GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

TRABAJO FIN DE GRADO

NAVEGACIÓN EXTERNALIZADA DE ROBOTS MÓVILES EN UN ALMACÉN

Autor: Nerea Zabala Orive
Director: Álvaro Sánchez Miralles

Madrid
Junio de 2019

AUTORIZACIÓN PARA LA DIGITALIZACIÓN, DEPÓSITO Y DIVULGACIÓN EN RED DE PROYECTOS FIN DE GRADO, FIN DE MÁSTER, TESINAS O MEMORIAS DE BACHILLERATO

1º. Declaración de la autoría y acreditación de la misma.

El autor D. NEREA ZABALA ORIVE

DECLARA ser el titular de los derechos de propiedad intelectual de la obra: NAVEGACIÓN EXTERNALIZADA DE ROBOTS MÓVILES EN UN ALMACEN

que ésta es una obra original, y que ostenta la condición de autor en el sentido que otorga la Ley de Propiedad Intelectual.

2º. Objeto y fines de la cesión.

Con el fin de dar la máxima difusión a la obra citada a través del Repositorio institucional de la Universidad, el autor **CEDE** a la Universidad Pontificia Comillas, de forma gratuita y no exclusiva, por el máximo plazo legal y con ámbito universal, los derechos de digitalización, de archivo, de reproducción, de distribución y de comunicación pública, incluido el derecho de puesta a disposición electrónica, tal y como se describen en la Ley de Propiedad Intelectual. El derecho de transformación se cede a los únicos efectos de lo dispuesto en la letra a) del apartado siguiente.

3º. Condiciones de la cesión y acceso

Sin perjuicio de la titularidad de la obra, que sigue correspondiendo a su autor, la cesión de derechos contemplada en esta licencia habilita para:

a) Transformarla con el fin de adaptarla a cualquier tecnología que permita incorporarla a internet y hacerla accesible; incorporar metadatos para realizar el registro de la obra e incorporar "marcas de agua" o cualquier otro sistema de seguridad o de protección.
b) Reproducirla en un soporte digital para su incorporación a una base de datos electrónica, incluyendo el derecho de reproducir y almacenar la obra en servidores, a los efectos de garantizar su seguridad, conservación y preservar el formato.
c) Comunicarla, por defecto, a través de un archivo institucional abierto, accesible de modo libre y gratuito a través de internet.
d) Cualquier otra forma de acceso (restringido, embargado, cerrado) deberá solicitarse expresamente y obedecer a causas justificadas.
e) Asignar por defecto a estos trabajos una licencia Creative Commons.
f) Asignar por defecto a estos trabajos un HANDLE (URL *persistente)*.

4º. Derechos del autor.

El autor, en tanto que titular de una obra tiene derecho a:

a) Que la Universidad identifique claramente su nombre como autor de la misma
b) Comunicar y dar publicidad a la obra en la versión que ceda y en otras posteriores a través de cualquier medio.
c) Solicitar la retirada de la obra del repositorio por causa justificada.
d) Recibir notificación fehaciente de cualquier reclamación que puedan formular terceras personas en relación con la obra y, en particular, de reclamaciones relativas a los derechos de propiedad intelectual sobre ella.

5º. Deberes del autor.

El autor se compromete a:

a) Garantizar que el compromiso que adquiere mediante el presente escrito no infringe ningún derecho de terceros, ya sean de propiedad industrial, intelectual o cualquier otro.
b) Garantizar que el contenido de las obras no atenta contra los derechos al honor, a la intimidad y a la imagen de terceros.
c) Asumir toda reclamación o responsabilidad, incluyendo las indemnizaciones por daños, que pudieran ejercitarse contra la Universidad por terceros que vieran infringidos sus derechos e

intereses a causa de la cesión.

d) Asumir la responsabilidad en el caso de que las instituciones fueran condenadas por infracción de derechos derivada de las obras objeto de la cesión.

6º. Fines y funcionamiento del Repositorio Institucional.

La obra se pondrá a disposición de los usuarios para que hagan de ella un uso justo y respetuoso con los derechos del autor, según lo permitido por la legislación aplicable, y con fines de estudio, investigación, o cualquier otro fin lícito. Con dicha finalidad, la Universidad asume los siguientes deberes y se reserva las siguientes facultades:

- La Universidad informará a los usuarios del archivo sobre los usos permitidos, y no garantiza ni asume responsabilidad alguna por otras formas en que los usuarios hagan un uso posterior de las obras no conforme con la legislación vigente. El uso posterior, más allá de la copia privada, requerirá que se cite la fuente y se reconozca la autoría, que no se obtenga beneficio comercial, y que no se realicen obras derivadas.
- La Universidad no revisará el contenido de las obras, que en todo caso permanecerá bajo la responsabilidad exclusive del autor y no estará obligada a ejercitar acciones legales en nombre del autor en el supuesto de infracciones a derechos de propiedad intelectual derivados del depósito y archivo de las obras. El autor renuncia a cualquier reclamación frente a la Universidad por las formas no ajustadas a la legislación vigente en que los usuarios hagan uso de las obras.
- La Universidad adoptará las medidas necesarias para la preservación de la obra en un futuro.
- La Universidad se reserva la facultad de retirar la obra, previa notificación al autor, en supuestos suficientemente justificados, o en caso de reclamaciones de terceros.

Madrid, a ...5..... de ...Junio..... de ...2019

ACEPTA

Fdo.....

Motivos para solicitar el acceso restringido, cerrado o embargado del trabajo en el Repositorio Institucional:

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

NAVEGACION EXTERNALIZADA DE ROBOTS MÓVILES EN UN ALMACÉN

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el curso académico 2018/2019. es de mi autoría, original e inédito y no ha sido presentado con anterioridad a otros efectos. El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

Fdo.: Nerea Zabala Orive Fecha: 05/06/19

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: Álvaro Sánchez Miralles Fecha: 10/06/19

COMILLAS
UNIVERSIDAD PONTIFICIA
ICAI

GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

TRABAJO FIN DE GRADO

NAVEGACIÓN EXTERNALIZADA DE ROBOTS MÓVILES EN UN ALMACÉN

Autor: Nerea Zabala Orive
Director: Álvaro Sánchez Miralles

Madrid
Junio de 2019

NAVEGACIÓN EXTERNALIZADA DE ROBOTS MÓVILES EN UN ALMACÉN

Autor: Zabala Orive, Nerea.
Directores: Sánchez Miralles, Álvaro.

Entidad Colaboradora: ICAI - Universidad Pontificia Comillas.

RESUMEN DEL PROYECTO

El objetivo de este proyecto es diseñar un sistema capaz de migrar los procesos de control y navegación de robots móviles fuera de los mismos, y así poderles proporcionar los recursos computacionales existentes en plataformas exteriores, como la nube o el borde de la red. Los recursos de estas dos plataformas pueden contribuir a la mejora de los sistemas de robots ya que proporcionan una computación más flexible y más amplia.

El uso de Vehículos Guiados Autónomamente (AGV´s) en los almacenes, nace a raíz de la creciente tendencia a la automatización, al poder desplazarse por la nave de manera autónoma son utilizados principalmente para el transporte de cargas, y con ellos, se han sustituido operarios que son menos eficientes y mas costosos. Gracias a los AGV´s, empresas como ASTI, Amazon Robotics o MIR, han conseguido crear sistemas de gestión de almacenes automáticos; pero el creciente boom de las comunicaciones, hace que estas empresas se quieran aprovechar de los recursos de computación que existen fuera de sus naves; para crear sistemas multi-robot, que puedan gozar de mayor capacidad de computación e información que les ayude a optimizar sus procesos. Además, el procesamiento adicional que viene a raíz de esta migración, trae consigo la capacidad de incrementar la complejidad de los algoritmos de big data o machine learning que usen los sistemas de control y/o navegación.

En concreto, durante este proyecto se ha utilizado un robot Lego EV3 para simular el transporte de cargas dentro de un almacén. Para la navegación del robot EV3, se plantea una solución basada únicamente en los datos que se reciben de sus dos encoders, que permitirán al sistema conocer, en todo momento, la localización del robot dentro del mapa preestablecido que se va a usar. La idea es proponer una solución basada en la calibración exhaustiva de los robots y el uso de un sistema centralizado, para que el robot navegue por el mapa recibiendo órdenes del sistema central, que llevará la gestión de los pedidos y el control del robot. El sistema central, calculará la ruta más corta para que el robot se mueva hasta su destino y mediante ordenes se guiará al EV3 al mismo. Para poder llevar a cabo este proyecto; se usa una API (Application Programming Interface) para la comunicación entre ambas partes, esta API proporciona un sistema de comunicación entre servidor y cliente, siendo así la herramienta que permite al ordenador gestionar los motores y el encoder del robot.

En la Figura 1 se puede ver un esquema de la distribución de las diferentes partes que intervienen en el proyecto.

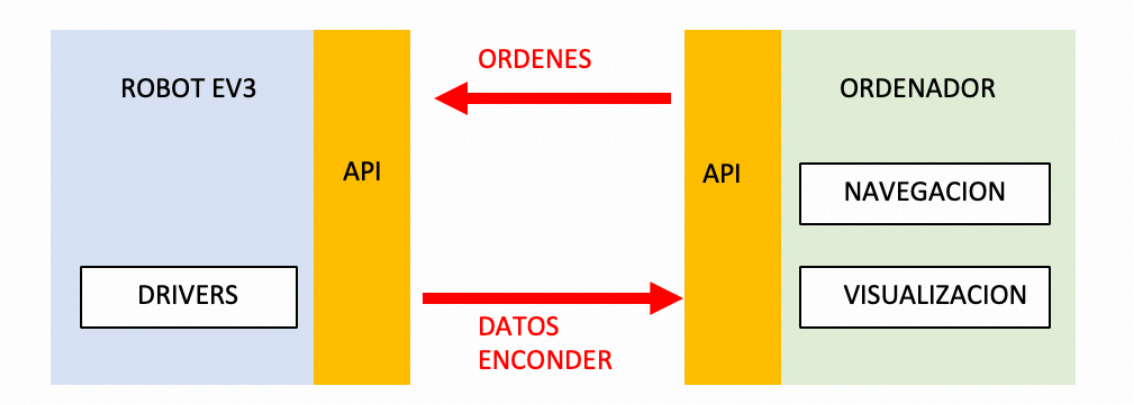


Figura 1: Esquema básico de las diferentes partes del sistema creado

En la Figura 1 se pueden ver las dos partes diferenciadas, el ordenador y el robot; que se comunican mediante sus API´s correspondientes. El ordenador le manda órdenes al robot y el robot a su vez manda información sobre el estado de sus encoders al ordenador. Ambas partes del sistema, tienen aplicaciones de software que añaden funcionalidades; el robot tiene una capa que gestiona su hardware, y contiene las funciones necesarias para poder utilizarlo, que es a lo que se ha llamado drivers; por otra parte el ordenador tiene dos capas, una que alimentada con la información del encoder, calcula las rutas del robot y las órdenes que se le van a mandar, que es la capa de navegación, y otra que con esa misma información gestiona el programa de visualización creado.

Tras la finalización de este proyecto, se tiene una plataforma con la que controlar el EV3 desde cualquier recurso computacional externo, y así el robot podrá ejecutar órdenes provenientes de cualquier sistema de navegación que se pueda comunicar con él mediante su API. En concreto, en este proyecto se han podido utilizar los recursos del ordenador para calcular las rutas del robot, gestionar sus movimientos y complementarlo con un visualizador del mapa. Como resultado, se pudo apreciar que la navegación del robot era satisfactoria, llegaba a sus puntos de destino siendo bastante preciso y las comunicaciones entre el robot y el ordenador son lo suficientemente rápidas para que la navegación sea exitosa.

EXTERNALISED NAVIGATION OF MOBILE ROBOTS IN A WAREHOUSE

Author: Zabala Orive, Nerea.
Directors: Sánchez Miralles, Álvaro.

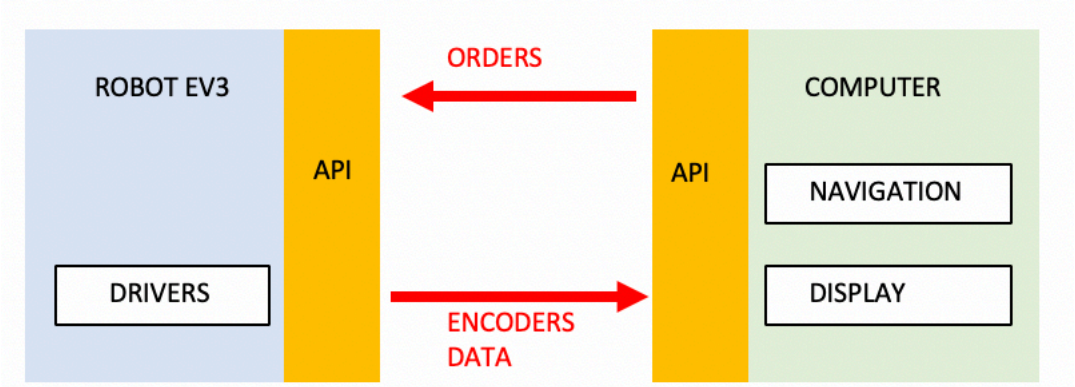
Collaborating Entity: ICAI - Universidad Pontificia Comillas

The objective of this project is the development of a system capable of migrating the navigation and control software of a mobile robot outside its physical boundaries; therefore, giving them the capacity to instantiate the navigation system across de whole computing strata. The computational resources available in these two platforms can enhance the performance of robot systems by providing them a more flexible and cost-efficient computation.

The use of Automated guided vehicles (AGV´s) in warehouses came along with the increasing automation trend, due to their capabilities of self-navigating the environment AGV´s are normally used to move payload through the environment, substituting less efficient and more expensive workers. Due to AGV´s along with other technologies, companies such as ASTI, Amazon Robotics or MIR have been able to create successful multi robot systems that help in the functioning of warehouses; but the development of new and promising technologies in the communications sector, has opened the possibility for this companies to provide their AGV´s with the resources the edge and cloud have to offer. These platforms make it possible for AGV´s to reduce their inside computational resources by using the resources provided by them and gives the system a wider range of information for it to use. The additional computation provided can be very beneficial por big data and machine learning algorithms used in robots.

During this project a Lego EV3 robot was used, to navigate a preestablished map simulating the movement of load in a warehouse. The navigation of the EV3 is based on the information provided by the robot's encoder, that will make it possible for the system to know where the robot is at all times. The idea is to create a navigation system based on a thorough calibration of the robot´s movements and a centralized architecture where the robot navigates due to the orders provided by the computer; and the computer is able to produce this order due to the information of the robot's encoders. The system will try to calculate the shortest path to guide the robot and it will manage the fulfillment of orders entering the system. An API (Application Programming Interface) is used to communicate both parts of the system, the robot and the computer; as information has to flow both ways, two API´s have to be created one on the robot´s side and another in the computers side.

In the Figure below, we can see the two different entities in this proyect, the robot EV3 and the computer, the software inside them and how the communicate.



The robot and the computer communicate by their API´s, the computer sends orders to the robot and the robot sends the encoders data to the computer. Both parts have software programs that provides them a specific functionality; the robot has a layer that provides the basic functions to manage its hardware, which are called drivers; and the computer has a navigation program which calculates the robots paths and manages the orders that enter the system, and a diplay program that shows the user a display of the warehouse map and the position of the robot inside it at all times.

Finally, after the completion of this proyect we will have a plataform that makes it posible to control de EV3 robot from any external computational resource that is able to communicate with the system created, by its API. We have observed that the navigation of the robot is succesfull, it arrives at it´s objective point navigating in a quite precise way and the comunications between the robot and the computer are fast enough for the fullfilment of th proyect.

Tabla de figuras

Tabla de tablas

Tabla de Ecuaciones

Tabla de Graficas

1. Introducción

1.1 Motivación

Las nuevas tecnologías, son una oportunidad para que las pequeñas y medianas empresas (PYME) puedan automatizar sus procesos, hasta ahora automatizar una fábrica o un almacén solo estaba al alcance de muy pocos, ya que suponía unos enormes costes de entrada y mantenimiento. Con la llegada de tecnologías como el big data, IoT, el procesamiento en la nube o la robótica colaborativa las PYMES pueden automatizar sus procesos de una manera más flexible y adaptándose rápidamente a medida que crecen [1].

El sector de la logística y los almacenes es cada vez más importante en nuestra industria, el comercio por internet ha creado consumidores, que quieren poder comprar cualquier cosa por la web y tenerla en casa en el mínimo tiempo posible. Para poder estar al día con las exigencias de los consumidores, las empresas de logística están intentando implementar nuevas tecnologías, que les permitan ser más competitivos y por eso mismo, este proyecto se ha realizado dirigido hacia ellos.

Hasta hace relativamente poco, la automatización de un almacén estaba solo al alcance de grandes naves, donde se instalaban raíles fijos, que movían los paquetes de un sitio a otro; pero ahora las cosas han cambiado, con la llegada de los AGV´s a la industria y nuevos softwares de gestión, se es capaz de incrementar la eficiencia de un almacén sin tener que soportar unos costes desmesurados [2]. Con la llegada de nuevas tecnologías como el 5G, que va a soportar el cloud y edge computing y va a proporcionar comunicaciones de ultra baja latencia [3], las empresas que utilizan AGV´s quieren poder darles a sus robots todos los servicios que la nube y el borde de la red, les puede ofrecer [1]. Para ello, se tienen que crear plataformas que externalicen el control y la navegación de los robots. Por eso mismo, la motivación de este proyecto reside en realizar una plataforma que comunique satisfactoriamente un robot con un sistema de procesamiento externo, para luego poder construir sobre ello y darle las funcionalidades necesarias.

Estas plataformas van a permitir la separación entre hardware y software en la robótica, el API del robot únicamente tendrá la misión de abstraer el hardware, al software que lo controla. Así, se podrá integrar de manera más simple distintas tecnologías en un mismo robot, cada sistema o funcionalidad podrá comunicarse con la API del robot para acceder a sus drivers o al control a bajo nivel del hardware, pudiendo así darle valor añadido al robot mediante funcionalidades implementadas en software fuera de él. Una de las empresas que tratan de expandir la modularidad de los robots es ROS, este sistema operativo de código abierto trata de generar una plataforma universal con la que poder desarrollar funcionalidades para cualquier tipo de robot [4].

