Projekt

Sterowniki robotów

Raport

Grająca płytka
GraP

Skład grupy (2):
Piotr Gogola, 226249
Maciej Szymczyk, 226282

Termin: wtTN17.05

Prowadzący:
mgr inż. Wojciech Domski

11.06.2018

Spis treści

1 Opis projektu **2**
1.1 Temat . . . 2
1.2 Opis zrealizowanych prac . . . 2

2 Użyte środowisko oraz biblioteki **2**

3 Konfiguracja mikrokontrolera **3**
3.1 Konfiguracja portów MCU . . . 3
3.2 Konfiguracja zegarów . . . 4
3.3 Konfiguracja peryferiów . . . 5
3.3.1 SPI2 . . . 5
3.3.2 I2C1 . . . 5
3.3.3 SAI1 . . . 5
3.3.4 DMA . . . 6
3.3.5 TIM16 . . . 6
3.3.6 USART2 . . . 6

4 Konfiguracja układów zewnętrznych **7**
4.1 Żyroskop L3GD20 . . . 7
4.2 Akcelerometr LSM303C . . . 7
4.3 Przetwornik Audio DAC . . . 8

5 Opis działania programu **10**

6 Zadania niezrealizowane **11**

1 Opis projektu

Kod źródłowy projektu

1.1 Temat

Tematem projektu jest grająca płytka (płytka rozwojowa STM32L476G-DISCO), czyli utworzenie oprogramowania wykorzystującego urządzenia peryferyjne takie jak żyroskop, akcelerometr i przetwornik Audio DAC dołączone do mikrokontrolera STM32L476VGT6 firmy STMicroelectronics.

Celem projektu jest napisanie oprogramowania na mikrokontroler STM32L476VGT6 pozwalającego na generowanie dźwięków różnych fali, np. sinusoidy, fali trójkątnej, prostokątnej itp., o zmiennej amplitudzie i częstotliwości. Rodzaj generowanej fali oraz jej parametry mają być sterowane ruchami oraz orientacją płytki deweloperskiej. Informacja ta będzie uzyskiwana dzięki odczytom z trzyosiowego żyroskopu L3GD20, oraz akcelerometru LSM303C.

Komunikacja mikrokontrolera STM32L476VGT6 z żyroskopem L3GD20 i akcelerometrem LSM303C zamontowanymi na płytce rozwojowej STM32L476G-DISCO odbywa się poprzez interfejs SPI. Żyroskop i akcelerometr współdzielą ten sam interfejs SPI mikrokokontrolera (te same piny mikrokontrolera), a wybór czujnika odbywa się za pomocą dwóch oddzielnych wyprowadzeń mikrokontrolera, tzw. chip-select. Żyroskop wykorzystuje SPI full-duplex, a akcelerometr half-duplex nazywany "3-wire SPI".

Komunikacja pomiędzy mikrokontrolerem, a przetwornikiem cyfrowo-analogowym odbywa się poprzez 2 interfejsy:

1. I^2C – komunikacja z rejestrami konfigurującymi działanie przetwornika
2. SAI – realizuje połączenie z przetwornikiem z wykorzystaniem interfejsu I^2S. Interfejs ten jest wykorzystywany do przesyłu danych w postaci cyfrowej do przetwornika.

