



Escuela Politécnica Superior 

TRABAJO FIN DE GRADO
GRADO EN INGENIERIA ELECTRÓNICA INDUSTRIAL

**PLATAFORMA RASPBERRY PI PARA TELEOPERACIÓN
DE ROBOT MÓVIL CON FUNCIONES VISUALES
AVANZADAS**

ALUMNO: JOSE JAVIER MARTINEZ SANCHEZ DE LA NIETA

TUTOR: FERNANDO DIAZ DEL RIO

SEPTIEMBRE 2022

Trabajo Fin de Grado
Ingeniería Electrónica Industrial

Plataforma Raspberry Pi para Teleoperación de Robot Móvil con Funciones Visuales Avanzadas

Autor:

Jose Javier Martínez Sánchez de la Nieta

Tutor:

Fernando Díaz del Río

Dpto. de Arquitectura y Tecnología de Computadores

Escuela Politécnica Superior

Universidad de Sevilla

Sevilla, 2022

AGRADECIMIENTOS

Estoy infinitamente agradecido con las personas que me han ayudado a que todo esto sea posible.

Gracias a mi familia, por todos los ánimos durante todos estos años.

Gracias a mis amigos, dentro y fuera de la carrera universitaria, que estoy seguro que sin ellos no habría llegado tan lejos.

Gracias a mi tutor Fernando, por todos los consejos, ayudas y clases chulas de robótica.

Y especialmente, gracias a Paola, por todo su apoyo y confianza.

RESUMEN

Este proyecto consiste en el desarrollo de un prototipo de un robot móvil con funciones visuales avanzadas.

El ensamblaje del robot desde cero, junto con el aprendizaje de Python como lenguaje de programación y una Raspberry Pi, han resultado en el desarrollo de este proyecto.

Mediante el prototipo se simula un proceso de automatización robótica, que consiste en una serie de tareas asignadas. Las tareas se determinan automáticamente gracias a detecciones a través de sensores y la cámara incorporada en el robot.

Palabras Clave: Raspberry Pi, IoT (Internet de las cosas), OpenCV, Python

ABSTRACT

This project consists of the development of a prototype of a mobile robot with advanced visual functions.

The assembly of the robot from scratch, together with the learning of Python as a programming language and a Raspberry Pi, have resulted in the development of this project.

Through the prototype, a robotic automation process is simulated, which consists of a series of indicated tasks. The tasks are determined automatically thanks to detections through sensors and the robot's built-in camera.

Keywords: Raspberry Pi, IoT (Internet of Things), OpenCV, Python

ÍNDICE

1	INTRODUCCIÓN	14
2	ROBOTS MÓVILES EN LA INDUSTRIA.....	16
2.1	DESARROLLO DE ROBOTS MÓVILES.....	16
2.2	TIPOS DE FUNCIONAMIENTO.....	16
2.2.1	Robot SCARA.....	17
2.2.2	Robot Paralelo.....	17
2.2.3	Robot Antropomórfico.....	18
2.2.4	Robot Cartesiano.....	19
2.3	INTERACCIONES ROBOT-HUMANO Y APLICACIONES ACTUALES 20	
2.4	SEGURIDAD EN LOS ROBOTS.....	22
2.5	PROGRAMACIÓN DE ROBOTS	25
2.5.1	Programación de nivel cero	25
2.5.2	Programación de robots de aprendizaje	25
2.5.3	Programación de robots de tercera generación.....	27
2.5.4	Programación de robots de cuarta generación:.....	28
2.5.5	Programación de robots de quinta generación	29
3	NAVEGACIÓN Y LOCALIZACIÓN DE ROBOTS MÓVILES	30
3.1	TELEOPERACIÓN DE ROBOTS.....	30
3.2	INTELIGENCIA Y VISIÓN ARTIFICIAL.....	30
3.3	GUIADO AUTÓNOMO EN APLICACIONES INDUSTRIALES.....	32
4	PLATAFORMA RASPBERRY PI 4 MODEL B.....	35
4.1	DESCRIPCIÓN DE LA PLATAFORMA.....	35
4.2	INSTALACIÓN Y ADAPTACIÓN DEL ENTORNO.....	37
5	PROCESAMIENTO DE INFORMACIÓN DE LOS ROBOTS	37

	11
5.1 CÁMARA RASPBERRY PI	38
5.2 BIBLIOTECA OPEN CV	39
5.3 DETECCIÓN DE OBJETOS, COLORES Y CÓDIGOS.....	40
5.4 GENERACIÓN DE OBJETO SOBRE DETECCIÓN DE FORMA (ArUco)	40
6 DESARROLLO DEL PROYECTO	42
6.1 INTRODUCCIÓN ROBOT RASPTANK	42
6.2 MECÁNICA Y HARDWARE	42
6.2.1 Raspberry Pi 4 Model B.....	46
6.2.2 Motor HAT	47
6.2.3 Cámara Raspberry	47
6.2.4 Iluminación LED.....	48
6.2.5 Tarjeta Micro SD.....	48
6.2.6 Baterías	49
6.2.7 Sensor de línea.....	49
6.2.8 Sensor de ultrasonidos	50
6.2.9 Cableado y conectores	51
6.3 CARACTERÍSTICAS Y ADAPTACIÓN SOFTWARE.....	52
6.3.1 Configuración del sistema operativo.....	52
6.3.2 Configuración del protocolo SSH.....	52
6.3.3 Configuración de la conexión Wifi.....	53
6.3.4 Instalación PuTTY	53
6.3.5 Lenguaje Python.....	54
6.3.6 MobaXterm	54
6.4 LIBRERÍAS Y HERRAMIENTAS UTILIZADAS EN LA PROGRAMACIÓN DEL PROYECTO.....	56
6.4.1 OpenCV.....	56
6.4.2 Numpy	56

	12
6.4.3	Thonny Python..... 56
6.4.4	PyCharm..... 57
6.4.5	Pyzbar 57
6.5	PROCESAMIENTO Y GUIADO AUTÓNOMO (AGV) 58
6.5.1	Diseño y programación de balizas y rutas 58
6.5.2	Sensores de guiado superficial..... 59
6.5.3	Sensores de distancia mediante ultrasonidos 60
6.5.4	Obstáculos y fallos de recorrido..... 60
6.6	RESULTADOS EXPERIMENTALES..... 60
6.6.1	Detección de obstáculos mediante ultrasonidos..... 60
6.6.2	Interpretación de Códigos QR a través de la cámara de la Raspberry 61
6.6.3	Seguimiento de línea en la superficie del suelo..... 61
6.7	ANÁLISIS Y VALORACIÓN DEL PROYECTO 63
6.7.1	Funcionalidad y posibles aplicaciones..... 63
6.7.2	Impacto medioambiental..... 65
6.7.3	Eficiencia energética..... 65
7	ECONOMÍA Y VIABILIDAD DEL PROYECTO 66
7.1	Presupuesto de recursos humanos..... 66
7.2	Presupuesto de materiales..... 67
7.3	Presupuesto final del proyecto 67
8	CONCLUSIONES 69
9	LINEAS FUTURAS..... 71
10	ANEXOS..... 74
11	REFERENCIAS BIBLIOGRAFICAS..... 81

1 INTRODUCCIÓN

Hablando de los robots se puede evidenciar que son un gran avance tecnológico en la humanidad y han venido para quedarse e impulsar el sistema de funcionamiento, producción y desarrollo industrial global actual.

Estas máquinas presentan unos factores importantes por los cuales han ido emergiendo principalmente en la evolución de los tiempos.

El humano siempre ha querido construir máquinas para facilitar el trabajo. Desde el año 400 a.C. se encuentran documentos que describen objetos autónomos, como la paloma de Arquitas de Tarento, una paloma de madera que funcionaba con vapor de agua, o el gallo de Estrasburgo en el año 1350, el cual movía las alas y el pico cuando se marcan las horas del reloj, que actualmente se cataloga como el autómatas más antiguo que existe.

En los años ochenta explotó la era robótica, aumentándose las ventas de robots muy significativamente, lo que dio lugar al nacimiento de la robótica inteligente y al uso de la denominada inteligencia artificial (IA).

Más tarde, en 1958 Joseph Engelberger y Devol empezaron a trabajar juntos en el desarrollo de las máquinas para el ámbito industrial. Ellos fueron los creadores del famoso robot Unimate, (Figura 1.1), un robot empleado para carga y descarga de material en una máquina de fundición por inyección que era controlado por interruptores, motores hidráulicos y finales de carrera. [15]



Figura 1.1: Robot UNIMATE.

Cabe mencionar el IRB6, que fue el primer robot de accionamiento totalmente eléctrico, construido en Suecia en 1973, y el primer lenguaje de programación (WAVE) desarrollado ese mismo año.

De tal modo, en las últimas décadas, la evolución de las máquinas robóticas ha ido avanzando a pasos agigantados, y lo sigue haciendo. La búsqueda de la productividad y eficiencia, y el aumento de demanda de bienes de consumo dan lugar a la utilización de la robótica inteligente en las empresas para una producción de menor coste, obteniendo una mayor cantidad de beneficios finales.

Sin olvidar los problemas que estas máquinas pueden llegar a solucionar de la vida cotidiana, facilitando así las vidas de los seres humanos.

La robótica industrial evoluciona a la robótica inteligente gracias a tecnologías que se tratan en este proyecto fin de grado como son el IoT (Internet of Things), el Big Data y los sistemas de inteligencia y visión artificial.

Algunos ejemplos más actuales se citan más adelante en el apartado (2.3) del documento.

2 . ROBOTS MÓVILES EN LA INDUSTRIA

2.1 DESARROLLO DE ROBOTS MÓVILES

La noción de robótica atiende a un concepto de estructura mecánica universal capaz de adaptarse a situaciones muy variadas (dentro de sus capacidades), en las que se requieren en mayor o menor medida las características de movilidad, autonomía, gobernabilidad, y polivalencia.

La robótica en general abarca una amplia gama de máquinas o dispositivos con muy diversas cualidades físicas y funcionales, cada una de ellas asociada a una estructura mecánica y campo de aplicación particular [21]. La robótica se nutre en gran medida de los progresos de la informática, la electrónica y la inteligencia artificial, teniendo en cuenta intrínsecamente los avances en sensores, controladores, servomecanismos o equipos de comunicación, entre otros.

Los investigadores actuales buscan la construcción de máquinas móviles capaces de trabajar en medios irregulares y desordenados que sean capaces de responder eficazmente en situaciones imprevistas y tomar sus propias decisiones a través de sensores en tiempo real.

2.2 TIPOS DE FUNCIONAMIENTO

Dentro de los tipos de robots se puede encontrar una diversidad enorme según su funcionamiento.

Se destacan entre algunos de los robots manipuladores:

2.2.1 Robot SCARA

Los robots SCARA [16] surgieron en Japón por la demanda de realización de tareas de bajos periodos de ciclo de trabajo, estas máquinas sacrifican la movilidad de su primer eje (muñeca) para una mayor velocidad en el transcurso de su trabajo. Son la opción por excelencia para trabajos de tipo Pick & Place (Figura 2.2)



Figura 2.2: Robot SCARA.

2.2.2 Robot Paralelo

Otros robots utilizados para Pick & Place son los robots paralelos (Figura 2.3). Estos robots están formados por dos bases, una fija y otra móvil unidas por cadenas cinemáticas. Se utilizan en líneas de producción en las que se necesitan movimientos muy rápidos y precisos. Se caracterizan por su buena relación carga/potencia [19].



Figura 2.3: Robot paralelo de Adept.

2.2.3 Robot Antropomórfico

Los robots antropomórficos [17] son los robots manipuladores industriales por excelencia, suelen tener un mayor campo de libertad de movimiento con respecto a los anteriores mencionados.

Esta configuración de robots se utiliza para mover grandes cargas con una repetibilidad alta. Se asemejan al movimiento de un brazo humano. (Figura 2.4)



Figura 2.4: Robot antropomórfico.

2.2.4 Robot Cartesiano

Los robots cartesianos [18] son aquellos robots que trabajan exclusivamente de forma lineal.

Normalmente tienen 3 actuadores que les permiten moverse en los ejes X, Y y Z. Se emplean mayormente en tareas de paletización de cargas considerables o en líneas de envasado. (Figura 2.5)

Las máquinas más comunes de este tipo son las de CNC



Figura 2.5: Robot cartesiano.

Los robots mostrados en las figuras anteriores no suelen tener tanta libertad de movimiento, ya que están diseñados para permanecer fijados a superficies o soportes mediante uniones mecánicas para una mejor eficiencia y estabilidad.

Por otro lado, los robots móviles pueden desempeñar estas mismas funciones, con la ventaja de poder moverse en el espacio simultáneamente. Los robots pueden presentar varios sistemas de movimientos como son:

- Robots con ruedas, utilizados para el transporte de personas o mercancías. Estos robots tienen la ventaja de ser versátiles para poder ser usados para diferentes propósitos gracias a la movilidad que les otorgan las ruedas. Su desventaja se presenta en la movilidad a través de superficies irregulares.

- Robots zoomórficos (con patas): Se adaptan perfectamente a las superficies irregulares ya que están basados en los animales terrestres, aunque su movilidad es limitada.
- Robots con orugas o cadenas: Son de robots de gran utilidad para explorar por superficies irregulares y de difícil acceso, es el tipo de robot utilizado en este proyecto. Son muy eficientes en movimientos de línea recta, pero son imprecisos en los giros.
- Otros robots: Se encuentran entre ellos los robots marinos, submarinos, manipuladores móviles, aéreos, humanoides o Transformers entre más variedad.

2.3 INTERACCIONES ROBOT-HUMANO Y APLICACIONES ACTUALES

Cada vez es más frecuente que los humanos y los robots compartan un espacio de trabajo. Esto ha desencadenado la necesidad de mejorar la comunicación entre humanos y robots, y de que el robot sea consciente de lo que puede esperar de las personas que le rodean y, del mismo modo, que las personas sepan lo que se puede esperar de los robots.

Un aspecto de la interacción con los robots que no es exclusivo de los robots móviles, es enseñarles las tareas que se espera que realicen. Se describe un método de enseñanza por demostración, en el que los componentes primitivos de los movimientos son aprendidos por un robot a través de la teleoperación [21]. El método es capaz de extrapolar desde un conjunto de movimientos básicos hasta el desarrollo de una tarea completa sin que el usuario tenga que demostrar todos los aspectos de la tarea.

Otro enfoque consiste en realizar gestos para mostrar al robot móvil lo que debe manipular o a dónde debe ir [22]. Esto requiere la definición de gestos que sean fácilmente comunicados por los humanos y fácilmente reconocidos por los sensores del robot móvil.

También se ha estudiado la forma en que un robot puede pedir ayuda. [23] Se describe cómo un robot móvil que puede desplazarse por un entorno de oficina, no tiene ningún manipulador, por lo que, por ejemplo, no puede pulsar el botón del ascensor. El robot dispone de algoritmos que le permiten encontrar a las personas y pedirles ayuda, teniendo en cuenta la imposición que supone para las personas a las que pide ayuda (la distancia de desplazamiento hasta el lugar de la ayuda) y la propia necesidad del robot de que el tiempo de realización de la tarea sea corto.

Otra cuestión a tener en cuenta cuando hay personas en el entorno es abordada por Sisbot [24]. Aquí se desarrolla un planificador que computa trayectorias que tienen en cuenta la comodidad y las expectativas de las personas que pueden estar cerca del robot. El plan asegura que el robot mantiene una distancia segura con todas las personas y trata de mantener el robot en el campo de visión de las personas para evitar apariciones sorpresivas.

