ROSCon 2022
2022/10/21

m2

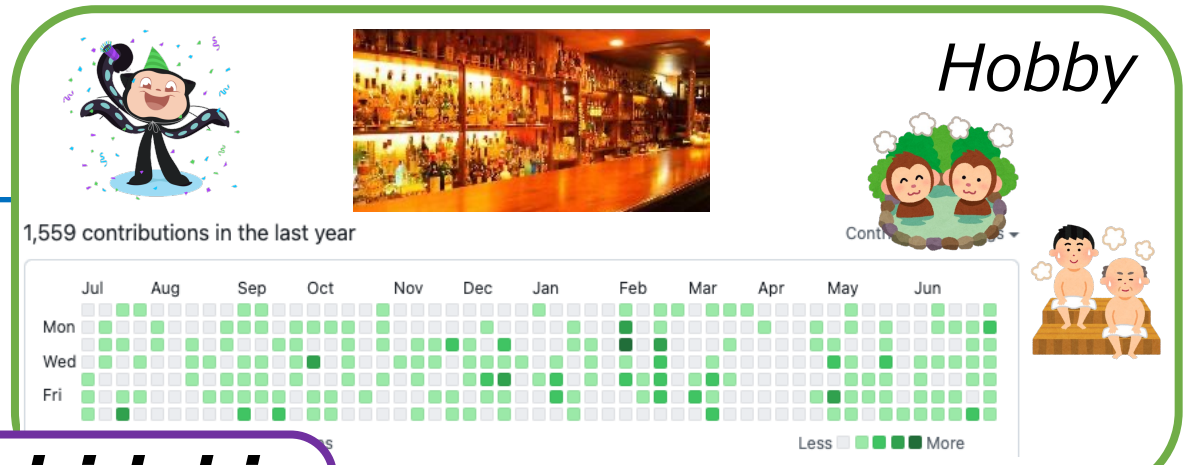
yet another runtime environment onto embedded devices

Hideki Takase, Shintaro Hosoai,
Yoichiro Hibara, Haruaki Tanaka (The University of Tokyo),
Hidetoshi Yugen (Kyoto University), Shoji Morita (eSOL Co.,Ltd.)

A part of this work is supported by JST CREST JPMJCR21D2 and the commissioned research (04001) by National Institute of Information and Communications Technology (NICT) , Japan.

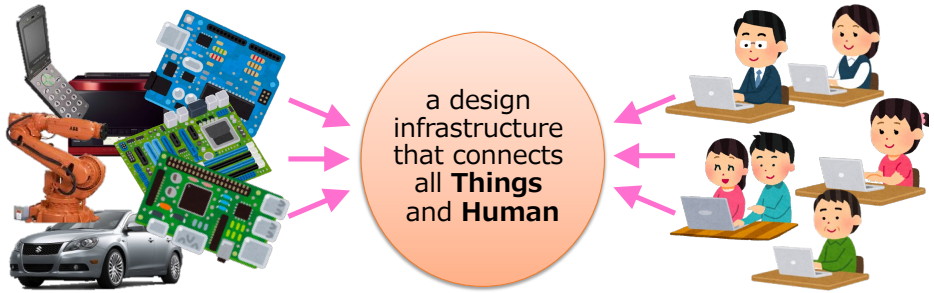
Affiliation

Hobby



@takasehideki

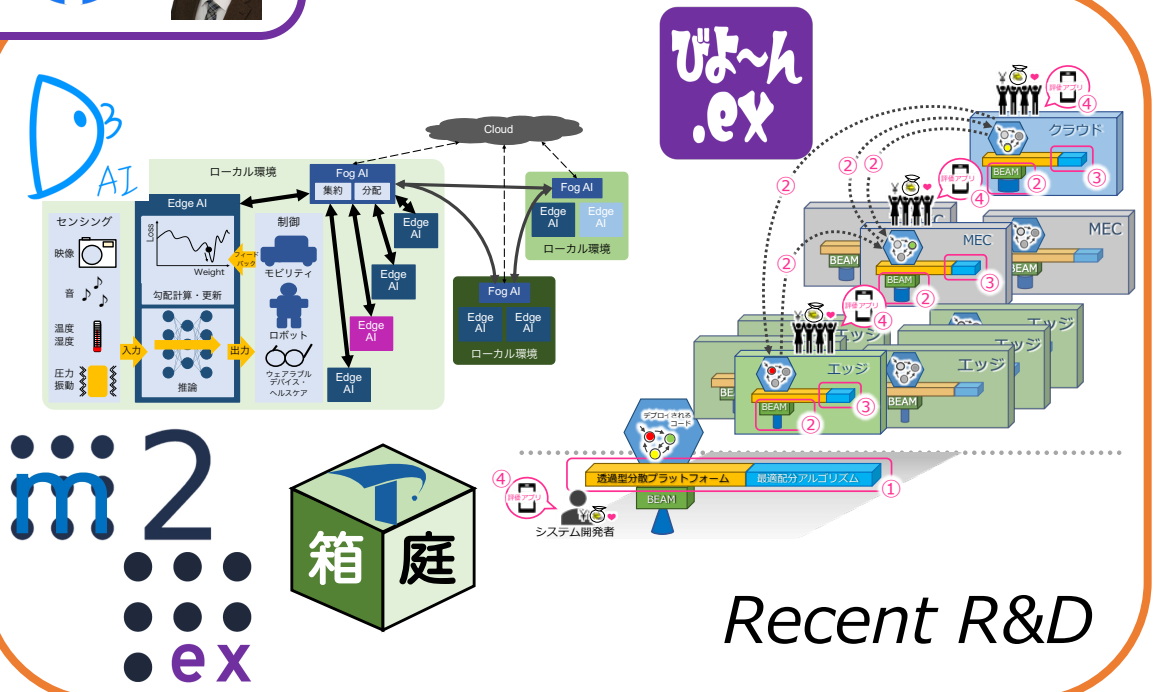
Cutting-Edge Platform and Design Methodology for embedded/IoT Computing



