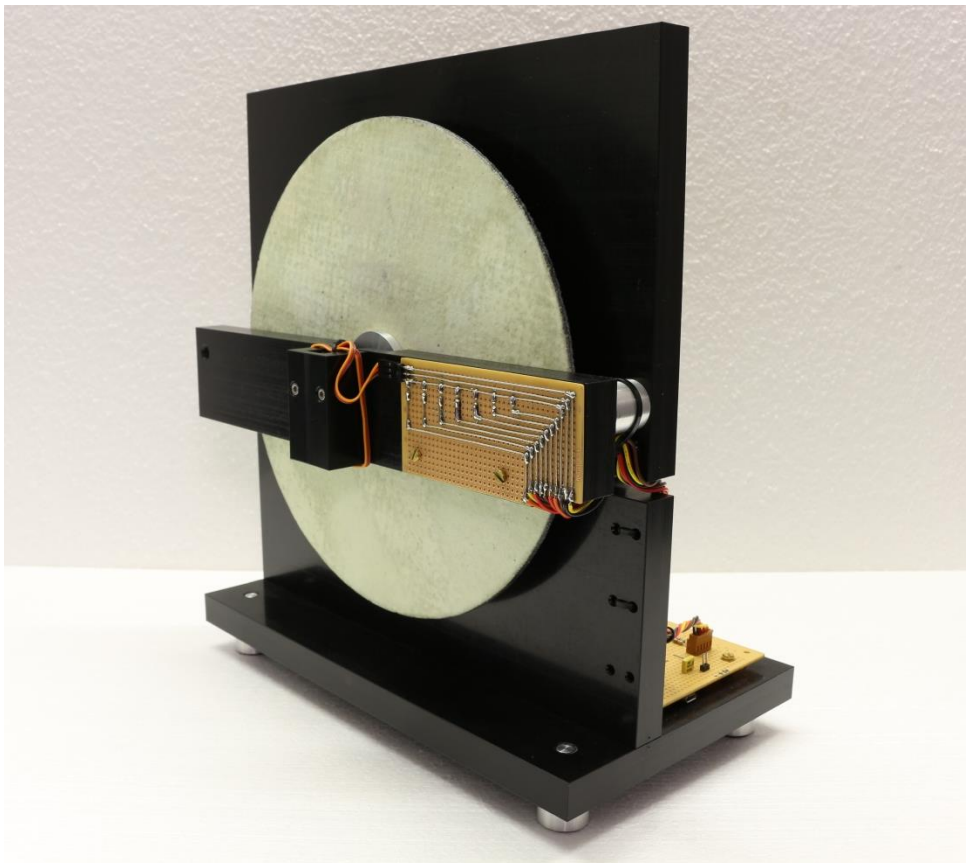
GYMNASIUM HILPOLTSTEIN

jugend forscht
Wir fördern Talente.

