Dokumentacja projektu
z Systemów Wbudowanych

Michał Kielski i Wojciech Franczuk

1. Opis projektu

Głównym założeniem pojektu było stworzenie gry korzystającej z akcelerometru i wyświetlacza. Gra miała polegać na przechodzeniu kulką przez labirynt. Sterowanie kulką miało odbywać się za pomocą akceleromatru. Gra oczywiście miała służyć głownie celom rozrywkowym.

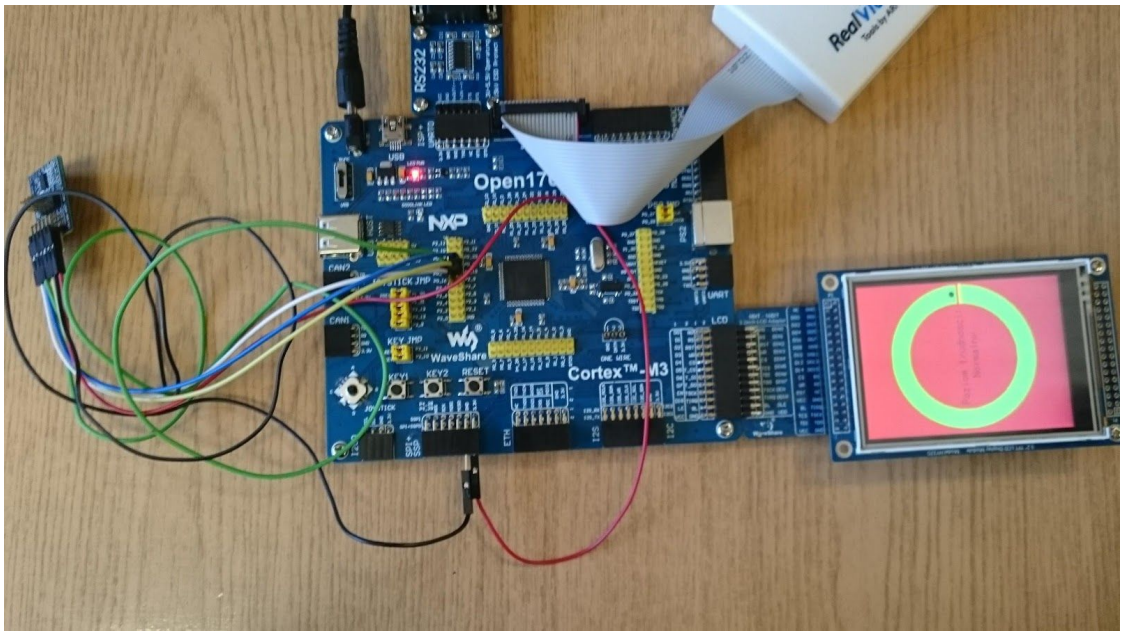
2. Założenia Projektu

Projekt polegał na wykorzystaniu akcelerometru(PMOD ACL) do sterowania kulką na ekranie LCD(Open1768). Miało to symulować gre, w której kulka musi przejść przez labirynt bez dotykania ścian. Dotknięcie ściany kończy się porażką i powrotem kulki do pozycji początkowej. Dodatkowo, wykorzystaliśmy oba przyciski na ARM'ie(Cortex M3), skonfigurowane do działania z przerwaniami EINT1 i EINT2, do wybierania opcji w naszej grze.
Akcelerometr komunikuje się się z procesorem za pomocą układu SSP0. Początkowo sprawiało to problemy, gdyż ekran LCD wykorzystuje ten sam sposób komunikacji. Udało się nam to jednak obejść, korzystając z alternatywnych wyjść pinów, znajdujących się na górnej części płytki. Dane pobierane są z odpowiednich rejestrów akcelerometru przy każdym przejściu pętli while, zaraz przed narysowaniem kulki.
Dane przetwarzane są w czasie rzeczywistym i wyświetlane na ekranie w postaci ruchu kulki. Układ nie komunikuje się z komputerem. Komunikacja z użytkownikiem ma miejsce poprzez ekran LCD, na którym wyświetla się aktualny poziom trudności(czyli szerokość pierścienia), oraz ewentualna informacja o przegranej lub wygranej, w razie wykrycia danego wydarzenia.

