

SPRAWOZDANIE ZE ZREALIZOWANYCH PRAC BADAWCZYCH – TEMAT 1

NR PROJEKTU	POIG.01.03.01-14-016/08-12
Tytuł projektu	Nowe materiały fotoniczne i ich zaawansowane zastosowania
Kierownik projektu	prof. dr hab. inż. Leszek R. Jaroszewicz
Temat badawczy I	Otrzymanie nowych związków ciekłokrystalicznych oraz wytworzenie mieszanin użytkowych nematyków o zróżnicowanej anizotropii optycznej i elektrycznej, ferro- i antyferroelektrycznych smektyków oraz kompozytów ciekłokrystalicznych jak i badania ich zastosowań.
Koordynator Tematu I	prof. dr hab. inż. Krzysztof Czupryński
Realizowany podtemat lub podtematy według harmonogramu Projektu	Opracowanie i wytworzenie zamkniętego układu detekcji do grupy rotacyjnych sejsmografów światłowodowych umożliwiających ich zastosowanie w sejsmologii rotacyjnej – Zespół fizyczny pod kierunkiem prof. dr hab. inż. Leszka R> Jaroszewicza
Okres sprawozdawczy	2009 - 2015

1. Wprowadzenie i geneza badań

Prowadzone od początku bieżącego wieku w Zakładzie Technicznych Zastosowań Fizyki Instytutu Fizyki Technicznej WTC WAT prace nad budową światłowodowego sejsmometru rotacyjnego doprowadziły do konstrukcji dwu klas przyrządów: układu dla badań sejsmologicznych związanych z trzęsieniami Ziemi klasy AFORS – autonomicznego światłowodowego sejsmografu rotacyjnego (-01 – zainstalowanego w zamku Książ, -02 – używanego do badań w WAT oraz -03 – pierwotnie zainstalowanego w obserwatorium sejsmologicznym PAN w Ojcowie) oraz układów dla badań ruchów rotacyjnych budowli klasy FOSREM – światłowodowego sejsmometru do monitoringu ruchów skrętnych (-01, -02, -BB – eksploatowanych przez IFT WAT w kooperacji z IG PAN). W ramach projektu POIG obie klasy przyrządów były wykorzystywane do zdalnego monitoringu efektów i zjawisk rotacyjnych, zaś wyniki tych prac opublikowano m.in. w postaci 3 rozdziałów w anglojęzycznych monografiach (Springer - 2 pozycje, InTech).

Zdobyte doświadczenie pozwoliło na sformułowanie dodatkowych wymagań na uniwersalny układ obróbki sygnału oraz danych z tych układów bazujący na koncepcji zamkniętego układu detekcyjnego wykorzystującego układ MIOC - (Multi-Functional Integrated Optics Chip) jako aktywnego optycznego elementu planarnego zawierającego m.in. układ modulatora fazowego pozwalającego na szerokopasmowe sterowanie, co umożliwia realizację układu z tzw. śledzeniem zera na bazie układu z zamkniętą pętlą sprzężenia zwrotnego. Realizacja tego układu – koncepcja, wykonanie prototypu oraz jego przebadanie zostało ostatecznie zrealizowane w projekcie jako usługa badawcza przez firmę InfoSCAN S.A., którego wynik jest prezentowany poniżej jako załącznik do tego sprawozdania.

AFORS-MIOC Układ detekcji

Opracowanie

wersja 1.0 • 2015-10-26

*InfoSCAN S.A.
Nowogrodzka 50
00-695 Warszawa
Polska*

Copyright © 2015 InfoSCAN S.A. All rights reserved.

No part of this publication may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without prior written permission from InfoSCAN S.A..

All copyright, confidential information, patents, design rights and all other intellectual property rights of whatsoever nature contained herein are and shall remain the sole and exclusive property of InfoSCAN S.A. The information furnished herein is believed to be accurate and reliable.

However, no responsibility is assumed by InfoSCAN S.A. for its use, or for any infringements of patents or other rights of third parties resulting from its use.

The InfoSCAN S.A. name and InfoSCAN S.A. logo are trademarks or registered trademarks of InfoSCAN S.A.

All other trademarks are the property of their respective owners

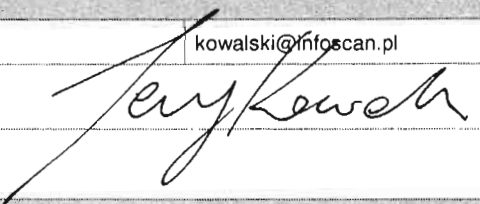
Opracowanie koncepcji

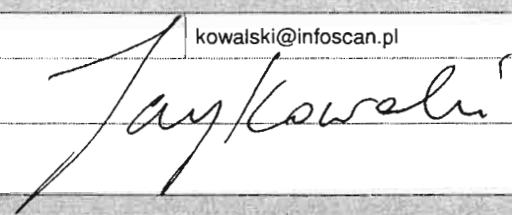
InfoSCAN Dział R&D

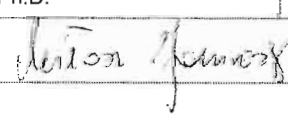
AFORS-MIOC Układ detekcji

Wersja: 1.0

Data wykonania: 2015-10-26

Executive Sponsor		
Jerzy K. Kowalski Ph.D.	kowalski@infoscan.pl	+48609106337
Podpis		Data 26.10.2015

Technology Sponsor		
Jerzy K. Kowalski Ph.D.	kowalski@infoscan.pl	+48609106337
Podpis		Data 26.10.2015

Project Manager		
Miłosz Jamróży Ph.D.	jamrozy@infoscan.pl	
Podpis		Data 26.10.2015

Spis Treści

1	WSTĘP	4
1.1	CEL PROJEKTU.....	4
1.2	OCZEKIWANY WYNIK PROJEKTU	4
2	WYMAGANIA.....	4
2.1	WYMAGANIA UKŁADU OPTYCZNEGO	4
2.2	WYMAGANIA UKŁADU ELEKTRONICZNEGO	4
2.3	WYMAGANIA UKŁADU TRANSMISJI DANYCH.....	5
2.4	WYMAGANIA ORGANIZACYJNE	5
3	KONCEPCJA	5
3.1	PODSTAWOWA KONCEPCJA UKŁADU POMIAROWEGO	5
3.2	KONCEPCJA UKŁADU DETEKCJI.....	8
3.3	KONCEPCJA ZASILANIA I TELETRANSMISJI	10
4	WYMAGANIA KONSTRUKCYJNE	11
4.1	WYMAGANIA DLA UKŁADU POMIAROWEGO.....	11
4.2	WYMAGANIA DLA UKŁADU ZASILANIA.....	12
4.3	WYMAGANIA DLA UKŁADU TELETRANSMISJI	13
5	PROJEKT KOSZTORYSU	14
6	WNIOSKI	15

Spis rysunków

<i>Rys. 1. Schemat blokowy projektowanego układu</i>	<i>5</i>
<i>Rys. 2. Sygnał cyfrowej pily zsynchronizowanej z sygnałem prostokątnej modulacji</i>	<i>8</i>
<i>Rys. 3. Odpowiedź interferometru przy prostokątnej modulacji</i>	<i>9</i>
<i>Rys. 4. Przesunięcie fazy dla modulacji przebiegiem piłokształtnym.....</i>	<i>9</i>
<i>Rys. 5. Układ przetwarzania sygnałów</i>	<i>12</i>
<i>Rys. 6. Schemat blokowy układu zasilania PoE.....</i>	<i>13</i>
<i>Rys. 7. Układ teletransmisji i łączności</i>	<i>14</i>

1 Wstęp

1.1 Cel projektu

Celem projektu jest opracowanie koncepcji zamkniętego układu detekcji dla czujników klasy AFORS na bazie zintegrowanych układów optycznych MIOC. W ramach zamówienia dostarczono specyfikację techniczną układu MIOC (Multi Functional Integrated Optics Chip) jako aktywnego optycznego elementu planarnego zawierającego m.in. układ modulatora fazowego pozwalającego na szerokopasmowe sterowanie co umożliwia realizację układu z tzw. śledzeniem zera w układzie z zamkniętą pętlą sprzężenia zwrotnego.

1.2 Oczekiwany wynik projektu

Oczekiwany wynik projektu jest przedstawienie opisu koncepcji i wymagań konstrukcyjnych pozwalających na sporządzenie dokumentów do zamówienia wykonania realizacji elektronicznego układu detekcji.

2 Wymagania

2.1 Wymagania układu optycznego

1. Opracowane rozwiązanie ma zapewnić działanie światłowodowego sejsmometru rotacyjnego w tzw. konfiguracji minimalnej, której konstrukcja nie jest elementem zamówienia.
2. Rozwiązanie dotyczy wykorzystania układu MIOC (Multi Functional Integrated Optics Chip) według dostarczonej specyfikacji (Examining_Sheet_1240305_EN.pdf) jako elementu aktywnego pozwalającego na realizację układu z tzw. śledzeniem zera.

2.2 Wymagania układu elektronicznego

1. Wymagane jest rozwiązanie elektroniczne, które zostanie bezpośrednio zaimplementowane do układów klasy AFORS oraz FOSREM zarówno pod względem wykorzystania dostępnego zasilania prądowego jak i generacji sygnałów wyjściowych oraz współpracy z zewnętrznym system telemetrycznym (Internet).
2. Dolny zakres pomiarowy dla AFORS i FOSREM z opracowanym rozwiązaniem nie może być wyższy od połowy zakresu pomiarowego tych urządzeń dla określonych szerokości pasm detekcji (od 0.1 Hz do 300 Hz).
3. Planowany teoretycznie górny zakres pomiarowy powinien wynosić 10π dla pasma od 0.1 Hz do 100 Hz.

2.3 Wymagania układu transmisji danych

1. Opracowany układ ma zapewniać możliwość wysyłania danych pomiarowych jak również możliwość konfiguracji parametrów wewnętrznych poprzez sieć Internet z wykorzystaniem protokołu HTTP/HTTPS.
2. Należy zapewnić możliwość poprawnej pracy w sieciach publicznych GSM/3G/LTE
3. Układ powinien mieć możliwość sterowania i kontroli za pośrednictwem przeglądarki internetowej.

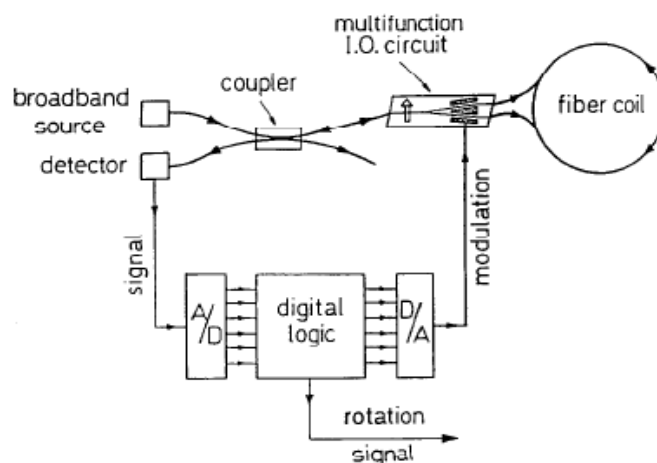
2.4 Wymagania organizacyjne

1. Konieczne jest określenie kosztów wytworzenia rozwiązania prototypowego jak i kosztów zestawu 10 układów.
2. Realizacja zamówienia musi bazować na zawartej umowie o współpracy naukowo-badawczej pomiędzy Zamawiającym i WAT. W przypadku braku takiej umowy Zamawiający przed realizacją zamówienia zobowiązuje się do jej zawarcia.

3 Koncepcja

3.1 Podstawowa koncepcja układu pomiarowego

Na podstawie wymagań określono podstawową koncepcję układu detekcyjnego pokazaną na schemacie blokowym rys. 1.



Rys. 1. Schemat blokowy projektowanego układu

Układ składa się z następujących bloków funkcjonalnych:

1. Układ optyczny składający się z szerokopasmowego źródła optycznego (broadband source), sprzęgacza optycznego (coupler), układu MIOC (multifunctional I.O. circuit), pętli światłowodowej (fiber coil) oraz detektora (detector),
2. Układ wzmacniacza sygnału z detektora,
3. Przetwornik Analogowo-Cyfrowy (A/D),
4. Cyfrowy Układ Logiczny (digital logic),
5. Przetwornik Cyfrowo-Analogowy pozwalający na sterowanie układem MIOC (D/A)
6. Układ przesyłania informacji o prędkości obrotowej (rotational signal).

Zaprojektowany układ jest przystosowany do pracy z układem optycznym o dwu zasadniczo różnych parametrach funkcjonalnych:

- a. układem z pętlą o długości ok. 15.000 m – wówczas na detektorze spodziewany jest sygnał o mocy optycznej $P=80 \mu W$, co dla czułości zastosowanej fotodiody $S=0.90 A/W$ daje prąd o natężeniu $I=70 \mu A$, zaś częstotliwość własna układu, czyli częstotliwość modulacji MIOC sygnałem prostokątnym wynosi około 6.800 Hz. W tym wypadku układ powinien zapewniać detekcję prędkości obrotowej z zakresu 10^{-8} rad/s do 10^{-6} rad/s w paśmie częstotliwości od 0.1 Hz do 100 Hz.
- b. układem z pętlą o długości ok. 5.000 m – wówczas na detektorze spodziewany jest sygnał o mocy optycznej $P=220 \mu W$, co dla czułości zastosowanej fotodiody $S=0.90 A/W$ daje prąd o natężeniu $I=200 \mu A$, zaś częstotliwość własna układu, czyli częstotliwość modulacji MIOC sygnałem prostokątnym wynosi około 20.450 Hz. W tym wypadku układ powinien zapewniać detekcję prędkości obrotowej z zakresu 10^{-6} rad/s do 10 rad/s w paśmie częstotliwości od 0.1 Hz do 100 Hz.

W obu wypadkach układ ma możliwość regulacji częstotliwości pracy w zakresie ± 300 Hz wokół wyżej wymienionych częstotliwości własnych, w celu doboru optymalnych parametrów pracy dla konkretnego układu. Natomiast zakres napięciowy generowanych przebiegów powinien mieścić się w przedziale ± 10 V.

Zaprojektowany układ jest przystosowany do współpracy z układem MOIC schematycznie przedstawionym poniżej (dane katalogowe), którego główne parametry podano w Tabeli 1 oraz Tabeli 2.

Tabela 1. Schematyczny układ MIOC

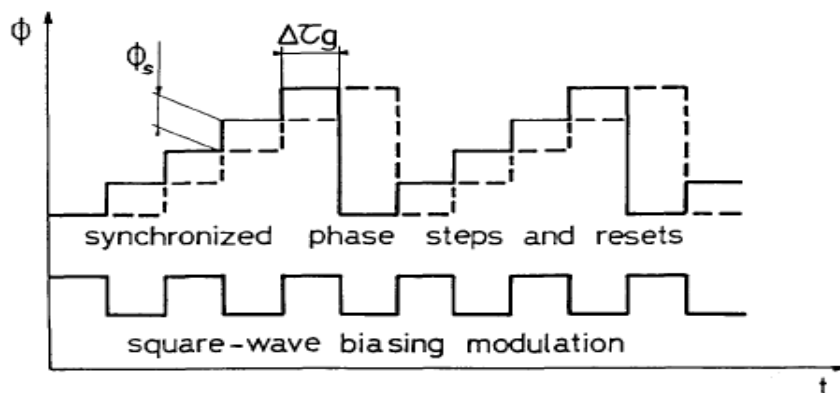
Tabela 2. Ogólne parametry układu MIOC

© InfoSCAN 2016. All rights reserved

3.2 Koncepcja układu detekcji

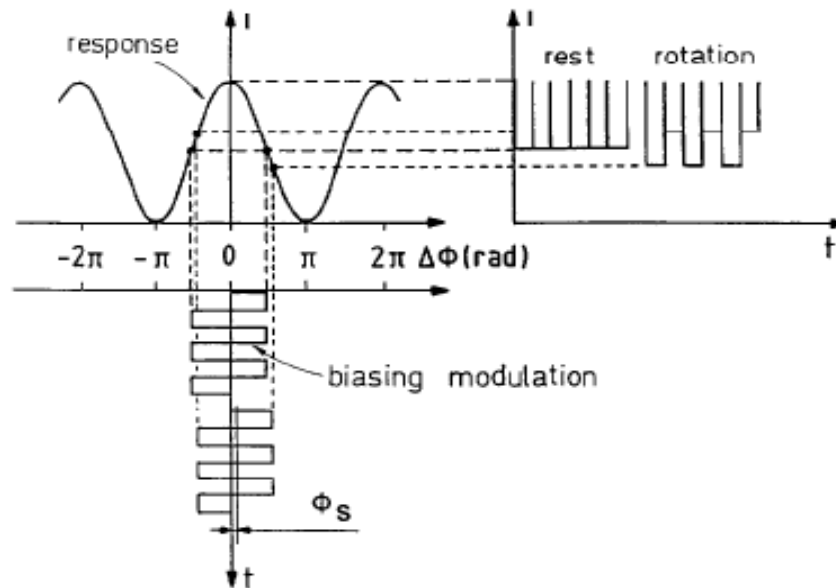
Istotą rozwiązania jest zastosowanie cyfrowej pętli sprzężenia zwrotnego, której zadaniem jest utrzymywanie interferometru w położeniu zerowym (zerowanie przesunięcia fazy wprowadzanego przez efekt Sagnaca). Takie rozwiązanie pozwala zapewnić liniowość skali oraz możliwość rozszerzenia zakresu pomiarowego.

Dlatego do sterowania modulatorem fazy występującym w MIOC przyjęto koncepcję zastosowania złożonego sygnału: pily i prostokąta, co przedstawiono schematycznie na rys.2. Sygnał ten powinien charakteryzować się amplitudą równą 2π lub wielokrotnością 2π . Czas trwania schodków równa się czasowi przebiegu fali optycznej przez pętlę czujnikową $\Delta\tau_g = cL/n$, gdzie c – prędkość światła w próżni, L – długość pętli, n – współczynnik załamania dla włókna. Natomiast wysokość schodków jest równy wprowadzonemu przesunięciu fazy Sagnaca $\phi_s = 4(\pi RL/\lambda c)\Omega$, gdzie R – promień nawinięcia pętli, λ – długość fali światła, zaś Ω – prędkość kątowna obrotu. Takie rozwiązanie pozwala na wyzerowanie czujnika. Szukaną wielkość ϕ_s otrzymujemy bezpośrednio z cyfrowego układu pętli sprzężenia zwrotnego, która jest wprost proporcjonalna do Ω i tą drogą podlega wyliczeniu w cyfrowym układzie logicznym.



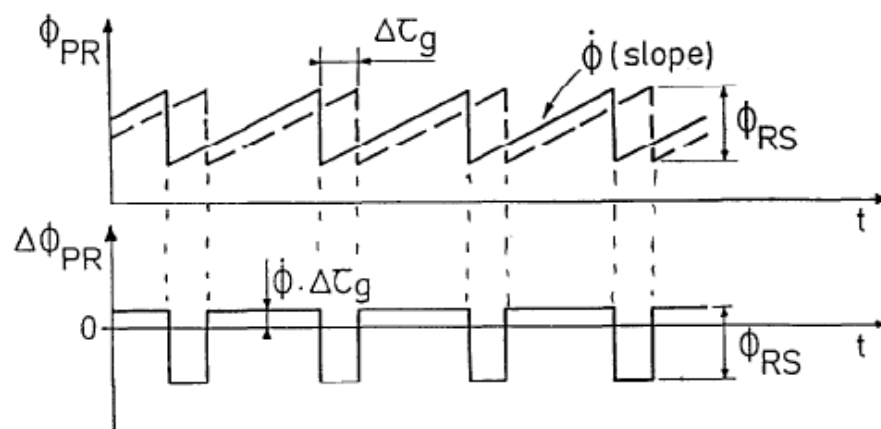
Rys. 2. Sygnał cyfrowej pily zsynchronizowanej z sygnałem prostokątnej modulacji

Dla poglądowego wyjaśnienia wymaganego rozwiązania na rys.3 przedstawiono spodziewaną odpowiedź interferometru dla modulacji prostokątnej o częstotliwości $f=1/2\Delta\tau_g$. W przypadku zerowego przesunięcia fazy (braku prędkości obrotowej Ω) odpowiedź interferometru składa się z wąskich szpilek, w miejscach przejścia sygnału modulującego przez zera oraz charakteryzuje się pewną składową stałą. Wprowadzenie nieodwracalnego przesunięcia Sagnaca (występowanie prędkości obrotowej Ω) powoduje zmodyfikowanie odpowiedzi interferometru w przybliżeniu do sygnału prostokątnego plus niekorzystne szpilki. Stąd zastosowanie odpowiednich metod demodulacji pozwalać będzie na odczytanie z takiego sygnału prędkości obrotowej, która wywołała dane przesunięcie fazy.



Rys. 3. Odpowiedź interferometru przy prostokątnej modulacji

Sugerowana i sprawdzona metoda demodulacji pokazana jest na rys. 4, gdzie poprzez wprowadzanie piłokształtnego przebiegu napięcia sterującego modulator fazy w MIOC, uzyskuje się przesunięcie fazy (pomiędzy dwoma przeciwnie propagującymi się w pętli czujnika promieniami światła) uzależnione od nachylenia przebiegu piłokształtnego oraz czasu $\Delta\tau_g$. Modyfikując nachylenie przebiegu piłokształtnego (zmiana napięcia zasilającego modulator) regulujemy wprowadzanie przesunięcie fazy, kompensując wielkość efektu Sagnaca. Tym sposobem sygnał sterujący modulator tak by nastąpiła kompensacja zmiany fazy (utrzymujący go w położeniu zerowym) jest sygnałem o szukanej wielkości fazy Sagnaca, co po odpowiedniej kalibracji powinno dawać sygnał o rejestrowanej prędkości obrotowej.



Rys. 4. Przesunięcie fazy dla modulacji przebiegiem piłokształtnym

Projektowany układ elektroniczny stosowany do obróbki i przetwarzania sygnałów musi zapewnić dynamikę pozwalającą na zapewnienie wymaganej dokładności w całym wymaganym zakresie prędkości obrotowych. W ramach układu logicznego powinien być zastosowany szybki układ FPGA zaprogramowany do generowania wymaganych sygnałów i przetwarzania ADC sygnałów analogowych z wymaganą częstotliwością i rozdzielczością.

Układ elektroniczny powinien zawierać również kompletny zestaw modułów pozwalających na realizację zadań wymaganych w pracy autonomicznej i grupowej z synchronizacją czasu zegarów wewnętrznych w tym:

1. Układ przetwarzania wyliczonego sygnału prędkości obrotowej, kierunku obrotu, zakresu pomiarowego do znormalizowanych wartości cyfrowych.
2. Układ pamięciowy pozwalający na zarejestrowanie i zapamiętanie do 30 dni ciągłych pomiarów z maksymalną częstotliwością z możliwością odczytu asynchronicznego w czasie rejestracji.
3. Układ sterujący pozwalający na obsługę wszystkich modułów wewnętrznych, kontrolę ich parametrów.
4. Układ pomiaru parametrów pracy urządzenia: temperatury wewnętrznej i zewnętrznej, przyspieszenia, położenia względem ziemi, prędkości obrotowej obudowy, położenia względem bieguna geograficznego, położenia geograficznego miejsca instalacji na bazie GPS/Galileo/Glonass. Możliwe jest uzyskiwanie położenia z zewnętrznej anteny GPS.
5. Układ powinien umożliwiać także synchronizację wewnętrznych zegarów czasu rzeczywistego w czasie pracy z jednostką centralną oraz z zewnętrznymi serwerami NTP.

3.3 Koncepcja zasilania i teletransmisji

Układ sterujący stosowany w układzie musi umożliwiać dostęp do parametrów oraz możliwość ustawiania ich poprzez interfejs typu WEB. Urządzenie powinno mieć wbudowany serwer WEB pozwalający na sterowanie i wizualizację aktualnych danych pomiarowych w szczególności zawierać:

1. Układ transmisji danych pozwalający na dołączenie urządzenia do jednostki centralnej w trybie pracy grupowej przy pomocy łącza min. 1000Mbps Gigabit Ethernet lub łącza światłowodowego.
2. Urządzenie powinno pozwalać na automatyczne zestawianie połączenia VPN z centralnym serwerem telemetrycznym wraz z automatycznym ponownym zestawianiem tunelu po przerwach w transmisji danych.
3. Urządzenie powinno zapewniać możliwość korzystania z sieci bezprzewodowych WWAN/WiFi w szczególności obsługiwać dostęp do sieci Internet poprzez sieci 3G/HDSPA oraz LTE.

4. Układ zasilania z wewnętrznym układem podtrzymania napięcia wystarczającym na min. 2 godz pracy bez zasilania zewnętrznego, układem zasilania PoE o wymaganej przez układ pomiarowy i elektroniczny mocy, układem zasilania z zewnętrznego źródła +12 - 24V.
5. Możliwość połączenia urządzenia do jednostki centralnej na odległość do 50 m przy pomocy przewodu FTP kat 6/7 oraz do 1000 m przy pomocy światłowodu.

Proponowane jest aby w interfejsie webowym znajdował się moduł pozwalający w czasie rzeczywistym obserwować bieżące wartości pomiarowe w trybie oscyloskopowym oraz parametry układu w postaci cyfrowej i graficznej.

Korzystnie jest aby układ sterująco-konwertujący umożliwiał przesyłanie aktualnych parametrów, danych pomiarowych, występujących zdarzeń, poprzez sieć Internet do centralnego serwera telemetrycznego. Transmisja musi być wykonywana z wykorzystaniem protokołu HTTP/HTTPS z autentykacją poszczególnych urządzeń pomiarowych. Transmisja musi być niezależna od topologii dostępnej sieci IP i nie może wymagać przydzielania publicznych adresów IP po stronie urządzenia.

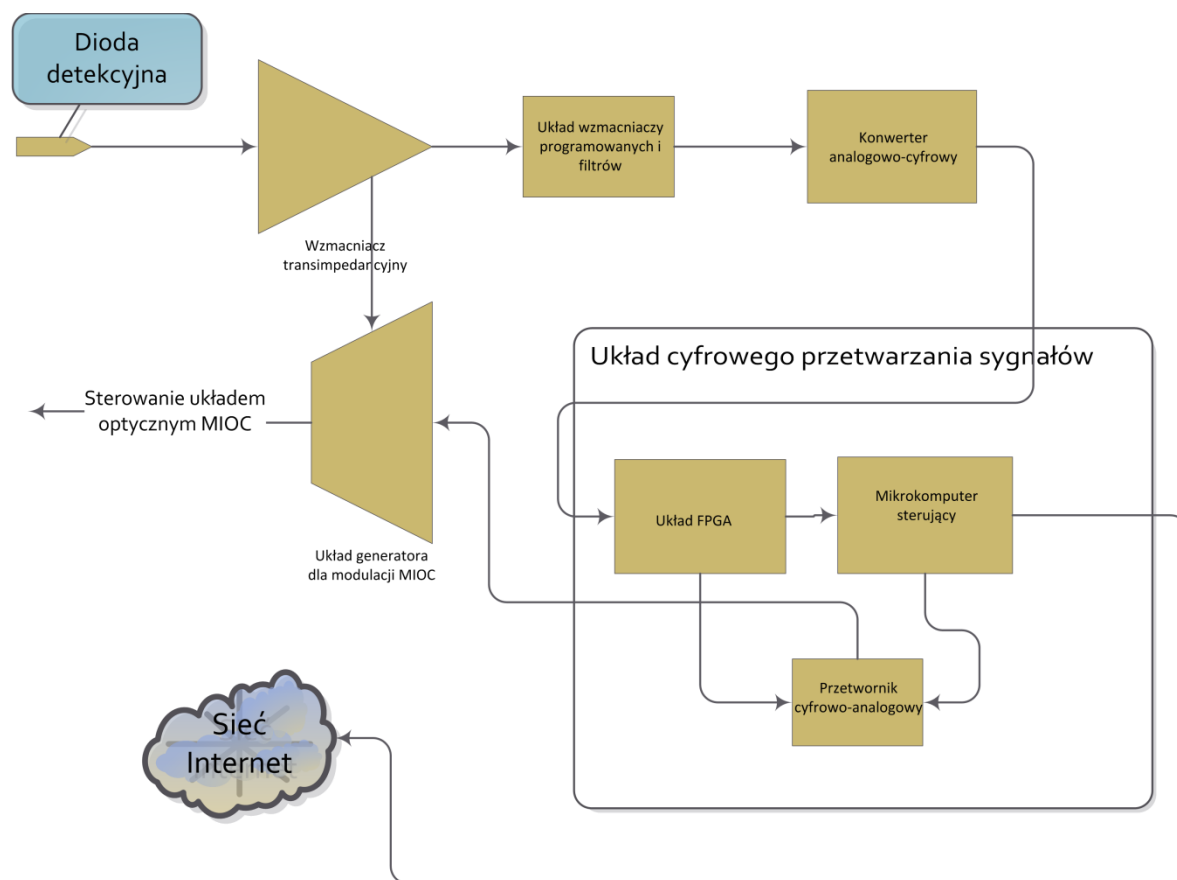
4 Wymagania konstrukcyjne

4.1 Wymagania dla układu pomiarowego

Układ pomiarowy według przedstawionej koncepcji będzie składał się z sensora detekcyjnego, dołączonego do niego wysokoczułego, niskoszumnego wzmacniacza o regulowanym wzmacnieniu wraz z cyfrowymi filtrami aktywnymi pozwalającymi na regulację parametrów.

Sygnał wyjściowy z układu wzmacniająco-filtrującego jest przekazywany na wejście szybkiego przetwornika analogowo-cyfrowego o dużej szybkości działania. Cyfrowy sygnał wyjściowy jest przekazywany do układu logicznego obliczającego wymagane parametry sterujące dla modulatora wbudowanego w MIOC. Proponuje się, że układ logiczny zostanie zbudowany na bazie szybkiego układu FPGA w którym zostaną zaprojektowane wymagane moduły logiczne, obliczeniowe i realizujące przekształcenia FFT na poziome sprzętowo.

