INTRODUCTION TO ECLIPSE ICEORYX™

WRITING A SAFE IPC FRAMEWORK FOR AUTONOMOUS ROBOTS AND CARS

FOSDEM 2020

Christian Eltzschig, Simon Hoinkis
(Robert Bosch GmbH)

iceoryx

BOSCH

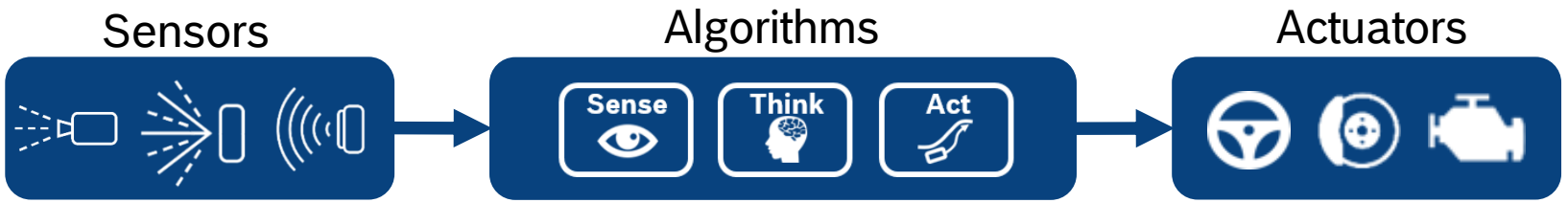
Agenda

1. Eclipse iceoryx
    1. Why does inter-process communication (IPC) matter?
    2. How does a typical middleware work?
    3. What is Eclipse iceoryx?
    4. Safety challenges & lessons learned
    5. Upcoming features

2. Demo of robot Larry

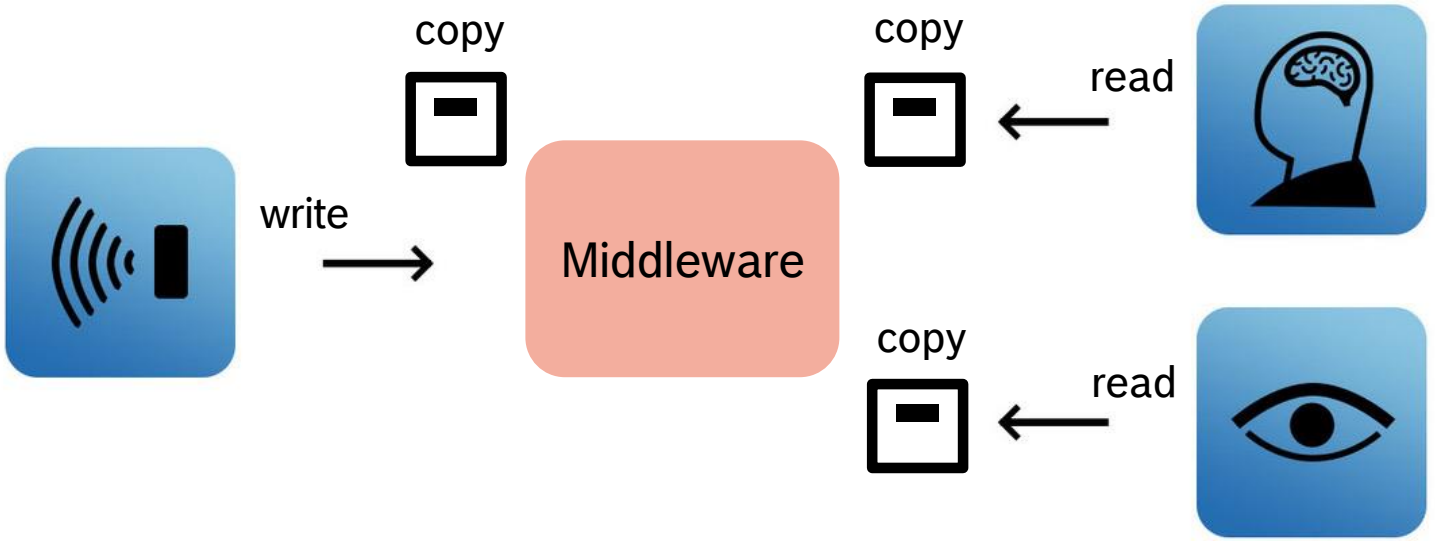
- Spare-time project not associated with Bosch

Introduction to Eclipse iceoryx
Why does IPC matter?

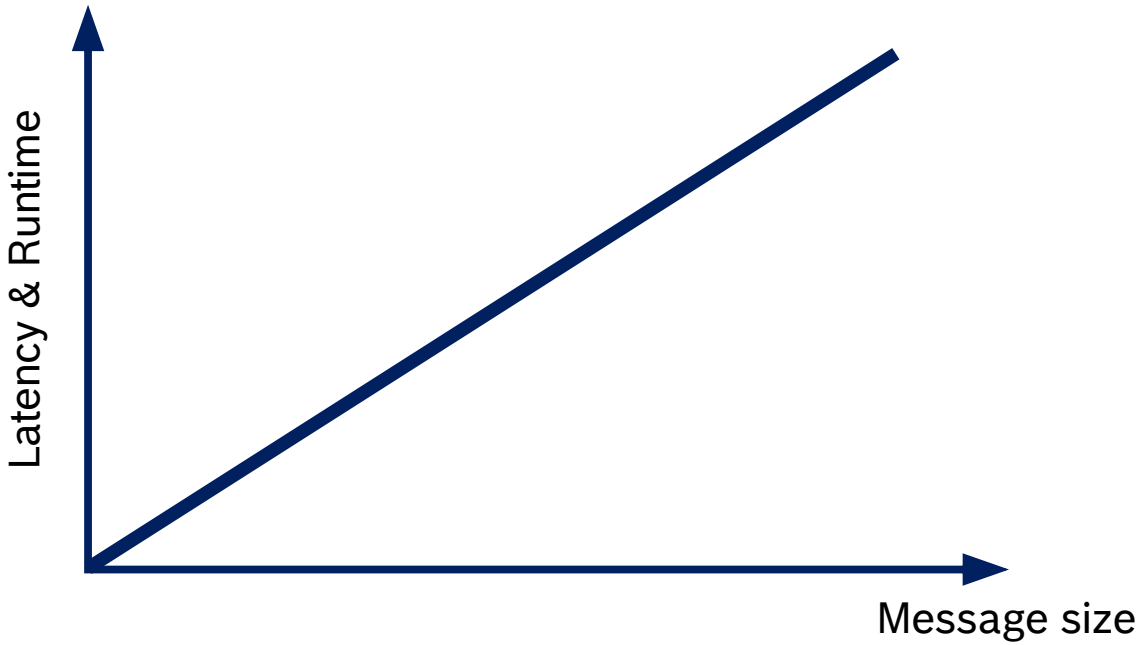


automated driving is a data processing chain with sensor input of up to 10 GB/s

Introduction to Eclipse iceoryx
How does a typical middleware work?

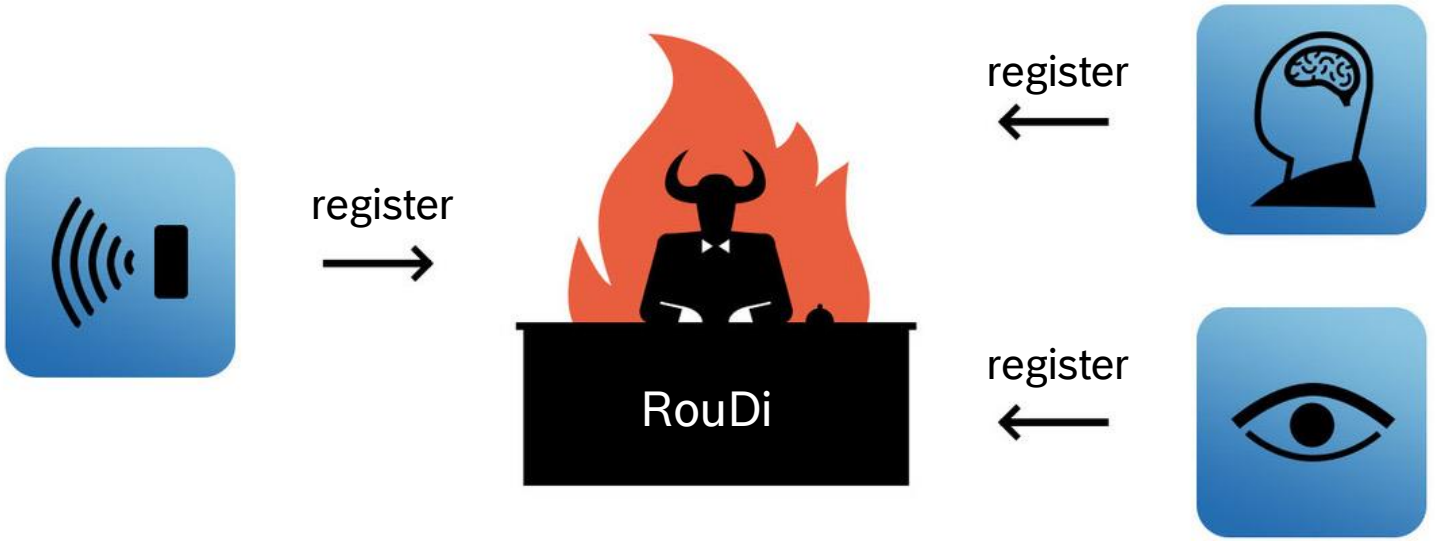


Introduction to Eclipse iceoryx
How does a typical middleware work?

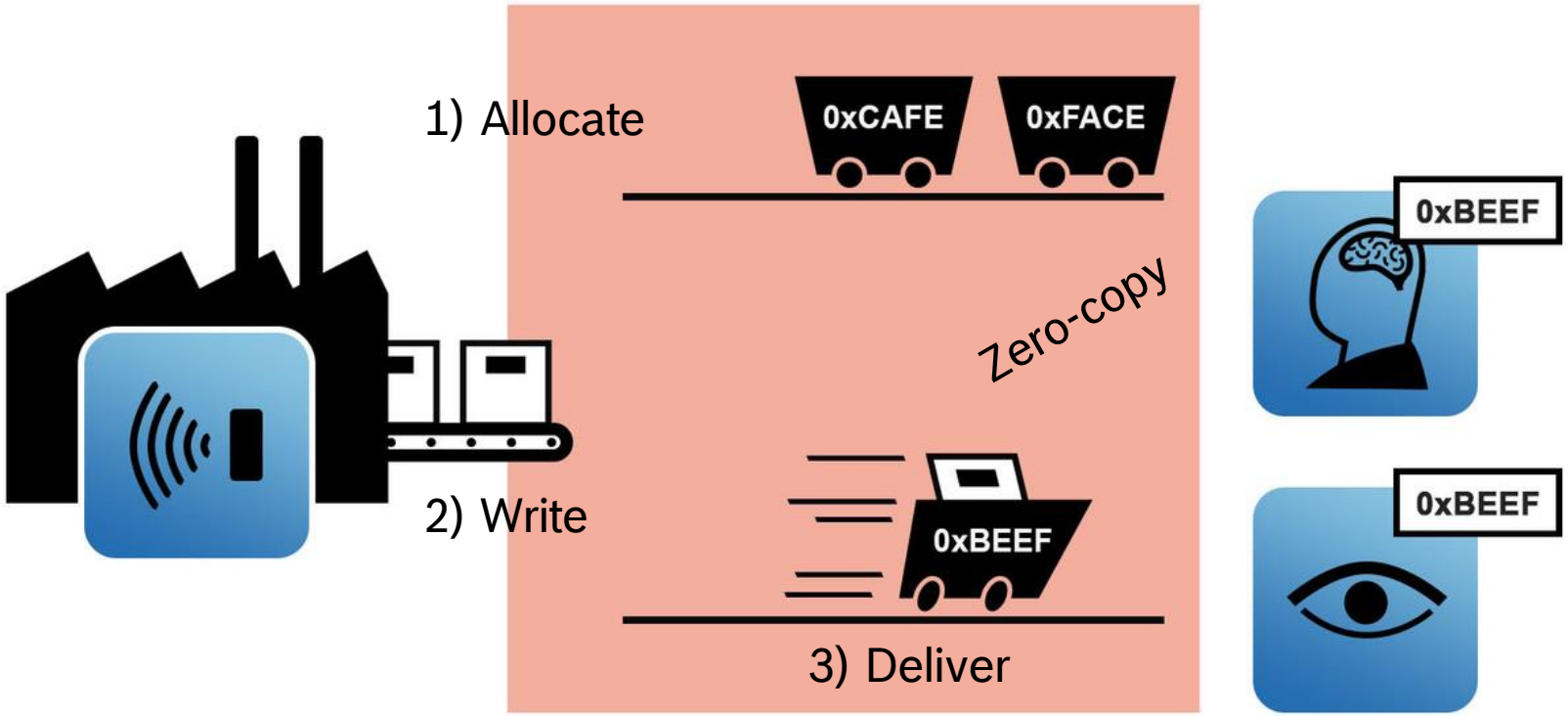


Non-linear scale

Introduction to Eclipse iceoryx
What is Eclipse iceoryx?

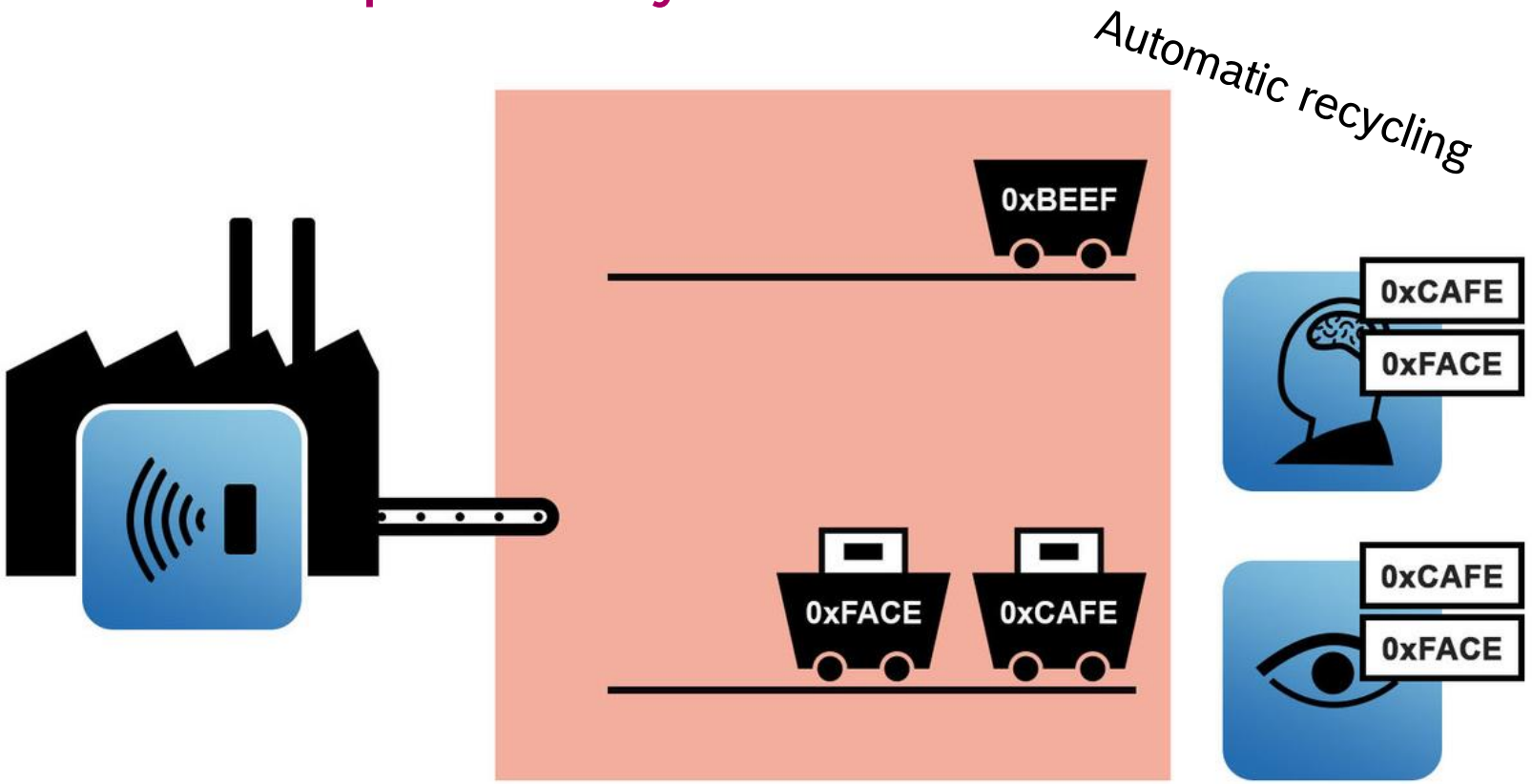


Introduction to Eclipse iceoryx
What is Eclipse iceoryx?



Memory pools in shared memory

Introduction to Eclipse iceoryx
What is Eclipse iceoryx?



Memory pools in shared memory

Introduction to Eclipse iceoryx
What is Eclipse iceoryx?

Introduction to Eclipse iceoryx
What is Eclipse iceoryx?

- Apache 2.0 license

- Written in C++11 (soon C++14)

- Runs on Linux and QNX

- Using state-of-the-art lock-free algorithms

- Bare-metal API best used with higher level API
  - Bindings available for ROS2 and eCal (Continental AG)
  - Can be used as implementation for Adaptive AUTOSAR communication API

Introduction to Eclipse iceoryx
Safety challenges & lessons learned

- We aim for ASIL-D (Automotive Safety Integrity Level) compliance

"Determinism is the key"

- No heap, but static memory pools
- Only subset of C++ standard template library used
- No undefined behavior
- No exceptions, but state-of-the-art error handling concept (`std::expected`)

"Lock-free programming is hard"

- Took us about 2 years to get the Safely-overflowing-FiFo (SoFi) right

"Start with the basic building blocks"

- Many components still not optimized and feature-complete

"Transparency builds trust"

- Develop with the community in the open

Introduction to Eclipse iceoryx
Upcoming features

- n:1 communication
- Request /response communication
- Various language bindings (Rust binding already available!)
- Integration with Eclipse Cyclone DDS
- Windows and macOS support

macOS

Write us:

iceoryx-dev@eclipse.org

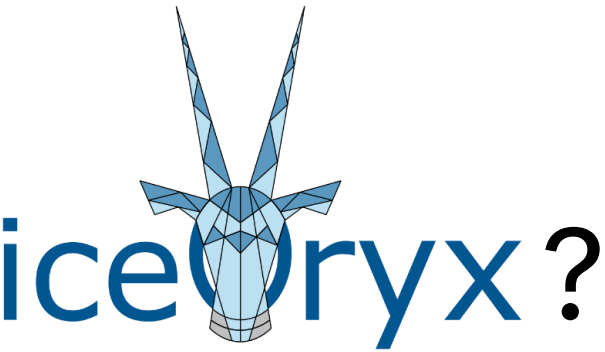
Join us:

https://github.com/eclipse/iceoryx

Demo of robot Larry

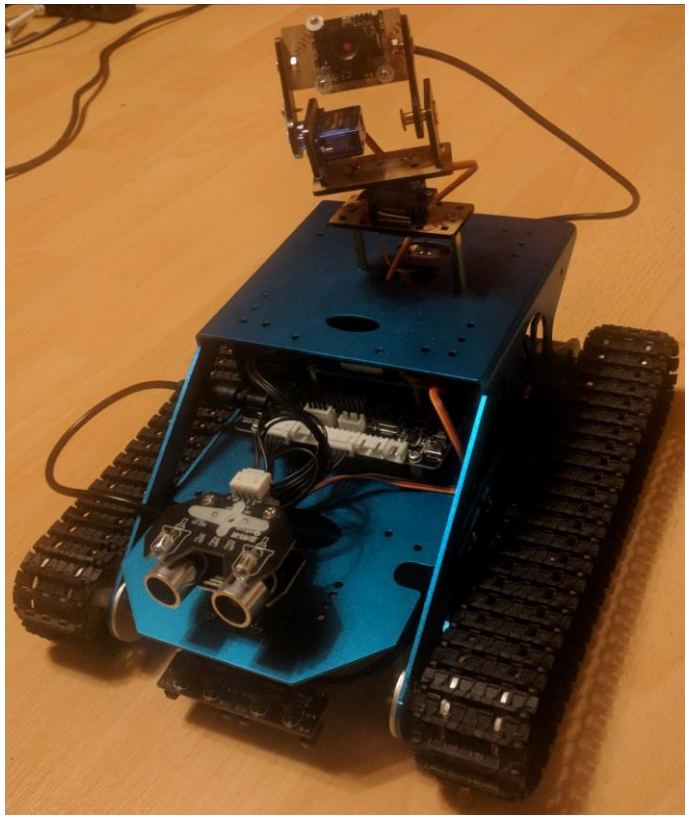
The one thing that you should take with you!

The Question:

Is it powered by 

?

Demo of robot Larry
Who is Larry?



- A robot with a variety of sensors
  - Camera
  - UltraSonic
  - Tracking Sensor
    (useful for Robo-Races)
  - Tracking Camera (localization)
  - In Progress: Stereo Camera
  - Planned: Microphone Array
  - Planned: Speaker
- Open Source Demonstrator for Ice0ryx
  - License: Apache 2.0

Demo of robot Larry

Can I build my own Larry?

Demo of robot Larry

Can I build my own Larry?

Yes you can!

The repository where everything comes together:

https://gitlab.com/larry.robotics/larry.robotics

The apps which run on Larry:

https://gitlab.com/el.chris/larry-services

User Interface for remote control:

https://gitlab.com/el.chris/larry-ui

3D Engine which provides some building blocks:

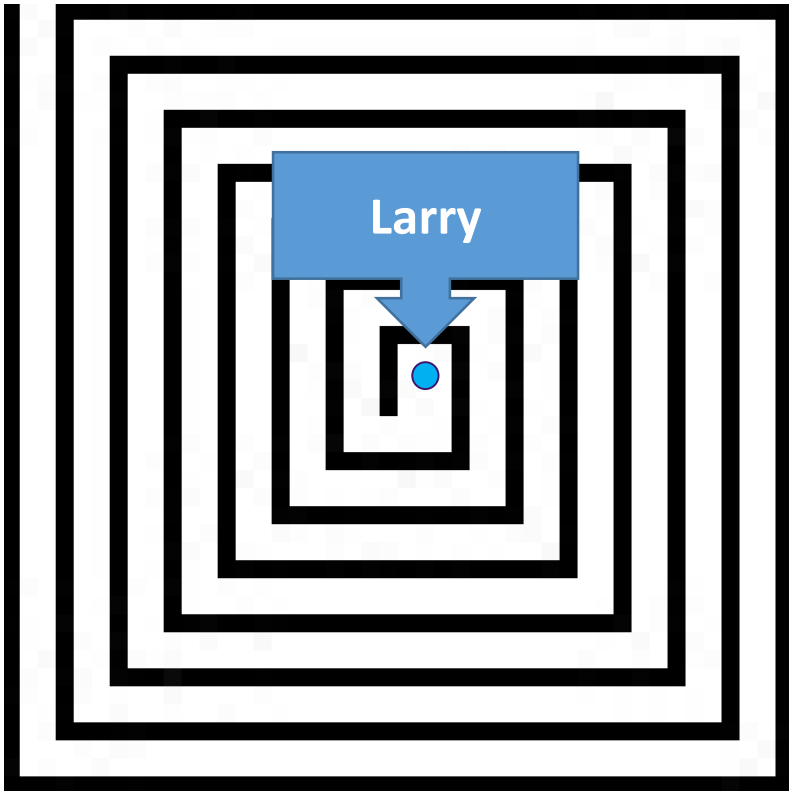
https://gitlab.com/el.chris/3delch

Demo of robot Larry

Let‘s help Larry to get out of a labyrinth

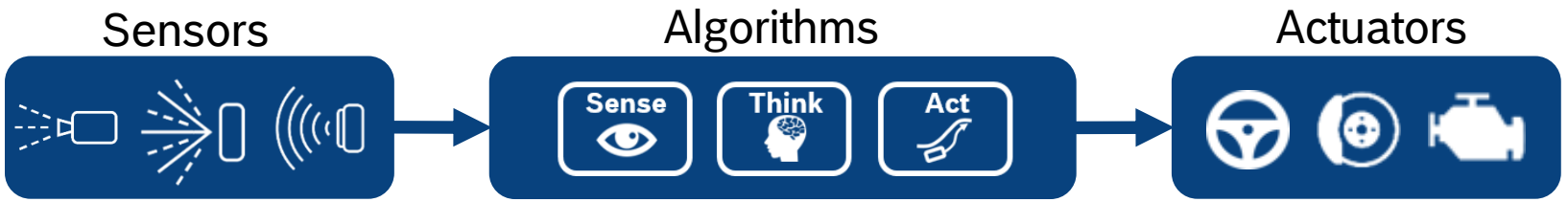
Two rules to get out of this labyrinth:

- **Rule 1**: no obstacle
  -> move forward
- **Rule 2**: obstacle ahead
  -> turn right



Demo of robot Larry

Let‘s help Larry to get out of a labyrinth



Ultrasonic sensor

detects obstacles

Two rules to get out of a labyrinth

Driver

Turns right, moves forward

Demo of robot Larry
Let‘s help Larry to get out of a labyrinth

```
runtime::PoshRuntime::getInstance( "/Explorer" );
auto ultraSonicSensorData =
    TypedSubscriber< larryServices::ultraSonicSensor_t >(
    capro::ServiceDescription( "larry", "UltraSonicSensor", "intern" ) );

larryServices::Driver driver;

Length minimumDistance = Length::Meter( 0.3 );

while ( true ) {
  auto sensorData = ultraSonicSensorData.tryReceive();
  if ( sensorData && sensorData->distance < minimumDistance.GetMeter() )
    driver.TurnRight();
  else if ( !sensorData || ( sensorData &&
              sensorData->distance >= minimumDistance.GetMeter() ) )
    driver.Forward();
}
```

Demo of robot Larry
Let‘s help Larry to get out of a labyrinth

1. Register at RouDi as Explorer application

```
runtime::PoshRuntime::getInstance( "/Explorer" );
```

2. Subscribe to the ultra sonic sensor

```
auto ultraSonicSensorData =
    TypedSubscriber< larryServices::ultraSonicSensor_t >(
    capro::ServiceDescription( "larry", "UltraSonicSensor",
                               "intern" ) );
```

Demo of robot Larry
Let‘s help Larry to get out of a labyrinth

3. Create driver class which controls the movement of Larry

```
Driver driver;
```

4. Define how close we want to get to an obstacle

```
Length minimumDistance = Length::Meter( 0.3 );
```

Demo of robot Larry
Let‘s help Larry to get out of a labyrinth

5. Enter the event loop

```
while ( true ) {
```

6. Receive the distance to the obstacle in front of us

```
auto sensorData = ultraSonicSensorData.tryReceive();
```

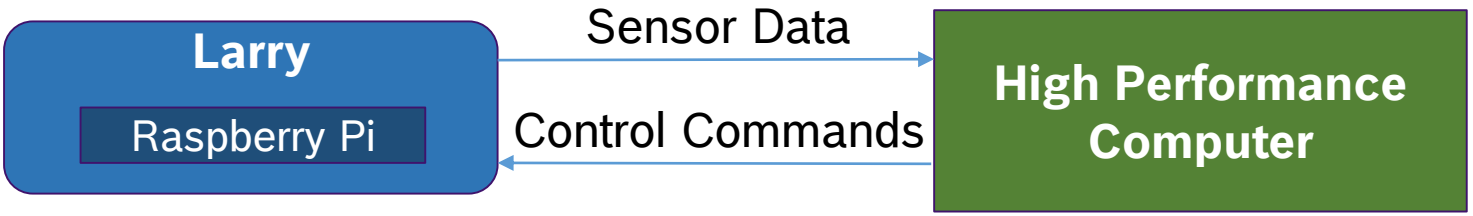
7. Obstacle detected: turn right

```
if ( sensorData &&
     sensorData->distance < minimumDistance.GetMeter() )
  driver.TurnRight();
```

8. No obstacle detected: go forward

```
else if ( !sensorData || ( sensorData &&
   sensorData->distance >= minimumDistance.GetMeter() ) )
  driver.Forward();
```

Demo of robot Larry
Larry‘s architecture



- Collects sensor data
- Basic sensor preprocessing
- Low level control
  - Obstacle detection
  - Emergency break
- High-level control

Demo of robot Larry

Ideas we could realize together!

Write us:
me@elchris.org

- Microphone array and speech recognition
- Speaker and voice output
- Virtual Larry Environment to train neuronal networks to handle Larry
- Stereo camera and 3D environment rebuilding
- Object detection
- Running multiple Larry's in a swarm
- Virtual Reality support to control Larry
- Win a robo race

Introduction to Eclipse iceoryx
Will iceoryx become a replacement for ROS2?

Introduction to Eclipse iceoryx
Will iceoryx become a replacement for ROS2?

No!

Support the community to make ROS2 even faster

ROS2 binding available here:

https://github.com/ros2/rmw_iceoryx

Introduction to Eclipse iceoryx
Will iceoryx become a replacement for ROS2?

If your ROS2 project has a performance bottleneck

consider switching to 

QUESTIONS?

BOSCH

Introduction to Eclipse iceoryx
References

- https://projects.eclipse.org/projects/technology.iceoryx
- https://github.com/eclipse/iceoryx
- https://github.com/ros2/rmw_iceoryx
- https://github.com/eclipse/iceoryx/wiki/Eclipse-iceoryx%E2%84%A2-in-1000-words

- https://github.com/elBoberido/iceoryx-rs

- https://openadx.eclipse.org/

Introduction to Eclipse iceoryx
Backup

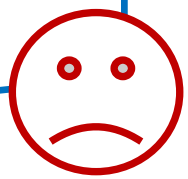
- Default memory pool configuration
  - https://github.com/eclipse/iceoryx/blob/master/iceoryx_posh/source/mepoo/mepoo_config.cpp#L42

- Shared memory limitations
  - Only fixed size messages are supported
    - E.g. no `std::vector` due to heap
    - Message size can be changed with every request of memory
  - No virtual members
  - Messages must have the same memory layout for the publishers and subscribers, no serialization, same compiler with same compiler flags

Introduction to Eclipse iceoryx
Backup

```
dynamic_size_message.msg

int32 one_int
float64 one_float
char[] char_array
```



[4 byte | 8 byte | 24 byte]
dynamic size
(heap allocation)

```
fixed_size_message.msg

int32 one_int
float64 one_float
char[100] char_array
```

[4 byte | 8 byte | 100 byte]
fixed sized (POD)

The message is not allowed to use heap-based data structures (e.g. STL containers with default allocators)