Universidad Católica de Santa María

Facultad de Ciencias e Ingenieras Físicas y Formales

Escuela Profesional de Ingeniería Mecánica, Mecánica Eléctrica y Mecatrónica

DISEÑO DEL SISTEMA DE CONTROL DE NAVEGACIÓN UTILIZANDO INTELIGENCIA ARTIFICIAL PARA UN CUADRÚPEDO DE RESCATE.

Tesis presentada por el Bachiller:
Montalvo Miranda, Giuliano Nikola

para optar el Título Profesional de
Ingeniero Mecatrónico

Asesor:
Mg. Mestas Ramos, Sergio Orlando

Arequipa - Perú

2021

UCSM-ERP

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

INGENIERIA MECANICA, MECANICA-ELECTRICA Y MECATRONICA

TITULACIÓN CON TESIS

DICTAMEN APROBACIÓN DE BORRADOR

Arequipa, 11 de Noviembre del 2021

Dictamen: 002040-C-EPIMMEM-2021

Visto el borrador del expediente 002040, presentado por:

2012802261 - MONTALVO MIRANDA GIULIANO NIKOLA

Titulado:

DISEÑO DEL SISTEMA DE CONTROL DE NAVEGACIÓN UTILIZANDO INTELIGENCIA ARTIFICIAL PARA UN CUADRÚPEDO DE RESCATE

Nuestro dictamen es:

APROBADO

1936 - MESTAS RAMOS SERGIO ORLANDO
DICTAMINADOR

2213 - QUISPE CCACHUCO MARCELO JAIME
DICTAMINADOR

2397 - CUADROS MACHUCA JUAN CARLOS
DICTAMINADOR

DEDICATORIA

Este trabajo se lo dedico a Dios y mis padres por los principios morales, vida, libertad bien común e igualdad que me brindan cada día con el fin de asumir y superar los retos por el bienestar propio y de la sociedad.

AGRADECIMIENTO

En primer lugar, agradezco a Dios por regalarme el don de la vida y guiar mis pasos a lo largo de esta, permitiéndome lograr mis objetivos personales y profesionales que me regalan la oportunidad de servir de mejor manera a nuestra sociedad.

A mis padres, Attilio y Magaly, por acompañarme y motivarme con cada paso que daba para lograr mis objetivos profesionales. Más aún, estando a mi lado en todo momento mostrándome su amor incondicional y total respaldo en cada proyecto que emprendido e impulsándome a continuar sin desfallecer, a pesar de las vicisitudes.

A mis abuelos, Atilio, Luisa, Teresa y Miguel, quienes siempre se han sentido orgullosos de mis logros y por quienes siempre me he sentido motivado a continuar con mis proyectos y regalarles ca uno de mis éxitos profesionales y personales.

A mis hermanos, Alfredo y Luisa, quienes siempre han velado por mi bienestar con sus oraciones y constante preocupación en mi crecimiento como persona, de igual forma, motivándome a tomar las mejores decisiones.

A mi ahijado Santiago, quien a pesar de su corta edad siempre me regala alegrías y motivaciones para continuar adelante, es quien ha sabido darme pequeñas lecciones de como a veces se necesita ser como un niño para afrontar la vida como un verdadero hombre.

A mis amigos y asesores, externos e internos, que siempre estuvieron motivándome a culminar mi tesis, siguiendo mis pasos, aconsejándome y regalándome de su tiempo cada vez que lo he necesitado.

Y mis dictaminadores, quienes a lo largo de mi formación universitaria no solo han logrado inculcarme conocimientos, sino en muchos casos me han brindado su amistad, consejos y principios que llevaré siempre en el corazón y en mi mente.

RESUMEN

En la presente investigación buscamos aportar a la mejora de los cuadrúpedos de rescate en cuanto a software se refiere, no solo ampliando, de esta manera, el campo de investigación con respecto a la inteligencia artificial y los softwares que se pueden desarrollar desde la aplicación de la misma, sino, además, ayudando al correcto desarrollo de nuestra nación. De manera que la presente tesis está enfocada en implementar un diseño del sistema de control de navegación utilizando inteligencia artificial para un cuadrúpedo de rescate.

