09 RPi gebruiken

Device programming 1

howest
/ we develop people

Inhoudsopgave

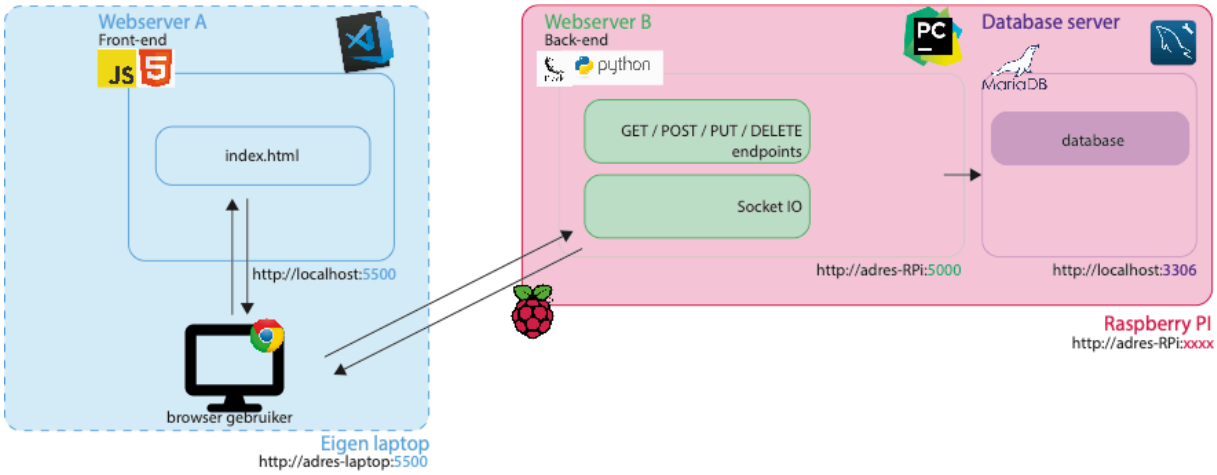
Doelstellingen

1 Verwijzen via het juist IPadres

In de vorige oefeningen konden we steeds het localhost (IP-)adres gebruiken om te verwijzen naar onze servers (front-, back- en databaseserver). Dit kwam omdat we aan het ontwikkelen waren op onze eigen laptop en alle servers dus op ditzelfde toestel draaiden.

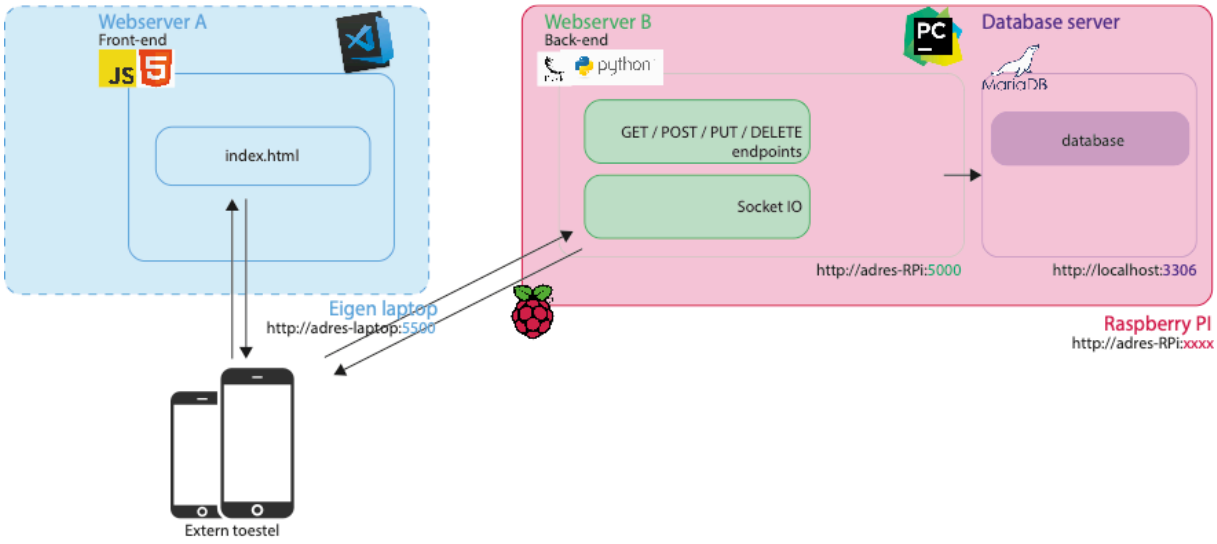
Nu de module Device Programming 1 op zijn einde loopt, zijn we klaar om het laatste puzzelstukje op zijn plaats te laten vallen.
We “verhuizen” de Flask backend server en Database naar onze PI.
Dit heeft enkele gevolgen voor de verwijzingen naar deze servers, meer bepaald het gebruik van de juiste IP-adres.

