PROCESORY SYGNAŁOWE

LABOLATORIUM #02 Generacja sygnału sinusoidalnego

Mateusz Kłosiński

July 6, 2025

1 Kodowanie sygnału w modulacji PCM

Modulacja impulsowo-kodowa (PCM) służy do kodowania analogowych sygnałów audio na postać cyfrowa w celu przesłania ich z mikrokontrolera STM32 do kodeka audio. Rysunek 1 przedstawia ten proces, pokazujac wejściowy sygnał sinusoidalny i odpowiadajacy mu sygnał PCM.

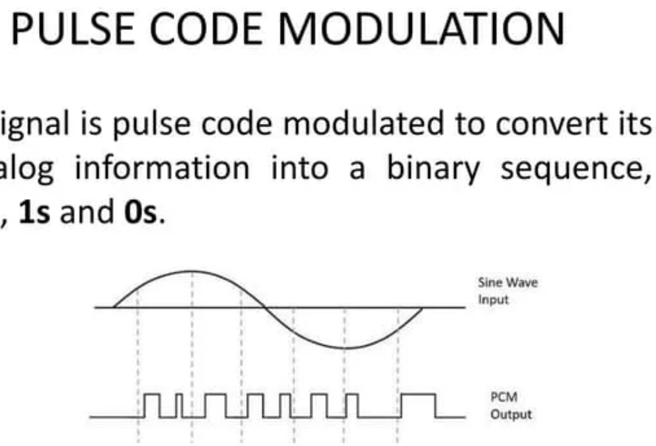

Figure 1: Sygnał sinusoidalny na wejściu i jego reprezentacja w PCM na wyjściu.

1.1 Format danych dla kodeka WM8994

Na płytce STM32F746G-Discovery kodek WM8994 komunikuje sie z mikrokontrolerem przez interfejs SAI (Serial Audio Interface) w standardzie I2S. Najważniejsze cechy formatu danych:

- **Rozdzielczość**: 16 bitów na próbke (format PCM liniowy, int16).
- **Kanały**: Stereo (2 kanały: lewy i prawy).
- **Czestotliwość próbkowania**: Standardowo 48 kHz, co pozwala na odtworzenie sygnałów do 24 kHz.
- **Skalowanie**: Wartości próbek sa skalowane do zakresu [-32768, 32767], gdzie 0 odpowiada ciszy, a wartości dodatnie i ujemne reprezentuja amplitude sygnału.

2 Generacja sygnału sinusoidalnego

Sygnał sinusoidalny o czestotliwości f (np. 1 kHz) jest generowany cyfrowo w próbkach, które nastepnie sa przesyłane do kodeka WM8994. Poniżej opisano trzy metody obliczania wartości sinusa: tablice stanów, szereg Taylora oraz algorytm CORDIC.

2.1 Tablica stanów (Lookup Table)

Aby wygenerować sygnał o określonej czestotliwości f_{out} przy użyciu tablicy LUT (Look-Up Table), należy kontrolować predkość, z jaka kolejne próbki sa odczytywane i wysyłane na wyjście (np. DAC lub PWM). Zakładajac stała czestotliwość próbkowania kodeka $f_s = 48\,kHz$, liczba próbek na cykl (długość tablicy LUT) N określa, jak płynnie sygnał jest aproksymowany.

