Interreg
EUROPEAN UNION
North-West Europe
REAMIT
European Regional Development Fund

WP T2 - Deliverable 8.2

User manual for the integrated IoT, Big Data, analytics, Decision support technology

Improving Resources Efficiency of Agribusiness supply chains by Minimizing waste using Internet of Things sensors (REAMIT)

July 2023

REAMIT deliverable 8.2

European Regional Development Fund

A User Manual for the Integrated IoT, Big Data, Analytics, and Decision Support Technology

SenX | 55 rue Charles Nungesser | 29490 Guipavas | senx.io

Table of content

Introduction

The REAMIT project aims to tackle the critical challenge of reducing food wastage in agribusiness supply chains by leveraging advanced technologies such as IoT, Big Data, analytics, and decision support systems. A key component for the project's success lies in the adoption of a robust time series platform, which offers numerous benefits in effectively addressing food wastage.

By utilizing a time series platform, we can explore patterns, trends, and anomalies associated with food wastage over time. Through the collection and analysis of data from IoT sensors deployed throughout the supply chain, valuable insights can be gained regarding the factors contributing to waste. These insights, when valuable and interpretable, enable informed decision-making and the implementation of targeted interventions. The real-time monitoring capabilities of a time series platform facilitate continuous data collection and processing from IoT sensors, enabling timely detection of potential issues or deviations from optimal conditions. This proactive approach allows for preventive measures to be taken to prevent food wastage before it occurs. Furthermore, a time series platform enables the utilization of historical data to forecast future patterns and identify potential sources of food wastage.

To answer its objectives, the REAMIT project deployed a Warp 10 time series platform for developing experimental applications and conducting studies. Warp 10 offers scalability to handle the large volumes of time series data generated by IoT sensors, ensuring efficient storage, processing, and analysis. With its powerful scripting language called Warpscript, extensive functionality is provided for manipulating and analyzing time series data. This empowers users to implement complex algorithms and extract valuable insights. The WarpView components and Discovery dashboards complement these capabilities by providing an interactive interface for visualizing and exploring time series data, facilitating easier interpretation and decision-making.

This manual provides detailed instructions for setting up a Big Data platform to develop applications that address the challenges of reducing food wastage in agribusiness supply chains.

1. Platform Setup

The preferred method for installing the Warp 10 platform is to use the latest binary package provided on the download page of the Warp 10 website. Bare-metal installation provides optimal performance. This approach is recommended for standalone servers or edge devices such as a Raspberry Pi. By installing natively, you can take full advantage of the system's resources.

Alternatively, Docker can be used for a faster and easier installation process. It is also important to note that on Windows servers, only docker installation is supported.

A. Bare-metal installation

1. Download and Extract Warp 10

Download the latest tar.gz file from either the official GitHub repository or the download page of the Warp10 website. The following commands assume you are using a Debian/Ubuntu-based test machine, which can be a virtual machine.

```
export W10VERSION=3.y.z   # change version number here
tar xf warp10-$W10VERSION.tar.gz
cd warp10-$W10VERSION
sudo useradd -m -d /opt/warp10 warp10
sudo mv * /opt/warp10
sudo chown -R warp10:warp10 /opt/warp10
```

Verify if Java is installed. Warp 10 supports Java 8+ (up to Java 17 LTS).

```
java -version   # version must be in range [1.8, 17]
```

If Java is not installed, install it using your package manager:

```
sudo apt-get install openjdk-17-jre
```

For easier manipulation during tests, add yourself to the `warp10` group and set group permissions for the folders you will modify:

```
sudo usermod -a -G warp10 $USER
sudo chmod -R g+rw /opt/warp10/etc /opt/warp10/macros
/opt/warp10/runners
sudo chmod g+x /opt/warp10/etc
```

Logout or reboot your system to apply the changes.

The Warp 10 installer will attempt to automatically detect your Java path. If you have a custom path or multiple Java versions installed, you need to explicitly define `JAVA_HOME` in `/opt/warp10/etc/warp10-env.sh`.

2. Bootstrap Warp 10

This will set up the environment for the Warp 10 platform:

Generate the standalone configuration file from templates.
Generate an empty LevelDB structure.
Start the bootstrap process as the `warp10` user:

```
sudo su warp10
/opt/warp10/bin/warp10.sh init standalone
```

The bootstrap process should end with the message "Warp 10 configuration has been generated in [path to configuration files]."
While exploring Warp 10, you may encounter certain limits that can be adjusted in the Warp 10 configuration, such as the maximum number of operations or the number of data points in a FETCH. All the Warp 10 configurations are located in `/opt/warp10/etc/conf.d/`.

3. Start Warp 10

To start Warp 10, run the following command:

```
/opt/warp10/bin/warp10.sh start
```

The Warp 10 standalone configuration files (`*.conf`) have been generated in the `etc/conf.d/` directory.
Logs are stored in the `logs` directory.
Data is stored in LevelDB, which can be found in the `leveldb` directory.
You can perform snapshots of the LevelDB data using the init script.

B. Containerized installation

An alternative way to set up the Warp 10 standalone platform is by using Docker. Official Docker builds are available on Docker Hub, which include the following components:

- Warp 10: The platform for storing and analyzing Geo Time Series data.
- WarpStudio: A web application that allows users to interact with the Warp 10 platform.
- Sensision: A service for monitoring Warp 10 platform metrics.

1. Start Warp 10

To start a Warp 10 instance using Docker, run the following command, specifying the desired Warp 10 version by replacing \`tag\`:

```
docker run -d -p 8080:8080 -p 8081:8081 warp10io/warp10:tag
```

This command starts the Warp 10 instance and binds the external ports \`8080\` for Warp 10 and \`8081\` for WarpStudio.

You can use environment variables to adjust the JVM heap size:

- Initial heap size (\`Xms\`): Use \`WARP10_HEAP\` variable.
- Maximum heap size (\`Xmx\`): Use \`WARP10_HEAP_MAX\` variable.

By default, the configuration is \`WARP10_HEAP=1g\` and \`WARP10_HEAP_MAX=1g\`. To modify the heap size, specify the desired values:

```
docker run -d -p 8080:8080 -p 8081:8081 -e WARP10_HEAP=8g -e
WARP10_HEAP_MAX=8g warp10io/warp10:tag
```

2. Mapping Volumes for Persistence

To ensure data persistence and configuration retention even if the Docker container is deleted, it's recommended to map a volume to the container's \`/data\` folder. Use the \`--volume\` option with the desired local folder path:

```
docker run -d -p 8080:8080 -p 8081:8081 --volume=/var/warp10:/data
warp10io/warp10:tag
```

In this example, the container's internal data folder `/data` is mapped to the local folder `/var/warp10`. Remember to use the same `--volume` option in all your other Docker commands for the Warp 10 image.

