```
1   //Lineární hodiny
2   //Marek Firla
3
4   // připojení potřebných knihoven
5   //#include <Arduino.h> //knihovny použité při ladění programu
6   //#include <Wire.h>
7   #include <DS3231.h> //knihovna zajišťující funkci modulu reálného času
8   #include <TM1637Display.h> //knihovna pro ovládání displeje
9   #include <avr/sleep.h> //knihovny pro uspání Arduina
10  #include <avr/wdt.h>
11
12  // nastavení čísel propojovacích pinů
13  #define SLP_HOD 0
14  #define SLP_MIN 1
15  #define DOWN_TIME 2
16  #define SET_TIME 3
17  #define UP_TIME 4
18  #define CLK 5
19  #define DIO 6
20  #define DIR_HOD 7
21  #define STEP_HOD 8
22  #define DIR_MIN 9
23  #define STEP_MIN 10
24  #define KON_HOD 11
25  #define KON_MIN 12
26  #define BUZZER 13
27
28  TM1637Display display(CLK, DIO); // vytvoření instance displej z knihovny
29  TM1637
30  DS3231 RTC; // inicializace RTC z knihovny
31
32  //deklarace proměnných
33  int Hodiny; // proměnná uchovávající aktuální Hodinu
34  int Minuty; // proměnná uchovávající aktuální minutu
35  bool h12; // logická proměnná pro nastavení 24 nebo 12 hodinového formátu
36  bool PM;  // logické proměnná pro rozlišení dopolední a odpoledního času
37  int HodinyPol; // proměnná uchovávající aktuální polohu hodinového
38  ciferníku vůči indikátoru
39  int MinutyPol; // proměnná uchovávající aktuální polohu hodinového
40  ciferníku vůči indikátoru
41  int HodinyAlarm; // proměnná uchovávající čas alarmu
42  int MinutyAlarm;
43  bool AlarmOvladani = false; //logická proměnná pro zapnutí nebo vypnutí
44  alarmu
45  bool AlarmStav = false; //logická proměnná indikující stav zvukové
46  signalizace
47
48  // proměnné sloužící pro ošetření zákmitů tlačítek
49  int enStiskSetTime = 1;      // proměnná povolení stisku
50  long stiskMilSetTime = 0;    // proměnná pro čas stisku
51  int stiskSetTime = LOW;      // proměnná pro kontrolu stisku
52  int stavTlacSetTime = 0;     // proměnná stavu tlačítka
53
54  int enStiskUpTime = 1;      // proměnná povolení stisku
55  long stiskMilUpTime = 0;    // proměnná pro čas stisku
56  int stiskUpTime = LOW;      // proměnná pro kontrolu stisku
57  int stavTlacUpTime = 0;     // proměnná stavu tlačítka
58
59  int enStiskDownTime = 1;      // proměnná povolení stisku
60  long stiskMilDownTime = 0;    // proměnná pro čas stisku
61  int stiskDownTime = LOW;      // proměnná pro kontrolu stisku
```

```
int stavTlacDownTime = 0;      // proměnná stavu tlačítka

int stavTlacKonHod = 0;        // proměnná stavu tlačítka
int stavTlacKonMin = 0;        // proměnná stavu tlačítka

int KrokyMin = 334;  //proměnná pro počet kroků motoru pro lineární posun o
rozestup jednotlivých označení minut na ciferníku
int KrokyHod = 1672; //proměnná pro počet kroků motoru pro lineární posun o
rozestup jednotlivých označení hodin na ciferníku

int Rychlost = 122; //přepočet na délku mezery mezi impulzy

int SetTimeMode = 0; //proměnná pro přepínání módů nastavení hodin

//hodnoty pro informativní nápisy na displeji
const uint8_t SEG_SET[] = {
  SEG_A | SEG_F | SEG_G | SEG_C | SEG_D ,           // S
  SEG_A | SEG_F | SEG_G | SEG_E | SEG_D ,           // E
  SEG_F | SEG_G | SEG_E | SEG_D ,                   // T
  0x00 ,
};

const uint8_t SEG_ON[] = {
  SEG_A | SEG_F | SEG_E | SEG_D | SEG_C | SEG_B ,   // O
  SEG_E | SEG_G | SEG_C | SEG_E ,                   // N
  0x00 ,
  0x00 ,
};

const uint8_t SEG_OFF[] = {
  SEG_A | SEG_F | SEG_E | SEG_D | SEG_C | SEG_B ,    // O
  SEG_E | SEG_F | SEG_A | SEG_G ,                   // F
