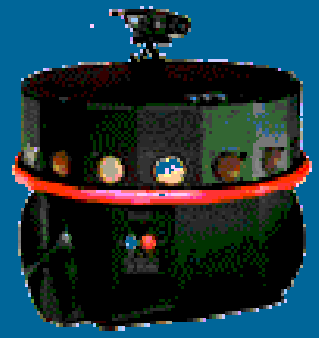
# Experimental Setup of an Hierarchical Control Architecture


Paulo Alvito

pja@isr.ist.utl.pt

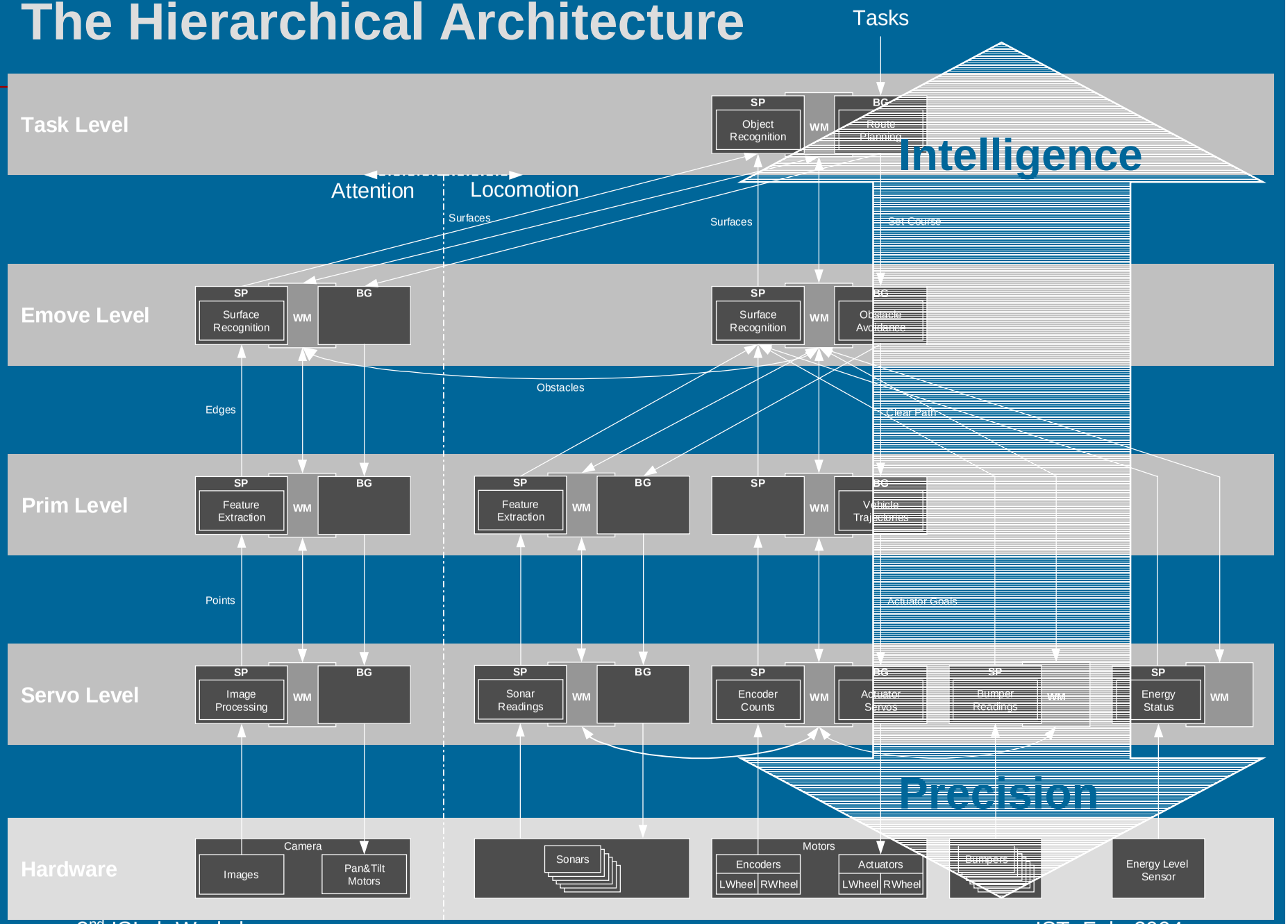
3rd ISLab Workshop

# Current Work



- Goal
  - Apply a control architecture to real mobile robot so that it can perform a find-and-deliver task in an office-building-like environment.
- Critical Issues
  - Different subsystems will have to be combined: navigation, obstacle avoidance and object recognition.
  - Each of these subsystems can be composed of different sensors and actuators.
  - Converging all the information to a decision point can be a bottleneck concerning fast reaction to events...
  - ... but it can be positive concerning the combination of different sources of knowledge and to provide a common actuation.