1.2 Opis zrealizowanych prac

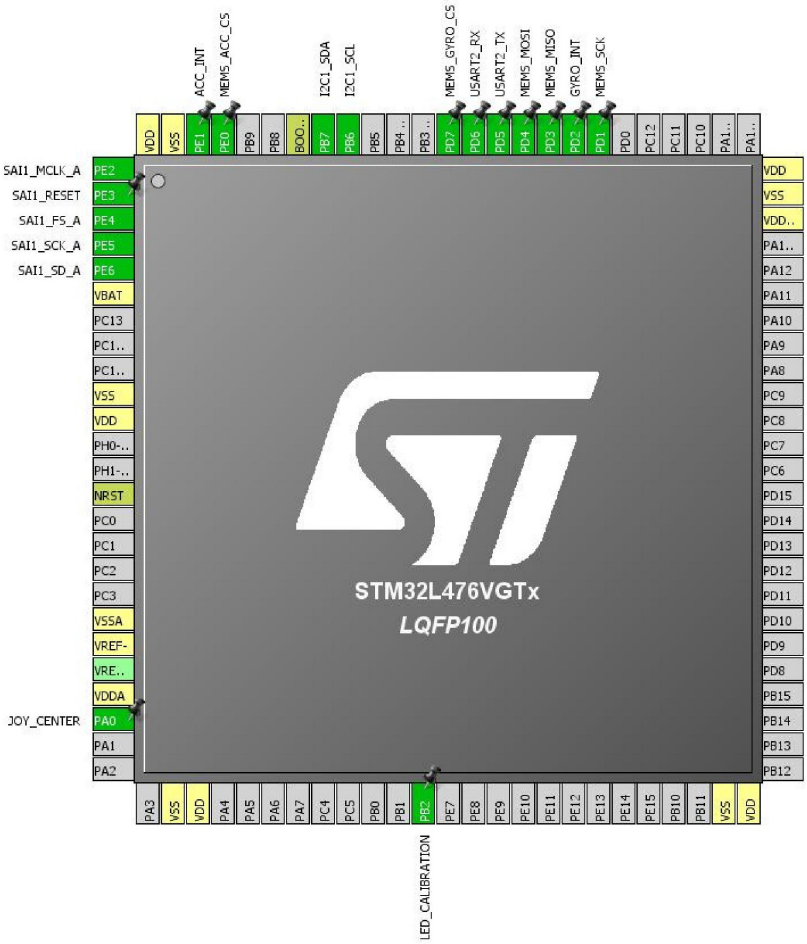
W projekcie udało się, zgodnie z założeniami, zaimplementować sterowniki do obsługi żyroskopu L32GD20, akcelerometru LSM303C, wykorzystując bibliotekę BSP skonfigurować i uruchomić przetwornik cyfrowo-analogowy Audio DAC CS43L22 oraz połączyć działanie wszystkich tych trzech układów zewnętrznych uzyskując zamierzony efekt jakim jest generowanie dźwięku sinusoidy, fali prostokątnej, piłokształtnej oraz trójkątnej o zmiennej amplitudzie i częstotliwości w zależności od orientacji płytki prototypowej.

2 Użyte środowisko oraz biblioteki

Projekt został napisany w języku C w środowisku Atollic TrueSTUDIO for STM32 w wersji 9.0.0. Konfiguracja peryferiów została wykonana w programie STM32CubeMX w wersji 4.25.0, który generuje kod konfigurujący dla zadanych peryferiów mikrokontrolera w bibliotece HAL. Do implementacji sterowników dla akcelerometru LSM303C oraz żyroskopu L3GD20 użyto biblioteki HAL, a w przypadku konfiguracji przetwornika cyfrowo-analogowego CS43L22 wykorzystano bibliotekę BSP dostarczoną przez firmę STMicroelectronics dla użytej płytki developerskiej STM32L476 Discovery.

