CAOS Projekt : 2 x 6 x 12 LED RGB Quader

Joey Zgraggen, Moritz Würth, Viktor Gsteiger

January 10, 2020

TODO: BILD VOM PROJEKTERZEUGNIS

Inhaltsverzeichnis

1 Ziel des Projektes

Die Grundidee von unserem Projekt war es zuerst einen 5 x 5 x 5 LED RGB Würfel zu bauen, jedoch haben wir uns dann im Prozess der Entscheidungsfindung dazu entschieden, etwas anderes zu bauen was sich vom "typischen" LED RGB Würfel unterscheidet. Deshalb haben wir die Dimensionen ein wenig angepasst, sodass wir am Ende einen dreidimenstionalen Bildschirm haben werden. Damit wird es uns immer noch möglich sein, gute 3D-Effekte anzeigen zu können. Die Anordnung der LED's kann als Koordinatensystem verstanden werden, bei dem man mittels x- und y-Koordinaten auf die einzelnen LED's zugreifen kann. Dies erleichtert es massiv, Effekte zu schreiben. Das Projekt lässt einen grossen Spielraum bezüglich Ideen zu. Sollte man also selbst Ideen haben, welche sich gut zu unserem Projekt ergänzen würden, kann man diese ohne weitere Probleme implementieren. Die Grundsoftware hat sehr einfache Schnittstellen, mit denen man einfach weitere Effekte schreiben kann.

2 Verwendete Materialien

Im Folgenden ist eine Auflistung der verschiedenen Materialien, welche für das Projekt verwendet wurden.

2.1 RGB LED's

Um den LED-Quader ein wenig bunter gestalten zu können wurden RGB's verwendet.:

- LED RGB Common Cathode 4-Pin F5 5MM Diode

2.2 Schieberegister

Schieberegister erweisen sich für ein solches Projekt als äusserst nützlich, um die vielen Pins der einzelnen RGB LED's ansprechen zu können.

Wir haben uns dabei an schon bereits existierende Projekte orientiert und uns für die 74HC595 8-bit Schieberegister entschieden. Diese erweisen sich in der Programmierung als intuitiv, weil man einfach mit Byte-Arrays arbeiten kann, um die einzelnen Schieberegister mit Informationen zu "befüllen". Weiter existieren für diese Schieberegister vorgefertigte Libraries, um die Kommunikation zu erleichtern. Dies nimmt einen nicht die Denkarbeit ab, wie man die LED's anordnen will und wie man die verschiedenen Farben anspricht, jedoch erleichtert es die direkte Kommunikation.

2.3 Transistoren

Anfangs wollten wir für eine garantierte Langlebigkeit unseres Projektes Transistoren verwenden, aber nachdem wir uns bezüglich der Notwendigkeit von Transistoren für unser Projekt informierten haben wir uns letzenendlich dazu entschieden keine zu verwenden.

2.4 Sensoren

Für das Projekt wurden zusätzlich Sensoren verwendet, um weitere Features zu gewährleisten. Folgende Sensoren stehen zur Auswahl:

- Temperatur und Luftfeuchtigkeitssensor:

- Infrarot-Sensor:
- Real-time clock:

3 Materialkosten

Produkt	**Menge**	**Preis**	**Total**
RGB LED's	250x	0.072.-	18.-
Kabel (verschiedene)	15m	0.83.-	12.50
Wiederstände	440x	0.036.-	16.-
74hc595	57x	0.50.-	30.-
Sensoren (verschiedene)	3x	3.30.-	10.-
Lochrasterplatine (verschiedene)	2x	6.00.-	12.-
Kuperdraht (Blau)	2 Rollen	6.80.-	13.60.-
Isolierband	1 Rolle	1.80.-	1.80.-
			113.90.-

4 Design

Wir haben uns überlegt eine kleine Box zu bauen, welcher die ganze Verkabelung in einem kleinen Raum im hinteren Bereich von dieser Box zu verstecken, damit es schöner aussieht. Wir haben uns für das Material Holz entschieden und werden die ganze Box schwarz färben, da dann der Effekt der RGB LED's entsprechend besser zur Geltung kommen. Wir haben schon einige Projekte gesehen, welche einen LED RGB Cube gebaut haben, bei denen auch eine Art Box gebaut wurde und wir fanden diese Idee gut, sodass wir uns für unser Projekt danach orientierten.

