

**Politechnika Warszawska**

W Y D Z I A Ł E L E K T R Y C Z N Y



# Praca dyplomowa magisterska

na kierunku Elektrotechnika

w specjalności Systemy Wbudowane

Synteza trajektorii fazowych oscylatora chaotycznego metodami generatywnymi

inż. Lidia Kocon  
numer albumu 318921

promotor  
dr. inż. Łukasz Makowski

WARSZAWA 2026



**Synteza trajektorii fazowych oscylatora chaotycznego metodami generatywnymi**  
**Streszczenie**

**Słowa kluczowe:** .....



**Synthesis of Phase-Space Trajectories of a Chaotic Oscillator Using Generative  
Methods**

**Abstract**

**Keywords:** .....



# Spis treści

<b>1 Wstęp</b>	<b>9</b>
<b>2 Metoda Theil-Sen</b>	<b>11</b>
2.1 Henri Theil . . . . .	11
2.2 Pranab Kumar Sen . . . . .	12
2.3 Metoda Theil-Sen . . . . .	13
2.4 Metoda najmniejszych kwadratów . . . . .	14
2.5 Porównanie metody . . . . .	15
<b>3 Porównanie metod przy użyciu środowiska Python</b>	<b>17</b>
3.1 Kod źródłowy Theil-Sen . . . . .	17
3.2 Kod źródłowy metody najmniejszych kwadratów . . . . .	20
3.3 Porównanie metod Theil-Sen i OLS . . . . .	21
3.3.1 Porównanie wyników . . . . .	22
<b>4 Implementacja estymatora Theil-Sen w środowisku STM32</b>	<b>25</b>
4.1 main.c . . . . .	25
4.2 thil_sen.c . . . . .	28
4.2.1 thil_sen.h . . . . .	30
4.3 Układ z rezystancyjnym dzielnicą napięcia . . . . .	31
4.3.1 Wykres i wynik . . . . .	31
4.3.2 Przeliczenie wartości . . . . .	32
4.4 Układ z diodą i rezystorem . . . . .	33
4.4.1 Wykres i wynik . . . . .	33
4.5 Napotkane problemy . . . . .	36
<b>5 Podsumowanie</b>	<b>39</b>
<b>Bibliografia</b>	<b>41</b>
<b>Spis rysunków</b>	<b>41</b>



# Rozdział 1

## Wstęp

W swojej książce David Blockley zauważał: „Niepodważalne jest, że inżynieria stanowi serce społeczeństwa” [david2012]. Inżynieria zajmuje się zarówno aspektami teoretycznymi, jak i praktycznymi, co umożliwiają rozwój innowacyjnych rozwiązań. Celem niniejszej pracy inżynierskiej jest połączenie tych dwóch elementów poprzez implementację estymatora Theil-Sen na mikrokontrolerze STM32.

Metoda Theil-Sen jest jedną z dokładniejszych technik stosowanych w analizie regresji, cieszącą się szczególnym uznaniem wśród metod odpornych na wartości odstające. Jedną z jej głównych zalet jest wysoka odporność na punkty odstające w zbiorach danych, co czyni ją niezawodnym narzędziem analitycznym nawet przy nieprecyzyjnych wynikach pomiarów. Umożliwia oszacowanie nachylenia prostej oraz punktu przecięcia z osią Y, dzięki czemu znajduje zastosowanie w szerokim zakresie dziedzin – od ekonomii, przez inżynierię, po metrologię.

Estymator został po raz pierwszy opisany przez Henriego Theila w artykule z 1950 roku. Theil zauważał, że dotychczasowe metody wyznaczania regionów ufności zakładały normalność rozkładu błędów. W swojej pracy pisał: „Regiony ufności dotychczas wyznaczano, zakładając normalność rozkładu. Głównym celem tego artykułu jest pokazanie, jak je wyznaczać bez takiego założenia” [theil1950]. Jego podejście pozwoliło na przeprowadzanie analizy bez konieczności przyjmowania, że błędy pomiarowe spełniają założenia o rozkładzie normalnym (rozkładzie Gaussa), co stanowiło przełom w ówczesnej statystyce.

W 1968 roku Pranab Kumar Sen, drugi z autorów tej metody, rozszerzył jej możliwości, uwzględniając przypadki, w których dwa punkty ze zbioru danych mają tę samą współrzędną x. W swojej pracy Sen pisał: „Niniejsze badanie skupia się na opracowaniu wielowymiarowego podejścia do tworzenia testów porządkowych opartych na takich samych rangach” [sen1968]. Dzięki temu metoda jest w stanie radzić sobie z bardziej złożonymi przypadkami, co znacznie zwiększyło jej uniwersalność i zastosowanie w analizie danych o powtarzających się wartościach.

Porównując metodę Theil-Sen z klasyczną metodą najmniejszych kwadratów, można zauważyc istotne różnice. Metoda najmniejszych kwadratów, mimo swojej popularności i szerokiego stosowania, nie radzi sobie najlepiej, gdy ma miejsce zniekształcenie wyników przez wartości odstające. Metoda Theil-Sen, dzięki swojej odporności na nieprawidłowości w danych, często okazuje się bardziej

## *Rozdział 1. Wstęp*

---

wiarygodnym wyborem w przypadku, gdy wyniki mimo anomalii w wartościach muszą być precyzyjne i niezawodne.

Ze względu na swoje zalety i uniwersalność, metoda Theil-Sen znajduje zastosowanie w wielu środowiskach programistycznych oraz na różnych platformach sprzętowych. Szczególnie ważna jest jej implementacja na mikrokontrolerach, które często służą do analizy danych w systemach czasu rzeczywistego. Celem niniejszej pracy dyplomowej jest implementacja metody Theil-Sen na mikrokontrolerze STM32, z wykorzystaniem języka C oraz środowiska programistycznego STM32CubeIDE. Proces implementacji będzie obejmował opracowanie algorytmu, wyznaczanie współczynnika nachylenia i punktu przecięcia z osią y oraz optymalizację obliczeń pod kątem ograniczonych zasobów obliczeniowych mikrokontrolera.

## Rozdział 2

# Metoda Theil-Sen

Formułę matematyczną estymatora Theil-Sen stworzyli holenderski profesor nauk ekonomicznych Henri Theil oraz indyjsko-amerykański profesor statystyki Pranab Kumar Sen. Pierwszy artykuł na jej temat ukazał się w 1950 roku, którego autorem był Henri Theil. Następna publikacja pojawiła się w 1968 roku autorstwa Pranab Kumar Sen. Sama metoda polega na dopasowywaniu linii trendu przez obliczanie mediany wszystkich nachyleń oraz wyrazów wolnych.

### 2.1 Henri Theil

Henri Theil urodził się w Holandii w 1925 roku. Swoją pracę naukową rozpoczynał jako student matematyki i fizyki na Uniwersytecie Utrecht w Amsterdamie. Theil jest najbardziej znany z ulepszenia metody najmniejszych kwadratów w dwóch etapach (2SLS) z 1953 roku.



Rysunek 1. Prof. dr. Henri Theil, autor Gemeente Rotterdam , r. 1966

Technika ta znacznie uprościła szacowanie modeli równań w ekonomii i stała się powszechnie stosowana. W tym samym roku został mianowany profesorem ekonometrii w Niderlandzkiej Szkole Ekonomii. W kolejnych latach założył tam między innymi Instytut Ekonometrii, który przężnie rozwijał się pod jego opieką. W 1966 roku przeniósł się do Stanów Zjednoczonych, na Uniwersytet w Chicago, gdzie kontynuował swoją pracę naukową. Zmarł 20 sierpnia 2000 roku w wieku 75 lat.

## 2.2 Pranab Kumar Sen

Pranab Kumar Sen urodził się w Kalkucie 7 listopada 1937 roku. Jest on autorem i współautorem wielu książek na temat statystyki. Znany jest głównie z opracowania estymatora Theil-Sen, był również członkiem Instytutu Statystyki Matematycznej oraz Amerykańskiego Towarzystwa Statystycznego (ASA). W 2010 roku stowarzyszenie przyznało mu nagrodę Wilks Memorial Award „za wybitne osiągnięcia w badaniach statystycznych, szczególnie w statystyce nieparametrycznej i biostatystyce oraz za wyjątkową służbę jako mentor dla doktorantów”. Sen zmarł w Chapel Hill, w Karolinie Północnej 31 grudnia 2023 roku w wieku 86 lat.



Rysunek 2. Prof. dr. Pranab Kumar Sen, autor Tom Fuldner, r. 2011

## 2.3 Metoda Theil-Sen

Metoda Theil-Sen, nazwana również estymatorem Theil-Sen, to technika dopasowywania prostej linii regresji na podstawie mediany nachyleń wszystkich linii przechodzących przez pary punktów istniejących w danym zbiorze.

Niech  $x_i, x_j, \dots, x_n$  oraz  $y_i, y_j, \dots, y_n$  będą niezależnymi punktami należącymi do zbioru liczb rzeczywistych.