3 Konfiguracja mikrokontrolera

3.1 Konfiguracja portów MCU

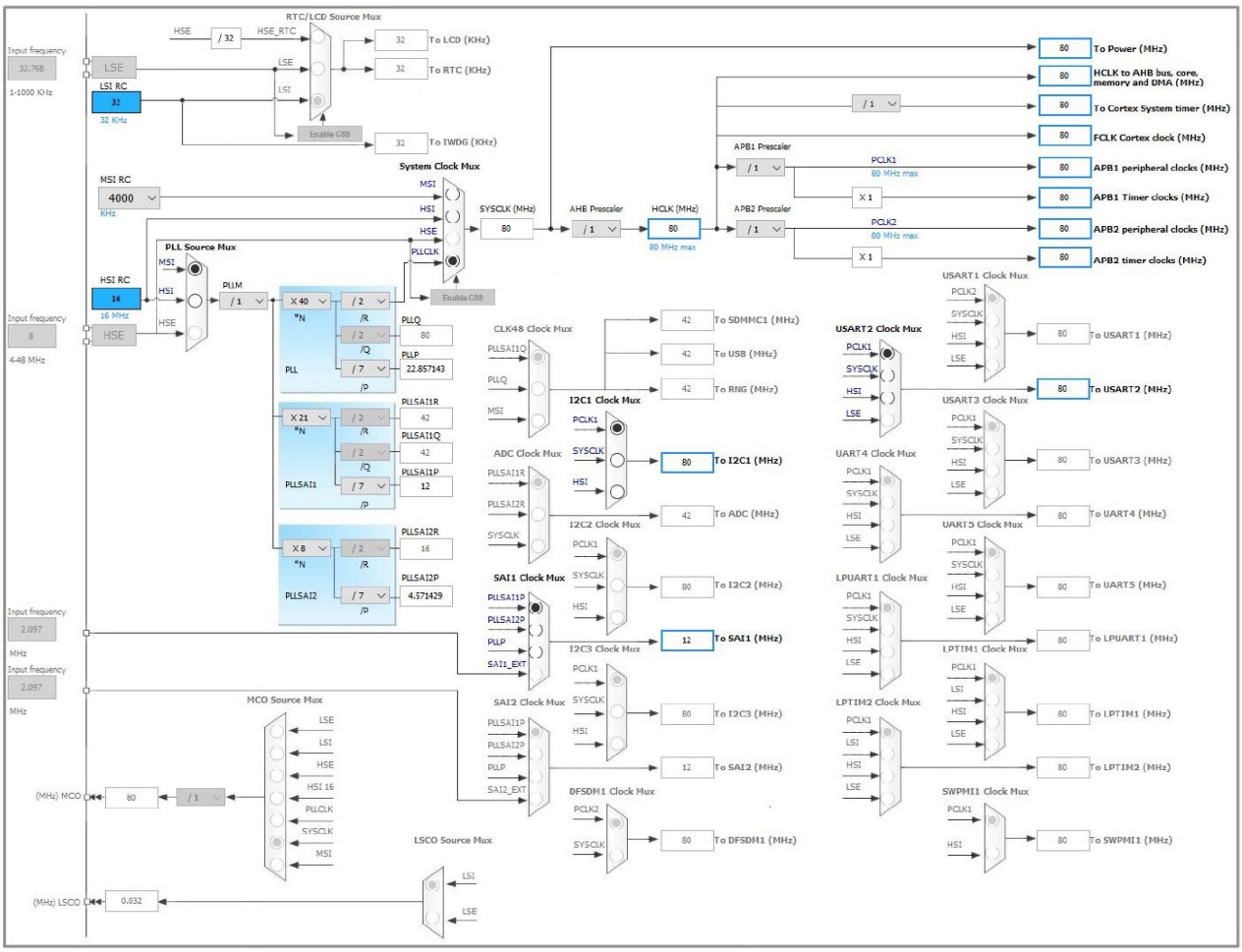


Rysunek 1: Konfiguracja pinów mikrokontrolera STM32L476VG w programie STM32CubeMX

Pin Nb	Pins	Functions	Labels
1	PE2	SAI1_MCLK_A	
2	PE3	GPIO_Output	SAI1_RESET
3	PE4	SAI1_FS_A	
4	PE5	SAI1_SCK_A	
5	PE6	SAI1_SD_A	
23	PA0	GPIO_Input	JOY_CENTER
37	PB2	GPIO_Output	LED_CALIBRATION
82	PD1	SPI2_SCK	MEMS_SCK
83	PD2	GPIO_EXTI2	GYRO_INT
84	PD3	SPI2_MISO	MEMS_MISO
85	PD4	SPI2_MOSI	MEMS_MOSI
86	PD5	USART2_TX	
87	PD6	USART2_RX	
88	PD7	GPIO_Output	MEMS_GYRO_CS
97	PE0	GPIO_Output	MEMS_ACC_CS
92	PB6	I2C1_SCL	
93	PB7	I2C1_SDA	
98	PE1	GPIO_EXTI1	ACC_INT

Peripherals	Modes	Functions	Pins
I2C1	I2C	`I2C1_SCL`	PB6
I2C1	I2C	`I2C1_SDA`	PB7
SAI1:SAI A	Master with Master Clock Out	`SAI1_SD_A`	PE6
SAI1:SAI A	Master with Master Clock Out	`SAI1_SCK_A`	PE5
SAI1:SAI A	Master with Master Clock Out	`SAI1_FS_A`	PE4
SAI1:SAI A	Master with Master Clock Out	`SAI1_MCLK_A`	PE2
SPI2	Full-Duplex Master	`SPI2_MISO`	PD3
SPI2	Full-Duplex Master	`SPI2_MOSI`	PD4
SPI2	Full-Duplex Master	`SPI2_SCK`	PD1
SYS	SysTick	`SYS_VS_Systick`	`VP_SYS_VS_Systick`
TIM16	`Enable_Timer`	`TIM16_VS_ClockSourceINT`	`VP_TIM16_VS_ClockSourceINT`
USART2	Asynchronous	`USART2_RX`	PD6
USART2	Asynchronous	`USART2_TX`	PD5

