(digitalRead(i + 4) == HIGH){
        activeButtons[p * 4 + i] = 1;
      }
    }
    if (p == 1) {
      digitalWrite(Spieler2buttonOutput, LOW);
      digitalWrite(Spieler1buttonOutput, LOW);
    }
  }
  return activeButtons;
}

addPointsToPlayer(player, points) {
  if player == 1 {
    Spieler1Punkte += points;
  }
}
```



```

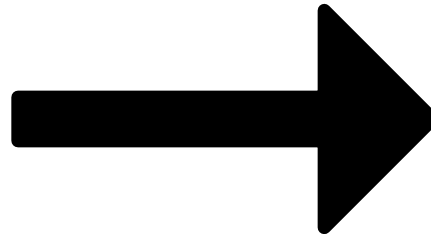
} else if player == 2 {
  Spieler2Punkte += points;
}
}

resetSpur(spur) {
  random = random(-1, 1); // Zufälliges Vorzeichen
  if random < 0 {
    Bewegungen[spur] = -1;
  } else {
    Bewegungen[spur] = 1;
  }
  positionen[spur] = Startpositionen[spur];
}

setup() {
  for button in Spielerbuttons {
    pinMode(button, INPUT);
  }
  pinMode(Spieler1buttonOutput, OUTPUT);
  pinMode(Spieler2buttonOutput, OUTPUT);
  pinMode(ButtonRechts, INPUT);
  pinMode(ButtonLinks, INPUT);
  attachInterrupt(digitalPinToInterrupt(ButtonLinks), interruptButtonLinks, CHANGE);
  attachInterrupt(digitalPinToInterrupt(ButtonRechts), interruptButtonRechts, CHANGE);
  // zufällige Richtungen zum Anfang
  for Bewegung in Bewegungen {
    random = random(-1, 1); // Zufälliges Vorzeichen
    if random < 0 {
      Bewegung = -1;
    }
