Politechnika Wrocławska

WYDZIAŁ ELEKTRYCZNY
KIERUNEK: ELEKTROTECHNIKA

Mateusz Weber
Nr albumu: 174483

Opracowanie i wykonanie modułu uruchomieniowego dla 16-bitowego mikrokontrolera Siemens SAB 80c167

Development and implementation of runtime module for 16-bit microcontroller Siemens SAB 80c167

INŻYNIERSKI PROJEKT DYPLOMOWY

Studia: stacjonarne
Opiekun: **dr inż. Krzysztof Dyrcz**
Instytut : Maszyn, napędów i pomiarów elektrycznych

....
ocena *data, podpis opiekuna*

Wrocław, 2012

SPIS TREŚCI:

1. WSTĘP

System uruchomieniowy jest cyfrowym, elektronicznym układem mikroprocesorowym, zdolnym do wykonywania poleceń wydanych przez użytkownika systemu, zapisanych w postaci skompilowanego programu. System komunikuje się z użytkownikiem poprzez urządzenia wejścia wyjścia, takie jak klawiatury, wyświetlacze, porty wejściowe/wyjściowe, natomiast z obiektami, z którymi mikrokontroler współpracuje, komunikacja odbywa się poprzez układy wejścia/wyjścia, pośredniczące w wymianie informacji pomiędzy kontrolerem a obiektem.

Cały system działa pod kontrolą programu zapisanego w pamięci mikrokontrolera. Zasadniczo system uruchomieniowy składa się z dwóch części:
- mikrokontrolera, sterującego pracą całego systemu,
- elementów zewnętrznych, służących do komunikacji z użytkownikiem, jak i z układami sterowanymi przez system.

Mikroprocesor jest pojedynczym, cyfrowym układem scalonym, o dużym stopniu integracji [1]. Sterowanie mikroprocesorem odbywa się za pomocą rozkazów, zapisanych w pamięci. Ciąg rozkazów, które wykonuje układ jest programem. Realizacja programu odbywa się poprzez pobieranie kolejnych rozkazów z pamięci i ich wykonywaniu. Mikroprocesor jest w stanie przeprowadzać takie działania jak:
- kopiowanie danych (pomiędzy rejestrami, pamięciami);
- działania arytmetyczne (np. dodawanie, porównanie);
- operacje bitowe;
- sterowanie pracą programu poprzez skoki;
- zaawansowane mikroprocesory mogą posiadać dodatkowe moduły takie jak umożliwiające operacje zmiennoprzecinkowe (za pomocą np. wbudowanych koprocesorów).

Mikroprocesor poza wymienionymi działaniami nie jest w stanie samodzielnie wykonywać innych operacji. Z tego względu konieczne może być podłączenie do szyny systemowej mikroprocesora dodatkowych elementów. Szyna systemowa składa się z linii adresowych, danych i sterujących. W zależności od przeznaczenia systemu i potrzeb, do szyny systemowej mogą być podłączone takie elementy jak:
- pamięci;
- układy czasowo – licznikowe;
- przetworniki A/D;
- sterowniki transmisji danych;
- porty wejść/wyjść.

Połączenie w jednej strukturze mikroprocesora, oraz w zależności od potrzeb i przeznaczenia, odpowiednich elementów podłączonych do szyny systemowej, nazywa się mikrokontrolerem (mikrokomputerem jednoukładowym) [2]. Układy takie spełniają w systemie uruchomieniowym funkcje kontrolne i obliczeniowe. Po wybraniu mikrokontrolera, spełniającego oczekiwania względem żądanych wymagań, do zbudowania systemu uruchomieniowego należy doposażyć mikrokontroler w odpowiednie elementy zewnętrzne.

W skład systemu uruchomieniowego mogą wchodzić różne elementy, w zależności od wymagań i przeznaczenia systemu. Elementy te nie są łączone z szyną systemową, ale są podłączane do portów wejść/wyjść mikrokontrolera. Przykładowo, w skład systemu uruchomieniowego mogą wchodzić takie elementy jak:
- klawiatury, przyciski, zadajniki stanu, umożliwiające komunikację z otoczeniem;
- wyświetlacze, przedstawiające użytkownikowi informacje;
- sygnalizatory, przykładowo diody LED, buzzery, dostarczające informacji, np. o stanie portu;
- czujniki;
- inne sterowniki do sterowania obiektu, elementy wykonawcze, pośredniczące w przesyle informacji.

Występowanie i ilość elementów zewnętrznych są ściśle związane z przeznaczeniem układu. Przeznaczenie systemów uruchomieniowych jest bardzo szerokie. Przykładowymi zastosowaniami może być:
- nauka programowania procesorów;
- poznanie zasad budowy systemów mikroprocesorowych;
- budowanie i uruchamianie układów sterowanych za pomocą mikroprocesorów;
- nauka metod sterowania układów współpracujących z układami mikroprocesorami;
- testowanie układów mikroprocesorowych i układów współpracujących w mikroprocesorami;
- sterowanie obiektem, procesem.

Najpowszechniejszym zastosowaniem systemów uruchomieniowych, jest nauka programowania mikroprocesorów, a także układów współpracujących z mikrokontrolerem i metod sterowania tych układów. Wskazane jest, aby nauka ich programowania odbywała się za pomocą konkretnego, działającego systemu, umożliwiającego obserwację efektów pracy. Systemy takie powinny cechować się uniwersalnością i różnorodnością funkcji, i układów, tak aby można było poznać, i nauczyć się jak najwięcej możliwości wykorzystania mikrokontrolerów. Systemy te powinny oferować również optymalne wykorzystanie możliwości mikrokontrolera, wybranego do kontroli całego układu. Zasadniczo mikrokontrolery programuje się w następujących językach:
- asembler;
- C;
- innymi programowania obiektowego.

Język asemblera jest językiem niskiego poziomu. Pozwala na tworzenie kodu źródłowego bazującego na podstawowych operacjach procesora. Programista ma dużą kontrolę działania procesora. Pisząc program w asemblerze można bezpośrednio zapisywać i odczytywać komórki pamięci, porty mikrokontrolera. Ze względu na złożoność programów pisanych w asemblerze, czas wymagany do napisania programu, oraz mniejszą efektywność w stosunku do języka C, programy pisane w asemblerze tworzone są w przypadku, gdy wymagany jest niewielki kod wynikowy oraz duża szybkość wykonywania programu. Język C jest optymalny pod względem szybkości i łatwości pisania programów, rozmiaru kodu wynikowego oraz szybkości jego wykonywania. Języki programowania obiektowego są popularne wśród osób nie znających asemblera ani C, bądź są wykorzystywane w przypadku bardzo złożonych programów (gdy rozmiar kodu nie ma znaczenia), niektóre z nich posiadają udogodnienia (na przykład w postaci gotowych bibliotek) ułatwiające programowanie układów mikroprocesorowych.

System uruchomieniowy może również pełnić rolę testową. Przykładowo przed wprowadzeniem programu do docelowego systemu, można wypróbować jego działanie w systemie uruchomieniowym. Pozwala to na sprawdzenie poprawności programu/zastosowanego algorytmu. Budowa modułowa układu uruchomieniowego pozwala na diagnostykę układów mikroprocesorowych. Podmiana układu na płytce modułu uruchomieniowego, bądź procesora (jeśli istnieje taka możliwość), pozwala na szybkie sprawdzenie poprawności działania zainstalowanego układu.

Wszystkie systemy uruchomieniowe powinny spełniać dodatkowe wymagania, w zależności od stawianych wymagań i przeznaczenia, takie jak:
- odporność na bezmyślne działania użytkownika;
- uszkodzone elementy powinny mieć możliwość łatwego ich wymienienia na nowe;
- zakłócenie zewnętrzne nie powinno, bądź mieć możliwie ograniczony wpływ na pracę systemu uruchomieniowego. Analogicznie, zakłócenie wewnętrzne nie powinno przeno sić się do otoczenia.

2. CEL I ZAKRES PRACY

Celem pracy jest opracowanie i wykonanie modułu uruchomieniowego dla 16 – bitowego mikrokontrolera SAB 80c167.

Zakres pracy obejmuje:

1. Zapoznanie się z wybranym typem mikrokontrolera (SAB 80c167);

2. Zaprojektowanie i wykonanie zestawu uruchomieniowego współpracującego z mikrokontrolerem i wybranymi elementami wykonawczymi;

3. Napisanie przykładowego oprogramowania do testowania wykonanego układu.

3. PROJEKT MODUŁU URUCHOMIENIOWEGO

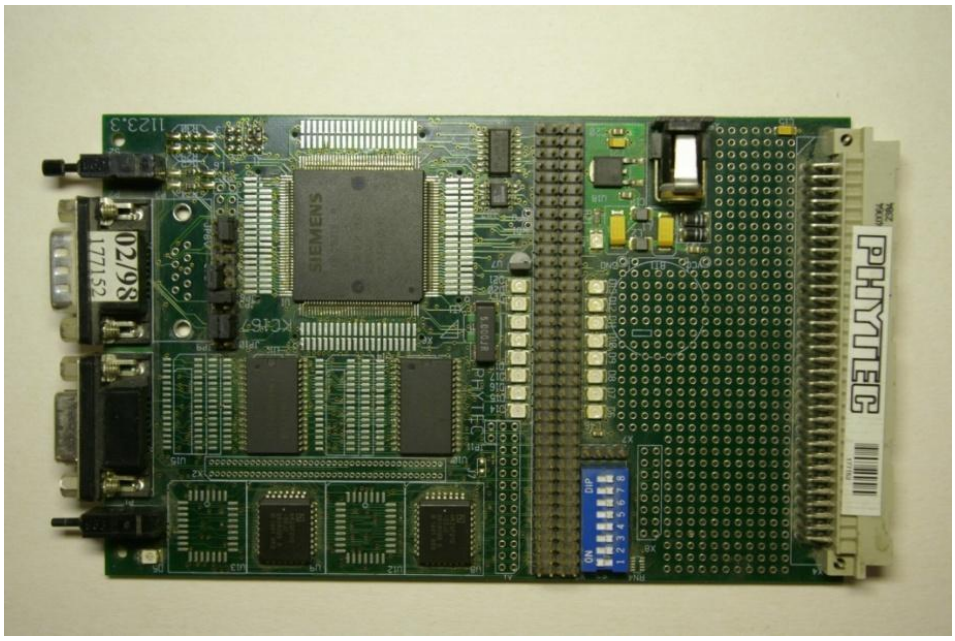
3.1. Założenia projektu

Projekt modułu uruchomieniowego zakładał przeznaczenie dydaktyczne układu. Miał on za zadanie zapoznać z sposobem budowania układów mikroprocesorowych i sterowaniem układów współpracujących z mikrokontrolerem. Z tego względu moduł został wyposażony w kilka różnych układów, które zostaną omówione w dalszych rozdziałach pracy:
- przetwornik analogowo cyfrowy;
- 8- bitowy cyfrowy port wyjściowy i wejście cyfrowe;
- klawiatura 12-sto przyciskowa;
- wyświetlacz LCD;
- wyjście z możliwością modulacji szerokości impulsów (PWM);
-buzzer piezoelektryczny.

3.1.1. Mikrokontroler Siemens SAB 80c167, płytka kitCON – 167

Układ SAB 80c167 jest 16 – bitowym mikrokontrolerem wykonanym w technologii CMOS. Posiada maksymalnie 111 bitów wejścia – wyjścia. Najważniejsze dane techniczne mikrokontrolera [3]:
- taktowanie zegara 33MHz;
- 2kB wewnętrznej pamięci RAM i 2kB zewnętrznej pamięci RAM;
- 16 kanałów 10-cio bitowego przetwornika A/D;
- 4 wyjścia z możliwością modulacji szerokości impulsów (PWM).

Mikrokontroler, wraz z dodatkowym wyposażeniem (zegar czasu rzeczywistego, port szeregowy, port CAN, 256 kB pamięci flash, dodatkowe 64kB pamięci SRAM, 16 diod LED przyłączonych po portu P2, zworki nastawiające) umieszczony jest na płytce firmy Phytec – kitCON – 167. Płytka zasilana jest napięciem stałym o wartości od +8 V do +12 V. Komunikację z komputerem umożliwia interfejs RS-232 za pomocą szeregowej transmisji danych. Porty mikrokontrolera zostały wyprowadzone na złącza, które są umieszczone w czterech rzędach po 76 pinów [4]. Rysunek 3.1 przedstawia płytkę kitCON – 167 z mikrokontrolerem SAB 80c167.



Rys. 3.1. Zdjęcie płytki Phytec kitCON – 167

3.1.2. Przetwornik analogowo – cyfrowy

Przetwornik analogowo – cyfrowy (A/D) jest układem umożliwiającym zamianę ciągłego sygnału analogowego na jego reprezentację w postaci cyfrowej. Proces przetwarzania sygnału analogowego na cyfrowy składa się z trzech etapów: próbkowania sygnału, jego kwantyzacji i kodowania. Przetwarzanie sygnału analogowego na cyfrowy jest wymagane w prawie każdym mikrokontrolerze, sterującym dowolnym procesem technologicznym, z tego względu, że parametrami procesów technologicznych są przeważnie wielkości nieelektryczne. Wartości te są przetwarzane za pomocą czujników na sygnał elektryczny, który z kolei musi zostać przetworzony na sygnał cyfrowy w przetworniku A/D.

Podstawowymi wielkościami charakteryzującymi przetworniki analogowo/cyfrowe są:

- długość słowa, czyli liczba bitów przetwornika,

- czas przetwarzania, czyli czas od rozpoczęcia przetwarzania do momentu, gdy wynik może zostać odczytany,

- zakres napięcia wejściowego.

