Politechnika Wrocławska

KoNaR
KOŁO NAUKOWE ROBOTYKÓW

Urządzenie diagnostyczne

w obudowie telefonu Nokia 2600

inż. Paweł Bardowski

KNSW

2010

Wrocław / Uniejowice

Spis treści

1. WSTĘP NA DOBRY POCZĄTEK

Celem niniejszej pracy było zbudowanie uniwersalnego, podręcznego urządzenia mieszczącego się w ręce, które jest odporne na wszelkie niekorzystne działania ze strony użytkownika lub środowiska, w którym będzie pracowało.

Pomysł na wykorzystanie w tej roli obudowy popularnego telefonu komórkowego wziął się z doświadczeń autora, który podczas wielu pomiarów w terenie, wykorzystując różne urządzenia pomiarowe nieopatrznie nie był w stanie w odpowiedni sposób zabezpieczyć sprzętu. W wyniku takich działań od czasu do czasu ów sprzęt spadał na podłoże betonowe oczywiście w sposób czyniący dla niego największe szkody, a tym samym wywołując całą gamę uczuć u operatora takiego urządzenia. Telefony komórkowe są urządzeniami typu „outside”, zatem przystosowana do złego traktowania obudowa idealnie nadaje się do budowy bardzo odpornych urządzeń. Jej cena (zwłaszcza z używanego sprzętu) jest na tyle niska, że dostępna dla przeciętnego zjadacza chleba. Dodatkowo od razu dostajemy klawiaturę oraz wyświetlacz w komplecie.

W funkcjonalności urządzenia trzeba przede wszystkim pomyśleć o woltomierzu/oscyloskopie. W czasach integracji technologii i usług, gdzie znaczącą rolę odgrywa komunikacja, dobrą cechą urządzenia jest także umiejętność komunikowania się z samochodami pozwalającego w prosty sposób monitorować stan magistrali CAN oraz pojawiających się na niej danych. Ponadto urządzenie to musi być w pełni mobilne. Zastosowanie w motoryzacji skłania do rozszerzenia funkcjonalności urządzenie dodatkowo o interfejs RS-485. Dzięki temu możliwe stanie się badanie, analiza i kontrola systemów będących w fazie projektowania lub rozwoju.

Pomimo, iż protokół CAN jest znany od ponad 20-stu lat, dopiero od niedawna dostrzeżono niewątpliwe zalety tej magistrali: przez swoją niezawodność, odporność na zakłócenia oraz niski koszt implementacji jest coraz częściej wykorzystywana przez konstruktorów i projektantów. Mnogość możliwości jej wykorzystania stwarza również szerokie pole dla rozwoju

urządzeń kontrolujących jej pracę jak i umożliwiających skuteczny wgląd do informacji, jakie są na niej przesyłane.

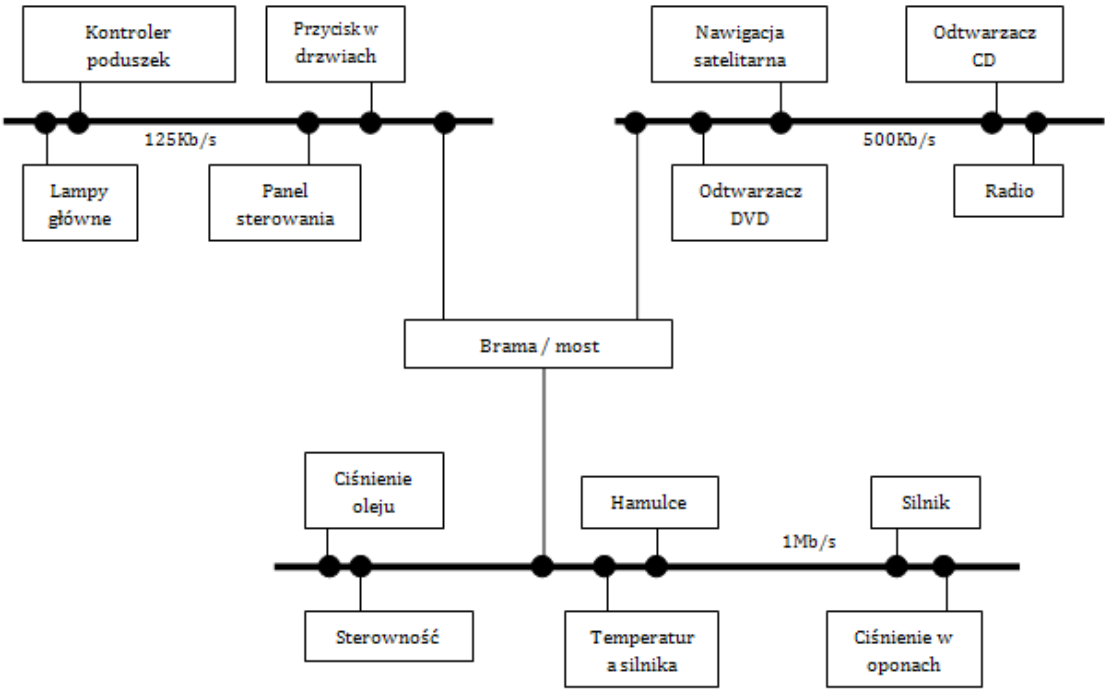
Wczesny rozwój magistrali CAN został głównie wspierany przez przemysł samochodowy, ponieważ wyposażano nim samochody osobowe, łodzie, ciężarówki i inne typy pojazdów. Dzisiaj magistrala CAN jest używana na wielu innych płaszczyznach, włączając w to automatykę przemysłową, aparaturę medyczną, automatykę budynkową, maszyny tkające oraz inne maszyny produkcyjne. Magistrala CAN oferuje skuteczny protokół komunikacji pomiędzy sensorami, urządzeniami wykonawczymi, sterownikami i innymi węzłami w aplikacjach czasu rzeczywistego. Jest znana ze swojej prostoty implementacji, pewności działania oraz dużej wydajności i odporności na błędy.

2. MAGISTRALA CAN

CAN jest protokołem szeregowej magistrali komunikacyjnej rozwijanym przez firmę Bosch (producent sprzętu elektrycznego w Niemczech) od początku lat 80tych. Następnie CAN został ustandaryzowany, jako ISO-11898 i ISO-11519 oraz zadomowił się, jako standardowy protokół w przemyśle samochodowym. We wczesnych latach przemysłu motoryzacyjnego wykorzystywane były pojedyncze mikrokontrolery używane do sterowania różnych urządzeń i podsystemów elektromechanicznych. Poprzez rozpowszechnienie elektroniki, pojazdy zostały wyposażane w magistralę CAN, dzięki czemu wszystkie sterowniki mogły być kontrolowane z centralnego punktu, a w ten sposób powiększając funkcjonalność przez dodanie modułowości oraz uzyskanie bardziej skutecznego procesu diagnostycznego.

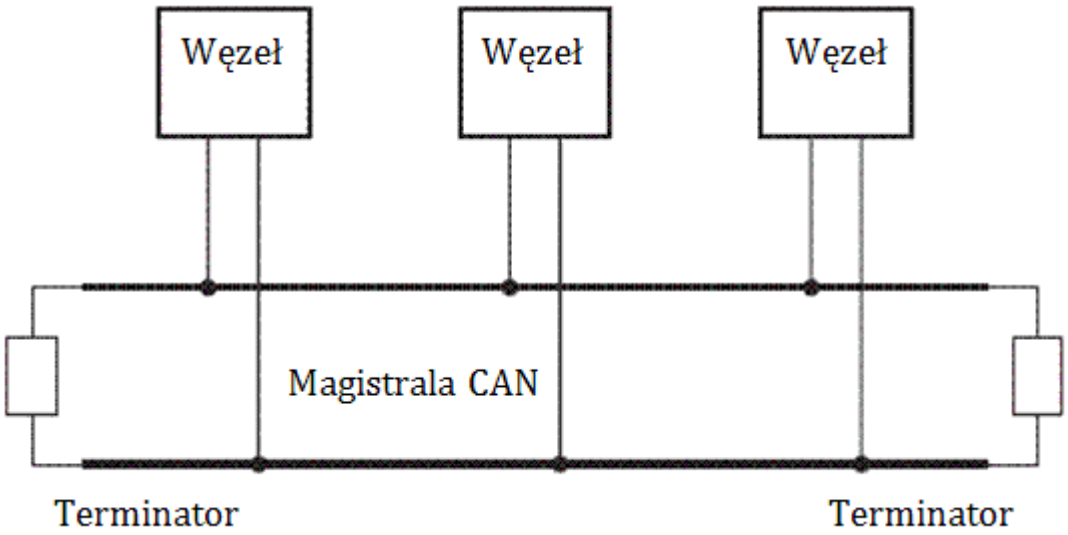
Protokół CAN jest oparty na topologii magistrali, a do komunikacji po szynie CAN wystarczy tani kabel dwużyłowy o impedancji falowej 120Ω. Szyna ma strukturę multimaster, gdzie każde urządzenie na szynie może wysyłać lub otrzymywać dane. Tylko jedno urządzenie może wysyłać dane w jednej chwili, a pozostałe urządzenia ustawiają się w tryb odbioru. Podczas gdy dwa lub więcej urządzeń próbuje wysyłać dane w tym samym czasie, zostaje dopuszczone jedynie jedno, które posiada najwyższy priorytet, a reszta z powrotem przechodzi w tryb odbioru.

Na rysunku 2.1 została przedstawiony typowy przykład magistrali CAN w samochodzie. Jak widać w typowym pojeździe wyposażonym w magistralę występuje więcej niż jedna szyna CAN z różnymi wariantami szybkości. Wolniejsze urządzenia, takie jak kontrola drzwi, sterownik klimatyzacji, moduły informacyjne mogą być dołączone do magistrali o dowolnej szybkości, jednak ze względu na odporność na zakłócenia z reguły wybiera się wolniejsze prędkości. Urządzenia, które wymagają szybszej odpowiedzi, takie jak system ABS, elektroniczny moduł przepustnicy, moduł kontroli przesyłania, są dołączone do szybkiej szyny CAN.



Rysunek 2.1. Typowa aplikacja magistrali CAN w pojeździe.

Przemysł motoryzacyjny wykorzystujący szynę CAN spowodował jej masową produkcję, przez co cena samych sterowników CAN znacząco spadła. Ocenia się, że około 400 milionów modułów CAN jest sprzedawanych każdego roku. Implementacja sterowników w strukturę mikrokontrolerów przyczyniło się do redukcji kosztów produkcji.



Rysunek 2.2. Przykład magistrali CAN.

Rysunek 2.2 przedstawia szynę CAN z trzema terminalami. Protokół CAN jest oparty na protokole CSMA/CD AMP (Carrier-Sense Multiple Access/Collision Detection with Arbitration on Message Priority), który jest podobny do protokołu użytego w sieci LAN Ethernet. Kiedy Ethernet dostrzega kolizję, wysyłające terminale po prostu przestają transmitować dane i czekają przypadkową ilość czasu, po czym wznawiają próbę wysyłania danych. Protokół CAN rozwiązuje problem kolizji używając mechanizmu arbitrażu, gdzie pierwszeństwo do wysyłania danych przypada terminalowi, któremu nadano najwyższy priorytet.

Czas	1	2	3	4	6	7	8	9	10
Stacja A	0	0	0	**1**	0	1	1	0	1
Stacja B	**0**	**0**	**0**	**0**	**1**	**1**	**0**	**1**	**1**
Stacja C	0	0	0	0	1	1	**1**	0	1
Magistrala	**0**	**0**	**0**	**0**	**1**	**1**	**0**	**1**	**1**
				Stacja **A** przechodzi w tryb odbioru			Stacja **C** przechodzi w tryb odbioru		Stacja **B** kontynuuje nadawanie

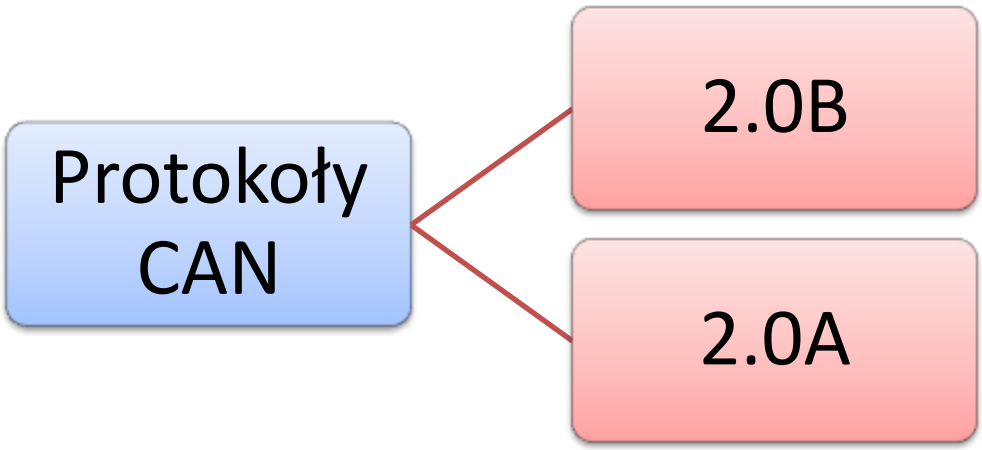
Tabela 2.1. Przykładowy przebieg arbitrażu magistrali CAN.

Tabela 2.1 przedstawia przykładowy przebieg arbitrażu, jaki może wystąpić na magistrali CAN. Poszczególnym momentom czasu ponumerowanym od 1 do 10 przypisane zostały kolejne bity nadawanej wiadomości przez trzy stacje. W momencie, gdy jedna ze stacji nada bit recesywny, a na magistrali będzie bit dominujący wymuszony przez inną stację, taka stacja odczytuje, że na magistrali jest inny stan niż ona nadała, wtedy przechodzi w stan odbioru i nie przeszkadza w dalszej transmisji innym

stacjom. Dzieje się tak w przypadku stacji A oraz C, podczas gdy stacja B kontynuuje transmisje.

Występują zasadniczo dwa typy protokołów CAN: 2.0A oraz 2.0B (rysunek 2.3). CAN 2.0A jest wcześniejszym standardem z 11-bitowym identyfikatorem, natomiast CAN 2.0B jest nowym rozszerzonym standardem z 29-bitowym identyfikatorem.

Sterowniki 2.0B są zupełnie kompatybilne wstecz ze sterownikami 2.0A oraz mogą przyjmować i transmitować wiadomości w obu formatach.



Rysunek 2.3. Podział protokołów CAN.

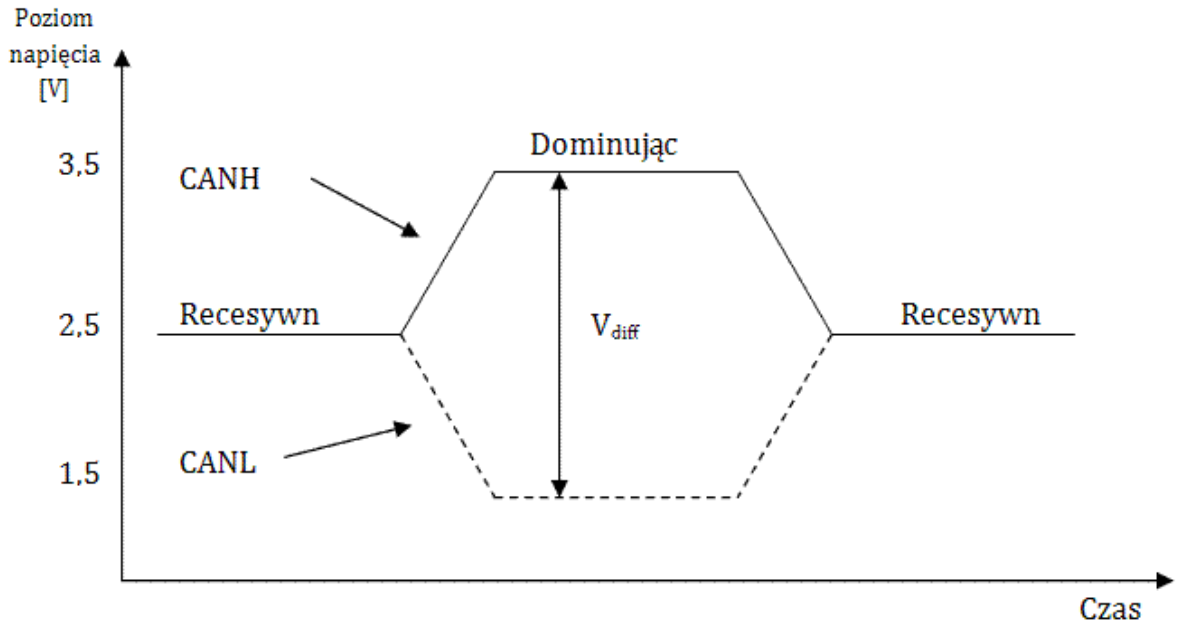
Są dwa typy sterowników 2.0A. Pierwsze są zdolne wysyłać i odbierać tylko wiadomości 2.0A a pojawienie się wiadomości w formacie 2.0B powoduje wystąpienie flagi błędu. Drugi typ sterowników 2.0A (znany jako 2.0B bierny) wysyła i odbiera dane w formacie 2.0A, ale odebranie wiadomości w formacie 2.0B jest ignorowane.

Niektóre cechy protokołu CAN:

- Szyna CAN ma budowę typu multimaster. Kiedy magistrala jest wolna, jakiekolwiek urządzenie podłączone do niej, może rozpocząć transmisję danych.
- Protokół szyny CAN jest elastyczny. Urządzenia podłączone do magistrali nie mają przypisanych adresów, co oznacza, że dane nie są transmitowane od węzła do innego węzła wykorzystując adresacje. Zamiast tego wszystkie terminale w systemie otrzymują każdą wiadomość pojawiającą się na magistrali, a w gestii terminalu pozostaje decyzja czy napływające dane są przydatne i należy je odebrać, czy

odrzucić. Pojedyncza wiadomość może być przeznaczona dla konkretnego terminala lub dla wielu terminali w zależności jak zaprojektowany jest system. Inną korzyścią wynikającą z braku adresacji jest to, że nowe urządzenia mogą być bez przeszkód dodawane lub zabierane bez potrzeby jakiejkolwiek konfiguracji.

