Projekt

Sterowniki robotów

Dokumentacja

Gra typu „Flappy Bird”
FlaBi

Skład grupy:
Paweł Łyszczarz, 259258
Mateusz Strembicki, 259263

Termin: srTP19

Prowadzący:
dr inż. Wojciech Domski

7 października 2023

Spis treści

1 Opis projektu **2**

2 Konfiguracja mikrokontrolera **2**
2.1 Konfiguracja pinów . . . 4
2.2 Protokół komunikacyjny - I2C . . . 4
2.3 Multimedia . . . 4
2.4 Urządzenia zewnętrzne . . . 5
2.5 Ekran - 4"TFT LCD using MIPI DSI . . . 5
2.6 Ustawienia multimedialne . . . 6
2.6.1 DMA2D . . . 6
2.6.2 DSIHOST . . . 6
2.6.3 LTDC . . . 6

3 Opis działania programu **6**
3.1 Etap II . . . 6
3.1.1 Dodanie obrazu (obrazów) . . . 6
3.1.2 Animacja . . . 6
3.1.3 Interfejs . . . 6
3.2 Etap III . . . 7
3.2.1 Reakcja na dotyk ekranu . . . 7
3.2.2 Ograniczenia pola gry . . . 7
3.2.3 Kolizja . . . 7

