Politechnika Poznańska

Wydział Automatyki, Robotyki i Elektrotechniki

Instytut Robotyki i Inteligencji Maszynowej

Zakład Sterowania i Elektroniki Przemysłowej


# Zadanie zaliczeniowe
## System IoT

# Aplikacje Mobilne i Wbudowane dla Internetu Rzeczy

Materiały do zajęć laboratoryjnych

dr inż. Dominik Łuczak, mgr inż. Adrian Wójcik

Dominik.Luczak@put.poznan.pl
Adrian.Wojcik@put.poznan.pl

# I. Cel realizacji zadania

## Wiedzy

Celem zadania jest zapoznanie z:

- strukturą dedykowanego systemu IoT,
- rolą komunikacji sieciowej i architektury serwer klient w kontekście IoT,
- architekturą REST na przykładzie praktycznym,
- dostępnymi narzędziami i technologiami umożliwiającymi implementacje elementów systemu IoT,
- rolą wbudowanych systemów sterowania i pomiaru w systemach IoT

## Umiejętności

Celem zadania jest nabycie umiejętności w zakresie:

- implementacji aplikacji serwera IoT,
- obsługi komunikacji szeregowej między serwerem IoT a wbudowanym system sterowania i pomiaru,
- implementacji mobilnej aplikacji klienckiej systemu IoT,
- implementacji webowej aplikacji klienckiej systemu IoT,
- implementacji desktopowej aplikacji klienckiej systemu IoT,
- obsługi komunikacji sieciowej między serwerem a klientem IoT,
- implementacji automatycznych testów,

## Kompetencji społecznych

Celem zadania jest kształtowanie właściwych postaw:

- wykorzystywanie systematycznych technik i metodyk tworzenia oprogramowania,
- wykorzystywanie narzędzi i systemów ułatwiających pracę w zespole,
- prawidłowe i systematycznie testowanie wytworzonego oprogramowania,
- prawidłowa dokumentacja wytworzonej funkcjonalności systemu oraz wyników testów,
- dobór właściwej technologii i narzędzi programistycznych do zadanego problemu,
- ugruntowanie rozumienia i świadomości znaczenia pozatechnicznych aspektów i skutków działalności inżyniera oraz związaną z tym odpowiedzialność za podejmowane decyzje,

# II. Opis zadania zaliczeniowego

Zadanie zaliczeniowe polega na integracji i rozwinięciu przygotowanych w trakcie kursu aplikacji serwera i klientów systemu Internetu Przedmiotów (IoT). Zadanie powinno być realizowane w grupach nie liczniejszych niż 4 osoby. Ocena zadania bazować będzie na prezentacji działania przygotowanego systemu (c) oraz raportu dokumentującego rozwój i wyniki testów systemu (d).

Terminem oddania zadania zaliczeniowego jest **9 lipca 2021 r.**

## a) Struktura systemu IoT

Na rys. 1 przedstawiono schemat ideowy systemu IoT. System powinien składać się z pięciu podstawowych elementów:

1. Wbudowanego układ sterowania i/lub pomiaru, tj. systemu mikroprocesorowego wyposażonego w minimum jeden czujnik pomiary / wejście użytkownika oraz minimum jeden element wykonawczy / wyjście użytkownika - np. nakładka **Sense Hat** lub dedykowany układ sterowania oparty o zestaw uruchomieniowy z mikrokontrolerem STM32. Ten element systemu ma umożliwiać odczyt stanu czujników / wejść użytkownika oraz obsługę elementów wykonawczych / wyjść użytkownika.

