Bydgoszcz, dnia 3 – 1 - 2008

Urządzenie sterujące ogrzewaniem z wykorzystaniem mikrokontrolera AVR.

Sebastian Łaskawiec

semestr 7 Telekomunikacji

specjalizacja ISSIZ

grupa 2

Spis treści

[1. Założenia projektowe 4](#_Toc187036460)

[2. Środowiska programowania 4](#_Toc187036461)

[3. Konwencje pisania kodu źródłowego 5](#_Toc187036462)

[4. Najistotniejsze miejsca w kodzie źródłowym 5](#_Toc187036463)

[5. Zabezpieczenia 7](#_Toc187036464)

[WatchDog 7](#_Toc187036465)

[Zabezpieczenie przed rozpoczęciem pracy bez ustawień. 7](#_Toc187036466)

[Przywracanie ustawień po zaniku zasilania 7](#_Toc187036467)

[6. Zestaw uruchomieniowy 7](#_Toc187036468)

[Płytka uruchomieniowa EvB 3.1. 8](#_Toc187036469)

[Elementy zawarte na płytce: 8](#_Toc187036470)

[Opis elementów na płytce 9](#_Toc187036471)

[Schemat ideowy płytki 9](#_Toc187036472)

[Zasilanie. 9](#_Toc187036473)

[Schemat ideowy. 10](#_Toc187036474)

[Widok rzeczywisty 10](#_Toc187036475)

[Diody LED 10](#_Toc187036476)

[Schemat ideowy 11](#_Toc187036477)

[Mikroprocesor ATMEGA 16. 11](#_Toc187036478)

[Schemat ideowy 11](#_Toc187036479)

[Odbiornik podczerwieni oraz czujnik temperatury. 12](#_Toc187036480)

[Schemat ideowy 12](#_Toc187036481)

[Zegar czasu rzeczywistego. 12](#_Toc187036482)

[Schemat ideowy 12](#_Toc187036483)

[Pamięć EEPROM z magistralą szeregową. 13](#_Toc187036484)

[Schemat ideowy 13](#_Toc187036485)

[Wyświetlacz LCD. 13](#_Toc187036486)

[Schemat ideowy. 13](#_Toc187036487)

[Złącze RS232. 14](#_Toc187036488)

[Złącze ISP. 14](#_Toc187036489)

[Programator. 14](#_Toc187036490)

[Kabel RS232. 15](#_Toc187036491)

[Zasilacz. 15](#_Toc187036492)

[Pilot. 16](#_Toc187036493)

[Układ MAX. 16](#_Toc187036494)

[7. Instrukcja obsługi układu 17](#_Toc187036495)

[8. Instrukcja obsługi aplikacji na PC 22](#_Toc187036496)

[9. Załączniki do dokumentacji 29](#_Toc187036497)

[10. Przyszłość projektu 29](#_Toc187036498)

# Założenia projektowe

Zaprojektowanie oprogramowania mikrokontrolera AVR do pracy, jako inteligentny czujnik temperatury. Urządzenie dokonując pomiaru ocenia czy temperatura w pomieszczeniu zawiera się w zakresie temperatur ustawionych przez użytkownika. W przypadku za niskiej temperatury urządzenie wysyła sygnał do włączenia ogrzewania - kaloryfera (lub innego urządzenia grzewczego). Jeżeli natomiast temperatura jest za wysoka mikrokontroler wysyła sygnał, aby uruchomić urządzenia chłodzące. Dodatkowo zostanie stworzone oprogramowanie na komputer klasy PC umożliwiające kontrolowanie stanu urządzenia oraz ustawianie parametrów.

* Funkcje czujnika
* Całościowe sterowanie pilotem (dekodowanie ramki RC5)
* Tryb pomiaru aktualnej temperatury (oprogramowanie czujnika DALLAS)
* Ustawienie i wyświetlanie aktualnej daty oraz godziny (oprogramowanie zegara czasu rzeczywistego PCF)
* Dwa tryby ustawienia temperatury – tryb nocny oraz dzienny
* Wyłączenie urządzenia (dalsze monitorowanie godziny, ale nie wysyłanie sygnałów do innych urządzeń odpowiedzialnych za utrzymanie temperatury)
* Ustawienie możliwości podłączenia się aplikacji
* Wykrycie zużycia energii 1
* Oprogramowanie interfejsu USB [[1]](#footnote-2)
* Funkcje aplikacji na PC
* Ustawianie oraz odczytywanie parametrów sterujących czujnikiem (temperatury, trybu itp.)

# Środowiska programowania

Przed wykonaniem projektu należało podjąć decyzje odnośnie środowisk programowania zarówno dla mikrokontrolera jak i dla komputera PC.

Dla mikrokontrolera AVR do wyboru są dwa języki programowania – C i Asembler. Język C jest moim zdaniem dużo bardziej czytelny i elastyczny, wprowadzenie modyfikacji do programu nie wymusza modyfikowania wielu linii kodu. Z drugiej strony kod wyprodukowany przez kompilator jest o wiele mniej optymalny, niż napisany w Asemblerze. Wykonując ten projekt postanowiłem postawić na przejrzystość i wygodę, nie na szybkość działania i optymalność kodu wynikowego.

Wybór środowiska na PC jest dużo bardziej złożony. Aplikację można by wykonać niemal w dowolnym języku. Zdecydowanie najbardziej optymalny byłby natywny język C++ z wykorzystaniem bibliotek WinAPI. Moim zdaniem język C++ nie jest już najwygodniejszym i najbardziej przejrzystym językiem programowania. Uważam, że przyszłość należy do języków wyższego poziomu i znacznie bardziej portowalnych. Bardzo interesująca jest platforma .NET i „flagowy” język tejże platformy C#.

