AKADEMIA GÓRNICZO-HUTNICZA

Raport z projektu

Miernik mocy do zastosowań LV

z przedmiotu

Sensory w Aplikacjach Wbudowanych

Elektronika i Telekomunikacja - Systemy Wbudowane, rok I studiów magisterskich

Mikołaj Pichita

Arkadiusz Rapacz

Tomasz Szydłak

14.06.2024r

Spis treści

1 Opis i założenie projektu

Celem projektu jest stworzenie urządzenia będącego w stanie mierzyć moc urządzeń pracujących w zakresie napięć LV (definicja napięcia Lov Voltage to mniej niż 1500VDC lub mniej niż 1000VAC RMS). Do realizacji takiego celu wymagane są: pomiar prądu, pomiar napięcia, akwizycja danych oraz ich transmisja do celu, celem może być urządzenie sterujące lub jak w naszym przypadku komputer osobisty.

1.1 Założenia projektowe i wysokopoziomowy opis projektu

Założenia projektowe odnośnie warstwy sprzętowej

- Pomiar prądu realizowany przy pomocy sensora Halla.
- Pomiar napięcia realizowany przy pomocy izolowanego wzmacniacza operacyjnego oraz wysokoimpedancyjnego dzielnika napięcia.
- Izolacja części wysokonapięciowej do sekcji niskonapięciowej.
- Dodatkowa warstwa izolacji pomiędzy sekcją niskonapięciową a sekcją komunikacyjną.
- Komunikacja z komputerem osobistym realizowana przy pomocy protokołu USB.

Założenia projektowe odnośnie warstwy oprogramowania

- Akwizycja danych za pomocą wbudowanego przetwornika ADC w mikrokontrolera.
- Transmisja danych przy pomocy protokołu UART.
- Odczytywanie danych diagnostycznych przy pomocy pinów GPIO.
- Uśrednianie pomiarów.
- Stworzenie "pakietu" danych, który będzie łatwy do interpretacji przez aplikację na komputery osobiste.
- Stworzenie aplikacji na komputery osobiste do odczytywania danych.

1.2 Podział odpowiedzialności

- Mikołaj - projekt wysokopoziomowy, projekt schematu oraz PCB, montaż, testowanie
- Arkadiusz - utworzenie aplikacji desktopowej odczytującej dane na porcie szeregowym
- Tomasz - programowanie mikrokontrolera, obliczanie wartości i testy oprogramowania

2 Sprzęt

2.1 Wybór komponentów

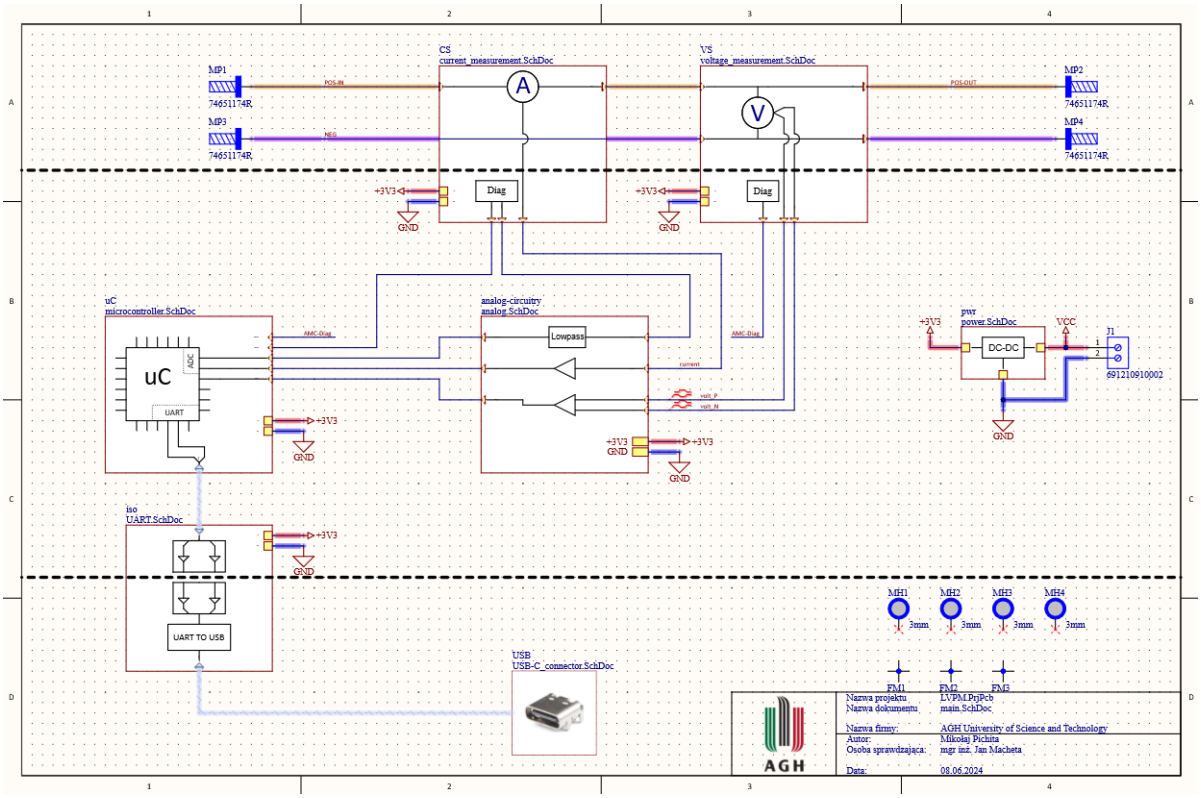
Opis wykorzystanych komponentów oraz uzasadnienie wyboru.

