

# Asistente de diseño de interfaz de control, seguimiento y sharing de robots en tiempo real

Manuel López Urbina



Director: Arturo Morgado Estévez

Universidad de Cádiz  
Escuela Superior de Ingeniería  
Ingeniería Informática

# Índice

- 1 Índice
- 2 Introducción
- 3 Objetivos
- 4 Desarrollo
- 5 Temporización
- 6 Herramientas software
- 7 Comunicaciones
  - Salas
  - Subscripción
  - Flujo de datos
- 8 Interfaz
- Inicio
- Configuración
- Control
- Seguimiento
- Permisos
- Menú principal
- Panel de administración
- 9 Robot de pruebas
  - Tecnologías
  - Esquema
- 10 Conclusiones
- 11 Referencias

# Introducción

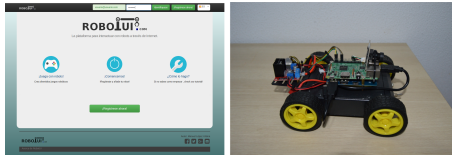


¿Por qué no añadir nuestros proyectos robóticos a la red?

# Objetivos

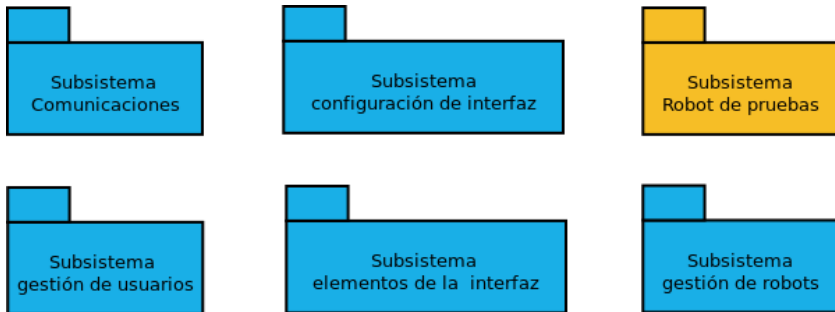
RobotUI es una combinación de un elemento software (aplicación web) y hardware (vehículo de pruebas y demostración).

- Desarrollar un sistema mediante el cual los usuarios puedan configurar una interfaz para el control de sus dispositivos robóticos.
- Permitir el control de los dispositivos robóticos haciendo uso de la interfaz previamente configurada tanto por su creador como por otros usuarios.
- Permitir la visualización del control que otros usuarios realizan de los dispositivos en tiempo real.
- No se debe requerir de amplios conocimientos de programación para su utilización.



## *RobotSharing*

# Subsistemas



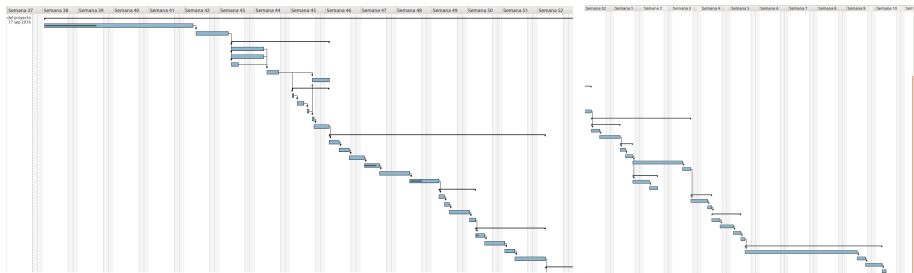
Modelo de ciclo de vida empleado:

- Desarrollo incremental
- Desarrollo en cascada

# Temporización

WBS	Nombre	Trabajo			
1	▼ RobotUI	135d 3h	1.5.8	▼ Personalización de la interfaz	9d 4h
1.1	Conocimiento del proyecto	22d	1.5.8.1	Personalización de acciones	1d 4h
1.2	Planificación y estudio del proyecto	5d	1.5.8.2	Personalización de sliders	2d
1.3	▼ Análisis de herramientas existentes	17d	1.5.8.3	Personalización de labels	2d
1.3.1	Estudio de Node.js	5d	1.5.8.4	Edición de la interfaz	4d
1.3.2	Estudio de Sails.js	5d	1.6	▼ Módulo de comunicaciones	18d
1.3.3	Estudio de Socket.io	2d	1.6.1	▼ Conexión cliente - robot	4d 7h
1.3.4	Códigos y pruebas	3d	1.6.1.1	Envío de órdenes al robot	2d
1.3.5	Estudio de MongoDB	2d	1.6.1.2	Captura de vídeo del robot	2d 7h
1.4	▼ Definición de requisitos	5d 6h	1.6.2	▼ Conexión cliente - servidor	3d
1.4.1	Arquitectura BBDD análisis	1d	1.6.2.1	Envío de vídeo capturado al servidor	1d 4h
1.4.2	Diseño de diagrama UML	2d	1.6.2.2	Envío de órdenes lanzadas al servidor	1d 4h
1.4.3	Requisitos funcionales	1d	1.6.3	Desarrollo de tests funcionales	8d
1.4.4	Requisitos no funcionales	6h 45min	1.6.4	Frontend - detalles de estilos	2d
1.4.5	Reunión de planificación con director del proyecto	1d	1.7	▼ Conexión servidor - cliente	4d
1.5	▼ Desarrollo aplicación	30d 4h	1.7.1	Difusión de vídeo a espectadores	2d
1.5.1	Sistema de autenticación - registro	2d	1.7.2	Difusión de comandos a espectadores	2d
1.5.2	Alta de dispositivos robóticos	2d	1.8	▼ Montaje del vehículo	3d
1.5.3	Internacionalización	1d	1.8.1	Búsqueda de elementos hardware	2d
1.5.4	Módulo de mensajes entre usuarios	3d	1.8.2	Montaje y conexiones	1d
1.5.5	Implementación de políticas de permisos	4d	1.9	▼ Programación del vehículo	4d 7h
1.5.6	Reunión de seguimiento con director del proyecto	3d 3h	1.9.1	Gpio	2d
1.5.7	▼ Elementos de la interfaz	5d 5h	1.9.2	Video streaming	1d 3h
1.5.7.1	Acciones	1d 1h	1.9.3	Fase de pruebas	1d 4h
1.5.7.2	Sliders	1d	1.10	Despliegue de la aplicación en producción	1d
1.5.7.3	Labels	1d 7h	1.11	▼ Documentación	23d
1.5.7.4	Video	1d 4h	1.11.1	Memoria	19d
			1.11.2	Resumen	2d
			1.11.3	Presentación	2d

# Temporización

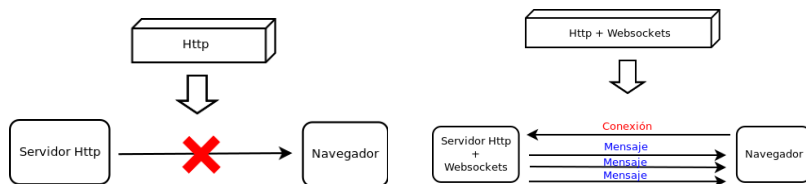


## Herramientas software



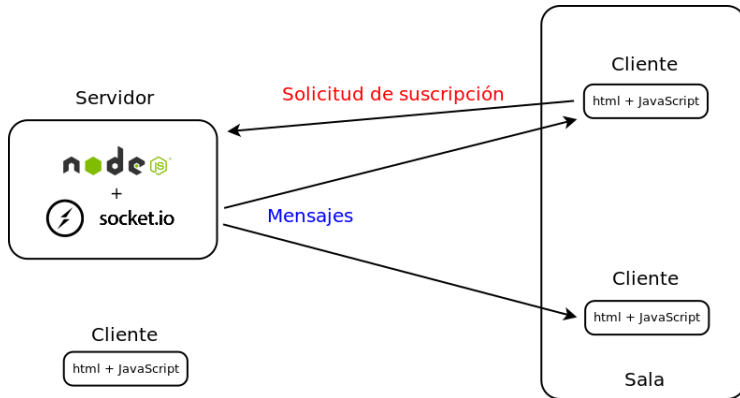


# Comunicaciones



Peticiones entre un navegador y un servidor HTTP con y sin el empleo de Websockets.