1.2 Objetivos

El objetivo de este proyecto es crear un software que mueva el robot Lego EV3 dentro de un mapa preestablecido, dándole órdenes sobre que movimientos realizar en cada momento desde un ordenador central, en el fondo se quiere simular el movimiento de estanterías en un almacén automatizado con robots móviles. El mapa constará de cuatro puntos de estanterías y un punto de recogida de productos, en concreto, el mapa utilizado será el que se puede ver en la Figura 2 las estanterías son los puntos verdes A1, A2, A3 y A4 y el punto de recogida el E1 azul. El mapa tiene unas dimensiones de 1 x 1 metros, que para el tamaño del EV3 son unas distancias considerables.

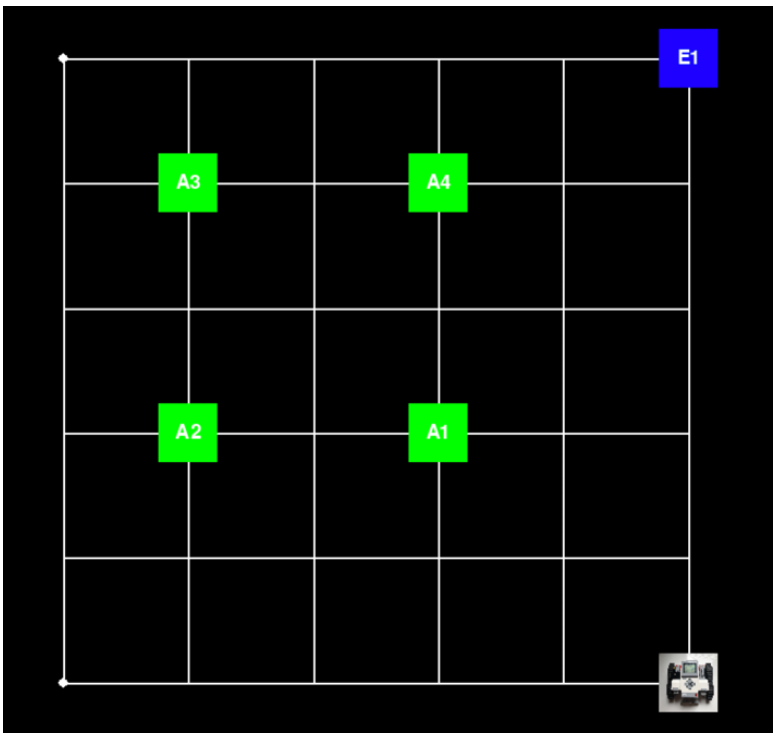


Figura 2: Mapa del almacén utilizado en el proyecto

Se dará por finalizado un pedido cuando el robot haya completados tres pasos, el primero llegar a la estantería, el segundo moverse al punto de recogida (E1) y el tercero y último volver al punto de la estantería inicial. Con esto se simula como si el robot hubiese levantado la estantería, la hubiese llevado a un punto de recogida donde alguien cogería el producto necesario y el robot la volviese a dejar donde estaba. Para la navegación del robot, se dividirá el mapa en una matriz, es decir se partirá el espacio en cuadrados como se puede ver en la Figura 2. El robot navegará por el mapa girando 90º tanto a la izquierda como a la derecha y avanzando recto de casilla en casilla.

En la Figura 3 se puede ver un ejemplo; si al sistema le entrase un pedido en la estantería A1, el robot podría realizar el siguiente recorrido para dar por finalizada la tarea. Primero pasaría por el punto A1, luego iría a E1 y por último volvería a A1. La ruta mostrada es una de las muchas que podrían utilizarse.

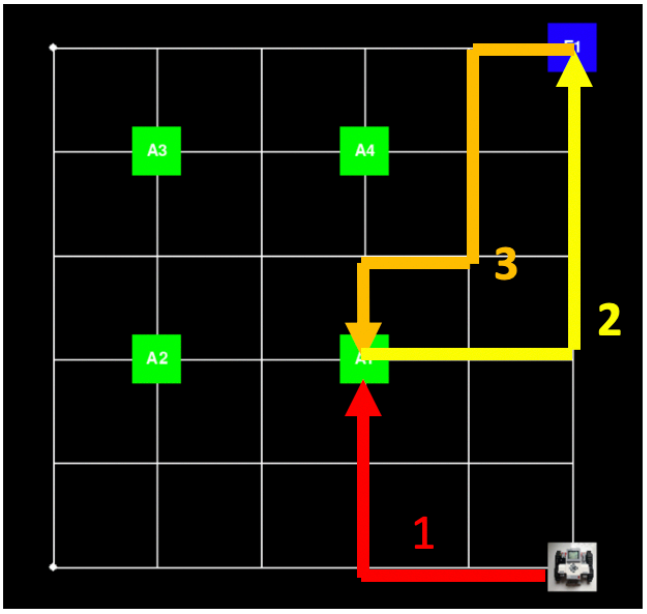


Figura 3: Mapa del almacén con una posible ruta si entra un pedido en la estantería A1

En este proyecto se quiere controlar en todo momento los movimientos del robot desde un ordenador central, es decir, se migrará todo el software que controle la navegación del robot a un ordenador cercano y desde ese ordenador se mandarán órdenes para que el robot se mueva. Concretamente, las órdenes que se le darán al robot serán muévete recto unos determinados metros y gira a la derecha o a la izquierda. El hecho de tener el software "pesado" fuera del robot, permite tener toda la potencia de un ordenador que es bastante mayor a la del EV3, para calcular rutas y además permite modificar el software de una manera rápida y dinámica.

Al llevarse, el software fuera del robot una parte muy importante de este proyecto son las comunicaciones entre robot y ordenador para ello se usará una Restful-API´s que es una forma ligera de compartir información entre cliente y servidor usando protocolo HTTP y arquitectura REST(Representational State Transfer). Al querer comunicación bidireccional, es decir, que el robot pueda mandar información y el ordenador pueda mandarle ordenes se tendrán dos API`s una en el ordenador y otra en el robot.

Para terminar, también se realizará un programa de visualización del almacén, es decir cuando el software empiece a funcionar se creará una pantalla en la que se visualizará el mapa del almacén, en el se podrá ver donde se encuentra el robot dentro del mapa e información actualizada sobre las estanterías y el robot.

2. Estado del arte

El e-commerce es un sector en pleno auge, la facturación del comercio por internet en España se ha multiplicado por ocho desde el año 2008 [5]. Durante estos últimos años se han ido creando multitud de tecnologías para automatizar almacenes, que responden a la necesidad de la industria de ser más rápidos en entregar y gestionar los pedidos. Automatizar un almacén reduce los costes operacionales, mejora la eficiencia del sistema y la calidad del servicio [6]. En esta sección se va a evaluar las tecnologías que ya existen para la automatización de almacenes, el uso de AGV´s en los mismos y las distintas tecnologías que existen para el desarrollo de software.

2.1 Automatización de almacenes

En esta sección, se van a analizar los dos pilares fundamentales de la automatización de un almacén, por un parte se analizará los softwares que se utilizan y por otra parte las tecnologías que existen para el movimiento de cargas dentro de los almacenes.

2.1.1 Software de gestión de almacenes

Un software de gestión de almacenes, habitualmente abreviado como SGA, permite a tiempo real gestionar la operativa de todos los procesos del almacén, desde la recolección de paquetes hasta su empaquetación y carga en el camión. Los objetivos de estos softwares son reducir los costes de gestión, acelerar las tareas y optimizar los recursos de tiempo y espacio [7]. Para que un software se pueda denominar SGA tienen que estar cubiertas las siguientes áreas [8]:

- **Entrada:** Esta sección engloba; el control de la entrada de los paquetes, la recolección de los datos logísticos y el etiquetado. Toda esta sección es la que se encarga de contabilizar que entra en el almacén y del etiquetado para poder localizar las diferentes cargas en los procesos siguientes.

- **Ubicación:** Esta sección se encarga de gestionar la ubicación de cada paquete dentro del almacén. Para ello, se utilizan distintas estrategias para decidir cual es la localización idónea del paquete, esta suele ser la que minimiza los movimientos que se tienen que realizar dentro del almacén.

- **Control de stock:** Esta sección se encarga de recabar información sobre el stock, suele también llevar a acabo un análisis sobre que productos tienen una rotación mayor para poderse anticipar a la demanda.

- **Salida:** Esta sección controla las salidas de paquetes del almacén, desde la recogida del paquete en su punto de almacenaje a la preparación de la carga para ser transportada a su destino.

- **Maquinaria**: Esta sección es especifica para el control de la maquinaria que haya dentro del almacén, no todas las SGA´s tienen esta sección, pero es muy útil si se dispone de un almacén automatizado.

2.1.2 Movimientos de cargas en un almacén

La manera mas arcaica de gestionar el movimiento de los paquetes en un almacén, es con el uso de carretillas elevadoras como la de la Figura 4 o con transpaletas manuales como la de la Figura 5. Con este tipo de sistemas, el operario está al mando en todo momento de la maquinaria utilizada y al ser un trabajo repetitivo y bastante predispuesto a errores, es el puesto de trabajo ideal para ser remplazado por maquinas autonomas.

Figura 4: Carretilla elevadora

Figura 5: Transpaleta manual

Si se quiere gestionar el movimiento de los productos reduciendo el capital humano necesario, hay principalmente dos opciones; la primera es hacer uso de railes o cintas transportadoras y la segunda es utilizar bots (“pequeños vehiculos electricos”) que mueven productos de un lado a otro. Ambas propuestas son parte de las llamadas good to person automation (GTP), que consiste en acercar el producto al trabajador en vez de que el trabajador se acerque al producto [9].

2.1.2.1 Cintas transportadoras

Esta tecnología usa raíles, rodillos, cintas transportadoras o guías aéreas para trasladar los productos de un punto del almacén a otro, es un movimiento estatico ya que la ruta que se sigue es siempre la misma. El uso de cintas transportadores es mas conveniente para almacenes de gran tamaño y el incoveniente de estas tecnologias es el gran coste inicial que conllevan, debido a que es necesario acondicionar el almacén entero para su uso. En la Figura 6 se puede ver un ejemplo de transporte por guías aéreas en un almacen.

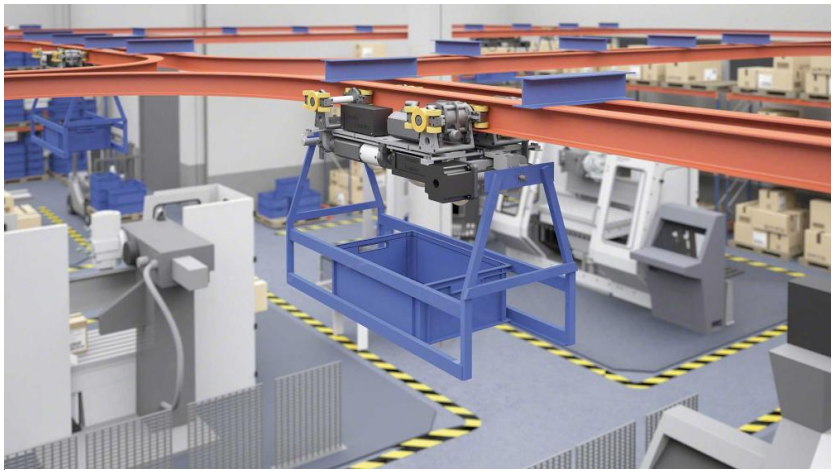


Figura 6: Transporte por guías aéreas en un almacén

Un ejemplo un poco menos común dentro de estas tecnologías es el que oferece la empresa Autostore System. Como se puede ver en la Figura 7 este sistema usa una matriz tridimensional de soporte que se instala en el local y dentro de esta matriz están las cajas con los productos almacenados, los robots navegarán por los railes situados en la capa mas elevada. Cuando entra un pedido al sistema, uno de los robots irá a la vertical correspondiente y elevará la caja con el producto deseado, llevandolo por último a la estacion de entrega de pedidos [10].

Figura 7: Red tridimensional de raíles de la empresa Autostore Systems

2.1.2.2 Robots móviles (*Bots*)

La otra alternativa es el uso de robots móviles, también denominados bots, el uso de estos robots presenta una solución mas flexible y menos costosa que las cintas transportadoras. Esta tecnología consiste en utilizar numerosos robots móviles que naveguen por el suelo del almacén, elevando las estanterías y llevándolas al punto de recogida de los productos. Los *bots* reducen la necesidad de acondicionar el espacio utilizado y dan la posibilidad de expandir el almacén sin tener que realizar grandes cambios. Un ejemplo del uso de *bots* para mover mercancías en un almacén es el Betty Bot, de la empresa *Amazon Robotics*.

Figura 8: Betty Bot de la empresa Amazon Robotics

Amazon compró la empresa *Kiva Systems* en marzo del 2012 por 775 millones de dólares, y desde entonces utiliza esta tecnología en sus almacenes [11]. En la Figura 8 se puede ver un Betty Bot. Estos bots están dotados de un mecanismo que levanta las estanterías a unos cuantos centímetros del suelo para poder moverlas por el almacén, también tienen sensores infrarrojos y sensores de presión para detectar colisiones, dos motores eléctricos sin escobillas para girar las ruedas y dos cámaras; una apuntando al suelo y otra a la estantería que se disponen a elevar.

2.2 AGV´s (Vehículos autónomamente guiados)

Los robots móviles están dotados principalmente de:

- **Actuadores**: Componente responsable de mover o controlar un mecanismo o sistema.
- **Sensores**: proporcionan información sobre su entorno o sobre su propio sistema.

Los AGV´s son robots móviles capaces de guiarse por el espacio ellos mismos, esto es posible mayoritariamente gracias a la información de sus sensores que difieren dependiendo de la tecnología que utilice el robot para guiarse por el entorno [12]. Se pueden distinguir distintos tipos de AGV`s en función de su tecnología de guiado y los mas comunes son los siguientes [13]:

Filoguiado: es el uso de líneas que guían a los robots por la superficie de la nave, estas líneas pueden ser o bien de colores o magnéticas. Los robots están dotados de sensores que detectan las líneas y consiguen seguirlas, pero la desventaja mas grande del uso de robots filoguiados es la poca flexibilidad a la hora de modificar la ruta de los robots.

Figura 9: Vehículo filoguiado

Laserguiado: En este caso, los bots emiten rayos laser que se reflejan en las superficies de su alrededor y reciben los rayos reflejados por sus sensores ópticos. Para poder utilizar estas tecnologías se tienen que instalar espejos catadióptricos, que son unos elementos que reflejan luz, en puntos clave del almacén para que el robot se oriente. Esta tecnología, permite cambiar y modificar la distribución del almacén de manera sencilla.

Mapeado: Esta solución utiliza cámaras y sensores LiDar`s, para realizar un modelo en 3D o 2D del entorno del robot y por tanto no necesita de ningún elemento externo para guiarse por el espacio. Estos vehículos tienen por detrás una tecnología muy puntera de reconocimiento de imagen e inteligencia artificial [14]**.**

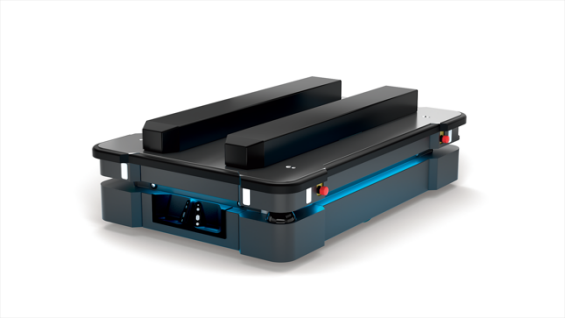


Figura 10: Vehículo de la empresa MiR que utiliza el mapeado

2.3 Robots conectados

En esta sección se van a presentar las distintas tecnologías que se usan para dar conectividad, dentro de una red a los robot y se van a mencionar que variedades de robots conectados existen.

2.3.1 Tecnologías de comunicación

Las tecnologías de comunicación son las que permiten darles conectividad a los robots, estas tecnologías se dividen en dos grandes grupos; las comunicaciones por el medio y las comunicaciones entre procesos. Las comunicaciones por el medio son el medio físico por el que la información viaja, mientras que las comunicaciones entre procesos son las distintas aplicaciones que permiten compartir información entre procesos distintos.

2.3.1.1 Comunicaciones por el medio

Para darle conectividad a un robot, lo primero que se tiene que diseñar es que tipo de comunicaciones se van a llevar a cabo por el medio. Hay distintas tecnologías tanto cableadas como inalámbricas que se pueden usar, y se van a exponer a continuación.

- **Cables:** El uso del cable es el método mas convencional, proporcionan el medio físico por el que se va a mandar la información. Si se quiere conectar el robot directamente a internet se usará un cable ethernet, en cambio si se quiere conectar a un servidor o a un ordenador con un cable USB de cualquier tipo valdría.

- **Bluetooth:** es una tecnología inalámbrica que permite transmitir información mediante ondas radio, esta enfocada a una transmisión de bajo alcance y bajo coste. Hay diferentes tipos de bluetooth dependiendo de la potencia emitida y el alcance de la señal. Hay 4 tipos distintos de tecnologías bluetooth y son las siguientes [15]:
  - Clase 1: Alcance aproximado de 100 metros y potencia máxima de 100mW
  - Clase 2: Alcance aproximado de 10 metros y potencia máxima de 2.5mW
  - Clase 3: Alcance aproximado de 1 metro y potencia máxima de 1mW
  - Clase 4: Alcance aproximado de 0.5 metros y potencia máxima de 0.5mW

- **Wifi:** es una tecnología inalámbrica que utiliza señales de radio para transmitir información por el aire, esta tecnología permite conectarse entre dispositivos o a internet, pero es imprescindible tener un router para ello.
  Existen dos tipos distintos de wifi dependiendo de los estándares IEEE 802.11 en los que estén basados [16]:
  - IEEE 802.11b, IEEE 802.11g e IEEE 802.11n estos tres estándares permiten transferir información en la banda de 2.4 GHz a velocidades de 11Mbps, 54Mbps y 600Mbps respectivamente.
  - IEEE 802.11ac también conocido como WIFI 5 es la versión mas moderna y transfiere información en la banda de 5GHz y puede alcanzar una velocidad de 1733Mbps.