1.1 Development



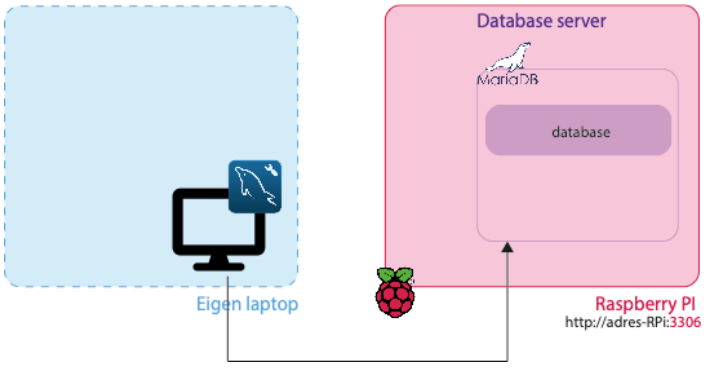
- Surf je van je eigen laptop naar de **frontend**? Dan kan je nog steeds **localhost** gebruiken omdat je browser en frontend website op hetzelfde toesten draaien.
- In de javascript file van je **frontend** moet je vanaf nu het **publiek IP adres** van je Raspberry PI gebruiken om te verwijzen naar de **backend**.
- Vanuit de **backend** code maak je nog steeds verbinding met de databank via **localhost** want de Flask en Databaseserver draaien op hetzelfde toestel.

1.2 Verbinding met de frontend via een extern toestel



- Surf je vanaf een ander toestel (bv laptop collega of gsm) naar de **frontend** die op je laptop draait? Dan gebruik je het **publiek IP-adres** van je laptop.
- Net zoals in het vorig voorbeeld zullen we in de javascript file van de **frontend** verwijzen naar het **publiek IP-adres** van de RPi.
- Vanuit de **backend** code maak je nog steeds verbinding met de databank via **localhost** want de Flask en Databaseserver draaien op hetzelfde toestel.

1.3 Verbinden met de database via Workbench



Verbinden we via Workbench (op onze eigen laptop) met de database op de RPi? Dan moet je het **publiek IP-adres** gebruiken van de RPi.

2 Project deployment

Nu we dit weten is het tijd om onze **backend** én **database** op te RPi te plaatsen.

Ten eerste exporteren we de database vanuit lokale MySQL database naar de mariaDB database op onze Raspbery PI.

Raspbian koos ervoor om niet langer meer MySQL te voorzien op hun image, maar kozen voor de (nieuwe) mariaDB databaseserver.
MariaDB is een Fork van MySQL.

Over het algemeen is mariaDB compatibel met databanken van MySQL.
Twee zaken waarmee je rekening moet houden.

- Bij het aanmaken van een dabase in MySQL kies je best een collation en characterset die ook onderteund wordt door mariaDB. Op deze manier zijn deze exports nadien eenvoudig uitwisselbaar.
  - Collation: utf8_general_ci
  - Characterset: utf8
- Aanmaken van een nieuwe user in mariaDB lukt niet vanuit workbench, dit moet je doen vanuit het commandprompt (eventueel via ssh).
  Dit is de reden dat er reeds een mct-user was aangemaakt op de image.

Ten tweede stellen we PyCharm zodanig in dat hij niet langer gebruik maakt van de interpreter van je eigen pc, maar de externe interpreter gebruikt van je RPi. We laten PyCharm ook de bestanden automatisch kopieren naar onze RPi én plaatsen de Flask webserver toegangkelijk voor externe aanvragen.

