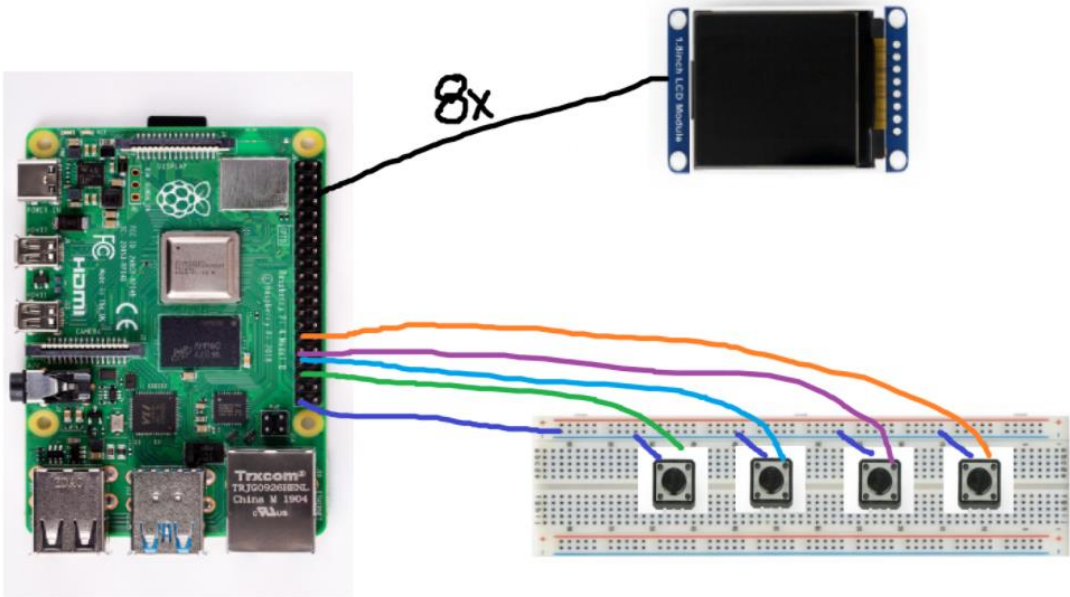


[Z7] KARTA PROJEKTU

Grupa laboratoryjna: L13 Podgrupa: 2		dr inż. Ariel Antonowicz
	Wojciech Szott (155 876)	
	Paweł Muryn (156 011)	
Prowadzący zajęcia:		
<i>StarShooter</i>		
CEL PROJEKTU	<p>Stworzenie gry polegającej na unikaniu pocisków i zestrzeliwaniu przeciwników.</p> <p>Dodatki: power-upy (laser, bomba, dodatkowe HP)</p>	
SCHEMAT POGLĄDOWY (UPROSZCZONY)		
		

WYKORZYSTANA PLATFORMA SPRZĘTOWA, ELEMENTY POMIAROWE I WYKONAWCZE	<p><i>Platforma sprzętowa:</i> Raspberry Pi</p> <p>Elementy:</p> <p>Waveshare Wyświetlacz LCD TFT kolorowy 1,8" 128x160px SPI</p> <p>Płytką Stykowa</p> <p>4 Przyciski</p>
--	--

1. Cel i zakres projektu.

Cel:

Celem projektu jest stworzenie gry inspirowanej kultową grą arkadową *Space Invaders*. Gracz steruje statkiem kosmicznym za pomocą fizycznych przycisków i ma za zadanie zestrzeliwać wrogie statki, jednocześnie unikając ich pocisków. Wrogowie przemieszczają się w trzech rzędach, powoli zbliżając się ku dołowi ekranu.

Rozgrywka kończy się w przypadku utraty wszystkich żyć przez gracza lub dotarcia przeciwników na sam dół ekranu. Zdobywanie punktów odbywa się poprzez eliminację przeciwników, z których czasami wypadają bonusy – ich podniesienie zapewnia specjalne umiejętności.