El proyecto que a continuación presentaremos, plantea el diseño antes mencionado, implementando sistemas de Navigation Stack que nos brinde un soporte necesario en el buen manejo de nuestro software, ampliando los conocimientos en esta área y llevando a una mayor comprensión de la inteligencia artificial y sus múltiples facetas, así como abrir un panorama extenso en la investigación de la misma, como una herramienta que nos ayude a solucionar los problemas sociales, políticos y económicos de nuestra nación.

El proyecto consta de cinco capítulos, los cuales procederemos a describir:

El **Capítulo I Aspectos Generales** presenta el planteamiento de la base científica que nos permitirá desarrollar el proyecto propuesto, la cual inicia con el planteamiento, e identificación de la problemática a resolver; para luego dar paso a la metodología de investigación a desarrollar durante el presente trabajo. Pasamos a la justificación del problema, la cual nos permitirá tener un amplio panorama del mismo, en otras palabras, lograremos plantear una hipótesis con la cual trazaremos unos objetivos, alcances y limitaciones, y contribuciones de la tesis. Para concluir, de esta manera, con los antecedentes con mayor relación a la presente tesis.

El **Capítulo II Marco Teórico** compila de una manera condensada los diferentes argumentos teóricos que nos permitan relacionar y comprender de una manera más profunda la presente tesis, prueba de ello es que encontraremos en este capítulo argumentos que van desde lo más específico hasta lo más general, de ahí que tenemos conceptos de robots, microcomputadoras, lenguaje de programación, industria y

autonomía, diseño y culminando con conceptos referidos a los cuadrúpedos y el rescate para la comprensión de nuestro proyecto.

El **Capítulo III Análisis Inicial** abarca parte del análisis necesario a realizar, con la finalidad de poder obtener los parámetros necesarios a considerar al momento de poder realizar el diseño e implementación de nuestro sistema de navegación, con estos parámetros podremos, también, realizar una elección adecuada con respecto a los componentes necesarios a utilizar durante todo el desarrollo de nuestro proyecto.

El **Capítulo IV Diseño** engloba cada uno de los aspectos necesarios para lograr el objetivo principal de nuestro proyecto, la cual la desglosaremos en cuatro partes importantes que serán cada uno de los componentes necesarios en los que nos enfocaremos para luego al realizar el ensamble de todos estos componentes en uno solo podamos llegar a los resultados necesarios que nos permitan cumplir con los objetivos trasados en la presente tesis. Por lo tanto, en una primera parte desarrollamos un interfaz gráfico que nos permita interactuar con el robot y cargar los programas de manera inalámbrica, luego procederemos a realizar un programa para el escaneo de un código QR mediante una cámara, la cual nos ayudará a determinar la posición del robot; a partir de ello construiremos un programa que nos permita trazar el camino a seguir, para así, finalmente poder diseñar el sistema de movimientos que debe realizar nuestro robot para ejecutar las acciones requeridas.

El **Capítulo V Resultados Finales** nos evidencia la realización de cada una de las simulaciones ejecutadas para cada diseño elaborado en el capítulo anterior, demostrando la efectividad y los resultados necesarios que nos puedan llevar a una conclusión al final de la presente tesis.

Para acabar, comprobamos si los resultados obtenidos nos han ayudado a concretar con cada uno de los objetivos trazados al inicio de nuestro proyecto y si con la ayuda de los parámetros necesarios a considerar hemos logrado obtener los resultados deseados.

Palabras clave: inteligencia artificial, navegación, Navigation Stack, Python, Raspberry, planificador de trayectorias, rescate, cuadrúpedo, robótica, autonomía.

ABSTRACT

In this research we seek to contribute to the improvement of rescue quadrupeds in terms of software, not only expanding, in this way, the field of research with respect to artificial intelligence and software that can be developed from the application of the same, but also helping the proper development of our nation. So that the present thesis is focused on implementing a design of the navigation control system using artificial intelligence for a rescue quadruped.

