



**FB Technik, Maschinenbau
Prof. Dr.–Ing. Agnes Pechmann**

Energy Dashboard Documentation

Prepared By

**Hari Santhosh Venkatachalam 7013776
Industrial Informatics**

Winter Semester

Important Note

The following report is a technical documentation of the Energy Dashboard of FischerTechnik which covers the technical stack to further enhance the features of the dashboard. It is targeted primarily geared towards an **Informatics** audience.

There are three important parts of the documentation.

- Part 1 covers the Setup and Installation of the required software and the instructions to get the code running
- Part 2 covers the Architecture of the design of the energy dashboard
- Part 3 describes the logic of the codebase and the places to make changes in the future going forward

Table of Contents

PART I	4
Setup and Installation	4
Development Environment Setup	4
Instructions for Running the codebase	4
PART II	5
Architecture and Design.....	5
Technology Stack User	5
Setting up SQLite Database	6
Table mapping to Labjack	9
PART III.....	10
Codebase Explained.....	10
Python Files	10
HTML/ JS Files.....	11
Changing Transact Connection	13
Modifying UI Layout.....	13
References.....	15

Table of figures

Figure 1: Energy Dashboard.....	5
Figure 2: Dashboard Architecture	6
Figure 3: DBVisualizer Setup.....	7
Figure 4: Select the energy.db file	8
Figure 5: Energy DB tables list	8
Figure 6: Labjack T7 Pro to TXT Controller connection	9
Figure 7: Project File Structure	10
Figure 8: Static files and resources.....	11
Figure 9: HTML files	11
Figure 10: Querying different ports from Labjack T7 Pro	12
Figure 11: Live Data Highcharts graph	13
Figure 12: REST Server URL	13
Figure 13: Changing the UI layout	14

PART I

Setup and Installation

Development Environment Setup

The Energy dashboard of FischerTechnik was developed completely in **Python**. Hence it is mandatory to install Python 3.7+ in your machine (Windows / Linux).

- Download **Python 3.7+** and add it to the PATH for directly executing in the command line (this will be done automatically, by the installer)
- Download a suitable IDE like **PyCharm** or **Spyder** to edit and work on Python code
- Download **DBVisualizer** free edition to work and edit on the SQLite DB
- Download **Visual Studio Code** or **Notepad++** to work and edit on the HTML and JS files

Instructions for Running the codebase

Make sure to have the FischerTechnik Setup running, power on all the Raspberry PI's, start the program in the TXT Controller and ensure the local REST Server is running. Also connect to **RoboPi-Network** to retrieve values from the REST Server and Labjack Controller

Download or Copy the codebase from HiWi drive - <\\capsrv01.hs-el.de\HiWi\ha6134 Hari Santhosh\EnergyApp>

1. Open a Command terminal in the directory 'EnergyApp'
2. Install all the dependencies needed using the command **pip install -r requirements.txt**
3. Run the Energy API by setting up the Flask server

→ set FLASK_APP=energy_api.py

→ python -m flask run

4. Run the dashboard application by launching the dashboard_energy.py

→ python dashboard_energy.py

5. Run the storage api which stores the energy readings when the order is placed by

→ python store_energy.py

6. Altogether all these **three Python scripts** will be executed simultaneously for the successful execution of the program
7. The application will be launched and running in the port 8050 (<http://localhost:8050/>)

