

Innovaatioprojekti (Robot)

User Guide

April 2020

Information and communication technology

Software production

TABLE OF CONTENTS

[1 PREFACE 3](#_Toc38902092)

[2 OPC – SERVER 4](#_Toc38902093)

[2.1 ABB IRC5 OPC Server 4](#_Toc38902094)

[2.2 Using ABB IRC5 OPC Server Configuration 4](#_Toc38902095)

[3 CONNECTION FROM OPC SERVER TO DATABASE 7](#_Toc38902096)

[3.1 Install Python and the required libraries 7](#_Toc38902097)

[3.2 Database configuration (How it’s done in Python, except 1.) 8](#_Toc38902098)

[3.3 Code execution (In Python 3.6) 9](#_Toc38902099)

[3.4 For – Loop code for data sending to Postgres 11](#_Toc38902100)

[3.5 Data from OPC server 11](#_Toc38902101)

[4 DATABASE/SERVER 12](#_Toc38902102)

[4.1 Installing required software to the server 12](#_Toc38902103)

[4.2 Database setup 13](#_Toc38902104)

[5 FRONTEND 14](#_Toc38902105)

[5.1 Technologies used 14](#_Toc38902106)

[5.2 Installation and running 14](#_Toc38902107)

[5.3 Points to consider 14](#_Toc38902108)

[6 FURTHER DEVELOPMENT 16](#_Toc38902109)

[6.1 OPC Server and Python connection 16](#_Toc38902110)

[6.2 Database/server 16](#_Toc38902111)

[6.3 Frontend 16](#_Toc38902112)

[7 SOURCES 17](#_Toc38902113)

[ATTACHMENTS 18](#_Toc38902114)

# PREFACE

The porpoise of this document is to give an idea of the “Innovaatioprojekti” course’s robot project that has been done with the cooperation of mechanical engineering students. The goal is to give a clear guide to get the setup working and ready for further development.

There are three main areas to consider when setting up the environment. These areas are the OPC-server which gathers data from the robot, database/server which stores the data and the frontend which fetches and displays the data.

# OPC – SERVER

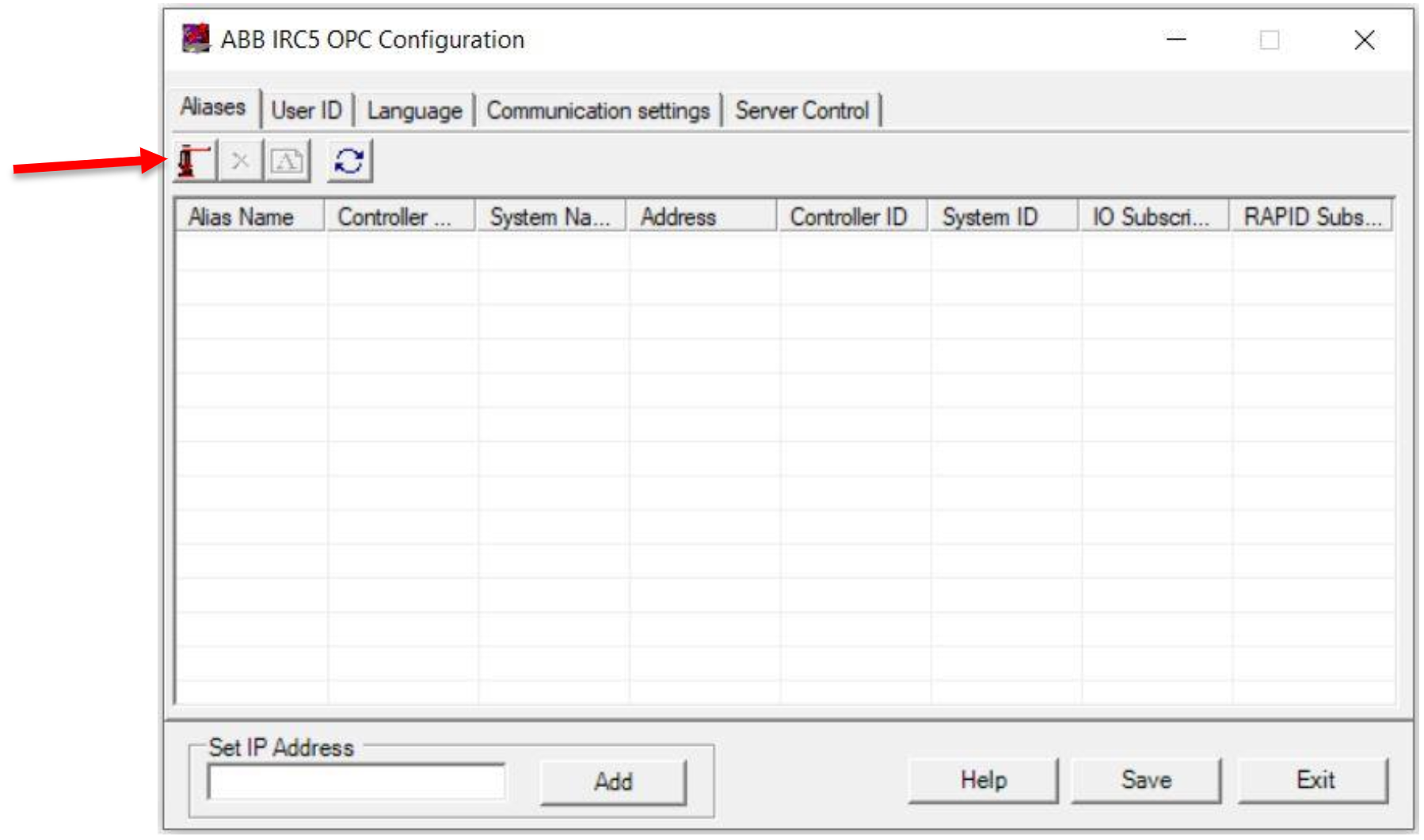
## ABB IRC5 OPC Server

The hardware in this project was TAMK's robot ABB IRB 4600 and the OPC server was ABB IRC5 OPC Server Configuration.