The project that we will present below, proposes the aforementioned design, implementing Navigation Stack systems that will provide us with the necessary support in the proper management of our software, expanding the knowledge in this area and leading to a greater understanding of artificial intelligence and its many facets, as well as opening an extensive panorama in the investigation of the same, as a tool to help us solve social, political and economic problems of our nation.

The project consists of five chapters, which we will proceed to describe:

Chapter I General Aspects presents the approach of the scientific basis that will allow us to develop the proposed project, which begins with the approach, and identification of the problem to be solved; to then give way to the research methodology to be developed during the present work. We move on to the justification of the problem, which will allow us to have a broad overview of it, in other words, we will manage to raise a hypothesis with which we will outline some objectives, scope and limitations, and contributions of the thesis. To conclude, in this way, with the antecedents with greater relation to the present thesis.

Chapter II Theoretical Framework compiles in a condensed way the different theoretical arguments that allow us to relate and understand in a deeper way the present thesis, proof of this is that we will find in this chapter arguments ranging from the most specific to the most general, hence we have concepts of robots, microcomputers, programming language, industry and autonomy, design and culminating with concepts related to quadrupeds and rescue for the understanding of our project.

Chapter III Initial Analysis covers part of the necessary analysis to be performed, in order to obtain the necessary parameters to be considered at the time of the design and implementation of our navigation system, with these parameters we can also make an appropriate choice with respect to the necessary components to be used throughout the development of our project.

Chapter IV Design encompasses each of the necessary aspects to achieve the main objective of our project, which we will break it down into four important parts that will be each of the necessary components on which we will focus and then when assembling all these components into one we can reach the necessary results that allow us to meet the objectives set out in this thesis. Therefore, in the first part we develop a graphical interface that allows us to interact with the robot and load the programs wirelessly, then proceed to make a program for scanning a QR code using a camera, which will help us determine the position of the robot; from this we will build a program that allows us to trace the path to follow, so that we can finally design the system of movements to be performed by our robot to perform the required actions.

Chapter V Final Results shows us the realization of each of the simulations executed for each design elaborated in the previous chapter, demonstrating the effectiveness and the necessary results that can lead us to a conclusion at the end of this thesis.

Finally, we check if the results obtained have helped us to achieve each of the objectives set at the beginning of our project and if with the help of the necessary parameters to be considered we have managed to obtain the desired results.

Keywords: artificial intelligence, navigation, Navigation Stack, Python, Raspberry, trajectory planner, rescue, quadcopter, robotics, autonomy.

ÍNDICE

LISTA DE FIGURAS

LISTA DE TABLAS

INTRODUCCIÓN

Actualmente la tecnología ha logrado un grado de desarrollo progresivo que le ha permitido al ser humano desarrollar softwares que nos permiten encontrar la ubicación de objetos, personas y lugares a través de sistemas de triangulación monitoreada por nuestros equipos móviles. Además, los mismos al estar unidos a un poderoso hardware nos brinda la posibilidad de poder realizar funciones de rescate, localizaciones específicas en espacios mucho más reducidos o incluso de objetos perdidos. En otras palabras, la tecnología de los sistemas de navegación nos permite facilitar y brindar un apoyo al ser humano al momento de realizar estas actividades con menor riesgo para la salud e integridad física.

En nuestra era el cuidado de la salud es muy importante, hemos entrado en una etapa en la que muchas personas suelen arriesgar su vida al momento de realizar sus actividades laborales, es por eso que en muchos centros de trabajo se brindan charlas de seguridad, uso del equipo de protección persona, entre otras medidas. Por tanto, las herramientas tecnológicas pueden ser un gran aliado al momento de realizar ciertas actividades que puedan poner en riesgo la integridad del ser humano. Prueba de ello son los equipos de detección de fallas que nos permiten aminorar los riesgos al momento de realizar alguna obra o construcción.

Por otro lado, en situaciones de desastres naturales y enfrentamientos cívicos son momentos de alto riesgo para la salud e integridad física de las unidades de rescate, por esta razón la tecnología brindada por los sistemas de navegación son una solución viable que contribuiría con la reducción de los peligros al momento de realizar estas actividades. Para ilustrar mejor tenemos el ejemplo de EMILY, el robot de rescate marino desarrollado en los Estados Unidos. En resumidas cuentas, la tecnología es y será un gran aliado al momento de reducir los riesgos para el ser humano en actividades de alto riesgo.

