Projekt

Sterowniki robotów

Dokumentacja

Piłka sterowana za pomocą żyroskopu

PSŻ

Skład grupy:
Stanisław Chędoska, 259354
Mikołaj Sęk, 258955

Termin: wt. TP 18:55

Prowadzący:
dr inż. Wojciech Domski

30 maja 2023

Spis treści

Bibilografia **21**

1 Opis projektu

Celem projektu jest stworzenie prostej gry bazującej na płytce STM32F429l-DISC1, wyposażonej m.in. w dotykowy ekran TFT LCD oraz żyroskop. Gra będzie polegać na manipulowaniu ruchem piłki wyświetlonej na ekranie, poprzez przechylanie płytki w różne kierunki (odczyty z żyroskopu). Dodatkowo, piłka będzie odbijać się od ścian wydając przy tym dźwięk generowany z zewnętrznego urządzenia podłączonego do odpowiedniego pinu GPIO z konwerterem cyfrowo-analogowym.

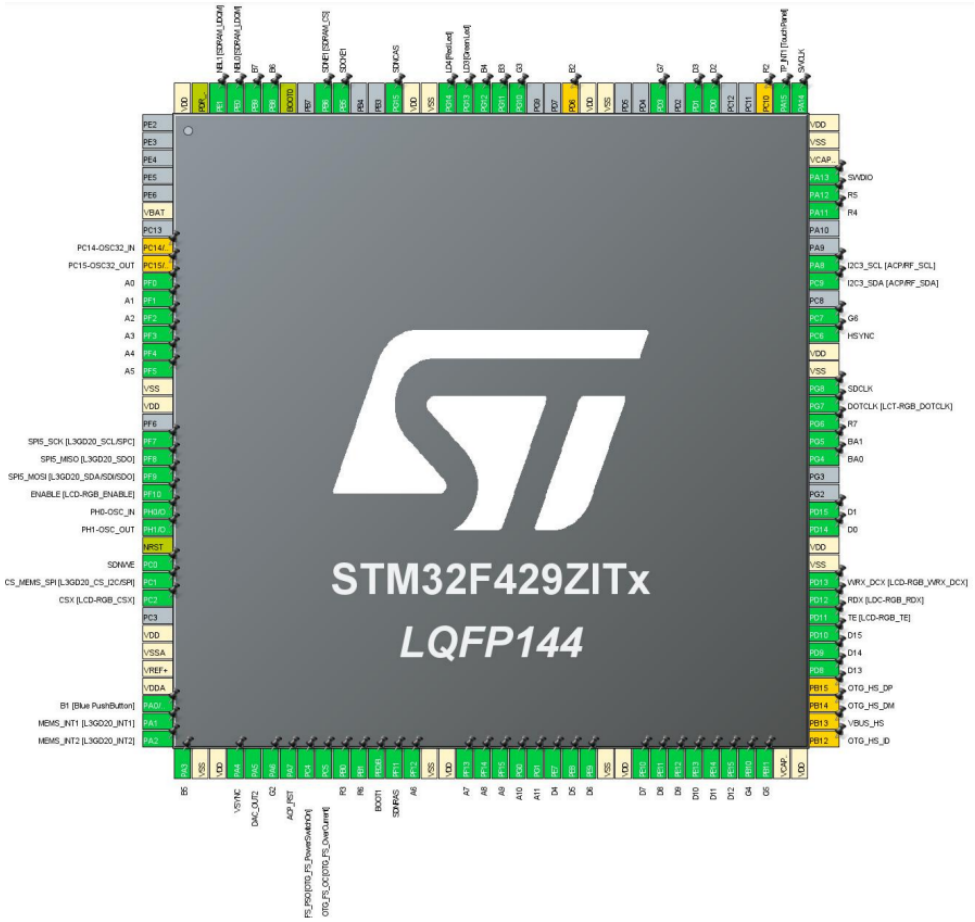
1.1 Założenia Projektowe

- Do rysowania piłki na ekranie wykorzystane zostanie układ kontrolujący LTDC.
- Do rysowania obrazu na ekranie LCD zostanie wykorzystany kontroler LTDC, sterujący wyświetlaczem za pomocą interfejsu RGB. Bufor z rysowanym obrazem będzie się znajdował w zewnętrznej pamięci SDRAM.
- Piłka zostanie przedstawiona jako zamalowane na jednolity kolor koło.
- Po zetknięciu się ze ścianą, wydany zostanie krótki 0.5 sekundowy dźwięk o stałej częstotliwości korzystając z przetwornika cyfrowo-analogowego.
- Żyroskop L3GD20 zwraca prędkości obrotowe. Aby uzyskać aktualny obrót, wartości te będą całkowane w czasie i w odpowiedni sposób przeliczane. Na podstawie pomiarów, piłka będzie liniowo przyśpieszała w kierunku wychylenia.
- wciśnięcie przycisku spowoduje sprowadzenie piłki na środek ekranu oraz wyzerowanie wyników całkowania pomiarów z żyroskopu (Pomiar wychylenia na podstawie żyroskopu, wraz z trwaniem czasu jest obarczony coraz większym błędem spowodowanym dryftem żyroskopu)

Docelowo pętla programu ma polegać na: sczytaniu pomiaru z żyroskopu i obliczeniu przechylenia. Następnie wyliczone zostanie przyśpieszenie piłki i nowa pozycja środka piłki po zastosowaniu przyśpieszenia. Jeśli nowa pozycja będzie znajdowała się w odległości równej (lub mniejszej) promieniowi od którejkolwiek z granic ekranu to piłka zostanie zatrzymana i wydany zostanie sygnał dźwiękowy. Jeśli nie to do bufora przechowującego piksele wpisane zostaną punkty znajdujące się w odległości (od środka piłki) mniejszej lub równej promieniowi w celu narysowania koła. Następnie kontroler LTDC wyświetli zapisany bufor na ekranie wykorzystując interfejs RGB.

2 Konfiguracja mikrokontrolera

Projekt piłki sterowanej zyroskopem zostanie wykonany na płytce STM32F429I-DISC1 zawierającą wyświetlacz LCD-THT o wymiarach 320 na 240 pikseli. Na rysunku poniżej przedstawiono uproszczoną konfigurację pinów mikrokontrolera.