Nachylenia wszystkich linii przechodzących przez pary punktów:

$$m_{ij} = \frac{y_j - y_i}{x_j - x_i}$$

Mediana nachyleń:

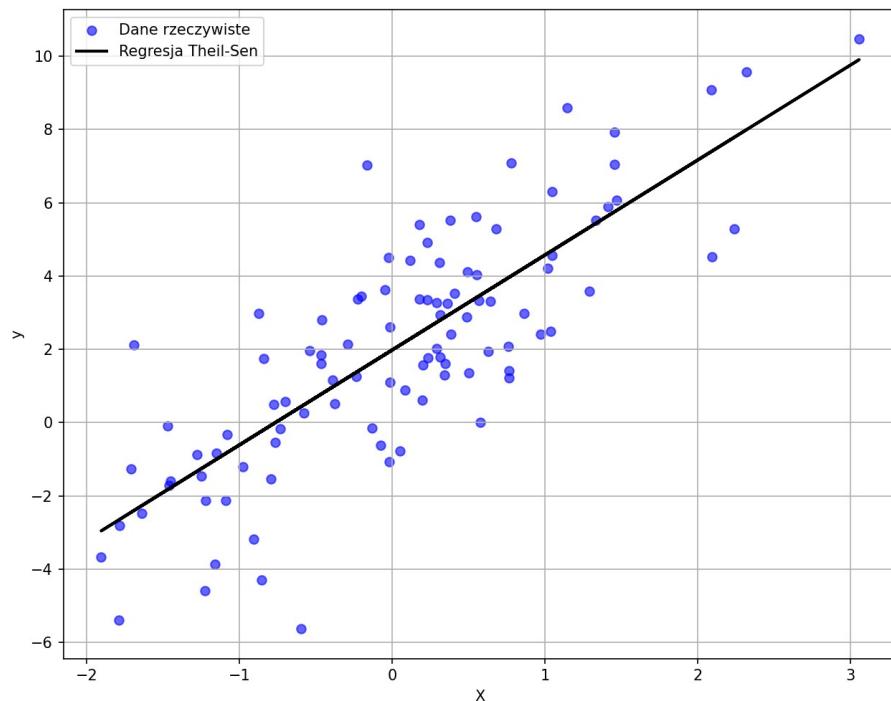
$$m = \text{median}a m_{ij}$$

Obliczenie wyrazu wolnego ( $b$ ) jako medianę wartości ( $y_i - mx_i$ ):

$$b = \text{median}a (y_i - mx_i)$$

Ostateczna linia regresji ma postać:

$$y = mx + b$$



Rysunek 3. Wykres metody Theil-Sen

Metoda ta jest szczególnie przydatna, gdy chcemy znaleźć linie tenu, ale występuje możliwość, że kilka nietypowych wartości zniekształci wynik.

Metoda Theil-Sen ma szerokie zastosowania w statystyce nieparametrycznej wymagającej założeń dotyczących rozkładu błędów, szerzej opisanych w artykule „Unbiasedness of the Theil-Sen estimator” z 2005 roku [wang2005].

## 2.4 Metoda najmniejszych kwadratów

Metoda najmniejszych kwadratów [linnik1962] [rand2010](z angielskiego Method of Least Squares OLS) jest techniką stosowaną w statystyce, polegającą na dopasowaniu linii prostych (lub krzywych) do zbioru danych [bartoszewicz1989] [zieliński1990]. Jej celem jest wyznaczenie linii regresji, która najlepiej dopasuje się do danych punktów. Przedstawia się ją jako prostą  $y = ax + b$ , gdzie "a" to nachylenie, "b" to punkt przecięcia z osią y.

Dla zbioru danych  $(x_1, y_1), (x_2, y_2) \dots (x_n, y_n)$ , szukana jest prosta  $y = ax + b$ , dla której suma kwadratów odchyleń między rzeczywistymi wartościami a wartościami przewidywanymi przez linię  $y_i = ax_i + b$  była minimalna.

Suma z reszty kwadratów

$$S(a, b) = \sum (y_i - (ax_i + b))^2$$

Aby znaleźć wartości dla a i b, należy zminimalizować funkcję S(a,b). By to osiągnąć, należy zróżniczkować funkcję S względem, a i b, a następnie przyrównać te pochodne do zera.

Dla a:

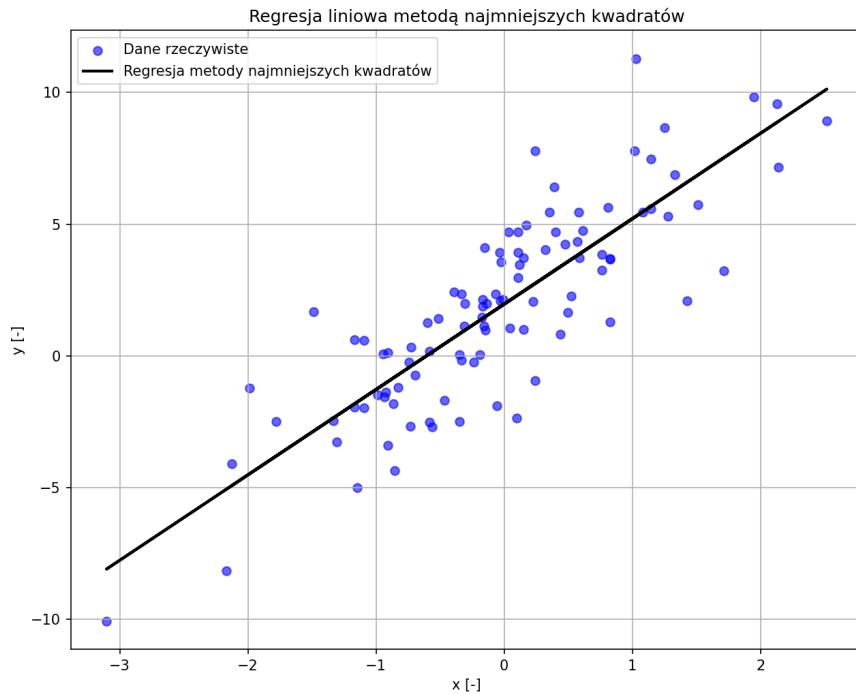
$$\frac{\partial S}{\partial a} = -2 \sum_{i=1}^n x_i(y_i - (ax_i + b)) = 0$$

Dla b:

$$\frac{\partial S}{\partial b} = -2 \sum_{i=1}^n (y_i - (ax_i + b)) = 0$$

Metoda najmniejszych kwadratów jest kluczowym narzędziem w analizie regresji, stosowanym w przeróżnych przypadkach. Dzięki efektywności obliczeniowej znajduje zastosowanie w takich dziedzinach jak analiza sygnałów, optymalizacja procesów technologicznych czy prognozowanie ekonomiczne.

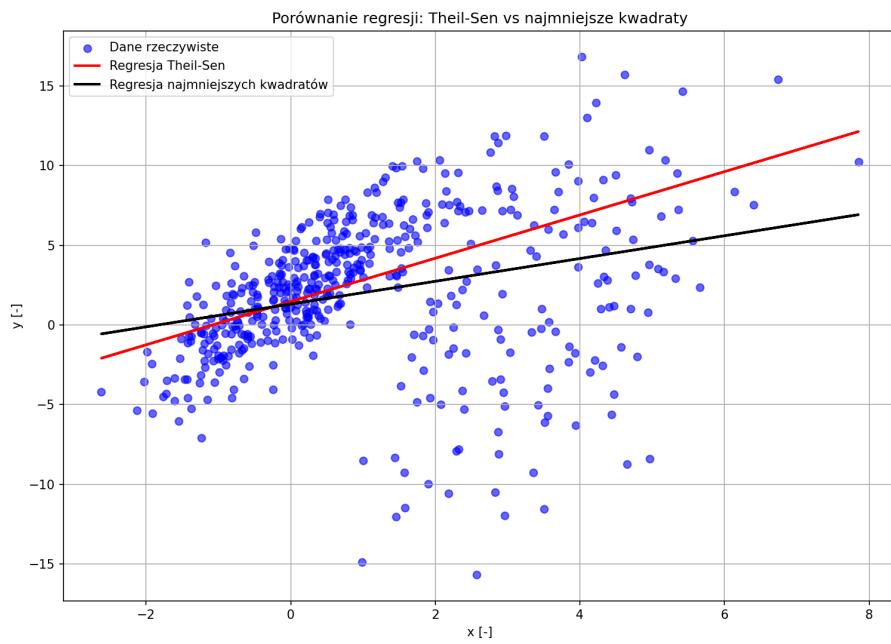
Jednym z przykładów zastosowania tej metody jest artykuł z 2006 roku, w którym wykorzystano ją do identyfikacji oraz separacji składowych harmonicznych prądu obciążenia w aktywnych filtrach mocy, co umożliwiło skuteczną kompensację zakłóceń i poprawę jakości energii elektrycznej [chudamani2006].



Rysunek 4. Wykres metody najmniejszych kwadratów

## 2.5 Porównanie metody

Metoda Theil-Sen i metoda najmniejszych kwadratów to dwa podejścia do wyznaczania linii regresji.



Rysunek 5. Wykres obu metod

Metoda najmniejszych kwadratów minimalizuje sumę kwadratów odchyleń między przewidywanymi a rzeczywistymi wartościami. Co sprawia, że jest szybsza, bardziej wydajna obliczeniowo oraz lepiej radzi sobie przy dużych zbiorach danych. Z kolei estymator Theil-Sen wyznacza nachylenie jako medianę nachyleń wyliczoną ze wszystkich możliwych par punktów, co czyni go dużo bardziej odpornym na wartości odstające. Fakt ten oznacza, że jest lepszą metodą w sytuacjach, gdzie istnieją dane o odstających wartościach i czas obliczeń nie jest istotny. Obie metody znajdują zastosowanie w analizie finansowej. W artykule z 2015 roku omówiono wykorzystanie metody najmniejszych kwadratów do prognozowania przyszłych wyników finansowych oraz przeprowadzono porównanie z metodą Theil-Sen. Analiza wykazała, że metoda Theil-Sen lepiej radzi sobie z wartościami odstającymi, zapewniając bardziej wiarygodne wyniki w porównaniu do metody najmniejszych kwadratów [james2015].