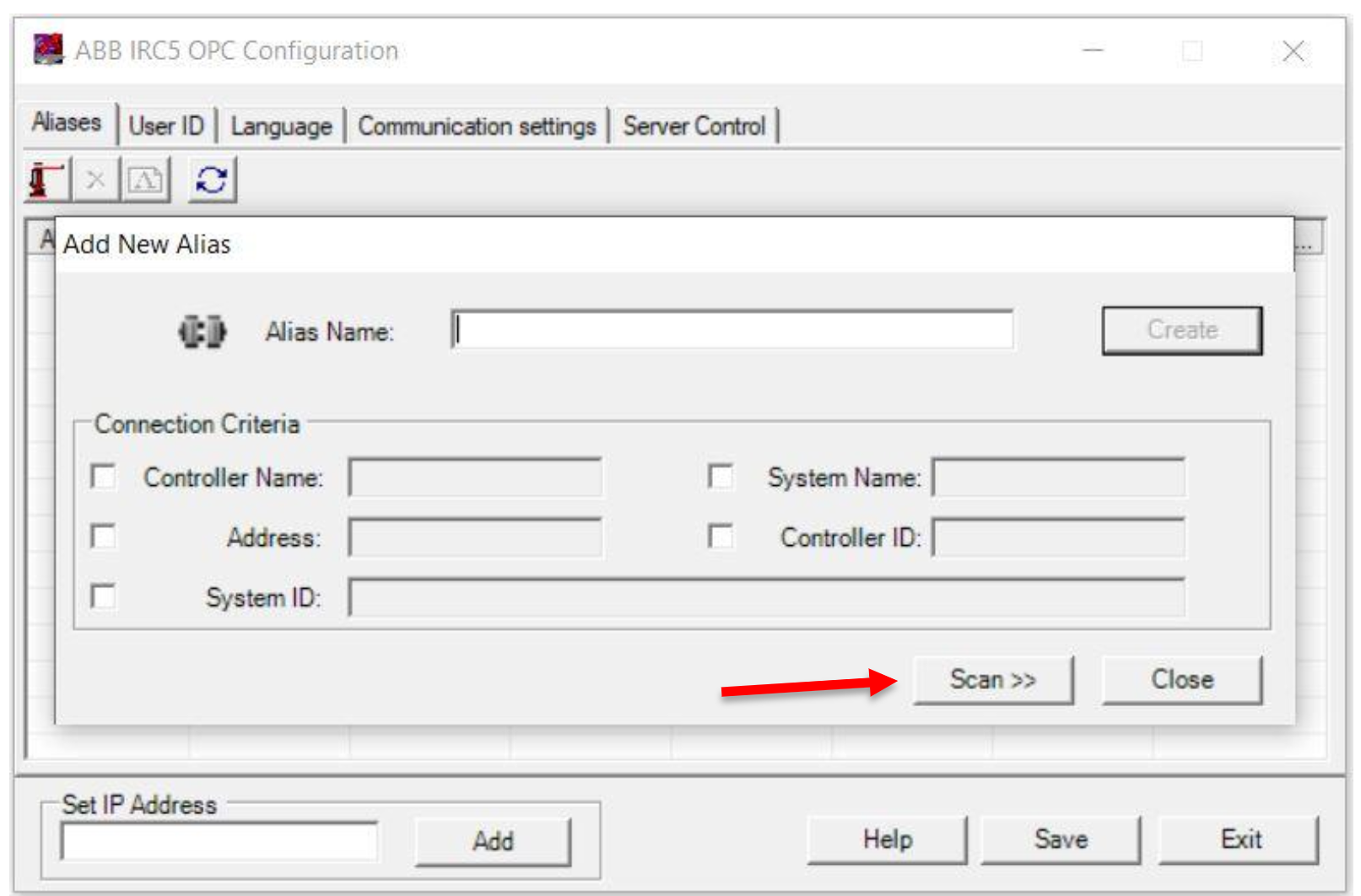
ABB IRC5 OPC Server Configuration can be downloaded from: <http://developer-center.robotstudio.com/opcserver>. ABB IRC5 OPC Server Configuration is used to create and manage ABB IRC5 robot controller gadgets. The alias represents a single robot controller. Each robot controller must be customized for OPC Server to use them.

