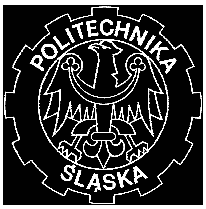


POLITECHNIKA ŚLĄSKA

WYDZIAŁ AUTOMATYKI, ELEKTRONIKI I INFORMATYKI

Praca dyplomowa magisterska

Pomiary potencjometryczne wieloskładnikowe z użyciem dedykowanego jonometru

Autor: Rafał Prusaczyk

Kierujący pracą: dr inż. Józef Wiora

Gliwice, listopad 2010

Spis treści

Spis tablic

Spis rysunków

Rozdział 1
Wprowadzenie

1.1. Wstęp

Zależność jaka występuje pomiędzy aktywnością oznaczanego jonu w roztworze a potencjałem odpowiedniego elektrochemicznego półogniwa jest wykorzystywana w potencjometrii. Metoda ta pozwala na zbadanie składu chemicznego roztworu. Rolę półogniwa najczęściej pełni elektroda jonoselektywna przy której występuje różnica potencjału na granicy faz materiał elektrodowy/elektrolit [**6**]. W celu wykonania pomiaru, oprócz elektrody pomiarowej, potrzebna jest także odpowiednia elektroda porównawcza. Do pomiaru występującej między elektrodami siły elektromotorycznej (SEM) wykorzystuje się jonometr, który w praktyce jest miliwoltomierzem wyskalowanym w odpowiednich jednostkach aktywności/stężenia poszczególnych jonów i posiadający bardzo dużą impedancję wejściową. Układ pomiarowy pokazany jest na rysunku 1.1.

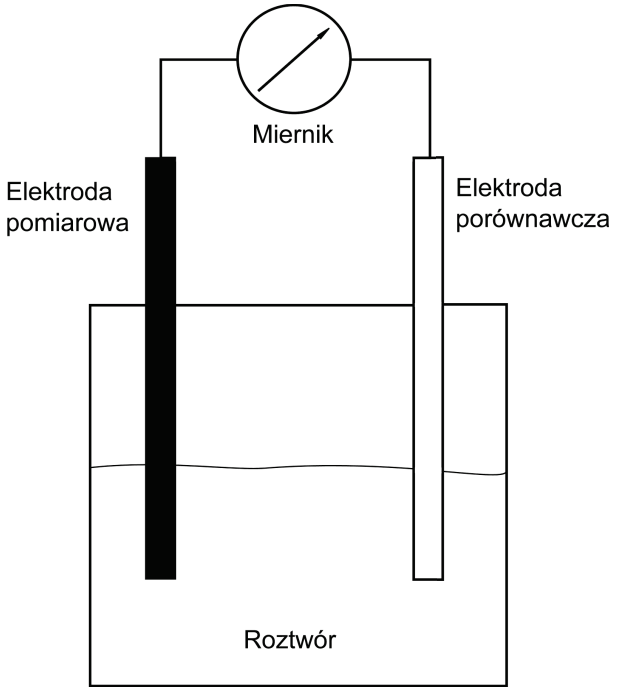
Różnica potencjałów między roztworem a elektrodą opisana jest wzorem Nernsta:

$$E = E_0 + \frac{\mathrm{R}T}{z\mathrm{F}} \ln\left(a_x\right) \tag{1}$$

gdzie:

a_x – aktywność danego jonu (kationu lub anionu) w roztworze,
z – wartościowość jonu oznaczanego,
E_0 – potencjał standardowy elektrody,
T – temperatura bezwzględna K,
$\mathrm{R} = 8{,}3143\,\mathrm{J\,K^{-1}mol^{-1}}$ – uniwersalna stała gazowa,
$\mathrm{F} = 0{,}6484\,\mathrm{C/mol}$ – stała Faraday'a.

Równanie Nernsta opisuje potencjał elektrody, która styka się z roztworem zawierającym tylko jeden rodzaj jonów potencjałotwórczych lub elektrody czułej tylko na jeden rodzaj jonów. W rzeczywistych warunkach analitycznych takie sytuacje nie występują.



RYSUNEK 1.1. Schemat układu pomiarowego.

Na potencjał elektrody najczęściej bezpośredni wpływ ma wiele różnych rodzajów jonów. Przy przeprowadzaniu badań analitycznych z użyciem elektrody jonoselektywnej niezbędne jest aby udział jonów przeszkadzających w mierzonym potencjale był statystycznie niezauważalny, czyli mniejszy od wartości przypisywanej elektrodzie jako odtwarzalność lub stabilność potencjału. Dla prowadzących badania nad nowymi konstrukcjami elektrod jonoselektywnych ważna jest możliwość porównywania właściwości różnych elektrod [**10**].

Aby mieć możliwość oszacowania zmiany potencjału elektrody, której przyczyną może być obecność w roztworze innych jonów niż jon oznaczany, niezbędne jest sformułowanie równania opisującego potencjał elektrody w roztworach zawierających różne rodzaje jonów potencjałotwórczych. Na podstawie wielu przeprowadzonych badań sformułowano ogólne równanie Nikolskiego-Eisenmana opisujące potencjał elektrody w roztworze zawierającym N rodzajów jonów:

$$E = E_0 + \frac{\mathrm{R}T}{z\mathrm{F}} \ln \left[a_i + \sum_{j \neq i}^{N} K_{i,j} \cdot (a_j)^{\frac{z_i}{z_j}} \right] \tag{2}$$

gdzie:

a_i, a_j – aktywność jonów głównego i oraz zakłócającego j w roztworze,

z_i, z_j – wartościowość jonów głównego i oraz zakłócającego j,

$K_{i,j}$ – współczynnik selektywności.

Powyższe równanie, jak każdy model matematyczny, zawiera uproszczenia, które są przyczyną niepełnej zgodności z właściwościami rzeczywistych elektrod jonoselektywnych. Pomimo tego równanie Nikolskiego-Eisenmana jest powszechnie uznane i używane do opisu właściwości elektrod jonoselektywnych [**10**].

Pomiar SEM za pomocą elektrody jonoselektywnej wymaga bardzo małej energii zewnętrznej. Z racji iż elektrody jonoselektywne są stosunkowo niewielkich rozmiarów, posiadane zasilanego bateryjnie jonometru pozwala również na łatwe wykonanie pomiarów w terenie. W takim zastosowaniu dokładność względna w przedziale 1–10% jest wystarczająca [**4**]. Ważniejsza od wysokiej dokładności pomiaru jest możliwość przeprowadzenia szybkich analiz wstępnych. Ogromne możliwości pomiarowe czujników chemicznych sprawiły iż są one stosowane w wielu gałęziach przemysłu. Obecnie elektrody jonoselektywne można zastosować w następujących dziedzinach:

- **Przemysł spożywczy** (pomiary zawartości azotanów przy przetwórstwie mięsa i ryb; pomiary zawartości chlorków w serach, maśle; oznaczanie zawartości potasu, sodu, węglanów, fluorków i/lub bromków w napojach alkoholowych);
- **Rolnictwo** (pomiary zawartości wapnia, chlorowców, sodu, potasu, azotanów w celu zapewnienia odpowiedniej jakości produkowanych pasz; onaczanie zawartości wapnia, azotanów, sodu, potasu, boru i chlorowców w glebie; oznaczenie azotu i azotanów oraz oznaczenie potasu i wapnia nawozów sztucznych);
- **Medycyna** (pomiar zawartości potasu, wapnia, chlorków i fluorków w próbkach krwi; oznaczanie zawartości jodków, fluorków, wapnia i jonów amonowych; analiza śliny, potu, kultur bakterii i próbek biologicznych);
- **Systemy wodne** (oznaczenie zawartości wapnia, potasu, sodu, srebra, ołowiu, kadmu, chlorowców, amonu oraz jonów siarczkowych i węglanowych wody naturalnej; mierzenie zawartości fluorków i azotanów w wodzie pitnej; oznaczenie zawartości jonów chlorowców, azotanów, potasu i sodu wody morskiej; pomiary zawartości miedzi, srebra, cyjanków i amonu w ściekach);
- **Produkty lecznicze** (oznaczenie zawartości fluorków w witaminach i pastach do zębów; pomiar zawartości chlorowców, miedzi, azotanów, wapnia w lekach);
- Dodatkowo elektrody jonoselektywne znajdują zastosowanie w geologii i górnictwie, metalurgii, przemyśle papierniczym oraz w wielu badaniach.

1.2. Cel i zakres pracy

Celem pracy jest wykonanie układu mikroprocesorowego pełniącego rolę jonometru, który byłby tańszy od rozwiązań komercyjnych. Pożądana funkcjonalność urządzenia:

- dokładność pomiaru na poziomie ± 0,1 mV;
- przenośność;
- możliwość zapisu danych na kartę SD oraz odczytu ich bezpośrednio w środowisku Windows;
- komunikacja z komputerem PC za pomocą portu USB;
- możliwość prezentacji wyników na ekranie LCD;
- możliwość równoczesnego wykonywania pomiaru kilku potencjałów;

Zakres pracy obejmuje:

- dobór podzespołów elektronicznych,
- zaprojektowanie schematu układu,
- zaprogramowanie urządzenia,
- przetestowanie poprawności pomiaru jonometrem.

1.3. Przegląd rozwiązań komercyjnych

Podstawową kwestią przed rozpoczęciem budowania układu jest rozeznanie się w ofertach producentów podobnych urządzeń dostępnych już na rynku. Gdyby okazało się, że budowane urządzenie jest równie funkcjonalne jak te oferowane sprzedaży detalicznej a jego budowa pochłonęłaby większe środki finansowe to cały projekt nie miałby sensu. Porównaniem objęto urządzenia trzech różnych firm.

Handylab12. Zaletą oferowanego przez firmę *SI Analytics GMBH* urządzenia jest pojemna pamięć pozwalająca zapisać 800 zestawów danych w ustalonych odstępach czasu. Wymiana baterii nie powoduje utraty zapisanych danych. Poza pomiarem napięcia możliwy jest również pomiar temperatury. Posiada odporną na wstrząsy obudowę.

pH Mobile Irda. Urządzenie firmy *Metrohm AG* posiada możliwość bezprzedowej komunikacji z komputerem PC lub drukarką. Wodoszczelna obudowa umożliwia pomiary w trudnych warunkach. Pamięć pozwala na zapisanie 200 wyników.

SensoDirect pH200. Przyrząd pomiarowy firmy *Lovibond Tintometer* umożliwia wykonywanie szybkich i dokładnych pomiarów. Niestety nie posiada dużej pamięci wewnętrznej co ogranicza jego funkcjonalność w przypadku potrzeby wykonania większej ilości pomiarów. Posiada antywstrząsową, wodoszczelną obudowę.

Podsumowanie. Zestawienie najważniejszych parametrów zawarto w tabeli 1.1. Urządzenia do porównania wybrano przy założeniu, że muszą być przenośne oraz posiadać możliwość zapisu danych do pamięci wewnętrznej. Analizując oferty trzech różnych firm można zauważyć, że urządzenia o parametrach przybliżonych do wykonywanego układu w każdym przypadku przekraczają cenę 2000zł. Żaden ze oferowanych przez rynek jonometrów nie posiada takich możliwości jakie postawiono przed budowanym układem. Główną zaletą prototypu jest możliwość jednoczesnego pomiaru kilku potencjałów, której nie posiada żadne z urządzeń komercyjnych poddanych porównaniu. Zakładana dokładność budowanego jonometru jest taka sama jak urządzenia *SensoDirect pH200*, jednak pod względem pojemności pamięci wewnętrznej, która dzięki wymiennej karcie SD jest praktycznie nieograniczona oraz możliwości podłączenia do komputera, budowany układ wydaje się być znacznie lepszy od rozwiązań komercyjnych.

TABLICA 1.1. Porównanie urządzeń dostępnych na rynku.

Parametr	**Handylab12**	**pH Mobile IrDa**	**SensoDirect pH200**
Dokładność	±1mV	±0,2mV	±0,1mV
Zakres	±2V	±1,2V	±2V
Pamięć	800 wyników	200 wyników	20 wyników
Interfejs	RS-232	IrDa	brak
Cena	2184zł	2730zł	2076zł

Rozdział 2

Projekt układu elektronicznego

Rynek oferuje szeroką gamę układów elektronicznych. W zależności od potrzeb można bardzo szczegółowo wybrać potrzebny produkt. Niestety projektując układ nie zawsze można dokładnie sprecyzować wartość parametru poszczególnego elementu, która idealnie spełni nasze oczekiwania. Przykładem może być łatwość doboru przetwornika analogowo-cyfrowego, przy którym wybór rozdzielczości, spełniającej wymagania dokładności, można wyliczyć przed wykonaniem układu. Trudniejszą sprawą jest wybór mikrokontrolera, którego parametry należy dobrać z „zapasem", gdyż dopiero po napisaniu oprogramowania znana będzie np. ilość pamięci zajmowanej przez program. Oczywistym staje się wiec to, że dopiero po wykonaniu prototypu elementy wchodzące w skład gotowego urządzenia mogą być dobrane znacznie lepiej.

By uniknąć późniejszych kłopotów z programowaniem założono, że jednostka centralna będzie komunikowała się z przetwornikiem A/C oraz kartą SD za pomocą interfejsu SPI[1]. Pozwola to na ograniczenie ilości połączeń mikrokonkrolera z tymi elementami. Wolniejsze działanie tego interfejsu w porównaniu z komunikacją równoległą nie ma wpływu na działanie układu, ponieważ wykonywane urządzenie przeznaczone jest do wykonywania pomiarów statycznych.

Z racji tego, że jonometr ma być przenośny, przy selekcji wymaganych komponentów układu należy wybrać te, które pobierają relatywnie mało energii elektrycznej. Napięcie zasilania powinno mieścić się w zakresie 4–6V . Wstępnie przyjęto, że układ będzie zasilany za pomocą 3 baterii typu AA (1,5V) połączonych szeregowo (4,5V) lub za pomocą portu USB.

Duża powszechność układów komunikujących się z komputerem PC za pomocą standardu RS-232 sprawiła, że w celu komunikacji z komputerem za pomocą portu USB zastosowany zostanie konwerter USB -> UART. Pozwala to wykorzystać powszechnie dostępny port USB w ten sam sposób jak rzadziej dziś dostępny w komputerach port COM z większą szybkością przesyłu danych.

---

[1] SPI (ang. Serial Peripheral Interface) – szeregowy interfejs urządzeń peryferyjnych.

Cena budowanego jonometru także nie jest bez znaczenia. Projektując układ należy zwrócić uwagę na kwestię kosztów związanych z zakupem podzespołów. Niestety ograniczona dostępność i konieczność zamówień dużych ilości jednakowych elementów wymusza zastosowanie w prototypie części droższych ale dostępnych w handlu detalicznym. Wnioski dotyczące możliwości zastosowań tańszych układów scalonych znajdują się w podsumowaniu pracy (rozdział 6.2). Zakładana przez projekt możliwość produkcji urządzenia na szerszą skalę, zwiększa ilość dostępnych ofert producentów, gdyż wiele elementów elektronicznych sprzedawana jest tylko w ilościach hurtowych.

2.1. Płyta główna układu

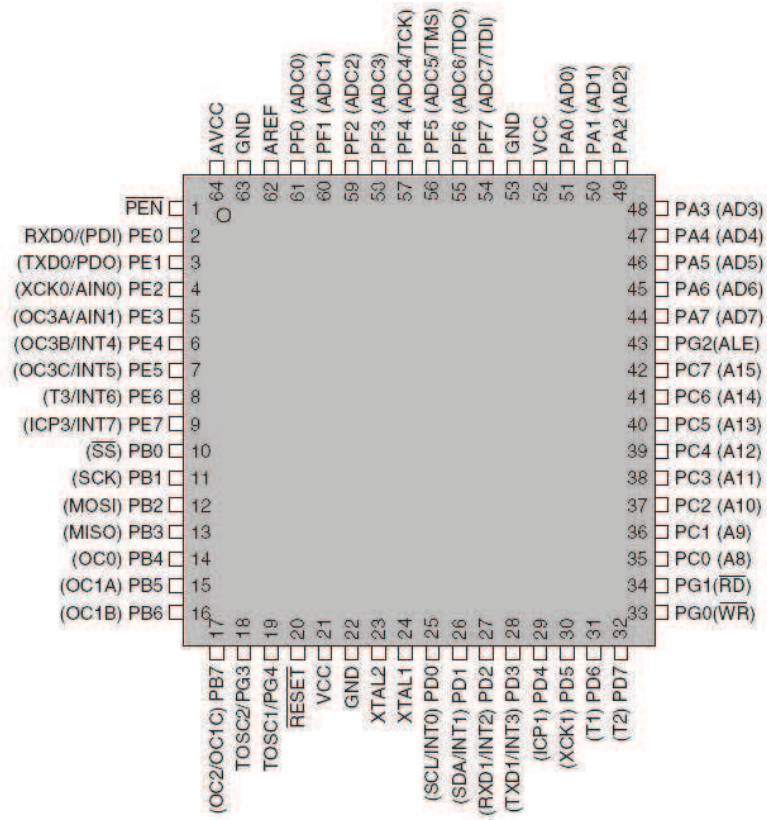
2.1.1. Mikrokontroler

Główny element układu, który łączy wszystkie podzespoły tworzące razem gotowe urządzenie, należało wybrać z wielką starannością. Założono iż jednostką centralną będzie mikrokontroler (MCU) z rodziny AVR, z racji dużej dostępności tych układów oraz prostoty ich programowania. Poszczególne systemy mikroprocesorowe firmy Atmel, różnią się miedzy sobą przede wszystkim ilością pamięci flash, pamięci RAM, ilością portów wejść/wyjść. Kolejne różnice stanowi dodatkowe wyposażenie, tj. interfejs SPI, wbudowany przetwornik A/C, częstotliwość taktowania procesora, ilość zużywanej energii elektrycznej. Mając na uwadze początkowe założenia projektu wybrano mikrokontroler ATmega128L. Ten układ z racji swojej szerokiej funkcjonalności i wydajności, jest bardzo popularny wśród projektantów układów elektronicznych.

Możliwości jakie daje układ ATmega128 są bardzo duże. Wybrano ten typ mikrokontrolera gdyż posiada on:

- 128k pamięci programu flash,
- szeregowy interfejs SPI,
- programowalny USART[1],
- wystarczającą liczbę portów wejść/wyjść.

[1] USART (ang. Universal Synchronous and Asynchronous Receiver and Transmitter) – uniwersalny synchroniczny i asynchroniczny odbiornik i nadajnik do szeregowej transmisji danych.



Rysunek 2.1. Konfiguracja wyprowadzeń ATmega128L.

ATmega128 występuje w dwóch typach. Mikokontroler oznaczony na końcu literą „L" różni się od tego bez litery niższą częstotliwością taktowania zegara (8MHz zamiast 16MHz,) a co za tym idzie mniejszym poborem energii elektrycznej [**2**]. Szybkość mikroprocesora nie ma wpływu na funkcjonalność układu, więc uboższa wersja ATmega128L jest w pełni wystarczająca i lepiej spełnia cele początkowe projektu. Możliwą konfigurację wyprowadzeń ATmega128L przedstawia rysunek 2.1. Układ ATmega128L wyposażony jest w 64 wyprowadzenia. Końcówki oznaczone symbolami PA1-PG4 mogą być portami wejścia/wyjścia lub pełnić specjalne funkcje (na rysunku 2.1 w nawiasach, zaznaczone skrótami). Specjalne funkcje mikrokontrolera wykorzystane w urządzeniu realizowane są poprzez wyprowadzenia SCK, MOSI, MISO (PORTB) - które po aktywacji interfejsu SPI, wykorzystano do komunikacji z przetwornikiem A/C oraz z kartą pamięci SD. Wyprowadzenia TXD1 oraz RXD1 (PORTD) z włączonym USART1 zapewniają komunikację z PC. PDO oraz PDI (PORTE) służą do kasowania i zapisywania pamięci flash za pomocą programatora ISP. TDI, TDO, TMS i TCK (PORTF) umożliwiają programowanie MCU

programatorem JTAG. Pozostałe wyprowadzenia służą jako porty wejścia/wyjścia. Dokładniejszy opis wyprowadzeń oraz rejestrów mikrokontrolera ATmega128L znajduje się w rozdziale 3.2.