Mikrokontroler SAB 80c167 posiada szesnaście 10 – bitowych kanałów przetwornika A/D. Do przetwarzania sygnałów użyto czterech kanałów, znajdujących się na pinach P5.0 – P5.3 mikrokontrolera. Zadajnikiem sygnału analogowego do przetwornika są cztery gniazda typu BNC znajdujące się na płytce uruchomieniowej. Dodatkowo na pinach P5.0 i P5.1 zostały umieszczone potencjometry umożliwiające regulację poziomu napięcia na tych portach.

3.1.3. 8 – bitowy cyfrowy port wyjściowy i cyfrowy port wejściowy

W projekcie został przewidziany jeden 8 – bitowy port wyjściowy, oraz jeden 8 –bitowy port wejściowy. Port wyjściowy zakładał możliwość zadawanie sygnałów cyfrowych do otoczenia, natomiast port wejściowy pobieranie sygnałów cyfrowych z zewnątrz, o wartości napięcia dla poszczególnych stanów +5V i 0V. Cyfrowy port wyjściowy powinien mieć możliwość podłączenia układu, zasilanego napięciem stałym o wartości przekraczającej 5V. Dodatkowo, w układzie powinna być sygnalizacja stanu (poprzez diody LED) każdego wejścia i wyjścia cyfrowego. Połączenie z mikrokontrolerem zostało zrealizowane przez port P8 – piny P8.0 – P8.7 (wyjście cyfrowe) i port P5 – piny P5.4 – P5.11 (wejście cyfrowe).

3.1.4. Klawiatura 12 – przyciskowa

Istotnym elementem układu jest klawiatura, umożliwiająca komunikację modułu z użytkownikiem. W projekcie zastosowano klawiaturę 12 – przyciskową, taka jak stosowana w aparatach telefonicznych. Ze względu na to, że klawiatura została wykonana jako matrycowa, do komunikacji z mikrokontrolerem wykorzystywanych jest 8 pinów. Ze względu na specyfikę obsługi klawiatury matrycowej, została ona podłączona do portu P2, do pinów P2.0. – P2.7. (na płytce kitCON-167 są podłączone diody LED wskazujące stan tego portu, co ułatwia programowanie i zrozumienie zasady działania takiej klawiatury).

3.1.5. Wyświetlacz LCD

Do komunikacji układu z użytkownikiem i otoczeniem został wykorzystany wyświetlacz LCD, umożliwiający wyświetlanie po szesnaście znaków w każdej z dwóch linii wyświetlacza. Zastosowany wyświetlacz jest wyświetlaczem alfanumerycznym, którego pracą kieruje sterownik HD44780. Komunikacja z mikrokontrolerem może przebiegać w trybie 4 – bitowym jak i 8 – bitowym, z możliwością odczytu flagi zajętości [8]. Opracowany moduł uruchomieniowy jest połączony z wszystkimi wyprowadzeniami wyświetlacza (8 bitów danych i 3 bity sterujące), więc sterowanie może odbywać się w trybie zarówno 4 – bitowym jak i 8 – bitowym, także z możliwością odczytu flagi zajętości. Przewidziano również możliwość załączenia podświetlania wyświetlacza w sposób programowy. Wyświetlacz połączony został z mikrokontrolerem przez port P3. Linie danych znajdują się na pinach P3.0 – P3.7, sygnały sterujące na pinach P3.8 i P3.9, oraz P3.13 i P3.15 (podświetlanie).

3.1.6. Wyjście PWM

Wyjście z możliwością modulacji szerokości impulsu umożliwia regulację sygnału napięciowego. Regulacja odbywa się poprzez odpowiednie załączanie i wyłączanie sygnału napięciowego o stałej amplitudzie. W rezultacie, w wyniku szybkich zmian napięcia uzyskuje się regulowany sygnał o mniejszej amplitudzie (tylko jeżeli wypełnienie okresu przebiegu będzie równe okresowi otrzymuje się sygnał o tej samej amplitudzie). Modulacja szerokości impulsów jest szeroko wykorzystywana do regulacji prędkości obrotowej silników elektrycznych. Mikrokontroler SAB 80c167 posiada cztery kanały z możliwością generowaniu sygnału PWM (port P7 – piny P7.0 – P7.3). W omawianym układzie wszystkie wyjścia zostały wyprowadzone. Dodatkowo, wyjścia PWM ze względów technicznych powinny mieć zapewnioną izolację galwaniczną obwodu modułu uruchomieniowego i obwodu dołączonego do modułu (aby nie obciążać wyjść mikrokontrolera, np. w przypadku dołączenia silniczka elektrycznego). Izolacja galwaniczna została zrealizowana poprzez optoizolację za pomocą transoptorów.

3.1.7. Brzęczyk piezoelektryczny

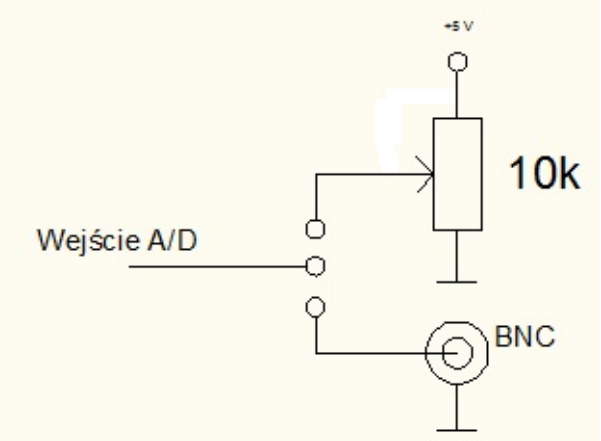
Projekt modułu uruchomieniowego uwzględniał instalację buzzera piezoelektrycznego. Układ powinien mieć możliwość sygnalizacji dźwiękowej stanu portu, do którego jest podłączony buzzer. Brzęczyk został połączony do pinu P7.6 mikrokontrolera.

3.2. Opis projektu

Elektryczne schematy ideowe poszczególnych układów zostały wykonane w programie „Auto Cad 2010”, w wersji studenckiej. Schemat montażowy oraz rozmieszczenie elementów na płytce zostało wykonane za pomocą programu „Altium Designer”. Program, przedstawiający działanie modułu uruchomieniowego został napisany w języku C, w środowisku „Keil µVision 4”.

3.2.1. Realizacja przetwornika analogowo – cyfrowego, pomiar napięcia

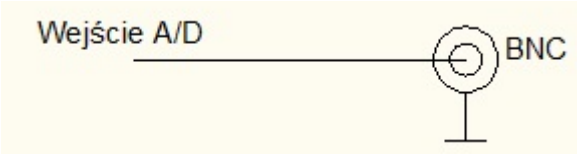
Zgodnie z założeniami projektu, układ został wyposażony w cztery wejścia przetwornika A/D (piny P5.0 – P5.3), do których sygnał jest zadawany przez gniazdo BNC. Dodatkowo, zainstalowano dwa potencjometry, z których odczepów sygnał napięcia jest podawany na pin 5.0. i 5.1. portu przetwornika. Wartość rezystancji potencjometru wynosi 10kΩ (zbyt mała wartość powodowałaby zbytnie obciążenie napięcia zasilającego). Schemat ideowy zadajnika z wyborem wejścia pokazano na rysunku 3.2.



Rys. 3.2. Schemat ideowy zadajnika sygnału napięciowego na pinach P5.0. i P5.1.

Wyboru pomiędzy pomiarem napięcia z suwaka potencjometru, a gniazdem BNC dokonuje użytkownik, odpowiednio umieszczając zworę w gnieździe.

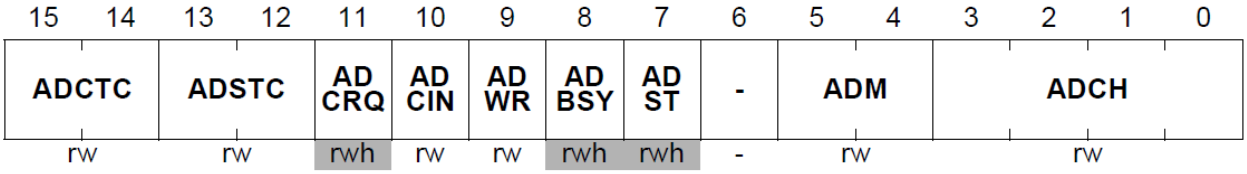
W przypadku kanałów przetwornika umieszczonych na pinie P5.2. i P5.3. istnieje możliwość pomiaru napięcia tylko z gniazda typu BNC. Schemat ideowy pokazano na rysunku 3.3.



Rys. 3.3. Schemat ideowy zadajnika sygnału napięciowego na pinach P5.2. i P5.3.

Obsługa przetwornika sprowadza się do wpisania odpowiednich wartości do rejestru ADCON mikrokontrolera, oraz odczytania wyniku zawierającego się w rejestrze ADDAT. Pomiaru napięcia można dokonywać w ograniczonym zakresie, zależnym od napięcia odniesienia V_{REF} i V_{GND}, w opracowanym module uruchomieniowym napięcia te są równe napięciu zasilania mikrokontrolera i wynoszą odpowiednio +5 V i 0V.

Według dokumentacji technicznej mikrokontrolera 16 – bitowy rejestr ADCON ma następującą składnię i odpowiada za następujące działania [3]:



- bity oznaczone jako ADCH odpowiadają za wybór kanału pomiaru przetwornika A/D, ustawienie bitu na kolejnych pozycja powoduje wybranie działania kolejnego kanału pomiarowego;

- tryb pracy przetwornika wybiera się ustawiając bity oznaczone ADM (wpisanie do ADM wartości 0 wybiera tryb standardowy);

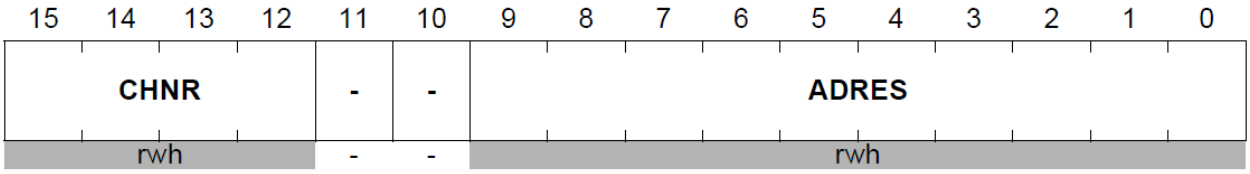
- przetwarzanie rozpoczyna się po ustawieniu bitu ADST;

- bit ADBSY oznacza flagę zajętości, przyjmuje ona wartość „1”, gdy przetwornik pracuje i „0”, gdy wynik pomiaru zostaje umieszczony w rejestrze ADDAT;

- bit ADWR jest bitem kontroli odczytu;
- ADCIN – oznacza włączenie kanału;
- ADCRQ – żądanie flagi kanału wejściowego;
- ADSTC – bity definiujące czas próbkowania przetwornika;

- ADCTC – definicja zegara konwersji przetwornika.

Rejestr ADDAT, zawierający wynik pomiaru przetwornika jest 16 – bitowy. Zawiera następujące informacje [3]:



- bity oznaczone jako ADRES zawierają 10 – bitowy wynik pomiaru przetwornika;

- bity oznaczone CHNR zawierają numer kanału, którego wynik pomiaru jest przechowywany.

Programowy sposób obsługi przetwornika został zaimplementowany w osobnej funkcji [8]:

```
int przetwornik(int kanal){
    ADCON = 0xB000;              //”zerowanie” przetwornika
    ADCON |= kanal & 0x000F;          //ustawnie portu pomiaru
    ADST = 1;                  //start
    while (ADBSY == 1);          //czekanie aż zakończy się pomiar
    return(ADDAT & 0x03ff);          //zwrócenie wyniku
          }
```

Argumentem funkcji jest numer kanału. Po odpowiednim ustawieniu rejestru ADCON, ustawieniu bitu startu ADST, oraz odczekaniu na zakończenie pomiaru funkcja zwraca wynik przechowywany w ADDAT, bez numeru kanału. Wynik, jest zapisany w postaci systemu dwójkowego, przy czym maksymalnej wartości odpowiada ustawienie wszystkich bitów ADRES w rejestrze ADDAT w stan wysoki „1" (wartość dziesiętna to 1023). Aby uzyskać napięcie wyrażone w woltach, należy pomnożyć wartość zawierającą się w ADRES przez stałą zależną od napięcia odniesienia, zgodnie z równaniem (3.1).

$$\frac{V_{REF}}{1023} = \frac{5}{1023} = 0{,}049 \qquad (3.1)$$

Odpowiada to podzieleniu wyniku przez 204.

W celu sprawdzenia poprawności działania przetwornika dokonano pomiaru napięcia zadawanego za pomocą multimetru VOREL 81780. Wyniki pomiarów przedstawiono w tabeli 3.1 (rozdzielczość odczytu przetwornika dobrano do rozdzielczości pomiaru multimetru).

Tabela 3.1. Porównanie napięcia zmierzonego przetwornikiem A/D z napięciem zmierzonym multimetrem VOREL81780:

Wartość zmierzonego napięcia		
Multimetr VOREL 81780 (zakres 20V)	Przetwornik A/D kanał 0	Przetwornik A/D kanał 1
V	V	V
0,00	0,00	0,00
0,51	0,51	0,51
1,00	1,00	1,00
1,49	1,50	1,50
2,01	2,02	2,02
2,50	2,51	2,51
3,02	3,03	3,03
3,51	3,53	3,53
4,01	4,04	4,04
4,54	4,57	4,57
4,98	5,01	5,01

Zgodnie z oczekiwaniami, dokładność pomiaru za pomocą mikrokontrolera jest zadawalająca. Pomiędzy kanałami przetwornika nie ma rozbieżności w pomiarach. Znikome różnice pomiędzy pomiarami multimetrem a przetwornikiem mikrokontrolera można wyjaśnić skończoną dokładnością multimetru oraz przetwornika A/D mikrokontrolera (dokładność multimetru VOREL 81780 została podana przez producenta jako: $\pm$0,5% + 1 cyfra [5]). Przetwornik A/D gwarantuje znacznie większą rozdzielczość odczytu wyniku pomiaru (w tym przypadku dostosowaną ją do rozdzielczości miernika).

