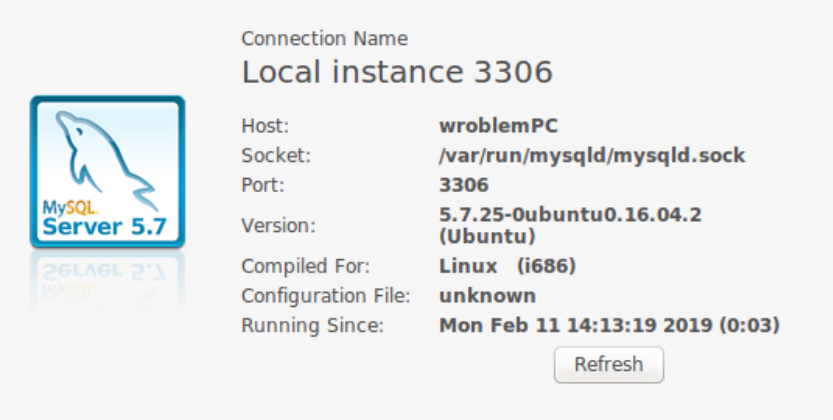
# Dokumentacja APKI do zbierania danych - SolarDynamics

Poznań, 11 lutego 2019 Michał Wróblewski

# 1 Wstęp

Apka na obecny moment (tj. luty 2019) zawiera kod na komputer główny w Pythonie, a współpracuje to z bazą danych na serwerze MySQL. System, na którym działałem to Ubuntu 16, a serwer w wersji 5.7 (screen poniżej).

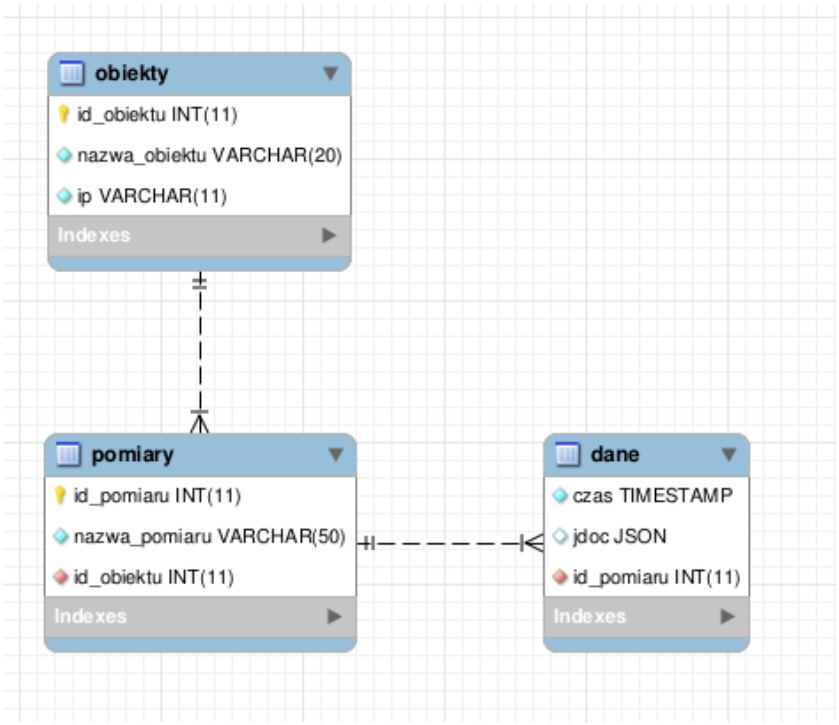



Rysunek 1: wersja serwera

Kod ma swoje repozytorium na GitHubie :
https://github.com/amadeuszsz/SolarCar/tree/wrobsi
Dalej przybliżę pewne zamysły dla kogoś kto miałby to kontynuować.

# 2 Baza danych

Cała baza sprowadza się do trzech tabel. Generalnie jakby ktoś chciał zmieniać kompozycje to radzę nie próbować tego upychać w mniejsza ilość bo nie będzie to przejrzyste.




