Gymnasium Thun

Informatikprojekt

Arduino Projekt - Snake

Luca Mezger

eingereicht bei

André Weyermann

17. Dezember 2022

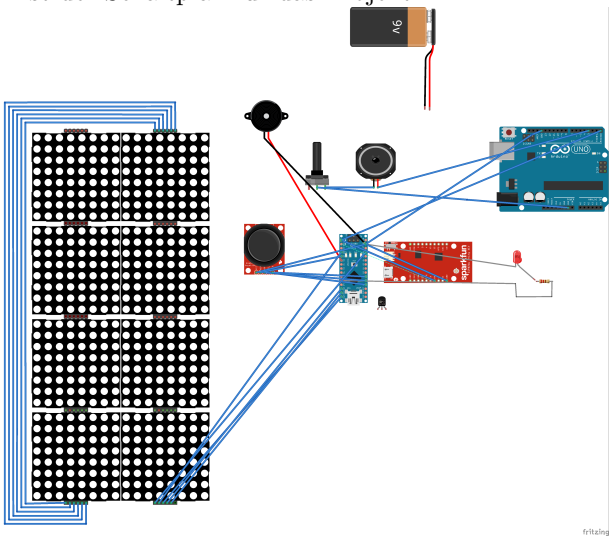
Zusammenfassung

Mein Ziel war es, Snake auf einem Arduino Nano zu erstellen. Das ganze wollte ich auf 2 8*32 Matrizen machen. Gesteuert sollte das ganze mit einem Joystick werden. Dazu kommt ein WiFI Modul, mit dem der aktuelle Score + Highscore automatisch auf die für alle ersichtliche Website lucamezger.com/arduino hochgeladen wird (Mehr dazu im passenden Abschnitt). Dazu wollte ich eine passende und steuerbare (Lautstärke) Musik abspielen(auf einem Dual-Speaker Setup aus einem alten Laptop). Auch ein "Rage Quit Sensor"durfte nicht fehlen. Das ist ein Sensor, der das Spiel sofort beendet, wenn jemand zu laut schreit. Die Helligkeit der Matrizen wird automatisch angepasst, je nach dem wie hell es ist. Damit die ganze Elektronik nicht überhitzt habe ich einen absolut nötigen Fan eingebaut. Schlussendlich um das ganze tragbar zu machen habe ich eine Batterie rangehängt.

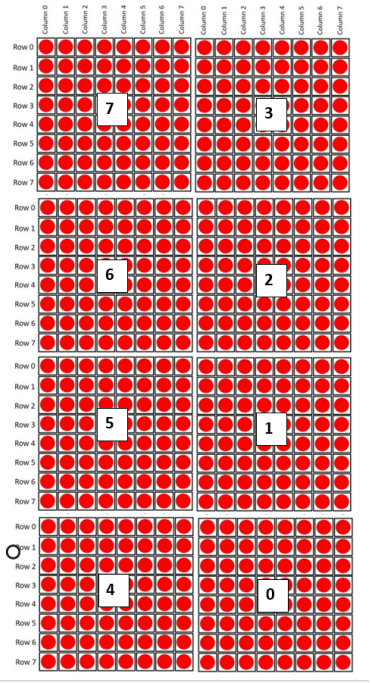
Inhaltsverzeichnis

1 Hardware / Circuit

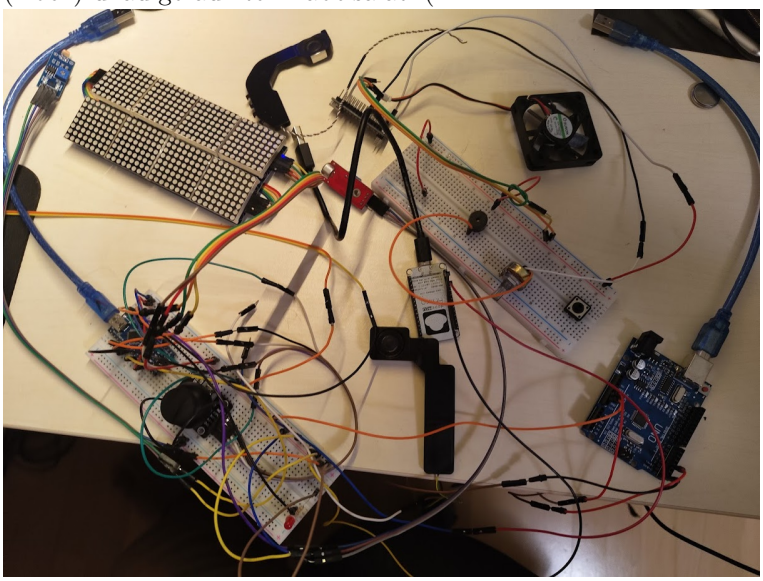
Hier ist der Schaltplan für das Projekt.



