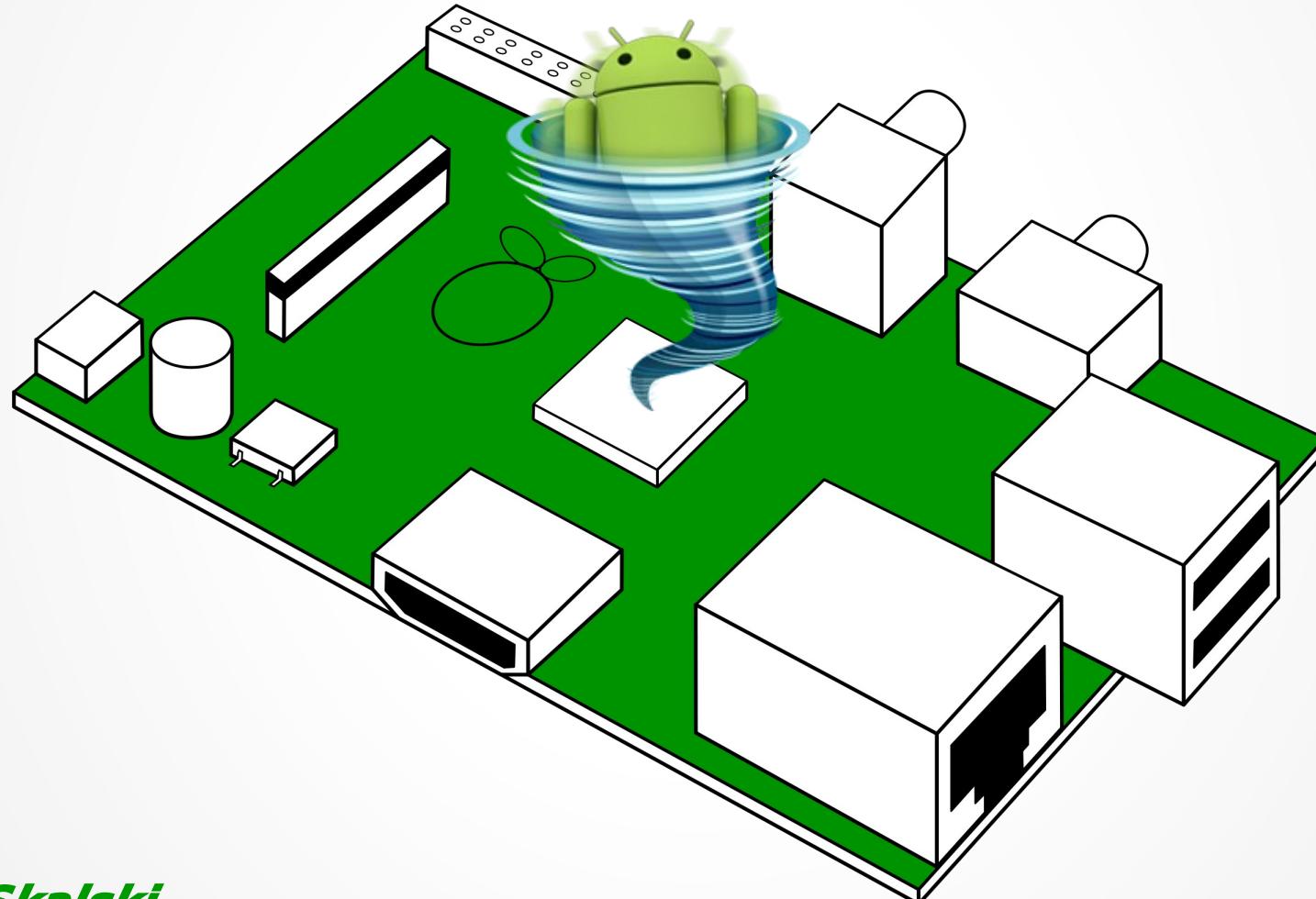


Raspberry Control - An Open Source and Open Hardware Smart Home



*Łukasz Skalski
lukasz.skalski@op.pl
www.lukasz-skalski.com*

Agenda

- Inteligentny dom - wybrane rozwiązania Open Source,
- Platformy *Open Hardware*,
- *Raspberry Control* - jeszcze jeden projekt typu HAS?
- *Raspberry Control* i Open Source,
- *Raspberry Control* a technologie “wearable”,
- Spoglądając wstecz - *Raspberry Control* w wersji 0.1 oraz 0.2,
- *Raspberry Control 2.0*:
 - architektura projektu,
 - aplikacja serwerowa,
 - aplikacja dla urządzeń z systemem Android,
 - aplikacja na urządzenia “wearable”,
- *Raspberry Control* - moduły wykonawcze i pomiarowe,
- Sesja Q&A

Inteligentny dom - wybrane rozwiązania Open Source

- *OpenHab,*



Inteligentny dom - wybrane rozwiązania Open Source

- *OpenHab,*



- *OpenRemote,*



Inteligentny dom - wybrane rozwiązania Open Source

- *OpenHab,*



- *OpenRemote,*



- *Freedomotic,*



Inteligentny dom - wybrane rozwiązania Open Source

- *OpenHab,*



- *OpenRemote,*



- *Freedomotic,*

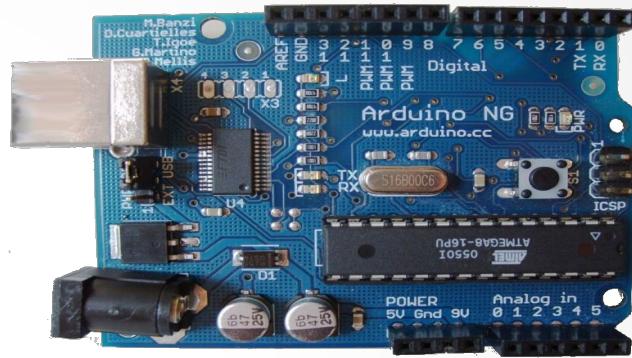


- *Calaos,*



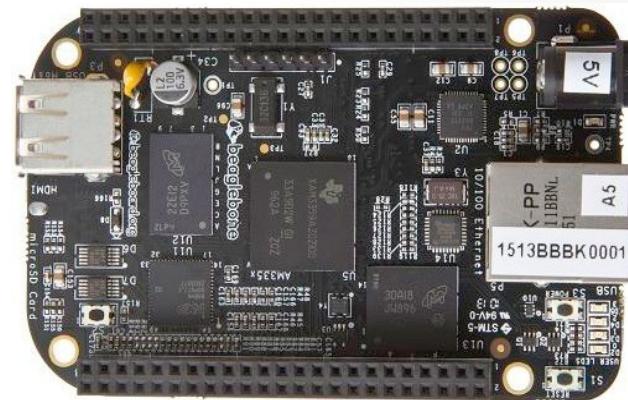
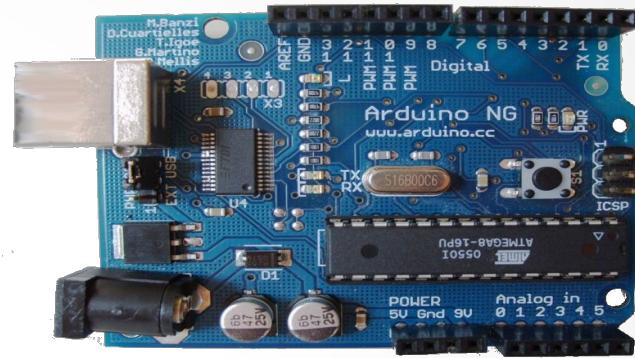
Platformy Open Hardware

- *Arduino*,



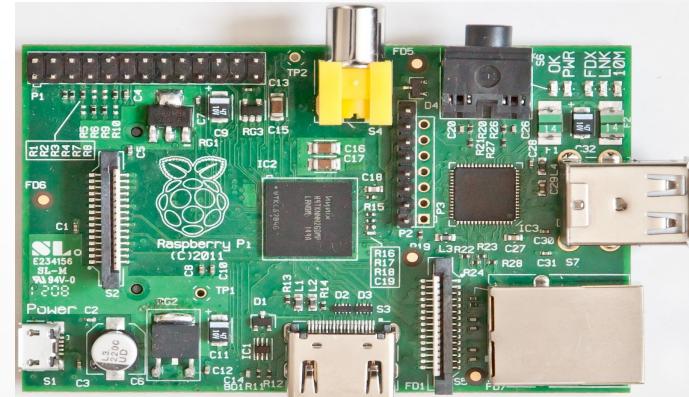
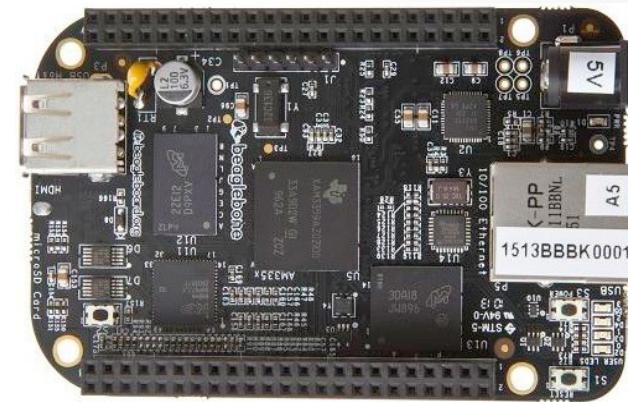
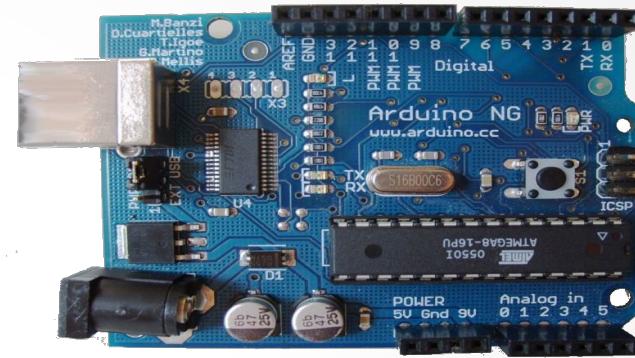
Platformy Open Hardware

- *Arduino*,
- *BeagleBone / BeagleBone Black*,



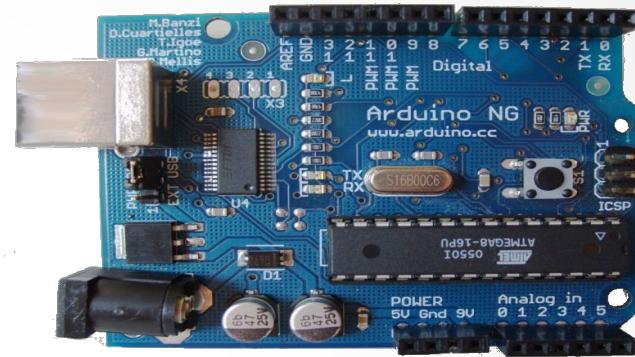
Platformy Open Hardware

- *Arduino*,
- *BeagleBone / BeagleBone Black*,
- *Raspberry Pi*,

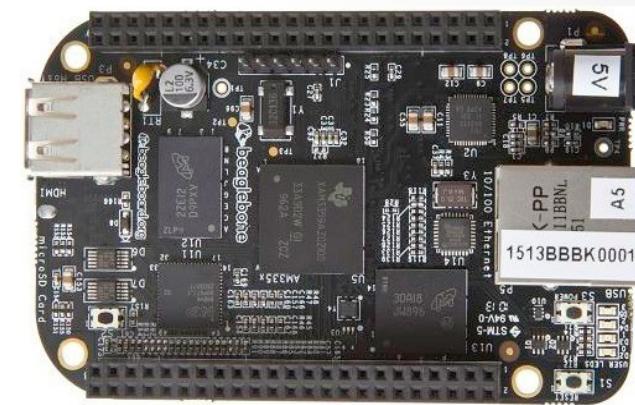


Platformy Open Hardware

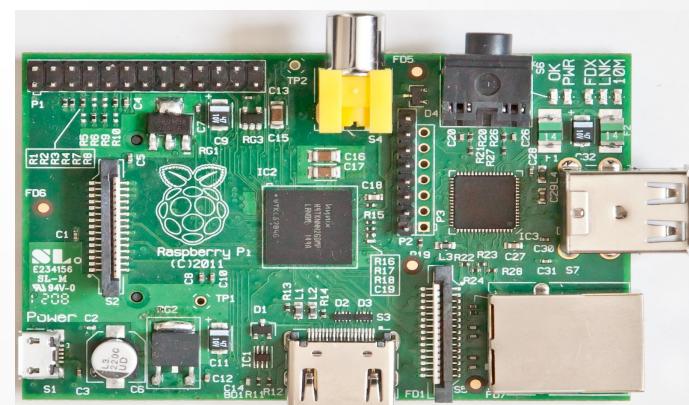
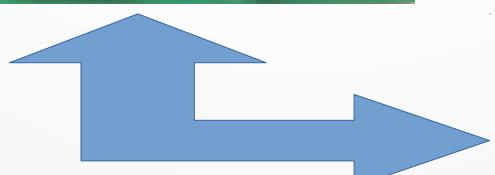
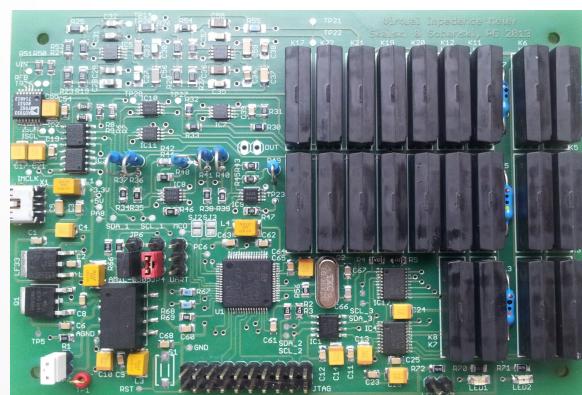
- *Arduino*,



- *BeagleBone / BeagleBone Black*,

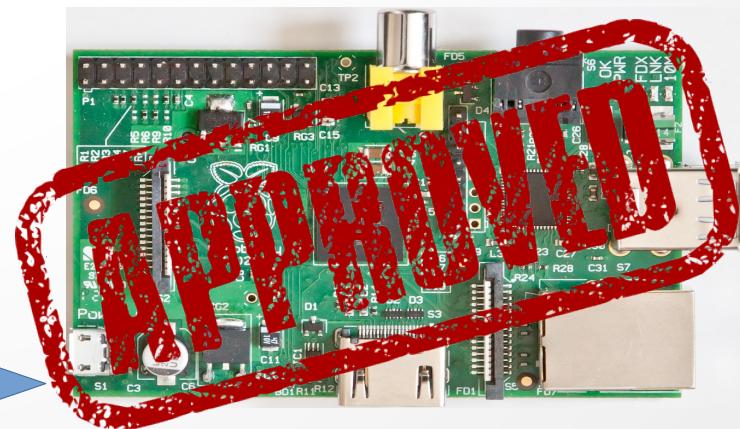
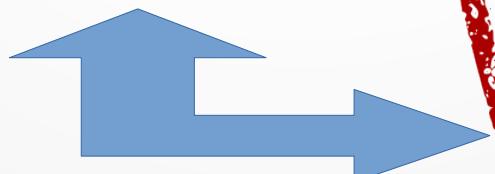
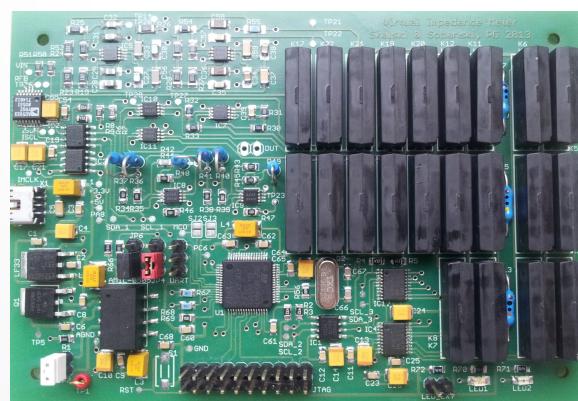
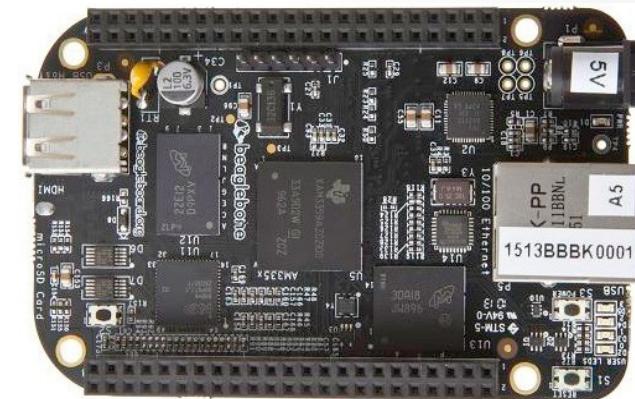
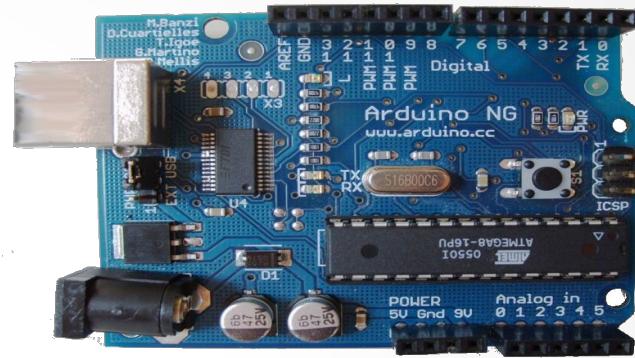


- *Raspberry Pi*,



Platformy Open Hardware

- *Arduino,*
- *BeagleBone / BeagleBone Black,*
- *Raspberry Pi,*



Raspberry Control - jeszcze jeden projekt typu *HAS*?

Czy naprawdę potrzebujemy jeszcze jednego projektu typu *HAS*?

Raspberry Control - jeszcze jeden projekt typu *HAS*?

Czy naprawdę potrzebujemy jeszcze jednego projektu typu *HAS*?



Raspberry Control - jeszcze jeden projekt typu *HAS*?