Los robots de cuidado personal se han convertido en avanzados sistemas interactivos humano-robot. Por ejemplo, Care-O-Bot [25], (ver figura 2.6), asistente de robot móvil, se encuentra ahora en su tercera generación con características que son potencialmente muy útiles para la comunidad de robots móviles industriales. La navegación se realiza a través de la odometría (mediciones del movimiento del vehículo) mejorada por la localización y el mapeo simultáneos (SLAM) basados en datos de escaneo láser delantero y trasero que se comparan con un mapa global. Un controlador jerárquico de tres niveles incluye el control de una rueda, el control de las cuatro ruedas y un planificador de trayectorias para permitir la planificación de trayectorias alrededor de obstáculos y a través de pasillos estrechos. El manipulador móvil omnidireccional incluye una bandeja y un brazo robótico y puede calcular trayectorias de manipulación sin colisiones a partir de los datos de una cámara en color y de sensores de detección y alcance de luz (LIDAR). El sistema también implementa la segmentación espacial para el aprendizaje de obstáculos y la interpretación de la nube tridimensional de puntos detectados por los sensores LIDAR para el reconocimiento de objetos.



Figura 2.6: Care-O-Bot.

Se trata de una de las tecnologías más avanzadas dentro del campo de los robots móviles, la cual sirve como inspiración para la realización de este proyecto.

2.4 SEGURIDAD EN LOS ROBOTS

Los robots, al ser unas máquinas controladas por software, deberían estar bien protegidos contra diversos ciberataques, y los desarrolladores de los sistemas robóticos deberían conocer las vulnerabilidades de los marcos de programación y componentes de los robots. Sin embargo, la mayoría de los proyectos implementan sus propias conexiones con la nube (a veces utilizando texto plano

ya que no es el ámbito principal de estos trabajos, como, por ejemplo, este TFG, estudiar la ciberseguridad del proyecto.)

Como los robots están conectados a internet y dependen de sensores y actuadores, se consideran sistemas ciber físicos (CPS) [26].

El concepto de SPI se trata de una red específicamente diseñada de componentes que colaboran entre sí y que sirven para supervisar y controlar el mundo físico. Entre los tipos de componentes se encuentran los físicos (sensores y actuadores) y los cibernéticos (computación y comunicación).

Estos componentes están integrados y su funcionamiento se gestiona y supervisa. [27]. Se espera que los CPS, cuando se conecten a internet, puedan revolucionar muchos ámbitos, como la sanidad, el transporte, la agricultura, la fabricación, la ingeniería civil, militar y espacial.

Hasta la fecha se han identificado y categorizado muchos ataques a los CPS [27],[28]. Algunos criterios utilizados para la clasificación de los ciberataques son: vectores de ataque (medio o ruta por el que el atacante accede a un sistema o red), impactos en el sistema y objeto del ataque. En la figura 2.7 se pueden apreciar casos específicos de cada uno de esos tipos.

Los ejemplos de las consecuencias de un ciberataque a la SPI son los siguientes:

- Daños físicos a un dispositivo CPS (hardware) y/o a objetos de su entorno.
- Pérdidas económicas o de imagen de corporativa.
- Lesiones o incluso la muerte de personas.

En [29] se analizan más amenazas y posibles consecuencias de los ciberataques a los SPI. Los robots suelen estar equipados con multitud de sensores, como por ejemplo cámaras RGB, micrófonos, unidades de medición inercial, escáneres láser, emisores IR y pueden desplazarse, por lo tanto, este tipo de SPI debe considerarse especialmente vulnerable a una amplia gama de categorías de riesgo derivadas de las ciber amenazas.

Dado que los fallos de seguridad no conocen límites y que el ámbito de la robótica es vulnerable a tantas inseguridades, es necesario prevenir estos ataques antes de que afecten a las máquinas. Para ello se deben identificar los

ataques para poder mitigar las amenazas y así asegurar el sistema. En esta sección se comentan algunos de ellos:

- **Robustez de la comunicación:** La transmisión de órdenes de un controlador al robot y la retroalimentación del robot al controlador requieren un medio de transmisión. Este medio de transmisión es vulnerable a la mayoría de los ataques. Garantizar una capa de seguridad en los canales de transmisión de la información reducirá sin duda las probables inseguridades. Dado que la comunicación es dedicada, el cifrado y la introducción de mecanismos de autenticación en los datos transmitidos restringirán los ataques de modificación, manipulación y apropiación del funcionamiento del robot.
- **Servicio de distribución de datos en ROS:** en los sistemas operativos para robots (ROS), los mensajes pueden transmitirse sin cifrado, lo que favorece las escuchas. Sin embargo, la integración del Servicio de Distribución de Datos (DDS) como capa de transporte conllevará la instalación de plugins que garanticen la autenticación, el control de acceso y la criptografía.
- **Seguridad de la nube:** La robótica en la nube es un campo que emana de la robótica integrada en el entorno de la computación en la nube. Se basa en el almacenamiento en la nube y en otras tecnologías de Internet de la infraestructura de la nube. Permite mejorar la memoria, la potencia de cálculo y la interconectividad de las aplicaciones robóticas. Los datos son recogidos por los sensores, y la información correspondiente se carga en el centro de computación remoto. La información se procesa y puede compartirse con otros robots.
- **Buses de comunicación:** La comunicación segura también puede ser proporcionada por los buses de comunicación. A diferencia de los buses tradicionales, los buses de comunicación se basan en Ethernet y, por lo tanto, pueden hacer uso de las características pertenecientes a TCP/UDP/IP.

- Creación de una célula robotizada cerrada: para la protección mecánica o de hardware de los robots es conveniente programar el funcionamiento de las máquinas dentro de un perímetro protegido físicamente para evitar daños que puedan producir estas al exterior.
- Seta de emergencia: como todas las máquinas automáticas, es necesario que tenga un botón de parada de emergencia, el cual puede prevenir también muchos daños físicos.

2.5 PROGRAMACIÓN DE ROBOTS

La programación orientada a los robots es la disciplina que se basa en el desarrollo de aplicaciones, herramientas y software para estas máquinas [31]. Al pertenecer al campo de las nuevas tecnologías, tiene una evolución vertiginosa desde los últimos años hasta ahora.

Se puede dividir la programación de robots en varios tipos:

2.5.1 Programación de nivel cero

La programación a nivel cero, también conocida como programación a nivel de hardware, se utiliza comúnmente en robots donde no se requiere mucha flexibilidad en el funcionamiento. Esta generación se basa en robots manipuladores básicos. El tipo de programación es típica de un controlador de tambor que, a medida que gira, abre válvulas que hacen que el manipulador del robot se mueva. Las ventajas de este tipo de programación es que es una programación simple y fiable, mientras que la desventaja es que es poco versátil.

2.5.2 Programación de robots de aprendizaje

El método de programación más utilizado en la actualidad es el de "enseñar y reproducir", también conocido como "guiar". Es el método de programación más utilizado en los robots industriales actuales.

Este tipo de secuenciación consta de dos métodos: la programación por puntos y la programación directa:

- Programación por puntos

Un programa de robot es una trayectoria en el espacio compuesta por múltiples destinos llamados puntos. En cada punto del programa, un operador o línea de código, indica si el robot debe detenerse o pasar al siguiente el punto.

Además, el operario tiene la capacidad de ordenar al robot que espere una señal en cualquier punto o que genere una señal en dicho punto.

En la programación por puntos, el operario mueve físicamente el robot a través de la trayectoria en el espacio. Mientras el robot se mueve, el controlador envía señales de retroalimentación de los sensores y actuadores en cada uno de los puntos del robot a lo largo de la trayectoria para registrar el programa del robot.

Una de las ventajas de la programación por puntos es que puede ser fácilmente programada para estas tareas por una persona que tenga pocos conocimientos de informática. Las desventajas de este método radican en que requiere muchos puntos para seguir una programación por pasos, por lo que es difícil de editar y requiere una gran cantidad de memoria.

- Programación directa:

En la programación directa (que es más común que la programación de por puntos, el operario posiciona el robot utilizando botones o interruptores en una consola de aprendizaje, e indica cuando el robot ha alcanzado un punto que debe introducirse y almacenarse en el programa. El sistema operativo convierte la información introducida en coordenadas del robot, o coordenadas cartesianas que describen esa posición en el espacio. Las coordenadas del programa se almacenan en la memoria y requieren mucha menos memoria que un programa por pasos para la misma tarea.

La consola de aprendizaje incluye teclas que permiten la ejecución de una secuencia de movimiento en una sola pulsación y permite su edición para poder insertar, borrar o cambiar puntos de la trayectoria en el programa. [32]

Las ventajas de este método de programación son que la programación es fácil, se domina rápidamente y es especialmente adecuada para paletizado y otras tareas de manipulación de materiales. Sus desventajas son que cada proveedor tiene sus propias interfaces hombre-máquina (HMI) y software, por lo que es difícil crear una capacidad de programación de robots generalizada,

2.5.3 Programación de robots de tercera generación

La tercera generación de lenguajes formales para la programación de robots puede considerarse de los primeros sistemas de programación de robots creados de alto nivel, como BASIC [33], adaptados para robots con sensores. BASIC tiene comandos que le permite trazar líneas rectas y en algunos casos círculos o curvas arbitrarias. Utilizando la programación por puntos, tienen la capacidad de generar, o esperar, una señal binaria en puntos predeterminados del programa del robot. Esto sirve para coordinar los movimientos del robot con otros dispositivos del entorno.

El uso de sensores elementales implica el uso de sensores binarios que suelen estar conectados al controlador del robot a través de una lógica orientada a los bits de entrada y salida.

Los sistemas operativos suministrados con los lenguajes de programación de primera generación se diseñaron para que fueran autónomos y pudieran ser utilizados por operarios sin formación informática.

Algunos de los lenguajes de programación de robots de tercera generación son: AR-BASIC [34]; D-TRAN, Seiko [35]; SIGLA, Olivetti [36]; VAL, Unimation Inc. [37, 38, 39]...

Estos sistemas de tercera generación tienen serias limitaciones que restringen sus aplicaciones robóticas. Son incapaces de realizar cálculos complejos, y son difíciles de interconectar con cámaras de visión y otros sensores complejos. Los lenguajes sólo proporcionan una comunicación limitada con otros ordenadores y controladores, y no son extensibles de forma que permita añadir comandos complejos y construir nuevas funciones en el lenguaje. La interfaz humano-

máquina (HMI) fue diseñada para permitir la flexibilidad en la programación sin que se requiera la experiencia del programador.

2.5.4 Programación de robots de cuarta generación:

Los lenguajes de programación de cuarta generación fueron desarrollados por informáticos para superar las limitaciones de extensibilidad, comunicación y potencia de cálculo de los lenguajes de las primeras generaciones.

Los robots de esta generación, robots móviles, tienen la complejidad computacional de un lenguaje informático estructurado moderno, así como la capacidad de participación en diversos procesos gracias a la inteligencia artificial.

Algunos de los lenguajes más recientes, como VAL II, se basan en lenguajes de programación anteriores. Aunque estos lenguajes tienen las extensiones robóticas esperadas de movimiento, comunicación de sensores y control, también se han mejorado mediante sistemas operativos con editores más potentes y nuevas capacidades de manejo de archivos.

Lamentablemente, muchos lenguajes de cuarta generación suministrados por los proveedores de robots son sistemas cerrados, lo que restringe sus capacidades de comunicación informática y de control entre varias máquinas.

Algunos de los robots funcionan modificando su geometría de forma similar a la programación de las máquinas herramienta de control numérico.

Algunos de los lenguajes de programación de robots de cuarta generación actualmente disponibles son:

- AI, Universidad de Stanford [40]; AMI, IBM Corporation [41];
- HELP; General Electric Company [43];
- RAPID, ABB Robotics [42]
- JARS, Jet Propulsion laboratory [44];
- MCI, McDonnell Douglas Automation Company (McAUTO) y el proyecto ICAM de la Fuerza Aérea de Estados Unidos [45];

- RAIL, Automatix Inc. [46];
- VAL-II, Unimation Inc. [47, 48].
- KRL, Kuka [49]

La principal ventaja del software de cuarta generación reside en su extensibilidad. La mayoría de lenguajes actuales son modificables y extensibles, esto permite que se puedan diseñar nuevas aplicaciones y funcionalidades para los robots para la satisfacción de los usuarios.

La peor desventaja de los lenguajes de cuarta generación es la dificultad de programar con ellos, pero estas debilidades suponen un reto para los investigadores a la hora de diseñar. Esto obliga a que la programación sea desarrollada por personas con experiencia y conocimientos de programación.

2.5.5 Programación de robots de quinta generación

Los robots actuales se clasifican dentro de la quinta generación.

En la quinta generación se encuentran las máquinas mejor dotadas de inteligencia artificial. Se tratan de robots inteligentes capaces de realizar tareas de gran variedad de forma autónoma. El “machine learning” actual permite que la evolución de los robots esté avanzando exponencialmente.

Con respecto al software utilizado actualmente para la programación de estas máquinas, se encuentra:

- OROCOS y ROS: Son frameworks que permiten varios tipos de lenguajes de programación de robots, y proporcionan entornos de desarrollo, simulación, librerías y componentes, muy útiles para reunir todo en un único paquete para programar robots o conjuntos de estas máquinas. [50]
- Lenguajes actuales de programación: entre ellos se encuentran C, C++, Python, Java, Matlab o PHP, entre otros, dependiendo de la aplicación.

3 NAVEGACIÓN Y LOCALIZACIÓN DE ROBOTS MÓVILES

3.1 TELEOPERACIÓN DE ROBOTS

La dificultad de programación de los robots para operaciones y tareas complejas y con capacidad de cambio ante situaciones ha llevado a cabo la necesidad del desarrollo de la teleoperación de estas máquinas.

Se llama teleoperación de robot al caso donde un operador humano, manipula un robot ubicado remotamente respecto a su posición.

La necesidad de teleoperación en un robot es incompatible con la autonomía robótica, ya que el control y la toma de decisiones son llevadas a cabo por el propio robot, mientras que, si este es teleoperado, el propio operario humano se encarga de tomar las decisiones y controlar manualmente la máquina.

Las tareas de manipulación compleja, y percepción del entorno en robots teleoperados, las realizan operarios cualificados en tiempo real, de modo que se cierra un bucle de control de alto nivel. Los avanzados sistemas de comunicación comparten información de realimentación sensorial con el operador, que son los datos relevantes para una toma de decisiones y control, como son imágenes, distancia, fuerza. etc.

En las tareas de manipulación remota de robots, por ejemplo, se utilizan brazos antropomórficos con controladores automáticos para reproducir los movimientos físicos del operario. Normalmente estos casos son utilizados en tareas de alto riesgo (mantenimiento de líneas eléctricas), entornos no estructurados o tareas difíciles de automatizar (operaciones de cirugía).

3.2 INTELIGENCIA Y VISIÓN ARTIFICIAL

La función de visión artificial en la robótica se creó en las décadas de 1980 y 1990. Los ingenieros idearon un método para dotar de visión a un robot. La pieza

se comparaba con un modelo matemático y se rechazaba si no complementaba la fórmula. Este método se aplica con mayor frecuencia en aplicaciones de manipulación y selección de materia en la industria del embalaje, recogida y colocación, desbarbado, rectificado y otros procesos industriales.

Se llama visión artificial al método de obtención, decodificación y procesamiento de una imagen, mediante una combinación de hardware y software, para la ejecución de unas determinadas funciones [51].

Los sistemas de visión artificial industriales exigen más estabilidad, confiabilidad de los datos obtenidos y solidez en comparación con sistemas de visión para uso académico, pero son menos restrictivos que aquellos sistemas que se usan para fines militares y gubernamentales.

Estos sistemas se basan en sensores digitales albergados en cámaras industriales con una óptica diseñada para la obtención de imágenes. (Figura 3.1). Este método general es ejecutado en la mayoría de los algoritmos para la visión artificial.

