PReGO: a Generative Methodology for Satisfying Real-Time Requirements on COTS-based Systems

– Definition and Experience Report –

UNIVERSITEIT VAN AMSTERDAM

Benjamin Rouxel <benjamin.rouxel@uva.nl>
Clemens Grelck <c.grelck@uva.nl>

SDU
UNIVERSITY OF SOUTHERN DENMARK

Ulrik Pagh Schultz <ups@mmmi.sdu.dk>

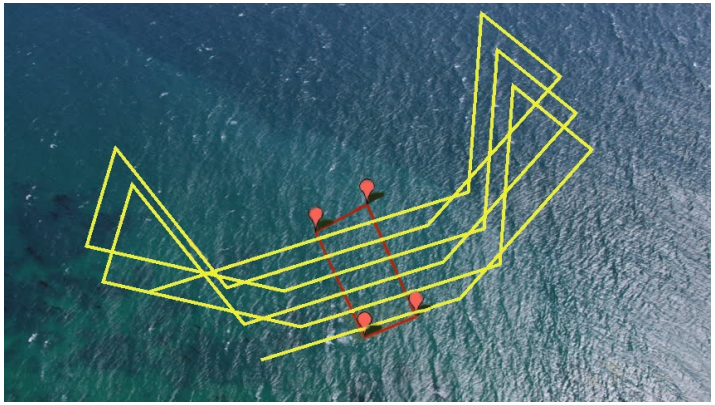
TNO innovation for life

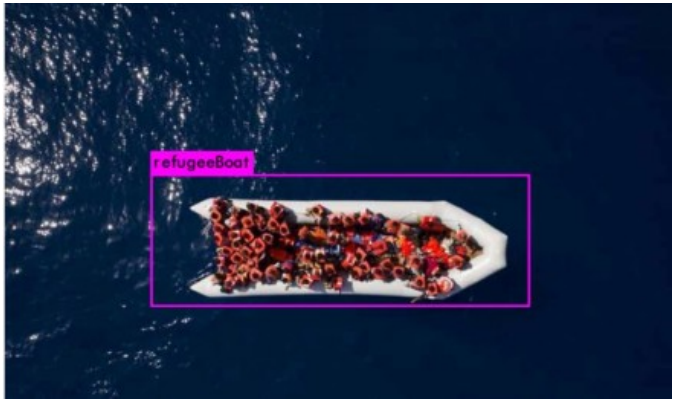
Benny Åkesson <benny.akesson@tno.nl>

Jesper Holst <jsh@sky-watch.com>
