PATIENT MONITORING

DEVICE

Dokumentacja do projektu.

Version 1.0

Bartłomiej Cerek

Arkadiusz Chudy

Spis treści

[Abstrakt 3](#_Toc483776195)

[1. Lower layer 3](#_Toc483776196)

[Akceletometr 3](#_Toc483776197)

[Czujnik pulsu/przepływu krwii 3](#_Toc483776198)

[2. Upper layer 4](#_Toc483776199)

[Baza danych 4](#_Toc483776200)

[Metoda do wykrywania upadków 5](#_Toc483776201)

[Odbieranie danych z akcelerometru – UART 6](#_Toc483776202)

[3. BLE 7](#_Toc483776203)

[Serwisy i charakterystyki 7](#_Toc483776204)

[4. Aplikacja mobilna 7](#_Toc483776205)

[Architektura 7](#_Toc483776206)

[Sposób pobierania danych z bazy 7](#_Toc483776207)

[Monitorowanie dostępu do internetu 8](#_Toc483776208)

[5. Podsumowanie 8](#_Toc483776209)

# Abstrakt

Przedstawiony dokument jest wstępna dokumentacją do projektu „Patient Monitoring Device”. Projekt jest tworzony w ramach konkursu o tematyce IoT organizowanego przez firmę Nordic Semiconductor.

# 1. Lower layer

## Akceletometr

Wykorzystywany jest moduł LSM303D, komunikacja z nim zachodzi przez interfejs I2C (TWI). Do prawidłowego działania konieczne są 4 połączenia: Vin (3,3V), GND oraz SCL i SDA. Dwa ostatnie należy wpiąć tak jak jest to ustawione w funkcji konfiguracyjnej TWI. Moduł musi być zainicjowany przez ustawienie 'jedynek' w rejestrach odpowiadających za odblokowanie akcelerometrów w kierunkach X,Y i Z.

Dane pozyskane z konwertera ADC należy złożyć (wartość w każdym kierunku podawana jest jako dwa bajty high i low), a następnie przeliczyć na procentową część przyspieszenia ziemskiego wdg wzoru x = a\_x \* 2.0 / 32678.0.

Na ich podstawie możemy wnioskować o pozycji pacjenta.

## Czujnik pulsu/przepływu krwii

Wykorzystywany jest moduł MAX30105 - czujnik cząsteczek, komunikacja z nim zachodzi przez interfejs I2C (TWI). Do prawidłowego działania konieczne są 4 połączenia: Vin (3,3V), GND oraz SCL i SDA. Dwa ostatnie należy wpiąć tak jak jest to ustawione w funkcji konfiguracyjnej TWI. Czujnik wyposażony jest w diodę emitującą promieniowanie IR, oraz możliwość pomiaru natężenia odbitej części promieniowania. Ponieważ krew natlenowana pochłania większą ilość promieniowania podczerwonego niż nie natlenowana na podstawie zmian odczytu ADC modułu wnioskować można o pulsie i przepływie krwi pacjenta. Ważną kwestią jest, że moduł musi być przytwierdzony bezpośrednio do skóry (niekoniecznie przy dużym naczyniu krwionośnym), a jego nacisk musi być STAŁY. Najlepiej uzyskać ten efekt stosując jakiegoś rodzaju elastyczny materiał (nawet gumkę recepturkę).

Jeżeli przebieg zmian wartości uzyskiwanej z ADC pokażemy na wykresie to ilość pików w ciągu 60 sekund (lub proporcja z krótszego czasu) będzie stanowiła puls pacjenta. Dane odbierane są w paczkach 4 bajtowych (unsigned int 16), z KOLEJKI w urządzeniu. Aby ograniczyć błędy i zmniejszyć ilość danych ustawiono urządzenie na uśrednianie 8 wartości, oznacza to, że do kolejki przesyłamy

Układ scalony kolejkuje wartości które odczytuje się przez wielokrotne odczytanie jednego rejestru. Przy wywołaniu funkcji odczytującej wartość HRS tak naprawdę odczytywana jest cała kolejka(w celu zwolnienia miejsca) i wybierana ostatnia próbka. Tylko w ten sposób można odczytywać w dowolnej chwili ostatnią wartość.