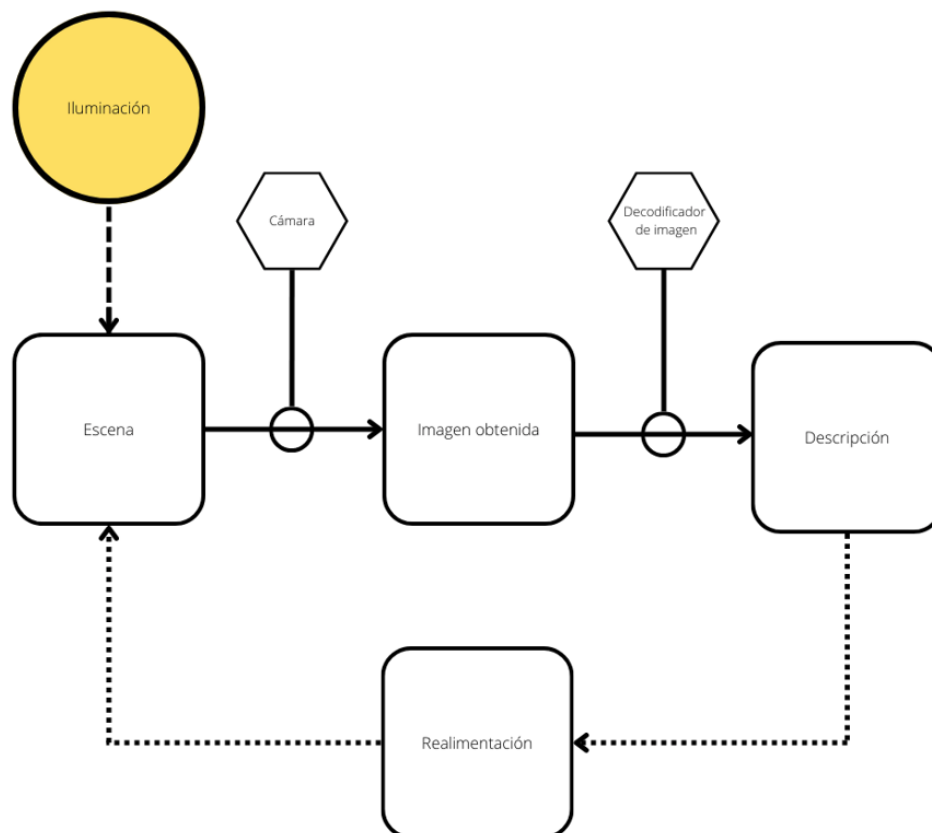


Figura 3.1: Arquitectura sistema de visión robótica.

En la arquitectura de los sistemas de visión robótica se destacan seis pasos de funcionamiento:

1. Detección: Proceso donde se detecta una imagen visual a través de la cámara del robot.
2. Preprocesamiento: Se mejoran los detalles de la imagen detectada, reduciendo el ruido y aplicando filtros de imagen.
3. Segmentación: Se divide la imagen en una zona de interés para analizar.
4. Posprocesamiento y descripción: Se calculan características para diferenciar objetos.
5. Reconocimiento: Se detecta el objeto de interés en la imagen analizada y procesada.
6. Interpretación: Una vez detectado el objeto de interés se le asigna el significado correspondiente en el programa.

Algunas aplicaciones habituales de la visión artificial en robótica son:

- Inspecciones de embalaje
- Mediciones
- Lectura de códigos de barra y QR, mediante escáner
- Evaluaciones de calidad y superficies de materiales
- Verificación y actuación en la orientación y manipulación de piezas
- Detección de defectos

En el apartado 5 de este documento se detallan más características acerca de la visión artificial en los robots.

3.3 GUIADO AUTÓNOMO EN APLICACIONES INDUSTRIALES

El guiado autónomo en los robots móviles o de servicio, se contempla como un sistema con capacidad para moverse o desplazarse de un lugar a otro de forma tele operada o autónoma con el objetivo de realizar una acción o tarea programada [55].

Se resalta la importancia de la búsqueda de la autonomía en los robots para conseguir máquinas que puedan trabajar de forma independiente, en entornos complejos, sin la necesidad constante de control por parte de un operador humano.

Además del movimiento autónomo, se busca la calidad de movimiento, ya que el robot móvil necesita esquivar ciertos obstáculos a medida que se desplaza y elegir la trayectoria más eficiente hasta sus destinos evitando las colisiones.

Un papel importante en las máquinas móviles lo tienen los sensores instalados en los robots en cuestión. Estos sensores son protagonistas en obtener toda la información del entorno que rodea a la máquina, procesarla y tomar decisiones para navegar de forma segura [56].

En algunas investigaciones se describen dos grupos de algoritmos basados en la navegación de robots móviles [57], en el primer grupo, se clasifican los robots que tienen información completa de los mapas del entorno por donde van a desplazarse, mientras que en el segundo grupo se clasifican los robots basados en algoritmos para navegación con información incompleta del entorno de funcionamiento, los cuales dispondrán de aplicaciones para navegación en lugares desconocidos, por ejemplo robots de exploración agrícolas o de vigilancia.

Por ejemplo, para el desplazamiento de las máquinas a través de pasillos se han estudiado varios tipos algoritmos distintos de control, entre los que se pueden encontrar:

- Dotación de dos cámaras laterales al robot para obtener imágenes sobre la velocidad aparente de la máquina mientras está en movimiento, estas imágenes se comparan dando como resultado un parámetro útil para el control de velocidad del robot [58].
- Detección de líneas de perspectiva del pasillo a través de las cámaras del robot para mantener la máquina en el centro del pasillo en todo momento [59].
- Utilización de una cámara para guiar el robot a lo largo del eje central del pasillo utilizando cálculo de flujo óptico y sus derivadas temporales [60]

- Uso de sensores de odometría y de detección de obstáculos con un algoritmo de control para el seguimiento de una pared implementando codificadores incrementales y sensores sonar [61].

4 PLATAFORMA RASPBERRY PI 4 MODEL B

4.1 DESCRIPCIÓN DE LA PLATAFORMA

La Raspberry PI 4 model B (Figura 4.2), es el último de los modelos de Raspberry, se trata de un ordenador pequeño de bajo coste desarrollado en reino unido, su objetivo de creación es poder poner en manos de las personas de todo el mundo el poder de la informática y la creación digital.

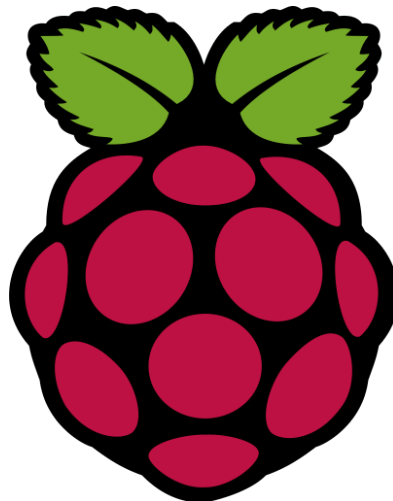


Figura 4.1: Logotipo Raspberry.

Sus especificaciones técnicas son las siguientes [62]:

Sistema de un chip	Broadcom BCM2711
CPU	Procesador de cuatro núcleos a 1,5 GHz, Cortex-A72
GPU	VideoCore VI
Memoria	1/2/4GB LPDDR4 RAM

Conectividad	802.11ac Wi-Fi / Bluetooth 5.0, Gigabit Ethernet
Video y sonido	2 x puertos micro-HDMI que admiten pantallas de 4K@60Hz a través de HDMI 2.0, puerto de pantalla MIPI DSI, puerto de cámara MIPI CSI, salida estéreo de 4 polos y puerto de vídeo compuesto.
Puertos	2 x USB 3.0, 2 x USB 2.0
Alimentación	5V/3A vía USB-C, 5V vía GPIO



Figura 4.2: Raspberry Pi 4 Model B.

4.2 INSTALACIÓN Y ADAPTACIÓN DEL ENTORNO

El primer paso para el funcionamiento de la Raspberry es instalarle el sistema operativo. Hay varias opciones de SO compatibles [64]. En este caso se ha instalado el sistema operativo Raspbian con escritorio (Figura 4.3)



Figura 4.3: Escritorio Raspbian (Raspberry Pi).

Más adelante se detalla toda la configuración inicial del sistema, en el apartado 6.3.

5 PROCESAMIENTO DE INFORMACIÓN DE LOS ROBOTS

El robot es esencialmente un sistema organizado que responde con acciones inteligentes a los estímulos que percibe, este proceso es gracias al conjunto de elementos que componen al robot, como la unidad de control (Raspberry PI), el sistema de accionamiento (actuadores y motores) o el sistema sensorial, que

está formado por los sensores encargados de recoger la información acerca del estado del propio robot y de su entorno.

En este apartado se citan métodos y herramientas, algunos utilizados en el proyecto, para la obtención de información sensorial relacionada con la visión artificial:

5.1 CÁMARA RASPBERRY PI

El robot cuenta con una cámara compatible con la Raspberry PI con un sensor de 8 megapíxeles. (Figura 5.1).



Esta cámara es utilizada como sensor de video e imágenes en el robot, para las funciones principales que se programan en el proyecto, (detección de códigos QR). Esta programación se detalla completamente más adelante, en el apartado (6.5).

Figura 5.1: Cámara Raspberry.

Los datos técnicos de la cámara son:

Sensor	Sony IMX219 8Mpx
Resolución de fotos	3238 x 2464 (máx.)
Resolución de vídeo	1080p30, 720p60 y 640x480p90
Interfaz	Puerto dedicado CSI
Lente de foco fijo en placa	
Óptica	1/4"
Dimensiones	25mm x 23.86mm x 9mm
Peso	3g

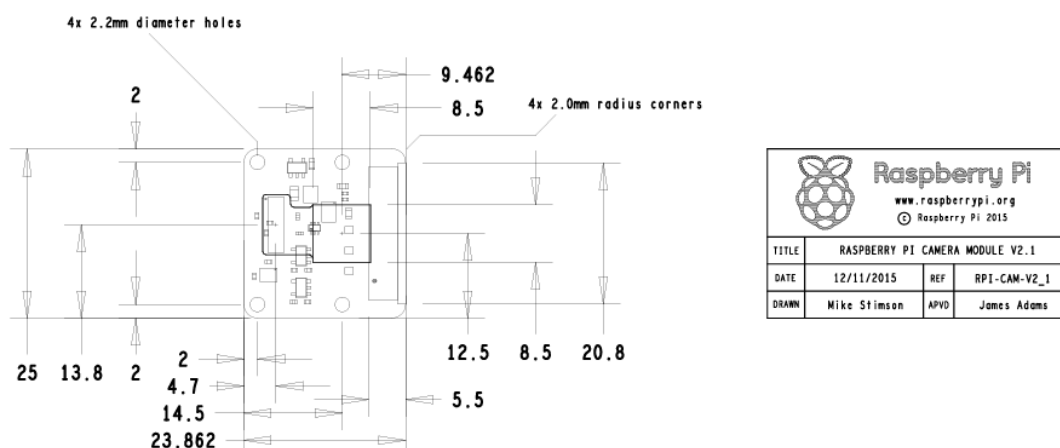


Figura 5.2: Dimensiones de la cámara Raspberry.

5.2 BIBLIOTECA OPEN CV

La librería Open CV (Open Source Computer Vision Library) [10] es una biblioteca de código abierto que incluye varios cientos de algoritmos de visión por ordenador en tiempo real. Esta biblioteca es una de las principales indispensables para la programación del robot del proyecto. Se permite su uso para fines comerciales y académicos gracias a su licencia BSD (Libre de permisos).

La librería Open CV, se instala en la configuración inicial del robot (ver apartado 6.4.1),



Figura 5.3: Librería Open CV.

Open CV permite que se puedan realizar las funciones de detección y procesamiento de códigos QR, técnica que se usa en este proyecto para el funcionamiento del robot, en el apartado (6.5.1). Además de esta función, la librería trae instaladas una serie de utilidades. Tiene diferentes características que ahorran bastante tiempo y trabajo para el programador tales como, segmentación del color, filtros morfológicos, reconocimiento de placas de vehículos. etc.

5.3 DETECCIÓN DE OBJETOS, COLORES Y CÓDIGOS

La biblioteca de Open CV tiene instalados montones de scripts capaces de reconocer colores, objetos, rostros fáciles (Anexo 3), códigos de barra. etc.

El funcionamiento de la mayoría de los scripts se basa en el procesamiento de las imágenes que son captadas a través de una cámara.

Es posible detectar objetos [65 y 66] mediante el algoritmo de las cascadas de HAAR (Anexo 4), que se basan en un método de machine learning, en el cual se aplica un enfoque sobre imágenes positivas y negativas dando como resultado el objeto detectado sobre la imagen. (Ver Figura 5.4)



Figura 5.4: Detección de objeto.

5.4 GENERACIÓN DE OBJETO SOBRE DETECCIÓN DE FORMA (ArUco)

El módulo “ArUco” está basado en la librería ArUco, una popular librería para la detección de marcadores fiduciales cuadrados, del estilo de los códigos QR, desarrollada por Rafael Muñoz y Sergio Garrido: [11]. Esta librería pertenece a la biblioteca de Open CV.

Uno de los enfoques más populares de la librería es el uso de marcadores fiduciales cuadrados binarios. La principal ventaja de estos marcadores es que un solo marcador proporciona suficientes datos (sus cuatro esquinas) para obtener la posición de la cámara mientras lo enfoca. Además, la codificación binaria interna los hace especialmente robustos, permitiendo la posibilidad de aplicar técnicas de detección y corrección de errores.

Con programación es posible aplicar la realidad aumentada utilizando esta librería [67], pero se encuentra fuera del alcance de este proyecto.



Figura 5.5: Marcadores fiduciales.

6 DESARROLLO DEL PROYECTO

6.1 INTRODUCCIÓN ROBOT RASPTANK

En el proyecto se persigue la emulación física de un proceso industrial de líneas de robots móviles con tecnología avanzada en inteligencia y visión artificial. Mediante este proceso se trata de mostrar un grado de autonomía en el funcionamiento de los robots, pudiéndose llevar a cabo en situaciones reales de tareas industriales totalmente automatizadas, sin depender en ningún momento de la mano de obra humana.

Se busca un sistema totalmente independiente en el ámbito funcional, mediante la adaptación de un proyecto ya existente a otro, creando un conjunto más completo para la funcionalidad deseada. Se añaden una serie de extras en la programación de los sensores del robot utilizado, entre otras cosas que se detallarán en los siguientes apartados.

6.2 MECÁNICA Y HARDWARE

La mecánica del robot se basa en una estructura de metacrilato oscuro translucido que conforma la mayoría de su chasis, incluye un brazo robótico de cuatro grados de libertad fabricado del mismo material, con una base fija. (Figura 6.1 y 6.2)

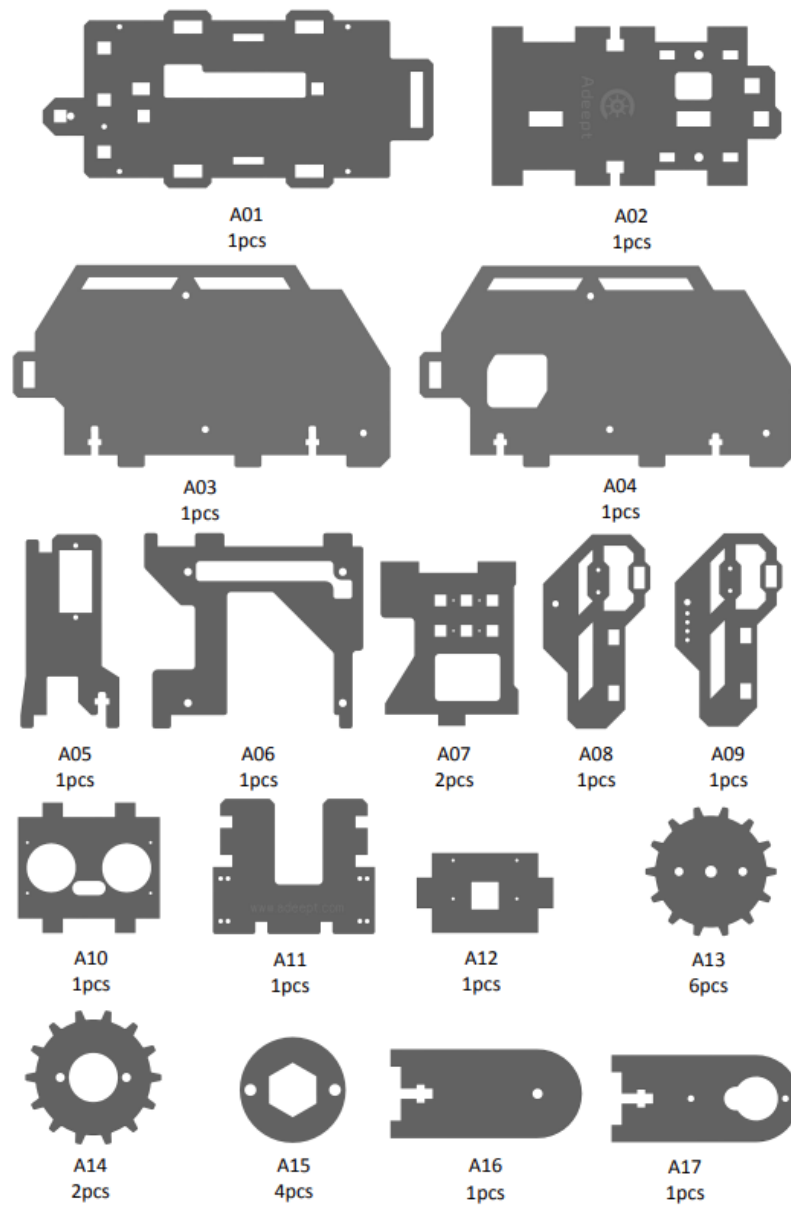


Figura 6.1: Piezas y mecanismos (1).

