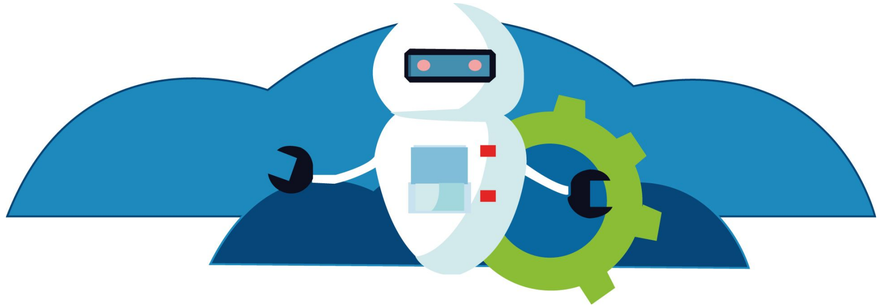
micro-ROS: bringing ROS 2 to MCUs

Francesca Finocchiaro - eProsima

December 16th, 2020



Overview

Who are we?

OFERA

eProsima
The Middleware Experts

FIWARE FOUNDATION

BOSCH

PIAP

micro-ROS

***Open-source** project,*
now benefiting from a huge participation from a growing community!

https://micro-ros.github.io/
https://www.eprosima.com/
francescafinocchiaro@eprosima.com

funded by European Commission

Why micro-ROS

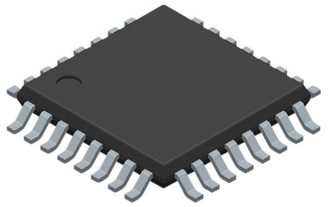
micro-ROS: puts ROS 2 onto microcontrollers!

A solution for creating ROS 2 nodes into embedded devices

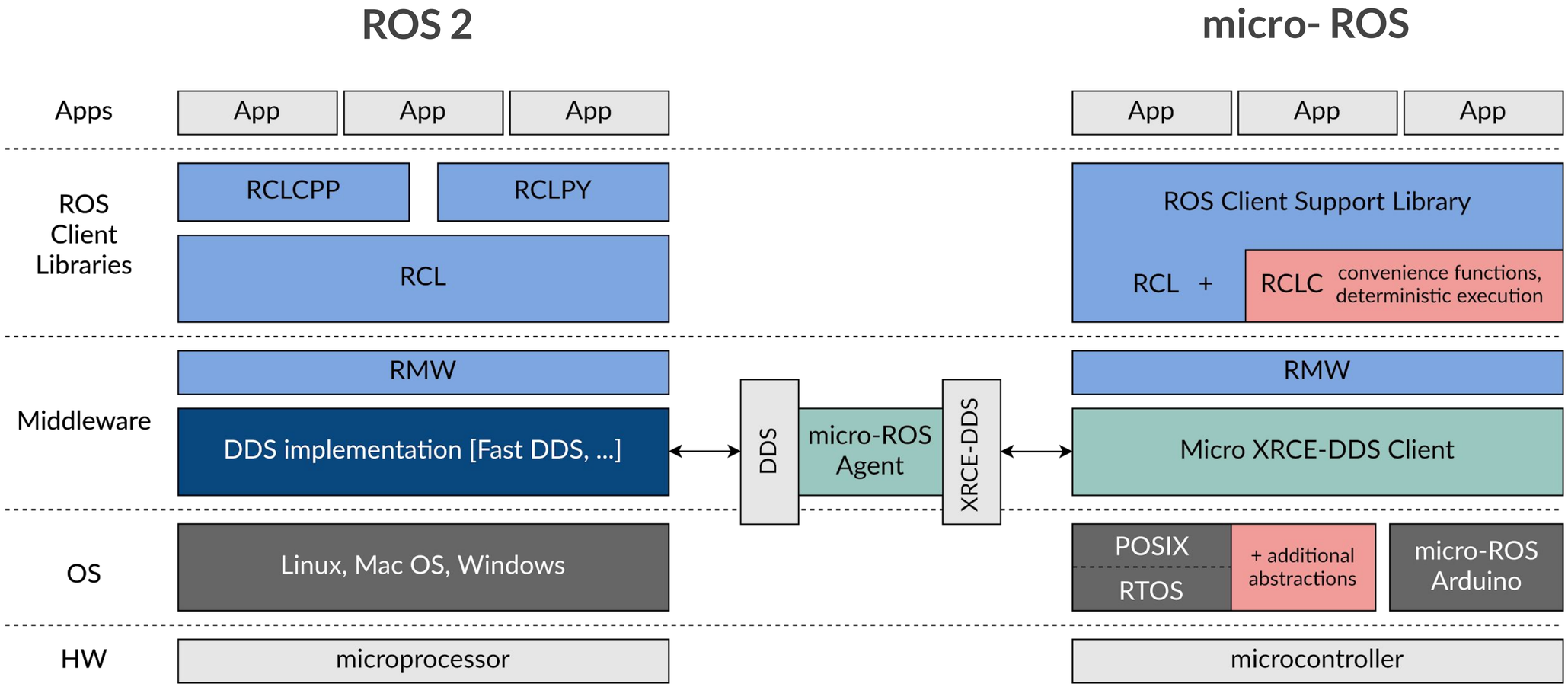
Why micro-ROS

Highlights

- Layer-compatible with ROS 2
- Integrated into ROS 2 ecosystem
- Allows to create a ROS 2 node with ~ all functionalities
- *Client/server* logics
- Middleware transports fully customizable
- Runs on different RTOSes and MCUs
- Platform-versatile cross-compilation tools
- Benefits of full QoS support
- Now supporting ***Foxy***
- A growing community!



micro-ROS architecture



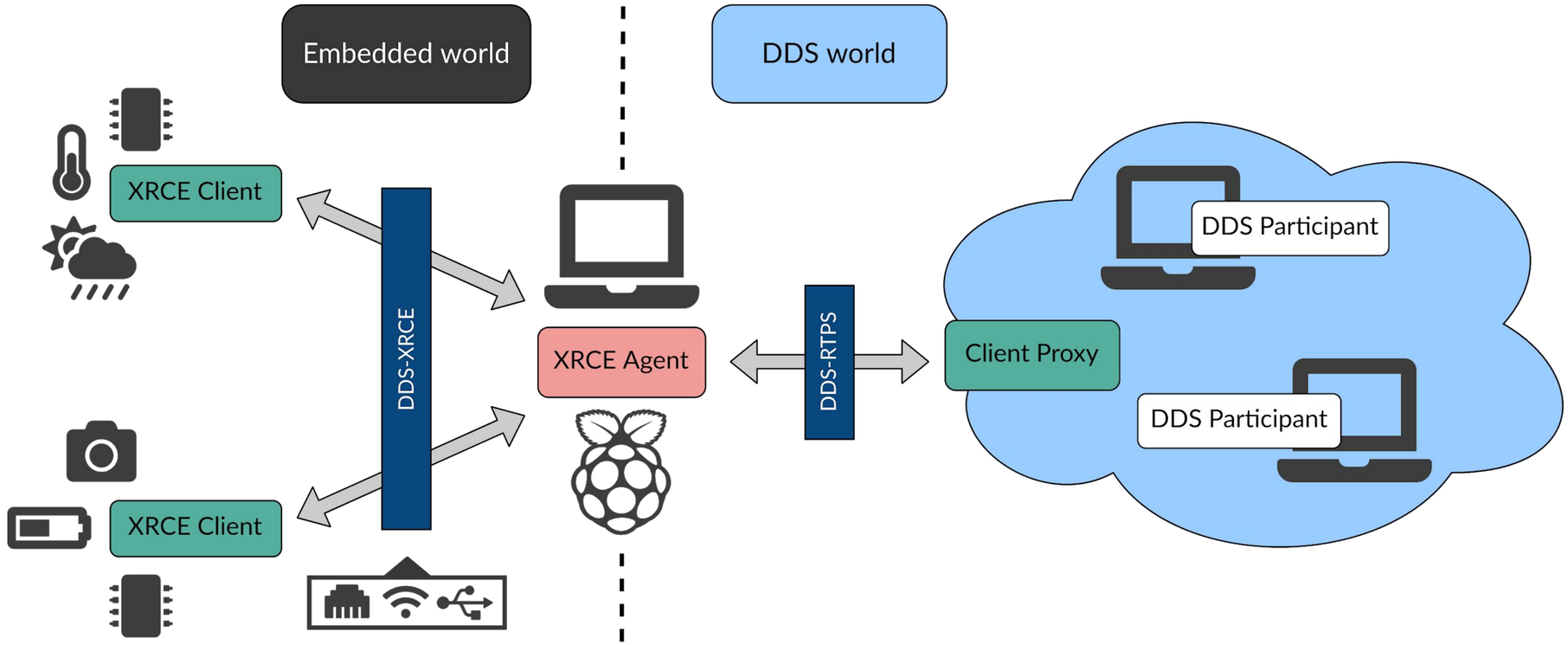
Middleware architecture

Micro XRCE-DDS: DDS for e**X**tremely **R**esource-**C**onstrained **E**nvironments.

Clients - XRCE entities on low-resource consumption devices.
Agent - XRCE entity connected with DDS global data space. Acts on behalf of Clients in the DDS world.