  SEG_E | SEG_F | SEG_A | SEG_G ,                   // F
  0x00 ,
};

const uint8_t SEG_ALARM[] = {
  SEG_E | SEG_F | SEG_A | SEG_B | SEG_C | SEG_G ,   // A
  SEG_F | SEG_E | SEG_D ,                           // L
  SEG_E | SEG_F | SEG_A | SEG_B | SEG_C | SEG_G ,   // A
  0x00 ,
};

uint8_t PomVypis[] = { 0x00, 0x00, 0x00, 0x00 };

//Hodnoty pro formátování displeje
long BlikaniMil;
bool StavBlikani = true;
long OdpocetMil;
bool OdpocetStav = true;

// funkce pro vypnutí vnitřních hodin
ISR (WDT_vect)
{
  wdt_disable();
}

void setup() {
  // nastavení pinů jako výstupních
  pinMode(DIR_MIN, OUTPUT);
  pinMode(STEP_MIN, OUTPUT);
```

```
123     pinMode(DIR_HOD, OUTPUT);
124     pinMode(STEP_HOD, OUTPUT);
125     pinMode(SLP_HOD, OUTPUT);
126     pinMode(SLP_MIN, OUTPUT);
127     pinMode(BUZZER, OUTPUT);
128
129     // připojení obou řídících pinů na zem
130     digitalWrite(DIR_HOD, LOW);
131     digitalWrite(STEP_HOD, LOW);
132     digitalWrite(DIR_MIN, LOW);
133     digitalWrite(STEP_MIN, LOW);
134
135     digitalWrite(SLP_HOD, LOW);
136     digitalWrite(SLP_MIN, LOW);
137
138     // nastavení pinů jako vstupních
139     pinMode(SET_TIME, INPUT_PULLUP);
140     pinMode(UP_TIME, INPUT_PULLUP);
141     pinMode(DOWN_TIME, INPUT_PULLUP);
142     pinMode(KON_HOD, INPUT_PULLUP);
143     pinMode(KON_MIN, INPUT_PULLUP);
144
145     RTC.setClockMode(false); //nastavení 24 hodinového formátu času
146
147     //funkce pro hardwarové přerušení právě probíhajícího programu
148     //na základě stisknutí tlačítka SET
149     attachInterrupt(digitalPinToInterrupt(SET_TIME), Vzbudit, RISING);
150
151     //Nastavení výchozí polohy
152     HomeHod();
153     HomeMin();
154   }
155
156   void loop() {
157     Mode(); //kontrola stisku tlačítka SET
158
159     Debounce (); //funkce pro odstranění zákmitů u tlačítek
160
161     if (SetTimeMode == 0)
162     {
163       Synchronizace(); //funkce pro synchronizaci aktuálního času a polohy
164   ciferníků
165       Alarm(); //funkce pro ověření spuštění alarmu
166
167       if (Hodiny == HodinyPol && Minuty == MinutyPol && AlarmStav == false)
168       {
169         enStiskSetTime == 1;
170         Uspat(); //přechod do režimu s nízkým odběrem proudu
171       }
172     }
173
174     else
175     {
176       NastaveniCasu(); //funkce pro nastavení času a alarmu
177     }
178
179   }
180
181   //ošetření zákmitu tlačítek
182   //pokud je stisk tlačítka zakázaný tak ho povol po uplynutí 200 milisekund
183   void Debounce () {
```

```
    if (enStiskSetTime == 0) {
      if (millis() - stiskMilSetTime > 200) {
        enStiskSetTime = 1;
      }
    }

    if (enStiskUpTime == 0) {
      if (millis() - stiskMilUpTime > 200) {
        enStiskUpTime = 1;
      }
    }

    if (enStiskDownTime == 0) {
      if (millis() - stiskMilDownTime > 200) {
        enStiskDownTime = 1;
      }
    }
  }

  //stisk tlačítka SET
  //nastavení prostředního tlačítka
  // pokud je zvuková signalizace aktivní vypni ji, pokud ne nastav aktuální
  čas hodin
  void Mode() {
    stavTlacSetTime = digitalRead(SET_TIME);
    if (stavTlacSetTime == LOW & enStiskSetTime == 1)
    {
      if (AlarmStav == true)
      {
        AlarmStav = false;
        AlarmOvladani = false;
      }

      else
      {
        SetTimeMode = SetTimeMode + 1;
        OdpocetMil = millis();
      }

      stiskSetTime = HIGH;
    }
    if (stiskSetTime == HIGH) {
      stiskMilSetTime = millis();
      enStiskSetTime = 0;