5 Code

5.1 Main

```
1 #include "RTClib.h" // Real time clock library
2 #include <SPI.h> // SPI Library used to clock data out to the
    shift registers
3 #include "DHT.h" // Humidity and Temperature sensor library
4 #include <IRremote.h>
5 const int RECV_PIN = 7;
6 IRrecv irrecv(RECV_PIN);
7 decode_results results;
8
9 #define latch_pin 49 // push the to storage register, Pin 12 at IC
     -> green
10 #define blank_pin 48 // to shut enable/disable the register. Low
    enables, Pin 13 at IC -> violett
11 #define data_pin 51 // used by SPI, must be pin 51 at Mega 2560,
    Pin 14 at IC -> brown
```

```
12 #define clock_pin0 52 // used by SPI, must be 52 at mega 2560, Pin
       11 at IC -> blue
13 #define DHTPIN 2 // Humidity/temperature sensor pin
14 #define DHTTYPE DHT11 // Humidity/temperature sensor model
15 #define cathode_pin0 24 // Cathode Pin, to be tested!
16 #define cathode_pin1 26 // Same as above!
17 #define button 2 // Button to change mode, to be replaced by
     infrared!
18
19 int currentAmountOfShifters = 27;  // To be set depending on the
     current setup
20 byte anodes0[27]; // Array of Anodes for layer 0
21 byte anodes1[27]; // Array of Anodes for layer 1
22 int currentEffect = 0; // Integer value of the current effect in
     play
23 unsigned long lastSignal = 0; // long value for last effect (still
       here until replaced by infrared)
24 int currentAmountOfEffects = 1; // For the button, to be replaced
25 int dispArray[6][12]; // Array containing all LEDs in one color
26 int letterBuffer[6][4]; // Letterbuffer for the Letters next to be
       loaded
27 DHT dht(DHTPIN, DHTTYPE); // Humidity/Temperature variable
28 RTC_DS1307 rtc; // Real time clock variable
29
30 void setup()
31 {
32   Serial.begin(115200); // Serial monitor for debugging
33   SPI.begin();
34   SPI.setClockDivider(SPI_CLOCK_DIV2);//Run the data in at 16MHz/2
         - 8MHz
35   dht.begin();
36
37   pinMode(latch_pin, OUTPUT);//Latch
38   pinMode(blank_pin, OUTPUT);//Output Enable  important to do this
         last, so LEDs do not flash on boot up
39   pinMode(cathode_pin0, OUTPUT);
40   pinMode(cathode_pin1, OUTPUT);
41   pinMode(button, INPUT);
42
43   attachInterrupt(digitalPinToInterrupt(button), blink, RISING);
44
45   lastSignal = millis();
46
47   digitalWrite(blank_pin, HIGH); //shut down the leds
48   digitalWrite(latch_pin, LOW);  //shut down the leds
49
50   // if (! rtc.begin()) {
51   // Serial.println("Couldn't find RTC");
52   // while (1);
53   // }
```

```
54   // if (! rtc.isrunning()) {
55   // Serial.println("RTC is NOT running!");
56   //rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
57   //}
58   irrecv.enableIRIn();
59   irrecv.blink13(true);
60 }
61
62 void loop()
63 {
64   Serial.println("Still going");
65   //welcomeAnimation();
66   //gameOfLifeAnimation();
67   test();
68   //firework();
69   //starAnimation();
70   //checkIRSensor();
71   delay(1);
72 }
73
74 /**
75   To handle the interrupt of the button input.
76 */
77 void blink() {
78   if (millis() - lastSignal > 200) {
79     lastSignal = millis();
80     currentEffect = (currentEffect + 1) % currentAmountOfEffects;
81   }
82 }
83
84 void changeEffect(int result) {
85     switch (result) {
86       case 0x97483BFB: //Keypad button "0"
87         Serial.println("Testeffect");
88         test();
89         break;
90       case 0xE318261B: //Keypad button "1"
91         Serial.println("Firework");
92         firework();
93         break;
94       case 0x511DBB: //Keypad button "2"
95         Serial.println("GOL");
96         gameOfLifeAnimation();
97         break;
98       case 0xEE886D7F:  //Keypad button "3"
99         Serial.println("Clock");
100        clockAnimation();
101        break;
102      case 0x52A3D41F: //Keypad button "4"
103        Serial.println("Stars");
```

```
104         starAnimation();
105         break;
106       case 0xD7E84B1B: //Keypad button "5"
107         Serial.println("Temperature effects");
108         tempSensorInfo();
109         break;
110       default:
111         Serial.println("Default");
112         break;
113     }
114   }
115
116   boolean checkIRSensor(){
117     if (millis() - lastSignal > 200) {
118       lastSignal = millis();
119       if (irrecv.decode(&results)){
120         Serial.println(results.value, HEX);
121         changeEffect(results.value);
122         irrecv.resume(); // Receive the next value
123         return true;
124       }
125     } else {
126       if (irrecv.decode(&results)){
127         //changeEffect(results.value);
128         irrecv.resume(); // Receive the next value
129         return false;
130       }
131       }
132   }
```