2.1.2. Przetwornik analogowo-cyfrowy

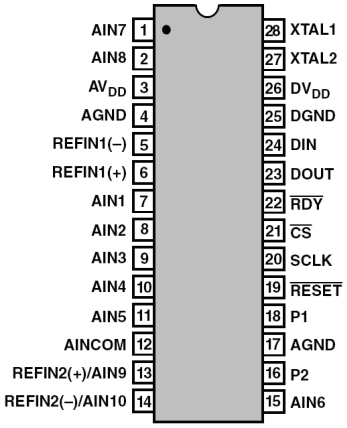
Układy cyfrowe, do jakich należą mikrokontrolery, operują na dyskretnym i skończonym zbiorze stanów wejściowych i wyjściowych. Jako że wartości w świecie rzeczywistym są analogowe, tj. ciągłe, zachodzi konieczność jak najdokładniejszego odwzorowania wielkości analogowej na wartość cyfrową. W tym celu wykorzystywany jest przetwornik analogowo-cyfrowy (ADC) [**11**].

Z punktu widzenia dokładności budowanego urządzenia, bardzo ważną kwestią jest odpowiedni dobór ADC. Pomimo iż mikrokontroler ATmega128L posiada wbudowany 10-bitowy przetwornik, nie jest on wystarczający. Znając zakres SEM wytwarzanej pomiędzy elektrodą pomiarową a elektrodą odniesienia, który dla budowanego jonometru wynosi $\pm$1V, można obliczyć, że rozdzielczość napięciowa będzie większa od zakładanej w celach pracy niepewności pomiaru.

Przykład dla przetwornika 10 bitowego:
- pełna skala pomiaru wynosi 2V;
- rozdzielczość przetwornika jest równa 10 bitów, czyli $2^{10} = 1024$ poziomy kwantyzacji;
- rozdzielczość napięciowa wynosi: $2V/1024 = 0{,}00195\text{V} = 1{,}95\text{mV}$.

Wybór przetwornika padł na 16-bitowy układ AD7708, firmy Analog Devices. Czynnikiem skłaniającym do podjęcia takiej decyzji był fakt, iż ADC komunikuje się z mikrokontrolerem za pomocą interfejsu SPI, pobiera mało energii elektrycznej, posiada 8 kanałów wejściowych, które umożliwiają wykonanie jednocześnie wieloskładnikowych pomiarów potencjometrycznych [**1**]. Zakres pomiarowy układu jest programowalny. Ze względu na wystarczają rozdzielczość programowo został ustalony na $\pm 1{,}28V$.

Przykład dla 16 bitowego przetwornika AD7708:
- pełna skala pomiaru wynosi 2,56V;
- rozdzielczość przetwornika jest równa 16 bitów, czyli $2^{16} = 65536$ poziomy kwantyzacji;
- rozdzielczość napięciowa wynosi: $2{,}56\text{V}/65536 = 0{,}000039\text{V} = 0{,}039\text{mV}$.



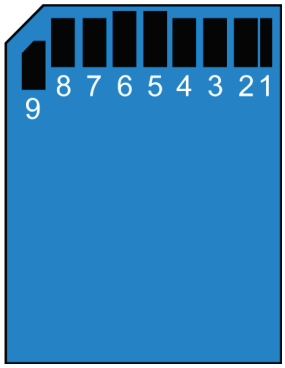
Rysunek 2.2. Konfiguracja wyprowadzeń AD7708.

Jak widać na przykładzie dla przetwornika AD7708, rozdzielczość napięciowa a co za tym idzie niepewność pomiaru z niej wynikająca jest wystarczająca.

Wyprowadzenia ADC przedstawione rysunku 2.2 zostały wykorzystane w następujących zastosowaniach:

- AIN1 – AIN8, AINCOM - wejścia napięciowe z układu przedwzmacniaczy;
- REFIN1(+), REFIN1(−) - wejścia napięcia referencyjnego;
- DIN, DOUT, SCLK, $\overline{\text{CS}}$ - komunikacja z MCU poprzez interfejs SPI;
- XTAL1, XTAL2 - wejścia rezonatora kwarcowego;
- AV_{DD}, DV_{DD}, AGND, DGND - napięcie zasilania dla części analogowej i cyfrowej oraz masa analogowa i cyfrowa.

Komunikacja pomiędzy mikrokontrolerem odbywa się na liniach DIN oraz DOUT. Przetwornik otrzymuje instrukcję na wejściu DIN jednocześnie odpowiadając MCU linią DOUT. Komunikacja w interfejsie SPI odbywa się synchronicznie za pomocą sygnału taktującego przesyłanego z mikrokontrolera do ADC na wejście SCLK. W przypadku większej liczby urządzeń peryferyjnych komunikujących się z MCU za pomocą interfejsu SPI, ważną rolę pełni wejście Chip Select ($\overline{\text{CS}}$). W przypadku gdy na lini tej znajduje się stan wysoki urządzenie nie reaguje na wysyłane instrukcje. Aby aktywować urządzenie na linię ($\overline{\text{CS}}$) należy podać sygnał w stanie niskim.



RYSUNEK 2.3. Konfiguracja wyprowadzeń karty SD w trybie SPI.

1–CS; 2–DI; 3–VSS1; 4–VDD; 5–SCLK; 6–VSS2; 7–DO; 8, 9 – nie używane w trybie SPI

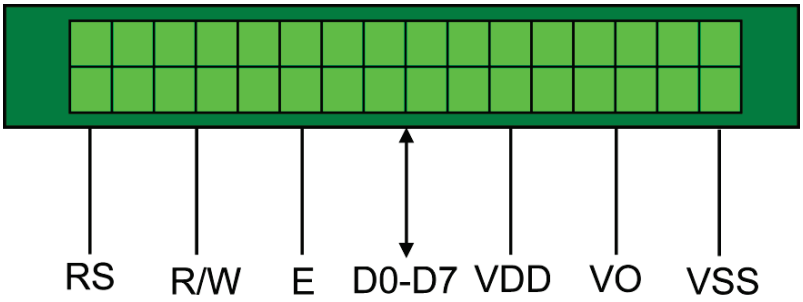
2.1.3. Karta SD

Planując rozbudowę MCU o obsługę kart pamięci przede wszystkim należy wyposażyć płytę główną układu o slot kart SD. Specjalna budowa takiego gniazda oraz specyficzne rozmieszczenie i zróżnicowana długość styków karty pamięci pozwala na wyjmowanie i wkładanie kart bez konieczności odłączania napięcia zasilania. Układ wyprowadzeń karty SD przedstawiono na rysunku 2.3. Karta pamięci posiada również przełącznik służący do zabezpieczenia karty przed zapisem. Należy jednak wiedzieć, że samo przełączenie przełącznika w pozycję zabraniającą zapisu na kartę nie spowoduje tego, że karta pamięci będzie chroniona przed zapisem. Slot kart SD musi być wyposażony w odpowiednie styki, których stan jest sprawdzany przez MCU. Opis poszczególnych styków karty SD przedstawiono w tabeli 2.1. Komunikacja podobnie jak w przypadku przetwornika analogowo-cyfrowego odbywa się na trzech liniach DO, DI oraz SCLK.

Komunikacja mikrokontrolera z kartą pamięci w trybie SPI umożliwia zastosowanie zarówno bardzo popularnych kart SD jak i mniej popularnych MMC. Z faktu tego wynika brak wykorzystania styków 8 i 9, które w karcie MMC nie występują.

2.1.4. Wyświetlacz LCD

Bardzo ważną funkcją jonometru jest możliwość obserwowania wyników w trybie *on-line*. Mając na uwadze zasilanie bateryjne wybór padł na wyświetlacz bez podświetlenia posiadający dwa wiersze po szesnaście znaków. Ze względu na prostotę programowania



Rysunek 2.4. Konfiguracja wyprowadzeń wyświetlacza LCD.

zdecydowano, że wyświetlacz LCD będzie posiadał sterownik zgodny z HD44780. Wyprowadzenia wyświetlacza przedstawiono na rysunku 2.4. Ze względu na dużą ilość wolnych wyjść w posiadanym mikrokontrolerze obsługę wyświetlacza przeprowadzono wszystkimi ośmioma liniami (D0–D7). Możliwe jest programowanie sterownika HD44780 za pomocą czterech linii (D0–D3), jednak w tym przypadku nie było to konieczne. Linia RS decyduje o tym czy do wyświetlacza przesłana zostanie komenda bądź znak (RS=0 – komenda, RS=1 – znak). R/W oznacza zapis bądź odczyt z pamięci wyświetlacza (R/W = 0 – zapis, R/W = 1 – odczyt). Linie VDD i VSS to odpowiednio zasilanie i masa układu. VO służy do regulacji kontrastu wyświetlanych na ekranie znaków.

2.1.5. Elementy dodatkowe

Całość urządzenia, poza układami opisanymi wcześniej składa się z ważnych elementów, które nie wymagają programowania ale z powodu początkowych założeń projektu istnieje konieczność ich zastosowania. Przykładem takiego elementu może być układ FT232 służący jako kowerter RS-232 na USB. Układ ten po podłączeniu z mikrokontrolerem

Tablica 2.1. Opis wyprowadzeń karty SD w trybie SPI.

Numer styku	Nazwa	Kierunek	Funkcja w trybie SPI
1	CS	Wejście	Chip Select – wybór karty
2	DI	Wejście	Data Input – dane z MCU do karty
3	VSS1	–	Ground – uziemienie
4	VDD	–	Supply Voltage – napięcie zasilania
5	SCLK	Wejście	SPI Clock – sygnał zegarowy trybu SPI
6	VSS2	–	Ground
7	DO	Wyjście	SPI Data output – dane z karty do MCU

umożliwia komunikację MCU z komputerem za pomocą gniazda USB. FT232 odbierany jest przez komputer jak port COM, jednak umożliwia pracę poprzez wiele powszechniejszy i szybszy w komunikacji port USB. Metoda połączeń FT232 z mikrokontrolerem jest opisana w dokumentacji tego układu. W celu dostarczenia napięcia referencyjnego dla przetwornika analogowo cyfrowego zastosowano układ ADR421 firmy Analog Devices. Źródło napięcia odniesienia zapewnia bardzo stabilne napięcie 2,5V wymagane przez ADC przy niskim poborze energii elektrycznej. Stabilność napięcia zasilania układu wynoszącego 3,3V zapewnia stabilizator napięcia LTC1627 firmy Linear Technology. Dodatkowym elementem, który okazał się konieczny do zastosowania jest pompa ładunkowa MCP1252 firmy Microchip. Zadaniem tego układu jest podwyższenie napięcia z 3,3V do wartości 5V. Zastosowanie MCP1252 zostało wymuszone przez fakt, że wszystkie dostępne w sprzedaży detalicznej wyświetlacze LCD wymagały aby wartość napięcia zasilania wynosiła 5V.

2.1.6. Schemat układu

Obecnie ogólnodostępne są programy wspomagające projektowanie płytek drukowanych. Wykonanie projektu płytki należy zacząć od utworzenia schematu połączeń elektrycznych układu. Program po przejściu w tryb PCB przekształca schemat elektryczny na widok rzeczywisty płytki z zachowaniem połączeń pomiędzy poszczególnymi wyprowadzeniami elementów układu. Kolejnym etapem jest ustawienie wielkości płytki, odpowiednie rozmieszczenie układów na płytce oraz poprowadzenie ścieżek łączących ze sobą odpowiednie komponenty. Schemat elektryczny wykonuje się zgodnie z instrukcjami zawartymi w dokumentacjach technicznych. Producenci oprócz szerokiego opisu układów scalonych zamieszczają także przykłady połączeń.

Schemat elektryczny układu. Schemat elektryczny całego układu znajduje się w dodatkach pracy (Dodatek B). Podzielony jest na trzy działy - zasilanie, część główna oraz porty komunikacyjne.

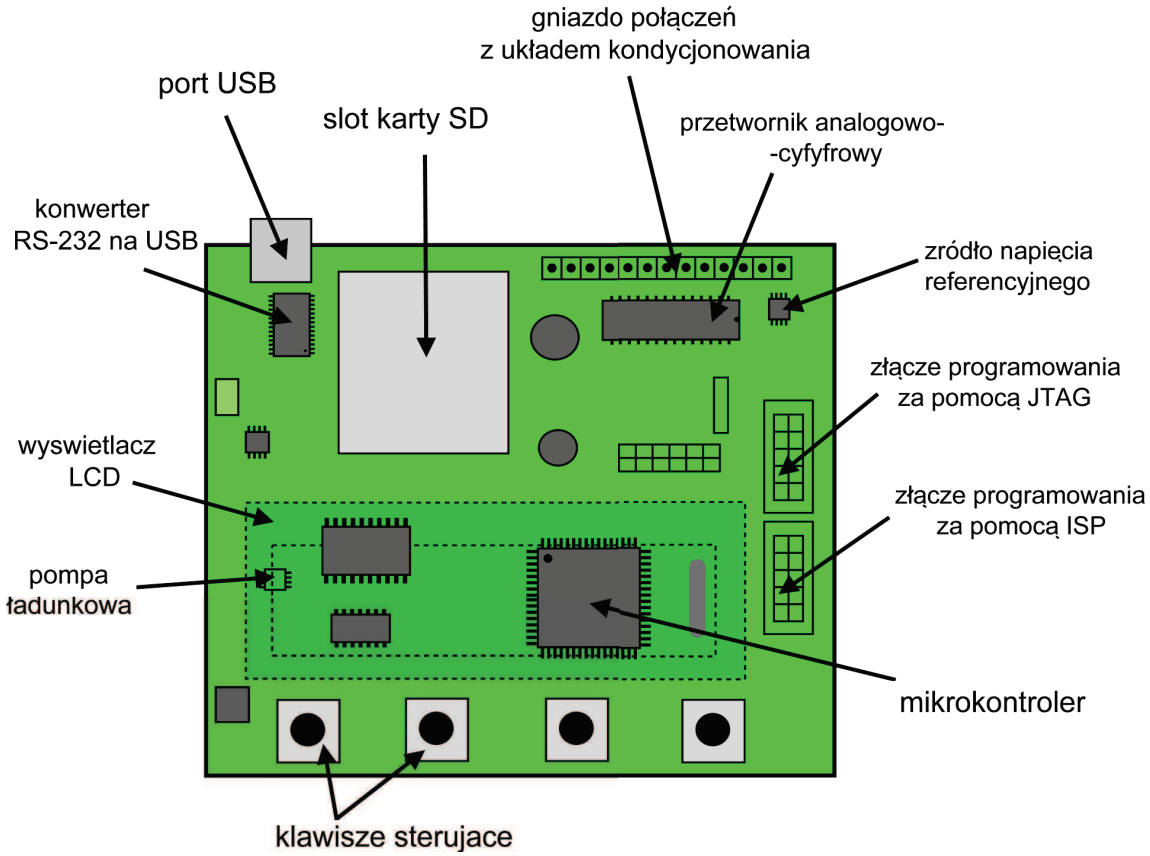
Stopień wejściowy części zasilającej umożliwia podłączenie napięcia z baterii (4–6V) lub USB (5V). Układ LTC1627 utrzymuje stałe napięcie 3,3V dla wszystkich bloków systemu przy zmieniającym się napięciu zasilania. MCP1252/3 ma za zadanie podwyższenie napięcia dla wyświetlacza LCD, przetwornika analogowo-cyfrowego oraz układu kondycjonowania sygnału. Obwody cyfrowe wytwarzają wewnątrz i na zewnątrz układu ciągłe, losowe zakłócenia elektromagnetyczne, które mogą mieć wpływ na dokładność pomiaru [**9**].

Aby zminimalizować wpływ tych zakłóceń na sygnał analogowy, dla napięcia zasilania układu kondycjonowania i przetwornika analogowo-cyfrowego zastosowano filtr dolnoprzepustowy o częstotliwości odcięcia wynoszącej 106Hz. Dodatkowo rozdzielono masę części analogowej z masą części cyfrowej co także ogranicza wpływ zakłóceń na wartość mierzonego napięcia.

Część główna schematu elektrycznego zawiera połączenia pomiędzy najważniejszymi elementami całego układu. Sercem układu jest mikrokontroler ATmega128L, który umożliwia odbieranie danych z ADC, zapis i odczyt danych z karty SD, odbieranie i wysyłanie danych z portu USB, obsługę przycisków oraz pisanie do LCD. W celu oszczędzania baterii układ FT232RL zasilany jest z portu USB. Aby uniknąć pracy ze zbyt niskim napięciem zasilania (co mogłoby mieć miejsce gdyby stan naładowania baterii był zbyt niski) zastosowano układ resetowania procesora DS1818-10. W przypadku gdy napięcie spadnie poniżej 2,8V układ ten wysyła do mikrokontrolera sygnał reset.

Porty komunikacyjne przedstawione są na trzeciej części schematu elektrycznego (Dodatek B). Układ wyposażony został w dwa gniazda służące do programowania. Umożliwia to programowanie mikrokontrolera w trybie JTAG oraz ISP. W układzie dodano złącze uniwersalne mające służyć do debugowania kodu programu w trakcje jego pisania. Elementami znajdującymi się w tej części schematu są także układy 74HCT125 oraz 74HC244M. Zadaniem ich jest podwyższenie napięcia sterującego wyświetlaczem do poziomu 5V. Dodatkowo na schemacie znajduje się gniazdo USB, slot kart pamięci oraz złącze układu kondycjonowania.

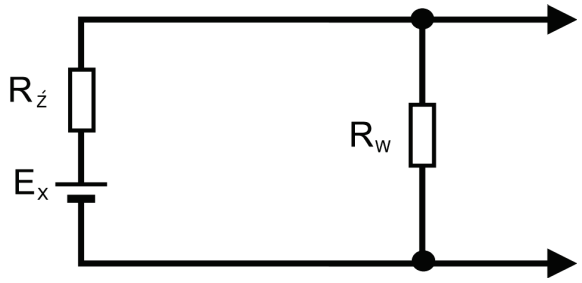
Schemat płytki PCB. Układ zmontowany jest na dwuwarstwowej płytce drukowanej. Szczegółowe rozmieszczenie elementów znajduje się w dodatkach pracy (Dodatek B). Uproszczony widok zawierający najważniejsze elementy układu przedstawiono na rysunku 2.5. Rozłożenie elementów na płytce umożliwia łatwe wkładanie i wyjmowanie karty SD oraz swobodny dostęp do gniazda USB. Klawisze sterujące umieszczono bezpośrednio obok wyświetlacza LCD w linii prostej by zminimalizować rozmiary urządzenia.



Rysunek 2.5. Widok płytki jonometru.

