```c
/*Anzeige eines zufälligen Buchstabens mit Morsecode.
 * Nach einer einstellbaren Zeit zwischen 1 und 7 Minuten ertönt das Stop Signal "HALT".
 * Der Vorgang kann vorher durch Drücken der Taste beendet werden.
 * Die Anzahl der gespielten Buchstaben wird angezeigt.
 * Es werden nur Buchstaben ausgewählt, die noch nicht "gewürfelt" wurden.
 * Als Anzeigedisplay wird ein E-Paper Displays 1,5x1,5Zoll verwandt.
 * Durch den großen Bedarf an Speicher für die Buchstabendarstellung
 * müssen etwa 50kByte Speicher vorhanden sein.
 *
 * Die Batterispannungsanzeige beginnt bei >3V(100%), und endet bei 2,2V(alles leer).
 * Die Schaltung funktioniert noch unter 2,2V, aber das Piepston hört bei 2,4V auf.
 *
 * Prozessor: ATMEGA 644  h-Fuse DFh; l-Fuse CCh
 * Version 0.0 4/18 pkr*/


#define F_CPU 4000000UL

#include <stdlib.h>
#include <string.h>
#include <avr/io.h>
#include <avr/wdt.h>
#include <avr/eeprom.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include "bibliothek/e-paper-1,54.c"


#define Sec_Faktor 47180        //46875 bei 4,0...MHz 3s (Frequenz/256)
#define Zeichenlaenge 70    //Punktlänge in ms für Morseausgabe
#define Tim_Int_ein (TIMSK1 |= (1<<TOIE1))
#define Tim_Int_aus (TIMSK1 &= ~(1<<TOIE1))
#define Geraeteport PORTC
#define Tasten PINC
#define Start_Taste 1
#define Summer      2
#define Minuten0    3
#define Minuten2    4
#define Minuten4    5
#define Spannung_Kanal0    0
#define Vorwahl_Minuten ((Tasten & (1<<Minuten0)) | (Tasten & (1<<Minuten2)) | (Tasten & (1<<
Minuten4)))

#define Start_Taste_an !(Tasten & (1<<Start_Taste))
#define Summer_an (Geraeteport |= (1<<Summer))
#define Summer_aus (Geraeteport &= ~(1<<Summer))

//UP
uint8_t Tastenabfrage();    //Wenn Taste gedrückt: 1, andernfalls 0
uint8_t Minutenabfrage();   //Abfrage der eingestelleten Minuten
void Morsezeichen(uint8_t Zeichen); //Gibt Morsezeichen aus. Zeichentabelle in Morsetabelle
void Morsetext(uint8_t Text[], uint8_t laenge); //Gibt Morsetext aus
uint8_t Spielbeginn(uint8_t Minuten);   //Rückgabe ist der zufällige Buchstabe
void Spielende();
void Grundinitialisierung();
void Grundtexte();
void Schreibe_Zahl_zweistellig(uint8_t Zahl, uint8_t X_Wert, uint8_t Y_Wert);
void Anfangsbildschirm();
uint16_t ADC_Wert();
void Batterieanzeige(uint16_t ADC_ausgabe);


//EEPROM
uint8_t ee_Takt EEMEM = 20;     //Minutentakt
uint8_t ee_Titel[] EEMEM = {"Stadt Land V 0.0 pkr4/18"};

//RAM
int Startzahl;  //zufällige Startzahl für den Zufallsgenerator
volatile uint8_t Takt = 20;   //Anzahl der Takte pro Minute: 20
uint8_t Dauer;  //Spieldauer in Minuten mit Schalter eingestellt
volatile uint8_t Min;    //Minutenzähler im Interrupt
volatile uint8_t Stop = 0; //Merker für Taste
uint8_t Auswahl = 0;
uint8_t Vorhandene[] = {30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30,
                                30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30};
uint8_t Anzahl_Buchstaben = 0;  //Anzahl der schon angezeigten Buchstaben
```

```
75 char Temporaer[] = {"01234"};
76 char Ueberschrift[] = {"LIZZYS"};
77 char Ueberschrift1[] = {"Stadt Land"};
78 char text2[] = {"Batt."};
79 char text3[] = {"Min."};
80 char text4[] = {"Anzahl"};
81 uint8_t Anfangstext[] = {H, A, L, L, O, leer, L, I, Z, Z, Y};  //Für Morseausgabe
82 uint8_t Endetext[] = {H, A, L, T};  //Für Morseausgabe
