Politechnika Świętokrzyska w Kielcach Wydział elektrotechniki automatyki i informatyki	
Technologie IoT rozproszone sieci sensoryczne	
Autor: **Marek Kopeć**	Grupa: **3ID15A**
Laboratorium nr: **1**	Data wykonania: **30.10.2018 r.**

1. Opisać GitHub

GitHub – portal, gdzie przetrzymywane jest repozytorium kodu raz za pośrednictwem którego możemy wymieniać się zmianami z innym deweloperami. GitHub mam dodatkowo wiele innych funkcji, jak choćby podgląd zmian, przeglądanie projektów, czy zarządzanie całym repozytorium z poziomu przeglądarki. Usług ta pozwala na prowadzenie repozytorium publicznego lub prywatnego. Niestety prowadzenie prywatnego repozytorium wiążę się z kosztami. Wiele popularnych edytorów kodu posiada funkcję automatycznego łączenia z GitHubem i przesyłania (updatowanie) danego projektu lub konkretnego pliku źródłowego.

System kontroli wersji służy do śledzenia zmian głównie w kodzie źródłowym oraz pomaga programista w łączeniu zmian dokonywanych w plikach przez wiele osób w różnym czasie. GitHub jest to typ rozproszonego systemu kontroli wersji. Rozproszony system kontroli wersji oznacza, że kod jest przetrzymywany lokalnie u każdego. Jeśli wprowadzimy zmiany w kodzie robimy je lokalnie (na naszym komputerze, nie w repozytorium GitHuba). Po zatwierdzeniu tych zmian wysyłamy je do repozytorium, a następnie osoby korzystające z naszego repozytorium pobierają nasze zmiany i łączą je ze swoimi lokalnymi modyfikacjami. Dzięki takiemu postępowaniu posiadamy historie zmian w kodzie co umożliwia nam powrót do wybranej wersji.

Podstawowe komendy:
git init - Inicjalizuje repozytorium GIT w danym katalogu
git add [nazwa_pliku] - Dodaje zmiany we wskazanym pliku do commita
git add . - Dodaje wszystkie zmienione pliki do commita
git add -p [nazwa_pliku] - Udostępnia możliwość dodania wybranych linii w zmodyfikowanym pliku do commita
git commit -m "[treść_commita]" - Dodaje opis do commita.
git add origin [adres_repozytorium] - Ustawia konkretny adres zdalnego repozytorium jako główne repozytorium
git push origin master - Wysłanie zmian do branacha zdalnego
git push –f - Wysłanie zmian do zdalnego repozytorium ignorując konflikty.
git checkout [nazwa_brancha] - Zmienia aktywny branch na wybrany przez użytkownika
git checkout [nazwa_pliku] -Usuwa zmiany w wybranym pliku
git checkout . - Usuwa zmiany we wszystkich zmienionych plikach

git checkout -b [nazwa_brancha] -Tworzenie nowego brancha z aktywnego brancha i przełączenie się na niego
git rebase master - Zaciągnięcie zmian z brancha głównego do brancha aktywnego
git push origin :[nazwa_brancha] - Usunięcie zdalnego brancha
git branch -d [nazwa_brancha] -Usuwanie brancha lokalnie. Nie można usunąć w ten sposób aktywnego brancha
git stash - Dodanie zmienonych plików do pamięci/stosu i usunięcie ich z aktywnego brancha
git pull –rebase - Pobranie najnowszych zmian z aktywnego brancha zdalnego
git stash pop - Przywrócenie zmodyfikowanych plików z pamięci/stosu
git stash clear - Czyszczenie pamięci/stosu
git remote prune origin - Pobranie aktualizacji o usuniętych branchach zdalnych
git fetch –all - Pobranie listy zdalnych branchy
git branch - Wyświetlenie listy lokalnych branchy
git branch –r - Wyświetlenie listy zdalnych branchy
git status - Wyświetlenie listy zmienionych plików
git diff [nazwa_pliku] - Szczegółowe wyświetlenie zmian w wybranym pliku
git reset HEAD - Resetowanie przygotowanych commitów (przed wysłaniem). Zmodyfikowane pliki są dostępne do ponownego dodania.
git reset HEAD –hard - usuwanie wszystkich zmian z brancha lokalnego i przywrócenie zmian z brancha zdalnego
git reset HEAD^ --hard - Usuwanie ostatniego commita z brancha
git reset HEAD^^ lub **git reset HEAD~2** -Obydwie komendy usuwają ostatnie 2 zmiany z brancha. Im więcej daszków (^) tym więcej commitów zostanie usuniętych.
git rebase -i HEAD~3 - Interaktywne zmienianie zawartości, opisów commitów. Commity mozna łączyć wtedy w jeden duży, zmienić jego opis, itd.

