Nastavni predmet:	Ugradbeni računalni sustavi
Vježba: 10	RTC – sat realnog vremena
Cilj vježbe:	Savladati uporabu RTC modula, sata realnog vremena radi izvođenja različitih projekata s vremenskom oznakom

Upute

Sve zadatke spremi na USB, a u bilježnici za sve zadatke napiši:

· postupak izrade programa
· objašnjenje korištenih naredbi
· dobivene rezultate po točkama
· odgovoriti u bilježnicu na postavljena pitanja vezana uz ovu vježbu
· Ukoliko u kòdu postoji greška, korigiraj i objasni!

Mikroupravljači najčešće nemaju ugrađen **sat realnog vremena RTC (Real Time Clock)**. Ukoliko imamo potrebu raditi projekte u kojima je važno stvarno (realno) vrijeme i/ili datum, moramo dodati RTC modul. Slika 1.
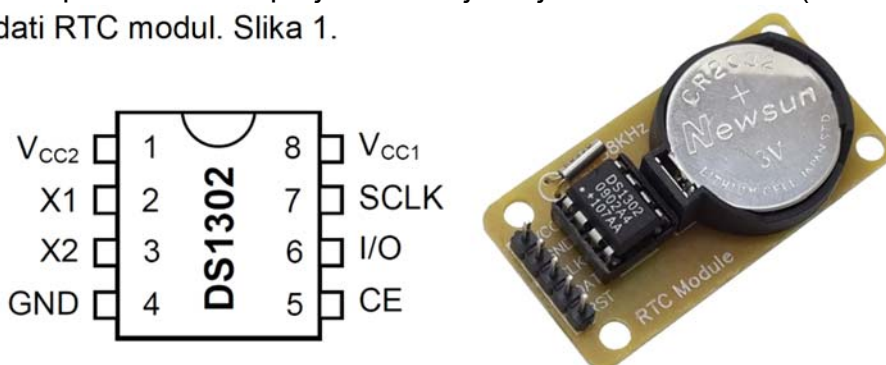
Slika 1. RTC modul s DS1302 integriranim krugom

RTC modul sadrži na sebi čip koji ima ugrađena brojila i nekoliko registara u kojima se pohranjuju vrijeme i datum. Osim RTC čipa, na pločici se nalazi kristal kvarca frekvencije 32768 Hz koji služi za davanje točnog signala takta, te baterija koja služi za rad brojila i čuvanje sadržaja registara kad uređaj nije priključen na napajanje (slika 2). Jednom kad se na RTC modulu podesi vrijeme i datum, on čuva vrijeme sve dok traje baterija, što je obično između 5 i 10 godina.
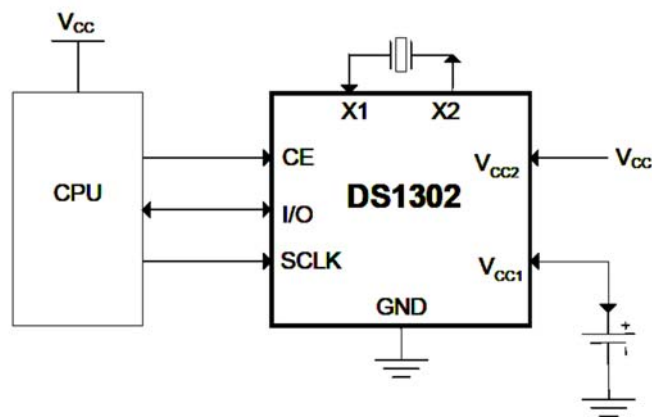
Slika 2. Tipična konfiguracija DS1302 RTC integriranog kruga [1]

Neki RTC integrirani krugovi, kao što je DS1302 ili DS1307 imaju i određenu količinu statičkog RAM-a, tzv. Non-Volatile RAM (NV-RAM). DS1302 ima 31 bajt NV-RAM-a, a DS1307 56 bajta NV-RAM-a.

Komunikacija DS1302 s mikroupravljačem ostvaruje se preko jednostavne vlastite sinkrone serijske komunikacije za koju je potrebno samo tri vodiča: CE (Chip Enable), I/O (data line) i SCLK (serial clock). Serijska vezna nije niti jedna od standardnih, nego se za komunikaciju brine biblioteka koju ćemo uključiti u program.

Podaci se s modula i na njega mogu prenositi bajt po bajt ili u nizu od 31 bajta od jednom. DS1302 dizajniran je tako da prilikom rada troši vrlo malo energije, tipično manje od 1µW [2]. Ima dvostruko napajanje, primarno i sekundarno kod nestanka primarnog napajanja.

