



**FACULTAD DE INFORMÁTICA
UNIVERSIDAD POLITÉCNICA DE MADRID**

**PRESENTACIÓN DE LA TESIS DE MÁSTER
MASTER DE INVESTIGACIÓN EN INTELIGENCIA ARTIFICIAL**

IMPROVING ROBUSTNESS IN ROBOTIC NAVIATION BY USING A SELF-RECONFIGURABLE CONTROL SYSTEM

AUTOR: Arturo Bajuelos Castillo

TUTORES: Nik Swoboda y Ricardo Sanz Bravo

SEPTIEMBRE, 2011

ÍNDICE

1. Introducción
2. Estado del Arte
3. ASys y *The Operative Mind*
4. Plataforma de Hardware/Software
5. Misión de Higgs
6. Arquitectura de Control
7. Implementación
8. Experimentos y Resultados
9. Conclusión
10. Bibliografía

INTRODUCCIÓN

- Actualmente, vivimos rodeados de sistemas técnicos.
- Muchos de esos sistemas técnicos han remplazado la labor de seres humanos. Otros nos han permitido hacer cosas que antes eran inimaginables
- Las nuevas demandas de autonomía para estos sistemas son **difíciles de alcanzar**. Este trabajo se centra en el problema:
 - La manutención de la **robustez y autonomía** en sistema técnicos que interactúan con entornos **dinámicos, complejos e impredecibles**.
- Buen ejemplo: industria robótica.



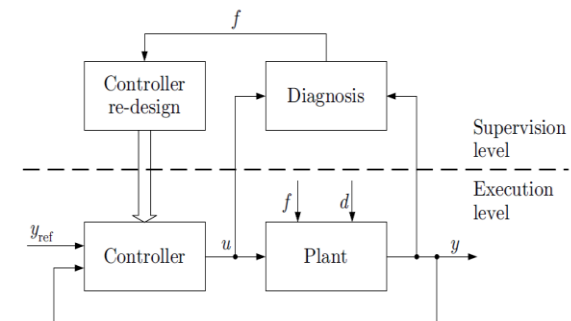
INTRODUCCIÓN

- Objetivo principal de la tesis:
 - **Diseño e implementación de una arquitectura de control para un robot móvil autónomo con capacidades de auto-reconfiguración.**
- Para alcanzar este objetivo, se realizaron la siguientes tareas:
 1. Estudio de las arquitecturas de agentes.
 2. Definición de una misión para un robot móvil, que requiera reconfiguración.
 3. Diseño de la arquitectura completa de control
 4. Implementación del sistema de control en un robot móvil.
 5. Experimentos y análisis de los resultados
- Otros objetivos la tesis:
 - Definir una guía de cómo emplear la introspección para aumentar la robustez en sistemas autónomos
 - Probar que las capacidades de introspección y auto-reconfiguración aumentan la robustez de sistemas de control robóticos.

ESTADO DEL ARTE

ARQUITECTURAS DE AGENTES

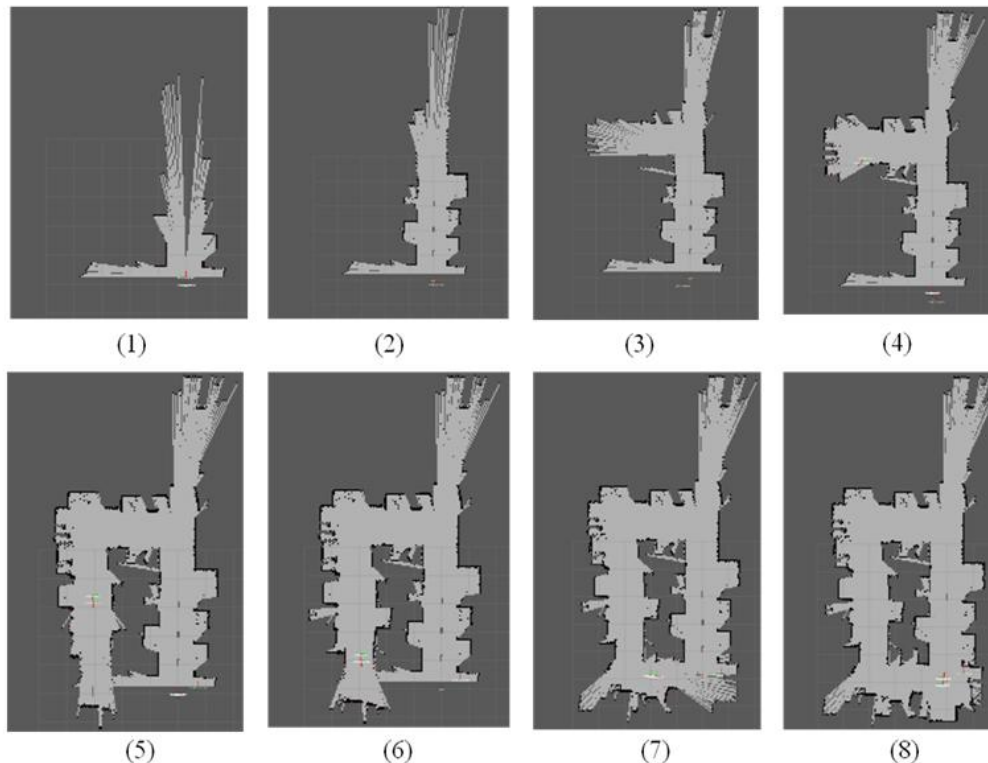
- Se define arquitectura de un agente como la “especificación de cómo un agente se puede descomponer en módulos y de como interactúan entre sí” (Maes, 1991). Se pueden categorizar en arquitecturas **deliberativas**, **reactivas** y **híbridas**.
- Las arquitecturas cognitivas** nos acercan al problema de autonomía en entornos complejos. Proponen procesos computacionales que se comportan como algunos sistemas cognitivos (como la mente humana).
- Sistema tolerantes a fallos. Tienen mecanismos que les permiten reaccionar a circunstancias y auto-adaptarse.