3. For Windows Users

If you are using Windows, ensure that Docker is installed and optionally DockStation. To start a Warp 10 instance on Windows, use the following command, specifying the volume mapping:

```
docker run --volume=c:\\warp10:/data -p 8080:8080 -p 8081:8081 -d -i
warp10io/warp10:tag
```

2. Data Ingestion

A. Generate Warp 10 Tokens

The Warp 10 platform employs a robust security model that enables precise control over data read and write permissions. This model revolves around the concepts of data producers, data owners, applications, and tokens for WRITE and READ operations.

To generate Warp 10 tokens, you can utilize the `warp10.sh` script along with a token envelope, which is a WarpScript stored in `warp10/tokens/`. A demo envelope, located at `tokens/demo-tokengen.mc2`, is provided, which generates a set of test tokens valid for 14 days.

Execute the following command:

```
./bin/warp10.sh tokengen tokens/demo-tokengen.mc2
```

Or, if you are using docker:

```
docker exec -u warp10 -it <container> ./bin/warp10.sh tokengen
tokens/demo-tokengen.mc2
```

The output will look like:

```
[{
  "ident" : "4f8c4edc7a42d112",
  "id" : "DemoWriteToken",
  "token" :
"hQfwWAHcJpHBHgpz2531HtcKAknSzpam6XzbfMcwBvaDZLZM6uWDD2xNF8kGUitE5UzUEWp
SdYQQXUQ8VQZc.De9kpFaERFMvbYxD7btd2i6UnyzfTni1F"
},{
  "ident" : "5d99a69053407ad2",
  "id" : "DemoReadToken",
  "token" :
"Lj.suW2XZru53seb.y7QulvE_qJ4CyMMEfaz6T.ozaBx79RJ1m1m_c.Bj7.jEGlM2C7MRlS
CpKgaCwVXEen6nZOo9KgbC5IakZTc5RvZ3qZX_mq_VXm9vNkANB5J7uWJExgGK6Ht9kAL1W_
1tBR1Dl2H9XJiWXKCmz3pYxJv7p7"
}]
```

Remember to save these demo tokens for future steps. You can also generate your own WRITE and READ tokens using the TOKENGEN command or extend the lifespan of existing tokens. For detailed information, refer to the warpscript TOKENGEN function documentation.

B. Prepare Data in GTS Data Format

Data is transmitted to the Warp 10 platform through HTTP POST requests to the Warp 10 API.

API Endpoint

The HTTP endpoint for sending data is `http(s)://host:port/api/vX/update`, where `vX` represents the version of the API you wish to use (currently `v0`). To authenticate requests and be accepted by the platform, include an `X-Warp10-Token` HTTP header with your write token.

Data Format

Data needs to be converted into the GTS input format.

- Each GTS (Geo Time Series) represents a distinct time series.
- A GTS is defined by its CLASS and LABELS values.

It is important to choose meaningful names and labels
Data is included in the body of the POST request, with one data point per line. Each line adheres to the GTS input format:

```
TS/LAT:LON/ELEV CLASS{LABELS} VALUE
```

Where:

- TS (optional): Timestamp of the reading in microseconds since the Unix Epoch. If omitted, the platform assigns the timestamp upon receiving the data point.
- LAT:LON (optional): Geographic coordinates of the reading, using the WGS84 format.
- ELEV (optional): Elevation of the reading in millimeters.
- CLASS: Class name of the reading, encoded as a URL-encoded UTF-8 character string. The character `{` (Unicode LEFT CURLY BRACKET, 0x007B) must be encoded.
- LABELS: Comma-separated list of labels, following the syntax `key=value` where both the key and value are URL-encoded UTF-8 character strings. If a key or value contains the characters `,` (Unicode COMMA, 0x002C), `}` (Unicode RIGHT CURLY BRACKET, 0x007D), or `=` (Unicode EQUALS SIGN, 0x003D), those characters must be encoded.
- VALUE: The value of the reading. It can be one of four types: LONG, DOUBLE (using a dot as the decimal separator), BOOLEAN (character T or F), or STRING (anything URL-encoded between single quotes). Since Warp 10 2.1, binary or multi-value data is also supported.

For example, in the REAMIT pilot test HMF, each sensor records temperature and bag status. Here's how the last 5 readings of two separate sensors are formatted:

```
1666797831000000// HMF.temperature{device_name=Sensor%206514,device_id=355523762896514} 23.22
=1666711435000000// 21.44
=1666625043000000// 21.48
=1666538651000000// 22.07
=1666452259000000// 23.32
1666788099000000// HMF.temperature{device_name=Sensor%206974,device_id=355523768866974} 14.74
=1666701706000000// 12.84
=1666615314000000// 13.66
=1666528922000000// 14.32
=1666442529000000// 13.58
1672326906000000// HMF.bag_status{device_name=Sensor%206514,device_id=355523762896514} T
=1672240514000000// T
=1672154122000000// T
=1672067731000000// T
=1671981339000000// T
1672317239000000// HMF.bag_status{device_name=Sensor%206974,device_id=355523768866974} F
=1672230846000000// F
=1672144453000000// F
=1672058060000000// F
=1671971667000000// F
```

Note: It is recommended to group data points from the same series in subsequent lines to optimize parsing speed (use the = symbol at the start of a new line to indicate the data point belongs to the same series).

C. Pushing Data

Configuring Warp 10 for Network Access

By default, Warp 10 listens only on the loopback address (127.0.0.1). If you intend to access Warp 10 from another machine on your network, you need to modify the `standalone.host` configuration to listen on any interface (0.0.0.0 on Linux). Edit the configuration file `etc/conf.d/00-warp.conf` using your preferred text editor and restart Warp 10.

Pushing Data with cURL

If you're using a Unix system, you can push data to Warp 10 using cURL. The Warp 10 platform listens on port 8080 within your container. In our setup, we have mapped this port to port 8080 on the host machine, so you can make your requests to `127.0.0.1:8080`.

1. Create a text file named "example.gts" using your preferred text editor and format the data as described in the previous section.

