

Final Year Project Report

K Cloud

IIOT customizable solution for data transfer in small to medium industrial control.

**SETU Waterford
Higher Diploma Computer Science**

Student: David Roche

Number: 93521243

Project Supervisor: Caroline Cahill

REVISION SHEET

Release No.	Date	Revision Description	Reviser
Rev 1.0	10-February-2024	Interim Report	D Roche
Rev 1.1	3-April-2024	Final Report	D Roche

EXTERNAL DOCUMENTATION

This document is best read with reference to the following external documents.

- GitHub Repository Master [Readme file](https://github.com/RocheDJ/kCloud/tree/main)
<https://github.com/RocheDJ/kCloud/tree/main>

DECLARATION OF AUTHENTICITY

I declare that the work which follows is my own, and that any quotations from any sources (e.g. books, journals, the internet) are clearly identified as such by the use of 'single quotation marks', for shorter excerpt and identified italics for longer quotations. All quotations and paraphrases are accompanied by (date, author) in the text and a fuller citation is the bibliography. I have not submitted the work represented in this report in any other course of study leading to an academic award.

Student..... Date

Workplace Mentor..... Date

TABLE OF CONTENTS

	<u>Page #</u>
<i>Revision Sheet</i>	2
<i>External Documentation</i>	2
<i>Declaration of Authenticity</i>	2
<i>Glossary Of Terms</i>	5
<i>Table Of Figures</i>	6
1.0 Introduction	8
1.1 Background	8
1.2 Generalized Customer requirements	9
Customer A:.....	9
Customer B:.....	9
Customer C:.....	9
Customer D:.....	9
1.3 Project Aims and Scope	10
2.0 Market Research	11
2.1 Terminology and Theory	11
2.2 Product research	13
3.0 System Overview	15
3.1 Application in brief	15
3.2 Real World Application	15
3.3 Detailed Overview	16
4.0 Design and Analysis	17
4.1 Data models	17
4.1.1 User Data Model	17
4.1.2 Process Variable Model or Object (PVO)	18
4.1.3 Process Data Model or Object (PDO)	19
4.1.3.1 Example Specific data for the PDO JSON	19
4.1.4 Control Data Model or Object (CDO)	20
4.1.5 Installation Model	20
4.2 Hardware	21
4.3 Software	22
4.3.1 New Technologies and platforms.....	23
4.3.2 Tools and IDE	24
4.4 Control Philosophy	25
4.4.1 Example inputs and outputs	25
4.4.2 Network Layout	26
4.5 Project Management	27
5.0 Functional design specification	28
5.1 Edge to Fog Development	28
5.1.1 Main	28

5.1.2 Local Webserver Process	29
5.1.3 API Web Client Process	31
5.1.4 API GET Client Process.....	31
5.1.5 Code Layout and Notes	32
5.2 Backend Development.....	34
The API implements JWT authentication, I tested this for several routes mainly within user management as well as PVO write.	37
5.2.1 Swagger UI	37
5.3 Front End Web Application	39
5.4 Native Android App	42
6.0 Issues and Resolutions.....	45
6.1 Fog Node Issues.....	45
6.1.1 Auto starting on Open WRT	45
6.1.2 Don't mention the firmware!.....	45
6.1.3 Communication between sub processes	45
6.2 Back end /Front End Issues	46
6.2.1 Node Versions	46
6.2.2 More Start stop Issues.	47
6.2.3 Frappe Charts Datasets.....	48
7.0 Reflection.....	49
7.1 Project Goals achieved and not.	49
7.2. What I learned.	50
7.2.1 A very brief word on AI.....	50
7.3 Future Development Work.	50
Bibliography.....	52
Appendix A – kEdge Program Flow chart.....	55
Appendix B- Ethical Approval Checklist	56
Appendix C- Open API documentation for backend	61
Appendix D- Database ER diagram	62
Appendix E-Technologies Used and end result.	63

GLOSSARY OF TERMS

Term	Description
RAD	Rapid Application Development
SCADA	Supervisory Control and Data Acquisition
SME	Small to Medium enterprise
JSON	JavaScript Object Notation Light wight self-describing, human readable data exchange format.
ERP	Enterprise resource planning (ERP) processes and systems that organizations use to manage day-to-day business activities including quality planning and maintenance.
AWS	Amazon web services
IT	Information Technology
OT	Operational Technology
CLI	Command Line Interface.
PLC	Programmable Logic Controller
IEC 61131-3	Worldwide recognized standard for programming and configuring industrial control devices. Part 3 outlines the basic software architecture and programming languages. Ladder diagram (LD)- graphical. Function block diagram (FBD)- graphical. Structured text (ST)- textual. Sequential function chart (SFC)-graphical.
IIOT	Industrial Internet of Things.
NDA	Non-disclosure Agreement.
UML	Unified Modeling Language
ST	A scripting language, one of the IEC 61131-3 languages.
ROI	Return on Investment.

TABLE OF FIGURES

Figure 1 ‘Top 10 Industrial trends 2023’ (Taparia, 2023)	11
Figure 2 Cloud v Fog (MOXA, 2019)	12
Figure 3 Initial Solution Proposal as submitted.	15
Figure 4 Python Routine that runs for one in 3 minutes.	15
Figure 5 Overview of Proposed Application	16
Figure 6 User Data Model.....	17
Figure 7 Process Variable Model.....	18
Figure 8 Process Data Model	19
Figure 9 Control Data Model	20
Figure 10 Installation model.	20
Figure 11 Software Technologies used.	22
Figure 12 Network and Sensor Layout	26
Figure 13 Screen shot from planning board in MS Teams, Feb 9th, 2024.	27
Figure 14 Signal Handler Class	28
Figure 15 Local web overview original wireframe.	29
Figure 16 Final Overview page.....	30
Figure 17 Fog Node Web server API calls.	30
Figure 18 SQLite PVO table showing data note Status field values 200.....	31
Figure 19 Example of a method that uses the Python Requests library to post information to the API backend.	31
Figure 20 Python Folder layout.	32
Figure 21 Main step type enum.....	33
Figure 22 A screen snip from early on in project development of handing JSON in SQLite.....	33
Figure 23 PVO JSON Object	35
Figure 24 Snip from PVO data table showing various PVO values.	36
Figure 25 SQL to Extract all PVO titles.	36
Figure 26 SQL query to extract average hourly temperature data.	36
Figure 27 API data base model showing SQL in use.....	37
Figure 28 API project Payout.	37
Figure 29 JS docs for PVO Post Swagger UI	38
Figure 30 Swagger UI for the API outlined in previous figure.....	38
Figure 31 Web App Overview page, original wire frame design.	39
Figure 32 Navigation widget for each Installation.....	39
Figure 33 Pasteurizer data in value element.	40
Figure 34 Power meter Data in Value element.	40
Figure 35 Date widget at the top.....	40
Figure 36 Web App Status page original wireframe design.	41
Figure 37 Web App Actual Screen with Trends.	41
Figure 38 Proposed Android App Screen	42
Figure 39 Android application Installation screen floating action button for NFC scanning.	43
Figure 40 Live PVO data.	43
Figure 41 Snippet of Code that Triggers a read of 32 bytes from address 0, when the on Tag discovered event is triggered.	44
Figure 42 Snip of Write Tag Routine.....	44
Figure 43 init.d config file for the pyio service	45
Figure 44 enabling the service.	45
Figure 45 Starting the service.	45
Figure 46 Webserver Process Pointer inside PyIo main file with shared memory space as argument.	46
Figure 47 Declaration of Shared memory variable.	46
Figure 48 Writing the Data to the Shared memory for the Web server.	46
Figure 49 Webserver Home page route handler.	46

Figure 50 What the user sees on the home page.	46
Figure 51 Installing latest node Version using NVM	47
Figure 52 using NVM to select latest version of node.....	47
Figure 53 API server starting in PM2	47
Figure 54 Web Server Start Script.	48
Figure 55 PM2 Status after running script.	48
Figure 56 Sveltekit Data set select solution.....	48
Figure 57 Realtime data on app and desktop.	49
Figure 58 Function block in use Codesys 3.5	51
Figure 59 Swagger UI landing page.	61
Figure 60Example of Swagger UI testing of Command Data Object (CDO) get.....	61

1.0 INTRODUCTION

The purpose of this document and project is to showcase the range of skills and technologies that I have learned and absorbed over the duration of the H Dip in computer science in SETU. As well as some other skills that I have gained through independent learning over the last two years.

1.1 Background

I graduated WIT with an Honours B-Tech in Electronic Engineering in 1997 and had not returned to third level education, until undertaking this course in 2021. For over twenty years I have been working with a small team of engineers in a company in Wexford providing automation control and instrumentation solutions to customers in a wide variety of areas ranging from SME, multinational and Semi State sector.

For that last number of years, the number of customers who want a more data focused element to their control systems has grown significantly. Also, the capability and sophistication of the Programmable Logic Controllers we use has also grown substantially. I believe it's useful for us to investigate other programming solutions outside of traditional IEC 61131-3 languages.

I believe there is a need for our company to be able to update our offering in this area to cater for existing customers and to help us attract new customers.

1.2 Generalized Customer requirements

To allow me to proceed with the project outside of an NDA I have put together several customer requests and generalized the various requirements. These requirements are as follows.

Customer A:

Have several small IO count process control machines:

They would like it if their customers could log into a portal to view the status of their machines mainly and start and stop the process remotely if possible.

“If they could do that on an app that would be great”. (Customer A)

They have, say for example, five key process values that they would like to be able to display for each of their machines. They would like to be able to log in and see all their machines’ current and previous values and to allow their customers to be able to login and view their machine/machines as well.

The data in this case will come from the control PLC on newer machines but they would like to be able to retrofit older machines.

Customer B:

Have an existing SCADA application to monitor their factory but would like to be able to chart and display certain KPIs live and historical on a dashboard to the wider group management outside that plant.

They would also like to be able to send E-Mail and SMS alerts to maintenance personnel in the event of certain alarms being raised and escalate those alarms in the event of the alarm not being acknowledged.

Communications from PLC/SCADA is over fixed broadband.

As stated, B have a large site, and would like to be able to add alarm messaging and tracing but for the moment are focused on power consumption in various areas of their factories, they would like to be able to log into and view power consumption totals and trends for a period ranging from hours to weeks.

Their data will come from an existing PLC/SCADA connection to power meters and IO.

Customer C:

Have several control systems already reporting into their existing web portal but would like to expand into monitoring new sites using a reporting solution that doesn’t require a full control system but still maintains the flexibility to use their existing portal.

Communications is over fixed broadband with a mix of mobile data.

They are keen to have the data uploaded in such a way as to make it available to their existing front-end developer.

Customer D:

Have a complex mobile control system that requires data to be sent back on a timed and event driven basis to a portal so that they can monitor machine location, the status of process consumables and any alarm events on the machine.

Communications is over mobile broadband.

They are looking to receive process events and reports from the system which will be of varying types.

1.3 Project Aims and Scope

The aim of this project is not to solve all customers' wishes or to provide the ultimate IIOT solution if one exists, but to put together the makings of a cost-effective EDGE to Cloud solution, that can be used as a solid foundation by us to offer our clients a RAD solution to meet and hopefully exceed most of their requirements.

Tasks to Accomplish.

- Installation or Site level: marshalling the data and buffering it for dispatch.
 - Reading the data from the Fieldbus/PLC/SCADA and sending it in a controlled manner to the Back end.
- The back-end Level: receiving and storing the data from each installation.
 - Acting as a go between the Front End and the Installations in the field.
- The Front-End Level: Providing the user interface that allows the user to Chart Plot and view the data from the Installation.

The project implements a common backend framework that allows the different installation types to exchange information in a defined way with it, and then to use a customizable front end based on customer requirements.

2.0 MARKET RESEARCH

When reviewing recent trends in control and automation engineering one of the recurring themes is the convergence of IT/OT. At a recent European industry showcase *SPS Drives* in Nuremberg Germany the top two industrial automation trends, according to research by IOT Analytics were Solutions that enable flexibility and IT/OT convergence. This was ahead of AI trends at five and six.

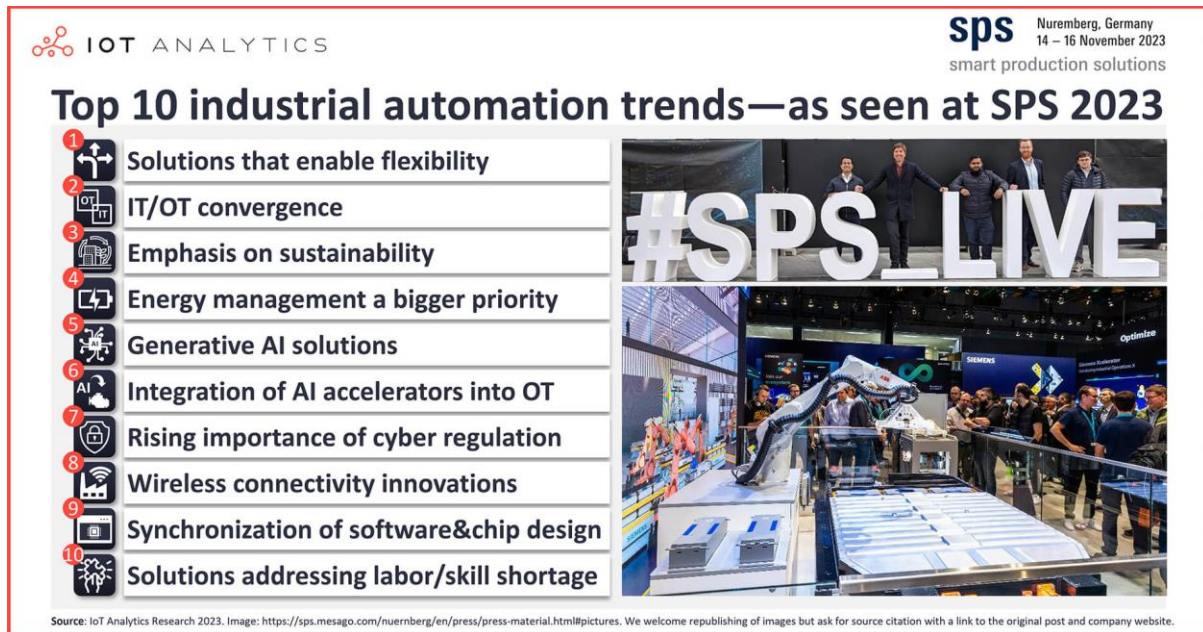


Figure 1 'Top 10 Industrial trends 2023' (Taparia, 2023)

2.1 Terminology and Theory

When starting down the IOT/IIOT development road it doesn't take long for some new terms and phrases to appear I have heard of the cloud ok but, FOG, EDGE thick and thin, new to me.

An article on TechTarget describes EDGE computing as '...a distributed information technology (IT) architecture in which client data is processed at the periphery of the network, as close to the originating source as possible.' (Bigelow, 2021)

My originating source in the kCloud project example is the pasteurizer.

The article goes on to say 'One of the easiest ways to understand the differences between edge, cloud, and fog computing is to highlight their common theme: All three concepts relate to distributed computing and focus on the physical deployment of compute and storage resources in relation to the data that is being produced. The difference is a matter of where those resources are located' (Bigelow, 2021)

In an article by Moxa a specialist in industrial conductivity they say 'Industry experts have warned that the cloud-computing models deployed in many IoT systems today are ill equipped to deal with the volume of data generated by the billions of IoT devices that are slated to go online in the next couple of years. The need for instant decision making coupled with concerns regarding data security has led the early adopters of the IoT to consider alternative computing models.' (MOXA, 2019)

This is where it all gets a little bit FOGGY, the FOG computing model is based around removing the need for vast quantities of data to be sent to the cloud for processing and remote decisions being made by closer to the edge with therefore reducing the data bandwidth required to the cloud.

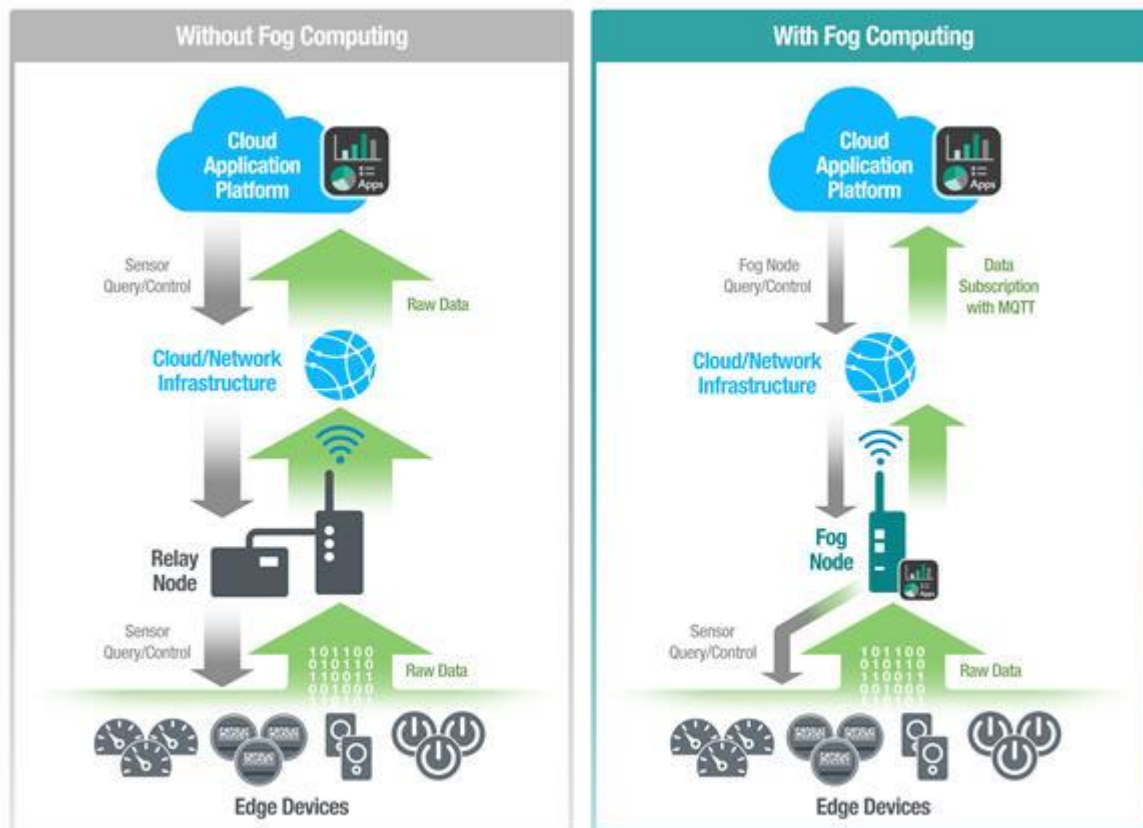


Figure 2 Cloud v Fog (MOXA, 2019)

There are also some drawbacks to putting all your eggs in the cloud basket, at least one of our customers has stated their concern with putting all the data in the cloud with the *what if I have no internet for a week and I need to run my equipment and get the data, can I?* This downside is also covered by Clive Longbottom in his article for Tech Target IoT agenda when he outlines the advantages and disadvantages of edge versus cloud computing concluding that,

‘..The trick is to implement a well-blended mix of an underlying cloud platform combined with the judicious use of edge computing to meet the organization's needs.’
(Longbottom, 2021)

He also goes on to further state ‘Herein lies the dichotomy: Trying to fully manage an IoT environment through a cloud-only platform isn't the optimal way of doing things.’
(Longbottom, 2021)

So, what I can conclude from this is that the idea of getting information into the cloud is an essential part of all IOT systems but when planning a solution, it is important to provide the correct solution not just the cloud solution.

