


Department Informations- und Elektrotechnik	Labor für Bussysteme und Sensorik	 HAW Hamburg Fakultät TI Technik und Informatik
Studiengruppe: BUP/01	BUP	Protokollführer: Martin Pyka Konstantin Meyer
Semester: WS19/20		
Professor: Prof. Dr. -Ing Robert Fitz	Testat:	
NeoPixel Snake Game		

Projektbericht

Inhaltsverzeichnis

1	Einleitung	1
2	Grundlagen	1
2.1	NeoPixel	1
2.1.1	Stromversorgung	2
2.2	Mikroprozessor	3
3	Umsetzung Hardware	3
3.1	Hardware Komponenten	3
3.2	Aufbau	3
4	Umsetzung Software	4
4.1	Webserver	4
4.1.1	Verbindung	4
4.1.2	Hauptseite	4
4.2	Spiel	4
4.2.1	Spielfeld	4
4.2.2	Schlange	5
4.2.3	Besonderheiten und Spielstart	5
5	Quellcode	5
5.1	Webserver	5
5.2	Snakespiel	6
6	Quellen	11

Abbildungsverzeichnis

1	Snake auf einem Telmac 1800 im Jahr 1978	1
2	Sequence Char	2
3	Data Transmission	2
4	Schaltbild: Anschlüsse an die Neopixel	3
5	ESP32 Mikroprozessor	3
6	Blockschaltbild	4
7	Anordnung der LEDs	5

1 Einleitung

Snake ist eins der ältesten und bekanntesten Videospiele. Das Prinzip wird in verschiedenen Varianten für viele Plattformen angeboten. Bei Snake geht es darum, mit einer Schlange auf einem viereckigen Spielfeld Futterteile einzusammeln. Mit jedem Teil wächst die Schlange. Die Herausforderung besteht darin, dass sich die Schlange nicht selber fressen darf.