3.2 Konfiguracja zegarów



Rysunek 2: Konfiguracja zegarów mikrokontrolera STM32L476VG w programie STM32CubeMX

3.3 Konfiguracja peryferiów

3.3.1 SPI2

Basic Parameters	
Frame Format	Motorola
Data Size	8 Bits
First Bit	MSB First
Clock Parameters	
Prescaler (for Baud Rate)	8
Baud Rate	10.0 MBits/s
Clock Polarity (CPOL)	Low
Clock Phase (CPHA)	1 Edge

3.3.2 I2C1

Timing configuration	
I2C Speed Mode	Standard Mode
I2C Speed Frequency (KHz)	100
Slave Features	
Clock No Stretch Mode	Disabled
General Call Address Detection	Disabled
Primary Address Length selection	7-bit

3.3.3 SAI1

Basic Parameters	
Protocol	Free
Audio Mode	Master Transmit
Frame Length	32 bits
Data Size	16 Bits
Slot Size	16 Bits
Output Mode	Mono
SAI SD Line Output	Mode Driven
Parameters	
First Bit	MSB First
Frame Synchro Active Level Length	16
Frame Synchro Definition	Channel Identification
Frame Synchro Polarity Active	Low
Frame Synchro Offset	Before First Bit
Slots Parameters	
Number of Slots (only Even Values)	2
Slot Active Final Value	0x00000003
Slot Active Use	Slot 0 and 1 only
Clock Parameters	
Master Clock Divider	Enabled
Audio Frequency	48 KHz
Real Audio Frequency	46.875 KHz
Error between Selected	-2.34 %
Advanced Parameters	
Output Drive	Enabled
Synchronization External	Disabled

3.3.4 DMA

DMA request	SAI1_A
Stream	DMA2_Channel1
Direction	Memory To Peripheral
Priority	Medium
DMA2_Channel1 DMA request Settings	
Mode	Circular
Peripheral Increment	Disable
Memory Increment	Enable
Peripheral Data Width	Word
Memory Data Width	Half Word

3.3.5 TIM16

Counter Settings

Prescaler (PSC - 16 bits value)	`TIM16_PRESCALER`
Counter Mode	Up
Counter Period (AutoReload Register - 16 bits value)	`TIM16_PERIOD`
Internal Clock Division (CKD)	No Division
Repetition Counter (RCR - 8 bits value)	0
auto-reload preload	Enable

`TIM16_PRESCALER` = 19999
`TIM16_PERIOD` = 3999
Co przy taktowaniu z częstotliwością 80MHz powoduje przepełnienie licznika (i wygenerowanie przerwania) co 1 sekundę (`Update_event` = `TIM_CLK`/((PSC + 1)*(ARR + 1)*(RCR + 1))) [1].

3.3.6 USART2

W celach debugowania programu został skonfigurowany interfejs UART2.

Basic Parameters

Baud Rate	115200
Word Length	8 Bits (including Parity)
Parity	None
Stop Bits	1
Advanced Parameters	
Data Direction	Receive and Transmit

4 Konfiguracja układów zewnętrznych

4.1 Żyroskop L3GD20

Zaimplementowano sterownik do żyroskopu L3GD20 znajdującego się na płytce developerskiej STM32L476 Discovery. Konfiguracja interfejsu SPI wykorzystywana do komunikacji mikrokontrolera z czujnikiem została wygenerowana za pomocą programu CubeMX. Zgodnie z dokumentacją płytki deweloperskiej [2] został uruchomiony interfejs SPI2 konfigurując piny mikrokontrolera jako:

- PD1 – `SPI2_SCK`
- PD3 – `SPI2_MISO`
- PD4 – `SPI2_MOSI`.

Skonfigurowano pin PD7 jako pin wyjściowy odpowiedzialny za włączenie (podanie stanu niskiego), oraz wyłączenie (stan wysoki) transakcji pomiędzy mikrokontrolerem pracującym jako urządzenie master oraz żyroskopem pracującym w trybie slave. Interfejs SPI działa w trybie full-duplex. Częstotliwość zegara taktującego linię `SPI2_SCK` wynosi 10MHz, co jest maksymalną częstotliwością pracy układu gwarantującą poprawny przesył danych [3]. Dodatkowo na pinie PD2 uruchomiono obsługę przerwania zewnętrznego od żyroskopu na opadające zbocze.