Main features:

- *Client-server architecture*
- *Request-response pattern*
- *Connection oriented*



eProsima
The Middleware Experts

RMW

- Implemented using Micro XRCE-DDS middleware in lower layers
  - Allows static configuration of memory resources

Micro XRCE-DDS configurable parameters

- Transport [UDP, serial, custom]
- Agent IP
- Agent Port
- Creation mode [XML, Ref]
- IP version [IPv4 - IPv6]

micro-ROS configurable parameters

- Max Publishers
- Max Subscriptions
- Max Clients
- Max Services
- Max Topics
- Max History
- Node name max length
- Type name max length
- Max Nodes
- Topic name max length

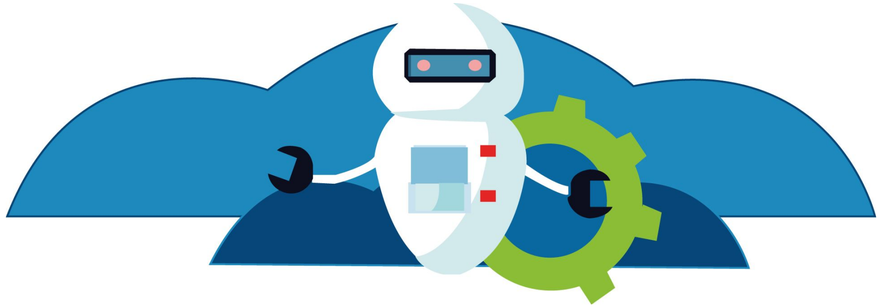
Configurability of these parameters allows preconfiguring the size of the library and tuning the size of the buffer to the memory needed

ROS Client Support Libraries

App	App
RCLCPP, RCLPY	RCLC
RCL, RCUtils, rosidl_typesupport	RCL, RCUtils, rosidl_typesupport

C99 library:
provides utility functions for creating ***nodes***, ***publishers***, ***subscribers*** & ***redesigned executor*** [deterministic and LET semantics, dynamic memory allocation only at startup, domain-specific scheduling]

Same as in ROS 2
(many functionalities not used)



Supported platforms

Supported RTOSes

NuttX

Zephyr

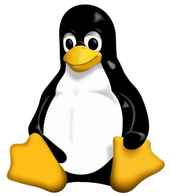
Zephyr™

FreeRTOS

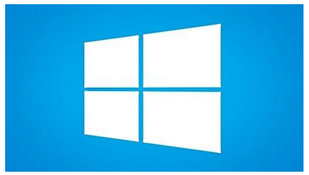
NEW!

micro-ROS Client

micro-ROS Agent



Linux

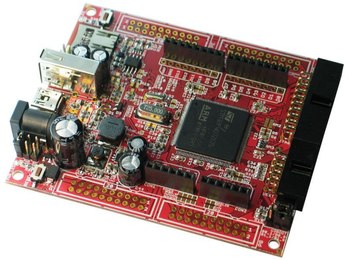


Windows

Supported HW

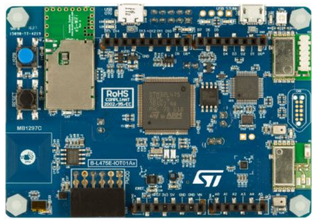
Officially supported HW...

Olimex LTD STM32-E407

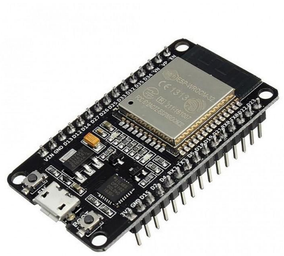


Crazyflie 2.1 drone

STM32L4 Discovery kit IoT



ESP32/ESP32-S2



Target: mid-range microcontrollers.

Currently supported:

- ARM-M4/M7 MCUs (STM32, i.MX RT ...)
- Xtensa MCUs (ESP32)

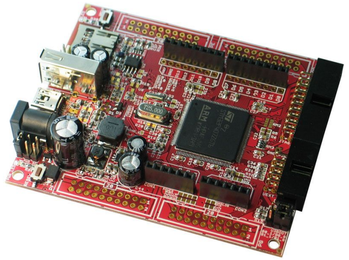
Typical features:

- ~ 1MB of flash memory
- ~ 200 KB of RAM memory
- < 500 mA consumption
- General purpose input/output pins (GPIO)
- Communication peripherals: USB, Ethernet, SPI, UART, I2C, CAN, etc

Supported HW

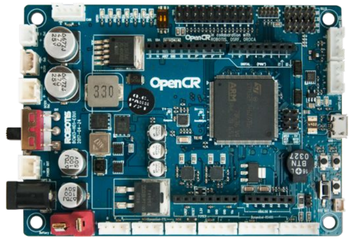
… + community-supported HW!

Olimex LTD
STM32-E407

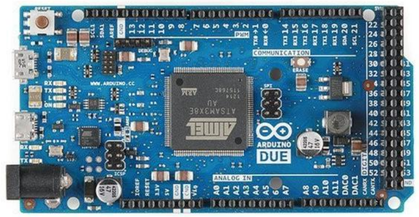


Crazyflie 2.1 drone

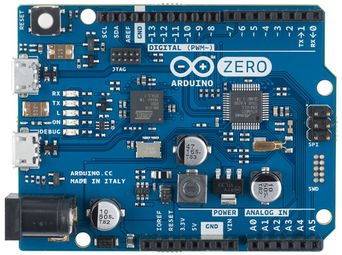
OpenCR 1.0



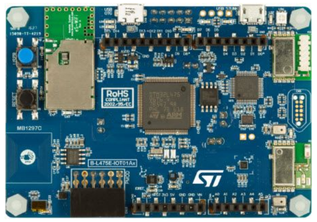
Arduino Due



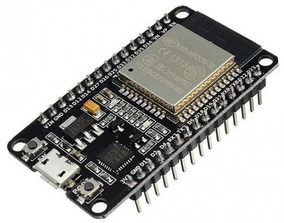
Arduino Zero



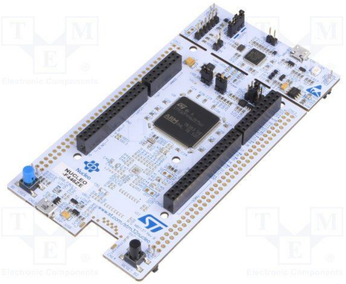
STM32L4
Discovery kit IoT



ESP32/ESP32-S2

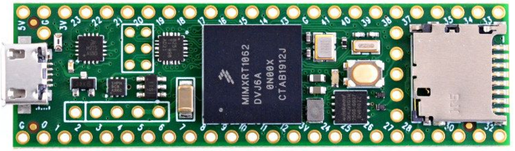


ST Nucleo
F446ZE/H743ZI/F746ZG



Teensy 3.2

Teensy 4.1



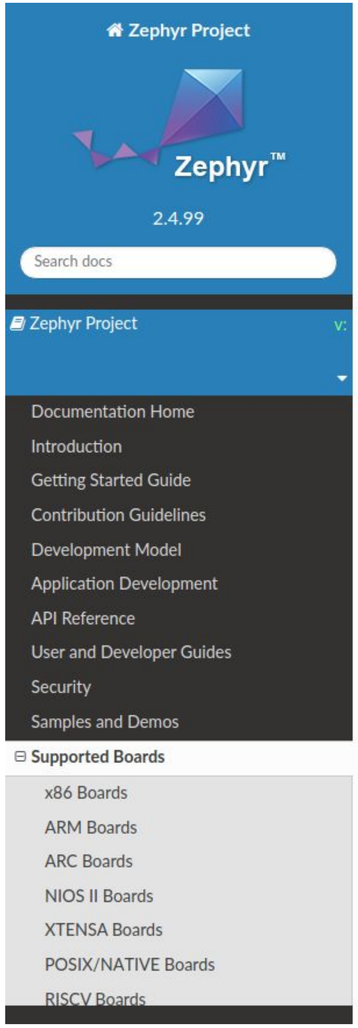
Porting new platforms

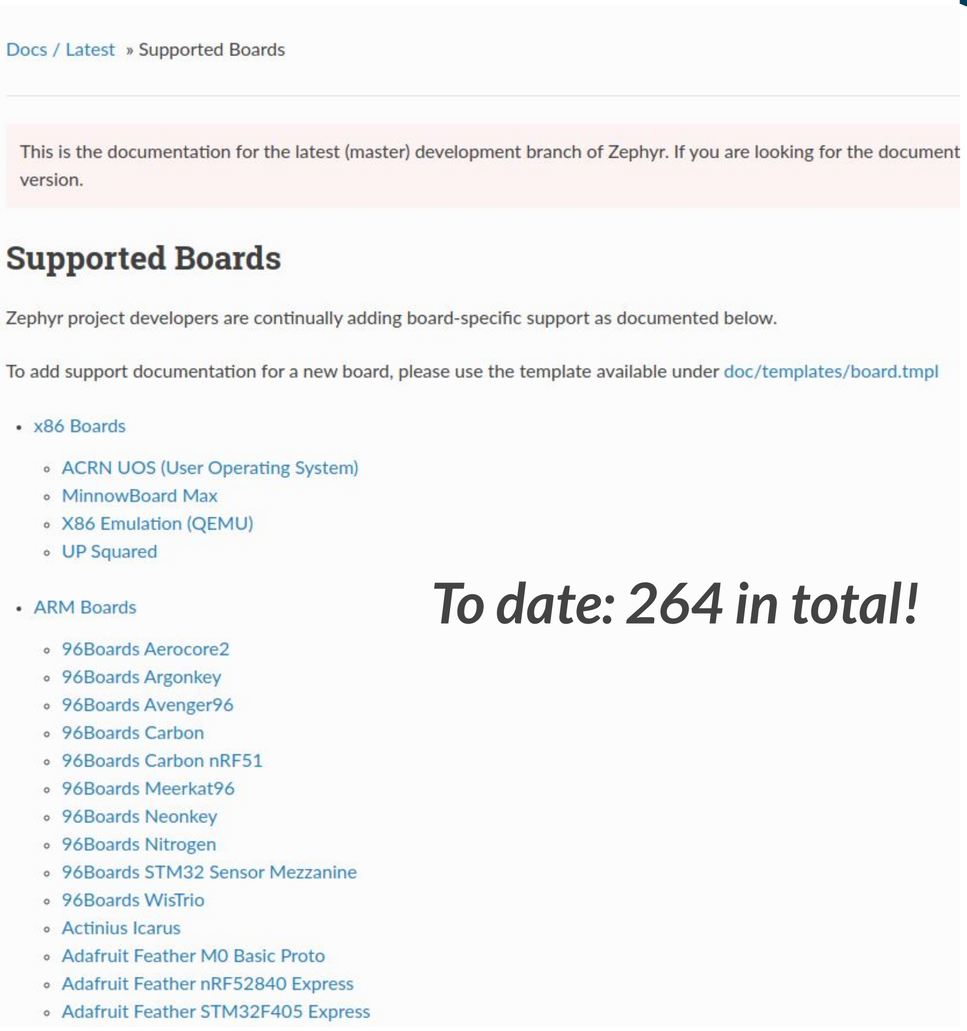
Porting new boards with Zephyr RTOS is super-easy thanks to the huge amount of boards already supported by The Zephyr Project!