- Sensor Halla TMCS1123 - wzmocniona izolacja na poziomie 1.3kV, szeroki zakres pomiarowy, prostota w integracji
- Izolowany wzmacniacz operacyjny AMC3330 - wzmocniona izolacja na poziomie 1.7kV, wbudowana przetwornica DC-DC, wyjście różnicowe
- Mikrokontroler STM32F0 - prosta architektura, mały footprint, 12bitowe przetworniki ADC
- Liniowy regulator napięcia LP3997 - prąd wyjścia 250mA, niskie napięcie spadkowe na regulatorze, stałe wyjście 3.3V
- Izolator cyfrowy Si8621 - duża przepustowość, dwa kanały(jeden sygnału nadającego, drugi odbierającego), możliwość zasilenia zarówno 3.3V jak i 5V
- Konwerter UART→USB FT232 - standardowe rozwiązanie w przemyśle
- Wzmacniacz operacyjny TLV9004 - mały footprint, cztery kanały

Pozostałe komponenty wykorzystane w projekcie są elementami pasywnymi. W przypadku doboru rezystorów zwracano uwagę na dokładność rezystancji w kluczowych elementach, takich jak dzielniki napięcia, oraz na maksymalną moc jaką w stanie są one wytrzymać. Kondensatory dobierano tak aby klasyfikacja napięciowa danego kondensatora była co najmniej dwukrotnie większa od spodziewanego napięcia w danym węźle. Dodatkowo przy mikrokontrolerze zastosowano kondensatory w obudowie 0204 minimalizując impedancję pasożytniczą elementu.

2.2 Opis schematu

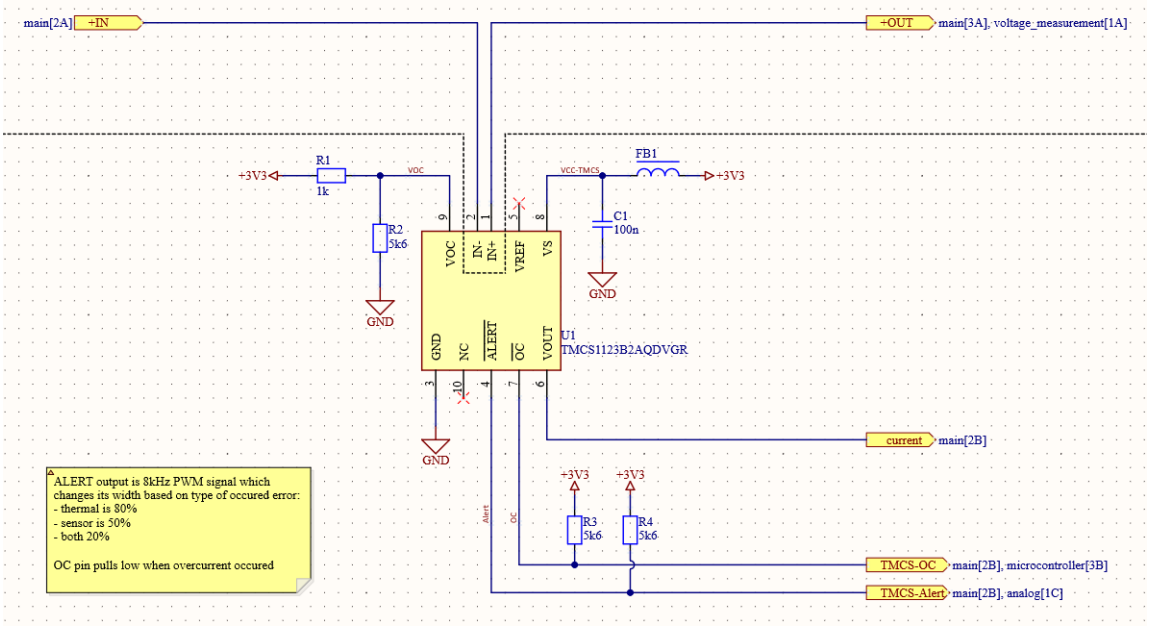
Projekt płytki PCB został wykonany w programie Altium Designer 24.



Rysunek 1: Schemat blokowy urządzenia po zaimplementowanych poprawkach

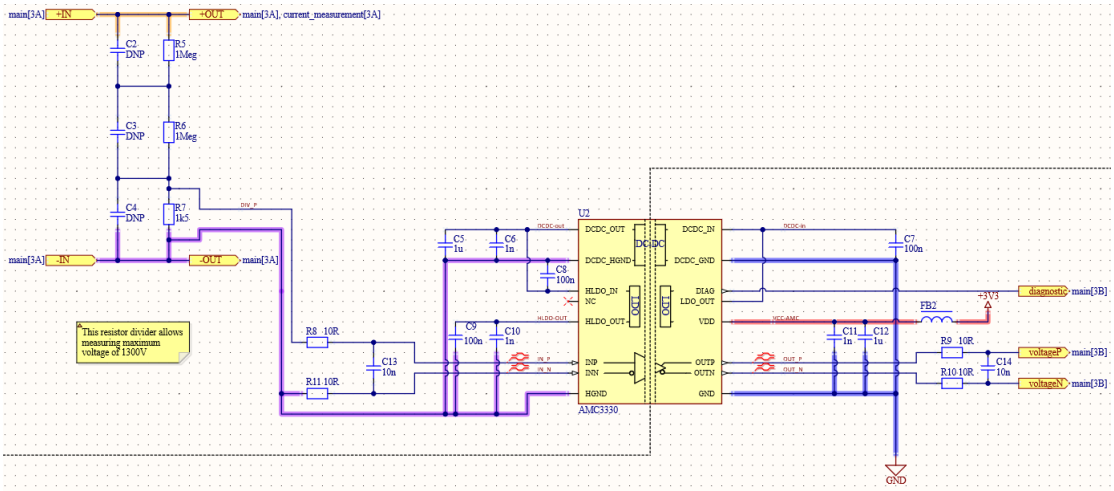
Schemat blokowy składa się łącznie z siedmiu bloków funkcyjnych:

- pomiar prądu
- pomiar napięcia
- front-end analogowy dla sygnałów pochodzących z sensorów
- mikrokontroler
- izolator cyfrowy i konwerter UART→USB
- złącze USB-C
- zasilanie



Rysunek 2: Fragment schematu zawierający układ odpowiedzialny za pomiar prądu

Wykorzystywany układ, TMCS1123, oprócz samego sensora Halla zawiera również kilka interesujących modułów. Sensor posiada układ diagnostyczny, kompensację temperatury, kompensację wpływu *zewnętrznych* zakłóceń w postaci pola magnetycznego oraz wyjście cyfrowe informujące o przekroczeniu zadanego limitu prądowego. Układ scalony jest produkowany w kilku wersjach, różnią się one możliwym zakresem mierzonego prądu. Wybrana wersja jest w stanie mierzyć prąd od -30A do +30A. Dodatkowo, przy pomocy dzielnika napięcia, została ustawiona wcześniej wspomniana wartość przekroczenia limitu prądowego. Sensor będzie zwracał dodatkowo informację o zbliżaniu się ustalonego progu zmieniając stan logiczny pinu OC. Wartość została ustawiona na 16.7A.

