crosser

Solution Overview Industrial IoT

Crosser ver. 2.2

Crosser Technologies AB

www.crosser.io

crosser

Table of Contents

crosser

Solution Overview Industrial IoT

Both factory owners and machine builders realize that there is a lot of value in the data generated by machines. Collecting and analyzing this data could help optimizing the uptime of the machines, optimizing the processes lines and optimizing administrative/business processes by integrating OT data with IT processes.

Even though the values are clear there are also several challenges:

- Technical challenges, like legacy and fragmented OT systems, OT/IT integration
- Business challenges, like IT & OT skill sets and system access, life cycle costs

There are also hidden challenges, such as access to skilled staff. Software developers are a scarce resource and availability of data scientists is even worse.

The values available by analyzing machine data is not only relevant to the automation engineers. Data scientists, business owners, and IT staff may need to be involved in the IoT projects to make them successful. Having a solution that allows all interested parties to understand what is happening to data and participate in the design choices significantly increases the chances of success.

Crosser has set out to provide a solution for edge streaming analytics that lowers the barriers of entry as well as the life cycle costs.

Why Edge?

Collecting all data generated from machines and store it centrally for further analysis may seem an obvious approach. In many cases, this is neither feasible nor optimal. This is where edge computing comes into play. In industrial IoT use cases, “edge” is the same as “close to the source”, whether that is on the machines themselves, on the factory floor or in an on-premise data center.

The main reasons for analyzing the data already at the edge are:

- Cost - Even though a lot of data may be available from the machines not all data may be relevant or be provided with unnecessary granularity. Selecting and aggregating data close to the source will save both bandwidth costs and costs for cloud services, which are typically charged based on data volume

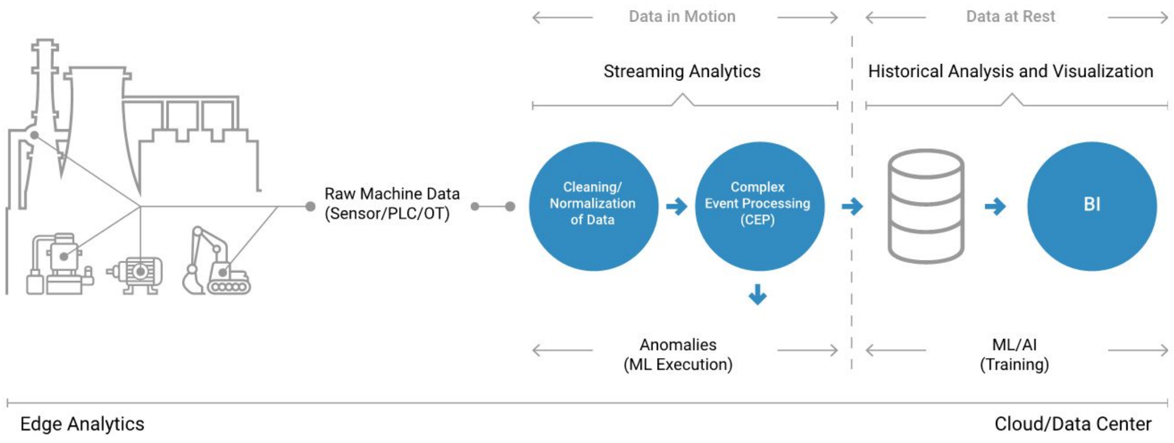
- Security - The machine data may contain sensitive information. By just sending the relevant data and maybe anonymizing it at the edge the security exposure is minimized

- On-premise integration - Not all use cases require data to be sent externally to a cloud service. It may be that the analysis of data from one machine can be used to optimise some settings in another machine. Or that the analysis results are needed in an on-premise ERP system. Sending data to the cloud for analysis is then adding no value, just increased security risks and higher costs.

- Reliability - In some cases, like mobile assets, network connectivity may not always be available or be unreliable. Buffering data locally until connectivity is available will overcome connectivity issues without data loss.

- Latency - For machine to machine communication, such as yield or process optimisation based on data analysis, the latency may be critical. Then again edge computing is the optimal choice, by avoiding the latency introduced by round-trips to the cloud.

In most scenarios edge analytics scenarios edge computing is combined with storage and long-term analysis in the cloud. A typical industrial IoT data processing pipeline is shown in the figure below.


Data in Motion
Data at Rest
Streaming Analytics
Historical Analysis and Visualization
Raw Machine Data (Sensor/PLC/OT)
Cleaning/ Normalization of Data
Complex Event Processing (CEP)
BI
Anomalies (ML Execution)
ML/AI (Training)
Edge Analytics
Cloud/Data Center

No data - No party!