Die Matrizen sind wie folgt angeordnet:



(Noch) unaufgeräumter Kabelsalat :(



2 Arduino Nano - Spiel

2.1 Info

Die Kommentare/Erklärungen im Code sind je nach dem auf Englisch oder Deutsch. Es kommt darauf an, wann ich den Code geschrieben hatte und wie Lust hatte die Kommentare zu verfassen ;)

Zuerst noch zu den globalen Variabeln die definiert wurden.

```
int pushButton = A2; //Pushbutton of joystick to change start scences
int digital_mic = 5; //Pin of Mic --> Rage quit sensor
int snakeX = 16;  //start the snake **HEAD** in the middle --> iNumMatrixRows * 8 / 2;
int snakeY = 8;   //Y position of snakes head
int preX;         //to add this LED if the snake has grown by one LED
int preY;
//Array for whole snake --> every part from head to tail
int snake[200][2] = { { snakeX, snakeY } };  //200 is the max length for the snake --> theoretically would be 16*32 = 512 possible
char direction = 'l'; //to which direction the snake is facing to --> l,r,u or d
int foodX;  //the position of the food/apple
int foodY;
int score = 0;          //score & is also equal to the snake length (score + 1 = length)
bool foodState = true;  //if Food should be turned on or off --> for flickering effect
```

```
14
15
16
17 const int flickerFrequencyFood = 50;                    // flicker
       frequency in Hz
18 const int delayTimeFood = 1000 / flickerFrequencyFood;  // delay
       time in ms
19 unsigned long lastFlickerTimeFood = 0;                  // variable
        to track the last time the Food LED was flicked
20
21 int updateFrequencySnake = 2;                          // flicker
       frequency in Hz
22 int delayTimeSnake = 1000 / updateFrequencySnake;  // delay time in
        ms
23 unsigned long lastUpdateTimeSnake = 0;                // variable to
       track the last time the Snake position was updated
```

2.2 Laufschrift

Die Laufschrift wurde mit der Library LedCotrol erstellt. Diese Laufschrift ist nicht effizient, aber man kann sie beliebig anpassen, wie man es gerne möchte. Als erstes muss man die Library importieren und initialisieren.

```
1 #include "LedControl.h"  //import Library for scrolling text
2 LedControl lc = LedControl(12, 11, 10, 8);  // initialize for
      scrolling text
```

Als nächstes werden die Buchstaben, die gebraucht werden definiert. Das habe ich so gemacht, dass ich sie mit setRow / setColumn setzen kann. Das heisst konkret pro Buchstabe braucht es 8 Bytes. Wie das genau funktioniert können Sie gerne der Dokumentation für LedControl entnehmen. Diese 8 Bytes habe ich in Hexadezimal geändert um Platz zu sparen (übersichtlicher).

```
1 byte A[8] = { 0x0, 0x18, 0x24, 0x42, 0x7e, 0x42, 0x42, 0x0 };  //
      alle Buchstaben von bin r zu hexadezimal umge ndert, wegen
      Platz
2 byte B[8] = { 0x0, 0x7c, 0x42, 0x42, 0x7c, 0x42, 0x42, 0x7c };
3 byte E[8] = { 0x0, 0x7e, 0x40, 0x40, 0x7c, 0x40, 0x40, 0x7e };
4 byte K[8] = { 0x0, 0x44, 0x48, 0x50, 0x60, 0x50, 0x48, 0x44 };
5 byte L[8] = { 0x0, 0x40, 0x40, 0x40, 0x40, 0x40, 0x40, 0x7c };
6 byte M[8] = { 0x0, 0x42, 0x66, 0x5a, 0x42, 0x42, 0x42, 0x42 };
7 byte N[8] = { 0x0, 0x42, 0x62, 0x72, 0x5a, 0x4e, 0x46, 0x42 };
8 byte S[8] = { 0x0, 0x3e, 0x40, 0x40, 0x3c, 0x2, 0x2, 0x7c };
9 byte Y[8] = { 0x0, 0x44, 0x44, 0x28, 0x10, 0x10, 0x10, 0x10 };
10 byte null[8] = { 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0 };
```

Die nächsten 4 Arrays sind um die LEDs, welche angezeigt werden zu speichern.

```
1 byte first_start_text_0to3[32] = {}; //Der erste Teil (Snake), die
      Matrixen 0 bis 3
2 byte second_start_text_0to3[32] = {}; //Der erste Teil (Snake), die
       Matrixen 4 bis 7
3 byte first_start_text_4to7[32] = {};//Der zweite Teil (By LM), die
      Matrixen 0 bis 3
```

```
byte second_start_text_4to7[32] = {};//Der zweite Teil (By LM), die
    Matrixen 4 bis
```

