TRABAJO FIN DE MÁSTER

MÁSTER EN INGENIERÍA INFORMÁTICA

Lazarillo

Plataforma robótica con Edge Computing para guía de caminos

Autor
Adrián Morente Gabaldón

Director
Juan José Ramos Muñoz

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

—

Granada, septiembre de 2022

Lazarillo: plataforma robótica con Edge Computing para guía de caminos

Adrián Morente Gabaldón

Palabras clave: palabra_clave1, palabra_clave2, palabra_clave3, ...

Resumen

Poner aquí el resumen.

Project Title: Project Subtitle

First name, Family name (student)

Keywords: Keyword1, Keyword2, Keyword3, ...

Abstract

Write here the abstract in English.

Yo, **Adrián Morente Gabaldón**, alumno de la titulación Máster en Ingeniería Informática de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI 77139229N, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Adrián Morente Gabaldón

Granada a 9 de septiembre de 2022.

D. **Nombre Apellido1 Apellido2 (tutor1)**, Profesor del Área de XXXX del Departamento YYYY de la Universidad de Granada.

Informa:

Que el presente trabajo, titulado ***Título del proyecto, Subtítulo del proyecto***, ha sido realizado bajo su supervisión por **Nombre Apellido1 Apellido2 (alumno)**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a X de mes de 201 .

Los directores:

Nombre Apellido1 Apellido2 (tutor1) **Nombre Apellido1 Apellido2 (tutor2)**

Agradecimientos

Poner aquí agradecimientos...

Índice general

Introducción

Lazarillo se trata de una plataforma robótica abierta cuyo propósito es proporcionar una arquitectura solvente y extensible que permita añadir nuevas características y utilidades.

Contiene un sistema operativo abierto basado en GNU/Linux y *The Yocto Project*, además de distintos servicios implementados con lenguajes de programación diferentes, mostrando así la interoperabilidad del software. Goza de conectividad inalámbrica, se conecta con un servidor que hace las veces de centro computacional en un paradigma de *edge computing* y está dotado de un panel web de administración (desde el cual se pueden enviar acciones remotas al robot).

Aunque se trata de una plataforma extensible y de propósito múltiple, el uso inicial para el que fue ideado es el de actuar como **asistente** y **guía** dentro de un espacio cerrado (ya podemos ver que el título asignado al proyecto es un pequeño guiño a la literatura española).

Capítulo 1

Estado del arte

TODO: listar robots existentes, describiendo el grado de extensibilidad de cada uno de ellos

TODO: listar plataformas móviles como la de una Roomba, diciendo por qué no podemos usarla (código cerrado)

TODO: describir sucíntamente ROS y por qué no es necesario usarlo existiendo Redis

TODO: listar placas como RPI

TODO: listar sistemas operativos para RPI como Raspbian

TODO: describir uso de yocto en lugar de Buildroot y otras alternativas

Capítulo 2

Especificación y requisitos

Se desea disponer de una plataforma robótica extensible, libre y abierta; que permita una buena extensibilidad de nuevas características mediante software. Además, se propone la implementación de un uso concreto para esta plataforma. En este capítulo listaremos los distintos requisitos y puntualizaremos sobre cada una de las decisiones tomadas para satisfacerlos.

2.1. Requisitos generales

2.1.1. Licencias

Se trata de un proyecto de software **libre** y de **código abierto**. Para ello, se publica bajo una licencia *GPLv3* en un repositorio público en *Github*.

Además de *Github*, existen otras alternativas de repositorios públicos que permiten utilizar *git* para control de versiones (como *Bitbucket* y *Gitlab* entre otras). El porqué de utilizar *Github* es meramente por aprovechar la licencia *premium* que se provee a los estudiantes de la UGR simplemente por matricularse; permitiendo así tener algunos repositorios privados a nuestra disposición.

Con el código público y la licencia elegida, cualquier usuario podrá descargar el software, compilarlo, ejecutarlo e incluso añadir modificaciones al código para ser probadas e integradas en la plataforma final. Para ello, en el propio repositorio se pone a disposición del interesado un fichero **README.md** con los pasos necesarios.

2.1.2. Especificaciones

El término *plataforma robótica* puede ser demasiado amplio para su manejo, por lo que en esta subsección definiremos algunos de los puntos más interesantes.

Extensibilidad

Ya que se desea disponer de una plataforma extensible cuyo comportamiento y características puedan ampliarse a través de software, se debe dotar al robot

de una arquitectura que permita este crecimiento.