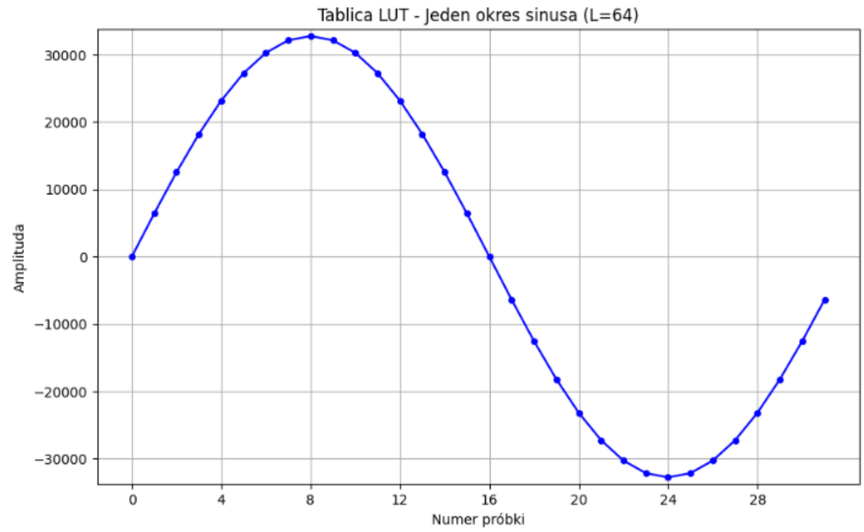

Figure 2: Tablica LUT sinus L = 64

Min Max = (-32767, 32767) => 16-bit signed Załóżmy, że tablica LUT zawiera N próbek jednej pełnej fali (np. sinusoidy). Czestotliwość wyjściowa f_{out} jest zwiazana z czestotliwościa próbkowania i indeksem odczytu:

$$f_{out} = \frac{f_s}{N}$$

Zatem, aby uzyskać sygnał o czestotliwości $f_{out} = 1\,kHz$ przy $f_s = 48\,kHz$, liczba próbek N powinna być:

$$N = \frac{f_s}{f_{out}} = \frac{48000}{1000} = 48$$

2.1 Przykład

- Czestotliwość próbkowania: $f_s = 48\,kHz$
- Żadana czestotliwość sygnału: $f_{out} = 1\,kHz$
- Długość tablicy LUT: $N = 48$

Obliczamy:

$$f_{out} = \frac{48000}{48} = 1000\,Hz$$

2.2 Szereg Taylora

Szereg Taylora aproksymuje funkcje sinus za pomoca sumy nieskończonej:

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \dots$$

W praktyce używa sie ograniczonej liczby wyrazów (np. 8), co zapewnia wystarczajaca precyzje dla małych katów x. Kat jest normalizowany do zakresu $[-\pi, \pi]$ przez operacje modulo:

$$x = \theta 2\pi$$

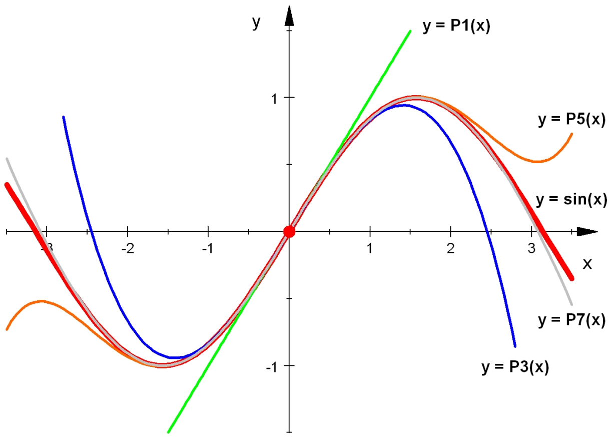

Figure 3: Szereg Taylora

Zalety:

- Nie wymaga pamieci na tablice.
- Możliwość dostosowania precyzji przez liczbe wyrazów.

Wady:

- Wysoka złożoność obliczeniowa (wiele operacji mnożenia i dzielenia).
- Wolniejsze od tablicy stanów.

2.3 Pseudokod algorytmu

Poniżej znajduje sie pseudokod algorytmu wyznaczajacego kolejne wartości szeregu Taylora.