Im Setup werden dann die Buchstaben zu den oben definierten Arrays gesetzt. So kann man alles immer selber einstellen (Also die Buchstaben einfach wechseln). Das +8 ist für die Matrix unter dran, das +16 für die Matrix 2 unten dran und so weiter. Die For-Loops sind um alle 8 Bytes für einen Buchstaben zu setzen, nicht nur eine Reihe.

```
//Set the letters to the array
  for (int i = 0; i < 8; i++) {
    first_start_text_0to3[i] = S[i];
    first_start_text_0to3[i + 8] = A[i];
    first_start_text_0to3[i + 16] = E[i];
    first_start_text_0to3[i + 24] = null[i];
  }

  for (int i = 0; i < 8; i++) {
    second_start_text_0to3[i] = B[i];
    second_start_text_0to3[i + 8] = null[i];
    second_start_text_0to3[i + 16] = L[i];
    second_start_text_0to3[i + 24] = null[i];
  }

  //Matrixes 4 to 7
  for (int i = 0; i < 8; i++) {
    first_start_text_4to7[i] = N[i];
    first_start_text_4to7[i + 8] = K[i];
    first_start_text_4to7[i + 16] = null[i];
    first_start_text_4to7[i + 24] = null[i];
  }

  for (int i = 0; i < 8; i++) {
    second_start_text_4to7[i] = Y[i];
    second_start_text_4to7[i + 8] = null[i];
    second_start_text_4to7[i + 16] = M[i];
    second_start_text_4to7[i + 24] = null[i];
  }
```

Um den Text anzuzeigen und hochzuschieben brauchen wir diesen Code. Diese Funktion wird wieder aufgerufen (siehe weiter unten). Dieser Code zeigt die Leds, schiebt alle eines nach oben, zeigt sie wieder und so weiter. Die Erklärungen dazu sind als Kommentare drin.

```
void first_start_text() {
  byte col_0; //um sich die erste Kolonne zu merken (f r Matrixen
    1 bis 3)
  byte col_01; //f r Matrixen 4 bis 7
  for (int y = 0; y < 32; y++) { //Loope durch alle Kolonnen durch
    und setze sie an.
    lc.setColumn(y / 8, 7 - (y % 8), first_start_text_0to3[y]);  //
     Bsp: y=10; y/8=1; 7-y%8=5
    lc.setColumn(y / 8 + 4, 7 - (y % 8), first_start_text_4to7[y]);
  }

  // Laufschrift nach oben schieben
```

```
  col_0 = first_start_text_0to3[0];  // Die erste Kolonne merken, da sie wieder unten eingef gt werde muss
  col_01 = first_start_text_4to7[0]; //das gleiche f r Matrixen 4-7
  for (int y = 0; y < 31; y++) {  // nur bis 30 resp. 31, da unten die letzte Kolonne neu zugeordnet wird
    first_start_text_0to3[y] = first_start_text_0to3[y + 1]; // shciebe alles +1 noch oben
    first_start_text_4to7[y] = first_start_text_4to7[y + 1];
  }
  first_start_text_0to3[31] = col_0;   // Hier die erste Kolonne am Ende wieder anh ngen
  first_start_text_4to7[31] = col_01;
}
```

Das gleiche wird nochmals gemacht für den 2. Druchgang. Siehe weiter unten mehr.

```
void second_start_text() {
  byte col_0;
  byte col_01;
  for (int y = 0; y < 32; y++) {
    lc.setColumn(y / 8, 7 - (y % 8), second_start_text_0to3[y]); // Bsp: y=10; y/8=1; y 8= y MOD 8
    lc.setColumn(y / 8 + 4, 7 - (y % 8), second_start_text_4to7[y]); // Bsp: y=10; y/8=1; y 8= y MOD 8
  }

  // Laufschrift nach oben schieben
  col_0 = second_start_text_0to3[0];  // Die erste Kolonne merken, da die am Ende ganz rechts
  col_01 = second_start_text_4to7[0];
  for (int y = 0; y < 31; y++) {  // nur bis 30 resp. 31, da unten die letzte Kolonne neu zugeordnet wird
    second_start_text_0to3[y] = second_start_text_0to3[y + 1];
    second_start_text_4to7[y] = second_start_text_4to7[y + 1];
  }
  second_start_text_0to3[31] = col_0;   // Hier die erste Kolonne am Ende wieder anh ngen
  second_start_text_4to7[31] = col_01;  // Hier die erste Kolonne am Ende wieder anh ngen
}
```

Diese 2 Funktionen werden in der Funktion start() aufgerufen. Die start() Funktion wird wiederum im Setup aufgerufen. Diese zeigt immer am Anfang die 2 Texte. Sobald der Joystick gedrückt wurde, zeigt sie den nächsten Text (BY LM). Wenn dann nochmals der Joystick gedrückt wird, startet das Spiel. Auf die Zeile 2 werde ich weiter unten nochmals genauer eingehen. Auch zur Funktion brightness() kommt noch mehr.

```
void start() {
  digitalWrite(8, LOW); //Set the ragequit LED off

  //scrollling Text --> do only once
  while (startBool) { //Wenn der Start aktiv ist. Also wenn die Startsequenz abgespielt werden soll
    while (analogRead(pushButton) > 300) { //solange wie der PushButton nicht gedr ckt wird
```

```
      first_start_text(); //f hre die oben definierte Funktion aus
      brightness(); //check and adjust brightness
    }
    delay(100); //kurzer Delay

    while (analogRead(pushButton) > 300) {//solange wie der
    PushButton nicht gedr ckt wird
      second_start_text(); //f hre die oben definierte Funktion
    aus
      brightness(); //check and adjust brightness
      startBool = false; //Die Start Sequenz ist vorbei, darum bis
    der Boolean auf false gesetzt.
    }
    delay(100); //kurzer Delay
  }
}
```

