Sprawozdanie

Rodzaj zajęć

Projekt

Przedmiot

Systemy Mikroprocesorowe i Wbudowane

Rok akademicki	Miasto	Kierunek	Semestr	Prowadzący	Grupa	Sekcja
2025/2026	G	INF	5	GB	5	9

Temat projektu

Talos 7 – Programowalna makroklawiatura

Skład sekcji		
	Imię i nazwisko	Uwagi
1	Szymon Wilczek	
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		

Temat projektu i opis założeń

Temat

Projekt nosi nazwę Talos 7. Jest to programowalna makroklawiatura (macro pad), która pełni funkcję hybrydowego urządzenia wejściowego, łączącego cechy klawiatury, myszy, kontrolera MIDI oraz urządzenia Serial (CDC).

Założenia funkcjonalne

Głównym celem projektu było stworzenie urządzenia typu *Driverless* – niewymagającego instalacji dedykowanego oprogramowania na komputerze użytkownika (stanąłem opozycją do rozwiązań komercyjnych jak Razer czy Corsair). Cała konfiguracja oraz logika działania przechowywana jest w pamięci mikrokontrolera. Kluczowymi założeniami, których się trzymałem podczas tworzenia projektu były:

- **Wieloplatformowość**: Działanie na systemach Windows, Linux, macOS bez dodatkowych (jakichkolwiek) sterowników
- **Konfiguracja przez przeglądarkę**: Wykorzystanie Web Serial API do zarządzania ustawieniami bezpośrednio z poziomu przeglądarki bazowanej na Chromium (Chrome, Opera, Edge).
- **Interfejs użytkownika:** Wyświetlacz OLED 128x64 prezentujący aktualną warstwę oraz status urządzenia.
- **Obsługa wielu protokołów**
    1. **USB HID** (Klawiatura, Mysz)
    2. **USB MIDI** (Kontrola DAW – używana przykładowo w OBS Studio)
    3. **USB CDC** (Wirtualny port szeregowy do konfiguracji i skryptowania)
- **Mechanizmy automatyzacji:** Możliwość wykonywania sekwencji klawiszy, wpisywania tekstu oraz uruchamiania skryptów systemowych.

Analiza zadania i dobór elementów

Analiza wymagań

Do realizacji projektu potrzebny był mi mikrokontroler obsługujący natywnie USB 2.0, posiadający wystarczającą ilość pamięci Flash na przechowywanie konfiguracji makr oraz odpowiednią liczbę pinów GPIO do obsługi przycisków i wyświetlacza.

Wybór mikrokontrolera: Raspberry Pi Pico (RP2040)

Wybrałem go ze względu na:

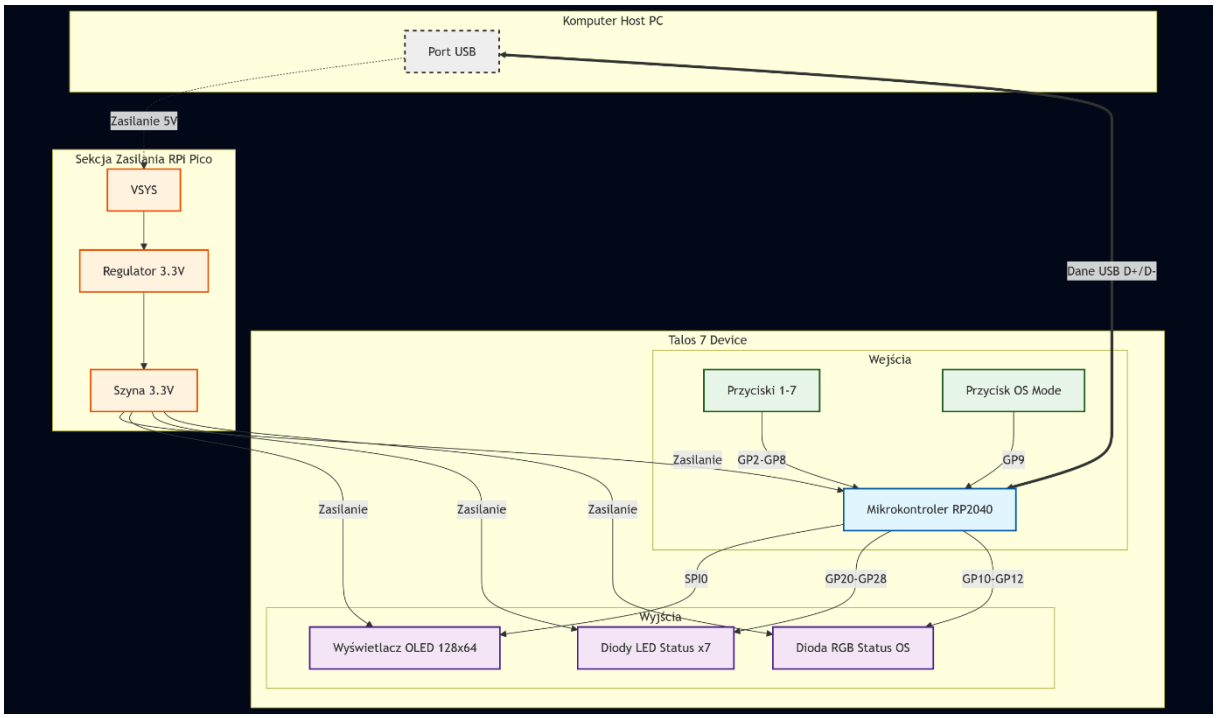
- Dostępność od ręki (nie wymagał on zakupu, posiadałem go w domu)
- Natywną obsługę USB: Niezbędną do implementacji klas HID/MIDI/CDC (TinyUSB)
- Duża ilość pamięci Flash (2MB): Wystarczająca na firmware.

