COGNIPILOT

For the Next Generation
of Assured Autonomy

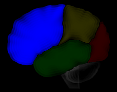
Meet CogniPilot

CogniPilot is a nonprofit, open-source organization focused on development of the future of robust autonomy and AI.

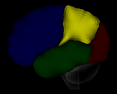
- Trusted by innovative organizations committed to transparent, secure, and scalable solutions in complex mission environments.

What We Do

CogniPilot offers a complete, integrated suite of AI-driven tools for autonomous navigation, decision-making, and mission planning.

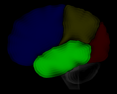


Corti – AI/ML based behavior, path planning, dynamic mission adaptation, complex tasks.



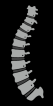
Pari – AI/ML for payloads, sensor fusion, spatial awareness.

Oculi – Vision systems, collision avoidance, visual inertial odometry.



Tempi – Communication, coordination, swarm control, cloud connect.

Cerebri – Firmware, controls, fail-safes and manual control systems.



Spinali – Open hardware leveraging industrial and automotive networking.

Why Choose CogniPilot?

- **Open-source and nonprofit-driven:** No corporate influence, fully secure, and open for collaboration.
- **Flexible and scalable:** Adapt, build, and deploy across a wide range of autonomous systems.
- **Comprehensive toolset:** Integrates vision, sensor fusion, swarm control, AI and more for seamless autonomous performance.
- **Future-proof:** Continuously evolving with contributions from a global community of developers and researchers.
- **Deep Hardware Integration:** CogniPilot works directly with the semiconductor industry to create the next generation of heterogeneous compute.

