

# Generator sygnału 1 kHz

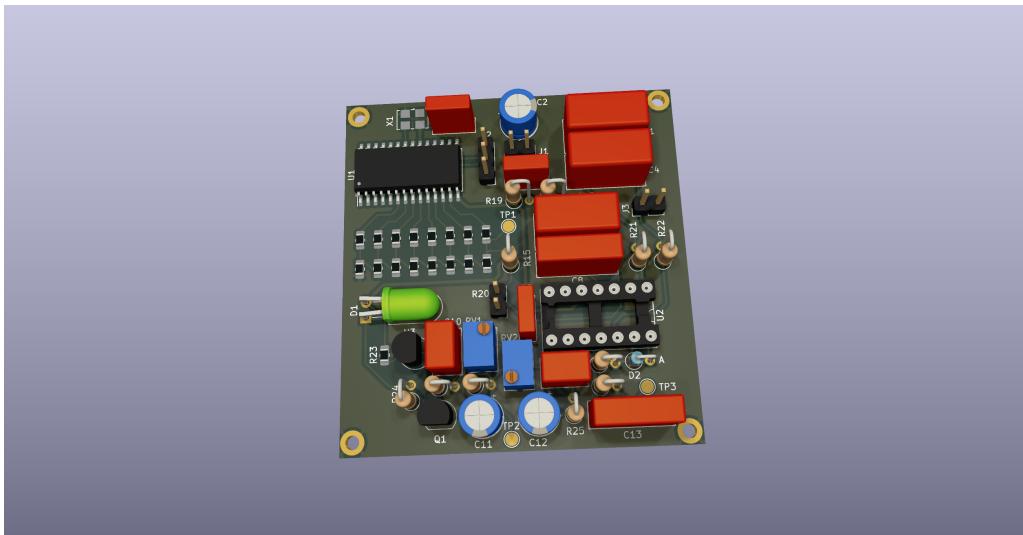
## Raport Techniczny

Szymon Januszek, szymon\_j@tutanota.com

Kwiecień 2023

## Streszczenie

Dokument zawiera opis budowy i działania generatora przebiegu 1 kHz.



# Spis treści

<b>1 Bezpośrednia synteza cyfrowa</b>	<b>3</b>
1.1 Mikrokontroler . . . . .	3
1.2 Przetwornik Cyfrowo-Analogowy . . . . .	3
1.3 Analiza zniekształceń przetwornika . . . . .	4
<b>2 Kształtowanie</b>	<b>4</b>
<b>3 Kontrola Amplitudy</b>	<b>5</b>
3.1 Pomiar amplitudy . . . . .	5
3.2 Element sprzęgający . . . . .	5
3.3 Źródło odniesienia . . . . .	6
3.4 Dostrajanie . . . . .	6
<b>4 Oprogramowanie</b>	<b>7</b>
4.1 Środowisko Deweloperskie . . . . .	8
4.2 Kod źródłowy . . . . .	8
<b>5 Eksperymenty</b>	<b>10</b>
<b>6 Schemat i płytka drukowana</b>	<b>10</b>

# 1 Bezpośrednia synteza cyfrowa

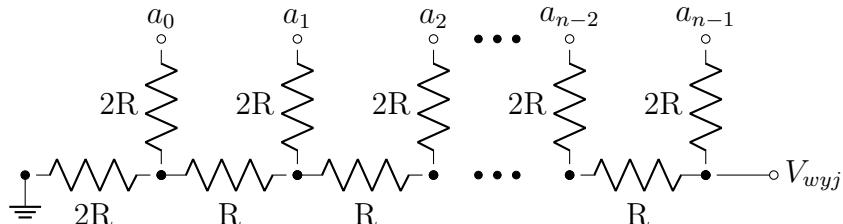
Aby wygenerować maksymalnie stabilny i dokładny względem częstotliwości sygnał, zdecydowałem się na wykorzystanie techniki **Bezpośredniej Syntezy Cyfrowej**, gdzie mikroprocesor wraz z układem Przetwornika Cyfrowo-Analogowego (DAC) generuje sygnał zbliżony do sygnału oczekiwanej. Umożliwia to zastosowanie wysokiej jakości generatora kwarcowego, o precyzyji rzędu 50 ppm<sup>1</sup>, w całym zakresie temperatur. Oznacza to, iż częstotliwość uzyskanego sygnału nie będzie odbiegać o więcej niż  $\pm 50 \text{ mHz}$  od wartości oczekiwanej.