TODO: explicar orquestación de servicios y mensajería.

Conectividad

Como plataforma inteligente y conectada que utiliza el paradigma del ***edge computing***, sabemos que el robot ha de ser un dispositivo embebido con conexiones al exterior como Bluetooth y/o WiFi. Que éstas vengan implícitas en la plataforma hardware utilizada facilitará mucho el trabajo, ya que nos ahorramos el hecho de tener que soldar componentes. Utilizaremos para ésto una *Raspberry Pi Model 3 B*, que pese a no ser el último modelo de la marca *Raspberry*, es la que tengo a disposición en casa. Además, satisface las necesidades de conectividad que comentábamos.

Por otro lado, en cuanto al requisito de computación en el borde, la plataforma deberá contener uno o más servicios que establezcan la comunicación con el servidor, además de enviar y recibir mensajes. Esta conexión con el servidor se realizará utilizando el protocolo ***Websockets***, ya que
TODO: (explicar bondades de WS).

Interfaz y experiencia de usuario

El robot contará con una pantalla táctil con la que proveerá la información necesaria al usuario (en función de las aplicaciones que necesite o decidan integrarse en el robot). Cualquier pantalla táctil que permita su conexión con la *Raspberry* debería servir, por lo que no entraremos a detallar limitaciones hardware. Para la interacción del usuario, el sistema contará con una **aplicación embebida de entorno gráfico** que permita el uso del robot. La tecnología usada por esta aplicación, se ha decidido que sea el framework de ***Qt***, que pese a tener modalidades de licencias de pago, permite un uso *open source*. Además, se trata de una tecnología con una amplia comunidad, una documentación muy rica y un gran soporte para dispositivos embebidos. Por otro lado, dispongo de amplia experiencia con dicha herramienta, lo que agiliza enormemente el tiempo de desarrollo.

Otra característica que sería de agradecer en la plataforma tiene que ver con la **reproducción de sonidos** que faciliten la comunicación con el usuario, así como la **accesibilidad**. No todo el mundo goza de capacidad visual o simplemente no están habituados a interfaces táctiles, por lo que emitir alertas y sonidos descriptivos facilitaría llegar a más usuarios de forma plena y satisfactoria.

Gestión experta y mantenimiento

Como hemos comentado en secciones anteriores, el robot goza de hardware provisto de conectividad inalámbrica. En este punto haremos uso de esta característica para ofrecer un método de mantenimiento, supervisión y gestión del robot, por parte de alguna "mano experta".

De forma similar a lo que comentamos sobre *Websockets* en el apartado de **Conectividad**, aquí necesitaremos de un método de administración de los distintos robots existentes desde un portal web externo a ellos. Un técnico encargado de gestionar los robots, accederá a una web alojada en el servidor externo que realiza *la computación en el borde*. Ésta será una web sencilla utilizando el protocolo común de *HTTPS*.

Inicialmente, este portal web servirá para listar los dispositivos conectados (es decir, los robots que han sido provisionados con el software de *Lazarillo* y se encuentran en funcionamiento), pero posteriormente permitirá enviar acciones remotas desde el servidor al robot (como reinicios, actualizaciones de software, acciones concretas a realizar por el robot, etc.).

Movilidad

El factor determinante que diferenciará a nuestro robot de un sistema empotrado inmóvil será la capacidad de desplazarse por el entorno. Ya sea para un uso u otro, el robot deberá venir dotado de un sistema hardware que le permita avanzar por el plano, conteniendo elementos como motores y ruedas o cintas móviles.

Además, si se desea que el robot sea **inteligente** y reconozca el entorno por el que se está moviendo, deberá dotarse de algún sistema de reconocimiento como **sensores de proximidad** o **cámaras**. Respecto a esto, si queremos seguir el paradigma de *edge computing* como venimos comentando, el procesamiento de estas señales podría realizarse en el servidor en lugar de en el propio robot, lo que también liberaría a la plataforma hardware del robot de una buena parte de la computación.

2.2. Objetivos opcionales

(TODO: listar cosas que podrían hacerse para las que no habrá tiempo.)

Capítulo 3

Implementación

En este capítulo comentaremos a fondo las decisiones tomadas que hemos comentado previamente en base a la especificación, además de describir las implementaciones propuestas con las distintas tecnologías utilizadas.

Pondremos el foco no solo en el software implícito en el robot sino también a los componentes desarrollados externos a él, pero que también conforman la arquitectura completa de la plataforma.

