Laboratorium z przedmiotu Systemy wbudowane (SW)

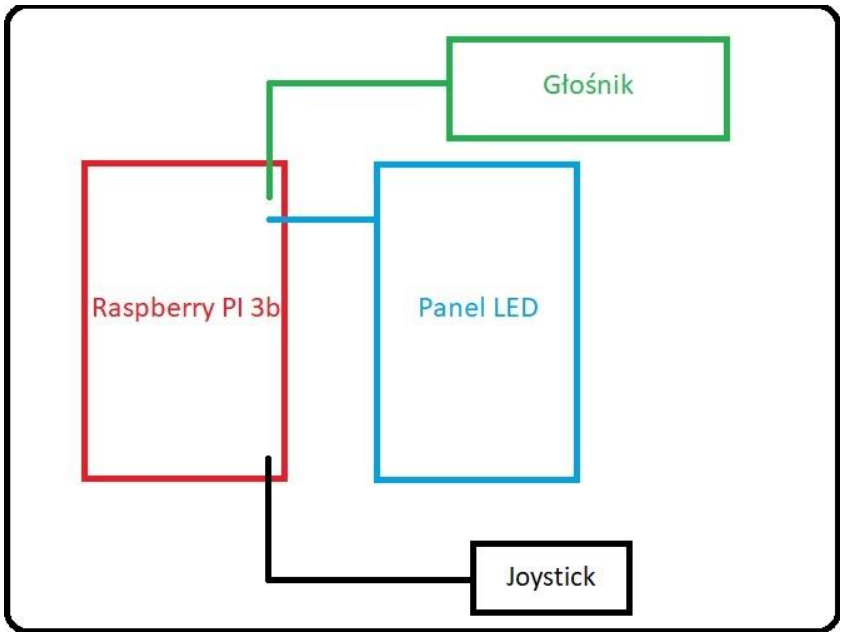
1. Karta projektu

Nazwa projektu: Gra w statki z komputerem

Prowadzący:	*Autorzy (tylko nr indeksu):*	*Grupa dziekańska:*	*I6.2*
mgr. inż. Ariel Antonowicz	*145 246* *144 482*	***Ocena:***	

Cel projektu:
Wykonanie klasycznej gry w statki z komputerem w oparciu o platformę Raspberry PI. Rozgrywka będzie widoczna na matrycowym panelu LED. Sterowanie odbywa się za pomocą joysticka. Dodatkowo zastosowane będą efekty audio (głośnik).

Schemat:



Wykorzystana platforma sprzętowa, czujniki pomiarowe, elementy wykonawcze:

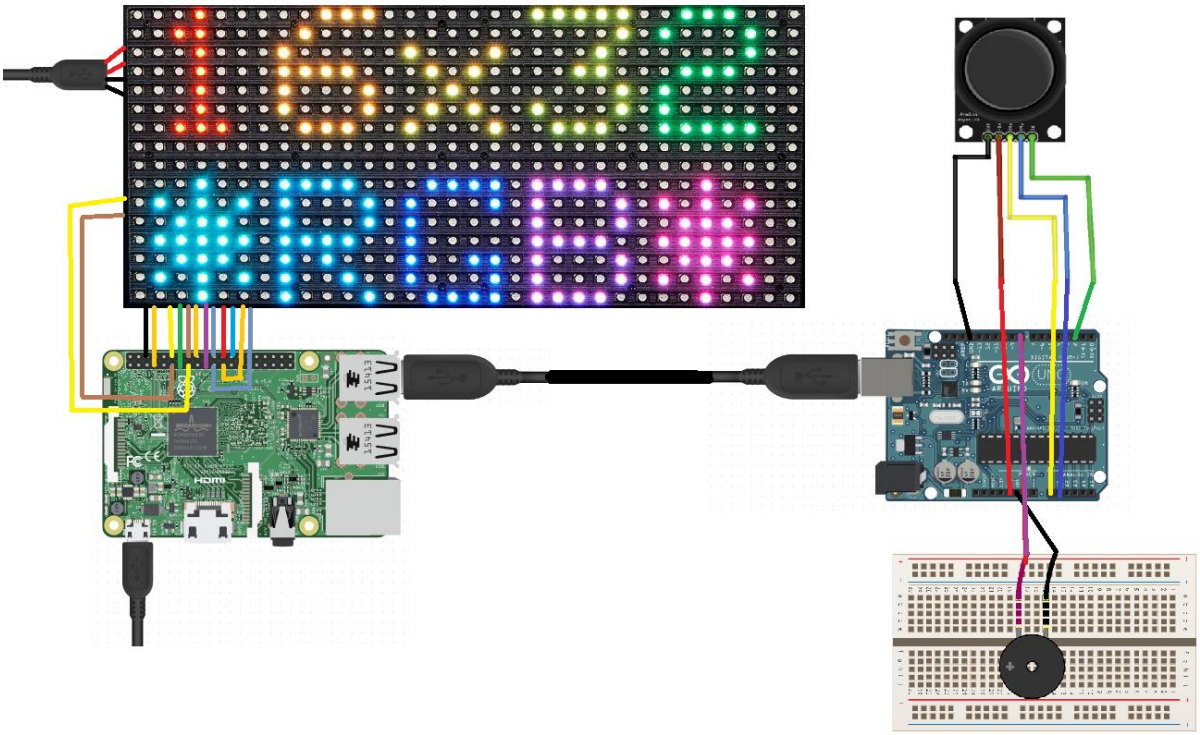
Raspberry PI 3B
Matryca LED RGB 32x16
Joystick z przyciskiem
Głośnik