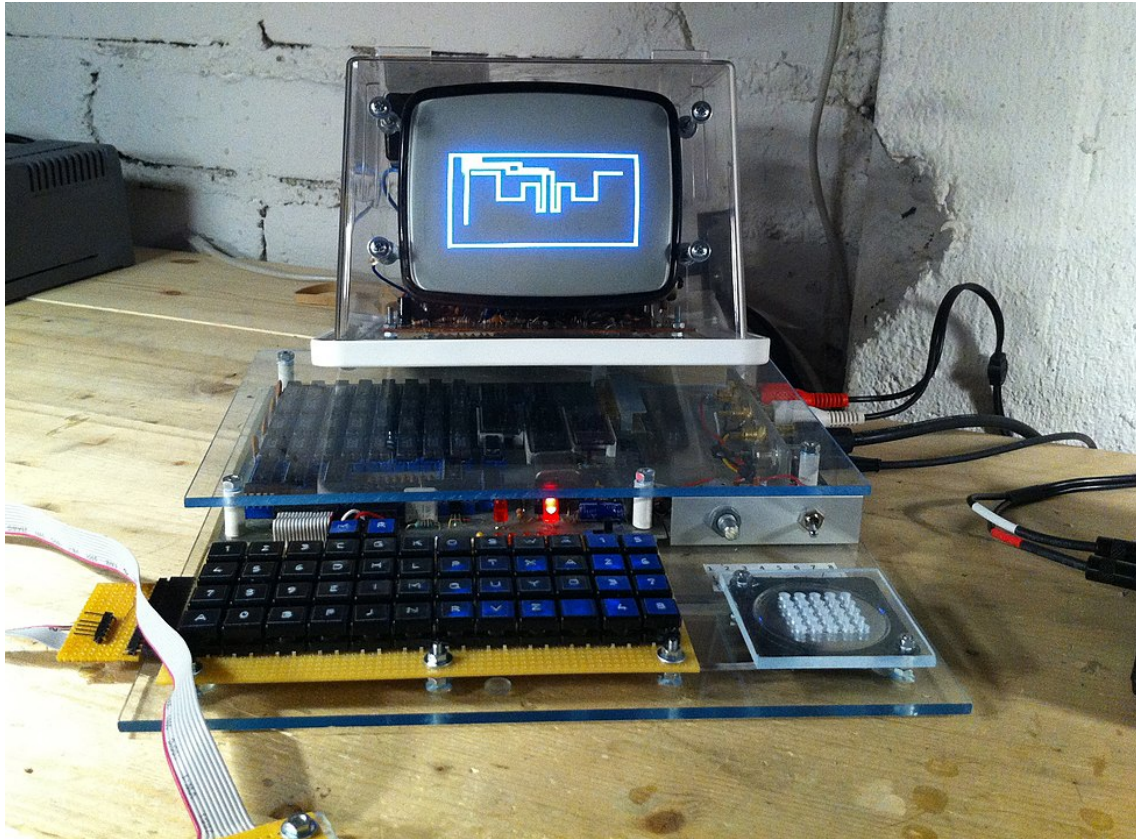


Abbildung 1: Snake auf einem Telmac 1800 im Jahr 1978¹

Ziel ist es ein Snake Spiel mit Hilfe von LEDs zu erstellen. Das Spielfeld soll durch die LEDs dargestellt werden und die Steuerung der Schlange soll dabei von einem Mobilgerät möglich sein.

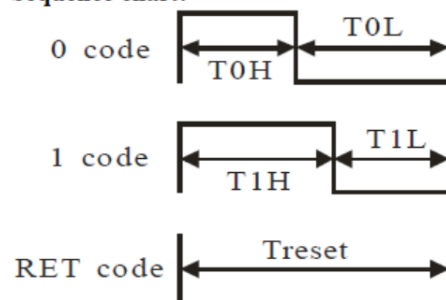
2 Grundlagen

2.1 NeoPixel

Neopixel sind digital adressierbare LEDs die in Verschiedenen Bauformen vorhanden sind. Für unser Projekt haben wir uns für die neuste Neopixel Generation die WS2812 entschieden. WS2812 sind LEDs, die mit Rotem, Grünem und Blauem Leuchtmittel und einem Treiberchip in ein winziges SMD Gehäuse integriert sind.

Mit digital adressierbaren LEDs kann eine große Anzahl von LEDs über die digitale Kommunikation mit kleinen integrierten Chips steuern, die diese digitalen Befehle lesen. Daten und Taktung erfolgen beim WS281xx über eine einzelne Leitung. Normalerweise liegt die Frequenz bei 800 KHz, was einer Taktperiode von 1.25 µsecs entspricht. Die Übermittlung einzelner Bits an einen WS2812 kommt über Rechtecksignale zustande. Um eine "1" zu Senden wird für 0.85 µsec High gesendet und für 0.4 µsec ein Low-Signal gesendet.

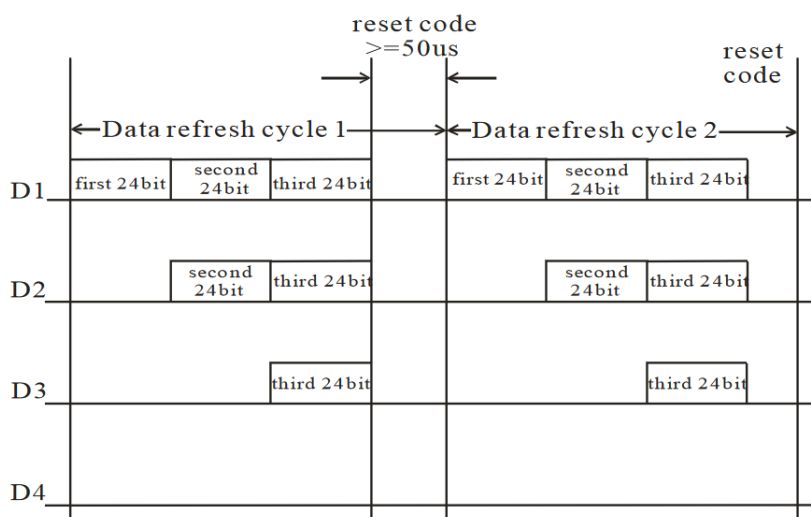
¹<https://commons.wikimedia.org/w/index.php?curid=39985153>

Sequence chart:Abbildung 2: Sequence Chart ²

Für eine "0" wird für 0.4 μsec ein High-Signal und für 0.85 μsec ein Low gesendet.