Dostępne są 4 przyciski:

Lewo – ruch w lewo

Prawo – ruch w prawo

Strzał - wykonanie strzału

Special – wykonanie ruchu specjalnego (jeśli jest dostępny) oraz reset przy końcu gry

Zakres sprzętowy:

Projekt opiera się na platformie Raspberry Pi, która pełni funkcję centralnej jednostki sterującej. W skład sprzętu wchodzi:

-Wyświetlacz LCD TFC 1.8 inch – odpowiedzialny za prezentację obrazu gry.

-Fizyczne przyciski – służące do sterowania statkiem i aktywowania dostępnych funkcji.

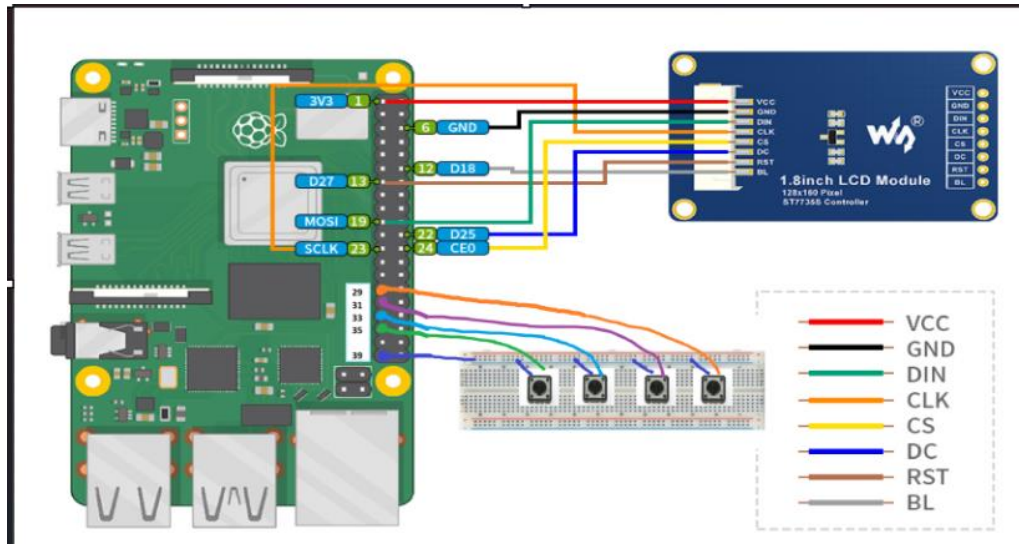
Raspberry Pi odczytuje sygnały z przycisków, przetwarza dane gry i aktualizuje wyświetlany obraz na ekranie LCD.

Zakres programistyczny:

Projekt składa się z bibliotek potrzebnych do obsługi wyświetlacza oraz pojedynczego programu w języku python, który steruje GUI gry

2. Schemat.

Schemat połączeniowy



Schemat ideowy

Wyświetlane obiekty:

- Postać gracza
- Statki przeciwników
- Pociski
- Bonusy
- Akcje Specjalne
- Teksty: Game Over (na koniec gry), Score (wynik punktowy), HP (punkty życia), Bonus (aktywny bonus)

Bloki kodu:

Inicjalizacja – inicjalizacja wyświetlacza i przycisków

Funkcje gracza – akcje gracza: poruszanie się strzelanie akcja specjalna

Update obiektów - ruch obiektów oraz wykrywanie kolizji

Rysowanie – Rysowanie wszystkich obiektów i tekstów

Update - główna pętla programu obsługująca logikę

3. Założenia projektowe a ich realizacja.

Spełnione:

- Funkcjonalna gra
- Różne poziomy
- 3 akcje specjalne
- Wyświetlanie licznika Score i HP
- Wyświetlanie informacji o końcu gry (Game Over)

Nie spełnione:

- *Korzystanie z BeagleBoneBlack (nie znaleźliśmy biblioteki wspierającej wyświetlacz)*
- *Wgranie obrazu jako grafiki (funkcja wyświetlająca obraz nie przyjmowała koordynatów)*



Dalszy rozwój:

- *Ekran startowy z wyborem trudności*
- *Większa różnorodność przeciwników i bonusów*
- *Dodanie dźwięku*
- *Zmiana grafiki i dodanie animacji*