## Using ABB IRC5 OPC Server Configuration

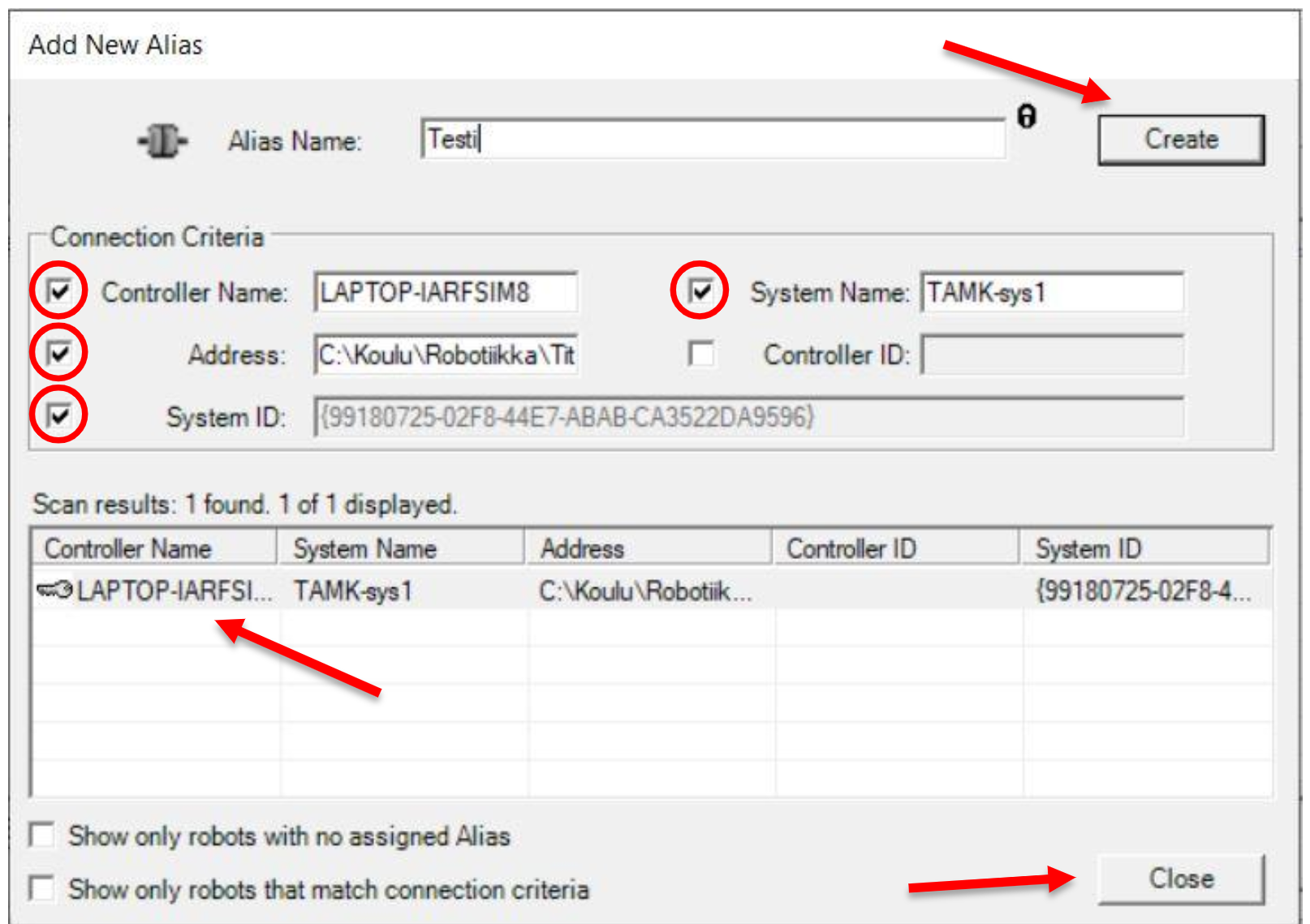
First add a new alias by clicking the “Add New Alias” button.



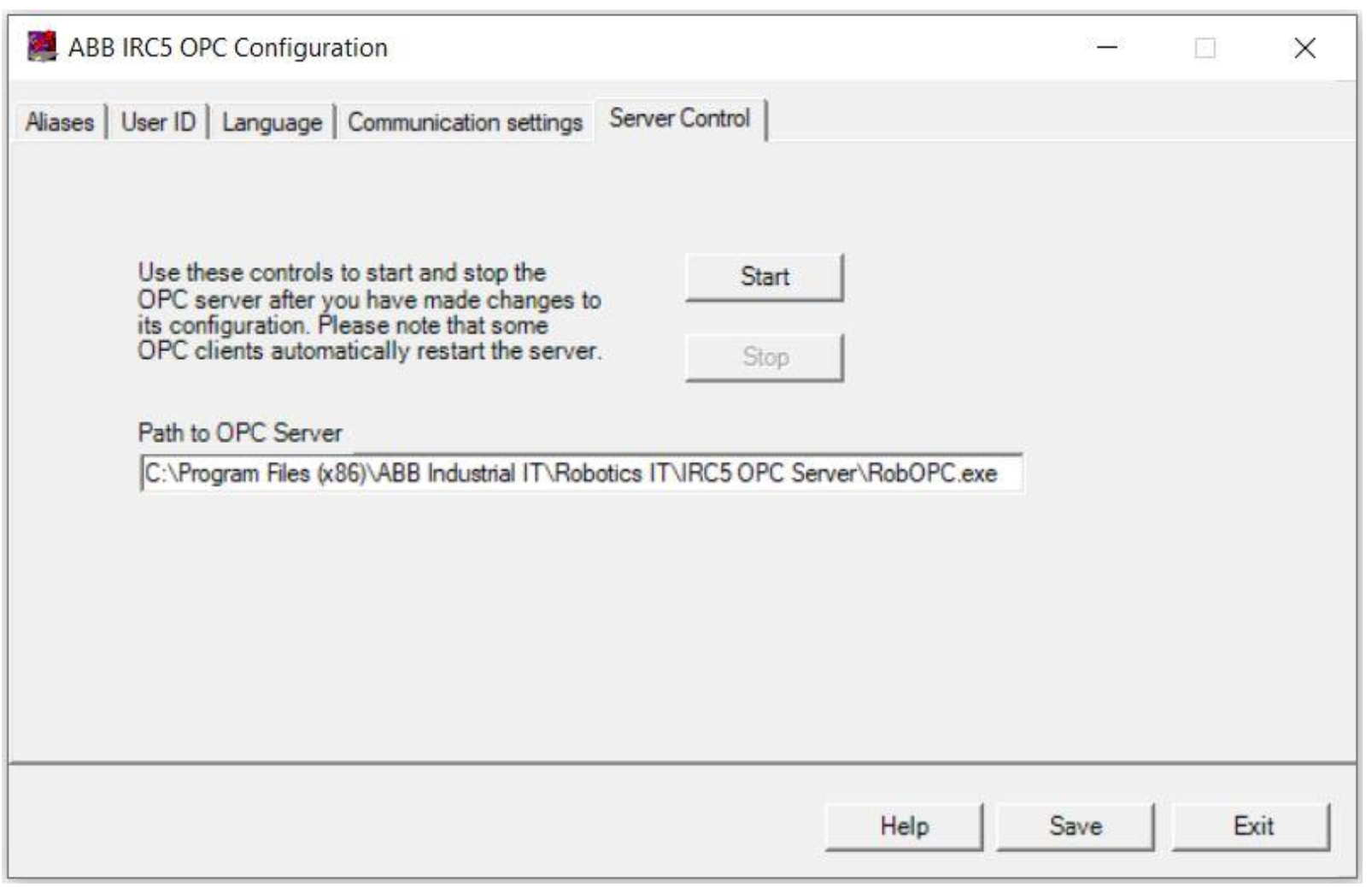
Next, the window below will open. Pressing the “Scan” button will allow the application to scan for available robotic drivers.