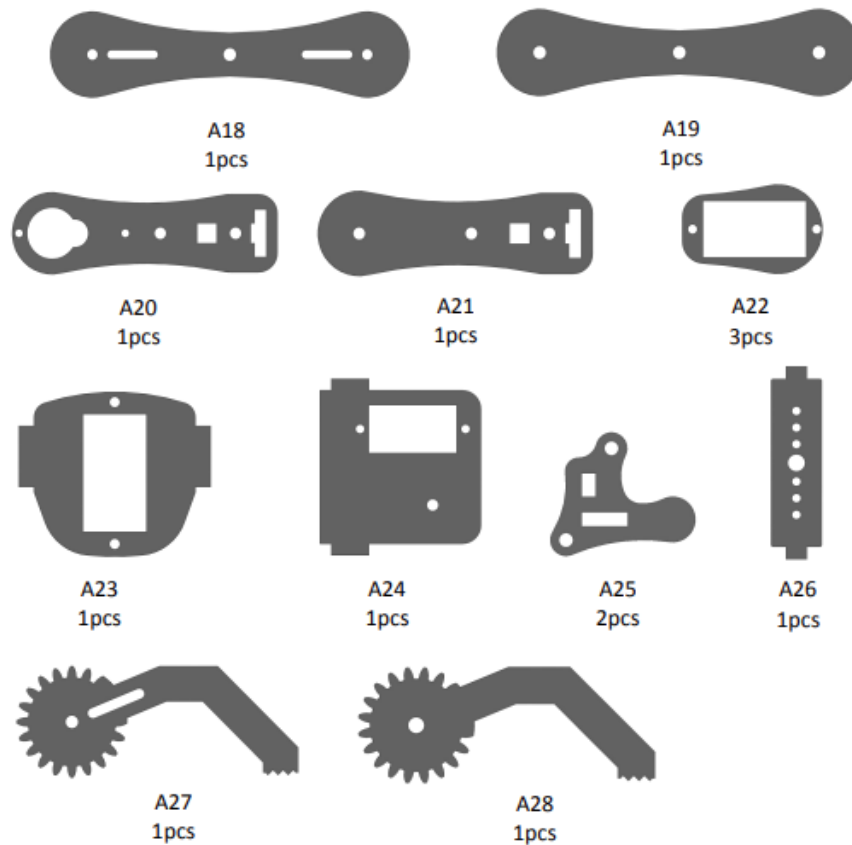


Figura 6.2: Piezas y mecanismos (2)

Cada una de estas piezas están ensambladas a mano meticulosamente, unidas mediante tornillos, tuercas y piezas con engranajes. (Figura 6.6)

La unión de todas las piezas da lugar a la estructura mecánica principal del robot, que presenta una forma de tanque (Figura 6.13) de guerra con un brazo robótico y una pinza en el extremo de éste, la cual le permite la realización de acciones teleoperadas o programadas, o la manipulación de objetos mediante el cierre o apertura de la pinza. (Elemento A27 y A28, Figura 6.2)

Para el accionamiento y movimiento del robot completo se usa un conjunto de motores eléctricos que se encargan de ejecutar las órdenes recibidas del software que se detalla más adelante.

Concretamente, el conjunto motriz del robot (Figura 6.3), está dividido en 5 servomotores y dos motores DC de corriente continua.



Figura 6.3

Los servomotores se encargan del movimiento del conjunto del brazo robótico antropomórfico y del conjunto de la cámara y el sensor de ultrasonidos, mientras que los dos motores DC de corriente continua engranan en un sistema de orugas (Figura 6.4).

Las orugas se encargan, del desplazamiento del robot completo en el plano del suelo.



Figura 6.4: Orugas del robot.

El robot cuenta con 12 Leds RGB (Figura 6.5) conectados en serie formando 4 líneas laterales de 3 chips cada una. Estos Leds programables se encargan de iluminar el robot mediante unas secuencias establecidas.

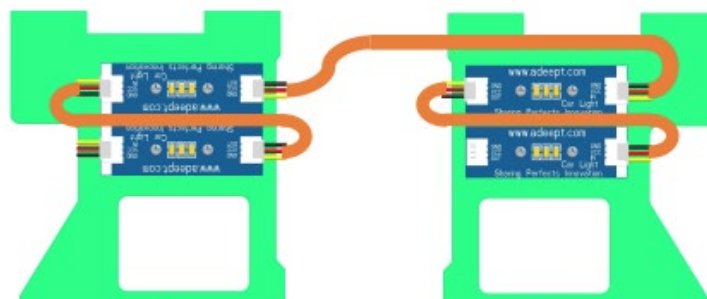


Figura 6.5 Módulos LED conectados en serie.

<p>M2 Nut</p>  <p>X14 www.adept.com</p>	<p>M2.5 Nut</p>  <p>X4 www.adept.com</p>	<p>M3 Nut</p>  <p>X12 www.adept.com</p>	<p>M3 Lock Nut</p>  <p>X3 www.adept.com</p>	<p>M2*10 Screw</p>  <p>X14 www.adept.com</p>
<p>M2.5*4 Screw</p>  <p>X4 www.adept.com</p>	<p>M2.5*8 Screw</p>  <p>X4 www.adept.com</p>	<p>M3*8 Screw</p>  <p>X28 www.adept.com</p>	<p>M3*12 Screw</p>  <p>X12 www.adept.com</p>	<p>M3*18 Screw</p>  <p>X1 www.adept.com</p>
<p>M3*30 Screw</p>  <p>X2 www.adept.com</p>	<p>M4*6 Screw</p>  <p>X2 www.adept.com</p>	<p>M3*10 Countersunk Head Screw</p>  <p>X2 www.adept.com</p>	<p>M1.4*6 Self-tapping Screw</p>  <p>X16 www.adept.com</p>	<p>M2.5*4+6 Copper Standoff</p>  <p>X8 www.adept.com</p>
<p>M3*6 Copper Standoff</p>  <p>X2 www.adept.com</p>	<p>M3*8 Copper Standoff</p>  <p>X1 www.adept.com</p>	<p>M3*18 Copper Standoff</p>  <p>X8 www.adept.com</p>	<p>M3*35 Copper Standoff</p>  <p>X2 www.adept.com</p>	<p>M3*30 Nylon Standoff</p>  <p>X1 www.adept.com</p>
<p>M3*40 Nylon Standoff</p>  <p>X2 www.adept.com</p>				

Figura 6.6: Tornillos y tuercas de la estructura del robot.

Sobre el hardware del robot se encuentran varios elementos que se encargan de entrelazar todo el sistema electrónico:

6.2.1 Raspberry Pi 4 Model B

Es el ordenador a bordo del robot. Se encarga de ejecutar todo el programa del proyecto para conseguir que el robot realice las funciones asignadas.

En la Raspberry Pi se realizan todas las conexiones del módulo Motor HAT, que viene incluido con el robot, el cableado de los motores, el sensor de ultrasonidos (figura 6.12), la tarjeta micro SD (figura 6.9), el sensor de línea (figura 6.11), y algunas más, que se detallan mejor en el apartado 4 de la memoria.

6.2.2 Motor HAT

Poderoso módulo multifuncional (Figura 6.7) que se integra a la Raspberry Pi para aportar una serie de funciones necesarias para el proyecto, entre los elementos que conforman la PCB se encuentran:

- Switch ON/OFF: Apaga o enciende la fuente de alimentación externa (Figura 6.7) de la Raspberry.
- Puerto USB con salida de 5V
- Puertos de conexión de servomotores
- Puertos de conexión de motores DC
- Puertos de conexión de Leds RGB
- Puerto de conexión de sensor de ultrasonido
- Conector de alimentación (3.7V)

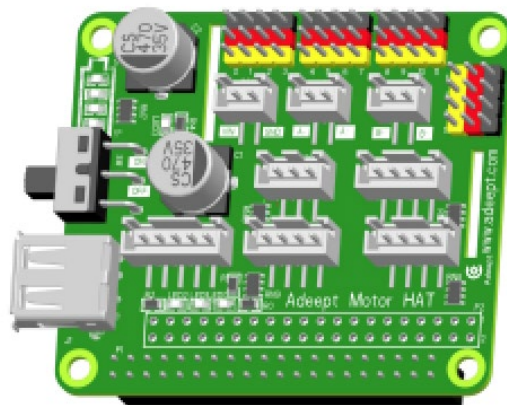


Figura 6.7 Módulo motor HAT.

6.2.3 Cámara Raspberry

La cámara (figura 5.1) se conecta directamente a la Raspberry, se encarga principalmente de capturar imágenes y vídeos para ser procesados por el microprocesador. Se detallan más funciones de ésta en el apartado (5.1).

6.2.4 Iluminación LED

Se interconectan en serie 4 módulos de 3 chips led cada uno para conformar la iluminación exterior del robot. (Figura 6.8)



Figura 6.8 Módulo LED.

6.2.5 Tarjeta Micro SD

Se utiliza una tarjeta micro SD de 64 GB para soportar el sistema operativo del robot y la memoria del sistema, así como algunos archivos de configuración que se detallan en el apartado siguiente (6.3) y el proyecto completo.

Es importante la elección de este elemento, ya que el rendimiento de la Raspberry depende de la tarjeta micro SD que se use para su funcionamiento.

Se necesita una tarjeta que tenga un buen rendimiento en escritura/lectura de datos. (Figura 6.9)



Figura 6.9 Tarjeta micro SD.

6.2.6 Baterías

Todo el hardware del robot se alimenta de dos baterías Litio-Ion tipo 18650 de 3.7V (Figura 6.10) que permiten operar con el robot de forma remota sin cables de alimentación.

Para activar la alimentación por baterías es necesario accionar el switch del Motor HAT, que se encuentra junto al puerto USB. (Figura 6.7)



Figura 6.10 Baterías 18650.

6.2.7 Sensor de línea

Se encarga de guiar el robot a través de las líneas delimitadas en la superficie del suelo. (Figura 6.11). Este funcionamiento se detalla en el apartado (6.5.2)



Figura 6.11 Sensor de línea.

6.2.8 Sensor de ultrasonidos

El sensor de ultrasonidos se usa en el robot para detectar obstaculos o paredes que se vaya a encontrar de frente durante su recorrido. De tal forma puede actuar en el momento preciso programado. (Figura 6.2.12). Funciona emitiendo y recogiendo sonidos ultrasónicos, de tal forma el tiempo que tarda la onde del sonido en llegar es directamente proporcional a la distancia del objeto que detecta. [69].

Este sensor se encuentra conectado al modulo Motor HAT al igual que los leds, el sensor de linea y los motores. En el apartado (6.5.3) se detalla su funcionamiento en el proyecto.

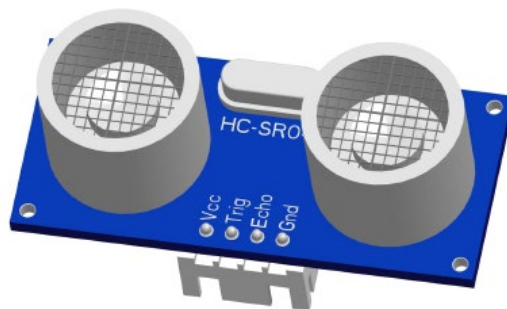


Figura 6.12 Sensor de ultrasonidos.

6.2.9 Cableado y conectores

El robot está dotado de líneas de cables que interconectan todos los elementos y sensores necesarios en el robot entre su brazo antropomórfico, el módulo Motor HAT y la Raspberry Pi.

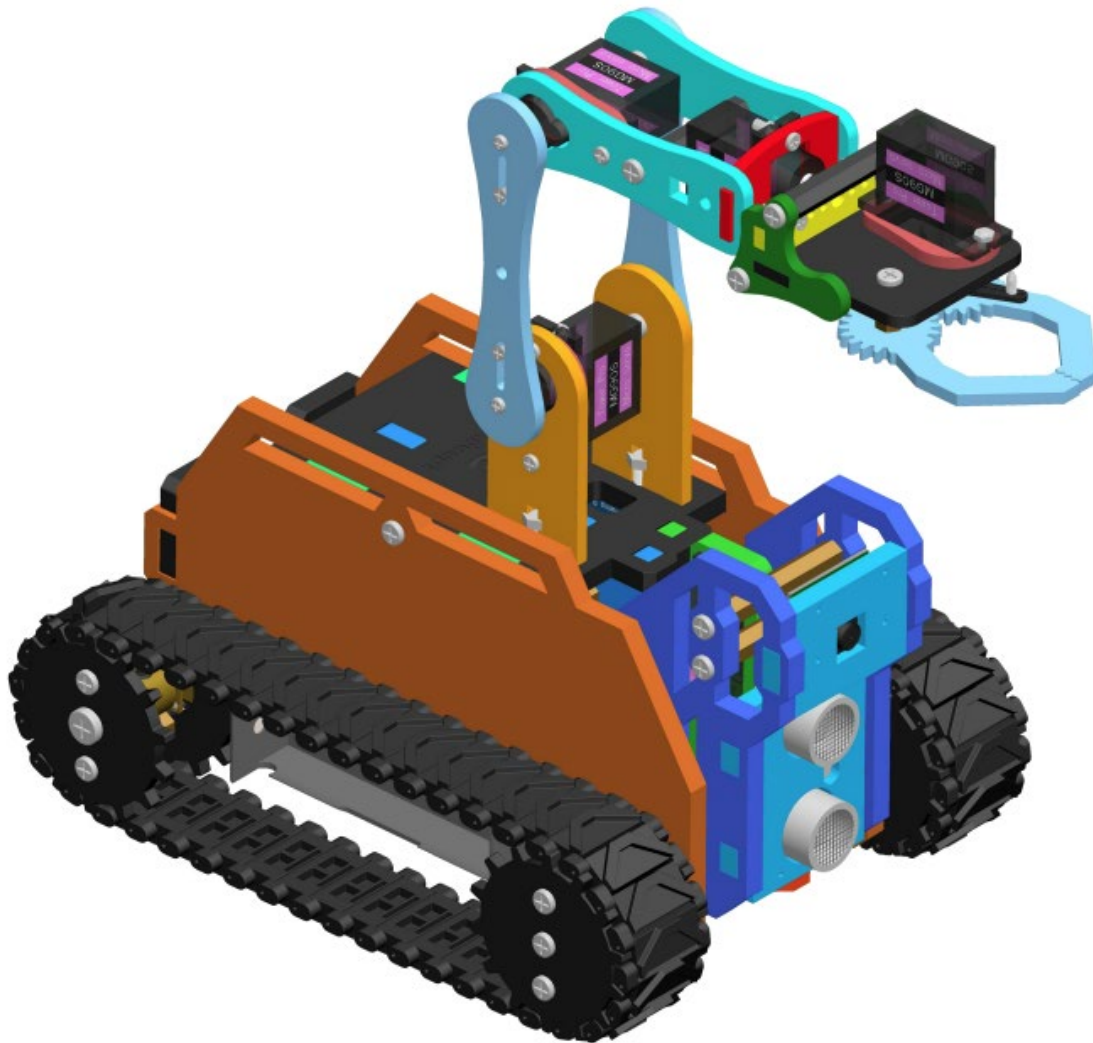


Figura 6.13 Robot ensamblado..