4. Listing kodu.

Inicjalizacja

#biblioteki

```

import random
import time
import os
import sys
import logging
from random import randint
import spidev as SPI
sys.path.append("../")
from PIL import Image, ImageDraw, ImageFont
from lib import LCD_1inch8
import RPi.GPIO as GPIO

# region LCD INIT

# Konfiguracja GPIO
GPIO.cleanup()
GPIO.setmode(GPIO.BCM)

# Konfiguracja Pinów
PIN_RIGHT = 5
PIN_LEFT = 6
PIN_SHOOT = 13
PIN_SPECIAL = 19

# Ustawienie pinów jako wejścia
GPIO.setup(PIN_LEFT, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(PIN_RIGHT, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(PIN_SHOOT, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(PIN_SPECIAL, GPIO.IN, pull_up_down=GPIO.PUD_UP)

# Przypisanie zdarzeń do pinów (występuje po zdefiniowaniu funkcji w callback)
GPIO.add_event_detect(PIN_LEFT, GPIO.FALLING, callback=move_left, bouncetime=100)
GPIO.add_event_detect(PIN_RIGHT, GPIO.FALLING, callback=move_right, bouncetime=100)
GPIO.add_event_detect(PIN_SHOOT, GPIO.FALLING, callback=shoot, bouncetime=100)
GPIO.add_event_detect(PIN_SPECIAL, GPIO.FALLING, callback=special, bouncetime=100)

# Inicjalizacja wyświetlacza
lcd = LCD_1inch8.LCD_1inch8()
lcd.Init()
lcd.clear()
lcd.bl_DutyCycle(50)
image = Image.new("RGB", (lcd.width, lcd.height), "WHITE") #podstawowe białe tło
draw = ImageDraw.Draw(image)
font = ImageFont.load_default() #podstawowa czcionka

```

Rysowanie

```
def draw_game():
    # tło
    draw.rectangle((0, 0, lcd.width, lcd.height), fill="WHITE")
    #gracz
    draw.rectangle((player_x, player_y, player_x + player_width, player_y + player_height), fill="blue")
    # pociski gracza
    for bullet in bullets:
        draw.rectangle((bullet[0], bullet[1], bullet[0] + bullet_width, bullet[1] + bullet_height), fill="red")
    # przeciwnicy
    for enemy in enemies:
        draw_enemies(enemy)
    # pociski przeciwników
    for bullet in enemy_bullets:
        draw.rectangle((bullet[0], bullet[1], bullet[0] + bullet_width, bullet[1] + bullet_height),
fill="orange")
    # spadające bonusy
    for bonus in drops:
        draw.rectangle((bonus[0], bonus[1], bonus[0] + drop_width, bonus[1] + drop_height), fill=bonus[2])
    # bonus bomba (opcjonalnie)
    if power == "bomb":
        draw.rectangle((bomb_x, bomb_y, bomb_x + bomb_width, bomb_y + bomb_height), fill="yellow")
    # bonus laser (opcjonalnie)
    elif power == "laser":
        draw.rectangle((laser_x, laser_y, laser_x + laser_width, laser_y - laser_height), fill="blue")
    # pasek statystyk, kolejno HP, score i aktywny bonus
    draw.rectangle((0, 112, lcd.width, lcd.height), fill="#33ccff")
    draw.text((10, 115), text=f"HP: {lives} Score: {score}", fill="black",font=font)
    draw.text((100, 115), text=f"S: {special_power}", fill="black",font=font)
    lcd.ShowImage(image)

#rysowanie game over na szarym tle
def draw_game_over():
    mid_x = lcd.width // 2
    mid_y = lcd.height // 2
    draw.rectangle((mid_x - 30, mid_y - 15, mid_x + 30, mid_y + 15), fill="gray")
    draw.text((mid_x-25, mid_y-7), text="Game Over", fill="red",font=font)
    lcd.ShowImage(image)
```