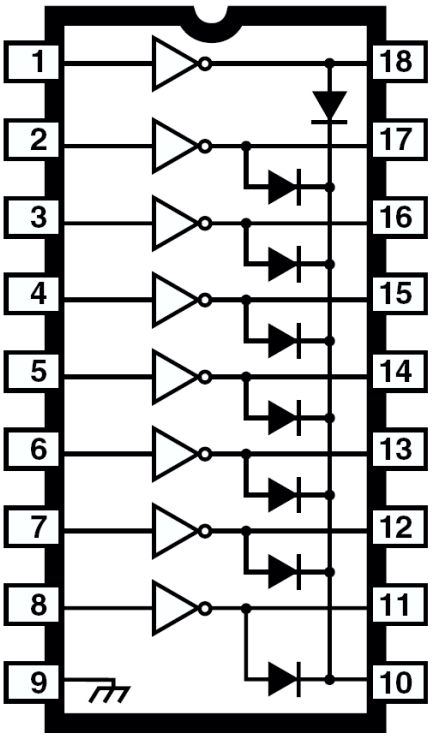
3.2.2. 8 – bitowy cyfrowy port wyjściowy i cyfrowy port wejściowy, operatory bitowe w języku C

Według założeń moduł uruchomieniowy powinien być wyposażony w jedno 8 – bitowe wejście cyfrowe, oraz jedno 8 – bitowe wyjście cyfrowe. Wyjście cyfrowe powinno umożliwiać podłączenie elementów wymagających napięcia większego niż 5 V. Stan logiczny każdego bitu powinien być sygnalizowany diodą LED. Wejścia i wyjścia cyfrowe zostały zrealizowane za pomocą tych samych układów scalonych ULN 2803. Sygnalizacja stanu logicznego odbywa się diodami LED dołączonymi do układu UCY 7404.Wejście cyfrowe zostało dołączone do portu P5, pinów P5.4 – P5.11 mikrokontrolera. Wyjście cyfrowe zostało wyprowadzone przez piny P8.0 – P8.7, portu P8 mikrokontrolera.

Układem pośredniczącym między mikrokontrolerem a dołączanym układem wejścia/wyjścia jest układ ULN 2803. Jest to układ scalony, w którym znajduje się 8 wzmacniaczy w układzie Darlingtona, cechujących się dużym wzmocnieniem sygnału. ULN 2803 umożliwia współpracę układów wykonanych w technologii CMOS i TTL. Zaletą układu jest możliwość dołączenia niezależnego zasilania, co powoduje możliwość zasilania układów napięciem wyższym niż 5 V. W obwodzie wejściowym układ zasilany jest napięciem 5 V, natomiast w obwodzie wyjściowym wyprowadzono dwa dodatkowe piny służące do zasilenia układu. Schemat wewnętrzny układu ULN 2903 przedstawiono na rysunku 3.4.

Najważniejsze parametry techniczne układu ULN 2803 [5]:

- maksymalne napięcie wyjściowe: 50 V;
- maksymalne napięcie wejściowe: 30 V;
- maksymalny prąd wejściowy długotrwały: 25 mA;
- maksymalny prąd wyjściowy długotrwały: 0,5 A.



Rys. 3.4. Schemat wewnętrzny układu ULN 2803 [5]

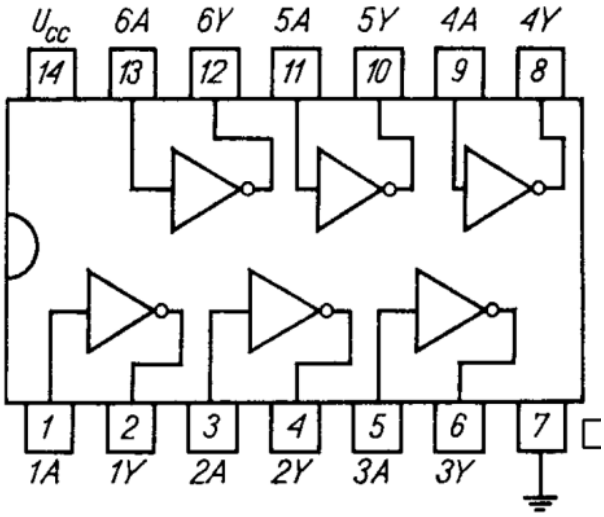
Pomiędzy portem mikrokontrolera a układem ULN 2803 zastosowano rezystory ograniczające prąd, pozwalającej uzyskać wartość prądu mniejszą od wartości maksymalnego prądu wyjściowego dopuszczonego długotrwale. Wartość wymaganej rezystancji została obliczona z prawa Oma, zgodnie z równaniem (3.2).

$$R = \frac{U}{I}\ [\Omega] \qquad (3.2)$$

Sygnalizacja stanu logicznego poszczególnych bitów portu odbywa się poprzez kolorowe diody LED – zielone w przypadku wyjścia cyfrowego i pomarańczowe w przypadku wejścia cyfrowego. Diody zostały dołączone do układu UCY 7404, który pełni rolę negatora. Schemat wewnętrzny układu UCY 7404 przedstawiono na rysunku 3.5.

Podstawowe parametry techniczne układu UCY 7404 [5]:

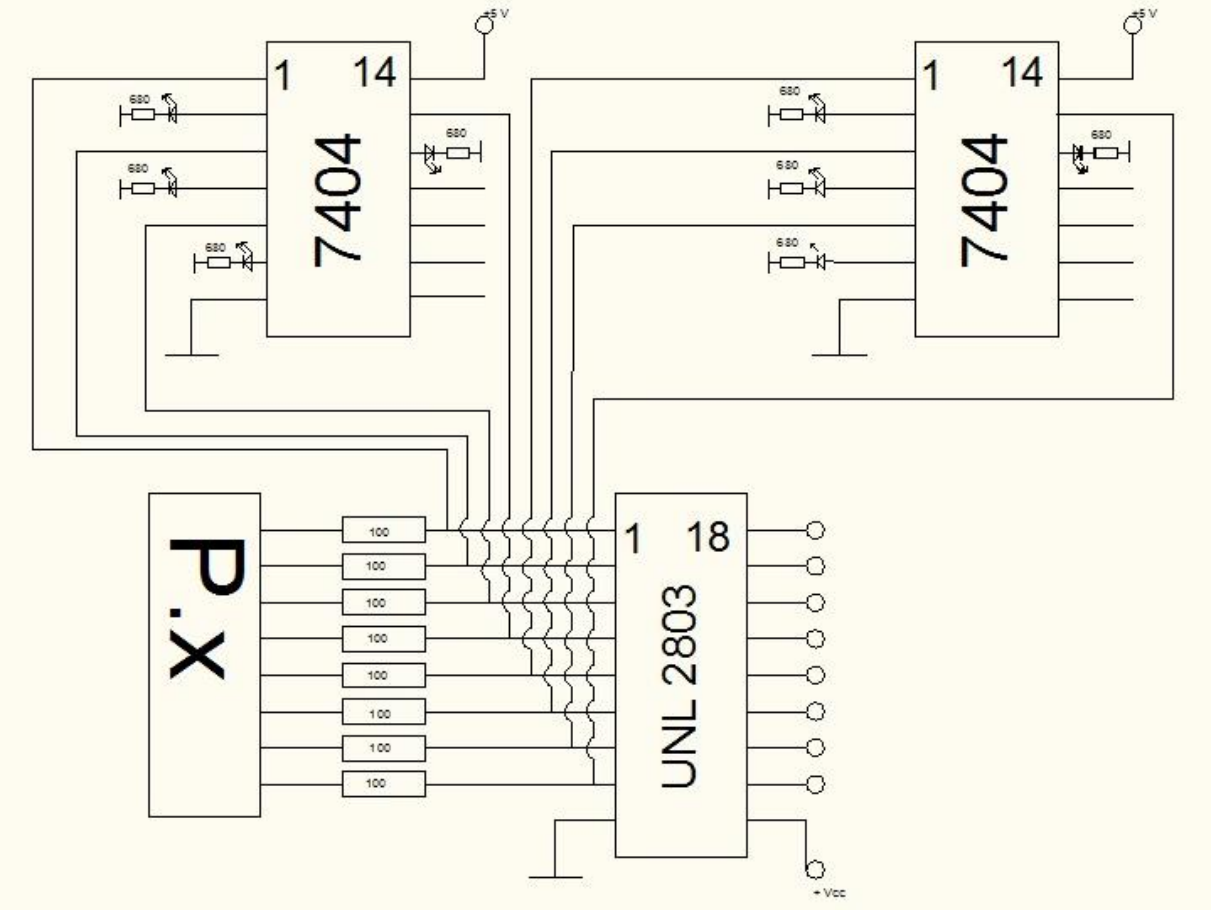
- napięcie zasilające: 4,75 V – 5,25 V (maksymalnie 7 V);
- minimalne napięcie wejściowe w stanie wysokim: 2 V;
- maksymalne napięcie wejściowe w stanie niskim: 0,8 V;
- minimalne napięcie wyjściowe w stanie wysokim: 2,4 V;
- maksymalne napięcie wyjściowe w stanie niskim: 0,4 V.



Rys. 3.5. Schemat wewnętrzny układu UCY 7404 [5]

Pomiędzy inwerterem a diodami sygnalizującymi, znajdują się rezystory ograniczające prąd płynący przez diody. Prąd należy ograniczyć do wartości około 20 mA. Wartość rezystancji została obliczona z prawa Oma, zgodnie z równaniem (3.2).

Rysunek 3.6 przedstawia sposób połączenia układu wyjścia cyfrowego.



Rys. 3.6. Schemat połączenia układu wejścia cyfrowego

W przypadku wejścia cyfrowego dodano dodatkowe rezystory ograniczające na wyjściu układu ULN 2803, o wartości 100Ω.

Port cyfrowy został wykorzystany w programie w trzech funkcjach:

- wskazanie w systemie dwójkowym wyniku pomiaru przetwornika A/D;

- powielenie stanu wejścia cyfrowego na wyjściu cyfrowym;

- ustawienie za pomocą klawiatury stanów logicznych poszczególnych bitów wyjścia cyfrowego.

Programowanie obwodu wyjścia i wejścia cyfrowego, oprócz podstawowych działań arytmetycznych, wymaga podstawowej znajomości operacji bitowych. Działania bitowe to [6]:

- negacja (NOT);
- koniunkcja (AND);
- alternatywa (OR);
- alternatywa rozłączna (XOR);
- przesunięcie bitowe.

Negacja (NOT) bitowa jest najprostszym działaniem na bitach. Wynikiem negacji jest liczba, która ma bity równe „1” na pozycjach, na których początkowo znajdowały się bity równe „0”, analogicznie bity przyjmują wartość „0”, gdy przedtem miały wartość „1”. W języku C za negacje bitową odpowiada operator „~” – tylda. Tabela 3.2 przedstawia działanie operacji NOT.

Tab. 3.2. Tablica prawdy funkcji NOT

NOT („~”)		
	0	1
	1	0

Koniunkcja (AND) daje w wyniku liczbę, która ma bity równe „1” tylko na tych pozycjach, na których oba argumenty funkcji miały bity równe „1”. W języku C operator „&” odpowiada za funkcję koniunkcji bitowej. Tabela 3.3 przedstawia działanie funkcji AND.

Tab. 3.3. Tablica prawdy funkcji AND

AND („&”)	0	1
0	0	0
1	0	1

Wynikiem alternatywy (OR) jest liczba, która przyjmuje na określonych bitach wartość „1”, gdy przynajmniej jeden z argumentów miał odpowiedni bit równy „1”. Operatorem alternatywy w języku C jest znak „|”. Tabela 3.4 pokazuje działanie funkcji „OR”.

Tab. 3.4. Tablica prawdy funkcji OR

OR („|”)	0	1
0	0	1
1	1	1

Funkcja alternatywy rozłącznej (XOR), zwanej też alternatywą wykluczającą, daje w wyniku liczbę, która mi bity równe „1” na tych pozycjach, na których tylko jeden z argumentów miał bit równy „1”. W pozostałych przypadkach wynikiem działania jest bit równy „0”. Znak „^” odpowiada funkcji alternatywy wykluczającej w języku C. Tabela 3.5 przedstawia działanie funkcji XOR.

Tab. 3.5. Tablica prawdy funkcji XOR

XOR („^”)	0	1
0	0	1
1	1	0

W języku C zostały zaimplementowane operatory przesunięcia bitowego w prawo („>>”) i w lewo („<<”). Przesunięcie przesuwa bity argumentu w odpowiednim kierunku. Dodatkowo jest możliwość zadawania liczy pozycji o jaką nastąpi przesunięcie. Bity będące na skraju słowa zostają tracone, natomiast w nowe miejsca wpisywane są wartości logiczne „0”. W zależności od kierunku przesunięcia bitowego bity tracone znajdują się po lewej stronie (przesunięcie w lewo), bądź w po prawej stronie (przesunięcie w prawo). Tabela 3.6 przedstawia działanie operacji przesunięcia bitowego (w przypadku długości słowa 4 – bity).

Tab. 3.6. Tablica prawdy przesunięcia bitowego

a	a<<1	a<<2	a>>1	a>>2
0001	0010	0100	0000	0000
0011	0110	1100	0001	0000
0101	1010	0100	0010	0001
1000	0000	0000	0100	0010
1111	1110	1100	0111	0011
1001	0010	0100	0100	0010

Warto zauważyć, że operacja przesunięcia bitowego w lewo o jeden bit odpowiada pomnożeniu argumentu przez dwa, analogicznie, przesunięcie bitowe w prawo odpowiada podzieleniu argumentu przez dwa.

