

**POLITECHNIKA ŚLĄSKA**

**WYDZIAŁ AUTOMATYKI, ELEKTRONIKI I INFORMATYKI**

Projekt z Systemów mikroprocesorowych

**Temat: Termometr dwupunktowy z zegarem**

Autorzy: Roman Gajdzik, Michał Nieużyła

Rok III, semestr V, grupa 5

Gliwice, styczeń 2012

Spis treści

[1. Wstęp 3](#_Toc282945161)

[2. Cel i zakres projektu 3](#_Toc282945162)

[3. Harmonogram 4](#_Toc282945163)

[4. Aplikacja sprzętowa 4](#_Toc282945164)

[4.1. Opis części sprzętowej 4](#_Toc282945165)

[4.2. Schemat ideowy oraz płytka drukowana 6](#_Toc282945169)

[5. Aplikacja programowa 7](#_Toc282945171)

[5.1. Opis części programowej 7](#_Toc282945172)

[5.2. Schemat blokowy 11](#_Toc282945173)

[5.3. Problemy w trakcie realizacji 11](#_Toc282945174)

[5.4. Aplikacja 12](#_Toc282945175)

[6. Dalszy rozwój oraz udoskonalenia 20](#_Toc282945176)

[7. Kosztorys 20](#_Toc282945177)

[8. Podsumowanie 21](#_Toc282945178)

[9. Literatura 21](#_Toc282945179)

# Wstęp

Projektem, nad którym zdecydowaliśmy się pracować podczas zajęć z Systemów Mikroprocesorowych jest dwupunktowa stacja mierząca temperaturę wraz z funkcją odmierzania czasu. Jako że temperatura jest podstawową informacją na temat stanu pogody, jak również to, że na rynku jest wiele urządzeń mierzących temperaturę pokojową oraz zewnętrzną, zdecydowaliśmy się samemu stworzyć takie urządzenie. Dodatkową motywacją do zmierzenia się z tym problemem była chęć sprawdzenia swojej wiedzy i umiejętności z zakresu technik mikroprocesorowych.

# Cel i zakres projektu

Celem projektu było zbudowanie urządzenia, który pozwalałoby na mierzenie temperatury w dwóch miejscach jednocześnie, a dodatkowo odmierzałoby czas. Wszystkie te informacje z założeń projektowych miały być wyświetlane na wyświetlaczu LCD wyposażonym w popularny sterownik HD44780. Dodatkowo należało zapewnić możliwość ustawiania daty oraz godziny. Staraliśmy się również ograniczyć koszty naszego projektu do jak najmniejszych rozmiarów.

Etapy projektu:

* Wybór technologii wykonania projektu.
* Wykonanie projektu zgodnie z założeniami.
* Problemy w trakcie tworzenia aplikacji
* Uruchomienie, weryfikacji i przetestowanie aplikacji.
* Nakreślenie ewentualnych kierunków rozwoju projektu.

# Harmonogram

1. Obsługa wyświetlacza
2. Komunikacja z czujnikiem temperatury
3. Komunikacja z zegarem czasu rzeczywistego
4. Trawienie płytki
5. Integracja całości
6. Testowanie urządzenia

Naszą pracę zaczeliśmy od stworzenia schematu ideowego urządzenia. Na podstawie tego schematu ustaliliśmy wstępny kosztorys. Następnie przystąpiliśmy do wykonania obwodu drukowanego. Najwięcej czasu poświęciliśmy na oprogramowanie naszego urządzenia. Równolegle z programowaniem trwaly prace testowe poszczególnych modułów układu.

# Aplikacja sprzętowa

## Opis części sprzętowej

Centralnym elementem naszego projektu jest mikroprocesor ATMega8. Jest to 8 bitowy procesor zawierający 8 KB pamięci Flash oraz posiadającym 23 linie I/O. W naszym układzie jest on taktowany wewnętrznym oscylatorem RC o częstotliwości 1 MHz.

Jako czujniki temperatury wykorzystaliśmy bardzo popularny układ firmy MAXIM DS18B20. Jest to w pełni cyfrowy termometr komunikujący się z urządzeniami sterującymi za pomocą magistrali 1Wire. Mierzy on temperatury w zakresie od -55°C do +125°C z programowalną rozdzielczością od 9 do 12 bitów, przy czym my wykorzystaliśmy rozdzielczość 12 bitów (skutkuje to 4 bitami odpowiadającymi części ułamkowej temperatury, jednak zwiększa to czas konwersji do 750ms). W zakresie od -10°C do +85°C producent podaje dokładność czujnika na poziomie ±0,5°C. Z racji wykorzystania magistrali 1Wire może on być zasilany zarówno z linii danych jak i z zewnętrznego źródła zasilania. W naszym projekcie wszystkie układy są zasilane zewnętrznie.

Do odmierzania czasu wykorzystaliśmy zegar czasu rzeczywistego firmy DALLAS DS1307.

Jest to układ który wykorzystując kwarc 32kHz samodzielnie odmierza czas oraz przechowuje go w swojej pamięci. Dodatkowo posiada kompensację roku przestępnego do roku 2100.

Układ komunikuje się z mikroprocesorem za pomocą magistrali I2C. Jest to synchroniczna magistrala wykorzystująca jedną linię danych oraz jedną linię zegarową.

Dodatkowo posiada on programowalne wyjście sygnału prostokątnego SQW/OUT, którego używamy do sterowania odczytem temperatury -jako że czujniki temperatury potrzebują 750ms na konwersję, użyliśmy sygnału o częstotliwości 1 Hz pojawiającego się na wyjściu SQW/OUT do wyzwalania przerwania zewnętrznego w mikroprocesorze. Taki sposób postępowania zapewnia nam odczyt temperatury co jedną sekundę.

