Katedra Telekomunikacji Multimedialnej i Elektroniki

Wydział Elektroniki i Telekomunikacji

Inżynierska Praca dyplomowa

**Aplikacja mobilna sterująca portami minikomputera Raspberry Pi**

Mikołaj Ignaszak

Promotor dr inż. Adam Kaliszan

Poznań 2015

**1. Wstęp**

Dynamicznie rozwijającym się segmentem rynku urządzeń elektronicznych jest budowa tzw. "inteligentnych domów". Rozmieszczanie różnego rodzaju czujników temperatury, ruchu itp. pozwala na monitorowanie

**2. Spis treści**

[**1. Wstęp** 2](#_Toc409370619)

[**2. Spis treści** 3](#_Toc409370620)

[**4. Przegląd wykorzystanego sprzętu oraz technologii** 4](#_Toc409370621)

[**4.1 Opis minikomputera Raspberry Pi** 4](#_Toc409370622)

[**4.2 Wykorzystane czujniki** 4](#_Toc409370623)

[**4.2.1 Czujnik ruchu PIR HC-SR501** 4](#_Toc409370624)

[**4.2.2 Czujnik temperatury i wilgotności powietrza DHT11** 4](#_Toc409370625)

[**4.2 System zarządzania relacyjnymi bazami danych MySQL** 4](#_Toc409370626)

[**4.3. Platforma .NET do programowania aplikacji mobilnych na system Windows Phone 8.1** 4](#_Toc409370627)

[**5. Projektowanie rozwiązanie** 5](#_Toc409370628)

[**5.1 Projekt układu łączącego Raspberry Pi z czujnikami.** 5](#_Toc409370629)

[**5.2 Projekt bazy danych** 5](#_Toc409370630)

[**5.3 Projekt aplikacji pobierającej i wyświetlającej dane z kontrolki** 5](#_Toc409370631)

**4. Przegląd wykorzystanego sprzętu oraz technologii**

**4.1 Opis minikomputera Raspberry Pi**

**4.2 Wykorzystane czujniki**

**4.2.1 Czujnik ruchu PIR HC-SR501**

**4.2.2 Czujnik temperatury i wilgotności powietrza DHT11**

**4.2 System zarządzania relacyjnymi bazami danych MySQL**

**4.3 Serwis Bitnami.com**

**4.3. Platforma .NET do programowania aplikacji mobilnych na system Windows Phone 8.1**

**5. Projektowanie rozwiązanie**

**5.1 Projekt układu łączącego Raspberry Pi z czujnikami.**

Pierwszym etapem pracy było zaprojektowanie układu łączącego opisane w poprzednim rozdziale czujniki z GPIO w Raspberry Pi. Napięcie na wyjściu czujnika ruchu PIR HC-SR501

wynosi 3.3 V, więc bez problemu pasuje do portu GPIO 23 w Raspberry PI. Styk zasilania został podłączony do pinu nr 2. Czujnik temperatury i wilgotności powietrza został podłączony zgodnie z dokumentacją [nr]. Pin układu z zasilanie 3.3 volta (pin nr 1 na płytce). Pin drugi czujnika został połączony z portem GPIO nr 4 minikomputera z równolegle podłączonym opornikiem o wartości rezystancji 5kOhm do zasilania. Napięcie odniesienia, czyli masa, został pobrany z portu nr 6 w Raspberry PI. Do tego wejścia zostały podłączone Piny nr 3 i 4 odpowiednio w HC-SR501 i DHT11. Dodatkowo do GPIO 3 została odpowiednio podłączona zielona dioda LED, która będzie informować o zarejestrowanym ruchu przez czujnik. Schemat układu jest przedstawiony poniżej.

Rys. Schemat układu

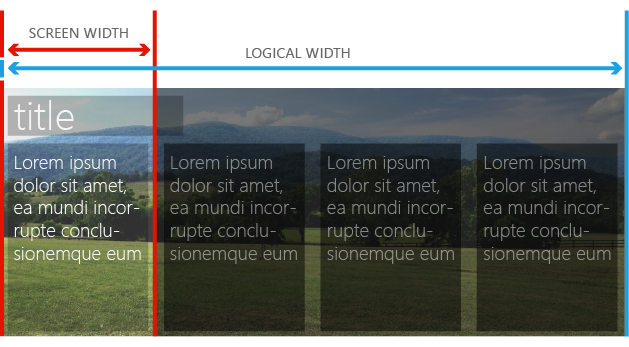
**5.2 Projekt bazy danych**

Raspberry Pi pomiary z czujników zapisuje w bazie danych. Dzięki temu wynika będą widoczne z Internetu i zostaną wyświetlone w aplikacji mobilnej. Do poprawnego działania programu nie jest potrzebna baza o rozbudowanej strukturze, było by to wręcz nie wskazane. Dlatego opiera się ona na trzech tabelach:

* przechowującej informacje o dacie wykrytego ruchu, przez czujnik alarmowy
* przechowującej wartość wilgotności powietrza i datę odczytu z czujnika
* przechowującej wartość temperatury powietrza i datę odczytu z czujnika

**5.3 Projekt aplikacji pobierającej i wyświetlającej dane z kontrolki**

Informacje zapisane w bazie danych mają być wyświetlane za pomocą aplikacji na telefonie komórkowym z zainstalowanym systemem Windows Phone. Sam program będzie w stylu aplikacji panoramicznej posiadającej trzy główne ekrany z danymi. Na pierwszym znajdować się będzie wykres temperatury powietrzu z określonego okresu, data i wartość ostatniego pomiaru. Na drugim ekranie będą podane takie same informacje lecz dla danych wilgotności powietrza. Trzeci ekran będzie zawierał listę momentów, w których czujnik ruchu zarejestrował aktywność w pokoju.

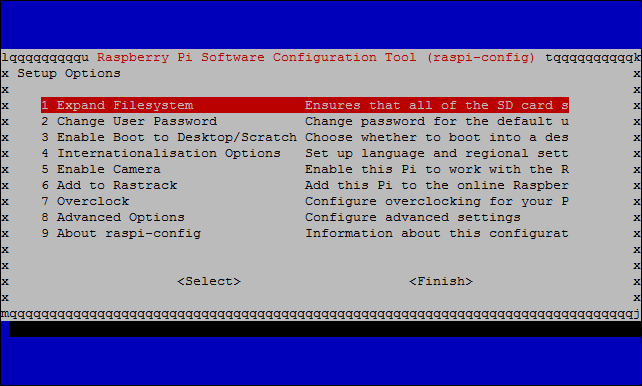


Rys. Przykładowa aplikacja panoramiczna (z dokumentacji środowiska .NET dla WP8 [nr] )

**6. Realizacja projektu**

**6.1 Przygotowanie komputera Raspberry Pi**

Aby rozpocząć pracę z minikomputerem Raspberry Pi w pierwszej kolejności należało zainstalować system operacyjny. Wybór padł na Rasbiana, który jest zmodyfikowaną i przystosowaną do pracy na Raspberry Pi wersją dystrybucji Linuxa o nazwie Debian. Sama instalacja polega na wgraniu obrazu systemu na kartę pamięci microSD. Po włączeniu komputera podłączonego kablem HDMI do system wyświetlił stronę konfiguracji.