Strontiumaluminat-Display



Jugend Forscht 2018

Martin Weinzierl, Raphael Schuster
Gymnasium Hilpoltstein

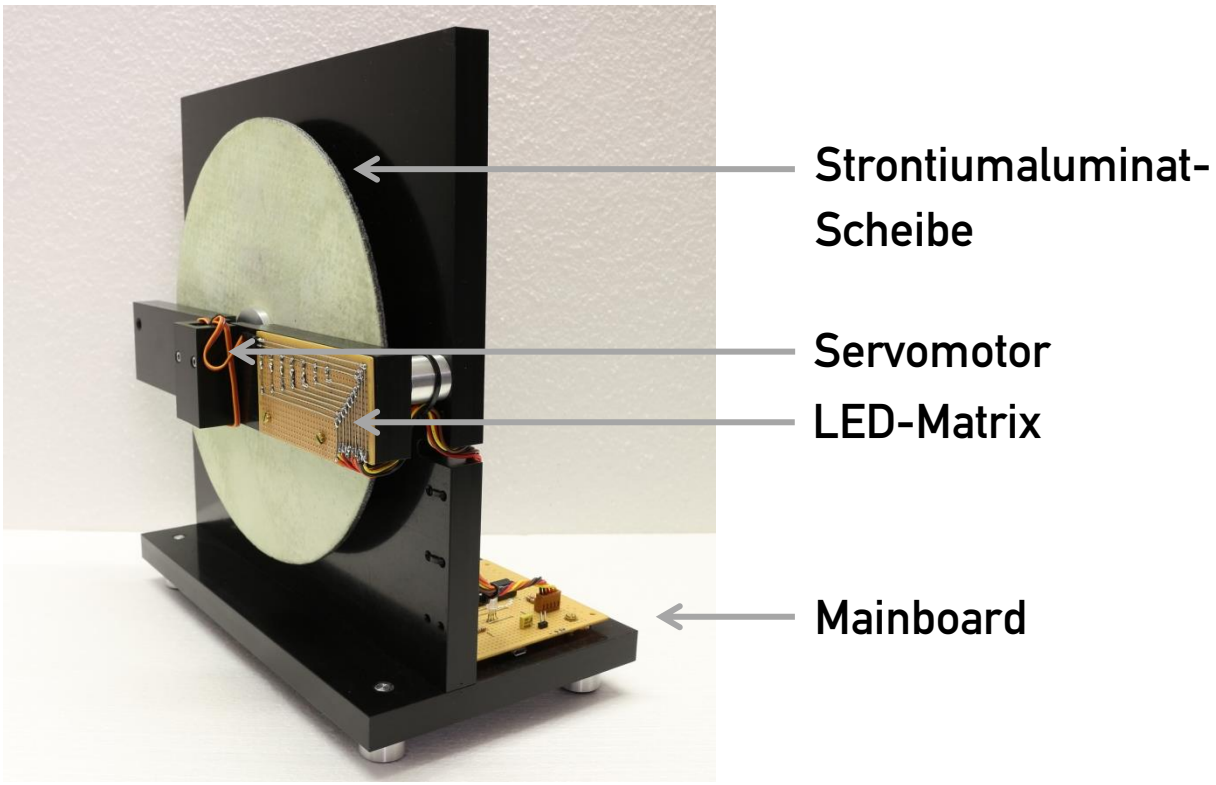
Thema: Strontiumaluminat-Display

Inhaltsverzeichnis:

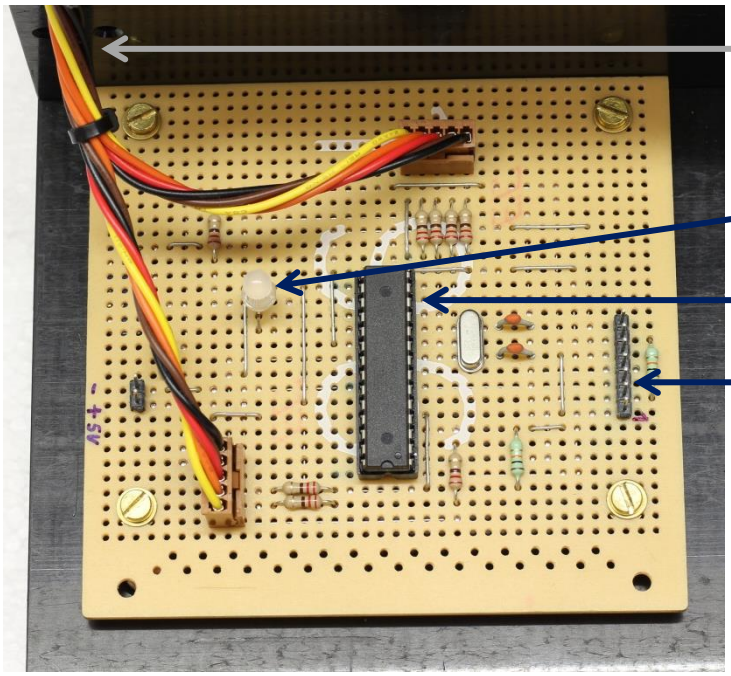
1. Projektbeschreibung

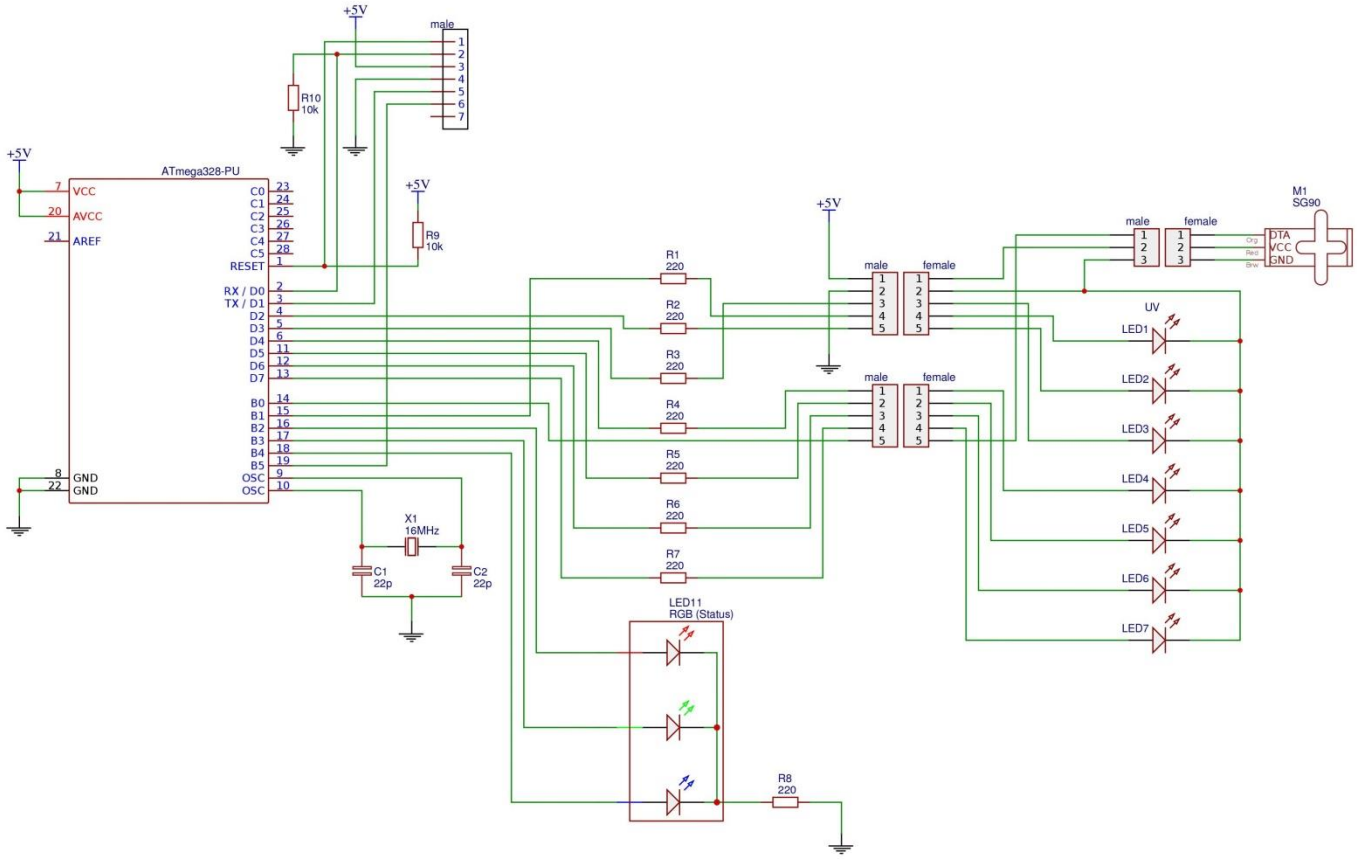
Wir haben ein Display gebaut, das mit einer LED-Matrix Buchstaben auf eine nachleuchtende Scheibe schreibt. Diese ist mit dem phosphoreszierenden Leuchtpulver Strontiumaluminat beschichtet. Die Schrift bleibt längere Zeit bestehen, ohne dass eine Stromzufuhr erfolgt. Wir dachten an eine Verwendung in dunkeln Räumen, z.B. als Kalender.

2. Aufbau



2.1. Aufbau des Mainboards





3. Programm

Wir haben das Programm zur Steuerung der LEDs selbst geschrieben. Um den Code übersichtlich zu halten und eine kurze Reaktionszeit des Motors zu gewährleisten, mussten wir zur Steuerung des Servomotors auf die Open Source Library Servo der Arduino LLC zurückgreifen.

Wir werden auf folgende Methoden genauer eingehen:

```
SERIALWAIT()
CHECKCHAR()
PRINT()
MOVE()
```

Die Auszüge aus dem Quelltext werden zum Verständnis gekürzt oder leicht verändert!