Wszelkie dane są wysyłane na wyświetlacz alfanumeryczny 2x16 znaków ( zgodny ze standardem HD44780 ).

Do sterowania układem wykorzystaliśmy cztery przyciski chwilowe wykorzystujące pięć linii I/O: każdy przycisk jest połączony ze swoją linią portu, oraz dodatkowo wszystkie przyciski naraz są dołączone do wejścia przerwania zewnętrznego INT1. Zapewnia nam to rozróżnianie przycisków podczas obsługi przerwania.

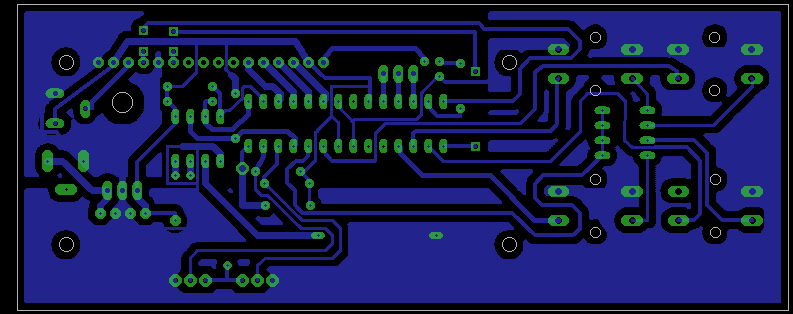
Układ zasilany jest z zewnętrznego zasilacza DC o napięciu 7-25 V.

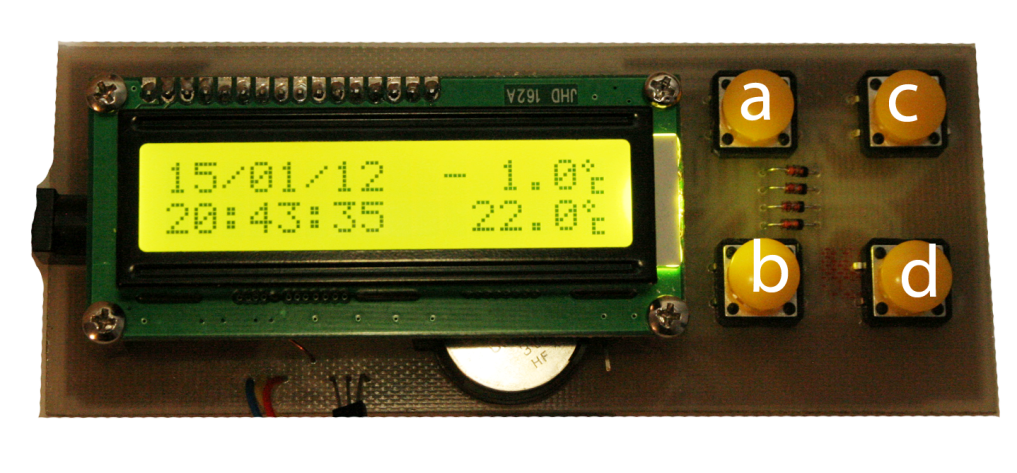
Za pomocą klucza tranzystorowego zrealizowaliśmy sterowanie podświetleniem wyświetlacza LCD.

Dodatkowo na płytce zamontowaliśmy złącze programatora ISP.

## Schemat ideowy oraz płytka drukowana

## C:\Users\mikevein\Documents\eagle\RTC+termo\1234.png





Finalny widok urządzenia

Opis przycisków: a –„+”, b – „-”, c – podświetlenie, d – „enter”

# Aplikacja programowa

## Opis części programowej

Do obsługi wyświetlacza LCD wykorzystaliśmy gotową bibliotekę autorstwa Radosława Kwietnia. Uznaliśmy, że nie ma sensu tworzenia własnych procedur obsługi wyświetlacza od początku, gdyż opierały by się na identyczym schemacie postępowania. Gdybyśmy jednak zdecydowali się na komercyjną sprzedaż naszego urządzenia musielibyśmy napisać własną bibliotekę.

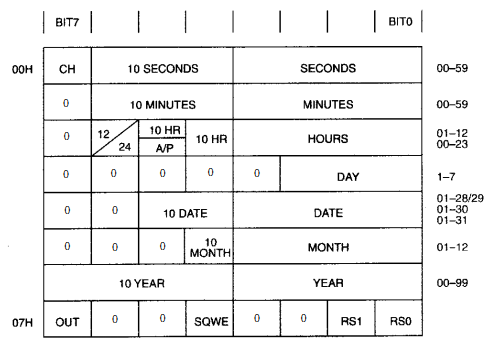
Na ekranie wyświetlana jest data oraz godzina w ścisle określonym formacie:

hh:mm:ss i dd/mm/rr – dlatego zdecydowaliśmy się na użycie dwóch łańcuchów znakowych cyklicznie wysyłanych na LCD, podczas gdy inne funkcje aktualizują dane w tychże łańcuchach.

Dodatkowo został zdefiniowany znak „ºC” zapisany w pamięci wyświetlacza, w przestrzeni użytkownika.

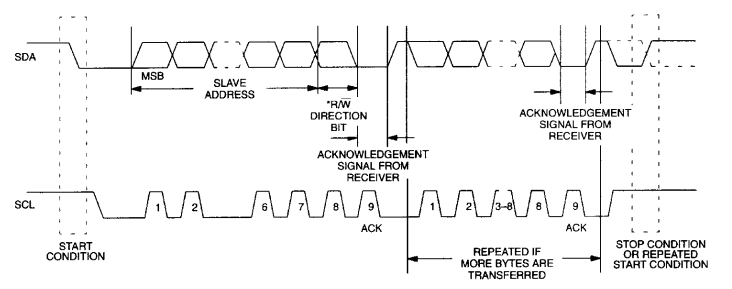
Zegar czasu rzeczywistego DS1307 wykorzystuje do komunikacji magistrale I2C, która jest sprzętowo zaimplementowana w procesorze ATMega8. Podstawe funkcje zapewniające komunikacje po tej magistrali są opisane w dokumentacji mikroprocesora.