Diese Kommunikationsart ist auch als NZR (Non Zero Return) bekannt. Das Datenpaket an einen Pixel beträgt 24-Bit (8 Bits pro Farbe) und ist in der Reihenfolge (R : Rot, G: Grün, B: Blau) kodiert:

R7, R6, ..., R0, G7, G6, ..., G0, B7, B6, ..., B0

Data transmission method:Abbildung 3: Data Transmission ²

Jede Farbe nimmt einen Wert von 0-255 an.

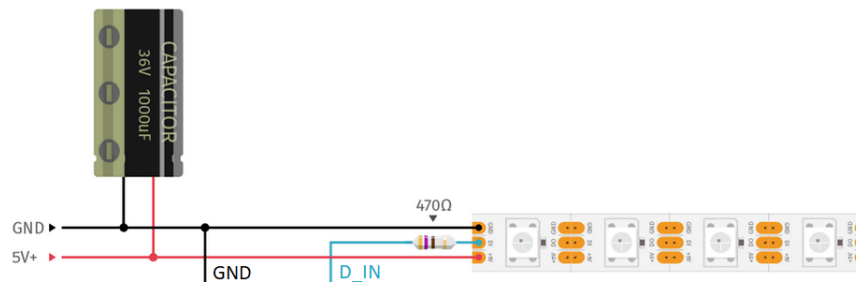
Die Datenpakete werden sequenziell über die Leitung geschickt, d.h. in jedem Zyklus werden die Daten an den Nachfolgenden Pixel weitergegeben. Sobald alle LEDs ihre Daten erhalten haben folgt eine Pause von 50 μsec und die Daten werden in den Datenlatches der IC's gespeichert und die LED leuchtet gemäß der Daten.

Die Bildwiederholfrequenz ist relativ niedrig (ca. 400 Hz) und so für flüssige Bewegungen nicht geeignet, für ein Snake-Spiel jedoch ausreichend.

2.1.1 Stromversorgung

Jede WS2812 LED benötigt bis zu 60mA. Es ist also ratsam, ein Netzteil zu verwenden. Dieses Netzteil sollte mit einem Elektrolyt-Kondensator unterstützt werden. Der GND des Netzteils und der GND des ESP32 müssen verbunden werden.

²<https://cdn-shop.adafruit.com/datasheets/WS2812.pdf>

Abbildung 4: Schaltbild: Anschlüsse an die Neopixel ³

2.2 Mikroprozessor

Als zentrale Steuereinheit wird ein ESP32 verwendet. Trotz seiner geringen Anschaffungskosten bietet dieser Mikroprozessor mit einem Xtensa 32-bit LX6 Controller und einem 8 MHz Takt eine sehr gute Performance. Zusätzlich hat der ESP32 einen On-Board WiFi-Chip, was weitere Hardware erspart.

Abbildung 5: ESP32 Mikroprozessor ⁴

3 Umsetzung Hardware

3.1 Hardware Komponenten

Neben dem NeoPixel LED-Streifen und dem Controller sind noch weitere Bauteile erforderlich. Aufgrund des Prototyp Charakters werden dafür Teile aus dem Labor / privatem Besitz verwendet. Insgesamt werden folgende Materialien benötigt:

- 5V Netzteil
- 470Ω Widerstand
- NeoPixel LED-Streifen
- ESP32
- Verdrahtungsteile (Laborleitungen, USB-Kabel, Breadboard etc.)

3.2 Aufbau

Die Bauteile werden entsprechend dem unten Aufgeführten Blockschaltbild verbunden. Es muss darauf geachtet werden, dass bei der Verbindung von ESP32 zum LED-Streifen zusätzlich zum Datenkanal auch die Masse Verbunden wird, damit der Stromkreis geschlossen wird.