83
84 uint16_t Morsetabelle[]   =    {0b1011100000000000, //A
85                                 0b1110101010000000, //B
86                                 0b1110101110100000, //C
87                                 0b1110101000000000, //D
88                                 0b1000000000000000, //E
89                                 0b1010111010000000, //F
90                                 0b1110111010000000, //G
91                                 0b1010101000000000, //H
92                                 0b1010000000000000, //I
93                                 0b1011101110111000, //J
94                                 0b1110101110000000, //K
95                                 0b1011101010000000, //L
96                                 0b1110111000000000, //M
97                                 0b1110100000000000, //N
98                                 0b1110111011100000, //O
99                                 0b1011101110100000, //P
100                                0b1110111010111000, //Q
101                                0b1011101000000000, //R
102                                0b1010100000000000, //S
103                                0b1110000000000000, //T
104                                0b1010111000000000, //U
105                                0b1010101110000000, //V
106                                0b1011101110000000, //W
107                                0b1110101011100000, //X
108                                0b1110101110111000, //Y
109                                0b1110111010100000  //Z
110                                };
111 //Ende RAM
112
113 uint8_t Tastenabfrage()
114 {
115     //wenn Taste losgelassen, dann Rückgabe 1, ansonsten 0
116     //Programm bleibt in der Schleife solange Taste gedrückt ist.
117     //Während der gedrückten Taste wird eine zufällige Startzahl erzeugt.
118
119     if(Start_Taste_an)
120     {
121         _delay_ms(100); //Tastenprellung
122         while (Start_Taste_an) {Startzahl++;}
123         _delay_ms(100); //Tastenprellung
124         if(!Start_Taste_an) return(1);
125     }
126     return(0);
127 }
128
129 uint8_t Minutenabfrage()
130 {
131     //Programm fragt die DIP Schaltereinstellung für die Minuten ab,
132     //und gibt den Wert zurück.
133     uint8_t Minuten = 1;
134
135     //Die Bitreihenfolge wird vertauscht und negiert wegen der besseren Einstellung am DIP Schalter
136
137     Minuten =  ~(((Vorwahl_Minuten & 0x20) >> 5) | ((Vorwahl_Minuten & 0x10) >> 3) |
    ((Vorwahl_Minuten & 0x08) >> 1)) & 0x07;
138
139    return(Minuten);
140 }
141
142
143
144 uint8_t Buchstabe_vorhanden(uint8_t Zahl)
145 {
146     uint8_t nz, iz;
147
148     iz = 0;
149
```

```
150     //überprüft, ob Buchstabe vorhanden ist
151     for (nz=0;nz<26;nz++)
152     {
153         if (Vorhandene[nz] == Zahl)
154         {
155             return(1);
156         }
157     }
158     //Anzahl der Buchstaben wird ermittelt
159     while (Vorhandene[iz] != 30)
160         {
161             iz++;
162         }
163 
164     Anzahl_Buchstaben = iz + 1;  //Anzahl wird in der globalen Variable gespeichert
165 
166     Vorhandene[iz] = Zahl;   //Neuer Buchstabe wird im Feld gespeichert
167 
168 return(0);
169 }
170 
171 void Morsezeichen(uint8_t Zeichen)
172 {