Rysunek 1: Konfiguracja wyjść mikrokontrolera w programie STM32CubeMX

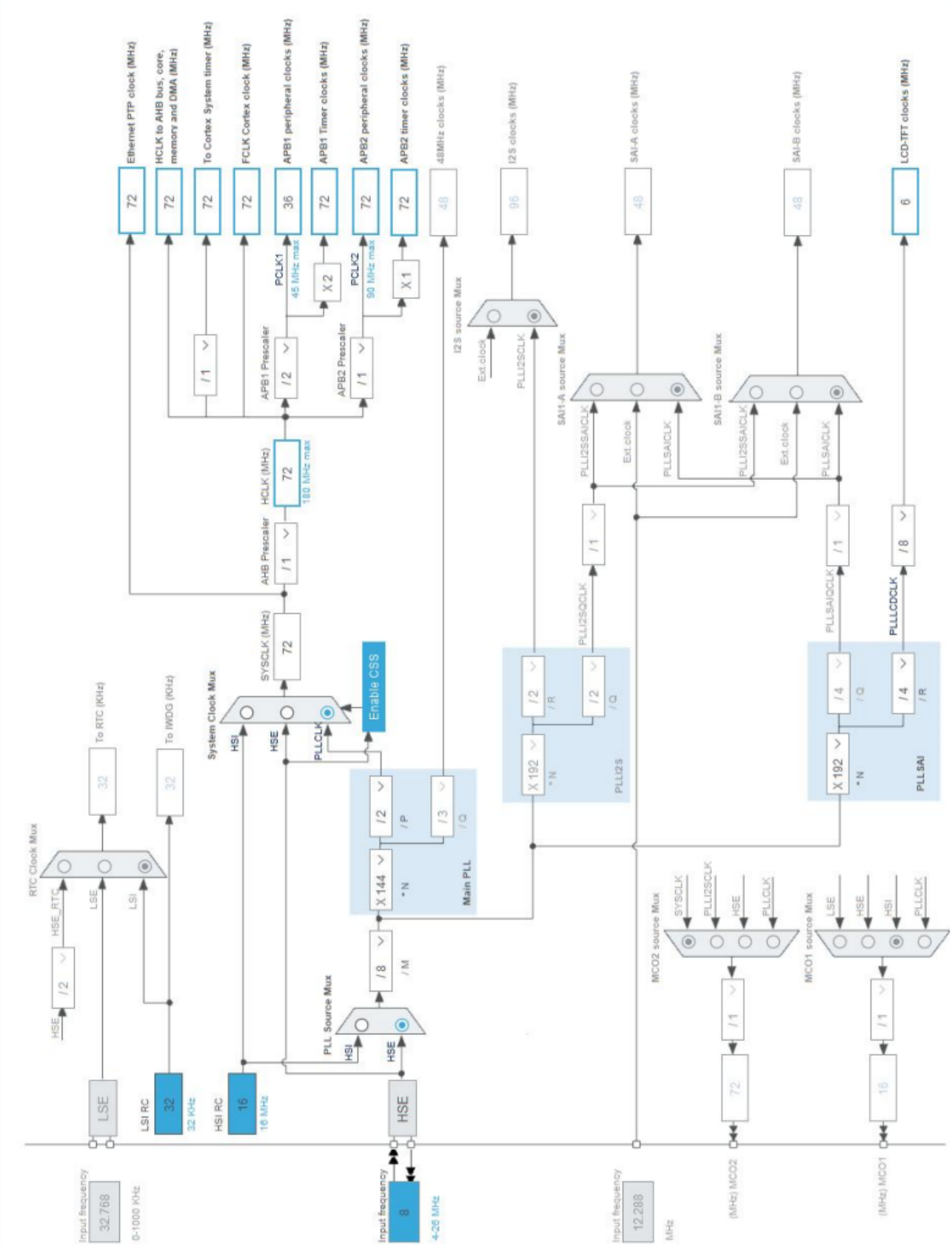
Względem poprzedniego etapu zmodyfikowana została konfiguracja zegarów mikrokontrolera, tak aby zegar taktujący układ LTDC otrzymywał sygnał 6 Mhz, Wprowadzone zmiany:

Main PLL:

- ***N** = X 144
- **/P** = / 2
- **/Q** = / 3

PLLSAI:

- ***N** = X 192
- **/Q** = / 4
- **/R** = / 4



Rysunek 2: Konfiguracja zegarów mikrokontrolera

2.1 Konfiguracja pinów

Numer pinu	PIN	Tryb pracy	Funkcja/etykieta
8	PC14	OSC32_IN*	
9	PC15	OSC32_OUT*	
10	PF0	FMC_A0	A0
11	PF1	FMC_A1	A1
12	PF2	FMC_A2	A2
13	PF3	FMC_A3	A3
14	PF4	FMC_A4	A4
15	PF5	FMC_A5	A5
19	PF7	SPI5_SCK	SPI5_SCK [L3GD20_SCL/SPC]
20	PF8	SPI5_MISO	SPI5_MISO [L3GD20_SDO]
21	PF9	SPI5_MOSI	SPI5_MOSI [L3GD20_SDA/SDI/SDO]
22	PF10	LTDC_DE	ENABLE [LCD-RGB_ENABLE]
23	PH0	OSC_IN	RCC_OSC_IN PH0-OSC_IN
24	PH1	OSC_OUT	RCC_OSC_OUT PH1-OSC_OUT
26	PC0	FMC_SDNWE	SDNWE
27	PC1	GPIO_Output	NCS_MEMS_SPI [L3GD20_CS_I2C/SPI]
28	PC2	GPIO_Output	CSX [LCD-RGB_CSX]
34	PA0/WKUP	GPIO_EXTI0	B1 [Blue PushButton]
35	PA1	GPIO_EXTI1	MEMS_INT1 [L3GD20_INT1]
36	PA2	GPIO_EXTI2	MEMS_INT2 [L3GD20_INT2]
37	PA3	LTDC_B5	B5
40	PA4	LTDC_VSYNC	VSYNC
41	PA5	DAC_OUT2	Wyjście sygnału audio
42	PA6	LTDC_G2	G2
43	PA7	GPIO_Output	ACP_RST
44	PC4	GPIO_Output	OTG_FS_PSO [OTG_FS_PowerSwitchOn]
45	PC5	GPIO_EXTI5	OTG_FS_OC [OTG_FS_OverCurrent]
46	PB0	LTDC_R3	R3
47	PB1	LTDC_R6	R6
48	PB2/BOOT1	GPIO_Input	BOOT1
49	PF11	FMC_SDNRAS	SDNRAS
50	PF12	FMC_A6	A6
53	PF13	FMC_A7	A7
54	PF14	FMC_A8	A8
55	PF15	FMC_A9	A9
56	PG0	FMC_A10	A10
57	PG1	FMC_A11	A11
58	PE7	FMC_D4	D4
59	PE8	FMC_D5	D5
60	PE9	FMC_D6	D6
63	PE10	FMC_D7	D7
64	PE11	FMC_D8	D8
65	PE12	FMC_D9	D9
66	PE13	FMC_D10	D10
67	PE14	FMC_D11	D11
68	PE15	FMC_D12	D12
69	PB10	LTDC_G4	G4

Tabela 1: Konfiguracja pinów mikrokontrolera cz.1

Numer pinu	PIN	Tryb pracy	Funkcja/etykieta
70	PB11	LTDC_G5	G5
77	PD8	FMC_D13	D13
78	PD9	FMC_D14	D14
79	PD10	FMC_D15	D15
80	PD11	GPIO_Input	TE [LCD-RGB_TE]
81	PD12	GPIO_Output	RDX [LDC-RGB_RDX]
77	PD8	FMC_D13	D13
78	PD9	FMC_D14	D14
79	PD10	FMC_D15	D15
80	PD11	GPIO_Input	TE [LCD-RGB_TE]
81	PD12	GPIO_Output	RDX [LDC-RGB_RDX]
82	PD13	GPIO_Output	WRX_DCX [LCD-RGB_WRX_DCX]
85	PD14	FMC_D0	D0
86	PD15	FMC_D1	D1
89	PG4	FMC_BA0	BA0
90	PG5	FMC_BA1	BA1
91	PG6	LTDC_R7	R7
92	PG7	LTDC_CLK	DOTCLK [LCT-RGB_DOTCLK]
93	PG8	FMC_SDCLK	SDCLK
96	PC6	LTDC_HSYNC	HSYNC
97	PC7	LTDC_G6	G6
99	PC9	I2C3_SDA	I2C3_SDA [ACP/RF_SDA]
100	PA8	I2C3_SCL	I2C3_SCL [ACP/RF_SCL]
103	PA11	LTDC_R4	R4
104	PA12	LTDC_R5	R5
105	PA13	SYS_JTMS-SWDIO	SWDIO
109	PA14	SYS_JTCK-SWCLK	SWCLK
110	PA15	GPIO_EXTI15	TP_INT1 [Touch Panel]
111	PC10*	LTDC_R2	R2
114	PD0	FMC_D2	D2
115	PD1	FMC_D3	D3
117	PD3	LTDC_G7	G7
122	PD6*	LTDC_B2	B2
125	PG10	LTDC_G3	G3
126	PG11	LTDC_B3	B3
127	PG12	LTDC_B4	B4
128	PG13	GPIO_Output	LD3 [Green Led]
129	PG14	GPIO_Output	LD4 [Red Led]
132	PG15	FMC_SDNCAS	SDNCAS
135	PB5	FMC_SDCKE1	SDCKE1
136	PB6	FMC_SDNE1	SDNE1 [SDRAM_CS]
139	PB8	LTDC_B6	B6
140	PB9	LTDC_B7	B7
141	PE0	FMC_NBL0	NBL0 [SDRAM_LDQM]
142	PE1	FMC_NBL1	NBL1 [SDRAM_UDQM]