2.2. Tor analogowy układu

W zbudowanym jonometrze mierzony sygnał analogowy jest przetwarzany na postać cyfrową a następnie poddawany obróbce za pomocą mikrokontrolera. Konwersję z postaci analogowej na postać cyfrową wykonuje przetwornik analogowo-cyfrowy. Specyfika pomiarów potencjometrycznych wymaga jednak zastosowania układu kondycjonowania sygnału w celu przeprowadzenia możliwie bezprądowego pomiaru SEM między elektrodami [4]. Aby wykorzystać w pełni dokładność wykonanego urządzenia błąd wynikający z metody pomiaru powinien być mniejszy od błędu urządzenia pomiarowego. Na rysunku przedstawiono schemat zastępczy wejścia wzmacniacza. Z racji tego, iż elektrody jonoselektywne posiadają wysoką rezystancję, impedancja wejściowa układu kondycjonowania powinna być odpowiednio wysoka. Budowany jonometr ma wykorzystać gotowy układ kondycjonowania oparty na wzmacniaczach pomiarowych AD8231, którego schemat ideowy znajduje się na rysunku 2.7. Błąd pomiaru powodowany błędnym dobraniem impedancji wejściowej

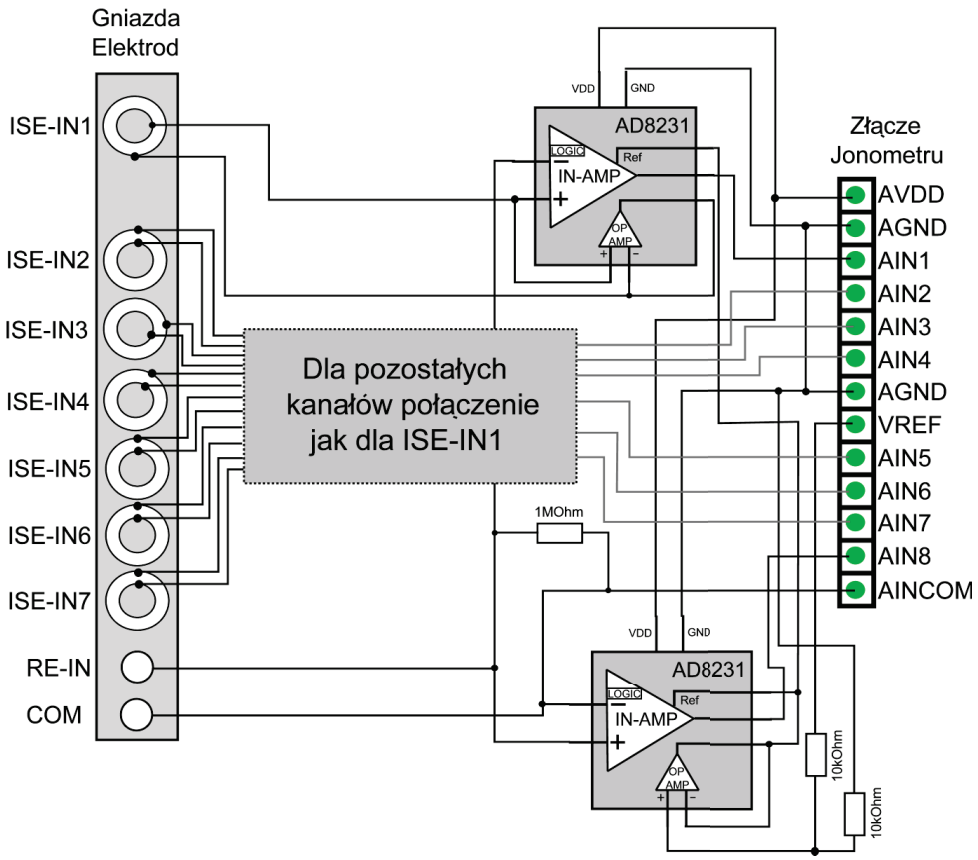


RYSUNEK 2.6. Elektryczny schemat zastępczy wejścia wzmacniacza.

E_X – ogniwo pomiarowe; R – rezystancja źródła; R_W – impedancja wejściowa wzmacniacza

wzmacniaczy można wyznaczyć za pomocą wzoru $\%\Delta = \frac{R}{R+R_W} \cdot 100$. Jeżeli SEM występująca pomiędzy elektrodami znajduje się w zakresie $\pm$1V, łatwo obliczyć, że aby błąd był mniejszy od $\pm$0,1mV impedancja wejściowa wzmacniaczy musi być 10^4 razy większa od rezystancji elektrod.

Znając impedancję wejściową wzmacniaczy układu kondycjonowania, która wynosi 10GΩ, należy mieć na uwadze fakt, iż chcąc wykonywać pomiar w pełni wykorzystujący dokładność budowanego jonometru, należy stosować elektrody o rezystancji nie przekraczającej 1MΩ. W celu zmierzenia SEM na elektrodach o większej rezystancji należy zastosować układ wzmacniaczy o odpowiednio wyższej impedancji wejściowej.



RYSUNEK 2.7. Schemat układu kondycjonowania sygnału.

Rozdział 3
Oprogramowanie

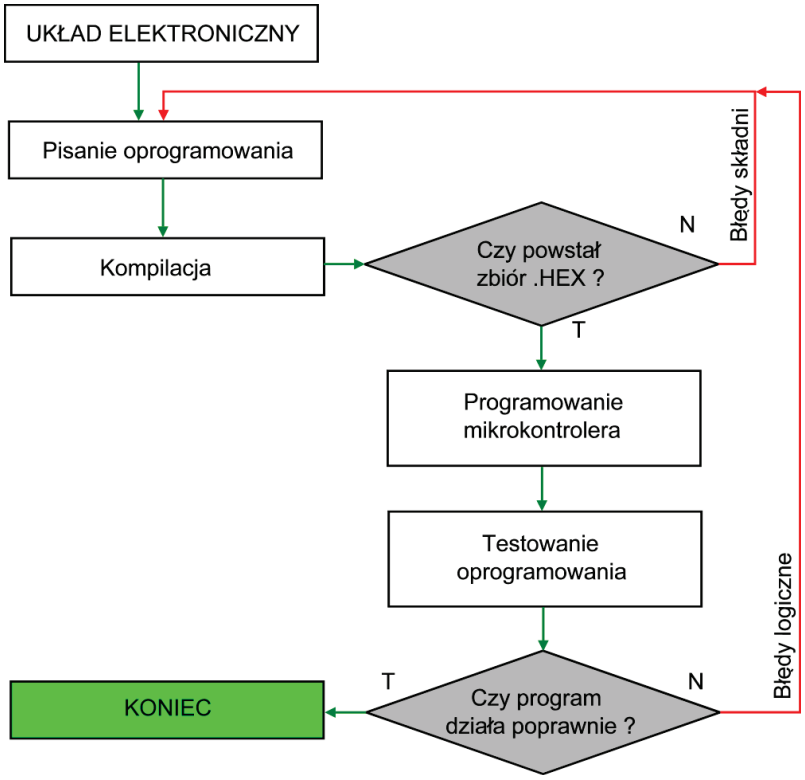
3.1. Metoda programowania

Mikrokontrolery AVR, do których należy zastosowany MCU ATmega128L, opierają się na architekturze harwardzkiej, której główną cechą jest rozdzielenie przestrzeni adresowej pamięci programu i przestrzeni adresowej pamięci danych. AVR-y należą do grupy układów RISC, w której większość rozkazów realizowana jest w jednym takcie zegara co zapewnia szybsze wykonywanie programu. Kolejną zaletą mikrokontrolerów AVR jest również to, że zaimplementowano w nich wiele rejestrów wewnętrznych, mogących pełnić funkcję akumulatora podczas wykonywania operacji arytmetycznych i logicznych.

Osoby programujące mikrokontrolery podzieliły się na zwolenników programowania w asemblerze i językach wysokiego poziomu, np. C lub Bascom. Programowanie w językach wysokiego poziomu posiada wiele zalet i udogodnień, jednak niektórych problemów nie da się rozwiązać bez zastosowania wstawek asemblerowych [5].

Program dla układu jonometru napisano w języku C/C++, wykorzystując darmowy pakiet programów WinAVR zawierającego kompilator AVR–GCC. Programowanie mikrokontrolera odbywało się poprzez interfejs JTAG za pomocą programatora JTAGcable II. Oprócz znajomości poleceń języka programowania, niezbędne jest posiadanie dokumentacji wszystkich programowalnych układów wchodzących w skład zbudowanego urządzenia. Dokumentacje elementów elektronicznych dostarczają wiele informacji dla programisty, niejednokrotnie zawierającymi krótkie przykłady gotowego kodu programu. Metoda tworzenia oprogramowania dla zbudowanego układu elektronicznego została przedstawiona w postaci diagramu na rysunku 3.1.

Posiadając odpowiednio dobrany mikrokontroler umożliwiający zrealizowanie procedur obsługi, których wymaga układ jonometru, można zająć się tworzeniem programu dla urządzenia. Opracowanie oprogramowania składa się z części edycyjnej, w której koduje się



RYSUNEK 3.1. Etapy tworzenia oprogramowania.

algorytmy, kompilacji umożliwiającej umożliwiającej znalezienie błędów składni oraz testów oprogramowania, umożliwiających znalezienie błędów logicznych w algorytmach [**7**]. Znalezienie błędu wymaga dokonania ich korekty i powtórnej kompilacji programu.

3.2. Program główny

Program główny zbudowanego jonometru scala kilka podprogramów, zawierających procedury i funkcje obsługi układów peryferyjnych. Poprawia to czytelność programu i ułatwia jego modyfikacje. Program znajduje się w pamięci mikrokontrolera. Z programowego punktu widzenia wyprowadzenia portów mikrokontrolera mogą być zastosowane jako wejścia/wyjścia lub pełnić funkcje specjalne. ATmega128L posiada sześć portów (PORTA-PORTF) z ośmioma wyprowadzeniami oraz jeden port z czterema wyprowadzeniami (PORTG). Dla każdego portu przyporządkowane są pamięci adresowe – rejestr danych (PORTx), rejestr kierunku danych (DDRx) oraz rejestr odczytu stanów wejściowych

(PINx). W rejestrze DDxn można zdefiniować, czy dane wyprowadzenie będzie wykorzystane jako wejście bądź wyjście. Logiczna jedynka oznacza, że dany port jest wyjściem, logiczne zero, że port jest wejściem. W przypadku gdy PORTxn jest skonfigurowany jako wejście, wpisanie do niego logicznej jedynki powoduje, podłączenie do wejścia napięcia zasilania (*pull–up*). Gdy PORTxn ustawiony jest jako wejście, zapisanie do niego logicznej jedynki, wymusza stan wysoki na wyjściu wyprowadzenia, a zapisanie logicznego zera, stan niski. Rejestr wejść PINxn przeznaczony jest tylko do odczytu, w przypadku konfiguracji PORTxn jako wejście, stan wysoki zapisywany jest jako logiczna 1 a stan niski jako logiczne 0. Konfiguracja portów wejść/wyjść mikrokontrolera ATmega128 przedstawiona jest w tabeli 3.1. Funkcje specjalne mikrokontrolera takie jak USART, SPI, przetwornik analogowo-cyfrowy, należy aktywować w odpowiednich rejestrach MCU. Po ich aktywacji na liniach, które są przez nie wykorzystywane, nie jest możliwe korzystanie z nich jako normalne porty wejścia/wyjścia.

Poniżej znajduje się opis powiązań programowych między MCU a poszczególnymi elementami całego układu. Poza funkcjami i procedurami, utworzono dodatkowo pliki z rozszerzeniem `.h`, w których zawarto definicje portów stałych lub komend ułatwiające sprawdzanie bądź korygowanie gotowego programu. Pełny kod programu dla jonometru znajduje się w dodatku A.

Obsługa LCD. Komunikacja mikrokontrolera z wyświetlaczem LCD odbywa się za pomocą 8 bitowej szyny danych (na liniach LCD_DB0–LCD_DB7) i trzech sygnałów sterujących E, R/W i RS. Każda instrukcja wysłana do wyświetlacza ma określony czas przetwarzania. Nadesłanie kolejnej instrukcji, przed końcem wykonywania poprzedniej może spowodować nieprawidłowe działanie wyświetlacza. Aby uniknąć tego problemu można programowo ustalić opóźnienie potrzebne na wykonanie danej komendy lub sprawdzać

TABLICA 3.1. Konfiguracja portów We/Wy ATmega128.

DDxn	PORTxn	Kierunek	Pull-up
0	0	Wejście	Nie
0	1	Wejście	Tak
0	1	Wejście	Nie
1	0	Wyjście	Nie
1	1	Wyjście	Nie

Rysunek 3.2. Drgania na stykach przycisku.

flagę zajętości (BF), która w czasie wykonywania instrukcji przechodzi w stan wysoki. Dzięki takiemu rozwiązaniu obsługa wyświetlacza jest wydajniejsza.

Procedury obsługujące wyświetlacz:

`void WriteToLCD (unsigned char v,unsigned char rs)` – Wysyła daną bądź instrukcję do wyświetlacza LCD. Włącza bufory podwyższające napięcie sterujące wyświetlaczem i sprawdza flagę zajętości.

`void lcd_init(void)` – Inicjalizuje wyświetlacz. Realizuje włączenie pompy ładunkowej, napięcia dla wyświetlacza LCD, ustawia tryb 8 bitowy i włącza wyświetlacz.

`void lcd_puts(char *str)` – Wysyła ciąg znaków do wyświetlacza LCD. Wpisany ciąg znaków pojawia się na wyświetlaczu, wykorzystuje procedurę `WriteToLCD`.

Obsługa klawiszy. Przyciski służące do obsługi urządzenia podłączono do czterech portów wejściowych mikrokontrolera. Zwolniony przycisk jest w stanie wysokim, przyciśnięcie powoduje zwarcie wejścia z masą. W programie dodatkowo zastosowano eliminacje drgań styków przycisków, która zabezpiecza by w momencie jednego przyciskania lub zwalniania klawisza mikrokontroler nie zinterpretował tego jako wielokrotne działanie. Problem przedstawiony został na rysunku 3.2. Sposobem na ominięcie tego zjawiska jest przeczekanie drgań styków przycisku.

Komunikacja z PC. Mikrokontroler ATmega128 wyposażony jest w USART. Umożliwia on wysyłanie oraz odbieranie danych z komputera PC. USART należy zainicjować programowo po czym porty na których odbywa się komunikacja nie mogą służyć już do normalnych operacji wejścia/wyjścia, pełnią wtedy funkcję specjalną jaką jest transmisja danych.

Procedury wykorzystywane w komunikacji z PC za pomocą USART:

`void USART_init(unsigned int myubrr)` – Inicjalizuje USART. Włącza odbiornik i nadajnik, ustala prędkość transmisji i format ramki.

`void USART_Transmit(unsigned char data)` – Wysyła znak do portu szeregowego.

`void USART_puts(char *str)` – Wysyła ciąg znaków do portu szeregowego, wykorzystuje procedurę `USART_Transmit`.

`unsigned char USART_Receive(void)` – Odbiera znak z portu szeregowego.

Obsługa przetwornika AD7708. Mikrokontroler komunikuje się z przetwornikiem analogowo-cyfrowym za pomocą interfejsu SPI. Przed rozpoczęciem wysyłania i odbierania danych z ADC zależy włączyć tryb SPI. Podobnie jak w przypadku aktywacji USART, po włączeniu trybu SPI, linie na których odbywa się komunikacja pełnią specjalne role i nie jest możliwe normalne korzystanie z portów na których odbywa się transmisja danych.

Komendy trybu SPI:

`void SPI_MasterInit(void)` – Aktywuje SPI w trybie master oraz ustala prędkość transmisji.

`void SPI_MasterTransmit_AD(unsigned char cData)` – Wysyła 8 bitów do urządzenia peryferyjnego.

`unsigned char SPI_MasterReceive_AD8(void)` – Odbiera 8 bitów z urządzenia peryferyjnego.

`unsigned short SPI_MasterReceive_AD16(void)` – Odbiera 16 bitów z urządzenia peryferyjnego.

W trybie SPI komunikacja odbywa się synchronicznie. Sygnał zegarowy wysyłany jest tylko w momencie wysyłania instrukcji z Mastera do Slave'a. Po wysłaniu instrukcji, której wynikiem ma być odpowiedź należy posłać do odbiornika dowolną instrukcję (np. ciąg zer) o długości oczekiwanej odpowiedzi. Dodatkowo do aktywacji urządzenia peryferyjnego używana jest linia CS.

Procedury obsługi przetwornika analogowo-cyfrowego:

`void AD7708_Init(void)` – Ustawia linie CS i RESET jako wyjścia a RDY jako wejście MCU. Aktywuje tryb SPI i włącza ADC.

`void ADC_SEND(unsigned char reg, unsigned char cmd)` – Wysyła komendę do wybranego rejestru przetwornika.

`unsigned char ADC_RECEIVE8(unsigned char reg)` – Odczytuje 8 bitową odpowiedź przetwornika.

`unsigned short ADC_RECEIVE16(unsigned char reg)` – Odczytuje 16 bitową odpowiedź przetwornika.

`unsigned short AD7708_READ_CH(void)` – Odczytuje wartość napięcia na wybranym kanale w postaci 16 bitowej, wykorzystuje funkcję `ADC_RECEIVE16`.

Na czas wykonywania procedur komunikacyjnych z ADC, na wejściu CS przetwornika analogowo-cyfrowego wymuszony jest stan niski.

Obsługa karty SD. Komunikacja mikrokontrolera z kartą pamięci odbywa się z wykorzystaniem interfejsu SPI. O ile zaimplementowanie obsługi karty pamięci dla samego zapisu i odczytu surowych danych nie jest bardzo skomplikowaną czynnością, to napisanie oprogramowania zapewniającego możliwość utworzenia na karcie plików w systemie FAT[1] wymaga bardzo dobrej znajomości systemu plików i sporych umiejętności w dziedzinie informatyki. W oprogramowaniu wykorzystano darmowy, kompletny program obsługi kart pamięci z systemem FAT (*FatFs*), który po odpowiedniej modyfikacji spełnia swoje zadanie. Obsługa kart pamięci za pomocą FatFS jest bardzo rozbudowana, posiada bardzo wiele funkcji. Poniżej opisano te z których korzystano w programie głównym.

Procedury obsługi karty pamięci:

`FRESULT f_mount (BYTE Drive, FATFS* FileSystemObject)` – Rezerwuje miejsce pracy dla modułu FatFs, montuje dysk.

`FRESULT f_open (FIL* FileObject, const TCHAR* FileName, BYTE ModeFlags)` – Tworzy nowy plik lub otwiera istniejący.

`FRESULT f_close (FIL* FileObject)` – Zamyka otwarty plik.

`int f_printf (FIL* FileObject, const TCHAR* Foramt, ...)` – Zapisuje sformatowany ciąg znaków do pliku.

[1] FAT (ang. File Allocation Table) – system plików stosowany w systemach operacyjnych, m. in. DOS i Windows

Rozdział 4
Uruchomienie układu

Aby wykorzystać jonometr do pomiaru napięcia na elektrodach jonoselektywnych należy użytkować urządzenie wraz z układem przedwzmacniaczy, które zapewnią odpowiednią impedancje wejściową i umożliwią wykonanie pomiaru. Jonometr może być zasilany z baterii lub napięcia z portu USB. Napięcie dla układu kondycjonowania dostarczane jest z urządzenia za pomocą złącza szpilkowego. Program zapisany w pamięci mikrokontrolera można modyfikować za pomocą programowania w trybie ISP[1] lub JTAG[2].