Optimization **Methodology**

Toward a world where **anyone can easily create awesome products**

Mission

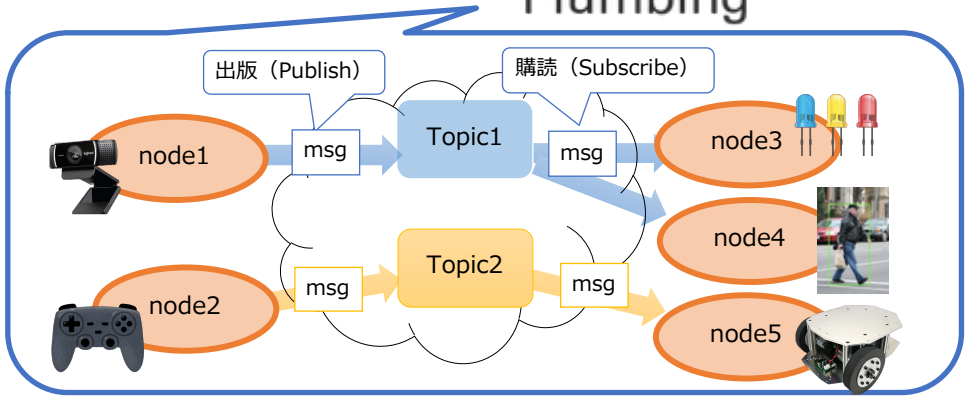


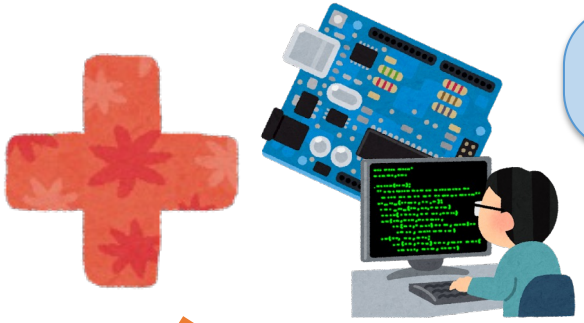
Recent R&D

2 meets EMB for IoT!!

Platform for accelerating the development of robot systems

http://www.ros.org/about-ros/

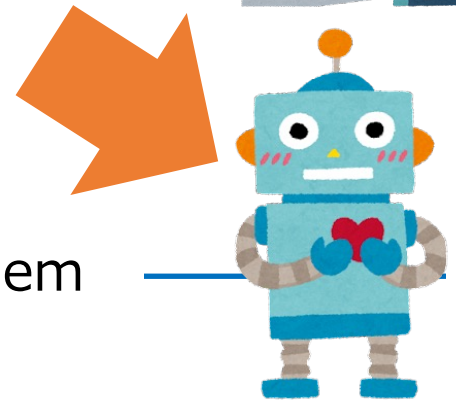




Integration of Embedded Technology

- **comm. performance**
  - responsiveness (latency)
  - real-time performance (variation)
- **Power consumption**

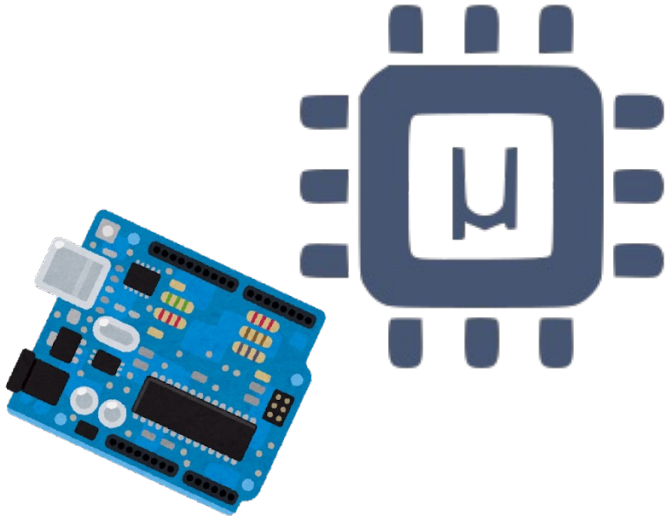
- The essence of ROS is **Plumbing**
  - Loosely coupled arch. of ROS nodes
  - Easy to register, delete, and restore them
  - Basis is pub/sub comm. via **Topic**



ROS 2/DDS communication tech. is also expected to be deployed in the IoT field

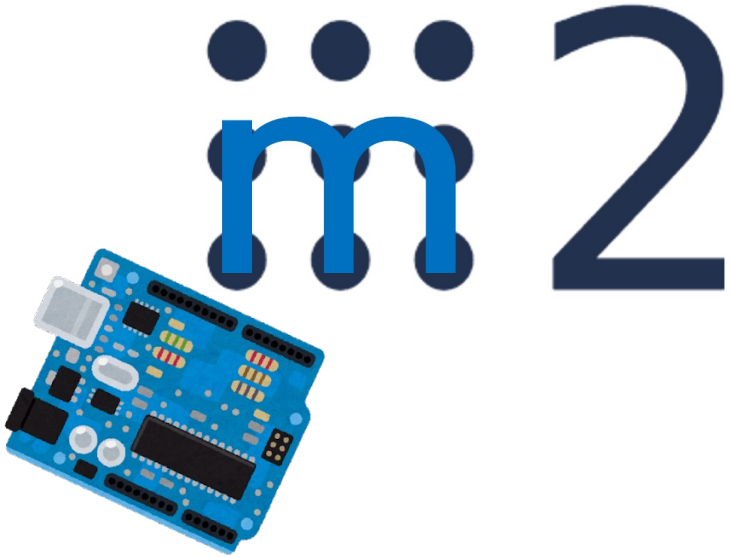
2x How to “embed”??