2. Push the file to Warp 10 using cURL (replace `WRITE` with your write token, and modify the URL if necessary):

```
curl -v -H 'Transfer-Encoding: chunked' -H 'X-Warp10-Token: WRITE' -T
example.gts 'http://127.0.0.1:8080/api/v0/update'
```

Note 1: The `chunked` option is useful for pushing large files that don't fit in RAM.

Note 2: If you are using Docker, by default, there is no binding between the Warp 10 API address and the host machine (Docker runs through a virtual machine). To access Warp 10, you need to replace `127.0.0.1` with the IP address of the container.

Pushing Multiple GTS Files

You can push multiple GTS files in a single cURL command by piping the output of `cat` to a chunked cURL request:

```
cat gts/*.gts | curl -T - -H 'Transfer-Encoding: chunked' -H
'X-Warp10-Token: myWriteToken' 'http://127.0.0.1:8080/api/v0/exec'
```

Pushing Data with Python

Alternatively, this small python program will take GTS text files as arguments, and will push them using the requests library (replace WRITE with your write token).

```
#!/usr/bin/env python
#  -*- coding: utf-8 -*-
import sys
# sudo apt install python-pip
# sudo pip install requests
import requests

if len(sys.argv) < 2:
    print ("Use: " + sys.argv[0] + " *.gtstxt")
    exit()

for filename in sys.argv[1:]:
    print("opening " + filename)
    GTSfile = open(filename, 'rb').read()
    headers = {"X-Warp10-Token": "WRITE"}
    url = "http://127.0.0.1:8080/api/v0/update"
    print(" push data to warp10... " + url)
    r = requests.post(url, headers=headers, data=GTSfile)
    print(" warp10 server answer " + str(r.status_code))
```

3. Building Applications

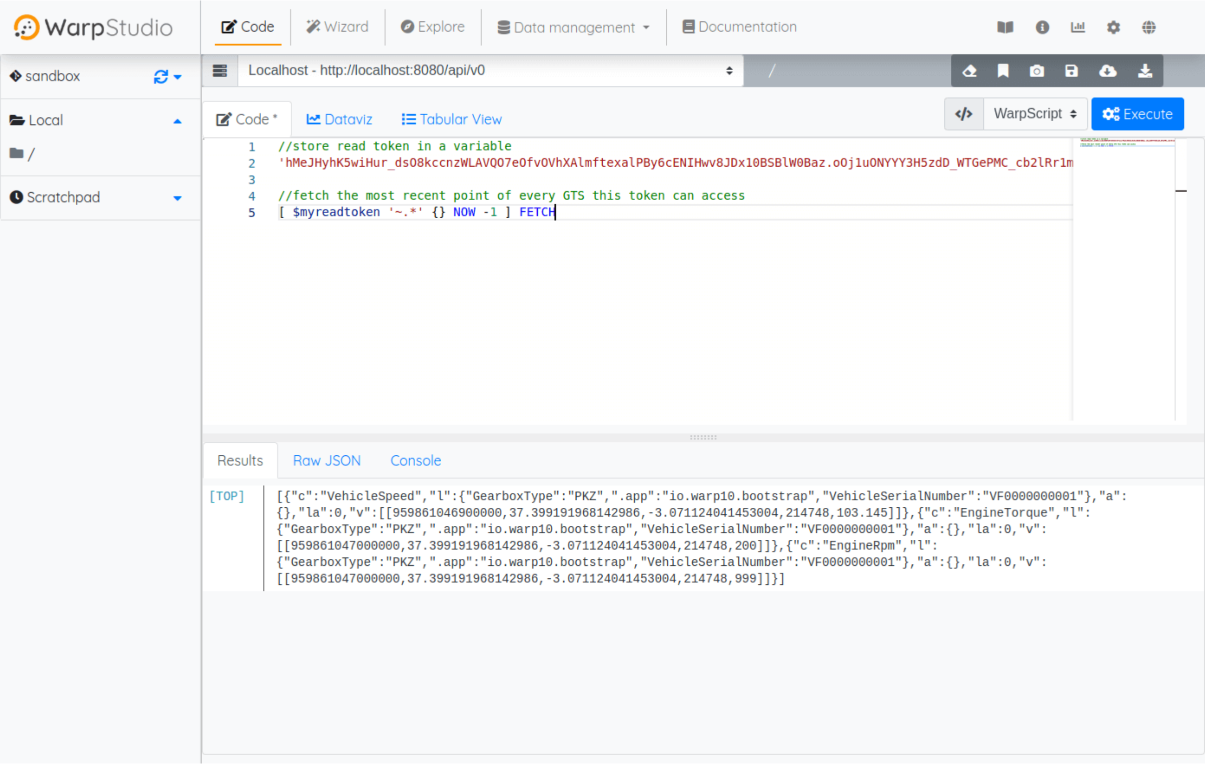
A. Enters WarpScript

Once backend configuration is done, you can execute WarpScript on your Warp 10 instance. For example, you can read data you pushed in the previous section using this curl command sending a warpscript instructing to fetch them (replace READ):

```
curl -v --data-binary "[ 'READ' '~.*' {} NOW -1 ] FETCH"
'http://127.0.0.1:8080/api/v0/exec'
```

If everything is working correctly, you should receive an HTTP 200 OK response with the most recent data point of each Geo Time Series (GTS) in JSON format.

The Warp 10 ecosystem offers several options to send WarpScript requests and interact with it in an IDE. The most straightforward approach is to use the online editor called WarpStudio, accessible at http://studio.senx.io.