Wybrałem język C# oraz platformę .NET ze względu na:

- przejrzystość zapisu

- potężne biblioteki

- bardzo przyjazny interfejs aplikacji do tworzenia GUI.

- bezproblemowa obsługa portów szeregowych (w Javie jest z tym kłopot)

# Konwencje pisania kodu źródłowego

Aplikacja, jak i kod źródłowe są napisane w języku angielskim. Uważam, że jest to najlepszy język do tworzenia opisów kodu źródłowego. Nigdy nie wiadomo czy dany produkt będzie w przyszłości rozwijany i przez kogo. Język angielski jest językiem uniwersalnym.

# Najistotniejsze miejsca w kodzie źródłowym

Plik main.c, funkcja main

Tutaj jest główna pętla programu. Po inicjalizacji wszystkich urządzeń układ może się znaleźć w jednym w dwóch głównych stanów – stanu komunikacji z komputerem lub pracy (dotyczy to także interakcji z użytkownikiem)

Plik convertions.c

W tym pliku znajdują się wszystkie funkcje odpowiedzialne za konwertowanie i rzutowanie jednego typu danych na drugi – nie wszystkie są używane. Ten plik „powstawał” z całym projektem, zawsze gdy była potrzeba zmienienia jednego typu danych na drugi, była dokładana funkcja, która była w stanie temu sporostać.

Plik delay.c

Wszystkie funkcje odpowiadające za opóźnienia.

Plik ds18d20.c

Obsługa czujnika temperatury Dallas. Implementacja ograniczała się do dokładnego zapoznania z formatem, jakim posługuje się Dallas. Po dokładnym zapoznaniu się z instrukcją użytkownika nie było problemów z oprogramowaniem urządzenia.

Plik global.h

Plik nagłówkowy całego projektu.

I2c.h

Biblioteki do programowego posługiwania się interfejsem I2C. Można także posługiwać się wbudowanym protokołem TWI (Two wire interface). Podczas wykonywania projektu chciałem się zapoznać z tym interfejsem. Najlepszym sposobem było na to zaimplementowanie funkcji realizujących komunikację między mikrokontrolerem a urządzeniami znajdującymi się na interfejsie. Zapis i odczyt jest realizowany przez dwa bloki funkcji. Zapis/Odczyt bitów i bajtów.

LCD.c

Biblioteka zajmująca się wyświetlaniem tekstu i innych danych na wyświetlaczu LCD. Najbardziej przydatnymi funkcjami tej biblioteki jest zapisywanie całego łańcucha znaków i liczby o określonej podstawie.

Options\_menu.c

Jak sama nazwa wskazuje zawiera funkcje do obsługi menu OPTIONS.

PC\_connection.c

Zawiera zbiór funkcji do komunikacji z komputerem PC poprzez UART. Wysyłanie komend jest bardzo prostym zabiegiem ograniczającym się do wysłania z komputera łańcucha znaków zakończonym „!”. Ten znak jest interpretowany jako koniec rozkazu.

PC\_menu.c

Obsługa menu połączenia komputera z AVR.

Pcf8583.c

Obsługa zegara czasu rzeczywistego. Oprócz zapisu i odczytu zegara zawiera także funkcje do ustawiania alarmu (początkowo odliczanie do godziny nocnej było wykonywane poprzez zapisanie godziny alarmu).

Rc5.c

Dekoder ramki RC5. Jest to największa biblioteka do AVR mojego autorstwa. Wymagała spędzenia wielu godzin nad instrukcjami poszczególnych producentów sprzętu. Szczegółowe informacje o poszczególnych formatach ramek są na stronie producentów.

Remote\_choice.c

Odpowiada za poprawne interpretowanie znaków wysyłanych przez pilota.

Temp.c

Odpowiada za odczyt i wyświetlenie temperatury na LCD.

Timer\_menu.c

Funkcje odpowiedzialne za komunikację z zegarem czasu rzeczywistego.

Uart.c

Komunikacja poprzez interfejs UART. Wysyła i odbiera komunikaty poprzez w/w interfejs. Podczas odbierania wiadomości znaki umieszczane są w buforze i jest przestawiana odpowiednia flaga. Dzięki takiemu zabiegowi łatwo sprawdzić, czy czekają jakieś odebrane dane.

# Zabezpieczenia

Zabezpieczenie układu jest integralną częścią projektu. Należy wprowadzić dwa rodzaje zabezpieczeń – programowe i sprzętowe. Sprzętowym zabezpieczeniem jest układ WatchDog, programowo zrealizowane jest zabezpieczenie przez niebezpiecznie wysoką temperaturą oraz pracą bez wprowadzenia ustawień.

## WatchDog

Watchdog jest to najczęściej układ elektroniczny mający za zadanie zrestartować układ jeżeli przebywa zbyt długo w stanie zawieszenia. Działa on w bardzo prosty sposób próbując zrestartować układ co pewien okres czasu. Gdy mikrokontroler działa normalnie, broni się przed taką próbą. Natomiast, gdy nastąpiło zawieszenie układu, wówczas zostaje on zresetowany. Dlatego tak ważne jest zabezpieczenie układu przed rozpoczęciem pracy bez wprowadzenia danych.