173     //Gibt das Zeichen als Morsecode aus, der in der Morsetabelle gespeichert ist
174     //Die eingestellte Zeit sind ms für ein Punkt (Zeichenlänge = 70)
175     uint8_t Zaehler;
176     uint16_t Code;
177     uint8_t Nullen = 0;
178 
179     Code = Morsetabelle[Zeichen];
180 
181     for(Zaehler=0; Zaehler<16; Zaehler++)
182     {
183         if(Code & 0x8000)
184         {
185             Summer_an;
186             _delay_ms(Zeichenlaenge);
187             Nullen = 0;
188         }
189         else
190         {
191             Summer_aus;
192             _delay_ms(Zeichenlaenge);
193             Nullen++;
194         }
195 
196         if(Nullen >=3) return;
197         Code <<= 1;
198     }
199 }
200 
201 void Morsetext(uint8_t Text[], uint8_t laenge)
202 {
203     uint8_t Zaehler;
204 
205     for(Zaehler=0;Zaehler<laenge;Zaehler++)
206     {
207         if(Text[Zaehler] == leer)
208         {
209             Summer_aus;
210             _delay_ms(7*Zeichenlaenge);
211         }
212         else
213         {
214             Morsezeichen(Text[Zaehler]);
215         }
216     }
217 }
218 
219 uint8_t Spielbeginn(uint8_t Minuten)
220 {
221     //Es wird eine Zufallszahl zwischen 0 und 25 für den Buchstaben ausgewählt
222     //und ist der Rückgabewert.
223     //Der Buchstabe wird geprüft, ob er schon mal vergeben wurde.
224     //Wenn der Buchstabe schon mal vokam, wird ein neuer "gewürfelt".
225 
```

```
226     //Vorher wird die Spieldauer ausgeschrieben.
227     //Der Signalton für den Spielbeginn wird eingeschaltet.
228
229     uint8_t zuf;
230
231     Stop = 1;   //Merker für Spielabbruch
232
233     Grundtexte();
234
235     //Anfangsminuten
236     utoa(Minuten, Temporaer, 10);
237     Schreibe_Text(Temporaer, sans_mono_24, 13, 45);
238
239     // Schreibe_Zahl(Anzahl_Buchstaben);
240     Schreibe_Zahl_zweistellig(Anzahl_Buchstaben, 20, 40);
241
242     srand(Startzahl);   //Start für den Zufallszahlengenerator
243     zuf = (uint8_t)rand()%26;
244     //Generiere solange neuen Buchstaben, bis der Buchstabe nicht vorhanden ist.
245      while (Buchstabe_vorhanden(zuf) == 1)
246      {
247         zuf = (uint8_t)rand()%26;
248      }
249
250    // Schreibe ausgewählten Buchstaben
251     Schreibe_Buchstabe(zuf, serif_72, 22, 170);
252     Batterieanzeige(ADC_Wert());
253
254     Display_Fram();
255
256     Tim_Int_ein;
257
258     // Signalausgabe Buchstabe
259      Morsezeichen(zuf);
260
261     return zuf;
262 }
263
264 void Spielende()
265 {
266     Tim_Int_aus;
267
268     Stop = 0;
269     Min = Dauer;    //Zähler für Timer wird zurückgestellt
270     Takt = 20;
271
272    // 0 Minuten ausgeben
273     Temporaer[0] = 0x30;
274     Temporaer[1] = 0;   //Stringende
275     Schreibe_Text(Temporaer, sans_mono_24, 13, 45);
276
277     //Buchstabenauswahl nochmal schreiben
278     Schreibe_Buchstabe(Auswahl, serif_72, 22, 170);
279
280     //Schreibe_Zahl(Anzahl_Buchstaben);
281     Schreibe_Zahl_zweistellig(Anzahl_Buchstaben, 20, 40);
282
283     Batterieanzeige(ADC_Wert());
284
285     Display_Fram();
286
287      //Endesignal Morseausgabe
288     Morsetext(Endetext, 4);
289 }
290
291 void Grundinitialisierung()
292 {
293
294     //Wenn der ganze Bildschirm beschrieben wird, eine Pause von 2s
295     //nach dem Schreiben einhalten.