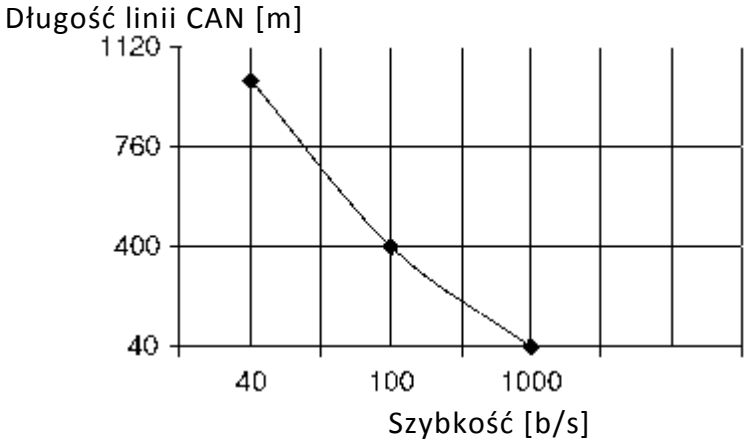
- Szyna CAN oferuje transmisję prośby, co oznacza, że jeden terminal na szynie jest w stanie poprosić o informację inne terminale. W ten sposób zamiast oczekiwać, informacja może zostać wysłana do terminala, kiedy jest mu akurat potrzebna. Na przykład w pojeździe, gdzie temperatura silnika jest ważnym parametrem, system może być zaprojektowany tak, aby wartość temperatury była przesyłanaw sposób cykliczny. Jednakże bardziej eleganckim rozwiązaniem jest poproszenie o temperaturę, kiedy ta informacja jest potrzebna. Rozwiązanie to znacznie zmniejsza ruch na magistrali.
- Szybkość komunikacji na magistrali CAN jest jedna dla danej magistrali. Można ustawić dowolną szybkość dla urządzeń dołączonych do szyny.
- Wszystkie urządzenia na magistrali mają możliwość wykrywania błędów. Urządzenie, które wykryje błąd natychmiastowo powiadamia wszystkie pozostałe urządzenia.
- Do szyny CAN może być dołączonych jednocześnie wiele terminali i nie ma żadnego logicznego ograniczenia co do ich liczby. W praktyce jednak ta liczba urządzeń podłączonych do szyny jest ograniczona czasem propagacji sygnałów oraz pojemnością magistrali.



Rysunek 2.4. Stany logiczne magistrali CAN.

Dane na magistrali CAN mogą przyjmować dwa stary: dominujący oraz recesywny. Na rysunku 2.4 przedstawiony został stan napięć na szynie. Magistrala definiuje logiczny bit 0 jako stan dominujący, a logiczny bit 1 jako recesywny. Podczas gdy na szynie występuje arbitraż, bit o stanie dominującym zawsze zwycięża nad bitem o stanie recesywnym. W stanie recesywnym różnica napięcia pomiędzy liniami CANL i CANH jest mniejsze niż minimalny próg (tj. mniej niż 0,5W na wejściu odbiornika i mniej niż 1,5V na wyjściu nadajnika). W stanie dominującym różnica napięć pomiędzy liniami jest większa od minimalnego progu.

Norma ISO-11898 magistrali CAN wyszczególnia, że urządzenie dołączone do niej musi być w stanie obsłużyć 40-metrowy kabel dla prędkości wynoszącej 1Mb/s. Możliwe jest zastosowanie dłuższych magistrali przy obniżeniu szybkości przesyłanych danych. Rysunek 2.5 przedstawia zależność długości szyny od szybkości komunikacji. Dla przykładu na szynie o długości tysiąca metrów można przesyłać dane z maksymalną szybkością 40Kb/s.

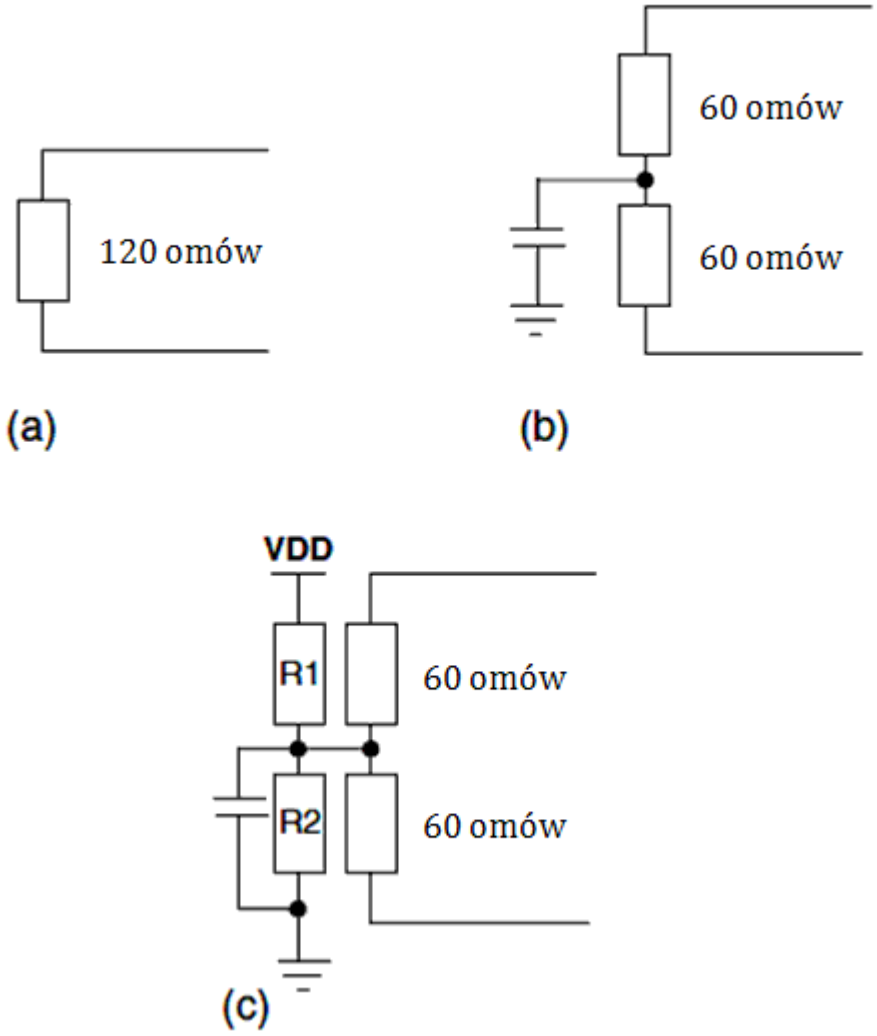


Rysunek 2.5. Zależność długości magistrali od szybkości przysyłanych danych.

Magistrala jest zakończona terminatorami, aby zminimalizować odbicia sygnału na linii. Norma ISO-11898 wymaga, by szyna miała impedancję równą 120 omów. Szyna może być zakończona jedną z następujących metod:

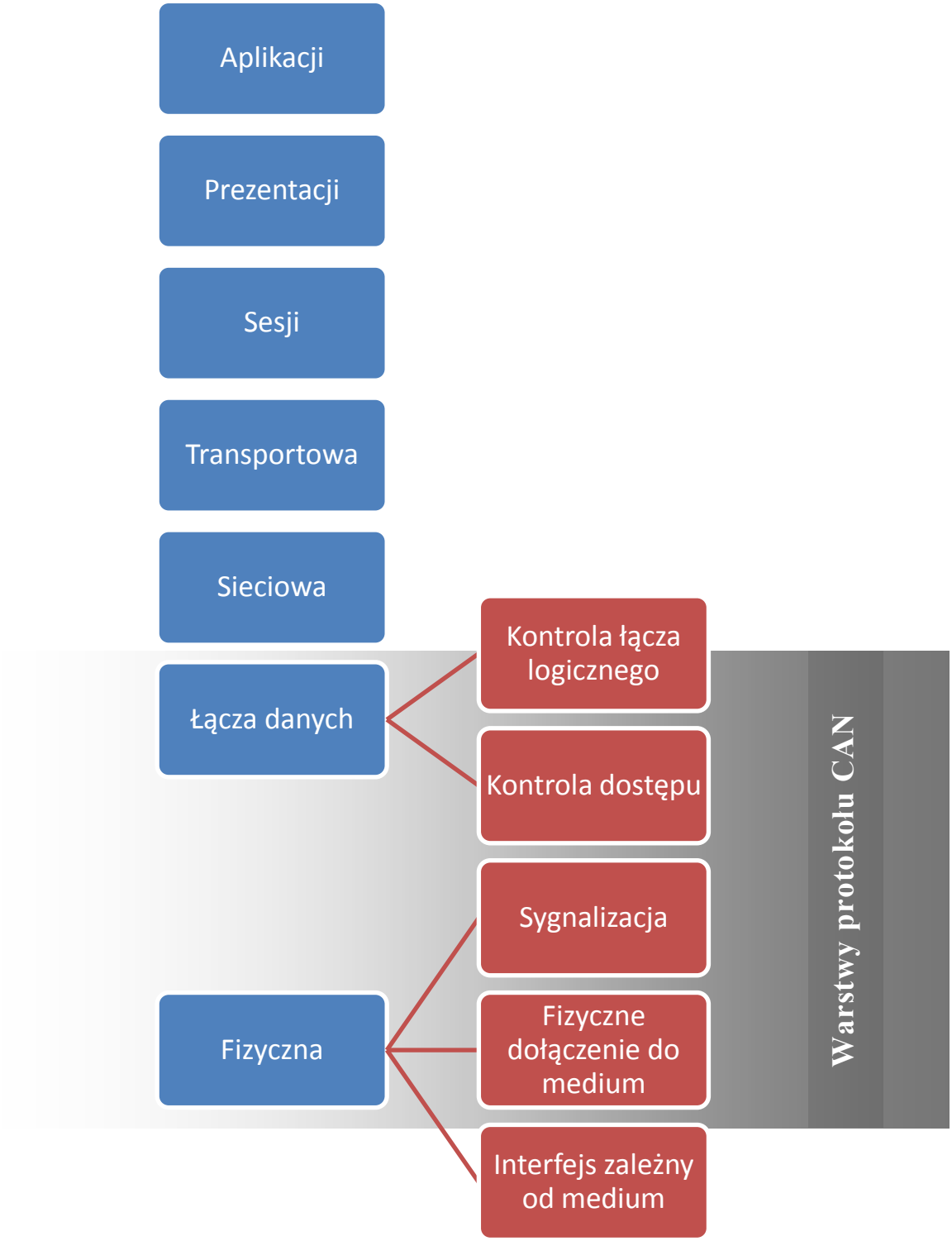
a) standardowy terminator,
b) podzielony terminator,
c) przesunięty podzielony terminator.

W standardowym zakończeniu, metodzie, która jest najbardziej popularna jest użyty 120-omowy opornik obu końcach szyny został przedstawiony na rysunku 2.6 (a). W przypadku podzielonego terminatora, końce magistrali są podzielone na dwa pojedynce rezystory o wartości 60 omów, ten przypadek widać na rysunku 2.6 (b). Dzięki tej metodzie uzyskuje się mniejszą emisję elektromagnetyczną przez co ta metoda zyskuje coraz większą popularność. Metoda przesuniętego podzielonego terminatora jest bardzo podobna do poprzedniej, ale oprócz tego występuje w niej układ rozdzielacza napięcia oraz kondensator – rysunek 2.6 (c). Ta metoda zwiększa wydajność osiągów EMC magistrali.



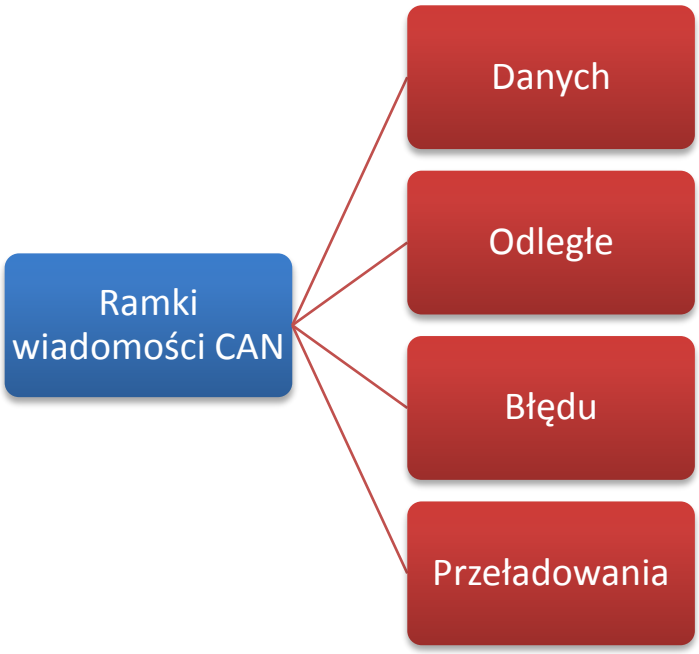
Rysunek 2.6. Metody zakończenia magistrali.

Wiele protokołów sieciowych jest opisywanych za pomocą siedmio-warstwowego modelu OSI (Open Systems Interconnection). Protoków CAN zawiera warstwę łącza danych oraz warstwę fizyczną modelu odniesienia OSI (pokazuje to rysunek 2.7). Warstwa łącza danych składa się z bloku kontroli łącza logicznego oraz bloku kontroli dostępu. Kontrola łącza logicznego zarządza powiadomieniami o przeciążeniach, filtracją oraz kontrolą danych. Natomiast blok kontroli łącza logicznego zajmuje się zarządzeniem zakapsułkowanych danych, kodowaniem ramek, detekcją błędów oraz szeregowaniem/deszeregowaniem danych. Warstwa fizyczna składa się z bloku sygnalizacji, fizycznego dołączenia do medium oraz odpowiedniego interfejsu.



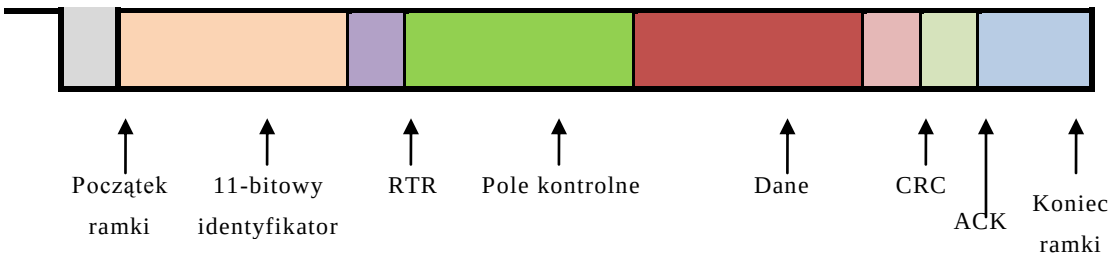
Rysunek 2.7. CAN, a model OSI.

Zasadniczo są cztery ramki wiadomości CAN, co ilustruje rysunek 2.8. Ramki danych oraz zdalne są ustawiane przez użytkownika, natomiast dwie pozostałe są ustawiane przez sprzętowy kontroler CAN.



Rysunek 2.8. Ramki wiadomości CAN.

Ramka danych występuje w dwóch formatach: standardowym (mającym 11-bitowy identyfikator ID) oraz rozszerzonym (mającym 29-bitowy ID). Ramka danych używana jest przez transmitujące urządzenie by wysyłać dane do urządzeń odbierających. Ramka ta jest najważniejszą ramką obsługiwaną przez użytkownika. Strukturę ramki danych widać na rysunku 2.9.



Rysunek 2.9. Standardowa ramka danych.

Standardowa ramka danych zaczyna się bitem początku (SOF), po czym następuje 11-bitowy identyfikator oraz bit zdalnego żądania transmisji RTR. Identyfikator wraz z bitem RTR tworzą 12-bitowe pole arbitrażu. Pole kontrole ma długość 6 bitów i pokazuje jak wiele bajtów danych znajduje się w polu dane. Pole danych może mieć wielkość od 0 do 8 bajtów. Po tym polu występuje pole CRC, które sprawdza, czy odebrana sekwencja bitów jest poprawna. Pole ACK jest złożone z dwóch bitów i jest wykorzystane przez

nadajnik, aby otrzymać potwierdzenie poprawnej ramki, z odbiornika. Koniec ramki zajmuje 7 bitów polem EOF.

W rozszerzonej ramce danych, pole arbitrażu jest szerokie na 32 bity. (29-bitowy identyfikator; 1 bit IDE, by zdefiniować wiadomość, jako rozszerzoną; 1 bit SRR, który jest nie używany; 1 bit RTR).

Zamierzeniem odległej ramki jest ubieganie się o transmisję odpowiedniej ramki danych. Np. węzeł A transmituje ramkę odległą z polem arbitrażu równym 234, wtedy węzeł B, jeżeli został poprawnie zainicjalizowany, może odpowiedzieć ramką danych z takim samym polem arbitrażu. Odległe ramki mogą być używane do implementacji zarządzania ruchem na magistrali typu żądanie-odpowiedź. W praktyce odległe ramki są niezbyt często używane [9].

Ramka błędu jest specjalną wiadomością, która narusza porządek ramki CAN. Jest ona transmitowana, kiedy węzeł wykryje błąd i spowoduje, że wszystkie pozostałe węzły też wykryją błąd i wyślą ramki błędu. Wtedy nadajnik automatycznie wznowi transmisje wiadomości. Jest wymyślny schemat liczników błędów, które zapewniają, że węzeł nie może zablokować ruchu na magistrali przez powtarzanie transmisji ramek błędów [9].

3. MAGISTRALA RS-485

Często w wielu systemach wykorzystywany jest inny interfejs niż CAN, po którym wymieniane są informacje. Standard RS-485 powstał w 1983.

Zaletą RS-485 jest transmisja różnicowa realizowana za pomocą skrętki dwuprzewodowej, ponieważ w takim przypadku zewnętrzne zakłócenia jednakowo oddziałują na obie linie sygnałowe. Związany z tym sygnał wspólny jest eliminowany na wejściu różnicowym odbiornika. Jest to główny powód, dla którego standard RS-485 jest wykorzystywany w rozległych sieciach i w trudnych warunkach przemysłowych, gdzie mogą wystąpić zewnętrzne zakłócenia transmisji [7].

Właściwości NADAJNIKÓW umożliwiające wielopunktową komunikację [6]:

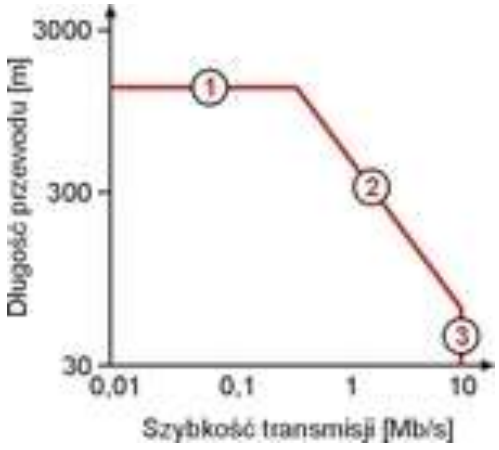
- jeden nadajnik może sterować do 32 jednostkowych obciążeń (obwody wprowadzające obciążenie do 1[mA]) oraz zastępczą rezystancją dopasowywującą Rt=60[ohm] lub większą
- prąd upływu nadajnika w stanie wyłączenia nie może przekraczać 100[mikroA]
- nadajnik powinien zapewnić różnicowe napięcie wyjściowe z przedziału (-1.5[V] do 5[V]) przy obecności napięcia wspólnego z zakresu (-7[V] do 12[V])
- nadajniki muszą mieć zabezpieczenie przed kolizją, jednoczesne nadawanie przez więcej niż jeden nadajnik nie może uszkodzić nadajnika

Właściwości ODBIORNIKÓW umożliwiające wielopunktową komunikację [6]:

- rezystancja wejściowa >=12[kohm]
- zakres napięcia wspólnego na wejściu odbiornika (-7[V] do 12[V])
- czułość wejścia różnicowego 200[mV] w całym zakresie napięcia wspólnego.

