Katedra Nanometrologii

LABORATORIUM:
LABORATORIUM PRZETWORNIKÓW A/C I C/A

3. PRZETWORNIKI C/A

IAD010402L

wzn.pwr.edu.pl/materialy-dydaktyczne

1. CEL ĆWICZENIA

Celem ćwiczenia jest poznanie funkcjonalności układów przetworników cyfrowo-analogowych (C/A) oraz obsługa tych urządzeń we współpracy z systemem mikrokontrolerowym Raspberry Pi Pico.

Zagadnienia do przygotowania:

- Rodzaje i budowa przetworników C/A
- Parametry przetwoników C/A, m. in. rozdzielczość, rodzaje błędów przetworników, czas ustalania sygnału
- Filtry rekonstrukcyjne – cel stosowania, parametry i realizacje układowe.

2. OPIS MAKIETY DYDAKTYCZNEJ

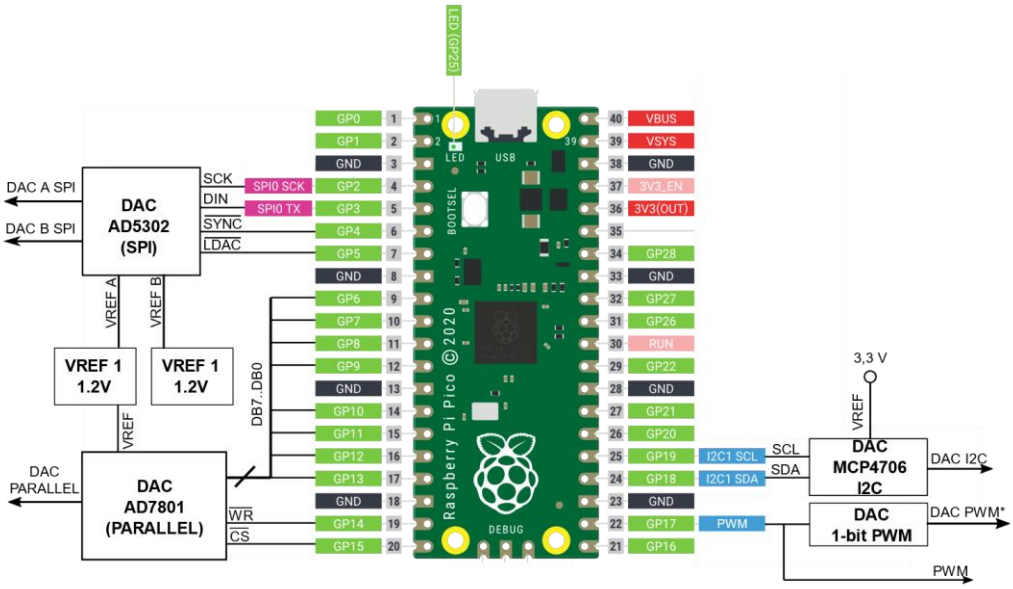
Makieta laboratoryjna składa się z nastepujących, zasadniczych elementów:

- Układów zasilania
- Systemu mikrokontrolerowego Raspberry Pi Pico
- Układów przetworników C/A
- Układów kondycjonowania sygnałów wyjściowych

Układy zasilania makiety wymagają podłączenia dwóch sekcji: napięć symetrycznych $\pm 15V$ oraz napięcia asymetrycznego $5V$. Napięciem symetrycznym, po przetworzeniu stabilizatorami liniowymi do poziomu napięć $\pm 5V$, są zasilane układy wzmacniaczy operacyjnych, służące jako wtórniki napięciowe, filtr sygnałowy i układy kondycjonowania sygnałów. Napięcie wejściowe $5V$ jest przetworzone do napięcia $3{,}3V$ i służy do zasilania układów przetworników cyfrowo-analogowych oraz mikroprocesorowego układu Raspberry Pi Pico.

Ogólny schemat podłączeń układów przetworników z systemem mikroprocesorowym na makiecie dydaktycznej przedstawia rys. 1. Makieta zawiera trzy układy scalone przetworników C/A: AD7801, AD5302 oraz MCP4706. Każdy z nich wyposażony jest w inny system komunikacji, zostały one podłączone również do różnych źródeł napięcia odniesienia. W kolejnych podrozdziałach zostaną przedstawione układy przetworników oraz sposoby sterowania nimi, w celu uzyskiwania pożądanych wartości sygnałów wyjściowych. Ponadto na makiecie znajduje się przetwornik 1-bitowy, zbudowany w oparciu o sygnał modulowany współczynnikiem wypełnienia impulsu (PWM – ang. *Pulse Width Modulation*).

