Politechnika Poznańska

Wydział Elektryczny

Instytut Automatyki i Inżynierii Informatycznej

Zakład Sterowania i Elektroniki Przemysłowej

****

Projekt Zaliczeniowy

Sławomir Smyczyński

Damian Kasperski

Mateusz Szymkowiak

Alicja Żmuda-Trzebiatowska

Arkadiusz Nawrot

APLIKACJE MOBILNE I WBUDOWANE DLA INTERNETU PRZEDMIOTÓW

Laboratorium

Prowadzący:

dr inż. Dominik Łuczak

dominik.Luczak@put.poznan.pl

Poznań 28.11.2016r.

**Dane szczegółowe**

Rok studiów: INŻ. IV

Rok akademicki: 2016/2017

Termin zajęć: poniedziałek g. 15:15

Data wykonania: 2017/01/15

Temat zajęć: Projekt Zaliczeniowy

Prowadzący: dr inż. Dominik Łuczak

Skład grupy (nazwisko, imię, nr indeksu):

1. Smyczyński Sławomir 116367
2. Kasperski Damian 116437
3. Szymkowiak Mateusz 119206
4. Żmuda-Trzebiatowska Alicja 115474
5. Nawrot Arkadiusz 116388

Ocena (*wypełnia prowadzący*)

Spełnienie wymogów redakcyjnych: ……………………………………… ………………………………………………………………………………

Wykonanie, udokumentowanie oraz opis wykonanych zadań: …………… ………………………………………………………………………………

Zastosowanie prawidłowego warsztatu programistycznego: ………………

………………………………………………………………………………

Spis treści

[I. Wstęp 4](#_Toc472451595)

[II. Konfiguracja systemu 4](#_Toc472451596)

[1. Konfiguracja Routera i sieci WiFi 4](#_Toc472451597)

[2. Konfiguracja komputera jednopłytkowego 5](#_Toc472451598)

[III. Elementy składowe Systemu 5](#_Toc472451599)

[3. Program Mikrokontrolera STM32F4 5](#_Toc472451600)

[4. Serwer WWW uruchomiony na komputerze jednopłytkowym 6](#_Toc472451601)

[5. Strona WWW (JavaScript + CSS + HTML) 7](#_Toc472451602)

[6. Aplikacja desktopowa ( C#) 8](#_Toc472451603)

[7. Aplikacja na urządzenie mobilne (Android) 9](#_Toc472451604)

[IV. Podsumowanie 9](#_Toc472451605)

