Politechnika Świętokrzyska w Kielcach Wydział Elektrotechniki, Automatyki i Informatyki	
Laboratorium IoT Rozproszone sieci sensoryczne	
Grupa: 3ID14B	Laboratorium 2
Data: 18.10.2018	Lesiak Karol

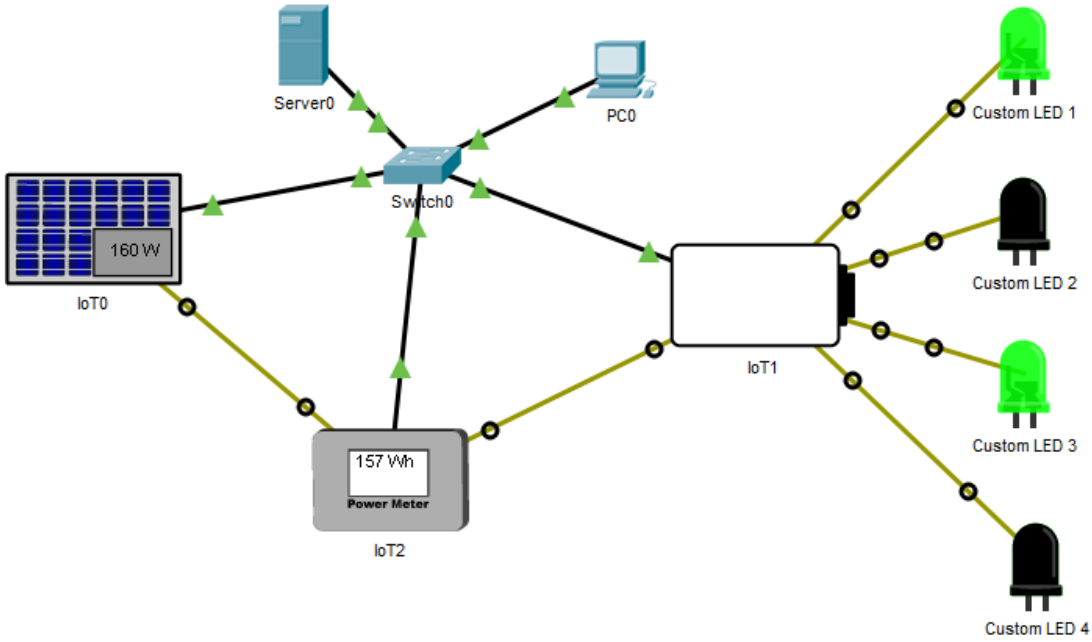
Cel laboratorium

Zapoznanie się z IoT przy stosowaniu symulacji Packet Tracer. Zapoznanie się z systemem kontroli wersji Git.

Krok1:Zainstalować Packet Tracer(PT) w wersji 7.1 lub wyższej
Krok2:Pobrać plik Lab1.pka z GitHub
Krok3:Przeprowadzić symulację:

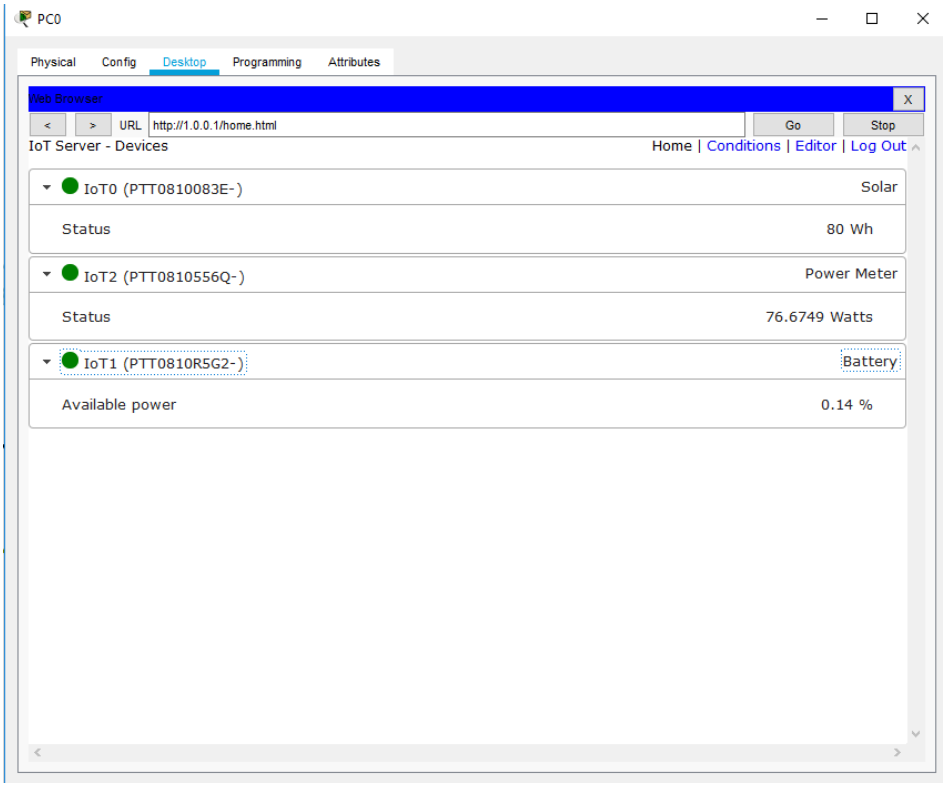
Podstawowa topologia IoT gdzie bateria jest ładowana za pomocą ogniwa fotowoltaicznego. W celu pomiaru ilości energii zastosowany został miernik(Power meter). Panel słoneczny, miernik oraz bateria połączone są do sieci komputerowej przy użyciu przełącznika.