Why CogniPilot: *Open-Source* **not** **Bloat-Ware**

Bloat-Ware

Vehicle Specific Firmware

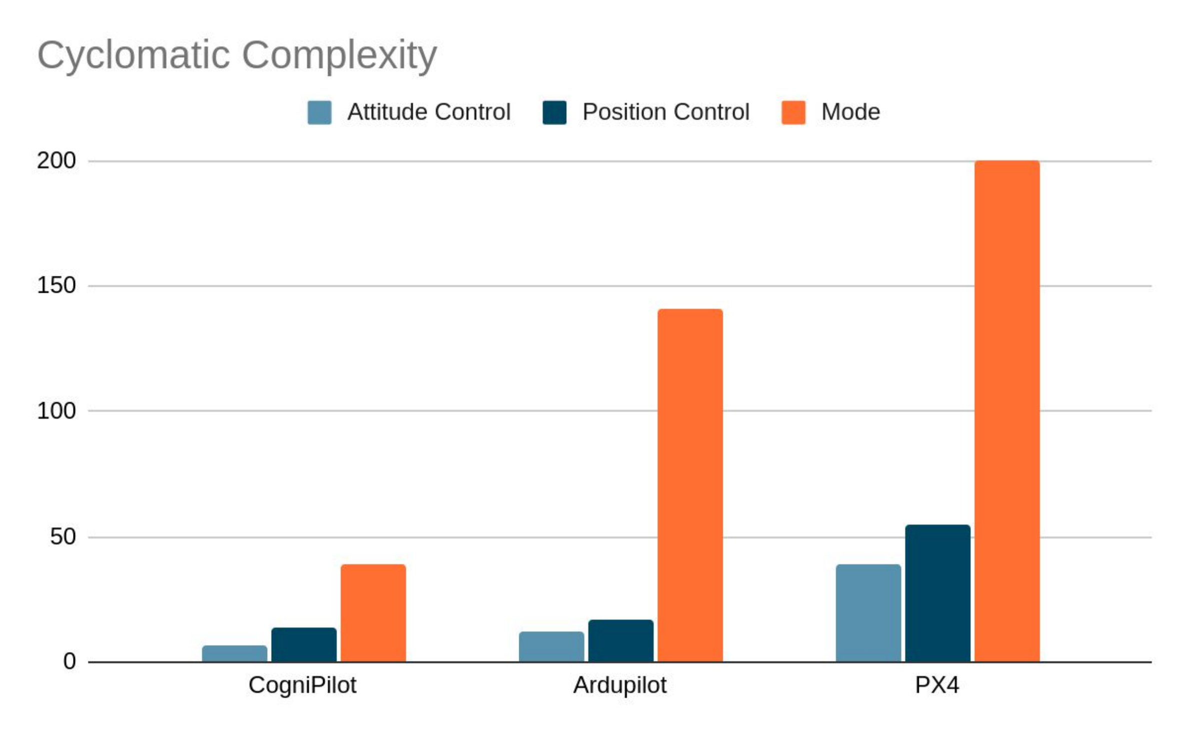
Common Reusable Drivers/ Libraries

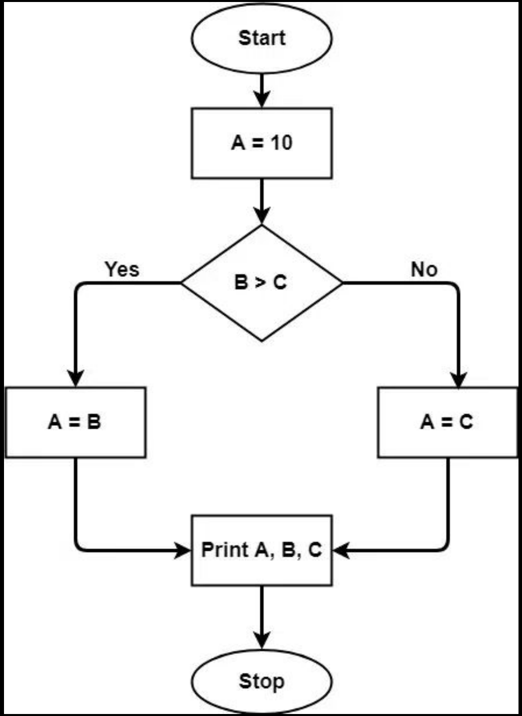
- Leveraging Zephyr RTOS and Rust

Vehicle Specific Firmware

- Enables application of formal methods and reduces cyclomatic complexity
  - Generalizing to Boat/Plane/etc leads to increased complexity
- Control synthesis in Modelica/Python/Matlab simplifies vehicle specific creation.

Cyclomatic Complexity Comparison





Why CogniPilot: *Expertise Separation*

Mechanical & **Software** Engineer

Mechanical Engineer

Software Engineer

Expertise Separation

- Larger developer pool
- Easier to train

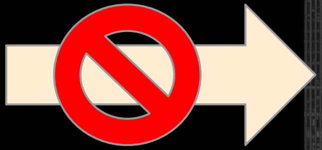
How do we enable this?

- Established APIs
- Formal methods

Why CogniPilot: **Safety is NOT an Afterthought**

Aerial Photography

Drone Racing



Safety Critical Systems
(e.g. Urban Air Mobility)