## 1.1 Mikrokontroler

Jako główny mikrokontroler wybrany został **AVR32DA28**<sup>[2]</sup> należący do rodziny 8-bitowych układów AVR. W porównaniu ze starszymi generacjami, producent dopuszcza pracę przy częstotliwościach przekraczających 8 MHz z napięciem 3 V. Jednocześnie dokonana została unifikacja modelu pamięci, co okaże być się przydatne przy pisaniu oprogramowania([4](#)).

## 1.2 Przetwornik Cyfrowo-Analogowy

Do implementacji przetwornika zrealizowany został układ drabinki R-2R (Rys.[1](#)), umożliwiając łatwą zmianę wartości liczbowej, przedstawionej jako liczba binarna na pinach mikrokontrolera, na ułamek napięcia zasilania.



Rysunek 1: Schemat drabiny R-2R [1]

Napięcie na węźle wyjściowym przetwornika można opisać w następujący sposób:

$$V_{wyj} = V_{ref} \times \frac{a_0 \times 2^0 + a_1 \times 2^1 + a_2 \times 2^2 + \dots + a_{N-1} \times 2^{N-1}}{2^N}$$

Przy  $N = 8$ , przetwornik ten pozwala na uzyskanie 256 różnych wartości. Dla  $V_{ref} = 3 \text{ V}$ , oznacza to, iż różnica pomiędzy dwoma dowolnymi stopniami wynosi ok.  $11.72 \text{ mV}$ .

Co ciekawe, okazuje się, iż impedancja wyjściowa takiego układu jest stała dla każdego poziomu [2](#).

<sup>1</sup>Części na milion

<sup>2</sup>Wynika to z Twierdzenie Thévenina, zakładającego zerową impedancję źródła oraz mikrokontrolera. W rzeczywistości ich wartości są znacznie mniejsze od wartości R.

Aby zachować użyteczność najniższych bitów, precyza wykorzystanych rezysto-rów musi być lepsza niż  $\frac{1}{2^N} \times 100\% \approx 0.3\%$ . Stąd, wykorzystane zostały rezistory przeznaczone do montażu powierzchniowego o precyzyji 0.1 %.

### 1.3 Analiza znieksztalceń przetwornika

Sygnal generowany w ten sposob nie opowiada jednak perfekcyjnej sinusoidzie (Rys. 2). Obecne jest tzw. znieksztalcenie kwantyzacji, wynikajace z faktu, iż sygnal zlo-zony jest z serii dyskretnych wartosci. Jednak jeśli dokladnie przyjrzymy się różnicy pomiedzy uzyskanym sygnalem, a idealna sinusoida, to okaże się, że pierwsza skla-dowa tej różnicy ma częstotliwość  $f_1 = f_0 \times N_{smp}$ .

Co więcej, stosunek mocy całego sygnału do mocy wyzszych składowych dla konwertera z N-bitami wynosi [3]

$$SQR = 1.76 + 6.02 \times N(dB) \quad (1)$$

Przy  $N = 8$  oraz  $N_{smp} = 1024$  oznacza to iż całkowita moc znieksztalceń wzgle-dem sygnału fundamentalnego wynosi  $-50$  dB, oraz jest rozłożona w paśmie mega-hercowym.