After scanning, select the correct robot controller from the list and select the markings as shown in “Connection Criteria”. Then press the “Create” button to create a new alias. Finally close the window.



In “Server Control” click the “Start” button and OPC server starts. After starting the server is up and the server can be stopped by pressing “Stop” button and restarting the server pressing the “Start” button again.



# CONNECTION FROM OPC SERVER TO DATABASE

In this project, the connection from the OPC server to the Postgre database happens with custom program made with Python coding language.

## Install Python and the required libraries

1. Install 32-bit python 3.6.x

<https://www.python.org/ftp/python/3.6.4/python-3.6.4.exe>

2. Install pywin32 -> for python 3.6 32-bit

pip install pywin32==224

If not working then: pip3 install pywin32==224

If not working then: py -m pip install pywin32==224

or

<https://github.com/mhammond/pywin32/releases>

-->pywin32-224.win32-py3.6.exe

(direct link: <https://github.com/mhammond/pywin32/releases/download/b224/pywin32-224.win32-py3.6.exe>)

3. Install DLL library

<http://www.gray-box.net/download_daawrapper.php>

(direct link:

<http://www.gray-box.net/send_daawrapper.php>)

graybox\_opc\_automation\_wrapper.zip -> x86 version -> gbda\_aut.dll

COMMAND LINE (AS ADMIN): (in directory where the file is located) regsvr32 gbda\_aut.dll

4. Install OpenOPC

pip3 install OpenOPC-Python3x

(in command line)

If not working then: pip install OpenOPC-Python3x

If not working then: py -m pip install OpenOPC-Python3x

## Database configuration (How it’s done in Python, except 1.)

1. Install neccessary libraries

pip3 install psycopg2

If not working then: pip install psycopg2

If not working then: py -m pip install psycopg2

1. importing psycopg2

import psycopg2

For example:

conn = psycopg2.connect(database="postgres", user="postgres", password="datasoft123", host="127.0.0.1", port="5432")

Used in this project:

conn = psycopg2.connect(database="test\_db", user="postgres", password="Robotti123", host="172.16.4.134", port="5432")

cur = conn.cursor()

1. Creating table to Postgre (Only if table not exist, “robottidata” is name of the table here)

create\_table\_query = '''CREATE TABLE public.robottidata

(value double precision,

name text COLLATE pg\_catalog."default",

quality text COLLATE pg\_catalog."default",

"time" timestamp without time zone); '''

cursor.execute(create\_table\_query)

conn.commit()

1. Insert (Only for testing)

sql\_test=f"INSERT INTO robottidata (name, value, quality, time) VALUES ('test', 1.00, 'yup\_test', '2020-01-01 00:00:00')"

cur.execute(sql\_test)

conn.commit()

## Code execution (In Python 3.6)

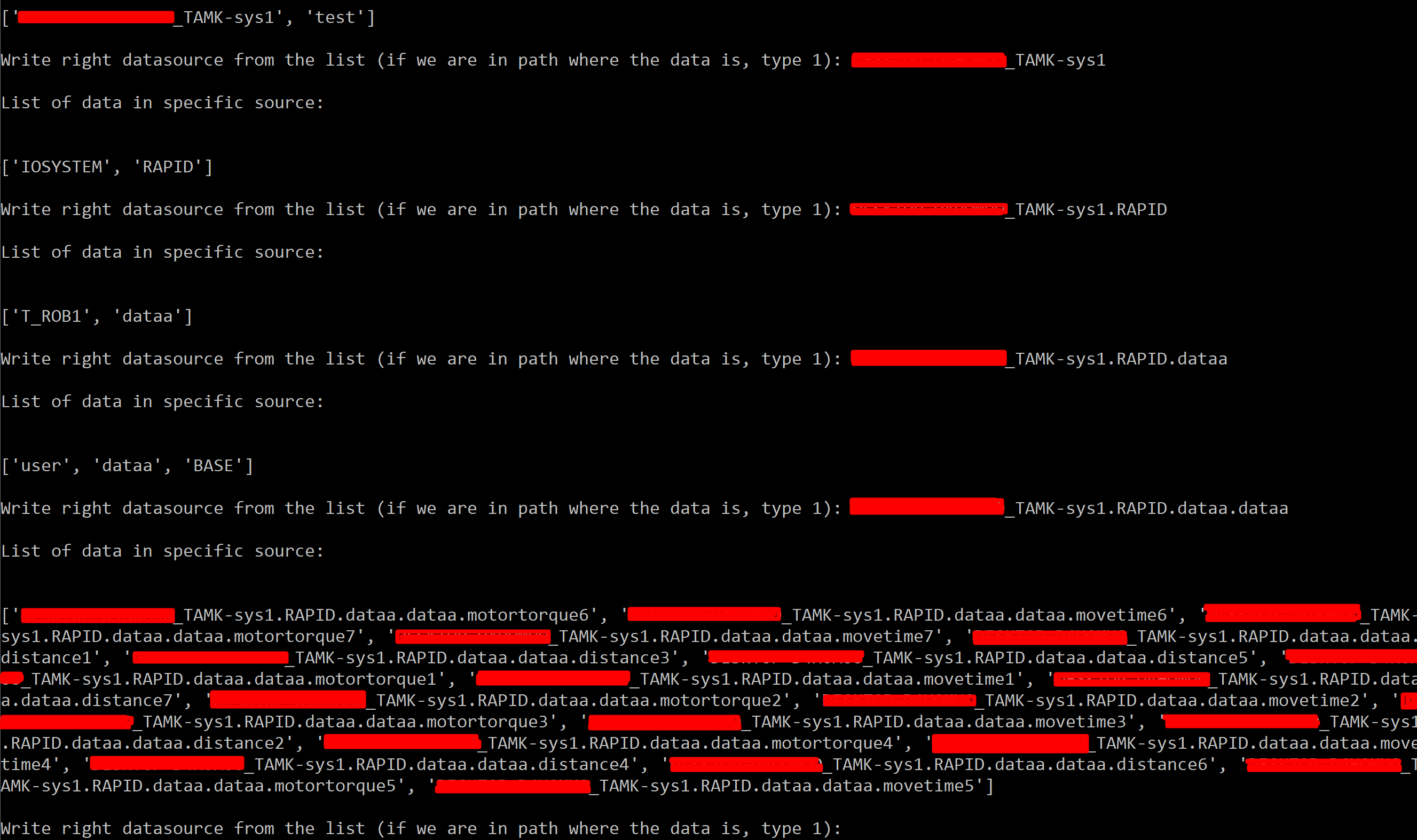
You can run python code with an automatic scripting program or run it step by step (recommended automatic code, if it not working then step by step).