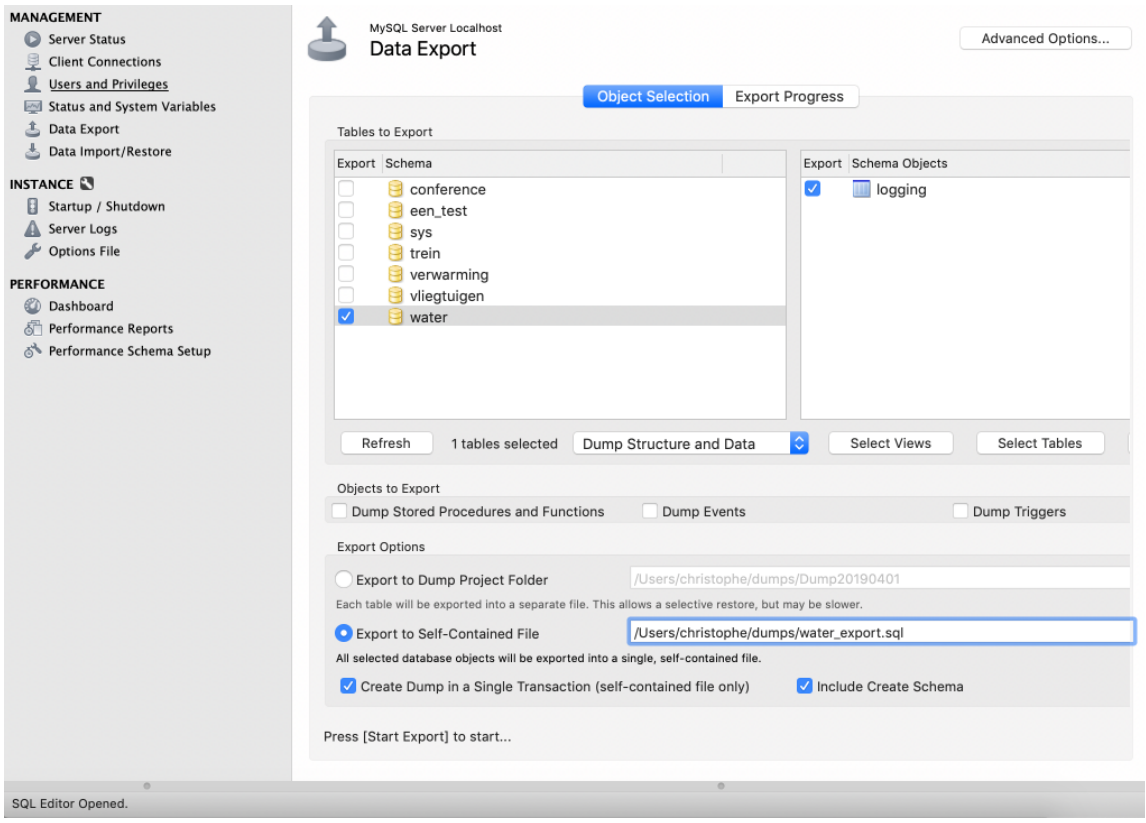
2.1 Database op RPi plaatsen

We verplaatsen we de lokale database naar de RPi. Hiervoor **exporteren** we de lokale database op onze laptop die we vervolgens **importeren** op de RPi.
Vergeet de user mct geen (lees)rechten te geven op de database op de RPi.

2.1.1 Export de lokale MySQL database

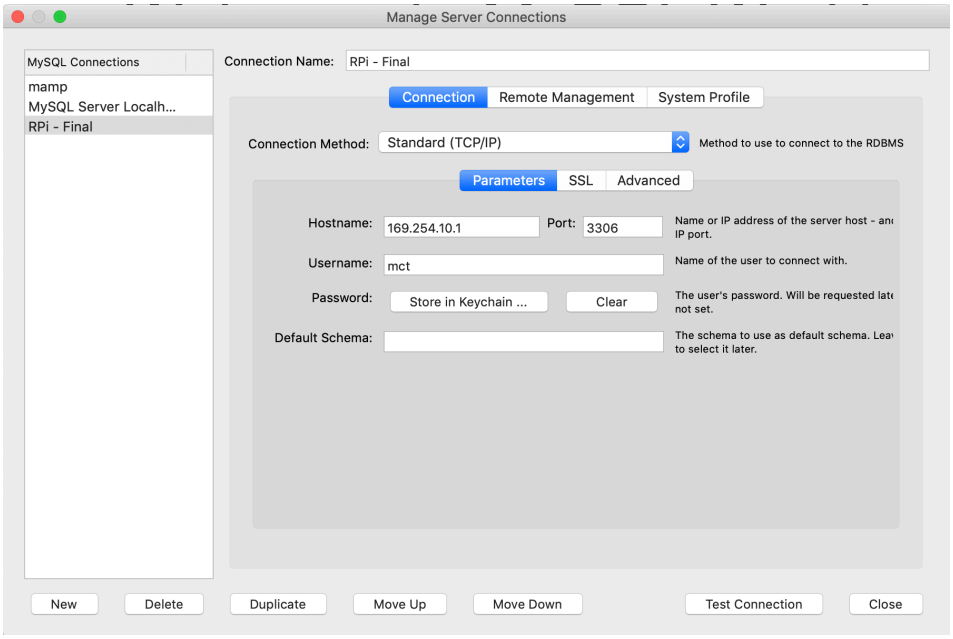
- Administration > Data export
- Kies de gewenste dabase (schema) en alle tabellen (schema objects).
- Export to Self-contained File
    - Create Dump in Single Transaction
    - Kies eventueel voor Include Create Schema

