A software toolkit for rapid development of AUVs

using MOOS-IvP with *MITFrontseat*, *HydroMAN* and *VECTORS*



Dr. Supun Randeni

Dr. Michael Menjamin, Prof. Michael Triantafyllou & Prof. Henrik Schmidt

Massachusetts Institute of Technology, Cambridge, MA

MOOS-DAWG 2022

Laboratory for Autonomous Marine Sensing Systems

MIT Sea Grant AUV laboratory

Massachusetts Institute of Technology

Design aspects of an AUV

Hullform Design

- Nose-cone & tail-cone design
- Shapes, sizes & placement of appendages

Actuator Design

- Actuation style; e.g. control surfaces, thrusters, fins, etc.
- Sizes & placement of actuators

Electronics Design

- Sensor selection
- Processing board selection; e.g. PC104, BeagleBoard, Raspberry Pi, etc.
- Designing I/O boards (if required)
- Electronics/power breakout board design
- Designing actuator driving & power management electronics

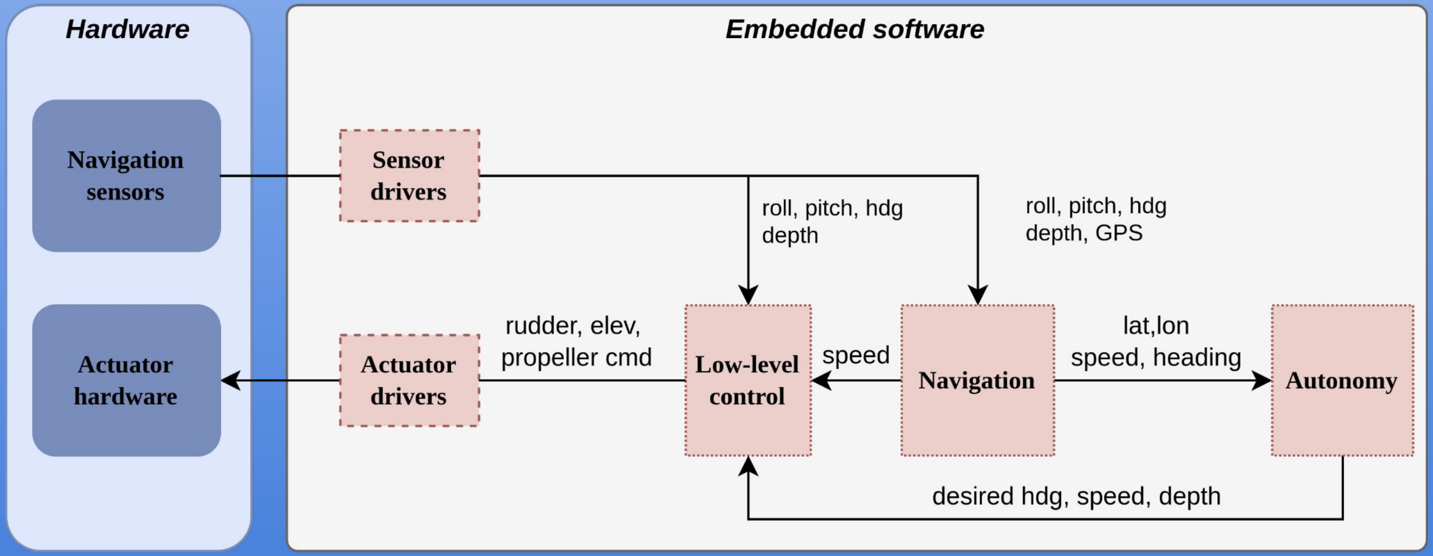
Integration design

- Internal component general arrangement (GA) design
- Watertight bottle design
- Bulkhead connector arrangement
- Hydrostatic ballasting method
- Structural design with room for extendability

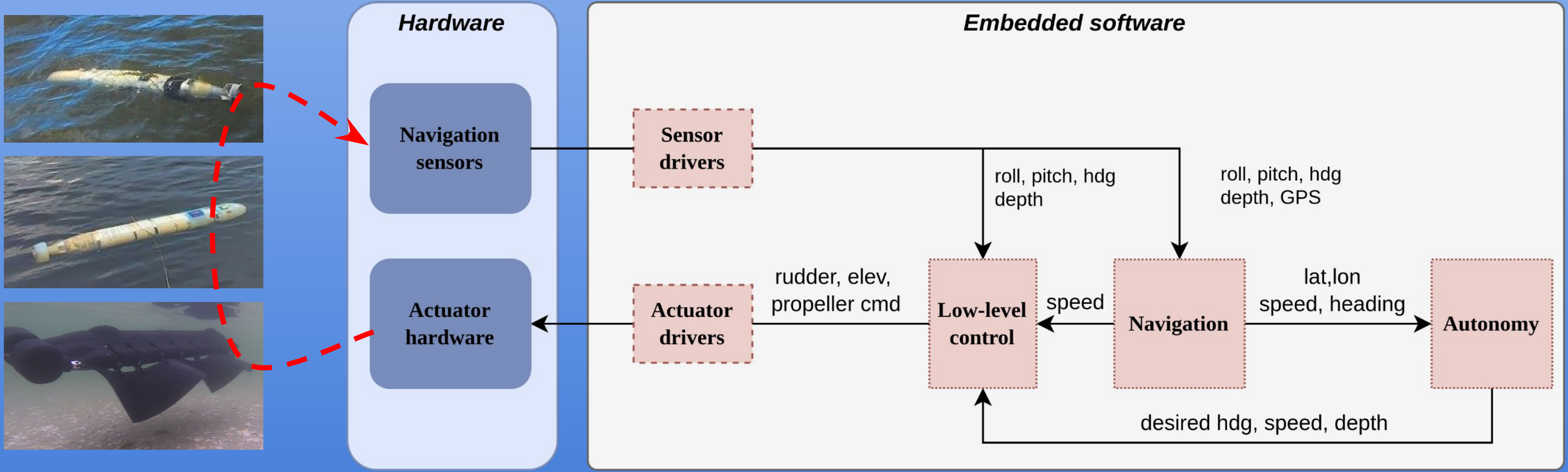
Software Design

- Selecting a middleware
- Sensor & actuator driver design
- Navigation software design
- Autonomy software design
- Low-level control software design
- Mission & safety management software design
- Communication software design

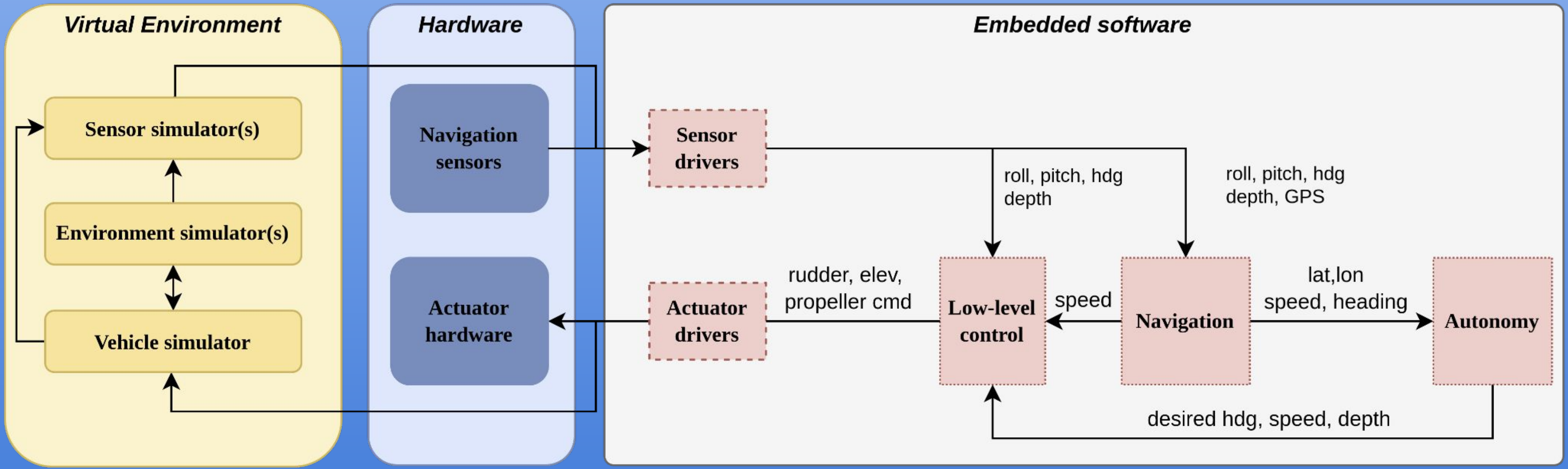
Typical software architecture of a UUV (a high-level overview)



Typical software architecture of a UUV
(a high-level overview)