Compatibilities to be aware of:

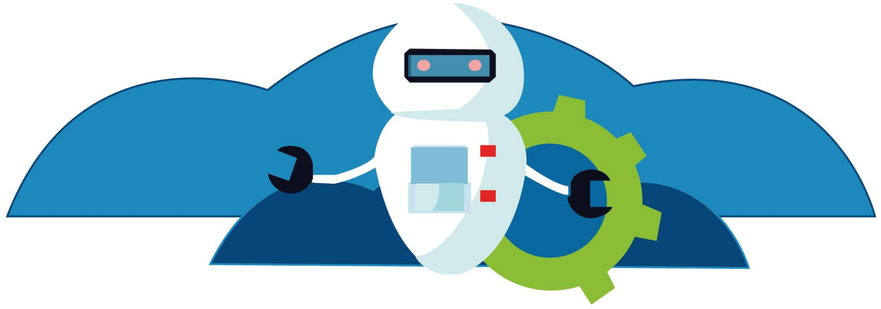
- *Memory resources*
- *Transports*

Zephyr™





To date: 264 in total!

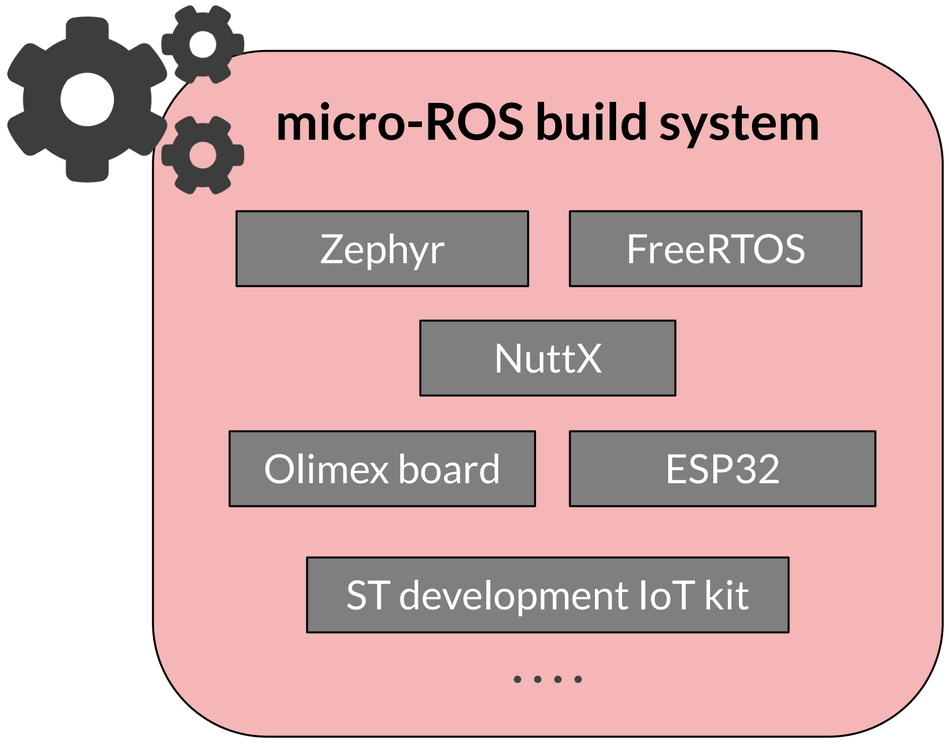


Recent developments and WIPs

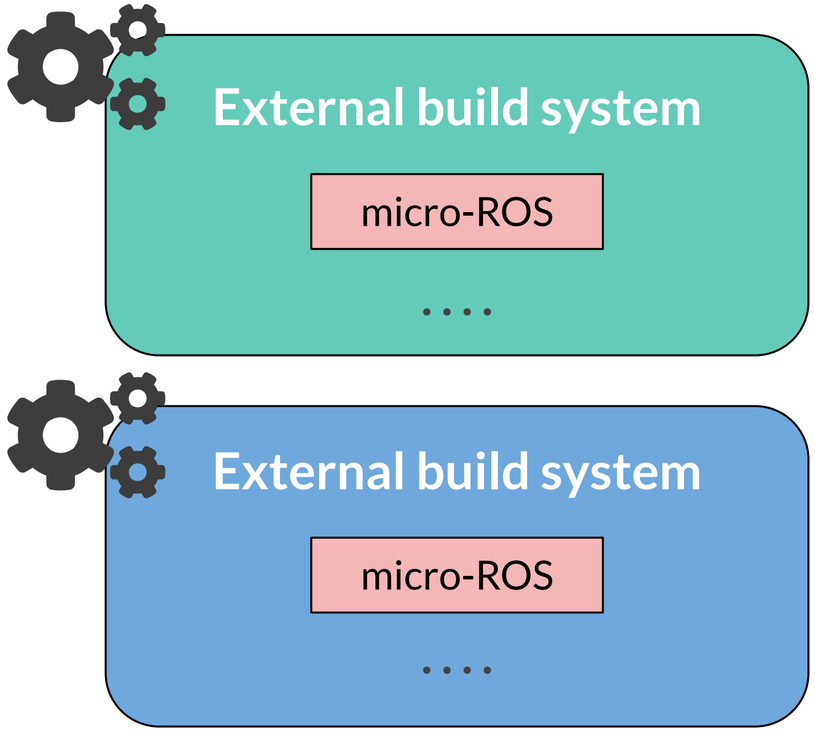
A twofold build system

The micro-ROS build system: now a two-tales story

Classic approach



New complementary approach



Achieved by generating standalone micro-ROS library & headers

