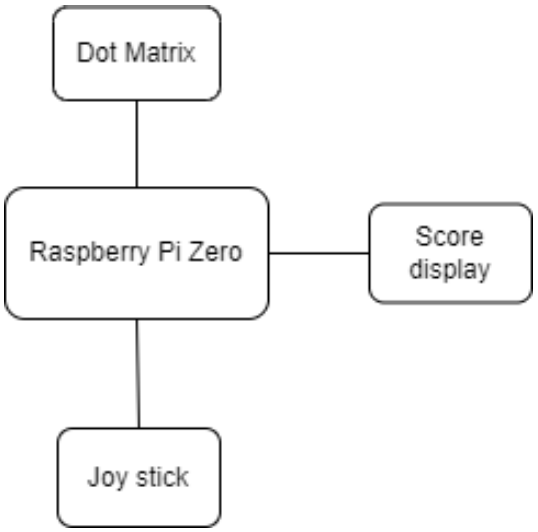
[Z7] KARTA PROJEKTU

Kacper Woźniak (151118)	**GRUPA L1**	**L1**	
Józef Godlewski (151847)		**L1**	
	Prowadzący zajęcia:		**dr inż. Przemysław Zakrzewski**

Raspberry Pi Tetris

CEL PROJEKTU	Implementacja podstawowych zasad gry Tetris, takich jak poruszanie klockami, obracanie ich i eliminowanie pełnych linii. Wykorzystanie możliwości programowania i sterowania GPIO (General Purpose Input/Output) na Raspberry Pi do kontroli klocków w grze. Implementacja interfejsu użytkownika na płycie LEDowej do wyświetlania planszy gry i wyświetlaczu lcd z wynikiem. Oprogramowanie obsługi przycisków, aby umożliwić graczowi sterowanie klockami w grze.

SCHEMAT POGLĄDOWY (UPROSZCZONY)



WYKORZYSTANA PLATFORMA SPRZĘTOWA, ELEMENTY POMIAROWE I WYKONAWCZE	*Raspberry pi zero, led dot matrix 8 x 32, 5 przycisków, matryca lcd*

1. Cel i zakres projektu.

Celem projektu było stworzenie funkcjonalnej implementacji gry Tetris, wykorzystując do tego platformę Raspberry Pi oraz różnorodne komponenty sprzętowe. Projekt obejmuje zintegrowane rozwiązania sprzętowe, takie jak matryca LED, przyciski, wyświetlacz LCD, a także programistyczne - napisanie kodu sterującego oraz obsługującego logikę gry.

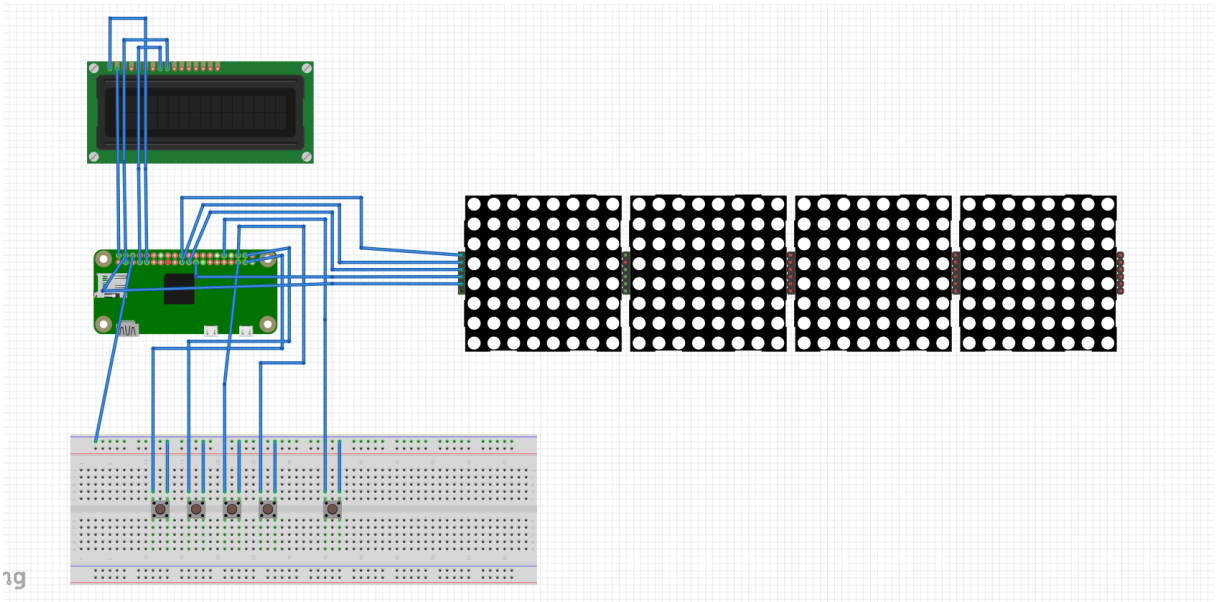
Zakres Sprzętowy:

Raspberry Pi Zero to główna jednostka obliczeniowa, kontrolująca całą logikę gry i obsługująca interakcje użytkownika. Wykorzystujemy matrycę LED 8x32 do wizualizacji planszy gry, gdzie każdy piksel reprezentuje pojedynczy element planszy. Przyciski (LEFT, RIGHT, DOWN, ROTATE, RESET) zapewniają interakcję użytkownika z grą, umożliwiające przesuwanie, obracanie i resetowanie gry. Wyświetlacz LCD 16x2 służy do wyświetlania informacji o bieżącym wyniku oraz najlepszym wyniku.

Zakres Programistyczny:

Zaimplementowaliśmy kod w języku Python, który steruje logiką gry, obsługą przycisków, wyświetlaczem LED oraz LCD. Do obsługi matrycy LED wykorzystaliśmy bibliotekę luma, umożliwiającą łatwe zarządzanie pikselami na wyświetlaczu. Do obsługi LCD wykorzystaliśmy bibliotekę RPLCD. Skonfigurowaliśmy piny GPIO na Raspberry Pi do obsługi przycisków, co umożliwia interakcję użytkownika. Zaimplementowaliśmy mechanizm zapisu i odczytu najlepszego wyniku z pliku, zapewniająca trwałość wyników między sesjami gry. Zaimplementowaliśmy również obsługę przerwań (np. KeyboardInterrupt) dla poprawnego zakończenia programu oraz zwolnienia zasobów GPIO i LCD.

2. Schemat.



Piny dla przycisków:

- LEFT_PIN - Pin 26
- RIGHT_PIN - Pin 20
- DOWN_PIN - Pin 19
- ROTATE_PIN - Pin 16
- RESET_PIN - Pin 12

3. Założenia projektowe a ich realizacja.

Jedyną rzeczą, której nie udało się zrealizować było użycie joysticka do sterowania klockami. Problem ten pojawił się, dlatego że kupiony przez nas joystick nie miał wyjścia na dane, więc nie mogliśmy podłączyć go do Raspberry Pi. W miejsce joysticka postanowiliśmy użyć mniej efektowne, ale równie efektywne przyciski. Możliwością rozwoju projektu jest zapisywanie w bazie danych najwyższego wyniku, zamiast zapisu w pliku.

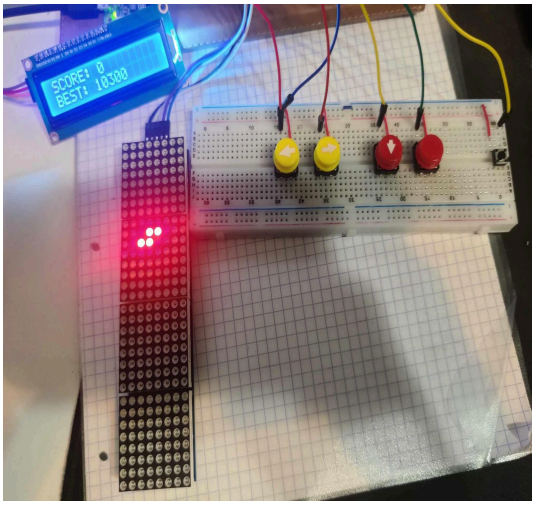
4. Listing kodu.