³<https://starthardware.org/viele-leds-mit-arduino-steuern-ws2812/>

⁴https://www.mouser.de/images/espressifsystems/lrg/ESP32-DevKitC_t.jpg

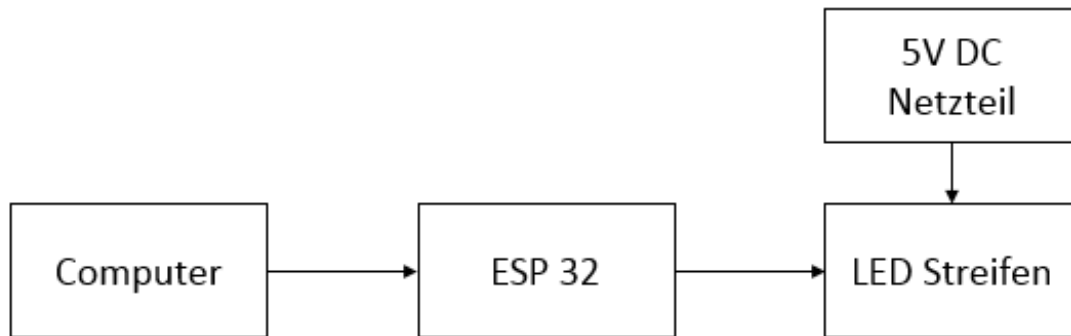


Abbildung 6: Blockschaltbild

4 Umsetzung Software

Die Software haben wir in logische Abschnitte geteilt. So haben wir den Webserver und dessen Implementierung getrennt von der Umsetzung des Spiels entwickelt und am Ende beide Teile zu einem lauffähigen Code Zusammengefügt.

4.1 Webserver

4.1.1 Verbindung

Der Mikroprozessor erzeugt einen drahtlosen Zugangspunkt (Wireless Access Point) deren SSID Name *Snake_HAW* heißt.

Der AP (Access Point) ist auf einen Client limitiert, somit lässt der Mikroprozessor nicht zu, dass sich zwei Clients gemeinsam mit dem Mikroprozessor verbinden.

Zum Verbinden braucht es den SSID-Namen und das Passwort: *haw.1234*. Nach erfolgreichem Verbinden kann die Hauptwebseite entweder über die IP-Adresse zb. *http://192.168.4.1/* oder über die Domain *http://SnakeGame.local* erreicht werden.

Um den SSID-Namen oder die IP-Adresse herauszufinden, kann man über den Seriellen Monitor, mit einer Baudrate von 115200, die Ausgaben bei System Start prüfen.

Die Domain *http://SnakeGame.local* wird vom Mikroprozessor selbst aufgelöst und leitet automatisch auf die Hauptwebseite, da ein DNS Dienst im Mikroprozessor vorhanden ist.

4.1.2 Hauptseite

Auf der Hauptseite kann man die Schlage steuern.

Man kann auf dem Display durch einfaches klicken und ziehen in eine bestimmte Richtung die Schlange steuern. Die Steuerung besteht aus einem JavaScript welches hier zu finden ist [<https://github.com/jeromeetienne/virtualjoystick.js>]. Das JavaScript wird an den Client geschickt und im Browser ausgeführt. Das Script sendet die aktuelle Richtung automatisch an den Mikroprozessor. Dort werden die Werte dann weiter verarbeitet.

4.2 Spiel

4.2.1 Spielfeld

Das Spielfeld wird im Spiel als Zweidimensionales Array gespeichert. Je nachdem welcher Wert in einem Feld enthalten ist, wird die entsprechende LED angesteuert. Blau steht für Schlangenteile und Rot für das Futter. Leere Felder leuchten nicht. Da der LED-Streifen wie unten dargestellt verlötet ist, muss eine Fallunterscheidung zwischen geraden und ungeraden Zeilen in der Ansteuerung implementiert werden.

