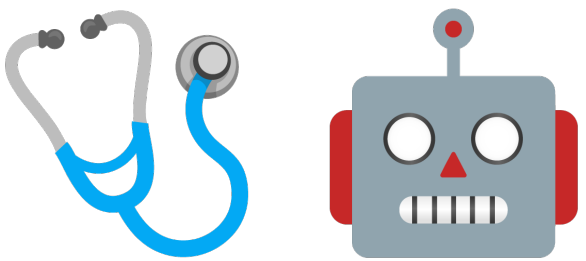
How is my robot? -
On the state of **ROS Diagnostics**

Christian Henkel, Oct 23 2024

ROSCon 2024, Odense



BOSCH

What is `ros/diagnostics`?

Intro

- Quickly observe the current state of your robot
- Deeper analysis of performance parameters
- Historic logging of these information

https://www.pickpik.com/power-plant-control-room-electric-old-instruments-controller-1726

A Brief History

Aug 2008	Nov 2010	Sep 2019
First commit	**REP 107** (link)	**First ROS 2 release (Dashing)**

BOSCH

Collecting evidence
Overview I

diagnostic_updater

- Collect information on the robot state
- Has a state [OK, WARN, ERROR]
- Can have a summary
- Can have additional key-value pairs

```
diagnostic_updater::DiagnosticStatusWrapper
  status;
status.summary(
  diagnostic_msgs::msg::DiagnosticStatus::WARN,
  "Motor RPM may be too low.");
status.add("Motor RPM", rpm);
publish(status);
```

diagnostic_common_diagnostics

- Premade diagnostic features
- Including NTP, CPU, RAM, ...

Getting and Overview
Overview II

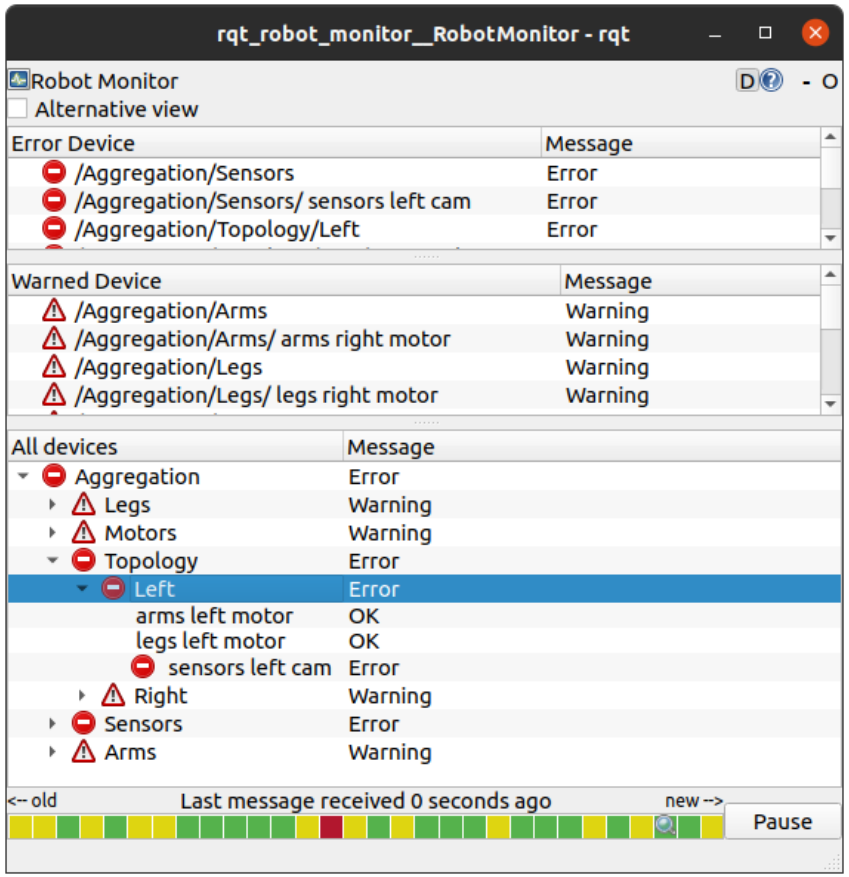
diagnostic_aggregator

- Summarize information according to different rules, e.g.

```
analyzers:
  ros__parameters:
    path: Aggregation
    arms:
      type: diagnostic_aggregator/GenericAnalyzer
      path: Arms
      startswith: [ '/arms' ]
    legs:
      type: diagnostic_aggregator/GenericAnalyzer
      path: Legs
      startswith: [ '/legs' ]
```

rqt_robot_monitor

- **NOT** in the ros/diagnostics repo
- **NO** diagnostics in the name



Upcoming Features
What we are working on

CLI tools for diagnostics

- PR 328
- Verbs: list, echo, csv

More documentation and tutorials

- PR 399
- And more

CONVINCE

Funded by the European Union

LTL (Linear temporal logic)-based monitoring

- CONVINCE Monitoring
- Check system state against desired properties

BOSCH

(My) Diagnostics Philosophy
Best Practices I

- Main purpose: Observe the **current state** of the robot
- Think of it as a control panel where an operator has **all the information** they need
- Try to **limit** the metrics to <10, ideally 2-3 per component
- **Warnings** are states that are unusual but allow continued operation
- **Errors** indicate states that do not allow the robot to operate further and shall be immediately addressed
- Think about a logging and diagnostics concept in your team and **document it**