296
297     Schreibe_Bildschirm_voll(weiss);  //0xFF
298     _delay_ms(2000);
299
300     Schreibe_Bildschirm_voll(weiss);  //weiss
301     _delay_ms(2000);
```

```
302 }
303
304 void Grundtexte()
305 {
306     //Hier stehen die Texte, die nicht verändert werden
307     Schreibe_Text(Ueberschrift, sans_mono_24, 5, 190);
308     Schreibe_Text(Ueberschrift1, sans_mono_12, 7, 180);
309     Schreibe_Text(text2, sans_mono_12, 8, 55);
310     Schreibe_Text(text3, sans_mono_12, 15, 55);
311     Schreibe_Text(text4, sans_mono_12, 22, 70);
312 }
313
314 void Schreibe_Zahl_zweistellig(uint8_t Zahl, uint8_t X_Wert, uint8_t Y_Wert)
315 {
316     //Zeige zweistellige Zahl an. Schrift: sans_mono_24
317     utoa(Zahl, Temporaer, 10);
318     if(Zahl < 10)
319     {
320         Schreibe_Text(Temporaer, sans_mono_24, X_Wert, Y_Wert);
321     }
322     else
323     {
324         Schreibe_Text(Temporaer, sans_mono_24, X_Wert, Y_Wert + 20);
325     }
326 }
327
328 void Anfangsbildschirm()
329 {
330
331     Grundtexte();
332
333     //Schreibe Spieldauer in Minuten
334     Dauer = Minutenabfrage();
335     Min = Dauer;
336     utoa(Dauer, Temporaer, 10);
337     Schreibe_Text(Temporaer, sans_mono_24, 13, 45);
338
339     //Schreibe Anzahl der Buchstaben
340     Schreibe_Zahl_zweistellig(Anzahl_Buchstaben, 20, 40);
341
342    // Schreibe "A" als Anfangsbuchstabe
343     Schreibe_Buchstabe(Auswahl, serif_72, 22, 170);
344
345     Batterieanzeige(ADC_Wert());
346
347     Display_Fram();
348     //Ausgabe Morsetext Anfang
349     Morsetext(Anfangstext, 11);
350
351 }
352
353 void Batterieanzeige(uint16_t ADC_ausgabe)
354 {
355     //Bei 2,66V beträgt der ADC_Wert 760
356     uint16_t Spannung;  //Batteriespannung * 100
357    //  utoa(ADC_ausgabe, Temporaer, 10);
358    // Schreibe_Text(Temporaer, sans_mono_12, 13, 190);
359     Spannung = (uint16_t) ((266 * (uint32_t)ADC_ausgabe) / 760);
360
361     if (Spannung > 280)
362     {
363           //Batteriestand Ausgabe 100%
364         Schreibe_Buchstabe(0, Batterie, 6, 40);
365     }
366     else if ((Spannung > 260) && (Spannung <= 280))
367     {
368         //Batteriestand Ausgabe 75%%
369         Schreibe_Buchstabe(1, Batterie, 6, 40);
370     }
371     else if ((Spannung > 240) && (Spannung <=260))
372     {
373         //Batteriestand Ausgabe 50%
374         Schreibe_Buchstabe(2, Batterie, 6, 40);
375     }
376     else if ((Spannung > 220) && (Spannung <= 240))
377     {
```

```
378         //Batteriestand Ausgabe 25%%
379         Schreibe_Buchstabe(3, Batterie, 6, 40);
380     }
381     else
382     {
383         //Batteriestand Ausgabe 0%
384         Schreibe_Buchstabe(4, Batterie, 6, 40);
385     }
386 
387 }
388 
389 uint16_t ADC_Wert()
390 {
391 //UP gibt den Digitalwert der Spannung am Kanal zurück
392     //760 = 2,66V
393 
394         ADMUX &= ~((1<<MUX3) | (1<<MUX2) | (1<<MUX1) | (1<<MUX0)); //Kanal 0
395 
396         ADCSRA |= (1<<ADSC);    //Starte Wandlung
397         while (!(ADCSRA & (1<<ADIF)));  //Schleife bis Wandlung fertig