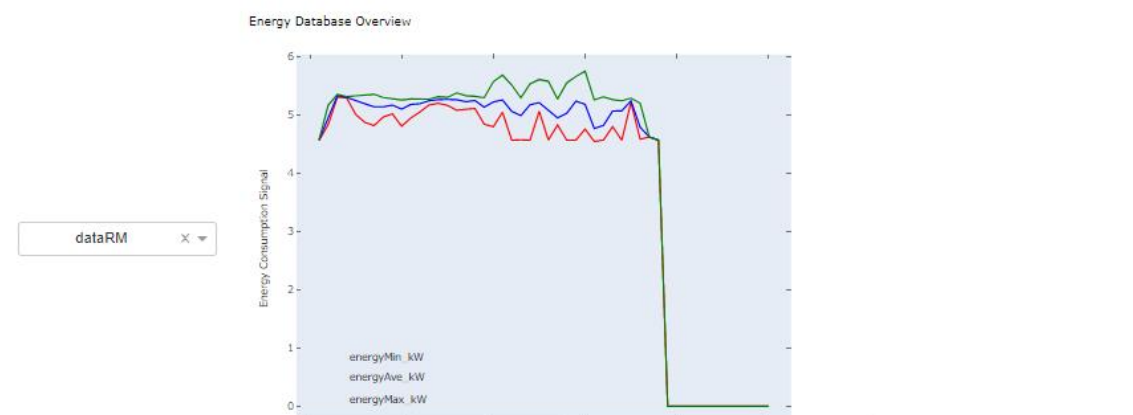
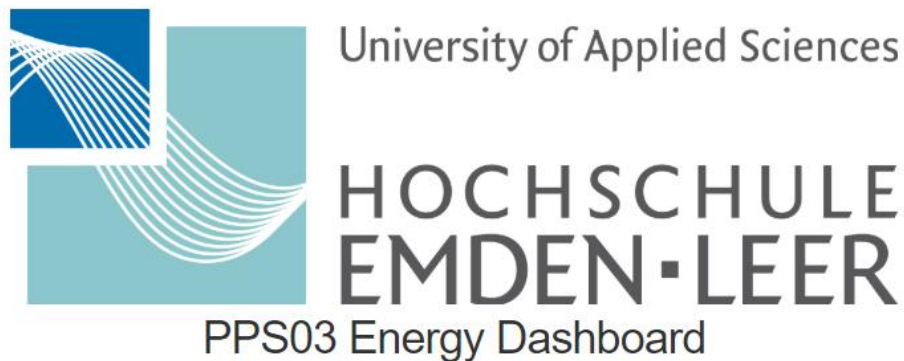


Figure 1: Energy Dashboard

PART II

Architecture and Design

Technology Stack User

1. Python **Dash** Framework has been used to primarily design the layout of the Energy Dashboard
2. Lightweight file-based **SQLite** is the Database of our choice
3. **Highcharts.js** provides us with the realtime graph plotting from Labjack
4. **MQTT** is the data transport protocol to broadcast energy information when the process is running in the FischerTechnik