Rysunek 2: struktura bazy

Dobra, o co tu chodzi...

Jako obiekt przyjąłem, że jest to urządzenie. Tabela pierwsza - "obiekty" zawiera listę takich urządzeń z id, które jednoznacznie identyfikuje obiekt, nazwą i adresem IP(zakładane na realizacje eteheretu dla CAN'a jakiś odpowiednik można zrobić). Poniżej przykładowe obiekty:

- baterie
- światła
- falownik
- itd

Każdy obiekt(urządzenie) będzie zwracał jakieś wartości nazwałem to "pomiar". Każdy pomiar ma swoje id, nazwe oraz przypisany obiekt po jego id. Dla falownika przykładowe pomiary:

- prądy
- częstotliwość
- temperatura
- itd

Druga tabela zawiera pomiary, powiązanie z obiektem jest przy pomocy id_obiektu.

Trzecia, ostatnia tabela to już bezpośrednie dane. Zawiera info o czasie, wartości w formacie JSON (taki format danych został ustalony) oraz id_pomiaru, którego dotyczą dane. Uznałem, że taki podział jest konieczny ze względu na ilość danych jakie auto będzie musiało przetwarzać. Każdy obiekt, urządzenie, pomiar ma swoje ID, po którym można je zidentyfikować.

### 2.1 Odtworzenie bazy

Baze możemy odtworzyć z pliku, który znajduje się na GitHubie w folderze "Baza...".

Ponadto w pliku sql_functions.py w pierwszej funkcji trzeba zmienić user'a - najprościej podać root'a i jego hasło z jakim się zainstalowało baze. Jeżeli nie będzie to root to należy pamiętać żeby użytkownik miał prawa do tabeli (edycja, wstawianie, itd).

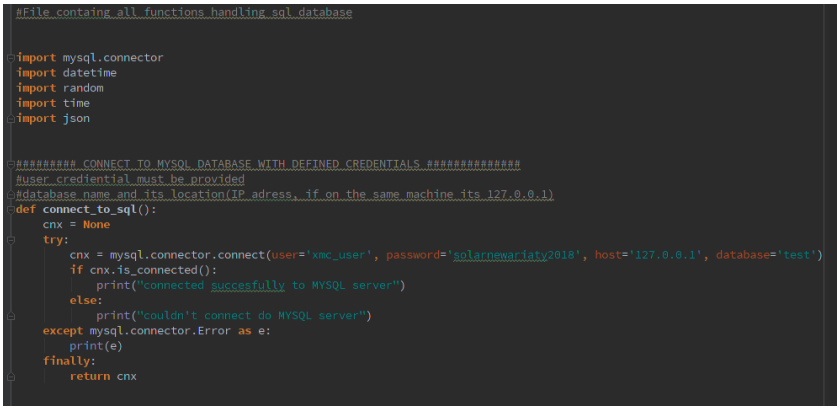
Jak już zainstalowaliśmy serwer, mamy plik do importu i zmieniliśmy user'a to postępujemy zgodnie z poniższą instukcją:

https://dev.mysql.com/doc/workbench/en/wb-admin-export-import-management.html

## 3 Kod

### 3.1 Kompilacja

Używałem Pythona 3.5 . Aby mieć pewność, że zbudujecie projekt proponuje zainstalować wszystkie pakiety, które ja miałem. Lista ich




Rysunek 3: zmiana usera

znajduję się na GitHubie w pliku requirements.txt . Trzeba przejść na ścieżkę gdzie wypakujecie sobie cały projekt z github'a i wpisać:

pip3 install -r requirements.txt lub

pip install -r requirements.txt

Jak to z paczuszkami bywa mogą być różne problemy ale to już indywidualnie trzeba googlować, raczej nie powinno być ich za dużo.

## 3.2 Struktura

Co do samego kodu to opisy funkcji znajdują się już bezpośrednio w kodzie ale opisze mniej więcej co znajduje się w jakim pliku żeby było łatwiej się poruszać.