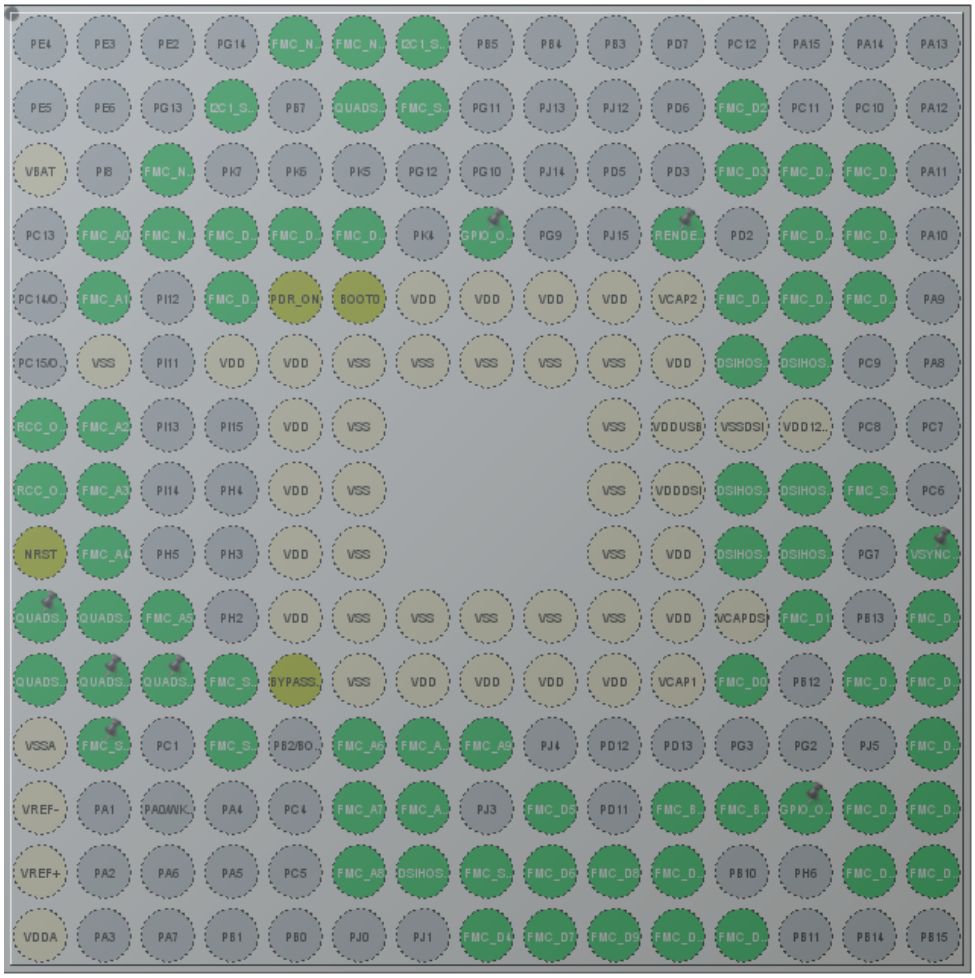
4 Podsumowanie **7**

1 Opis projektu

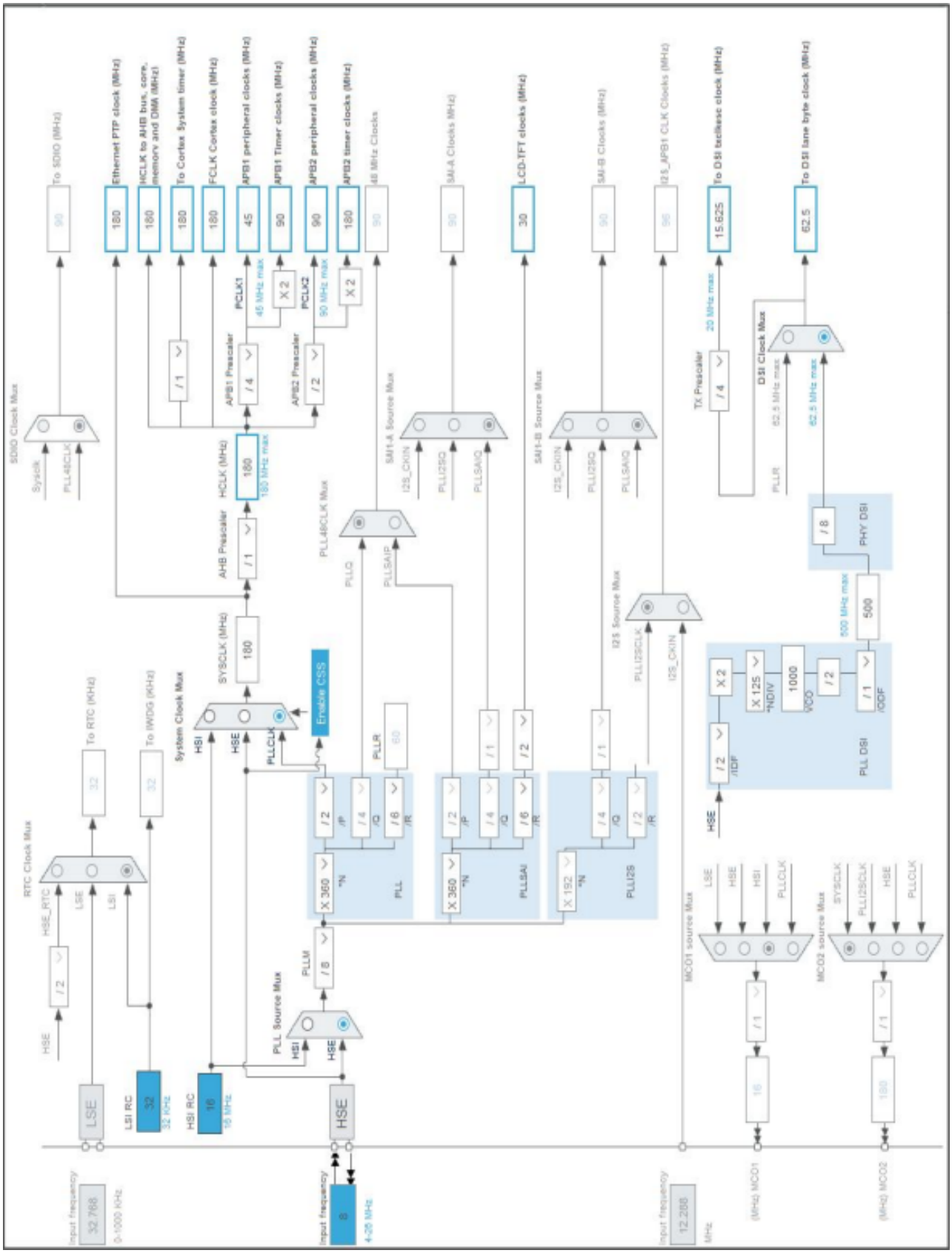
Projekt będzie się opierał o animowaną grę 2D zawierającą menu, inspirowaną popularną aplikacją „Flappy Bird". Gra zostanie zaprogramowana na płytce STM32F469-DISCOVERY, która wyposażona jest w ekran dotykowy, którym posłużymy się do obsługi gry. Zamysł gry polega na unikaniu przeszkód przez latającą postać poprzez poruszanie się w górę i opadanie, podczas gdy tło z przeszkodami będzie się stale poruszać w stronę postaci.

2 Konfiguracja mikrokontrolera

W przypadku tego projektu planujemy rozwinąć bardziej stronę software'ową gdyż cały projekt mechaniczny gry będzie nie lada wyzwaniem. Z tego też powodu nie planujemy żadnych rozszerzeń projektu w postaci zewnętrznych reakcji układu poprzez np. dodatkowe peryferia płytki czy też inne moduły, dlatego też ich ilość została zminimalizowana do zera. Stąd wszystkie ustawienia mikrokontrolera, piny, zegary itp. pozostają standardowe. Projekt nie przewiduje również z racji na brak dodatkowych modułów komunikacji poprzez różne protokoły komunikacyjne.