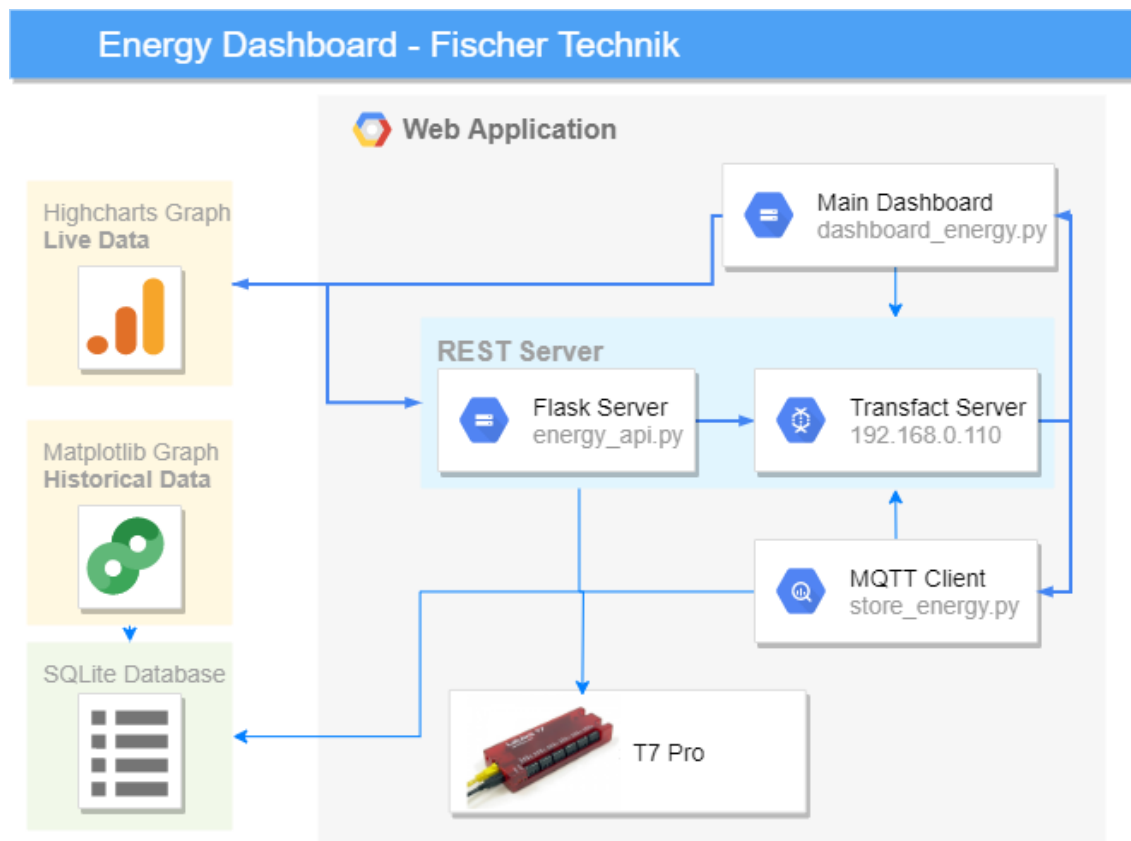


Figure 2: Dashboard Architecture

Setting up SQLite Database

1. Open **DBVisualizer** tool, and for setting it up for the first time, select on 'Create New Database Connection'
2. Give a name for the database, this will be the name which gets displayed once we load the database file inside the DBVisualizer