Tabela 2: Konfiguracja pinów mikrokontrolera cz.2

Piny **PG11**, **PG12**, **PA3**, **PB8**, **PB9** służą do kodowania wartości jasności barwny niebieskiej piksela. **PA6**, **PG10**, **PB10**, **PB11**, **PC7**, **PD3** służą do kodowania wartości jasności barwny zielonej piksela. **PB0**, **PA11**,**PA12**, **PB1**, **PG6** służą do kodowania wartości jasności barwny niebieskiej piksela. Piny **PG7**, **PF10** czy **PC6** służą do kontroli, sterowania i taktowania wyświetlacza. Do komunikacji z żyrokopem L3GD20 wykorzystywane są piny **PF7** - SCK, **PF8** - MISO, **PF9** - MOSI. Pin **PA5** jest wykorzystywany jako wyjście przetwonika cyfrowo analogowego i jest podłączony do modułu głośnika.

2.2 LTDC

[3] Układ ten służy do kontroli ekranu wyświetlacza LCD-THT. Kontroler ten stale przesyła zawartość określonego buforu w którym znajdują się informacje o kolorach pikseli które mają zostać wyświetlone na ekranie. Ekran ma wymiary 240 pikseli w pionie i 320 pikseli w poziomie.

Parametr	Wartość
Synchronization for Width	
Horizontal Synchronization Width	10
Horizontal Back Porch	20
Active Width	240
Horizontal Front Porch	10
HSync Width	9
Accumulated Horizontal Back Porch Width	29
Accumulated Active Width	269
Total Width	279
Synchronization for Height	
Vertical Synchronization Height	2
Vertical Back Porch	2
Active Height	320
Vertical Front Porch	4
VSync Height	1
Accumulated Vertical Back Porch Height	3
Accumulated Active Height	323
Total Height	327
Synchronization for Height	
Horizontal Synchronization Polarity	Active Low
Vertical Synchronization Polarity	Active Low
Data Enable Polarity	Active Low
Pixel Clock Polarity	Normal Input
Backgruond Color	
Red	0
Green	0
Bluew	0

Tabela 3: Konfiguracja peryferium LTDC

Ponadto skonfigurowano poszczególne warstwy LTDC na których wyświetlany będzi obraz:

Parametr	Wartość
Window Position	
Layer 0 - Window Horizontal Start	0
Layer 0 - Window Horizontal Stop	240
Layer 0 - Vertical Start	0
Layer 0 -Window Vertical Stop	320
Pixel Parameters	
Layer 0 - Pixel Format	RGB565
Blending	
Layer 0 - Alpha constant for blending	255
Layer 0 - Default Alpha value	0
Layer 0 - Blending Factor1	Alpha constant x Pixel Alpha
Layer 0 - Blending Factor2	Alpha constant x Pixel Alpha
Frame Buffer	
Layer 0 - Color Frame Buffer Start Adress	0x0
Layer 0 - Color Frame Buffer Line Length (Image Width)	240
Layer 0 - Color Frame Buffer Number of Lines (Image Height)	320

Tabela 4: Konfiguracja Warstwy Layer 0 peryferium LTDC

Parametr	Wartość
Window Position	
Layer 1 - Window Horizontal Start	0
Layer 1 - Window Horizontal Stop	57
Layer 1 - Vertical Start	0
Layer 1 - Window Vertical Stop	56
Pixel Parameters	
Layer 1 - Pixel Format	RGB565
Blending	
Layer 1 - Alpha constant for blending	255
Layer 1 - Default Alpha value	255
Layer 1 - Blending Factor1	Alpha constant x Pixel Alpha
Layer 1 - Blending Factor2	Alpha constant x Pixel Alpha
Frame Buffer	
Layer 1 - Color Frame Buffer Start Adress	0x0
Layer 1 - Color Frame Buffer Line Length (Image Width)	56
Layer 1 - Color Frame Buffer Number of Lines (Image Height)	57

Tabela 5: Konfiguracja Warstwy Layer 1 peryferium LTDC

2.3 SDRAM

Jest to pamięć zewnętrzna, rozszerzająca pamięć dostępną do wykorzystania przez mikrokontroler. Na niej zapisywany będzie bufor przechowujący obraz który ma być rysowany na ekranie. Użycie SDRAM jest konieczne gdyż bufor obrazu może zajmować duże ilości pamięci (W szczególności w przypadku podwójnego buforowania).

Parametr	Wartość
SDRAM controll	
Bank	bank SDRAM 2
Number of column address bits	8
Number of row address bits	12
CAS latency	3
Write protection	Disabled
SDRAM common clock	2
SDRAM common burst read	Disabled
SDRAM comon read pipe delay	2

Tabela 6: Konfiguracja peryferium SDRAM cz.1

Parametr	Wartość
SDRAM timing in memory clock cycles	
Load mode register to active delay	2
Exit self-refresh delay	7
Self-refresh time	4
SDRAM common row cycle delay	7
Write recovery time	3
SDRAM common row precharge delay	2
Row to column delay	2

Tabela 7: Konfiguracja peryferium SDRAM cz.2

2.4 DAC

Konwerter Cyfrowo-analogowy będzie służył do wytworzenia sygnału o zadanej częstotliwości (od 20Hz do 20kHz) by wytworzyć dźwięk.

Parametr	Wartość
DAC Out2 Settings	
Output Buffer	Enable
Trigger	Timer 2 Trigger Out Event
Wave generation mode	Disabled

Tabela 8: Konfiguracja peryferium DAC

2.5 SPI

Za pomocą komunikacji na SPI5 odbierane będą pomiary z żyrokopu L3GD20.