Układ DS1307 przechowuje w swoje pamięci aktualne wartości sekund,minut itd. (zakodowanych w kodzie BCD ) w ośmiu rejestrach od 0x00 do 0x07.



Komunikacja za pomocą magistrali I2C opiera się na wykorzystaniu dwóch linii: linii danych SDA oraz lini zegara SCL. Dane z magistrali odczytywane są w chwilach, gdy linia SCL znajduje się w stanie wysokim. Przy SCL w stanie niskim możliwa jest zmiana stanu linii SDA.

Komunikacja z układem odbywa się w ściśle określony sposób: przy rozpoczęciu transmisji musi być wysłany START – przejście SDA ze stanu wysokiego na niski przy SCL w stanie wysokim. Następnie wysyłany jest 7 bitowy adres urządzenia (każde urządzenie wykorzystujące I2C ma ściśle określony adres) oraz bit kierunku : 1 - odczyt, 0 – zapis. Po tym przesyłane są bajty do zapisu na urządzenie typu slave lub z urządenia pobierane są kolejne bajty z pamięci.



Aby usprawnić komunikację z układem zegara napisaliśmy dwie procedury : readFromTWI(int adr) odczytująca bajt z podanego adresu oraz writeToTWI(char adr, char dane) zapisująca podany bajt pod podany adres. Funkcje te wykorzystują sprzętowy interfejs I2C.

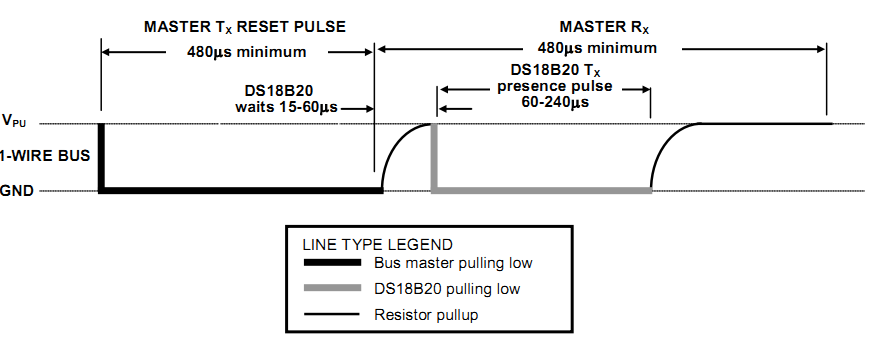
W zegarze niezbędna jest opcja ustawienia czasu oraz daty. Zostało to zrealizowane za pomocą dwóch funkcji SetDate() i SetTime(). Ich działanie jest zbliżone do siebie:

Po wywołaniu funkcji wyświetlany jest odpowiedni komunikat a następnie pobierane są odpowiednie dane z rejestrów układu DS1307 ( dla funkcji SetTime pobierane są wartości sekund, minut i godzin, znów dla SetDate pobierane są wartości dni, miesięcy i lat ). Dane te są przechowywane w łańcuchu znakowym, który jest na bieżąco wyświetlany na LCD. Następnie w pętli aktualizowane są kolejne cyfry tego łańcucha znakowego za pomocą przycisków „+” oraz „-”. Dla każdej cyfry są zaimplementowane odpowiednie ograniczenia, dzięki czemu nie można ustawić godziny 30 bądź 15 miesiąca. Klawisz „Enter” powoduje zatwierdzenie aktualnej cyfry i przejście do następnej. Gdy wszystkie cyfry są juz zatwierdzone zostaje sprawdzona poprawność daty tudzież godziny – w razie błędnego ustawienia zostaje wyświetlony odpowiedni komunikat a następnie użytkownik zostaje poproszony o ponowne ustawienie daty bądź godziny. Po poprawnych ustawieniach funkcja wysyła zmodyfikowane dane do rejestrów układu DS1307.

Do odczytu temperatury zostały użyte czujniki DS18B20 komunikujące się z mikroprocesorem za pomocą magistrali 1Wire. Z racji problemów związanych z obsługą kilku urządzeń na jednej linii zdecydowaliśmy się użyć osobnych linii I/O do każdego czujnika.

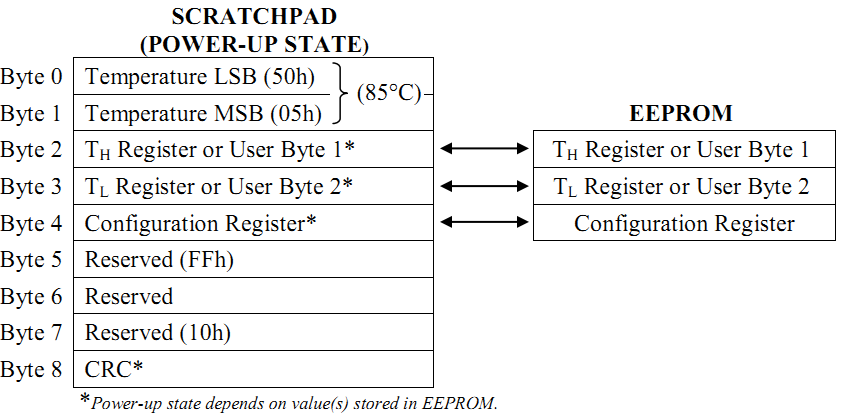
Magistrala 1Wire wykorzystuje do komunikacji tylko jedną linię danych (oraz linię GND i opcjonalnie linię zasilania – dzięki wbudowanemu kondensatorowi układy mogą być zasilane z linii danych).