5.2 Logic

5.2.1 bitShifterLogic

```
1     /*
2    Method to shift the current information in the anode arrays
       onto the shifters. Currently a single led lights up for
       (50/30)/2 times
3 */
4     void shiftToShifter(int miliSeconds)
5 {
6   //
7   long a = 100L;
8   long b = 54L;
9   long delayProHertz = (a/b);
10  miliSeconds = constrain (miliSeconds,    100, 100000);
11  //Serial.println("To be displayed for:");
12  //Serial.println(miliSeconds);
13  for (int timeSpent = 0; timeSpent < miliSeconds; timeSpent =
      timeSpent + 100) {
14    int currentShifter = currentAmountOfShifters - 1;
```

```
15        for (int hrtz = 0; hrtz < currentAmountOfShifters; hrtz++) {
16          digitalWrite(blank_pin, HIGH);//shut down the leds
17          for (int i = currentAmountOfShifters - 1; i >= 0; i--) {
18            if (i == currentShifter) {
19              SPI.transfer(anodes0[i]);
20            } else {
21              SPI.transfer(0b00000000);
22            }
23          }
24
25          // shift register to storage register
26          digitalWrite(latch_pin, HIGH);
27          digitalWrite(latch_pin, LOW);
28          digitalWrite(blank_pin, LOW);  //enable pins
29
30          digitalWrite(cathode_pin0, LOW);
31          digitalWrite(cathode_pin1, LOW);
32          delay(delayProHertz);
33          digitalWrite(cathode_pin0, HIGH);
34          digitalWrite(cathode_pin1, HIGH);
35
36          //digitalWrite(blank_pin, HIGH);//shut down the leds
37          //for (int i = currentAmountOfShifters - 1; i >= 0; i--) {
38          //  if (i == currentShifter) {
39          //    SPI.transfer(anodes1[i]);
40          //  } else if ((i + 1) % currentAmountOfShifters ==
              currentShifter) {
41          //    SPI.transfer(anodes1[i]);
42          //  } else if ((i + 2) % currentAmountOfShifters ==
              currentShifter) {
43          //    SPI.transfer(anodes1[i]);
44          //  } else if ((i + 3) % currentAmountOfShifters ==
              currentShifter) {
45          //    SPI.transfer(anodes1[i]);
46          //  } else if ((i + 4) % currentAmountOfShifters ==
              currentShifter) {
47          //    SPI.transfer(anodes1[i]);
48          //  } else if ((i + 5) % currentAmountOfShifters ==
              currentShifter) {
49          //    SPI.transfer(anodes1[i]);
50          //  } else {
51          //    SPI.transfer(0b00000000);
52          //  }
53          //}
54
55          // shift register to storage register
56          //digitalWrite(latch_pin, HIGH);
57          //digitalWrite(latch_pin, LOW);
58          //digitalWrite(blank_pin, LOW);  //enable pins
59
```

```
60          //digitalWrite(cathode_pin1, LOW);
61          //delay(delayProHertz);
62          //digitalWrite(cathode_pin1, HIGH);
63
64          currentShifter--;
65        }
66    }
67    memset(anodes0, 0, sizeof(anodes0));
68    memset(anodes1, 0, sizeof(anodes1));
69 }
```

