UAV autopilot architectures versus AADL

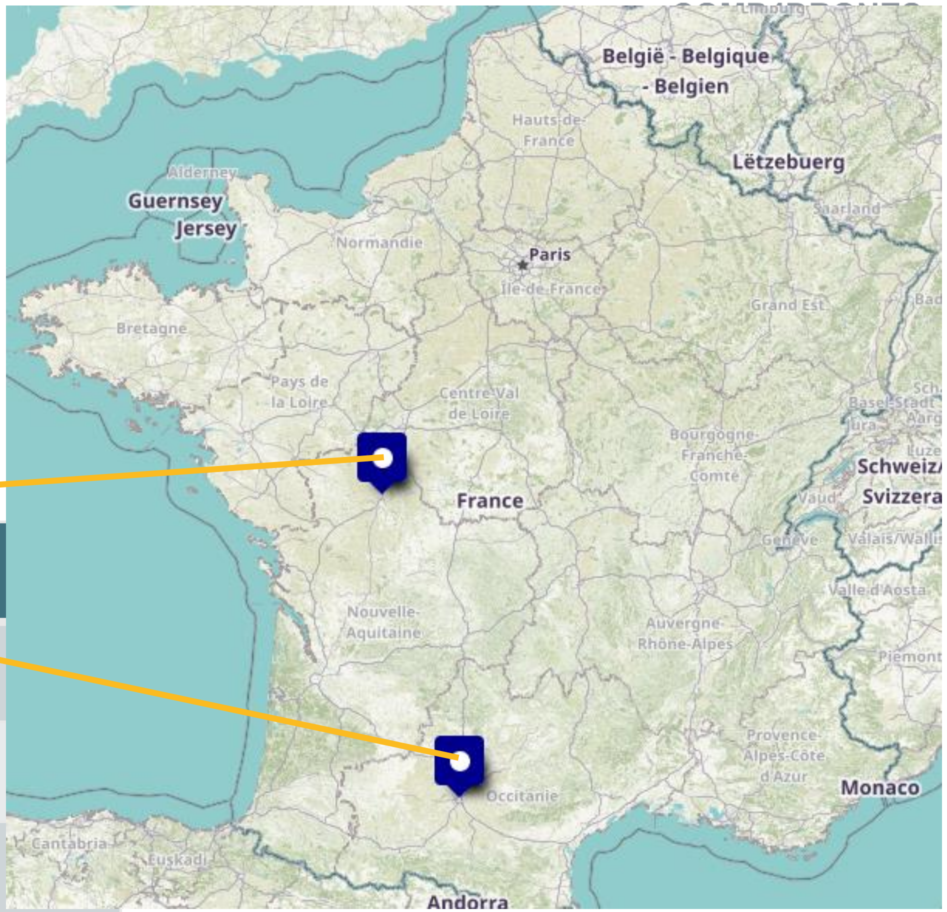
ADEPT 2024
Emmanuel GROLLEAU (LIAS, ISAE-ENSMA, France)
Uses a lot of material from Gautier HATTENBERGER (ENAC, France)

ECSEL Joint Undertaking
Electronic Components and Systems for European Leadership

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826610. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Spain, Austria, Belgium, Czech Republic, France, Italy, Latvia, Netherlands.

Results from a joint work

LIAS, ISAE-ENSMA	ENAC
Matheus Ladeira (PhD. 2023)	Gautier Hattenberger
Emmanuel Grolleau	Fabien Bonneval
Yassine Ouhammou	Alexandre Bustico
Soulimane Kamni (PhD. 2024)	
Several interns	



PaparazziUAV

https://wiki.paparazziuav.org/wiki/Main_Page

Log in

Main page | Discussion | Read | View source | View history | Search PaparazziUAV

Paparazzi
The Free Autopilot

- Home
- Hardware
- Software
- FAQ
- Downloads
- Remembering Hecto

Communication

- Mailing list
- Gitter chat
- Contact
- Wiki account

Development

- How to contribute
- Developer Guide
- Doxygen docs
- Readthedocs pages
- Git repository
- Build tests

Wiki tools

- Recent changes
- Random page
- Editing Help

Tools

- What links here
- Related changes
- Special pages
- Permanent link
- Page information

Print/export

- Create a book
- Download as PDF
- Printable version

Welcome to Paparazzi UAV

Blog | Mailinglist | Chat | Download | GitHub | YouTube

Paparazzi UAV (Unmanned Aerial Vehicle) is an open-source drone hardware and software project encompassing autopilot systems and ground station software for multicopters/multirotors, fixed-wing, helicopters and hybrid aircraft that was founded in 2003. Paparazzi UAV was designed with autonomous flight as the primary focus and manual flying as the secondary. From the beginning it was designed with portability in mind and the ability to control multiple aircraft within the same system. Paparazzi features a dynamic flight plan system that is defined by mission states and using way points as “variables”. This makes it easy to create very complex fully automated missions without the operators intervention. For more project information, see here.

Legal Disclaimer

The Paparazzi software source and hardware design is distributed without any guarantee. Before flying, please refer to your country's national aviation regulation for Unmanned Aerial Systems, or the one of the country you intend to overfly.

General

- System Overview
  General overview of the Paparazzi system
- Getting Started
  An overview of getting started with Paparazzi
- FAQ
  Frequently Asked Questions
- Downloads
  Software source code, hardware schematics, etc.
- User List and Gallery
  List of users. Photos, Videos, etc.
- Media, Papers and Links

Latest Stable Release: v6.4.0_stable

Semaphore CI Build Status

Download as tarball or checkout the **v6.4** branch from git.