native ROS 2
on **SBC**



micro-ROS
on EMB board

various kernels & boards
serial, UDP, Bluetooth trans.
fully compatible with **rclc**
XRCE-DDS (**with agent**)

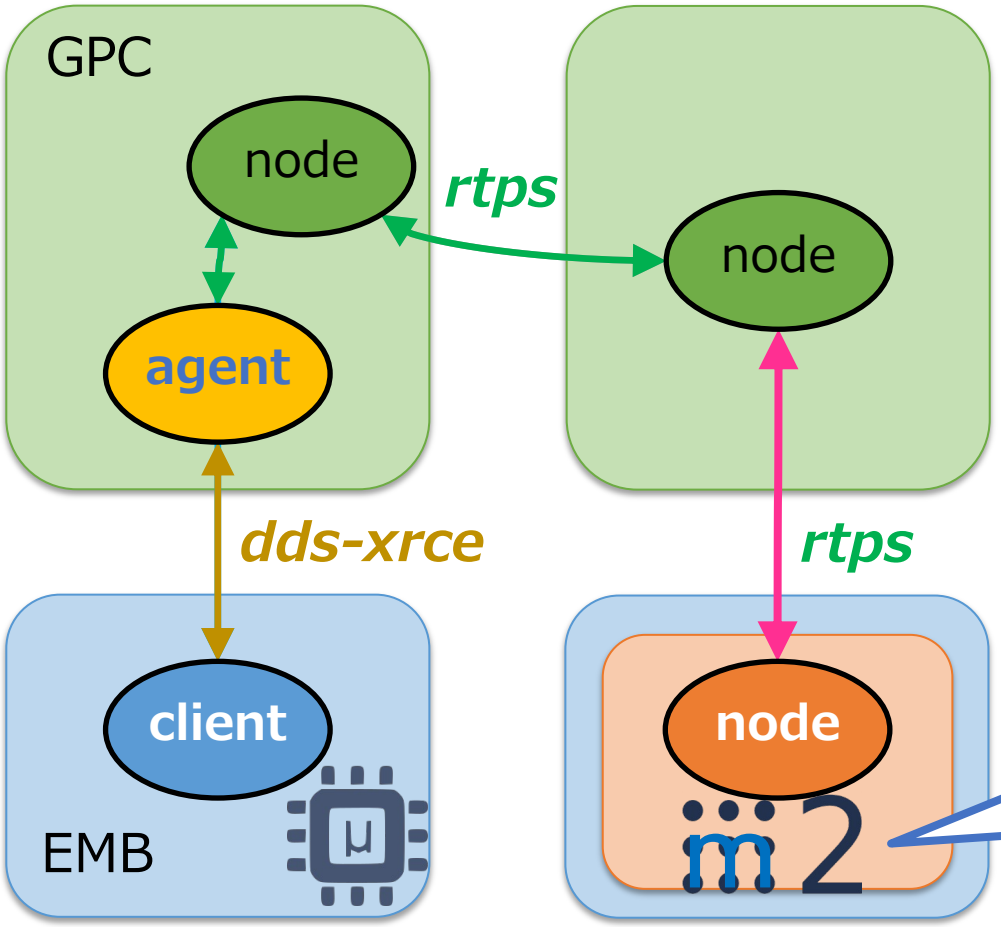


mROS 2
on EMB board

better perf. & mem. usage
partially compatible with rclcpp
only **for Topic comm.**,
and many unsupported features
such as QoS, Service,,,

mROS-base/mros2

Dive into comm.