U tablici 1, dane su adrese registara kao i sadržaj pojedinih registara.

RTC

READ	WRITE	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	RANGE
81h	80h	CH	10 Seconds			Seconds				00–59
83h	82h		10 Minutes			Minutes				00–59
85h	84h	$12/\overline{24}$	0	$\frac{10}{\overline{AM/PM}}$	Hour	Hour				1–12/0–23
87h	86h	0	0	10 Date		Date				1–31
89h	88h	0	0	0	10 Month	Month				1–12
8Bh	8Ah	0	0	0	0	0	Day			1–7
8Dh	8Ch	10 Year				Year				00–99
8Fh	8Eh	WP	0	0	0	0	0	0	0	—
91h	90h	TCS	TCS	TCS	TCS	DS	DS	RS	RS	—

Tablica 1. Adrese i sadržaj registara RTC integriranog kruga DS3102

Kako bi se pojednostavnio proces čitanja i pisanja na RTC modul, koristit ćemo biblioteku DS1302.h u kojoj se nalaze funkcije za komunikaciju s RTC modulom, funkcije za čitanje i podešavanje vremena i datuma, te funkcije za čitanje i pisanje u statički RAM.

Cjelokupni kod datoteka **DS1302.h i DS1302.cpp** u kojima se mogu pronaći funkcije za rad s RTC modulom, dani su u prilogu na kraju ovih upute za LV.

Osim DS1302 RTC modula koji ćemo koristiti na LV, često se koristi Tiny-RTC modul s DS1307 i I2C komunikacijom koji na sebi ima još i 32 KB EEPROM (slika 3). Uz njega, u upotrebi je i nešto skuplji i točniji RTC modul s ugrađena dva alarma i INT linijom za aktivaciju prekida. Također koristi I2C komunikaciju (slika 3).
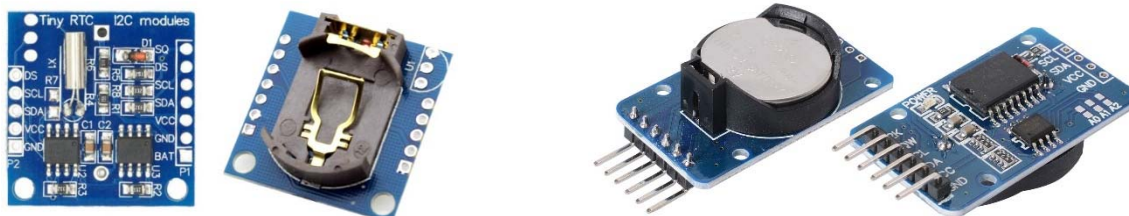

Slika 3. Tiny-RTC DS1307 modul i DS3231 RTC modul

Prije pokretanja programa potrebno je u Arduino IDE dodati DS1302.h biblioteku.

Zadatak 1. Spoji RTC modul prema shemi i podesi trenutno vrijeme. Podešeno vrijeme prati na Serial monitoru. Nakon učitavanja programa na Arduino, u slučaju problema pri ispisu vremena na Serial monitor, odspojite VCC vodič s RTC modula.

1) Nakon što podesiš vrijeme, potrebno je dio koda koji upisuje novo vrijeme i datum u RTC modul staviti u komentar pomoću //.
2) U prilogu na kraju LV potraži slijedeće funkcije: rtc.setDOW(WEDNESDAY); rtc.setTime(12, 55, 0; rtc.setDate(13, 2, 2019); rtc.getDOWStr(); rtc.getDateStr(); rtc.getTimeStr();