Releases can be found at https://github.com/paparazzi/paparazzi/releases

Paparazzi UAV Blog

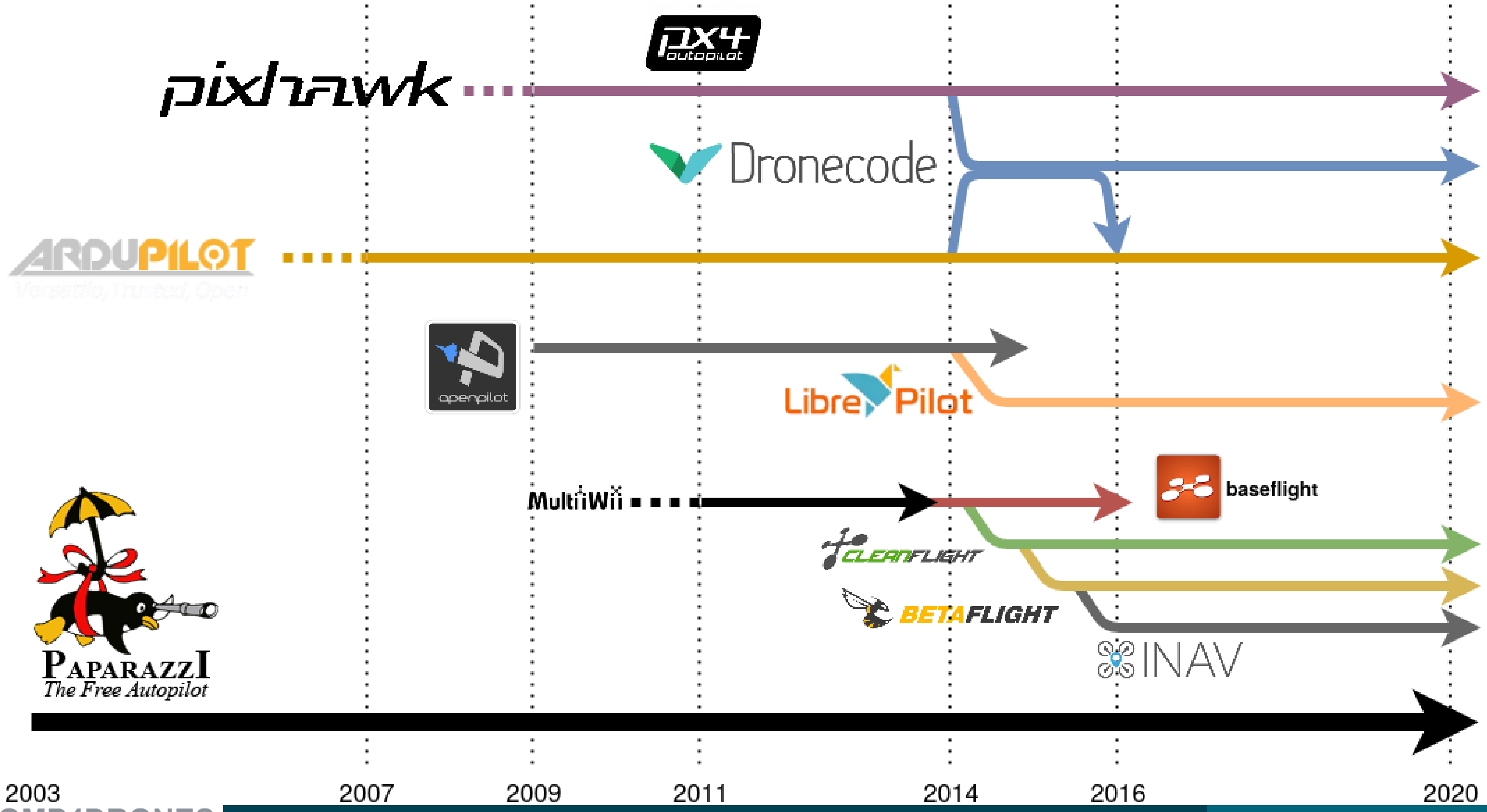
ENAC Team wins at IMAV2022

The International Conference and Competition on MAV was organized by TUDelft mid-

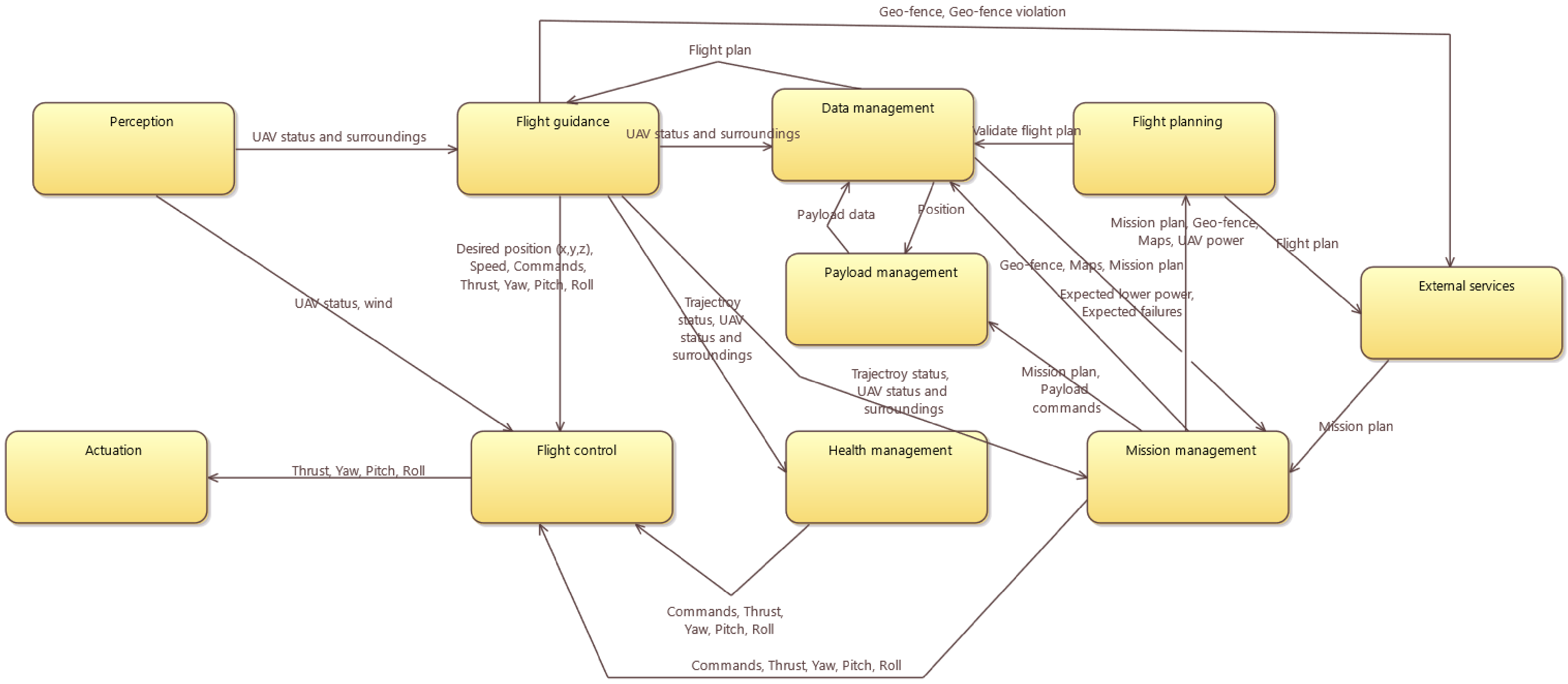
Console (2) | Queries (353) | Debug log (1329) | Request | PHP includes (1127) | MediaWiki: 1.37.1 | PHP: 7.3.27-1~deb10u1 | Time: 0.92181 | Memory: 11.11 MB (Peak: 11.44 MB)