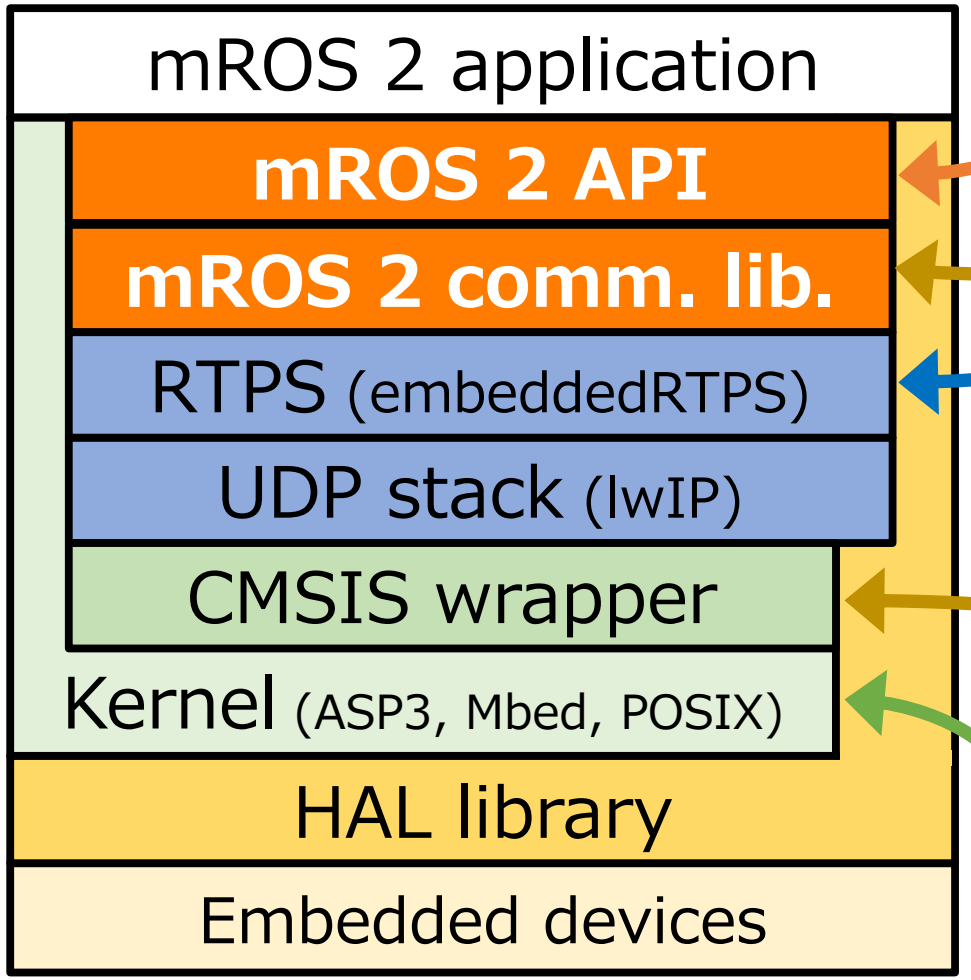


- **RTPS**: comm. protocol of DDS
  - SPDP/SEDP: autonomously searches/establishes communication partners/paths (w/o master)
  - GPOSes are essential!! (ROS 2 itself)
- **micro-ROS**: de-fact of EMB platform
  - employ Micro-XRCE-DDS (default)
  - ➢**agent** is necessary as the master

RTPS communication directly from EMB without an agent!!

※GPC: General Purpose Computer
EMB: Embedded Micro-controller Board

Software Stack



pub/sub messaging for Topic
partially compliant with rclcpp

autonomous communication
in accordance with RTPS specification

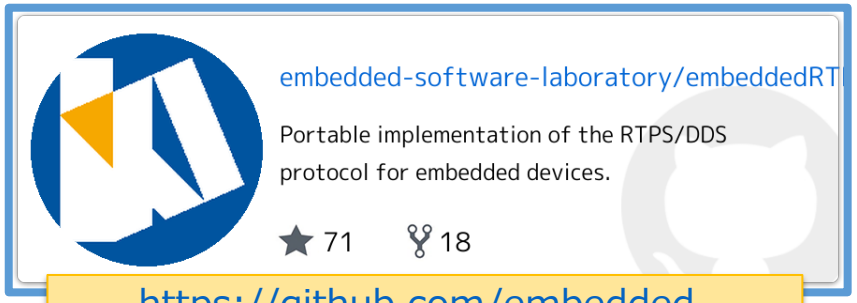
lightweight and efficient process
by C/C++ for <~200MHz / <~1MB

real-time kernels for EMB
contributes perf. and mem. usage

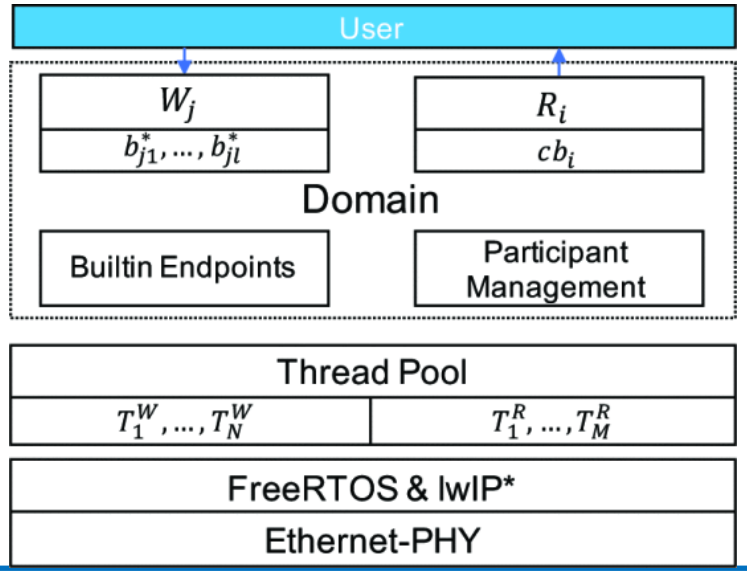
mROS-base/mros2

Note: embeddedRTPS [A. Kampmann+ ITSC'2019]

- Portable RTPS implementation by C++
  - lwIP (Raw Mode) for UDP/IP
  - Micro-CDR for serialization
  - some code dependencies with FreeRTOS
- Main features and contributions
  - Discovery: SPDP & SEDP features
  - Interoperability: tested with FastDDS 2.3.1
  - QoS Policies: support best-effort & reliable
  - UDP Multicast: support multicast locators
    - ✓Message size is limited to buffer size of lwIP



https://github.com/embedded-software-laboratory/embeddedRTPS

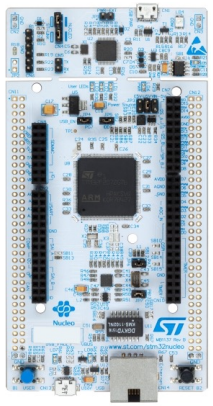


mROS-base/mros2

Getting started!!