Procedura komunikacji jest ściśle określona: po pierwsze wysyłany jest impuls RESET, po którym urządzenia zgłaszają gotowość (tzw. Presence pulse):

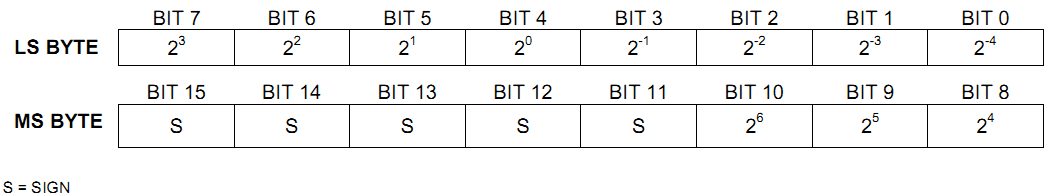


Po zgłoszeniu się urządzeń na magistrali, wysyłana jest komenda odwołująca się do pamięci ROM czujników – może być to żądanie odczytu indywidualnych 64bitowych kodów urządzeń. W naszym projekcie na liniach znajdują się jedynie pojedyncze czujniki, dlatego w tym kroku wysyłamy zawsze komendę SKIP ROM(0xCC).

Po tym następuje przekazanie komendy funkcjonalnej – CONVERT T(0x44) powodującej rozpoczęcie konwersji temperatury lub READ SCRATCHPAD(0xBE) – odczytu zawartości rejestrów czujnika poczynając od adresu 0.



Interesujące nas dane znajdują się w dwóch pierwszych bajtach pamięci:



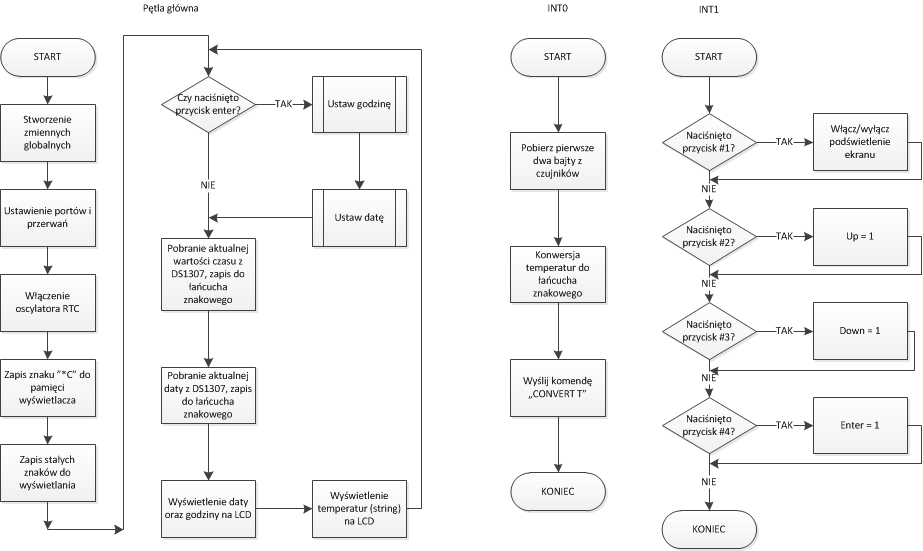
Po odbiorze dwóch bajtów z czujnika możemy przystąpić do dekodowania informacji. Służy do tego funkcja convertTemp(char lowByte, char highByte, char nr).

Starszy bajt na bitach od 7 do 3 zawiera informacje o znaku liczby: gdy S=1 ustawiana jest flaga informująca czy liczba była ujemna. Następnie dane dzielone są na część ułamkową oraz całkowitą za pomocą odpowiednich przesunięć bitowych. W kolejnym kroku, zależnie od podanego numeru czujnika oraz stanu flagi dane są przetwarzane. Gdy liczba jest ujemna dane są negowane w celu powrotu z systemu uzupełnienia do dwóch ( do liczb dodatnich ). Liczby całkowite przeliczane są na znaki przez dzielenie modulo 10 i zapisywane do łańcucha znakowego wskazywanego przez zmienną nr przekazaną w argumencie funkcji( są przygotowane do przesłania na ekran LCD ). Z racji tego, że przeliczanie liczby na string za pomocą funkcji sprintf pochłania bardzo dużo zasobów, ułamek przeliczany jest w następujący sposób: dane w formacie 2-1, 2-2, 2-3, 2-4 są mnożone razy 10/16 co pozwala na uzyskanie cyfry z zakresu 0-9. Taka dokładność jest wystarczająca w termometrze domowym. Łańcuchy znakowe zawierające temperatury zarówno z jednego jak i drugiego czujnika wysyłane są na wyświetlacz.

Z racji tego, że wykorzystujemy pełną, 12-bitową rozdzielczość czujnika niezbędny czas konwersji temperatury wynosi 750 ms. Aby uniknąć opóźnień programowych zdecydowaliśmy się na odczyt z czujników temperatury za pomocą przerwań, wykorzystując sygnał wychodzący z układu DS1307 (1Hz) podłączony pod wejście INT0 mikroprocesora. W programie obsługi przerwania najpierw są odczytywane i konwertowane bieżące dane z czujników, a następnie wysyłany jest rozkaz CONVERT\_T. Zapewnia nam to odczyt dokładnie co jedną sekundę, co jest wystarczającym czasem potrzebnym na konwersję przez czujniki. Mankamentem tego rozwiązania jest początkowe wyświetlenie na wyświetlaczu LCD wartości 85°C ( jest to domyślna wartość rejestrów czujnika po podłączeniu zasilania ). Jednak wartość ta jest wyświetlana tylko przez jedną sekundę, gdyż w następnym odczycie pobierana jest już aktualna temperatura.

