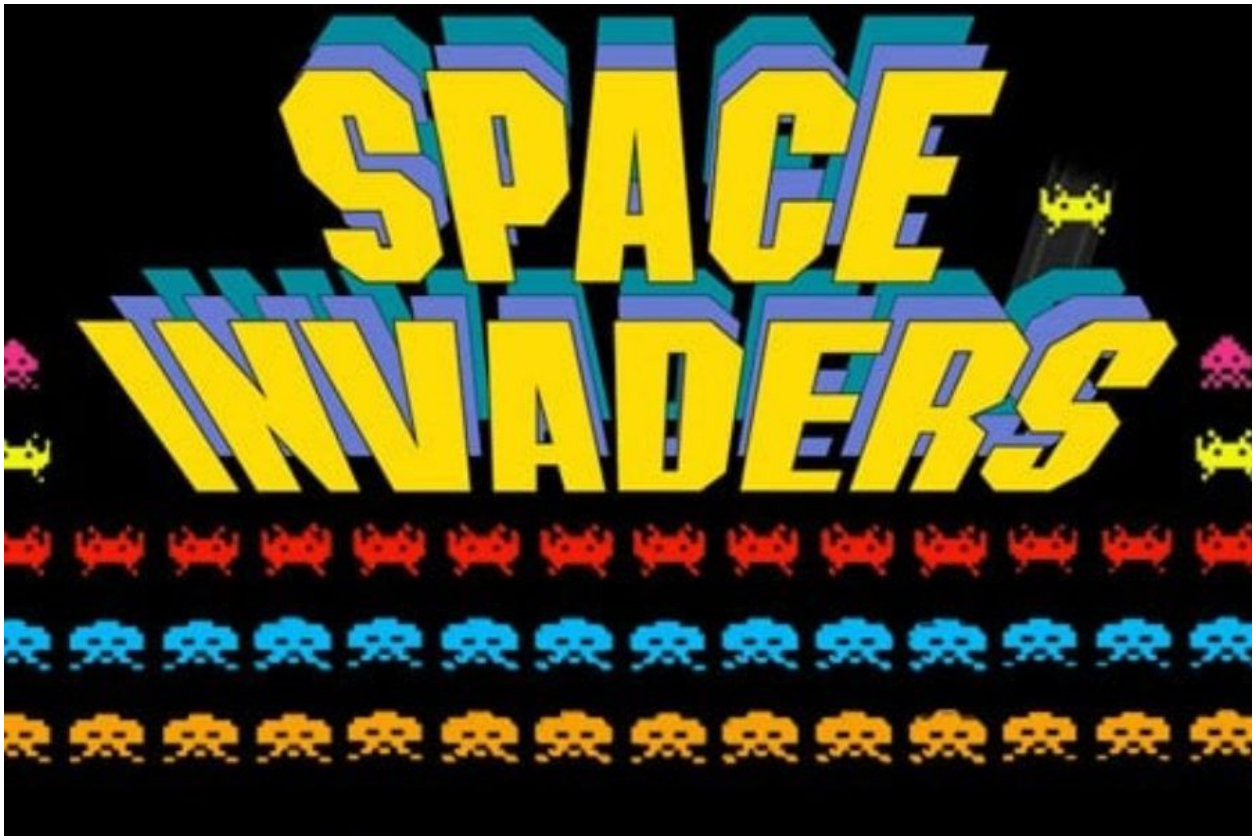


Projekt demonstracyjnej adaptacji gry Space Invaders w oparciu o układ ATMega 2560