A twofold build system

micro-ROS as an ESP-IDF component

ESPRESSIF

freeRTOS

micro-ROS as a Zephyr module

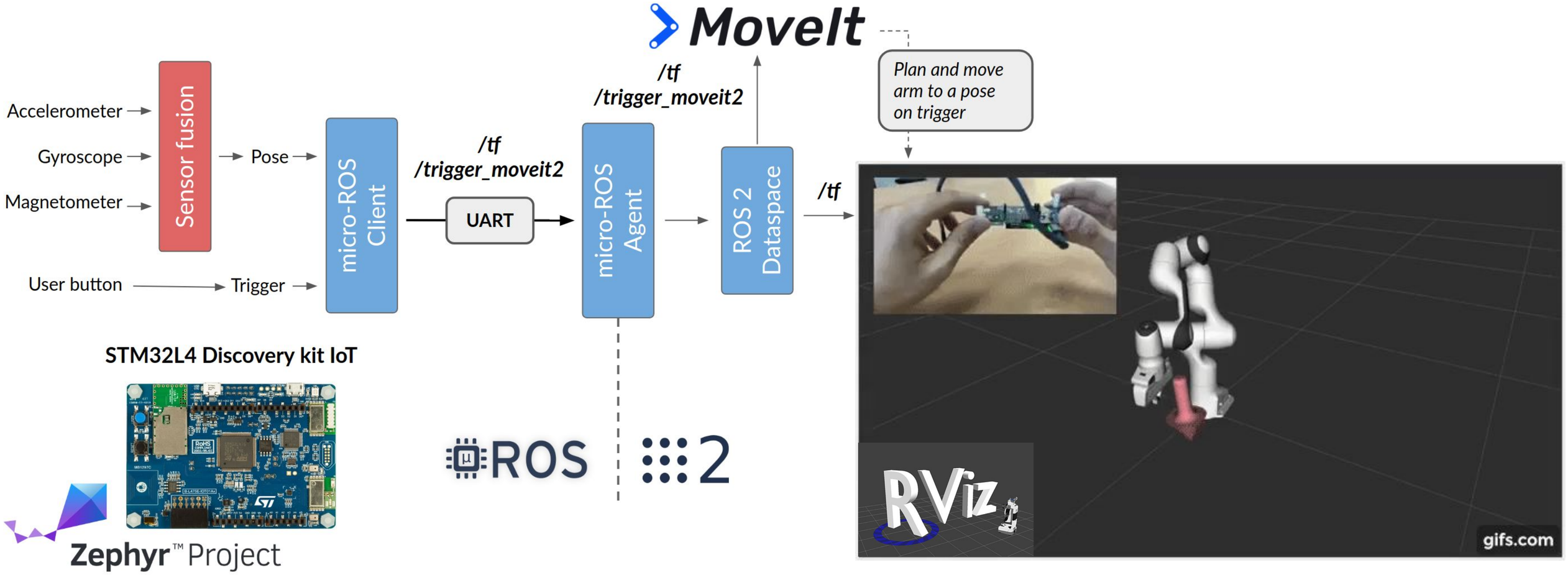
Zephyr™

micro-ROS into Arduino IDE

ARDUINO™

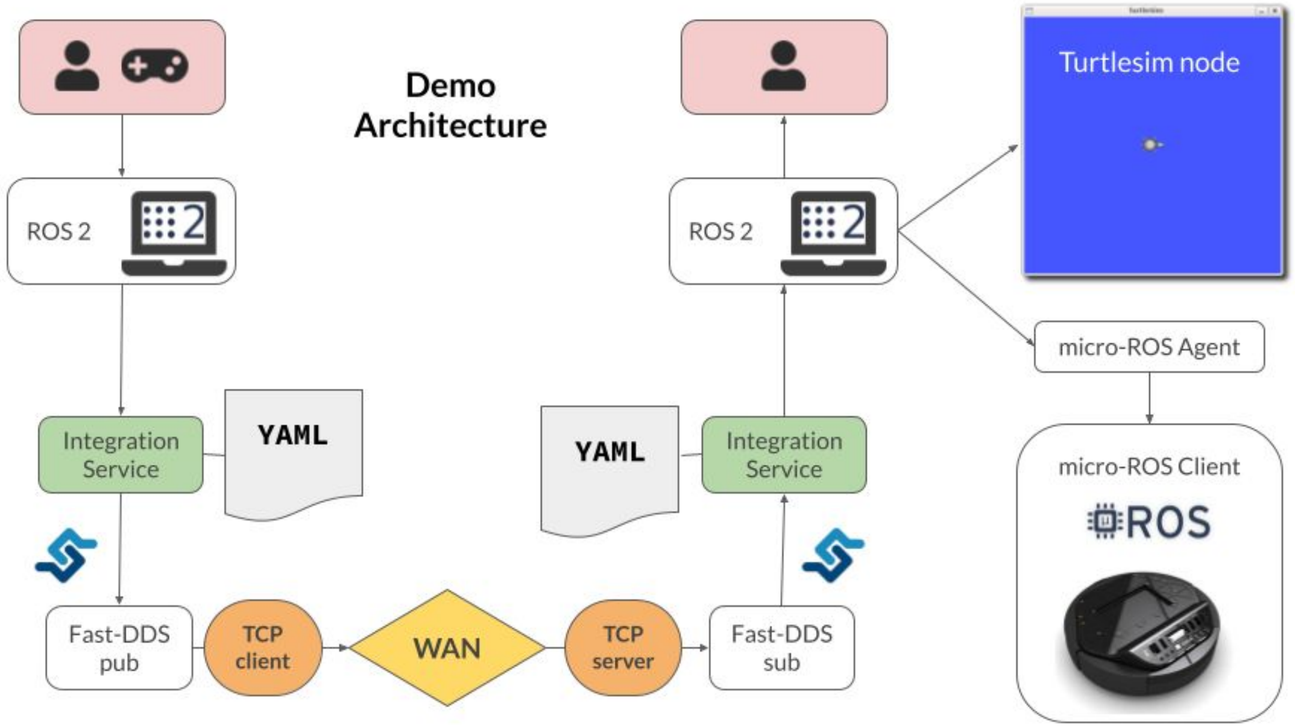
… and more are to come!

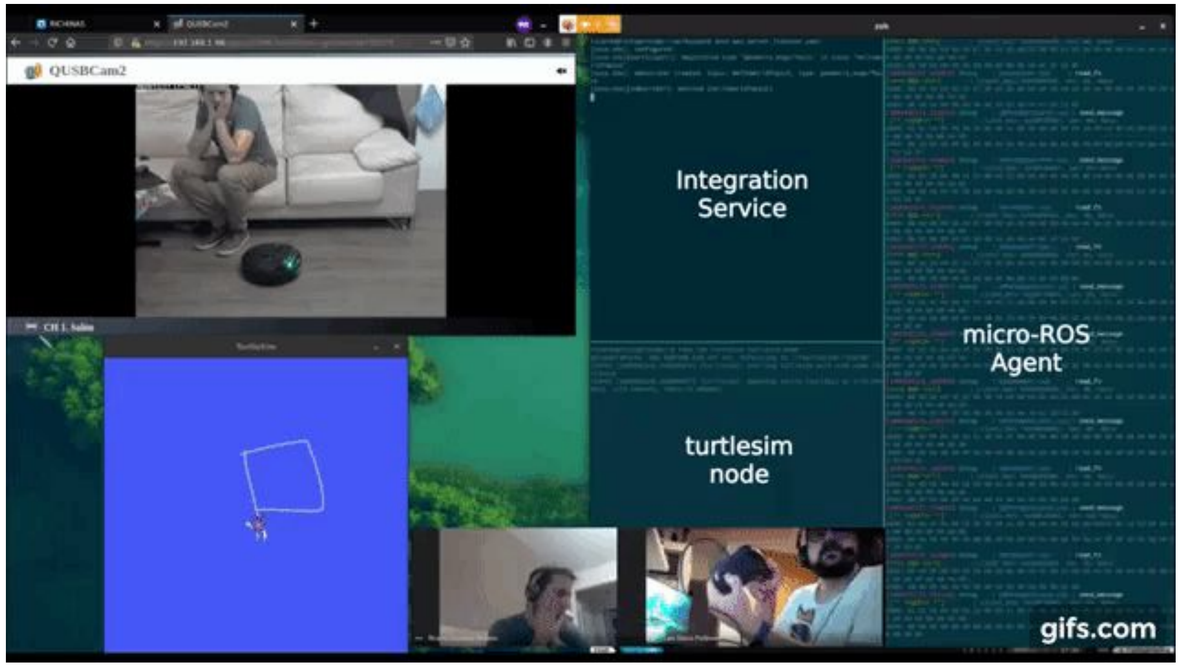
micro-ROS & MoveIt 2



Demo I

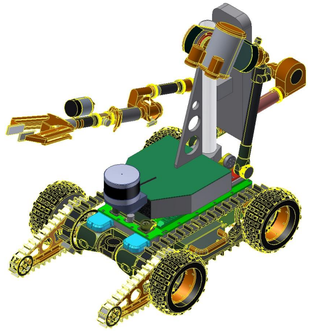
ROS 2 to micro-ROS TCP tunneling via Integration Service!