2.3.1.2 Comunicaciones entre procesos

En este apartado se van a explicar los métodos que hay para la comunicación entre procesos a nivel red, es decir con que tecnologías se pueden comunicar dos programas que están siendo usados dentro de una misma red de comunicaciones. Dentro del uso de el protocolo HTTP hay diferentes estándares para el diseño y desarrollo de cualquier servicio web [17].

- **REST (Representational State Transfer):** Hace tiempo que se popularizó por ser una arquitectura flexible que permite desarrollar API´s ligeras [18]. Es una arquitectura sin estados, que usa protocolo HTTP para transmitir información en cualquier formato de datos.

- **SOAP (Simple Object Access Protocol):** A diferencia de REST, SOAP es un protocolo y no una arquitectura y solo permite transmitir datos de tipo XML, pero en cambio, se puede usar con cualquier capa de aplicación de red como HTTP, TCP o UDP. SOAP es menos flexible que REST y esta mas enfocado a aplicaciones con complejidades mas altas.

2.3.2 Procesamiento de la información

Dentro de los robots, hay una tarea imprescindible que es el procesamiento de la información procedente de sensores, actuadores y cualquier otra tercera parte que quiera comunicarse con él. Esta información puede procesarse tanto dentro del mismo robot en su propio hardware, como en la nube o el borde de la red. Si se quiere procesar la información fuera del robot, es decir, en la nube o en el borde se necesitan plataformas potentes para el envío y recepción de datos. A continuación, se va a explicar lo que es el procesamiento en la nube y en el borde de la red.

- **Computación en la nube:** Es un servicio que permite el acceso a recursos de almacenamiento y procesamiento, de centros de datos, en cualquier parte del mundo, a través de internet. Es un servicio que destaca por su flexibilidad ya que es posible aumentar o reducir estos recursos sin grandes problemas [19].

- **Computación en el borde de la red:** El borde de la red consta de los servicios de computación que hay cerca de donde se esta generando la información, y por tanto, la computación en el borde de la red permite acercar el procesamiento de la información mas cerca de donde se esta creando [20]. Los edge services o servicios del borde de la red son algo relativamente nuevo y se prevé que para 2020 sea una industria multimillonaria. [21]

2.3.3 Arquitecturas de sistemas de robots conectados

Dependiendo de su arquitectura, podemos distinguir dos tipos de sistemas multi- robot, los centralizados y los descentralizados. Los sistemas centralizados tienen una agente

central que toma todas las decisiones, mientras que los descentralizados lo carecen y las decisiones son tomadas por cada robot individualmente [22].

2.3.3.1 Sistema Descentralizado

En un sistema descentralizado, el robot analizará los datos proporcionados por sus sensores y actuará en base a esa información. Un factor determinante en un robot que vaya a ser utilizado en un sistema como estos es que la capacidad de procesamiento tiene que ser lo bastante alta para tomar decisiones en base a la información de sus sensores y esto puede encarecer los costes del robot. En estos sistemas los robots pueden tener comunicaciones con un recurso exterior, pero este no le dará ordenes concretas sino mas bien objetivos. En la Figura 11 se puede ver un ejemplo de como serian las partes de un sistema de control de robots descentralizado.

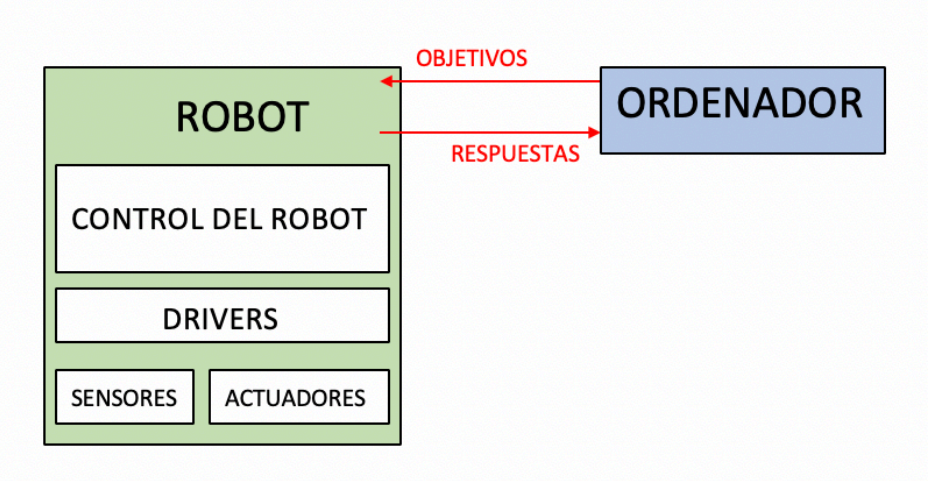


Figura 11: Esquema de un sistema descentralizado

2.3.3.2 Sistema Centralizado

En cambio, en un sistema centralizado, el robot mandará toda la información recabada por los sensores al agente central y este tomará las decisiones por él. Pudiendo tener robots mas baratos de fabricar por el bajo procesamiento que necesitan. Otra ventaja de utilizar un sistema como estos es que al tener toda la lógica de programación fuera del robot, es mas fácil corregir errores o actualizar el software con implementaciones nuevas. Por ejemplo, si se tienen 100 robots trabajando en un almacén y llega una nueva versión del software que los gestiona, solo se tendría que actualizar el ordenador central y no ir uno por uno actualizando todos. En la Figura 12 se puede ver un ejemplo de las partes de un sistema centralizado.

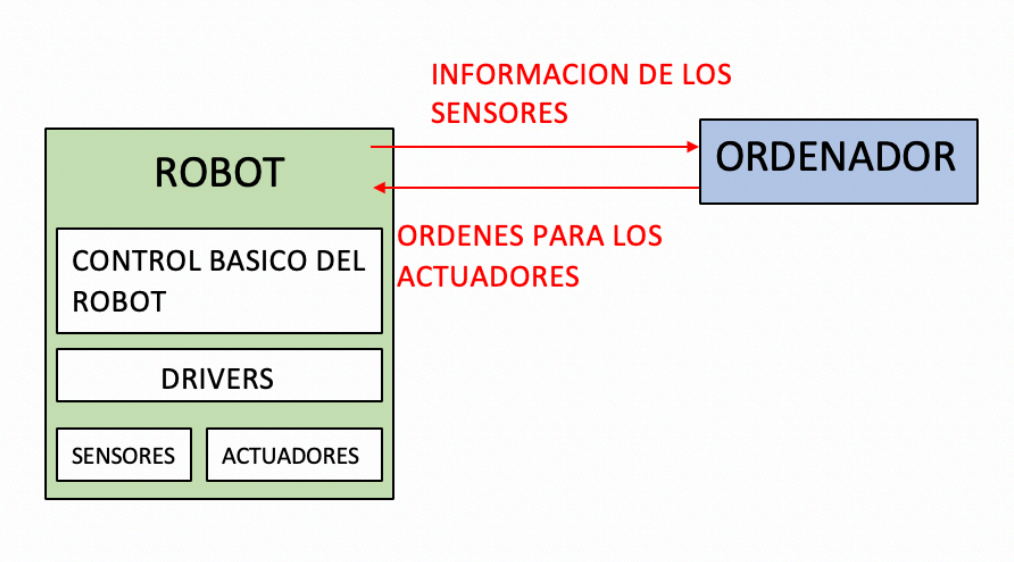


Figura 12: Esquema de un sistema centralizado

2.3.4 Tipos de robots conectados

Al darle conectividad a un robot, se puede tanto conectar con otros robots, servidores, con la nube o incluso el borde de la red. En esta sección se van a explicar que tecnologías son posibles gracias a la conectividad de los robots.

2.3.4.1 Network Robots

Un "Network robot" es un robot conectado a una red de comunicaciones como internet o la LAN, esta red puede ser tanto inalámbrica como por cable y los robots intercambian información de sus sensores, actuadores o controles dentro de ella para colaborar entre ellos o con un tercero [23]. Estos sistemas, permiten a los robots colaborar y compartir información dándoles la capacidad de ser mas eficientes y lograr objetivos mas complejos que si actuasen solos, por ejemplo, dos robots mapeando una habitación pueden compartir información sobre sus sensores y así la tarea será completada en menos tiempo.

2.3.4.2 Robótica en la nube

La robótica en la nube da un paso mas con respecto a *Network robotics*, esta tecnología, permite que los robots utilicen los recursos de la infraestructura cloud para mejorar sus servicios. La robótica en la nube es una rama de la robótica que va de la mano y depende inequívocamente del cloud computing, almacenamiento cloud y redes cloud, los sistemas de multi-robots se benefician de esta plataforma pudiendo acceder a recursos computacionales y servicios compartidos. La gran ventaja del cloud es su flexibilidad, una empresa que utilice sus servicios de procesamiento, puede alquilar únicamente los recursos que esté utilizando en cada instante. Por ejemplo, en épocas donde una fabrica no esta a pleno funcionamiento, se necesitarán menos servicios que

cuando si que lo esta y por tanto pagarán menos por ellos [24]. El uso de esta tecnología es especialmente interesante para robots que utilicen algoritmos de machine learning o de Big data ya que estos procesos son computacionalmente muy costos y consumen mucha batería algo muy poco deseable en el caso de robots móviles.

2.3.4.3 Robótica en el borde de la red

El uso del borde de la red en robótica se puede denominar Edge robotics y su gran ventaja con respecto a la nube, es que la información no tiene que viajar hasta los servidores de la nube; sino que se procesa en un servidor cercano al robot y esto permite tener unas comunicaciones mas rápidas. En el control de un robot existen tareas mas criticas que otras, las criticas, no podrían llevarse a la nube por el retardo que se produciría durante el envío de la información, en cambio, si esta información pasa a procesarse en el borde, la latencia de las comunicaciones se reduce y por tanto podría llegar a ser posible externalizar las tareas criticas [25]. Para el buen uso de colud y edge robotics es muy importante la calidad y rapidez de la conectividad, con la llegada del 5G, empresas como Ericsson, prometen que a diferencia del 4G este va soportar tecnologías para el edge y cloud computing, además de tener una red que ofrecerá procesos de ultra-baja latencia muy beneficiosos para el funcionamiento de los robots [26].

2.4 Algoritmos de rutas entre dos puntos

Un algoritmo de rutas es aquel que busca un camino entre dos puntos, es importante que estos algoritmos traten de buscar la ruta mas corta y los mas conocidos son los siguientes:

- **Dijkstra:** En este algoritmo, se usa la distancia mínima de todos los posibles nodos al nodo objetivo, se va creando una lista de nodos visitado y no visitados y se marcan como visitados aquellos nodos que tengan la menor distancia al objetivo en cada paso, cuando se han evaluado todos los nodos sabremos cual es la ruta mas corta.

- **A*:** Este algoritmo usa el coste de llegar al siguiente nodo y el coste en llegar desde ese siguiente nodo al objetivo para decidir cual será su siguiente posición [27].

2.5 Lego Mindstorms

Lego Mindstorms, es una gama de juguetes de robótica creada por la empresa Lego, muy utilizados en el ámbito educacional y de investigación [28]. Desde el año 1998 ha habido tres generaciones distintas de robots, con diferentes bricks, que es el hardware principal y la fuente de alimentación del robot, además de sensores y actuadores. En la Tabla 1 se pueden ver las tres generaciones de Lego Mindstorms con sus características principales.

Tabla 1: Comparación de modelos Lego Mindstorms RCX, NXT y EV3. [29]

RCX	NXT	EV3
		
• 32 KB RAM 16 KB ROM • 3 entradas y salidas. • Pantalla LCD • Comunicaciones mediante infrarrojo	• 64 KB RAM 256 KB Flash • 4 entradas y salidas. • Pantalla LCD • Comunicaciones mediante Bluetooth o USB	• 64 KB RAM 16 MB Flash • 4 entradas y salidas • Pantalla LCD • Comunicaciones mediante USB, mini-USB, infrarrojos, Bluetooth y Wifi con hardware adicional

2.5.1 Sistemas operativos

Un sistema operativo (S.O) es software que gestiona el hardware y que puede proveer servicios al usuario mediante aplicaciones. Para el control de robots existen entre otros los sistemas operativos que se mencionan a continuación.

Lego Mindstorms: Es el sistema operativo por defecto de todos los robots Lego y únicamente se puede usar para este tipo de robots. Permite el control de todos los sensores y actuadores que vienen en sus paquetes, es un sistema operativo muy robusto, pero no es de código abierto.

ROS: ROS son las siglas de “Robot Operating System” y es un sistema operativo de código abierto que se empezó a desarrollar en el 2007. Este sistema operativo se desarrolló para impulsar la robótica colaborativa y para que diferentes laboratorios, fabricas o incluso entornos educacionales se pudiesen beneficiar de las tecnologías desarrolladas por otras entidades [30].

Ev3dev: Es un sistema operativo basado en Linux para el Lego EV3, es de código abierto y permite programar este robot con lenguajes como C++, java o Python. Este sistema operativo permite gestionar todos los sensores y actuadores del Lego EV3 y al ser de código abierto permite personalizar cualquiera de sus funciones. El gran potencial de este sistema operativo es la flexibilidad que proporciona al usuario, ya que, al poderse utilizar con casi todos los lenguajes de programación, permite utilizar librerías muy potentes desarrolladas en otros lenguajes de programación.

2.5.2 Lenguajes de programación

En esta sección se van a comentar los lenguajes de programación mas utilizados en la actualidad, cada lenguaje tiene sus particularidades y por tanto es importante a la hora de realizar un proyecto escoger bien que lenguaje se va a utilizar.

Python: Python es un lenguaje interpretado y de alto nivel que permite la programación orientada a objetos. Su flexibilidad a la hora de escribir código y la sintaxis avanzada que tiene, permiten reducir el numero de líneas de código necesarias para implantar una funcionalidad.

C++: es un lenguaje compilado y orientado a objetos que destaca sobre los demás por su rapidez y por la gestión de los recursos que permite hacer. [31].

Java: Java es un lenguaje interpretado y orientado a objetos en el que los programas se ejecutan en un maquina virtual que permite tener un entorno mas controlado [32].

2.5.3 Repositorios de código

Los repositorios de código son plataformas online que permiten guardar código y hacer control de sus versiones, son muy útiles cuando varias personas desarrollan un mismo código a la vez, ya que permite generar distintas ramas que luego se unen de manera ordenada y sin conflictos.
Git es un software de versiones creado por Linus Torvalds y actualmente tiene dos plataformas de desarrollo:

- **GitHub**: Fue desarrollada en el 2010 y en 2018 fue comprada por la Microsoft, esta plataforma solamente permite tener repositorio privados si se paga por ellos [33].
- **GitLab:** Desarrollada en el año 2013 y la gran diferencia respecto de GitHub es que permite tener repositorios privados sin pagar por ellos.

2.5.4 IDE

Un IDE es un “Integrated Development Enviroment” e integra en un solo programa las herramientas de desarrollo, compilación y ejecución de codigo. [34] Los dos IDE´s mas utilizados son los siguientes:

- **Eclipse:** Un IDE diseñado principalmente para código en Java pero que se puede utilizar con otros lenguajes como Python o C ++.

- **Pycharm:** Es un IDE para el desarrollo de código en Python de la empresa JetBrains, es compatible con Windows, Linux y macOS. Pycharm es gratuito para estudiantes y tiene un terminal incluido, se puede usar con repositorios como Git y viene con un depurador de código [35].

3 Recursos utilizados

En esta sección se van a exponer que recursos se han utilizado en este proyecto tanto a nivel hardware como software y la razón de haberlos utilizado.

3.1 Hardware

El hardware utilizado en este proyecto tiene dos partes diferenciadas, el hardware del robot y el hardware del ordenador y a continuación se van a explicar sus características.

3.1.1 Hardware del robot

En la Figura 13 se puede ver un esquema del hardware del robot, las diferentes partes utilizadas aparecen numeradas y se expondrán a continuación todas sus características.

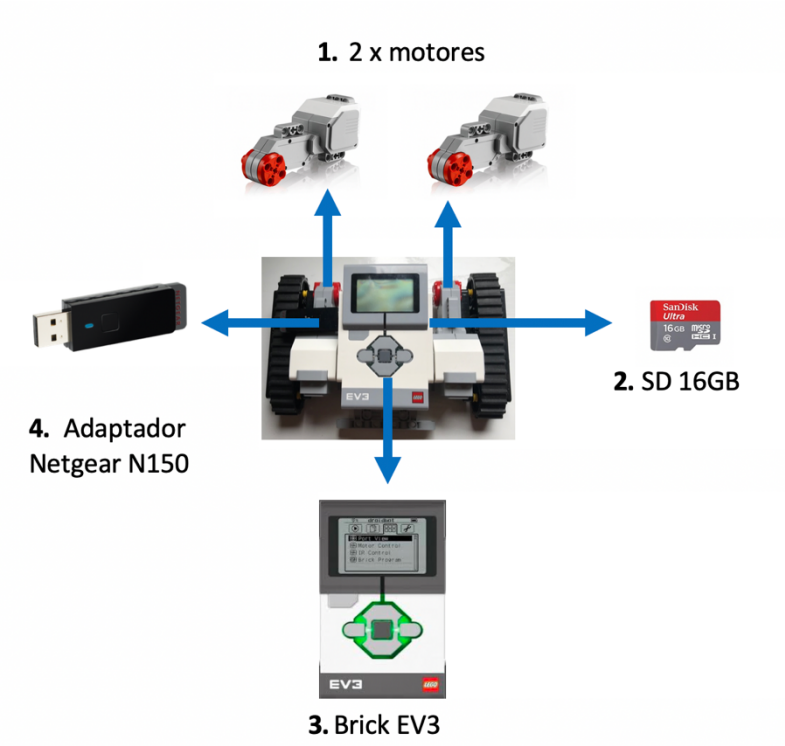


Figura 13: Esquema de las partes de hardware del robot

1. **Motores:** Estos motores funcionan entre 160-170 rpm con un torque de rotación de 20Ncm, también incluyen un sensor de rotación con una resolución de 1º que es el

que proporciona la información con la que se va a localizar al robot. Se utiliza un motor por rueda que para mover el robot por el suelo.

2. **Storage Disk de 16GB:** En ella se guarda el sistema operativo ev3dev utilizado en el robot, se utiliza para que el brick del Lego arranque desde ella y así poder modificarle el sistema operativo que se va a utilizar. En ella se guardan todos los archivos del robot.