Istnieją dwa tryby pracy układu WatchDog – sprzętowa (nie wymaga żadnej implementacji ze strony programu), oraz programowa. Metoda programowa polega na ustaleniu czasu, po jakim nastąpi reset układu i restartowanie licznika Watchdoga w programie. Postanowiłem pozostawić metodę sprzętową w projekcie, gdyż istnieje w programie wiele wewnętrznych pętli i nawet, gdy w każdej następować reset licznika, to i tak układ zachowywać się dziwnie i od czasu do czasu się restartował.

## Zabezpieczenie przed rozpoczęciem pracy bez ustawień.

Należy także zastanowić się nad sytuacją, gdy układ nie posiada wprowadzonych danych i rozpoczyna pracę. Taka sytuacja mogła by prowadzić do absurdalnych zachowań układu. Aby tego uniknąć wprowadzono zabezpieczenie takie, że układ nie rozpocznie pracy, gdy nie ma wprowadzonej daty, godziny, temperatury i czasu rozpoczęcia godziny nocnej.

## Przywracanie ustawień po zaniku zasilania

Niestety w tym projekcie nie została taka ewentualność przewidziana. Wymagało by to modyfikacji płytki uruchomieniowej (a konkretniej układu zegara RTC, tak aby był podtrzymywany baterią).

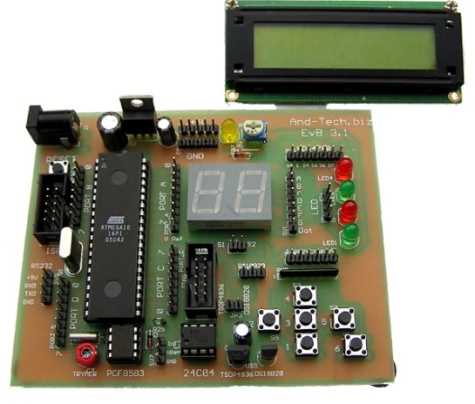
## Zabezpieczenie przed wysoką temperaturą

Układ posiada zabezpieczenie przed wprowadzaniem niepoprawnych danych w postaci zbyt wysokiej i zbyt niskiej temperatury, za której utrzymanie jest odpowiedzialny układ.

# Zestaw uruchomieniowy

Projekt oparty jest na gotowej płytce uruchomieniowej Evb 3.1. Jest ona wyposażona w elementy niezbędne do wykonania zaplanowanych funkcji. Mówiąc o zestawie gotowym do pracy nie możemy zapomnieć o innych integralnych składnikach naszego zestawu. Należą do niego zasilacz, wyświetlacz LCD 2 x 16 znaków, pilot zdalnej obsługi, programator STK200 oraz układ umożliwiający komunikację z komputerem.

## Płytka uruchomieniowa EvB 3.1.

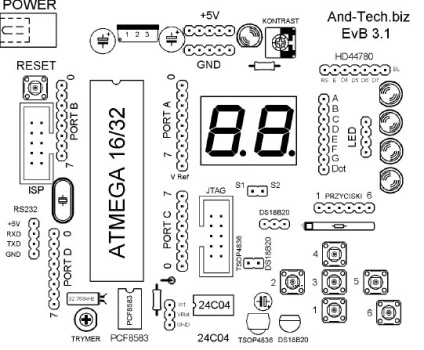


Prezentowana powyżej płytka zawiera wiele elementów funkcyjnych. Opisane zostaną tylko elementy ważne z punku widzenia projektu.

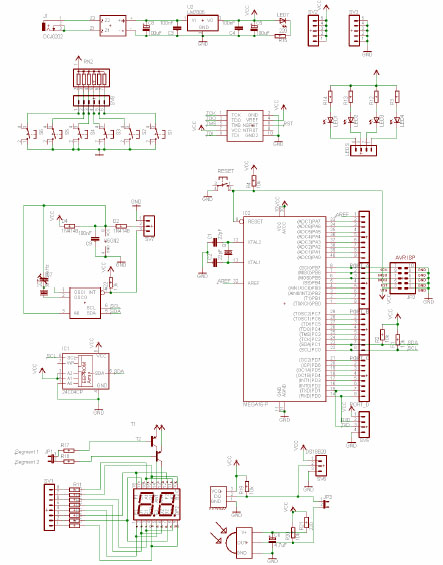
### Elementy zawarte na płytce:

* Procesor AVR ATMega16
* Pamięć EEPROM
* Zegar czasu rzeczywistego
* Odbiornik podczerwieni
* Czujnik temperatury
* 6 przycisków
* 4 diody LED
* Wyświetlacz 2 x 7 segmentów
* Złącze RS232
* Złącze ISP
* Złącze JTAG
* 5 pinów napięcia + 5V
* 5 pinów masy