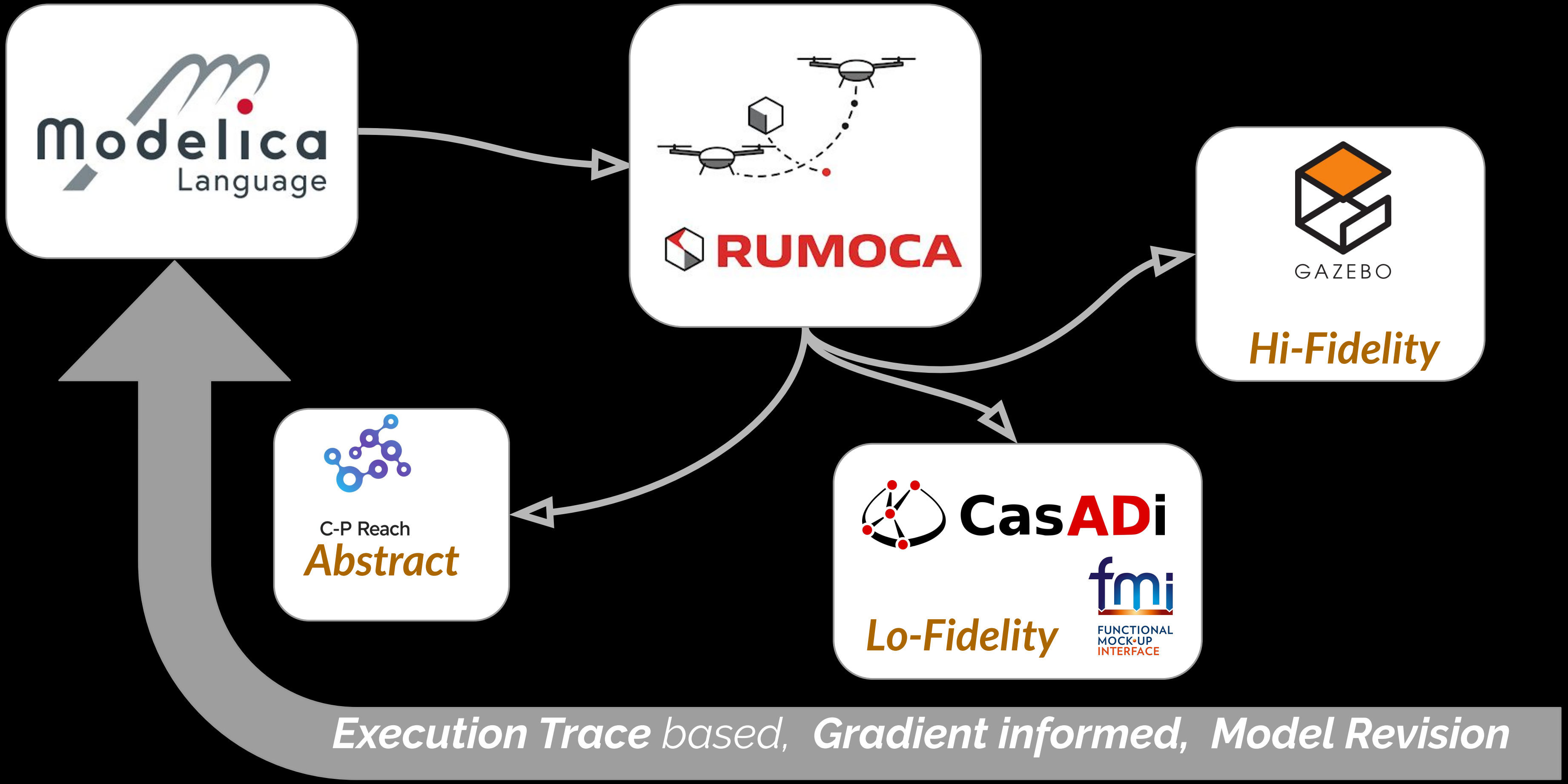
Safety Critical

- When human-life is at stake, safety must be a key design driver

How do we enable this?

- Correct-by-construction software
- Safety critical drivers and hardware capable of SIL ratings

Rumoca: Cyber Physical Model Translator



The Need for Cohesive System Descriptions

ROS based common formats (and their current limitations):

- **URDF**
  - URDF was designed to describe the properties of a single robot, not a complete robotic system or environment.
  - **No Native Sensor Models:** There are no standard tags for sensors like cameras, IMUs, or LiDAR. While you can create a link to represent the sensor's physical placement, you cannot describe its properties (e.g., field of view, noise model, update rate) within the URDF standard.
  - **Reliance on Extensions:** Simulators like Gazebo get around this by introducing their own custom tags (e.g., <gazebo>, <sensor>) inside the URDF file. However, this makes the file **less portable**, as a standard URDF parser will simply ignore these proprietary extensions.
  - **Simplified Actuation:** URDF describes joint limits (position, velocity, effort) but has **no native tags for modeling actuators**, transmissions (like gearboxes), or control loops (e.g., PID controllers). And **no closed kinematic chains** allowed (IE four-bar linkage or delta robot)
- **SDF**
  - Multiple robots and objects. Can describe the entire world, including physics, lighting, and environmental models.
  - Native sensor models and plugins. Closed kinematic loops and complex joint types.
  - Does not describe the communication protocols and networking paths in a system, energy requirements of components, software version requirements for a truly distributed complex system of systems.

HCDF - Hardware Configuration Descriptive Format

Approach takes the best of SDF but removes some of the unneeded components. Designed to describe what all the components of a system are supposed to do and how the interact in a complete Cyber Physical System.

```xml
<!--Dynamic surface examples-->
<surface name="">
    <prop name="">
    </prop>
    <hydrofoil name="">
    </hydrofoil>
    <aerofoil name="">
    </aerofoil>
    <aileron name="">
    </aileron>
    <elevon name="">
    </elevon>
    <rudder name="">
    </rudder>
    <wheel>
    </wheel>
    <track>
    </track>
</surface>
```

```xml
<!--Actuator examples-->
<motor name="">
    <servo name="">
    </servo>
    <brushless name="">
    </brushless>
    <brushed name="">
    </brushed>
    <gas name="">
    </gas>
    <pneumatic name="">
    </pneumatic>
    <hydraulic name="">
    </hydraulic>
    <soft name="">
    </soft>
</motor>
```

```xml
<!--Fuel system examples-->
<energy_storage name="">
    <battery name="">
    </battery>
    <tank name="">
    </tank>
</energy_storage>
```