### Code execution with automatic code

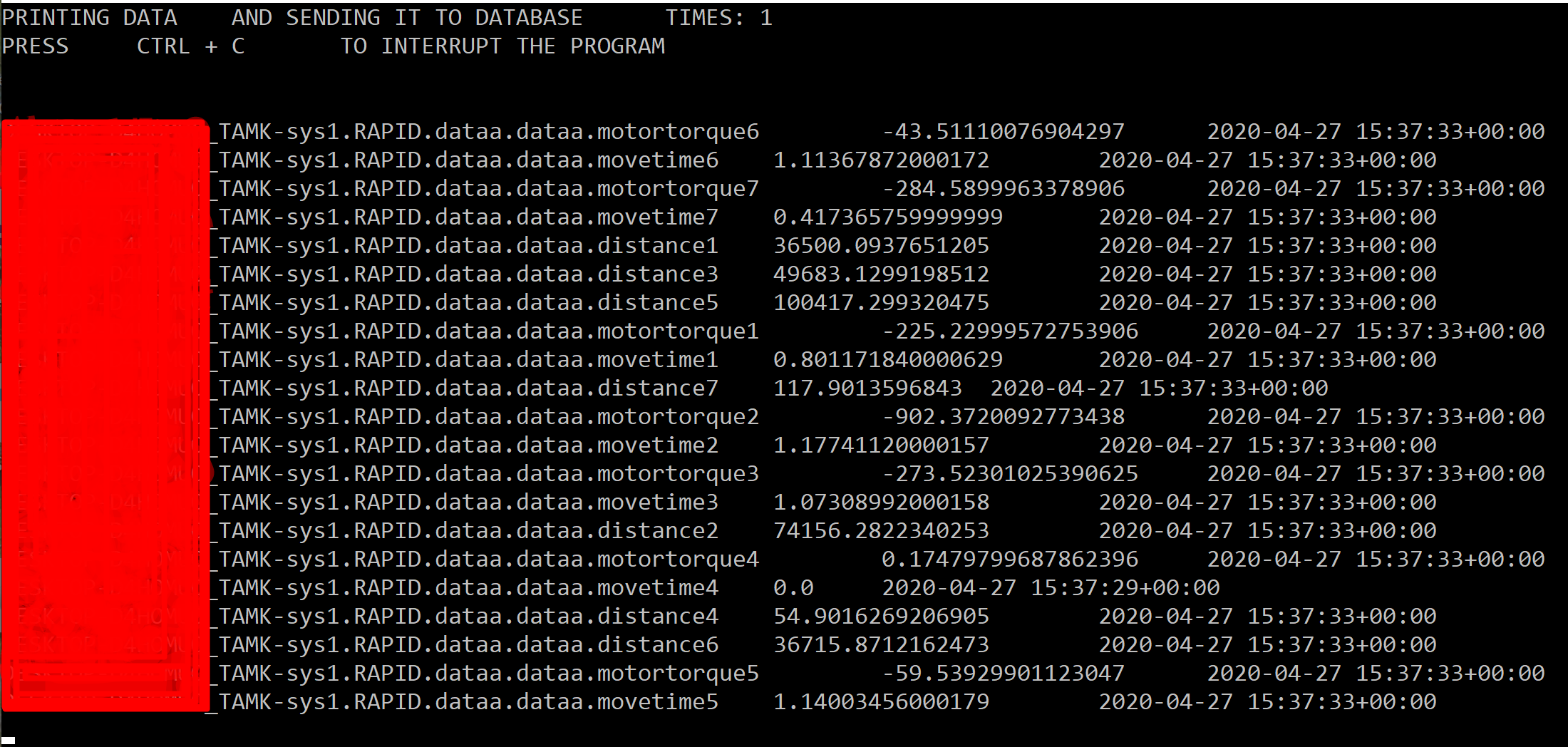
1. Open file (postgres\_python\_opc-exe.py) in text editor and

configure correct host, database, tablename, user, password and opc-server in there.

1. Then you can run file (postgres\_python\_opc-exe.py) in Python. You have to find correct path to datasource like next figures.