2.3 Game Snake

Mein Spiel brauch für die ANzeige auf der LED Matrix eine andere Library als für die Laufschrift. Nämlich die LedMatrix (https://github.com/R3sal/LedMatrix). Diese hat den Vorteil, dass sie effizienter ist als die Andere, da sie nur eine LED aufsmal updated anstatt der ganzen Matrix. Weniger Ressourcen werden verschwendet. Diese Library wird so initialisert.

```
#include <LedMatrix.h>   //Library for game

int mc[] = { //the order of the matrices (for Library game)
  3, 2, 1, 0,
  7, 6, 5, 4
};

bool md[] = { //If the matrices should be rotated by 180  , more on
     the documentation of the Library
  false, false, false, false,
  false, false, false, false
};

const int iNumMatrixRows = 4; //how many rows do I have?
const int iNumMatrixColumns = 2; //and how many Columns

LedMatrix lm = LedMatrix(12,                    //Data pin
                         11,                    //CLK pin
                         10,                    //CS pin
                         8,                     //intensity
                         mc,                    //matrix
    configuration
                         md,                    //matrix directions (
    are the matrices rotated by 180  )
                         iNumMatrixRows,        //the number of rows
                         iNumMatrixColumns);    //the number of
    columns
```

Im Setup müssen wir zuerst die Matrix clearen, damit alle LEDs korrekt aus sind. Wir müssen auch die pinModes für den Joystick angeben

```
  //clears the display and set all Leds to be turned off
  lm.ClearDisplay();

   pinMode(A0, INPUT); //X Value of Joystick
  pinMode(A1, INPUT); //Y Value
  pinMode(pushButton, INPUT_PULLUP);  //pushbutton = change
    scrolling texts
```

2.3.1 updateSnake()

Die Steuerung beim Spiel funktioniert mithilfe dem Joystick. Diesen haben wir im Setup initialisiert. Diese Funktion wird im Loop immer wieder wiederholt. Alles ist im Code erklärt.

```
void updateSnake() {

  // calculate the current time --> damit es nicht immer l uft und
     damit man die Geschwindigkeit steuern kann
  unsigned long currentTimeSnake = millis();

  // check if it's time to update the Snake
  if (currentTimeSnake - lastUpdateTimeSnake >= delayTimeSnake) {


    //preX/Y ist die LED die gerade ausgemacht wurde, also eine LED
     hinter der Schlange. Diese LED brauchen wir, wenn die SChlange
     wachsen will oder eins nach vorne gehen will
    preX = snake[score][0]; //preX/Y ist immer so weit hinter dem
    Schlangekopf wie der Score also der Score + 1 und l nge sind
    gleich
    preY = snake[score][1];
    lm.SetLed(snake[score][0], snake[score][1], false); //setze
    diese LED aus um Schlange nach vore zu bewegen
    lm.UpdateMatrix();  //this is needed to send the changes to the
     matrices

    //push everytime snake one Led forward Verschiebe jedes Array
    im Array um Schlange zu bewegen
    for (int i = score; i > 0; i--) {
      snake[i][0] = snake[i - 1][0];
      snake[i][1] = snake[i - 1][1];
    }


    if (analogRead(A0) > 900 && direction != 'l') { //Wenn Joystick
     nach rechts und Richtung nicht nach links(sonst w rde die
    Schlange in sich selber laufen) mache Direction nach rechts
      direction = 'r';
    } else if (analogRead(A0) < 100 && direction != 'r') {//Wenn
    Joystick nach links und Richtung nicht nach reichts(sonst
    w rde die Schlange in sich selber laufen) mache Direction nach
     links
      direction = 'l';
    }

```

```
29      if (analogRead(A1) > 900 && direction != 'u') {//Wenn Joystick
        nach oben und Richtung nicht nach unten(sonst w rde die
        Schlange in sich selber laufen) mache Direction nach oben
30        direction = 'd';
31      } else if (analogRead(A1) < 100 && direction != 'd') {//Wenn
        Joystick nach unten und Richtung nicht nach oben(sonst w rde
        die Schlange in sich selber laufen) mache Direction nach unten
32        direction = 'u';
33      }
34
35
36    //Hier wird ja nach der Richtung der Schlangenkopf bewegt
37      if (direction == 'r') {
38        //Serial.println("right");
39        snake[0][0]++;
40      } else if (direction == 'l') {
41        //Serial.println("left");
42        snake[0][0]--;
43      } else if (direction == 'u') {
44        //Serial.println("up");
45        snake[0][1]++;
46      } else if (direction == 'd') {
47        //Serial.println("down");
48        snake[0][1]--;
49      }
50
51
52      //Wall detection --> walls = death --> W nde sind bei X0, X15,
         Y0 und Y31
53      if (snake[0][1] < 0 || snake[0][1] > 15 || snake[0][0] < 0 ||
        snake[0][0] > 31) {
54        gameOver();
55      }
56
57      //snake detection --> snake in snake = death
58      for (int i = 1; i <= score; i++) { //Mit dieser Loop wird
           berprft    ob der Schlangenkopf mit der Schlange kollidiert
        ist
59        if (snake[0][0] == snake[i][0] && snake[0][1] == snake[i][1])
         {
60          gameOver();
61        }
62      }
63
64      //check if snake has eaten an apple --> if yes, the snake
        should grow
65      if (snake[0][0] == foodX && snake[0][1] == foodY) {
66        grow();
67        //Serial.println("snake grown");
68      }
69
70
71
72
73    //hier wird dann die Schlange angeschalten, damit man sie auf der
         Matrix sieht
74      for (int i = 0; i <= score; i++) {
```

```
75        lm.SetLed(snake[i][0], snake[i][1], true);
76        lm.UpdateMatrix();  //this is needed to send the changes to
      the matrices
77      }
78
79
80      // update the last update time, for the time managing
81      lastUpdateTimeSnake = currentTimeSnake;
82    }
83  }
```