Parametr	Wartość
Basic Parameters	
Frame Format	Motorola
Data Size	8bit
First Bit	MSB First
Basic Parameters	
Prescaler (for Baud Rate)	16
Baud Rate	4.5 Mbit/s
Clock Polarity (CPOL)	LOW
Clock Phase (CPHA)	1 Edge
Advanced Parameters	
CRC Calculation	Disabled
NSS Signal Type	Software

Tabela 9: Konfiguracja peryferium SPI5

2.6 TIM2

Timer 2 jest wykorzystywany do wyzwalania wpisu nowej wartośći do przetwornika cyfrowo-analogowego. Za każdym razem gdy licznik doliczy do ockreślonej wartości wyzwalany jest Update Event. Cżęstotliwośc timera zostałą ustawiona na 19,726 kHz.

$$f_{update} = \frac{F_{clk}}{(Prescaler + 1) \cdot (Period + 1)} = \frac{72Mhz}{73 \cdot 51} \approx 19,726kHz \tag{1}$$

Parametr	Wartość
Counter Settings	
Prescaler (PSC - 16 bits value)	72
Counter Mode	Up
Counter Period	50
Internal Clock Division (CKD)	No Division
auto-reload preload	Disable
Trigger Output (TRGO) Parameters	
Master/Slave Mode (MSM bit)	Disable
Trigger Event Selection	Update Event

Tabela 10: Konfiguracja peryferium Timera TIM2

2.7 DMA

DMA umożliwia bezpośrednią wymianę danych pomiędzy pamięciącia a układami peryferyjnymi. Pozwala na odciążenie procesoraprzy przesyłaniu danych. Jest to szczególnie przydatne pdzy odtwarzaniu audio za pomocą DAC gdzię z okresloną przez timer cżęstotliwośćią trzeba przesyłać do przetwornika kolejne wartości, tak aby umożliwośc wytworzenie fali dziwiękowej. Zastosowanie DMA pozwala ograniczyć pracę procesora przy tej operacji jedynie do zainicjoalizowania odsłuchu dźwięku, Następnie DMA przesyła kolejne wartośći zapisane w pamieci RAM do przetwornika. Wykorzystano DMA2, kanał 6.

Parametr	Wartość
DMA Mode and Configuration	
DMA Request	DAC2
Stream	DMA1 Stream 6
Direction	Memory To Peripheral
Priority	High
DMA Request Settings	
Mode	Normal
Peripheral	
Increment Address	-
Data Width	Word
Burst Size	-
Memory	
Increment Address	-
Data Width	Word
Burst Size	-

Tabela 11: Konfiguracja peryferium Timera TIM2

2.8 RCC

[1] Peryferium kontrolujące zegary systemowe mikrokontrolera, Umożliwia wykorzystanie zewnętrznego źródłą HSE poprzez wybranie opcji: **High Speed Clock** - *Crystal/Ceramic Resonator*.

Parametr	Wartość
System Parameters	
VDD voltage (V)	3.3V
Instruction Cache	Enabled
Prefetch Buffer	Enabled
Data Cache	Enabled
Flash Latency(WS)	2 WS (3 CPU cycle)
RCC Parameters	
HSI Calibration value	16
TIM Prescaler Select	Disabled
HSE Startup Timeout	100
LSE Startup Timeout	5000
Power Parameters	
Power Regulator Voltage	Power Regulator Voltage Scale 3
Power Over Drive	Disabled

Tabela 12: Konfiguracja peryferium RCC

3 Urządzenia zewnętrzne

Do tej pory w projekcie wykorzystano żyroskop L3GD20 oraz wyświetlacz LCD-THT.

3.1 Żyroskop - L3GD20

Dane z żyroskopu są wykorzystywane do określenie aktualnego wychylenia urządzania względem poziomu. Komunikacja z żyroskopem odbywa się za pomocą interfejsu SPI (może być także po I2C). W każdym cyklu w pętli while odczytywane są wszystkie 6 rejestrów z danymi o prędkości obrotowej (po 16 bitów na każdą oś, XYZ). Wartości z żyroskopu to prędkości obrotowe wokół osi, więc aby uzyskać orientację urządzenia w przestrzeni względem punktu początkowego, należy scałkować wartości i po odpowiednich transformacjach otrzyma się wartości kątów. [4] Układ L3GD20 poza możliwością pomiarów prędkości kątowych, jednostka pozwala także pobierać wartości zmierzonej temperatury otoczenia (maksymalna częstotliwość pomiarów to 1 Hz). Z racji tego że maksymalna częstotliwość SPI w jakiej możliwa jest komunikacja z żyroskopem wynosi 10 MHz, interfejs SPI został skonfigurowany po stronie mikrokontrolera na 4.5 Mhz.

Rejestr	Wartość
WHO_AM_I (0x20)	0xD4
CTRL_REG1 (0x20)	0x0F
CTRL_REG2 (0x21)	0x00
CTRL_REG3 (0x22)	0x00
CTRL_REG4 (0x23)	0x20
CTRL_REG5 (0x24)	0x10

Tabela 13: Konfiguracja rejestrów żyroskopu L3GD20

- **WHO_AM_I** - Rejestr z informacją o identyfikatorze urządzenia. Wykorzystywany do sprawdzenia poprawności komunikacji.
- **CTRL_REG1** - Określono częstotliwość danych wyjściowych na 95Hz (**ODR** - Output data rate), Aktywowanotryb normalnej pracy i aktywowano odczyty ze wszystkich 3 osi.
- **CTRL_REG2** - ustawiono filtr górnoprzepustowy w tryb domyślny. Określono minimalną częstotliwość graniczną filtru na 7.2 Hz
- **CTRL_REG3** - Wyłącznie wszystkich przerwań, program będzie sczytywał dane w każdym kolejnym cyklu działania programu.
- **CTRL_REG4** - Określono skalę na 2000 dps (degrees per second). Wybrano tryb interfejsu SPI na 4 przewodowy z CS.

- **CTRL_REG5** - aktywuje filtr górnoprzepustowy. Wyłączenie kolejki FIFO.

Rejestr	Wartość
OUT_X_L (0x28)	READ
OUT_X_H (0x29)	READ
OUT_Y_L (0x2A)	READ
OUT_Y_H (0x2B)	READ
OUT_Z_L (0x2C)	READ
OUT_Z_H (0x2D)	READ

Tabela 14: Konfiguracja rejestrów żyroskopu L3GD20

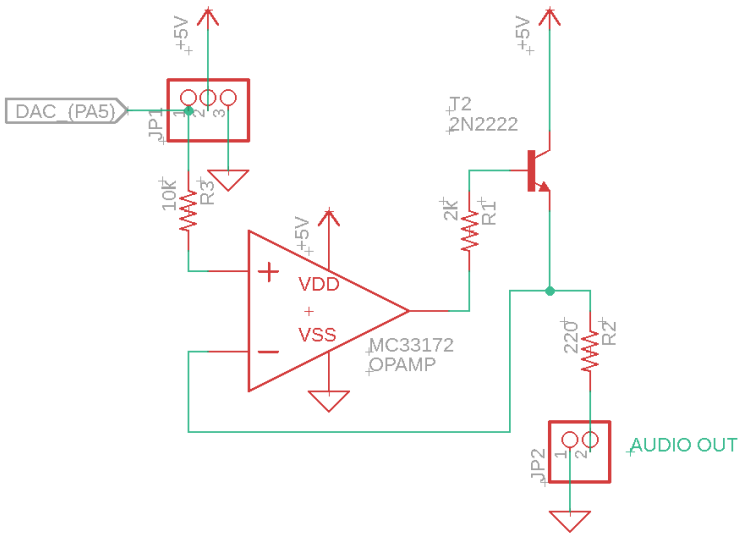
- **OUT_X_L, OUT_X_H** - Prędkość obrotowa wokół osi X (H - starszy bajt, L - młodszy bajt)
- **OUT_Y_L, OUT_Y_H** - Prędkość obrotowa wokół osi Y (H - starszy bajt, L - młodszy bajt)
- **OUT_Z_L, OUT_Z_H** - Prędkość obrotowa wokół osi Z (H - starszy bajt, L - młodszy bajt)

układ żyroskopu L3GD20 posiada rónież kolejkę FIFO, lecz nie jest ona wykorzystywana, układ działa w trybie Bypass, czyli dane w rejestrach do odczytu są nadpisywane jak tylko pojawi się nowy pomiar.