## Rozdział 3

# Porównanie metod przy użyciu środowiska Python

W celu obrazowego porównania obu metod należy przygotować program w języku Python, który zmierzy czas wykonywania obliczeń, wyznaczy współczynniki nachylenia i wyraz wolny dla każdej z metod oraz przedstawi wyniki na wykresie [amit2021] [mark2022] [luciano2015].

### 3.1 Kod źródłowy Theil-Sen

Program realizuje obliczenia za pomocą algorytmu Theil-Sen, wyznaczając współczynnik nachylenia i wyraz wolny. Następnie generuje wykres, na którym przedstawione są dane wejściowe oraz dopasowana do nich linia regresji, umożliwiając wizualną ocenę działania estymatora.

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 from sklearn.metrics import mean_squared_error
```

**Listing 1.** Program Theil-Sen

Na listingu 1 importowane są potrzebne biblioteki: NumPy do operacji numerycznych, Matplotlib do tworzenia wykresów oraz mean\_squared\_error do obliczania błędu średniokwadratowego (MSE), który jest miarą stosowaną do oceny jakości dopasowania modelu do danych.

```
1 def mediana(tablica):
2     tablica_posortowana = np.sort(tablica)
3     dlugosc = len(tablica_posortowana)
4     if dlugosc % 2 == 0:
5         return (tablica_posortowana[dlugosc // 2 - 1] + tablica_posortowana[
6             dlugosc // 2]) / 2.0
7     else:
8         return tablica_posortowana[dlugosc // 2]
```

**Listing 2.** Program Theil-Sen

Funkcja służy do obliczania mediany z tablicy. Na samym początku sortuje dane od najmniejszych do największych, wykorzystując funkcję `np.sort()`. Następnie zwraca wartości mediany. Jeśli zbiór elementów jest parzysty, funkcja zwraca średnią dwóch środkowych liczb, a jeśli jest nieparzysty, to środkową wartość.

```
1
2 def regresja_theil_sen(x, y):
3     n = len(x)
4     nachylenia = []
5     wyrazy_wolne = []
```

**Listing 3.** Program Theil-Sen

Na Listing 3 zdefiniowana jest funkcja `regresja_theil_sen`, która implementuje estymator Theil-Sen. Funkcja rozpoczyna od obliczenia długości wektora `x` oraz inicjalizacji dwóch pustych list: `nachylenia` i `wyrazy_wolne`. Listy te będą przechowywać odpowiednio wszystkie obliczone nachylenia i wyrazy wolne.

```
1 for i in range(n):
2     for j in range(i + 1, n):
3         if x[i] != x[j]:
4             nachylenie = (y[j] - y[i]) / (x[j] - x[i])
5             nachylenia.append(nachylenie)
6 mediana_nachylenia = mediana(nachylenia)
```

**Listing 4.** Program Theil-Sen

Zostały zdefiniowane pętle `for`, w których obliczane są nachylenia każdej pary punktów ( $x_i, x_j$ ) zgodnie ze wzorem regresji metodą Theil-Sena. Następnie, przy użyciu metody `append()`, obliczone wartości nachyleń są dodawane do listy. Po zakończeniu pętli, na podstawie zgromadzonych wartości, obliczana jest mediana nachyleń.

```
1 for i in range(n):
2     wyrazy_wolne.append(y[i] - mediana_nachylenia * x[i])
3 mediana_wyrazu = mediana(wyrazy_wolne)
4 wynik = { 'nachylenie': mediana_nachylenia, 'wyraz_wolny': mediana_wyrazu
5 }
6 return wynik
```

**Listing 5.** Program Theil-Sen

Na tej samej zasadzie obliczany jest wyraz wolny. Zgromadzone wartości wyrazów wolnych są dodawane do listy, a następnie obliczana jest ich mediana. Na koniec zwracany jest wynik zawierający obliczone wartości nachylenia oraz wyrazu wolnego.

```
1 x = np.random.randn(100, 1)
2 y = 3 * x.flatten() + 2 + np.random.randn(100) * 2
```

**Listing 6.** Program Theil-Sen

Ustawienie ziarna losowości zapewnia otrzymanie zawsze takiego samego punktu wyjścia, więc przy każdym uruchomieniu kodu generowane liczby będą identyczne. Co zapewni te same wyniki. Następnie stworzona zostaje tablica  $x$  zawierająca 100 losowych liczb w przedziale od 0 do 10 oraz tablica  $y$ , której każda z wartości jest wynikiem równania  $y = 3x + 2$  z dodatkiem szumu (losowych odchyłek).

```
1 wynik = regresja_theil_sen(x, y)
2 y_przewidziane_theil_sen = przewidz(x, wynik)
3 mse = mean_squared_error(y, y_przewidziane_theil_sen)
```

**Listing 7.** Program Theil-Sen

Zmienna  $wynik$  zawiera dopasowany model regresji Theil-Sen do danych wejściowych  $x$  i  $y$ . Na podstawie tego modelu przewidywane są wartości  $y$  dla każdego punktu  $x$ . Na koniec obliczany jest błąd średniokwadratowy (MSE), który ocenia jakość dopasowania modelu do rzeczywistych danych. Jest on wyrażony jako średnia kwadratów różnic między wartościami rzeczywistymi a wartościami przewidywanymi przez model.

```
1 print("Metoda Theil-Sena:")
2 print(f"Nachylenie (współczynnik kierunkowy): {model_theil_sen.coef_[0]:.2f}")
3 print(f"Przesunięcie (punkt przeciecia z osią Y): {model_theil_sen.intercept_:.2f}")
4 print(f"Blad średniokwadratowy (MSE): {mse:.2f}")
```

**Listing 8.** Program Theil-Sen

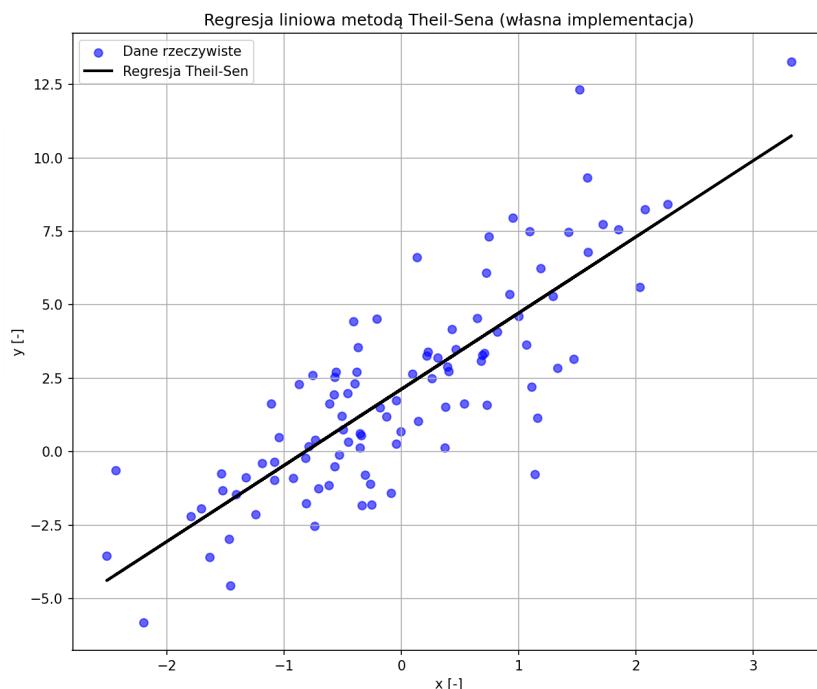
Kod wyświetla za pomocą funkcji `print` wartości współczynników regresji Theil-Sena, w tym nachylenie, przesunięcie oraz obliczony błąd średniokwadratowy (MSE). Wyniki są zaokrąglone do dwóch miejsc po przecinku.