      Op deze manier moet je straks geen lege databank aanmaken.
- Start Export

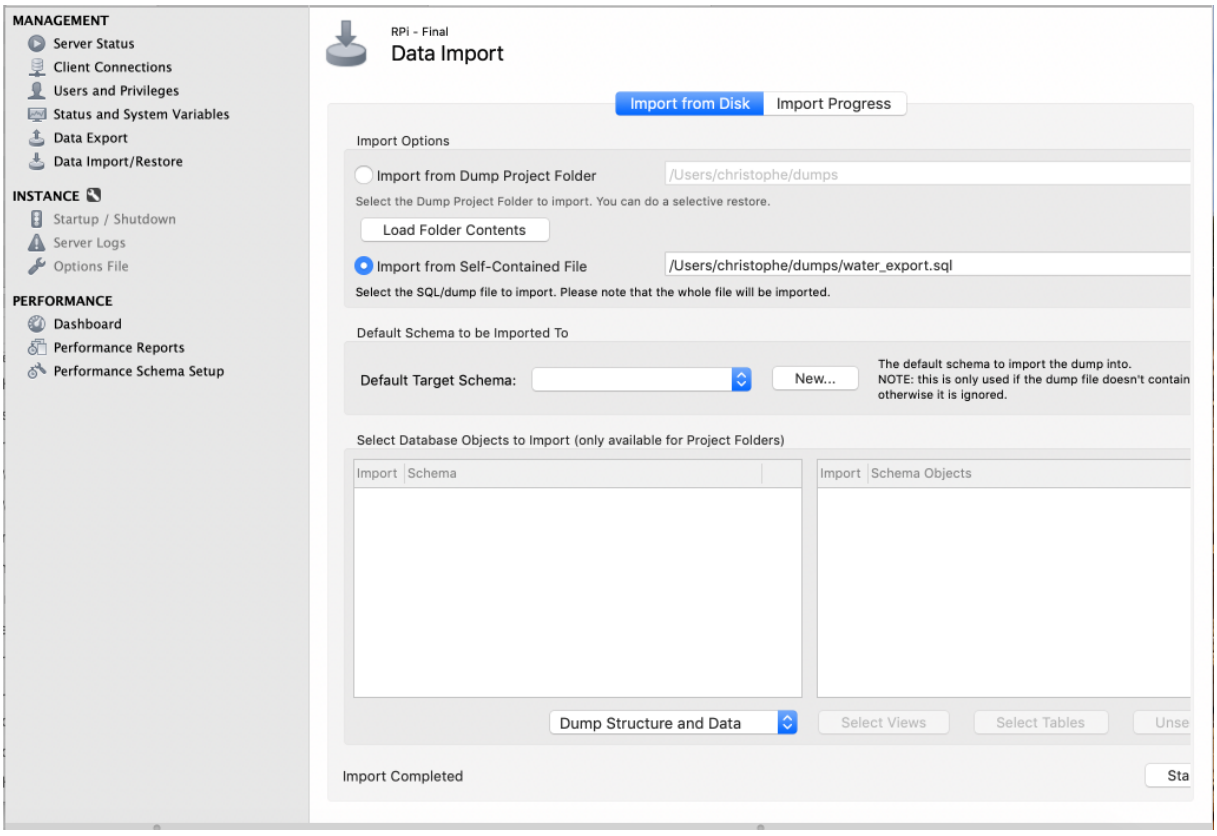


2.1.2 Import de database in de mariaDB database op de RPi

- Maak vanuit workbench connectie met de RPi, Database > Connect to database
    - Hostname: extern ipadres van de RPi
    - Port: 3306 (de poort waarop mariaDB draait op de RPi)
    - Username: mct
    - Paswoord: mct