# The Hierarchical Architecture

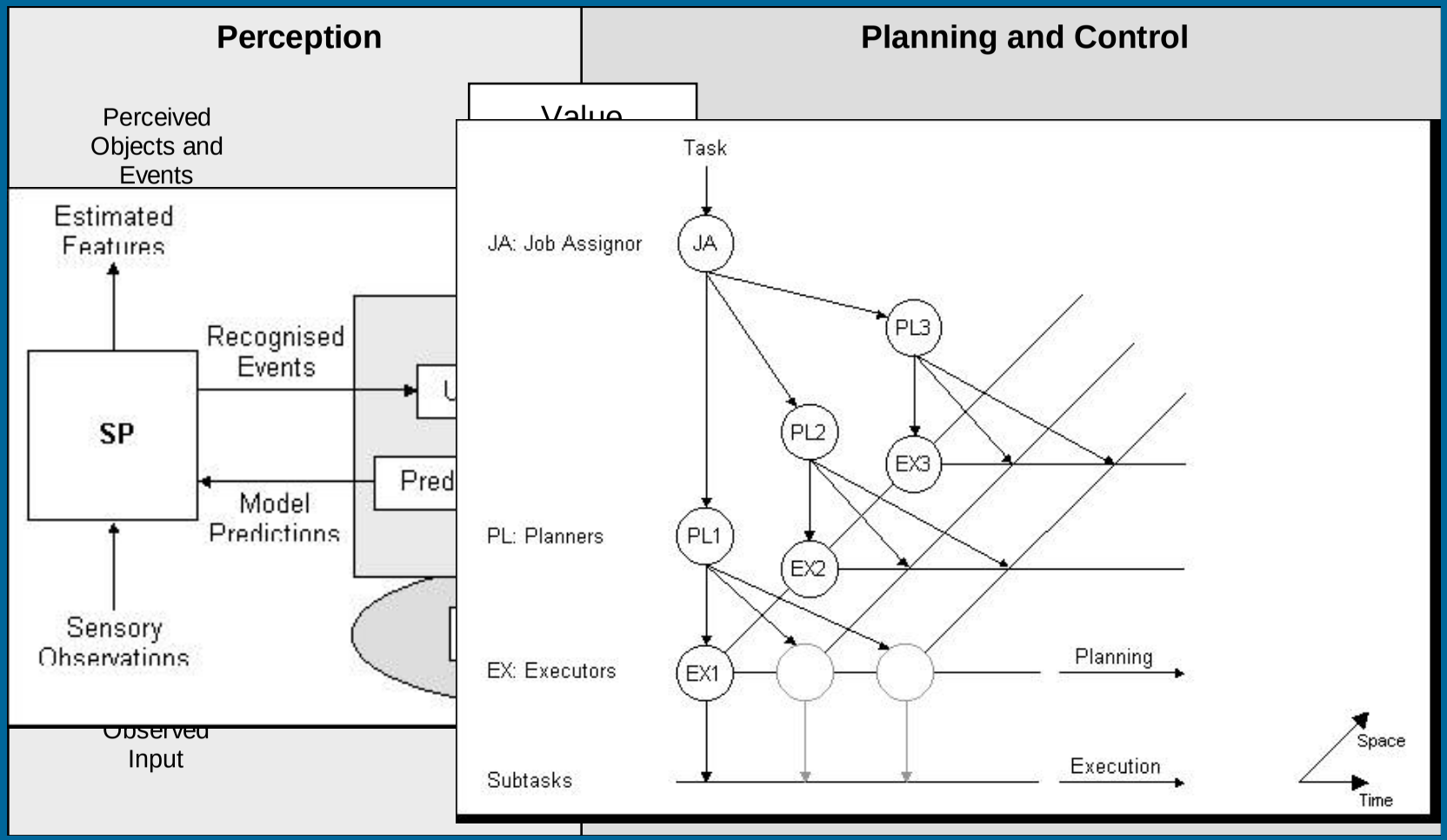


# Advantages

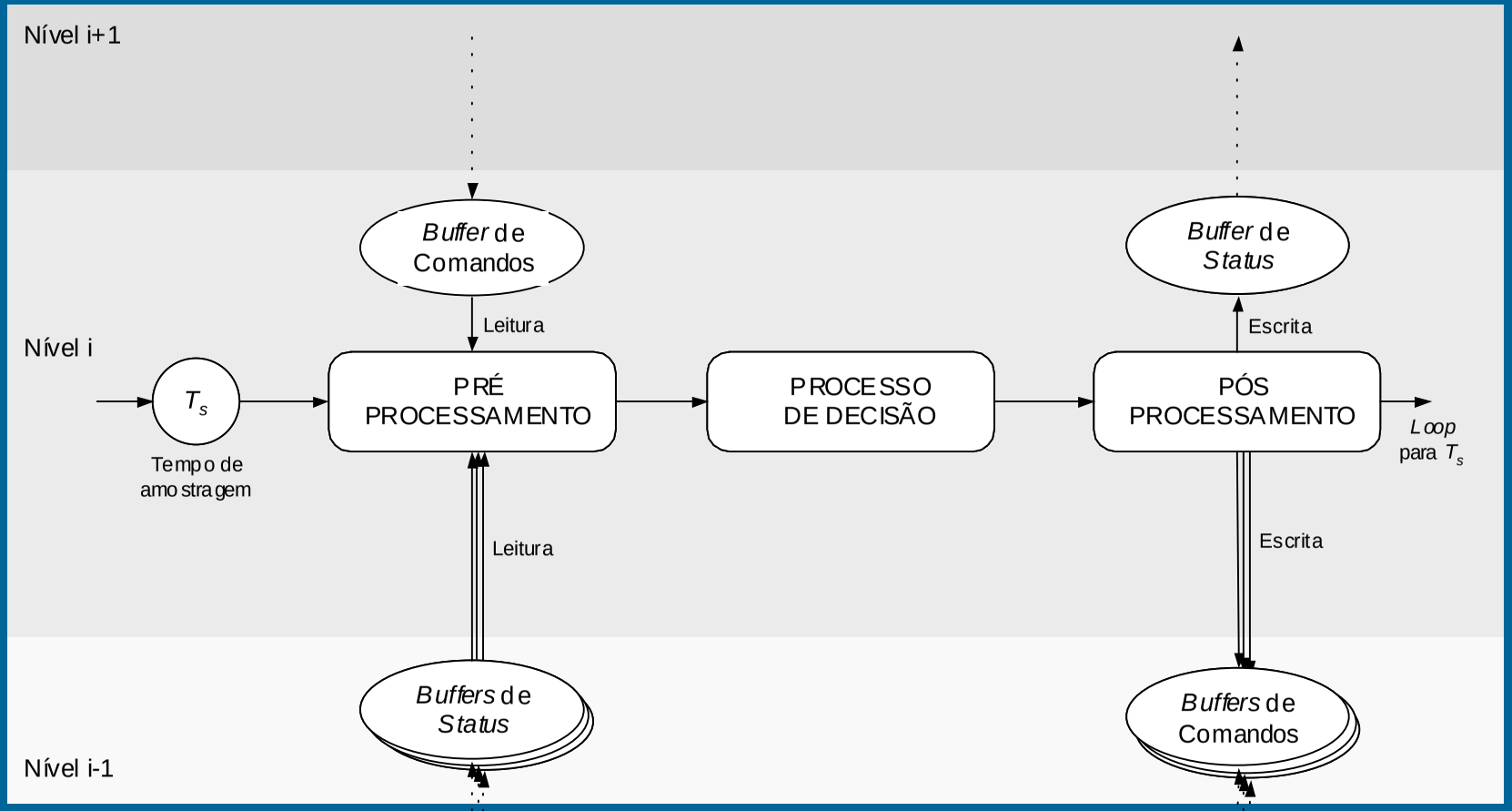
- Easy to divide the system into modules and to establish relations between them.
- The architecture is organized into different levels of abstraction.
- When we are going up in the hierarchy we are following the IPDI principle – Saridis (1983), we are *increasing intelligence while decreasing precision*.
- Higher levels plan based on simbolic descriptions and lower levels deal with signals.
- The hierarchy “drives” the system for the goal but, but at the same time it leaves some compliance so that modules can react to unspected events.