2.3.2 Grow()

Im vorherigen Abschnitt sahen wir bereits die grow() Funktion. Hier ist aufgelistet was sie genau tut.

```
1  void grow() {
2    score++; //Erh he Score um eiens
3    delay(10);
4    Serial.println(score);
5    Serial.write(score); // Send Score to Arduino ESP2886 Chip, to
       push Highscore to Website and push it to Arduino Nano to say
       the score (Talkie Library not compatible with LedControl and
       ESP Chip)
6
7    updateFrequencySnake += 1;                    //make the game
       faster every time a food gets eaten
8    delayTimeSnake = 1000 / updateFrequencySnake;  //also update the
       delay, otherwise it will have no effect
9    generateFood();                               //to get new food
10   tone(3, 1500, 100); //make Sound on buzzer for feedback
11
12   snake[score][0] = preX; //Das setzt die neuen X und Y Werte f r
       die letze LED, da die Schlange eines gewachsen ist.
13   snake[score][1] = preY;
14 }
```

2.3.3 Food

Das Essen wird immer mit generateFood() zufällig erstellt. Das Food hat eine X und Y Value für auf die Matrix.

```
1  void generateFood() {
2    foodX = random(0, 31);
3    foodY = random(0, 15);
4  }
```

Die Funktion wird im setup() aufgerufen um das erste Essen zu erstellen. Sonst immer, wenn ein Essen gegessen wurde.

Das Essen flackert auch um es unterscheiden zu können. Das machen wir im Loop mit diesem Code hier.

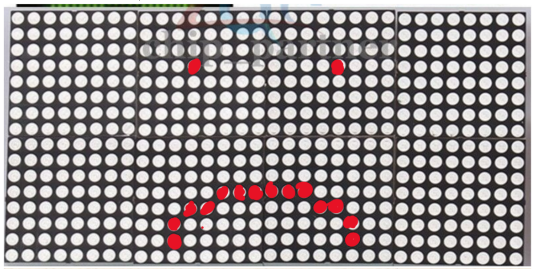
```
1    // calculate the current time (for Food)
2    unsigned long currentTimeFood = millis();
3    // check if it's time to flicker the Food LED
```

```
  if (currentTimeFood - lastFlickerTimeFood >= delayTimeFood) {
    // toggle the state of the LED
    lm.SetLed(foodX, foodY, foodState);
    lm.UpdateMatrix(); //update states
    foodState = !foodState; //setze den foodState um um die LED
    immer wechseln zu lassen, zwischen an und aus = flackern

    // update the last flicker time
    lastFlickerTimeFood = currentTimeFood;
  }
```

2.3.4 gameOver()

Zuerst müssen wir die LEDs festlegen für die GameOver Aniamtion. Diese Animation ist ein Smiley.



```
//Game Over Animation
int gameOverSmiley[16][2] = { { 11, 3 }, { 20, 3 }, { 13, 11 }, {
    14, 11 }, { 15, 11 }, { 16, 11 }, { 17, 11 }, { 18, 11 }, { 11,
     12 }, { 12, 12 }, { 19, 12 }, { 20, 12 }, { 10, 13 }, { 10, 14
     }, { 21, 13 }, { 21, 14 } };  //sad smiley
```