5.2.2 setLedOnCalculations

```
1
2     void setLedOn(int x, int y, int red, int green, int blue, int
         layer)
3 {
4    x = constrain (x,     0, 11);       // x can only be between 0 and
       11 as we have 12 leds length
5    y = constrain (y, 0, 5);            // y can only be between 0 and
       5 as we have 6 height
6    red = constrain (red,    0, 1);    // Red can either be 1 or 0
7    green = constrain (green,  0, 1); // Green can either be 1 or 0
8    blue = constrain (blue,   0, 1);  // Blue can either be 1 or 0
9    layer = 0;      // layer can only be 0 or 1 as we only have two
       layers
10
11   int whichByte = int(((x * 3 + 36 * y) + 1) / 8); // Calculate
       which byte be have to change
12   int whichBit = ((((y * 36 + x * 3) + 1) % 8) - 1); //  Calculate
       which bit in that byte we have to set on
13   if(whichBit == -1) {
14     whichBit = 3;
15   }
16   Serial.println("Currently␣in␣byte:");
17   Serial.println(whichByte);
18
19   Serial.println("Currently␣in␣bit:");
20   Serial.println(whichBit);
21   /**
22      Choses between the two available layers of our LED RGB Cubic.
23   */
24       if (whichBit == 7) {
25         bitWrite(anodes0[whichByte], whichBit, blue);
26         bitWrite(anodes0[whichByte + 1], 0, green);
27         bitWrite(anodes0[whichByte + 1], 1, red);
28       } else if (whichBit == 6) {
29           bitWrite(anodes0[whichByte], whichBit, blue);
30           bitWrite(anodes0[whichByte], whichBit + 1, green);
31           bitWrite(anodes0[whichByte + 1], 0, red);
32       } else {
```

```
33          bitWrite(anodes0[whichByte], whichBit, blue);
34          bitWrite(anodes0[whichByte], whichBit + 1, green);
35          bitWrite(anodes0[whichByte], whichBit + 2, red);
36        }
37    //Serial.println(anodes0[2]);
38
39 }
```

5.3 Effects

5.3.1 welcome

```

1
2     void welcomeAnimation() {
3    char welcomeMsg[29] = "Uni␣Basel␣CAOS␣Projekt␣2019␣";
4    //if(checkIRSensor()){
5    //      return;
6    //}
7    printLetters(welcomeMsg);
8 }
```

5.3.2 clock

```

1
2     /*
3     Simple clock output together with the real time clock, still
        has to be tested.
4 */
5 void clockAnimation() {
6    while (true) {
7      DateTime now = rtc.now();
8      char timeStamp[11];
9      sprintf(timeStamp, "%02d:%02d:%02d", now.hour(), now.minute(),
           now.second());
10     if(checkIRSensor()){
11         return;
12     }
13     printLetters(timeStamp);
14     char date[13];
15     sprintf(date, "%02d/%02d/%04d", now.day(), now.month(), now.
         year());
16     if(checkIRSensor()){
17         return;
18     }
19     printLetters(date);
20   }
21 }
```