Ole Jørgensen <olj@sky-watch.com>

Time, Ene
Multi/Many

An UAV to Save Lives



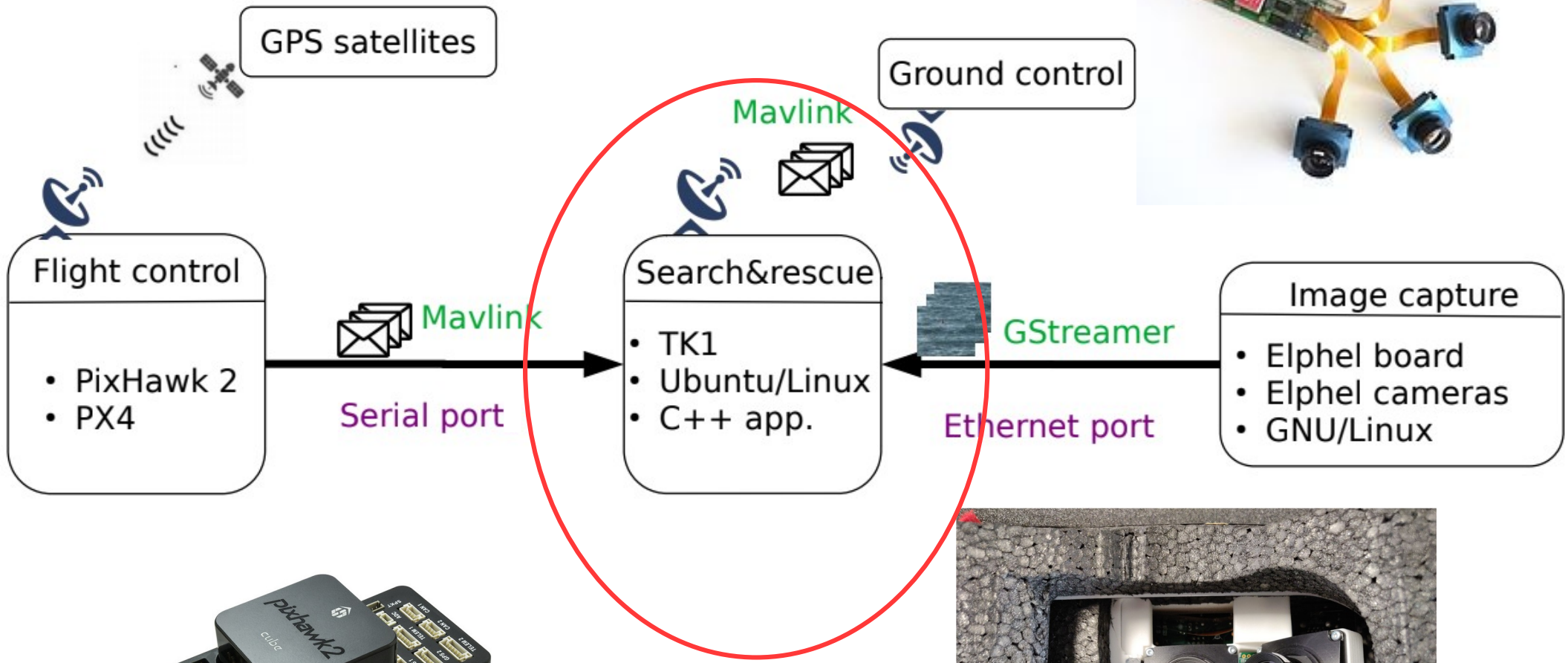


- Fixed-wings drone
- Fly above the sea
- Detect life boats
- Contact rescue teams

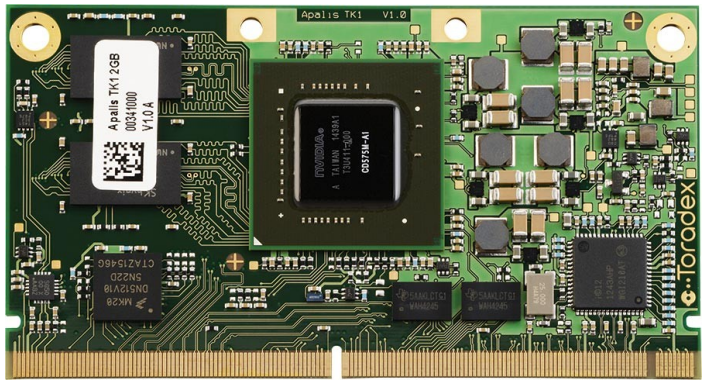
Requirements

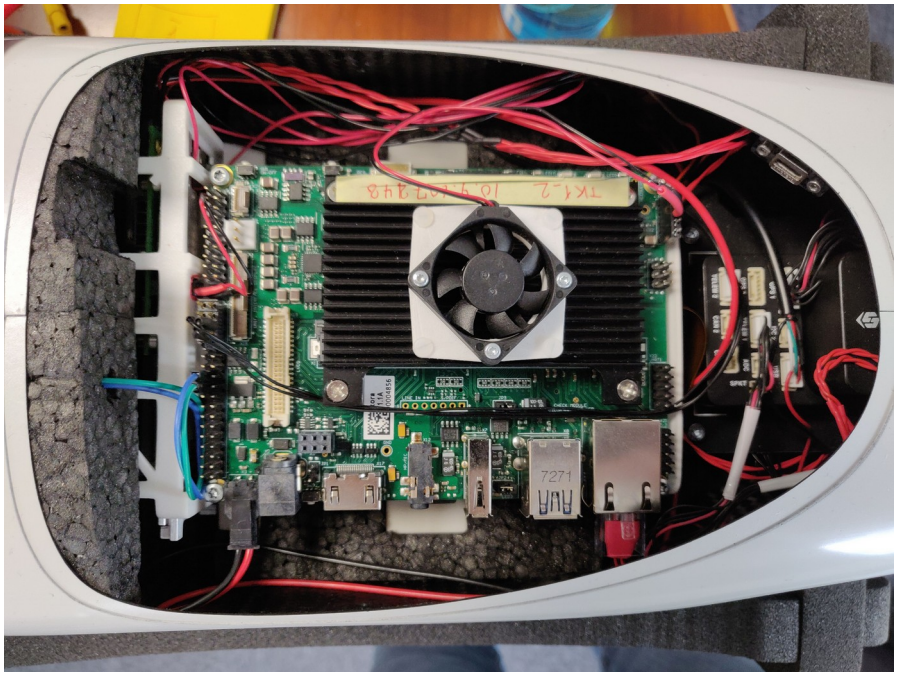
- Record videos/images at 2fps
- Relaunch if crashed (software)
- Always detect in the widest possible area
- Limit false positive
- No false negative