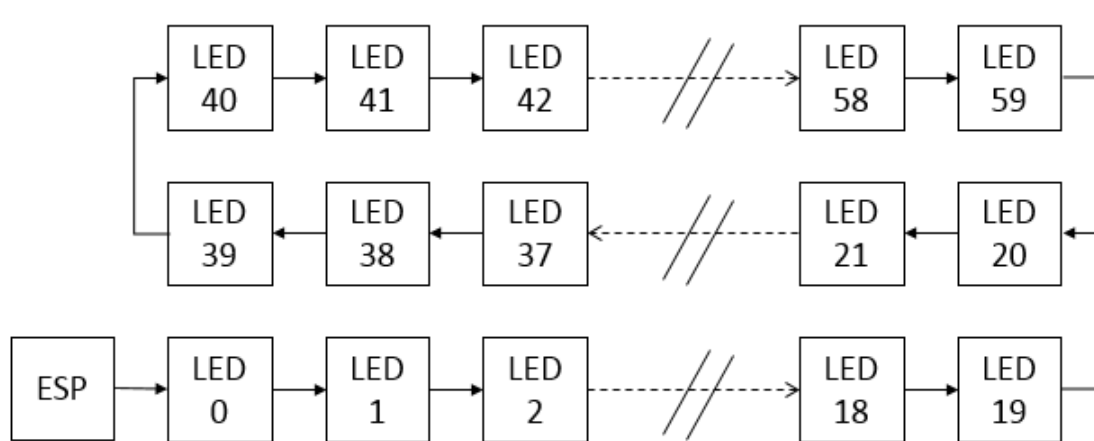


Abbildung 7: Anordnung der LEDs

In der untersten Zeile ist die Nummerierung wie man sie erwarten würde. Das Feld mit der Koordinate (0,0) hat auch die LED 0. Eine Zeile darüber wird es komplizierter. Hier ist die Nummerierung umgedreht. Das Feld (0,1) würde man an LED 20 vermuten es befindet sich jedoch an Adresse 39.

4.2.2 Schlange

Die Schlange ist im Programm durch eine Liste dargestellt. Jedes Element der Schlange ist ein eigenes Element der Liste. Bei einem Schritt wird das letzte Element abgebaut und vorne in der entsprechenden Richtung ein neues erzeugt. Kann vorne kein neues Element erzeugt werden weil das Feld bereits durch ein Teil der Schlange belegt ist, so hat man verloren. Ist das Feld auf dem sich das neue Schlangenteil befindet durch ein Futterstück belegt, so wird das letzte Stück nicht gelöscht und die Schlange wächst um ein Segment.

4.2.3 Besonderheiten und Spielstart

Da diese Prototypversion ausschließlich über ein einfaches User Interface zur Vorgabe der Richtung verfügt, wird kein Highscore und Spielstand ausgegeben. Um das Spiel zu starten muss entweder nach dem Hochfahren oder nach einem Game-Over der Joystick nach Rechts bewegt werden.

In dieser Version von Snake ist das Feld nicht durch Wände begrenzt. Läuft die Schlange an einer Seite aus dem Feld so kommt sie auf der gegenüberliegenden Seite wieder hinein.

5 Quellcode

5.1 Webserver

Die HTML Seite und das Virtual Joystick wurden für den Bericht gekürzt. Das gesamte Projekt ist auf Github zu finden ⁵

```

1 #include "main.h"
2 #include "webserv.h"
3 #include <WiFi.h>
4 #include <ESPmDNS.h>
5 #include <ESPAsyncWebServer.h>
6
7 AsyncWebServer server(80);
8
9 const char *ssid = "Snake_HAW";
10 const char *password = "haw.1234";
11
12 const char index_html[] PROGMEM = "<html> <head> <meta charset=\"utf-8\">
    <meta name=\"viewport\" content=\"width=device-width,
    user-scalable=no, minimum-scale=1.0, maximum-scale=1.0\"> ...
13

