# Verification of Autonomous Robots
## A "humble" Roboticist Bottom Up Approach

Félix Ingrand
LAAS-CNRS

RoboSoft:
Software Engineering for Robotics
Royal Academy of Engineering, London, UK
13-14, November 2019

Travelling from Toulouse to London for this presentation has produced
$CO_2$: 13,4 kg



Part of this work is funded by the H2020 European project AI4EU

# This is "Minnie"

Segway RMP 440
R&D platform... not an autonomous car but... definitely not a toy robot

- Fast (up to 8 m/s)
- GPS
- Gyro (measures theta/wz)
- IMU (angular velocities and acceleration)
- 2 recent CPUs (but I only use one)

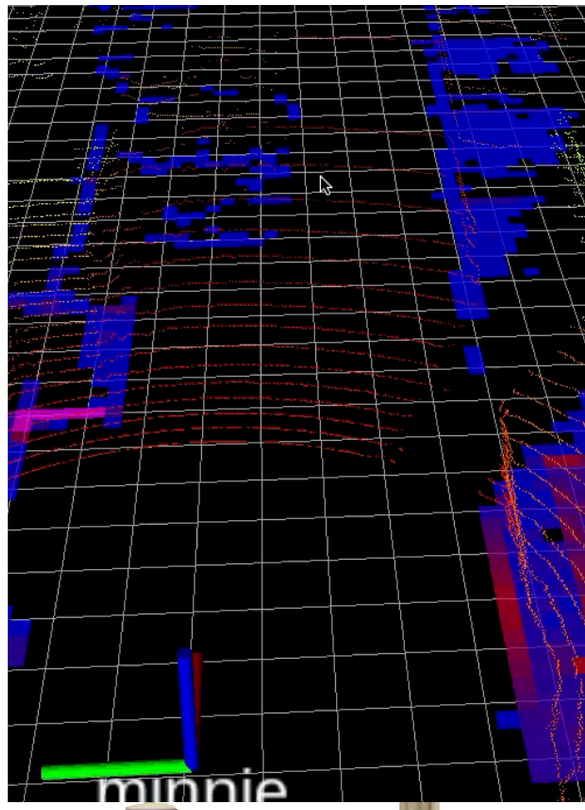
LAAS CNRS

# Let's focus on Minnie's velodyne

- Velodyne lidar sensor
  - they are used on most "autonomous cars" (except Tesla)
  - Point Cloud: 64K 3D points at 10 Hz...
  - ... at 5 m/s, Minnie has moved 50cm while scanning
  - you need to register where you are while scanning
  - rebuild the corrected scan

KEY FEATURES

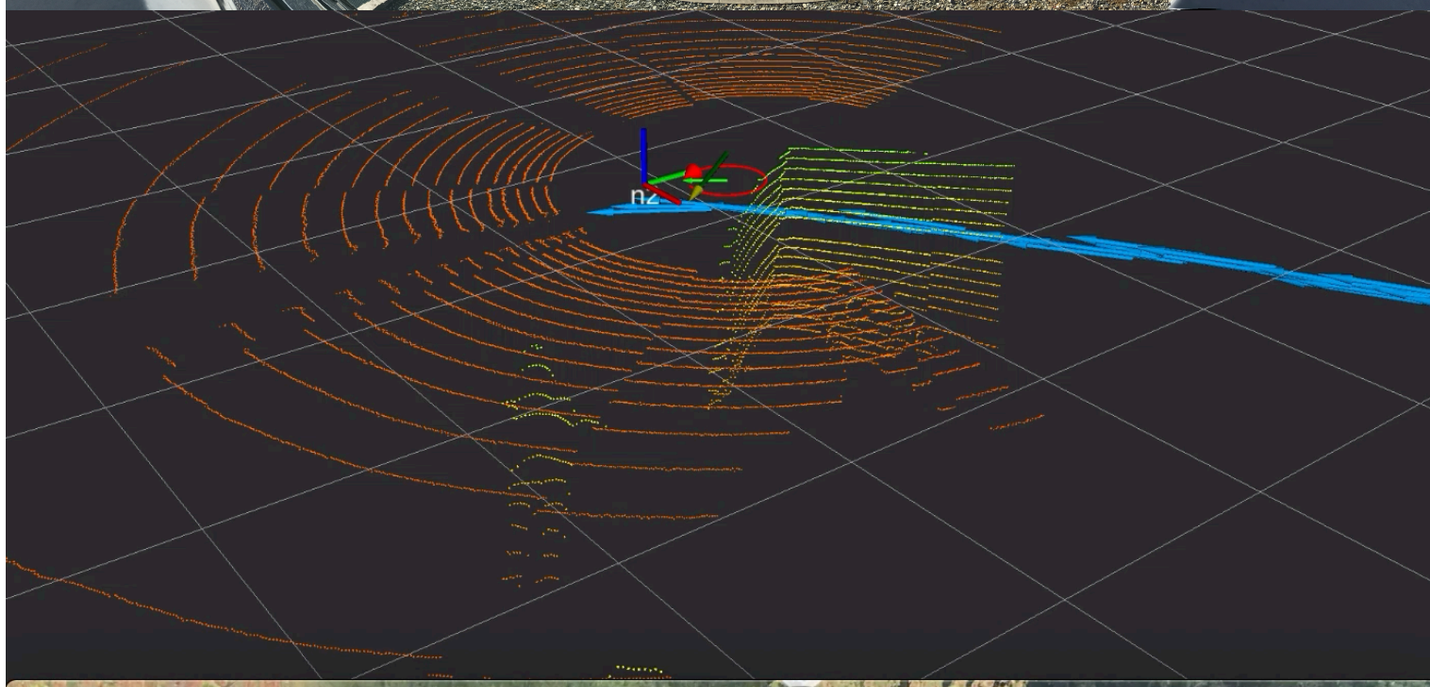
- Dual Returns
- ± 2 cm accuracy
- 1kg (plus 0.3kg for cabeling)
- 32 Channels
- 80m-100m Range
- Up to ~1.39 Million Points per Second
- 360° Horizontal FOV
- +10° to -30° Vertical FOV
- Low Power Consumption
- Rugged Design



LAAS CNRS

# What do we use Minnie for?

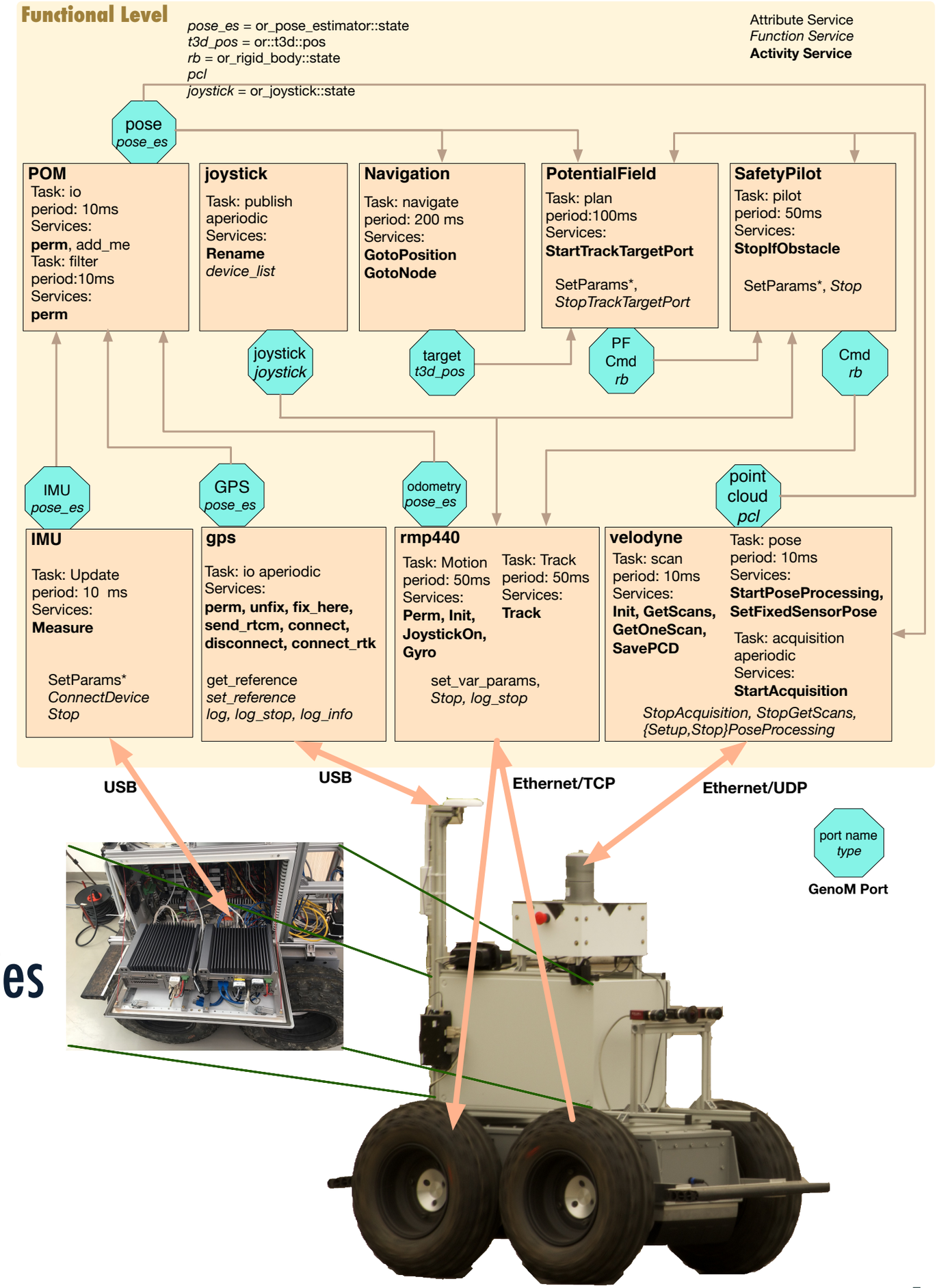
- Patrolling the parking lot in the lanes/crossings graph
- without scratching my colleague's car (parked or moving)
- without bumping in people



# Minnie software architecture

- 9 components
- 9 ports
- (13 + 9) tasks (period)
- 38 activity services (with automata)
- 41 function services
- 43 attribute services
- 170 codels (14k loc) and their WCET
- 200k loc for all components + libraries

- Everything runs onboard...



*joystick* = or_joystick::state

pose
*pose_es*

**POM**
Task: io
period: 10ms
Services:
**perm**, add_me
Task: filter
period:10ms
Services:
**perm**

**joystick**
Task: publish
aperiodic
Services:
**Rename**
*device_list*

**Navigation**
Task: navigate
period: 200 ms
Services:
**GotoPosition**
**GotoNode**

**PotentialField**
Task: plan
period:100ms
Services:
**StartTrackTargetPort**
SetParams*,
*StopTrackTargetPort*

**SafetyPilot**
Task: pilot
period: 50ms
Services:
**StopIfObstacle**
SetParams*, *Stop*

joystick
*joystick*

target
*t3d_pos*

PF
Cmd
*rb*

Cmd
*rb*

IMU
*pose_es*

GPS
*pose_es*

odometry
*pose_es*

point
cloud
*pcl*

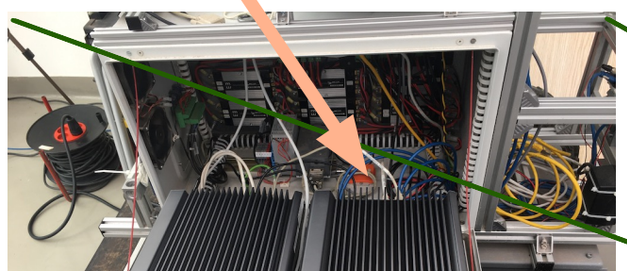
**IMU**
Task: Update
period: 10 ms
Services:
**Measure**
SetParams*
*ConnectDevice*
*Stop*

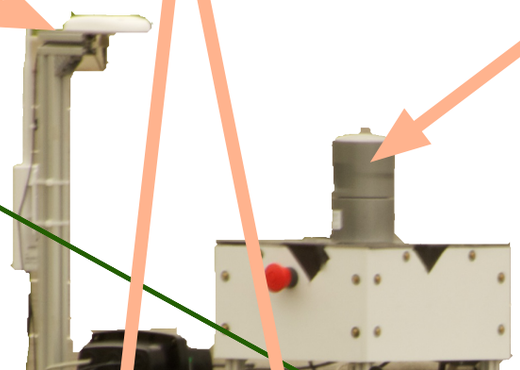
**gps**
Task: io aperiodic
Services:
**perm, unfix, fix_here, send_rtcm, connect, disconnect, connect_rtk**
get_reference
*set_reference*
*log, log_stop, log_info*

**rmp440**
Task: Motion
period: 50ms
Services:
**Perm, Init, JoystickOn, Gyro**
Task: Track
period: 50ms
Services:
**Track**
set_var_params,
*Stop, log_stop*

**velodyne**
Task: scan
period: 10ms
Services:
**Init, GetScans, GetOneScan, SavePCD**
Task: pose
period: 10ms
Services:
**StartPoseProcessing, SetFixedSensorPose**
Task: acquisition
aperiodic
Services:
**StartAcquisition**
*StopAcquisition, StopGetScans, {Setup,Stop}PoseProcessing*

**USB** **USB** **Ethernet/TCP** **Ethernet/UDP**





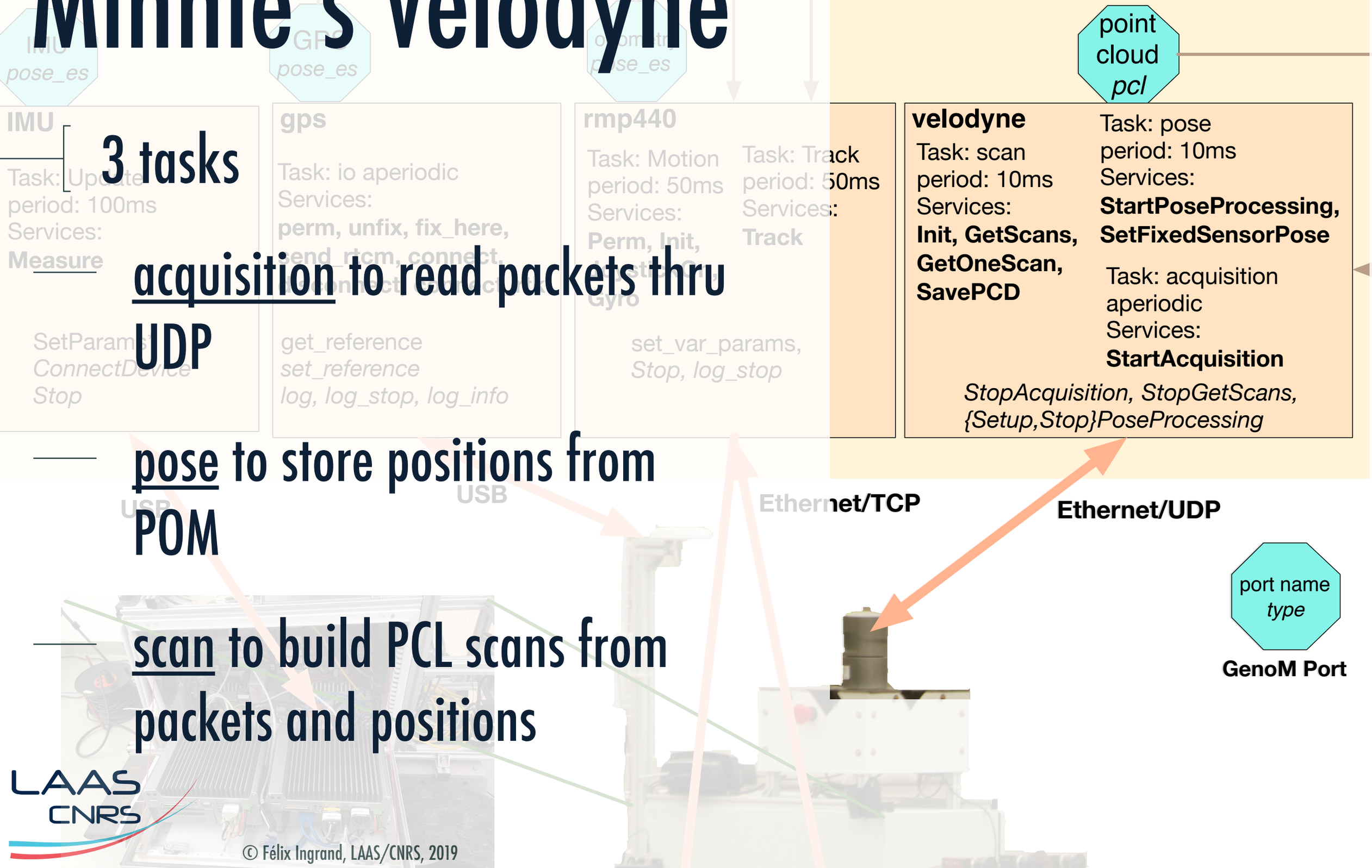
port name
*type*

**GenoM Port**

LAAS CNRS

# Minnie's Velodyne

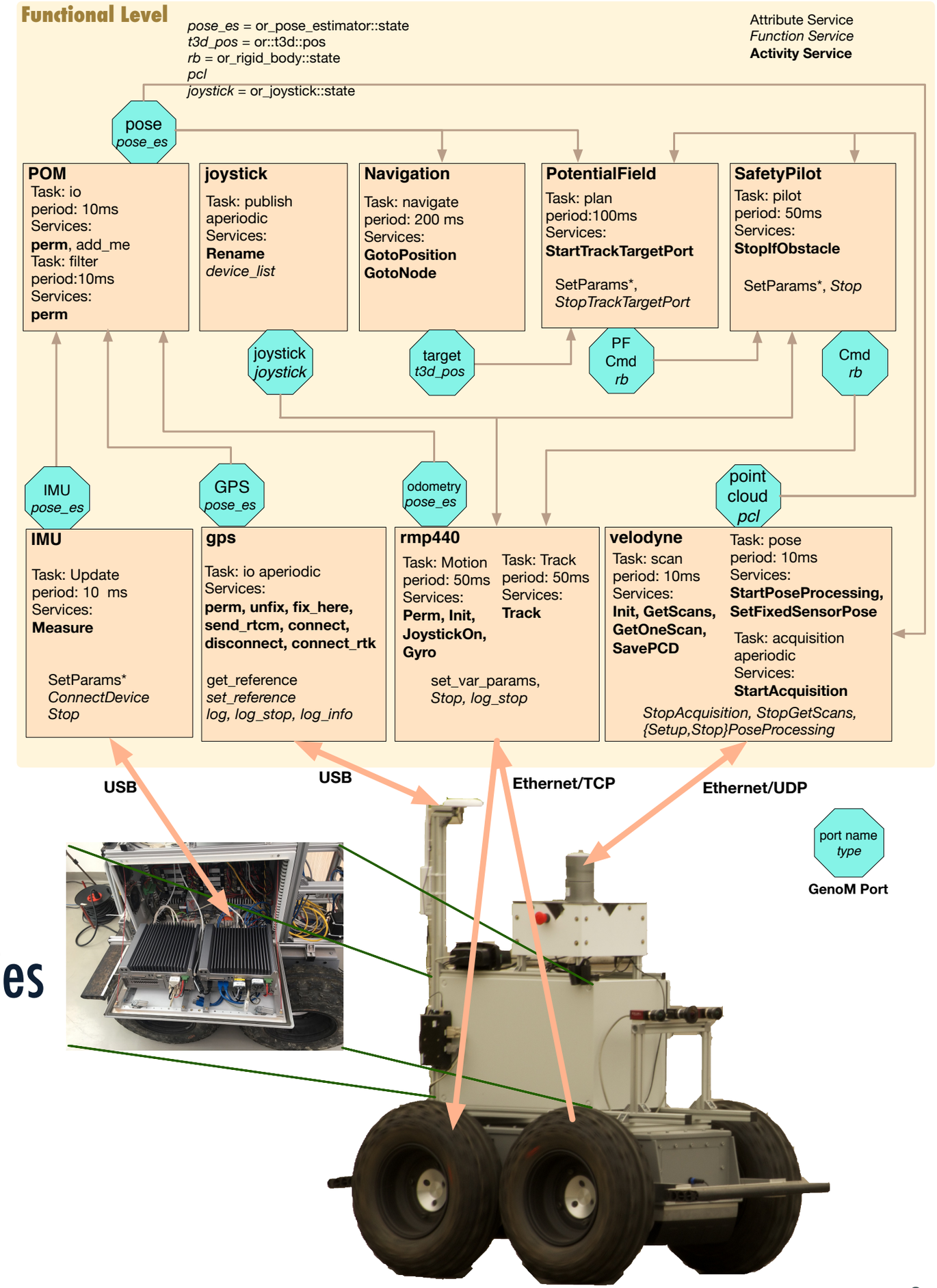
- 3 tasks
- acquisition to read packets thru UDP
- pose to store positions from POM
- scan to build PCL scans from packets and positions



# Minnie software architecture

- 9 components
- 9 ports
- (13 + 9) tasks (period)
- 38 activity services (with automata)
- 41 function services
- 43 attribute services
- 170 codels (14k loc) and their WCET
- 200k loc for all components + libraries

- Everything runs onboard...



# One day, the "formal models" Genies gave me 3 ~~wishes~~ models...

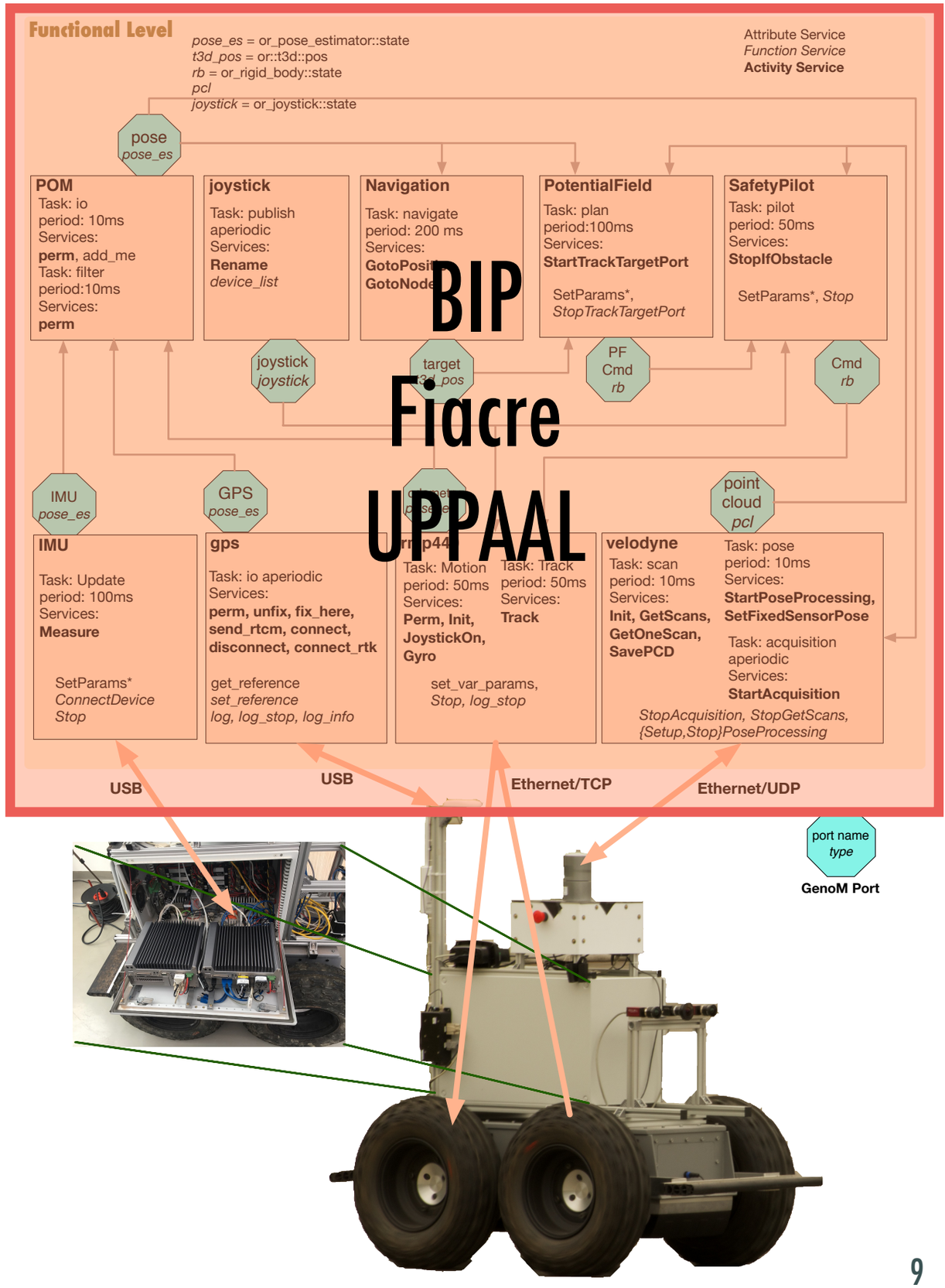
Formal Methods/Frameworks

- BIP (Verimag)
- Fiacre/TINA (LAAS/VerTICS)
- UPPAAL (UPPsala & AALborg University)

BIP

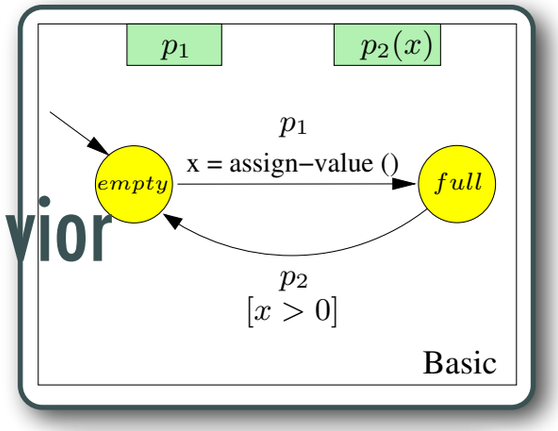
Fiacre

UPPAAL



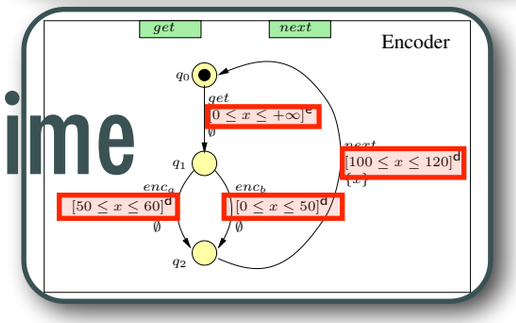
# BIP Model

## Behavior



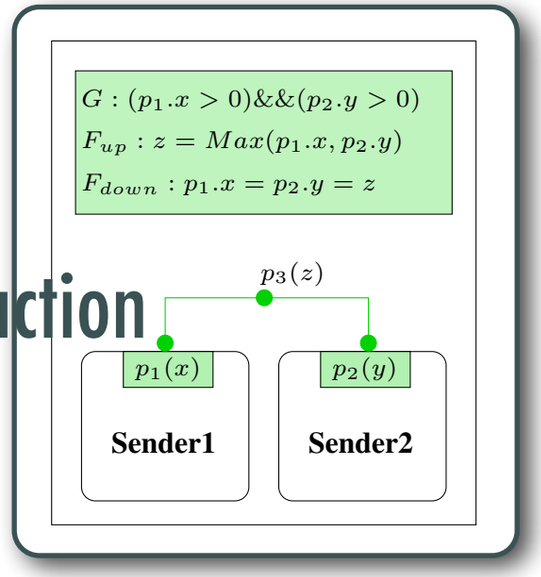

```
port type IntPort (int x)
port type ePort ()

atomic type Basic
   data int x = 0
   export port ePort p1() is p1
   export port intPort p2(x) is p2

   place empty
   place full

   initial to empty

   on p1 from empty to full
      do { x = assign-value();}
   on p2 provided [x > 0]
   from full to empty
end
```