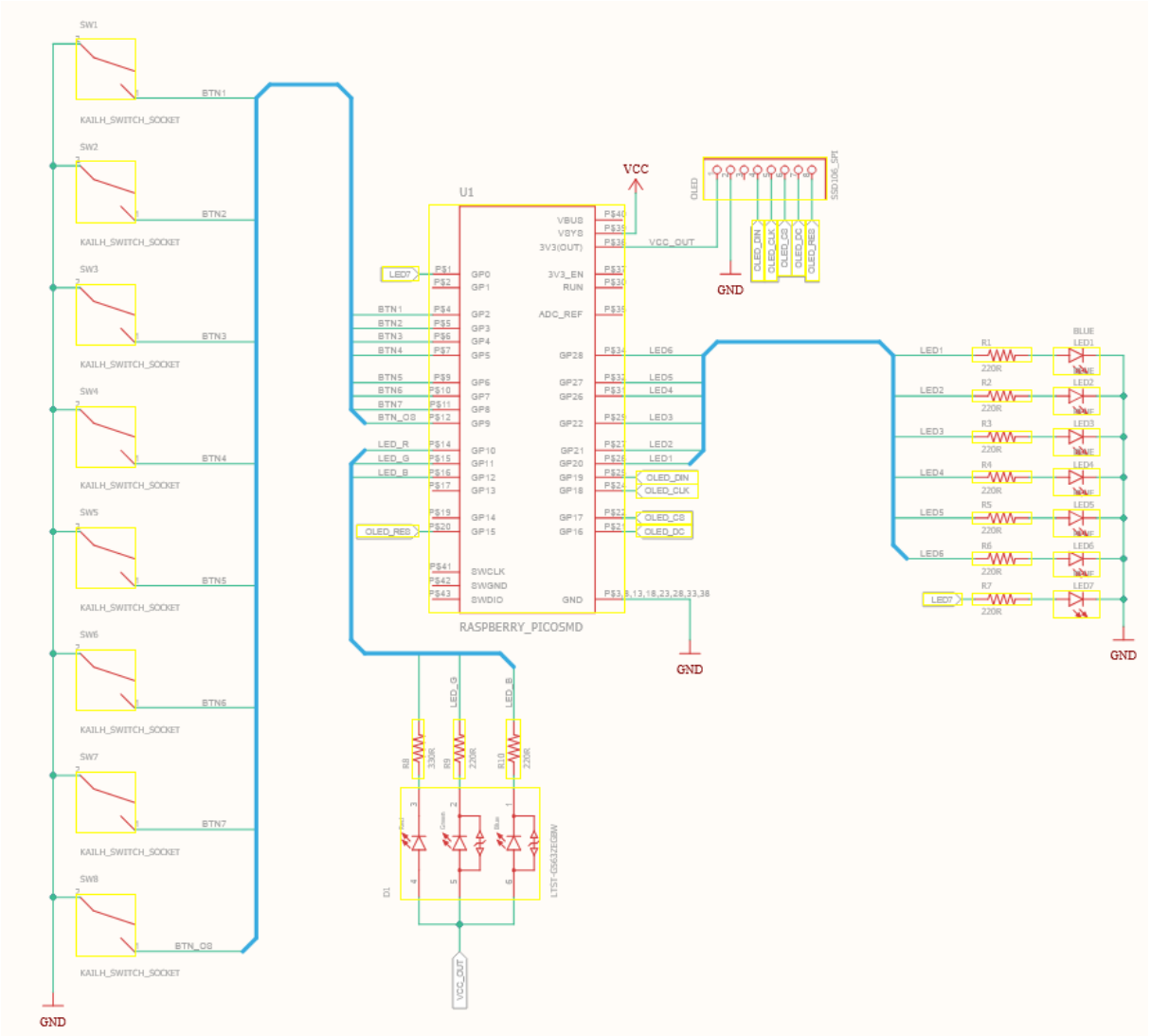
Wybór elementów peryferyjnych

1. **Przyciski (MX Switch):** Zastosowałem mechaniczne przełączniki typu Cherry MX ze względu na trwałość, komfort użytkowania i szeroką dostępność.
2. **Wyświetlacz OLED SSD1306** (0,96” – SPI): Popularny, tani wyświetlacz z dobrą biblioteką obsługi. Wybrałem interfejs SPI ze względu na wyższą prędkość odświeżania w porównaniu do I2C, co jest istotne przy animacjach wygaszania ekranu w moim projekcie oraz zaimplementowanej przerywnikowo grze.
3. **Diody LED (0805)**: Sygnalizacja aktywności danego przycisku oraz warstwy.
4. **Dioda RGB (Wspólna Anoda)**: Do sygnalizacji trybu pracy systemu operacyjnego (Zielony – Linux, Niebieski – Windows, Cyjanowy (Niebieski + Zielony) – macOS).

