ROS 2

Sensor sampling and image processing

Roberto Masocco
roberto.masocco@uniroma2.it

University of Rome Tor Vergata
Department of Civil Engineering and Computer Science Engineering

June 7, 2023

TOR VERGATA
UNIVERSITY OF ROME
School of Engineering

Recap

ROS 2 offers a common framework for the development of **robotics software**, providing services for:

- **organizing** and **building** a **distributed** software architecture;
- establishing mostly self-configured **inter-process communication** among modules;
- modules **configuration** and **management**.

The last two lectures will show **real applications** of these tools.

New code examples are available.

This lecture is here.

Roadmap

1. Sensor sampling
2. Image processing
3. Software tools of the trade
4. Composition

Roadmap

1. Sensor sampling

2. Image processing

3. Software tools of the trade

4. Composition

Sensor sampling basics

From theory to practice

Sampling a sensor consists of reading **measurements** from it, to be fed to a control loop or some other subsystem.
It requires:

- the definition of a **sampling frequency**;
- the implementation of an **encoding**;
- the application of **post-processing** steps (*e.g.*, **filtering**).

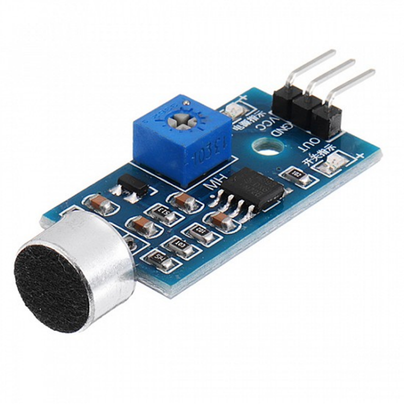

Figure 1: Analog sound sensor.

Sensor sampling with ROS 2

Driver modules

Sampling is generally handled by **microcontrollers**, but when the sampling frequency is not too high, *e.g.*, down to some ms, it can be carried out by a higher-level device.
To implement a ROS 2 sensor sampling module, one has to develop a **driver node**, *i.e.*, an application that:

- configures the **sensor hardware** to run as required;
- ensures **stable sampling frequency** and **low jitter**;
- outputs data with a **standard interface** and **low latency**.

The achievement of the first goal depends on the **sensor**, the second on the **system** (hardware and software!), while the third one is solved by ROS 2 (**messages**, **QoS**).

Must take the best of both worlds: robotics and system programming!

Anatomy of a driver node

Guidelines and best practices

In essence, a **driver node** always consists of:

- an **enable service**, to be called to start or stop the sampling;
- a **hardware configuration** routine, to be run at startup or when enabled;
- a **sampling loop**, to be run at a fixed frequency in a separate **thread**;
- a **publisher** using a common message type and an appropriate QoS policy;
- a set of **parameters** to configure the sensor and the sampling loop;
- **launch files** and **configuration files**, to configure remapping rules and node behaviour.

Roadmap

1. Sensor sampling

2. Image processing

3. Software tools of the trade

4. Composition

Roadmap

1 Sensor sampling

2 Image processing

3 Software tools of the trade

4 Composition

Roadmap

1. Sensor sampling
2. Image processing
3. Software tools of the trade
4. Composition