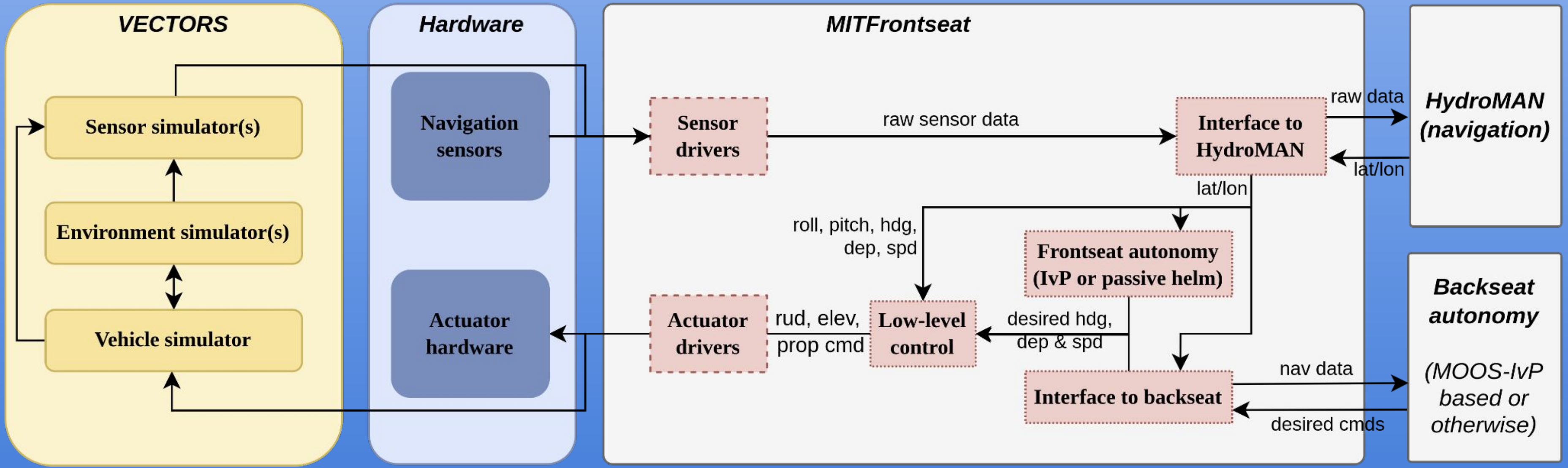


Virtual environment



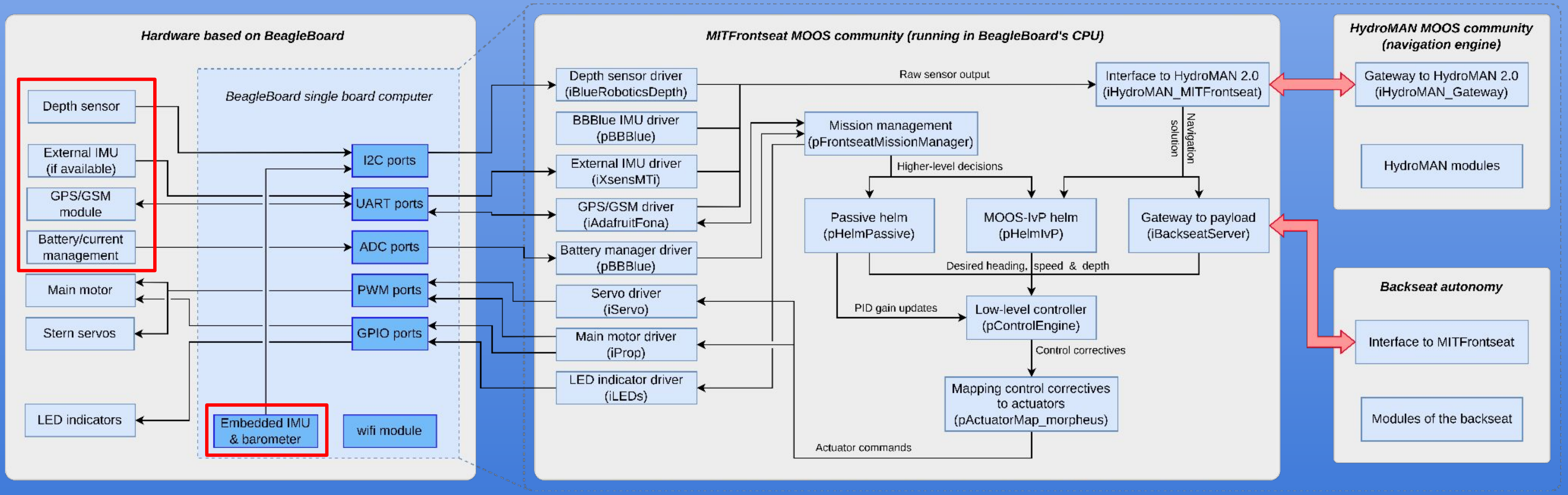
MITFrontseat, *HydroMAN* and *VECTORS* with MOOS-IvP

- **MITFrontseat** - Frontseat software of the vehicle
- **HydroMAN** **-** Self-leaning, vehicle flight dynamic model-aided navigation engine
- **VECTORS** **-** Virtual environment for construction and testing of oceanic robotics systems



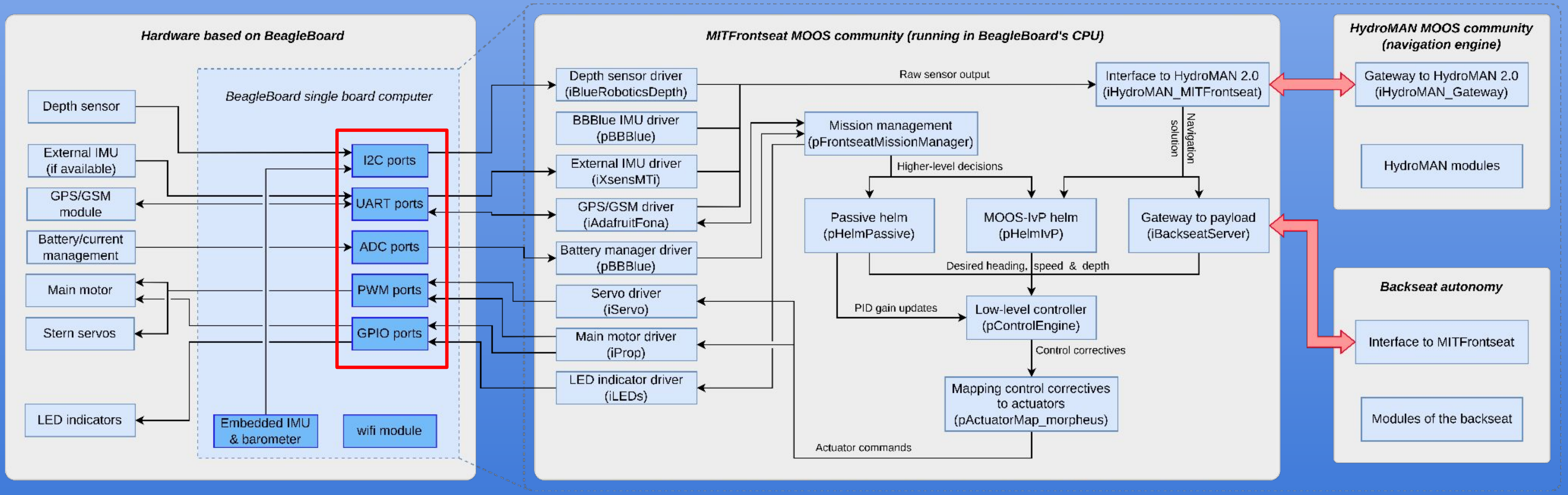
Architecture of *MITFrontseat* (sensor drivers)

- **Navigation sensors**: Depth, IMU, GPS, GSM, battery/current management, embedded IMU (for BBBlue), Barometer



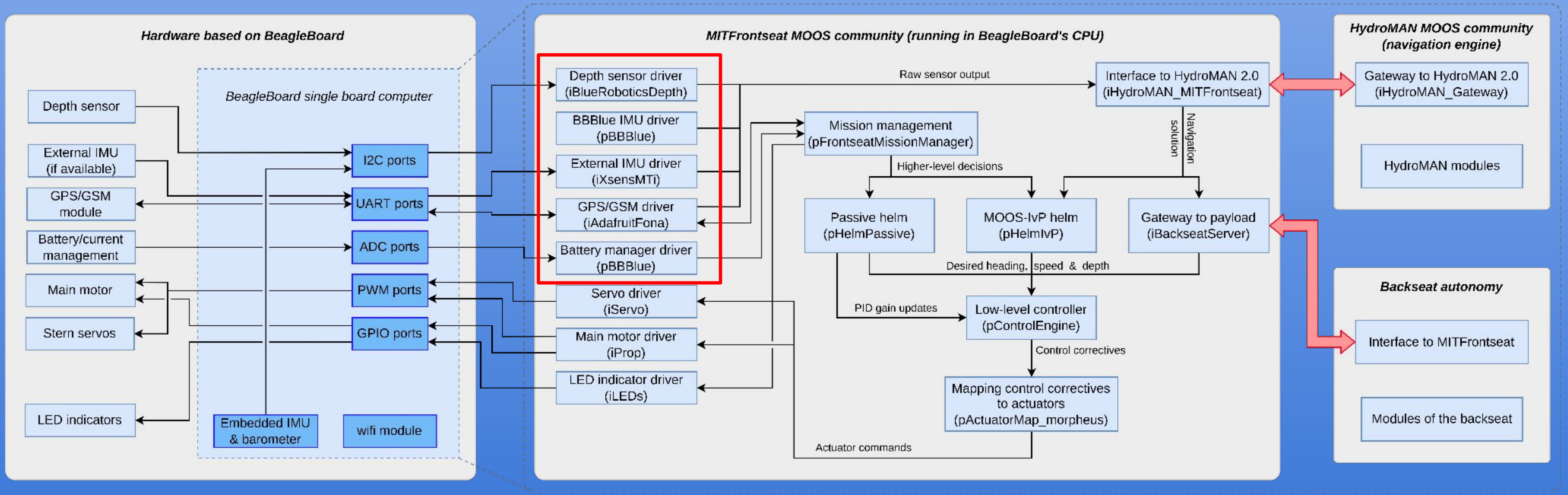
Architecture of *MITFrontseat* (sensor drivers)

- **Navigation sensors**: Depth, IMU, GPS, GSM, battery/current management, embedded IMU (for BBBlue), Barometer
- **Hardware interfaces**: I2C, UART, ADC, PWM, GPIO



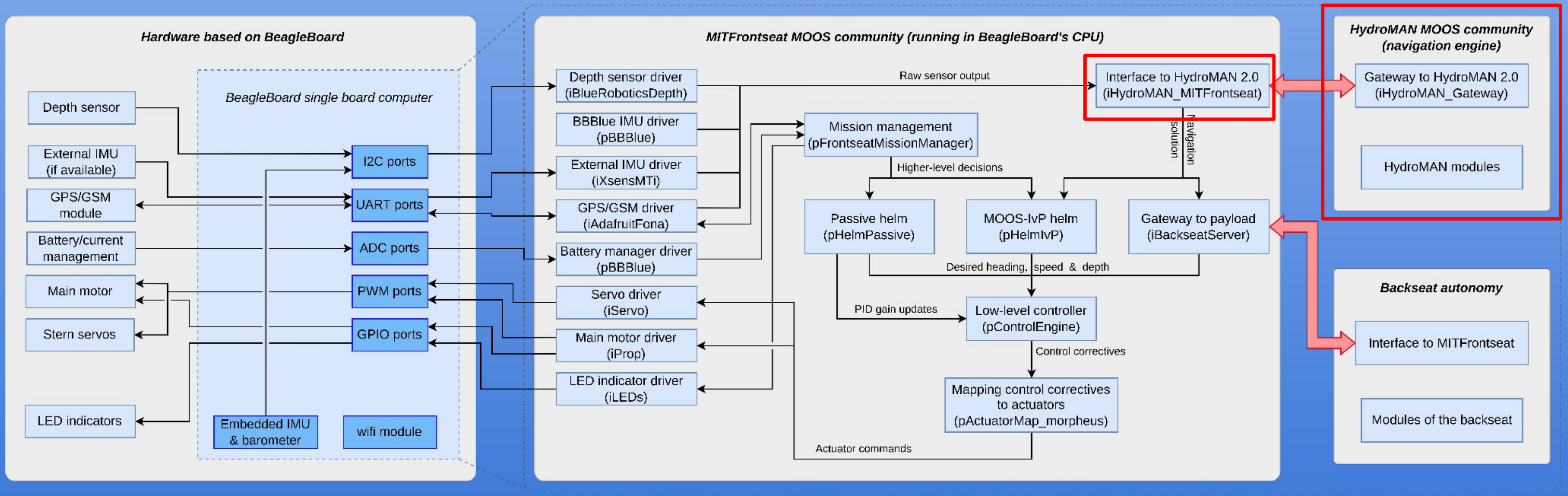
Architecture of *MITFrontseat* (sensor drivers)

- **Navigation sensors**: Depth, IMU, GPS, GSM, battery/current management, embedded IMU (for BBBlue), barometer
- **Hardware interfaces**: I2C, UART, ADC, PWM, GPIO
- **Sensor drivers:** Communicates with sensors and publishes raw sensor data to the MOOSDB