3. **Brick Lego EV3:** El brick utilizado en este proyecto es el EV3 de *Lego Mindstorms* y es el que proporciona, tanto la fuente de energía para todos los periféricos conectados como el núcleo de procesamiento del robot. El brick tiene: una pantalla, 4 puerto de salida y otros 4 de entrada, un puerto USB, un puerto de tarjeta SD y 5 botones para poder interactuar con el sistema operativo. Las características técnicas de este brick se pueden ver en la Tabla 1.

4. **Adaptador Netgear N150**: El EV3 soporta wifi con un adaptador adicional Netgear N150, como se comentará en la sección 4.7.2 la tecnología que se usa para comunicar el robot EV3 con el ordenador es wifi y por tanto es necesario el uso de este adaptador para llevar a cabo las comunicaciones.

3.1.2 Hardware del ordenador

En esta sección se van a mencionar las características de hardware del ordenador utilizado en este proyecto, en la Figura 14 se puede ver el ordenador y a continuación que especificarán sus características.

Figura 14: MacBook pro 13

MacBook pro-13: Este ordenador tiene un procesador de doble núcleo de 2.3GHz Intel Core i5, 256 GB de almacenamiento, tarjeta grafica Intel Iris Plus Graphics 640, pantalla retina y dos puertos Thunderbolt 3. Además, permite conectarse a la red de comunicaciones wifi utilizada y tiene distintas plataformas para el desarrollo de código en cualquier lenguaje de programación. Se utiliza este ordenador porque es el que se tiene disponible y cumple todos los requisitos necesarios para poder llevar a cabo el proyecto.

3.2 Software

El software de este proyecto se puede dividir en dos, el del robot y el del ordenador en esta sección se va a exponer y explicar el software de ambas partes.

3.2.1 Software del Robot

En esta sección, se enuncia y explica el software utilizado en el robot Lego EV3, tanto el sistema operativo, el lenguaje de programación como las librerías mas importantes, y se hará una breve descripción del código implementado. En la Figura 15 se puede ver un esquema del software implantado en el robot con las plataformas y librerías utilizadas enumeradas sobre ella; a continuación, se explicarán mas en detalle estos elementos.

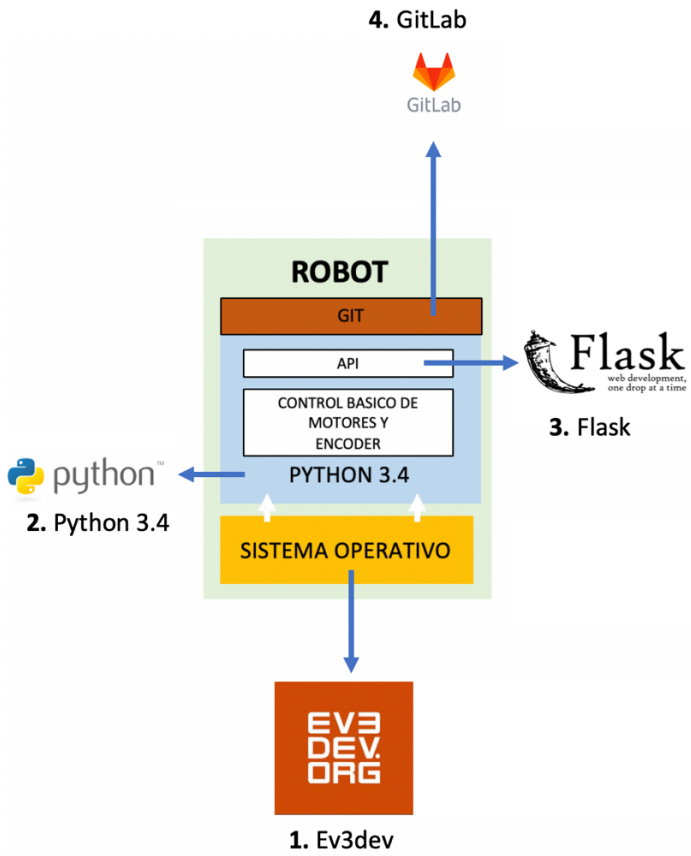


Figura 15: Esquema del software del robot EV3

1. **Ev3dev**: El sistema operativo (S.O) por defecto del Ev3 es Lego Mindstorms, en este proyecto se quiere utilizar Python como lenguaje de programación, ya que permitirá utilizar frameworks y librerias para poder, por ejemplo, generar la API del robot, por ello, se decide cambiar el S.O al ev3dev. Ev3dev, es un S.O de código abierto que contiene todas las funciones necesarias para utilizar el hardware del EV3.

2. **Python 3.4:** Python es el lenguaje de programación utilizado, por su facilidad a la hora de escribir código y por los frameworks y librerías que tiene, que resultan útiles a la hora de darle funcionalidades al programa. Además, Python 3 contiene un entorno virtual al que se le llama v*irtualenv* que permite generar entornos aislados, para trabajar con distintas versiones de librerías y paquetes. El *virtualenv* es útil, porque permite instalar paquetes y librerías únicamente para un proyecto y no para todo el sistema y por tanto se consigue por ejemplo simular el entorno del robot en el ordenador.

3. **Flask:** Flask, es un microframework para Python, un framework es una herramienta que proporciona una serie de utilidades para el desarrollo de software; un microframework es un framework minimalista, es decir, te proporciona los recursos mínimos necesarios. En concreto, Flask es un framework para el desarrollo web que esta basado en Jinja2 y Werkzeug, con el que se puede generar una API reduciendo notablemente la cantidad de líneas de código utilizadas. El uso de Flask es muy recomendable para prototipado rápido de proyectos debido a su minimalismo y simplicidad.

4. **GitLab:** Gitlab es una plataforma para guardar código online, que permite control de versiones y la creación de distintas ramas en el proyecto. El uso de un repositorio Git en el robot fue especialmente útil, al desarrollar el código que se ejecuta en el robot desde el ordenador, cuando se realizan modificaciones hay que mover el código al robot y vía cable es bastante tedioso. Al tener el repositorio de código clonado tanto en el robot como en el ordenador, cuando se quería cargar código nuevo en el robot, bastaba con subir ese código al repositorio, y desde el terminal del robot actualizar el mismo.

Una vez explicadas las diferentes plataformas utilizados en el software del EV3 se va a explicar en detalle como se diseñó el funcionamiento del robot, es decir, se va a explicar el diseño de los bloques API y control de motores y encoder, que se pueden ver en la Figura 15. Para implementar estas dos funcionalidades se desarrolló en Python 3.4 un único programa con varios hilos, un hilo permite ejecutar tareas de forma simultanea dentro de un mismo programa, esto es necesario debido a que el robot debe estar esperando cualquier notificación que le llegue desde el ordenador a la vez que gestiona sus drivers, para ello se utiliza la librería *threading*.

Para empezar, se crea una clase con todas las características y funciones que se van a utilizar asociadas al robot, esta clase se llama Robot y las funciones y variables que contiene se pueden ver en la Figura 16; contiene variables como id, speed, meters, encoder y libre que definen el estado del robot en cada momento, además las variables tank, tankdif, a y b contienen instancias de distintas clases de la librería ev3dev que se

utilizarán para mover los motores y acceder a la información del encoder, y por último contiene un diccionario ordenado donde se irán guardando las órdenes que debe realizar. Se puede ver en la Figura 16 que a la clase Robot se le ha asociada la característica de singleton, esta característica hace que solo sea posible crear una única instancia de la clase Robot y así evitar problemas con el control de su hardware. Las funciones de esta clase corresponden a los movimientos que el robot puede realizar y se van a explicar en detalle en el capitulo 4.

```
"""
Clase que describe al Robot

"""
@singleton
class Robot:

  def __init__(self):
    self.id = 1
    self.tank = MoveTank(OUTPUT_A, OUTPUT_B)
    self.tankdif = MoveDifferential(OUTPUT_A, OUTPUT_B, EV3Rim, STUD_MM)
    self.a = LargeMotor('outA')
    self.b = LargeMotor('outB')
    self.speed = 0
    self.meters = 0
    self.encoder = 0
    self.libre = 0
    self.movements = {}
    OrderedDict(self.movements)



  def Run(self):
    self.tank.on(self.speed*0.995 ,self.speed)

  def stop(self):
    self.tank.off()

  def right(self, deg):
    self.tankdif.turn_right(SpeedPercent((self.speed)),deg)

  def left(self, deg):
    self.tankdif.turn_left(SpeedPercent((self.speed)),deg)
```

Figura 16: Código de la clase Robot

La clase Robot se utilizará en ambos hilos del programa y por tanto será instanciada como una variable global. En la Figura 17 se puede ver el diseño teórico del funcionamiento del software del robot. Para empezar, el ordenador desde el programa de navegación que se explicará en la sección 4.5, le manda al robot datos en formato JSON a su API, estos datos entrarán por distintas rutas que se explicarán en detalle en la sección 4.7.3.1 y contienen una acción que debe ejecutar el robot, una vez llegada esa información la API guardará en el diccionario movements de la clase Robot, la información que contiene ese JSON, por último el proceso de control, está en todo momento verificando si hay alguna acción en movements y por tanto, en el momento

que la haya la ejecuta, y posteriormente notifica al programa de navegación del ordenador que ha terminado.

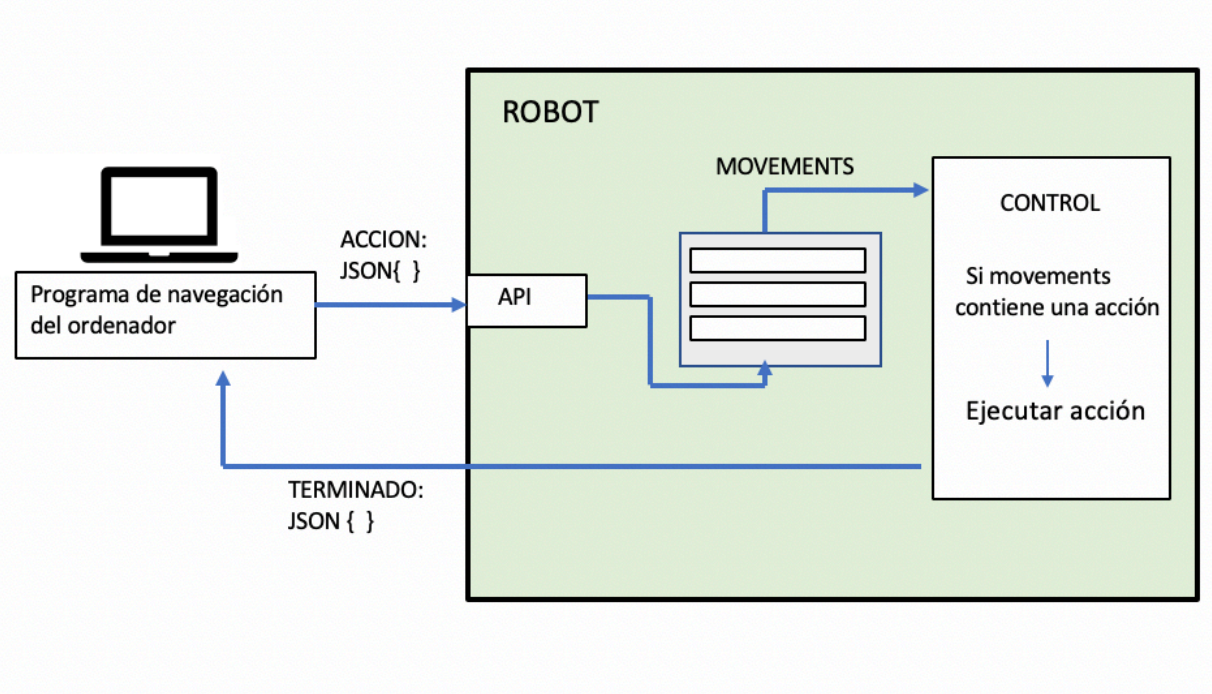


Figura 17: Diagrama de flujo del software del robot

3.2.1 Software del ordenador

En la Figura 18 se puede ver un esquema con las distintas capas de software del ordenador utilizadas, el mas bajo nivel es el sistema operativo, sobre el que se usa un IDE para poder desarrollar los programas en Python tanto de navegación como de visualización. Además, el código desarrollado se guarda a su vez en un repositorio privado de GitLab. En esta sección se van a desarrollar las características de los elementos numerado en la Figura 18.

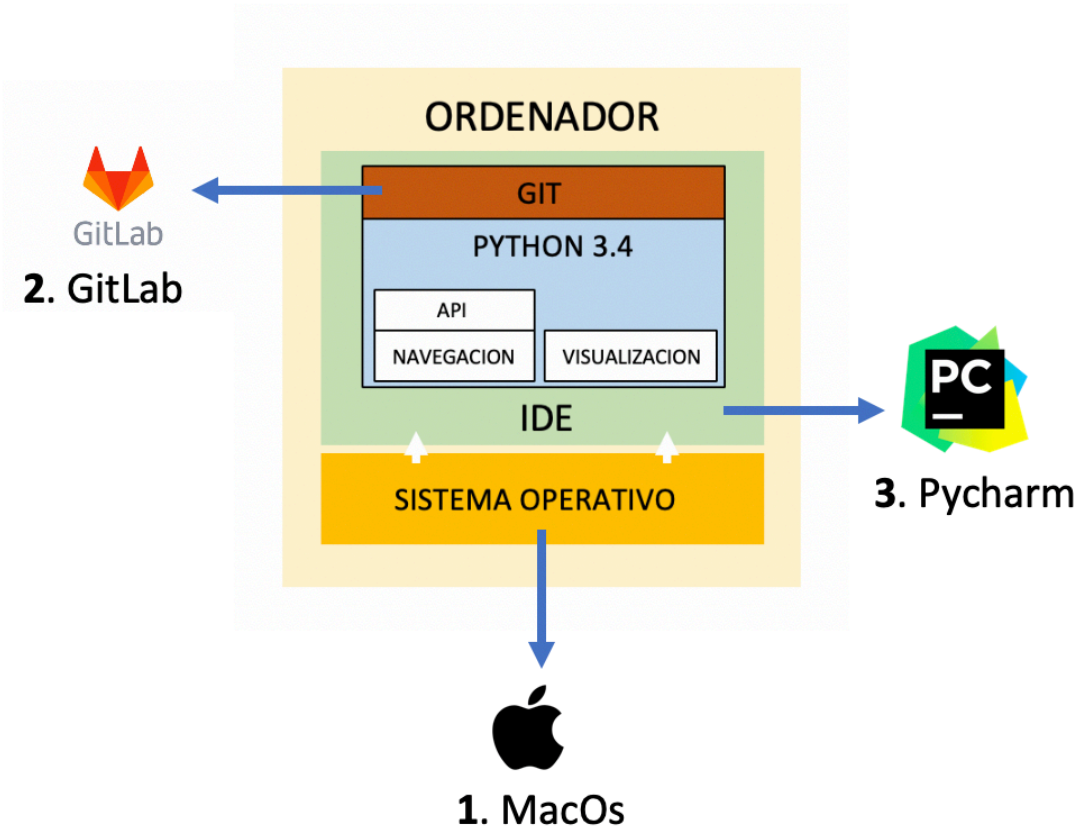


Figura 18: Esquema del software del ordenador

1. **MacOs**: El sistema operativo del ordenador es MacOS Mojave 10.14.3. Este sistema operativo, como todos los demás sirve para gestionar el hardware del ordenador y permite el uso de aplicaciones sobre él para aplicar distintas funcionalidades.

2. **Pycharm:** El IDE utilizado es Pycharm, un IDE gratuito para la comunidad de estudiantes. Este IDE permite desarrollar código de manera más optima ya que contiene un depurador y permite gestionar la venvs y los repositorios de Git desde el mismo IDE.

3. **GitLab:** Se utiliza Gitlab porque a diferencia de sus competidores permite tener repositorios privados de manera gratuita. Con Gitlab se guarda el código en repositorios web y permite tener el control de las versiones anteriores del proyecto.

Con estas tres plataformas se desarrollan los programas de navegación, API y visualización que se pueden ver en la Figura 18 como bloques blancos; los bloques de navegación y API se encargan de calcular las rutas del robot en función de los pedidos y mandar la ordenes al robot, ambos son hilos de un mismo programa, por otro lado, visualización gestiona la pantalla en la que se puede visualizar el mapa del almacén en todo momento y es un programa aparte. En la Figura 19 se puede ver el diagrama de flujos del funcionamiento del software de navegación junto a su API; los pedidos que se vayan a realizar se guardan en una lista, la función llamada asignar pedido verifica si hay algún pedido en la lista, y si lo hay y el robot esta libre, llama a la función gestionar pedido que calcula las rutas a seguir y define los movimientos que se van a tener que realizar, guardándolos en un diccionario ordenado llamado movimientos. La función gestionar pedidos sigue funcionando hasta que se han calculado todas las rutas y movimientos para que el robot llegue a su destino y se van guardando en el diccionario. Paralela a estas funciones existe la función envío, que verifica si hay algún movimiento en la lista y si el robot no esta realizando ninguna acción, manda la siguiente acción al robot en formato JSON. El robot le notifica al sistema de navegación que ha terminado mediante su API que se explica con mas detalle en la sección 4.7.3.2.