ESTADO DEL ARTE

TÉCNICAS PARA ROBOTS MÓVILES

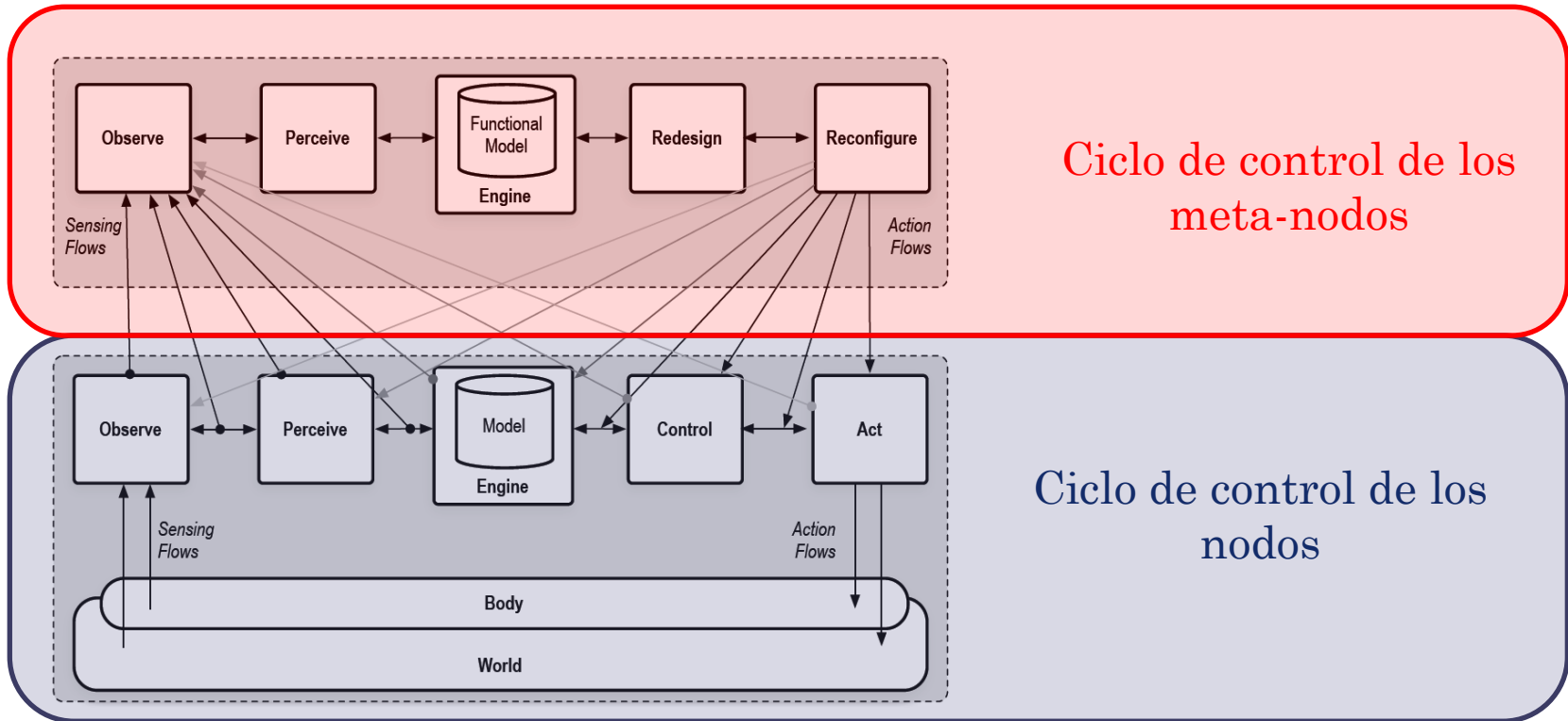
- **SLAM (Localización y Mapeo Simultaneo)** – técnica en que un robot construye un mapa de un entorno desconocido y, al mismo tiempo, se mantiene localizado en ese entorno.



ASYS Y THE OPERATIVE MIND

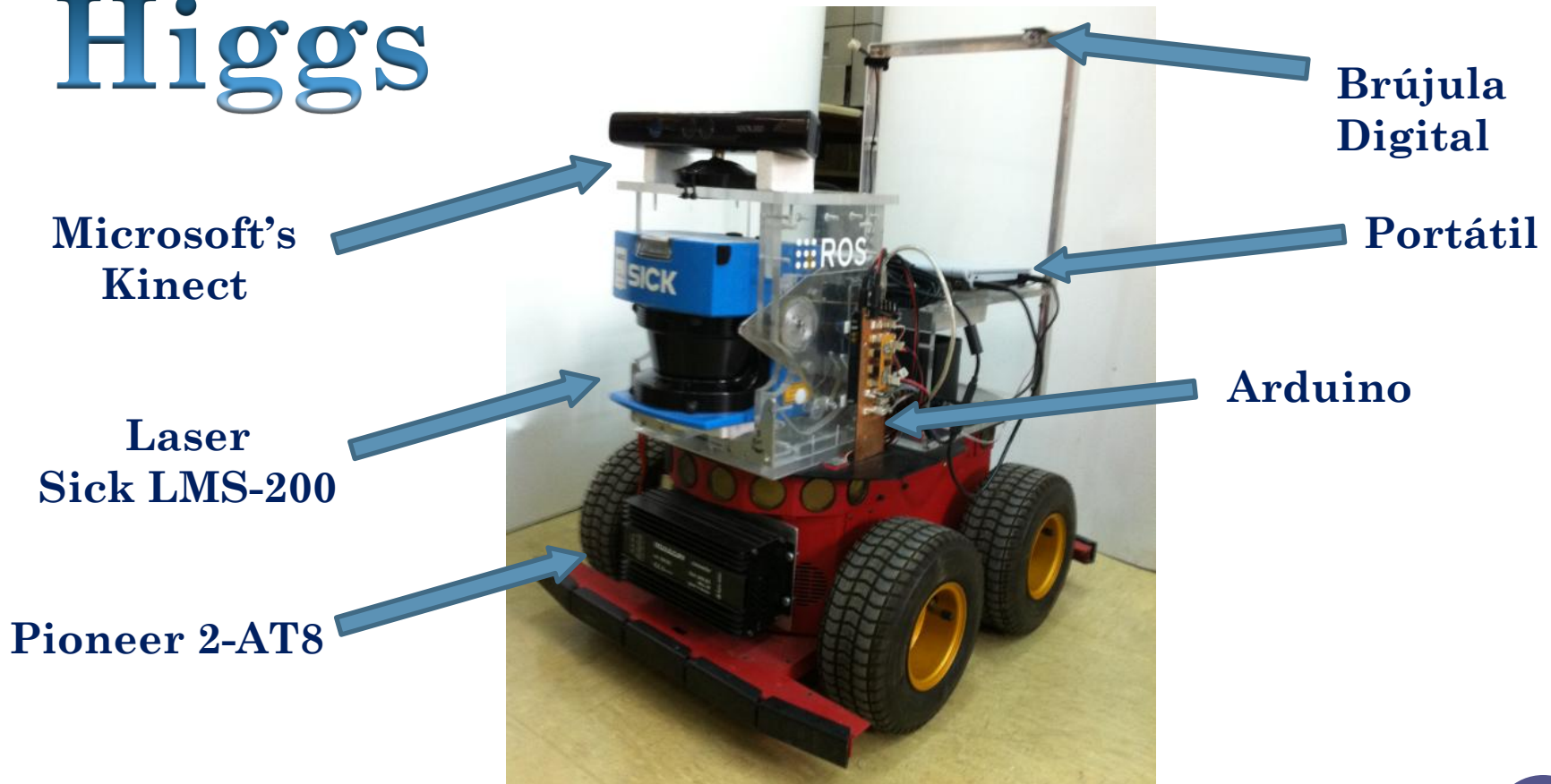
- **ASys** – proyecto a largo plazo del Laboratorio de Sistemas Autónomos (ASLab) de la Universidad Politécnica de Madrid. Su principal línea de investigación es el desarrollo de tecnología para la construcción de sistemas autónomos
- Uno de los objetivos de ASys: el uso de patrones de diseño, como vehículo para la explotación de arquitecturas reutilizables.
- Uno de los principales pilares de ASys: **un sistema verdaderamente autónomo utiliza modelos para realizar su actividad, pudiendo explotar modelos de sí mismo.**
- ***The Operative Mind***: arquitectura general que consiste en una red de nodos cognitivos, en cada uno realiza un ciclo de control, siguiendo un determinado patrón.