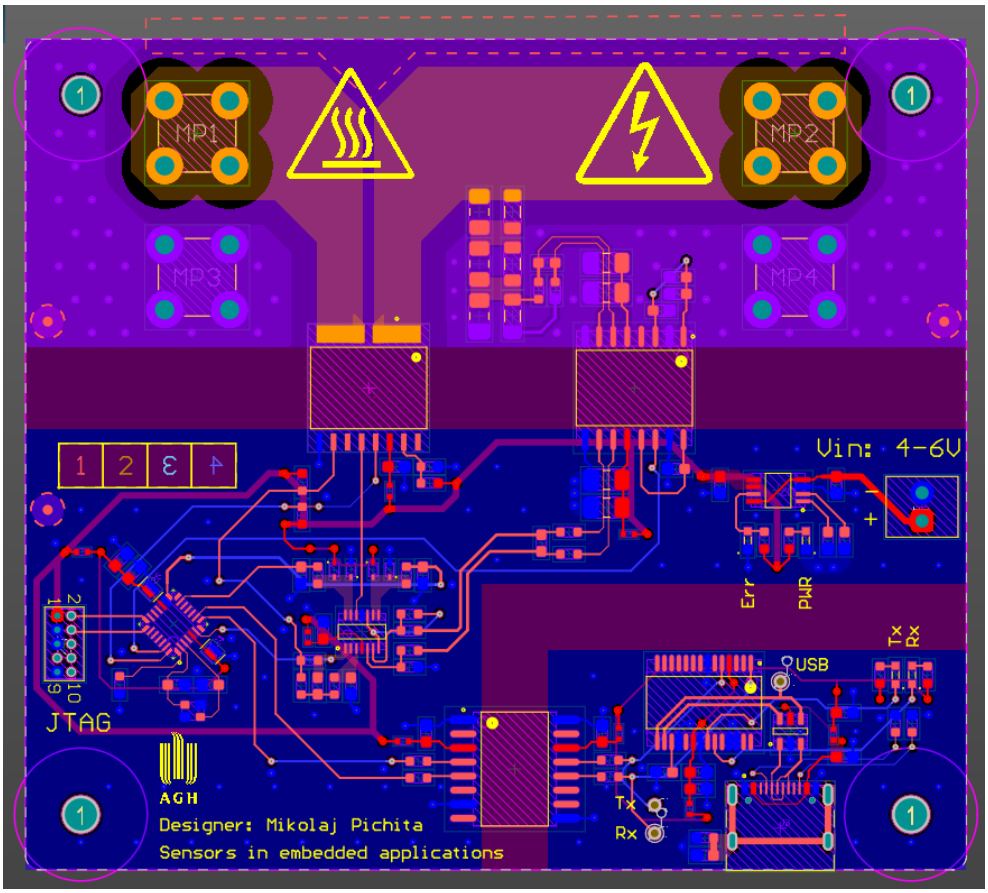


Rysunek 3: Fragment schematu zawierający układ odpowiedzialny za pomiar napięcia

Izolowany wzmacniacz operacyjny AMC3330 posiada wbudowaną przetwornicę DC-DC. Dodatkowo, układ posiada wbudowany moduł diagnostyczny moniturujący zachowanie wbudowanej przetwornicy. Maksymalne napięcie różnicowe jakie można podać między wejścia układu to +- 1V. Mając to na uwadze, dzielnik napięcia został dobrany w taki sposób aby zmaksymalizować zakres napięcia na rezystorze *pomiarowym*. Maksymalna wartość mierzonego napięcia wynosi 1300VDC.