3. In the select 'Database Driver' dropdown choose '**SQLite**' and click on Next

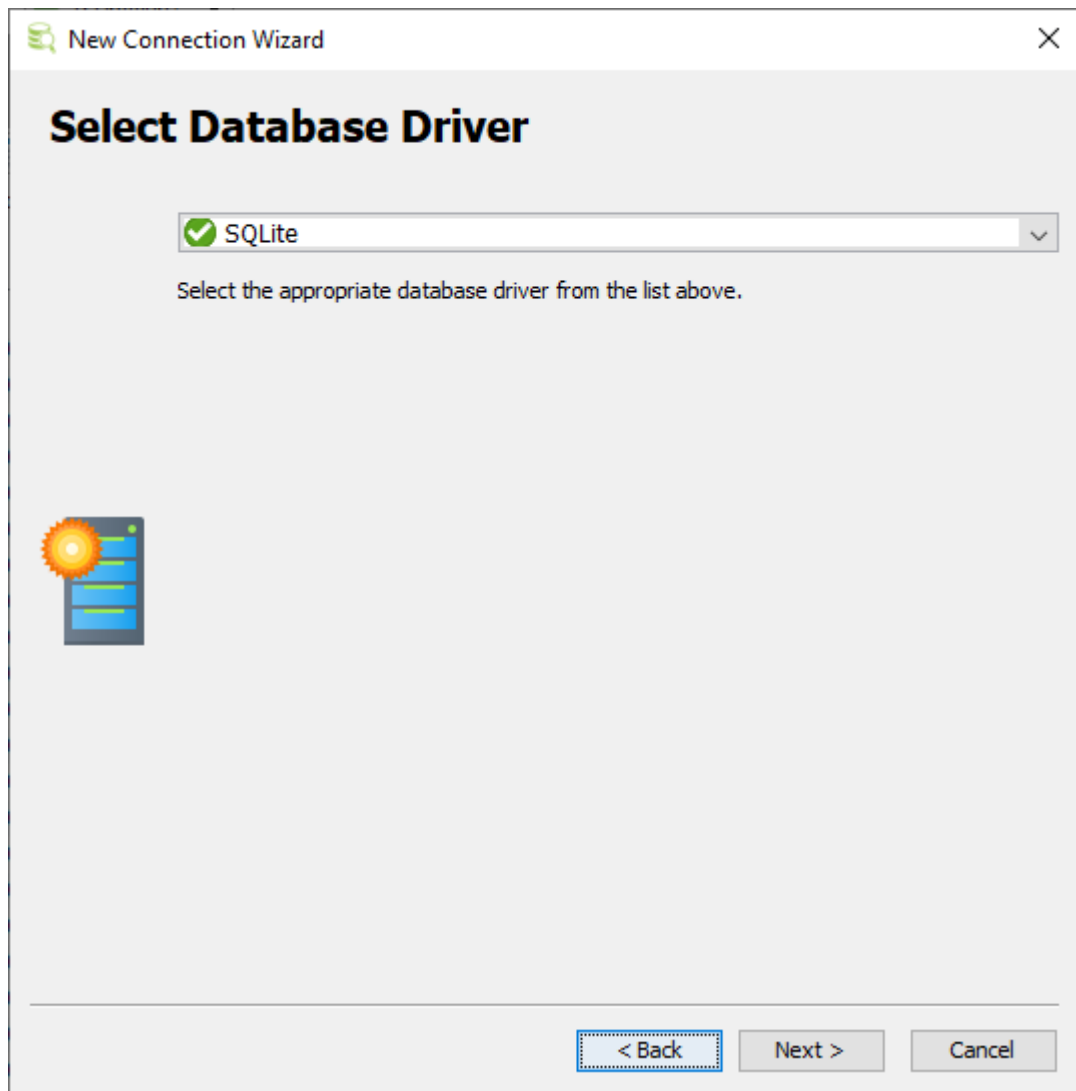


Figure 3: DBVisualizer Setup

4. In the next screen select the "**energy.db**" file and click Finish

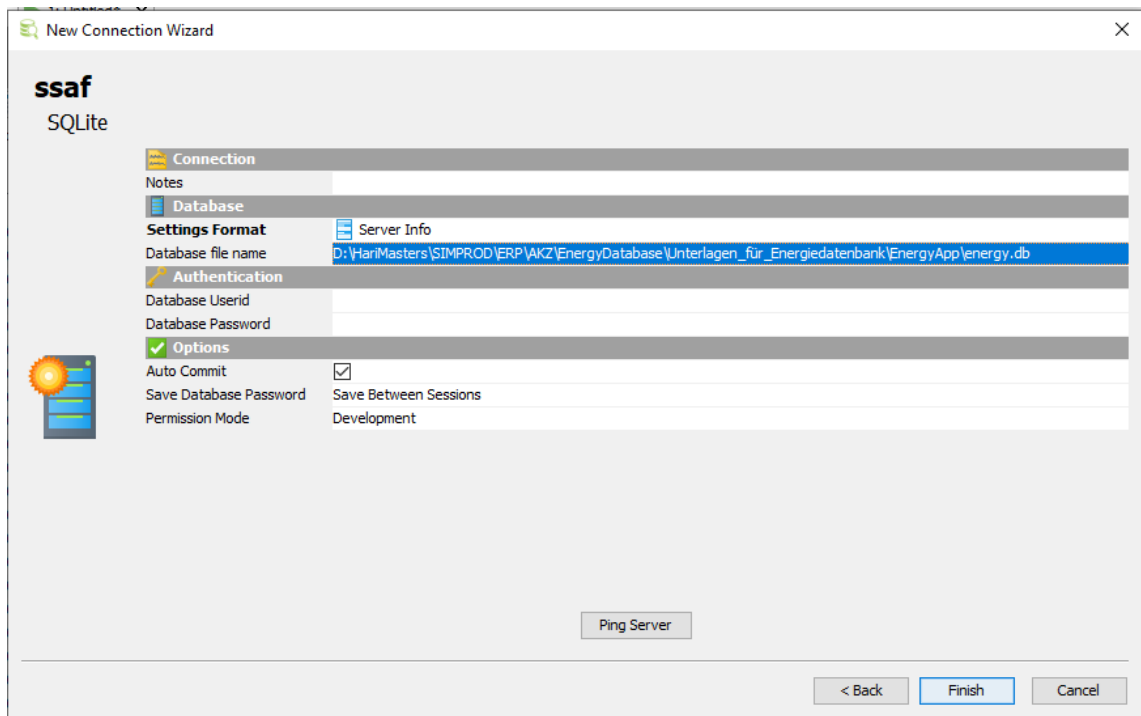


Figure 4: Select the energy.db file

Upon successful connection to the Database, the tables and the views are loaded as shown below

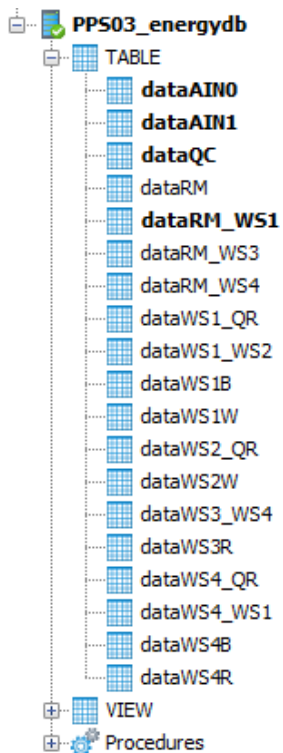


Figure 5: Energy DB tables list

Table mapping to Labjack

The following table gives an overview of the database table which reads and stores the data from the corresponding port of LabJack.