### Opis elementów na płytce



### Schemat ideowy płytki

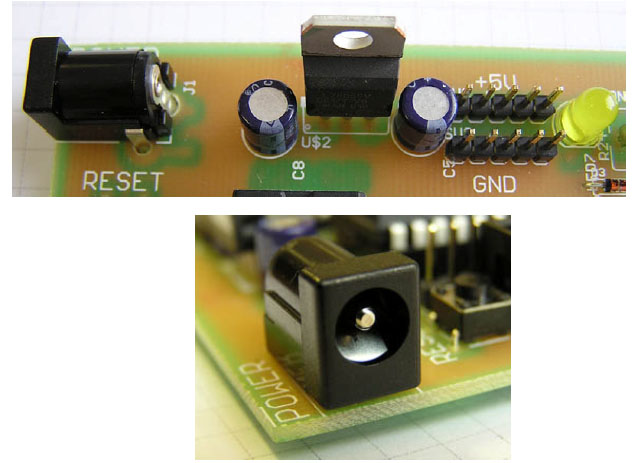


## Zasilanie.

### Schemat ideowy.



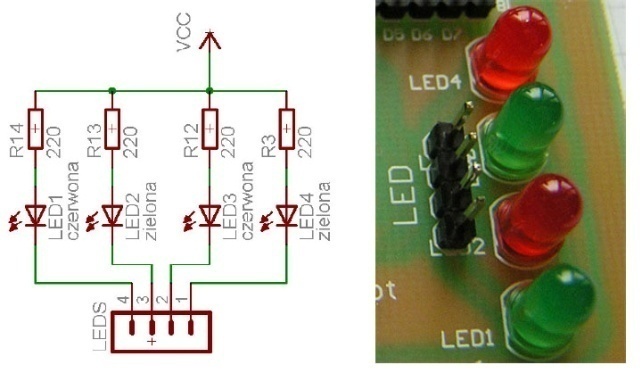
### Widok rzeczywisty



Do płytki należy podłączyć dowolny zasilacz o napięciu powyżej 8 V oraz natężeniu co najmniej 300 mA. Polaryzacja wtyczki zasilającej jest obojętna, ponieważ dla zabezpieczenia płytki przed niewłaściwym podłączeniem zasilania zastosowano mostek prostowniczy. Po podłączeniu zasilania do płytki powinna zapalić się żółta dioda LED. W przypadku nadmiernego nagrzewania sie stabilizatora napięcia U2 należy zainstalować na nim radiator.

## Diody LED

### Schemat ideowy



Diody są podłączone do +5V poprzez drabinkę rezystorów o wartości 220 Ω, w celu włączenia diody musimy podłączyć ją do masy (0 logicznego). Każda z nich informuję nas o czymś innym.

LED1 – nadawanie sygnałów z pilota

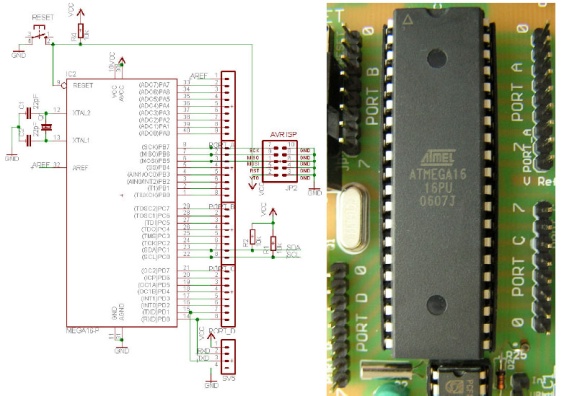
LED2 – tryb połączenia z komputerem

LED3 – układ włączony / wyłączony

LED4 – włączenie ogrzewania

## Mikroprocesor ATMEGA 16.

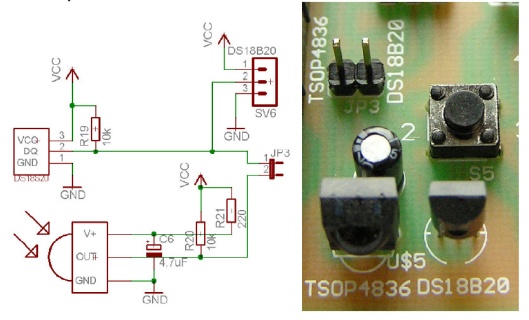
### Schemat ideowy



Procesor ATmega16 to 8-bitowy komputer jednoukładowy zbudowany na architekturze AVR-RISC firmy Atmel, co zapewnia bardzo szybkie wykonywanie instrukcji. Na wykonanie większości z rozkazów potrzeba tylko jednego cyklu. Układ posiada 32 we/wy, zawiera m. in. programowalny seryjny UART, przetwornik A/D 10-bitowy 8 kanałowy, interfejs SPI, watchdog-timer, komparator i trzy timery/liczniki - dwa 8-bitowe i jeden 16-bitowy, cztery kanały z wyjściem PWM, JTAG-interface i zegar czasu rzeczywistego z oddzielnym oscylatorem. Funkcja zerowania przy załączeniu zasilania i 6 położeń o niskim poborze prądu. Programy zapisywane są w pamięci Flash EPROM i mogą być programowane podczas, kiedy układ jest zamontowany w gotowym systemie (ISP).

## Odbiornik podczerwieni oraz czujnik temperatury.

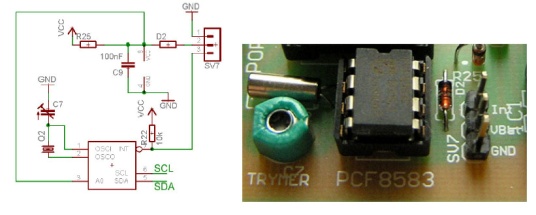
### Schemat ideowy



Odbiornik podczerwieni to TSOP4836 oraz czujnik temperatury to DS18B20.

## Zegar czasu rzeczywistego.

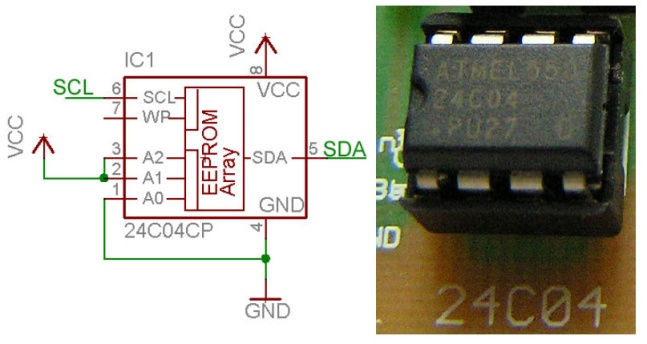
### Schemat ideowy



Zegar czasu rzeczywistego PCF 8583.