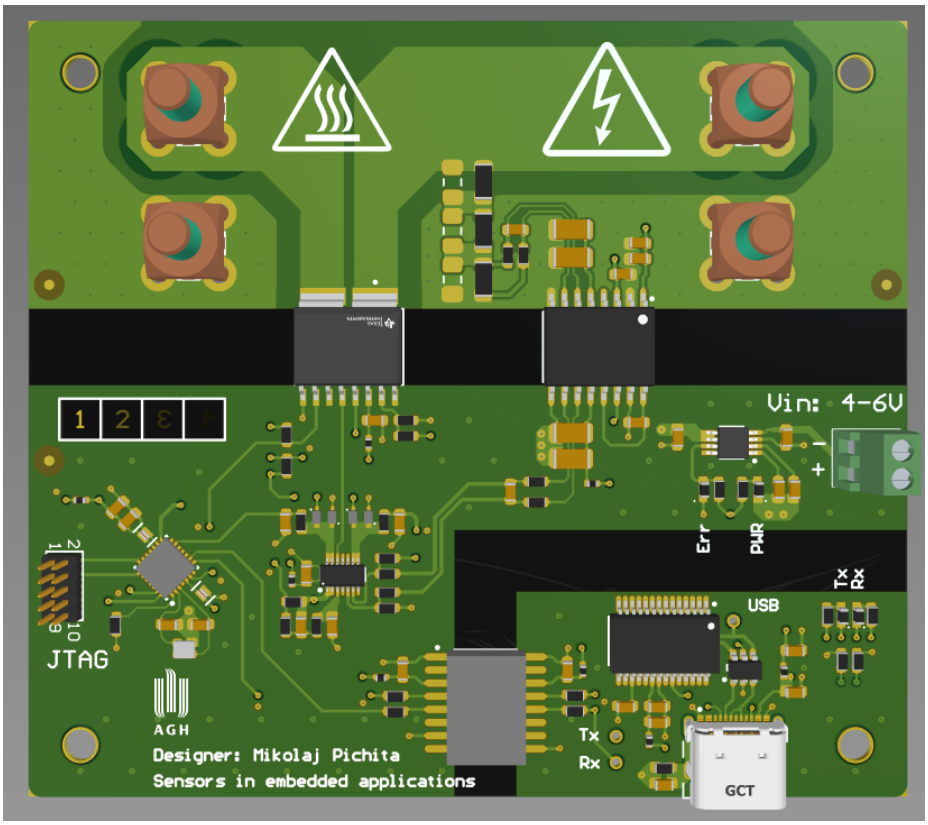
2.3 PCB

Najważniejszą rzeczą w trakcie projektowania było zachowanie odstępów zapewniających izolację, zarówno sensor TMCS jak i wzmacniacz AMC w pewien sposób zdefiniowały szerokość odstępu między częścią mocy a częścią niskonapięciową. W trakcie testów do urządzenia pomiarowego został podłączony komputer osobisty i aby uniknąć uszkodzenia sprzętu, wykorzystano dodatkowy izolator cyfrowy.



Rysunek 4: Widok 2D miernika mocy

Płytka posiada cztery warstwy gdzie dwie wewnętrzne służą jako poligony odniesienia dla sygnałów znajdujących się na warstwach Top i Bottom. Częstą praktyką jest umieszczanie na PCB *okienek* wskazujących daną warstwę płytki. Pozwala to na pewne ubezpieczenie, które zmniejsza szansę, że wykonawca płytki nie pomyli warstw miedzi. Na płycie zamieszczono 3 punkty *fiducial marker*. Wymiary płytki to 85x75mm.



Rysunek 5: Widok 2D miernika mocy

3 Oprogramowanie

3.1 Omówienie oprogramowania mikrokontrolera

Oprogramowanie na mikrokontroler powstawało w dwóch etapach. Pierwszy etap polegał na stworzeniu działającego programu na płytkę ewaluacyjną STM32F407, drugi zaś na przeniesieniu kodu na użyty w projekcie mikrokontroler STM32F0. Oba mikrokontrolery są z tej samej rodziny, więc w założeniu takie działanie miało na celu przyspieszenie stworzenia projektu (kod mógłby powstawać zanim płytka zostałaby złożona).

Pierwszy etap zawierał w sobie zebranie wymagań, jakie mikrokontroler miał spełniać i czynności, jakie miał wykonywać. Zawierały się w nich:

- pomiar prądu (wartość z ADC)
- pomiar napięcia (2 wartości z ADC)
- pomiar wypełnienia sygnału PWM (wykorzystywany timer)
- pomiar dwóch wartości diagnostycznych (wartości 1/0 z GPIO)
- wysyłanie danych za pomocą UART

Kolejnym krokiem było utworzenie projektu w programie STM32CubeIDE, wybraniu odpowiedniej płytki a następnie na zdefiniowaniu pinów, które miały zająć się określonymi zadaniami. Po wykonaniu tej czynności przystąpiono do pisania kodu, który miał się zająć kolejnymi problemami.

Pomiary, wymagające ADC, były wykonywane za pomocą pojedynczego ADC oraz trzech kanałów, jakie posiadał. Wykorzystany był tylko jeden ADC, ponieważ docelowy mikrokontroler posiadał tylko jeden taki moduł. Kanały zostały wykorzystane jako sposób szybkiego odczytywania informacji z kilku wejść. Wszystkie pomiary były wykonywane za pomocą DMA (Direct Memory Access), żeby uniknąć niepotrzebnego obciążania kontrolera. Po wykonaniu trzech odczytów z trzech kanałów wykonywane było przerwanie, za pomocą którego obliczano wartości prądu i napięcia, średnią prądu i napięcia z wielu pomiarów, a także wysyłano dane za pomocą UART.

Prąd był mierzony za pomocą pojedynczego kanału ADC, a jego odczytywana wartość była obliczana na wartość w mA×100, żeby uniknąć wykonywania obliczeń na liczbach zmiennoprzecinkowych.

$$I = (\frac{CurrentAVR \cdot 3300 \cdot 100}{4096} - 165000) \cdot 20 \tag{1}$$

Gdzie:

- $CurrentAVR$ - średnia wartość z ostatnich 10 pomiarów prądu

Reszta wartości stałych została dobrana tak, żeby z 12 bitowej wartości otrzymać prawdziwą wartość prądu. Uwzględniono offset na elemencie, który pozwala na pomiar prądu dodatniego jak i ujemnego.

Napięcie było mierzone różnicowo za pomocą dwóch kanałów ADC, odejmowane od siebie w celu usunięcia offsetu, a następnie jego wartość zostawała odczytana (w mV×10, żeby uniknąć wykonywania obliczeń na liczbach zmiennoprzecinkowych).

$$V = \frac{VoltageAVR \cdot 33 \cdot 3012}{4096} \tag{2}$$

Gdzie:

- $VoltageAVR$ - średnia wartość z ostatnich 10 pomiarów prądu

Reszta wartości stałych została dobrana tak, żeby z 12 bitowej wartości otrzymać prawdziwą wartość napięcia. Uwzględniono offset na elemencie poprzez odjęcie różnicowo dwóch wartości wejściowych. Wartości liczbowe przystosowane były do założenia, że rezystor pomiarowy na dzielniku napięcia będzie miał 332Ω. W finalnej wersji kodu wartości zostały odpowiednio zmienione, żeby obejmowały one nowy rezystor o wartości 1500Ω.

PWM był mierzony za pomocą opcji "PWM input" dostępnej dla połączonych kanałów dla timera. Pozwalało to na szybkie mierzenie sygnału za pomocą pojedynczego wejścia na mikrokontrolerze.