5.3.3 temperatureEffects

```

1
2     /**
```

```
3    Read sensory input, print it on the screen and then display
       temperature and humidity on the screen and then a
       corresponding effect
4 */
5
6 int sunArray[6][12];
7 int rainDrops0[5][12];
8 int rainDrops1[5][12];
9
10 void tempSensorInfo() {
11   float h = dht.readHumidity();
12   float t = dht.readTemperature();
13   const char *cel = "C";
14   const char *perc = "%";
15   char bufferShort1[10];
16   dtostrf(h, 5, 1, bufferShort1);
17   strcat(bufferShort1, perc);
18   if(checkIRSensor()){
19         return;
20   }
21   printLetters(bufferShort1);
22   char bufferShort2[10];
23   dtostrf(t, 5, 1, bufferShort2);
24   strcat(bufferShort2, cel);
25   if(checkIRSensor()){
26         return;
27   }
28   printLetters(bufferShort2);
29   if (dht.readHumidity() > 100) {
30     rainEffect(30000);
31   } else if (dht.readTemperature() > 24) {
32     sun(30000);
33   } else {
34     clouds(30000);
35   }
36 }
37
38 void rainEffect(int seconds) {
39   for (int s = 0; s < seconds; s++) {
40     for (int x = 0; x < 12; x++) {
41       setLedOn(x, 5, 1, 1, 1, 0);
42       setLedOn(x, 5, 1, 1, 1, 1);
43       int r = random(100);
44       if (r < 35) {
45         rainDrops0[4][x] = 1;
46       }
47       r = random(100);
48       if (r < 35) {
49         rainDrops1[4][x] = 1;
50       }
```

```
    }
    rainDropFall(rainDrops0, rainDrops1);
  }
}

void rainDropFall(int rainDrops0[][12], int rainDrops1[][12]) {
  setLed2DArraySingleColor(rainDrops0, 0, 0, 0, 1, 5, 12);
  setLed2DArraySingleColor(rainDrops1, 1, 0, 0, 1, 5, 12);
  if(checkIRSensor()){
      return;
  }
  shiftToShifter(1000);
  int tempArray[5][12];
  for (int x = 0; x < 12; x++) {
    for (int y = 4; y >= 0; y--) {
      if (rainDrops0[y][x] == 1) {
        tempArray[y][x] == 0;
        if (y - 1 > 0) {
          tempArray[y - 1][x] == 1;
        }
      }
    }
  }
  memcpy(rainDrops0, tempArray, sizeof(tempArray));
  for (int x = 0; x < 12; x++) {
    for (int y = 4; y >= 0; y--) {
      if (rainDrops1[y][x] == 1) {
        tempArray[y][x] == 0;
        if (y - 1 > -1) {
          tempArray[y - 1][x] == 1;
        }
      }
    }
  }
  memcpy(rainDrops1, tempArray, sizeof(tempArray));
}

void sun(int seconds) {
  int sunArray[6][12];
  sunArray[1][9] = 1;
  sunArray[1][10] = 1;
  for (int i = 8; i < 12; i++) {
    for (int j = 2; j < 4; j++) {
      sunArray[j][i] = 1;
    }
  }
  sunArray[4][9] = 1;
  sunArray[4][10] = 1;
  for (int i = 0; i < seconds; i++) {
    setLed2DArraySingleColor(sunArray, 0, 1, 1, 0, 6, 12);
```

```
    setLed2DArraySingleColor(sunArray, 1, 1, 1, 0, 6, 12);
    if(checkIRSensor()){
        return;
    }
    shiftToShifter(1000);
    shiftSunToLeft(sunArray);
  }
}

void shiftSunToLeft(int sunArray[][12]) {
  int tempArray[6][12];
  for (int x = 0; x < 12; x++) {
    for (int y = 0; y < 6; y++) {
      if (x - 1 == -1) {
        x = 11;
      }
      tempArray[y][x - 1] = sunArray[y][x];
    }
  }
  memcpy(sunArray, tempArray, sizeof(tempArray));
}

void clouds(int seconds) {
  int cloudArray[6][12];
  for (int i = 0; i < 12; i++) {
    cloudArray[6][i] = 1;
  }
  int turn = 0;
  for (int i = 0; i < seconds; i++) {
    for (int i = 0; i < 12; i++) {
      cloudArray[5][i] = 0;
    }
    if (turn == 0) {
      for (int i = 1; i < 12; i = i + 2) {
        cloudArray[5][i] = 1;
      }
      turn = 1;
    } else {
      for (int i = 0; i < 12; i = i + 2) {
        cloudArray[5][i] = 1;
      }
      turn = 0;
    }
    setLed2DArraySingleColor(cloudArray, 0, 1, 1, 1, 6, 12);
    setLed2DArraySingleColor(cloudArray, 1, 1, 1, 1, 6, 12);
    if(checkIRSensor()){
        return;
    }
    shiftToShifter(1000);
  }
```

```
151 }
```

5.3.4 firework

```
1
2     typedef struct color {
3   int r;
4   int g;
5   int b;
6 } color;
7
8 typedef struct rocket {
9   int x;
10  int y;
11  color c;
12  int timer;
13  int maxHeight;
14  int layer;
15  boolean burnt;
16 } rocket;
17
18 rocket rocketArray0[6];
19
20 void firework() {
21   for (int i = 0; i < 6; i++) {
22     rocketArray0[i] = spawnRocket(0);
23   }
24   while (true) {
25     //if(checkIRSensor()){
26       //  return;
27     //}
28     for (int i = 0; i < 6; i++) {
29       for(int t = 0; t <= rocketArray0[i].maxHeight; t++){
30         burnRocket(i);
31         shiftToShifter(1000);
32       }
33     }
34   }
35 }
36
37 struct rocket spawnRocket(int l) {
38   rocket r;
39   r.x = random(12);
40   r.y = 0;
41   color c;
42   c.r = random(2);
43   c.g = random(2);
44   c.b = random(2);
45   r.c = c;
46   r.timer = 0;
47   r.maxHeight = random(2,6);
```

```
48   r.layer = l;
49   return r;
50 }
51
52 struct rocket explodeRocket(rocket r) {
53   color c = r.c;
54   setLedOn(r.x, r.y, c.r, c.g, c.b, r.layer);
55   //setLedOn(r.x - 1, r.y - 1, c.r, c.g, c.b, r.layer);
56   //setLedOn(r.x + 1, r.y + 1, c.r, c.g, c.b, r.layer);
57   //setLedOn(r.x + 1, r.y - 1, c.r, c.g, c.b, r.layer);
58   //setLedOn(r.x - 1, r.y + 1, c.r, c.g, c.b, r.layer);
59   return spawnRocket(r.layer);
60 }
61
62 void burnRocket(int iterate) {
63   rocket r = rocketArray0[iterate];
64   if (r.timer == r.maxHeight) {
65       rocketArray0[iterate] = explodeRocket(r);
66       return;
67   } else {
68     color c = r.c;
69     setLedOn(r.x, r.y, c.r, c.g, c.b, r.layer);
70     r.y++;
71     int ran = random(100);
72     if (r.x > 0 && r.x < 12) {
73       if (ran < 15) {
74         r.x--;
75       } else if (ran < 30) {
76         r.x++;
77       }
78     }
79     r.timer++;
80     rocketArray0[iterate] = r;
81     return;
82   }
83 }
```