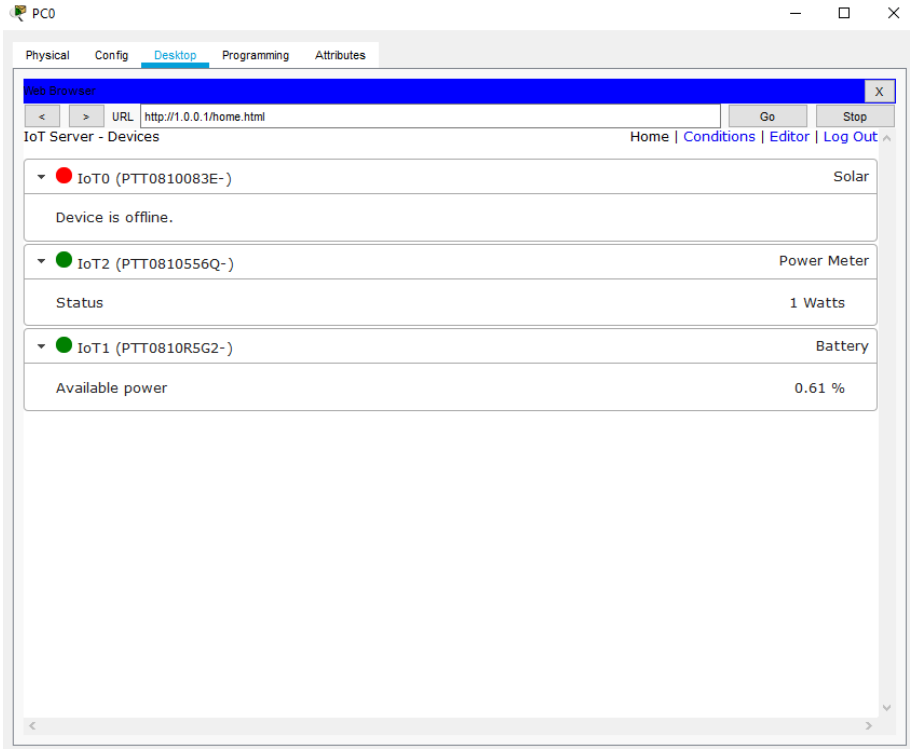
Symulacja



Cisco Packet Tracer symuluje upływ czasu, dzięki czemu mamy możliwość przeprowadzenia symulacji. Ogniwo fotowoltaiczne osiąga moc od 0 do 160 W. Ogniwo fotowoltaiczne dokonuje konwersji energii słonecznej w energię elektryczną w wyniku zjawiska fotowoltaicznego.
W prezentowanym układzie bateria nie jest w stanie się naładować. W związku z czym diody LED zależnie od symulowanej pory dnia, w dzień diody LED świecą lecz nie zawsze wszystkie, a w nocy gasną. Układ można rozbudować o dodatkowy panel słoneczny aby zapewnić odpowiednią ilość energii.

Podłączając się do serwera za pomocą PC uzyskujemy Dostep do statusu podłączonych urządzeń. Wyświetlana jest informacja o poziomie naładowania oraz informacje związane z ogniwem fotowoltaicznym i miernikiem. Po odłączeniu ogniwa uzyskuje ono status Device is offline.





System kontroli wersji

System kontroli wersji oprogramowanie służące do śledzenia zmian głównie w kodzie źródłowym oraz pomocy programistom w łączeniu zmian dokonanych w plikach przez wiele osób w różnym czasie. System kontroli wersji śledzi wszystkie zmiany dokonywane na pliku (lub plikach) i umożliwia przywołanie dowolnej wcześniejszej 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. Dobrym zwyczajem jest opisanie co ta zmiana wprowadza do kodu w zakresie funkcjonalnym

git add origin [adres_repozytorium, np. https://github.com/username/moje-repozytorium.git]
: 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, to znaczy, że jeśli wystapią konflikty to pliki zostaną nadpisane właśnie wysłaną wersją. Trzeba stosować to bardzo ostrożnie.

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^^

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

Bibliografia:

- https://blog.piotrnalepa.pl/2013/05/19/git-podreczny-zestaw-niezbednych-komend-dla-kazdego-webdevelopera-i-nie-tylko/?fbclid=IwAR2tiKTlZCv9gPd_Upq38W1_Odsuj417O9B4lMAsjGP-koiQJHJ9lCwuDVI

- https://git-scm.com/book/pl/v1/Podstawy-Gita-Rejestrowanie-zmian-w-repozytorium