THE OPERATIVE MIND



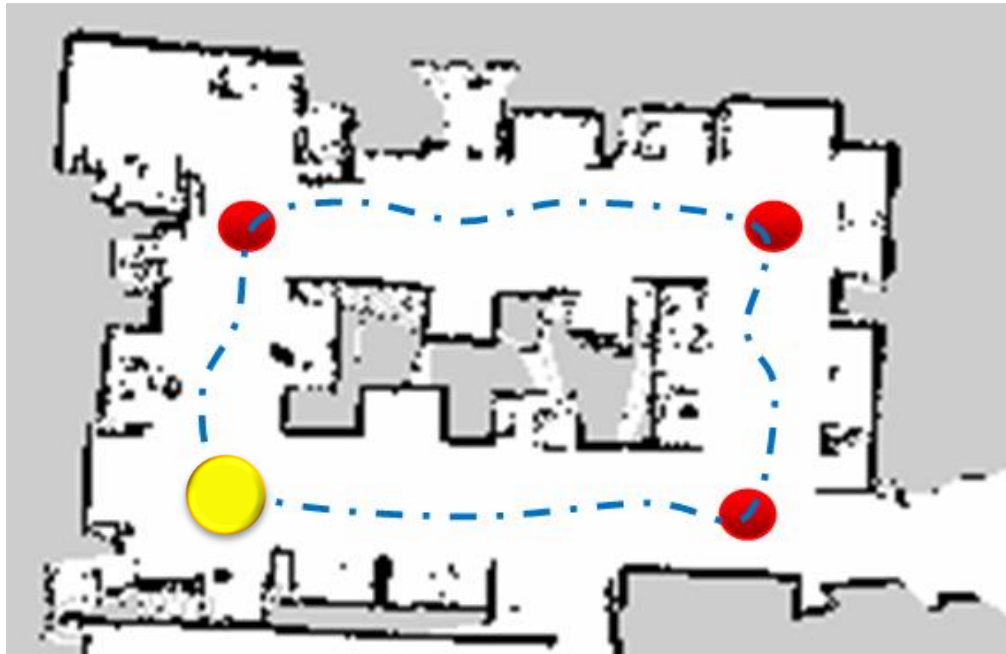
PLATAFORMA DE HARDWARE

Higgs



MISIÓN DE HIGGS

- La misión de Higgs consiste en una misión de vigilancia por la Sala de Calculo, del DISAM



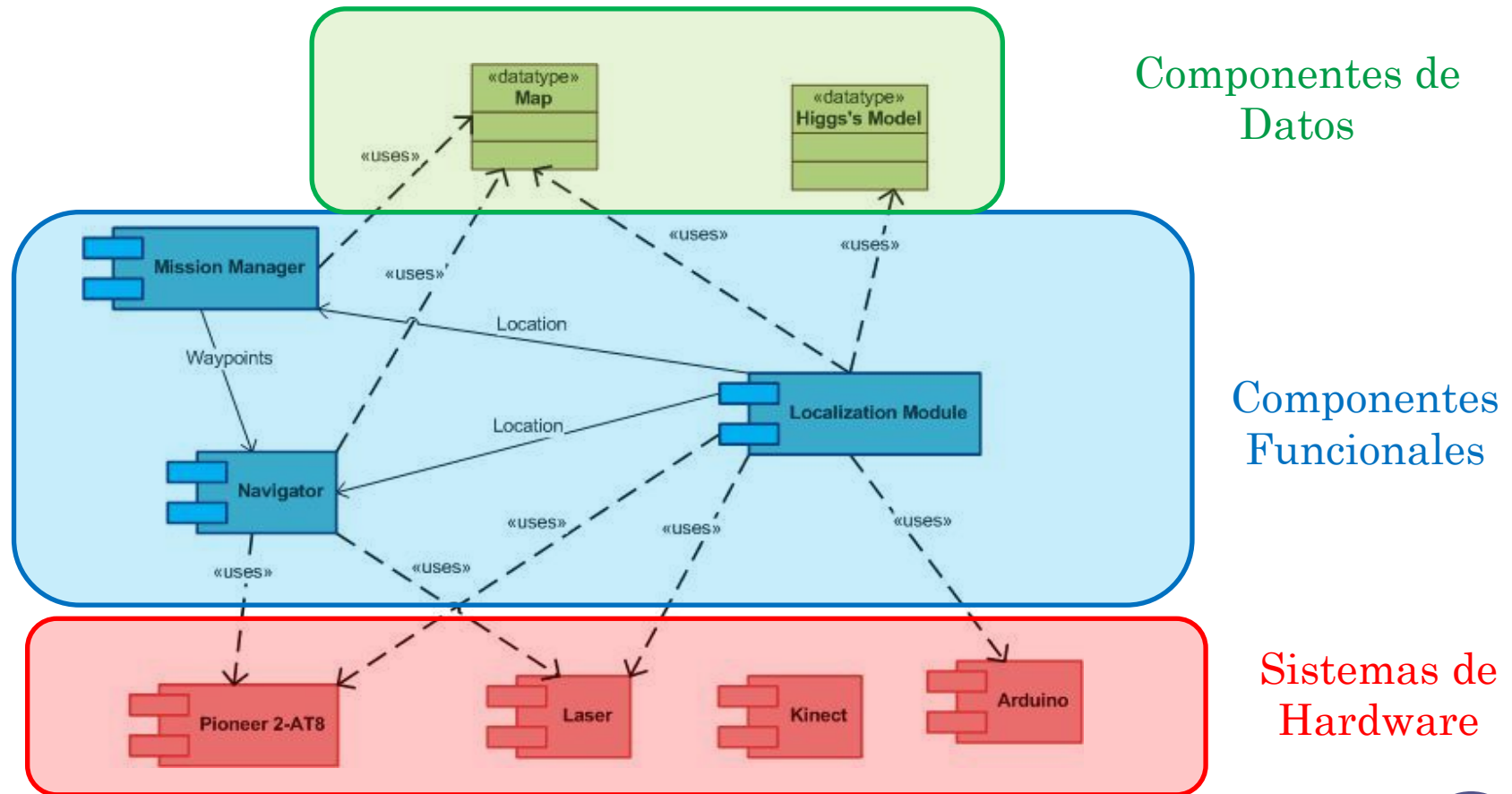
MISIÓN DE HIGGS

- Higgs debe utilizar su plataforma Pioneer para moverse y el laser como sensor principal para localizarse y evitar obstáculos.
- Debe disponer de un mapa y de saber su posición inicial.
- En un determinado instante, se va a simular un **fallo del laser**, que Higgs debe superar.
- Los objetivos principales de Higgs para la misión son:
 - **Higgs tiene que navegar por el pasillo rectangular de la Sala de Calculo, evitando obstáculos.**
 - **Higgs tiene que localizarse correctamente.**
 - **Higgs tiene que recuperarse del fallo del laser y continuar con su misión**