Sygnał cyfrowy z układu logicznego jest przekazywany do przetwornika cyfrowo-analogowego, a wyjściowe napięcie po odpowiednim wzmacnieniu steruje modulatorem generującym przebieg piłokształtny dla układów optycznych.



Rys. 5. Układ przetwarzania sygnałów

Jednocześnie obliczany jest sygnał będący odwzorowaniem prędkości obrotowej. Ten sygnał po obróbce cyfrowej jest przekazywany do mikrokomputera sterującego gdzie jest zapisywany w pamięci masowej i jednocześnie używany do wizualizacji sygnału wynikowego.

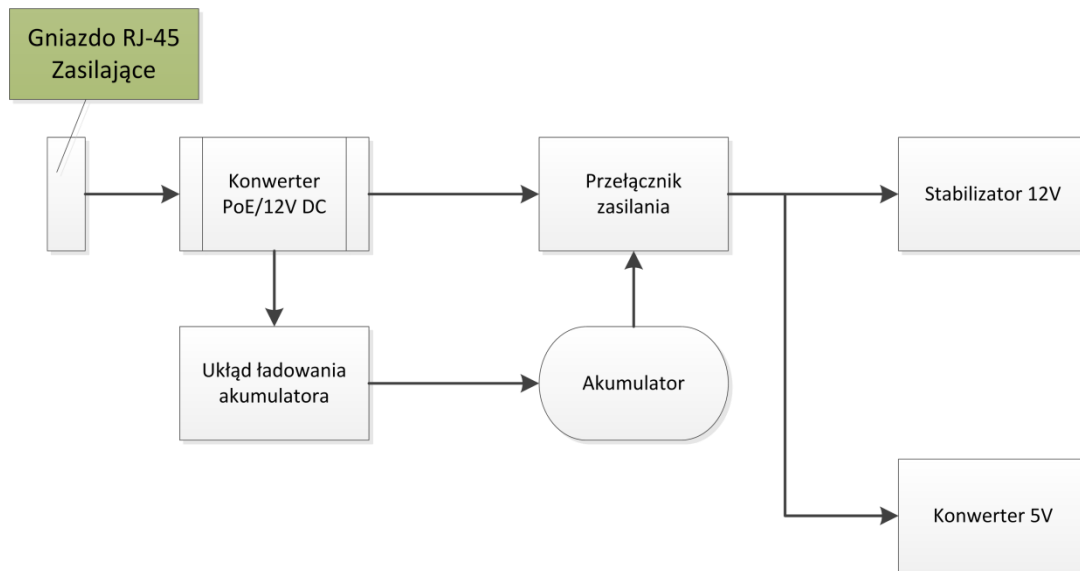
W układzie mikrokomputerowym realizowana jest także funkcja detekcji zdarzeń i wysyłania odpowiednich informacji do serwera telemetrycznego.

4.2 Wymagania dla układu zasilania

Cały układ elektroniczny zasilany jest z pojedynczego napięcia 12V z wbudowanym specjalizowanym akumulatorowym układem podtrzymania napięcia. Układ ten wykorzystuje akumulatory Li-On zapewniające pracę bez zasilania zewnętrznego przez min. 2 godziny. Przyjęto że całkowita moc pobierana przez cały układ elektroniczny oraz układ zasilania laserem nie przekroczy 15W. Dlatego wymagana pojemność akumulatora wewnętrznego została określona na 30Wh co przy napięciu akumulatora 7.2V (dwa ogniwa Li-On) daje pojemność na poziomie 4,2 Ah. Jednak przy założeniu całkowitej sprawności przetwornic stosowanych w układach zasilania na 80%, wymagana pojemność akumulatora wynosi ok 5,2 Ah. W związku z tym można zastosować akumulator o pojemności powyżej tej wartości dostępny na rynku.

Przyjęto, że zasilanie zewnętrzne będzie odbywało się poprzez Power over Ethernet – przy pomocy tego samego przewodu którym układ będzie podłączony do sieci Internet. Dostępne są gotowe rozwiązania zasilaczy i konwerterów pozwalających na uzyskanie w takim układzie wymaganej mocy.

Poniżej przedstawiono schemat blokowy układu zasilania:



Rys. 6. Schemat blokowy układu zasilania PoE

4.3 Wymagania dla układu teletransmisji

Układ teletransmisji został zaprojektowany do współpracy z siecią Internet. Wbudowany w mikrokomputer interfejs Ethernet 1000Mbps pozwoli na szybkie i wydajne przesyłanie danych pomiarowych w czasie rzeczywistym.

Wbudowany system operacyjny Linux wraz z serwerem WEB – Apache i serwerem baz danych MySQL pozwala na uruchomienie serwera www realizującego konsolę operatorską, wizualizację wyników pomiarowych i monitorowanie i sterowanie całym układem.

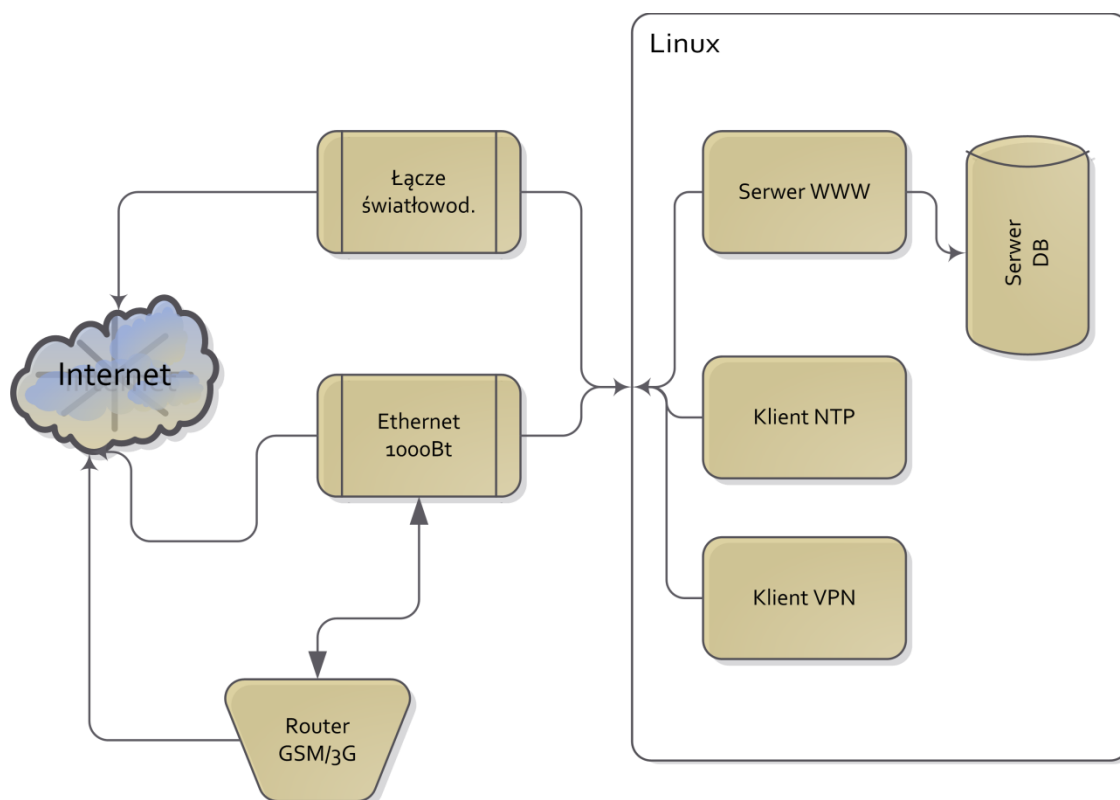
Opcjonalnie można dołączyć konwerter światłowodowy pozwalający na realizację łącza Ethernet przy pomocy światłowodu nawet do długości 2000m. W takim przypadku wymagane jest podłączenie do złącza Ethernet zasilacza PoE.

W ramach tworzenia lokalnej sieci, która może składać się z jednego lub większej liczby układów pomiarowych, możliwe jest dołączenie routera GSM/3G/LTE realizującego dostęp do aktualnie dostępnej sieci publicznej.

Każdy mikrokomputer wewnętrzny zapewnia funkcjonalność klienta VPN co pozwala na stworzenie rozległej bezpiecznej sieci czujników.

Możliwe jest uruchomienie lokalnego serwera telemetrycznego zapewniającego zasilanie dla wielu układów pomiarowych, przechowywanie danych przez okres dłuższy niż możliwy w wewnętrznej pamięci czujników, synchronizację czasu oraz routing do dostępnej sieci telekomunikacyjnej. W ten sposób można zrealizować sieć czujników w lokalnym miejscu, które będą pracowały z zsynchronizowanymi zegarami.

Schemat układu teletransmisji umieszczono poniżej:



Rys. 7. Układ teletransmisji i łączności

5 Projekt kosztorysu

Na bazie przedstawionej koncepcji opracowano wstępny kosztorys układu detekcyjnego (bez elementów optycznych). Zestawienie kosztów przedstawiono w tabeli poniżej. Umieszczone koszty nie zawierają podatku VAT.

Opis elementu	Kod	RAZEM		124 800,00 zł	724 200,00 zł	
		Cena	Liczba	Wartość	Liczba	Wartość
Moduł wzmacniacza transimpedancyjnego z izolacją galwaniczną	MAMP3	15 200,00 zł	1	15 200,00 zł	10	152 000,00 zł
Moduł wzmacniaczy programowanych i układów filtów	MFLT3	12 400,00 zł	1	12 400,00 zł	10	124 000,00 zł
Moduł konwertera analogowo-cyfrowego	MDAC3	8 300,00 zł	1	8 300,00 zł	10	83 000,00 zł
Moduł układów logicznych	MDIGI3	4 700,00 zł	1	4 700,00 zł	10	47 000,00 zł
Moduł mikrokomputera z układem FPGA	MKOMP3	5 300,00 zł	1	5 300,00 zł	10	53 000,00 zł
Oprogramowanie mikrokomputera	MPROG3	58 200,00 zł	1	58 200,00 zł	1	58 200,00 zł
Moduł zasilania i sterowania lasera	MLAS3	3 400,00 zł	1	3 400,00 zł	10	34 000,00 zł
Moduł generatora sygnału modulacji	MSINGEN3	2 800,00 zł	1	2 800,00 zł	10	28 000,00 zł
Moduł bazowy do układów analogowych z czujnikami MEMS	MBASE3	3 900,00 zł	1	3 900,00 zł	10	39 000,00 zł
Zasilacz PoE +12V	MPOW3	2 800,00 zł	1	2 800,00 zł	10	28 000,00 zł
Konwerter PoE 230V AC/54V PoE	EPOEINJ	2 100,00 zł	1	2 100,00 zł	10	21 000,00 zł
Obudowa i elementy mechaniczne	MMECH3	3 200,00 zł	1	3 200,00 zł	10	32 000,00 zł
Montaż końcowy i uruchomienie		2 500,00 zł	1	2 500,00 zł	10	25 000,00 zł

Uwaga: koszty oprogramowanie nie multiplikują się z liczbą wykonanych egzemplarzy.

Z wyliczeń wynika, że koszt wykonania jednego egzemplarza powinien zamknąć się na poziomie ok 125 tys. zł, zaś koszty wykonania serii 10 egzemplarzy w kwocie ok 725 tys. zł. Trzeba zaznaczyć, że wymienione koszty nie zawierają kosztów modułu laserowego, diody detekcyjnej oraz wymaganych elementów optycznych w tym MIOC, pętli światłowodowej oraz sprzęgaczy. Dokładne koszty można określić po wykonaniu projektu technicznego i kosztorysu szczegółowego wymaganych elementów z uwzględnieniem stosowanej technologii i preferowanych dostawców.

6 Wnioski

Zaprojektowany układ pomiarowy realizuje wymagania postawione projektowi. Koncepcja układu oraz założenia pozwolą na skonstruowanie prototypów realizujących wymagane funkcjonalności w konkretnej technologii.

AFORS-MIOC Układ detekcji

Projekt techniczny

wersja 1.0 • 2015-12-27

*InfoSCAN S.A.
Nowogrodzka 50
00-695 Warszawa
Polska*

Copyright © 2015 InfoSCAN S.A. All rights reserved.

No part of this publication may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without prior written permission from InfoSCAN S.A.

All copyright, confidential information, patents, design rights and all other intellectual property rights of whatsoever nature contained herein are and shall remain the sole and exclusive property of InfoSCAN S.A. The information furnished herein is believed to be accurate and reliable.

However, no responsibility is assumed by InfoSCAN S.A. for its use, or for any infringements of patents or other rights of third parties resulting from its use.

The InfoSCAN S.A. name and InfoSCAN S.A. logo are trademarks or registered trademarks of InfoSCAN S.A.

All other trademarks are the property of their respective owners

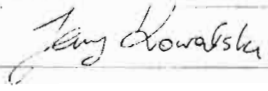
Opracowanie projektu technicznego zamkniętego układu detekcyjnego dla czujników klasy AFORS na bazie modułu MIOC

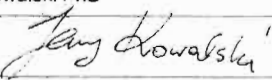
InfoSCAN Dział R&D

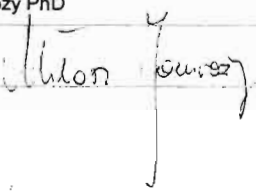
AFORS-MIOC Układ detekcji

Wersja: 1.0

Data wykonania: 2015-12-27

Executive Sponsor		
Jerzy K. Kowalski PhD	kowalski@infoscan.pl	+48609106337
Podpis 		Data: 2015-12-27

Technology Sponsor		
Jerzy K. Kowalski PhD	kowalski@infoscan.pl	+48609106337
Podpis 		Data: 2015-12-27

Project Manager		
Miłosz Jamroży PhD	jamrozy@infoscan.pl	
Podpis 		Data: 2015-12-27

Spis Treści

SPIS RYSUNKÓW I TABEL.....	5
1 WSTĘP.....	7
1.1 CEL PROJEKTU.....	7
1.2 OCZEKIWANY WYNIK PROJEKTU	7
2 WYMAGANIA.....	7
2.1 WYMAGANIA UKŁADU OPTYCZNEGO.....	7
2.2 WYMAGANIA UKŁADU ELEKTRONICZNEGO	7
2.3 WYMAGANIA UKŁADU TRANSMISJI DANYCH.....	8
2.4 WYMAGANIA ORGANIZACYJNE	8
3 KONCEPCJA	8
3.1 PODSTAWOWA KONCEPCJA UKŁADU POMIAROWEGO	8
3.2 KONCEPCJA UKŁADU DETEKCJI.....	11
3.3 KONCEPCJA ZASILANIA I TELETRANSMISJI	13
4 WYMAGANIA KONSTRUKCYJNE.....	14
4.1 WYMAGANIA DLA UKŁADU POMIAROWEGO.....	14
4.2 WYMAGANIA DLA UKŁADU ZASILANIA.....	15
4.3 WYMAGANIA DLA UKŁADU TELETRANSMISJI	16
5 PROJEKT TECHNICZNY UKŁADU DETEKCJI AFORS-MIOC.....	18
5.1 BUDOWA UKŁADU AFORS-MIOC	18
5.2 UKŁAD CYFROWY UKŁADU AFORS-MIOC	18
5.2.1 Moduł cyfrowy	18
5.2.2 Płytką procesorowa MicroZed	19
5.2.3 Płytką BKO	21
5.2.4 Płytką DIGI	21
5.2.5 Przypisanie sygnałów układu Zynq.....	24
5.3 UKŁAD POMIAROWY UKŁADU AFORS-MIOC	24
5.3.1 Płytką BASE	24
5.3.2 Płytką AMP	27

5.3.3	Płytki FLT	29
5.3.4	Płytki ADC	31
5.3.5	Płytki LAS.....	33
5.3.6	Płytki MODGEN	35
5.3.7	Płytki modułu zasilania niskoszumnego STAB5.....	37
5.4	KONSTRUKCJA MECHANICZNA.....	38
5.5	PROGRAM DSP	39
5.6	TESTY UKŁADÓW ELEKTRONICZNYCH	51
5.6.1	Warunki standardowe przeprowadzania pomiaru.....	51
5.6.2	Badanie poziomu składowej stałej sygnału TIA w zależności od czasu.....	53
5.6.3	Badanie poziomu składowej stałej sygnału TIA w zależności od poziomu mocy lasera ..	54
5.6.4	Badanie wpływu poziomu sygnału modulacyjnego.....	55
5.6.5	Badanie wpływu doboru częstotliwości filtra LPF2	56
5.6.6	Badanie wpływu nasycenia wzmacniacza PGA2	59
5.7	UWAGI KOŃCOWE.....	64
6	DODATKI	65
6.1	OPROGRAMOWANIE FPGA VHD DLA OBSŁUGI AD7986	65
6.2	PROGRAM FPGA VHD DLA OBSŁUGI MBASE	69
6.3	PROGRAM GENERACJI SYGNAŁÓW ZEGAROWYCH.....	74
6.4	PROGRAM FPGA DLA OBSŁUGI GENERATORA MODULACJI	76
6.5	FOTOGRAFIE PROTOTYPU	81

Spis rysunków i tabel

Rys. 1. Schemat blokowy projektowanego układu	8
Tabela 1. Schematyczny układ MIOC	10
Tabela 2. Ogólne parametry układu MIOC.....	10
Rys. 2. Sygnał cyfrowej piły zsynchronizowanej z sygnałem prostokątnej modulacji	11
Rys. 3. Odpowiedź interferometru przy prostokątnej modulacji.....	12
Rys. 4. Przesunięcie fazy dla modulacji przebiegiem piłokształtnym.....	12
Rys. 5. Układ przetwarzania sygnałów	15
Rys. 6. Schemat blokowy układu zasilania PoE	16
Rys. 7. Układ teletransmisji i łączności	17
Rys. 8. Płytki procesorowa MicroZed firmy AVNET.....	19
Rys. 1. Płytki DIGI	22
Rys. 2. Schemat płytki DIGI.....	23
Rys. 3. Płytki MBASE.....	25
Rys. 4. Schemat płytki BASE	26
Rys. 5. Płytki AMP.....	27
Rys. 6. Schemat płytki AMP.....	28
Rys. 7. Widok płytki FLT.....	29
Rys. 8. Schemat płytki FLT	30
Rys. 9. Widok płytki ADC	31
Rys. 10. Schemat płytki ADC.....	32
Rys. 11. Widok płytki LAS	33
Rys. 12. Schemat płytki LAS.....	34
Rys. 13. Moduł układu generatora modulacji	36
Rys. 14. Widok płytki STAB5.....	37
Rys. 15. Schemat płytki STAB5	38
Rys. 16. Widok wizualizacji układu mechanicznego.....	38
Rys. 17. Obudowy przeznaczonej do zamontowania układu	39
Tabela 3. Zmiana poziomu wejściowego sygnału świetlnego w czasie	53
Tabela 4. Zmiana poziomu wejściowego sygnału świetlnego w zależności od mocy lasera	54
Tabela 5. Zmiana poziomu wejściowego sygnału świetlnego w zależności od poziomu modulacji	55
Tabela 6. Zmiana amplitudy sygnału na wyjściu wzmacniacza PGA2	59

Rys. 18. Widok na układ AMP z widocznym białym światłowodem dołączonym do diody detekcyjnej	63
Rys. 19. Widok na płytkę mikrokontrolera ZEND podłączoną do sieci lokalnej.....	64
Rys. 20. Widok prototypu gotowego do pracy	81
Rys. 21. Widok wnętrza prototypu	82
Rys. 22. Część elektroniczna z widoczną pętlą światłowodową	83
Rys. 23. Płytką mikrokomputera z układami towarzyszącymi	84

1 Wstęp

1.1 Cel projektu

Celem projektu jest opracowanie projektu technicznego zamkniętego układu detekcji dla czujników klasy AFORS na bazie zintegrowanego układu optycznego MIOC. W ramach zamówienia dostarczono specyfikację techniczną układu MIOC (Multi-Functional Integrated Optics Chip) jako aktywnego optycznego elementu planarnego zawierającego m.in. układ modulatora fazowego pozwalającego na szerokopasmowe sterowanie, co umożliwi realizację układu z tzw. śledzeniem zera na bazie układu z zamkniętą pętlą sprzężenia zwrotnego.

1.2 Oczekiwany wynik projektu

Oczekiwany wynik projektu jest przedstawienie opisu projektu technicznego układu detekcji wraz z dokumentacją i prototypowym egzemplarzem.

2 Wymagania

2.1 Wymagania układu optycznego

1. Opracowane rozwiązanie ma zapewnić działanie światłowodowego sejsmometru rotacyjnego w tzw. konfiguracji minimalnej, której konstrukcja nie jest elementem zamówienia.
2. Rozwiązanie dotyczy wykorzystania układu MIOC (Multi-Functional Integrated Optics Chip) według dostarczonej specyfikacji (Examining_Sheet_1240305_EN.pdf) jako elementu aktywnego pozwalającego na realizację układu z tzw. śledzeniem zera.

2.2 Wymagania układu elektronicznego

1. Wymagane jest rozwiązanie elektroniczne, które zostanie bezpośrednio zaimplementowane do układów klasy AFORS oraz FOSREM zarówno pod względem wykorzystania dostępnego zasilania prądowego jak i generacji sygnałów wyjściowych oraz współpracy z zewnętrznym system telemetrycznym (Internet).
2. Dolny zakres pomiarowy dla AFORS i FOSREM z opracowanym rozwiązaniem nie może być wyższy od połowy zakresu pomiarowego tych urządzeń dla określonych szerokości pasm detekcji (od 0.1 Hz do 300 Hz).
3. Planowany teoretycznie górny zakres pomiarowy powinien wynosić 10rad/s dla pasma od 0.1 Hz do 100 Hz.

2.3 Wymagania układu transmisji danych

Opracowany układ ma zapewniać możliwość wysyłania danych pomiarowych jak również możliwość konfiguracji parametrów wewnętrznych poprzez sieć Internet z wykorzystaniem protokołu HTTP/HTTPS.

1. Należy zapewnić możliwość poprawnej pracy w sieciach publicznych GSM/3G/LTE.
2. Układ powinien mieć możliwość sterowania i kontroli za pośrednictwem przeglądarki internetowej.

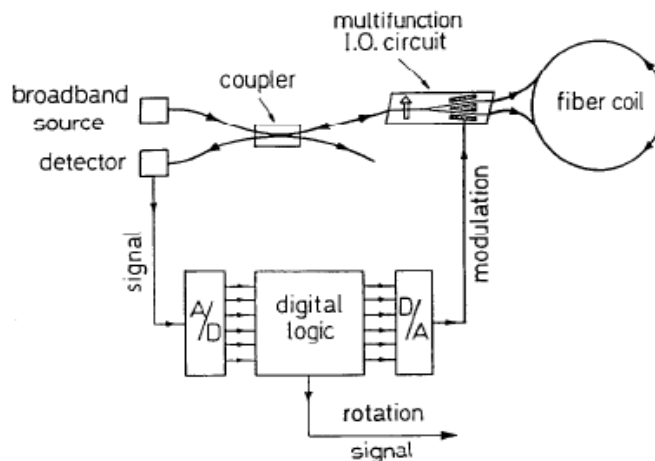
2.4 Wymagania organizacyjne

1. Wymagane jest zaprojektowanie i wykonanie 1 egzemplarza prototypu zamkniętego układu detekcyjnego.
2. Realizacja zamówienia bazuje na zawartej umowie o współpracy naukowo-badawczej pomiędzy Zamawiającym i WAT.

3 Koncepcja

3.1 Podstawowa koncepcja układu pomiarowego

Na podstawie wymagań określono podstawową koncepcję układu detekcyjnego pokazaną na schemacie blokowym rys. 1.



Rys. 1. Schemat blokowy projektowanego układu

Układ składa się z następujących bloków funkcjonalnych:

1. Układ optyczny składający się z szerokopasmowego źródła optycznego (broadband source), sprzęgacza optycznego (coupler), układu MIOC (multifunctional I.O. circuit), pętli światłowodowej (fiber coil) oraz detektora (detector),
2. Układ wzmacniacza sygnału z detektora (detector),
3. Przetwornik Analogowo-Cyfrowy (A/D),
4. Cyfrowy Układ Logiczny (digital logic),
5. Przetwornik Cyfrowo-Analogowy pozwalający na sterowanie układem MIOC (D/A)
6. Układ przesyłania informacji o prędkości obrotowej (rotational signal).

Zaprojektowany układ jest przystosowany do pracy z układem optycznym o dwu zasadniczo różnych parametrach funkcjonalnych:

- a. układem z pętlą o długości ok. 15.000 m – wówczas na detektorze spodziewany jest sygnał o mocy optycznej $P=80 \mu W$, co dla czułości zastosowanej fotodiody $S=0.90 A/W$ daje prąd o natężeniu $I=70 \mu A$, zaś częstotliwość własna układu, czyli częstotliwość modulacji MIOC sygnałem prostokątnym wynosi około 6.800 Hz. W tym wypadku układ powinien zapewniać detekcję prędkości obrotowej z zakresu 10^{-8} rad/s do 10^{-6} rad/s w paśmie częstotliwości od 0.1 Hz do 100 Hz.
- b. układem z pętlą o długości ok. 5.000 m – wówczas na detektorze spodziewany jest sygnał o mocy optycznej $P=220 \mu W$, co dla czułości zastosowanej fotodiody $S=0.90 A/W$ daje prąd o natężeniu $I=200 \mu A$, zaś częstotliwość własna układu, czyli częstotliwość modulacji MIOC sygnałem prostokątnym wynosi około 20.450 Hz. W tym wypadku układ powinien zapewniać detekcję prędkości obrotowej z zakresu 10^{-6} rad/s do 10 rad/s w paśmie częstotliwości od 0.1 Hz do 100 Hz.

W obu wypadkach układ ma mieć możliwość regulacji częstotliwości pracy w zakresie ± 300 Hz wokół wyżej wymienionych częstotliwości własnych, w celu doboru optymalnych parametrów pracy dla konkretnego układu. Natomiast zakres napięciowy generowanych przebiegów powinien mieścić się w przedziale ± 10 V.

Zaprojektowany układ jest przystosowany do współpracy z układem MIOC schematycznie przedstawionym poniżej (dane katalogowe), którego główne parametry podano w Tabeli 1 oraz Tabeli 2.

Tabela 1. Schemat budowy i wymiarowania układu MIOC

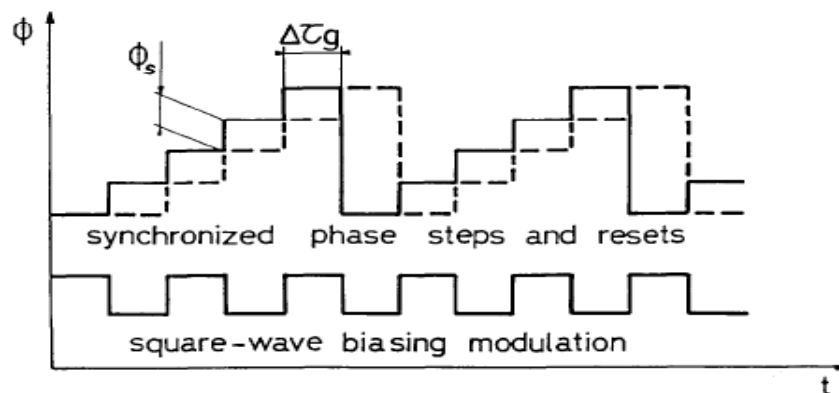
Tabela 2. Ogólne parametry układu MIOC

Strona 10 z 84

3.2 Koncepcja układu detekcji

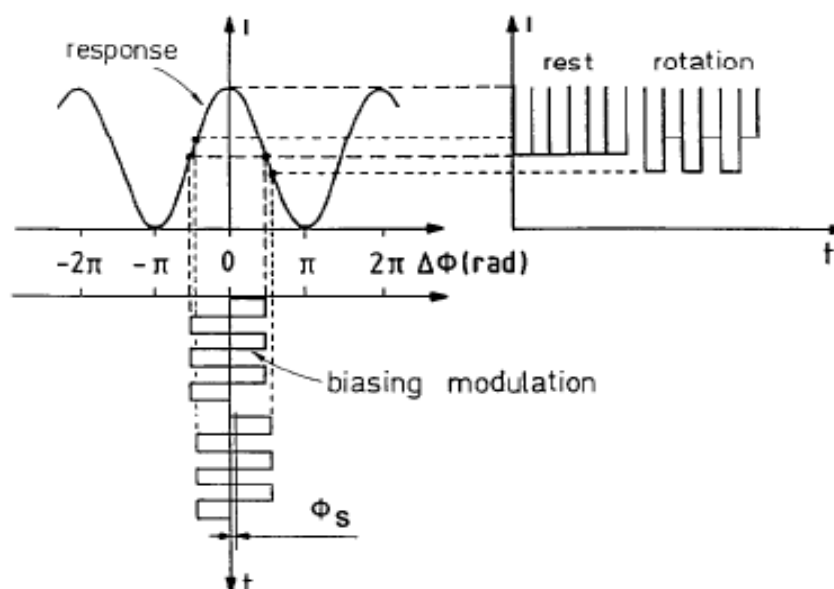
Istotą rozwiązania jest zastosowanie cyfrowej pętli sprzężenia zwrotnego, której zadaniem jest utrzymywanie interferometru w położeniu zerowym (zerowanie przesunięcia fazy wprowadzanego przez efekt Sagnaca). Takie rozwiązanie pozwala zapewnić liniowość skali oraz możliwość rozszerzenia zakresu pomiarowego.