2. Cel i zakres projektu

Celem projektu było wykonanie urządzenia pozwalającego rozegrać grę w statki przeciwko komputerowi. W zakres projektu wchodzi połączenie całego układu składającego się między innymi z matrycy LED RGB 32x16, joysticka oraz głośnika.

Na początku gry w sposób losowy tworzone są plansze dla obu graczy, a następnie rozpoczyna się rozgrywka. Każdy strzał pokazywany jest na matrycy. Jeżeli uda się trafić statek, zapalana jest czerwona dioda w miejscu strzału. Jeżeli spudłujemy zapala się niebieska dioda. W momencie zatopienia statku wszystkie sąsiednie pola automatycznie zmieniają kolor na niebieski. Komputer do momentu trafienia statku wykonuje losowe strzały, a w momencie, gdy uda mu się trafić któryś z statków, strzela losowo w pola sąsiadujące z trafionym strzałem. Jeżeli uda mu się zatopić statek, ponownie wybiera pole w sposób losowy. Po zakończeniu rozgrywki w konsoli na komputerze pojawia się komunikat o tym kto wygrał.

3. Schemat



4. Projekt a realizacja

Po pierwszym podłączeniu matrycy do zasilania okazało się, że lewa część matrycy jest wadliwa (matryca 16x32 składa się z dwóch matryc 16x16). Jedynie prawa strona matrycy działa poprawnie (co jest widoczne na filmie). Niestety nie mogliśmy uzyskać zamiennika ze sklepu, ponieważ w celu podłączenia matrycy do gniazdka musieliśmy odciąć końcówki pierwotnych kabli. Joystick oraz buzzer (pierwotnie miał to być głośnik) podłączyliśmy do Arduino Uno, które następnie zostało podłączone do Raspberry Pi, inaczej niż w początkowym zamyśle (podłączenie bezpośrednio do Raspberry Pi). Dodanie Arduino zostało wymuszone przez to, że nie mogliśmy uruchomić protokołu SPI1 (piny MISO, MOSI CS dla SPI1) który domyślnie jest wyłączony. Po wpisaniu zmian do */boot/config.txt* i przekopaniu ogromnej ilości forów dla zapaleńców Raspberry nadal nie mogliśmy sprawić, aby wszystko zaczęło działać. Podłączenie buzzera do Arduino wynikało również z faktu, że jakość dźwięku była dużo lepsza niż w przypadku podłączenia do Raspberry.