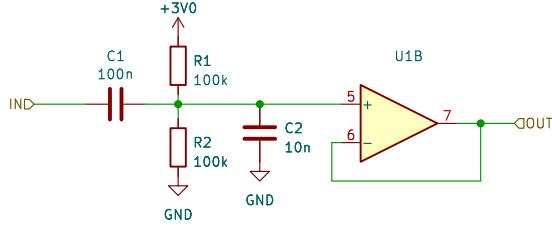


Rysunek 2: Przebieg z widocznym błędem kwantyzacji

## 2 Kształtowanie

Mimo wszystko nalezy usunąć składowe wyzszej częstotliwości. Funkcje tą realizuje kondensator C2 (Rys. 3).

Dodatkowo, rzeczywiste wzmacniacze operacyjne nie są w stanie generować sygnałów bardzo bliskich ( $|V_{wyj} - V_{zasi}| < 300$  mV) szyn zasilania. Tłumienia tego dokonują komponenty pasywne C1, R1, R2, zachowując wartość średkową sygnału na poziomie 1.5 V.



Rysunek 3: Schemat filtru

### 3 Kontrola Amplitudy

Ostatnią zmienną, którą należy rozpatrzyć jest amplituda sygnału wyjściowego. Na tym etapie poziom sygnału jest wyższy od oczekiwanej wartości 0.5 V. Potrzebna jest więc metoda dokładnego tłumienia, nie wprowadzająca dodatkowych zniekształceń i zakłóceń.

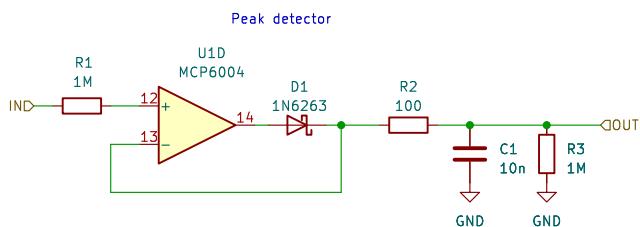
Rozwiązanie oparte o precyzyjny potencjometr pozostaje jednak wrażliwe na zmianę absolutnej wartości napięcia zasilania, czy też dryft elementów związanych ze zmianą temperatury.

Tak więc, niezbędny jest układ aktywnie korygujący amplitudę sygnału. Cyfrowa implementacja takiego rozwiązania nie jest możliwa, ponieważ przetwornik DAC nie posiada wystarczającej głębokości bitowej. Stąd też, wybrałem rozwiązanie w pełni analogowe.

Przy użyciu wzmacniaczy operacyjnych możemy uzyskać pętlę sprzężenia zwrotnego, korygującą amplitudę sygnału względem stałego napięcia odniesienia. Do realizacji tego rozwiązania potrzebujemy następujących elementów.

#### 3.1 Pomiar amplitudy

Konieczny jest pomiar amplitudy sygnału wyjściowego. Do tego celu wykorzystany został tzw. Detektor szczytowy (Rys. 4).



Rysunek 4: Przykładowy schemat połówkowego detektora szczytowego

#### 3.2 Element sprzęgający

Wymagany jest układ umożliwiający tłumienie sygnału wyjściowego względem sygnału sterującego:

$$V_{wyj} \propto \frac{V_{wej}}{V_{ster}} \quad (2)$$

Fizyczna implementacja takiego układu okazuje się być nietrywialna. Możliwy jest układ realizujący operacje mnożenia wartości sygnałów. Jego realizacja jest jednak złożona i wymaga relatywnie dużo części.

Alternatywna metoda wykorzystuje element zmieniający swój opór elektryczny względem innego czynnika. Przykładem historycznej realizacji takiego układu jest zastosowanie niewielkiej lampy żarowej, której opór włókna jest wprost proporcjonalny do jego temperatury.

Istnieje jednak znacznie prostsze i wydajniejsze rozwiązanie.

Popularnym elementem o zmiennym oporze jest fotorezystor. W przeciwnieństwie do np. tranzystorów jFET, przy stałym natężeniu światła, zachowuje się on jak faktyczny rezistor, tzn. prąd przepływający jest wprost proporcjonalny do przyłożonego napięcia  $I \propto V$ , tym samym jego zastosowanie nie wprowadzi zniekształceń mogących wynikać z nieliniowością elementów półprzewodnikowych.