Rysunek 1: Konfiguracja wyjść mikrokontrolera w programie STM32CubeMX



Rysunek 2: Konfiguracja zegarów mikrokontrolera

2.1 Konfiguracja pinów

PIN	Tryb pracy	Funkcja/etykieta
PD4 *	**GPIO_Output**	**RENDER_TIME**
PG6 *	**GPIO_Output**	**VSYNC_FREQ**
PH7 *	**GPIO_Output**	—
VDDA	**POWER**	**POWER**
PK3 *	**GPIO_Output**	—
PB8	**I2C1_SCL**	—
PB9	**I2C1_SDA**	—
PF6 - PF10	**QUADSPI**	—
PJ2	**GPIO_Output**	**DSIHOST**
PH0	**RCC_OSC_IN**	—
PH1	**RCC_OSC_OUT**	—
PC0	**FMC_SDNWE**	—
PC2	**FMC_SDNE0**	—
PC3	**FMC_SDCKE0**	—
PE0	**FMC_NBL0**	—
PE1	**FMC_NBL1**	—

* – „The pin is affected with an I/O function”

2.2 Protokół komunikacyjny - I2C

Procesor do komunikacji z ekranem w naszej płytce wykorzystuje protokół komunikacyjny I2C, oto jego ustawienia parametrów:

I2C clock speed (Hz)	**400000**
Clock no stretch mode	**disabled**
Primary address length	**7-bit**
Diual addres acknowledged	**disabled**

2.3 Multimedia

Płytka korzysta z kontrolera obrazu LTDC zapewniajacego 24-bitowe połączenie RHB z ekranem, DSIHOST'a, który tłumaczony jest jako Display Serial Interface, korzystając przy tym z FMC czyli kontroli przydziału i odczytów z pamięci przy wyświetlaniu obrazów.

2.4 Urządzenia zewnętrzne

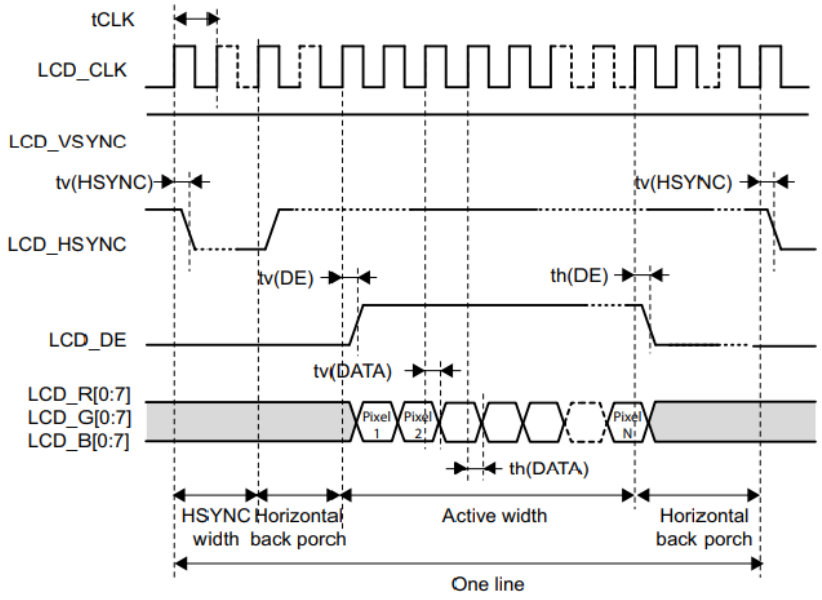
2.5 Ekran - 4"TFT LCD using MIPI DSI

[2] Jedynym jeśli można go tak nazwać - elementem zewnętrznym schematu naszego projektu jest dotykowy ekran przymocowany do naszej płytki. Ekran ten pozwala na wyświetlanie wybranych obrazów oraz interakcję z systemem poprzez dotyk.