Dwie wartości diagnostyczne, które mogły osiągać jedynie stan wysoki lub niski były odczytywane za pomocą wejść GPIO. Ustawione był przerwania, które sprawiały, że w sytuacji kiedy wejście weszłoby w przeciwny stan byłoby to wpisywane do odpowiedniej zmiennej a następnie wysyłane przez UART.

Drugi etap programowania to było przeniesienie kodu na wykorzystany na płytce mikrokontroler STM32F0. Moduły dostępne na STM32F407 nie były w większości kompatybilne z STM32F0 i zostały zamienione. Lista wymagań i rozwiązań na tym mikrokontrolerze wygląda następująco:

- pomiar prądu (wartość z ADC)
- pomiar napięcia (wartość z ADC)

- pomiar wypełnienia sygnały PWM (wartość z ADC)
- pomiar dwóch wartości diagnostycznych (wartości 1/0 z GPIO)
- wysyłanie danych za pomocą UART

Prąd był mierzony za pomocą ADC, a jego odczytywana wartość była obliczana na wartość w mA×100, żeby uniknąć wykonywania obliczeń na liczbach zmiennoprzecinkowych.

$$avg\_c = (avg\_c \cdot 0.6) + (raw\_c \cdot 0.4) \tag{3}$$

$$I = (\frac{avg\_c \cdot 3300 \cdot 100}{4096} - 165000) \cdot 20 \tag{4}$$

Gdzie:

- avg_c - średnia wartość prądu, obliczana w sposób wykładniczy
- raw_c - dane pobierane z przetwornika

Reszta wartości stałych została dobrana tak, żeby z 12 bitowej wartości otrzymać prawdziwą wartość prądu. Uwzględniono offset na elemencie, który pozwala na pomiar prądu dodatniego jak i ujemnego.

Napięcie było mierzone za pomocą ADC. Różnicowe odejmowanie zostało już wykonane w układzie za pomocą wzmacniacza operacyjnego, więc jedyne co zostało to pomiar wartości. Odczytywana wartość była obliczana na wartość w mV×10, żeby uniknąć wykonywania obliczeń na liczbach zmiennoprzecinkowych.

$$avg\_v = (avg\_v \cdot 0.6) + (raw\_v \cdot 0.4) \tag{5}$$

$$V = \frac{avg\_v \cdot 33 \cdot 668 \cdot 2}{4096} \tag{6}$$

Gdzie:

- avg_v - średnia wartość napięcia, obliczana w sposób wykładniczy
- raw_v - dane pobierane z przetwornika

Reszta wartości stałych została dobrana tak, żeby z 12 bitowej wartości otrzymać prawdziwą wartość napięcia. Wartości liczbowe zostały zmienione, żeby uwzględnić zmianę rezystora w dzielniku napięcia na taki o wartości 1500Ω.

Wypełnienie sygnału PWM było mierzone za pomocą ADC. Na płytce został utworzony układ filtrujący. Na jego wyjściu można odczytać stałą wartość napięcia, która zmienia się wraz z wartością wypełnienia. 0% wypełnienia to 0V, 50% wypełnienia to 1.65V, a 100% wypełnienia to 3.3V czyli logiczne 1.

Dwie wartości diagnostyczne, które mogły osiągać jedynie stan wysoki lub niski były odczytywane za pomocą wejść GPIO. Wejścia te odczytywały wartości tych pinów i na bieżąco wysyłały za pomocą UART.

Przy przesyłaniu wiadomości protokołem UART uzgodniono pewien format, dzięki któremu aplikacja uruchomiona na komputerze osobistym będzie w stanie poprawnie zidentyfikować wysyłane do niej informacje. Wysyłanymi informacjami są: to dwa 32-bitowe *inty* przekazujące wartości napięcia oraz prądu wraz ze znakami a także 3 16-bitowe *inty*, za pomocą których przekazywane były sygnały diagnostyczne i współczynnik wypełnienia PWM przerobiony na wartość analogową. Te dane rozdzielono średnikami i wysyłano cały czas w tej samej kolejności.

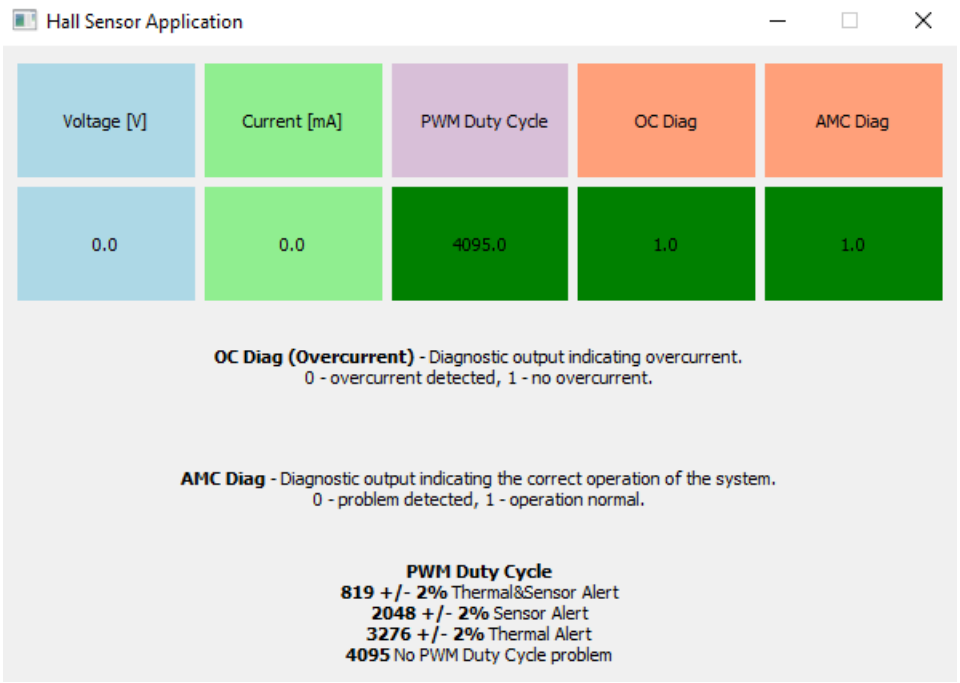
3.2 Aplikacja desktopowa do wyświetlania danych z sensora

Proces realizacji aplikacji desktopowej służącej do wyświetlania danych otrzymywanych na konkretnym porcie szeregowym rozpoczęty został od przygotowania środowiska programistycznego oraz zainstalowania odpowiednich bibliotek zewnętrznych.