2.2 Product research

Before development started, I took a closer look at some of the existing products in the marketplace and what they were offering. Having used the Raspberry Pi during the course I started by looking into the Pi as an industrial platform rather than the traditional PLC, this led me to a German company called **Kunbus**.

Kunbus is the company behind the “RevPi” P.L.C which uses the power of the Raspberry Pi and wraps it in an industrial PLC or in their words ‘Industrial Raspberry Pi for control and IIOT Projects’

(Kunbus gmbh, 2023)

They allow OEM manufactures to own brand the RevPi device for their specific application. Doing some further internet research into this led me to a company **CloudRail**.

‘CloudRail is a fully managed solution to acquire data from industrial environments, preprocess it locally and send it to any cloud.’

(CloudRail, 2023)

They use the RevPi to connect IO Link sensors to the cloud supporting AWS Azure IBM, SAP and more cloud providers.

Their ongoing cost starts at €24 per month per unit with a minimum of 2 units required plus, the cloud rail box hardware and IO link master with a cost of €787.10 and €418.580 list at the time of writing Jan 2024 all prices being ex VAT and discounts. (CloudRAIL GmbH, 2023)

Support and Development are in Germany there is no option to take the system and run locally on private server if required or to export data to an existing platform dashboard.

Another solution that appeared in my search windows was **Cumulocity** from another German provider.

‘Cumulocity IoT platform simplifies things for you with self-service tools and a configuration-driven approach. Cumulocity IoT is a leading self-service IoT platform, top rated by independent analysts, with fast ROI.’ (Software AG , 2024)

This is a large platform concentrating on creating a code free solution, it does offer a lot in terms of getting data from the EDGE to the center of the cloud and has a large GitHub repository with “example applications created using Cumulocity SDK”. (Software AG, 2024)

There was a limited trial version but no public pricing information available, they did offer On-premises or dedicated cloud deployment for the enterprise-based clients.

Another company that came up was **Trendlog** they are a Danish company that supply a product that captures information from machines and either displays it in local real time or transmits it to the cloud for further processing.

‘When you need to gather information from production machinery, robots, or sensors, we have models for both the simple and more advanced installations.’ (Trendlog, 2024)

What caught my eye was their TL Collect UNO product which is a standalone data collector that can transmit data from up to eight machines. This data is then sent to a Cloud API to be processed into trend dashboards, maintenance planners and other web data tools.

The final product I investigated is from **IFM** electronics and is called **Moneo**, they have developed a very nice back-end portal and hardware interface for their existing range of sensors. To quote from their website ‘As an IIoT platform, ifm moneo combines the level of operation technology with the level of information technology. The sensor data generated in

the production plants can be read and processed easily and used as a basis for sustainable corporate decisions.’

(IFM Electronic, 2024)

Support and development are available in Ireland but there is limited self-hosting support and integration into existing SCADA systems with an emphasis on larger ERP systems.

What I found is that there are products out there that could meet some of our customers’ requirements, from my investigations a lot of them require connecting to a bespoke portal or buying bespoke hardware and most of the solutions technical support is outside of Ireland. I believe that incorporating a flexible edge to cloud solution can greatly enhance the service we offer our existing customers and help us compete in the marketplace.

3.0 SYSTEM OVERVIEW

The project implements an example project that will cover as many of the generic customer requirements as possible.

3.1 Application in brief

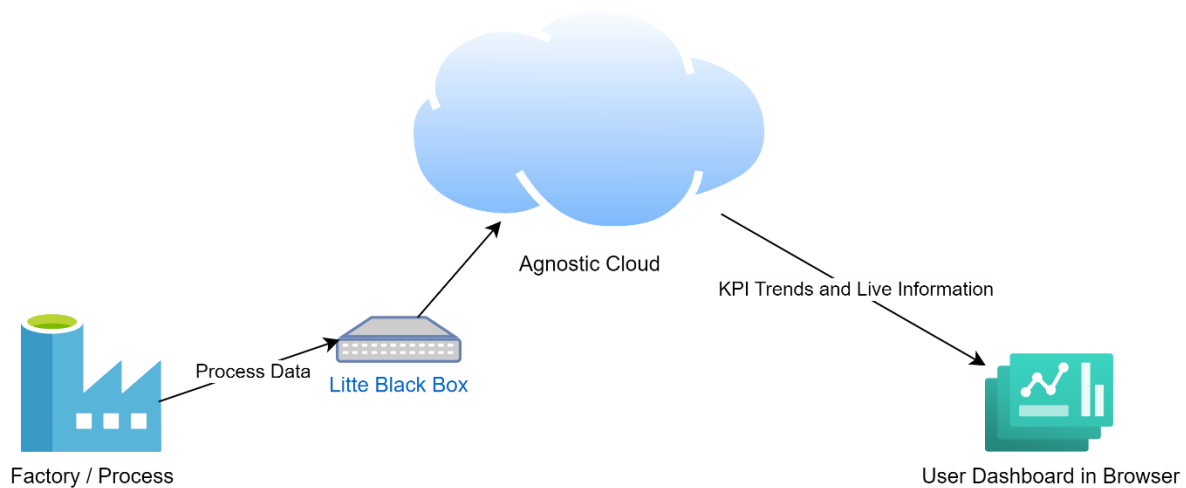


Figure 3 Initial Solution Proposal as submitted.

3.2 Real World Application

To better visualise the solution, I use the example of a batch pasteurizer i.e. a machine that will heat a batch of liquid to a setpoint temperature hold it there for a predetermined amount of time and then release it.

An agitator for mixing the liquid run for 1 minute every 3 during the cycle and can be manually run at any time outside of the pasteurization sequence.

```

14 # -----
15 # how many minutes have we been running
16 # agitator runs one minute every 3 minutes
17 # so if the total minutes ran is devisable by 3 without a remainder
18 # run the agitator
19 def Calc_OneInThree(dStartTime):
20     now = datetime.now()
21     dtDiff = now - dStartTime
22     iRuntime = dtDiff.total_seconds()
23     minutes = divmod(iRuntime, 60)
24     iRemainder = divmod(minutes[0], 3) # 3*60
25     if iRemainder[1] == 0.0:
26         return True
27     return False
28 # -----

```

Figure 4 Python Routine that runs for one in 3 minutes.

Our application records the temperature and level in the tank, the state of the agitator and heater.

The user will be able to open a local web page hosted on a FOG Node, which will give them a summary of the status.

They are also able to log on to a cloud-based portal to access status and trend information for the process and allow them to start and stop the machine.

3.3 Detailed Overview

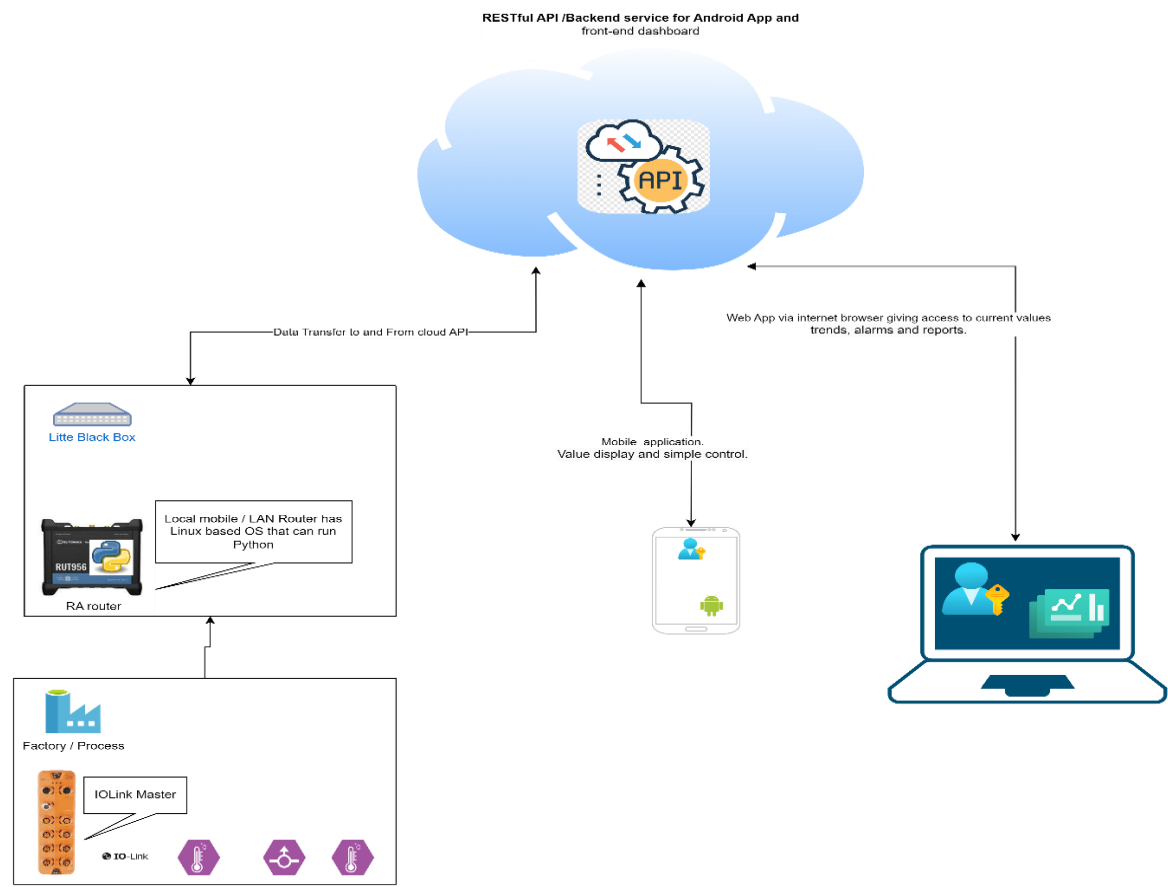


Figure 5 Overview of Proposed Application

4.0 DESIGN AND ANALYSIS.

The key element of this project is data, so I started my design process by looking at the data models we are going to need.

4.1 Data models

All customers will require some type of user data for logging in and account verification.

- **User Data** –
 - Logon Information: User Identifier, Username, Email Address, password etc.

From talking to customers the information, they are looking for can be broken broadly into two types

- **Process Values** –
 - Numeric values that are assigned to a particular sensor or single measurement.
 - E.g., Temperature, Pressure, Volume Power Consumption etc.
- **Data Values** –
 - Data that is a summary, report and event or a log of some kind a structure in and of itself.
 - E.g., Job cards, Alarm Events,

4.1.1 User Data Model

The following UML model shows the user data model as envisaged allowing for remote client connection types of administrators at group and global level in the future.

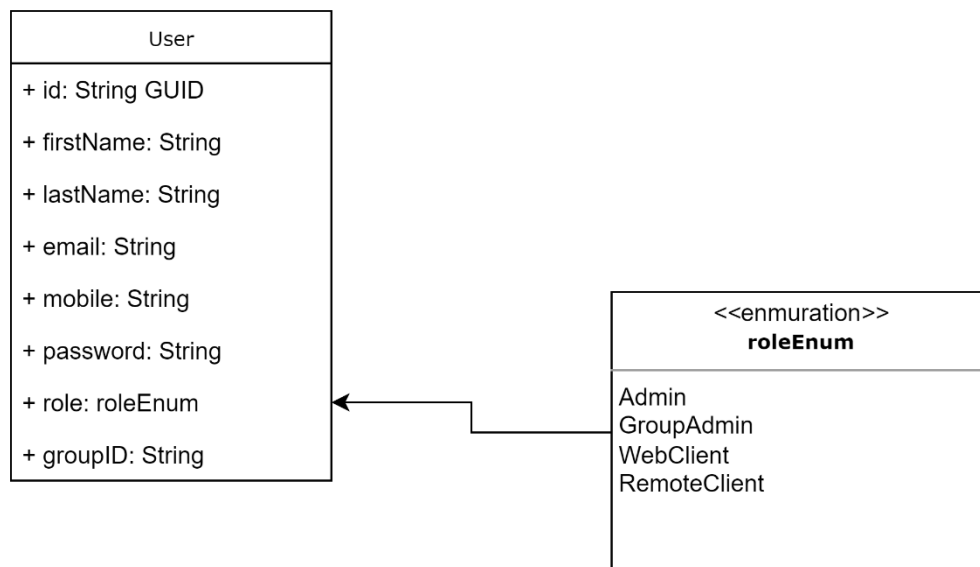


Figure 6 User Data Model

4.1.2 Process Variable Model or Object (PVO)

The process variable model is used to describe a single value or KPI, at a given point in time.

Process Variable Model

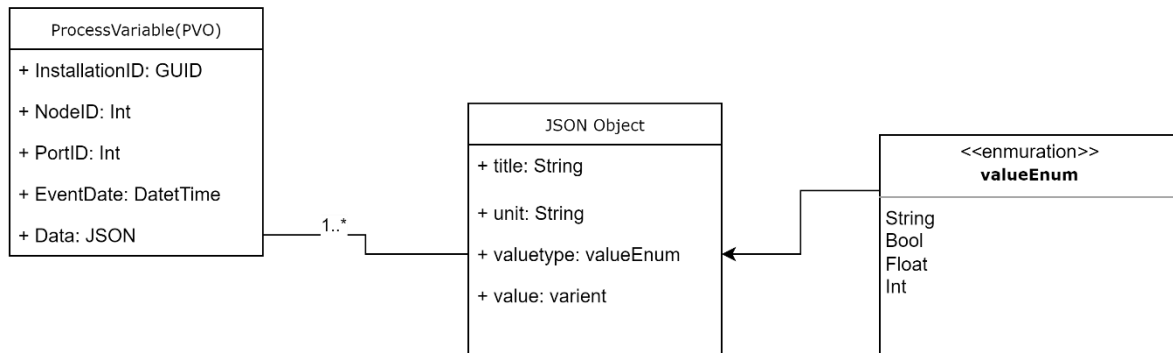


Figure 7 Process Variable Model

The PortID, is the port id on the IO link master e.g.1 to 4 for the AL1350, NodeID is the last octet of the Ip-address of the IO-link master or if no master is present, it can be the last octet of the router ip-address.

4.1.3 Process Data Model or Object (PDO)

The process data model is used to record event data such as report information or alarms at a given point in time, e.g., process starts, or end reports, operator entered information or event data.

Process Data Model

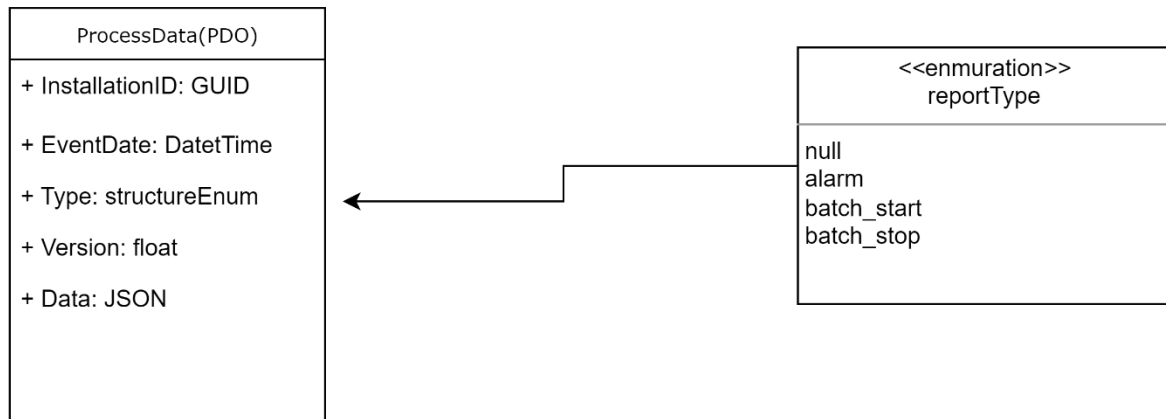


Figure 8 Process Data Model

There are two PDO, data structures implemented in the batch pasteurization example.

4.1.3.1 Example Specific data for the PDO JSON

This will vary depending on the specific application in this case we have a pasteurizer.

Batch Start

The batch start JSON consists of the following information.

Batch Start	: UTC time of Batch Start.
Start Temperature	: Temperature sensor value at the start.
Start Volume	: Level sensor scaled volume value at the start.
Hold Temperature	: Temperature Batch must reach.
Hold Duration	: Time we hold temperature for.

Batch Stop

The batch start JSON consists of the following information.

Batch Start	: UTC time of Batch Start.
Hold Start	: UTC time of Hold Start (hold is when setpoint temperature is reached).
Batch Stop	: UTC time of Batch Stop.
Stop Temperature	: Temperature sensor value at the start.
Stop Volume	: Level sensor scaled volume value at the start.
Batch Duration	: Time Batch was pasteurised for.
Batch Code	: 0=Pass, 1=Stopped by User, 2=Failed to heat, 3=other Fail.

4.1.4 Control Data Model or Object (CDO)

This data model is used to send control information from the back end to the EDGE/FOG device.