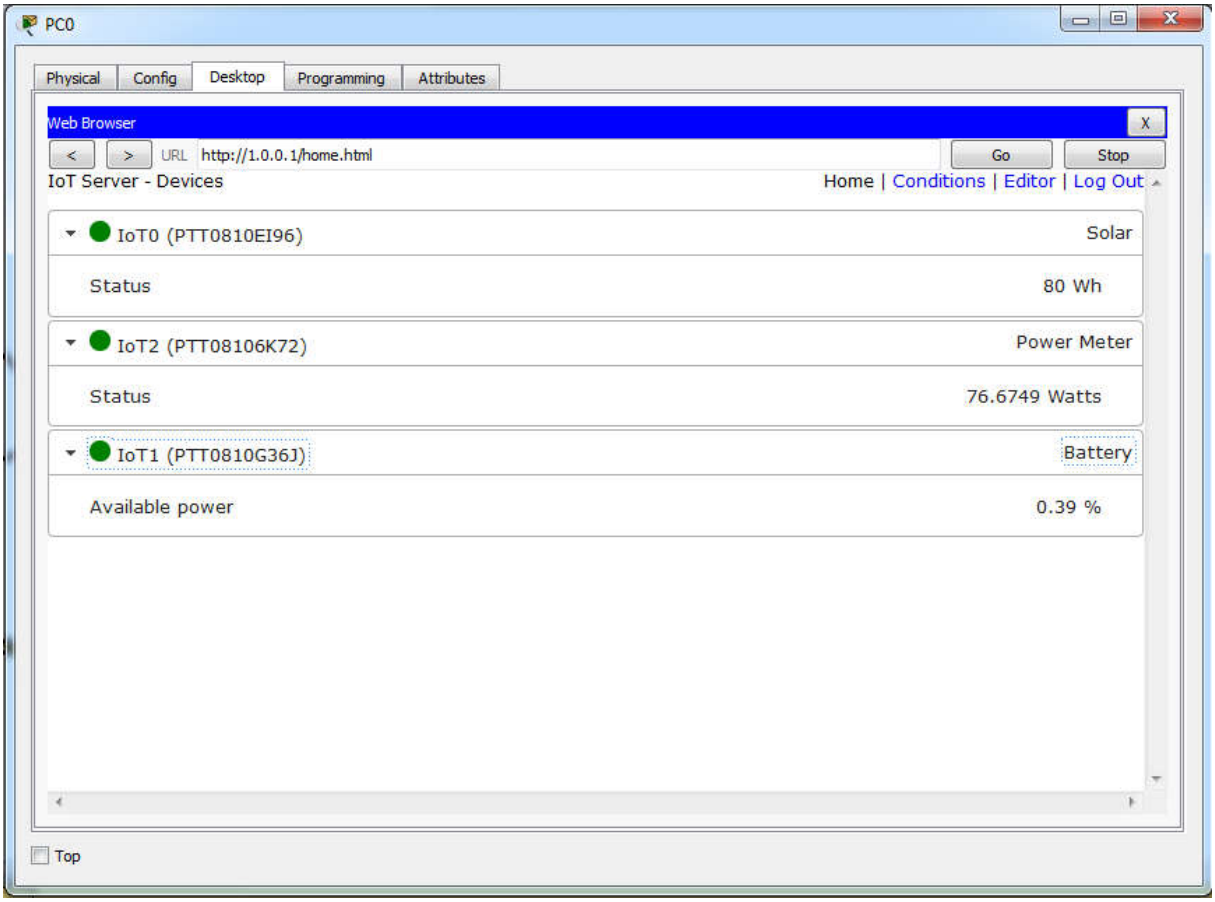
2. Część obserwacyjna

Obserwacja:

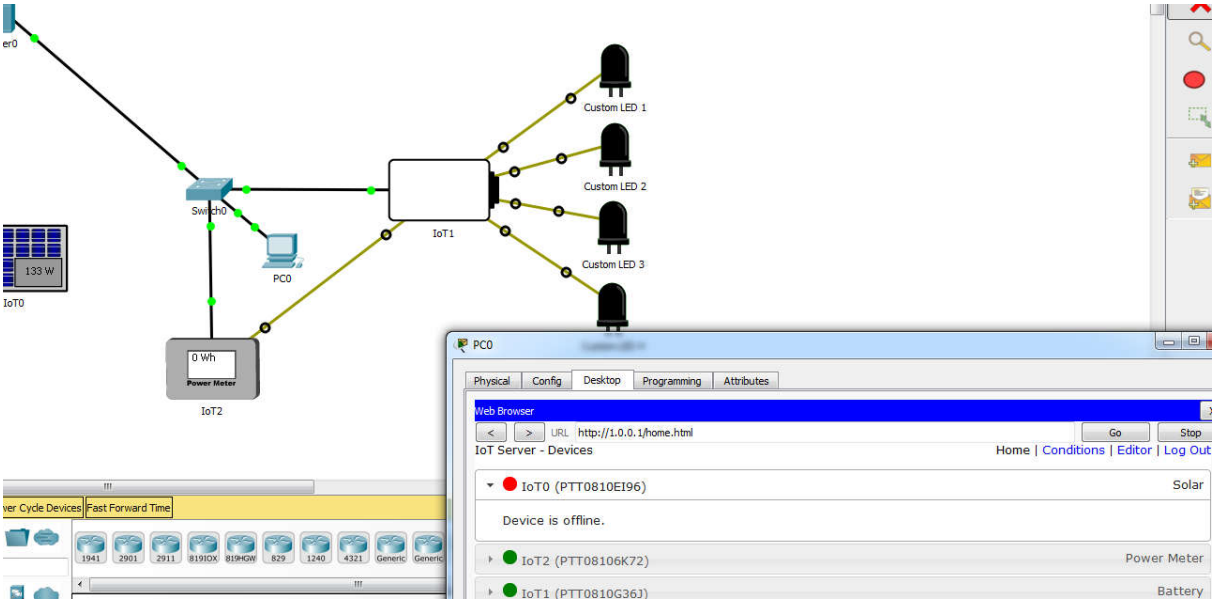
- **Zaobserwuj w jaki sposób bateria ładowana jest za pomocą ogniwa fotowoltaicznego**

Bateria w tym układzie jest ładowana za pomocą ogniwa fotowoltaicznego w określonych godzinach. Ze względu, że jest to układ symulowany ładowanie baterii odbywa się w określonych godzinach. W godzinie nocnej bateria nie jest ładowana analogicznie jak dzieje się to w przyrodzie. Przyglądając się czasowi czas ładowania zaczyna się od godziny 8^{00} i trwa do 18^{00}

- **Połącz się z serwerem za pomocą PC. Desktop -> Web Browser. Wpisz IP serwera i podaj dane do logowania**



- **Odłącz panel słoneczny i zaobserwuj działanie systemu przy pomocy PC**



System wykrył że urządzenie znajduję się w stanie „offline”. Bateria została bardzo szybko rozładowana. Status urządzenia zmienił się na czerwony. System wykrył, że bateria nie została doszczętnie rozładowana (posiadała jeszcze 0.38% energii) lecz jej ładunek był zbyt mały aby zasilić diody. Diody przestały świecić. Miernik wskazywał wartość 1 Wat.

- **Opisz działanie urządzeń. Jakie możliwości rozbudowy posiada symulowany system?**

Panel solarny – generuję prąd przetwarzając energię świetlną następnie wysyła ją do miernika gdzie energia ta zostaje zmierzona (zostaje podany jej aktualna moc w Wh) i następnie energia ta trafia do baterii gdzie jest gromadzona. Z baterii energia ta jest przesyłana do czterech diod LED które w wyniku jej świecą. Każdy z elementów IoT jest podłączony do przełącznika, który tworzy sieć dzięki czemu będąc podłączonym do tej sieci dzięki serwerowi jesteśmy w stanie odczytywać aktualny status urządzeń i ich wartości.

System ten można rozbudować o wszelaki czujniki i nimi zarządzać. Myślę, że w zależności co do zastosowania tego układu można by go rozbudować np. o czujnik ruchu który zapalałby diody jedynie gdy były one potrzebne lub inny czujnik, który mógłbym sygnalizować nas za pomocą diod o zaistniałym zdarzeniu.