24.11.2019

Autor projektu: Szymon Sandura

w ramach przedmiotu: Techniki mikroprocesorowe i systemy wbudowane

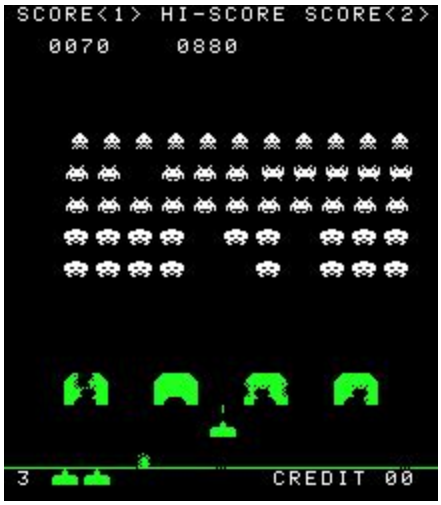
Prowadzący: mgr inż. Jan Duda

Wstęp

Powyższy projekt stanowi prostą adaptację kultowej gry Space Invaders. Inspiracji do jej powstania można szukać w grze z 1962 roku o nazwie Spacewar. Pierwsza wersja Space Invaders powstała w roku 1978 i oryginalnie została wydana w formie automatu do gier (ang. Arcade System Board) Taito 8080.
System ten został wyprodukowany przez japońską firmę Taito Corporation i opiera się na układzie Intel 8080 z taktowaniem zegara wynoszącym 2Mhz oraz na dedykowanym chipsecie do obsługi dźwięku SN76477 (taktowanie ~2Mhz) firmy Texas Instrument. Autorem projektu i pomysłodawcą jest Tomoshiro Nishikado.

Założenia pierwowzoru

Space Invaders opiera się na prostych zasadach. Rozgrywka polega na zestrzeliwaniu kolejnych zbliżających się z góry ekranu fal przeciwników. Gracz steruje statkiem wyposażonym w działo laserowe i może przemieszczać się w jednym wymiarze, tj. w lewo lub prawo.



Przykład implementacji gry Space Invaders

Cele

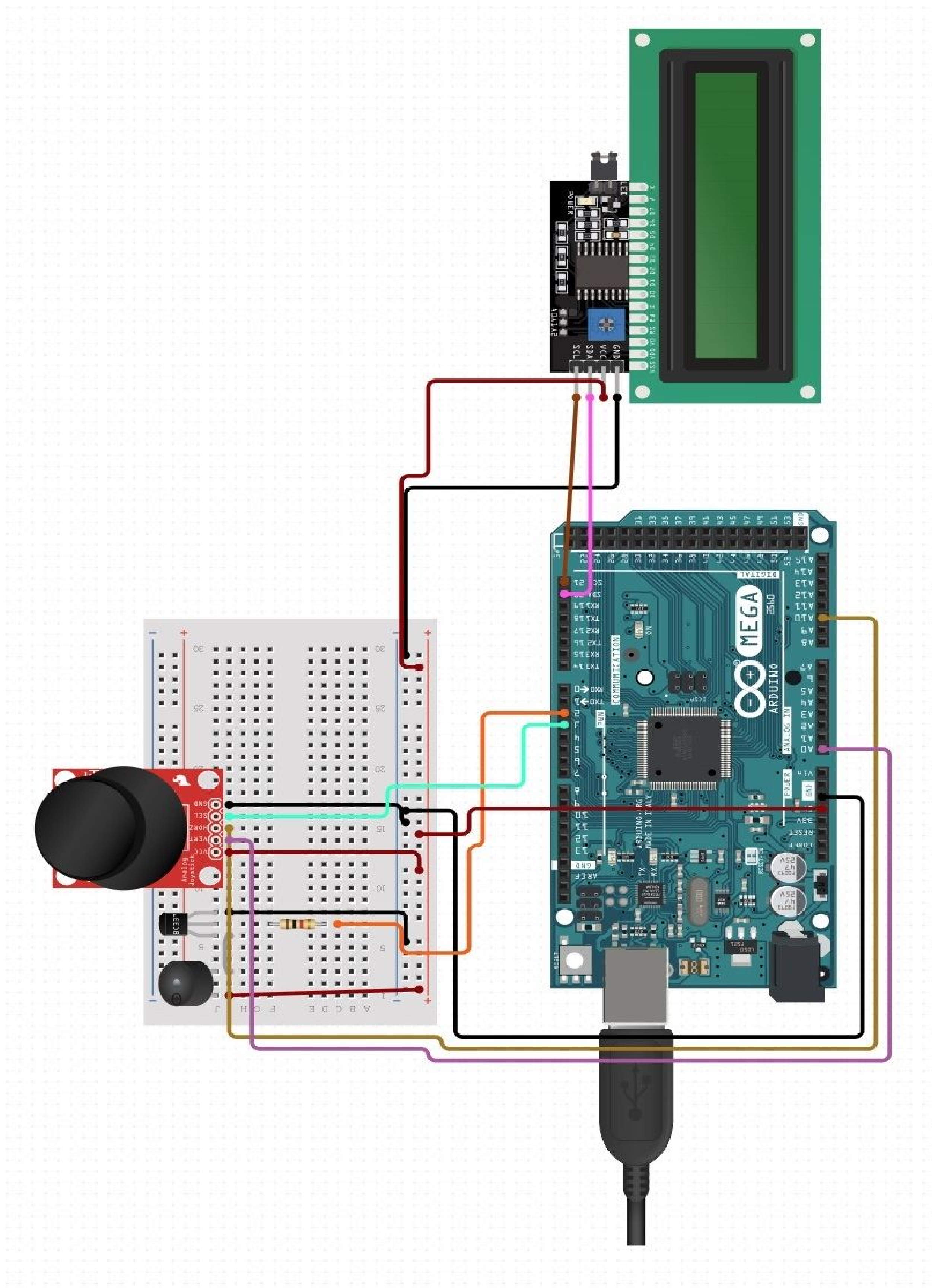
1. Odwzorowanie podstawowych mechanik z oryginalnej gry: ruch przeciwników, ruch pocisku, przemieszczanie się statkiem.
2. Tworzenie poziomów za pomocą generatora liczb pseudolosowych
3. Implementacja sterowania poprzez gałkę analogową i wbudowany w nią przycisk
4. Implementacja wyświetlacza LCD
5. Obsługa asynchronicznych wątków celem uniezależnienia prędkości obiektów na ekranie
6. Stworzenie prostej ścieżki dźwiękowej występującej podczas włączenia układu oraz między kolejnymi falami
7. Zbudowanie prostego (ideowego) wzmacniacza celem wzmocnienia sygnału audio

