Welcome to Culture WorkSpace

Our Host:

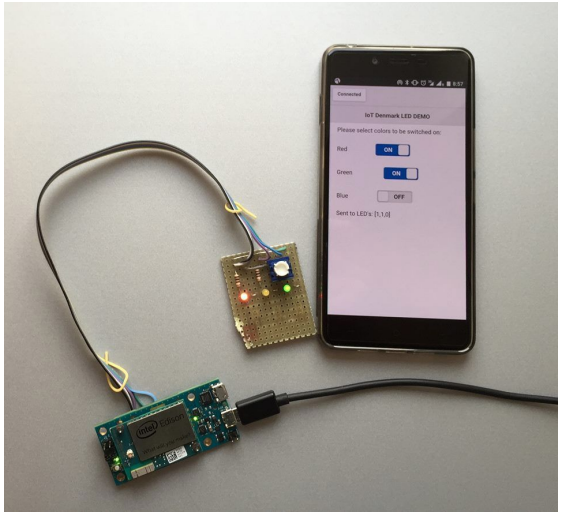
MORTEN ETZERODT

Culture Workspace,
Valdemarsgade 18B, 8000 Aarhus C.

IoT Denmark

Why?

IoT METEOR



History

1 - Intro and open discussion

2 - Hands on prototyping tools

3 - Wireless communication for IoT

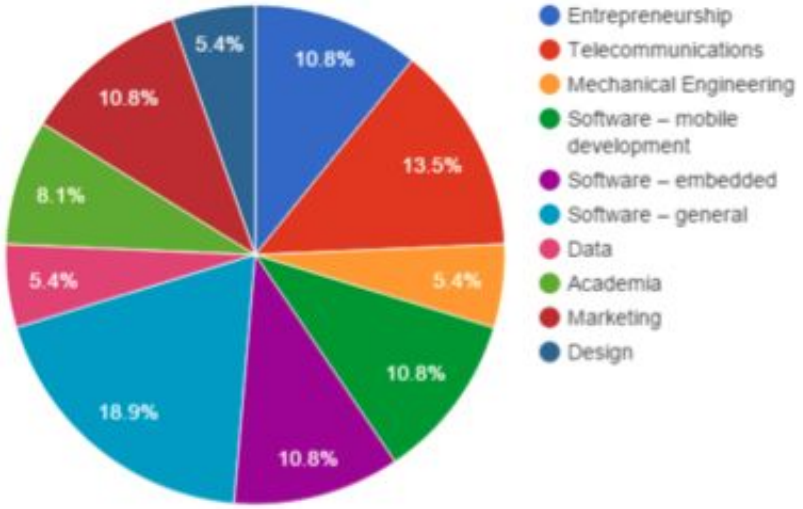
4 - PaaS and IoT

5 - Business models within IoT

6 - IoT Workshop

IoT Enthusiasts

IoT METEOR



Hello Aarhus

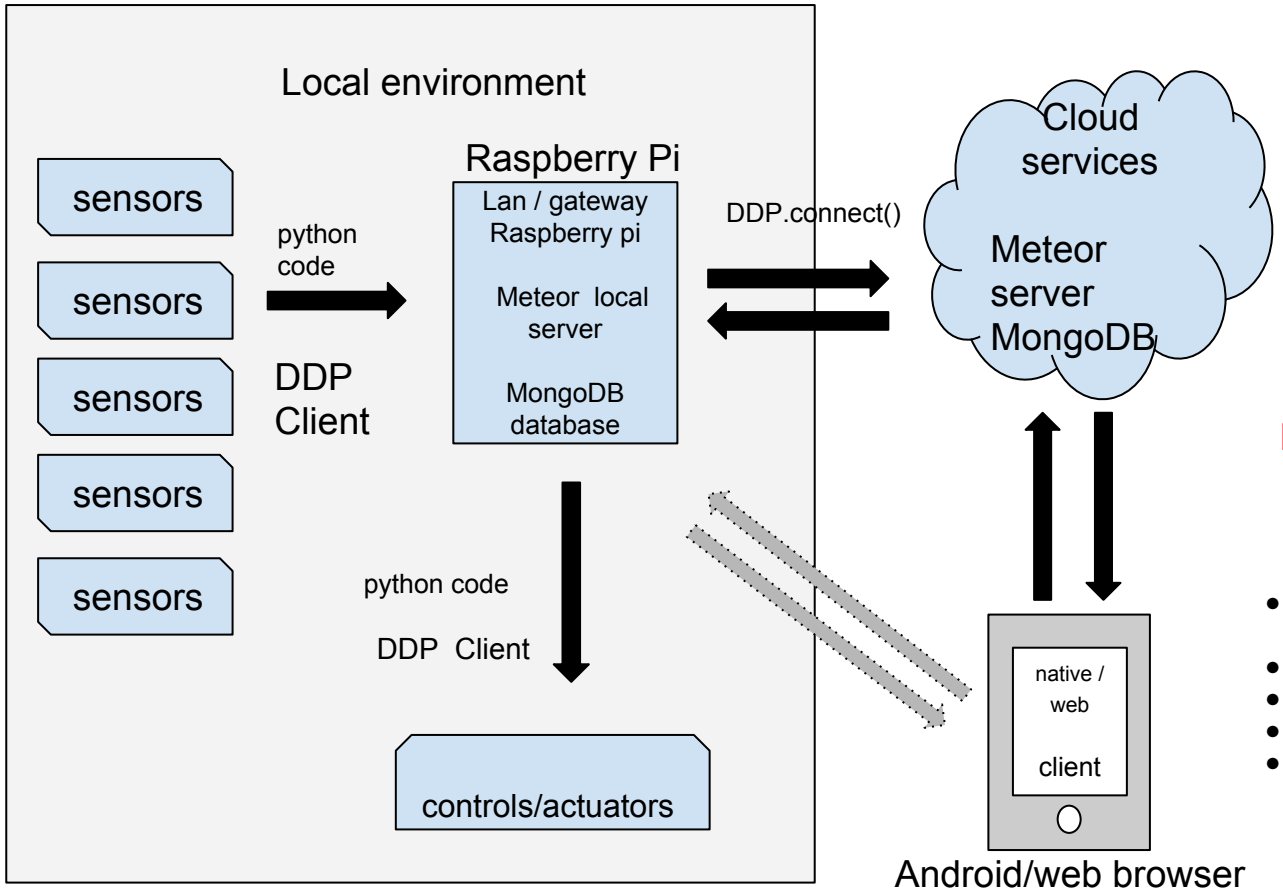
IoT METEOR

Looking forward to meeting all of you today

IoT Meetup #7 - Building IoT apps with Meteor JS Framework And Raspberry Pi Device

Overall Picture

IoT METEOR



Development areas:

- Python code sensor/actuator
- Meteor/JS on RPI/Linux
- Cloud web server
- Database - MongoDB
- Android app with notifications