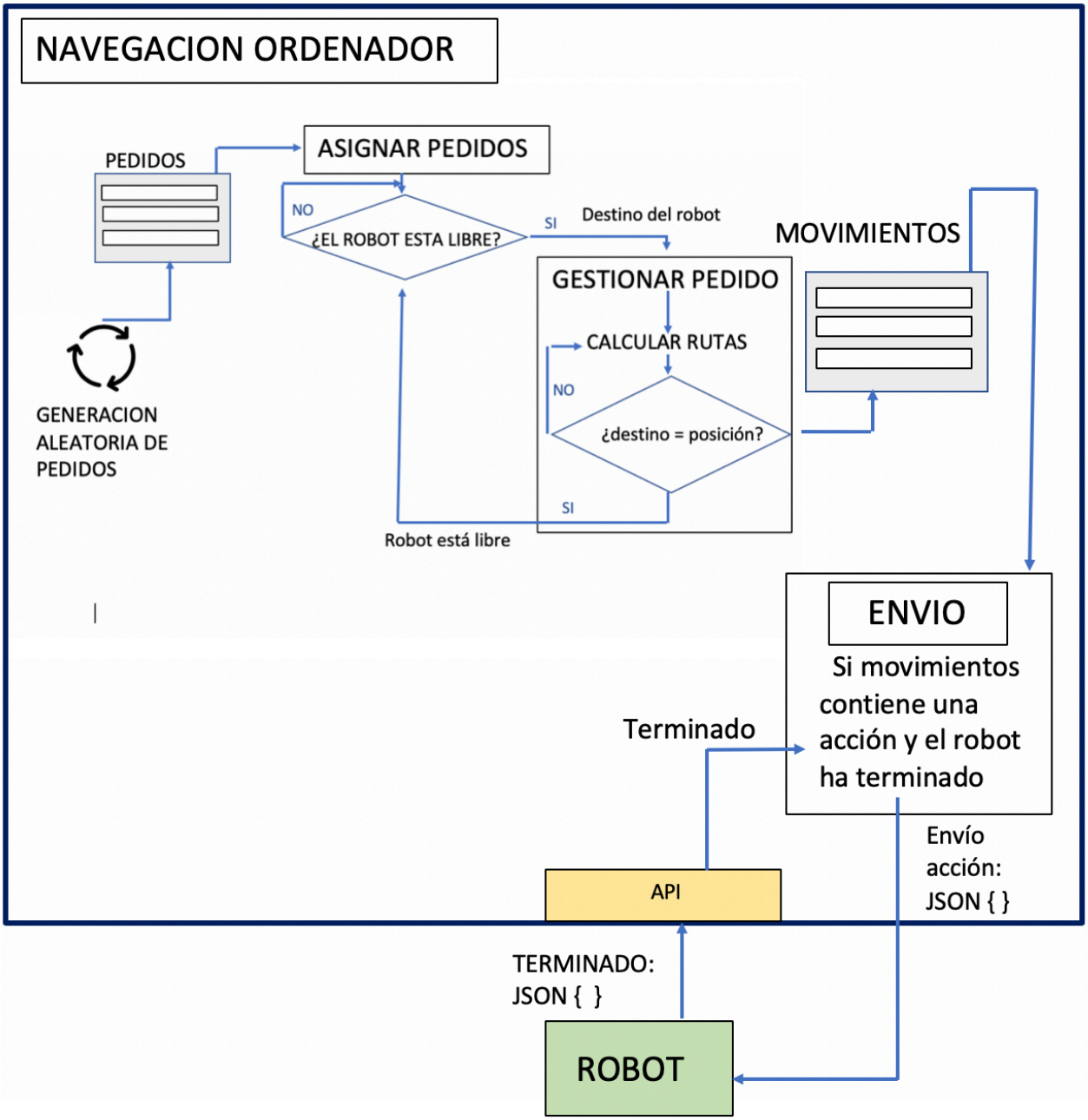


Figura 19: Diagrama de flujos del software del ordenador.

3.3 Comunicaciones

En esta sección, se van a exponer las distintas tecnologías de comunicación utilizadas. En concreto, se va a diferenciar entre las comunicaciones en el medio, entre sistemas operativos y entre procesos.

3.3.1 Comunicaciones en el medio

Las comunicaciones en el medio son aquellas tecnologías que proporcionan el medio físico en el que la información viaja. Se tenia claro desde el principio que para este proyecto el medio de comunicación debería de ser inalámbrico, ya que se esta utilizando un robot que se mueve por el suelo. En concreto, la tecnología utilizada fue wifi, el wifi es más rápido y tiene más ancho de banda que el bluetooth, a parte, permite por ejemplo que se comunique el robot a un servidor en la nube y, por tanto, hace que el proyecto sea mas escalable, mientras que el bluetooth limitaría en este aspecto.

3.3.2 Comunicaciones entre sistemas operativos

Las comunicaciones entre sistemas operativos se usaron para acceder al terminal del Lego EV3 de manera remota. Esto es imprescindible ya que es la única manera de poder instalar los paquetes utilizados en el robot, además permite también, gestionar la ejecución y modificación de archivos y control de errores que en la pantalla del EV3 no aparecen. El tipo de comunicación entre sistemas operativos que se utilizó fue SSH (Secure Shell) que es un protocolo que proporciona una transferencia cifrada entre el cliente y el servidor.

3.3.3 Comunicaciones entre procesos

Para comunicar software con software y poder compartir información como variables o estados se decidió utilizar una Restful-API, el motivo de utilizar tecnología REST y no otra es porque es mas ligera y al tenerse que implantar dentro del EV3 cuanto mas ligera sea mejor. Se utilizará un micro-framework para Python llamado Flask que permite generar RESTful – API`s reduciendo el numero de líneas de código necesarias, ya que abstrae parte de la complejidad al usuario. En concreto, se utilizan dos API´s una para mandar información desde el ordenador hasta el robot y otra para mandar información desde el robot hasta el ordenador. Las rutas y sus métodos se definen en detalle en la sección 4.7.3.

4 Realización del proyecto

En esta sección se va a exponer como se llevó a cabo el proyecto, que pasos se siguieron y como se desarrollaron las distintas partes.

4.1 Montaje EV3

Lo primero que se realizó fue el montaje del EV3 en este caso se quería un montaje fácil y sencillo con dos ruedas y espacio para poder conectar el adaptador wifi sin problemas. En la Figura 20 se puede ver como quedó el robot al terminarlo de montar, se usaron dos ruedas de tipo tanque porque eran las mas estables y al estar todo el tren acoplado al eje del motor se evita que derrapen, al contrario que si se usa un montaje de 4 ruedas. Se utilizan dos motores, el derecho conectado al puerto de salida B y el izquierdo al puerto A.

Figura 20: Montaje del EV3

4.2 Software del robot

Para empezar a usar el EV3, lo primero que se hizo fue instalarle todos los programas y paquetes necesarios para darle las funcionalidades que se quieren implantar.

4.2.1 Instalación del sistema operativo

Como se mencionó en la sección 3.2.1 el sistema operativo que se va a utilizar para el EV3 es *ev3dev*, para ello hay que grabar una imagen que contiene el nuevo sistema operativo en la SD y así poder arrancar el robot desde ella, sustituyendo al sistema operativo Lego Mindstorms. Los pasos que se siguieron son los siguientes:

Primer paso: Se grabó la imagen del sistema operativo en la SD, para ello se descargó la imagen del sistema opertivo desde la web oficila de ev3dev y se utilizó el programa *balenaEtcher* para grabarla en la SD. Como se puede ver en la Figura 21 al abrir el programa *balenaEtcher* bastanta con seleccionar la imagen descargada, seleccionar la SD y darle a Flash.

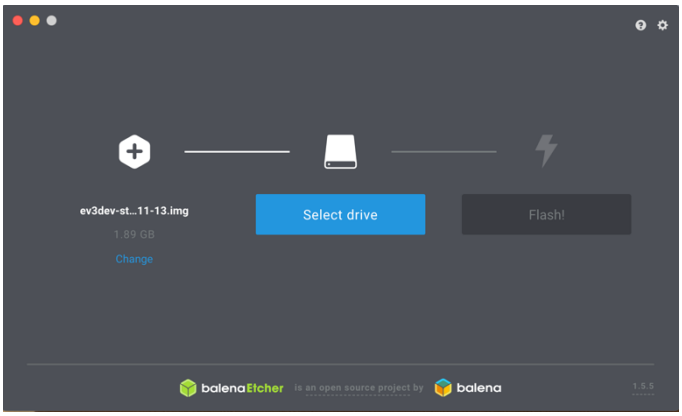


Figura 21: Programa BalenaEtcher

Segundo paso: Por último, se enciende el EV3 con la tarjeta SD metida para que arranque con el nuevo sistema operativo, a diferencia del arranque normal en la pantalla aparece el logo personalizado de *ev3dev.*

4.2.2 Instalación de PIP

Para poder usar paquetes de Python en el robot primero se tiene que instalar el administrador de paquetes pip-3 y para ello primero se establece una conexión SSH con el EV3 para poder acceder de manera remota al terminal del mismo. Una vez establecida esa conexión se ejecutan los siguientes comandos en su terminal:

- `sudo apt-get update:` este comando actualiza los punteros
- `install python3-pip:` este comando instala el administrador de paquetes pip.

Una vez instalado pip3, se podrá instalar cualquier paquete en el EV3.

4.2.3 Virtualenv

Igual que con el administrador de paquetes, para usar el *virtualenv* en el EV3 primero se tiene que instalar, para ello se ejecutan los siguientes comandos en el terminal del EV3:

- `apt-get install virtualenv virtualenvwrapper python3-setuptools python3-smbus python3-pilq`
- `source /etc/bash_completion.d/virtualenvwrapper`

```
- mkvirtualenv ev3_py34 --python=/usr/bin/python3.4 --
  system-site-packages
```

Una vez ejecutados estos comandos ya se tiene disponible un *virtualenv* llamado ev3_py34 que permitirá guardar en él, todas las librerías y paquetes; para trabajar dentro del *virtualenv* bastará con ejecutar `workon ev3_py34` en el terminal de la consola.

4.3 Movimientos del robot

Una vez terminado de instalar todo el software en el EV3 se generó un programa que le diese los movimientos básicos, es decir, moverse recto, girar en torno a su eje tanto a la izquierda como a la derecha y pararse. Este programa se ejecuta dentro del EV3 para después poder mandarle ordenes desde fuera sobre que movimiento realizar.

4.3.1 Movimiento Recto

Para el movimiento recto del robot, se necesitaba no únicamente que fuese recto, sino que se pudiese especificar cuantas unidades del encoder se quería mover el robot, para así poder controlar las distancias que recorría. Para ello se usaron varias funciones de la librería ev3dev.

Para empezar, se valoró la opción de usar la función **run_to_abs_pos()** de la que se puede ver el código en la Figura 22.

```
def run_to_abs_pos(self, **kwargs):
    """
    Run to an absolute position specified by `position_sp` and then
    stop using the action specified in `stop_action`.
    """
    for key in kwargs:
        setattr(self, key, kwargs[key])
    self.command = self.COMMAND_RUN_TO_ABS_POS
```

Figura 22: código de la función run_to_abs_pos() de la librería ev3dev

Esta función pertenece a la clase ***LargeMotor***, de la que se instancia un objeto por cada motor que se quiera utilizar. Se puede ver en la Figura 22 que a esta función se le pasan unos argumentos llamados ****kwargs**, que lo que permiten es no limitar ni el numero ni el tipo de argumentos que se le pasan a la función. Luego, dentro de esta función se comparan esos **kwargs** con una serie de key´s que son palabras clave definidas anteriormente. Para poder mover los motores con esta función basta con pasarle la posición final en la que se quiere establecer el encoder y la velocidad a la se quiere que vaya.

Como **run_to_abs_pos( )** resultó no ser demasiado precisa, se decidió personalizar el movimiento del robot con una serie de funciones mas básicas, en concreto se usaron tres funciones del ev3dev para mover el robot y son las siguientes:

La función **on,** de la clase **MoveTank** permite accionar los motores a la velocidad que se le especifique en *left_speed* y *right_speed*. En la Figura 23 se puede ver el código de esta función, directamente sacado de la librería ev3dev.

```
def on(self, left_speed, right_speed):
    """
    Start rotating the motors according to ``left_speed`` and ``right_speed`` forever.
    Speeds can be percentages or any SpeedValue implementation.
    """
    (left_speed_native_units, right_speed_native_units) = self._unpack_speeds_to_native_units(left_speed, right_speed)

    # Set all parameters
    self.left_motor.speed_sp = int(round(left_speed_native_units))
    self.right_motor.speed_sp = int(round(right_speed_native_units))

    # This debug involves disk I/O to pull speed_sp so only uncomment
    # if you need to troubleshoot in more detail.
    # log.debug("%s: on at left-speed %s, right-speed %s" %
    #     (self, self.left_motor.speed_sp, self.right_motor.speed_sp))

    # Start the motors
    self.left_motor.run_forever()
    self.right_motor.run_forever()
```

Figura 23: función on() de la librería ev3dev

La función **off** también de la clase **MoveTank** se utiliza para parar ambos motores a la vez. Se puede ver el código en la Figura 24.

```
def off(self, motors=None, brake=True):
    """
    Stop motors immediately. Configure motors to brake if ``brake`` is set.
    """
    motors = motors if motors is not None else self.motors.values()

    for motor in motors:
        motor._set_brake(brake)

    for motor in motors:
        motor.stop()
```

Figura 24: función off() de la librería ev3dev

Por último, se usó la propiedad **position,** una propiedad de la clase **LargeMotor**, que devuelve la posición del encoder. Se puede ver el código en la Figura 25.

```
@property
def position(self):
    """
    Returns the current position of the motor in pulses of the rotary
    encoder. When the motor rotates clockwise, the position will increase.
    Likewise, rotating counter-clockwise causes the position to decrease.
    Writing will set the position to that value.
    """
    self._position, value = self.get_attr_int(self._position, 'position')
    return value
```

Figura 25: propiedad position() de la librería ev3dev

Con las dos funciones, **on()** y **off (),** y la propiedad **position** se consiguió mover el EV3 de manera mas precisa. Se le llamó **run_until ()** a esta nueva función que lo que hace es encender los motores e ir comparando el valor del encoder en cada momento, con el valor al que debería llegar, cuando llega o en su defecto se pasa, se paran los motores. En la Figura 26 se puede ver el esquema de funcionamiento de **run_until ().**

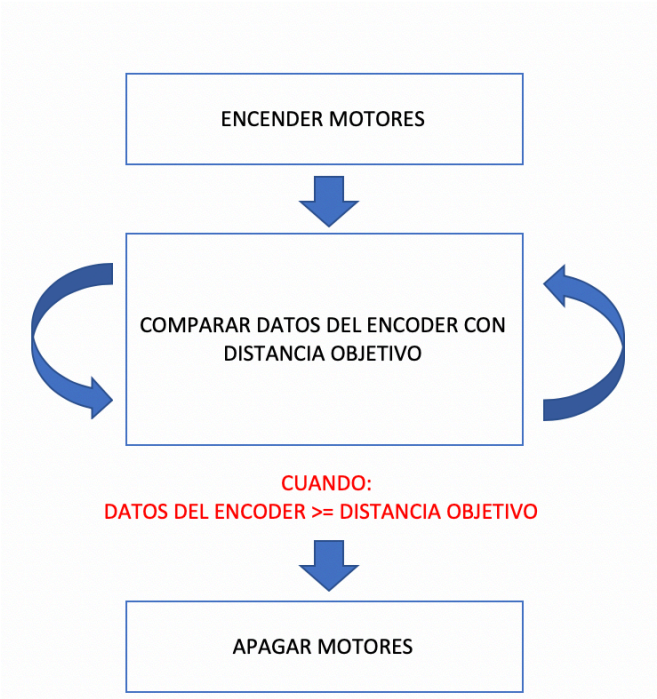


Figura 26: Esquema del funcionamiento de la función run_until()

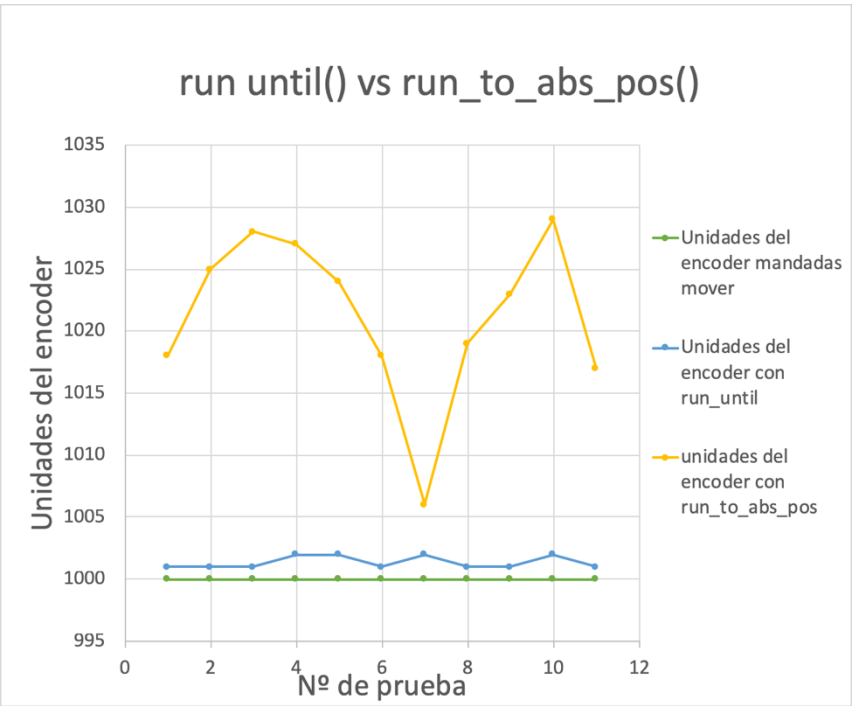
Para comprobar cuanto más preciso era la nueva función **run_until( )** que **run_to_abs_pos( )** se procedió a hacer una serie de pruebas, en concreto, se le mando al robot moverse unas determinadas unidades del encoder y se analizó cuanto se había movido en realidad.

Como se puede ver en la Tabla 2 la función **run_to_abs_pos()** se suele desviar entre 15 y 30 unidades del encoder, si cada unidad del encoder representa un grado de rotación, el robot se esta moviendo bastante mas de lo que debería, en cambio con **run_until()** la desviación ronda una o dos unidades, por eso se decidió utilizar esta función para la realización del proyecto.

Tabla 2: Resultados de movimientos del encoder con las funciones run_until y run_to_abs_pos

Tabla de unidades movidas por el encoder vs mandadas a mover		
Unidades del encoder mandadas a mover	Unidades del encoder movidas **run_until()**	Unidades del encoder movidas **run_to_abs_pos()**
1000	1001	1018
1000	1001	1025
1000	1001	1028
1000	1002	1027
1000	1002	1024
1000	1001	1018
1000	1002	1006
1000	1001	1019
1000	1001	1023
1000	1002	1029
1000	1001	1017

Para poder ver los resultados mas claramente, se ha elaborado la Gráfica 1; donde se puede ver en verde, las unidades teóricas que se debería mover el encoder con ambas funciones, en amarillo las unidades que se mueve realmente con **run_to_abs_pos()** y en azul las de **run_until().** En el grafico, se puede ver que run_to_abs_pos se queda claramente mas lejos del valor teórico que run_until.