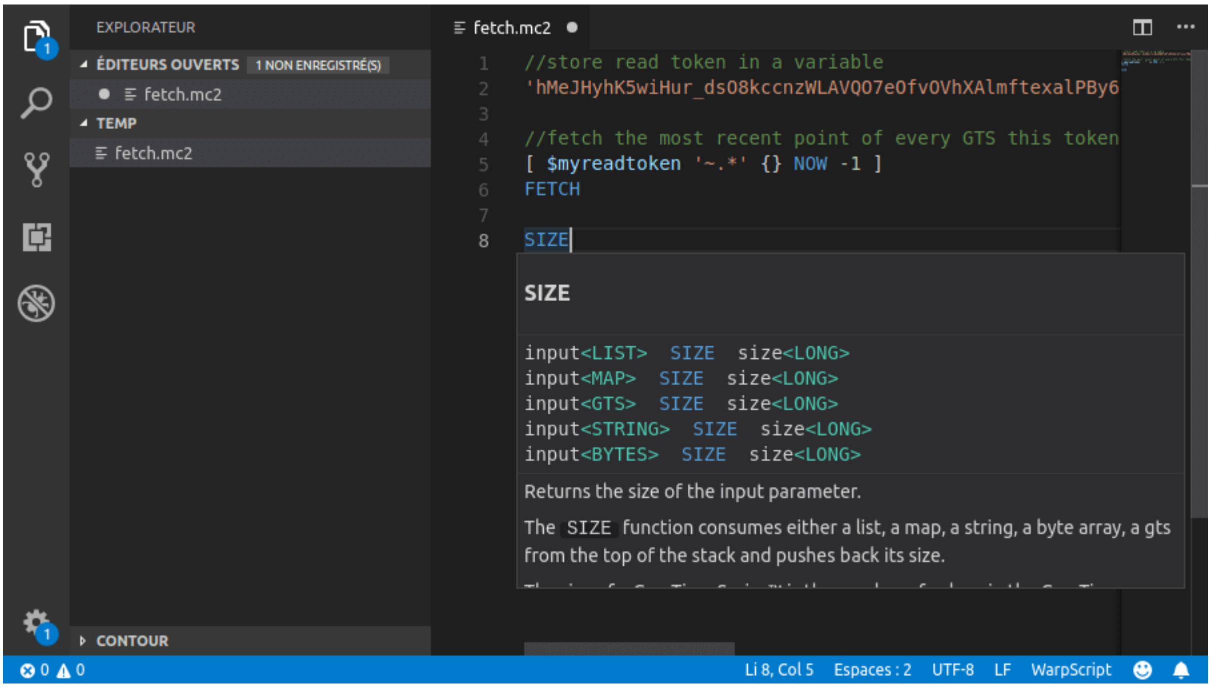
```
//store read token in a variable
'hMeJHyhK5wiHur_ds08kccnzWLAVQ07eOfvOVhXAlmftexalPBy6cENIHwv8JDx10BSBlW0Baz.oOj1uONYYY3H5zdD_WTGePMC_cb2lRr1m

//fetch the most recent point of every GTS this token can access
[ $myreadtoken '~.*' {} NOW -1 ] FETCH
```

```
[{"c":"VehicleSpeed","l":{"GearboxType":"PKZ",".app":"io.warp10.bootstrap","VehicleSerialNumber":"VF0000000001"},"a":
{},"la":0,"v":[[959861046900000,37.399191968142986,-3.071124041453004,214748,103.145]]},{"c":"EngineTorque","l":
{"GearboxType":"PKZ",".app":"io.warp10.bootstrap","VehicleSerialNumber":"VF0000000001"},"a":{},"la":0,"v":
[[959861047000000,37.399191968142986,-3.071124041453004,214748,200]]},{"c":"EngineRpm","l":
{"GearboxType":"PKZ",".app":"io.warp10.bootstrap","VehicleSerialNumber":"VF0000000001"},"a":{},"la":0,"v":
[[959861047000000,37.399191968142986,-3.071124041453004,214748,999]]}]
```

Additionally, there are other tooling options available, with the most popular being Visual Studio Code (VSCode) with the WarpScript extension.

Both WarpStudio and the WarpScript extension in VSCode provide features such as code highlighting, script auto-completion, and snippet templating, which enhance the speed and convenience of writing WarpScript code.



```
//store read token in a variable
'hMeJHyhK5wiHur_ds08kccnzWLAVQ07e0fv0VhXAlmftexalPBy6

//fetch the most recent point of every GTS this token
[ $myreadtoken '~.*' {} NOW -1 ]
FETCH

SIZE
```

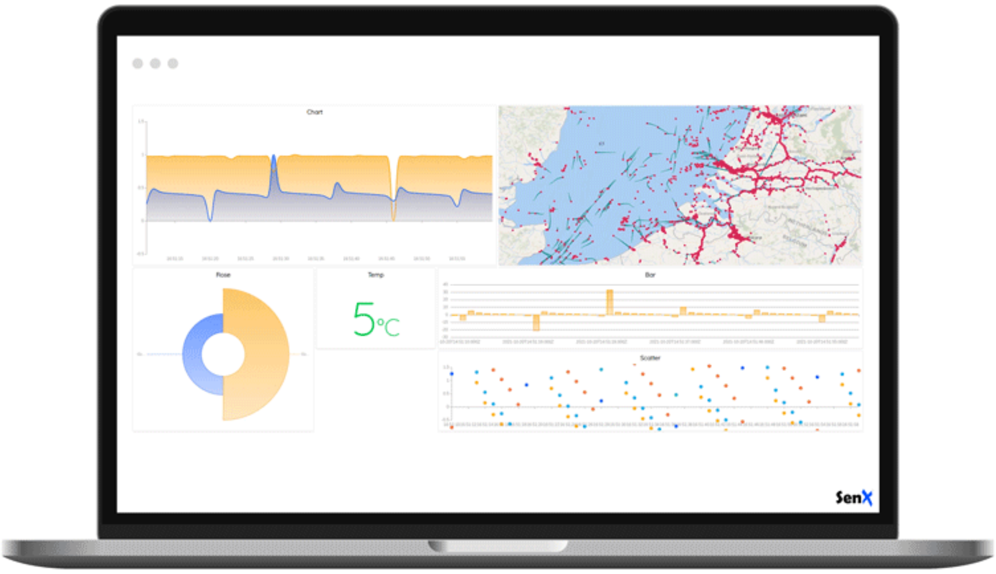
SIZE

```
input<LIST>  SIZE  size<LONG>
input<MAP>  SIZE  size<LONG>
input<GTS>  SIZE  size<LONG>
input<STRING>  SIZE  size<LONG>
input<BYTES>  SIZE  size<LONG>
```

Returns the size of the input parameter.

The `SIZE` function consumes either a list, a map, a string, a byte array, a gts from the top of the stack and pushes back its size.

Moreover, these tools integrate visualization capabilities, allowing you to render charts and leverage WarpView components and Discovery's dynamic dashboards. This means you can visualize and analyze data within the same environment.



Furthermore, WarpScript JSON outputs, WarpView components, and Discovery dashboards can be seamlessly integrated into application development.