```
1 plt.figure(figsize=(10, 6))
2 plt.scatter(x, y, color='blue', label='Dane rzeczywiste', alpha=0.6)
3 plt.plot(x, y_pred, color='black', label='Regresja Theil-Sen', linewidth=2)
4 plt.xlabel('x')
5 plt.ylabel('y')
6 plt.title('Regresja metoda Theil-Sen')
7 plt.legend()
8 plt.grid(True)
9 plt.show()
```

**Listing 9.** Program Theil-Sen

W fragmencie kodu tworzony jest wykres, na którym przedstawiane są rzeczywiste dane, oraz linia regresji uzyskana za pomocą metody Theil-Sena. Rzeczywiste dane są reprezentowane przez

niebieskie punkty, natomiast przewidywana linia regresji Theil-Sena jest zaznaczona czarną linią. Dodatkowo wykres zawiera opisy osi, tytuł oraz podziałkę, co ułatwia interpretację wyników.



Rysunek 6. Wizualizacja metody Theil-Sen

Po przeprowadzonych obliczeniach metodą Theil-Sen wyświetlany jest wynik nachylenia, wyrazu wolnego oraz błędu średniokwadratowego.

Nachylenie (współczynnik kierunkowy): 2.59

Przesunięcie (punkt przecięcia z osią Y): 2.13

Błąd średniokwadratowy (MSE): 3.81

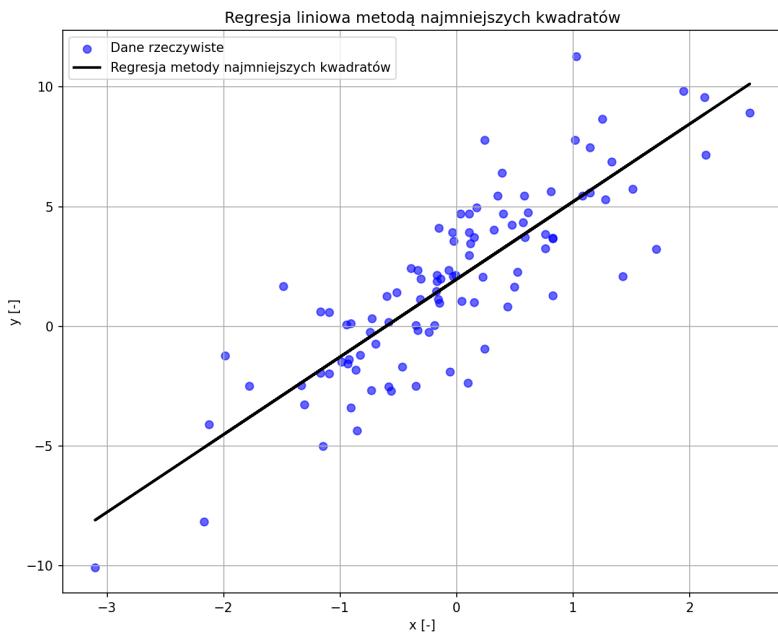
### 3.2 Kod źródłowy metody najmniejszych kwadratów

Program realizuje obliczenia linii regresji metodą najmniejszych kwadratów (OLS), dopasowując ją tak, aby minimalizować różnice między danymi a modelem. Następnie generuje wykres, na którym przedstawione są punkty danych i dopasowana linia, co pozwala ocenić działanie metody.

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 from sklearn.linear_model import LinearRegression
4 from sklearn.metrics import mean_squared_error
```

Listing 10. Program metody najmniejszych kwadratów

Na Listing 10 przedstawiono wykorzystanie modułu `sklearn.linear_model` oraz klasy `LinearRegression`, która służy do przeprowadzenia regresji liniowej metodą najmniejszych kwadratów. Dodatkowo, do oceny jakości dopasowania modelu zastosowano błąd średniokwadratowy. Wszystkie inne kroki w kodzie, takie jak dopasowanie modelu do danych, przewidywanie wartości czy wizualizacja wyników, pozostają bez zmian.



Rysunek 7. Wizualizacja metody najmniejszych kwadratów

Po wykonaniu obliczeń metodą najmniejszych kwadratów wyznaczana jest wartość współczynnika nachylenia, wyrazu wolnego oraz błędu średniokwadratowego.

Nachylenie (współczynnik kierunkowy): 2.78

Przesunięcie (punkt przecięcia z osią Y): 2.08

Błąd średniokwadratowy (MSE): 4.89

### 3.3 Porównanie metod Thei-Sen i OLS

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3 from sklearn.linear_model import TheilSenRegressor
4 from sklearn.metrics import mean_squared_error
5 import time

```

Listing 11. Program dla obu metod

W tym przypadku zostanie wykorzystana już gotowa biblioteka Theil-Sen z dodaną biblioteką 'time', która będzie odpowiedzialna za mierzenie czasu, co pozwoli na sprawdzenie, która z metod jest szybsza.

```
1 num_outliers = 100
2 indices = np.random.choice(range(500), num_outliers, replace=False)
3 x[indices] = x[indices] + np.random.uniform(2, 5, (num_outliers, 1))
4 y[indices] = y[indices] + np.random.uniform(-10, 10, num_outliers)
```

---

**Listing 12.** Program dla obu metod

Użyta tu komenda służy do wyboru losowych punktów odstających oraz zwiększenia wartości x dla punktów.

```
1 start_time_theil_sen = time.perf_counter()
2 model_theil_sen = TheilSenRegressor()
3 model_theil_sen.fit(x, y)
4 y_przewidziane_theil_sen = model_theil_sen.predict(X)
5 end_time_theil_sen = time.perf_counter()
6 time_theil_sen = end_time_theil_sen - start_time_theil_sen
```

---

**Listing 13.** Program dla obu metod

Na listingu 13 dokonywany jest pomiar czasu trwania obliczeń dla metody Theil-Sen. Proces ten polega na zainicjowaniu modelu regresji oraz dopasowaniu go do danych. Zapisujemy czas rozpoczęcia pomiaru, a następnie wykonujemy wszystkie operacje związane z dopasowaniem. Na końcu mierzymy czas zakończenia obliczeń, a różnica między tymi dwoma momentami daje czas trwania obliczeń dla metody.

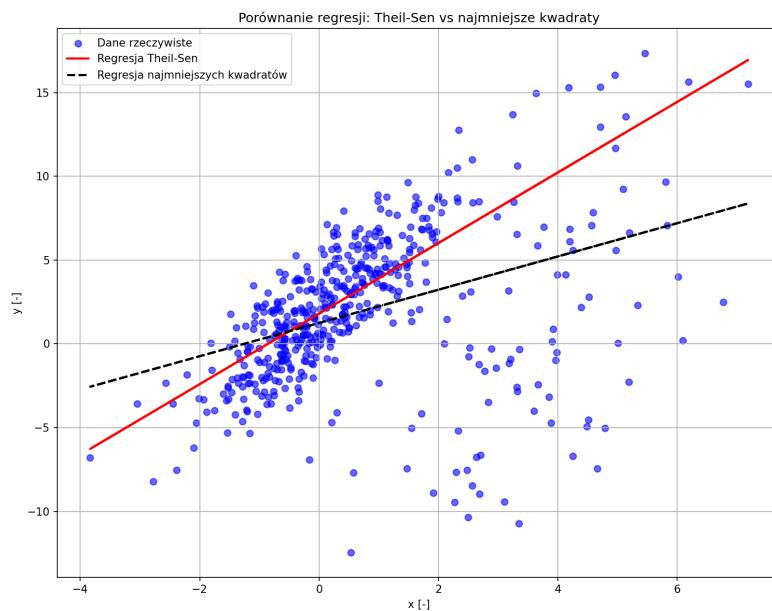
### 3.3.1 Porównanie wyników

	Theil-Sen	OLS
Nachylenie (współczynnik kierunkowy)	2.81	1.87
Przesunięcie (współczynnik przesunięcia z osią Y)	1.87	1.51
Czas obliczeń [s]	0.2806	0.0003
Błąd średniokwadratowy (MSE)	4.68	4.89

**Tabela 1.** Porównanie wyników

W tym przypadku celowo jest dodane dużo punktów oraz wartości odstającej, by lepiej zaobserwować różnicę w obu metodach. Z tabeli i wykresu można wysnuć wnioski, iż metoda Theil-Sen (linia czerwona) poprawniej dopasowała linię regresji do podanych punktów w porównaniu do metody najmniejszych kwadratów (czarna prosta). Za to w tej drugiej wynik udało się uzyskać w krótszym czasie. Co zgadza się z początkowymi założeniami odnośnie do obu metod [hristo2019] [james2015].

Metoda Theil-Sena posiada szereg zalet, które czynią ją atrakcyjną alternatywą w stosunku do innych technik regresji, szczególnie w kontekście danych zawierających wartości odstające. Dzięki



Rysunek 8. Wizualizacja obu metod

swojej odporności na wpływ punktów odstających metoda ta może zapewniać dokładniejsze wyniki w przypadkach, gdy tradycyjne metody, takie jak regresja najmniejszych kwadratów, mogą być podatne na zwiększenia wynikające z takich anomalií. Z tego powodu implementacja estymatora Theil-Sena na mikrokontrolerze STM32 może okazać się korzystnym rozwiązaniem, szczególnie w aplikacjach, w których dane wejściowe charakteryzują się dużą zmiennością lub obecnością wartości odstających.



## Rozdział 4

# Implementacja estymatora Theil-Sen w środowisku STM32

Kody w STM32 stanowią główną część pracy. Pierwszy z nich, czyli plik main.c, odpowiada za konfigurację mikrokontrolera. Kolejne pliki, theil\_sen.c oraz theil\_sen.h, zawierają implementację samego algorytmu Theil-Sen i odpowiadają za jego działanie. Taki podział kodu pozwala na wyraźne rozgraniczenie funkcji związanych z obsługą mikrokontrolera od właściwej logiki obliczeniowej [UM1727] [ionescu2020] [jacko2019].

### 4.1 main.c

Jest to główny plik odpowiadający za konfigurację mikrokontrolera oraz dostarczanie wyników [francuz2015] [prata2016].