Funkcja Update

```
def update():
    global game_over, power
```

```

if game_over:
    draw_game_over() #rysowanie game over
    return
if en_count <= 0: #kolejny poziom jeśli wszyscy przeciwnicy nie żyją
    set_level() #tworzenie poziomu
if power == "bomb": #obsługa bomby jeśli aktywna
    move_bomb()
elif power == "laser": #obsługa lasera jeśli aktywny
    move_laser()
move_bullets() #ruch pocisków gracza
move_enemy_bullets() #ruch pocisków przeciwników
move_enemies() # ruch przeciwników
move_bonuses() # ruch bonusów
check_collisions() # sprawdzanie kolizji
draw_game() #rysowanie gry
time.sleep(0.02)

```

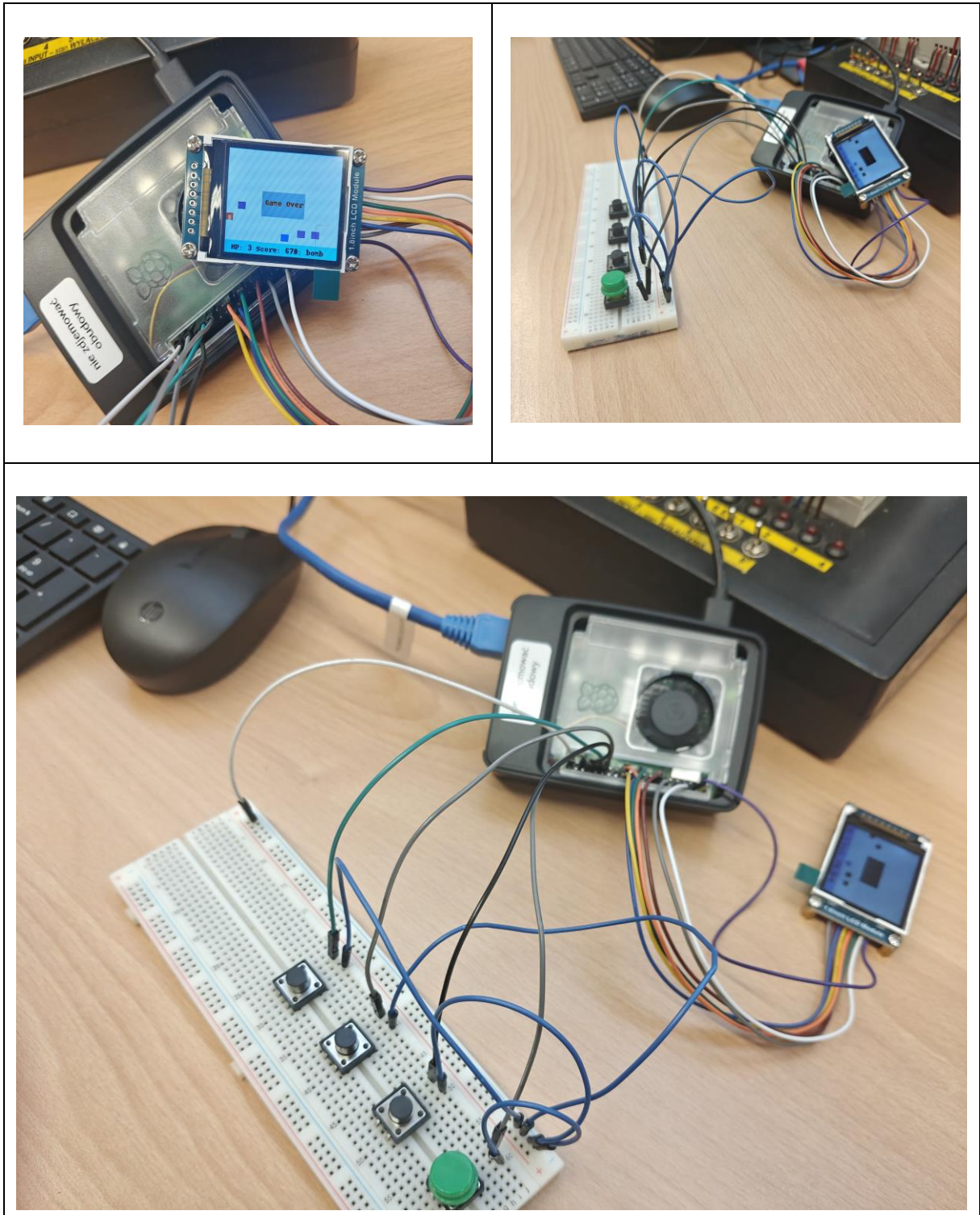
Main Loop

```

# Główna pętla gry
try:
    reset_game()
    while True:
        update()
except KeyboardInterrupt:
    print("END CTRL + C")
    GPIO.cleanup() # czyszczenie GPIO
finally:
    GPIO.cleanup() # czyszczenie GPIO
    lcd.clear() # czyszczenie wyświetlacza