## time



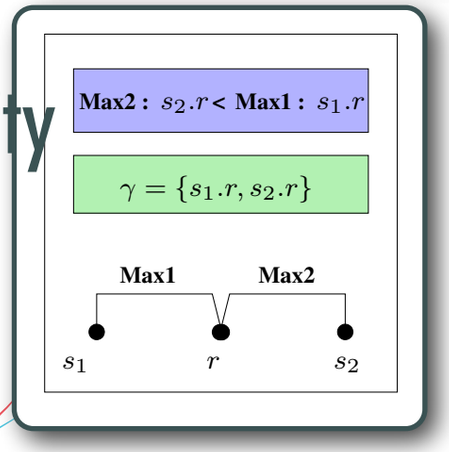

```
atomic type Encoder
   export port intPort get
   export port intPort intPort next
   port intPort enc_a compute
   port intPort enc_b

   clock x unit millisecond

   place q0
   place q1
   place q2

   initial to q0

   on get from q0 to q1
   when x in [0,-] eager
   on enc_a from q1 to q2
   when x in [50,60] delayable
   on enc_b from q1 to q2
   when x in [0,50] delayable
   on next from q2 to q0
   when x in [100,120] delayable
   reset x
end
```

## Interaction

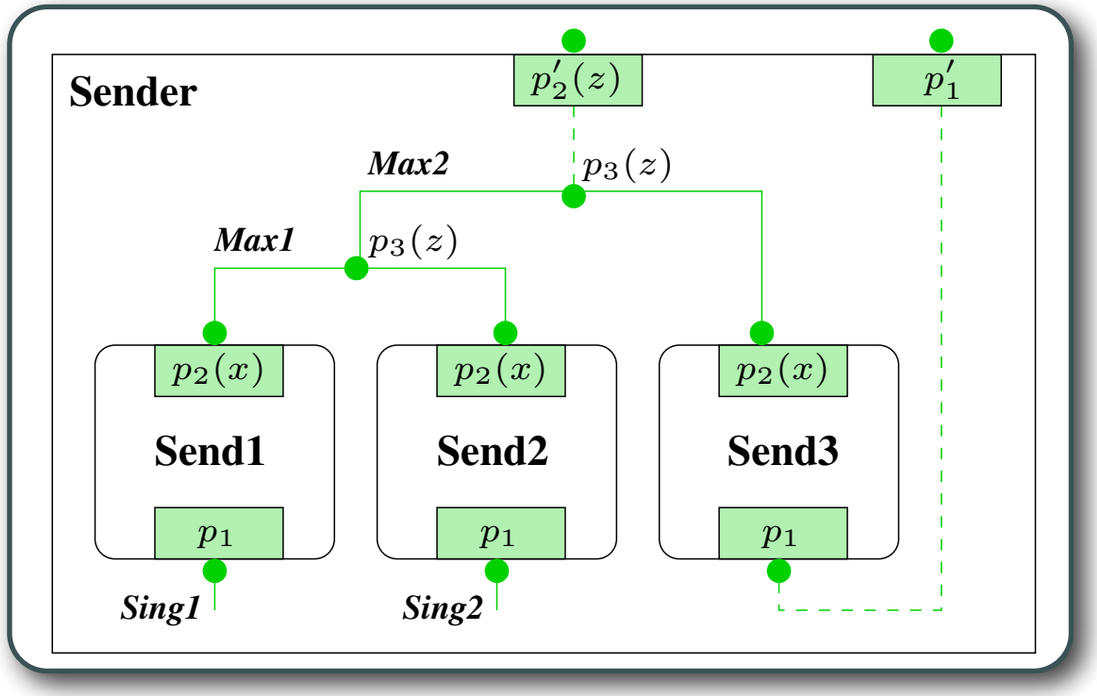



```
connector type Max (intPort p1, intPort p2)
   data int z
   define [ p1p2 ]

   on p1p2 provided (p1.x > 0) && (p2.y > 0)
      up { z = Max (p1.x , p2.y);}
      down { p1.x = p2.y = z ;}

   export port intPort p3(z)

end
```

## Priority




```
connector Max1 (s1, r)
connector Max2 (s2, r)

priority maximal if (s1.x > s2.x)
      Max2 < Max1
```




```
compound type Sender

component Basic Send1
component Basic Send2
component Basic Send3

connector Max Max1(Sender1.p2,Send2.p2)
connector Max Max2(Max1.p3,Send3.p2)
connector Singleton Sing1 (Send1.p1)
connector Singleton Sing2 (Send2.p1)

export port Intport p2' is Max2.p3
export port Intport p1' is Send3.p1

end
```

# Run Time Verification with BIP Engine

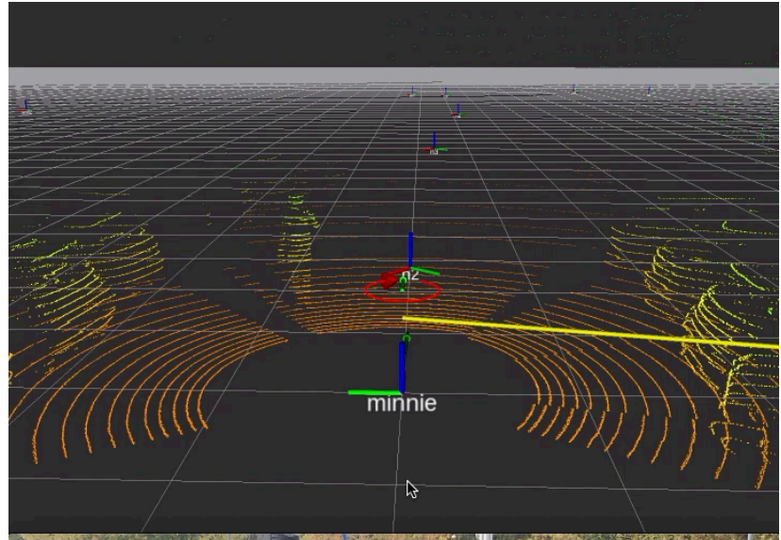
emacs@ashitaka-ubuntu.laas.fr

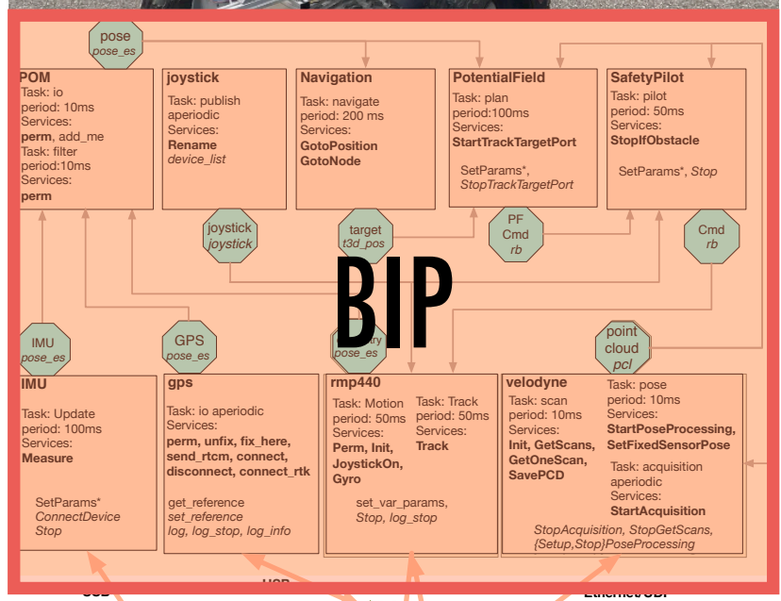
File Edit Options Buffers Tools Help

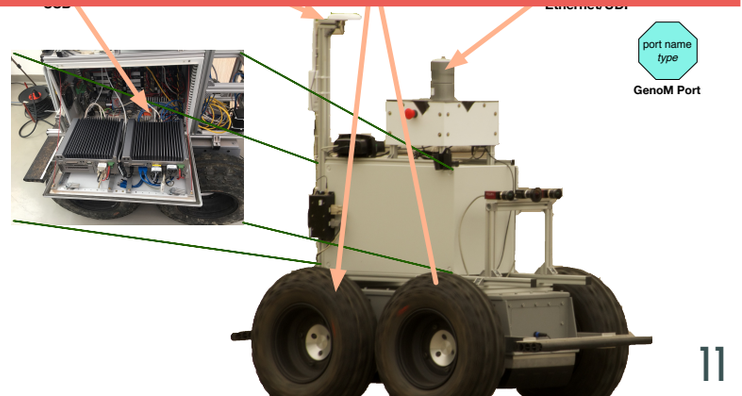
Save Undo

```
/* --- Generated by genom 2.99.33. Do not edit ------------------------- */


/* This model is automatically generated - Mohammed Foughali and Felix Ingrand 2016-2019 */

@cpp(include="gps_BIPGenoM3.hpp,IMUDriver_BIPGenoM3.hpp,joystick_BIPGenoM3.hpp,Navigation_BIPGenoM3.hpp,pom_BIPGenoM3.hpp,PotentialField_B
IPGenoM3.hpp,rmp440_BIPGenoM3.hpp,velodyne_BIPGenoM3.hpp,SafetyPilot_BIPGenoM3.hpp,stdio.h,global_functions.hpp",src="external/global_func
tions.cpp,external/gps_BIPGenoM3.cpp,external/IMUDriver_BIPGenoM3.cpp,external/joystick_BIPGenoM3.cpp,external/Navigation_BIPGenoM3.cpp,ex
ternal/pom_BIPGenoM3.cpp,external/PotentialField_BIPGenoM3.cpp,external/rmp440_BIPGenoM3.cpp,external/velodyne_BIPGenoM3.cpp,external/Safe
tyPilot_BIPGenoM3.cpp")

/* package */
package gps_IMUDriver_joystick_Navigation_pom_PotentialField_rmp440_velodyne_SafetyPilot_BIPGenoM3

/* definition of external types */
extern data type genom_activity_ptr
extern data type genom_event

extern data type genom_gps_abort_activity_activity_ptr
extern data type genom_gps_connect_port_activity_ptr
extern data type genom_gps_connect_service_activity_ptr
extern data type genom_gps_kill_activity_ptr
extern data type genom_gps_get_param_activity_ptr
extern data type genom_gps_get_last_fix_activity_ptr
extern data type genom_gps_connect_activity_ptr
extern data type genom_gps_disconnect_activity_ptr
extern data type genom_gps_set_rtk_port_activity_ptr
extern data type genom_gps_connect_rtk_activity_ptr
extern data type genom_gps_set_reference_activity_ptr
extern data type genom_gps_get_reference_activity_ptr
extern data type genom_gps_unfix_activity_ptr
extern data type genom_gps_fix_activity_ptr
extern data type genom_gps_fix_here_activity_ptr
extern data type genom_gps_send_rtcm_activity_ptr
extern data type genom_gps_rtk_stats_activity_ptr
extern data type genom_gps_log_activity_ptr
extern data type genom_gps_log_stop_activity_ptr
extern data type genom_gps_log_info_activity_ptr
extern data type genom_IMUDriver_abort_activity_activity_ptr
extern data type genom_IMUDriver_connect_port_activity_ptr
extern data type genom_IMUDriver_connect_service_activity_ptr
extern data type genom_IMUDriver_kill_activity_ptr
extern data type genom_IMUDriver_SetDevice_activity_ptr
extern data type genom_IMUDriver_SetPublishOrientation_activity_ptr
extern data type genom_IMUDriver_SetBaudrate_activity_ptr
extern data type genom_IMUDriver_SetVerbose_activity_ptr
extern data type genom_IMUDriver_Stop_activity_ptr
extern data type genom_IMUDriver_ConnectDevice_activity_ptr
extern data type genom_IMUDriver_GetHeading_activity_ptr
extern data type genom_IMUDriver_Measure_activity_ptr
extern data type genom_joystick_abort_activity_activity_ptr
extern data type genom_joystick_connect_port_activity_ptr
-:---  gps_IMUDriver_joystick_Navigation_pom_PotentialField_rmp440_velodyne_SafetyPilot_BIPGenoM3.bip   Top L22   Git:master  (BIP)
```

Complete BIP model of the 9 components (27000 loc), calling the lib codels.



# Run Time Verification with BIP Engine

```
atom type propertyXXX()

        clock c unit millisecond
        export port Port scanXXX()
        export port Port reportXXX()

        place idle, busy

        initial to idle

        on scanXXX
        from idle to busy
        do {c = 0;}

        on scanXXX
        from busy to busy
        provided (c<= 2000)
        do {printf("propertyXXX <= 2000.\n"); c = 0;}

        on reportXXX
        from busy
        provided
        do { prin                    rtyXXX > 2000.\n");}

end

connector sync4 give_
                Compo
                point
                point
                prop_

connector sync4 give_
                Compo
                point
                point
                prop_

connector sync2 emerg
connector sync2 emerg
```

Point Cloud is monitored:
2 seconds without being refreshed
=>emergency stop

odometry
*pose_es*

point cloud
*pcl*

**rmp440**

Task: Motion
period: 50ms
Services:
**Perm, Init, JoystickOn, Gyro**

Task: Track
period: 50ms
Services:
**Track**

set_var_params,
*Stop, log_stop*

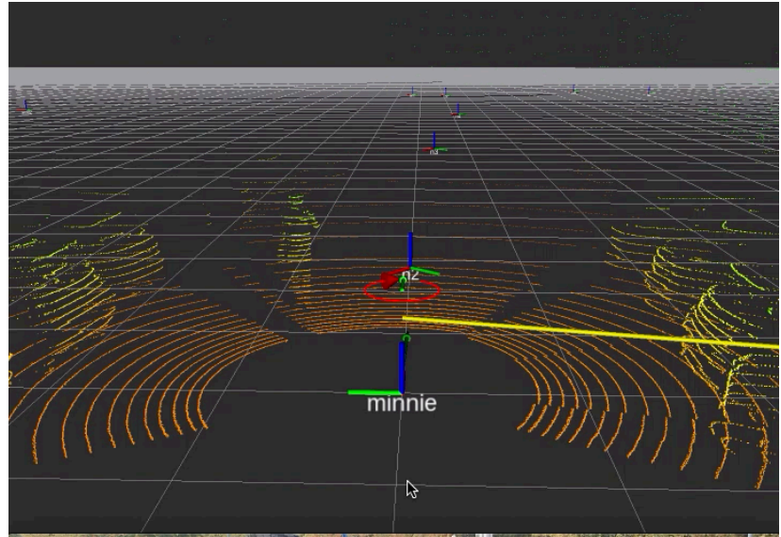
**velodyne**

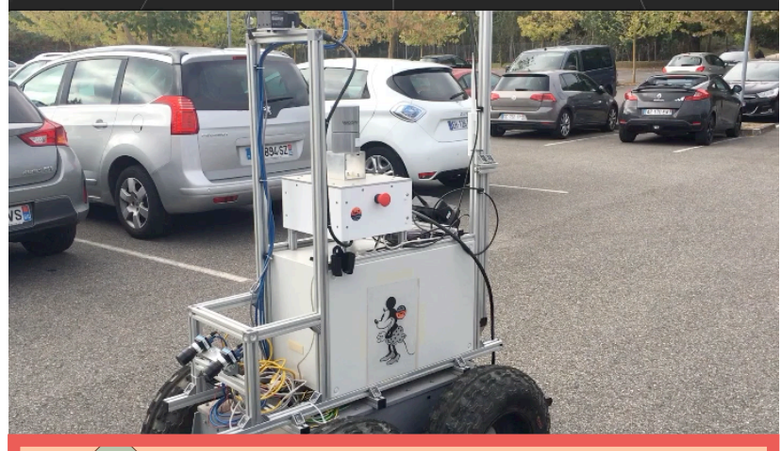
Task: scan
period: 10ms
Services:
**Init, GetScans, GetOneScan, SavePCD**

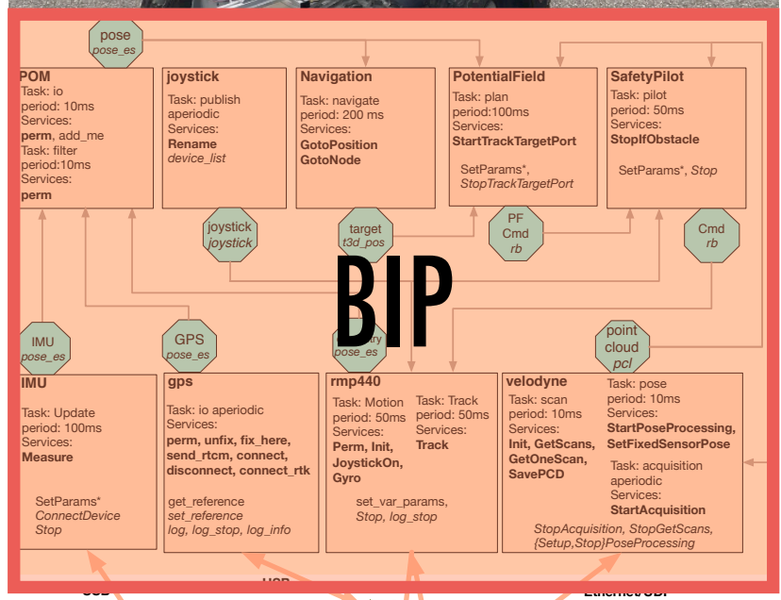
Task: pose
period: 10ms
Services:
**StartPoseProcessing, SetFixedSensorPose**

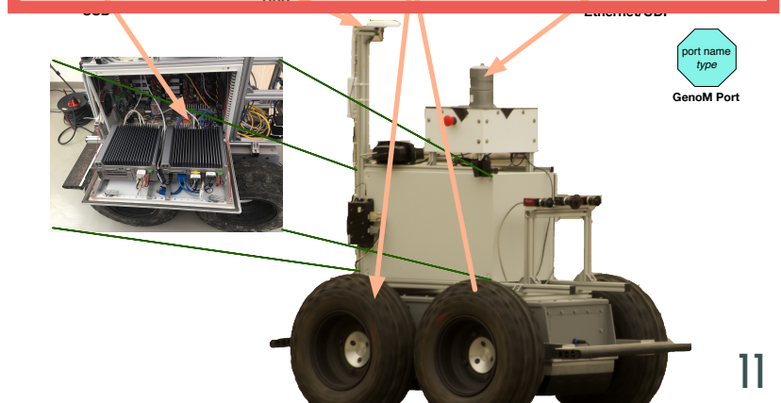
Task: acquisition
aperiodic
Services:
**StartAcquisition**

*StopAcquisition, StopGetScans, {Setup,Stop}PoseProcessing*





# Run Time Verification with BIP Engine

```
activity Track() {
  doc  "Start tracking a reference port";
  validate trackControl(in rmp);
  codel <start>trackStart(inout rs_mode,
                          out max_accel,
                          port in cmd_vel) yield track_main, end;
  codel <track_main>pumpReference(in rs_mode,
                           port in cmd_vel,
                           out ref) yield pause::track_main, end;
  codel <end,stop>stopTrack(inout rs_mode,
                            out ref) yield ether;
  task TrackTask;
  throw not_connected, port_not_found, bad_ref, cmd_stop_track,
    motors_off, emergency_stop, power_cord_connected;
  interrupts JoystickOn, Track;
};

function Stop() {
  doc "Stop current Tracking";
  interrupts JoystickOn, Track;
};
```

```
activity GetScans ( in double firstAngle = :"First angle of the scan (in degrees)",
                    in double lastAngle  = :"Last angle of the scan (in degrees)",
                    in double period     = :"Time in between scan acquisitions",
                    in double timeout    = :"Timeout used when stamping packets")
{
  doc  "Acquire full scans from the velodyne sensor periodically";
  task scan;

  validate velodyneGetScansValidate(in firstAngle, in lastAngle, in period);

  codel<start>          velodyneGetScansStart(in acquisition_params) yield copy_packets;
  codel<copy_packets>   velodyneGetOneScanCopyPackets(in acquisition_params, out mutex_buffer) yield stamp_packets;
  codel<stamp_packets>  velodyneGetOneScanStampPackets(in acquisition_params,
                                                       out mutex_pose_data, in timeout) yield pause::stamp_packets, build_scan;
  codel<build_scan>     velodyneGetOneScanBuildScan(in acquisition_params,
                                                    in firstAngle, in lastAngle) yield end;
  codel<end>            velodyneGetOneScanEnd(in acquisition_params,  in auto_save_pcd, out auto_save_pcd_count,
                                              in auto_save_pcd_prefix, port out point_cloud, port out point_cloud2, inout usec_delay)
                                              yield wait;
  codel<wait>           velodyneGetScansWait(in period) yield pause::wait, copy_packets;

  interrupts            GetOneScan, SavePCD, GetScans;

  after               Init;

  throws              e_params, e_runtime, e_interface, e_not_implemented, e_port, e_timeout;

};
```

odometry
*pose_es*

point cloud
*pcl*

**rmp440**

Task: Motion
period: 50ms
Services:
**Perm, Init, JoystickOn, Gyro**

Task: Track
period: 50ms
Services:
**Track**

set_var_params,
*Stop, log_stop*

**velodyne**

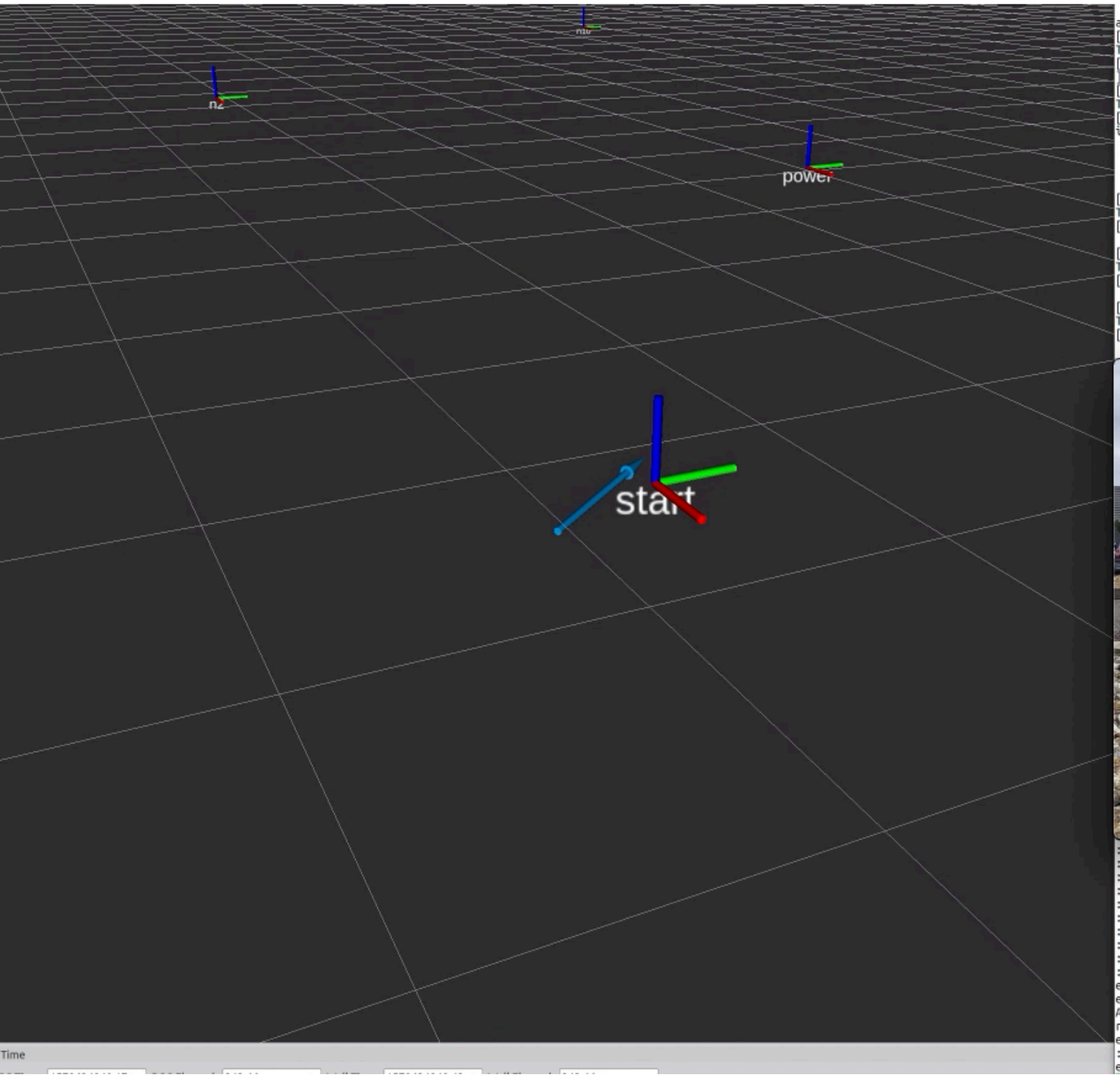
Task: scan
period: 10ms
Services:
**Init, GetScans, GetOneScan, SavePCD**