Dopuszczalna długość magistrali jest determinowana przez straty sygnału w linii transmisyjnej oraz rozsynchronizowanie przy danej szybkości transmisji. Niezawodność transmisji gwałtownie spada przy rozsynchronizowaniu bitów

danych powyżej 10% czasu trwania bitu. Na rysunku 3.1 przedstawiono wykres zależności długości przewodu od prędkości transmisji w standardzie RS-485 przy rozsynchronizowaniu 10% [7].



Rysunek 3.1. Dopuszczalna długość przewodu przy określonej szybkości transmisji

RS-485 stał się w przemyśle popularnym standardem z kilku powodów. Zapewnia on możliwość transmisji charakteryzującej się dużą odpornością na zaburzenia, możliwością występowania napięć wspólnych w szerokim zakresie (od –7V do 12V) oraz dużą szybkością transmisji nawet przy znacznych długościach magistrali. Ta ostatnia dodatkowo zwiększana może być przy użyciu specjalnych obwodów korekcyjnych w odbiorniku [8].

RS485 spełnia wymagania pełnej sieci wielodostępowej, a standard ten obsługuje do 32 nadajników i odbiorników na pojedynczej (2-przewodowej) szynie. Wraz z wprowadzeniem "automatycznych" repetytorów oraz wysoko impedancyjnych nadajników / odbiorników "ograniczenie" to może być zwiększone do setek (a nawet tysięcy) węzłów w sieci. RS485 zwiększa zakres "trybu wspólnego/powszechnego" (common mode) zarówno dla nadajników jak i odbiorników w trybie "trójstanowym" z możliwością wyłączenia napięcia. Nadajniki RS485 umożliwiają również przeciwstawianie się problemowi "kolizji danych" (szyna natłoku danych) oraz błędnego działania szyny [12].

W celu rozwiązania problemu "kolizji danych", często pojawiającego na magistralach wielodostępnych (konwertery, repetytory, kontrolery mikroprocesorowe) stosuje się następująca budowę. Urządzenia pozostają w trybie odbiorczym tak długo, aż będą gotowe na nadawanie danych. Systemy z pojedynczym urządzeniem nadrzędnym (master) (dostępne jest wiele innych schematów komunikacyjnych) oferują bezpośrednie oraz proste metody

unikania "kolizji danych" w typowym dwuprzewodowym, jednokierunkowym (half-duplex), wielodostępowym systemie. Urządzenie nadrzędne (master) nadaje prośbę o rozpoczęcie komunikacji do węzła podrzędnego (slave) poprzez zaadresowanie tej jednostki. Osprzęt (hardware) wykrywa bit startowy transmisji i automatyczne uaktywnia (w locie) nadajnik RS485. Gdy znak jest nadany osprzęt (hardware) powraca do trybu odbioru w czasie ok. 1-2 mikrosekund (przynajmniej z konwerterami, repetytorami R.E. Smith oraz układem zdalnych wejść/wyjść (I/O)) [12].

Możliwe jest wysłanie dowolnej ilości znaków, a nadajnik automatycznie przełączy się przy nadaniu kolejnego znaku, wliczając każdą prędkość nadawania i/lub każdą specyfikację komunikacji np. 9600,N,8,1). Gdy jednostka podrzędna (slave) jest już zaadresowana może natychmiast odpowiedzieć. Wynika to z bardzo szybkiego czasu wyłączenia nadajnika. Nadajnik wyłącza automatyczne urządzenie. Nie jest konieczne wprowadzanie dużych opóźnień do sieci, w celu uniknięcia "kolizji danych". Ponieważ opóźnienia nie są wymagane, można konstruować sieci wykorzystujące w 100% szerokość pasma do przesyłania danych [12].

4. MIKROKONTROLERY Z RDZENIEM ARM

4.1. Architektura

Historia powstania mikroprocesorów ARM wywodzi się z lat 80–tych, kiedy to grupa inżynierów pod kierownictwem Rogera Wilsona i Steave`a Ferbera, pracująca dla firmy Acorn, rozpoczęła projektowanie rdzenia będącego rozwinięciem znanego mikroprocesora 6502 firmy MOS Technology. Firma Acorn budowała komputery w oparciu o ten procesor, więc celem było opracowanie nowego, wydajniejszego procesora, który miał się charakteryzować podobną architekturą do 6502. Było to duże wyzwanie, ponieważ rok wcześniej zespół ten zajmował się projektowaniem mikroprocesorów 8–bitowych z pamięcią programu o pojemności 32 kB. Pierwsza wersja testowa mikroprocesora ujrzała światło dzienne w 1985 roku, a już rok później ukończono wersję produkcyjną ARM2, który był 32–bitowym mikroprocesorem z 26–bitową przestrzenią adresową. W późnych latach 80–tych firma Acorn rozpoczęła współpracę z firmą Apple w celu opracowania udoskonalonego rdzenia ARM. W tym samym czasie z firmy Acorn wydzieliła się firma ARM Ltd. Efektem współpracy było powstanie mikroprocesora ARM6 zastosowanego w palmtopie Apple Newton. Obecnie firma ARM zajmuje się sprzedażą licencji na aplikowanie rdzeni rodziny ARM [1].

Zestaw instrukcji procesora ARM stanowi rozwinięcie zestawu instrukcji MOS 6502. Główne zmiany dotyczą zwiększenia efektywności potokowego przetwarzania instrukcji[1]. Zgodnie z założeniami architektury RISC[2], rozkazy są tak skonstruowane, aby wykonywały się w ściśle określonym czasie - zwykle w jednym cyklu maszynowym[3] [2].

[1] Jeden ze sposobów w jaki uzyskuje się większą szybkość wykonywania kodu poprzez równoległe przetwarzanie danych informatycznych.
[2] RISC (ang. Reduced Instruction Set Computers) - nazwa architektury mikroprocesorów, która charakteryzuje się zredukowaną listą rozkazów.
[3] Cykl maszynowy to cykl, podczas którego następuje wymiana danych między procesorem a pamięcią lub układem wejścia wyjścia (odczyt albo zapis).

4.2. Rdzeń ARM7tdmi

Rdzeń został wyposażony w 31 32-bitowych rejestrów, z których każdy jest równouprawniony i na każdym z nich możliwe jest wykonywanie dowolnych operacji z listy rozkazów. Ponieważ licznik rozkazów jest również rejestrem (R15), przez to procesor potrafi zaadresować do 4GB pamięci. Na rysunku 4.3 przedstawiono przykładowy potok instrukcji. Wypełnianie instrukcjami zajmuje 3 cykle zegarowe, po tym czasie instrukcje wykonywane są w ciągu jednego cyklu, wykorzystując równoległe wykonywanie poszczególnych poziomów potoku, tj. pobieranie, dekodowanie i wykonywanie instrukcji. Sporym problemem w wykonywaniu programu wykorzystując potok są rozgałęzienia. W przypadku, kiedy wystąpi jakieś rozgałęzienie potok musi być wyczyszczony oraz od nowa zapełniony nowymi instrukcjami, tracąc przy tym czas wykonywania instrukcji. Architektura procesorów ARM radzi sobie z tym problemem w bardzo interesujący sposób. W zależności od stanu wykonywanego programu prawie każda instrukcja może zostać wykonana warunkowo. Efektem działania takiego kodu jest minimalizacja rozgałęzień oraz większa efektywność jego wykonywania.

Rdzeń ARM7TDMI ma architekturę, w której rozkazy i dane należą do jednej przestrzeni adresowej.

Pamięć może być adresowana w postaci 8, 16 lub 32 słów danych, przy czym procesor zapisuje dane w porządku big–endian[4] lub little–endian[5] [1].

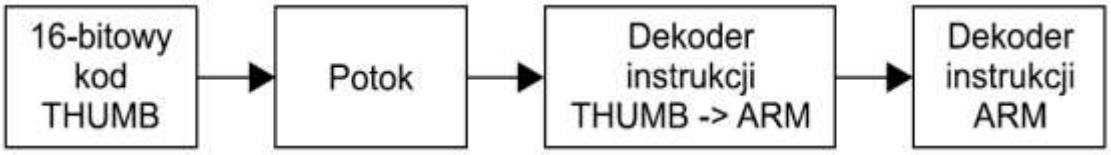
Instrukcja	ADDS R0, R4, R8	ADCS R1, R5, R9	ADCS R2, R6, R10	ADCS R3, R7, R11	ORR R0, R0, #3	ORR R2, R2, #3
Pobranie		ADDS R0, R4, R8	ADDS R0, R4, R8	ADCS R2, R6, R10	ADCS R3, R7, R11	ORR R0, R0, #3
Dekodowanie			ADDS R0, R4, R8	ADDS R0, R4, R8	ADCS R2, R6, R10	ADCS R3, R7, R11
Wykonanie				ADDS R0, R4, R8	ADDS R0, R4, R8	ADCS R2, R6, R10

Rysunek 4.1. Sposób wykonywania rozkazów przez procesory z rdzeniem ARM [1]

W mikrokontrolerze STR731FV2, jaki został wykorzystany w urządzeniu zaimplementowany został system pamięci flash zorganizowany w komórki

[4] Big endian to forma zapisu danych, w której najbardziej znaczący bajt (zwany też grubym bajtem, z ang. high-order byte) umieszczony jest, jako pierwszy.
[5] ittle endian to forma zapisu danych, w której mniej znaczący bajt (zwany też dolnym bajtem, z ang. low-order byte) umieszczony jest, jako pierwszy.

pamięci o szerokości 32-bitów, który umożliwia przechowywanie kodu programu jak i stałych danych. Pamięć jest obsługiwana przez CPU bez żadnych opóźnień przy maksymalnej częstotliwości taktowania 36 MHz.



Rysunek 4.2. Sposób dekodowania rozkazów THUMB [1]

W rdzeniu ARM7 zastosowano dwa podzbiory rozkazów jakie może wykonywać. Różnią się one między sobą gęstością upakowania kodu oraz szybkością wykonywania. Do pierwszej grupy należą instrukcje 32-bitowe ARM. Są one na bieżąco wykonywane przez procesor bez żadnych przekształceń. Zapewnia to dużą wydajność w działaniu oraz lepsze zarządzanie dużymi obszarami pamięci. Drugą grupę instrukcji stanowią instrukcje 16-bitowe o nazwie THUMB. Rozkazy te są podzbiorem rozkazów ARM, ale przy wykonywaniu są one tłumaczone do pełnej 32-bitowej instrukcji ARM, co przedstawia rysunek 4.4. Powoduje to wolniejsze działanie kodu, ale gdy decydującym parametrem jest objętość kodu, znacznie przewyższają pod tym względem rozkazy ARM.

5. ZAŁOŻENIA WSTĘPNE

Na świecie sprzedawane są miliony telefonów firmy Nokia, poprzez tak ogromną popularność tych urządzeń produkcja na masową skalę jest bardzo ekonomiczna. Wykorzystując ten fakt możliwe stało się uzyskanie bardzo dobrej obudowy zaprojektowanej wręcz do celów przenośnych i do pracy w trudnych warunkach oraz wiele innych części takich jak wyświetlacz, przyciski, itp. Przez zastosowanie obudowy z telefonu Nokia 2600 urządzenie stało się bardzo odporne na niekorzystne warunki pracy, małe i poręczne. Takie rozwiązanie wymusiło jednak konieczność zastosowania elementów w technice SMD[6] oraz maksymalną miniaturyzację wszystkich komponentów, zaczynając od projektu płytki a kończąc na fizycznej wielkości użytych elementów. Pomimo ograniczonej przestrzeni zbudowane urządzenie ma następujące cechy:

- kolorowy wyświetlacz graficzny 132x132 piksele,
- podświetlana klawiatura matrycowa,
- zasilanie akumulatorowe (Li-Ion),
- 2 kanały CAN: odbiór oraz nadawanie,
- kanał CAN odbiór pracujący z dowolną polaryzacją,
- obsługa interfejsu RS-485,
- prosty oscyloskop/woltomierz,
- graficzny intuicyjny interfejs użytkownika,
- przycisk do szybkiego wysyłania ramek CAN,
- wymienne panele.

Budowa urządzenia całkowicie zmienia właściwości telefonu Nokia 2600 tym samym zachowując jego wygląd i większość cech fizycznych. Jednakże wiele z komponentów przerobiono tak, aby pasowały do nowych funkcji urządzania.

Na rysunku 5.1 jest przedstawiony schemat blokowy urządzenia. Można z niego wyodrębnić 7 głównych bloków, w których skład wchodzą: wyświetlacz, kontroler, zasilanie, klawiatura, can, RS-485 oraz przetwornik A/C.

[6] SMD - ang. Surface Mount Device

Rysunek 5.1. Schemat blokowy urządzenia

Blok zasilanie obejmuje swoim zasięgiem pozostałe bloki. Wszystkie bloki wymagają napięcia zasilania 5V, natomiast blok wyświetlacza potrzebuje dodatkowo 3,3V oraz około 6V. Wszystkie te napięcia dostarcza blok zasilania z akumulatora Litowo-jonowego. Do zapewnienia takie funkcjonalności wyposażony jest w dwie przetwornice napięcia oraz jeden stabilizator liniowy. Tutaj także znajduje się włącznik urządzenia doprowadzający napięcie z akumulatora.

Blok kontrolera to mikrokontroler STR731FV2 oraz wszystkie elementy niezbędne do jego poprawnej pracy. Do tego bloku dołączone są pozostałe współpracujące z mikrokontrolerem bloki.

W bloku CAN znajdują się 3 sterowniki linii CAN oraz wyprowadzenia do dołączenia urządzenia do badanej magistrali.

Blok RS-485 zawiera sterownik linii oraz wyprowadzenia do dołączenia urządzenia do linii.

Przetwornik A/C składa się z dzielnika napięcia oraz zewnętrznego szybkiego przetwornika A/C. Pomimo, że mikrokontroler posiada już

wbudowany przetwornik okazał się on zbyt wolny oraz mało dokładny do tego zastosowania.

Klawiatura składa się z matrycy 19-stu przycisków oraz układu podświetlania.

Ostatnim blokiem jest blok wyświetlacza. W tym bloku poza samym wyświetlaczem jest również zastosowany scalony konwerter napięcia 3,3V na 5V i odwrotnie.

6. SPRZĘTOWA REALIZACJA PRACY

6.1. Mikrokontroler STR731FV2

6.1.1. Wybór mikrokontrolera

Do budowy urządzenia diagnostycznego magistrali CAN potrzebny był mikrokontroler, który sprosta wielu wymaganiom, jakie stawia projekt:

a) wbudowany kontroler CAN,
b) odpowiednia ilość portów wejścia/wyjścia,
c) wystarczająca moc obliczeniowa potrzebna do obsługi wszystkich peryferii,
d) niski pobór prądu,
e) duża ilość pamięci flash oraz ram.

Rysunek 6.1. Obudowa wybranego mikrontrolera

Ze względu na spełnienie powyższych warunków oraz wiele innych zalet takich jak 3 moduły CAN, 256kB pamięci flash oraz małą 100-wyprowadzeniową obudowę widoczną na rysunku 6.1 (ważna cecha ze względu na znacznie ograniczone rozmiary urządzenia) wybrano mikrokontroler **STR731FV2.** Tabela 6.1 przedstawia wybrany mikrokontroler (zacieniowany obszar) w porównaniu do pozostałych jednostek z rdzeniem ARM7TDMI.

	STR730FZx		STR735FZx		**STR731FVx**			STR736FVx		
Pamięć flash - bajty	128K	256K	128K	256K	64K	128K	**256K**	64K	128K	256K
RAM - bajty	16K				16K					
Peryferia	10 liczników TIM, 112 I/O, 16 kanałów ADC				6 liczników TIM, 72 I/O, 12 kanałów ADC					
Moduły CAN	3		0		**3**			0		
Napięcie zasilania	4,5 do 5,5 V									
Temperatura pracy	-40 do +85°C / -40 do +105°C									
Obudowa	TQFP144 20 x 20				**TQFP100 14 x 14**					

Tabela 6.1. Cechy rodziny mikrokontrolerów STR7x

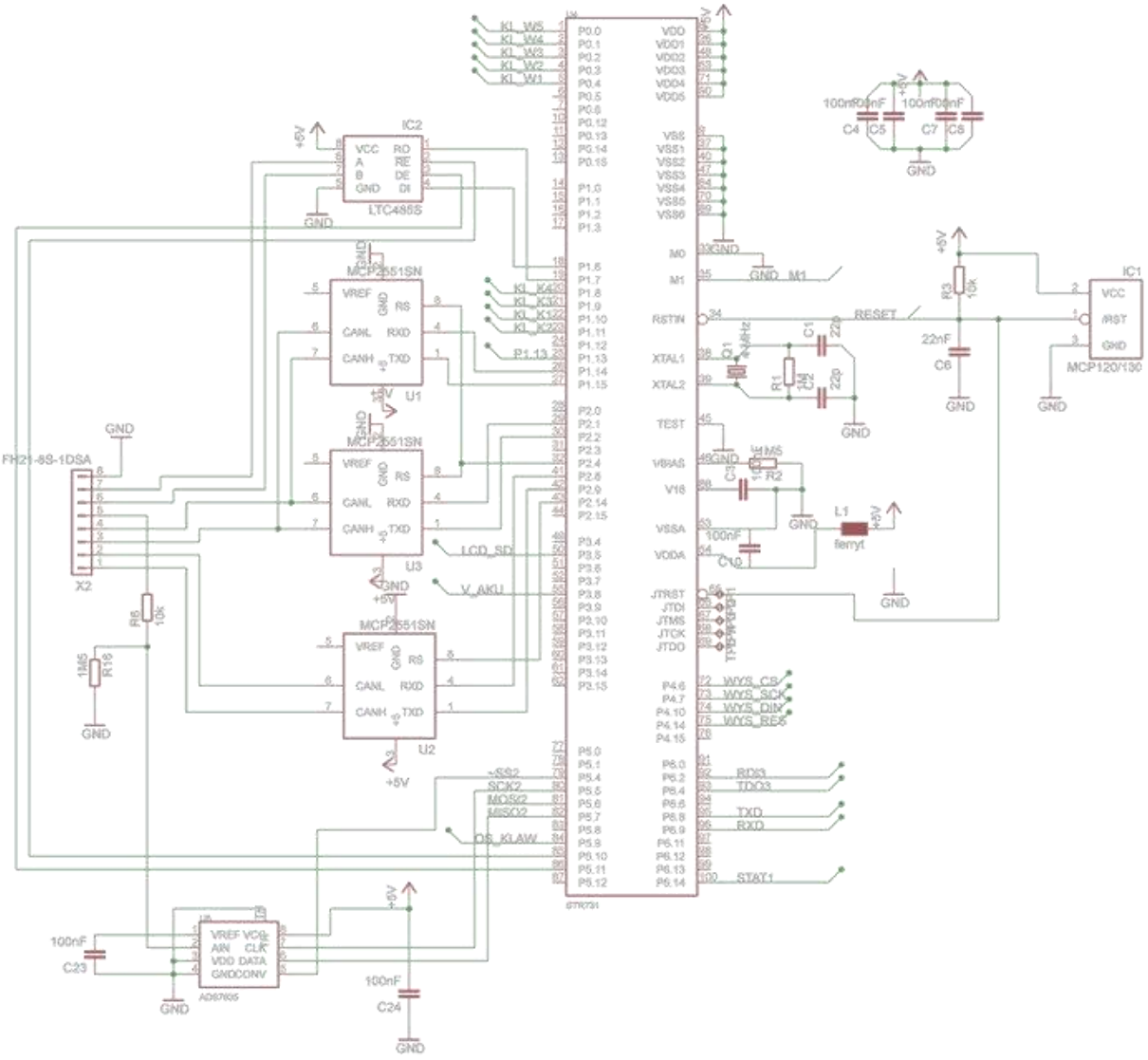
Sercem rodziny mikrokontrolerów STR7 firmy ST Microelectronics jest 32-bitowe CPU[7] ARM7TDMI zaprojektowane przez firmę ARM[8]. Jeszcze do niedawna użycie 32-bitowych mikrokontrolerów wymagało od przeciętnego konstruktora wiele wysiłku. Przyczyniały się do tego skomplikowane rozwiązania układowe, które wymagały projektowania skomplikowanych płytek drukowanych oraz dodawanie wielu elementów zewnętrznych jak np. dodatkowy stabilizator 1,8V do zasilania rdzenia. Niebanalnym argumentem była także cena, która była na tyle wysoka, że wiele problemów rozwiązywano kilkoma mikrokontrolerami 8-bitowymi. Sytuacja zmieniła się diametralnie po wprowadzeniu na rynek mikrokontrolerów z rdzeniem ARM. Dzięki wielu wbudowanym komponentom w strukturę układu (szczególnie pamięci flash oraz ram) implementacja tego typu układów stała się bardzo popularna i tania.