Cabe concluir que, con la finalidad de promover las nuevas tecnologías y ponerlas al alcance de nuestras manos, plantearemos un sistema de navegación que nos permita ampliar el campo de investigación, ponernos a la vanguardia de las exigencias tecnológicas y a su vez proporcionar una ayuda para las unidades de rescate. Por ello, en la presente tesis haremos tangible la posibilidad de desarrollar un sistema de navegación que nos permita aminorar cualquier tipo de riesgo para el ser humano.

CAPÍTULO I

1. ASPECTOS GENERALES

1.1.Planteamiento del problema:

Desde sus inicios la robótica ha sido un medio por el cual el ser humano ha podido y puede realizar actividades, que muchas veces, se encuentran fuera de su alcance y posibilidades físicas, evitando, de igual manera, situaciones de alto riesgo.

Para la Asociación Japonesa de Robótica Industrial (JIRA) los robots son: "dispositivos capaces de moverse de modo flexible análogo al que poseen los organismos vivos, con o sin funciones intelectuales, permitiendo operaciones en respuesta a las ordenes humanas" (Lemus, s.f.).

En pocas palabras, un robot, puede realizar funciones similares a las de cualquier ser vivo, siempre y cuando tenga una estructura adecuada y contenga las ordenes apropiadas, para realizar dicha tarea, implementadas por su programador, resumiéndolo en un conjunto de 3 partes donde:

- Los sensores son los que permiten un estudio del estado del mecanismo y el entorno, captando así magnitudes físicas o alteraciones en su entorno (luz, sonido, temperatura, etc.).
- Los actuadores son los sistemas electromecánicos que permiten generar el movimiento de un mecanismo. Estos pueden ser sistemas eléctricos, mecánicos o neumáticos.
- Los sistemas de control van a permitir el correcto funcionamiento de un mecanismo.

En la actualidad se ha generado un gran campo de desarrollo con respecto a los robots con funcionalidades de seres vivos y apariencia de los mismos, generando una gran rama que se dedica a la fabricación de los denominados cuadrúpedos robóticos, los cuales suelen tener apariencia de canes, felinos o arácnidos.

Actualmente se puede observar que hay grandes avances con respecto al campo de apoyo y rescate de víctimas en casos de ciertos tipos de desastres, permitiendo que estos cuadrúpedos puedan realizar la localización, monitoreo y transporte de las víctimas. Para

ello, los cuadrúpedos no solo necesitan de una buena estructura para el buen desarrollo de estas tareas, sino que debe de ser acompañado por un software que les permita realizar las operaciones de la manera más adecuada.

Por estas razones, en este proyecto se buscará diseñar el sistema de control de navegación de un cuadrúpedo de rescate, utilizando el Navigation Stack para proporcionarle al cuadrúpedo de rescate la habilidad, imitando comportamientos inteligentes, de poder cumplir con las necesidades presentes y latentes en nuestro país, considerando que el Perú es un país altamente vulnerable ante los desastres naturales y conflictos civiles. Y a su vez, considerando las necesidades de apoyo a las unidades de rescate, entidades privadas, militares y policía nacional de la asistencia de un cuadrúpedo de rescate altamente capacitado de funciones que le permitan a estas mismas el desarrollar una labor de alto riesgo sin poner sus vidas en peligro.

Se buscará que el diseño del sistema de control de navegación del mismo sea el más óptimo y sencillo de implementar, teniendo en cuenta las dificultades que se puedan presentar durante un desastre natural o un conflicto civil, no solo para el desarrollo del mismo, sino en cuanto a que el operador pueda manipularlo con facilidad y no se necesite de un experto en la materia.

1.2. Metodología de la investigación:

Diseño de investigación: Investigación aplicativa.

Sujeto de prueba: Cuadrúpedo de rescate.

Instrumentos: Computadora, programa Python y Open CV.

Técnicas: Raspberry PI 4B, Python.

Procedimiento: Pruebas en módulos simulados.