Każdy z sygnałów wyjściowy przetworników na makiecie dydaktycznej jest podawany na układ wtórnika napięciowego, zbudowanego z wykorzsytaniem wzmacniacza operacyjnego. Sygnały wyjściowe przetworników, zaprezentowane na rys. 1, odpowiadają opisowi złącz koncentrycznych znajdujących się na makiecie dydaktycznej. Do tych złącz każdorazowo należy podłączać przewody sygnałowe w celu obserwacji przebiegów sygnałów wyjściowych na oscyloskopach lub mierzenia wartości napięć na woltomierzach lub multimetrach.



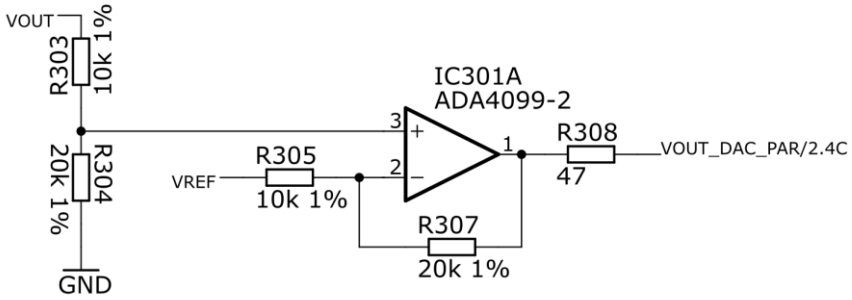
Rysunek 1. Schemat ogólny układów na makiecie dydaktycznej

3. OPIS UKŁADÓW PRZETWORNIKÓW CYFROWO-ANALOGOWYCH

3.1. PRZETWORNIK AD7801 Z RÓWNOLEGŁĄ SZYNĄ DANYCH

3.1.1. Opis układu AD7801

Przetwornik C/A AD7801 jest przetwornikiem 8-bitowym z równoległą szyną sterującą. Do przetwornika doprowadzony jest sygnał odniesienia o wartości napięcia $V_{ref} = 1{,}2V$. Ponadto, sygnał wyjściowy poddany jest przetworzeniu przez układ przesunięcia napięcia, umożliwiający uzyskanie bipolarnych sygnałów wyjściowych (wyjście DAC PARALLEL) w zakresie $\pm 5V$.



Rysunek 2. Realizacja układu wytwarzania napięcia bipolarnego

Wynik konwersji cyfrowo-analgowej w docelowym układzie, zgodnie z notą katalogową układu [1], można zapisać następującym równaniem (1):

$$V_O = R_{304}\frac{\left(1+\frac{R_{307}}{R_{305}}\right)}{R_{303}+R_{304}}\cdot\frac{2V_{ref}D}{256} - V_{ref}\frac{R_{307}}{R_{305}} \qquad (1)$$

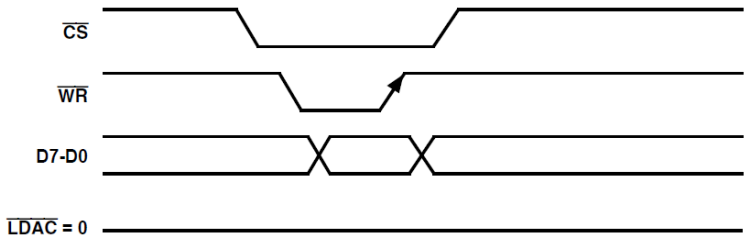
Dla elementów zaznaczonych na rys. 2, powyższe równanie przyjmuje następującą postać:

$$V_O = \left(\frac{4{,}8 \cdot D}{256} - 2{,}4\right) V \tag{2}$$

gdzie D oznacza wartość słowa bitowego.

3.1.2. Komunikacja z układem AD7801

W makiecie laboratoryjnej skonfigurowano układ przetwornika AD7801 w trybie automatycznej aktualizacji [1]. Tryb ten polega na tym, że niektóre sygnały wejściowe układu przetwornika mają z góry ustalone stałe wartości, a użytkownik ma za zadanie dokonywać zmian pozostałych sygnałów wejściowych w odpowiednim reżimie czasowym. Do wyboru układu oraz synchronizacji danych służą dwa sygnały, aktywne w stanie niskim: $\overline{WR}$ (z ang. *write*) i $\overline{CS}$ (z and. *chip select*). Poniższy rysunek przedstawia schemat wystawiania danych cyfrowych oraz sygnałów sterujących w trybie automatycznej aktualizacji. Przetwornik rozpocznie oczekiwanie na dane wejściowe dopiero po wybraniu układu sygnałem $\overline{CS}$, zaś szyna danych zostanie odczytana podczas narastającego zbocza sygnału $\overline{WR}$.