Konfiguracja rejestrów żyroskopu

Rejestr	Bit	Opis
`CTRL_REG3`	`I1_Int1`	Uruchomienie generowania przerwań na pinie INT1 żyroskopu
`CTRL_REG3`	`H_Lactive`	Przerwanie sygnalizowane stanem niskim na pinie INT1 żyroskopu
`INT1_CFG`	`ZHIE`	Przerwanie generowane w momencie przekroczenia wartości zapisanej w parze rejestrów `INT1_THS_ZH` i `INT1_THS_ZL`
`INT1_THS_ZH` `INT1_THS_ZL`	Wartość: 5000	Rejestry przechowują MSB i LSB wartości 5tys., po której przekroczeniu generowane jest przerwanie
`INT1_DURATION`	WAIT	Przerwanie generowane jest jeśli odczytywana wartość utrzymuje się przez przynajmniej tyle cykli ile wpisano w bitach `INT1_DURATION[0:6]`
`INT1_DURATION`	BIT[0:6] = 20	Przerwanie generowane jest jeśli odczytywana wartość utrzymuje się przez przynajmniej 20 cykli odczytu (0.1s)
`CTRL_REG1`	Xen	Odczyt prędkości obrotowej wokół osi X
`CTRL_REG1`	Yen	Odczyt prędkości obrotowej wokół osi Y
`CTRL_REG1`	Zen	Odczyt prędkości obrotowej wokół osi Z
`CTRL_REG1`	DR0	Częstotliwość pomiarów 190Hz
`CTRL_REG1`	BW1	Częstotliwość graniczna 50Hz

4.2 Akcelerometr LSM303C

W przeciwieństwie do żyroskopu L3GD20, komunikacja z akcelerometrem odbywa się w trybie half-duplex. Ze względu na to, że interfejs SPI tego urządzenia podłączony jest do tego samego interfejsu SPI mikrokontrolera co żyroskop, wymagane jest przełączanie trybu pracy peryferium w trakcie działania programu. W tym celu wykorzystano makrodefinicje:

```
#define __SPI_DIRECTION_2LINES(__HANDLE__)    do{\
    CLEAR_BIT((__HANDLE__)->Instance->CR1, SPI_CR1_RXONLY | SPI_CR1_BIDIMODE |
        SPI_CR1_BIDIOE);\
    }while(0);

#define __SPI_DIRECTION_2LINES_RXONLY(__HANDLE__)    do{\
    CLEAR_BIT((__HANDLE__)->Instance->CR1, SPI_CR1_RXONLY | SPI_CR1_BIDIMODE |
        SPI_CR1_BIDIOE);\
```

```
    SET_BIT((__HANDLE__)->Instance->CR1, SPI_CR1_RXONLY);\
    }while(0);

#define __SPI_DIRECTION_1LINE_TX(__HANDLE__) do{\
    CLEAR_BIT((__HANDLE__)->Instance->CR1, SPI_CR1_RXONLY | SPI_CR1_BIDIMODE |
        SPI_CR1_BIDIOE);\
    SET_BIT((__HANDLE__)->Instance->CR1, SPI_CR1_BIDIMODE | SPI_CR1_BIDIOE);\
    }while(0);

#define __SPI_DIRECTION_1LINE_RX(__HANDLE__) do {\
    CLEAR_BIT((__HANDLE__)->Instance->CR1, SPI_CR1_RXONLY | SPI_CR1_BIDIMODE |
        SPI_CR1_BIDIOE);\
    SET_BIT((__HANDLE__)->Instance->CR1, SPI_CR1_BIDIMODE);\
    } while(0);
```

Listing 1: Makrodefinicje zmieniające tryb pracy SPI

Konfiguracja rejestrów akcelerometru

Rejestr	Bit	Opis
`CTRL_REG5_A`	`H_LACTIVE`	Przerwanie sygnalizowane stanem niskim na pinie XL akcelerometru
`CTRL_REG3_A`	`INT_XL_DRDY`	Przerwanie jest generowane, gdy dane zostaną zaktualizowane
`CTRL_REG4_A`	`IF_ADD_INC`	Automatyczna inkrementacja adresu rejestrów podczas jednorazowego odczytu lub zapisu kilku rejestrów po sobie w pamięci
`CTRL_REG4_A`	`I2C_DISABLE`	Wyłączenie komunikacji po stronie akcelerometru za pomocą interfejsu I2C
`CTRL_REG4_A`	`SIM`	Włączenie odczytu z i zapisu do pamięci akcelerometru za pomocą interfejsu SPI
`CTRL_REG1_A`	XEN	Odczyt przyspieszenia wzdłuż osi X
`CTRL_REG1_A`	YEN	Odczyt przyspieszenia wzdłuż osi Y
`CTRL_REG1_A`	ZEN	Odczyt przyspieszenia wzdłuż osi Z
`CTRL_REG1_A`	ODR0 i ODR1	Częstotliwość pomiarów 100Hz
`CTRL_REG1_A`	BDU	Dane w rejestrach `OUT_X_x_A`, `OUT_Y_x_A`, `OUT_Z_x_A` nie są aktualizowane dopóki poprzednie wartości nie zostaną odczytane