Task: pose
period: 10ms
Services:
**StartPoseProcessing, SetFixedSensorPose**

Task: acquisition
aperiodic
Services:
**StartAcquisition**

*StopAcquisition, StopGetScans, {Setup,Stop}PoseProcessing*

# Run Time Verification with BIP Engine

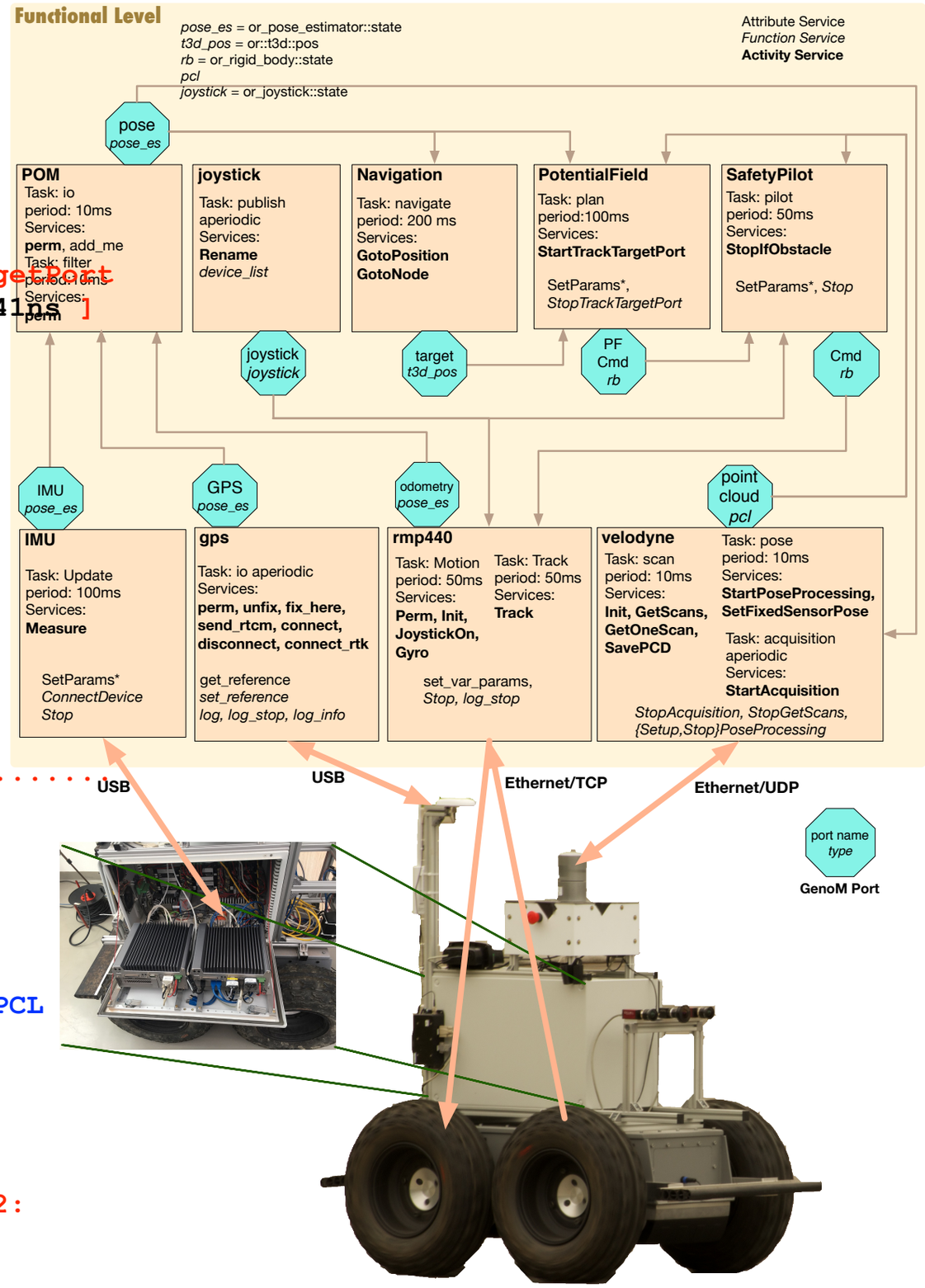



```
        [GenoM3] PotentialField Exiting PotentialField_codel_service_StartTrackTargetPort_compute_speed codel with ::Potenti
alField::write_cmd.
[BIP ENGINE]: WARNING: state #2416982 and global time 5min3s55ms751us842ns: violation of the following timing constraint ROO
T.Compound_velodyne_.velodyne_.StartPoseProcessing_velodyne_inst_1:
[BIP ENGINE]:   ROOT.Compound_velodyne_.velodyne_.StartPoseProcessing_velodyne_inst_1 resume [ -INFTY, 5min3s52ms785us699ns
]
[BIP ENGINE]: WARNING: state #2416983 and global time 5min3s55ms766us597ns: violation of the following timing constraint ROO
T.Compound_PotentialField_.PotentialField_.StartTrackTargetPort_PotentialField_inst_1:
[BIP ENGINE]:   ROOT.Compound_PotentialField_.PotentialField_.StartTrackTargetPort_PotentialField_inst_1 resume [ -INFTY, 5m
in3s53ms567us36ns ]
        [GenoM3] velodyne Calling velodyne_codel_service_StartAcquisition_main codel.
        [GenoM3] velodyne Exiting velodyne_codel_service_StartAcquisition_main codel with ::velodyne::poll.
        [GenoM3] rmp440 Calling rmp440_codel_service_Track_track_main codel.
        [GenoM3] rmp440 Exiting rmp440_codel_service_Track_track_main codel with ::rmp440::pause::track_main.
[BIP ENGINE]: WARNING: state #2416991 and global time 5min3s56ms783us917ns: violation of the following timing constraint ROO
T.Compound_rmp440_.rmp440_.Track_rmp440_inst_1:
[BIP ENGINE]:   ROOT.Compound_rmp440_.rmp440_.Track_rmp440_inst_1 resume [ -INFTY, 5min3s54ms765us930ns ]
        [GenoM3] pom Calling pom_codel_task_filter_exec  codel.
[BIP ENGINE]: WARNING: state #2417000 and global time 5min3s57ms880us160ns: violation of the following timing constraint ROO
T.Compound_pom_.pom_.perm_filter_pom:
[BIP ENGINE]:   ROOT.Compound_pom_.pom_.perm_filter_pom resume [ -INFTY, 5min3s54ms869us45ns ]
        [GenoM3] gps Calling gps_codel_task_io_publish  codel.
[BIP ENGINE]: WARNING: state #2417002 and global time 5min3s58ms185us203ns: violation of the following timing constraint ROO
T.Compound_gps_.gps_.perm_io_gps:
[BIP ENGINE]:   ROOT.Compound_gps_.gps_.perm_io_gps resume [ -INFTY, 5min3s56ms378us755ns ]
```

```
::Navigation::SetTarget               ::PotentialField::SetHitMinZ          ::SafetyPilot::SetStopLateralDistance
::Navigation::SetTargetToCurrentPose  ::PotentialField::SetMaxSpeed         ::SafetyPilot::SetVerbose
::Navigation::SetThreshold            ::PotentialField::SetNu               ::rmp440::SetRobotYaw
::Navigation::SetVerbose              ::PotentialField::SetPsi              ::tf2::SetVerbose
::PotentialField::SetAlpha            ::PotentialField::SetRepForceDistance ::velodyne::SetDelay
::PotentialField::SetDistanceToGoal   ::PotentialField::SetVerbose          ::velodyne::SetFixedSensorPose
::PotentialField::SetEpsilon          ::SafetyPilot::SetHitMinDistance      ::velodyne::SetVerbose
::PotentialField::SetHitMaxDistance   ::SafetyPilot::SetMaxSpeed            ::velodyne::SetupPoseProcessing
::PotentialField::SetHitMaxZ          ::SafetyPilot::SetMinPCLHit
::PotentialField::SetHitMinDistance   ::SafetyPilot::SetStopDistance
eltclsh > ::velodyne::SetDelay 2000000
eltclsh > navigation_track
Are you sure the robot is ready and safe to move? (Y/N): y
rmp440::3
eltclsh > Navigation::Go
::Navigation::GotoNode     ::Navigation::GotoPosition ::Navigation::GotoTarget
eltclsh > ::Navigation::GotoNode power &
```

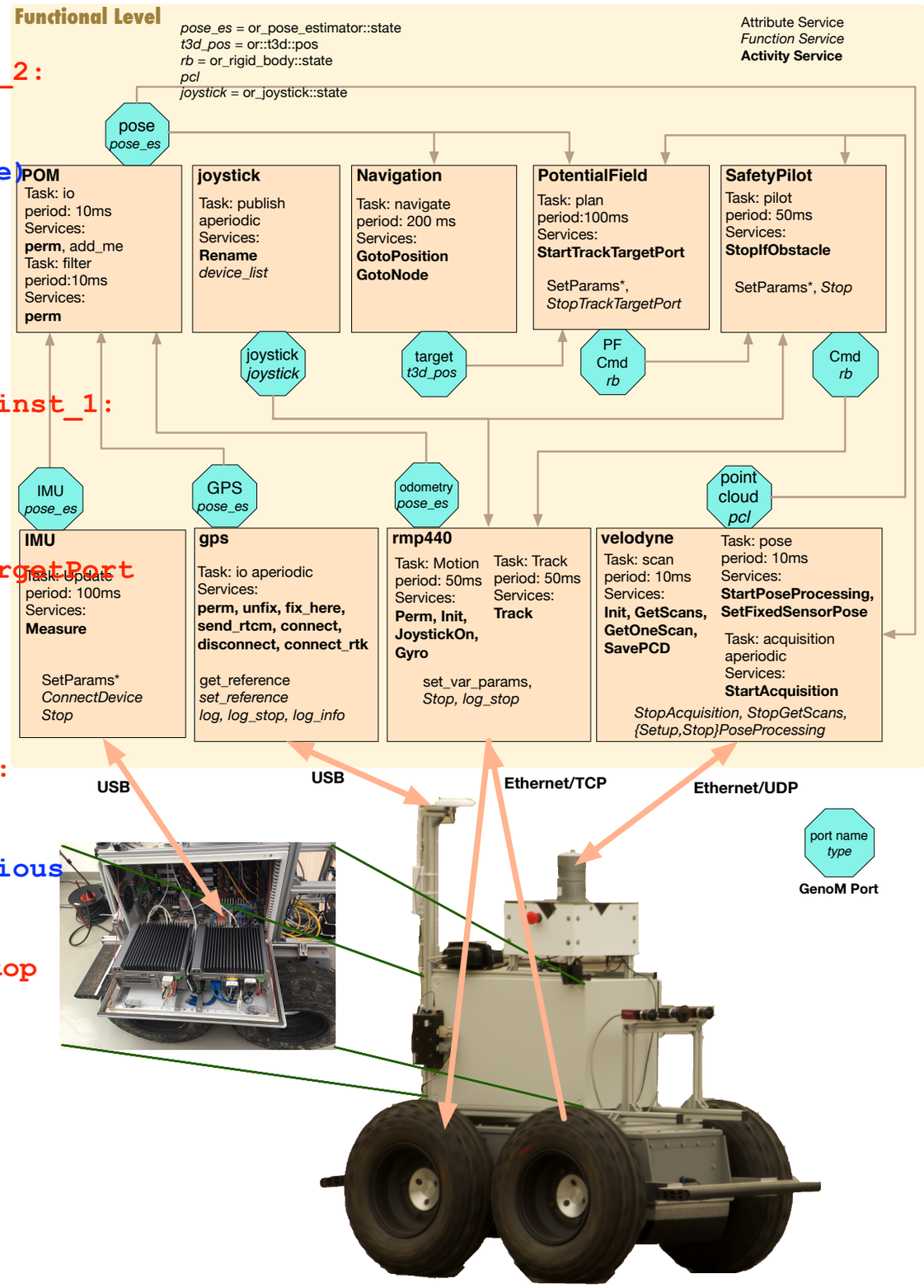
# Run Time Verification with BIP Engine

```
[BIP ENGINE]: WARNING: state #5375644 and global time
12min27s265ms544us809ns: violation of the following timing
constraint ROOT.Compound_gps_.gps_.perm_io_gps:
....
****************************
We send the SetDelay command
****************************
...
[BIP ENGINE]:
ROOT.Compound_PotentialField_.PotentialField_.StartTrackTargetPort
_PotentialField_inst_1 resume [ -INFTY, 12min27s264ms952us841ns ]
...
[BIP ENGINE]: WARNING: state #5375932 and global time
12min27s309ms870us794ns: violation of the following timing
constraint
ROOT.Compound_velodyne_.velodyne_.GetScans_velodyne_inst_2:
...
****************************
The SetDelay request has been executed.
****************************
...
[BIP ENGINE]: WARNING: state #5375959 and global time
12min27s314ms182us292ns: violation of the following timing
constraint ROOT.Compound_gps_.gps_.perm_io_gps:
.....................................................................
propertyXXX <= 2000.
[BIP ENGINE]: WARNING: state #5376193 and global time
12min27s347ms697us691ns: violation of the following timing
constraint
Got a PointCloud of 19968 width, 1 height, size 19968.
****************************
The codel (which writes the PCL port) is being delayed, so PCL
above is not published.
****************************
Starting 2000000 delay.
[BIP ENGINE]: WARNING: state #5376210 and global time
12min27s350ms238us328ns: violation of the following timing
constraint ROOT.Compound_rmp440_.rmp440_.Track_rmp440_inst_2:
```



# Run Time Verification with BIP Engine

```
[BIP ENGINE]: WARNING: state #5388409 and global time
12min29s343ms736us482ns: violation of the following timing
constraint ROOT.Compound_rmp440_.rmp440_.Track_rmp440_inst_2:
propertyXXX > 2000.
****************************
The propery is being triggered in the BIP engine (see above)
exactly when the 2 seconds have elapsed (see below).
****************************
Ending 2000000 delay.
[BIP ENGINE]: WARNING: state #5388434 and global time
12min29s348ms75us105ns: violation of the following timing
constraint
ROOT.Compound_Navigation_.Navigation_.GotoNode_Navigation_inst_1:
......
[BIP ENGINE]: WARNING: state #5388466 and global time
12min29s352ms728us417ns: violation of the following timing
constraint
ROOT.Compound_PotentialField_.PotentialField_.StartTrackTargetPort
_PotentialField_inst_1:
......
[BIP ENGINE]: WARNING: state #5388486 and global time
12min29s355ms544us686ns: violation of the following timing
constraint
ROOT.Compound_velodyne_.velodyne_.GetScans_velodyne_inst_2:

****************************
next rmp440 step is to stop the robot (12ms after the previous
loop and 7ms after monitor triggered).
****************************
     [GenoM3] rmp440 Calling rmp440_codel_service_Track_stop
codel.
rmp440 StopTrack
```



# UPPAAL

File Edit View Tools Options Help

Editor | Simulator | ConcreteSimulator | Verifier

**Enabled Transitions**

exe: Perm_act_io_pom → urg
exe: Perm_act_plan_PotentialField → urg

Reset | Next

**Simulation Trace**

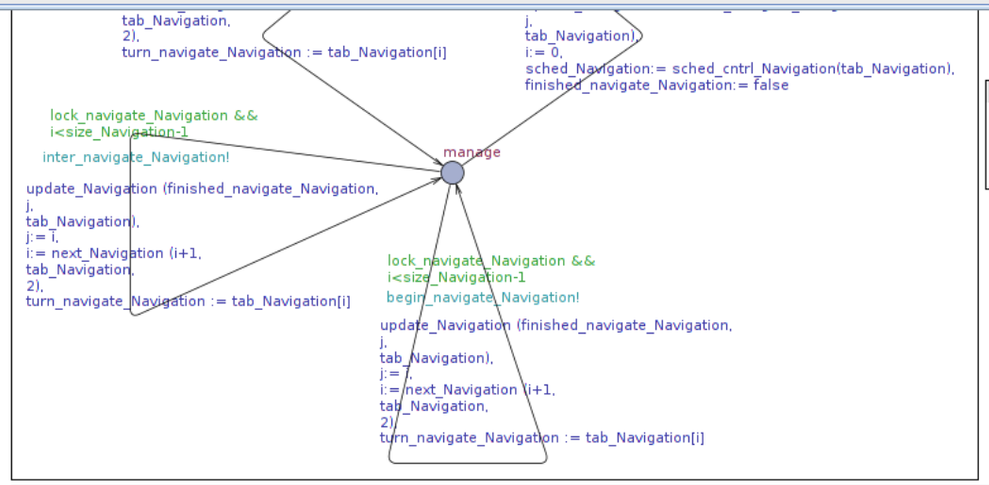
(wait, ini, unspawned, start, start, start_, ethe

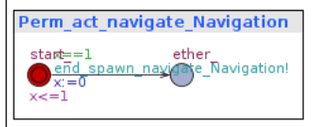
Trace File:

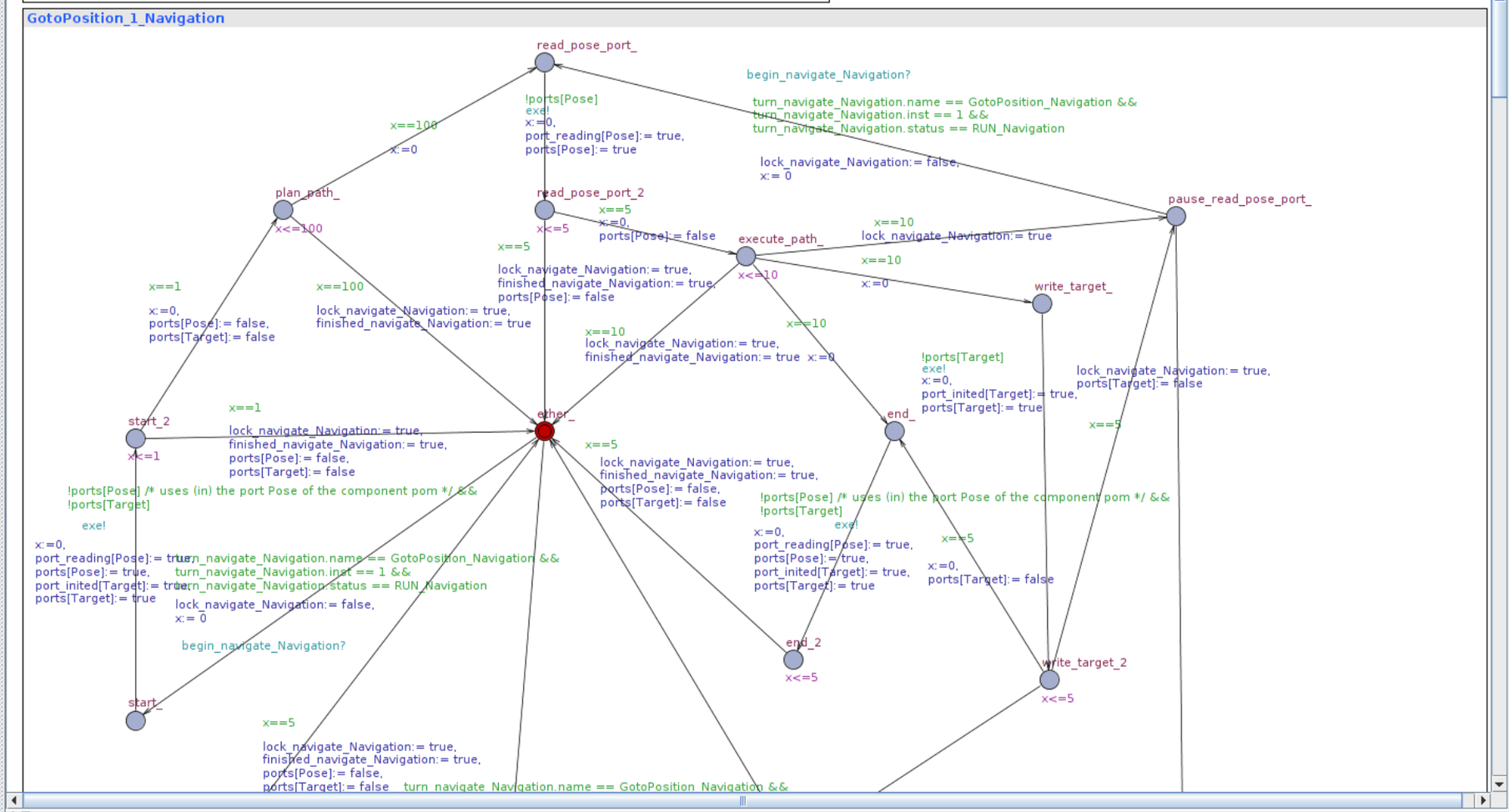
Prev | Next | Replay | Open | Save | Random

Slow — Fast

<Global variables>
Man_navigate_Navigation
Man_push_Localization
Perm_act_io_pom
Perm_act_filter_pom
Man_plan_PotentialField
Man_pilot_SafetyPilot
<Constraints>
cl.x = 0
CT_Navigation.x = 0
timer_navigate_Navigation.x = 0
Perm_act_navigate_Navigation.x = 0
GotoPosition_1_Navigation.x = 0
GotoPosition_2_Navigation.x = 0
CT_Localization.x = 0
timer_push_Localization.x = 0
Perm_act_push_Localization.x = 0
PushPoses_1_Localization.x = 0
PushPoses_2_Localization.x = 0
CT_pom.x = 0
timer_io_pom.x = 0
Perm_act_io_pom.x = 0
timer_filter_pom.x = 0
Perm_act_filter_pom.x = 0
CT_PotentialField.x = 0
timer_plan_PotentialField.x = 0
Perm_act_plan_PotentialField.x = 0
StartTrackTargetPort_1_PotentialFiel
StartTrackTargetPort_2_PotentialFiel
CT_SafetyPilot.x = 0
timer_pilot_SafetyPilot.x = 0
Perm_act_pilot_SafetyPilot.x = 0
SpeedMergeAndAvoidObstacle_1_Saf
SpeedMergeAndAvoidObstacle_2_Saf
cl.x = CT_Navigation.x
CT_Navigation.x = timer_navigate_Na
timer_navigate_Navigation.x = Perm_
Perm_act_navigate_Navigation.x = Go
GotoPosition_1_Navigation.x = GotoP
GotoPosition_2_Navigation.x = CT_Loc
CT_Localization.x = timer_push_Localiz
timer_push_Localization.x = Perm_act
Perm_act_push_Localization.x = Push
PushPoses_1_Localization.x = PushPo
PushPoses_2_Localization.x = CT_pon
CT_pom.x = timer_io_pom.x
timer_io_pom.x = Perm_act_io_pom.x
Perm_act_io_pom.x = timer_filter_pom
timer_filter_pom.x = Perm_act_filter_p
Perm_act_filter_pom.x = CT_Potentiall
CT_PotentialField.x = timer_plan_Pote
timer_plan_PotentialField.x = Perm_a
Perm_act_plan_PotentialField.x = Sta
StartTrackTargetPort_1_PotentialFiel
StartTrackTargetPort_2_PotentialFiel
CT_SafetyPilot.x = timer_pilot_SafetyP
timer_pilot_SafetyPilot.x = Perm_act_
Perm_act_pilot_SafetyPilot.x = Speed
SpeedMergeAndAvoidObstacle_1_Saf




urg | cl | CT_Navigation | timer_navigate_Navigation | Man_navigate_Navigation | Perm_act_navigate_Navigation | GotoPosition_1_Navigation | GotoPosition_2_Navigation

# UPPAAL

## the UPPAAL model (9000 loc) of all the components

emacs@ashitaka-ubuntu.laas.fr

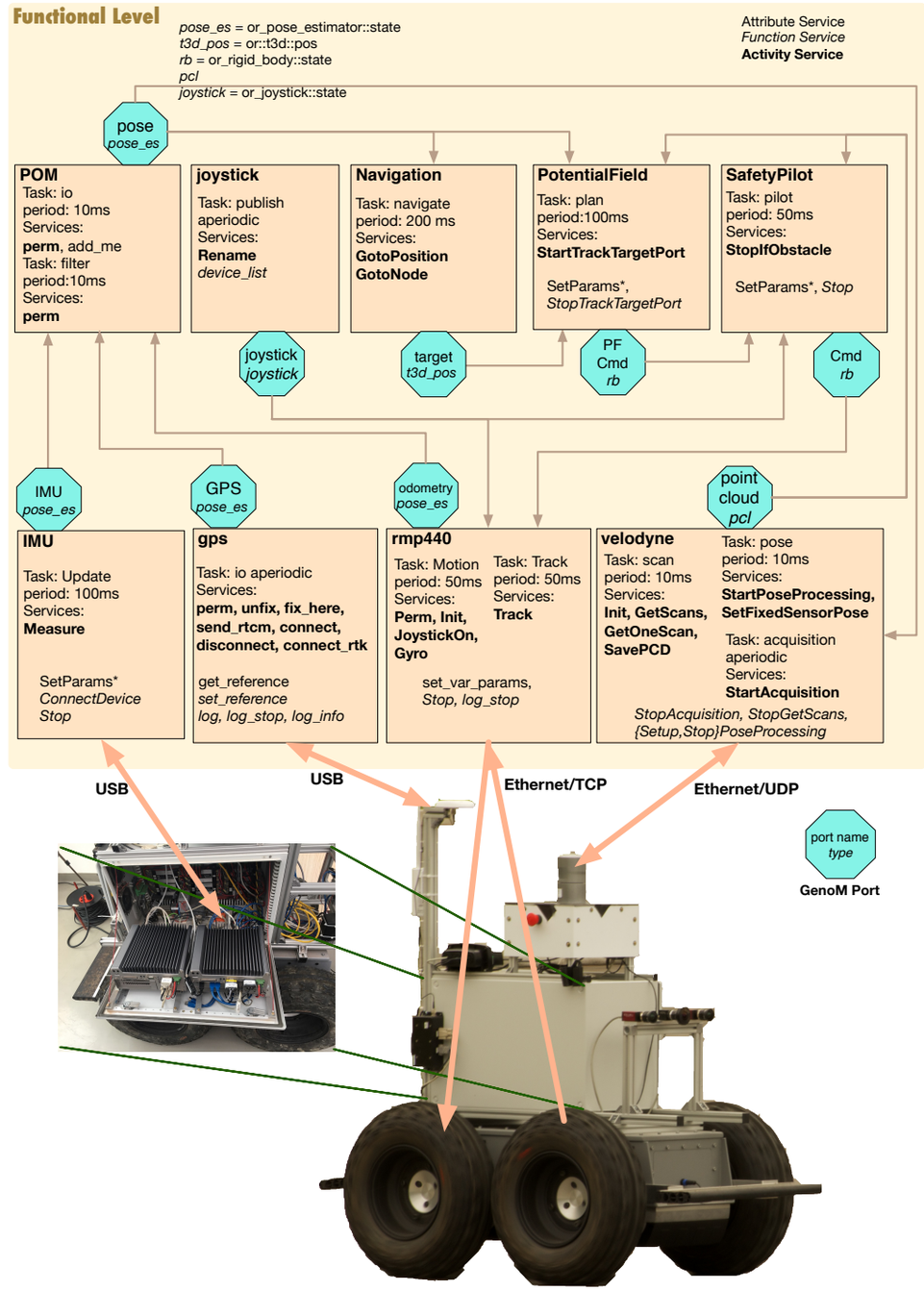
File Edit Options Buffers Tools C Help

Save Undo

```
/* --- Generated by genom 2.99.33. Do not edit ------------------------- */

/* This UPPAAL sepcification is automatically generated. */



/* ports */

const int state := 0;
const int info := 1;
const int ImuPose := 2;
const int device := 3;
const int Target := 4;
const int frame := 5;
const int PFCmd := 6;
const int PF_Att := 7;
const int PF_Rep := 8;
const int PF_Res := 9;
const int Pose := 10;
const int Status := 11;
const int StatusGeneric := 12;
const int point_cloud := 13;
const int point_cloud2 := 14;
const int Cmd := 15;
const int nports:= 16;
bool ports[nports]:= {false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false};


/* urgency */

urgent chan exe;

/* channels, variables and functions (gps) */
/* channels */

urgent chan recv_urg_gps,
       shuttimer_io_gps,
       begin_io_gps, inter_io_gps;
chan recv_gps;

broadcast chan end_spawn_io_gps;

urgent broadcast chan go_io_gps;

/* constants & global variables */
bool shut_gps:= false;

bool sched_gps:= false;
bool lock_io_gps:= true;
```

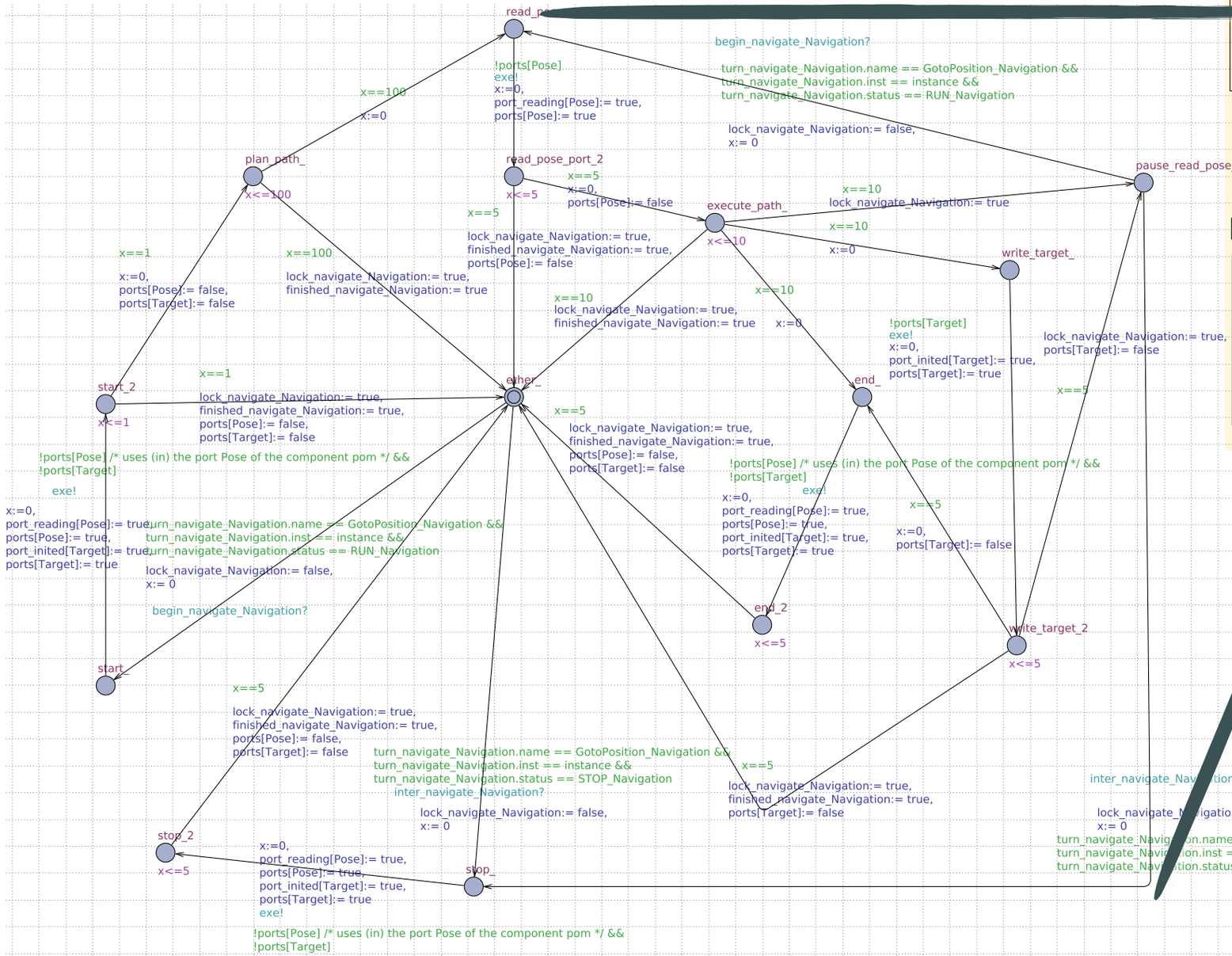
-:---  gps-IMUDriver-joystick-Navigation-pom-PotentialField-rmp440-velodyne-SafetyPilot.xta  Top L1  Git-master (C/l Abbrev)