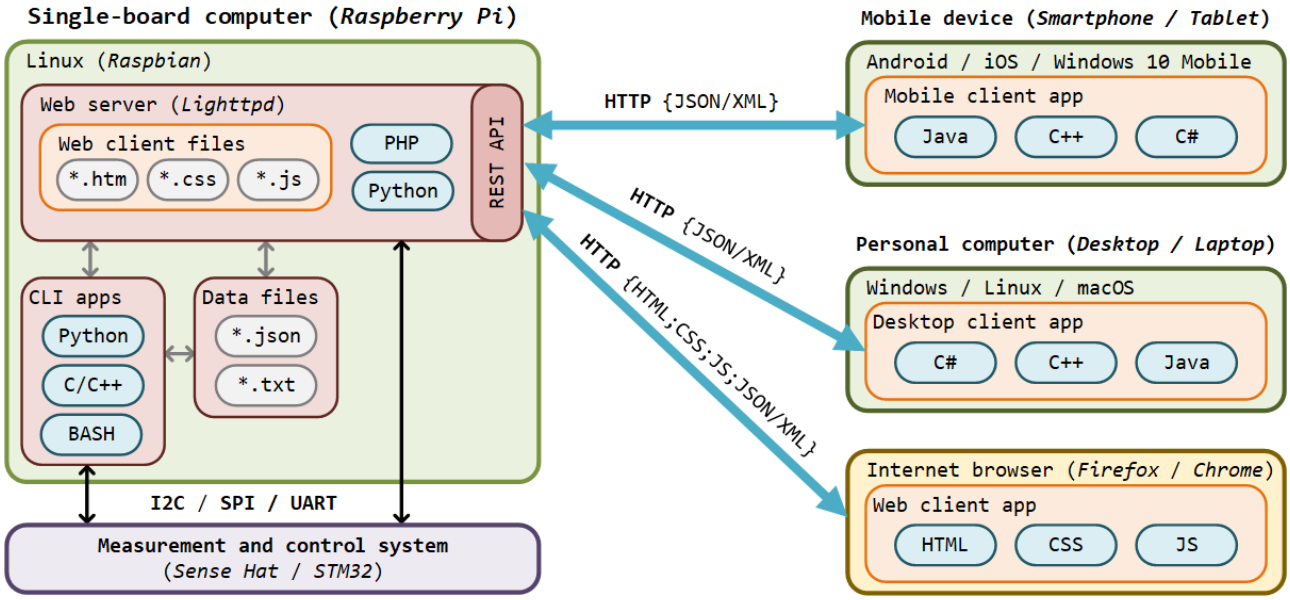
2. Komputera jednopłytkowego (ang. *Single-board computer*, SBC) umożliwiającym komunikację sieciową, uruchomienie serwera WWW oraz komunikację szeregową (np. I2C, SPI, UART) - np. **Raspberry Pi**. Ten element systemu ma pośredniczyć między układem wbudowanym a aplikacjami klienta za pomocą serwera WWW.

3. Urządzenie mobilne z systemem **Android**, iOS lub Windows 10 Mobile z dedykowaną aplikacją klienta.

4. Komputer osobisty z systemem **Windows**, **Linux** lub macOS z dedykowaną, okienkową aplikacją klienta.

5. Przeglądarka internetowa (uruchamiana w dowolnym środowisku) np. **Firefox** lub **Chrome**, umożliwiająca wykorzystanie dedykowaną, webową aplikację klienta.

Wszystkie aplikację klienta zawierać mają graficzny interfejs użytkownika (ang. *graphical user interface*, GUI) umożliwiający interakcję z układem wbudowanym, tj. podgląd stanu czujników / wejść użytkownika i obsługę elementów wykonawczych / wyjść użytkownika. Wszystkie elementy mogą zostać zastąpione przez maszyny wirtualne lub emulatory.




*Rys. 1. Schemat ideowy struktury systemu IoT.*

## b) Specyfikacja systemu IoT

Przygotowywany system IoT musi spełniać następujące **wymogi podstawowe**:

1. System umożliwia pobieranie podstawowych danych pomiarowych ze wszystkich czujników oraz wysyłanie podstawowych komend sterujących do wszystkich dostępnych elementów wykonawczych.
2. System umożliwia pobieranie danych ze wszystkich dostępnych wejść użytkownika oraz wysyłanie danych do wszystkich dostępnych wyjść użytkownika.
3. System udostępnia trzy graficzne interfejsy użytkownika w postaci *aplikacji mobilnej*, *aplikacji webowej* i *aplikacji desktopowej*.
4. Wszystkie trzy aplikacje klienta umożliwiają podgląd danych pomiarowych i wejścia użytkownika za pomocą dynamicznie generowanego interfejsu użytkownika (np. w formie listy lub tabeli).
5. Wszystkie trzy aplikacje klienta umożliwiają podgląd danych pomiarowych za pomocą wykresu przebiegu czasowego.
6. Wszystkie trzy aplikacje klienta umożliwiają próbkowanie danych pomiarowych z okresem nie większym niż 1000 ms.
7. GUI wszystkich trzech aplikacji klienta zawiera informacje o jednostkach fizycznych wielkości pomiarowych i parametrów systemu.
8. Wszystkie trzy aplikacje klienta umożliwią sterowanie pojedynczymi elementami wykonawczymi lub wyjściami użytkownika w pełnym dostępnym zakresie kontroli.
9. Wszystkie trzy aplikacje klienta umożliwiają konfiguracje parametrów systemu (np. adres IP, numer portu, wersja API, etc.).