4 Projekt elektroniki

Aby zapewnić pewien stopień bezpieczeństwa płytki i zwiększyć możliwości prądowe, wyjście audio DAC z pinu *PA5* zostało podłączone do bufora w postaci wzmacniacza operacyjnego *MC33172* działjaącego w pętli sprzężenia zwrotnego w tranzystorem *T2* (Doprowadza napiecie odkładające się na rezystorze *R2* by było równe napięciu zadanemu na wejściu nieodwracającym). Do podłaczenia głośnika wykorzystano dodatkowy tranzystor z racji na niską wydajnośc prądową wybranego wzamcniacza operacyjnego. Wybrany głośnik ma rezystancję równą 8Ω

Na wyjścou pinu PA5 z przetwornika cyforowo-anologowego będą pojawiały się napiecia w zakresie od 0V do 3.3V. Układ będzie zasilany bezpośrednio z płytki napięciem 5V.



Rysunek 3: Schemat sterownika głośnika

4.1 Lista elementów

Do wykoannia tego prostego układu wykorrzystano:

Element	informacje
Lista Elemntów	
Wzmacniacz Operacyjny	MC33172
Rezystor [R3]	10KΩ
Rezystor [R2]	220Ω
Rezystor [R1]	2kΩ
Tranzystor NPN [T2]	2N2222
Głoścnk [AUDIO OUT]	8Ω

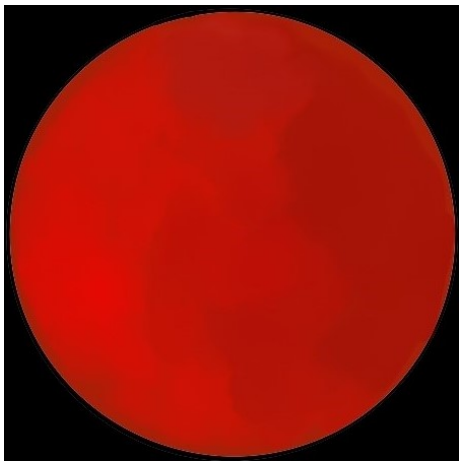
Tabela 15: Lista wykorzystanych elementów

5 Opis działania programu

Do obsługi wykorzystano bibliotekę ili9341 która inicjalizuje ekran, umożliwia ona wygodne skonfigurowanie działania ekranu LCD. Poza tym wukorzystano funkcje z pliku *stm32f4xx_hal_ltd.c* służące między innymi do przeładowania zawartości ekranu czy przesunięcia pozycji wybranej warstwy.

5.1 Obraz piłki

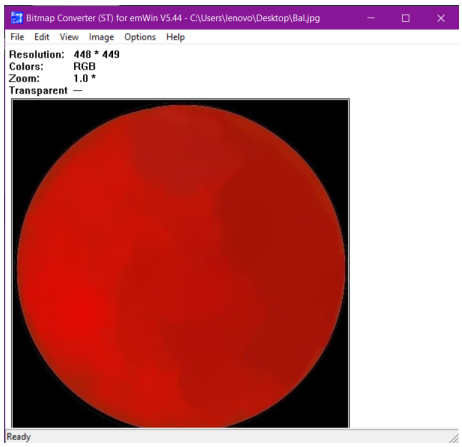
Pierwszym krokiem do wyświetlenia piłki na ekranie LCD, jest samo jej utworzenie. Obraz został narysowany za pomocą środowiska Adobe Fresco. Pierwszą warstwę zdecydowano się pomalować na czarno, ze względu na prostszą konfigurację tła ekranu, która musi być w tym samym kolorze. Następnie, obraz zapisany w pliku z rozszerzeniem `.jpg` przycięto możliwie blisko krawędzi koła w domyślnym edytorze.



Rysunek 4: Finalny obraz piłki wyświetlany na ekranie LCD

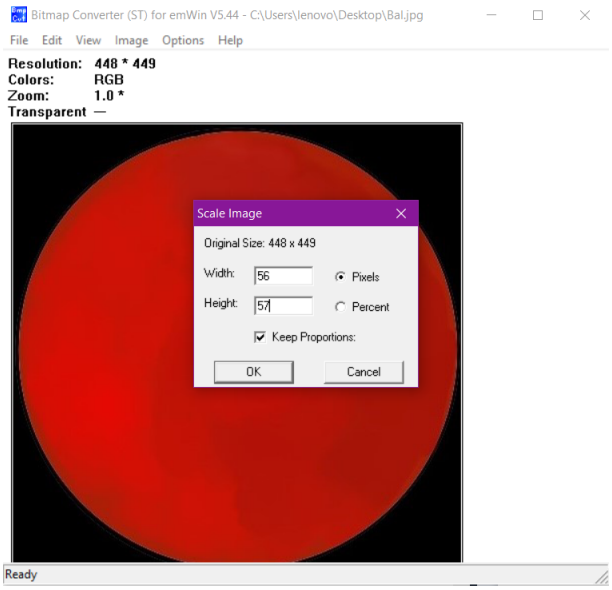
Ponieważ projekt nie opiera się na wyświetlaniu dużej ilości różnych obrazów, nie ma potrzeby korzystania z zewnętrznych pamięci. Dlatego też powstałą grafikę należało wgrać do pamięci **Flash**. W tym celu obraz przekonwertowano do pliku z rozszerzeniem `.c` za pomocą narzędzia dostarczonego wraz ze środowiskiem STM32CubeIDE. Program nazywa się "BmpCvtST"i można go znaleźć w katalogu: `C:\Users\$USER\STM32Cube\Repository\STM32Cube_FW_F4_V1.27.1\Middlewares\ST\STemWin\Software`.

Aby wczytać obraz do konwersji, należy wejść w zakładkę `File` → `Open` i wybrać żądany plik, w tym przypadku piłkę.



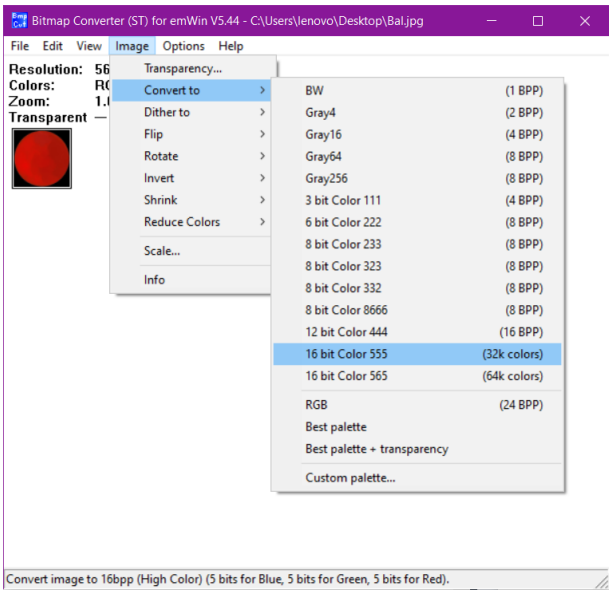
Rysunek 5: Obraz wgrany do programu BmpCvtST

Obraz został przeskalowany do takiego rozmiaru, aby zajmował około 20% szerokości ekranu LCD, która wynosi 240 pikseli. Nowe wymiary obrazu to 56x57 pikseli. Opcję skalowania można znaleźć w zakładce `Image` → `Scale`.



Rysunek 6: Przeskalowanie obrazu

Zgodnie z dokumentacją techniczną [?], obraz przekonwertowano do 16 bitowego modelu RGB565 wykonując następujące kroki: `Image` → `Convert to` → `16 bit Color 565`. Na samym końcu, w oknie "Format Specification"wybrano opcję "High Color (565), red and blue swapped".