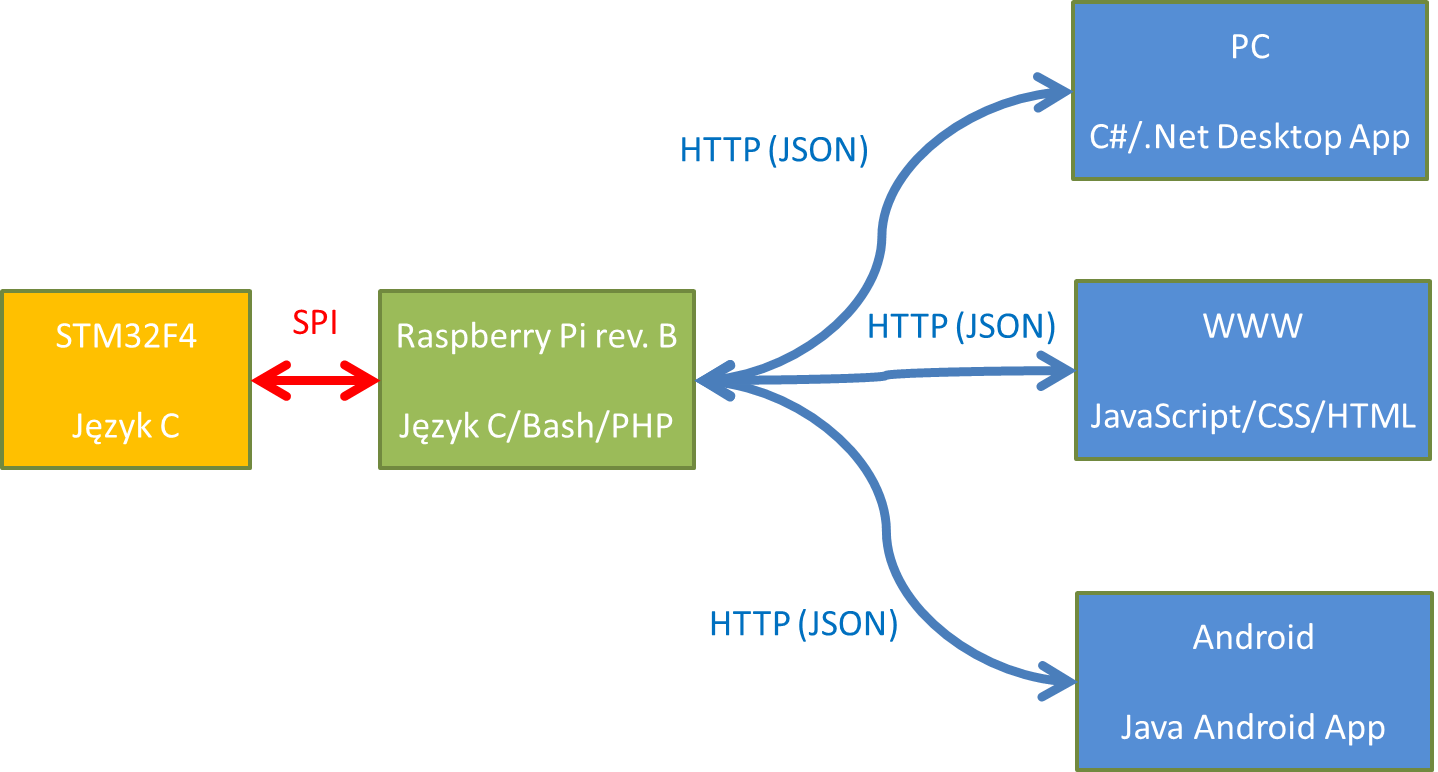
[Literatura 9](#_Toc472451606)

# Wstęp

W niniejszym raporcie przedstawiono opis oraz przebieg realizacji projektu zaliczeniowego z laboratorium przedmiotowego. Projekt ma charakter programistyczny. Ingeruje w sobie implementacje pięciu składowych elementów programistycznych, które były poznawane i ćwiczone w trakcie realiacji zajęć laboratoryjnych:

* Program w języku C mikrokontrolera ARM STM32F4,
* Serwer WWW wykorzystujący skrypty PHP uruchomiony na komputerze jednopłytkowym Raspberry Pi 3 Model B,
* Strona WWW z wykorzystaniem języków JavaScript, CSS oraz HTML,
* Aplikacja desktopowa w języku C# dla Windowsa,
* Aplikacja na system Android w języku Java.

Na rysunku ( Rys. 1) przedstawiono podglądowy schemat blokowy realizowanego systemu



Rys. 1 Schemat blokowy realizowanego projektu.

Z uwagi na złożoność zrealizowanego systemu projekt powstawał w ścisłej współpracy pięcioosobowego zespołu. Umożliwiło to wygodny rozdział pracy na członków zespołu.

W niniejszym sprawozdaniu skorzystano z szablonu sprawozdania dostępnego na stronie internetowej Zakładu Sterowania i Elektroniki Przemysłowej. [1]

# Konfiguracja systemu

## Konfiguracja Routera i sieci WiFi

Komunikacja pomiędzy serwerem, a poszczególnymi urządzeniami odbywa się za pośrednictwem sieci WiFi – aplikacja na system android umożliwia połączenie się z wybraną siecią, zaś program i strona otwierane na komputerze PC połączą się z serwerem po połączeniu przez użytkownika komputera z siecią za pośrednictwem karty sieciowej. Stworzony system działał na sieci lokalnej, która była zlokalizowana na routerze D-Link. W tym celu konieczna była wcześniejsza konfiguracja urządzenia dostępowego. Została ona przedstawiona w tabeli 1.1.

*Tabela 1.1 Konfiguracja routera.*

|  |  |
| --- | --- |
| Kanał sieci | 11 |
| Prędkość przesyłu danych | Najlepsza (dobierana automatycznie) |
| Zezwolenie na WMM | Tak |
| Typ zabezpieczeń | WPA2 - Personal |
| Typ szyfru | Auto(TKIP/AES) |
| PSK/EAP | PSK |
| Pasmo sieci | 2.4 GHz |

Bardzo istotne jest ustawienie kanału sieci na 11, ponieważ komputer jednopłytkowy Raspberry Pi pracuje tylko na wybranych kanałach.