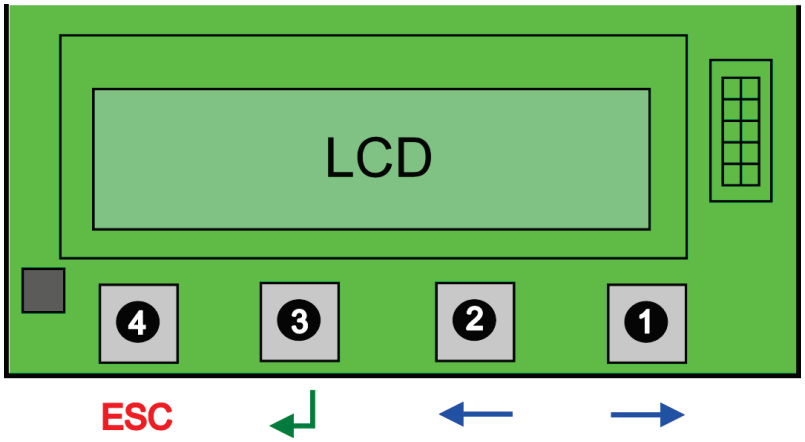
4.1. Czynności wstępne

Pierwszą czynnością, którą należy wykonać by korzystać z urządzenia jest podłączenie źródła zasilania. Jeżeli doprowadzone napięcie mieści się w przedziale 4–6V na wyświetlaczu przez 5 sekund pojawi się napis „ZASILANIE OK!" po czym urządzenie przejdzie do pierwszego stopnia Menu. W przypadku gdy napięcie będzie niewystarczające urządzenie się nie uruchomi. Możliwe jest podłączenie jednocześnie zasilania z baterii i portu USB, co oznacza, że podłączenie urządzenia do komputera nie wymaga odłączenia baterii.

Programując układ należy pamiętać, że urządzenie nie jest przystosowane do zasilania poprzez programator. W czasie programowania zalecane jest zasilanie poprzez port USB, gdyż odłączenie napięcia (np. poprzez wyładowanie baterii) w czasie zapisywania programu do pamięci flash może spowodować uszkodzenie mikrokontrolera.

[1] ISP (ang. In-System Programming) – tryb programowania systemów mikroprocesorowych umożliwiający zaprogramowanie układu bez demontażu z urządzenia w którym pracuje

[2] JTAG (ang. Joint Test Action Group) – tryb programowania stosowany do uruchamiania i programowania systemów mikroprocesorowych, pozwalający na programowanie układu w gotowym urządzeniu



RYSUNEK 4.1. Opis funkcji przycisków.

4.2. Menu urządzenia

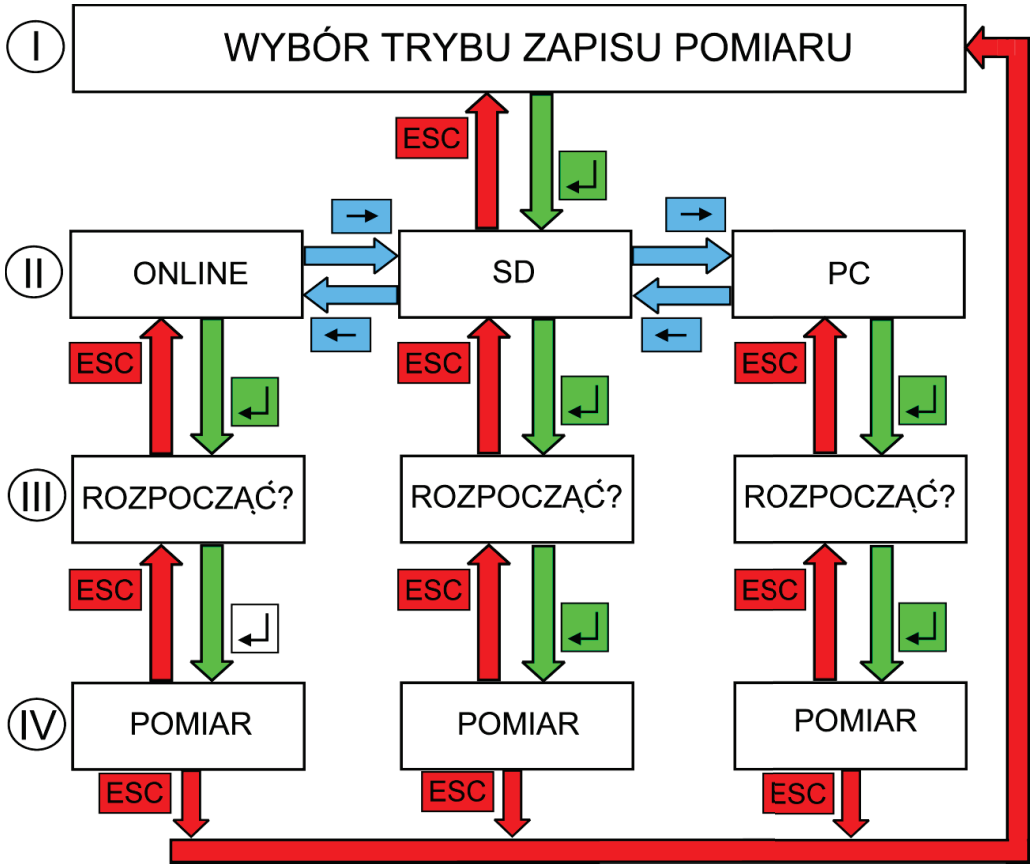
Urządzenie przechodzi do pierwszego stopnia menu po podłączeniu odpowiedniego zasilania. Sterowanie odbywa się za pomocą czterech przycisków umieszczonych pod wyświetlaczem LCD. Funkcje przycisków przedstawiono na rysunku 4.1. Przycisk 4 służy do przerwania pomiaru lub powrotu do poprzedniego stopnia menu, 3 jest przyciskiem zatwierdzenia. Klawisze 1 i 2 służą do przełączania możliwych opcji na poziomie jednego stopnia menu.

Po włączeniu urządzenia użytkownik wybiera tryb zapisu wyników pomiaru. Układ menu przedstawiono na rysunku 4.2.

4.3. Pomiar

Przed wykonaniem pomiaru należy podłączyć układ kondycjonowania sygnału. Gdy pomiar wykonywany jest w terenie, zaleca się korzystanie z wodoszczelnej obudowy, która ograniczy możliwość uszkodzenia układu.

Po włączeniu zasilania należy wybrać odpowiedni tryb pomiaru. Aby tego dokonać należy przycisnąć klawisz 3 a następnie klawiszami 1 i 2 wybrać odpowiedni tryb zapisu danych pomiarowych i zatwierdzić ponownie klawiszem 3. Przed rozpoczęciem pomiaru wyświetlony zostanie komunikat „CZY ROZPOCZAC?”. Przed zatwierdzeniem należy podłączyć elektrody do odpowiednich gniazd wejściowych układu kondycjonowania



RYSUNEK 4.2. Układ menu jonometru.

oraz zanurzyć w badanym roztworze. Jeżeli zostało to wykonane, należy przycisnąć klawisz 3. W przypadku wciśnięcia przycisku 4 urządzenie wraca do poprzedniego stopnia menu.

Jonometr może pracować w następujących trybach wyświetlania/zapisu danych:

ONLINE – w tym trybie użytkownik może obserwować wyniki pomiarów wyłącznie na wyświetlaczu LCD. Z powodu ograniczonych wymiarów ekranu LCD, w jednej chwili mogą być wyświetlanie wyniki tylko z czterech kanałów. Urządzenie co dwie sekundy wskazuje na przemian wyniki pomiarów z kanałów 1–4 i 5–7.

SD – w trybie SD urządzenie zapisuje wyniki z wszystkich kanałów na karcie pamięci. Wyniki pomiarów zapisane są w pliku tekstowym. Zapis danych następuje w odstępach jednosekundowych. Na wyświetlaczu wskazywane są wyniki pomiarów w taki sam sposób jak w

trybie ONLINE. Do zapisu danych potrzebna jest karta SD z systemem plików FAT.

PC – w celu przekazania wyników pomiarów do komputera PC niezbędny jest program do komunikacji po RS-232. Po podłączeniu urządzenia do komputera należy uruchomić terminal komunikacyjny, ustawić parametry transmisji i nawiązać połączenie. Prędkość przesyłu ustawiono na 2400 bitów na sekundę, format ramki to 8 bitów danych, dwa bity stopu, brak bitu parzystości. Aby odebrać wyniki pomiarów, należy wcisnąć na klawiaturze przycisk ENTER. Na wyświetlaczu wyniki przedstawione są na przemian, co dwie sekundy kanały 1–4 i 5–7.

Rozdział 5
Analiza niepewności pomiaru

Budowany jonometr jest cyfrowym przyrządem pomiarowym, działającym w sposób dyskretny w czasie. Jedną z podstawowych informacji, którą powinna dysponować osoba wykonująca pomiar za pomocą przyrządu pomiarowego jest znajomość przedziału, w którym z przyjętym prawdopodobieństwem, znajduje się wartość poprawna wielkości mierzonej. W tym celu dokonuje się analizy niepewności pomiaru.

5.1. Sposób wyznaczania niepewności pomiaru

Pomiar wielkości mierzonej ma na celu wyznaczenie najbardziej prawdopodobnej wartości $\widehat{x}$ z przedziału wartości $[\widehat{x} - U,\ \widehat{x} + U]$, w którym na pewnym poziomie ufności znajduje się wartość poprawna wartości mierzonej X [**9**].

$$\widehat{x} - U \leq X \leq \widehat{x} + U$$

$$U = k \cdot u_c \tag{3}$$

gdzie:

X – prawdziwa wartość wielkości mierzonej,
$\widehat{x}$ – estymowana wartość mierzona x, wyznaczona na podstawie wielkości wejściowych $\widehat{x}_1$,... $\widehat{x}_j$, z uwzględnieniem błędów systematycznych
U – niepewność rozszerzona,
u_c – złożona niepewność standardowa,
k – współczynnik rozszerzenia zapewniający przyjęty poziom ufności.

W celu określenia niepewności wyniku pomiaru należy uwzględnić wpływ wszelkich istotnych wielkości wpływowych, wraz z tymi, które nie są znane podczas wykonywania pomiaru oraz takich, które można uznać za losowe [**6**].

Złożoną niepewność standardową u_c wyznacza się jako pierwiastek sumy kwadratów cząstkowych niepewności standardowych szacowanych metodą typu A (u_A) oraz metodą

typu B (u_B):

$$u_c = \sqrt{u_A^2 + u_B^2} \tag{4}$$

Błędy systematyczne, których wartość jest znana przed lub w trakcie pomiaru, powinny być uwzględnione w procesie estymacji wartości mierzonej w postaci odpowiedniej poprawki. Błędy systematyczne, których nie uwzględnia się w procesie estymacji wielkości mierzonej, są szacowane metodą typu B.

Szacowanie niepewności metodą typu A. Metoda szacowania niepewności typu A ma zastosowanie, jeśli dla kilku niezależnych obserwacji wielkości mierzonej w tych samych warunkach pomiarowych obserwowany jest rozrzut otrzymanych wartości. Miarą rozrzutu jest odchylenie standardowe zbioru pomiarów. Im mniejsza wartość odchylenia standardowego otrzymanych wyników, tym większa jest precyzja pomiaru [**6**]. Jeżeli można uznać, że błąd związany z wielkością mierzoną zmienia się losowo z pomiaru na pomiar, to źródła niepewności należy analizować metodą typu A. Za wartość oczekiwaną wielkości mierzonej przyjmuje się średnią z n wyników pomiarów. Niepewność standardową wyznaczonego wyniku pomiaru estymuje się za pomocą odchylenia standardowego średniej:

$$u_A = \sqrt{\frac{1}{n(n-1)} \sum_{i=1}^{n} (x_i - \overline{x})^2} = \frac{1}{\sqrt{n}} \widehat{\sigma}(x_i) \tag{5}$$

gdzie:

n – liczba pomiarów wykonanych w jednakowych warunkach,
x_i – i-ty pomiar,
$\widehat{\sigma}(x_i)$ – estymowana wartość odchylenia standardowego,
$\overline{x}$ – wartość średnia serii pomiarów $\overline{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$.

Cechą niepewności wyznaczanych metodą typu A jest to, że niepewność u_A zmniejsza się $1/\sqrt{n}$ razy wraz z ilością wykonanych pomiarów. Z tego powodu niepewności złożone szacowane metodą typu A i B analizuje się oddzielnie, pozwala to łatwiej oszacować dokładność pomiaru podczas zmiany ilości powtórzeń pomiaru.

Szacowanie niepewności metodą typu B. Niepewność typu B wyznaczana jest za pomocą analizy naukowej opartej na wszystkich dostępnych informacjach na temat zmienności wielkości mierzonej. Informacje te mogą pochodzić z wcześniej przeprowadzonych

pomiarów, wiedzy o zjawiskach zachodzących w trakcie pomiaru, posiadanego doświadczenia oraz danych podanych przez producenta urządzenia pomiarowego. Aby właściwie oszacować niepewność metodą typu B należy posiadać szeroką wiedzę o zjawiskach występujących w procesie pomiaru. Tylko bardzo dobra znajomość procesu pomiaru pozwala na uwzględnienie wszystkich istotnych czynników i właściwe oszacowanie niepewności [**6**].

5.2. Niepewność pomiaru wykonanego urządzenia

Pomiar budowanym jonometrem, sprowadza się do pomiaru wartości napięcia, które za pomocą układu kondycjonowania przekazywane jest do przetwornika analogowo-cyfrowego a następnie przetwarzane na postać cyfrową. Na niepewność pomiaru wpływają elementy części analogowej, czyli parametry wzmacniaczy układu kondycjonowania oraz przetwornika A/C. Niepewność pomiaru oszacowano metodą typu B. W metodzie tej wykorzystać można informacje podane przez producentów układów wykorzystanych przy pomiarze napięcia (wzmacniacze i przetwornik A/C).

Szacowanie niepewności standardowej wzmacniacza AD8231. Najważniejsze parametry wzmacniacza, które wpływają na niepewność pomiaru napięcia zostały podane przez producenta i przedstawione w tabeli 5.1. Parametry zostały wyznaczone dla źródła napięcia o rezystancji 1MΩ. Dane producenta odczytano dla następujących wartości:

V_S – napięcie zasilania $V_S = 5V$,
V_{Ref} – napięcie referencyjne $V_{Ref} = 2{,}5V$,
G – wzmocnienie $G = 1$,
T_A – temperatura w czasie pomiaru $T_A - (5 - 30)°C$.

Dla podanych przez producenta błędów przyjęto rozkład prostokątny błędu. Niepewność standardową obliczono poprzez iloraz wartości błędu przez $\sqrt{3}$.

Wykorzystując dane z tabeli 5.1 wyznaczono niepewność pomiaru napięcia jaką wprowadza wzmacniacz AD8231. Niepewność tą wyznaczono obliczając pierwiastek z sumy kwadratów poszczególnych składowych wpływających na pomiar.

$$u_W = 151{,}15\mu\text{V}$$

Szacowanie niepewności standardowej przetwornika analogowo-cyfrowego AD7708. Parametry wzmacniacza mające wpływ na dokładność przetwarzania sygnału analogowego na postać cyfrową zostały podane przez producenta w dokumentacji ADC. Tabela 5.2 przedstawia wartości najważniejszych błędów przetwornika analogowo-cyfrowego. Parametry podane przez producenta odczytano dla następujących warunków:

V_S – napięcie zasilania $V_S = 5V$,
V_{Ref} – napięcie referencyjne $V_{Ref} = 2{,}5V$,
T_A – temperatura w czasie pomiaru $T_A - (5 - 30)°C$.

Podobnie jak dla wzmacniacza, dla podanych przez producenta błędów przyjęto rozkład prostokątny błędu. Niepewność standardową obliczono poprzez iloraz wartości błędu przez $\sqrt{3}$. Błędy wyznaczono dla zakresu pomiarowego ±1,28V

Niepewność pomiaru napięcia za pomocą przetwornika AD7708 wyznaczono korzystając z wartości z tabeli tabeli 5.1. Niepewność wyznaczono obliczając pierwiastek z sumy kwadratów poszczególnych składowych wpływających na pomiar.

TABLICA 5.1. Parametry wzmacniacza AD8231.

Parametr	Wartość podana przez producenta	Niepewność standardowa
Napięcie niezrównoważenia	4μV	2,31μV
Wejściowy prąd polaryzujący	250pA (250μV)	144,34μV
Wejściowy prąd niezrównoważenia	20pA (20μV)	11,55μV
Wpływ temperatury na błąd wzmocnienia	75μV	43,30μV

TABLICA 5.2. Parametry przetwornika analogowo-cyfrowego AD7708.

Parametr	Wartość podana przez producenta	Niepewność standardowa
Błąd kwantyzacji	39,06μV	22,55μV
Błąd nieliniowości całkowej	19,2μV	11,09μV
Błąd zera	38,4μV	22,17μV
Błąd wzmocnienia	22,5μV	12,99μV
Błąd niezrównoważenia	19,5μV	11,26μV

$$u_P = 37{,}66\mu V$$

Niepewność złożona pomiaru napięcia zbudowanym jonometrem. Niepewność złożoną pomiaru napięcia wykonanym jonometrem z wykorzystaniem metody szacowania niepewności typu B przedstawia wzór:

$$u_B = \sqrt{u_W^2 + u_P^2} \tag{6}$$

gdzie:

u_B – niepewność pomiaru napięcia jonometrem oszacowana metodą typu B,

u_W – niepewność standardowa wpływająca na pomiar napięcia przez zastosowany wzmacniacz AD8231,

u_P – niepewność standardowa wpływająca na pomiar napięcia przez zastosowany przetwornik analogowo-cyfrowy AD7708.

Podstawiając do powyższego wzoru wyznaczone wartości niepewności u_W oraz u_P otrzymujemy wartość niepewności złożonej pomiaru napięcia elektrod jonoselektywnych, która wynosi:

$$u_B = 155{,}77\mu\text{V}$$

Rozdział 6
Wnioski z pracy

6.1. Analiza kosztów

Koszty, które zostały poniesione przy budowie urządzenia mają istotne znaczenie z punktu widzenia ewentualnej produkcji seryjnej jonometru, zakładanej w celach pracy. Można je podzielić na koszty rzeczywiste budowy prototypu jonometru, oraz przybliżoną cenę produkcji urządzenia na większą skalę. Analizę kosztów wykonano dla dwóch przypadków. Pierwsza analiza dotyczy zbudowania urządzenia identycznego z prototypem, druga – urządzenia z elementami o podobnych parametrach lecz tańszych od tych zastosowanych w zbudowanym jonometrze.

Koszty prototypu. Analizę przeprowadzono dla urządzenia o identycznych elementach, które zostały zastosowane w prototypie jonometru. Pod uwagę wzięto ceny hurtowe. Zestawienie cen poszczególnych elementów przedstawiono w tabeli 6.1.

Tablica 6.1. Koszty elementów prototypu.

Element	Typ	Cena
Mikrokontroler	ATmega128L	22,96 zł
Przetwornik A/C	AD7708BR	14,42 zł
Źródło napięcia referencyjnego	ADR421	10,08 zł
Konwerter RS232/USB	FT232RL	7,98 zł
Wyświetlacz LCD	WMC1602M	13,55 zł
Stabilizator napięcia 3V3	LTC1627	9,10 zł
Przyciski	TACTDQ120	13,96 zł
Płytka PCB wraz z lutowaniem elementów		94,32 zł
Pozostałe elementy		86 zł
SUMA		**272,37 zł**

Koszty urządzenia porównywalnego parametrami. Po zbudowaniu urządzenia i ocenie jego zgodności z zakładanymi w celach pracy parametrami, można przeanalizować możliwość zastąpienia zastosowanych w jonometrze elementów, innymi.

