PATIENT MONITORING DEVICE

Dokumentacja do projektu.

Version 1.0

Bartłomiej Cerek
Arkadiusz Chudy

Spis treści

1. 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.

2. 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.

3. 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.

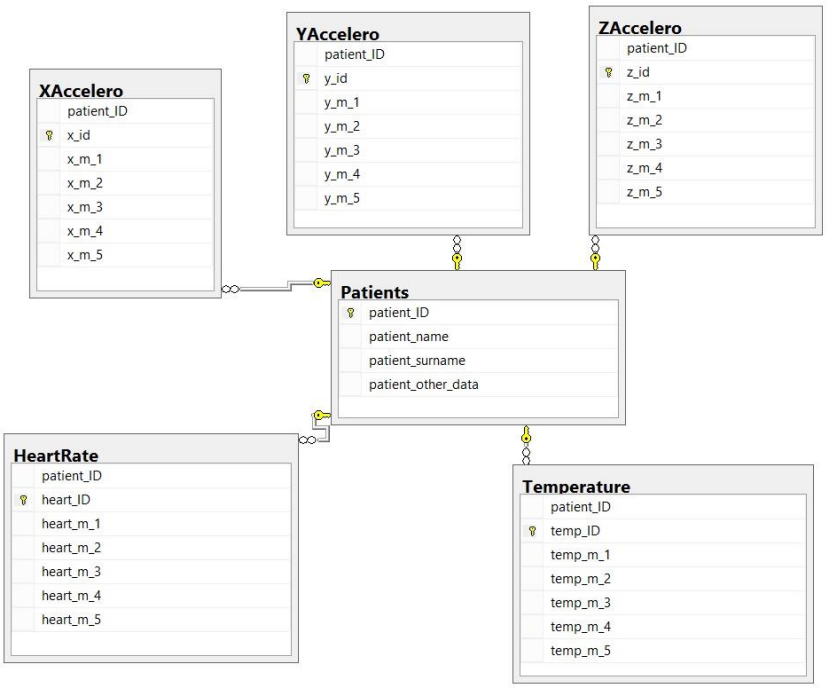
```
#region Insert/Select

private void Insert(float[] data, DBTableEnum type)
{
    string tableName = type.ToString();
    string columnName = SwitchToColumnName(type);

    float data1 = data[0];
    float data2 = data[1];
    float data3 = data[2];
    float data4 = data[3];
    float data5 = data[4];

    string sql = string.Format("INSERT INTO {0} (patient_ID, {1}1, {1}2, {1}3, {1}4, {1}5) VALUES ({2}, {3}, {4}, {5}, {6}, {7});",
        tableName, columnName, patientNumber, data1, data2, data3, data4, data5);

    using (SqlTransaction transaction = connection.BeginTransaction())
    {
        try
        {
            SqlCommand cmd = new SqlCommand(sql, connection, transaction);
            cmd.ExecuteNonQuery();
            transaction.Commit();
        }
        catch(Exception)
        {
            transaction.Rollback();
        }
    }

}
```

Rysunek 1 Metoda dla INSERTA z biblioteki

```
private DataTable Select(DBTableEnum type, int patientID)
{
    string tableName = type.ToString();
    string columnName = SwitchToColumnName(type);
    string idColumn = RefactorToIdString(columnName);

    string sql = string.Format("SELECT TOP 1 {0}1, {0}2, {0}3, {0}4, {0}5 FROM {1} WHERE patient_ID = {2} ORDER BY {3} DESC",
        columnName, tableName, patientID, idColumn);

    SqlCommand cmd = new SqlCommand(sql, connection);

    DataTable dt = new DataTable();
    dt.Load(cmd.ExecuteReader());

    return dt;
}

#endregion
```

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

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.

4. 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.

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.