398         ADCSRA &= ~(1<<ADIF);   //Lösche ADIF
399 
400         return ADC;     //Rückgabe des Digitalwertes
401 }
402 
403 //Timer1 Routine alle 3s
404 ISR(TIMER1_OVF_vect)
405 {
406 
407         TCNT1 = 0xFFFF -Sec_Faktor;     //Zähler mit Anfangswert laden für 3s
408 
409         Takt--; //Taktzeit ist der Sekundenzähler
410         if(!Takt)
411         {
412             Takt = 20;  //Taktzeit wiederherstellen
413 
414             Schreibe_Buchstabe(Auswahl, serif_72, 22, 170);
415 
416             //Anzahl der schon gespielten Buchstaben
417             Schreibe_Zahl_zweistellig(Anzahl_Buchstaben, 20, 40);
418 
419             Min--;
420             //Ausgabe der verbleibenden Minuten
421             utoa(Min, Temporaer, 10);
422             Schreibe_Text(Temporaer, sans_mono_24, 13, 45);
423 
424             Display_Fram();
425 
426             if (Min == 0)
427             {
428                 Min = Dauer;
429                 Spielende();
430             }
431         }
432 }
433 
434 void EEPROM_lesen()
435 {
436     Takt = eeprom_read_byte(&ee_Takt);
437 }
438 
439 void Ports_init()
440 {
441     //Port D für die Anzeige, Port C für die Tasten und Summer
442         DDRD &= ~(1<<E_Busy);   //Busy Eingang
443         PORTD |= (1<<E_Busy);   //Pullup Widerstand einschalten
444         DDRD |= (1<<E_Reset) | (1<<E_DC) | (1<<E_CS) | (1<<E_Clk) | (1<<E_DIN); //als Ausgang
445         DDRC &= ~((1<<Start_Taste) | (1<<Minuten0) | (1<<Minuten2) | (1<<Minuten4));  //Taste und
    Schalter Minuten als Eingang
446         PORTC |= ((1<<Start_Taste) | (1<<Minuten0) | (1<<Minuten2) | (1<<Minuten4));  //Pullup
    Widerstände einschalten
447         DDRC |= (1<<Summer);    //Suumer als Ausgang
448         DDRC &= ~(1<<Spannung_Kanal0);  //Analogeingang0 als Eingang
449         ADMUX |= (1<<REFS1);    //Referenzspannung 1,1V
450         ADCSRA |= (1<<ADEN);    //AD-Wandler freigeben
451         return;
```

```
452 }
453
454 void TMR1_init()
455 {
456         TCCR1B |= (1<<CS12);    //Vorteiler 256
457         TIMSK1 |= (1<<TOIE1);   //Interrupt bei Überlauf
458         TCNT1 = 0xFFFF -Sec_Faktor;     //Zähler mit Anfangswert laden für 1s
459 }
460
461 int main()
462 {
463
464     wdt_disable();
465     Ports_init();
466     EEPROM_lesen();
467     Reset();
468
469     Display_Init(lut_full_update);
470     Grundinitialisierung();
471
472     Reset();
473
474     Display_Init(lut_partial_update); //Ab jetzt nur noch teilweise Aktualisierung
475
476     Anfangsbildschirm();
477  //   _delay_ms(2000);
478
479     TMR1_init();
480     Tim_Int_aus;
481     sei();
482
483     while(1)
484       {
485
486         if(Tastenabfrage() == 1)
487         {
488             if(Stop == 0)
489             {
490                 Auswahl = Spielbeginn(Dauer);
491             }
492             else
493             {
494                 Spielende();
495             }
496
497         }
498      }
499
500     return (0);
501 }
502
503
504
```