5.3.5 globalArrayOperations

```
1
2     /**
3   Sets the the 2d Array given to a specific height to a single
       color.
4   Combinations of r, g ,b.
5 */
6 void setLed2DArraySingleColor(int currArray[][12], int layer, int
     r, int g, int b, int maxH, int maxW) {
7   for (int x = 0; x < maxW; x++) {
8     for (int y = 0; y < maxH; y++) {
9       if (currArray[y][x] == 1) {
10        setLedOn(x, y, r, g, b, layer);
```

```
      }
    }
  }
}
```

5.3.6 gol

```

    int cells[6][12];

int isActive(int i, int j) {
  return cells[i][j];
}

int getNumberOfActiveNeighbors(int i, int j) {
  int neighbors[8][2] = {
    {i - 1, j - 1}, {i - 1, j}, {i - 1, j + 1},
    {i, j - 1}, {i, j + 1},
    {i + 1, j - 1}, {i + 1, j}, {i + 1, j + 1}
  };
  int numActiveCells = 0;
  for (int i = 0; i < 8; i++) {
    int k = (neighbors[i][0] + 12) % 12;
    int l = (neighbors[i][1] + 6) % 6;

    if (isActive(l, k) == 1) {
      numActiveCells += 1;
    }
  }
  return numActiveCells;
}

void updateCells() {
  int cellCopy[6][12];
  for (int x = 0; x < 12; x++) {
    for (int y = 0; y < 6; y++) {
      int nActiveNeighbors = getNumberOfActiveNeighbors(x, y);
      if (!isActive(x, y) == 1 && nActiveNeighbors == 3) {
        cellCopy[y][x] = true;
      } else if (isActive(x, y) == 1 && (nActiveNeighbors == 2 ||
         nActiveNeighbors == 3)) {
        cellCopy[y][x] = true;
      } else {
        cellCopy[y][x] = false;
      }
    }
  }
  for (int x = 0; x < 12; x++) {
    for (int y = 0; y < 6; y++) {
      cells[y][x] = cellCopy[y][x];
    }
```

```
  }
}

void fillRandom(int probability) {
  for (int i = 0; i < 6; i++) {
    for (int j = 0; j < 12; j++) {
      int r = random(100);
      if (r <= probability) {
        cells[i][j] = 1;
      }
    }
  }
}

/*
   Simple Game of Life animation
*/

void gameOfLifeAnimation() {
  fillRandom(20);
    for(int z = 0; z < 20; z ++) {
    setLed2DArraySingleColor(cells, 0, 1, 0, 0, 6, 12);
    if(checkIRSensor()){
        return;
    }
    shiftToShifter(2000);
    updateCells();
  }
}
```

5.3.7 stars

```

    typedef struct starColor {
  int r;
  int g;
  int b;
} starColor;

typedef struct star {
  int x;
  int y;
  int direct; // 1 = from left to right, -1 = from right to left
  starColor color;
  int timer;
  int layer;
} star;

star starArray0[6];
//star starArray1[6];