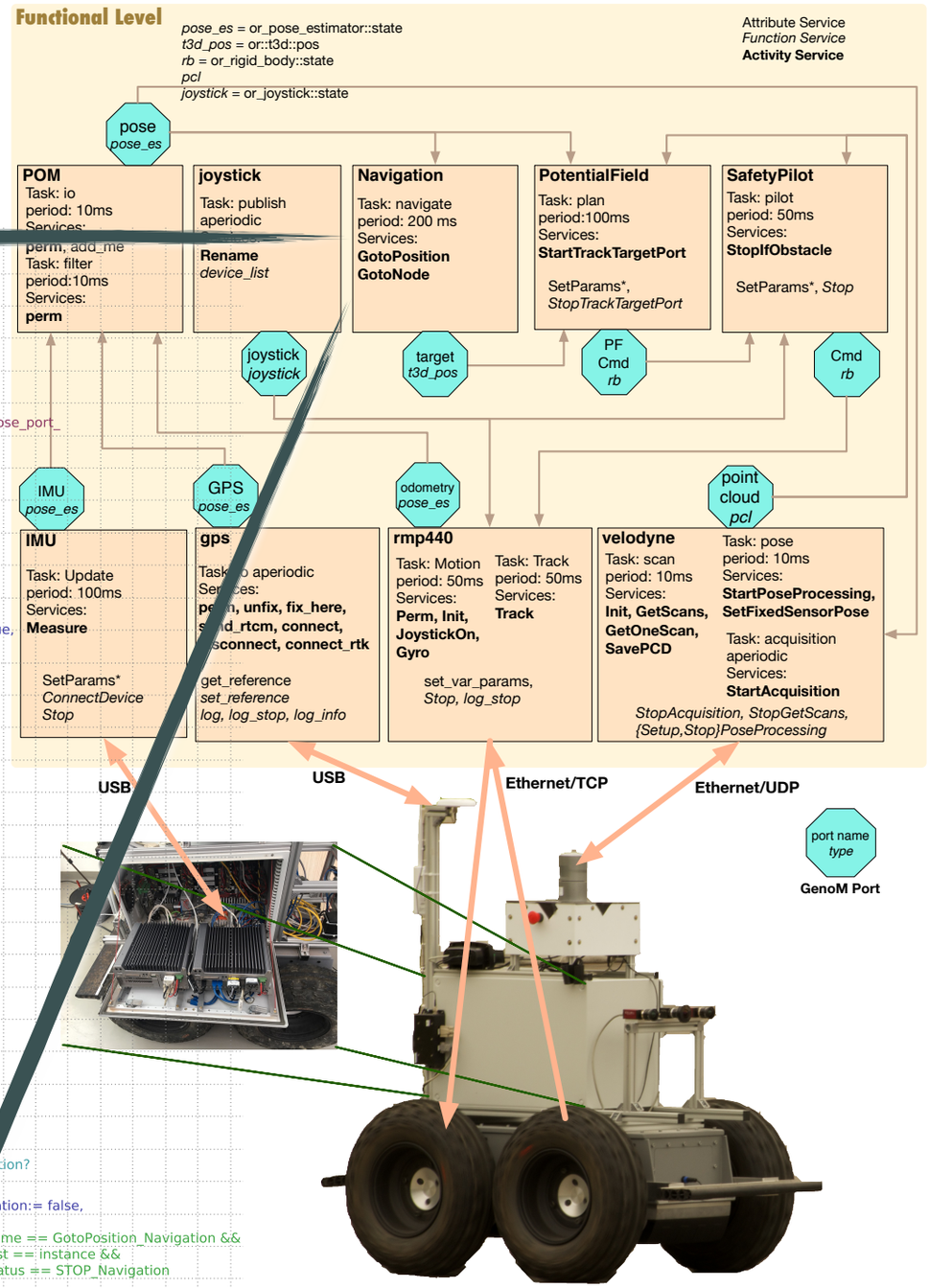
Mark set



# UPPAAL

## the UPPAAL model (9000 loc) of all the components

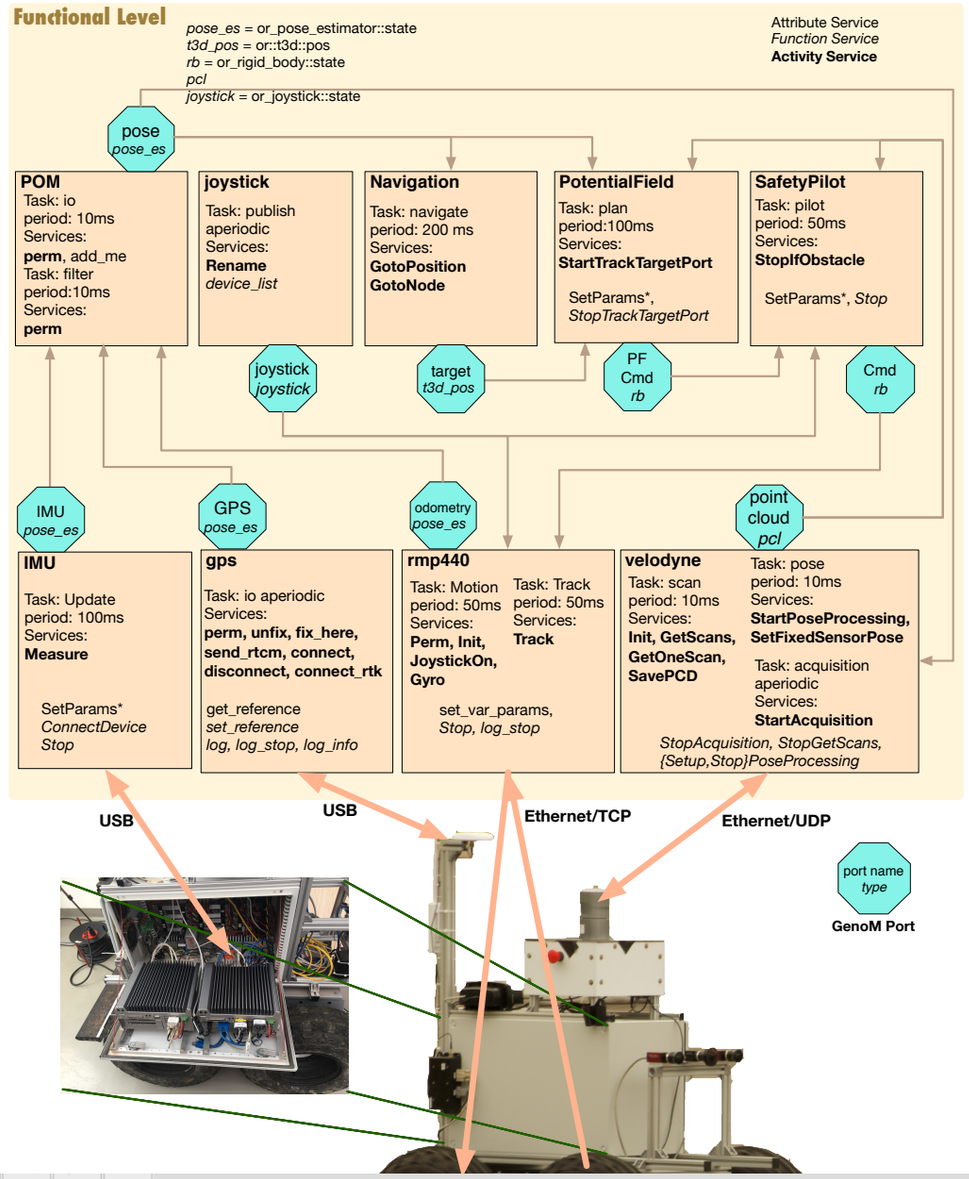



# Model Checking

# Verification with UPPAAL

```
/* All ports are properly written before being read
*/
A[] (ports_read[p] imply ports_write[p])


/* Navigation: bound between stop request and writing
a new target (to current pose) */
/* take advantage of the CT_Navigation clock reset
only once (when sending the stop request)
to verify a bounded-response property without
additional processes*/
/* bound = 202.5 ms, verification time 442.256s,
memory consumption ~1gb */
CT_Navigation.Stop_ -->
(GotoPosition_1_Navigation.ether_ and
CT_Navigation.x<=2025)

/* absence of (service) deadlock */
/* verification time 40 to 60s each, memory
consumption ~250mb */
Man_navigate_Navigation.manage -->
Man_navigate_Navigation.start
Man_io_pom.manage --> Man_io_pom.start
Man_filter_pom.manage --> Man_filter_pom.start
Man_push_Localization.manage -->
Man_push_Localization.start
Man_plan_PotentialField.manage -->
Man_plan_PotentialField.start
Man_pilot_SafetyPilot.manage -->
Man_pilot_SafetyPilot.start
```






Editor | Simulator | ConcreteSimulator | Verifier

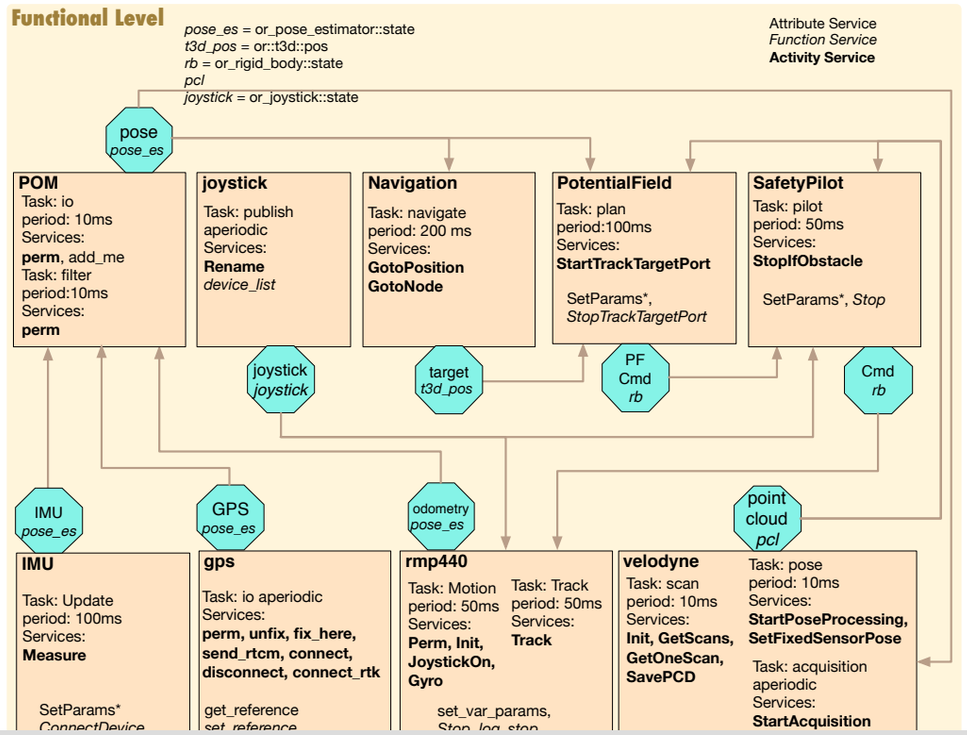
Overview

```
/* Query file for properties verifiable without the need of an observer */
/* Navigation: bound between stop request and writing a new target (to current pose) */
/* take advantage of the CT_Navigation clock reset only once (when sending the stop request)
to verify a bounded-response property without additional processes*/
/* bound = 202.5 ms, verification time 442.256s, memory consumption ~1gb */
CT_Navigation.Stop_ --> (GotoPosition_1_Navigation.ether_ and CT_Navigation.x<=2025)
/* absence of (service) deadlock */
/* verification time 40 to 60s each, memory consumption ~250mb */
Man_navigate_Navigation.manage --> Man_navigate_Navigation.start
Man_io_pom.manage --> Man_io_pom.start
Man_filter_pom.manage --> Man_filter_pom.start
Man_push_Localization.manage --> Man_push_Localization.start
Man_plan_PotentialField.manage --> Man_plan_PotentialField.start
A[] (not port_reading[Pose] or port_inited[Pose])
A[] (and (not port_reading[Pose] or port_inited[Pose]) and (not port_reading[PFCmd] or port_inited[PFCmd]) and (not port_reading[Cmd] or port_inited[Cmd]) )
Man_pilot_SafetyPilot.manage --> Man_pilot_SafetyPilot.start
```

LAAS CNRS

# Verification with UPPAAL

```
/* All ports are properly written before being read
*/
A[] (ports_read[p] imply ports_write[p])


/* Navigation: bound between stop request and writing
a new target (to current pose) */
/* take advantage of the CT_Navigation clock reset
only once (when sending the stop request)
to verify a bounded-response property without
additional processes*/
/* bound = 202.5 ms, verification time 442.256s,
memory consumption ~1gb */
CT_Navigation.Stop_ -->
(GotoPosition_1_Navigation.ether_ and
CT_Navigation.x<=2025)
```




Editor | Simulator | ConcreteSimulator | Verifier

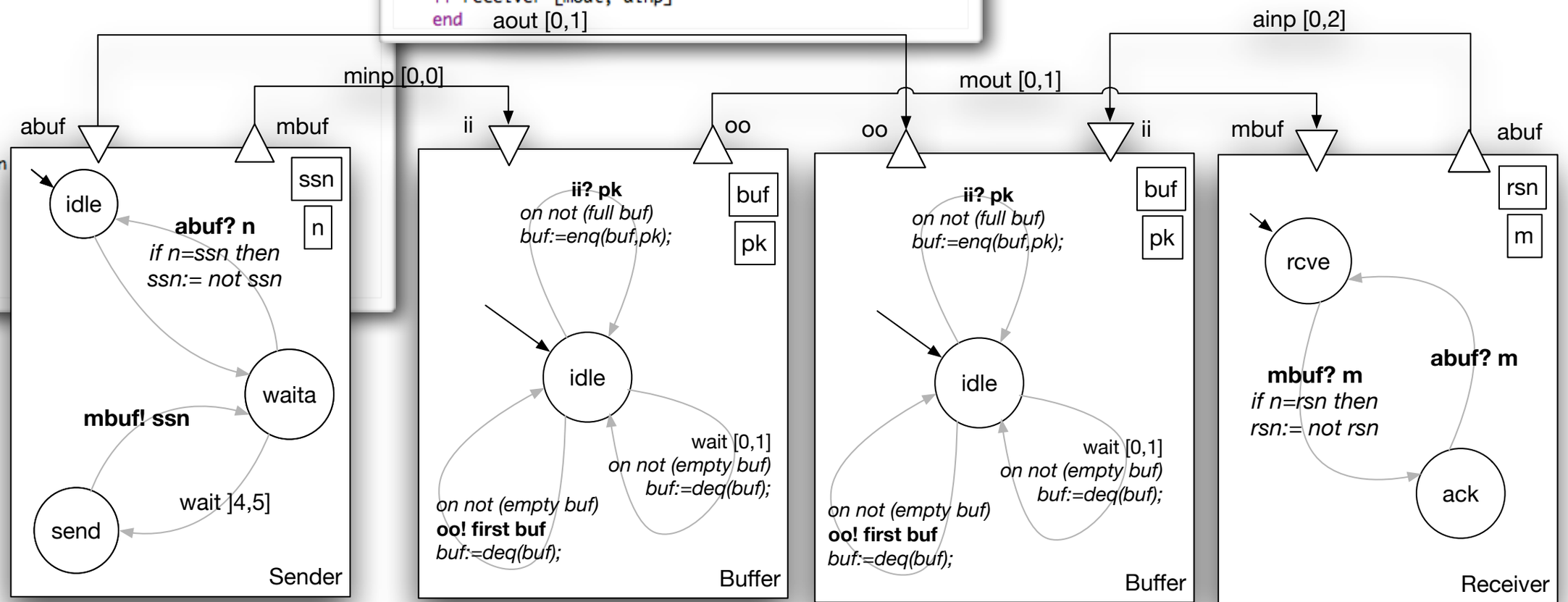
**Overview**

```
/* Query file for properties verifiable without the need of an observer */
/* Navigation: bound between stop request and writing a new target (to current pose) */
/* take advantage of the CT_Navigation clock reset only once (when sending the stop request)
to verify a bounded-response property without additional processes*/
/* bound = 202.5 ms, verification time 442.256s, memory consumption ~1gb */
CT_Navigation.Stop_ --> (GotoPosition_1_Navigation.ether_ and CT_Navigation.x<=2025)
/* absence of (service) deadlock */
/* verification time 40 to 60s each, memory consumption ~250mb */
Man_navigate_Navigation.manage --> Man_navigate_Navigation.start
Man_io_pom.manage --> Man_io_pom.start
Man_filter_pom.manage --> Man_filter_pom.start
Man_push_Localization.manage --> Man_push_Localization.start
Man_plan_PotentialField.manage --> Man_plan_PotentialField.start
A[] (not port_reading[Pose] or port_inited[Pose])
A[] (and (not port_reading[Pose] or port_inited[Pose]) and (not port_reading[PFCmd] or port_inited[PFCmd]) and (not port_reading[Cmd] or port_inited[Cmd]) )
Man_pilot_SafetyPilot.manage --> Man_pilot_SafetyPilot.start
```

# Fiacre Model example: Alternate Bit Protocol

```
/* Processes */

process buffer [ii: in packet, oo: out packet] is
    states idle
    var buff : queue 1 of packet := {||},
        pkt: packet
    from idle
    select
       /* getting new packet */
       ii?pkt;
       on not (full buff);  // should be redundant but prevents
                            // queue exception if time-out too small
       buff := enqueue (buff,pkt);
       to idle
    [] /* putting first packet */
       on not (empty buff);
       oo!first buff;
       buff := dequeue buff;
       to idle
    [] /* losing a packet */
       wait [0,1];
       on not (empty buff);
       buff := dequeue buff;
       #lost;
       to idle
    end

process sender [mbuff: out packet, abuff: in packet] is
    states idle, send, waita
    var ssn, n: seqno := false  // ssn is current sequence number
    from idle
        /* should also retrieve data from user */
        to waita
    from send
        mbuff! ssn;
        to waita
    from waita
      select
         abuff? n;
         if n = ssn
         then ssn := not ssn
         end;
         to idle
        [] wait ]4,5];
         /* resend */
         to send
      end
```

```
process receiver [mbuff: in packet, abuff: out packet] is
    states rcve, ack
   var rsn: seqno := false, m: packet := true
      // rsn is expected sequence number
    from rcve
        mbuff? m;
        if m = rsn then
           /* also should deliver data to user */
           rsn := not rsn;
           to ack
        else
           // reject duplicate
           to ack
        end
    from ack
        abuff! m;
        to rcve


/* Main component */

component abp is
    port minp : packet in [0,0],
         mout : packet in [0,1],
         ainp : packet in [0,2],
         aout : packet in [0,1]
    par * in
       sender [minp, aout]
    || buffer [minp, mout]
    || buffer [ainp, aout]
    || receiver [mout, ainp]
    end
```



# Fiacre

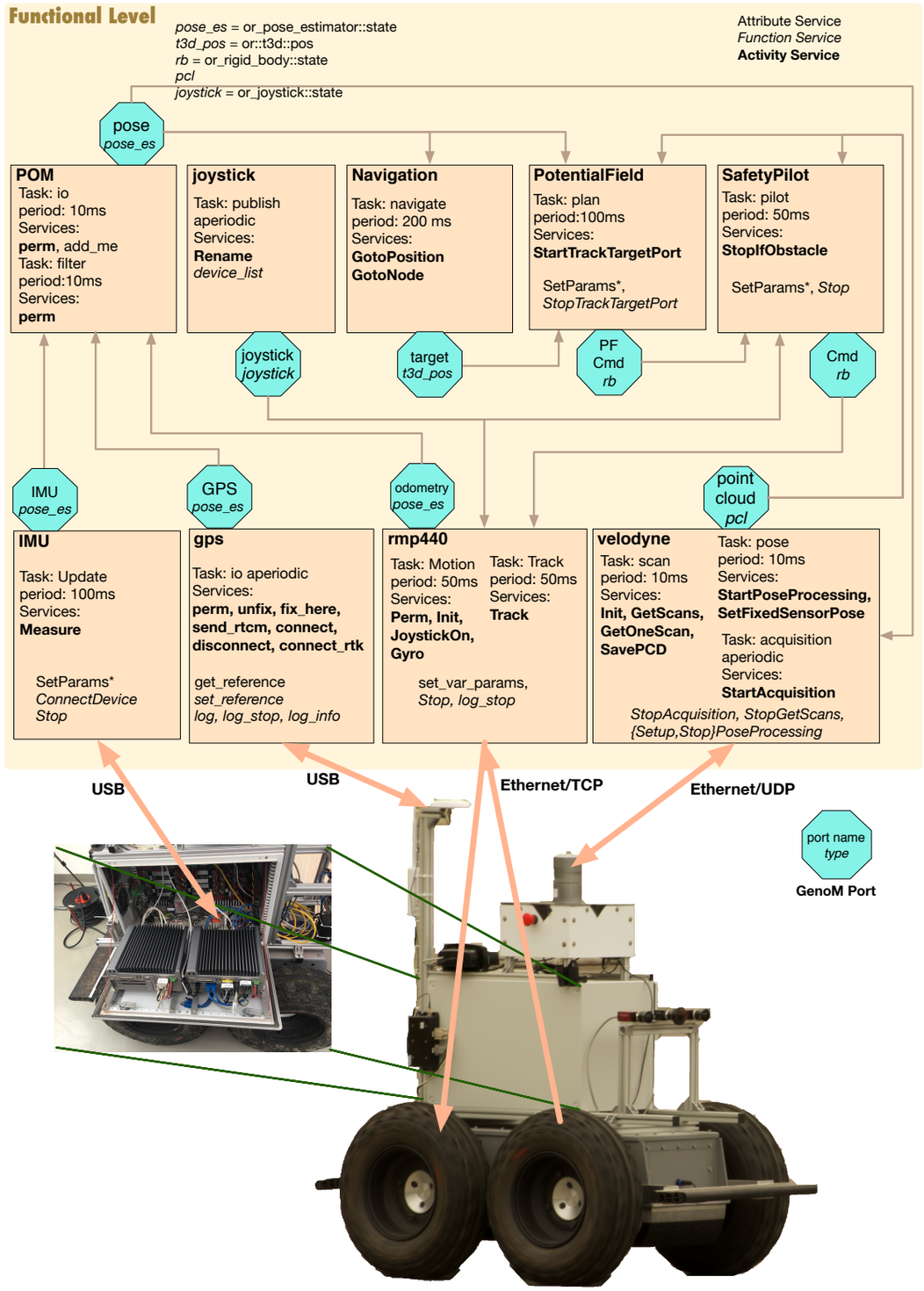
## Fiacre model (9500 loc) of all the components