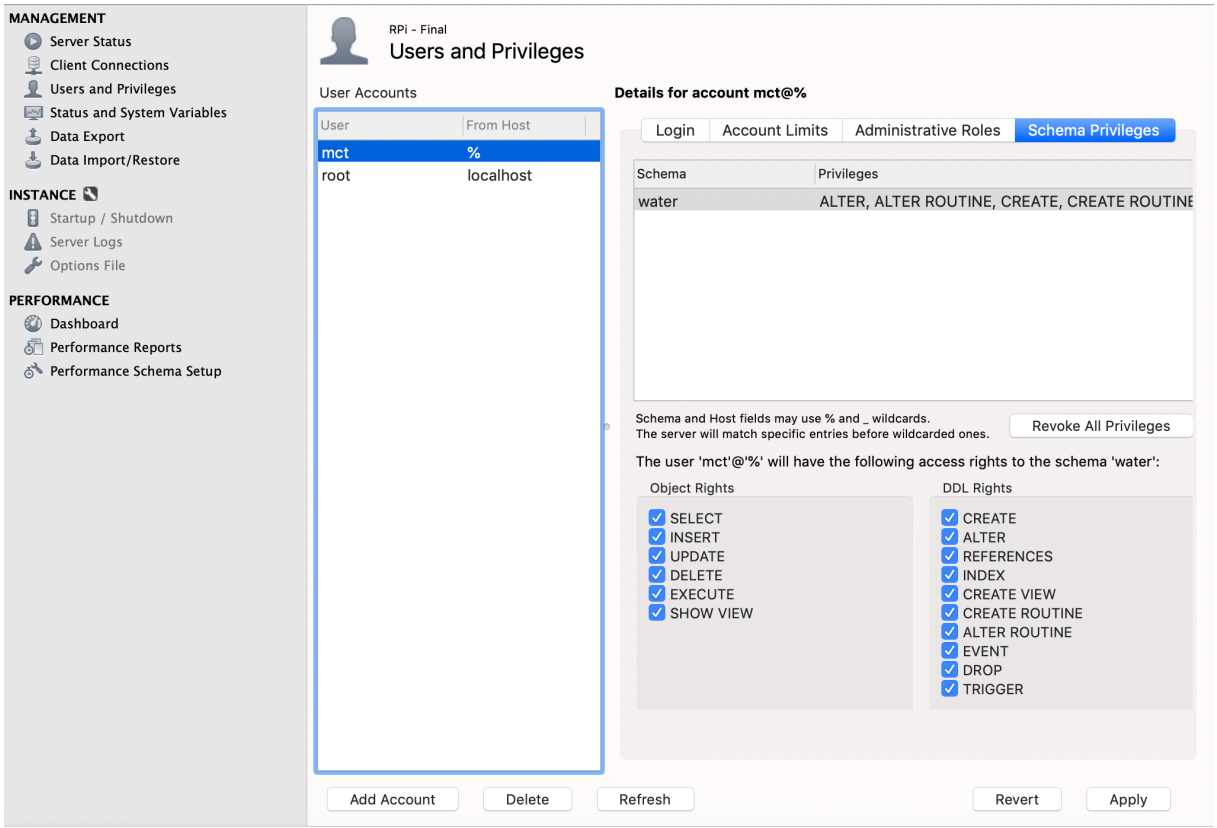


- Administration > Data Import/Restore
  - Import from Self-Contained File
  - Start Import



2.1.3 Rechten geven aan database

- Administration > Users and Privileges
- Kies de <<mct>> user
- Schema Privileges
- Add entry…
- Selected Schema: <<geïmporteerde database>>
- Kies de gewenste rechten die je aan de user wil geven
    - (in development kan “SELECT ALL” handig zijn)
- Apply



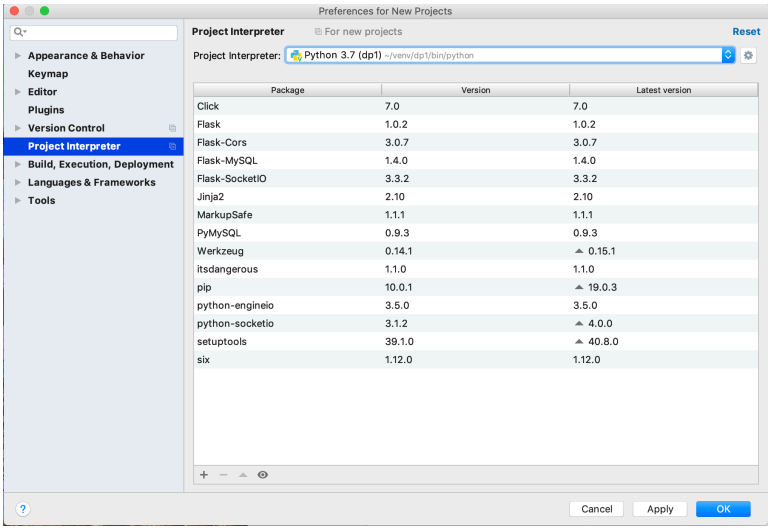
2.2 Flask project via PyCharm verplaatsen

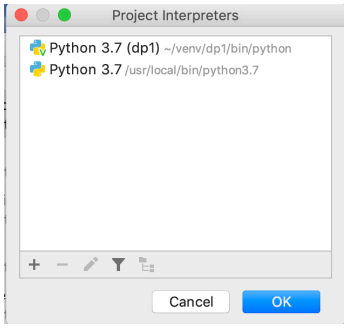
2.2.1 Eenmalig de interpreter instellen in PyCharm