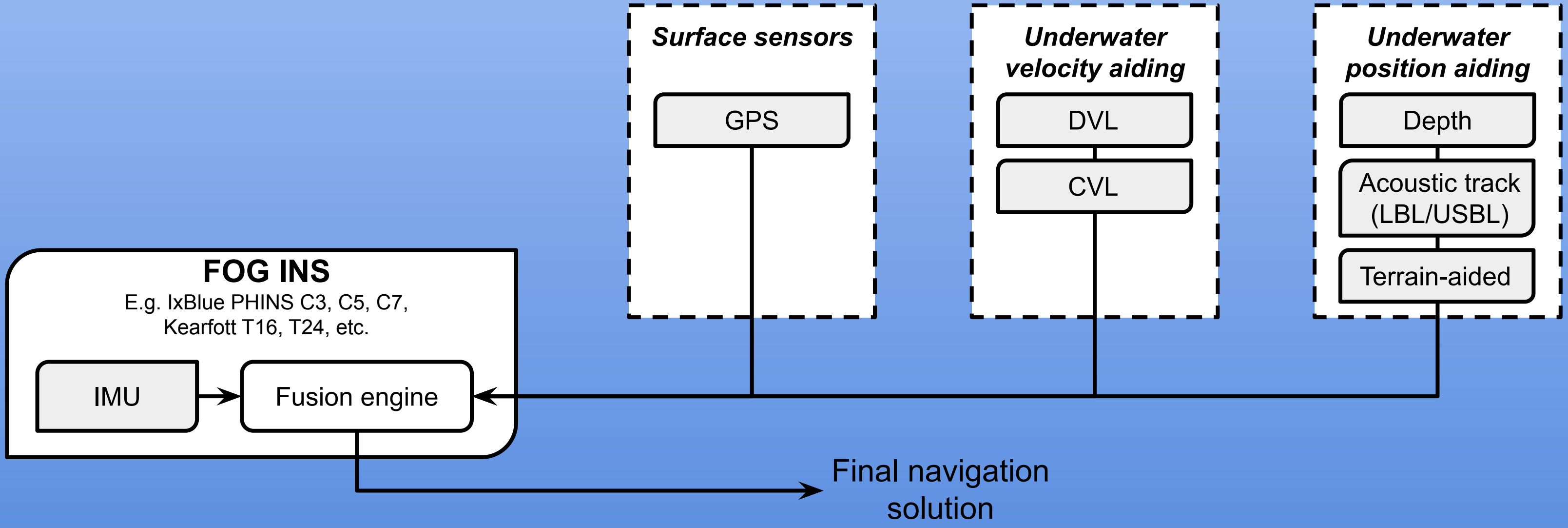


Architecture of *MITFrontseat* (navigation)

- **Navigation sensors**: Depth, IMU, GPS, GSM, battery/current management, embedded IMU (for BBBlue), barometer
- **Hardware interfaces**: I2C, UART, ADC, PWM, GPIO
- **Sensor drivers**: Communicates with sensors and publishes raw sensor data to the MOOSDB
- **HydroMAN interface**: Passes raw sensor data to the HydroMAN, and receives the final navigation solution in return.
- **HydroMAN**: A self-learning vehicle flight dynamic model aided navigation system



Conventional INS-aided navigation



While underwater - dead-reckoning with velocity aiding

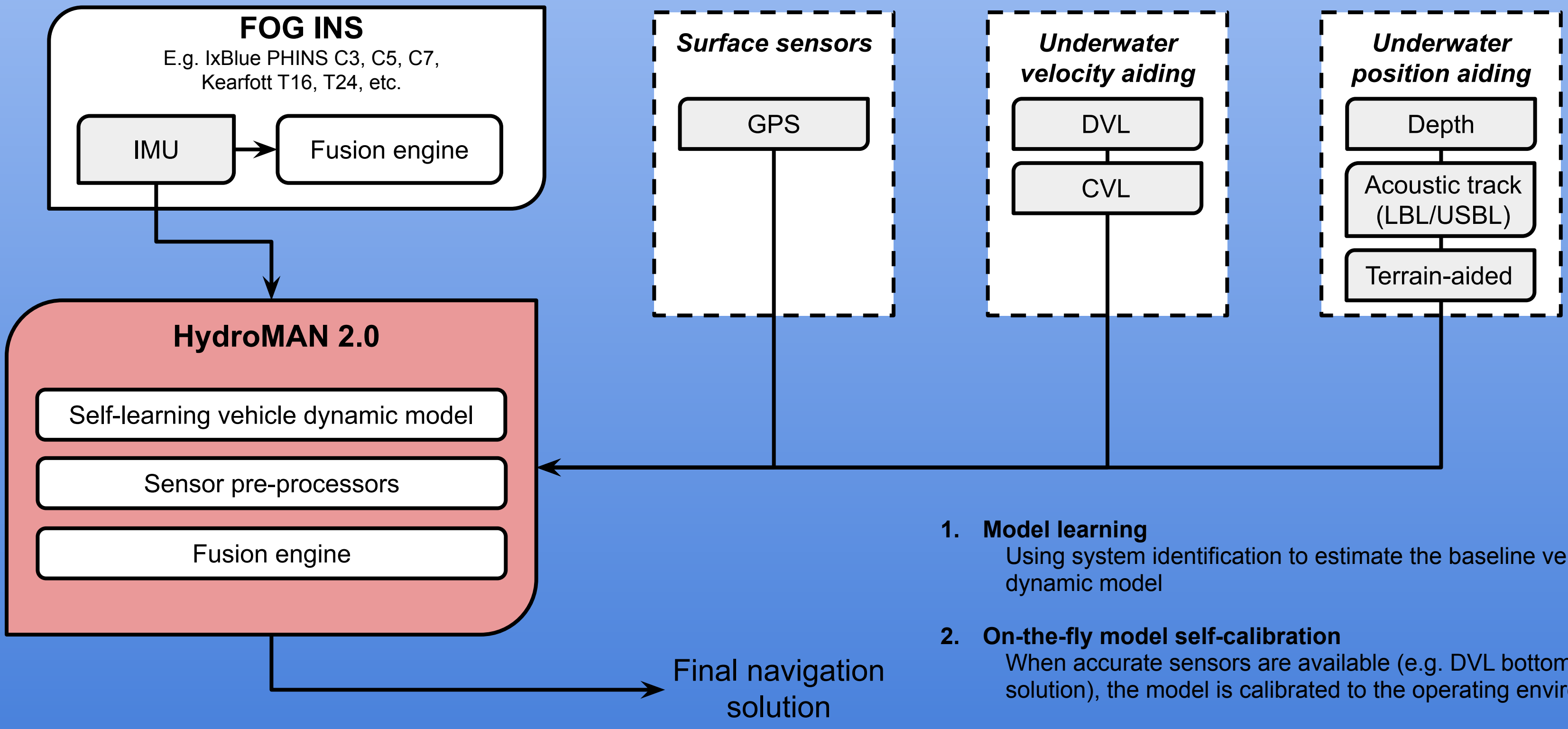
- DVL bottom-track
  - Accurate navigation (i.e. <0.2% - 0.05% DT) when DVL bottom-lock is available
  - Max range: 30 m for 1200 kHz, 200m for 300kHz
  - Power hungry
- CVL
  - Less accurate as compared to DVL
  - Max range: ~300m
  - Power hungry

While underwater - position aiding

- Acoustic positioning (e.g. LBL, USBL, SBL)
  - Potential outages and outliers
  - Time-lags in the position fix
  - Power hungry (specially active acoustic systems)
- Terrain-aided
  - Large uncertainty

HydroMAN Navigation System

A self-learning navigation fusion engine



1. **Model learning**
   Using system identification to estimate the baseline vehicle flight dynamic model

2. **On-the-fly model self-calibration**
   When accurate sensors are available (e.g. DVL bottom-lock, LBL solution), the model is calibrated to the operating environment

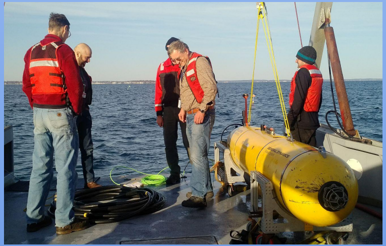
3. **Model aiding for navigation**
   When accurate sensors are unavailable or turned off, the model aids the navigation engine

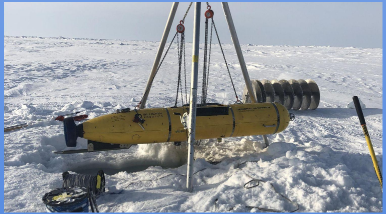
HydroMAN Navigation System

Advantages over conventional INS-aided navigation

1. Improved model-aided navigation for low-cost AUVs with no INS and/or DVL
2. Able to maintain navigation accuracy when DVL bottom track is unavailable
3. Able to switch off navigation sensors to save power
4. Able to effectively use time-lagged acoustic navigation updates
5. Able to limit the vehicle to an IMU


Sandshark model-aided navigation
at MIT sailing pavilion (2018)


Pre-ICEX20 engineering tests
at Mass Bay (2019)


ICEX20 under-ice navigation
at Beaufort Sea, Arctic (2020)


MIT variant of MK-39 EMATT
at MIT sailing pavilion (2020)


Morpheus AUV
at MIT sailing pavilion (2021)


GPS-denied navigation
at MIT sailing pavilion (2021)


(on-going)

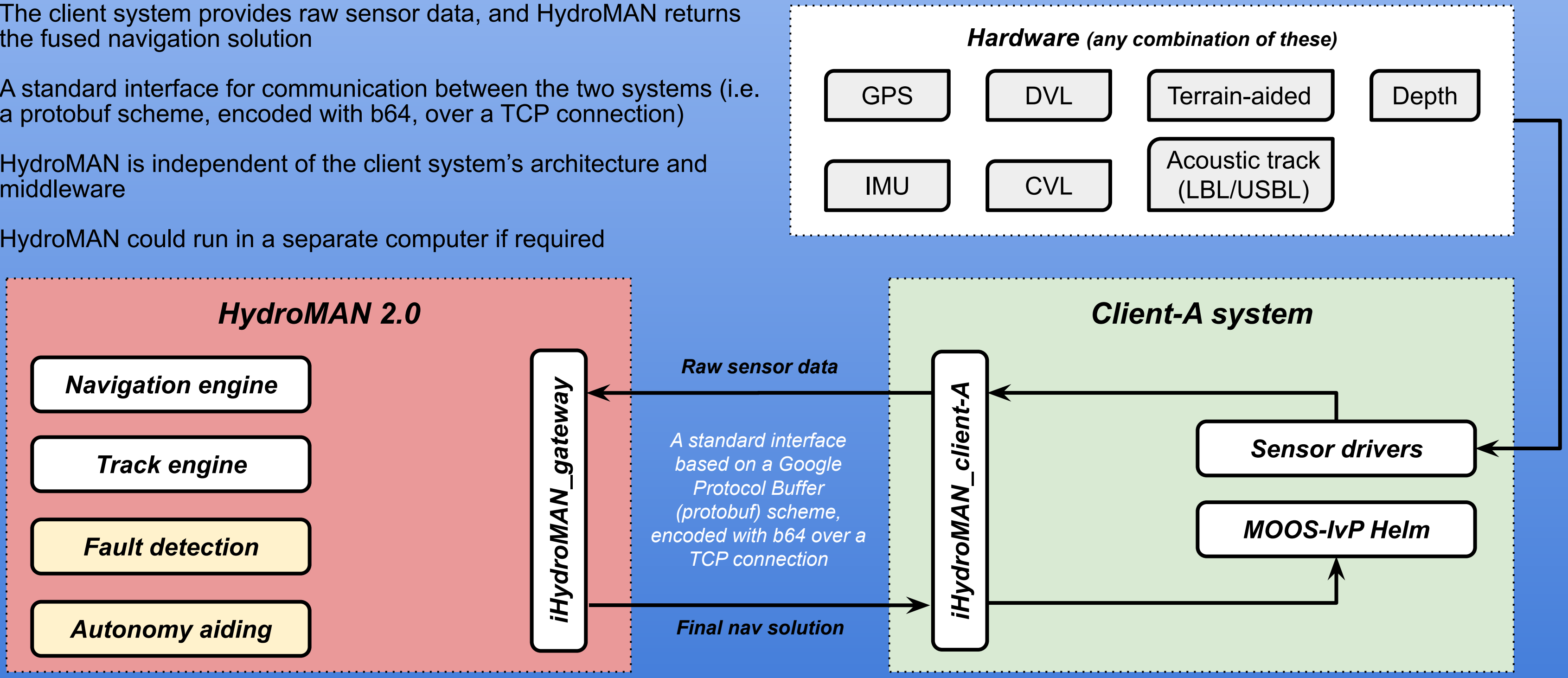
(on-going)