6.1.2. Implementacja w urządzeniu

Dzięki bogatemu wyposażeniu mikrokontroler STR7FV2 nie wymaga wiele elementów zewnętrznych do uruchomienia. Na rysunku 6.2 jest schemat najbliższego otoczenia mikrokontrolera. Ze względu na wiele operacji wykonywanych w ciągu sekundy zdecydowano się ustawić częstotliwość pracy rdzenia na 32 MHz. Do tego celu niezbędny jest rezonator kwarcowy 4 MHz oraz wykorzystanie wbudowanej pętli do pomnożenia tej częstotliwości. Do każdej pary wyprowadzeń zasilania dodano kondensator ceramiczny o wartości

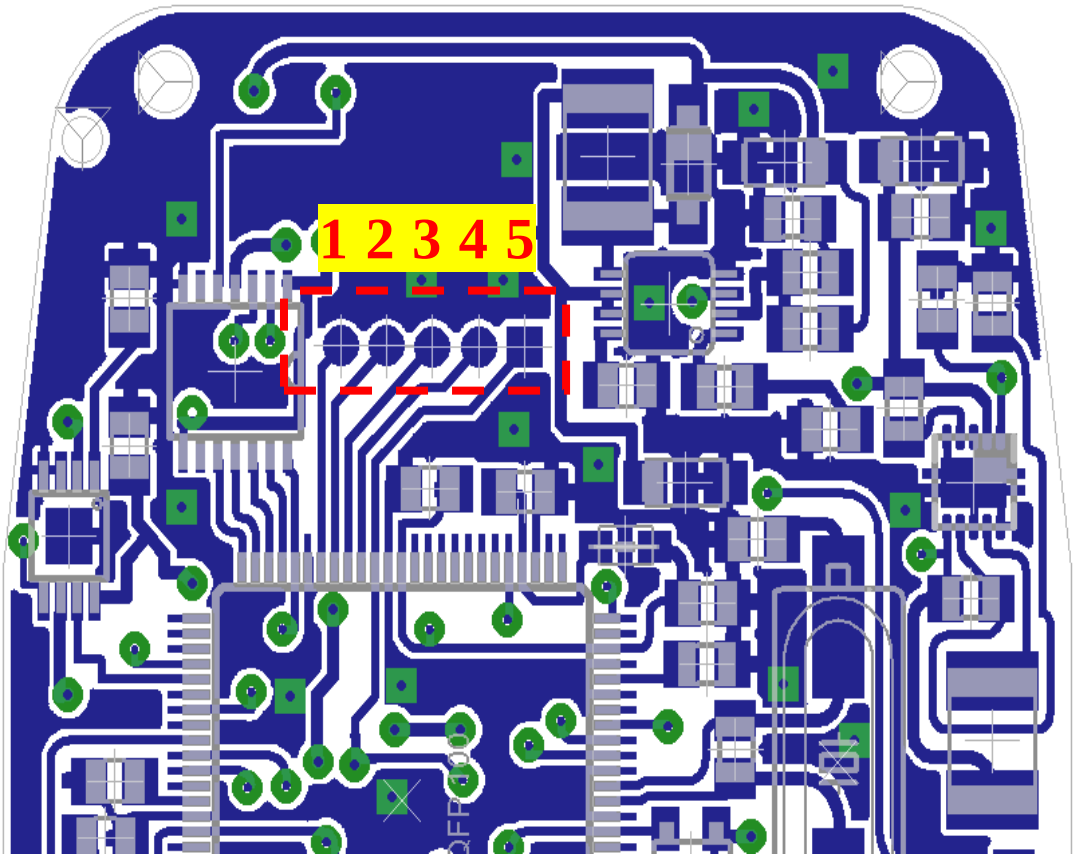
[7] CPU – ang. Central Processing Unit - centralna jednostka obliczeniowa. Układ scalony o wielkiej integracji zawierający w jednym układzie miliony tranzystorów.
[8] ARM – ang. Advanced RISC Machine, pierwotnie Acorn RISC Machine

100nF. Aby układ mógł poprawnie rozpocząć pracę wskazany jest dodatkowy układ resetu, pomimo tego jest także miejsce na wlutowanie odpowiednio dołączonych kondensatora i rezystora. W mikrokontrolerze znajduje się osobne zasilanie bloku przetwornika A/C, które powinno być dobrze odfiltrowane. Dokonano tego przy użyciu elementu ferrytowego. Po wyposażeniu mikrokontrolera w wymienione elementy dodatkowe staje się możliwe jego uruchomienie.



Rysunek 6.2. Schemat otoczenia mikrokontrolera

Urządzenie diagnostyczne wykorzystuje wiele wbudowanych w mikrokontroler modułów.



Rysunek 6.3. Wyprwadzenie sygnałów infterfejsu JTAG

Do zaprogramowania mikrokontrolera wykorzystano interfejs JTAG. Umożliwia on swobodne programowanie oraz debugowanie bez konieczności wylutowywania układu z urządzenia. Na rysunku 6.3 przedstawiono miejsce do podłączenia interfejsu w urządzeniu, które znajduje się po stronie BOTTOM płytki, poniżej znajduje się opis wyprowadzeń.

1. TDO (ang. Test Data Out) - wyjście danych,
2. TCK (ang. Test Clock) - wejście sygnału zegarowego,
3. TMS (ang. Test Mode Select) - wybór trybu pracy,
4. TDI (ang. Test Data In) - wejście danych,
5. TRST (ang. Test Reset) - zerowanie (opcjonalne).

6.2. Klawiatura membranowa z podświetlaniem

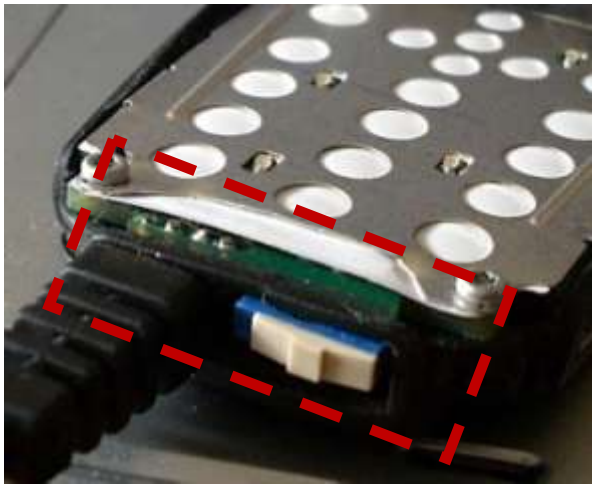
Rysunek 6.4. Widok układu klawisz na płytce drukowanej

Zdecydowano się wykorzystać przyciski, jakie są w obudowie telefonu Nokia 2600 (rysunek 4.9). Wymusiło to odpowiednie zaprojektowanie płytki drukowanej. Zasada działania tej klawiatury polega na zwarciu dwóch odsłoniętych pól na płytce. W miejsca gdzie przypadają przyciski musi być wydzielona płaska powierzchnia, dlatego umieszczanie elementów elektronicznych jest niemożliwe. Jedynym wyjątkiem są diody podświetlające, na które zostało specjalnie przygotowane miejsce. Aby klawiatura działała dobrze należało dokładnie zmierzyć położenie przycisków i w ten sam precyzyjny sposób odwzorować je na płytce drukowanej, widać to na rysunku 6.4.

Rysunek 6.5. Przyciski klawiatury

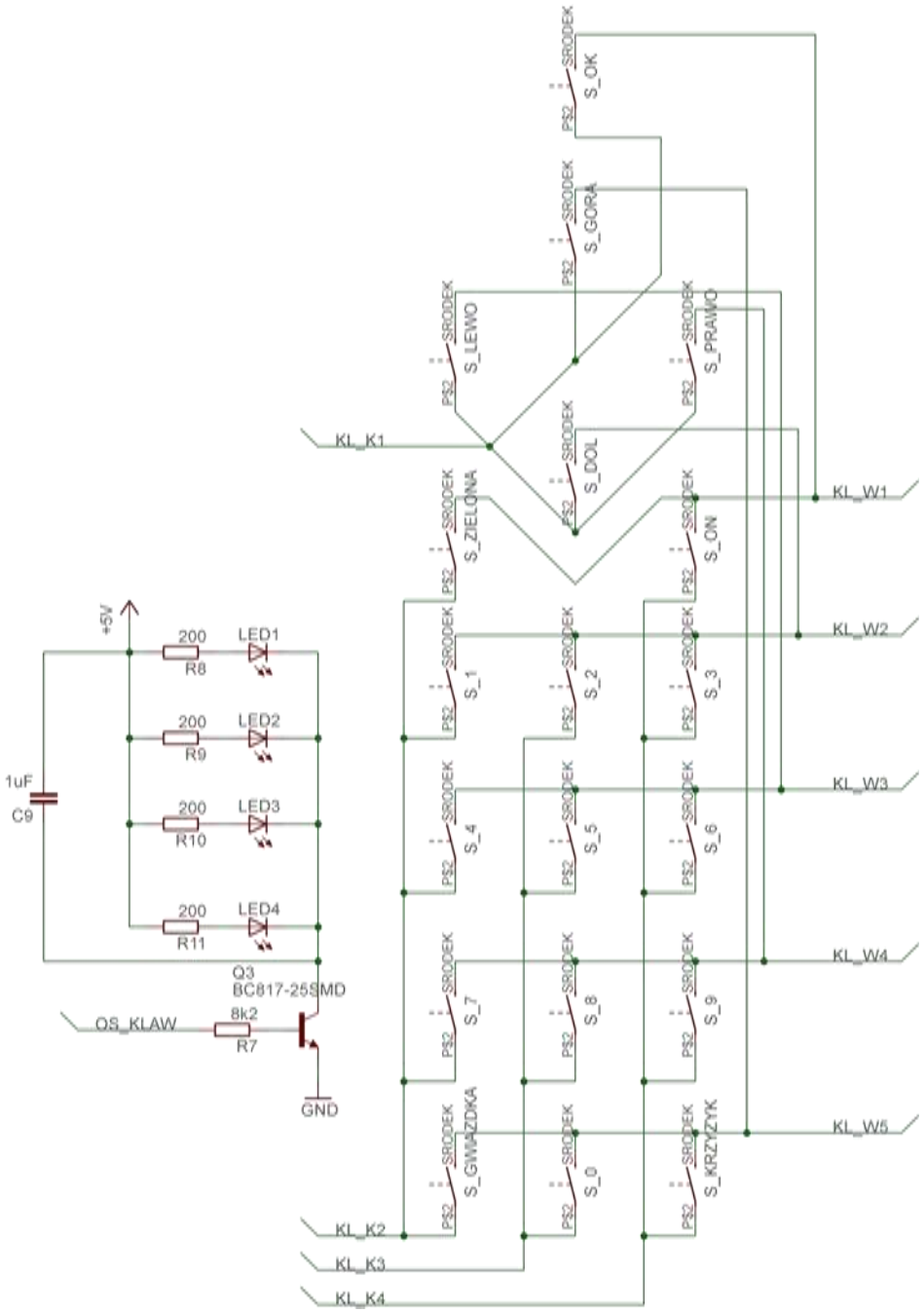
6.2.1. Zasada działania klawiatury

Przyciski zostały połączone w układ matrycy. Zasada działania takiego układu jest bardzo prosta, mianowicie kontroler podaje napięcie +5V na jedną z kolumn połączonych klawiszy, po czym sprawdza, w którym wierszu się ono pojawiło. Na rysunku 6.7 jest zamieszczony schemat klawiatury wraz z układem podświetlania. Ponieważ na płytce nie ma miejsca na dodatkowy kontroler klawiatury. Rolę tą pełni mikrokontroler STR7FV2 wykorzystując swoje porty I/O, bada stany na wyjściach klawiatury oraz obsługuje program do eliminacji drgań styków. Jest to bardzo niepożądane zjawisko, które może trwać do 2ms.



Rysunek 6.6. Modyfikacja blaszki dociskającej membranki przycisków i trzymającej wyświetlacz

Pomiędzy płytką drukowaną a klawiszami znajdują się metalowe membrany przyklejone do białej folii, która trzyma je na właściwym miejscu. Następnie całość jest przyklejona do płytki słabym klejem oraz dociśnięta metalową blaszką. Rysunek 6.6 pokazuje wykonaną modyfikację blaszki pochodzącej z telefonu Nokia 2600. Ponieważ u dołu płytki znajdują się miejsca lutownicze do przewodów oraz włącznika zasilania, należało odpowiednio ściąć w tym miejscu blaszkę, aby nie odstawała i nie powodowała zwarcia



Rysunek 6.7. Schemat bloku klawiatury i podświetlania

6.2.2. Podświetlanie klawiszy

Podświetlanie wykorzystuje zielone superjasne diody LED[9] w obudowach SMD 0603 w celu zmniejszenia pobieranego prądu. Ponieważ porty I/O mikrokontrolera mają ograniczoną wydajność prądową konieczne było użycie dodatkowego tranzystora do sterowania diodami. Baza tranzystora Q3 została podłączona przez rezystor dobrany tak, aby wysterować diody przy ich maksymalnym prądzie, została podłączona do wyjścia PWM mikrokontrolera. Pozwala to sterować nie tylko binarnie (włącz/wyłącz) diodami, ale również ich jasnością. Dzięki temu można przy odpowiedniej jasności świecenia diod znacznie zredukować pobór prądu.

Rysunek 6.8. Efekt działającego podświetlania.

Przy uruchamianiu podświetlania ważną rzeczą był dobór odpowiednich rezystorów R6, R9-11 tak, aby diody przy zasilaniu napięciem +5V świeciły najjaśniej. W wyniku przeprowadzonych pomiarów, ustalono prąd oraz napięcie diod, po czym obliczono wartość rezystancji przekształcając odpowiednio wzór prawa Ohma:

$$R = \frac{5V - U_d}{I_d}$$

[9] LED (ang. light-emitting diode) – dioda zaliczana do półprzewodnikowych przyrządów optoelektronicznych, emitujących promieniowanie w zakresie światła widzialnego

Gdzie:

R – obliczona wartość rezystancji
U_d – napięcie świecenia diody
I_d – prąd świecenia diody

Otrzymana wartość wynosi 200Ω. Diody wtedy świecą dość jasno, aby podświetlić klawiaturę w świetle dziennym. Na rysunku 6.8 jest widoczny efekt działania uruchomionego podświetlania.

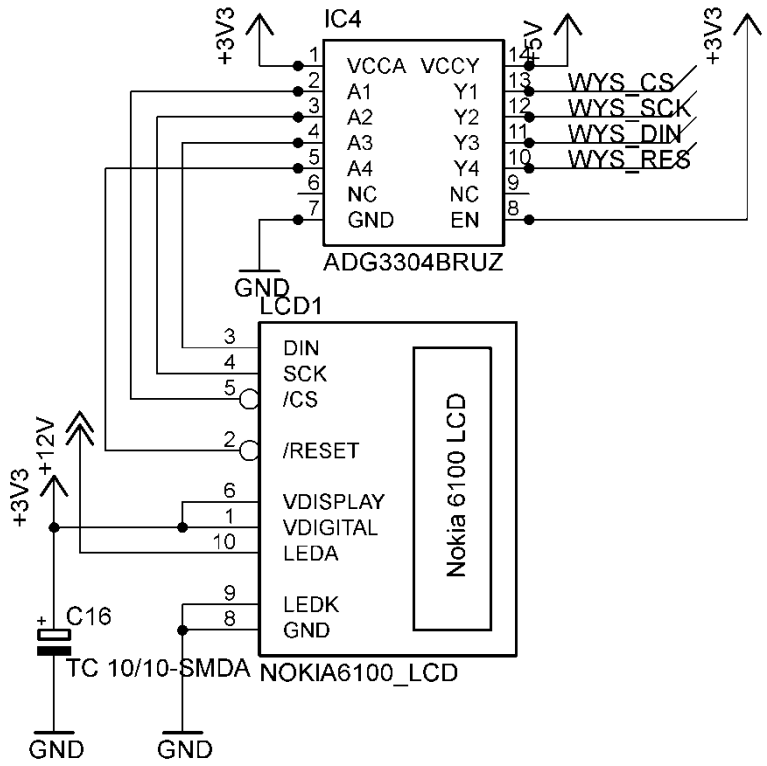
6.3. Wyświetlacz

6.3.1. Charakterystyka wyświetlacza

Rysunek 6.9. Wygląd użytego wyświetlacza LCD

Wyświetlacz jest bardzo ważnym elementem w urządzeniu, ponieważ pokazuje otrzymane wyniki analizy oraz inne parametry urządzenia. Użyty wyświetlacz pochodzi z Nokii 6100, jego wygląd przedstawia rysunek 6.9. Charakteryzuje się dobrym kontrastem oraz rozdzielczością 132 x 132 piksele. Dużą zaletą jest zintegrowane podświetlanie oraz zwarta obudowa. Jedynym mankamentem okazało się niewielkie 10-cio pinowe złącze, którego przylutowanie wymagało dużej precyzji, a gdy wyświetlacz nie jest na stałe przymocowany złącze czasami nie przewodzi niektórych sygnałów.