Klawiatura wykorzystuje przerwanie zewnętrzne INT1. Każdy przycisk jest połączony ze swoją linią portu oraz wszystkie naraz poprzez bramkę logiczną stworzoną z diod podłączone są do wejścia INT1. Zgłoszenie przerwania oznacza, że został naciśnięty któryś przycisk, dlatego też w procedurze obsługi sprawdzamy który przycisk został naciśnięty a następnie ustawiamy „1” w odpowiedniej zmiennej globalnej. W ten sposób zrealizowane są przyciski „+”, „-”, „Enter”, natomiast czwarty przycisk zmienia stan linii odpowiadającej za podświetlenie wyświetlacza na przeciwny, czyli w efekcie włącza i wyłącza zasilanie diody LED. Po każdym użyciu przez inne procedury zmiennej globalnej przechowującej informacje o naciśnięciu danego przycisku odpowiednia zmienna jest zerowana i przygotowana na następne przyciśnięcie klawisza.

## Schemat blokowy



## Problemy w trakcie realizacji

Pierwszym problemem na który napotkaliśmy było błędne założenie co do obsługi przycisków – przyciski były bezpośrednio razem podłączone do protu INT1, przez co niemożliwe było rozpoznanie, który przycisk został naciśnięty. Wymusiło to zastosowanie bramki AND stworzonej z diod impulsowych 1N4148. Problem został rozwiązany, klawiatura jest w pełni funkcjonalna.

Następnym problemem, z którym przyszło nam się zmierzyć była niemożność uruchomienia magistralii 1Wire. Co dziwniejsze na innej, identycznej płytce oraz przy tych samych procedurach obsługi magistrala działa bez problemów. Do tej pory nie znaleźliśmy przyczyny takie sytuacji.

Kolejną trudnością okazało się spalenie stabilizatora napięcia 7805 oraz mikroprocesora ATMega8. Było to prawdopodobnie spowodowane przypadkowym zwarciem.

## Aplikacja

Poniżej przedstawiamy kod programu. Zdecydowaliśmy nie załączać kodów związanych z obsługą wyświetlacza czyli plików hd44780.h oraz hd44780.c gdyż nie jest to tematem naszych rozważań.

**#define** F\_CPU 1000000UL

**#include** <avr/io.h>

**#include** <avr/interrupt.h>

**#include** <C:/avr/HD44780\_4bit.h>

**#include** <C:/avr/Twowire.c>

**#include** <C:/avr/HD44780\_4bit.c>

**#include** <C:/avr/1wire.h>

**#include** <C:/avr/1wire.c>

**#define** secs 0x00

**#define** mins 0x01

**#define** hours 0x02

**#define** days 0x04

**#define** months 0x05

**#define** years 0x06

//-------------------------------zmienne globalne-------------------------

**volatile** **char** menu=0;

**volatile** **char** up=0;

**volatile** **char** down=0;

**volatile** **unsigned** **char** temperature1[7];

**volatile** **unsigned** **char** temperature2[7];

**volatile** **char** l1[2],l2[2],minusflag;

**int** marker=1;

**char** **readFromTWI**(**int** adr) //funkcja odczytu z układu DS1307 - czyta bajt z podanego adresu

{

start\_TWI();

write\_TWI(0xD0);

write\_TWI(adr);

start\_TWI();

**return**(read\_TWI(0xD1));

}

**void** **writeToTWI**(**char** adr, **char** dane ) //funkcja zapisu do układu DS1307 - zapisuje bajt pod podany adres

{

start\_TWI();

write\_TWI(0xD0);

write\_TWI(adr);

write\_TWI(dane);

stop\_TWI();

}

**void** **convertTemp**(**char** lowByte, **char** highByte, **char** nr) //konwersja temperatury pobranej z DS18B20

{

cli();

**char** ulamek = lowByte;

**char** flag=0;

**if**(highByte>7){ //sprawdzanie znaku odebranej liczby [1 - ujemna]

flag=1;

**switch**(nr){

**case** (0):

minusflag|=(1<<0);

**break**;

**case**(1):

minusflag|=(1<<1);

**break**;

}

}

**else**

{

**switch**(nr){

**case** (0):

minusflag&=~(1<<0);

**break**;

**case**(1):

minusflag&=~(1<<1);

**break**;

}

}

highByte&=0x0F;

highByte=highByte<<4;

lowByte&=0xF0;

lowByte=lowByte>>4;

lowByte|=highByte; //liczba całkowita

**if**(flag) // gdy liczba jest ujemna, bajt jest negowany

{

lowByte=~(lowByte);

ulamek=~(ulamek);

**switch**(nr){

**case** (0):

temperature1[0]='-';

**break**;

**case**(1):

temperature2[0]='-';

**break**;

}}

**else**

{

temperature1[0]=0x20;

temperature2[0]=0x20;

}

**if**(nr==0)

{

temperature1[2]=lowByte%10;

lowByte-=temperature1[2];

temperature1[1]=(lowByte/10);

**if**(temperature1[1]==0)

temperature1[1]+=0x20;

**else**

temperature1[1]+=0x30;

temperature1[2]+=0x30;

ulamek&=0x0F;

ulamek\*=10; //przeliczanie ułamka na częsć dziesiętną

ulamek/=16;

temperature1[4]=ulamek+0x30;

}

**else**

{

temperature2[2]=lowByte%10;

lowByte-=temperature2[2];

temperature2[1]=(lowByte/10);

**if**(temperature2[1]==0)

temperature2[1]+=0x20;