Dlatego też do sterowania modulatorem fazy występującym w MIOC przyjęto koncepcję zastosowania złożonego sygnału: piły i prostokąta, co przedstawiono schematycznie na rys. 2. Sygnał ten powinien charakteryzować się amplitudą równą 2π lub wielokrotnością 2π . Czas trwania schodków równa się czasowi przebiegu fali optycznej przez pętlę czujnikową $\Delta\tau_g = cL/n$, gdzie c – prędkość światła w próżni, L – długość pętli, n – współczynnik załamania dla włókna. Natomiast wysokość schodków jest równy wprowadzonemu przesunięciu fazy Sagnaca $\phi_s = 4(\pi RL/\lambda c)\Omega$, gdzie R – promień nawinięcia pętli, λ - długość fali światła, zaś Ω - prędkość kątowa obrotu. Takie rozwiązanie pozwala na wyzerowanie czujnika. Szukaną wielkość ϕ_s otrzymujemy wówczas bezpośrednio z cyfrowego układu pętli sprzężenia zwrotnego, która jest wprost proporcjonalna do Ω i tą drogą podlega wyliczeniu w cyfrowym układzie logicznym.



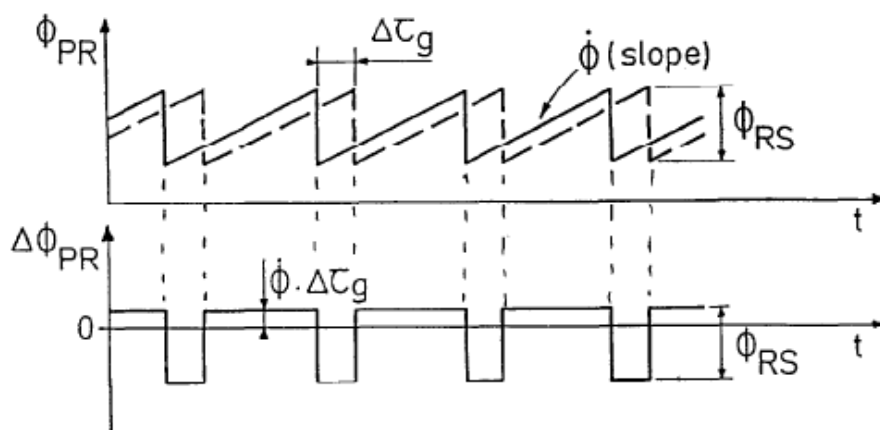
Rys. 2. Sygnał cyfrowej piły zsynchronizowanej z sygnałem prostokątnej modulacji

Dla poglądowego wyjaśnienia wymaganego rozwiązania na rys. 3 przedstawiono spodziewaną odpowiedź interferometru dla modulacji prostokątnej o częstotliwości $f=1/2\Delta\tau_g$. W przypadku zerowego przesunięcia fazy (braku prędkości obrotowej Ω) odpowiedź interferometru składa się z wąskich szpilek, w miejscach przejścia sygnału modulującego przez zera oraz charakteryzuje się pewną składową stałą. Wprowadzenie nieodwracalnego przesunięcia Sagnaca (występowanie prędkości obrotowej Ω) powoduje zmodyfikowanie odpowiedzi interferometru w przybliżeniu do sygnału prostokątnego plus niekorzystne szpilki. Stąd zastosowanie odpowiednich metod demodulacji pozwalać będzie na odczytanie z takiego sygnału prędkości obrotowej, która wywołała dane przesunięcie fazy.



Rys. 3. Odpowiedź interferometru przy prostokątnej modulacji

Sugerowana i sprawdzona metoda demodulacji pokazana jest na rys. 4, gdzie poprzez wprowadzanie piłokształtnego przebiegu napięcia sterującego modulator fazy w MIOC, uzyskuje się przesunięcie fazy (pomiędzy dwoma przeciwnie propagującymi się w pętli czujnika promieniami światła) uzależnione od nachylenia przebiegu piłokształtnego oraz czasu $\Delta\tau_g$. Modyfikując nachylenie przebiegu piłokształtnego (zmiana napięcia zasilającego modulator) regulujemy wprowadzanie przesunięcie fazy, kompensując wielkość efektu Sagnaca. Tym sposobem sygnał sterujący modulator tak by nastąpiła kompensacja zmiany fazy (utrzymujący go w położeniu zerowym) jest sygnałem o szukanej wielkości fazy Sagnaca, co po odpowiedniej kalibracji powinno dawać sygnał o rejestrowanej prędkości obrotowej.



Rys. 4. Przesunięcie fazy dla modulacji przebiegiem piłokształtnym

Projektowany układ elektroniczny stosowany do obróbki i przetwarzania sygnałów musi zapewnić dynamikę pozwalającą na zapewnienie wymaganej dokładności w całym zakładanym zakresie prędkości obrotowej. W ramach układu logicznego powinien być zastosowany szybki układ FPGA zaprogramowany do generowania wymaganych sygnałów i przetwarzania ADC sygnałów analogowych z wymaganą częstotliwością i rozdzielczością.

Układ elektroniczny powinien zawierać również kompletny zestaw modułów pozwalających na realizację zadań wymaganych w pracy autonomicznej i grupowej z synchronizacją czasu zegarów wewnętrznych w tym:

1. Układ przetwarzania wyliczonego sygnału prędkości obrotowej, kierunku obrotu, zakresu pomiarowego do znormalizowanych wartości cyfrowych.
2. Układ pamięciowy pozwalający na zarejestrowanie i zapamiętanie do 30 dni ciągłych pomiarów z maksymalną częstotliwością z możliwością odczytu asynchronicznego w czasie rejestracji.
3. Układ sterujący pozwalający na obsługę wszystkich modułów wewnętrznych, kontrolę ich parametrów.
4. Układ pomiaru parametrów pracy urządzenia: temperatury wewnętrznej i zewnętrznej, przyspieszenia, położenia względem ziemi, prędkości obrotowej obudowy, położenia względem bieguna geograficznego, położenia geograficznego miejsca instalacji na bazie GPS/Galileo/Glonass. Możliwe jest uzyskiwanie położenia z zewnętrznej anteny GPS.
5. Układ powinien umożliwiać także synchronizację wewnętrznych zegarów czasu rzeczywistego w czasie pracy z jednostką centralną oraz z zewnętrznymi serwerami NTP.

3.3 Koncepcja zasilania i teletransmisji

Układ sterujący stosowany w układzie musi umożliwiać dostęp do parametrów oraz możliwość ustawiania ich poprzez interfejs typu WEB. Urządzenie powinno mieć wbudowany serwer WEB pozwalający na sterowanie i wizualizację aktualnych danych pomiarowych w szczególności zawierać:

1. Układ transmisji danych pozwalający na dołączenie urządzenia do jednostki centralnej w trybie pracy grupowej przy pomocy łącza min. 1000Mbps Gigabit Ethernet lub łącza światłowodowego.
2. Urządzenie powinno pozwalać na automatyczne zestawianie połączenia VPN z centralnym serwerem telemetrycznym wraz z automatycznym ponownym zestawianiem tunelu po przerwach w transmisji danych.
3. Urządzenie powinno zapewniać możliwość korzystania z sieci bezprzewodowych WWAN/WiFi w szczególności obsługiwać dostęp do sieci Internet poprzez sieci 3G/HDSPA oraz LTE.

4. Układ zasilania z wewnętrznym układem podtrzymania napięcia wystarczającym na min. 2 godziny pracy bez zasilania zewnętrznego, układem zasilania PoE o wymaganej przez układ pomiarowy i elektroniczny mocy, układem zasilania z zewnętrznego źródła +12 - 24V.
5. Możliwość połączenia urządzenia do jednostki centralnej na odległość do 50 m przy pomocy przewodu FTP kat 6/7 oraz do 1000 m przy pomocy światłowodu.

Proponowane jest aby w interfejsie webowym znajdował się moduł pozwalający w czasie rzeczywistym obserwować bieżące wartości pomiarowe w trybie oscyloskopowym oraz parametry układu w postaci cyfrowej i graficznej.

Korzystnie jest aby układ sterująco-konwertujący umożliwiał przesyłanie aktualnych parametrów, danych pomiarowych, występujących zdarzeń, poprzez sieć Internet do centralnego serwera telemetrycznego. Transmisja musi być wykonywana z wykorzystaniem protokołu HTTP/HTTPS z autentykacją poszczególnych urządzeń pomiarowych. Transmisja musi być niezależna od topologii dostępnej sieci IP i nie może wymagać przydzielania publicznych adresów IP po stronie urządzenia.

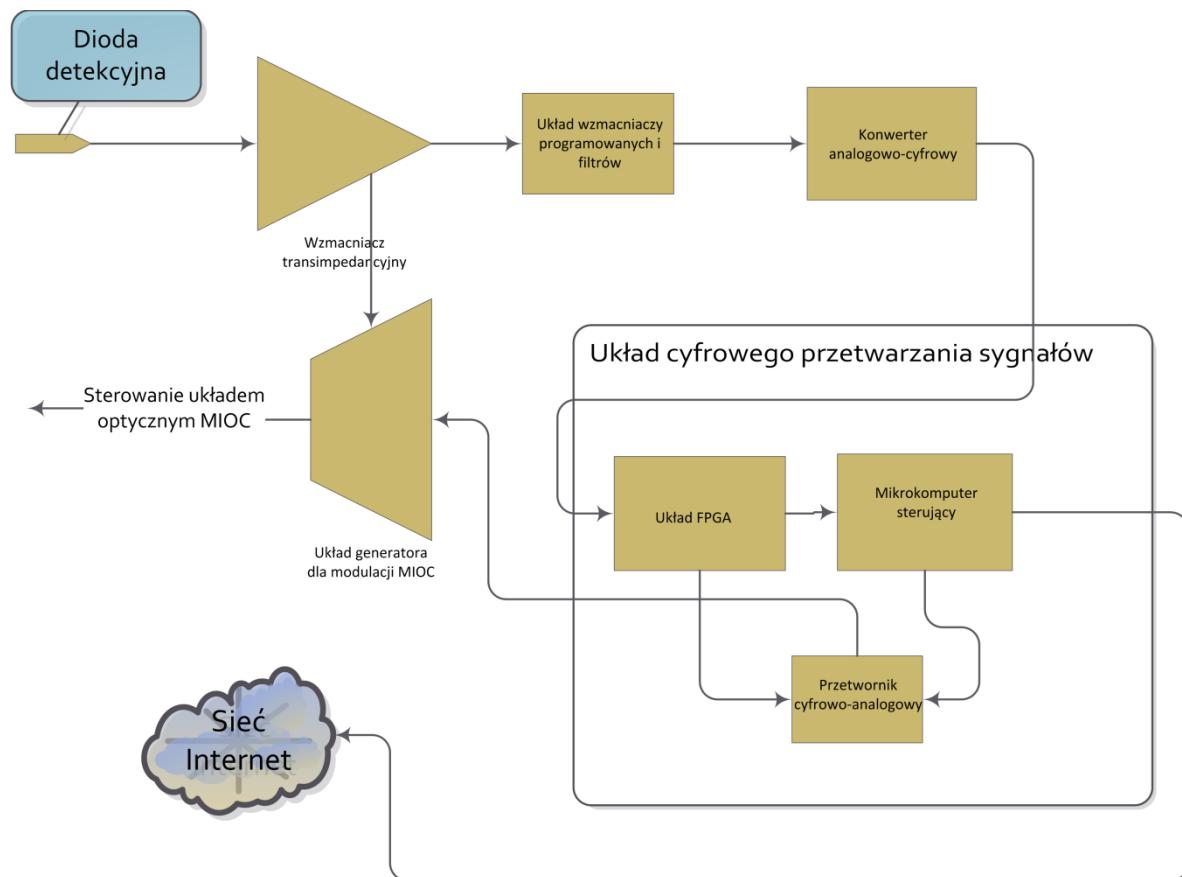
4 Wymagania konstrukcyjne

4.1 Wymagania dla układu pomiarowego

Układ pomiarowy według przedstawionej koncepcji (rys. 5) będzie składał się z sensora detekcyjnego, dołączonego do niego wysokoczułego, niskoszumnego wzmacniacza o regulowanym wzmacnieniu wraz z cyfrowymi filtrami aktywnymi pozwalającymi na regulację parametrów.

Sygnał wyjściowy z układu wzmacniająco-filtrującego jest przekazywany na wejście szybkiego przetwornika analogowo-cyfrowego o dużej szybkości działania. Cyfrowy sygnał wyjściowy jest przekazywany do układu logicznego obliczającego wymagane parametry sterujące dla modulatora wbudowanego w MIOC. Proponuje się, że układ logiczny zostanie zbudowany na bazie szybkiego układu FPGA w którym zostaną zaprojektowane wymagane moduły logiczne, obliczeniowe i realizujące przekształcenia FFT na poziomie sprzętowym.

Sygnał cyfrowy z układu logicznego jest przekazywany do przetwornika cyfrowo-analogowego, a wyjściowe napięcie po odpowiednim wzmacnieniu steruje modulatorem generującym przebieg piłokształtny dla układu MIOC.



Rys. 5. Układ przetwarzania sygnałów

Jednocześnie obliczany jest sygnał będący odwzorowaniem prędkości obrotowej. Ten sygnał po obróbce cyfrowej jest przekazywany do mikrokomputera sterującego gdzie jest zapisywany w pamięci masowej i jednocześnie używany do wizualizacji sygnału wynikowego.

W układzie mikrokomputerowym realizowana jest także funkcja detekcji zdarzeń i wysyłania odpowiednich informacji do serwera telemetrycznego.

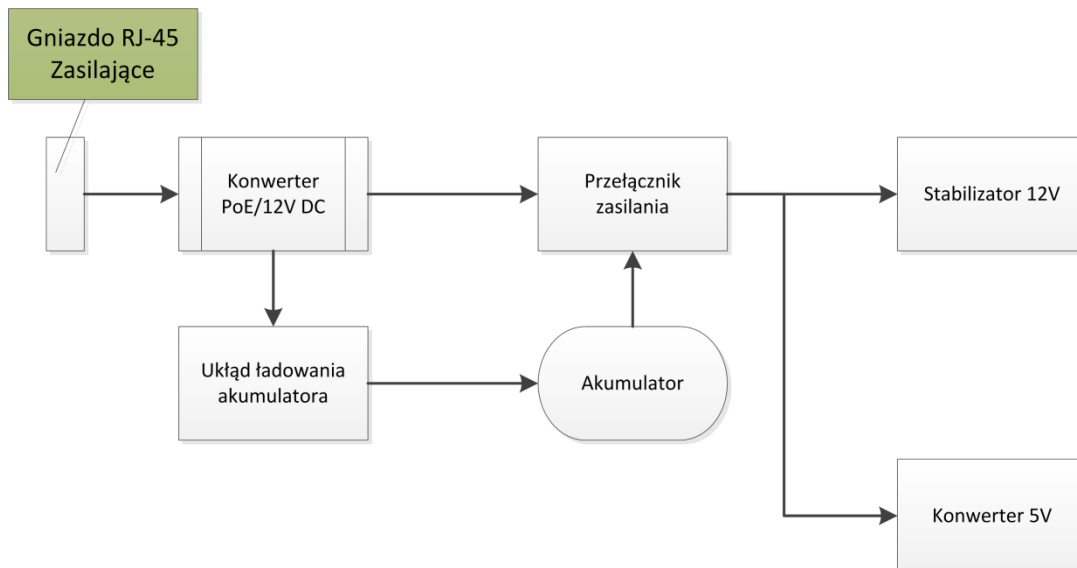
4.2 Wymagania dla układu zasilania

Cały układ elektroniczny zasilany jest z pojedynczego napięcia 12V z wbudowanym specjalizowanym, akumulatorowym układem podtrzymania napięcia. Układ ten wykorzystuje akumulatory Li-On zapewniające pracę bez zasilania zewnętrznego przez min. 2 godziny. Przyjęto że całkowita moc pobierana przez cały układ elektroniczny oraz układ zasilania laserem nie przekroczy 15W. Dlatego wymagana pojemność akumulatora wewnętrznego została określona na 30Wh co przy napięciu akumulatora 7.2V (dwa ogniwa Li-On) daje pojemność na poziomie 4,2 Ah. Jednak przy założeniu całkowitej sprawności przetwornic stosowanych w układach zasilania na 80%,

wymagana pojemność akumulatora wynosi ok 5,2 Ah. W związku z tym można zastosować akumulator o pojemności powyżej tej wartości dostępny na rynku.

Przyjęto, że zasilanie zewnętrzne będzie odbywało się poprzez Power over Ethernet – przy pomocy tego samego przewodu którym układ będzie podłączony do sieci Internet. Dostępne są gotowe rozwiązania zasilaczy i konwerterów pozwalających na uzyskanie w takim układzie wymaganej mocy.

Poniżej na rys. 6 przedstawiono schemat blokowy układu zasilania:



Rys. 6. Schemat blokowy układu zasilania PoE

4.3 Wymagania dla układu teletransmisji

Układ teletransmisji został zaprojektowany do współpracy z siecią Internet. Wbudowany w mikrokomputer interfejs Ethernet 1000Mbps pozwoli na szybkie i wydajne przesyłanie danych pomiarowych w czasie rzeczywistym.

Wbudowany system operacyjny Linux wraz z serwerem WEB – Apache i serwerem baz danych MySQL pozwala na uruchomienie serwera www realizującego konsolę operatorską, wizualizację wyników pomiarowych i monitorowanie oraz sterowanie całym układem.

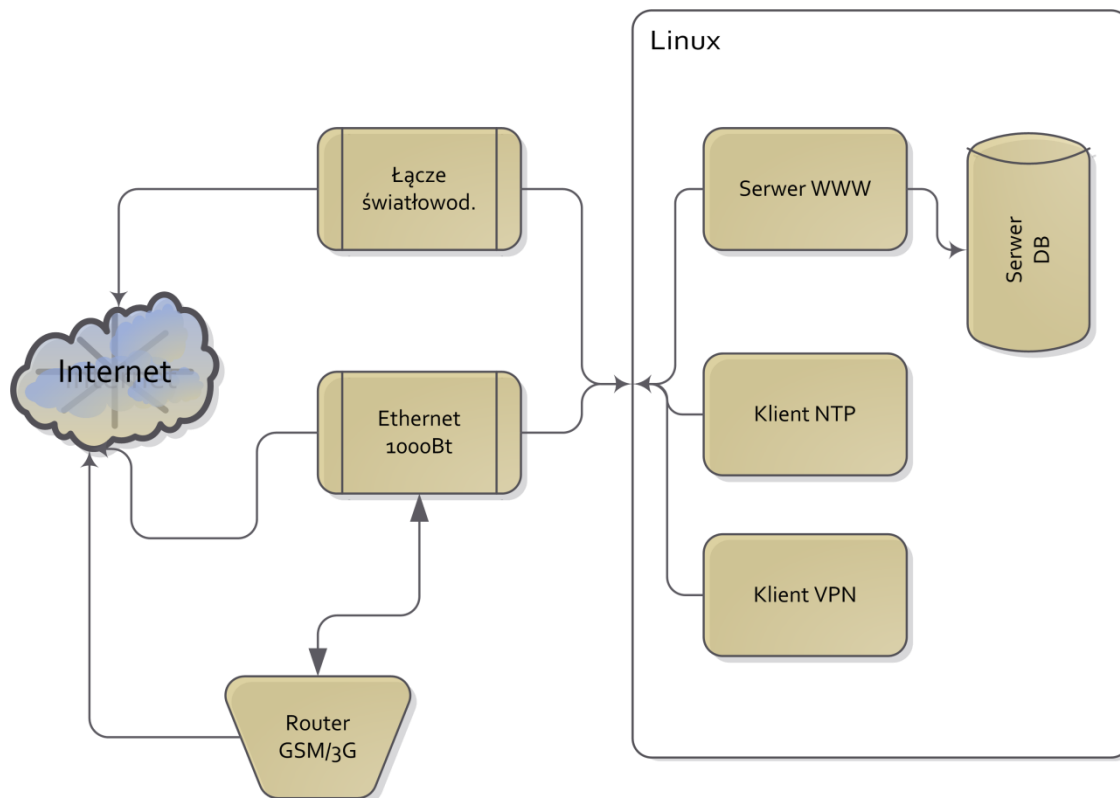
Opcjonalnie można dołączyć konwerter światłowodowy pozwalający na realizację łącza Ethernet przy pomocy światłowodu nawet do długości 2000m. W takim przypadku wymagane jest podłączenie do złącza Ethernet zasilacza PoE.

W ramach tworzenia lokalnej sieci, która może składać się z jednego lub większej liczby układów pomiarowych, możliwe jest dołączenie routera GSM/3G/LTE realizującego dostęp do aktualnie dostępnej sieci publicznej.

Każdy mikrokomputer wewnętrzny zapewnia funkcjonalność klienta VPN co pozwala na stworzenie rozległej bezpiecznej sieci czujników.

Możliwe jest uruchomienie lokalnego serwera telemetrycznego zapewniającego zasilanie dla wielu układów pomiarowych, przechowywanie danych przez okres dłuższy niż możliwy w wewnętrznej pamięci czujników, synchronizację czasu oraz routing do dostępnej sieci telekomunikacyjnej. W ten sposób można zrealizować sieć czujników w lokalnym miejscu, które będą pracowały z zsynchronizowanymi zegarami.

Schemat układu teletransmisji umieszczono poniżej na rys. 7:



Rys. 7. Układ teletransmisji i łączności

5 Projekt techniczny układu detekcji AFORS-MIOC

5.1 Budowa układu AFORS-MIOC

Układ powstał na bazie doświadczeń związanych z prowadzonymi wcześniej podobnymi układami pomiarowymi. Zastosowanie nowoczesnego układu MIOC spowodowało potrzebę odpowiedniej modyfikacji konstrukcji. Wprowadzono także nowe, wydajniejsze rozwiązania związane głównie pojawieniem się nowych układów scalonych typu SOC. Integrują one w jednej obudowie dwurdzeniowy procesor ARM Cortex A8 oraz duży układ FPGA. Umożliwia to realizację wymaganych systemów komunikacji między układowej oraz umożliwia elastyczne przypisywanie funkcji i sygnałów do wyprowadzeń układu scalonego. Dodatkowo powoduje zredukowanie całkowitego kosztu układów elektronicznych niezbędnych do realizacji zaprojektowanego układu.

5.2 Układ cyfrowy układu AFORS-MIOC

5.2.1 Moduł cyfrowy

Moduł cyfrowy obsługuje połączenia z innymi modułami systemu oraz komunikację z komputerem PC.

Moduł cyfrowy jest zrealizowany z zastosowaniem trzech układów:

- Płytki procesora MicroZed firmy AVNET
- Płytki bazowej BKO firmy AVNET
- Płytki drukowanej DIGIT

Płytką procesora MicroZed zawiera:

- Interfejs cyfrowy USB do komunikacji z komputerem PC
- Interfejs sieci Internet ze złączem RJ-45
- Interfejs USB
- Gniazdo karty SD

Płytką BKO zawiera:

- Gniazdko do założenia płytki procesora MicroZed
- Układ zasilania I/O dla układu MicroZed
- Interfejs do sygnałów cyfrowych wyprowadzonych na dwa złącza IDC

Płytką DIGIT zawiera:

- Gniazdko do założenia płytki BKO
- Gniazdko zasilania
- Gniazdko standardu MM dla dołączania innych modułów

Płytką MicroZed jest dołączona poprzez złącza szerokopasmowe do płytki drukowanej BKO. Płytką BKO jest dołączana poprzez złącza IDC do płytki DIGIT.

5.2.2 Płytką procesorową MicroZed

Płytką procesora jest zrealizowana w postaci gotowej płytki drukowanej zestawu ewaluacyjnego MicroZed firmy AVNET. Płytką MicroZed zawiera układ FPGA serii Zynq firmy Xilinx. Jest to szybki układ programowalny z dwurdzeniowym sprzętowym procesorem ARM Cortex A-9.

Został zastosowany procesor wbudowany ARM9 oraz układ FPGA firmy Xilinx.

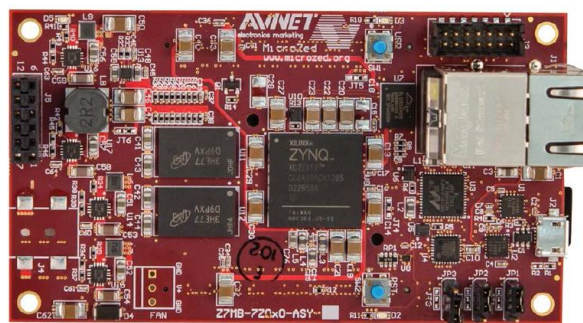
Moduł procesora wykonuje następujące działania:

- Obsługa bieżąca strumieni sygnałów i ich zapisywanie,
- Sterowanie parametrami wzmacniania, filtrowania oraz przetwarzania.

Moduł elektroniczny jest wyposażony interfejs cyfrowy USB. Interfejs USB jest przeznaczony do współpracy z komputerem PC. Posiada również interfejs sieci Internet 10/100.

Zastosowanie izolacji galwanicznej pomiędzy modułem procesora oraz innymi modułami zapewnia rozdzielanie masy analogowej modułów od zaszumionej masy cyfrowej procesora. Można też uniknąć pętli sygnałów masy pomiędzy modułami w przypadku dołączania zewnętrznych przyrządów pomiarowych takich jak oscyloskop czy woltomierz.

Płytką drukowaną zestawu ewaluacyjnego MicroZed firmy AVNET ma wymiary 2.25" x 4.00" (57.15 mm x 101.6 mm) oraz wysokość 25.5mm przedstawiona jest na rys. 8.

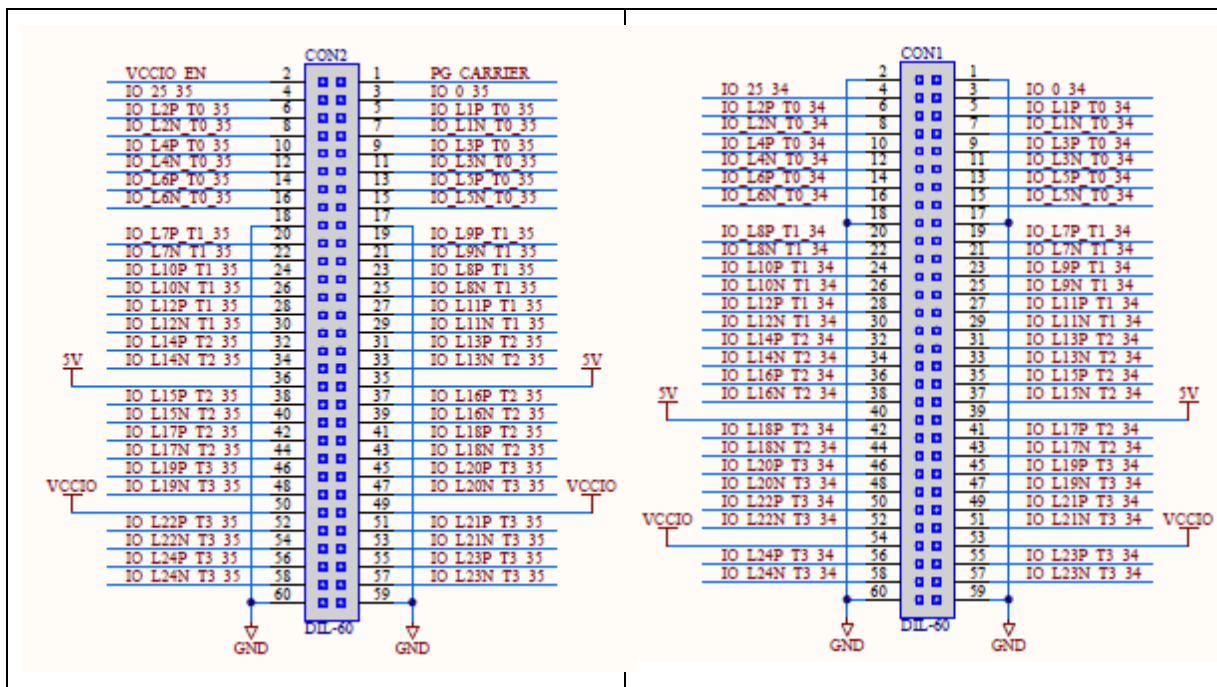


Rys. 8. Płytką procesorową MicroZed firmy AVNET

Płytki ma cztery otwory montażowe o średnicy 0.125" (3.175mm). Standardowa wysokość do płytki bazowej to 5mm, przy montażu ze słupkami M3x5.

Płytki zawiera dwa złącza 0.8mm BergStak® 100-position Dual Row, BTB Vertical Receptacles (61082-101400LF) firmy FCI (3.7mm). Gniazdka połączeń na płycie bazowej są typu 0.8 mm BergStak® 100-position Dual Row BTB Vertical Plugs (61083-101400LF) firmy FCI (4.7mm). Razem złącza wymagają odstępu płytek równego 5mm. Zalecenia konstruowania płytki bazowej są zamieszczone w dokumencie MicroZed™ Carrier Design Guide.