## Pamięć EEPROM z magistralą szeregową.

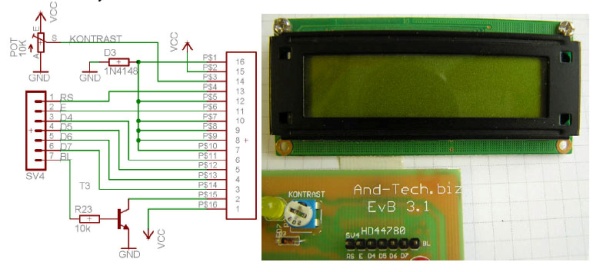
### Schemat ideowy

****

Pamięć EEPROM z magistralą szeregową 24C04.

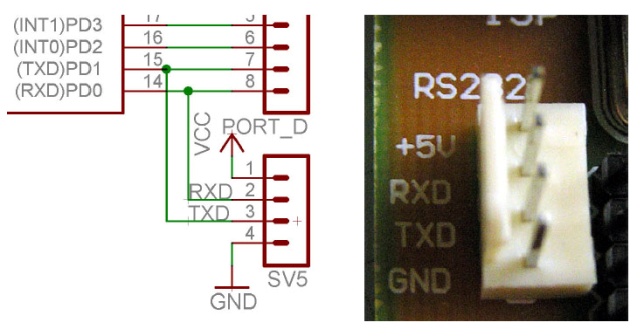
## Wyświetlacz LCD.

### Schemat ideowy.



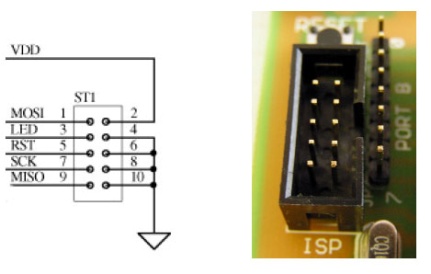
Wyświetlacz LCD HD44780 wraz z możliwością ustawiania kontrastu wyświetlanych napisów.

## Złącze RS232.

****

Złącze RS232 wykorzystywane jest do komunikacji z komputerem PC.

## Złącze ISP.

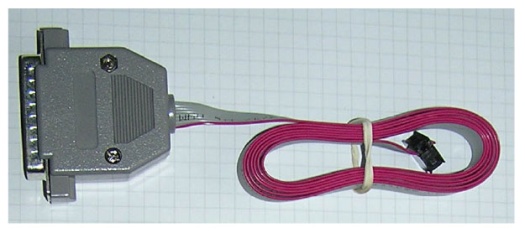
****

Złącze ISP wykorzystywane jest do programowania procesora zainstalowanego na płytce.

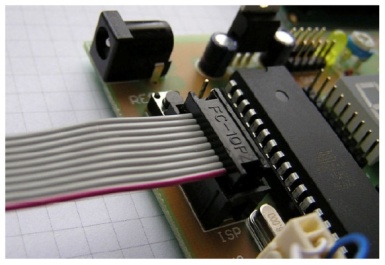
Do złącza podłącza sie programator poprzez który, z komputera PC, wgrywane jest oprogramowanie procesora.

## Programator.

Prezentowany poniżej programator jest kompatybilny z programatorem STK200/300.



Podłączenie programatora do układu.



## Kabel RS232.

Do komunikacji układu z komputerem został użyty przewód pracujący na złączu portu RS232 (COM). Na zdjęciu poniżej został przedstawiony także adapter RS232 – USB.



## Zasilacz.

Zastosowany zasilacz to wersja uniwersalna. Ma on wiele wtyczek w tym jedną pasującą do układu. Jego napięcie wyjściowe może być nastawione na 1 z 7 możliwości, w naszym przypadku to 9V (minimalne wymaganie to 8 V). Maksymalny prąd obciążenia naszego zasilacza to 1000 mA (nasz układ pracyje przy około 300 mA).



## Pilot.

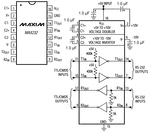
Zastosowany pilot został wybrany metodą prób i błędów (brak specyfikacji technicznej pilotów – częstotliwości ich pracy). Na zdjęciu poniżej zostały zaznaczone przyciski funkcyjne używane do obsługi naszego układu.

Są to przyciski:

* POWER
* TELETEXT (w czerwonej ramce, I od lewej strony)
* SLEEP (w zielonej ramce, II od lewej strony)
* MUTE (w żółtej ramce, III od lewej strony)
* VOL – /+
* ENTER
* Klawisze numeryczne

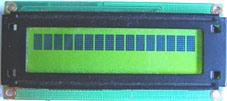
## Układ MAX.

Układ Max 232 ma na celu poprawienie transmisji danych pomiędzy układem a komputerem.

****

# Instrukcja obsługi układu

Po podłączeniu układu do zasilacza, układ włącza się on automatycznie. Zapala się pomarańczowa dioda informująca o włączeniu układu oraz wyświetlacz LCD. (widok poniżej)



W tym momencie należy użyć przycisku RESET, który znajduje się nad złączem ISP.



Układ informuje użytkowanika, że jest gotowy do pracy. Wskazuje też użytkownikowi na wprowadzenie własnych ustawień. W tym celu musimy posłużyć się pilotem. Naciskając przycisk “Sleep” (zielony) przechodzimy do ustawień podstawowych parametrów układu.