(on-going)

HydroMAN 2.0

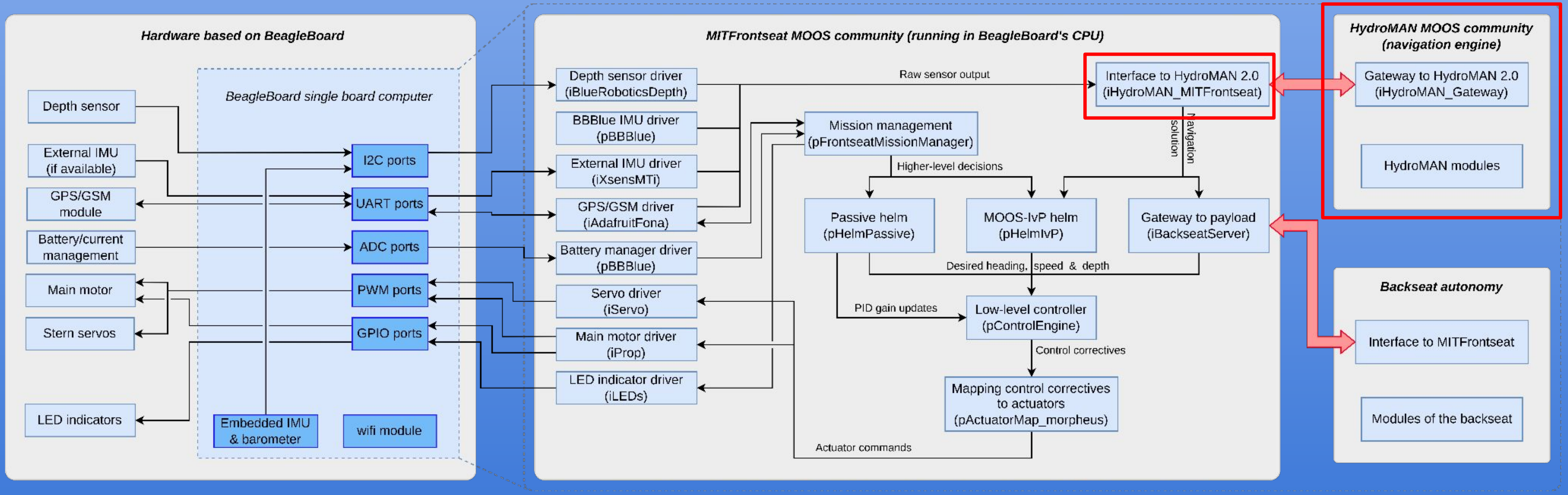
Generalizing HydroMAN as an independent navigation engine

- The client system provides raw sensor data, and HydroMAN returns the fused navigation solution
- A standard interface for communication between the two systems (i.e. a protobuf scheme, encoded with b64, over a TCP connection)
- HydroMAN is independent of the client system's architecture and middleware
- HydroMAN could run in a separate computer if required



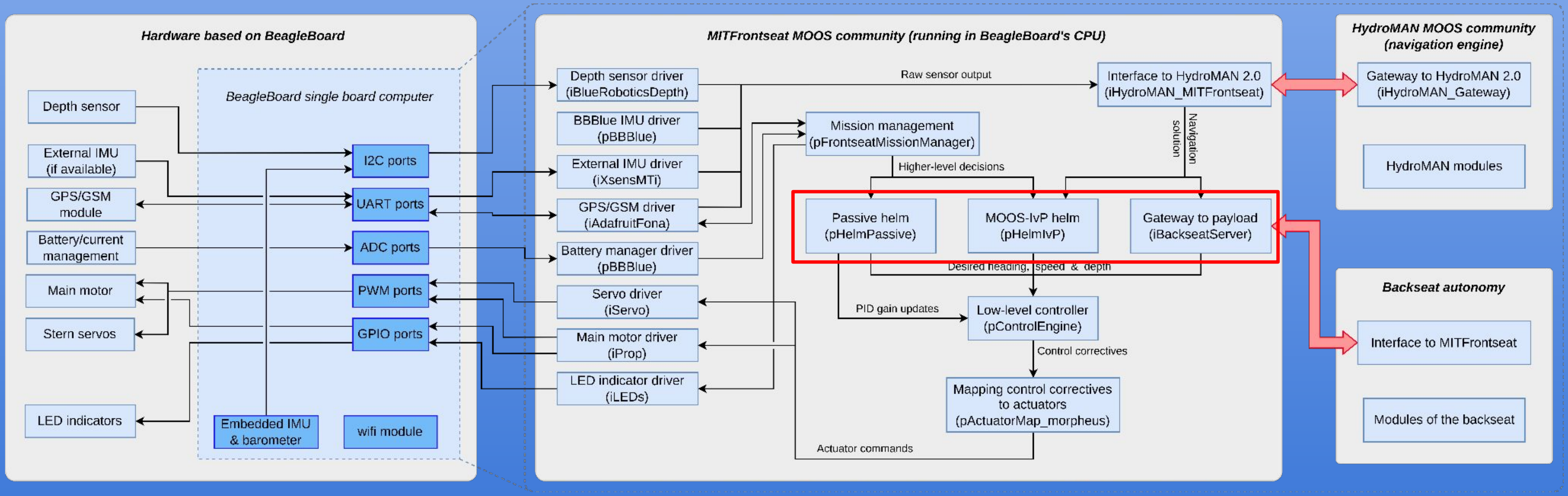
Architecture of *MITFrontseat* (navigation)

- **Navigation sensors**: Depth, IMU, GPS, GSM, battery/current management, embedded IMU (for BBBlue), barometer
- **Hardware interfaces**: I2C, UART, ADC, PWM, GPIO
- **Sensor drivers**: Communicates with sensors and publishes raw sensor data to the MOOSDB
- **HydroMAN interface**: Passes raw sensor data to the HydroMAN, and receives the final navigation solution in return.
- **HydroMAN**: A self-learning vehicle flight dynamic model aided navigation system



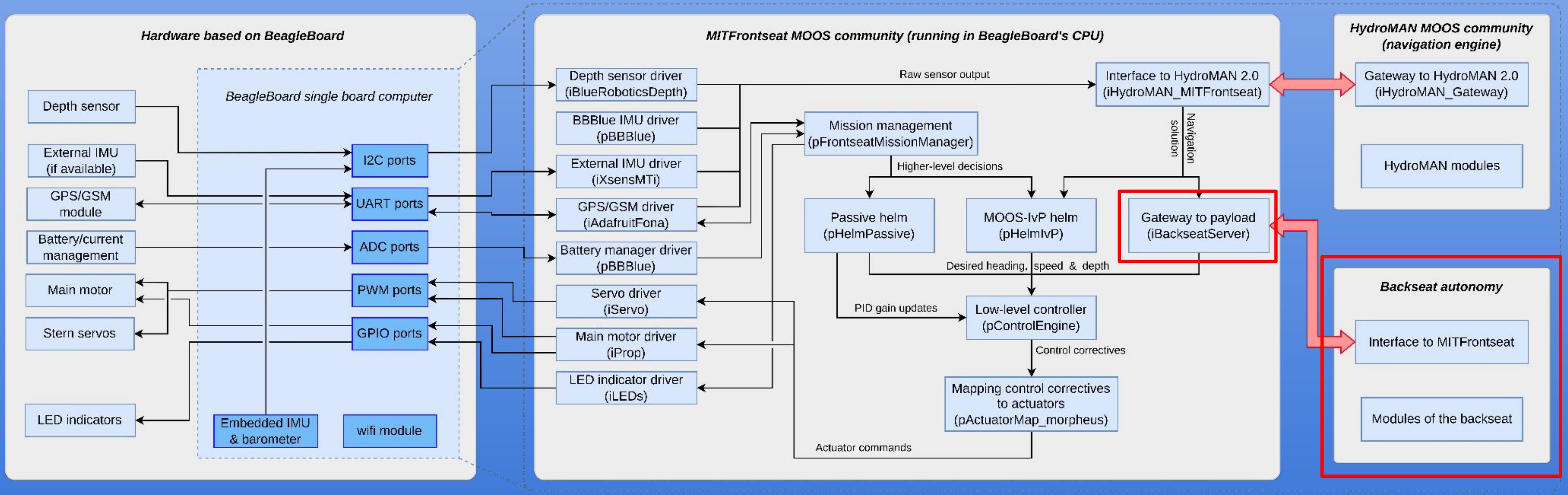
Architecture of *MITFrontseat* (autonomy)

- **Autonomy system**: Produces desired heading, depth and speed



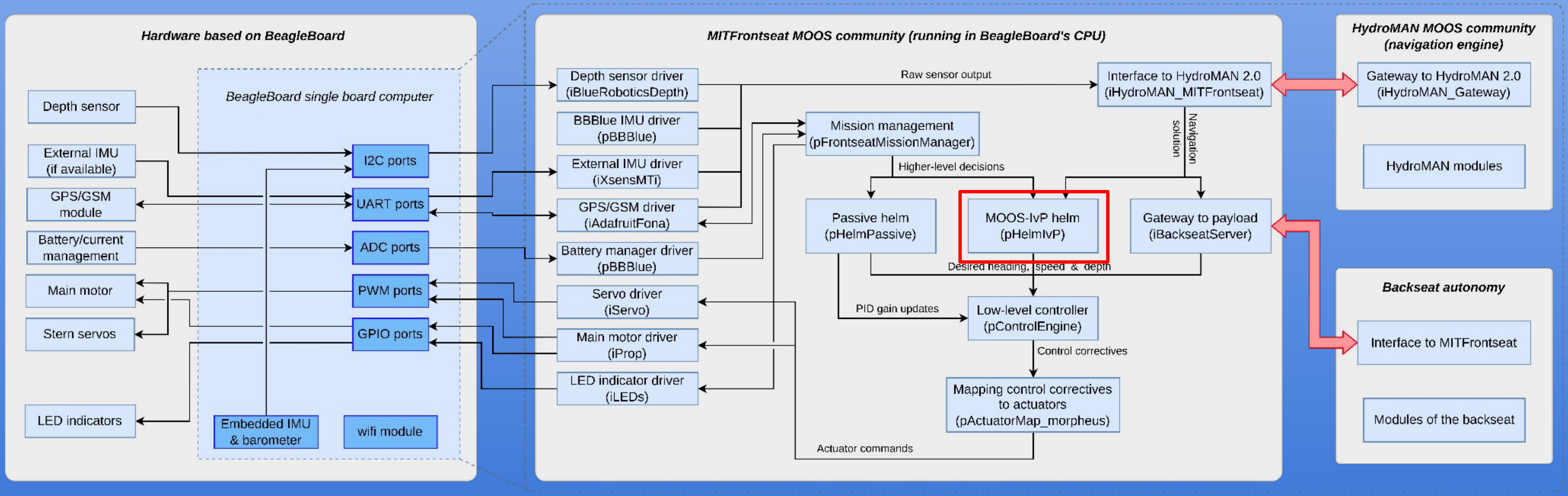
Architecture of *MITFrontseat* (autonomy)