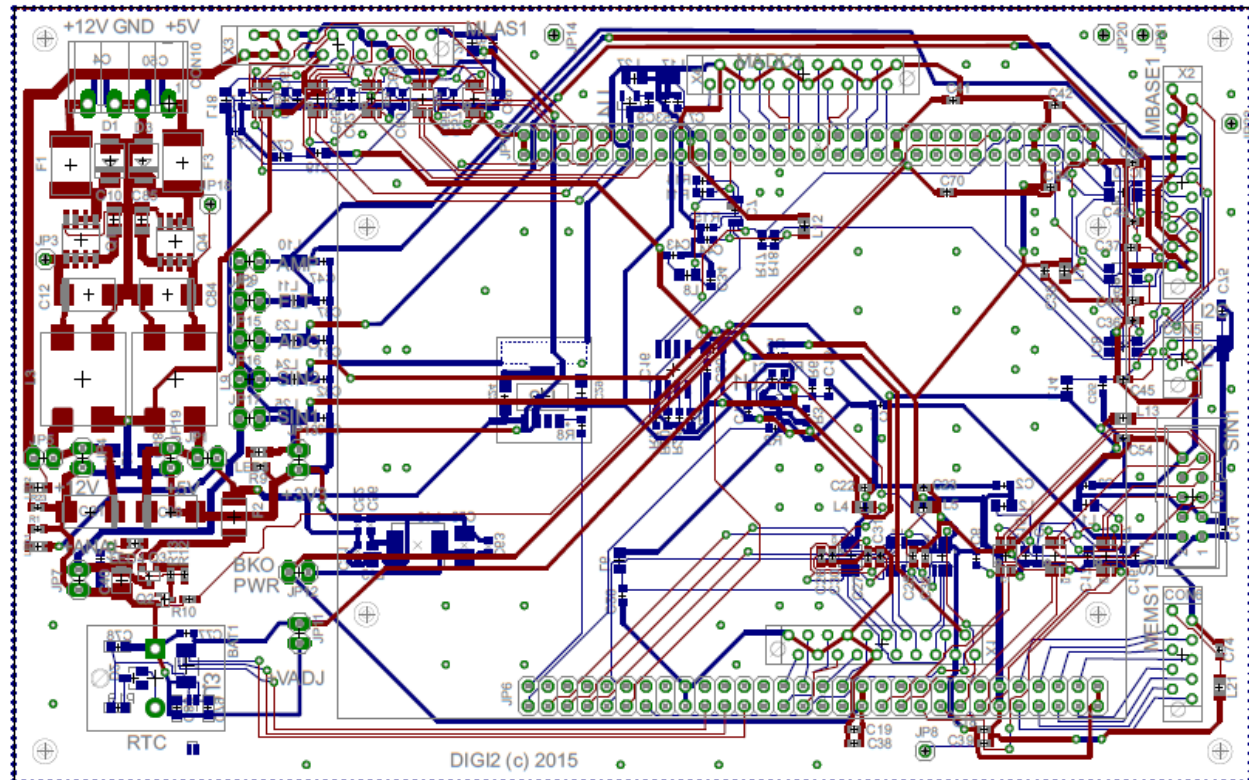
5.2.3 Płytki BKO



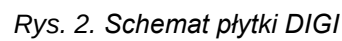
Płytki BKO jest przeznaczona do zapewnienia odpowiednich napięć zasilania oraz do wyprowadzenia sygnałów z płytki MICRO-ZEND do złącz sygnałowych.

5.2.4 Płytki DIGI

Do płytki DIGI dołączana jest płytki BKO z zamontowaną na niej płytką MicroZed. Służą do tego dwa 40-to pinowe złącza IDC. Zasilanie jest dostarczane przez poczwórne złącze zaciskowe. Gniazdo IDC-10 służy do dołączania modułu MODGEN. Pozostałe złącza standardu MM służy do dołączania modułu BASE, ADC i LASER.



Rys. 1. Płytki DIGI



5.2.5 Przypisanie sygnałów układu Zynq

Płytką MicroZed firmy Avnet zawiera układ procesorowy Xilinx XC7Z010-1CLG400C.

Większość wyprowadzeń, które dostarcza MIO Bank1 została na stałe przypisana do portów peryferyjnych płytki MicroZed: port SDIO0 do karty microSD (poprzez translator poziomu logicznego 1.8V do 3.3V), port USB0 (USB 2.0 Host/Device) do gniazdka USB typ A, port UART1 do gniazdka USB microAB (poprzez USB2,0 Full-Speed USB-UART bridge), nóżka MIO[51] do przycisku, nóżka MIO[47] do diody LED, port ETHERNET 0 (10/100/1000) do gniazdka RJ-45.

Pozostałe wyprowadzenia, które dostarcza MIO Bank0, czyli MIO[0] oraz MIO[9-15] są dołączone do gniazdka standardu Digilent Pmod (IDC12) Sygnały te są również dostępne na takim samym gnieździe na płycie NBCC-FMC.

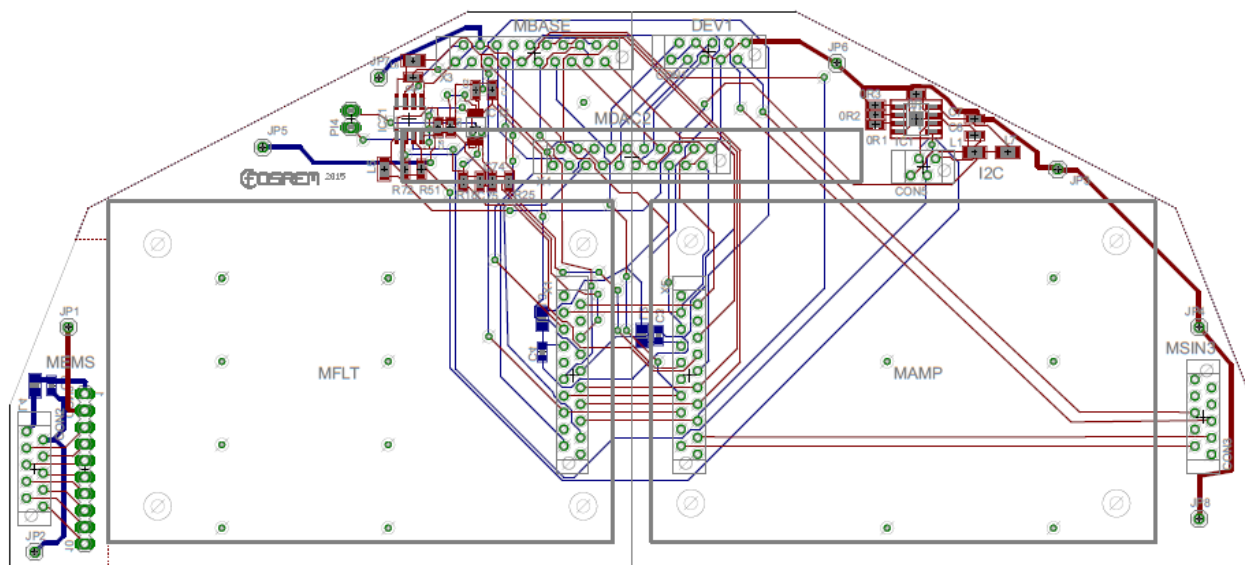
Porty SPI0, SPI1, UART1, I2C0, I2C1, CAN0, CAN1 mogą być dołączone (poprzez wewnętrzny multiplexer EMIO) do wyprowadzeń dołączonych bezpośrednio do struktury FPGA układu Zynq.

Przypisanie pozostałych sygnałów systemu do wyprowadzeń układu FPGA jest opisane w dodatku.

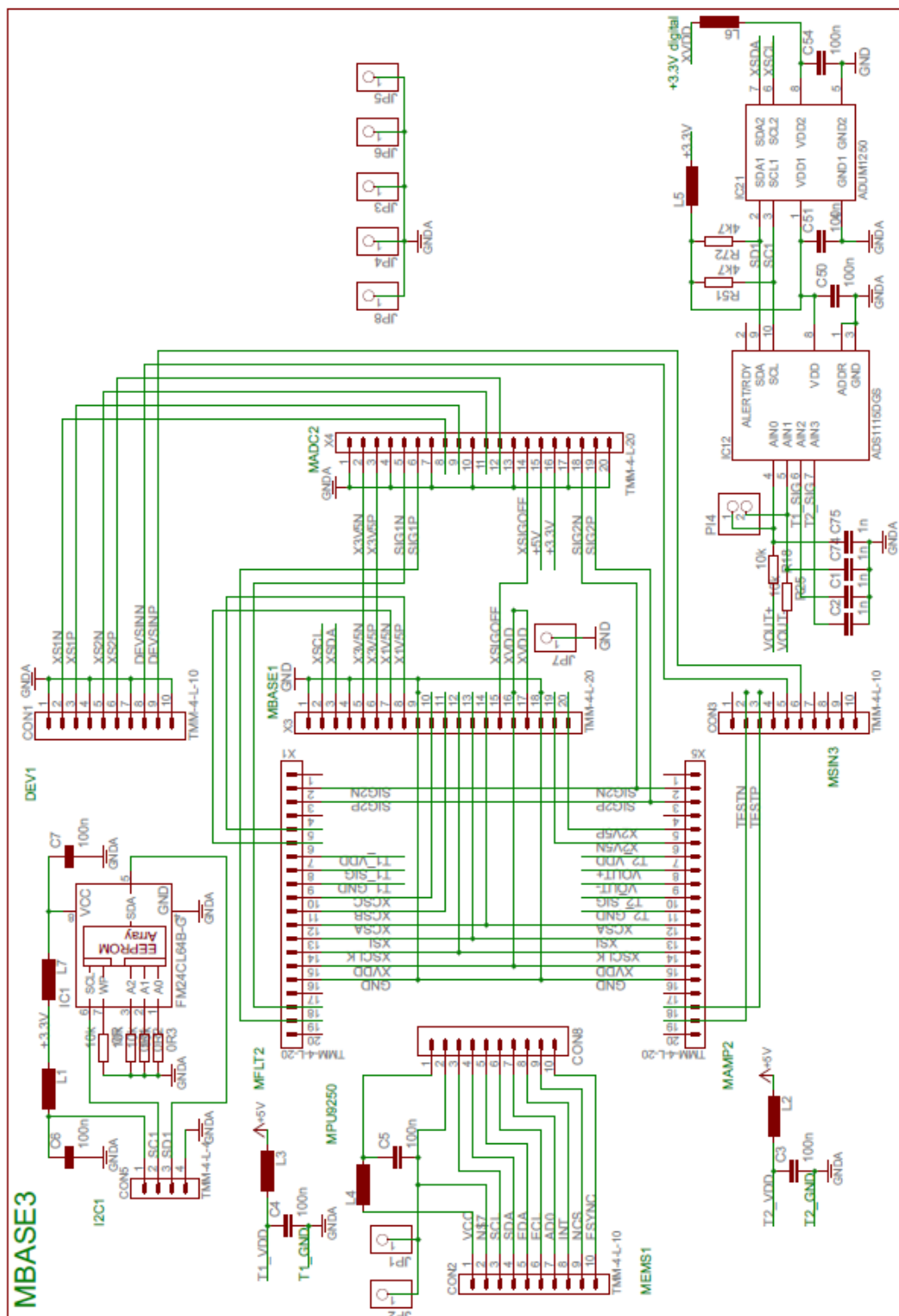
5.3 Układ pomiarowy układu AFORS-MIOC

5.3.1 Płytką BASE

Płytką BASE służy do dołączenia płytek modułów MAMP, MFLT, MADC oraz modułu MEMS. Złącza standardu MM służą do dołączenia modułu MODGEN oraz wyprowadzenia sygnałów pomiarowych potrzebnych do kalibracji. Moduł MEMS typu MPU-9250 dołączony jest do złącza typu „goldpin 2.54 mm”.



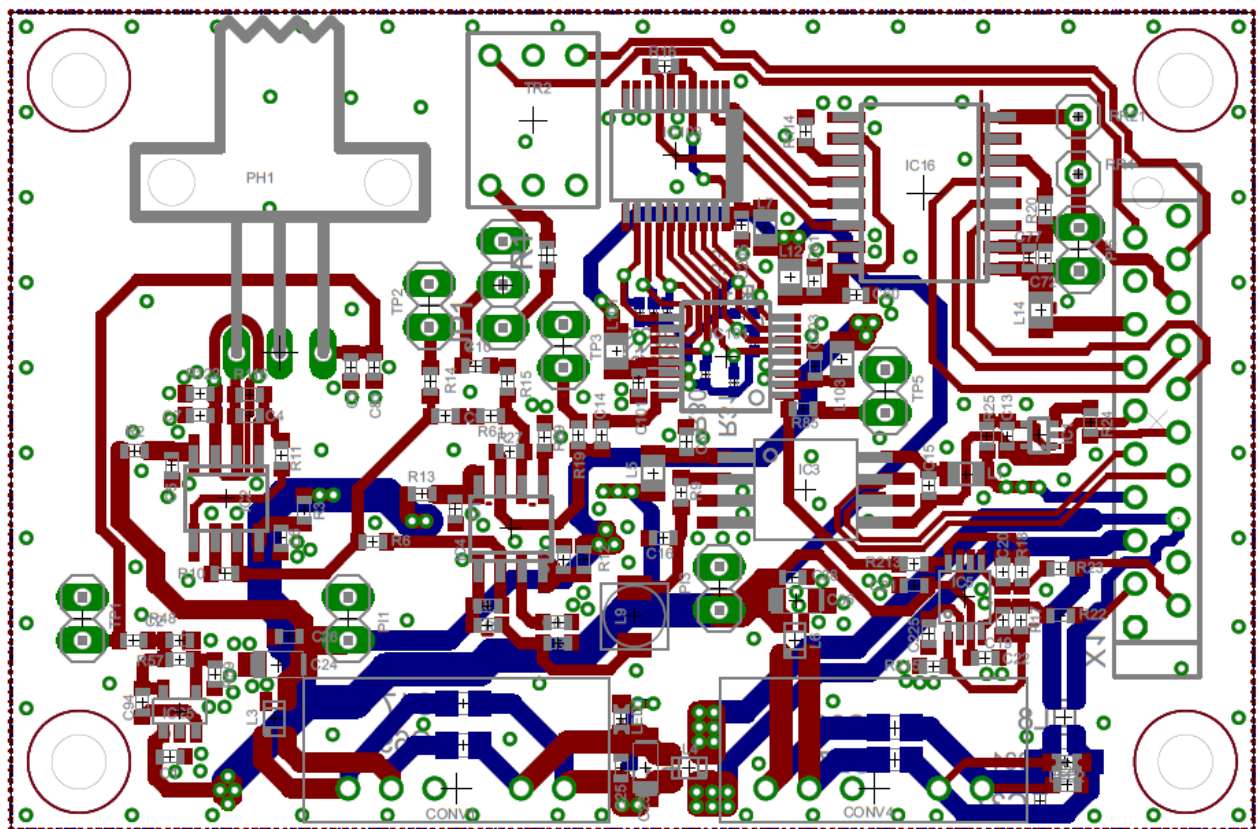
Rys. 3. Płytki MBASE



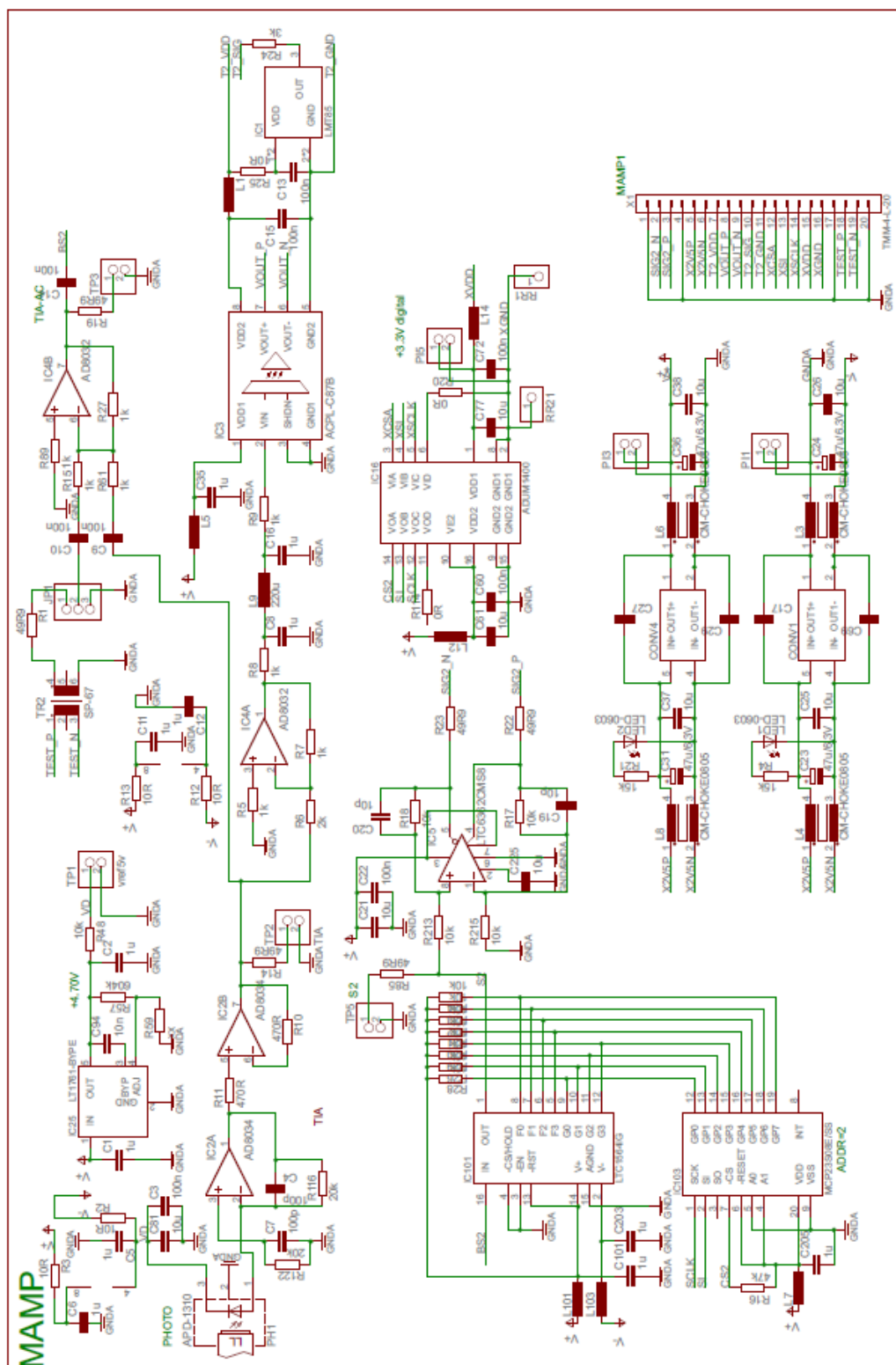
Rys. 4. Schemat płytki BASE

5.3.2 Płytki AMP

Moduł MAMP jest dołączany bezpośrednio do modułu BASE poprzez złącze standardu MM20. Płytki modułu jest umieszczona w metalowym pudełku ekranującym typu PERANCEA CFL2T (75X50X25mm). Na płytce są umieszczone wsporniki do mocowania fotodiody dołączonej do światłowodu.



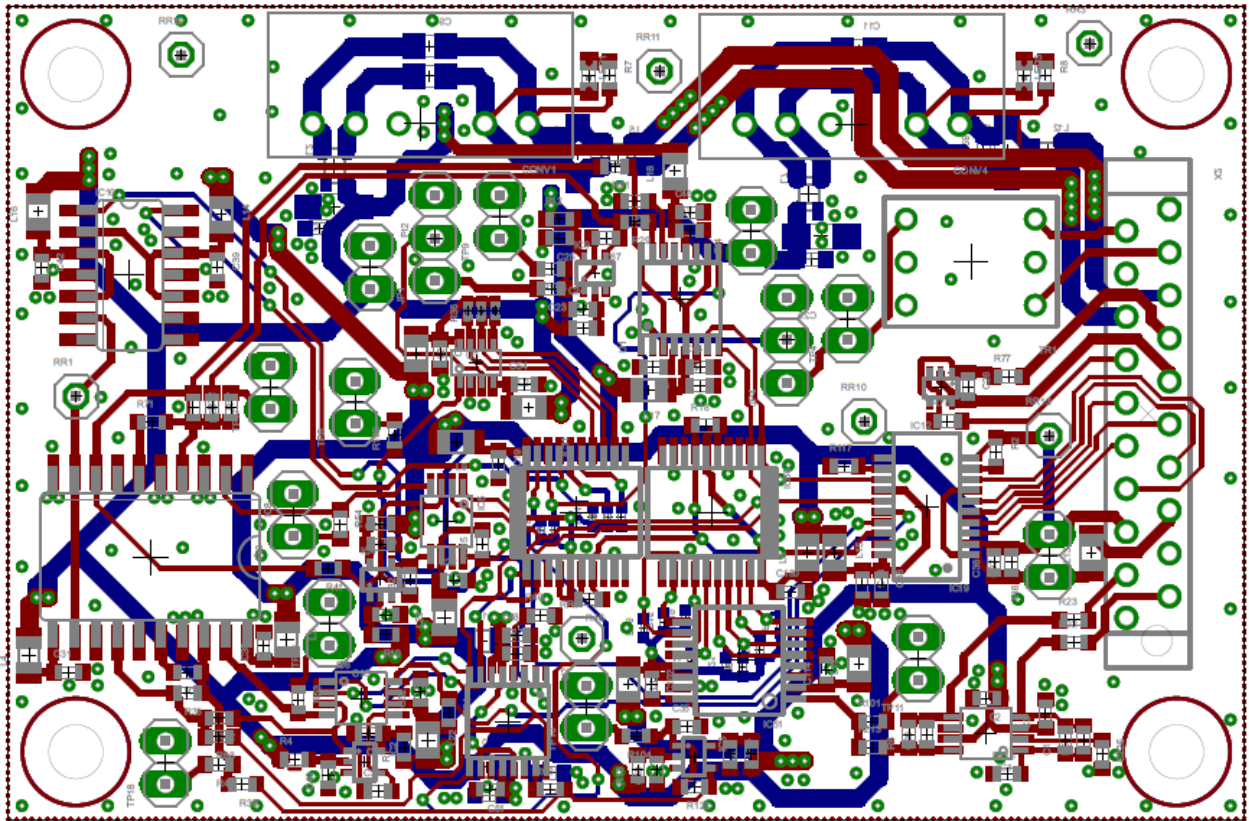
Rys. 5. Płytki AMP



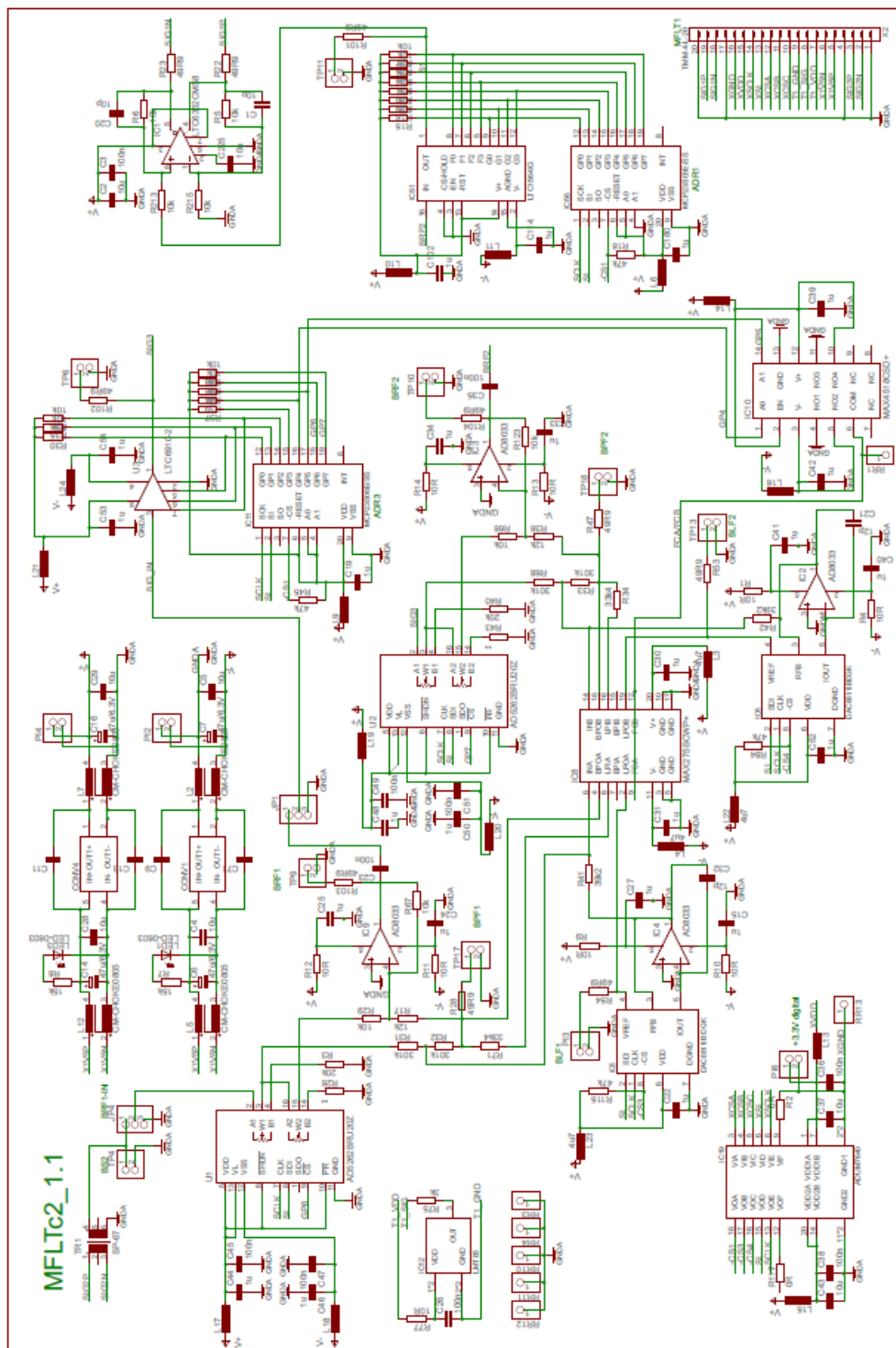
Rys. 6. Schemat płytki AMP

5.3.3 Płytki FLT

Moduł MFLT jest dołączany bezpośrednio do modułu BASE poprzez złącze standardu MM20. Płytki modułu jest umieszczona w metalowym pudełku ekranującym typu PERANCEA CFL2T (75X50X25mm).



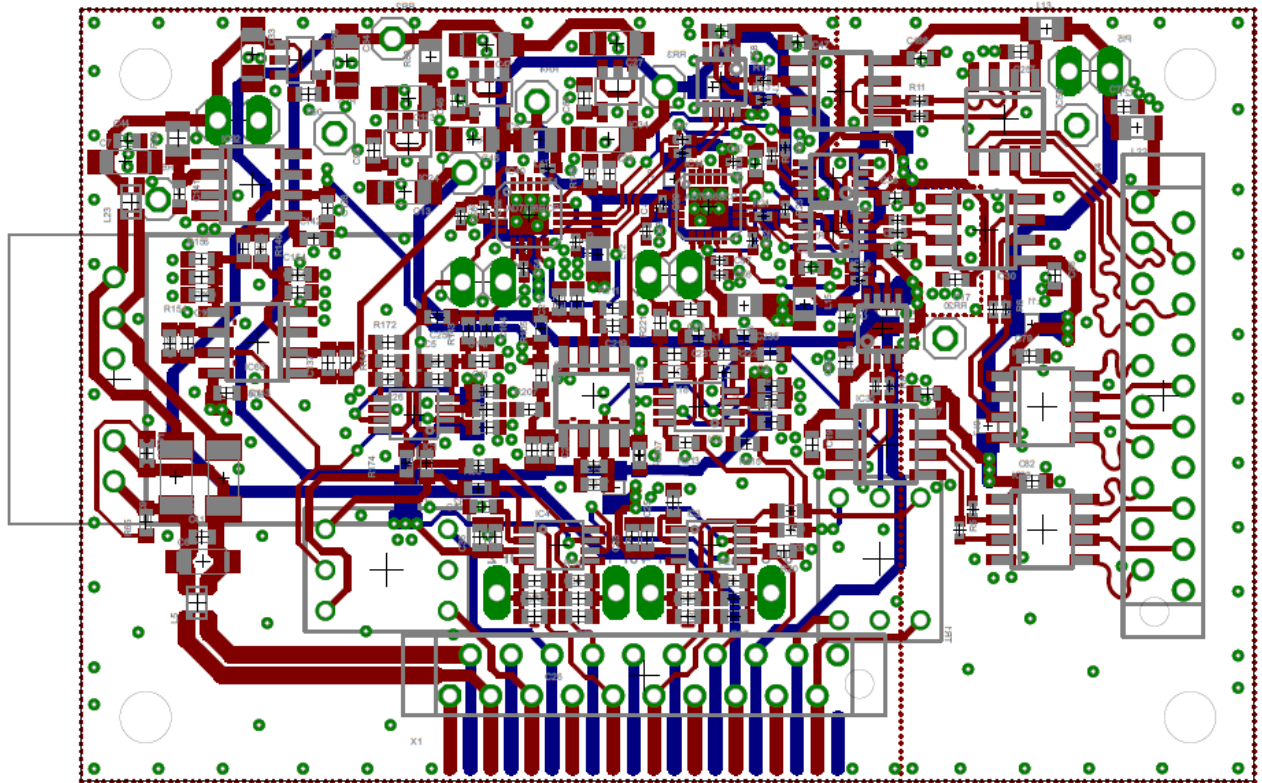
Rys. 7. Widok płytki FLT



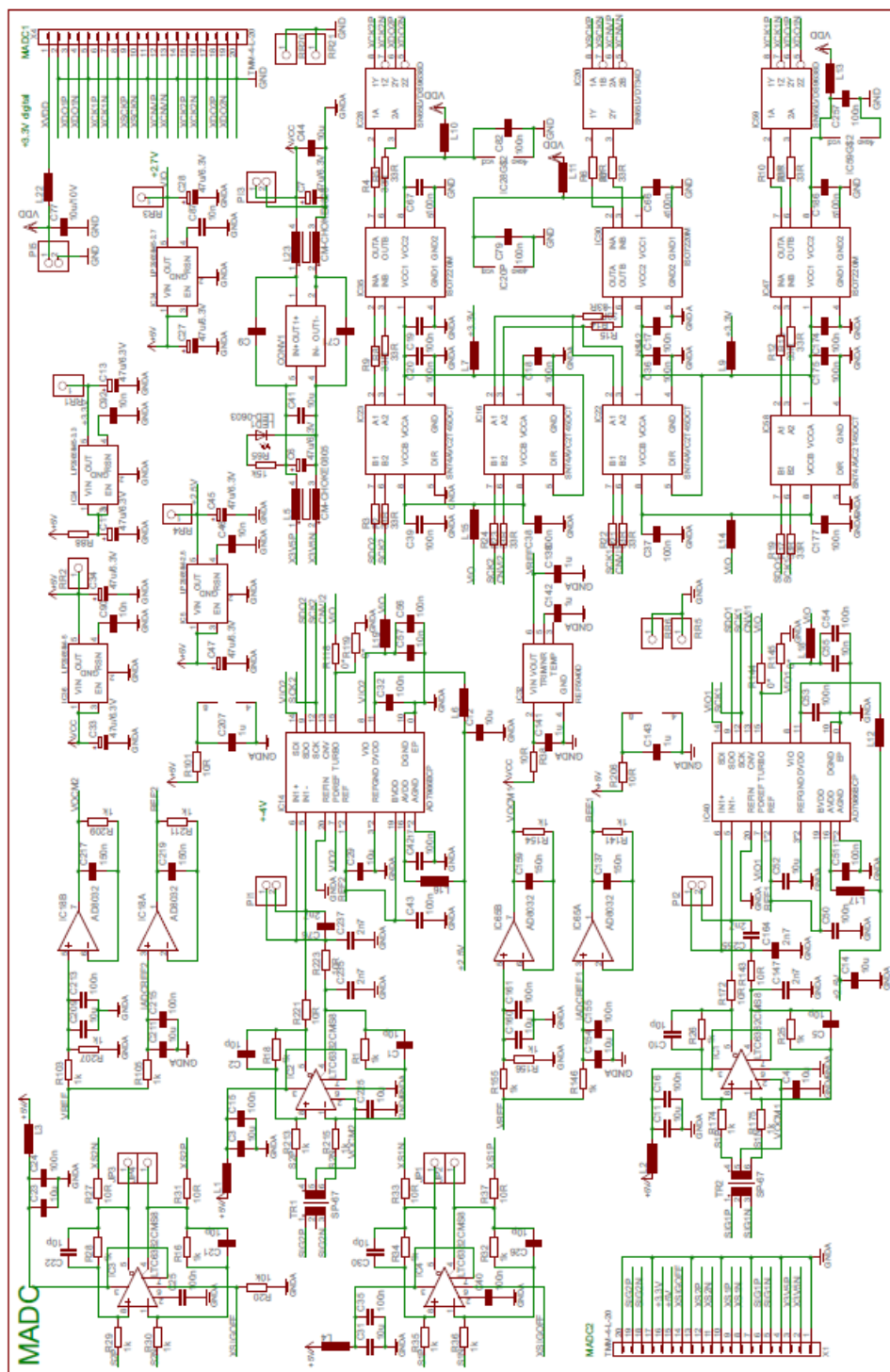
Rys. 8. Schemat płytki FLT

5.3.4 Płytki ADC

Moduł MADC jest dołączany bezpośrednio do modułu BASE poprzez złącze standardu MM20. Płytki modułu jest umieszczona w metalowym pudełku ekranującym typu PERANCEA CFL2T (75X50X25mm). Dodatkowe złącze MM20 służy do dołączenia do płytki DIGI.