Rysunek 7: Konwersja do modelu RGB565 16 bitowego

Gotową grafikę zapisano jako plik z rozszerzeniem `.c` (jedyna możliwa opcja). Sugerując się konstrukcją plików obrazów w przykładzie dostarczonym od producenta, zmieniono typ tablicy z typu zdefiniowanego przez narzędzie BmpCvtST na typ `const uint16_t`. Plik ze zmienionym ręcznie rozszerzeniem o nazwie "ball_graphic.h"umieszczono w katalogu `Core/Inc` projektu.

5.2 Konfiguracja LCD

Konfiguracja została przeprowadzona w oparciu o dokumentację techniczną kontrolera LTDC [?] oraz przykład załączony przez producenta o nazwie "LTDC_Display_2Layers". W projekcie wykorzystyna jest jedynie warstwa druga, zatem konfiguracja pierwszej została pominięta. Za ustawienie całego ekranu LCD odpowiada funkcja `MX_LTDC_Init()`. Rozmiar samej warstwy zmieniono na 60x60 pikseli, dokładniej mówiąc zmodyfikowano współrzędne jej narożników, tj.: X_0, $Y_0 = 0$ oraz X_1, $Y_1 = 60$. Współczynnik "alpha"zmieniono na wartość 255 (maksymalną), co oznacza nieprzeźroczystość warstwy. Zaktualizowano wartości rozmiarów obrazu wyświetlanego na warstwie, czyli wspomniane 56 pikseli szerokości oraz 57 pikseli wysokości. Do składnika `FBStartAdress` przypisano adres początkowy tablicy z piłką. Na koniec do składowych RGB warstwy drugiej jak i tła wpisano wartość 0, co przekłada się na kolor czarny samego ekranu i wyodrębnienie okrągłego kształtu piłki.

Po poprawnej kompilacji, obraz został wyświetlony w lewym górnym rogu ekranu. Do wstępnego testu poruszania się piłki po ekranie, zaimplementowano funkcję "ball_bounce()". Funkcja ta aktualizuje położenie piłki na ekranie zmieniając jej współrzędne o pewną stałą wartość. Po zetknięciu się ze ścianą ekranu, piłka odbija się i leci w przeciwnym kierunku. Za ustawienie nowej pozycji warstwy odpowiada funkcja `HAL_LTDC_SetWindowPosition_NoReload()` a za odświeżenie ekranu LCD – funkcja `HAL_LTDC_Reload()`.

5.3 Wyświetlanie piłki na ekranie w określonej pozycji

[6] [2] Z racji tego że piłka jest jedynym elemntem wyświetlanym w jednym momencie na ekranie, wykorzystano dostępność i funkcjonalnosc dwóch warstw w układzie peryferyjnym LTDC.

- pierwsza warstwa *Layer 0* została skonfigurowana jako tło o wymirach całego ekranu 240 na 320 pikseli, przedstawia ono poprostu czarne tło, na którym będzie poruszać się piłka.

- druga warstwa *Layer 1* to warstwa na której wyświetlana jest piłka, ma ona okreslony rozmiar 56 na 57 pikseli, czyli rozmiar piłki. tablica pikseli z których składa się obrazek piłki została zdefiniowana w pliku nagłówkowym *ball_graphic.h*. Jest ona zapisana (zapisana w pamieci programowej flash mikrokontrolera) w postaci jednowymiarowej tablicy typu *uint16_t*, W tablicy znajduje sie zapisanych 3192, 16-bitowych wartości okreslających barwę jednego piksela. Jako adres pamięci gdzie zapisany zostal obraz piłki zainicjowano warstwę przekazujac wskaźnik do tablicy:

```
pLayerCfg1.FBStartAdress = (uint32_t)&ball2;
HAL_LTDC_ConfigLayer(&hltdc, &pLayerCfg1, 1);
```

każdy piksel jest określony poprzez 16 bitową zmienną *uint16_t* ponieważ tak został skonfigurowany układ LTDC, by wyświetlać obraz w formacie RGB565 gdzie kolory czerwony i niebieski są określone 5 bitami (32 odcienie) a kolor zielony 6 bitami (64 odcienie) co razem pozwala uzyskać $2^{16} = 65536$ różnych kolorów.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
r7	r6	r5	r4	r3	g7	g6	g5	g4	g3	g2	b7	b6	b5	b4	b3

Rysunek 8: RGB565

Piłka jest odrysowywana w odpowiednim miejscu poprzez manipulowanie pozycją warstwy *Layer 1* na ekranie. W tym celu wykorzystywana jest funkcja:

```
HAL_LTDC_SetWindowPosition_NoReload(&hltdc, m_ball.Y_screen_pos, m_ball.
    X_screen_pos, 1);
```

gdzie argumentami są pozycja X i Y piłki, oraz indeks warstwy na której znajduje się piłka. następnie wywoływana jest funkcja przełamująca zawartość ekranu. Po zakończeniu operacji przeładowania ekranu pojawia się sygnalizujące o tym przerwanie, które zwalnia globalną flagę i program przechodzi dalej.

wstawianiem pikseli w odpowiednie miejsca na ekranie zajmuje się kontroler LTDC, należy zadać odpowiednie wartości współrzędnych warstwy na której znajduje się piłka by móc kontrolować "przemieszczanie"się piłki po ekranie.

5.4 Pobieranie danych z żyroskopu

Jak opisano wyżej wykorzystano układ żyroskopu L3GD20, z którym komunikowano się poprzez interfejs SPI. Aby umożliwić łatwe wpisywanie wartości i odczyt z poszczególnych rejestrów układu stworzone zostały odpowiednie funkcje:

```
void L3GD20_send(uint8_t address, uint8_t data, L3GD20 *L3GD20_data);
uint8_t L3GD20_recive(uint8_t addres, L3GD20 *L3GD20_data);
```

Ponieważ dane prędkości obrotowych wokół poszczególnych osi są zapisywane w dwóch oddzielnych rejestrów (H - starszy bajt i L - młodszy bajt), więc aby odczytać wartości trzeba odczytać najpierw pierwsze 8 bitów z rejestru L a następnie połączyć je za pomocą operacji OR ze starszymi bitami przesuniętymi o 8 pozycji w lewo,

```
L3GD20_data->Y_raw = L3GD20_recive(L3GD20_OUT_Y_L, L3GD20_data);
L3GD20_data->Y_raw |= L3GD20_recive(L3GD20_OUT_Y_H, L3GD20_data) << 8;
```

Z racji tego że w sytuacji w której układ pozostawał zupełnie bez ruchu a odczyty z jego rejestrów zmierzonych wartości nie były zerowe, doświadczalnie wyznaczone przesunięcie do odczytu z każdej osi. Po pomnożeniu tych wartości przez stałą **SENSITIVITY = 0.1** otrzymano wartość prędkości obrotowych w stopniach na sekundę. Następnie wartości odczytane z żyroskopu i przetworzone na stopnie na sekundę zostają scałkowane w celu otrzymania orientacji układy mikrokontrolera.

$$\phi = \int_{t_{start}}^{t_{now}} \omega \, dt. \tag{2}$$

t_{start} określa czas od któego rozpoczęto całkowanie, t_{now} określa chwilę obecną.

```
ball_data->ctrlX_angle += L3GD20_data->X_val * delta_time;
ball_data->ctrlY_angle += L3GD20_data->Y_val * delta_time;
ball_data->ctrlZ_angle += L3GD20_data->Z_val * delta_time;
```

Ponieważ odczyt z żyroskopu i całkowanie tych wyników w celu otrzymania orientacji wiąże się zwiększający się błędem rośnie zgodnie z zależnością

$$|\phi_{true} - \phi_{output}| \approx \sqrt{t_{now} - t_{start}} \tag{3}$$

dlatego wprowadzono możliwość przestawienia wartości t_{start} na chwilę obecną t_{now} co zeruje błąd i wylicza aktualną orientację od momentu przestawienia, W tym celu wykorzystano przycisk znajdujący się na płytce, wciśniecie go zeruje wartość dryftu. W ten sposób wyliczona zostaje aktualna orientacja układu z mikrokontrolerem a wyniki wychylenia zostają zapisane do struktury *Ball_control_data* która wykorzysta je do obliczania dynamiki piłki.