Rysunek 3. Przebiegi sygnałów sterujących i słowa danych układu AD7801

W najprostszym ujęciu, sygnały sterujące oraz szynę danych można zmieniać w środowisku interpretera języka MicroPython za pomocą sekwencji komend wykonywanych w nieskończonej pętli programu głównego. Wówczas można uzyskiwać napięcie stałe, lub, sygnały zmienne, cyklicznie modyfikując wartości bitowe słowa danych (np. w pętli). Podstawowe komendy, pozwalające na ustawienie pożądanej wartości bitowej, wymagają wybrania portów ogólnego przeznaczenia jako wyjść sygnałów sterujących i danych, oraz nadawania im pożądanych wartości.

W pierwszej kolejności należy zaimportować klasę `Pin` z modułu `machine`:

```
# import podstawowych klas z modułu machine
from machine import Pin
```

Komendy konfiguracyjne pinów mikrokontrolera Raspberry Pi Pico mają następującą postać:

```
# konfigurajca wyprowadzeń równoległych i konfiguracyjnych
dac_b0 = Pin(nr_pinu_b0, Pin.OUT)
dac_wr = Pin(nr_pinu_wr, Pin.OUT)
```

przy czym wartości (numery) `nr_pinu_b0`, `nr_pinu_wr` należy dobrać tak, by odpowiadały właściwym wyprowadzeniom układu Raspberry Pi Pico.

Sterowanie sygnałami wyjściowymi może się odbywać za pomocą następujących komend:

```
dac_b0.value(0)     alternatywnie     dac_b0.off()
dac_b1.value(1)     alternatywnie     dac_b1.on()
dac_wr.value(1)     alternatywnie     dac_wr.on()
```

Efektywniejszym sposobem modyfikacji słów sterujących i słowa danych jest wykorzystanie tzw. maszyny stanów portów wyjściowych PIO (ang. *programmed input/output*), będącą kontrolerem układów GPIO, działającym niezależnie od rdzenia mikrokontrolera. Poniżej przedstawiono przykładowy zapis konifguracji maszyny stanów PIO do wystawiania zawartości bufora wewnętrznego FIFO na porty wyjściowe.

```
import rp2
from machine import Pin

# maszyna stanów PIO zapisująca 1 słowo do przetwornika równoległego
# po otrzymaniu danej do swojego bufora FIFO z głównego programu
@rp2.asm_pio(
    #ustawienie pinow, kolejno: CS, WR
    set_init=(rp2.PIO.OUT_HIGH, rp2.PIO.OUT_HIGH),
    #ustawienie wyjść danych. shift right, w celu wyjustowania bitów
    out_init=(rp2.PIO.OUT_LOW,)*8, out_shiftdir=rp2.PIO.SHIFT_RIGHT
    )
def smproc():
    #poczatek petli wrap
    wrap_target()
    #oczekiwanie az do FIFO wejsciowego maszyny stanow zostanie przekazana wartosc z
programu glownego
    pull()
    # wystawienie zawartosci FIFO na wyjscia SM
    out(pins,8)
    # wyzerowanie pinow sterujacych CS i WR, oczekiwanie [5] cykli zegara
    set(pins,0b00) [5]

    # ustawienie pinow sterujacych CS i WR
    set(pins,0b11)
    #koniec programu i zapetlenie do wrap_target()
    wrap()
```

Dyrektywa `@rp2.asm_pio` zapewnia odpowiada za konfigurację odpowiednich wyprowadzeń: dla linii sterujących $\overline{WR}$ i $\overline{CS}$ oraz dla linii słowa bitowego. Procedura `smproc()` zapewnia odpowiednie sterowanie liniami sterującymi przetwornika $\overline{WR}$ i $\overline{CS}$. Ponieważ słowo bitowe powinno zostać przesłane do bufora FIFO maszyny stanów, w samej procedurze `smproc()` nie występuje jego wartość. Wartość tę należy podać, wywołując komendę przesłania słowa bitowego do rejestru FIFO z programu głównego. Wraz z wywołaniem procedury konfiguracji