## Konfiguracja komputera jednopłytkowego

Komputerowi jednopłytkowemu zostało nadane statyczne IP – 192.168.0.102. Ustawienie nie odbywa się na samym urządzeniu, a na serwerze DHCP, który został skonfigurowany, aby przydzielać stałe IP do adresu fizycznego (MAC) Raspberry Pi. Postawiony na urządzeniu serwer to dostępny na darmowej licencji serwer http Lighttpd z uruchomionym modułem FastCGI oraz zainstalowanym PHP w wersji 5. W celu uruchomienia komunikacji protokołem SPI, trzeba zezwolić na tę komunikację. Można to zrobić z poziomu konsoli, wpisując komendę *raspiconfig*, a następnie wybierając w ustawieniach SPI -> Enable.

# Elementy składowe Systemu

## Program Mikrokontrolera STM32F4

Źródło danych dla zrealizowanego systemu stanowi mikrokontroler ARM STM32F4. Jego program został napisany w języku C z wykorzystaniem bibliotek producenta StdPeriph. Jako środowisko programistyczne wybrano aplikację CooCox [2], które stanowi w pełni darmowy IDE (z ang. Integrated Development Environment) stworzone specjalnie dla procesorów ARM. Pomocnym okazał się też program STM Studio, który rozszerzał możliwości debugera środowiska programistycznego o podglądu zmieniających się wartości w czasie rzeczywistym. Warstwę sprzętową stanowi zestaw uruchomieniowy STM32F4-DISCOVERY, którego głównym element stanowi mikrokontroler STM32F407VG o częstotliwości zegara 168 MHz.

Zaimplementowany program umożliwia generację przebiegu trójkątnego o zadanej amplitudzie i stałej amplitudzie. W tym celu, jako podstawę czasu wykorzystano wbudowany układ czasowy, którego okres zliczania został ustalony na 1 s. Co tak ustawiony czas następuję inkrementacją lub dekrementacja licznika, zależnie od jego wartości. Jeżeli wartość licznika osiągnęła 10 następuje jej dekrementacja, natomiast ponowna inkrementacja jest wykonywana, gdy licznik osiągnie wartość 0. W ten sposób osiągnięto przebieg o okresie T = 20 s. Przeskalowanie amplitudy odbywa się poprzez podzielnie wartości licznika przez 10 oraz przemnożenie jej razy wartość amplitudy użytkownika. Domyślna wartość amplitudy to 10 j.

**void** **TIM2\_IRQHandler**() //generacja przebiegu trójkątnego

{

**if** (TIM\_GetITStatus(TIM2, TIM\_IT\_Update) == *SET*) // przerwanie sprzętowe – 1 s

{

TIM\_ClearITPendingBit(TIM2, TIM\_IT\_Update); //kasowanie flagi przerwania

**if**(zbocze==0) A\_temp++; // inkrementacja

**else** A\_temp--; // dekrementacja

**if** (A\_temp==10) zbocze=1; //zmień na zbocze opadające

**else** **if**(A\_temp==0) zbocze=0; //zmień na zbocze narastające

D=(A\_temp\*A\_usr)/10; //skalowanie amplitudy do użytkownika

}

}

Listing 1.1 Generacja przebiegu trójkątnego.

Tak zebrane dane transmitowane są poprzez magistralę SPI typu full-duplex. W tym celu konwertowane są one do postaci tekstowej. Mikrokontroler AVR pracuję w roli Slave, natomiast minikomputer Raspberry w roli Mastera. Master inicjuję proces aktualizacji danych poprzez wysłanie wartości ‘A’ (0x41). Dalsze wysyłanie (losowych) danych powoduję przesyłanie próbki pomiarowej. Stan taki trwa dopóki program Mastera odbierze znak końca linii LF (0x0A) – wtedy też Master konwertuje otrzymane dane z postaci tekstowej do liczby całkowitej jednocześnie przerywając ramkę komunikacyjną. Zmianę wartości amplitudy zrealizowano w analogiczny sposób – na początku ramki wysyłana jest zmienna ‘B’(0x42), po czym transmitowana ramka w formie stałej tekstowej. Ostatnim znakiem jest więc znak LF (0x0A), co stanowi sygnał dla mikrokontrolera, że nastąpił koniec przesyłu danych oraz można uaktualnić wartość amplitudy.