System Overview



Hardware Specifications





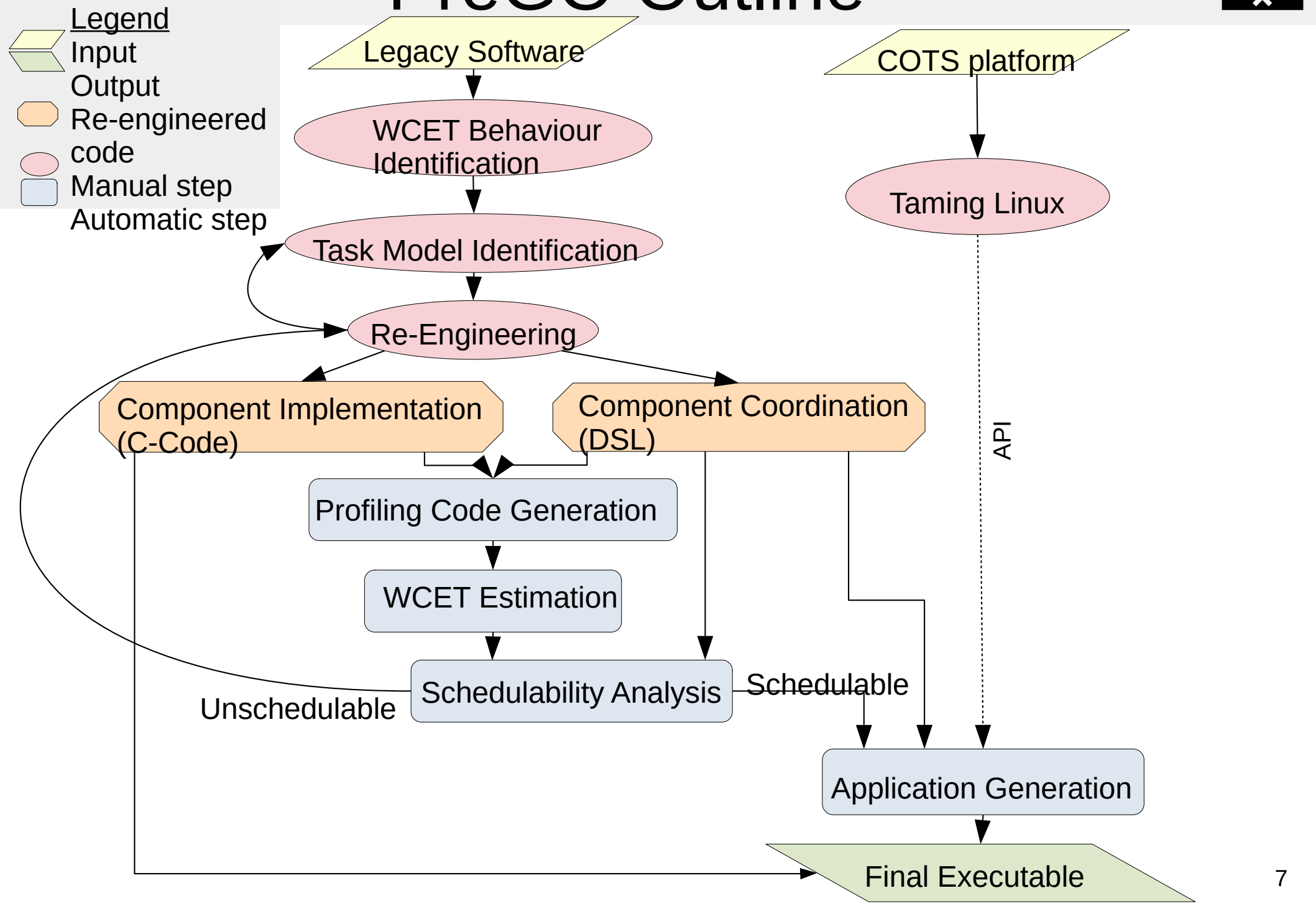
- Toradex Apalis TK1
- Cortex-A15 (4 cores)
- Kepler GPU (192 cuda cores)
- RAM 2GB
- Power dissipation 3-15W