- Steps for embedded board

```
$ git clone https://github.com/mROS-base/mros2-mbed
$ cd mros2-mbed
$ ./build.bash all NUCLEO_F767ZI echoback_string
$ picocom -b 115200 /dev/ttyACM0
```



- Step for the host

```
$ docker run --rm -it --net=host ros:humble /bin/bash ¥
  -c "mkdir -p ~/ros2/src && cd ~/ros2/src
  git clone https://github.com/mROS-base/mros2-host-examples &&
  cd mros2-host-examples &&
  colcon build --packages-select mros2_echoreply_string &&
  cd ../ && source install/setup.bash &&
  ros2 run mros2_echoreply_string echoreply_node”
```

mROS-base/mros2

workspace > echoback_string > app.cpp

```cpp
#include "mbed.h"
#include "mros2.h"
#include "std_msgs/msg/string.hpp"
#include "EthernetInterface.h"

void userCallback(std_msgs::msg::String *msg)
{
  printf("subscribed msg: '%s'\r\n", msg->data.c_str());
}

int main() {
  EthernetInterface network;
  network.set_dhcp(false);
  network.set_network(IP_ADDRESS, SUBNET_MASK, DEFAULT_GATEWAY);
  nsapi_size_or_error_t result = network.connect();

  printf("mbed mros2 start!\r\n");
  printf("app name: echoback_string\r\n");
  mros2::init(0, NULL);
  MROS2_DEBUG("mROS 2 initialization is completed\r\n");

  mros2::Node node = mros2::Node::create_node("mros2_node");
  mros2::Publisher pub = node.create_publisher<std_msgs::msg::String>("to_linux", 10);
  mros2::Subscriber sub = node.create_subscription<std_msgs::msg::String>("to_stm", 10, userCallback);

  osDelay(100);
  MROS2_INFO("ready to pub/sub message\r\n");

  auto count = 0;
  while (1) {
    auto msg = std_msgs::msg::String();
    msg.data = "Hello from mros2-mbed onto " + quote(TARGET_NAME) + ": " + std::to_string(count++);
    printf("publishing msg: '%s'\r\n", msg.data.c_str());
    pub.publish(msg);
    osDelay(1000);
  }

  mros2::spin();
  return 0;
}
```

Currently Supported

mROS-base/mros2-asp3-f767zi

reference implementation of mROS 2 for STM32 NUCLEO-F767ZI with TOPPERS/ASP3 kernel

★ 14 2

POSIX

ubuntu

mROS-base/mros2-mbed

reference implementation of mROS 2 for Mbed OS

★ 29 4

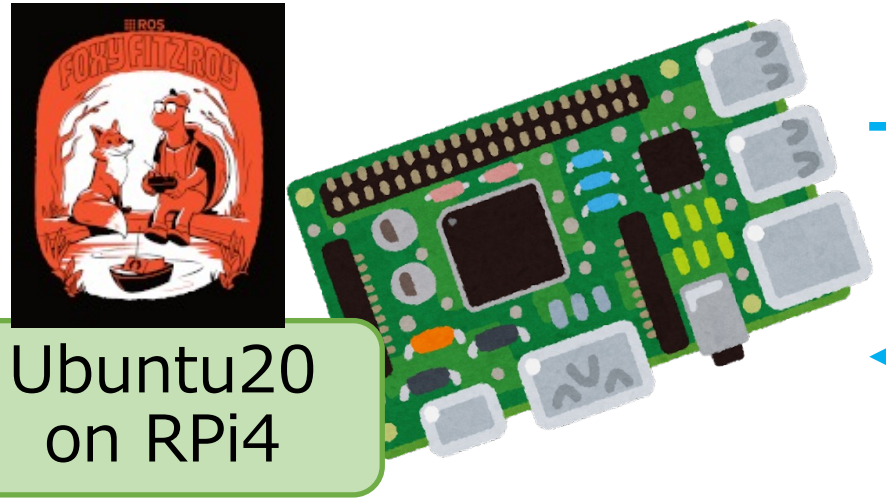
mROS-base/mros2-posix

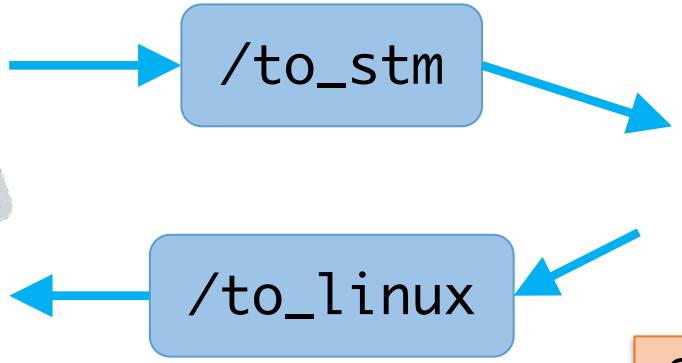
reference implementation of mROS 2 for POSIX layer

