Obiekty Internetu Rzeczy (OBIR)
2020 Zima
Specyfikacja zadania projektowego Z9

Zrealizować system (serwer) udostępniający opisane niżej zasoby za pomocą protokołu CoAP. Serwer powinien działać na platformie EBSimUnoEth, używanej w ćwiczeniu lab. realizowanym zdalnie. Dla wspieranych zasobów należy zaprojektować URI, oraz – tam, gdzie nie jest to oczywiste lub doprecyzowane w niniejszej specyfikacji – ich stan i reprezentację. Serwer ma współpracować z podanym klientem CoAP (patrz niżej), w zakresie wynikającym z podanych niżej możliwości stworzonej przez Zespół implementacji protokołu CoAP. Serwer powinien umieć generować różne kody odpowiedzi, stosownie do sytuacji. W przypadku błędu, serwer powinien zwracać payload diagnostyczny. Do systemu (kodu źródłowego) należy dołączyć opisujący go dokument.

Można korzystać ze (a) „standardowych" (dostarczanych z Arduino IDE) bibliotek dla platformy Arduino, oraz (b) bibliotek, których użyto do wykonania zadań na ćwiczeniach laboratoryjnych. Można także korzystać z fragmentów kodu, użytych na ćwiczeniach laboratoryjnych (np. fragmentów zawartych w instrukcjach do ćwiczeń). Można także wykorzystać istniejącą implementację CoAPa (całą lub we fragmentach), ale w tym przypadku konieczne pełne zrozumienie pobranego z tej implementacji kodu źródłowego (ważne!). Jeśli wykorzystujemy istniejącą bibliotekę, możemy usunąć z niej elementy niepotrzebne do realizacji zadania (ze względu na ograniczoność zasobów).

Uwaga: wyraźnie zaznaczamy, które części kodu źródłowego są pobrane (a nie zrobione przez nas). Jednoznacznie identyfikujemy źródło pobranego kodu źródłowego (url, licencja). Są to warunki konieczne tego, aby praca nie była plagiatem.
Uwaga: fragmenty pobranego kodu źródłowego usuwamy poprzez ich zamianę na komentarz, a nie fizyczne usunięcie.

Poza wymienionymi elementami kodu, całe oprogramowanie niezbędne do realizacji projektu musi być stworzone samodzielnie przez Zespół projektowy. Kod źródłowy powinien być dobrze skomentowany (średnio ok. 25% linii własnego kodu powinno być opatrzonych komentarzem). Wskazówka: komentowanie cudzego kodu to doskonały sposób zrozumienia go!

Zakres wsparcia protokołu CoAP:

1. Obsługa wiadomości NON (GET i/lub PUT, zależnie od potrzeb dla danego zasobu). Obsługa opcji Content-Format, Uri-Path, Accept. Obsługa tokena i MID.

2. Obsługa żądań CON (GET i/lub PUT, zależnie od potrzeb dla danego zasobu) i CoAP PING. Wybrana metryka z p. 3 poniżej powinna być traktowana jako „zasób o długim czasie dostępu". W tym przypadku należy stosownie zareagować na żądanie CON, aby uniknąć retransmisji. Wysyłanie odpowiedzi CON (z retransmisją) dla wybranej metryki z p. 3 poniżej. Uwaga: aby sprawdzić mechanizm retransmisji, należy programistycznie zapewnić ignorowanie potwierdzeń (ACK) wysyłanych przez klienta. Uwaga: dla realizacji retransmisji po zadanym czasie, wskazane rozważenie użycia metody `ObirMilis()` z klasy `ObirFeatures`.

