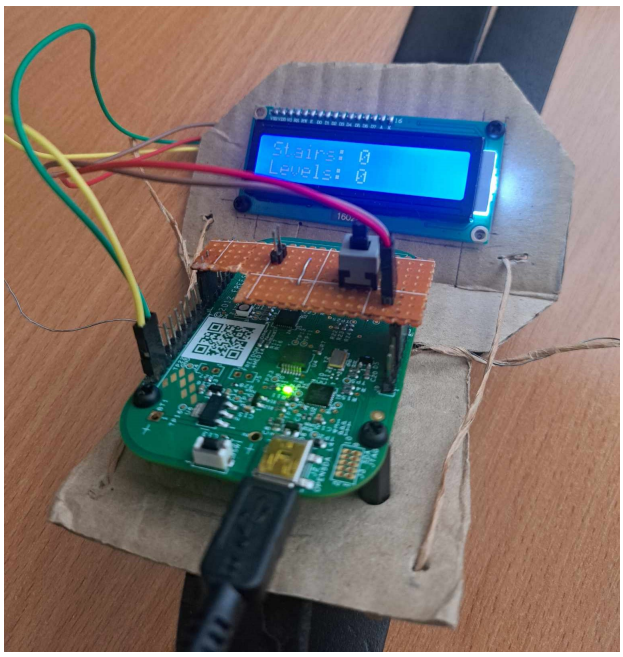
AGH

Piętromierz



Przedmiot: Systemy Mikroprocesorowe II

Autor: Mateusz Szpot
Prowadzący: mgr inż. Sebastian Koryciak
Koordynator przedmiotu: dr inż Mariusz Sokołowski

1 Opis projektu:

1.1 Wstęp:

Wykonany projekt ma działać jako piętromierz, który na podstawie zliczonych schodów pokonanych przez użytkownika, wyliczy liczbę pięter. Nie ma to jednak być precyzyjne narzędzie a coś, co pozwoli na ocenienie średniej aktywności fizycznej, tak jak to robią krokomierze zainstalowane w większości smartfonów lub te znajdujące się w smartwatchach.

Do jego zbudowania posłużyłem się płytką deweloperską z wbudowanym mikrokontrolerem **Freescale Kinetis KL05Z** oraz akcelerometrem **MMA8451Q**. Do wyświetlenia liczby pokonanych schodów oraz pięter użyty został wyświetlacz LCD 16x2 z płytką do komunikacji protokołem I^2C.

Kod projektu został napisany w języku C za pomocą programu Visual Studio Code. Jego obecna wersja znajduje się na GitHubie pod linkiem.

1.2 Opis działania:

Piętromierz posiada dwa tryby: Tryb Debugowy oraz Tryb Działania. Pierwszy z nich służy do pobierania danych z akcelerometru (przyspieszeń w każdej z osi) i wysyłania ich za pomocą protokołu UART. Drugi natomiast jak sama nazwa wskazuje odpowiada za liczenie pokonanych schodów, przeliczanie ich na piętra oraz wyświetlanie tych danych na ekranie. Przełączenie między trybami możliwe jest poprzez wpięcie zwory między dwa wystające goldpiny i restart mikrokontolera. Spowoduje to przejście z Trybu Działania do Trybu Debugowego.

Aby móc pozyskać dane w Trybie Debugowym należy podłączyć płytkę do laptopa lub komputera stacjonarnego i nacisnąć przycisk. Minimalna ilość pomiarów jaką urządzenie jest w stanie zrobić wynosi 100 (wynika ona z mechanizmu tłumiącego drgania zestyków zastosowanego w kodzie). Po zebraniu odpowiedniej ilości punktów pomiarowych można drugi raz nacisnąc przycisk. Urządzenie wykona wtedy ostatnie 100 pomiarów i zakończy transmisje. Dane podawane są w formacie częściowo zgodnym z rozszerzeniem .csv (jedyne co trzeba zrobić to zamienić '.' na ',' aby dane były interpretowane jako liczby zmiennoprzecinkowe).

Tryb Działania nie wymaga od użytkownika żadnych dodatkowych czynności. Po zamocowaniu urządzenia na pasku i przymocowaniu go do biodra będzie ono gotowe do działania.