HCDF - Hardware Configuration Descriptive Format

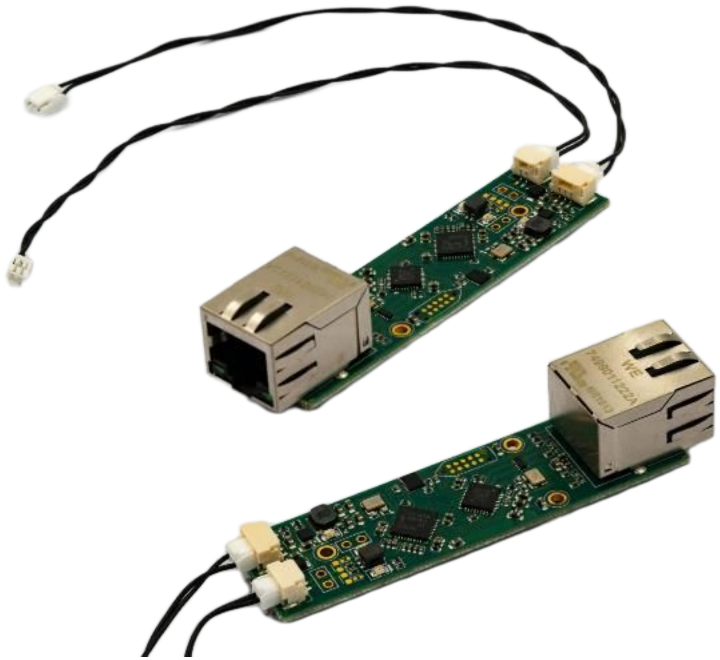
The future of advanced robotics is TSN.

```xml
<!--Computational and controller examples-->
<mcu name="">
    <pose_cg></pose_cg>
    <mass></mass>
    <board></board>
    <software name="">
        <version></version>
        <params>
        </params>
    </software>
</mcu>
<comp name="">
    <pose_cg></pose_cg>
    <mass></mass>
    <board></board>
    <software name="">
        <version></version>
        <params>
        </params>
    </software>
</comp>
```

```xml
<!--Digital and physical linkage examples-->
<link name="">
    <digital name="">
        <wired name="">
        </wired>
        <wireless name="">
        </wireless>
    </digital>
    <power name="">
        <consumer name="">
        </consumer>
        <provider name="">
        </provider>
    </power>
    <physical name="">
        <fixed name="">
        </fixed>
        <rotational name="">
        </rotational>
        <translational name="">
        </translational>
    </physical>
</link>
```

```xml
<!--Sensor examples-->
<sensor name="">
    <optical>
    </optical>
    <inertial>
    </inertial>
    <rf>
    </rf>
    <chemical>
    </chemical>
    <force>
    </force>
    <electrical>
    </electrical>
</sensor>
```

Building out the Transition to T1:   T1ADAPT

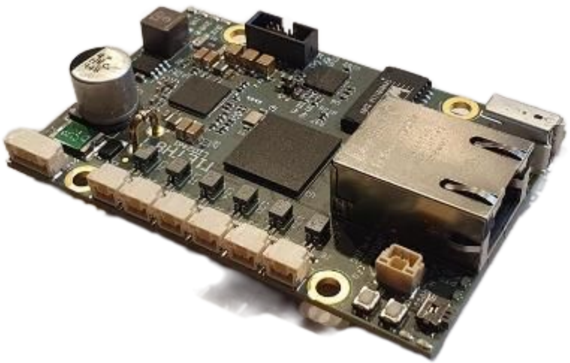


100base-T1 to RJ45 Adapter

- Stand alone T1 “two wire” ethernet adapter reference design for fast evaluation or adaptation
  - Laptop to T1 network
  - Between two T1 adapters
  - Allow existing embedded board to use T1

- NXP components
  - TJA1101 Ethernet PHY
  - LPC microcontroller
  - USB or other 5V input