Električna shema
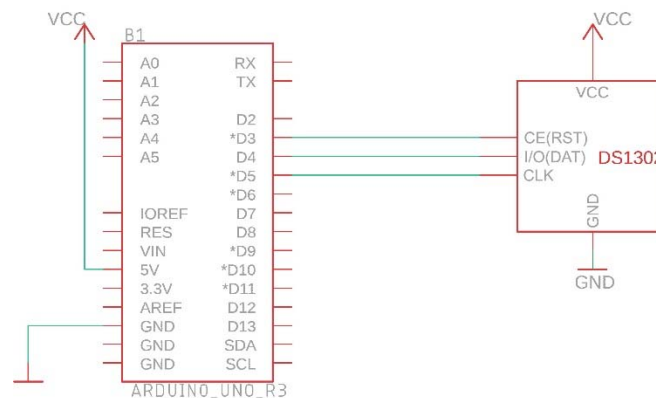
Kòd zadatka

```
// DS1302_Serial_Easy (C)2010 Henning Karlsen
// web: http://www.henningkarlsen.com/electronics
//
// A quick demo of how to use my DS1302-library to
// quickly send time and date information over a serial link
//
// I assume you know how to connect the DS1302.
// DS1302:  CE pin    -> Arduino Digital 2
//          I/O pin   -> Arduino Digital 3
//          SCLK pin  -> Arduino Digital 4

#include <DS1302.h>

// Init the DS1302
//   rtc (CE,DAT,CLK)
DS1302 rtc(2, 3, 4);

void setup()
{
    // Set the clock to run-mode, and disable the write protection
    rtc.halt(false);
    rtc.writeProtect(false);

    Serial.begin(9600);

    // Slijedeće tri linije treba staviti u komentar pomoću „//"
    // ako ne želiš više mijenjati podešeno vrijeme i datum
    rtc.setDOW(WEDNESDAY);       // Set Day-of-Week to WEDNESDAY
    rtc.setTime(12, 55, 0);      // Set the time to 12:55:00 (24hr format)
    rtc.setDate(13, 2, 2019);    // Set the date to February 13th, 2019
}
```

```
void loop()
{
    // Send Day-of-Week
    Serial.print(rtc.getDOWStr());
    Serial.print(" ");

    // Send date
    Serial.print(rtc.getDateStr());
    Serial.print(" -- ");

    // Send time
    Serial.println(rtc.getTimeStr());

    // Wait one second before repeating :)
    delay(1000);
}
```

Zadatak 2. Bez promjene spoja, doradi prethodni programski kod tako da Arduino pojedinačno ispisuje datum, vrijeme i dan u tjednu.

Kòd zadatka

```
#include <DS1302.h>

// Init the DS1302
//   rtc (CE,DAT,CLK)
DS1302 rtc(2, 3, 4);

// Init a Time-data structure
Time t;

void setup()
{
    // Set the clock to run-mode, and disable the write protection
    rtc.halt(false);
    rtc.writeProtect(false);

    // Setup Serial connection
    Serial.begin(9600);

}

void loop()
{
    // Get data from the DS1302
    t = rtc.getTime();

    // Send date over serial connection
    Serial.print("Danas je ");
    Serial.print(t.date, DEC);
    Serial.print(". ");
```

```
    Serial.print(rtc.getMonthStr());
    Serial.print(" ");
    Serial.print(t.year, DEC);
    Serial.println(". godine.");

    // Send Day-of-Week and time
    Serial.print("To je ");
    Serial.print(t.dow, DEC);
    Serial.println(". dan u tjednu. ");
    Serial.print("Od ponoci je proslo "); // (ponedjeljak je 1. dan)
    Serial.print(t.hour, DEC);
    Serial.print(" sati, ");
    Serial.print(t.min, DEC);
    Serial.print(" min i ");
    Serial.print(t.sec, DEC);
    Serial.println(" sekundi.");

    // Send a divider for readability
    Serial.println("--------------------------------------------------");

    // Wait one second before repeating :)
    delay(1000);
}
```
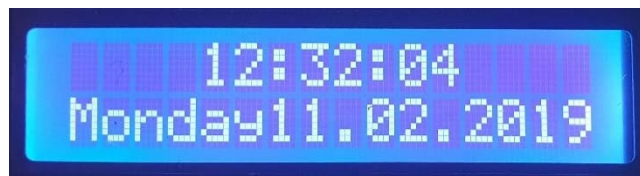
Zadatak 3. Doradi programski kod tako da na LCD zaslonu ispisuješ trenutno vrijeme, dan u tjednu i datum, prema priloženoj slici.