COMP4DRONES

Open-source UAV projects history



Main functions of an autopilot



Main problem tackled in C4D-wp3

COMP4DRONES

Helping drone manufacturers to design safe drones

- Most drone manufacturers use off-the-shelf open source autopilots
- An open source autopilot is an autopilot generation framework
- Several customization means
  - Companion board and network ad-hoc standard (MAVLink)
    - E2E Delays 10’s to 100’s ms
  - Local process or thread sharing cores with the AP using a middleware
    - E2E Delays several to 10’s ms
  - Local function inserted within the AP control thread
    - E2E Delays < 1 ms

How to help designers to design it safely without too much time and effort?

Open source autopilot generation frameworks

COMP4DRONES

A very complex code

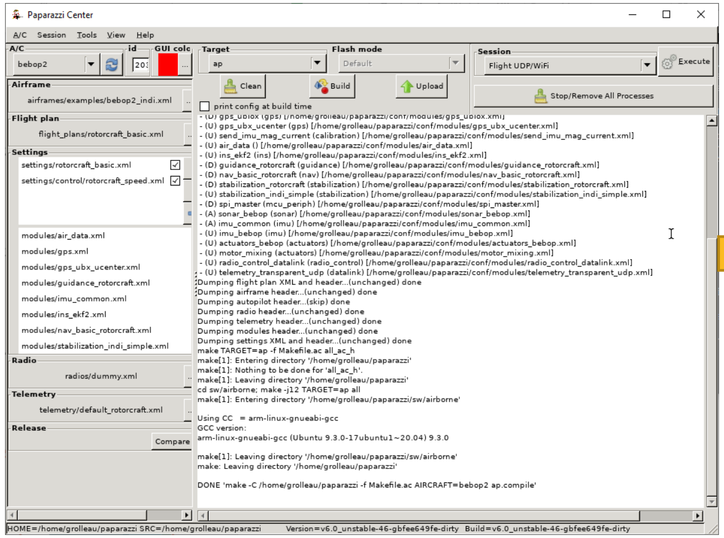
	Paparazzi	PX4
# C files # loc	2’015 513’517	526 182’009
# C++ files # loc	137 40’261	1025 287’175
# Makefiles # loc	366 56’815	534 26’663
# config files # loc	1260 (XML) 160’656	260 (Kconfig) 1’819

- <10% of which is used for a single autopilot instance

Objective

COMP4DRONES

Retro-engineer AP to allow its customization and analysis

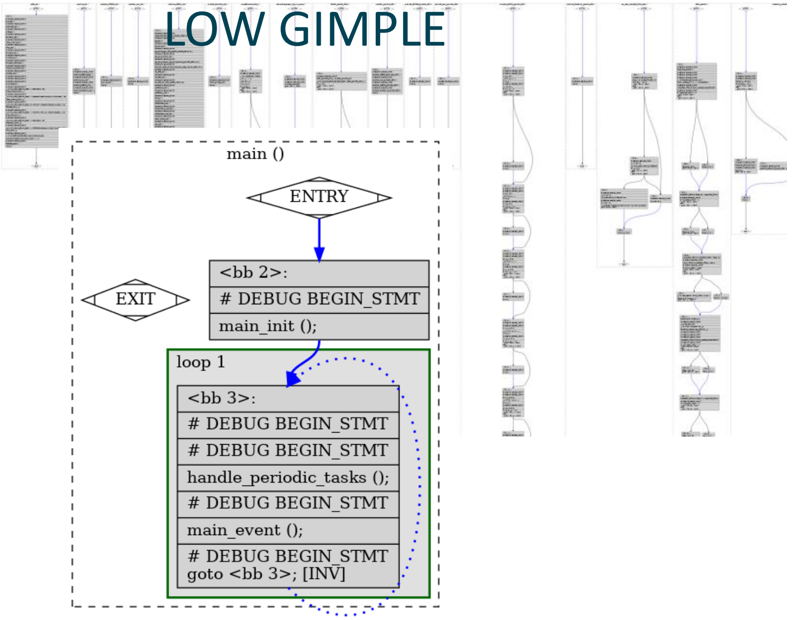


paparazzi

gcc

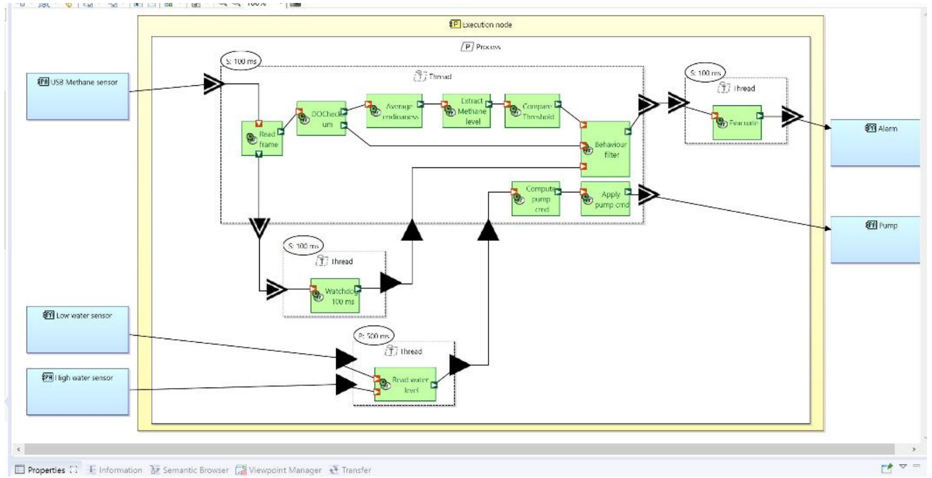
Abstract Syntax Tree

LOW GIMPLE



ANTLR+Java

ADL+C4D point of view for Capella



Analysis & Design tools

One must model to live and not live to model

'One must eat to live, and not live to eat.', L'Avare (The Miser), Molière

Main objectives of our model:

⇒performance analysis (schedulability, E2E delays)