emacs@ashitaka-ubuntu.laas.fr

File Edit Options Buffers Tools Fiacre Help

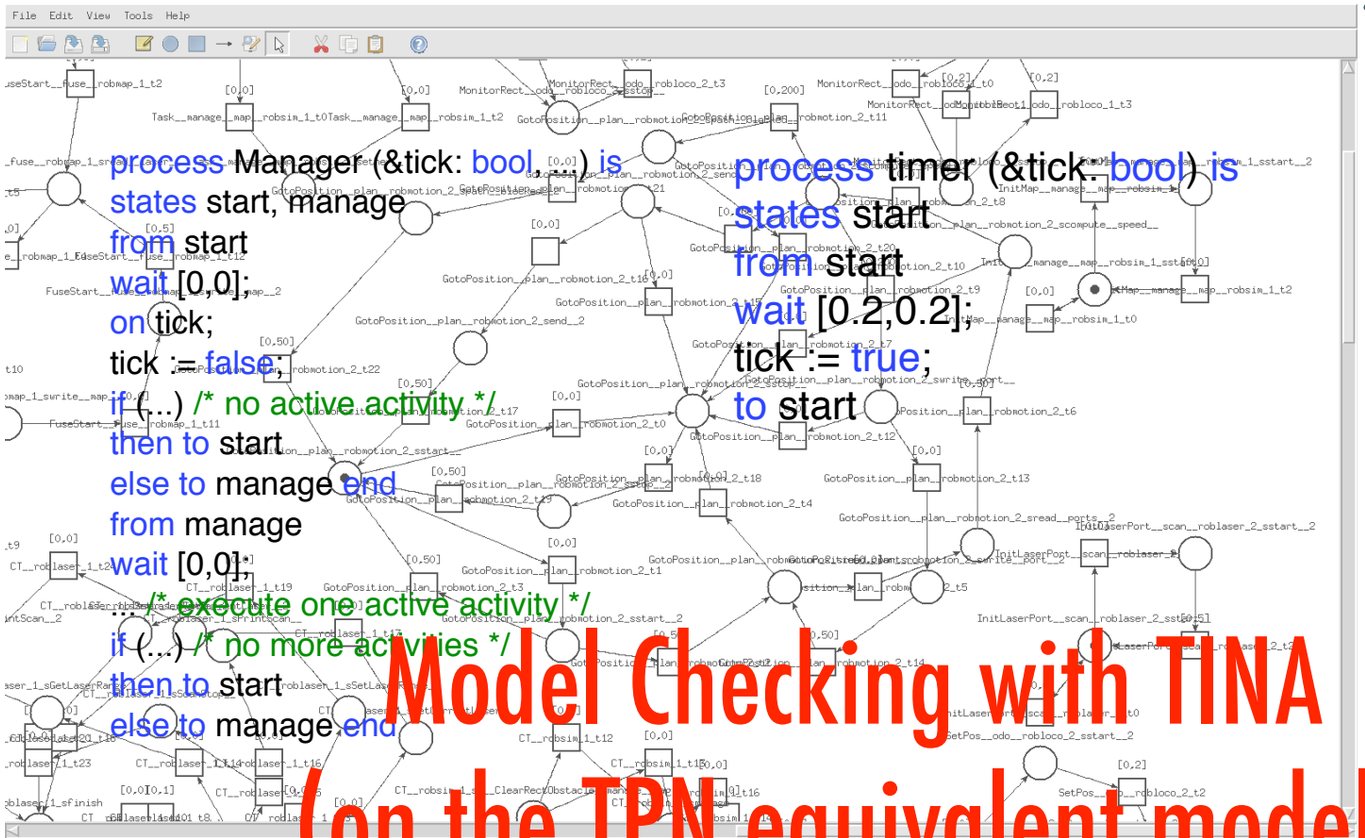
Save Undo

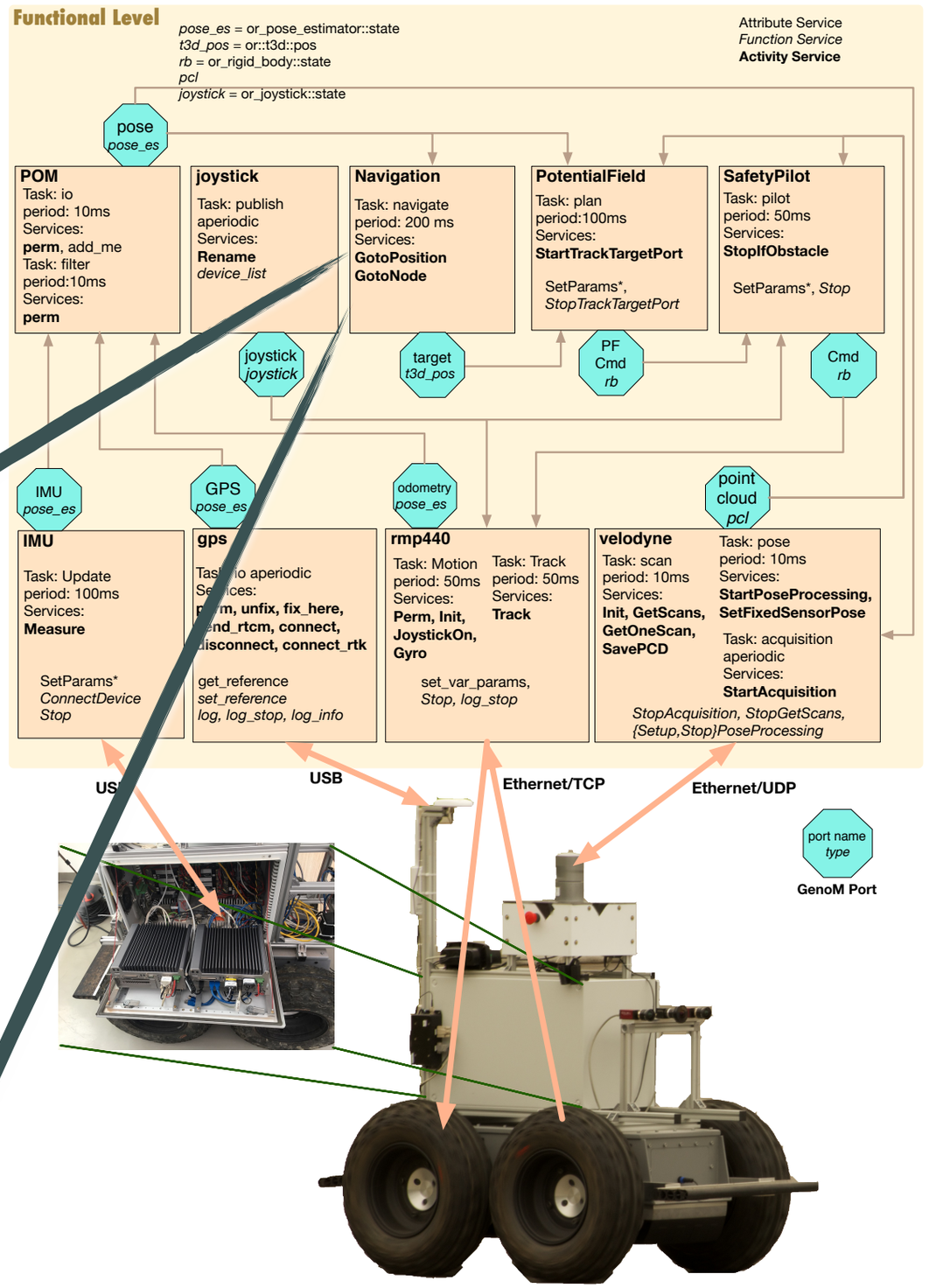
```
/* --- Generated by genom 2.99.33. Do not edit ------------------------- */

/* This module is automatically generated.
   Services codels' temporization is set according to their respective WCET
                          Mohammed Foughali and Felix Ingrand */

/* constants & types */

/* Global constants & types: ports mutual exclusion */
const PortsNumber: nat is 16

type PortsArray is array 16 of bool

const CodNumber_gps: nat is 28 /* number of codels: determines the size of the mutex boolean array */

const Max_gps: nat is 2 /* maximum of instances allowed per one self-compatible service, freely changeable */

const Nbactivities_io_gps: nat is 8/* number of services for io task*/

const SelfIncomp_io_gps: 0..Nbactivities_io_gps is 1/* number of self-incompatible services */

const maxinterrupts_gps: nat is 2 /* maximum length of the interrupts queue */

const Nbnonact_gps: nat is 8 /* number of attributes and functions */

const width_io_gps: nat is SelfIncomp_io_gps*2+(Nbactivities_io_gps-SelfIncomp_io_gps)*Max_gps

const width_gps: nat is width_io_gps /* size of the activities part*/

const size_gps: nat is width_gps + Nbnonact_gps + 1

type service_gps is union /* possible names of services */
        connect_io_gps |
        disconnect_io_gps |
        set_rtk_port_io_gps |
        connect_rtk_io_gps |
        unfix_io_gps |
        fix_io_gps |
        fix_here_io_gps |
        send_rtcm_io_gps |
        set_reference_gps |
        rtk_stats_gps |
        log_gps |
        log_stop_gps |
        log_info_gps |
        get_param_gps |
        get_last_fix_gps |
        get_reference_gps |
        None_gps
end

type servicetype_gps is union
        Attribute_gps |
        Function_gps |
        Activity_gps
end

type status_gps is union
-:---  gps-IMUDriver-joystick-Navigation-pom-PotentialField-rmp440-velodyne-SafetyPilot.fcr   Top L34    (Fiacre)
```



# Fiacre

## Fiacre model (9500 loc) of all the components




## Model Checking with TINA (on the TPN equivalent model)

# Verification with Fiacre

## Progress of activities

property no_block is (navigation/navigate/manager/state manage) leadsto (navigation/navigate/manager/state start)

Verification with TINA: TRUE

## Position port update bounded in time

## Navigation Stop leads to rmp440 stopping the robot

property bounded_stop_1 is (Navigation/control_task/state Stop_req) leadsto (Navigation/GotoPosition/state stop) within [0,0.2]

....

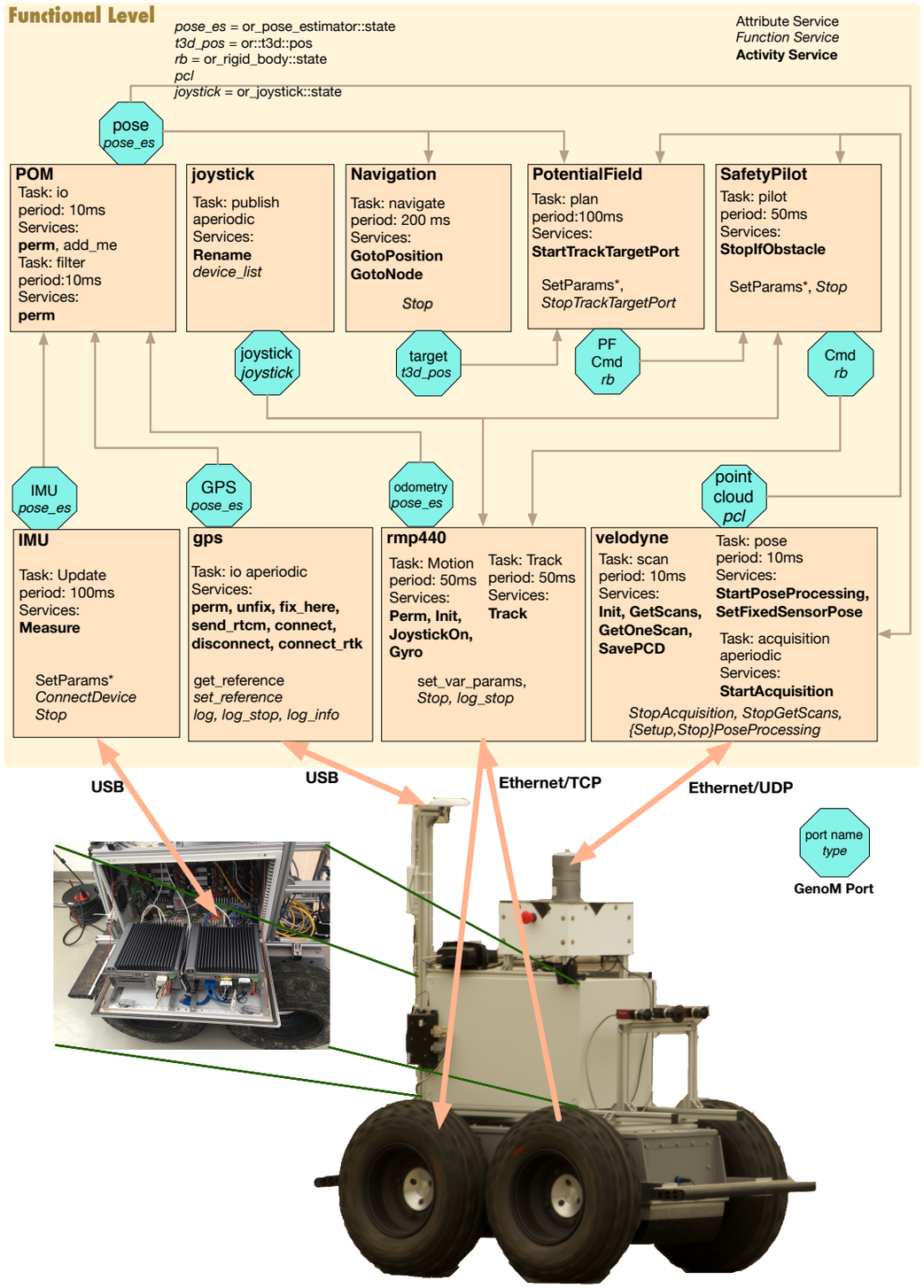
property bounded_stop_x is (rmp440/Track/state stop) leads to leave (rmp440/Track/state update) within [0,0.05]

## Schedulability of execution tasks

property schedulability_main is always (microkopter/main/state executing ⇒ not (main_period_signal))

Verification with TINA: FALSE

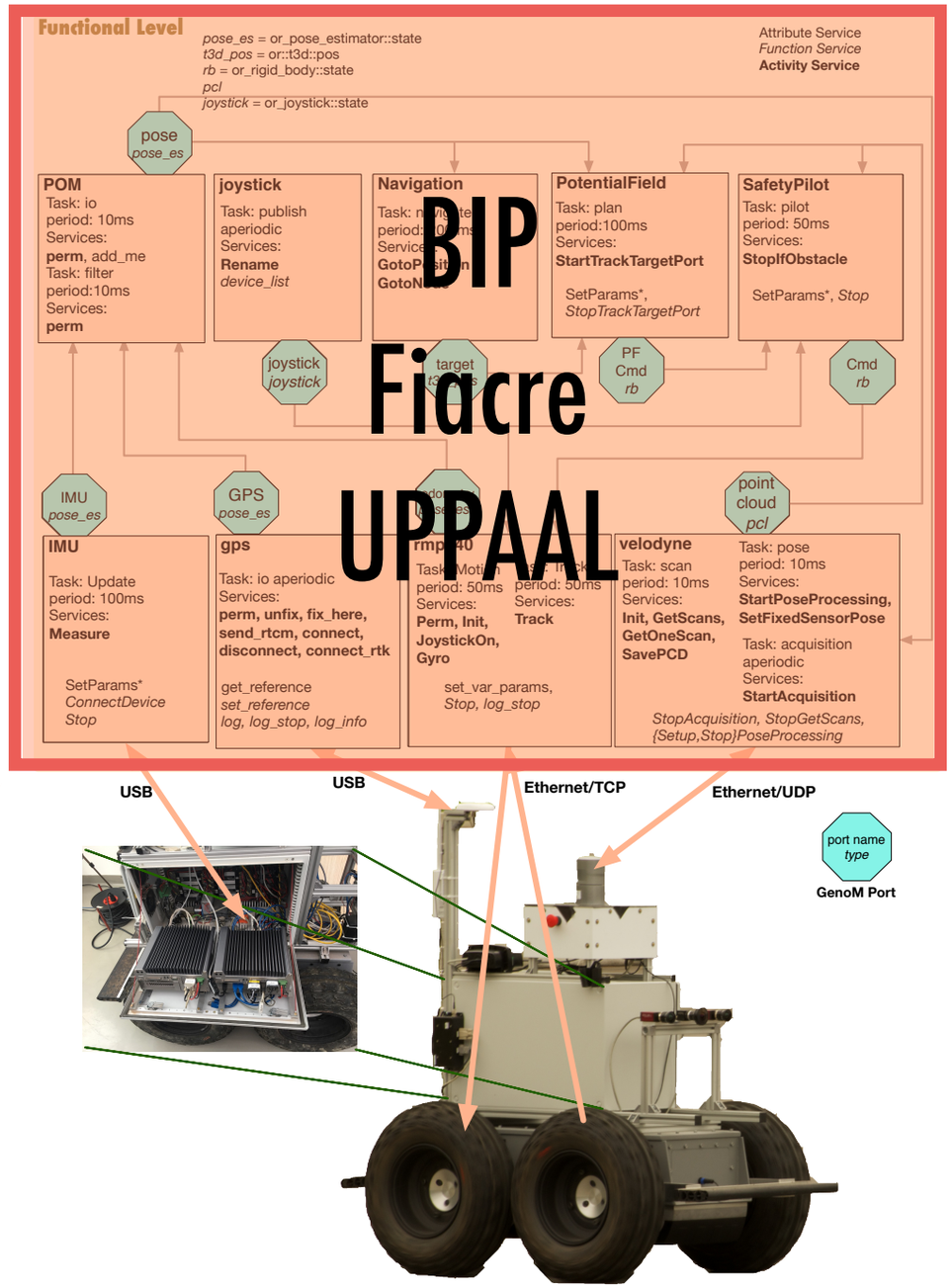
Hold for all tasks with an octo-core but not with a quad-core ODROID-C0



# What is in these formal models? What's the catch? How did we get them? (sorry, no Genies or slave student involved)

- BIP (Verimag)
- Fiacre/TINA (LAAS/VerTICS)
- UPPAAL (UPPsala & AALborg University)

Formal Methods/ Frameworks



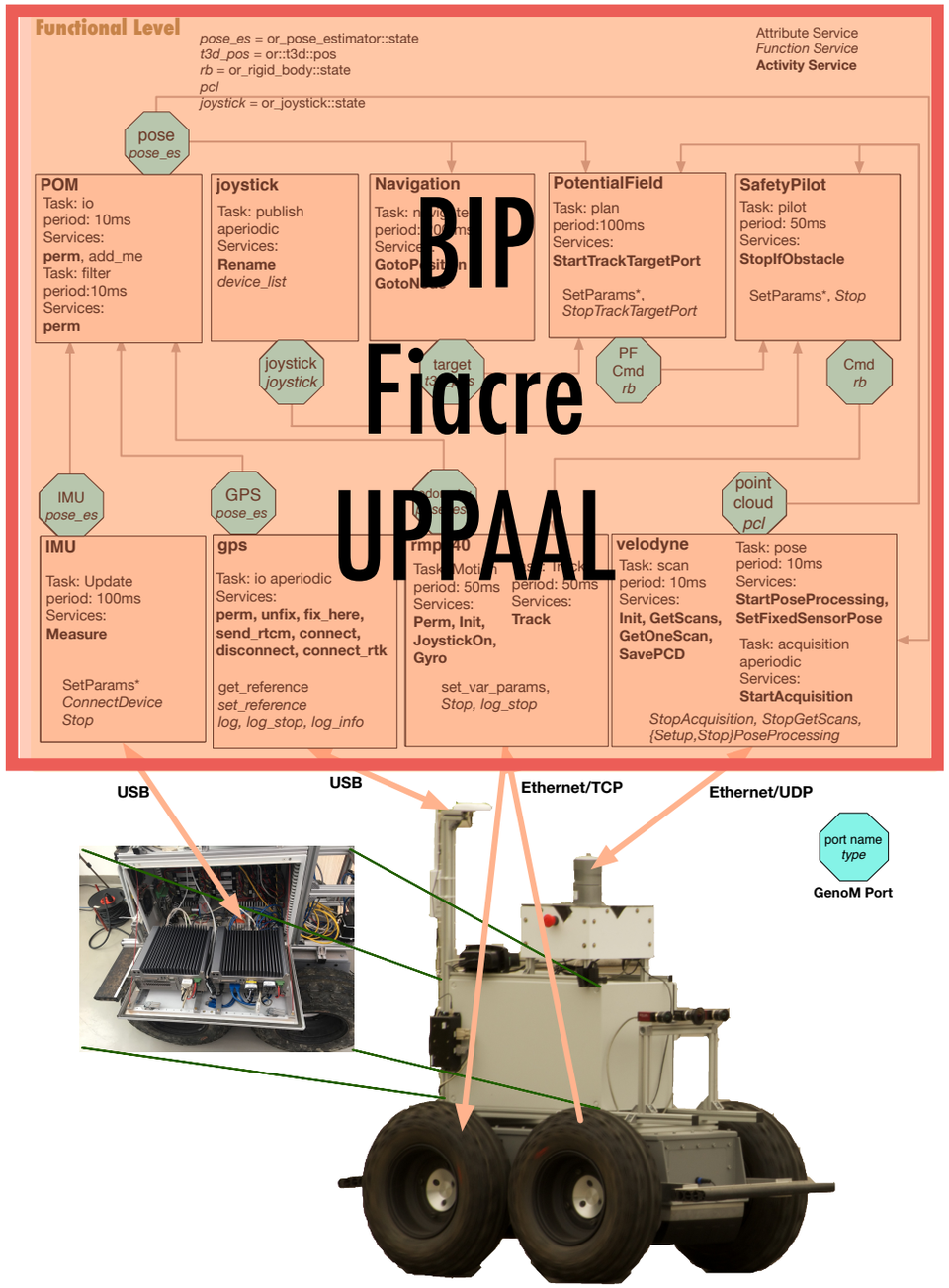
# What is in these formal models? What's the catch? How did we get them? (sorry, no Genies or slave student involved)

- Functional level : GenoM
- Modules
  - Services (control flow)
  - Ports (data flow)

Specification: Model-Driven Software Engineering

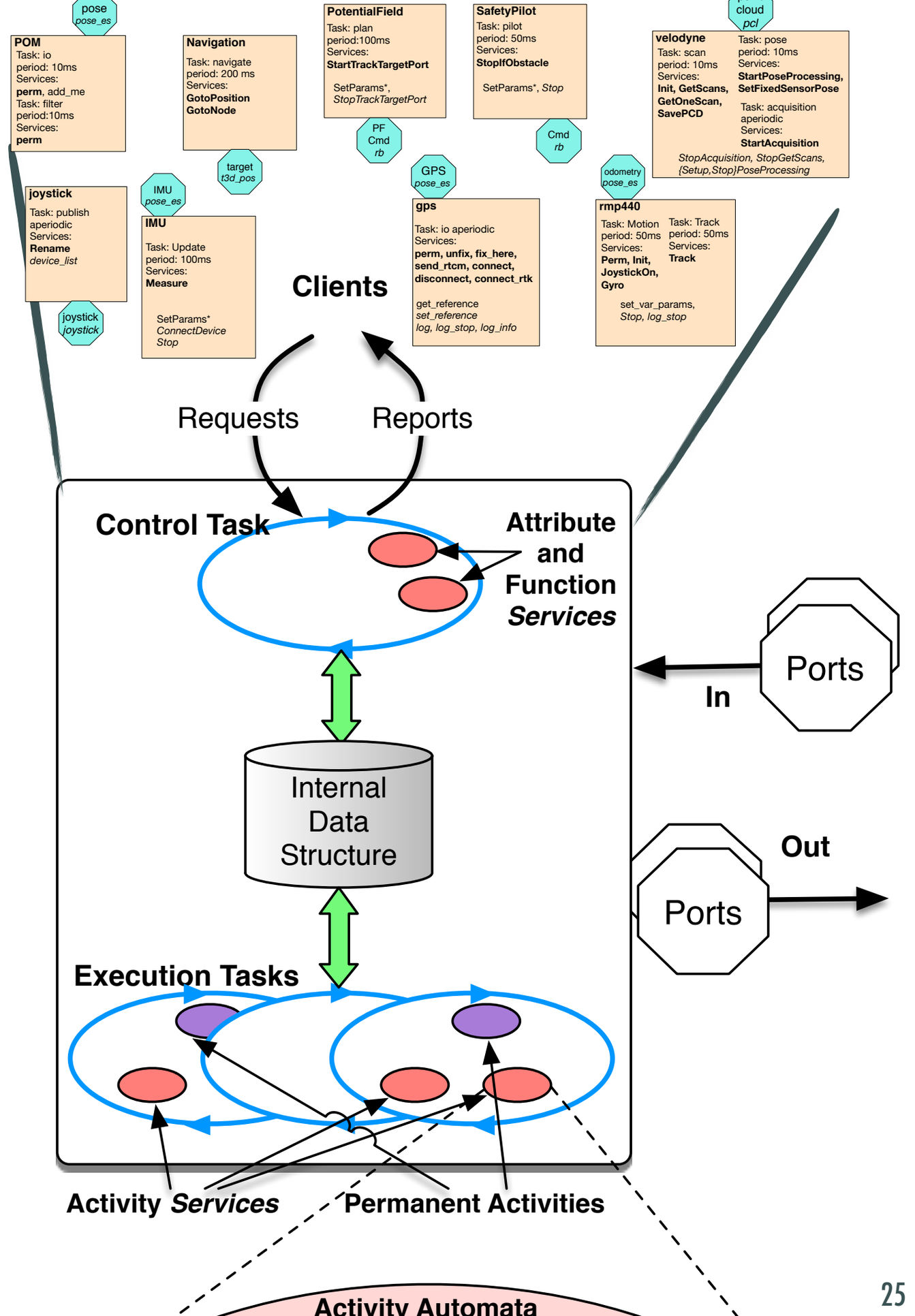
- BIP (Verimag)
- Fiacre/TINA (LAAS/VerTICS)
- UPPAAL (UPPsala & AALborg University)

Formal Methods/ Frameworks



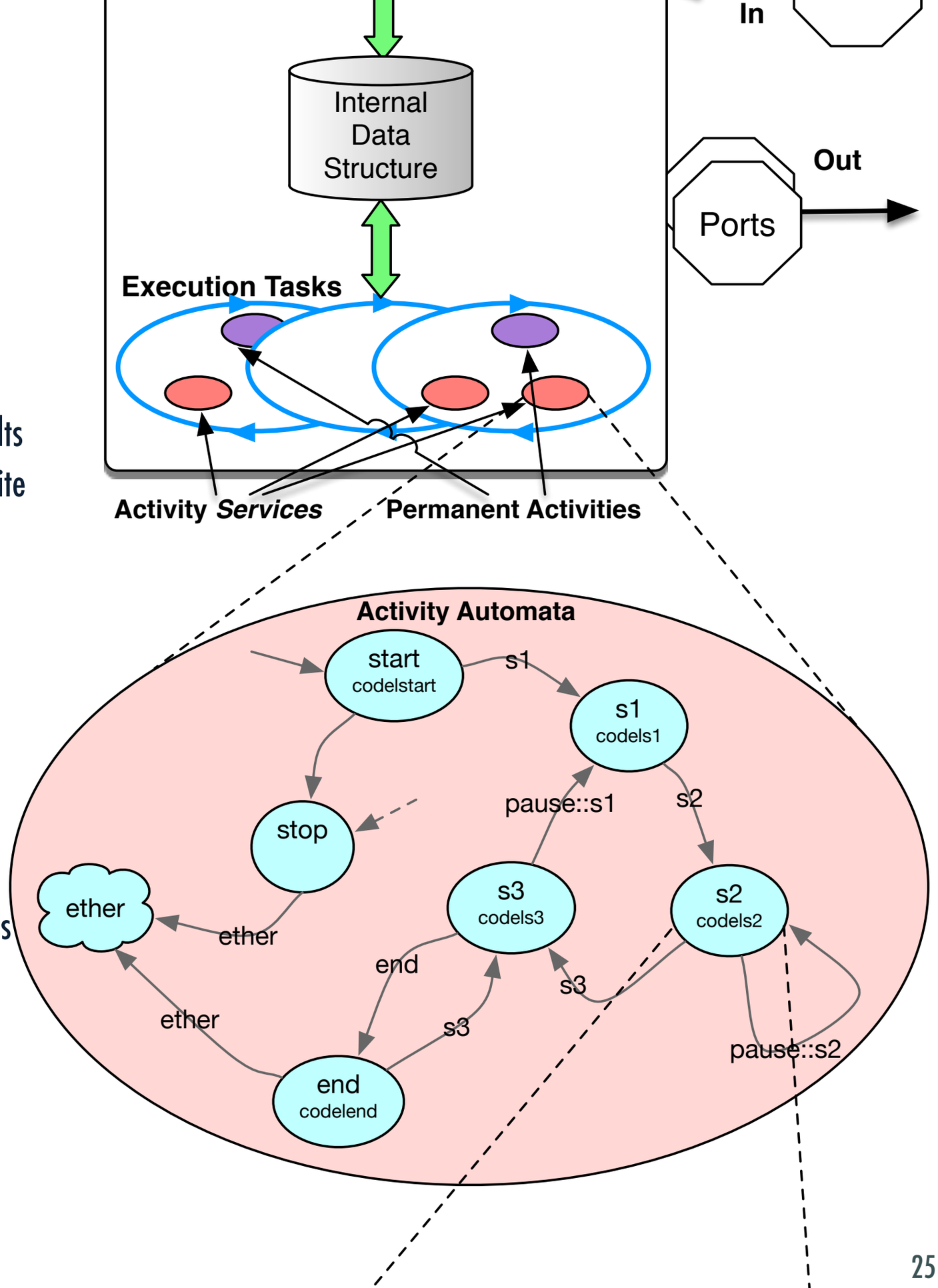
# GenoM internal

- To design a typical generic module which will be instantiated according to each specific module
- a module is a program
- a module has I/O
  - control: requests to start services/reports their results
  - data: ports in(to read external data) and out (to write external data)
- it supports a cyclic control task (aperiodic)
- and one or more cyclic execution tasks, periodic or aperiodic
- it provides services (fast and slow) to which we will associate C/C++ code
  - in the control task: attribute and function services (fast)
  - and the executions task(s): activity services (slow)
- services share a common Internal Data Structure for the needs of their computation (parameters, computed values, internal state variables, etc)