Gráfica 1: Datos del movimiento del encoder con run_until y run_to_abs_pos

4.3.2 Giros

Para girar el robot sobre su mismo eje, se activan ambos motores en sentidos contrarios. Así, si ambos motores giran a la misma velocidad, el robot creara una circunferencia alrededor suyo de diámetro igual a la separación entre sus ruedas. En la Figura 27 se puede ver que en el caso del EV3 utilizado para este proyecto que la distancia entre ruedas es de 17.8 cm.

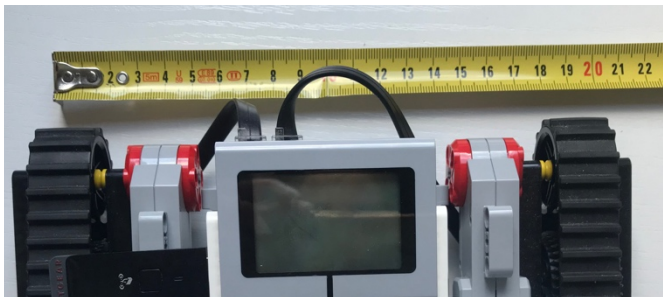


Figura 27: Imagen de la distancia entre ruedas en el EV3 utilizado

Una vez sabida la distancia entre ruedas, se pueden usar la Ecuación 1 y Ecuación 2 para sacar cuantas unidades del encoder se tienen que girar los motores para que el robot gire en torno a su eje unos grados determinados.

La Ecuación 1 calcula el arco a realizar por el EV3 si quiere girar X grados:

$$Arco\ a\ realizar = \frac{X^{\underline{o}}}{360} * distancia\ entre\ ruedas$$

Ecuación 1 Arco a realizar por EV3 para girar X grados.

La Ecuación 2 convierte ese arco en unidades del encoder, para poder controlarlo y pararlo cuando haya llegado a la posición deseada.

$$Unidades\ del\ encoder = \frac{Arco\ a\ realizar}{Circunferencia\ de\ la\ rueda}$$

Ecuación 2 Unidades del encoder que se tienen que mover los motores en función al arco a realizar

Al estar usando unas ruedas tipo tanque, con una cinta larga, se usa como circunferencia de las ruedas la que se puede ver con una flecha roja en Figura 28, es decir, la de la rueda que va acoplada al eje del motor.

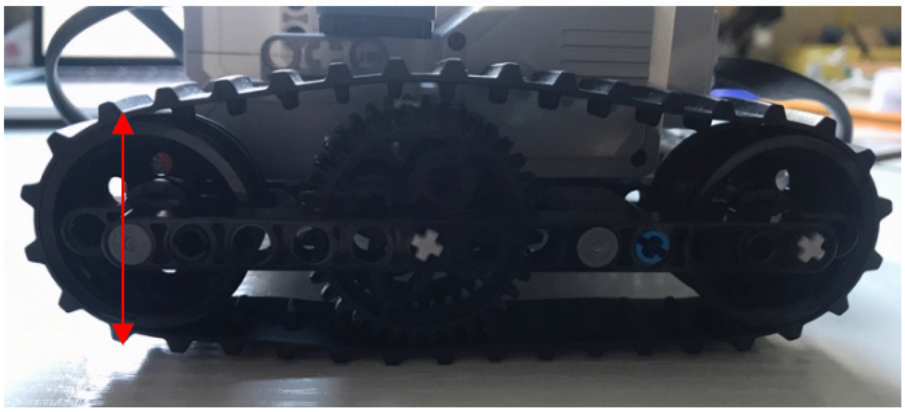


Figura 28: Imagen mostrando el diámetro de rueda utilizado

Para girar el robot se utiliza la clase **MoveDifferential** de la librería *ev3dev*, esta clase permite girar el robot en el sitio, utilizando la Ecuación 1 y la Ecuación 2. En concreto, se usan las dos funciones que se pueden ver en la Figura 29 **turn_left**, mueve el robot en sentido antihorario y **turn_right** en sentido horario. La única diferencia entre estas dos funciones es que cambia que rueda gira hacia delante y cual hacia detrás.

```
def turn_right(self, speed, degrees, brake=True, block=True):
    """
    Rotate clockwise 'degrees' in place
    """
    self._turn(speed, abs(degrees), brake, block)

def turn_left(self, speed, degrees, brake=True, block=True):
    """
    Rotate counter-clockwise 'degrees' in place
    """
    self._turn(speed, abs(degrees) * -1, brake, block)
```

Figura 29: Código de las funciones turn_left y turn_right

4.4 Calibración del robot

Una vez conseguido que el robot se mueva recto y gire entorno a su eje, se procedió a calibrar estos movimientos. Como el robot estará navegando únicamente con la información del encoder, es muy importante que los movimientos sean precisos, porque un pequeño error, al cabo de muchos movimientos se va acumulando y puede descolocar al robot totalmente de donde debería de estar.

4.4.1 Calibración del movimiento recto

Dentro de la calibración del movimiento recto del robot hay dos partes, una es lo recto que va y otra es si se ha movido la distancia que efectivamente se le ha mandado. Como se menciona anteriormente en la sección 4.3.1 la función **run_until()** es muy precisa ajustando las distancia que se mueve el robot con la distancia objetivo y por tanto no hará falta calibrar esa parte.

Se procede a calibrar como de recto se mueve el EV3, se comprobó que, al activar los dos motores a la misma velocidad, el robot no iba exactamente recto, porque uno de los dos motores daba mas potencia que el otro. Para calibrarlo se fue cambiando la velocidad de uno de los motores para que acabase avanzando lo más recto posible. Al hacer un par de pruebas se llegó a la conclusión de que se debía multiplicar la velocidad del motor izquierdo por 0.995 ya que estaba avanzando mas que el derecho. En la Figura 30 se puede ver como queda la función que manda activar los motores después de hacer la calibración.

```
def Run(self):
    self.tank.on(self.speed*0.995, self.speed)
```

Figura 30: Código de la función que activa los motores después de la calibración

4.4.2 Giros

Los giros, son el movimiento mas importante a la hora de calibrar, si los giros no son precisos, al mover el robot recto el error se magnifica. Como se puede ver en la Figura 31, un error de solamente cinco grados en el giro, si después se avanza medio metro, se convierte en una desviación de 0.04 metros del punto de destino, aproximadamente un 10% de error, esto es inaceptable para el funcionamiento de este proyecto y por eso se procedió a calibrar exhaustivamente los giros del robot.

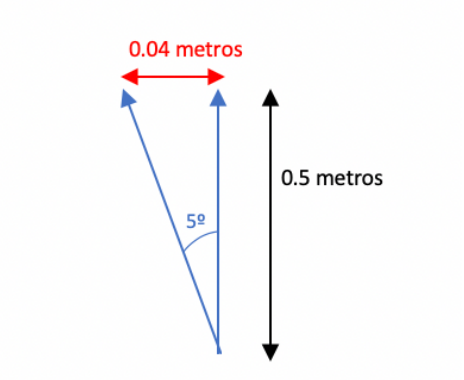


Figura 31: Desviación del robot por un error de 5 grados

Para la calibración de los giros, se fueron modificando variables y comprobando el resultado. El giro del EV3 depende de 3 parámetros como vimos en la sección 4.3.2 y son; la velocidad de los motores, la distancia entre las ruedas y el diámetro de estas. Por simplicidad se decidió modificar únicamente una variable que fue STD_MM que es la distancia entre ruedas.

En la Tabla 3 se puede ver como fue el proceso, se muestran a la izquierda la imagen del robot en su posición inicial y en la derecha una imagen del robot después de haber girado dos vueltas enteras entorno a su eje, según se baja en la tabla y se modifica STD_MM se puede ver que la posición final es mas parecida a la inicial y que por tanto el giro ha sido mas exacto. Finalmente, el STD_MM utilizado fue 162.5 mm y con ese valor los giros salen suficientemente precisos.

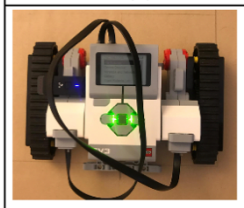
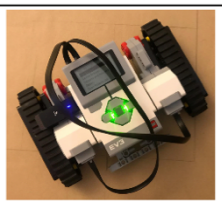
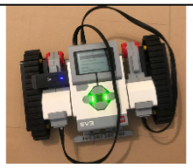
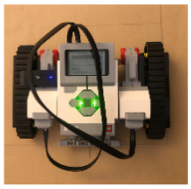
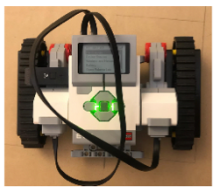
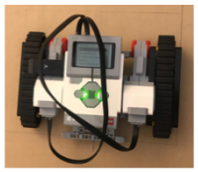
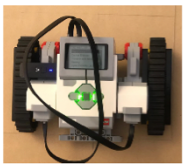
GIRO HACIA LA IZQUIERDA	
STD_MM = 170 mm	
Inicio	Fin
	
STD_MM = 165 mm	
Inicio	Fin
	
STD_MM = 163.5 mm	
Inicio	Fin
	
STD_MM = 162.5 mm	
Inicio	Fin
	

Tabla 3: Muestra la calibración de los giros del EV3

4.5 Navegación en el mapa

Una vez conseguido que los movimientos del robot fuesen precisos, se procedió a determinar como se iba a mover el robot en el mapa, el programa que calcula las rutas del robot se ejecuta en el ordenador. Para la navegación del robot se dividió el mapa en cuadrados, cada cuadrado tiene asignada una coordenada que es la de su centro y se va a ir moviendo de cuadrado en cuadrado hasta llegar a su destino. Para explicar como navega el robot dentro del mapa se va a dividir la explicación en dos partes, la primer es la elección del siguiente cuadrado y la segunda es como se mueve el robot de cuadrado en cuadrado.

4.5.1 Elección de la siguiente casilla

Una vez establecida la casilla final a la que el robot quiere llegar, el algoritmo va a ir tomando la decisión sobre a que casilla se mueve el robot, así de casilla en casilla se llegará al destino final. Para decidir cual es la siguiente casilla, se procede como se va a explicar a continuación:

Primer paso: Se analizan todos los cuadrados adyacentes a la posición del robot, como podemos ver en la Figura 32 si el robot esta en la posición A y el destino es B se analizarán las casillas 1, 2, 3 y 4 que son las que están en rojo.

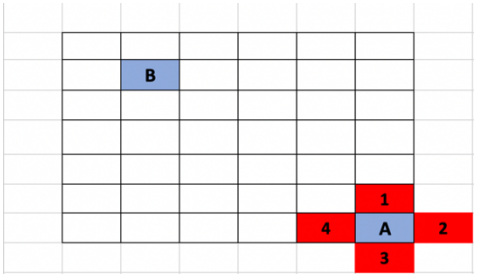


Figura 32: Primer paso de la elección de la siguiente casilla

Segundo paso: Una vez localizados todos los cuadrados adyacentes, se descartan aquellos que no son posibles, eso son los que estén fuera de los limites del mapa o los que contengan en ellos un punto de estantería o recogida distinto al destino. Como se puede ver en la Figura 33 en este ejemplo se descartarían las casillas 2 y 3 por estar fuera de los limites de el mapa

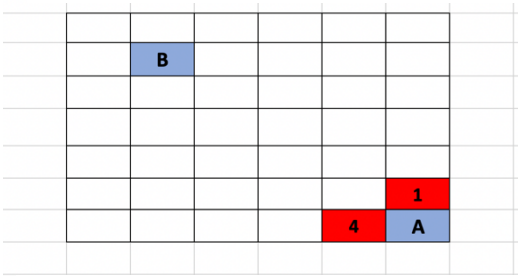


Figura 33: Segundo paso de la elección de la siguiente casilla

Tercer paso: se decide a que casilla dentro de las posibles se va a mover el robot y para ello se calcula la distancia entre las casillas y el cuadrado objetivo y se elige la que minimice esa distancia.

Se repetirán estos pasos, hasta que el robot llegue al cuadrado de destino.

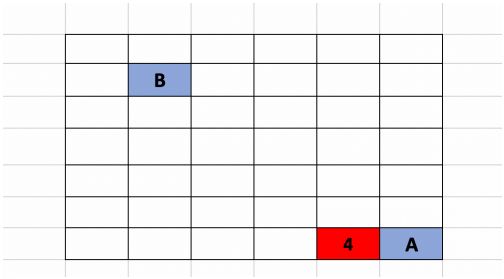


Figura 34: Tercer paso de la elección de la siguiente casilla

En el caso de llegar al tercer paso, y tener dos casillas que tengan la misma distancia al objetivo, se elige aquella casilla que minimice los movimientos que necesita hacer el robot, es decir se elige aquella casilla que evite que el robot tenga que girar. Así se evitan giros innecesarios por parte del robot y se hace que la navegación sea más optima.

Al haber casillas del mapa que son estanterías, no se quiere que el robot las esté cruzando todo el rato y por eso el sistema de navegación, no deja que el robot pase por ellas, a no ser que vayan a recoger un pedido justo en esa estantería. Las estanterías, son analizadas por el algoritmo de navegación como obstáculos a evitar, y en el caso de querer utilizar más de un robot, la manera de evitar colisiones entre ellos sería exactamente la misma, al hacer el análisis los robots serían otro obstáculo más en el algoritmo.

4.5.2 Movimientos hacia la siguiente casilla

Cuando el programa ya sabe cual es la siguiente casilla se procede a calcular los movimientos que tiene que hacer el robot para llegar hasta ella. El movimiento del robot hacia la siguiente casilla consta de dos pasos, el primero es el giro entorno a su propio eje, para orientar el robot hacia su destino y el segundo es mover recto el robot la distancia entre su posición y la siguiente casilla. Para poder calcular cuanto tiene que girar el robot y la distancia que debe avanzar, se guardan cuatro variables en la instancia del robot y son las siguientes:

- **Coordenada actual del robot**: Variable que almacena la coordenada del centro de la casilla en la que se encuentra el robot.
- **Coordenada de la siguiente casilla:** Variable que almacena la coordenada de la siguiente casilla a la que se va a mover el robot
- **Recta anterior:** variable que almacena la recta que ha realizado el robot en su movimiento anterior, en su defecto, si es el primer movimiento, se tiene asignada la recta de su orientación inicial que es siempre la misma.
- **Recta objetivo:** la recta a realizar por el robot, es decir la que une la coordenada actual con la coordenada de la siguiente casilla.

Con estas variables primero se calcula el giro del robot, para ello se utiliza la librería *sympy* que tiene un modulo de geometría que permite calcular ángulos entre rectas. En concreto se usa la función ***closing_angle*** a la que se le pasan dos segmentos de rectas y devuelve el ángulo mas pequeño entre ellas. En la Figura 35 se puede ver el ángulo que giraría el robot en el ejemplo que usamos anteriormente, si se quiere ir hacia la casilla 4 desde A, se calcula el ángulo entre L1 que es la recta anterior por defecto al principio y L2 que es la recta objetivo.

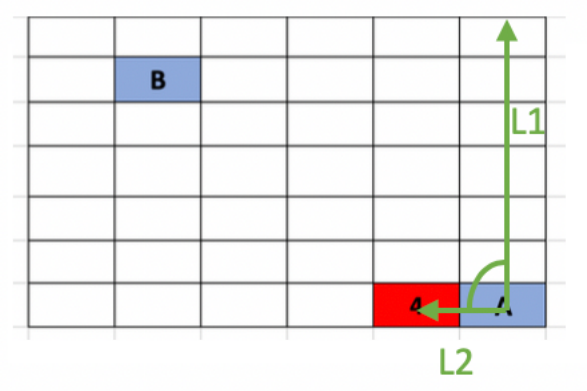


Figura 35: Mapa que representa que ángulo gira el robot

Una vez se ha orientado el robot, se avanza hacia la casilla siguiente; para ello se calcula la distancia entre la coordenada actual del robot y la coordenada de la siguiente casilla. Se usa la función ***distance*** de la librería **sympy**, que devuelve la distancia euclidia entre dos puntos.

4.6 Ordenes desde el ordenador

Una vez se tiene el software que le da los movimientos básicos al robot funcionando como vimos en la sección 4.4 y el programa que calcula las rutas que debe seguir el EV3 como se explica en la sección 4.5 queda generar las ordenes que va a dar el ordenador al robot para que todo funcione conjuntamente. Habrá tantas ordenes como movimientos se le han asignado al robot, así se tiene total control sobre el desplazamiento del robot. Las ordenes son las que se pueden ver en la Tabla 4.

Tabla 4: Ordenes desde el ordenador

Orden	**Descripción**
Recto	Orden que manda al robot moverse recto, hay que especificarle cuantas unidades del encoder se tendrá que mover y a que velocidad se quiere que se mueva
Derecha	Orden que manda al robot girar hacia la derecha en torno a su propio eje, se le tiene que especificar cuantos grados se quiere mover y la velocidad de giro
Izquierda	Orden que manda al robot girar hacia la izquierda en torno a su propio eje, se le tiene que especificar cuantos grados se quiere mover y la velocidad de giro
Parar	Orden que manda al robot pararse.

4.7 Comunicaciones

Una vez determinadas las ordenes que se le van a dar al robot, se procede a gestionar las comunicaciones, es decir diseñar como van a llegar esas ordenes desde el ordenador al robot.

4.7.1 SSH

Durante la preparación del proyecto se usó varias veces la conexión SSH con el EV3, esta conexión permite acceder remotamente al terminal. Para realizar la conexión SSH con el robot primero se tiene que conectar a la red, es decir, tener una IP asignada. Para ello, se debe conectar al router vía wifi y como meter la contraseña del wifi directamente en el robot es bastante tedioso, se conectó vía USB al ordenador y se le compartió internet por cable. Una vez conectado, el robot ya tiene una IP asignada que en este caso fue **192.168.1.105.** Con esta IP y ejecutando el comando `robot@192.168.1.105` en el terminal del ordenador como se puede ver en la Figura 36 se accede al terminal del EV3. Esta conexión se utiliza tanto para instalar todo tipo de paquetes en el EV3 como se vio en la sección 4.2.2 como para ejecutar scripts en el propio robot.