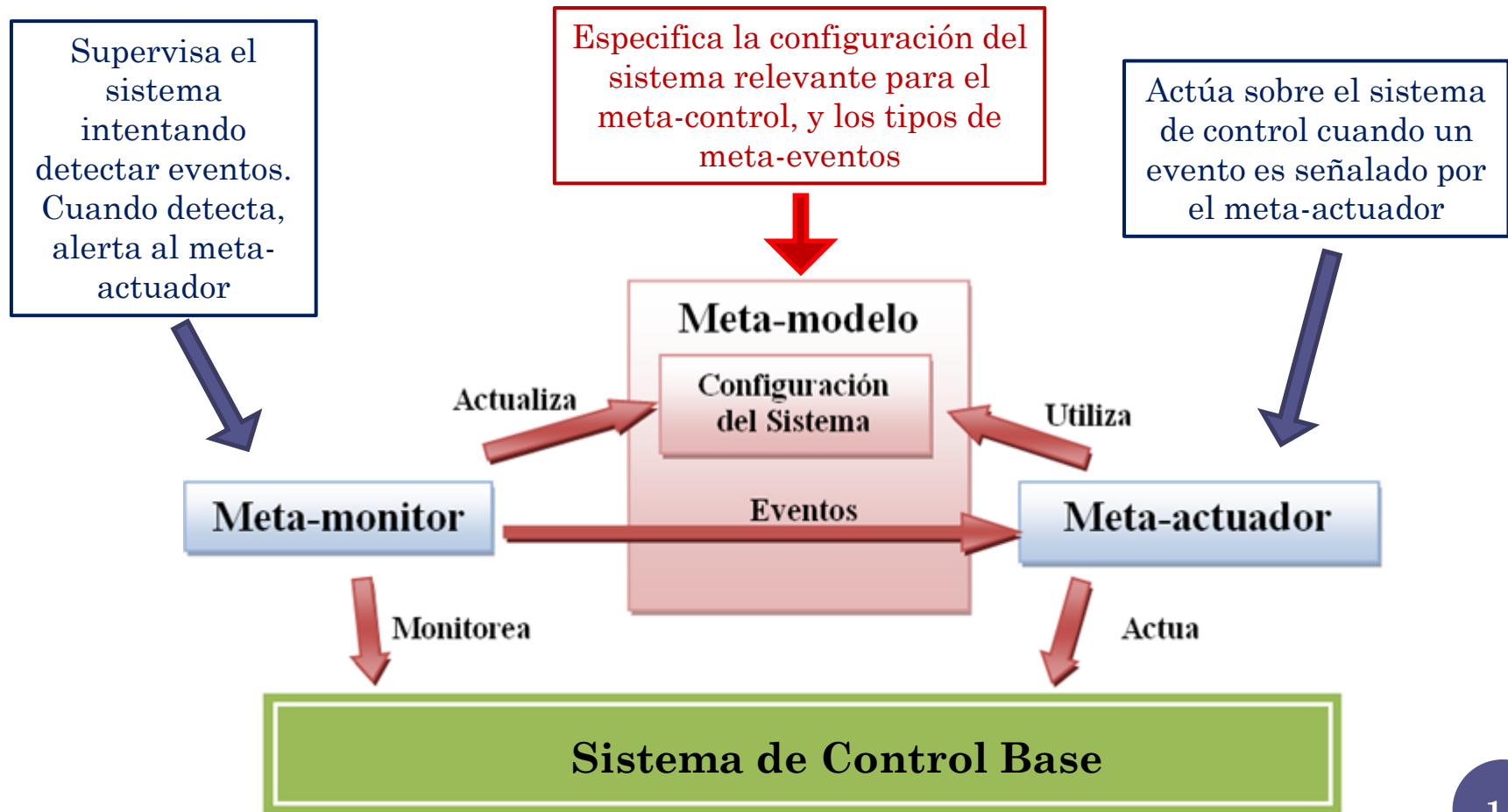
ARQUITECTURA DE CONTROL

- La arquitectura desarrollada se puede descomponer en 2 partes (o capas):
 - **La arquitectura base:** es responsable por la navegación de Higgs por el pasillo de la Sala de Calculo. Es incapaz de reconfigurarse, por eso es incapaz de superar el fallo del laser que va a ocurrir durante la ejecución de la misión.
 - **La arquitectura de meta-control:** especifica una capa adicional de meta-control que gestiona el sistema de control base. Este sistema de meta-control tiene la capacidad de reconfigurar todo el sistema, lo que permite superar el fallo del laser.