- **Autonomy system**: Produces desired heading, depth and speed
- **Payload autonomy**: Interfaces with a payload autonomy system; a.k.a. backseat driver (MOOS-IvP based or otherwise)



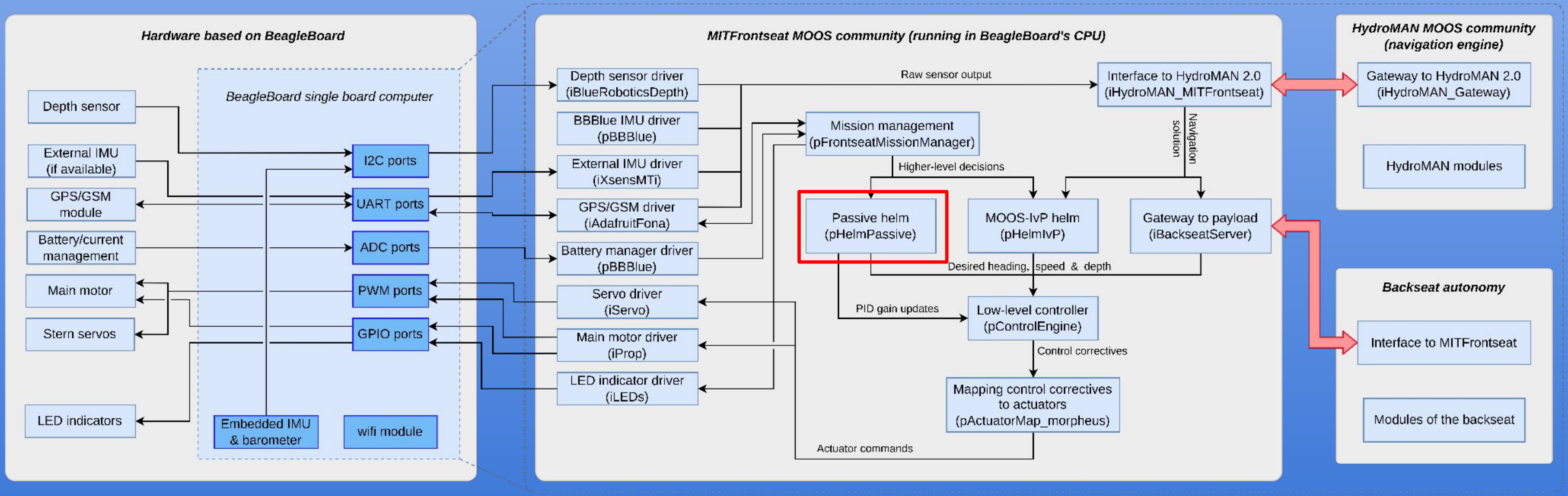
Architecture of *MITFrontseat* (autonomy)

- **Autonomy system**: Produces desired heading, depth and speed
- **Payload autonomy**: Interfaces with a payload autonomy system; a.k.a. backseat driver (MOOS-IvP based or otherwise)
- **`pHelmIvP` on frontseat**: Run the `pHelmIvP` instance inside the MITFrontseat MOOS community. With `pFrontseatMissionManager` watching over the helm.



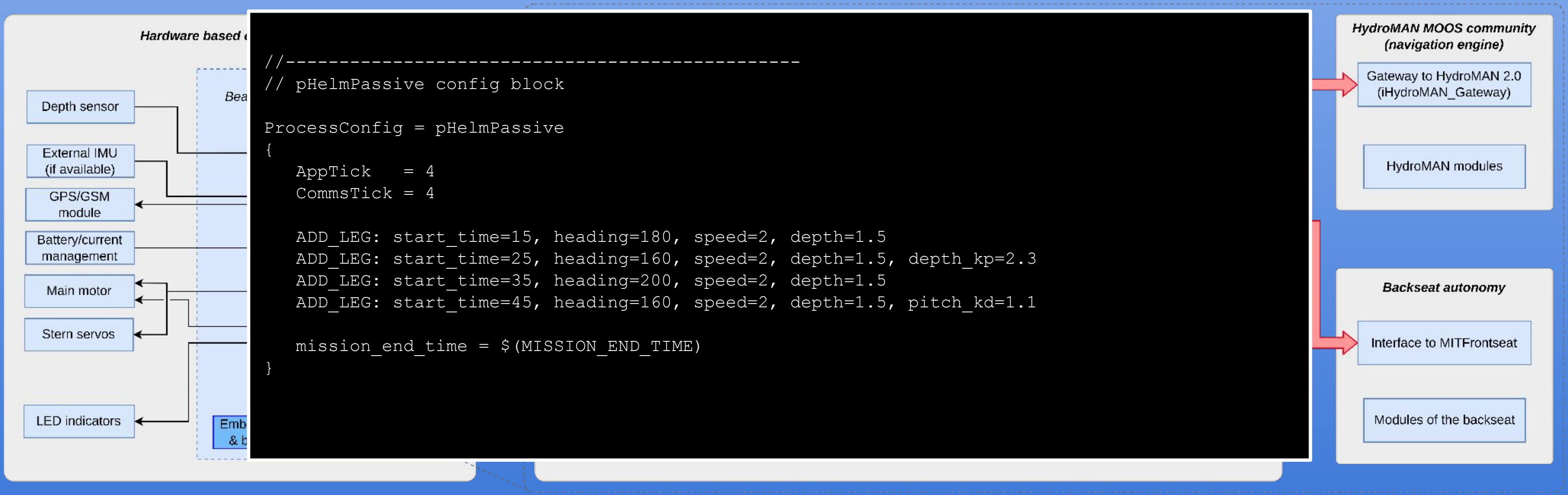
Architecture of *MITFrontseat* (autonomy)

- **Autonomy system**: Produces desired heading, depth and speed
- **Payload autonomy**: Interfaces with a payload autonomy system; a.k.a. backseat driver (MOOS-IvP based or otherwise)
- **`pHelmIvP` on frontseat**: Run the `pHelmIvP` instance inside the MITFrontseat MOOS community. With `pFrontseatMissionManager` watching over the helm.
- **Simple passive helm (`pHelmPassive`)**: No navigation is required. Very useful during initial testing phase of the vehicle.



Architecture of *MITFrontseat* (autonomy)

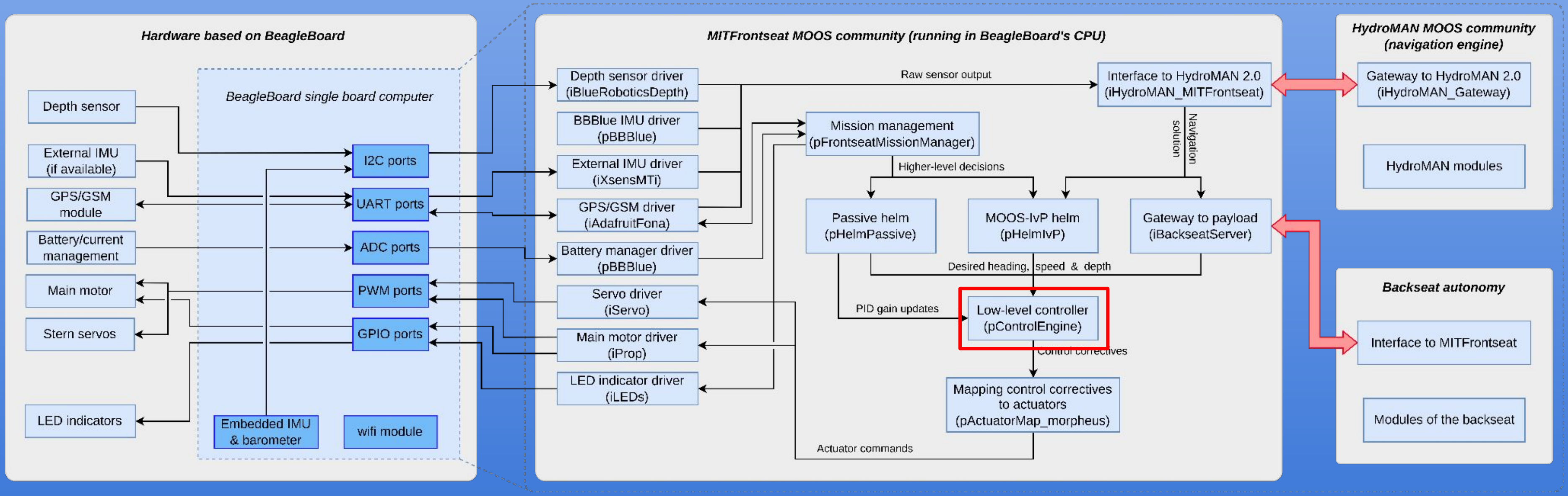
- **Autonomy system**: Produces desired heading, depth and speed
- **Payload autonomy**: Interfaces with a payload autonomy system; a.k.a. backseat driver (MOOS-IvP based or otherwise)
- **`pHelmIvP` on frontseat**: Run the `pHelmIvP` instance inside the MITFrontseat MOOS community. With `pFrontseatMissionManager` watching over the helm.
- **Simple passive helm (`pHelmPassive`)**: No navigation is required. Very useful during initial testing phase of the vehicle.




```
//---------------------------------------------------
// pHelmPassive config block

ProcessConfig = pHelmPassive
{
   AppTick    = 4
   CommsTick = 4

   ADD_LEG: start_time=15, heading=180, speed=2, depth=1.5
   ADD_LEG: start_time=25, heading=160, speed=2, depth=1.5, depth_kp=2.3
   ADD_LEG: start_time=35, heading=200, speed=2, depth=1.5
   ADD_LEG: start_time=45, heading=160, speed=2, depth=1.5, pitch_kd=1.1

   mission_end_time = $(MISSION_END_TIME)
}
```