Control Data Model

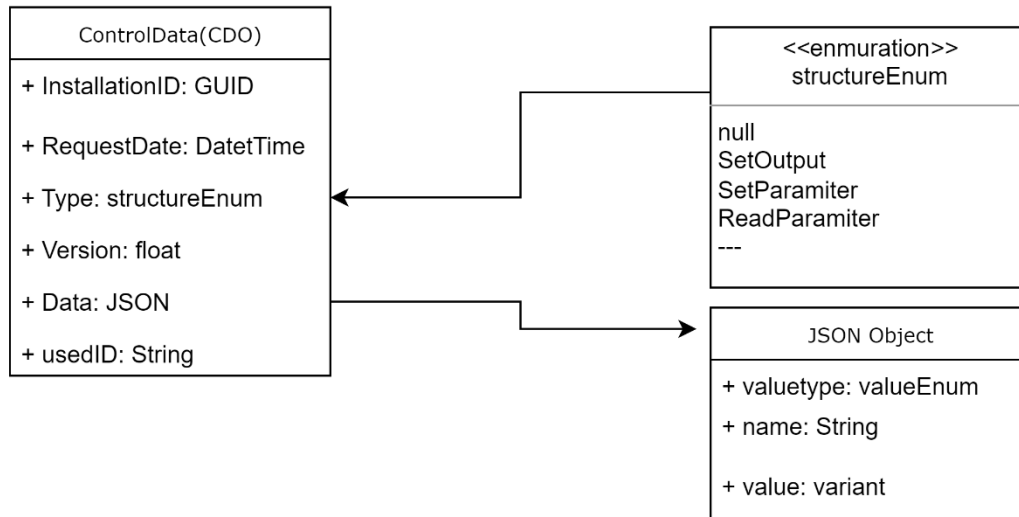


Figure 9 Control Data Model

In the JSON object, value is determined by the value enumeration, and will be either String Boolean, Float, or Integer.

4.1.5 Installation Model

This is used to hold the details of each site as follows,

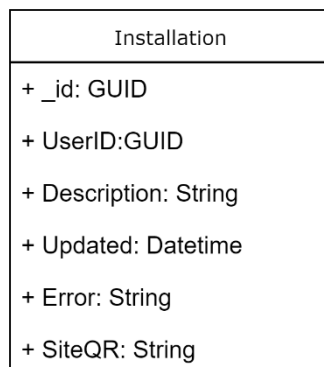


Figure 10 Installation model.

4.2 Hardware.

This project utilizes the hardware currently available to me. From the customer requirements and market research we will use an IO-Link connection for sensors and field equipment.

‘IO-Link is a serial, bi-directional point-to-point connection for signal transmission and energy supply under any networks, fieldbuses, or backplane buses.’

(IO-Link Community, 2018)

- IO-Link Master
 - For testing purposes, I have sourced a 4 port IO link Master from IFM electronics an AL1350 ‘For connection of up to four IO-Link devices’.... Reliable transmission of machine data, process parameters, and diagnostic data to the IT environment’.
(IFM Electronic, 2023)
- 4G Industrial Router
 - I will use a Teltonika RUT 956 Industrial cellular router which has a Linux based OS (Open Wrt) than can run scripting languages such as Python. It is ‘a compact industrial 4G (LTE) router equipped with 4x Ethernet ports, WiFi, Dual-SIM, GPS, an I/O connector block, RS232/RS485 and RutOS software for advanced networking solutions.’
(Teltonika, 2024)
- Sensors
 - I will be using several sensors to give a range of value types all these sensors will be IO link enabled, values measured consist of temperature, conductivity, level, and pressure.

In terms of the hardware, I have used most of it before but prior to doing this course had only ever used discrete and hardwired version of the sensors and never the IO-Link functionality for a customer product. The router itself was only ever used as a router and was most certainly never seen as a possible process controller..... Until now.

4.3 Software.

- The following diagram graphically illustrates the software technologies used.

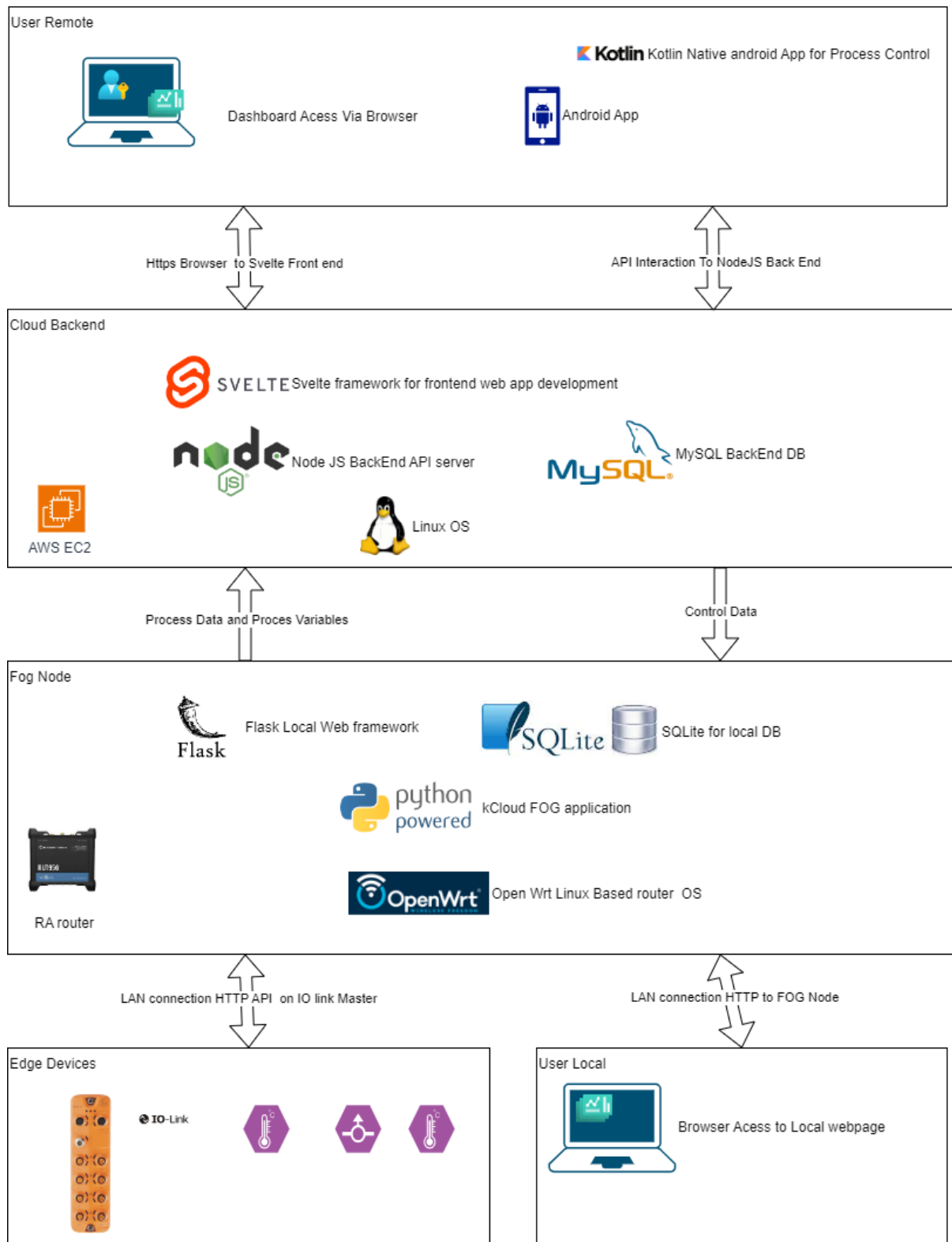


Figure 11 Software Technologies used.

A diagram of the completed software with reference to the above planned can be found in Appendix E

4.3.1 New Technologies and platforms.

This project builds on several new technologies and greatly expands on the knowledge gained during the course.

Using Python on the fog node to do the control is a complete break from what would normally be done in our company, but it is a great opportunity to assess its potential for process control and for me to develop further my proficiency in it as a development language. The following table outlines some of the technologies used in the project that are either completely new to me or are used in a different way.

Name	Previous course use	Use in this application
OpenWRT	None	Reduced lightweight Linux based OS used on the RUT95x routers.
Python	Introduced to me during the course and used for on some assignments in IOT and DevOps	Greatly expanded my knowledge here, running as a process / subprocess on OpenWrt. As well as the use of User defined types enumerated lists.
Flask	Introduced to me during the course and used for on one assignment in IOT	A good fit in this application because of it's light footprint and flexibility. Used here for multiple pages and API endpoints hosted on the router.
SQLite	None	This is the backbone database of the local Fog Node. The project uses the sqlite3 library for python which is a 'DB-API 2.0 (PEP 249) compliant interface to the SQLite library' (python.org, 2024).
AWS EC2	Introduced to me during the course and used for on some assignments DevOps	I chose to use Ubuntu Linux as the platform here running on an Amazon EC2 instance rather than using AWS Linux as we did on the course. This will allow me to use a stand-alone ubuntu server in the future if required.
Node JS/Express	Used Node JS and HAPI during Full Stack development use with express as API server is new.	Full setup of Node JS on Ubuntu, development of API backend and DB integration.
Mari-aDB/MySQL	My SQL introduced during Databases module. Maria DB	This project was to originally use MariaDB which 'Created by MySQL's original developers, MariaDB is compatible with MySQL and Oracle, and is guaranteed to always be open source.....' (MariaDB, 2024). However, I found during initial testing on Linux that the JSON capabilities of MySQL 8.0 worked better so I used MySQL8 as the backbone data base of the backend server.
Svelte	Used during Full Stack development	On this project I am using the Svelte framework to develop the foundations of a full featured dashboard for the project. The user interface on the front end while simple to use, uses accordion plugins, Check box controls and drop-down lists that I have not used before in this framework.

Kotlin	Used during Mobile development	To make the overall project more saleable to customers I have included some modifications to the Kotlin based application developed for 'Mobile applications Assignment ' (Roche, 2024) I modified the previous application to include an NFC read interface to select /check in at the instillation. As well as modifying the UI for this application and converting the app to use the kKloud API back end.
--------	--------------------------------	--

Table 1 Software technologies used.

One of the key elements of this project I have not investigated before is the inclusion of a customizable JSON object as a JSON type column in the SQL tables.

According to Dave Stokes on open source .com 'MySQL's addition of a JSON data type makes the relational database easier to use and blurs the lines between SQL and NoSQL databases.' (Stokes, 2017) turns out he is correct.

It allowed me to develop a fixed upload framework around a flexible loose formatted JSON NoSQL data type.

4.3.2 Tools and IDE

The following tools are used in developing this project.

Name	Function	Used Before	Notes
VS Code	IDE for Python and JS development	Yes	Ended up using for all development.
Postman	Used to test API local and remote	Yes	I used this at the start but once the Swagger UI was implemented not as much.
Win SCP	Used in transferring app files to RUT95X	No	Verry useful tool in the end I used it to deploy to the Development Server and local router.
DB Browser for SQLite	Used to inspect and manipulate data in SQLite	No	Used at the start and during the project as a quick simple tool to manipulate the SQLite DB.
DBeaver	DB browser and tool for Linux /windows	No	Comprehensive lineup of tools and great data visualization but has a steep learning curve.
HeidiSQL	DB browser and tool for Windows	Yes	Not as advanced as DBeaver but covers what I need good visualization of SQL queries.
CLI MySQL	The only way to setup the data bases at the start and configure them.	No	I prefer the GUI tools but luckily there are lots of online tutorials and instructions, so job was not too hard in the end.

Table 2 List of development tools used.

4.4 Control Philosophy.

As stated in section 3.2 I am using a real-world example of a pasteurizer to develop the solution.

The following section shows the inputs and outputs for the application.

4.4.1 Example inputs and outputs

Inputs				
Index	Description	Connection	Protocol	PVO Index
AI_00	Level	IO master Node 1Port 1	Io Link	0
AI_01	Conductivity	IO master Node 1 Port 2	Io Link	1
AI_02	Temperature	IO master Node 1 Port 2	Io Link	2
AI_03	Volume	IO master Node 1 Port 2	Io Link	4
DI_00	Agitator Active	IO master Node 1 Port 3	Io Link	3

Table 3 Inputs

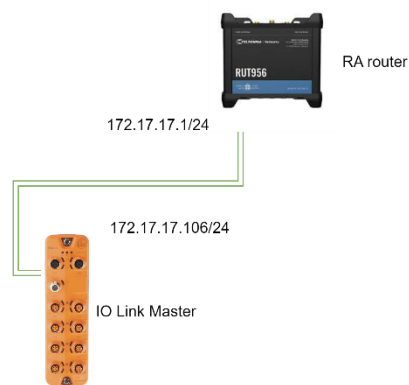
Outputs			
Index	Description	Connection	Protocol
DQ_00	Agitator	IO master node 1Port 3	Io Link
DQ_01	Heater	IO master Node 1 Port 4	Io Link

Table 4 Outputs

Soft Inputs (Key Presses)	
Index	Description
SDI_00	Start Process
SDI_01	Stop Process
SDI_02	Start Agitator
SDI_03	Stop Agitator

Table 5 Software /Trigger inputs

4.4.2 Network Layout



Port 1 (input)

LR2750 Level Sensor

Port 2 (input)

LDL 100 Temperature and Conductivity

Port 3 (Output)

DQ0 Agitator

Port 4 (Output)

DQ1 Heater

Figure 12 Network and Sensor Layout

4.5 Project Management

This section outlines the sprints involved in putting the project together. To help manage the project I used MS Teams, I setup a separate channel for the project and added a planner tab to display the tasks involved. I chose teams for several reasons, although my company don't use it to manage projects this is a good opportunity to try it out determine suitability.

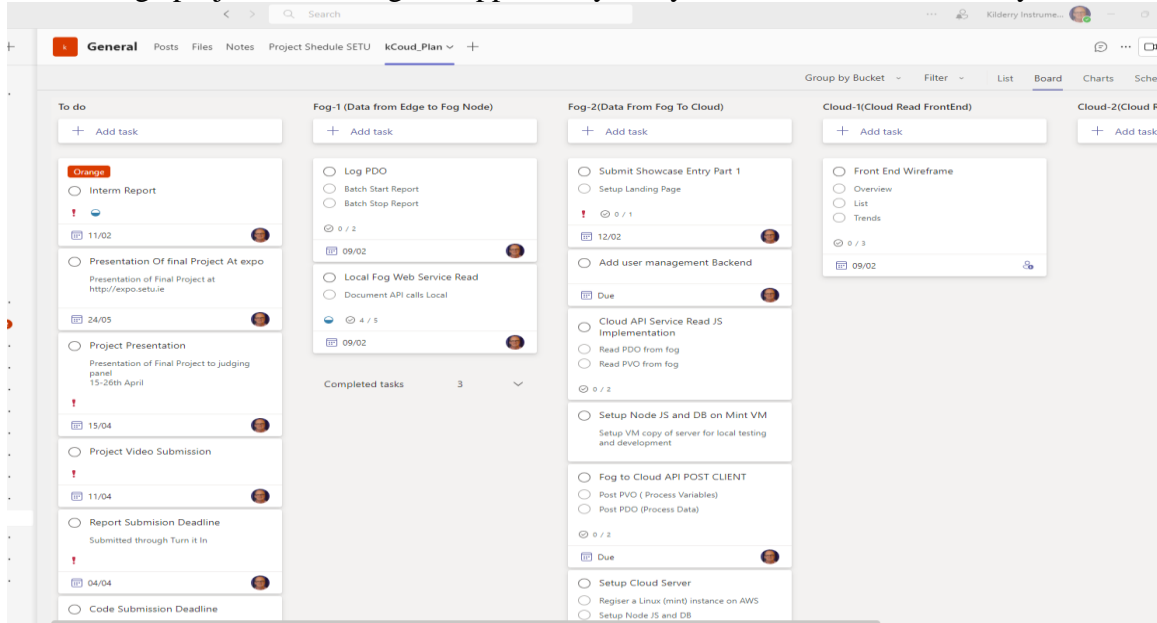


Figure 13 Screen shot from planning board in MS Teams, Feb 9th, 2024.

A ramp phase for the project was carried out between December 2023 and the end of January 2024 where hardware and software platforms and tools were selected, and initial scratch coding was completed.

Project implementation took place over a ten-week time frame commencing on the 29th of January and to be completed by the 29th of March to meet submission deadlines. This was divided up as follows, into four two-week sprints with a week of contingency at the end, which was needed.

- Fog 1- Sprint 1, Jan 28th to Feb 9th
 - Get Data from Sensors into Fog Node local Database.
 - Local web service to display sensor Data.
- Fog 2 – Sprint 2, Feb 12th to Feb 23rd
 - Sensor Data from the Fog node and send to cloud backend.
 - Local web Service to Start and Stop Process
- Cloud 1-Sprint 3, Feb 26th to March 8th
 - User web app data Dashboard
 - Current value Overview page
 - Grid display of latest values
 - Historical trend.
- Cloud 2- Sprint 4, March 11th to March 22nd
 - User web app data Dashboard
 - Start Stop Process
 - Android Native App
 - NFC scan tag on Pasto and show current values.

March 22nd to March 31st was debugging and contingency.

5.0 FUNCTIONAL DESIGN SPECIFICATION

This section outlines how the overall application is put together.

5.1 Edge to Fog Development

This section outlines the work involved in creating the Fog Node, this is the Python part of the application that runs on the RUT95x device and connects to the IO-link master.

I chose python over Node Red, Bash scripting or LUA because of its low overhead, established community, and ability to run on a wide variety of operating systems, and because of its portability between platforms.

During the RAMP phase of the project I had initially planned on using files to store the PVO and PDO data as JSON but after doing some initial testing on that I quickly decided there must be a better way so after a not so brief internet search I came across SQLite ‘..a C-language library that implements a small, fast, self-contained, high-reliability, full-featured, SQL database engine. SQLite is the most used database engine in the world.’ (SQLite, 2024) After doing some tests with python and the SQLite python sqlite3 library I confirmed SQLite as my FOG DB backbone.

A flow chart outlining the python program flow within the fog node can be found in Appendix A – kEdge Program Flow chart.

I divided the python program into four sections/processes.

5.1.1 Main

The main process carries out the following operations,

- Read the PVO inputs at a maximum rate of every 300ms, this is settable in a local “.env” config file. If there is a change of value that value is logged to the DB for live local display.
- Log the PVO variables at a set interval for uploading to the cloud.
- The application will record a PDO when the sequence starts and stops.
- The main program has a minimum cycle time which can be set to minimise loading on the io-link bus and controller itself.

