Akademia Górniczo - Hutnicza im. Stanisława Staszica w Krakowie

Wydział Informatyki, Elektroniki i Telekomunikacji
Katedra Informatyki

Dokumentacja projektu.

Smart Hospital. System "inteligentnego szpitala".

Praca została wykonana w ramach zajęć projektowo-laboratoryjnych z przedmiotu: Technologie Internetu Rzeczy.

Kierunek studiów: Informatyka, stacjonarne, semestr VI. Opiekun projektu: mgr inż. Marek Konieczny..

Spis treści.

1. Informacje wstępne.

Niniejszy projekt powstał w ramach przedmiotu Technologie Internetu Rzeczy, przeprowadzonego na Akademii Górniczo - Hutniczej im. Stanisława Staszica w Krakowie, Wydział Informatyki, Elektroniki i Telekomunikacji, kierunek Informatyka, rok III, semestr VI, rok akademicki 2015/2016.

Grupa ćwiczeniowa:
Czwartek, 11.15

Opiekun:
mgr inż. Marek Konieczny

Skład zespołu:
Grabis Wojciech
Grochal Bartłomiej
Sepielak Damian

Repozytorium Github: https://github.com/bgrochal/SmartHospital

2. Opis problemu.

Współcześnie coraz większą sympatię zyskuje koncepcja Internetu Rzeczy - zespołu urządzeń współpracujących ze sobą w sieci, mających na celu ułatwienie wykonywania codziennych czynności przez użytkownika. Jednym z obszarów, w których mógłby znaleźć zastosowanie system oparty o koncepcję Internetu Rzeczy jest niewątpliwie szpital.

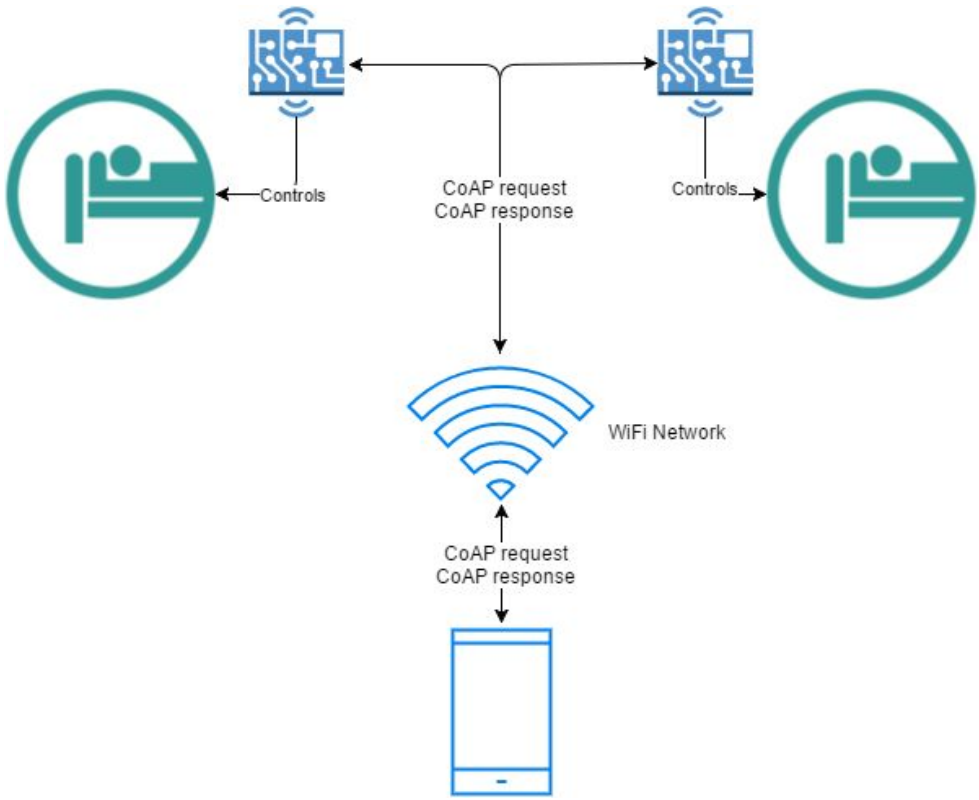
Zespół projektowy opracował koncepcję "inteligentnego szpitala" składającego się z wielu "inteligentnych łóżek", którymi mogłaby w wygodny sposób zarządzać jedna osoba przy pomocy aplikacji mobilnej, webowej lub desktopowej. Zarządzający (lekarz, pielęgniarka, inny pracownik szpitala) dla każdego "inteligentnego łóżka" mógłby w sposób zdalny, z dowolnego miejsca w szpialu wykonać miedzy innymi następujące czynności: zmienić położenie łóżka, sprawdzić aktualny stan zdrowia pacjenta, czy też dowiedzieć się natychmiast o nagłych sytuacjach alarmowych.