Programowa realizacja wskazania wyniku pomiaru przetwornika A/D przez port cyfrowy polega na wpisaniu wartości ADRES rejestru ADDAT, na port 8 mikrokontrolera. Ze względu na to, że wartość wyniku jest liczą 10 – bitową, a port 8 jest portem 8 – bitowym, konieczne jest wykonanie operacji przesunięcia bitowego o dwie pozycje w prawo, czyli podzieleniu wyniku prze cztery. Programowo wskazanie napięcia zmierzonego zostało zrealizowane w następujący sposób:

```
x=przetwornik(kanal)/4;
P8=x;
```

Powielenie stanu bitowego portu wejściowego na porcie wyjściowym jest prostą operacją przypisania. Ponieważ wejście cyfrowe jest dołączone do pinów P5.4. – P5.11. portu 16 – bitowego należy wartość portu piątego przesunąć o 4 bity w lewo. Rozwiązanie w programie mieści się w jednej linijce kodu:

```
P8=P5>>4;
```

Zmiana stanu logicznego na cyfrowym porcie wyjściowym za pomocą klawiatury jest możliwa po naciśnięciu przycisku klawiatury. Kolejne naciśnięcie tego samego przycisku powoduje kolejną negacje stanu odpowiadającego bitu. W tym przypadku została wykorzystana zależność logiczna: a ^ b ^ b = a. W programie funkcja ta została zapisana następująco:

```
P8^=1<<a;          //a jest wartością odczytaną z klawiatury
```

3.2.3. Klawiatura 12 – przyciskowa, obsługa, drgania styków

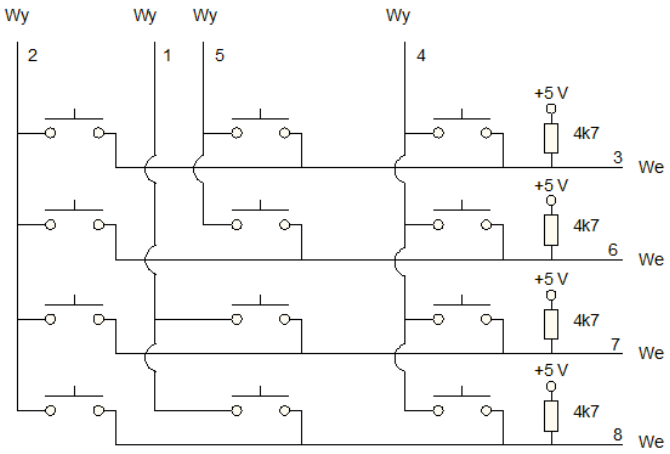
Układem pośredniczącym w komunikacji pomiędzy użytkownikiem a mikrokontrolerem, jest klawiatura matrycowa, 12 – przyciskowa. Wykonanie matrycowe klawiatury pozwala na efektywniejsze wykorzystanie dostępnych pinów mikrokontrolera. Przykładowo, klawiatura matrycowa, połączona 6 przewodami z mikrokontrolerem umożliwia odczyt dziewięciu różnych przycisków.

W projekcie modułu uruchomieniowego zastosowano klawiaturę matrycową, połączoną przez 8 pinów z mikrokontrolerem. Klawiatura została dołączona do portu P2, pinów P2.0. – P2.7. mikrokontrolera. Klawiatura posiada dwanaście przycisków, których układ i oznaczenie jest takie jak na klawiaturze aparatu telefonicznego. Rysunek 3.7 przedstawia zdjęcie wykorzystanej klawiatury.

Rys. 3.7. Zdjęcie wykorzystanej klawiatury

Obsługa klawiatury matrycowej wymaga innego, bardziej złożonego algorytmu odczytu stanów klawiszy, niż standardowej klawiatury. Najprostszy algorytm odczytu przycisków zakłada ustawienie połowy dostępnych pinów klawiatury jako wejście mikrokontrolera i połączenie z zasilaniem, a drugą połowę jako wyjście z mikrokontrolera i ustawienie w stan wysoki „1”. Następnie należy ustawiać stan niski na kolejnych wyjściach kontrolera i odczytywać stan wejść. Zmiana wartości logicznej na „0”, na którymkolwiek z pinów pracujących jako wejście, umożliwia odczytanie, który klawisz został naciśnięty. Mikrokontroler SAB 80c167 wymaga dołączenia tzw. rezystorów podciągających, „pull-up”, o wartości około 4,7 kΩ do pinów działających jako wejścia klawiatury, pomiędzy zasilaniem.

Wewnętrzny schemat elektryczny, wraz z numerem pinu i rezystorami podciągającymi klawiatury przedstawia schemat na rysunku 3.8.



Rys. 3.8. Schemat wewnętrzny z zaznaczonymi numerami i funkcją pinu klawiatury, oraz rezystorami podciągającymi

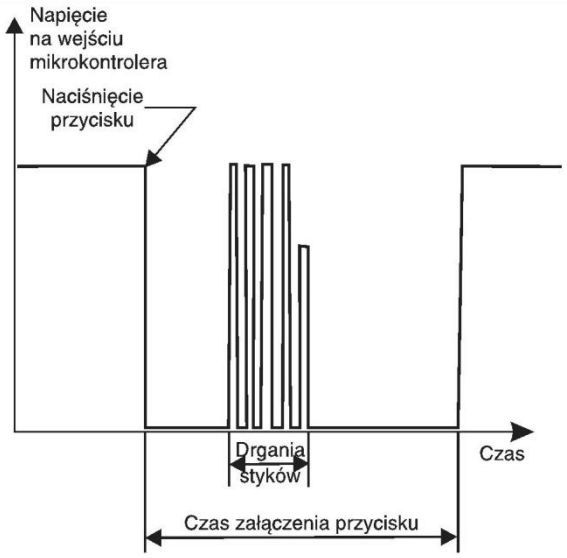
Piny klawiatury oznaczone jako wejścia (We), wraz z rezystorami podciągającymi zostały podłączone do pinów P2.0. – P2.3. mikrokontrolera, a piny oznaczone jako wyjścia (Wy), podłączono do pinów P2.4 – P2.7.

Algorytm odczytu stanu klawiatury został napisany w programie jako osobna funkcja:

```
int klaw(void)
{

int klawisz;
DP2 = 0xfff0;      //1 - pin jest wyjściem, 0 - wejściem
P2 = 0xfff0;

while(1){
        P2 =  0xffe0;                       //pin 1 klawiatury w stan niski
        if(P2 ==0xffeb){klawisz=8;break;}
        if(P2 == 0xffe7){klawisz=0;break;}

        P2 = 0xffd0;                        //pin 2 klawiatury w stan niski
        if(P2 == 0xffde){klawisz=1;break;}
             if (P2 == 0xffdd){klawisz=4;break;}
        if(P2 ==0xffdb){klawisz=7;break;}
        if(P2 == 0xffd7){klawisz=10;break;}          //gwazdka

        P2 = 0xffb0;                        //pin 4 klawiatury w stan niski
        if(P2 == 0xffbe){klawisz=3;break;}
         if (P2 == 0xffbd){klawisz=6;break;}
        if(P2 ==0xffbb){klawisz=9;break;}
        if(P2 == 0xffb7){klawisz=11;break;}          //kratka

        P2 = 0xff70;                        //pin 5 klawiatury w stan niski
        if(P2 == 0xff7e){klawisz=2;break;}
             if (P2 == 0xff7d){klawisz=5;break;}

        }

return(klawisz);
}
```

Analizując poszczególne wartości liczbowe (zapisane systemem szesnastkowym) w przedstawionej funkcji, łatwo zauważyć, że (zgodnie z wcześniejszym schematem i algorytmem odczytu stanu klawiatury matrycowej) zmieniana jest wartość logiczna bitów działająca jako wyjście. Po każdej zmianie stanu wyjścia sprawdzana jest wartość logiczna pinów, które są ustawione jako wejście. Działania te powtarzają się, dopóki nie zostanie wciśnięty jakiś przycisk. Po wykryciu zmiany wartości logicznej wejścia, odpowiadającej naciśnięciu przycisku klawiatury, funkcja dekodując przycisk, który został wciśnięty zwraca jego numer. Mikrokontroler powtarza całą funkcję na tyle szybko, że naciśnięcie dowolnego przycisku jest niezauważalnie szybko wychwytywane. Przedstawioną funkcję można zapisać w sposób bardziej zwięzły, za pomocą dwóch pętli *for*, jednakże program jest wtedy zdecydowanie mniej czytelny.

Naciśnięciu przycisku klawiatury towarzyszy zjawisko drgania styków („Bouncing”). Zjawisko to ma podłoże mechaniczne [9]. Polega ono na drganiu styków klawiatury przy naciskaniu i puszczaniu klawiszy. Towarzyszy temu generowanie losowej ilości impulsów, które są bardzo niekorzystne, ze względu na brak możliwości odczytania ile razy klawisz został naciśnięty. Drganie styków zostaje odczytane przez mikrokontroler jako kilkukrotne naciśnięcie klawisza. Na rysunku 3.9 pokazano przebieg oscylograficzny, prezentujący zjawiska towarzyszące naciśnięciu klawisza [9].



Rys. 3.9. Przykładowy przebieg obrazujący drgania styków [9]

Ilość impulsów wygenerowanych przez drgające styki jest zależna od budowy mechanicznej klawiatury, ale zawsze jest to liczba losowa. Efektem drania styków może być brak kontroli nad programem (mikrokontroler interpretuje każdą zmianę stanu jako naciśnięcie klawisza).

W opracowanym module uruchomieniowym drgania styków powodowały przeskakiwanie menu i brak kontroli nad ilością odczytanych przez mikrokontroler naciśnięć klawisza.

Rozwiązaniem problemu może być instalacja dodatkowych elementów wspomagających pracę klawiatury (takie jak wyspecjalizowane układy typu MAX681x, diody, filtry RC). Wadą takiego rozwiązania jest konieczność instalacji dodatkowych układów. Wpływ drgań styków może zostać również ograniczony w sposób programowy. Istnieje wiele algorytmów niwelujących wpływ dodatkowych impulsów. W programie wykorzystano najprostszy z możliwych algorytmów. Po wykryciu naciśnięcia klawisza w programie generowane jest opóźnienie, trwające na dyle długo, aby drgania ustały i odczytane zostało pojedyncze naciśnięcie klawisza:

```
b=klaw();
for(st=400;st>0;st--){czekaj;}          //drgania styków
```

polecenie *czekaj* zostało wcześniej zdefiniowane jako:

```
#define czekaj for(c=3000;c>0;c--)
```

Czas generowanego opóźnienia został dobrany metodą prób. Zalecany czas opóźnienia wynosi około 50ms, jednakże jest on uzależniony od właściwości mechanicznych klawiatury.

3.2.4. Wyświetlacz LCD

Stan pracy modułu uruchomieniowego i informacje o przebiegu wykonywania programu zostały przedstawione na wyświetlaczu LCD. Użyty wyświetlacz umożliwia wyświetlanie po szesnaście znaków w dwóch liniach. Każdy znak zdefiniowany jest na polu 5 na 7 pikseli, pozwala to na wyświetlanie dowolnych znaków. Wyświetlacz oferuje możliwość zdefiniowania przez użytkownika do ośmiu znaków, można w ten sposób wyświetlać polskie znaki. Możliwe jest również włączenie podświetlania w kolorze zielonym. Napięcie zasilające wynosi +5 V. Rysunek 3.10 przedstawia zdjęcie wykorzystanego w module uruchomieniowym wyświetlacza.

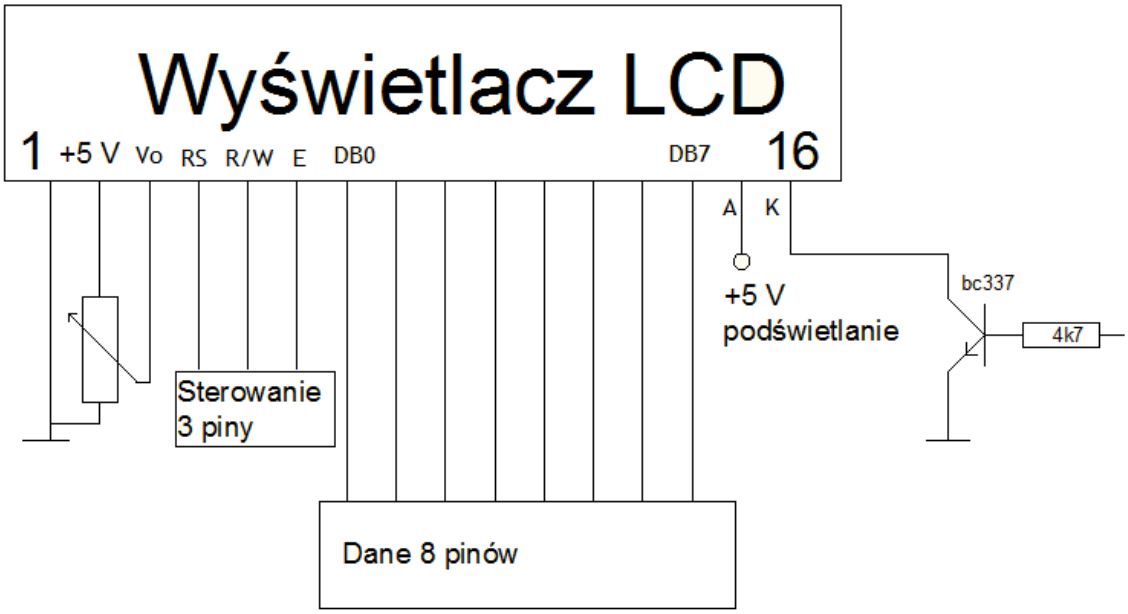
Rys. 3.10. Zdjęcie wykorzystanego wyświetlacza

Ze względu na złożony proces sterowania omawianym wyświetlaczem oraz znaczną ilość danych do przetworzenia, wyświetlacze LCD są standardowo wyposażone w specjalizowane procesory, które zarządzają wyświetlaniem – czyli sterowniki wyświetlacza. Wybrany wyświetlacz został wyposażony w sterownik firmy Hitachi, HD44780. Mikrokontroler wysyła dane do sterownika (znaki do wyświetlenia, instrukcje), który przetwarzając je, zapala odpowiednie punkty, bądź wykonuje polecenia sterujące pracą wyświetlacza. Po wysłaniu instrukcji do wyświetlacza, musi on przetworzyć te dane i wykonać odpowiednie działanie. Na wykonanie tych operacji sterownik potrzebuje określonego czasu. W tym czasie sterownik jest zajęty i nie może przyjmować kolejnych poleceń (możliwe jest tylko odczytanie stanu).