Specyfikacja wewnętrzna urządzenia

Schemat blokowy i ideowy

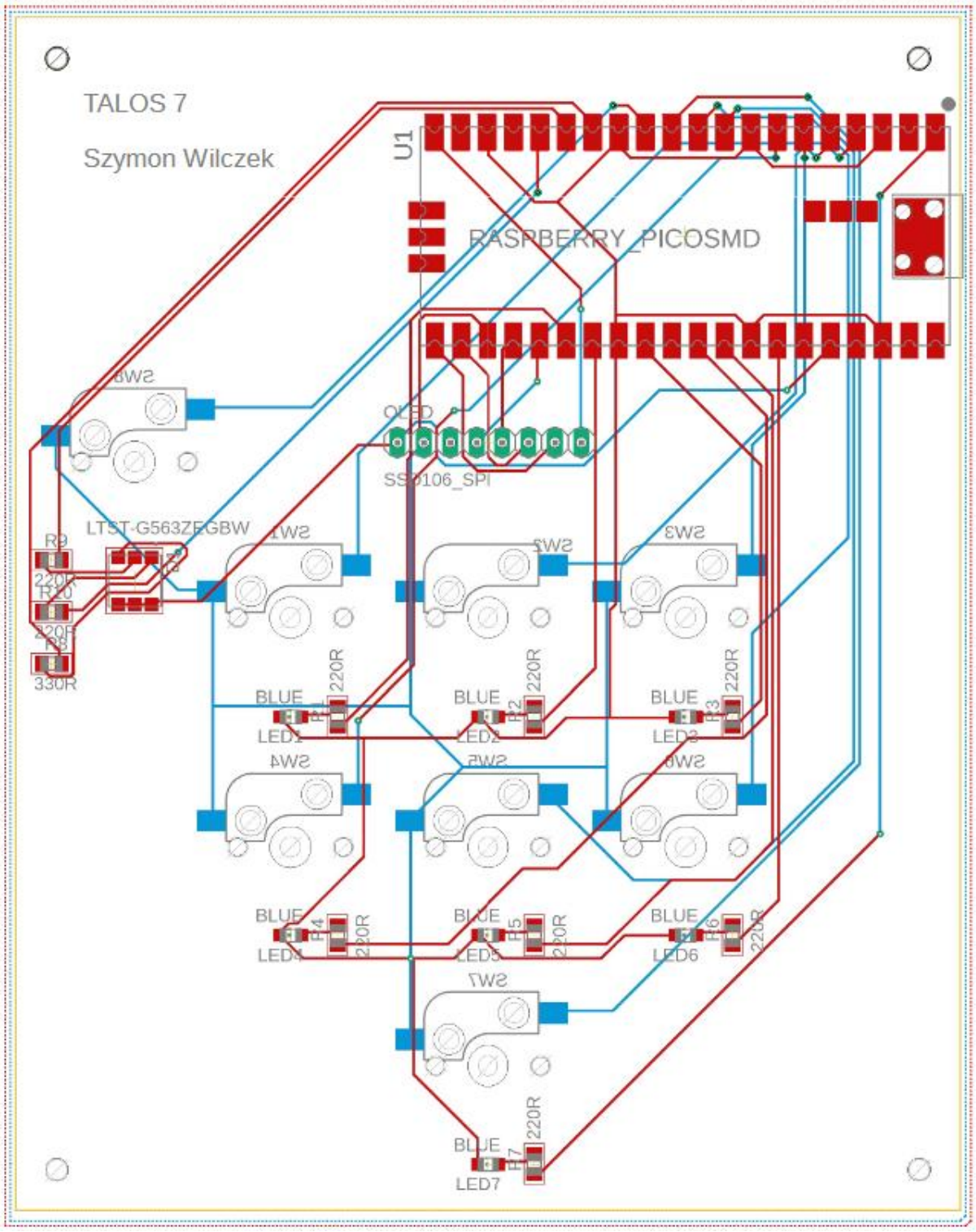




Opis funkcji bloków

- **Blok Mikrokontrolera (RP2040):** Serce układu, zarządza komunikacją USB i logiką makr.
- **Blok Wejścia (Przyciski):** 7 przycisków podłączonych do GP2-GP8 z programowym odkłócaniem (debounce) realizowanym w firmware.
- **Blok Wyjścia (OLED):** Wyświetlacz podłączony przez SPI0 (GP16-GP19)
- **Blok Zasilania:** Układ zasilany bezpośrednio z portu USB z wewnętrzną stabilizacją 3.3V na płytce Pico.

Schemat montażowy i PCB



Algorytm oprogramowania

Główna pętla programu realizuje cykliczne przetwarzania zadań:

- **Inicjalizacja** (Timery, USB, GPIO, OLED).
- **Pętla nieskończona**:
  - **Reset licznika watchdog** (timeout 8s).
  - **Obsługa zdarzeń USB** (TinyUSB).
  - **Obsługa komend** przychodzących z aplikacji webowej.
  - **Zarządzanie wygaszaczem** ekranu (animacje).
  - **Sprawdzenie przycisku** zmiany OS (GP9).
  - **Skanowanie przycisków**: Jeśli przycisk wciśnięty -> wykonanie makra.

Opis zmiennych

Z racji faktu, że projekt posiada bardzo dużo zmiennych, ponieważ firmware jest bardzo modularny, poniżej załączam opis jedynie ważniejszych zmiennych globalnych:

- `g_detected_platform`: Przechowuje aktualnie wybrany system operacyjny