# Relations Inside a Single Node

- Fundamental Elements Inside a Single Node



# Methodology Application

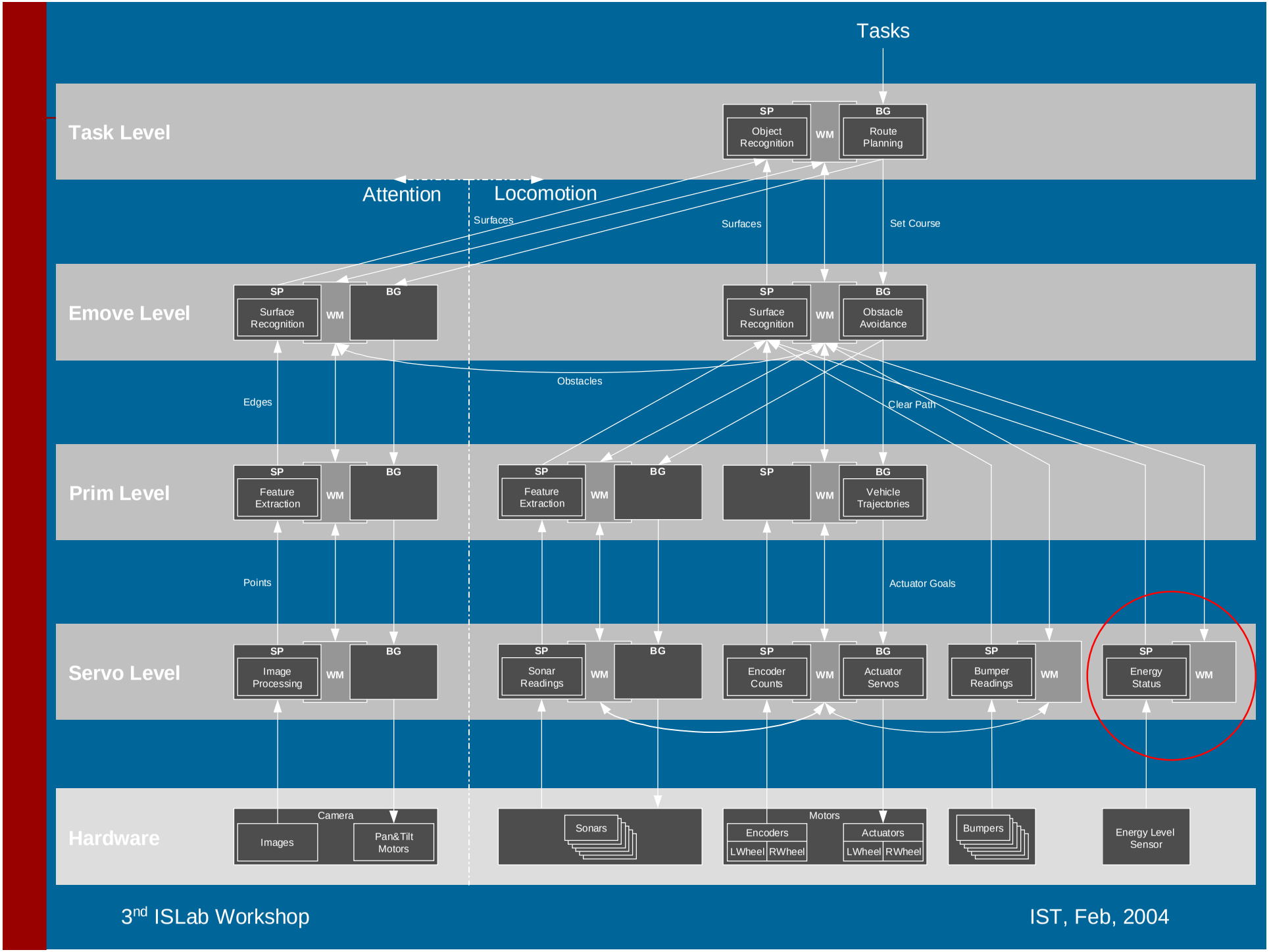



- Each node is a *thread* and is working assyncronously
- Cyclic looping structure
- There exist a *mutex* (mutual exclusion) variable for each shared buffer.

# Benefits of Threads vs. Processes

- Less time to create a new thread than a process, because the newly created thread uses the current process address space.
- Less time to terminate a thread than a process.
- Less time to switch between two threads within the same process, partly because the newly created thread uses the current process address space.
- Less communication overheads -- communicating between the threads of one process is simple because the threads share everything: address space, in particular. So, data produced by one thread is immediately available to all the other threads.

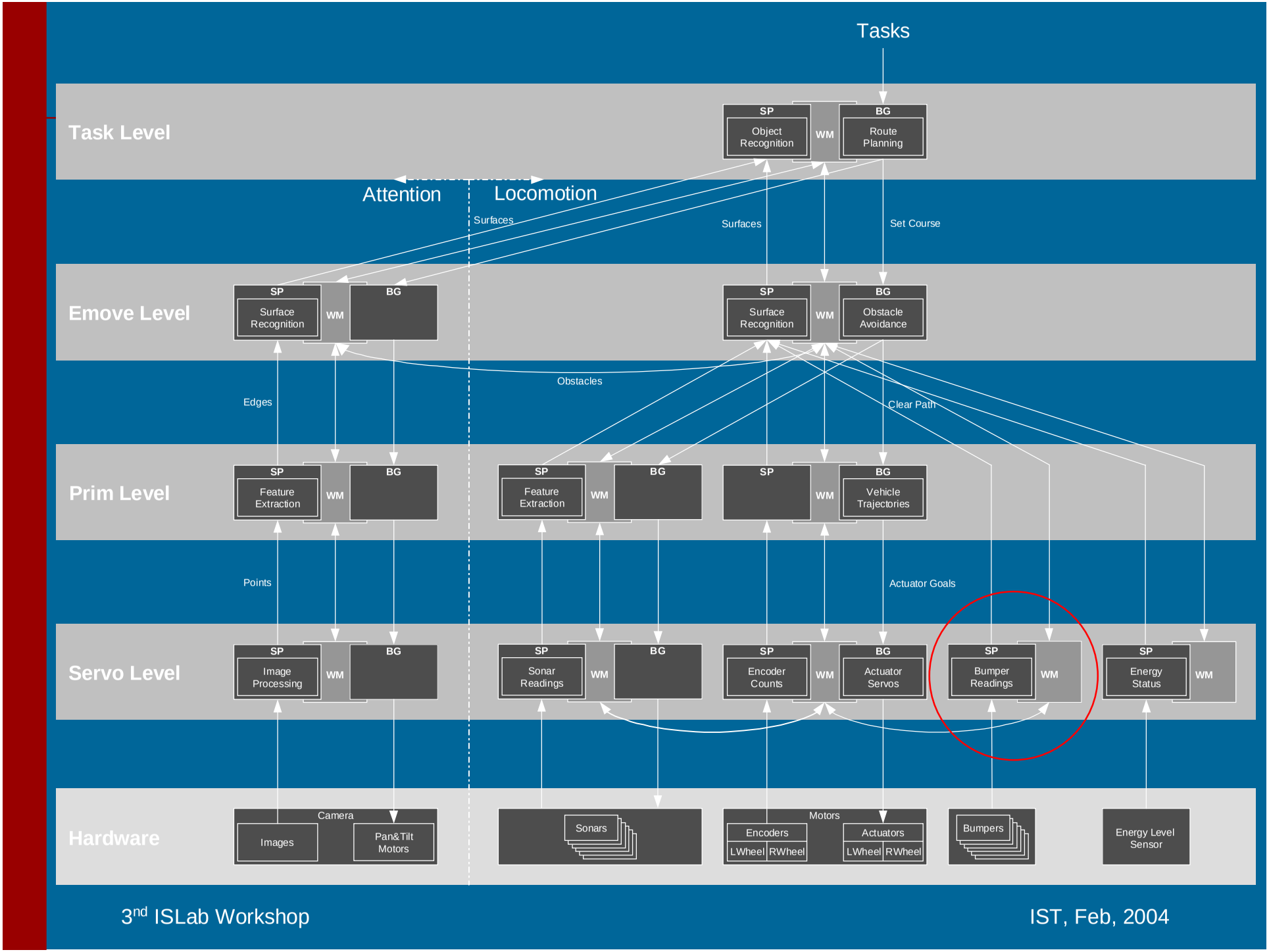
# Experimental Setup


Tasks
Task Level
SP
Object Recognition
WM
BG
Route Planning
Attention
Locomotion
Surfaces
Surfaces
Set Course
Emove Level
SP
Surface Recognition
WM
BG
SP
Surface Recognition
WM
BG
Obstacle Avoidance
Obstacles
Edges
Clear Path
Prim Level
SP
Feature Extraction
WM
BG
SP
Feature Extraction
WM
BG
SP
WM
BG
Vehicle Trajectories
Points
Actuator Goals
Servo Level
SP
Image Processing
WM
BG
SP
Sonar Readings
WM
BG
SP
Encoder Counts
WM
BG
Actuator Servos
SP
Bumper Readings
WM
SP
Energy Status
WM
Hardware
Camera
Images
Pan&Tilt Motors
Sonars
Motors
Encoders
LWheel
RWheel
Actuators
LWheel
RWheel
Bumpers
Energy Level Sensor