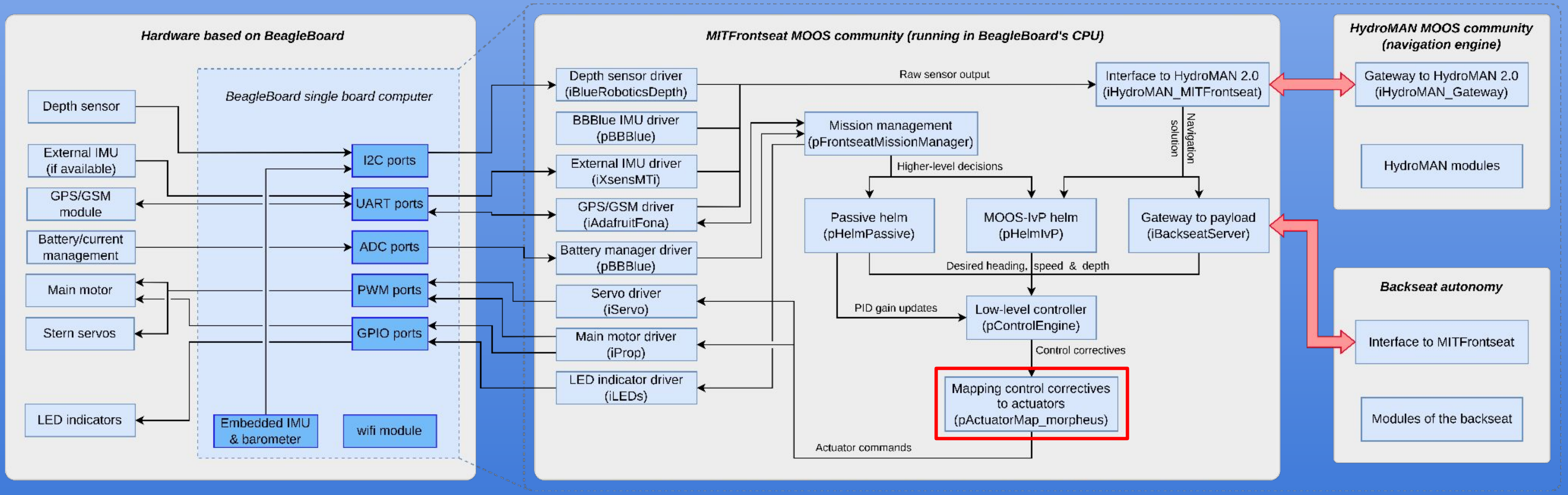
Architecture of *MITFrontseat* (low-level control)

- **Control engine (`pControlEngine`)**: Produces control correctives in speed, heading, depth, pitch and roll. Able to take in dynamic PID updates.



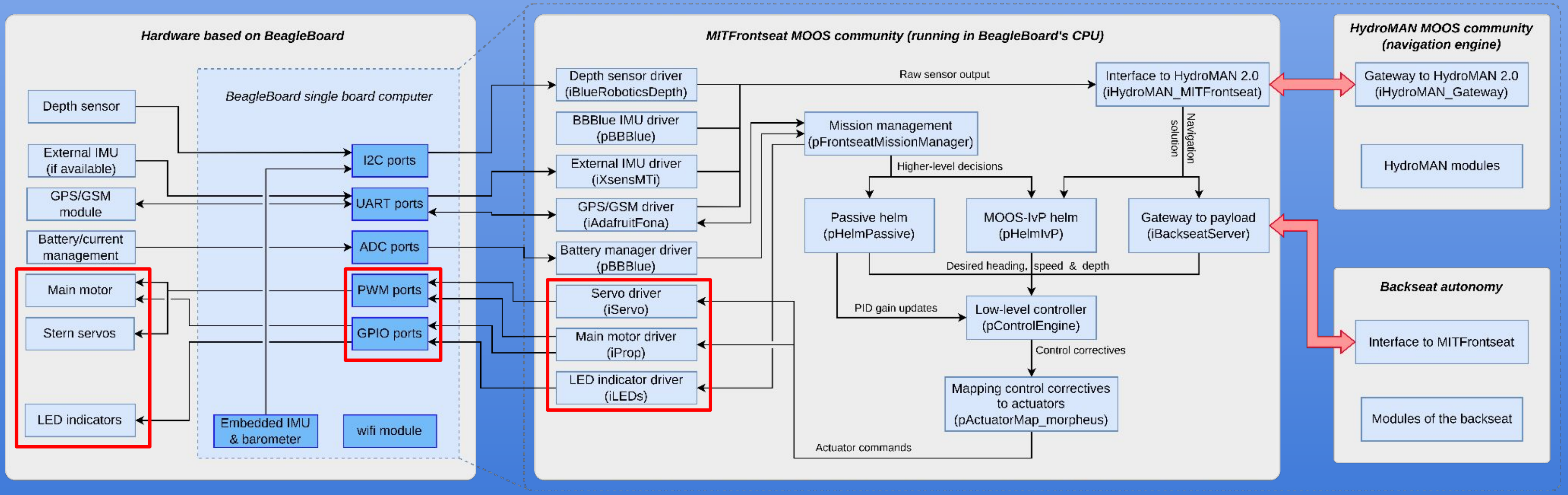
Architecture of *MITFrontseat* (low-level control)

- **Control engine (`pControlEngine`)**: Produces “control correctives” in speed, heading, depth, pitch and roll. Able to take in dynamic PID updates.
- **Actuator mapping (e.g. `pActuatorMap_CRay`)**: Maps out “control correctives” to the actuators of a specific vehicle class



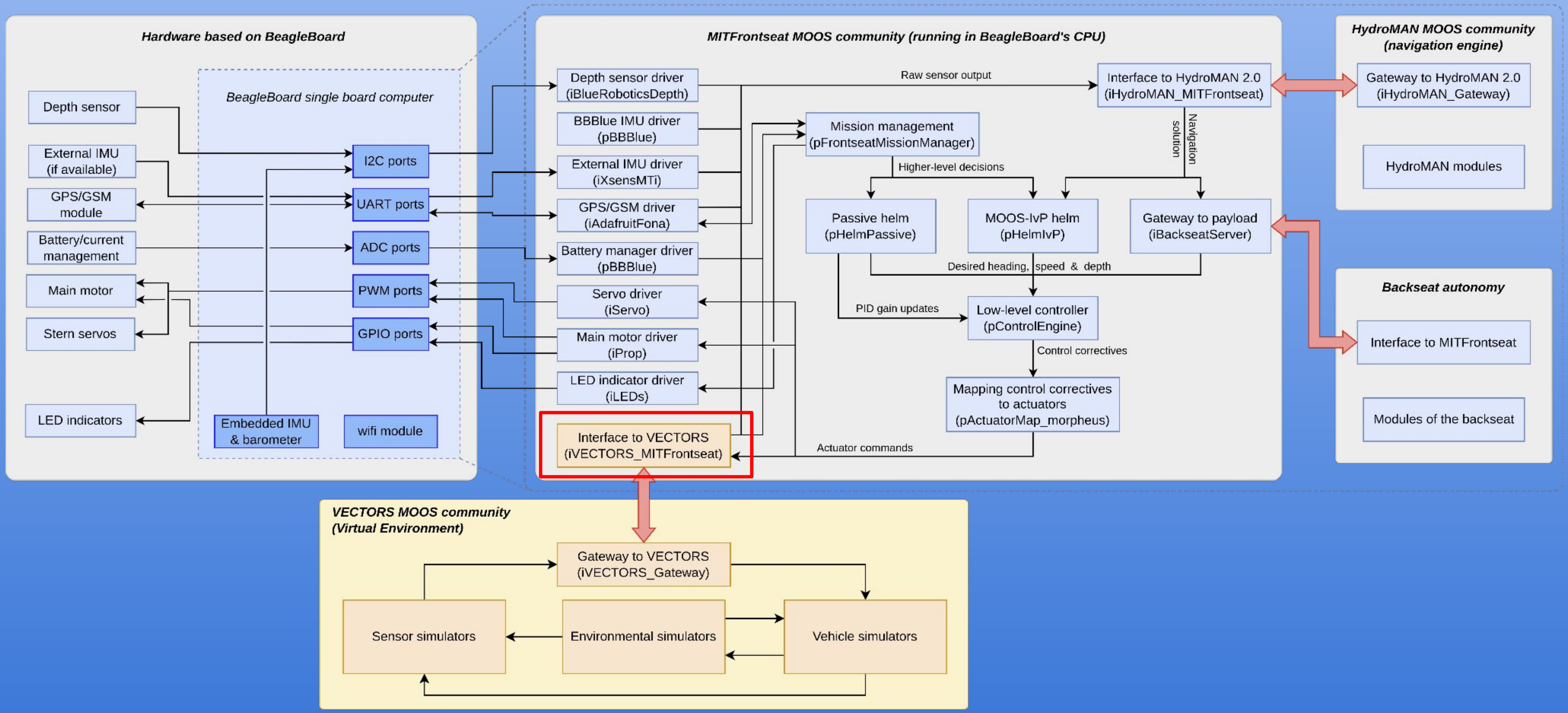
Architecture of *MITFrontseat* (low-level control)

- **Control engine (`pControlEngine`)**: Produces “control correctives” in speed, heading, depth, pitch and roll. Able to take in dynamic PID updates.
- **Actuator mapping (e.g. `pActuatorMap_CRay`)**: Maps out “control correctives” to the actuators of a specific vehicle class
- **Actuator drivers**: Sends the commands to actuators



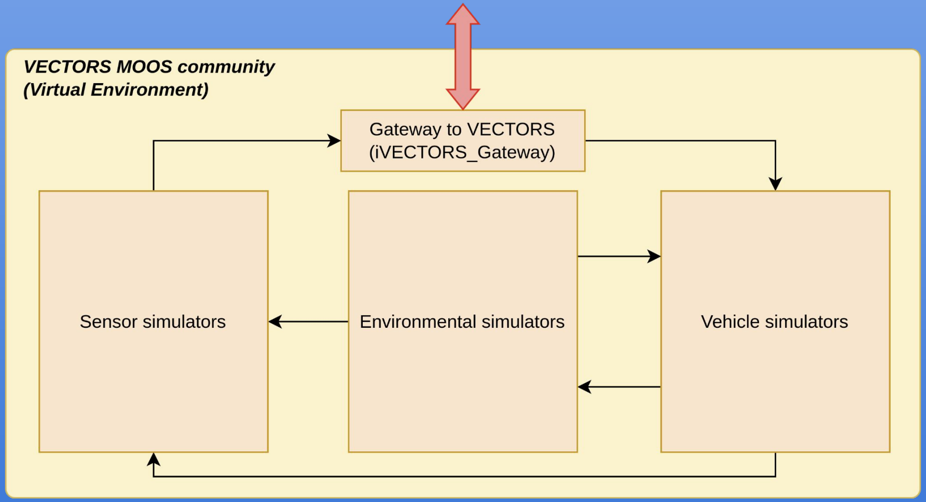
VECTORS

Virtual Environment for Construction and Testing of Oceanic Robotics Systems



Architecture of VECTORS

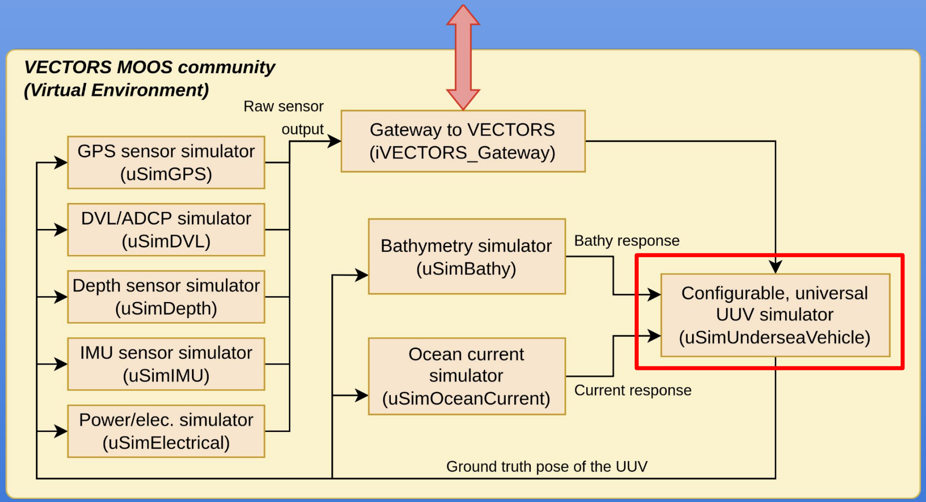
- **Vehicle simulators**: Simulates the motion of a vehicle platform according to its actuator movements and surrounding environmental forces
- **Environmental simulators**: Simulates environmental features such as bathymetry (both above and below surface), water currents, waves, etc.
- **Sensor simulators:** According to vehicle's ground truth motion response, environmental factors and sensor error model, a stream of raw sensor data outputs will be published