**else**

temperature2[1]+=0x30;

temperature2[2]+=0x30;

ulamek&=0x0F;

ulamek\*=10; //przeliczanie ułamka na częsć dziesiętną

ulamek/=16;

temperature2[4]=ulamek+0x30;

}

sei();

}

**void** **SetDate**() //ustawianie daty

{

**LCD\_Clear**();

**LCD\_Home**();

**LCD\_WriteText**("SET DATE");

**\_delay\_ms**(1500);

**LCD\_Clear**();

writeToTWI(0x07,0x00); //ustawienie adresu do odczytu [DS1307]

**char** received;

**unsigned** **char** j,k,l,c;

**char** line2[9],line[7];

line2[2]=0x2F;

line2[5]=0x2F;

line2[8]=0;

menu=0;

**LCD\_Clear**();

**for** (j=4; j<7; j++) //pobranie aktualnej daty [bajty 4-6]

{

received=readFromTWI(j);

line[7-(2\*(j-3))]=(received&0x0F);

line[6-(2\*(j-3))]=((received&0xF0)>>4);

}

k=0;

**while**(k<6)

{

l=0;

**for**(j=0; j<6;j++)

{

**if**(l==2||l==5 || l==8)

{l++;}

line2[l]=line[j]+0x30;

l++;

}

**LCD\_Home**();

**LCD\_WriteText**(line2);

**if** (k==0 || k==1)

{

**LCD\_GoTo**(k,1);

**LCD\_WriteData**(89);

**\_delay\_ms**(200);

**LCD\_GoTo**(k,1);

**LCD\_WriteData**(' ');

}

**if** (k==2 || k==3)

{

**LCD\_GoTo**(k+1,1);

**LCD\_WriteData**(77);

**\_delay\_ms**(200);

**LCD\_GoTo**(k+1,1);

**LCD\_WriteData**(' ');

}

**if** (k==4 || k==5)

{

**LCD\_GoTo**(k+2,1);

**LCD\_WriteData**(68);

**\_delay\_ms**(200);

**LCD\_GoTo**(k+2,1);

**LCD\_WriteData**(' ');

}

**if** (k==6 || k==7)

{

**LCD\_GoTo**(k+3,1);

**LCD\_WriteData**(68);

**\_delay\_ms**(200);

**LCD\_GoTo**(k+3,1);

**LCD\_WriteData**(' ');

}

**if**(up==1)

{

up=0;

line[k]++;

**if** (k==1 || k==3 || k==5 || k==0)

{

**if** (line[k]==0x0A)

{

line[k]=0x00;

}

}

**if** (k==2)

{

**if** (line[k]==0x02)

{

line[k]=0x00;

}

}

**if** (k==4)

{

**if** (line[k]==0x04)

{

line[k]=0x00;

}

}

**if** (k==7)

{

**if** (line[k]==0x08)

{

line[k]=0x00;

}

}

**if** (k==6)

{

line[k]=0x00;

}

}

**if**(down==1) //nacisniety klawisz "-"

{

down=0;

c=line[k]--;

**if** (k==1 || k==3 || k==5 || k==0)

{

**if** (c==0)

{

line[k]=0x09;

}

}

**if** (k==2)

{

**if** (c==0x00)

{

line[k]=0x01;

}

}

**if** ( k==4 )

{

**if** (c==0x00)

{

line[k]=0x03;

}

}

**if** (k==6)

{

line[k]=0x00;

}

**if** ( k==7 )

{

**if** (c==0x00)

{

line[k]=0x07;

}

}

}

**if**(menu==1) //enter

{

k++;

menu=0;

}

}

**for**(j=4; j<7; j++) //wysłanie zmienionych wartoci, zapisanie ich w pamięci DS1307

{

received=(((line[6-(2\*(j-3))]&0x0F)<<4)|(line[7-(2\*(j-3))]&0x0F));

**LCD\_WriteData**(received);

writeToTWI(j,received);

}

//błędnie ustawiona data - komunikat błędu

**if** ((line[2]==0x01 && line[3]>0x02) ||(line[4]==3 && line[5]>0x01) || (line[0]==0 && line[1]==0) || (line[2]==0 && line[3]==0))

{

**LCD\_Home**();

**LCD\_Clear**();

**LCD\_WriteText**("Blad ustawienia");

**LCD\_GoTo**(0,1);

**LCD\_WriteText**("daty!");

**\_delay\_ms**(3000);

**LCD\_Clear**();

**LCD\_Home**();

**LCD\_WriteText**("Ustaw jeszcze");

**LCD\_GoTo**(0,1);

**LCD\_WriteText**("raz!");

**\_delay\_ms**(3000);

SetDate();

}

**LCD\_Clear**();

writeToTWI(0x07,0x10);

}

**void** **SetTime**(**void**) //funkcja ustawiania czasu

{

**LCD\_Clear**();

**LCD\_Home**();

**LCD\_WriteText**("SET TIME");

**\_delay\_ms**(1500);

**LCD\_Clear**();

writeToTWI(0x07,0x00); //ustawienie adresu do odczytu

**char** received;

**unsigned** **char** j,k,l,c;

**char** line2[9],line[6];

line2[2]=0x3A;

line2[5]=0x3A;

line2[8]=0;

menu=0;

**LCD\_Clear**();

**for** (j=0; j<3; j++) //pobranie aktualnego czasu

{

received=readFromTWI(j);

line[5-(2\*j)]=(received&0x0F);

line[4-(2\*j)]=((received&0xF0)>>4);

}

k=0;

**while**(k<6) //aktualizacja cyfr [po kolei]

{

l=0;

**for**(j=0; j<6;j++)

{

**if**(l==2||l==5)

{l++;}

line2[l]=line[j]+0x30;

l++;