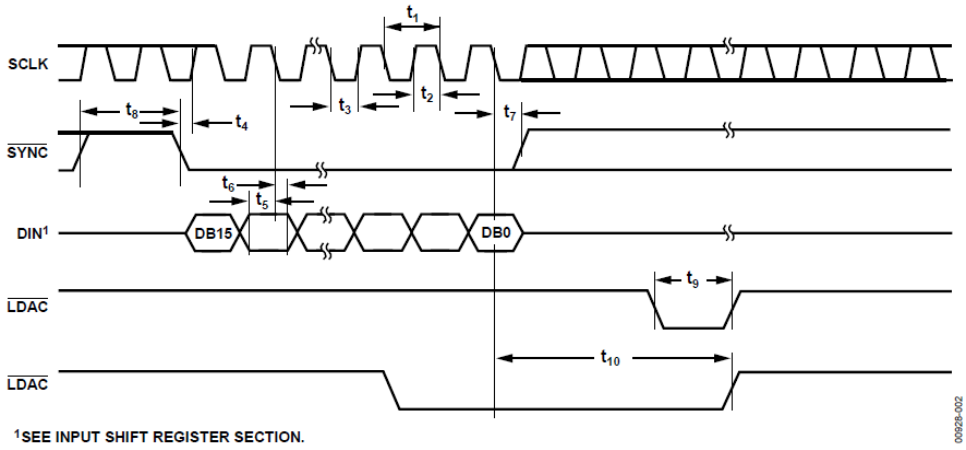
maszyny stanów dla pożądanych portów wejścia/wyjścia, przykładowy kod w języku MicroPython ma następującą postać

```
#inicjalizacja maszyny stanow 0, realizujace program w smproc, taktowanej 1 MHz,
#ktorej poczatkowym pinem do set jest Pin(14) a poczatkowym do out jest Pin(6)
sm = rp2.StateMachine(0, smproc, freq=1000000, set_base = Pin(14), out_base=Pin(6))
#aktywacja maszyny stanow
sm.active(1)
while True:
    for i in range(0, 512):
        sm.put(i)
```

W powyższym kodzie `sm` oznacza obiekt klasy rp2.StateMachine, dla którego zdefiniowano następujące parametry: maszynę stanów `smproc`, częstotliwość pracy `1MHz` oraz dwa piny początkowe dla grupy wyprowadzeń: pinów konfiguracyjnych oraz pinów słowa danych. W następnym kroku, za pomocą `sm.active(1)` odbywa się aktywacja maszyny stanów, zaś w pętli nieskończonej programu następuje wysyłanie do bufora FIFO maszyny stanów wartości `i`.

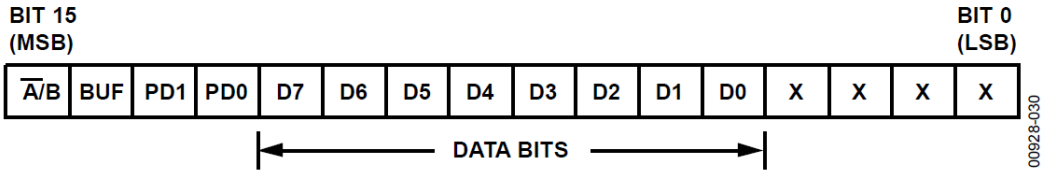
3.2. DWUKANAŁOWY PRZETWORNIK AD5302 Z KOMUNIKACJĄ SPI

Przetwornik AD5302 posiada dwa kanały przetwarzania słowa cyfrowego na sygnał analogowy. Wersja wprzetwornika wykorzystywana w niniejszym ćwiczeniu pozwala na konwersję 8-bitową. Źródłem napięcia odniesienia kanału A przetwornika jest źródło wzorcowe o napięciu 1,2*V*, zaś dla kanału B – napięcie o wartości 1,2*V*, uzyskane z dzielnika napięcia zasilania 3,3*V*, podanego na wejście wtórnika napięciowego. Komunikacja SPI pozwala na wysterowanie jednego albo drugiego kanału przetwornika podczas przesłania 16-bitowego słowa cyfrowego. Sposób komunikacji z przetwornikiem przedstawia rys. 5 [2].



Rysunek 4. Sygnały transmisji SPI układu AD5302

Sygnał $\overline{\text{SYNC}}$ odpowiada sygnałowi $\overline{\text{CS}}$ transmisji SPI. Ponadto, $\overline{\text{LDAC}}$ stanowi sygnał synchronizacji, powodującym przesłanie zawartości rejestrów wejściowych do rejestrów wyjściowych przetwornika i na wyjście układu.



Rysunek 5. Słowo transmisji SPI układu AD5302

Zgodnie z rysunkiem 5, za pomocą jednego cyklu transmisji, można wysłać słowo sterujące przetwornika o wartości D7..D0, dla kanału A lub B w zależności od stanu pierwszego, najstarszego bitu transmisji. Wysterowanie obu kanałów przetwornika jest możliwe poprzez wysłanie dwóch słów 16-bitowych, osobno dla kanału A i B, przy zachowaniu niskiego stanu $\overline{\text{LDAC}}$, a następnie zsynchronizować wyjście analogowe poprzez zmianę sygnału $\overline{\text{LDAC}}$ na wysoki. Pola PD1 i PD0 bitów transmisji decydują o trybie poboru mocy, natomiast pole BUF konfiguruje buforowanie napięcia odniesienia. Dla potrzeb niniejszego ćwiczenia, trzy wspomniane pola powinny mieć wartość 0.