VECTORS

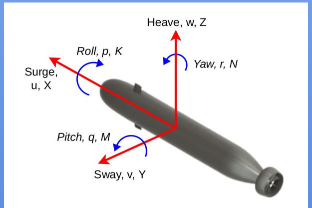
`uSimUnderseaVehicle:` *A configurable, physics-based 6-DOF UUV simulator*

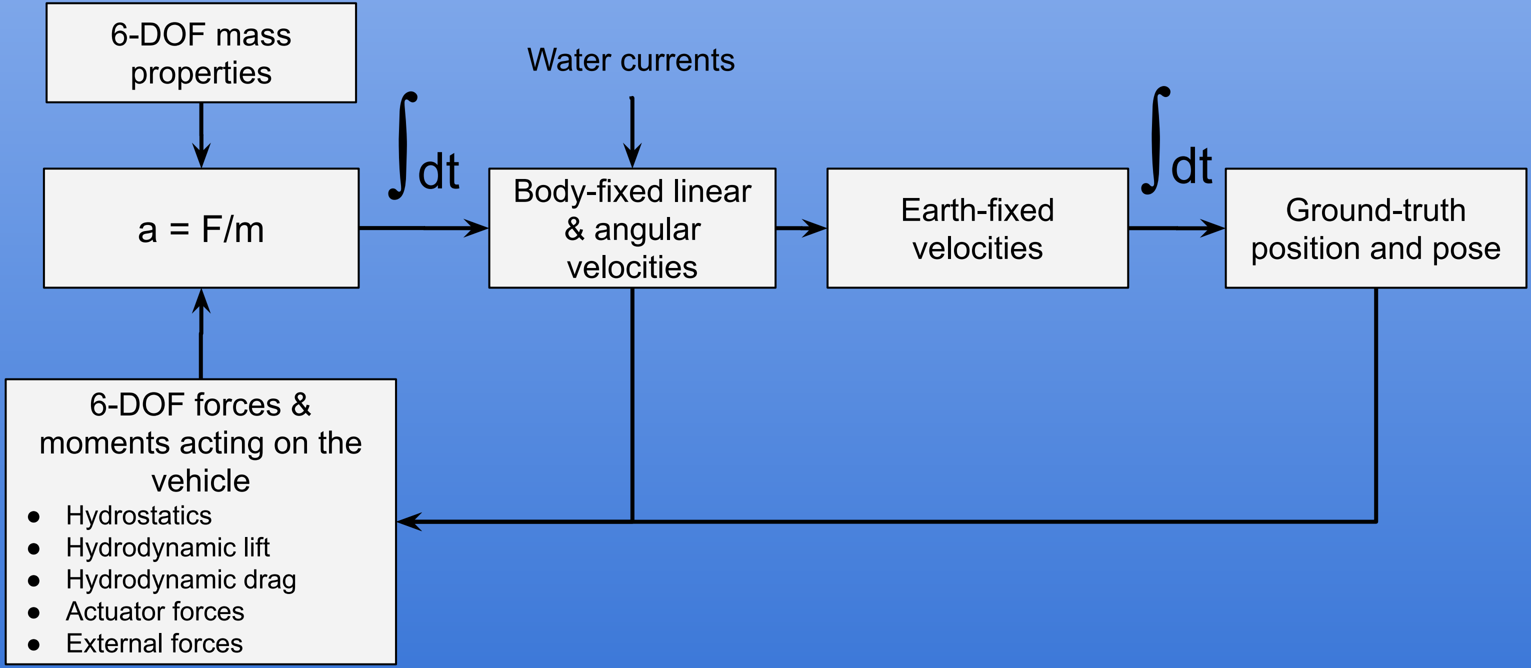
- Auto-generates hydrodynamic coefficients on-startup using empirical formulae
  - Configurable hullform shape
  - Configurable number of additional hulls
  - Configurable control surfaces with shape, size, position, orientation, buoyancy, etc.
  - Configurable static surfaces (i.e. fins, wings, shrouds) with shape, size, position, orientation, etc.
- Simulates effective velocity due to currents
- Simulates free surface hydrostatic variation
- Simulates seabed grounding forces



VECTORS

`uSimUnderseaVehicle:` *A configurable, physics-based 6-DOF UUV simulator*

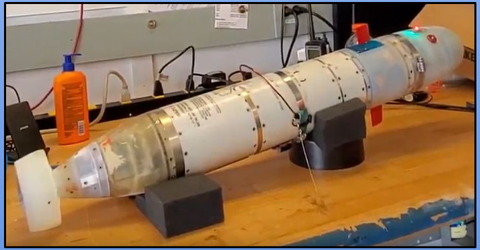


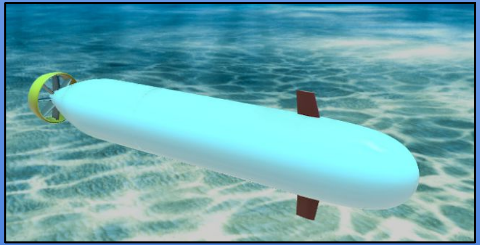


VECTORS

`uSimUnderseaVehicle:` *Configured for Morpheus UUV*

```
//--------------------------------------------------
// uSimUnderseaVehicle config block for Morpheus

ProcessConfig = uSimUnderseaVehicle
{
   AppTick   = 4
   CommsTick = 4
   log_path         = .
   log_name         = LOG_$(COMMUNITY)__hydro_summary_
   plot_name        = LOG_$(COMMUNITY)__veh_config_
   log_name__time_sufix = false

   // ------------- CONFIGURING THE MAIN HULL OF THE VEHICLE -------------

   HULLFORM_PROFILE: length=0.912, diameter=0.124, cd=1.1, cd_res=0.008, cd_axial=0.02, profile = 0.4560:0.0| 0.355:0.062|
   0.253:0.062| 0.152:0.062| 0.051:0.062| -0.051:0.062| -0.152:0.062| -0.253:0.062| -0.355:0.037| -0.456:0.0


   // ------------- CONFIGURING THE ACTUATORS OF THE VEHICLE -------------

   ADD_ACTUATOR: name=prop, index=0, type=fixed_thruster, Xprop=7.9763

   ADD_ACTUATOR: name=rudder_top, index=1, type=control_surface, orientation=0, xg=-0.42, xb=-0.42, zg=-0.0285, zb=-0.0285,
   surface_area=0.000977, surface_ar=3, surface_deltae=0.9

   ADD_ACTUATOR: name=rudder_btm, index=2, type=control_surface, orientation=0, xg=-0.42, xb=-0.42, zg=0.0285, zb=0.0285,
   surface_area=0.000977, surface_ar=3, surface_deltae=0.9

   ADD_ACTUATOR: name=elevator_port, index=3, type=control_surface, orientation=90, xg=-0.42, xb=-0.42, yg=-0.0285, yb=-0.0285,
   surface_area=0.000977, surface_ar=3, surface_deltae=0.9

   ADD_ACTUATOR: name=elevator_stbd, index=4, type=control_surface, orientation=90, xg=-0.42, xb=-0.42, yg=0.0285, yb=0.0285,
   surface_area=0.000977, surface_ar=3, surface_deltae=0.9

   ADD_ACTUATOR: name=fwdmorph_top, index=5, type=control_surface, orientation=0, xg=0.25, xb=0.25, zg=-0.0285, zb=-0.0285,
   surface_area=0.0014655, surface_ar=3, surface_deltae=0.9, is_deployed=false

   ADD_ACTUATOR: name=fwdmorph_top, index=6, type=control_surface, orientation=0, xg=0.25, xb=0.25, zg=0.0285, zb=0.0285,
   surface_area=0.0014655, surface_ar=3, surface_deltae=0.9, is_deployed=false
}
```





VECTORS

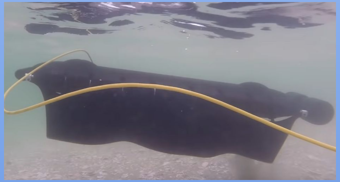
uSimUnderseaVehicle: *Configured for C-Ray UUV*

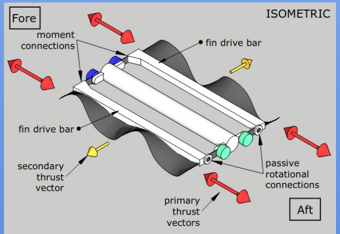
Main center hull:

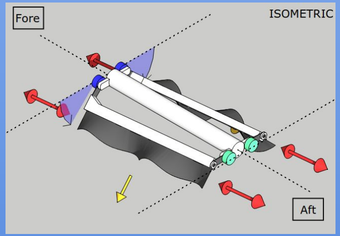
- Modeled as a rigid-body hull

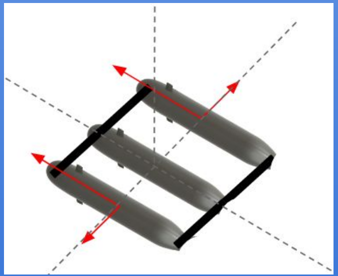
Fins:

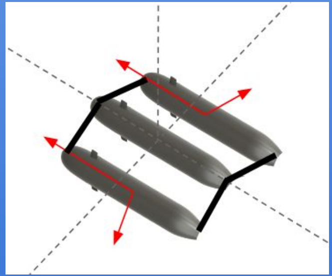
- Also modeled as rigid-body hulls, but connected to the main hull; i.e. all hydrodynamic forces/moments acting on fins get transferred to the center hull
- Two propellers are modeled on each fin to simulate the axial and side thrust generated from undulating fins
- As the tilt servos operate, the relative angles & CG (=CB) will shift accordingly.
- Depending on relative roll and relative pitch of the fins, the thrust directions will also change







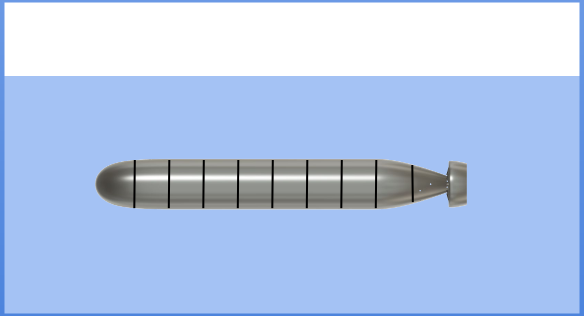




VECTORS

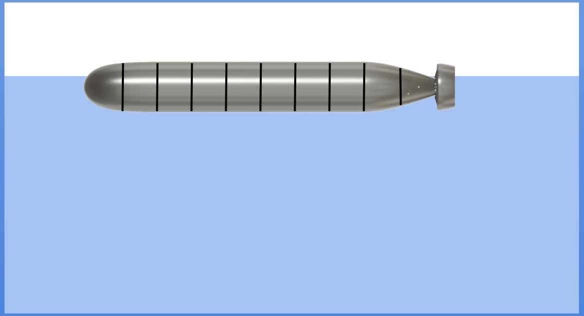
`uSimUnderseaVehicle:` *Free surface hydrostatics*