Zadatak 4. Prouči slijedeći programski kod koji čita i pohranjuje podatke u NV-RAM RTC modula. Zadatak ne treba rješavati, nego proučiti na koji način zapisati neki podatak u određenu memorijsku lokaciju, te kako pročitati podatak s određene adrese.

```
// Izvor koda isti kao u zadatku 1

#include <DS1302.h>

DS1302_RAM ramBuffer;
DS1302 rtc(2, 3, 4);

void setup()
{
    Serial.begin(9600);
}

void bufferDump(char st[])
{
    Serial.write(st);
```

```
        Serial.println("");
        for (int i = 0; i<31; i++)
        {
                Serial.print("0x");
                Serial.print(ramBuffer.cell[i], HEX);
                Serial.print(" ");
        }
        Serial.println("");
        Serial.println("-------------------------------------------------");
}

void comment(char st[])
{
        Serial.println("");
        Serial.print("---> ");
        Serial.write(st);
        Serial.println("");
        Serial.println("");
}

void loop()
{
        Serial.println("");
        bufferDump("Initial buffer");

        comment("Filling buffer with data...");
        for (int i = 0; i<31; i++)
                ramBuffer.cell[i] = i;

        comment("Writing buffer to RAM...");
        rtc.writeBuffer(ramBuffer);
        bufferDump("Buffer written to RAM...");

        comment("Clearing buffer...");
        for (int i = 0; i<31; i++)
                ramBuffer.cell[i] = 0;
        bufferDump("Cleared buffer...");

        comment("Setting byte 15 (0x0F) to value 160 (0xA0)...");
        rtc.poke(15, 160);

        comment("Reading buffer from RAM...");
        ramBuffer = rtc.readBuffer();
        bufferDump("Buffer read from RAM...");

        int temp;
        comment("Reading address 18 (0x12). This should return 18, 0x12.");
        temp = rtc.peek(18);
        Serial.print("Return value: ");
        Serial.print(temp, DEC);
        Serial.print(", 0x");
        Serial.println(temp, HEX);
```

```
    Serial.println("");
    Serial.println("");
    Serial.println("***** End of demo *****");

    while (1) {};
}
```

Zadatak 5. Doradi prethodni spoj dodavanjem tri tipkala (ili jednog tipkala i inkrementalnog enkodera) i jedne LED diode. Potrebno je ugraditi funkciju alarma.

- a) Pomoću tipke SW_MENU mijenjati između četiri mogućnosti: podešavanje znamenki hh → mm → ss → normalna operacija prikaza vremena i praćenja alarma → hh →...
- b) Tipke SW_UP i SW_DOWN koristiti za povećavanje i smanjivanje odabrane znamenke: hh = [0 – 24], mm = [0 – 59], ss = [0 – 59].
- c) Odabranu znamenku pohraniti u NV-RAM (lokaciju odabrati po želji)
- d) Kod nailaska sata na podešeno vrijeme alarma, potrebno je uključiti LED diodu
- e) Pritiskom obje tipke SW_UP i SW_DOWN istovremeno, LED dioda se isključuje.
- f) Prilikom isključenja napajanja i ponovnog uključenja, uređaj treba učitati podešeno vrijeme alarma i nastaviti kao da nije bilo prekida

Električna shema
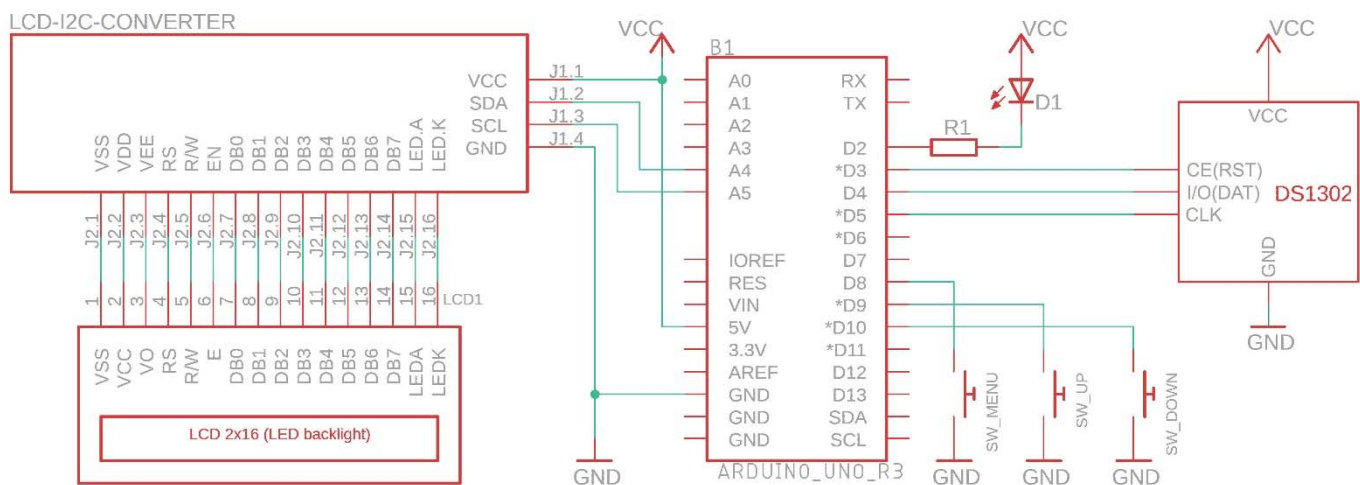
Zadatak 6. Doradi prethodni program tako da omogućiš uključivanje grijanja kroz dva intervala u danu. Npr. jedan interval od 06:30:00 do 07:00:00, a drugi interval od 15:30:00 do 22:00:00. Za potrebe vježbe, intervale podešavati u terminima ss i/ili mm.