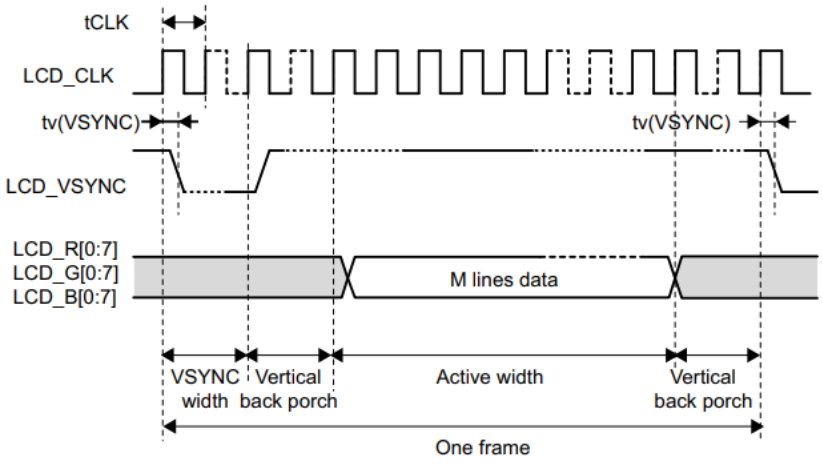
[3] Ekran wykorzystujemy do wyświetlania menu gry, które będzie jednocześnie interaktywne, przez co użytkownik będzie mógł rozpocząć grę lub ją zakończyć. Po rozpoczęciu nowej gry ekran będzie wyświetlał nowe okno, które będzie zawierało liczne animacje związane z dynamiką gry, okno to również będzie reagowało na interakcje użytkownika.

Peryferium	Adres	Bus
LCD-TFT	0x4001 6800 - 0x4001 6BFF	APB2

Tabela 1: Konfiguracja peryferium USART



Rysunek 3: LCD-TFT horizontal timing diagram



Rysunek 4: LCD-TFT vertical timing diagram

2.6 Ustawienia multimedialne

Płytka korzysta z kontrolera obrazu LTDC zapewniajacego 24-bitowe połączenie RHB z ekranem, DSIHOST'a, który tłumaczony jest jako Display Serial Interface, korzystając przy tym z FMC czyli kontroli przydziału i odczytów z pamięci przy wyświetlaniu obrazów.

2.6.1 DMA2D

Tabela 2: Tabela DMA20

Color Mode	ARGB8888

2.6.2 DSIHOST

Określenie ilości pikseli oraz częstotliwości pracy zegara LDTC .

Tabela 3: Tabela DSIHOST

Color Coding	RGB(16 bits) - DSI mode
Maximum Command Size	200 pixels
LTDC Pixel Clock	30 MHz
Time Out Clock	62500 kHz

2.6.3 LTDC

Miejsce w pamięci, gdzie przechowywane są dane dla LTDC.

Tabela 4: Tabela LTDC

Pixel Format	RGB565
Frame Buffer Start Adress	0xC0000000

3 Opis działania programu

3.1 Etap II

Podczas realizacji tego etapu projektu zrealizowane zostało zapoznanie się z działaniem wyświetlacza dotykowego oraz jego możliwościami.

3.1.1 Dodanie obrazu (obrazów)

Mianowicie udało się wyświetlić obraz na ekranie, a nawet kilka obrazów. Narzędzie wykorzystywane do tego, pomaga przy ustawianiu oraz edycji zdjęć. Dzięki temu mogliśmy podejrzeć sposób w jaki zapisywany jest obraz (obrazy) na wyświetlaczu.

3.1.2 Animacja

Kolejnym etapem było stworzenie prostej animacji tła. Realizowane jest to poprzez dodanie grafiki o dużej szerokości i odpowiednim przesuwaniem jej podczas działania programu. W kolejnym etapie na tej podstawie wyrysowany zostanie obiekt do skakania oraz przeszkody, które również będą się przesuwać [4].

3.1.3 Interfejs

Stworzony został prosty interfejs graficzny, na którym jest możliwość opuszczenia aplikacji poprzez naciśnięcie przycisku `Wyjdź`. Pojawia się również przycisk `Nowa Gra`, po wciśnięciu którego użytkownik zostaje przeniesiony do nowego okna, w którym toczyć będzie się gra. Po przegranej przewiduje się możliwość powrotu użytkownika do `menu` [1].

3.2 Etap III

Podczas realizacji tego etapu projektu zrealizowane zostało implementacja działania na dotyk ekranu oraz uproszczony algorytm sprawdzania kolizji do omijania przeszkód. Dopracowany został również interfejs użytkownika

3.2.1 Reakcja na dotyk ekranu

Na obraz tła naniesione zostało nowe zdjęcie imitujące „postać", która omijać ma przeszkody. Dla tego obiektu zaimplementowano reakcję na dotyk ekranu. Mianowicie podczas dotykania ekranu wyświetlacza wciskany jest niewidoczny przycisk, który skonfigurowany jest w taki sposób, aby wywoływał imitację ruchu obiektu do góry o konkretną wartość pikseli. Jest zawsze stała wartość. Obiekt ten ma również inną zaletę – po rozpoczęciu gry spada od cały czas w dół, dzięki czemu udało się odwzorować dosyć dobrze założenie gry. Podsumowują gracz musi odpowiednio klikać w ekran, aby dobrze sterować obiektem, aby ten ominą przeszkody i doszedł do mety.