Even though the ultimate goal may be to run advanced machine learning algorithms that detect anomalies or extract other high-value results from the data, the reality is that in most industrial IoT projects the main issue is to get hold of the relevant data in a format that can be used for analysis. This initial step is crucial to be successful with any type of more advanced processing. It is in many cases also a good starting point that allows data to be collected and analyzed offline to get an understanding of the potential features that can be extracted. Once a better understanding of the potential is available more advanced algorithms can be implemented to optimize the processing.

The goal with the Crosser Edge streaming analytics solution is to provide a framework that supports our customers through the journey from the initial,

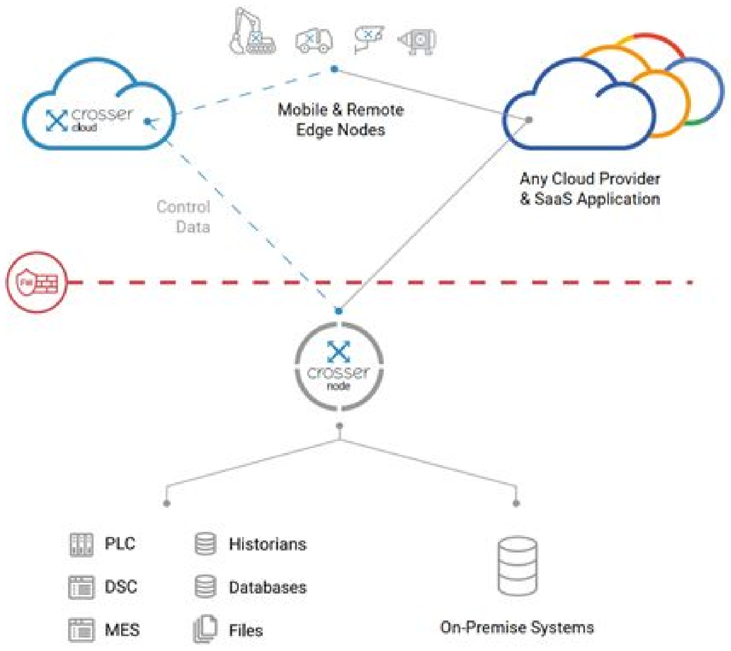
“Collect → Package → Ship” use case all the way to “Collect → Analyze → Act”,

where advanced algorithms and machine learning is applied in the edge. Independent where on the journey you are the same simple tools and ease of management are available to help with the deployment of all your edge streaming analytics needs.

The Crosser Edge Streaming Analytics Solution

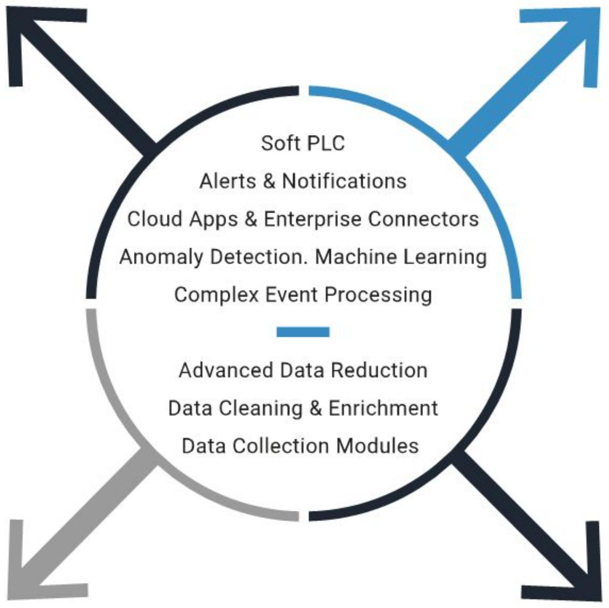
The Crosser solution consists of two main parts, the Edge Node and the Cloud Service. The Edge node is a generic software platform where the actual processing of sensor data is executed by configuring it with different process flows. The Cloud service is used to build processing flows using a graphical editor and then deploy these flows onto any number of edge nodes.

The Cloud service also monitors the edge nodes and provides a dashboard with status information. Sensor data is never sent to the Cloud service, it's fully controlled by the flows running on the edge nodes and will only be sent to destinations configured in the flows.