```

```
/*
   Star animation which randomly spawns stars on the left and
      right and explodes them if they touch.
*/

struct star createStar(int layer) {
  star s;
  s.y = random(6);
  starColor c;
  c.r = random(2);
  c.g = random(2);
  c.b = random(2);
  s.color = c;
  s.timer = random(10);
  int ran = random(100);
  s.layer = layer;
  if (ran < 50) {
    s.direct = 1;
    s.x = 0;
  } else {
    s.direct = -1;
    s.y = 11;
  }
}

void starAnimation() {
  Serial.println("Star animation started");
  for (int i = 0; i < 6; i++) {
    starArray0[i] = createStar(0);
    //starArray1[i] = createStar(1);
  }
  Serial.println("Stars created!");
  while (true) {
    for (int i = 0; i < 6; i++) {
      runStar(starArray0[i], i);
      Serial.print("Star timer is: ");
      Serial.println(starArray0[i].timer);
      //runStar(starArray1[i], i);
    if(checkIRSensor()){
        return;
    }
    shiftToShifter(1000);
    }
  }
}

void runStar(star s, int index) {
  if (s.timer == 0) {
    setLedOn(s.x, s.y, s.color.r, s.color.g, s.color.b, s.layer);
    areSameCoordinates(s, index);
```

```
69      if (s.x + 1 == 12 && s.x - 1 == -1) {
70        if (s.layer == 0) {
71          starArray0[index] = createStar(0);
72        } else {
73          //starArray1[index] = createStar(0);
74        }
75      }
76      if (s.direct == 1) {
77        s.x++;
78      } else {
79        s.x--;
80        if (s.x < 0) {
81          if (s.layer == 0) {
82            starArray0[index] = createStar(s.layer);
83          } else {
84            //starArray1[index] = createStar(s.layer);
85          }
86        }
87      }
88    } else {
89      s.timer--;
90    }
91  }
92
93  void areSameCoordinates(star s, int index) {
94    if (s.layer == 0) {
95      for (int r = 0; r < 6; r++) {
96        if (r == index) {
97          continue;
98        } else {
99          star other = starArray0[r];
100         if (other.x == s.x && other.y == s.x) {
101           explodeStar(s);
102           explodeStar(other);
103           starArray0[r] = createStar(s.layer);
104           starArray0[index] = createStar(s.layer);
105         }
106       }
107     }
108   } else {
109     for (int r = 0; r < 6; r++) {
110       if (r == index) {
111         continue;
112       } else {
113         //star other = starArray1[r];
114         //if (other.x == s.x && other.y == s.x) {
115           //explodeStar(s);
116           //explodeStar(other);
117           //starArray1[r] = createStar(s.layer);
118           //starArray1[index] = createStar(s.layer);
```

```
119           }
120         }
121       }
122     }
123
124 void explodeStar(star s) {
125   starColor c = s.color;
126   setLedOn(s.x, s.y, c.r, c.g, c.b, s.layer);
127   setLedOn(s.x - 1, s.y - 1, c.r, c.g, c.b, s.layer);
128   setLedOn(s.x + 1, s.y + 1, c.r, c.g, c.b, s.layer);
129   setLedOn(s.x + 1, s.y - 1, c.r, c.g, c.b, s.layer);
130   setLedOn(s.x - 1, s.y + 1, c.r, c.g, c.b, s.layer);
131 }
```

6 Features

6.1 Alphabet

Durch die Klasse Alphabet ist es einfach möglich, auf dem Bildschirm Buchstaben und Sätze darzustellen. Die Klasse unterscheidet nicht zwischen klein und Grossbuchstaben und kann das gesamte Alphabet plus ein paar Sonderzeichen plus alle Zahlen darstellen. Die einzelnen Buchstaben sind alle Hardgecoded und deshalb macht diese Klasse Sinn, weil man dies sonst jeweils einzeln machen müsste. Die Klasse Alphabet wird vom Willkommens Effekt, von den tempereature effects und vom clock Effekt benutzt und ist dadurch ein sehr zentraler Effekt.
Die Buchstaben werden dabei von links nach rechs einer nach dem anderen in den Buchstaben-Buffer geladen, dann jeweils, sofern man nicht beim letzten Buchstaben ist, drei mal auf den Hauptbildschirm rausgeshiftet und nach jedem einzelnen shiften angezeigt. Beim letzten Buchstaben wird dieser so lange nach links geshiftet, bis er vom Bildschirm verschwindet und die Kontrolle wird zurück an die rufer Methode gegeben.