Incoming commands: None

Outgoing status: Battery State

WM: Battery States

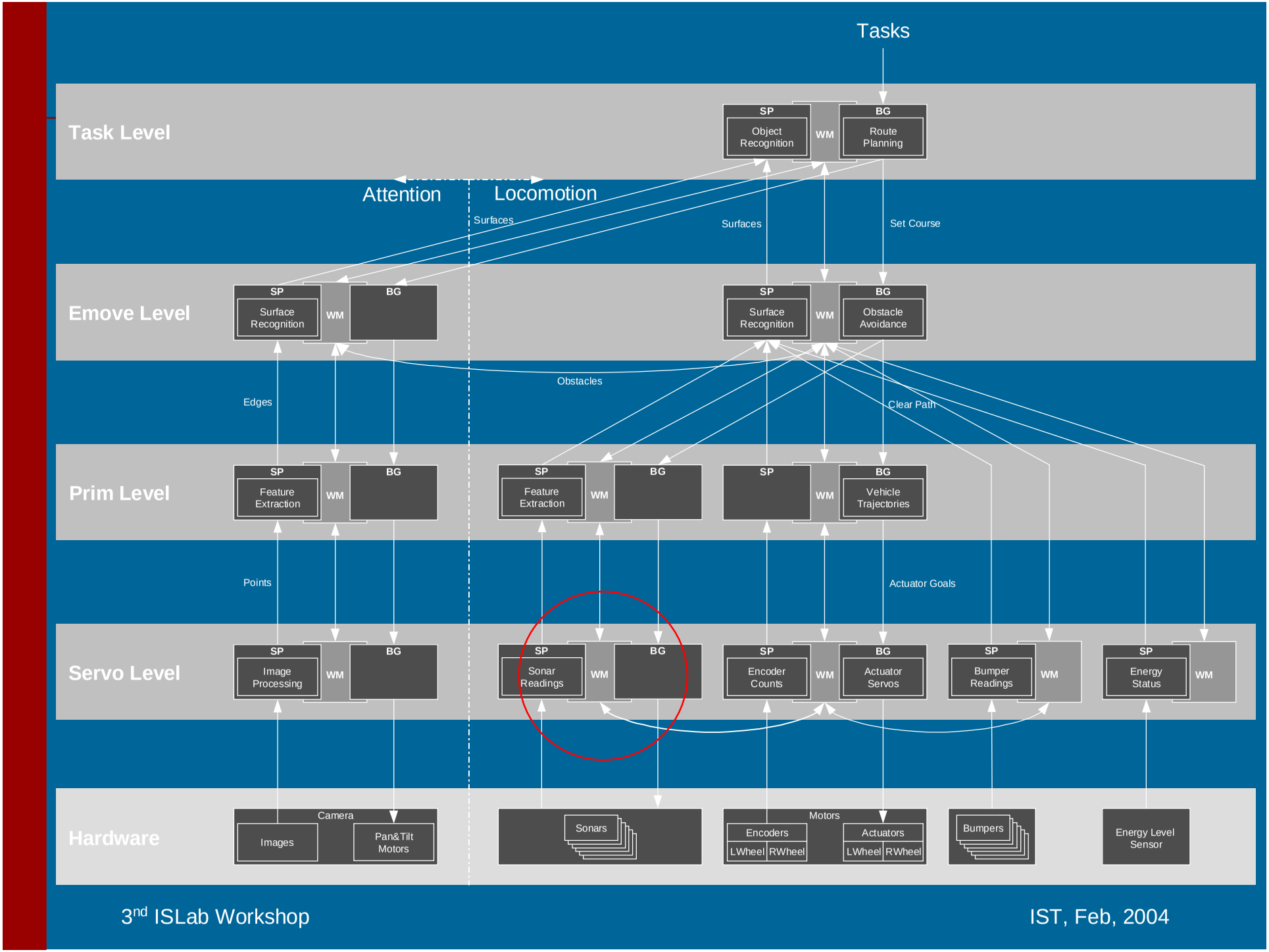

Tasks
Task Level
SP
Object Recognition
WM
BG
Route Planning
Attention
Locomotion
Surfaces
Surfaces
Set Course
Emove Level
SP
Surface Recognition
WM
BG
SP
Surface Recognition
WM
BG
Obstacle Avoidance
Obstacles
Edges
Clear Path
Prim Level
SP
Feature Extraction
WM
BG
SP
Feature Extraction
WM
BG
SP
WM
BG
Vehicle Trajectories
Points
Actuator Goals
Servo Level
SP
Image Processing
WM
BG
SP
Sonar Readings
WM
BG
SP
Encoder Counts
WM
BG
Actuator Servos
SP
Bumper Readings
WM
SP
Energy Status
WM
Hardware
Camera
Images
Pan&Tilt Motors
Sonars
Motors
Encoders
LWheel
RWheel
Actuators
LWheel
RWheel
Bumpers
Energy Level Sensor

Incoming commands: None

Outgoing status: One Bumper is On?

WM: Bumpers[]

! The node over the motors is aware of this WM, providing low-level reactivity.


Tasks
Task Level
SP
Object Recognition
WM
BG
Route Planning
Attention
Locomotion
Surfaces
Surfaces
Set Course
Emove Level
SP
Surface Recognition
WM
BG
SP
Surface Recognition
WM
BG
Obstacle Avoidance
Obstacles
Edges
Clear Path
Prim Level
SP
Feature Extraction
WM
BG
SP
Feature Extraction
WM
BG
SP
WM
BG
Vehicle Trajectories
Points
Actuator Goals
Servo Level
SP
Image Processing
WM
BG
SP
Sonar Readings
WM
BG
SP
Encoder Counts
WM
BG
Actuator Servos
SP
Bumper Readings
WM
SP
Energy Status
WM
Hardware
Camera
Images
Pan&Tilt Motors
Sonars
Motors
Encoders
LWheel
RWheel
Actuators
LWheel
RWheel
Bumpers
Energy Level Sensor