The Edge node uses a flow-based programming paradigm where common functionality is abstracted into modules, which are then connected into flows to specify the sequence of operations to apply to the data. Collector modules get data from external sources, such as a machine. Compute modules perform some operation on the data, such as reformatting, aggregation, conditions and so forth. Action modules deliver the results to external systems, such as cloud services, another machine or on-premise system.

The Crosser solution comes with an extensive library of modules with the intention to cover common use cases. It is also possible to run custom C# or Python code in code modules or build fully customized nodes using the SDK. The combination of pre-built modules that implements complex operations and a visual designer makes it easy to implement different types of edge streaming analytics, starting from “collect, package and ship” use cases all the way to advanced use cases where machine learning models are used to analyze the data already at the edge.



Stream-based processing

In a stream-based processing engine, like the Crosser Edge node, data is processed as soon as it arrives and the result of one processing step is immediately delivered to the next step until you reach the end of the flow and data leaves the system by being sent to some destination. This is very much like an assembly line where items pass a number of stations. In each station, you perform the same operation on each item that passes by. In the case of streaming analytics “stations” equals modules and “items” are the data messages.

Since data is processed as soon as it arrives there is typically no long-term storage being involved. Long-term storage like databases can still be used as a source or destination of data being processed. Short-term storage can be used to be able to perform calculations over sequences of messages. To build clean and efficient processing flows a single processing step (module) should preferably perform a single operation on a single data item. Therefore you may need to break up more complex data structures, like arrays, into a sequence of messages that can be processed as a stream of data.

Sometimes multiple streams of data need to be combined, for example, to provide a multi-sensor input array to a machine learning model. The data can arrive over the same input interface (module), or arrive through different interfaces. In either case, data will arrive at different points in time, even though the message frequency may be the same. It can also be that some sensors send data at a higher frequency than others. In cases like this, it may be necessary to synchronize data from multiple sources on time, e.g. combine messages received during a one-second window into a single message that can be used as input by a processing module.

The module library has full support for going back and forth between complex and simple data structures and different message formats (see below).

Message formats

The data to be processed by an edge node can come in many shapes and forms, depending on the sources generating the data. In many cases, these are legacy systems without any type of metadata describing the data. The Crosser streaming analytics engine has been designed to support a very flexible data format where no assumptions are made about specific formatting conventions.

Internally, a dynamic message structure is passed between modules and the structure of these messages can be modified by using dedicated modules in the flows. It is often advantageous to have a common format internally to be able to apply the same processing flow on data independent on the sources of the data. When delivering results the data needs to be formatted to match the expectations of the receiving systems. Having a system that supports flexible data models allows data from any system to be delivered to any other system, optionally processed or conditioned on the way.

Edge Node

At the core of the Edge node is Crosser's in-house developed real-time streaming analytics engine, which runs in .NET core. This is a high-performance, low footprint engine that manages the data flowing in and out of the edge node. It comes with an integrated MQTT broker and HTTP server for easy integration with common data sources. The flow engine manages the flows configured on the edge node by downloading the modules used and routing data between modules.

Flows are just configurations, the actual edge software is not changed when deploying new flows. This greatly simplifies the life cycle management since no software needs to be updated and new flows can be added without interruption of any existing flows. The Edge node software is deployed as a Docker container and runs on any host that supports Docker, including most Linux, Windows and MacOS systems. The base image size is ~100MB and you need ~100MB of free RAM to run the edge node. Even on the most lightweight hardware platforms you can still process thousands of messages per second. To support a fully standardized python environment, which is not possible within the .NET core framework, we offer the option to run an external python environment. This allows running any Python code, including machine learning frameworks, while still accessing the functionality using a module in a flow. With this extension the container image will be larger, depending on the libraries required.

The Crosser Edge node can be installed on any host supporting Docker by downloading the image from Crosser's Docker registry.