6.3 CARACTERÍSTICAS Y ADAPTACIÓN SOFTWARE

En el proyecto se han utilizado una serie de configuraciones en la Raspberry necesarias que se detallan a continuación:

6.3.1 Configuración del sistema operativo

Formateando previamente la tarjeta micro SD, se instala el sistema operativo Raspbian [63] en la carpeta raíz de la tarjeta micro SD.

Este sistema operativo se trata de una distribución de Linux adaptada para las Raspberry para mejorar el rendimiento y efectividad de estos dispositivos para aplicaciones de propósito general. Sin embargo, hay una versión más actual del sistema operativo (Raspberry Pi OS) que no se usa en este proyecto para evitar fallos de compatibilidad entre versiones a lo largo del desarrollo.

6.3.2 Configuración del protocolo SSH

Una vez grabada la imagen del sistema operativo en la tarjeta micro SD, se necesita habilitar el protocolo SSH en la configuración de la Raspberry Pi.

El protocolo SSH permite conectar y trabajar en la Raspberry desde otro dispositivo de forma remota estableciendo entre ellos una conexión en la misma red de forma segura, estando cifrada toda la información compartida entre los dispositivos.

La configuración de habilitación del protocolo en la Raspberry se puede hacer internamente, operando desde el propio sistema operativo, o de forma más simple, creando un archivo vacío en la carpeta raíz del sistema operativo (albergado en la tarjeta micro SD). El nombre de este archivo es "ssh" sin extensiones.

6.3.3 Configuración de la conexión Wifi

Además de la configuración del protocolo SSH es necesario agregar otro archivo a la carpeta raíz de la tarjeta micro SD con la configuración de la conexión wifi a la que se va a conectar la Raspberry utilizando dicho protocolo. (Figura 6.14)

En primer lugar, se crea un archivo con el nombre (wpa_supplicant) y extensión (.txt) en el que se escribe el código de configuración de la red wifi a la que se desea conectar la Raspberry PI. Ver (Anexo 1).

Una vez cumplimentada correctamente la información en el archivo, este se guarda como (wpa_supplicant.conf) en la carpeta raíz de la tarjeta micro SD (Figura 6.14)

Nombre	Fecha	Tipo	Tamaño	Duración
ssh	22/03/2021 20:50	Archivo	0 KB	
wpa_supplicant.conf	22/03/2021 21:15	Archivo CONF	1 KB	

Figura 6.14: Archivos de configuración inicial de Raspbian.

6.3.4 Instalación PuTTY

Una vez se ha configurado el protocolo SSH y el archivo para la conexión wifi, se podrá acceder al símbolo del sistema de Raspbian mediante PuTTY, programa que es necesario instalar en el dispositivo desde el cual se pretende acceder a Raspbian.

PuTTY es un programa que permite acceder como cliente de servidor al sistema de la Raspberry a través de conexión SSH [5].

Una vez dentro del símbolo del sistema de Raspbian se instala el proyecto base en el sistema [6] sobre el cual se trabaja en este trabajo fin de grado para añadirle nuevas funcionalidades más avanzadas.

6.3.5 Lenguaje Python

Es un lenguaje de programación multiplataforma orientado a objetos que permite realizar cualquier tipo de programa desde servidores de red hasta aplicaciones de Windows.

Este lenguaje, además de ser utilizado en el proyecto base por defecto, es el más adecuado para este tipo de aplicaciones.

Es un lenguaje con un gran potencial y diversidad por tener la capacidad de ser utilizado en varias plataformas que además posee un gran número de librerías que permiten trabajar con él. De esta forma abre un abanico de posibilidades para conseguir los objetivos marcados en este proyecto.

Para llevar a desarrollar toda la programación del proyecto ha sido necesario instalar Python 3.7 [7] y MobaXterm [8].

6.3.6 MobaXterm

MobaXterm [9] es un cliente SSH similar a PuTTY, pero con la ventaja de poder trabajar desde el escritorio de la Raspberry conectada remotamente, ya que PuTTY solo puede trabajar a través de una consola de comandos.

Este cliente es imprescindible en el proyecto para poder crear scripts dentro del propio sistema Raspbian y así poder depurar el código del proyecto más rápidamente.

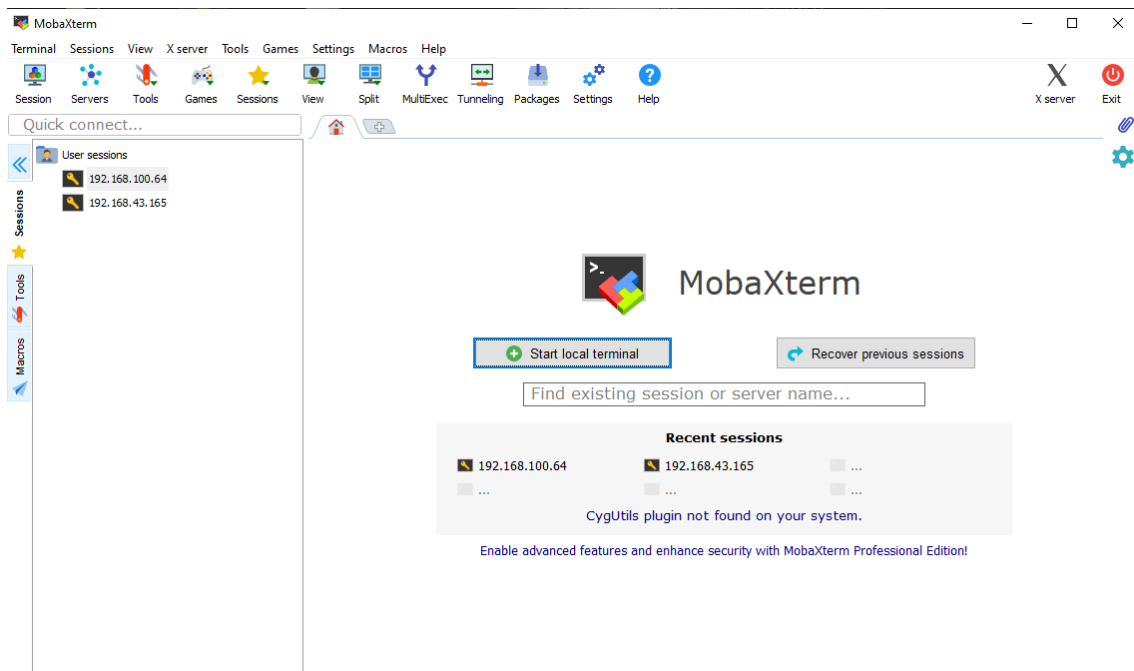


Figura 6.15: Interfaz de usuario de MobaXterm.

En la figura mostrada (Figura 6.15) se puede apreciar que dependiendo donde esté el robot físicamente, se conectará a través de una dirección IP u otra a la aplicación MobaXterm a través del wifi mediante el protocolo SSH.

Según se detalla en la configuración de conexión a la red wifi en el apartado (6.3.3) el robot se conectará con la dirección IP local del router fijo o de la conexión compartida wifi portátil del teléfono móvil, en el caso de que se encuentre fuera del alcance de otra conexión wifi configurada anteriormente en (6.3.3).

6.4 LIBRERÍAS Y HERRAMIENTAS UTILIZADAS EN LA PROGRAMACIÓN DEL PROYECTO

Para la programación del proyecto hay un conjunto de librerías que son imprescindibles para el correcto funcionamiento del robot. La mayoría de estas librerías se instalan automáticamente en la instalación del sistema operativo, pero hay otras que se deben de instalar manualmente.

6.4.1 OpenCV

Open CV es una librería de visión por ordenador de código abierto. Es imprescindible para el correcto funcionamiento del robot, e incluye una gran cantidad de herramientas muy útiles

En el apartado (5.2) se describen más detalles acerca de esta librería.

6.4.2 Numpy

Esta librería se encarga de dar soporte al proyecto para poder trabajar con grandes vectores y matrices, así como una gran colección de funciones matemáticas de alto nivel [71].

6.4.3 Thonny Python

Thonny Python es el IDE (Entorno de desarrollo interactivo) que se ha utilizado en el proyecto para desarrollar todo el código de programación. (Figura 6.16)

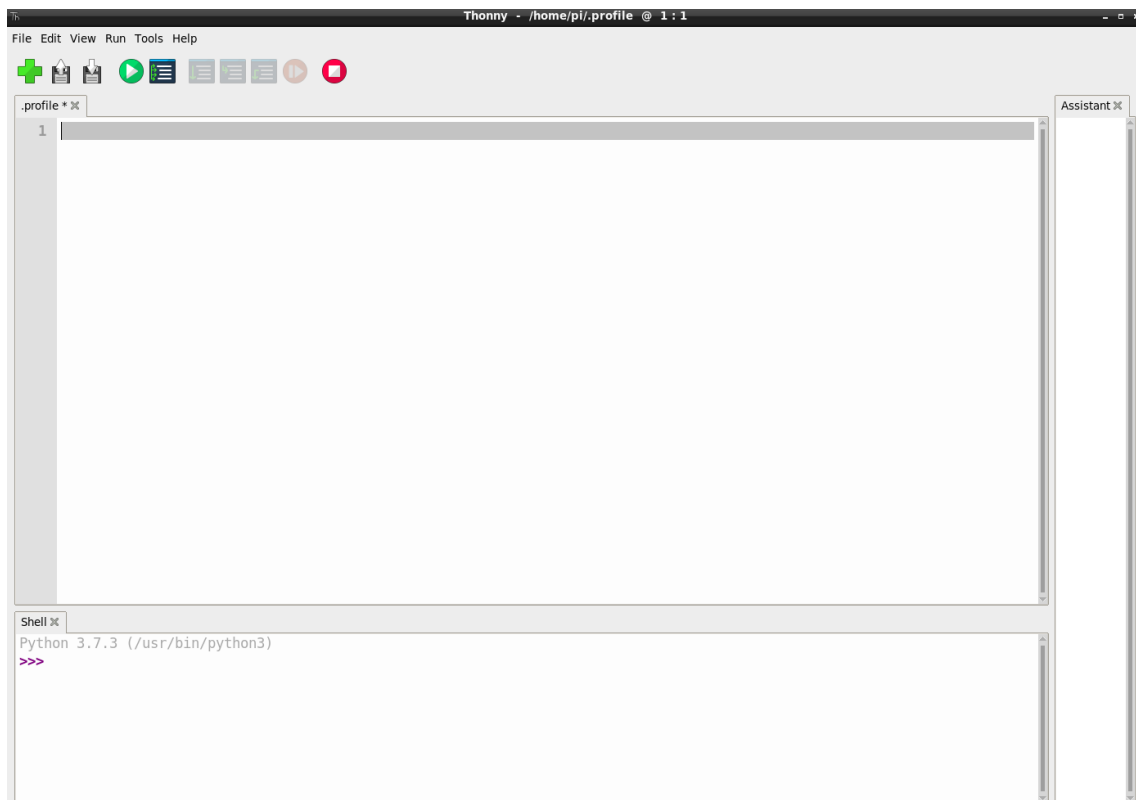


Figura 6.16: Entorno de programación en Thonny Python.

6.4.4 PyCharm

Es otro IDE más profesional que se ha utilizado para navegar entre scripts del proyecto base más rápidamente desde otro dispositivo externo a la Raspberry [72].

Se ha utilizado en menor medida que el IDE del apartado anterior ya que, debido a los recursos de software que requiere, no se recomienda su instalación en la Raspberry.

6.4.5 Pyzbar

Pyzbar [73] es una de las librerías protagonistas del proyecto. Permite la lectura/interpretación de códigos de barra y códigos QR a través de la cámara de la Raspberry. Todo su funcionamiento se detalla en el apartado siguiente. (6.5.1)

6.5 PROCESAMIENTO Y GUIADO AUTÓNOMO (AGV)

6.5.1 Diseño y programación de balizas y rutas

En este apartado se detalla el proceso de programación para el guiado del robot por un circuito de prueba. Para ello se utilizan una serie de códigos QR que se han generado arbitrariamente. En el (Anexo 8) se pueden apreciar los valores asignados de los QR utilizados en cada lista.

Se utilizan 3 listas diferentes para diferenciar las secuencias de maniobras que realiza el robot: “*leftList*, *rightList* y *frontList*.” (ver Anexo 8).



Figura 6.17: Robot en posición de recorrido en el circuito de pruebas.

Cada lista almacena valores de códigos QR cifrados en formato decimal, del tipo "211116". De esta forma se consolidan las diferentes balizas por todo el circuito dependiendo del código QR detectado por la cámara del robot. (Figura 6.17)

En el momento en el que el robot detecta un código QR válido comienza la maniobra asignada por ese código a través del programa, de lo contrario, si se encuentra con un obstáculo antes de detectar dicho código QR, la máquina lo detectará como una interferencia en su camino y detendrá su movimiento. (ver Anexo 6)

En el circuito de pruebas el robot realiza un seguimiento de línea continuo entre detección y detección de códigos QR, pudiendo haber saltos entre diferentes posiciones y orientaciones de líneas. De tal forma que el robot va circulando a través de un grafo, donde la línea por la que circula son las aristas y los códigos QR son los nodos.

Actualmente en el código del robot, tiene programado un diccionario de códigos QR para evitar que un único código QR ordene varias veces la misma secuencia al robot al leerse varias veces, de esta forma se evita el colapso de lecturas de códigos en el programa.

El robot, a través del grafo, simula que va completando una lista de tareas hasta tener todas completadas [70].

En la figura (6.20) se puede apreciar el circuito de pruebas desarrollado.

6.5.2 Sensores de guiado superficial

El robot usa un sensor de guiado superficial que viene incluido en el paquete del robot RASPTANK [68]. Se trata de un sistema que funciona con sensores infrarrojos (Figura 6.11), haciendo guiar el robot a través de una línea.

El código que se utiliza se ha modificado disminuyendo el tiempo de refresco del sensor para obtener una mayor repetibilidad en la medición de la superficie a analizar. De esta forma se arreglan problemas de desviación del robot respecto a la línea que se sigue

El código (Anexo 2) realiza una comparación entre dos de los sensores infrarrojos que incorpora el módulo del sistema de guiado del robot. Cuando un sensor detecta la línea negra del circuito de pruebas, significa que el robot se va a desviar de su trayectoria. Por tanto, se programa un pequeño giro con los motores para devolver al robot a su trayectoria correcta.

6.5.3 Sensores de distancia mediante ultrasonidos

En el frontal del robot se aloja el sensor de ultrasonidos, justo debajo de la cámara (Figura 6.12). Este sensor se utiliza para determinar la distancia entre el robot y el objeto que se encuentra delante. (Ver apartado 6.2.8).

De tal forma que una vez que el sensor detecta que el robot está a una distancia programada del código QR objetivo (baliza), entonces realiza la secuencia o acción determinada para esa baliza.

6.5.4 Obstáculos y fallos de recorrido

El robot tiene programada una parada de emergencia en el momento que detecta un obstáculo en el recorrido del circuito automatizado que se le ha asignado. Se puede apreciar en la línea del Anexo 6.