Wymagania sprzętowe

- Arduino ATmega 2560 lub zamiennik
- ekran LCD o wymiarach 16 x 2 ustawiony w orientacji pionowej (16 wierszy, 2 kolumny) korzystający z interfejsu I2C
- Gałka analogowa z wbudowanym przyciskiem (alternatywnie osobny
- Pasywny buzzer zasilany prądem stałym
- Opornik 1k Ohm
- Tranzystor BC337 typu NPN

Schemat

Na następnej stronie przedstawiono przykładowy schemat podłączenia układu.


MEGA
ARDUINO

Opis projektu programu

Pełny kod programu wraz z listą zmiennych globalnych dostępny jest na repozytorium autora pod adresem:
https://github.com/HerringTheCoder/Arduino-Space-Invaders/blob/master/sketch_SpaceInvaders/sketch_SpaceInvaders.ino

I. Inicjalizacja aplikacji

```
void setup()
{
  randomSeed(analogRead(5)); // using unconnected A5 pin to generate random numbers
  pinMode(JOYS_SW_DIO, INPUT_PULLUP);
  pinMode(BUZZER_DIO, OUTPUT);
  digitalWrite(JOYS_SW_DIO, HIGH);
  lcd.init();       // initialize the lcd
  lcd.backlight(); //turn on backlight
  loadLevel();
  playMusic();
}
```

W pierwszym kroku zdefiniowano generator liczb losowych na bazie wybranego, nie podłączonego pinu analogowego. Nieprzewidywalny charakter występujących na nim szumów posłuży do tworzenia zróżnicowanej sekwencji poziomów.
Następnie przypisany zostaje pin wykorzystany do obsługi przycisku wbudowanego w gałkę w trybie INPUT_PULLUP, czyli z wykorzystaniem wewnętrznego rezystora podciągającego oraz pin odpowiedzialny za podawanie sygnału do głośnika.

W kolejnym etapie należy zainicjować wyświetlacz LCD oraz włączyć jego podświetlenie.
Po tych operacjach następuje załadowanie pierwszego poziomu oraz odtworzenie prostej melodii.