BOSCH

Comparison to other concepts
Best Practices II

Diagnostics vs

- Logging
  - Logging is (a lot) more verbose
  - Captures the inner state of a SW component
  - Are (usually) for later consumption and analysis
- Bagfiles
  - It is useful to record diagnostics in bagfiles
  - Will also contain (non-critical) state info
- Testing
  - Diagnostics help to find causes for failing test more quickly
  - But don't replace testing
  - Crucial diagnostics mat be tested themselves

Antipatterns
Best Practices III

- In general, diagnostics are not meant to be used functionally
  - The error handling that a robotic system does by itself should not depend on diagnostics
  - Diagnostics should help a human observer or technician to understand a problem that was not recovered from
- The "right" amount of red
  - Diagnostics must be tuned in a way such that they really mean a problem
  - Otherwise, human observers get used to seeing error messages and don't recognize critical ones
  - In a similar theme, warnings should not be too frequent to not become meaningless
- Diagnostics must be received
  - Diagnostics are meant as a communication method from robot to human
  - So, in fully autonomous systems, they must be logged correctly and evaluated retroactively
  - It is also worth to differentiate between roles,
    - for example, if an end user will see and/or understand diagnostics content or
    - if it must only be consumed by trained technicians

What we learned today
Summary

`diagnostic_aggregator`: Aggregates diagnostic messages from different sources into a single message.

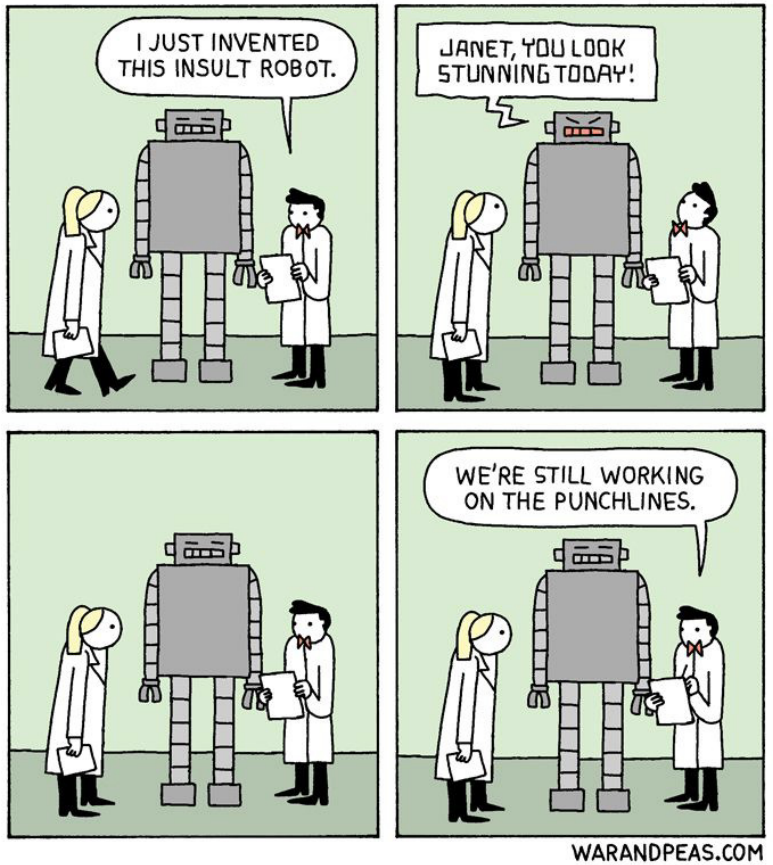
`diagnostic_analysis`: *Not ported to ROS2 yet*
#contributions-welcome

`diagnostic_common_diagnostics`: Predefined nodes for monitoring the Linux and ROS system.

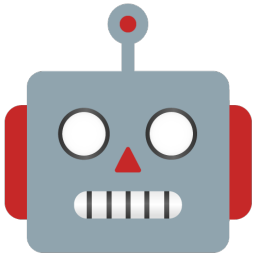
`diagnostic_updater`: Base classes to publishing custom diagnostic messages for Python and C++.

`self_test`: Tools to perform self tests on nodes.

- Diagnostics are a useful tool for robotic systems
- More important than the technical implementation is the content
- Think about a logging and diagnostics concept in your team and **document it**



How was my robot?



Christian Henkel

Bosch Research

ct2034

ros/diagnostics

Any **contributions** are very welcome

- manageable size of codebase
- efficient CI
- quickish feedback