⇒help the designer to understand where and when data is used and changed

⇒allow bridges to and from tools

Example of drone sensors&actuators

Bebop 2

Dual core Cortex A9 + GPU + 8GO flash memory

- OS Linux SMP PREEMPT

Sensors

- IMU
  - Magnetos 3-axes (AKM 8963) **I2C-1**
  - Gyros et Acceleros 3-axes (MPU6050) **I2C-2**
- Ground speed and position
  - Vertical camera optical flow sensor (∀16 ms compare images) **I2C-0**
- Position
  - GNSS Ublox Neo M8N (GPS and Galileo and (GLONASS or BeiDou)) **UART**
    - Frame frequency configurable between 1 and 30Hz
- Altitude
  - Baro MS5607 **I2C-1**
- Low altitude
  - Sonar **SPI** to trigger readings **Analog** values

Actuators, clocks, payload

Bebop 2

WiFi Module

GPIO

- Misc alimentations
- On/Off button

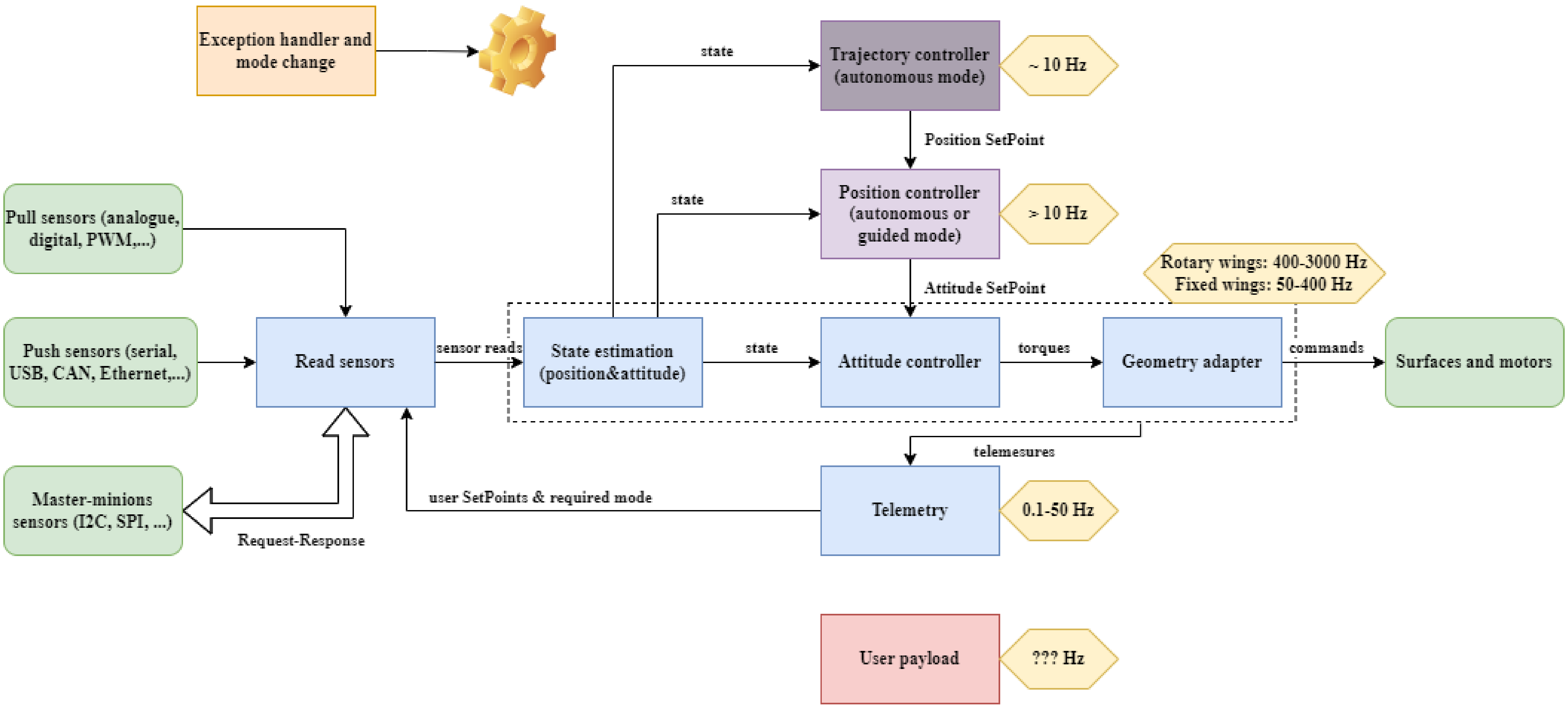
PWM

- Heating résistances for the IMU
- Clocks for gyro&acceleros
- Clocks for cameras