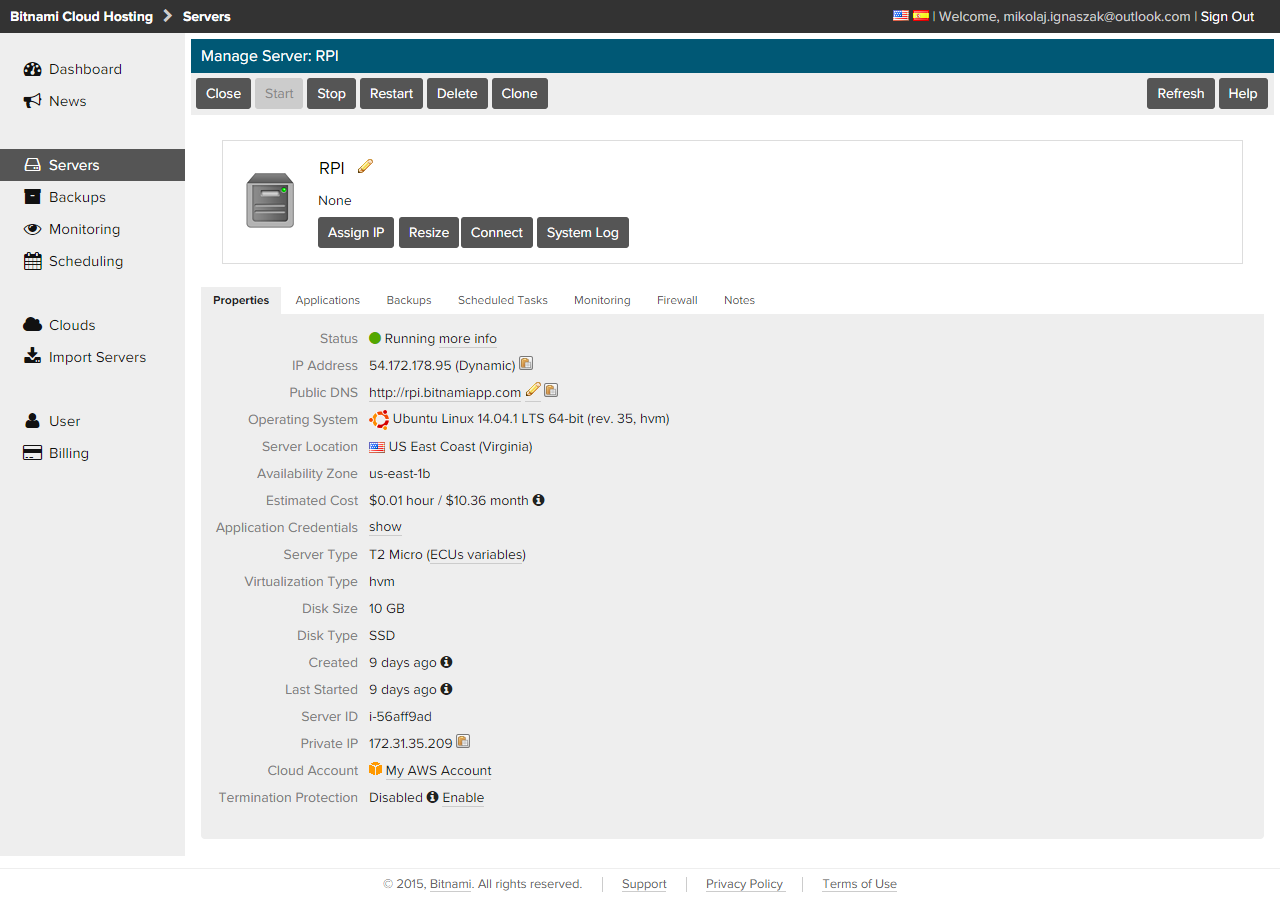
Rys. Konfiguracja systemu Raspbian

Oprócz podstawowych ustawień (m. in. hasło użytkownika, język) został włączony zdalny dostęp przez protokół SSH. Dzięki temu użytkownik może, za pomocą wiersza poleceń, wprowadzać zmiany na Raspberry Pi, z innego komputera znajdującego się w tej samej sieci.

Dodatkowo włączony jest protokół SCP do bezpiecznej wymiany plików.

**6.2 Utworzenie bazy danych na zewnętrznym serwerze**

Przed rozpoczęciem prac nad zestawieniem układu elektronicznego łączącego zewnętrzne czujniki z Raspberry Pi została utworzona baza danych. Pierwszym zadaniem było wybranie serwera, na którym miałyby znajdować się dane. Wybór padł na serwis bitnami.com, opisany w rozdziale 4.3. Serwer jest tworzony w łatwy sposób. Posiada on 1000 MB pamięci RAM oraz 10 GB pamięci dyskowej. Pracuje pod kontrolą systemu Ubunti Linux w wersji 14.04 LTS 64-bit. Automatycznie instalowane jest na nim oprogramowanie Apache'a v. 2.4.10 (serwer HTTP), serwer baz danych MySQL v. 5.5.40 oraz technologia PHP-FPM (służąca do interpretowania skryptów napisanych w języku PHP). Adres internetowy serwera to http://rpi.bitnamiapp.com. Co prawda, dana usługa jest płatna, lecz dzięki skorzystaniu z paczki "Student Developer Pack" serwisu Github, oraz bezpłatnej wersji AWS (Amazon Web Services) na dwanaście miesięcy możliwe jest korzystanie z serwera przez rok bez ponoszenia kosztów.