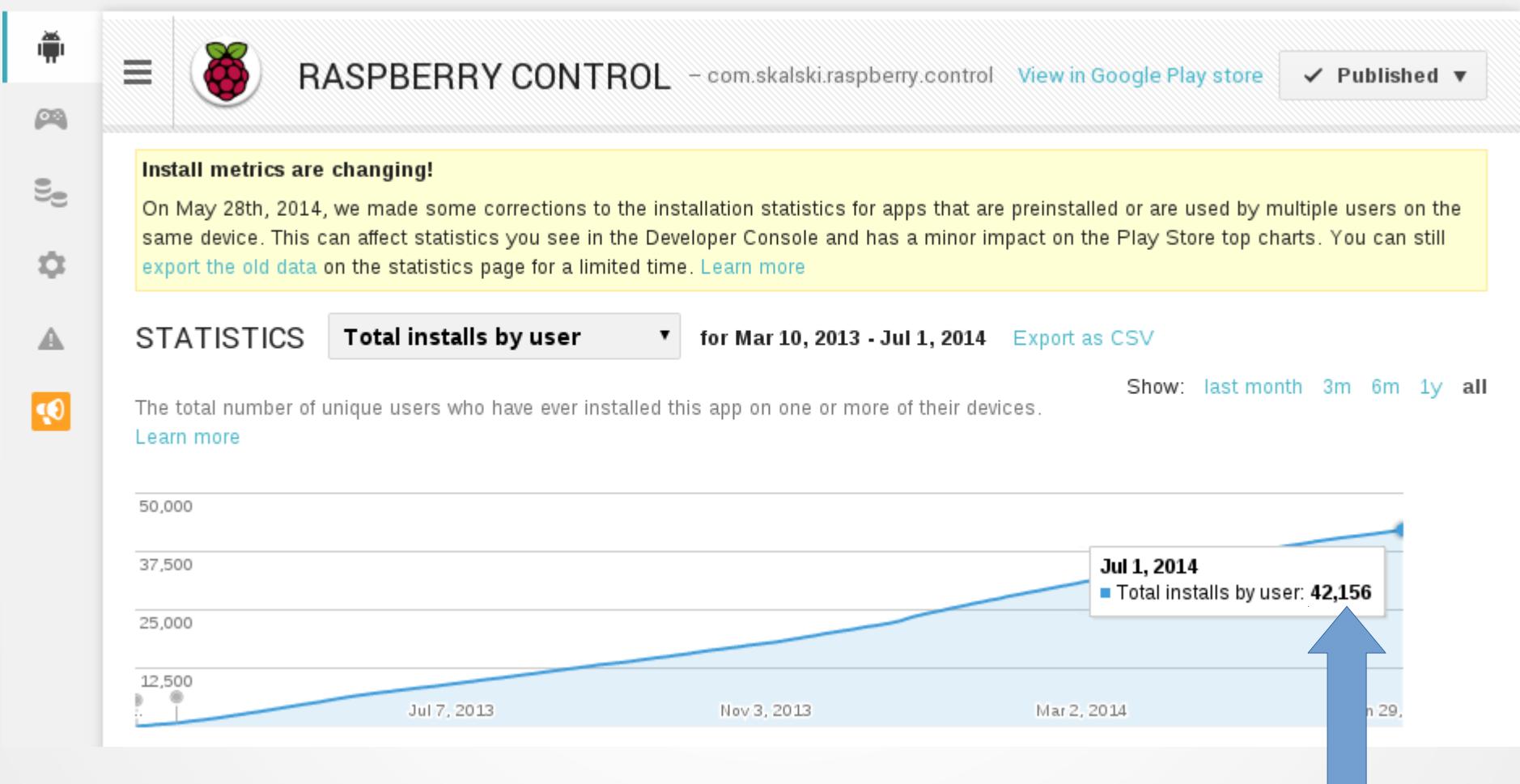
Czy naprawdę potrzebujemy jeszcze jednego projektu typu *HAS*?



Dlaczego więc rozpoczęliśmy pracę nad *Raspberry Control 2.0*?

Raspberry Control - jeszcze jeden projekt typu HAS?

- duża popularność pierwszej wersji aplikacji - prawie **45000 pobrań** w *Google Play!*,



Raspberry Control - jeszcze jeden projekt typu HAS?

- nieoczekiwany “rozgłos” w sieci,

FEB
7

[連結資料更新] Raspberry Control - 使用 Android 手機控制樹莓派

一段時間沒上 “Raspberry Control” 網站，剛發現他的網站出了一些問題，所以網頁中的樹莓派安裝程式無法下載。不過我已經將下載網址更新好了，現在可以直接下載了。

另外在我所提供的官方網頁 [LINK](#) 中，裡面同時也提供了樹莓派安裝程式的下載以及三個安裝 “Raspberry Control”的說明網頁，很榮幸！大家現在看到的這個網頁也是其中一個。

最新版本 : Version 0.2 (2013-03-24)
Raspberry Control 官方連結網址：LIKASZ-SKALSKI.COM ([LINK](#) 2014/02/17 網址更新)



from: likasz-skalski.com

介紹大家一個好用的軟體，這是安裝在 Android 裝置上面的軟體，可以經由它來控制與監控樹莓派，特點如下：



Accueil The MagPi Livre ENI Liens Articles Média-kit A propos

ARCHIVES DU MOT-CLEF RASPBERRY CONTROL

Contrôlez votre Raspberry Pi depuis un smartphone Android

Le 5 décembre 2013 by admin

Le père Noël étant passé un peu en avance, j'ai l'occasion de troquer un vieux téléphone sous Windows Mobile (ben oui nul n'est parfait...) contre un Galaxy S4 sous Android !

Que fait un RasPinaute qui a un smartphone Android ? Il essaie de le connecter sur son Raspberry Pi... C'est ce que j'ai fait et je vais, si vous le voulez bien, vous en conter l'histoire. Elle commence bien entendu [lire la suite](#) →

Publié dans [Android](#), [Documentation](#), [Framboise314](#), [Information](#) | Mots-clés : [android](#), [howto](#), [lukasz](#), [raspberry control](#), [Raspberry Pi](#), [raspberry pi how to](#), [shellinabox](#), [skalski](#) | 34 Réponses



Infinitely Gala –
A Linux
Beginners Blog!

Follow this beginners guide to
installing the latest fast and beautiful
next generation Linux.

How do I control my Raspberry Pi from my Android Phone?

1 JANUARY 2014 ALLEN KNIGHT

“Raspberry Control” now is available for **FREE** in Google Play!!!

Check out this Android app and create your own DCC controlled layout with “Raspberry Control” and a Sprogli, Hornby Elite or whatever.

Raspberry Control – Control Raspberry Pi with your Android Device 5

You have to download an app on your Android device, and you have to install a driver on your Raspberry Pi. The download link is down the page at the URL above. You can save this file on your main computer and drag it onto the SD card if you can't be fagged to do it on the Pi.

Mis apuntes de Raspberry Pi (a.k.a. My notes about the Pi)

News almost daily!
EN ES
Find... Go

News PIKISS About me

José Cerrejón Glez
Full time developer influenced by Apple, Linux, 80's Era and so on.
[Read more](#)

Raspberry Control application with Android

Raspberry Control application - examples



Raspberry Control - Control Raspberry Pi with your Android Device.

Application Features:

- secure connection via SSH,
- easy control and monitoring GPIO,
- monitor and graph temperatures remotely using DS18B20,

Google Docs' Notes
Raspberry Pi
Compilation in Spanglish of links and notes I'm finding about this interesting computer.
[link](#)

Raspberry Control - jeszcze jeden projekt typu HAS?

- pozytywne oceny i komentarze użytkowników aplikacji,

Blas M. on Sep 11, 2013 at 10:05 PM

Works beautiful and easy setup Galaxy S3 One of the better apps to control raspberry gpios, temperature sensor is a breeze too. All I need, is to learn how to rename/label gpios and change temp from celcius to farenheit. Thank you Lucasz Skalski!!!!!

Kemp M. on Apr 14, 2013 at 11:12 PM

Excellent! This took all of 5 minutes to set up!

Vinod M. on May 3, 2014 at 11:47 AM

Works pretty well waiting for the update !!!

Miguel Martinez G. on Dec 29, 2013 at 7:49 PM

Perfect app Thanks to developper to do that. It's easy to use and it works very well In my samsung

rakesh mg on Dec 28, 2013 at 8:10 PM

Excellent app Was easy to set up... Now one step closer to a remote controlled toy truck

Henry U. on Apr 14, 2014 at 9:35 PM

Perfect also for newcomer I'm a newcomer but with reading the installation hints I was also able to control the GPIO with this app at Samsung S4. Thanks !

Haydzthe C. on Jun 15, 2014 at 1:02 PM

Loving this amazing app but... This is the best raspberry pi control app that I have ever used but I would rate 5 stars if you added a terminal option and a raspberry pi camera feed through the app. But overall I'm loving it. Thanks Hayden



Raspberry Control - jeszcze jeden projekt typu HAS?

- liczne pytania użytkowników o możliwość implementacji nowych funkcjonalności i dodania obsługi nowego sprzętu,

Bjoe B. on Mar 12, 2013 at 9:18 PM

[..] Addons: *Would be nice to be able to tell a gpio to turn on or off when temp exceeds a value*
Option for tablet -larger or phone layout smaller Temperature function works fine Works on nexus 7

Marcin B. on May 28, 2013 at 1:19 PM

Great app!!! Great and powerfully application! I waiting for more options :)

Andre S. on May 28, 2013 at 8:25 PM

Should be great, But currently no Support for RaspBMC

Keenan S. on Jul 31, 2013 at 2:16 PM

Great app. Waiting for new features. It works with my Samsung p6200 (Samsung Galaxy Tab plus)

Vuk Z. on Sep 5, 2013 at 10:11 PM

DHT22 Hi, great app, can you tell me is support for DHT11 or DHT22 coming any time soon? regards

Mohamed H. on Nov 6, 2013 at 9:53 PM

Great Nice app. Must add gpio to widgets

Emmanuel G. V. on May 7, 2013 at 1:08 AM

Almost perfect The terminal don't work well. I can just write one command and then don't write nothing

Raspberry Control i Open Source

Priorytety dla *Raspberry Control 2.0*:

- współpraca ze społecznością Open Source:
 - całość projektu rozwijana na GitHub'ie (licencje *GPLv2* oraz *GPLv3*),
 - prosty i czytelny kod umożliwiający szybką jego analizę,
 - modułowa budowa ułatwiająca łatwą rozbudowę aplikacji,
- wsparcie dla urządzeń "wearable",



Raspberry Control - technologie “wearable”

Dlaczego wsparcie dla technologii “wearable”:

- + stanowią przyszłość elektroniki użytkowej i systemów HAS,



Raspberry Control - technologie “wearable”

Dlaczego wsparcie dla technologii “wearable”:

- + stanowią przyszłość elektroniki użytkowej i systemów HAS,
- + według przeprowadzonych badań [1] urządzenia “wearable” będą kluczową gałęzią elektroniki użytkowej przez najbliższe lata (według szacunków to 8mln dostarczonych urządzeń w 2014, 23mln w 2015 i 45mln w 2017),



[1] <http://techcrunch.com/2014/02/12/wearables-market-heating-up/>

Raspberry Control - technologie “wearable”

Dlaczego wsparcie dla technologii “wearable”:

- ✚ stanowią przyszłość elektroniki użytkowej i systemów HAS,
- ✚ według przeprowadzonych badań [1] urządzenia “wearable” będą kluczową gałęzią elektroniki użytkowej przez najbliższe lata (według szacunków to 8mln dostarczonych urządzeń w 2014, 23mln w 2015 i 45mln w 2017),
- ✚ rosnące zapotrzebowanie ze strony użytkowników:

[...] if you could make widgets for the android app that would be for individual GPIO control. I am using this to control lighting in my house and it would be nice to be able to turn them off and on without having to open the app every time.”,



[1] <http://techcrunch.com/2014/02/12/wearables-market-heating-up/>

Raspberry Control - technologie “wearable”

Dlaczego wsparcie dla technologii “wearable”:

- ✚ stanowią przyszłość elektroniki użytkowej i systemów HAS,
- ✚ według przeprowadzonych badań [1] urządzenia “wearable” będą kluczową gałęzią elektroniki użytkowej przez najbliższe lata (według szacunków to 8mln dostarczonych urządzeń w 2014, 23mln w 2015 i 45mln w 2017),
- ✚ rosnące zapotrzebowanie ze strony użytkowników:

[...] if you could make widgets for the android app that would be for individual GPIO control. I am using this to control lighting in my house and it would be nice to be able to turn them off and on without having to open the app every time.

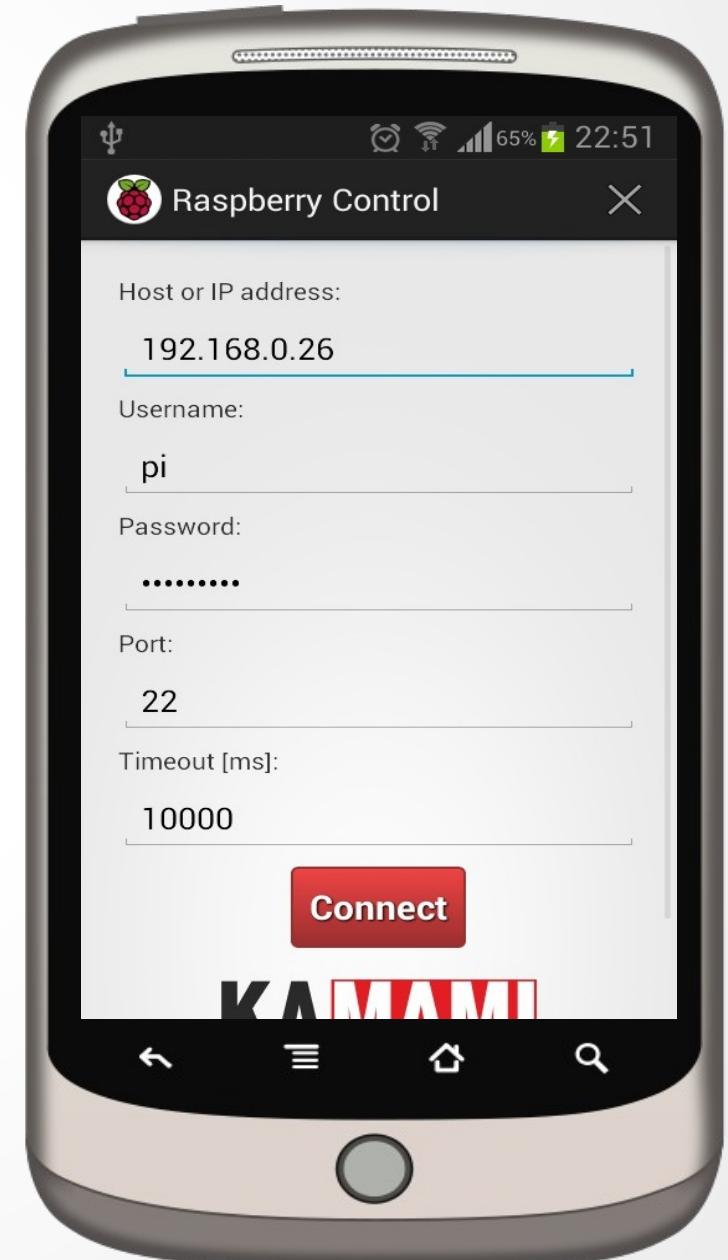
- ✚ nowe wyzwanie dla programistów pod kątem tworzenia UI (aplikacje bazujące na kontekście, dostarczające minimum wymaganych funkcji i informacji),



Spoglądając wstecz - Raspberry Control w wersji 0.1 oraz 0.2

Raspberry Control w wersjach 0.1 oraz 0.2 to:

- szyfrowane połączenie (protokół SSH),



Spoglądając wstecz - Raspberry Control w wersji 0.1 oraz 0.2

Raspberry Control w wersjach 0.1 oraz 0.2 to:

- szyfrowane połączenie (protokół SSH),
- proste i intuicyjne menu,



Spoglądając wstecz - Raspberry Control w wersji 0.1 oraz 0.2

Raspberry Control w wersjach 0.1 oraz 0.2 to:

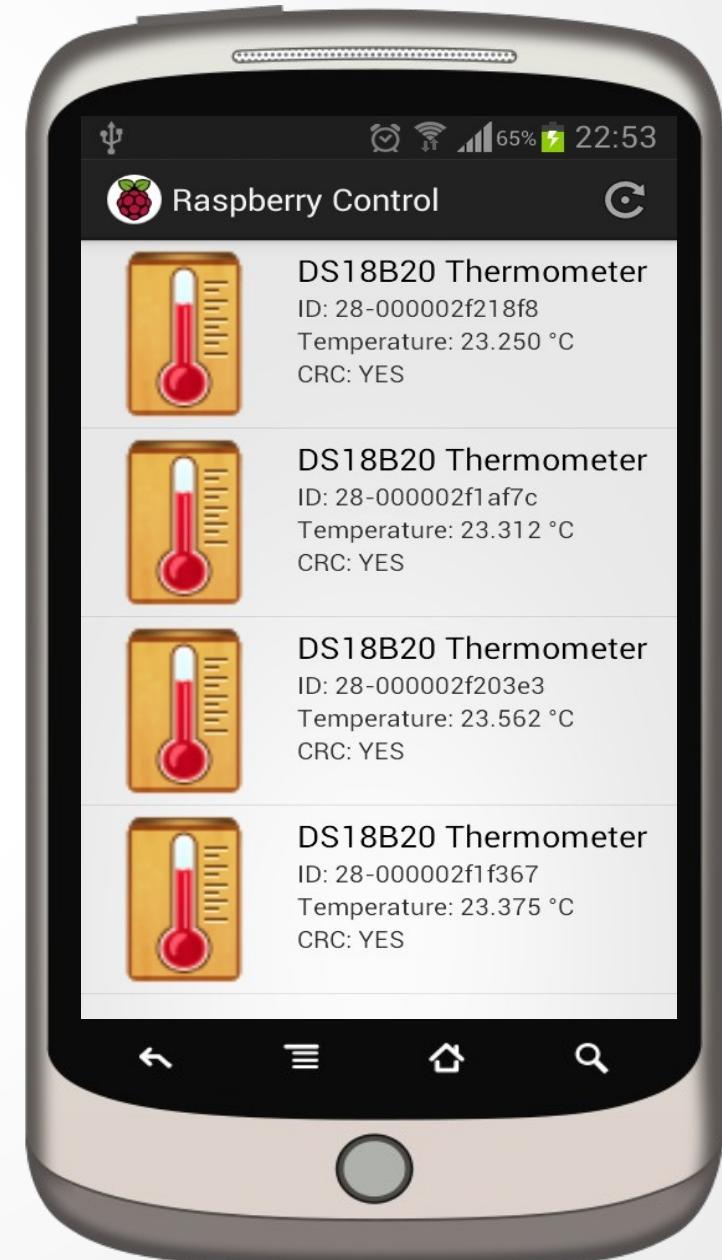
- szyfrowane połączenie (protokół SSH),
- proste i intuicyjne menu,
- zapis/odczyt wyprowadzeń GPIO,