System "inteligentnego szpitala" powinien spełniać następujące wymagania funkcjonalne:

- Lekarz ma możliwość rejestracji stanu wyjątkowego występującego u pacjenta. Jeśli pacjent zgłosi zagrożenie poprzez naciśnięcie przycisku, w panelu lekarza pojawi się odpowiednia informacja.
- Lekarz ma możliwość pobrania aktualnego położenia łóżka pacjenta oraz zmiany tego stanu.
- Lekarz ma możliwość skojarzenia urządzenia z nazwą pacjenta (na przykład imieniem i nazwiskiem, numerem PESEL, unikalnym numerem itp.) oraz pobrania tej nazwy.
- Lekarz ma możliwość pobrania temperatury pacjenta jednorazowo lub w określonych odstępach czasowych.
- Lekarz ma możliwość wglądu do historii pomiarów temperatury pacjenta zapisanych w pliku. Ponadto lekarz ma możliwość usunięcia tego pliku.
- Lekarz ma możliwość rejestracji programu spadku temperatury pacjenta, podając: czas pomiędzy kolejnymi pomiarami oraz kolejne wartości maksymalne tych pomiarów. Jeżeli któryś pomiar przekroczy predykowaną wartość, automatycznie powinna wystąpić sytuacja alarmowa (jak w podpunkcie odpowiadającym pierwszemu wymaganiu). Rezultat wykonania programu zapisywany jest do pliku.
- Lekarz ma możliwość wglądu do pliku będącego rezultatem wykonania zadanego przez niego programu, jak również usunięcia tego pliku z pamięci urządzenia.

3. Wizja rozwiązania.

System składa się z wielu urządzeń IoT - "inteligentnych łóżek", pełniących rolę serwerów. Każde urządzenie posiada własny adres IP z pewnej określonej sieci. Po stronie klienckiej znajduje się lekka aplikacja mobilna, webowa lub desktopowa, w której zarządzający może wysyłać żądania zgodne z protokołem CoAP do poszczególnych serwerów. Każde urządzenie posiada określone zasoby implementujące zadane żądania CoAP, odpowiedzialne za realizację poszczególnych funkcji systemu.



Rysunek 1. Diagram ideowy architektury systemu.

4. Implementacja funkcjonalności.

Poniżej zamieszczono tabelaryczne zestawienie żądań protokołu CoAP dopuszczalnych dla poszczególnych zasobów w aplikacji oraz opis funkcjonalności udostępnianych przez te żądania.

Zasób	GET	PUT	POST	DELETE	Observable
Alert	+	-	-	-	+
Bed	+	+	-	-	-
Patient	+	+	-	-	-
Temperature	+	-	-	-	+
TemperatureHistory	+	-	-	+	-
TemperatureProgram	+	+	-	+	-

Tabela 1. Zestawienie żądań CoAP dopuszczalnych dla poszczególnych zasobów.

Zasób **Alert**:

- GET - zwraca napis *"Everything is OK."* w przypadku, gdy pacjent nie zgłasza alarmu lub napis *"ALERT!"* w przeciwnym przypadku.
- Observable - pozwala na pobieranie stanu alarmu (żądaniem GET) w odstępach pięciosekundowych.

Zasób **Bed**:

- GET - zwraca napis *"Bed position is: [pozycja]."*, gdzie *[pozycja]* jest wartością pozycji łóżka w skali: 0-31.
- PUT - przesyła pozycję łóżka, która jest interpretowana jako liczba w zakresie 0-31. W przypadku poprawnego wykonania zwraca napis: *"Bed position set to: [pozycja]."*, gdzie *[pozycja]* jest przesłaną wartością pozycji łóżka.

Zasób **Patient**:

- GET - zwraca nazwę pacjenta przyporządkowaną danemu urządzeniu lub napis *"NONAME"* jeśli nazwa ta nie jest ustawiona.
- PUT - przesyła nazwę pacjenta, którą należy przyporządkować danemu urządzeniu. W przypadku poprawnego wykonania zwraca napis: *"Patient name set to: [nazwa]."*, gdzie *[nazwa]* jest przesłaną nazwą pacjenta.

Zasób **Temperature**:

- GET - zwraca napis *"Temperature of patient is [temperatura]."*, gdzie *[temperatura]* jest wartością temperatury pacjenta. Ponadto wartość pomiaru wraz z jego datą i godziną zapisywana jest do pliku tekstowego *temperatures.txt*.
- Observable - pozwala na pobieranie stanu temperatury (żądaniem GET) w odstępach pięciosekundowych.

Zasób **TemperatureHistory**:

- GET - zwraca zawartość pliku tekstowego *temperatures.txt* zawierającego historię pomiarów temperatury pacjenta.
- DELETE - usuwa plik *temperatures.txt* z urządzenia pacjenta. W przypadku poprawnego wykonania zwraca napis: *"History of temperatures deleted successfully."*.

Zasób **TemperatureProgram**:

- GET - zwraca zawartość pliku tekstowego *program_output.txt* zawierającego wyniki programu ustalonego przez lekarza dla danego pacjenta.
- PUT - umożliwia zarejestrowanie i uruchomienie przez lekarza programu predykcji temperatury dla danego pacjenta. Przesyłana wiadomość musi być zgodna z quasi-protokołem:

  [czas] [pomiar1] [pomiar2] ... [pomiarN]

  gdzie: *[czas]* jest wartością odstępu czasowego (mierzonego w sekundach) pomiędzy kolejnymi pomiarami, *[pomiarI]* jest wartością maksymalną *I*-tego pomiaru temperatury. Program predykcji jest wykonywany w osobnym wątku, przez co działa w sposób nieblokujący dla urządzenia i pozwala na wywoływanie innych żądań podczas wykonywania programu predykcji. W przypadku poprawnego zarejestrowania i uruchomienia programu żądania zwraca napis: *"Program registered and started successfully."*. Jeżeli dany pomiar przekracza wartość przewidywaną przez lekarza, generowany jest alarm.
- DELETE - usuwa plik *program_output.txt* z urządzenia pacjenta. W przypadku poprawnego wykonania zwraca napis: *"Program's output file deleted successfully."*.

5. Spis tabel.

6. Spis rysunków.