Zainicjowanie komunikacji z przetwornikiem może nastąpić poprzez wywołanie następujących sekwencji komend w języku MicroPython.

```
# import podstawowych klas z modułu machine
from machine import Pin, SPI
# konfiguracja wyprowadzeń SPI
spi = SPI(SPI_n, baudrate=BAUDRATE, mosi = Pin(pin_mosi), sck = Pin(pin_sck))
```

W powyższym opisie należy zastąpić wyrażenia w kolorze czerwonym odpowiednimi wartościami (instancja/numer modułu SPI, częstotliwość taktowania sygnału zegarowego, numery wyprowadzeń).

Ze względu na 16-bitowe słowo transmisji (dane+konfiguracja), wysyłane słowo powinno zostać sformułowane w postaci bitowej, ponieważ metoda `write(buf)` klasy `spi`, przyjmuje jako argument dane typu buforowego. W związku z tym, można utworzyć argument typu buforowego za pomocą następującej konstrukcji:

```
# inicjalizacja bufora
buffer=bytearray(2)
# przypisanie bajtów danych do bufora
buffer[0] = liczba_MSB
buffer[1] = liczba_LSB
```

Wartość napięcia wyjściowego omawianego przetwornika jest dana następującą zależnością [2]:

$$V_{OUT} = \frac{V_{ref} D}{2^N} \tag{3}$$

Gdzie N oznacza rozdzielczość bitową przetwornika (w typ przypadku 8), natomiast D – dziesiętną reprezentację słowa danych w zakresie 0-255.

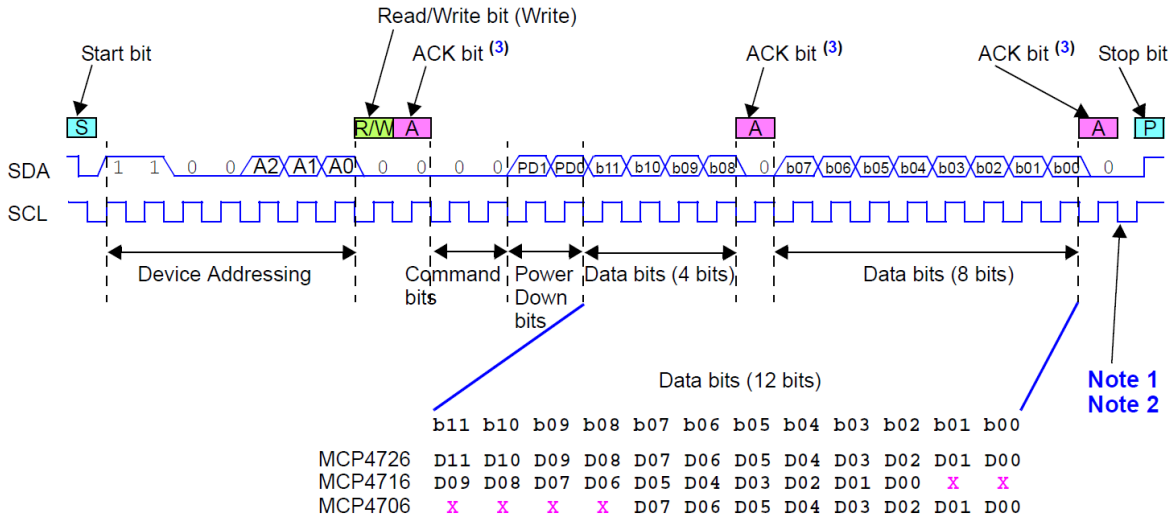
3.3. PRZETWORNIK MCP4706 Z KOMUNIKACJĄ I2C

Przetwornik MCP4706 jest przetwornikiem jednokanałowym, z magistralą I²C. Przetwornik zawiera wewnętrzne rejestry konfiguracyjne oraz sterujące konwersją, a także wewnętrzną nieulotną pamięć EEPROM, pozwalającą na przechowywanie wartości sygnału wyjściowego po odłączeniu zasilania. Przetwornik w układzie makiety wykorzystuje napięcie zasilające jako napięcie odniesienia. Wynik konwersji cyfrowo-analogowej jest dany następującą zależnością [3]:

$$V_{OUT} = \frac{V_{ref} D}{2^N} \cdot G \tag{4}$$

Gdzie N oznacza rozdzielczość bitową przetwornika (w typ przypadku 8), natomiast D – dziesiętną reprezentację słowa danych w zakresie 0-255. Litera G oznacza wzmocnienie, które jest równe jedności.

Za pomocą transmisji magistralą I²C można w rozmaity sposób konfigurować pracę układu, w tym zmieniać źródło napięcia odniesienia, wzmocnienie, przełączać układ w tryb niskiego poboru mocy, oraz konfigurować pamięć nieulotną EEPROM. Przy ustawieniach fabrycznych, w podstawowej konfiguracji, wzmocnienie przetwornika G będzie wynosiło 1, zaś źródłem napięcia odniesienia będzie napięcie zasilające. Z pozostałymi trybami pracy układu można zapoznać się, dokumentacją techniczną [3]. W podstawowej konfiguracji, do natychmiastowego wysterowania wyjścia analogowego, można się posłużyć ramką danych, zaprezentowaną na 6. W takim układzie,



Rysunek 6. Przykład ramki danych z konfiguracją rejestru wyjściowego

W tym przypadku należy zaadresować układ przetwornika (zgodnie z konwencją magistrali I²C), a następnie przesłać dane konfiguracyjne wraz ze słowem danych.