Udostępniane zasoby:

1. Zasób opisujący pozostałe zasoby. Ścieżka /.well-known/core. Ścieżki i atrybuty pozostałych zasobów powinny być określone przez Zespół.
   GET: pobranie reprezentacji zasobu (w formacie CoRE Link Format).
   Uwaga: jeśli Zespół ma zaimplementować opcje Block2 i Size2 (patrz wyżej), to będziemy je testować na tym zasobie. A zatem będzie to zasób „duży" (ścieżki i atrybuty należy dobrać tak, aby długość reprezentacji zasobu wynosiła ok. 60B).

2. Zbiór wyrażeń arytmetycznych zapisanych za pomocą ciągu znaków w notacji polskiej odwrotnej (RPN), np. "*n n * 2 * n 3 * + 7 +*" (odpowiednik *2n^2+3n+7*).
   PUT: dodanie nowego wyrażenia do zbioru (zaczynamy od zbioru pustego, a liczba elementów w zbiorze zwiększa się o 1 po każdej operacji PUT). Gdy wyczerpie się pamięć przydzielona na wyrażenia, serwer powinien odesłać odpowiedni kod błędu. Uwagi dla ambitnych: 1. Można dodać wykrywanie niepoprawnych wyrażeń i odsyłanie kodu błędu. 2. W przypadku wyczerpania pamięci na wyrażenia można w odpowiedzi umieścić opcję Size1.
   GET: pobranie wszystkich wyrażeń ze zbioru. Uwaga: jeśli Zespół ma zaimplementować opcję Etag (patrz wyżej), to będziemy ją testować na tym żądaniu GET.
   GET: pobranie wartości wybranego wyrażenia dla argumentu *n* zadanego w komponencie query w URI. Uwaga: można założyć, że współczynniki wyrażenia oraz argumenty są liczbami całkowitymi.

3. Trzy metryki (statystyki) opisujących wymianę wiadomości/datagramów między klientem CoAP a platformą EBSimUnoEth. Metryki powinny być zaprojektowane przez Zespół.
   GET: pobranie reprezentacji metryki1.
   GET: pobranie reprezentacji metryki2.
   GET: pobranie reprezentacji metryki3.
   Uwaga: jeśli Zespół ma zaimplementować opcję Observe (patrz wyżej), to jedna z metryk powinna być obserwowalna (to na niej będziemy testować opcję Observe).
   Uwaga: jeśli Zespół ma zaimplementować obsługę żądań CON (patrz wyżej), to jedna z metryk powinna być traktowana jako „zasób o długim czasie dostępu". W tym przypadku należy stosownie zareagować na żądanie CON, aby uniknąć retransmisji.
   Uwaga: jeśli Zespół ma zaimplementować wysyłanie odpowiedzi CON (patrz wyżej), to odpowiedzi takie powinny być generowane dla jednej z metryk. Odpowiedzi CON będziemy testować na tym zasobie.

Odbiór projektu:

1. Demonstracja i interakcja z klientem CoAP według uprzednio przygotowanego przez Zespół skryptu („scenariusza"). Wszystkie wymienione w tej specyfikacji funkcjonalności powinny zostać zademonstrowane. Wymagane funkcjonalności powinny zostać zebrane w osobnej liście; przy nich powinny znaleźć się odniesienia do odpowiednich pozycji (testów) ze skryptu demonstracyjnego. Uwaga: przed rozpoczęciem demonstracji, konieczne jest właściwe ustawienie działania klienta CoAP (np. czy domyślnie wysyłać żądania NON czy CON).

2. Demonstracja II: interakcja z klientem CoAP w sposób określony przez prowadzących.

3. Omówienie architektury oprogramowania: zaprezentowanie przez Zespół krótkiej prezentacji (podczas prezentacji słownej pokazujemy treści, np. diagram z architekturą, z dokumentu – patrz niżej, nie tworzymy oddzielnej prezentacji) + dyskusja z prowadzącymi.

4. Omówienie wybranych przez prowadzących fragmentów kodu źródłowego. Uwaga: ten element odbioru ma w szczególności na celu potwierdzenie samodzielnego wytworzenia kodu.