6.3.2. Schemat podłączenia wyświetlacza



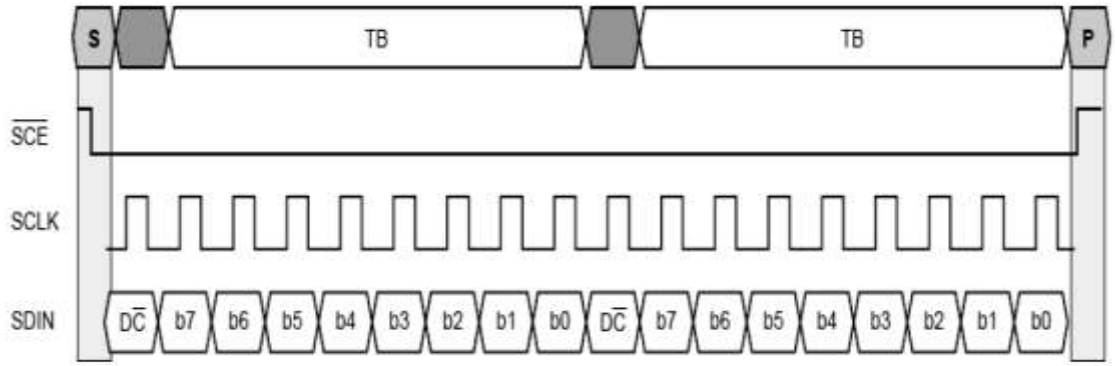
Rysunek 6.10. Sprzętowy interfejs do wyświetlacza LCD

Na schemacie z rysunku 6.10 widać dodatkowy układ IC4 pośredniczący pomiędzy sygnałami z mikrokontrolera, a samym wyświetlaczem. Jest to scalony niskonapięciowy 1,15V do 5,5V, 4-kanałowy, dwukierunkowy translator poziomu logicznego firmy Analog Devices. Dzięki użyciu tego układu zredukowano ilość potrzebnego miejsca na płytce do minimum, ponieważ alternatywnym rozwiązaniem było zastosowanie do każdej linii odpowiednio podłączonego rezystora z diodą zenera.

6.3.3. Komunikacja z wyświetlaczem

Nokia 6100 wykorzystuje dwuprzewodowy szeregowy interfejs SPI (zegar i dane). Mikrokontroler STR731 posiada moduł BSPI, umożliwia on komunikację z urządzeniami typu SLAVE, np. wyświetlaczem jednak działa on tylko w trybie 8 lub 16-bitowym. Dlatego sprzętowo obsługiwana komunikacja pomiędzy procesorem, a wyświetlaczem nie jest możliwa do zrealizowania, gdyż do obsługi wyświetlacza potrzebna jest transmisja 9-bitowa, gdzie

pierwszy bit określa, czy pozostałe 8 bitów jest komendą dla wyświetlacza czy danymi – rysunek 6.11. Transmisja danych odbywa się **tylko w jednym kierunku** – do wyświetlacza.

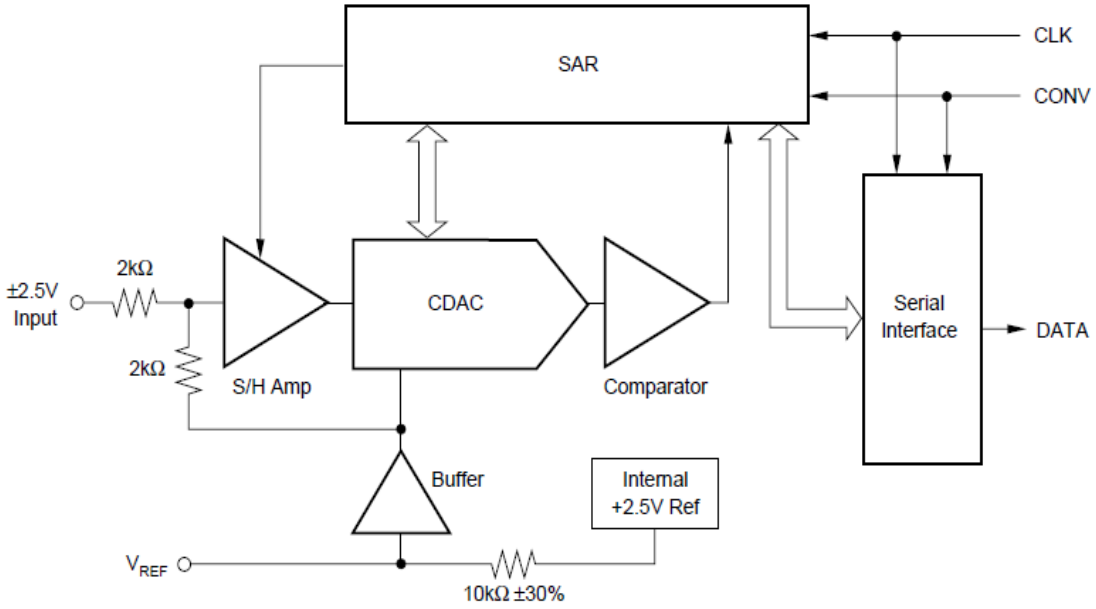


Rysunek 6.11. Interfejs szeregowy SPI przesyła bajty instrukcji i danych. [5]

Nota katalogowa wyświetlacza mówi, że największa prędkość taktowania magistrali może wynosić 6MHz. Ze względu na obsługę programową uzyskaną przez odpowiednie sterowanie liniami wejścia/wyjścia mikrokontrolera uzyskano rzeczywistą prędkość poniżej 1MHz, a obsługa wyświetlacza zajmuje sporo czasu procesora, który działa nieefektywnie. W sytuacji kiedy procesor trafia na instrukcję sterowania portami musi wyczyścić cały potok instrukcji, a potem po wykonaniu operacji na porcie, powrotem go zapełnić.

6.4. Dodatkowy przetwornik A/C

Użyty w urządzeniu przetwornik A/C służy do prostych pomiarów napięcia, dzięki czemu ułatwia pracę przy odnajdywaniu magistrali CAN oraz umożliwia pomiary, jakie często mają miejsce przy diagnozowaniu systemów. Do tego celu zdecydowano się na użycie bardzo szybkiego, 12-bitowego przetwornika o niskim zużyciu prądu oraz możliwie najmniejszej obudowie – wybrano przetwornik typu SAR[10] firmy BURR-BROWN, ADS 7835.



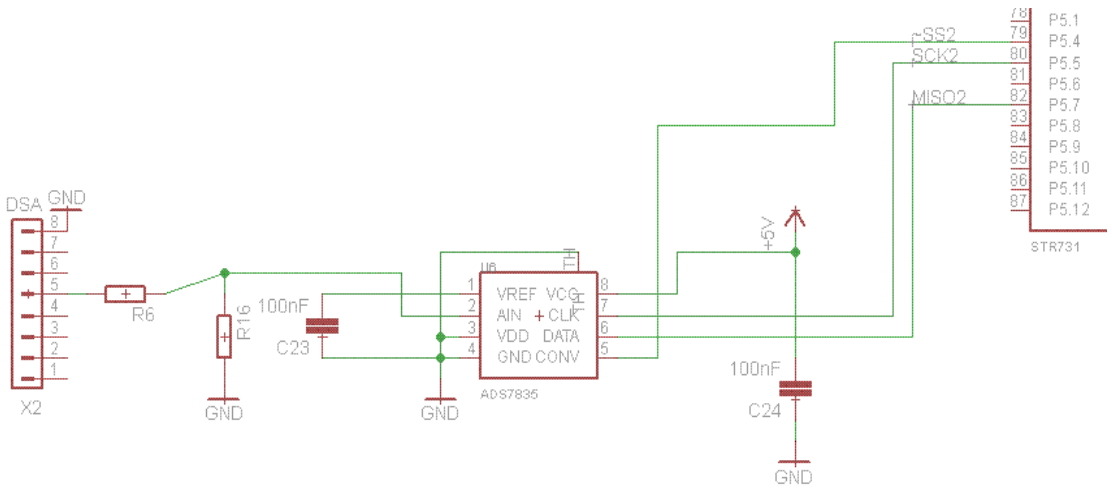
Rysunek 6.12. Schemat blokowy przetwornika ADS7835 [11]

Przetwornik z sukcesywną aproksymacją (próbkowaniem bitowym) działa na zasadzie porównywania wartości napięcia wejściowego z napięciem odniesienia wytworzonym za pomocą przetwornika cyfrowo-analogowego w iteracyjnym procesie obsługiwanym przez układ sterujący. Algorytm działania układu sterującego polega na ustawianiu (wartość "1") kolejnych bitów słowa danych dla przetwornika C/A poczynając od najważniejszego bitu słowa (MSB) i w przypadku, kiedy napięcie wejściowe będzie mniejsze od napięcia odniesienia z przetwornika C/A to dany bit słowa danych jest kasowany (wartość "0") w przeciwnym wypadku jest pozostawiany (wartość "1") i

[10] Przetwornik o sukcesywnej aproksymacji

realizowana jest kolejna iteracja algorytmu aż do osiągnięcia ostatniego bitu słowa danych (LSB). Tak ustawione słowo danych jest reprezentacją cyfrową napięcia wejściowego. Ze względu na iteracyjny charakter pracy przetwornika jego częstotliwość próbkowania jest znacząco mniejsza od uzyskiwanej w przetwornikach o przetwarzaniu bezpośrednim i w znacznym stopniu zależy od wielkości słowa danych – rozdzielczości przetwornika, szybkości pracy przetwornika C/A i w końcu komparatora i układu sterującego. Ta metoda przetwarzania A/C wykorzystywana jest we współczesnej telekomunikacji. [10]

Całą zasadę działania ukazuje schemat blokowy na rysunku 6.12. Dzięki wbudowanemu dzielnikowi napięcia wejściowego udało się dodatkowo zminimalizować ilość występujących elementów na płytce.



Rysunek 6.13. Schemat podłączenia przetwornika A/C

Schemat z rysunku 6.13 jest niemalże identyczny ze schematem aplikacyjnym z noty katalogowej. Jedyne co dodano to dodatkowy dzielnik napięcia, aby układ mógł działać z napięciami do 40V. Układ dołączony jest trzema połączeniami ze sprzętowym modułem BSPI[11] mikrokontrolera.

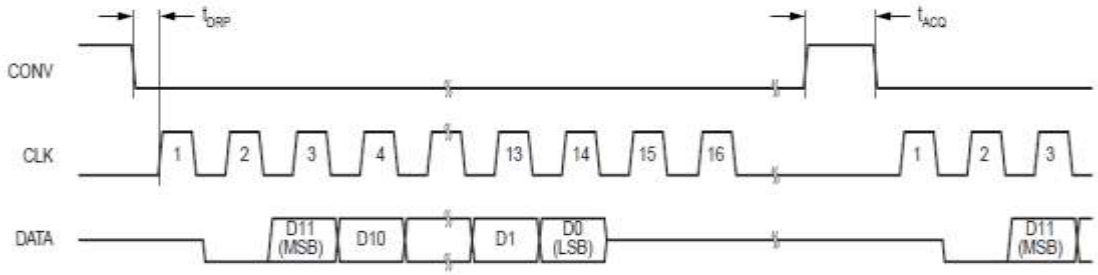
Układ ADS7835, aby uruchomić proces konwersji potrzebuje zewnętrznego sygnału zegarowego. Taki sygnał powinien mieścić się w granicach, pomiędzy 200kHz (próbkowanie 12, kHz), a 8MHz (próbkowanie 500kHz). Maksymalna częstotliwość taktowania układu peryferyjnego BSPI wynosi 6MHz, dlatego próbkowanie sygnału może odbywać się z maksymalną częstotliwością 250kHz.

[11] Buffored SPI

$f_{próbki}$	moc przy 8 MHz
500kHz	17,5mW
250kHz	13,5mW
100kHz	10,5mW

Tabela 6.2. Pobór mocy w zależności od częstotliwości sygnału zegarowego

Pomimo, że próbkowanie nie jest możliwie największe przez układ i tak jest na tyle spore, aby sprostać wymaganiom przy wielu pomiarach napięcia w motoryzacji. Tabela 6.2 pokazuje pobór mocy w zależności od częstotliwości sygnału zegarowego. Można z niej odczytać, że przy mniejszej częstotliwości próbkowania moc pobierana przez układ jest mniejsza, co jest dodatkową zaletą.



Rysunek 6.14. Typowy przebieg czasowy dla interfejsu SPI [11]

Istnieje kilka trybów pracy związanych ściśle z podłączeniem układu ADS7835. W projekcie zostało wybrane połączenie interfejsem SPI. Na rysunku 6.14 jest przedstawiony typowy przebieg czasowy dla tego interfejsu. Zgodnie ze schematem z rysunku 6.13 wyprowadzenie CONV układu zostało połączone do wyjścia SS mikrokontrolera i odpowiednio CLK do SCK, a DATA do wejścia MISO.

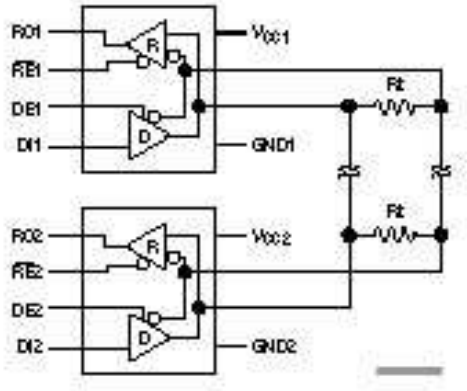
6.5. RS-485

Zaimplementowano dodatkowo obsługę protokołu RS-485. Wykorzystuje ona układ peryferyjny mikrokontrolera UART, dzięki czemu procesor jest bardziej odciążony. Schemat ideowy został przedstawiony na rysunku 6.16. Do

ustawienia odpowiednich poziomów i zabezpieczenia portów mikrokontrolera służy układ LTC485S firmy Linear Technology. Główne cechy układu zostały przedstawione poniżej.

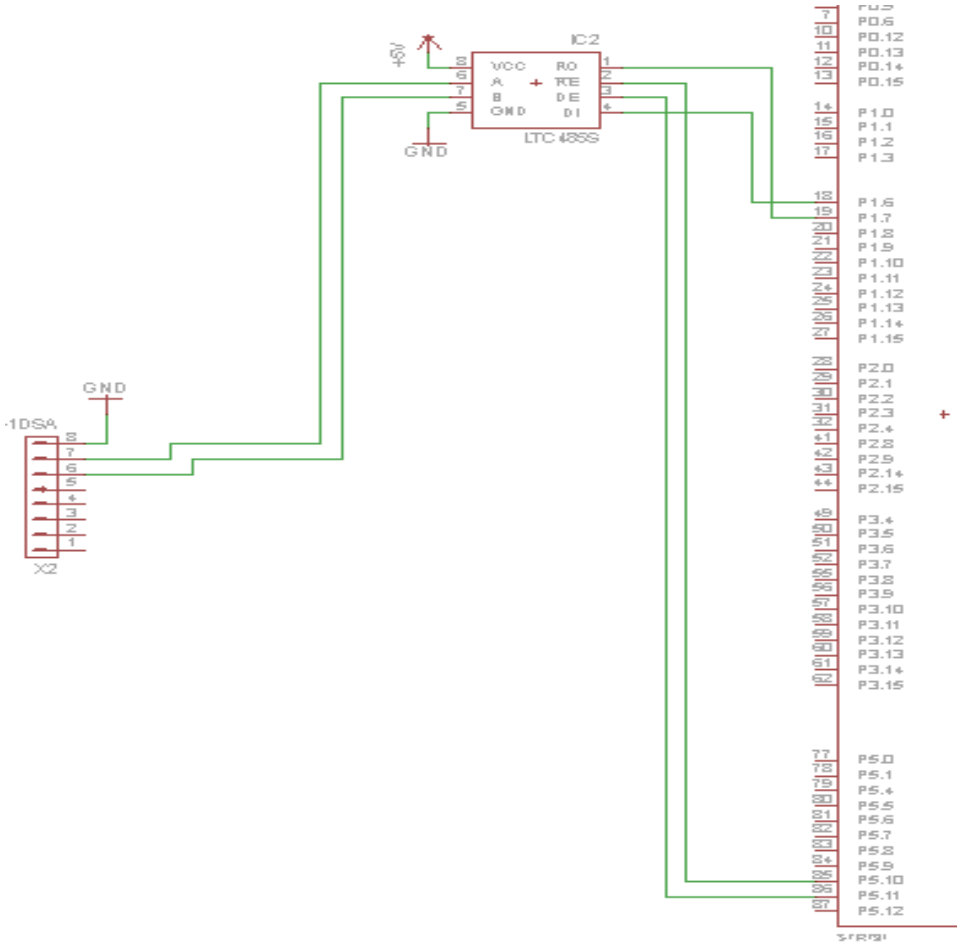
- Niski pobór prądu: Typowo 300µA
- Zaprojektowany dla aplikacji z interfejsem RS485
- Pojedyncze zasilanie 5V
- Ochrona termiczna
- Impedancja wyjściowa sterownika pozwala na dołączenie do 32 terminali do szyny
- 70mV typowa histeza wejścia
- 30ns typowy czas propagacji sterownika
- Zabezpieczenie wejść ±60V

Na schemacie z rysunku 6.16 widać dwa połączenia danych, które są bezpośrednio dołączone do portów UARTu mikrokontrolera. Wejścia sterujące układu LTC485S sterowane są ręcznie portami wejścia/wyjścia.



Rysunek 6.15. Typowa aplikacja układu LTC485S

Ze względu na założenie, że urządzenie będzie się dołączać do już istniejącego systemu, nie jest ono wyposażone w terminatory końcowe (oznaczone jako Ft) – rysunek 6.15.