Dodatkową funkcjonalność stanowi dołączona obsługa przycisku użytkownika oraz diod LED. Możliwe jest sterowanie stanem diod LED poprzez przycisk na płytce lub z poziomu aplikacji. W tym celu w protokole uwzględniono dodatkowy wariant ramki komunikacji który umożliwia sprawdzenie stanu diod oraz ich zmianę.

Z uwagi no obszerność oraz typowy charakter napisanego programu, kompletne kody źródłowe dostępne są jako załącznik do niniejszego opracowania.

## Serwer WWW uruchomiony na komputerze jednopłytkowym

Na komputerze został zainstalowany serwer Lighttpd, zaś programy na nim zawarte zostały napisane w stworzonym na system Raspbian środowisku Geany. Dodatkowo program w języku C został zrealizowany z wykorzystaniem dostępnej na bezpłatnej licencji biblioteki wiringPi dedykowanej i służącej do obsługi GPIO komputera Raspberry Pi. Przekazywanie danych przez serwer składa się niejako z trzech poziomów. Skrypt PHP (Listing 2.1) odbiera zapytania wraz z parametrami ze stron internetowych i aplikacji.

<?php

$zadanie = $\_GET["z"];

$par1 = $\_GET["p"];

$wynik=**exec**("./AMuruchom.sh **$zadanie** **$par1**");

$json=**json\_encode**($wynik);

**echo** "**$json**";

?>

Listing 2.1 Skrypt PHP.

Następnie wywoływany jest skrypt AMuruchom.sh (Listing 2.2) w języku powłoki BASH. Przy wywołaniu następuje przekazanie do niego odebranych przez PHP parametrów.

**cd** **/**var**/**www**/**html

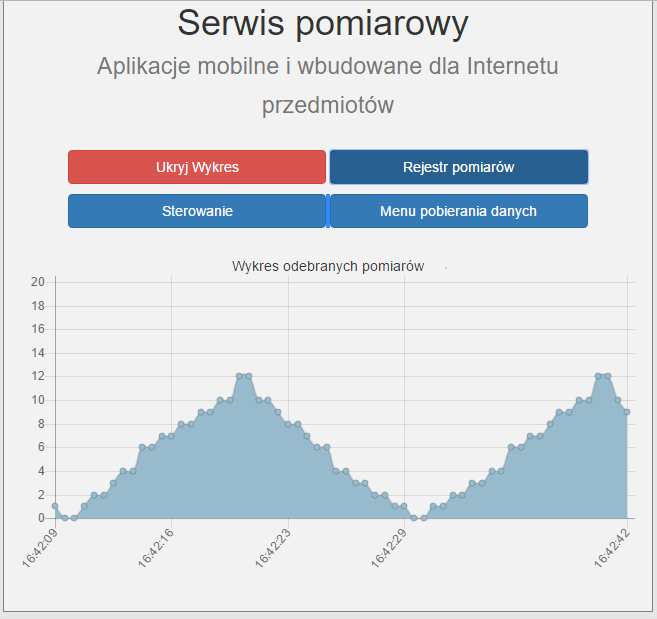
**sudo** **./**main **$1** **$2**

Listing 2.2 Skrypt Bash.

Skrypt BASH zmienia bieżący folder oraz wywołuje program w języku C z przekazanymi parametrami. W zależności od otrzymanych parametrów program wchodzi w różny tryb pracy. Może on odbierać aktualne wartości, parametry i stan urządzenia lub zmienia te parametry oraz stan urządzenia. Każdorazowo głównym zadaniem programu jest inicjalizacja i uruchomienie komunikacji protokołem SPI. Częstotliwość komunikacji została ustawiona na 250 kHz, za między wysłaniem poszczególnych bajtów zostało wprowadzone opóźnienie 10 µs. Dane są przekazywane tekstowo. Na końcu każdego pakietu danych wysyłany jest znak końca linii. Następnie odebrane dane są zwracane na zewnątrz programu do skryptu języka powłoki BASH, który przekazuje te dane do skryptu PHP. Odebrane dane w skrypcie PHP są konwertowane do formatu JSON, a następnie zwracane urządzeniu, które aktualnie odpytuje serwer.