Rys. Wygląd panelu administracyjnego serwera

Wraz z oprogramowaniem MySQL na serwerze instalowane jest narzędzie phpMyAdmin ułatwiające administrowanie bazą danych MySQL. Za jego pomocą utworzona została nowa baza o nazwie RPI. W niej będą się znajdować tabele przechowujące dane o odczytach z czujników. Pierwszą z nich nazywa się "humidity" i przechowuje dane o pomiarach wilgotności powietrza z czujnika DHT11. Zapytanie ją tworzące wygląda następująco:

CREATE TABLE `humidity` (

`ID` int(11) NOT NULL AUTO\_INCREMENT,

`DATE` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

`VALUE` varchar(255) CHARACTER SET utf8 COLLATE utf8\_polish\_ci NOT NULL,

`UNIT` varchar(255) CHARACTER SET utf8 COLLATE utf8\_polish\_ci NOT NULL DEFAULT '%'

);

Opis kolumn:

* ID - kolumna identyfikująca wpis w tabeli, posiadająca auto inkrementacją
* DATE - kolumna typu timestamp wskazująca datę odczytu, jako domyślną wartość przyjmuje moment dodawania wpisu
* VALUE - kolumna typu varchar, zapisująca wartość wilgotności powietrza odczytaną z czujnika
* UNIT - kolumna typu varchar przechowująca symbol jednostki zmierzonej wartości, domyślnie przyjmuje znak '%'

Kolejną tworzona jest tabela o nazwie "move\_detection" przechowująca informacje o wykrytym ruchu przez czujnik.

CREATE TABLE `move\_detection` (

`ID` int(11) NOT NULL AUTO\_INCREMENT,

`DATE` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP

)

Opis kolumn:

* ID - kolumna identyfikująca wpis w tabeli, posiadająca auto inkrementacją
* DATE - kolumna typu timestamp wskazująca datę wykrytego ruchu, jako domyślną wartość przyjmuje moment dodawania wpisu

Ostatnia tabela przechowuje dane po temperaturze powietrza i nazywa się "temperature".

CREATE TABLE `temperature` (

`ID` int(11) NOT NULL AUTO\_INCREMENT,

`DATE` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

`VALUE` varchar(255) CHARACTER SET utf8 COLLATE utf8\_polish\_ci NOT NULL,

`UNIT` varchar(255) CHARACTER SET utf8 COLLATE utf8\_polish\_ci NOT NULL DEFAULT 'C' )

Opis kolumn:

* ID - kolumna identyfikująca wpis w tabeli, posiadająca auto inkrementacją
* DATE - kolumna typu timestamp wskazująca datę odczytu, jako domyślną wartość przyjmuje moment dodawania wpisu
* VALUE - kolumna typu varchar, zapisująca wartość temperatury powietrza odczytaną z czujnika
* UNIT - kolumna typu varchar przechowująca symbol jednostki zmierzonej wartości, domyślnie przyjmuje znak 'C'

**6.1 Układ kontrolki alarmowej i komunikacja czujników z Raspberry Pi i bazą danych**

Kolejnym krokiem w realizacji projektu było zestawienie układu elektronicznego łączącego czujniki z Raspberry PI. Obwód przedstawiony w punkcie 5.1 został złożony na płytce stykowej i podłączony do odpowiednich portów Raspberry Pi.

Rys. Zdjęcie zmontowanego prototypu

Aby była możliwa komunikacja i sterowanie zewnętrznymi czujnikami konieczne było utworzenie odpowiedniego skryptu w jednym z języków programowania. Wybór padł na język python, który w wersji 2.7.3 jest zawarty w systemie Rasbian. Sama technologia umożliwia m. in. komunikacje z bazą danych MySQL, a rozbudowana o moduł RPi.GPIO umożliwia sterowanie portami Raspberry Pi.

W celu obsługi czujnika DHT11 wykorzystany został kod zamieszczony przez firmę Adafruit w serwisie Github. Przekształca odczyty z urządzenia tak aby były zrozumiałe dla odbiorcy. Korzystając z poradnika umieszczonego na stronie malinowepi.pl pobrano wyżej wspomniany kod, zainstalowano dodatkowe biblioteki (bcm2835-1.36) oraz uruchomiono skrypt **Adafruit\_DHT\_googledocs.ex.py** wysyłający odczytane pomiary do arkusza kalkulacyjnego w witrynie Google Docs. Służył on jako baza do końcowego skryptu obsługującego kontrolkę alarmu.