1. Now the data should flow from the opc server to the database



### Code execution one by one

1. Do the following steps **one by one**:

import OpenOPC

import time

import pywintypes

import datetime

pywintypes.datetime = pywintypes.TimeType

opc=OpenOPC.client()

opc.servers()

1. Pick the right server from the showed list:

opc.connect('SERVER\_NAME')

This project: opc.connect(' ABB.IRC5.OPC.Server.DA ')

1. Pick the right robot from the list:

opc.list()

opc.list('ROBOT\_NAME')

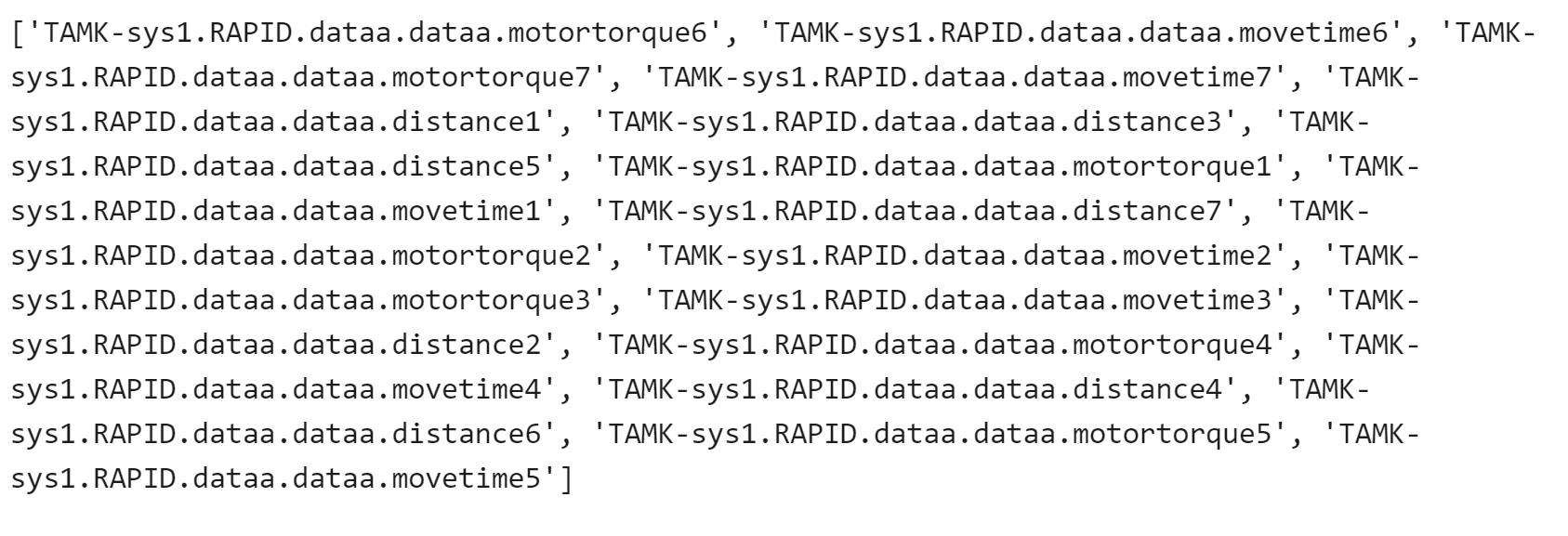
This project: opc.list('TAMK-sys1')

1. Next branch of the path is separeted by a point (aka subdirectory)

opc.list('ROBOT\_NAME.SUBDIRECTORY.ETC')

This project: opc.list('TAMK-sys1.RAPID.dataa.dataa')

Program output:



1. Next group this data like this:

tags = ['data','data1','data2']

This project data is group like that:

tags = ['TAMK-sys1.RAPID.dataa.dataa.motortorque6', 'TAMK-sys1.RAPID.dataa.dataa.movetime6', 'TAMK-sys1.RAPID.dataa.dataa.motortorque7', 'TAMK-sys1.RAPID.dataa.dataa.movetime7', 'TAMK-sys1.RAPID.dataa.dataa.distance1', 'TAMK-sys1.RAPID.dataa.dataa.distance3', 'TAMK-sys1.RAPID.dataa.dataa.distance5', 'TAMK-sys1.RAPID.dataa.dataa.motortorque1', 'TAMK-sys1.RAPID.dataa.dataa.movetime1', 'TAMK-sys1.RAPID.dataa.dataa.distance7', 'TAMK-sys1.RAPID.dataa.dataa.motortorque2', 'TAMK-sys1.RAPID.dataa.dataa.movetime2', 'TAMK-sys1.RAPID.dataa.dataa.motortorque3', 'TAMK-sys1.RAPID.dataa.dataa.movetime3', 'TAMK-sys1.RAPID.dataa.dataa.distance2', 'TAMK-sys1.RAPID.dataa.dataa.motortorque4', 'TAMK-sys1.RAPID.dataa.dataa.movetime4', 'TAMK-sys1.RAPID.dataa.dataa.distance4', 'TAMK-sys1.RAPID.dataa.dataa.distance6', 'TAMK-sys1.RAPID.dataa.dataa.motortorque5', 'TAMK-sys1.RAPID.dataa.dataa.movetime5']

1. Now we can read data

opc.read(tags, group='test')

opc.read(group='test')

## For – Loop code for data sending to Postgres

while True:

try:

# value = opc.read(tags,group='test',update=1)

# print (value)

for \_name, \_value, \_quality, \_time in opc.iread( group='test'):

print(\_name, \_value, \_time)

new\_name = \_name.split(".")[-1]

new\_dt = \_time[:19]

datetime.datetime.isoformat(datetime.datetime.strptime(new\_dt, '%Y-%m-%d %H:%M:%S'))

sql=f"INSERT INTO robottidata (name, value, quality, time) VALUES ('{new\_name}', {\_value}, '{\_quality}', '{new\_dt}')"

cur.execute(sql, (new\_name, \_value, \_quality, new\_dt))

conn.commit()

except OpenOPC.TimeoutError:

print ("TimeoutError occured")

time.sleep(1)