# Comunicaciones - Salas



Representación de una sala compuesta por dos clientes.

# Comunicaciones - Suscripción

Sails incorpora dos modalidades de suscripción:

- **Suscripción a clase:** Permite al socket escuchar la creación de nuevas instancias de modelo mediante el método *publishCreate()*.
- **Suscripción a modelo:** Permite al socket escuchar los cambios de modelos a través de los métodos *publishUpdate* y/o *publishDestroy* de una instancia o conjunto de instancias en concreto.

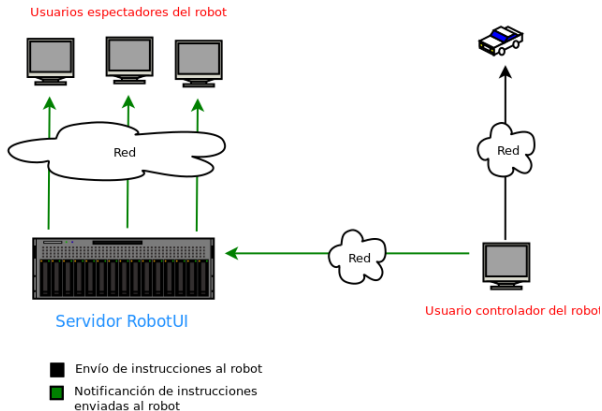
```
user_subscribe: function (req, res, next) {  
  //Update and destroy  
  User.find(function foundUsers(err, users) {  
    if (err) return next(err);  
    User.subscribe(req.socket, users);  
  });  
  
  //Create  
  User.watch(req);  
}
```

# Comunicaciones - Envío de mensajes

```
//Cambio de estado a online
User.update( user.id, { online: true }, function (err){
  if (err) return next(err);

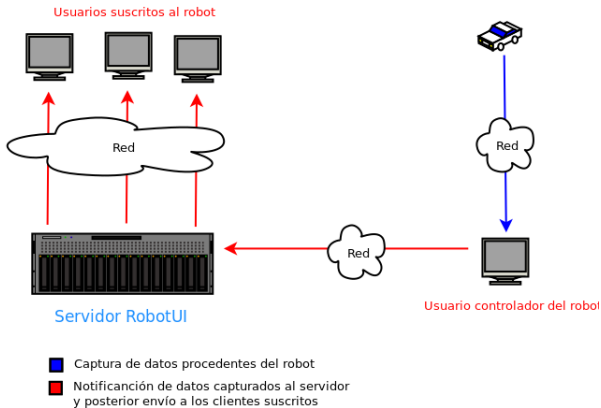
  // Informar a otros clientes (sockets abiertos)
  // que el usuario esta logueado
  User.publishUpdate(user.id, {
    loggedIn: true,
    id: user.id
  });
});
```

# Comunicaciones - Flujo de datos



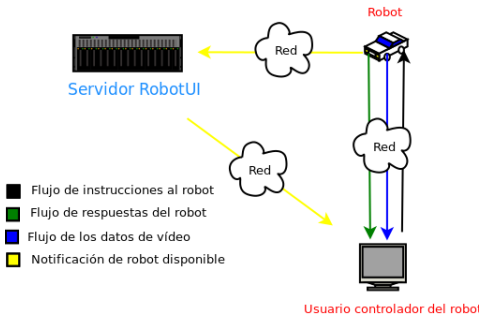
Envío de instrucciones al robot.

