Gniazdko-WiFi Documentation

Wydanie 0.0.1

KNI-Genbit

November 02 2016

Spis treści

Treść:

ROZDZIAŁ 1

Cel

Pojawiła się idea, aby wykonać prototyp gniazdka Wi-Fi. Coś takiego jak gniazdko SUPLA, ale wykonane własnymi siłami. Byłby to praktyczny projekt wymagający wiedzy z różnej dziedzin informatyki, który może wzbudzać zainteresowanie członków zajmujących się rożnymi dziedzinami.

Wymagane jest stworzenie:

- prototypu aplikacji na komputer,
- aplikacji mobilnej,
- aplikacji na układ mikrokontrolera i urządzenia elektronicznego do sterowania zasilaniem,
- aplikacji serwerowej do utrzymywania komunikacji z gniazdkiem.

W przypadku tego projektu najrozsądniej wydaje się wykorzystać architekturę klient-serwer. Wówczas polega ona na ustaleniu, że serwer zapewnia usługi dla klientów, zgłaszających do serwera żądania obsługi. Klient czegoś żąda, serwer to zapewnia.

ROZDZIAŁ 2

Architektura

Wówczas będziemy mieli 1 **serwer/klaster**, który steruje gniazdkami, a także **dwa rodzaje klientów**:

- odpowiedzialnych za sterowanie gniazdkami,
- gniazdka.

W przypadku każdej sieciowej aplikacji sieciowej musimy stworzyć **protokół** tzn. **język komunikacji komponentów**, który będzie zrozumiały dla serwera i dla każdego rodzaju klientów. Wydaje mi się, że forma działania takiego protokołu musi wynikać z specyfiki dostępu do Internetu z poziomu mikrokontrolera, aby nie pożerało to ogromnej ilości prądu podczas ciągłego działania. Dzięki ustalonemu protokołowi możemy budować równolegle i niezależnie od siebie komponenty, a potem je spiąć z sobą.

Będziemy potrzebowali także przechowywać informacje o użytkownikach (login, hasło), a dla nich informacje o urządzeniach (identyfikator, stan). Dane te - ze względu na swój charakter - nie mogą być przechowywane w pamięci operacyjnej aplikacji, bo wystarczy awaria serwera, aby coś się utraciło.

ROZDZIAŁ 3

Protokół

Protokół aplikacji oprzemy na szeroko stosownym i opisanym w literaturze bezstanowy protokole HTTP/1.1. Jest powszechnie stosowany w aplikacjach sieciowych, dostępne jest szereg bibliotek i literatury na jego temat. Jest prostszy w użyciu i debugowaniu od protokołów binarnych, więc będzie przyjaźniejszy dla nowych ludzi.

Todo

Dyskusja czy stosować protokół HTTP, czy np. Protobuf lub inne.

Protokół HTTP jest protokołem poziomu aplikacji dla hipermedialnych systemów informatycznych. W obu wersjach jest to protokół bezstanowy, tzn. ani serwer (ani klient) nie przechowuje informacji o tym, jakie były wcześniej zapytania pomiędzy określonym serwerem i klientem oraz nie posiada stanu wewnętrznego. Wykorzystuje wyłącznie protokół TCP do komunikacji klient-serwer. W najczęstszej konfiguracji działa na porcie 80.

Protokół HTTP jest protokołem żądania i odpowiedzi. Klient wysyła żądanie do serwera w postaci linii żądania zawierającej informacje metody żądania, URI i wersji protokołu, a następnie komunikatu podobnego do wiadomości MIME (ang. Multipurpose Internet Mail Extensions) zawierającego modyfikatory żądania, metainformacje o docelowym serwerze i kliencie (dalej: nagłówki, ang. headers) oraz ewentualnie - oddzielone pustym wierszem – zawartość ciała (ang.body). W odpowiedzi otrzymuje kod statusu odpowiedzi, metainformacje i ciało oddzielone poprzez pusty wiersz.

W przypadku protokołu proponuje się tylko częściową implementacje protokołu HTTP, ale kompatybilną z popularnymi klientami tego protokołu.

Przykładowe żądanie bez dodatkowych informacji przedstawia:

```
1 GET /sciezka HTTP/1.1
2 User-Agent: curl/7.47.0
3
4 HELLO
```

gdzie:

- `GET /sciezka HTTP/1.1` - linia żądania, składająca się z:
- `GET` – metoda HTTP wykorzystywaną do komunikacji, co w przypadku tego protokołu oznacza wartość POST, PUT, GET lub DELETE,
- `/sciezka` – adres punktu wejścia dla danego żądania (ang. entry point),
- `HTTP/1.1` – wersja protokołu HTTP,
- `User-Agent: curl/7.47.0` – nagłówek żądania o nazwie `User-Agent` i wartośći `curl/7.47.0`

- `HELLO` - treść ciała żądania

W przypadku zamiaru przekazania ciała żądania należy wykorzystać nagłówek `'Content-Length'` dla określenia rozmiaru ciala, a następnie po przesłaniu nagłówków przesłać dodatkowy enter i przesłać ciało w formie tekstowej.

Do uwierzytelniania użytkowników wykorzystać mechanizm „Basic Access Authentication" przedstawiony w RFC2617. Możliwość zachowania stanów pomiędzy żądaniami zapewnić powiązanie określonych danych z konkretnym użytkownikiem po stronie aplikacji.

3.1 Ścieżki

Wszelkie zasoby udostępniane przez API są identyfikowane przez ścieżkę. To jest podstawowa forma do identyfikowania jakiegoś rodzaju danych, które mają zostać zmienione przez API.

Todo

Zdefiniować strukturę API z wykorzystaniem OpenAPI Specification, które wykorzystuje JSON Schema

Todo

Przedstawić komunikacje z wykorzystaniem mscgen i sphinxcontrib-mscgen

Indeksy i tabele

- genindex
- modindex
- search