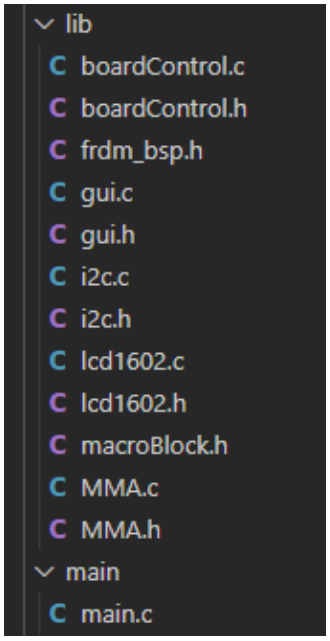
1.3 Organizacja kodu:

Kod został podzielony na kilka plików w celu lepszej organizacji i czytelności. Widok utworzonych plików w edytorze kodu widoczny jest na rysunku 1. Każdy z widocznych plików odpowiada za:

- ***i2c.h, i2c.c*** - komunikacja z modułami za pomocą protokołu I^2C,
- ***lcd1602.h, lcd1602.c*** - obsługa ekranu lcd 16x2,
- ***frdm_bsp.h*** - dostarcza funkcje i użyteczne makra (np. DELAY),

Wyżej wymienione pliki są bibliotekami, których autorami są prowadzący przedmiotu - mgr inż. Sebastian Koryciak oraz dr inż. Mariusz Sokołowski. Reszta plików została napisana przeze mnie.

- ***boardControl.h, boardControl.c*** - w tym pliku znajdują się funkcje wykonywane w pętli głównej programu, funkcja do ustawiania rejestrów mikrokontrolera oraz funkcje do obsługi przerwań,
- ***MMA.h, MMA.c*** - dostarcza funkcji, służących do konfiguracji, ustawienia przerwania oraz pozyskiwania danych z akcelerometru,
- ***gui.h, gui.c*** - funkcje służące do wyświetlania liczby pokonanych schodów i pięter na ekranie lcd,
- ***macroBlock.h*** - zawiera przydatne makra takie jak tresholdy, nazwy pinów czy wartości przydatne przy obsłudze licznika,
- ***main.c*** - główny plik kodu. W nim znajduje sie funkcja main z pętlą główną oraz handlery przerwań od Portu A oraz licznika PIT.



Rysunek 1: System plików

2 Zasada działania i opis algorytmu:

2.1 Zasada działania:

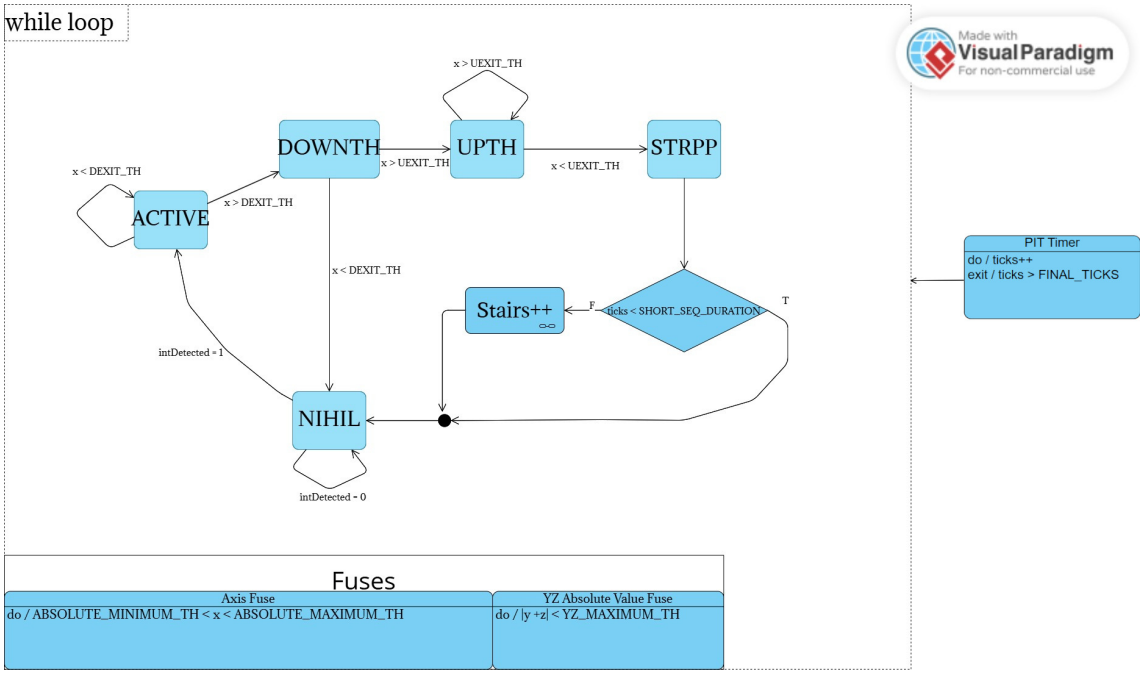
Zliczanie schodów działa na zasadzie wykrywania sekwencji na odpowiedniej osi. W przypadku tego projektu wybraną osią jest oś X, natomiast pozostałe osie posłużą do rozróżnienia czy osoba idzie po płaskiej powierzchni czy idzie po schodach.