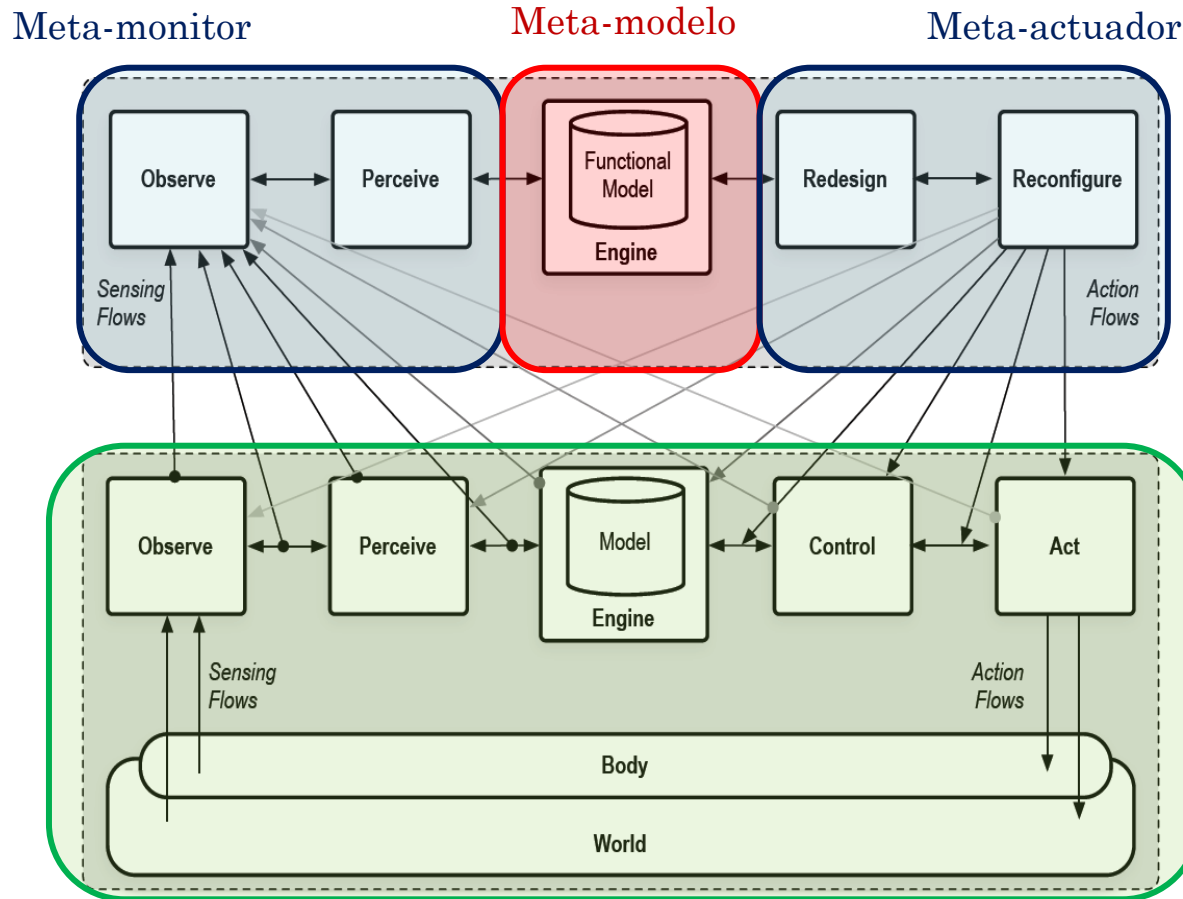
ARQUITECTURA DE CONTROL - BASE



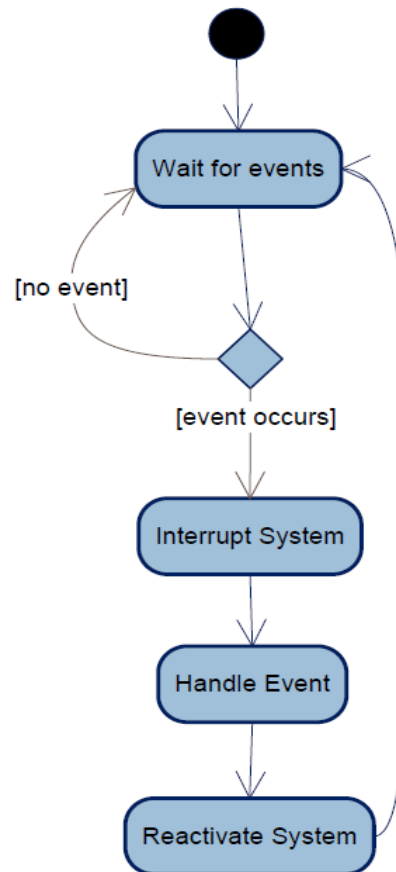
ARQUITECTURA DE CONTROL – CAPA DE META-CONTROL