Der Rest wird im gameOver() geregelt. Game Over ist, wenn die Schlange in eine Wand oder sich selber läuft.

```
void(* resetFunc) (void) = 0; //declare reset function @ address 0
    --> to restart game after game Over
void gameOver() {
    //mache d   d   d   Sound wenn gestorben, auf Piezo Buzzer
  tone(3, 1500, 115);
  delay(100);
  tone(3, 1200, 200);
  delay(150);
  tone(3, 800, 300);

    //L sche alles auf der Matrix
  lm.ClearDisplay();
  lm.UpdateMatrix();

  //Schluss animation
  //I know it would be possible to do it with a for loop (for(int i
     = 0; i<16; i++)) --> but strangly if I do this I can't upload
    my code 9 out of 10 times, and if I can upload it, the LED's
    show some bullshit
  //If you wanna know why, I have no Idea, maybe my Arduino is
    broken :(
```

```
  //To try on a other Arduino just uncomment this code below:
  //for(int i=0; i<16; i++){
    //lm.SetLed(gameOverSmiley[i][0],gameOverSmiley[i][1], true);
  //}
  lm.SetLed(gameOverSmiley[0][0], gameOverSmiley[0][1], true);
  lm.SetLed(gameOverSmiley[1][0], gameOverSmiley[1][1], true);
  lm.UpdateMatrix();
  delay(400);
  lm.SetLed(gameOverSmiley[2][0], gameOverSmiley[2][1], true);
  lm.SetLed(gameOverSmiley[3][0], gameOverSmiley[3][1], true);
  lm.SetLed(gameOverSmiley[4][0], gameOverSmiley[4][1], true);
  lm.SetLed(gameOverSmiley[5][0], gameOverSmiley[5][1], true);
  lm.SetLed(gameOverSmiley[6][0], gameOverSmiley[6][1], true);
  lm.SetLed(gameOverSmiley[7][0], gameOverSmiley[7][1], true);
  lm.UpdateMatrix();
  delay(400);
  lm.SetLed(gameOverSmiley[8][0], gameOverSmiley[8][1], true);
  lm.SetLed(gameOverSmiley[9][0], gameOverSmiley[9][1], true);
  lm.SetLed(gameOverSmiley[10][0], gameOverSmiley[10][1], true);
  lm.SetLed(gameOverSmiley[11][0], gameOverSmiley[11][1], true);
  lm.UpdateMatrix();
  delay(400);
  lm.SetLed(gameOverSmiley[12][0], gameOverSmiley[12][1], true);
  lm.SetLed(gameOverSmiley[13][0], gameOverSmiley[13][1], true);
  lm.SetLed(gameOverSmiley[14][0], gameOverSmiley[14][1], true);
  lm.SetLed(gameOverSmiley[15][0], gameOverSmiley[15][1], true);
  lm.UpdateMatrix();

  delay(4000); //Warte und zeige das Smiley
  resetFunc(); //Beginne das Spiel neu
}
```

2.4 Brightness Control

Die Helligkeit der Matrix wird mithilfe einem Sensor automatish gesteuert. Das wird im Loop immer wiederholt. Zuerst müssen wir den Input des Sensor definieren.

```
pinMode(A5, INPUT); //Value of brightness
```

Das hier ist die eigentliche Funktion.

```
void brightness(){
  for(int i = 0; i<8; i++){
      lm.SetIntensity(16 - (analogRead(A5)/64), i); //divide by 64
    because sensor has values from 0 to 1024 but Matrix has only 0
    to 15 intensities
      Serial.println(16 - (analogRead(A5)/64));
  }                                                  //then subtract
    this value from 16 to get lower intensity if darker, not the
    other way around
}
```

2.5 Automatic Rage Quit

Wir haben eine LED die anzeigt, wenn man einen Rage Quit hatte. Dazu wird das Spiel sofort beendet, –> Also gamOver(). Im Setup wird der Pin für die LED und für das Mikrofon(nur Lautstärke) angegeben.

```
pinMode(8, OUTPUT); //LED for ragequit

 pinMode(digital_mic, INPUT); //0 or 1, screaming = 1
```

Im Loop wird dann immer geprüft ob der Schwellenwert überschritten wurde. Wenn ja macht es die LED an und löst ein gameOver() aus.

```
    if  (digitalRead(digital_mic)==0)
  {
    digitalWrite(8, HIGH); //Set LED on
      gameOver();;

  }
```

2.6 Highscore Upload

Der Score wird in der Funktion grow() über die RX und TX Pins an 2 andere Arduinos weitergegeben. Der eine spricht den Score laut aus und der andere pusht in mit einem HTTP Request auf die Website lucamezger.com/arduino/data.