Table	Labjack Port	FischerTechnik Station
dataAIN0	AIN0	Storage/Lager
dataAIN1	AIN1	BAZ1
dataAIN2	AIN2	BAZ2
dataAIN3	AIN3	Robo
dataAIN4	AIN4	QS1
dataAIN5	AIN5	QS2
dataAIN6	AIN6	BAZ3
dataAIN7	AIN7	BAZ4

This is based on the connection to the LabJack T7 pro from the FischerTechnik TXT Controllers. When the connections are altered the logic can be changed accordingly



Figure 6: Labjack T7 Pro to TXT Controller connection

Note: In the future, by updating the 'store_energy.py' a combination of two stations can be measured and updated in the SQLite Database as a new table. The other tables in the database are originally created from the Excel 'Energy Demand FT.xlsx'

PART III

Codebase Explained

Python Files

The file structure of the project appears as shown below.

The primary Python files containing the core logic for the dashboard are highlighted as shown:

pycache	1/16/2020 3:44 PM	File folder	
assets	12/16/2019 10:20 AM	File folder	
static	1/16/2020 3:55 PM	File folder	
AKZ_PI_poweroff.py	12/18/2019 5:31 PM	JetBrains PyChar...	1 KB
bk_dashboard.py	1/10/2020 9:42 PM	JetBrains PyChar...	7 KB
dashboard_energy.py	1/21/2020 1:40 PM	JetBrains PyChar...	10 KB
Demo_real_time_plotter.py	1/13/2020 3:46 PM	JetBrains PyChar...	1 KB
Energy Demand FT.xlsx	12/11/2019 12:52 PM	Microsoft Excel W...	93 KB
energy.db	1/22/2020 6:55 PM	Data Base File	176 KB
energy_api.py	1/16/2020 3:44 PM	JetBrains PyChar...	4 KB
Labjack_reader.py	1/8/2020 10:03 PM	JetBrains PyChar...	2 KB
ordnung_db_creator.py	12/11/2019 7:07 PM	JetBrains PyChar...	1 KB
README.md	1/21/2020 3:17 PM	Markdown Source...	1 KB
Real_time_plotter.py	1/10/2020 11:57 PM	JetBrains PyChar...	2 KB
requirements.txt	1/17/2020 8:13 PM	Text Document	5 KB
RoboPi_shutdown.py	1/8/2020 10:18 PM	JetBrains PyChar...	2 KB
store_energy.py	1/22/2020 6:54 PM	JetBrains PyChar...	2 KB

Figure 7: Project File Structure

- **dashboard_energy.py:** This file contains the main structural layout of the dashboard, all User Interface (UI) related changes are to be made in this file and will be immediately reflected in the web application
- **energy_api.py:** This file consists of the Flask server which is responsible for rendering the '**static**' assets which fetches and retrieves information from the Labjack T7 Pro Controller. We **won't be changing this file** anytime as it is stable and only renders the static assets to the browser
- **store_energy.py:** This file creates a MQTT listener and starts **storing energy information to the SQLite Database** when the order is executing it's workingstep(arbeitsschritte), this file should be edited when we wanted

to **change the table name** where the information is getting logged and handle the logic accordingly for processing different working steps.

- **RoboPi_shutdown.py**: A script to shutdown all the Raspberry Pi's before switching off the system.

Note: All other python files in the workspace are utility files which are used for testing purposes to experiment and build solutions, they are **not currently used** or referenced anywhere.