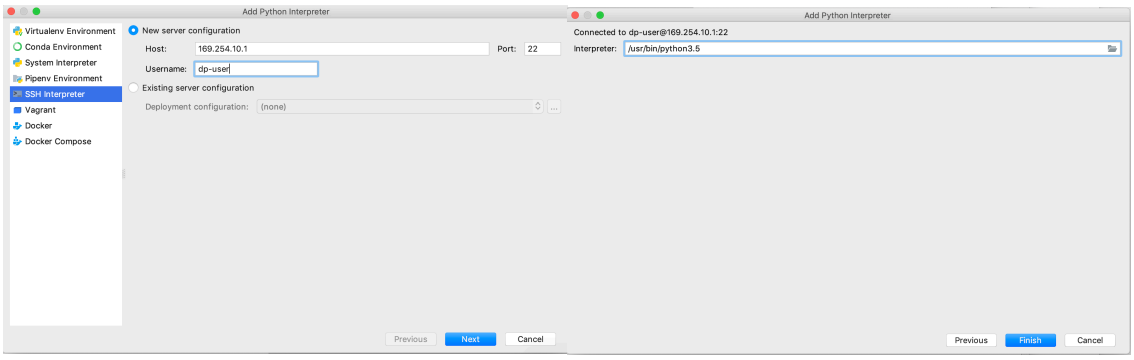
Het kan dat je dit reeds deed in de module Datacommunicatie

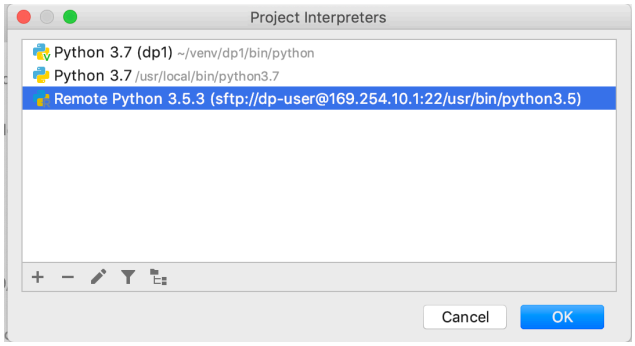
- Sluit al je projecten af in PyCharm

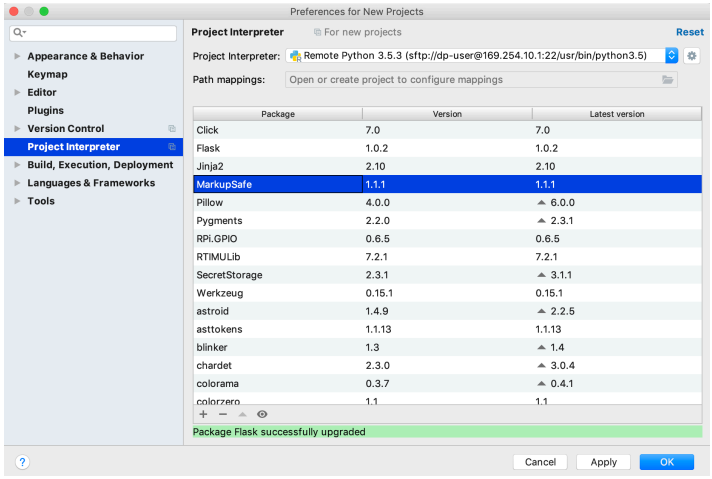
- Ga naar Preferences (for new projects) > Project interpreter > Show All
- + Voeg een nieuwe interpreter toe
- SSH interpreter
    - Host: <<adres van de RPi>>
    - Port: <<22>>
    - Username: <<dp-user>>
- Next
    - Password: <<dp-user>>
- Standaard verwijst hij naar de folder van Python 2. Pas dit aan naar de folder van de laatst ondersteunde Python versie op je RPi (3.5)
    - Interpreter: /usr/bin/python3.5
- In pycharm zie je nu een overzicht van alle geinstalleeerde packages op deze interpreter. Je kan de packages eventueel updaten naar de laatste versie.