Das hier ist der Code für den sprechenden Arduino Nano

```
#include "Talkie.h" //to let the speaker talk
#include "Vocab_US_Large.h" //all the words used in my Program
#include "Vocab_Special.h"

Talkie voice; //Talkie bereit machen
int highscore; //die variable Highscore inittialiseren

void displayScore() {
  //Say the score via the speaker —> only till 29, but easily
    scalable
  if (highscore == 1) { //checkt wie hoch der Score ist
  voice.say(sp3_ONE); //sagt den score  ber  Speaker
} else if (highscore == 2) {
  voice.say(sp3_TWO);
} else if (highscore == 3) {
  voice.say(sp3_THREE);
} else if (highscore == 4) {
  voice.say(sp3_FOUR);
} else if (highscore == 5) {
  voice.say(sp3_FIVE);
} else if (highscore == 6) {
  voice.say(sp3_SIX);
} else if (highscore == 7) {
  voice.say(sp3_SEVEN);
} else if (highscore == 8) {
  voice.say(sp3_EIGHT);
} else if (highscore == 9) {
  voice.say(sp3_NINE);
} else if (highscore == 10) {
```

```
29    voice.say(sp3_TEN);
30  } else if (highscore == 11) {
31    voice.say(sp3_ELEVEN);
32  } else if (highscore == 12) {
33    voice.say(sp3_TWELVE);
34  } else if (highscore == 13) {
35    voice.say(sp3_THIRTEEN);
36  } else if (highscore == 14) {
37    voice.say(sp3_FOURTEEN);
38  } else if (highscore == 15) {
39    voice.say(sp3_FIFTEEN);
40  } else if (highscore == 16) {
41    voice.say(sp3_SIXTEEN);
42  } else if (highscore == 17) {
43    voice.say(sp3_SEVENTEEN);
44  } else if (highscore == 18) {
45    voice.say(sp3_EIGHTEEN);
46  } else if (highscore == 19) {
47    voice.say(sp3_NINETEEN);
48  } else if (highscore == 20) {
49    voice.say(sp3_TWENTY);
50  } else if (highscore == 21) {
51    voice.say(sp3_TWENTY);
52    voice.say(sp3_ONE);
53  } else if (highscore == 22) {
54    voice.say(sp3_TWENTY);
55    voice.say(sp3_TWO);
56  } else if (highscore == 23) {
57    voice.say(sp3_TWENTY);
58    voice.say(sp3_THREE);
59  } else if (highscore == 24) {
60    voice.say(sp3_TWENTY);
61    voice.say(sp3_FOUR);
62  } else if (highscore == 25) {
63    voice.say(sp3_TWENTY);
64    voice.say(sp3_FIVE);
65  } else if (highscore == 26) {
66    voice.say(sp3_TWENTY);
67    voice.say(sp3_SIX);
68  } else if (highscore == 27) {
69    voice.say(sp3_TWENTY);
70    voice.say(sp3_SEVEN);
71  } else if (highscore == 28) {
72    voice.say(sp3_TWENTY);
73    voice.say(sp3_EIGHT);
74  }else if (highscore == 29) {
75    voice.say(sp3_TWENTY);
76    voice.say(sp3_NINE);
77  }
78  }
79
80  void setup(){
81    Serial.begin(9600); //set baud to 9600
82  }
83  int previousScore = 0; //to check that the score only went up by 1
        (to eliminate errors)
84  void loop(){
```

```
  int value = Serial.read(); //read the Value from the RX pin

  if(value > 0){ //If value is correct
    highscore = value;
    if(highscore == previousScore+1 || highscore == 1){ //if the
    value went up by one (to eliminte errors)
        displayScore(); //say the highscore
        previousScore = highscore; //update previousScore
        Serial.println(highscore);
    }
  }

}
```

Der Code für den Arduino für den Upload ist der hier:

```
#include <Grandeur.h> //service for HTTP REquests
#include "WiFi.h" //to connect to WiFi



// WiFi credentials --> my HotSPot
const char* ssid = "lucaarduino";
const char* passphrase = "42069420";

// Grandeur credentials --> removed so no one can see it
const char * apiKey = "";
const char* token = "";
const char* deviceId = "";

Grandeur::Project project;

unsigned long current = millis(); //to do it every 3 seconds
int highscore = 0;


void setup() {
    Serial.begin(9600);
    // This connects the device to WiFi.
    connectToWiFi(ssid, passphrase);
    // This connects the device to internet.
    project = grandeur.init(apiKey, token);
}

int preScore; //eliminate errors

void loop() {

  int value = Serial.read();

//same as previous
  if(value == preScore+1 || value == 1){
    highscore = value;
    Serial.println(highscore);
    preScore = value;
  }


```

```
    //Serial.println(highscore);
    // This sends data to internet.
    if(project.isConnected() && millis() - current > 3000) {
        project.device(deviceId).data().set("millis", highscore);
        current = millis();
    }

    // This runs the SDK when the device WiFi is connected.
    if(WiFi.status() == WL_CONNECTED) project.loop();
}
```

Der Code der auf der Website liegt ist womöglich gleich umfassend wie der Arduino Code. Auf den werde ich aber nicht eingehen. Für Erklärungen findet Ihr Kommentare im Code. Ich werde ihn auch nicht hier rein klatschen, da der Projektbeschrieb sonst sicherlich über 69 Seiten hätte. Der Code ist aber auf https://github.com/Luca-Melop/ArduinoData ersichtlich.