```

5. Zdjęcia zrealizowanego układu.



6. Zrzuty ekranu aplikacji (jeśli występują).

7. Podsumowanie i wnioski.

Podsumowanie i wnioski dotyczące projektu

Celem projektu było stworzenia gry inspirowanej klasycznym tytułem *Space Invaders* na platformie Raspberry Pi. Zamysłem było stworzenie gry, w której gracz steruje postacią, unika przeszkód, eliminuje przeciwników oraz korzysta z różnych bonusów.

Realizacja celu: Projekt udało się zrealizować w większości z założeniami, tworząc grywalną wersję gry, która spełnia podstawowe wymagania:

1. **Sterowanie gracza:** Gracz może poruszać statkiem kosmicznym w lewo i prawo oraz strzelać.
2. **Rozgrywka:** Gra oferuje różne poziomy trudności, trudność poziomów jest losowa i zależy od typu i ilości przeciwników
3. **Akcje specjalne i bonusy:** Gracz ma możliwość używania specjalnych umiejętności, które wypadają z pokonanych wrogów.
4. **Wyświetlanie informacji:** Gra wyświetla wynik, życie gracza i bonusy, a także pokazuje komunikat „Game Over” po zakończeniu gry.

Problemy i wyzwania: Projekt nie obył się bez problemów:

1. **Sprzęt:** Wykorzystanie BeagleBoneBlack jako alternatywnej platformy okazało się niemożliwe z powodu nie znalezienia przez nas odpowiednich bibliotek do obsługi wyświetlacza LCD. Z tego powodu projekt został przeniesiony na platformę Raspberry Pi, co umożliwiło zrealizowanie gry.
2. **Obrazy na ekranie:** Problemy pojawiły się w przypadku próby wgrania grafik na wyświetlacz. Funkcja wyświetlania obrazów nie przyjmowała koordynatów, co uniemożliwiło implementację własnych obrazów obiektów.

Dalszy rozwój: Proponowane ścieżki dalszego rozwoju projektu

1. **Ekran startowy:** Dodanie menu startowego z wyborem trudności gry pozwoliłoby na dostosowanie poziomu trudności do gracza.
2. **Większa różnorodność przeciwników i bonusów:** Obecnie przeciwnicy różnią się jedynie ilością HP. Można wprowadzić większą unikalność lub nowe typy przeciwników
3. **Dźwięk:** Do projektu można dodać głośnik, co pozwoliłoby na dodanie efektów dźwiękowych
4. **Rewolucja graficzna:** Odkrycie sposobu na wyświetlanie własnych obrazów pozwoliłoby na zmianę sceny graficznej czy też dodanie animacji

Wnioski: Projekt gry na Raspberry Pi z wyświetlaczem LCD był ciekawym przedsięwzięciem, które pozwoliło na połączenie kilku dziedzin, takich jak elektronika, programowanie i projektowanie gier. Choć napotkaliśmy pewne trudności, całość stała się ciekawym doświadczeniem, które uświadomiło nam możliwości, jakie daje fizyczny sprzęt. Raspberry Pi, jako platforma do realizacji tego projektu, okazało się być doskonałym narzędziem do tworzenia interaktywnych aplikacji, mimo że niektóre aspekty sprzętowe były bardziej wymagające, niż początkowo zakładaliśmy.