Las ideas que por una razón u otra no hayan llegado a implementarse a tiempo, serán definidas en profundidad en apartados posteriores.

3.1. Sistema operativo

Para dotar a la *Raspberry* de un sistema operativo ligero, extensible y hecho a medida, decidimos obviar sistemas ya existentes como *Raspbian* (S.O. de culto para la mayoría de usuarios de este computador) (TODO: ref a Raspbian) y confeccionar el nuestro propio a través de una herramienta libre y gratuita como es *The Yocto Project* (TODO: ref a yocto).

¿Por qué omitimos un sistema operativo que ya existe?

Pues bien, la respuesta es fácil. *Raspbian* es un sistema operativo de uso general que permite utilizar la *Raspberry* como cualquier ordenador normal, ya sea para ofimática, desarrollo de software, consumo de recursos multimedia o incluso videojuegos. Esto provoca que el sistema operativo en cuestión venga con demasiado *bloatware* (TODO: poner en el glosario) preinstalado de base; y llevaría más tiempo modificar la imagen de *Raspbian* para que no contenga todo el software no deseado que confeccionar un sistema operativo a medida.

Además, deseamos que el robot solo muestre una aplicación embebida en pantalla en lugar de un entorno de escritorio normal, por lo que podemos prescindir de este entorno completo y configurar nuestro nuevo sistema para que solo muestre una aplicación y ahorre tiempo en el arranque.

3.2. Arquitectura de Lazarillo

En esta sección enumeraremos los distintos servicios y módulos específicos creados para Lazarillo y describiremos el propósito para el que han sido programados.

(TODO: insertar diagrama de arquitectura)

3.2.1. Servicios internos

Veamos ahora una breve explicación del propósito que satisfacen los módulos software internos del robot.

lazarillo-hmi

Se trata de la aplicación que muestra la interfaz táctil al usuario.

messages-definition

Este módulo no es en sí un servicio sino una **librería** donde se definen los mensajes que utilizan internamente el resto de servicios para comunicarse.

service-base

Este módulo define el esqueleto abstracto de cada uno de los servicios del robot. Heredando el comportamiento de esta librería, cada uno de los nuevos servicios se ahorran procedimientos rutinarios como instanciar la conexión con la base de datos y el bróker de mensajería.

web-gateway

Este servicio es el *puerto de entrada* a Lazarillo. Realiza la comunicación mediante *Websockets* con ***lazarillo-admin***, el portal web a través del cual un técnico puede administrar los distintos robots. ***Web-gateway*** actúa como intérprete de los mensajes provenientes del *socket* y los publica en la mensajería interna del robot (basada en *Redis*) para informar al resto de servicios de cualquier acción emitida por el servidor.

3.2.2. Componentes externos

Como venimos comentando, además del software embebido en el robot, disponemos de otros módulos externos a él que completan la plataforma.

lazarillo-admin

Este es el título asignado al servicio web que tanto venimos comentando a lo largo de este documento, a través del cual podemos visualizar y gestionar los distintos robots conectados al servidor.

Está programado con *NodeJS* (TODO: ref a node) y utiliza librerías para hacer las veces de servidor con *HTTPS* (para la web) y con *Websockets* (para

la comunicación con el robot).

Capítulo 4

Trabajos futuros

Un buen *brainstorming* sobre el desarrollo de una idea novedosa y (se supone que) útil siempre es motivador y consigue animar a un programador a llevar la idea a la práctica, intentando la consecución del mayor número de requisitos posibles. Sin embargo, el tiempo siempre es uno de los factores más determinantes a la hora de dictaminar cómo de lejos se llega con el proyecto.

Dicho esto, en este capítulo comentaremos cuáles de las ideas iniciales no llegaron a finalizarse a tiempo para la entrega del proyecto, y cuáles se decidieron postergar en el tiempo para un futuro desarrollo.

Cabe destacar que dada la naturaleza libre del proyecto, cualquier interesado o interesada podría continuar con la parte del desarrollo que más le interese; ya que esta idea nació como una herramienta abierta a la que poder contribuir y de la que cualquiera pueda sacar utilidad.

Entrando ya en materia, dividiremos este capítulo en breves secciones relacionadas con las ya vistas en el capítulo de **Implementación**, detallando los cabos sueltos que hayan quedado en cada uno de ellos.

4.1. Sistema operativo

4.2. Servicios propios

4.3. Otros

TODO: rellenar secciones (tras terminar capítulo de Implementación)

Capítulo 5

Conclusiones