# 2. Upper layer

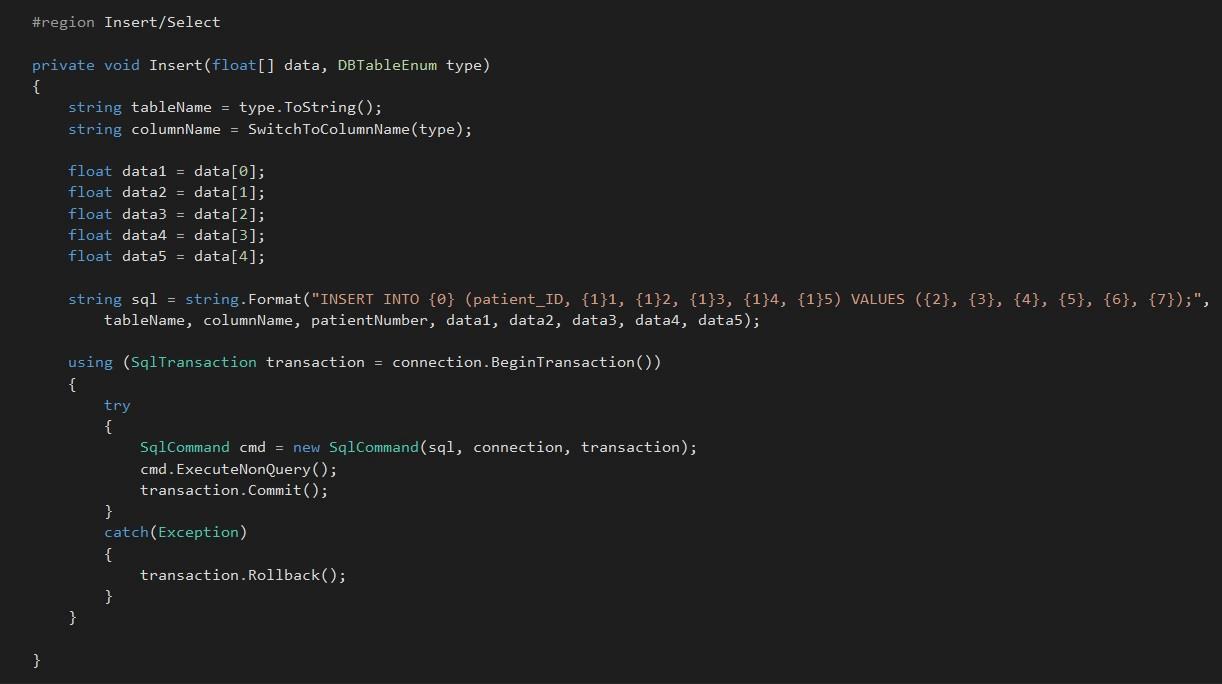
## Baza danych

W projekcie wykorzystujemy zaprojektowaną pod nasze wymagania bazę danych. Baza (chwilowo testowa – w końcowej fazie projektu przejdziemy na wersję finalną) znajduje się na własnym serwerze w prywatnej sieci. Serwer bazodanowy opiera się na MSSQL 2014, połączenie z bazą możliwe jest z wykorzystaniem VPNa. Baza zawiera 6 tabeli : jedna główna (Patients) oraz 5 tabeli pomiarowych do których zbieramy dane z pomiarów (HeartRate, Temperatura, 3 x Akcelerometr – na każdą z osi).

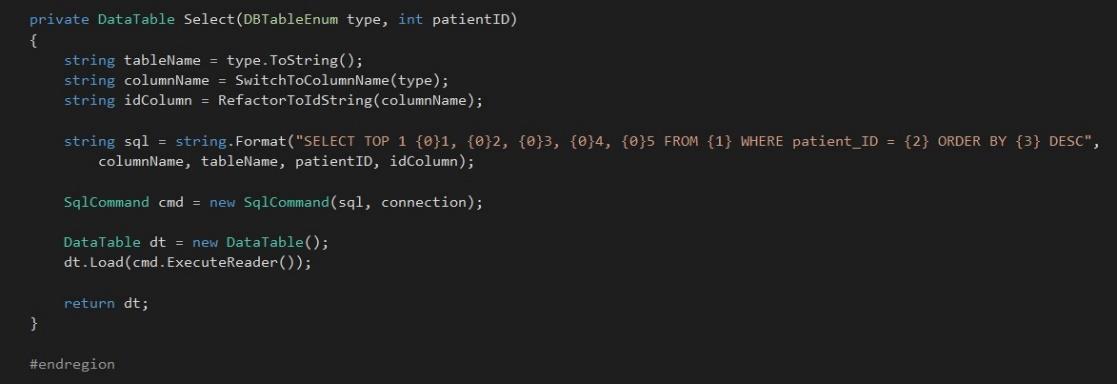
Każda z tabeli pomiarowych posiada 7 kolumn – jedna to PrimaryKey dla danej paczki pomiarowej, druga to ForeignKey mówiący o ID badanego pacjenta, pozostałe 5 kolumn przeznaczone jest na zachowanie wartości paczki danych. Typy kolumn pod paczkę danych to „float” – wartości otrzymywane z sensorów przerabiamy na dane zmiennoprzecinkowe o pojedynczej precyzji używając do tego programu napisanego w LabVIEW.