Der gesamte Code und Schaltplan kann unter folgender URL eingesehen und weiterverwendet werden:

https://goo.gl/wezfWD

Uns sind die Verbreitung und der Nachbau des Projekts sehr wichtig, deshalb geben wir den Code und die Pläne online frei.

3.1. Serieller Input als Textquelle

```
128   void SERIALWAIT() {
129     bool wait = true;
131     digitalWrite(blue, HIGH);
134     do {
135       if (Serial.available() > 0) {
136         string1 = Serial.readString();
140         wait = !wait;
141         digitalWrite(blue, LOW);
142       }
143     } while (wait == true && millis() <= delay6);
144     if (millis() >= delay6) {
147       string1 = string2;
148     }
150     Serial.end();
151   }
```

SERIALWAIT wird als erste Methode aufgerufen. Sie ist dafür zuständig am seriellen Input auf eine UTF8-Zeichenübertragung zu warten. Die Signal-LED leuchtet blau, während die Schleife auf Input wartet (Z.131 & 141).

Dadurch kann man ohne den Programmcode des Microcontrollers zu ändern variable Texte anzeigen lassen. Die Schleife (Z. 134 – 143) wird so lange ausgeführt, bis eine Übertragung eingeht (Z. 135) oder das Zeitlimit überschritten wird (Z. 143). Wenn dieser Fall eintritt, wird mit dem bereits in den Code integrierten Standardtext „**JuFo**“ fortgefahren (Z. 147). Man kann – falls eine Übertragung erfolgt ist – den Microcontroller vom FTDI-Chip (Sender) trennen. Der Microcontroller ist eine alleinstehende Schaltung, die bis zur Unterbrechung der Stromzufuhr den anfangs gesendeten Text im Speicher behält.

3.2. Zeichenüberprüfung

```
232   void CHECKCHAR() {
233     for (int i = 0; i <= string1.length(); i++) {
238       char1 = string1.charAt(i);
239       switch (char1) {
240         case 'A': case 'a': {
242             PRINT(B0111111);
243             PRINT(B1000100);
244             PRINT(B1000100);
245             PRINT(B1000100);
246             PRINT(B0111111);
248           } break;
249         case 'B': case 'b': {
251             PRINT(B1111111);
252             PRINT(B1001001);
253             PRINT(B1001001);
254             PRINT(B1001001);
255             PRINT(B0110110);
257           } break;
258         case 'C': case 'c': { ...
```

Die Methode **CHECKCHAR** verwendet die empfangene Zeichenkette aus **SERIALWAIT**. Sie überprüft jeden einzelnen Buchstaben (Z. 238) und führt anschließend die passenden Befehle aus, um die Zeichen auf das Display zu schreiben.

Man muss nicht darauf achten, stets Versalien (Großbuchstaben) zu verwenden. Das Programm kommt auch mit Gemeinen (Kleinbuchstaben) zurecht.

Aus den einzelnen Binärwerten der oben abgebildeten **PRINT**-Funktionen kann man die Buchstaben (grau hinterlegt) erkennen. Jede einzelne **1** stellt eine leuchtende LED dar.

3.3. Druck

```
172   void PRINT(byte dots) {
173     dots <<= 1;
177     PORTD = dots;
178     digitalWrite(blue, HIGH);
179     delay(delay1);
180     digitalWrite(blue, LOW);
181     PORTD = 0;
185     MOVE(angle1);
186   }
```

Die **PRINT**-Funktion benötigt einen Binärwert, der die anzuschaltenden LEDs angibt. Wir verwenden sieben UV-LEDs. Da ein Binärwert eines **PORT**-Registers aber acht Bits verarbeitet, muss der Wert verschoben werden (Z. 158). Pin **0** wird nämlich nicht verwendet.

```
B00111111 --> B01111110
```

Das letzte Bit (kann auch weggelassen werden) wird gelöscht und eine **0** rechts angefügt.

Anschließend wird der Wert auf das **PORTD**-Register übertragen (Z.177), wodurch die LED-Matrix zu leuchten beginnt. Die Belichtungszeit kann man durch die Variable **delay1** anpassen. Die Funktion **MOVE** (Z. 185) bewegt anschließend den Servomotor weiter, damit eine neue Reihe geschrieben werden kann.