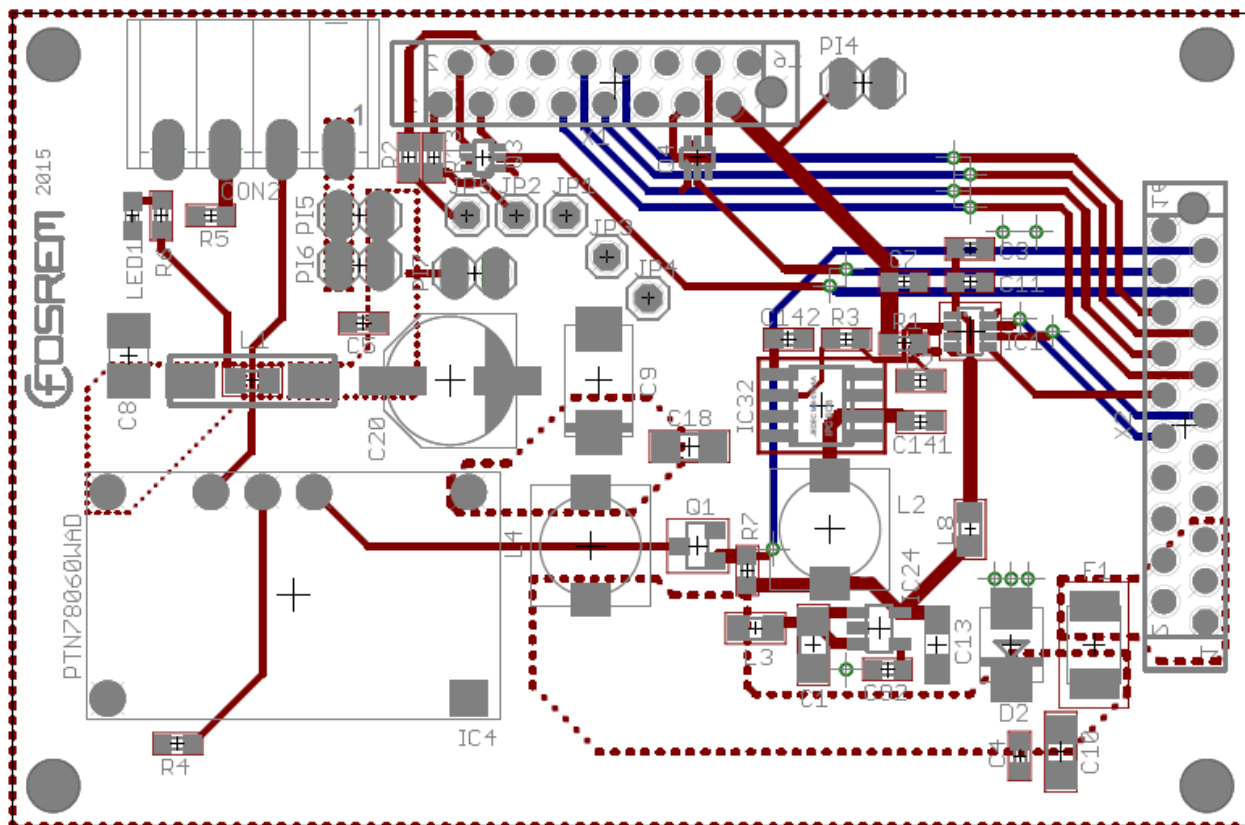
Rys. 9. Widok płytki ADC



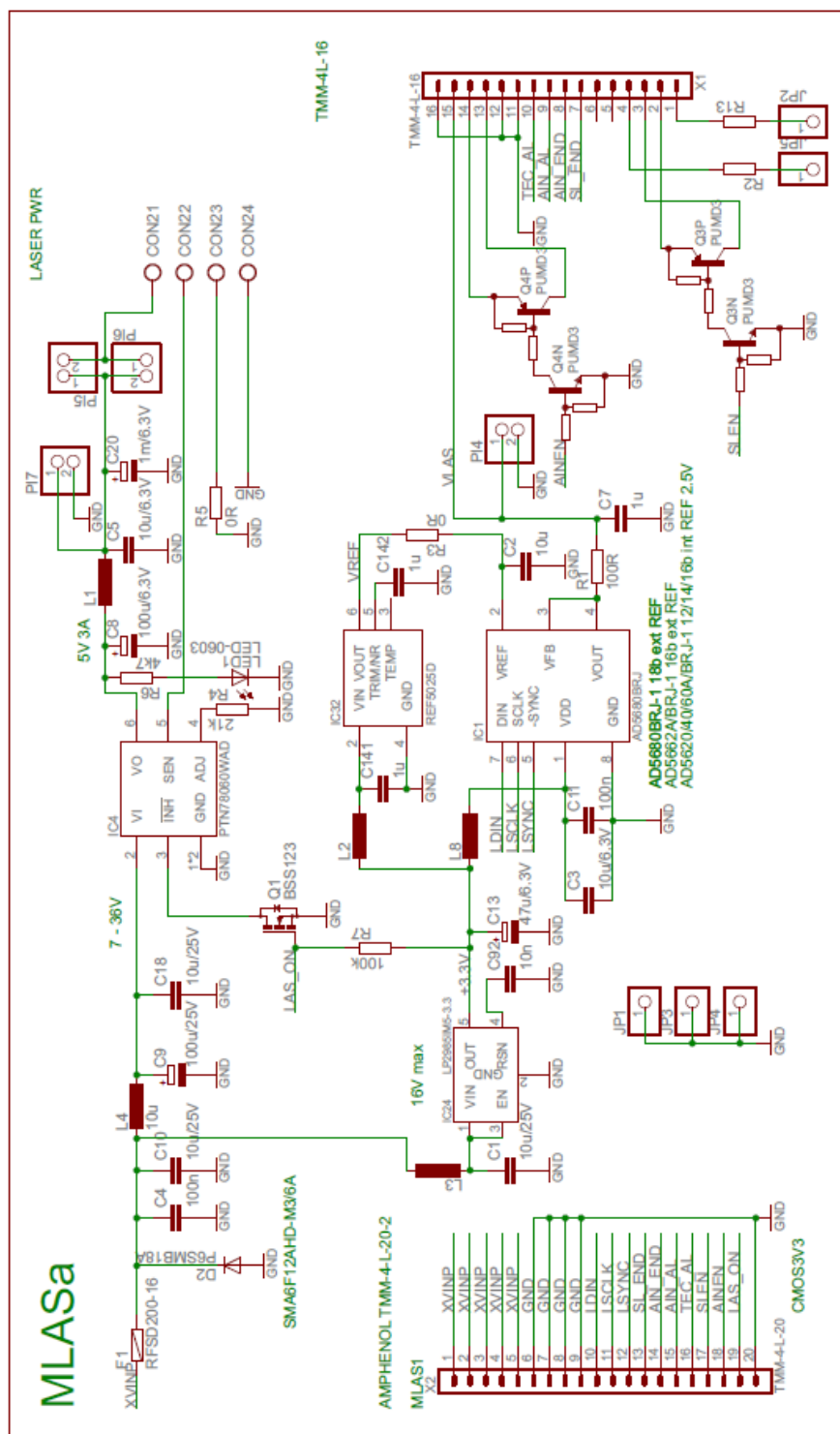
Rys. 10. Schemat płytki ADC

5.3.5 Płytki LAS

Płytki LAS zawiera złącze MM20 do dołączania do modułu DIGI. Ma też dwa złącza do dołączenia taśm zasilania i sterowania lasera.



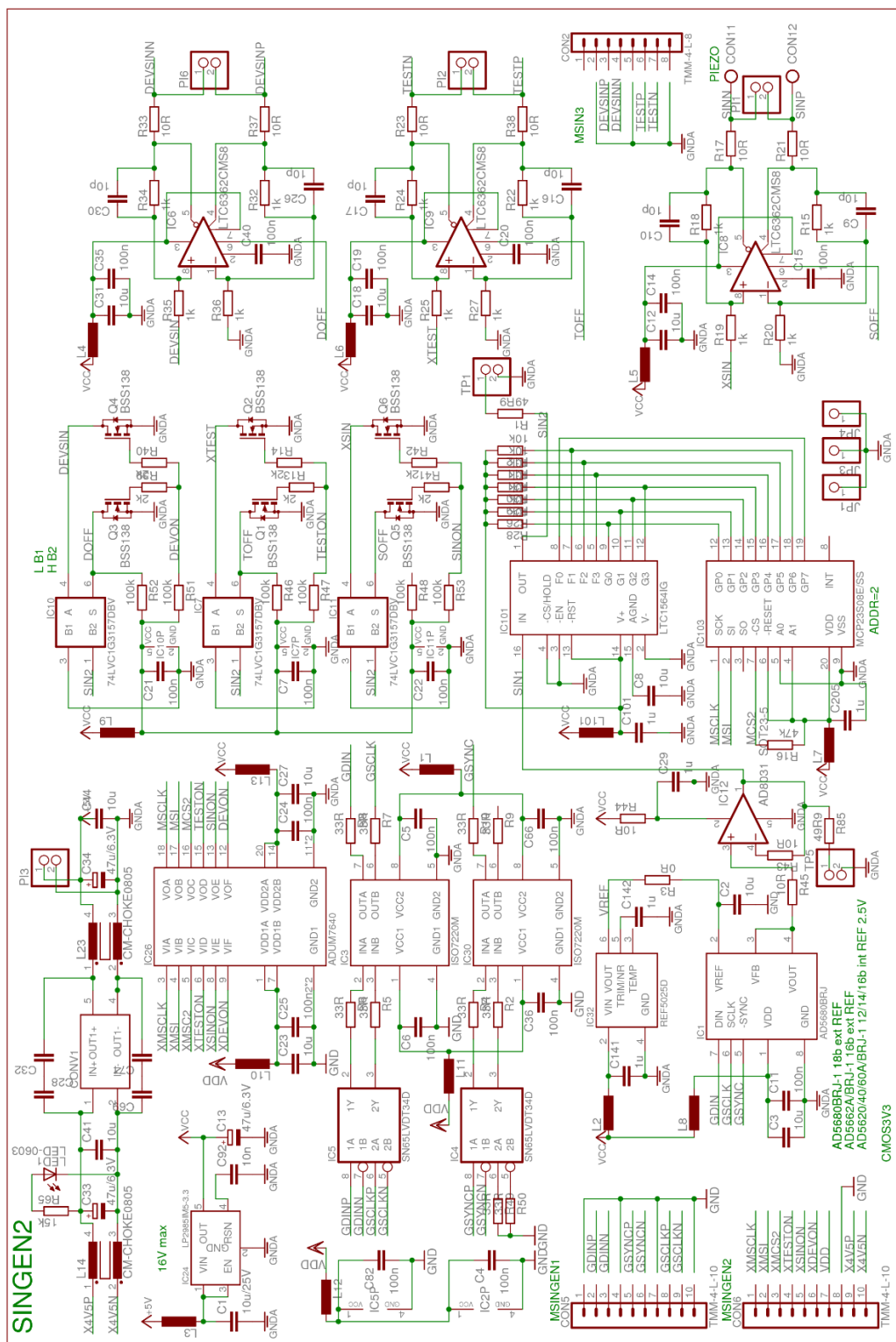
Rys. 11. Widok płytki LAS



Rys. 12. Schemat płytki LAS

5.3.6 Płytki MODGEN

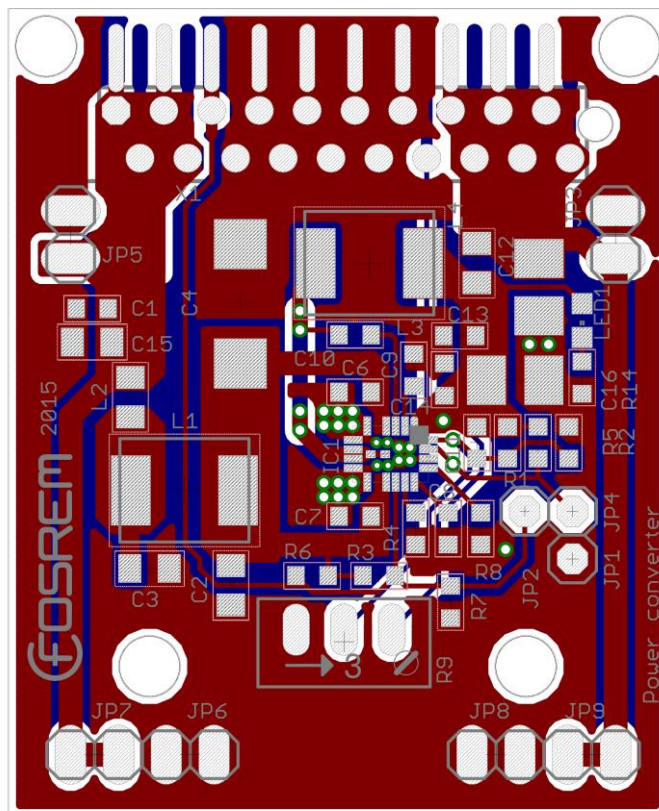
Płytki MODGEN jest przeznaczona do generowania przebiegu sterującego dla modulatora MIOC. Jest mocowana w pobliżu modulatora optycznego. Odpowiednie złącze pozwala na dołączenie modulatora. Drugie złącze pozwala na doprowadzenie sygnału do gniazda BNC w celach pomiarowych. Złącze IDC10 pozwala na dołączenie do modułu DIGI. Moduł ten został zaprojektowany jako uniwersalny generator sygnałów modulujących zarówno dla układów z MIOC jak i z niezależnym modulatorem piezoelektrycznym. Pozwala na generowanie z rozdzielczością do 18 bitów sygnału piłokształtnego, sinusoidalnego oraz innych definiowanych cyfrowo przez oprogramowanie.



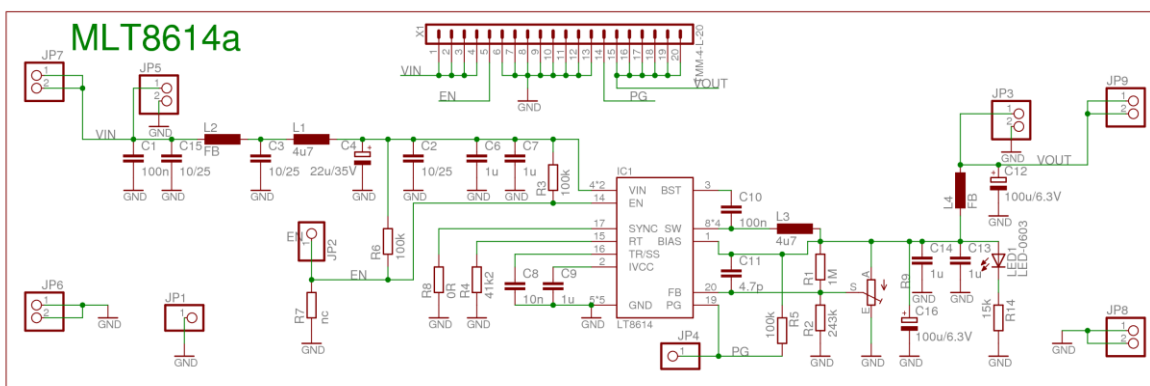
Rys. 13. Moduł układu generatora modulacji

5.3.7 Płytki modułu zasilania niskoszumnego STAB5

Moduły zasilania niskoszumnego zawierają specjalną przetwornicę pozwalającą na uzyskanie napięcia zasilania 5V przy bardzo niskich szumach generowanych na tym napięciu. Uzyskano to przez zastosowanie układu LT8614 o wysokiej częstotliwości przetwarzania oraz bardzo wysokiej sprawności. Moduł jest zasilany napięciem zasilania 12V (lub w granicach od 9 do 18 V) i generuje napięcie 5V o maksymalnym prądzie pobieranym na poziomie 3A.



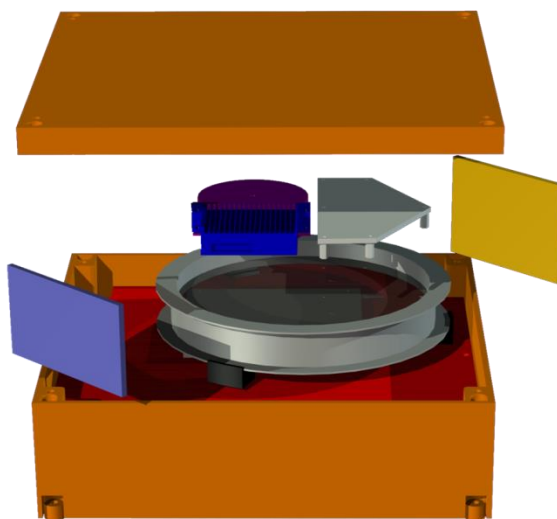
Rys. 14. Widok płytki STAB5



Rys. 15. Schemat płytki STAB5

5.4 Konstrukcja mechaniczna

Układ pomiarowy został umieszczony wraz z elementami optycznymi w typowej obudowie teletechnicznej klasy IP67. Poniżej przedstawiono wizualizację projektu mechanicznego.



Rys. 16. Widok wizualizacji układu mechanicznego

Doprowadzenie sygnałów i zasilania odbywa się łączem Ethernet PoE wykonanym również zgodnie z IP67. Pozwala to na połączenie układu detekcyjnego z układem rejestrującym przy pomocy pojedynczego przewodu FTP klasy 6e o długości do 100m.

Po zamknięciu układ jest odporny na opady deszczu oraz ochlapywanie. Pomarańczowy kolor został wybrany ze względu na lepszą widoczność w słabo zwykle oświetlonych miejscach umieszczania tego typu czujników. Obudowa jest wykonana z tworzywa poliestrowego wzmocnionego włóknem szklanym. Zapewnia to wysoką odporność na uderzenia oraz pełną odporność na korozję. Użyta uszczelka zabezpiecza przed odstawianiem się wilgoci do wrażliwych układów elektronicznych.



Rys. 17. Obudowy przeznaczonej do zamontowania układu

5.5 Program DSP

Dla realizacji wymaganych funkcji został opracowany specjalny program pracujący na module ARM pozwalający na realizację obliczeń w zakresie przetwarzania sygnałów wymaganych do pracy układu detekcyjnego.

Poniżej przedstawiono kod źródłowy tego programu

```
#include <std.h>
#include <tsk.h>
#include <stdio.h>
#include <string.h>

#include "core/cl_ipc_inbound.h"
#include "core/cl_ipc_outbound.h"

#include <assert.h>

#include <std.h>
#include <hwi.h>
#include <c6x.h>
#include <c62.h>

#include "edma3_drv.h"

#include "core/DspQDMA.h"
```

```
using namespace MityDSP;

//bools to control if messages were achieved- info o main_thread
bool recD=false;
bool recC=false;

// Forward declarations
void main_task();
void debugPrint(char *buffer);
int handleInboundMessageCOM(void *apBuffer, uint32_t anLength, void *apUserArg) ;
int handleInboundMessageDAT(void *apBuffer, uint32_t anLength, void *apUserArg) ;

// Object for sending debug messages (these are received and printed to stdout by tcDspApp)
tcCL_IPCOutbound* gpDebug;

//COMMUNICATION
// Object for sending GPPMSGQ1 messages that the ARM will receive
tcCL_IPCOutbound* gpOutboundCOM;
// Object for receiving DSPMSGQ0 messages that the DSP will receive
tcCL_IPCInbound* gpInboundCOM;

//DATA
tcCL_IPCOutbound* gpOutboundDAT;
tcCL_IPCInbound* gpInboundDAT;

/**
 * Main routine.
 */
int main(int argc, char* argv[])
{
    // initialize the DSPLink system
    tcCL_IPCInit::GetInstance();

    // Launch an initialization task
    TSK_Attrs* lpAttrs = new TSK_Attrs;
    *lpAttrs = TSK_ATTRS;
    lpAttrs->name = "Initialize";
    lpAttrs->stacksize = 8192*2;
    lpAttrs->priority = 5;

    //THE following two lines are used to stop application to debug.To be commented if not debugging
    volatile Int i=1;
    while(i);

    TSK_create((Fxn)main_task,lpAttrs);

    return 0;
}

/**
 * Initialize the inbound and outbound IPC objects.
 * \return None.
 */
void main_task()
{
    // Message to ARM core.
    char lpReturnMessageD[] = "DSP Initialization DATA CHANNEL finished.";
    char lpReturnMessageC[] = "DSP Initialization CONTROL CHANNEL finished.";
    // Buffer for return message
    char* lpMessageBufferC = NULL;
```



```
char* lpMessageBufferD = NULL;
// Create the outbound debug link
gpDebug = new tcCL_IPCOutbound("debug");

// Create the inbound link for messages to the DSP
gpInboundCOM = new tcCL_IPCInbound();
gpInboundDAT = new tcCL_IPCInbound();

gpInboundCOM->Open("DSPMSGQ0C", 8);
gpInboundDAT->Open("DSPMSGQ0D", 8);

// Create the outbound controller for sending messages to the ARM
gpOutboundCOM = new tcCL_IPCOutbound("GPPMSGQ1C");
gpOutboundDAT = new tcCL_IPCOutbound("GPPMSGQ1D");

if (NULL != gpInboundCOM)
{
    // Register a callback function to handle messages from the ARM for COM
    gpInboundCOM->RegisterCallback(handleInboundMessageCOM, (void*)NULL);
}

if (NULL != gpInboundDAT)
{
    // Register a callback function to handle messages from the ARM for DAT
    gpInboundDAT->RegisterCallback(handleInboundMessageDAT, (void*)NULL);
}

// Now that initialization is complete, let the ARM know with a message

// Obtain a dsplink buffer for the return message
lpMessageBufferC = (char*)gpOutboundCOM->GetBuffer(strlen(lpReturnMessageC) + 1);
lpMessageBufferD = (char*)gpOutboundDAT->GetBuffer(strlen(lpReturnMessageD) + 1);
// Make sure we received a valid buffer
if (NULL != lpMessageBufferC)
{
    // Copy our message to the buffer
    strcpy(lpMessageBufferC, lpReturnMessageC);
    strcpy(lpMessageBufferD, lpReturnMessageD);

    // Send the message back to the ARM
    gpOutboundCOM->SendMessage(lpMessageBufferC);
    gpOutboundDAT->SendMessage(lpMessageBufferD);
}

char MesOK[] = "STILL ALIVE-OK";

//main loop
while(1)
{
    // wait(1000);

    long delay=0;
    while(delay!=10000000)delay++;

    strcpy(lpMessageBufferC, MesOK);
    gpOutboundCOM->SendMessage(lpMessageBufferC);

    if(recD==true)
    {
        //TODO- Behaviour if DATA obtained
        recD=false;
    }
    if(recC==true)
```

```

        {
            //TODO-Behaviour if CONTROL obtained
            recC=false;
        }

        long delay2=0;
        while(delay2!=10000000)delay2++;
    }
}

/**
 * Callback function that handles messages from the ARM.
 *
 * \param apBuffer Pointer to message buffer.
 * \param anLength Length of message.
 * \param apUserArg Pointer to user defined argument
 *
 * \return 0 on sucess.
 */
int handleInboundMessageCOM(void* apBuffer, uint32_t anLength, void* apUserArg)
{
    recC=true;
    int retval = 0;
    // The return message to the ARM
    char lpReturnMessage[] = "COMMUNICATION channel. DSP Received Message = \';
    // Buffer for return message
    char* lpMessageBuffer = NULL;

    // Obtain a dsplink buffer for the return message
    lpMessageBuffer = (char*)gpOutboundCOM->GetBuffer(strlen(lpReturnMessage) + strlen((const char*)apBuffer)
+ 2);

    // Make sure we received a valid buffer
    if (NULL != lpMessageBuffer)
    {
        // Copy our message to the buffer
        strcpy(lpMessageBuffer, lpReturnMessage);
        // Append the received message to the buffer
        strcpy((char*)&lpMessageBuffer[strlen(lpReturnMessage)], (const char *)apBuffer);
        // Null terminate the string
        lpMessageBuffer[strlen(lpMessageBuffer)+1] = 0;
        // Append the closing quotation
        lpMessageBuffer[strlen(lpMessageBuffer)] = '\';

        // Send the message back to the ARM
        retval = gpOutboundCOM->SendMessage(lpMessageBuffer);
    }
    else
    {
        retval = -1;
    }
    return retval;
}

int handleInboundMessageDAT(void* apBuffer, uint32_t anLength, void* apUserArg)
{
    recD=true;
    int retval = 0;
    // The return message to the ARM
    char lpReturnMessage[] = "DATA channel. DSP Received Message = \';
    // Buffer for return message
    char* lpMessageBuffer = NULL;

    // Obtain a dsplink buffer for the return message
    lpMessageBuffer = (char*)gpOutboundDAT->GetBuffer(strlen(lpReturnMessage) + strlen((const
char*)apBuffer) + 2);

```

```
// Make sure we received a valid buffer
if (NULL != lpMessageBuffer)
{
    // Copy our message to the buffer
    strcpy(lpMessageBuffer, lpReturnMessage);
    // Append the received message to the buffer
    strcpy((char*)&lpMessageBuffer[strlen(lpReturnMessage)], (const char *)apBuffer);
    // Null terminate the string
    lpMessageBuffer[strlen(lpMessageBuffer)+1] = 0;
    // Append the closing quotation
    lpMessageBuffer[strlen(lpMessageBuffer)] = '\\';

    // Send the message back to the ARM
    retval = gpOutboundDAT->SendMessage(lpMessageBuffer);
}
else
{
    retval = -1;
}

return retval;
}

/**
 * Function for sending debug messages to the ARM.
 *
 * \param buffer    Null terminated string to be printed.
 *
 * \return None.
 */
void debugPrint(char* pMsg)
{
    // The length of the message to be sent
    int len = strlen(pMsg);
    // Pointer to dsplink buffer where to write the message
    char* pBuffer;

    // Make sure the debug IPC outbound object has been initialized
    if (gpDebug == NULL)
        return;

    // Get a buffer for the message
    pBuffer = (char *)gpDebug->GetBuffer(len+1);

    // Check that the buffer is valid
    if (pBuffer)
    {
        // Copy the message to the buffer
        strcpy(pBuffer, pMsg);
        // Send the message
        gpDebug->SendMessage(pBuffer);
    }
}
```

Do obsługi pozostałych funkcji kontrolno-sterując stworzono specjalny program na drugi rdzeń procesora ARM.

```

/**
 *      \file:   main.cpp
 *
 * \brief      ARM side main routine for DSP Hello World application.
 *              The ARM sets itself up, sends a message to the DSP,
 *              and then waits for a return message before exiting.
 *
 *      o 0
 *      | /      Copyright (c) 2005-2010
 *      (CL)---o Critical Link, LLC
 *      \
 *      O
 */

#include "dspapp.h"
#include <stdio.h>
#include <string.h>
#include <unistd.h>

#include "ipc_inbound.h"
#include "ipc_outbound.h"

using namespace MityDSP;

//Handlers of messages from DSP
int handleInboundMessageDAT(void *Buffer, uint16_t Length, void *UserArg);
int handleInboundMessageCOM(void *Buffer, uint16_t Length, void *UserArg);

// initialisation variables- checking if achieved messages from both channels
volatile bool gbDoneC = false;
volatile bool gbDoneD = false;

int main(int argc, char* argv[])
{
    // tcDspApp class for booting and loading the DSP
    tcDspApp*          lpDspApp = NULL;

    //Communication channels- In fo ARM- out for DSP and vice versa

    tcIPCInbound*      lpMessageInboundCOM = NULL;
    tcIPCOutbound*     lpMessageOutboundCOM = NULL;

    //Data channels
    tcIPCInbound*      lpMessageInboundDAT = NULL;
    tcIPCOutbound*     lpMessageOutboundDAT = NULL;

    char                lpMessage[] = "Test DSP";
    // Pointer to buffer obtained from dsplink
    char*               lpMessageBufferC = NULL;
    char*               lpMessageBufferD = NULL;
    // Message from standard input
    char                lpUsrMsg[180] = {0};

    // Check application usage
    if (argc < 2)
    {
        printf("usage: HelloWorld dsp_image.out\n");
        return -1;
    }

    // Create the DspApp object
    lpDspApp = new tcDspApp();

    // Load the DSP.out file
    printf("Loading file %s\n", argv[1]);
    lpDspApp->LoadApp(argv[1]);