# Comunicaciones - Flujo de datos



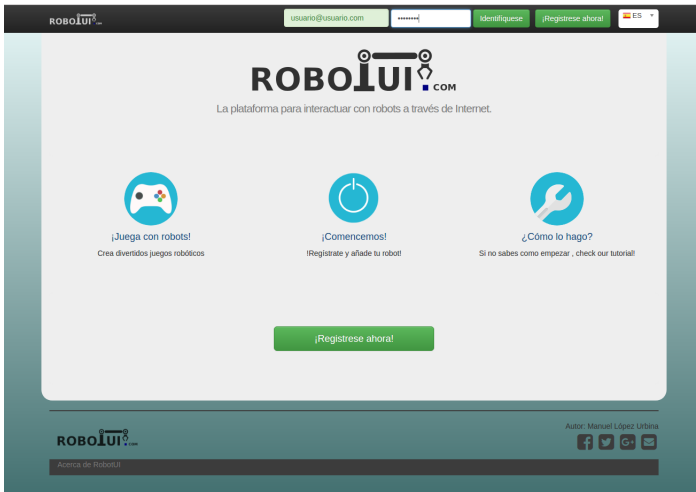
Captura y difusión de datos obtenidos del robot.

# Comunicaciones - Flujo de datos



Envío de notificación de robot disponible.