3.4. Bewegen des Servomotors

```
188   void MOVE(int addangle) {
189     int newangle = angle + addangle;
190     if (newangle > 180) {
191       newangle = 180;
192     } else if (newangle < 0) {
193       newangle = 0;
194     }
195     int invangle = MAP(newangle, 0, 180, 180, 0);
196     servo.write(invangle);
197     delay(delay5);
198     angle = newangle;
199   }
```

Die Funktion **MOVE** benötigt einen Winkel der zu dem bereits bestehenden Winkel addiert werden soll (Z. 188f). Wenn der Winkel größer oder kleiner als der Bewegungsbereich des Servomotors ist, wird der Wert auf das Maximum oder Minimum des Wertes gesetzt (Z.190 – 194). Unsere Schrift wird von links nach rechts (auf unserer Scheibe: mathematisch positiver Umlauf- oder Drehsinn) gelesen, der Motor bewegt sich aber in die entgegengesetzte Richtung, deshalb muss der Wert durch die **MAP**-Funktion (Z. 195) invertiert werden.

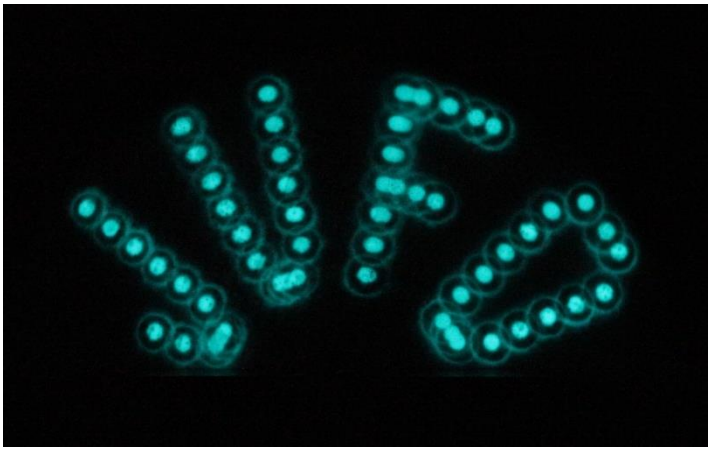
Es wird folgende Formel verwendet:

$$\texttt{MAP(x,a,b,c,d)} = \frac{(x-b)*(c-d)}{(a-b)+d}$$

Anschließend wird der neue Winkel als **angle** in der gleichnamigen Variablen gespeichert (Z. 198).

4. Schlussfolgerung

Unser Modell hat sich im Praxistest sehr gut als Kalender bewiesen. Dafür war in unserem Fall jedoch ein Computer nötig, um dem Microcontroller genaue Daten zu liefern.



Wir dachten ebenfalls an eine Verwendung als Warnschild. Die grüne neonähnliche Farbe kann man bereits von weitem erkennen. Diese ist auch im Falle eines Stromausfalls weiter lesbar.

5. Ausblick

Weitere Zusatzmodule für die 1x7-Pin-Schnittstelle sind geplant:

- Infrarot-Modul

  Microcontroller lässt sich durch eine normale Fernbedienung (Buchstaben auf dem Ziffernblock) steuern

- Display-Modul

  Texte lassen sich durch einen zweiten Microcontroller und ein LCD-Display auswählen

6. Materialliste

1 x ATmega328-PU Microcontroller
7 x UV-LED (5 mm)
1 x RGB Status LED
2 x 10 kΩ (¼ W) Widerstand
8 x 220 Ω (¼ W) Widerstand
2 x 22 pF Keramikkondensator
1 x 16 MHz Quarz
1 x SG90 Servomotor
1 x lackierbare Scheibe (Ø = ca. 24 cm)
1 x Spraydose Klarlack
40 g Strontiumaluminat
div. Verbindungskabel
div. Polyoxymethylen-Teile

7. Beteiligte Personen

Martin Weinzierl	1. Gruppenmitglied
Raphael Schuster	2. Gruppenmitglied
Johann Weinzierl	Projektbetreuer (privat)
Dipl. Phys. Philipp Weinhardt	Projektbetreuer
Dr. Stefan Strömsdörfer	Physikunterricht zur Phosphoreszenz
Ernst Sindel	Informatikunterricht zu den Kontrollstrukturen
Sebastian Dorr	Idee für die Nutzung der 1x7-Pin-Schnittstelle für Zusatzmodule

Schriftart entspricht DIN 1451 Mittelschrift