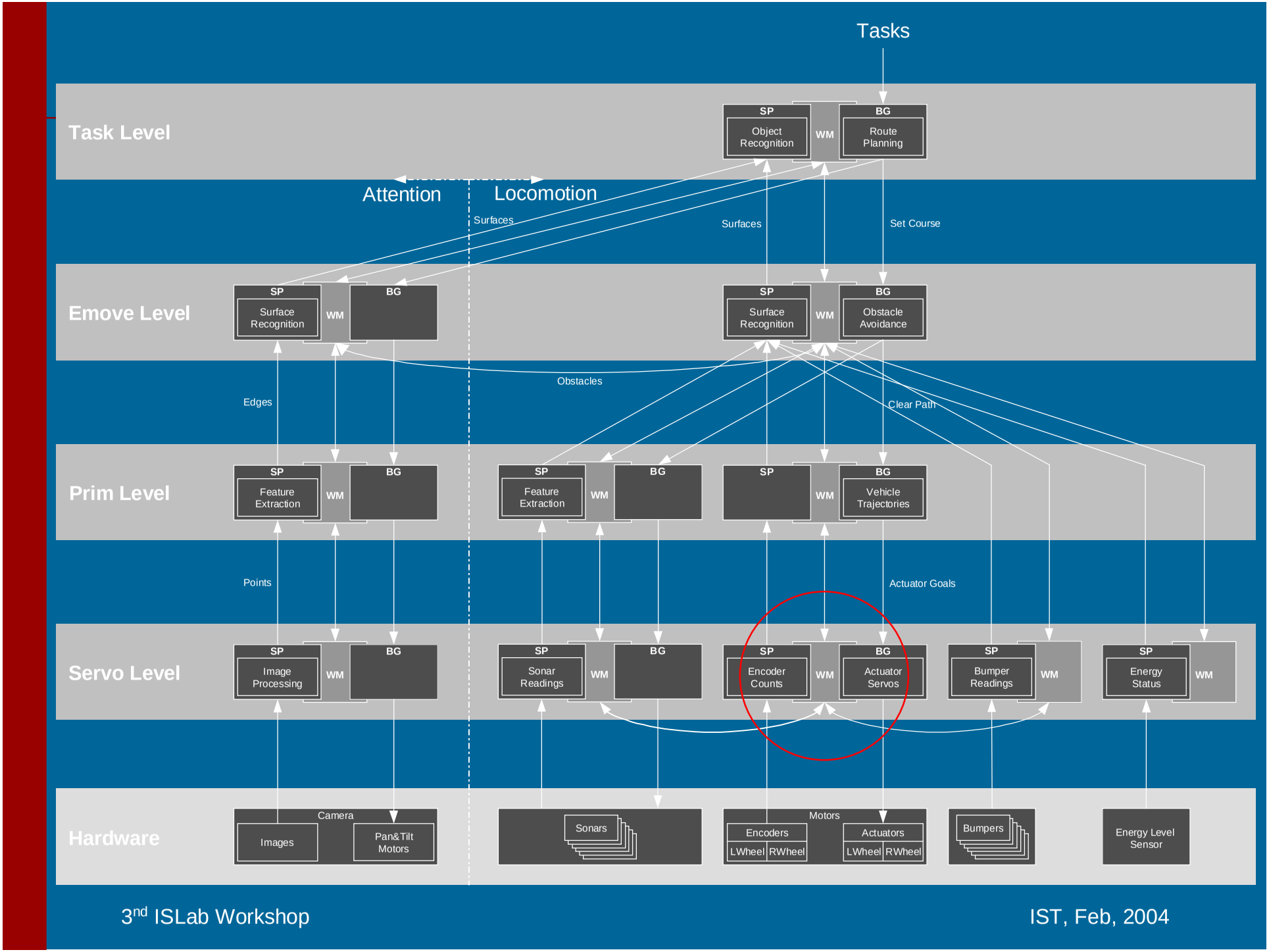
Incoming commands: Sonar Config (All, Front, Rear, Stop, etc)

Outgoing status: Sonar Config Ok?

New Read Ok?

WM: SonarRead[], SonarMinRead, SonarFirerate

! The node over the motors is aware of this WM, providing low-level reactivity.

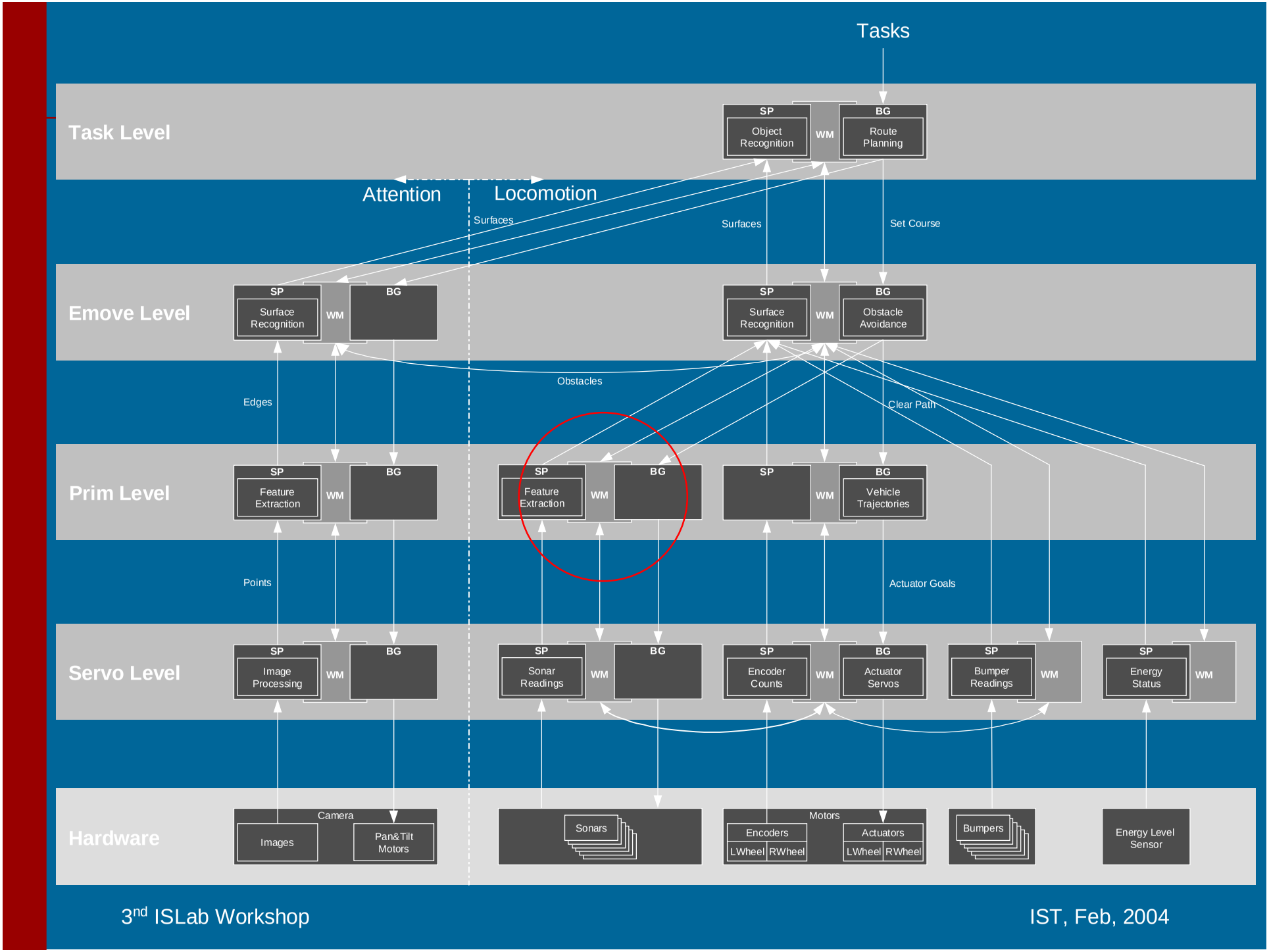

Tasks
Task Level
SP
Object Recognition
WM
BG
Route Planning
Attention
Locomotion
Surfaces
Surfaces
Set Course
Emove Level
SP
Surface Recognition
WM
BG
SP
Surface Recognition
WM
BG
Obstacle Avoidance
Obstacles
Edges
Clear Path
Prim Level
SP
Feature Extraction
WM
BG
SP
Feature Extraction
WM
BG
SP
WM
BG
Vehicle Trajectories
Points
Actuator Goals
Servo Level
SP
Image Processing
WM
BG
SP
Sonar Readings
WM
BG
SP
Encoder Counts
WM
BG
Actuator Servos
SP
Bumper Readings
WM
SP
Energy Status
WM
Hardware
Camera
Images
Pan&Tilt Motors
Sonars
Motors
Encoders
LWheel
RWheel
Actuators
LWheel
RWheel
Bumpers
Energy Level Sensor

Incoming commands: OdometrySetPosition; motorsDiffDist (incr_dist, incr_steer); SetVelocity

Outgoing status: Command Ok?