3.2.2 Ograniczenia pola gry

Podczas wykonywania animacji skoku należało zadbać, aby nasz obiekt nie wyszedł poza obszar ekranu, więc udało się zaimplementować ograniczenie górne oraz dolne dla obiektu tak, aby nie wyleciał on poza mapę gry.

3.2.3 Kolizja

Testowane były różne podeścia co do wykrywania kolizji obiektu z przeszkodami. Jako, że operowaliśmy na obrazkach, ciężko było znaleźć dobre podejście, które w pełni oddałoby sens gry. Myśleliśmy nad zaimplementowaniem odcinka czasu, w którym wiemy, że obiekt będzie mógł mieć kolizję z przeszkodami, a następnie sprawdzać czy ona następuje. Jednak w tym momencie uzależniamy grę od kolejnego parametru, który nie jest potrzebny.
Zdecydowaliśmy więc, że algorytm kolizji polegać będzie na sprawdzeniu czy pozycja obiektu nie jest przypadkiem tam gdzie przeszkoda. Dzieje się to w oparciu o analizę przesunięcia tła z przeszkodami oraz aktualnej pozycji tego tła oraz obiektu. Mianowicie wiemy w jakiej odległości są rozmieszczone między sobą oraz jak przesuwa się obrazek, więc znamy dokładną pozycję tła. Znamy również dokładną pozycję obiektu, ponieważ od może poruszać się tylko w osi OY, więc pobieramy jedynie aktualną y–kową pozycję. Następnie sprawdzamy, czy pozycja jak i obręcz kółka nie nachodzi nachodzi na pozycję przeszkody. Jeśli tak to gra jest przerywana i użytkownik zostaje cofany do menu startowego. Jeśli nie to gra toczy się dalej.

4 Podsumowanie

Można powiedzieć, że projekt został skończony w 90% gdyż nie każde zadanie czy kazdy cel został osiągnięty w sposób zadowalający. Implementacja algorytmu kolizji zabrała bardzo dużo czasu i nie jest ona najlepszym rozwiązaniem, ale spełnia swoje zadanie. Realizacja projektu przebiegała sprawnie i bezproblemowo. Wnioski co do samej płytki są ciekawe, ponieważ według nas, peryferia jak i wyświetlacz płytki nie nadaje się do stworzenia gier. Gdy dzieje się bardzo dużo na ekranie wyświetlacza znacząco spowalnia on swoje działanie gdyż ma bardzo dużo rzeczy do obliczenia jak ciągłe śledzenie pozycji tła bądź pozycji wirtualnego obiektu do omijania przeszkód. Sądzimy natomiast, że mikrokontroler z tym wyświetlaczem idealnie sprawdzi się jako sterownik do np. smart domu, gdzie jego jedynymi funkcjami będzie proste menu oraz parę suwaków bądź przycisków. Znacznie lepiej zniesie on sterowanie prostym menu i paroma opcjami niż sterowaniem i obliczaniem złożonej gry, która w zamyśle wydaje się być bardzo prosta w realizacji.

Link do repozytorium GitHub: `https://github.com/Strembicki-Mateusz/FlaBi_Game`

Literatura

[1] T. Gray, T. Gray. Display basics. *Projected Capacitive Touch: A Practical Guide for Engineers*, strony 101–115, 2019.

[2] G. İŞNAS, N. ŞENYER. Comparison of touchgfx and lvgl embedded hardware gui libraries. *Gazi University Journal of Science Part C: Design and Technology*, 9(3):373–384, 2021.

[3] G.-w. Shin, Y.-h. Kim, D.-G. Jeong, Y.-h. Lee. Hardware design of mipi c-phy compatible csi-2/dsi rx interface. *JOURNAL OF PLATFORM TECHNOLOGY*, 5(4):16–26, 2017.

[4] A. Zharikov, D. Kozin, P. Nekrasov. Design and implementation of home assistant and touchgfx interaction based on stm32 microcontroller. *2022 Moscow Workshop on Electronic and Networking Technologies (MWENT)*, strony 1–3. IEEE, 2022.