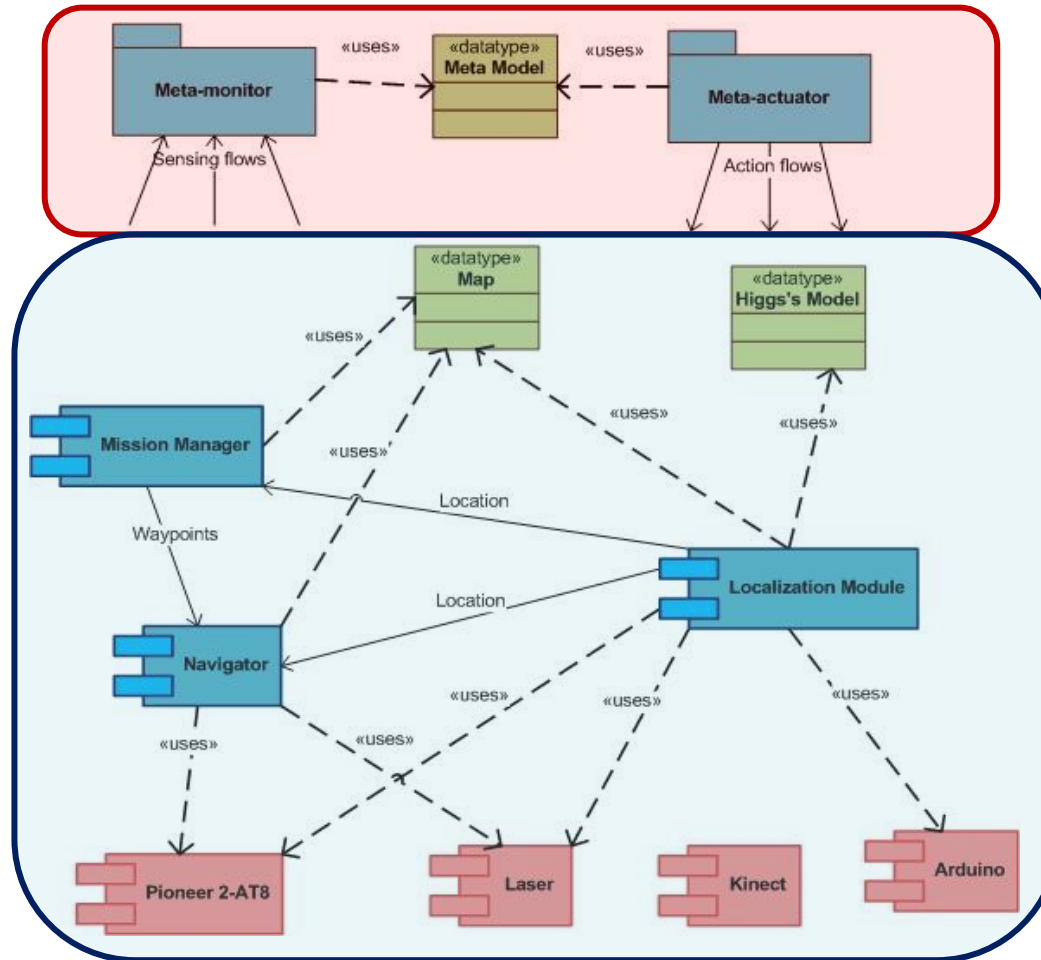
ARQUITECTURA DE CONTROL – MAPEO CON *THE OPERATIVE MIND*



ARQUITECTURA DE CONTROL – META-ACTUADOR



ARQUITECTURA DE CONTROL



Capa de meta-control

Arquitectura Base

IMPLEMENTACIÓN

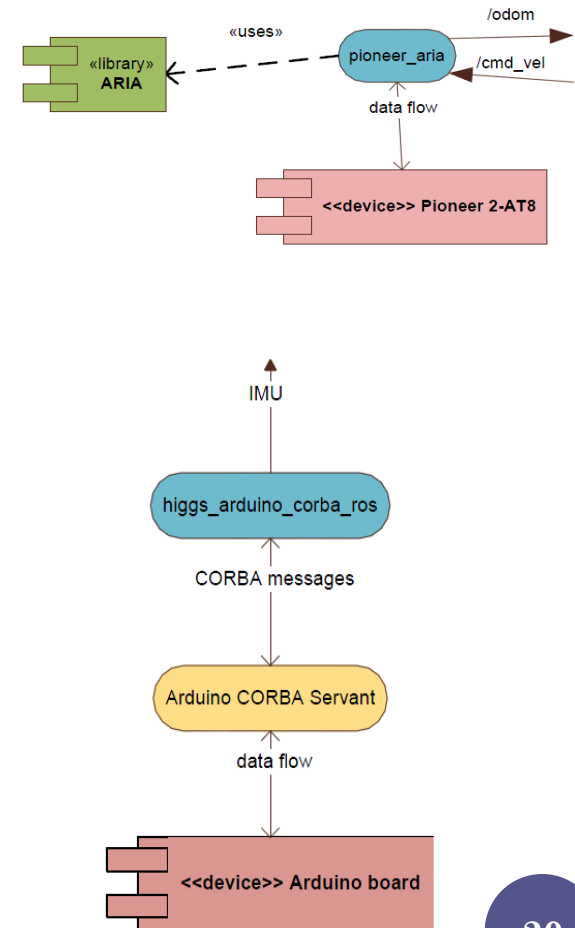
- El sistema de control basado en la arquitectura desarrollada fue implementado en ROS (Robot Operating System), que demostró tener grandes ventajas:
 - Presenta una estructura modular, con nodos ejecutables, que pueden ser mapeados a los componentes definidos por la arquitectura.
 - Cada nodo dispone de distintas interfaces, que pueden ser accedidas remotamente.
 - Proporciona varias implementaciones de algoritmos para navegación en robots móviles.
 - Proporciona varias utilidades para monitorizar nodos y el sistema de control

IMPLEMENTACIÓN – NODOS REUTILIZADOS

- **Driver Kinect:** Paquete *openni_camera* para obtener las imágenes RGB y la nube de puntos (*PointCloud2*). Paquete *pointcloud_to_laserscan* para convertir la nube de puntos (*PointCloud2*) en un barrido laser (*LaserScan*).
- **Driver laser:** paquete *sicktoolbox_wrapper* . El nodo publica un barrido laser (*LaserScan*)
- **EKF para estimación de la posición:** paquete *robot_pose_ekf*. Integra el valor de la odometria con la brújula.
- **Servidor del Mapa:** paquete *map_server*. Carga el mapa de un par de ficheros y emite el mapa para el sistema.

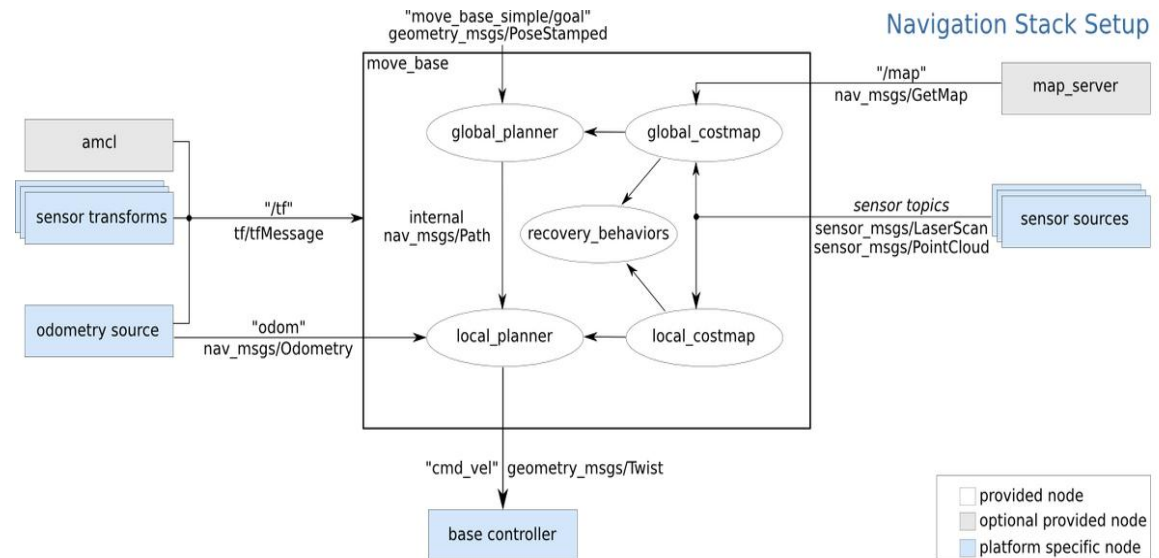
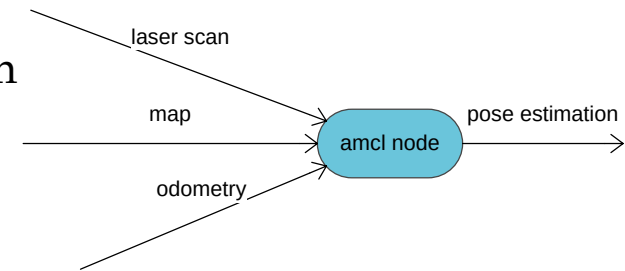
IMPLEMENTACIÓN – NODOS IMPLEMENTADOS

- **Driver de la plataforma base (Pioneer 2-AT8):** paquete *pioneer_aria*, basado en el paquete ROSARIA, que utiliza la librería ARIA de C++.
- **Driver Arduino:** paquete *higgs_arduino_corba_ros*. Consiste en un wrapper CORBA→ROS para reutilizar el driver de la placa Arduino ya implementado en CORBA por miembros del ASLab.



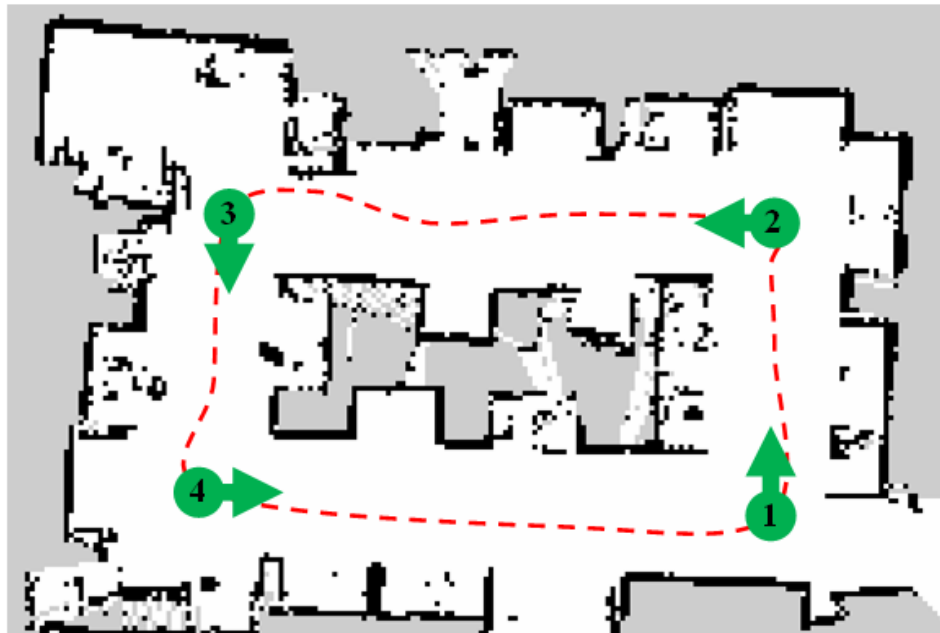
IMPLEMENTACIÓN – LOCALIZACIÓN Y NAVEGACIÓN

- **Componente para la localización:** paquete *amcl*. Utiliza el método de filtro de partículas con localización adaptativa de Monte Carlos (Thrun, Burgard y Fox en 2003)
- **Componente para la navegación:** paquete *move_base*. Alberga varios elementos, entre los cuales, un planificador global y local.