6.6 RESULTADOS EXPERIMENTALES

Tras las pruebas realizadas en un circuito de pruebas, se obtienen datos relevantes para el análisis de los resultados experimentales:

6.6.1 Detección de obstáculos mediante ultrasonidos

La distancia programada de detección de los códigos QR varía entre 22 y 23 cm. Esta distancia varía por la incertidumbre de la medición del sensor de

ultrasonidos de ($\pm 1\text{cm}$) y la minúscula variación de rugosidad de la superficie en la que se encuentra el código adherido.

6.6.2 Interpretación de Códigos QR a través de la cámara de la Raspberry

La detección de los códigos en el circuito descrito en el apartado (6.5.1) como se ha descrito, se lleva a cabo en el momento de cada giro o acción del robot.

La probabilidad de que la detección del código a través de la cámara falle es muy baja. Depende principalmente de la iluminación exterior. Ya que, cuando hay suficiente iluminación, el robot detecta los códigos el 99% de los intentos.

Se ha medido la cantidad de luz necesaria para saber a partir de qué nivel las lecturas son correctas. La cantidad de luz necesaria debe ser mayor a 6 lux. (Figura 6.18)

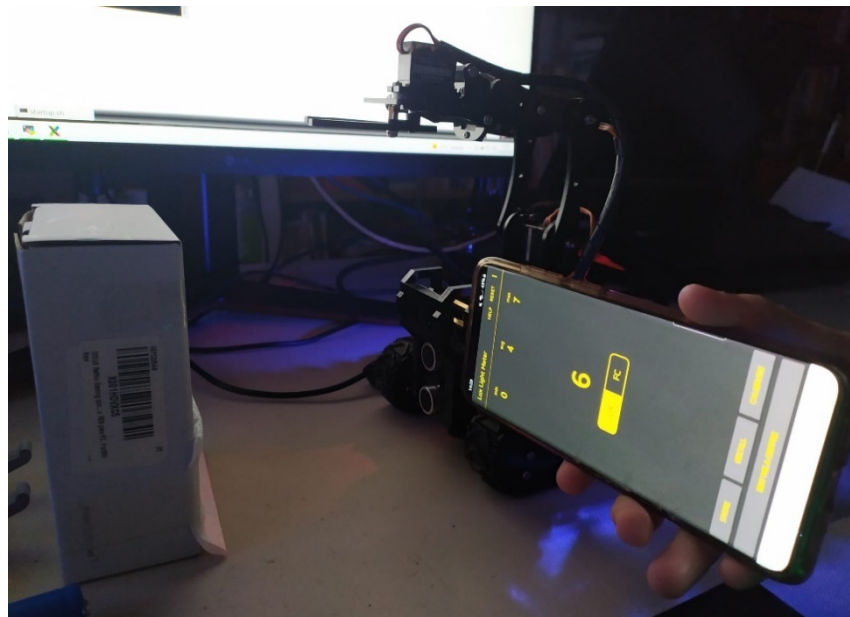


Figura 6.18: Test de iluminación.

6.6.3 Seguimiento de línea en la superficie del suelo

A través de las pruebas realizadas en el circuito que se detallan en el apartado (6.5.2), el guiado del robot a través de las líneas en la superficie del suelo se

debe establecer con anchuras de líneas comprendidas entre 1 y 2.5 cm (Figura 6.19) para su funcionamiento óptimo.

En caso de que estas líneas sean más gruesas que este intervalo, el robot no funcionará correctamente debido a la imposibilidad de funcionamiento del sensor de línea.

Por el contrario, en el caso de que las líneas sean más finas, el robot se acabaría desviando de su trayectoria, según se detalla en el apartado (6.5.2)

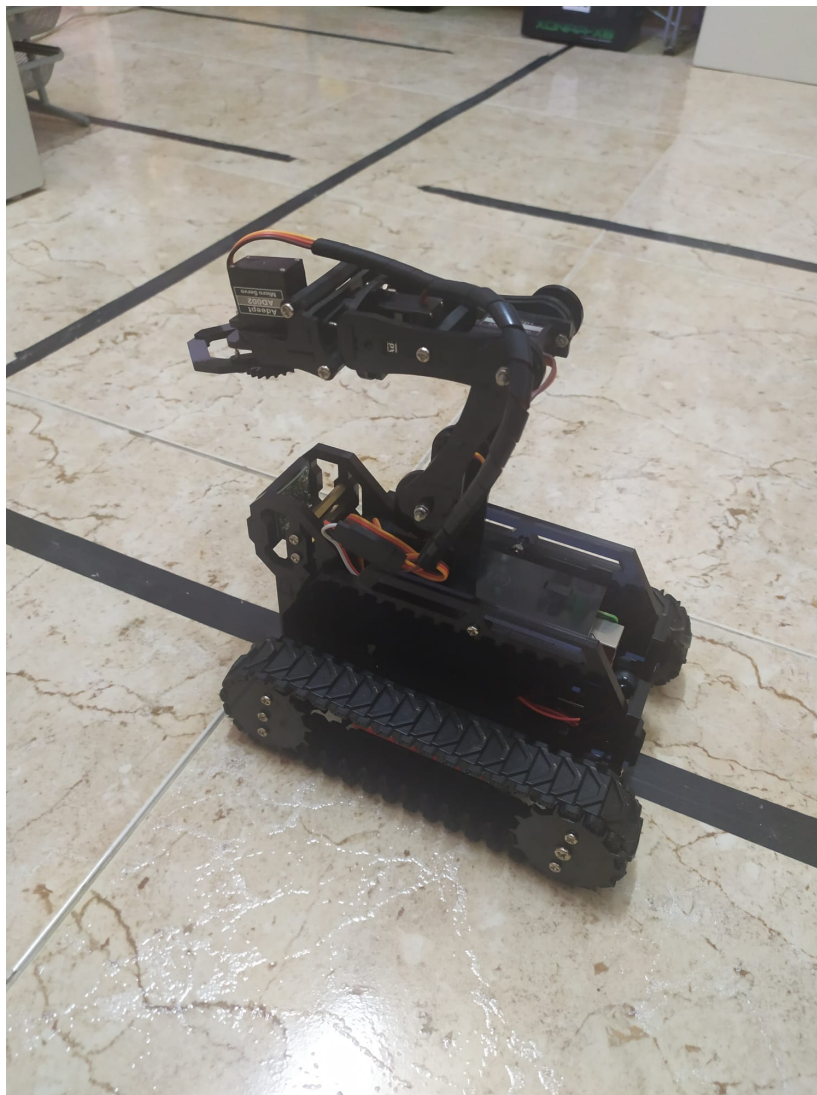


Figura 6.19: Robot sobre una línea del circuito.

6.7 ANÁLISIS Y VALORACIÓN DEL PROYECTO

6.7.1 Funcionalidad y posibles aplicaciones

Este apartado se centra en mostrar las utilidades y aplicaciones reales que se buscan con el desarrollo de este proyecto.

Mediante la programación desarrollada en este prototipo, se pretende simular un robot industrial que trabaje a gran escala en algún tipo de fábrica, almacén o instalación estructurada en la que pueda desempeñar su función.

De modo que el robot pueda trabajar de forma independiente y automática una vez se programe en un escenario específico de funcionamiento.

Este prototipo se ha desarrollado para que pueda funcionar usando unos sensores muy básicos y de bajo coste, que consigue una robustez ante fallos bastante alta.

Su algoritmo de programación le permite interpretar un mapa automáticamente según lo vaya recorriendo, sin tener que aprenderse la zona de trabajo previamente, según se detalla el funcionamiento en el apartado (6.5.1). Esto le hace un robot extremadamente versátil para una adaptación a nuevas zonas de trabajo.

Su método de navegación utilizando como recursos una línea en la superficie del suelo y unos códigos QR para ir guiando al robot, hace que sea una forma muy económica de generar un espacio de trabajo.

De esta forma los códigos intercambiables entre sí para la determinación de distintos órdenes para el robot móvil.

Agrupando los 3 últimos puntos descritos en el apartado anterior (6.6.1, 6.6.2 y 6.6.3) se describe el funcionamiento completo del prototipo desarrollado, para ello se ha construido un circuito en un entorno interior (Figura 6.20) en el cual se han realizado todas las pruebas de funcionamiento.

6.7.2 Impacto medioambiental

Respecto al impacto ambiental de este proyecto caben destacar distintos aspectos beneficiosos para el planeta como son el ahorro de agua y combustible fósil. Esto se debe a que un robot de consumo eléctrico como el del proyecto en cuestión permite aminorar el combustible y mantenimiento de muchas máquinas industriales convencionales a las que puede sustituir.

Además, la integración de este robot a gran escala en empresas implicaría la sustitución de mucha mano de obra humana, lo cual evitaría la contaminación por transporte y gestiones de los trabajadores.

6.7.3 Eficiencia energética

En términos de energía, el consumo y el uso eficiente de las máquinas es algo esencial. De este modo se logra contribuir con la disminución de costos generales y de contaminación al medio ambiente.

Para ello es importante revisar periódicamente algunas piezas claves de este sector como pueden ser las fuentes de energía de las máquinas, la climatización de las máquinas o de las instalaciones donde trabajan, uso de iluminación led de bajo consumo y el mantenimiento y revisión de software y hardware de las máquinas [74].

7 ECONOMÍA Y VIABILIDAD DEL PROYECTO

En este capítulo se detalla el cálculo del presupuesto del proyecto en cuestión. Se opta por dividir los costes totales estimados entre costes de recursos humanos y costes por materiales necesarios.

7.1 Presupuesto de recursos humanos

Se ha calculado el salario medio de un ingeniero medio en España [12] obteniéndose un salario bruto por año de 37900€, lo que supone 28337.8€ netos anuales, calculando un total de 13.11€ por hora aproximadamente.

Se tiene en cuenta las horas empleadas por el tutor en este trabajo en un total de 25€ por hora.

Coste recursos humanos			
Descripción	Horas	€/hora	Total (€)
Investigación	127	13.11	1664.7
Desarrollo del proyecto y algoritmos	63	13.11	825.93
Experimentación y resultados	54	13.11	707.94
Redacción de la memoria	50	13.11	655.5
Desarrollo de la defensa	20	13.11	262.2
TOTAL ALUMNO	314 h		4116.27 €
Videoconferencias y reuniones	3	25	75
Revisión memoria	4	25	100
TOTAL TUTOR	7 h		175 €
TOTAL	321 horas		4291.27 €

Tabla 6.7.1. Costes de recursos humanos

7.2 Presupuesto de materiales

En la siguiente tabla se reflejan los gastos utilizados en materiales para el desarrollo del proyecto.

Coste materiales necesarios			
Descripción	Cantidad	€/Unidad	Total (€)
Robot Rasptank + Sensores + Cámara Raspberry	1	105.99	105.99
Baterías Litio-Ion	4	8.50	34
Cargador baterías	15.89	1	15.89
Papel para impresión de códigos	1	0.50	0.50
Raspberry Pi 4 Model B	84.99	1	84.99
Tarjeta micro SD	1	9.90	9.90
Cinta aislante	4	0.55	2.2
Coste uso del ordenador	290 horas	0.0105	3.06
Cuota internet	290 horas	0.042	12.18
Coste de gestiones	-	50	50
TOTAL			318.71 €

Tabla 6.7.2. Costes de materiales necesarios

7.3 Presupuesto final del proyecto

Se recogen los gastos totales aplicados en el proyecto en la siguiente tabla (Tabla 6.7.3) aplicando el coste del I.V.A.

Coste del proyecto	
Descripción	Total
Costes recursos humanos	4291.27 €
Costes de materiales necesarios	318.71 €
Coste total sin I.V.A.	4609.98 €
Coste total con I.V.A. (21%)	5578.08 €

Tabla 6.7.3. Costes totales del proyecto

Finalmente, el coste del proyecto asciende a un total de cinco mil quinientos setenta y ocho euros con ocho céntimos (5578.08 €)

8 CONCLUSIONES

A lo largo de este capítulo se evalúa el trabajo realizado y se extraen las conclusiones pertinentes. Se repasa el trabajo realizado analizando los resultados.

En primer lugar, el montaje físico del prototipo, ha sido un trabajo lento y preciso, a pesar de contar con el manual de instalación hardware y software del producto base [6] sobre el que se ha trabajado.

La separación de las piezas de metacrilato que conforman el chasis del robot una a una ha sido una tarea ardua, que continuaba con el ajuste del brazo antropomórfico.

Seguido del montaje de todos los engranajes, motores y servomotores, que finalmente se conectaron todos a la Raspberry junto con el Motor HAT y todos los sensores y módulos led.

Posteriormente, la tarea de instalación de todo el software necesario para el funcionamiento, que ha sido algo más fácil, imprescindible para poder comenzar a entender el funcionamiento y realizar la ingeniería inversa a todo el código del robot para poder identificarlo y modificarlo.

El aprendizaje de un nuevo lenguaje de programación desde cero, Python, ha llevado el proyecto a un nivel de investigación enorme, ya que para la creación de todos los nuevos scripts y funciones se debe entender cómo funciona todo el código.

Una vez se empieza a crear y editar scripts, con el conocimiento adquirido, llega el momento de prueba y error donde se realiza la creación de nuevas funciones y la depuración del código para un funcionamiento perfecto.

En el código se consigue automatizar el proceso en el que el robot comienza, una vez se ejecuta el programa, a seguir la línea que tiene en su parte inferior hasta encontrar un obstáculo o una baliza que le haga actuar. Este comportamiento se detalla aún más en el apartado de Diseño, codificación y programación de balizas y rutas (6.5.1).

El robot circula por un circuito de pruebas mientras realiza el proceso automatizado, saltando entre diferentes posiciones. De tal forma que su funcionamiento se basa en un grafo.

El grafo siempre se puede extender aludiendo más nodos y aristas, siempre que se desee y el robot se mantenga alimentado de las baterías para poder recorrerlo.

Este prototipo demuestra que es posible aplicar una visión artificial sencilla y de bajo costo a muchas aplicaciones industriales reales para robots móviles.

Además, el uso de balizas codificadas (códigos QR) permite una gran flexibilidad en las tareas, navegación, decisiones, etc.

9 LINEAS FUTURAS

En este apartado se citan algunas ideas de continuación del proyecto o adaptaciones como son el desarrollo de un potente HMI que permita tele operar el robot de forma remota, pudiendo activar o desactivar los procesos desarrollados en este proceso mediante una interfaz visual atractiva. (Figura 9.1)



Figura 9.1

Una idea para un desarrollo distinto es aprovechar la teoría de grafos generada en este proyecto cambiando el tipo de balizas con un hardware añadido al robot, por ejemplo, adaptando un sistema de reconocimiento RFID o NFC que permita a la maquina leer etiquetas de esta tecnología en el entorno y así poder crear un grafo distinto adaptado a esta tecnología análogo a los nodos de códigos QR. De tal forma el robot también puede utilizar las mismas aristas con distintos nodos en el mismo grafo. (Ver Figura 9.2)

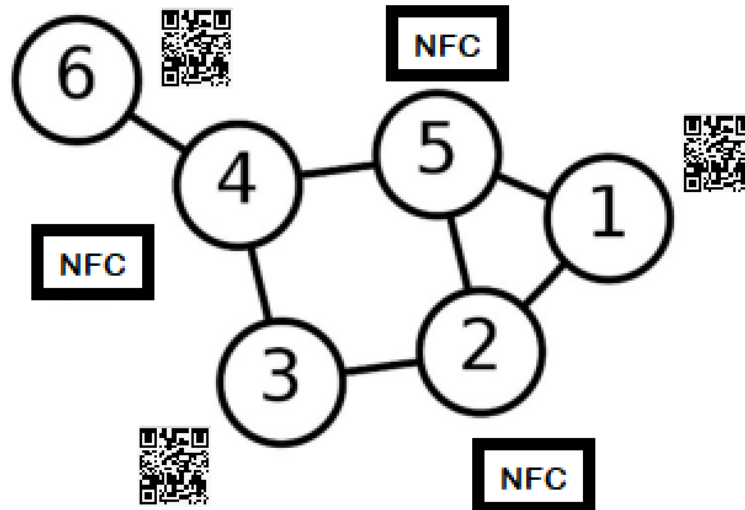


Figura 9.2

También se le pueden añadir más cámaras al robot para aumentar su capacidad de visión, o nuevos módulos o zumbadores para avisar de sus acciones o trayectorias.

