3D models for food processing systems
Marel

Developer Manual

BSc Computer Science
Spring 2018

Authors:
Daníel Freyr Sigurðsson
Egill Jónsson
Helgi Björn Bergmann
Örlygur Ólafsson

Introduction

This solution is a graphical simulation of a running SENSOR-X-302. It utilizes the web socket of the PLUTO system running on the machines and replicates the state of the machine, for maintenance and monitoring purposes. It has a graphical interface which pinpoints exactly where errors occur on a 3D representation of the machine. As for diagnostics, the solution offers detailed information about the components inside the machine.

Ownership

The solution was developed by students from Reykjavík University for Marel. All files used in the solution are considered a product of Marel. Consult with the product development team before further development and/or use of these files.

Acquiring the solution

The solutions repository is hosted and available on Bitbucket. Use the following link:
https://bitbucket.org/lokteamrefresh/3ddisplay/src/master/
For access privileges contact: orlygurolafs@gmail.com
Make sure to include in the email, your preferred email address and contact information.

Set up the production environment

The production environment is very similar to the running environment, for developing within the solution all you need is the repository. However for testing the solution the running environment has to be set up. The setup of the environment has been mostly automated for Linux, for Windows some of the steps have to be done manually (as described).

Linux

1. Prerequisites - The solution is a internet browser based and requires a internet browser that can read and compile html5 documents with corresponding CSS. This step will not be included in this guide. We recommend Mozilla Firefox to run the solution.
In addition you'll need nodejs and npm, installation of which are both included in step three, to run the dependency installer.

2. Acquiring privileges - In order to be able to run this setup guide, first off make sure that the production.sh script has execution privileges. This is done by navigating to this folder in a terminal and writing the following command:

```
$ chmod +x production.sh
```

3. Running the dependency installer - This script will acquire and install nodejs, npm, postgres, set up the user for the database, give the user required privileges, set up the database and tables as well as populating them with corresponding data. To run the script, make sure you are logged in as the administrator with administrator privileges. Navigate to the folder including the production.sh file (included in the repository of this solution) and in the terminal command prompt input the following:

```
./production.sh
```

4. Setting up a server - To run the solution you'll need to start a local server that receives and manipulates GET and POST requests from your browser done via node and npm. The server has a few dependencies that are required to run. These dependencies will be acquired automatically via express and npm. To do so, open a terminal window and navigate to the root folder of the solution (~/path-to-solution/3ddisplay) then typing in the following command:

```
npm install
```

The setup of the server should now be complete. However you might need to modify the config file for postgresql to allow local network traffic through to the server. This step might not always be necessary and is described in the TROUBLESHOOTING section of this document. (For advanced users the config file will be located in "/etc/postgresql/*/main/pg_hba.conf").

Windows

1. Prerequisites -first off you'll need a few programs that the solution builds on, they are as follows:
Internet Browser - The recommended browser for this is Mozilla Firefox, available at: https://www.mozilla.org/en-US/firefox/new/
Nodejs - The recommended version for this solution is (current stable) 8.11.1 available at: https://nodejs.org/en/
POSTGRES SQL - The tool for working with our database, this and PGADMIN are available at: https://www.postgresql.org/
Node Package Manager - Although this is included in the list of dependencies, this is bundled in with node. So after you've acquired node you should have npm installed as well. Documentation for npm is available with this link: https://www.npmjs.com/package/npm

2. Setting up database - For full functionality of the solution to work, the database and its tables need to be correctly modified and set-up.
To do this open a command prompt (located at: C:\Users\user\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\System Tools) and in the terminal, navigate to 'path-to-project/3ddisplay/public/BASH'. From there, input the following commands:

`psql -d postgres -U postgres -a -f CREATEUSER.sql`

followed by:

`psql -d marel -U marel -a -f CREATETABLES-WIN.sql`

In some cases psql won't be recognized by the command prompt and requires an additional setup step. This is described in the TROUBLESHOOTING section below.

Now the database and its tables should be setup.

3. Setting up a server - To setup the server a package.json file needs to be generated, however npm will compile this file for you. From the command prompt navigate to "path-to-project/3ddisplay" (where path-to-project is the path to the storage on which the solution is stored) and input the following command:

`npm install`

The solution should be setup and ready to run.