5. Najważniejsze fragmenty kodu z komentarzami

```
# rozstawienie  2. masztowca
for i in range(3):
    sasiedzi = []
    sasiedzi2 = []
    flag = 0
    #losowanie wspolrzednych.
    while (flag == 0):
        x = random.randrange(0, 10)
        y = random.randrange(0, 10)
        sasiedzi = sasiad(macierz, x, y)
        if macierz[x][y] == 0:
            for z in sasiedzi:
            #warunek,aby pola statku sąsiadowały ze sobą bokami, nie rogami.
                if macierz[z[0]][z[1]] == 0 and abs(z[0] - x) + abs((z[1] - y)) != 2:
                    macierz[x][y] = 1
                    macierz[z[0]][z[1]] = 1
                    statki.append([[x, y], [z[0], z[1]]])
                    sasiedzi2 = sasiad(macierz, z[0], z[1])
                    sasiedzi.extend(sasiedzi2)
                    for k in sasiedzi:
                        if k[0] == x and k[1] == y:
                            pass
                        elif k[1] == z[1] and k[0] == z[0]:
                            pass
                        else:
                            macierz[k[0]][k[1]] = '#'
                    flag = 1
                    break
```

```python
#Funckja służaca do wyświetlenia planszy na matrycy
def wypisz_plansze(macierz):
    for i in range(10):
        for j in range(10):
            if macierz[i][j] == 1:
                matrix.SetPixel(j + 1, i + 1, 255, 255, 255)
            print(macierz[i][j], end=" ")
        print('')

#Funkcja do wypisana planszy w konsoli
def wypisz_plansze_1(macierz):
    for i in range(10):
        for j in range(10):
            print(macierz[i][j], end=" ")
        print('')

#Funkcja sprawdzająca czy na podstawie strzałow udało nam się zatopić statek.
def czy_zatopiony(statki, ruchy_strzalow):
    for statek in statki:
        sprawdz = all(item in ruchy_strzalow for item in statek)
        if sprawdz == True:
            return True
    return False

#Funkcja zwracająca współrzędne zatopionego statku
def zwroc_wspolrzedne_zatopionego_statku(statki, ruchy_strzalow):
    for statek in statki:
        sprawdz = all(item in ruchy_strzalow for item in statek)
        if sprawdz == True:
            return statek
```

