Gepetto Documentation

Release 0.1.0

Auke Willem Oosterhoff

October 23, 2014

Contents

Welcome to Geppetto's documentation. Geppetto provides a simple and clean HTTP REST API to interact the GPIO pins of the Raspberry Pi. All models of the Raspberry Pi are supported.

Geppetto is written in Python and has been tested with 2.7, 3.3 and 3.4. Geppetto's source code is distributed under a MIT license.

Geppetto is named after Mister Geppetto, a character created by Carlo Collodi in his novel The Adventures of Pinocchio.

CHAPTER 1

Contents

1.1 Installation

Note: Gepppetto works with Python 2.7 and Python >= 3.3.

Download the source from GitHub and install the requirements.

```
$ git clone https://github.com/OrangeTux/Geppetto.git
```

Install Python dependencies:

```
$ cd Geppetto
$ virtualenv .env
$ source .env/bin/activate
$ pip install -r requirements.txt
```

Setup the database and create a user with an API key:

```
$ pynt setup_db create_user
[ build.py - Starting task "setup_db" ]
[ build.py - Completed task "setup_db" ]
[ build.py - Starting task "create_user" ]
User <User 1, api_key: b'fcba99ca-3360-4683-a54d-1ce8ad1f20f3'> has been created.
[ build.py - Completed task "create_user" ]
```

You can use generated API key for authentication

Geppetto uses Quick2Wire's GPIO Admin to use GPIO pins without root permissions. Download this utility and follow the installation instructions

1.2 Quickstart

Note: Do not start Geppetto like this in a production environment! Running Geppetto as root is dangerous!

Inside your virtualenv set the environmnent *GEPPETTO_ENV* and start the server:

```
$ export GEPETTO_ENV=dev
$ .env/bin/python server.py
* Running on http://127.0.0.1:5000/
* Restarting with reloader
```

Now you interact with the API on 127.0.0.1:5000. See *Usage* how to use the API and how to toggle GPIO pins.

1.3 Deployment

This section describes how to deploy Geppetto with Nginx and uWSGI. You can use other web servers and other WSGI servers if you desire.

Note: Throughout this page the location */var/www/geppetto* is used. If you put Geppetto somewhere else, set correct path.

1.3.1 Nginx

Install Nginx.

```
$ sudo apt-get install nginx
```

Nginx runs default as user *www-data* and so this user will run Geppetto. Add the user *www-data* to the group *gpio*, so it can interact with the GPIO pins without root permissions.

```
$ sudo adduser www-data gpio
```

Put the configuration below inside */etc/nginx/sites-available/geppetto*.

```
server {
  server_name localhost;

  location / {
    try_files $uri @geppetto;
  }

  location @geppetto {
    include uwsgi_params;
    uwsgi_modifier1 30;
    uwsgi_pass unix:/var/www/geppetto/data/geppetto.sock;
  }
}
```

and create a symbolic link to */etc/nginx/sites-enabled/*:

```
$ sudo ln -s /etc/nginx/site-available/geppetto /etc/nginx/sites-enabled/geppetto
```

Restart Nginx.

```
$ sudo service restart nginx
```

1.3.2 uWSGI

Install uWSGI inside your virtualenv, but install it also system wide. Althought the latter is not necessarily required, the system wide installation copies some uWSGI configuration to the correct places. In order you install uWSGI inside your virtualenv Python's development headers are required.

```
$ sudo apt-get install uwsgi python-dev
$ pip install uwsgi
```

When running Gepppetto we use uWSGI binary inside the virtualenv. The reason that we use this binary is that the version from PyPi is newer than the one from the Debian repositories.

Put the following configuration inside */etc/uwsgi/apps-available/geppetto.ini*.

```
[uwsgi]
chdir = /var/www/geppetto
uid = www-data
gid = www-data
chmod-socket = 666
socket = /var/www/geppetto/data/geppetto.sock
module = server
callable = app
virtualenv = /var/www/geppetto/.env
plugins = python
env = GEPPETTO_ENV=prod
```

Create a symbolic link to the *app-enabled* folder:

```
ln -s /etc/uwsgi/apps-available/geppetto.ini /etc/uwsgi/apps-enabled/geppetto.ini
```

Now you can start uWSGI. Make sure you run the version of uWSGI installed in your virtualenv!

```
$ cd /var/www/geppetto
$ source .env/bin/activate
$ uwsgi --emperor /etc/uwsgi/sites-enabled --daemonize /var/www/geppetto/logs/uwsgi.log
```

Congratulations, Geppetto is running on port 80 of your Raspberry Pi.

1.4 Usage

1.4.1 Request

The following example shows how to enable GPIO 3, or SOC GPIO 22. When you run Geppetto in development mode the Authorization header redundant. Authentication has been disabled in development mode.

```
>>> import requests, json, base64
>>> # Create Authorization header.
>>> auth = {'Authorization': base64.b64encode('fcba99ca-3360-4683-a54d-1ce8ad1f20f3').encode('utf-8')}
>>> # Create body of POST request.
>>> data = json.dumps({'value': 1})
>>> # Do POST request.
>>> requests.post('http://localhost:5000/gpio/3/setpoint', headers=auth, data=data)
<Response [200]>
```

1.4.2 Pin numbering

Geppetto uses the WiringPi's numbering scheme for selecting GPIO pins.