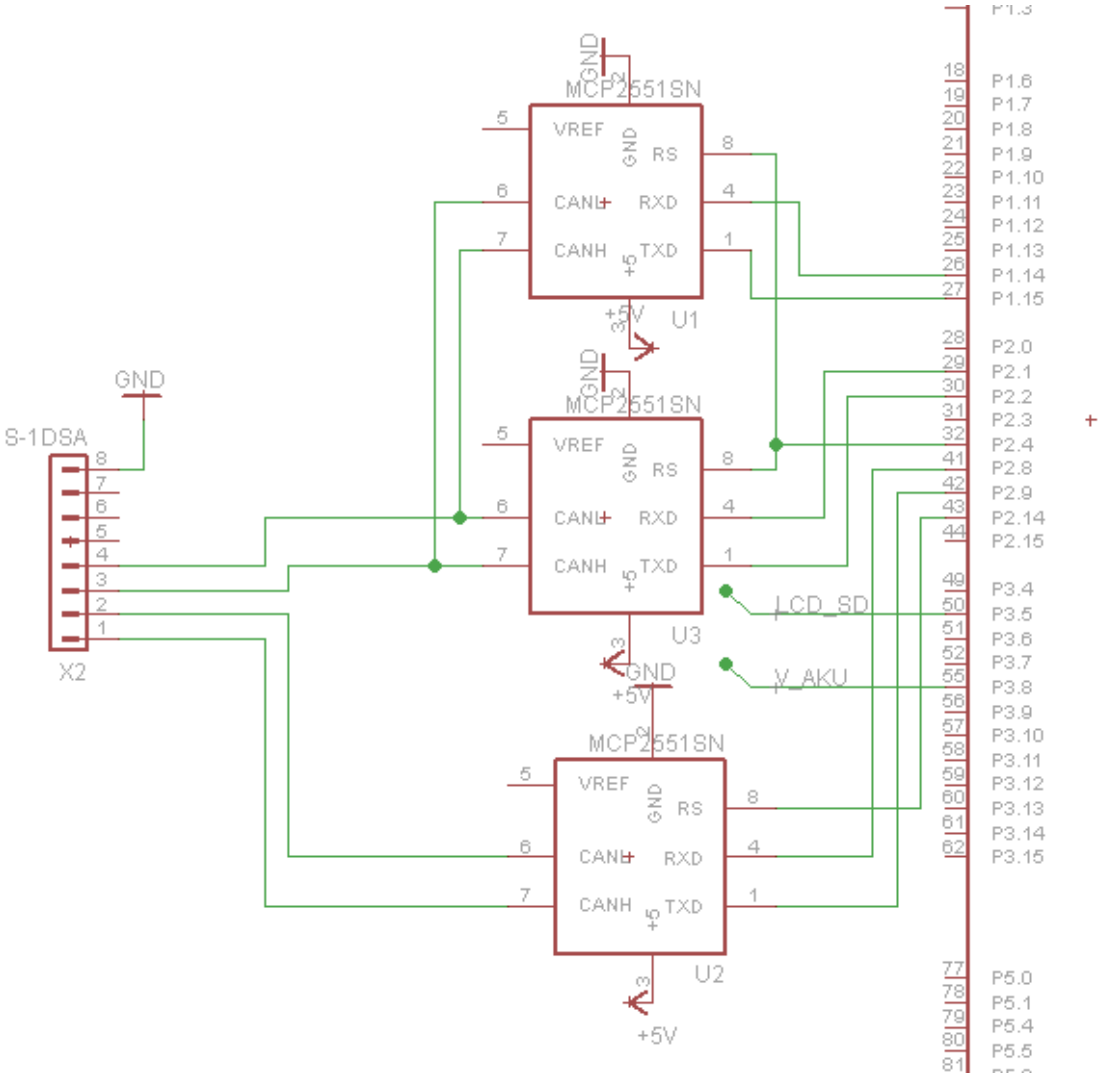


Rysunek 6.16. Schemat ideowy bloku RS-485

6.6. Moduł CAN

Moduł CAN wykorzystuje wewnętrzny peryferyjny moduł CAN mikrokontrolera. Korzystając z tego udogodnienia, budowa wymagała dodania jedynie trzech konwertorów poziomów, które dodatkowo zabezpieczają mikrokontroler przed przepięciami z magistrali. Na schemacie ideowym (rysunek 6.17) widać je oznaczonych symbolami U1, U2 i U3. Linia CANH układu U1 została dołączona do linii CANL układu U3 oraz podobnie linie CANL układu U1 i CANH układu U3. Są to linie kanału odbiorczego, połączenie takie dało w rezultacie przy odpowiednim odczytywaniu danych przez mikrokontroler możliwość podłączenia kanału odbiorczynie nie zwracając uwagi na odpowiednie dołączenie linii. Wszystkie trzy układy posiadają wejście

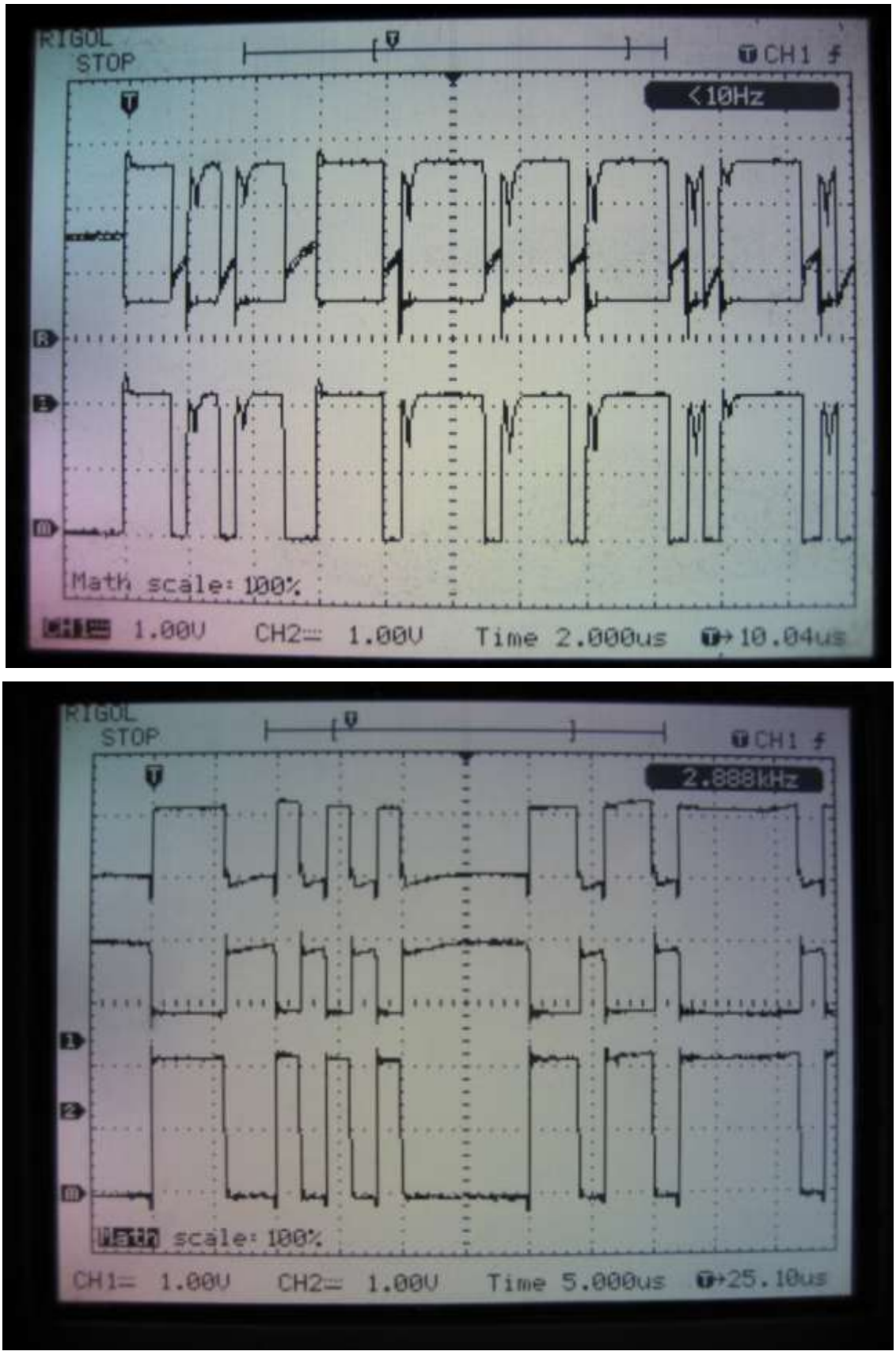
RS, którym steruje mikrokontroler. Jest ono przeznaczone o graniczenia zużycia prądu, podczas gdy urządzenie nie korzysta z magistrali CAN.



Rysunek 6.17. Schemat ideowy bloku CAN

Rysunek 6.18. Widok zamontowanych na płytce trzech sterwoniczków CAN

Rysunek 6.19. Ekran wyboru parametrów kanału odbiorczego CAN

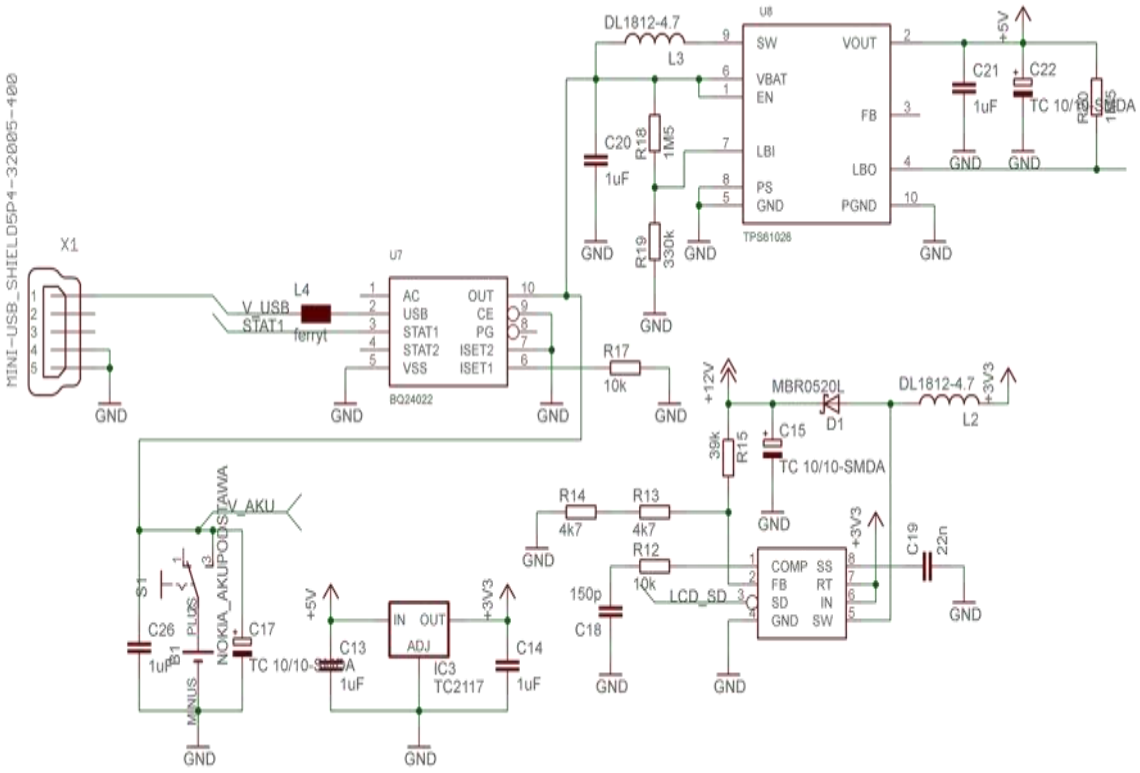


Rysunek 6.20. Testy modułu CAN na oscyloskopie

Po zmontowaniu i uruchomieniu urządzenia, jedną z pierwszych rzeczy, która została sprawdzona była poprawność działania kanałów CAN. W tym celu połączono kanał nadawczy z kanałem odbiorczym urządzenia, a oba przewody zostały dołączone do kanałów w oscyloskopie – ilustruje to rysunek 6.20. Pierwszy wykres czasowy na rysunku przedstawia nałożone na siebie dwie linie magistrali: CANH oraz CANL. Przy prędkości 500kb ich wygląd odbiega od sygnałów prostokątnych, jednak po ich zsumowaniu, uzyskano bardzo przejrzysty przebieg sygnału, co widać na dolnym wykresie.

6.7. Zasilanie akumulatorowe

Szczególną uwagę przy projektowaniu urządzeniu poświęcono zasilaniu poszczególnych komponentów. Blok ten wykorzystuje 2 przetwornice oraz jeden stabilizator. Uwidacznia to szczegółowo schemat na rysunku 6.21.

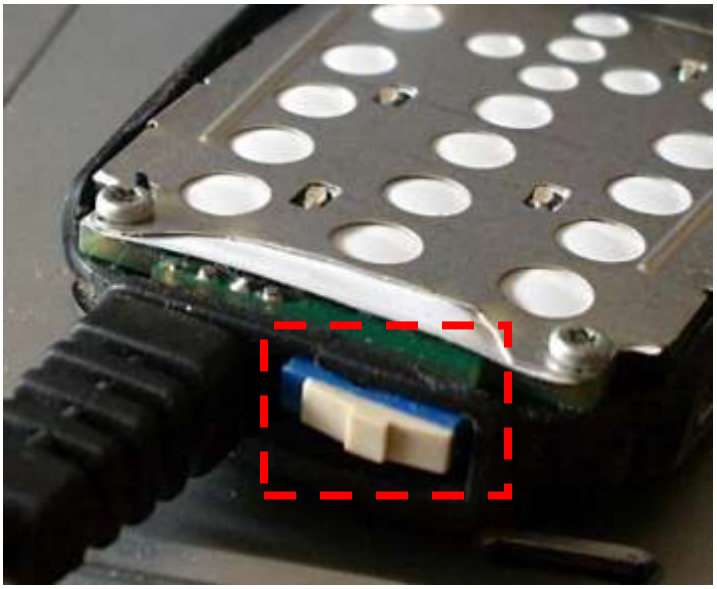


Rysunek 6.21. Schemat ideowy bloku zasilania

Rysunek 6.22. Wygląd zastosowanego akumulatora

Wykorzystano oryginalny akumulator litowo-jonowy firmy Nokia. Jego wygląd przedstawia rysunek 6.22.

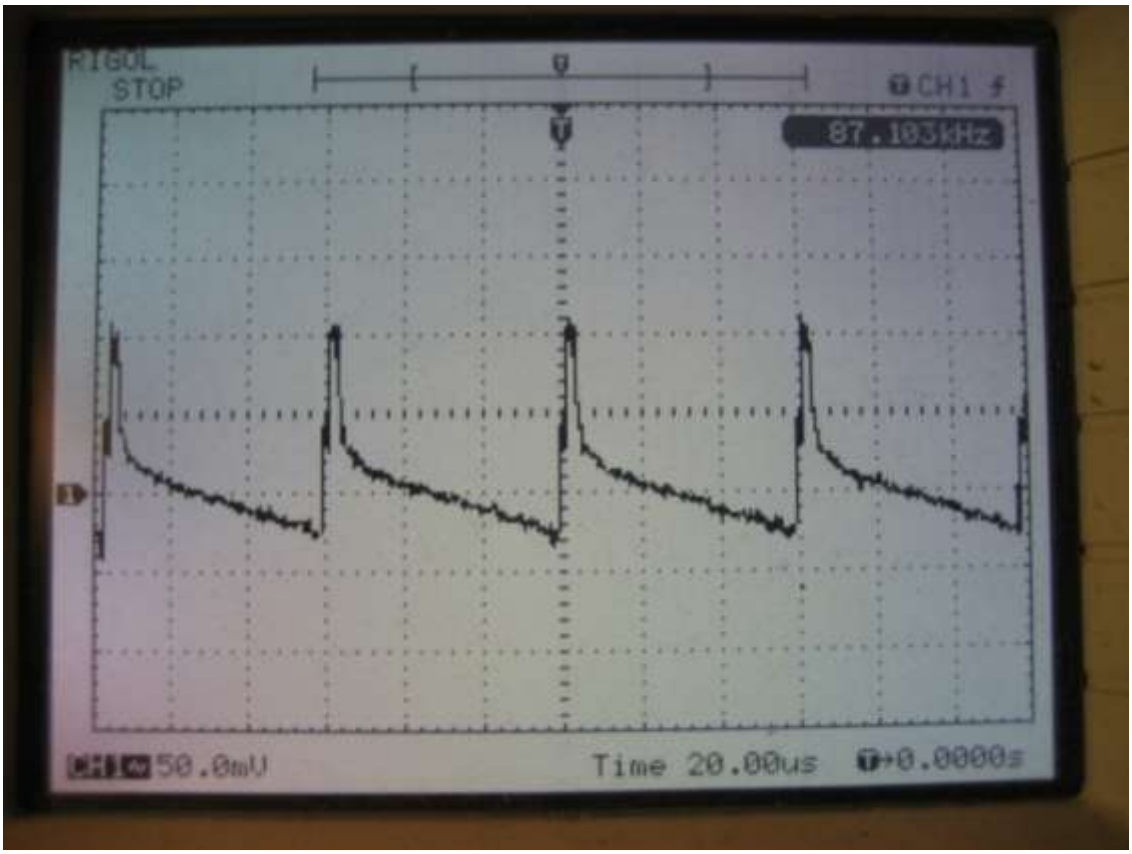
Umiejscowienie włącznika urządzenia znajduje się u jego spodu(rysunek 6.23). Zdecydowano się na zwykły przełącznik suwakowy, którego wyłączenie w sposób pewny odłączy akumulator od urządzenia, co uniemożliwi jego rozładowanie.



Rysunek 6.23. Włącznik zasilania

Rysunek 6.24. Przetwornica napięcia 5V

Zastosowanie przetwornicy umożliwiło obsługę akumulatora oraz dużą wydajność, co znacznie wydłuża czas działania. Poprzez dużą miniaturyzację udało się uzyskać rozmiar całej przetwornicy zajmujący minimum miejsca na płytce, co widać na rysunku 6.24. Wadą zastosowania przetwornicy są tętnienia zasilania, lecz przez zastosowanie odpowiedniego filtrowania są na akceptowalnym poziomie.