  }

  // LED-Streifen initialisieren
  Neopixel.begin();
}

loop() {
  pressedSpielerButtons = getPressedPlayerButtons();
  if buttonLinksgedrueckt && buttonRechtsgedrueckt {
    if startPressTime == 0 {
      startPressTime = currentTime;
    } else {
      timeDiff = currentTime - startPressTime;
      currentTimePressed = timeDiff;
    }
  } else if nicht buttonLinksgedrueckt && nicht buttonRechtsgedrueckt {
    startPressTime = 0;
    currentTimePressed = 0;
  }
  if currentTimePressed >= resetPressTime {
    // button links und button rechts wurden für 3 sec gedrückt
    playing = false;
    menu = "Spielerauswahl";
    Bewegungen = {1, 1, 1, 1};
    for Bewegung in Bewegungen {
      random = random(-1, 1); // Zufälliges Vorzeichen
      if random < 0 {
        Bewegung = -1;
      }
    }
    positionen = Startpositionen;
  }
  if nicht playing {
    if menu == "Spielerauswahl" {
      Zeige "1P - 2P" auf LEDmatrix an;
      if buttonLinksgedrueckt {
        Spieleranzahl = 1;
        buttonLinksgedrueckt = false;
      } else if buttonRechtsgedrueckt {
        Spieleranzahl = 2;
        buttonRechtsgedrueckt = false;
      }
      if eine der Spielertasten gedrueckt {