The main process code runs in a continuous loop, until the application is terminated.

I use as SignalHandler Class with a can_run flag to capture the shutdown request and force the application to stop.

```
class SignalHandler:
    shutdown_requested = False
    ProcessName = ''
    def __init__(self, inProcessName):
        self.ProcessName = inProcessName
        signal.signal(signal.SIGINT, self.request_shutdown)
        signal.signal(signal.SIGTERM, self.request_shutdown)

    def request_shutdown(self, *args):
        print('Request to shutdown received, stopping-' + self.ProcessName)
        self.shutdown_requested = True

    def can_run(self):
        return not self.shutdown_requested
```

Figure 14 Signal Handler Class

I used this method to try and carefully stop the process after doing some research on the topic and coming across a good step by step guide by Augustas Pelakauskas called ‘How to Run Python Script as a Service (Windows & Linux)’ (Pelakauskas, 2022)

The Python application auto starts as a process on the router operating system using a basic procd script stored in the /etc/init.d/ , see section 6.6.1 for more.

The inputs from the IO-Link master are read by means of the API server on the master itself and decoded as per the data sheets for the respective devices.

All API requests use the standard python requests library.

5.1.2 Local Webserver Process.

Local web pages allow the user view input and output status and any error messages and trends on the local fog node this can be used for monitoring or back up in case of internet outages. In the case of this application, it is also used to Stop and Start the process.

There appears to be two main frameworks for developing webserver application on Python Flask and Django. In ‘Django Vs. Flask: Understanding The Major Differences’ (Simplilearn, 2023) gives a great overview of both frame works the pros and cons etc.

Given the small size of the fog node and the fact that I failed to get Django to load on OpenWrt then Flask it is. I have also had some experience during the course using Flask to develop a web page on a raspberry pi for ‘Systems and Network assignment IOT’ (Roche, 2022). This however was a verry different beast.

A wireframe diagram of the original design of the landing page design is shown in the following figure.

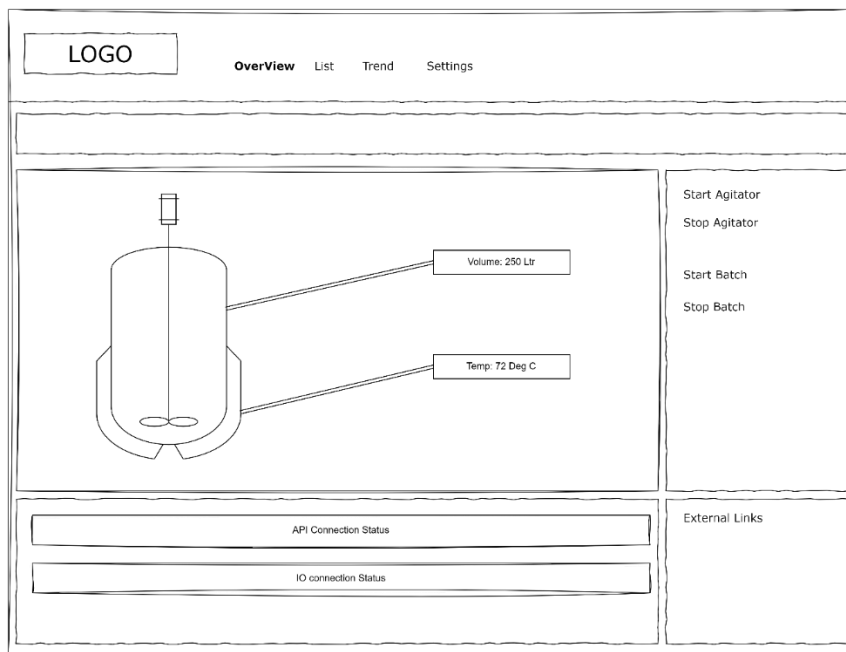


Figure 15 Local web overview orignal wireframe.

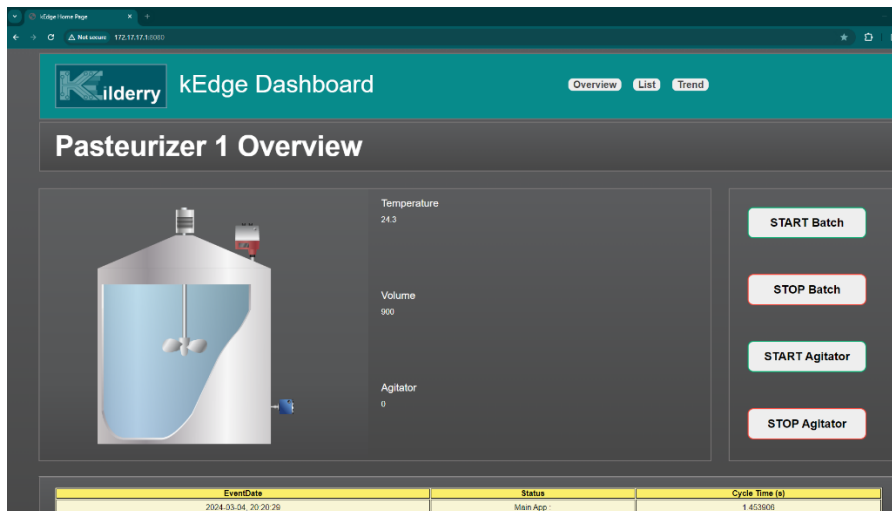


Figure 16 Final Overview page.

The webserver hosts a number of API interfaces that allow the local client side scripting to perform actions on the application, table of these API calls is as follows

Method	Path	Arguments	Result
GET	localhost /list/read	None	JSON Object containing the current PVO values.
GET	localhost /data/read{ pdoKey , StartDate , StopDate }	pdoKey-title of variable to be read StartDate - Start date time for data StopDate – End Date time for Data	Returns a JSON array with the of value, title and event date for the given input arguments.
POST	localhost /trigger/set{ setIndex,setValue)	setIndex – the index number of the value to be triggered. setValue – the value 0 or 1 to be set to the trigger	The Web server reads a trigger value from a data table in the range 0 to 7. If the value is 0 the application can perform some action start stop agitation Start stop the process etc. For the demonstration application the trigger bits are as shown in table 5

Figure 17 Fog Node Web server API calls.

5.1.3 API Web Client Process

This process reads the local database and sends data up to the kCloud server. It checks the PVO and PDO tables for data outstanding to be uploaded, this is done by means of a status field which is the HTML status of the upload API call to the server.

id	jData	InstallationID	NodeID	PortID	EventDate	Status	error
70	421e1f4-7d91-47b7-8019-e7115dccc93d	570	106	3	2024-03-03 20:28:09	200	["title": "Appliator", "unit": "Q", "valueType": 2, "value": 0]
71	e42b190-7d93-46b7-ba3e-c72004126061	570	106	1	2024-03-03 20:28:09	200	["title": "Volume", "unit": "L", "valueType": 3, "value": 878.0]
72	efb26930-1f93-4af3-a267-396404643069	570	106	4	2024-03-03 20:28:09	200	["title": "Heater", "unit": "Q", "valueType": 2, "value": 0]
73	421e1f43-2cc8-49e7-bd53-9a967454f1d	570	106	1	2024-03-03 20:29:41	200	["title": "Level", "unit": "mm", "valueType": 2, "value": 0]
74	ca0b4437-d1f4-49e3-b022-e9f8cdebcb6	570	106	2	2024-03-03 20:29:41	200	["title": "Conductivity", "unit": "uS", "valueType": 3, "value": 0.0]

Figure 18 SQLite PVO table showing data note Status field values 200.

The service runs on a timed basis so that information is only sent to the cloud at the setpoint interval.

```
# ----- Methods -----
def post_pvo(data_in):
    try:
        headers = {'Content-type': 'application/json', 'Accept': 'text/plain'}
        sApiUrl = settings.API_POST_ENDPOINT + "PVO"

        jsonData = json.loads(data_in['jData'])

        # Construct the JSON object
        dataJSON = {'InstallationID': data_in['InstallationID'],
                    'NodeID': "%i" % data_in['NodeID'],
                    'PortID': "%i" % data_in['PortID'],
                    'EventDate': data_in['EventDate'],
                    'jData': jsonData,
                    'error': data_in['error']}

        jsonData = json.dumps(dataJSON, indent=2, ensure_ascii=False)
        response = requests.post(url=sApiUrl, data=jsonData, headers=headers, timeout=10)
        responseCode = response.status_code
        if (responseCode == 200): # all good
            web_post_status.status = "kCloud Client : POST PVO OK"
            return 200
        else:
            web_post_status.status = "kCloud Client : POST PVO CODE : %s" % responseCode
            return responseCode
    except Exception as error:
        if settings.DEBUG:
            web_post_status.status = "kCloud Client : POST PVO: " + str(error)
            return 999
```

Figure 19 Example of a method that uses the Python Requests library to post information to the API backend.

One of the challenges in getting the Web client process to function was coming to grips with sending the data to the API. During the RAMP phase I experimented with the Python Requests: 'HTTP for Humans' (Reitz, 2019) library and I was able to adapt the calls and set headers etc to enable the data to be uploaded.

5.1.4 API GET Client Process

When I originally designed the application and developed the kEdge flow chart as shown in Appendix A, I envisioned an API GET client as a separate process that would call the get/set CDO API on the back end update the software triggers accordingly allowing the user to control the system remotely.

To minimise separate processes interacting with the database and to minimise the amount of coding I chose to add this functionality to the API Web client process instead.

5.1.5 Code Layout and Notes

The Python application uses the following folder layout.

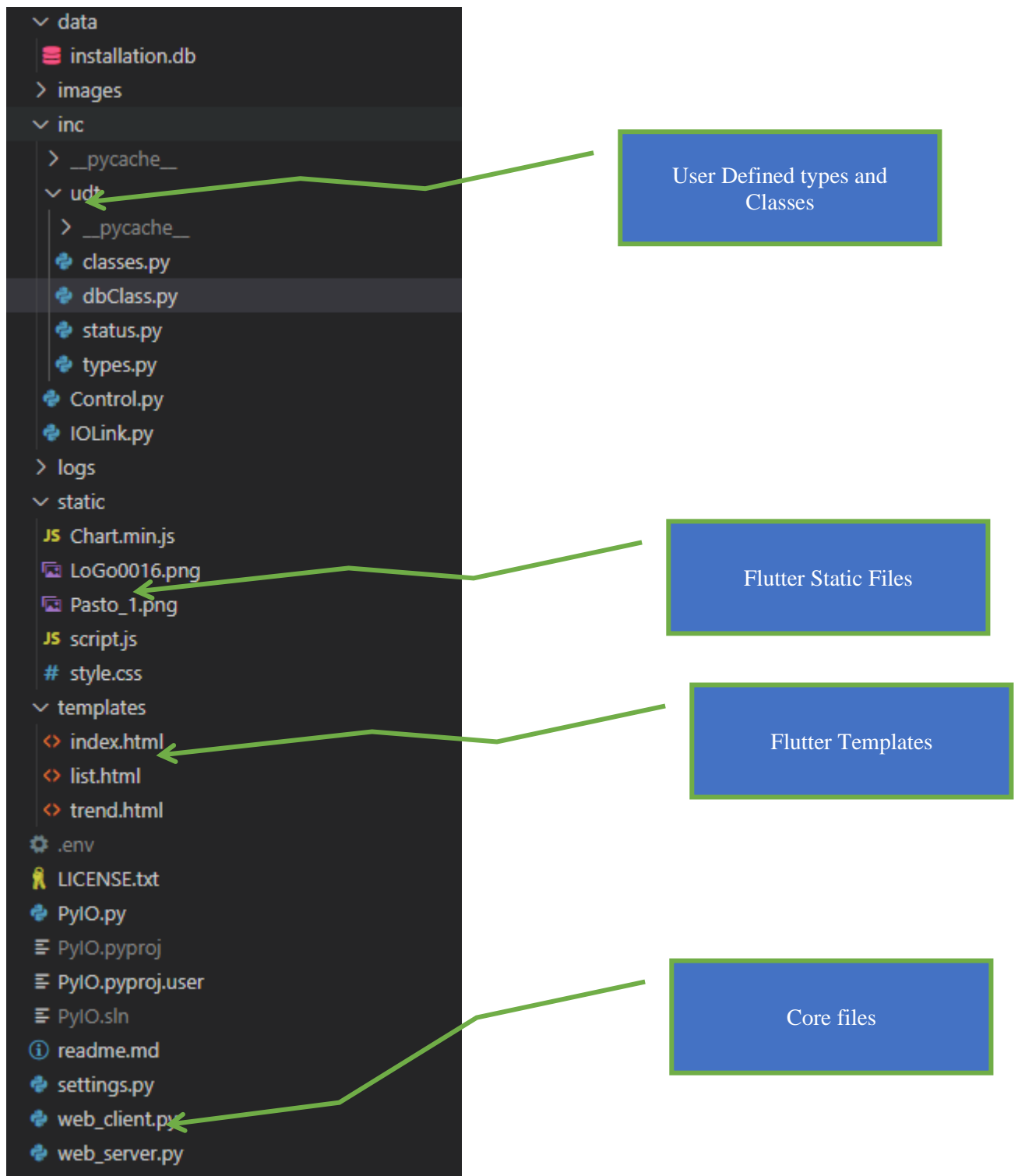


Figure 20 Python Folder layout.

The processes follow a state variable layout whereby they run based on an enumerated state variable these are declared in `../inc/udt/types.py`.

```

19
20 class MainStepType(Enum):
21     init = 0
22     load_config = 2
23     start_web_client = 3
24     start_web_server = 4
25     read_inputs = 5
26     check_event_triggers = 6
27     check_timed_triggers = 7
28     control_logic = 8
29     write_outputs = 9
30     wait = 10
31     check_old_data = 11
32     error = 999
33
34

```

Figure 21 Main step type enum.

To interact with the SQLite data base, I created a dbclass object that can be declared inside each process that requires DB access.

This Class can be found in `../inc/udt/dbclass.py`

One of the advantages in using and SQL database on the FOG node and having to write simple API handlers using flask, was that when it came to the back end some of the heavy lifting in terms of writing SQL queries had already been proven.

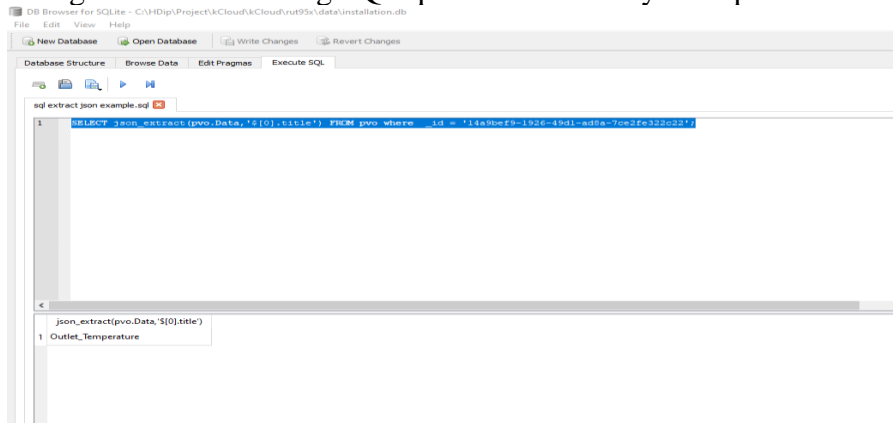


Figure 22 A screen snip from early on in project development of handing JSON in SQLite

5.2 Backend Development

This application is responsible for taking in the information from the fog nodes, allowing access to the data from front end web app and user authentication.

The backend development is implemented in Node JS Express and is hosted on a Linux server and uses MySql as the back-end data base. The Linux server is hosted on an EC2 AWS instance but can also be hosted on a local in factory server.

Why chose Express with Node JS? Having used HAPI during the course, my initial thoughts were to go down that route but during the RAMP phase of the project I experimented with Express just to do API's and test ideas. I did some research and came across many articles such as 'Express vs. Hapi: The Battle For Being Best Node.js Framework' (Dhaduk, 2021) which attempted to clear things up and to me in the end, express did what I needed it to do so I stuck with it.

Interaction to the server is via Restful APIs, API calls outside of user validation is as follows,

Method	Path	Arguments	Result
POST	... /pdo	JSON formatted PDO model.	OK if stored ok.
POST	.../pvo	JSON formatted PVO model.	OK if stored ok.
GET	../cdo/{id}	Instilation id of the Fog Node	If there is a CDO waiting for the node then a JSON formatted cdo model is returned.
POST	../cdo/{id}	Instilation id of the Fog Node,JSON formatted cdo model	OK if stored ok.
GET	../pvo/{id}	Instilation id of the Fog Node	A JSON array of the latest updated PVO values.
GET	../pdo/{id,startDate,StopDate}	Instilation id of the Fog Node, a UTC start date and stop date	A JSON array of PDO for the given instilation id between Start date and Stop date.
GET	../pvo/titles/{id}	Instilation id of the Fog Node	A JSON array of all unique PVO titles for Node

			the given instiation ID.
GET	../pvo/{id,title,startDate,StopDate}	Instilation id of the Fog Node, the title of the process variable to read and a UTC start date and stop date	A JSON array of values for the given instilation id and process variable between between Start date and Stop date.
GET	../read/status/{id}	User ID	Returns a JSON array of

All API specifications are documented using OpenAPI Specification (formerly Swagger Specification).

Some screen shots of the Open API docs can be found in Appendix C

One of the challenges faced when developing the back end was the level and type of SQL queries needed to interact with the JSON data type.

Fortunately, there are several SQL commands that allow interaction with JSON types. The MySQL documentation proved to be a great source here.(MySQL, 2023)

Remember the PVO format as outlined in Section 4.1.2 of this report.