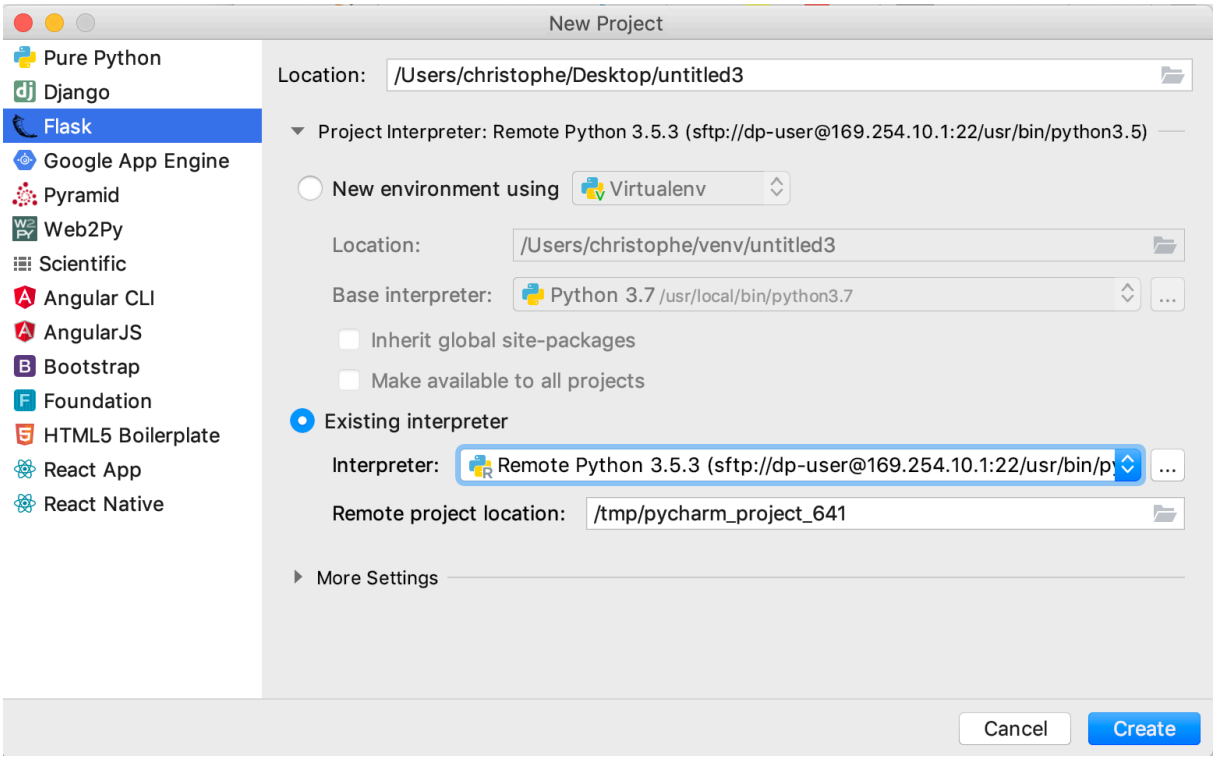




2.2.2 Nieuw Flask project aanmaken naar RPi

Kies bij de interpreter voor de interpreter van je RPi.

Je zal zien dat er de locatie bijkomt waar PyCharm de files naartoe kopiert.


New Project
Pure Python
Django
Flask
Google App Engine
Pyramid
Web2Py
Scientific
Angular CLI
AngularJS
Bootstrap
Foundation
HTML5 Boilerplate
React App
React Native
Location: /Users/christophe/Desktop/untitled3
Project Interpreter: Remote Python 3.5.3 (sftp://dp-user@169.254.10.1:22/usr/bin/python3.5)
New environment using Virtualenv
Location: /Users/christophe/venv/untitled3
Base interpreter: Python 3.7 /usr/local/bin/python3.7
Inherit global site-packages
Make available to all projects
Existing interpreter
Interpreter: Remote Python 3.5.3 (sftp://dp-user@169.254.10.1:22/usr/bin/p
Remote project location: /tmp/pycharm_project_641
More Settings
Cancel
Create

3 Doe het licht maar uit!

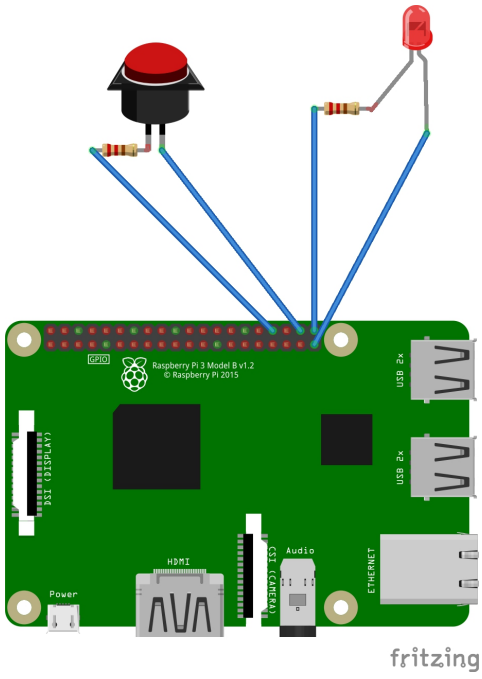
Bij deze oefening zullen we gebruikmaken van de Raspberry Pi met de image van Device Programming 1. Deze kan je vinden op http://edu.laprudence.be/dp1/image/raspberrypi_dp_4GB.img.zip

Schrijf deze naar de SD-kaart start je Raspi op en verbind deze met je laptop zodat je via APIPA kan werken.