```

```

printf("Starting application.\n");

// Create the objects to handle incoming messages from the DSP
lpMessageInboundCOM = new tcIPCInbound((char*)"GPPMSGQ1C");
lpMessageInboundDAT = new tcIPCInbound((char*)"GPPMSGQ1D");

//initialise message inbounds for both channels
if (NULL != lpMessageInboundCOM)
{
    // Register the callback for handling messages from the DSP
    lpMessageInboundCOM->Register(handleInboundMessageCOM, (void*)NULL);
    // Intiaillize the inbound controller to create the thread that handles the callbacks
    lpMessageInboundCOM->Initialize();
}
if (NULL != lpMessageInboundDAT)
{
    lpMessageInboundDAT->Register(handleInboundMessageDAT, (void*)NULL);
    lpMessageInboundDAT->Initialize();
}

// Create the object used to send messages to the DSP
lpMessageOutboundCOM = new tcIPCOutbound((char*)"DSPMSGQ0C");
lpMessageOutboundDAT = new tcIPCOutbound((char*)"DSPMSGQ0D");

// Wait for the DSP to finish initialization
while(false == gbDoneC || gbDoneD==false);

// Get a buffer for a message to the DSP
lpMessageBufferC = (char*)lpMessageOutboundCOM->GetBuffer(strlen(lpMessage)+1);
lpMessageBufferD = (char*)lpMessageOutboundDAT->GetBuffer(strlen(lpMessage)+1);

// Copy the message to the dsplink buffer
strcpy(lpMessageBufferC, lpMessage);
strcpy(lpMessageBufferD, lpMessage);

//A test- Sending the messages to the DSP vis both channels
printf("Sending a message to the DSP via com\n");
lpMessageOutboundCOM->SendMessage(lpMessageBufferC);

printf("Sending a message to the DSP via DAT\n");
lpMessageOutboundDAT->SendMessage(lpMessageBufferD);

// Wait for a message to be received from the DSP or for user to quit
printf("Waiting for DSP response (type \'q\' to quit)...\n");

//infinite loop-now threads will handle the messages
while( lpUsrMsg[0] != 'q')
{
    fgets(lpUsrMsg, 180, stdin);
}

printf ("Application exit requested. Closing \n");
lpDspApp->StopApp();
//No stop,except q when waiting for DSP !
}

```

```

int handleInboundMessageDAT(void *apBuffer, uint16_t anLength, void *apUserArg)

```

```
{
    printf("ARM received a DATA message from the DSP:\n");

    // Print the message we received
    printf("\tdSP Message DATA = \"%s\"\n", (char *)apBuffer);

    // Notify the main function that we have received a message from the DSP and are done
    gbDoneD = true;

    return 0;
}

int handleInboundMessageCOM(void *apBuffer, uint16_t anLength, void *apUserArg)
{
    printf("ARM received a  COMMUNICATION message from the DSP:\n");

    // Print the message we received
    printf("\tdSP Message COMMUNICATION = \"%s\"\n", (char *)apBuffer);

    // Notify the main function that we have received a message from the DSP and are done
    gbDoneC = true;

    return 0;
}
```

W tabeli poniżej pokazano przypisanie sygnałów SoC w układzie detekcyjnym

Signal	Connector	Module	CON1	Net Name	JX1	Zynq AP SoC	Zynq SoC Pin Name
FAINEN	MLAS1.18		3	IO_0_34	9	Bank 34, R19	IO_0_34
FLDIN	MLAS1.10		4	IO_25_34	10	Bank 34, T19	IO_25_34
FLAS_ON	MLAS1.19		5	IO_L1P_T0_34	11	Bank 34, T11	IO_L1P_T0_34
FLSCLK	MLAS1.11		6	IO_L2P_T0_34	12	Bank 34, T12	IO_L2P_T0_34
			7	IO_L1N_T0_34	13	Bank 34, T10	IO_L1N_T0_34
			8	IO_L2N_T0_34	14	Bank 34, U12	IO_L2N_T0_34
			9	IO_L3P_T0_34	17	Bank 34, U13	IO_L3P_T0_34
			10	IO_L4P_T0_34	18	Bank 34, V12	IO_L4P_T0_34
UNII_O	DIGI-11AA		11	IO_L3N_T0_34	19	Bank 34, V13	IO_L3N_T0_34
			12	IO_L4N_T0_34	20	Bank 34, W13	IO_L4N_T0_34
			13	IO_L5P_T0_34	23	Bank 34, T14	IO_L5P_T0_34
			14	IO_L6P_T0_34	24	Bank 34, P14	IO_L6P_T0_34
			15	IO_L5N_T0_34	25	Bank 34, T15	IO_L5N_T0_34
FSLEN	MLAS1.17		16	IO_L6N_T0_34	26	Bank 34, R14	IO_L6N_T0_34
SDA1	MBASE1.3 CON5.3	*DIGI IC16.5 FM24CL64 ADDR=1	19	IO_L7P_T1_34	29	Bank 34, Y16	IO_L7P_T1_34

		*BASE IC1.5 FM24CL64 ADDR=0 *BASE IC12.9 ADS1115 ADDR=0					
			20	IO_L8P_T1_34	30	Bank 34, W14	IO_L8P_T1_34
SCL1	MBASE1.2 CON5.2	*DIGI IC16.6 FM24CL64 ADDR=1 *BASE IC1.6 FM24CL64 ADDR=0 *BASE IC12.10 ADS1115 ADDR=0	21	IO_L7N_T1_34	31	Bank 34, Y17	IO_L7N_T1_34
			22	IO_L8N_T1_34	32	Bank 34, Y14	IO_L8N_T1_34
			23	IO_L9P_T1_34	35	Bank 34, T16	IO_L9P_T1_34
XDO1P	MADC1.3	DATA OUT1+	24	IO_L10P_T1_34	36	Bank 34, V15	IO_L10P_T1_34
			25	IO_L9N_T1_34	37	Bank 34, U17	IO_L9N_T1_34
XDO1N	MADC1.4	DATA OUT1-	26	IO_L10N_T1_34	38	Bank 34, W15	IO_L10N_T1_34
XSCKP	MADC1.9	SCLK IN+	27	IO_L11P_T1_34	41	Bank 34, U14	IO_L11P_T1_SRCC_34
XCK1P	MADC1.6	SCLK OUT1+	28	IO_L12P_T1_34	42	Bank 34, U18	IO_L12P_T1_MRCC_34
XSCKN	MADC1.10	SCLK IN-	29	IO_L11N_T1_34	43	Bank 34, U15	IO_L11N_T1_SRCC_34
XCK1N	MADC1.7	SCLK OUT1-	30	IO_L12N_T1_34	44	Bank 34, U19	IO_L12N_T1_MRCC_34
XCNPV	MADC1.12	CONVERT+	31	IO_L13P_T2_34	47	Bank 34, N18	IO_L13P_T2_MRCC_34
XCK2P	MADC1.15	SCLK OUT2+	32	IO_L14P_T2_34	48	Bank 34, N20	IO_L14P_T2_SRCC_34
XCNPV	MADC1.13	CONVERT-	33	IO_L13N_T2_34	49	Bank 34, P19	IO_L13N_T2_MRCC_34
XCK2N	MADC1.16	SCLK OUT2-	34	IO_L14N_T2_34	50	Bank 34, P20	IO_L14N_T2_SRCC_34
			35	IO_L15P_T2_34	53	Bank 34, T20	IO_L15P_T2_DQS_34
XDO2P	MADC1.18	DATA OUT2+	36	IO_L16P_T2_34	54	Bank 34, V20	IO_L16P_T2_34
			37	IO_L15N_T2_34	55	Bank 34, U20	IO_L15N_T2_DQS_34
XDO2N	MADC1.19	DATA OUT2-	38	IO_L16N_T2_34	56	Bank 34, W20	IO_L16N_T2_34
			39, 40	5V	59,60	5V	5V
			41	IO_L17P_T2_34	61	Bank 34, Y18	IO_L17P_T2_34
FLSYNC	MLAS1.12		42	IO_L18P_T2_34	62	Bank 34, V16	IO_L18P_T2_34

			43	IO_L17N_T2_34	63	Bank 34, Y19	IO_L17N_T2_34
FSL_END	MLAS1.13		44	IO_L18N_T2_34	64	Bank 34, W16	IO_L18N_T2_34
			45	IO_L19P_T3_34	67	Bank 34, R16	IO_L19P_T3_34
FAIN_END	MLAS1.14		46	IO_L20P_T3_34	68	Bank 34, T17	IO_L20P_T3_34
			47	IO_L19N_T3_34	69	Bank 34,R17	IO_L19N_T3_VREF_34
FAIN_AL	MLAS1.15		48	IO_L20N_T3_34	70	Bank 34, R18	IO_L20N_T3_34
AN_ON	DIGI-APWR	DIGI- Analog PWR Enable	49	IO_L21P_T3_34	73	Bank 34, V17	IO_L21P_T3_DQS_34
FTEC_AL	MLAS1.16		50	IO_L22P_T3_34	74	Bank 34, W18	IO_L22P_T3_34
FXCSC	MBASE1.10	MFLT1.10 CS4 IC8.1 BRF2 DAC8811	51	IO_L21N_T3_34	75	Bank 34, V18	IO_L21N_T3_DQS_34
FXSI	MBASE1.13	MAMP1.13, MFLT1.13	52	IO_L22N_T3_34	76	Bank 34, W19	IO_L22N_T3_34
			53, 54	VCCIO	77, 78	Bank 34 VCCO	VCCO 34
FXCSB	MBASE1.11	MFLT1.11 CS3 IC5.8 BRF1 DAC8811	55	IO_L23P_T3_34	81	Bank 34, N17	IO_L23P_T3_34
FXCSA	MBASE1.14	* MAMP1.12 CS2 IC103.7 MCP23S08E ADDR=2 (AMP2 + LPF2 LTC1564) * MFLT1.12 CS1 IC56.7 MCP23S08E ADDR=1 (AMP1 + LPF1 LTC1564) * MFLT1.12 CS1 IC11.7 MCP23S08E ADDR=3 (GP0-GP2 AMP LTC6910-2) (GP4-GP5 FCA/B MAX4518) (GP6 SIGAMP2 AD5362.9) (GP7 SIGAMP3 AD5362.9)	56	IO_L24P_T3_34	82	Bank 34, P15	IO_L24P_T3_34
FXSCLK	MBASE1.12	MAMP1.14, MFLT1.14	57	IO_L23N_T3_34	83	Bank 34, P18	IO_L23N_T3_34
FXSIGOFF	MBASE1.15	MADC2.14 EXTSIGNALENABLE IC3.6 IC4,IC6 LTC6362.6	58	IO_L24N_T3_34	84	Bank 34, P16	IO_L24N_T3_34
			TP7	IO_L11P_T1_13	87	Bank 13, U7	IO_L11P_T1_SRCC_13
			TP6	IO_L12P_T1_13	88	Bank 13, T9	IO_L12P_T1_MRCC_13
			TP9	IO_L11N_T1_13	89	Bank 13, V7	IO_L11N_T1_SRCC_13
			TP8	IO_L12N_T1_13	90	Bank 13, U10	IO_L12N_T1_MRCC_13
			TP11	IO_L15P_T2_13	91	Bank 13, V8	IO_L15P_T2_DQS_13
			TP10	IO_L19P_T3_13	92	Bank 13, T5	IO_L19P_T3_13
			TP13	IO_L15N_T2_13	93	Bank 13, W8	IO_L15N_T2_DQS_13

			TP12	IO_L19N_T3_13	94	Bank 13, U5	IO_L19N_T3_VREF_13
			1,2, 17,18 59,60	GND	16	GND	GND

Signal	Connector	Module	CON2	Net Name	JX2	Zynq AP SoC	Zynq SoC Pin Name
PG_CARRIER	[Open drain]	DIGI-TPS3103	1	PG_CARRIER	11	Bank 500, C7	PS_POR_B_500
VCCIO_EN	]	[Input CMOS1.8V	2	VCCIO_EN	10	PG_1V8	N/A
			3	IO_0_35	13	Bank 35, G14	IO_0_35
FXMSCLK	MSINGEN2.10		4	IO_25_35	14	Bank 35, J15	IO_25_35
			5	IO_L1P_T0_35	17	Bank 35, C20	IO_L1P_T0_AD0P_35
FXMSI	MSINGEN2.9		6	IO_L2P_T0_35	18	Bank 35, B19	IO_L2P_T0_AD8P_35
			7	IO_L1N_T0_35	19	Bank 35, B20	IO_L1N_T0_AD0N_35
FXMSC2	MSINGEN2.8		8	IO_L2N_T0_35	20	Bank 35, A20	IO_L2N_T0_AD8N_35
			9	IO_L3P_T0_35	23	Bank 35, E17	IO_L3P_T0_DQS_AD1P_35
FXTESTON	MSINGEN2.7		10	IO_L4P_T0_35	24	Bank 35, D19	IO_L4P_T0_35
			11	IO_L3N_T0_35	25	Bank 35, D18	IO_L3N_T0_DQS_AD1N_35
FXSINON	MSINGEN2.6		12	IO_L4N_T0_35	26	Bank 35, D20	IO_L4N_T0_35
			13	IO_L5P_T0_35	29	Bank 35, E18	IO_L5P_T0_AD9P_35
FXDEVON	MSINGEN2.5		14	IO_L6P_T0_35	30	Bank 35, F16	IO_L6P_T0_35
			15	IO_L5N_T0_35	31	Bank 35, E19	IO_L5N_T0_AD9N_35
			16	IO_L6N_T0_35	32	Bank 35, F17	IO_L6N_T0_VREF_35
			19	IO_L9P_T1_35	35	Bank 35, L19	IO_L9P_T1_DQS_AD3P_35
SDA2	DIGI-RTC	U3.5 MCP79410	20	IO_L7P_T1_35	36	Bank 35, M19	IO_L7P_T1_AD2P_35
			21	IO_L9N_T1_35	37	Bank 35, L20	IO_L9N_T1_DQS_AD3N_35
SCL2	DIGI-RTC	U3.6 MCP79410	22	IO_L7N_T1_35	38	Bank 35, M20	IO_L7N_T1_AD2N_35
			23	IO_L8P_T1_35	41	Bank 35, M17	IO_L8P_T1_AD10P_35
MFP2	DIGI-RTC	U3.7 MCP79410	24	IO_L10P_T1_35	42	Bank 35,	IO_L10P_T1_AD11P_35

						K19	
			25	IO_L8N_T1_35	43	Bank 35, M18	IO_L8N_T1_AD10N_35
			26	IO_L10N_T1_35	44	Bank 35, J19	IO_L10N_T1_AD11N_35
GDINP	MSINGEN2.19		27	IO_L11P_T1_35	47	Bank 35, L16	IO_L11P_T1_SRCC_35
GSYNCP	MSINGEN2.16		28	IO_L12P_T1_35	48	Bank 35, K17	IO_L12P_T1_MRCC_35
GDINN	MSINGEN2.18		29	IO_L11N_T1_35	49	Bank 35, L17	IO_L11N_T1_SRCC_35
GSYNCP	MSINGEN2.15		30	IO_L12N_T1_35	50	Bank 35, K18	IO_L12N_T1_MRCC_35
GSCLKP	MSINGEN2.13		31	IO_L13P_T2_35	53	Bank 35, H16	IO_L13P_T2_MRCC_35
			32	IO_L14P_T2_35	54	Bank 35, J18	IO_L14P_T2_AD4P_SRCC_35
GSCLKN	MSINGEN2.12		33	IO_L13N_T2_35	55	Bank 35, H17	IO_L13N_T2_MRCC_35
			34	IO_L14N_T2_35	56	Bank 35, H18	IO_L14N_T2_AD4N_SRCC_35
			35, 36	5V	57, 58	5V	5V
			37	IO_L16P_T2_35	61	Bank 35, G17	IO_L16P_T2_35
FSINPWRON	SINGENstd.4	SINPWRON	38	IO_L15P_T2_35	62	Bank 35, F19	IO_L15P_T2_DQS_AD12P_35
			39	IO_L16N_T2_35	63	Bank 35, G18	IO_L16N_T2_35
FSINSYNC	SINGENstd.8	FSINSYNC	40	IO_L15N_T2_35	64	Bank 35, F20	IO_L15N_T2_DQS_AD12N_35
			41	IO_L18P_T2_35	67	Bank 35, G19	IO_L18P_T2_AD13P_35
FSINCLK	SINGENstd.7	SINCLK	42	IO_L17P_T2_35	68	Bank 35, J20	IO_L17P_T2_AD5P_35
			43	IO_L18N_T2_35	69	Bank 35, G20	IO_L18N_T2_AD13N_35
FSINSDI	SINGENstd.6	SINSDI	44	IO_L17N_T2_35	70	Bank 35, H20	IO_L17N_T2_AD5N_35
			45	IO_L20P_T3_35	73	Bank 35, K14	IO_L20P_T3_AD6P_35
FSINCS	SINGENstd.5	SINCS	46	IO_L19P_T3_35	74	Bank 35, H15	IO_L19P_T3_35
			47	IO_L20N_T3_35	75	Bank 35, J14	IO_L20N_T3_AD6N_35
FSINF32	SINGENstd.9	SINF32	48	IO_L19N_T3_35	76	Bank 35, G15	IO_L19N_T3_VREF_35
			49, 50	VCCIO	77, 78	Bank 35 VCCO	VCCO 35
SCL	MEMS1.3		51	IO_L21P_T3_35	81	Bank 35, N15	IO_L21P_T3_DQS_AD14P_35
FSYNC	MEMS1.10		52	IO_L22P_T3_35	82	Bank 35, L14	IO_L22P_T3_AD7P_35

SDA	MEMS1.4		53	IO_L21N_T3_35	83	Bank 35, N16	IO_L21N_T3_DQS_AD14N_35
NCS	MEMS1.9		54	IO_L22N_T3_35	84	Bank 35, L15	IO_L22N_T3_AD7N_35
EDA	MEMS1.5		55	IO_L23P_T3_35	87	Bank 35, M14	IO_L23P_T3_35
INT	MEMS1.8		56	IO_L24P_T3_35	88	Bank 35, K16	IO_L24P_T3_AD15P_35
ECL	MEMS1.6		57	IO_L23N_T3_35	89	Bank 35, M15	IO_L23N_T3_35
AD0	MEMS1.7		58	IO_L24N_T3_35	90	Bank 35, J16	IO_L24N_T3_AD15N_35
			TP15	IO_L20P_T3_13	93	Bank 13, Y12	IO_L20P_T3_13
			TP14	IO_L21P_T3_13	94	Bank 13, V11	IO_L21P_T3_DQS_13
			TP17	IO_L20N_T3_13	95	Bank 13, Y13	IO_L20N_T3_13
			TP16	IO_L21N_T3_13	96	Bank 13, V10	IO_L21N_T3_DQS_13
			TP18	IO_L22P_T3_13	97	Bank 13, V6	IO_L22P_T3_13
			TP20	IO_L22N_T3_13	99	Bank 13, W6	IO_L22N_T3_13
			TP19	IO_L6N_T0_13	100	Bank 13, V5	IO_L6N_T0_VREF_13
			17,18, 59,60	GND	92	GND	GND

5.6 Testy układów elektronicznych

W ramach sprawdzenia poprawności działania prototypu wykonano szereg testów ukierunkowanych na określenie wartości istotnych parametrów sterowania układu.

5.6.1 Warunki standardowe przeprowadzania pomiaru

Testy zostały przeprowadzone w zbliżonych warunkach. Dla przeprowadzenia niektórych testów warunki były modyfikowane lub uzupełniane jak to zostało dodatkowo zaznaczone przy ich opisie.

Ustawienia regulacyjne układu FOSREMr2:

- Temperatura pokojowa 22°C
- Częstotliwość sygnału generatora sinusa: 41 kHz
- Amplituda sygnału generatora sinusa: 40000, zakres od 1 (min) do 65535 (max)
- Moc lasera sterowana cyfrowo od 1 (min) do 65535 (1/1, max), standardowo ustawiona na maksimum mocy
- Wzmocnienie wzmacniacza TIA (transimpedancyjnego) równe 10 kΩ.

- Wzmocnienie wzmacniacza PGA2 równe 2 V/V.
- Pasma filtru LPF2 równe 50 kHz.
- Wzmocnienie wzmacniacza PGA3 równe 1 V/V.
- Wzmocnienie wzmacniacza PGA1 równe 2 V/V.
- Pasma filtru LPF1 równe 30 kHz.

Aparatura pomiarowa:

- Oscyloskop Keysight MS)-X 4104A, akwizycja w trybie High Res, podłączenie 1:1 kablami ekranowanymi do punktów testowych, programowy moduł pomiarowy

Warunki pomiarowe:

- Układ pracuje w standardowej konfiguracji, zdjęta pokrywa, występują drgania zewnętrzne, po długim czasie wygrzewania.

Mierzone sygnały:

Kanał CH2: Sygnał TIA – bezpośrednio po wzmacniaczu TIA (moduł MAPMP)

Kanał CH3: Sygnał BS2 – po wzmacniaczu odcinającym składową stałą (moduł MAPMP)

Kanał CH4: Sygnał S2 – po wzmacniaczu PGA2 z filtrem FLT2 (moduł MAPMP)

5.6.2 Badanie poziomu składowej stałej sygnału TIA w zależności od czasu

Tabela 3. Zmiana poziomu wejściowego sygnału świetlnego w czasie

Moc lasera poziom	GS2 V/V	DC TIA V	AC TIA Vpp	BS2 Vpp	S2 Vpp	S2 Vrms	Czas Min	Uwagi rys
65535	1	-2.07	1.37	1.35	1.42	0.498	0	Sc3
65535	1	-1.94	1.30	1.28	1.34	0.475	2	Sc4
65535	1	-1.90	1.28	1.25	1.32	0.462	5	Sc5
65535	1	-1.87	1.25	1.23	1.29	0.454	8	Sc6
65535	1	-1.85	1.23	1.21	1.28	0.449	12	Sc7
65535	1	-1.83	1.22	1.20	1.27	0.444	15	Sc8

Wniosek: Sygnał zmienia poziom z czasem. Nie ma to większego wpływu na pomiar wartości Omega. Jednak wymaga doboru wzmocnienia poszczególnych elementów toru pomiarowego tak aby nie występowało zjawisko nasycenia.

5.6.3 Badanie poziomu składowej stałej sygnału TIA w zależności od poziomu mocy lasera

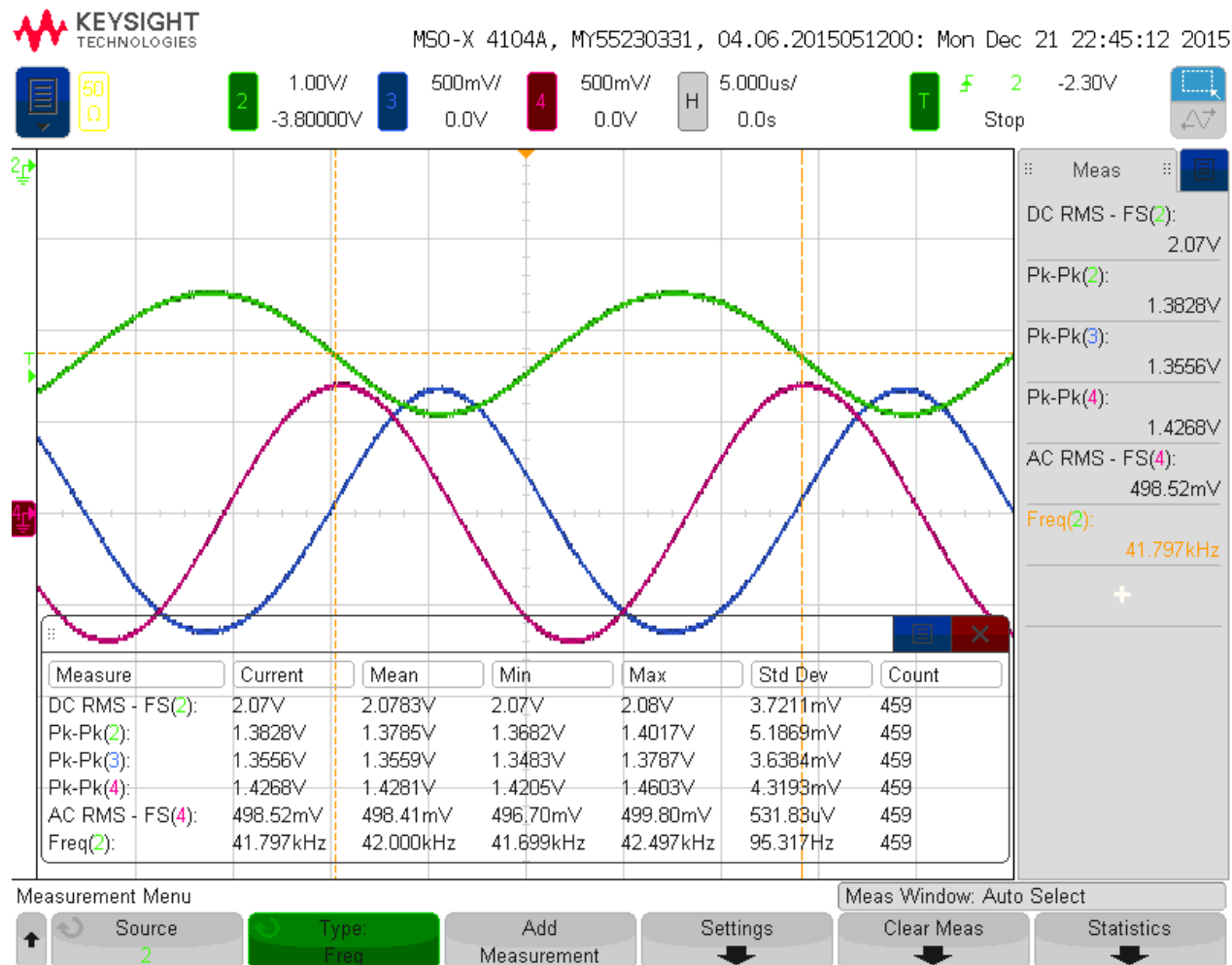


Tabela 4. Zmiana poziomu wejściowego sygnału świetlnego w zależności od mocy lasera

Moc lasera poziom	GS2 V/V	DC TIA V	AC TIA Vpp	BS2 Vpp	S2 Vpp	S2 Vrms	Uwagi rys
65535 = 1/1	1	-1.83	1.237	1.21	1.272	0.446	Sc9
49152 = 3/4	1	-1.39	0.929	0.925	0.974	0.341	Sc10
32768 = 1/2	1	-0.730	0.487	0.488	0.515	0.179	Sc11
16384 = 1/4	1	-0.84	0.056	0.058	0.060	0.020	Sc12

65535 =1/1	1	-1.83	1.218	1.210	1.282	0.452	Sc13
------------	---	-------	-------	-------	-------	-------	------

Wniosek: Zmiany nie są liniowe. Początkowy spadek sygnału z mocą lasera jest nieduży. Poniżej połowy mocy lasera sygnał jest zbyt niski.