Kiedy sterownik jest zajęty, ustawiana jest przez niego tzw. flaga zajętości. Odczytywanie stanu flagi pozwala na precyzyjne, szybkie wydawanie kolejnych poleceń sterownikowi wyświetlacza. Współpraca mikrokontrolera z wyświetlaczem nie wymaga odczytu flagi zajętości. Należy wtedy, po każdorazowym wysłaniu danych do wyświetlacza odczekać pewien okres czasu, uzależniony od wykonywanej operacji. W tym przypadku, należy wygenerować programowe opóźnienie. Zrezygnowanie z odczytu flagi zajętości umożliwia nie podłączanie jednego wyprowadzenia wyświetlacza do LCD.

Zastosowany wyświetlacz, może współpracować z mikrokontrolerem w trybie 4 – bitowym i 8 – bitowym. Tryby te różnią się między sobą ilością bitów danych połączonych z mikrokontrolerem. Połączenie 8 – bitowe wymaga podpięcia wszystkich bitów danych do mikrokontrolera, a tryb 4 – bitowy, podłączenia 4 starszych bitów.

Wykonany moduł uruchomieniowy jest połączony z wszystkimi wyprowadzeniami wyświetlacza. Umożliwia więc pracę w trybie 8 – bitowym z odczytem stanu flagi zajętości. Wyświetlacz został połączony z mikrokontrolerem poprzez port P3, przy czym bitom danych odpowiadają piny P3.0 – P3.7, natomiast piny P3.8., P3.9., P3.13. i P3.14. odpowiadają za sygnały sterujące i możliwość programowego załączenia podświetlania. Ze względu na stosunkowo duży prąd wymagany do podświetlenia wyświetlacza (około 100mA), użyto tranzystora bipolarnego, typu NPN, do załączania podświetlania. Rysunek 3.11 przedstawia schemat podłączenia wyświetlacza, wraz z oznaczeniem wyprowadzeń.



Rys. 3.11. Schemat podłączenia wyświetlacza LCD, wraz z oznaczeniem wyprowadzeń

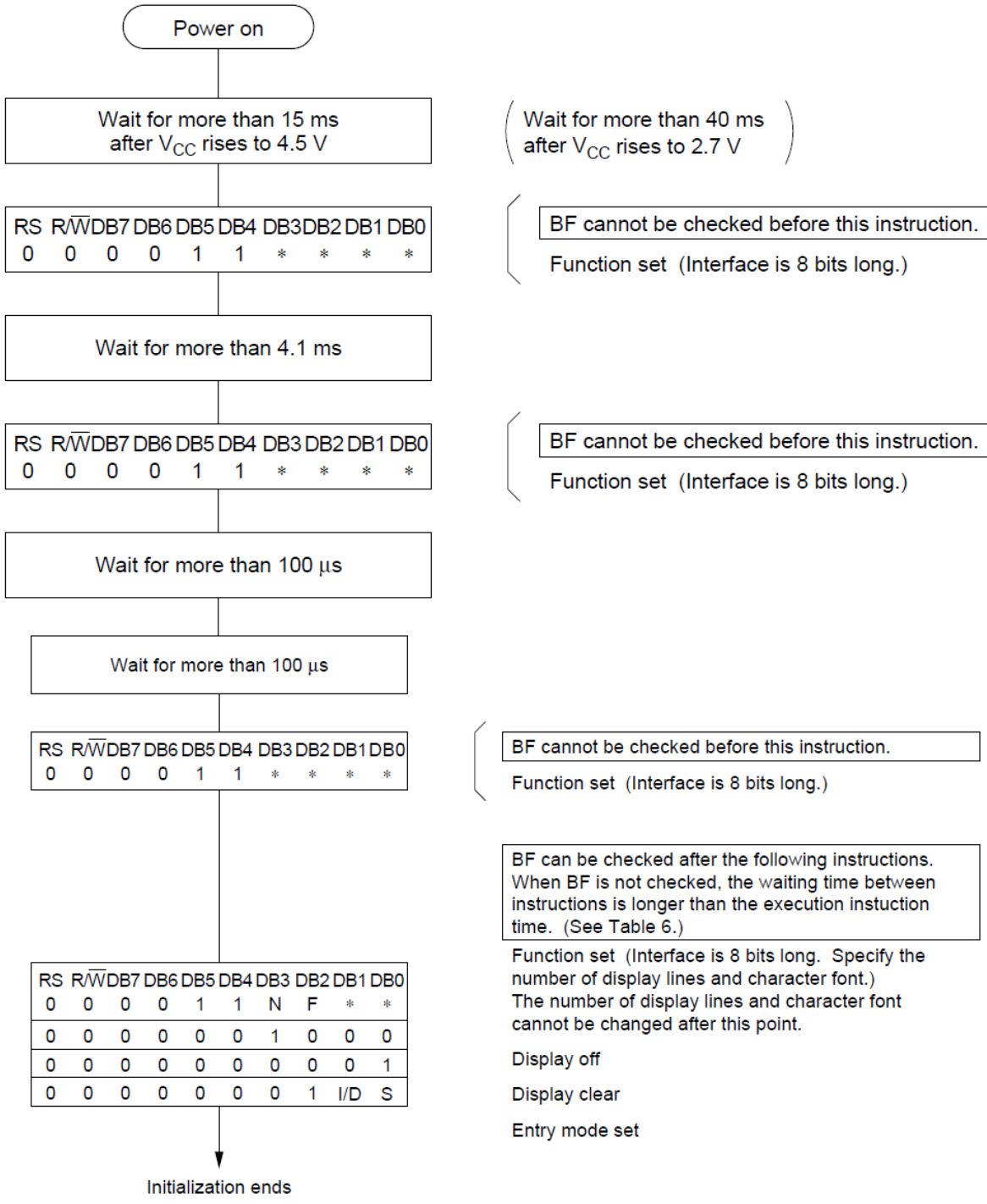
Funkcję poszczególnych wyprowadzeń wyświetlacza objaśnia tabela 3.7 [5].

Tab. 3.7. Oznaczenia poszczególnych wyprowadzeń wyświetlacza LCD [5]

Nr wyprowadzenia	Oznaczenie	Funkcja
1	GND	Masa
2	VCC	Napięcie zasilające +5 V
3	V0	Napięcie kontrastu
4	RS	Wybór rodzaju przesyłanych informacji (0 - komendy, 1- dane)
5	R/W	Wybór kierunku transmisji
6	E	Sygnał zezwalający (enable)
7	DB0	Bit danych
8	DB1	Bit danych
9	DB2	Bit danych
10	DB3	Bit danych
11	DB4	Bit danych
12	DB5	Bit danych
13	DB6	Bit danych
14	DB7	Bit danych
15	A(+)	Anoda (podświetlanie)
16	K(-)	Katoda (podświetlanie)

Wysłanie jakichkolwiek informacji do sterownika wyświetlacza może nastąpić tylko przy zboczu opadającym sygnału zezwalającego – enable. Sygnał R/W decyduje o kierunku transmisji, czy odbywa się ona z mikrokontrolera do wyświetlacza, czy odczytywana jest zawartość pamięci sterownika (stan niski „0” oznacza przesyłanie informacji do sterownika). Sygnał RS decyduje, czy podawana (bądź odczytywana) informacja jest komendą, czy daną.

Po zasileniu wyświetlacza i wyregulowania napięcia kontrastu (V_0) na pierwszej linii zostają wyświetlone prostokąty. Oznacza to, że sterownik wyświetlacza nie został zainicjalizowany oraz, że napięcie kontrastu (V_0) jest odpowiednie. Przed rozpoczęciem wysyłania właściwych poleceń do sterownika wyświetlacza należy go zainicjalizować. Inicjalizacja sterownika polega na wysłaniu kombinacji odpowiednich poleceń [5]. Zgodnie z dokumentacją techniczną producenta, algorytm inicjalizacji sterownika przedstawiono na rysunku 3.12.



Rys. 3.12. Algorytm inicjalizacji sterownika HD44780 [5]

W programie modułu uruchomieniowego została napisana funkcja inicjalizująca wyświetlacz, zgodnie z algorytmem podanym przez producenta:

```
void lcdini(void)
{
int i,c;
czekaj;
DP3 &= 0x5c00;        //ustawienie portu 3 jako wyjście, 1 - pin jest wyjściem, 0 - wejście
DP3 |=0xa3ff;
P3_clr;        //wyzerowanie portu 3
for(i=0;i<=3;i++){
            enable_set;
            P3 |= 0x0038;
            enable_clr;
            czekaj;
            P3_clr;
            }

 P3_clr;
 enable_set;
 P3 |= 0x0008;
 enable_clr;
 czekaj;
 P3_clr;

 enable_set;
 P3 |= 0x0001;      //wyczyszczenie wyświetlacza
 enable_clr;
 czekaj;
 P3_clr;

 enable_set;
 P3 |= 0x0006; //kursor przesuwa się w prawo po zapisaniu
 enable_clr;
 czekaj;
 P3_clr;

 enable_set;
 P3 |= 0x000c; //włączenie wyświetlacza, bez kursora i migania kursora
 enable_clr;
 czekaj;
 P3_clr;
}
```

Polecenia *enable_set, enable_clr, P3_clr, czekaj* zostały wcześniej w programie zdefiniowane:

```
#define enable_set P3|=0x2000
#define enable_clr P3&=0xdfff
#define   P3_clr P3&=0x5c00
#define czekaj for(c=3000;c>0;c--)
#define RS_set P3|=0x0100;
```

Po zakończeniu procesu inicjalizacji sterownika wyświetlacza, można rozpocząć normalne korzystanie z niego. Tabela 3.8 przedstawia rozkazy sterownika HD44780 [7].

Tab. 3.8. Rozkazy sterownika HD44780 [7]

Instrukcja	Kod instrukcji									
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Display clear	0	0	0	0	0	0	0	0	0	1
Display/cursor home	0	0	0	0	0	0	0	0	1	X
Entry mode set	0	0	0	0	0	0	0	1	I	S
Display On/Off	0	0	0	0	0	0	1	D	C	B
Display/cursor shift	0	0	0	0	0	1	S	R	X	X
Function set	0	0	0	0	1	D	N	F	X	X
CG RAM set	0	0	0	1	Adres pamięci CG ROM					
DD RAM set	0	0	1	Adres pamięci DD RAM						
Busy flag read	0	1	B	Adres pamięci DD RAM						
Data write	1	0	Zapisywany bajt danych							
Data read	1	1	Odczytywany bajt danych							

- Instrukcja „Display clear” powoduje wypełnienie całej zawartości wyświetlacza spacjami, ustawienie kursora w lewym, górnym rogu i wyłączenie trybu przesuwania okna. Instrukcja na wykonanie potrzebuje maksymalnie 1,64 ms;
- „Display/kursor Home” ustawia kursor w lewym górnym rogu wyświetlacza, na pozycji pierwszego znaku. Czas wykonania instrukcji wynosi 1,64 ms;
- „Entry mode set” – określenie trybu pracy wyświetlacza. Ustawienie bitu oznaczonego przez „S” powoduje przesuewanie okna, po wpisaniu znaku, wyzerowanie tego bitu powoduje przesuwanie kursora po wpisaniu znaku. Maksymalny czas trwania tej instrukcji wynosi 40μs;
- „Display On/Off” – ustawienie kolejnych bitów oznaczonych jako D, C, B, powoduje odpowiednio włączenie wyświetlacza, kursowa i migania kursora;
- „Display/kursor Shift” – ustawienie bitu oznaczonego przez S, przesuwa zawartość okna (wartość logiczna „0” powoduje przesuwanie kursora), bit oznaczony jako R decyduje o kierunku przesuwu okna, w prawo bit przyjmuje wartość „1”, analogicznie za przesuwanie w lewo, wartość logiczna bitu R wynosi „0”;
- „Function set” ustawia parametry pracy wyświetlacza. Bit D oznacza jaki interfejs został zastosowany do komunikacji z mikrokontrolerem, wartość bitu D = „1” mówi, że komunikacja odbywa się w trybie 8 – bitowym, D = „0” – komunikacja przebiega w trybie 4 – bitowym. Bit N, decyduje czy wyświetlacza działa jako dwuwierszowy, bądź jednowierszowy, przyjmując odpowiednio wartości „1” i „0”. Bit F należy ustawić w przypadku matrycy złożonej z 5 · 10 pikseli, w przypadku matrycy 5 · 7 należy bit ustawić w stan niski;
- „CG RAM set” – ustawia adres w pamięci generatora znaków;
- **„**DD RAM set” – ustawia adres w pamięci wyświetlacza;
- „Busy flag read” – odczyt stanu flagi zajętości;
- „Data read” – odczyt danych z pamięci wyświetlacza (bądź pamięci CG RAM sterownika);
- „Data write” – zapis danych do pamięci wyświetlacza (bądź pamięci CG RAM).

Do obsługi wyświetlacza modułu uruchomieniowego, napisano osobne procedury: obsługi zapisu danych do wyświetlacza, obsługi przesuwania ekranu, oraz czyszczenia zawartości wyświetlacza. Przy programowaniu wyświetlacza nie wykorzystano możliwości odczytu stanu flagi zajętości.