Motors

- BrushLess Driver Controller (BLDC) **I2C-1**

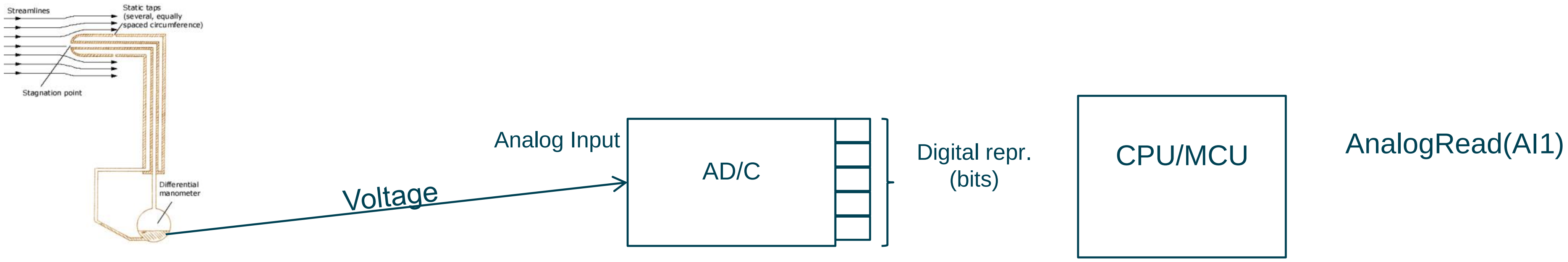
Heart of an autopilot



Analog sensors

The system « Pulls » the value from the sensor

Conversion delay ½ to 40 µs

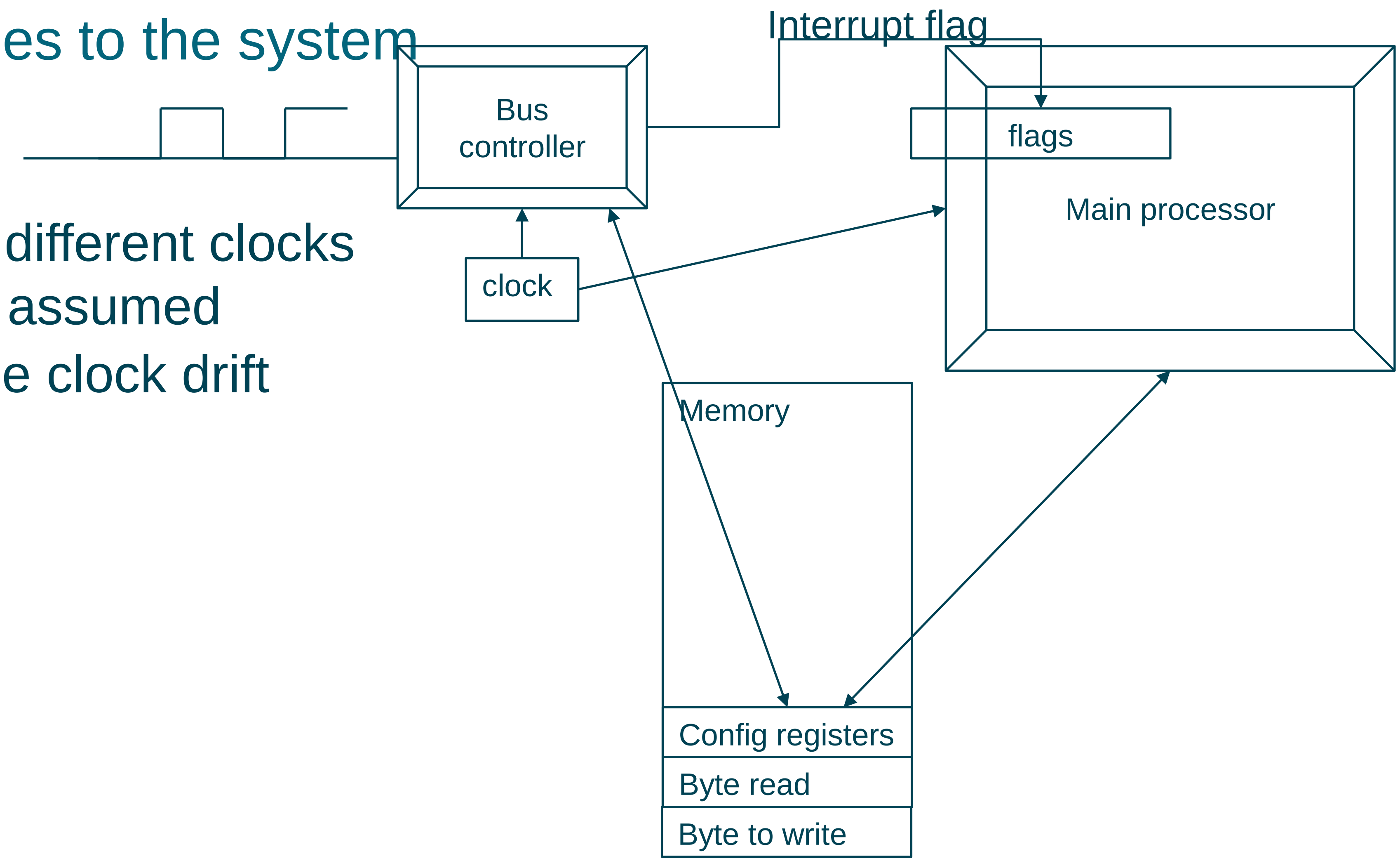


Pitot tube

Sporadic sensors (UART/USB/CAN/..)

Sensor « pushes » values to the system

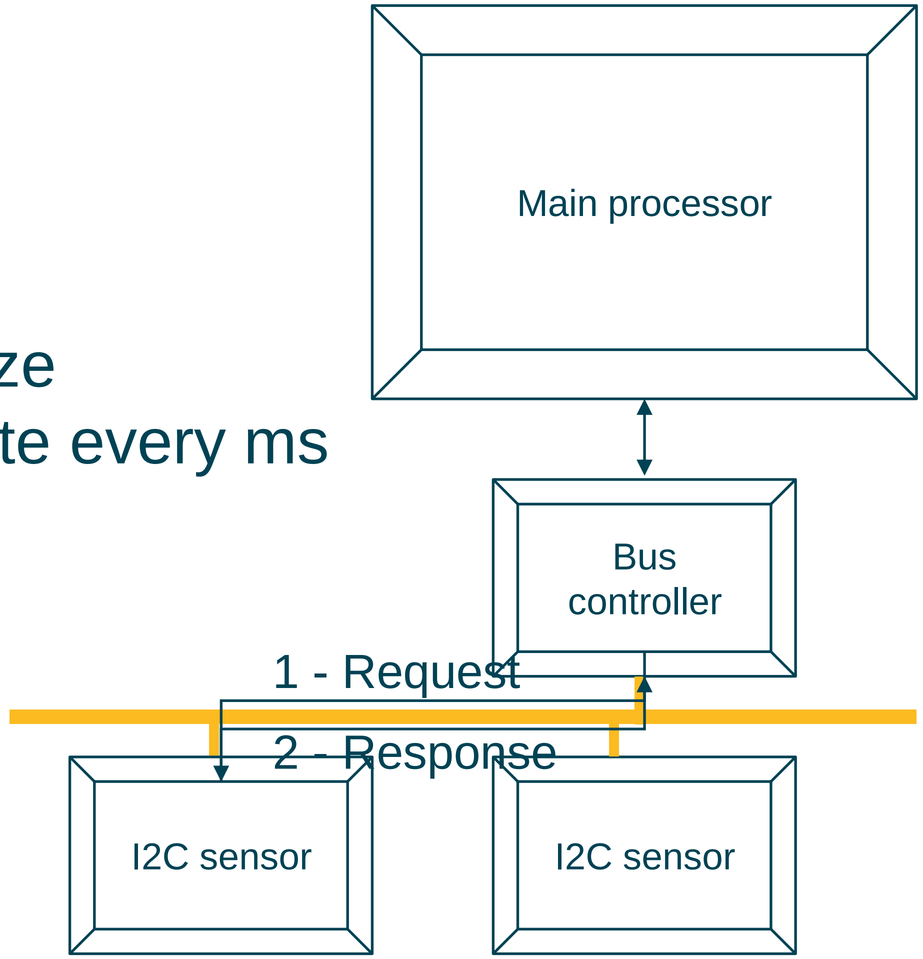
- No control of when a frame arrives
- Sensor and CPU have different clocks
- Minimum delay can be assumed
  - E.g. 5% admissible clock drift