5.6.4 Badanie wpływu poziomu sygnału modulacyjnego

Mierzone sygnały: (dodatkowo)

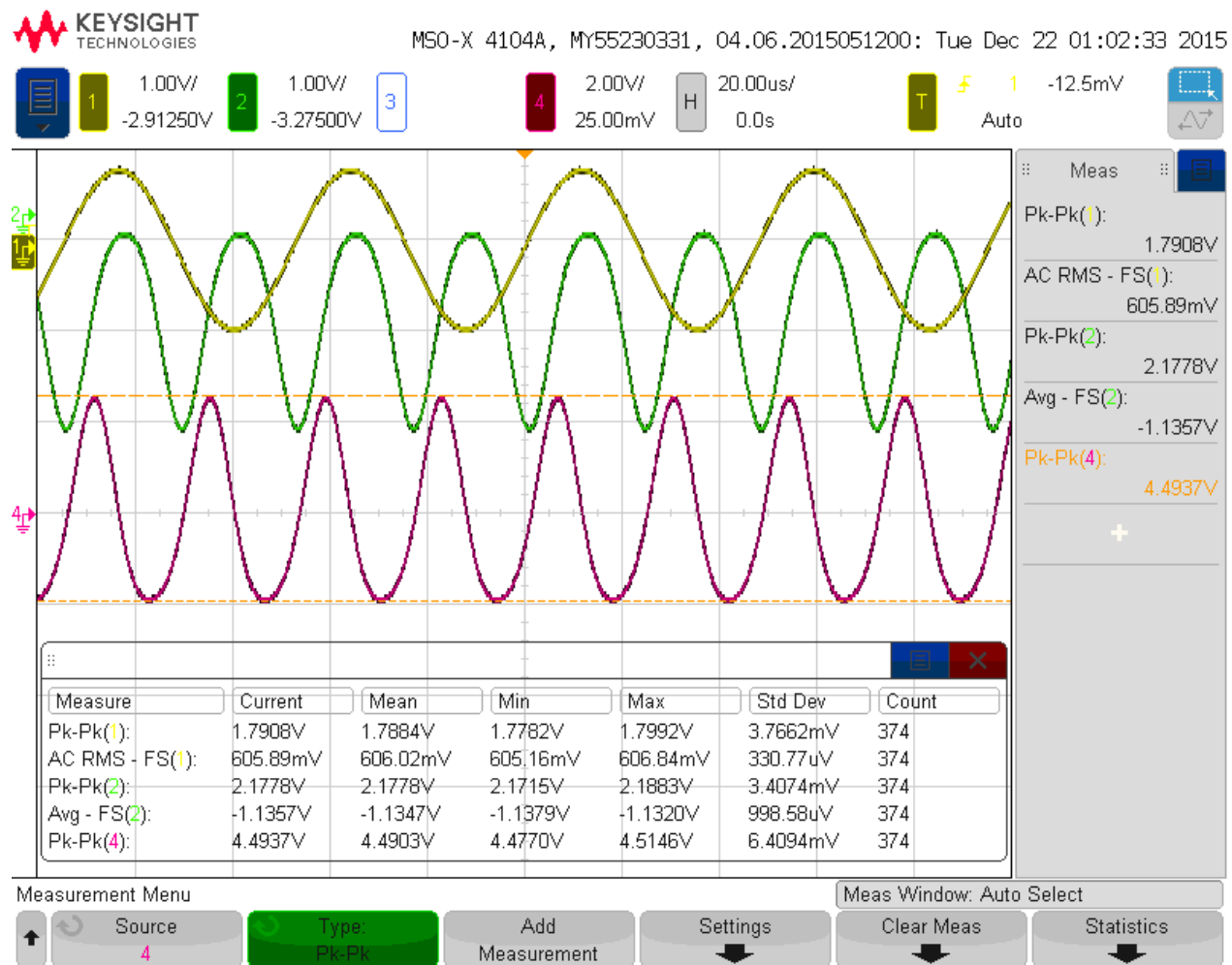
Kanał CH1: Sygnał MOD –sygnał modulujący (moduł MODGEN)

Amplituda sygnału optycznego zależy od poziomu modulacji zadawanego poziomem napięcia z modułu MODGEN. Dla sprawdzenia użyto sygnału sinusoidalnego co pozwala na dokładniejszą obserwację zniekształceń i zakłóceń. Nie wpływa to na normalną pracę układu.

Tabela 5. Zmiana poziomu wejściowego sygnału świetlnego w zależności od poziomu modulacji

Gen. sygnału poziom	SIN Vpp	SIN Vrms	GS2 V/V	DC TIA V	AC TIA Vpp	S2 Vpp	Uwagi rys
65535 =1/1	1.7884	0.606	2	-1.1347	2.1778	4.4903	Sc27 Sygnał zniekształcony
49152 = 3/4	1.3550	0.454	2	-1.5439	1.6241	3.3380	Sc28
40000	1.1058	0.370	2	-1.7788	1.2087	2.4773	Sc26
32768 = 1/2	0.916	0.303	2	-1.9643	0.876	1.18001	Sc31
16384 = 1/4	0.468	0.152	2	-2.3139	0.506		Sc29
40000	1.1062	0.370	2	-1.8072	1.23	2.5155	Sc30

Wzrost poziomu sygnału modulacji powoduje wzrost poziomu sygnału modulowanego w wejściowym sygnale świetlnym. Jednak powyżej tego poziomu następuje wyraźne zniekształcenie kształtu sygnału wejściowego. Powoduje to degradację jakości pomiarów.



Wniosek

Optymalny poziom sygnału modulacyjnego to ok. 400mVrms (ok. 40000).

5.6.5 Badanie wpływu doboru częstotliwości filtra LPF2

Zastosowany w urządzeniu programowany cyfrowo filtr dolnoprzepustowy LPF1 i LPF2 jest zintegrowany w jednym układzie scalonym na wyjściu wzmacniacza PGA. Ma on maksymalne pasmo 150kHz. Sygnał wejściowy z fotodiody zawiera dominujący sygnał drugiej harmonicznej (42kHz) i użytkowy sygnał pomiarowy pierwszej harmonicznej (21kHz). Ponadto występują dodatkowe sygnały będące zakłóceniami, szczególnie czwarta harmoniczna.

Sc25 150KHz g=2V/V

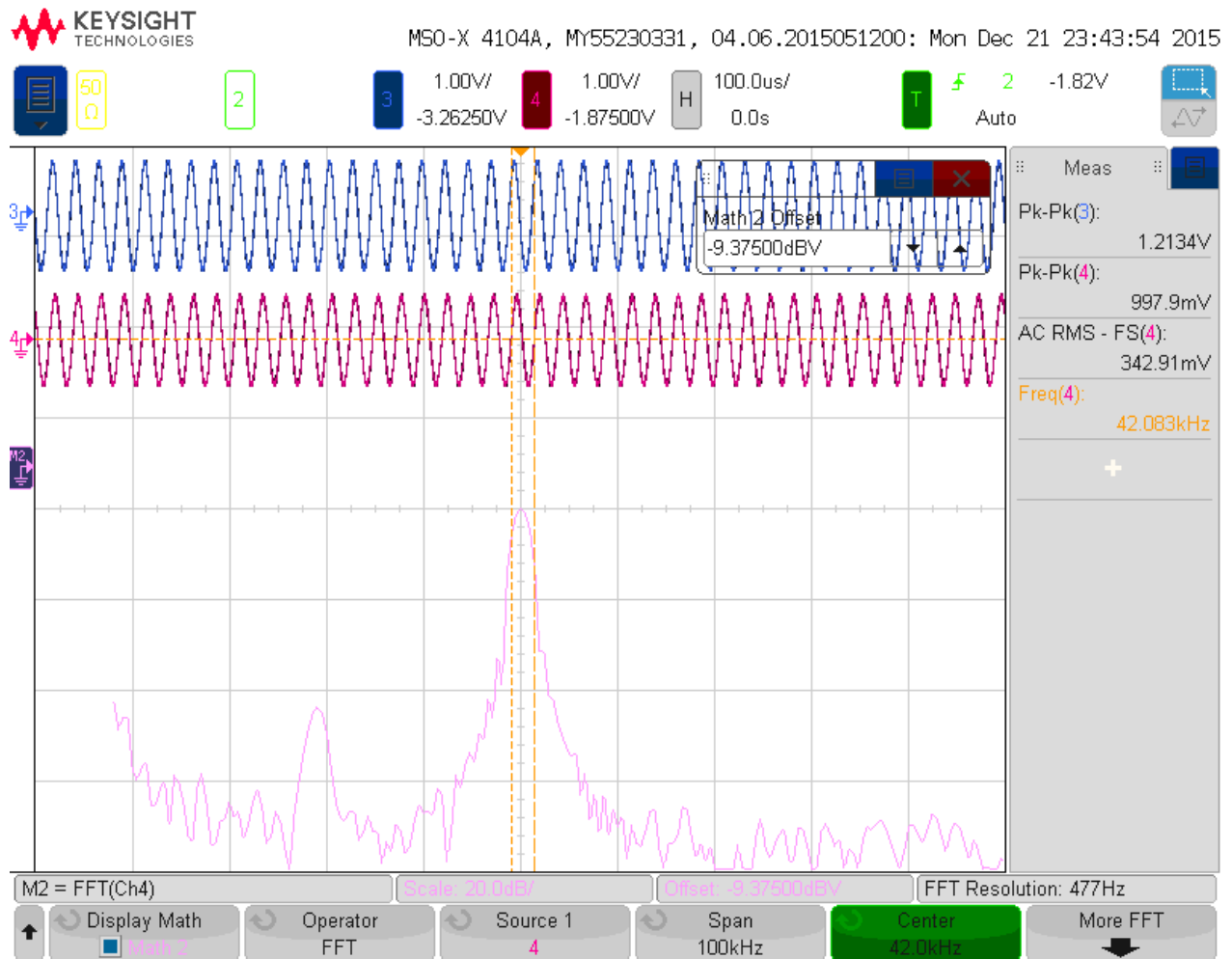


Dla niezakłóconego przenoszenia sygnału pierwszej i drugiej harmonicznej najniższa częstotliwość odcięcia filtra to 50kHz.

LPF2 = 50kHz



Zmniejszenie częstotliwości odcięcia filtra LPF2 do 40kHz powoduje wzrost tłumienia sygnału wejściowego o 2.5dBV.



Zwiększenie częstotliwości odcięcia filtra LPF2 do 60kHz nie powoduje zmian tłumienia sygnału. Nie powoduje też zmiany poziomu czwartej harmonicznej.

Wniosek: Najbardziej odpowiednia częstotliwość odcięcia filtra LPF2 to 50kHz. Podobne zjawisko obserwuje się dla filtra FPF1 (moduł MFLT) zrealizowanego z zastosowaniem takiego samego typu układu scalonego. Jednak optymalna częstotliwość wynosi 30kHz.

5.6.6 Badanie wpływu nasycenia wzmacniacza PGA2

Zastosowany w urządzeniu wzmacniacz programowany cyfrowo z wewnętrznym filtrem dolnoprzepustowym PGA1 i PGA2 ma specyficzny sposób pracy.

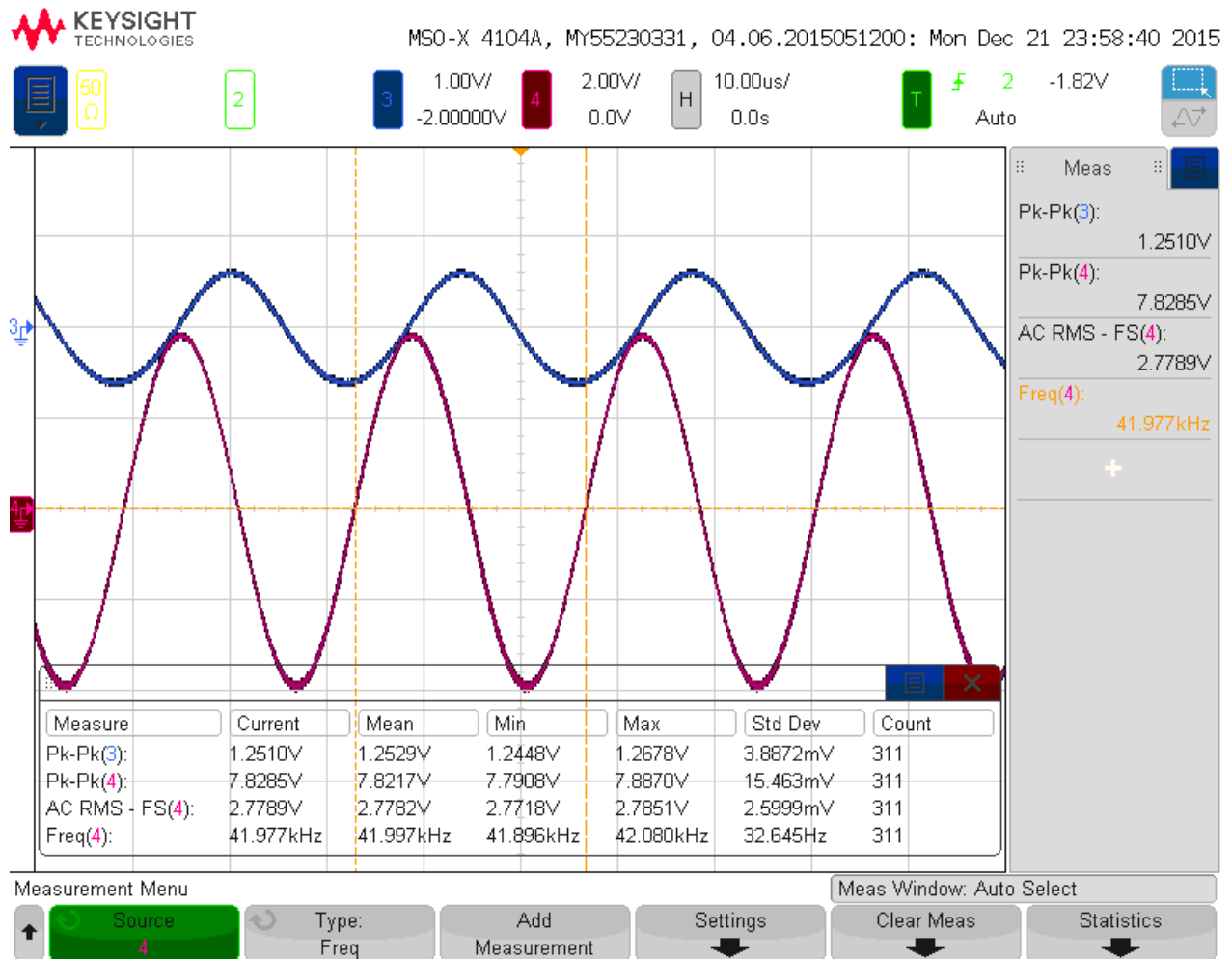
Tabela 6. Zmiana amplitudy sygnału na wyjściu wzmacniacza PGA2

GS2 V/V	BS2 Vpp	S2 Vpp	S2 Vrms	Czas Min	Uwagi rys
------------	------------	-----------	------------	-------------	--------------

1	1.2699	1.32772	0.461	0:00'	Sc16
2	1.2594	2.6265	0.927		Sc17
3	1.2575	3.9608	1.3939		Sc18
4	1.2553	5.2558	1.858		Sc19
5	1.2538	6.5359	2.3177		Sc20
6	1.2529	7.81216	2.7782		Sc21
7	1.2527	8.6121	3.0613		Sc22

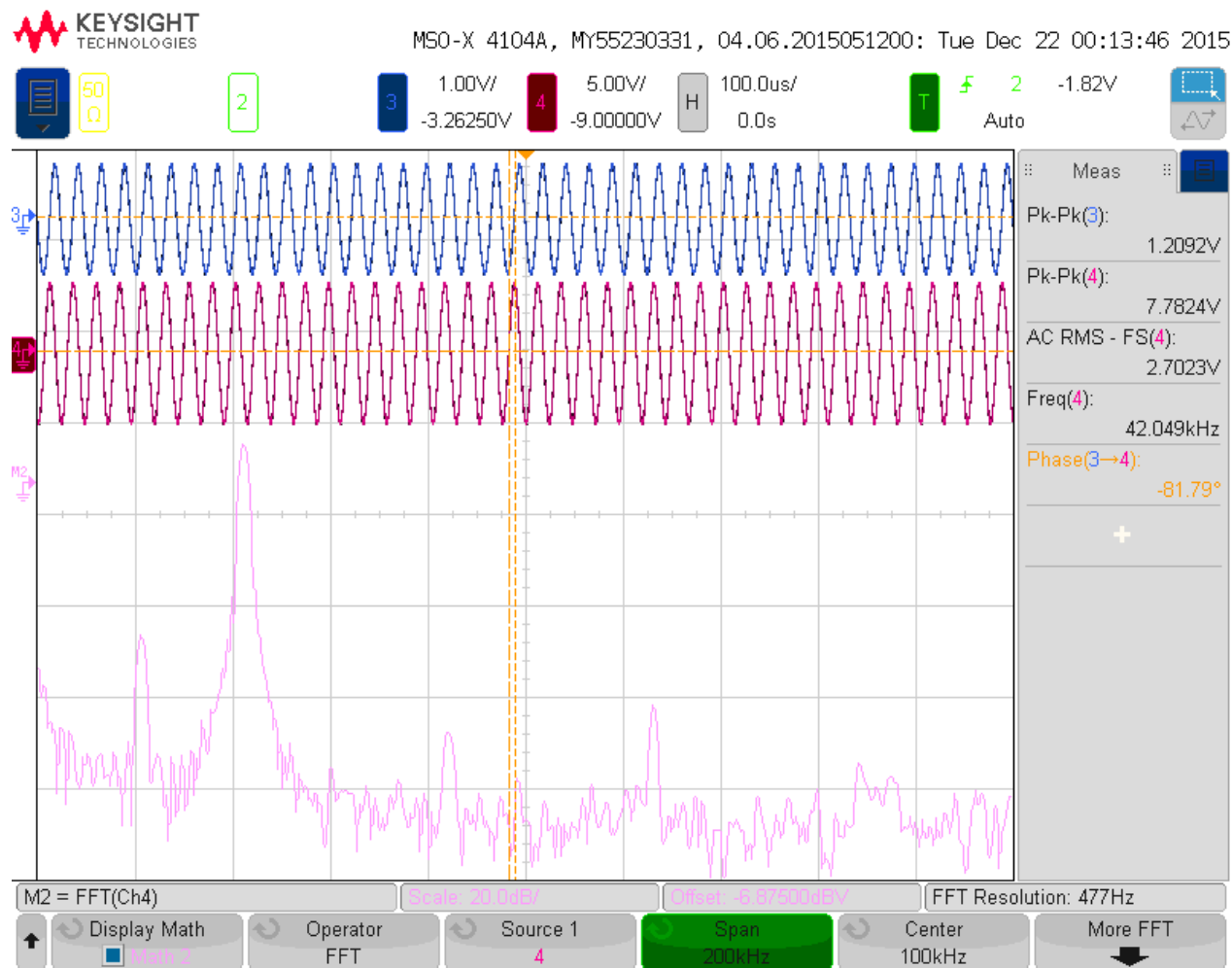
Wzrost amplitudy sygnału na wyjściu wzmacniacza programowanego cyfrowo PGA2 jest liniowy do poziomu sygnału wyjściowego ok. 7Vpp (dla zasilania $\pm 5V$). Dalszy wzrostysterowania powoduje raczej zjawisko ograniczania amplitudy sygnału wyjściowego bez typowego zjawiska nasycania. Nie powoduje to istotnego zniekształcania kształtu sygnału (wzm. 5V/V). Ze wzrostemysterowania występuje też wzrost przesunięcia fazy.

Sc20 GS2=5V/V



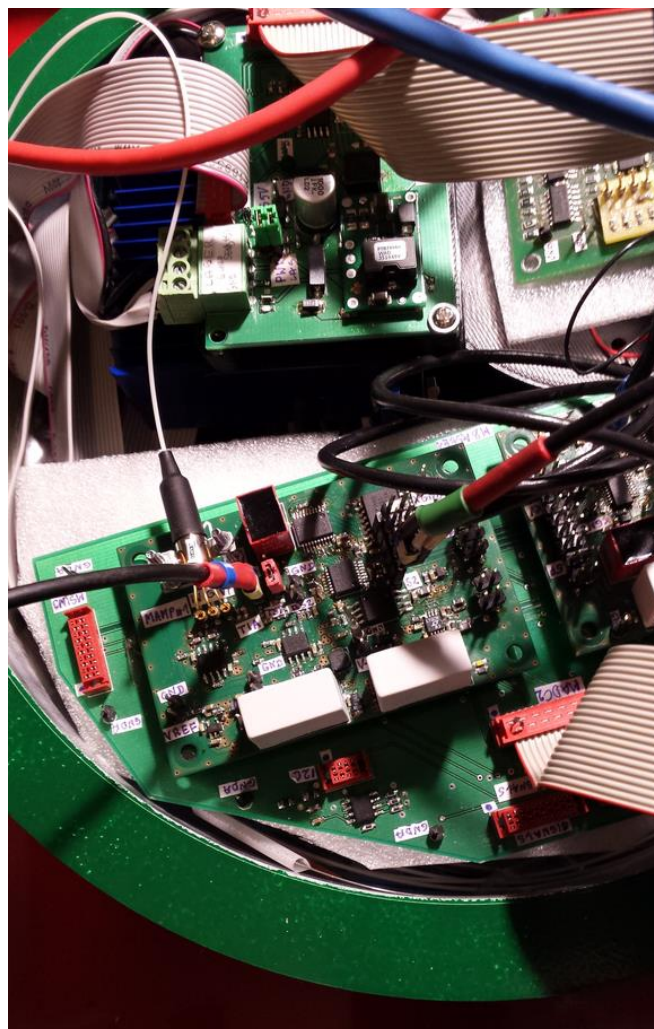
Dopiero dalszy wzrost wysterowania wzmocnienia powoduje wzrost zniekształceń i zwiększenie amplitudy harmonicznych, szczególnie czwartej. Takie zachowanie jest głównie spowodowane zastosowanym na wyjściu układu scalonego wewnętrznym filtrem LPF.

Sc24 Wzmocnienie GS2 = 6V/V

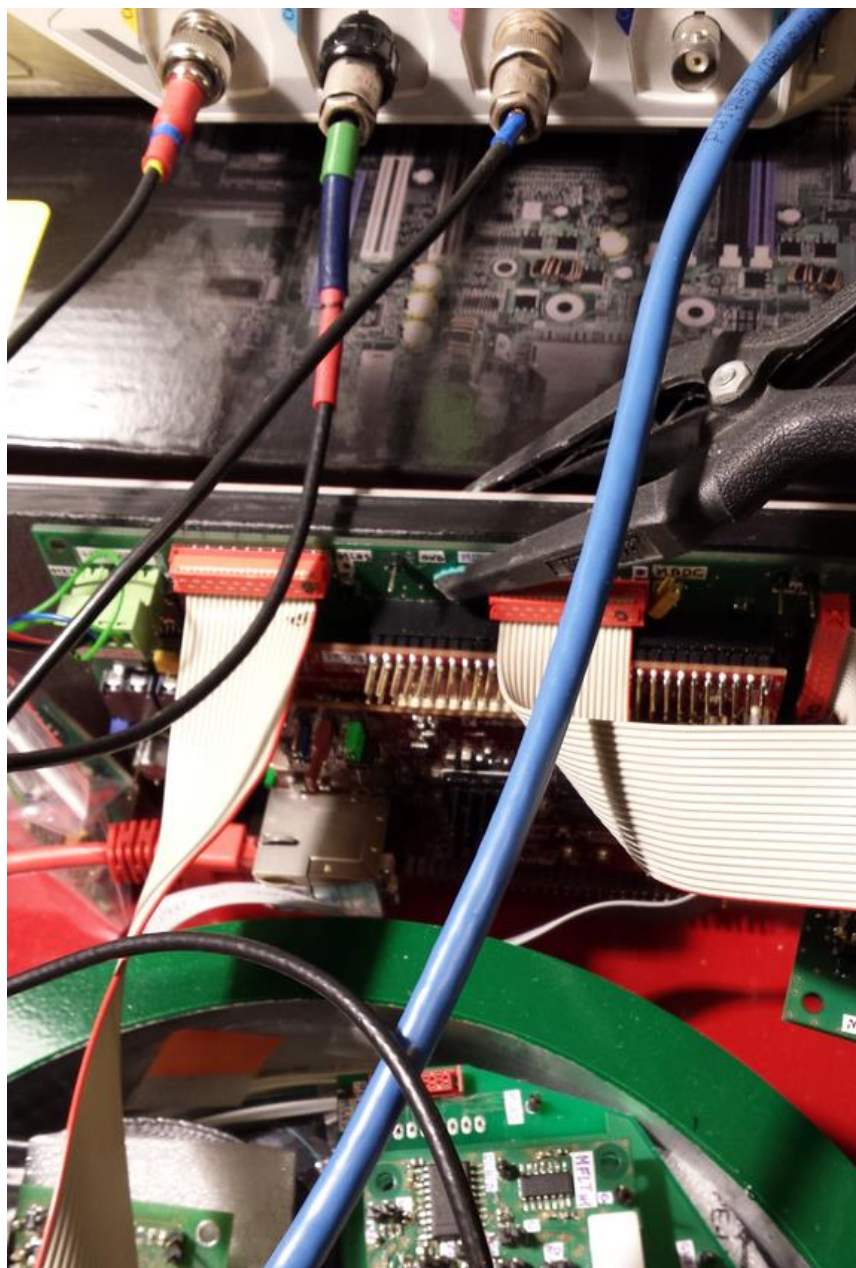


Uzyskane wyniki odnoszą się również do wzmacniacza PGA1 (moduł MFLT).

Poniżej pokazano fotografię układu podczas testów.



Rys. 18. Widok na układ AMP z widocznym białym światłowodem dołączonym do diody detekcyjnej



Rys. 19. Widok na płytkę mikrokontrolera ZEND podłączoną do sieci lokalnej

5.7 Uwagi końcowe

Wszystkie niezbędne pliki zawierające dokumenty produkcyjne, pliki oprogramowania, schematy, projekty płytek drukowanych zostały załączone w postaci elektronicznej na odpowiednich nośnikach danych.

6 Dodatki

6.1 Oprogramowanie FPGA VHD dla obsługi AD7986

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

library unisim;
use unisim.vcomponents.all;

entity ad7986_des is
    port (
        rst_i          : in  std_logic;
        clk_i          : in  std_logic;
        en_i           : in  std_logic; -- start conversion

        clk_spi_i      : in  std_logic;
        clk_cnv_i      : in  std_logic;

        mbase_xsckp    : out std_logic;
        mbase_xsckn    : out std_logic;
        mbase_xcnvp    : out std_logic;
        mbase_xcnvn    : out std_logic;
        mbase_xcklp    : in  std_logic;
        mbase_xckln    : in  std_logic;
        mbase_xdolp    : in  std_logic;
        mbase_xdoln    : in  std_logic;
        mbase_xck2p    : in  std_logic;
        mbase_xck2n    : in  std_logic;
        mbase_xdo2p    : in  std_logic;
        mbase_xdo2n    : in  std_logic;

        mbase_c_xsckp  : out std_logic;
        mbase_c_xsckn  : out std_logic;
        mbase_c_xcnvp  : out std_logic;
        mbase_c_xcnvn  : out std_logic;

        wr_data        : out std_logic_vector(35 downto 0); -- 2chns
        wr_en          : out std_logic;
        smpl_err       : out std_logic;

        cnv_high       : in  std_logic_vector(7 downto 0);
        xfer_time      : in  std_logic_vector(7 downto 0);
        xfer_start     : in  std_logic_vector(7 downto 0);

        dbg           : out std_logic_vector(31 downto 0)
    );
```

```
end ad7986_des;

architecture rtl of ad7986_des is
  component ad7986_core is
    port (
      rst_i      : in  std_logic;
      clk_i      : in  std_logic;
      en_i       : in  std_logic; -- start conversion

      clk_spi_i  : in  std_logic;
      clk_cnv_i  : in  std_logic;

      xsck_o     : out std_logic;
      xcnv_o     : out std_logic;
      xck1_i     : in  std_logic;
      xdo1_i     : in  std_logic;
      xck2_i     : in  std_logic;
      xdo2_i     : in  std_logic;

      xsckc_o    : out std_logic;

      wr_data    : out std_logic_vector(35 downto 0); -- 2chns
      wr_en      : out std_logic;
      smpl_err   : out std_logic;

      cnv_high   : in  std_logic_vector(7 downto 0);
      xfer_time  : in  std_logic_vector(7 downto 0);
      xfer_start : in  std_logic_vector(7 downto 0);

      dbg        : out std_logic_vector(31 downto 0)
    );
  end component ad7986_core;

  signal xsck : std_logic;
  signal xcnv : std_logic;
  signal xck1 : std_logic;
  signal xdo1 : std_logic;
  signal xck2 : std_logic;
  signal xdo2 : std_logic;
  signal xsckc : std_logic;
begin

  core: ad7986_core
    port map (
      rst_i => rst_i,
      clk_i => clk_i,
      en_i  => en_i,

      clk_spi_i => clk_spi_i,
      clk_cnv_i => clk_cnv_i,

      xsck_o => xsck,
```

```
    xcnv_o => xcnv,
    xck1_i => xck1,
    xdo1_i => xdo1,
    xck2_i => xck2,
    xdo2_i => xdo2,

    xsckc_o => xsckc,

    wr_data => wr_data,
    wr_en   => wr_en,
    smpl_err => smpl_err,

    cnv_high   => cnv_high,
    xfer_time  => xfer_time,
    xfer_start => xfer_start,

    dbg => dbg
);

xsckds: OBUFDS
  port map (
    O => mbase_xsckp,
    OB => mbase_xsckn,
    I  => xsck
  );

xcnvds: OBUFDS
  port map (
    O => mbase_xcnvp,
    OB => mbase_xcnvn,
    I  => xcnv
  );

xcklds: IBUFDS
  generic map (
    DIFF_TERM   => TRUE,
    IBUF_LOW_PWR => FALSE
  )
  port map (
    I  => mbase_xcklp,
    IB => mbase_xckln,
    O  => xck1
  );

xdolds: IBUFDS
  generic map (
    DIFF_TERM   => TRUE,
    IBUF_LOW_PWR => FALSE
  )
  port map (
```