Pamięć zaprogramowanego mikrokontrolera ATmega128 jest zajęta w około 80%. Największą ilość pamięci zajmuje oprogramowanie do obsługi systemu plików FAT (FatFs) – około 60%. Obsługa plików FAT jest bardzo rozbudowana a w urządzeniu wykorzystywane są tylko cztery jej funkcje. W przypadku gdyby udało się napisać nową procedurę obsługi plików bądź zrezygnować z zapisu na karcie plików w systemie FAT możliwym byłoby ograniczenie pamięci programu poniżej 32kB. Pozwoliłoby to zastosować tańszy mikrokontroler ATmega32U2, z wbudowanym kontrolerem obsługi USB [**3**]. Takie rozwiązanie pozwoliłoby na wyeliminowanie z układu konwertera FT232RL. Jednakże, z racji mniejszej ilości portów wejścia/wyjścia sterowanie wyświetlaczem należałoby ograniczyć do czterech linii i wybrać tylko jedną możliwość programowania mikrokontrolera (JTAG lub ISP). Koszt ATmega32U2 to 10,22 zł co obniża koszt urządzenia o 20,72 zł.

Przetwornik analogowo-cyfrowy AD7708 można by zastąpić 16. bitowym przetwornikiem LTC2460. Przetwornik ten posiada wbudowane źródło napięcia referencyjnego, dodatkowo jest tańszy od ADC zastosowanego w prototypie urządzenia [**8**]. Przetwornik analogowo-cyfrowy LTC2460 posiada interfejs SPI jednak w porównaniu do AD7708 ma nieznacznie gorsze parametry dokładnościowe. Zastosowanie tego ADC zwiększyłoby niepewność pomiaru zbudowanego urządzenia. Przetwornik LTC2460 jest tańszy od AD7708, wbudowane źródło napięcia referencyjnego pozwala na rezygnację z ADR421 co zmniejsza koszt urządzenia o 18,51 zł.

Modyfikując wartości z tabeli 6.1 o ceny zastępczych elementów otrzymujemy wartość całego urządzenia o 39,23 zł niższą. Kwota ta jest mniejsza o 14% ceny prototypu. Minusem zastosowania tańszego mikrokontrolera jest ograniczona możliwość ulepszania wyprodukowanych urządzeń np. poprzez aktualizację oprogramowania, która mogłaby nie zmieścić się w pamięci MCU.

6.2. Podsumowanie

Efektem końcowym wykonanej pracy jest urządzenie, które wraz z układem kondycjonowania sygnału może służyć do wykonywania wieloskładnikowych pomiarów potencjometrycznych. Prace wykonywane były zgodnie z wyznaczonymi celami i wymaganiami.

Przegląd gotowych rozwiązań komercyjnych jonometrów pozwolił uzyskać informację ich wadach i zaletach. Nie udało się znaleźć dostępnych w handlu urządzeń umożliwiających przeprowadzanie pomiarów wielokanałowych co stawia wykonany prototyp ponad nimi. Komercyjne urządzenia dostarczyły informacji pomocnych w sformułowaniu wymagań względem funkcjonalności urządzenia umożliwiając zdefiniowanie celów i potrzeb projektowych.

Projekt był podzielony na dwie części. Pierwsza, polegająca na zaprojektowaniu układu elektronicznego i druga, która miała na celu napisanie oprogramowania umożliwiającego obsługę urządzenia. W rozdziale 2. opisano najważniejsze właściwości głównych elementów układu elektronicznego wraz z metodologią ich wzajemnych połączeń. Efektem końcowym jest kompletny schemat elektryczny, na podstawie którego wykonano płytkę drukowaną. W rozdziale 2. przedstawiono również ideowy schemat układu kondycjonowania sygnału oraz wyjaśniono celowość jego zastosowania w pomiarach potencjometrycznych.

Metoda tworzenia oprogramowania dla urządzeń mikroprocesorowych opisana została w rozdziale 3. niniejszej pracy. Opisane zostały zarówno ogólne czynności konieczne przy programowaniu mikrokontrolerów, jak również konkretny sposób programowania budowanego urządzenia. Objaśnienie programu głównego ograniczone zostało do najważniejszych komend obsługujących urządzenia peryferyjne. Całość programu znajduje się w dodatkach pracy.

W rozdziale 4. zawarto informacje związane z poprawnym użytkowaniem zbudowanego urządzenia. Opisano możliwości zasilania, tryby pracy oraz sposób obsługi jonometru.

Analiza niepewności pomiaru toru analogowego zbudowanego jonometru przedstawiona została w rozdziale 5. Na całkowitą niepewność pomiaru, oprócz zbudowanego układu, istotne znaczenie mają parametry układu kondycjonowania sygnału. W głównej mierze na niepewność wpływa impedancja wejściowa zastosowanych wzmacniaczy pomiarowych. Jeżeli, w porównaniu do rezystancji elektrod jonoselektywnych, jest ona zbyt niska może dojść do sytuacji, że fakt ten znacznie wpłynie na zwiększenie niepewności

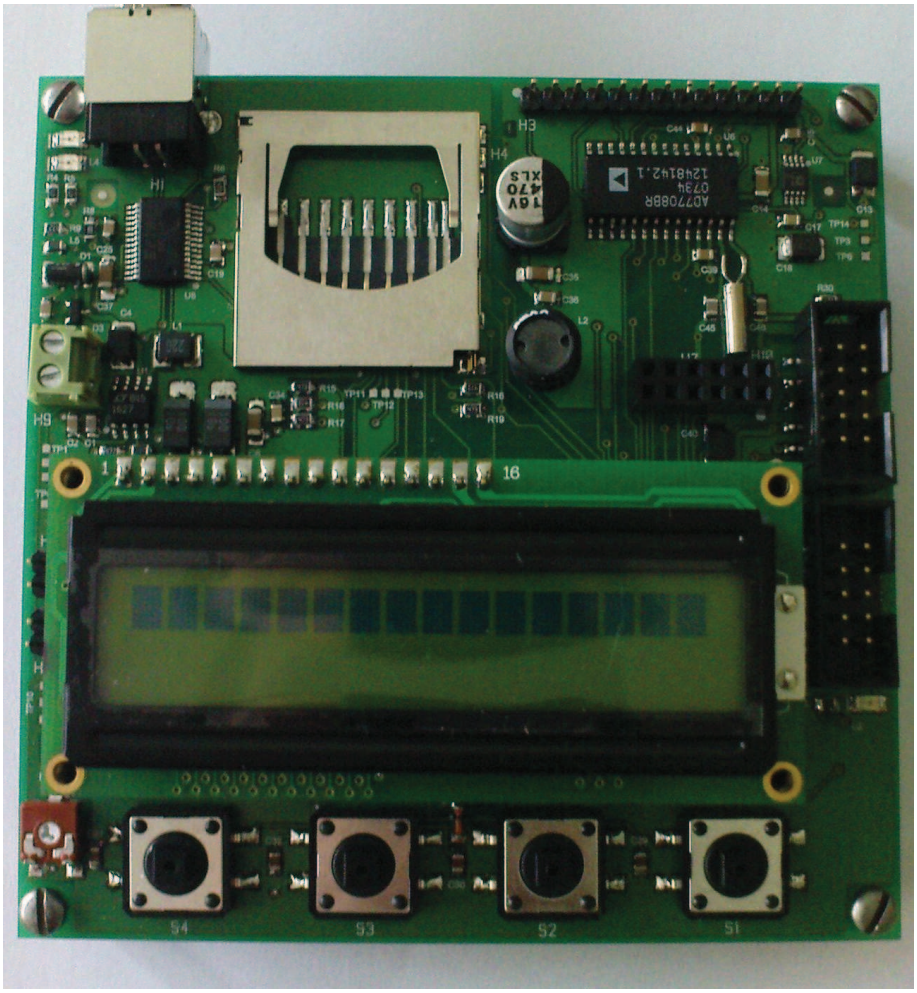
pomiaru jonometrem. Obliczenia niepewności pomiaru napięcia spowodowanych zastosowanymi w układzie kondycjonowania wzmacniaczami AD8231 wykazały, że jest ona kilkukrotnie większa od niepewności wprowadzanej przez ADC. Parametry dokładnościowe układu kondycjonowania przesądziły o fakcie, iż nie udało się osiągnąć zakładanej dokładności pomiaru napięcia na elektrodach jonoselektywnych. Zakładana w celach pracy wartość ±0,1mV została przekroczona o ponad połowę i wyniosła u = ±0,15 mV.

Należy pamiętać, że przy podawaniu wyniku pomiaru zaleca się stosować niepewność rozszerzoną. W obliczeniach rozszerzonej niepewności pomiaru, należy przemnożyć wartość otrzymanej niepewności standardowej przez współczynnik rozszerzenia k. Przyjmując wartość współczynnika rozszerzenia k = 2, który odpowiada 95% poziomowi ufności, otrzymana wartość niepewności rozszerzonej wynosiłaby U = ±0,3 mV. Oznacza to trzykrotne przekroczenie założeń początkowych. Poprawić dokładność pomiaru mogłoby zastosowanie układu kondycjonowania opartego na wzmacniaczach o wyższej impedancji wejściowej. Dla celów szybkich pomiarów w terenie, otrzymana wartość niepewności rozszerzonej pomiaru jonometrem wydaje się być wystarczająca.

W przeprowadzonej analizie kosztów budowanego jonometru nie uwzględniono ceny układu kondycjonowania sygnału. W porównaniu do rozwiązań komercyjnych, które pozwalają na wykonanie pomiarów w trudnych warunkach atmosferycznych, zbudowane urządzenie powinno posiadać także wodoszczelną obudowę. Szacując koszt układu kondycjonowania sygnału, którego seryjne wykonanie nie powinno przekroczyć kwoty 150zł i kosztów obudowy wynoszącej około 100zł, całkowita cena jonometru zamykałaby się w sumie około 500zł co czyniłoby urządzenie znacznie bardziej dostępne dla użytkownika niż rozwiązania aktualnie dostępne na rynku.

6.3. Propozycja dalszego rozwoju

Efektem końcowym projektu jest urządzenie prototypowe, które z pewnością nie jest wolne od wad. Dalsze prace nad projektem powinny uczynić zbudowany układ produktem w pełni konkurencyjnym do oferowanych przez rynek urządzeń. Analiza kosztów wykazała, że wartym zastanowienia jest możliwość zastąpienia oprogramowania obsługi karty SD. Wymaga to sporo wiedzy i doświadczenia informatycznego, jednak miałoby to wpływ



RYSUNEK 6.1. Płytka jonometru.

na dodatkowe obniżenie ceny gotowego jonometru. Zbudowane urządzenie umożliwia przeprowadzenie wielu usprawnień oprogramowania np. wprowadzenie funkcji kalibracji przetwornika A/C, zmian zakresu pomiarowego lub częstotliwości próbkowania mierzonego sygnału.

Testy urządzenia wykazały, że układ bardzo dobrze radzi sobie z pomiarem napięcia źródła o niewielkiej rezystancji wewnętrznej. Wartość zmierzona napięcia pokrywa się z wartością zadaną. Wahania występują tylko na najmniej znaczącym bicie, co oznacza, że szumy układu praktycznie nie wpływają na pomiar napięcia.

Niestety podczas połączenia urządzenia z układem kondycjonowania sygnału, na wartość napięcia zadanego do układu kondycjonowania z niewiadomych przyczyn wpływ miało samo urządzenie. Po sprawdzeniu napięcia wychodzącego z układu kondycjonowania okazało się, że urządzenie wskazywało prawidłową wartość jednak nie pokrywała się ona z

napięciem na elektrodach. Nie udało się wyjaśnić przyczyn tego zjawiska. Możliwym jest, że powodem zaburzeń wartości napięcia na elektrodach jest wadliwe działanie układu kondycjonowania sygnału. Należałoby zaprojektować i zbudować nowy układ co nie jest celem tej pracy, jednak daje możliwości dla dalszego rozwoju projektu. Nowy układ kondycjonowania sygnału oparty na wzmacniaczach pomiarowych o wyższej impedancji wejściowej, wpłynąłby także na poprawę dokładności pomiaru napięcia.

Zbudowany układ przedstawiony jest na rysunku 6.1. Problemy z współpracą zbudowanego urządzenia z układem kondycjonowania uniemożliwiły wykonanie obudowy dla jonometru. Mogłoby dojść do sytuacji, że oba układy nie mieszczą się w jednej obudowie. Zabezpieczenie zewnętrzne urządzenia należy wykonać dopiero po zapewnieniu odpowiedniej współpracy zbudowanego urządzenia z układem kondycjonowania sygnału.

Bibliografia

[1] Analog Devices, Inc., One Technology Way, P.O. Box 9106; Norwood, MA 02062-9106, USA. *8-/10-Channel, Low Voltage, Low Power, Σ–Δ ADCs*, 2001.

[2] Atmel Corporation, 2325 Orchard Parkway; San Jose, USA. *8-bit AVR Microcontroller with 128K Bytes In-System Programmable Flash Manual – ATmega128/ATmega128L*, 2008.

[3] Atmel Corporation, 2325 Orchard Parkway; San Jose, USA. *8-bit AVR Microcontroller with 8/16/32K Bytes of ISP Flash and USB Controller Manual – ATmega8U2/ATmega16U2/ATmega32U2*, 2009.

[4] K. Cammann. *Zastosowanie elektrod jonoselektywnych*. WNT, Warszawa, wydanie 1, 1977.

[5] J. Doliński. *Mikrokontrolery AVR w praktyce*. Wydawnictwo BTC, Warszawa, wydanie 1, 2003.

[6] A. Kozyra, A. Wiora, J. Wiora. *Szacowanie niepewności w pomiarach potencjometrycznych*. Wydawnictwo Pracowni Komputerowej Jacka Skalmierskiego, Katowice, wydanie 1, 2007.

[7] A. Krysiak. *Programowanie mikrokontrolerów rodziny AVR Część 1*. Studio Wydawniczo-Typograficzne „Typoscript" Andrzej Ploch, Wrocław, wydanie 1, 2000.

[8] Linear Technology Corporation, 1630 McCarthy Blvd.; Milpitas, CA 95035-7417, USA. *Ultra-Tiny, 16-Bit Σ–Δ ADCs with 10ppm/°C Max Precision Reference Manual – LTC2460/LTC2462*, 2009.

[9] J. Piotrowski. *Podstawy miernictwa*. WNT, Warszawa, wydanie 1, 2002.

[10] K. Ren. *Selektywność elektrod jonoselektywnych z ciekłymi membranami*. Wydawnictwo Naukowe Uniwesytetu im. Adama Mickiewicza w Poznaniu, Poznań, wydanie 1, 2000.

[11] C. S. Ulrich Tietze. *Układy półprzewodnikowe*. WNT, Warszawa, wydanie 3, 1996.