Ulaskom u programski mod, prvi interval prikazati u gornjem redu, a drugi u donjem retku. Redoslijed podešavanja izvesti kako slijedi: podešavanje znamenki hh_1 → mm_1 → ss_1 → hh_2 → mm_2 → ss_2→ normalna operacija prikaza vremena i praćenja alarma → hh_1 → mm_1 →… U tom modu, stvarno vrijeme prikazati u sredini prvog retka zaslona, u drugom retku prikazati početak prvog nadolazećeg intervala za uključivanje grijanja. Unutar podešenih intervala, potrebno je uključiti LED diodu.

LITERATURA:

1. Maxim Integrated Products, DS1302 Trickle-Charge Timekeeping Chip,
   https://datasheets.maximintegrated.com/en/ds/DS1302.pdf (pregledano 11.02.2019.)

2. Rinky-Dink Electronics., Library: DS1302,
   http://www.rinkydinkelectronics.com/library.php?id=5 (pregledano 11.02.2019.)

PRILOG1: DS1302.h

```c
/*
DS1302.h - Arduino library support for the DS1302 Trickle Charge Timekeeping Chip
Copyright (C)2010 Henning Karlsen. All right reserved

You can find the latest version of the library at
http://www.henningkarlsen.com/electronics

This library has been made to easily interface and use the DS1302 RTC with
the Arduino.

If you make any modifications or improvements to the code, I would appreciate
that you share the code with me so that I might include it in the next release.
I can be contacted through http://www.henningkarlsen.com/electronics/contact.php

This library is free software; you can redistribute it and/or
modify it under the terms of the GNU Lesser General Public
License as published by the Free Software Foundation; either
version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public
License along with this library; if not, write to the Free Software
Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA
*/
#ifndef DS1302_h
#define DS1302_h

#if defined(ARDUINO) && ARDUINO >= 100
#include "Arduino.h"
#else
#include "WProgram.h"
#endif

#define FORMAT_SHORT 1
#define FORMAT_LONG  2

#define FORMAT_LITTLEENDIAN     1
#define FORMAT_BIGENDIAN        2
#define FORMAT_MIDDLEENDIAN     3

#define MONDAY          1
#define TUESDAY         2
#define WEDNESDAY       3
#define THURSDAY        4
#define FRIDAY          5
#define SATURDAY        6
#define SUNDAY          7

#define TCR_D1R2K   165
#define TCR_D1R4K   166
#define TCR_D1R8K   167
#define TCR_D2R2K   169
#define TCR_D2R4K   170
#define TCR_D2R8K   171
#define TCR_OFF     92

class Time
{
public:
        uint8_t                 hour;
```

```cpp
        uint8_t             min;
        uint8_t             sec;
        uint8_t             date;
        uint8_t             mon;
        uint16_t            year;
        uint8_t             dow;

        Time();
};

class DS1302_RAM
{
public:
        byte    cell[31];

        DS1302_RAM();
};

class DS1302
{
public:
        DS1302(uint8_t ce_pin, uint8_t data_pin, uint8_t sclk_pin);
        Time    getTime();
        void    setTime(uint8_t hour, uint8_t min, uint8_t sec);
        void    setDate(uint8_t date, uint8_t mon, uint16_t year);
        void    setDOW(uint8_t dow);

        char    *getTimeStr(uint8_t format = FORMAT_LONG);
        char    *getDateStr(uint8_t slformat = FORMAT_LONG, uint8_t eformat = FORMAT_LITTLEENDIAN,
char divider = '.');
        char    *getDOWStr(uint8_t format = FORMAT_LONG);
        char    *getMonthStr(uint8_t format = FORMAT_LONG);

        void    halt(bool value);
        void    writeProtect(bool enable);
        void    setTCR(uint8_t value);

        void            writeBuffer(DS1302_RAM r);
        DS1302_RAM      readBuffer();
        void            poke(uint8_t addr, uint8_t value);
        uint8_t             peek(uint8_t addr);

private:
        uint8_t _ce_pin;
        uint8_t _data_pin;
        uint8_t _sclk_pin;
        uint8_t _burstArray[8];

        uint8_t     _readByte();
        void    _writeByte(uint8_t value);
        uint8_t      _readRegister(uint8_t reg);
        void    _writeRegister(uint8_t reg, uint8_t value);
        void    _burstRead();
        uint8_t     _decode(uint8_t value);
        uint8_t     _decodeH(uint8_t value);
        uint8_t     _decodeY(uint8_t value);
        uint8_t     _encode(uint8_t vaule);
};
#endif
```