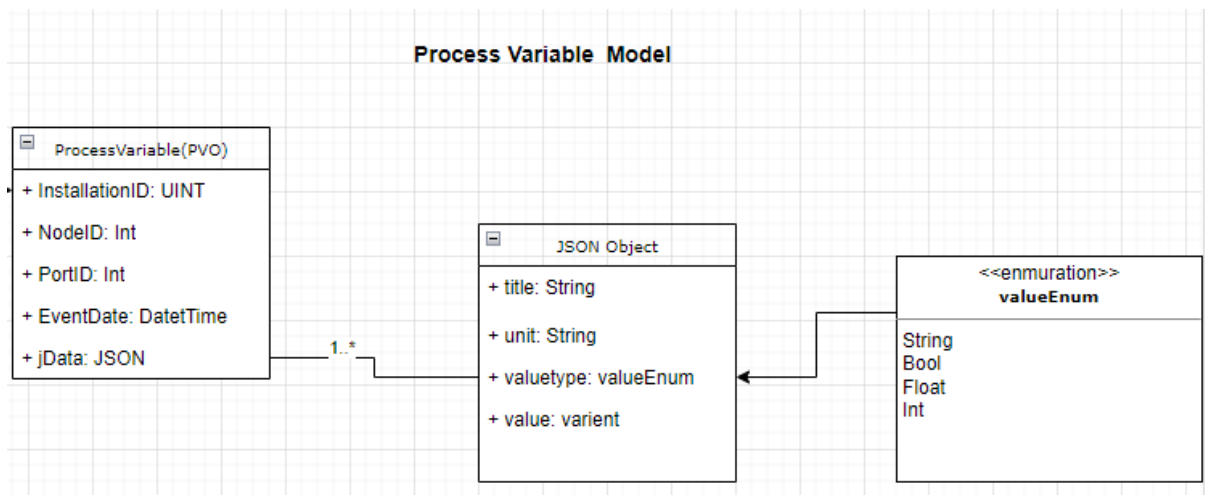


Figure 23 PVO JSON Object

This yielded a data table as visualized in the following figure.

id	InstallationID	NodeID	PortID	EventDate	jData
29,081	1,111,125,017	106	1	2024-03-25 14:31:26	{"unit": "mm", "title": "Level", "value": 447, "valueType": 5}
29,082	1,111,125,017	106	2	2024-03-25 14:31:26	{"unit": "uS", "title": "Conductivity", "value": 0, "valueType": 5}
29,083	1,111,125,017	106	2	2024-03-25 14:31:26	{"unit": "C", "title": "Temperature", "value": 23.5, "valueType": 5}
29,084	1,111,125,017	106	3	2024-03-25 14:31:26	{"unit": "QI", "title": "Agitator", "value": 0, "valueType": 2}
29,085	1,111,125,017	106	1	2024-03-25 14:31:26	{"unit": "L", "title": "Volume", "value": 894, "valueType": 5}
29,086	1,111,125,017	106	4	2024-03-25 14:31:26	{"unit": "QI", "title": "Heater", "value": 0, "valueType": 2}

Figure 24 Snip from PVO data table showing various PVO values.

The jData field is the embedded JSON data that describes the process variable being uploaded.

For the data shown above the units are **mm** (milli meters) for level, **uS** (micro siemens) for conductivity, **C** (degrees Celsius) for temperature, **L** (liters) for Volume and **QI** (Digital input) for Heater and agitator.

The following SQL extracts a list of all the PVO titles being uploaded.

```

1 SELECT DISTINCT json_extract(jData, '$.title') AS Title FROM kcloud.pvo;

```

Title
"Temperature"
"Agitator"
"Volume"
"Heater"
"Level"
"Conductivity"
"Line 1 Power"

Figure 25 SQL to Extract all PVO titles.

We can use the JSON extract SQL within more complex statements such as the one shown below which extracts the average hourly temperature value.

```

1 SELECT installationid, json_extract(jData, '$.title') as Title, hour(EventDate) as HH,
2 avg(json_extract(jData, '$.value')) AS AvgTemp, min(json_extract(jData, '$.value')) AS MinTemp, max(json_extract(jData, '$.value')) AS MaxTemp FROM pvo WHERE
3 (EventDate BETWEEN '2024-03-01 18:00:00' AND '2024-03-06 18:59:59') AND
4 (json_extract(jData, '$.title') = 'Temperature') AND (installationid = 570) group by HH ORDER BY HH desc;

```

installationid	Title	HH	AvgTemp	MinTemp	MaxTemp
570	"Temperature"	22	24.4	24.4	24.4
570	"Temperature"	21	26.021153846153865	25.7	26.2
570	"Temperature"	20	25.546666666666665	25.2	26.1
570	"Temperature"	19	25.2	25.2	25.2

Figure 26 SQL query to extract average hourly temperature data.

The following figure shows SQL being used in a routine that extracts PVO information from the database for a particular installation, title, and time.

```

718 //
719
720 async function readPVO_Specific(
721   sInstallationID,
722   pdoVariableKey,
723   dtStart,
724   dtEnd,
725   sInterval
726 ) {
727   return new Promise(async function (resolve, reject) {
728     dbConnection.ping((err) => {
729       resolve(err);
730     });
731     if (disconnected) {
732       dbConnection = mysql.createConnection(ConnectionData);
733     }
734     if (sInterval == 'hourly') {
735       var myQuery =
736         "SELECT installationid,json_extract(jData, '$.title') as Title, hour(EventDate) as HH, day(EventDate) as D,"
737         + "avg(json_extract(jData, '$.value')) as AvgVal,min(json_extract(jData, '$.value')) as MinVal,max(json_ext"
738         + "FROM pvo WHERE (EventDate BETWEEN '" + dtStart + "' AND '" + dtEnd + "')"
739         + " AND (json_extract(jData, '$.title') = '" + pdoVariableKey + "') AND (installationid = " + sInstallation
740         + " group by HH,DD,HH ORDER BY HH,DD asc";
741     } else if (sInterval == 'daily') {
742       var myQuery =
743         "SELECT installationid,json_extract(jData, '$.title') as Title, day(EventDate) as DD, month(EventDate) as M,"
744         + "avg(json_extract(jData, '$.value')) as AvgVal,min(json_extract(jData, '$.value')) as MinVal,max(json_ext"
745         + "FROM pvo WHERE (EventDate BETWEEN '" + dtStart + "' AND '" + dtEnd + "')"
746         + " AND (json_extract(jData, '$.title') = '" + pdoVariableKey + "') AND (installationid = " + sInstallation
747         + " group by MM,DD ORDER BY MM,DD asc";
748     } else {
749       var myQuery =
750         "SELECT installationid,json_extract(jData, '$.title') as Title,json_extract(jData, '$.unit') as Unit,"
751         + "json_extract(jData, '$.value') as Value,EventDate "
752         + "FROM pvo WHERE (EventDate BETWEEN '" + dtStart + "' AND '" + dtEnd + "')"
753         + " AND (json_extract(jData, '$.title') = '" + pdoVariableKey + "') AND (installationid = " + sInstallation
754         + " group by MM,DD,HH ORDER BY MM,DD,HH asc";
755     }
756     dbConnection.execute(myQuery, (err, result) => {
757       resolve(result);
758     });
759   });
760 }
761

```

Figure 27 API data base model showing SQL in use.

Before I implemented the Swagger UI, I implemented some local API tests using Mocha to test the database connection and authentication these can be found in the test subfolder.

```

1 const { assert } = require("chai");
2 const { assertSubset } = require("../test-utils.js");
3 const { kCloudService } = require("../kcloud-service.js");
4 const { maggie,maggieCredentials, testUsers, homer,homerCredentials } = require("../fixtures.js");
5 const users = new Array(testUsers.length);
6
7 Run|Debug|Show in Test Explorer
8 suite("User API tests", () => {
9   setup(async () => {
10     await kCloudService.clearAuth();
11     await kCloudService.createUser(maggie);
12     await kCloudService.createUser(homerCredentials); // create a jwt for test user
13     await kCloudService.createUser(maggieCredentials); // create a jwt for test user
14     for (let i = 0; i < testUsers.length; i += 1) {
15       users[i] = await kCloudService.createUser(testUsers[i]);
16     }
17   });
18   teardown(async () => {});
19
20 Run|Debug
21 test("
22   const
23   assertSubset(homer, newUser);
24   assert.isDefined(newUser.id);
25
26 Run|Debug
27 test("delete all user api2" async () => {
28   let returnedUsers = await kCloudService.getAllUsers();
29   assert.equal(returnedUsers.length, 4);
30   await kCloudService.deleteAllUsers(); // delete them all
31   await kCloudService.createUser(maggie); // create and authenticate the test user so we call the g
32   await kCloudService.authenticate(maggieCredentials);
33

```

Figure 28 API project Playout.

The API implements JWT authentication, I tested this for several routes mainly within user management as well as PVO write.

5.2.1 Swagger UI

When looking for a way to best utilize my time for both documenting then implementing the APIs I decided to use Swagger UI to try and achieve both in the shortest time possible.

Although I found the JDocs cumbersome to work with, after a while you figure out where you are going wrong and can make good progress. Googling was helpful as always, but one web page I found helped a long way to get started 'How to Document an Express API with Swagger UI and JSDoc' (Bartolo, 2020) gives a good introduction to the Swagger UI.

I wrote the jdocs information in line with the code for easy access rather than in separate files, as shown in the figure below.

```

101 //----- API Calls -----
102 //----- PVO Write To DB -----
103 /**
104  * @swagger
105  * /pvo:
106  *   post:
107  *     tags:
108  *       - P.V.O.
109  *     summary: Add a Process Variable to the Database.
110  *     requestBody:
111  *       required: true
112  *       content:
113  *         application/json:
114  *           schema:
115  *             $ref: '#/components/schemas/PVO'
116  *     responses:
117  *       200:
118  *         description: PVO ADDED
119  *         content:
120  *           application/json:
121  *             schema:
122  *               $ref: '#/components/schemas/SQLReturn'
123  */
124
125 app.post("/", async function (req, res) {
126   const webReq = req;
127   const data = webReq.body;
128   try {
129     await writePVOData(data.InstallationID, data.NodeID, data.PortID, data.EventDate, data.jData).then(
130       (response) => {
131         res.send(response);
132       },
133       (response) => {
134         console.log(" Then Failure:" + response);
135         res.send(response);
136       }
137     );
138   } catch (error) {
139     res.status(500).send(error);
140   }
141 }
142

```

Figure 29 JS docs for PVO Post Swagger UI

The above code is compiled via the swagger UI to the document shown below,

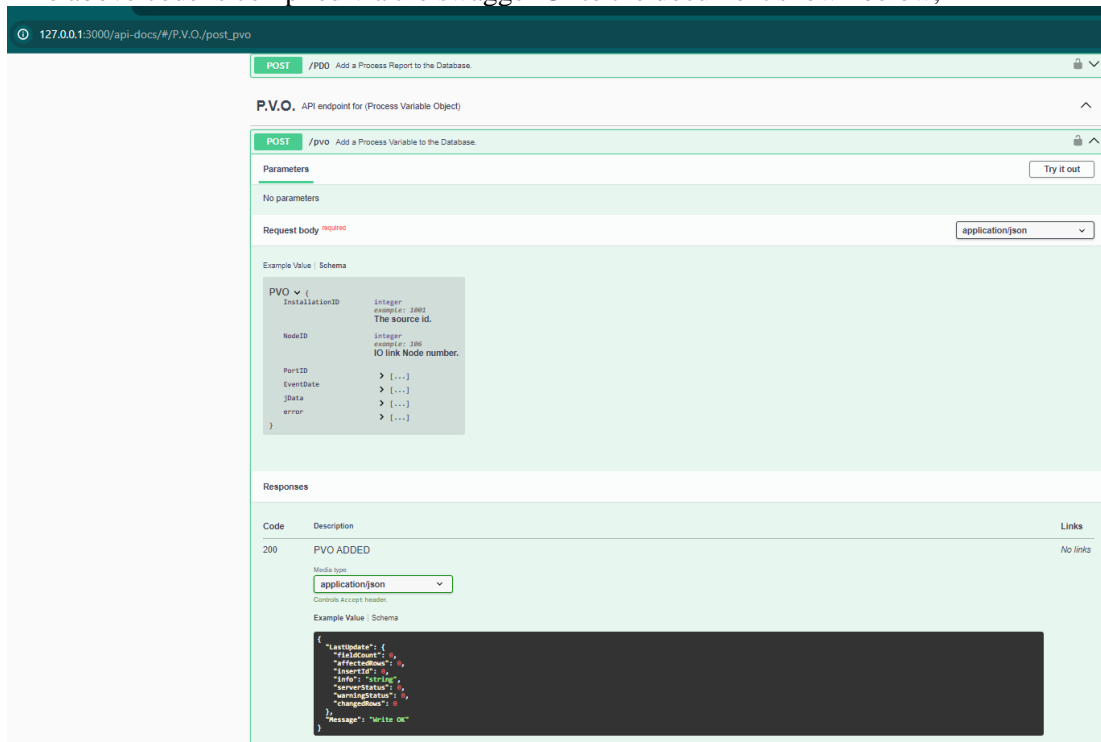


Figure 30 Swagger UI for the API outlined in previous figure.

I used the Swagger UI extensively to test the functionality of the operation of the API and the validity of the contained data.

5.3 Front End Web Application

The web application is written using the Svelte-kit frame work and Bulma CSS for most user style elements. It enables the user to,

- View online / offline systems.
- Select an installation to show status trends and generate reports.

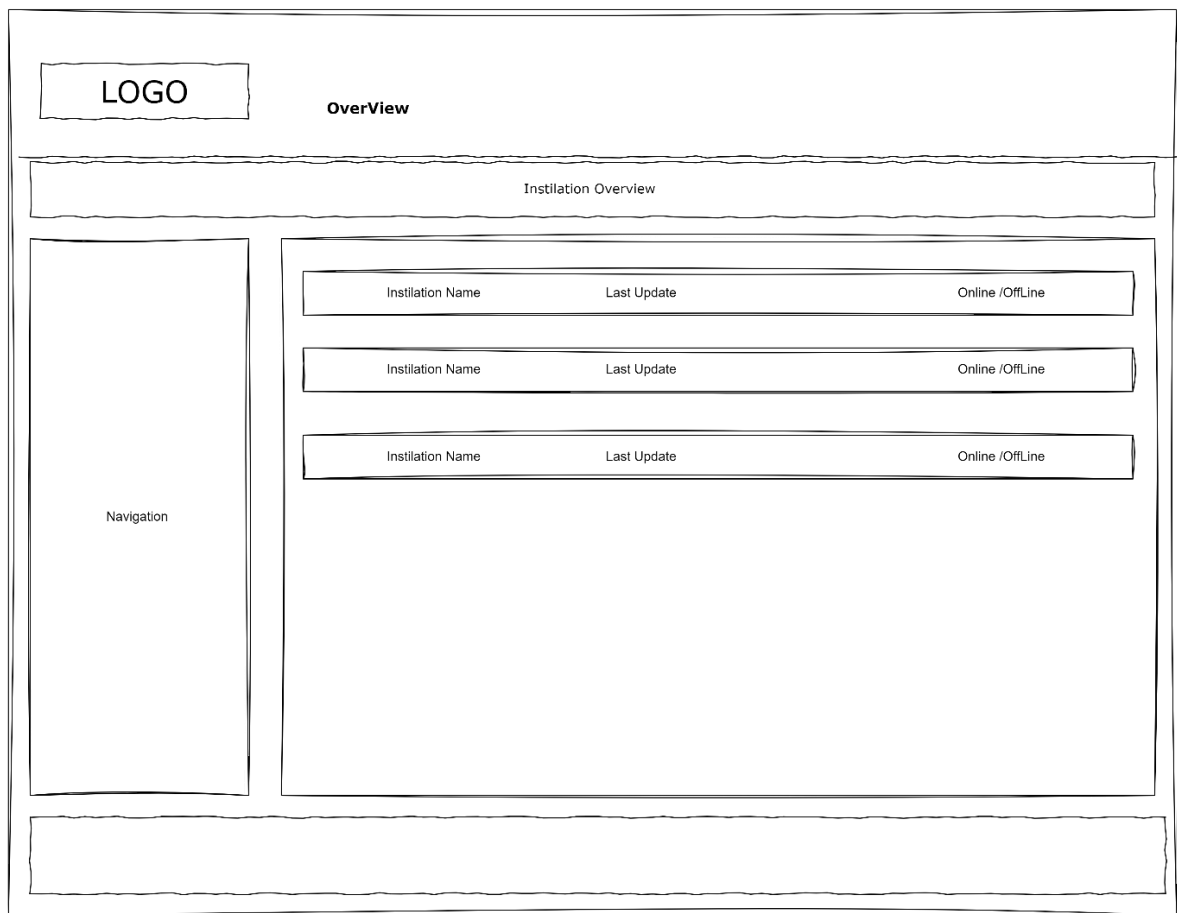
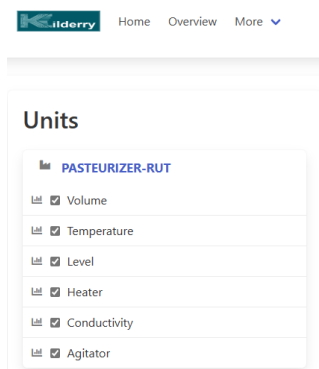


Figure 31 Web App Overview page, original wire frame design.

I investigated several downloadable Svelte kit dashboard templates, checking to see how they looked and if they would suit my design idea, in the end, I chose none of them and opted to utilise simple Bulma columns for layout and design reuseable widgets that would suit the needs of the project and offer a broader learning experience,



One of the Bulma elements used is accordion navigation. I developed the panel widget using this which allows the user to select the installation by name. It then expands into a drop-down list with check boxes to allow selection of PVO.

Figure 32 Navigation widget for each Installation.

Another element reused in the Web app is the value box used to display latest PVO values note the title, vinit and Value all come from the JSON jData field of the PVO.

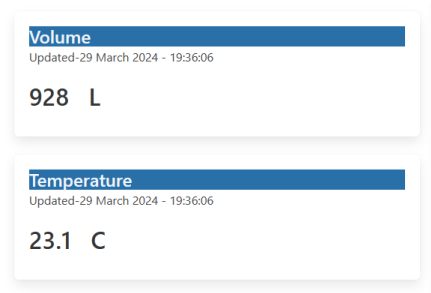


Figure 33 Pasteurizer data in value element.

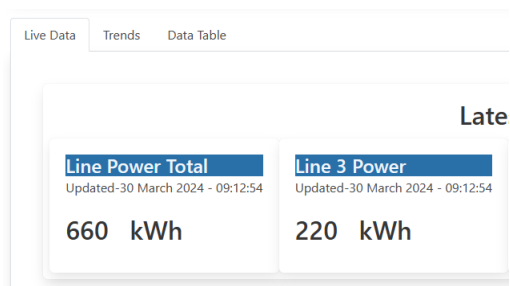


Figure 34 Power meter Data in Value element.