- Embedded oriented
- Performance
- Low energy
- Long time support

Operating Environment

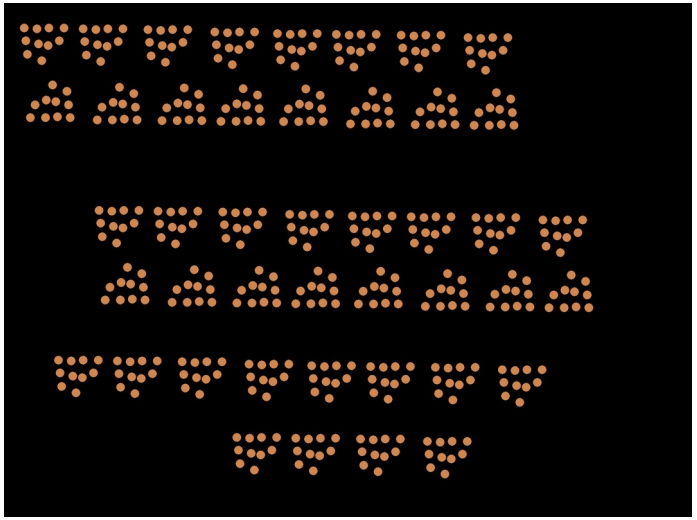
- No (easy) possibility for RTOS
- Need GStreamer
- Need Cuda drivers

PreGO Outline

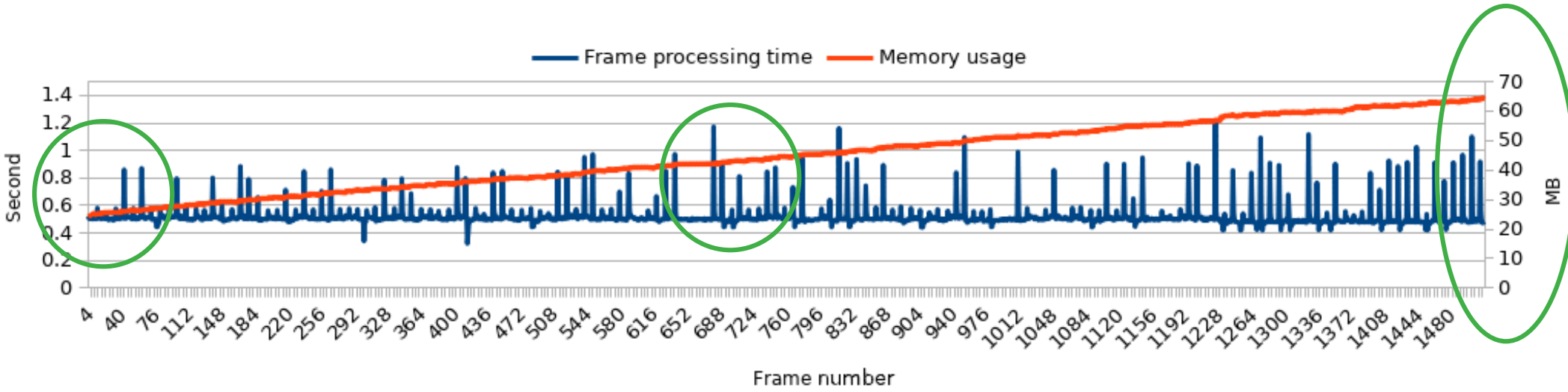


Worst-case scenario

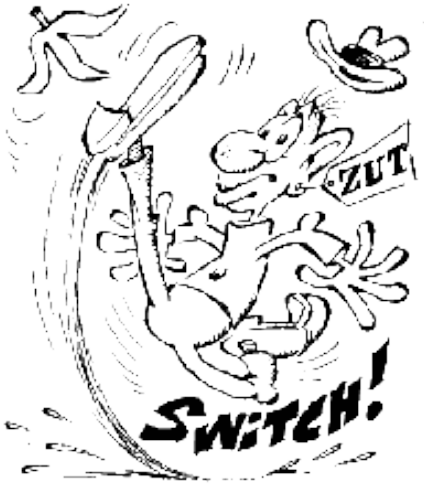
- Highest altitude: 1000m
- Object minimum size: 0.5m^2
- 2 frames per seconds
- Boats in every frames
- Number of pixels? Objects position? Maximum number of objects?