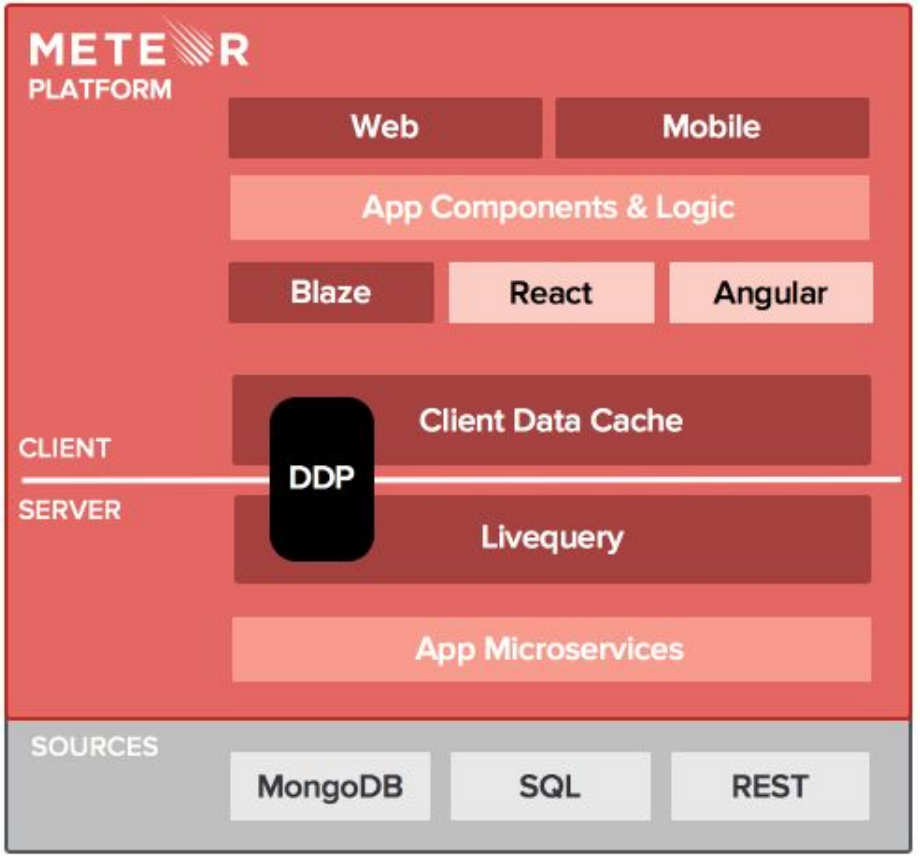
System Components

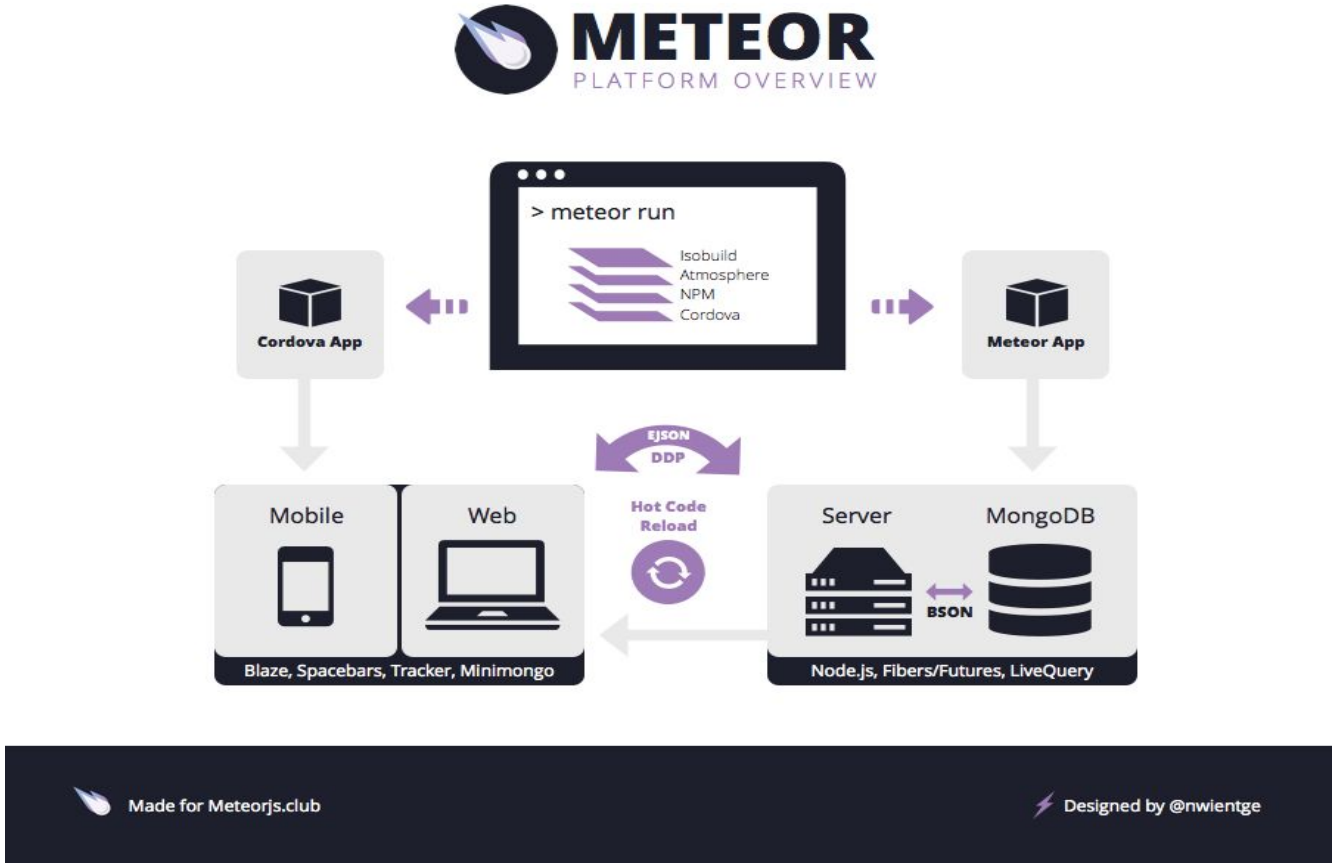
IoT METEOR

- Cloud Server - Meteor hosted at Galaxy Services
- Database - MongoDB hosted at mLab.com
- Raspberry Pi Device (Pi2 , Pi3 or Pi Zero)
- Low level code to interact with GPIO - python-ddp client
- Sensor - DHT2302 temperature and humidity sensor
- Actuator - LED diode for debugging

What is Meteor?

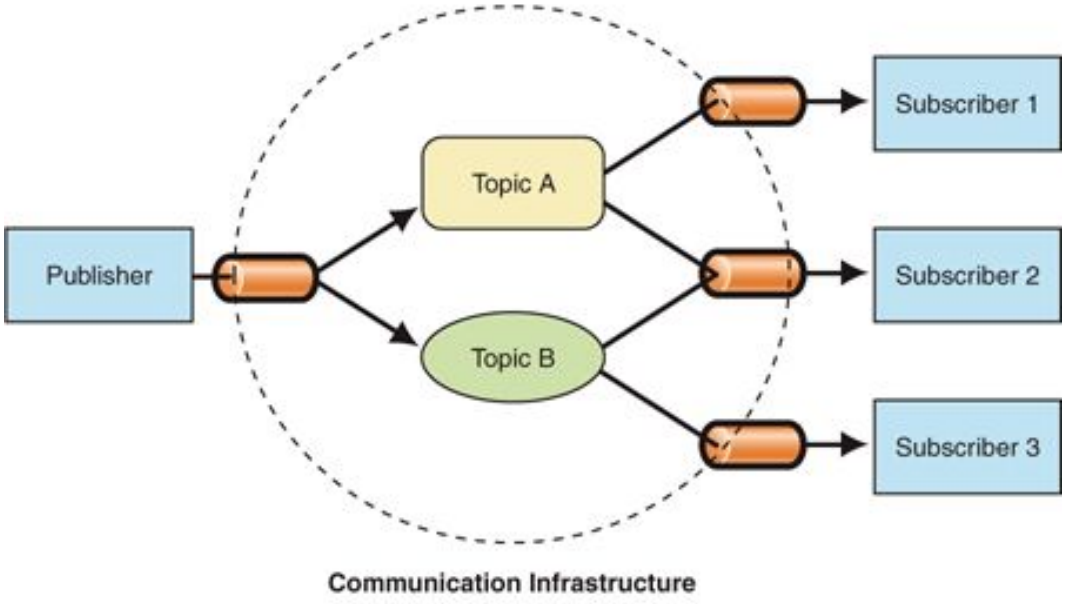
- Open-source platform for building web and mobile apps in JavaScript
- Built to power the next generation of apps
  - Rich user interfaces
  - Collaborative multi-user applications
  - Cross platform apps
  - Fast development




METEOR
PLATFORM OVERVIEW
> meteor run
Isobuild
Atmosphere
NPM
Cordova
Cordova App
Meteor App
EJSON
DDP
Hot Code
Reload
Mobile
Web
Server
MongoDB
BSON
Blaze, Spacebars, Tracker, Minimongo
Node.js, Fibers/Futures, LiveQuery
Made for Meteorjs.club
Designed by @nwientge

Publish-Subscribe Pattern IoT METEOR

Based on DDP websockets



Development Requirements IoT METEOR

- Installed Meteor platform - via meteor.com site
- Text Redactor - atom/sublimetext/nodepad++ etc.