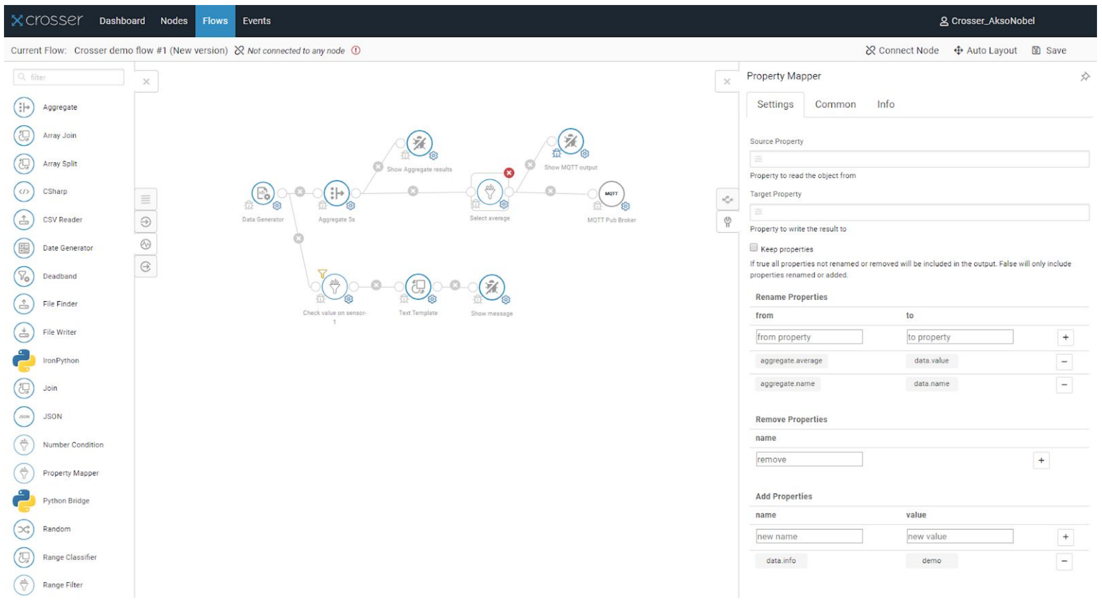
Crosser Edge on Azure IoT Edge

Since the Edge node is just a single Docker container it can also be hosted on an Azure IoT Edge device, where the whole Edge node is seen as a module on the Azure IoT Edge. The Crosser module library has connectors for getting data from and sending data to the edge hub. The Crosser Edge node can then be used as a streaming analytics engine on the IoT Edge where the flows can easily be implemented to process data from the edge hub.
Flows are still built and deployed using the Crosser Cloud service, without changing the Docker container running on the IoT Edge. In addition to process data from the edge hub, the Crosser collector and action modules can be used to get data from other external sources and results can be sent to destinations other than the IoT hub in the Azure cloud.

Crosser Cloud

With the Cloud service you can do the following:

- Build flows with the Flow Studio visual designer
- Manage flow versions
- Deploy new or changed flows onto any group of nodes
- Use labels to create flexible groups of nodes
- Manage resources, such as C#/Python code, tag lists and ML models
- Manage credentials, such as access information and certificates for external services
- Register new nodes
- Monitor the status of your nodes



Current Flow: Crosser demo flow #1 (New version) Not connected to any node

Property Mapper

Source Property

Property to read the object from

Target Property

Property to write the result to

Keep properties

If true all properties not renamed or removed will be included in the output. False will only include properties renamed or added.

Rename Properties

from	to
from property	to property
aggregate.average	data.value
aggregate.name	data.name

Remove Properties

name
remove

Add Properties

name	value
new name	new value
data.info	demo

Flow Studio

The Flow Studio is a visual design tool with a drag'n drop interface to build flows by connecting modules from the library. Each module typically has some settings that must be configured, such as connectivity information and which data that should be processed. Data streams can be combined at any point in a flow by just connecting multiple streams to the same input. In the same way, multiple paths can be created by connecting an output to the inputs of multiple modules. More advances split/join operations can be achieved by using dedicated modules that applies some condition on the data.

In many cases, you don't know the exact format of the data that should be processed, or what the output of a module looks like. It is, therefore, crucial to be able to see what is happening at each stage. The Flow Studio has a LiveView where you can test your flow by running it interactively on a node and check the result of each stage in the flow. You can either run your flow in a safe sandbox environment, which is an Edge node deployed as part of the Cloud service or by connecting the Flow Studio to any of the nodes you have deployed. The latter mode means that you can test your flow with real data accessible by that node. When in LiveView you can toggle the display of data in and out of each module.

Edge Director

Once you have finished building or updating your flow it's time to leave the Flow Studio and enter the second part of Crosser Cloud, the Edge Director. Here you can deploy your new flow onto any number of nodes, by operating on a group of nodes. Groups of nodes are created by adding labels to the nodes. A label is a text string and each node can have any number of labels, to support different types of grouping including overlapping and hierarchical groups.

Flows have versions and as soon as a flow is deployed on a node the current version is locked. To make changes a new version must be created. In this way, you never end up having a flow with the same name running in multiple places and doing different things.