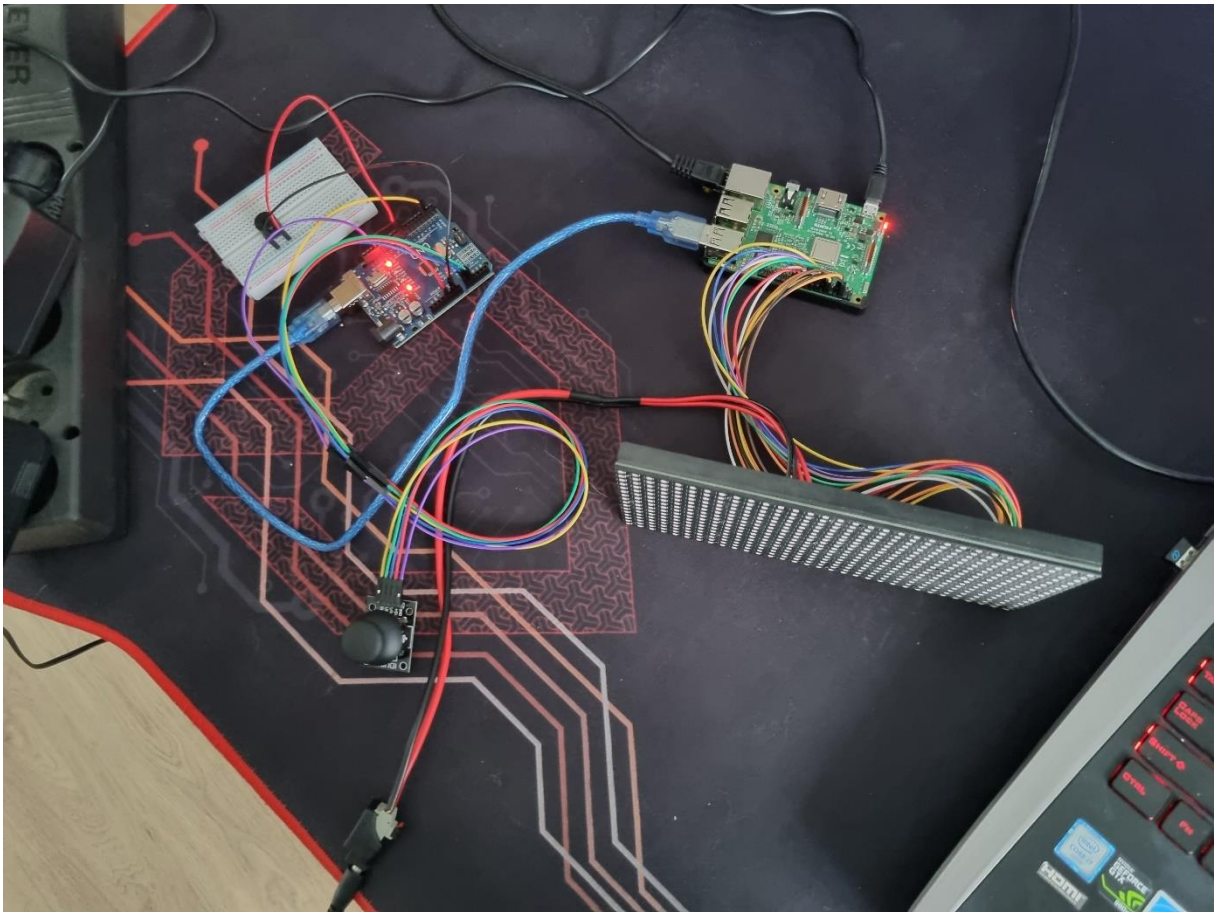
```
#petla gry
while (zycie_gracza != 0 and zycie_komputera != 0):
    #ruch gracza
    if (ruch == 'gracz'):
        if pierwszy == 1:
            #początkowe ustowienie joysticka w lewym górnym rogu.
            dane = ruch_gracza(runda_gracza, plansza_strzalow_gracza, 1, 21)
            pierwszy = 0
        else:
            #Odczytywanie współrzędnych na podstawie joysticka
            dane = ruch_gracza(runda_gracza, plansza_strzalow_gracza, x_dane + 1, y_dane + 21)
        x = dane[1] - 1
        y = dane[0] - 21

        if plansza_komputera[x][y] != 0: # jeżeli uda się trafić w statek
            zycie_komputera = zycie_komputera - 1
            plansza_strzalow_gracza[x][y] = 1
            matrix.SetPixel(y + 21, x + 1, 255, 0, 0)
            ruchy_gracza.append([x, y])
            #sprawdzenie czy udało się zatopić statek
            if czy_zatopiony(statki_komputera, ruchy_gracza) == True:
                wspolrzedne_zatopionego = zwroc_wspolrzedne_zatopionego_statku(statki_komputera, ruchy_gracza)
                sasiedzi_zatopionego = []
                #odczytanie współrzędnych sąsiadów zatopinego statku
                for wsp in wspolrzedne_zatopionego:
                    sasiedzi_zatopionego.extend(sasiad(plansza_komputera, wsp[0], wsp[1]))
                for pole in sasiedzi_zatopionego:
                    wsp_x = pole[0]
                    wsp_y = pole[1]
                    # zapalnie niebieskich diód wokół zatopionego statku.
                    if plansza_komputera[wsp_x][wsp_y] == 0:
                        plansza_strzalow_gracza[wsp_x][wsp_y] = 'X'
                        matrix.SetPixel(wsp_y + 21, wsp_x + 1, 0, 0, 255)
                    else: #zapalenie czerwonej diody w miejscu gdzie znajduej się statek.
                        plansza_strzalow_gracza[wsp_x][wsp_y] = 1
                        matrix.SetPixel(wsp_y + 21, wsp_x + 1, 255, 0, 0)
                ruchy_gracza = []
        else:#nietrafiony strzał gracza.
            plansza_strzalow_gracza[x][y] = 'X'
            matrix.SetPixel(y + 21, x + 1, 0, 0, 255) #zapalenie niebieskiej diody w miejscu strzału.

        ruch = 'komputer'
        #zapamietanie współrzendych poprzedniego strzału, aby ustawić w tym miejscu joystick.
        x_dane = x
        y_dane = y
```

```
    if (ruch == 'komputer'): #ruch komputera
        if dobre_ruchy == []: #sprawdzenie czy występują jakieś potencjalnie dobre ruchy(w poprzednim ruchu trafiono statek).
            flaga = 0
            while (flaga == 0):
                #losowanie współrzędnych
                x = random.randrange(0, 10)
                y = random.randrange(0, 10)
                if [x, y] not in ruchy_komputera:
                    flaga = 1
            ruchy_komputera.append([x, y])
            if plansza_gracza[x][y] != 0: # trafiony statek.
                zycie_gracza = zycie_gracza - 1
                #dodanie do listy dobre_ruchy pól na których znajduje się kolejny maszt.
                dobre_ruchy.extend(sasiad_bez_rogow(plansza_gracza, x, y))
                chwilowe_ruchy_komputera.append([x, y])
                plansza_strzalow_komputera[x][y] = 1
                matrix.SetPixel(y + 1, x + 1, 255, 0, 0) #ustawienie czerwonej diody.

                if czy_zatopiony(statki_gracza, chwilowe_ruchy_komputera) == True: #zatopiony statek
                #dodanie do historii ruchów sąsiadów zatopionego statku.
                    ruchy_komputera.extend(sasiad(plansza_gracza, x, y))
                    sasiedzi_zatopionego = sasiad(plansza_gracza, x, y)
                    for pole in sasiedzi_zatopionego:
                        wsp_x = pole[0]
                        wsp_y = pole[1]

                        if plansza_gracza[wsp_x][wsp_y] != 0:
                            plansza_strzalow_komputera[wsp_x][wsp_y] = 1
                            #zapalenie czerwonej diody w miejscu zatopionego statku.
                            matrix.SetPixel(wsp_y + 1, wsp_x + 1, 255, 0, 0)
                        else:
                            #zapalenie niebieskiej diody dookoła zatopionego statku
                            plansza_strzalow_komputera[wsp_x][wsp_y] = 'X'
                            matrix.SetPixel(wsp_y + 1, wsp_x + 1, 0, 0, 255)
                            ruchy_komputera.append([wsp_x, wsp_y])
                    #wyzerowanie list z ruchami.
                    dobre_ruchy = []
                    chwilowe_ruchy_komputera = []

            else:#nietrafiony strzał
                plansza_strzalow_komputera[x][y] = 'X'
                #ustawienie diody na niebieską
                matrix.SetPixel(y + 1, x + 1, 0, 0, 255)
        else: # dostępne są jakieś dobre ruchy.
            flaga = 0
            #wylosowanie strzału
            while (flaga == 0):
                strzal = random.choice(dobre_ruchy)
                if strzal not in ruchy_komputera:
                    flaga = 1

            ruchy_komputera.append(strzal)
            x, y = strzal[0], strzal[1]
            dobre_ruchy.remove(strzal)#usunięcie wylosowanego strzału z listy dobrych ruchów.
            if plansza_gracza[x][y] != 0:  # komputer trafił
                chwilowe_ruchy_komputera.append([x, y])
                zycie_gracza = zycie_gracza - 1
                if czy_zatopiony(statki_gracza, chwilowe_ruchy_komputera) == True: # zatopiony statek
                    wspolrzedne_zatopionego = zwroc_wspolrzedne_zatopionego_statku(statki_gracza,
                                                                                   chwilowe_ruchy_komputera)
                    sasiedzi_zatopionego = []

                    for wsp in wspolrzedne_zatopionego:
                        sasiedzi_zatopionego.extend(sasiad(plansza_gracza, wsp[0], wsp[1]))
                    for pole in sasiedzi_zatopionego:
                        wsp_x = pole[0]
                        wsp_y = pole[1]
                        #ustawienie czerwonej diody w miejscu zatopionego statku
                        if plansza_gracza[wsp_x][wsp_y] != 0:
                            plansza_strzalow_komputera[wsp_x][wsp_y] = 1
                            matrix.SetPixel(wsp_y + 1, wsp_x + 1, 255, 0, 0)
                        # ustawienie niebieskiej diody wokół zatopionego statku
                        else:
                            ruchy_komputera.append([wsp_x, wsp_y])
                            plansza_strzalow_komputera[wsp_x][wsp_y] = 'X'
                            matrix.SetPixel(wsp_y + 1, wsp_x + 1, 0, 0, 255)
                    dobre_ruchy = []
                    ruchy_komputera.extend(sasiad(plansza_gracza, x, y))
                    chwilowe_ruchy_komputera = []


                else:# strzał trafiony, lecz nie zatopił jeszcze statku.
                    plansza_strzalow_komputera[x][y] = '1'
                    matrix.SetPixel(y + 1, x + 1, 255, 0, 0)
                    #dodanie do list dobre_ruchy sąsiadów trafionego pola
                    dobre_ruchy.extend(sasiad_bez_rogow(plansza_gracza, x, y))
            else:# strzał nietrafiony spośród dobrych ruchów.
                plansza_strzalow_komputera[x][y] = 'X'
                # ustawienie niebieskiej diody w miejscu pudła
                matrix.SetPixel(y + 1, x + 1, 0, 0, 255)

        print("PLANSZA STRZAŁÓW KOMPUTERA\n")
        wypisz_plansze_1(plansza_strzalow_komputera)
        ruch = 'gracz'

#Wypisanei zwycięzcy
 if zycie_gracza == 0:
     print("WYGRAŁ KOMPUTER")
     time.sleep(2)
     sys.exit(0)
 else:
     print("WYGRAŁ GRACZ")
     time.sleep(2)
     sys.exit(0)
```