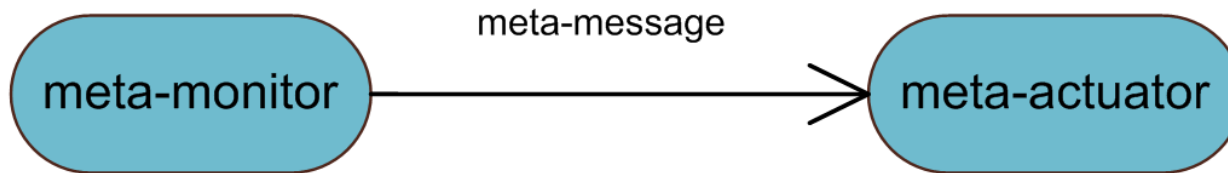
IMPLEMENTACIÓN – GESTOR DE LA MISIÓN

- **Gesto de la misión:** paquete *mission_manager*. Asigna varias posiciones destino al *move_base* para que el robot ejecute la trayectoria deseada por el pasillo rectangular de la Sala de Cálculo.



IMPLEMENTACIÓN – META-CONTROL

- Se ha desarrollado el paquete *meta_controller* que contiene :
 - Nodo ***meta_monitor*** : se subscribe al topic */rosout* para obtener los mensajes log de otros nodos. Monitoriza también la presencia de nodos en el sistema
 - Mensaje de tipo ***MetaMessage***: es publicado en el tópico */meta_events* por el *meta_monitor* y recibido por el *meta_actuator*. Contiene el nombre del evento detectado y información de la configuración del sistema.
 - Nodo ***meta_actuator***: Al recibir un mensaje de tipo *MetaMessage*, interrumpe el sistema, reconfigura el sistema según el evento y reactiva el sistema. Utiliza comandos de ROS para reconfigurar, como *rostopic kill*, *rostopic run* y *rostopic launch*.

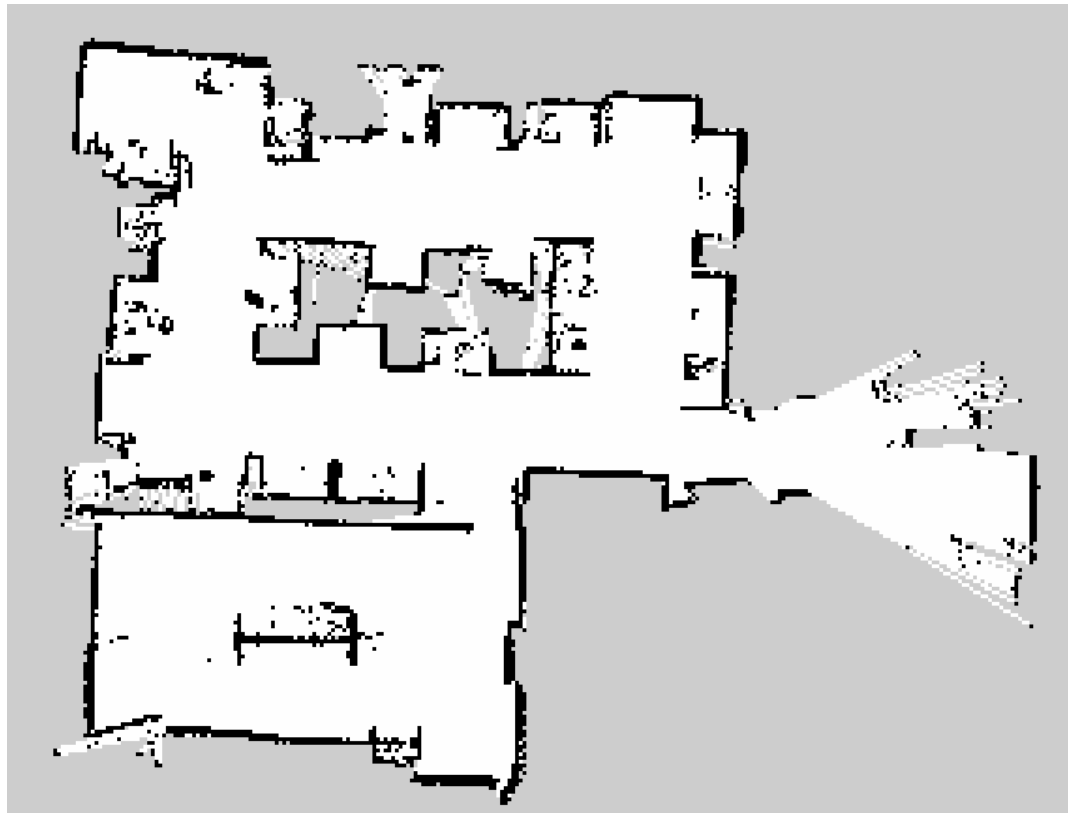


EXPERIMENTOS Y RESULTADOS

○ Experimentos realizados:

- Construcción del mapa de la Sala de Cálculo: se hizo un SLAM, utilizando el paquete *gmapping*, que tiene la implementación del algoritmo Gmapping desarrollado por Grisetti, Stachniss y Burgard en 2006.
- Experimentos con el sistema de control base para navegación sin meta-control.
- Experimentos con el sistema de control con meta-control y fallo simulado del laser – misión de Higgs
- Experimentos con un sistema de control en que el meta-control fue extendido para integrar “inteligentemente” ambos sensores (Kinect y Laser).