B. WarpScript Capabilities

Warpscript is not solely designed for querying data; it is a comprehensive programming language in itself. Its internal data representations are specifically optimized for working with time series data, making it a powerful tool for time series analytics. In addition to its querying capabilities, Warpscript offers a wide range of functionalities and libraries that enable efficient and advanced time series analysis.

Below we give an overall presentation of features that can be of interest for this report. Details can be found in the official WarpScript documentation that is available online.

Warpscript encompasses several common processing frameworks that are widely used in time series analysis. These frameworks include:

1. Time Synchronization: Warpscript provides functionalities to synchronize time series data by aligning timestamps and ensuring consistency across multiple time series. This allows for accurate comparisons and analysis of data points across different time intervals. See BUCKETIZE.

2. Sliding Window: With Warpscript, you can perform computations and calculations using sliding windows. This framework enables the evaluation of data within a specified time window, facilitating tasks such as aggregations, moving averages, and statistical calculations. See MAP.

3. Multivariate Analysis: Warpscript supports multivariate analysis, allowing for the examination and exploration of relationships between multiple time series. This framework enables the detection of dependencies, correlations, and patterns among different variables, providing insights into complex time series datasets. See REDUCE.

In addition to these common processing frameworks, Warpscript offers advanced functionalities that further enhance time series analysis:

1. Anomaly Detection: Warpscript provides techniques for detecting anomalies and outliers within time series data. These include statistical methods, machine learning algorithms, and threshold-based approaches, which can identify unusual patterns or deviations from expected behavior. See #Outlier.

2. Forecasting: Warpscript includes forecasting capabilities, enabling the prediction of future values based on historical time series data. With official supported extensions, it provides access to various forecasting models, such as ARIMA, Exponential Smoothing, and Prophet, allowing you to generate accurate forecasts and make informed decisions. See Warp10-ext-forecasting.

3. Signal Processing: Warpscript incorporates a wide range of signal processing functions, allowing for tasks such as smoothing, filtering, denoising, and feature extraction. These capabilities enable the analysis and manipulation of time series data to reveal underlying patterns and extract relevant information. See #Gts.

4. Machine Learning: Warpscript seamlessly integrates into machine learning pipelines as a powerful machine learning enabler. Its strength lies in its ability to efficiently handle data preparation during the learning phase and enable fast inferences that can be performed in proximity to the input data. Moreover, Warpscript boasts extensive compatibility with the Python ecosystem and popular machine learning frameworks such as ONNX and PMML. See Python tooling, warp10-ext-pmml, warp10-ext-onnx.

5. Geospatial Analysis: Warpscript includes geospatial analysis capabilities, allowing for the integration of location-based information with time series data. This framework facilitates tasks such as geospatial aggregation, spatial-temporal analysis, geofencing, and distance calculations, enabling insights from spatiotemporal datasets. See #Geo.

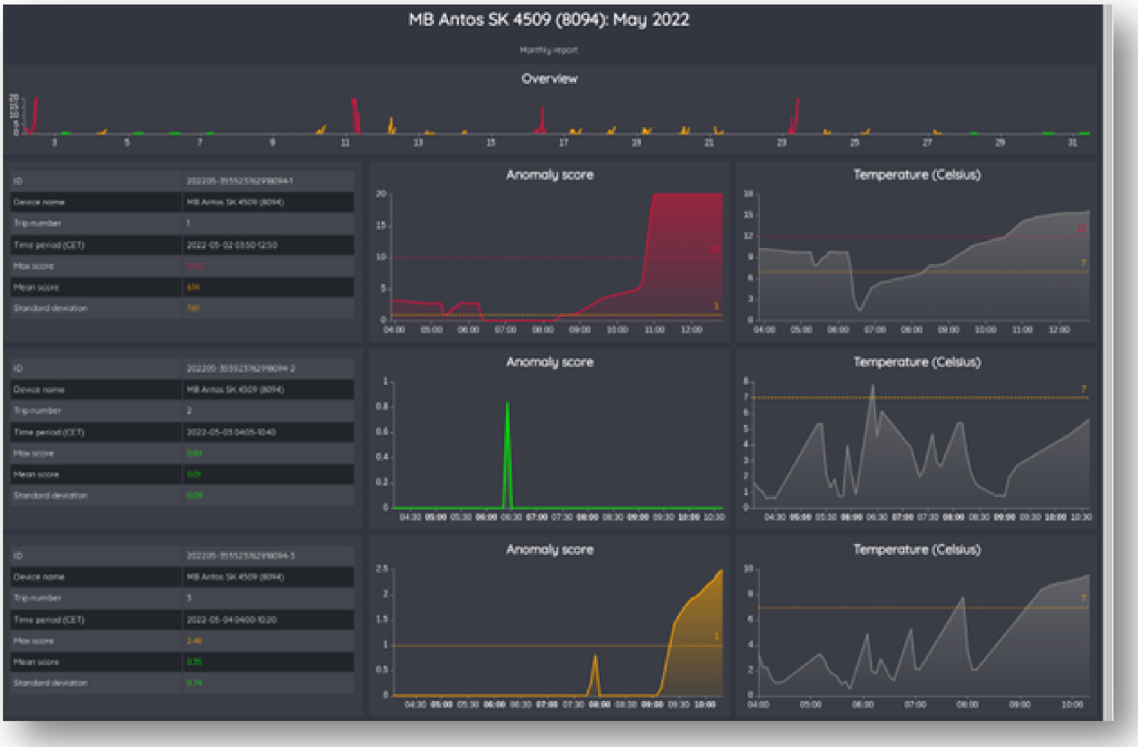
C. Examples of REAMIT Analytics Applications

1. Detect refrigeration failures across transportation trips

Biogros, a prominent organic food supplier in Luxembourg, has partnered with Reamit to study the ability to enhance their transportation operations. Sensors were strategically installed in their transportation trucks to monitor environmental conditions, including temperature and humidity. Leveraging the power of Reamit's Warp10 platform, the sensor data was ingested in Geo Time Series format for efficient storage and analysis.

To detect anomalies in the sensor readings and assess trip quality, we developed a trip quality score formula. This formula, implemented in WarpScript, intelligently processes the temperature data to generate curated anomaly scores for each sensor. By doing so, refrigeration failures during transportation can be identified and addressed.