Kolejnym etapem w budowaniu programu pobierającego dane i wysyłające je na serwer była obsługa czujnika ruchu. Na samym początku dodano do kodu odwołanie do biblioteki RPi.GPIO umożliwiającej komunikację z portami wejścia - wyjścia Raspberry Pi. Za jej pomocą ustawiono tryb BCM (oznaczenia pinów wg Broadcom SOC channel), GPIO 23 (podłączenie czujnika) jak wejście oraz GPIO 3 (dioda led) jako wyjście. W odpowiedzi na zboczę narastające dodatniego impulsu na wejściu odpowiadającego za czujnik ruchu wywoływana jest obsługi tego zdarzenia. W niej wysyłany jest impuls dodatni na wyjście, do którego podpięta jest dioda led, powodujący jej świecenie.

Ostatnią funkcjonalnością dodaną do skryptu jest możliwość wysyłania zapytań do bazy danych. Wykorzystano do tego moduł MySQLdb, a jego obsługę wykonano zgodnie z poradnikiem [nr].

Kod źródłowy skryptu wraz z komentarzami:

import RPi.GPIO as GPIO #Moduł obsługujący porty Raspberry Pi

import subprocess

import re

import sys

import time

import datetime

import gspread

import MySQLdb #Moduł zapewniający kontakt z bazą danych

#Zmienne przechowujące numerów pinów, do których podpięte są czujnik ruchu i dioda

PIR = 23

LED = 3

#Ustawienie typów pinów

GPIO.setmode(GPIO.BCM)

GPIO.setup(PIR,GPIO.IN)

GPIO.setup(LED,GPIO.OUT)

#Dane bazy danych i użytkownika

dbUser = 'exten'

dbPassword = 'mp12345'

dbHost = 'rpi.bitnamiapp.com'

dbFolder = 'RPI\_ALARM'

port = 3306

#Zmienna wskazująca okres pomiaru temperatury oraz wilgotności powietrza jeżeli nie została podana przy uruchamianiu skryptu

t = 30

if len(sys.argv) == 2:

t = float(sys.argv[1])

#Funkcja obsługjąca zdarzenie wykrycia ruchu przez czujnik

def moveDetectorFunction(channel):

GPIO.output(LED, True)

insert("INSERT INTO `MOVE\_DETECTION`(`DATE`) VALUES ('%s')" % datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S"))

GPIO.output(LED,False)

#Funkcja wysyłająca zapytanie do bazy danych

def insert(query):

try:

db = MySQLdb.connect(dbHost, dbUser, dbPassword, dbFolder)

curs=db.cursor()

curs.execute(query)

db.commit()

except:

db.rollback()

#Ustawienie wywołania funkcji obsługi zdarzenia podczas wystąpienia zbocza narastającego na pinie z podpbiętym czujnikiem ruchu

GPIO.add\_event\_detect(PIR, GPIO.RISING, callback=moveDetectorFunction)

#Główna pętla programu

try:

while(True):

GPIO.output(LED,False)

#Uruchomienie programu odczytującego dane z DHT11

output = subprocess.check\_output(["./Adafruit\_DHT", "11", "4"]);

#Wydzielenie informacji o temperaturze powietrza z wyniku działania programu

matches = re.search("Temp =\s+([0-9.]+)", output)

if (not matches):

time.sleep(3)

continue

temp = float(matches.group(1))

#Wydzielenie informacji o wilgotności powietrza z wyniku działania programu

matches = re.search("Hum =\s+([0-9.]+)", output)

if (not matches):

time.sleep(3)

continue

humidity = float(matches.group(1))

#Wysłanie do bazy danych informacji o temperaturze i wilgotności powietrza

insert("INSERT INTO `HUMIDITY`(`DATE`, `VALUE`) VALUES ('%s','%.1f')" % (datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S"),humidity))

insert("INSERT INTO `TEMPERATURE`(`DATE`, `VALUE`) VALUES ('%s','%.1f')" % (datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S"),temp))

#Zatrzymanie działania pętli na zadany czas

time.sleep(t)

except KeyboardInterrupt:

GPIO.cleanup()

Dodatkowo, utworzono skrypt wywołujący powyższy program przy uruchamianiu systemu. Jego kod wygląda następująco:

### BEGIN INIT INFO

# Provides: scipt\_alarm

# Required-Start: $all

# Required-Stop: $all

# Default-Start: 2 3 4 5

# Default-Stop: 0 1 6

# Short-Description: alarm script

# Description: Run Raspberry Pi alarm service

### END INIT INFO

cd /home/pi/Documents/projekty

sudo python kontrolka\_alarmu.py 1800 &

Opis poszczególnych linijek:

# Provides: scipt\_alarm - określa nazwę skryptu

# Required-Start: $all - określa obiekty, które muszą być uruchomione przed skryptem

# Required-Stop: $all - określa obiekty, które mają być zakończone po zamknięciu skryptu

# Default-Start: 2 3 4 5 - definiuje poziomy uruchomienia systemu, gdzie skrypt powinien rozpoczęty

# Default-Stop: 0 1 6 - definiuje poziomy uruchomienia systemu, gdzie skrypt powinien zatrzymany

# Short-Description: alarm script - krótki opis skryptu

# Description: Run Raspberry Pi alarm service - pełny opis skryptu

cd /home/pi/Documents/projekty - przejście do lokalizacji programu, który będzie uruchomiony

sudo python kontrolka\_alarmu.py 1800 & - uruchomienie programu działającego w tle z uprawnieniami użytkownika root, który będzie pobierał dane z czujnika DHT11 co 1800 sekund (30 minut)

Szcegółowy opis informacji nagłówkowych na stronie wiki.debian.org [nr].

**6.3 Komunikacja bazy danych z aplikacją mobilną**

Istotną kwestią jest komunikacja bazy danych MySQL z aplikacją mobilną na telefonie z systemem Windows Phone 8. Z powodu braku odpowiednich bibliotek do obsługi MySQL w platformie .NET dla WP8 potrzebne było stworzenie odpowiedniego skryptu. Za bazę do jego napisania posłużył kod zamieszczony na forum centrum programistów na stronie Microsoft'u [nr]. Kod został napisany w języku PHP i wykorzystuje architekturę REST.

<?php

//Dane dostępu do bazy danych

$pool\_conn['my\_host'] = "localhost";

$pool\_conn['db\_name'] = "rpi\_alarm";

//Tabela pobierana jest z argumentu podawanego w linku

$pool\_conn['table\_name'] = htmlspecialchars($\_GET["table"]);

$pool\_conn['my\_user'] = "exten";

$pool\_conn['my\_pass'] = "mp12345";

//Pobieranie opcjonalnych argumentów przedziału czasu

if($\_GET["min"] === null)

$pool\_conn['min\_date'] = "(select min(DATE) from ".$pool\_conn['table\_name'].")";

else

$pool\_conn['min\_date'] = "'".htmlspecialchars($\_GET["min"])."'";

if($\_GET["max"] === null)

$pool\_conn['max\_date'] = "(select max(DATE) from ".$pool\_conn['table\_name'].")";

else

$pool\_conn['max\_date'] = "'".htmlspecialchars($\_GET["max"])."'";

//Połączenie z bazą danych

mysql\_connect($pool\_conn['my\_host'],$pool\_conn['my\_user'],$pool\_conn['my\_pass']);

@mysql\_select\_db($pool\_conn['db\_name']) or die( "Unable to select database");

//Utworzenie dokumentu xml

$xml = "<?xml version=\"1.0\" encoding=\"UTF-8\" standalone=\"yes\"?>";

$root\_element .= $pool\_conn['table\_name']."s";

$xml .= "<$root\_element>";

//Pobranie danych z danej tabeli

$sql = "SELECT \* FROM ".$pool\_conn['table\_name']." WHERE DATE >= ".$pool\_conn['min\_date']." AND DATE <= ".$pool\_conn['max\_date'];

$result = mysql\_query($sql);

if (!$result) {

die('Invalid query: ' . mysql\_error());

}

//Sprawdzenie, czy zapytanie zwróciło wyniki i rozpoczęcie pętli tworzącej dokument xml

if(mysql\_num\_rows($result)>0)

{

while($result\_array = mysql\_fetch\_assoc($result))

{

$xml .= "<".$pool\_conn['table\_name'].">";

foreach($result\_array as $key => $value)

{

//Znacznik zawierający nazwę tabeli

$xml .= "<$key>";

//Dodanie pomiędzy znacznikami wartości z kolumny

$xml .= "$value";

//Zamknięcie elemntu

$xml .= "</$key>";

}

$xml.="</".$pool\_conn['table\_name'].">";

}

}

//Zamknięcie korzenia

$xml .= "</$root\_element>";

//Wysłanie nagłówka dokumentu xml do przeglądarki

header ("Content-Type:text/xml");

//Zwrócenie dokumentu

echo $xml;