```
30 /* USER CODE BEGIN Includes */  
31 #include "string.h"  
32 #include "stdio.h"  
33 #include "theil_sen.h"  
34 /* USER CODE END Includes */
```

**Listing 14.** src/main.c

Na listingu 14 "string.h" dodaje funkcje do pracy z ciągami znaków. "stdio.h" umożliwia korzystanie z funkcji wejścia/wyjścia, takich jak printf do wyświetlania informacji. "theil\_sen.h" wprowadza deklaracje funkcji związanych z algorytmem Theil-Sen.

```
42 /* USER CODE BEGIN PD */  
43 #define MAX_VALUES 4095  
44 /* USER CODE END PD */
```

**Listing 15.** src/main.c

Ustawienie tablicy na maksymalną wartość równą 4095, czyli maksymalną wartość bitową mikrokontrolera.

```
53 /* USER CODE BEGIN PV */
54 uint32_t value_adc;
55 uint32_t i;
56 uint32_t dane_wypelnione = 0;
57 char msg[100];
58 float x[MAX_VALUES];
59 float y[MAX_VALUES];
60 r = 330;
61 /* USER CODE END PV */
```

**Listing 16.** src/main.c

```
107 /* USER CODE BEGIN 2 */
108 HAL_DAC_Start(&hdac1, DAC_CHANNEL_1);
109 HAL_ADCEx_Calibration_Start(&hadc1, ADC_SINGLE_ENDED);
110 /* USER CODE END 2 */
```

**Listing 17.** src/main.c

Pierwsza funkcja ma za zadanie uruchomić przetwornik cyfrowo-analogowy (DAC) na kanale 1. Druga kalibruje przetwornik analogowo-cyfrowy (ADC) w trybie pojedynczego wejścia, co poprawia dokładność pomiarów.

```
113 /* USER CODE BEGIN WHILE */
114 while (1)
115 {
116     if (!dane_wypelnione) {
117
118         for(i = 0; i < MAX_VALUES; i++)
119         {
120             HAL_ADC_Start(&hadc1);
121             HAL_ADC_PollForConversion(&hadc1, HAL_MAX_DELAY);
122             HAL_DAC_SetValue(&hdac1, DAC_CHANNEL_1, DAC_ALIGN_12B_R, i);
123             value_adc = HAL_ADC_GetValue(&hadc1);
124
125             x[i] = value_adc/r;
126             y[i] = value_adc
127         }
128         dane_wypelnione = 1;
129     }
130 HAL_Delay(100);
131 /* USER CODE END WHILE */
```

**Listing 18.** src/main.c

W pętli while znajduje się pętla for, która wykonuje się, dopóki zmienna "i" nie osiągnie wartości MAX\_VALUES, zdefiniowana wcześniej w kodzie jako 4095. Przy każdym wykonaniu pętli zmienna "i" jest inkrementowana, co powoduje zwiększenie wartości DAC o 1.

Pętla rozpoczyna się od komendy uruchamiającej ADC. Następnie program oczekuje na zakończenie przetwarzania sygnału analogowego przez ADC i ustawia odpowiednią wartość na DAC. Wynik z ADC jest odczytywany za pomocą funkcji HAL\_ADC\_GetValue i zapisywany do zmiennej value\_adc. Wartości y są równe wartościom value\_adc i zapisywane do tablicy.

Zmienne x są otrzymywane w analogiczny sposób, jednak dodatkowo dzielone są przez rezystancję "r" równą  $330 \Omega$ , zgodnie ze wzorem  $I = U/R$ . Takie podejście pozwala uzyskać wyniki z precyzją do kilku miejsc po przecinku. Aktualnie wartość value\_adc mieści się w zakresie od 0 do 4095. Dopiero po przeliczeniu na odpowiednią skalę (od 0 do 3,3 V) dane będą gotowe do prezentacji. Proces ten zostanie szczegółowo opisany w dalszej części pracy.

Aby lepiej zrozumieć cel i sposób działania, należy wyjaśnić, dlaczego używane są przetworniki DAC i ADC w tej implementacji. DAC pełni rolę wymuszenia, czyli generuje sygnał napięciowy sterujący, który jest podawany na wejście układu. ADC z kolei mierzy napięcie wynikowe, odpowiadające temu wymuszeniu.

Warto również wspomnieć o roli rezystora  $330 \Omega$ . Jego obecność ma na celu ograniczenie prądu w obwodzie, co zapobiega uszkodzeniu elementów, takich jak na przykład diody. W zależności od układu można tę wartość dowolnie zmieniać.

```

132
133     /* USER CODE BEGIN 3 */
134
135     size_t dlugosc = sizeof(x) / sizeof(x[0]);
136
137     WynikTheilSena wynik = TheilSen_Estymator(x, y, dlugosc);
138
139     sprintf(msg, "Wyraz wolny: %.2f\r\nNachylenie: %.2f\r\n", wynik.
140             wyraz, wynik.nachylenie); // @suppress("Float formatting support")
141     HAL_UART_Transmit(&huart2, (uint8_t*)msg, strlen(msg), HAL_MAX_DELAY
142 );  

143     HAL_Delay(200);
144 }
145 /* USER CODE END 3 */

```

**Listing 19.** src/main.c

Fragment kodu rozpoczyna się od obliczenia długości tablicy x, co realizowane jest przez podzielenie jej rozmiaru w bajtach przez rozmiar pojedynczego elementu. Następnie wywoływana jest funkcja TheilSen\_Estymator, która przyjmuje jako argumenty tablice x i y oraz ich długość. Funkcja zwraca strukturę WynikTheilSena, zawierającą wartości wyrazu wolnego i nachylenia.

Obliczone wartości są formatowane jako tekst za pomocą funkcji sprintf i zapisywane do zmiennej msg, która została zadeklarowana na początku programu. Ostatecznie wiadomość jest przesyłana przez UART za pomocą funkcji HAL\_UART\_Transmit, co umożliwia wyświetlenie wyników na urządzeniu zewnętrznym.

Aby zapewnić poprawne działanie i wyświetlanie wartości zmiennoprzecinkowych na urządzeniu zewnętrznym, konieczne było odpowiednie skonfigurowanie ustawień programu zgodnie z dokumentacją dotyczącą obsługi zmiennych typu float [UM2609].

```
190 /* USER CODE BEGIN 4 */  
191 void HAL_UART_TxCpltCallback(UART_HandleTypeDef *huart);  
192 void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart);
```

**Listing 20.** src/main.c

Te funkcje służą do odbioru danych z mikrokontrolera przez UART za pomocą Tx i Rx.

## 4.2 thil\_sen.c

Plik realizuje procedurę obliczeniową opartą na algorytmie estymacji Theila-Sena, dedykowanym do wyznaczania parametrów regresji liniowej, takich jak współczynnik kierunkowy i wyraz wolny. Algorytm został zaimplementowany z uwzględnieniem optymalizacji pod kątem ograniczonych zasobów sprzętowych mikrokontrolera, co pozwala na efektywne przetwarzanie danych i uzyskiwanie wyników o wysokiej precyzyji.

```
1 #include "theil_sen.h"  
2 #include "stdlib.h"  
3 #include "stddef.h"
```

**Listing 21.** src/theil\_sen.c

Nagłówek theil\_sen.h włącza plik deklarujący funkcję i struktury związane z implementacją metody Theil-Sen. Stdlib.h jest to standardowa biblioteka C, która umożliwia sortowanie za pomocą funkcji qsort. Ostatnia biblioteka stddef.h pozwala na zdefiniowanie rozmiaru tablicy size\_t.

```
5 #define MAX_ELEMENTOW 4095  
6  
7 static int porownaj_floaty(const void* a, const void* b) {  
8     float roznica = (*(const float*)a - *(const float*)b);  
9     return (roznica > 0) - (roznica < 0);  
10 }
```

**Listing 22.** src/theil\_sen.c

Zostaje zdefiniowana wielkość MAX\_ELEMENTOW jako 4095. Funkcja porownaj\_floaty porównuje dwie wartości. Jest używana jako funkcja pomocnicza w qsort do sortowania tablic. Parametry a, b są to wskaźniki typu float, za pomocą których obliczana jest różnica między wartościami.

```
12 static float mediana(float* tablica, size_t dlugosc) {  
13     qsort(tablica, dlugosc, sizeof(float), porownaj_floaty);  
14     if (dlugosc % 2 == 0) {  
15         return (tablica[dlugosc / 2 - 1] + tablica[dlugosc / 2]) / 2.0f;
```

---

```

16     } else {
17         return tablica[dlugosc / 2];
18     }
19 }
```

---

**Listing 23.** *src/theil\_sen.c*

Funkcja mediana oblicza medianę wartości w tablicy liczb zmiennoprzecinkowych. Sortuje tablicę za pomocą qsort (korzysta z funkcji porownaj\_floaty). Mediana obliczana jest zgodnie z matematyczną definicją: Jeśli liczba elementów jest parzysta, zwraca średnią z dwóch środkowych elementów. Jeśli nieparzysta, zwraca środkowy element.

---

```

21 WynikTheilSena TheilSen_Estymator(const float* x, const float* y, size_t
22     dlugosc) {
23
24     float nachylenia[MAX_ELEMENTOW];
25     float wyrazy_wolne[MAX_ELEMENTOW];
26     size_t indeks_nachylen = 0;
27     size_t indeks_wyrazow = 0;
```

---

**Listing 24.** *src/theil\_sen.c*

Funkcja TheilSen\_Estymator oblicza estymację metodą Theila-Sena, przyjmując jako argumenty tablice współrzędnych  $x$  i  $y$  oraz ich długość. W jej wnętrzu deklarowane są tablice nachylenia i wyrazy\_wolne do przechowywania obliczonych wartości, a także zmienne pomocnicze indeks\_nachylen i indeks\_wyrazow, służące do określenia indeksu w dalszych obliczeniach.

---