Tak więc, łącząc w obudowie światłoszczelnej fotorezystor wraz z diodą LED, uzyskamy układ o zmiennym oporze elektrycznym, kontrolowany przez prąd diody<sup>3</sup>.

$$R \propto \frac{1}{I_{Led}} \quad (3)$$

Umożliwia to łatwą implementację układu sprzężenia (Eq. 2).

Powyższy model pomija fakt, że w rzeczywistości opór fotorezystora zależy od natężenia padającego światła  $R = \left(\frac{1}{\phi}\right)^A$ . Nie stanowi jednak to dużego problemu, ponieważ układ ten znajduje się w pętli sprzężenia zwrotnego.

$$R \propto \frac{1}{(I_{Led})^A} \quad (4)$$

### 3.3 Źródło odniesienia

Na tym etapie potrzebne jest jedynie stabilne oraz dostrajalne źródło napięcia odniesienia. Do tego celu w zupełności wystarczy układ LM4041 wraz z precyzyjnym potencjometrem.

### 3.4 Dostrajanie

Szczególnie trudnym okazało się zapewnienie stabilnej pracy układu. Wartości elementów w pętli kompensacji zostały w większości dobrane empirycznie. Konieczne było zastosowanie dużej pojemności kondensatorów elektrolitycznych, ze względu na długi czas propagacji sygnału przez resztę układu.

---

<sup>3</sup>Zakładam  $\phi \propto I$

## 4 Oprogramowanie

W pliku `sine_tab.cpp` znajduje się definicja tabeli wartości funkcji sinus. Składa się ona z  $N_{smp} = 1024$  próbek, a wartość  $i$ -tej próbki zdefiniowana jest w następujący sposób:

$$f(i) = 2^N + \left\lfloor (2^N - 1) \times \sin \left( \frac{2\pi \times i}{N_{smp}} \right) \right\rfloor \quad (5)$$

$$f(i) = 128 + \left\lfloor 127 \times \sin \left( \frac{2\pi \times i}{1024} \right) \right\rfloor \quad (6)$$

Każda próbka jest 8-bitową liczbą całkowitą z zakresu  $\{1 \dots 255\}$ , jednocześnie tabela przedstawia cały jeden okres funkcji sinus, poza wartością  $\sin(2\pi)$ . Następną próbkę stanowi wartość  $\sin(0)$ . Dzięki temu reprodukowany sygnał nie zawiera błędów co okres.

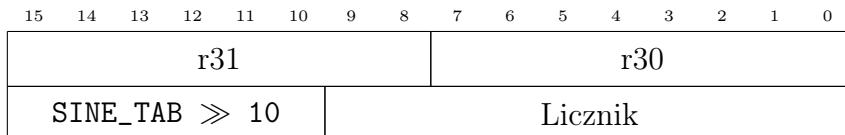
Zadaniem oprogramowania mikrokontrolera pozostaje wczytanie kolejnych wartości z pamięci, a następnie przestawienie pinów w odpowiedni stan. Wykorzystując 8-bitową naturę architektury AVR możliwe jest ustawienie stanu całego rejestru (tj. grupy 8 pinów) poprzez pojedynczy zapis do obszaru peryferialnego pamięci (Linia 34)

W tym miejscu napotykamy jednak pewien problem - synteza powyższego sygnału wymaga produkcji ponad miliona próbek na sekundę, tym czasem zastosowany mikroprocesor jest przystosowany do pracy przy częstotliwości nie przekraczającej 24 MHz. Przy zastosowanym generatorze kwarcowym o częstotliwości pracy 12.288 MHz oznacza to, że mamy dokładnie **12** cykli maszynowych na wygenerowanie następnej próbki. Co więcej, aby zachować precyzję częstotliwości musimy wykorzystać wszystkie 12 cykli za każdym razem. Stąd też, cały algorytm zaimplementowany został bezpośrednio w assembly. Pozwala to na dokładną kontrolę nad czasem wykonania całego cyklu, a jednocześnie umożliwia szereg optymalizacji będących poza zasięgiem kompilatora.