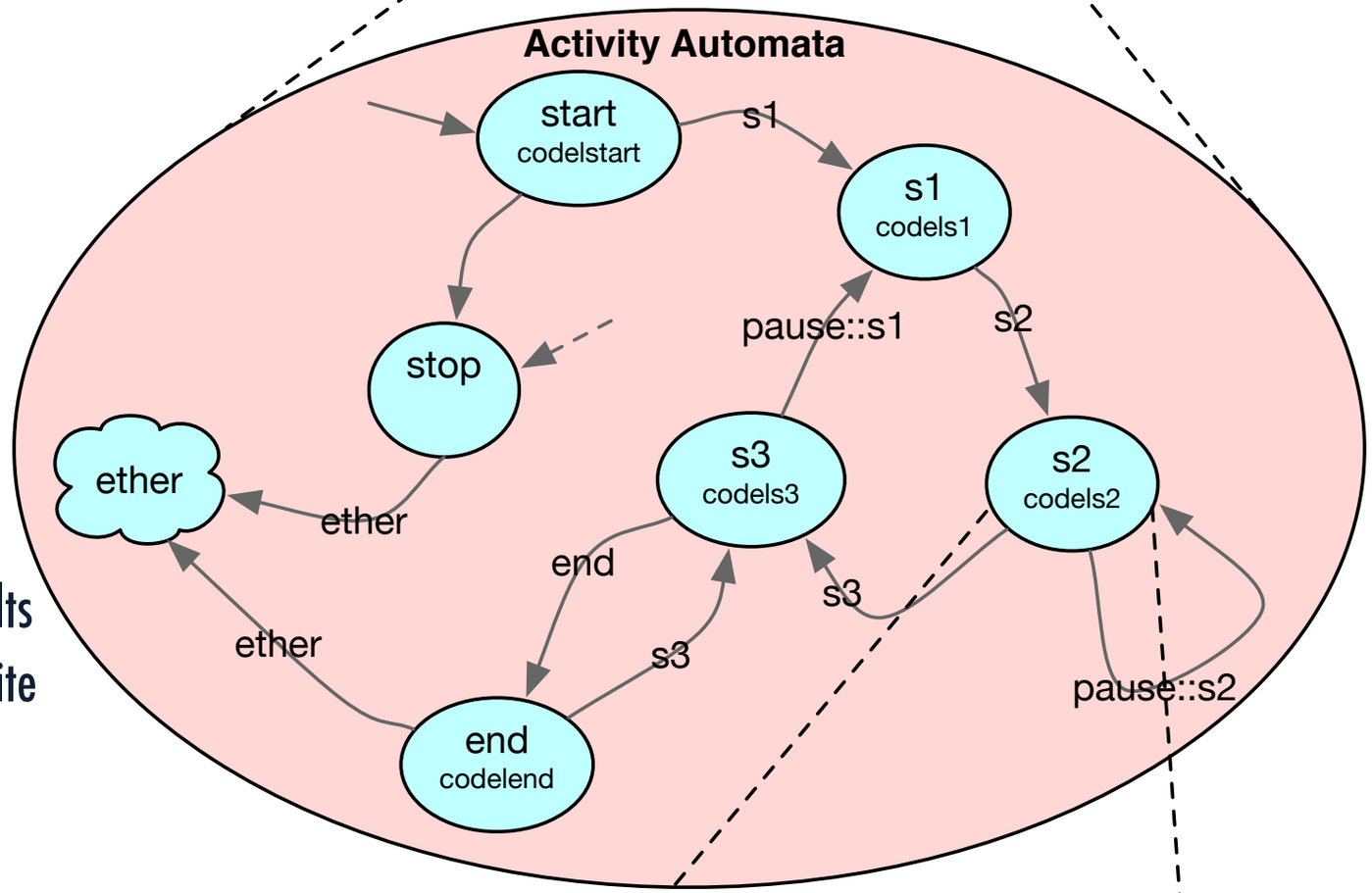
# GenoM internal

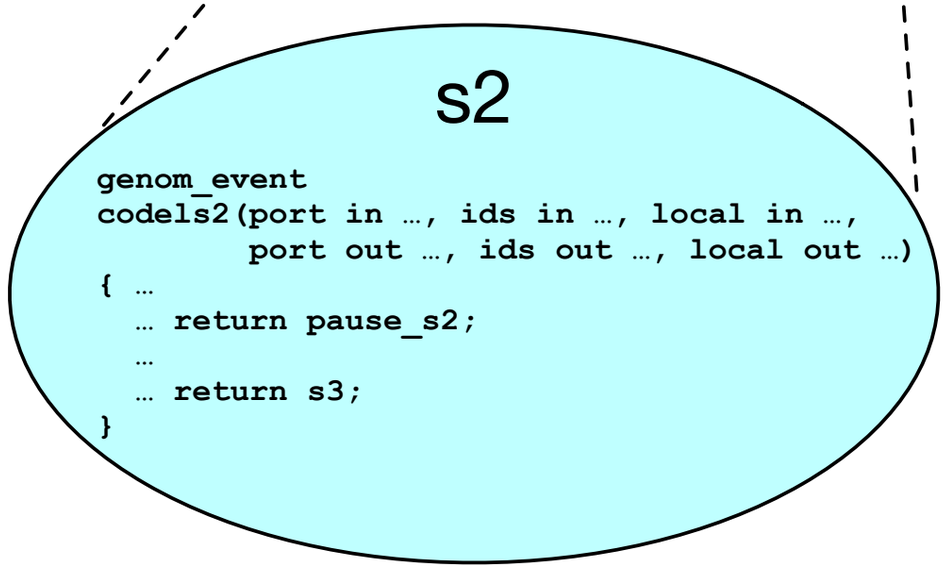
- To design a typical generic module which will be instantiated according to each specific module
- a module is a program
- a module has I/O
    - control: requests to start services/reports their results
    - data: ports in(to read external data) and out (to write external data)
- it supports a cyclic control task (aperiodic)
- and one or more cyclic execution tasks, periodic or aperiodic
- it provides services (fast and slow) to which we will associate C/C++ code
    - in the control task: attribute and function services (fast)
    - and the executions task(s): activity services (slow)
- services share a common Internal Data Structure for the needs of their computation (parameters, computed values, internal state variables, etc)
- activity services define automata to perform their processing



# GenoM internal

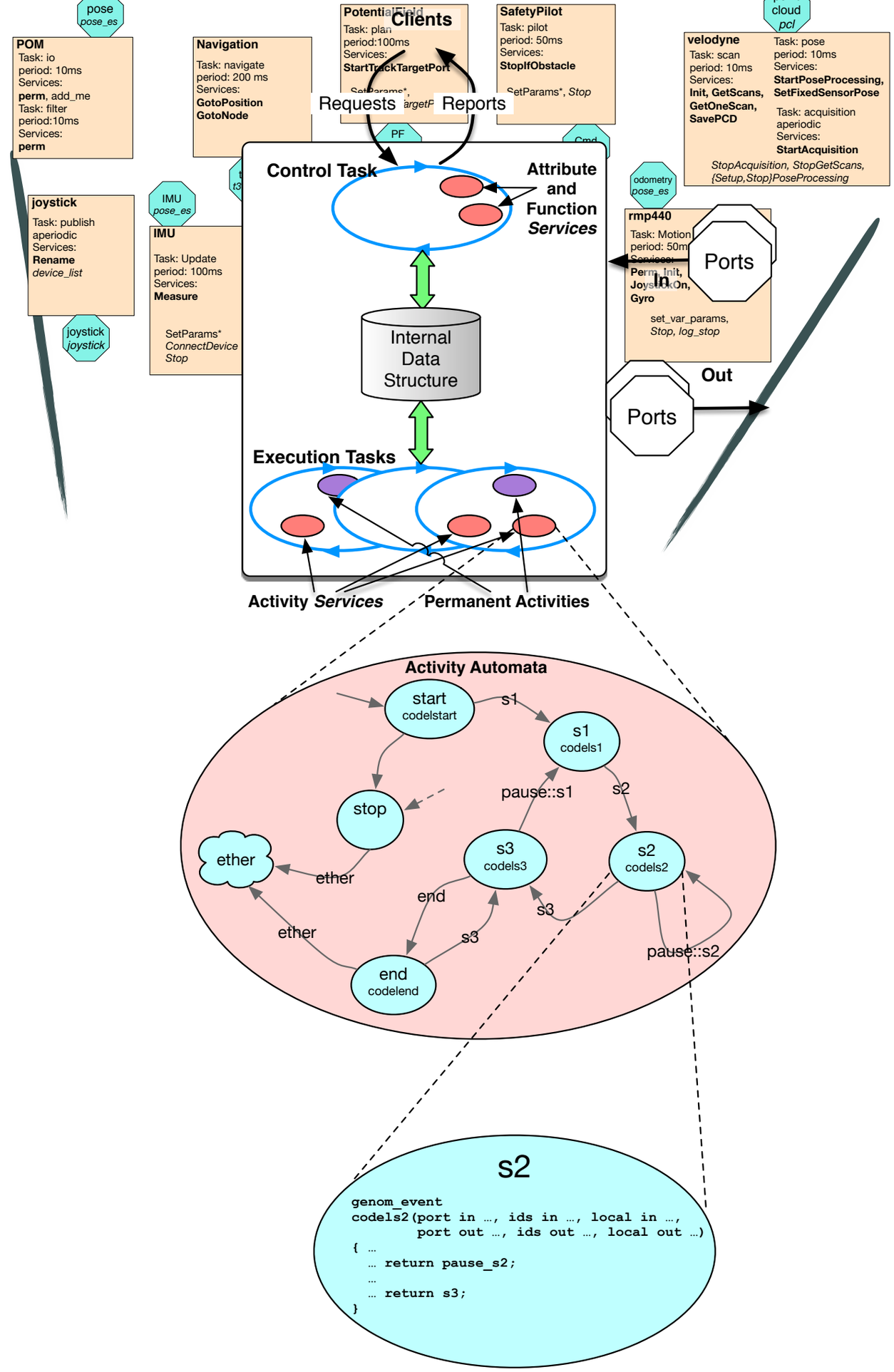
- To design a typical generic module which will be instantiated according to each specific module
- a module is a program
- a module has I/O
    - control: requests to start services/reports their results
    - data: ports in(to read external data) and out (to write external data)
- it supports a cyclic control task (aperiodic)
- and one or more cyclic execution tasks, periodic or aperiodic
- it provides services (fast and slow) to which we will associate C/C++ code
    - in the control task: attribute and function services (fast)
    - and the executions task(s): activity services (slow)
- services share a common Internal Data Structure for the needs of their computation (parameters, computed values, internal state variables, etc)
- activity services define automata to perform their processing
- each step is associated to a codel ( C/C++ code)



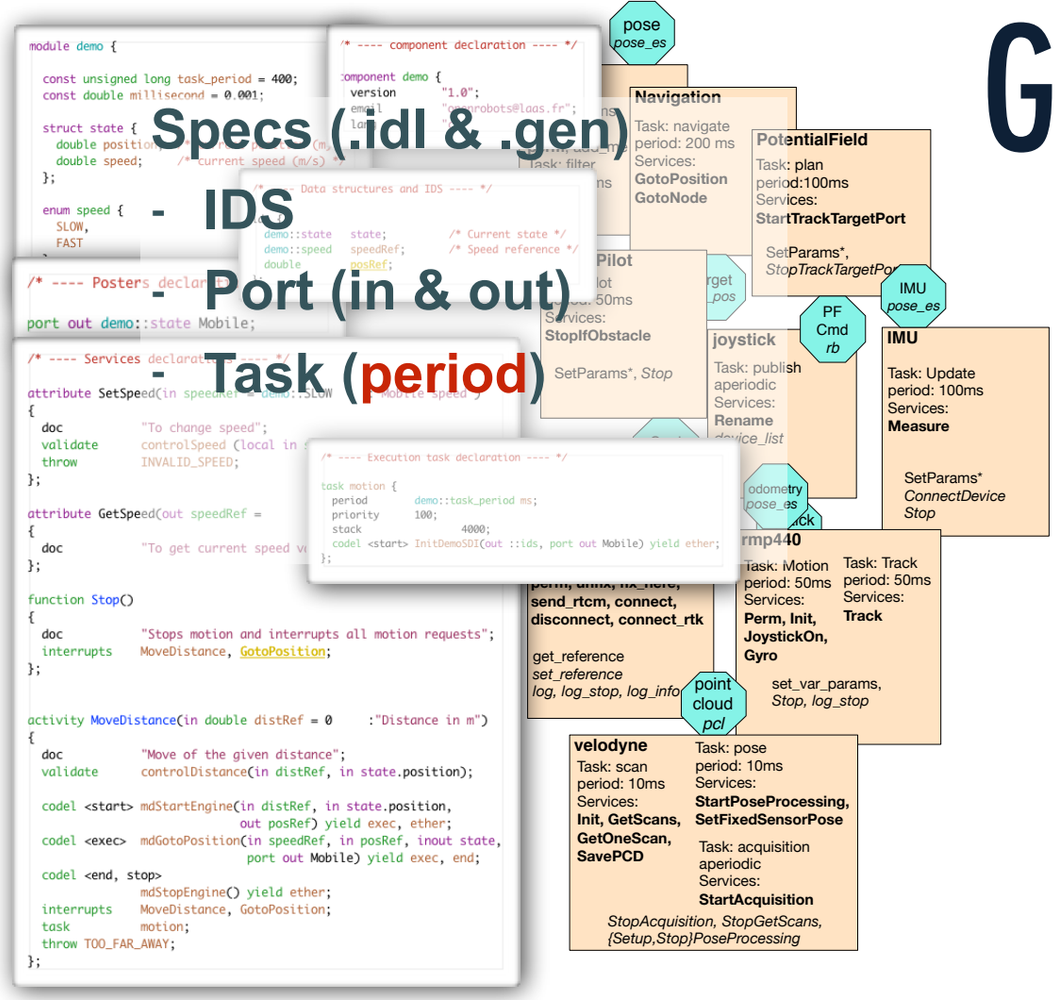



# GenoM specifications

- IDS
- Ports (in & out)
- Execution Tasks
  - (periodic or aperiodic)
- Services
  - Attribute, function
  - Activity
    - automata
    - and attached codels with WCET...

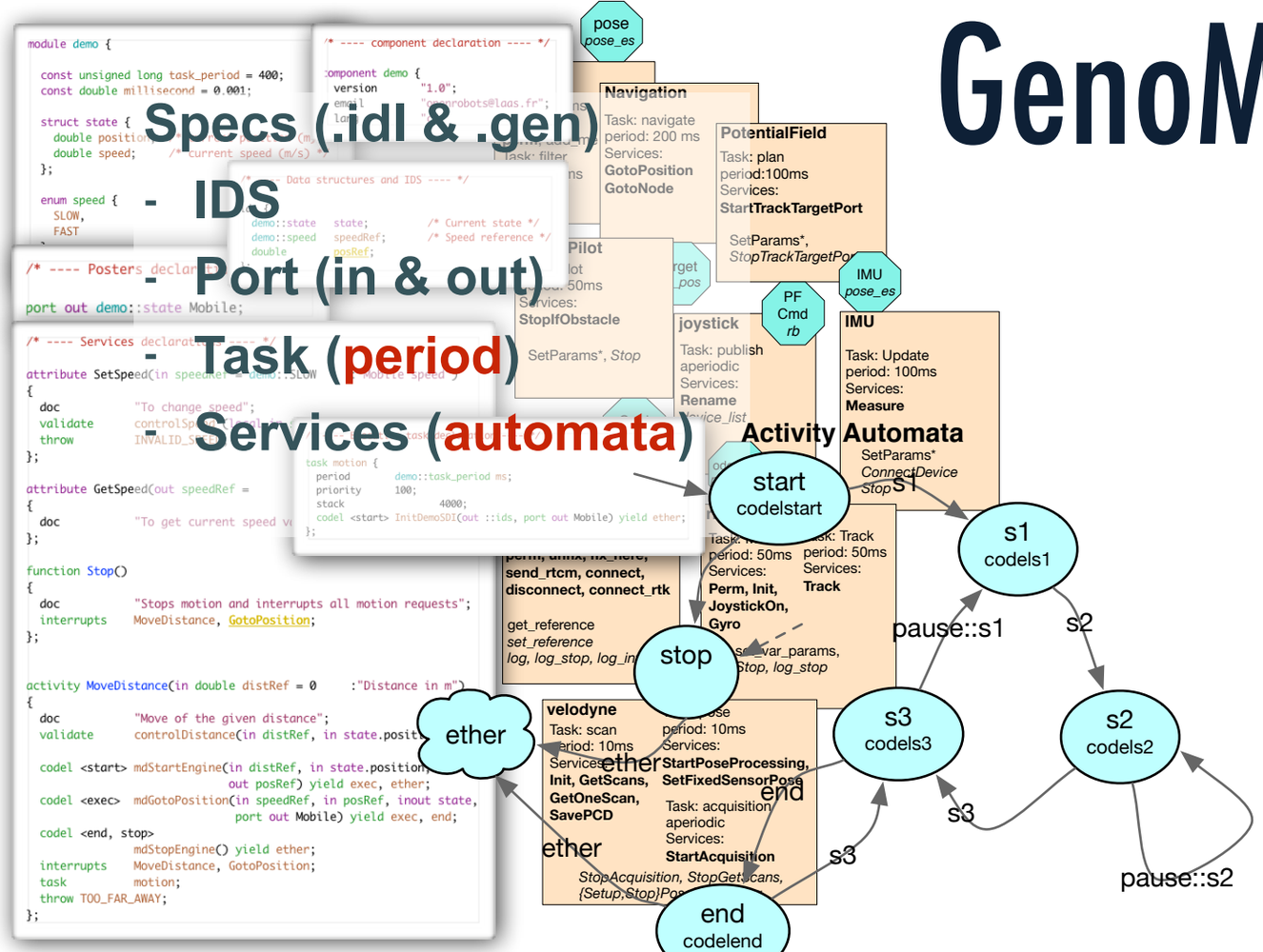


# GenoM workflow



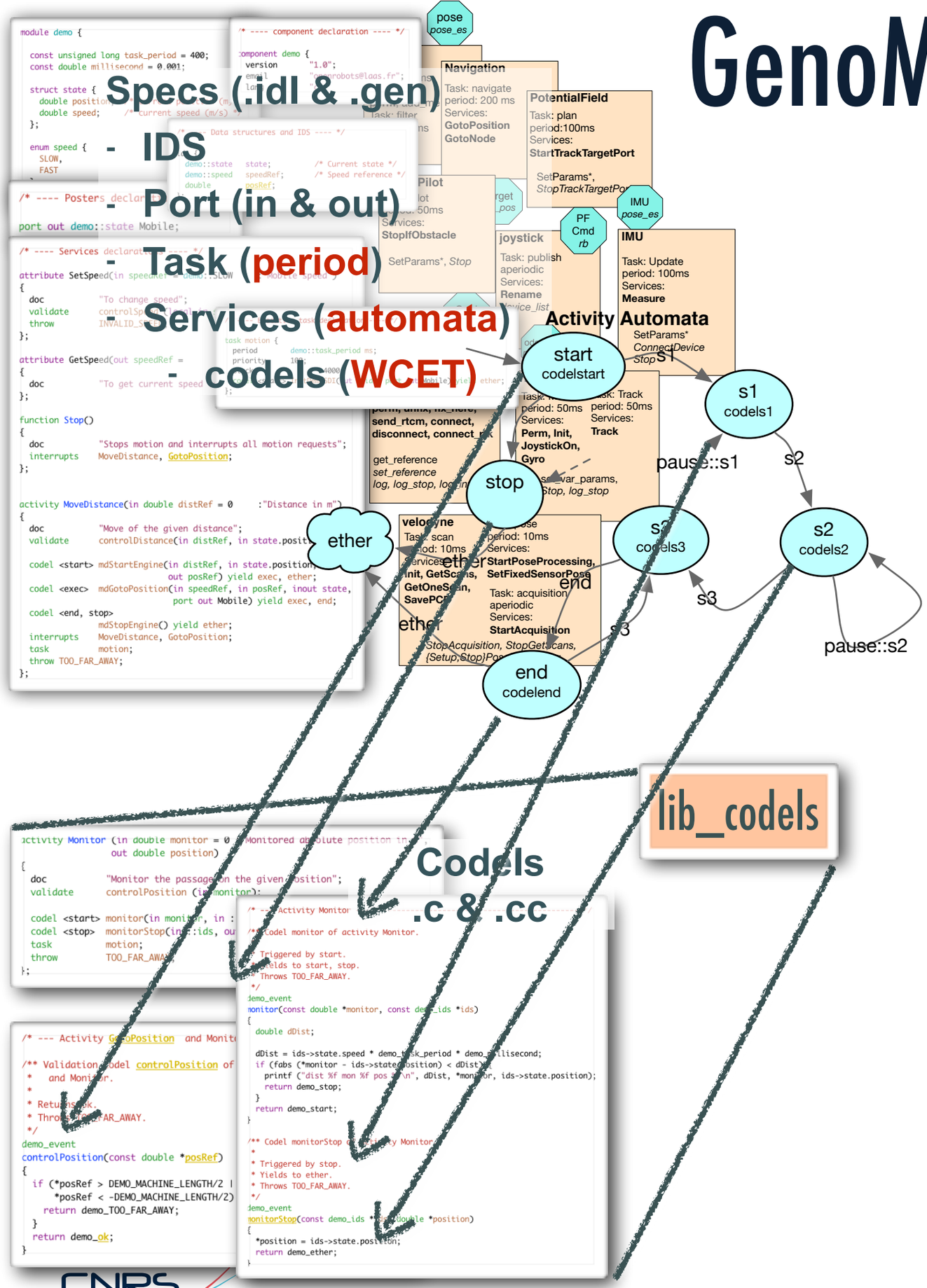
# GenoM workflow

- Specs (.idl & .gen)
- IDS
- Port (in & out)
- Task (period)
- Services (automata)



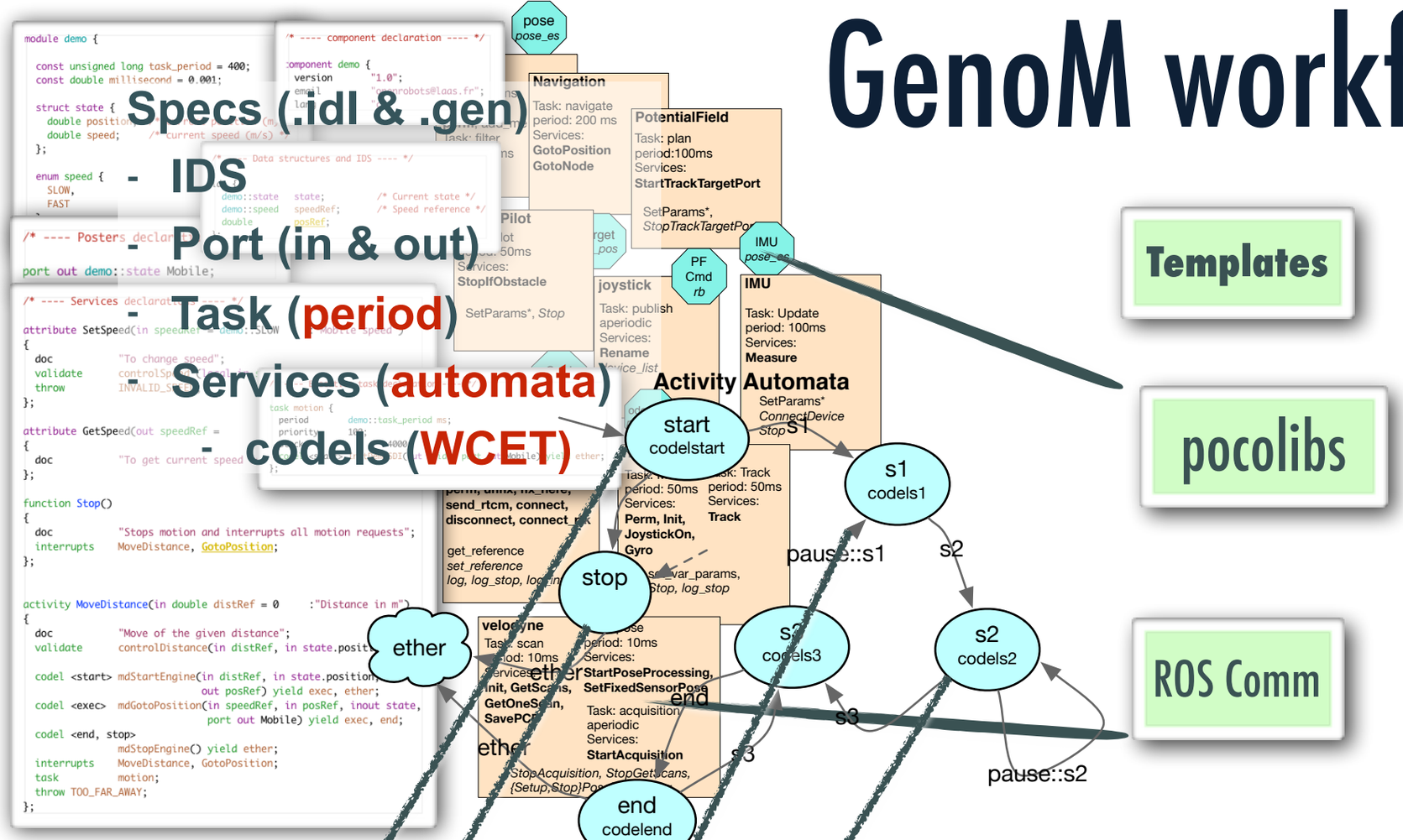
# GenoM workflow

- Specs (.idl & .gen)
- IDS
- Port (in & out)
- Task (period)
- Services (automata)
- codels (WCET)



# GenoM workflow

- Specs (.idl & .gen)
- IDS
- Port (in & out)
- Task (period)
- Services (automata)
- codels (WCET)



# GenoM workflow

- Specs (.idl & .gen)
- IDS
- Port (in & out)
- Task (period)
- Services (automata)
- codels (WCET)

```
module demo {

  const unsigned long task_period = 400;
  const double millisecond = 0.001;

  struct state {
    double position;
    double speed;     /* current speed (m/s) */
  };

  enum speed {
    SLOW,
    FAST
  };
```

```
/* ---- Posters declaration ---- */

port out demo::state Mobile;
```

```
/* ---- Services declaration ---- */

attribute SetSpeed(in speedRef = demo::SLOW  "Mobile speed")
{
  doc           "To change speed";
  validate      controlSpeed(speedRef);
  throw         INVALID_SPEED;
};

attribute GetSpeed(out speedRef =
{
  doc           "To get current speed";
};

function Stop()
{
  doc           "Stops motion and interrupts all motion requests";
  interrupts    MoveDistance, GotoPosition;
};


activity MoveDistance(in double distRef = 0     :"Distance in m")
{
  doc           "Move of the given distance";
  validate      controlDistance(in distRef, in state.position);

  codel <start> mdStartEngine(in distRef, in state.position,
                              out posRef) yield exec, ether;
  codel <exec>  mdGotoPosition(in speedRef, in posRef, inout state,
                               port out Mobile) yield exec, end;

  codel <end, stop>
                mdStopEngine() yield ether;
  interrupts    MoveDistance, GotoPosition;
  task          motion;
  throw TOO_FAR_AWAY;
};
```

```
/* ---- component declaration ---- */

component demo {
  version       "1.0";
  email         "openrobots@laas.fr";
  lang          "c";
  ...
};

/* ---- Data structures and IDS ---- */

demo::state   state;          /* Current state */
demo::speed   speedRef;       /* Speed reference */
double        posRef;
```

```
task motion {
  period        demo::task_period ms;
  priority      100;
  ...
  codel <start> start(...) yield ether;
};
```

pose
*pose_es*

**Navigation**
Task: navigate
period: 200 ms
Services:
**GotoPosition**
**GotoNode**

**PotentialField**
Task: plan
period:100ms
Services:
**StartTrackTargetPort**
SetParams*,
*StopTrackTargetPo*

**Pilot**
lot
50ms
Services:
**StopIfObstacle**
SetParams*, *Stop*

IMU
*pose_es*

PF
Cmd
*rb*

**joystick**
Task: publish
aperiodic
Services:
**Rename**

**IMU**
Task: Update
period: 100ms
Services:
**Measure**

**Activity Automata**
SetParams*
*ConnectDevice*
*Stop*

start
codelstart

s1
codels1

s1

pause::s1

s2

stop

ether

s2
codels3

s2
codels2

s3

end

pause::s2

end
codelend

**velodyne**
Task: scan
period: 10ms
Services:
**Init, GetScans, GetOneScan, SavePCD**

period: 10ms
Services:
**StartPoseProcessing, SetFixedSensorPose**

Task: acquisition
aperiodic
Services:
**StartAcquisition**
*StopAcquisition, StopGetScans, {Setup;Stop}Pos*

**send_rtcm, connect, disconnect, connect_ntrip**
get_reference
*set_reference*
*log, log_stop, log_in*

Task: Track
period: 50ms
Services:
**Track**

period: 50ms
Services:
**Perm, Init, JoystickOn, Gyro**
*set_var_params, Stop, log_stop*

```
activity Monitor (in double monitor = 0  "Monitored absolute position in m",
                  out double position)
{
  doc           "Monitor the passage on the given position";
  validate      controlPosition (in monitor);

  codel <start> monitor(in monitor, in :
  codel <stop>  monitorStop(in ::ids, ou
  task          motion;
  throw         TOO_FAR_AWA
};
```

```
/* --- Activity GotoPosition  and Monito

/** Validation codel controlPosition of
 *   and Monitor.
 *
 * Returns ok.
 * Throws TOO_FAR_AWAY.
 */
demo_event
controlPosition(const double *posRef)
{
  if (*posRef > DEMO_MACHINE_LENGTH/2 |
      *posRef < -DEMO_MACHINE_LENGTH/2)
    return demo_TOO_FAR_AWAY;
  }
  return demo_ok;
}
```

```
/* --- Activity Monitor
/** Codel monitor of activity Monitor.

 * Triggered by start.
 * Yields to start, stop.
 * Throws TOO_FAR_AWAY.
 */
demo_event
monitor(const double *monitor, const demo_ids *ids)
{
  double dDist;

  dDist = ids->state.speed * demo_task_period * demo_millisecond;
  if (fabs (*monitor - ids->state.position) < dDist) {
    printf ("dist %f mon %f pos %f\n", dDist, *monitor, ids->state.position);
    return demo_stop;
  }
  return demo_start;
}

/** Codel monitorStop of activity Monitor.
 *
 * Triggered by stop.
 * Yields to ether.
 * Throws TOO_FAR_AWAY.
 */
demo_event
monitorStop(const demo_ids *ids, double *position)
{
  *position = ids->state.position;
  return demo_ether;
}
```