W projekcie wysyłamy dane co 2.5 sekundy w paczkach 5 pomiarów. Taki efekt spowodowany jest tym, aby ograniczyć ruch sieciowy. Ustalone wartości są wartością konfigurowalną - w dalszych aspektach projektu mogą ulec zmianie.

Do połączenia się z bazą oraz wysyłania odpowiednich kwerend używamy wcześniej napisanej przez nas biblioteki. Biblioteka „NordicDatabaseDLL” została napisana w VisualStudio w technologii .NET przy użyciu języka C#. Dzięki zaimplementowanym w niej funkcjom możemy otwierać i zamykać połaczenie z bazą oraz wykonywać SELECTy i INSERTy na odpowiednich tabelach, podając tylko dane wejściowe w postaci tabeli floatów o długości 5.

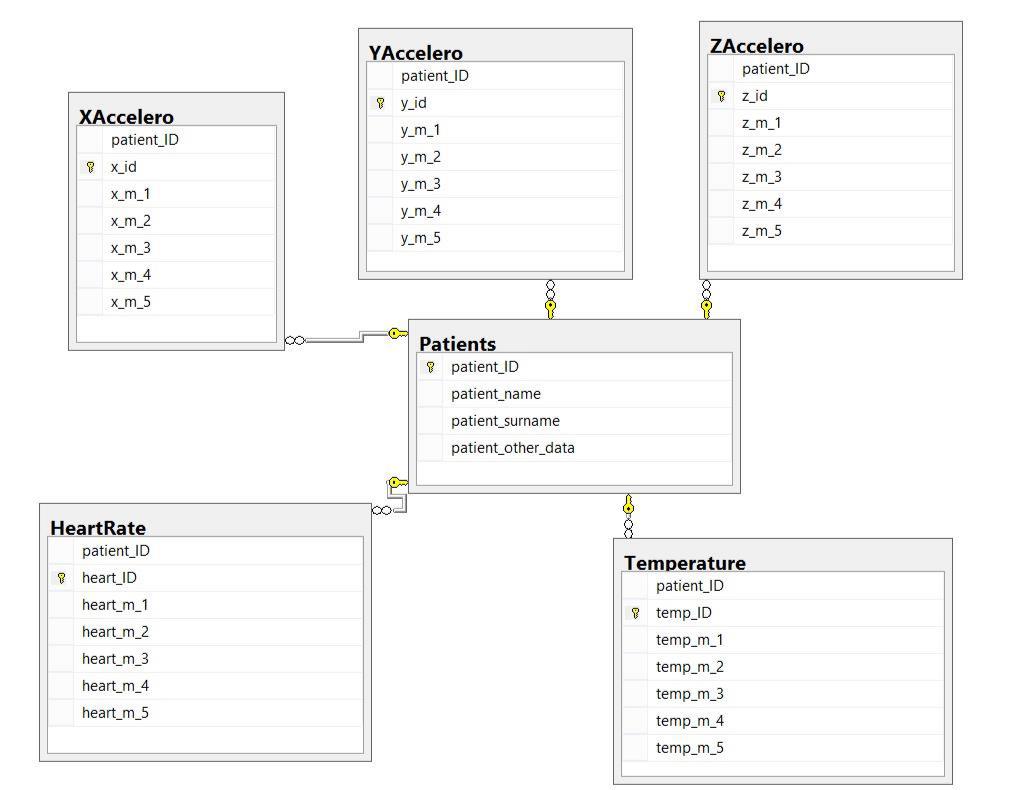


*Rysunek 1 Metoda dla INSERTA z biblioteki*



*Rysunek 2 Metoda dla SELECTA z biblioteki*

INSERT do bazy wykonywany jest w momencie zebrania 5 kolejnych pomiarów. Taka paczka, wraz z ID pacjenta dla którego wykonywany jest pomiar, wysyłana jest do bazy po wcześniejszym przygotowaniu kwerendy (SQL).