?>

Dane można wyświetlić wpisując w przeglądarce adres:

http://rpi.bitnamiapp.com/api.php?table=nazwa\_tabeli&min=data\_od&max=data\_do

gdzie:

nazwa\_tabeli: nazwa tabeli, z której chcemy pobrać dane. Możliwe wartości: humidity, temperature, move\_detection

data\_od: opcjonalny parametr wskazujący dolną granicę przedziału czasowego, z którego chcemy pobrać dane. Format daty: YYYY-MM-DD godzina:minuta:sekunda

data\_ do: opcjonalny parametr wskazujący górną granicę przedziału czasowego, z którego chcemy pobrać dane. Format daty: rok-miesiąc-dzień godzina:minuta:sekunda

Przykładowy wynika dla adresu http://rpi.bitnamiapp.com/api.php?table=humidity&min=2015-01-20%2010:00:00&max=2015-01-20%2015:00:00 :

<humiditys>

<humidity>

<ID>737</ID>

<DATE>2015-01-20 10:16:54</DATE>

<VALUE>39.0</VALUE>

<UNIT>%</UNIT>

</humidity>

<humidity>

<ID>738</ID>

<DATE>2015-01-20 10:46:57</DATE>

<VALUE>39.0</VALUE>

<UNIT>%</UNIT>

</humidity>

<humidity>

<ID>739</ID>

<DATE>2015-01-20 11:16:59</DATE>

<VALUE>38.0</VALUE>

<UNIT>%</UNIT>

</humidity>

<humidity>

<ID>740</ID>

<DATE>2015-01-20 11:47:05</DATE>

<VALUE>38.0</VALUE>

<UNIT>%</UNIT>

</humidity>

<humidity>

<ID>741</ID>

<DATE>2015-01-20 12:17:07</DATE>

<VALUE>38.0</VALUE>

<UNIT>%</UNIT>

</humidity>

<humidity>

<ID>742</ID>

<DATE>2015-01-20 12:47:10</DATE>

<VALUE>38.0</VALUE>

<UNIT>%</UNIT>

</humidity>

<humidity>

<ID>743</ID>

<DATE>2015-01-20 13:17:13</DATE>

<VALUE>38.0</VALUE>

<UNIT>%</UNIT>

</humidity>

<humidity>

<ID>744</ID>

<DATE>2015-01-20 13:47:15</DATE>

<VALUE>38.0</VALUE>

<UNIT>%</UNIT>

</humidity>

<humidity>

<ID>745</ID>

<DATE>2015-01-20 14:17:18</DATE>

<VALUE>39.0</VALUE>

<UNIT>%</UNIT>

</humidity>

<humidity>

<ID>746</ID>

<DATE>2015-01-20 14:47:20</DATE>

<VALUE>39.0</VALUE>

<UNIT>%</UNIT>

</humidity>

</humiditys>

**6.4 Aplikacja mobilna**

Założeniem tworzenia aplikacji była prezentacja w przystępny sposób wyników pomiarów na ekranie telefonu. Nawiązując do projektu

**Literatura i odnośniki**

https://education.github.com/pack/offers - opis oferty Students Developer Pack

http://www.micropik.com/PDF/dht11.pdf - dokumentacja czujnika DHT11

http://www.mpja.com/download/31227sc.pdf - dokumentacja czujnika PIR HC-SR501

http://www.raspberrypi.org/documentation/ - dokumentacja minikomputera Raspberry Pi

http://dev.mysql.com/doc/refman/5.5/en/index.html - dokumentacja MySQL 5.5

https://docs.python.org/2.7/ - dokumentacja języka Python 2.7.9

http://sourceforge.net/p/raspberry-gpio-python/wiki/Examples/ - przykłady użycia modułu kontrolowania portów GPIO Raspberry Pi w języku Python

http://msdn.microsoft.com/library/windows/apps/ff402535.aspx - dokumentacja technologii do tworzenia aplikacji pod system Windows Phone 8

https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code - repozytorium firmy Adafruit z skryptami dla Raspberry Pi

http://malinowepi.pl/post/80178679087/dht11-czujnik-temperatury-i-wilgotnosci-uklad - poradnik podłączenia czujnika DHT11 do Raspberry Pi

http://mysql-python.sourceforge.net/MySQLdb.html - poradnik obsługi bazy danych MySQL za pomocą języka python.

https://wiki.debian.org/LSBInitScripts/ - opis informacji nagłówkowych skryptu uruchamianego przy starcie systemu Rasbian