Templates

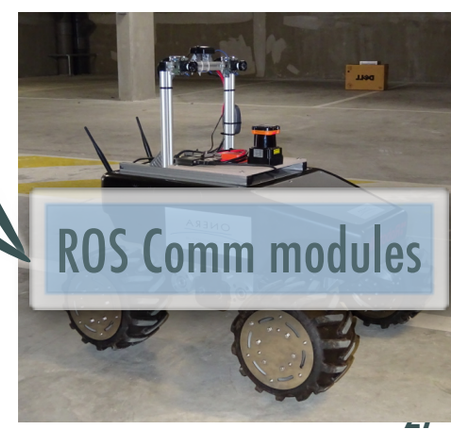
pocolibs

ROS Comm

lib_codels

Codels
.c & .cc






CNRS

# GenoM workflow

- Specs (.idl & .gen)
- IDS
- Port (in & out)
- Task (period)
- Services (automata)
- codels (WCET)

Templates

pocolibs

ROS Comm

UPPAAL

BIP

FIACRE

lib_codels

lib_BIP_Engine

BIP model

Fiacre model

pocolibs modules

UPPAAL model

BIP modules

TINA model

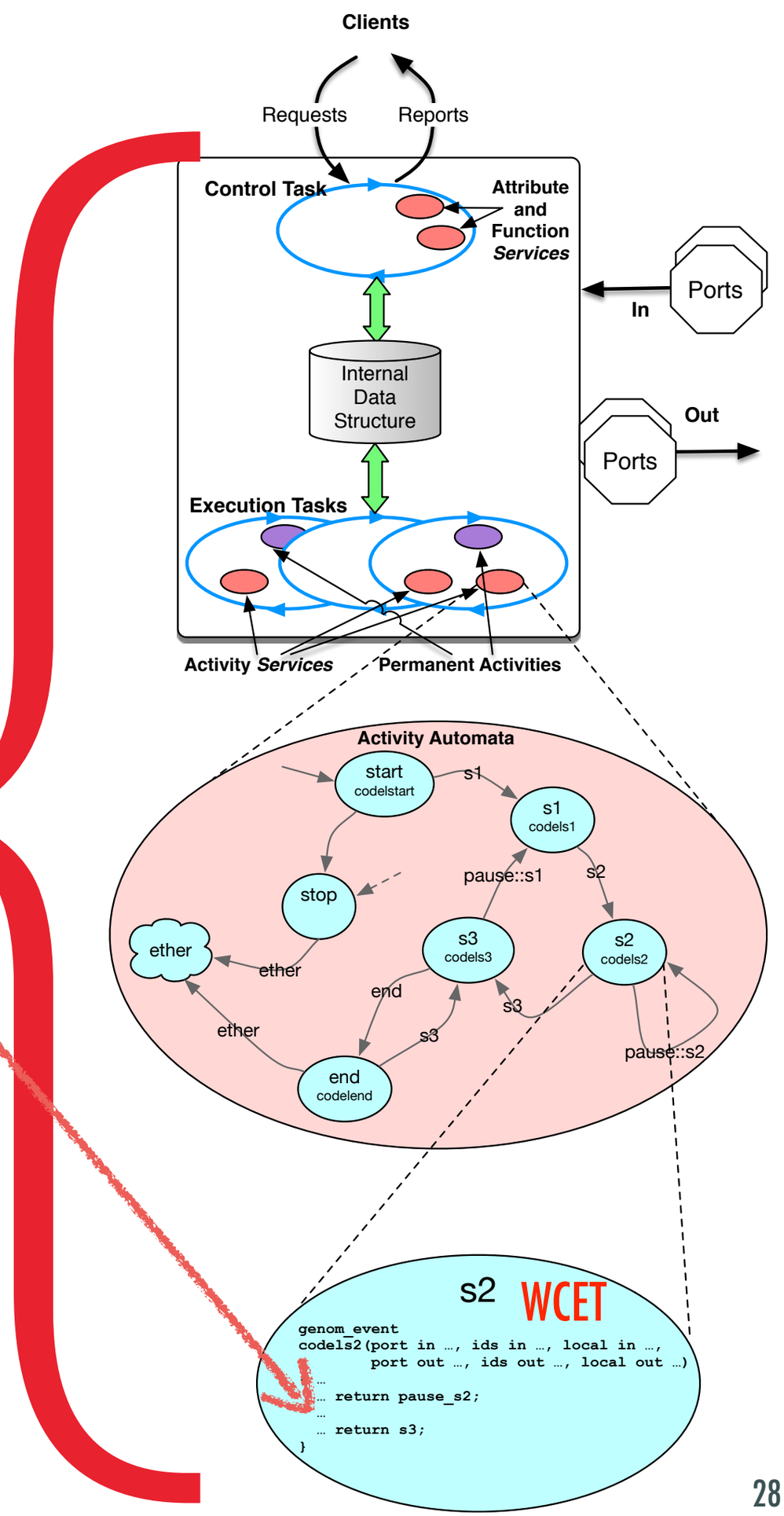
ROS Comm modules

Codels
.c & .cc

# Formal specifications

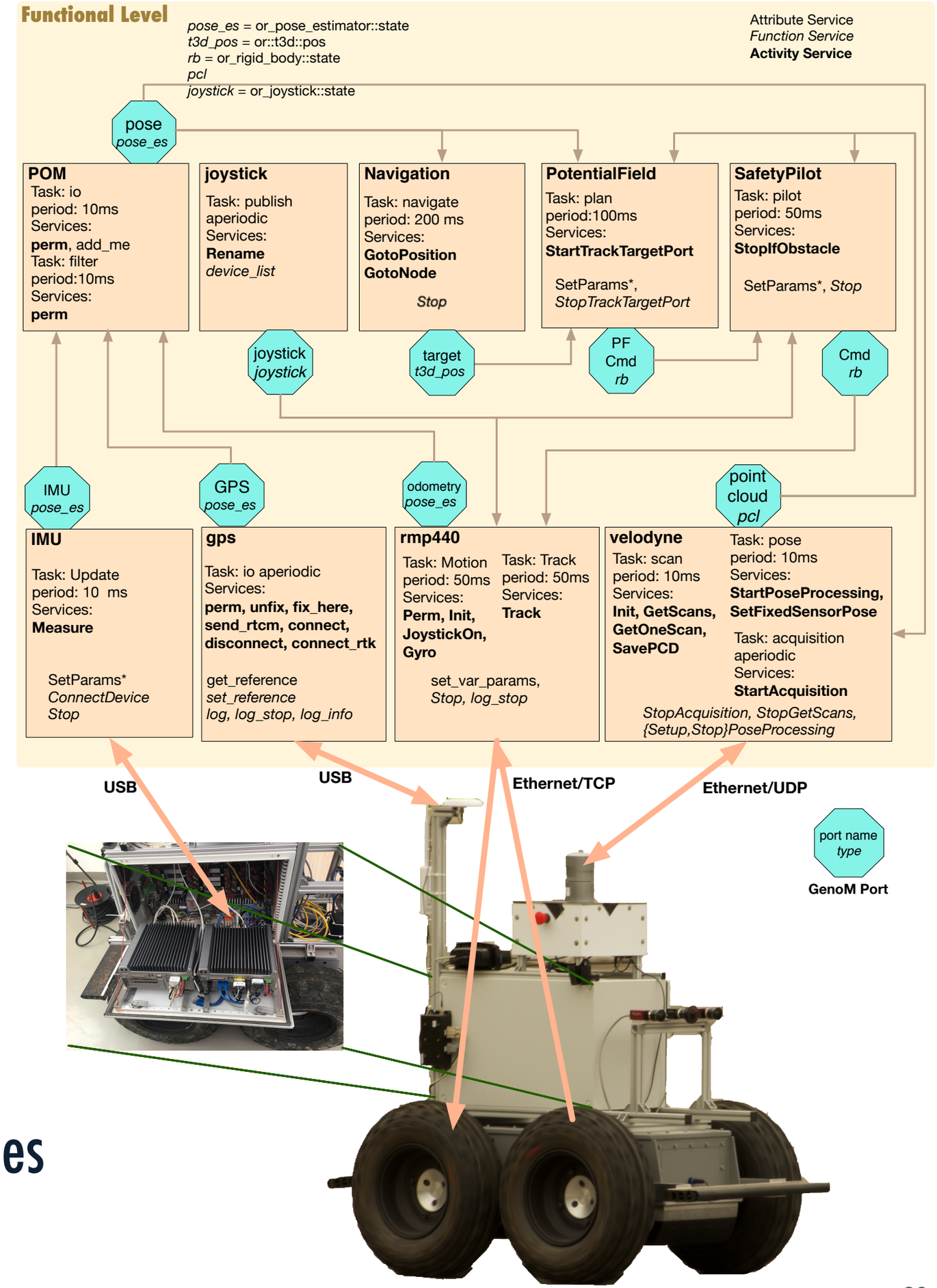
- All this get translated to : BIP, UPPAAL, Fiacre/TINA

- Except the specific user defined C/C++ code which is inside a codel (it is abstracted with a WCET)



# Minnie software architecture

- 9 components
- 9 ports
- (13 + 9) tasks (period)
- 38 activity services (with codels automata)
- 41 function services
- 43 attribute services
- 170 codels (14k loc) and their WCET
- 200k loc for all components + libraries



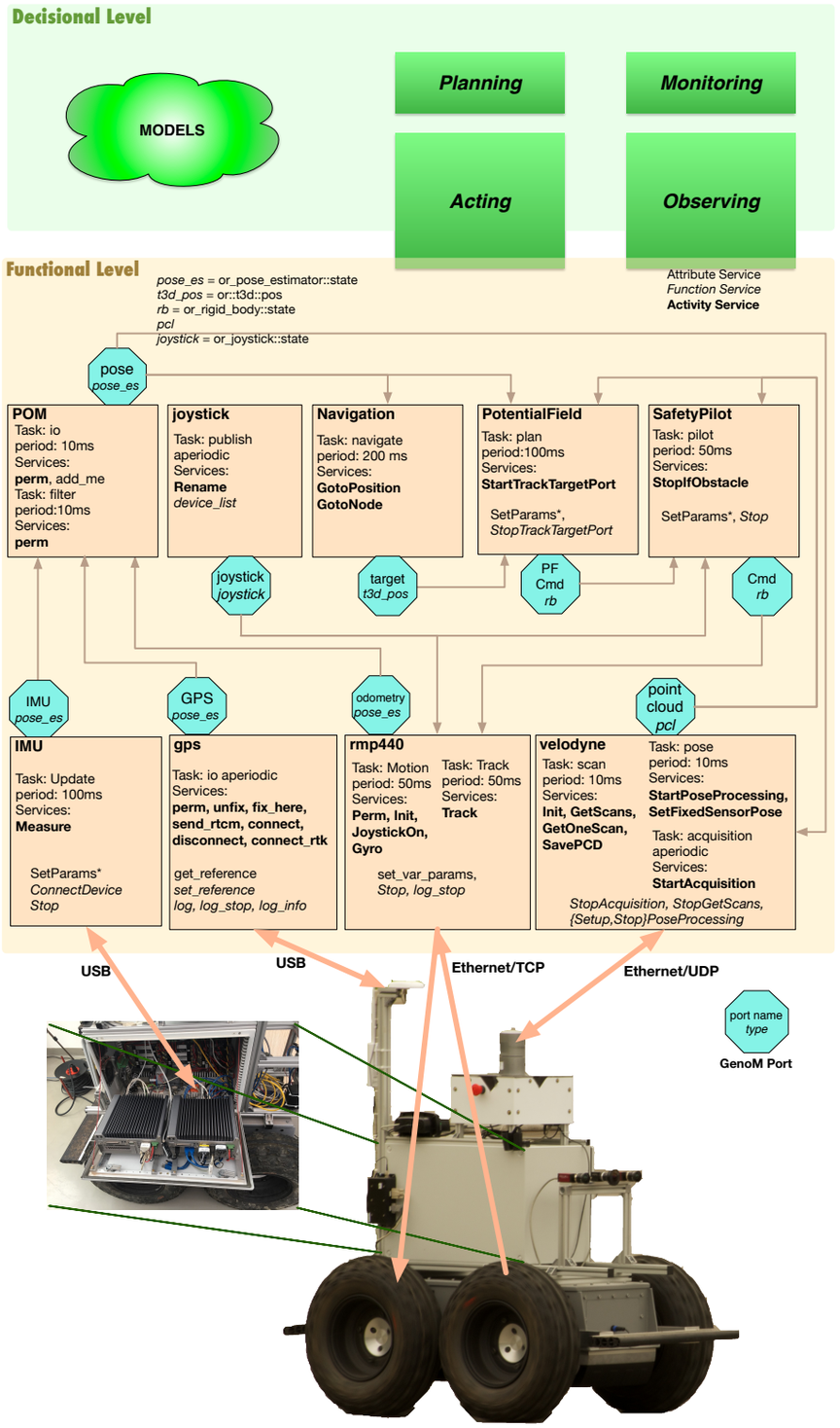
# Current GenoM V&V templates

| Formal Frameworks \ Middleware | Offline | Online PocoLibs | Online ROS |
|---|---|---|---|
| **BIP** | - RT D-Finder | ++ BIP Engine | ++ BIP Engine |
| **FIACRE** | ++ TINA | Hippo | Under Dev |
| **UPPAAL** | +++ | | |
| **UPPAAL SMC** | ++ | | |

- The BIP-PocoLibs/ROS models for the BIP Engine are complete and functional
- The Fiacre template is complete and tested on numerous modules (model over multiple modules and ports communication), UPPAAL has a slight performance advantage
- Between Fiacre and UPPAAL there are pros and cons
- The Fiacre Engine Hippo is under development (alpha)

# Why focus on a bottom up approach?

- Apart from learned models, most decisional components rely on explicit models which can be fed to V&V tools
- Lower level components often lack formalism
- Beware, if we try to check high level properties, on unverified low level components, we are building on quick sands...
- Try to start from existing “robotics” frameworks



# V&V models: Different situations over a complete autonomous system

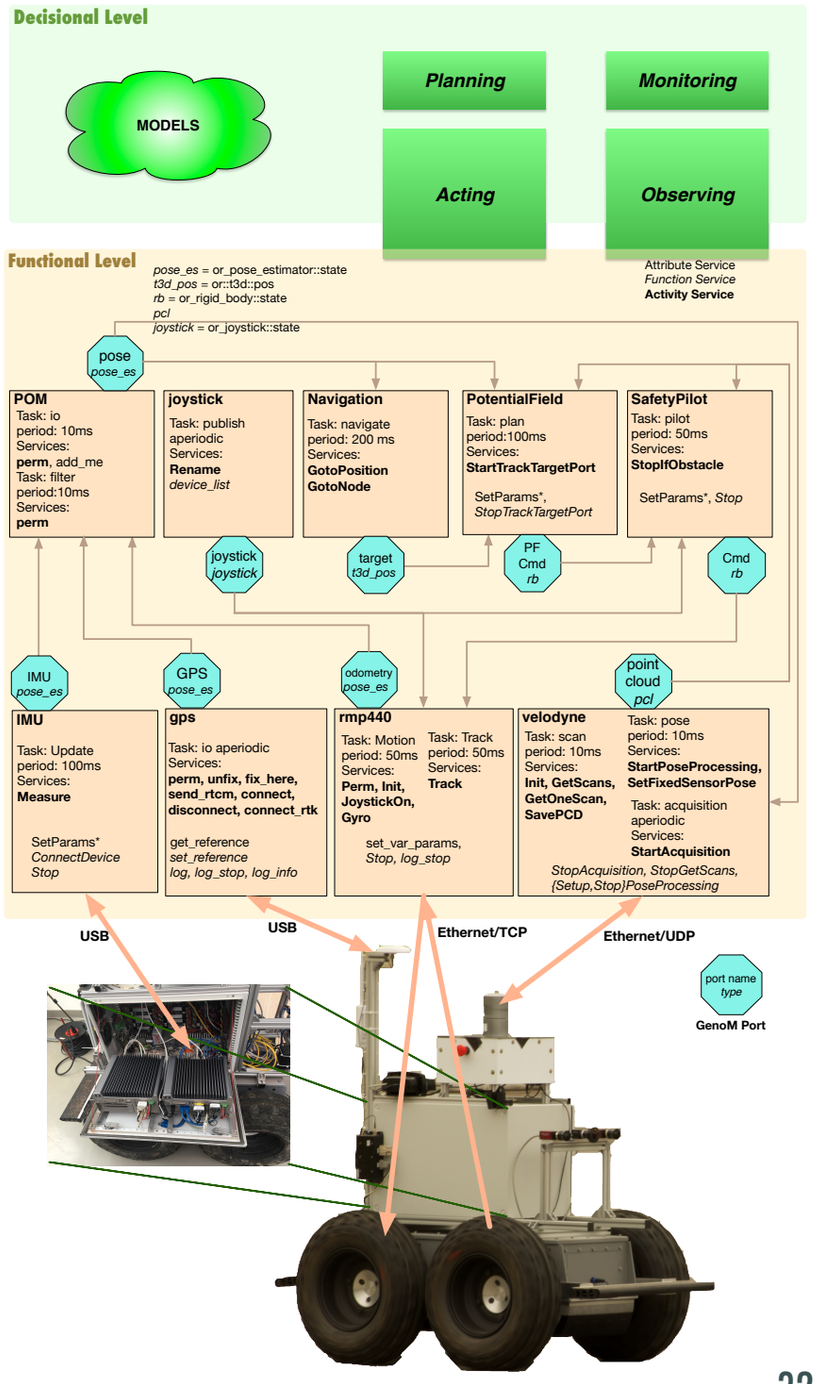
**Formal models:** decisional : planning (e.g. UPPAAL, model checking), monitoring, FDIR (e.g. RMPL), observing (e.g. DyKnow)

**Learned models:** Reinforcement learning models, perception models, action models, etc.

**Specification models:** Robotics software engineering models: e.g. GenoM, Oroccos, MAUVE, BIC, SmartMDSD, RobotML, etc.

**Programming directly the Model:** Orccad, Esterel, Scade, RoboChart, etc. (beware of acceptability)

**No Model...** Frama-C, Spark, Isabelle/HOL



# V&V models: Different situations over a complete autonomous system

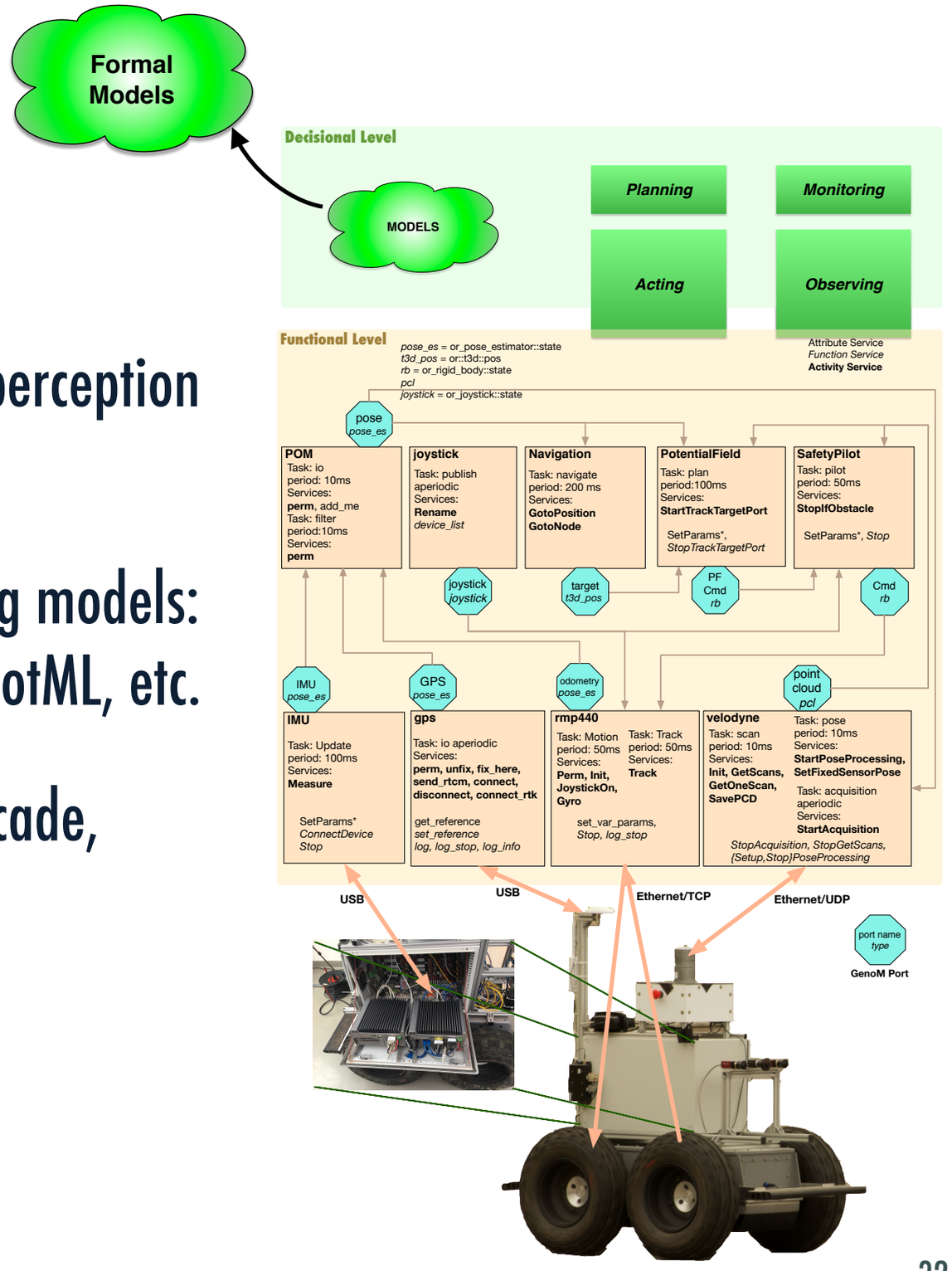
**Formal models:** decisional : planning (e.g. UPPAAL, model checking), monitoring, FDIR (e.g. RMPL), observing (e.g. DyKnow)

**Learned models:** Reinforcement learning models, perception models, action models, etc.

**Specification models:** Robotics software engineering models: e.g. GenoM, Oroccos, MAUVE, BIC, SmartMDSD, RobotML, etc.

**Programming directly the Model:** Orccad, Esterel, Scade, RoboChart, etc. (beware of acceptability)

**No Model...** Frama-C, Spark, Isabelle/HOL



# V&V models: Different situations over a complete autonomous system

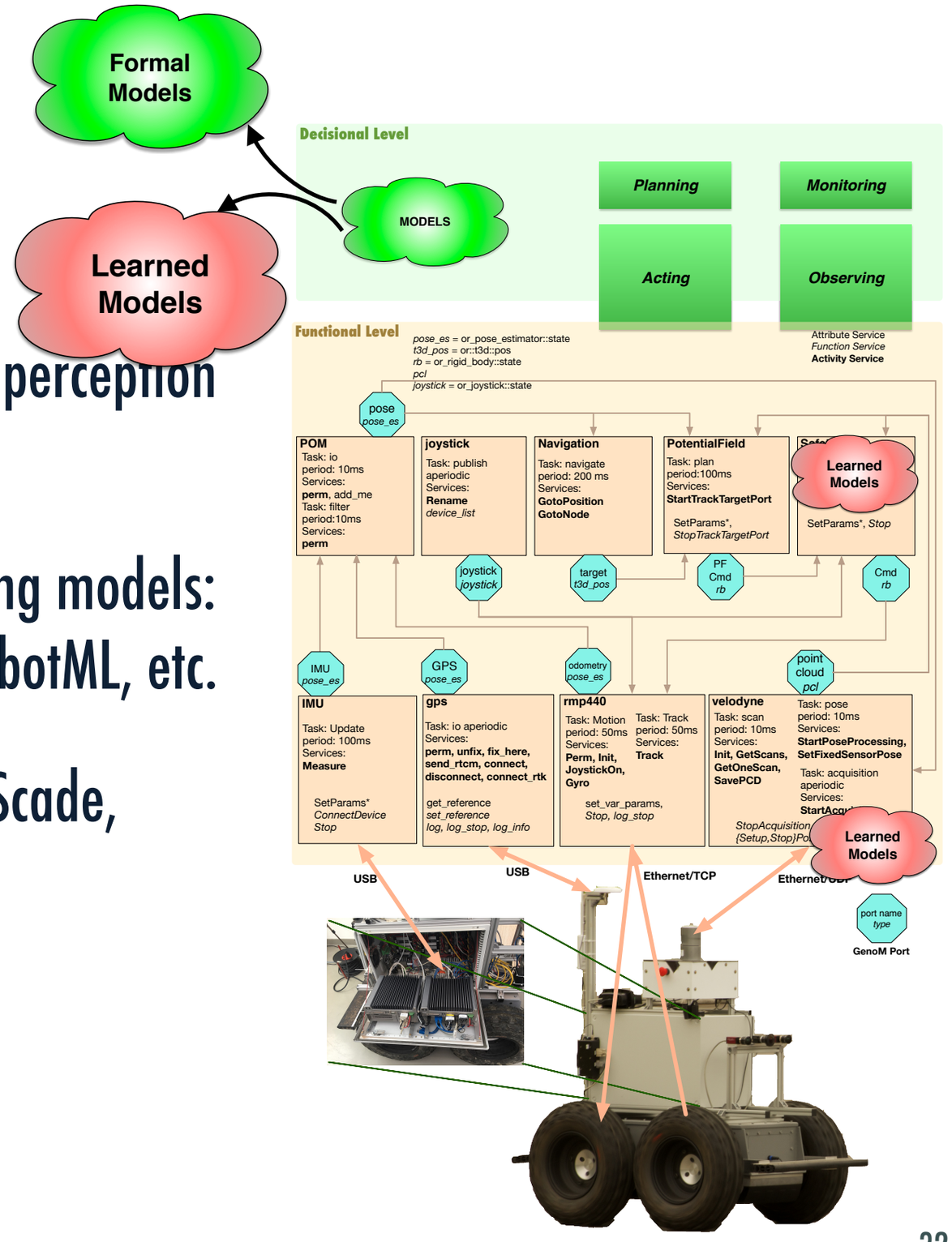
**Formal models:** decisional : planning (e.g. UPPAAL, model checking), monitoring, FDIR (e.g. RMPL), observing (e.g. DyKnow)

**Learned models:** Reinforcement learning models, perception models, action models, etc.

**Specification models:** Robotics software engineering models: e.g. GenoM, Oroccos, MAUVE, BIC, SmartMDSD, RobotML, etc.