```
        I  => mbase_xdo1p,  
        IB => mbase_xdo1n,  
        O  => xdo1  
    );  
  
xck2ds: IBUFDS  
    generic map (  
        DIFF_TERM    => TRUE,  
        IBUF_LOW_PWR => FALSE  
    )  
    port map (  
        I  => mbase_xck2p,  
        IB => mbase_xck2n,  
        O  => xck2  
    );  
  
xdo2ds: IBUFDS  
    generic map (  
        DIFF_TERM    => TRUE,  
        IBUF_LOW_PWR => FALSE  
    )  
    port map (  
        I  => mbase_xdo2p,  
        IB => mbase_xdo2n,  
        O  => xdo2  
    );  
  
xsckdsc: OBUFDS  
    port map (  
        O => mbase_c_xsckp,  
        OB => mbase_c_xsckn,  
        I  => xsckc  
    );  
  
xcnvdsc: OBUFDS  
    port map (  
        O  => mbase_c_xcnvp,  
        OB => mbase_c_xcnvn,  
        I  => xcnv  
    );  
  
end rtl;  
-- vim: set ts=3 expandtab:
```

6.2 Program FPGA VHD dla obsługi MBASE

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity mbase is
  port (
    S_AXI_ACLK      : in  std_logic;
    S_AXI_ARESETN   : in  std_logic;

    S_AXI_AWADDR     : in  std_logic_vector(15 downto 0);
    S_AXI_AWPROT     : in  std_logic_vector(2 downto 0);
    S_AXI_AWVALID    : in  std_logic;
    S_AXI_AWREADY    : out std_logic;

    S_AXI_WDATA      : in  std_logic_vector(31 downto 0);
    S_AXI_WSTRB      : in  std_logic_vector(3 downto 0);
    S_AXI_WVALID     : in  std_logic;
    S_AXI_WREADY     : out std_logic;
    S_AXI_BRESP      : out std_logic_vector(1 downto 0);
    S_AXI_BVALID     : out std_logic;
    S_AXI_BREADY     : in  std_logic;

    S_AXI_ARADDR     : in  std_logic_vector(15 downto 0);
    S_AXI_ARPROT     : in  std_logic_vector(2 downto 0);
    S_AXI_ARVALID    : in  std_logic;
    S_AXI_ARREADY    : out std_logic;

    S_AXI_RDATA      : out std_logic_vector(31 downto 0);
    S_AXI_RRESP      : out std_logic_vector(1 downto 0);
    S_AXI_RVALID     : out std_logic;
    S_AXI_RREADY     : in  std_logic;

    -- laser module
    mlas_fldin_o      : out std_logic;
    mlas_flsclk_o     : out std_logic;
    mlas_flsync_o     : out std_logic;
    mlas_fslen_o      : out std_logic;
    mlas_fainen_o     : out std_logic;
    mlas_fsl_end_i    : in  std_logic;
    mlas_fain_end_i   : in  std_logic;
    mlas_fain_al_i    : in  std_logic;
    mlas_ftec_al_i    : in  std_logic;
    mlas_flas_on_o    : out std_logic; -- pwr

    -- mbase
    mbase_fxsi_o      : out std_logic;
    mbase_fxscclk_o   : out std_logic;
    mbase_fxcsa_o     : out std_logic;
```

```

    mbase_fxcsb_o      : out std_logic;
    mbase_fxcsb_o      : out std_logic;
    mbase_fxsigoff_o   : out std_logic;

    -- etc
    metc_an_on_o       : out std_logic;
    metc_unii_o        : out std_logic;

    o_dbg              : out std_logic_vector(31 downto 0)
);
end mbase;

```

architecture rtl of mbase is

```

-- AXI4LITE signals
signal axi_awready : std_logic;
signal axi_wready  : std_logic;
signal axi_bvalid  : std_logic;
signal axi_arready : std_logic;
signal axi_rdata   : std_logic_vector(31 downto 0);
signal axi_rvalid  : std_logic;

signal reg_data_out : std_logic_vector(31 downto 0);

---- Number of Slave Registers 4
-- day: 5 bits = 0-31
-- month: 4 bits = 0-15
-- year: 4 bits = 0-15 (+2015)
-- major: 3 bits = 0-7
-- minor: 7 bits = 0-127
-- ----- 23 bits
-- 1001 - magic = 9h
-- -----
constant CORE_VERSION_ID : std_logic_vector(4 downto 0) :=
    std_logic_vector(to_unsigned(10, 5));
constant CORE_VERSION_MAJOR : std_logic_vector(2 downto 0) :=
    std_logic_vector(to_unsigned(0, 3));
constant CORE_VERSION_MINOR : std_logic_vector(6 downto 0) :=
    std_logic_vector(to_unsigned(1, 7));
constant CORE_VERSION_DAY : std_logic_vector(4 downto 0) :=
    std_logic_vector(to_unsigned(14, 5));
constant CORE_VERSION_MONTH : std_logic_vector(3 downto 0) :=
    std_logic_vector(to_unsigned(11, 4));
constant CORE_VERSION_YEAR : std_logic_vector(3 downto 0) :=
    std_logic_vector(to_unsigned(00, 4)); -- 2015 + 0 = 2015

constant CORE_VERSION : std_logic_vector(31 downto 0) :=
    "1001" & CORE_VERSION_ID & CORE_VERSION_MAJOR & CORE_VERSION_MINOR &
    CORE_VERSION_DAY & CORE_VERSION_MONTH & CORE_VERSION_YEAR;

constant OFF_VERSION : integer := 0;
constant OFF_CTRL : integer := 1;

```

```

constant OFF_MLAS      : integer := 2;
constant OFF_MLAS_RB   : integer := 3;
constant OFF_MBASE     : integer := 4;

--
-- rst      - 0
-- an_on    - 1
-- unii_o    - 2
signal ctrl_reg_def : std_logic_vector
    (31 downto 0) := x"0000000" & "0001";
signal ctrl_reg      : std_logic_vector(31 downto 0) := ctrl_reg_def;

----- MLAS -----
-- 0 - fldin
-- 1 - flsclk
-- 2 - flsync
-- 3 - fslen
-- 4 - fainen
-- 5 - flas_on
signal mlas_reg_def : std_logic_vector
    (31 downto 0) := x"0000000" & "00100100";
signal mlas_reg      : std_logic_vector(31 downto 0) := mlas_reg_def;

-- 0 - fsl_end
-- 1 - fain_end
-- 2 - fain_al
-- 3 - ftec_al
signal mlas_rb_reg   : std_logic_vector(31 downto 0) := (others => '0');

----- MBASE -----
-- 0 - fxsi
-- 1 - fxscclk
-- 2 - fxcsa
-- 3 - fxcsb
-- 4 - fxscs
-- 5 - fxsigoff
signal mbase_reg_def : std_logic_vector
    (31 downto 0) := x"0000000" & "00011100";
signal mbase_reg      : std_logic_vector(31 downto 0) := mbase_reg_def;

begin

-- I/O Connections assignments
S_AXI_AWREADY <= axi_awready;
S_AXI_WREADY  <= axi_wready;
S_AXI_BRESP   <= "00";
S_AXI_BVALID  <= axi_bvalid;
S_AXI_ARREADY <= axi_arready;
S_AXI_RDATA   <= axi_rdata;
S_AXI_RRESP   <= "00";

```

```
S_AXI_RVALID    <= axi_rvalid;

process (S_AXI_ACLK)
begin
    if rising_edge(S_AXI_ACLK) then
        if S_AXI_ARESETN = '0' then
            axi_awready <= '0';
            axi_wready  <= '0';
            axi_bvalid  <= '0';

            ctrl_reg <= ctrl_reg_def;
            mlas_reg <= mlas_reg_def;
            mbase_reg <= mbase_reg_def;
        else
            -- reset core
            if ctrl_reg(0) = '1' then
                ctrl_reg <= ctrl_reg_def;
            end if;

            if S_AXI_AWVALID = '1' and S_AXI_WVALID = '1' and
                axi_awready = '0' and axi_wready = '0' and
                S_AXI_WSTRB = x"f" then

                case to_integer(unsigned(
                    S_AXI_AWADDR(S_AXI_AWADDR'length - 1 downto 2)))
                is
                    when OFF_CTRL =>
                        ctrl_reg <= S_AXI_WDATA;

                    when OFF_MLAS =>
                        mlas_reg <= S_AXI_WDATA;

                    when OFF_MBASE =>
                        mbase_reg <= S_AXI_WDATA;

                    when others => null;
                end case;

                axi_awready <= '1';
                axi_wready  <= '1';
            else
                axi_awready <= '0';
                axi_wready  <= '0';
            end if;

            if axi_awready = '1' and S_AXI_AWVALID = '1' and
                axi_wready = '1' and S_AXI_WVALID = '1' and
                axi_bvalid = '0' then
                axi_bvalid <= '1';
            elsif S_AXI_BREADY = '1' and axi_bvalid = '1' then
                axi_bvalid <= '0';
            end if;
        end if;
    end if;
end process;
```



```

        end if;
    end if;
end process;

process (S_AXI_ACLK)
begin
    if rising_edge(S_AXI_ACLK) then
        if S_AXI_ARESETN = '0' then
            reg_data_out <= (others => '0');
            axi_arready <= '0';
            axi_rvalid <= '0';
            axi_rdata <= (others => '0');
        else
            if S_AXI_ARVALID = '1' and axi_arready = '0' then
                reg_data_out <= (others => '0');

                case to_integer(unsigned(
                    S_AXI_ARADDR(S_AXI_ARADDR'length - 1 downto 2)))
                is
                    when OFF_VERSION =>
                        reg_data_out <= CORE_VERSION;

                    when OFF_CTRL =>
                        reg_data_out <= ctrl_reg;

                    when OFF_MLAS =>
                        reg_data_out <= mlas_reg;

                    when OFF_MLAS_RB =>
                        reg_data_out <= mlas_rb_reg;

                    when OFF_MBASE =>
                        reg_data_out <= mbase_reg;

                    when others => null;
                end case;

                axi_arready <= '1';
            else
                axi_arready <= '0';
            end if;

            if S_AXI_ARVALID = '1' and axi_arready = '1' and
                axi_rvalid = '0' then
                axi_rdata <= reg_data_out;
                axi_rvalid <= '1';
            elsif axi_rvalid = '1' and S_AXI_RREADY = '1' then
                axi_rvalid <= '0';
            end if;
        end if;
    end if;
end process;

```

```

        end if;
    end process;

    -----
    --      LASER
    -----

    metc_an_on_o  <= ctrl_reg(1);
    metc_unii_o   <= ctrl_reg(2);

    mlas_fldin_o  <= mlas_reg(0);
    mlas_flsclk_o <= mlas_reg(1);
    mlas_flsync_o <= mlas_reg(2);
    mlas_fslen_o  <= mlas_reg(3);
    mlas_fainen_o <= mlas_reg(4);
    mlas_flas_on_o <= mlas_reg(5);

    mlas_rb_reg(0) <= mlas_fsl_end_i;
    mlas_rb_reg(1) <= mlas_fain_end_i;
    mlas_rb_reg(2) <= mlas_fain_al_i;
    mlas_rb_reg(3) <= mlas_ftec_al_i;

    mbase_fxsi_o    <= mbase_reg(0);
    mbase_fxscclk_o <= mbase_reg(1);
    mbase_fxcsa_o   <= mbase_reg(2);
    mbase_fxcsb_o   <= mbase_reg(3);
    mbase_fxcscl_o  <= mbase_reg(4);
    mbase_fxsigoff_o <= mbase_reg(5);

end rtl;
-- vim: set ts=3 expandtab:

```

6.3 Program generacji sygnałów zegarowych

```

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity mclk is
    port (
        clk_i : in  std_logic;
        rst_i : in  std_logic;

        cnv    : out std_logic;
        sinf1   : out std_logic;
        sinf2   : out std_logic;

        sw1     : in  std_logic;
        sw2     : in  std_logic
    );
end mclk;

```

```
architecture rtl of mclk is
begin
    process (clk_i, rst_i)
        variable i : integer range 0 to 3 := 0;
        variable x64 : boolean := false;
        variable x32 : boolean := false;
    begin
        if rst_i = '0' then -- fclk_0_n
            cnv <= '0';
            sinfl <= '0';
            sinf2 <= '0';

            i := 0;
            x64 := false;
            x32 := false;
        elsif rising_edge(clk_i) then
            if i = 3 then
                if x64 = true then
                    x64 := false; -- 64x
                else
                    x64 := true; -- 64x
                    if x32 = true then
                        x32 := false;
                    else
                        x32 := true;
                    end if;
                end if;
            end if;

            if x32 = true then
                cnv <= '1';
            else
                cnv <= '0';
            end if;

            if sw1 = '0' then
                if x32 = true then
                    sinfl <= '1';
                else
                    sinfl <= '0';
                end if;
            else
                if x64 = true then
                    sinfl <= '1';
                else
                    sinfl <= '0';
                end if;
            end if;

            if sw2 = '0' then
                if x32 = true then
```

```

        sinf2 <= '1';
    else
        sinf2 <= '0';
    end if;
else
    if x64 = true then
        sinf2 <= '1';
    else
        sinf2 <= '0';
    end if;
end if;

    i := 0;
else
    i := i + 1;
end if;
end if;
end process;
end rtl;
-- vim: set ts=3 expandtab:

```

6.4 Program FPGA dla obsługi generatora modulacji

```

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity singenv1 is
    port (
        S_AXI_ACLK      : in  std_logic;
        S_AXI_ARESETN   : in  std_logic;

        S_AXI_AWADDR     : in  std_logic_vector(15 downto 0);
        S_AXI_AWPROT     : in  std_logic_vector(2 downto 0);
        S_AXI_AWVALID    : in  std_logic;
        S_AXI_AWREADY    : out std_logic;

        S_AXI_WDATA      : in  std_logic_vector(31 downto 0);
        S_AXI_WSTRB      : in  std_logic_vector(3 downto 0);
        S_AXI_WVALID     : in  std_logic;
        S_AXI_WREADY     : out std_logic;
        S_AXI_BRESP      : out std_logic_vector(1 downto 0);
        S_AXI_BVALID     : out std_logic;
        S_AXI_BREADY     : in  std_logic;

        S_AXI_ARADDR     : in  std_logic_vector(15 downto 0);
        S_AXI_ARPROT     : in  std_logic_vector(2 downto 0);
        S_AXI_ARVALID    : in  std_logic;
        S_AXI_ARREADY    : out std_logic;

        S_AXI_RDATA      : out std_logic_vector(31 downto 0);
        S_AXI_RRESP      : out std_logic_vector(1 downto 0);
    );
end entity singenv1;

```

```

    S_AXI_RVALID    : out std_logic;
    S_AXI_RREADY    : in  std_logic;

    i_sinf32        : in  std_logic; -- pass-through
    o_sinfsw        : out std_logic;

    msinvl_fsinf32_o  : out std_logic;
    msinvl_fsinsync_o : out std_logic;
    msinvl_fsincl_k_o : out std_logic;
    msinvl_fsinsdi_o  : out std_logic;
    msinvl_fsincs_o   : out std_logic;
    msinvl_fsinpwrn_o : out std_logic;

    o_dbg           : out std_logic_vector(31 downto 0)
);
end singenv1;

```

architecture rtl of singenv1 is

```

-- AXI4LITE signals
signal axi_awready : std_logic;
signal axi_wready  : std_logic;
signal axi_bvalid  : std_logic;
signal axi_arready : std_logic;
signal axi_rdata   : std_logic_vector(31 downto 0);
signal axi_rvalid  : std_logic;

signal reg_data_out : std_logic_vector(31 downto 0);

---- Number of Slave Registers 4
-- day: 5 bits = 0-31
-- month: 4 bits = 0-15
-- year: 4 bits = 0-15 (+2015)
-- major: 3 bits = 0-7
-- minor: 7 bits = 0-127
-- ----- 23 bits
-- 1001 - magic = 9h
-- -----
constant CORE_VERSION_ID : std_logic_vector(4 downto 0) :=
    std_logic_vector(to_unsigned(20, 5));
constant CORE_VERSION_MAJOR : std_logic_vector(2 downto 0) :=
    std_logic_vector(to_unsigned(0, 3));
constant CORE_VERSION_MINOR : std_logic_vector(6 downto 0) :=
    std_logic_vector(to_unsigned(1, 7));
constant CORE_VERSION_DAY : std_logic_vector(4 downto 0) :=
    std_logic_vector(to_unsigned(15, 5));
constant CORE_VERSION_MONTH : std_logic_vector(3 downto 0) :=
    std_logic_vector(to_unsigned(11, 4));
constant CORE_VERSION_YEAR : std_logic_vector(3 downto 0) :=
    std_logic_vector(to_unsigned(00, 4)); -- 2015 + 0 = 2015

```

```

constant CORE_VERSION : std_logic_vector(31 downto 0) :=
    "1001" & CORE_VERSION_ID & CORE_VERSION_MAJOR & CORE_VERSION_MINOR &
    CORE_VERSION_DAY & CORE_VERSION_MONTH & CORE_VERSION_YEAR;

constant OFF_VERSION : integer := 0;
constant OFF_CTRL    : integer := 1;

--
-- rst      - 0
-- sync     - 1
-- clk      - 2
-- sdi      - 3
-- cs       - 4
-- pwron    - 5
-- sinfsw   - 6
--
signal ctrl_reg_def : std_logic_vector
    (31 downto 0) := x"000000" & "00010001";
signal ctrl_reg     : std_logic_vector(31 downto 0) := ctrl_reg_def;

begin

    -- I/O Connections assignments
    S_AXI_AWREADY <= axi_awready;
    S_AXI_WREADY  <= axi_wready;
    S_AXI_BRESP   <= "00";
    S_AXI_BVALID  <= axi_bvalid;
    S_AXI_ARREADY <= axi_arready;
    S_AXI_RDATA   <= axi_rdata;
    S_AXI_RRESP   <= "00";
    S_AXI_RVALID  <= axi_rvalid;

    process (S_AXI_ACLK)
    begin
        if rising_edge(S_AXI_ACLK) then
            if S_AXI_ARESETN = '0' then
                axi_awready <= '0';
                axi_wready  <= '0';
                axi_bvalid  <= '0';

                ctrl_reg <= ctrl_reg_def;
            else
                -- reset core
                if ctrl_reg(0) = '1' then
                    ctrl_reg <= ctrl_reg_def;
                end if;

                if S_AXI_AWVALID = '1' and S_AXI_WVALID = '1' and
                    axi_awready = '0' and axi_wready = '0' and
                    S_AXI_WSTRB = x"f" then

```

```

        case to_integer(unsigned(
            S_AXI_AWADDR(S_AXI_AWADDR'length - 1 downto 2)))
        is
            when OFF_CTRL =>
                ctrl_reg <= S_AXI_WDATA;

            when others => null;
        end case;

        axi_awready <= '1';
        axi_wready  <= '1';
    else
        axi_awready <= '0';
        axi_wready <= '0';
    end if;

    if axi_awready = '1' and S_AXI_AWVALID = '1' and
        axi_wready = '1' and S_AXI_WVALID = '1' and
        axi_bvalid = '0' then
        axi_bvalid <= '1';
    elsif S_AXI_BREADY = '1' and axi_bvalid = '1' then
        axi_bvalid <= '0';
    end if;
end if;
end if;
end process;

process (S_AXI_ACLK)
begin
    if rising_edge(S_AXI_ACLK) then
        if S_AXI_ARESETN = '0' then
            reg_data_out <= (others => '0');
            axi_arready  <= '0';
            axi_rvalid   <= '0';
            axi_rdata    <= (others => '0');
        else
            if S_AXI_ARVALID = '1' and axi_arready = '0' then
                reg_data_out <= (others => '0');

                case to_integer(unsigned(
                    S_AXI_ARADDR(S_AXI_ARADDR'length - 1 downto 2)))
                is
                    when OFF_VERSION =>
                        reg_data_out <= CORE_VERSION;

                    when OFF_CTRL =>
                        reg_data_out <= ctrl_reg;

                    when others => null;
                end case;
            end if;
        end if;
    end process;
end process;

```

```
        axi_arready <= '1';
    else
        axi_arready <= '0';
    end if;

    if S_AXI_ARVALID = '1' and axi_arready = '1' and
        axi_rvalid = '0' then
        axi_rdata <= reg_data_out;
        axi_rvalid <= '1';
    elsif axi_rvalid = '1' and S_AXI_RREADY = '1' then
        axi_rvalid <= '0';
    end if;
end if;
end if;
end process;

msinvl_fsinf32_o    <= i_sinf32;
msinvl_fsinsync_o   <= ctrl_reg(1);
msinvl_fsinclclk_o  <= ctrl_reg(2);
msinvl_fsinsdi_o    <= ctrl_reg(3);
msinvl_fsincs_o     <= ctrl_reg(4);
msinvl_fsinpwrn_o   <= ctrl_reg(5);
o_sinfsw            <= ctrl_reg(6);

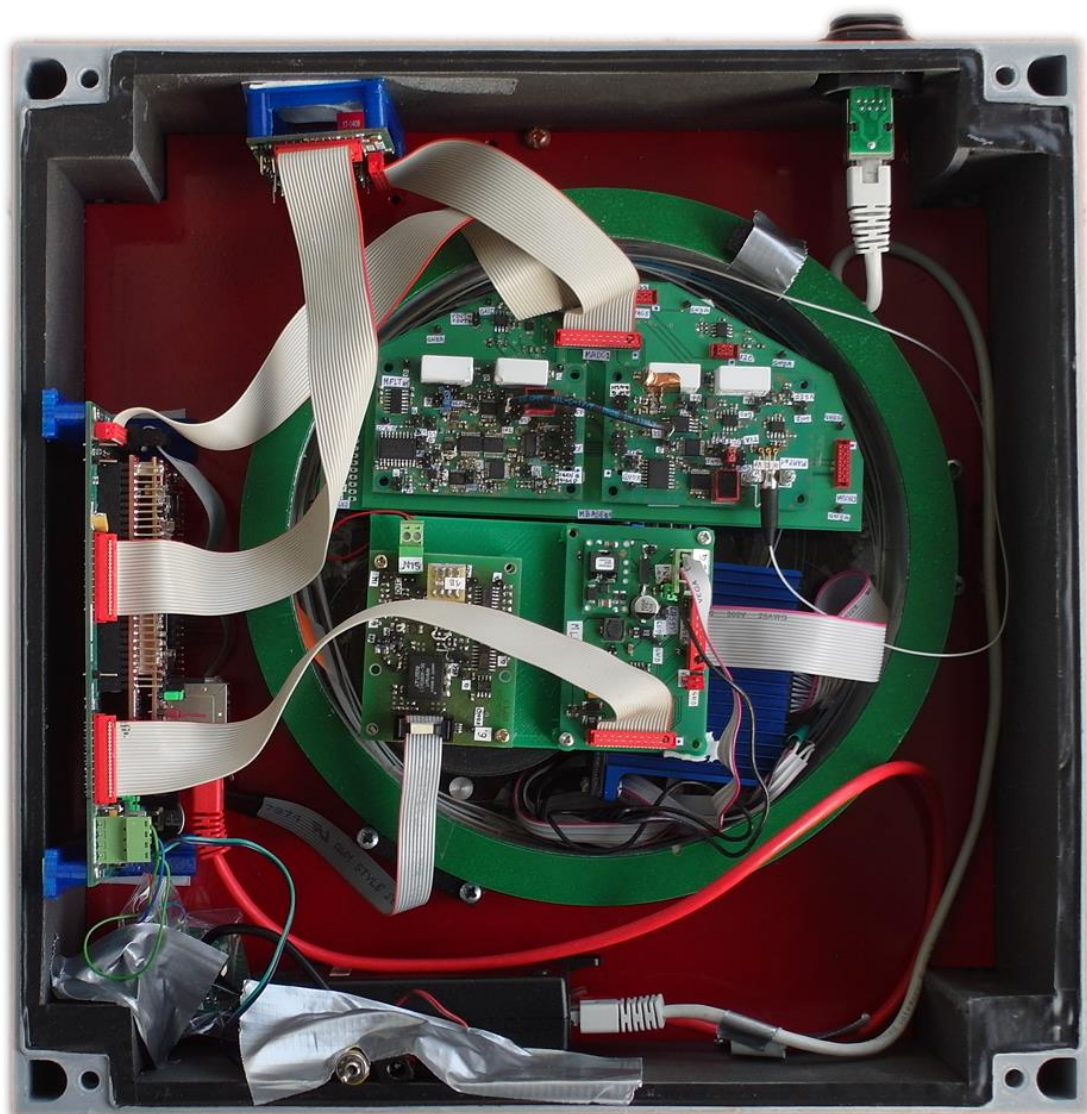
end rtl;
-- vim: set ts=3 expandtab:
```


6.5 Fotografie prototypu

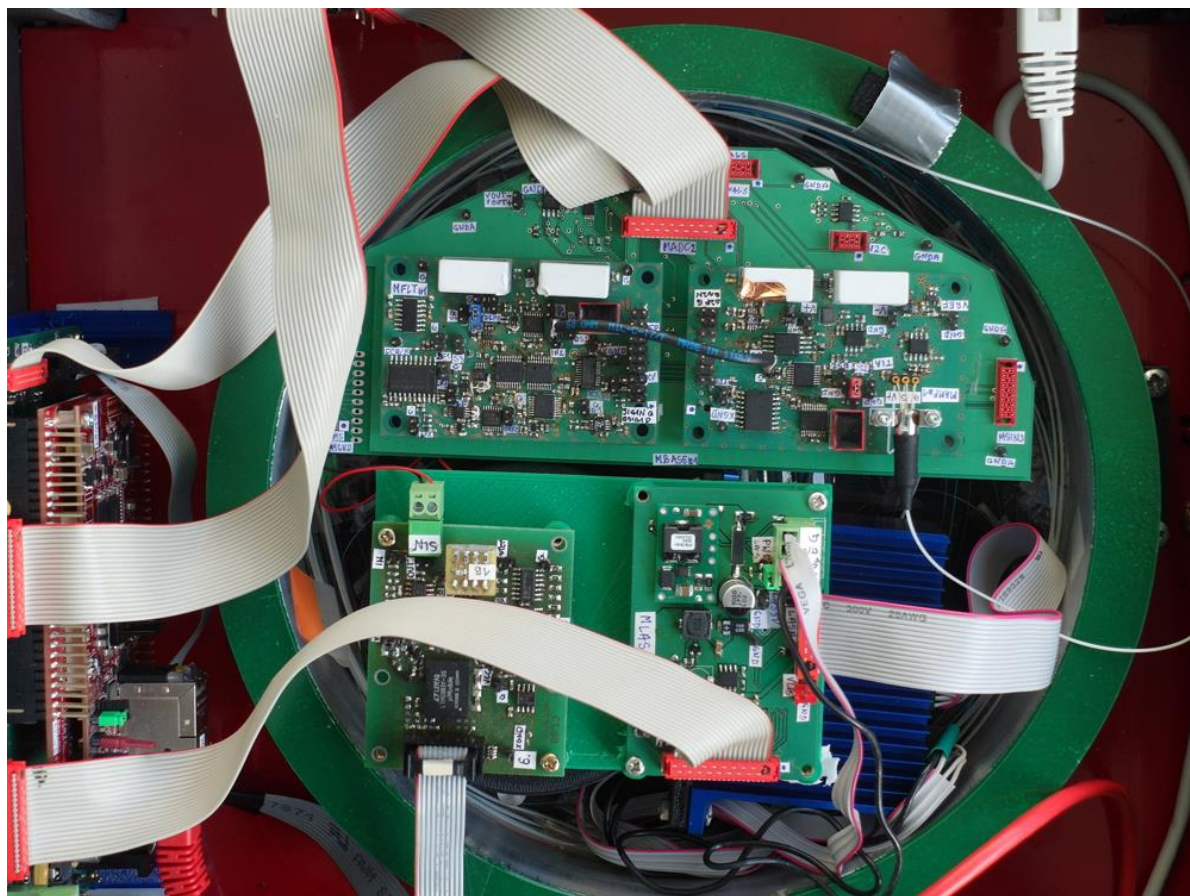


Rys. 20. Widok prototypu gotowego do pracy

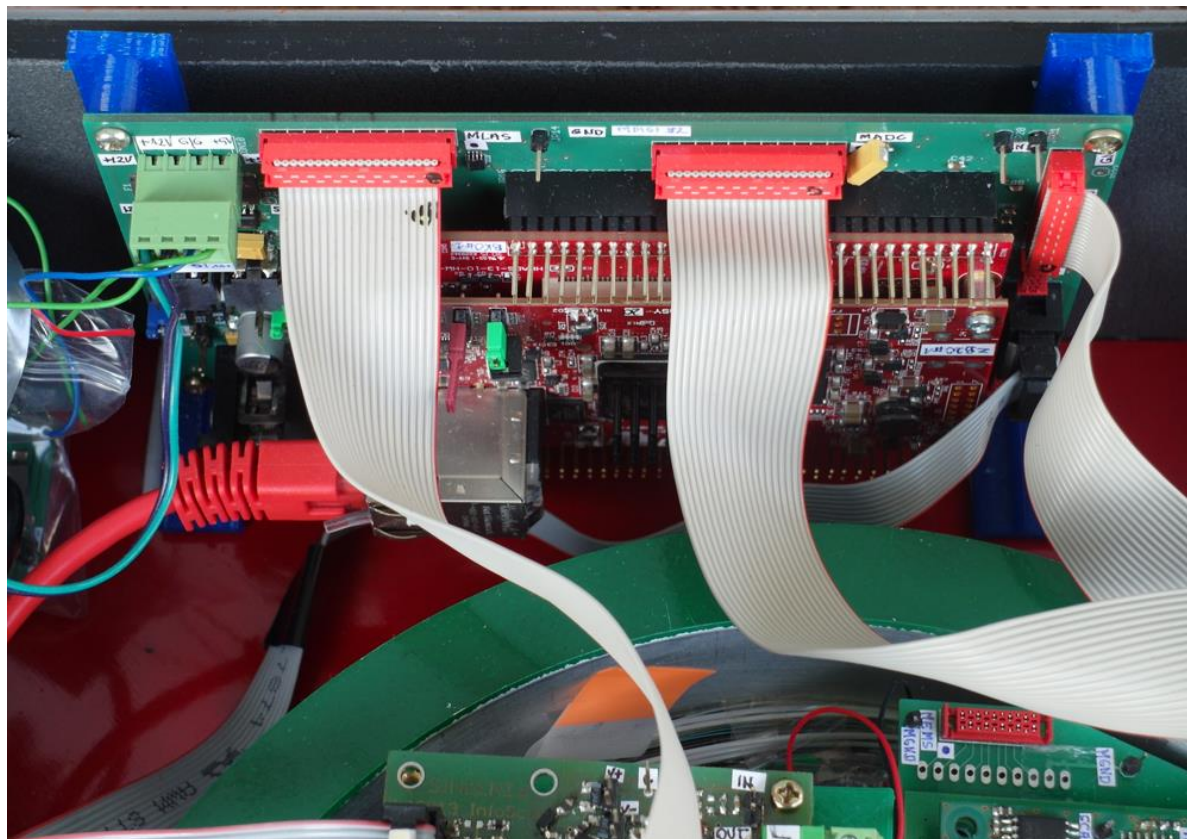
Gniazdo RJ-45 przeznaczone
do transmisji danych i
zasilania



Rys. 21. Widok wnętrza prototypu



Rys. 22. Część elektroniczna z widoczną pętlą światłowodową



Rys. 23. Płytką mikrokomputera z układami towarzyszącymi