HTML/ JS Files

The HTML and JS files are residing inside the static folder. They are responsible for the Live data charts which are embedded as an **Iframe** in our Energy Dashboard







	__pycache__	1/16/2020 3:44 PM	File folder	
	assets	12/16/2019 10:20 AM	File folder	
	static	1/16/2020 3:55 PM	File folder	
	AKZ_PI_poweroff.py	12/18/2019 5:31 PM	JetBrains PyChar...	1 KB
	bk_dashboard.py	1/10/2020 9:42 PM	JetBrains PyChar...	7 KB
	dashboard_energy.py	1/21/2020 1:40 PM	JetBrains PyChar...	10 KB

Figure 8: Static files and resources

Inside the static folder, we have 13 HTML files which contains the logic to get values from the Labjack T7 Pro's port.















	livedata0.html	1/20/2020 8:13 PM	HTML File	4 KB
	livedata1.html	1/20/2020 2:53 PM	HTML File	3 KB
	livedata2.html	1/20/2020 3:03 PM	HTML File	3 KB
	livedata3.html	1/20/2020 3:03 PM	HTML File	3 KB
	livedata4.html	1/20/2020 3:03 PM	HTML File	3 KB
	livedata5.html	1/16/2020 3:51 PM	HTML File	2 KB
	livedata6.html	1/16/2020 3:51 PM	HTML File	2 KB
	livedata7.html	1/16/2020 3:52 PM	HTML File	2 KB
	livedata8.html	1/16/2020 3:52 PM	HTML File	2 KB
	livedata9.html	1/16/2020 3:53 PM	HTML File	2 KB
	livedata10.html	1/16/2020 3:53 PM	HTML File	2 KB
	livedata11.html	1/16/2020 3:54 PM	HTML File	2 KB
	livedata12.html	1/16/2020 3:54 PM	HTML File	2 KB
	livedata13.html	1/16/2020 3:54 PM	HTML File	2 KB

Figure 9: HTML files

The naming of the file is done to represent the Port number of Labjack T7 Pro's port. For example, livedata0.html plots the data for AIN0 Port of Labjack

HTML File	LabJack Port Measured
livedata0.html	AIN0
livedata1.html	AIN1
livedata2.html	AIN2
livedata3.html	AIN3
livedata4.html	AIN4
livedata5.html	AIN5
livedata6.html	AIN6
livedata7.html	AIN7
livedata8.html	AIN8
livedata9.html	AIN9
livedata10.html	AIN10
livedata11.html	AIN11
livedata12.html	AIN12
livedata13.html	AIN13

The logic to combine two signals inside a single HTML can be accommodated in the future by modifying the contents of the JS snippet available inside the HTML files.

Note: The ports (AIN0 - AIN7) are the ones connected to the FischerTechnik Stations. hence the rest of the ports won't have significant value.

```
function fetchSingleData() {
    var stName = "AIN4";
    $.ajax({
        url: 'http://localhost:5000/labjackvalues/'+stName,
        success: function(point) {
            var series = chart.series[0],
                shift = series.data.length > 20; // shift if the series is
                                                // longer than 20
            // add the point
            var newpoint = JSON.parse(point);
        }
    });
}
```

Figure 10: Querying different ports from Labjack T7 Pro

Highcharts.js supports representing multiple lines in the Live chart view. A sample example is attached in the References section [1]