Przygotowanie środowiska programistycznego

Aplikację desktopową służącą do wyświetlania danych otrzymanych z naszego skonstruowanego urządzenia zdecydowano się napisać korzystając z języka programistycznego 'Python', wykorzystując dodatkowo biblioteki takie jak: PyQt5 do tworzenia interfejsów graficznych oraz PySerial w celu zapewnienia dostępu do danych otrzymywanych na porcie szeregowym.

W celu umożliwienia uruchomienia aplikacji na jednostkach bez zainstalowanego Pythona oraz wykorzystanych bibliotek, zastosowano narzędzie 'PyInstaller' do stworzenia samodzielnego pliku wykonywalnego **AppSensor.exe**. Podczas procesu budowania, ustawiono flagi –**onefile** - wszystkie zależności oraz pliki pakowane są w jeden plik wykonywalny oraz –**windowed** - tworzenie aplikacji z interfejsem GUI bez konsoli.

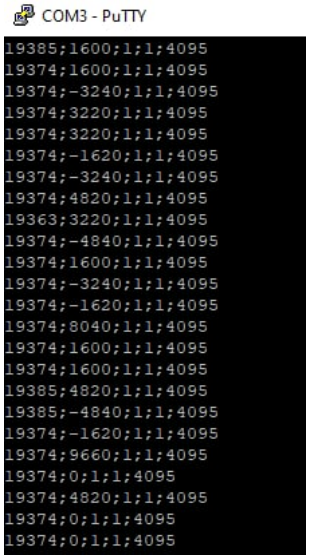


Rysunek 6: Okno aplikacji pomiarowej

Kod źródłowy aplikacji desktopowej

- **Wyszukiwanie sensora**
  Na potrzebny projektu napisano funkcję sprawdzającą listę dostępnych urządzeń podłączonych do jednostki przez port szeregowy USB. Funkcja ta wyszukuje urządzenie o nazwie 'USB Serial Port' (nazwa naszego urządzenia). W przypadku, gdy funkcja znajdzie urządzenie o takiej nazwie, zwraca numer portu szeregowego, do którego owe urządzenie jest podłączone, natomiast w przypadku, gdy na liście dostępnych urządzeń nie ma naszego urządzenia funkcja zwraca **None**.

- **Inicjalizacja interfejsu użytkownika**
  W przypadku, gdy zostanie znalezione podłączone urządzenie o odpowiedniej nazwie, wówczas zwracany jest port szeregowy, do którego urządzenie zostało podłączone. W następnym kroku inicjowany jest interfejs użytkownika dla odpowiedniej wartości portu ('COM' do którego podłączono sensor) oraz prędkości transmisji (baudrate).



Rysunek 7: Otrzymywane pakiety na porcie szeregowym

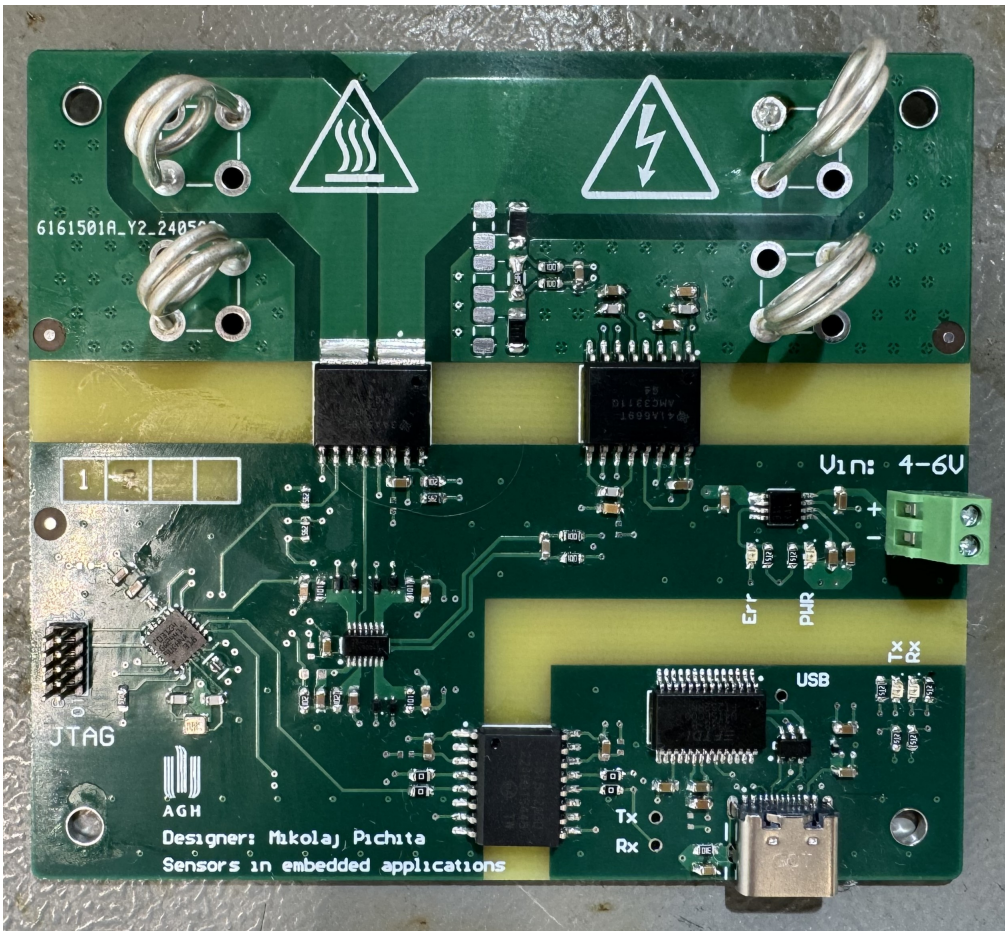
- **Pobieranie danych z portu szeregowego**
  Dane z sensora przesyłane są z wykorzystaniem protokołu komunikacyjnego UART na port szeregowy jednostki komputerowej. Przesyłane pakiety danych zostały zaprezentowane na rysunku 7. Jest to pięć wartości oddzielonych średnikiem, odpowiednio; napięcie, natężenie prądu, flaga diagnostyczna dotycząca przeciążenia prądowego, flaga diagnostyczna informująca o poprawnym działaniu urządzenia, wartość wypełnienia PWM. Prewencyjnie otrzymane dane są dekodowane dzięki wykorzystaniu 'UTF-8', usuwane są białe znaki, a następnie pakiet jest separowany na pięć wartości oddzielonych od siebie średnikiem. Wartości te wpisywane są do tablicy, z której następnie są pobierane w celu reprezentacji poszczególnych wartości.