Cloud service requirements:
Meteor Galaxy account (Cloud Server) - galaxy.meteor.com
mLab account (DataBase) - mlab.com

Application Skeleton

IoT METEOR

Directory Structure

Client

- Collections
- Compatibility
- Conf
- Lib
- Routers
- Startup
- Stylesheets
- Subscriptions
- Views

Both

- Collections
- Lib
- Packages
- Public
- Private

Server

- Lib
- Publications
- Startup

Application Skeleton

IoT METEOR

Client Folder

- HTML
- CSS
- JavaScript
- Blaze - Templates

Templates structure

- Helpers
- Events
- Attach behavior to template

Common Folder / BOTH /

- Collections - creating MongoDB instances
- Lib / router / DDP connections
- Resources
- Public
- Private

Server Folder

- Publications
- Methods
- Bootstrap /initial constants/

Configuration Files

- settings.json - deployment options
- config.push.json - Third party APIs
- mobile-config.js - cordova Android and Apple settings
- _defines.js - for global variables

Meteor internal .meteor folder

- Packages
- Platforms
- .id
- Versions
- Local folder

Installing new packages - www.atmospherejs.com

Routing Process

IoT METEOR

Iron Router

Installation from terminal:

meteor add iron-router

```
Router.configure({

  // we use the  appBody template to define the layout for the entire app
  layoutTemplate: 'main',

  // // the appNotFound template is used for unknown routes and missing devices
  notFoundTemplate: 'appNotFound',

  // // show the appLoading template whilst the subscriptions below load their data
  loadingTemplate: 'appLoading',

  // wait on the following subscriptions before rendering the page to ensure
  // the data it's expecting is present
  waitOn: function() {
     if (Meteor.user()){
    return [

        Meteor.subscribe('local-items'),
        Meteor.subscribe('local-leds'),
        Meteor.subscribe('local-sensors'),
        Meteor.subscribe('activeusers'),
        Meteor.subscribe('local-notifications'),
        Meteor.subscribe('local-notifications-rules')

    ];
     }
     else {
        return  Meteor.subscribe('activeusers');
     }

  }
});
```

Data Store with MongoDB

IoT METEOR

Three Steps:

1. Create DB instance

```
Sensors = new Meteor.Collection('sensors');
```

2. Call meteor.method from client or external machine/server

```
Template.sensorList.events({
      'click .led':  function(e) {
          e.preventDefault();
          Meteor.call('ledcontrol');
      }
});
```

3. Execute MongoDB query at the server side

```
Meteor.methods({
    ledcontrol: function(pinled) {
  Leds.update(pinled._id, {$set: { name: pinled.name, checked: ! pinled.checked , time : new Date()}});

});
```

DDP Protocol

IoT METEOR

The Distributed Data Protocol is a simple protocol building on an top of websocket and SockJS protocol. The hole internal ecosystem of the Meteor platform rely on it and as a websocket protocol provides real-time data exchange and duplex communication.

DDP is designed to provide two basic functionality operators:

- The first one is to allow the client to invoke remote procedure calls to the server.
- The second is to allow data exchange between client and the server by the publish - subscribe pattern

Connection to external Meteor Server

Messages:

DDP.connect('URL');

- `connect` (client -> server)
  - `session`: string (if trying to reconnect to an existing DDP session)
  - `version`: string (the proposed protocol version)
  - `support`: array of strings (protocol versions supported by the client, in order of preference)
- `connected` (server->client)
  - `session`: string (an identifier for the DDP session)
- `failed` (server->client)
  - `version`: string (a suggested protocol version to connect with)