Dentro de las posibles mejoras a implementar en el proyecto se comprenden algunos aspectos interesantes como son:

- Programación de detección de línea en superficies irregulares o con planos inclinados.
- Seguimiento de códigos QR dinámicos en movimiento, mediante el movimiento de la cámara con ayuda del servomotor de la misma.
- Preparación del robot para entornos exteriores o de climatología adversa.
- Implementación de un modo exploratorio, realizando una trayectoria en espiral en caso de pérdida de trayectoria
- Programación de HMI para tele operación en modo manual o automático con visión del robot en tiempo real.
- Implementación de nuevos módulos al robot como pueden ser cámaras extra, zumbadores o displays para obtener/mostrar información.

Entre otras opciones, también se encuentran:

- Multiplicar el mismo robot para crear una línea de montaje o línea de cooperación entre robots con la misma función y características, de tal modo que tengan la posibilidad de intercambiar mercancía física entre ellos o incluso entablar una comunicación software que les permita compartir estados de nodos o banderas de variables para el funcionamiento de la línea de robots en conjunto.
- Implementación en sanidad: Es posible la implementación de este robot en salas blancas sanitarias, salas de operaciones, hospitales o en manipulación de material sanitario frágil y delicado.



Figura 9.3: Carro robotizado utilizado en sanidad.

- Implementación en manipulación de mercancías comestibles o de transporte de alimentos: Otra idea es la manipulación o transporte de alimentos de forma limpia y segura.

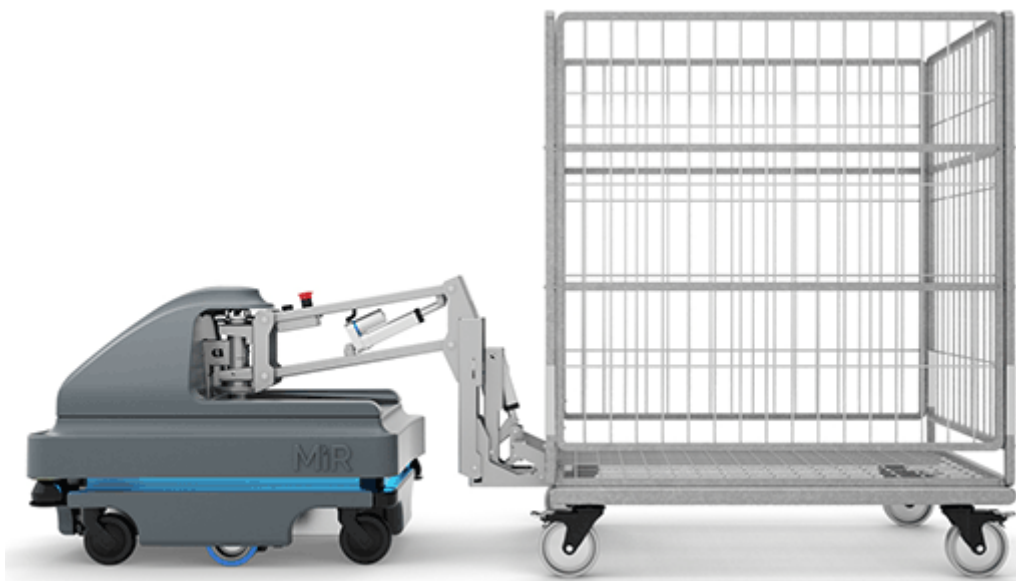
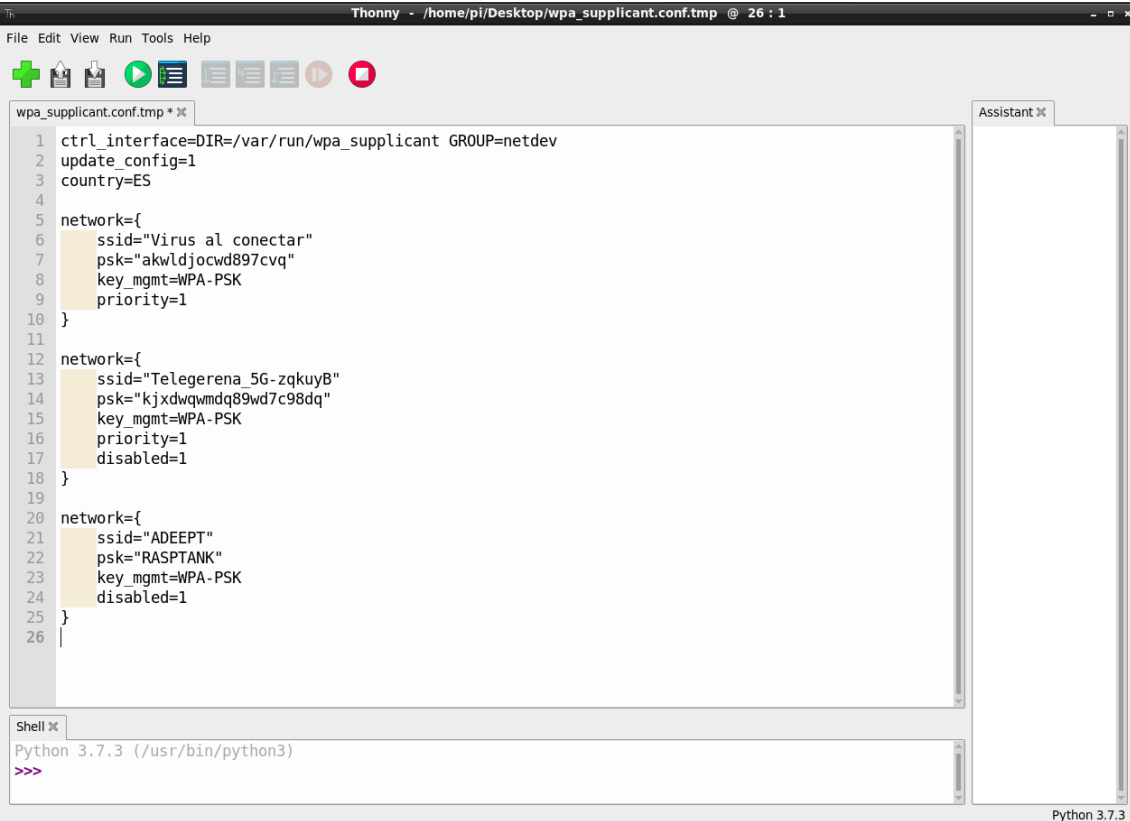


Figura 9.4: Carro robotizado utilizado para transporte de mercancías.

10 ANEXOS

ANEXO 1: Archivo de configuración de conexión WIFI, alojado en la carpeta raíz del SO Raspbian (en la tarjeta micro SD).



The image shows a screenshot of the Thonny IDE interface. The main window displays a file named 'wpa_supplicant.conf.tmp' with the following configuration:

```
1 ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
2 update_config=1
3 country=ES
4
5 network={
6     ssid="Virus al conectar"
7     psk="akwldjocwd897cvq"
8     key_mgmt=WPA-PSK
9     priority=1
10 }
11
12 network={
13     ssid="Telegerena_5G-zqkuyB"
14     psk="kjxdwqwdq89wd7c98dq"
15     key_mgmt=WPA-PSK
16     priority=1
17     disabled=1
18 }
19
20 network={
21     ssid="ADEEPT"
22     psk="RASPTANK"
23     key_mgmt=WPA-PSK
24     disabled=1
25 }
26 |
```

Below the editor, there is a 'Shell' window showing the Python 3.7.3 prompt:

```
Python 3.7.3 (/usr/bin/python3)
>>>
```

The status bar at the bottom right indicates 'Python 3.7.3'.

ANEXO 2: Script de funcionamiento del sensor de seguimiento de línea del robot.

```

160
161     def trackLineProcessing(self):
162         status_right = GPIO.input(line_pin_right)
163         status_middle = GPIO.input(line_pin_middle)
164         status_left = GPIO.input(line_pin_left)
165
166         if status_middle == 1:
167             move.move(22, 'forward', 'no', 1) #22 por defecto
168         elif status_left == 1:
169             move.move(32, 'no', 'right', 1)
170         elif status_right == 1:
171             move.move(32, 'no', 'left', 1)
172         else:
173             move.move(22, 'forward', 'no', 1) #22 por defecto
174
175         time.sleep(0.05)
176

```

ANEXO 3: Detección de rostros faciales.

```

1  import cv2
2
3  #Se cargan los clasificadores requeridos
4  face_cascade = cv2.CascadeClassifier('C:\Program Files\Lib\site-packages\cv2\data\haarcascade_frontalface_alt.xml')
5
6  #Se utiliza un fichero de imagen para la búsqueda de caras
7  img = cv2.imread('d:\cara.jpg')
8
9  while(True):
10     #Se convierte la imagen a blanco y negro
11     gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
12     #Se buscan las coordenadas de los rostros
13     caras = face_cascade.detectMultiScale(gray, 1.3, 5)
14     #Se dibuja un rectángulo en las coordenadas de cada rostro
15     numCaras = 0
16     for (x,y,w,h) in caras:
17         cv2.rectangle(img, (x,y), (x+w,y+h), (125,255,0), 2)
18         numCaras = numCaras + 1
19     #Se muestra la imagen
20     cv2.imshow('img',img)
21     #Pulsando la tecla "q" salimos del programa
22     if cv2.waitKey(1) & 0xFF == ord('q'):
23         break
24
25 print ("Número de caras detectadas: {}".format(numCaras))

```

Anexo 4: Detección de objetos

```
1 import cv2
2 from matplotlib import pyplot as plt
3
4 img = cv2.imread("image.jpg")
5
6 img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
7 img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
8
9
10 stop_data = cv2.CascadeClassifier('stop_data.xml')
11
12 found = stop_data.detectMultiScale(img_gray,
13                                   minSize=(20, 20))
14
15 amount_found = len(found)
16
17 if amount_found != 0:
18
19
20     for (x, y, width, height) in found:
21
22
23
24
25         cv2.rectangle(img_rgb,(x, y),
26                       (x + height, y + width),
27                       (0, 255, 0), 5)
28
29 plt.subplot(1, 1, 1)
30 plt.imshow(img_rgb)
31 plt.show()
```

Anexo 5: Funcionamiento interno del sensor de ultrasonidos. [6]

```
1 import RPi.GPIO as GPIO
2 import time
3
4 Tr = 11
5 Ec = 8
6
7 GPIO.setwarnings(False)
8 GPIO.setmode(GPIO.BCM)
9 GPIO.setup(Tr, GPIO.OUT,initial=GPIO.LOW)
10 GPIO.setup(Ec, GPIO.IN)
11
12 def checkdist(): #Reading distance
13     for i in range(5): # Remove invalid test results.
14         GPIO.setwarnings(False)
15         GPIO.setmode(GPIO.BCM)
16         GPIO.setup(Tr, GPIO.OUT,initial=GPIO.LOW)
17         GPIO.setup(Ec, GPIO.IN)
18         GPIO.output(Tr, GPIO.LOW)
19         time.sleep(0.000002)
20         GPIO.output(Tr, GPIO.HIGH)
21         time.sleep(0.000015)
22         GPIO.output(Tr, GPIO.LOW)
23         while not GPIO.input(Ec):
24             pass
25         t1 = time.time()
26         while GPIO.input(Ec):
27             pass
28         t2 = time.time()
29         dist = (t2-t1)*340/2
30         if dist > 9 and i < 4: # 5 consecutive times are invalid data, return the last test data
31             continue
32         else:
33             return (t2-t1)*340/2
34
35
36 if __name__ == '__main__':
37     while 1:
38         print(checkdist())
39         time.sleep(1)
```

ANEXO 6: Funcionamiento del proceso automático del robot.

```

446 @staticmethod
447 def set_video_source(source):
448     Camera.video_source = source
449
450 @staticmethod
451 def frames():
452     camera = cv2.VideoCapture(Camera.video_source)
453     if not camera.isOpened():
454         raise RuntimeError('Could not start camera.')
455
456     cvt = CVThread() #Se inician los subprocessos del robot.
457     cvt.start()
458     diccionario.clear() #Se limpia la lista de códigos QR almacenados en el diccionario.
459     fuc = Functions()
460
461     global flagprocessing #Variable de tipo bandera, (0 = continua la marcha, 1 = se detiene el robot, 2 = proceso finalizado)
462     while True:
463         _, img = camera.read() #Se obtiene el frame actual de la cámara.
464
465         detections = decode(img, symbols=[ZBarSymbol.QRCODE]) #Se procesa el frame obtenido para obtener códigos QR.
466
467         dist_ultra = ultra.checkdist() #Se lee el valor actual del sensor de ultrasonidos.
468
469         if(detections != 0): #Continuar si hay una imagen procesada.
470             for barcode in detections: #Buscar si hay algun QR en la imagen procesada.
471                 myData = barcode.data.decode('utf-8')
472                 if ((myData in frontList) or (myData in rightList) or (myData in leftList)): #Comprobar a qué lista pertenece el QR detectado.
473                     if(diccionario.get(myData) == None): #Comprobar si el código QR está añadido en el diccionario de códigos.
474                         diccionario.setdefault(myData, '0') #Si no está registrado, se añade al diccionario con valor "0". (Accion de QR no finalizada).
475                     if(diccionario.setdefault(myData) == '0'): #Si la acción del QR no está finalizada se procede a hacer los chequeos posicionamiento del robot.
476                         if (dist_ultra > 0.2): #El sensor de ultrasonidos detecta que el robot está lejos del QR objetivo.
477                             flagprocessing = 1 #El robot continua la marcha.
478                             fuc.trackLineProcessing() #El robot avanza hacia el QR objetivo usando el sensor de línea.
479
480                         elif (dist_ultra <= 0.6): #El robot se detiene cuando el sensor de ultrasonidos detecta que
481                             move.motorStop() #está cerca del QR objetivo.
482                             time.sleep(0.5)
483
484                         #Maniobras del robot durante la marcha dependiendo del codigo QR que detecte.
485                         if(myData in leftList):
486                             position.secRobot_8() #Secuencia 8 (Giro a la izquierda).
487                         if(myData in rightList):
488                             position.secRobot_7() #Secuencia 7 (Giro a la derecha).
489                         elif(myData in frontList):
490                             position.secRobotfinal() #Secuencia final (Maniobra de celebración).
491                             move.motorStop()
492                             flagprocessing = 2
493                             print("STOP")
494                             break #Se termina el proceso.
495
496                         diccionario.pop(myData) #Se almacena como completada la acción en el diccionario
497                         diccionario.setdefault(myData, '1')
498                         time.sleep(0.3)
499                         flagprocessing = 0 #El robot sigue su ruta
500
501     if(flagprocessing == 0): #El robot por defecto se desplaza hacia adelante usando el sensor de línea como guía.
502         fuc.trackLineProcessing() #
503         if(dist_ultra <= 0.1): #En el caso de que el robot se encuentre un obstáculo, detendrá su marcha.
504             print("STOP obstaculo")
505             move.motorStop()
506             flagprocessing = 1
507