```

⁵https://github.com/BalooSLU/neopixel_snake_game


```

14     const char virtualjoystick_js[] PROGMEM = "var VirtualJoystick\t=
function(opts)\n{\n\topts\t\t\t\t= opts\t\t\t\t||
{};\n\t\tthis._container\t\t= opts.container\t\t|| document.body;" ...
15
16     void notFound(AsyncWebServerRequest * request)
17 {
18     request->send_P(404, "text/plain", "Not found");
19 }
20
21 void init_web(void)
22 {
23     Serial.begin(115200);
24     WiFi.softAP(ssid, password);
25     Serial.println();
26     Serial.print("IP address: ");
27     Serial.println(WiFi.softAPIP());
28     if (!MDNS.begin("SnakeGame"))
29     {
30         Serial.println("Error setting up MDNS responder!");
31         while (1)
32         {
33             delay(1000);
34         }
35     }
36     server.on("/", HTTP_GET, [](AsyncWebServerRequest *request) {
37         request->send_P(200, "text/html", index_html);
38     });
39     server.on("/joystick.html", HTTP_GET, [](AsyncWebServerRequest
*request) {
40         request->send_P(200, "text/html", index_html);
41     });
42     server.on("/virtualjoystick.js", HTTP_GET, [](AsyncWebServerRequest
*request) {
43         request->send_P(200, "application/javascript",
virtualjoystick_js);
44     });
45
46     // Handle a PUT request
47     server.on("/left", left);
48     server.on("/right", right);
49     server.on("/up", up);
50     server.on("/down", down);
51     server.onNotFound(notFound);
52     server.begin();
53     MDNS.addService("http", "tcp", 80);
54 }

```

Listing 1: Code für den Webserver

5.2 Snakespiel

```

1  /*****
2  Snake Spiel
3  Konstantin Meyer & Martin Pyka
4  *****/
5  #include <NeoPixelBus.h>
6  #include <ESPAsyncWebServer.h>
7
8

```

```
9 #define WIDTH 20
10 #define HEIGHT 12
11 #define ZERO(v) v = 0
12 #define LED_COUNT 240
13 #define LED_PIN 23
14
15 typedef enum {
16     UP,
17     DOWN, LEFT, RIGHT
18 } direction;
19
20 typedef struct snake_part {
21     short x;
22     short y;
23     struct snake_part *next;
24 } snakePart;
25
26 typedef struct {
27     int points;
28     char name[20];
29 }highscoreEntry;
30
31
32 const int menu_items = 3;
33 const char highscoreFile[] = {"bestenliste"};
34 volatile direction move = DOWN;
35 int field[WIDTH][HEIGHT];
36
37 NeoPixelBus<NeoGrbFeature, Neo800KbpsMethod> strip(LED_COUNT, LED_PIN);
38
39 void Sleep(int time) {
40     int i = millis() + time;
41     while(millis() < i);
42 }
43
44 void neoPixel_setup()
45 {
46     strip.Begin();
47     strip.Show();
48 }
49
50 //Funktion gibt das Spielfeld aus
51 void printField(int points)
52 {
53     int y, x;
54     RgbColor red(128,0,0);
55     RgbColor black(0);
56     RgbColor blue(0,0,128);
57     RgbColor c(0);
58
59     for (y = 0; y < HEIGHT; y++)
60     {
61         for (x = 0; x < WIDTH; x++)
62         {
63             switch (field[x][y])
64             {
65                 case 0:
66                     c = black;
67                     break;
```

```
68         case 1:
69             c = blue;
70             break;
71         case 2:
72             c = red;
73             break;
74     }
75     if (y % 2 == 0) {
76         strip.SetPixelColor((y*WIDTH)+x, c);
77     } else {
78         strip.SetPixelColor(((y+1)*WIDTH)-x-1, c);
79     }
80
81
82     }
83
84     }
85     strip.Show();
86 }
87
88 //Funktion erzeugt ein neues Futerteil auf dem Spielfeld
89 void food()
90 {
91     int x_rand, y_rand, result = 1;
92     do
93     {
94         x_rand = rand() % (WIDTH); //Diese Zufallsfunktion erzeugt keine
95         gleichm ig verteilten Zufallszahlen
96         y_rand = rand() % (HEIGHT); //f r eine nicht
97         sicherheitskritische Anwendung wie hier ist sie jedoch hinreichend.
98
99         if (field[x_rand][y_rand] == 0)
100         {
101             field[x_rand][y_rand] = 2;
102             result = 0;
103         }
104     } while (result);
105 }
106
107 //Schrittfunktion bewegt die Schlange einen Schritt weiter und reagiert
108 auf entsprechende Besonderheiten (GameOver, Futter) gibt 0 bei
109 niederlage zur ck
110 int step(snakePart **first, snakePart **last, int *points)
111 {
112     snakePart *newPart, *temp;
113     int x, y, x_new, y_new, state;
114
115     switch (move)
116     {
117     case UP:
118         x = 0;
119         y = 1;
120         break;
121     case DOWN:
122         x = 0;
123         y = -1;
124         break;
125     case LEFT:
126         x = -1;
```

```

123     y = 0;
124     break;
125     default:
126         x = 1;
127         y = 0;
128         break;
129 }
130 x_new = ((*first)->x + x + WIDTH) % WIDTH;
131 y_new = ((*first)->y + y + HEIGHT) % HEIGHT;
132
133 state = field[x_new][y_new];
134
135 if (state != 1)
136 {
137     newPart = (snakePart *)malloc(sizeof(snakePart)); //Speicher
//allokieren f r neues Schlängenteil
138     if (newPart == NULL)
139         return 0;
140     newPart->x = x_new;
141     newPart->y = y_new;
142     field[x_new][y_new] = 1;
143
144     (*first)->next = newPart;
145     (*first) = newPart;
146
147     if (state == 0)
148     {
149         temp = (*last)->next;
150         field[(*)last->x][(*)last->y] = 0;
151         free(*last);
152         *last = temp;
153     }
154     else
155     {
156         food();
157         *points += 1;
158     }
159     return 1;
160 }
161 else
162     return 0;
163 }
164
165 //interrupthandler f r left
166 void left(AsyncWebServerRequest *request)
167 {
168     Serial.println("Left");
169     move = LEFT;
170     request->send_P(200, "text/plain", "");
171 }
172
173 //interrupthandler f r right
174 void right(AsyncWebServerRequest *request)
175 {
176     Serial.println("Right");
177     move = RIGHT;
178     request->send_P(200, "text/plain", "");
179 }
180