http://meteorpedia.com/ - DDP and REST clients info and tutorials site

DDP Protocol

IoT METEOR

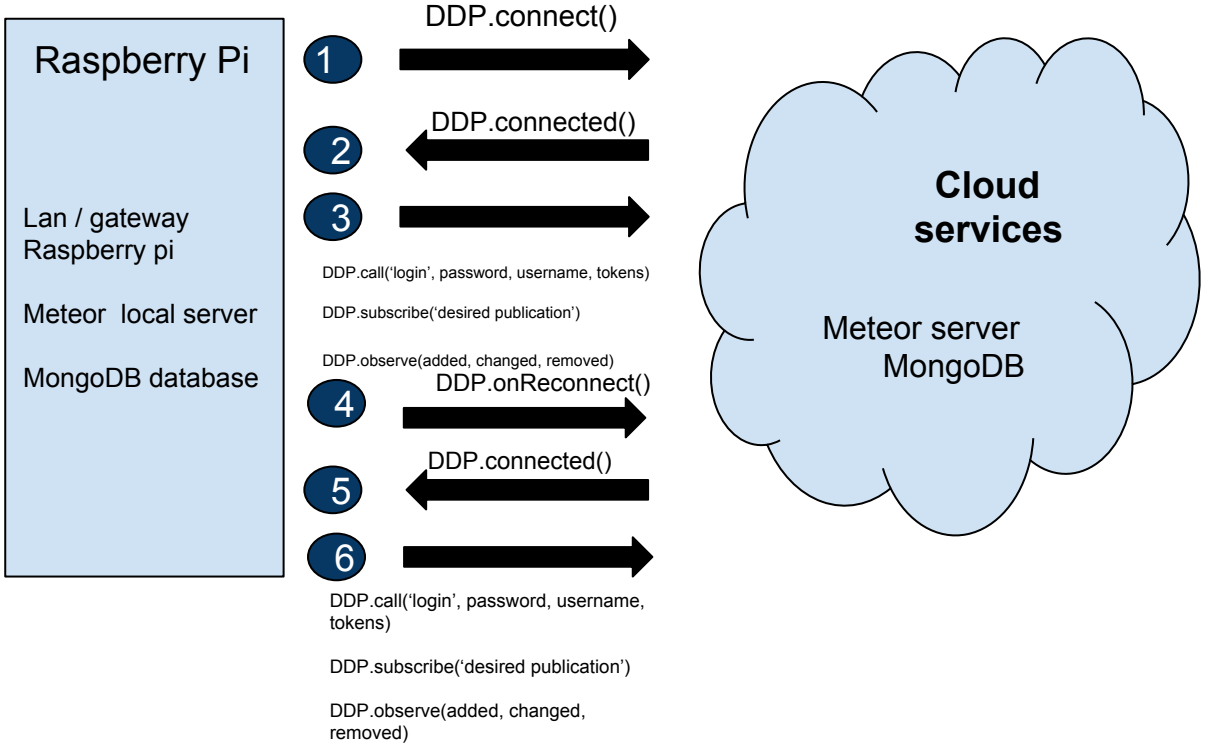
Connection to external Meteor Server

```
remote = DDP.connect('http://testddp.meteor.com/');

remote.call('login' , {"password":"123456","user":{"email":"a@a.a"}} , function(error, result){

    //THE USER ID FROM THE CLOUD USER TABLE
    console.log(result.id);


} );


RemoteServerItems = new Meteor.Collection('items', {connection: remote});

RemoteServerLeds = new Meteor.Collection('leds', {connection: remote});

RemoteServerSensors = new Meteor.Collection('sensors', {connection: remote});
```

DDP Protocol

IoT METEOR

Handling connection over DDP protocol



Requirements in order to run Meteor platform on Raspberry Pi:

- OS - Raspbian - official site /raspberrypi.org/
- Python v2 or v3 - included into OS
- Node.js - installed via apt-get package manager
- MongoDB - installed via apt-get package manager
- Python-Meteor DDP Client - installed via pip python package manager

Installation script for node.js and mongoDB on Raspberry Pi

meteor_installer.sh writen by Niels Jakobsen

```
#!/bin/bash
echo UPDATING SYSTEM SOFTWARE - UPDATE
apt-get update
echo UPDATING SYSTEM SOFTWARE - UPGRADE
apt-get dist-upgrade
echo "deb http://mirrordirector.raspbian.org/raspbian/ jessie main contrib non-free rpi" > /etc/apt/sources.list.d/jessie.list
apt-get update
echo INSTALLING MONGODB
apt-get install build-essential debian-keyring autoconf automake libtool flex bison mongodb
echo CLEANING UP
apt-get autoremove --purge
apt-get clean
rm /etc/apt/sources.list.d/jessie.list
mkdir /root/data
mkdir /root/data/db
apt-get update
echo INSTALLING NODE.JS
wget http://conoroneill.net.s3.amazonaws.com/wp-content/uploads/2015/07/node-v0.10.40-linux-arm-v7.tar.gz
tar -zxvf node-v0.10.40-linux-arm-v7.tar.gz
cd usr/local
sudo cp -R * /usr/local/
npm install -g npm
apt-get install git-core git scons build-essential scons libpcre++-dev libboost-dev libboost-program-options-dev libboost-thread-dev libboost-filesystem-dev
echo INSTALLING SCREEN AND OTHER DEPENDENCIES
apt-get install screen
apt-get install dtrx
echo DONE
```

Script running Meteor build code on Raspberry Pi

meteor_run.sh writen by Niels Jakobsen

```
#!/bin/bash
path=
projectname=
serverip=localhost
port=3000
is_setup=0
db_process=
db_up=1
function meteor_run {
        while [ $db_up == 1 ]; do
                mongo --eval "db.stats()"
                db_up=$?
        done
        echo "Runnning..."
        cd $path
        export MONGO_URL="mongodb://localhost:27017/$projectname"
        export ROOT_URL="http://$serverip"
        export PORT="$port"
        node main.js
        echo "Shutting down..."
}
function check_setup {
        #Check wether any setup needs to be done
        if [ -d "$path/programs/server/node_modules" ]; then
                is_setup=1
        fi
}
function setup {
        cd "$path/programs/server"
        npm install
        if [ -d "$path/programs/server/npm/npm-bcrypt/node_modules/bcrypt" ]; then
                rm -rf "$path/programs/server/npm/npm-bcrypt/node_modules/bcrypt"
                npm install bcrypt
        fi
}
function unwrap {
        if [ ! -d "$path" ]; then
                echo Unwrapping files...
                tar -xzf "$path.tar.gz"
                mv bundle "$projectname"
        fi
}
function start_db {
        echo Starting DB...
        export LC_ALL=C
        mongod --journal --dbpath /root/data/db &
}
while [ "$1" != "" ];do
        case $1 in
                -d | --directory )      shift
                                        path=$1
                                        ;;
                -n | --name )   shift
                                        projectname=$1
                                        ;;
                -i | --ip )     shift
                                        serverip=$1
                                        ;;
                -p | --port )   shift
                                        port=$1
                                        ;;
        esac
        shift
done
db_process=$(ps aux | grep -v grep | grep mongodb)
echo "$db_process"
if [ "$db_process" == "" ];then
        start_db
fi
unwrap
echo Checking setup...
check_setup
if [ $is_setup -eq 0 ]; then
        echo Setting up...
        setup
fi
meteor_run
```