Safety Error?

WM: x, y, θ

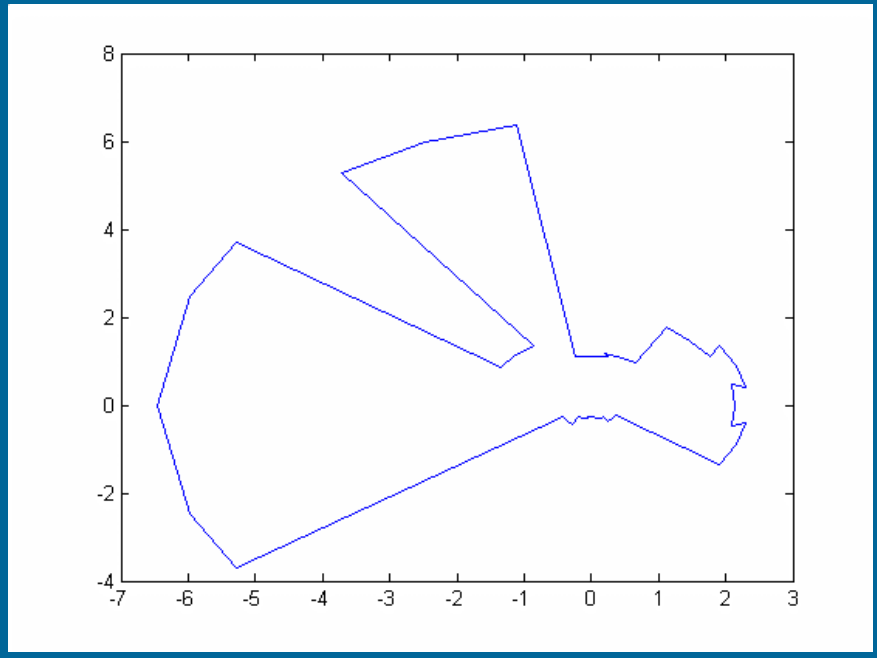

Tasks
Task Level
SP
Object Recognition
WM
BG
Route Planning
Attention
Locomotion
Surfaces
Surfaces
Set Course
Emove Level
SP
Surface Recognition
WM
BG
SP
Surface Recognition
WM
BG
Obstacle Avoidance
Obstacles
Edges
Clear Path
Prim Level
SP
Feature Extraction
WM
BG
SP
Feature Extraction
WM
BG
SP
WM
BG
Vehicle Trajectories
Points
Actuator Goals
Servo Level
SP
Image Processing
WM
BG
SP
Sonar Readings
WM
BG
SP
Encoder Counts
WM
BG
Actuator Servos
SP
Bumper Readings
WM
SP
Energy Status
WM
Hardware
Camera
Images
Pan&Tilt Motors
Sonars
Motors
Encoders
LWheel
RWheel
Actuators
LWheel
RWheel
Bumpers
Energy Level Sensor

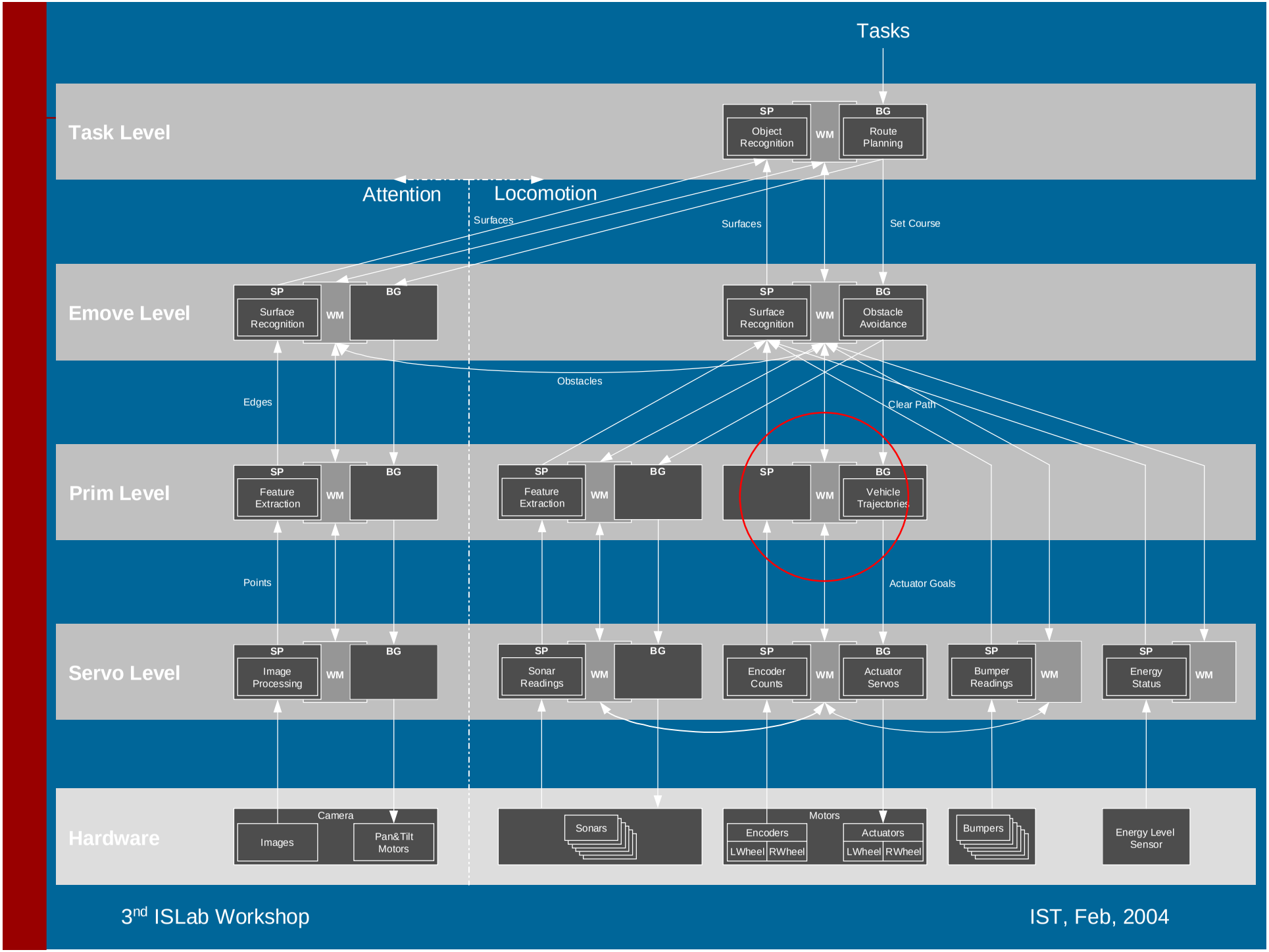
Incoming commands: Sonar Config

Outgoing status: New Read Ok?