## Data from OPC server

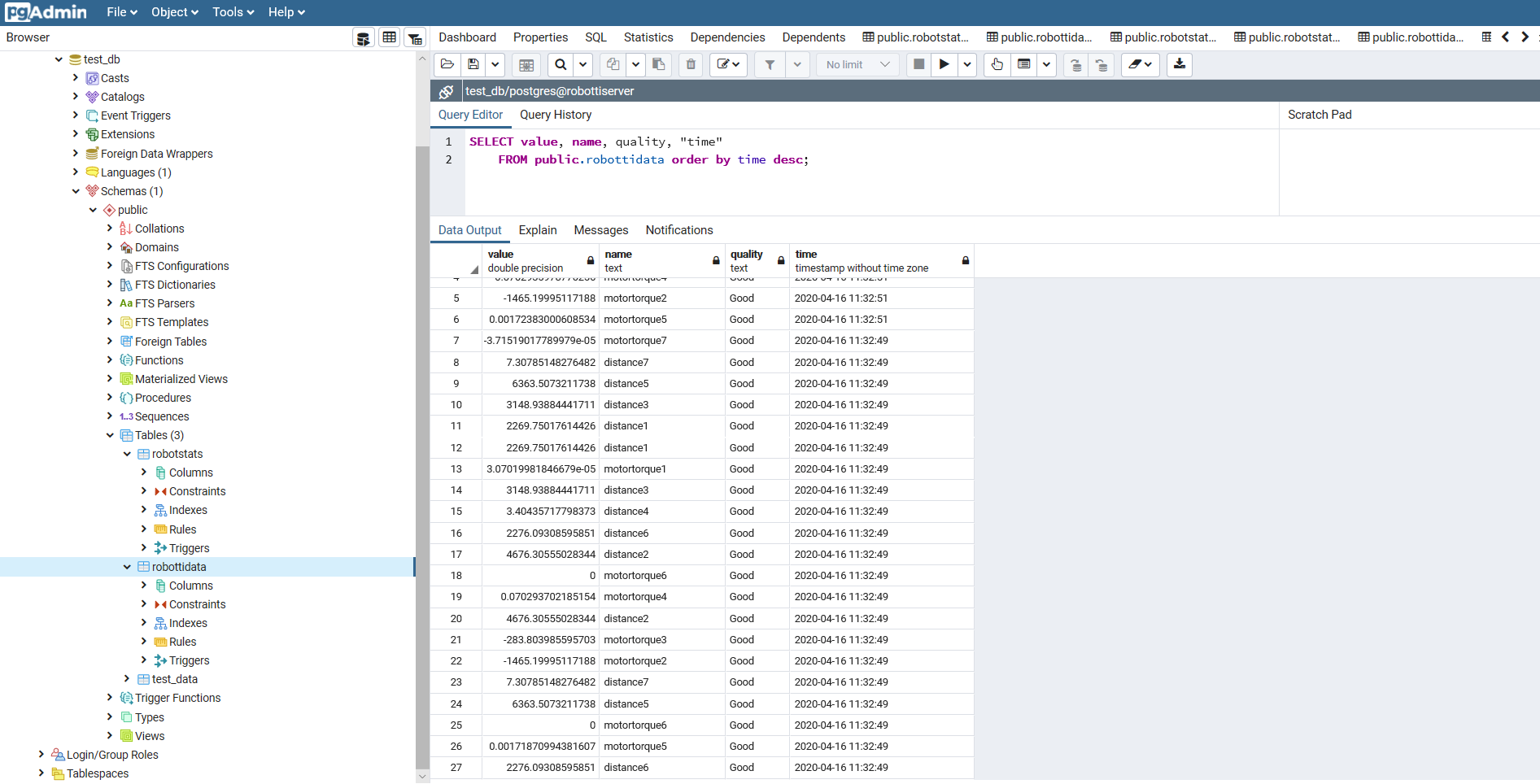


FIGURE 3. Data sent to the OPC server to database using Python

# DATABASE/SERVER

## Installing required software to the server

For all parts of this process connection to OpenVPN student-vpn-client is required**(OpenVPN setup link)**.

First set up a tamk linux virtual machine with debian. Install node, npm and postgress to the machine**(Postgress, Node/NPM links)**. Then download git repository from **(GIT link)** and then cd to project folder /nodejsProject and run npm install. After that running node server.js should work. You may need to make adjustments to db.config.js file depending on your postgres installation.

If server.js runs correctly, it should be able to get all data in the table using

curl -H "Accept: application/xml" -H "Content-Type: application/xml" -X GET <http://172.16.4.134:8080/api/robots>.

IP adress needs to be linux machines ip address.

In oder to connect the OPC server to Postgres, few postgres files have to be modified to allow access for outside connections to postgres. The line implicated by the arrow should be added to pg.hba.conf file.

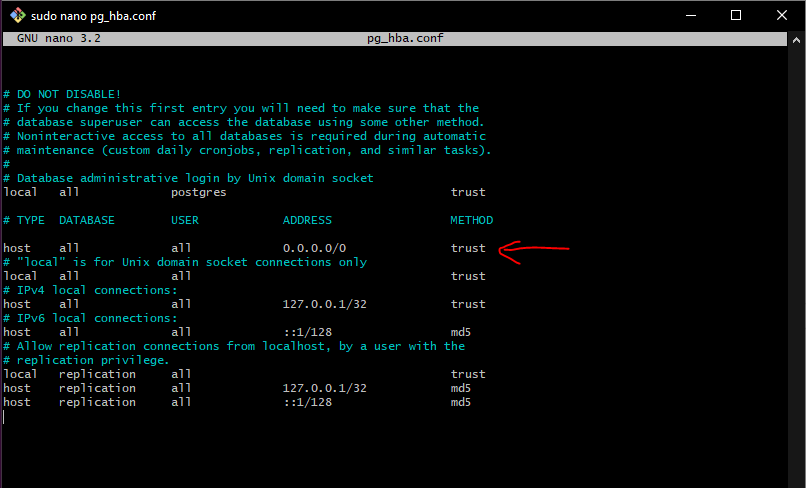


Figure 1. Postgres pg\_hba.conf file with changes

After that you should modify postgresql.conf file to match Figure 2. Take note of port number and listen address variables.