- `button_states[NUM_BUTTONS]`: Tablica aktualnych stanów logicznych przycisków.
- `config_data_t`: Struktura przechowująca całą konfigurację makr (warstwy, przypisania przycisków, preferencje użytkownika – nazewnictwa przycisków, warstw, przypisanie ikonek)

Opis funkcji procedur

Analogicznie do zmiennych, załączyłem jedynie najważniejsze procedury w systemie:

- `main()`: Punkt wejścia, inicjalizacja i główna pętla
- `execute_macro(layer, button)`: Funkcja dispatchera („rozdzielacza”), która na podstawie typu makra (`KEY_PRESS`, `TEXT`, `SCRIPT`, `MIDI`, itd.) wywołuje odpowiednią funkcję wykonawczą.
- `hardware_init()`: Konfiguruje piny GPIO, ustawia kierunki (IN/OUT) i inicjalizuje peryferia (SPI dla OLED).
- `buttons_timer_callback()`: Funkcja wywoływana cyklicznie przez timer, odpowiedzialna za odczyt i filtrację (debouncing) stanu przycisków.

Opis interakcji oprogramowania z układem

Oprogramowanie cyklicznie odpytuje stan pinów wejściowych (polling). Po wykryciu zmiany stanu, system sprawdza konfigurację dla danego przycisku w aktywnej warstwie i wysyła odpowiedni raport USB (HID Report lub MIDI Packet) do komputera użytkownika. Oprogramowanie steruje również fizycznie diodami LED, włączając diodę odpowiadającą aktywnej warstwie oraz podczas naciśnięcia przycisku odpowiednio nią pojedynczo miga – sygnalizując kliknięcie.