Na wyświetlaczu widać godzinę, przedstawianą w trybie 24h, ustawioną obecnie w układzie. Grot strzałki po prawej stronie godziny ukazuje nam możliwość przejścia dalej, w celu zobaczenia kolejnych danych. W tym celu używamy przycisku zgodnego ze strzałką – “Vol +”.



Na zdjęciu powyżej przedstawiona jest data ustawiona w układzie. Jest ona przedstawiana w trybie DD/MM/RRRR. Grot strzałki po prawej stronie daty ponownie ukazuje nam możliwość przejścia dalej, w celu zobaczenia kolejnych danych. Wykonujemy taki sam krok jak przed chwilą.



Napis “SET TEMP” oznacza ustawienia temperatury. W celu jej ustawieni klikamy na pilocie przycisk “Enter”.



Mrugający znacznik na pierwszej cyfrze oznacza, że przeszliśmy w tryb zmiany ustawienia temperatury. Kolejne temperatury oznaczają

**<temp. w dzień / temp. w nocy +- gr. błędu C>**

Wybierane wartości temperatury są podawane w stopniach Celsjusza. W celu wpisania pożądanych wartości posługujemy się klawiszami numerycznymi na pilocie. Po ich wpisaniu układ automatycznie wyjdzie z trybu wpisywania. Przechodzimy dalej.



Ukazuje nam się napis “SET TIME”. Oznacza on ustawienia godziny. Jest ona przedstawiana w trybie HH:MM:SS. W celu jej ustawieni klikamy na przycisk „enter”. Ustawiamy ją analogicznie jak temperatury.



Idziemy dalej.



Ukazuje nam się napis “SET DATE”. Oznacza on ustawienia daty. Jest ona przedstawiana w trybie DD/MM/RRRR. Ustawiana jest tak samo jak godzina po naciśnięciu przycisku “Enter”.



Idziemy dalej.



W tym miejscu ustawiana jest godzina początku i czas trwania trybu nocnego wyrażona w pełnych godzinach.

**< godz. początku trybu nocnego/ czas trwania >**

Postępujemy analogicznie do czynności wykonanych wcześniej.

Jest to ostatnie ustawienie, gdyż nie widzimy strzałki po prawej stronie. Jeśli nie chcemy zmieniać poszczególnych etapów ustawień omijamy je idąc dalej przyciskiem “Vol +”. Przycisk “Vol - ” umożliwia nam cofanie się do poprzednich opcji. W celu wyjścia z przedstawionej części opcji klikamy ponownie przycisk “Sleep”. Jego zadaniem jest wejście lub wyjście z tej części opcji.

W celu wejścia do kolejnej części opcji klikamy na przycisk “Mute” w żółtej otoczce.

Widzimy teraz poniższy napis.



Klikając “Vol +” przechodzimy dalej.

Pojawia się nam zapytanie “WORK ?”. Mamy do wyboru 2 opcje ON lub OFF. Mają one na celu włączenie lub wyłączenie pracy układu. Zmieniamy je przyciskiem “Enter”.



Kolejną opcją jest “PC CONNECTION”, czyli połączenie z komputerem. Opcja ta w trybie “Enable” umożliwia prace z komputerem poprzez port RS232, natomiast w trybie “Disable” wyłącza tą możliwość. Zmieniamy je przyciskiem “Enter”.



Po wyjściu z ustawień pokazuje nam się widok poniżej.



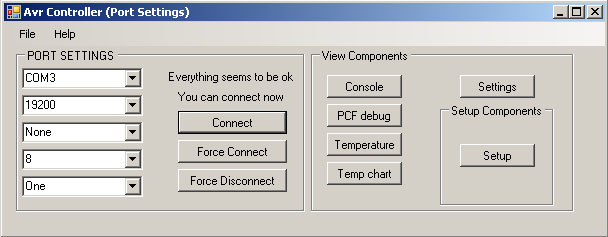
“Ready” oznacza, że układ pracuje. Informacje poniżej to:

**GG:MM/godz. pocz. tryb. nocnego / godz. końca tryb. nocnego / D - tryb dzień, N - tryb noc/ temp. bieżąca**

# Instrukcja obsługi aplikacji na PC

Integralną częścią projektu jest oprogramowanie na PC komunikujące się z układem.

W naszym przypadku komunikacja ta odbywa się przez port RS232. Po podłączeniu układu do portu RS232 lub USB możemy odpalić naszą aplikacje. Pojawia się jej okno(widok poniżej).



Jak widać główne okno aplikacji jest podzielone na dwie podstawowe części.

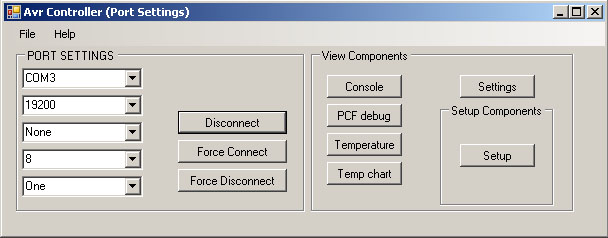
Pierwsza z nich to “PORT SETTINGS”. W tej części ustawiane są parametry połączenia i zestawiane jest samo połączenie. Po lewej stronie widnieje 5 rozwijanych pasków. Tutaj ustawiamy parametry połączenia. W gruncie rzeczy są one takie jak na rysunku poza jedną, która może ulegać zmianie. Jest ona w pierwszej linii “COM3”. Jest to port pod który podpięty jest nasz układ. W celu nawiązania połączenia klikamy na przycisk “Connect”. Na wyświetlaczu pokazuję się napis “Connecting...” oraz pasek postępu zestawiania połączenia.