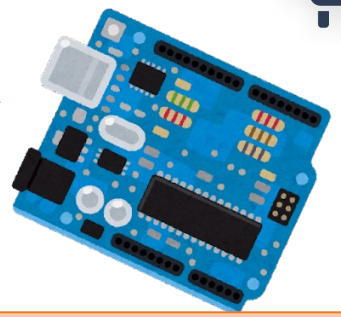
★ 0 1

mROS-base/mros2

Evaluation mros2@v0.3.1 with Foxy







STM32 NUCLEO-F767ZI

Round-Trip Time for UInt16, Twist, String by rclcpp::WallTimer.get_clock() on the host

	uros-serial	uros-udp	uros-rtps	mros2-asp3	mros2-mbed
API	rclc (galactic)			mROS 2 API v0.3.1	
RTPS	Micro XRCE DDS		embeddedRTPS		
protocol	USART	UDP	RTPS on UDP		
RTOS	FreeRTOS v2			TOPPERS/ASP3	Mbed OS 6
compiler	8.3.1		9.3.1	7.3.1	10.3.1

mROS-base/mros2

https://github.com/mROS-base/eval/releases/tag/v0.1.1

Evaluation Results

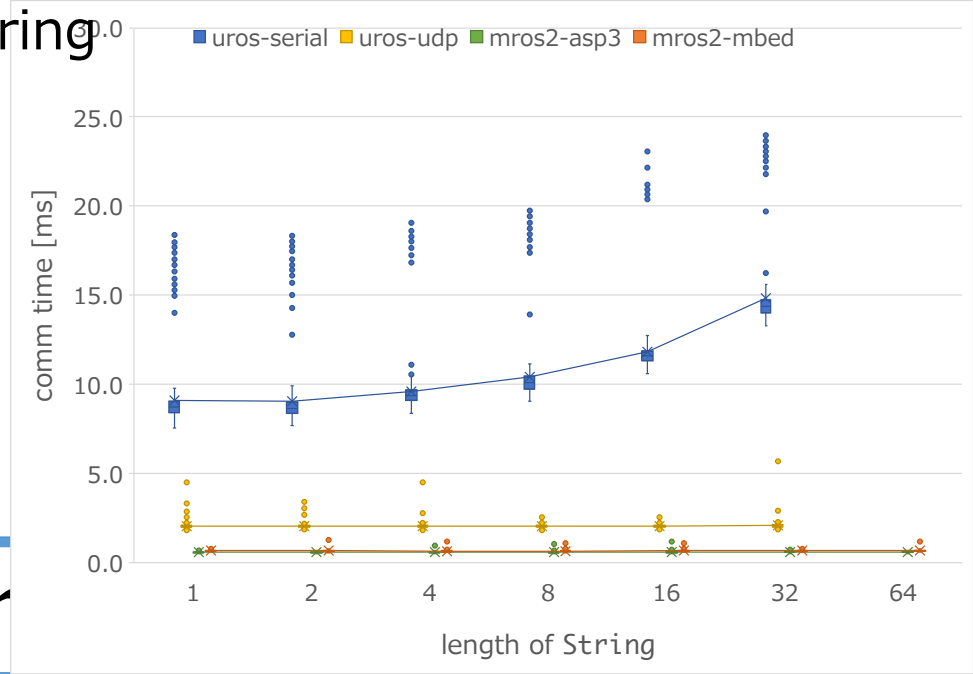
- UInt16 [ms]

	uros-serial	uros-udp	uros-rtps	mros2-asp3	mros2-mbed
avg	11.710	2.109	5.182	0.570	0.646
max	17.370	4.240	11.190	0.810	0.940
min	7.590	1.900	1.940	0.490	0.560
std.p	3.094	0.244	2.684	0.067	0.081

- Twist [ms]

	uros-serial	uros-udp	uros-rtps	mros2-asp3	mros2-mbed
avg	19.530	2.304	5.508	0.593	0.703
max	25.510	11.250	9.860	0.850	0.880
min	15.590	2.050	2.610	0.520	0.640
std.p	3.666	0.904	1.551	0.065	0.042

- String



- mem size (binary for Twist)

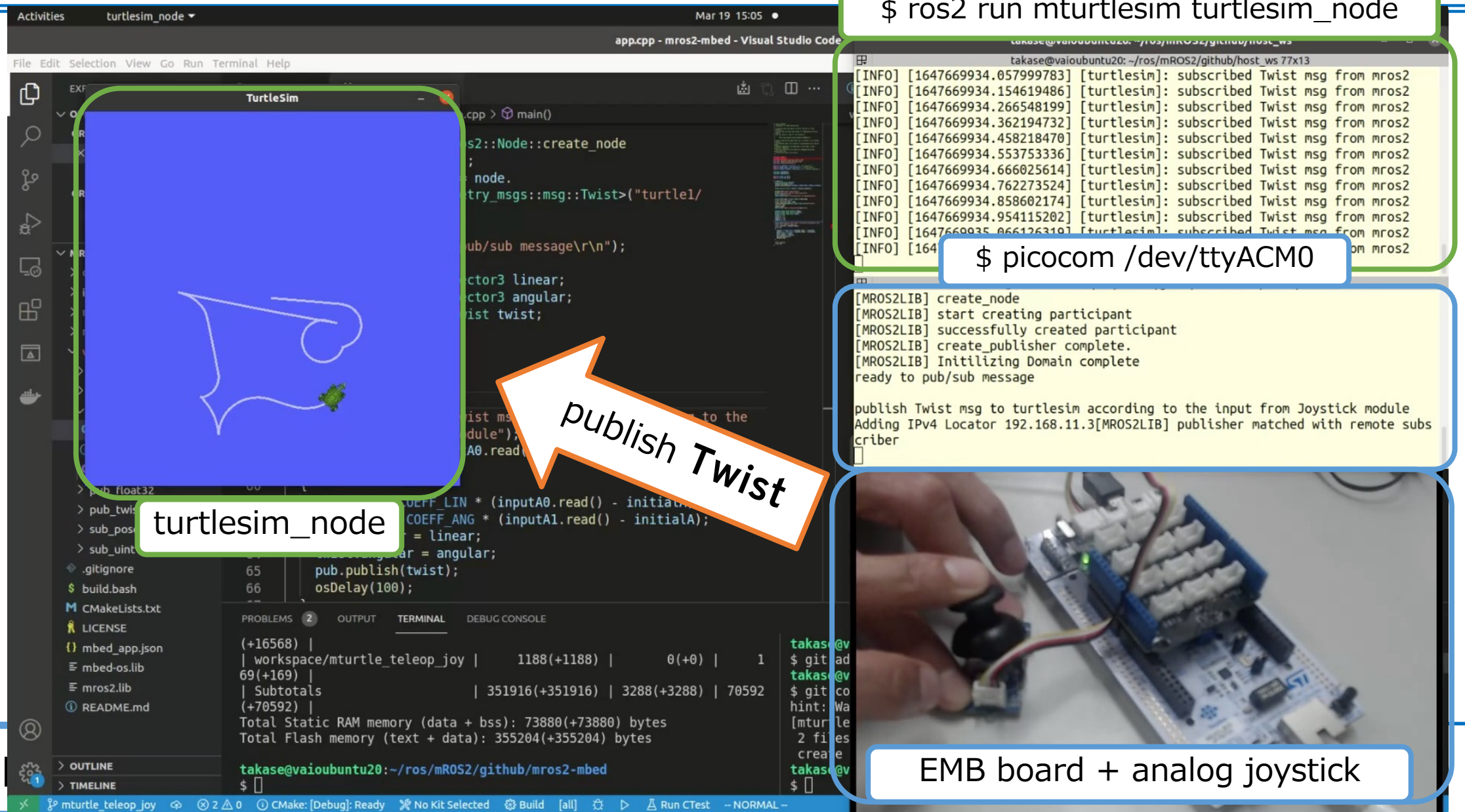
	uros-serial	uros-udp	uros-rtps	mros2-asp3	mros2-mbed
text	209,836	233,656	174,752	90,551	393,312
data	356	356	576	16,632	3,336
bss	110,280	108,160	282,016	111,800	70,688
total	**320,472**	**342,172**	**457,334**	**225,983**	**469,336**