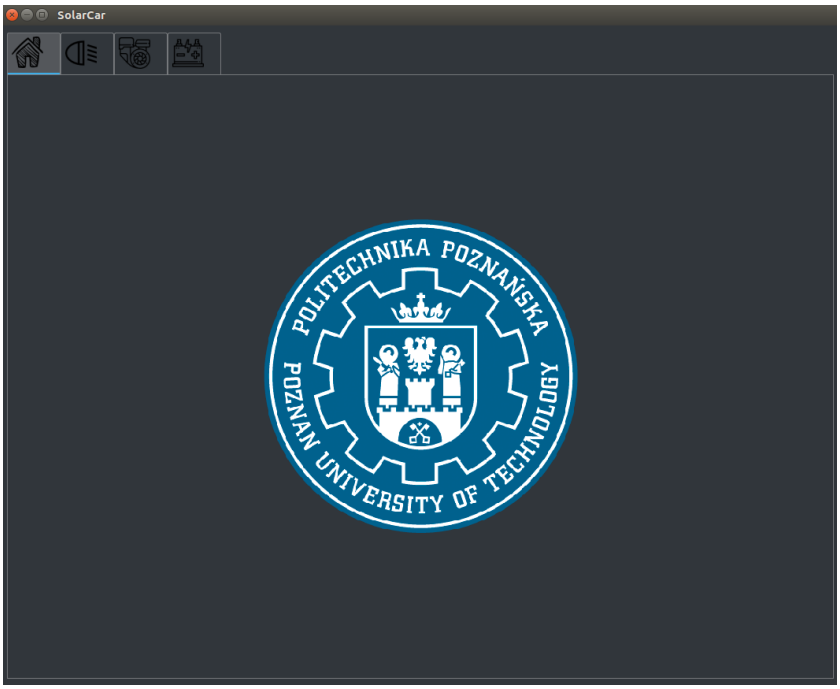
- folder "baza..." zawiera plik do zaimportowania bazy danych, o którym pisałem wyżej
- folder "mywidgets" tam znajdują się poszczególne widgety, które sam robiłem i były wykorzystywane:
  - kalendarz - do wyznaczania zakresu daty to narysowania historycznych danych
  - plotter - okienko do rysowania wykresów

  - plot_controller - do sterowania wykresem, wybierania co ma się rysować plus pauzowanie itp
  - mytab - jedna zakładka, którą można wywołać wielokrotnie, na niej ustawiałem wszystkie powyższe widgety

- sql_functions.py - plik z funkcjami do pracy z bazą danych, pobieranie danych itd..
- xmc.py - plik z pomocnicza klasa, która zarządza jednym pomiarem (np. temperatura), miała także słuzyć przy komunikacji po Ethernecie ale jest to nie aktualne
- mythread.py - klasa z wątkiem, który można pauzować
- home.py, battery.py , lights.py itp... są to pliki z funkcjonalnością poszczególnych zakładek
- main.py główny plik, od którego startuje program