Algorithm 1 Taylor Series Algorytm postepowania

{Function to compute sine using Taylor series approximation}
Input: angle (float)
Output: sine_value (int16_t)
$x \leftarrow fmod(angle, 2.0 \cdot \pi)$
if $x > \pi$ **then**
  $x \leftarrow x - 2.0 \cdot \pi$
end if
$result \leftarrow x$
$term \leftarrow x$
$x2 \leftarrow x \cdot x$
for $i = 1$ to 7 **do**
  $term \leftarrow term \cdot (-x2)/((2 \cdot i) \cdot (2 \cdot i + 1))$
  $result \leftarrow result + term$
end for
return $round(32767.0 \cdot result)$ as int16_t

2.4 Algorytm CORDIC

Algorytm CORDIC (Coordinate Rotation Digital Computer) jest metoda obliczeniowa służaca do obliczania funkcji trygonometrycznych, takich jak sinus, poprzez iteracyjne obracanie wektora w układzie współrzednych. Na podstawie załaczonego schematu (Rysunek 1) można zrozumieć jego działanie krok po kroku.

Na schemacie widzimy układ współrzednych z wektorem poczatkowym M_{in} o współrzednych (x_{in}, y_{in}), który reprezentuje punkt startowy. Kat poczatkowy β określa orientacje wektora wzgledem osi x. Celem algorytmu jest obrócenie tego wektora o kat θ, aby uzyskać nowy wektor (x_R, y_R), gdzie y_R aproksymuje wartość sinusa kata θ po odpowiednim skalowaniu.

Zasada działania Algorytm CORDIC wykorzystuje iteracyjne obroty wektora o predefiniowane katy α_i, które maleja z każda iteracja (np. $\alpha_0 = \arctan(1)$, $\alpha_1 = \arctan(1/2)$, itd.). Proces opiera sie na nastepujacych krokach:

1. **Inicjalizacja**: Rozpoczynamy od wektora $(x_0, y_0) = (K, 0)$, gdzie $K \approx 0.607$ jest stała skalujaca, która kompensuje zmniejszanie długości

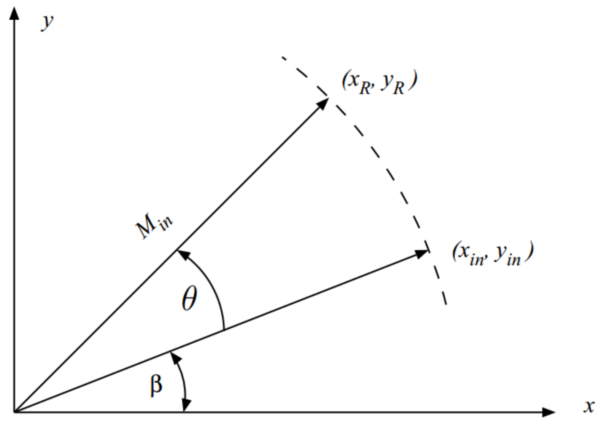

Figure 4: Schemat ilustrujacy działanie algorytmu CORDIC.

wektora podczas obrotów. Kat docelowy θ jest poczatkowo równy katowi wejściowemu.

2. **Obrót**: W każdej iteracji i wektor jest obracany o kat α_i. Kierunek obrotu zależy od znaku θ: - Jeśli $\theta \geq 0$, obrót jest zgodny z ruchem wskazówek zegara ($\sigma_i = +1$). - Jeśli $\theta < 0$, obrót jest przeciwny do ruchu wskazówek zegara ($\sigma_i = -1$). Nowe współrzedne sa obliczane według wzorów:

$$x_{i+1} = x_i - \sigma_i \cdot y_i \cdot 2^{-i}$$

$$y_{i+1} = y_i + \sigma_i \cdot x_i \cdot 2^{-i}$$

Kat θ jest aktualizowany: $\theta_{i+1} = \theta_i - \sigma_i \cdot \alpha_i$.

3. **Konwergencja**: Po n iteracjach (np. 16), gdy θ zbliża sie do zera, współrzedna y_n aproksymuje $\sin(\theta)$ pomnożona przez stała skalujaca K. Wartość y_n jest nastepnie skalowana przez $1/K$, aby uzyskać dokładny sinus.

Zalety i wady Zalety:

- Niskie zużycie zasobów obliczeniowych (tylko dodawanie, odejmowanie i przesuniecia bitowe).
- Dobra precyzja przy odpowiedniej liczbie iteracji.