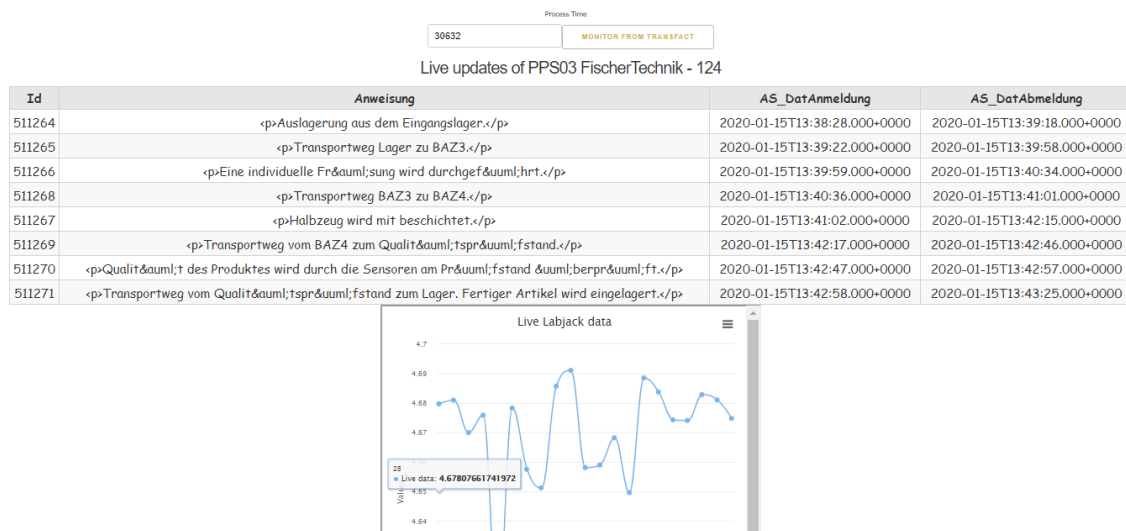


Figure 11: Live Data Highcharts graph

Changing Transfact Connection

In order to change the connection settings of Global Transfact system to Local transfact system, the **only place** to change is the **REST Server's** settings, the logic refers directly to the running REST Server and uses the information provided by it accordingly.

```
22 from matplotlib import pyplot as plt
23
24
25 url_labjack = "http://localhost:5000/labjackvalues"
26 url_restserver = "http://192.168.0.110:8080/api/arbeitschritte?chargen="
27 external_stylesheets = ['https://codepen.io/chriddyp/pen/bWLwgP.css']
```

Figure 12: REST Server URL

The dashboard_energy.py refers to the REST Server which is running on the Local IP 192.168.0.110 to get all the values to be displayed in the Energy dashboard table, hence it is not necessary to change any settings in this file.

Modifying UI Layout

The **dashboard_energy.py** UI layout is as shown, it follows a declarative syntax and modifying the contents of the page here will be reflected accordingly in the Energy Dashboard. The **app.layout** contains all the elements which are enclosed in the UI

```
86 app.layout = html.Div(style={'text-align': 'center'}, children=[  
87     html.Img(src=app.get_asset_url('emden_leer.png'), style={  
88         'height': '50%',  
89         'width': '50%'  
90     })),  
91     html.H1(children='PPS03 Energy Dashboard'),  
92     html.Div([  
93         dcc.Dropdown(  
94             id='demo-dropdown',  
95             options=[  
96                 {'label': 'dataRM', 'value': 'dataRM'},  
97                 {'label': 'dataQC', 'value': 'dataQC'},  
98                 {'label': 'dataRM_WS1', 'value': 'dataRM_WS1'},  
99                 {'label': 'dataRM_WS3', 'value': 'dataRM_WS3'},  
100                 {'label': 'dataRM_WS4', 'value': 'dataRM_WS4'}  
101             ]  
102         )  
103     ]) ])
```

Figure 13: Changing the UI layout

For instance, in order to change the primary title of the Dashboard, modifying the contents of line number 91, and changing the text of children will change the title accordingly. For controlling the interaction of different elements in the layout the best place to look and refer is the **Dash** documentation [2], which has a series of examples and explains in detail how **callback functions** are used to drive the changes in the dashboard dynamically.

To change the dropdown values and their corresponding mapping the function **update_energy_table** provides the mapping to change the iframe src to render the corresponding Live energy data, this can be later worked on to be done automatically based on the working steps (Arbeitsschritte) progress.

Contact

Hari Santhosh Venkatachalam

Email - hari.santhosh.venkatachalam@stud.hs-emden-leer.de

Telephone - +49-1786873054

References

1. Highcharts working with Live data
<https://www.highcharts.com/docs/working-with-data/live-data>
2. Python Dash documentation <https://dash.plot.ly/>
3. Labjack Python Library
<https://labjack.com/support/software/examples/ljm/python>
4. Labjack Tools to access the T7 Pro using GUI Kepler
<https://labjack.com/support/software/installers/ljm>
5. Flask Documentation <https://flask-doc.readthedocs.io/en/latest/>
6. Python SQLite DB documentation
<https://docs.python.org/2/library/sqlite3.html>
7. Pandas Ecosystem documentation <https://pandas.pydata.org/pandas-docs/stable/ecosystem.html>
8. Pandas API reference documentation <https://pandas.pydata.org/pandas-docs/stable/reference/index.html>
9. Pandas SQL functionality documentation
https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.to_sql.html
10. Pip Installation tutorial <https://medium.com/@boscacci/why-and-how-to-make-a-requirements-txt-f329c685181e>