## Strona WWW (JavaScript + CSS + HTML)

Aplikacja webowa została napisana w języku znaczników HTML. Do opisania interfejsu wykorzystano własne style CSS oraz te dostępne w frameworku Bootstrap. Skrypty pozwalające na ingerowanie w interfejs użytkownika napisano w JavaScript i AngularJS. Do narysowania wykresu wykorzystano repozytorium Charts.js oraz Angular-Chart.js. Głównym celem serwisu webowego jest możliwość wyświetlenia danych wczytanych z serwera. Pliki strony powinny zostać umieszczone na serwerze w folderze /var/www/html. Dzięki takiemu rożwiązaniu strona będzie dostępna dla każdego użytkownika połączonego z tą samą siecią co serwer. Następnie należy uruchomić przeglądarkę i w miejsce adresu wpisać adres IP serwera.



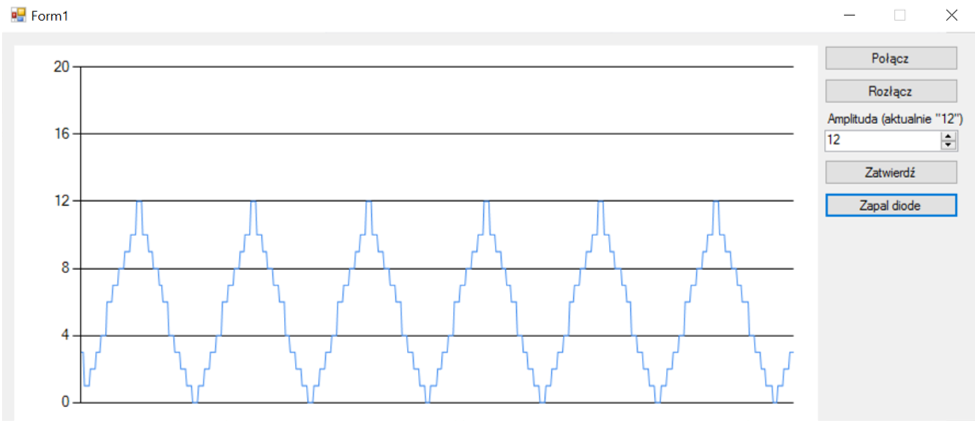
Rys. 3.1 Interfejs użytkownika.

Budowa strony została oparta o stworzenie menu składającego się na 4 przyciski. Oprócz nich na stronie znajdują się 4 moduły, każdy przypisany do odpowiedniego przycisku. Po uruchomieniu strony ich klasa to „hidden”, w wyniku czego są one niewidoczne. Zmienne opisujące klasy danych obiektów powiązano z kontrolerem za pomocą usługi $scope. Takie rozwiązanie pozwoliło na zmianę widoczności i wyglądu przycisków w odpowiedzi na naduszenie ich. Do obsługi strony został utworzony moduł AngularJS o nazwie app, kontroler myCtrl oraz usługa receiveData. Kontroler zajmuje się wszystkim co związane z interfejsem aplikacji. Na początku kontrolera utworzono zmienne przechowujące początkowe stany wszystkich przycisków oraz wykresu. Funkcje odpowiedzialne za zmiany wyglądu przycisków zostały oparte o proste instrukcje warunkowe. W celu automatycznego aktualizowania danych utworzono interwał wywołujący funkcję pobierania danych, która odwołuje się do samodzielnie stworzonej usługi *receiveData* w celu pobrania danych z serwera. Do odbierania danych wykorzystano zapytanie $http.get wywołujące skrypt PHP znajdujący się na serwerze.Po otrzymaniu danych, wywołane zostają funkcje aktualizujące tabelę oraz wykres. Kolejne dane wykresu zostały dopisywane przy użyciu metody push. W celu poprawienia wydajności strony, ograniczono ilość wyświetlanych punktów do 60 próbek. Ilość próbek na wykresie i ilość wierszy tabeli są sprawdzane za pomocą metody lenght. W przypadku przekroczenia określonej wartości część obiektu zostaje wycięta przy użyciu metody slice. Możliwość działania na tabeli stworzonej w pliku HTML umożliwiła komenda document.getElementById("dataLog"). Do regulacji wartości amplitudy wykorzystano suwak pozwalający na wybór wartości z zakresu od 5 do 20. Stan diody można zmienić przez naciśnięcie przycisku. Zadana wartość zostaje przekazana za pomocą wywołania $http.get z odpowiednim argumentem..