Rysunek 6.25. Tętnienia napięcia 5V

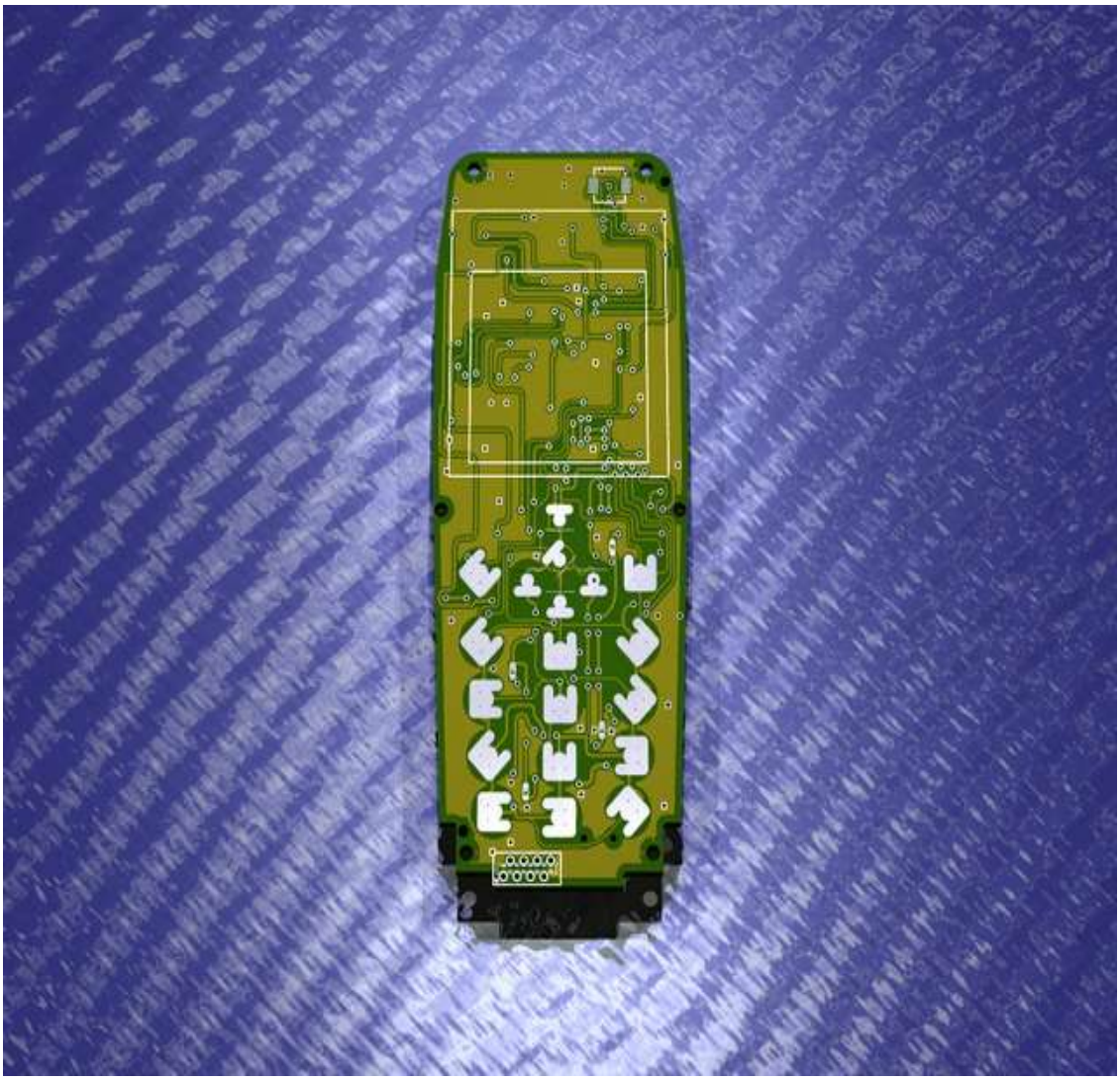
6.8. Płytka drukowana

Płytka została zaprojektowana w programie Eagle firmy Cadsoft. Projektowanie komputerowe było niezbędne ze względu na dużą złożoność płytki. Na początku projektowania konieczne okazało się dodanie wielu elementów do biblioteki programu. Kolejnym krokiem, który okazał się bardzo czasochłonny było dokładne zmierzenie oraz odwzorowanie krawędzi płytki. Było to zadanie o tyle trudne gdyż płytka ma kształt ściśle dopasowany do obudowy.

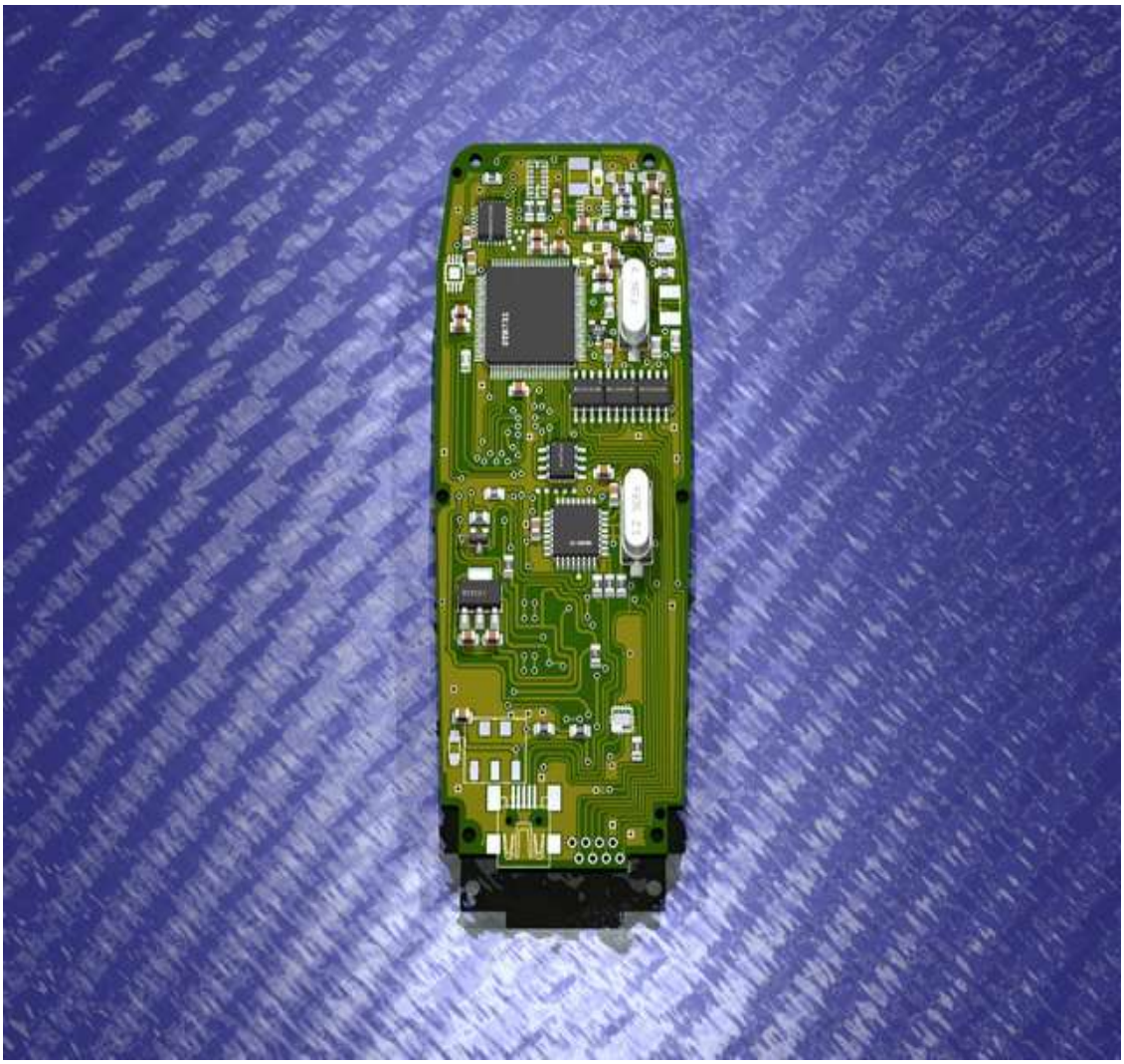
Przed oddaniem gotowego projektu płytki do produkcji, użyto narzędzia programu Eagle do generowanie trójwymiarowego obrazu płytki, efekt działania programu został przedstawiony na rysunkach 6.26 i 6.27.

Na rysunku 6.28 widać rozmieszczenie ścieżek na płytce, kształt krawędzi oraz niezbędne otwory. Z uwagi na fakt, że w warstwie TOP[12] płytki umieszczono wyświetlacz oraz klawiaturę, ograniczyło to możliwość ustawiania elementów elektroniczny do tego stopnia, że występuje jedynie gniazdo na wyświetlacz oraz u dołu płytki złącze sygnałowe i główny włącznik zasilania.

[12] Warstwa TOP – górna warstwa płytki gdzie widok ścieżek nie jest odbiciem lustrzanym.



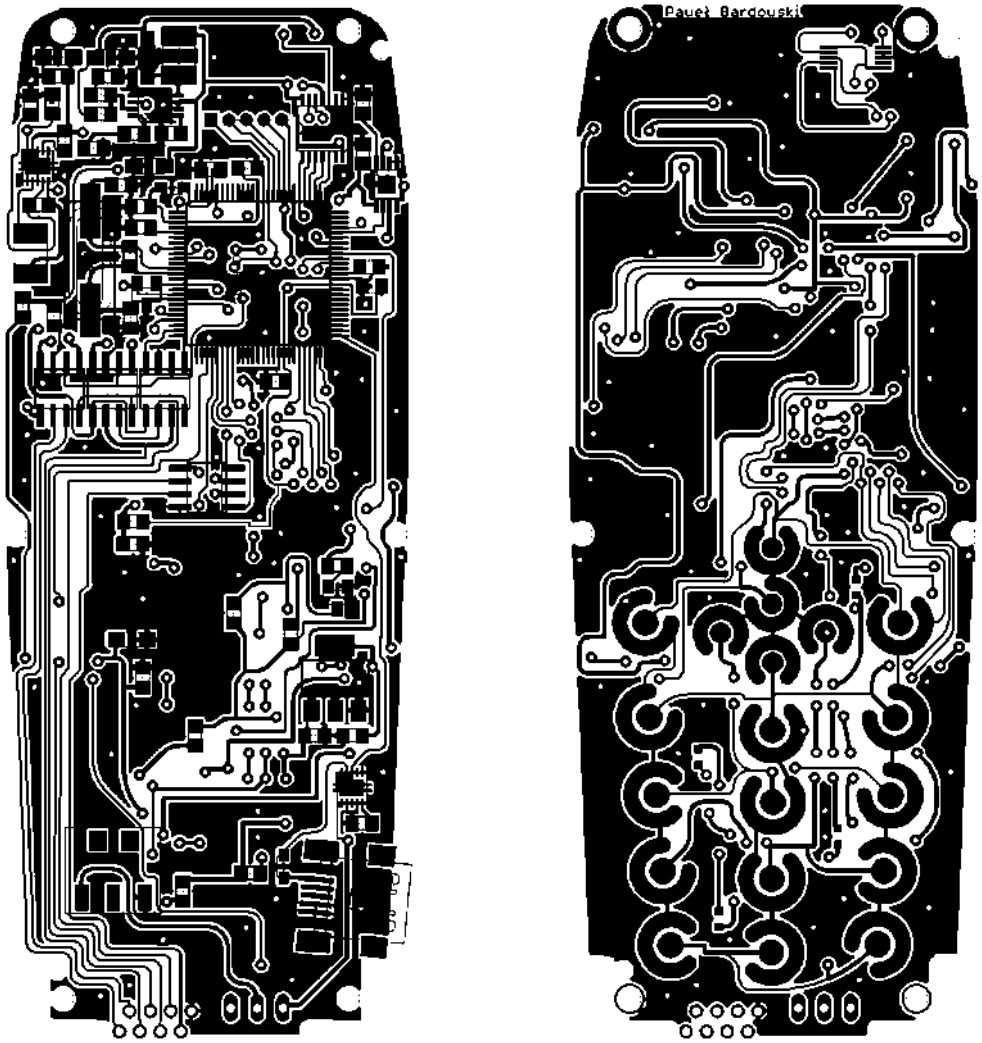
Rysunek 6.26. Komputerowy projekt płytki PCB, widok z góry



Rysunek 6.27. Komputerowy projekt płytki PCB, widok z dołu

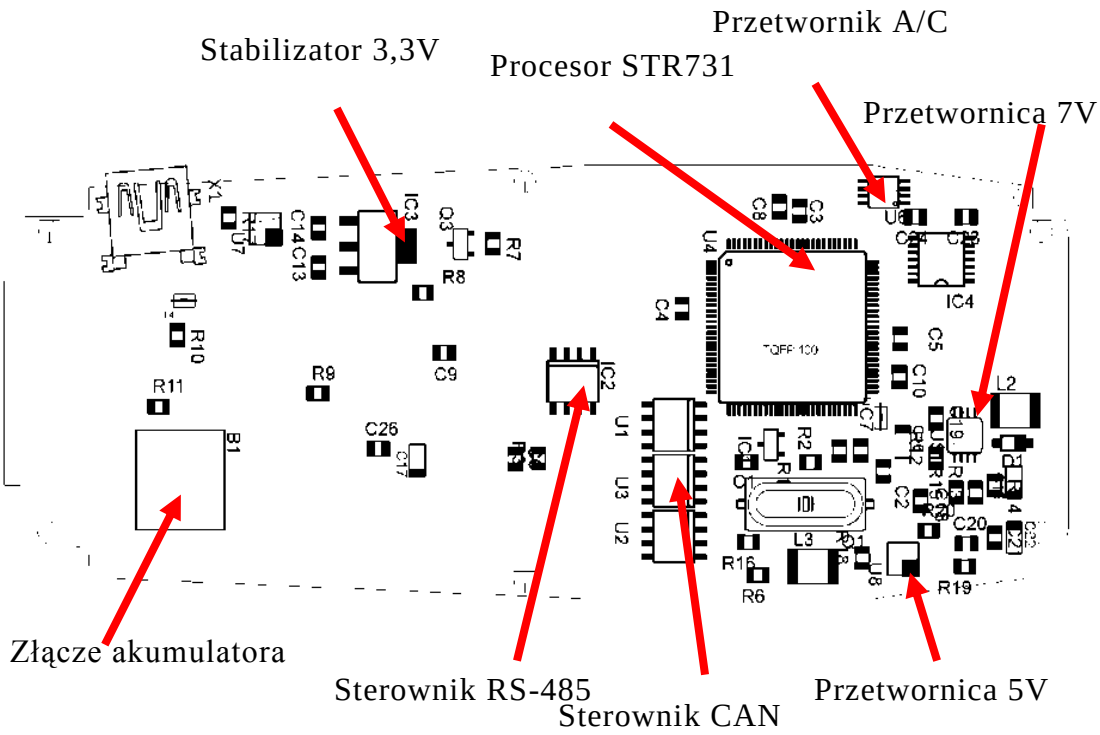
Reszta elementów została ciasno upakowana po stronie BOTTOM[13] w większości pod wyświetlaczem. Uwarunkowane jest to miejscem gdzie można stosować przeplotki. Na obu warstwach wolne miejsca wypełniono poligonami podłączonymi do masy, minimalizując w ten sposób ewentualne zakłócenia.

[13] Warstwa BOTTOM – warstwa spodnia płytki, na której widok przedstawia ścieżki w odbiciu lustrzanym.



Rysunek 6.28. Układ ścieżek płytki, wartstwa BOTTOM (z lewej strony) oraz TOP (z prawej strony)

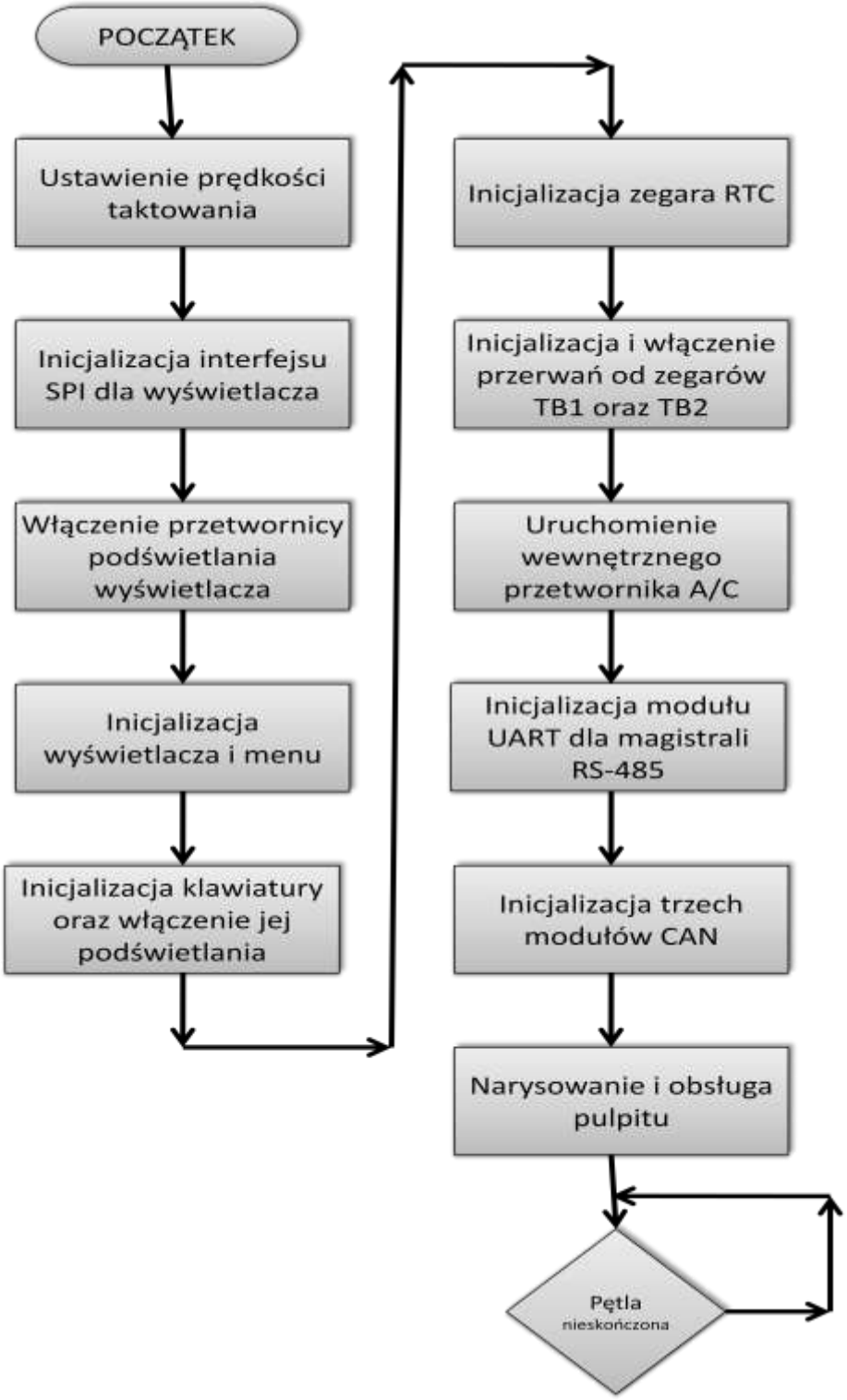
Rysunek 6.29 przedstawia rozmieszenie elementów. Takie ułożenie wynika z dopasowania mechanicznego płytki do obudowy. Z lewej strony rysunku widać złącze do akumulatora B1, złącze USB do ładowania X1.