4.3 Przetwornik Audio DAC

Implementując komunikację z przetwornikiem cyfrowo analogowym CS43l22 wykorzystano bibliotekę BSP realizującą zarówno warstwę komunikacji z przetwornikiem przez interfejs I2C umożliwiającym konfigurację pracy przetwornika (częstotliwość pracy, liczba kanałów itp.) oraz komunikację przez interfejs SAI odpowiedzialny za transmisję danych w postaci cyfrowej do przetwornika Audio DAC. Wzorując się na dołączonym do biblioteki HAL przykładzie `SAI_AudioPlay` uruchomiono generator fali. Przykład ten posiada zaimplementowany kod odpowiedzialny za konfigurację peryferium do pracy i został on również wykorzystany do konfiguracji przetwornika w projekcie wprowadzając zmiany dotyczące np. częstotliwości granicznej generowanego sygnału. Transmisja próbek generowanej fali odbywa się poprzez DMA skonfigurowanego do pracy z peryferium SAI. Częstotliwość generowanego dźwięku zależna jest od ustawienia częstotliwości zegara taktującego SAI. Próbki przesyłane są synchronicznie do przetwornika CS43L22. Częstotliwość przesyłania danych zależna jest od częstotliwości podanej na linii SAI MCLK [4] oraz wielkości ramki danych.

Kolejne próbki sygnału wtaktowywane są do pamięci przetwornika z częstotliwością 48kHz. Modulacja częstotliwości generowanego sygnału sterowana jest liczbą próbek na okres fali. Przykładowo w celu osiągnięcia fali o częstotliwości 1Hz należy wygenerować 48 tys. próbek na okres.

Konfiguracja rejestrów przetwornika cyfrowo-analogowego

Adres	Nazwa rejestru	Ustawione bity/wartość	Opis
0x04	Power Control 2	`PDN_HPB`[1:0]=0b10 `PDN_HPA`[1:0]=0b10	Kanał prawy i lewy na wyjście słuchawkowe zawsze włączony, kanał na głośniki zawsze wyłączony
0x04	Power Control 2	`PDN_SPKB`[1:0]=0b11 `PDN_SPKA`[1:0]=0b11	Kanał prawy i lewy wyjściowy na głośniki zawsze wyłączony
0x05	Clocking Control	AUTO	Automatyczna detekcja częstotliwości taktowania zegara, dzielenie sygnału zegarowego otrzymanego na pinie MCLK przez 2
0x05	Clocking Control	MCLKDIV	Wewnętrzne dzielenie sygnału zegarowego otrzymanego na pinie MCLK przez 2
0x06	Interface Control 1	DAC-DIF[1:0]=0b01	Format danych zgodny z protokołem I^2S, wielkość danych do 24 bitów
0x1E	Beep & Tone Configuration	TREBCF[1:0]=11	Górna częstotliwość graniczna 15kHz
0x1A	PCMx Volume: PCMA	0x0A	Wzmocnienie sygnału odczytanego z linii SDIN na kanale PCMA o +5dB
0x1B	PCMx Volume: PCMB	0x0A	Wzmocnienie sygnału odczytanego z linii SDIN na kanale PCMB o +5dB

5 Opis działania programu

Po włączeniu mikrokontrolera następuje inicjalizacja peryferiów mikrokontrolera STM32L476VGT6, układów zewnętrznych umieszczonych na płytce Discovery (L32GD20, LSM303C, CS43L22) oraz struktur danych wykorzystywanych w programie.