- **Aktualizowanie wyświetlanych danych** W celu zapewnienia responsywności oraz stabilności aplikacji, zastosowano QTimer. Dzięki wykorzystanemu timerowi aplikacja może periodycznie wykonywać funkcje bez blokowania głównego wątku, co zapobiega możliwemu zawieszaniu się aplikacji przez wzgląd na wysokie zużycie procesora. Kolejnym plusem takiego rozwiązania jest przejrzystość interfejsu graficznego, prezentując użytkownikowi najnowsze dane, bez opóźnień.

  Interfejs graficzny składa się z pięciu pól prezentujących odpowiednio podpisane wartości. Wyjaśnienie wskazań indykatorów odpowiedzialnych za flagi diagnostyczne zostało umieszczone wraz z wyjaśnieniem możliwych błędów dotyczących wartości wypełnienia PWM na samym dole aplikacji okienkowej.

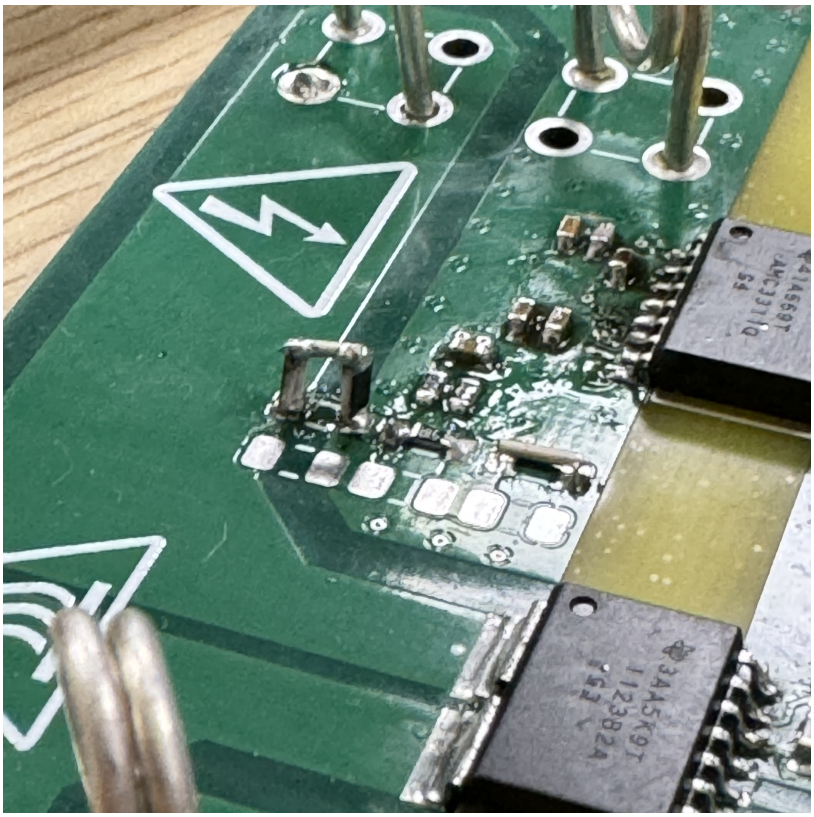
4 Uruchomienie

Po zebraniu wszystkich zamówionych części oraz otrzymaniu płytki PCB przystąpiono do procesu montażu. Dzięki umieszczeniu wszystkich komponentów na górnej warstwie odbyło się to bez większych problemów. Takie umieszczenie komponentów przyda się w późniejszym debugowaniu warstwy sprzętowej. W trakcie montażu zauważono pierwsze drobne błędy jakie popełniono w trakcie projektowania płytki: footprint udostępniony przez producenta diod schottkyego nie nadawał się do ręcznego montażu, przy złączu zasilającym płytkę umieszczono, na warstwie silkscreen, znaki pozwalające rozróżnić polaryzację zasilania - jak się okazało rozmieszczono je błędnie, polaryzacja diod LED została obrócona. Co prawda są to głównie błędy kosmetyczne, nie zakłócają one w żadnym stopniu pracy płytki ale należy je odnotować.