PRILOG2: DS1302.cpp

```cpp
/*
DS1302.cpp - Arduino library support for the DS1302 Trickle Charge Timekeeping Chip
Copyright (C)2010 Henning Karlsen. All right reserved

You can find the latest version of the library at
http://www.henningkarlsen.com/electronics

This library has been made to easily interface and use the DS1302 RTC with
the Arduino.

If you make any modifications or improvements to the code, I would appreciate
that you share the code with me so that I might include it in the next release.
I can be contacted through http://www.henningkarlsen.com/electronics/contact.php

This library is free software; you can redistribute it and/or
modify it under the terms of the GNU Lesser General Public
License as published by the Free Software Foundation; either
version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public
License along with this library; if not, write to the Free Software
Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA
*/
#include "DS1302.h"

#define REG_SEC          0
#define REG_MIN          1
#define REG_HOUR         2
#define REG_DATE         3
#define REG_MON          4
#define REG_DOW          5
#define REG_YEAR         6
#define REG_WP           7
#define REG_TCR          8

/* Public */

Time::Time()
{
      this->year = 2010;
      this->mon = 1;
      this->date = 1;
      this->hour = 0;
      this->min = 0;
      this->sec = 0;
      this->dow = 5;
}

DS1302_RAM::DS1302_RAM()
{
      for (int i = 0; i<31; i++)
            cell[i] = 0;
}

DS1302::DS1302(uint8_t ce_pin, uint8_t data_pin, uint8_t sclk_pin)
{
      _ce_pin = ce_pin;
      _data_pin = data_pin;
      _sclk_pin = sclk_pin;
```

```cpp
        pinMode(_ce_pin, OUTPUT);
        pinMode(_sclk_pin, OUTPUT);
}

Time DS1302::getTime()
{
        Time t;
        _burstRead();
        t.sec = _decode(_burstArray[0]);
        t.min = _decode(_burstArray[1]);
        t.hour = _decodeH(_burstArray[2]);
        t.date = _decode(_burstArray[3]);
        t.mon = _decode(_burstArray[4]);
        t.dow = _burstArray[5];
        t.year = _decodeY(_burstArray[6]) + 2000;
        return t;
}

void DS1302::setTime(uint8_t hour, uint8_t min, uint8_t sec)
{
        if (((hour >= 0) && (hour<24)) && ((min >= 0) && (min<60)) && ((sec >= 0) && (sec<60)))
        {
                _writeRegister(REG_HOUR, _encode(hour));
                _writeRegister(REG_MIN, _encode(min));
                _writeRegister(REG_SEC, _encode(sec));
        }
}

void DS1302::setDate(uint8_t date, uint8_t mon, uint16_t year)
{
        if (((date>0) && (date <= 31)) && ((mon>0) && (mon <= 12)) && ((year >= 2000) &&
(year<3000)))
        {
                year -= 2000;
                _writeRegister(REG_YEAR, _encode(year));
                _writeRegister(REG_MON, _encode(mon));
                _writeRegister(REG_DATE, _encode(date));
        }
}

void DS1302::setDOW(uint8_t dow)
{
        if ((dow>0) && (dow<8))
                _writeRegister(REG_DOW, dow);
}

char *DS1302::getTimeStr(uint8_t format)
{
        char *output = "xxxxxxxx";
        Time t;
        t = getTime();
        if (t.hour<10)
                output[0] = 48;
        else
                output[0] = char((t.hour / 10) + 48);
        output[1] = char((t.hour % 10) + 48);
        output[2] = 58;
        if (t.min<10)
                output[3] = 48;
        else
                output[3] = char((t.min / 10) + 48);
        output[4] = char((t.min % 10) + 48);
        output[5] = 58;
        if (format == FORMAT_SHORT)
                output[5] = 0;
```

```cpp
        else
        {
                if (t.sec<10)
                        output[6] = 48;
                else
                        output[6] = char((t.sec / 10) + 48);
                output[7] = char((t.sec % 10) + 48);
                output[8] = 0;
        }
        return output;
}

char *DS1302::getDateStr(uint8_t slformat, uint8_t eformat, char divider)
{
        char *output = "xxxxxxxxxx";
        int yr, offset;
        Time t;
        t = getTime();
        switch (eformat)
        {
        case FORMAT_LITTLEENDIAN:
                if (t.date<10)
                        output[0] = 48;
                else
                        output[0] = char((t.date / 10) + 48);
                output[1] = char((t.date % 10) + 48);
                output[2] = divider;
                if (t.mon<10)
                        output[3] = 48;
                else
                        output[3] = char((t.mon / 10) + 48);
                output[4] = char((t.mon % 10) + 48);