*Rysunek 3 Schemat testowej bazy*

## Metoda do wykrywania upadków

W bibliotece została zaimplementowana specjalistyczna klasa **FallChecker** odpowiedzialna za wykrywanie upadku pacjenta. Klasa posiada metodę, która pobiera podane argumenty :

* tabele X, Y i Z akcelerometru
* różnicę wartości przy której wykrywany jest upadek
* ilość próbek z których liczona ma być różnica do weryfikacji

Algorytm do rejestracji składu się z kilku etapów :

1. Jeżeli podane długości tabeli X, Y, Z są mniejsze od zadanej wartości próbek, to zwracana jest wartość **FALSE** **–** **upadek nie nastąpił**.
2. Jeżeli podane długości są odpowiednie (większe od zadanej wartości próbek) następują kolejne obliczenia.
3. Algorytm analizuje wartości znajdujące się w każdej z tabel zaczynając od indeksu 0. Analizowane jest tyle próbek ile zostało podane w zadanej wartości próbek w funkcji.
4. Liczona jest delta między każdą z wartości bezwzględnych z próbek. Jeżeli któraś z obliczonych delt jest większe od tej zadanej – algorytm zwraca wartość **TRUE – upadek nastąpił.**
5. Po analizie danej ilości próbek (od indeksu 0) algorytm przeskakuje do kolejnego indeksu (jeżeli jest to możliwe) i wykonuje dalsze obliczenia. Akcja ta powtarzana jest dla całej tablicy.

Metoda do wykrywania upadków zaimplementowana jest tak w aplikacji na PC jak i aplikacji mobilnej. Obydwu aplikacjom przy wykryciu upadku pacjenta towarzyszy zmiana koloru interfejsu na odcień czerwieni. Gdy upadek nie jest już wykrywany – kolor powraca do wyjściowego.

## Odbieranie danych z akcelerometru – UART

Do odbierania danych po magistrali UART wykorzystaliśmy środowisko LabVIEW oraz bibliotekę NI-VISA. Program odbiera dane wysyłane przez nRF52 jako kolejne stringi. Następnie program dokonuje obróbki – dostosowuje stringa do późniejszej zamiany na 32-bitową liczbę zmiennoprzecinkową o pojedynczej precyzji (w zamyśle – możliwa implementacja z podwójna precyzją).

W pętli głównej programu znajduje się klaster tabel (typedef), do których dane z akcelerometru (już po zamianie na „float”) są wrzucane. Wszelkie błędy (błędny string, obcięte dane, błąd konwersji) są

eliminowane. Gdy taki error się pojawia – brana jest poprzednia dana z tabeli, tak by zachować ciągłość akwizycji.

Program jest także przystosowany do wyświetlania danych z 3 osi akcelerometru na jednym, wspólnym wykresie. Wartości z tabel przy każdym pomiarze są wyciągane i umieszczane na plocie. Pozwala nam to nie tylko zbierać i przetwarzać dane, ale również je wizualizować.

W projekcie znajduję się także subVI nazwany FGV. Jest to nasza funckjonalna zmienna globalna, która (już przygotowana do odbierana wszystkich danych, łącznie z heartRatem) wykorzystując predefiniowany klaster danych oraz niezainicjowany rejestr przesuwny, wkłada dane do tabel klastra i daje output w zależności od rodzaju przychodzącej do niej danej (output -> tabela XAccelero jeżeli przychodząca dana to wartość z osi X akcelerometru).

Na wyjściu wcześniej opisanej FGV powstanie kolejna funkcja sprawdzająca, czy długość tabeli jest równa 5. Jeżeli tak -> rozpoczynamy komunikacje bazodanową. Jej implementacja (z wykorzystaniem biblioteki NordicDatabaseDLL) powstanie z wykorzystaniem metod .NET w środowisku LabVIEW.