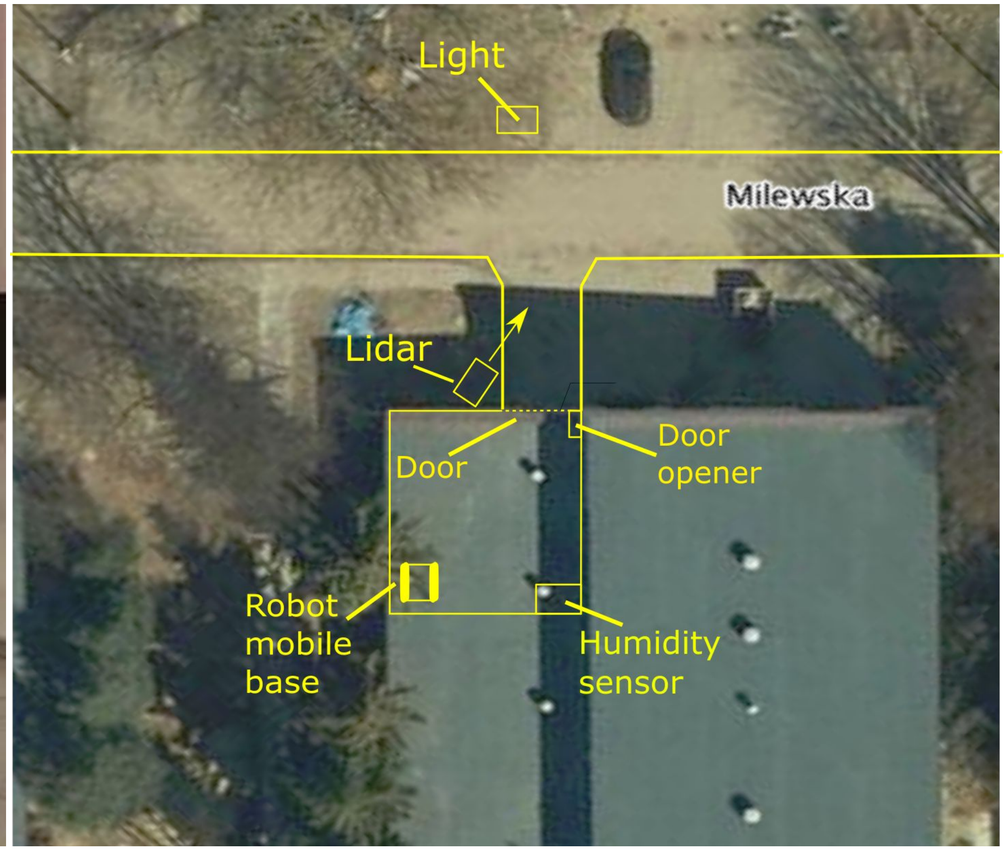




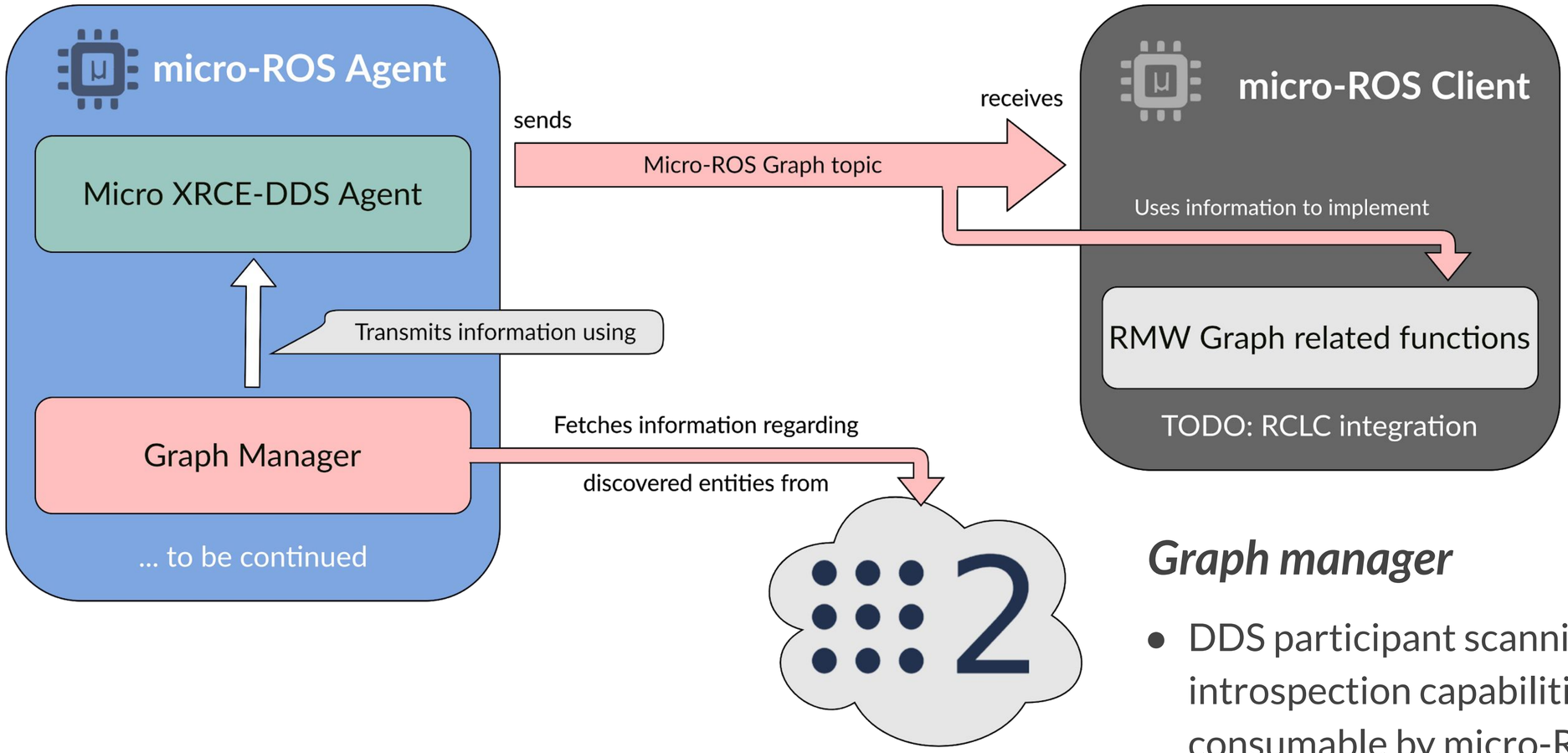
Demo II

micro-ROS enabling smart warehouses duties





Graph support



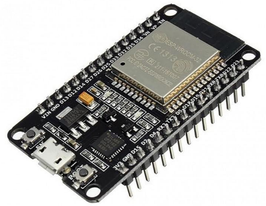
Graph manager

- DDS participant scanning the network: provides introspection capabilities to user. ROS 2 topology consumable by micro-ROS
- micro-ROS topology info available to ROS 2

WIP: P2P functionality

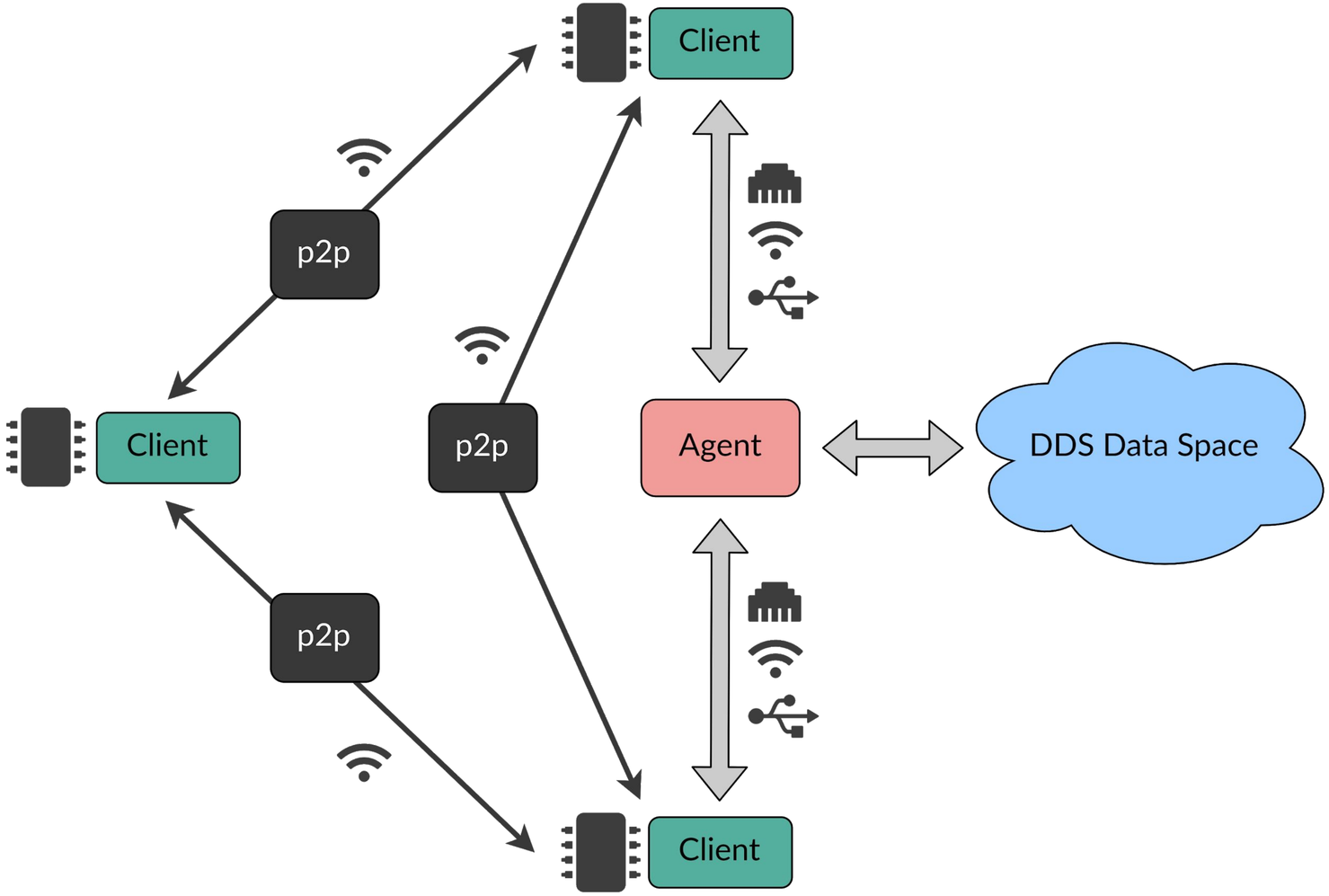
P2P prototype

- Clients send info about themselves on broadcast
- Clients can choose whether to connect via the Agent or by P2P [WIP]
- At present, P2P offers limited set of functionalities
- Tried on:



ESP32

FreeRTOS



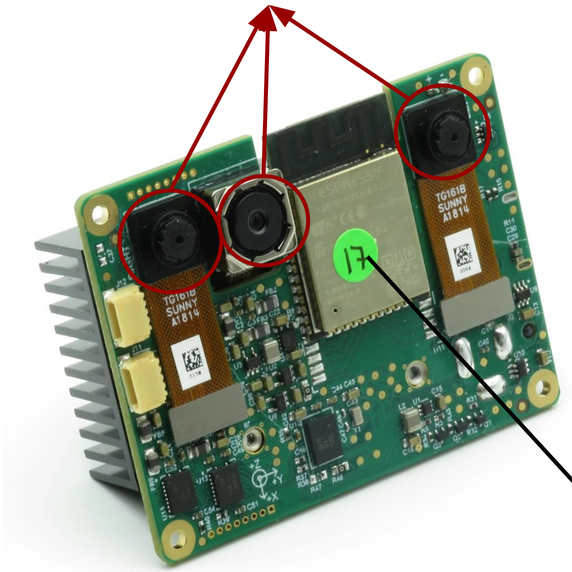
WIP: micro-ROS goes AIoT!

DEPTH ai

Open-source platform - custom hardware, firmware, software & AI training - that combines ***neural inference***, ***depth vision***, and ***feature tracking***.

Enables for ***embedded artificial intelligence*** and ***spatial AI/CV***.