Spoglądając wstecz - Raspberry Control w wersji 0.1 oraz 0.2

Raspberry Control w wersjach 0.1 oraz 0.2 to:

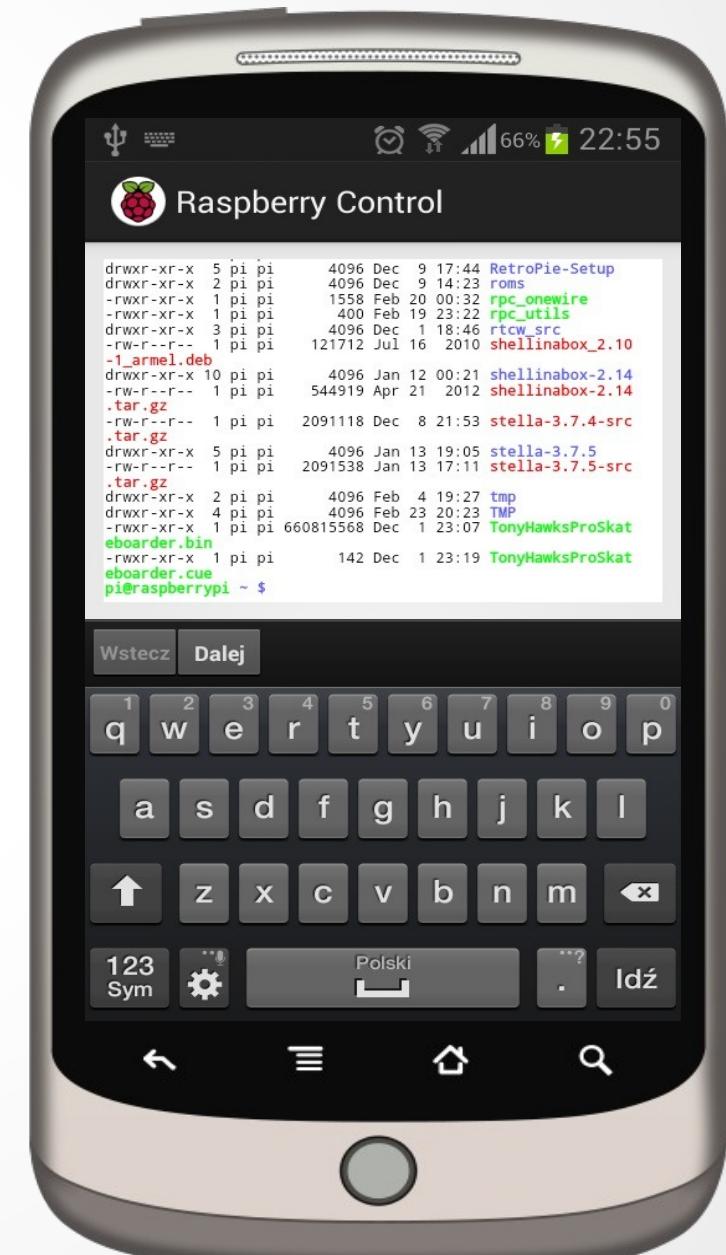
- szyfrowane połączenie (protokół SSH),
- proste i intuicyjne menu,
- zapis/odczyt wyprowadzeń GPIO,
- zdalny monitoring temperatury (DS18B20),



Spoglądając wstecz - Raspberry Control w wersji 0.1 oraz 0.2

Raspberry Control w wersjach 0.1 oraz 0.2 to:

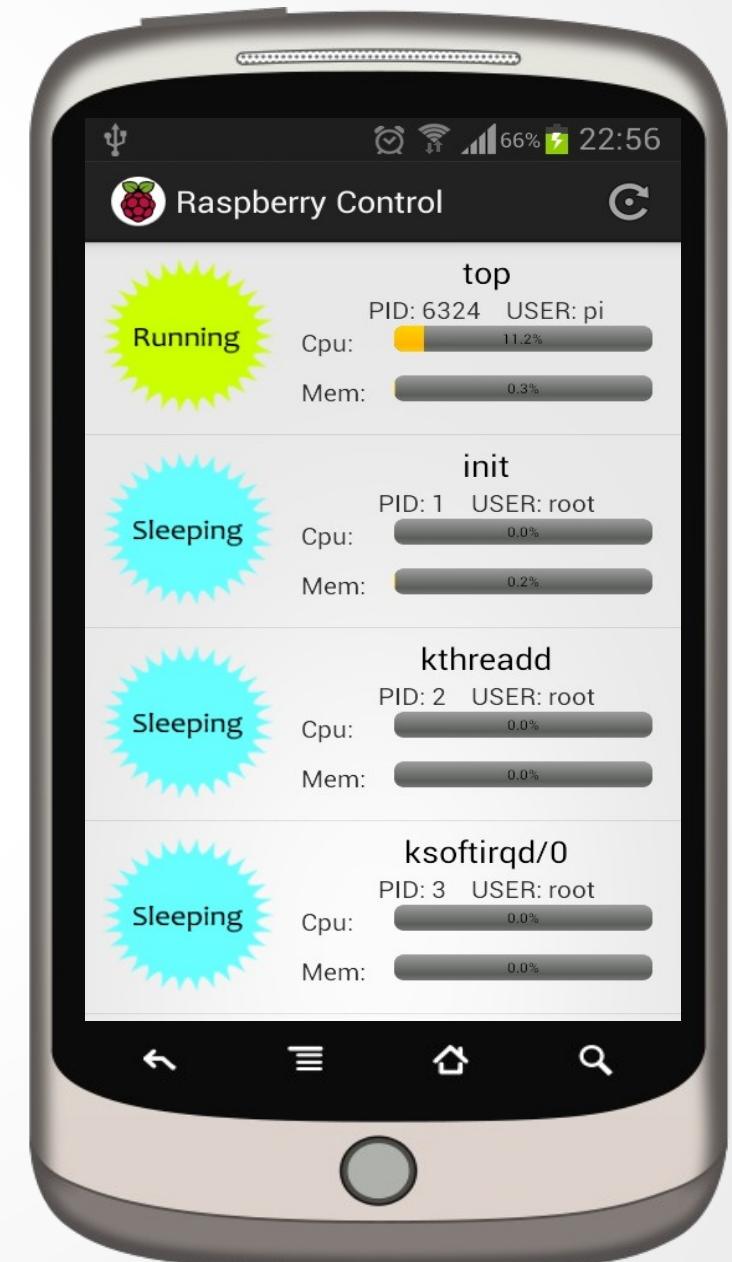
- ✚ szyfrowane połączenie (protokół SSH),
- ✚ proste i intuicyjne menu,
- ✚ zapis/odczyt wyprowadzeń GPIO,
- ✚ zdalny monitoring temperatury (DS18B20),
- ✚ zdalny emulator terminala,



Spoglądając wstecz - Raspberry Control w wersji 0.1 oraz 0.2

Raspberry Control w wersjach 0.1 oraz 0.2 to:

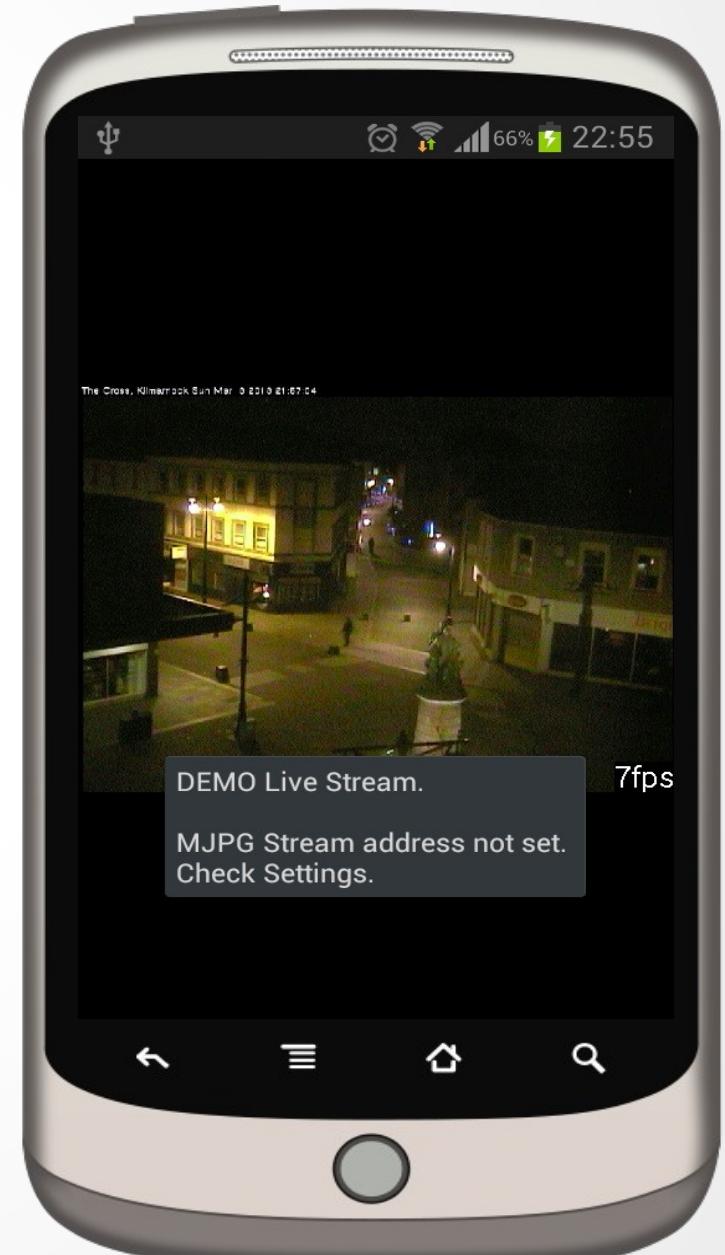
- szyfrowane połączenie (protokół SSH),
- proste i intuicyjne menu,
- zapis/odczyt wyprowadzeń GPIO,
- zdalny monitoring temperatury (DS18B20),
- zdalny emulator terminala,
- zdalne zarządzanie procesami,



Spoglądając wstecz - Raspberry Control w wersji 0.1 oraz 0.2

Raspberry Control w wersjach 0.1 oraz 0.2 to:

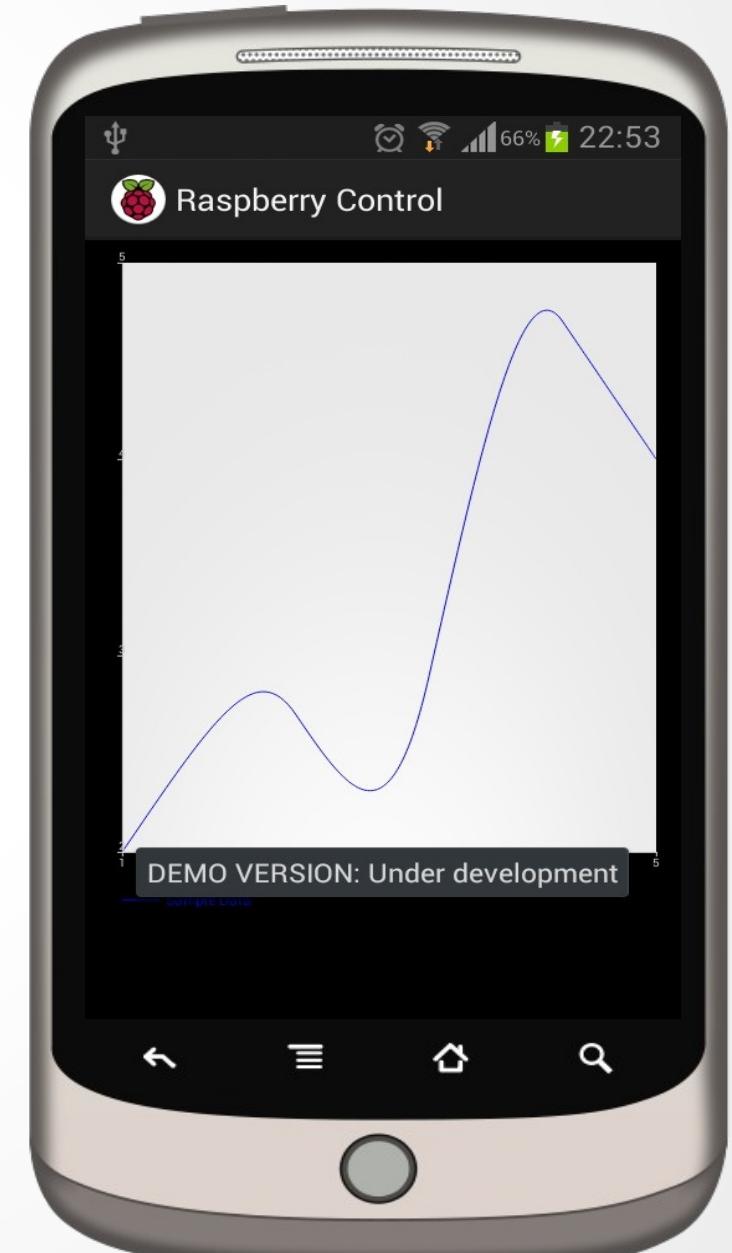
- ✚ szyfrowane połączenie (protokół SSH),
- ✚ proste i intuicyjne menu,
- ✚ zapis/odczyt wyprowadzeń GPIO,
- ✚ zdalny monitoring temperatury (DS18B20),
- ✚ zdalny emulator terminala,
- ✚ zdalne zarządzanie procesami,
- ✚ wbudowany klient MJPEG (streaming obrazu),



Spoglądając wstecz - Raspberry Control w wersji 0.1 oraz 0.2

Raspberry Control w wersjach 0.1 oraz 0.2 to:

- ✚ szyfrowane połączenie (protokół SSH),
- ✚ proste i intuicyjne menu,
- ✚ zapis/odczyt wyprowadzeń GPIO,
- ✚ zdalny monitoring temperatury (DS18B20),
- ✚ zdalny emulator terminala,
- ✚ zdalne zarządzanie procesami,
- ✚ wbudowany klient MJPEG (streaming obrazu),
- ✚ zapis/odczyt magistral 1-Wire oraz I2C,



Spoglądając wstecz - Raspberry Control w wersji 0.1 oraz 0.2

Raspberry Control w wersjach 0.1 oraz 0.2 to:

- ✚ szyfrowane połączenie (protokół SSH),
- ✚ proste i intuicyjne menu,
- ✚ zapis/odczyt wyprowadzeń GPIO,
- ✚ zdalny monitoring temperatury (DS18B20),
- ✚ zdalny emulator terminala,
- ✚ zdalne zarządzanie procesami,
- ✚ wbudowany klient MJPEG (streaming obrazu),
- ✚ zapis/odczyt magistral 1-Wire oraz I2C,
- ✚ sterowanie urządzeniami z wykorzystaniem IR,



Raspberry Control 2.0 - techniczne założenia projektowe

- odejście od jednokierunkowego modelu “*pytanie-odpowiedź*”, uniemożliwiającego implementację systemu asynchronicznych powiadomień przesyłanych do podłączonych klientów,

Raspberry Control 2.0 - techniczne założenia projektowe

- odejście od jednokierunkowego modelu “*pytanie-odpowiedź*”, uniemożliwiającego implementację systemu asynchronicznych powiadomień przesyłanych do podłączonych klientów,
- wykluczenie metody aktywnego “*pollingu*” jako metody nieoptymalnej, zwłaszcza dla urządzeń mobilnych i urządzeń z gałęzi “*wearable*”,

Raspberry Control 2.0 - techniczne założenia projektowe

- odejście od jednokierunkowego modelu “*pytanie-odpowiedź*”, uniemożliwiającego implementację systemu asynchronicznych powiadomień przesyłanych do podłączonych klientów,
- wykluczenie metody aktywnego “*pollingu*” jako metody nieoptymalnej, zwłaszcza dla urządzeń mobilnych i urządzeń z gałęzi “*wearable*”,
- potrzeba implementacji wydajnej, dwukierunkowej komunikacji generującej niskie obciążenie sieci oraz zużycie baterii - brak blokujących wywołań,