Do wykrycia sekwencji należy ustawić dwa główne tresholdy. Po przekroczeniu pierwszego (ujemny)program czeka aż zostanie przekroczony drugi (dodatni). Gdy obydwa tresholdy zostaną osiągnięte, oznacza to, iż sekwencja została wykryta. Tak w dużym uproszczeniu działa program piętromierza. Zdarza się jednak iż takowa sekwencja zostaje wykryta, mimo że użytkownik nawet nie wszedł na schodek. Przyczyną mogą być między innymi ruchy lub uderzenia wywołujące podobną sekwencję (np. pojedynczy krok lub obrót). Dlatego oprócz samych tresholdów należy wprowadzić mechanizmy zabezpieczające przed tego typu przypadłościami.

Implementacja algorytmu została stworzona w oparciu o "Maszynę Stanów z Bezpiecznikami" (ang. Fused State Machine). Jest to prosta maszyna stanów wykonywana w pętli while, natomiast wspomniane bezpieczniki są warunkami, które natychmiastowo terminują wykonanie pętli. Jej diagram został przedstawiony na rysunku 2. Diagram został wykonany na stronie **Visual Paradigm** [1]

Stanem początkowym jest stan **NIHIL**. W tym stanie mikrokontroler czeka na pojawienie się przerwania od akcelerometru oraz wyświetla aktualną liczbę pokonanych schodów oraz pięter na ekranie. Samo przerwanie ustawione jest na

[1] https://online.visual-paradigm.com/pl/diagrams/



Rysunek 2: Diagram "Maszyny Stanów z Bezpiecznikami"

poziomie pierwszego tresholdu z pewnym debouncingiem (dzięki czemu mamy pewność, iż wartość która się pojawi nie nadeszła skokowo - eliminujemy tym samym wpływ nagłych uderzeń). Po wykryciu takowego przerwania zmieniana jest zmienna *intDetected* i maszyna przechodzi do kolejnego stanu.

Tym stanem jest stan **ACTIVE**. Stan ten sygnalizuje wejście w pętle zabezpieczoną bezpiecznikami. Wyjścei z niego jest możliwe albo przez przekroczenie tresholdu zwanego **DEXIT** lub po upłynięciu czasu ustawionego na timerze.

Stan **DOWNTH** (down treshold) pokazuje przekoczenie tresholda pomocniczego. Został on ustawiony aby wychwytywać sekwencje, które aktywowały maszynę, ale nie były w stanie przekroczyć drugiego tresholdu. Dzięki temu możliwa jest wcześniejsza terminacja pętli dzięki czemu nie trzeba czekać na przepełnienie licznika.

Wejście w stan **UPTH** (up treshold) sugeruje, że wykryta sekwencja jest na 85% tą właściwą. Jest to oznaka przekroczenia dwóch tresholdów. Wyjście z tego stanu następuje tylko do stanu kolejnego lub poprzez wywołanie jednego z bezpieczników.

Ostatni stan maszyny to **STRPP** (stair++). W nim następuje decyzja czy daną sekwencję liczyć jako dobrą (zwiększenie liczby schodków o 1) czy jako zbyt którtką (gdy czas trwania sekwencji jest mniejszy niż ustawiony czas **SHORT_SEQ_DURATION**). Po tym stanie niezależnie od wartości na osi X, pętla jest terminowana i maszyna znajduje się w stanie **NIHIL**, gdzie cały algorytm zaczyna się od początku.

Maszyna stanów posiada dwa bezpieczniki związane z wartościami przyspieszeń na osi oraz jeden bezpiecznik związany z timerem. Bezpiecznik **Axis Fuse** odpowiada za to, aby wartości występujące w trakcie trwania sekwencji mieściły się w odpowiednich widełkach. Zapobiega to sytuacjom, gdy urządzenie zostanie uderzone lub gdy użytkownik nagle podskoczy. W takich przypadkach przyspieszenie może być wyższe, przez co pętla zostanie opuszczona. Drugi bezpiecznik o

nazwie **YZ Absolute Value Fuse** sprawdza czy wartość bezwzględna sumy pozostałych osi (tj. osi Y i Z) nie przekracza wartości **YZ_MAXIMUM_TH**. Ma to na celu zgrubne wykrycie czy sekwencja na osi X pochodzi od wchodzenia po schodach, czy od kroków na płaskiej powierzchni (osie te «w szczególności oś Y» są bardziej aktywne przy wykonywaniu kroków). Jest to niestety metoda zgrubna, która czasami może niwelować chcianą sekwencję. Jednak do tak prostego projektu jej zastosowanie w zupełności wystarczy.

Cała pętla jest dodatkowo "nadzorowana" przez licznik **PIT Timer**. Ma on dwa zadania. Po pierwsze pilnuje aby sekwencja nie była zbyt długa tj. aby nie było dużej równicy w czasie przekraczania obydwu tresholdów. Po drugie (co zostało wspomniane przy opisie stanu **STRPP**) umożliwia sprawdzenie czy finalnie wykryta sekwencja nie jest zbyt krótka.

2.2 Przykłady sekwencji:

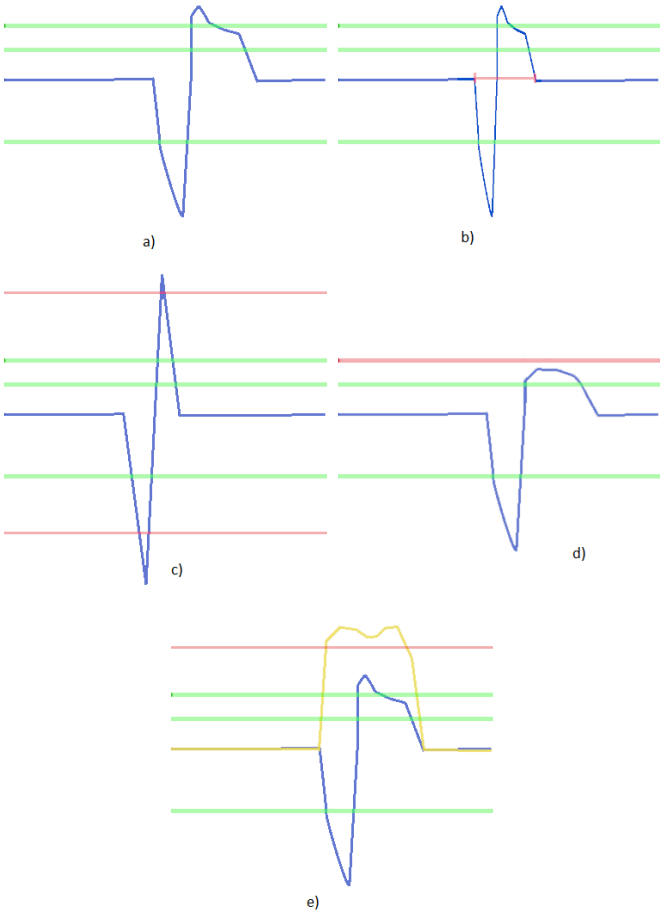
Przykładowe sekwencje zostały przedstawione na rysunku 3.

W przypadku przebiegu a) obydwa główne tresholdy jak i treshold pomocniczy zostały przekroczone. Jest on także odpowiedniej długości, dlatego ten przebieg jest uznawany za poprawny.

Sekwencja przedstawiona na przykładzie b) również przekroczyła wszystkie tresholdy, jednak z powodu krótkiego czasu trwania została odrzucona.

Przebieg c) również jest za krótki, jednak zostałby on odrzucony wcześniej, gdyż przekroczył wartości maksymalne (zaznaczone czerwonymi liniami).

W podpunkcie d) widać natomiast, iż sekwencja została wyzwolona przerwaniem, jednak osiągnęła tylko treshold pomocniczy. Dlatego zostaje ona zignorowana.



Rysunek 3: Przykładowe sekwencje

Ostatni podpunkt pokazuje sytuację, gdy mimo tego, że na osi X wartości sygnałów się zgadzają, to na osiach YZ przekraczają one pewną wartość. Jest to sytuacja opisana we wcześniejszym podpunkcie przy bezpieczniku **YZ_MAXIMUM_TH**.

3 Możliwości rozwoju:

Projekt ten jest efektem miesięcznej pracy. W przyszłości niektóre rzeczy mogłyby zostać zoptymalizowane. Należą do nich m.in.:

- Optymalizacja zużycia energii urządzenia,
- Optymalizacja algorytmu,
- Minimalizacja urządzenia,
- Ulepszenie algorytmu rozróżniania wchodzenia po schodach a wykonywania kroków.