Dodatek A

Kody programów

```
1  /* Program dla układu ATMEGA128L, pełniący funkcję jonometru */
2
3  /* USTAWIENIA */
4
5  /* Prędkość transmisji 2400 */
6  #define BAUD 2400
7  #define MYUBRR  F_CPU/BAUD/16-1
8
9  /* Pliki nagłówkowe */
10
11 //#define F_CPU 8000000L
12 #include <stdio.h>
13 #include <avr/io.h>
14 #include <util/delay.h>
15 #include <avr/pgmspace.h>
16 #include <avr/interrupt.h>
17 #include <string.h>
18 #include <inttypes.h>
19 #include <stdlib.h>
20 #include "hd44780.h"
21 #include "keys.h"
22 #include "usart.h"
23 #include "spi.h"
24 #include "AD7708.h"
25 #include "xitoa.h"
26 #include "ff.h"
27 #include "diskio.h"
28
29 /* Definicje funkcji */
30
31 /* Inicjalizacja sprzętu */
32
33 void io_init(void)  //Ustawienia kierunków portów uC (we/wy)
34 {
35 DDRA = 0x84; // wyjście na pin7 i pin2 (MPC_SHD i SD_CS)
36 DDRB = 0xF6; // wyjście na pin7-pin4 + pin 2-pin1 (BUF_OE_n, LCD_E, LCD_R/W, LCD_RS, MOSI, SCK)
37 DDRC = 0xFF;     // wyjście na pin7-pin0 (syg DB7-DB0)
```

```
38 DDRD = 0x00;     // wejście
39 DDRE = 0x04;     // wyjście na pin2 (syg 5V_LCD_EN)
40 DDRG = 0x00; // wejscie SW2, SW3
41 PORTG = 0x18; // wymuszony stan wysoki na wejsciu SW2, SW3
42
43 sei();
44 }
45
46 /********* Początek programu **********/
47
48 int main(void)
49 {
50    /* Inicjalizacja portów I/O */
51    io_init();
52
53    /* Inicjalizacja wyświetlacza */
54    lcd_init();
55
56    /* Inicjalizacja przetwornika A/C */
57    AD7708_Init();
58
59    // Deklaracja zmiennych globalnych
60
61    char nazwa[30] ="0:Pomiar1.txt";
62    char seria[60]="";
63    char seria1[30]="";
64    char seria2[30]="";
65
66    unsigned short pomiark1=0;
67    unsigned short pomiark2=0;
68    unsigned short pomiark3=0;
69    unsigned short pomiark4=0;
70    unsigned short pomiark5=0;
71    unsigned short pomiark6=0;
72    unsigned short pomiark7=0;
73    int MENU =0;
74
75    char pom1[10]="";
76    char pom2[10]="";
77    char pom3[10]="";
78    char pom4[10]="";
79    char pom5[10]="";
80    char pom6[10]="";
81    char pom7[10]="";
```

```
82      char onln[10]="";
83
84    /********** Główna pętla programu *********/
85
86    /********** WYBÓR TRYBU ZAPISU POMIARU *********/
87
88    LCD_LOCATE(0,0);
89    lcd_puts("ZASILANIE OK!");
90    _delay_ms(5000);
91    LCD_CLEAR;
92
93
94  STOPIEN1:
95  LCD_CLEAR;
96
97    while(1)
98    {
99
100   if(MENU==0)
101   {
102   LCD_LOCATE(0,0);
103   lcd_puts("Wybor Trybu");
104   LCD_LOCATE(0,1);
105   lcd_puts("      ENT");
106
107   if(S3)
108   {
109   _delay_ms(80);
110     while(S3);
111   MENU=2;
112   LCD_CLEAR;
113   }
114   }
115
116   if(MENU==2)
117   {
118   LCD_LOCATE(0,0);
119   lcd_puts("Tryb SD");
120   LCD_LOCATE(0,1);
121   lcd_puts("ESC  ENT  <-  ->");
122
123   if(S1)
124   {
125   _delay_ms(80);
```

```
126     while(S1);
127   MENU=3;
128   LCD_CLEAR;
129   }
130
131   if(S2)
132   {
133   _delay_ms(80);
134     while(S2);
135   MENU=1;
136   LCD_CLEAR;
137   }
138
139   if(S3)
140   {
141   _delay_ms(80);
142     while(S3);
143   MENU=22;
144   LCD_CLEAR;
145   }
146
147   if(S4)
148   {
149   _delay_ms(80);
150     while(S4);
151   MENU=0;
152   LCD_CLEAR;
153   }
154   }
155
156   if(MENU==1)
157   {
158   LCD_LOCATE(0,0);
159   lcd_puts("Tryb ONLINE");
160   LCD_LOCATE(0,1);
161   lcd_puts("ESC  ENT  <-  ->");
162
163   if(S1)
164   {
165   _delay_ms(80);
166     while(S1);
167   MENU=2;
168   LCD_CLEAR;
169   }
```

```
170
171  if(S2)
172  {
173  _delay_ms(80);
174    while(S2);
175  MENU=3;
176  LCD_CLEAR;
177  }
178
179  if(S3)
180  {
181  _delay_ms(80);
182    while(S3);
183  MENU=21;
184  LCD_CLEAR;
185  }
186
187  if(S4)
188  {
189  _delay_ms(80);
190    while(S4);
191  MENU=0;
192  LCD_CLEAR;
193  }
194  }
195
196  if(MENU==3)
197  {
198  LCD_LOCATE(0,0);
199  lcd_puts("Tryb PC");
200  LCD_LOCATE(0,1);
201  lcd_puts("ESC  ENT  <-  ->");
202
203  if(S1)
204  {
205  _delay_ms(80);
206    while(S1);
207  MENU=1;
208  LCD_CLEAR;
209  }
210
211  if(S2)
212  {
213  _delay_ms(80);
```

```
214     while(S2);
215    MENU=2;
216    LCD_CLEAR;
217    }
218
219    if(S3)
220    {
221    _delay_ms(80);
222     while(S3);
223    MENU=23;
224    LCD_CLEAR;
225    }
226
227    if(S4)
228    {
229    _delay_ms(80);
230     while(S4);
231    MENU=0;
232    LCD_CLEAR;
233    }
234    }
235
236   //*****ZAPYTANIE 3 STOPIEN
237
238    if(MENU==22)    // Zapytanie zapisu SD
239    {
240    LCD_LOCATE(0,0);
241    lcd_puts("ROZPOCZAC SD?");
242    LCD_LOCATE(0,1);
243    lcd_puts("ESC  ENT");
244
245    if(S3)
246    {
247    _delay_ms(80);
248     while(S3);
249    MENU=322;
250    LCD_CLEAR;
251    }
252
253    if(S4)
254    {
255    _delay_ms(80);
256     while(S4);
257    MENU=2;
```

```
258  LCD_CLEAR;
259  }
260  }
261
262  if(MENU==21)    // Zapytanie ONLINE
263  {
264  LCD_LOCATE(0,0);
265  lcd_puts("ROZPOCZAC ONL?");
266  LCD_LOCATE(0,1);
267  lcd_puts("ESC  ENT");
268
269  if(S3)
270  {
271  _delay_ms(80);
272   while(S3);
273  MENU=321;
274  LCD_CLEAR;
275  }
276
277  if(S4)
278  {
279  _delay_ms(80);
280   while(S4);
281  MENU=1;
282  LCD_CLEAR;
283  }
284  }
285
286  if(MENU==23)    // Zapytanie PC
287  {
288  LCD_LOCATE(0,0);
289  lcd_puts("ROZPOCZAC PC?");
290  LCD_LOCATE(0,1);
291  lcd_puts("ESC  ENT");
292
293  if(S3)
294  {
295  _delay_ms(80);
296   while(S3);
297  MENU=323;
298  LCD_CLEAR;
299  }
300
301  if(S4)
```

```
302  {
303  _delay_ms(80);
304    while(S4);
305  MENU=3;
306  LCD_CLEAR;
307  }
308  }
309
310 //***** ZAPIS 4 STOPIEN
311
312  if(MENU==322)     // SD
313  {
314
315  while(!S4)
316  {
317  // Czynności SD
318  LCD_CLEAR;
319
320 start:
321    while(SDINS)
322    {
323   LCD_LOCATE(0,0);
324    lcd_puts("Wloz karte SD");
325    _delay_ms(500);
326    }
327    _delay_ms(500);
328   LCD_CLEAR;
329
330    if(SDWP)
331    {
332   LCD_LOCATE(0,0);
333    lcd_puts("Karta chroniona");
334   LCD_LOCATE(0,1);
335    lcd_puts("przed zapisem!");
336
337    _delay_ms(1000);
338   LCD_CLEAR;
339
340    while(SDWP)
341    {
342   LCD_LOCATE(0,0);
343    lcd_puts("Wyjmij karte");
344    }
345    _delay_ms(1000);
```

```
346   goto start;
347   }
348
349 // Karta włożona i gotowa do zapisu danych
350
351         SPI_MasterInit_SD();
352
353         uint8_t stan;
354         stan = disk_initialize(0);
355
356         resultat = f_mount(0, &fs);
357         if (resultat == 0)
358         {
359         LCD_LOCATE(0,0);
360     lcd_puts("Dysk zamontowany");
361         _delay_ms(1000);
362         }
363         else
364         {
365         LCD_LOCATE(0,0);
366     lcd_puts("Blad dysku");
367         _delay_ms(1000);
368         }
369 f_open(&plik, nazwa, FA_CREATE_ALWAYS | FA_WRITE);
370 f_printf(&plik, "AIN1-AINC AIN2-AINC AIN3-AINC AIN4-AINC AIN5-AINC AIN6-AINC AIN7-AINC\n");
371
372         LCD_CLEAR;
373
374         while (!S4)
375         {
376         _delay_ms(80);
377         while(S4);
378
379 //AIN1C
380
381         SPI_MasterInit();
382                 ADC_SEND(WRITE_TO_REG|MODE_REG,CHOP_EN|NEGBUF_DIST|CONT_CONV);
383                 ADC_SEND(WRITE_TO_REG|FILTER_REG,SF_255);
384                 ADC_SEND(WRITE_TO_REG|ADCCON_REG,AIN1C|BIPOL|MIV2560);
385                 pomiark1 = AD7708_READ_CH();
386
387 //AIN2C
388
389         SPI_MasterInit();
```

```
390                     ADC_SEND(WRITE_TO_REG|MODE_REG,CHOP_EN|NEGBUF_DIST|CONT_CONV);
391                     ADC_SEND(WRITE_TO_REG|FILTER_REG,SF_255);
392                     ADC_SEND(WRITE_TO_REG|ADCCON_REG,AIN2C|BIPOL|MIV2560);
393                     pomiark2 = AD7708_READ_CH();
394
395 //AIN3C
396
397             SPI_MasterInit();
398                     ADC_SEND(WRITE_TO_REG|MODE_REG,CHOP_EN|NEGBUF_DIST|CONT_CONV);
399                     ADC_SEND(WRITE_TO_REG|FILTER_REG,SF_255);
400                     ADC_SEND(WRITE_TO_REG|ADCCON_REG,AIN3C|BIPOL|MIV2560);
401                     pomiark3 = AD7708_READ_CH();
402
403 //AIN4C
404
405             SPI_MasterInit();
406                     ADC_SEND(WRITE_TO_REG|MODE_REG,CHOP_EN|NEGBUF_DIST|CONT_CONV);
407                     ADC_SEND(WRITE_TO_REG|FILTER_REG,SF_255);
408                     ADC_SEND(WRITE_TO_REG|ADCCON_REG,AIN4C|BIPOL|MIV2560);
409                     pomiark4 = AD7708_READ_CH();
410
411 //AIN5C
412
413             SPI_MasterInit();
414                     ADC_SEND(WRITE_TO_REG|MODE_REG,CHOP_EN|NEGBUF_DIST|CONT_CONV);
415                     ADC_SEND(WRITE_TO_REG|FILTER_REG,SF_255);
416                     ADC_SEND(WRITE_TO_REG|ADCCON_REG,AIN5C|BIPOL|MIV2560);
417                     pomiark5 = AD7708_READ_CH();
418
419 //AIN6C
420
421             SPI_MasterInit();
422                     ADC_SEND(WRITE_TO_REG|MODE_REG,CHOP_EN|NEGBUF_DIST|CONT_CONV);
423                     ADC_SEND(WRITE_TO_REG|FILTER_REG,SF_255);
424                     ADC_SEND(WRITE_TO_REG|ADCCON_REG,AIN6C|BIPOL|MIV2560);
425                     pomiark6 = AD7708_READ_CH();
426
427 //AIN7C
428
429             SPI_MasterInit();
430                     ADC_SEND(WRITE_TO_REG|MODE_REG,CHOP_EN|NEGBUF_DIST|CONT_CONV);
431                     ADC_SEND(WRITE_TO_REG|FILTER_REG,SF_255);
432                     ADC_SEND(WRITE_TO_REG|ADCCON_REG,AIN7C|BIPOL|MIV2560);
433                     pomiark7 = AD7708_READ_CH();
```

```
434
435 //Obróbka wyników
436
437                 dtostrf((((pomiark1-32768)/32768.00000)*2560),6,2,pom1);
438                 dtostrf((((pomiark2-32768)/32768.00000)*2560),6,2,pom2);
439                 dtostrf((((pomiark3-32768)/32768.00000)*2560),6,2,pom3);
440                 dtostrf((((pomiark4-32768)/32768.00000)*2560),6,2,pom4);
441                 dtostrf((((pomiark5-32768)/32768.00000)*2560),6,2,pom5);
442                 dtostrf((((pomiark6-32768)/32768.00000)*2560),6,2,pom6);
443                 dtostrf((((pomiark7-32768)/32768.00000)*2560),6,2,pom7);
444
445 sprintf(seria, " %smV  %smV  %smV  %smV  %smV  %smV %smV\n",
446 pom1, pom2, pom3, pom4, pom5, pom6, pom7);
447
448 //Zapis pomiarów do pliku na karcie SD
449                 f_printf(&plik, seria);
450
451
452                 // Wyświetlanie wyników on-line na LCD
453                 LCD_CLEAR;
454                 LCD_LOCATE(0,0);
455                 lcd_puts("Trwa zapis na SD");
456                 LCD_LOCATE(0,1);
457                 lcd_puts("ESC");
458         }
459
460         LCD_CLEAR;
461
462 //Zamknięcie pliku i odmontowanie dysku
463         f_close(&plik);
464         f_mount(0, NULL);
465
466         while(!S4)
467         {
468         _delay_ms(80);
469         while(S4);
470         LCD_LOCATE(0,0);
471    lcd_puts("Zapis zakonczony");
472         }
473  // KONIEC CZYNNOŚCI SD
474  }
475  _delay_ms(80);
476  while(S4);
477  MENU=0;
```

```
478  LCD_CLEAR;
479  }
480
481  if(MENU==321)     // ONLINE
482  {
483
484  while(!S4)
485  {
486  // Czynności ONLINE ****
487
488 //AIN1C
489
490           SPI_MasterInit();
491                    ADC_SEND(WRITE_TO_REG|MODE_REG,CHOP_EN|NEGBUF_DIST|CONT_CONV);
492                    ADC_SEND(WRITE_TO_REG|FILTER_REG,SF_255);
493                    ADC_SEND(WRITE_TO_REG|ADCCON_REG,AIN1C|BIPOL|MIV2560);
494                    pomiark1 = AD7708_READ_CH();
495
496 //AIN2C
497
498           SPI_MasterInit();
499                    ADC_SEND(WRITE_TO_REG|MODE_REG,CHOP_EN|NEGBUF_DIST|CONT_CONV);
500                    ADC_SEND(WRITE_TO_REG|FILTER_REG,SF_255);
501                    ADC_SEND(WRITE_TO_REG|ADCCON_REG,AIN2C|BIPOL|MIV2560);
502                    pomiark2 = AD7708_READ_CH();
503
504 //AIN3C
505
506           SPI_MasterInit();
507                    ADC_SEND(WRITE_TO_REG|MODE_REG,CHOP_EN|NEGBUF_DIST|CONT_CONV);
508                    ADC_SEND(WRITE_TO_REG|FILTER_REG,SF_255);
509                    ADC_SEND(WRITE_TO_REG|ADCCON_REG,AIN3C|BIPOL|MIV2560);
510                    pomiark3 = AD7708_READ_CH();
511
512 //AIN4C
513
514           SPI_MasterInit();
515                    ADC_SEND(WRITE_TO_REG|MODE_REG,CHOP_EN|NEGBUF_DIST|CONT_CONV);
516                    ADC_SEND(WRITE_TO_REG|FILTER_REG,SF_255);
517                    ADC_SEND(WRITE_TO_REG|ADCCON_REG,AIN4C|BIPOL|MIV2560);
518                    pomiark4 = AD7708_READ_CH();
519
520 //AIN5C
521
```

```
522             SPI_MasterInit();
523                     ADC_SEND(WRITE_TO_REG|MODE_REG,CHOP_EN|NEGBUF_DIST|CONT_CONV);
524                     ADC_SEND(WRITE_TO_REG|FILTER_REG,SF_255);
525                     ADC_SEND(WRITE_TO_REG|ADCCON_REG,AIN5C|BIPOL|MIV2560);
526                     pomiark5 = AD7708_READ_CH();
527
528 //AIN6C
529
530             SPI_MasterInit();
531                     ADC_SEND(WRITE_TO_REG|MODE_REG,CHOP_EN|NEGBUF_DIST|CONT_CONV);
532                     ADC_SEND(WRITE_TO_REG|FILTER_REG,SF_255);
533                     ADC_SEND(WRITE_TO_REG|ADCCON_REG,AIN6C|BIPOL|MIV2560);
534                     pomiark6 = AD7708_READ_CH();
535
536 //AIN7C
537
538             SPI_MasterInit();
539                     ADC_SEND(WRITE_TO_REG|MODE_REG,CHOP_EN|NEGBUF_DIST|CONT_CONV);
540                     ADC_SEND(WRITE_TO_REG|FILTER_REG,SF_255);
541                     ADC_SEND(WRITE_TO_REG|ADCCON_REG,AIN7C|BIPOL|MIV2560);
542                     pomiark7 = AD7708_READ_CH();
543
544 //Obróbka wyników
545
546                     dtostrf((((pomiark1-32768)/32768.00000)*2560),6,2,pom1);
547                     dtostrf((((pomiark2-32768)/32768.00000)*2560),6,2,pom2);
548                     dtostrf((((pomiark3-32768)/32768.00000)*2560),6,2,pom3);
549                     dtostrf((((pomiark4-32768)/32768.00000)*2560),6,2,pom4);
550                     dtostrf((((pomiark5-32768)/32768.00000)*2560),6,2,pom5);
551                     dtostrf((((pomiark6-32768)/32768.00000)*2560),6,2,pom6);
552                     dtostrf((((pomiark7-32768)/32768.00000)*2560),6,2,pom7);
553
554                     // Wyświetlanie wyników on-line na LCD
555                     LCD_CLEAR;
556                     LCD_LOCATE(0,0);
557                     sprintf(onln, "%s  %s", pom1, pom2);
558                     lcd_puts(onln);
559                     LCD_LOCATE(0,1);
560                     sprintf(onln, "%s  %s", pom3, pom4);
561                     lcd_puts(onln);
562                     _delay_ms(800);
563                     LCD_CLEAR;
564                     LCD_LOCATE(0,0);
565                     sprintf(onln, "%s  %s", pom5, pom6);
```

```
566                     lcd_puts(onln);
567                     LCD_LOCATE(0,1);
568                     sprintf(onln, " %s", pom7);
569                     lcd_puts(onln);
570                     _delay_ms(800);
571
572  // KONIEC CZYNNOŚCI ONLINE ***
573  }
574  _delay_ms(80);
575  while(S4);
576  MENU=0;
577  LCD_CLEAR;
578  }
579
580  if(MENU==323)    // PC
581  {
582          LCD_CLEAR;
583          LCD_LOCATE(0,0);
584          lcd_puts("Polaczenie z PC");
585          LCD_LOCATE(0,1);
586          lcd_puts("Gdy END wylacz");
587
588   /* Inicjalizacja USART 8 bitów danych, dwa bity stopu */
589    USART_init(MYUBRR);
590
591    /* Inicjalizacja SPI - w trybie Master */
592    SPI_MasterInit();
593
594    USART_puts("Wyslij znak by otrzymac wyniki pomiarow. By zakonczyc rozlacz transmisje
595    i odlacz zasilanie");
596    _delay_ms(1000);
597
598  while(!S4)
599  {
600  // Czynności PC
601
602    while(!(USART_Receive()=="P"))
603    {
604 //AIN1C
605
606          SPI_MasterInit();
607                  ADC_SEND(WRITE_TO_REG|MODE_REG,CHOP_EN|NEGBUF_DIST|CONT_CONV);
608                  ADC_SEND(WRITE_TO_REG|FILTER_REG,SF_255);
609                  ADC_SEND(WRITE_TO_REG|ADCCON_REG,AIN1C|BIPOL|MIV2560);
```

```
610                     pomiark1 = AD7708_READ_CH();
611
612 //AIN2C
613
614             SPI_MasterInit();
615                     ADC_SEND(WRITE_TO_REG|MODE_REG,CHOP_EN|NEGBUF_DIST|CONT_CONV);
616                     ADC_SEND(WRITE_TO_REG|FILTER_REG,SF_255);
617                     ADC_SEND(WRITE_TO_REG|ADCCON_REG,AIN2C|BIPOL|MIV2560);
618                     pomiark2 = AD7708_READ_CH();
619
620 //AIN3C
621
622             SPI_MasterInit();
623                     ADC_SEND(WRITE_TO_REG|MODE_REG,CHOP_EN|NEGBUF_DIST|CONT_CONV);
624                     ADC_SEND(WRITE_TO_REG|FILTER_REG,SF_255);
625                     ADC_SEND(WRITE_TO_REG|ADCCON_REG,AIN3C|BIPOL|MIV2560);
626                     pomiark3 = AD7708_READ_CH();
627
628 //AIN4C
629
630             SPI_MasterInit();
631                     ADC_SEND(WRITE_TO_REG|MODE_REG,CHOP_EN|NEGBUF_DIST|CONT_CONV);
632                     ADC_SEND(WRITE_TO_REG|FILTER_REG,SF_255);
633                     ADC_SEND(WRITE_TO_REG|ADCCON_REG,AIN4C|BIPOL|MIV2560);
634                     pomiark4 = AD7708_READ_CH();
635
636 //AIN5C
637
638             SPI_MasterInit();
639                     ADC_SEND(WRITE_TO_REG|MODE_REG,CHOP_EN|NEGBUF_DIST|CONT_CONV);
640                     ADC_SEND(WRITE_TO_REG|FILTER_REG,SF_255);
641                     ADC_SEND(WRITE_TO_REG|ADCCON_REG,AIN5C|BIPOL|MIV2560);
642                     pomiark5 = AD7708_READ_CH();
643
644 //AIN6C
645
646             SPI_MasterInit();
647                     ADC_SEND(WRITE_TO_REG|MODE_REG,CHOP_EN|NEGBUF_DIST|CONT_CONV);
648                     ADC_SEND(WRITE_TO_REG|FILTER_REG,SF_255);
649                     ADC_SEND(WRITE_TO_REG|ADCCON_REG,AIN6C|BIPOL|MIV2560);
650                     pomiark6 = AD7708_READ_CH();
651
652 //AIN7C
653
```

```
654           SPI_MasterInit();
655                   ADC_SEND(WRITE_TO_REG|MODE_REG,CHOP_EN|NEGBUF_DIST|CONT_CONV);
656                   ADC_SEND(WRITE_TO_REG|FILTER_REG,SF_255);
657                   ADC_SEND(WRITE_TO_REG|ADCCON_REG,AIN7C|BIPOL|MIV2560);
658                   pomiark7 = AD7708_READ_CH();
659
660 //Obróbka wyników
661
662                   dtostrf((((pomiark1-32768)/32768.00000)*2560),6,2,pom1);
663                   dtostrf((((pomiark2-32768)/32768.00000)*2560),6,2,pom2);
664                   dtostrf((((pomiark3-32768)/32768.00000)*2560),6,2,pom3);
665                   dtostrf((((pomiark4-32768)/32768.00000)*2560),6,2,pom4);
666                   dtostrf((((pomiark5-32768)/32768.00000)*2560),6,2,pom5);
667                   dtostrf((((pomiark6-32768)/32768.00000)*2560),6,2,pom6);
668                   dtostrf((((pomiark7-32768)/32768.00000)*2560),6,2,pom7);
669
670    sprintf(seria, " %smV  %smV  %smV  %smV  %smV  %smV %smV\n",
671    pom1, pom2, pom3, pom4, pom5, pom6, pom7);
672
673     USART_puts(seria);
674     }
675    }
676   _delay_ms(80);
677   while(S4);
678   MENU=0;
679   LCD_CLEAR;
680   }
681   }
682 }
```

```
1  /* Procedury obsługi przetwornika A/C */
2
3  #include<avr/io.h>
4  #include "AD7708.h"
5  #include "spi.h"
6  #include <util/delay.h>
7  #include "hd44780.h"
8
9  void AD7708_Init(void)
10          {
11          CLR_AD_CS;      // chip select wyłączony
12          /* Ustawia RESET i CS jako wyścia ATMEGI, pozostałe w DDRA bez zmian */
13          DDR_AD_OUTA |= (1<<AD_RESET)|(1<<AD_CS);
14          /* Ustawia RDY jako weście ATMEGI, pozostałe w DDRD bez zmian */
```

```
15          DDR_AD_OUTD &= ~(1<<AD_RDY);
16
17          /* Tryb Power-ON reset */
18          SET_AD_CS;
19          SET_AD_RESET; // reset ustawiony
20          _delay_ms(100);
21          CLR_AD_RESET; // reset wyłączony, urządzenie w trybie Power-ON reset
22          _delay_ms(400); //czas włączania ADC
23
24          /* Inicjalizacja SPI */
25          SPI_MasterInit();
26
27          //Filtr SF (odświeżanie ADC)
28          ADC_SEND(WRITE_TO_REG|FILTER_REG,SF_255);
29
30          //Kanał AIN1 - AIN2, tryb Bipolarny, zakres +/- 2,56V
31          ADC_SEND(WRITE_TO_REG|ADCCON_REG,AIN12|BIPOL|MIV2560);
32
33          //CHOP aktywny, tryb pracy ciągłej
34          ADC_SEND(WRITE_TO_REG|MODE_REG,CHOP_EN|NEGBUF_DIST|CONT_CONV);
35          CLR_AD_CS;
36          }
37
38  unsigned short AD7708_READ_CH(void)
39          {
40          return ADC_RECEIVE16(READ_FROM_REG|ADCDATA_REG);
41          }
42
43  void ADC_SEND(unsigned char reg, unsigned char cmd)
44          {
45          SET_AD_CS;
46          SPI_MasterTransmit_AD(reg);
47          SPI_MasterTransmit_AD(cmd);
48          CLR_AD_CS;
49          }
50
51  unsigned char ADC_RECEIVE8(unsigned char reg)
52          {
53          SET_AD_CS;
54          SPI_MasterTransmit_AD(reg);
55          return SPI_MasterReceive_AD8();
56          CLR_AD_CS;
57          }
58
```

```
59 unsigned short ADC_RECEIVE16(unsigned char reg)
60         {
61         SET_AD_CS;
62         SPI_MasterTransmit_AD(reg);
63         return SPI_MasterReceive_AD16();
64         CLR_AD_CS;
65         }
66
67 void AD7708_Cal(void)
68         {
69 /*Kanał AIN1 - AIN2*/
70
71 SET_AD_CS;
72 ADC_SEND(WRITE_TO_REG|ADCCON_REG,AIN12|BIPOL|MIV2560); // Kanał, tryb, zakres
73 ADC_SEND(WRITE_TO_REG|MODE_REG,INT_ZS_CAL);
74 while(!((0x07&(ADC_RECEIVE8(READ_FROM_REG|MODE_REG)))==0x01)); //jeżeli MD Bits różne od 001 czekaj
75 ADC_SEND(WRITE_TO_REG|MODE_REG,INT_FS_CAL);
76 while(!((0x07&(ADC_RECEIVE8(READ_FROM_REG|MODE_REG)))==0x01)); //jeżeli MD Bits różne od 001 czekaj
77
78 /*Kanał AIN3 - AIN4*/
79 ADC_SEND(WRITE_TO_REG|ADCCON_REG,AIN34|BIPOL|MIV2560); // Kanał, tryb, zakres
80 ADC_SEND(WRITE_TO_REG|MODE_REG,INT_ZS_CAL);
81 while(!((0x07&(ADC_RECEIVE8(READ_FROM_REG|MODE_REG)))==0x01)); //jeżeli MD Bits różne od 001 czekaj
82 ADC_SEND(WRITE_TO_REG|MODE_REG,INT_FS_CAL);
83 while(!((0x07&(ADC_RECEIVE8(READ_FROM_REG|MODE_REG)))==0x01)); //jeżeli MD Bits różne od 001 czekaj
84
85 /*Kanał AIN5 - AIN6*/
86 ADC_SEND(WRITE_TO_REG|ADCCON_REG,AIN56|BIPOL|MIV2560); // Kanał, tryb, zakres
87 ADC_SEND(WRITE_TO_REG|MODE_REG,INT_ZS_CAL);
88 while(!((0x07&(ADC_RECEIVE8(READ_FROM_REG|MODE_REG)))==0x01)); //jeżeli MD Bits różne od 001 czekaj
89 ADC_SEND(WRITE_TO_REG|MODE_REG,INT_FS_CAL);
90 while(!((0x07&(ADC_RECEIVE8(READ_FROM_REG|MODE_REG)))==0x01)); //jeżeli MD Bits różne od 001 czekaj
91
92 /*Kanał AIN7 - AIN8*/
93 ADC_SEND(WRITE_TO_REG|ADCCON_REG,AIN78|BIPOL|MIV2560); // Kanał, tryb, zakres
94 ADC_SEND(WRITE_TO_REG|MODE_REG,INT_ZS_CAL);
95 while(!((0x07&(ADC_RECEIVE8(READ_FROM_REG|MODE_REG)))==0x01)); //jeżeli MD Bits różne od 001 czekaj
96 ADC_SEND(WRITE_TO_REG|MODE_REG,INT_FS_CAL);
97 while(!((0x07&(ADC_RECEIVE8(READ_FROM_REG|MODE_REG)))==0x01)); //jeżeli MD Bits różne od 001 czekaj
98 CLR_AD_CS;
99 }
```

```
1 /*
2 Plik AD7708.h
```

```
3  */
4  #ifndef AD7708
5  #define AD7708
6
7  #define ADC_NOP asm volatile("nop\n\t""nop\n\t" "nop\n\t" "nop\n\t" ::);
8
9  #define DDR_AD_OUTA DDRA
10 #define AD_RESET DDA1
11 #define AD_CS   DDA0
12 #define DDR_AD_OUTD DDRD
13 #define AD_RDY   DDD0
14
15 /* AD_RESET */
16 #define SET_AD_RESET PORTA &=  ~_BV(PA1)
17 #define CLR_AD_RESET PORTA |=   _BV(PA1)
18
19 /* AD_CS */
20 #define SET_AD_CS PORTA &=  ~_BV(PA0)
21 #define CLR_AD_CS PORTA |=   _BV(PA0)
22
23 /* ADC_RDY */
24 #define ACD_NOT_RDY        PIND  &  _BV(PD0)
25
26 /* Komendy do rejestru komunikacji (COMMUNICATIONS REGISTER) */
27 #define WRITE_TO_REG  0x00 //ZAPIS
28 #define READ_FROM_REG 0x40 //ODCZYT
29
30 /* Adresy rejestrów AD7708 */
31 #define STATUS_REG   0x00
32 #define MODE_REG     0x01
33 #define ADCCON_REG   0x02
34 #define FILTER_REG   0x03
35 #define ADCDATA_REG  0x04
36 #define ADCOFF_REG   0x05
37 #define ADCGAIN_REG  0x06
38 #define ADCIO_REG    0x07
39 #define ADCT1_REG    0x0C
40 #define ADCT2_REG    0x0D
41 #define ADCID_REG    0x0F
42
43 /*** Komendy ustawiające poszczególne rejestry AD7708 ***/
44
45 /* FILTER REGISTER (FILTER_REG) */
46 #define SF_13 0x0D
```

```
47 #define SF_69 0x45  //odświeżanie ADC z f=19,79 Hz
48 #define SF_255 0xFF
49
50 /* CONTROL REGISTER (ADCCON_REG) Wywołanie (Kanał+Tryb+Zakres) */
51
52 //Kanał pomiędzy wejściami
53
54 #define AIN1C 0x00
55 #define AIN2C 0x10
56 #define AIN3C 0x20
57 #define AIN4C 0x30
58 #define AIN5C 0x40
59 #define AIN6C 0x50
60 #define AIN7C 0x60
61 #define AIN8C 0x70
62 #define AIN12 0x80
63 #define AIN34 0x90
64 #define AIN56 0xA0
65 #define AIN78 0xB0
66
67 //Tryb
68 #define UNIPOL 0x08
69 #define BIPOL  0x00
70
71 //Zakres
72 #define MIV20     0x00
73 #define MIV40     0x01
74 #define MIV80     0x02
75 #define MIV160    0x03
76 #define MIV320    0x04
77 #define MIV640    0x05
78 #define MIV1280   0x06
79 #define MIV2560   0x07
80
81 /* MODE REGISTER (MODE_REG) Wywołanie (CHOP+NEGBUF+TRYB PRACY) */
82
83 // CHOP aktywny/nieaktywny
84 #define CHOP_EN    0x00
85 #define CHOP_DIST  0x80
86
87 // NEGBUF aktywny/nieaktywny
88 #define NEGBUF_DIST    0x00
89 #define NEGBUF_EN      0x40
90
```

```
91  // TRYB PRACY ADC
92  #define PD_MODE          0x00
93  #define IDLE_MODE        0x01
94  #define SING_CONV        0x02
95  #define CONT_CONV        0x03
96  #define INT_ZS_CAL       0x04
97  #define INT_FS_CAL       0x05
98  #define SYS_ZS_CAL       0x06
99  #define SYS_FS_CAL       0x07
100
101 void AD7708_Init(void);
102 void ADC_SEND(unsigned char reg, unsigned char cmd);
103 unsigned short AD7708_READ_CH(void);
104 unsigned short ADC_RECEIVE16(unsigned char reg);
105 unsigned char ADC_RECEIVE8(unsigned char reg);
106 void AD7708_Cal(void);
107 #endif
```

```
1  /* Procedury obsługi LCD */
2
3  #include<avr/io.h>
4  #include<util/delay.h>
5  #include "hd44780.h"
6
7  /*-------------------------------------------------------*/
8  /* Zapis danej lub instrukcji */
9
10 void WriteToLCD (unsigned char v,unsigned char  rs)
11 {
12      unsigned char bf;
13      LCD_DATA = v;
14
15      CLR_LCD_E;
16      if(rs) SET_LCD_RS;else CLR_LCD_RS;
17      CLR_LCD_RW;
18
19      LCD_NOP;
20      SET_LCD_E;
21      LCD_NOP;
22      CLR_LCD_E;
23      LCD_NOP;
24
25      CLR_LCD_RS;
26      //SET_LCD_RW;
```

```
27
28      SET_BUF_OE;
29      SET_LCD_RW;
30      CLR_LCD_RS;
31  do
32  {
33      LCD_NOP;
34      SET_LCD_E;
35      LCD_NOP;
36      CLR_LCD_E;
37      LCD_NOP;
38      bf = LCD_BUSY_F;
39      LCD_NOP;
40
41  }while(bf);
42
43      CLR_BUF_OE;
44  }
45
46  /*-------------------------------------------------------------*/
47  /* Inicjalizacja wyświetlacza */
48
49  void lcd_init(void)
50  {
51
52      SET_MCP_SHDN_n; //włączenie charge pump
53      LCD_NOP;
54      LCD_NOP;
55      SET_5V_LCD_EN; //włączenie napięcia dla LCD
56      _delay_ms(45);
57      WriteToLCD (0x38 , LCDCOMMAND) ;  //tryb 8 bitowy
58      _delay_ms(5);
59      WriteToLCD (0x38 , LCDCOMMAND) ;
60  // Można sprawdzać Busy Flag
61      LCD_DISP_ON;
62      LCD_CLEAR ;
63      LCD_ENTRY_MODE(LCDINCREMENT) ;
64          LCD_CLEAR;
65  }
66
67  /*-------------------------------------------------------------*/
68  /* Wyswietla tekst na aktualnej pozycji kursora */
69
70  void lcd_puts(char *str)
```

```
71 {
72     unsigned char i =0;
73     while(str[i])
74         LCD_WRITE_DATA(str[i++]);
75 }
```

```
1  /*
2  Plik hd44780.h
3  */
4
5  /************************************************************************
6  LCD_LOCATE(x,y); - współrzędne wyświetlacza (kolumna 0-15, wiersz 0-1)
7  lcd_puts("NAPIS"); - wyświetla napis na wyświetlaczu
8  ************************************************************************/
9
10 #ifndef LCD_HD44780
11 #define LCD_HD44780
12 #define LCD_DATA PORTC
13
14 /* RS */
15 #define SET_OUT_LCD_RS  DDRB  |=  _BV(PB4)
16 #define SET_LCD_RS      PORTB |=  _BV(PB4)
17 #define CLR_LCD_RS      PORTB &= ~_BV(PB4)
18
19 /* RW */
20 #define SET_OUT_LCD_RW  DDRB  |=  _BV(PB5)
21 #define SET_LCD_RW      PORTB |=  _BV(PB5)
22 #define CLR_LCD_RW      PORTB &= ~_BV(PB5)
23
24 /* E */
25 #define SET_OUT_LCD_E   DDRB  |=  _BV(PB6)
26 #define SET_LCD_E       PORTB |=  _BV(PB6)
27 #define CLR_LCD_E       PORTB &= ~_BV(PB6)
28
29 /* D0 */
30 #define SET_OUT_LCD_D0  DDRC  |=  _BV(PC0)
31 #define SET_LCD_D0      PORTC |=  _BV(PC0)
32 #define CLR_LCD_D0      PORTC &= ~_BV(PC0)
33 #define IS_SET_LCD_D0   PINC  &   _BV(PC0)
34
35 /* D1 */
36 #define SET_OUT_LCD_D1  DDRC  |=  _BV(PC1)
37 #define SET_LCD_D1      PORTC |=  _BV(PC1)
38 #define CLR_LCD_D1      PORTC &= ~_BV(PC1)
```

```
39
40 /* D2 */
41 #define SET_OUT_LCD_D2  DDRC  |=  _BV(PC2)
42 #define SET_LCD_D2      PORTC |=  _BV(PC2)
43 #define CLR_LCD_D2      PORTC &= ~_BV(PC2)
44
45 /* D3 */
46 #define SET_OUT_LCD_D3  DDRC  |=  _BV(PC3)
47 #define SET_LCD_D3      PORTC |=  _BV(PC3)
48 #define CLR_LCD_D3      PORTC &= ~_BV(PC3)
49
50 /* D4 */
51 #define SET_OUT_LCD_D4  DDRC  |=  _BV(PC4)
52 #define SET_LCD_D4      PORTC |=  _BV(PC4)
53 #define CLR_LCD_D4      PORTC &= ~_BV(PC4)
54
55 /* D5 */
56 #define SET_OUT_LCD_D5  DDRC  |=  _BV(PC5)
57 #define SET_LCD_D5      PORTC |=  _BV(PC5)
58 #define CLR_LCD_D5      PORTC &= ~_BV(PC5)
59
60 /* D6 */
61 #define SET_OUT_LCD_D6  DDRC  |=  _BV(PC6)
62 #define SET_LCD_D6      PORTC |=  _BV(PC6)
63 #define CLR_LCD_D6      PORTC &= ~_BV(PC6)
64
65 /* D7 */
66 #define SET_OUT_LCD_D7  DDRC  |=  _BV(PC7)
67 #define SET_LCD_D7      PORTC |=  _BV(PC7)
68 #define CLR_LCD_D7      PORTC &= ~_BV(PC7)
69 #define IS_SET_LCD_D7   PINC  &   _BV(PC7)
70
71 /* BUF_OE_n */
72 #define SET_OUT_BUF_OE  DDRB  |=  _BV(PB7)
73 #define SET_BUF_OE      PORTB |=  _BV(PB7)
74 #define CLR_BUF_OE      PORTB &= ~_BV(PB7)
75
76 /* LCD_BUSY */
77 #define SET_IN_BUSY_FL  DDRE  &= ~_BV(PE3)
78 #define LCD_BUSY_F      PINE  &   _BV(PE3)  //PINE  &  0x08
79
80 /* MCP_SHDN_n */
81 #define SET_MCP_SHDN_n      PORTA |=  _BV(PA7)
82 #define CLR_MCP_SHDN_n      PORTA &= ~_BV(PA7)
```

```
83
84  /* 5V_LCD_EN */
85  #define SET_5V_LCD_EN       PORTE |=  _BV(PE2)
86  #define CLR_5V_LCD_EN       PORTE &= ~_BV(PE2)
87
88  #define LCD_NOP asm volatile("nop\n\t""nop\n\t" "nop\n\t" "nop\n\t" ::);
89
90  #define LCDCOMMAND 0
91  #define LCDDATA    1
92
93  #define LCD_LOCATE(x,y)             WriteToLCD(0x80|((x)+((y)*0x40)), LCDCOMMAND)
94  #define LCD_CLEAR                   WriteToLCD(0x01, LCDCOMMAND)
95  #define LCD_HOME                    WriteToLCD(0x02, LCDCOMMAND)
96  #define LCD_DISP_ON                 WriteToLCD(0x0C, LCDCOMMAND)
97  #define LCD_DISP_OFF                WriteToLCD(0x08, LCDCOMMAND)
98
99  /* IDS */
100 #define LCDINCREMENT                0x00
101 #define LCDDECREMENT                0x02
102 #define LCDDISPLAYSHIFT             0x01
103
104 #define LCD_ENTRY_MODE(IDS)         WriteToLCD(0x04|(IDS), LCDCOMMAND)
105
106 /* BCD */
107 #define LCDDISPLAY                  0x04
108 #define LCDCURSOR                   0x02
109 #define LCDBLINK                    0x01
110
111 #define LCD_DISPLAY(DCB)            WriteToLCD(0x08|(DCB), LCDCOMMAND)
112
113 /* RL */
114 #define LCDLEFT                     0x00
115 #define LCDRIGHT                    0x04
116
117 #define LCD_SHIFT_DISPLAY(RL)       WriteToLCD(0x18|(RL), LCDCOMMAND)
118 #define LCD_SHIFT_CURSOR(RL)        WriteToLCD(0x10|(RL), LCDCOMMAND)
119
120 #define LCD_WRITE_DATA(D)           WriteToLCD((D),LCDDATA)
121
122
123 void lcd_init(void);
124 void WriteToLCD(unsigned char v,unsigned char rs);
125 void lcd_puts(char *str);
126
```

```
127 #endif

1  /* Procedury obsługi interfejsu SPI */
2
3  #include<avr/io.h>
4  #include "spi.h"
5  #include "AD7708.h"
6
7          void SPI_MasterInit(void)
8          {
9          /* Ustawia MOSI, SCK i SS jako wyścia, pozostałe w DDRB bez zmian */
10         DDR_SPI |= (1<<DD_MOSI)|(1<<DD_SCK)|(1<<DD_SS);
11         /* Aktywuje SPI, Tryb Master, stan wysoki gdy SCK nieaktywne,
12         częstotliwość sygnału zegarowego fck/16 */
13         SPCR = (1<<SPE)|(1<<MSTR)|(1<<CPOL)|(1<<CPHA)|(1<<SPR1)|(1<<SPR0);
14         SPSR |= (1<<SPI2X);   //SPSR &= ~(1<<SPI2X);
15         }
16
17         void SPI_MasterInit_SD(void)
18         {
19         /* Ustawia MOSI, SCK i SS jako wyścia, pozostałe w DDRB bez zmian */
20         DDR_SPI |= (1<<DD_MOSI)|(1<<DD_SCK)|(1<<DD_SS);
21         /* Aktywuje SPI, Tryb Master, częstotliwość sygnału zegarowego SPI2X */
22         SPCR = (1<<SPE)|(1<<MSTR)|(1<<SPR1)|(1<<SPR0);
23         SPSR |= (1<<SPI2X);
24         }
25
26
27         void SPI_MasterTransmit_AD(unsigned char cData)
28         {
29         /* Początek transmisji */
30         SPDR = cData;
31         /* Czekaj na zakończenie transmisji */
32         while(!(SPSR & (1<<SPIF)));
33         }
34
35         unsigned short SPI_MasterReceive_AD16(void)
36         {
37         unsigned short data_read;
38
39         while(ACD_NOT_RDY); // Jeżeli nie ma danych w Mode Reg - czekaj
40         SPDR = 0x00; //Aktywacja zegara dla odczytu
41         /* Poczekaj na koniec transmisji */
42         while(!(SPSR & (1<<SPIF)));
```

```
43          data_read = SPDR<<8;  //zapis danych (pierwsze 8 bitów)
44
45          SPDR = 0x00; //Aktywacja zegara dla odczytu
46          /* Poczekaj na koniec transmisji */
47          while(!(SPSR & (1<<SPIF)));
48          data_read |= SPDR;  //zapis danych (drugie 8 bitów)
49          /* Zwróć wartość rejestru SPDR */
50          return data_read;
51          }
52
53          unsigned char SPI_MasterReceive_AD8(void)
54          {
55          unsigned char data_read8;
56          SPDR = 0x00; //Aktywacja zegara dla odczytu
57          /* Poczekaj na koniec transmisji */
58          while(!(SPSR & (1<<SPIF)));
59          data_read8 = SPDR; //zapis danych (8 bitów)
60          /* Zwróć wartość rejestru SPDR */
61          return data_read8;
62          }
```

```
1  /*
2  Plik spi.h
3  */
4  #ifndef T_SPI
5  #define T_SPI
6
7  #define DD_MOSI DDB2
8  #define DD_SCK  DDB1
9  #define DD_SS   DDB0
10 #define DDR_SPI DDRB
11
12
13 void SPI_MasterInit(void);
14 void SPI_MasterTransmit_AD(unsigned char cData);
15 unsigned short SPI_MasterReceive_AD16(void);
16 unsigned char SPI_MasterReceive_AD8(void);
17 void SPI_MasterInit_SD(void);
18
19 #endif
```

```
1  /* Procedury obsługi USART */
2
3  #include<avr/io.h>
```