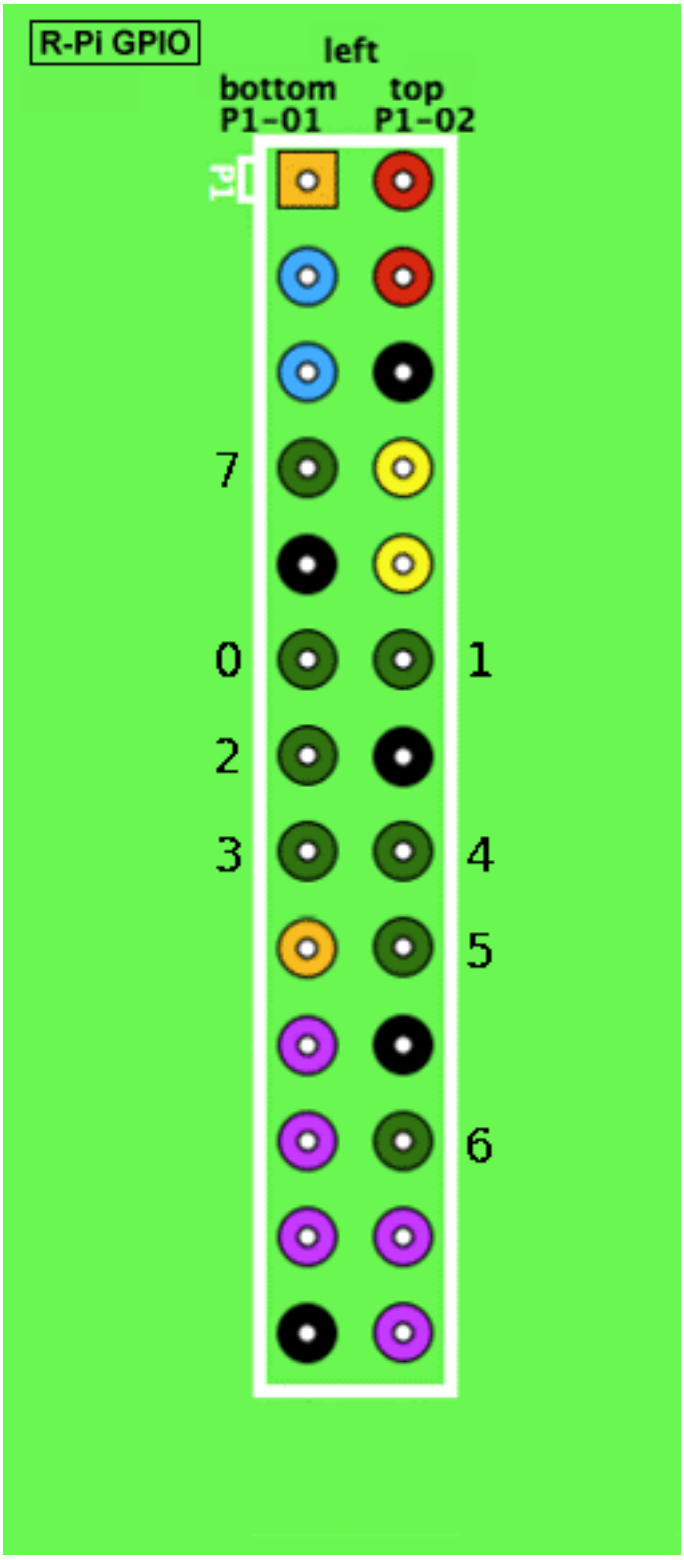

Image is modification of image from Wikipedia and is available under CC BY-SA 3.0.

1.5 Developing

1.5.1 Tests

If you want to run the unittests you've to install pytest and some other dependencies:

```
pip install -r test_dependecies.txt
```

Run tests with:

```
py.test tests/
```

1.5.2 Docs

If you want to build the documentation you've to get the Diaoul Sphinx theme . This Git repository has been added as submodule. Fetch the data:

```
$ git submodule init
$ git submodule update
```

The documenation can be build either with:

```
$ sphinx-build -aE -b html docs docs/_build
```

or:

```
$ make -C docs html
```

The documentation can be found at *docs/_build/html*.

1.6 API

1.6.1 GPIO

GET /test_authentication
Endpoint to test authentication.

POST /gpio/ (**int:** *pin_nr*) **/setpoint**
Enable or disable a GPIO pin.

Example Request:

```
POST /gpio/14/setpoint HTTP/1.1
Accept: application/json
Content-Type: application/json

{
    "value": 1
}
```

Example Response:

```
HTTP/1.1 200 SUCCESS
Content-Type: text/json
```

```
{
    "value": 1
}
```

Request Headers

- **Accept** – application/json
- **Content-Type** – application/json

Response Headers

- **Content-Type** – application/json

Status Codes

- **200** – Success
- **400** – Request body is invalid.
- **403** – Unauthorized.
- **404** – GPIO pin doesn't exists.
- **422** – Request body is not valid JSON.

/gpio

`POST /gpio/(int:pin_nr)/setpoint`, 7

/test_authentication

`GET /test_authentication`, 7