Konfiguracja układu I²C odbywa się za pomocą języka MicroPython następującymi komendami:

```
# import podstawowych klas z modułu machine
from machine import Pin, I2C
```

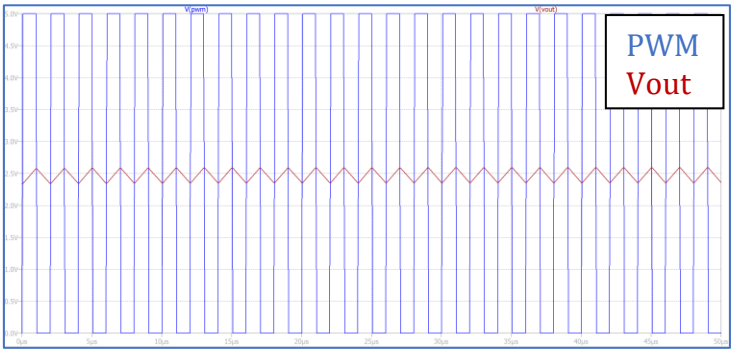
```
# definicja wyprowadzeń I2C DAC
i2c = I2C(I2C_n,scl=Pin(pins_cl), sda=Pin(pin_sda), freq=100000)

Wysłanie do urządzenia danych odbywa się w następujący sposób.

# wysłanie do urządzenia o 7-bitowym adresie addr bajtów zmiennej buffer
i2c.writeto(addr,buffer)
```

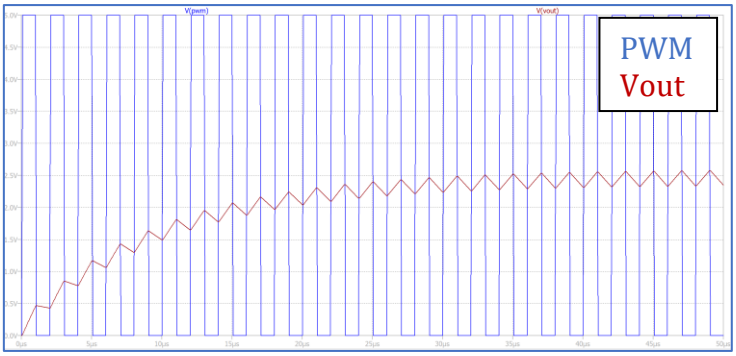
3.4. PRZETWORNIK 1-BITOWY PWM

W makiecie dydaktycznej zamieszczono układ przetwornika, opartego o generację sygnału modulacji współczynnika wypełnienia impulsu (ang. *Pulse Width Modulation*) a następnie jego filtrację filtrem dolnoprzepustowym. Wynik działania takiego układu zaprezentowany jest na rys. 7:



Rysunek 7. Filtracja sygnału PWM filtrem dolnoprzepustowym

W układzie makiety dydaktycznej sygnał PWM jest możliwy do uzyskania za pomocą systemu Raspberry Pi Pico. Wyjścia sygnału PWM są poprowadzone do trzech gniazd koncentrycznych: przez wtórnik napięcia, przez filtr dolnoprzepustowy RC, oraz przez filtr dolnoprzepustowy Sallena-Key'a 2. rzędu. Filtr RC posiada częstotliwość graniczną zbliżoną do 10 kHz, natomiast filtr aktywny (Sallena-Key'a) – ok. 16 kHz. Parametry filtrów dolnoprzepustowych decydują o czasie narostu sygnału wyjściowego i jednocześnie o częstotliwości granicznej generowanego sygnału. Czas narostu sygnału wyjściowego przedstawiono na rys. 8. W celu uzyskania obrazu oscyloskopowego czasu narostu, można cyklicznie włączać i wyłączać modulację PWM sygnału dla danej częstotliwości i współczynnika wypełnienia.



Rysunek 8. Czas narostu sygnału wyjściowego

Zmieniając cyklicznie szerokość generowanych impulsów PWM, można uzyskiwać sygnał wyjściowy o zmiennej amplitudzie. Częstotliwość modulacji będzie decydować o amplitudzie zafalowań, widocznych na rys. 7 i 8 w postaci przebiegu piłokształtnego. Napięcie wyjściowe przetwornika można w przybliżeniu określić zależnością:

$$V_{OUT} = k \cdot U_{zas} \tag{5}$$

Gdzie $k = \frac{\tau}{T}$ oznacza współczynnik wypełnienia impulsu (0-1).

W celu generowania impulsów PWM, można wykorzystać dowolne wyprowadzenie ogólnego przeznaczenia układu Raspberry Pi Pico. W tym celu należy skonfigurować wybrany port w do pracy w trybie PWM, a następnie modyfikować parametry generowanego sygnału. Przykładowy kod w języku MicroPython służący do konfiguracji wyprowadzenia PWM przedstawiono poniżej.

```
# import podstawowych klas z modułu machine
from machine import PWM

# Utworzenie obiektu pwm klasy PWM, z wyjściem na port GPIO 25
pwm = PWM(Pin(25))

# Ustawienie częstotliwości sygnału PWM
pwm.freq(1000)

# Ustawienie współczynnika wypełnienia impulsów PWM
pwm.duty_u16(6554)
```

Należy zwrócić uwagę na format danych argumentów, jakie przyjmują poszczególne metody klasy pwm.

4. OPIS ZADAŃ DO WYKONANIA

4.1. ZADANIE 1
OBSŁUGA PRZETWORNIKÓW C/A (1,5 PKT.)

Celem zadania jest nawiązanie komunikacji z każdym z przetworników dostępnych na makiecie dydaktycznej. Uwaga: wszystkie przetworniki mogą być obsługiwane jednocześnie.

4.1.1. Obsługa przetwornika C/A AD7801 z równoległą szyną danych.

W celu obsługi przetwornika z równoległą szyną danych, należy wykonać następujące czynności:

- skonfigurować wyprowadzenia układu Raspberry Pi Pico odpowiadające szynie danych przetwornika oraz sygnałów sterujących, zgodnie z rys. 1 i opisem zawartym w punkcie 3.1
- ustalić pożądaną wartość bitową na szynie danych, wskazaną przez prowadzącego.
- w odpowiedni sposób modyfikować sygnały sterujące, zgodnie z przebiegami zawartymi na rys. 3 oraz instrukcją – pkt. 3.1.2

Mierzyć za pomocą woltomierza lub multimetru wartości napięć wyjściowych przetwornika.

4.1.2. Obsługa przetwornika C/A AD5302 z magistralą SPI

W celu obsługi przetwornika z magistralą SPI, należy wykonać następujące czynności:

- Skonfigurować wyprowadzenia układu Raspberry Pi Pico odpowiadające komunikacji SPI, zgodnie z rys. 1 i opisem zawartym w punkcie 3.2
- Przygotować zmienną odpowiadającą pożądanym nastawom słowa konfiguracyjno-sterującego, wraz z 8-bitowym słowem danych, wskazanym przez prowadzącego
- Spowodować wysłanie danych przez port SPI układu Raspberry Pi Pico, zgodnie z opisem zawartym w punkcie 3.2.

Mierzyć za pomocą woltomierza lub multimetru wartości napięć wyjściowych przetwornika.

4.1.3. Obsługa przetwornika C/A MCP4706 z magistralą I^2C

W celu obsługi przetwornika z magistralą I^2C, należy wykonać następujące czynności:

- Skonfigurować wyprowadzenia układu Raspberry Pi Pico odpowiadające komunikacji I^2C, zgodnie z rys. 1 oraz opisem zawartym w punkcie 3.3
- Przygotować zmienną odpowiadającą pożądanym nastawom słowa konfiguracyjno-sterującego, wraz z 8-bitowym słowem danych, wskazanym przez prowadzącego
- Spowodować wysłanie danych przez port I^2C układu Raspberry Pi Pico, zgodnie z opisem zawartym w punkcie 3.3

Mierzyć za pomocą woltomierza lub multimetru wartości napięć wyjściowych przetwornika.

4.1.4. Obsługa przetwornika 1-bit PWM

W celu uzyskania konwersji cyfrowo-analogowej, należy wygenerować sygnał PWM na odpowiednim wyprowadzeniu układu mikrokontrolerowego Raspberry Pi Pico. Generacja sygnału PWM w układzie Raspberry jest dozwolona na wszystkich wyprowadzeniach ogólnego przeznaczenia. Sposób konfiguracji portu wejścia wyjścia w przedstawiono w punkcie 3.4. W ramach ćwiczenia należy określić amplitudę zafalowań oraz czas narostu sygnału, zgodnie z punktem 3.4.

4.2. ZADANIE 2
BADANIE CHARAKTERYSTYK UKŁADÓW PRZETWORNIKÓW C/A (2 PKT.)

Celem zadania jest określenie parametrów, charakteryzujących przetworniki C/A. W niniejszym ćwiczeniu przetworniki zostane poddane następującym badaniom: nieliniowość całkową (ang. *Integral Nonlinearity,* INL), nieliniowość różniczkową (ang. *Differential Nonlinearity*, DNL) oraz błąd przeniesienia kodu (ang. *major-carry transition*), powodujący tzw. *glitch* (z ang. *glitch* to usterka).