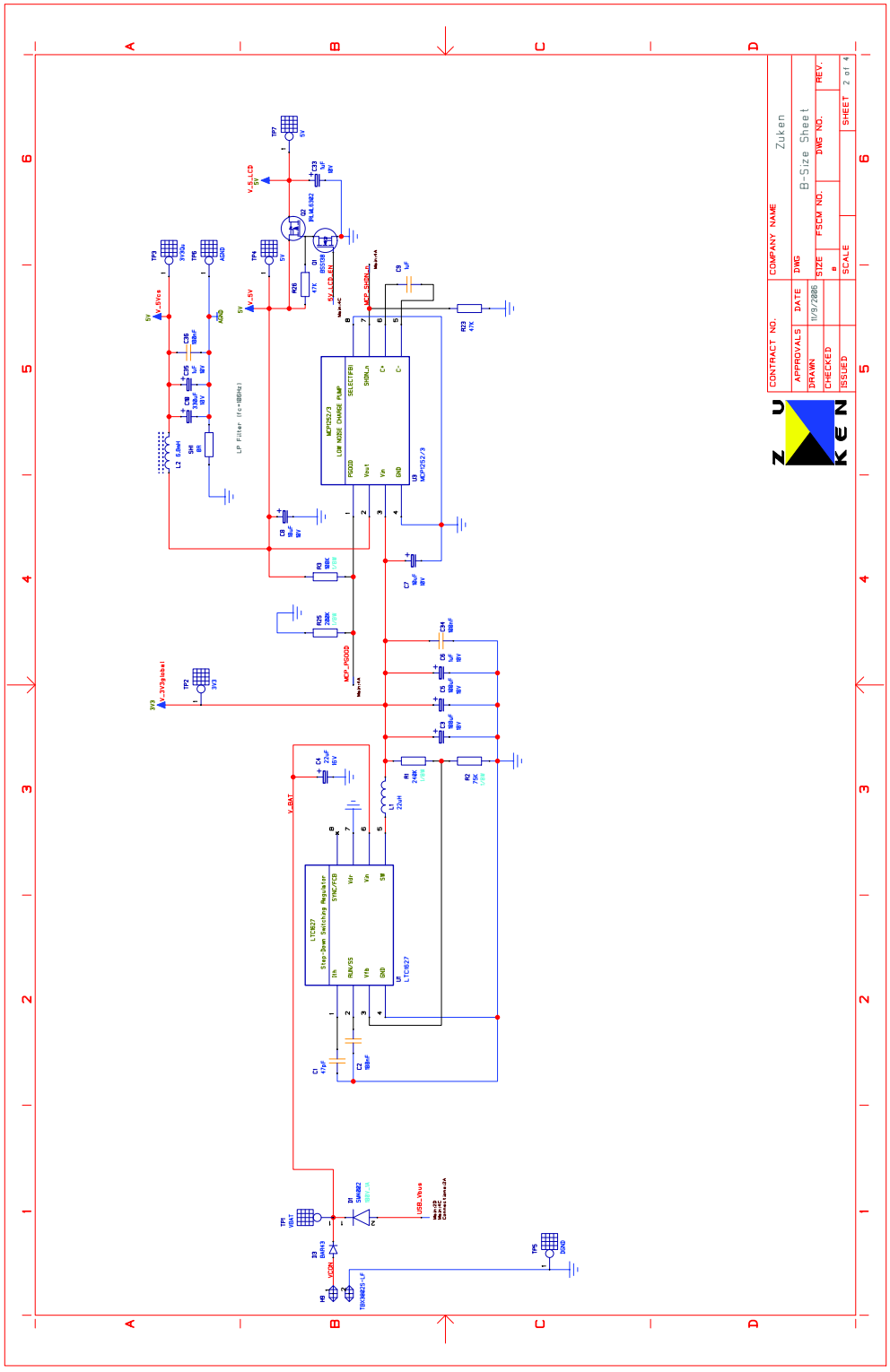
```
4  #include "usart.h"
5
6      /* Inicjuje port szeregowy AVRa */
7      void USART_init(unsigned int myubrr)
8      {
9       /* Ustala prędkość transmisji */
10      UBRR1H = (unsigned char)(myubrr>>8);
11      UBRR1L = (unsigned char)myubrr;
12
13      /* Włącza odbiornik i nadajnik */
14      UCSR1B = (1<<RXEN)|(1<<TXEN);
15
16      /* format ramki 8 bitów danych, dwa bity stopu, brak bitu parzystości */
17      UCSR1C = (1<<USBS)|(3<<UCSZ0);
18     }
19
20     /* Wyślij znak do portu szeregowego  */
21     void USART_Transmit(unsigned char data)
22     {
23     /* Poczekaj aż bufor nadajnika będzie pusty */
24     while ( !( UCSR1A & (1<<UDRE)) );
25
26     /* Wyślij dane do na rejestr wychodzący UDR */
27     UDR1 = data;
28     }
29
30        /* Ciąg znaków do portu szeregowego  */
31     void USART_puts(char *str)
32     {
33      unsigned char i =0;
34      while(str[i])
35           USART_Transmit(str[i++]);
36     }
37
38        /* Odbierz znak z portu szeregowego */
39     unsigned char USART_Receive(void)
40     {
41      /* Poczekaj na dane */
42      while ( !(UCSR1A & (1<<RXC)) );
43      /* Odbierz dane z bufora odbiornika */
44      return UDR1;
45     }
```