Kolejne bajty pobierane są przez instrukcję `ld` (Linia 31), która jednocześnie inkrementuje wskaźnik tworzony przez parę rejestrów `r30` i `r31`.

Problemem pozostaje jedynie wydajne zawijanie wskaźnika po dotarciu do 1024-tej wartości. Ponownie wykorzystując 8-bitową nature układów AVR, możliwe jest bardzo wydajne rozwiążanie tego problemu.

Jednym z atrybutów udostępnianych przez kompilator GCC jest `__attribute__((aligned (P)))`, który pozwala na wymuszenie wyrównania pozycji pierwszego bajtu danej zmiennej do wielokrotności  $P$ . Oznacza to, że pozycja tablicy w pamięci przyjmuje postać  $P \times 1024$ .



Rysunek 5: Schemat rozkładu bitów we wskaźniku

Tak więc, niższy bajt 16-bitowego wskaźnika będzie ulegał stałym zmianom na całej swojej długości, natomiast w bajcie wyższym, dzięki wyrównaniu tabeli, zmieniać się będą jedynie dwa najniższe bity (Rys. 5).

Zostaje tylko nie pozwolić na zmianę stanu 6 najwyższych bitów wskaźnika. Możemy to osiągnąć używając sprytnej sztuczki, zajmującej łącznie 2 cykle zegarowe (Linia 37).

$$r31' = ((r31 \wedge 11_{bin}) \vee r20) \quad (7)$$

Po pierwsze zerujemy sześć najwyższych bitów górnego rejestru instrukcją ANDI z liczbą  $3_{dec}$ . Następnie instrukcja OR przywraca poprawny stan wyższych bitów, które na początku działania programu zostały zapisane w rejestrze 20 (Linijka 25).

Powyższe podejście implementacyjne spełnia postawione wcześniej wymagania dotyczące szybkości działania.

## 4.1 Środowisko Deweloperskie

Wykorzystałem zintegrowane środowisko [PlatformIO](#). Zapewnia ono pełną integrację otwartego zestawu narzędzi z edytorem VSCode.

## 4.2 Kod źródłowy

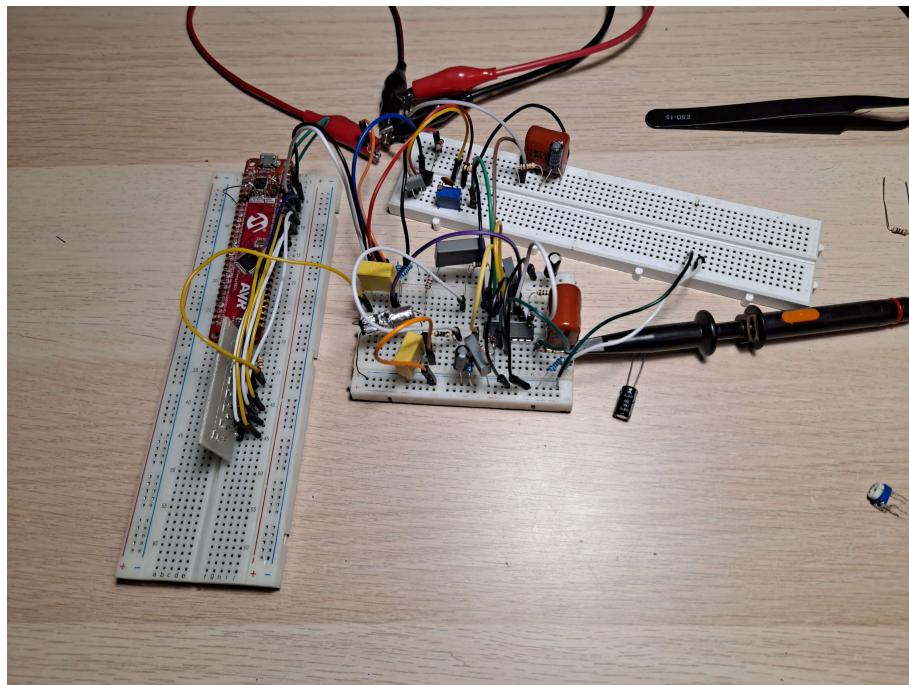
```

1 ;© Szymon Januszek 2023
2 #include <avr/io.h>
3 #include "config.h"
4
5 .extern SINE_TAB
6 .global main
7
8 main:
9     ;Wyzeruj rejestr 1
10    eor r1, r1
11
12    ;użyj wejścia zegarowego
13    ldi r16, 0xD8; MAGICZNA WARTOŚĆ do odblokowywania konfiguracji
14    ldi r17, 0x03; CLKOUT=0 / CLKSEL=EXTCLK
15    out CPU_CCP, r16; odblokuj konfiguracje systemową
16    sts CLKCTRL_MCLKCTRLA, r17; użyj zewnętrznego zegara
17
18    out CPU_CCP, r16
19    sts CLKCTRL_MCLKCTRLB, r1; wyłącz skalowanie zegara
20
21    ldi r16, 0xFF
22    out PORT_DIR r16 ;ustaw piny na wyjście
23    out PORT_OUT, r1
24
25    ;wczytaj wskaźnik SINE_TAB do pary rejestrów Z (r30, r31)
26    ldi r20, hi8(SINE_TAB)