論よりRUN!!

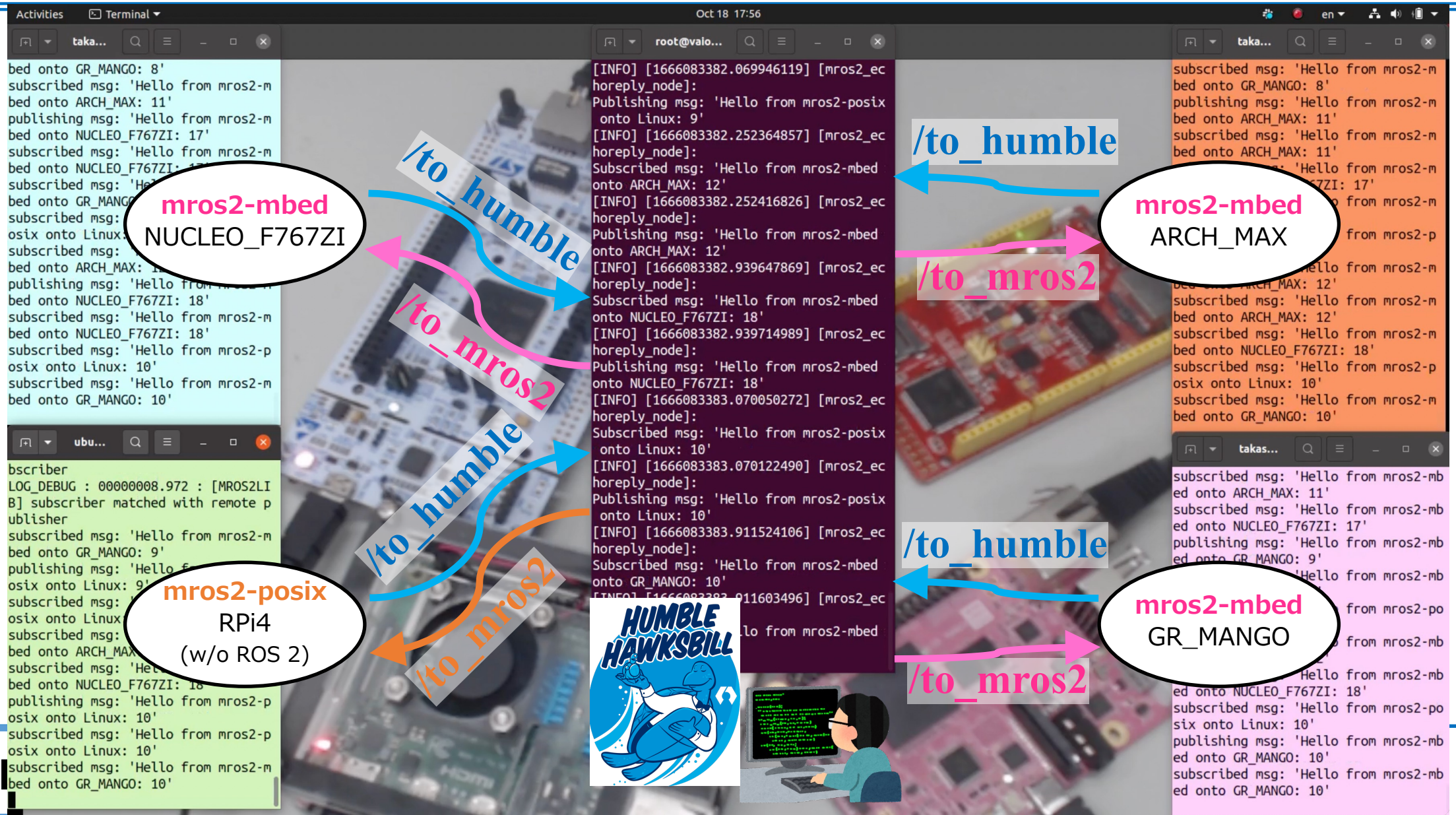
"ron yori run"