II. Obsługa wyświetlacza

Do obsługi wyświetlacza wykorzystano bibliotekę LiquidCrystal_I2C dostępną pod adresem: https://github.com/johnrickman/LiquidCrystal_I2C

Jest to specjalna wersja znanej biblioteki LiquidCrystal zapewniająca kompatybilność z interfejsem I2C, ale poza drobnymi różnicami we wstępnej konfiguracji, obsługa programowa tej biblioteki jest prawie identyczna. Stworzenie obiektu reprezentującego wyświetlacz zostało zdefiniowane w następujący sposób:

```
/* global variables */
LiquidCrystal_I2C lcd(0x3f,16,2);  // set the LCD address to 0x3f for a 16 chars and 2 line display
```

Uwaga! Przypisany adres może zależeć od producenta wyświetlacza LCD oraz od zastosowanego układu. Celem znalezienia adresu zaleca się użycie programu I2cScanner dostępnego pod adresem: https://playground.arduino.cc/Main/I2cScanner/

III. Obsługa dźwięku

Stworzona ścieżka dźwiękowa składa się z wybranych wysokości dźwięku oraz rytmu. W tym celu zadeklarowano częstotliwości odpowiadające wybranym nutom w systemie równomiernie temperowanym przy założeniu wzorcowego A=440Hz :

```
 /*sound section*/
#define NOTE_G4 391 //G4
#define NOTE_A4 440 //A4
#define NOTE_C5 523 //C5
#define NOTE_D5 587 //D5
#define NOTE_E5_FLAT 622 //E flat (Eb)
#define NOTE_E5 659  //E5
```

Dźwięki występujące w wyższych oktawach generuje się przez podwojenie częstotliwości dźwięku z oktawy niższej (np. A4= 440Hz, A5= 880Hz, A6= 1760Hz itd.)

Zmienny rytm uzyskano natomiast dzięki zastosowaniu opóźnień. Długość opóźnienia determinuje tempo odtwarzania muzyki.

```
#define TEMPO 200 /*sets the sustain parameter of each note/pause, higher value = slower */
```

Przykładowy zapis melodii:

```
void playMusicBreak()
{
  tone(BUZZER_DIO, NOTE_A4);
  delay(TEMPO);
  noTone(BUZZER_DIO);
  delay(TEMPO);
  tone(BUZZER_DIO, NOTE_C5);
  delay(TEMPO);
  noTone(BUZZER_DIO);
  delay(TEMPO);
  tone(BUZZER_DIO, NOTE_G4);
  delay(TEMPO);
  tone(BUZZER_DIO, NOTE_A4);
  delay(TEMPO);
  noTone(BUZZER_DIO);
}
```

IV. Generowanie mapy gry

Mapa gry przechowywana jest w postaci tablicy znaków o domyślnym dwukrotnością wielkości ekranu. Pierwsza połowa tabeli przechowuje obiekty widoczne dla gracza i pokrywa się z tablicą wyświetlacza. Druga połowa tabeli to przestrzeń zarezerwowana dla statków przeciwnika.
Każdy występujący obiekt ma przydzielony stosowny symbol:

```
#define SHIP_ASCII 0b10101010 /* Katakana 工 sign */
#define ALIEN_ASCII 0b01011000 /* Upper Case 'X' */
#define BULLET 0b10100101 /* - sign */
#define EXPLOSION 0b00101010 /*   * character */
```

Poniższa tabela obrazuje powyższy koncept mapy gry (tu dla wyświetlacza o wymiarach 10 x 2):

			X	X		X			X		X								
		X			X		X	X		X		X	*				-	-	工

Przestrzeń niewidoczna Przestrzeń ekranu LCD

V. Obsługa sterowania

Sterowanie w grze pozwala na wykonanie trzech akcji: skręt w lewo, skręt w prawo oraz wystrzelenie pocisku. Zmiana pozycji zrealizowana została za pomocą gałki analogowej z przetwornikiem analogowo-cyfrowym o rozdzielczości 10 bitów (zakres wartości od 0 do 1023). Dla zapewnienia responsywności zadeklarowano wartości graniczne (ang. deadzone) przy których odbywa się przemieszczenie statku. Wartości zostały dobrane w sposób heurystyczny.

```
/*analog input secton*/
#define JOYS_VRY_DEADZONE_LEFT 750
#define JOYS_VRY_DEADZONE_RIGHT 150
```

W pętli głównej programu co ok. 20ms odczytywana jest bieżąca pozycja gałki, a następnie porównywana z wartościami granicznymi, co przy spełnieniu warunków skutkuje wywołaniem stosownej metody: turnLeft (linie 79-86), turnRight (linie 90-98) lub w przypadku naciśnięcia przycisku fire (linie 130-146).