                output[5] = divider;
                if (slformat == FORMAT_SHORT)
                {
                        yr = t.year - 2000;
                        if (yr<10)
                                output[6] = 48;
                        else
                                output[6] = char((yr / 10) + 48);
                        output[7] = char((yr % 10) + 48);
                        output[8] = 0;
                }
                else
                {
                        yr = t.year;
                        output[6] = char((yr / 1000) + 48);
                        output[7] = char(((yr % 1000) / 100) + 48);
                        output[8] = char(((yr % 100) / 10) + 48);
                        output[9] = char((yr % 10) + 48);
                        output[10] = 0;
                }
                break;
        case FORMAT_BIGENDIAN:
                if (slformat == FORMAT_SHORT)
                        offset = 0;
                else
                        offset = 2;
                if (slformat == FORMAT_SHORT)
                {
                        yr = t.year - 2000;
                        if (yr<10)
                                output[0] = 48;
                        else
                                output[0] = char((yr / 10) + 48);
                        output[1] = char((yr % 10) + 48);
```

```cpp
                output[2] = divider;
            }
            else
            {
                yr = t.year;
                output[0] = char((yr / 1000) + 48);
                output[1] = char(((yr % 1000) / 100) + 48);
                output[2] = char(((yr % 100) / 10) + 48);
                output[3] = char((yr % 10) + 48);
                output[4] = divider;
            }
            if (t.mon<10)
                output[3 + offset] = 48;
            else
                output[3 + offset] = char((t.mon / 10) + 48);
            output[4 + offset] = char((t.mon % 10) + 48);
            output[5 + offset] = divider;
            if (t.date<10)
                output[6 + offset] = 48;
            else
                output[6 + offset] = char((t.date / 10) + 48);
            output[7 + offset] = char((t.date % 10) + 48);
            output[8 + offset] = 0;
            break;
    case FORMAT_MIDDLEENDIAN:
            if (t.mon<10)
                output[0] = 48;
            else
                output[0] = char((t.mon / 10) + 48);
            output[1] = char((t.mon % 10) + 48);
            output[2] = divider;
            if (t.date<10)
                output[3] = 48;
            else
                output[3] = char((t.date / 10) + 48);
            output[4] = char((t.date % 10) + 48);
            output[5] = divider;
            if (slformat == FORMAT_SHORT)
            {
                yr = t.year - 2000;
                if (yr<10)
                    output[6] = 48;
                else
                    output[6] = char((yr / 10) + 48);
                output[7] = char((yr % 10) + 48);
                output[8] = 0;
            }
            else
            {
                yr = t.year;
                output[6] = char((yr / 1000) + 48);
                output[7] = char(((yr % 1000) / 100) + 48);
                output[8] = char(((yr % 100) / 10) + 48);
                output[9] = char((yr % 10) + 48);
                output[10] = 0;
            }
            break;
    }
    return output;
}

char *DS1302::getDOWStr(uint8_t format)
{
    char *output = "xxxxxxxxx";
    Time t;
    t = getTime();
```

```cpp
    switch (t.dow)
    {
    case MONDAY:
        output = "Monday";
        break;
    case TUESDAY:
        output = "Tuesday";
        break;
    case WEDNESDAY:
        output = "Wednesday";
        break;
    case THURSDAY:
        output = "Thursday";
        break;
    case FRIDAY:
        output = "Friday";
        break;
    case SATURDAY:
        output = "Saturday";
        break;
    case SUNDAY:
        output = "Sunday";
        break;
    }
    if (format == FORMAT_SHORT)
        output[3] = 0;
    return output;
}


char *DS1302::getMonthStr(uint8_t format)
{
    char *output = "xxxxxxxxx";
    Time t;
    t = getTime();
    switch (t.mon)
    {
    case 1:
        output = "January";
        break;
    case 2:
        output = "February";
        break;
    case 3:
        output = "March";
        break;
    case 4:
        output = "April";
        break;
    case 5:
        output = "May";
        break;
    case 6:
        output = "June";