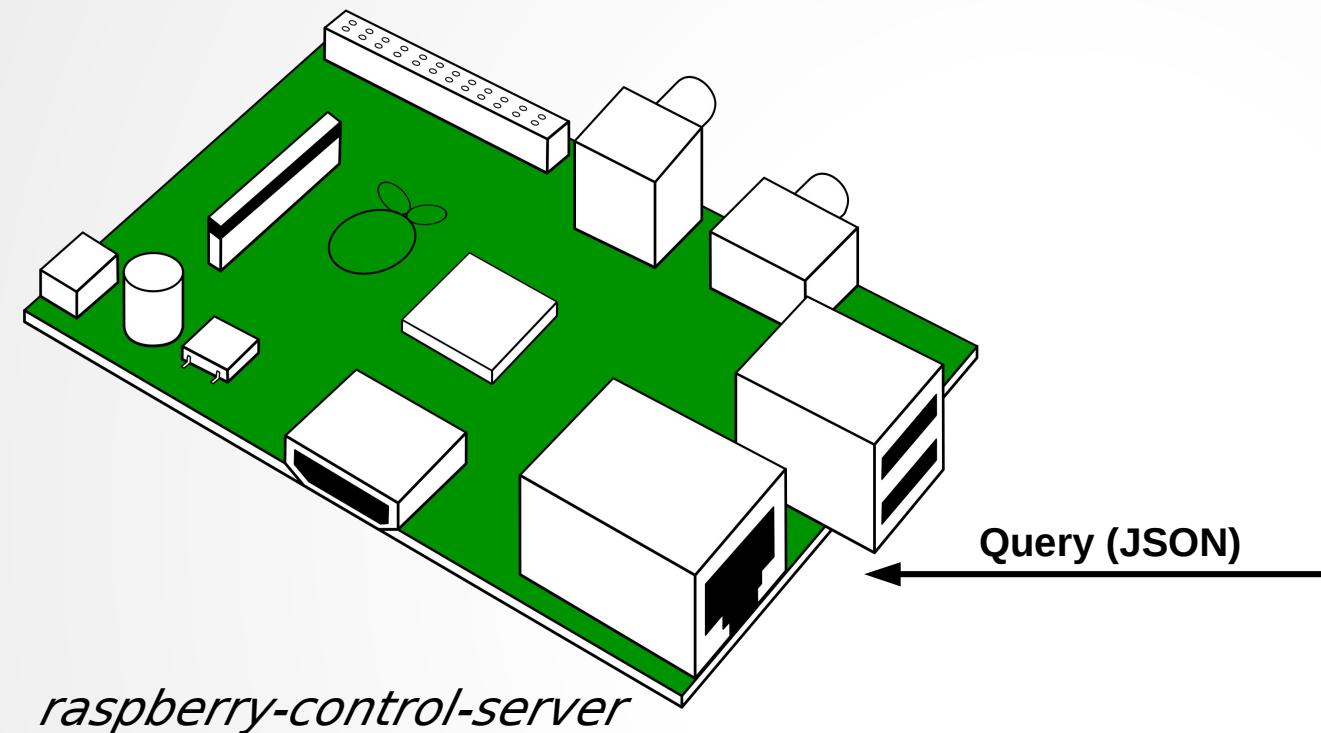
Raspberry Control 2.0 - techniczne założenia projektowe

- odejście od jednokierunkowego modelu “*pytanie-odpowiedź*”, uniemożliwiającego implementację systemu asynchronicznych powiadomień przesyłanych do podłączonych klientów,
- wykluczenie metody aktywnego “*pollingu*” jako metody nieoptymalnej, zwłaszcza dla urządzeń mobilnych i urządzeń z gałęzi “*wearable*”,
- potrzeba implementacji wydajnej, dwukierunkowej komunikacji generującej niskie obciążenie sieci oraz zużycie baterii - brak blokujących wywołań,
- wyspecyfikowanie ustandaryzowanego formatu wymiany danych pomiędzy wszystkimi podłączonymi urządzeniami - *JSON*,

Raspberry Control 2.0 - techniczne założenia projektowe

- odejście od jednokierunkowego modelu “*pytanie-odpowiedź*”, uniemożliwiającego implementację systemu asynchronicznych powiadomień przesyłanych do podłączonych klientów,
- wykluczenie metody aktywnego “*pollingu*” jako metody nieoptymalnej, zwłaszcza dla urządzeń mobilnych i urządzeń z gałęzi “*wearable*”,
- potrzeba implementacji wydajnej, dwukierunkowej komunikacji generującej niskie obciążenie sieci oraz zużycie baterii - brak blokujących wywołań,
- wyspecyfikowanie ustandaryzowanego formatu wymiany danych pomiędzy wszystkimi podłączonymi urządzeniami - *JSON*,
- przenośność oprogramowania serwera pomiędzy różne platformy sprzętowe (*Raspberry Pi*, *Odroid*, *BeagleBone Black*, ...), poprzez wykorzystanie standardowych interfejsów jądra Linux,

Raspberry Control 2.0 - architektura

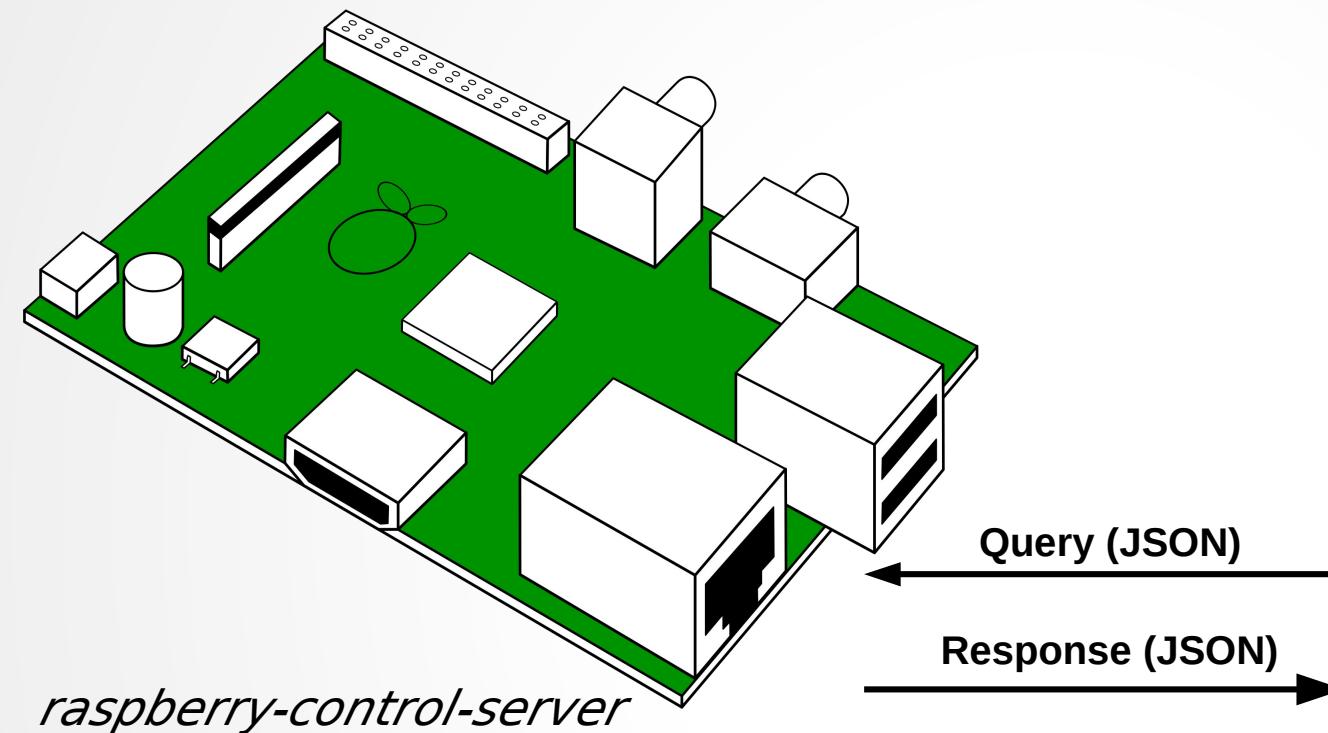


raspberry-control-wearable-tizen

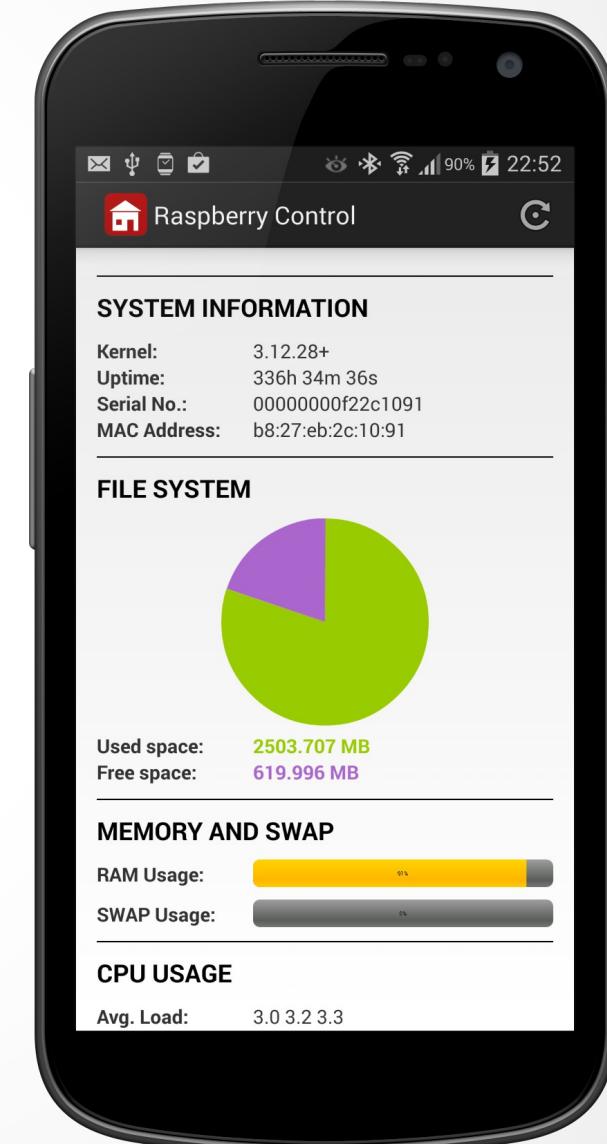


raspberry-control-client

Raspberry Control 2.0 - architektura

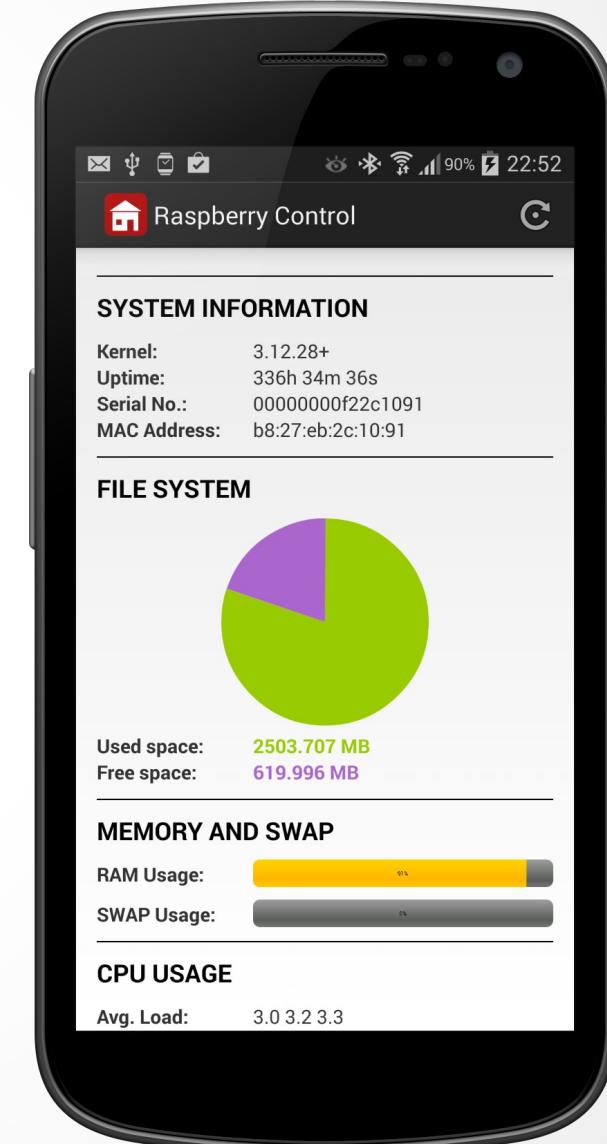
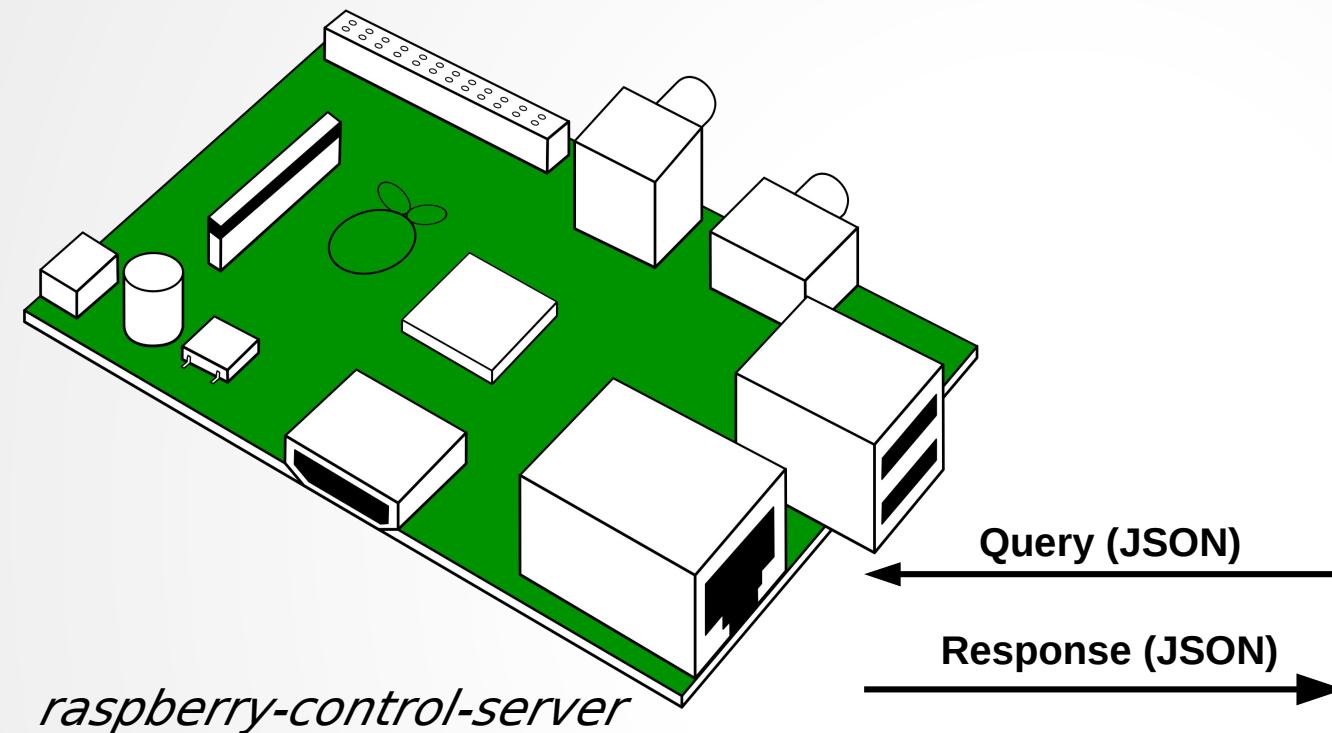