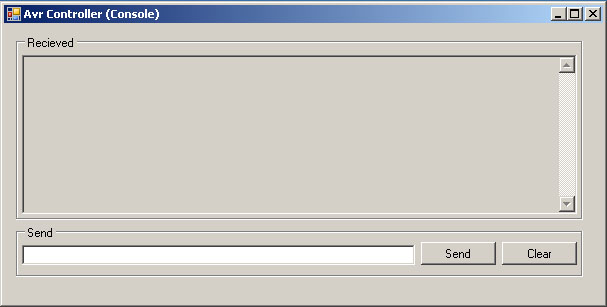
Po nawiązaniu wyświetla nam się kolejny napis, który oznajmia, że układ jest gotowy do pracy.



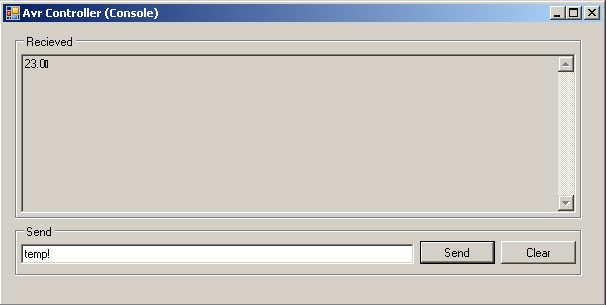
Możemy teraz przejść do drugiej części naszej aplikacji o nazwie “View components”. Znajdziemy tu odczyt godziny, daty oraz obecnej temperatury z układu. Możliwe jest także odczytanie zapisanych w układzie parametrów oraz ich modyfikacja.



Będę objaśniał po kolei poszczególne funkcje. Pierwsza z nich to “Console”. Klikamy na nią.

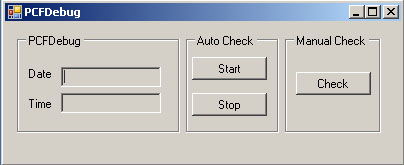


Pojawia się nam oko konsoli z którego możemy zadać układowi pytanie o poszczególne parametry. Przykładem niech będzie zapytanie o temperaturę obecną. Wpisujemy w linii komend “Send” napis *temp!* i potwierdzamy go przyciskiem Send. Otrzymujemy odpowiedź w następującej formie.



Termometr układu wykrywa temperaturę otoczenia na poziomie 23,0 stopnia C.

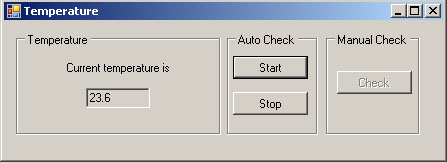
Kolejną funkcją jest PCF debug.



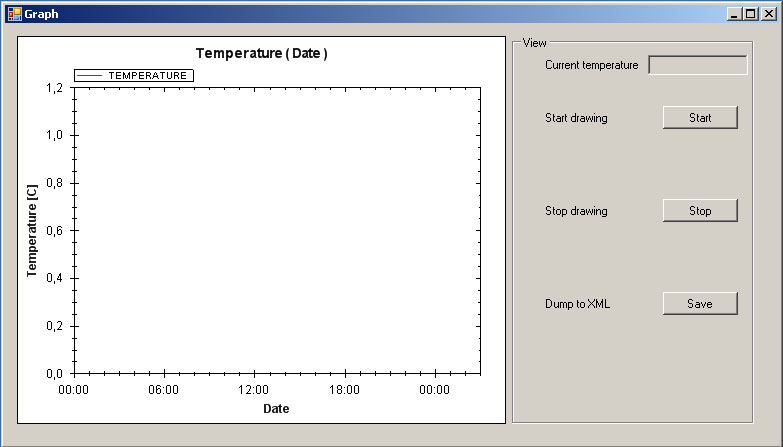
Tutaj mamy możliwość sprawdzenia jednokrotnego lub śledzenia zmian zegara i daty ustawionych w układzie. W celu sprawdzenie godziny i daty w danym momencie klikamy na przycisk Check. W celu ich śledzenia na bieżąco klikamy Start, aby zatrzymać Stop. Na rysunku poniżej został sposób przedstawiania informacji przez aplikacje.



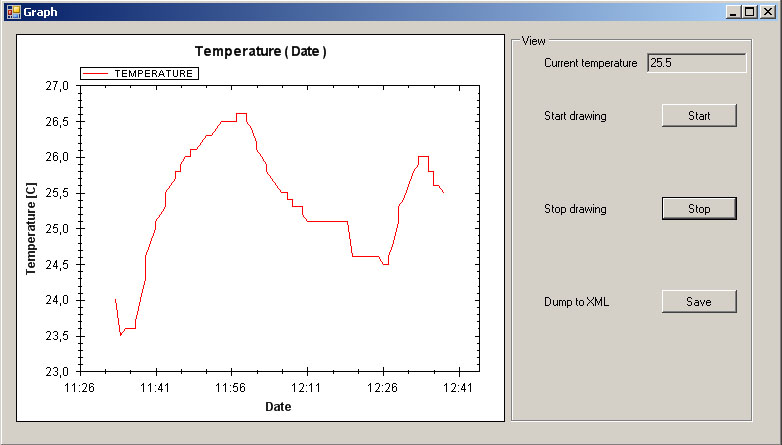
Kolejną opcją jest Temperature. Działa ona podobnie jak poprzednia funkcja, tylko efektem końcowym jest wyświetlanie temperatury.