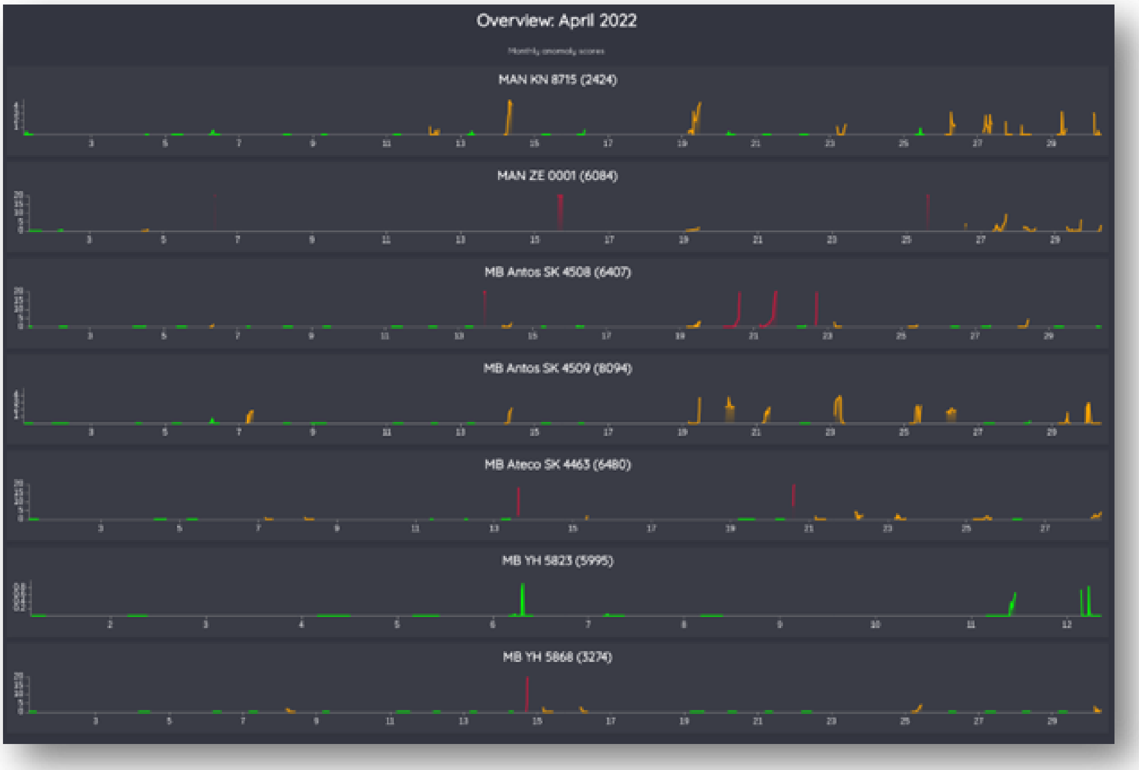
To visualize and report on the quality of their transportation, we deployed a set of dynamic dashboards using WarpScript code. Two examples are depicted below.



ID	[illegible]
Device name	MB Antos SK 4509 (8094)
Trip number	1
Time period (CET)	2022-05-02 03:50-12:50
Max score	[illegible]
Mean score	[illegible]
Standard deviation	[illegible]

ID	[illegible]
Device name	MB Antos SK 4509 (8094)
Trip number	2
Time period (CET)	2022-05-03 04:05-10:40
Max score	0.84
Mean score	0.01
Standard deviation	0.09

ID	[illegible]
Device name	MB Antos SK 4509 (8094)
Trip number	3
Time period (CET)	2022-05-04 04:00-10:20
Max score	2.48
Mean score	0.35
Standard deviation	0.74

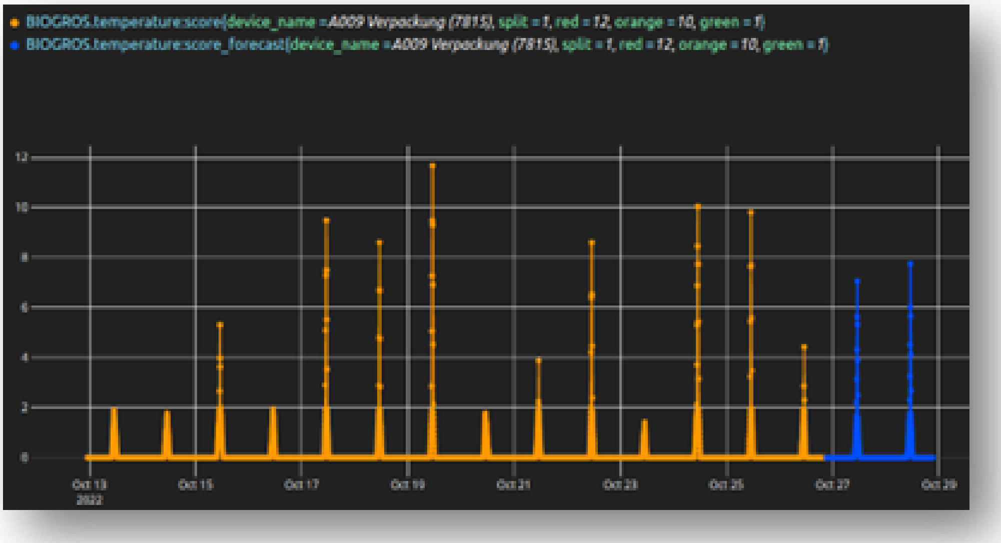


These dashboards, rendered by Reamit's Discovery, provide comprehensive insights into the transportation process. Operators can easily interact with the dynamic graphs to explore and analyze the data, enabling quick decision-making and continuous improvements.

2. Forecast temperatures in cold room storage facilities

Furthermore, we extended the same pipeline to include static sensors in cold room storage. By applying the anomaly detection formula to these sensors, we were able to identify deviations from the expected temperature ranges and detect anomalies in storage conditions.

Additionally, due to the stationarity of the data collected from static sensors, we capitalized on this property to develop forecast models for each sensor. Leveraging the same anomaly scoring formula, we incorporated it into the forecast models to predict potential issues in storage conditions. An exemplary result of the forecasting models is illustrated in the provided screenshot of a VsCode tab using WarpScript extension. It reveals a distinct seasonal pattern in the forecasted data.