Master/Minion sensors (I2C/SPI/…)

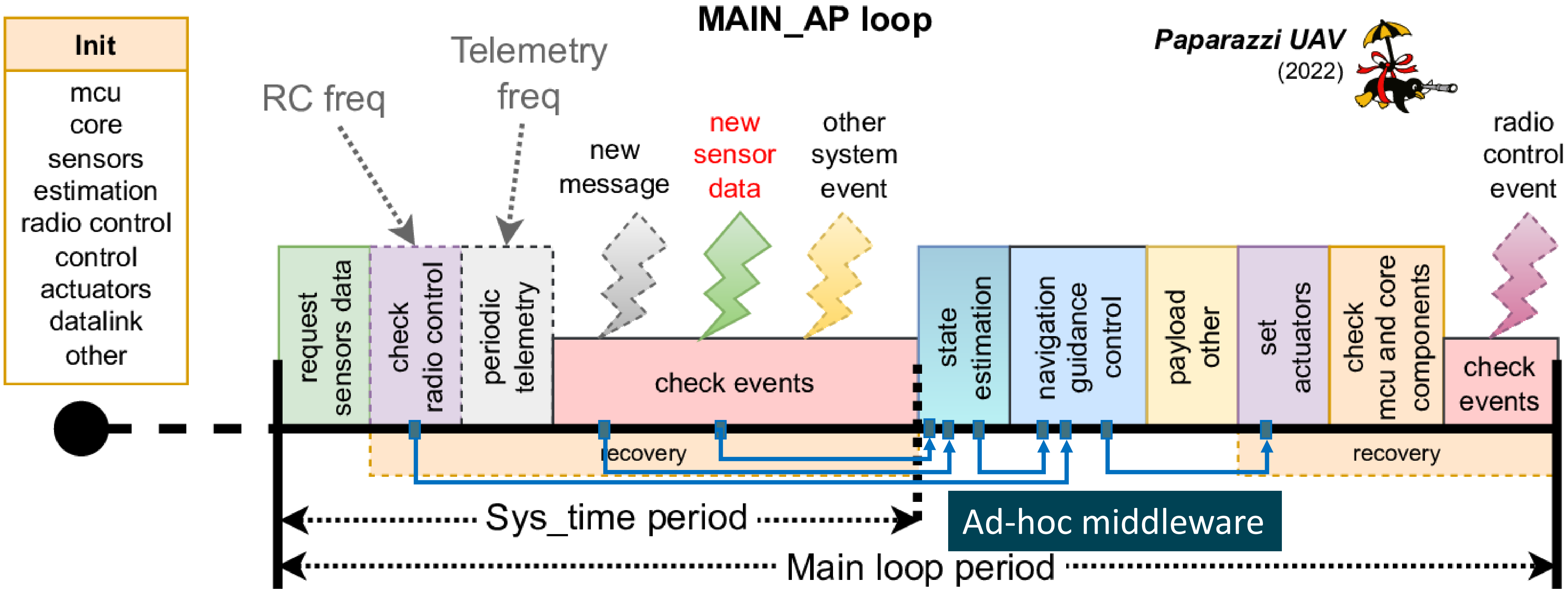
Request - response

- More and more present in UAV
  - ~50% of sensors on a Bebop 2
- Can be surprisingly slow
  - « Fast » 400kHz I2C
- 300 to 500 µs depending on the frame size
- But at 1kHz the whole loop should execute every ms



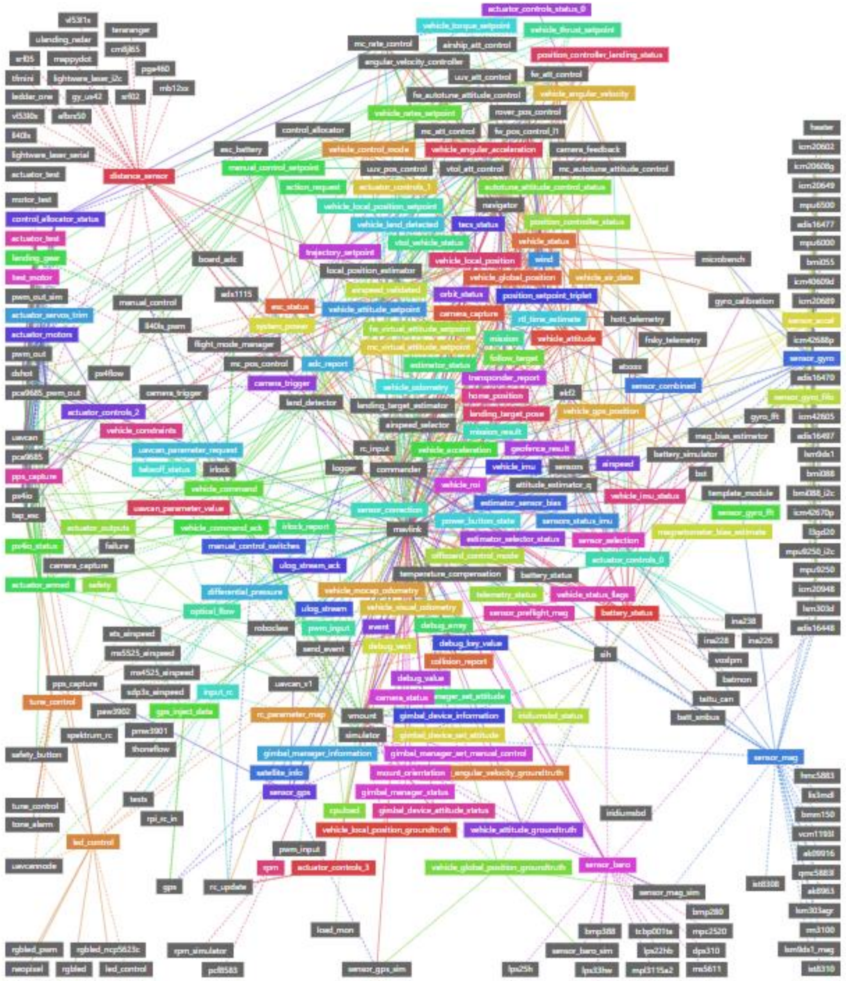
Internal architecture of Paparazzi

A fine tuned monolithic central thread, executing functions as a cyclic executive