5.5 Dynamika i fizyka piłki

Stworzony został dodatkowy zestaw funkcji który przetwarza orientację mikrokontrolera (za pomocą żyroskopu) i wylicza na tej podstawie odpowiednie siły i przyśpieszenia jakie działają na wirtualną piłkę w wyniku odziaływania grawitacji. Na początku wyliczane jest przyśpieszenie piłki zgodnie ze wzorem (przykład dla obliczania pozycji na osi X): (g - dowolny współczynnik, niekoniecznie siła ciężkości)

$$\ddot{x} = sin(\phi_x) \cdot g \tag{4}$$

$$x = \int\int \ddot{x} \tag{5}$$

Wzrór ten jest realizowany przez poniższy kod:

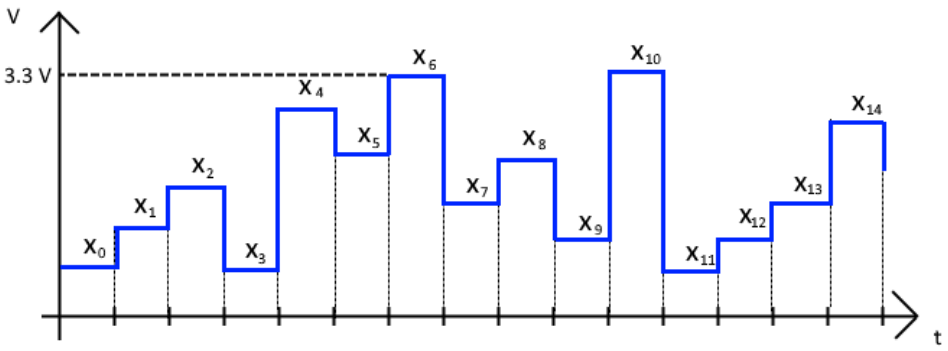
```
1   float delta_time = delta_time_ms / 1000.0;
2   float X_radAngle = (ball_data->ctrlX_angle * PI_CONST / 180.0);
3   float Y_radAngle = (ball_data->ctrlY_angle * PI_CONST / 180.0);
4
5   ball_data->X_screen_speed += sinf(X_radAngle) * GRAVITY_CONST * delta_time;
6   ball_data->Y_screen_speed += sinf(Y_radAngle) * GRAVITY_CONST * delta_time;
7
8   if(ball_data->X_screen_speed > 0)
9       ball_data->X_screen_speed -= FRICTION_CONST * delta_time;
10  else
11      ball_data->X_screen_speed += FRICTION_CONST * delta_time;
12
13  if(ball_data->Y_screen_speed > 0)
14      ball_data->Y_screen_speed -= FRICTION_CONST * delta_time;
15  else
16      ball_data->Y_screen_speed += FRICTION_CONST * delta_time;
17
18  ball_data->X_screen_pos += (ball_data->X_screen_speed * delta_time);
19  ball_data->Y_screen_pos += (ball_data->Y_screen_speed * delta_time);
```

W kodzie uwzględniono również tarcie uniemożliwiające piłce nieskończony ruch. Siła tarcia oddziałująca na piłkę jest aplikowana zawsze w kierunku przeciwnym do kierunku w którym aktualnie porusza się piłka, przez to piłka w sytuacji braku wychylenia piłka zawsze wytraci swoją energię i zatrzyma się. Ponadto stworzono funkcje zajmującą się odbijaniem piłki od ścian, polega na zmianie prostopadłej do ściany wartości prędkości na przeciwny znak. Dodatkowo przy każdym uderzeniu od ściany wytracana jest energia piłki poprzez pomnożenie przez odpowiedni współczynnik ($0 < k < 1$) aktualnej prędkości ruchu piłki.

```
1   ball_data->X_screen_speed = -X_screen_speed * WALL_ENERGY_LOSS_CONST;
```

5.6 Generowanie melodii

Aby wygenerować z głośników słyszalny dla człowieka dźwięk konieczne jest zadanie na ten głośnik sygnału zmiennego o częstotliwości w zakresie od 20 Hz do 20kHz. W pliku *ball_sound.h* stworzona została tablica wartości typu *uint32_t*. W tablicy tej znajdują się kolejne próbki dźwięku które będą zadawane na przetwornik cyfrowo-analogowy. Pobierając kolejne próbki w określonych odstępach czasu i nastawiając wyjście mikrokontrolera *PA5* adekwatnym do cyfrowej wartości próbki napięciem można wytworzyć dowolny sygnał zmienny w czasie.

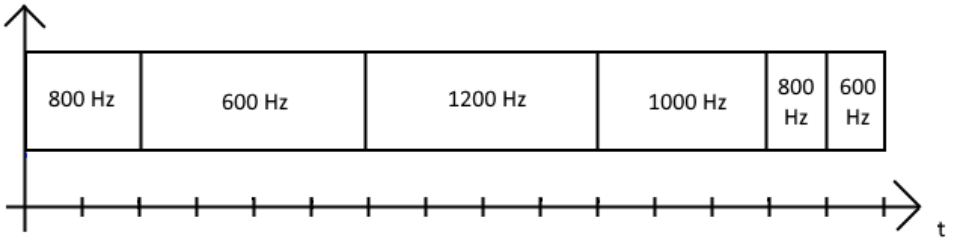


Rysunek 9: Sygnał na wyjściu z przetwornika DAC

Zdecydowano że do generowania dźwięku wykorzystany zostanie sygnał odwzorowujacy sinusoidę o zadanej częstotliwości. Za pomocą funkcji

```
addSineWaveAt(uint32_t frequency, uint32_t amplitude, uint32_t timer_frequency
    , uint32_t begin, uint32_t end)
```

można stworzyć dowolną melodię. Funkcja pozwala zdefiniować częstotliwość sinusoidy, amplitudę (wychylenie od poziomu 2048), początek oraz koniec odtwarzania sygnału w postaci indeksów pozycji wcześniej wspomnianej tablicy posiadającej wszystkie próbki. Aby poprawnie obliczyć częstotliwość sinusoidy (w kontekście już odtwarzanego sygnału) konieczna jest znajomość częstotliwości timera TIM2 który odpowiada za pobieranie kolejnych próbek i dyktuje szybkość odtwarzania sygnału.