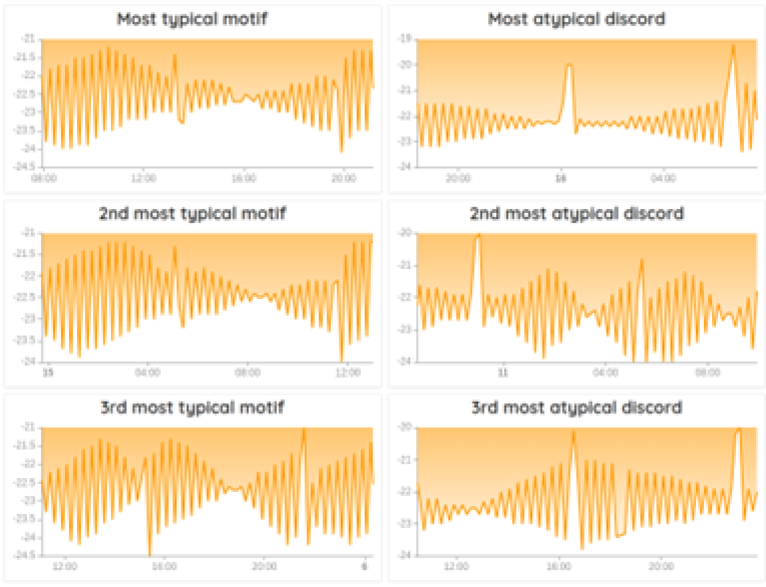
The forecast results were shared with Biogros. The correlation between the forecasted patterns and real-life operations is helpful for Biogros in optimizing their storage practices and ensuring the preservation of organic food products at optimal temperatures.

3. Discover interpretable patterns by matrix profiling temperature data

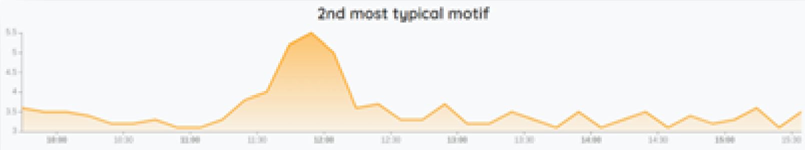
Yumchop is a pioneering company specializing in food preparation and catering, offering a unique automated vending machine solution. Yumchop Foods' innovative approach provides a hygienic and completely contactless Grab & Go alternative to conventional catering services.

Yumchop joined forces with Reamit, sharing their data to explore and uncover potential benefits. Initially, the focus was on anomaly detection, but limited interpretability was found in the identified anomalies. Thus, the focus shifted towards developing tools to effectively describe the data.

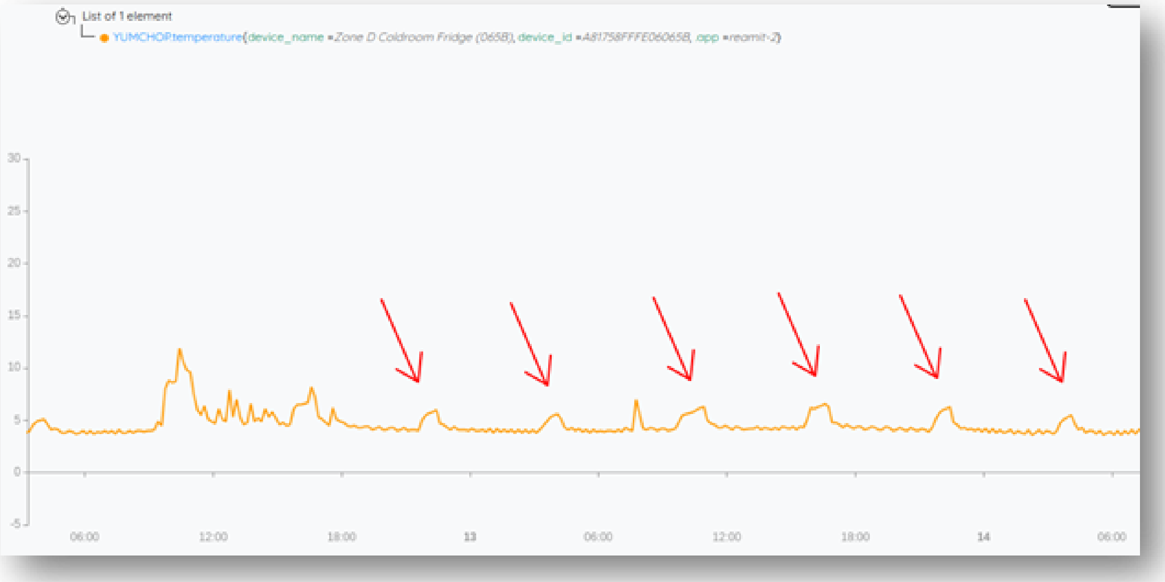
To achieve this, an unsupervised data exploration method was adopted, utilizing statistical features and leveraging WarpScript to identify seasonal cycles. With appropriate time bucketization and cycle length, a matrix profiling algorithm was applied to the time series data, enabling the detection of recurring patterns. See an output example below.



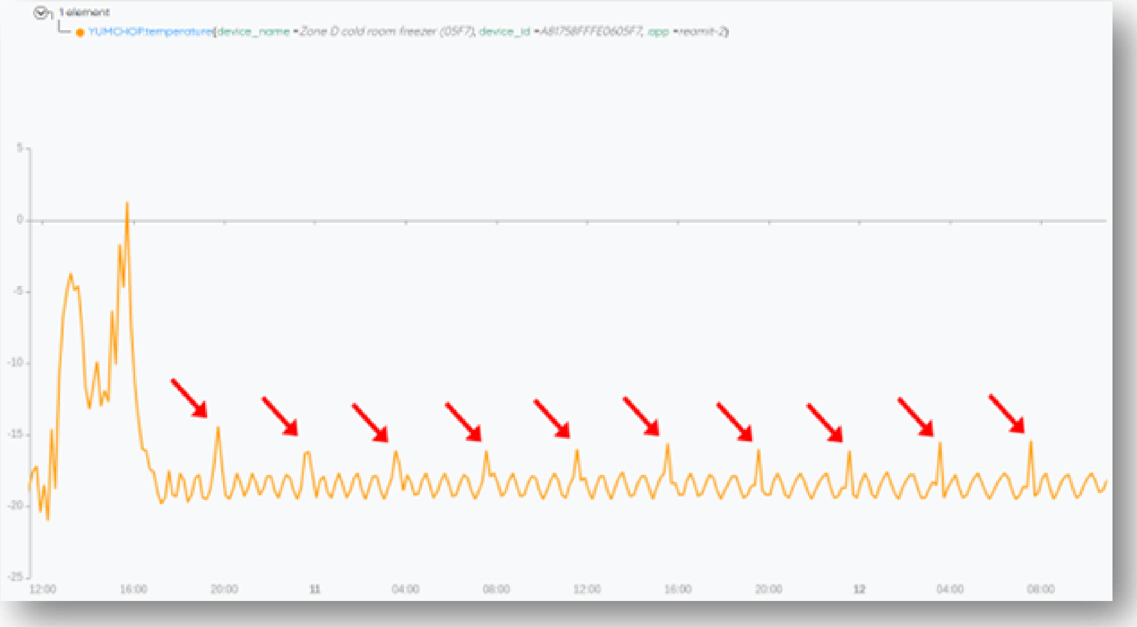
For sensors placed in refrigerators, we noticed that the second pattern in the matrix profile consistently indicated a defrost operation. It is depicted below.



