Zenoh
A Crash Course in Building Cloud-to-Microcontroller Applications

Zenoh Team
SMART FACTORY

An interesting use case...
What is a smart factory?

An interconnected network of machines, communication mechanisms, and computing power.

https://www.sap.com/insights/what-is-a-smart-factory.html
Smart factories are built around data for **realtime operations, monitoring, data analytics**, etc.

Data in **motion** and data at **rest** are both crucial.
A variegated world

Different devices and systems live together

Sensors, robots, control systems, cloud, etc. need to cooperate and interact
Microcontrollers are as important as data centres

Data are generated and consumed by sensors, actuators, and apps in data centres
Speaking many languages

Multiple **communication protocols** (e.g. DDS, OPC-UA, MQTT, PROFINET, REST, etc.)

Multiple **network technologies** (e.g. WiFi, 5G, Serial, Bluetooth, etc.)
Communication in smart factories is a mix of

**Peer-to-peer** (e.g. between robots for coordination)

**Infrastructured** (e.g. with the control room)
Let’s build a virtual fence for our smart factory!
Virtual fence and robots

Sensors and actuators are ubiquitous: from the building to the robots.

Distributed control system using a virtual fence to stop robots.
What we are going to build

Virtual fence
Remote robot controller
Store robots telemetry
Mobile robots
Remote robot controller
Dashboard
Virtual fence: let’s sense

ESP32 with ultrasonic sensor

ESP32 with LED

Pub distance
factory1/room42/distance

Sub distance

WiFi

Zenoh

LED turns red if too close

Pub alarm light
factory1/room42/led/green
factory1/room42/led/red
Zenoh runs everywhere

Native libraries and API bindings for many programming languages

Over various network technologies: from transport layer to data link

On embedded and constrained devices
Extending beyond a LAN
Zenoh supports many topologies

**Peer-to-peer**
Clique and mesh topologies

**Brokered**
Clients communicate through a router or a peer

**Routed**
Routers forward data to and from peers and clients
Storing data

- Pub distance
- Store distance
- Sub distance
- Pub alarm light
- Pub luminosity
- Factory1/*/distance
- Factory2/*/distance
- Store luminosity factory/*/luminosity
- Sub distance
- Sub luminosity
Retrieving data

Pub distance

Sub distance
Pub alarm light
Pub luminosity

Store distance

factory1/*/*/distance

Store luminosity
factory*//*/luminosity

factory2/*/*/distance

Get
factory*/**
Zenoh is extensible

Enhance zenoh with plugins
Interact with other technologies

“Coming together is a beginning; keeping together is progress; working together is success.” - Edward Everett Hale
Robots are coming
Extend ROS2

zenoh-bridge-dds --scope /bot1
zenoh/TCP

demo.zenoh.io

REST plugin

HTTP

Zenoh ROS2 teleop
Zenoh REST URL:
http://demo.zenoh.io:8000

Config

Drive

STOP

Acceleration

Remote controller

Lidar data

Zenoh plugin for DDS

ROS2

DDS

Zenoh plugin for DDS
Getting a twin

Store motion commands
bot*/rt/cmd_vel

Pub motion commands
bot1/rt/cmd_vel
bot2/rt/cmd_vel

Get motion commands
bot1/rt/cmd_vel
Pub motion commands
bot2/rt/cmd_vel

Sub motion commands
bot1/rt/cmd_vel
Sub motion commands
bot2/rt/cmd_vel
Record and replay ROS2

Zenoh plugin for DDS

zenoh-bridge-dds --scope /bot1

get(/bot1/...)

put(/bot2/...)

ros2-replay -i /bot1 -o /bot2


Zenoh plugin for DDS

demo.zenoh.io

Store motion commands

InfluxDB backend plugin

Zenoh plugin for DDS

Zenoh plugin for DDS

Zenoh plugin for DDS
Key takeaways
Zenoh and the smart factory

Seamless mixing of **real time** and **stored** data

Different **devices** and **networks**: from **micro-controllers** to the **cloud**

**Integration** with third-party **technologies**
Zenoh and robotics

Out-of-the-box integration with ROS2 robots

Intra-robot and inter-robot communication

High throughput, low latency, low wire overhead, and simple API
Zenoh

Unifies data in motion, data in-use, data at rest and computations from embedded microcontrollers up to powerful data centres.

Provides a location-transparent API for high performance pub/sub and distributed queries across heterogeneous systems.

Facilitates data representation transcoding, geo-distributed storage and distributed computed values in a plug-and-play fashion.

“Some people want it to happen, some wish it would happen, others make it happen.” – Michael Jordan
Zenoh-Flow: A data flow programming framework for the Cloud-to-Thing
Motivation

Zenoh holds a promise: becoming the backbone of many (new) distributed applications.

Zenoh-Flow builds on that promise: providing a Cloud-to-Thing framework that eases the “making” of complex applications.
Ambition

Simpler & Resilient foundations

Declarative

Automatic deployment

Unified abstraction

Rust-based
What it looks like

Data flow description (YAML)

ZFCTL

Zenoh-Flow daemon

Zenoh-Flow daemon
"Pylot is an autonomous vehicle platform for developing and testing autonomous vehicle components (e.g., perception, prediction, planning) on the CARLA simulator and real-world cars."

https://github.com/erdos-project/pylot
Zenoh

Zero Overhead Pub/sub, Store/Query and Compute.

zenoh /zeno/ unifies data in motion, data at rest and computations. It elegantly blends traditional pub/sub with geo distributed storage, queries and computations, while retaining a level of time and space efficiency that is well beyond any of the mainstream stacks.

Get started

Don’t forget to visit Zenoh’s website...

https://zenoh.io/

...and the blog

Zenoh-Pico: Above and Beyond

09 June 2022 -- Paris.

In a previous blog post, we introduced Zenoh-Pico, an implementation of Zenoh for microcontrollers and embedded devices, along with a preliminary performance results and its integration on off-the-shelf robots (by bridging both legacy ROS2+DDS and Zenoh systems or by making it a full-fledged Zenoh system).

In this post, we will dive deeper on Zenoh-Pico, show how Zenoh-Pico is capable of:

- exchanging close to 2.5M msg/s for small payloads, and over 25 Gbps for larger messages,
- achieving end-to-end latency (i.e., one way delay) as small as 45 μsec and 15 μsec for unicast and multicast transports, respectively,
- minimizing the overhead in the wire down to 5 bytes per data transmission,
- fitting all its capabilities in less than 50KB footprint, which can be quickly reduced to ~15KB in tailored compilation setups, and
- provides simple to use and yet powerful APIs.