3.1 Benodigdheden

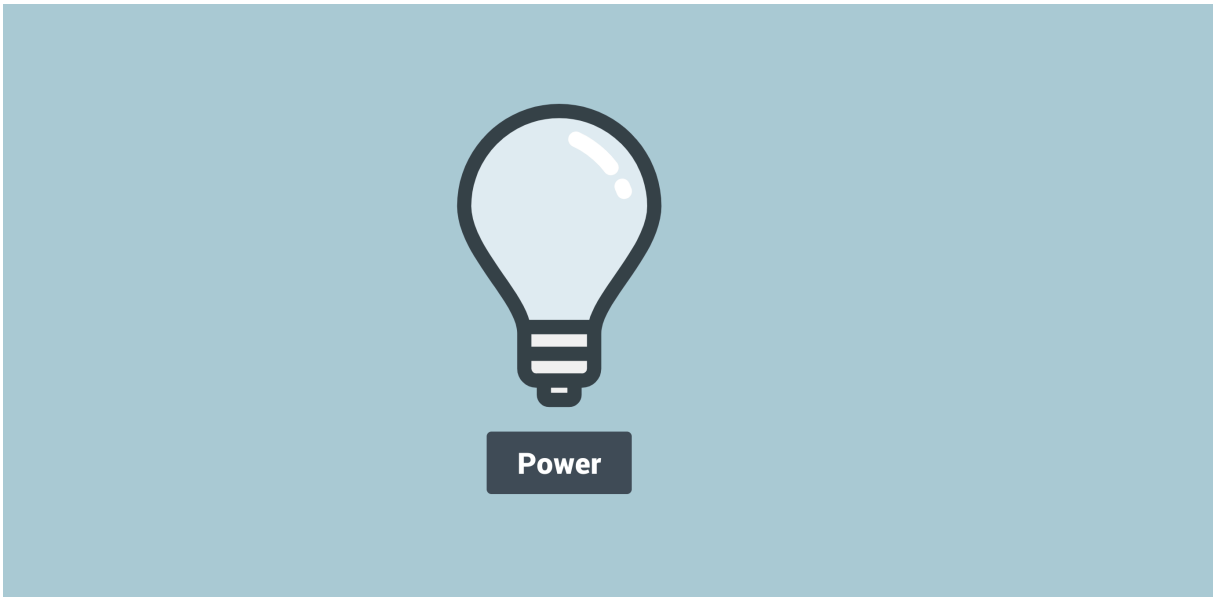
Voor dit labo hebben we twee extra componenten nodig die je tijdens de les prototyping en/of de les datacom gemaakt hebt, een gesoldeerde drukknop en een gesoldeerde led.
Deze zullen we verbinden met de correcte GPIO pinnen van onze Raspi.

We verbinden het led je met de ground van pin 39 en op pin 40 (GPIO21). De knop verbinden we met de ground op pin 34 en op pin 38 (GPIOpin 20).

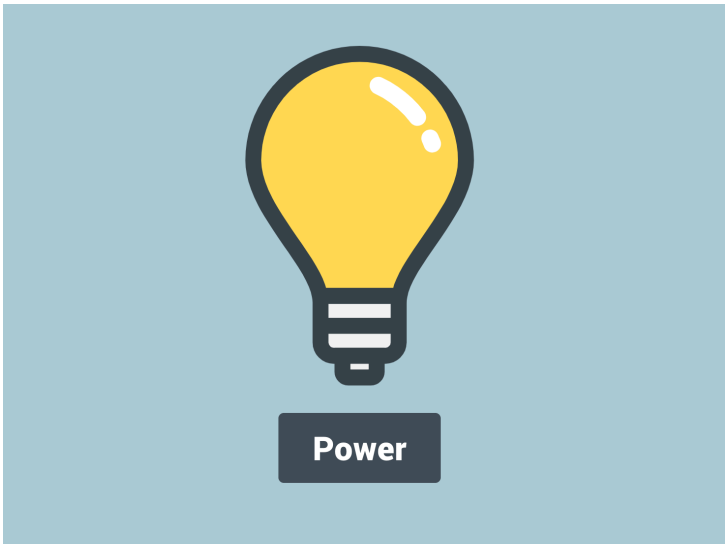


In het bronmateriaal vind je de werkende code om je voorbeeld nu te laten werken op de Raspi.
Start de app.py op (op de raspPi).
Bij het drukken op de fysieke knop zou de led moeten aangaan, bij nogmaals drukken op de fysieke knop zou het ledje terug moeten uitgaan.

Nu zullen de frontend programmeren met sockets, zodat je de led kan aan en uitzetten vanop een website.



Bij het drukken op de powerknop zullen met via socket een emit doen naar de backend waar we dezelfde functie zullen uitvoeren als bij het drukken op de fysieke knop, waarna we een emit terugsturen naar de frontend.
Hierna zullen we de klasse “active” toevoegen aan de lightbulb, zodat die oplicht.



Procesbeschrijving:

- Connectie maken met de socket
- Bij het drukken op de 'power'knop een emit versturen naar de backend
- In de backend de emit opvangen en het ledje toggelen
- Vanuit de backend een emit naar de frontend om te laten weten dat het licht getoggled is
- In de frontend de lightbulb togglen