Procedura zapisu danych do wyświetlacza:

```
void pisanie (char s)
{
 int c;
 P3_clr;
 RS_set;
 enable_set;
 P3 |= s;
 enable_clr;
 czekaj;
}
```

Procedura przesuwania ekranu wyświetlacza:

```
void przesuwanie(void)
{
 int c;
 P3_clr;
 enable_set;
 P3 |= 0x0018;
 enable_clr;
 czekaj;
}
```

Procedura czyszczenia ekranu wyświetlacza:

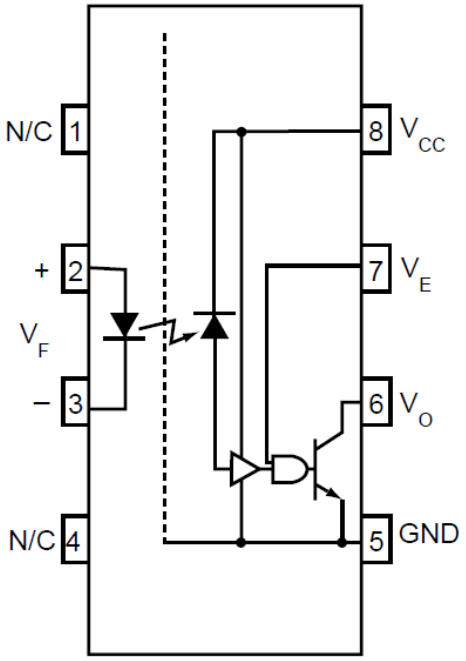
```
void wyczysc(void)
{
 int c,a;
 P3_clr;
 enable_set;
 P3|=0x0001;
 enable_clr;
 for(a=5;a>0;a--){czekaj;}
}
```

Polecenia *enable_set, enable_clr, P3_clr, RS_set, czekaj* zostały wcześniej w programie zdefiniowane:

```
#define enable_set P3|=0x2000
#define enable_clr P3&=0xdfff
#define   P3_clr P3&=0x5c00
#define czekaj for(c=3000;c>0;c--)
#define RS_set P3|=0x0100;
```

3.2.5. Wyjście PWM

Układ modulacji szerokości impulsów pozwala na regulację poziomu napięcia w zakresie od 0 do + 5 V. Parametrami modulacji szerokości impulsów jest rozdzielczość i częstotliwość taktowania. Wyjście PWM zostało zrealizowane poprzez układ 6N137, będący transoptorem. Zapewnia on izolację galwaniczną obwodu wyjścia PWM z mikrokontrolera i obwodu dołączonego do modułu poprzez transoptor. Rozwiązanie takie zapewnia niewielką obciążalność wyjść PWM mikrokontrolera. Zabezpiecza on również mikrokontroler przed zakłóceniami i zwarciami zewnętrznymi. SAB 80c167 posiada 4 wyjścia z możliwością generowania przebiegu PWM. Znajdują się one na pinach P7.0. – P7.3. Wszystkie wyjścia zostały w module wyprowadzone, tak samo wszystkie zabezpieczono transoptorami. Oprócz całkowitej izolacji galwanicznej, transoptor cechuje się bardzo dużą szybkością działania – 10 Mbitów/s i szybkim narostem impulsów, transoptor wymaga również osobnego zasilania strony wtórnej. Umożliwia on współpracę układów wykonanych w technologii CMOS i TTL [5]. Rysunek 3.12 przedstawia schemat wewnętrzny transoptora 6N137.

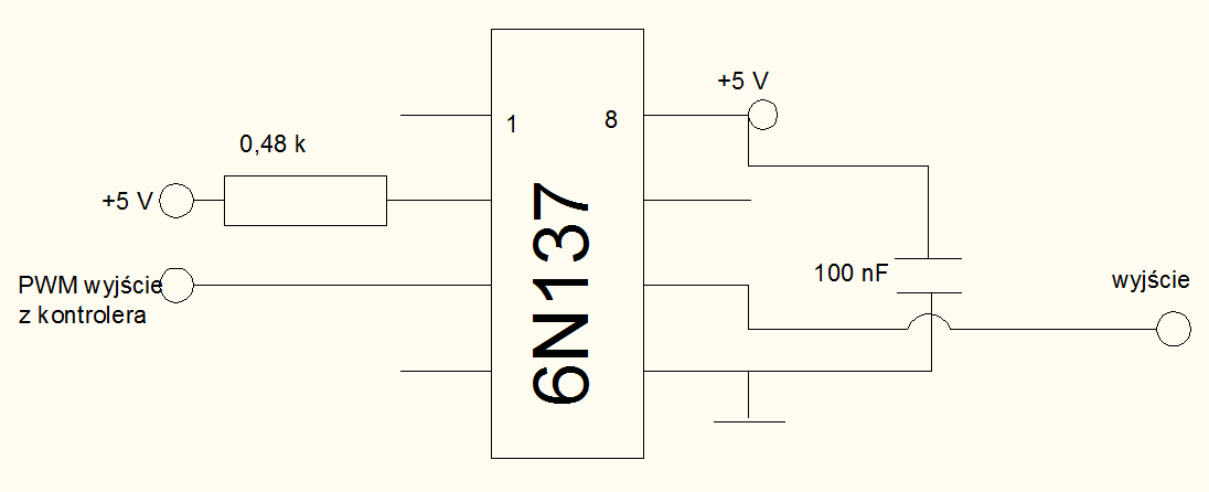


Rys. 3.12. Schemat wewnętrzny transoptora 6N137 [5]

Sposób połączenia transoptora wynika z jego budowy wewnętrznej. Zasilenie nóżki nr 7 oznaczonej jako V_E , powoduje, że na wyjściu (nóżka nr 6 – V_O) otrzymuje się przebieg będący wynikiem dysjunkcji wartości logicznej V_E i wejścia. W module uruchomieniowym nie wykorzystano tej właściwości. Dodatkowo, producent zaleca włączenie równolegle do zasilania obwodu wyjściowego kondensator o wartości około 100nF. Według specyfikacji technicznej zalecanym prądem wejściowym stanu wysokiego jest poziom 15 mA. Aby zapewnić optymalne warunki pracy transoptora dodano rezystory, ograniczające prąd wejściowy dla transoptora. Wartości rezystancji obliczono z przekształconego prawa Oma, według równania (3.3).

$$R = \frac{U}{I} = \frac{5}{0{,}015} = 333{,}3\ \Omega \tag{3.3}$$

Z równania wynika, że wartość rezystorów powinna być większa od 333Ω. W konsekwencji dobrano wartości rezystorów ograniczających na poziomie 480Ω. Zapewnia to prąd wejściowy transoptora na poziomie nie przekraczającym 11mA. Na rysunku 3.13 został przedstawiony schemat połączenia transoptora z mikrokontrolerem.



Rys. 3.13. Schemat połączenia transoptora 6N137

Programowanie wyjść PWM odbywa się (podobnie jak przetwornika A/D) za pomocą nastawiania odpowiednich wartości w rejestrach mikrokontrolera odpowiadających za pracę układu PWM. Obsługa działania pojedynczego kanału PWM sprowadza się do programowania dwóch rejestrów odpowiedzialnych za okres i wypełnienie generowanego przebiegu, oraz dwóch rejestrów obsługujących tryby pracy i działanie kanałów modulacji impulsów. Rejestr odpowiedzialny za okres przebiegu jest oznaczony jako PPx (x – oznacza numer kanału). Jest to rejestr 16 – bitowy, podobnie jak kolejny rejestr PWx – odpowiedzialny za wypełnienie przebiegu. Rejestrami sterującymi są PWMCON0 i PWMCON1. Poniżej przedstawiono schemat rejestru PWMCON0 [3]:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PIR 3	**PIR 2**	**PIR 1**	**PIR 0**	**PIE 3**	**PIE 2**	**PIE 1**	**PIE 0**	**PTI 3**	**PTI 2**	**PTI 1**	**PTI 0**	**PTR 3**	**PTR 2**	**PTR 1**	**PTR 0**
rwh	rwh	rwh	rwh	rw	rw	rw	rw	rw	rw	rw	rw	rwh	rwh	rw	rw

- bity rejestru oznaczone PTRx odpowiadają za połączenie zegara z rejestrem PTx, który jest rejestrem liczącym, odpowiedzialnym za generowany przebieg czasowy. Ustawienie bitu odpowiadającego poszczególnym kanałom oznacza rozpoczęcie odliczania w rejestrze PTRx;
- PTIx – określenie częstotliwości zegara taktującego timer PTx, stan wysoki zadaje obniżoną częstotliwość równą częstotliwości zegara podzielonej przez 64;
- PIEx – pozwolenie na ustawienia flagi przerwania danego kanału (bit ustawiony);
- PIRx – ustawienie bitu powoduje aktywację flagi żądania danego kanału.

Drugim rejestrem kontrolującym działanie wyjść PWM jest rejestr PWMCON1. Poniżej przedstawiono schemat tego rejestru [3]:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PS3	**PS2**	**-**	**PB 01**	**-**	**-**	**-**	**-**	**PM3**	**PM2**	**PM1**	**PM0**	**PEN 3**	**PEN 2**	**PEN 1**	**PEN 0**
rw	rw	-	rw	-	-	-	-	rw	rw	rw	rw	rw	rw	rw	rw

- bity oznaczone PENx odpowiadają za włączenie wyjścia generowanego przebiegu (ustawienie bitu na pozycji odpowiadającej danemu kanałowi aktywuje jego wyjście);
- PMx – ustawienie bitu na pozycji danego kanału wybiera tryb pracy wyjścia PWM (domyślnie przebieg jest wyrównany do krawędzi okresu, ustawienie bitu centruje względem okresu generowany przebieg);
-PBx – ustawienie stanu logicznego „1” przełącza kanał 0 i 1 w tryb „burst mode”;
- PSx – ustawienie stanu wysokiego powoduje pracę wyjścia PWM w trybie „single shot”.

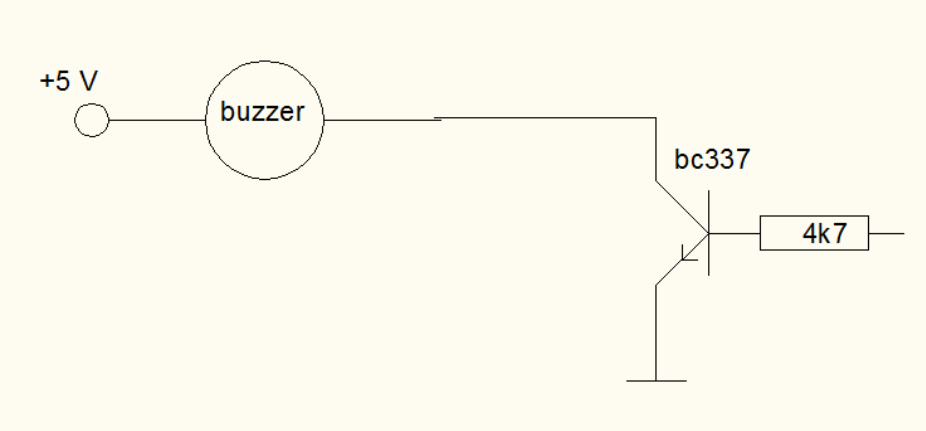
Modulacja szerokości impulsów została zrealizowana w programie przez osobną procedurę [8]:

```
void pwm( int kanal)
{
 DP7=0x000f;          // P7.0 - 7.3 - wyjścia
 if(kanal==0){
                PP0=przetwornik(0);          // okres
                PTR0=1;              // Stert
                PEN0=1;              // zezwolenie na wyjście
                PW0 = 0x0200;
                }
 else if(kanal==1){
                PP1=przetwornik(1);          // okres
                PTR1=1;              // Stert
                PEN1=1;              // zezwolenie na wyjście
              PW1=0x0100;  //wypełnienie
                }
 else if(kanal==2){
                PP2=0x00ff;            // okres
                PTR2=1;              // Stert
                PEN2=1;              // zezwolenie na wyjście
                PW2=0x00bd;                    //wypełnienie
                }
 else if(kanal==3){
                PP3=0x00ff;            // okres
                PTR3=1;              // Stert
                PEN3=1;              // zezwolenie na wyjście
                PW3=0x003f;                    //wypełnienie
                }
}
```

Analizując kod źródłowy można zauważyć, że kanał 0 i 1, zostały wyposażone w możliwość regulacji okresu generowanego przebiegu. Zmiana okresu odbywa się przez zmianę napięcia mierzonego za pomocą przetwornika. Dla kanału 0 i 1 wyjścia PWM zmiana napięcia na odpowiednio kanale 0 i 1 przetwornika A/D, powoduje zmianę czasu trwania okresu. Wypełnienie dla kanału 0 i 1 zostało ustawione jako stałe i wynosi dla kanału 0 wartość, równej połowie maksymalnego napięcia możliwego do zmierzenia przetwornikiem A/D mikrokontrolera. Łatwo zauważyć, że wypełnienie jest więc regulowane od 50% do 100%. Wypełnienie przebiegu dla kanału 1 zostało ustawione na wartość równą jednej czwartej napięcia możliwego do zmierzenia przetwornikiem A/D mikrokontrolera. Wypełnienie jest więc regulowane od 25% do 100%. Parametry (okres i wypełnienie) kanałów 2 i 3 zostały ustawione na stałą wartość i po przeliczeniu wypełnienie wynosi 75% i 25%. Teoretycznie wartości te powinny odpowiadać napięciu 3,75 V i 1,25 V. Dokonano sprawdzającego pomiaru na wejściu transoptora, aby skonfrontować poprawność zaprogramowania PWM. Zmierzone wartości dla kanału 2 i 3 wyniosły odpowiednio 3,45 V i 1,14 V.