Config Ok?

WM: SonarRead[]; SonarLines[]




Tasks
Task Level
SP
Object Recognition
WM
BG
Route Planning
Attention
Locomotion
Surfaces
Surfaces
Set Course
Emove Level
SP
Surface Recognition
WM
BG
SP
Surface Recognition
WM
BG
Obstacle Avoidance
Obstacles
Edges
Clear Path
Prim Level
SP
Feature Extraction
WM
BG
SP
Feature Extraction
WM
BG
SP
WM
BG
Vehicle Trajectories
Points
Actuator Goals
Servo Level
SP
Image Processing
WM
BG
SP
Sonar Readings
WM
BG
SP
Encoder Counts
WM
BG
Actuator Servos
SP
Bumper Readings
WM
SP
Energy Status
WM
Hardware
Camera
Images
Pan&Tilt Motors
Sonars
Motors
Encoders
LWheel
RWheel
Actuators
LWheel
RWheel
Bumpers
Energy Level Sensor

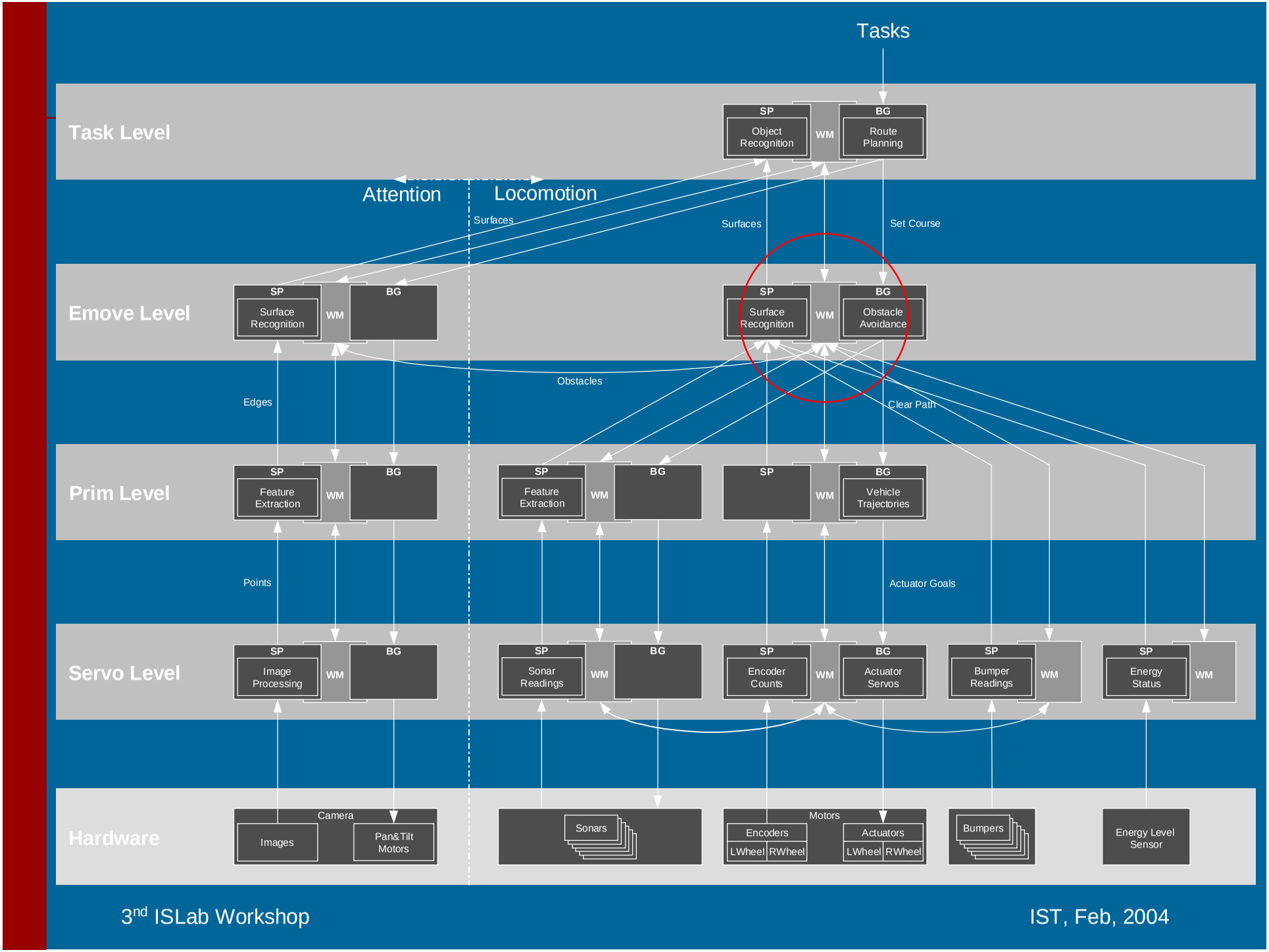
Incoming commands: GotoXY; FollowCorridor; CrossDoor; Avoid Obstacle;

Outgoing status: Command Ok?

Safety Error?

WM: x, y, θ, velocities

! This node is aware of sonars prim WM, so that it can identify walls and obstacles.


Tasks
Task Level
SP
Object Recognition
WM
BG
Route Planning
Attention
Locomotion
Surfaces
Surfaces
Set Course
Emove Level
SP
Surface Recognition
WM
BG
SP
Surface Recognition
WM
BG
Obstacle Avoidance
Obstacles
Edges
Clear Path
Prim Level
SP
Feature Extraction
WM
BG
SP
Feature Extraction
WM
BG
SP
WM
BG
Vehicle Trajectories
Points
Actuator Goals
Servo Level
SP
Image Processing
WM
BG
SP
Sonar Readings
WM
BG
SP
Encoder Counts
WM
BG
Actuator Servos
SP
Bumper Readings
WM
SP
Energy Status
WM
Hardware
Camera
Images
Pan&Tilt Motors
Sonars
Motors
Encoders
LWheel
RWheel
Actuators
LWheel
RWheel
Bumpers
Energy Level Sensor