      stiskSetTime = LOW;
    }
  }

  //funkce pro nastavení aktuálního času
  void NastaveniCasu() {
    //Načtení potřebných hodnot
    if (SetTimeMode == 1)
    {
      Hodiny = RTC.getHour(h12, PM);
      Minuty = RTC.getMinute();
      display.setBrightness(7, true);
      //display.setBrightness(0x0f);
      display.clear();
      display.setSegments(SEG_SET);
      delay(1000);
      SetTimeMode = 2;
```

```
245     }
246     //Nastavení hodin
247     if (SetTimeMode == 2)
248     {
249       Blikani();
250       UpHodiny();
251       DownHodiny();
252       Odpocet();
253     }
254     //Nastavení minut
255     if (SetTimeMode == 3)
256     {
257       Blikani();
258       UpMinuty();
259       DownMinuty();
260       Odpocet();
261     }
262     //Uložení hodnot do RTC modulu
263     if (SetTimeMode == 4)
264     {
265       RTC.setHour(Hodiny);
266       RTC.setMinute(Minuty);
267       RTC.setClockMode(false);
268       display.showNumberDecEx(RTC.getHour(h12, PM), 0b01000000, true, 2, 0);
269       display.showNumberDecEx(RTC.getMinute(), 0b01000000, true, 2, 2);
270       delay(1000);
271       display.clear();
272       display.setSegments(SEG_ALARM);
273       Hodiny = HodinyAlarm;
274       Minuty = MinutyAlarm;
275       delay(1000);
276       SetTimeMode = 5;
277     }
278     //Nastavení alarmu - hodiny
279     if (SetTimeMode == 5)
280     {
281       Blikani();
282       UpHodiny();
283       DownHodiny();
284       Odpocet();
285     }
286     //Nastavení alarmu - minuty
287     if (SetTimeMode == 6)
288     {
289       Blikani();
290       UpMinuty();
291       DownMinuty();
292       Odpocet();
293     }
294     //Nastavení alarmu - zapnutí
295     if (SetTimeMode == 7)
296     {
297       if (AlarmOvladani == false)
298       {
299         display.setSegments(SEG_OFF);
300       }
301       if (AlarmOvladani == true)
302       {
303         display.setSegments(SEG_ON);
304       }
305       ToggleOnOff();
```

```
306       Odpocet();
307     }
308     //Uložení nastavení alarmu
309     if (SetTimeMode == 8)
310     {
311       HodinyAlarm = Hodiny;
312       MinutyAlarm = Minuty;
313       display.showNumberDecEx(HodinyAlarm, 0b01000000, true, 2, 0);
314       display.showNumberDecEx(MinutyAlarm, 0b01000000, true, 2, 2);
315       delay(2000);
316       display.clear();
317       SetTimeMode = 0;
318     }
319   }
320
321   //zvyšuje nastavení hodinové hodnoty
322   void UpHodiny() {
323     stavTlacUpTime = digitalRead(UP_TIME);
324     if (stavTlacUpTime == LOW & enStiskUpTime == 1)
325     {
326       Hodiny = Hodiny + 1;
327       if (Hodiny > 23)
328       {
329         Hodiny = 0;
330       }
331       OdpocetMil = millis();
332       stiskUpTime = HIGH;
333     }
334     if (stiskUpTime == HIGH) {
335       stiskMilUpTime = millis();
336       enStiskUpTime = 0;
337       stiskUpTime = LOW;
338     }
339   }
340
341   //snižuje nastavení hodinové hodnoty
342   void DownHodiny() {
343     stavTlacDownTime = digitalRead(DOWN_TIME);
344     if (stavTlacDownTime == LOW & enStiskDownTime == 1)
345     {
346       Hodiny = Hodiny - 1;
347       if (Hodiny < 0)
348       {
349         Hodiny = 23;
350       }
351       OdpocetMil = millis();
352       stiskDownTime = HIGH;
353     }
354     if (stiskDownTime == HIGH) {
355       stiskMilDownTime = millis();
356       enStiskDownTime = 0;
357       stiskDownTime = LOW;
358     }
359   }
360
361   //zvyšuje nastavení minutové hodnoty
362   void UpMinuty() {