Przedstawiony sposób odbierania danych poprzez UART jest tylko sposobem testowania oprogramowania. Domyślnie wszystko zostanie przeniesione na BLE. Do komunikacji po bluetoothie wykorzystanie zostanie bluetooth dongle (zbieranie danych po COM porcie) oraz BLE Toolkit w

środowisku LabVIEW.

# 3. BLE

## Serwisy i charakterystyki

Wszystkie wykorzystywane serwisy, charakterystyki i UUID są zdefiniowane i obsłużone w plikach our\_service.h i our\_service.c . Wszystkie charakterystyki są typu notify aby ułatwić ich odczyt.

Serwis o UUID ACC0 przechowuje dane odbierane z akcelerometru. Posiada on 3 charakterystyki

1ACC, 2ACC i 3ACC odpowiadające za kolejno osie X, Y i Z.

Serwis o UUID EE00 przechowuje dane odbierane z czujnika cząsteczek MAX30105. Posiada on charakterystykę EE10 w której przechowywane są ostatnie wartości wyciągnięte z kolejki modułu.

# 4. Aplikacja mobilna

## Architektura

W ramach projektu stworzona została aplikacja mobilna osadzona na systemie Android. Aplikacja została stworzona w programie Visual Studio wykorzystując framework Xamarin oraz język C#.

Aplikacja składa się z 3 activities :

* **WebConnection** – activity będą formą ekranu przejściowego, jeżeli urządzenie nie jest połączone z internetem, to ekran przełącza się na główny View tego activity
* **InfoActivity** – ekran startowy aplikacji, użytkownik może wybierać badanego pacjenta z ich listy, po wyborze na ekranie pojawiają się informacje o danej osobie, po kliknięciu określonego guzika użytkownik przechodzi do ekranu głównego aplikacji
* **MainActivity** – activity odpowiedzialne za wykreślanie wykresów akcelerometru o hearRate’a na ekranie urządzenia, użytkownik ma możliwość przełączania się pomiędzy wykresami w czasie rzeczywistym

Główne activity w programie korzysta z dwóch klas – **HeartPlot** oraz **AcceleroPlot**. Te klasy są odpowiedzialne za akwizycję danych z bazy dla danego typu wykresu oraz dodawanie ich do swojego pola danych wykresu. W głównym programie do wykresu przypisywane są dane z jednej z tych klas i odświeżane w czasie rzeczywistym.

Użytkownik chcąc zmienić wyświetlanie danego wykresu może uczynić to klikając w guzik **CHANGE CHART** znajdujący się w górnej części ekranu. Nie występuje tutaj żadne przekierowanie do innego acitivity, gdyż tylko dane wykresu oraz granice jego osi Y są podmienianie na wartości z danej klasy Plot.

## Sposób pobierania danych z bazy

W aplikacji zostały zaimplementowane struktury, które na bieżąco badają ilość rekordów znajdujących się w bazie danych. Gdy w bazie wystąpi INSERT, ostatnia wartość indeksu pomiaru weryfikowana jest z zapisana ówcześnie wartością w programie. Jeżeli nowo otrzymana wartość jest większa – następuje operacja SELECT i dane są pobierane a następnie obrabiane i dodawane do wykresu.

Jeżeli aplikacja zostanie włączona, lecz urządzenie (nRF 52) nie jest włączone, to ekran główny aplikacji zostanie niezainicjowanym wykresem. Dopiero w momencie zmian ilości rekordów w bazie dochodzi do dalszych akcji.

## Monitorowanie dostępu do internetu

W aplikacji zostały także zaimplementowane struktury, które na bieżąco monitorują stan sieciowy urządzenia.

Jeżeli przy włączeniu aplikacji urządzenie nie miało dostępu do internetu, to użytkownik zostanie poproszony o jego włączenie, gdyż jest on niezbędny do działania kolejnych komponentów.

W momencie gdy użytkownik znajduje się na głównym ekranie aplikacji i dostęp do sieci zostanie utracony – występuje stosowny komunikat i powrót do ekranu głównego WebConnection activity. Gdy dostęp zostanie przywrócony następuje przeskok – omijane jest InfoActivity, a użytkownik ponownie może śledzić wielkości parametrów fizjologicznych danego pacjenta.

# 5. Podsumowanie

Dokumentacja zawiera idee, które mogą ulec zmianie w trakcie rozwoju pracy. Wszelkie dane w postaci kodów źródłowych znajdują się na naszym repozytorium na githubie.