Опис команд управління таблом

# Загальні відомості

Управління таблом організоване на базі API-серверу, створеного на базі FastAPI - сучасного вебфреймворку для Python.

В контексті роботи даного технічного виробу, API-сервер дозволяє:

* надсилати текстові дані у форматі json для виводу на табло з програмного середовища Python;
* отримувати дані про поточні параметри роботи табло у форматі json (зворотний зв’язок із керуючою програмою);
* здійснювати контроль помилок запиту та валідацію даних, що надсилаються.

# Необхідні програмні компоненти і умови для здійснення управління

1. Середовище Python, версії 3.7 і вище;
2. Інстальований і імпортований в програмний застосунок управління модуль **requests**;
3. Мережева доступність IP-адреси і порту API-сервера з місця хостингу програмного застосунку управління таблом.

# Опис структури запиту

API-сервер табла підтримує наступні типи веб-запитів:

* GET-запити - для отримання поточних параметрів роботи табла, або результатів виконання запитів (зворотний зв’язок);
* PUT-запит - для оновлення текстових даних, що виводяться на табло.

## Варіанти GET-запитів

Існує 2 опції GET запиту: *запит повного статусу параметрів табла* та *запит статусу окремої зони табла.*

Отримана відповідь надходитиме у вигляді структурованих даних json.

### Запит повного статусу табла:

requests**.get**(http://**<ip>**:**<port>**/).json()

де:

<ip> - IPv4 адреса за якою доступний API-сервер;

<port> - значення порту, на якому працює API (за замовчуванням: *8000*);

Приклад виконання:

>>> requests.get("<http://192.168.111.106:8000/>").json()

{'dataset': {'0': {'text': ' ', 'brt': 4, 'addr': 8}, '1': {'text': '', 'brt': 127, 'addr': 9}, '2': {'text': '', 'brt': 127, 'addr': 10}, '3': {'text': '', 'brt': 127, 'addr': 11}, '4': {'text': '', 'brt': 127, 'addr': 12}}}

де:

'0'...'4' - ключ словника: ідентифікатор зони табла;

'text' - поточне значення тексту, що виводиться;

'brt' - поточне значення яскравості (4-255);

'addr' - адреса мікроконтролера зони на внутрішній шині обміну даними i2c.

Запит статусу окремої зони табла:

requests**.get**(http://**<ip>**:**<port>**/zone/**0**).json()

де:

<ip> - IPv4 адреса за якою доступний API-сервер;

<port> - значення порту, на якому працює API (за замовчуванням: *8000*);

0 - ідентифікатор зони табла: *0* - верхній ряд, *1* - другий згори, …,*4* - нижній;

Приклад виконання:

>>> requests.get(["http://192.168.111.106:8000/zone/0"](http://192.168.111.106:8000/zone/0)).json()

{'text': 'HELLO!', 'brt': 4, 'addr': 8}

Використання PUT-запиту. Вивід тексту на табло.

Структура запиту

requests**.put**("**http://<ip>:<port>/payload**", {“0”: “text0”, “1”: “text1”, “2”: “text2”, “3”: “text3”, “4”: “text4”}).json()

де:

“http://<ip>:<port>/payload” - типовий URL для задачі виводу текста на табло;

“0” … “4” - ідентифікатор відповідної зони табла;

“text0” … “text4” - довільний текст для відображення у зоні табло з відповідним ідентифікатором;

.json() - необов’язковий елемент, що необхідний для отримання статусу виконання запиту (зворотний зв’язок).

Приклади виконання:

>>>requests.put("[http://192.168.111.106:8000/payload](http://192.168.111.106:8000//payload/)", '{"0": " WINDOWS", "1": "MUST DIE"}').json()

**{}**

Після виконання цього запиту у верхній зоні табла з’явиться текст “WINDOWS” і фраза “MUST DIE” у зоні під нею. Інформація в інших зонах табла залишиться незмінною. Функція повертає керуючій програмі порожній python-словник {}.

Якщо в запиті для виводу тексту міститиметься більше ключів та їх значень, аніж фізично доступних текстових зон на табло, то текст “надлишкових” зон буде проігнорований, а надлишкові ключі та значення буде повернено у керуючу програму:

>>>requests.put("[http://192.168.111.106:8000/payload](http://192.168.111.106:8000//payload/)",'{"0": "WINDOWS", "1": "MUST DIE", "2": "RIDNA", "3": "UKRAYINA", "4": "PEREMOZHE", "5": "PUTIN", "6": "HUILO"}').json()

**{'5': 'PUTIN', '6': 'HUILO'}**

Вимоги до тексту, що надсилається на табло

Табло наразі не підтримує символи з адресацією поза межами ASCII.   
Тим не менш, пристрій спроможний відображати усі літери кириличної абетки. Щоправда, через це вивід літер має певні особливості і існують певні вимоги до тексту:

1. В обох абетках доступний вивід тільки символів верхнього регістру;
2. Вивід символів латинської абетки виконується символами верхнього регістру латинки. Символи нижнього регістру відсутні;
3. Символи, що спільні для обох абеток задаються символами верхнього регістру латинської абетки;
4. Кириличні символи, яких немає в латинці закодовані на символах нижнього регістру латинської розкладки із наступними відповідностями:

| **Cyrillic** | **Latin** | **Cyrillic** | **Latin** | **Cyrillic** | **Latin** |
| --- | --- | --- | --- | --- | --- |
| Б | b | И | y | Щ | w |
| Г | g | Л | l | Й | j |
| Д | d | П | p | Ю | o |
| Є | e | Ф | f | Я | q |
| Ж | h | Ц | c | Ь | і |
| З | z | Ч | t |  |  |
| Ї | r | Ш | s |  |  |

1. Текст мусить займати не більше 10 знакомісць на 1 зону;
2. Для очистки зони достатньо надіслати текст, що складається лише з “пробілу”;

Приклад виводу тексту кирилицею:

>>>requests.put("[http://192.168.111.106:8000/payload](http://192.168.111.106:8000//payload/)",'{"0": "bABOBHqHyj"}').json()

{}

Результат:

