An example of distributed computing in the autonomous driving

About Easymile

Autonomous solutions ready today

EasyMile provides **software** and **complete solutions** for driverless mobility and goods transportation.

We partner with blue-chip manufacturers to **automate** their vehicles. Our technology is built on **safety-by-design** and **ready for deployment today** with clear client benefits.

EasyMile at a glance

Since
2014

30+ PhDs

240+

ISO 9001
certified

5
locations

22
nationalities

Recognition

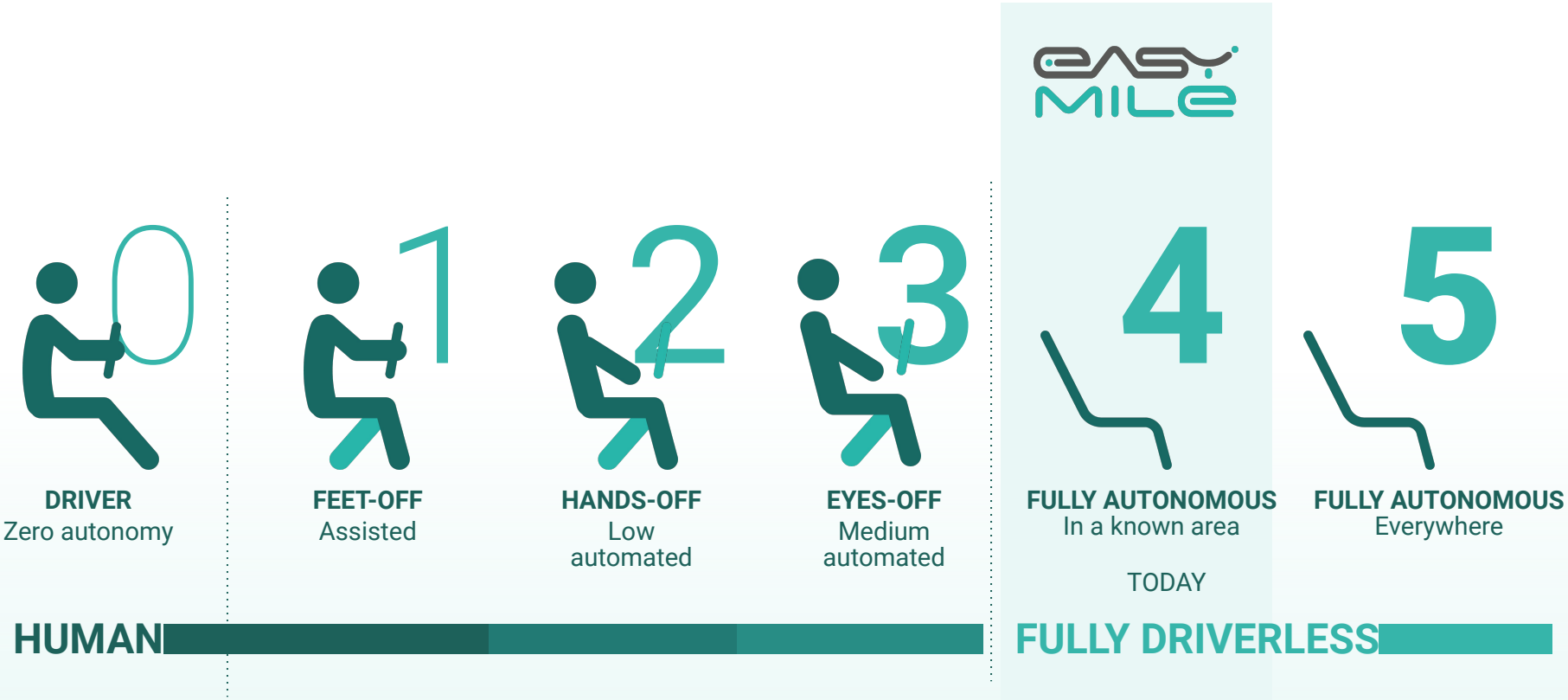
IFOY AWARD
international intralogistics and forklift truck of the year

La FRENCH TECH FT120

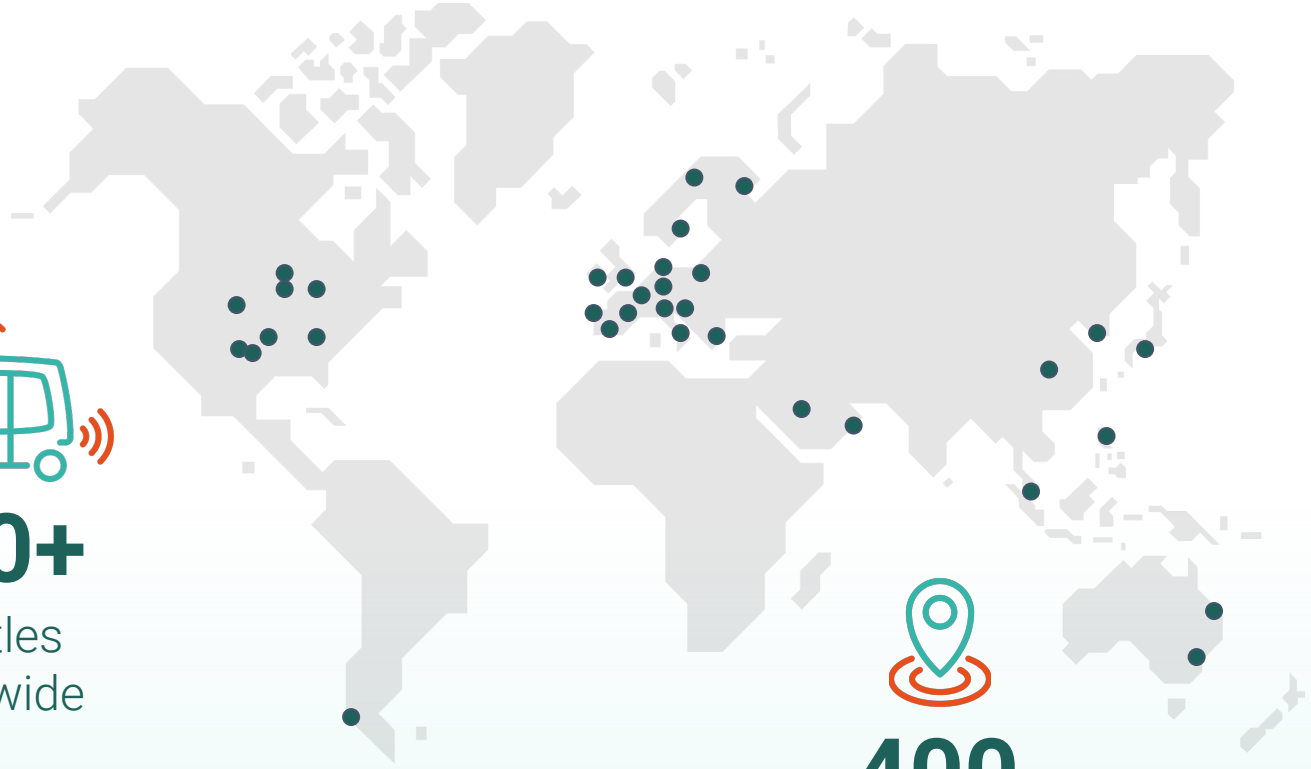
Shareholders and partners

ALSTOM
SEARCHLIGHT
bpifrance
Continental
McWin
NextStage Capital Entrepreneur

Focused on Level 4 of autonomous driving



Most deployed driverless shuttle in the world

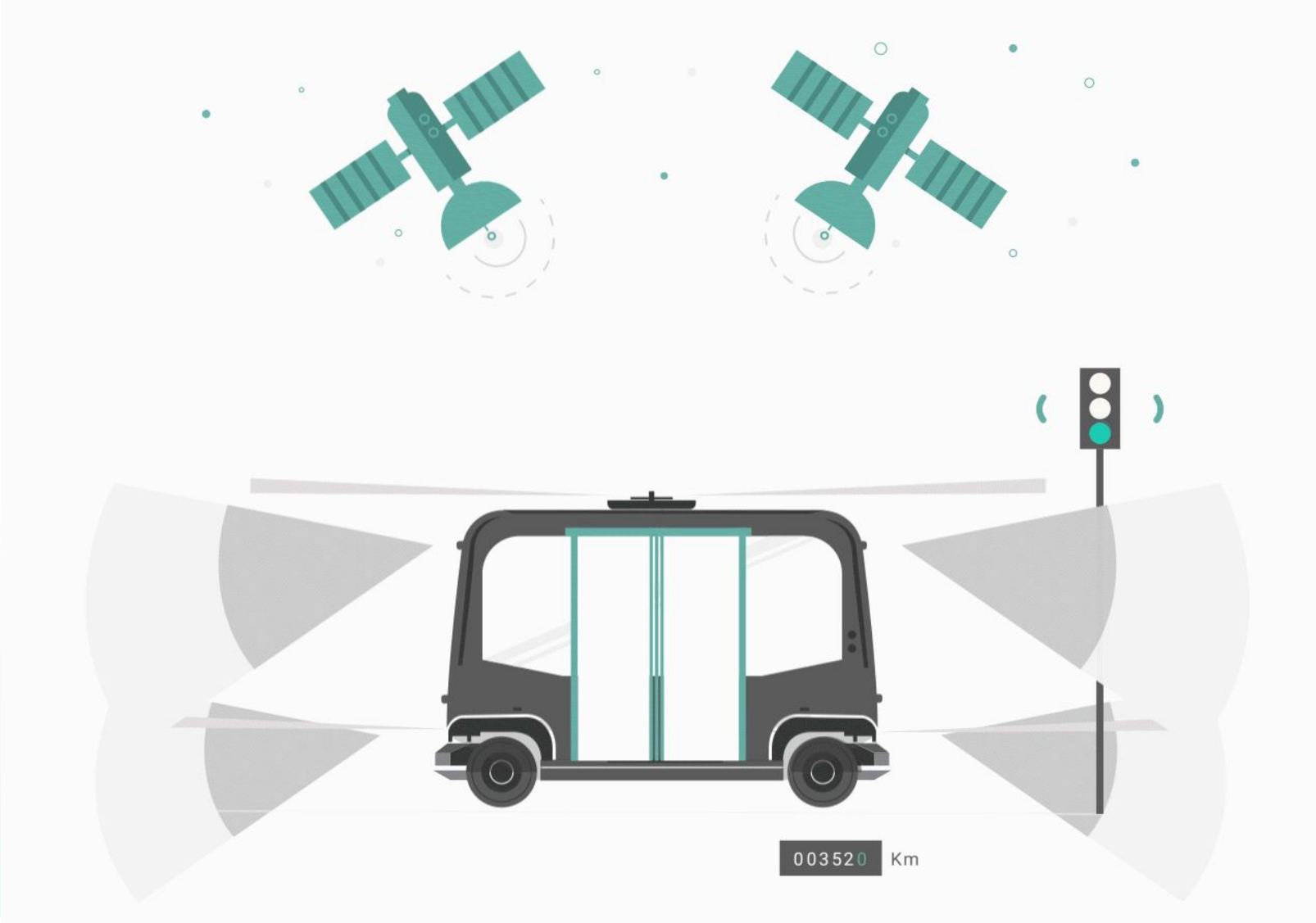


180+
Shuttles
worldwide

400
Deployments
In 30+ countries

> 1,000,000km
Autonomous driving

EZ10 Shuttle overview



Distributed computing

Broad definition

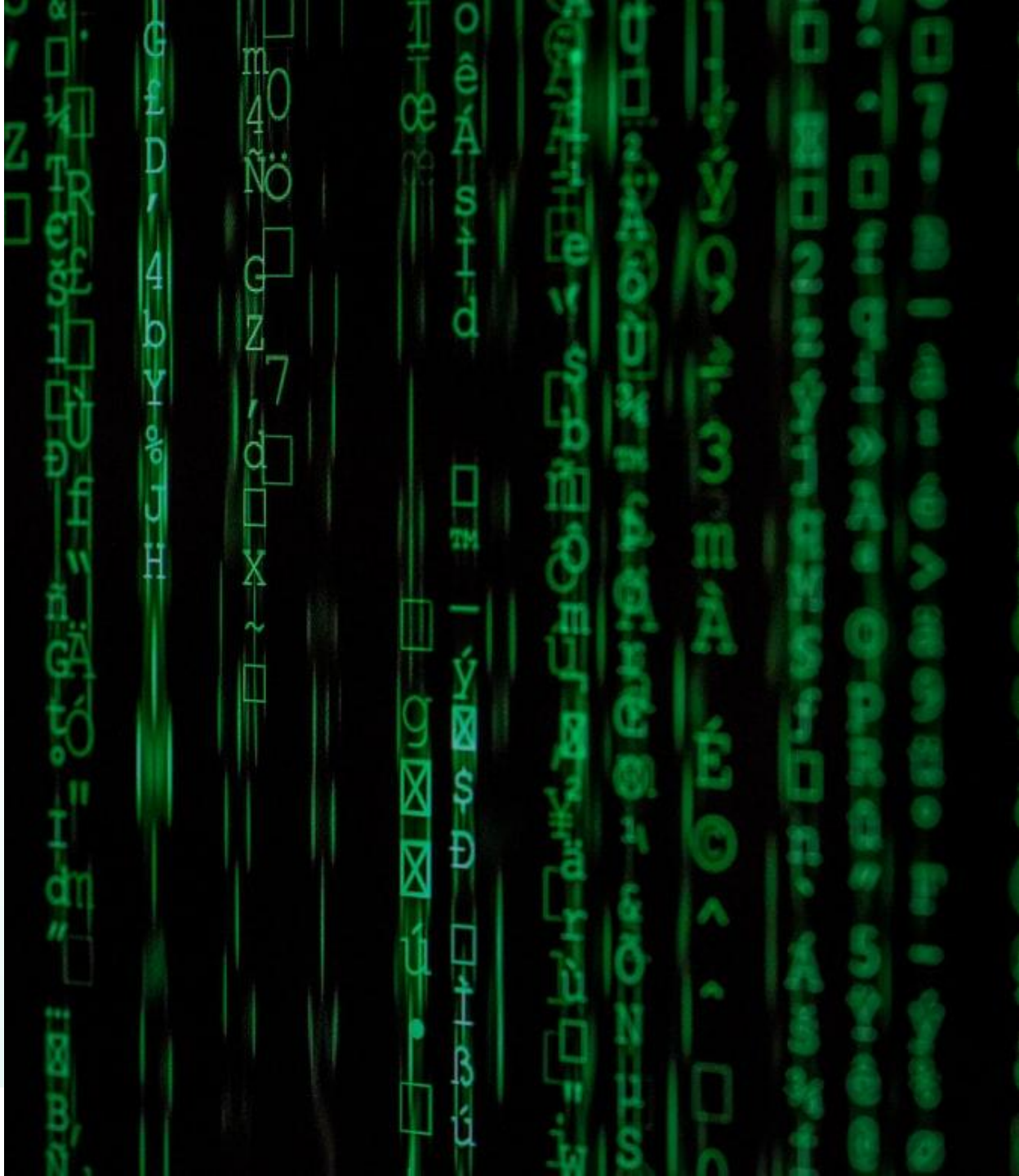
(from wikipedia)
“A ***distributed system*** is a system whose components **are located on different [processing units]**, which communicate and coordinate their actions in order **to achieve a common goal**.”

“While there is no single definition of a distributed system, the following defining properties are commonly used as:

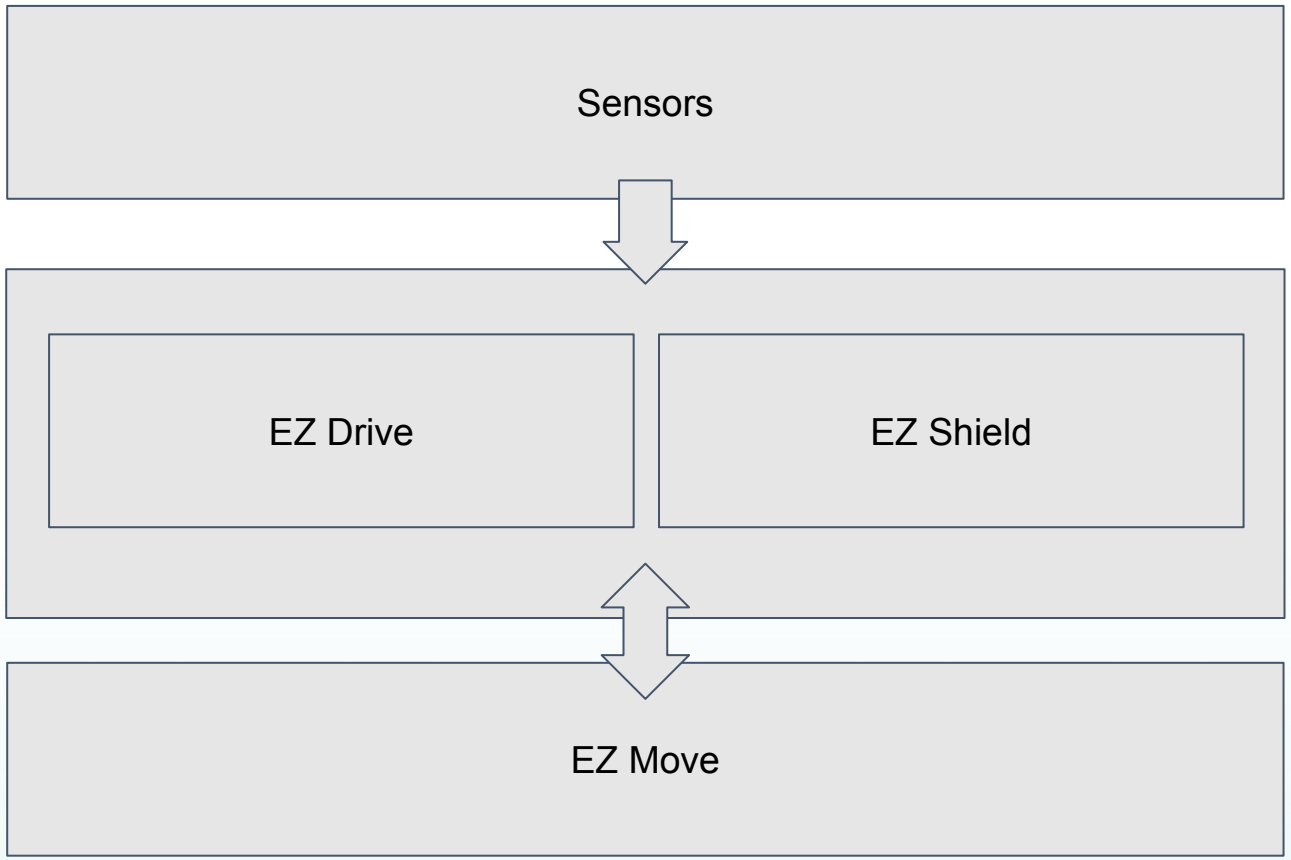
- There are several autonomous computational entities (computers or nodes), each of which has its own local memory.
- The entities communicate with each other by message passing.”

We usually have different strategies for distribution:

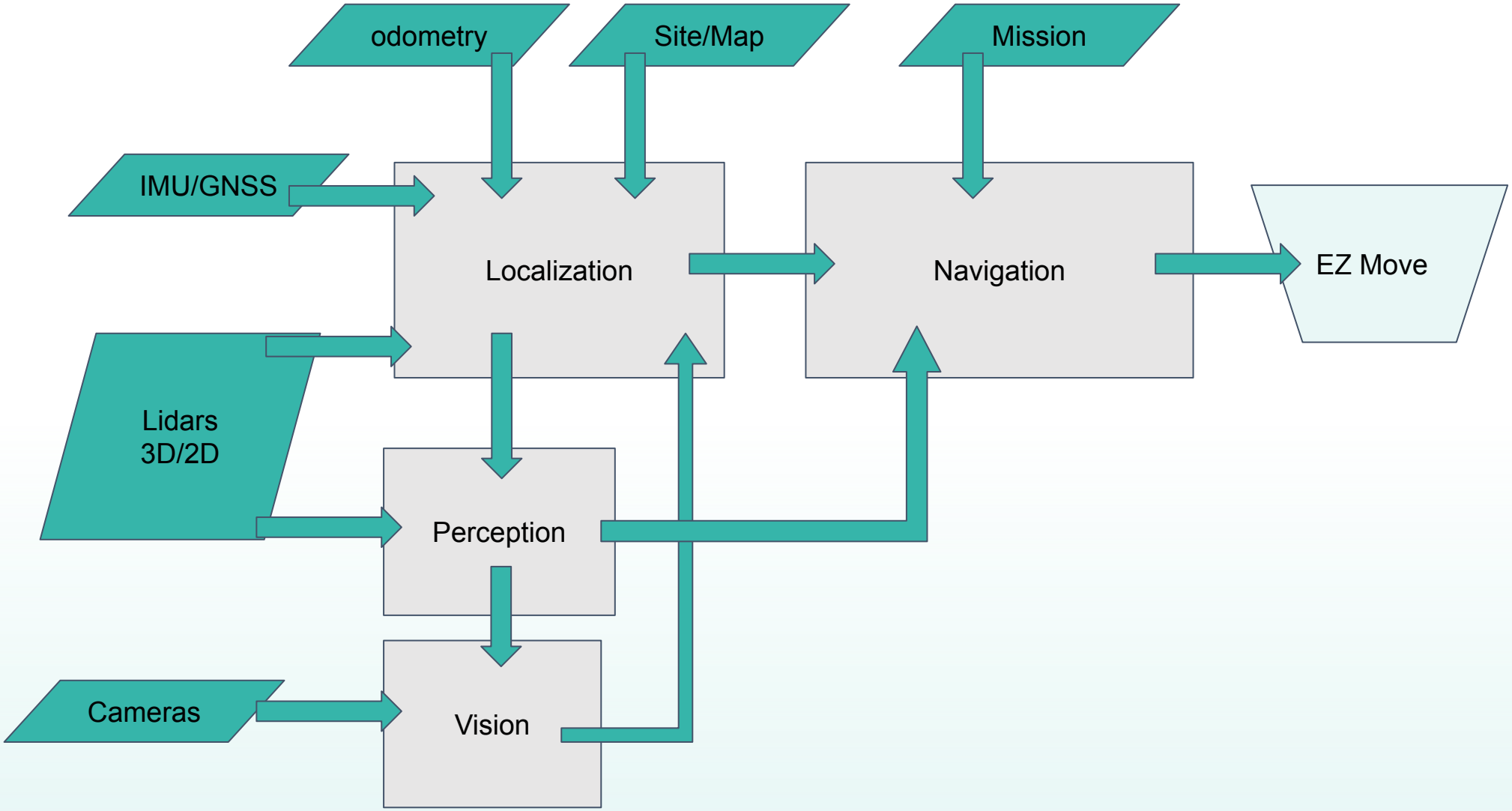
- Parallel algorithms
- Distributed algorithms
- Centralized algorithms



coarse grain view of the on-board system



EZ Drive Components interactions



EZDrive task handling

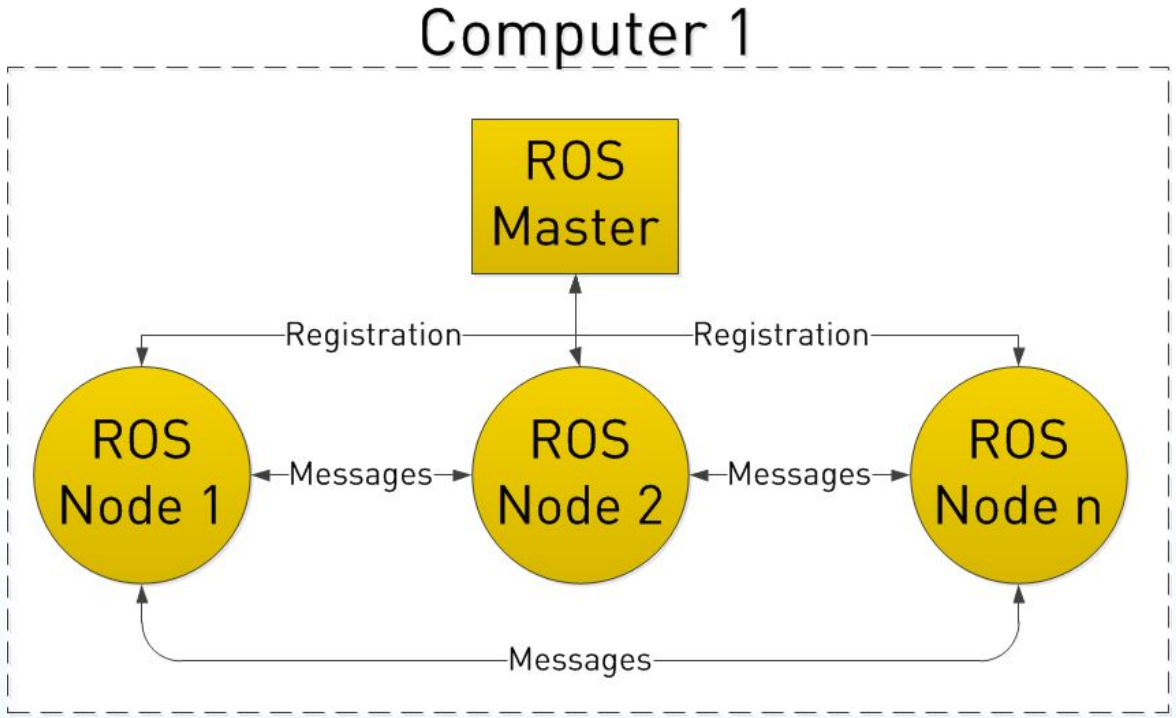
—

EZDrive uses a ROS-like framework.

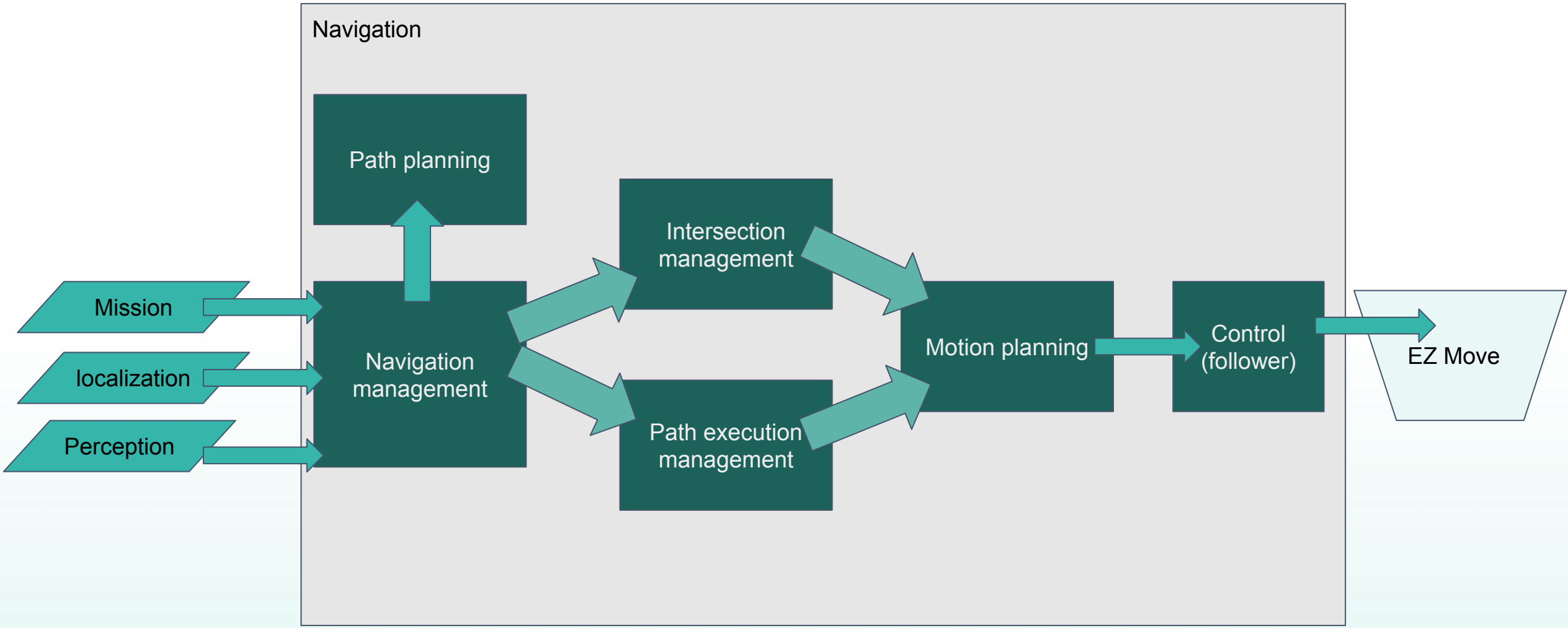
A ROS system is comprised of a number of **nodes**, each of which communicates with the other nodes using a publish/subscribe **messaging** model.

These messages could be consumed by any number of other nodes, on the **local or** any **remote** computers.

ROS is providing a turn key solution for the message massing framework and is widely used by the robot industry.



Navigation detailed view



EZ Drive robot system

The whole system can be described as a graph of sub-computations leading to driving commands sent to the platform (EZ Move).

As each node is a dedicated process, the Linux kernel's scheduler, computation intensive tasks are done on GPU (the CPU is mainly managing IOs and middleware)

Some crashes may be recovered by restarting a node, some may require restarting of many of them (if not all of them).

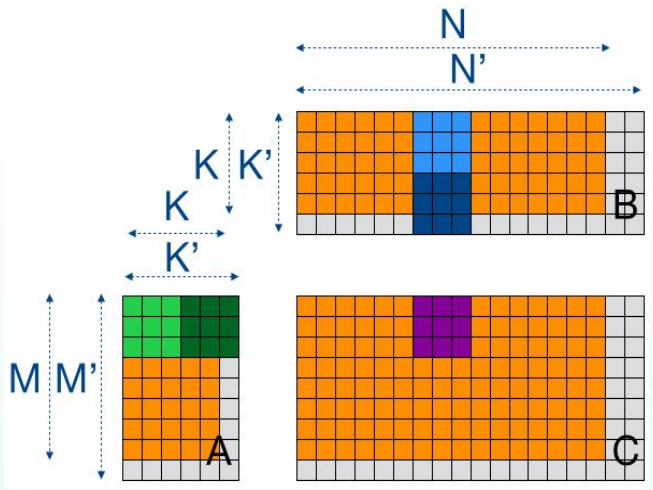
There is no distributed consensus, each node being responsible for its results, each input is terminal (one node does not need to wait for another to receive a message).

EZDrive computation offloading

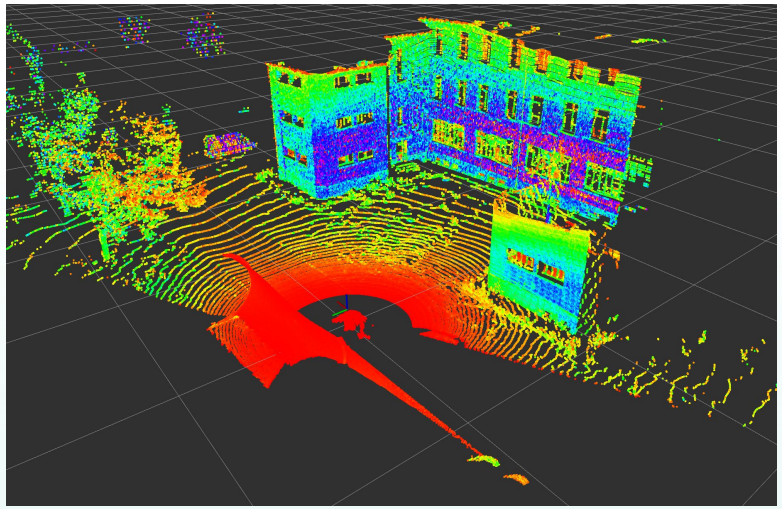
Some CPU intensive tasks linked to computer vision and machine learning are run on GPU (nvidia).

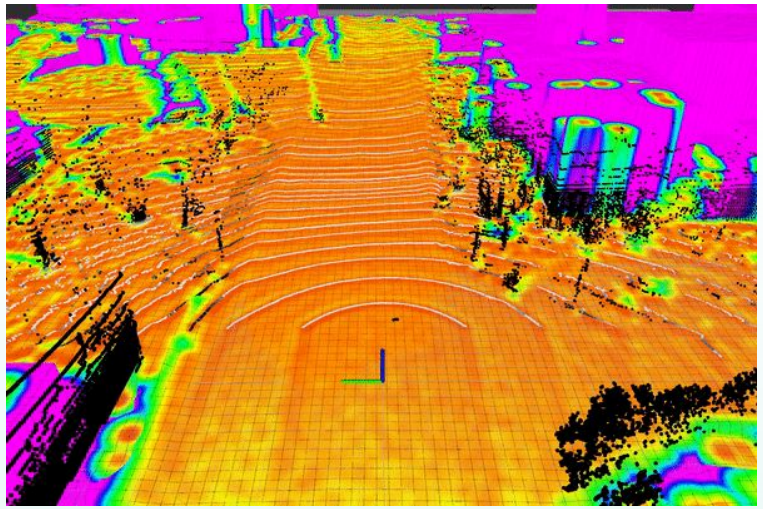
We use CUDA framework to describe and run work unit on the GPU to maximize parallelization of computation.

The overall treatment (say neural network) is sequential, but work is parallelized (parallelized algorithm)



matrix multiplications





computer vision

EZShield supervision

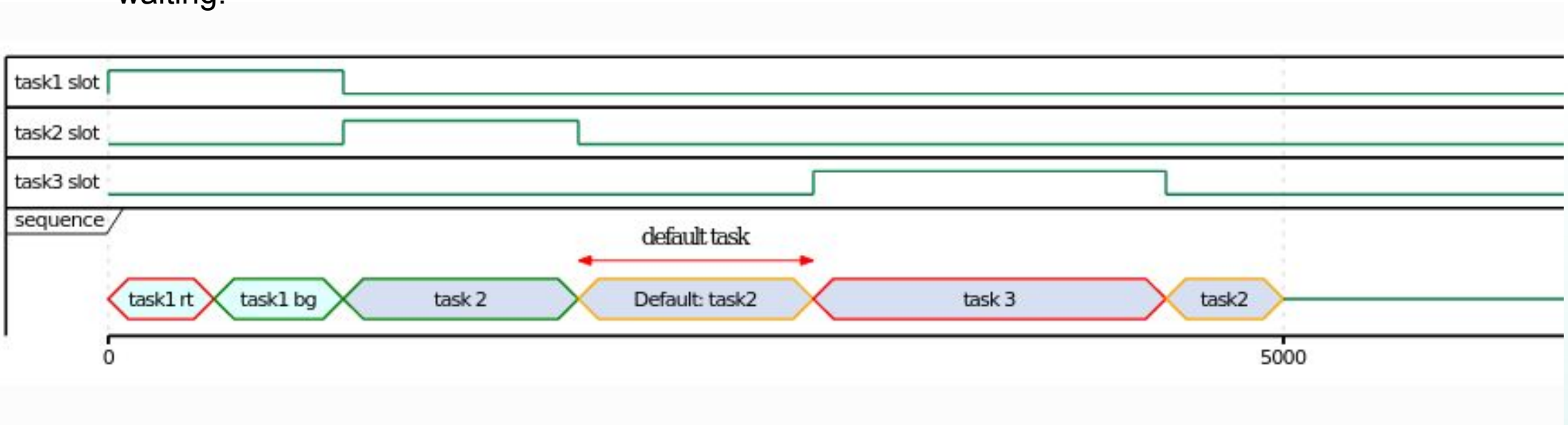
EZshield is responsible for the system supervision (domain of operation):

It performs its own verifications on the system status; any inconsistent or diverging situation lead to an emergency stop.

EZShield Sequencer

EZShield uses a custom sequencer (very basic scheduling) for running real time tasks:

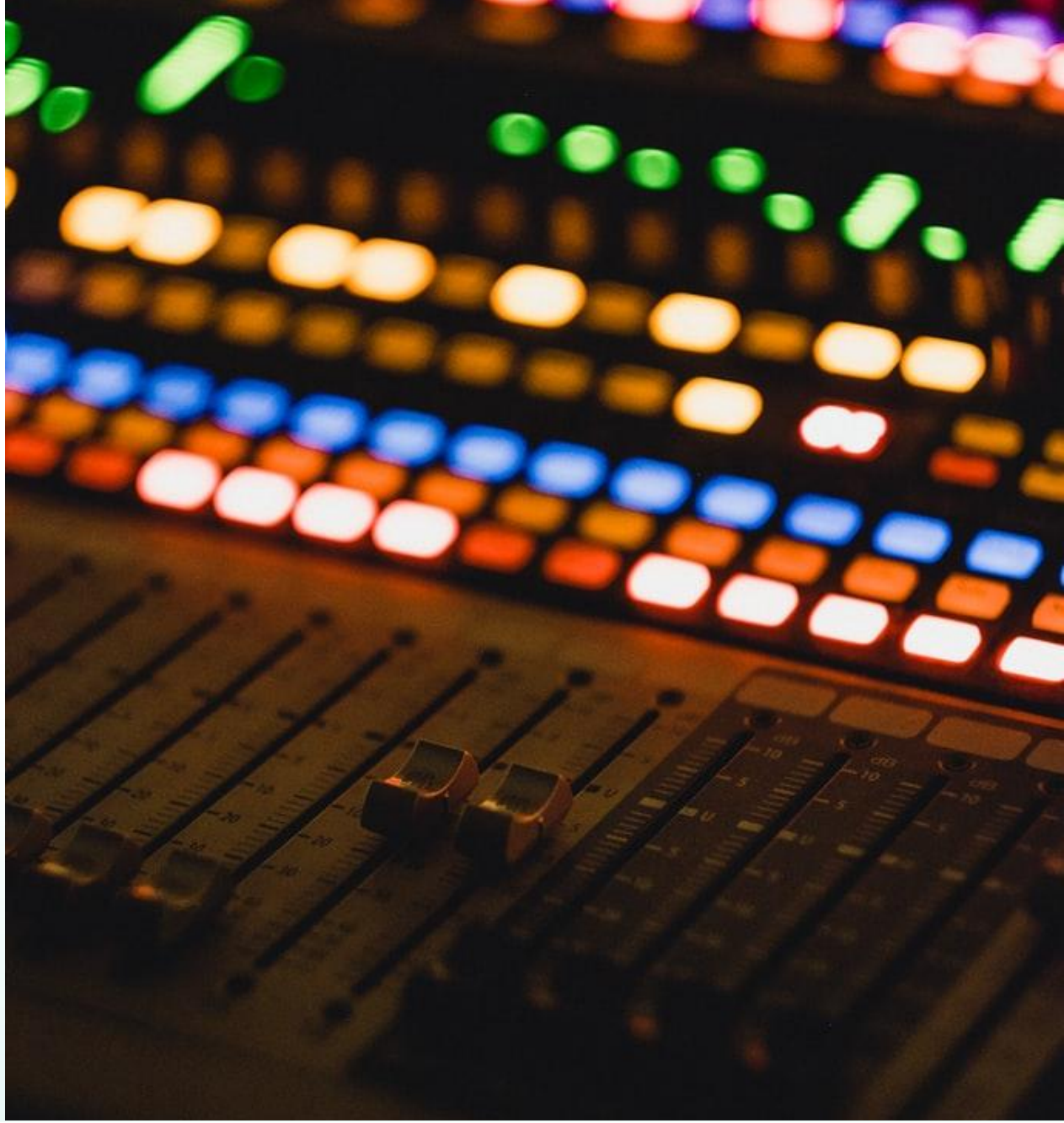
- each tasks has a reserved timeslice
- task may have a “background” non critical part to be done
- when a task does not consume all its timeslice, the remaining part is used to run its background task (or other background tasks).
- A default task may be designated to be run when no remaining background tasks are waiting.



EZShield scheduling objectives

The sequencer main objectives are:

- **Isolate** algorithmic tasks from each other (dealing with a set of tasks whose maturity is not homogeneous) to protect critical tasks (like anti collision)
- **customisable**: make it easy to add and remove task on the product without needing reworking the whole scheduling strategy.
- **Mode ready**: Prepare future use cases where we handles several **execution modes**, whose sequence is different.



EZShield and overtime tasks

Task are designed and tested to run in the allocated time slice; they are expected to yield **before** the end of their slots.

In the unlikely situation where it tries to go overtime, we explored several strategies:

- Panic immediately. Simple, (rude)
- Use a Fault Manager to handle faulty tasks recovery (more complex),
- Log the deadline miss (current behaviour), save and restore the context in the next slot assigned to this task. (Concretely, delaying the execution by one cycle will not affect algorithm's behavior. A limit on the number of deadline misses is set to three misses in less than 100 milliseconds.)

Future

Distributed fleets

—

Fleet of vehicle may be seen as a distributed system communicating to establish a better perception of the environment (say obstacles on the road, traffic jam, too many people to cary etc…) communicating thru V2X, for example.

Collaborative HD cartography, deep learning, and *“who knows?”*

Questions?

Thank you for attending!

any feedback is welcome

sebastien.gonzalve .at. easymile.com sylvain.assemat .at. easymile.com

Thank you

Connect with us

#AutonomousFuture

Backup slides

Growing vehicle portfolio

Goods

Available now

TLD

KAMAG

TERBERG SPECIAL VEHICLES

EASY MILE

People Mover

Available now

LIGIER GROUP

STELLANTIS

ALSTOM


MESSE 2020
TLD
EASY MILE
Tract Easy