MR-T1ETH8

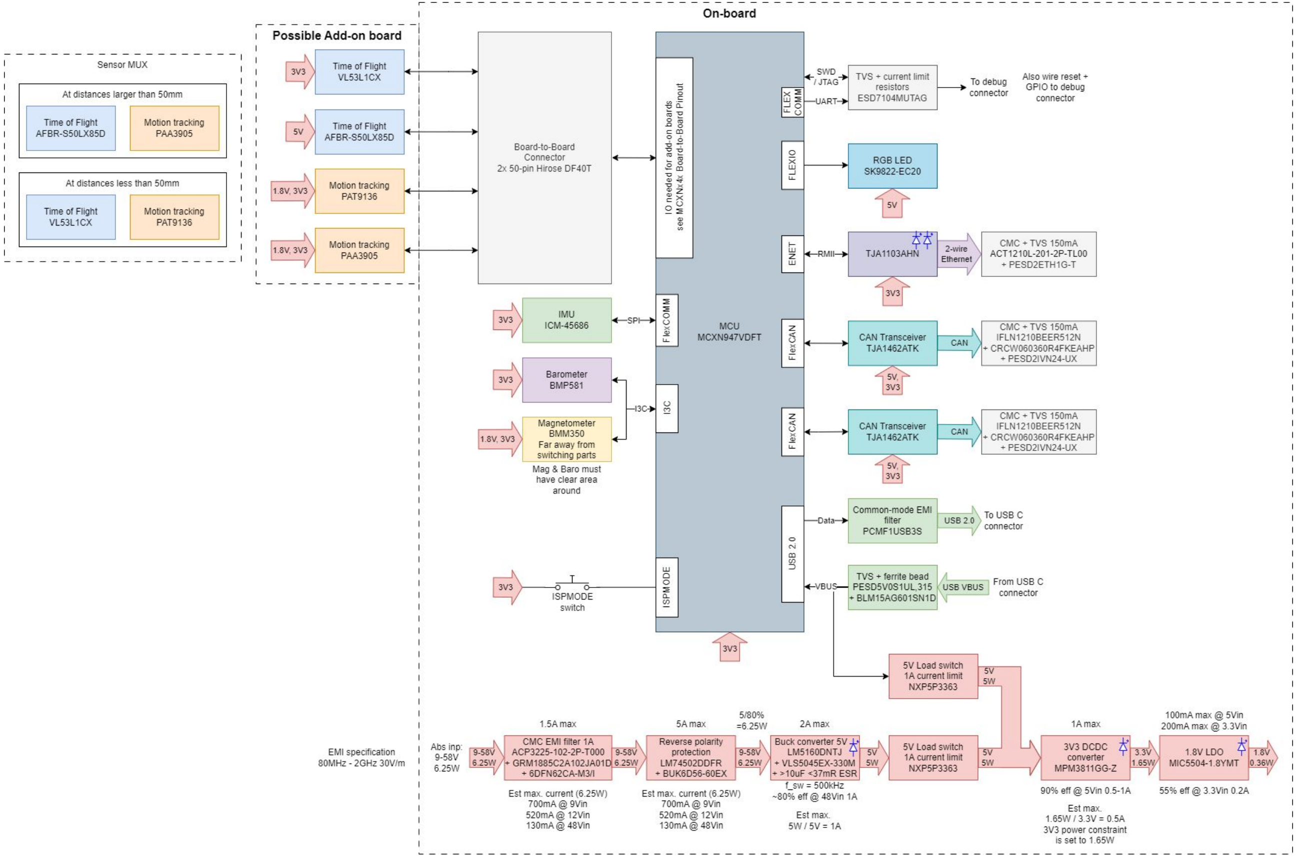


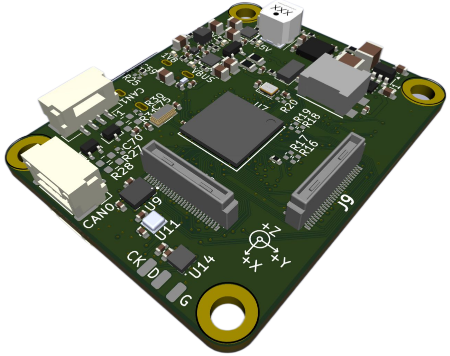
8Port 100base-T1 Ethernet switch

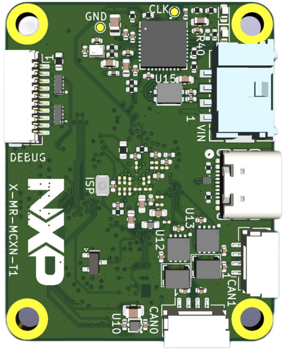
- 100BaseT1 “two wire” ethernet switch application reference design*
  - (6x) 100Base-T1 Two wire Ethernet
  - (1x) 100Base-TX Ethernet w/ traditional RJ45
  - (1x) 1000base-TX Gigabit w/ IX industrial connector

- NXP parts
  - SJA1110 10 port ethernet switch IC supporting TSN
  - VR5510 automotive PMIC
  - SE050 Secure Element with NFC interface
- Small 75x50mm board

SPINALI MCXN T1 HUB (MR-MCXN-T1)







Join the Future of Assured Autonomy

Software Development:

https://github.com/CogniPilot

Discussions/Social:

https://discord.com/invite/CogniPilot

Release Documentation:

COGNIPILOT https://cognipilot.org

Foundation Team:

COGNIPILOT https://cognipilot.com/team