2.7 Speaker

Der Arduino spielt immer ein Lied im Hintergrund. Das habe ich jedoch nicht selber programmiert, sonder aus dem Internet geklaut (https://create.arduino.cc/projecthub/410027/rickroll-piezo-buzzer-a1cd11). Jedoch habe ich den Code noch so abgeändert wie ich wollte. Der Song spielt auf einem Speaker, denn ich aues einem alten Laptop genommen habe ;) Die Lautstärke kann man steuern mit einen Potentiometer.

```
#define  a3f    208     // 208 Hz
#define  b3f    233     // 233 Hz
#define  b3     247     // 247 Hz
#define  c4     261     // 261 Hz MIDDLE C
#define  c4s    277     // 277 Hz
#define  e4f    311     // 311 Hz
#define  f4     349     // 349 Hz
#define  a4f    415     // 415 Hz
#define  b4f    466     // 466 Hz
#define  b4     493     // 493 Hz
#define  c5     523     // 523 Hz
#define  c5s    554     // 554 Hz
#define  e5f    622     // 622 Hz
#define  f5     698     // 698 Hz
#define  f5s    740     // 740 Hz
#define  a5f    831     // 831 Hz

#define rest    -1

int speaker = 3; // Pin of Speaker


int beatlength = 100; // determines tempo ---> lower = faster
float beatseparationconstant = 0.3;

int threshold;
```

```
27
28 int a; // part index
29 int b; // song index
30 int c; // lyric index
31
32
33 // Parts 1 and 2 (Intro)
34
35 int song1_intro_melody[] =
36 {c5s, e5f, e5f, f5, a5f, f5s, f5, e5f, c5s, e5f, rest, a4f, a4f};
37
38 int song1_intro_rhythmn[] =
39 {6, 10, 6, 6, 1, 1, 1, 1, 6, 10, 4, 2, 10};
40
41 // Parts 3 or 5 (Verse 1)
42
43 int song1_verse1_melody[] =
44 { rest, c4s, c4s, c4s, c4s, e4f, rest, c4, b3f, a3f,
45   rest, b3f, b3f, c4, c4s, a3f, a4f, a4f, e4f,
46   rest, b3f, b3f, c4, c4s, b3f, c4s, e4f, rest, c4, b3f, b3f, a3f,
47   rest, b3f, b3f, c4, c4s, a3f, a3f, e4f, e4f, e4f, f4, e4f,
48   c4s, e4f, f4, c4s, e4f, e4f, e4f, f4, e4f, a3f,
49   rest, b3f, c4, c4s, a3f, rest, e4f, f4, e4f
50 };
51
52 int song1_verse1_rhythmn[] =
53 { 2, 1, 1, 1, 1, 2, 1, 1, 1, 5,
54   1, 1, 1, 1, 3, 1, 2, 1, 5,
55   1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 3,
56   1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 4,
57   5, 1, 1, 1, 1, 1, 1, 1, 2, 2,
58   2, 1, 1, 1, 3, 1, 1, 1, 3
59 };
60
61
62
63 // Parts 4 or 6 (Chorus)
64
65 int song1_chorus_melody[] =
66 { b4f, b4f, a4f, a4f,
67   f5, f5, e5f, b4f, b4f, a4f, a4f, e5f, e5f, c5s, c5, b4f,
68   c5s, c5s, c5s, c5s,
69   c5s, e5f, c5, b4f, a4f, a4f, a4f, e5f, c5s,
70   b4f, b4f, a4f, a4f,
71   f5, f5, e5f, b4f, b4f, a4f, a4f, a5f, c5, c5s, c5, b4f,
72   c5s, c5s, c5s, c5s,
73   c5s, e5f, c5, b4f, a4f, rest, a4f, e5f, c5s, rest
74 };
75
76 int song1_chorus_rhythmn[] = //to get note length
77 { 1, 1, 1, 1,
78   3, 3, 6, 1, 1, 1, 1, 3, 3, 3, 1, 2,
79   1, 1, 1, 1,
80   3, 3, 3, 1, 2, 2, 2, 4, 8,
81   1, 1, 1, 1,
82   3, 3, 6, 1, 1, 1, 1, 3, 3, 3, 1, 2,
83   1, 1, 1, 1,
```

```
  3, 3, 3, 1, 2, 2, 2, 4, 8, 4
};


void setup()
{
  pinMode(speaker, OUTPUT);
  Serial.begin(9600);
  a = 4;
  b = 0;
  c = 0;
}

void loop()
{
  play();
}


void play() {
  int notelength;
  if (a == 1 || a == 2) {
    // intro
    notelength = beatlength * song1_intro_rhythmn[b];
    if (song1_intro_melody[b] > 0) {
      tone(speaker, song1_intro_melody[b], notelength);
    }
    b++;
    if (b >= sizeof(song1_intro_melody) / sizeof(int)) {
      a++;
      b = 0;
      c = 0;
    }
  }
  else if (a == 3 || a == 5) {
    // verse
    notelength = beatlength * 2 * song1_verse1_rhythmn[b];
    if (song1_verse1_melody[b] > 0) {
      tone(speaker, song1_verse1_melody[b], notelength);
      c++;
    }
    b++;
    if (b >= sizeof(song1_verse1_melody) / sizeof(int)) {
      a++;
      b = 0;
      c = 0;
    }
  }
  else if (a == 4 || a == 6) {
    // chorus
    notelength = beatlength * song1_chorus_rhythmn[b];
    if (song1_chorus_melody[b] > 0) {
      tone(speaker, song1_chorus_melody[b], notelength);
      c++;
    }
    b++;
    if (b >= sizeof(song1_chorus_melody) / sizeof(int)) {
```

```
141         Serial.println("");
142         a++;
143         b = 0;
144         c = 0;
145       }
146     }
147     delay(notelength);
148     noTone(speaker);
149     delay(notelength * beatseparationconstant);
150     if (a == 7) { // loop back around to beginning of song
151       a = 1;
152     }
153   }
```

2.8 Cooling

Der Arduino kann sich natürlich auch noch coolen. Das wird automatisch gemacht mit einem Temperatur Sensor. Den Cooler habe ich von einem alten NAS gestohlen. Der Code braucht keine Erklärung. Es ist einfach wenn ein gewisser Grad an Temperatur überschritten wurde (z.B. 40°) schalte Cooler auf Digital Pin an. Der Cooler wäre eigentlich für 12Volt, daher ist er hier etwas schwächer.