Ze względu na długość i powtarzalność kodu, został on załączony jako osobny plik źródłowy.

## Aplikacja desktopowa ( C#)

Kolejny element składowy projektu stanowi aplikacja desktopowa stworzona w języku C#. Program powstał w środowisku programistycznym Visual Studio. Najważniejszym zadaniem realizowanym przez aplikację jest pobranie danych z dedykowanego serwera poprzez odwołania metodą GET. Tak odebrane dane prezentowane są na wykresie generowanym w czasie rzeczywistym. Interwał czasowy pobierania danych z serwera został ustawiony na okres 500 ms. Wygląd aplikacji został przedstawiony na rysunku 4.1.



Rys. 4.1 Interfejs aplikacji

W momencie połączenia aplikacji z serwerem odebrana zostaje aktualna amplituda przebiegu oraz stan diod LED. Odbierane cyklicznie próbki są nanoszone na wykres. Do momentu wyświetlenia 600 próbek na wykresie, wykres ulega zagęszczeniu. Następnie po każdorazowym nadejściu nowej próbki, najstarsza jest usuwana, co powoduje przesuwanie się całego wykresu w lewo. Wszystkie odebrane z serwera dane są w formacie JSON, dlatego konieczne była konwersja danych do tablicy typu string. W tym celu konieczne było napisanie funkcji odpowiadającej za konwersję z formatu JSON do string. Nie było to bardzo skomplikowane zadanie ze względu na stałą formę przesyłanych danych. Funkcja ta została przedstawiona na Listingu 4.1.

**private** string Json\_to\_string**(**string tekst**)**

**{**

int znaki**=** tekst**.**Length**;**

string tekst2**;**

tekst2**=**tekst**.**Remove**(**0**,** 1**);**

tekst**=**tekst2**.**Remove**(**znaki **-** 2**,** 1**);**

**return** tekst**;**

**}**

Listing. 4.1 Konwersja danych z formatu JSON do string.

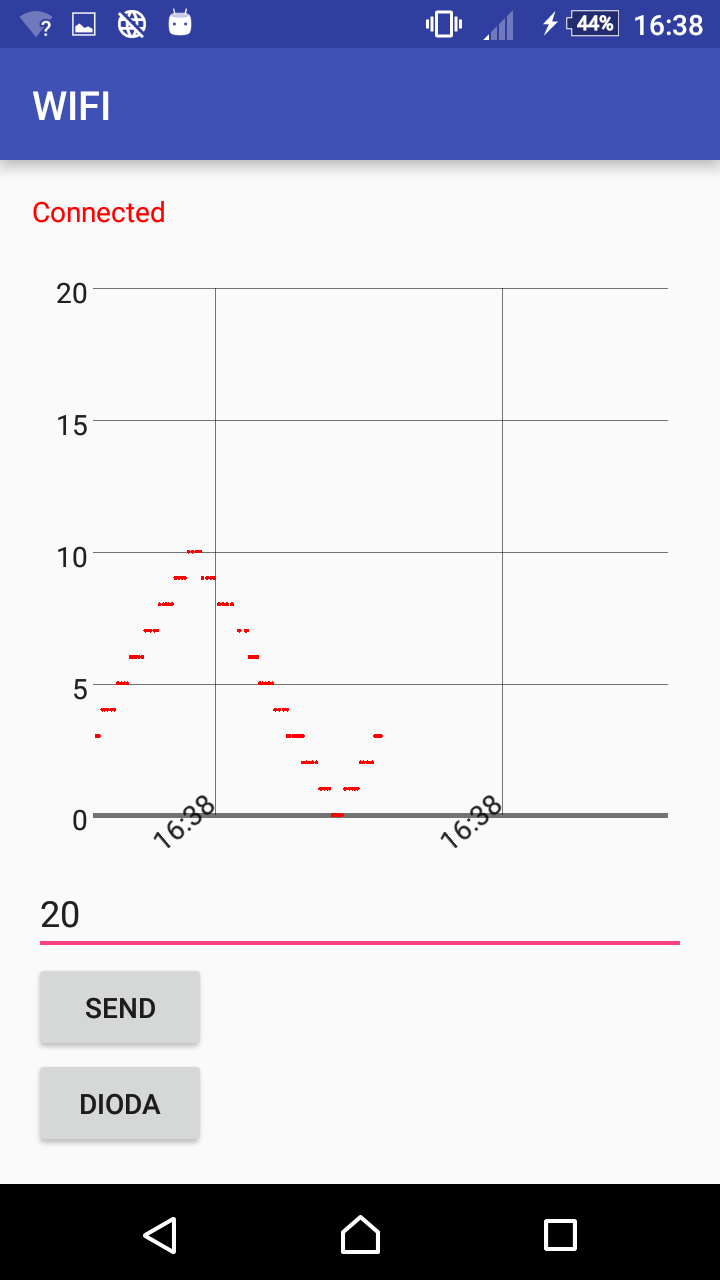
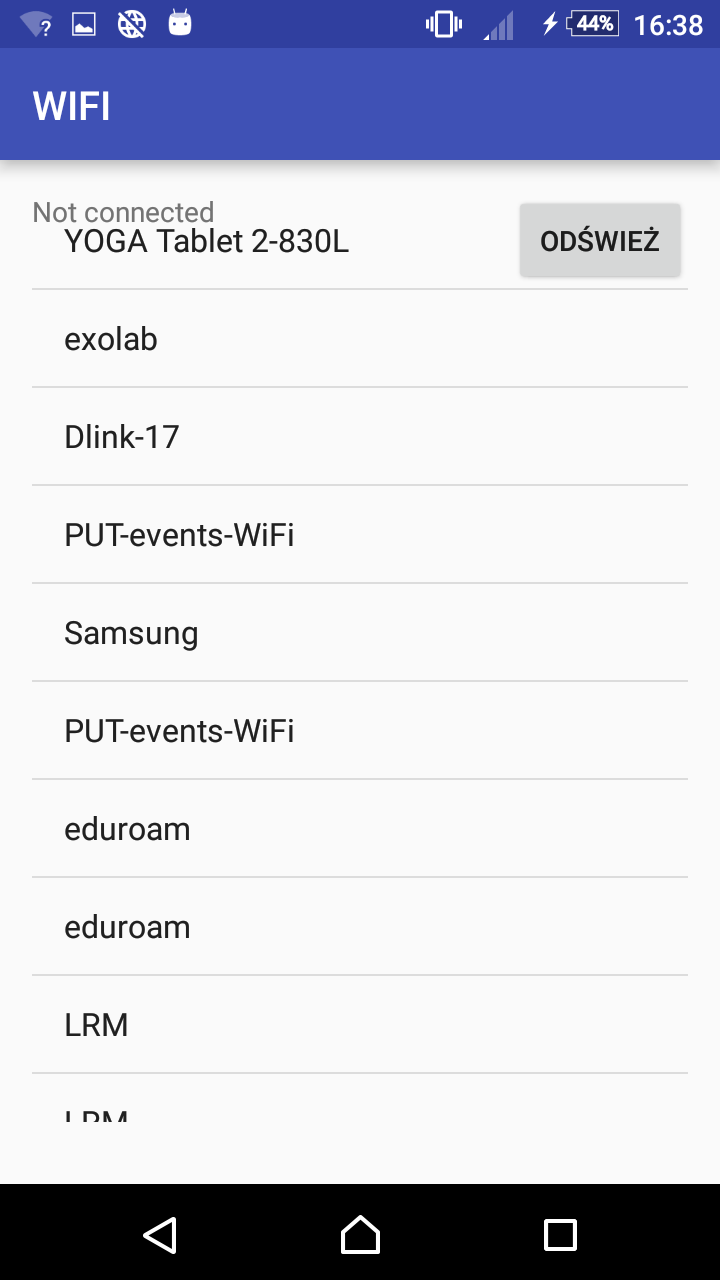
Następnie w razie potrzeby programu, odebrane dane są konwertowane na inne formaty, jednakże te konwersje dokonywane są już za pomocą narzędzi wbudowanych w języku C#.

Ponadto program umożliwia zmianę wartości amplitudy sygnału generowanego przez mikrokontroler oraz zmianę stanu diod LED. Dodatkowo w oknie programu pokazywana jest także informacja o aktualnej amplitudzie oraz stanie diod.