```

28     for (size_t i = 0; i < dlugosc - 1; i++) {
29         for (size_t j = i + 1; j < dlugosc; j++) {
30             if (x[j] != x[i]) {
31                 if (indeks_nachylen < MAX_ELEMENTOW) {
32                     nachylenia[indeks_nachylen++] = (y[j] - y[i]) / (x[j] - x[i]);
```

---

**Listing 25.** *src/theil\_sen.c*


---

```

42     float mediana_nachylenia = mediana(nachylenia, indeks_nachylen);
```

---

**Listing 26.** *src/theil\_sen.c*

Na listingu 25 i 26 obliczana jest mediana wartości przechowywanych w tablicy nachylenia, która zawiera obliczone nachylenia dla wszystkich par punktów. Funkcja mediana przyjmuje jako argumenty tablicę nachylenia oraz liczbę elementów w niej zapisanych (indeks\_nachylen) i zwraca wartość mediany, przypisywaną do zmiennej mediana\_nachylenia.

---

```

43
44     for (size_t i = 0; i < dlugosc; i++) {
45         if (indeks_wyrazow < MAX_ELEMENTOW) {
46             wyrazy_wolne[indeks_wyrazow++] = y[i] - mediana_nachylenia * x[i]
47         };
48 }
```

---

```
47     }
48 }
49 float mediana_wyrazu = mediana(wyrazy_wolne, indeks_wyrazow);
50
51 WynikTheilSena wynik = {mediana_nachylenia, mediana_wyrazu};
52 return wynik;
```

**Listing 27.** src/theil\_sen.c

Na listingu 27 obliczany jest wyraz wolny dla każdego punktu danych, wykorzystując wcześniej wyznaczoną medianę nachyleń. Dla każdego punktu ( $x[i]$ ,  $y[i]$ ), obliczana jest wartość  $y[i]$  - mediana nachylenia  $x[i]$ , która zostaje zapisana w tablicy `wyrazy_wolne`, o ile jej maksymalna pojemność nie zostanie przekroczena. Następnie wyznaczana jest mediana z tablicy `wyrazy_wolne`, która reprezentuje wyraz wolny prostej regresji. Funkcja zwraca wynik w postaci struktury `WynikTheilSena`, zawierającej medianę nachyleń jako nachylenie prostej oraz medianę wyrazów wolnych jako punkt przecięcia z osią y.

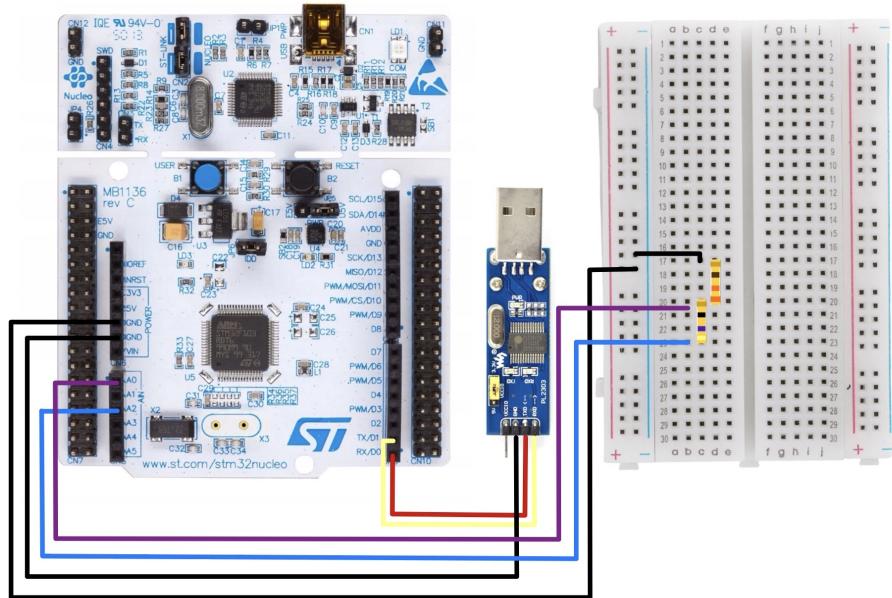
#### 4.2.1 thil\_sen.h

Jest to kod do implementacji metody Theila-Sena.