6.2 Stars

Im Effekt stars werden zufällig Sterne erzeugt, welche entweder von links nach rechts oder umgekehrt fliegen. Wenn sich zwei Sterne treffen explodieren diese in einem Effekt. Die Sterne haben alle verschiedene Farben, welche auch zufällig sind.
Die Sterne sind alle eigene structs, welche die nötigen Informationen beihnalten. In der Update methode werden alle existierenden Sterne fortgeführt in ihrem Pfad.

6.3 Firework

Der Feuerwerk Effekt funktioniert ähnlich wie der Sternen-Effekt, jedoch fliegen hier die Raketen nur von unten nach oben und explodieren auf einer zufälligen Höhe. Die Farbe der Raketen ist auch zufällig.
Eine Rakete ist ein struct, welche alle nötigen Informationen beinhaltet.

6.4 Welcome

Der Willkommens Effekt ist ein sehr einfacher Effekt, welcher einfach mit der Methode Alphabet eine Willkommensnachricht auf dem Bildschirm anzeigt und einmal nach dem Aufstarten angezeigt wird.

6.5 Game of Life

Der Game of Life Effekt ist eine repräsentation des berühmten Game of Life des britischen Mathematikers John Horton Conway. Dabei kommen zufällig Zellen zum Leben und sterben beziehungsweise pflanzen sich nach bestimmten Regeln fort. Dabei können sie auch über den Bildschirmrand einen Effekt haben und zählen dann zu den Nachbaren auf der anderen Seite. Für mehr Informationen empfielt es sich, den Wikipedia Artikel zum Game of Life zu lesen.

7 Sensoren

7.0.1 Temperatur- und Luftfeuchtigkeitssensor

Der Temperatur- und Luftfeuchtigkeitssensor, welchen wir für unser Projekt verwenden ist vom Typ DHT11. Dies ist ein eher einfacher Sensor, welcher jedoch mehr als genügt für unsere Zwecke. Dieser liest die aktuelle Temperatur und Luftfeuchtigkeit jede Sekunde und diese Information wird dann auf dem Bildschirm angezeigt. Mit Hilfe dieser Informationen wird dann aus einem von drei Effekten ausgewählt. Entweder wird dann ein Regen, Wolken oder eine Sonne auf dem zweidimensionalen Bildschirm angezeigt. Die Berechnung hierfür ist jedoch sehr einfach und rudimentär, weshalb der Bildschirm sich nicht für eine genaue Wettervorhersage eignet.

7.0.2 Infrarot-sensor

Der Infrarot-sensor empfäng Input von einer externen Fernbedienung und man kann damit zwischen den verschiedenen Effekten wechseln. Dabei werden bei allen Effekten regelmässig die Informationen vom Sensor abgefragt und dann wird dies abgearbeitet. Wir haben uns hierbei extra gegen einen Interrupt gesteuerten Dateninput entschieden, weil dies das Effekthandling erschweren würde. So ist maximal ein Effekt zur gleichen Zeit am laufen und bricht ab, sobald man mit der Fernbedienung einen anderen Effekt auswählt.
Weiter kann man mit der Fernbedienung das Snake Spiel steuern und so die längste Schlange kreieren.

8 Aufbau des zweidimensionalen Bildschirmes

8.1 Bildschirm

8.2 Schieberegister

Wir bereits angesprochen helfen uns die Schieberegister die vielen verschiedenen Pins der RGB LED's anzusprechen. Die Schieberegister sind auf zwei Platinen angeordnet und sind mit einander mittels den entsprechenden outputs miteinander verknüpft. Wichtig war es dabei zu beachten, dass der Daten in- und output korrekt verlötet wurden, weil wir die Teilaufgaben unseres Projektes untereinander aufgeteilt haben und die Anordnung der Schieberegister mit dem Code übereinstimmen müssen.

8.3 Probleme beim Aufbau

Durch die anspruchsvolle Steuerung von 216 einzelnen Pins kommt es natürlich zu Schwierigkeiten. Unsere erste Schwierigkeit war, dass wir zuerst mit einer zu kleinen und einzelnen Platine gearbeitet haben. Dies hatte zur Folge, dass die Lötarbeit schnell unübersichtlich und ungenau wurde. Auch war es so extrem schwer, die Lötstellen sauber zu verlöten. Wir haben uns nach

einigem herumprobieren dafür entschieden, von einer auf zwei Platinen umzusteigen, was die Arbeit massiv vereinfacht hat.

9 Zusammenfassung

Quellen:

-