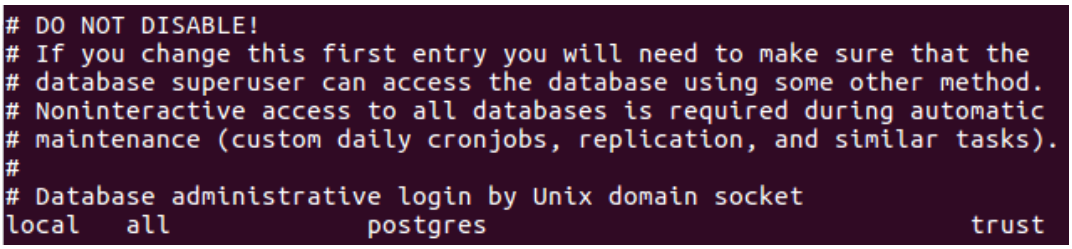
TROUBLESHOOTING

Linux

Modifying the config file of postgresql - In some cases starting a server as localhost will receive a fatal error, this may be due to the config file being set to reject such traffic. To open postgres SQL for localhost traffic, make sure you are logged in as administrator and open a terminal window and type:

`:~$ sudo nano ~/etc/postgresql/*/main/pg_hba.conf`

This step might require a password for the currently logged in administrator linux user.
In this document scroll down to the section showed in the screenshot below.



The fields on the bottom line of the screenshot are as follows:

`# TYPE DATABASE USER ADDRESS METHOD`

The server uses the database *marel* and user *marel*, in future applications creating a line which allows your machine (ip address) or local network (CIDR) to host and receive local traffic would be advisable. But for testing and trial purposes changing the last word to *trust* will allow the connection unconditionally. "This method allows anyone that can connect to

the PostgreSQL database server to login as any PostgreSQL user they wish, without the need for a password or any other authentication" (https://www.postgresql.org/docs/9.1/static/auth-pg-hba-conf.html). This can be dangerous but serves the purpose of testing the system, as there is no sensitive data currently being stored in the database.

Once done modifying your *pg_hba.conf*, save the changes and close the editor.

After modifying the file, a service restart is required this is done by opening a terminal and input the command:

```
:~$ service postgresql restart
```

(you might be prompted with a password, use the currently logged in user password for linux).
You should now be able to start the server and access the webpage (See Running the solution).

WINDOWS

If the command "psql" is not unknown when running it from command prompt, you need to have Postgres SQL installed on your machine and have the file path configured. Control Panel -> System -> Advanced System Settings -> PATH -> EDIT. Here you append the following string to your path:
For PostgresSQL version 10

```
;C:\Program Files\PostgreSQL\10\bin
```

For PostgresSQL version 9.2

```
;C:\Program Files\PostgreSQL\9.2\bin
```

Further assistance

For further assistance in setup and configuration, contact: helgi25@simnet.is

Virtual- and physical machine connection

Virtual machine

The system itself is has a connection string that tries to connect to a previously configured virtual machine (run by other developers). The setup of a new virtual machine, a permit has to be acquired through Marel. The virtual machine runs a console application for Sensor-X-302. To run an instance of the virtual machine, download and install VMWare workstation 14 player. Available at:
https://my.vmware.com/en/web/vmware/downloads.

Once the system image file is setup and started, you'll need to change the connection string in appMain.js (located in 3ddisplay/public/javascripts/appMain.js) to the ip address of your virtual machine.

For more information on how to utilize the virtual machine for testing purposes, consult with Development team or read the operation manual of PLUTO (acquired at Marel).

Physical machine

If you wish to connect the system to a physical machine. Connect the food processing system (PLUTO) to the same network as the computer running the system (typically with an ethernet cable). Once connected acquire the IP address of the food processing system and change the connection string in appMain.js (located in: 3ddisplay/public/javascripts/appMain.js) to the IP address of the machine.

Running the solution

Linux and windows

Starting the server - To start and open the server for local traffic on port 3000 navigate to the root folder of the solution, in a terminal / command prompt window, and type in the following command:

```
node app
```

Open the solution - The solution and server should be accessible on your internet browser. Simply open a browser window and navigate to:

```
localhost:3000
```

Coding rules

For readability and ease of access the following rules were made and agreed upon in the setup stages of the solution.

JAVASCRIPT

Camel Casing for variables.
Pascal Casing for functions.
Spaces are used between operators and comparators.
Braces around functions are opened in the same line as it is declared and closed in the line below the last operation of the function.
New line between functions, however variable declaration can be in consecutive lines.
Indentation use tab instead of spaces.
Examples:

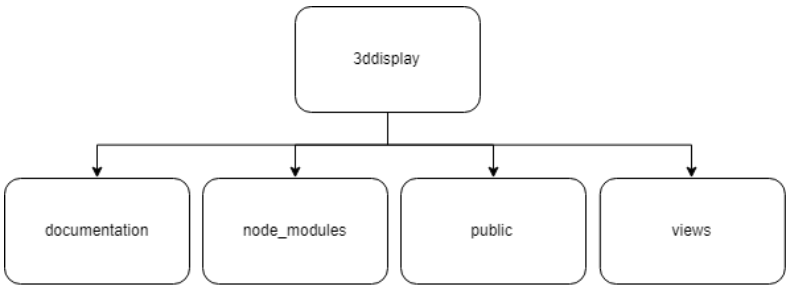
```
var developerName = 'Daniel';
function ThisIsFunction (firstName) {
	return firstName;
}

function toCelsius (fahrenheit) {
	return (5 / 9) * (fahrenheit - 32);
}
```

Folder Structure

This chapter goes a little into the solutions structure and architecture, what's located where and describes what these files and folders are used for.