```
1 #ifndef THEIL_SEN_H
2 #define THEIL_SEN_H
3
4 #include <stddef.h>
5
6 typedef struct {
7     float nachylenie;
8     float wyraz;
9 } WynikTheilSena;
10
11 WynikTheilSena TheilSen_Estymator(const float* x, const float* y, size_t
12     dlugosc);
13
14 #endif
```

**Listing 28.** src/theil\_sen.h

Zawiera definicję struktury `WynikTheilSena` do przechowywania nachylenia i wyrazu wolnego linii regresji oraz deklarację funkcji `TheilSen_Estymator`, która oblicza parametry regresji dla podanych danych  $x$  i  $y$ .



Rysunek 9. Układ pomiarowy rezistory

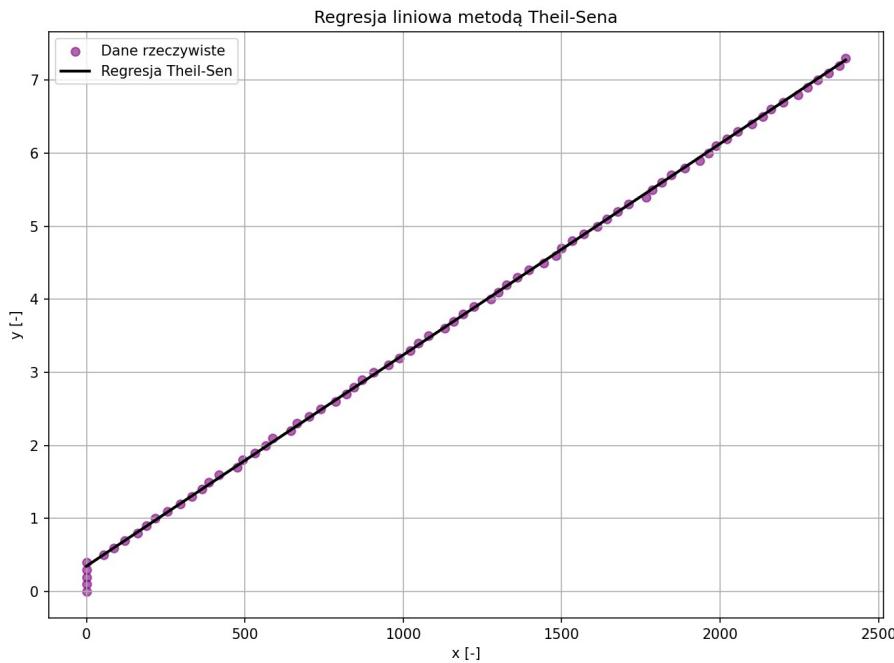
## 4.3 Układ z rezystancyjnym dzielnikiem napięcia

### 4.3.1 Wykres i wynik

Po wykonaniu programu uzyskano dane wyjściowe. Na podstawie tych danych wygenerowano wykres ilustrujący zależność wartości ADC/330 od wartości ADC (y od x). Na wykresie znajduje się linia regresji, wyznaczona metodą Theila-Sena. W celu poprawy czytelności wykresu liczba punktów została ograniczona do około 80, co stanowi reprezentatywną próbkę danych. Taka redukcja liczby punktów umożliwia lepszą interpretację wyników i wizualizację, bez utraty istotnych informacji.

Obliczenia metodą Theil-Sen nachylenia oraz punktu przecięcia z osią y zostały przeprowadzone dla wybranych punktów danych przy użyciu wcześniej opracowanego programu w środowisku Python. Z uwagi na przejrzystość wykresu obliczenia zostały przeprowadzone na ograniczonych wcześniej danych.

Wyniki uzyskane z programu STM32CubeIDE są bardzo zbliżone do tych otrzymanych w Pythonie. Niewielkie różnice mogą wynikać z faktu, że tym pierwszym obliczenia są wykonywane na pełnej próbce danych, obejmującej 4096 punktów. W przeciwnieństwie do tego, w wersji programu w Pythonie, obliczenia zostały przeprowadzone na ograniczonej liczbie punktów (około 80), co może wpływać na dokładność i precyzję uzyskanych wyników. Mimo tych drobnych rozbieżności wyniki pozostają zbliżone, co świadczy o poprawności obu analizy, z uwzględnieniem różnic w liczbie punktów danych użytych do obliczeń.



**Rysunek 10.** Wykres ADC/330 od ADC

**Metoda Theil-Sena:**  
**Przesunięcie (wynik.wyraz):** 0.3491  
**Nachylenie (wynik.nachylenie):** 0.0029  
**Błąd średniokwadratowy:** 0.0033

**Rysunek 11.** Wynik ADC/330 od ADC w środowisku Python

Expression	Type	Value
<code>(*) value_adc</code>	uint32_t	2384
<code>(*) i</code>	uint32_t	4096
<code>(*) wynik.wyraz</code>	float	-1.49822044
<code>(*) wynik.nachylenie</code>	float	0.00355871883

**Rysunek 12.** Wynik ADC/330 od ADC

#### 4.3.2 Przeliczenie wartości

Uzyskane dane są przedstawione w postaci wartości bitowych. Aby przeliczyć je na wartości napięcia wyrażone w [V] oraz prądu w [mA], należy zastosować poniższe zależności:

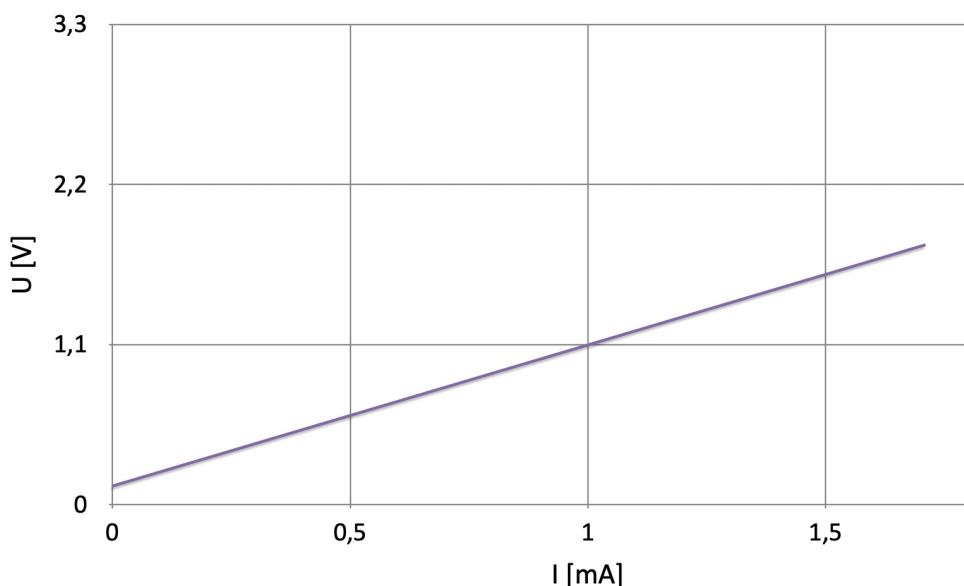
$$U = \frac{ADC}{4096} * 3,3[V]$$

$$I = \frac{ADC/330}{4096} * 1000[mA]$$

gdzie:

- 4096 to maksymalna wartość z przetwornika DAC w mikrokontrolerze, wynikająca z 12-bitowej rozdzielczości (zakres od 0 do  $2^{12} - 1$ ),
- 3.3 V to maksymalne napięcie przetwornika,
- 1000 to zamiana z A na mA.

Widać, że charakterystyka jest liniowa, co zgadza się z teoretycznymi założeniami dla rezystorów.



Rysunek 13. Wykres U od I

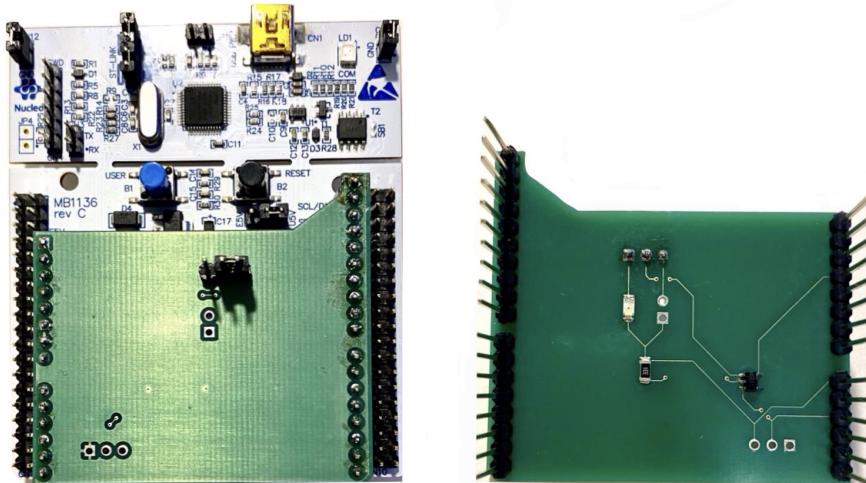
## 4.4 Układ z diodą i rezystorem

Celem tego fragmentu pracy jest zaprezentowanie układu testowego, który umożliwia ocenę poprawności działania implementacji metody Theil-Sen dla przebiegu nieliniowego.

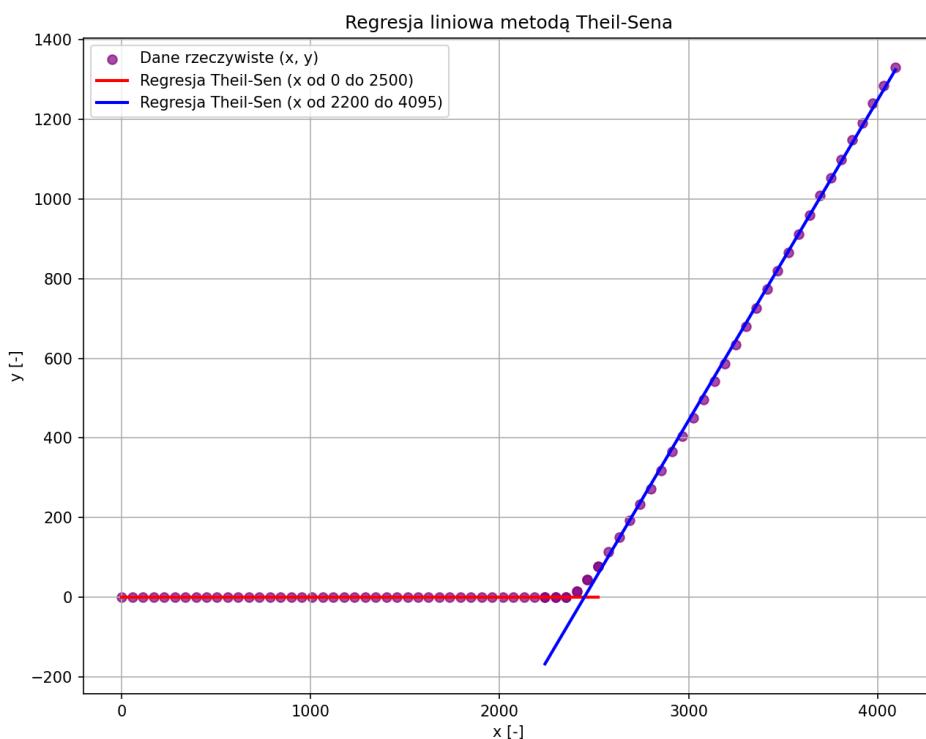
### 4.4.1 Wykres i wynik

W przeprowadzonym eksperymencie stworzono wykres porównujący sygnał odebrany przez przetwornik analogowo-cyfrowy (ADC) z sygnałem zadanym, generowanym przez przetwornik cyfrowo-analogowy (DAC). Na podstawie otrzymanych danych stworzono wykres, który pozwala wizualnie ocenić, czy układ działa zgodnie z założeniami projektowymi oraz, czy przetwarzanie sygnałów przebiega poprawnie. Analiza wykresu umożliwia szybkie wykrycie ewentualnych błędów w działaniu układu oraz ocenę jakości pomiarów.

Na wykresie przedstawiono trzy estymacje regresji liniowej dla różnych zakresów danych. Pierwsza estymacja (czerwona linia) została obliczona dla przedziału po x od 0 do 2500, czyli w zakresie, w



Rysunek 14. Płytki PCB



Rysunek 15. Wykres ADC od DAC

którym charakterystyka diody pozostaje płaska. Druga estymacja (niebieska linia) obejmuje zakres od 2200 do 4095, gdzie charakterystyka diody liniowo rośnie. Zakresy te celowo nałożono na siebie, co umożliwia wyznaczenie ich punktu przecięcia.

Dodatkowo można zauważyc, że wykres jest zgodny z teoretyczną charakterystyką diody, co potwierdza poprawność zarówno układu, jak i zastosowanej metody. Otrzymane dane są wiarygodne, a układ działa zgodnie z oczekiwaniemi.

```

Metoda Theil-Sena:
Czerwona prosta (zakres od 0 do 2500)
Przesunięcie (wyraz wolny): -0.00
Nachylenie (wynik.nachylenie): 0.00

Niebieska prosta (zakres od 2200 do 4096)
Przesunięcie (wyraz wolny): -1969.82
Nachylenie (wynik.nachylenie): 0.80

Punkt przecięcia: x = 2448.29, y = 0.01

```

Rysunek 16. Wyniki ADC od DAC w środowisku Python

Podobnie jak w poprzednim przypadku, wybrano reprezentatywną próbkę danych, składającą się z około 80 punktów pomiarowych. Na podstawie tych danych, przy użyciu algorytmu zaimplementowanego we wcześniejszej części pracy w środowisku Python, obliczono wartości nachylenia oraz wyrazu wolnego.

Wyraz wolny i nachylenie w przypadku pierwszej regresji (linia czerwona) wynoszą 0, co jest zgodne z założeniami. Charakterystyka diody w tym zakresie pozostaje płaska, co oznacza brak obserwowalnego wzrostu. Druga część analizy obejmuje przedział od 2200 do 4095, w którym charakterystyka diody rośnie liniowo. Dodatkowo wyznaczono punkt przecięcia prostych, co umożliwia określenie napięcia progowego diody  $U_f$ .

Aby osiągnąć też ten efekt również w środowisku STM32, niezbędne będzie dokonanie drobnych modyfikacji w pliku theil\_sen.c. Zmiany te umożliwią uwzględnienie podzielnie zakresu na dwie części.