        break;
    case 7:
        output = "July";
        break;
    case 8:
        output = "August";
        break;
    case 9:
        output = "September";
        break;
    case 10:
        output = "October";
```

```
                    break;
            case 11:
                    output = "November";
                    break;
            case 12:
                    output = "December";
                    break;
        }
        if (format == FORMAT_SHORT)
                output[3] = 0;
        return output;
}

void DS1302::halt(bool enable)
{
        uint8_t _reg = _readRegister(REG_SEC);
        _reg &= ~(1 << 7);
        _reg |= (enable << 7);
        _writeRegister(REG_SEC, _reg);
}

void DS1302::writeProtect(bool enable)
{
        uint8_t _reg = (enable << 7);
        _writeRegister(REG_WP, _reg);
}

void DS1302::setTCR(uint8_t value)
{
        _writeRegister(REG_TCR, value);
}

/* Private */

uint8_t DS1302::_readByte()
{
        pinMode(_data_pin, INPUT);

        uint8_t value = 0;
        uint8_t currentBit = 0;

        for (int i = 0; i < 8; ++i)
        {
                currentBit = digitalRead(_data_pin);
                value |= (currentBit << i);
                digitalWrite(_sclk_pin, HIGH);
                delayMicroseconds(1);
                digitalWrite(_sclk_pin, LOW);
        }
        return value;
}

void DS1302::_writeByte(uint8_t value)
{
        pinMode(_data_pin, OUTPUT);
        shiftOut(_data_pin, _sclk_pin, LSBFIRST, value);
}

uint8_t DS1302::_readRegister(uint8_t reg)
{
        uint8_t cmdByte = 129;
        cmdByte |= (reg << 1);

        uint8_t readValue;

        digitalWrite(_sclk_pin, LOW);
```

```cpp
        digitalWrite(_ce_pin, HIGH);

        _writeByte(cmdByte);
        readValue = _readByte();

        digitalWrite(_ce_pin, LOW);

        return readValue;
}

void DS1302::_writeRegister(uint8_t reg, uint8_t value)
{
        uint8_t cmdByte = (128 | (reg << 1));

        digitalWrite(_sclk_pin, LOW);
        digitalWrite(_ce_pin, HIGH);

        _writeByte(cmdByte);
        _writeByte(value);

        digitalWrite(_ce_pin, LOW);
}

void DS1302::_burstRead()
{
        digitalWrite(_sclk_pin, LOW);
        digitalWrite(_ce_pin, HIGH);

        _writeByte(191);
        for (int i = 0; i<8; i++)
        {
                _burstArray[i] = _readByte();
        }
        digitalWrite(_ce_pin, LOW);
}

uint8_t        DS1302::_decode(uint8_t value)
{
        uint8_t decoded = value & 127;
        decoded = (decoded & 15) + 10 * ((decoded & (15 << 4)) >> 4);
        return decoded;
}

uint8_t DS1302::_decodeH(uint8_t value)
{
        if (value & 128)
                value = (value & 15) + (12 * ((value & 32) >> 5));
        else
                value = (value & 15) + (10 * ((value & 48) >> 4));
        return value;
}

uint8_t        DS1302::_decodeY(uint8_t value)
{
        uint8_t decoded = (value & 15) + 10 * ((value & (15 << 4)) >> 4);
        return decoded;
}

uint8_t DS1302::_encode(uint8_t value)
{
        uint8_t encoded = ((value / 10) << 4) + (value % 10);
        return encoded;
}

void DS1302::writeBuffer(DS1302_RAM r)
{
```

```cpp
        digitalWrite(_sclk_pin, LOW);
        digitalWrite(_ce_pin, HIGH);

        _writeByte(254);
        for (int i = 0; i<31; i++)
        {
                _writeByte(r.cell[i]);
        }
        digitalWrite(_ce_pin, LOW);
}

DS1302_RAM DS1302::readBuffer()
{
        DS1302_RAM r;

        digitalWrite(_sclk_pin, LOW);
        digitalWrite(_ce_pin, HIGH);

        _writeByte(255);
        for (int i = 0; i<31; i++)
        {
                r.cell[i] = _readByte();
        }
        digitalWrite(_ce_pin, LOW);

        return r;
}

void DS1302::poke(uint8_t addr, uint8_t value)
{
        if ((addr >= 0) && (addr <= 30))
        {
                addr = (addr * 2) + 192;

                digitalWrite(_sclk_pin, LOW);
                digitalWrite(_ce_pin, HIGH);

                _writeByte(addr);
                _writeByte(value);

                digitalWrite(_ce_pin, LOW);
        }
}

uint8_t DS1302::peek(uint8_t addr)
{
        if ((addr >= 0) && (addr <= 30))
        {
                addr = (addr * 2) + 193;

                uint8_t readValue;

                digitalWrite(_sclk_pin, LOW);
                digitalWrite(_ce_pin, HIGH);

                _writeByte(addr);
                readValue = _readByte();

                digitalWrite(_ce_pin, LOW);

                return readValue;
        }
        else
                return 0;
}
```