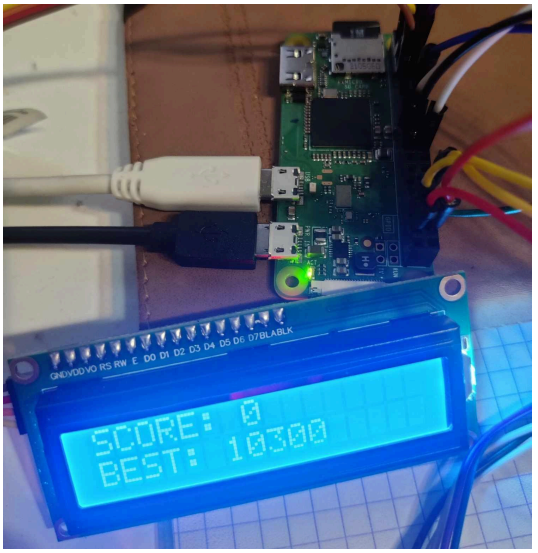
Najważniejsze fragmenty kodu (logika gry jest pominięta):

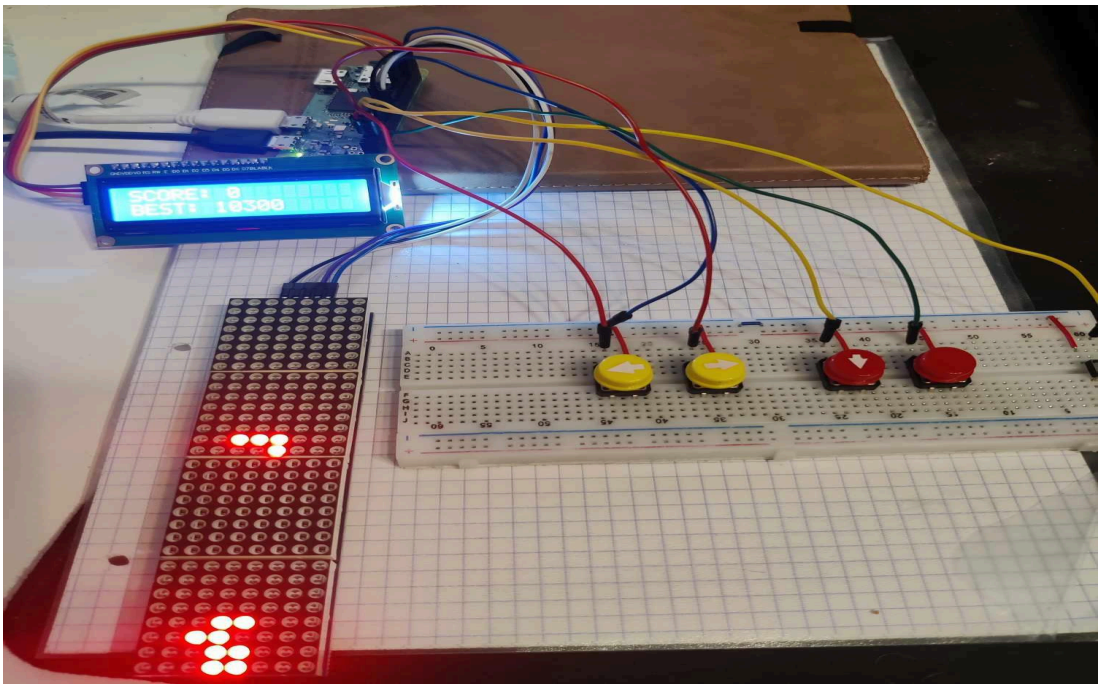
```
# GPIO Piny dla przycisków
LEFT_PIN = 26
RIGHT_PIN = 20
DOWN_PIN = 19
ROTATE_PIN = 16
RESET_PIN = 12

# Inicjalizacja GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setup(LEFT_PIN, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(RIGHT_PIN, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(DOWN_PIN, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(ROTATE_PIN, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(RESET_PIN, GPIO.IN, pull_up_down=GPIO.PUD_UP)

# Konfiguracja wyświetlacza LED
serial = spi(port=0, device=0, gpio=noop())
device = max7219(serial, cascaded=4, block_orientation=-90, rotate=3)

img = Image.new("1", (COLS, ROWS), "black")
pixelMap = img.load()



# Klasa Delays do obsługi opóźnień
class Delays:
    button = dict()
    def __init__(self):
        self.button[LEFT_PIN] = time()
        self.button[RIGHT_PIN] = time()
        self.button[DOWN_PIN] = time()
```

```
        self.button[ROTATE_PIN] = time()
        self.last = time()



# Klasa LCD do obsługi wyświetlacza LCD
class LCD:
    def draw_score(self, score, high_score):
        self.lcd.clear()
        self.lcd.write_string('SCORE: {}\n\rBEST: {}'.format(score,
high_score))

    def draw_game_over(self, score):
        self.lcd.clear()
        self.lcd.write_string("SCORE: {}\n\r   GAME OVER!
".format(score))

    def __init__(self, high_score):
        self.lcd = CharLCD(i2c_expander='PCF8574', address=0x27, port=1,
cols=16, rows=2, dotsize=8)
        self.draw_score(0, high_score)



# Dodanie obsługi przerwań dla przycisków
GPIO.add_event_detect(ROTATE_PIN, GPIO.FALLING, callback=lambda x:
tetris.move_block(ROTATE_PIN), bouncetime=200)
GPIO.add_event_detect(RESET_PIN, GPIO.FALLING, callback=lambda x: reset(),
bouncetime=1000)

try:
    while True:
        while tetris.isPlaying:
            if GPIO.input(LEFT_PIN) == GPIO.LOW and time() -
delays.button[LEFT_PIN] > 0.15:
                delays.button[LEFT_PIN] = time()
                tetris.move_block(LEFT_PIN)
            elif GPIO.input(RIGHT_PIN) == GPIO.LOW and time() -
delays.button[RIGHT_PIN] > 0.15:
                delays.button[RIGHT_PIN] = time()
                tetris.move_block(RIGHT_PIN)
            if GPIO.input(DOWN_PIN) == GPIO.LOW and time() -
delays.button[DOWN_PIN] > 0.05:
                delays.button[DOWN_PIN] = time()
                tetris.move_block(DOWN_PIN)

            if time() - delays.last > tetris.interval:
                tetris.move_block(DOWN_PIN)
                delays.last = time()
            tetris.draw_board()
        else:
            sleep(1)
except KeyboardInterrupt:
    GPIO.cleanup()
    tetris.lcd.lcd.clear()
```