Rysunek 10: Zmiana częstotliwości sygnału w czasie

Odpowiednia kombinacja sygnałów o określonych częstotliwościach odgrywanych na określonych odcinkach czasu pozwala na wytworzenie melodii. Do wytworzenia sinusoidy w określonej częstotliwości zapisanej w postaci zdyskretyzowanych próbek w tablicy wykorzystano następujący wzór:

$$\{i \in \mathbb{N} : i < K\}, \quad x_i = A \cdot sin(2 \cdot \pi \cdot i \cdot \frac{f}{f_{TIM2}}) \tag{6}$$

gdzie A to amplituda sygnału (w praktycznej implementacji przybiera wartości od 0 do 2048), i to indeks kolejnej próbki w tablicy, f to częstotliwość sygnału wyjściowego, f_{TIM2} to częstotliwość z jaką wystawiane są kolejne próbki. Wzór 6 ten jest realizowany za pomocą poniższego kodu:

```
void addSineWaveAt(uint32_t frequency, uint32_t amplitude, uint32_t
    timer_frequency, uint32_t begin, uint32_t end)
{
amplitude /= 2;
float period = 1.0 / timer_frequency;
if(end > SAMPLE_NUMBER)
{
  end = SAMPLE_NUMBER;
}
for(int t = begin; t < end; t++)
{
  int sample = amplitude * sinf(2.0 * PI_CONST_SOUND * t * period * frequency);
  sound_wav[t] = sample + (int)sound_wav[t];
```

```
13      if(sound_wav[t] > 4095) sound_wav[t] = 4095;
14    }
15      }
```

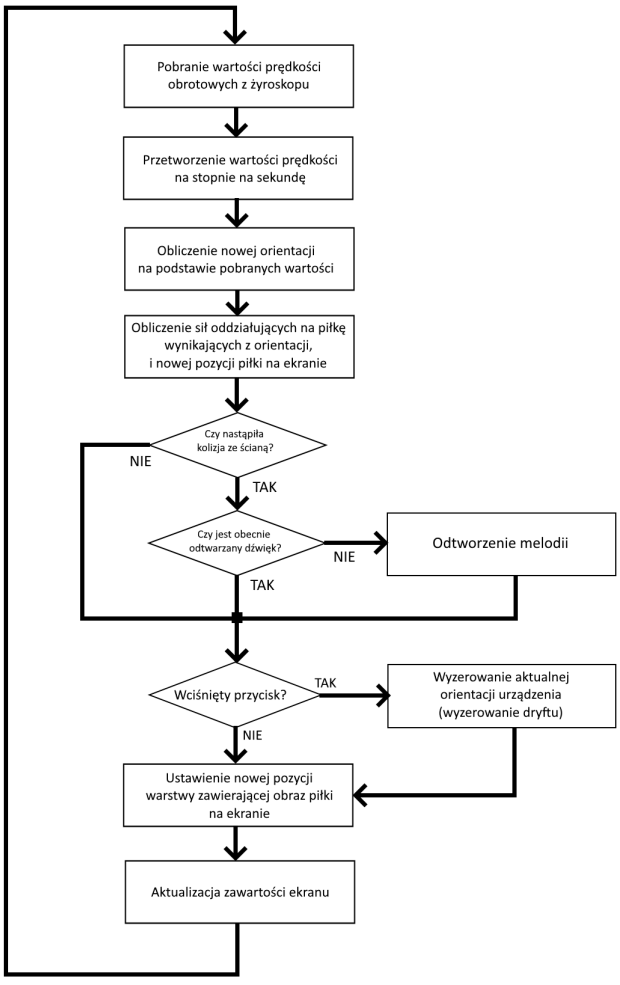
W funkcji tej dodano dodatkowe zabezpieczenia uniemożliwiające wprowadzenie wartości powyżej 4096 (największa wartość jaką można przetworzyć za pomocą przetwornika DAC równa wartości 3,3V). A także zabezpieczenie przed przekroczeniem zakresu tablicy próbek.

Aby przesyłać zapisane w tablicy próbek dane z pamięci ram do przetwornika DAC wykorzystano DMA umożliwiający bezpośrednią wymianę z pominięciem CPU co pozwala na zmniejszenie obciążenia procesora. Peryferium DAC zostało tak skonfigurowane być wyzwalanym za pomocą *Timer 2 Trigger Event* czyli za każdym razem gdy timer odliczy swój okres, wyzwolone zostanie wydarzenie powodujące wpisanie do przetwornika kolejnej próbki z tablicy. Początkowo jako adres w pamięci jest podany adres pierwszego elementu w pamięci tablicy, dzięki opcji *Increment Aaddress* DMA automatycznie przechodzi po kolejnych elementach tablicy aż dojdzie do końca (bez trybu *Circular*). Do pojedynczego odtworzenia dźwięku wystarczy wywołać funkcję:

```
1   HAL_DAC_Start_DMA(&hdac, DAC_CHANNEL_2, (uint32_t*)sound_wav, SAMPLE_NUMBER,
        DAC_ALIGN_12B_R);
```

DAC_ALIGM_12B_R to makro odnoszące się do sposobu wypełniania reejestru który jest 16 bitowy, *R* okresla że będą one wyrównane do prawej strony. [5]

5.7 Diagram głównej pętli programu



Rysunek 11: Diagram głównej pętli programu

6 Zadania niezrealizowane

Zrezygnowano z umieszczenia grafiki piłki w pamięci zewnętrzne SDRAM. Jest to spowodowane tym że nie było takiej potrzeby, obraz piłki ma wymiary zaledwie 57 na 56 pikseli. Więc zajmuje on zaledwie 6,3 KB pamięci. Wykorzystanie pamięci zewnętrznej byłoby konieczne w przypadku generowania pełnowymiarowego obrazu 240 pikseli na 320, szczególnie przy wykorzystaniu dwóch buforów na których na zmianę tworzony jest obraz.

7 Podsumowanie

Udało się poprawnie zrealizować wszystkie początkowe cele funkcjonalne. Niektóre elementy zostały rozszerzone. Wstępnie zakładano odtwarzanie sygnału o jednolitej, stałej częstotliwości, udało się to rozszerzyć do prostego zestawu narzędzi do generowania prostych melodii. Projekt pozwolił na bliższe poznanie technik generowania obrazu i grafiki od strony niskopoziomowej. W czasie styczności z płytką udało się stworzyć poboczne projekty wykorzystujące możliwości generowania i wyświetlania grafiki na wyświetlaczy LCD.

Repozytorium Github grupy: `https://github.com/chedoska/PSZ_Sterowniki_Robotow_Projekt`

Literatura

[1] G. Brown. Discovering the STM32 Microcontroller, 2016.

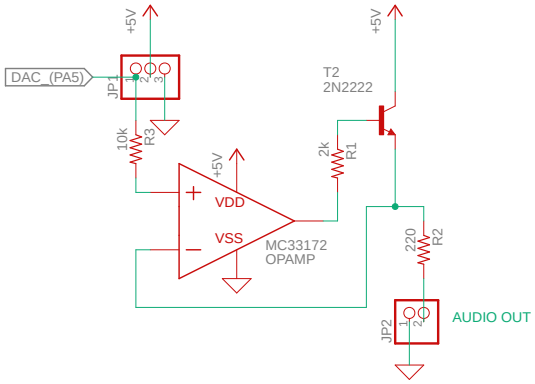
[2] L. Davidian. STM32: using the LTDC display controller, 2017.

[3] STMicroElectronics. Getting started with the STM32F429 Discovery kit , User manual, 2013.

[4] STMicroElectronics. L3GD20: 3-axis digital output gyroscope , Application note, 2014.

[5] STMicroElectronics. Audio and waveform generation using the DAC in STM32 productsl, 2020.

[6] STMicroElectronics. LCD-TFT display controller (LTDC) on STM32 MCUs, Application note, 2023.



30.05.2023 17:55 C:\Users\chedo\OneDrive\Pulpit\SEMESTR_6\Sterowniki Robotów PRO\GyroBall\untitled.sch (Sheet: 1/1)