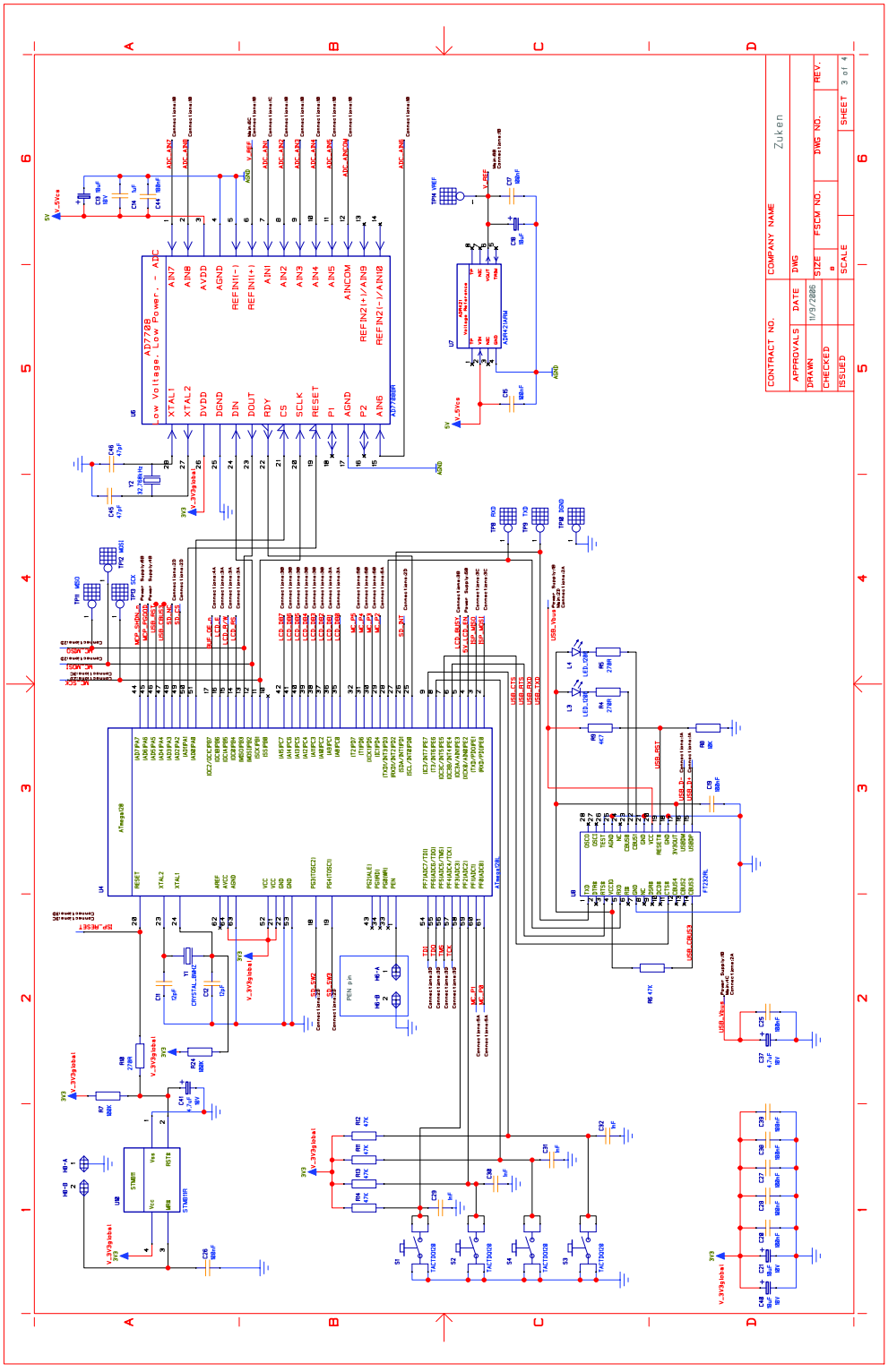
```
1  /*
```

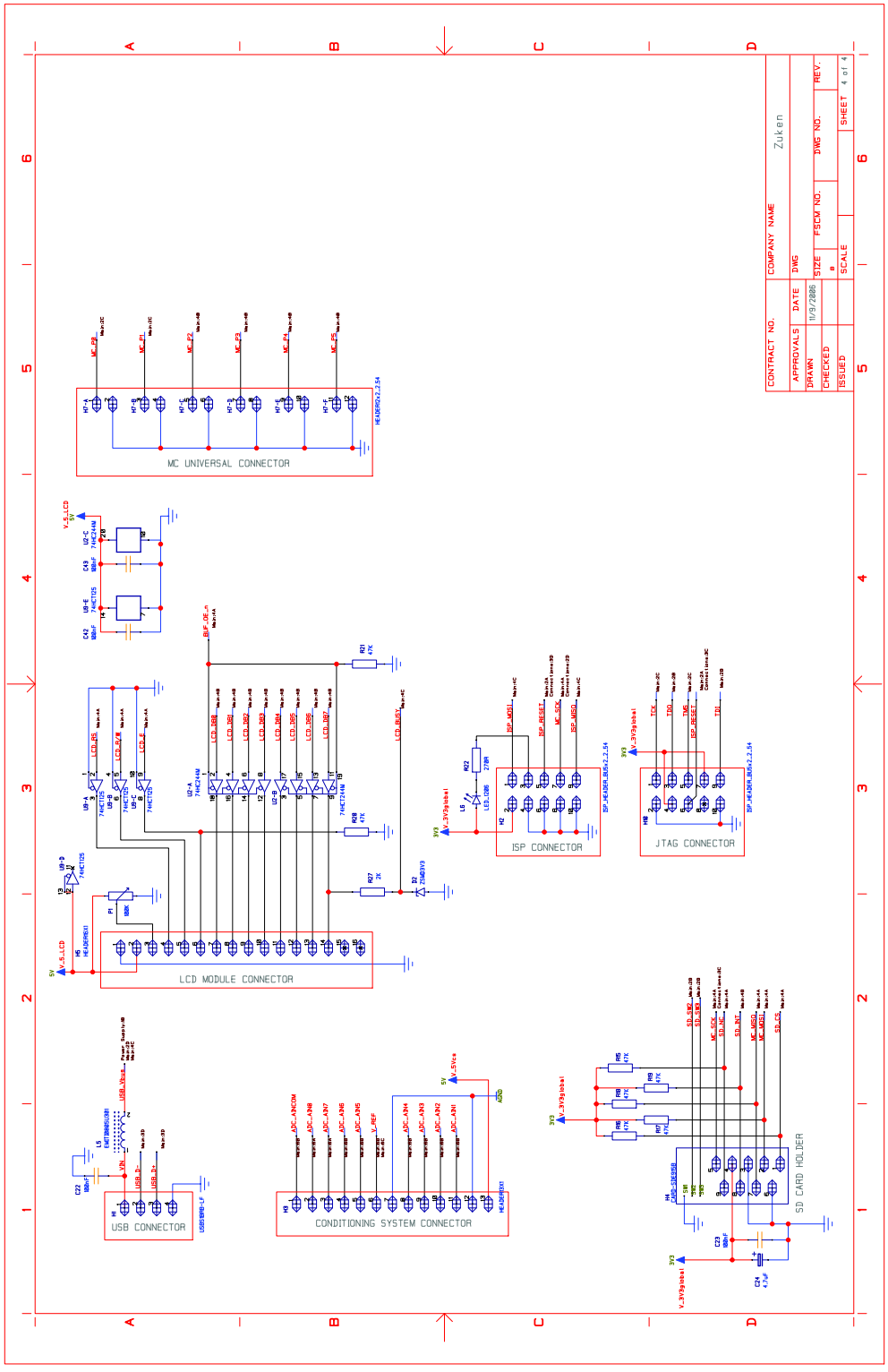
```
2  Plik usart.h
3  */
4
5  #ifndef USART
6  #define USART
7
8  void USART_init(unsigned int myubrr);
9  void USART_Transmit(unsigned char data);
10 void USART_puts(char *str);
11 unsigned char USART_Receive(void);
12
13 #endif
```

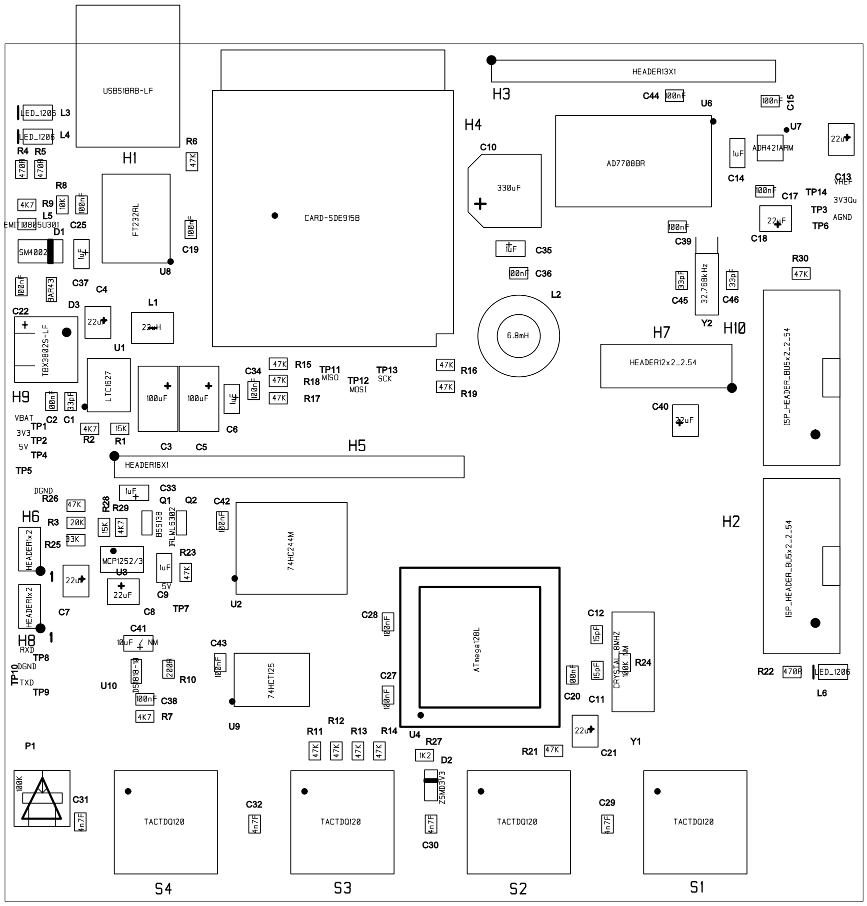
Dodatek B

Schematy


Zuken
B-Size Sheet
COMPANY NAME
CONTRACT NO.
APPROVALS
DRAWN
CHECKED
ISSUED
DATE
DWG
SIZE
FSCM NO.
DWG NO.
REV.
SCALE
SHEET 2 of 4
LP Filter (fc=8MHz)
LTC827
Step-Down Switching Regulator
MCP1252/3
LOW NOISE CHARGE PUMP
MCP_PGOOD
MCP_SHDN_n
V_Batt
V_5V_LCD
V_-5V
V_+3V3global
USB_VBUS


Low Voltage, Low Power, ADC
AD7798
ATmega128
FT232RL
Zuken
COMPANY NAME
CONTRACT NO.
APPROVALS
DRAWN
CHECKED
ISSUED
DATE
DWG
SIZE
FSCM NO.
DWG NO.
REV.
SCALE
SHEET 3 of 4


MC UNIVERSAL CONNECTOR
LCD MODULE CONNECTOR
ISP CONNECTOR
JTAG CONNECTOR
USB CONNECTOR
CONDITIONING SYSTEM CONNECTOR
SD CARD HOLDER
Zuken
COMPANY NAME
CONTRACT NO.
APPROVALS
DRAWN
CHECKED
ISSUED
DATE
DWG
SIZE
SCALE
FSCM NO.
DWG NO.
REV.
SHEET 4 of 4


HEADER13X1
H3
H4
H1
CARD-SDE915B
AD7708BR
FT232RL
ADR421ARM
330uF
6.8mH
32.768kHz
HEADER12x2_2.54
H7
H10
H9
LTC1627
100uF
HEADER16X1
H5
H6
H8
H2
74HC244M
74HCT125
ATmega128L
CRYSTAL_8MHZ
ISP_HEADER_BU5x2_2_54
MCP1252/3
TACTDQ120
S4
S3
S2
S1