raspberry-control-wearable-tizen



raspberry-control-client

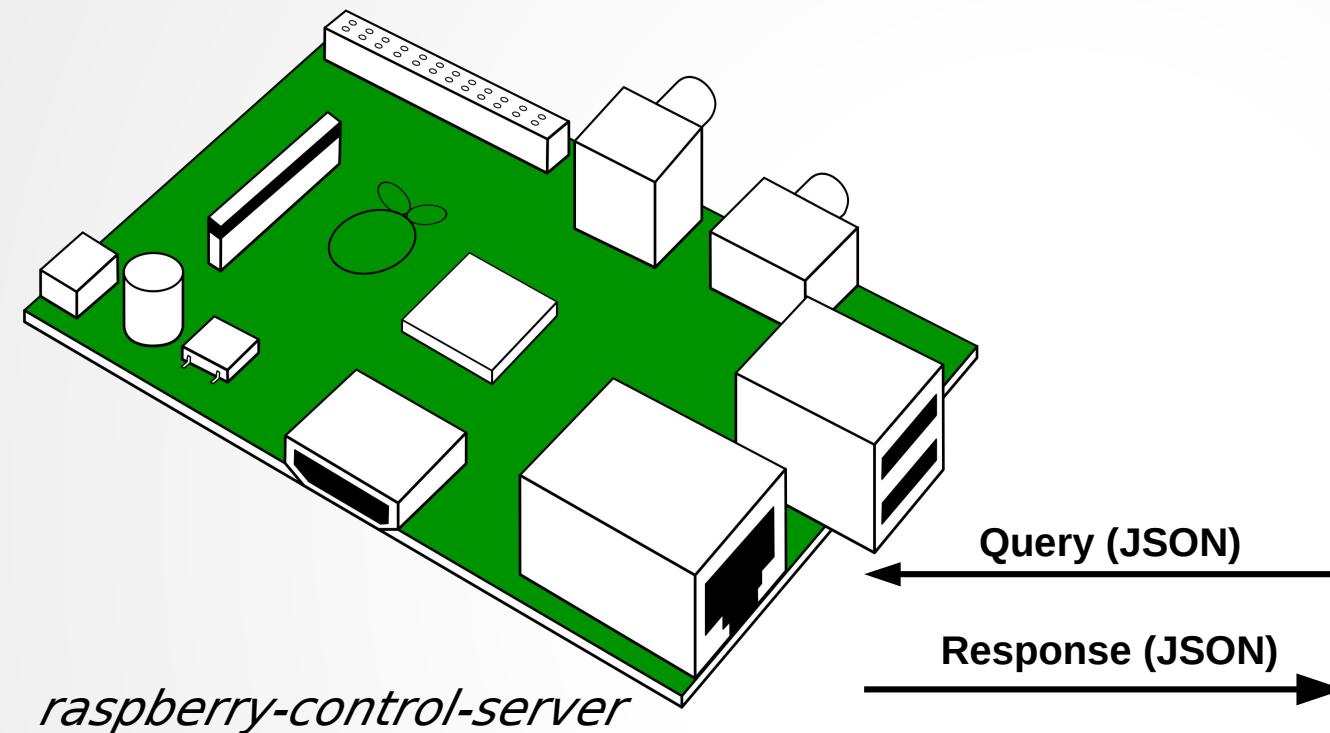
Raspberry Control 2.0 - architektura



raspberry-control-wearable-tizen

raspberry-control-client

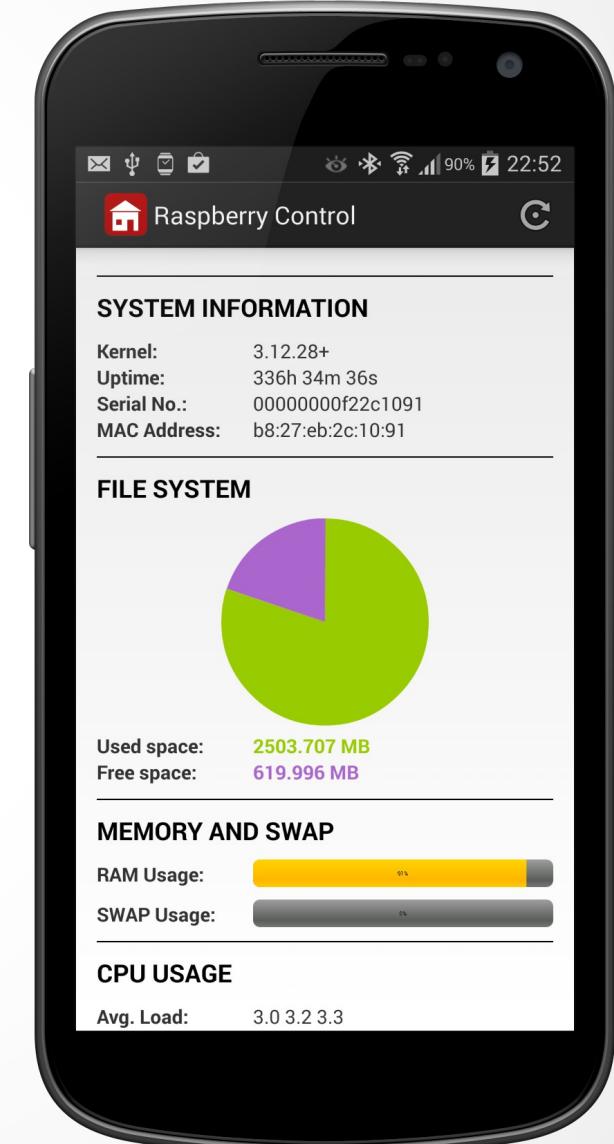
Raspberry Control 2.0 - architektura



raspberry-control-server

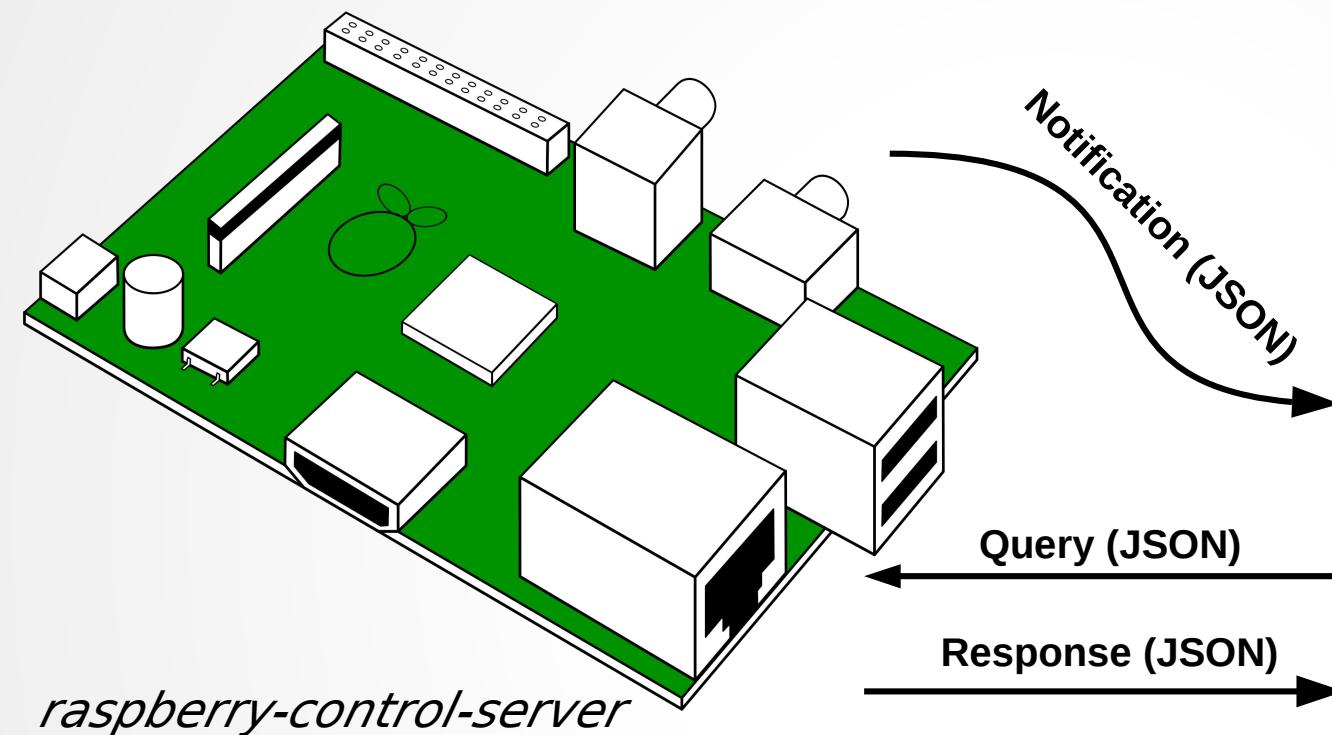


raspberry-control-wearable-tizen

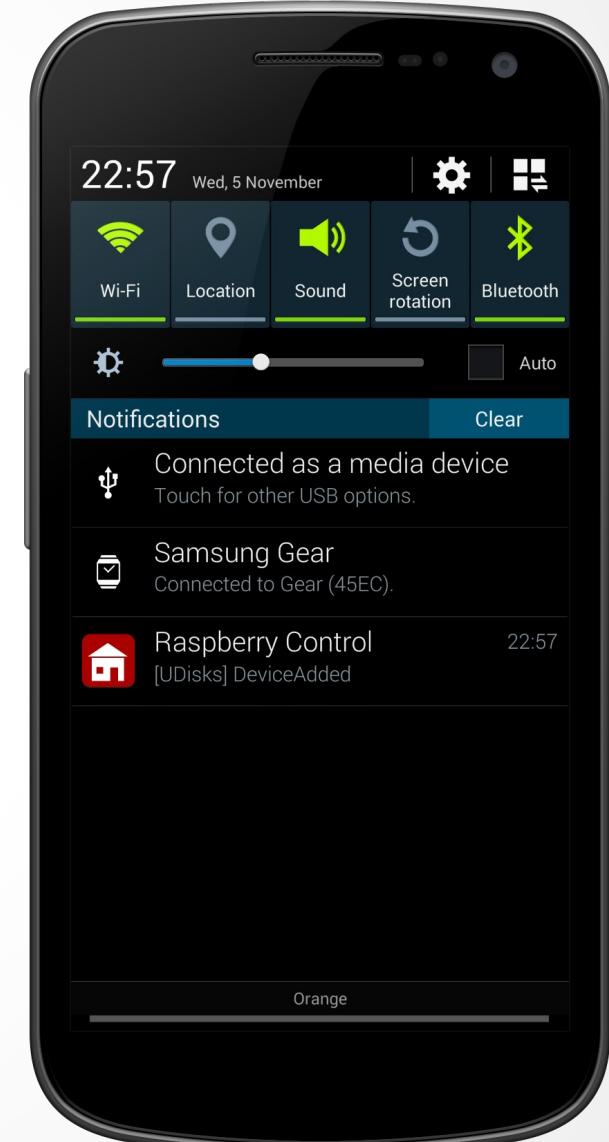


raspberry-control-client

Raspberry Control 2.0 - architektura



raspberry-control-server



raspberry-control-client



raspberry-control-wearable-tizen

Raspberry Control 2.0 - serwer

- aplikacja napisana od podstaw w języku C z wykorzystaniem bibliotek *glib-2.0*, *libwebsocket* oraz *Jansson*,

Raspberry Control 2.0 - serwer

- aplikacja napisana od podstaw w języku C z wykorzystaniem bibliotek *glib-2.0*, *libwebsocket* oraz *Jansson*,
- w chwili obecnej aplikacja serwera to tylko ~1700 linii kodu,

Raspberry Control 2.0 - serwer

- aplikacja napisana od podstaw w języku C z wykorzystaniem bibliotek *glib-2.0*, *libwebsocket* oraz *Jansson*,
- w chwili obecnej aplikacja serwera to tylko ~1700 linii kodu,
- do głównych zadań aplikacji należy generowanie odpowiedzi na polecenia przychodzące ze strony podłączonych klientów, monitorowanie zasobów sprzętowych oraz wysyłanie rozgłoszeniowych notyfikacji,

Raspberry Control 2.0 - serwer

- aplikacja napisana od podstaw w języku C z wykorzystaniem bibliotek *glib-2.0*, *libwebsocket* oraz *Jansson*,
- w chwili obecnej aplikacja serwera to tylko ~1700 linii kodu,
- do głównych zadań aplikacji należy generowanie odpowiedzi na polecenia przychodzące ze strony podłączonych klientów, monitorowanie zasobów sprzętowych oraz wysyłanie rozgłoszeniowych notyfikacji,
- generyczny kod ułatwiający portowanie aplikacji na inne platformy sprzętowe oraz dystrybucje,

Raspberry Control 2.0 - serwer

- aplikacja napisana od podstaw w języku C z wykorzystaniem bibliotek *glib-2.0*, *libwebsocket* oraz *Jansson*,
- w chwili obecnej aplikacja serwera to tylko ~1700 linii kodu,
- do głównych zadań aplikacji należy generowanie odpowiedzi na polecenia przychodzące ze strony podłączonych klientów, monitorowanie zasobów sprzętowych oraz wysyłanie rozgłoszeniowych notyfikacji,
- generyczny kod ułatwiający portowanie aplikacji na inne platformy sprzętowe oraz dystrybucje,
- automatyczne zarządzenia procesem kompilacji poprzez *CMake*,

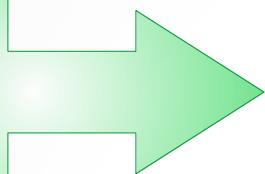
Raspberry Control 2.0 - serwer

- aplikacja napisana od podstaw w języku C z wykorzystaniem bibliotek *glib-2.0*, *libwebsocket* oraz *Jansson*,
- w chwili obecnej aplikacja serwera to tylko ~1700 linii kodu,
- do głównych zadań aplikacji należy generowanie odpowiedzi na polecenia przychodzące ze strony podłączonych klientów, monitorowanie zasobów sprzętowych oraz wysyłanie rozgłoszeniowych notyfikacji,
- generyczny kod ułatwiający portowanie aplikacji na inne platformy sprzętowe oraz dystrybucje,
- automatyczne zarządzanie procesem komplikacji poprzez *CMake*,
- projekty niebędące integralną częścią serwera, ale uzupełniające jego funkcjonalność:
 - *shellinaboxd*,
 - *mjpg-streamer*,
 - *lircd*,

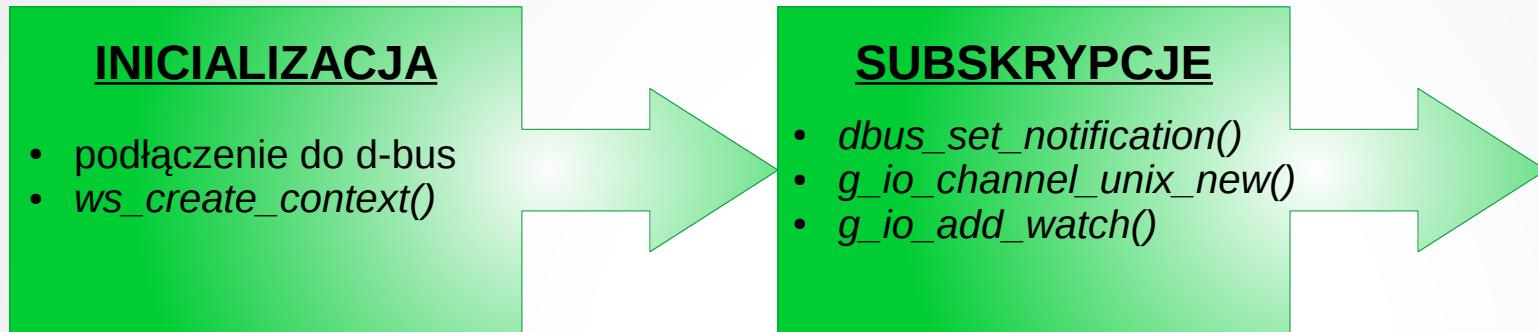
Raspberry Control 2.0 - serwer

INICJALIZACJA

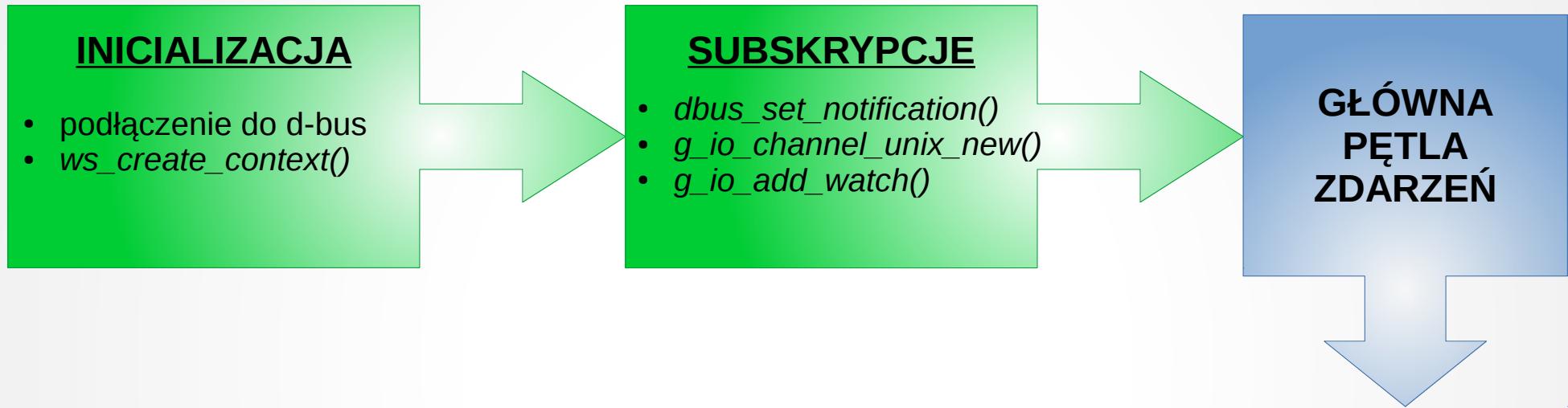
- połączenie do d-bus
- *ws_create_context()*