First run



AVG energy consumption: 3.8 J/frame



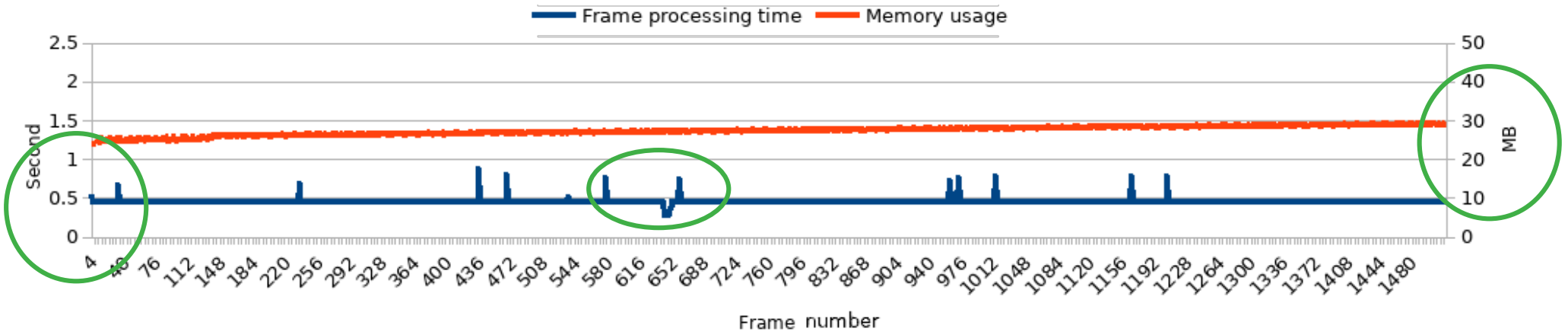
- Execution time not stable
- Execution time too high

Enforce timing requirements

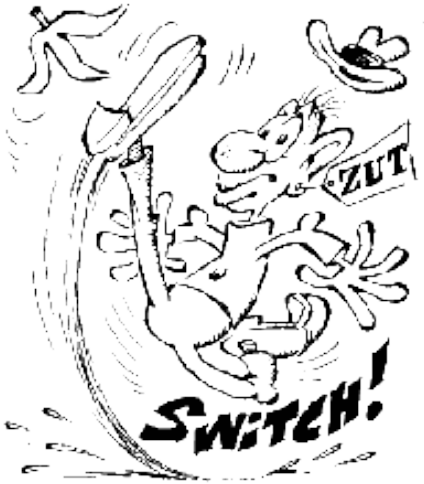
Taming Linux

- Avoid kernel interference
- Disable desktop applications
- Interrupt handlers on 1 core
- Threaded interrupts
- Disable freq. scaling governor
- Avoid preemption of RT-priority task
  - Disable RT throttling