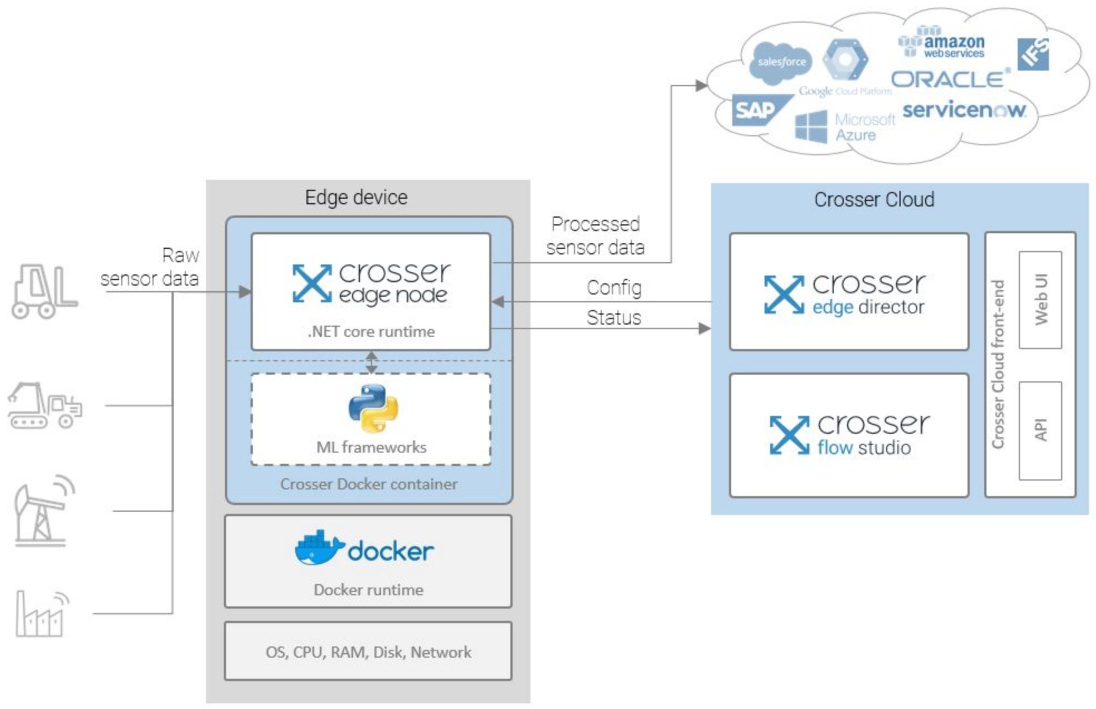
The Edge Director is also the central place for managing resources used by flows. For example, if you have built a tag list for your OPC UA server you can upload that file and store it centrally. When building a flow containing an OPC UA module you then just refer to the central resource. Same thing with credentials, if you have an account with an SMS service you store the credentials for that service in the Edge Director. When you want to send an SMS from a flow you just refer to those credentials from the module in the flow, without having to expose the credentials to everyone in your organization.

Credentials are stored securely and the actual content is never seen except when first entering the information. A final example is a trained machine learning model that you want to use in the Python module and load it into the TensorFlow runtime. By uploading this model file to the Edge Director it will automatically be downloaded by all Edge nodes running the flow where it is used and you don't need to worry about distributing this file.

All Edge nodes report their status to the Cloud service on a regular basis and this information is presented on a dashboard so you can ensure that all nodes are operational and also that your latest flow is not overloading the small gateway you are trying to run it on. Since the actual data being processed in the Edge nodes is not sent to the Cloud service the dashboard only presents summary statistics, like the number of messages and data volumes being processed by each Edge node.