The importance of the middleware

Example: PX4 uORB topics in this ad-hoc middleware

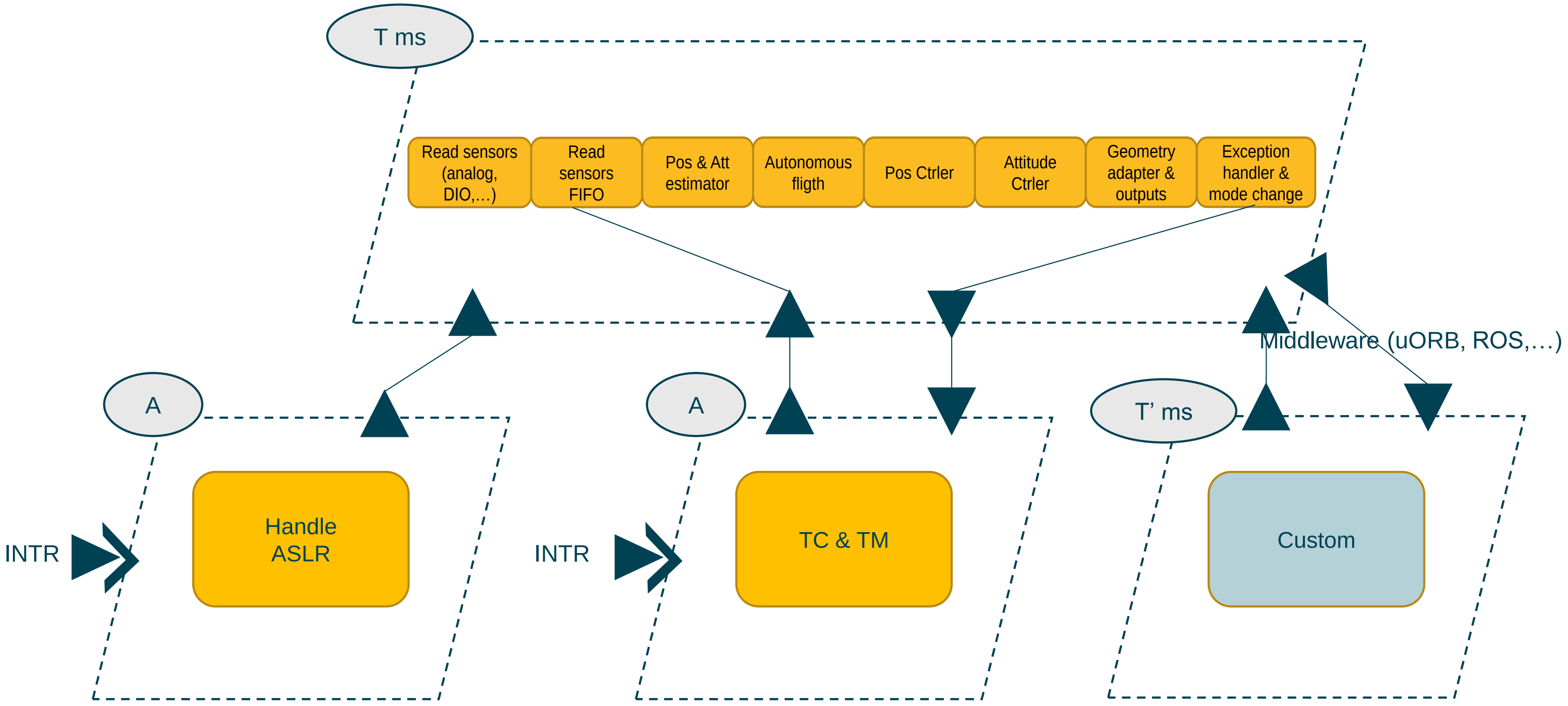


Can I change the attitude information bewteen its measurement and its use in the INDI controller?

What is the maximum age of this state information when my custom function runs?

