```
1  //07_Larson Scanner
2
3  //Bei diesem Programm handelt sich es um einen Larson-Scanner mit PWM.
4  //Die Position stellt den hellsten Punkt der 16 LED's dar die umliegenden LED's (+-2) sollen
   dunkler erscheinen durch PWM.
5  //Die restlichen LED's sollen leicht klimmen aka Night-Rider .
6  //In der Funktion pulse_width_modulation dient die erste Schleife (zeit) dazu alles zu
   verzögern um somit die Geschwindigkeit des Scanners zu modifizieren,
7  //die zweite Schleife (pwm) zählt die 32 Abstufungen damit ist gemeint das der hälste Punk
   bei allen Durchgängen den Zustand HIGH besitzt die nebenliegenden nur noch die ersten 16
   usw... ,
8  //die dritte Schleife (led) prüft Anhand der Funktion brightness den Zustand der 16 LED's
9  //Die Brightness Funktion ermittelt durch den absoluten Abstand vom hellsten Punkt (pos) die
   helligkeit der LED und gibt den jeweiligen Wert zurück wie oft die LED den Zustand HIGH von
   32x besitzt
10 //was mit den Wert pwm verglichen wird (32 Abstufungen) um so HIGH oder LOW in das Bit-Array
   zu schreiben.
11
12 int datenPin = 8;       // SER
13 int speicherPin = 9;    // RCLK
14 int taktPin = 10;        // SRCLK
15 int var[16] = {0};      // Bit Array
16
17
18 void setup()
19 {
20     pinMode(taktPin, OUTPUT);
21     pinMode(speicherPin, OUTPUT);
22     pinMode(datenPin, OUTPUT);
23 }
24 ////////////////////////////////////////////////////////
25 void loop()
26 {
27     static int pos=0;  // Position des hellsten Punktes (LED)
28
29     while(pos<16) // Von Position 0 bis Position 15 = vorwärts
30     {
31         pulse_width_modulation(pos);
32         ++pos;
33     }
34     // Übergang wenn pos = 16
35
36     while(pos>0) // Von Position 15 bis Position 0 = rückwärts
37     {
38         --pos;
39         pulse_width_modulation(pos);
40     }
41     // Übergang bei pos = 0
42 }
43 ////////////////////////////////////////////////////////
44 void pulse_width_modulation(const int pos)
45 {
46     for(int zeit=0; zeit<1; ++zeit) // Verzögerung für Geschwindigkeit des sich zu bewegenden
   Punktes
47     {
48         for (int pwm=0; pwm<32; ++pwm) // Pulsweitenmodulation der LED's (32 Abstufungen)
49         {
50             for (int led=0; led<16; ++led) // Durchläuft Anzahl der LED's
51             {
52                 var[led] = (brightness(led, pos) > pwm); // Zustände Led speichern in Bit
   Array
53             }
54             setzePins(var);      // Daten in Speicherregister schieben nachem alle Zustände
   ermittelt wurden
55             digitalWrite(speicherPin, HIGH); // Pins Speicherregister setzen
56         }
57     }
58 }
59 ////////////////////////////////////////////////////////
60 uint8_t brightness(const int led, const int pos) // Feststellung wie weit LED von Pos
   entfernt = Helligkeit
61 {
```

```
62      switch (abs(led-pos)) // Helligkeit feststellen
63      {
64          case 0:     return 32;
65          case 1:     return 16;
66          case 2:     return 6;
67          case 3:     return 2;
68          default:    return 1;
69      }
70  }
71  // Alle Pins am Schieberegister auf LOW setzen
72  void resetPins()
73  {
74      digitalWrite(taktPin, LOW);
75      digitalWrite(speicherPin, LOW);
76      digitalWrite(datenPin, LOW);
77  }
78  // Übertragung Daten Array an Schieberegister
79  void setzePins(int daten[])
80  {
81      for(int i=16; i>=0; i--)
82      {
83          resetPins();
84          digitalWrite(datenPin, daten[i]);
85          digitalWrite(taktPin, HIGH);
86      }
87  }
```