# Interfaz - Inicio

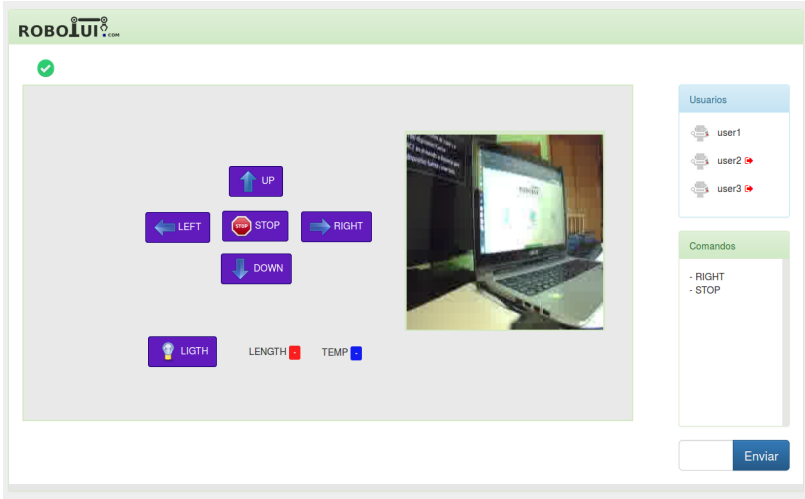




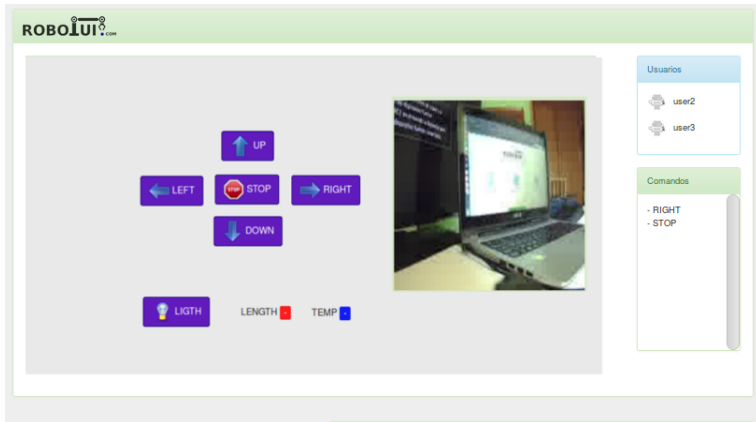
# Interfaz - Configuración



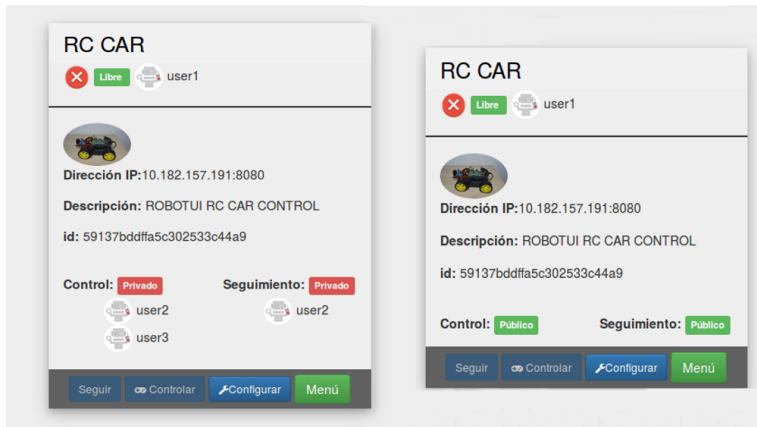
# Interfaz - Control



# Interfaz - Seguimiento



# Interfaz - Permisos





Panel informativo de un robot privado frente a uno público.

# Interfaz - Permisos

Permisos


**Público \* :**


Control: ☐ Seguimiento: ☐

Usuario	Nombre	Control	Seguimiento	Acciones
	user2	✓	✓	<a href="#" style="background-color: red; color: white; padding: 2px 5px;">Eliminar</a>
	user3	✓	✗	<a href="#" style="background-color: red; color: white; padding: 2px 5px;">Eliminar</a>

---

**Usuarios \* :**

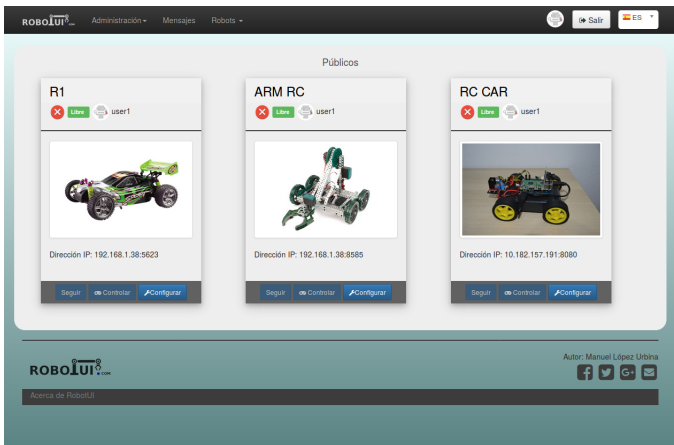
 user2 ✕

 user4 ✕

Añadir

Panel de configuración de permisos.

# Interfaz - Robots disponibles



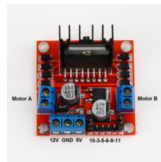
# Interfaz - Panel de administración

Robots										
Online	Estado	Nombre	Descripción	direction	Propietario	Conducción	Visión	Acciones		
	Libre	Buggy	Buggy	192.168.1.120:1234	user1	Público	Público	Interfaz	Editar	Eliminar
	Libre	ARM RC	ARM	10.182.157.191:4234	user1	Privado	Privado	Interfaz	Editar	Eliminar
	Libre	RC Car	RC ROBOTUI CAR	192.168.1.130:8085	user1	Público	Público	Interfaz	Editar	Eliminar