Cameras



ESP32

freeRTOS

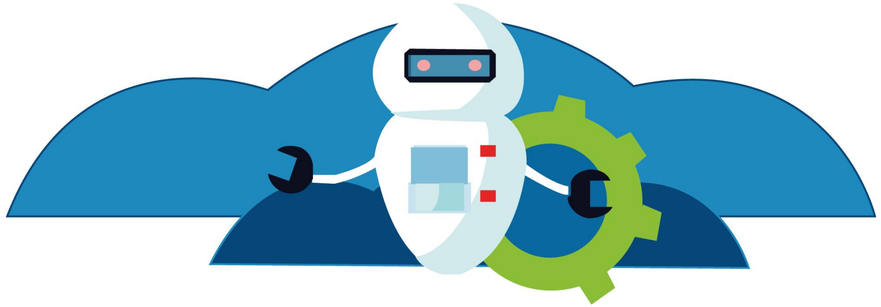
LUX onjs

+

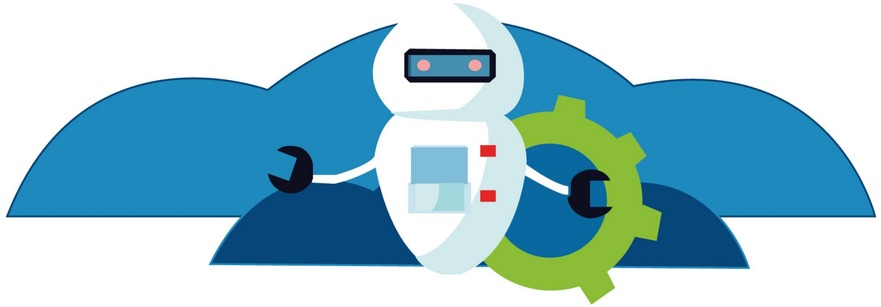
ROS

WIP

micro-ROS on DepthAI via ESP32 support: combining embedded artificial intelligence with ROS 2 ecosystem!



Thanks for your attention!



Q&A time

QoS

Two possibilities for entities creation:

- ***By XML (on Client) - default***
- ***By reference (on Agent) - allows full use of QoS***

Users can write custom QoS on the Agent's side. Each entity has its own label and the Client creates the entities using this reference label.

Advantages of using creation by reference:

- Reduces memory consumption of micro-ROS Client inside the MCU.
- Full set of DDS QoS available

```
rclc_publisher_init_default(&publisher, &node, ROSIDL_GET_MSG_TYPE_SUPPORT(std_msgs, msg,
Int32), "my_qos_label");
rcl_publish(&publisher, &msg, NULL);
```

```
<data_writer profile_name="my_qos_label__dw">
    <historyMemoryPolicy>PREALLOCATED_WITH_REALLOC</historyMemoryPolicy>
    <qos>
      <reliability>
        <kind>RELIABLE</kind>
      </reliability>
    </qos>
    <topic>
        <kind>NO_KEY</kind>
        <name>rt/my_topic_name</name>
        <dataType>std_msgs::msg::dds_::Int32_</dataType>
        <historyQos>
            <kind>KEEP_LAST</kind>
            <depth>20</depth>
        </historyQos>
    </topic>
</data_writer>
```

Memory profiling

Transport: UDP

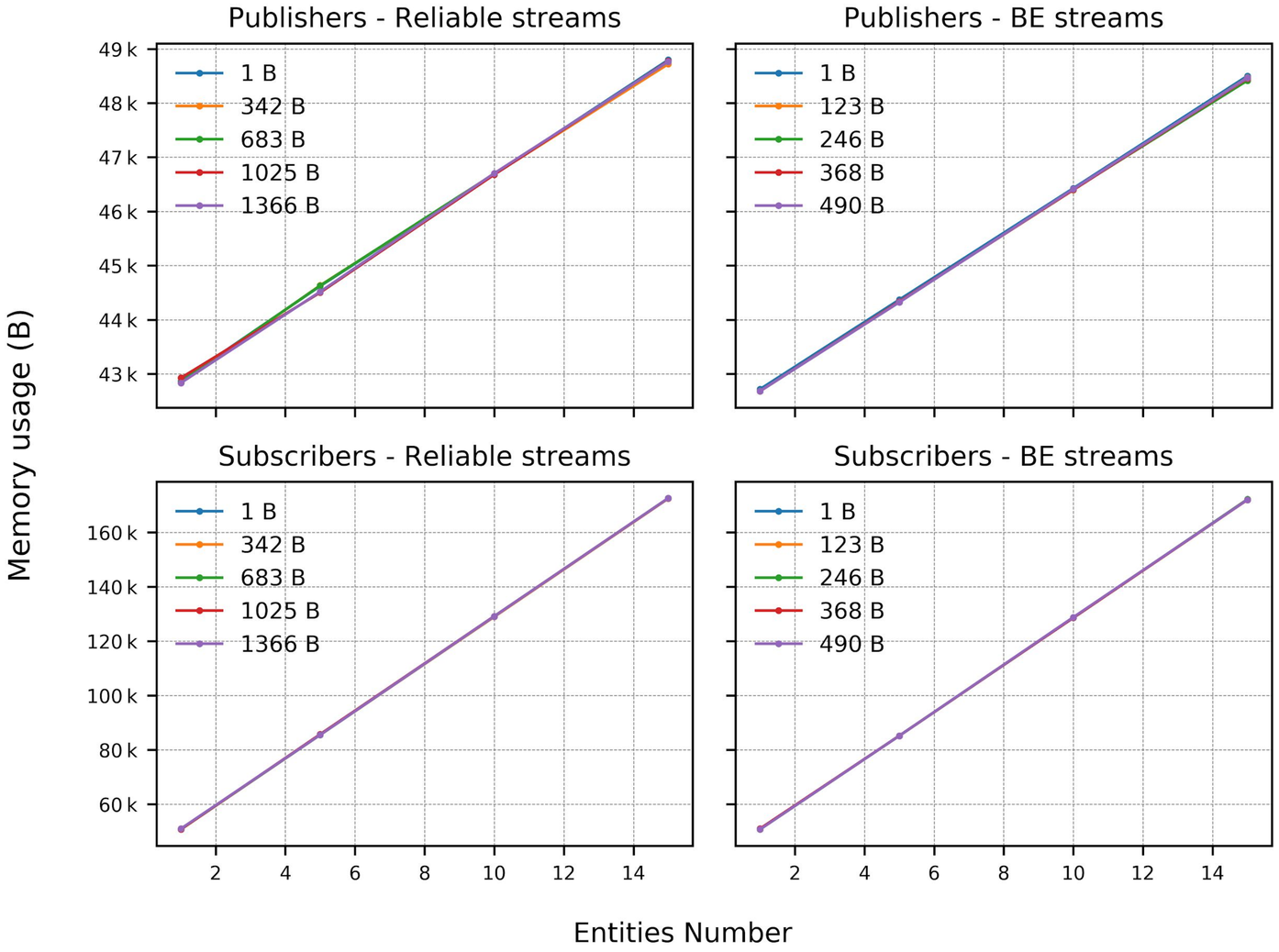
Creation: by XML

RMW history = 4

MTU = 512 B

XRCE history = 4

- *Total memory consumed by 1 pub ~ 400 B*
- *Total memory consumed by 1 sub ~ 8700 B*



Memory profiling

Transport: UDP

Creation: by XML

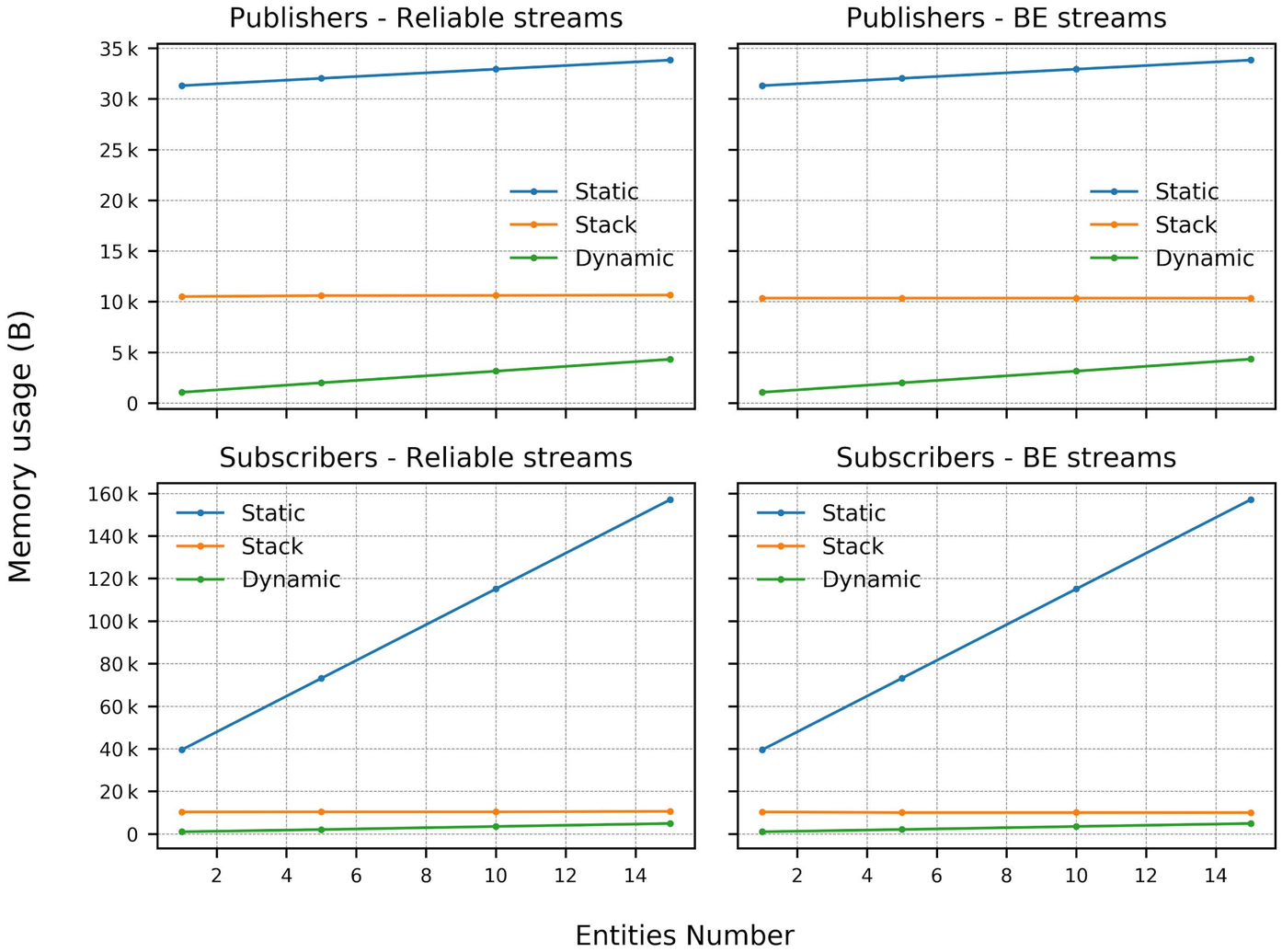
RMW history = 4

MTU = 512 B

XRCE history = 4

Overall memory:

- *Static*
- *Stack*
- *Dynamic*



Memory profiling

Transport: UDP

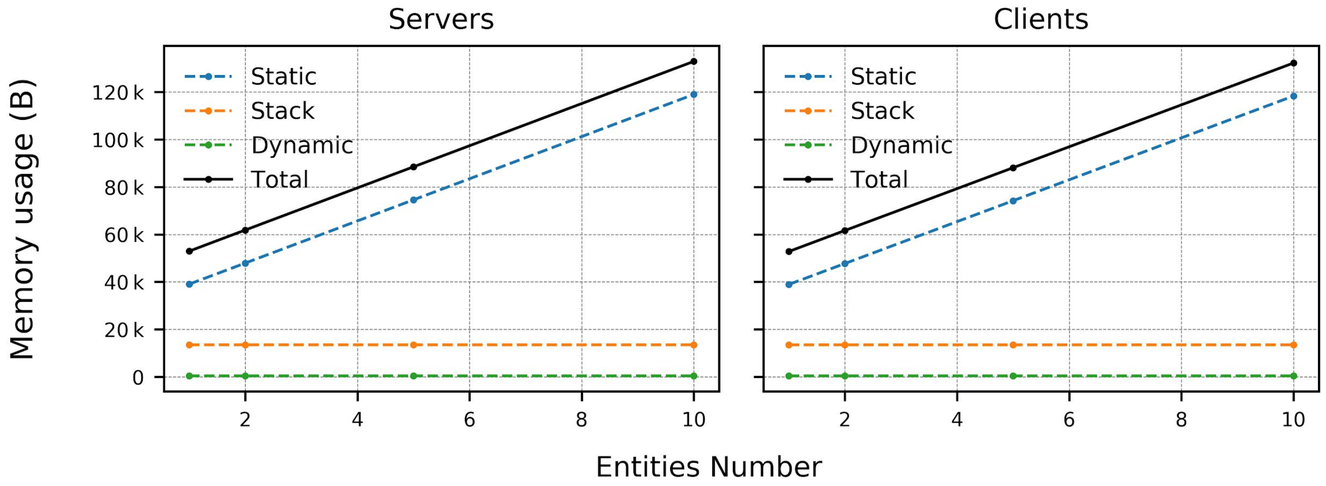
Creation: by XML

Comm stream: Reliable

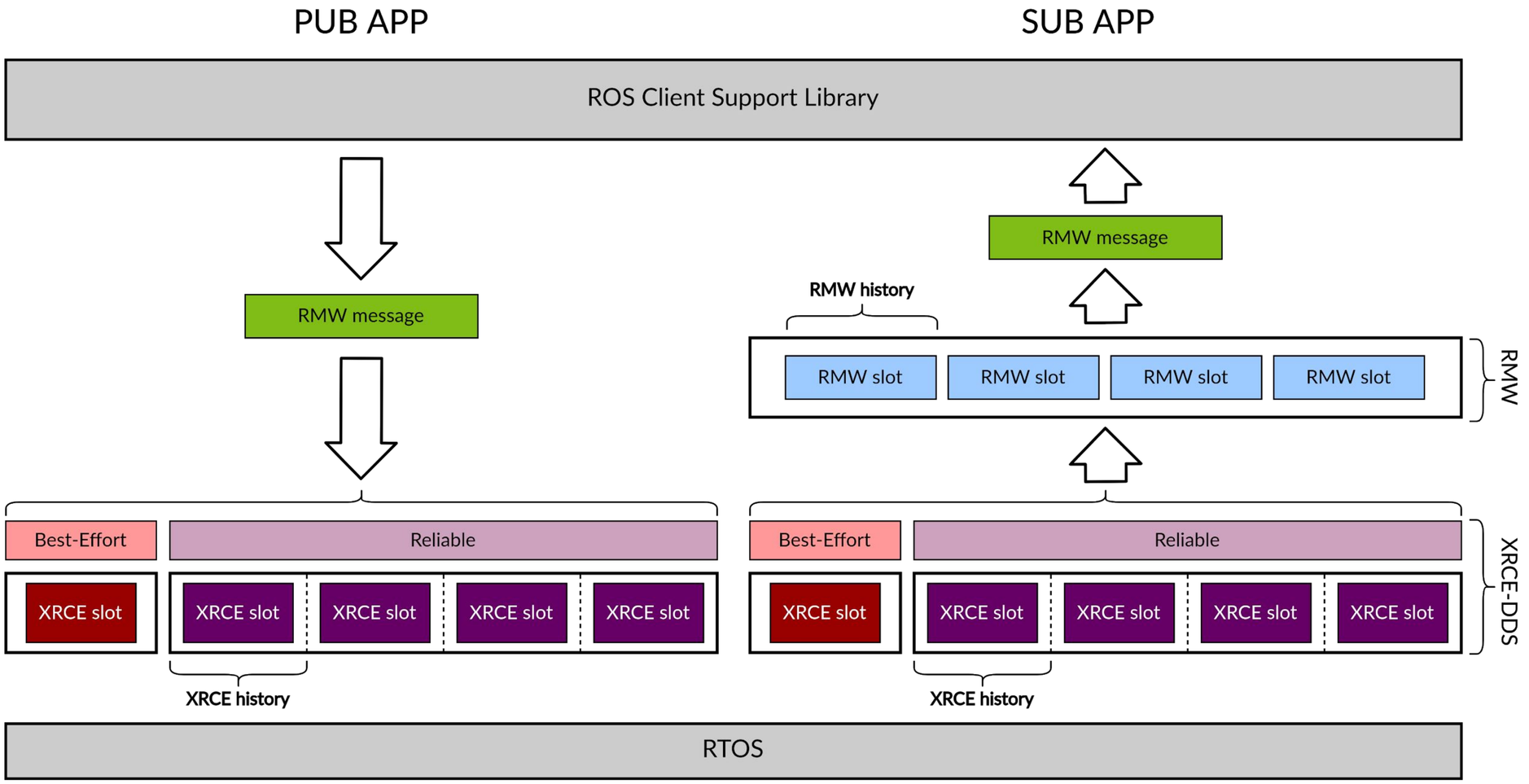
RMW history = 4

MTU = 512 B

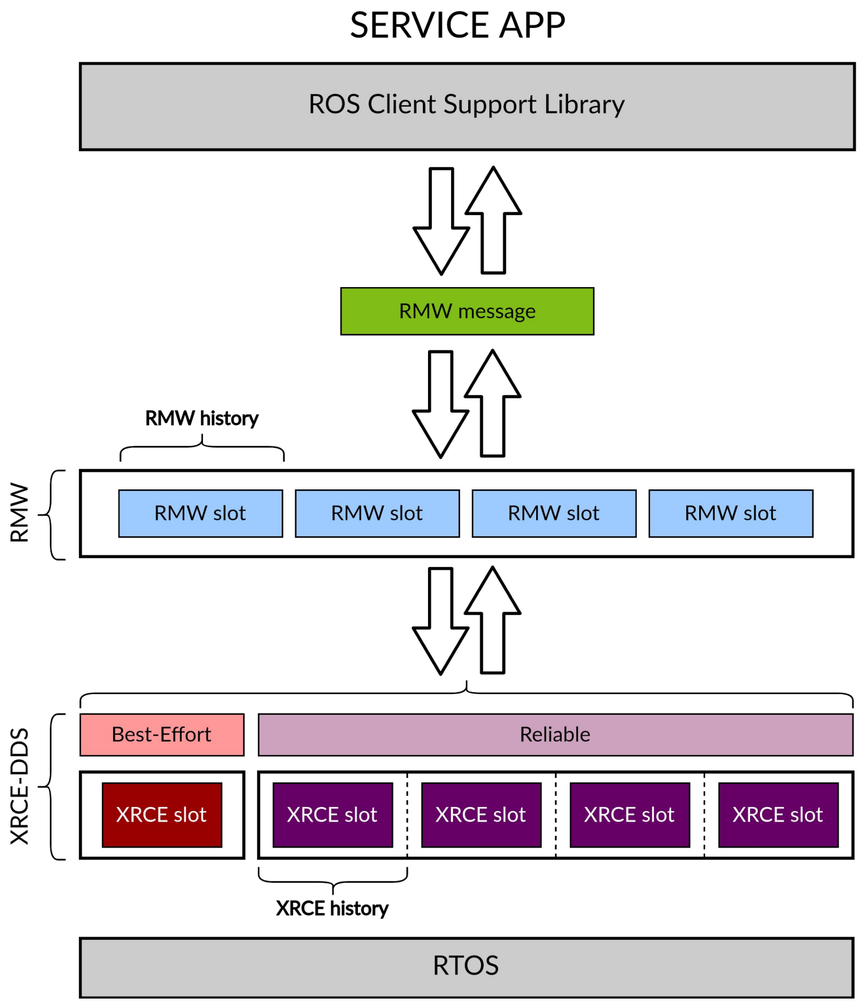
XRCE history = 4



Memory management



Memory management

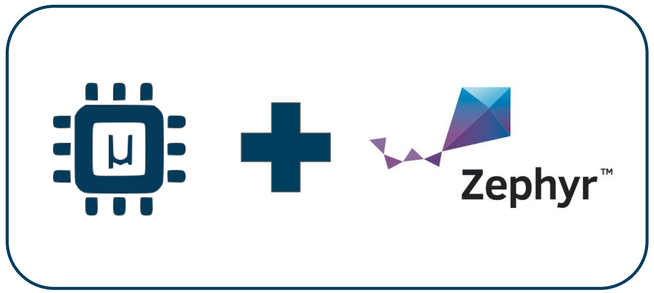


micro-ROS hands-on

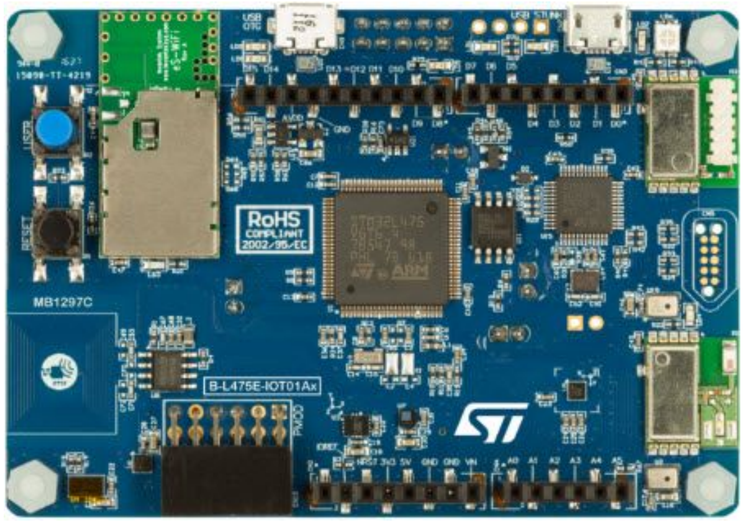
Pablo Garrido - eProsima

December 16th, 2020

Basic Zephyr sensors demo



STM32L4
Discovery kit IoT



LED