Based on the data analysis (as shown in the WarpStudio screenshot below), defrost operations typically occur every 6 hours under normal circumstances. This valuable insight provides the ability to predict the timing of the next defrost operation, unless any external events disrupt this recurring pattern.



Similarly, in the figure below, when considering sensors in cold room freezers, consistent motifs were observed, reflecting the effects of defrost operations on temperature. These motifs occurred approximately every 4 hours.



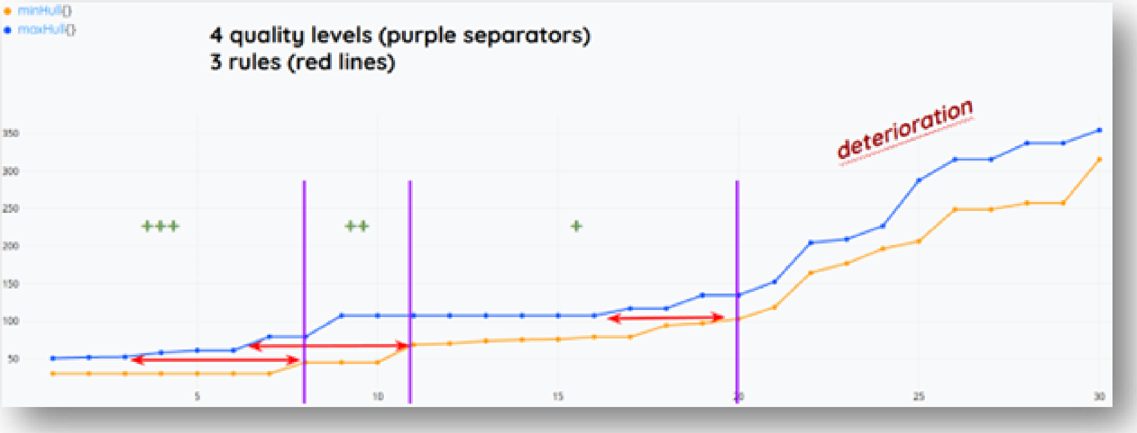
Note that atypical discordant patterns, such as the one observed at the beginning, may occur in the data. These patterns are likely a result of significant movements within the cold room or instances where the door remains open for extended periods.

4. Classify chicken samples using portable Raman spectroscopy

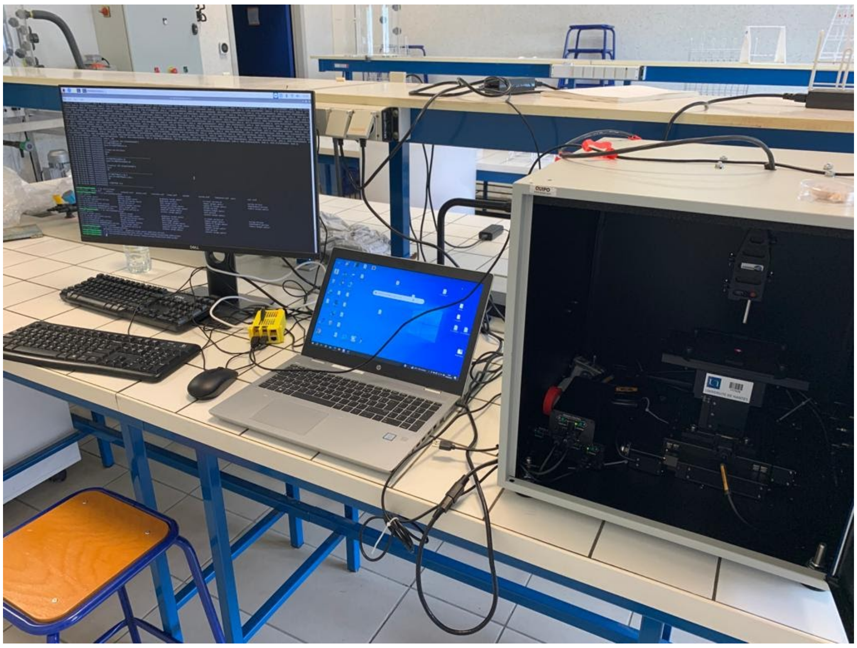
The goal of this study was to utilize portable Raman spectroscopy for the classification of chicken samples. To achieve this, a series of steps were undertaken. Analytics code was done in WarpScript and data was formatted as Time Series in Warp 10 format.

First, a custom module was developed to parse and convert files into a suitable time series data model. The preprocessing techniques were thoroughly refined and cross-validated in collaboration with Reamit partners. As a result, a standard preprocessing step was established for operational analysis, which involved using polynomial regression of degree five for baseline correction.

Next, an AI expert method was devised for discriminating food samples. This method relied on computing Euclidean distances between each sample and the mean spectra of the first day. Based on these distances, the samples were classified into four groups: "extra fresh," "fresh," "okay," and "expired." To define the group boundaries, expert rules were established using a threshold technique, with a focus on specific peaks to further refine these rules. This model is close to the depiction in the figure below.



The model's inference code was written in WarpScript and was efficiently implemented on the edge, directly connected to a prototype of a portable Raman spectrometer. Specifically, the Raspberry Pi was chosen as the platform for inference. It was equipped with a fully functional Warp10 platform, enabling the seamless transmission and local storage of the model's inference results. The local storage capability also facilitated debugging, improvements, and the creation of graphical representations as needed. The picture below shows the development of the prototype.



In conclusion, this study successfully developed and deployed a portable Raman spectroscopy-based classification model for chicken samples. The methodology ensured accurate and efficient classification, making it a promising tool for food quality assessment and monitoring.

contact@senx.io