Po inicjalizacji następuje kalibracja odczytów akcelerometru względem osi X. Wyznaczana jest wartość minimalna i maksymalna przyspieszenia wzdłuż tej osi. Różnica pomiędzy wartością maksymalną, a minimalną uzyskana podczas kalibracji wykorzystywana jest do obliczenia częstotliwości generowanej fali w zależności od wartości odczytanej wartości przyspieszenia względem tej osi. Proces kalibracji sygnalizowany jest poprzez miganie czerwonej diody. Podczas kalibracji należy obracać płytkę wokół osi Y akcelerometru. Zakończenie kalibracji następuje po naciśnięciu joistick'a. W tym momencie dioda zostaje wyłączona.

Następnie do bufora DMA kopiowany jest sygnał sinusoidalny. W kolejnym kroku uruchamiana jest transmisja danych z bufora za pomocą DMA w trybie cyklicznym do SAI.

Główna pętla programu obsługuje zdarzenia dotyczące obsługi informacji o przerwaniach z akcelerometru i żyroskopu. W zależności od odczytanych z tych sensorów wartości następuje zmiana generowanej fali, zmiana częstotliwości lub amplitudy. Przerwanie od żyroskopu generowane jest w sytuacji, gdy odczytana wartość prędkości obrotu wokół lokalnej osi Z przekroczy próg 5 000 (wartości bezwzględnej). W zależności od kierunku obrotu płytki przełączenie fali następuje do następnego lub poprzedniego typu. Aby zapobiec przypadkowemu przełączaniu typu fali, podczas ruchu należy mieć przyciśnięty przycisk joistick'a.

Ze względu na wrażliwość akcelerometru na drgania, w celu eliminacji gwałtownych zmian częstotliwości zastosowano filtr dolnoprzepustowy pod postacią ruchomej średniej (ang. moving average) z 20 ostatnich próbek.

Operacje zmiany częstotliwości wykonywana jest w funkcji `WG_changeFrequency`.

```
1  void WG_changeFrequency(int16_t xVal)
2    int32_t correctedXVal = (xVal + 16384) - audioState.min;
3    if (correctedXVal < 0) {
4      correctedXVal = 0;
5    } else if (correctedXVal > 32767) {
6      correctedXVal = 32767;
7    }
8    MA_addValue(&movingAvg, correctedXVal);
9
10   audioState.frequency = MA_getAverage(&movingAvg) * audioState.ratio + 50;
11   if (audioState.frequency < 50) {
12     audioState.frequency = 50;
13   } else if (audioState.frequency > 12000) {
14     audioState.frequency = 12000;
15   }
16
17   audioState.maxSamples = 48000 / audioState.frequency;
18 }
```

Listing 2: Zmiana częstotliwości generowanej funkcji

W liniach 2-7 następuje normalizacja odczytanej wartości do przedziału wartości od 0 do 32767. W linii 8 otrzymana wartości jest dodawana do średniej ruchomej. W linii 10 wyliczana jest wartość średnia z ostatnich 20 próbek i konwertowana jest ona na częstotliwość z zakresu od 0 do 11950Hz. Dodawana jest wartość 50 w celu przesunięcia częstotliwości do zakresu od 50Hz do 12kHz. W linii 17 wyliczana jest liczba próbek przypadająca na okres przy zadanej częstotliwości. Dolna granica częstotliwości wynika z dolnej częstotliwości granicznej ustawionej na 50Hz. Górna częstotliwość wynika z częstotliwości przesyłu próbek, oraz górnej częstotliwości granicznej ustawionej na 15kHz. Przy częstotliwości pracy przetwornika równej 48kHz, gdy generowana fala ma mieć częstotliwość 12kHz, to na okres przypadają 4 próbki (48kHz/12kHz = 4). Gdyby zejść do 3 próbek na okres, to uzyskana częstotliwość dźwięku wyniosłaby 16kHz co jest wartością przewyższającą górną częstotliwość graniczną.

Zmiany są aktualizowane w buforze w odpowiedzi na przerwania generowane przez DMA w momencie zakończenia transmisji połowy lub całości zawartości bufora:

```
void HAL_SAI_TxCpltCallback(SAI_HandleTypeDef *hsai) {
  updatePosition = BUFFER_SIZE / 2;
}

void HAL_SAI_TxHalfCpltCallback(SAI_HandleTypeDef *hsai) {
  updatePosition = 0;
```