VI. Asynchroniczne zdarzenia w programie - pętla główna

Układ ATmega 2560 dysponuje procesorem jednowątkowym. Oznacza to, że w danym momencie można obsłużyć wyłącznie jedną operację.
Kluczowym warunkiem udanej rozgrywki w grze akcji jest responsywność sterowania, co w praktyce realizuje się skracając opóźnienia między operacjami odczytu pozycji kontrolera. Konflikt powstaje jednak w momencie kiedy na tym samym wątku trzeba obsłużyć również przemieszczanie się statków przeciwnika, które powinno odbywać się nawet kilka lub kilkanaście razy wolniej.
W związku z tym należy zaimplementować asynchroniczne odwołania.
Aby uzyskać ten efekt czas gry podzielono na cykle odczytu (nazwa luźno nawiązuje do cykli procesora), których interwał wynosi ustaloną długość opóźnienia. Dodatkowo wprowadzono stałą odnoszącą się do skali czasu, która determinuje po ilu cyklach następuje wyjście z pętli odczytu.
Przykładowo dla domyślnych wartości:

```
/*Speed control*/
#define READ_DELAY 20
#define TIMESCALE 5 /*sets the time scale of objects (enemies, bullets) moving on the map,
```

Teoretyczny czas jednej tury wynosi:
$READ\_DELAY\ (20ms) * TIMESCALE\ (5) = 100ms$
Po upłynięciu tego czasu wykonują się procedury moveBullets (linie 150-176) oraz moveEnemies (linie 178-198), które aktualizują pozycję pocisków (przesunięcie w stronę statków wroga) oraz pozycję przeciwników (przesunięcie w stronę gracza).

Po zakończeniu 32. tury (czyli po upłynięciu czasu potrzebnego do przebycia drogi między krańcowymi punktami tabeli/mapy gry) następuje ponowne wygenerowanie poziomu, licznik tur resetuje się i pojawia się przerywnik muzyczny.

Podsumowanie

Proces tworzenia projektu był bardzo dobrą okazją do zapoznania się z elementami wykraczającymi poza zakres laboratorium takimi jak: obsługa gałki analogowej, interfejsu I2C, obsługa pasywnego buzzera, ujęcie powszechnego zapisu nutowego za pomocą fizycznych własności. Największym jednak wyzwaniem było wdrożenie asynchronicznych odwołań. We współczesnych implementacjach języków (w tym w pełnej, bieżącej wersji C++) proces ten został zautomatyzowany i jest częścią standardowych bibliotek. Układy Arduino operują więc na ograniczonej wersji bibliotek z mniejszym narzutem, co najpewniej zostało podyktowane małą mocą obliczeniową i małą pamięcią (często już zajętą przez liczne biblioteki dedykowane podłączonym elementom).

Poza dostrzeżonymi korzyściami, widać również możliwości dalszego rozwoju projektu.
Wśród potencjalnych pomysłów można wymienić:
- Zmianę wyświetlacza na wykonany w technologii OLED o znacznie większej rozdzielczości i obsłudze wielu kolorów
- Podłączenie wyświetlacza LED z bieżącą punktacją
- Zapisywanie najlepszych wyników na zewnętrznym nośniku
- Wprowadzenie poziomów trudności (np. przez zewnętrzne sterowanie skalą czasu)
- Odtwarzanie muzyki podczas rozgrywki (najlepiej za pomocą zewnętrznego chipsetu)

Pomimo wielu pomysłów należy jednak zwrócić uwagę na ograniczenia samego układu ATmega 2560. Podczas tworzenia kodu i testowania okazało się, że należy ograniczyć responsywność kontrolera (zwiększyć opóźnienie odczytu), ponieważ zbyt częste odczyty spowalniają wykonywanie pozostałych procedur. Podczas dodawania kolejnych elementów może się więc okazać, że moc obliczeniowa nie jest wystarczająca do utrzymania aktualnych warunków gry. Oczywiście możliwa jest optymalizacja poprzez zastosowanie wskaźników lub wstawek assemblerowych, ale należy pamiętać, że Arduino Mega 2560 zwyczajnie nie jest układem dedykowanym do tworzenia skomplikowanych, wielowątkowych gier.