EXPERIMENTOS Y RESULTADOS – CONSTRUCCIÓN DEL MAPA



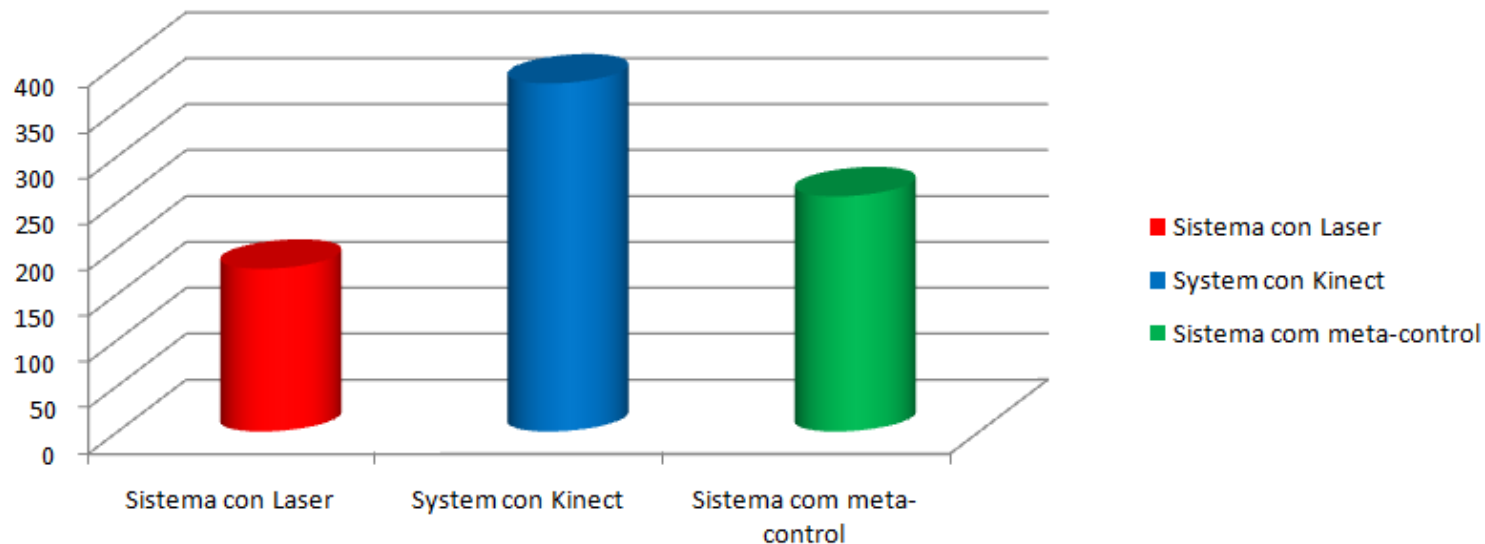
EXPERIMENTOS Y RESULTADOS – SISTEMA DE CONTROL BASE

	Lap 1	Lap 2	Lap 3	Lap 4	Lap 5	Lap 6	Lap 7	Mean $\pm$ St Dev
Laser	180.5	198.5	158.9	202.0	153.0	211.6	135.9	177.2 $\pm$ 26.4
Kinect	224.9	302.0	346.2	536.4	489.5	331.7	422.2	379.0 $\pm$ 101.3

EXPERIMENTOS Y RESULTADOS – SISTEMA DE CONTROL CON META-CONTROL

○ Ver vídeo

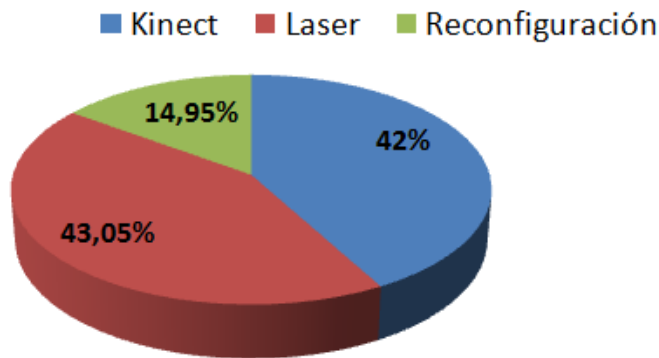
	Lap 1	Lap 2	Lap 3	Lap 4	Lap 5	Lap 6	Lap 7	Mean $\pm$ St Dev
Lap Time	208.8	242.9	238.3	214.5	276.7	367.0	245.0	256.2 $\pm$ 9.7
Reconfiguration Time	12.2	11.1	12.4	11.7	11.0	10.7	10.6	11.4 $\pm$ 0.7



EXPERIMENTOS Y RESULTADOS – SISTEMA DE HÍBRIDO

- Se extendió el meta-control para que el sistema utilice, por defecto, el Kinect.
- Cuando al robot se siente perdido, los nodos de meta-control lo detectan y reconfiguran el sistema para que utilice el Laser.
- Cuando el robot llega a un waypoint utilizando el Laser, los nodos de meta-control reconfiguran el sistema para que se utilice el Kinect.

Tiempo medio por vuelta:
263.1 sec



CONCLUSIÓN

- Se diseñó una arquitectura de control que define meta-elementos capaces de hacer el sistema tolerante a fallos y, de una manera muy básica, auto-consciente.
- Se implementó un sistema de control para un robot móvil, basado en la arquitectura, con capacidades de reconfiguración.
- Se realizaron experimentos en un robot móvil físico, para probar la arquitectura y el sistema de control desarrollado.
- Se demostró experimentalmente que:
 - El robot era capaz de localizarse y navegar en el ambiente especificado, evitando obstáculos dinámicos y estáticos.
 - El robot es robusto a fallos de sus dispositivos, gracias a la capa de meta-control del sistema de control.
 - La capa de meta-control puede ser fácilmente extensible para cambiar el comportamiento del sistema de control

CONCLUSIÓN – TRABAJOS FUTUROS

- La arquitectura de meta-control puede ser integrada en otras arquitecturas.
- El sistema de control pueden ser extendidos para mejorar aún más la robustez y autonomía del robot móvil.
- La capa de meta-control puede ser implementada utilizando técnicas más sofisticadas como procesos de Markov, Redes Bayesianas o Redes de Neuronas Artificiales.



BIBLIOGRAFÍA

- Albus, J., A. Meystel, A. Barbera, M. Del Giorno, y R. Finkelstein. "4D/RCD: A reference model architecture for unmanned vehicle systems version 2.0." Technical Report, National Institute of Standards and Technology, 2002.
- Blanke, M, M Kinnaert, J. Lunze, and M Staroswiecki. *Diagnosis and Fault-Tolerant Control*. Berlin: Springer-Verlag Berlin Heidelberg, 2006.
- Brooks, R.A. "A robust layered control system for a mobile robot." *IEEE Journal of Robotics and Automation*, 1986: 2(1):14-23.
- Conley, Ken. *Introduction to ROS*. 2011.
<http://www.ros.org/wiki/ROS/Introduction> (accessed 2011).
- Garcia, M. *Fusión sensorial en plataforma robótica móvil (Sensing fusion in a robotic mobile platform)*. 2009.
- Grisetti, Giorgio, Cyrill Stachniss, and Wolfram Burgard. "Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filters." *IEEE Transactions on Robotics*, 2006.

BIBLIOGRAFÍA

- Hernandez, C. "Estudio de las arquitecturas cognitivas: La necesidad de incorporar mecanismos de autoconsciencia en los sistemas de control inteligente." MSc Thesis, Madrid, Spain, 2007.
- Hernandez, C., I. Lopez, and R. Sanz. "The Operative Mind: a functional, computational and modeling approach to machine consciousness." *International Journal of Machine Consciousness* (World Scientific Publishing Company), 2009: 83-98.
- Maes, P. "The agent network architecture (ANA)." In *SIGART Bulletin*, 115-120. 1991.
- Sanz, R., and J. Zalewski. "Pattern-based control systems engineering." *IEEE Control Systems Magazine*, June 2003: 43-60.
- Thrun, S, W. Burgard, and D. Fox. *Probabilistic Robotics*. Cambridge, MA: MIT Press, 2005.
- Wise, Melonee. *ROS Tutorials: Understanding Topics*. 2011. (accessed 2011).