Następny przycisk Temp Chart przechodzi do modułu kreślącej zmiany temperatury w czasie.

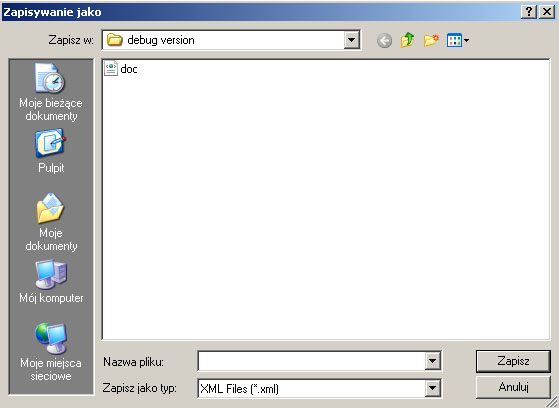


Na osi Y przedstawiana jest wartość temperatury, a na osi X przedstawiony jest czas, który automatycznie się moduluje i dopasowuje do jego upływu. W celu rozpoczęcia kreślenia naciskamy przycisk Start.



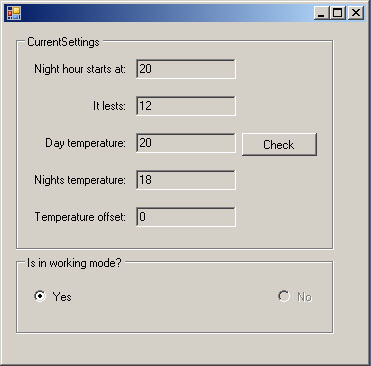
Po zatrzymaniu przyciskiem Stop wykres przestaje być rysowany, po ponownym włączeniu rysowania wykres jest malowany jednak o godziny, w której został ponownie wystartowany. Okres pomiędzy czasem zatrzymania a wznowienia rysowany jest poziomą linią prostą.

Istnieje możliwość zapisania pobranych danych w formacie XML. Załączone są tam informacje na temat godziny daty i temperatury. Aby to uczynić klikamy Save.

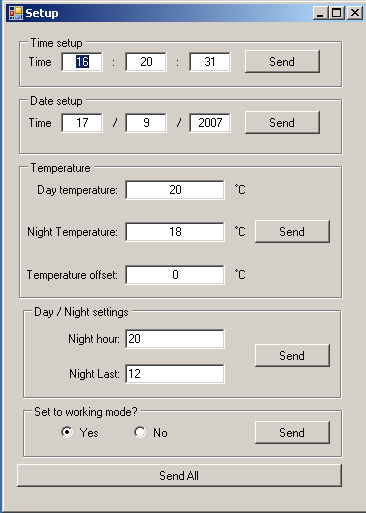


W tym momencie wybieramy ścieżkę oraz nazwę docelowego pliku.

Kolejną funkcją jest Settings. Po kliknięciu Check pokazują się nam aktualne ustawienia układu wprowadzone przy jego pierwszym włączeniu lub po późniejszych modyfikacjach.



Ostatnim możliwym do wyboru przyciskiem jest Setup. W tej zakładce możemy modyfikować wszystkie ustawania parametrów naszego sterownika. Ich autoryzowanie odbywa się poprzez wysłanie jednego lub wszystkich składników na raz. Klikamy przycisk Send przy zmodyfikowanym parametrze lub w celu uaktualnienia wszystkich ustawień Send All umiejscowionym na samym dole okna.



Ostatnim zagadnieniem wymagającym omówienia jest logowanie. W katalogu roboczym aplikacji znajduje się plik log.txt. Umieszczone są w nim informacje o wyjątkach, które zostały złapane w aplikacji oraz o wszystkich krokach, jakie wykonał użytkownik.

# Załączniki do dokumentacji

* Schemat UML aplikacji przeznaczonej na platformę PC (diagram klasowy) w pliku PDF
* Dokumentacja kodu zarówno aplikacji na PC jak i na AVR
* Instrukcja użytkownika uruchamiana z poziomy aplikacji

# Przyszłość projektu

Niniejszego projektu nie można jeszcze uznać za skończonego. Zostały wykonane założenia projektowe, ale urządzenie nie nadaje się do sprzedaży komercyjnej. Wymagane jest wykonanie jeszcze wielu testów w różnych warunkach. Wykonując program starałem się sprawdzić jak urządzenie się zachowuje przy absurdalnych danych wejściowych. Należy jednak podkreślić, iż sprawdzanie aplikacji przez twórcę jest bardzo nieefektywne, ponieważ to co dla programisty i twórcy projektu jest czymś normalnym, dla „szarego” użytkownika jest abstrakcją.

Do zrobienia jeszcze pozostało:

* Modyfikacja płytki i układu w taki sposób, aby było możliwe odtworzenie danych i normalna praca po zaniku prądu
* Zmiana interfejsu komunikacji między PC i układem (obecnie jest wykorzystywany UART, należy stworzyć rozwiązanie oparte na interfejsie USB po stronie komputera)
* Usprawnienie na podstawie analizowania pliku log.txt aplikacji pod względem ergonomii i wygody
* Testowanie układu w różnych warunkach

1. Wykrywanie zużycia energii oraz oprogramowanie interfejsu USB są traktowane, jako zadania dodatkowe. Wykonanie któregokolwiek z tych zadań skutkuje podwyższeniem oceny. [↑](#footnote-ref-2)