## 3.3 Opis GUI

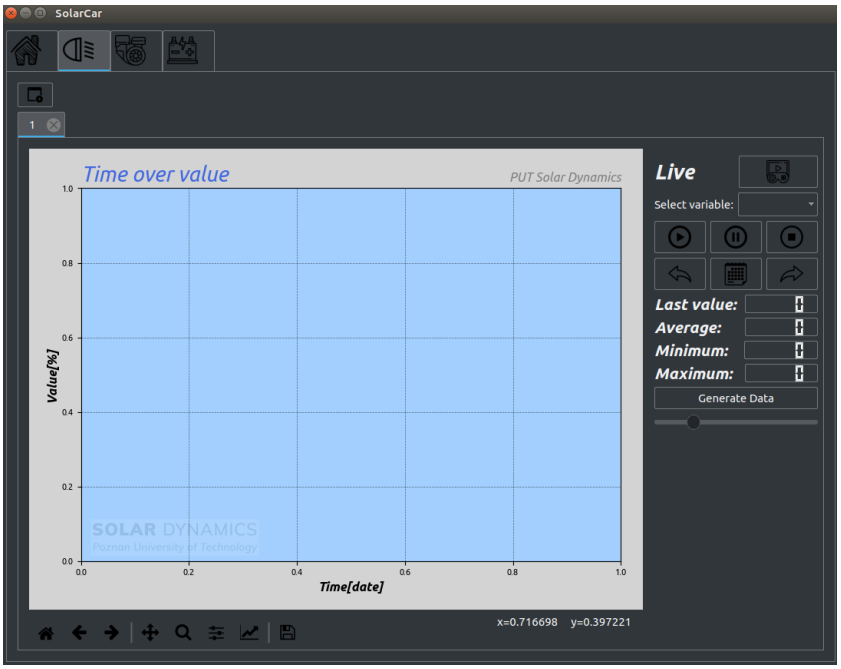
Apka składa się podstawowo z zakładek. Są to odpowiednio ikona świateł silnika itd... Pierwszą zakładką jest ekran powitalny, tutaj docelowo jakąś ładną grafikę z logo koła trzeba by wrzucić. Screen poniżej




Rysunek 4: gui

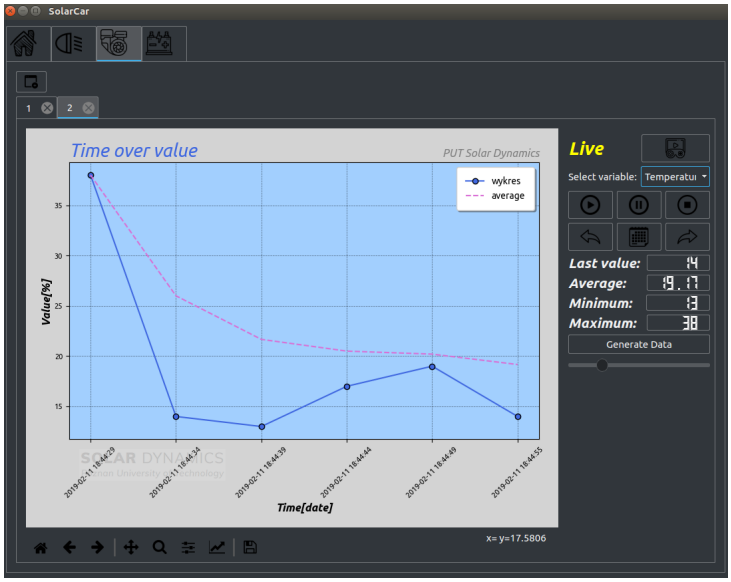
Pierwsza zakładka to światła, następna to falownik, która wygląda identycznie lecz ma dostęp do innych danych. Każda zakładka pobiera dane z odpowiedniego miejsca w bazie danych. Główna część to wykresy, plik odpowiedzialny za jej działanie to mywidgets
plotter.py. Po prawej od wykresów jest widget do sterowania nim (plik mywidgets
plot_controler.py. Zaimplementował na razie takie funkcji: -generowanie losowych danych (do testów) trzeba kliknąć guzik 'generate data' i suwakiem dostosować częstliwość -wyżej informacje dot. dane (max,min,avrage...) -przyciski do sterowania, działą start/stop/pauza + kalendarz -wybór

co chcemu obserowować, czyli pomiary przypisane z bazdy danych (menu rozwijane “select variable”) -jeżeli chcemy mieć wykres na żywa (wcześenij włączyć generowanie danych) to klikamy przycisk obok napisu Live -napis Live zmienia się kolarami w zależności od stanu wykresu na żywo(stop/pauza itd.)