```

```
181 //interrupthandler f r up
182 void up(AsyncWebServerRequest *request)
183 {
184     Serial.println("Up");
185     move = UP;
186     request->send_P(200, "text/plain", "");
187 }
188
189 //interrupthandler f r down
190 void down(AsyncWebServerRequest *request)
191 {
192     Serial.println("Down");
193     move = DOWN;
194     request->send_P(200, "text/plain", "");
195 }
196
197 //Funktion zur initialisierung des Spiels
198 void gameInit(snakePart **first, snakePart **last)
199 {
200     int x, y;
201
202     for (x = 0; x < WIDTH; x++)
203     {
204         for (y = 0; y < HEIGHT; y++)
205         {
206             ZERO(field[x][y]);
207         }
208     }
209
210     *first = (snakePart *)malloc(sizeof(snakePart));
211     (*first)->x = 1;
212     (*first)->y = 2;
213
214     *last = (snakePart *)malloc(sizeof(snakePart));
215     (*last)->x = 1;
216     (*last)->y = 1;
217
218     (*last)->next = *first;
219     field[1][1] = 1;
220     field[1][2] = 1;
221     field[1][3] = 2;
222 }
223
224 //Funktion zum starten des Spiels
225 void gameStart(void)
226 {
227     snakePart *first = NULL, *last;
228     int points = 0, resume;
229     move = DOWN;
230     Serial.println("GameStart Waiting for RIGHT");
231     while(move!=RIGHT);
232
233     gameInit(&first, &last);
234
235     Serial.println("GameINIT succesfull");
236     do
237     {
238         resume = step(&first, &last, &points);
239         printField(points);
```

```
240         Sleep(400);  
241     } while (resume);  
242 }
```

Listing 2: Code für das Snakespiel

6 Quellen

https://github.com/BalooSLU/neopixel_snake_game
<https://cdn-shop.adafruit.com/datasheets/WS2812.pdf>
<https://learn.adafruit.com/adafruit-neopixel-uberguide>
<https://starthardware.org/viele-leds-mit-arduino-steuern-ws2812/>
<https://github.com/jeromeetienne/virtualjoystick.js>
https://www.mouser.de/images/espressifsystems/lrg/ESP32-DevKitC_t.jpg
Von jpl - Eigenes Werk, CC BY-SA 4.0,
<https://commons.wikimedia.org/w/index.php?curid=39985153>