For trending I used frappe charts from Frappe ‘.. a remote technology company committed to building delightful applications and services’ (Frappe, 2024). I constructed a common date range widget to select the data range for the chart with common ranges available at the press of a button.

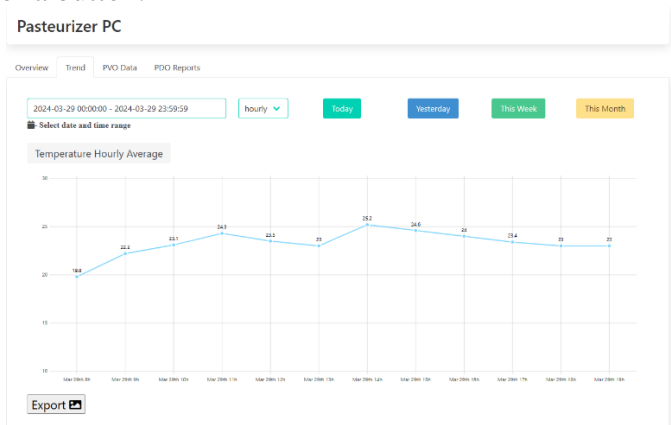


Figure 35 Date widget at the top.

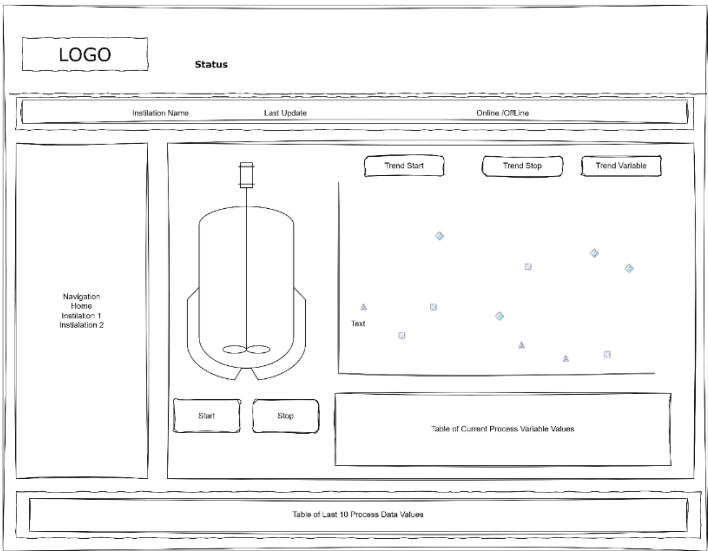


Figure 36 Web App Status page original wireframe design.

The final design of the page uses tabs to select various functions, this led to a less cluttered simpler layout.

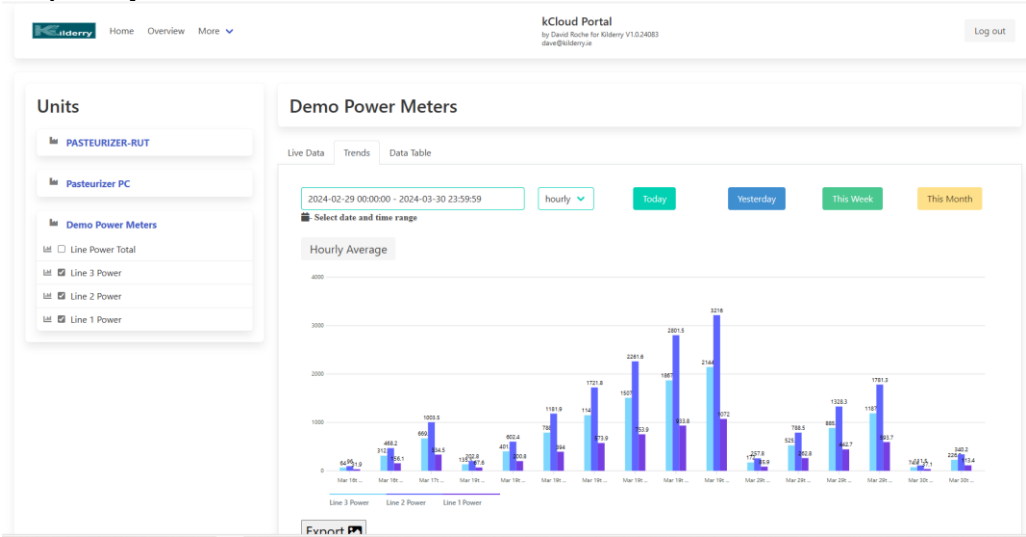


Figure 37 Web App Actual Screen with Trends.

For the most part the web application is Server side rendered with svelte kit loading the pages first and sending to the browser for display.

5.4 Native Android App

This original plan had been to radically alter the application developed for ‘Mobile Application Development Assessment.’ (Roche, 2024) .

Not all the functionality intended was added. This was due to a business decision to concentrate on adding an additional possible implementation example and spend more time on developing that instead of spending it on the Android application.

There was some additions and modification done which are included with the project. namely to communicate fully to the kCloud back end of the application via volley as well as take the information from the PVO and display on the app. A rudimentary NFC tag scan function was also added.

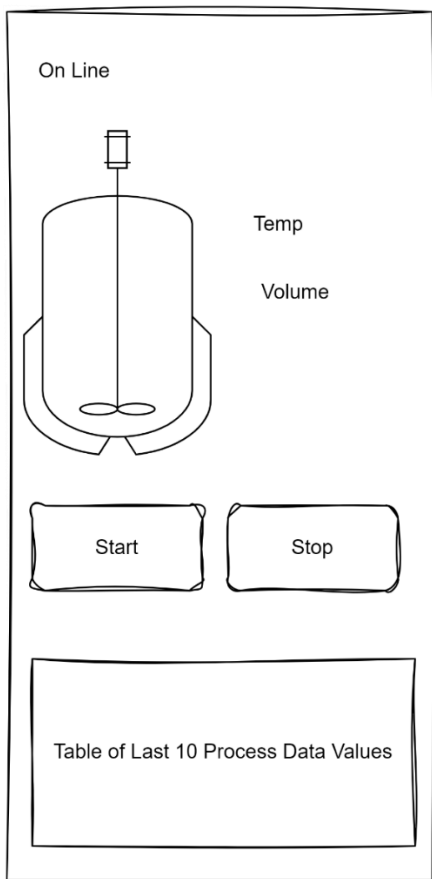


Figure 38 Proposed Android App Screen



Figure 39 Android application Installation screen floating action button for NFC scanning.

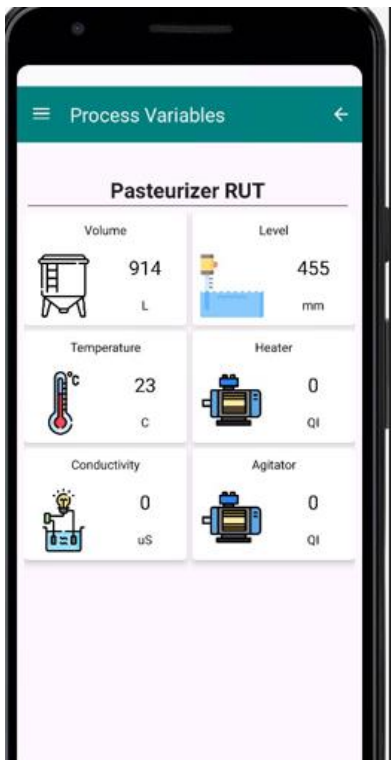


Figure 40 Live PVO data.

On researching NFC I found a great article on medium .com by Carlos Javier Torres Pensa Where he states that NFC ‘allows us to wirelessly pair and connect devices over short distances instantly.’ (Pensa, 2021). If the installation ID is programmed into the NFC chip then that can act as a locator for reporting in the future e.g.

- I can know a user has been in proximity to the installation as the NFC tag scanned tag is mounted in place.
- In the future I can send up location reports for sites that maybe don’t have GPS or No coverage, eg underground maintenance panels etc.

In his article Pensa goes on to give an example of writing to an RFID tag and reading back however he uses a different tag type then the one I have. The tag type I have is ISO-15693 (ISO, 2019), which is catered for directly in android studio by means of the ‘NfcV class’. (Android Developer, 2024) .

After doing, some more investigating and reading lots of online forms I developed enough knowledge to be able to write the ID number of an instillation to the RFID tag and read it back. The NfcV class transceiver method receives a byte array of commands depending on the tag and replies with a byte array.

```
//Read Nfc V tags ISO-15693
var mNfcV = NfcV.get(tag)
if (mNfcV != null) {
    mNfcV.connect() // connect to the tag
    val tagUid: ByteArray = mNfcV.tag.id // store tag UID for use in addressed commands

    var response: ByteArray
    val offset = 0 // offset of first block to read 0 each block is 4 bytes
    val blocks = 8 // number of blocks to read
    val cmdReadMultiple =
        byteArrayOf(
            0x60.toByte(), // flags: addressed (= UID field present)
            0x23.toByte(), // command: READ MULTIPLE BLOCKS
            0x00.toByte(),
            0x00.toByte(),
            0x00.toByte(),
            0x00.toByte(),
            0x00.toByte(),
            0x00.toByte(),
            0x00.toByte(),
            0x00.toByte(), // placeholder for tag UID
            (offset and 0x0ff).toByte(), // first block number
            ((blocks - 1) and 0x0ff).toByte() // number of blocks (-1 as 0x00 means one block)
        )
    try {
        // copy in the tag ID for the read command
        System.arraycopy(tagUid, 0, cmdReadMultiple, 2, 8);
        // uses transceiver to send byte command to tag
        response = mNfcV.transceive(cmdReadMultiple)
        // Installation id stored as string number max 12 chars
    }
}
```

Figure 41 Snippet of Code that Triggers a read of 32 bytes from address 0, when the on Tag discovered event is triggered.

```
fun WriteData(currentTag: NfcV) {
    val id: ByteArray = currentTag.tag.id
    val dataString = "1111125017".padEnd(12, ' ')
    val byteArray = dataString.toByteArray()
    val offset = 0 // offset of first block to read
    val blocks = 2 // number of blocks to write
    val data: ByteArray = byteArray
    val cmdWriteMultiple =
        byteArrayOf(
            0x22.toByte(), // flags: addressed (= UID field present)
            0x21.toByte(), // command: write command
            0x00.toByte(),
            0x00.toByte(),
            0x00.toByte(),
            0x00.toByte(),
            0x00.toByte(),
            0x00.toByte(),
            0x00.toByte(),
            0x00.toByte(), // placeholder for tag UID
            (offset).toByte(), // offset
            0x00.toByte(),
            0x00.toByte(),
            0x00.toByte(),
            0x00.toByte()
        )
    // data
    System.arraycopy(id, 0, cmdWriteMultiple, 2, 8);
    for (i in 0 until blocks - 1) {
        cmdWriteMultiple[10] = (offset + i and 0x0ff).toByte()
        //System.arraycopy(data, 4 + i, cmdWriteMultiple, 11, 4)
        val iSourcePos = 4 + i
        System.arraycopy(data, iSourcePos, cmdWriteMultiple, 11, 4)
        val response: ByteArray = currentTag.transceive(cmdWriteMultiple)
        response
    }
}
```

Figure 42 Snip of Write Tag Routine

6.0 ISSUES AND RESOLUTIONS

6.1 Fog Node Issues

6.1.1 Auto starting on Open WRT

My initial python development had been completed using the Linux Mint OK on a local virtual Machine, this worked well, and I was able to setup auto restart using systemd. When I moved to the RUT95X and Open WRT no sysrtemd! Back to the drawing board, eventually after what seemed like forever trying various scripts and hacks from far too many forms to mention, I went back to the source and referenced the OpenWrt docs and found two articles that steered me true.

The first ‘Managing Services’ (OpenWrt, 2023) and the second an example ‘Create a sample procd init script’ (Oostdijk, 2024) combining bit from these and reacquainting myself once more with the VI editor, I was able to write an init.d script and call it on startup.

```
root@RUT955:/etc/init.d# cat pyio_service
#!/bin/sh /etc/rc.common
USE_PROCD=1
START=95
STOP=01
start_service() {
    procd_open_instance
    procd_set_param command python3 "/root/pyio/PyIO.py"
    procd_close_instance
}

root@RUT955:/etc/init.d#
```

Figure 43 init.d config file for the pyio service

```
root@RUT955:/etc/init.d# service pyio_service enable
root@RUT955:/etc/init.d#
```

Figure 44 enabling the service.

```
root@RUT955:/etc/init.d# service pyio_service start
root@RUT955:/etc/init.d#
```

Figure 45 Starting the service.

6.1.2 Don't mention the firmware!

The firmware for the router was updated to version 7.6.3 I thought I should update mine bad idea, this caused issues when installing python and PIP as the router manufactures no longer supported the Open WRT package repositories. So, I had to revert to a slightly older version where the router manufacturer allowed updating of packages directly from OpenWrt.

6.1.3 Communication between sub processes

The python program uses process to manage the main sequence and web server separately one of the issues with this is sharing data between them.

For example, I want to be able to show the status of the main task on the webpage, so I need to share it between the main task to the web server task.

The first versions of the program used a status table in the local db to store the information, this worked but was resource hungry and lead to a lot of DB interaction.

I overcame this, after many hours googling and testing by using shared memory between processes.

‘Python types can be converted to arrays of bytes and stored in a Shared Memory and read as arrays of bytes and converted back into Python types.’ (BROWNLEE, 2023)

```
211 pWebServer = Process(
212     target=startWEBServer, args=(8080, shared_mem_status_web_server_main)
213 )
```

Figure 46 Webserver Process Pointer inside PyIo main file with shared memory space as argument.

```
68
69 # create a shared memory
70 shared_mem_status_web_server_main = shared_memory.SharedMemory(
71     create=True, size=150
72 ) # for status of main task to web server
73
```

Figure 47 Declaration of Shared memory variable.

```
88 def log_status():
89     # compile the status of the PyIO main routine and send to the web
90     # server by means of shared memory.
91     dtNow = datetime.now()
92     sDateTime = dtNow.strftime("%Y-%m-%d, %H:%M:%S")
93     sString = (
94         sDateTime + ";" + main_status.status + ";" + str(main_status.cycle_time) + ";"
95     )
96     bString = sString.encode("utf-8")
97     iLen = len(bString)
98     shared_mem_status_web_server_main.buf[:iLen] = bString
99
```

Figure 48 Writing the Data to the Shared memory for the Web server.

```
# ----- Web Routes -----
# local index page
@app.route("/")
def home():
    sStatus = bytes(sm_main_status.buf[:150]).decode('utf-8')
    aStatus = sStatus.split(";")
    sEventDate = aStatus[0]
    sStatus = aStatus[1]
    sCycleTime = aStatus[2]
    return render_template("index.html", EventDate = sEventDate, Status = sStatus, CycleTime=sCycleTime)
```

Figure 49 Webserver Home page route handler.

EventDate	Status	Cycle Time (s)
2024-03-04, 20:20:29	Main App :	1.453906

Figure 50 What the user sees on the home page.

6.2 Back end /Front End Issues

Leaving aside any coding issues one of the frustrating things about starting projects like this is getting the environment right.

6.2.1 Node Versions

The latest version available for Ubuntu Linux via apt updates is v10.19.0.

I needed v18.00.00 or more for the API back end. After more head scratching and googling, I found a great tutorial by Brian Boucheron ‘How To Install Node.js on Ubuntu 20.04’ (Boucheron, 2023).

I decided to install Node Version Manager that would allow me to select and run various versions of node on the server.


```

ubuntu@ip-172-30-2-52:~$ nvm install v20.11.1
Downloading and installing node v20.11.1...
Downloading https://nodejs.org/dist/v20.11.1/node-v20.11.1-linux-x64.tar.xz...
#####
Computing checksum with sha256sum
Checksums matched!
Now using node v20.11.1 (npm v10.2.4)
ubuntu@ip-172-30-2-52:~$

```

Figure 51 Installing latest node Version using NVM

```

ubuntu@ip-172-30-2-52:~/kcloud-web$ nvm use v20.11.1
Now using node v20.11.1 (npm v10.2.4)
ubuntu@ip-172-30-2-52:~/kcloud-web$

```

Figure 52 using NVM to select latest version of node.

6.2.2 More Start stop Issues.

One of the issues I came across again on the back end was auto starting the API and Web Servers and keeping them running once I closed the SSH session.

I thought of using rc.local, that didn't work I tried systemd but that gave me issues with the wrong node versions AGAIN...

Enter PM2 'PM2 is a process runner, basically will keep your API listening even when you end the SSH session with your Ubuntu server' (JonathanSanchez.Dev, 2021)

The API server I could start directly by calling `pm2 start src/server.js`.

```

ubuntu@ip-172-30-2-52:~/kcloud$ pm2 start src/server.js

```



```

-----
Runtime Edition
PM2 is a Production Process Manager for Node.js applications
with a built-in Load Balancer.

Start and Daemonize any application:
$ pm2 start app.js

Load Balance 4 instances of api.js:
$ pm2 start api.js -i 4

Monitor in production:
$ pm2 monitor

Make pm2 auto-boot at server restart:
$ pm2 startup

To go further checkout:
http://pm2.io/
-----

[PM2] Spawning PM2 daemon with pm2_home=/home/ubuntu/.pm2
[PM2] PM2 Successfully daemonized
[PM2] Starting /home/ubuntu/kcloud/src/server.js in fork_mode (1 instance)
[PM2] Done.

```

id	name	mode	u	status	cpu	memory
0	server	fork	0	online	0%	27.4mb

```

ubuntu@ip-172-30-2-52:~/kcloud$

```

Figure 53 API server starting in PM2

I wrote a small script to start the Web Server as it was a little more complex.


```

start-server.sh
1  #!/bin/bash
2  API_DIR="/home/ubuntu/kcloud-web"
3
4  SERVER_COMMAND="npm run start >> webstartlog.txt"
5
6  # Change to the server directory and start the server
7  cd "$API_DIR"
8
9  eval "$SERVER_COMMAND"
10
11 #exit 1
12

```

Figure 54 Web Server Start Script.

```

ubuntu@ip-172-30-2-52:~/kcloud-web$ pm2 start start-server.sh
[PM2] Starting /home/ubuntu/kcloud-web/start-server.sh in fork_mode (1 instance)
[PM2] Done.