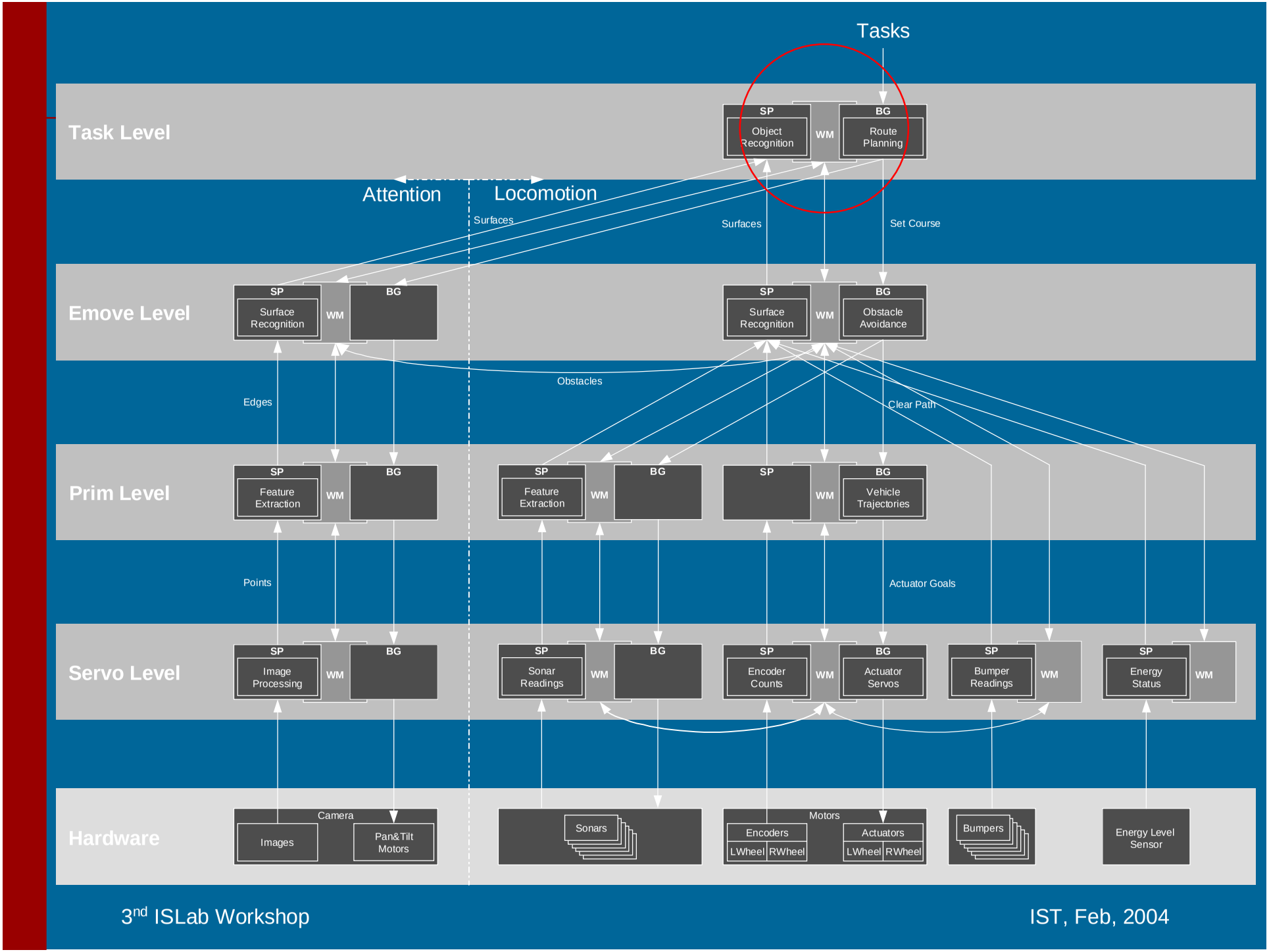
Incoming commands: Goto Office Number... ; Goto charger;

Outgoing status: Command Ok?

Safety Error?

WM: Geometrical Map

```
typedef struct GeometricalMap
{
   int MapPointID;
   float x_abs;
   float y_abs;
} GeoMap;
```

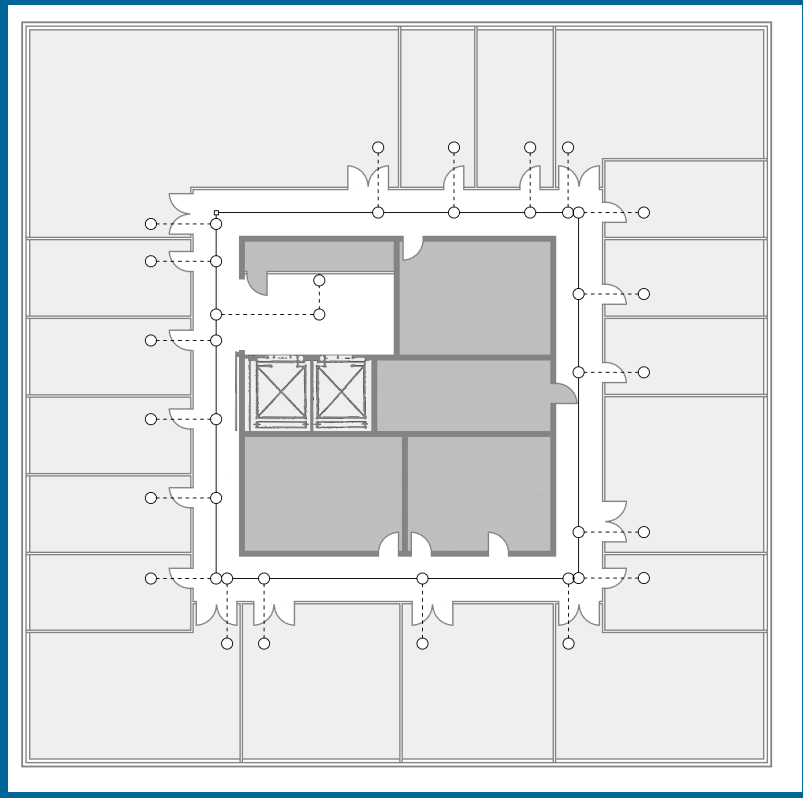

Tasks
Task Level
SP
Object Recognition
WM
BG
Route Planning
Attention
Locomotion
Surfaces
Surfaces
Set Course
Emove Level
SP
Surface Recognition
WM
BG
SP
Surface Recognition
WM
BG
Obstacle Avoidance
Obstacles
Edges
Clear Path
Prim Level
SP
Feature Extraction
WM
BG
SP
Feature Extraction
WM
BG
SP
WM
BG
Vehicle Trajectories
Points
Actuator Goals
Servo Level
SP
Image Processing
WM
BG
SP
Sonar Readings
WM
BG
SP
Encoder Counts
WM
BG
Actuator Servos
SP
Bumper Readings
WM
SP
Energy Status
WM
Hardware
Camera
Images
Pan&Tilt Motors
Sonars
Motors
Encoders
LWheel
RWheel
Actuators
LWheel
RWheel
Bumpers
Energy Level Sensor

Incoming commands: Come to my office; Go to office...

Outgoing status: Mission Completed?

WM: Topological Map

```
typedef struct TopologicalMap
{
   string_point MapPoint;
   int MapPointID;
   string_point North;
   string_point East;
   string_point South;
   string_point West;
   float distanceN;
   float distanceE;
   float distanceS;
   float distanceO;
} TopMap;
```



Example: The robot receives an order to go to office C

Task Level: Based on the Topological Map, computes the shortest path. In order to reach C, GOTO A, then GOTO B and finally GOTO C. Meanwhile if the robot recognizes a corner it can resets its encoders.

Emove Level: When GOING to each of these waypoints, try to FOLLOW WALLs. AVOID OBSTACLES if necessary. Finally in the goal CROSS DOOR.

Prim Level: Sets robot velocities (linear and rotational). Steers Robot and Moves it by a distance. Computes robot trajectory.

Servo Level: Processes low-level commands.

# Future Work

- Increase intelligence in higher nodes
- Work on the Value Judgement:
  - Learn from previous experiences;
  - Chose, from a set of algorithms, the most adequate to accomplish a particular task;
  - .....