Root

The root folder of the repository contains the server files of the database server, as well as the folder structure for the rest of the solution. The file "app.js" describes how the server handles GET and POST requests from the internet browser. Other files include the README.md and package.json, the readme markdown is shown in the root of the repository online. The contents of the package.json file are the compile instructions for node in order to run the server. ("~$ node app", see Running the solution).



Documentation

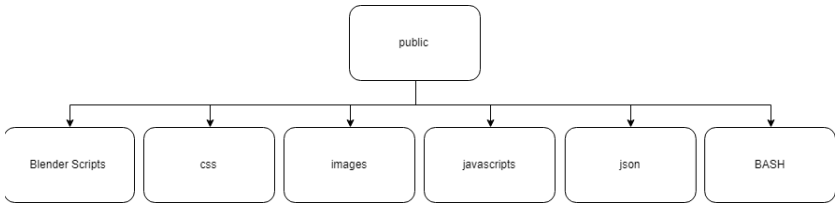
This folder holds documentations, READMEs and other various guides. These files include but are not limited to: Coding rules, Preparing a model, Operation manual and Developer manual.

Node_modules

This folder contains all dependencies, acquired by npm. When the npm install command is run in the setup process. These are the dependencies required to run the server. For further information on the contents of this folder see: https://docs.npmjs.com/files/folders

Public

This folder contains the logic of the website, it contains no files but is a directory for everything used by the html to display the site, excluding the html itself.



- **Blender Scripts** - Contains pre-processing scripts (PYTHON) used in Blender. These are tools for reducing the size of the CAD drawings.
- **CSS** - Contains the style sheets for the HTML code.
- **Images** - Contains textures for models and buttons.
- **Javascripts** - The logic (business logic layer) of the website.
    - **appMain.js** - handles all three.js logic (rendering) and data conversion
    - **three.js** - is the plugin required to run the three.js webgl
    - **WSClient** - handles all websocket communication with the machines.
- **json** - the drawings themselves. Once the drawings have been pre-processed and are ready to be displayed on the browser in the .json format.
- **BASH** - are the setup scripts and related files used in installation of the solution.
    - **Bash files** (.sh) to run installation of dependencies and query databases.
    - **SQL files** (.sql) are used by the BASH files to query the database. They were decoupled from the bash files in order to separate queries from other logic.
    - **CSV files** (.csv) are data files that are used to populate the database tables. Similarly to the SQL files, they were decoupled in order to being able to swap out database content when swapping between machines.

Views

This folder contains html files used to manage the structure of the website. These files however do not contain the .html extension. The files included in this folder contain the .dust extension which is the result of the inclusion of the dust view engine.

Architecture

In this chapter we will be running through the tools, programs and extensions used in order for the solution to run, as well as briefly going through the relation between each of them.

Database

Logic of the database server is written in javascript. Node, NPM and express provide the structure for the server to run during setup. Node also allows for runtime queries to the database from POST and GET requests to the server.

NodeJS - allows server side runtime execution of javascript (in order to query database).
Node Package Manager (NPM) - used to pack code and retrieve packs of reusable code (extensions).
Express - provides a minimalistic framework for nodejs web applications.
PostgresSQL - The server establishes a connection to the local database which is run on postgresSQL.

Website

The website attempts connection to a websocket server (virtual- or physical food processing machine). The connection to the database server happens runtime when the website attempts to query the server. Either with a mouse click on components or alarms or with the search bar.

Three.js - Javascript library that uses WebGL to display 3D computer graphics on a browser. The extension loops through rendering once per frame.
Dust View Engine - The dust view engine is a html formatting template for browsers, the advantages of this type of file is the potential use of programming comparative operators and data structures within the html itself. The use of this extension is mostly to bring forth the data structures retrieved from the database to the html/dust files. For more information on the dust view engine, visit: http://www.dustjs.com/docs/syntax/
JSON - JS Object Notation, used as a file format of models to be able to read and display on browser with Three.js

Model processing

Blender - Free open source 3D computer graphics manipulation tool. Used to remove unnecessary components, reduce the complexity of the drawings and export the drawings to the desired format (JSON).
Custom tools - Extensions written in python, executed by Blender to speed up the processes of reduction, UV mapping and removal of unnecessary components.

Three.js export plugin - to be able to export the CAD drawings to json format, you need this addon for blender. The setup of this addon is described in the repository of the addon (hosted on github). Available here:
https://github.com/mrdoob/three.js/tree/dev/utils/exporters/blender