- Hull is divided into N sections
- If all sections are under the surface:
  - effective_buoyancy = buoyancy
  - effective_CB = CB
- If any part of a section is above the surface, the above surface surface volume is reduced from buoyancy, and effective_CB is re-calculated accordingly



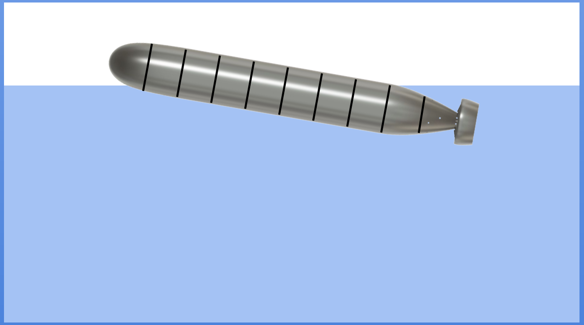
Effective_buoyancy = buoyancy

Effective_CB = CB



Effective_buoyancy is reduced

Effective_CB is lowered



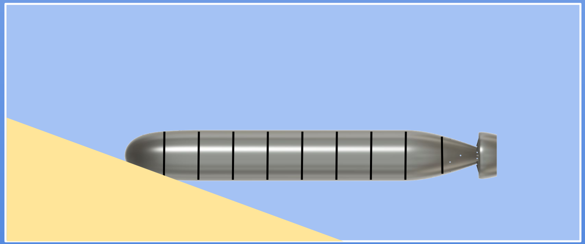
Effective_buoyancy is reduced

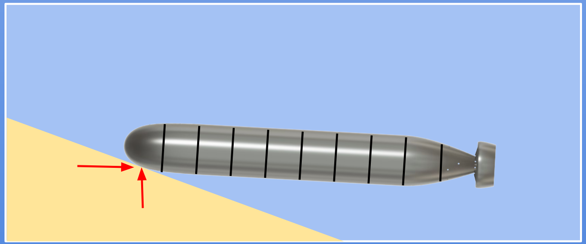
Effective_CB is moved lower astern

VECTORS

`uSimUnderseaVehicle:` *Grounding simulation (beta)*

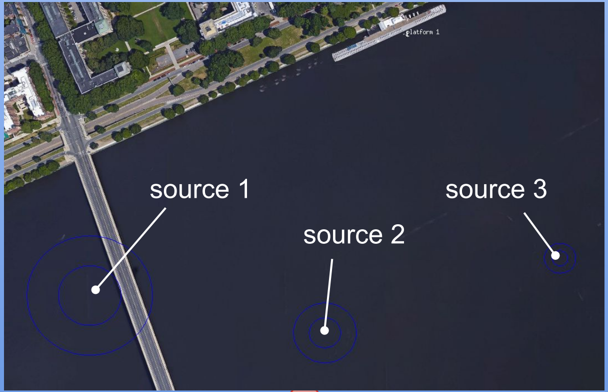
- Hull is divided into N (10) sections
- If any section is below the seabed (i.e. provided by *uSimBathy*) a reaction force is provided to the hull accordingly





VECTORS

`uSimOceanCurrent:` *A configurable ocean current simulator*

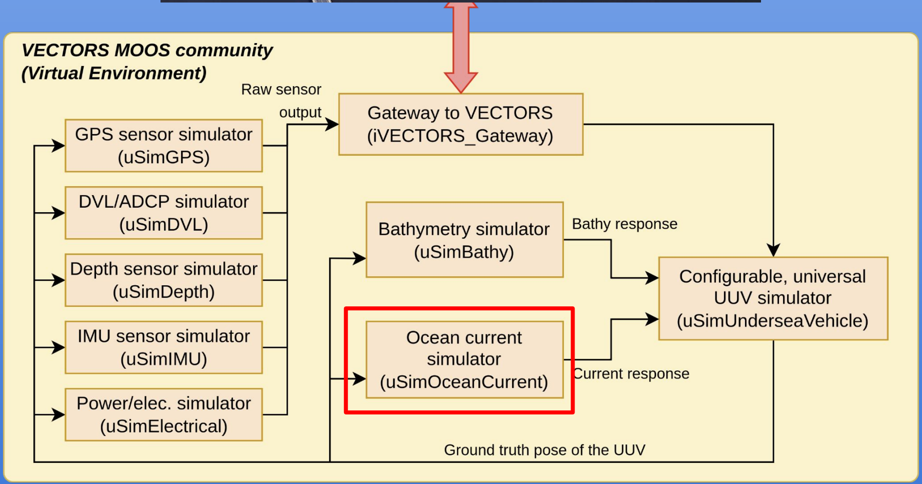


- Any number of radial underwater current sources can be configured with:
  - location
  - mean speed
  - amplitude
  - Wavelength
  - Current variation with depth as:

    *speed = depSq*depth^2 + dep*depth + mean_speed*

- At each vehicle's ground-truth position, the resultant current velocity is calculated, and a CURRENT_RESPONSE is posted

```
//------------------------------------------------
// uSimVECTORS_OceanCurrents config block

ProcessConfig = uSimVECTORS_OceanCurrents
{
   AppTick    = 4
   CommsTick = 4

   #ifdef BATCH_SIMULATION yes
      ADD_SOURCE: x=-400, y=-1650, mean_speed=$(VECTORS_CURRENT_SPD),
      amplitude=$(VECTORS_CURRENT_AMPL), wavelength=$(VECTORS_CURRENT_WAVELEN), depSq=0, dep=0

   #else
      ADD_SOURCE: x=-100, y=-400, mean_speed=0.2, amplitude=0.05, wavelength=20, depSq=-0.0005,
      dep=-0.0001
      ADD_SOURCE: x=-200, y=-300, mean_speed=0.3, amplitude=0.10, wavelength=10, depSq=-0.0005,
      dep=-0.0001
      ADD_SOURCE: x=-400, y=-350, mean_speed=0.1, amplitude=0.00, wavelength=40, depSq=-0.0005,
      dep=-0.0001
   #endif
}
```

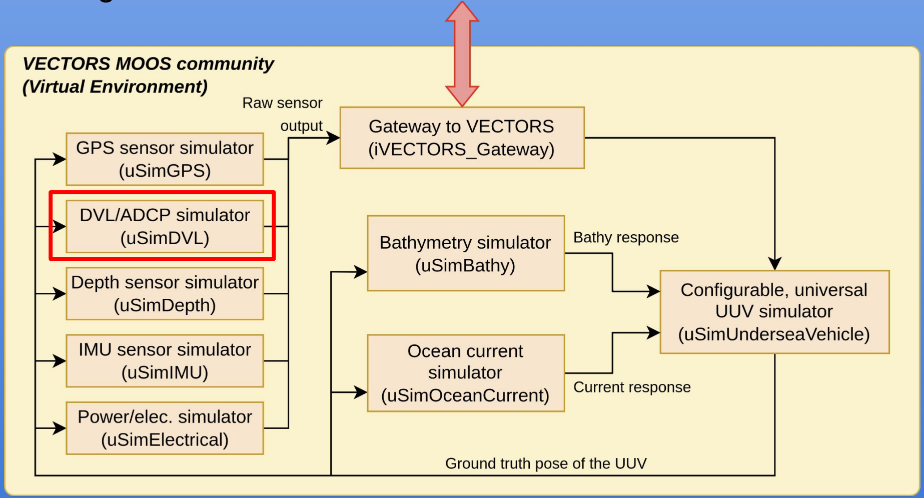


VECTORS

`uSimVECTORS_DVL`: *A DVL and ADCP simulator*

```
//-------------------------------------------------
// uSimVECTORS_DVL config block
ProcessConfig = uSimVECTORS_DVL
{
   AppTick    = 4
   CommsTick = 4

   platform_id              = 1         // Vehicle ID (unsigned int)
   sensor_id                = 1         // If there are >1 sensors per UUV (unsigned int)

   // ----------------------------------
   // Sensor axis vs. AUV axis
   deg_around_auv_x         = 180.0     // in deg
   deg_around_auv_y         = 0.0       // in deg
   deg_around_auv_z         = 0.0       // in deg

   // ----------------------------------
   // DVL specs and error model
   dvl_update_interval      = 0.5       // in s
   dvl_random_error         = 0.2       // in m/s
   dvl_bias_error           = 0.05      // in m/s
   dvl_scale_error          = 0.05      // Scale error as a percentage of velocity

   max_range                = 50        // in m
   min_range                = 0.3       // in m
   max_speed                = 10        // in m/s
   bin_height               = 0.50      // in m
   water_track_mode_exists = true       // bool
   water_track_bin          = 1         // the bin number, starting with nearest bin

   // auto_adjust_update_interval = false // This mode is not supported yet.
   // ----------------------------------
   // ADCP mode and specs
   dvl_adcp_dual_mode       = true      // bool
   adcp_random_error        = 0.3       // in m/s
   adcp_bias_error          = 0.06      // in m/s
   adcp_scale_error         = 0.06      // Scale error as a percentage of velocity

}
```

- Match the sensor to a particular vehicle
- Able to have more than one DVL per vehicle (e.g. upward and downward looking)
- Configurable sensor orientation w.r.t. vehicle axis
- Configurable DVL bottom-track error model
- Configurable range, blanking distance, update rate
- Configurable DVL water-track mode & water-track bin
- DVL-ADCP dual mode support
- Configurable ADCP error model
-



MITFrontseat - HydroMAN - VECTORS with MOOS-IvP in action

Simulating the autonomy stack of C-Ray vehicle developed by Pliant Energy Systems

Credit for RViz visualization tool implementation with MOOS-ROS bridge:
Ethan Park <Park@pliantenergy.com>

Acknowledgements

BATTELLE

Office of Naval Research ONR Science & Technology

LOCKHEED MARTIN

PES

MIT Laboratory for Autonomous Marine Sensing Systems

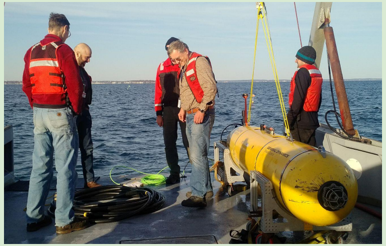
MIT SAILING

MIT Sea Grant AUV laboratory

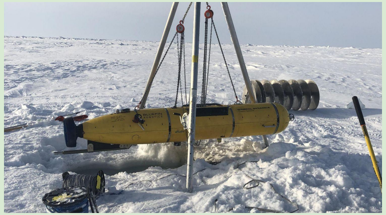
Pre - HydroMAN

Sandshark model-aided navigation
at MIT sailing pavilion (2018)

HydroMAN 1.0



Pre-ICEX20 engineering tests
at Mass Bay (2019)



ICEX20 under-ice navigation
at Beaufort Sea, Arctic (2020)

MIT variant of MK-39 EMATT
at MIT sailing pavilion (2020)

HydroMAN 2.0

Morpheus AUV
at MIT sailing pavilion (2021)

GPS-denied navigation
at MIT sailing pavilion (2021)

(on-going)

(on-going)

(on-going)