```

ANEXO 7: Secuencias de maniobras del robot.

```

1  from __future__ import division
2  import RPi.GPIO as GPIO
3  import sys
4  import Adafruit_PCA9685
5  import move
6  import servo
7  import time
8  import QR
9  import ultra
10 from pyzbar.pyzbar import decode
11 import camera_opencv
12
13 pwm = Adafruit_PCA9685.PCA9685()
14
15 with open('/home/pi/adeept_rasptank/server/leftList.txt') as list0:
16     leftList = list0.read().splitlines()
17
18 with open('/home/pi/adeept_rasptank/server/rightList.txt') as list1:
19     rightList = list1.read().splitlines()
20
21 with open('/home/pi/adeept_rasptank/server/frontList.txt') as list2:
22     frontList = list2.read().splitlines()
23
24 def secRobot_0(): #Parado
25     move.move(0, 'forward', 'no', 0.5)
26     time.sleep(0.5)
27     move.motorStop()
28
29 def secRobot_1(): #Atras
30     move.move(60, 'backward', 'no', 0.5)
31     time.sleep(0.05)
32     move.motorStop()
33
34 def secRobot_2(): #Atras derecha
35     move.move(60, 'backward', 'right', 0.5)
36     time.sleep(0.05)
37     move.motorStop()
38
39 def secRobot_3(): #Atras izquierda
40     move.move(60, 'backward', 'left', 0.5)
41     time.sleep(0.05)
42     move.motorStop()
43
44 def secRobot_4(): #Delante
45     move.move(60, 'forward', 'no', 0.5)
46     time.sleep(0.05)
47     move.motorStop()
48
49 def secRobot_5(): #Delante derecha
50     move.move(60, 'forward', 'right', 1)
51     time.sleep(0.15)
52     move.motorStop()
53
54 def secRobot_6(): #Delante izquierda
55     move.move(80, 'forward', 'left', 0.5)
56     time.sleep(1)
57     move.motorStop()
58
59 def secRobot_7(): #Giro derecha
60     move.move(80, 'no', 'right', 1)
61     time.sleep(0.7)
62     move.motorStop()
63
64 def secRobot_8(): #Giro izquierda

```

```

64 def secRobot_8(): #Giro izquierda
65     move.move(80,'no','left',1)
66     time.sleep(0.758)
67     move.motorStop()
68
69 def secRobotfinal(): #Maniobra final
70     secRobot_7()
71     time.sleep(0.5)
72     openclamp()
73     armshake()
74
75 def openclamp(): #Abre pinza
76     pwm.set_pwm(15, 0, 200)
77
78 def armshake(): #Agitación del brazo del robot
79
80
81     pwm.set_pwm(12, 0, 380)
82     pwm.set_pwm(13, 0, 380)
83     time.sleep(0.5)
84     pwm.set_pwm(13, 0, 250)
85     time.sleep(0.5)
86     pwm.set_pwm(13, 0, 380)
87     time.sleep(0.5)
88     pwm.set_pwm(13, 0, 250)

```

Anexo 8: Listas de valores de códigos QR utilizados.

leftList.text ✕		rightList.text ✕		frontList.text ✕	
1	111111	1	211111	1	311111
2	111112	2	211112	2	311112
3	111113	3	211113	3	311113
4	111114	4	211114	4	311114
5	111115	5	211115	5	311115
6	111116	6	211116	6	311116
7	111117	7	211117	7	311117
8	111118	8	211118	8	311118
9	111119	9	211119	9	311119
10	111120	10	211120	10	311120
11	111121	11	211121	11	311121
12	111122	12	211122	12	311122
13	111123	13	211123	13	311123
14	111124	14	211124	14	311124
15	111125	15	211125	15	311125

11 REFERENCIAS BIBLIOGRAFICAS

[1] HISTORIA DE ROBOTS

Fernando Reyes Cortés. (2011). *Robótica: Control de Robots Manipuladores*. Gran Vía de las Corts Catalanes, 594, Barcelona: MARCOMBO S.A.

[2] DESARROLLO HISTORICO Y EVOLUCION DE LA ROBOTICA

José Andrés Somolinos Sánchez. (2002). *Avances en robótica y visión por computador*. Cuenca, Castilla la Mancha: Universidad de Castilla la Mancha.

[3] BATERIAS 18650 3.7V

https://www.google.com/search?q=baterias+18650&rlz=1C1CHBF_esES912ES912&sxsrf=ALiCzsYA6AUYG3FiHceBi811slkcYC7haQ:1656094785746&source=inms&tbm=isch&sa=X&ved=2ahUKEwjwsqzr2cb4AhXL4YUKHUpWBqEQ_AUoAnoECAEQBA&biw=1920&bih=937&dpr=1#imgrc=-uBwHCcxVtH6HM

[4] INSTALACION PUTTY

<https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html>

[5] INFORMACION PUTTY

<https://the.earth.li/~sgtatham/putty/0.77/html/doc/>

[6] PROYECTO BASE RASPTANK

https://github.com/adeept/adeept_rasptank

[7] INSTALACION PYTHON 3.7

<https://www.python.org/downloads/>

[8] INSTALACION MOBAXTERM

<https://mobaxterm.mobatek.net/download.html>

[9] INFORMACION MOBAXTERM

<https://mobaxterm.mobatek.net/documentation.html>

[10] OPENCV

<https://opencv.org/>

[11] ArUco

https://docs.opencv.org/master/d5/dae/tutorial_aruco_detection.html

[12] SALARIO INGENIERIA

<https://www.jobted.es/salario/ingeniero>

[13] EFICIENCIA ENERGETICA EN ROBOTS INDUSTRIALES

Angie Valle. (2017). *EFICIENCIA ENERGETICA EN ROBOTS INDUSTRIALES*. Junio 2022, de Ferroindustrial Sitio web: <https://fierrosindustrial.com/noticias/eficiencia-energetica-robots-industriales/>

[14] IMAGEN HMI

Bailey Hudson. (2021). *Robot HMI for Projects Automation*. Junio 2022, de Roboticstomorrow Sitio web: <https://www.roboticstomorrow.com/story/2021/03/robot-hmi-for-projects-automation/16462/>

[15] ROBOT UNIMATE

Lisa Nocks. (2007). *The Robot: The Life Story of a Technology*. London: GREENWOOD TECHNOGRAPHIES.

[16] ROBOT SCARA

Karl Mathia. (2010). *Robotics for Electronics Manufacturing: Principles and Applications in cleanroom automation*. New York: Cambridge University Press.

[17] ROBOT ANTROPOMORFICO

Rafael Iñigo Madrigal, Enric Vidal Idiarte. (2002). *Robots industriales manipuladores*. Universitat Politècnica de Catalunya, Barcelona: EDICIONS UPC.

[18] ROBOT CARTESIANO

Roger Miranda Colorado. (2016). *Cinemática y dinámica de robots manipuladores*. México: Alfaomega Grupo Editor S.A.

[19] ROBOT PARALELO

Miguel Díaz-Rodríguez, Héctor Fabio Quintero-Riaza, Luis Adriana Mejía-Calderón, Germán Holguín, Marlon Herrera-López. *Aplicación de los Robots Paralelos. Manipuladores Paralelos: Síntesis, Análisis y Aplicaciones*, 2018.

[20] TIPOS DE ROBOTS

Sicma21. (2021). *Robots industriales: tecnología y aplicaciones*. Junio 2022, de Sicma21 Sitio web: <https://www.sicma21.com/robots-industriales-tecnologia-y-aplicaciones/>

[21] B. D. Argall, B. Browning, and M. M. Veloso, "Teacher feedback to scaffold and refine demonstrated motion primitives on a mobile robot," *Robotics and Autonomous Systems*, vol. 59, pp. 243-255, 2011.

[22] M. R. Pedersen, C. Hoilund, and V. Kruger, "Using human gestures and generic skills to instruct a mobile robot arm in a feeder filling scenario," presented at the *International Conference on Mechatronics and Automation (ICMA)*, 2012.

[23] S. Rosenthal and M. Veloso, "Mobile Robot Planning to Seek Help with Spatially-Situated Tasks," presented at the *Twenty-Sixth AAAI Conference on Artificial Intelligence (AAAI-12)*, Toronto, Canada, 2012.

[24] E. A. Sisbot, L. F. Marin-Urias, R. Alami, and T. Simeon, "A Human Aware Mobile Robot Motion Planner," *Robotics, IEEE Transactions on*, vol. 23, pp. 874-883, 2007.

[25] B. Graf, U. Reiser, M. Hägele, K. Mauz, and P. Klein, "Robotic home assistant Care-O-bot® 3 - product vision and innovation platform," in *Advanced Robotics and its Social Impacts (ARSO)*, 2009 *IEEE Workshop on*, 2009, pp. 139-144.

[26] R. R. Rajkumar, I. Lee, L. Sha, and J. Stankovic, "Cyber-physical systems: the next computing revolution", in *Proc. of the 47th Design Autom. Conf.*, Anaheim, CA, USA, 2010, pp. 731–736 (doi: 10.1145/1837274.1837461)

[27] Y. Ashibani and Q. H. Mahmoud, "Cyber physical systems security: Analysis, challenges and solutions", *Computers & Secur.*, vol. 68, pp. 81–97, 2017 (doi: 10.1016/j.cose.2017.04.005).

- [28] A. Humayed, J. Lin, F. Li, and B. Luo, "Cyber-physical systems security – a survey" *IEEE Internet of Things J.*, vol. 4, no. 6, pp. 1802–1831, 2017 (doi: 10.1109/JIOT.2017.2703172).
- [29] C. W. Axelrod, "Managing the risks of cyber-physical systems", in *Proc. IEEE Long Island Syst., Appl. and Technol. Conf. LISAT 2013*, Farmingdale, NY, USA, 2013 (doi: 10.1109/LISAT.2013.6578215).
- [30] Miguel Díaz-Rodríguez, Hector Fabio Quintero-Riaza, Luis Adriana Mejía-Calderón, Germán Holguin, Marlon Herrera-López, et al.. *Aplicación de los Robots Paralelos. Manipuladores Paralelos: Síntesis, Análisis y Aplicaciones*, 2018. hal-01907282
- [31] Jobses, Christopher. (1988). *ROBOT PROGRAMMING*. 10.13140/RG.2.1.4270.7368.
- [32] *Operating Manual for T3 Industrial Robot , Version 3.0. Cincinnati Millicron*, 1980.
- [33] *Robotics Control, Sensing, Vision, and Intelligence*
K.S. Fu. R. C Gonzalez, C.S.G. Lee
Publicado por McGraw Hill Book Company, 1987
ISBN 10 0070226253 ISBN 13 9780070226258
- [34] Gilbert, A., G. Pelton, R. Wang. and S. Motiwalla,. *ARBASIC: An Advanced and User-friendly Programming System for Robots*. *SME Robots-8 Conference, Detroit , MI, June 1984*.
- [35] *Seiko D-TRAN Robotics Manual. Seiko Instruments USA Inc . November 1983*.
- [36] Salmon, M. *SIGLA - The Olivetti SIGMA Robot Programming Language. Paper in Proceedings of 8th International Symposium on Industrial Robots, Stuttgart, 1978 .*
- [37] *User's Guide to VAL, Version 12. Unimation Inc. , 398-H2A, June 1980*.
- [38] *VAL Univision Supplement. Version 13 (VSN). 2nd Ed. , Unimation Inc., July 1981*.
- [39] Shimano, B. E. *VAL - A Versatile Robot Programming and Control System. Paper in Proceedings IEEE Computing Society 3rd International Computer Software Applications Conference, Chicago, IL, 1979*.
- [40] *AL User's Manual. Stanford University, Third Ed., 1981 .*
- [41] *IBM Robot System/I: AML Reference Manual . IBM Corporation. 1982*.
- [42] *Technical reference manual RAPID Instructions, Functions and Data types RobotWare 5.13 Document ID: 3HAC 16581-1*
- [43] *Allegro Operator 's Manual (A12 Assembly Robot). General Electric Co., 1982*.
- [44] *JARS User's Guide and Manual. 1980 .*

- [45] *Robotic System for Aerospace Batch Manufacturing, Task B High Level Language User's Manual. Wright Patterson Air Force Base, 1981.*
- [46] *RAIL User's Manual. Automatix Inc . 1982.*
- [47] *User's Guide to VAL-II. Part I: Control from the System Terminal, Version X2. Unimation Inc . April 1983.*
- [48] *User's Guide to VAL-II. Part II: Communications with a Supervisory System, Version X2. Unimation Inc. April 1983.*
- [49] *SOFTWARE KRC1/KRC2/KRC3 Reference Guide Release 4.1*
- [50] *Lenguajes de programación de robots industriales. Una perspectiva histórica: del Control Numérico a los frameworks robóticos, Versión 1.0 Diciembre 2018, Fernando Comins Tello*
- [51] *Introducción a la visión artificial, Una guía para la automatización de procesos y mejorar la calidad, COGNEX*
- [52] *SOMOLINOS SÁNCHEZ, José Andrés. (2002). Avances en robótica y visión por computador. Cuenca, Castilla la Mancha: Universidad de Castilla la Mancha.*
- [53] *VEHÍCULOS DE GUIADO AUTÓNOMO (AGV) EN APLICACIONES INDUSTRIALES: UNA REVISIÓN, Revista Politécnica, vol. 15, núm. 28, 2019, junio, pp. 117-137*
- [54] *NAVEGACION DE ROBOTS MÓVILES EN ENTORNOS CON DISCONTINUIDADES: UNA REVISIÓN, Revista Politécnica ISSN 1900-2351 (Impreso), 2256-5353 (En línea), Año 14, Volumen 14, Número 27, Páginas 103-115, Julio – Diciembre 2018*
- [55] *Murphy, R.R. Introduction to AI Robotics. Editorial MIT Press. ISBN: 0-262-13383-0, 2000.*
- [56] *Sales, Daniel Oliva; Shinzato, Patrick; Pessin, Gustavo; Osório, Fernando S; Wolf, Denis F, Vision – Based Autonomous Navigation System Using ANN and FSM Control. In: Proceedings of the IEEE LARS/EnRI, São Bernardo do Campo, SP. IEEE Society Press. v.1. pag 85 – 90, 2011*
- [57] *González Sánchez, Ramón et al, “Algoritmo de navegación reactiva de robots móviles para tareas bajo invernadero”, XXVII Jornadas de Automática, ISBN: 84-689-9417-0, Almería, España, 2006*
- [58] *Santos, Victor., Sandini, G., Curotto, F., Garibaldi, S., Divergent stereo in autonomous navigation: from bees to robots, International Journal of Computer Vision 14 159–177. 1995*
- [59] *Isard, Michael and Blake, Andrew. Contour tracking by stochastic propagation of conditional density. In Proc. Europea Conf. Computer Vision 1996 (AAAI'98), pages 343– 356. Springer Verlag, 1996.*
- [60] *Dev, A., Kröse, B., Groen, F., Navigation of a mobile robot on the temporal development of the optic flow, in: Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robots and Systems IROS'97, Grenoble, págs. 558–563, September 1997.*

- [61] Calisi, Daniele, *Mobile robots and vehicles motion systems: a unifying framework*, Tesis Doctoral, Universidad de Roma, Italia, 2009.
- [62] DOCUMENTACIÓN RASPBERRY PI
<https://www.raspberrypi.com/documentation/computers/raspberry-pi.html>
- [63] RASPBIAN SO
<https://www.raspberrypi.com/software/operating-systems/>
- [64] CÁMARA RASPBERRY
<https://www.raspberrypi.com/documentation/accessories/camera.html>
- [65] Reconocimiento de objetos a través de la metodología Haar Cascades, Ezequiel Ángel Jeremías Ambrogio
- [66] <https://es.acervolima.com/detectar-un-objeto-con-opencv-python/>
- [67] Satya Mallick. (2016). *Homography examples using OpenCV (Python / C ++)*. Junio 2022, de learnopencv Sitio web:
<https://learnopencv.com/homography-examples-using-opencv-python-c/>
- [68] PRODUCTO RASPTANK
https://www.adeept.com/adeept-rasptank-wifi-wireless-smart-robot-car-kit-for-raspeberry-pi-4-3-model-b-b-tank-tracked-robot-with-4-dof-robotic-arm-opencv-target-tracking_p0121.html
- [69] SENSOR DE ULTRASONIDOS
Pedro Porcuna López. (2016). *Robótica y domótica básica con Arduino*. Bogotá, Colombia: Editorial Ra-ma.
- [70] VIDEO DE FUNCIONAMIENTO EN CIRCUITO DE PRUEBAS
<https://www.youtube.com/watch?v=01ALg10L-HY>