```

id	name	mode	u	status	cpu	memory
1	npm	fork	45	errored	0%	0b
0	server	fork	0	online	0%	100.5mb
2	start-server	fork	0	online	0%	3.0mb

```

[PM2][WARN] Current process list is not synchronized with saved list. Type 'pm2 save' to synchronize.
ubuntu@ip-172-30-2-52:~/kcloud-web$

```

Figure 55 PM2 Status after running script.

6.2.3 Frappe Charts Datasets

One of the things I wanted to do was to dynamically add and remove data sets from the current displayed chart.

If I removed a data set, the chart would not update and would also leave the entire web app unresponsive. If I minimised the window or refreshed the page in the address bar the correct chart would appear with the data set removed but the app was still unresponsive.

After hours of re referencing data and checking loaded data and trying different solutions. I overcame it by simply having two charts in the page and using a toggle every time the chart data was changed.

```

216 {#if xToggleChart}
217   <Chart
218     data={PowerChartData}
219     height={ChartHeight}
220     type={ChartType}
221     isNavigable={enableNavigation}
222     valuesOverPoints={showValuesOverPoints}
223     xIsSeries={xAxisIsSeries}
224     {tooltipOptions}
225     {axisOptions}
226     barOptions={barOptions}
227     bind:this={chartRef}
228   />
229 {:else}
230   <Chart
231     data={PowerChartData}
232     height={ChartHeight}
233     type={ChartType}
234     isNavigable={enableNavigation}
235     valuesOverPoints={showValuesOverPoints}
236     xIsSeries={xAxisIsSeries}
237     {tooltipOptions}
238     {axisOptions}
239     barOptions={barOptions}
240     bind:this={chartRef}
241   />
242 {/if}

```

Figure 56 Sveltekit Data set select solution.

7.0 REFLECTION

If one was to google “how to eat an elephant” chances are you would get “one piece at a time”. This project had a lot of moving parts, and I did somewhat regret taking on quite so big an elephant. But in the end the job was done. I must really stick more closely to the ‘Scotty Principle’ (urbandictionary, 2005) .

I found Python to be a very adaptable language with great online resources. It allowed me to develop code to run on both PC and Linux. Giving the benefit of being able to run on my actual fog node and on the development PC.

Trying to shoehorn a control data capture and HMI application into a router took a lot more effort than expected.

7.1 Project Goals achieved and not.

In terms of creating a foundation that can be built on for us to help customers achieve the goals I set out at the start, I believe I have achieved a lot of what I initially set out to do,

- A Python based Fog Node Example machine with,
- data uploading in a defined controlled way to a,
- NodeJS Express API Server with MySQL data back bone to allow user interaction via,
- A Svelte-kit based web app for desktop and
- A Kotlin Android application.

In those terms a success.

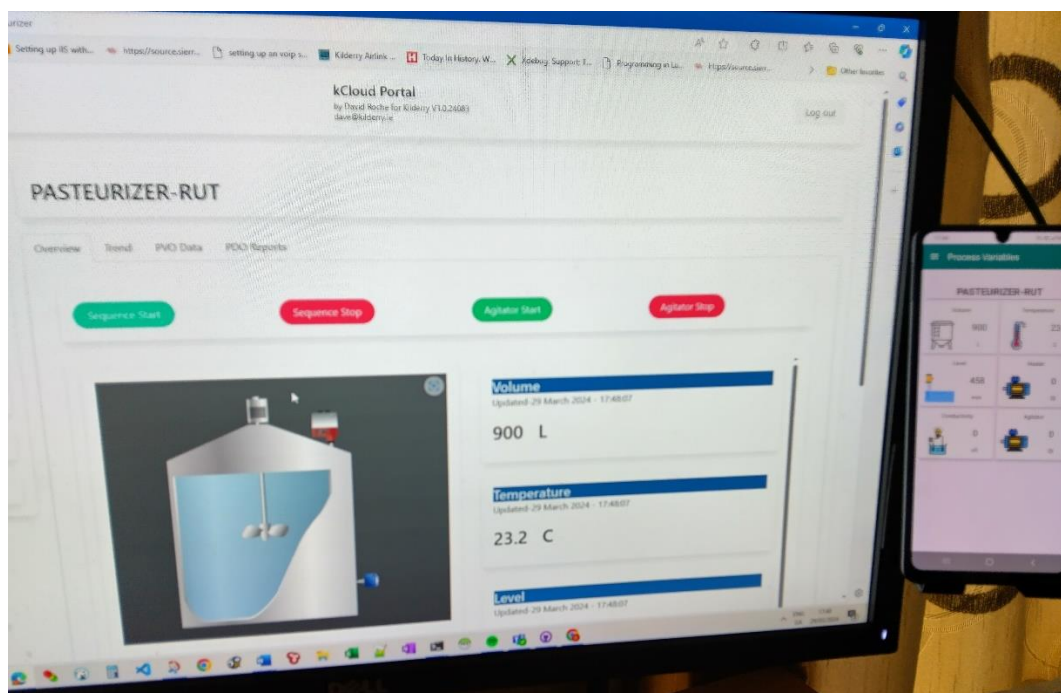


Figure 57 Realtime data on app and desktop.

I had initially planned on doing a lot more development on the Android application however two main factors limited me in this.

The Fog node application when deployed on the router took a lot of time to get working correctly. This was mainly due to the limited resources on the router coupled with the demands I was placing on those resources. There was also a business need to develop a

function block in Codesys 3.5 that would work with the kCloud backend which took up the time I had allocated to the android development, but as outlined in section 7.3 this is something to be developed in the future.

Would I use Python as a control platform?

For a one off mainly Research and Development application yes, no problem its responsive and straight forward to develop with lots of plugins and resources. Also, the flexibility to offer customers local logging and full featured web servers is very attractive. It also has an advantage of being cross platform.

It does appear to be for me anyway a little resource hungry on lean devices such as the router we were testing on.

Overall, a good tool to have in the bag if needed.

7.2. What I learned.

This course has been a great learning experience and well worth doing, being exposed to different technologies and platforms has greatly helped broaden my understanding of what is happening in the greater industry right now.

It will indeed help me to be able to understand the technologies that are coming into automation and the ones that are already there.

7.2.1 A very brief word on AI.

One of the biggest talking points in the last few years has and is the rise of Artificial Intelligence, or I guess the more publicly accessible versions of AI, as AI has been around in limited forms and concepts for a long time.

Doing the project, I ventured into the Bing copilot, yes, I know not very exciting, I wanted to see if it could take me on a wild AI journey. Think I must have asked the wrong questions as I just got lost and so did the copilot. I did get one AI to design me a nice logo for kCloud though, you can see it on the GitHub repo. That is as far as I got into AI for now anyway.

7.3 Future Development Work.

One of the aims of this project from an industrial point to view was to have a system of getting information from a local machine into a server or cloud space for the customer, without the addition of expensive third-party hardware or licencing. A Structured Text ST function block has been developed. This is to transmit a value from standard Codesys 3.5 programmable PLC to the API server. This has been used to transmit values for the power meters demo screens. This will be further developed over the coming months.

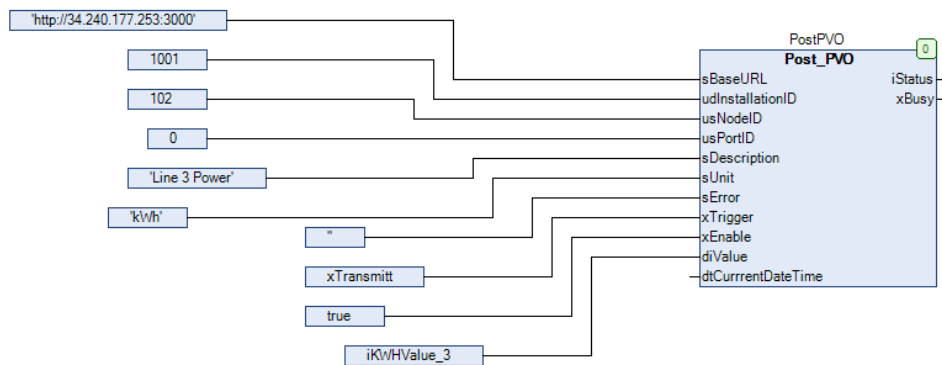


Figure 58 Function block in use Codesys 3.5

BIBLIOGRAPHY

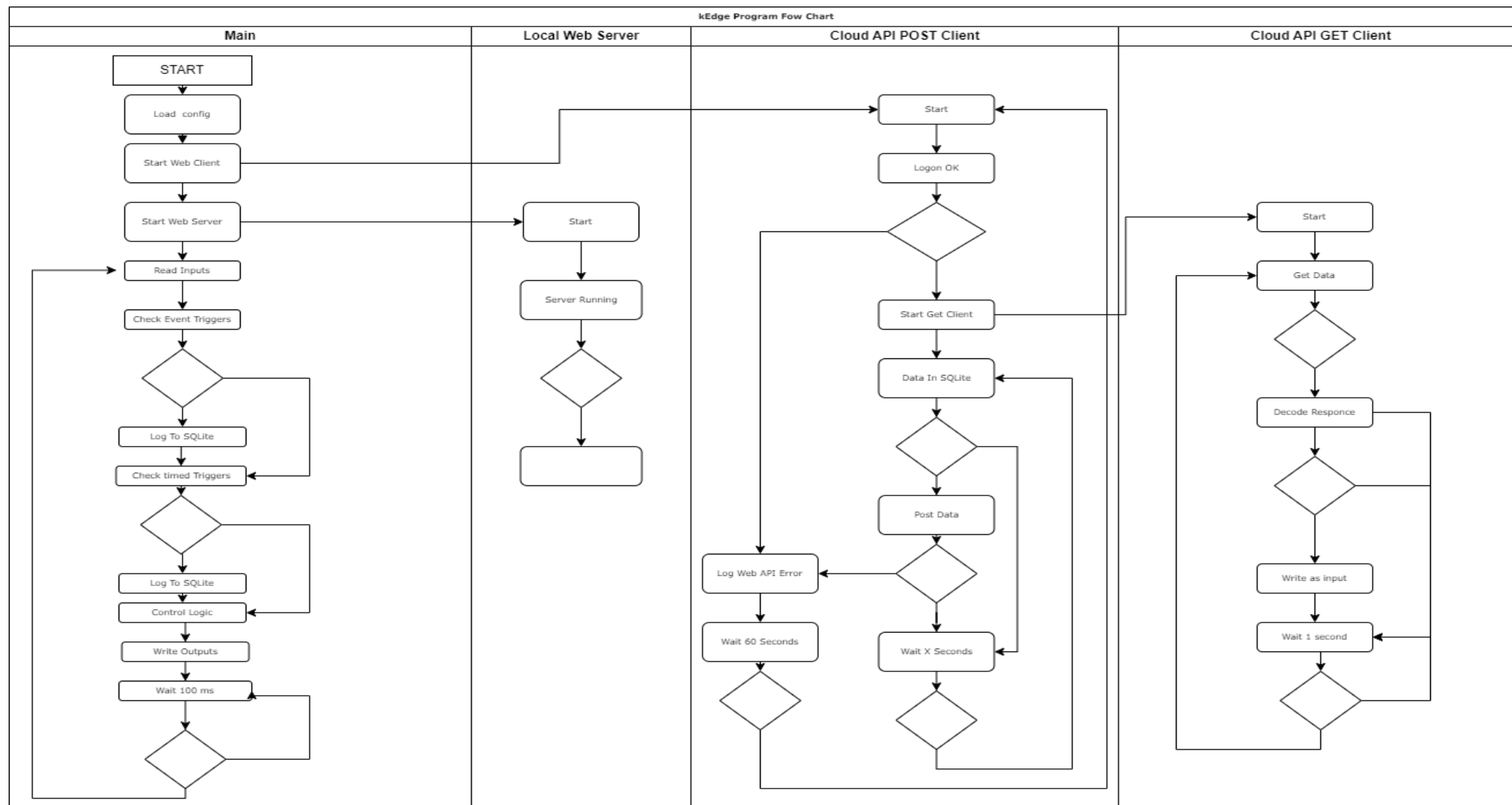
- AG, S., 2024. *Software AG Packages*. [Online]
Available at: https://www.softwareag.com/en_corporate/platform/iot/iot-analytics-platform.html#packages
- Android Developer, 2024. *NfcV Developers reference*. [Online]
Available at: <https://developer.android.com/reference/kotlin/android/nfc/tech/NfcV>
[Accessed 19 March 2024].
- Bartolo, K., 2020. *How to Document an Express API with Swagger UI and JSDoc*. [Online]
Available at: <https://dev.to/kabartolo/how-to-document-an-express-api-with-swagger-ui-and-jsdoc-50do>
[Accessed 3 Feb 2024].
- Bigelow, S. J., 2021. *What is edge computing? Everything you need to know*. [Online]
Available at: <https://www.techtarget.com/searchdatacenter/definition/edge-computing>
[Accessed 10 February 2024].
- Boucheron, B., 2023. *How To Install Node.js on Ubuntu 20.04*. [Online]
Available at: <https://www.digitalocean.com/community/tutorials/how-to-install-node-js-on-ubuntu-20-04#option-3-installing-node-using-the-node-version-manager>
[Accessed 25 February 2024].
- BROWNLEE, J., 2023. *How to Use SharedMemory in Python*. [Online]
Available at: <https://superfastpython.com/multiprocessing-sharedmemory/#Example of Using SharedMemory with Strings>
[Accessed 24 February 2024].
- CloudRAIL GmbH, 2023. *Cloud Rail Pricing*. [Online]
Available at: <https://cloutrail.com/pricing/>
- CloudRail, 2023. *CloudRail Industrial Cloud Connectivity #IIOT*. [Online]
Available at: <https://cloutrail.com/>
- Dhaduk, H., 2021. *Express vs. Hapi: The Battle For Being Best Node.js Framework*. [Online]
Available at: <https://www.simform.com/blog/express-vs-hapi/>
[Accessed 27 January 2024].
- Frappe, 2024. *Frappe Homepage*. [Online]
Available at: <https://frappe.io/>
[Accessed 2 March 2024].
- IFM Electronic, 2023. *AL1350- IO Link Master*. [Online]
Available at: <https://www.ifm.com/ie/en/product/AL1350>
[Accessed 4 January 2024].
- IFM Electronic, 2024. *Moneo the IIoT platform for industry and production*. [Online]
Available at: <https://www.ifm.com/ie/en/shared/moneo>
- IO-Link Community, 2018. *IO-Link System Description*. [Online]
Available at: https://io-link.com/share/Downloads/At-a-glance/IO-Link_System_Description_eng_2018.pdf
[Accessed 4 January 2024].
- ISO, 2019. *ISO/IEC 15693-2:2019*. [Online]
Available at: <https://www.iso.org/standard/73601.html>
[Accessed 1 April 2024].
- JonathanSanchez.Dev, 2021. *How to deploy Node Express API to an AWS EC2 Ubuntu instance*. [Online]
Available at: <https://jonathans199.medium.com/how-to-deploy-node-express-api-to-ec2-instance-in-aws-bc038a401156>
[Accessed 20 Feb 2024].
- Kunbus gmbh, 2023. *Industrial RaspberryPi for control and IIOT Projects*. [Online]
Available at: <https://revolutionpi.de/revolution-pi-serie>

- Longbottom, C., 2021. *Edge Computing Vs Cloud Computing whats the difference*. [Online]
Available at: <https://www.techtarget.com/iotagenda/tip/Comparing-edge-computing-vs-cloud-computing>
[Accessed 26 January 2024].
- MariaDB, 2024. *MariaDB Community Server*. [Online]
Available at: <https://mariadb.com/products/community-server/>
[Accessed 8 February 2024].
- MOXA, 2019. *Does Your IoT Application Need Fog Computing*. [Online]
Available at: <https://www.moxa.com/en/articles/does-your-iot-application-need-fog-computing>
[Accessed 26 January 2024].
- MySQL, 2023. *MySQL 8.3 Reference Manual - Functions That Search JSON Values*. [Online]
Available at: https://dev.mysql.com/doc/refman/8.3/en/json-search-functions.html#function_json-extract
[Accessed 28 January 2024].
- Oostdijk, J., 2024. *Open WRT- Create a sample procd init script*. [Online]
Available at: <https://openwrt.org/docs/guide-developer/procd-init-script-example>
[Accessed 21 02 2024].
- OpenWrt, 2023. *Managing services*. [Online]
Available at: https://openwrt.org/docs/guide-user/base-system/managing_services
[Accessed 21 February 2024].
- Pelakauskas, A., 2022. *How to Run Python Script as a Service (Windows & Linux)*. [Online]
Available at: <https://oxylabs.io/blog/python-script-service-guide>
[Accessed 5 January 2024].
- Pensa, C. J. T., 2021. *NFC From Scratch (With a Practical Example)*. [Online]
Available at: <https://medium.com/flux-it-thoughts/nfc-from-scratch-with-a-practical-example-ce0c7995595b>
[Accessed 19 March 2024].
- python.org, 2024. *sqlite3 — DB-API 2.0 interface for SQLite*. [Online]
Available at: <https://docs.python.org/3/library/sqlite3.html>
[Accessed 8 February 2024].
- Reitz, K., 2019. <https://docs.python-requests.org/en/latest/index.html>. [Online]
Available at: <https://docs.python-requests.org/en/latest/index.html>
[Accessed 25 March 2024].
- Roche, D., 2022. *Computer Systems and Networks Assignment IOT*. [Online]
Available at: <https://github.com/RocheDJ/TankCheck/tree/main>
[Accessed 07 February 2024].
- Roche, D. J., 2024. *HDip Mobile Development Assignment*. [Online]
Available at: <https://github.com/RocheDJ/DataViewLogger>
- Simplilearn, 2023. *Django Vs. Flask: Understanding The Major Differences*. [Online]
Available at: <https://www.simplilearn.com/flask-vs-django-article>
[Accessed 6 February 2024].
- Software AG , 2024. *Softwareaeg.com*. [Online]
Available at: https://www.softwareag.com/en_corporate/platform/iot/iot-analytics-platform.html#packages
- Software AG, 2024. *Software AG Git*. [Online]
Available at: <https://github.com/SoftwareAG>
[Accessed 6 January 2024].
- SQLite, 2024. *SQLite Homepage*. [Online]
Available at: <https://www.sqlite.org/index.html>
[Accessed 6 February 2024].
- Stokes, D., 2017. *What you need to know about JSON in MySQL*. [Online]
Available at: <https://opensource.com/article/17/5/mysql-json>
[Accessed 8 February 2024].