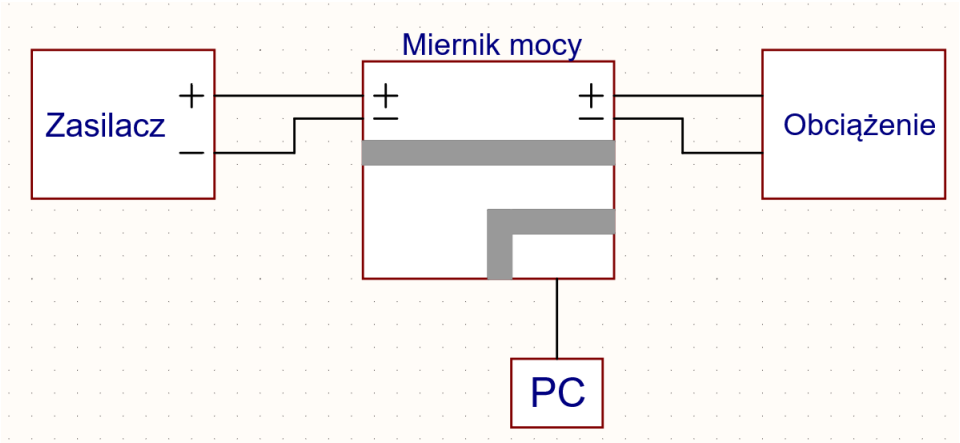
Rysunek 8: Zmontowana płytka miernika

Przed testami wysokonapięciowymi przeprowadzono testy funkcjonalne płytki. Kontroler bez problemu można było zaprogramować, transmisja po UART działa poprawnie - urządzenie po podpięciu do komputera jest wykrywane jako urządzenie szeregowe i da się zaobserwować przychodzące komunikaty, odczyt z kanałów ADC działa - wartości są wyświetlana na porcie szeregowym. Niepokojącą informacją był odczyt z kanałów dotyczących pomiaru napięcia. Na parze różnicowej odczytywano wartości, które nie były zgodne z wartościami otrzymanymi w symulacjach i opisanymi w datasheecie. Jak się okazało, został popełniony błąd przy projektowaniu bloku pomiaru napięcia. Fragment schematu widoczny na rysunku 3 jest poprawiony. Przed tym, rezystor pomiarowy był umieszczony między dwoma innymi rezystorami, a nie tak jak teraz - między rezystorem a potencjałem NEG. Po wprowadzeniu poprawki na PCB, odczytywane wartości zgadzały się z notą katalogową.



Rysunek 9: Zmodyfikowany dzielnik na płytce

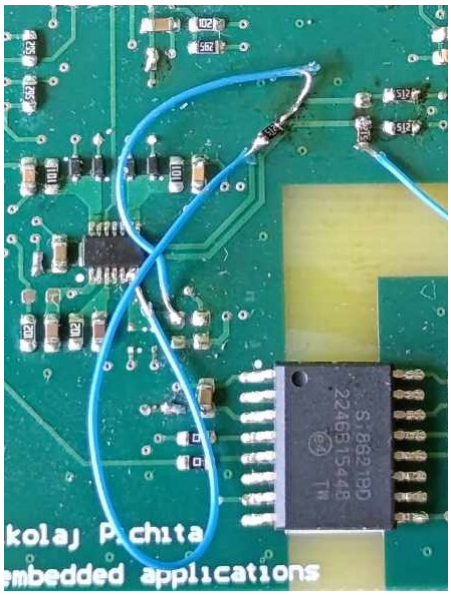
Po testach funkcjonalnych zostały przeprowadzone testy dokładności pomiarów.



Rysunek 10: Zmodyfikowany dzielnik na płytce

Podczas pierwszej sesji testowania napotkano ponownie problemy z układem mierzącym na-

pięcie. Zwracana wartość nie była w żadnym skorelowana z napięciem zasilacza. Płytka została zaprojektowana z myślą o wykorzystaniu różnicowych przetworników ADC, jak się okazało, zamówiony mikrokontroler STM32F0 nie posiada różnicowego wejścia ADC. Gdy połączy się to z faktem, że wbudowana przetwornica we wzmacniaczu AMC zakłócała linie zasilającą to można zauważyć dlaczego odczyty z pomiaru napięcia były takie niedokładne. Aby zaradzić temu problemowi delikatnie zmodyfikowano płytkę PCB. Przerobiono jeden kanał układu scalonego TLV9004 tak aby nie był on buforem lecz wzmacniaczem różnicowym. Część odpowiedzialna za pomiar prądu działała poprawnie.



Rysunek 11: Bufor przerobiony na wzmacniacz różnicowy

Po zaimplementowaniu tych zmian na płycie, już w trakcie drugiej sesji, przeprowadzono pomiary dokładności układów mierzących.

Napięcie zasilacza [V]	Zmierzone napięcie [V]	Błąd [%]
100	105.4	5.4
200	204.5	2.25
300	304.6	1.53
400	403.6	0.9
500	503.7	0.74
600	602.7	0.45
700	701.7	0.24
800	800.8	0.1
840	840.6	0.07

Tabela 1: Pomiary dokładności układu mierzącego napięcie

Prąd zasilacza [A]	Zmierzony prąd [A]	Błąd [%]
0.620	0.634	2.2
1	1.007	0.7
2	2.022	1.1
3	3.021	0.7
3.65	3.663	0.36
5	5.009	0.17
6	6.026	0.44
7	7.044	0.63
9	9.021	0.23

Tabela 2: Pomiary dokładności układu mierzącego prąd

Błąd przy *niskich* wartościach prądu i napięcia jest dosyć zauważalny. W przypadku pomiaru napięcia należałoby dodatkowo skompensować niedokładne wartości rezystorów w kodzie na kontroler. Gdy mierzone wartości rosną, naturalnie można zobaczyć mniejszy błąd, który wynosi w większości pomiarów mniej niż 1%. Na tej podstawie można stwierdzić, że zaprojektowane urządzenie działa zgodnie z założeniami przedstawionymi w rozdziale 1.1.

Bibliografia

- Nota katalogowa TMCS1123 `www.ti.com/product/TMCS1123`
- Nota katalogowa AMC3330 `www.ti.com/product/AMC3330`
- Nota katalogowa STM32F0 `www.st.com/en/microcontrollers-microprocessors/stm32f031g4.html`
- Nota katalogowa Si8620 `www.skyworksinc.com/search?q=si8621`
- Wpis na blogu `www.allaboutcircuits.com/technical-articles/jtag-connectors-and-interfaces/`
- Biblioteka PyQt5 `https://www.riverbankcomputing.com/software/pyqt/`
- Biblioteka PySerial `https://github.com/pyserial/pyserial`
- Narzędzie PyInstaller `https://pyinstaller.org/en/stable/`