3.2.6. Brzęczyk piezoelektryczny, tranzystor BC 337

Projekt modułu uruchomieniowego zakładał możliwość sygnalizacji dźwiękowej. Zostało to zrealizowane poprzez brzęczyk piezoelektryczny. W projekcie uwzględniono, aby nie obciążać pinów mikrokontrolera. Z tego powodu brzęczyk połączono poprzez tranzystor. Użyto tranzystora typu NPN o oznaczeniu BC 337. Jest to tranzystor bipolarny zawarty w obudowie TO – 92. Dopuszczalny prąd przewodzenia tego tranzystora to 0,8 A [5].
Sposób połączenia brzęczyka pokazano na rysunku 3.14.



Rys. 3.14. Schemat połączenia brzęczyka piezoelektrycznego

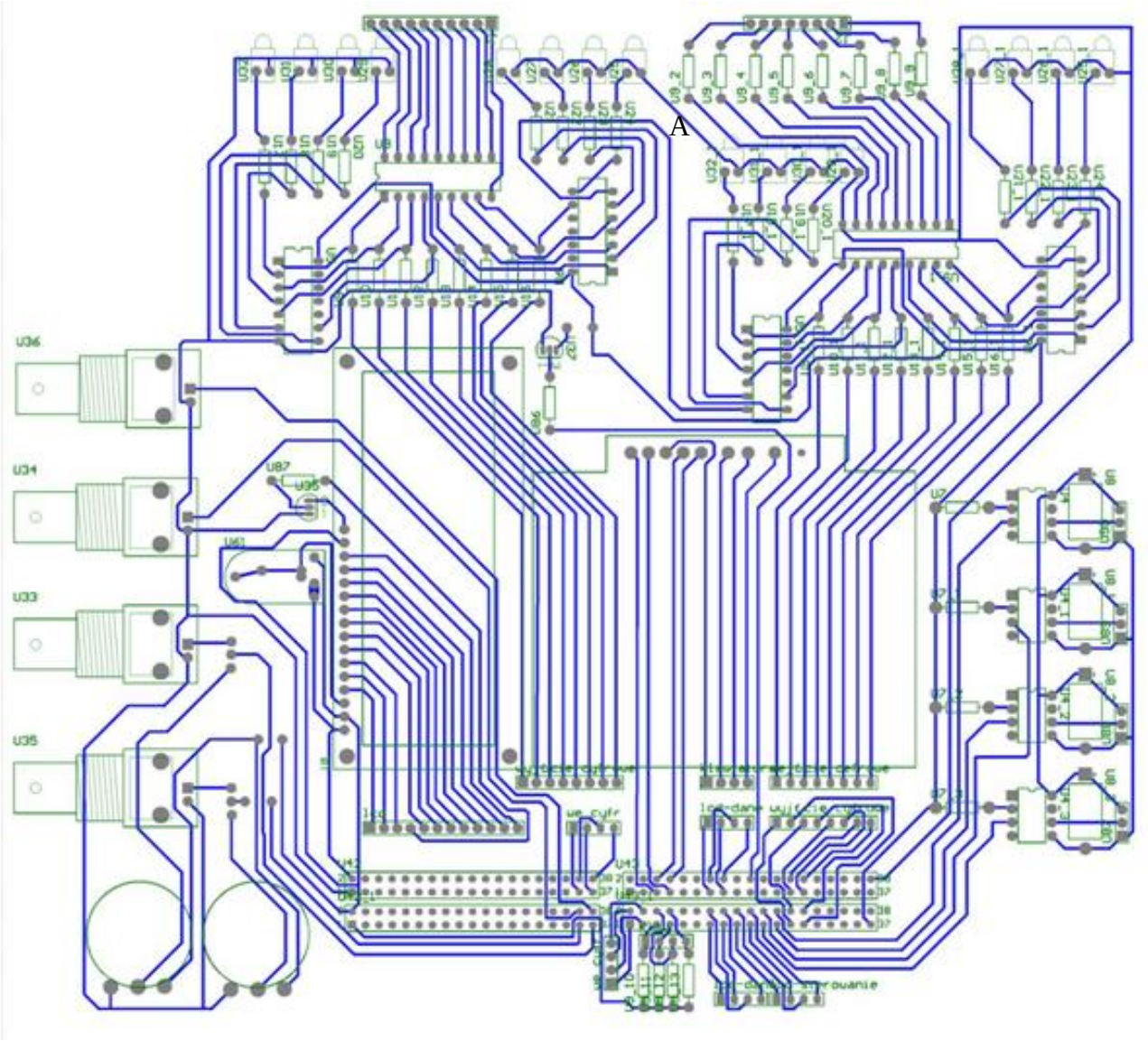
3.2.7. Płytka modułu uruchomieniowego

Płytka modułu uruchomieniowego została zaprojektowana w programie „Altium Designer" (połączenia elektryczne i rozmieszczenie elementów). Połączenie z płytką kitCON – 167 zostało zrealizowane przez taśmy. Na płytce modułu uruchomieniowego zostały wlutowane złącza odpowiadające układem i ilością złączom znajdującym się na płytce z mikrokontrolerem. Tabela 3.9 prezentuje rozkład i funkcje poszczególnych pinów wyprowadzonych z mikrokontrolera, na płytce kitCON – 167 [4].

Tab. 3.9. Rozkład i funkcja złącz umieszczonych na płytce kitCON – 167 [4]

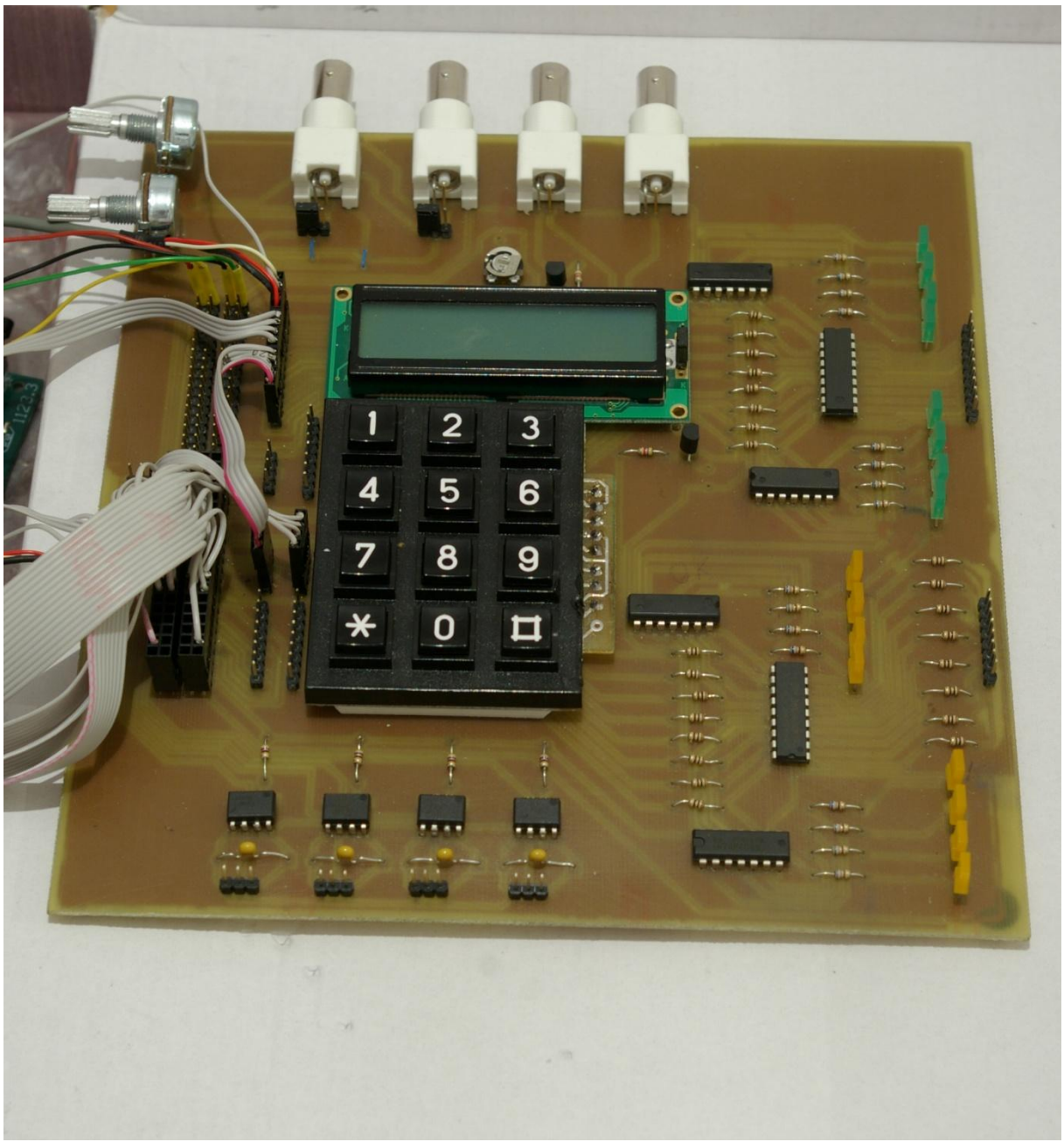
Supply Voltage	PIN 1	VCC	2	VCC	3	GND	4	GND
Data-Bus	5	D0	6	D2	7	D4	8	D6
	9	D1	10	D3	11	D5	12	D7
	13	D8	14	D10	15	D12	16	D14
	17	D9	18	D11	19	D13	20	D15
Address-Bus	21	A0	22	A2	23	A4	24	A6
	25	A1	26	A3	27	A5	28	A7
	29	A8	30	A10	31	A12	32	A14
	33	A9	34	A11	35	A13	36	A15
	37	A16	38	A18	39	A20	40	A22 / TXDC
	41	A17	42	A19	43	A21 / RXDC	44	A23
Control-Signals	45	/RD-U	46	/RD-U	47	/RESO-U	48	/RES-U
	49	/WRL	50	ALE	51	/EA	52	/NMI-U
	53	/CS0(P6.0)	54	/CS2(P6.2)	55	/CS4(P6.4)	56	P6.6
	57	/CS1(P6.1)	58	/CS3(P6.3)	59	/HLD-U	60	P6.7
Special Purose	61	VREF	62	VREF	63	P2.8	64	
	65	VREF	66	VREF	67	VPP	68	
Analog Input	69	P5.0	70	P5.2	71	P5.4	72	P5.6
	73	P5.1	74	P5.3	75	P5.5	76	P5.7
	77	P5.8	78	P5.10	79	P5.12	80	P5.14
	81	P5.9	82	P5.11	83	P5.13	84	P5.15
	85	P2.0	86	P2.2	87	P2.4	88	P2.6
	89	P2.1	90	P2.3	91	P2.5	92	P2.7
Digital-Port P2	93	P2.8	94	P2.10	95	P2.12	96	P2.14
	97	P2.9	98	P2.11	99	P2.13	100	P2.15
Digital-Port P3	101	P3.0	102	P3.2	103	P3.4	104	P3.6
	105	P3.1	106	P3.3	107	P3.5	108	P3.7
	109	P3.8	110	P3.10, TXD0	111	/WRH	112	/RDY-U
	113	P3.9	114	P3.11, RXD0	115	P3.13	116	P3.15
Digital-Port P7	117	P7.0	118	P7.2	119	P7.4	120	P7.6
	121	P7.1	122	P7.3	123	P7.5	124	P7.7
Digital-Port P8	125	P8.0	126	P8.2	127	P8.4	128	P8.6
	129	P8.1	130	P8.3	131	P8.5	132	P8.7
	133		134		135		136	
	137		138		139		140	
	141		142		143		144	
	145		146		147		148	
Supply Voltage	149	VCC	150	VCC	151	GND	152	GND

Po zaprojektowaniu ścieżek elektrycznych, zostały one przeniesione na płytkę z laminatu. Dodatkowo, w celu ułatwienia trawienia ścieżek, miejsca puste wypełniono miedzią. Zabieg taki skraca czas trawienia i zmniejsza ryzyko przetrawienia połączeń. Schemat elektryczny zaprojektowanej płytki z widocznymi konturami elementów został przedstawiony na rysunku 3.15.