Wady:

- Wolniejszy od tablicy stanów, ale szybszy od szeregu Taylora.
- Wymaga tablicy predefiniowanych katów α_i.

Algorithm 2 CORDIC Algorithm Pseudokod

{Funkcja obliczania sinusa za pomoca algorytmu CORDIC}
Input: theta (float) - kat w radianach
Output: sine_value (int16_t) - wartość sinusa w zakresie [-32767, 32767]
{Normalizacja kata do zakresu [0, 2)}
while $theta \geq 2.0 \cdot \pi$ **do** $theta \leftarrow theta - 2.0 \cdot \pi$
while $theta < 0.0$ **do** $theta \leftarrow theta + 2.0 \cdot \pi$
{Określenie kata i znaków dla odpowiedniej ćwiartki}
$angle \leftarrow 0.0$
$sign\_sin \leftarrow 1$
$sign\_cos \leftarrow 1$
if $0.0 \leq theta < \pi/2.0$ **then**
    $angle \leftarrow theta$
    $sign\_sin \leftarrow 1$
    $sign\_cos \leftarrow 1$
else if $\pi/2.0 \leq theta < \pi$ **then**
    $angle \leftarrow \pi - theta$
    $sign\_sin \leftarrow 1$
    $sign\_cos \leftarrow -1$
else if $\pi \leq theta < 3.0 \cdot \pi/2.0$ **then**
    $angle \leftarrow theta - \pi$
    $sign\_sin \leftarrow -1$
    $sign\_cos \leftarrow -1$
else
    $angle \leftarrow 2.0 \cdot \pi - theta$
    $sign\_sin \leftarrow -1$
    $sign\_cos \leftarrow 1$
end if
{Inicjalizacja zmiennych CORDIC}
$x \leftarrow cordic\_gain$
$y \leftarrow 0.0$
$z \leftarrow angle$
{Iteracje CORDIC}
for $i = 0$ to $CORDIC\_ITERATIONS - 1$ **do**
    $dx \leftarrow x \cdot (1.0/(1 \ll i))$
    $dy \leftarrow y \cdot (1.0/(1 \ll i))$
    if $z \geq 0.0$ **then**
        $x \leftarrow x - dy$
        $y \leftarrow y + dx$
        $z \leftarrow z - cordic\_angles[i]$
    else
        $x \leftarrow x + dy$
        $y \leftarrow y - dx$
        $z \leftarrow z + cordic\_angles[i]$
    end if
end for
{Zwróć przeskalowana wartość sinusa}
return $round(sign\_sin \cdot y \cdot 32767.0)$ as int16_t

3 DO ZAPAMIETANIA

- Wyznaczanie sinusa o określonej czestotliwości dla LUT oraz CMSISowych funkcji
- Zasada działana CORDIC oraz Szeregu Taylora (algorytm do napsiania)

4 Zadania Labolatoryjne

Wygeneruj sinusoide o czestotliwości 500Hz oraz 3kHz, $Fs = F_{CPU} = 200$ Mhz, jeśli chcesz wysłać bufor wyjściowy na wyjście CODECA wm8994, Fs = F_sampling codeca czyli np fs = 48kHz.

1. Przy użyciu tablic LUT, przy czestotliwości taktowania procesora aktualnej $F_{CPU} = 200$ Mhz
2. Przy użyciu funkcji CMSIS DSP arm_sin_f32(float32_t x) oraz arm_sin_q15(q15_t x), czym jest notacja q15, jak przekształcać liczby na q15? Dlaczego Q15 (patrz zeszłe lab)
3. Przy użyciu algorytmu CORDIC (napisz własna funkcje) oraz sprawdź poprawność jej działania.
4. Dodaj sygnal 1Khz z sygnalem 500Hz oraz 3kHz (przygotowanie do next lab)

5 Przydatne Linki

Dobre wytlumaczenie algorytmu CORDIC:
https://www.youtube.com/watch?v=bre7MVlxq7o

Tylor Series
https://www.youtube.com/watch?v=FF3U50yXdHU