}

**LCD\_Home**();

**LCD\_WriteText**(line2);

**if** (k==0 || k==1)

{

**LCD\_GoTo**(k,1);

**LCD\_WriteData**(94);

**\_delay\_ms**(200);

**LCD\_GoTo**(k,1);

**LCD\_WriteData**(' ');

}

**if** (k==2 || k==3)

{

**LCD\_GoTo**(k+1,1);

**LCD\_WriteData**(94);

**\_delay\_ms**(200);

**LCD\_GoTo**(k+1,1);

**LCD\_WriteData**(' ');

}

**if** (k==4 || k==5)

{

**LCD\_GoTo**(k+2,1);

**LCD\_WriteData**(94);

**\_delay\_ms**(200);

**LCD\_GoTo**(k+2,1);

**LCD\_WriteData**(' ');

}

**if**(up==1) //nacisniety klawisz "+"

{

up=0;

line[k]++;

**if** (k==5 || k==3 || k==1)

{

**if** (line[k]==0x0A)

{

line[k]=0x00;

}

}

**if** (k==4 || k==2 )

{

**if** (line[k]==0x06)

{

line[k]=0x00;

}

}

**if** (k==0 )

{

**if** (line[k]==0x03)

{

line[k]=0x00;

}

}

}

**if**(down==1) //nacisniety klawisz "-"

{

down=0;

c=line[k]--;

**if** (k==5 || k==3 || k==1)

{

**if** (c==0)

{

line[k]=0x09;

}

}

**if** (k==4 || k==2 )

{

**if** (c==0x00)

{

line[k]=0x05;

}

}

**if** ( k==0 )

{

**if** (c==0x00)

{

line[k]=0x02;

}

}

}

**if**(menu==1) //enter

{

k++;

menu=0;

}

}

**for**(j=0; j<3; j++) //zapis zaktualizowanych danych do DS1307

{

received=(((line[4-(2\*j)]&0x0F)<<4)|(line[5-(2\*j)]&0x0F));

**LCD\_WriteData**(received);

writeToTWI(j,received);

}

//błędne ustawienie czasu

**if** (line[0]==0x02 && line[1]>0x03)

{

**LCD\_Home**();

**LCD\_Clear**();

**LCD\_WriteText**("Blad ustawienia");

**LCD\_GoTo**(0,1);

**LCD\_WriteText**("czasu!");

**\_delay\_ms**(3000);

**LCD\_Clear**();

**LCD\_Home**();

**LCD\_WriteText**("Ustaw jeszcze");

**LCD\_GoTo**(0,1);

**LCD\_WriteText**("raz!");

**\_delay\_ms**(3000);

SetTime();

}

**LCD\_Clear**();

writeToTWI(0x07,0x10);

}

**int** **main**(**void**)

{

//--------------definicja znaków dla LCD------------------------------

**char** znak[8]={0x08,0x14,0x08,0x00,0x07,0x04,0x04,0x07};

temperature2[3]=temperature1[3]=0x2E; //kropka

temperature2[5]=temperature1[5]=0x01; //własny znak "\*C"

temperature2[6]=temperature1[6]=0; //koniec łańcucha

//-----------------------------------------------------------------------

**char** u;

u=readFromTWI(0); //po resecie oscylator RTC zawsze włączony

writeToTWI(0,(u&0x7F)); //zapis 0 do CH

writeToTWI(0x07,0x10); //SQWE enable -

//--------------------------------------ustawienia portów i przerwań

LCD\_RS\_DIR=0xff;

DDRD=0x00;

PORTD=0xF0;

MCUCR|=0x0A;

GICR |= 1 << INT1;

GIFR &= ~(1 << INTF1);

GICR |= 1 << INT0;

GIFR &= ~(1 << INTF0);

sei();

//-----------------------------------------------

**char** line2[9]; //linia dla godziny, stałe znaki ":"

line2[2]=0x3A;

line2[5]=0x3A;

line2[8]=0;

**char** line3[9]; //linia dla daty, stałe znaki "/"

line3[2]=0x2F;

line3[5]=0x2F;

line3[8]=0;

**char** received;

**LCD\_Initalize**(); //inicjalizacja wyswietlacza LCD - interfejs 4bit

DDRC |= (0<<5); //ustawienia portów

DDRC |= (0<<6);

PORTC |= (1<<5);

PORTC |= (1<<6);

**LCD\_WriteCommand**(0x48); //tworzenie własnego znaku - zapis do pamieci

**for**(**unsigned** **char** i=0; i<8; i++) //wyswietlacza pod adres 0x01

{

**LCD\_WriteData**(znak[i]);

}

**for**(;;) //główna pętla programu

{

**if**(menu==1) //nacisniety przycisk enter - przejscie do ustawienia daty i godziny

{

SetTime();

SetDate();

}

cli(); //wyłączenie przerwan - eliminacja błędów przy transmisji

received=readFromTWI(days); //pobranie daty: dni, miesiąc, rok

line3[0]=((received&0xF0)>>4)+0x30;

line3[1]=(received&0x0F)+0x30;

received=readFromTWI(months);

line3[3]=((received&0xF0)>>4)+0x30;

line3[4]=(received&0x0F)+0x30;

received=readFromTWI(years);

line3[6]=((received&0xF0)>>4)+0x30;

line3[7]=(received&0x0F)+0x30;

received=readFromTWI(secs); //pobranie godziny: sekundy, minuty, godziny

line2[7]=(received&0x0F)+0x30;

line2[6]=((received&0xF0)>>4)+0x30;

received=readFromTWI(mins);

line2[4]=(received&0x0F)+0x30;

line2[3]=((received&0xF0)>>4)+0x30;

received=readFromTWI(hours);

line2[1]=(received&0x0F)+0x30;

line2[0]=((received&0xF0)>>4)+0x30;