**Programming directly the Model:** Orccad, Esterel, Scade, RoboChart, etc. (beware of acceptability)

**No Model...** Frama-C, Spark, Isabelle/HOL



# V&V models: Different situations over a complete autonomous system

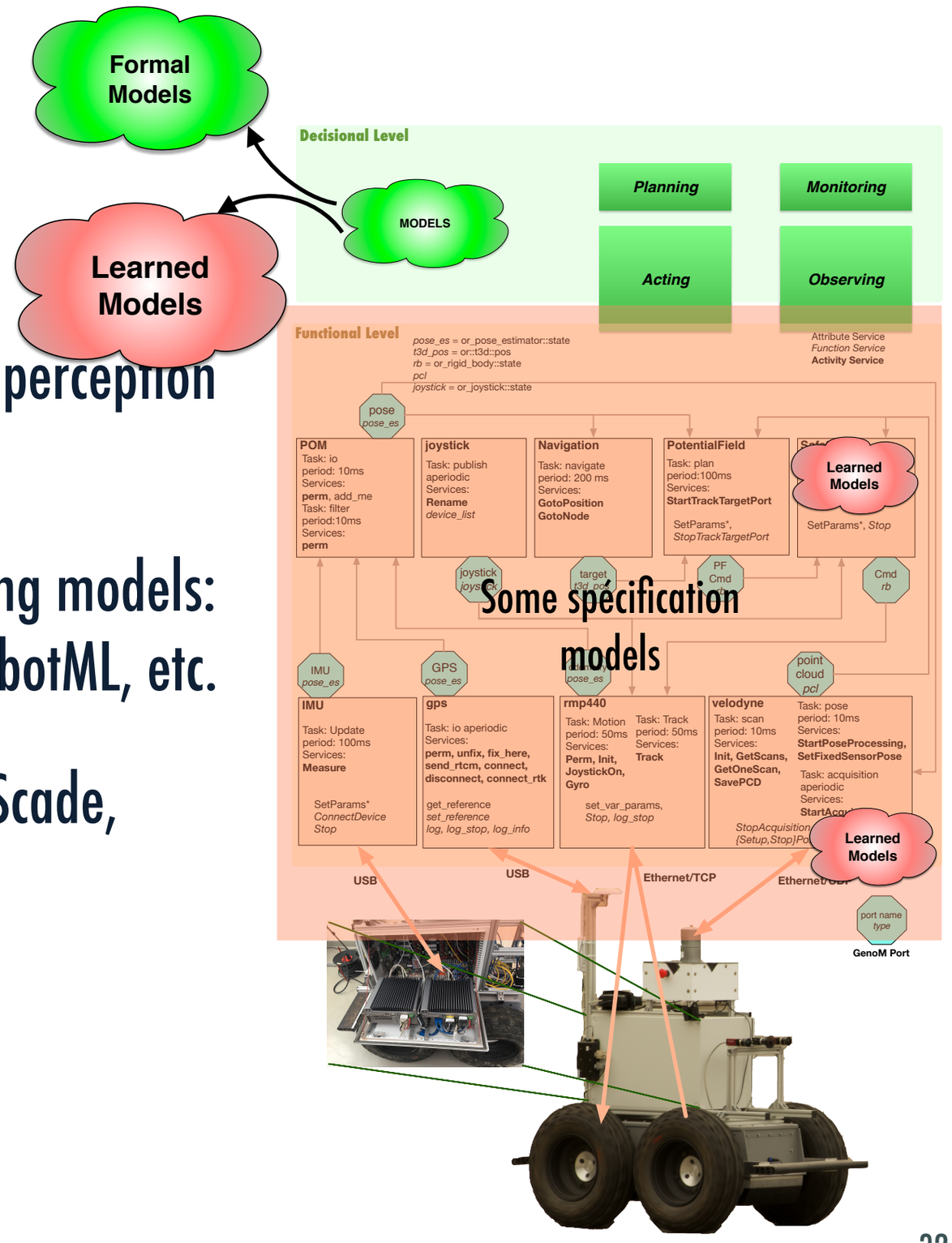
**Formal models:** decisional : planning (e.g. UPPAAL, model checking), monitoring, FDIR (e.g. RMPL), observing (e.g. DyKnow)

**Learned models:** Reinforcement learning models, perception models, action models, etc.

**Specification models:** Robotics software engineering models: e.g. GenoM, Oroccos, MAUVE, BIC, SmartMDSD, RobotML, etc.

**Programming directly the Model:** Orccad, Esterel, Scade, RoboChart, etc. (beware of acceptability)

**No Model...** Frama-C, Spark, Isabelle/HOL



# V&V models: Different situations over a complete autonomous system

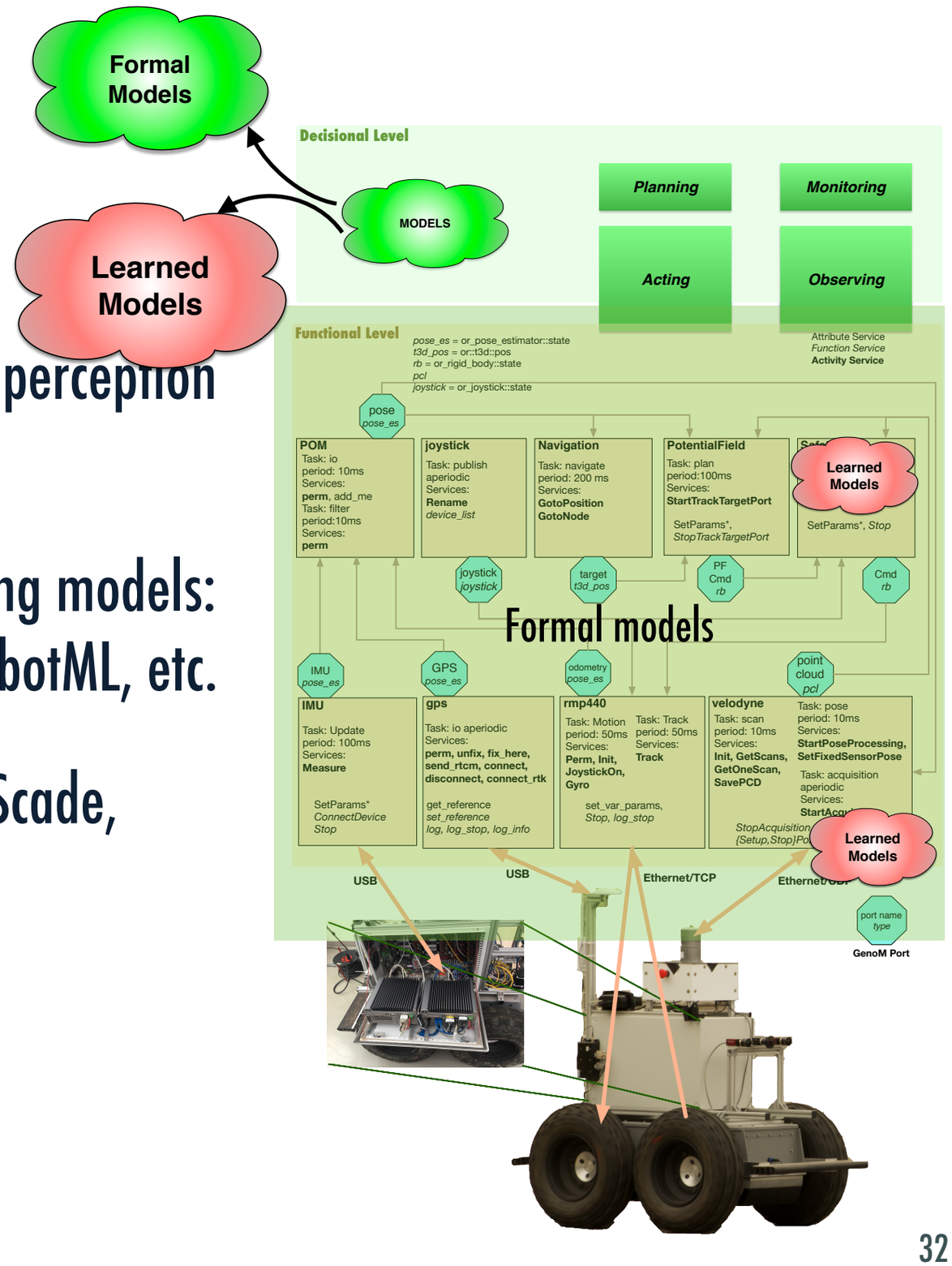
**Formal models:** decisional : planning (e.g. UPPAAL, model checking), monitoring, FDIR (e.g. RMPL), observing (e.g. DyKnow)

**Learned models:** Reinforcement learning models, perception models, action models, etc.

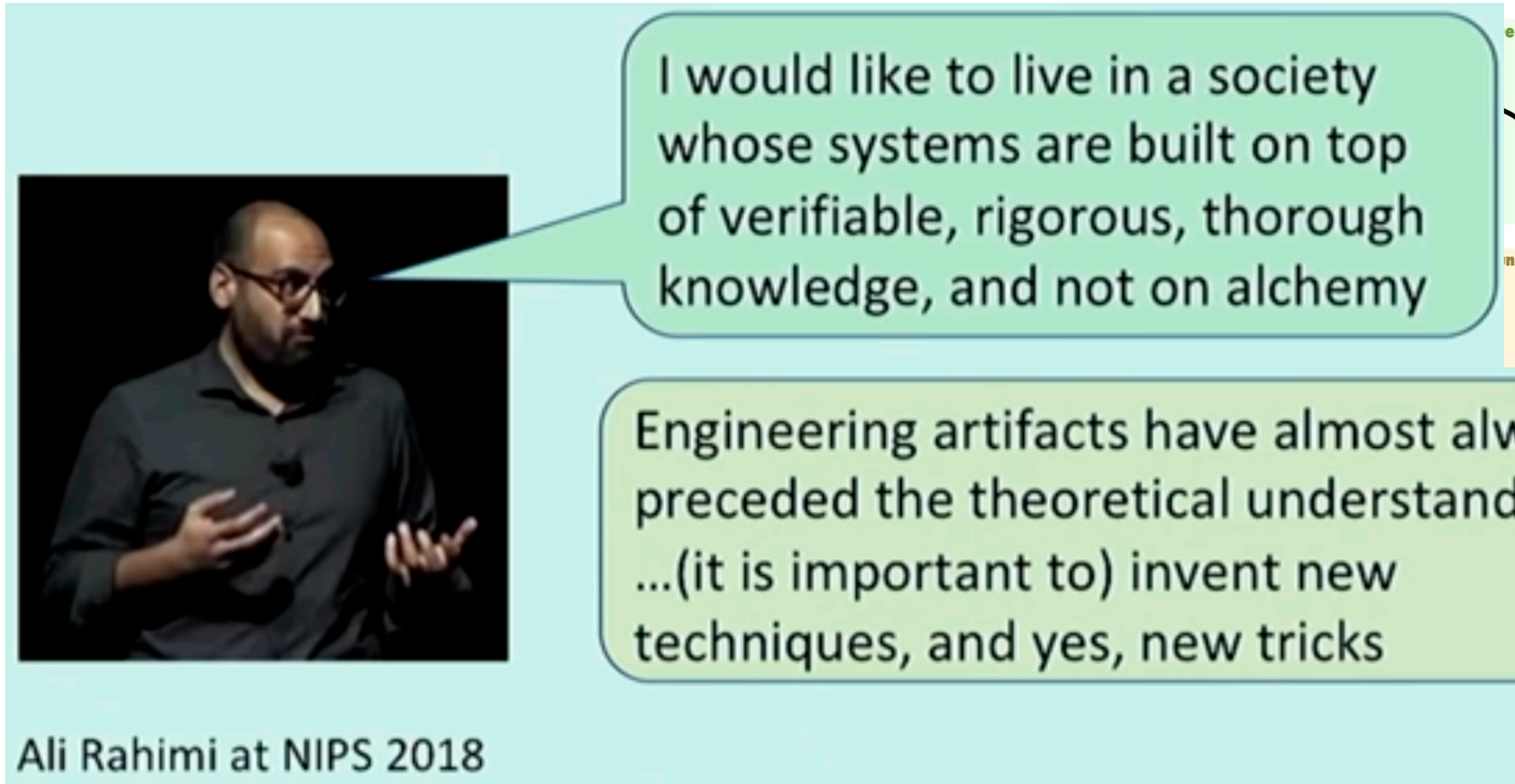
**Specification models:** Robotics software engineering models: e.g. GenoM, Oroccos, MAUVE, BIC, SmartMDSD, RobotML, etc.

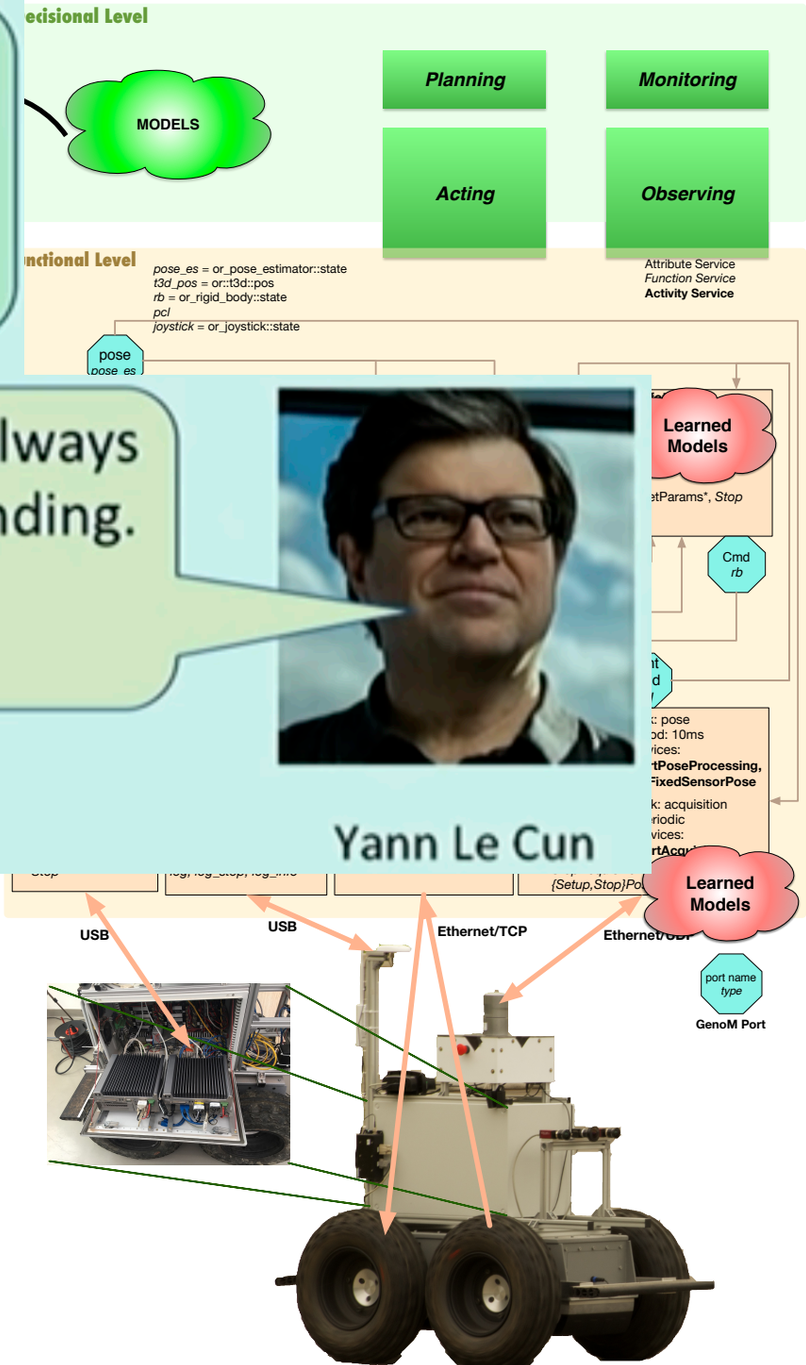
**Programming directly the Model:** Orccad, Esterel, Scade, RoboChart, etc. (beware of acceptability)

**No Model...** Frama-C, Spark, Isabelle/HOL



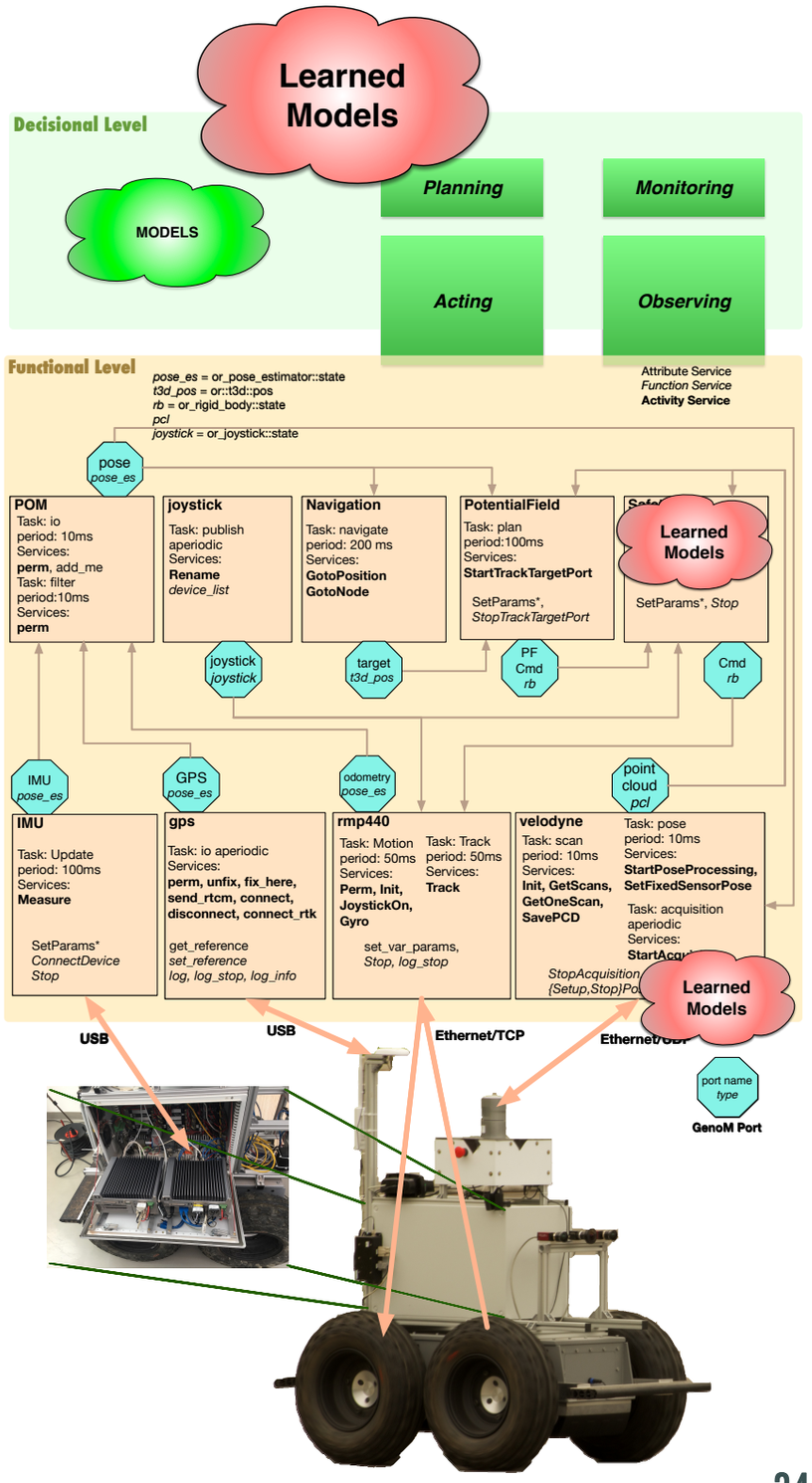
# V&V of learned models...



# V&V of learned models…

- Machine learning is the new AI… but is "model-less"
- Proper environment modeling
- Properly characterize the bound of the learned model
- Learn error models but stick with the "regular" approach (easier to certify)
- Use multiple sources to improve the confidence (sensor results fusion)
- Consistency checking over different information channels
- Safety bag around these components (run time verification)





[1] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, "Concrete Problems in AI Safety," arXiv.org, 1606.06565v2, vol. cs.AI. 21-Jun-2016. http://arxiv.org/abs/1606.06565v2

[2] S. A. Seshia, D. Sadigh, and S. S. Sastry, "Towards Verified Artificial Intelligence," arXiv.org, 1606.08514v3 vol. cs.AI. 28-Jun-2016. http://arxiv.org/abs/1606.08514v3

## Functional Level

*pose_es* = or_pose_estimator::state
*t3d_pos* = or::t3d::pos
*rb* = or_rigid_body::state
*pcl*
*joystick* = or_joystick::state

Attribute Service
*Function Service*
**Activity Service**

pose *pose_es*

**POM**
Task: io
period: 10ms
Services:
**perm**, add_me
Task: filter
period:10ms
Services:
**perm**

**joystick**
Task: publish
aperiodic
Services:
**Rename**
*device_list*

**Navigation**
Task: navigate
period: 200 ms
Services:
**GotoPosition**
**GotoNode**

**PotentialField**
Task: plan
period:100ms
Services:
**StartTrackTargetPort**
SetParams*,
*StopTrackTargetPort*

**SafetyPilot**
Task: pilot
period: 50ms
Services:
**StopIfObstacle**
SetParams*, *Stop*

joystick *joystick* | target *t3d_pos* | PF Cmd *rb* | Cmd *rb*

IMU *pose_es* | GPS *pose_es* | odometry *pose_es* | point cloud *pcl*

**IMU**
Task: Update
period: 100ms
Services:
**Measure**
SetParams*
*ConnectDevice*
*Stop*

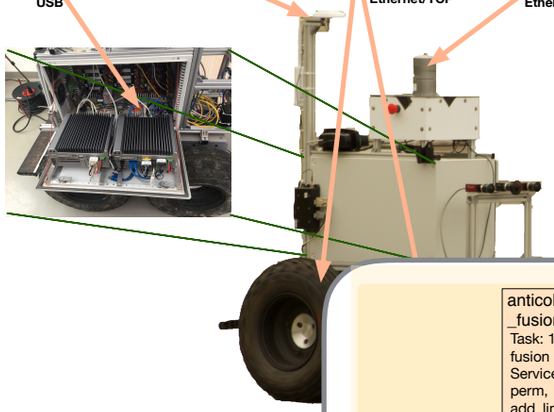
**gps**
Task: io aperiodic
Services:
**perm, unfix, fix_here, send_rtcm, connect, disconnect, connect_rtk**
get_reference
*set_reference*
*log, log_stop, log_info*

**rmp440**
Task: Motion period: 50ms
Services:
**Perm, Init, JoystickOn, Gyro**
Task: Track period: 50ms
Services:
**Track**
set_var_params,
*Stop, log_stop*

**velodyne**
Task: scan
period: 10ms
Services:
**Init, GetScans, GetOneScan, SavePCD**
Task: pose
period: 10ms
Services:
**StartPoseProcessing, SetFixedSensorPose**
Task: acquisition
aperiodic
Services:
**StartAcquisition**
*StopAcquisition, StopGetScans, {Setup,Stop}PoseProcessing*

USB | USB | Ethernet/TCP | Ethernet/UDP

port name *type*
**GenoM Port**



## Functional Level

door_ pose | furious_ markers | furious _loc

**LWRI**
Task: high 50ms
Services:
**permanent**, **sendMotion, receiveControlDatas, updateControlParams, GotoQ, GotoQRel, GotoXXAbs, GotoXXRel, QViaPoints, CartViaPoints, RecordViaPoint, openDoor**
GetQ, GetX,... getCartesianTorques,
*SetSpeedLimits, ...*
*setTheoricalPositionAdjust*

**ODS**
Task: cycle 60ms
Services:
**permanent, openDoor**
Task: baseUpdate 40ms
Services:
**permanent**,
Task: lwrUpdate 20ms
Services:
**permanent**,
Task: localize 40ms
Services:
**permanent**,
*setMode, homing, killODS*

motion | control Params | port_init | control Datas | ods_Datas | base_init | rf_poster

**LWRC**
Task: motion 10ms
Services:
**permanent, Init,**
*ReceiveMotion, sendControlDatas,*
setDebugMode

**RF**
Task: cycle 50ms
Services:
**permanent, RF, Calibrate, MoveTo, MoveSpineOnly, sendDatas**
SetLinearSpeedLimit,
SetLinearAccLimit, ...,
GetOdo

USB Serial Line/ Ethernet | Socket CAN Interface

Attribute Service
*Function Service*
**Activity Service**

GenoM

## Functional Level

**RWLSensor**
Task:
sense 200 ms
Services:
**CheckLight**
**ResetLightsSeq**
**UnstowSensor**
**StowSensor**

LightLvl | Lights

**Navigation**
Task:
navigate 200 ms
Services:
**GotoPosition**
**GotoNode**
Stop

**Teleop**
Task:
check 100ms
Services:
**Op**
*Stop*

Joystick | Target | Teleop Cmd | Nav graph | Pose | PFCmd

**POM**
Task: io 10ms
Services:
**perm, add_me**
Task: filter 10ms
Services:
**Perm**

**PotentialField**
Task:
plan 100ms
Services:
SetParams
**TrackTarget**
*Stop*

**SafetyPilot**
Task:
pilot 40ms
Services:
SetParams*
*Stop*
**MergeAndAvoid**

Cmd

GPS | IMU | Odometry (speed) | Laser

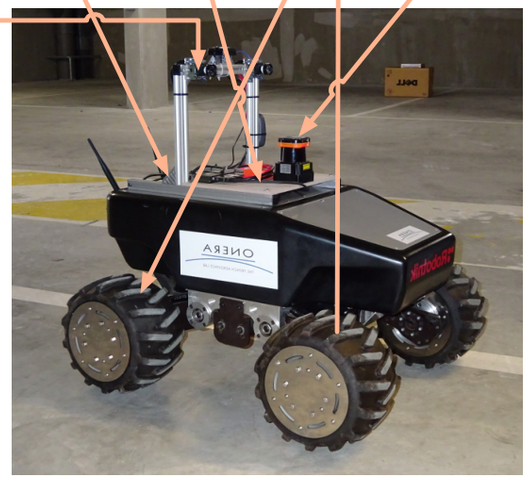
**GPSDriver**
Task:
update 100ms
Services:
SetParams*
*ConnectDevice*
**Measure**
*Stop*

**IMUDriver**
Task:
update 10ms
Services:
SetParams*
*ConnectDevice*
**Measure**
*Stop*

**RobotDriver**
Task: rosTOgenom
Task: genomToros 40ms
Services:
**Perm**
Services:
**Perm**

**LaserDriver**
Task:
scan 100ms
Services:
SetParams*
*ConnectDevice*
**StartScan**
*Stop*

Attribute
*Function*
**Activity**



## GenoM3 Components

**anticollision _fusion**
Task: 10ms
fusion
Services:
perm,
add_limiter

speed_ limit | feedback

**follower**
Task: 20ms
follow
Services:
follow_path
stop

*x2 front and back*

**CC- straight**
Task: 40ms
check
Services:
perm

speed_ limit | speed_ limit

**CC- on_path**
Task: 40ms
check
Services:
perm

real_path | command | odometry | laser_ scan

**lms_1xx**
Task: 40ms
scan
Services:
acquire

**low_level**
Task: ap
vehicle_state
Services:
perm
connect_can
Task: 20ms
command
Services:
perm
openDoors
closeDoors

*pose_es* = or_pose_estimator::state
rotor_input = sequence<double, max_rotors>
rotor_measures = sequence<rotor_state, max_rotors>

**pom**
Task: io
period: 1ms
Services:
**perm**,
add_me
Task: filter
period:1ms
Services:
**perm**

**maneuver**
Task:
plan ap
Services:
**Goto**
**WayPoint**
**TakeOff**
Task: exec
5ms
Services:
**perm**

state *pose_es* | desired state *pose_es*

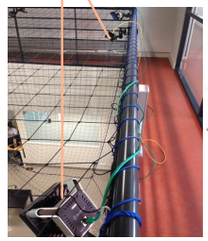
mocap *pose_es*

**optitrack**
Task:
publish ap
Services:
**perm**
**connect**

IMU *pose_es*

**rotorcraft**
Task:
main 1ms
Services:
**perm**
**calibrate_imu**
**start**
**servo**
**stop**
Task:
comm ap
Services:
**perm**
**connect**
**monitor**

actual velocity *rotor_meas ure*



port name *type*
**GenoM Port**

LAAS CNRS

# Conclusion

- Tools must be “usable” and “adopted” by roboticists
- When there are models... there is hope!
  - Adapt the models and the V&V techniques
- Try to keep the overall consistency
- AI/decisional components are mostly OK (wrt formal V&V)
- V&V and certification of “learned model” based components remain a challenge

LAAS CNRS

# Research agenda

- If we want to stay in the loop, we should stop looking at toy robots/problems... NHTSA now allows testing with cars without steering wheels... (level 4)
- Deeper model (codel arguments, SDI, algo, check the codel, etc) & Run Time Verification
- Better linked models between functional level and decisional level (Planning/Acting/Monitoring)
- Address V&V of learned models
- HAVV : Human Aware V&V (Human in the loop: uncontrollable model)

LAAS CNRS



# All the code is open source and available... please use it with your own framework

## https://redmine.laas.fr/projects/minnie

Thanks to
Verimag: Jacques Combaz, Mohammed Foughali, Saddek Bensalem (BIP)
Mälardalen University: Cristina Seceleanu (UPPAAL-SMC)
LAAS: Anthony Mallet (GenoM), Bernard Berthomieu (Fiacre/TINA),
Silvano Dal Zilio (Fiacre/TINA), Pierre Emanuel Hladik (Fiacre/Hippo)

LAAS CNRS



Part of this work is funded by the H2020 European project AI4EU

AI4EU