Running Meteor Build on Raspberry Pi

(assumed that you are transfer the build version of the meteor app you created from the development PC to the raspberry pi and you are now typing the command at the raspberry pi same directory)

From Terminal:

$ sudo ./meteor_run.sh -d <NAMEOFTHEBUILD.ZIP>-n <NAMEOFTHEBUILD>

Installation of Python-DDP client for Meteor

http://meteorpedia.com - hub for all existing DDP clients for now
or
https://github.com/hharnisc/python-meteor - `$ pip install python-meteor`

Raspberry PI

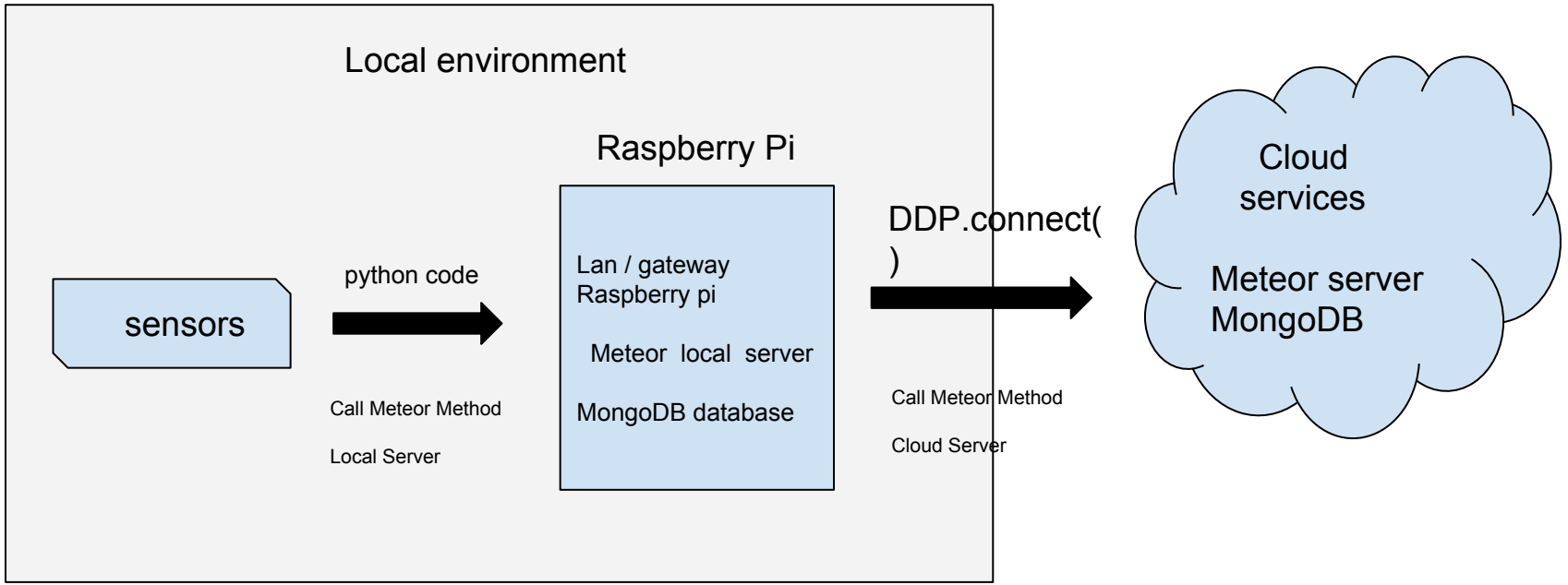
IoT METEOR

Example Python Code

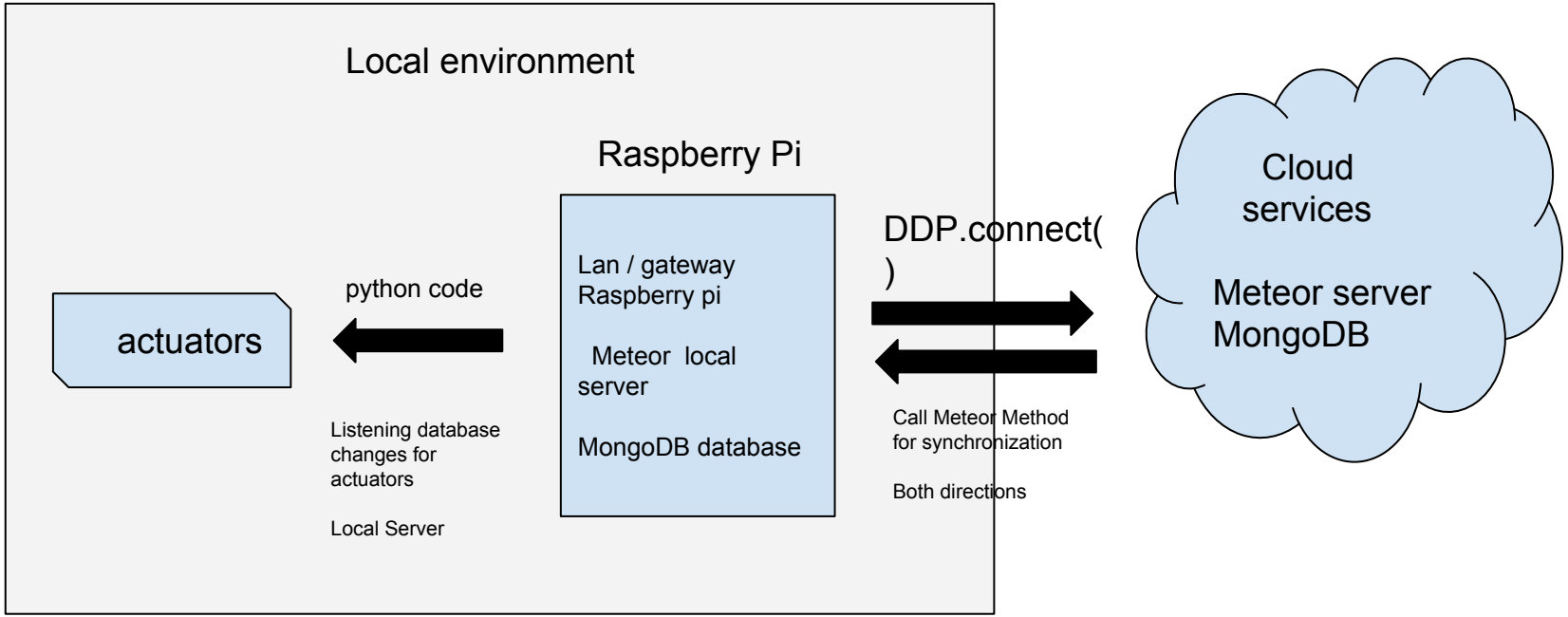
```
import time
import fcntl, socket, struct
import RPi.GPIO as GPIO
import Adafruit_DHT
import os
import commands

process_name= "meteor_run.sh"
IP = commands.getoutput("hostname -I")


starttime=time.time()

#sensor = Adafruit_DHT.DHT11
sensor = Adafruit_DHT.AM2302
pin = 4
a = 0
humidity, temperature = Adafruit_DHT.read_retry(sensor, pin)

GPIO.setmode(GPIO.BOARD)
GPIO.setup(12, GPIO.OUT)
GPIO.setup(16, GPIO.OUT)
GPIO.setup(18, GPIO.OUT)
GPIO.setup(22, GPIO.OUT)
GPIO.setup(11, GPIO.OUT)
GPIO.setup(15, GPIO.OUT)

ledcontrol1 = 12
ledcontrol2 = 16
ledcontrol3 = 18
ledcontrol4 = 22
ledcontrol5 = 11
ledcontrol6 = 15

ledname1 = 'First'
ledname2 = 'Second'
ledname3 = 'Third'
ledname4 = 'Forth'
ledname5 = 'Fifth'
ledname6 = 'Sixth'

from MeteorClient import MeteorClient




client = MeteorClient('ws://127.0.0.1:3000/websocket')
# client = MeteorClient('ws://testddp.meteor.com/websocket')