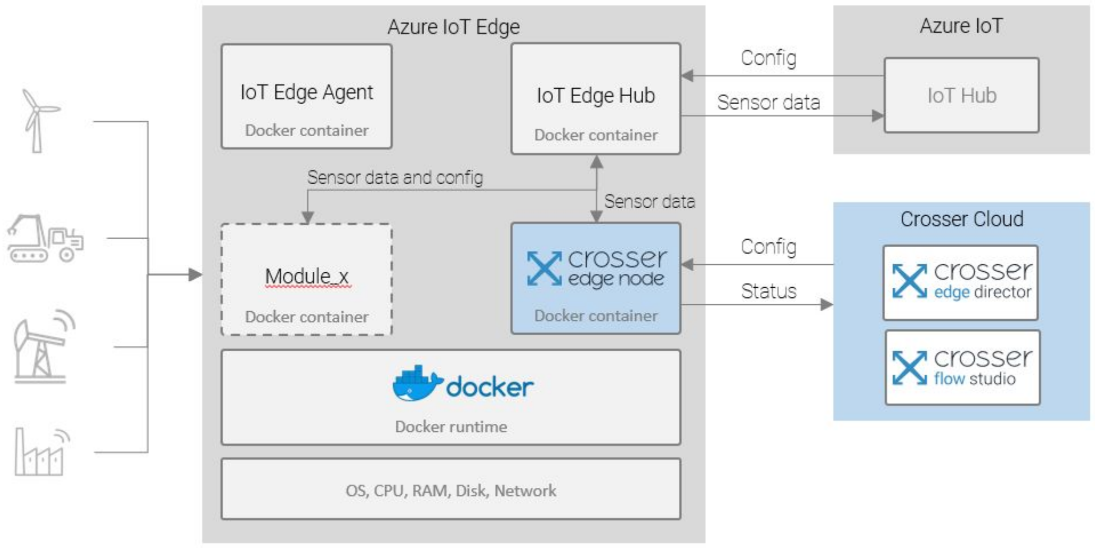
Deployment options

The Crosser streaming analytics solution can be deployed in two ways, stand-alone and on Azure IoT Edge. When deployed in stand-alone mode the Edge node is installed on any Docker compatible host using the standard Docker tools to download container image.



Stand-alone deployment

The second deployment option is to run the Crosser Edge node on the Azure IoT Edge platform. It is the same Docker container image but here it is hosted on the IoT Edge and is deployed using the Azure tools. The Crosser Edge node is available on the Azure IoT Edge marketplace.



Deployment on Azure IoT Edge

In both deployment scenarios, the Crosser Cloud service is used to build and deploy flows on the Edge nodes and Edge nodes running on the IoT Edge are monitored in the same way as stand-alone nodes. Stand-alone and IoT Edge hosted nodes can be managed from the same Cloud account and the same flows can run in both environments, as long as they don't use edge hub data.

Security

Security is an important aspect of any industrial IoT project and the Crosser edge streaming analytics solution has been designed with security as a top priority. Security comes into play in different places, who has access to what data, who has access to what systems?

The following security principles have been implemented in the Crosser solution:

- All communication between Edge nodes and the Cloud service is initiated at the Edge nodes. This is to avoid having to open ports in firewalls for inbound internet traffic.
- Edge nodes must register with the cloud service using a unique ID and access key before any management operations can be performed.
- All communication between Edge nodes and the Cloud service is encrypted using HTTPS on standard ports.
- Data outputs from flows running on Edge nodes implement the security mechanisms supported by the receiving end.
- Endpoints for receiving data into an Edge node, such as MQTT and HTTP can be configured to require authentication and encryption.
- Sensitive information, such as credentials, is stored securely in the Cloud service and the information is never exposed to end users, except when first entered into the system.

Module library

The Crosser module library consists of three broad module categories:

- Collector modules - Gets data from an external data source, such as OPC UA servers, Modbus PLCs, MQTT devices and HTTP APIs.
- Compute modules - Processes message data in a flow. Examples of compute modules:
    - Data formatting: Change message structure and property names, split/join arrays
    - Processing: Aggregate data over tumbling windows, calculate statistics over sliding windows, smooth values...
    - Conditions and filters: Range filter, deadband...
    - Code modules: Run your own C# or Python code to process message data
- Action modules - Send the results to external systems, such as Azure FileStorage, AWS S3, Slack, Nexmo SMS, HTTP REST APIs.

All modules also have advanced filtering capabilities to specify which messages that should be processed by the module, by applying conditions on any parts of the message.

It is also possible to build custom modules using the Crosser SDK and C#. These modules will then be available in the Flow Studio for any users belonging to your organisation, or if you allow, any user of the Crosser Cloud service.

crosser

Why Crosser

1. ONE PLATFORM FOR BOTH IT, DATA SCIENTIST & OT TEAMS
   Simplifies decentralization and lets teams cooperate easily without dependencies on individuals.

2. SHORT LEARNING CURVE
   Empowers non-developers to quickly and easily build advanced Industrial IoT projects.

3. MORE & FASTER INNOVATION
   Visual development with pre-built modules encourage more innovation.

4. FREES UP DEVELOPER TIME
   Allows developers to focus on more value-added tasks in core business.

5. FASTER AND LOW RISK
   Enables faster POCs and Time to Market.

6. LOWER COST
   Dramatically reduces Total Cost of Ownership.