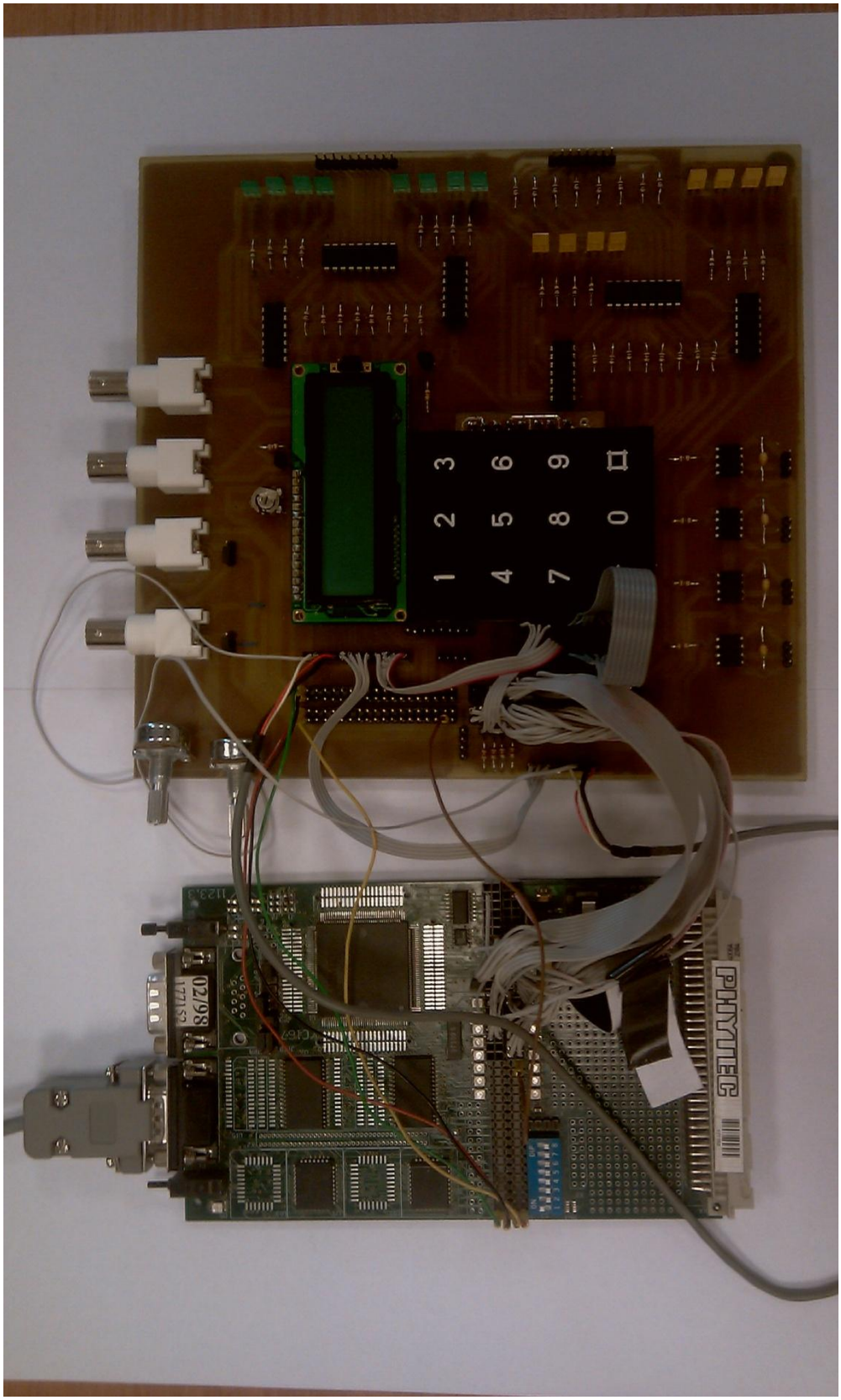
Rys 3.15. Schemat połączeń elektrycznych zaprojektowanego modułu uruchomieniowego

Następnie płytka została wytrawiona, po czym wywiercono otwory na elementy i wlutowano je. Została wybrana przewlekana metoda montażu płytki. Zarówno trawienie jak i montaż został wykonany w Laboratorium Techniki Mikroprocesorowej Politechniki Wrocławskiej. Rysunek 3.16 przedstawia zdjęcie gotowej płytki z wlutowanymi elementami.



Rys. 3.16. Zdjęcie przedstawiające płytkę modułu uruchomieniowego

Połączenie płytki modułu uruchomieniowego z mikrokontrolerem zostało zrealizowane przez wielożyłowe taśmy przewodzące typu FFC (Fexible Flat Cable). Rysunek 3.17 przedstawia sposób realizacji tego połączenia.



Rys. 3.17. Zdjęcie przedstawiające sposób połączenia modułu uruchomieniowego z płytką kitCON – 167

3.2.8. Program obsługujący moduł uruchomieniowy

W celu sprawdzenia poprawności zaprojektowania i budowy modułu, a także przedstawienia działania modułu uruchomieniowego, został napisany program, wykorzystujący wszystkie jego elementy składowe. Program został napisany w języku C. Programowanie odbywało się w środowisku programistycznym „Keil µVision 4". Środowisko to pozwala na programowanie w języku C i w asemblerze. Napisany program pozwala uruchomić sześć różnych demonstracji pokazujących działanie poszczególnych układów modułu uruchomieniowego. Poszczególne demonstracje uruchamia się naciskając klawisze klawiatury. Informacje o działaniu są pokazywane na wyświetlaczu LCD. Naciskając klawisz „0" na klawiaturze program wraca do menu głównego. Naciśnięcie poszczególnych przycisków odpowiada następującym działaniom:
- klawisz „4" – pomiar napięcia przetwornikiem A/D. Możliwy jest wybór kanału. Wynik jest przedstawiany na wyświetlaczu LCD;
- klawisz „5" – powoduje odczytywanie stanu cyfrowego portu wejściowego i odtworzenie dokładnie tego samego stanu na porcie wyjścia cyfrowego;
- klawisz „6" – program umożliwia za pomocą klawiatury zadawanie wartości logicznych na porcie wyjścia cyfrowego;
- klawisz „7" – program po zmierzeniu wartości napięcia na kanale 0 bądź 1, ustawia stan wyjść cyfrowych, tak aby jego wartość w systemie dwójkowym odpowiadała wartości zmierzonego napięcia;
- klawisz „8" – uruchamia obsługę wyjścia PWM. Jest możliwość wyboru kanału wyjściowego;
- klawisz „9" – wartość zmierzonego napięcia jest wyświetlana na wyświetlaczu LCD, a także na cyfrowym porcie wyjściowym.

Poniżej znajduje się kod źródłowy funkcji *main*, programu demonstrującego działanie modułu uruchomieniowego:

```
void main(void)
{
int a,b,c,kanal,st;
double x;
char dosp[100];
char napis1[] = "  Modul uruchomieniowy dla SAB 80c167";
char napis2[] = " 4-A/D pomiar,5-wy.c.(odl),6-wy.c.(kl),  7-wy.c.(A/D), 8-pwm,9-pom+wy.c.";
char napis3[] = "Prosze wybrac                    kanal";
char napisustkl[] = " Prosze naciskac                  klawisze";
char napis5[] =     " Prosze zadawac                  napiecia";

lcdini();    //inicjalizacja wyświetlacza

for(a=0;napis1[a]!=NULL;a++){pisanie(napis1[a]);}      //wrzucenie napisu do LCD
for(b=65;b>0;b--){
                  for(a=250;a>0;a--){czekaj;}
                  przesuwanie();
                  }
wyczysc();      //wyczyszczenie LCD
for(a=0;napis2[a]!=NULL;a++){pisanie(napis2[a]);}   //wrzucenie napisu menu głównego do LCD

//menu główne - pętla główna
while(1){
     b=klaw2();
     for(st=400;st>0;st--){czekaj;}                  //drgania styków
```

```
//uruchomienie pomiaru A/D
if(b==4){                                                    b=0;
            wyczysc();
            for(a=0;napis3[a]!=NULL;a++){pisanie(napis3[a]);}   //wrzucenie napisu do lcd
              while(b<4 || b>7){                                                  b=klaw();
                                                  for(st=400;st>0;st--){czekaj;}}
            kanal=b-4;
              while(b!=0){                                                  x=przetwornik(kanal);
                                                  wyczysc();
                    x/=204;                                  sprintf((char*)dosp,"Kanal %d
                    zmierzyl:                  %lf V",kanal,x);
                    for(a=0;dosp[a]!=NULL;a++){pisanie(dosp[a]);}
                       for(a=0;a<100;a++){dosp[a]=0;}
                    b=klaw2();            //funkcja czytająca przez chwilę klawiaturę
                       for(st=200;st>0;st--){czekaj;}}
            wyczysc();
            for(a=0;napis2[a]!=NULL;a++){pisanie(napis2[a]);}}

    //powielanie wejścia na wyjście cyfrowe
    if(b==5){
            DP8=0xff;
            wyczysc();
            for(a=0;napis5[a]!=NULL;a++){pisanie(napis5[a]);}
            while(b!=0){
              P8=P5>>4;
              b=klaw2();                                          for(st=200;st>0;st--
    ){czekaj;}}
     P8=0xff;
     wyczysc();
     for(a=0;napis2[a]!=NULL;a++){pisanie(napis2[a]);}}

    //ustawianie klawiaturą wartości logicznych na porcie cyfrowym
    if(b==6){
     wyczysc();
            for(a=0;napisustkl[a]!=NULL;a++){pisanie(napisustkl[a]);}
     DP8=0xff;
     while(b!=0){
          b=klaw2();
           for(st=200;st>0;st--){czekaj;}                                  a=b-4;
          if(a>0){P8^=1<<a;}}
     P8=0xff;
     wyczysc();
            for(a=0;napis2[a]!=NULL;a++){pisanie(napis2[a]);}

//ustawianei wartości logicznych na porice cyfrowym przetwornikiem A/D
if(b==7){
     b=0;
            wyczysc();
            for(a=0;napis3[a]!=NULL;a++){pisanie(napis3[a]);}
            while(b<4 || b>5){
                        b=klaw();
                        for(st=400;st>0;st--){czekaj;}}
                        wyczysc();
            kanal=b-4;
```

```
        sprintf((char*)dosp,"Kanal: %d",kanal);
        for(a=0;dosp[a]!=NULL;a++){pisanie(dosp[a]);}
        for(a=0;a<100;a++){dosp[a]=0;}
        DP8=0xff;   //port 8 działa jako wyjście
        while(b!=0){
            x=przetwornik(kanal);
            x/=4;
            P8=x;
            b=klaw2();                                          for(st=200;st>0;st--
){czekaj;}}
        wyczysc();
        for(a=0;napis2[a]!=NULL;a++){pisanie(napis2[a]);}}

       //pwm
       if(b==8){
        wyczysc();
        for(a=0;napis3[a]!=NULL;a++){pisanie(napis3[a]);}
        while(b<4 || b>7){
                b=klaw();
                for(st=400;st>0;st--){czekaj;}}
        kanal=b-4;
        wyczysc();
        sprintf((char*)dosp,"Kanal: %d",kanal);
        for(a=0;dosp[a]!=NULL;a++){pisanie(dosp[a]);}
        while(b!=0){
            pwm(kanal);
            b=klaw2();                                          for(st=200;st>0;st--
){czekaj;}}
        PWMCON0&=0xfff0;//zatrzymanie PTR
        PWMCON1&=0xfff0;//wyłączenie wyjścia
        wyczysc();
        for(a=0;napis2[a]!=NULL;a++){pisanie(napis2[a]);}}

      //pomiar ad i wyjście cyfrowe strowane przetwornikiem
    if(b==9){
        b=0;
        wyczysc();
        for(a=0;napis3[a]!=NULL;a++){pisanie(napis3[a]);}   //wrzucenie napisu do lcd
        while(b<4 || b>7){
                b=klaw();
                for(st=400;st>0;st--){czekaj;}}
        kanal=b-4;
        DP8=0xff;
        while(b!=0){
            x=przetwornik(kanal);
            wyczysc();
            P8=x/4;
            x/=204;
        sprintf((char*)dosp,"Kanal %d zmierzyl:                 %lf V",kanal,x)
for(a=0;dosp[a]!=NULL;a++){pisanie(dosp[a]);}
        for(a=0;a<100;a++){dosp[a]=0;}
        b=klaw2();                                              for(st=200;st>0;st--){czekaj;}}
        wyczysc();
              for(a=0;napis2[a]!=NULL;a++){pisanie(napis2[a]);}}przesuwanie();
              }
              }
```

4. UWAGI I WNIOSKI

Wszystkie założone cele projektu inżynierskiego zostały zrealizowane. Moduł uruchomieniowy został zaprojektowany zgodnie z wcześniejszymi założeniami. Każdy z zastosowanych układów został przetestowany i sprawdzony. Napisany program, demonstrujący działanie modułu dodatkowo sprawdza poprawność projektu i montażu układu. Gruntowana analiza działania układu modułu uruchomieniowego pozwala na stwierdzenie, że ostateczna praca działa zgodnie z założeniami i oczekiwaniem.

Do końcowego projektu wprowadzono dwie zmiany. Dodano dodatkowe połączenie, oznaczone na schemacie elektrycznym (strona 31, rysunek 3.15) literą „A". Powodem był brak połączenia z masą dwóch układów UCY 7404. Zmiana przyniosła oczekiwane efekty. Kolejną zmianą była zmiana poprowadzenia przewodów z sygnałami „enable" i podświetlenia wyświetlacza LCD. Początkowo przewody te zostały dołączone do pinów, odpowiadających za szeregową transmisję danych z komputera do mikrokontrolera. Jakakolwiek próba zapisu tych bitów, kończyła się zerwaniem połączenia komputera z mikrokontrolerem. Sygnał „enable" został przeniesiony na pin P3.13., natomiast sygnał podświetlania LCD przeniesiono na pin P3.15. Zmiany połączenia dokonano na taśmie łączącej płytkę kitCON – 167 z modułem uruchomieniowym. Zmiana połączenia umożliwiła stabilne połączenie mikrokontrolera z komputerem. Brak znaczących poprawek w projekcie dowodzi również poprawności wykonanego projektu połączeń i dobrym montażu układów.

Opracowanie i wykonanie modułu uruchomieniowego pozwoliło na zapoznanie się ze sposobami projektowania i wykonywania systemów współpracujących z mikrokontrolerami. Poznano również specyfikę programowania mikrokontrolerów w języku C, sposób doboru elementów współpracujących z systemami mikroprocesorowymi, a także ich programowaniem.

LITERATURA:

[1] K. Dyrcz, C. Kowalski, Z. Żarczyński, *„Podstawy techniki mikroprocesorowej”*, Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław 1999 r.

[2] P. Gałka, P. Gałka, *„Podstawy programowania mikrokontrolera 8051”*, Wydawnictwo Naukowe PWN, Warszawa 2006 r.

[3] Dokumentacja techniczna mikrokontrolera Siemens SAB 80c167

[4] Dokumentacja techniczna układu Phytec kitCON – 167

[5] Dokumentacje techniczne układów wykorzystanych do realizacji projektu modułu uruchomieniowego (ULN 2803, UCY 7404, LCD LMG – SSC2B16, sterownik LCD HD 44780, transoptor 6N137), instrukcja obsługi multimetru VOREL 81780

[6] http://pl.wikibooks.org/wiki/C - programowanie w języku C, operacje bitowe

[7] http://radzio.dxp.pl – wyświetlacz LCD

[8] http://www.keil.com – przykłady programów dla mikrokontrolera typu c16x

[9] www.easy-soft.net.pl – klawiatura matrycowa, drgania przycisków