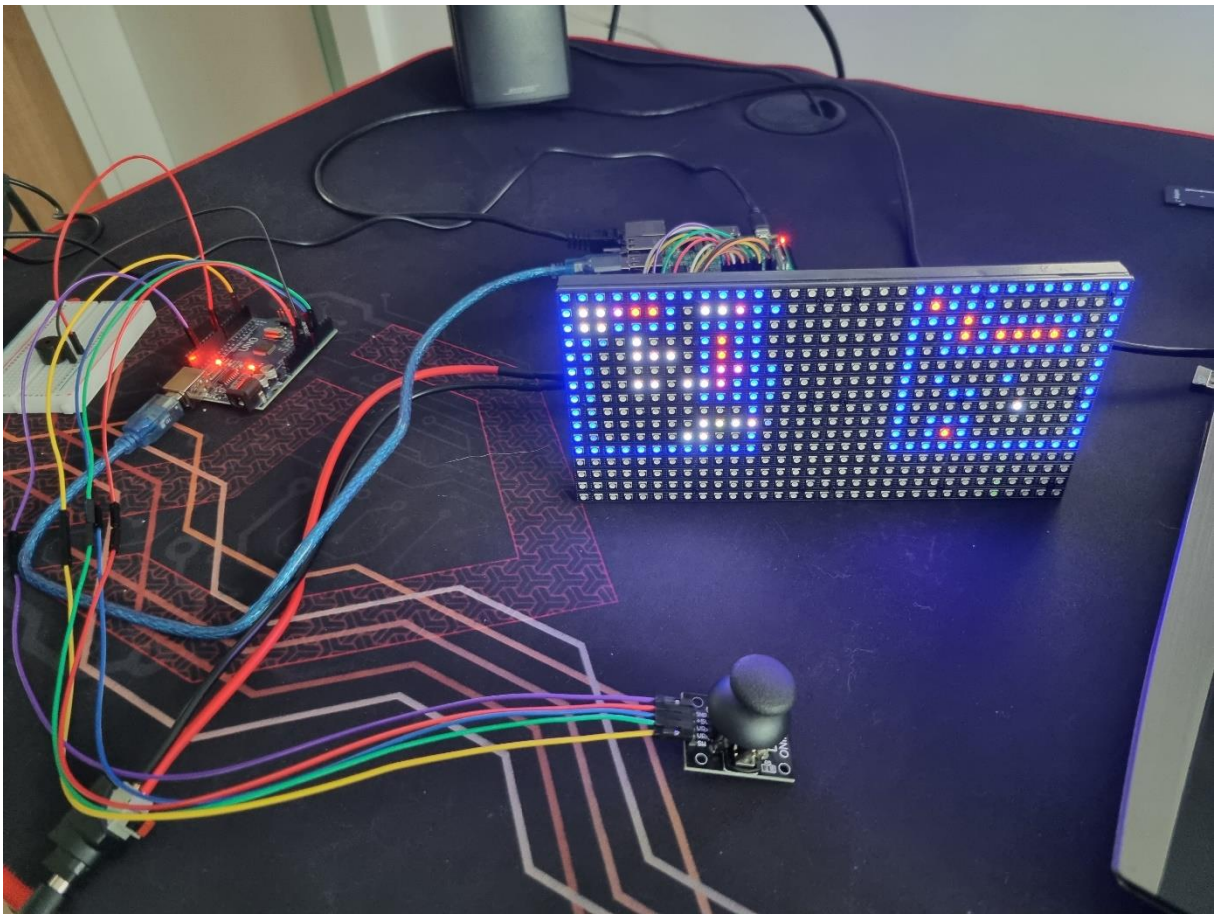
```python
#odczytanie współrzędnych z joysticka
def ruch_gracza(runda_gracza, plansza_strzalow_gracza,x,y):
    ser = serial.Serial('/dev/ttyUSB0',9600)
    dane_pelne = []
    dane = []
    licznik = 0
    matrix.SetPixel(y,x,255,255,255)
    while (runda_gracza == 1):
        read = int(ser.readline().strip())
        if read >= 0 and read <= 1023:
            read_serial = read
        if licznik == 2:
            dane.append(read_serial)
            #sprawdzenie, czy dane są przekazywanie dalej w poprawny sposób
            if dane[0] in [0,1]:
                dane_pelne = dane
            else:
                continue
            dane = []
            licznik = 0
            #po wciśnięciu joysticka oddawany jest strzał i wychodzimy z petli
            if dane_pelne[0] == 0:
                matrix.SetPixel(y,x,0,0,0)
                matrix.SetPixel(y,x,255,0,255)
                time.sleep(2)
                matrix.SetPixel(y,x,0,0,0)
                runda_gracza = 0
            #ruch góra/dół
            elif dane_pelne[1] < 100:
                if plansza_strzalow_gracza[x-1][y-21] == 'X':
                    matrix.SetPixel(y,x,0,0,255)
                elif plansza_strzalow_gracza[x-1][y-21] == 1:
                    matrix.SetPixel(y,x,255,0,0)
                else:
                    matrix.SetPixel(y,x,0,0,0)
                if x > 1:
                    x = x - 1
                matrix.SetPixel(y,x,255,255,255)
            elif dane_pelne[1] > 900:
                if plansza_strzalow_gracza[x-1][y-21] == 'X':
                    matrix.SetPixel(y,x,0,0,255)
                elif plansza_strzalow_gracza[x-1][y-21] == 1:
                    matrix.SetPixel(y,x,255,0,0)
                else:
                    matrix.SetPixel(y,x,0,0,0)
                if x < 10:
                    x = x + 1
                matrix.SetPixel(y,x,255,255,255)
            #ruch w lewo/prawo
            elif dane_pelne[2] < 100:
                if plansza_strzalow_gracza[x-1][y-21] == 'X':
                    matrix.SetPixel(y,x,0,0,255)
                elif plansza_strzalow_gracza[x-1][y-21] == 1:
                    matrix.SetPixel(y,x,255,0,0)
                else:
                    matrix.SetPixel(y,x,0,0,0)
                if y < 30:
                    y = y + 1
                matrix.SetPixel(y,x,255,255,255)
            elif dane_pelne[2] > 900:
                if plansza_strzalow_gracza[x-1][y-21] == 'X':
                    matrix.SetPixel(y,x,0,0,255)
                elif plansza_strzalow_gracza[x-1][y-21] == 1:
                    matrix.SetPixel(y,x,255,0,0)
                else:
                    matrix.SetPixel(y,x,0,0,0)
                if y > 21:
                    y = y - 1
                matrix.SetPixel(y,x,255,255,255)
        else:
            dane.append(read_serial)
            licznik = licznik + 1
    #zwracamy wspolrzedne joysticka na matrycy
    return [y,x]
```

```arduino
//wysyłanie współrzędnych Joysticka
void setup() {
  pinMode(SW_PIN, INPUT);
  digitalWrite(SW_PIN, HIGH);
  Serial.begin(9600);
}

void loop() {
  Serial.println(digitalRead(SW_PIN));
  Serial.println(analogRead(X_PIN));
  Serial.println(analogRead(Y_PIN));
```

6. Zdjęcie fizycznego urządzenia/połączeń



7. Zrzuty z ekranu aplikacji



8. Podsumowanie i wnioski

Projekt został zrealizowany z lekkimi zmianami sprzętowymi.

Wnioski: Utworzenie planszy należy zacząć od rozłożenia czteromasztowca. Stawiając statki w sposób losowy, rozpoczynając od najmniejszych często dochodziło do sytuacji w której nie było możliwości rozłożenia największego statku i program się zawieszał. Dopiero zamiana kolejności rozłożenia statków przyniosła oczekiwany rezultat.

Ulepszeniem projektu mogła by być przede wszystkim wymiana matrycy. Poza tym oczywistym faktem, moglibyśmy dodać jakieś efekty wizualne przed i po zakończeniu rozgrywki (np. menu gry i efekty typu wybuchający fajerwerk po wygranej).

Link do filmu: link!