Rysunek 6.29. Rozmieszczenie elementów w warstwie BOTTOM

7. OPROGRAMOWANIE MIKROKONTROLERA

7.1. Uruchomienie oraz inicjalizacja mikrokontrolera

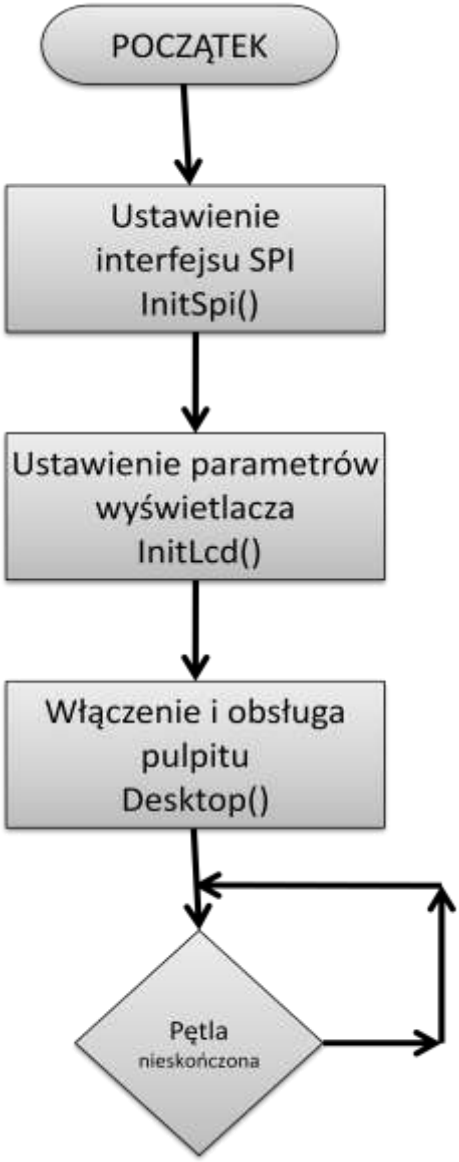


Rysunek 7.1. Inicjalizacja urządzenie po uruchomieniu

Po uruchomieniu urządzenia mikrokontroler włączą i konfiguruje wszystkie moduły, proces ten przedstawia algorytm na rysunku 7.1.

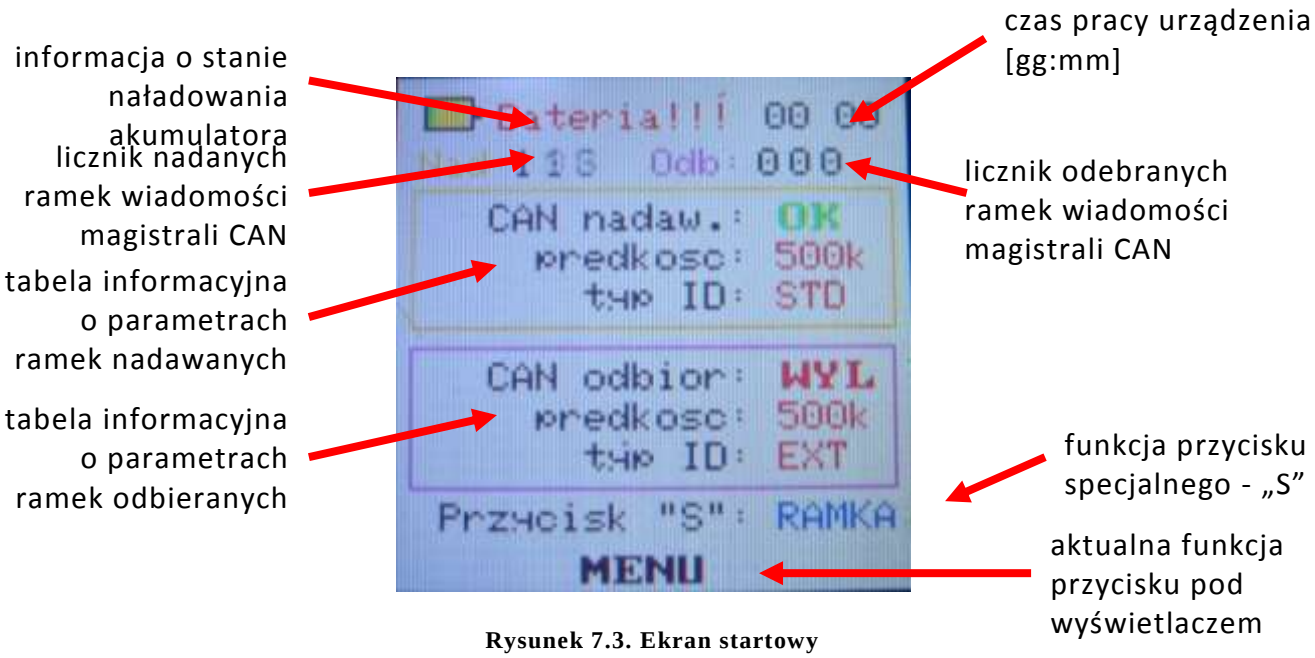
7.2. Wyświetlacz

Ponieważ wyświetlacz nie jest najważniejszym elementem wprowadzono system flag, aby inne ważniejsze procesy tj. obsługa interfejsu CAN i inne, nie były zablokowane.



Rysunek 7.2. Przebieg inicjalizacji wyświetlacza

Po uruchomieniu funkcji z rysunku 7.2, na ekranie pojawia się ekran startowy widoczny na rysunku 7.3.



Rysunek 7.3. Ekran startowy

Rysunek 7.4. Widok ekranu głównego menu

Po przyciśnięciu przycisku znajdującego się pod wyświetlaczem, narysowane zostaje graficzne menu widoczne na rysunku 7.4. Gdy się to stanie ustawiane są odpowiednie flagi, które uruchamiają miganie prostokąta wyboru funkcji oraz obsługę przycisków.

Rysunek 7.5. Ekran ustawiania kontrastu

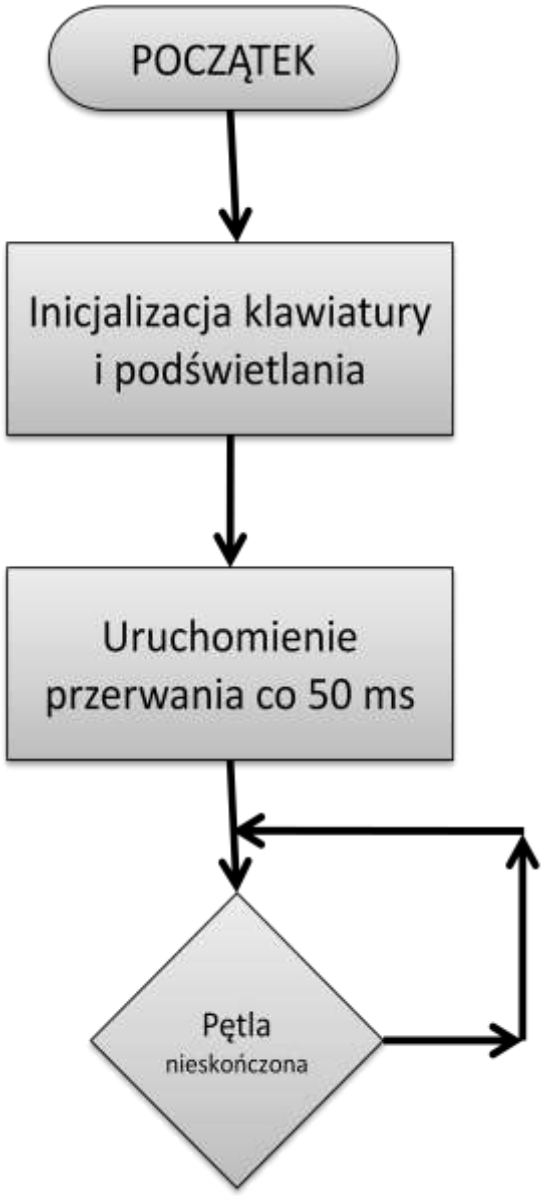
Każdy wyświetlacz różni się między sobą. Pomimo odpowiedniego dobrania parametrów już na etapie projektowania, może okazać się, że przy zastosowaniu innego wyświetlacza, ustawienia trzeba skorygować. Najbardziej krytyczną wartością pod tym względem jest kontrast, który można indywidualnie ustawić. Ekran do ustawiania kontrastu jest przedstawiony na rysunku 7.5.

7.3. Klawiatura z podświetlaniem

W związku z tym, że włączenie diod zwiększa zapotrzebowania na prąd urządzenia, zastosowano ich gaszenie po upływie określonego czasu od naciśnięcia jakiegokolwiek guzika z klawiatury. Czas po jakim czasie diody gasną jest regulowany, dzięki czemu można go optymalnie dopasować do indywidualnych upodobań (rysunek 7.6).

Rysunek 7.6. Ekran regulacji długości świecenia

Rysunek 7.7 pokazuje proces uruchomienia obsługi klawiatury. Jest on bardzo krótki ponieważ po ustawieniu portów i innych potrzebnych parametrów sprawdzanie stanu klawiszy opiera się na przerwaniu widocznym na rysunku 7.8.



Rysunek 7.7. Programowa obsługa klawiatury.

Do inicjalizacja klawiatury służy jedna funkcja InitKlawiatura(), która została załączona poniżej:

```
void InitKlawiatura(void)
{
	//konfiguracja podswietlania
	CFG_PeripheralClockConfig(CFG_CLK_GPIO5, ENABLE);
	GPIO0_InitSpiLcd.GPIO_Mode = GPIO_Mode_OUT_PP;
	GPIO0_InitSpiLcd.GPIO_Pins = GPIO_PIN_9;
	klaw_potsw_off;
	GPIO_Init (GPIO5, &GPIO0_InitSpiLcd);
	czas_pods_klaw = 10; //podswietlanie klawiatury na 10sekund
	czas_pods_klaw_aktualny = czas_pods_klaw;

	//konfiguracja klawiszy

	CFG_PeripheralClockConfig(CFG_CLK_GPIO0, ENABLE);
	GPIO0_InitSpiLcd.GPIO_Mode = GPIO_Mode_INOUT_WP;
```

```
      GPIO0_InitSpiLcd.GPIO_Pins = GPIO_PIN_0 | GPIO_PIN_1 |
GPIO_PIN_2 | GPIO_PIN_3 | GPIO_PIN_4;
      GPIO_Init (GPIO0, &GPIO0_InitSpiLcd);

      CFG_PeripheralClockConfig(CFG_CLK_GPIO1, ENABLE);
      GPIO0_InitSpiLcd.GPIO_Mode = GPIO_Mode_OUT_PP;
      GPIO0_InitSpiLcd.GPIO_Pins = GPIO_PIN_8 | GPIO_PIN_9 |
GPIO_PIN_10 | GPIO_PIN_11;
      GPIO_BitWrite(GPIO1, KL_K4, Bit_RESET);
      GPIO_BitWrite(GPIO1, KL_K3, Bit_RESET);
      GPIO_BitWrite(GPIO1, KL_K2, Bit_RESET);
      GPIO_BitWrite(GPIO1, KL_K1, Bit_RESET);
      GPIO_Init (GPIO1, &GPIO0_InitSpiLcd);

}
```

Rysunek 7.8. Obsługa przerwania co 50ms wykorzystywanego przez klawiaturę

Obsługa klawiatury wykorzystuje przerwania licznika TB1 mikrokontrolera, co 50ms. Poprzez dobór takiego okresu udało się w prosty sposób poradzić sobie z efektem drgania styków, który mógłbym powodować błędne odczytanie stanu klawiszy przez mikrokontroler. Mikrokontroler po kolei sprawdza stan wszystkich klawiszy i odpowiednim przypadku podejmuje odpowiednie działania dotyczące ich dalszej obsługi. Dzieje się to na zasadzie sprawdzania kolumn, w jakie są połączone przyciski. potem kolumna klawiszy jest przełączana na kolejną itd.

Funkcja przełączająca wybraną kolumnę klawiszy do odczytu:

```
void Wybierz_Klawisze(unsigned int klawisze)
{
	GPIO_BitWrite(GPIO1, KL_K4, Bit_RESET);
	GPIO_BitWrite(GPIO1, KL_K3, Bit_RESET);
	GPIO_BitWrite(GPIO1, KL_K2, Bit_RESET);
	GPIO_BitWrite(GPIO1, KL_K1, Bit_RESET);
	GPIO_BitWrite(GPIO1, klawisze, Bit_SET);
}
```

Części funkcji przerwania sprawdzająca stan klawiszy:

```
Wybierz_Klawisze(KL_K1); //wybór kolumny pierwsszej

if (KL_W5)
	{
		Menu_WGORE();
	}
if (KL_W2) Menu_WDOL();
if (KL_W3) Menu_LEWO();
if (KL_W4) Menu_PRAWO();
if (KL_W1) Menu_OK();

Wybierz_Klawisze(KL_K2);


Wybierz_Klawisze(KL_K4);

if (KL_W1) Menu_ESC();
```

7.4. Obsługa interfejsu RS-485

Przykładowa funkcja uruchamiająca układ peryferyjny UART znajduje się poniżej:

```
void uart_init(void)
{
	/*Enable Clock for UART0*/
	  CFG_PeripheralClockConfig(CFG_CLK_UART0 , ENABLE);
	/*Enable Clock for GPIO6*/
	  CFG_PeripheralClockConfig(CFG_CLK_GPIO6 , ENABLE);
```

```
            /* EIC Clock Enable */
             CFG_PeripheralClockConfig(CFG_CLK_EIC , ENABLE);

        GPIO_uart_InitStruct.GPIO_Mode = GPIO_Mode_AF_PP;
        GPIO_uart_InitStruct.GPIO_Pins = UART0_Tx_Pin ;
        GPIO_Init (GPIO6, &GPIO_uart_InitStruct);

        /* UART Configuration */
          UART_StructInit(&UART_InitStruct);             /*Initialize
Structure with default values*/
          UART_InitStruct.UART_BaudRate  = 115200;         /* Baud
rate value */
          UART_InitStruct.UART_Mode = UART_Mode_8D_P ;   /*Set the
UARTx Mode  8 bit Data + Parity */
          UART_InitStruct.UART_Loop_Standard = UART_Standard;
/*Disable LoopBack*/
          UART_InitStruct.UART_StopBits = UART_StopBits_1; /* STOP
bits number */
          UART_InitStruct.UART_Rx = UART_Rx_Enable;
          UART_InitStruct.UART_FIFO =
UART_FIFO_Enable;//UART_FIFO_Disable;
          UART_Cmd(UART0, ENABLE);

          UART_Init(UART0, &UART_InitStruct);   /*Configure Uart0*/

          UART_ITConfig(UART0, UART_IT_RxBufFull, ENABLE);

          EIC_IRQChannelConfig(UART0_IRQChannel, ENABLE );
             EIC_IRQChannelPriorityConfig(UART0_IRQChannel,13);
             EIC_IRQCmd(ENABLE);
          //UART_ByteSend(UART0, 'A');   /*Send "A" character*/
}
```

8. ZAKOŃCZENIE

Celem pracy było zbudowanie działającego urządzenia, które pozwoli monitorować ruch na magistrali CAN. Efektem pracy jest urządzenie, które na to pozwala, a ponadto posiada dodatkowe możliwości w postaci m. in. dodatkowej obsługi magistrali RS-485, jak również woltomierza. Możliwy jest odczyt na magistrali CAN dowolnych wiadomości z dowolną prędkością. Poprzez specjalne połączenie dwóch kanałów CAN, urządzenie można wpiąć do magistrali w celu odczytywania informacji również z dowolną polaryzacją. Dedykowana linia CAN umożliwiająca wysyłanie dowolnych wiadomości z dowolną prędkością, co daje bardzo duże możliwości diagnostyczne. Całe urządzenie jest małe i poręczne. Kolorowy czytelny wyświetlacz graficzny o dużej rozdzielczości sprawia, że analiza danych oraz cała obsługa urządzenia jest bardzo intuicyjna. Użyto specjalną klawiaturę membranową z podświetlaniem. Jej znakomitą właściwością jest duża miniaturyzacja oraz niezawodność działania, a dzięki podświetlaniu praca z urządzeniem stała się możliwa także w ciemności. Zastosowano łatwo dostępny akumulator litowo-jonowy wraz z zaawansowanym blokiem zasilania, co znacznie podniosło efektywność układu tym samym minimalizując straty energii. Ponieważ wydajne zasilanie akumulatorowe wystarcza na długie godziny pracy urządzenia, a mocna i solidna obudowa oraz wyraźny wyświetlacz są również ważnymi cechami, jakie udało się w nim zrealizować. Urządzenie można nazwać według początkowych założeń: mobilnym. Dzięki zapisowi przechwyconych ramek wiadomości na magistrali CAN, możliwa stała się późniejsza interpretacja i analiza większości zdarzeń, jakie pojawiają się na magistrali. Zastosowanie elementów możliwie najmniej pobierających prąd, oraz zastosowanie wydajnego akumulatora litowo-jonowego rejestracje zdarzeń można prowadzić przez długi czas. Duży problem sprawiało przylutowanie małych elementów powierzchniowych. Pomimo wielu starań łatwo było je przepalić, lub uszkodzić ścieżki na płytce, co niestety miało miejsce i konieczne stało się użycie kolejnej płytki.

Pomimo napotkanych wielu trudności, udało się rozwiązać większość z nich, co dało dużo wiedzy na przyszłość o projektowaniu i uruchamianiu tego typu urządzeń.

Mały wycinek KoNaRu

Na koniec składam ogromne podziękowania panu dr inż. Jarosławowi Emilianowiczowi za wiele pomysłów oraz uwag dotyczących realizacji projektu, Kołu Naukowemu Systemów Wbudowanych oraz Kołu Naukowemu Robotyków KoNaR, a w szczególności Robertowi Budzińskiemu, bez którego inicjatywy ta publikacja, by nie powstała.

BIBLIOGRAFIA

[1] Mikrokontrolery z rdzeniem ARM, część 1,
Elektronika praktyczna, 12/2005 str. 90-93
[2] http://pl.wikipedia.org/wiki/Architektura_ARM, pobrano 6.06.2009
[3] ST Microelectronics STR73x microcontroller datasheet. 2008
[4] http://reifel.org/PICUserInterface/, pobrano 06.06.2009
[5] Nokia 6100 LCD Display Driver, Revision 1, James P. Lynch
[6] http://www.forum.elektronika.xorg.pl/topic/196-interface-szeregowy-rs-485/, pobrano 6.06.2009
[7] http://www.elektronikab2b.pl/content/view/3404/lang,pl/,
pobrano 6.06.2009
[8] http://www.automatykab2b.pl/component/option,com_content/
Itemid,58 /id,2528/view,article/, pobrano 6.06.2009
[9] http://free.of.pl/c/can1/index.htm, pobrano 6.06.2009
[10] http://pl.wikipedia.org/wiki/Przetwornik_analogowo-cyfrowy,
pobrano 5.04.2009
[11] BURR-BROWN - ADS 7835 datasheet
[12] http://pl.wikipedia.org/wiki/RS485, pobrano 5.04.2009