4.2.1. Nieliniowość całkowa

Nieliniowość całkową wyznacza się z następujących wzorów [5]:

$$INL[k] = 100\% \cdot \frac{\varepsilon[k]}{U_{FSR} \cdot G} \tag{6}$$

$$\varepsilon[k] = U[k] - G \cdot (Q \cdot k + U_0) \tag{7}$$

Gdzie:

- U_{FSR} oznacza wartość wyjściową pełnej skali, czyli dla przetworników o wyjściach napięciowych różnicę pomiędzy napięciem wyjściowym dla największej i najmniejszej liczby bitowej
- G – wzmocnienie
- $U[k]$ – sygnał wyjściowy dla liczby bitowej k
- Q – idealna szerokość sygnału dla jednego bitu, tj. $\frac{U_{FSR}}{2^N}$
- U_0 – idealna wartość sygnału dla słowa bitowego 0.

INL jest maksymalną wartością uzyskaną dla wszystkich k. W ćwiczeniu należy obliczać wartość INL dla każdego przetwornika, dla kilku (np. 5) przedziałów skali przetwarzania.

4.2.2. Nieliniowowść różniczkowa

Błąd ten opisywany jest jako różnica pomiędzy dwoma wartościami sygnału wyjściowego dla sąsiednich słów bitowych a idealną szerokością sygnału dla jednego bitu, podzieloną przez tę wartość.

$$DNL[k] = \frac{W[k] - Q}{Q} \tag{8}$$

Gdzie $W[k]$ oznacza różnicę sygnałów wyjściowych dla sąsiednich słów bitowych, skorygowane o wartość wzmocnienia: $W[k] = G \cdot (U[k+1] - U[k])$.

4.2.3. Błędy typu *glitch*

W celu zbadania podatności przetworników cyfrowo-analogowych na błędy typu glitch, należy naprzemiennie na wejścia przetworników C/A podawać słowa bitowe, w których następują największe różnice w liczbie kodowej, np. zmiana wartości najstarszego bitu kodu

oraz wszystkich młodszych na przeciwną. Zmianę sygnału należy wykonywać w sposób cykliczny tak, aby wynik można było obserwować na oscyloskopie.

4.3. ZADANIE 3
GENERACJA SYGNAŁÓW ZA POMOCĄ PRZETWORNIKÓW (1,5 PKT.)

Zadanie polega na wystawianiu słów kodowych na wejścia przetworników w taki sposób, by na wyjściu przetworników można było obserwować przebiegi okresowe.

4.3.1. Generacja sygnału prostokątnego

Generacja sygnału prostokątnego polega na naprzemiennym sterowaniu przetowrników C/A słowami bitowymi o wartościach odpowiadającym pożądanym wartościom, np. dwóm skrajnym wartościom skali.

4.3.2. Generacja sygnału trójkątnego

Sygnał trójkątny można uzyskać, wykorzystując jako słowo sterujące, podawane na przetwronik, licznik inkrementowany w pętli programu.

4.3.3. Generacja sygnału sinusoidalnego

Jedną z metod generacji sygnału sinusodialnego jest przygotowanie próbek sygnału sinusoidalnego w tablicy programowej, a następnie wystawianie kolejnych próbek na wejście przetwronika C/A. Można to zrealizować poprzez inkrementację indeksu tablicy próbek w pętlach programowych. Jest to tak zwana metoda tablicowa (LUT – ang. *Look-Up Table*). Bardziej elastyczną metodą generacji sygnału sinusoidalnego jest metoda DDS, będąca zmodyfikowaną wersją metody tablicowej. Jej opis wykracza jednak poza zakres ćwiczenia.

5. BIBLIOGRAFIA

[1] Nota katalogowa przetwornika AD7801 [Online]
https://www.analog.com/media/en/technical-documentation/data-sheets/AD7801.pdf

[2] Nota katalogowa przetwornika AD5302 [Online]

https://www.analog.com/media/en/technical-documentation/data-sheets/ad5302_5312_5322.pdf

[3] Nota katalogowa przetwornika MCP4706 [Online]:

https://ww1.microchip.com/downloads/en/DeviceDoc/22272C.pdf

[4] Texas Instruments (SPRAA88A) – “Using PWM Output as a Digital-to-Analog Converter ona TMS320F280x Digital Signal Controller”, 2008

[5] IEEE 1658-2011. Standard for Terminology and Test Methods of Digital-to-Analog Converter Devices, 2012

PROTOKÓŁy

CZŁONKOWIE GRUPY			
1.			
2.			
3.			
4.			
5.			
6.			
NR GRUPY	**NUMER ĆWICZENIA**	**DATA ZAJĘĆ**	**PODPIS PROWADZĄCEGO**