Taming Linux: results

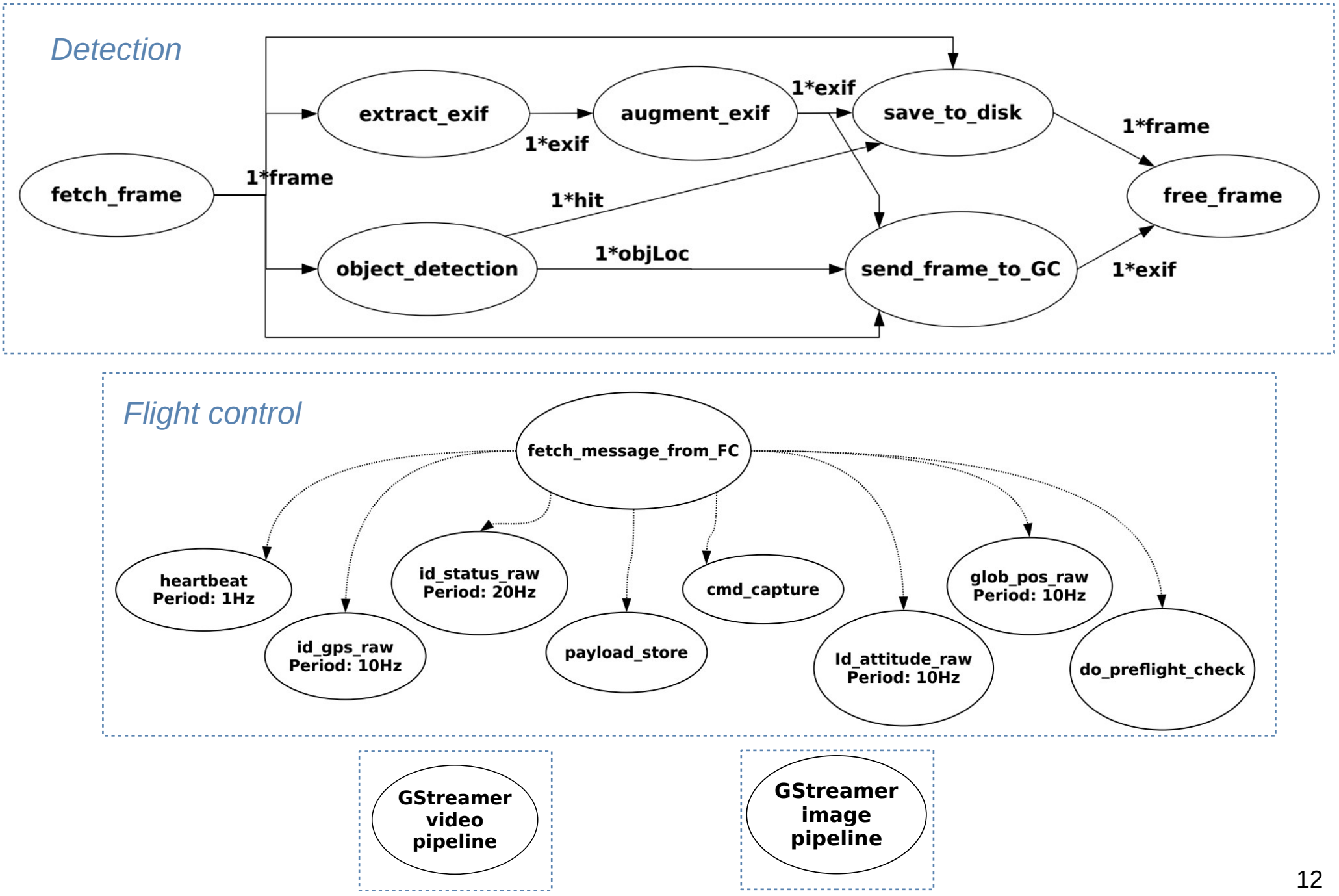


AVG energy consumption: 4.9 J/frame



- More stable
- Few deadline misses
- Larger energy consumed
- Soft real-time ok

Task identification

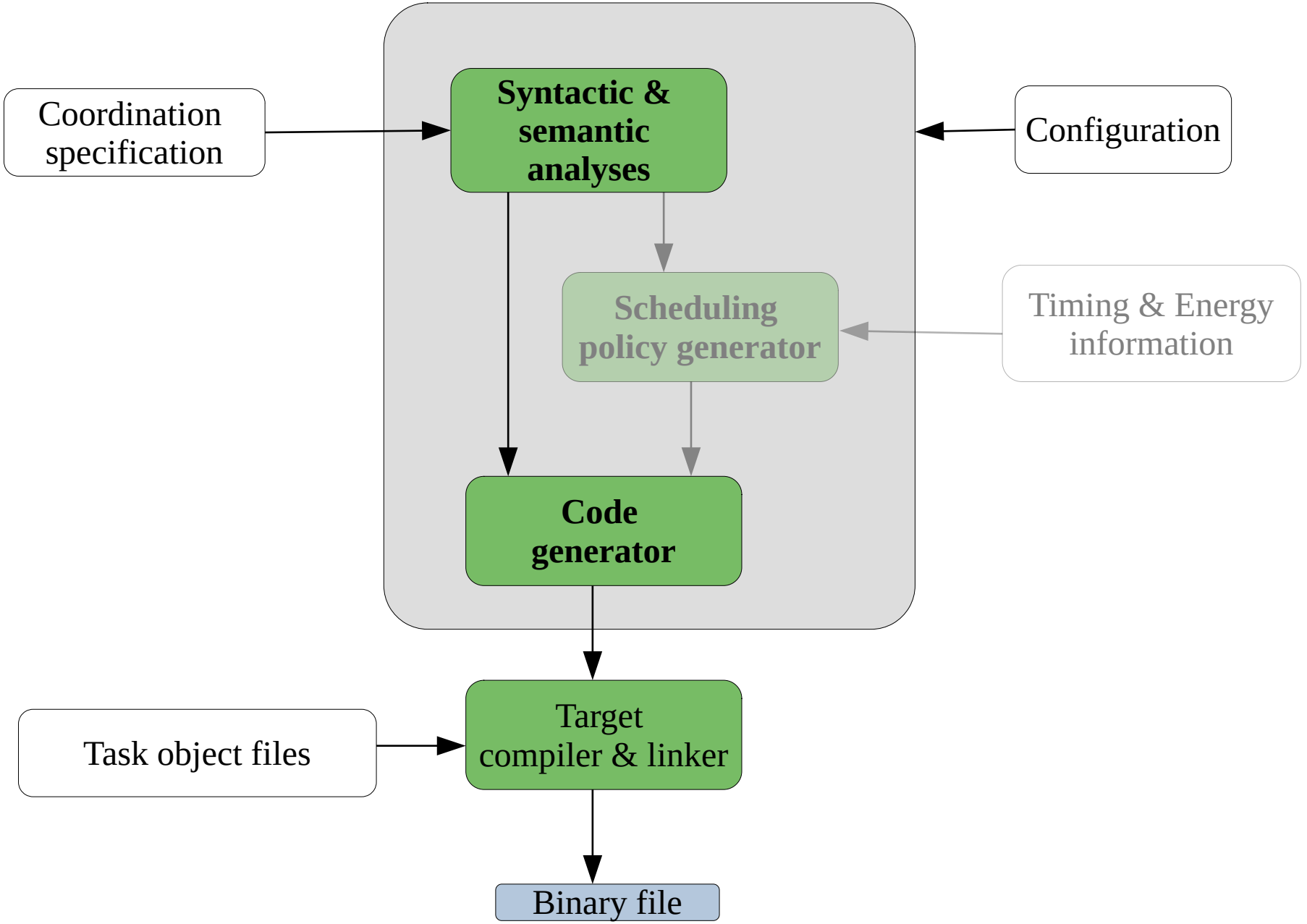


Re-engineering

- Isolate tasks code
- Create the high-level representation
  - DSL from [Roeder et al., Coordination'2020]

```
app Drone {
    components {
        fetch_message_from_FC {
            outputs [... (capture, 1,mavlink_msg)]
        }
        ...
        ctrl_cmd_capture {
            inputs [(in, 1, mavlink_msg)]
        }
        fetch_new_frame {
            outputs [(out,1, frame)]
            period 500 ms
        }
        object_detection {
            inputs [(in,1, frame)]
            outputs [(hit,1, integer) (objLoc,1, vector_int)]
        }
        ...
    }
    edges {
        fetch_message_from_FC.capture ->
                ctrl_cmd_capture.in
        fetch_new_frame.out ->
                object_detection.in & ...
        ...
    }
```