```
5 #define MAX_ELEMENTOW 2500
```

Listing 29. src/theil\_sen\_PCB.c

W pierwszej kolejności dokonana zostanie zamiana wartości, przyjmując maksymalną liczbę elementów równą 2500. Oznacza to, że program uwzględnia jedynie wartości do 2500, czyli zakres, w którym charakterystyka diody pozostaje płaska i nie wykazuje wzrostu.

Wyniki uzyskane w środowisku STM32 są zgodne z danymi otrzymanymi w środowisku Pythona, co potwierdza, że implementacja algorytmu została przeprowadzona poprawnie.

W drugim zakresie, obejmującym przedział od 2200 do 4095, wartość MAX\_ELEMENTÓW zostaje przywrócona do 4095.

Expression	Type	Value
<code>↳ value_adc</code>	<code>uint32_t</code>	1551
<code>↳ i</code>	<code>uint32_t</code>	4096
<code>↳ wynik.wyraz</code>	<code>float</code>	0
<code>↳ wynik.nachylenie</code>	<code>float</code>	0

Rysunek 17. Wyniki ADC od DAC dla wartości od 0 do 2500

```
29 | for (size_t i = 2200; i < dlugosc - 1; i++) {
30 |   for (size_t j = i + 1; j < dlugosc; j++) {
```

Listing 30. src/theil\_sen\_PCB.c

```
44 | for (size_t i = 2200; i < dlugosc; i++) {
```

Listing 31. src/theil\_sen\_PCB.c

Na listingu 30 i 31 początkowa wartość zmiennej "i" została ustawiona na 2200, co powoduje, że obliczenia będą przeprowadzane w zakresie od 2200 do 4095.

Expression	Type	Value
<code>↳ value_adc</code>	<code>uint32_t</code>	1551
<code>↳ i</code>	<code>uint32_t</code>	4096
<code>↳ wynik.wyraz</code>	<code>float</code>	-1841.92468
<code>↳ wynik.nachylenie</code>	<code>float</code>	0.85296154

Rysunek 18. Wyniki ADC od DAC dla wartości od 2200 do 4095

Jak widać, nachylenie jest bliskie 0,85, a punkt przecięcia będzie równy około -1840. W obu przypadkach otrzymywane wyniki są zbliżone, co świadczy o zgodności implementacji oraz poprawności działania zarówno wersji kodu uruchomionej na mikrokontrolerze STM32, jak i tej działającej w środowisku Pythona. Taka zgodność wyników sugeruje, że proces obliczeniowy został przeprowadzony prawidłowo, a algorytm działa zgodnie z oczekiwaniemi.

## 4.5 Napotkane problemy

Podczas implementacji kodu ograniczona pamięć mikrokontrolera uniemożliwiła przeprowadzenie najdokładniejszych z możliwych obliczeń.

Dla równania:  $x[i] = \text{value\_adc} / r;$

Wynik dzielenia `value_adc` przez rezystancję równą 330 jest zaokrąglany do całkowitych wartości w zakresie od 0 do 7, co obniża precyzję obliczeń.

Aby zwiększyć dokładność, można zmodyfikować działanie, dzieląc przez liczbę zmienoprzecinkową:  $r = 330.0$ ;

W tym przypadku uzyskiwana jest dokładność nawet do 11 miejsc po przecinku, np.  $x[1] = 0.00606060587$ . Jednak w kodzie `theil_sen.c` występują wartości typu `float`, które mają precyzję do 8 miejsc po przecinku, co oznacza, że nie są wystarczające do dalszego przetwarzania danych,

zwłaszcza że metoda wymaga bardziej złożonych obliczeń, generujących jeszcze większą liczbę cyfr po przecinku.

Logicznym rozwiązaniem tego problemu byłoby zamiana wartości na typ double, który oferuje precyzję do 16 miejsc po przecinku. Niestety, ograniczona pamięć mikrokontrolera nie pozwala na przetwarzanie danych o tak dużej precyzji.

Ostatecznie precyzja oferowana przez typ float jest wystarczająca do realizacji dokładnych obliczeń w ramach bieżącego zastosowania. Jednak pozostawia to pole do dalszego rozwoju i ulepszania tego rozwiązania.



## Rozdział 5

### Podsumowanie

Niniejsza praca dotyczy implementacji estymatora Theil-Sen na mikrokontrolerze STM32, będącego metodą obliczania linii regresji odporną na wartości odstające. Kluczową zaletą tej metody jest jej niezawodność w przypadku danych pomiarowych, które mogą być zakłócone lub zniekształcone przez szum, co jest częstym zjawiskiem w pomiarach opartych na czujnikach. Dodatkowo estymator Theil-Sen został porównany z klasyczną metodą najmniejszych kwadratów, aby uwypuklić jego wady i zalety oraz zidentyfikować odpowiednie obszary jego zastosowań.

Przegląd literatury wskazuje, że metoda Theila-Sena cieszy się dużym zainteresowaniem w różnych dziedzinach analizy danych. W prognozowaniu cen Bitcoina, gdzie występuje wysoka zmienność, wykazuje ona większą odporność na wartości odstające w porównaniu do innych metod, co czyni ją bardziej efektywną w warunkach szybko zmieniającego się rynku [[bitcoin2021](#)]. Z kolei w artykule z 2024 roku estymator Theil-Sen został zastosowany w analizie danych hydrologicznych i atmosferycznych, gdzie wykazał lepszą skuteczność niż metoda najmniejszych kwadratów [[zahra2024](#)]. Szczególnie wyróżniał się odpornością na wartości odstające, zapewniając bardziej wiarygodne wyniki. Dodatkowo charakteryzował się mniejszym błędem obliczeniowym w porównaniu do innych metod regresyjnych.

Przeprowadzone w pracy testy potwierdzają, że estymator Theil-Sen jest lepszym wyborem do zadań, gdzie występują skrajne odchylenia w danych, podczas gdy klasyczna metoda najmniejszych kwadratów może prowadzić do zniekształconych wyników, co jest zgodne z obecnym stanem wiedzy. Krytyczna analiza wyników wskazuje, że wybór metody zależy od charakterystyki danych wejściowych oraz wymaganej odporności na błędy pomiarowe.

W pracy zaimplementowano estymator Theil-Sen w języku C z użyciem środowiska STM32CubeIDE oraz zaprojektowano układ pomiarowy z płytą PCB, co pozwoliło na przeprowadzenie testów w realnych warunkach. Testy te wykazały, że mimo ograniczonych zasobów mikrokontrolera STM32, możliwe jest skuteczne realizowanie zaawansowanych algorytmów statystycznych. Wartością dodaną projektu jest wykazanie, że nawet systemy wbudowane, o ograniczonej pamięci i mocy obliczeniowej, mogą być używane do realizacji zaawansowanych algorytmów, co wpisuje się w aktualne kierunki rozwoju technologii.

## *Rozdział 5. Podsumowanie*

---

Podsumowując, celem pracy było opracowanie i implementacja algorytmu Theil-Sen w środowisku STM32CubeIDE, co zostało osiągnięte. Implementacja została przeprowadzona poprawnie, algorytm działa zgodnie z założeniami, a wyniki testów potwierdzają jego efektywność. Ponadto, praca wykazała użyteczność estymatora Theil-Sen w systemach wbudowanych, szczególnie w zastosowaniach wymagających analizy danych z płyt PCBA. Mimo pewnych ograniczeń związanych z pamięcią i mocą obliczeniową udało się uzyskać wyniki o wysokiej jakości, co dowodzi, że metoda ta jest użytecznym narzędziem w takich systemach. Cel pracy został osiągnięty, a założenia dotyczące porównania metod, implementacji oraz analizy wyników zostały zrealizowane. W przyszłości warto rozważyć dalszą optymalizację algorytmu oraz rozszerzenie implementacji o dodatkowe metody statystyczne.

# Spis rysunków

1	Prof. dr. Henri Theil, autor Gemeente Rotterdam , r. 1966 . . . . .	11
2	Prof. dr. Pranab Kumar Sen, autor Tom Fuldner, r. 2011 . . . . .	12
3	Wykres metody Theil-Sen . . . . .	13
4	Wykres metody najmniejszych kwadratów . . . . .	15
5	Wykres obu metod . . . . .	15
6	Wizualizacja metody Theil-Sen . . . . .	20
7	Wizualizacja metody najmniejszych kwadratów . . . . .	21
8	Wizualizacja obu metod . . . . .	23
9	Układ pomiarowy rezystory . . . . .	31
10	Wykres ADC/330 od ADC . . . . .	32
11	Wynik ADC/330 od ADC w środowisku Python . . . . .	32
12	Wynik ADC/330 od ADC . . . . .	32
13	Wykres U od I . . . . .	33
14	Płytki PCB . . . . .	34
15	Wykres ADC od DAC . . . . .	34
16	Wyniki ADC od DAC w środowisku Python . . . . .	35
17	Wyniki ADC od DAC dla wartości od 0 do 2500 . . . . .	36
18	Wyniki ADC od DAC dla wartości od 2200 do 4095 . . . . .	36