Multithreaded architecture

I/O thread and 1 or 2 central control thread

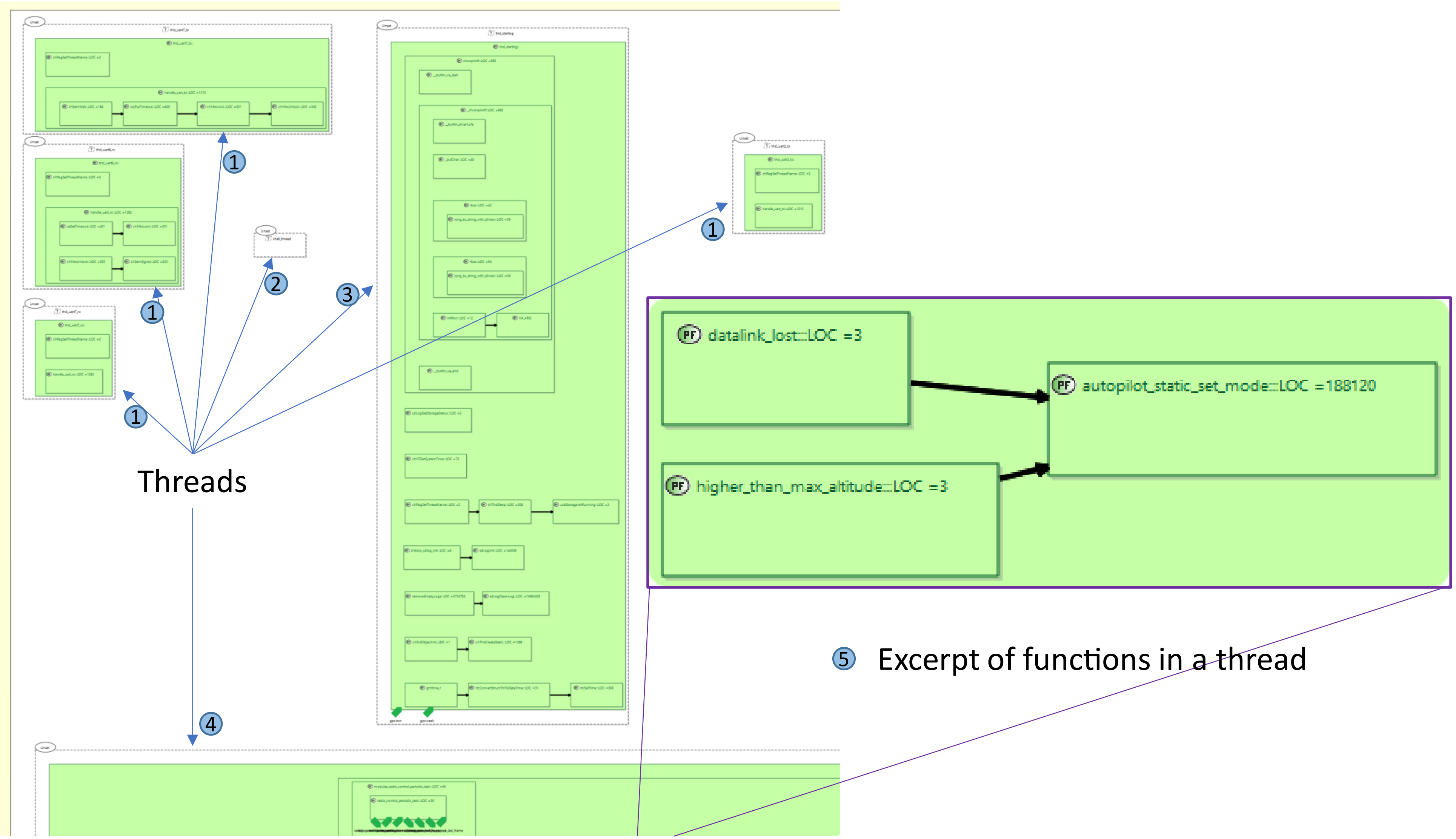


What did we miss in AADL?

Self-preparing to face AADL experts critics

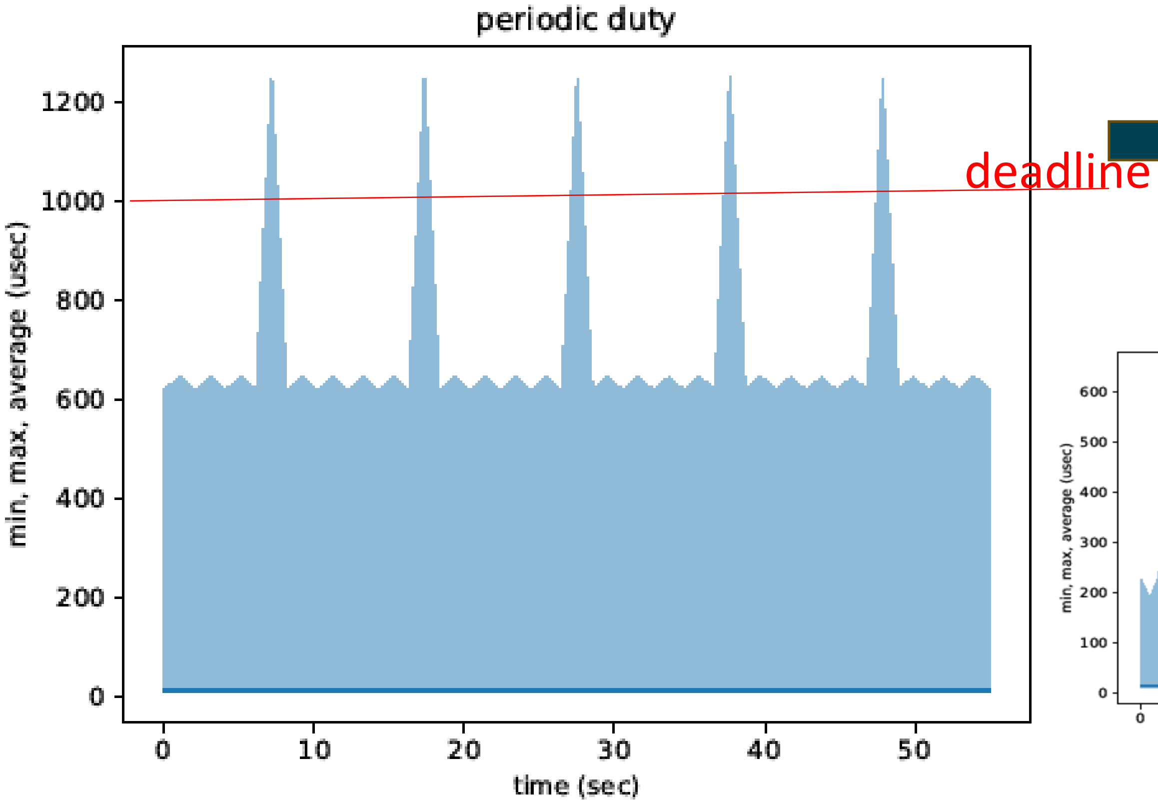
- Model both a cyclic executive executed in a thread and threads and processes
- Ability to express, for an internal « function » (or is it a cyclic executive thread?), a « period », and dependencies
    - « *f* is executed once every 10 periods, with an initial offset of 2 periods »
    - « *f* is sporadic with a minimal inter-activation of 5 periods »
    - « *f* precedes *g*, but both *f* and *g* have different periods »
- Represent hiérarchies of functions
- Have a « light » and abstract representation of data accesses through middlewares
- Connect functions to an ontology

Our partial attempt as a Capella viewpoint

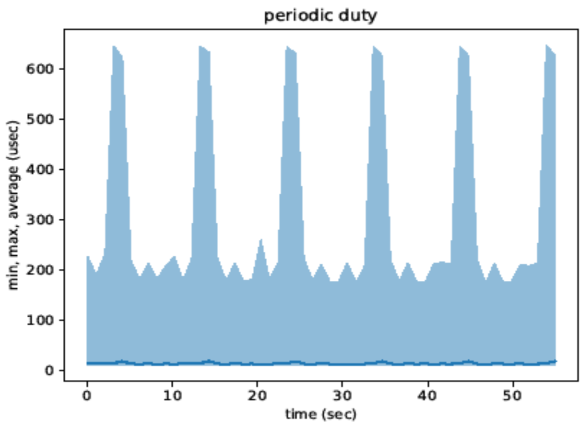


What we did with our model?

Illustrated the use of GCD+, a tool to pick offsets in an offset free system



Changed initial offsets of custom periodic functions to avoid overloads of the main control loop



Discussion

- Several systems, including UAV o-t-c autopilots use a mix
  - Processes and Threads
  - One or several threads run a cyclic executive
    - Including periodic, sporadic, precedence constrained tasks with offsets
- In the context of retro-engineering it is interesting to present the user
  - Hierarchical functions calls
  - That can be more or less detailed depending on the needs
- The vast use of middlewares requires a simple representation of the accesses, with an easy way to distinguish the scope (intra-thread, multi-threadn multi-process, distributed)