        if Spieleranzahl == 2 {
          menu = "Punktauswahl";
        } else {
          menu = "Schwierigkeitsauswahl";
        }
      }
    } else if menu == "Punktauswahl" {
      Zeige "max: " + PunktLimit + " pts" auf LEDmatrix an;
      if buttonLinksgedrueckt und nicht buttonRechtsgedrueckt {
        if PunktLimit == 30 {
          PunktLimit = 90;
        } else {
          PunktLimit -= 20;
        }
        buttonLinksgedrueckt = false;
      }
      if buttonRechtsgedrueckt und nicht buttonLinksgedrueckt {
        if PunktLimit == 90 {
          PunktLimit = 30;
        } else {
          PunktLimit += 20;
        }
        buttonRechtsgedrueckt = false;
      }
      if eine der Spielertasten gedrueckt {
        menu = "Schwierigkeitsauswahl";
      }
    } else if menu == "Schwierigkeitsauswahl" {
      if Schwierigkeit == 0 {
        Zeige "easy" auf LEDmatrix an;
      } else if Schwierigkeit == 1 {
        Zeige "normal" auf LEDmatrix an;
      } else if Schwierigkeit == 2 {
        Zeige "hard" auf LEDmatrix an;
      }
    }
  }
}
```



```

} else if Schwierigkeit == 3 {
  Zeige "s" auf LEDmatrix an;
}
}
if buttonLinksgedrueckt und nicht buttonRechtsgedrueckt {