## Aplikacja na urządzenie mobilne (Android)

Aplikacja na system Android została napisana w Android Studio, które jest darmowym i oficjalnym IDE firmy Google. Głównym zadaniem aplikacji jest pełna komunikacja z lokalnym serwerem http. Program został napisany w języku Java, a layout stworzony z wykorzystaniem języka znaczników XML. Do wykonywania zapytań do serwera wykorzystana została biblioteka Volley, natomiast wykres został stworzony z wykorzystaniem biblioteki GraphView.

Serwer znajduje się w lokalnej sieci Wi-Fi, z którą należy połączyć również nasze urządzenie mobilne. W przypadku korzystania z sieci zabezpieczonej hasłem konieczne jest wpisanie go w programie (257 linia pliku MainActivity.java) i ponowne zbudowanie apk. Lista dostępnych sieci wyświetlana jest od razu po starcie aplikacji, jeżeli ta, z którą chcemy się połączyć nie jest wykrywalna, należy wcisnąć przycisk *odśwież*, by na nowo wyszukać dostępne sieci. Wybranie elementu z listy automatycznie połączy nas z siecią i rozpocznie się etap cyklicznych zapytań *GET* do serwera. Zapytania wysyłane są co 10ms, w wyniku czego następuje nasłuchiwanie odpowiedzi serwera. Dodatkowo została stworzona kolejka requestów, by odbierać je we właściwej kolejności. W przypadku wystąpienia błędu i nie otrzymania informacji zwrotnej informacja jaki wystąpił błąd zostanie wyświetlona na ekranie. Na podstawie otrzymywanych danych tworzony jest w czasie rzeczywistym wykres. Wykorzystuje on bibliotekę GraphView. Dodatkowo wpisując w pole *EditText* wartość z zakresu 5-20 i wciskając przycisk *Send* możliwe jest zdalne regulowanie amplitudy. Dodatkowo dostępny jest również przycisk *Dioda* przyciśnięcie go powoduje wysłanie informacji, by dioda znajdująca się na mikrokontrolerze zmieniła swój stan, aktualny stan diody sygnalizowany jest w aplikacji. Napis *Connected* znajdujący się nad wykresem jest koloru zielonego, gdy dioda jest zapalona lub czerwonego, gdy jest zgaszona.



Rysunek 5.1 Aplikacja mobilna

# Podsumowanie

Ostatnim etapem realizacji projektu było połączenie i integracja elementów składowych systemu. W pierwszej kolejności dokonano połączenia układu mikrokontrolera oraz minikomputera poprzez magistralę SPI. Po naprawie drobnych błędów (nieprawidłowe ustawienia inicjalizacji magistrali SPI oraz rozbieżności w analizie transmitowanych danych) udało uzyskać się komunikacje typu full-duplex. Następnym krokiem było równoległe łączenie aplikacji na system Android, strony internetowej oraz aplikacji komputerowej w języku C# z serwerem, a mianowicie bezpośrednie skomunikowanie się ze skryptem PHP. Podczas połączenia pojedynczego urządzenia z serwerem nie zauważono żadnych problemów. Zanotowano jednak sporadyczne błędy komunikacji podczas połączenia wielu urządzeń z serwerem. Przyczyną prawdopodobnie jest wielokrotne, jednoczesne wywołanie programu odpowiedzialnego za komunikację serwera z mikrokontrolerem.

Za pośrednictwem sieci WiFi możliwe jest zdalne sterowanie oraz obserwacja procesu z komputera, telefonu oraz każdego urządzenia mającego możliwość komunikacji oraz dostęp do sieci internetowej. W rezultacie powstał kompletny i działający system wieloplatformowy umożliwiający wizualizację i sterowanie procesem realizowanym zdalnie.

# Literatura

1. **Łuczak D.** Szablon pracy dyplomowej. [Online] 04 05 2016. [Zacytowano: 28 11 2016.] http://www.zsep.cie.put.poznan.pl.

2. **CooCox.** CooCox - Free/Open ARM Cortex-M Development Tool-chain. [Online] 2016. www.coocox.org/.

3. Przeglądarka FirefoxMozilla. [Online] 30 11 2016. https://www.mozilla.org/pl/firefox/desktop/.

4. **Łuczak D.** Wprowadzenie PHP. [Online] 15 11 2016. [Zacytowano: 28 11 2016.] http://www.zsep.cie.put.poznan.pl.