**LCD\_GoTo**(0,0);

**LCD\_WriteText**(line3); //wyswietlenie daty

**LCD\_GoTo**(0,1);

**LCD\_WriteText**(line2); //wyswietlenie godziny

**LCD\_GoTo**(10,0);

**LCD\_WriteText**((**char**\*)temperature1); //wyswietlenie temperatury z pierwszego, "zewnetrznego" czujnika

**LCD\_GoTo**(10,1);

**LCD\_WriteText**((**char**\*)temperature2); //wyswietlenie temperatury z drugiego, "wewnetrznego" czujnika

**LCD\_GoTo**(10,0);

**if**((minusflag&0x01)==1) //temperatura ujemna - wyswietlenie znaku "-"

**LCD\_WriteData**('-');

**else** **LCD\_WriteData**(' ');

**LCD\_GoTo**(10,1);

**if**((minusflag&0x02)==2)

**LCD\_WriteData**('-');

**else** **LCD\_WriteData**(' ');

sei();

}

**return**(0);

}

**ISR**(INT1\_vect) //przerwanie zewnętrzne INT1 - obsługa klawiatury

{

**if** ((PIND & (1 << PD4)) == 0) //klawisz podswietlający

PORTB^=(1<<PB2);

**if** ((PIND & (1 << PD5)) == 0) //klawisz "-"

down=1;

**if** ((PIND & (1 << PD6)) == 0) //klawisz "+"

up=1;

**if** ((PIND & (1 << PD7)) == 0) //klawisz "enter"

menu=1;

}

**ISR**(INT0\_vect) //przerwanie zewnętrzne INT0 - odbiór temperatury co 1 sec

{

**for**(**int** i=0; i<2; i++)

{

**WireRST**(i);

**send\_byte**(SKIP\_ROM,i);

**send\_byte**(WIRE\_READ,i); //odczyt danych z czujnika, dwa bajty

l1[i]=**read\_byte**(i);

l2[i]=**read\_byte**(i);

convertTemp(l1[i],l2[i],i); //konwersja surowych danych na temperaturę w formacie xx.x\*C

**WireRST**(i);

**send\_byte**(SKIP\_ROM,i);

**send\_byte**(CONVERT\_T,i); //komenda 1Wire - rozkaz przeprowadzenia konwersji temperatury

}

}

# Dalszy rozwój oraz udoskonalenia

Projekt mógłby zostać udoskonalony poprzez dodanie większej ilości czujników temperatury. Dodatkowo dołączenie czujników wilgotności oraz temperatury pozwoliłoby przekształcić nasz projekt w pełni funkcjonalną stację pogodową. Następną możliwością byłaby archiwizacja danych – np. na karcie SD.

Oczywiście na pewno priorytetem będzie optymalizacja kodu, by wykonywał się bardziej efektywnie.

Jedną z dodatkowych funkcji mógłby być również budzik – jego implementacja byłaby stosunkowo prosta: wystarczyłoby porównywać zadaną godzinę z pobieranymi na bieżąco danymi z układu RTC, a następnie w przypadku zgodności uruchamiać generator sygnału dźwiękowego.

# Kosztorys

|  |  |  |
| --- | --- | --- |
| **Element** | **Ilość** | **Cena** |
| ATMega8 | 1 | 5.60 |
| DS18B20 | 2 | 3.99 |
| DS1307 | 1 | 4.80 |
| Wyświetlacz LCD 2x16 znaków | 1 | 13.20 |
| Potencjometr montażowy 1k | 1 | 0.20 |
| Diody 1N4148 | 4 | 0.30(10szt) |
| Rezystory (4k7, 10k, 100R) | 7 | 0.20(10szt) |
| Podstawka DIL8 | 1 | 0.20 |
| Podstawka DIL28 | 1 | 0.30 |
| Bateria CR2032 | 1 | 1.00 |
| Gniazdo pod baterię | 1 | 0.70 |
| Przyciski tact-switch | 4 | 0.50 |
| Stabilizator 7805 | 1 | 0.70 |
| Listwa goldpin | 1 | 0.40 |
| Złącze zasilające | 1 | 0.40 |
| Gniazdo goldpin | 1 | 0.60 |
| Kwarc 32kHz | 1 | 0.40 |
| BC107 | 1 | 0.60 |
| Słupki dystansowe | 4 | 0.75 |
| Zasilacz DC 12V | 1 | 7.40 |
| **SUMA** |  | **49.98 zł** |

# Podsumowanie

Projekt dwupunktowego termometru z funkcją odmierzania czasu okazał się bardzo ciekawym zagadnieniem. Dostarczył nam wiele godzin cennej praktyki inżynierskiej oraz dał okazję do poznania tajników techniki mikroprocesorowej. Podczas realizacji owego projektu zdobyliśmy cenną wiedzę oraz umiejętności takie jak np. projektowanie układów elektronicznych, wytrawianie płytek PCB. Jesteśmy zadowoleni z efektów naszej pracy oraz możemy stwierdzić, że nasze urządzenie działa w stopniu bardzo dobrym. Nasze urządzenie na pewno znajdzie zastosowanie w życiu codziennym i okaże się bardzo użytecznym.

# Literatura

* <http://www.atmel.com/dyn/resources/prod_documents/doc2486.pdf> - dane techniczne procesora ATMega8
* <http://radzio.dxp.pl/hd44780/> - strona autora bibliotek obsługujących wyświetlacz LCD
* <http://datasheets.maxim-ic.com/en/ds/DS1307.pdf> - dane techniczne zegara DS1307
* [**http://datasheets**.maxim-ic.com/en/ds/**DS18B20**.pdf](http://datasheets.maxim-ic.com/en/ds/DS18B20.pdf) - dane techniczne czujnika temperatury DS18B20