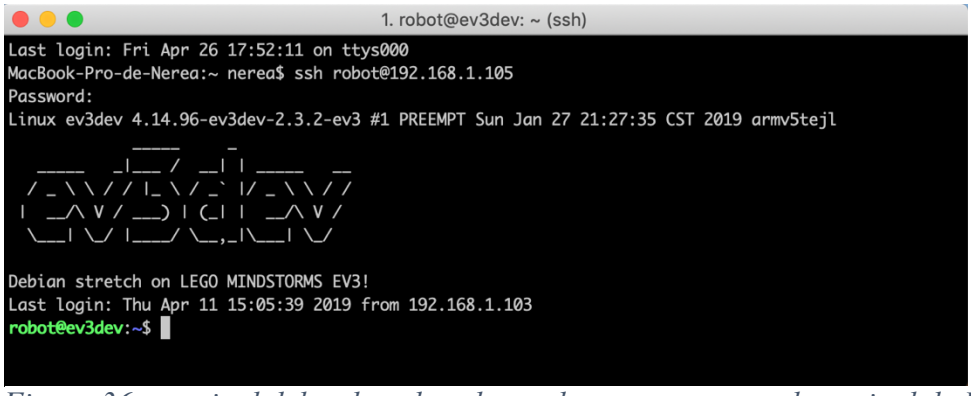


Figura 36: terminal del ordenador al acceder remotamente al terminal de EV3

4.7.2 Wifi

Como se comentó en la sección 3.1.2 el robot necesita un adaptador para poder conectarse a la red wifi, una vez enchufado el adaptador al puerto USB del robot este ya puede ser parte de la red. A continuación, se estableció una conexión SSH con el EV3 y una vez se accede a su terminal se utiliza *Connman*; un gestor de conectividades para dispositivos Linux con el que se puede gestionar las conectividades desde el terminal de forma rápida. Se puede ver en la Figura 37 el terminal del EV3 y los pasos que se dieron para conectarlo al wifi.

```
robot@ev3dev:~$ connmanctl
Error getting VPN connections: The name net.connman.vpn was not provided by any
connmanctl> enable wifi
Enabled wifi
connmanctl> scan wifi
Scan completed for wifi
connmanctl> services
*AO Wired                ethernet_b827ebbde13c_cable
                         wifi_e8de27077de3_hidden_managed_none
    AH04044914           wifi_e8de27077de3_41483034303434393134_managed_psk
    Frissie              wifi_e8de27077de3_46726973736965_managed_psk
    ruijgt gast          wifi_e8de27077de3_7275696a67742067617374_managed_psk
    schuur               wifi_e8de27077de3_736368757572_managed_psk
connmanctl> agent on
Agent registered
connmanctl> connect wifi_e8de27077de3_41      # You can use the TAB key at this poin
connmanctl> connect wifi_e8de27077de3_41483034303434393134_managed_psk
Agent RequestInput wifi_e8de27077de3_41483034303434393134_managed_psk
  Passphrase = [ Type=psk, Requirement=mandatory ]
Passphrase? ************
Connected wifi_e8de27077de3_41483034303434393134_managed_psk
connmanctl> quit
```

Figura 37: Terminal del EV3 con los comandos para conectarlo al wifi con Connman

Para utilizar wifi es necesario tener cerca del dispositivo un router conectado a internet mediante ethernet, este router se puede definir como un intermediario, entre cliente y servidor, que sirve para transformar señales radio en binarias y viceversa. En la Figura 37 se puede ver como es la conectividad del robot y el ordenador en la red domestica en la que se está utilizando, tanto el router como el ordenador y el robot tienen asignada un IP privada que es la que utilizarán para comunicarse entre ellos. El router, a parte de tener la IP privada con la que se comunica dentro de esta red domestica, también tendrá una IP publica de cara a la conexión con la red de la operadora, esta IP publica tiene que ser a diferencia de la privada única en el mundo.

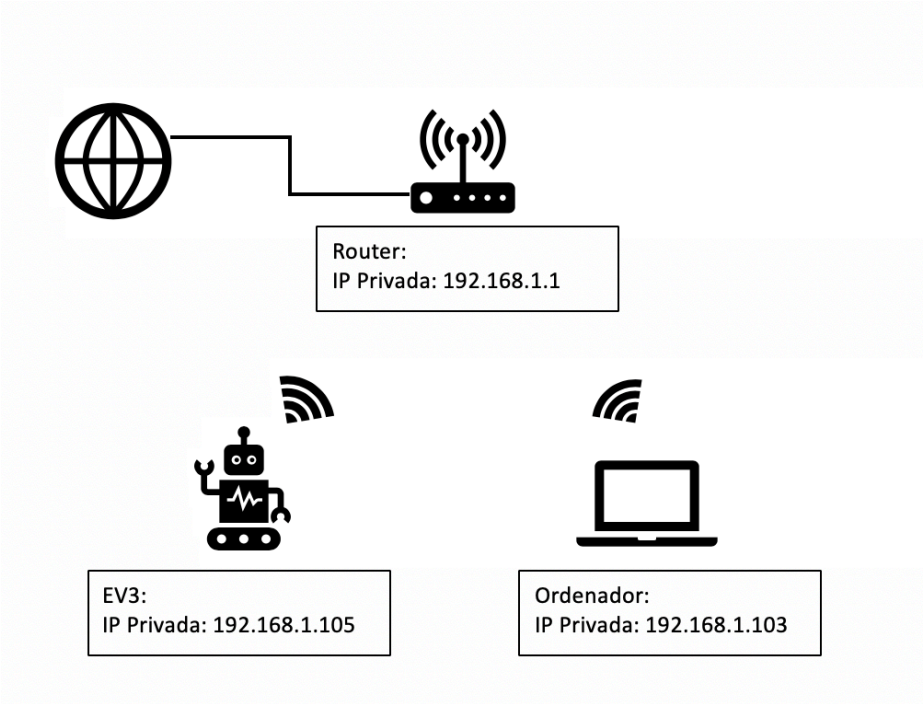


Figura 38: Esquema de IP´s en los dispositivos utilizados

4.7.3 RESTful-API

Una API (Application Programming Interface) es un interfaz para la comunicación entre dos programas con un conjunto de normas y protocolos estándar; en este caso se va a utilizar una API para comunicar el robot con el ordenador y viceversa. Una RESTful-API, utiliza el estándar REST que transporta datos mediante el protocolo HTTP entre cliente y servidor, REST permite enviar cualquier tipo de datos por ejemplo XML, JSON o binarios; en este caso, se enviarán únicamente JSON. Python tiene un micro-framework llamado Flask, del que ya hemos hablado en la sección 3.2.1 que permite generar un RESTful-API con un numero no demasiado alto de líneas de código.

En este proyecto se distinguen dos API`s independientes, una que permite enviar ordenes desde el ordenador al robot y otra que permite que el robot le notifique al ordenador cuando ha terminado y le proporcione datos sobre su encoder. Para no acumular ordenes en el robot, solo se le pasará una nueva orden cuando notifique que ha terminado de moverse, esto permite cambiar la ruta mas fácilmente si algo ocurriese.

4.7.3.1 API Ordenador – Robot

Esta API es la que se encarga de mandar las ordenes del ordenador al robot, se ha diseñado la API con cuatro rutas como podemos ver en la Figura 39.

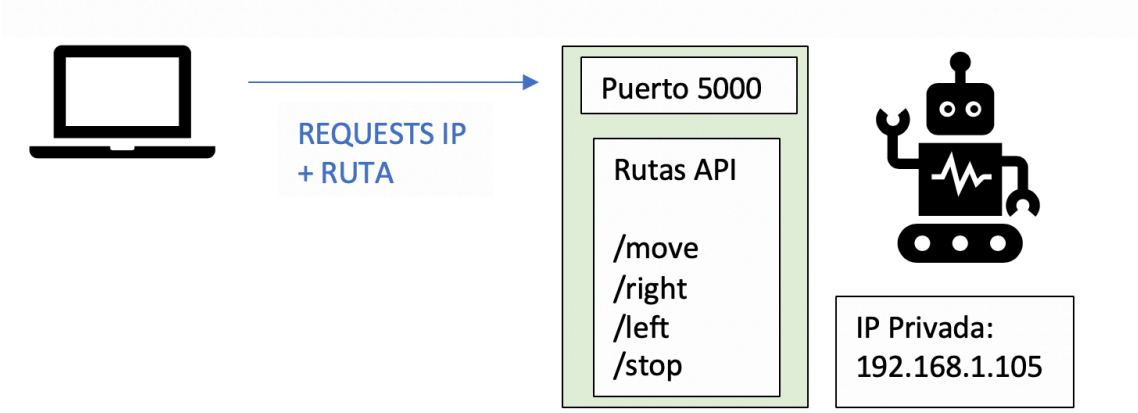


Figura 39: Esquema del funcionamiento de la API ordenador-robot

El robot esta escuchando por el puerto 5000 a cualquier IP que le mande un request correspondiente a una de las cuatro rutas que se han diseñado, las rutas se corresponden con las ordenes que el ordenador le puede mandar al robot. A continuación, se van a explicar un poco mas a fondo cada una de las rutas.

/move: esta ruta es de tipo POST, el ordenador le pasará un JSON que contiene los metros que se tiene que mover el robot y la velocidad a la que se mueve.
- url: 192.168.1.105:5000/move
- JSON {“metros”: , “velocidad”: }

/right: esta ruta es de tipo POST y se le pasará un JSON con los grados que tiene que girar el robot hacia la derecha y la velocidad de giro**.**
- url: 192.168.1.105:5000/right
- JSON {“grados”: , “velocidad”: }

/left: esta ruta es de tipo POST y se le pasará un JSON con los grados que se tiene que girar el robot hacia la izquierda y la velocidad de giro.
- url: 192.168.1.105:5000/left
- JSON {“grados”: , “velocidad”: }

/stop: esta ruta es de tipo GET cuando se hace un GET desde el ordenador a esta ruta el robot se parará.
- url: 192.168.1.105:5000/stop
- GET

4.7.3.2 API Robot-Ordenador

Este API se encarga de notificarle al ordenador cuando el robot se ha terminado de mover, y puede entonces el ordenador mandarle la siguiente orden, a parte, también es la que proporciona los datos del encoder al ordenador. En la Figura 40 se puede ver un esquema de como funciona esta API y que rutas tiene.

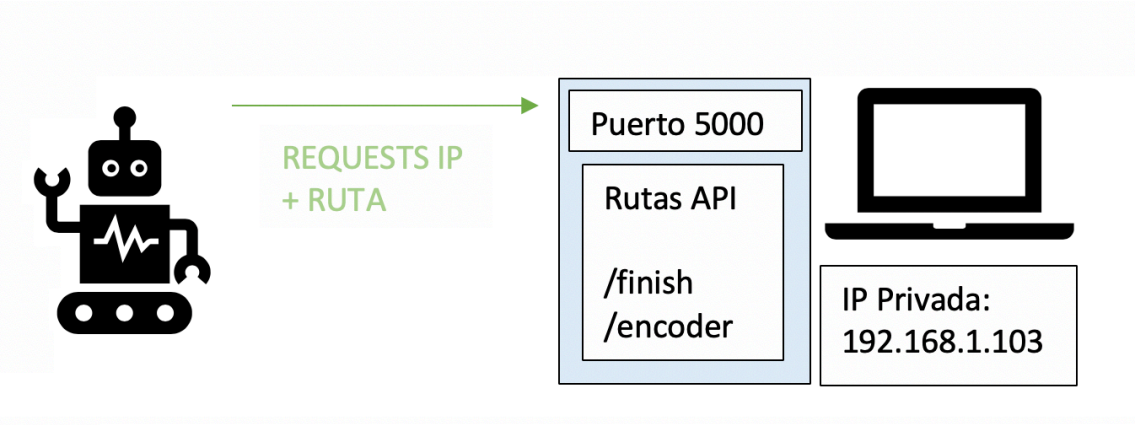


Figura 40: Esquema del funcionamiento de la API Robot - Ordenador

El ordenador esta escuchando por el puerto 5000 a cualquier request que le llegue a su IP con la ruta correspondiente. En esta API hay dos rutas una para notificar al sistema que el robot ha terminado de moverse y otra para pasarle los datos del encoder, a continuación, se van a explicar estas rutas mas detalladamente.

/finish: esta es una ruta tipo POST a la que el robot pasa su id y así en el caso de tener más de un robot, se puede saber cual de los robots esta notificando que ha terminado.
- url: 192.168.1.103:5000/finish
- JSON {“id”: }

/encoder: esta ruta es de tipo POST y el robot mandará un JSON con la información del encoder en ese momento.
- url: 192.168.1.103:5000/encoder
- JSON {“encoder1”: , “encoder2”: }

4.5 Visualización

En esta sección, se va a explicar como se realizó el programa de visualización del mapa. Con este programa, se puede ver la disposición del almacén, la posición del robot en todo momento y los pedidos que hay en cada estantería. Para llevar a cabo este programa, se utilizó la librería *pygame* de Python, que permite generar una pantalla de visualización y añadirle figuras customizadas.

Para poder representar de manera exacta el mapa en la pantalla, se dividió la pantalla en cuadriculas, de la misma forma que se divide el mapa en el programa de navegación. Las dimensiones en el mapa están escaladas 1:5 con respecto a las dimensiones reales y para poder trasladar las posiciones de las cosas a la pantalla se generó una función llamada **to_pygame()** a la que se le pasan coordenadas del mapa real y las convierte en coordenadas del mapa de visualización.

El programa de visualización necesita la información del programa de navegación para poder representar tanto los puntos de estantería como el robot, al ser procesos independientes se necesita una comunicación entre procesos para llevar a cabo las comunicaciones. Se valoró la opción de crear otra API, que comunicase ambos procesos, pero tanto pygame como Flask, deben de correr en los hilos principales del programa y por tanto, el programa de visualización no puede exponer un API, a la vez que esta utilizando pygame. Por tanto, se decidió comunicar ambos procesos escribiendo y leyendo de un archivo de texto. Cuando el robot notifica al sistema de navegación que ha terminado de realizar un movimiento, el sistema de navegación escribe las nuevas coordenadas del robot en el archivo de texto, que posteriormente leerá el programa de visualización y actualizará su pantalla.

La pantalla de visualización se puede ver en la Figura 41. La pantalla consta del mapa del almacén con las estanterías marcadas en verde y el punto de recogida marcado en azul; también se puede visualizar la posición del robot dentro del mapa. Además, a la derecha de la pantalla, se muestran una serie de variables correspondiente al robot que son las siguientes:

- **Robot:** esta línea muestra que id lleva el robot que aparece en pantalla.
- **Objetivo:** esta línea muestra cual es la coordenada objetivo del robot en este instante, en este caso al no haber ningún pedido, el objetivo es el mismo punto de inicio donde esta el robot.
- **Posición**: muestra las coordenadas del robot en el mapa.
- **Gestionando pedido**: esto muestra cual es la estantería asociada al pedido que esta gestionando el robot, al igual que en objetivo al no haber ningún pedido en el sistema, se declara el punto de inicio.

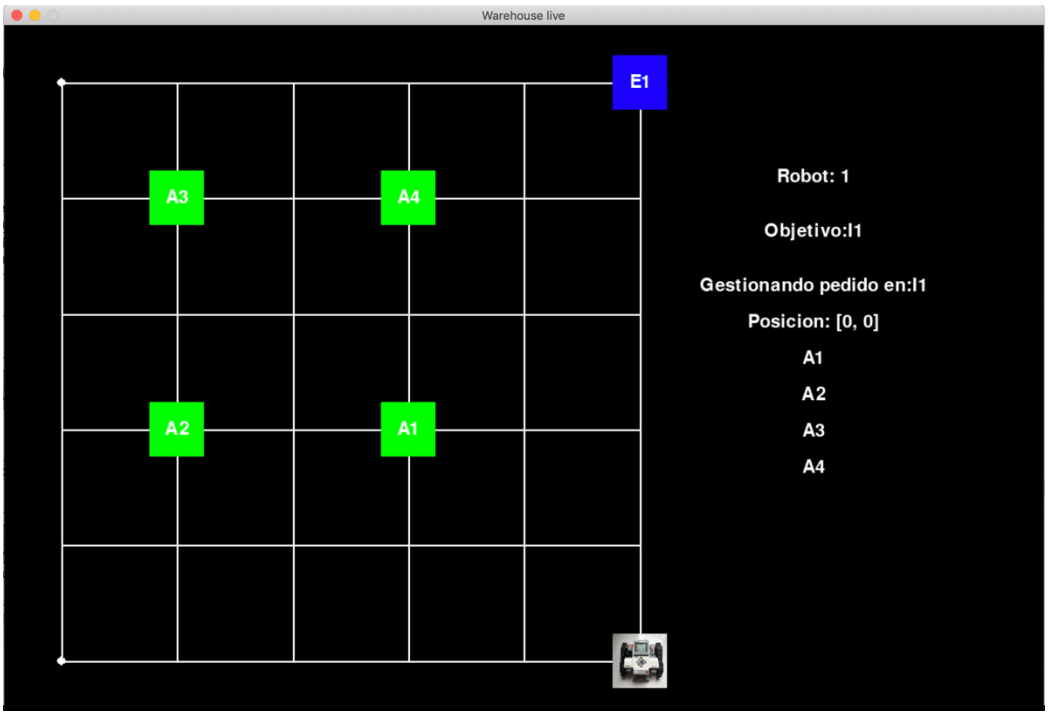


Figura 41: Pantalla de visualización

5 Resultados

En este apartado se va a mostrar el funcionamiento del sistema creado en este proyecto. En concreto, se van a presentar los resultados de precisión de la navegación del robot, la velocidad de gestión de pedidos, la velocidad de las comunicaciones y las rutas calculadas.

5.1 Rutas calculadas por el sistema

Para empezar, se va a mostrar cuales son las rutas finales que el algoritmo calcula para ir de un punto a otro. En concreto se van a mostrar las rutas correspondientes a la gestión de pedidos en las cuatro estanterías, se recuerda que para dar por gestionado un pedido hay que llegar hasta la estantería después ir al punto de recogida y por último volver al punto de la estantería.

Pedido en A1: se va a analiza la ruta que el robot sigue si entra un pedido en la estantería A1, para ello, se ha utilizado el mismo programa de visualización para concatenar las posiciones del robot en una pantalla el resultado se puede ver en Figura 42.