```

smartag-python.py

Sensor readings data transmission flow

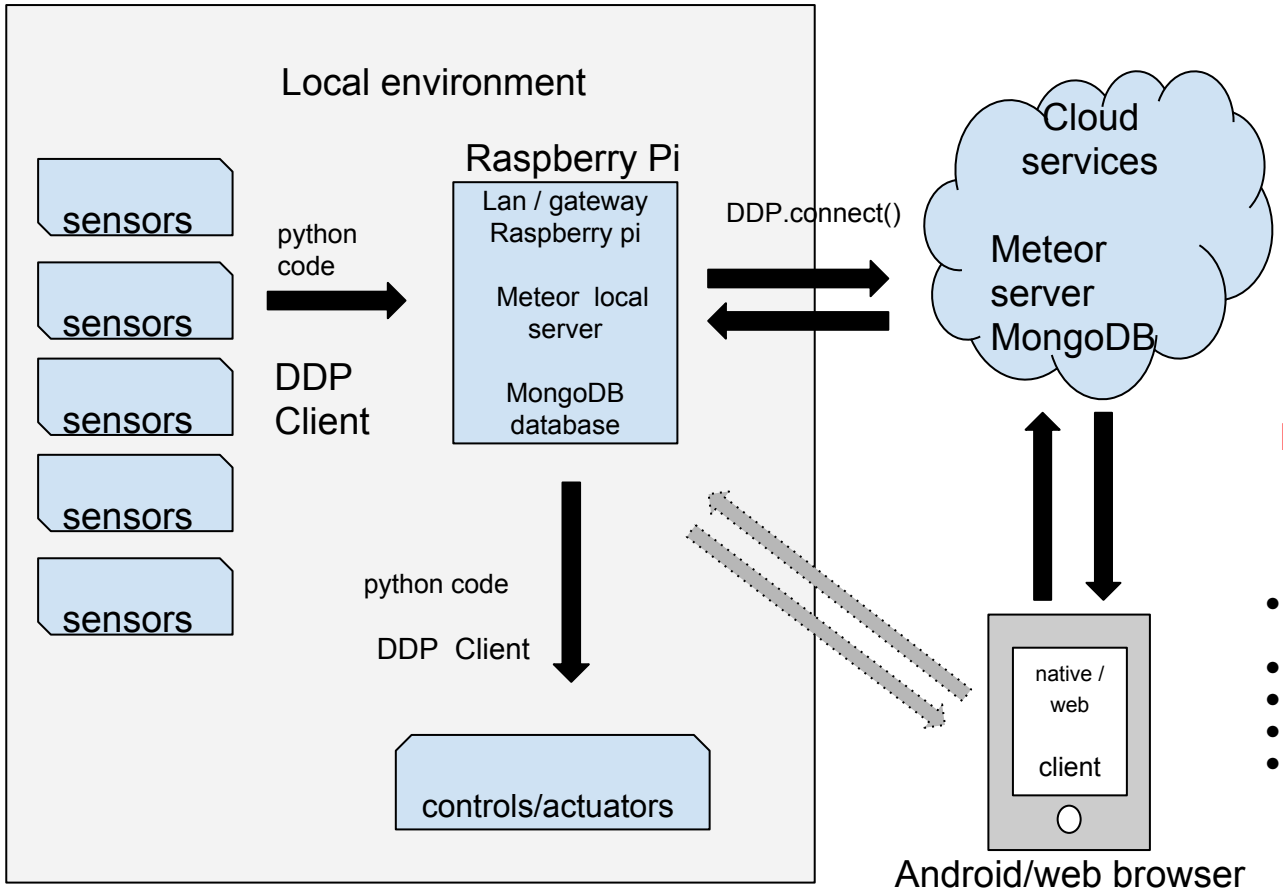


Actuators control data transmission flow



Overall Picture

IoT METEOR

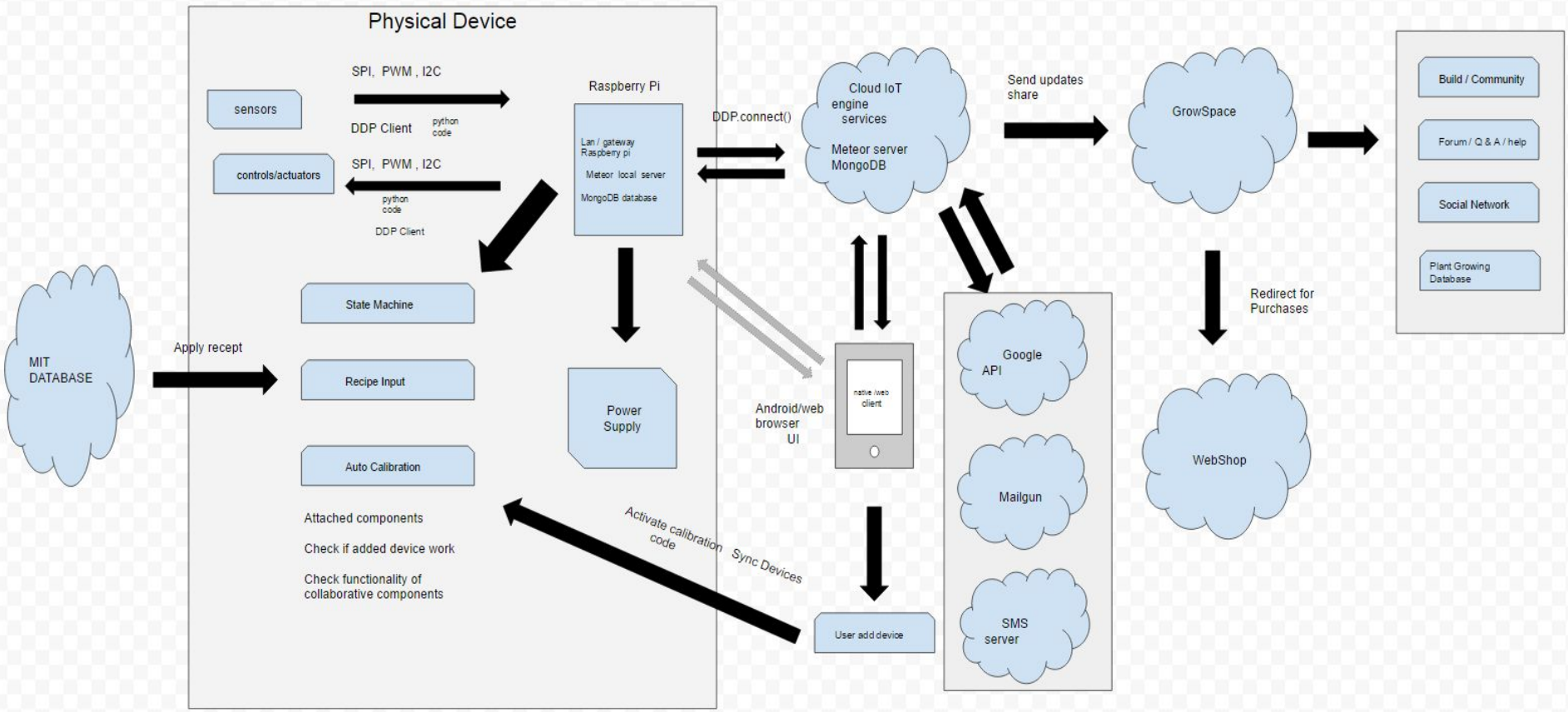


Development areas:

- Python code sensor/actuator
- Meteor/JS on RPI/Linux
- Cloud web server
- Database - MongoDB
- Android app with notifications

MyGrow Overall

IoT METEOR



THE END!

Thank you for the participation

and enjoy the networking :)