```

```
27      mov r31, r20
28      mov r30, r1
29
30  loop:
31      ;pobierz następny bajt z pamięcia oraz zwięksuj wskaźnik
32      ld r2, Z+
33
34      ;wyślij nowe dane na pin'y
35      out PORT_OUT, r2
36
37      ;zamknij cykl licznika
38      andi r31, 0x03
39      or r31, r20
40
41      ;dopełnij ilość instrukcji do dokładnie 12 cykli maszynowych
42      nop
43      nop
44      nop
45      nop
46
47      rjmp loop
```

## 5 Eksperymenty



Rysunek 6: Złożony układ eksperymentalny

Przedstawiony układ został przetestowany z następującymi różnicami względem schematu:

- Mikrokontroler: ATMega4809
- Rezystory drabiny R-2R:  $10\text{ k}\Omega$ ,  $20\text{ k}\Omega$

Układ ATMega4809 zachowuje pełną kompatybilność kodu źródłowego z AVR32DA28, gdyż oba mikrokontrolery bazują na tej samej wersji architektury AVR.

Generator w czasie pracy pobierał  $\approx 7\text{ mA}$ . Amplituda była poprawnie kontrolowana oraz zachowywała stabilność przy zmiennym obciążeniu w zakresie  $1\text{ k}\Omega - 10\text{ k}\Omega$ . Układ stabilizował się po około 1-2 sekundach od zmiany obciążenia.

Użyty oscyloskop nie pozwolił na użyteczny pomiar poziomu zniekształceń.

## 6 Schemat i płytka drukowana

Schemat oraz płytka zostały wykonane przy użyciu otwartego programu KiCAD. Aby zachować czytelność, schemat podzielony jest cztery oddzielne karty. Płytki zostały poprowadzona na dwóch warstwach. Ze względu na brak dostępnosci niektórych części w obudowach do montażu przewlekanego, nie możliwe było wykonanie płytka jednowarstwowej bez konieczności lutowania części z obu stron.

## Źródła

- [1] CC BY-SA 2.0. URL: <https://commons.wikimedia.org/wiki/File:R2r-ladder.png>.
- [2] Microchip corp. *AVR32DA Final Data Sheet*. URL: <https://ww1.microchip.com/downloads/aemDocuments/documents/MCU08/ProductDocuments/DataSheets/AVR32DA28-32-48-Data-Sheet-40002228B.pdf>.
- [3] Analog Devices Inc. Ken Gentile. *The Effect of DAC Resolution on Spurious Performance*. URL: <https://www.analog.com/media/en/training-seminars/design-handbooks/Technical-Tutorial-DDS/Section4.pdf>.



Rysunek 7: Kot autora wspomagający proces twórczy