Profiling & Scheduling



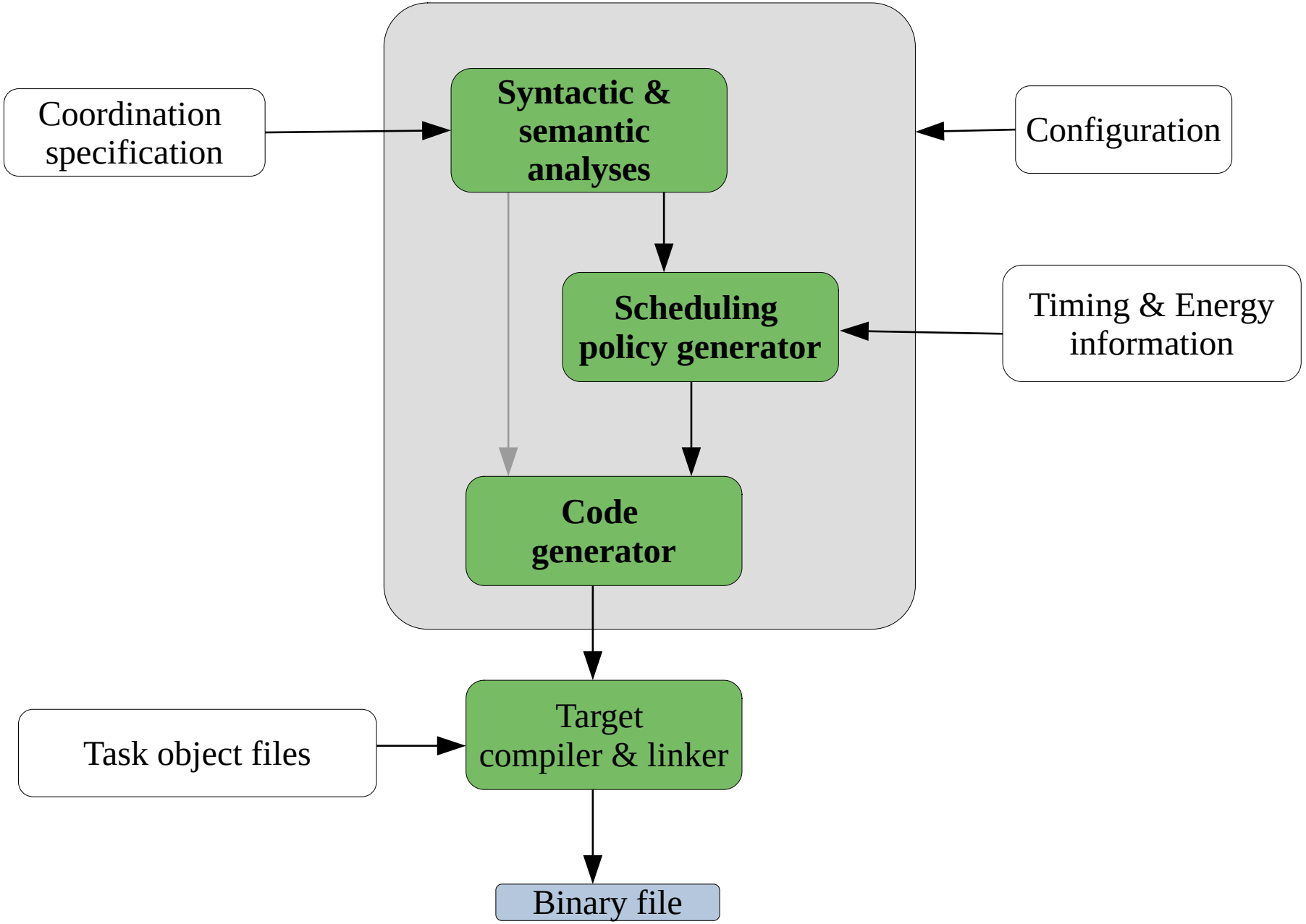
Code Generation

Profiling code

- Sequential version
- Main function
- FIFO buffer management
- PowProfiler
  - [Adam et al., IJPP'2019]

```
buffer_frame_t fetch_new_frame_out;
...

void pop_frame(buffer_frame_t *buffer, frame** res) {
    ...
}
void push_frame(buffer_frame_t *buffer, frame* val) {
    ...
}

void __coord_fetch_new_frame_() {
    frame* out = NULL;
    user_fetch_new_frame(&out);
    push_jpegFrame1(&fetch_new_frame_out, out);
}
int main(int argc, char **argv) {
    main_init();

    for (size_t __i__ = 0; __i__ < 100; ++__i__) {
        powprof_newiter(__i__, 100);

        powprof_start("fetch_new_frame");
        fetch_new_frame_();
        powprof_stop("fetch_new_frame");

        powprof_start("object_detection");
        object_detection_();
        powprof_stop("object_detection");
        ...
    }

    main_clean();
    return EXIT_SUCCESS;
}
```

Profiling & **Scheduling**



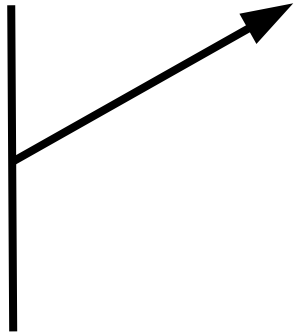
Scheduling

- Most controllable policy
  - Partitioned, no migration, fixed-priorities, non-preemptive
- Schedulability analysis
  - [Casini et al., RTSS'2018]
- OS configurations
  - Set affinity
  - Single core per thread
  - Single priority per task + SCHED_FIFO
  - Disabled RT throttling

Code Generation

Profiling code

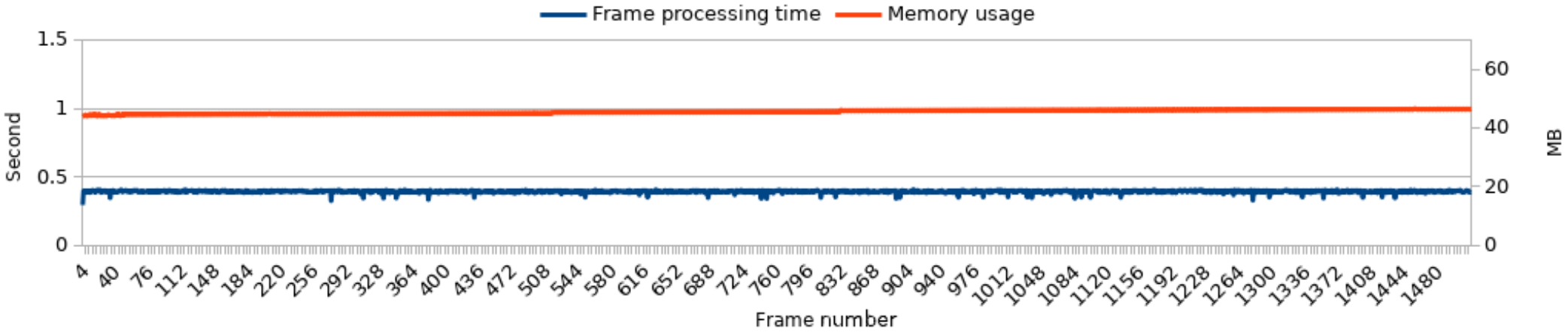
- Sequential version
- Main function
- FIFO buffer management



Final code

- Parallel version
- Threads/ Processes management
- Synchronisation

Final run



AVG energy consumption: 3.1 J/frame

Future work

- Improve task management
  - Thread pool
  - Multi-mode scheduling
- Improve energy management
  - Tailored scheduling policy

Take away

- PReGO methodology
  - Design real-time systems
  - Automatic generation
- Timing control in stock Linux

https://bitbucket.org/uva-sne/coordinationcompiler