A screenshot of a cell phone

Description automatically generated

Figure 2. Postgres postgresql.congf file with changes

After changes have been made restart postgres.

LINKS

**OpenVPN:** https://gitlab.tamk.cloud/examples/gitlab-guide-wiki/-/wikis/student-vpn-fi

**Postgres:** <https://www.thegeekstuff.com/2009/04/linux-postgresql-install-and-configure-from-source/>

**Node / NPM:** https://linuxize.com/post/how-to-install-node-js-on-debian-10/

**Git repository:** https://gitlab.tamk.cloud/tamk-projects/robot-innovation-projecti -node-back folder

## Database setup

First a database needs to be created (<https://www.liquidweb.com/kb/creating-and-deleting-a-postgresql-database/>). The table for data is created when OPC-server python code connects to linux postgres database. More on that in chapter 2 of this document.

# FRONTEND

## Technologies used

Framework: React

<https://reactjs.org/>

Stylesheet used: SCSS.

<https://sass-lang.com/>

Coded with: VS Code – repository might contain config files for VS Code.

UI components built with material-ui library and layout done with CSS Grid.

<https://material-ui.com/>

<https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Grid_Layout>

<https://css-tricks.com/snippets/css/complete-guide-grid/>

Charts use the Victory library:

<https://formidable.com/open-source/victory/docs/victory-chart/>

<https://github.com/FormidableLabs/victory-chart/>

## Installation and running

Clone the project:

git clone <https://gitlab.tamk.cloud/tamk-projects/robot-innovation-project>

Navigate to front directory:

cd react-front

Install dependencies:

npm install

Run:

npm start

View project:

http://localhost:3000/

## Points to consider

We recommend Visual Studio Code for your code editor.

Tip: you can use Vs Code's terminal (opens with ctrl+ö or Ctrl+`) to run commands like the ones used in installation progress.

When working with styling it is recommended with reusable components and variables(colors) that they are created in react-front/styles/components or a folder equivalent to that to avoid repeating code.

Always remember when making a new global .scss file to add it to index.scss file: @import ‘new\_file.scss’.

When making new charts it is recommended that you add them in the MainPageCharts.js file into the chartsArray-array for automatical mapping. Using the Victory documentation, you can make and modify charts in many ways.

The code is written using React functional component with Hooks. Using functional components and hooks is not required, any kind of React works but it can be argued to be the future of React.

If you don’t have much experience in React, you can read React’s own site but a good way to learn is The Net Ninja’s YouTube playlist. There is React Context & Hooks Tutorial and a general React tutorial linked in the sources of this document.

# FURTHER DEVELOPMENT

## OPC Server and Python connection

NodeRed would seem like a good further development opportunity to replace Python. NodeRed was abandoned due to connection problems because our robot's OPC server uses the DA protocol.

## Database/server

Database could use more cURL functions, such as search by date/time with start and end times. Add functionality to select single class of data such as “motortorque 6” from database to front, so all data is not sent when asking for single data.

## Frontend

There are many ways to use Victory to display the data. Consider a method to display data that is not continuous, which can be most of the data, so that it only displays relevant values in an easy to read way. Activity data for example doesn’t seem to be readily available and you need to come up with a logic that displays the active time of the robot in question, if activity is a metric that you wish to display. That being said the demo charts are no means ready and might be implemented wrongly so I’d advice to dig deeper into the Victory library and look for examples online and read the official documents.

As stated in Database/server section, right now, the frontend can only receive all data as bulk. Backend needs to expose frontend a way to request a single set of data like motortorque6.

What our application lacks now is continuous subscription of data from database to frontend. Database gets new data as it comes from OPC. But frontend isn’t aware of new changes unless you add real-time subscription with web-socket.

Working with database people is essential in order to have the data stored in a way you think is easily fetched and used.

In short frontend needs real-time data subscription and the charts needs to be made compatible with incoming data.

# SOURCES

OPC DA | PYTHON | SERVER | CLIENT

<https://www.youtube.com/watch?v=lbubVMlPP6Y>

OpenOPC for Python

<http://openopc.sourceforge.net/>

Python ohjelmointikieli

<https://www.python.org/about/>

Python opiskelua (perusteet)

<https://www.learnpython.org/>

OPC DA Auto Wrapper

<http://gray-box.net/daawrapper.php?lang=en>

React Tutorial Playlist by The Net Ninja

<https://www.youtube.com/watch?v=yZ0f1Apb5CU&list=PL4cUxeGkcC9i0_2FF-WhtRIfIJ1lXlTZR>

React Context & Hook Tutorial

<https://www.youtube.com/watch?v=6RhOzQciVwI&list=PL4cUxeGkcC9hNokByJilPg5g9m2APUePI>

Websocket tutorial examples

<https://www.smashingmagazine.com/2019/03/realtime-charts-graphql-postgres/>

<https://www.graphql.college/graphql-subscriptions-with-react-node-apollo-and-postgres/>

Front and back gitlab repository

<https://gitlab.tamk.cloud/tamk-projects/robot-innovation-project>

ATTACHMENTS

A screenshot of a cell phone

Description automatically generated

Architectural diagram of project