3. Interfejs użytkownika

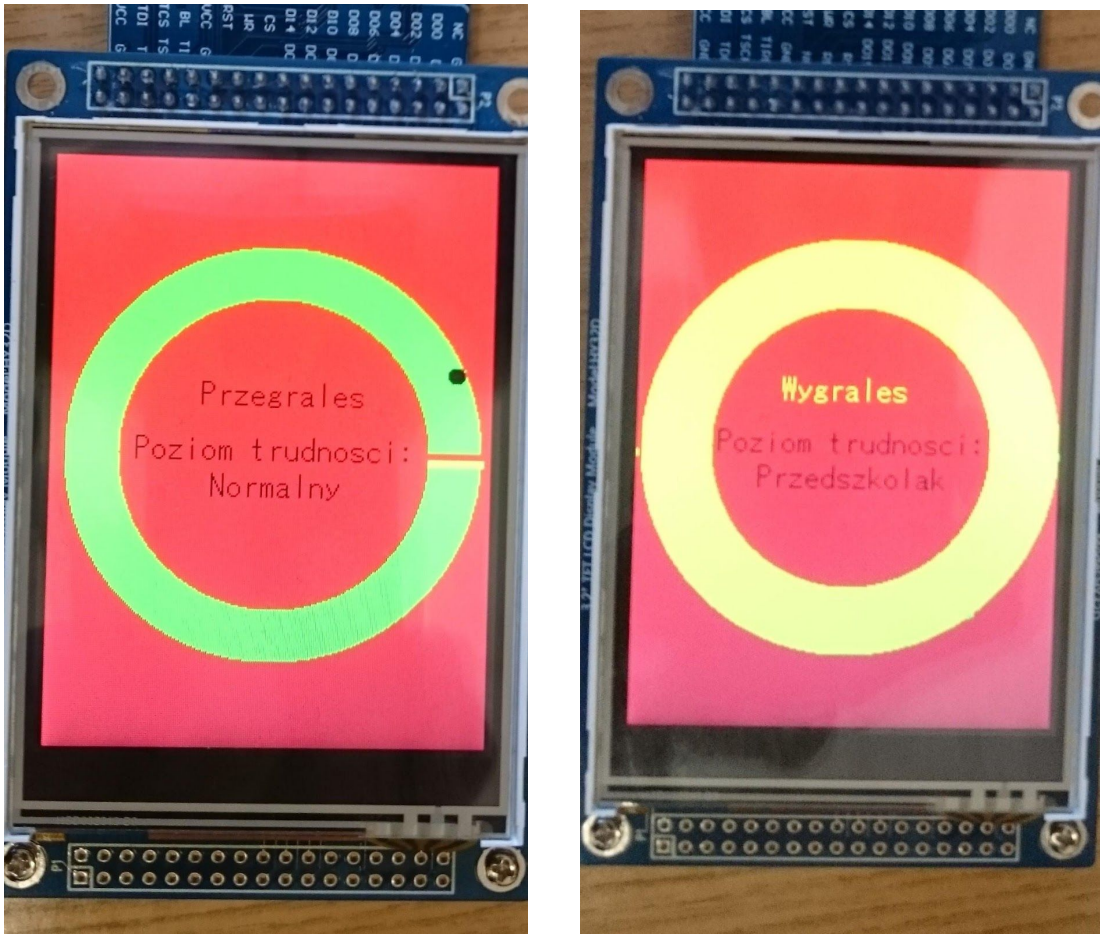
Sterowanie w grze opiera się na dwóch przyciskach zawartych w płytce ewaluacyjnej oraz układzie z akcelerometrem. Przycisk podpisany jako "KEY 2" służy do wyboru jednego z czterech dostępnych poziomów trudności. Natomiast przycisk "KEY 1" potwierdza wybór poziomu i rozpoczyna grę. Po przyciśnięciu w pierścieniu pojawia się kulka, która sterowana jest akcelerometrem. Układ

akcelerometru powinien być ustawiony wyprowadzeniami do lewej strony wyświetlacza (tak jak na zdjęciu 1), aby jego wychylenia były zgodne z ruchem kulki.



Zdjęcie 1. Układ wykorzystany w projekcie

Na zdjęciu 2 pokazane są dwa przykładowe stany gry. Po lewej stronie po zetknięciu kulki z czerwonym obszarem pojawia się informacja o przegranej grze. Następnie gra wczytuje się od początku z łatwiejszym poziomem (jeśli taki istnieje). Po prawej stronie widoczna jest informacja o wygranej, która pojawia się po dotarciu kulką do żółtego obszaru po przejściu całego pierścienia. Po wygranej automatycznie zmienia się poziom na trudniejszy (o ile taki istnieje).