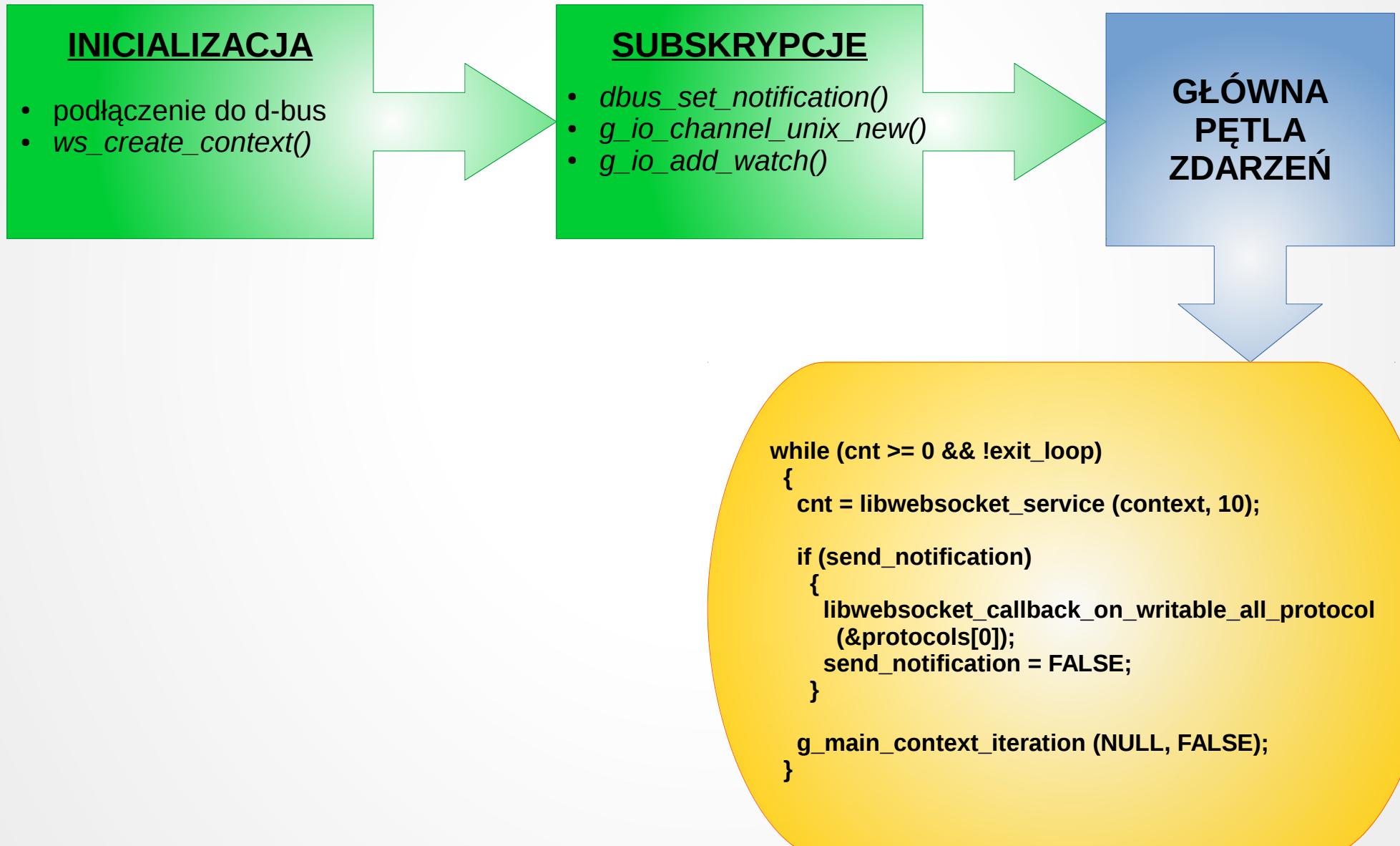
Raspberry Control 2.0 - serwer



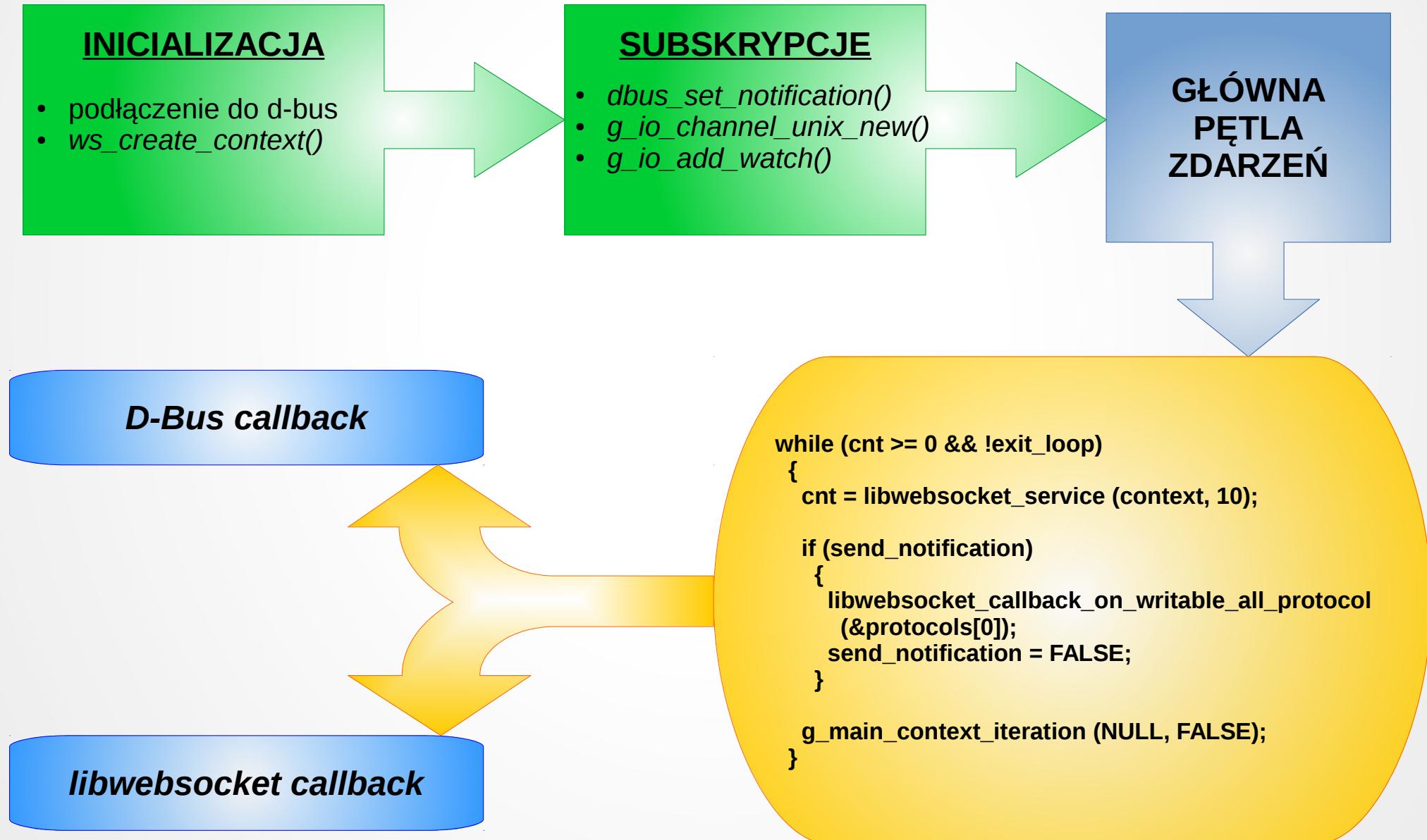
Raspberry Control 2.0 - serwer



Raspberry Control 2.0 - serwer



Raspberry Control 2.0 - serwer



Raspberry Control 2.0 - serwer

```
raspberry_control_callback ()
{
    switch (reason)
    {

        case LWS_CALLBACK_ESTABLISHED:
            /* connection established */
            break;

        case LWS_CALLBACK_CLOSED:
            /* connection closed */
            break;

        case LWS_CALLBACK_SERVER_WRITEABLE:
            /* send data to client */
            break;

        case LWS_CALLBACK_RECEIVE:
            psd->len = parse_json (received_data);
            if (psd->len > 0)
                libwebsocket_callback_on_writable ();
            break;

    }
}
```

Raspberry Control 2.0 - serwer

```
raspberry_control_callback ()  
{  
    switch (reason)  
    {  
  
        case LWS_CALLBACK_ESTABLISHED:  
            /* connection established */  
            break;  
  
        case LWS_CALLBACK_CLOSED:  
            /* connection closed */  
            break;  
  
        case LWS_CALLBACK_SERVER_WRITEABLE:  
            /* send data to client */  
            break;  
  
        case LWS_CALLBACK_RECEIVE:  
            psd->len = parse_json (received_data);  
            if (psd->len > 0)  
                libwebsocket_callback_on_writable ();  
            break;  
  
    }  
}
```



```
{  
    "RunCommand": {  
        "cmd": "CommandName",  
        "args": "args"  
    }  
}
```

Raspberry Control 2.0 - serwer

```
raspberry_control_callback ()  
{  
    switch (reason)  
    {  
  
        case LWS_CALLBACK_ESTABLISHED:  
            /* connection established */  
            break;  
  
        case LWS_CALLBACK_CLOSED:  
            /* connection closed */  
            break;  
  
        case LWS_CALLBACK_SERVER_WRITEABLE:  
            /* send data to client */  
            break;  
  
        case LWS_CALLBACK_RECEIVE:  
            psd->len = parse_json (received_data);  
            if (psd->len > 0)  
                libwebsocket_callback_on_writable ();  
            break;  
  
    }  
}
```

```
{  
  
    "RunCommand": {  
  
        "cmd": "CommandName",  
  
        "args": "args"  
  
    }  
}
```

cmd:
+ GetGPIO,

Raspberry Control 2.0 - serwer

```
raspberry_control_callback ()  
{  
    switch (reason)  
    {  
  
        case LWS_CALLBACK_ESTABLISHED:  
            /* connection established */  
            break;  
  
        case LWS_CALLBACK_CLOSED:  
            /* connection closed */  
            break;  
  
        case LWS_CALLBACK_SERVER_WRITEABLE:  
            /* send data to client */  
            break;  
  
        case LWS_CALLBACK_RECEIVE:  
            psd->len = parse_json (received_data);  
            if (psd->len > 0)  
                libwebsocket_callback_on_writable ();  
            break;  
  
    }  
}
```

```
{  
  
    "RunCommand": {  
  
        "cmd": "CommandName",  
  
        "args": "args"  
  
    }  
}
```

cmd:

- + GetGPIO,
- + SetGPIO,

Raspberry Control 2.0 - serwer

```
raspberry_control_callback ()  
{  
    switch (reason)  
    {  
  
        case LWS_CALLBACK_ESTABLISHED:  
            /* connection established */  
            break;  
  
        case LWS_CALLBACK_CLOSED:  
            /* connection closed */  
            break;  
  
        case LWS_CALLBACK_SERVER_WRITEABLE:  
            /* send data to client */  
            break;  
  
        case LWS_CALLBACK_RECEIVE:  
            psd->len = parse_json (received_data);  
            if (psd->len > 0)  
                libwebsocket_callback_on_writable ();  
            break;  
  
    }  
}
```



cmd:

- ✚ GetGPIO,
- ✚ SetGPIO,
- ✚ GetTempSensors,

Raspberry Control 2.0 - serwer

```
raspberry_control_callback ()  
{  
    switch (reason)  
    {  
  
        case LWS_CALLBACK_ESTABLISHED:  
            /* connection established */  
            break;  
  
        case LWS_CALLBACK_CLOSED:  
            /* connection closed */  
            break;  
  
        case LWS_CALLBACK_SERVER_WRITEABLE:  
            /* send data to client */  
            break;  
  
        case LWS_CALLBACK_RECEIVE:  
            psd->len = parse_json (received_data);  
            if (psd->len > 0)  
                libwebsocket_callback_on_writable ();  
            break;  
  
    }  
}
```

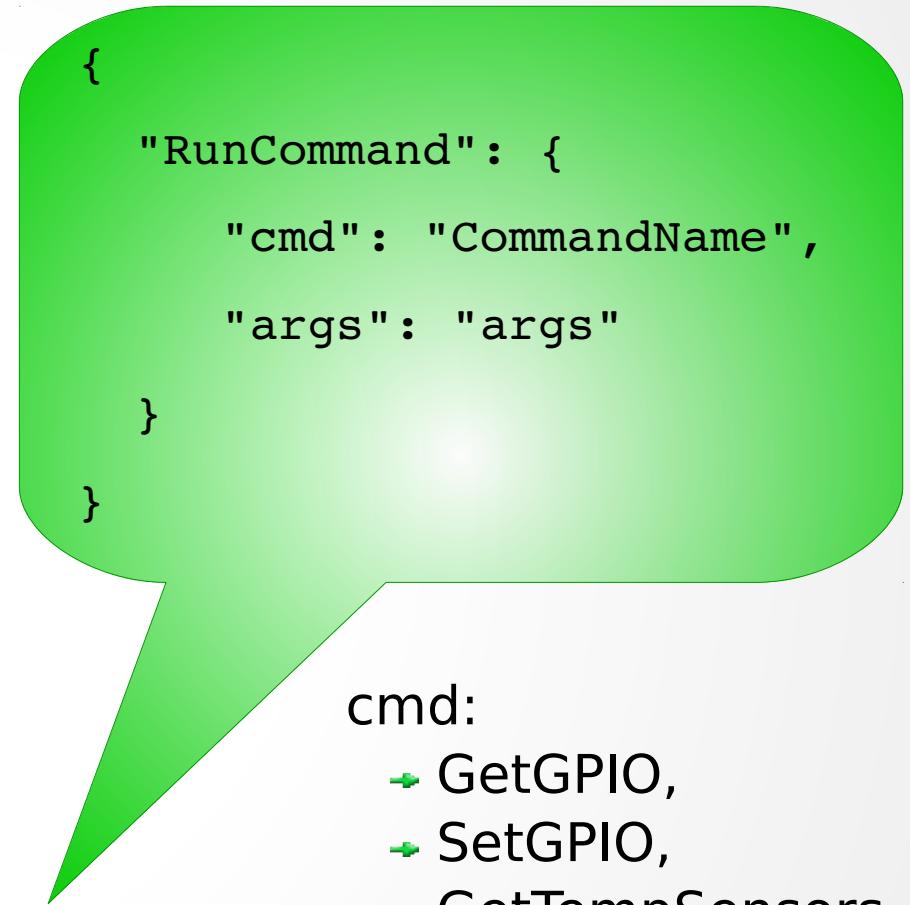


cmd:

- ➔ GetGPIO,
- ➔ SetGPIO,
- ➔ GetTempSensors,
- ➔ GetProcesses,

Raspberry Control 2.0 - serwer

```
raspberry_control_callback ()  
{  
    switch (reason)  
    {  
  
        case LWS_CALLBACK_ESTABLISHED:  
            /* connection established */  
            break;  
  
        case LWS_CALLBACK_CLOSED:  
            /* connection closed */  
            break;  
  
        case LWS_CALLBACK_SERVER_WRITEABLE:  
            /* send data to client */  
            break;  
  
        case LWS_CALLBACK_RECEIVE:  
            psd->len = parse_json (received_data);  
            if (psd->len > 0)  
                libwebsocket_callback_on_writable ();  
            break;  
  
    }  
}
```



cmd:

- ➔ GetGPIO,
- ➔ SetGPIO,
- ➔ GetTempSensors,
- ➔ GetProcesses,
- ➔ KillProcess,

Raspberry Control 2.0 - serwer

```
raspberry_control_callback ()  
{  
    switch (reason)  
    {  
  
        case LWS_CALLBACK_ESTABLISHED:  
            /* connection established */  
            break;  
  
        case LWS_CALLBACK_CLOSED:  
            /* connection closed */  
            break;  
  
        case LWS_CALLBACK_SERVER_WRITEABLE:  
            /* send data to client */  
            break;  
  
        case LWS_CALLBACK_RECEIVE:  
            psd->len = parse_json (received_data);  
            if (psd->len > 0)  
                libwebsocket_callback_on_writable ();  
            break;  
    }  
}
```

```
{  
  
    "RunCommand": {  
  
        "cmd": "CommandName",  
  
        "args": "args"  
    }  
}
```

- cmd:
- ➔ GetGPIO,
 - ➔ SetGPIO,
 - ➔ GetTempSensors,
 - ➔ GetProcesses,
 - ➔ KillProcess,
 - ➔ GetStatistics,

Raspberry Control 2.0 - serwer

```
raspberry_control_callback ()  
{  
    switch (reason)  
    {  
  
        case LWS_CALLBACK_ESTABLISHED:  
            /* connection established */  
            break;  
  
        case LWS_CALLBACK_CLOSED:  
            /* connection closed */  
            break;  
  
        case LWS_CALLBACK_SERVER_WRITEABLE:  
            /* send data to client */  
            break;  
  
        case LWS_CALLBACK_RECEIVE:  
            psd->len = parse_json (received_data);  
            if (psd->len > 0)  
                libwebsocket_callback_on_writable ();  
            break;  
    }  
}
```

```
{  
  
    "RunCommand": {  
  
        "cmd": "CommandName",  
  
        "args": "args"  
    }  
}
```

- cmd:
- + GetGPIO,
 - + SetGPIO,
 - + GetTempSensors,
 - + GetProcesses,
 - + KillProcess,
 - + GetStatistics,
 - + SendIR,

Raspberry Control 2.0 - klient

- aplikacja napisana w języku Java z wykorzystaniem wielu otwartoźródłowych modułów i bibliotek,

Raspberry Control 2.0 - klient

- aplikacja napisana w języku Java z wykorzystaniem wielu otwartoźródłowych modułów i bibliotek,
- proste zarządzanie zależnościami oraz komplikacja z wykorzystaniem systemu budowania *Gradle*:

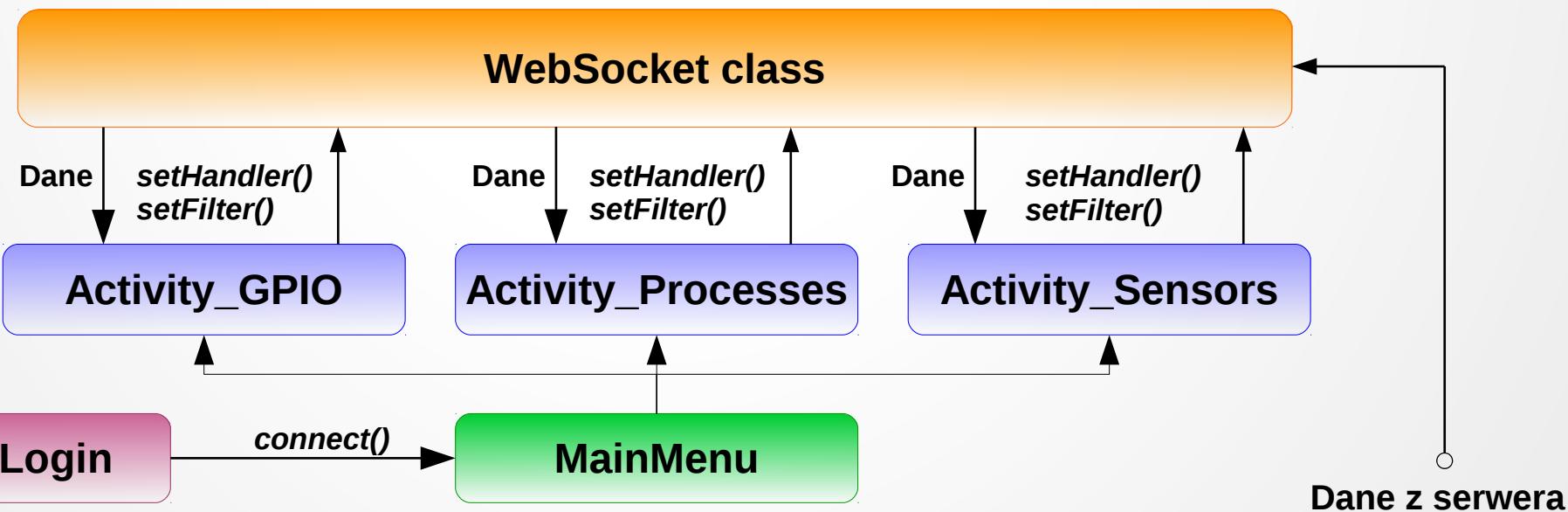
```
./gradlew build
```

Raspberry Control 2.0 - klient

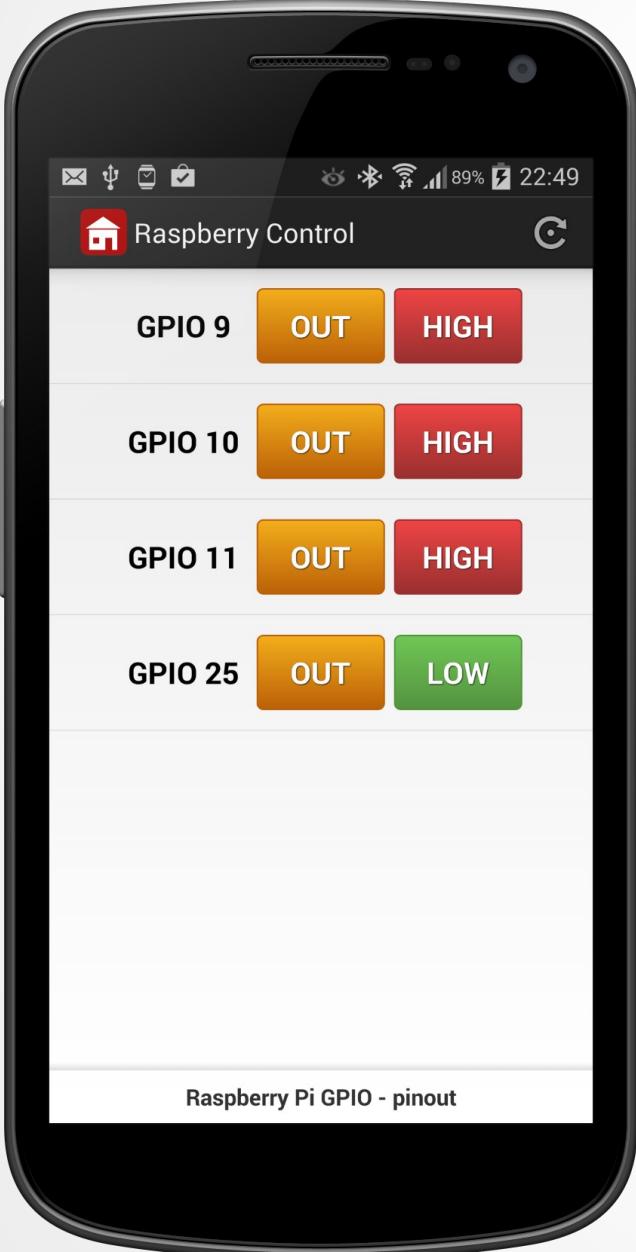
- aplikacja napisana w języku Java z wykorzystaniem wielu otwartoźródłowych modułów i bibliotek,
- proste zarządzanie zależnościami oraz komplikacja z wykorzystaniem systemu budowania *Gradle*:

```
./gradlew build
```