363     stavTlacUpTime = digitalRead(UP_TIME);
364     if (stavTlacUpTime == LOW & enStiskUpTime == 1)
365     {
366       Minuty = Minuty + 1;
```

```
      if (Minuty > 59)
      {
        Minuty = 0;
      }
      OdpocetMil = millis();
      stiskUpTime = HIGH;
    }
    if (stiskUpTime == HIGH) {
      stiskMilUpTime = millis();
      enStiskUpTime = 0;
      stiskUpTime = LOW;
    }
  }

  //snižuje nastavení minutové hodnoty
  void DownMinuty() {
    stavTlacDownTime = digitalRead(DOWN_TIME);
    if (stavTlacDownTime == LOW & enStiskDownTime == 1)
    {
      Minuty = Minuty - 1;
      if (Minuty < 0)
      {
        Minuty = 59;
      }
      OdpocetMil = millis();
      stiskDownTime = HIGH;
    }
    if (stiskDownTime == HIGH) {
      stiskMilDownTime = millis();
      enStiskDownTime = 0;
      stiskDownTime = LOW;
    }
  }

  //přepíná mezi zapnutím a vypnutím alarmu
  void ToggleOnOff() {
    stavTlacDownTime = digitalRead(DOWN_TIME);
    if (stavTlacDownTime == LOW & enStiskDownTime == 1)
    {
      display.clear();
      AlarmOvladani = !AlarmOvladani;
      OdpocetMil = millis();
      stiskDownTime = HIGH;
    }
    if (stiskDownTime == HIGH) {
      stiskMilDownTime = millis();
      enStiskDownTime = 0;
      stiskDownTime = LOW;
    }

    stavTlacUpTime = digitalRead(UP_TIME);
    if (stavTlacUpTime == LOW & enStiskUpTime == 1)
    {
      display.clear();
      AlarmOvladani = !AlarmOvladani;
      OdpocetMil = millis();
      stiskUpTime = HIGH;
    }
    if (stiskUpTime == HIGH) {
      stiskMilUpTime = millis();
      enStiskUpTime = 0;
```

```
      stiskUpTime = LOW;
    }
  }

  //nastavení výchozí polohy hodinového ciferníku
  void HomeHod() {
    digitalWrite(DIR_HOD, LOW);
    digitalWrite(SLP_HOD, HIGH);

    while (1)
    {
      stavTlacKonHod = digitalRead(KON_HOD);

      if (stavTlacKonHod == LOW)
      {
        HodinyPol = 0;
        digitalWrite(SLP_HOD, LOW);
        break;
      }
      digitalWrite(STEP_HOD, HIGH);
      delayMicroseconds(Rychlost);
      digitalWrite(STEP_HOD, LOW);
      delayMicroseconds(Rychlost);
    }
  }

  //nastavení výchozí polohy minutového ciferníku
  void HomeMin() {
    digitalWrite(DIR_MIN, HIGH);
    digitalWrite(SLP_MIN, HIGH);

    while (1)
    {
      stavTlacKonMin = digitalRead(KON_MIN);

      if (stavTlacKonMin == LOW)
      {
        MinutyPol = 0;
        digitalWrite(SLP_MIN, LOW);
        break;
      }

      digitalWrite(STEP_MIN, HIGH);
      delayMicroseconds(Rychlost);
      digitalWrite(STEP_MIN, LOW);
      delayMicroseconds(Rychlost);
    }
  }

  //zajišťuje synchronizaci aktuálního a zobrazovaného času
  void Synchronizace() {
    Hodiny = RTC.getHour(h12, PM);
    Minuty = RTC.getMinute();

    if (Hodiny > 12)
    {
      Hodiny = RTC.getHour(h12, PM) - 12;
    }

    if (Hodiny > HodinyPol)
    {
```

```
      digitalWrite(DIR_HOD, HIGH);
      digitalWrite(SLP_HOD, HIGH);
      delay(1);
      while (Hodiny > HodinyPol)
      {
        HodPosun();
      }
      digitalWrite(SLP_HOD, LOW);
    }

    if (Minuty > MinutyPol)
    {
      digitalWrite(DIR_MIN, LOW);
      digitalWrite(SLP_MIN, HIGH);
      delay(1);
      while (Minuty > MinutyPol)
      {
        MinPosun();
      }
      digitalWrite(SLP_MIN, LOW);
    }

    if (Hodiny < HodinyPol)
    {
      HomeHod();
    }

    if (Minuty < MinutyPol)
    {
      HomeMin();
    }
  }

  //posun o 1 minutu
  void MinPosun() {
    for (int i = 0; i <= KrokyMin; i++)
    {