The RUN is mightier than the word

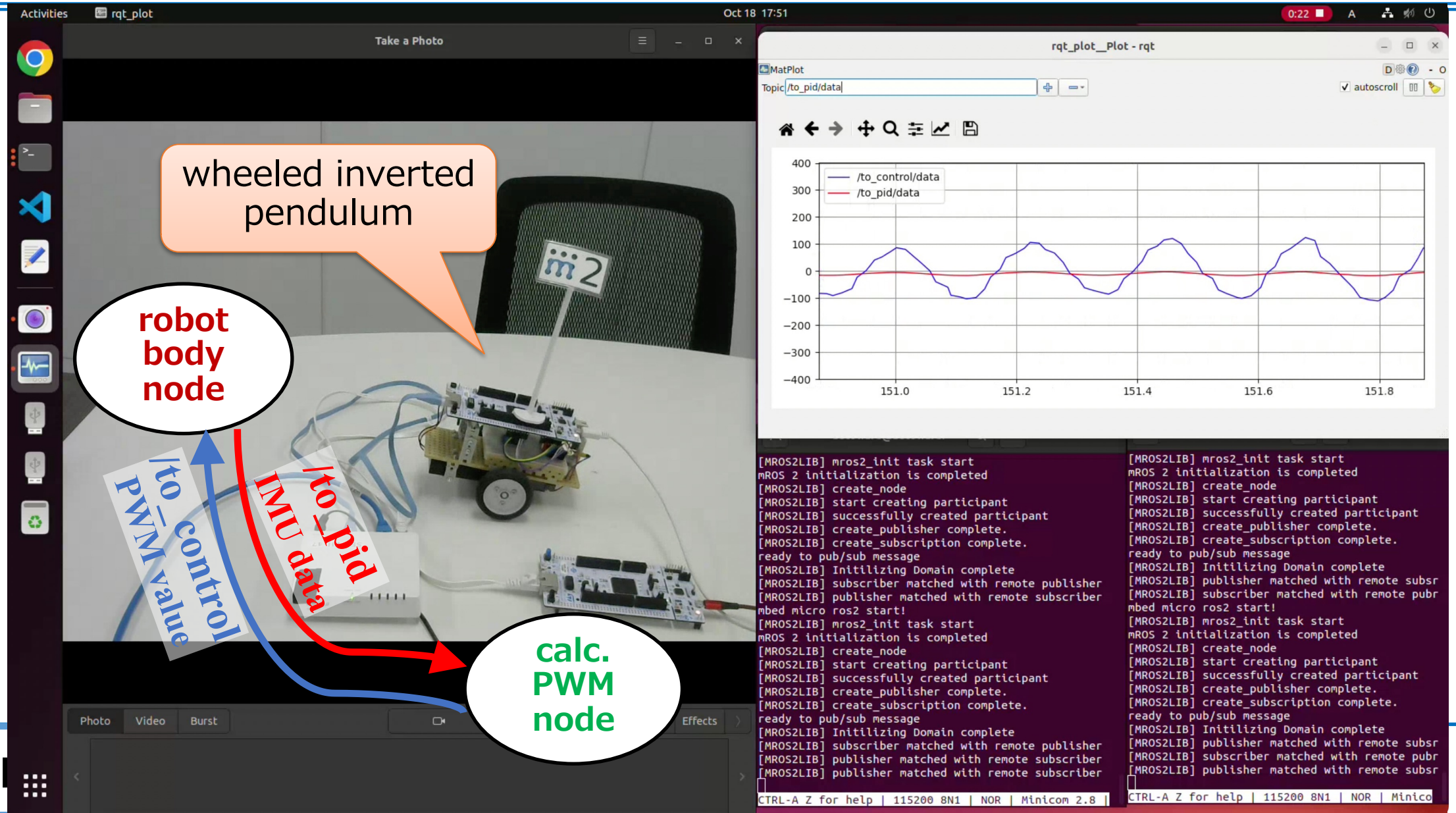
https://twitter.com/takasehideki/status/1505066116921524228



論よりRUN!! part II



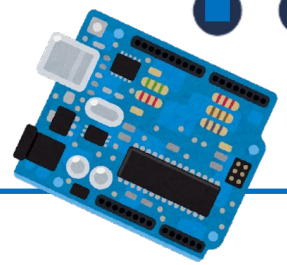
論よりRUN!! Part III powered by eSOL



Conclusion

- Agent-less and lightweight runtime environment for ROS 2
- Our Contribution
  - mROS 2 enables programs running on embedded devices to communicate autonomously with nodes on the native ROS 2
  - mROS 2 would contribute to the construction of distributed robot systems with excellent communication performance

If you wanna communicate **only with Topic**,
please consider to try our mROS 2
as one of the candidates🤔🤔🤔



mROS-base/mros2

Check now!! & What's next??

mROS-base/mros2

agent-less and lightweight communication library compatible with rclcpp for embedded d ...

104 7

https://github.com/mROS-base/mros2

Please give us the Star!
& your contribution!!

- **porting to other boards and kernels**
- implement new targets with POSIX-compliant RTOS
- support QoS control, Service, Action, ...
- design a dedicated board for real robots??

mROS-bas