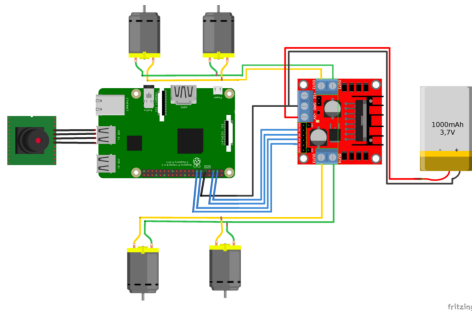
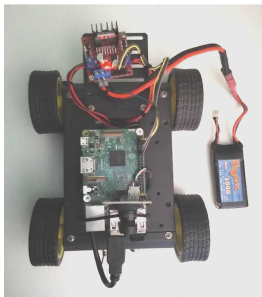
Usuarios				
Online	Nombre	Email	Roll	Acciones
	user1	user1@user1.com		<a href="#">Añadir robot</a> <a href="#">Mostrar robots</a> <a href="#">Editar</a> <a href="#">Eliminar</a>
	user2	user2@user2.com		<a href="#">Añadir robot</a> <a href="#">Mostrar robots</a> <a href="#">Editar</a> <a href="#">Eliminar</a>
	user3	user3@user3.com		<a href="#">Añadir robot</a> <a href="#">Mostrar robots</a> <a href="#">Editar</a> <a href="#">Eliminar</a>
	user4	user4@user4.com		<a href="#">Añadir robot</a> <a href="#">Mostrar robots</a> <a href="#">Editar</a> <a href="#">Eliminar</a>

# Robot de pruebas - Tecnologías





# Robot de pruebas - Esquema



# Conclusiones

La realización del proyecto *RobotUI: Asistente de diseño de interfaz de control, seguimiento y sharing de robots en tiempo real* como trabajo de fin de carrera, se ha caracterizado por:

- La elaboración de un vehículo de pruebas haciendo uso de una Raspberry Pi 3 Model B.
- Aprendizaje a la utilización del framework Sails.js.
- Trabajo con eventos en tiempo real mediante el empleo de WebSockets, biblioteca Socket.io.
- Empleo de una base de datos no relacional como Mongo DB.
- Transmisión de gran cantidad de datos entre cliente servidor y servidor cliente. Streaming de vídeo y emisión de comandos entre otros datos.

- 
 Mike Cantelon, Alex R. Young, Marc Harter, T.J. Holowaychuk, and Nathan Rajlich.  
*Node.js in Action. Manning Publications, 2017.*
- 
 Irl Nathan Mike McNeil.  
*Sails.js in Action. Manning Publications, 2017.*
- 
 Irl Nathan.  
 Activityoverlord, an application to learn sails.js.  
<https://github.com/irlnathan/activityoverlord>. Visitado el 19-01-2017.
- 
 Andrew Lombardi.  
*WebSocket: Lightweight Client-Server Communications. O'Reilly Media, 2012.*
- 
 Official documentation.  
 Node JS Documentation. <https://nodejs.org/es/docs/>. Visitado el 02-05-2017.
- 
 Official documentation.  
 Sails JS Documentation. <https://sailsjs.com/documentation/reference>.  
 Visitado el 14-03-2017.
- 
 Rohit Rai.  
*Socket.IO Real-Time Web Application Development. Packt, 2013.*
- 
 Lakshminarasimhan Srinivasan, Julian Scharnagl, and Klaus Schilling.

Analysis of WebSockets as the New Age Protocol for Remote Robot Tele-operation. Technical report, University of Wuerzburg, Department of Robotics and Telematics, 11 2013.



Lakshminarasimhan Srinivasan, Julian Scharnagl, Zhihao Xu, Nicolas Faerber, Dinesh K. Babu, and Klaus Schilling.

Design and Development of a Robotic Teleoperation System using Duplex WebSockets suitable for Variable Bandwidth Networks. Technical report, University of Wuerzburg, Department of Robotics and Telematics, 11 2013.



Nazirah Ahmad Zaini, Norliza Zaini, Mohd Fuad Abdul Latip, and Nabilah Hamzah.

Remote Monitoring System based on a Wi-Fi Controlled Car Using Raspberry Pi. Technical report, Universiti Teknologi MARA (UiTM) Shah Alam, Malaysia, Faculty of Electrical Engineering, 12 2016.

# Asistente de diseño de interfaz de control, seguimiento y sharing de robots en tiempo real

Manuel López Urbina



Director: Arturo Morgado Estévez

Universidad de Cádiz  
Escuela Superior de Ingeniería  
Ingeniería Informática