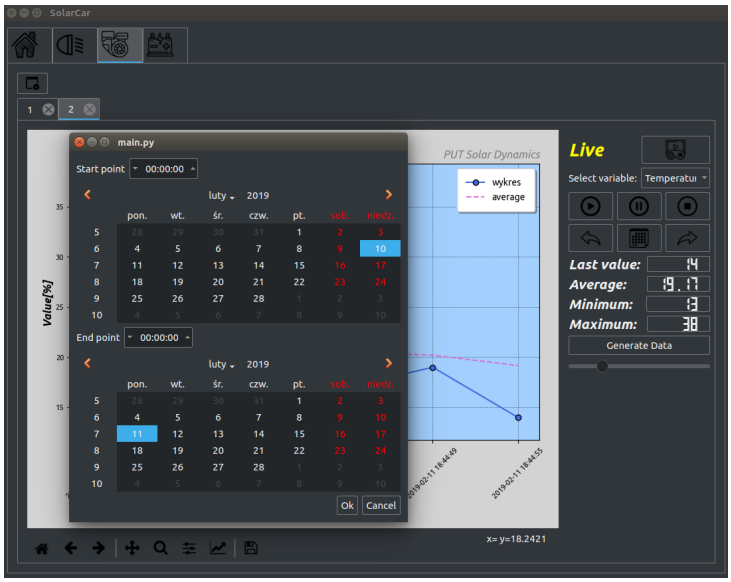

Rysunek 5: gui

Przykład działania poniżej, na zakładce od falownika wybrałem pomiar Temperatury. Zaczałem generować dane i wygląda to tak jak na zdjęciu poniżej(kolor żółty ”Live” bo pauza).



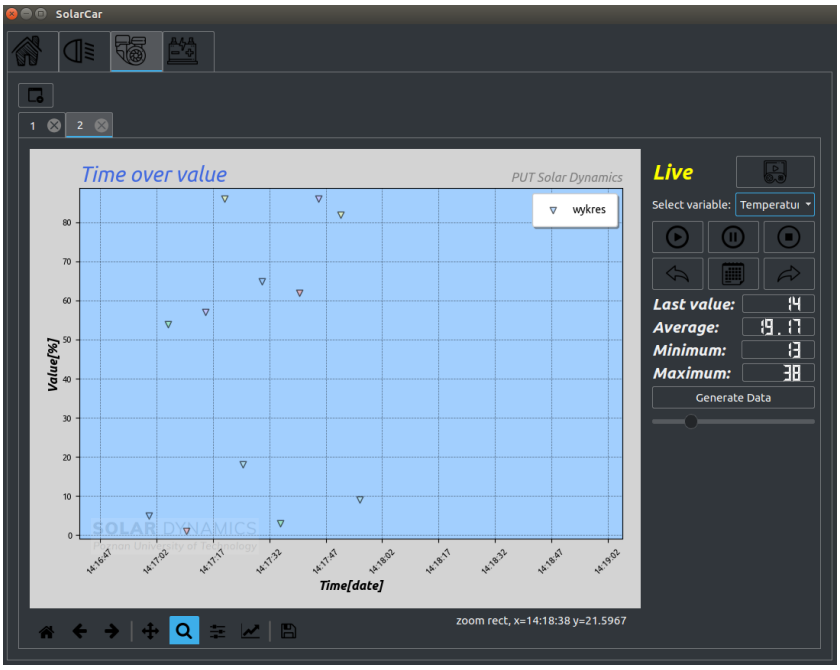

Rysunek 6: gui

Po kliknięciu w kalendarz możemy wybrać jakie dane chcemy zobaczyć (wybieramy date). Wcześniej na głównej zakładce musimy wybrać jakiego pomiaru to dotyczy.




Rysunek 7: gui

Tak wyglądają dane historyczne, można skalować przy pomocy narzędzie co znajdują się na dolnym pasku.




Rysunek 8: gui

# 4 Końcowe uwagi

Jakby się znalazła osoba co ma kontynuować apke to chętnie udzielę jakieś szczegółowe info jakby miała pytania :).