8.0 BIBLIOGRAPHY

- Taparia, A., 2023. *The top 10 industrial automation trends—as seen at SPS 2023*. [Online]
Available at: <https://iot-analytics.com/top-10-industrial-automation-trends/>
[Accessed 6 January 2024].
- Teltonika, 2024. *Teltonika Wiki Knowledge Base*. [Online]
Available at: <https://wiki.teltonika-networks.com/view/RUT956>
[Accessed 24 January 2024].
- Trendlog, 2024. *IIoT Devices*. [Online]
Available at: <https://www.trendlog.dk/iiot-devices/>
- urbandictionary, 2005. *Urban Dictionary*. [Online]
Available at: <https://www.urbandictionary.com/define.php?term=Scotty%20Principle>
[Accessed 2 April 2024].


APPENDIX A – KEDGE PROGRAM FLOW CHART



APPENDIX B- ETHICAL APPROVAL CHECKLIST

This Ethics Checklist must be completed for all final year undergraduate, taught postgraduate and research projects in the School of Science and Computing.

View your response(s)

 Respondent: **David Roche** (Group: CM-HDIPCS) Submitted on: Saturday, 18 November 2023, 10:47 AM

Ethics Checklist for Undergraduate, Taught Postgraduate and Research Projects in the School of Science and Computing

All students in the School of Science and Computing who are either (1) in the final year of an undergraduate/BSc degree, or (2) on a taught postgraduate/MSc programme **must complete this Ethics Checklist before conducting their project** regardless of the project type or discipline. The Checklist should also be completed by anyone (whether staff member or student) conducting a **research project** (whether programmatic or not) within the School.

The purpose of this Ethics Checklist is to **identify projects that will require formal ethical approval** from the School Research Ethics Committee, or the SETU Research Ethics Committee, before they can proceed.

Students/applicants should note that this Ethics Checklist is a **formal declaration**, and great care must be taken to **answer all questions accurately**. Students should consult with their project supervisors/advisors regarding any aspects or questions that they are unsure of before completing and submitting the Ethics Checklist.

Students/applicants must **answer all questions** presented to them until the Checklist questionnaire is completed.

Feedback Report

No human experimentation issues (UG).

No animal experimentation issues (N/A).

No issues regarding the use of human tissues.

No animal tissue or biological fluids issues.

No ionising radiation issues.

No primary data collection issues (N/A).

No underage/vulnerable people issues (UG).

No issues regarding existing/secondary data use (N/A).

No controversial data issues.

No issues related to the collection of rare or protected plants.

No issues regarding the use of genetically modified (GM) plant material.

Instructions:

1. If the above feedback is **entirely green** then, based on your answers, there is **no need to apply for ethical approval** for your project.
2. If **any** part of the above feedback is **yellow/amber**, then there is at least one issue with your project that needs to be reviewed and **you must apply for ethical approval** to continue your project.
3. If **any** part of the above feedback is **red** then there is a serious ethical issue and **you cannot continue your project** as currently planned.

It is recommended that you print this Feedback Report to a PDF file for your records. You should also forward and discuss this Feedback Report PDF with your project supervisor. They will be able to advise if you have any further questions or if you need to apply for ethical approval.

- 1 * Are you a student on a **final year undergraduate** programme, a **taught postgraduate** programme, or are you conducting a **research project**?
 - ☒ Final Year Undergraduate
 - ☐ Taught Postgraduate
 - ☐ Postgraduate Research Project
 - ☐ Other Research Project
- 2 * What is the **working title** of your project?

K Cloud –II.O.T. Ccustomizable solution for data transfer in small to medium industrial control.
- 3 * Who are the project **supervisors/advisors/principal investigators**?

Colm Dunphy / TBC
- 4 * Does your project involve **human experimentation**?

☐ Yes ☒ No
- 5 * Does your project involve **live animal experimentation**?

☐ Yes ☒ No
- (6) * Is the planned animal experimentation limited to **non-invasive procedures only** (such as feeding, weighing, or taking naturally voided faecal or hair samples), and does **not** involve any invasive procedures (such as taking rectal faecal samples or blood) from live animals?

☐ Yes ☐ No
- 7 * Does your project involve the use of **human** remains/cadavers/tissues/cells/biological fluids/embryos/foetuses?

☐ Yes ☒ No
- (8) * Do you intend to only use established **commercial human cell lines**, and no other **human** remains/cadavers/tissues/cells/biological

fluids/embryos/foetuses in your project?

☐ Yes ☐ No

9 * Does your project involve the use of **animal cells, tissues or biological fluids**?

☐ Yes ☒ No

(10) * Do you intend to only use (1) **established commercial animal cell lines**, or (2) **slaughterhouse-derived tissues/fluids**, or (3) **fluids collected as part of routine animal husbandry** (e.g. milk) and no other animal tissues or biological fluids in your project?

☐ Yes ☐ No

11 * Does your project involve the **collection of rare or protected plants**?

☐ Yes ☒ No

12 * Does your project involve the generation or use of **genetically modified (GM) plant material**?

☐ Yes ☒ No

(13) * Do you agree to (1) only use **established genetically modified (GM) plant cell lines, seeds, or plant products** in your project, (2) **not generate new plant mutations** using chemical or other means, and (3) follow specified SETU **containment and use protocols** for GM plant materials at all times?

☐ Yes ☐ No

14 * Does your project involve the use of **ionising radiation**? (e.g. use of gamma ray spectrometry)

☐ Yes ☒ No

(15) * Do you agree to carefully **follow the instructions** of the SETU designated **Radiation Protection Officer (RPO)**, and **adhere to all legal requirements** as set out in the Radiological Protection Act 1991 (Ionising Radiation) Regulations ([2019](#)), regarding the use of ionising radiation materials and equipment?

☐ Yes ☐ No

16 * Does your project involve the **collection of any new (or primary) data** from **individual people or groups**?

☐ Yes ☒ No

(17) * Does your project involve the **collection of any new (or primary) individual or group data** that is **personally or uniquely identifying**? (e.g. data about people or organisations/companies/groups that could be used to identify those individuals or groups; data collection might take any form, including internet and social media data, etc.)

☐ Yes ☐ No

(18) * Will you ensure that participants who you are collecting data from are provided with **fair warning** and must provide **explicit informed consent** for any data collected?

☐ Yes ☐ No

(19) * Will you ensure that any project-related data collection, data storage, and data use is in **full compliance** with the **EU General Data Protection Regulation (GDPR)** and the **Data Protection Act (2018)**?

☐ Yes ☐ No

(20) * Does any of the data that you intend to collect include **sensitive or private personal information** about individuals, or **commercially sensitive information** about organisations/companies/groups?

☐ Yes ☐ No

21 * Does your project involve **persons under the age of 18 years** (i.e. minors), or **any vulnerable groups**? (e.g. prisoners, refugees, those in care, addiction service users, etc.)

☐ Yes ☒ No

22 * Does your project involve the use of **existing (or secondary) human data?** (i.e. data originally collected for another purpose)

☐ Yes ☒ No

(23) * Is the existing or secondary human data you intend to use either (1) **anonymous/non-personally identifying** and in the **public domain**, or (2) available with **explicit and specific informed consent or permission** for the data to be **legally** reused in the way you intend?

☐ Yes ☐ No

(24) * Are any aspects of the primary/secondary data you intend to use for the project **controversial** in nature?

☐ Yes ☐ No

25 * Before you submit the Ethics Checklist, you must **confirm all of the following:**

- ☒ I understand that the Ethics Checklist is a formal declaration.
- ☒ I have answered all questions on the Ethics Checklist carefully and truthfully.
- ☒ The supervisor/advisor (or principal investigator) for the project is present as the Ethics Checklist is being submitted, or they have given me explicit permission to submit it in their absence.
- ☒ I have had adequate ethics training and/or instruction prior to completing the Ethics Checklist.
- ☒ I understand, and agree to abide by, the general ethical principle of "do no harm" for this project.
- ☒ I will follow the instructions given in the Feedback Report.

26 * Authentication Code (ask your project supervisor/advisor for this code)

Enter Student Number:

Enter the Authentication Code below and click "Verify Code"

Note: If an INVALID authentication code is used then this submission is NULL and VOID

8829

APPENDIX C- OPEN API DOCUMENTATION FOR BACKEND

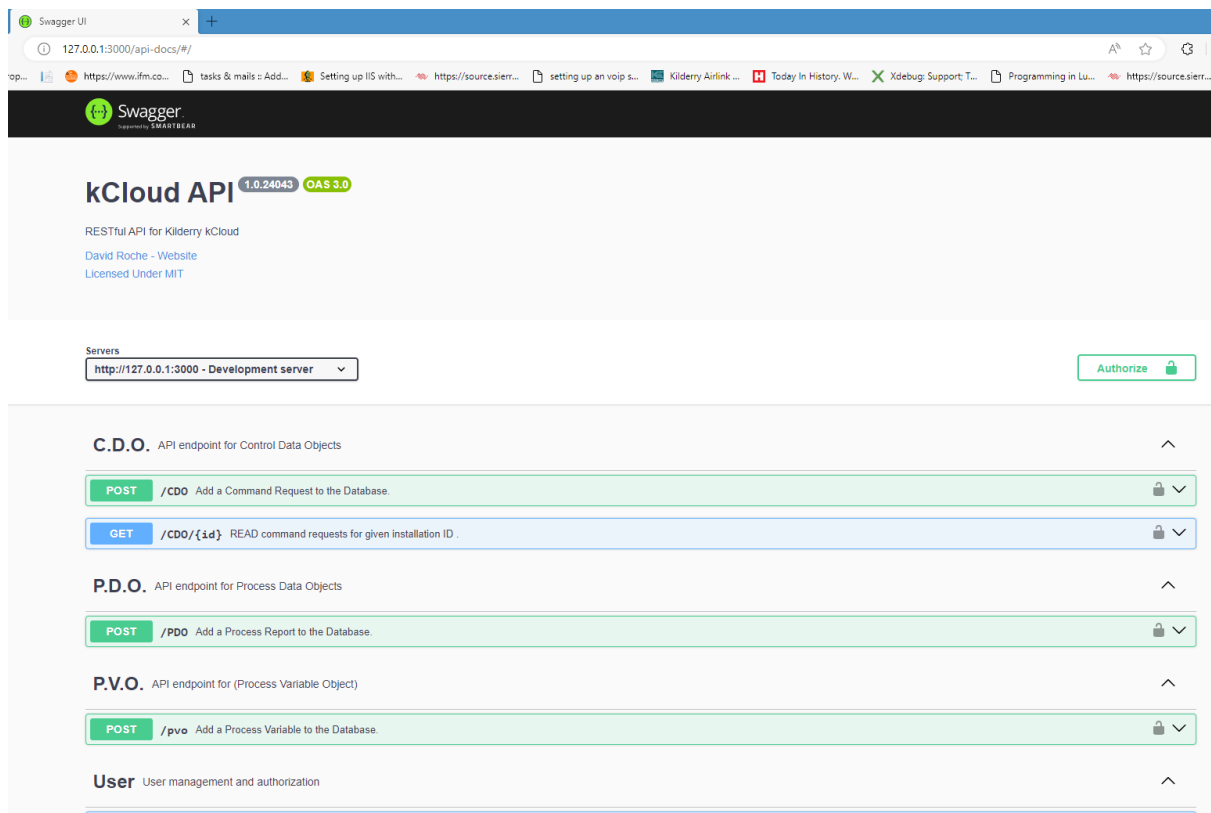


Figure 59 Swagger UI landing page.

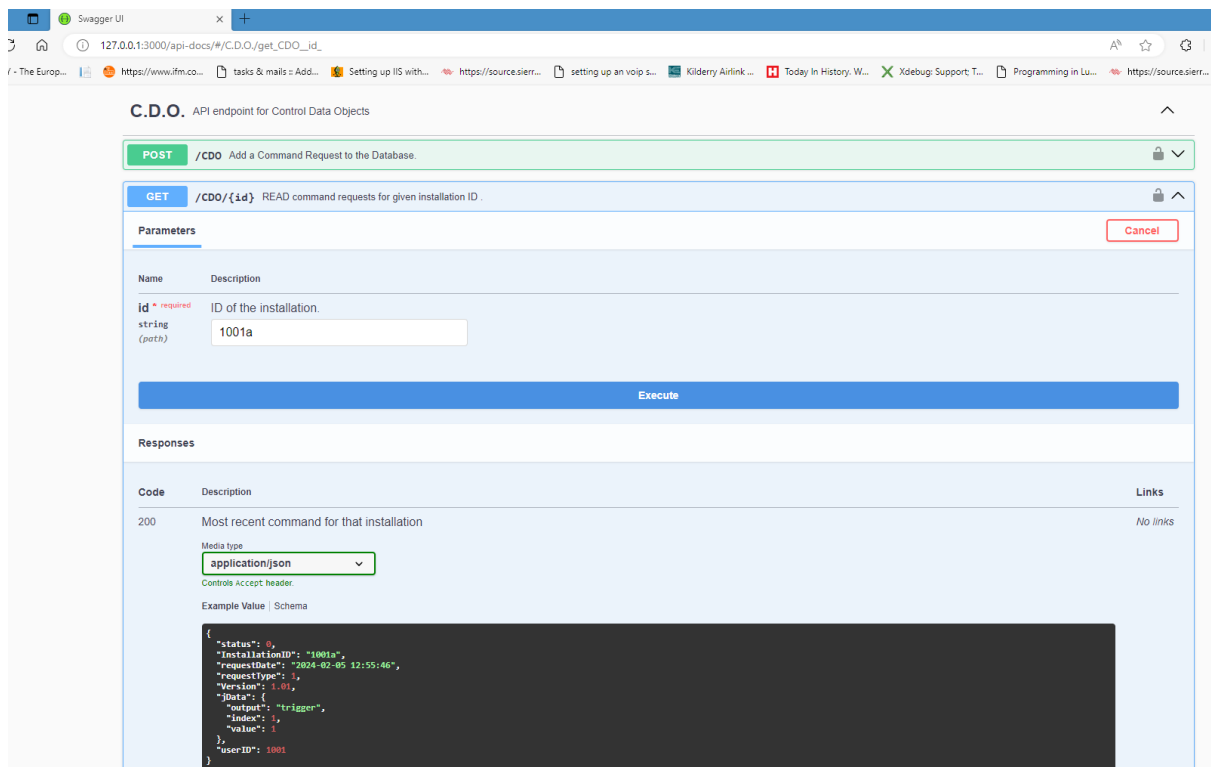
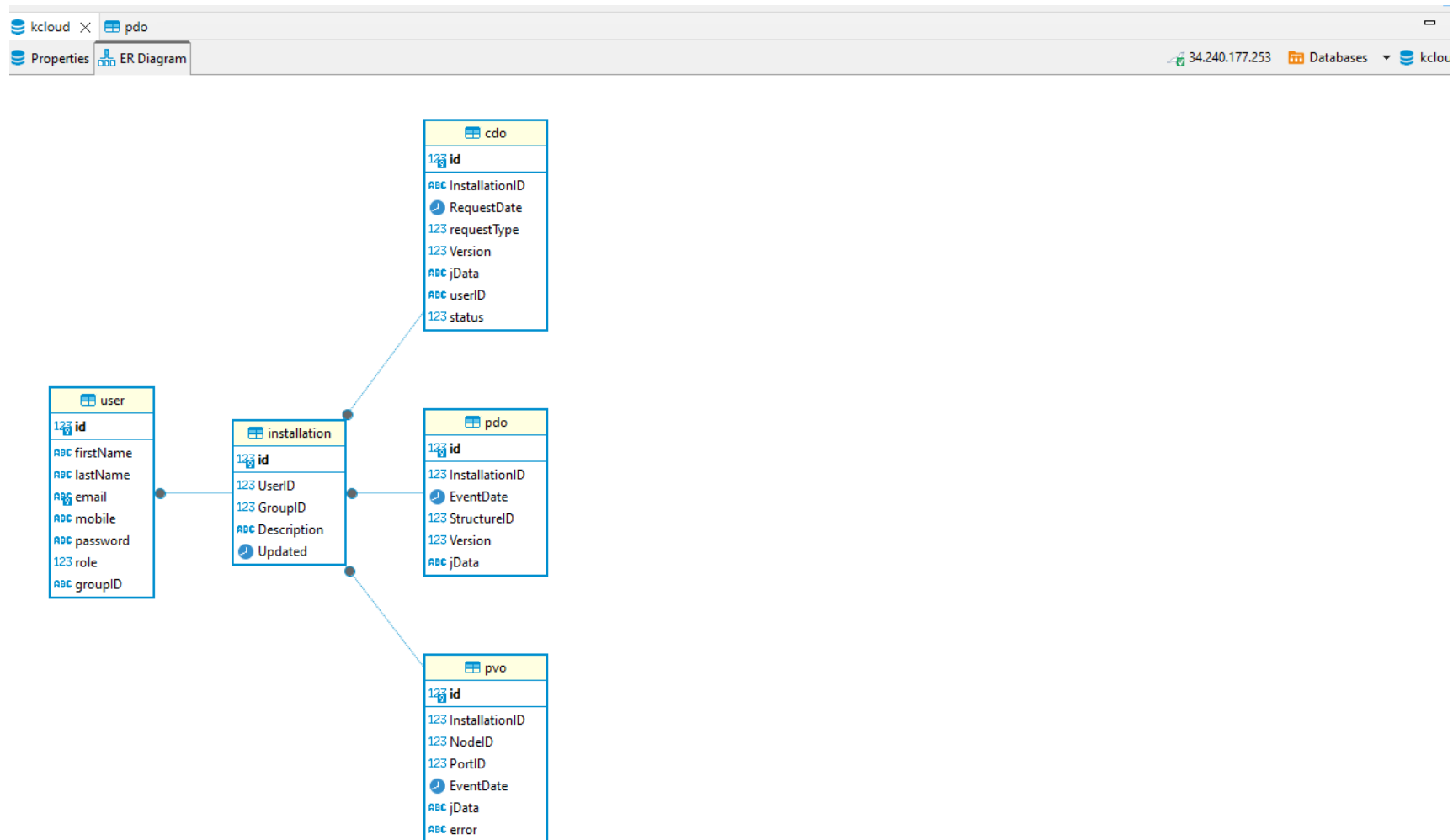


Figure 60 Example of Swagger UI testing of Command Data Object (CDO) get.

APPENDIX D- DATABASE ER DIAGRAM



APPENDIX E-TECHNOLOGIES USED AND END RESULT.