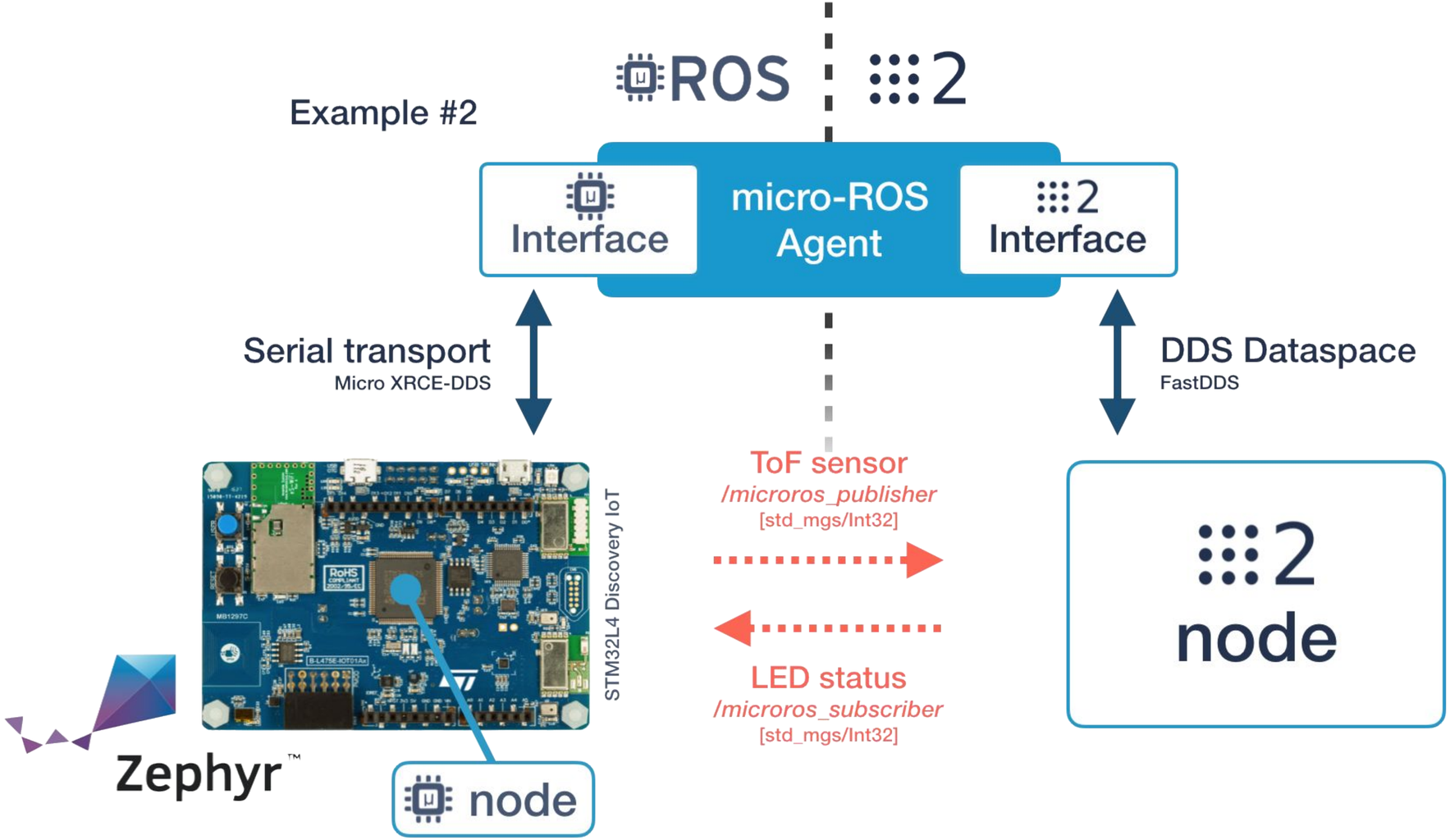
IMU Sensor

ToF Sensor

https://github.com/micro-ROS/micro_ros_showcase_demo

Basic Zephyr sensors demo

Example #2

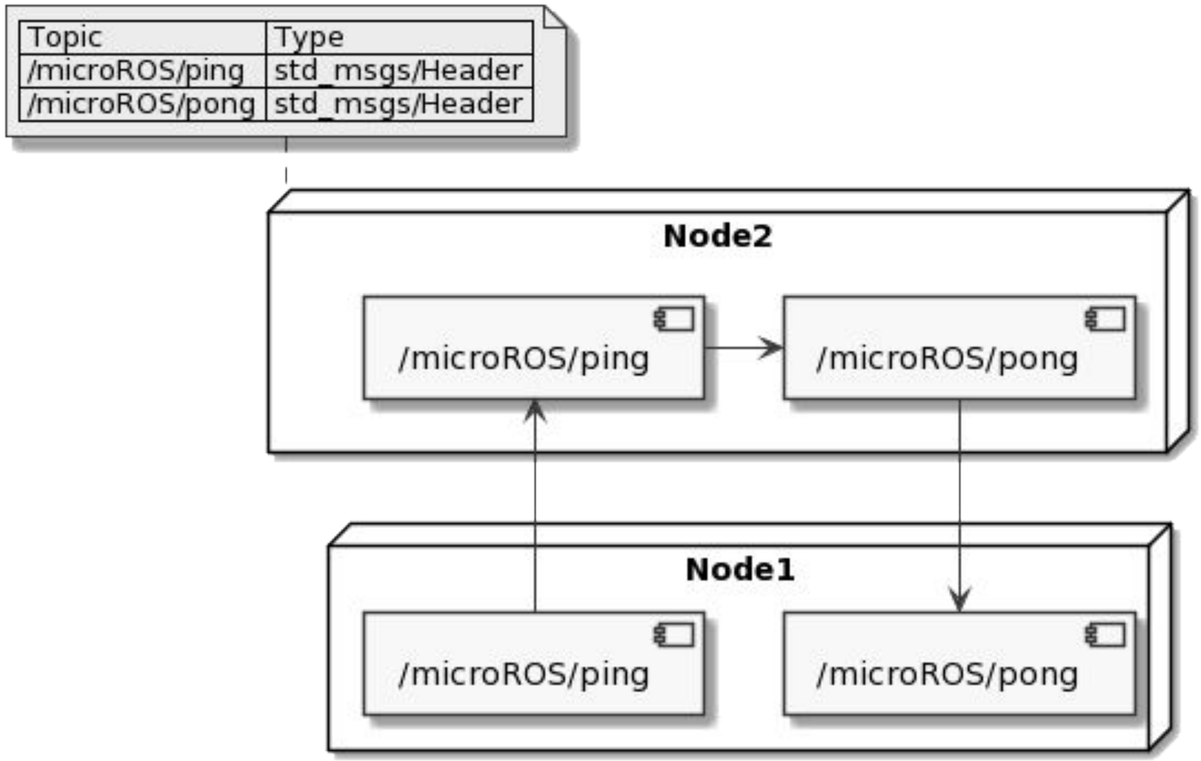


https://github.com/micro-ROS/micro_ros_showcase_demo

Ping-pong demo

Ping pong demo

- All nodes sends **ping** messages every 5 seconds
- When a node receives a **ping** message with a different header it answers with a **pong**
- A node counts all the peers based on the received **pong** messages



https://github.com/micro-ROS/micro_ros_showcase_demo