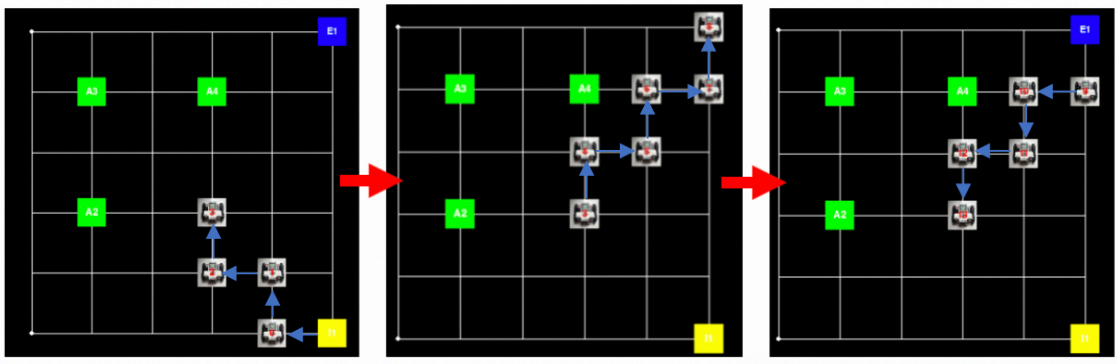


Figura 42: Ruta seguida por el robot para gestionar un pedido en A1

Se puede ver en la que el robot llega a la estantería y al punto de recogida sin atravesar ninguna de las demás estanterías ya que están declaradas como obstáculos.

Pedido en A2: En la Figura 43 se muestra la ruta seguida por el robot para gestionar un pedido en la estantería A2.

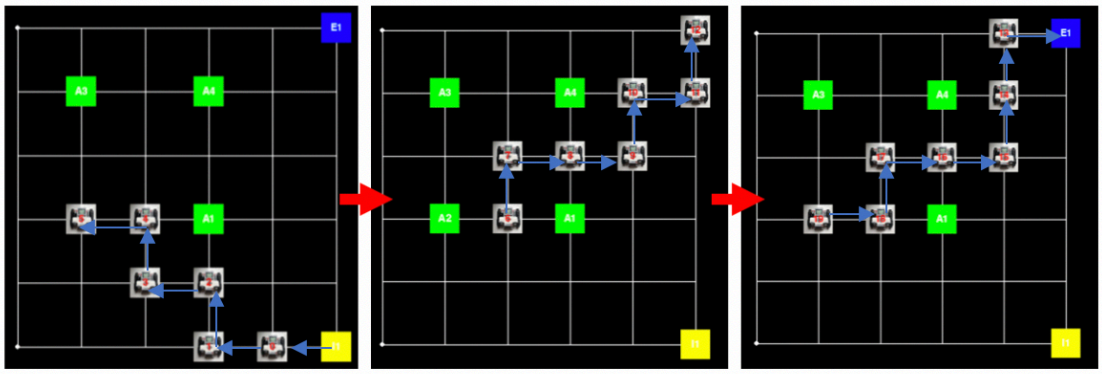


Figura 43: Ruta seguida por el robot para gestionar un pedido en A2

Pedido en A3: En la Figura 44 se muestra la ruta seguida por el robot para gestionar un pedido en la estantería A3.

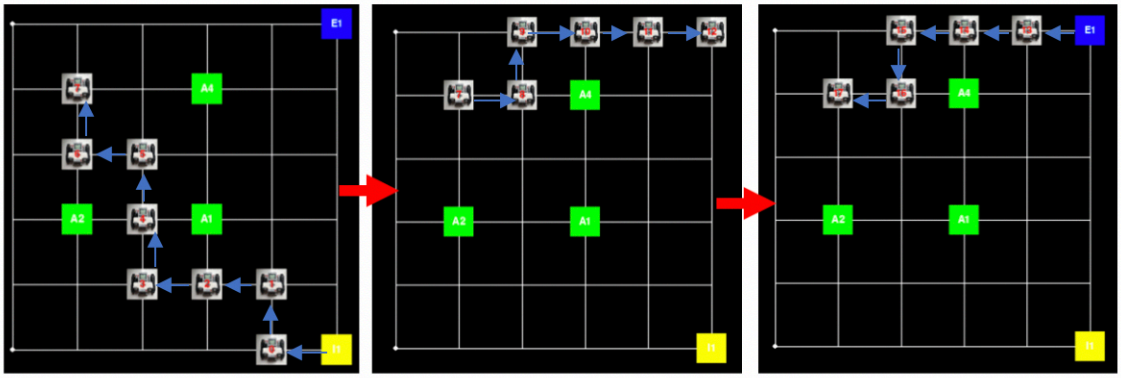


Figura 44: Ruta seguida por el robot para gestionar un pedido en A3

5.2 Precisión del movimiento del robot en el mapa

Para evaluar la precisión de los movimientos del robot, se va a realizar la recogida de un pedido en cada estantería. Para evaluar su precisión se va a ver como de lejos se queda el EV3 del punto objetivo durante la recogida del pedido, en concreto para cada pedido se evaluará la llegada al punto de estantería y al punto de recogida.
Como los errores de precisión, se acumulan durante la navegación del robot, cuantos más pedidos gestiona sin ser posicionado manualmente en el origen, más se podrá observar cuanta precisión tiene, por eso estos cuatro pedidos se gestionarán sin interrumpir nunca el funcionamiento del sistema.

En la Tabla 5 y la Tabla 6 se puede ver los cuatro pedidos gestionados y dos fotos por cada pedido; uno en la estantería y otro en el punto de recogida. El robot llega satisfactoriamente a su objetivo en todas las pruebas, pero se puede ver que en las últimas pruebas la precisión es menor.

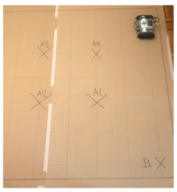
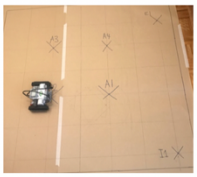
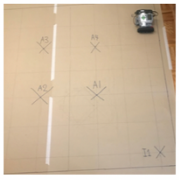
Pedido 1	
A1	E1
	
Pedido 2	
A2	E1
	

Tabla 5: Fotos del robot gestionando pedidos en A1 y A2

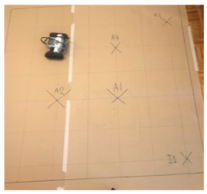
Pedido 3	
A3	E1
	
Pedido 4	
A4	E1
	

Tabla 6: Fotos del robot gestionando pedidos en A3 y A4

5.3 Tiempo en gestionar un pedido

En este apartado se va a evaluar como de rápido gestiona el sistema los pedidos, para ello se realizarán seis pedidos aleatorios y se contará el tiempo que tarda el EV3 en realizar cada uno de ellos y así poder sacar resultados.

Tabla 7: Tiempos en gestionar pedidos

Estantería del pedido	Tiempo tardado (min:sec)
A1	2:20
A1	2:24
A3	2:06
A4	1:50
A2	3:01
A1	2:21

Como se puede observar por los datos, el tiempo tardado en gestionar el pedido depende fuertemente de en que estantería está, por eso se realiza la media de tiempo de todos los pedidos para poder tener un resultado mas fiable la media se puede ver en la Tabla 8.

Tabla 8: Tiempo medio en gestionar un pedido

Media tiempo en gestionar un pedido	**2 minutos 21 segundos**

5.4 Velocidad de las comunicaciones

Para evaluar la velocidad de las comunicaciones, se va a utilizar un programa llamado *iperf*, que permite conocer el ancho de banda existente en las comunicaciones entre el robot y el ordenador. Se va a evaluar el ancho de banda en ambos sentidos, tanto desde el robot al ordenador como del ordenador al robot. Los resultados obtenidos se pueden ver en la Tabla 9.

Tabla 9: Tabla que muestra la velocidad de las comunicaciones

Sentido de las comunicaciones	**Ancho de banda**
Ordenador -> Robot	2.54 Mbit/sec
Robot -> Ordenador	1.96Mbit/sec

6 Conclusiones

En este apartado se van a mostrar las conclusiones sacadas después de haber realizado el proyecto y se plantearán algunas mejoras y desarrollos adicionales que serían beneficiosas para su funcionamiento.

En la sección 4.5 de la memoria, se explica el algoritmo de rutas y sus características, tras la finalización del proyecto se puede concluir que este algoritmo calcula exitosamente las rutas a seguir y estas rutas siempre llegan a su objetivo; además, se consiguen evitar las estanterías declaradas como obstáculos. A la hora de implementar este sistema en la realidad, se le podrían añadir funcionalidades muy interesantes gracias a que el algoritmo permite declarar obstáculos que el robot evitará; como evitar choques al usar mas de un robot, declarándolos obstáculos móviles o definir una ruta entera como obstáculo, para permitir que los operarios entren en momentos puntuales al entorno de los robots.

La precisión de los movimientos del robot se ha conseguido gracias a una calibración exhaustiva de sus movimientos, como se explica en la sección 4.4 . Al finalizar el proyecto se puede concluir que el robot navega con bastante precisión como se puede observar en la sección de resultados 5.2 . El problema reside en que los errores de precisión en los movimientos del robot se acumulan a lo largo de su funcionamiento, por eso sería recomendable generar algún tipo de sistema, que cada cierto tiempo permita recalibrar al robot, y así evitar que el robot se desvié de su trazado. Por ejemplo, como el robot siempre pasa por el punto de recogida, se podría tener un puesto de calibración allí y así prevenir grandes desajustes en la posición del robot.

Por último, tal y como se planteaba en la sección de objetivos 1.2, se ha conseguido migrar el software de control y navegación fuera del robot; consiguiendo así que la capacidad de computación limitada del Lego EV3 no suponga un problema a la hora de querer implantar funcionalidades que requieran muchos recursos computacionales, como puede ser el algoritmo que calcula las rutas. Si a este aumento de recursos de computación, se le incorporan unas comunicaciones mucho más rápidas para el envío de información entre las partes como las que se prometen en un futuro muy cercano con el 5G, o la flexibilidad y robustez que las plataformas de computación en la nube y en el borde de la red pueden ofrecer, se pueden conseguir sistemas de robots industriales que requieran menos recursos dentro de los mismos, y por tanto puedan ser mas baratos y con menos predisposición a la obsolescencia.

Bibliografía

[1] ASTI, «ASTI,» 24 Marzo 2019. [En línea]. Available: https://asti.es/es/noticias/agvs-inteligentes-y-conectados-la-novedad-del-sector-que-asti-mobile-robotics-presenta-en-hannover-messe-2019-para-la-industria. [Último acceso: 1 Junio 2019].

[2] Daifuku, «Daifuku,» Agosto 2003. [En línea]. Available: https://www.daifuku.com/mx/solution/technology/automatedwarehouse/. [Último acceso: Junio 2019].

[3] F. Cedo, «El 5G en la industria: robótica en tiempo real,» 2019. [En línea]. Available: https://www.lavanguardia.com/economia/20190225/46664294477/mwc-2019-5g-robotica-industria.html. [Último acceso: 20 Mayo 2019].

[4] A. Robotics, «Acutronic Robotics,» 30 septiembre 2018. [En línea]. Available: https://acutronicrobotics.com/news/modularity-for-unifying-robotics/. [Último acceso: Junio 2019].

[5] A. Cabirta, «El comercio electronico gana el terreno al tradicional en España: supone el 20% del gasto,» 22 Marzo 2019. [En línea]. Available: https://www.bbva.com/es/el-comercio-electronico-gana-terreno-al-tradicional-en-espana-supone-el-20-del-gasto/. [Último acceso: 23 Mayo 2019].

[6] A. a. v. a. convencional, «Mecalux,» 31 Agosto 2018. [En línea]. Available: https://www.mecalux.es/blog/almacen-automatico-convencional. [Último acceso: mayo 2019].

[7] ¿. e. e. S. d. G. d. Almacenes?, «ASoftware,» 16 Septiembre 2019. [En línea]. Available: https://www.assoftware.es/que-es-el-software-de-gestion-de-almacenes/. [Último acceso: 1 Junio 2019].

[8] ¿. e. u. SGA?, «Mecalux,» [En línea]. Available: https://www.mecalux.es/manual-almacen/almacen/que-es-un-sga. [Último acceso: 2 Junio 2019].

[9] M. Wulfraat, «canadiangrocer,» 10 Mayo 2013. [En línea]. Available: http://www.canadiangrocer.com/blog/goods-to-person-automation-technology-presents-opportunities-for-grocery-distributors-25896. [Último acceso: 23 Marzo 2019].

[10] S. A. A. I.-d. r. o. a. s. c. p. t. f. d. centres, «MWPVL,» Abril 2012. [En línea]. Available: http://www.mwpvl.com/html/swisslog_autostore_review.html. [Último acceso: 24 Marzo 2019].

[11] A. t. A. M. o. Robotics, «The new york times,» 19 Marzo 2012. [En línea]. Available: https://dealbook.nytimes.com/2012/03/19/amazon-com-buys-kiva-systems-for-775-million/. [Último acceso: 15 Marzo 2019].

[12] J.Sankari y R Imtiaz, «Automated guided vehicle (AGV) for industrial sector,» *IEEE, International Conference on Intelligent Systems and Control (ISCO),* 2016.

[13] S. Monzon, «ATRIA INNOVATION,» 13 Noviembre 2018. [En línea]. Available: http://atriainnovation.com/agvs-vehiculos-de-guiado-automatico-todavia-no-sabes-lo-que-son/. [Último acceso: 3 Junio 2019].

[14] L. Mcconney, «The future of AGV´s: New Technology to keep an eye on,» 5 Diciembre 2016. [En línea]. Available: https://www.conveyco.com/future-agvs-new-technology-keep-eye/. [Último acceso: 2 Junio 2019].

[15] Bluetooth, «Wikipedia,» [En línea]. Available: https://es.wikipedia.org/wiki/Bluetooth. [Último acceso: 1 Junio 2019].

[16] Wifi, «Wikipedia,» [En línea]. Available: https://es.wikipedia.org/wiki/Wifi. [Último acceso: 25 Mayo 2019].

[17] S. v. S. y. REST, «Adictos al trabajo,» 10 Enero 2014. [En línea]. Available: https://www.adictosaltrabajo.com/2014/01/10/soavs-soap-rest/. [Último acceso: 23 Mayo 2019].

[18] S. v. R. v. J. c. [2019], «RAYGUN,» 2018. [En línea]. Available: https://raygun.com/blog/soap-vs-rest-vs-json/. [Último acceso: 1 Junio 2019].

[19] «IBM,» [En línea]. Available: https://www.ibm.com/es-es/cloud/learn/what-is-cloud-computing. [Último acceso: 28 Mayo 2019].

[20] W. i. e. computing, «GE digital,» [En línea]. Available: https://www.ge.com/digital/blog/what-edge-computing. [Último acceso: 1 Junio 2019].

[21] p. q. e. '. c. e. h. t. a. l. nube, «Innovadores by Inndux,» 5 Abril 2019. [En línea]. Available: https://innovadores.larazon.es/es/not/por-que-el-edge-computing-esta-haciendo-temblar-a-la-nube. [Último acceso: 23 Mayo 2019].

[22] M. A. B. B. C. K. M. M. O. R. R. S. Sabattini Lorenzo, «The PAN-Robots Project: Advanced Automated Guided Vehicle Systems for Industrial Logistics,» *IEEE Robotics & Automation Magazine ,* vol. 25, nº 1, pp. 55-64, 2017.

[23] T. Akimoto y N. Hagita, «Introduction to a Network Robot System,» *IEEE International Symposium on Intelligent Signal Processing and Communications,* 2006.

[24] N. G. N. G. K. H. A. R. P. R. M. J. Seyed Ali, «Cloud robotics: A software architecture: For heterogeneous large-scale autonomous robots,» *IEEE World Automation Congress (WAC),* 2016.

[25] I. A. T. D. T. C. o. C. N. -. T. U. Dresden, H. Wu, G. T. Nguyen, H. Salah y F. H. Fitzek, « Mobile Edge Cloud for Robot Control Services in Industry Automation,» *IEEE Annual Consumer Communications & Networking Conference (CCNC),* 2019.

[26] N. R. S. R. G. S. Hubertus Munz, «What will 5G bring to industrial robtics,» 2018. [En línea]. Available: https://www.ericsson.com/en/blog/2018/12/what-will-5g-bring-to-industrial-robotics. [Último acceso: Mayo 2019].

[27] A. s. algorithm, «Wikipedia,» [En línea]. Available: https://en.wikipedia.org/wiki/A*_search_algorithm. [Último acceso: 22 Mayo 2019].

[28] J. Penalva, «LEGO Mindstorms y WeDo a prueba: así se enfrenta LEGO a la época dorada de la robótica y programación,» *Xtaka,* 2018.

[29] «WIkipedia,» [En línea]. Available: https://es.wikipedia.org/wiki/Lego_Mindstorms.

[30] A. I. t. R. O. S. T. U. R. A. Framework, «Developers,» [En línea]. Available: https://www.toptal.com/robotics/introduction-to-robot-operating-system. [Último acceso: 1 Junio 2019].

[31] W. l. C++, «codementor,» [En línea]. Available: http://www.bestprogramminglanguagefor.me/why-learn-c-plus-plus. [Último acceso: 21 Mayo 2019].

[32] P. v. Java, «Activestate,» [En línea]. Available: https://www.activestate.com/blog/python-vs-java-duck-typing-parsing-whitespace-and-other-cool-differences/. [Último acceso: 21 Mayo 2019].

[33] Wikipedia, «Wikipedia,» [En línea]. Available: https://es.wikipedia.org/wiki/Git. [Último acceso: 1 Junio 2019].

[34] «EcuRed,» [En línea]. Available: https://www.ecured.cu/IDE_de_Programaci%C3%B3n. [Último acceso: 2 Junio 2019].

[35] W. i. P. IDE, «Midnfire solutions,» [En línea]. Available: https://medium.com/@mindfiresolutions.usa/what-is-pycharm-ide-cc0735784f64. [Último acceso: 1 Junio 2019].

[36] L. s. m. e. p. g. t. almacén, «EL Boletin,» 2017. [En línea]. Available: https://www.elboletin.com/noticia/149320/hoy-en-la-red/los-softwares-mas-eficientes-para-gestionar-tu-almacen.html. [Último acceso: Marzo 2019].

[37] ¿. f. u. a. automatico?, «Mecalux,» 2019. [En línea]. Available: https://www.mecalux.es/blog/como-funciona-almacen-automatico. [Último acceso: Mayo 2019].

[38] F. Carlotti, «Interempresas,» [En línea]. Available: https://www.interempresas.net/Logistica/Articulos/240924-La-Industria-40-facilita-la-automatizacion-a-los-pequenos-almacenes.html.