Specyfikacja zewnętrzną

Elementy sterujące

Urządzenie posiada 7 głównych przycisków mechanicznych oraz 1 przycisk systemowy do zmiany OS.

- **Przyciski 1-7:** Wyzwalają zaprogramowane makra.
- **Przycisk OS (GP9)**: Zmienia tryb emulacji skrótów klawiszowych oraz wpisywania znaków typu Unicode (np. CTRL vs CMD)

Elementy wykonawcze

- **Wyświetlacz OLED:** Pokazuje logo, nazwę aktualnej warstwy, ikony oraz animację wygaszacza.
- **Diody LED** (Czerwone): Sygnalizują, która warstwa jest aktywna, bądź który przycisk został wciśnięty.
- **Dioda RGB**: Kolor informacje o wybranym systemie:
    a. Zielony: Linux
    b. Niebieski: Windows
    c. Cyjan: macOS

Reakcja na zdarzenia

- **Podłączenie USB**: Wyświetlenie logo startowego, inicjalizacja USB CDC.
- **Wciśnięcie przycisku**: Natychmiastowe wysłanie kodu klawisza / myszy.
- **Brak aktywności**: Po określonym czasie włącza się wygaszacz ekranu OLED.

Skrócona instrukcja obsługi

1. Podłącz urządzenie do portu USB.
2. Wejdź na stronę konfiguratora (https://talos-7.vercel.app)
3. Kliknij „*Connect*” i wybierz urządzenie „*Talos 7*”.
4. Przypisz akcje do przycisków w interfejsie graficznym.
5. Kliknij „*Save*”, aby zapisać konfigurację do pamięci urządzenia.

Opis montażu i uruchomienia

Montaż

Układ zmontowałem na zaprojektowanej płytce PCB. Diody, rezystory, wyświetlacz OLED oraz Raspberry Pi Pico zamontowano po stronie górnej, gniazda dla switchy mechanicznych (Kailh) zamontowano po stronie dolnej. Obudowa została wydrukowana w technologii 3D.

Problemy i rozwiązania

Podczas uruchamiania napotkano następujące wyzwania:

- **Problem:** Oderwanie jednego piu (kanał Czerwony) podczas odlutowywania diody RGB.
  - **Rozwiązanie:** W wyniku uszkodzenia mechanicznego (oderwało się pole lutownicze), urządzenie nie wyświetla koloru czerwonego. Początkowo dioda miała mieć następującą symbolikę: Pomarańczowy – Linux, Niebieski – Windows, Różowy – macOS. Naprawiłem to zamieniając kanał Czerwony na korzystanie z dwóch kanałów osobno oraz połączenie ich w kolor Cyjanowy.
- **Problem:** Problemy z rozpoznawaniem urządzenia przez system Windows (Composite Parent Device).
  - **Rozwiązanie:** System Windows miał trudności z poprawną emulacją urządzenia złożonego (Composite Device), co powodowało błędy sterownika. Wymagało to wielokrotnych prób i diagnostyki deskryptorów USB, aż do uzyskania stabilnej konfiguracji, którą system poprawnie identyfikuje.
- **Problem:** Błędy komunikacji USB CDC na systemie Linux.
  - **Rozwiązanie:** Konieczność dodania reguł `udev`, aby użytkownik bez uprawnień `root` miał dostęp do urządzenia i poprawnie mógł je skonfigurować przez sieciowy interfejs.

Testy

Przeprowadzono testy poprawności działania:

- Test wszystkich klawiszy w edytorze tekstu
- Test wysyłania komunikatów MIDI w programie DAW.
- Test konfiguracji przez Web Serial API (Chrome).
- Test stabilności pracy (długotrwałe działanie wygaszacza ekranu).

Wnioski

Opracowany układ spełnia wszystkie założenia projektowe. Udało się stworzyć w pełni funkcjonalne urządzenie HID, które jest konfigurowalne bez instalowania sterowników. Platforma RP2040 okazała się bardziej wydajna i elastyczna niż przypuszczałem początkowo. Projekt ma duży potencjał rozwojowy, np. o dodanie obsługi enkoderów obrotowych lub komunikacji Bluetooth (Pico W).

Literatura

Dokumentacja Raspberry Pi Pico C SDK

Specyfikacja standardu USB HID (Device Class Definition for Human Interface Devices)

Dokumentacja biblioteki TinyUSB

Repozytorium projektu