  if Schwierigkeit == 0 {
    Schwierigkeit = 3;
  } else {
    Schwierigkeit -= 1;
  }
  buttonLinksgedrueckt = false;
}
if buttonRechtsgedrueckt und nicht buttonLinksgedrueckt {
  if Schwierigkeit == 3 {
    Schwierigkeit = 0;
  } else {
    Schwierigkeit += 1;
  }
  buttonRechtsgedrueckt = false;
}
if eine der Spielertasten gedrueckt {
  menu = "Spielerauswahl";
  playing = true;
}
}
} else {
  Zeige Spieler1Punkte + " : " + Spieler2Punkte auf LEDmatrix an;
  // leds von 1 bis 18 sind die erste Spur
  // leds von 19 bis 38 sind die 2te Spur
  // leds von 39 bis 57 sind die 3te Spur
  // leds von 58 bis 76 sind die 4te Spur
  if currentTime >= lastTickTime + tickDelay {
    lastTickTime = currentTime;
    // resettet das Spielfeld
    if not won {
      Neopixel.clear();
      for (int spur = 0; spur <= Schwierigkeit; spur++) {
        positionen[spur] += Bewegungen[spur];
        updateSpurPosition(spur, positionen[spur]);
        if positionen[spur] == 18 {
          if pressedSpielerButtons[4 + spur] == 1 {
            // Spieler 2 trifft genau in die Mitte
            Bewegungen[spur] = -2;
            addPointsToPlayer(2, 4);
          }