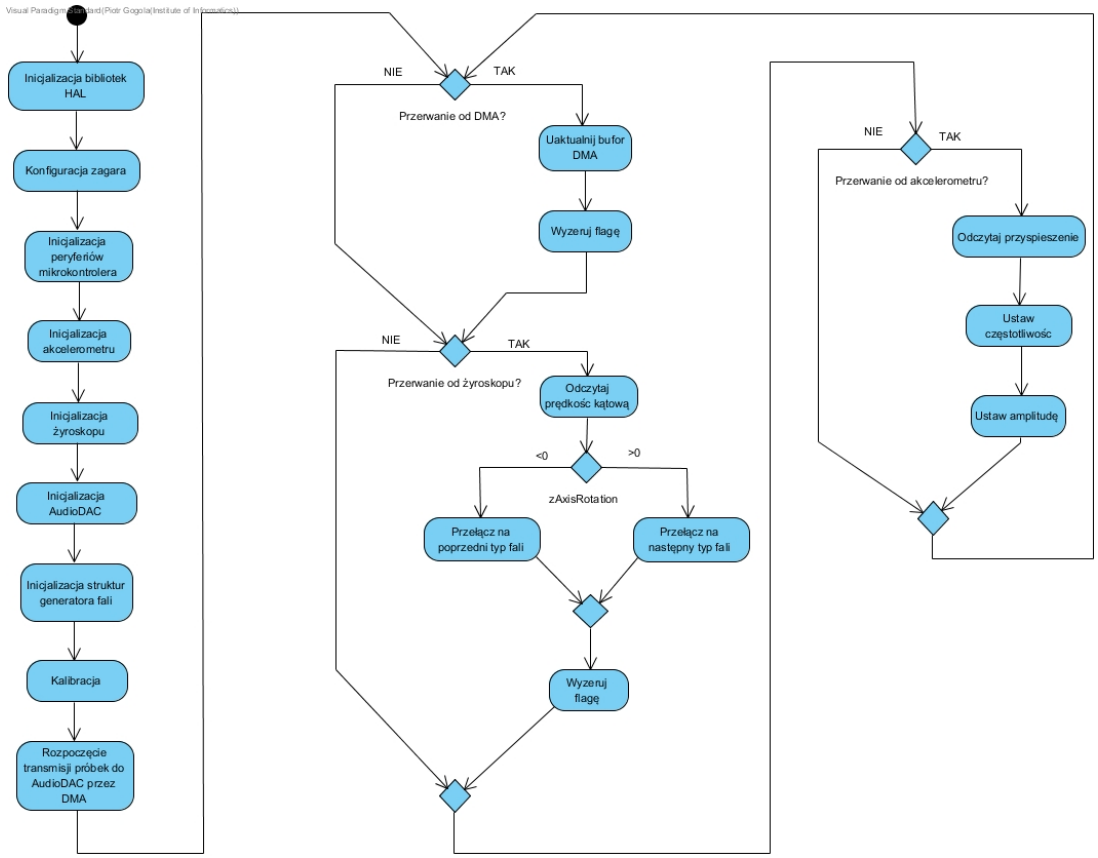
```
}
```

Listing 3: Funckje przerwań DMA

W trakcie przerwania ustawiana jest zmienna `updatePosition` informująca, od którego indeksu ma się rozpocząć aktualizowanie bufora.

Diagram sterowania

Diagram aktywności obrazujący sekwencję działania programu został przedstawiony na rysunku 3.



Rysunek 3: Diagram aktywności programu

6 Zadania niezrealizowane

Podczas projektu chcieliśmy zapewnić większą rozpiętość częstotliwości generowanego sygnału. Ze względu na sposób połączenia przetwornika cyfrowo-analogowego CS43L22 z mikrokontrolerem tylko na wyjściu peryferium Serial Audio Interface okazało się to utrudnione, stąd częstotliwość sygnału jest modulowana w sposób programowy powodując utratę faktycznej postaci generowanej funkcji przy wyższych częstotliwościach. Rozwiązaniem mogłoby okazać się wykorzystanie przetwornika cyfrowo-analogowego wbudowanego w mikrokontroler z wyjściem podłączonym do wejścia analogowego wzmacniacza audio będącego elementem urządzenia CS43L22 i praca w trybie Analog Input Passthrough. Przy takim rozwiązaniu częstotliwość sygnału analogowego byłaby regulowana przez częstotliwość pracy przetwornika.

Literatura

[1] ST. STM32 cross-series timer overview. July 2016.

[2] ST. Discovery kit with STM32L476VG MCU. March 2016.

[3] ST. L3GD20: 3-axis digital output gyroscope. June 2014.

[4] CIRRUS LOGIC. CS43L22, Low Power, Stereo DAC w/Headphone & Speaker Amps. March 2010.