- modularna budowa kodu umożliwiająca szybkie tworzenie nowych funkcjonalności:



Raspberry Control 2.0 - klient



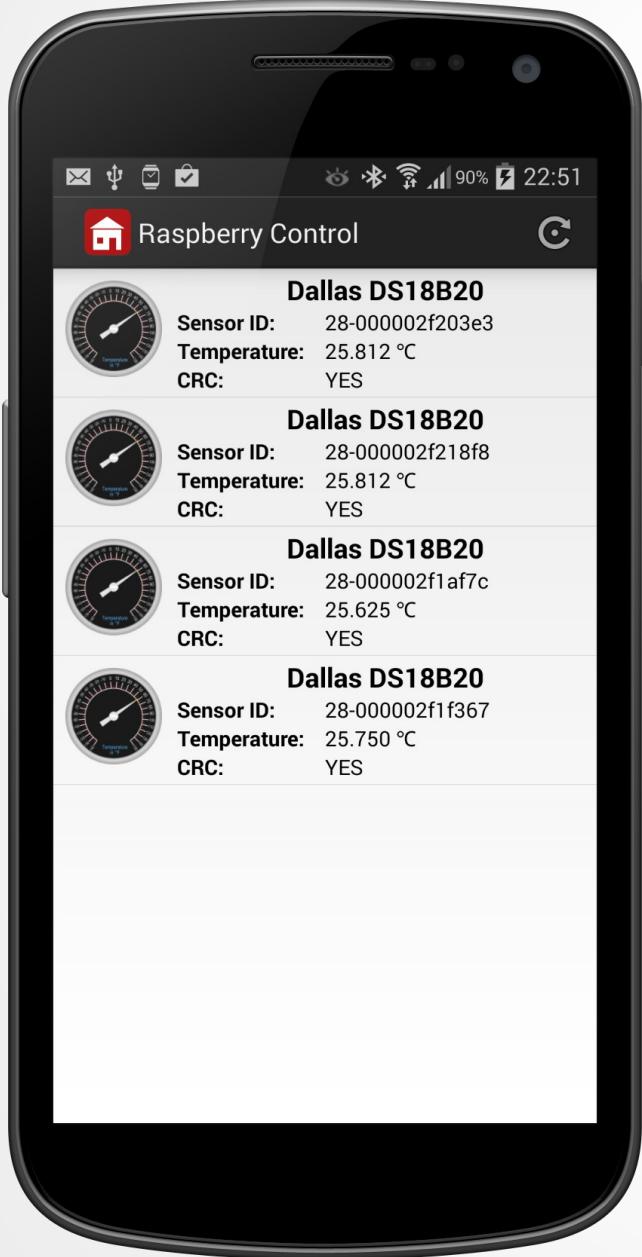
Opcja “GPIO Control”:

- wygodne sterowanie wszystkimi wyprowadzeniami GPIO wyeksportowanymi do przestrzeni użytkownika,
- prosta i szybka zmiana wartości wyjściowej lub kierunku pracy wybranego wyprowadzenia,
- schemat złącza GPIO wyświetlany na podstawie numeru seryjnego płytki deweloperskiej,

Obiekt JSON:

```
{  
    "Revision": "000f",  
    "GPIOState": [  
        {  
            "value": 1,  
            "gpio": 9,  
            "direction": "out"  
        },  
        {  
            "value": 0,  
            "gpio": 25,  
            "direction": "out"  
        }  
    ]  
}
```

Raspberry Control 2.0 - klient



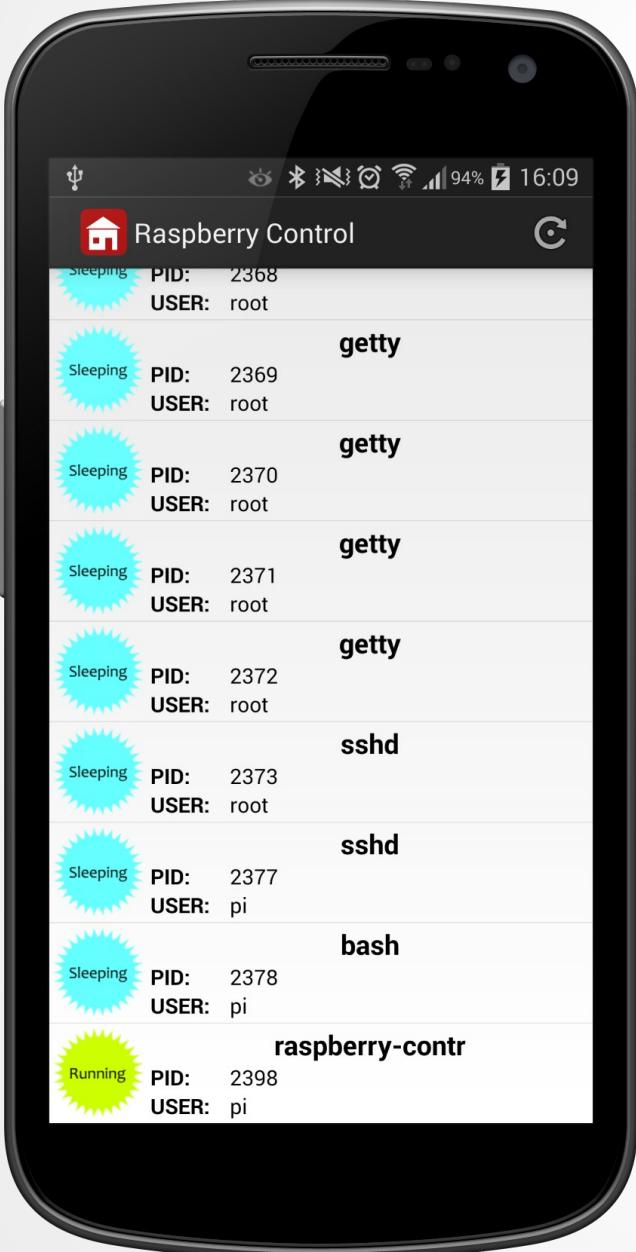
Opcja “Temperature Sensors”:

- wygodny odczyt wartości temperatur z wszystkich czujników serii DS18*20 podłączonych do magistrali 1-Wire,
- informacja o numerze seryjnym i sumie CRC,
- automatyczne skanowanie magistrali 1-Wire po stronie serwera,

Obiekt JSON:

```
{  
    "TempSensors": [  
        {  
            "type": "Dallas DS18B20",  
            "crc": "YES",  
            "id": "28-000002f203e3",  
            "temp": 25.687  
        },  
        {  
            "type": "Dallas DS18B20",  
            "crc": "YES",  
            "id": "28-000002f1f367",  
            "temp": 25.625  
        }  
    ]  
}
```

Raspberry Control 2.0 - klient



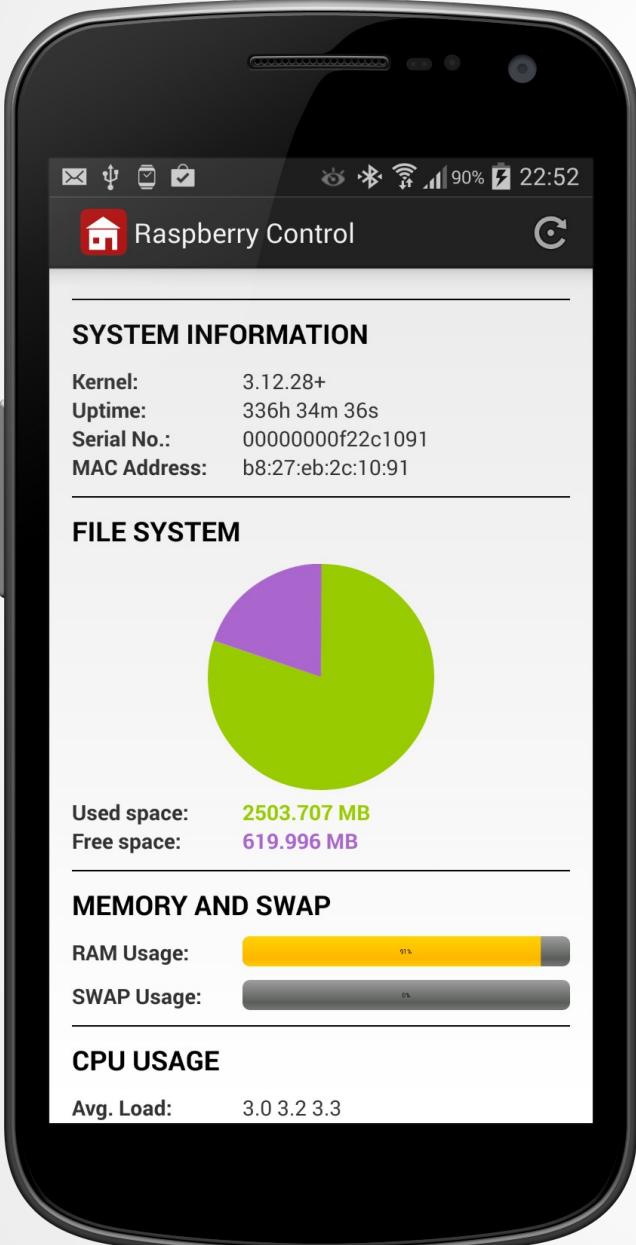
Opcja “Processes Management”:

- wyświetla listę wszystkich procesów uruchomionych po stronie serwera,
- dostarcza informacji o numerze PID, właściwemu oraz aktualnym stanie procesu,
- umożliwia zdalne zamykanie procesu po stronie serwera,

Obiekt JSON:

```
{  
    "Processes": [  
        {  
            "pid": 1,  
            "user": "root",  
            "name": "init",  
            "state": "S (sleeping)"  
        },  
        {  
            "pid": 2398,  
            "user": "pi",  
            "name": "raspberry-contr",  
            "state": "R (running)"  
        }  
    ]  
}
```

Raspberry Control 2.0 - klient



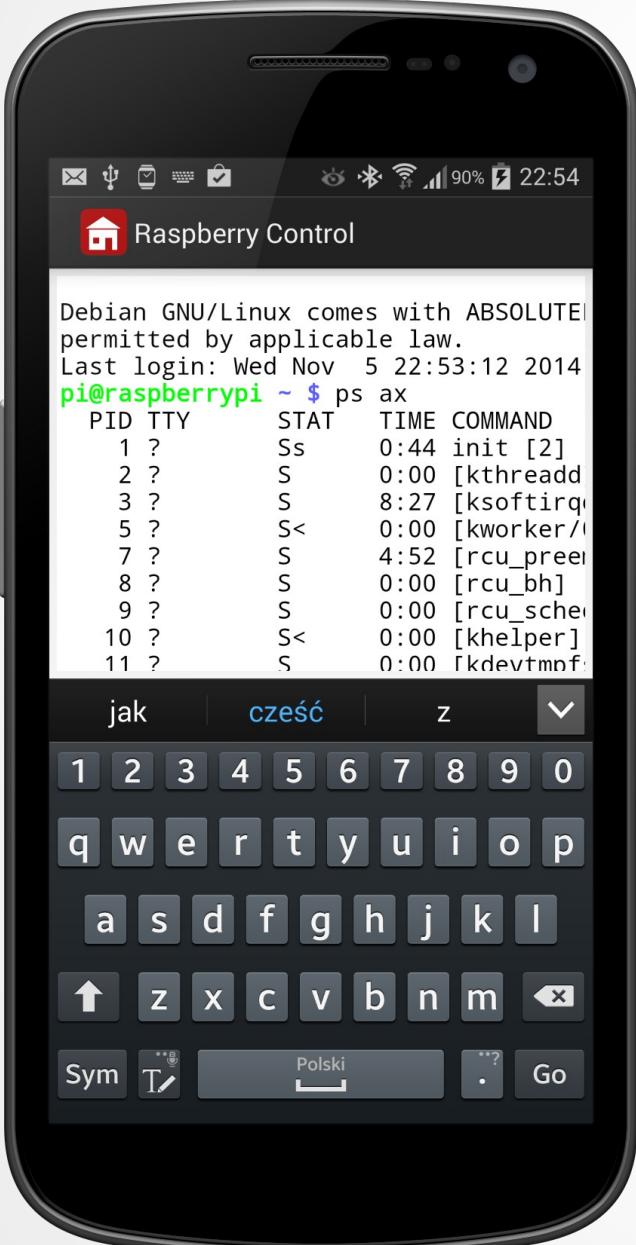
Opcja “Monitor System Resources”:

- monitorowanie wybranych parametrów serwera, między innymi: temperatury i obciążenia procesora, stopnia wykorzystania pamięci RAM i SWAP, itd.
- automatyczne odświeżanie statystyk z wybranym interwałem czasowym,

Obiekt JSON:

```
{  
    "Statistics": {  
        "kernel": "3.12.28+",  
        "uptime": "350h 11m 5s",  
        "serial": "0000000f22c1091",  
        "cpu_temp": 54,  
        "mac_addr": "b8:27:eb:2c:10:91",  
        "free_space": 619.53125,  
        "used_space": 2504.171875,  
        "ram_usage": 93,  
        "swap_usage": 0,  
        "cpu_load": "3.0 3.1 3.2",  
        "cpu_usage": 100  
    }  
}
```

Raspberry Control 2.0 - klient



Opcja “**Terminal Emulator**”:

- prosty i szybki dostęp do linii poleceń serwera poprzez usługę **shellinabox**,
- w dalszych planach rozwoju aplikacji ewentualne zastąpienie **shellinabox** natywną implementacją klienta po stronie aplikacji,

Uruchomienie daemona **shellinabox** po stronie serwera:

sudo apt-get install shellinabox

```
/usr/bin/shellinaboxd -q –background=/var/run/shellinaboxd.pid \
-c /var/lib/shellinabox -p 8081 -t
```

Raspberry Control 2.0 - klient



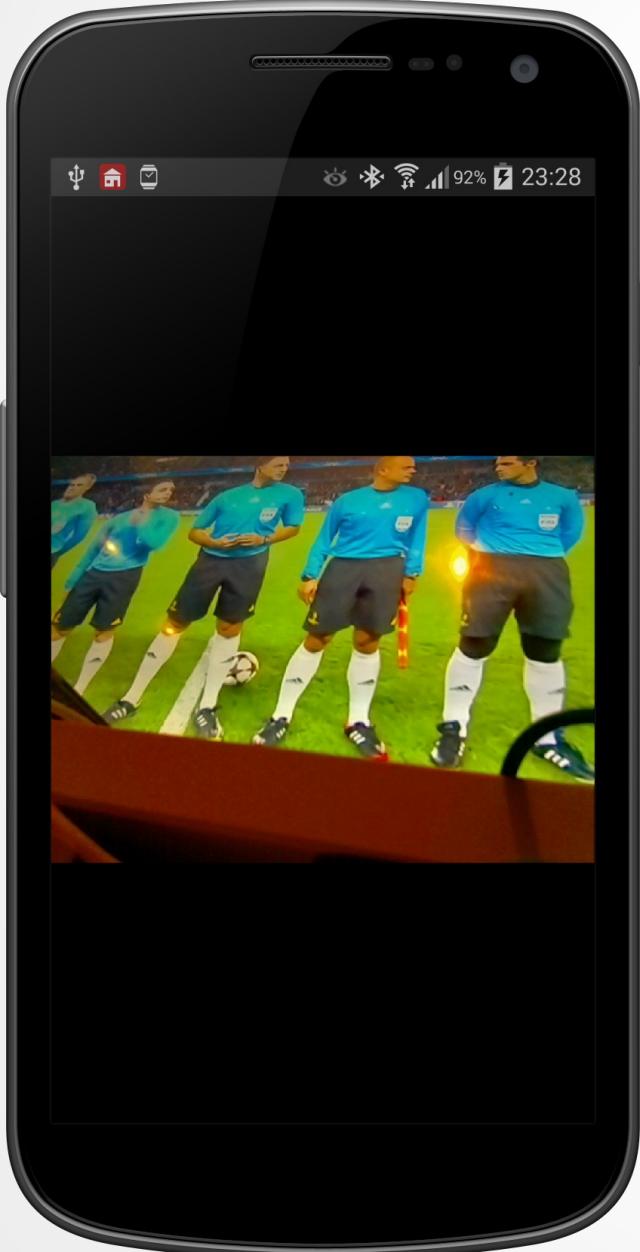
Opcja “*IR Remote Control*”:

- sterowanie zdefiniowanymi przez użytkownika urządzeniami zdalnymi za pomocą sygnałów podczerwieni oraz modułu *LIRC*,
- konfiguracja funkcji przycisków po stronie aplikacji mobilnej,

Konfiguracja modułu *LIRC* po stronie serwera:

```
sudo apt-get install lirc  
  
vi /etc/modules  
  
+++ lirc_dev  
+++ lirc_rpi gpio_out_pin=22  
  
vi /etc/lirc/hardware.conf  
  
/.../  
  
cp ~/lircd.conf /etc/lirc/lircd.conf  
  
(http://lirc.sourceforge.net/remotes/)
```