Przygotowywany system IoT powinien spełniać następujące **wymogi dodatkowe**:

1. Implementacja systemu powinna wykorzystywać architekturę Representational state transfer (REST).
2. Implementacja aplikacji klienta z zachowaniem analogicznej architektury aplikacji oraz nazewnictwa metod we wszystkich trzech środowiskach.
3. Całość kodu źródłowego komentowana jest wg wspólnego standardu (np. Doxygen).
4. Automatyczne uruchomienie niezbędnych skryptów serwera po uruchomieniu systemu.
5. Konfiguracja (oraz dekonfiguracja) serwera wymaga uruchomienia pojedynczego skryptu.
6. Wszystkie trzy aplikacje klienta umożliwiają próbkowanie danych pomiarowych z okresem nie większym niż 100 ms.
7. Aplikacje serwera i klientów umożliwiają podgląd wszystkich wielkości fizycznych mierzonych przez czujniki.

*Np. czujnik LSM9DS1 umożliwia pomiar nie tylko kątów RPY ale również indukcji magnetycznej, przyspieszeń kątowych i liniowych, pomiar temperatury możliwy jest zarówno z czujnika LPS25H jak i HTS221, etc.*

8. Przy realizacji zadania wykorzystano system kontroli wersji (np. Git).
9. Do testowania wytworzonego systemu wykorzystano automatyczne testy.

10. Aplikacja mobilna wykorzystuje wzorzec architektoniczny zapewniający separację interfejsu użytkownika od logiki aplikacji (np. Model-View-Controller, Model-View-Presenter, Model-View-VieModel) [1][2].
11. Aplikacja desktopowa wykorzystuje wzorzec architektoniczny zapewniający separację interfejsu użytkownika od logiki aplikacji (np. Model-View-Controller, Model-View-Presenter, Model-View-ViewModel) [3].
12. System wykorzystuje cyfrowe przetwarzanie sygnałów pomiarowych, np. w postaci filtrów cyfrowych.
13. System wykorzystuje bezpieczną komunikacje sieciową: protokół HTTPS z użyciem certyfikatu z podpisem własnym [4][5].
14. Jednolita szata graficzna (*styl* lub *temat*) wszystkich aplikacji klienta.

### c) Prezentacja systemu

Prezentacja systemu polegać ma na demonstracji spełnienia założonych wymogów. Prezentacja systemu odbyć może się na trzy sposoby:

- Prezentacja on-line za pomocą wideokonferencji z wykorzystaniem narzędzi umożliwiających udostępnienie widoku z pulpitu prezentera lub kamery.
- Prezentacja off-line w postaci nagrania materiału - w konwencji analogicznej do prezentacji on-line.
- Prezentacja w laboratorium - jeden reprezentant grupy dokonuje prezentacji na żywo w sali laboratoryjnej.

Prezentacja oceniana jest na podstawie procentowej skuteczności realizacji podanych wymogów. W celu uzyskania oceny pozytywnej należy zrealizować wszystkie wymogi podstawowe. Jeżeli wymogi podstawowe zostały spełnione, ocena wyrażona w procentach wyznaczana jest wg formuły (1):

$$P = \frac{1}{2} \cdot \left(1 + \frac{WD}{8}\right) \cdot 100\% \tag{1}$$

gdzie: $P$ - procentowa ocena z prezentacji zadania,
$WD$ - liczba spełnionych wymogów dodatkowych,

### d) Raport z realizacji zadania

Raport dokumentujący realizacje zadania powinien składać się z czterech podstawowych sekcji:

1. **Opis specyfikacji**: tj. które wymogi systemu zamierzano zrealizować i czy przyjęto dodatkowe założenia. W tej sekcji należy też określić jakie docelowe platformy zostaną wykorzystane.
2. **Implementacja systemu**: sekcja ta powinna być podzielona na cztery podsekcje:
   (a) Aplikacje serwera,
   (b) Mobilna aplikacja klienta,
   (c) Webowa aplikacja klienta,
   (d) Desktopowa aplikacja klienta.