        } else if positionen[spur] == 17 {
          if pressedSpielerButtons[4 + spur] == 1 {
            // Spieler 2 trifft kurz davor
            Bewegungen[spur] = -1;
            addPointsToPlayer(2, 2);
          }
        }
        } else if positionen[spur] == 19 {
          if pressedSpielerButtons[4 + spur] == 1 {
            // Spieler 2 trifft kurz danach
            Bewegungen[spur] = -1;
            addPointsToPlayer(2, 2);
          }
        }
        } else if positionen[spur] == 0 {
          if pressedSpielerButtons[spur] == 1 {
            // Spieler 1 trifft genau in die Mitte
            Bewegungen[spur] = 2;
            addPointsToPlayer(1, 4);
          }
        }
        } else if positionen[spur] == 1 {
          if pressedSpielerButtons[spur] == 1 {
            // Spieler 1 trifft kurz davor
            Bewegungen[spur] = 1;
            addPointsToPlayer(1, 2);
          }
        }
        } else if positionen[spur] == -1 {
          if pressedSpielerButtons[spur] == 1 {
            // Spieler 1 trifft kurz danach
            Bewegungen[spur] = 1;
            addPointsToPlayer(1, 2);
          }
        }
        } else if positionen[spur] < -1 {
          // Wenn das Licht an Spieler 1 vorbei das Spielfeld verlässt
          addPointsToPlayer(1, -1);
          resetSpur(spur);
        } else if positionen[spur] > 19 {
          // Wenn das Licht an Spieler 2 vorbei das Spielfeld verlässt
          addPointsToPlayer(2, -1);
          resetSpur(spur);
        } else {
          // Wenn man drückt, aber das Licht nirgends in der Nähe ist, gibts Punkteabzug
          if pressedSpielerButtons[spur] == 1 {
            addPointsToPlayer(1, -2);
          }
          if pressedSpielerButtons[4 + spur] == 1 {
            addPointsToPlayer(2, -2);
          }
        }
      }
    }
  }
  // Win check
  if Spieler1Punkte >= PunktLimit {
    Zeige "player 1 wins" auf LEDmatrix an;
    won = true;
  } else if Spieler2Punkte >= PunktLimit {
    Zeige "player 2 wins" auf LEDmatrix an;
    won = true;
  }
}
}
```

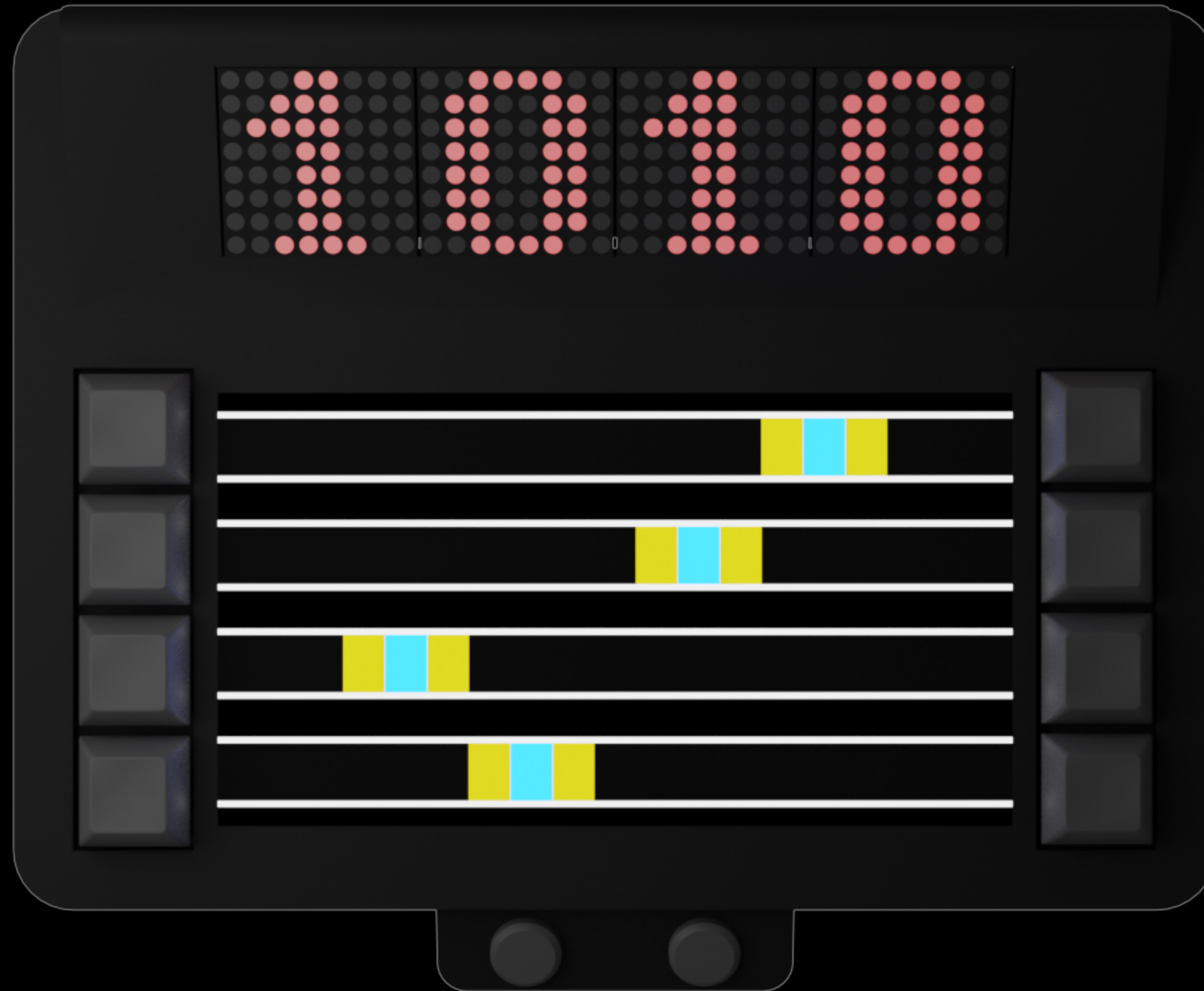
3D-Render

Perspektive 1



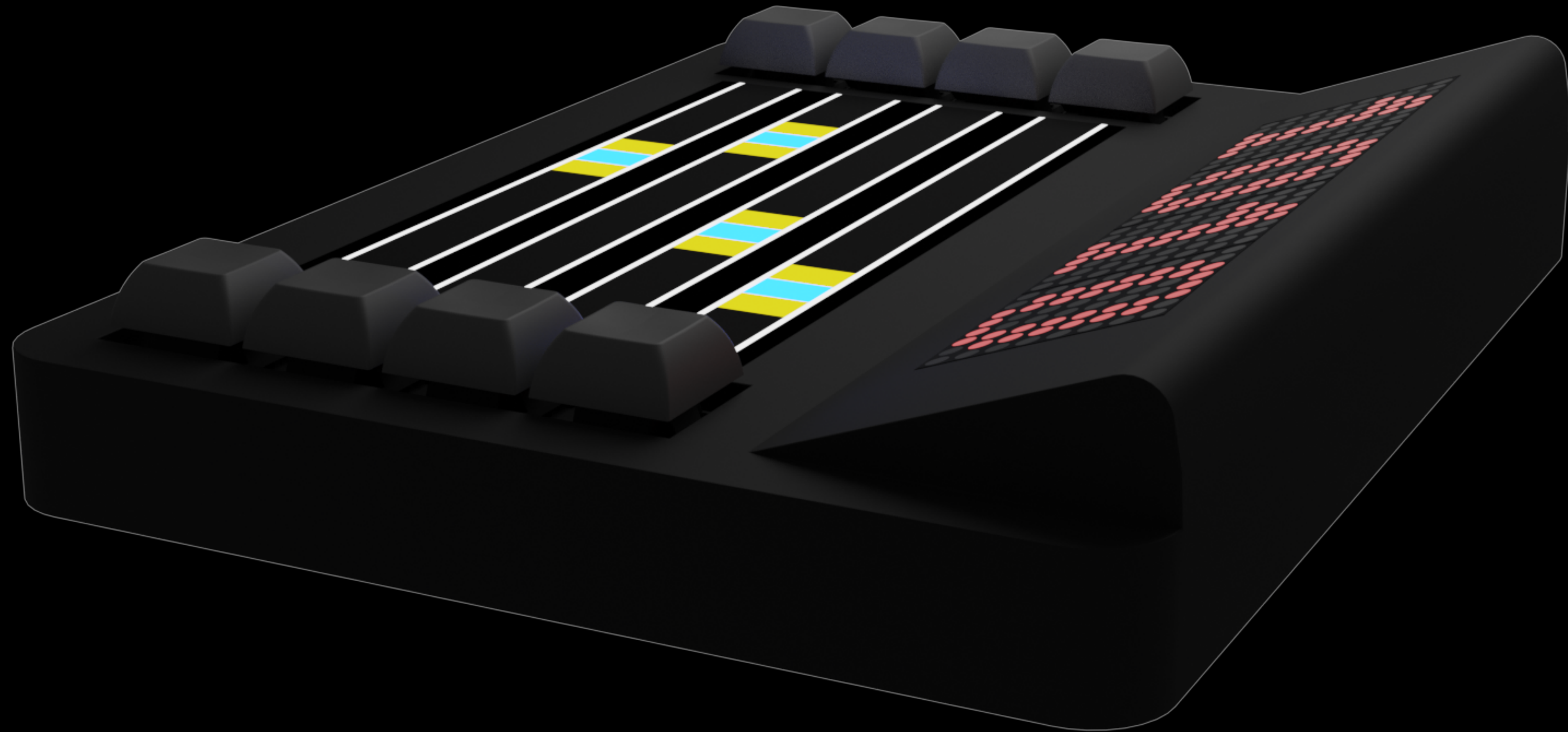
3D-Render

Top View



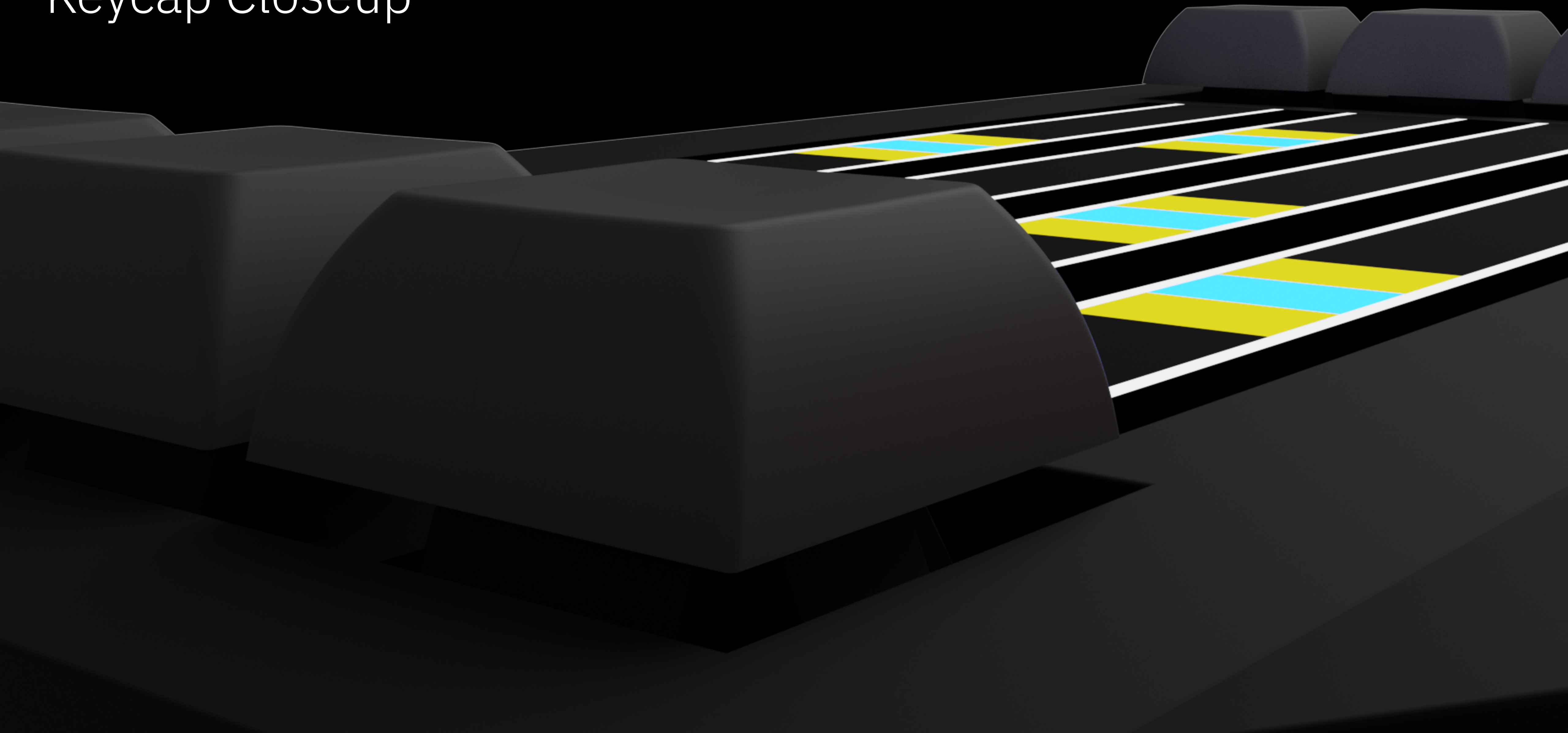
3D-Render

Perspektive 2



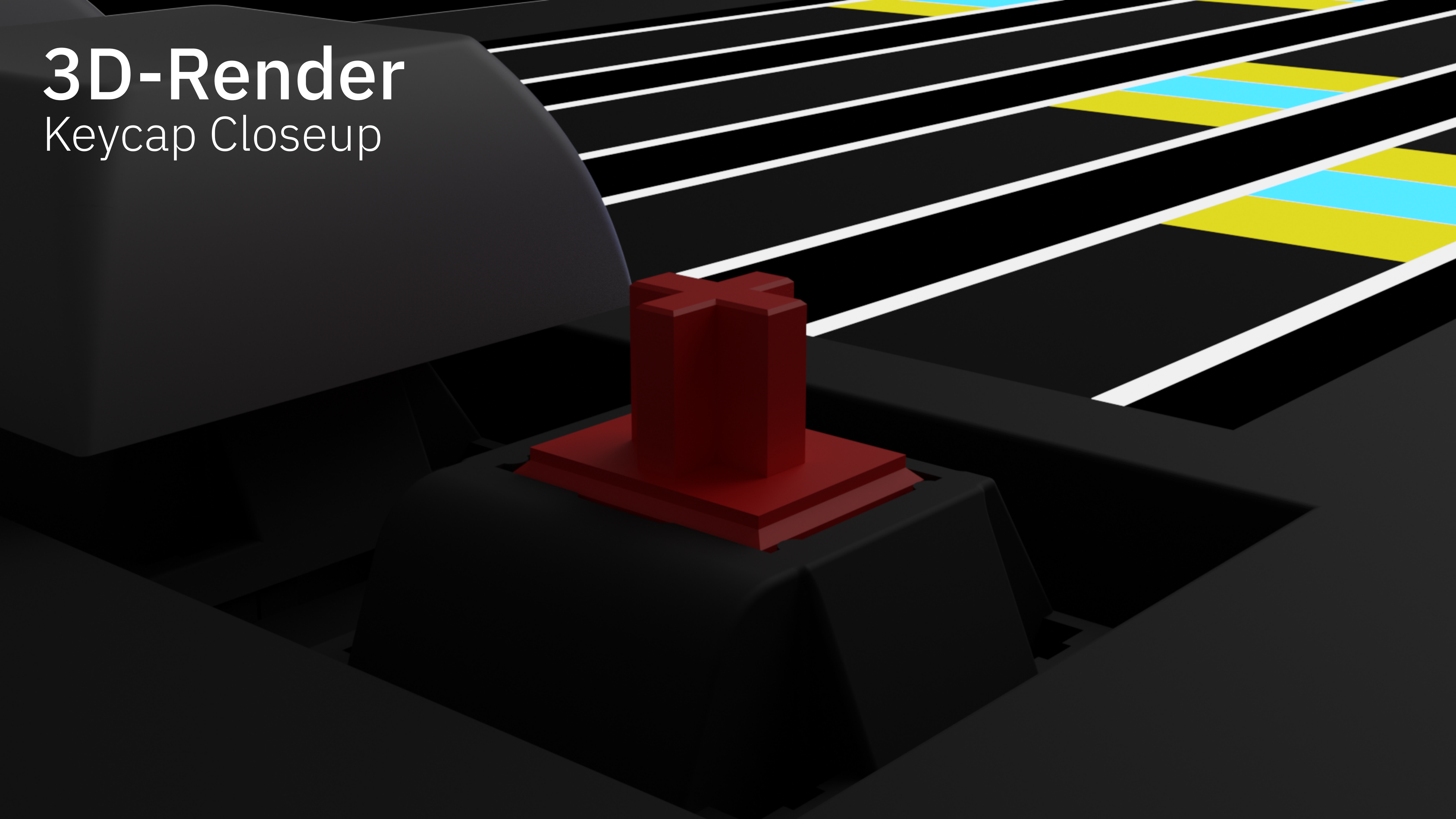
3D-Render

Keycap Closeup



3D-Render

Keycap Closeup



3D-Render

Limited Edition