Zdjęcie 2. Informacje dla użytkownika

4. Opis algorytmow i kodu

Kulka jest w programie reprezentowana przez strukturę o nazwie “Ball”:

```
15  struct Ball{
16      uint16_t x;
17      uint16_t y;
18      uint16_t prevX;
19      uint16_t prevY;
20  };
```

Posiada 4 zmienne - *x* i *y* to aktualne współrzędne kulki według których kulka jest rysowana na wyświetlaczu natomiast *prevX* i *prevY* to współrzędne z poprzedniej iteracji głównej pętli służące do zmazania poprzedniego rysunku kulki. Takie rozwiązanie zapewnia odpowiednią płynność ruchu kulki.

Program zaczyna się od skonfigurowania ustawień wyświetlacza LCD i komunikacji za pomocą SSP0. W celu porozumiewania się z akcelerometrem, wymagana jest komunikacja za pomocą 16 bitowej ramki. Pierwszy bit ramki to informacja, czy dane będą czytane czy wpisywane do akcelerometru. Kolejny bit informuje urządzenie o trybie pobierania informacji z kilku rejestrów naraz, z czego nie korzystamy. Następne sześć bitów ramki to adres rejestru z którego będziemy czytać/wpisywać dane. Ostatnie osiem bitów to dane, które mają zostać wpisane do rejestru, albo dane zwracane, w przypadku funkcji read. W funkcji acelerometr_setup zawiera się wpisywanie wszystkich niezbędnych informacji do rejestrów układu peryferyjnego, między innymi włączenie trybu measure mode, czyli rozpoczęcie wstawiania danych do odpowiednich rejestrów przez urządzenie.

Następnie następuje wpisanie odpowiednich opcji w rejestry akcelerometru(m. in. wejście w measure mode), a następnie konfiguracji przerwań od przycisków “KEY 1” i “KEY 2” (odpowiednio dla startu i wyboru poziomu trudności).

Główna pętla programu odczytuje odchylenia dla osi X i osi Y. Dla każdej osi dane są przechowywane w dwóch 8-bitowych rejestrach (mniej i bardziej znaczącym). Komunikacja jest skonfigurowana na ramki 16-bitowe dlatego po odczytaniu danych należy wyzerować pierwsze 8 bitów z 16-bitowej odpowiedzi (wartości zwróconej przez funkcję *SSP0_write(...)*) i następnie przesunąć wartości z bardziej znaczących rejestrów na bardziej znaczące miejsca w wyniku.

```
423             uint16_t x1 = SSP0_write( (1<<15) + 0x3200); //os x niskie bity
424             delay(10);
425             uint16_t y1 = SSP0_write( (1<<15) + 0x3400); //os y 1
426             delay(10);
427             uint16_t x2 = SSP0_write( (1<<15) + 0x3312); //os x wysokie bity
428             delay(10);
429             uint16_t y2 = SSP0_write( (1<<15) + 0x3512); //os y 2
430             x1 &= 0xFF; //zerowanie pierwszych 8 bitow
431             y1 &= 0xFF; //jw
432             x2 &= 0xFF;
433             y2 &= 0xFF;
434
435             short int wynikX = x1 + (x2 << 8);
436             short int wynikY = y1 + (y2 << 8);
```

Po ustaleniu odchylenia w obydwu osiach, kulka zostaje przesunięta o wartość podawaną przez akcelerometr, następnie sprawdzana są warunki zwycięstwa i porażki (dotykanie przez kulkę mety lub czerwonego koloru na ekranie), mazana

jest kulka w poprzednim położeniu, a na koniec rysowana jest kulka w nowym położeniu.
W razie wystąpienia przerwania od “KEY 2”, czyli wyboru poziomu trudności, rysowanie i przemieszczanie kulki zostaje wstrzymane aż do momentu naciśnięcia przycisku “KEY 1”.

5. Kontrola błędów

Nasz program jest dość prosty i nie wymaga rozbudowanej obsługi błędów. Jedynie od strony mechanicznej, to znaczy połączenia kabli, istnieje możliwość wystąpienia niechcianego zachowania. Upewniliśmy się jednak, że w takiej sytuacji, kulka przestaje się poruszać. Jako że akcelerometr jest cały czas w ruchu w czasie rozgrywki, taka sytuacja jest dość prawdopodobna, było to więc działanie wymagane.