   Sekcja nie powinna zawierać całości kodu źródłowego a jedynie opis wykorzystanych narzędzi, przyjętych architektur aplikacji, zbudowanych interfejsów oraz instrukcje obsługi aplikacji klienckich. Wygodnym sposobem prezentacji wyników implementacji jest zunifikowany język modelowania (ang. *Unified Modeling Language*, UML).

3. **Wyniki testów i integracji systemu**: opis przyjętej metodologii testowania poszczególnych funkcjonalności oraz wyniki przeprowadzonych testów.
4. **Wnioski i podsumowanie**: opisanie które wymogi i założenia zostały spełnione i w jakim stopniu, a które nie zostały zrealizowane.

Dokument powinien być przygotowany zgodnie z wytycznymi redakcyjnymi raportu laboratoryjnego.

### e) Organizacja kodu źródłowego

Całość kodu źródłowego - w tym całość projektów w wykorzystanych IDE - musi zostać przekazana prowadzącemu po realizacji zadania. Przekazane pliki powinny być uporządkowane w następujący sposób:

- Archiwum `server_app.zip` zawierające wszystkie źródła aplikacje serwera oraz skrypty do konfiguracji i dekonfiguracji serwera.
- Archiwum `client_app_desktop.zip` zawierające wszystkie źródła desktopowej aplikacji klienckiej.
- Archiwum `client_app_mobile.zip` zawierające wszystkie źródła mobilnej aplikacji klienckiej.
- Archiwum `client_app_web.zip` zawierające wszystkie źródła webowej aplikacji klienckiej.
- Dokument `report_pl.pdf` zawierający raport.

Całość powinna być spakowana w archiwum o nazwie `System_IoT_Gx.zip` gdzie `x` to identyfikator grupy (`A-Z`).

## III. Środki dydaktyczne

| | |
|---|---|
| Sprzętowe | • zestaw komputerowy,<br>• urządzenie mobilne z systemem Android,<br>• komputer jednopłytkowy Raspberry Pi, |
| Programowe | • maszyna wirtualna Raspberry Pi,<br>• emulator urządzenia mobilnego z systemem Android,<br>• klient SSH (np. Bitwise SSH Client),<br>• edytor teksu (np. Notepad++, Visual Studio Code),<br>• serwer WWW (np. Lighttpd, Apache),<br>• Android Studio IDE,<br>• przeglądarka internetowa (np. Chrome, Firefox),<br>• Visual Studio IDE,<br>• interpreter Python,<br>• GNU Octave. |

W ramach realizacji zadania można korzystać z dowolnych narzędzi programistycznych, które pozwolą na realizację zadania wg. wytycznych. Sugerowane narzędzia bazują na rozwiązaniach prezentowanych na zajęciach laboratoryjnych.

## Bibliografia

1. *MVC vs. MVP vs. MVVM on Android* [online]. [B.d.] [udostępniono 2020-05-15]. Dostępne z: http://academy.realm.io/posts/eric-maxwell-mvc-mvp-and-mvvm-on-android/. Library Catalog: academy.realm.io.
2. *Android Architecture Pattern - MVC Example* [online]. 2019 [udostępniono 2020-05-15]. Dostępne z: https://androvaid.com/android-mvc-example/. Library Catalog: androvaid.com.

3. *Understanding MVC, MVP and MVVM Design Patterns* [online]. [B.d.] [udostępniono 2020-05-15]. Dostępne z: https://www.dotnettricks.com/learn/designpatterns/understanding-mvc-mvp-and-mvvm-design-patterns.
4. *OpenSSL* [online]. [B.d.] [udostępniono 2020-05-15]. Dostępne z: https://www.openssl.org/.
5. II, Thomas Hunter. *How to generate a Self Signed SSL Certificate for lighttpd* [online]. 2011 [udostępniono 2020-05-15]. Dostępne z: https://thomashunter.name/posts/2011-11-23-how-to-generate-a-self-signed-ssl-certificate-for-lighttpd. Library Catalog: thomashunter.name.