Umieszczenie wyników projektu w repozytorium:

Wyniki projektu należy umieścić w przydzielonym przez prowadzących zajęcia indywidualnym repozytorium GIT (tam gdzie umieszczono emulator EBSimUnoEth) jednego z członków zespołu, w utworzonym w nim katalogu o nazwie "Projekt". Wyniki powinny znaleźć się w wybranym repozytorium co najmniej 24h przed sesją odbioru, a po umieszczeniu ich w repozytorium, jego właściciel powinien o tym fakcie poinformować prowadzących za pomocą e-maila.

Zawartość katalogu o nazwie „Projekt” powinna być następująca:

1. Dokument opisujący system (architektura oprogramowania: komponenty, interfejsy, przyjęte rozwiązania programistyczne, sposób testowania systemu i wykonane testy, skrypt dla celów demonstracji wraz ze wspomnianą wyżej listą funkcjonalności, zrzuty ekranu pokazujące interakcję z wtyczką CoAP).
   Uwaga: proszę nie dostarczać papierowej wersji dokumentu!

2. Pełny, skomentowany kod źródłowy.

Uwaga: w katalogu „Projekt” nie należy umieszczać plików, które nie są niezbędne dla odbioru projektu, a zajmują dużo przestrzeni dyskowej, np. wyników kompilacji (plików HEX) czy plików dostępnych w innych miejscach a nie wytworzonych w ramach prowadzonych prac projektowych (np. pliku FirefoxPortableESR_52_6_0_CoAP.zip).

Uwaga: wyniki projektu należy umieścić w repozytorium co najmniej 24h przed terminem odbioru (tj. datą i godziną slotu odbioru projektu dla danego zespołu). Inaczej nie ma czasu na wstępne sprawdzenie plików i ew. korespondencję w razie problemów.

Kryteria oceny (w nawiasie podana liczba punktów, w sumie 40):

1. Komentarze w kodzie źródłowym (3 punkty).

2. Zapewnienie wymaganej funkcjonalności (15 punktów).

3. Sprawna i kompletna demonstracja, wg dobrze przygotowanego skryptu (3 punkty).

4. Realizacja z naciskiem na oszczędzanie zasobów (pamięć danych i kodu, obciążenie procesora) (3 punkty).

5. Znajomość kodu źródłowego, własnego i ew. pobranego z wybranej implementacji CoAPa (12 punktów). Uwaga: aby zaliczyć projekt, konieczne jest uzyskanie co najmniej 6 punktów ze znajomości kodu źródłowego.

6. Jakość dokumentu (4 punkty). Uwaga: dokument jest obowiązkowy. Do oceny dokumentu nie wliczamy elementów pobranych (np. dokumentacji wybranej implementacji CoAPa). Uwaga: dokument musi zawierać zestawienie przeprowadzonych testów (jest to warunek konieczny uzyskania liczby punktów większej od zera).

Klient CoAP

Klientem CoAP jest wtyczka Copper (klient CoAP) dla przeglądarki Mozilla Firefox.

Uwaga: wtyczka Copper nie jest kompatybilna z aktualną wersją przeglądarki Mozilla Firefox. W projekcie należy użyć zestawu składającego się ze (a) starszej wersji przeglądarki Firefox oraz (b) wtyczki Copper. Zestaw należy pobrać stąd:

http://cygnus.tele.pw.edu.pl/olek/rozne/mmm/FirefoxPortableESR_52_6_0_CoAP.zip

Po pobraniu, rozpakowaniu i uruchomieniu powyższej wersji przeglądarki, nie należy uaktualniać Firefox'a (aktualizacja zniszczy wtyczkę Copper). Dodatkowo, podczas pracy tej wersji Firefox'a nie wolno uruchamiać innych wersji tej przeglądarki.

Proszę używać powyższego oprogramowania tylko do prac nad projektem OBIR. Nie należy używać go do innych zastosowań.