5. Zdjęcia zrealizowanego układu.







6. **Zrzuty ekranu aplikacji (jeśli występują).**

ZDJĘCIE NR 1
(7 cm x 7 cm)

ZDJĘCIE NR 2
(7 cm x 7 cm)

ZDJĘCIE NR 3
(9 cm x 15 cm)

7. **Podsumowanie i wnioski.**

Podsumowanie:

Projekt Tetris na platformie Raspberry Pi był fascynującym doświadczeniem, łączącym programowanie, obsługę GPIO, interfejs użytkownika na wyświetlaczu LED oraz integrację z wyświetlaczem LCD. Projekt pozwolił na praktyczne zastosowanie umiejętności programowania

w języku Python na platformie Raspberry Pi. Wykorzystaliśmy różnorodne biblioteki, takie jak RPi.GPIO, luma.led_matrix, PIL, czy RPLCD, co przyczyniło się do rozwoju naszych umiejętności programistycznych. Wykorzystanie GPIO do obsługi przycisków umożliwiło interaktywną kontrolę gry Tetris. Zastosowanie przerwań dla przycisków pozwoliło na płynne i responsywne sterowanie klockami w grze. Połączenie wyświetlacza LED do przedstawiania planszy gry oraz wyświetlacza LCD do prezentacji wyników i informacji o stanie gry było ważnym aspektem projektu. To doświadczenie rozszerzyło nasze umiejętności w obszarze interfejsów użytkownika. Projekt wymagał dostosowania do dostępnych zasobów sprzętowych, takich jak zastąpienie joysticka przyciskami ze względu na problemy z podłączeniem joysticka do Raspberry Pi.

Wnioski:
Realizacja projektu Tetris na Raspberry Pi dostarczyła nam nie tylko praktycznej wiedzy z obszaru programowania i obsługi sprzętu, ale również wzbogaciła naszą zdolność do rozwiązywania problemów. Projekt ten pozwolił nam lepiej zrozumieć interakcję między oprogramowaniem a sprzętem, a także umożliwił praktyczne zastosowanie różnorodnych umiejętności programistycznych w jednym projekcie. Zdobycie doświadczenia w obszarze integracji różnych modułów oraz optymalizacji kodu było cennym aspektem nauki. Wnioski z tego projektu mogą posłużyć jako solidna baza do dalszego rozwijania umiejętności w dziedzinie programowania na platformie Raspberry Pi oraz tworzenia interaktywnych projektów z użyciem GPIO.