      digitalWrite(STEP_MIN, HIGH);
      delayMicroseconds(Rychlost);
      digitalWrite(STEP_MIN, LOW);
      delayMicroseconds(Rychlost);
    }
    MinutyPol = MinutyPol + 1;
  }

  //posun o 1 hodinu
  void HodPosun() {
    for (int i = 0; i <= KrokyHod; i++)
    {
      digitalWrite(STEP_HOD, HIGH);
      delayMicroseconds(Rychlost);
      digitalWrite(STEP_HOD, LOW);
      delayMicroseconds(Rychlost);
    }
    HodinyPol = HodinyPol + 1;
  }

  //Mód šetření energií
  void Uspat() {
    digitalWrite(DIR_HOD, LOW);
    digitalWrite(STEP_HOD, LOW);
```

```
  digitalWrite(DIR_MIN, LOW);
  digitalWrite(STEP_MIN, LOW);
  digitalWrite(SLP_HOD, LOW);
  digitalWrite(SLP_MIN, LOW);
  display.clear();
  // vypne ADC
  ADCSRA = 0;
  // clear various "reset" flags
  MCUSR = 0;
  // allow changes, disable reset
  WDTCSR = bit (WDCE) | bit (WDE);
  // Nastavení přerušení a časový interval
  WDTCSR = bit (WDIE) | bit (WDP3) | bit (WDP0);    // nastav WDIE, a 8
sekundový delay
  wdt_reset();
  set_sleep_mode (SLEEP_MODE_PWR_DOWN);
  noInterrupts ();
  sleep_enable();
  attachInterrupt (0, Vzbudit, FALLING);
  EIFR = bit (INTF0);
  MCUCR = bit (BODS) | bit (BODSE);
  MCUCR = bit (BODS);
  interrupts ();
  sleep_cpu ();

  sleep_disable();
}

// vypnout mód pro šetření energií
void Vzbudit()
{
  sleep_disable();
  detachInterrupt (0);
}

//podprogram zajišťující problikávání právně nastavované hodnoty na
displeji
void Blikani() {
  if (StavBlikani)
  {
    BlikaniMil = millis();
    StavBlikani = false;
  }

  if ((millis() - BlikaniMil) < 500)
  {
    if (SetTimeMode == 2 || SetTimeMode == 5)
    {
      PomVypis[0] = 0x00;
      PomVypis[1] = 0x00;
      PomVypis[2] = display.encodeDigit(Minuty / 10);
      PomVypis[3] = display.encodeDigit(Minuty % 10);
      display.setSegments(PomVypis);
    }
    if (SetTimeMode == 3 || SetTimeMode == 6)
    {
      PomVypis[0] = display.encodeDigit(Hodiny / 10);
      PomVypis[1] = display.encodeDigit(Hodiny % 10);
      PomVypis[2] = 0x00;
      PomVypis[3] = 0x00;
      display.setSegments(PomVypis);
```

```
      }
    }
    if ((millis() - BlikaniMil) > 500)
    {
      display.showNumberDecEx(Hodiny, 0b01000000, true, 2, 0);
      display.showNumberDecEx(Minuty, 0b01000000, true, 2, 2);

      if ((millis() - BlikaniMil) > 1000)
      {
        StavBlikani = true;
      }
    }
  }

  //vypne režim pro nastavení aktuálního času při absenci vstupu, která trvá
  déle než jednu minutu
  void Odpocet() {
    if ((millis() - OdpocetMil) > 60000)
    {
      if (SetTimeMode < 4)
      {
        RTC.setHour(Hodiny);
        RTC.setMinute(Minuty);
        RTC.setClockMode(false);
        Hodiny = RTC.getHour(h12, PM);
        Minuty = RTC.getMinute();
        display.showNumberDecEx(Hodiny, 0b01000000, true, 2, 0);
        display.showNumberDecEx(Minuty, 0b01000000, true, 2, 2);
        delay(1000);
        display.clear();
        SetTimeMode = 0;
      }
      else
      {
        SetTimeMode = 8;
      }
    }
  }

  //funkce alarmu
  //pokud je alarm aktivní srovnej aktuální a nastavený čas pro alarm, pokud
  jsou shodné zapni zvukovou signalizaci
  void Alarm() {
    if (RTC.getHour(h12, PM) == HodinyAlarm && RTC.getMinute() == MinutyAlarm
  && AlarmOvladani == true)
    {
      AlarmStav = true;
    }

    if (AlarmStav == true)
    {
      for (int i = 0; i <= 3; i++)
      {
        tone(BUZZER, 500);
        delay(50);
        noTone(BUZZER);
        delay(50);
      }
      delay(1000);
    }
  }
```