Raspberry Control 2.0 - klient



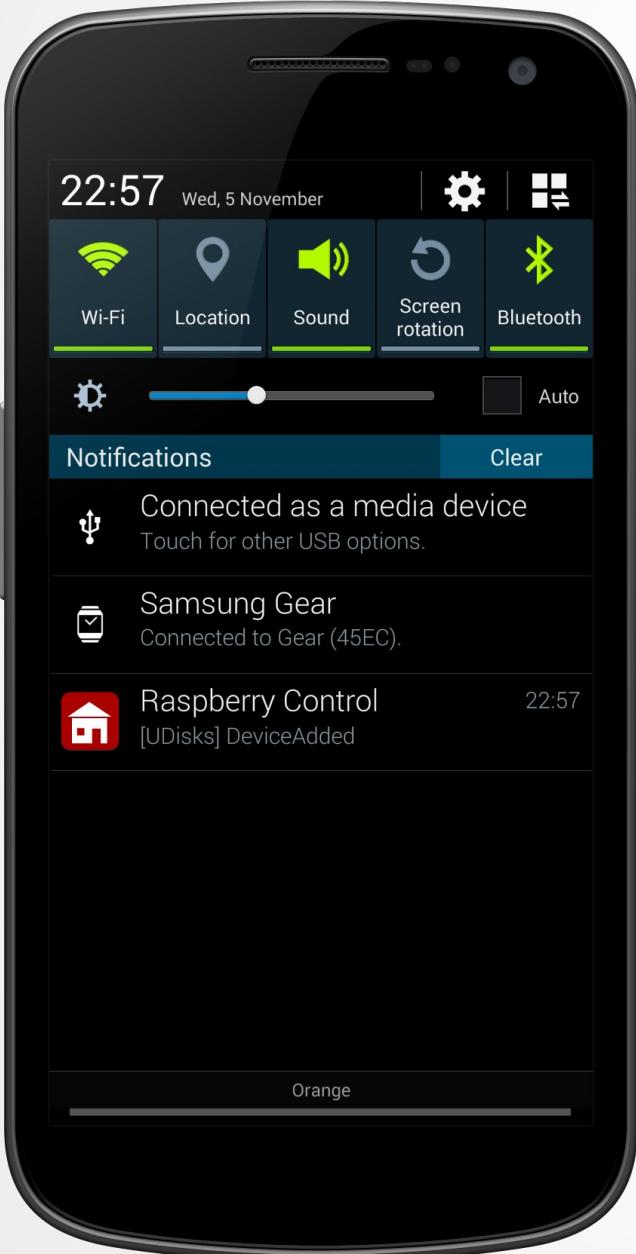
Opcja “**Webcam Surveillance**”:

- natywny klient MJPG umożliwiający uzyskanie podglądu w czasie rzeczywistym z kamer podłączonych do serwera,

Konfiguracja modułu **mjpg-stream** po stronie serwera:

```
apt-get install libv4l-dev libjpeg8-dev imagemagick  
cd $HOME  
git clone https://github.com/lukasz-skalski/mjpg-  
streamer.git  
  
cd mjpg-streamer/mjpg-streamer-experimental  
make  
  
export LD_LIBRARY_PATH=.  
. ./mjpg_streamer -o "output_http.so -p 8082 -w ./www" -i  
"input_raspicam.so -x 640 -y 480 -fps 20 -ex night"
```

Raspberry Control 2.0 - klient



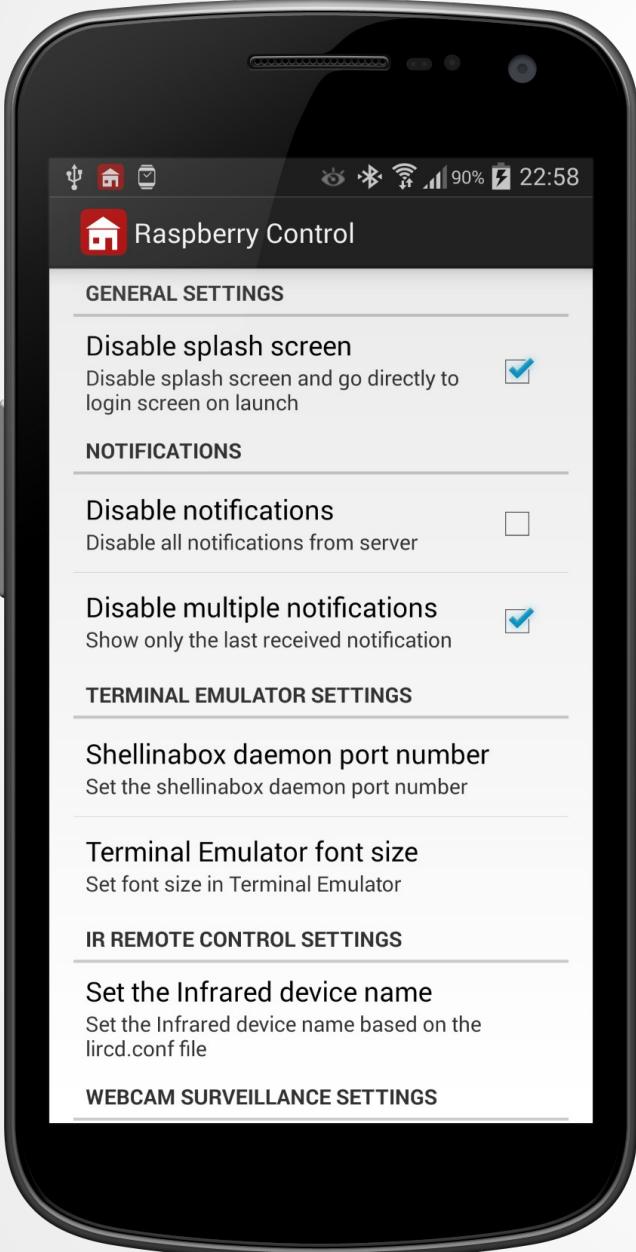
Powiadomienia:

- zapewniają szybkie, asynchroniczne informacje o zmianie wybranych parametrów serwera,
- prosty mechanizm dodawania nowych subskrypcji na sygnały D-Bus po stronie serwera - `dbus_set_notification()`,
- notyfikacje w sposób rozgłoszeniowy przesyłane są do wszystkich aktualnie podłączonych do serwera klientów,
- możliwość konfiguracji powiadomień po stronie aplikacji klienta,

Obiekt JSON:

```
{  
    "Notification" : "[Udisks] DeviceAdded"  
}
```

Raspberry Control 2.0 - klient



Wybrane opcje konfiguracji aplikacji:

- ✚ możliwość zablokowania ekranu startowego – bezpośrednie przejście do ekranu logowania po uruchomieniu aplikacji,
- ✚ możliwość całkowitego wyłączenia oraz prostej konfiguracji otrzymywanych notyfikacji,
- ✚ opcje konfiguracji aktywności “Terminal Emulator” (numer portu serwisu *shellinabox* oraz rozmiar czcionki),
- ✚ ustawienie nazwy urządzenia zdalnego na potrzeby modułu *LIRC*,
- ✚ ustawienia natywnego klient *MJPG* (adres stream'u oraz informacje dotyczące liczby uzyskanych *FPS*-ów),

Raspberry Control 2.0 - klient



Wybrane komponenty/biblioteki otwartoźródłowe:

- *UpdateChecker*,
- *NotBoringActionBar*,
- *Android Sliding Up Panel*,
- *HoloGraphLibrary*,
- *LensesDialog*,
- *SuperToasts*,
- *SecureWebSockets*,
- ***shellinabox***,
- ***mjpg-stream***,
- ***LIRC***,

Raspberry Control 2.0 - wearable

- aplikacja napisana w języku JavaScript oraz HTML+CSS, pracująca pod kontrolą systemu operacyjnego *Tizen*,



Raspberry Control 2.0 - wearable

- aplikacja napisana w języku JavaScript oraz HTML+CSS, pracująca pod kontrolą systemu operacyjnego *Tizen*,
- całość komunikacji z serwerem (wymiana danych w formacie JSON) odbywa się za pośrednictwem urządzenia mobilnego,



Raspberry Control 2.0 - wearable

- aplikacja napisana w języku JavaScript oraz HTML+CSS, pracująca pod kontrolą systemu operacyjnego *Tizen*,
- całość komunikacji z serwerem (wymiana danych w formacie JSON) odbywa się za pośrednictwem urządzenia mobilnego,
- do realizacji połączenia pomiędzy urządzeniem *Samsung Gear 2* a telefonem, wykorzystano protokół *SAP* (*Samsung Accessory Protocol*),



Raspberry Control 2.0 - wearable

- aplikacja napisana w języku JavaScript oraz HTML+CSS, pracująca pod kontrolą systemu operacyjnego *Tizen*,
- całość komunikacji z serwerem (wymiana danych w formacie JSON) odbywa się za pośrednictwem urządzenia mobilnego,
- do realizacji połączenia pomiędzy urządzeniem *Samsung Gear 2* a telefonem, wykorzystano protokół *SAP* (*Samsung Accessory Protocol*),
- instalacja aplikacji na urządzeniu odbywa się równolegle do instalacji aplikacji mobilnej,



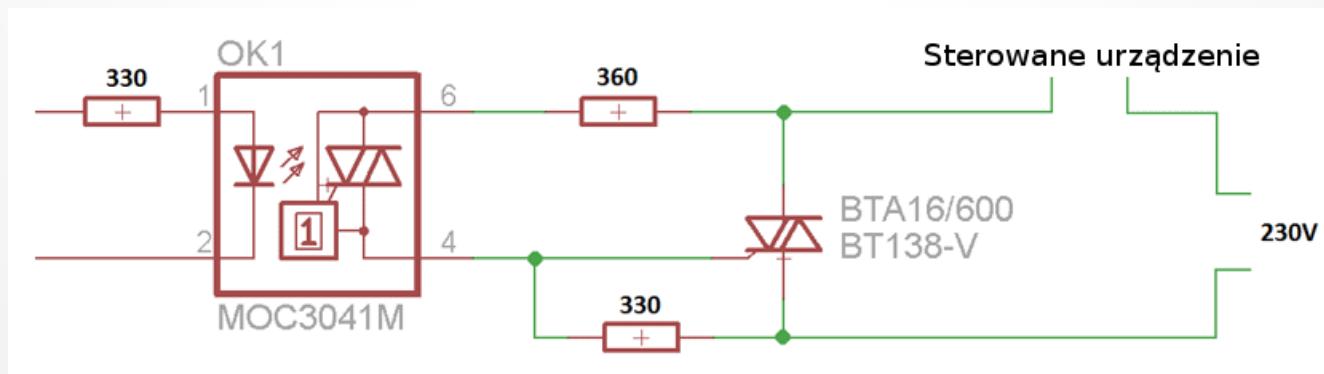
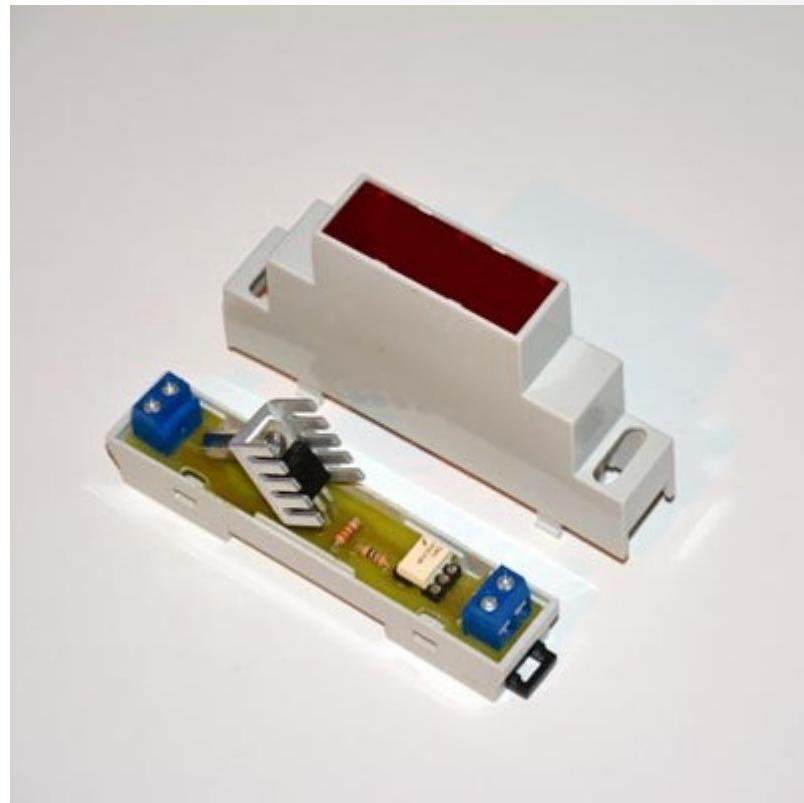
Raspberry Control 2.0 - wearable

- aplikacja napisana w języku JavaScript oraz HTML+CSS, pracująca pod kontrolą systemu operacyjnego *Tizen*,
- całość komunikacji z serwerem (wymiana danych w formacie JSON) odbywa się za pośrednictwem urządzenia mobilnego,
- do realizacji połączenia pomiędzy urządzeniem *Samsung Gear 2* a telefonem, wykorzystano protokół *SAP* (*Samsung Accessory Protocol*),
- instalacja aplikacji na urządzeniu odbywa się równolegle do instalacji aplikacji mobilnej,
- logika działania aplikacji “zegarkowej” jest analogiczna do zadań wykonywanych przez aplikację mobilną - parsowanie tych samych obiektów JSON,



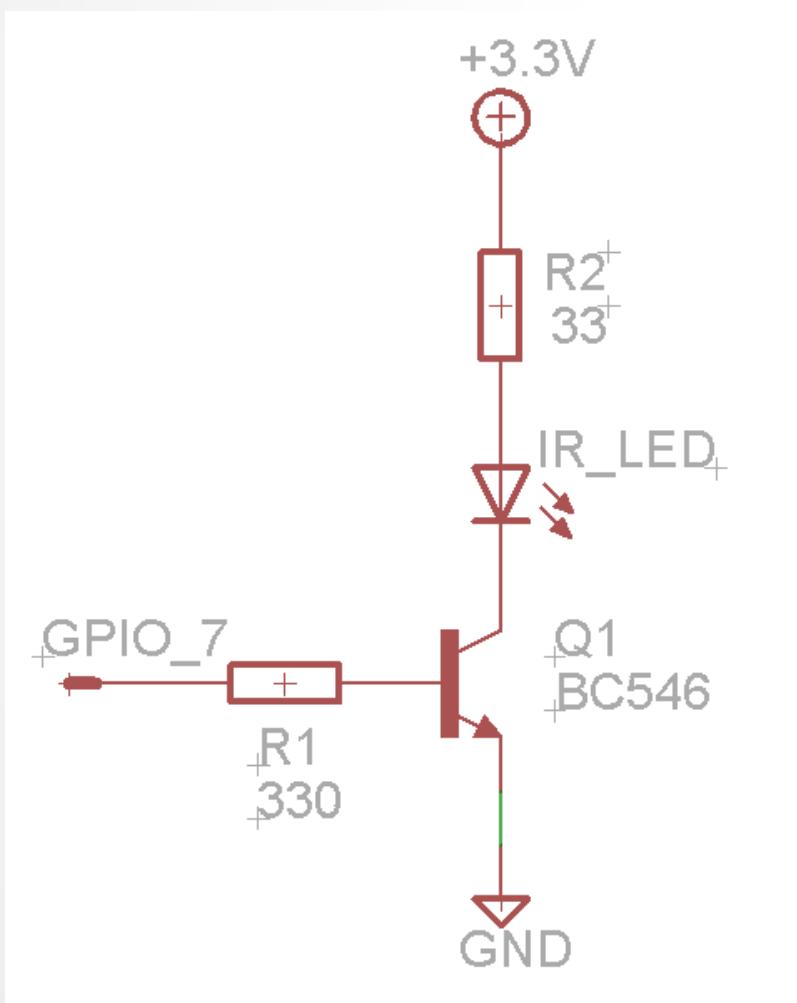
Raspberry Control 2.0 - proste moduły wykonawcze i pomiarowe

Moduł I - triakowy moduł sterowania włącz/wyłącz (koszt: ~5zł)

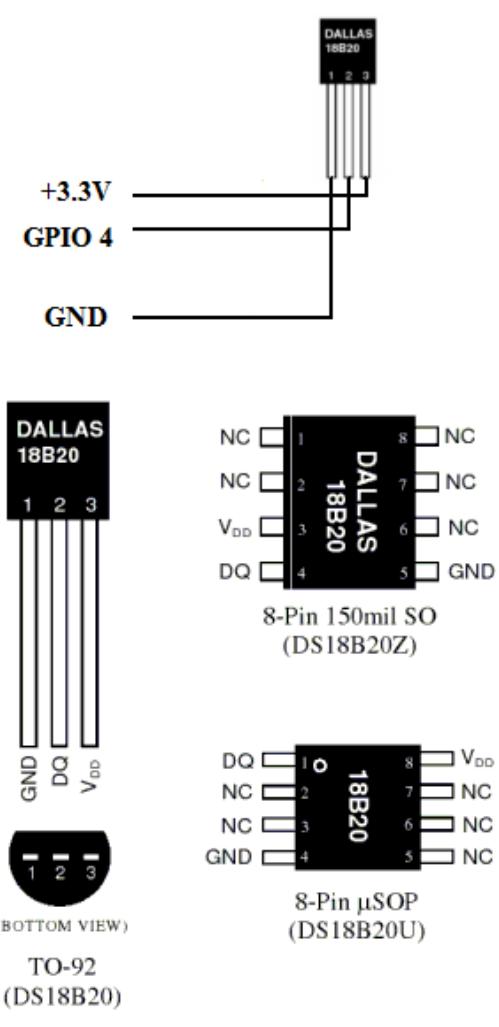


Raspberry Control 2.0 - proste moduły wykonawcze i pomiarowe

Moduł II - moduł sterowania za pomocą podczerwieni (koszt ~3zł)



Moduł III - moduły czujników temperatury (koszt ~4zł)



Raspberry Control 2.0 - proste moduły wykonawcze i pomiarowe



Raspberry Control 2.0

Dziękuję za uwagę :)



www.lukasz-skalski.com