Análisis de datos: En tiempo real y gráficos de respuesta.

1.3. Justificación del problema:

Es común que conforme vayan pasando los años la tecnología siga avanzando y los países vayan a la vanguardia de la misma, invirtiendo en muchos casos, tiempo y dinero en realizar investigaciones que les permita estar al frente de los avances tecnológicos. Es por eso que en la presente investigación se buscara ampliar el área de investigación en cuanto a tecnología e innovación, viendo que es una necesidad que transciende al tiempo y espacio.

Asimismo, la presente investigación profundizará en temas como el mejoramiento de los sistemas de programación de los cuadrúpedos y la investigación del mismo a nivel nacional e internacional, a través de artículos de investigación y tesis antes realizadas, y la importancia que tiene el profundizar en dicho campo como universidad y nación. Esta profundización aportará un mayor conocimiento de la inteligencia artificial y ampliará el campo de acción e investigación a este campo, su programación y su relación con la ingeniería mecatrónica.

Dado que la tecnología siempre va en constante avance y optimización, la presente investigación, busca aportar a la mejora de los cuadrúpedos de rescate en cuanto a software se refiere. Se prevé que dichas mejoras van a redundar siempre en el desempeño, diseño, control, etc. Al ser un país en busca de mejora y producir cada vez más ingresos la investigación tendrá impactos fuertes en las inversiones en el país, cuando se abra un campo de investigación en dicha tecnología.

Cabe concluir que en este aspecto existen muy pocas investigaciones con respecto a la inteligencia artificial en el país, sin embargo, durante los últimos 10 años el gobierno peruano ha apostado por invertir cada vez más en el desarrollo de tecnología e innovación, pero a lo largo de estos 10 años no hemos recibido donaciones y transferencias que aporten con la investigación de la tecnología y el país ha asumido este gasto. Por otro lado, a partir de esta investigación se pueden abrir puertas para que la nación sea un punto visible ante el mundo y genere inversiones internacionales.

Finalmente, el tesista busca culminar su proceso formativo y aplicar de manera profesional sus conocimientos adquiridos a lo largo de toda su formación académica, para la obtención del título profesional en ingeniería mecatrónica.

1.4. Hipótesis:

Teniendo en cuenta la facilidad y simplicidad conceptual con la que trabaja el Navigation Stack, y considerando los avances logrados a nivel mundial con respecto a la inteligencia artificial en el mundo tecnológico, es que la presente investigación pretende abrir paso en esta rama de la robótica para que el Perú pueda crecer en el desarrollo tecnológico y así convertirse en una potencia, en cuanto a tecnología, a nivel global.

En resumen, dado que el avance en la inteligencia artificial y sus múltiples aplicaciones van en un crecimiento constante y cada vez es más accesible a su conocimiento es probable obtener el diseño de un sistema de control de navegación utilizando inteligencia artificial para un cuadrúpedo de rescate.

1.5. Objetivos:

1.5.1. Objetivo general:

Diseñar el sistema de control de navegación, utilizando inteligencia artificial para un cuadrúpedo de rescate.

1.5.2. Objetivos específicos:

- Obtener el modelo algorítmico adecuado para nuestro sistema de control.
- Diseñar una interfaz que permita al usuario la manipulación y control del cuadrúpedo de rescate.
- Realizar módulos experimentales bajo el enfoque del Navigation Stack para una mayor facilidad de pruebas simuladas.
- Validar el diseño del sistema de control.

1.6. Alcances y limitaciones:

1.6.1. Alcances:

Este proyecto presentará mejoras en el control de un cuadrúpedo de rescate, tomando en consideración la importancia que se quiere lograr.

1.7. Contribuciones de la tesis:

A continuación, se presenta las contribuciones de este proyecto:

- Obtención de un control óptimo para cuadrúpedos de rescate.
- Desarrollo de una plataforma de simulación que permita la correcta y fácil manipulación del cuadrúpedo.

1.8. Antecedentes:

1.8.1. Antecedentes locales:

Bravo y Villegas (2017) "Diseño e implementación de un prototipo de brazo robótico (4GDL) teleoperado para manipulación de sustancias tóxicas asistido con visión artificial y redes neuronales para laboratorios farmacéuticos" Universidad Católica de Santa María, Arequipa – Perú.

En su investigación dieron una alternativa para la manipulación de sustancias tóxicas, para poder salvaguardar la salud de los farmacéuticos, con el desarrollo de un brazo robótico controlado por una placa Raspberry PI. Dentro de sus conclusiones lograron realizar el diseño cinemático y dinámico del prototipo, con lo cual pudieron asegurar su buen funcionamiento. Por otro lado, determinaron algoritmos inteligentes que les permitan controlar con facilidad y efectividad el brazo robótico. Por lo tanto, esta investigación es de gran importancia, debido a que demuestra que la robótica ayuda a la humanidad en situaciones que sean capaces de generar riesgo o peligro al ser humano.

1.8.2. Antecedentes nacionales:

Flores y Garay (2017) "Diseño e Implementación de un Robot Móvil de Desplazamiento Autónomo Basado en un Controlador Proporcional Derivativo" Universidad Nacional del Callao, Callao – Perú.

Mediante la investigación realizada se ha buscado poder controlar la posición y trayectoria de un robot dentro de un plano cartesiano, brindándole cierta información necesaria, la misma que ha sido ofrecida por el operador; por otro lado, se ha empleado el uso de un software que permita un mejor desempeño del proyecto, para este caso se utilizó el Labview. De igual forma, a través de este proyecto se puede llegar a determinar que un diseño apropiado y los componentes adecuados permiten el correcto funcionamiento del robot al momento de realizar operaciones como evadir obstáculos y calcular trayectorias. De esta forma la investigación nos ayuda ya que esta cuenta con un desarrollo adecuado al momento de generar las trayectorias con elementos sencillos, como por ejemplo el uso de un sensor ultrasónico, de esta manera también nos muestra que el uso adecuado de las técnicas necesarias permite el mejoramiento de las mismas y con esto permite la posibilidad de ampliar estos temas en estudios posteriores, de esta manera podemos obtener un ejemplo de un correcto funcionamiento para un planificador de trayectorias.

Olaya y Rodríguez (2018) "Diseño de una arquitectura embebida para el cálculo cinemático inverso de una extremidad robótica hexápoda de tres grados de libertad mediante el algoritmo cordic en FPGA" Universidad Privada Antenor Orrego, Trujillo – Perú.

En la presente investigación, han logrado realizar el cálculo cinemático en software para las articulaciones las extremidades de un robot hexápodo, con la finalidad de hacer más sencillo el trabajo para algunos estudiantes, para lo cual han empleado el método CORDIC para un dispositivo FPGA. Podemos observar que han logrado adaptar las ecuaciones cinemáticas inversas, con lo cual se puede obtener un modelo, de igual forma han logrado realizar un análisis algorítmico de CORDIC, haciendo con esto el resumen de las arquitecturas, consideraciones y métodos según el modelo matemático. Por lo tanto,

la presente investigación nos provee un ejemplo de que los modelos cinemáticos desarrollados en los robots, de cualquier tipo y material, tiene un gran estudio matemático con diferentes herramientas, lo cual permite que estos sean precisos para diferentes tipos de actividades.

1.8.3. Antecedentes internacionales:

Francés (2021) "Diseño e Implementación del Control de la Navegación Autónoma de un Modelo a Escala de Embarcación" Universitat Politècnica de València, Valencia – España.

En la presente investigación se quería desarrollar un control de una embarcación para poder ejecutar las funciones de alcanzar el punto de coordenada solicitado u obtener el control manual de la embarcación de manera remota, para ello se utilizó el microcontrolador ARDUINO y la plataforma de programación libre que nos brinda el mismos al momento de adquirirlo. Obtuvieron grandes resultados con respecto al alcance de las coordenadas brindadas por el operador, todo esto gracias al uso de herramientas y módulos de GPS. De esta manera el proyecto presenta grandes avances con el control de la embarcación al determinar una trayectoria específica a seguir y con ello poder llegar a la coordenada final. En consecuencia, la investigación nos ayuda a ver dos puntos muy importantes para nosotros, en primer lugar, el control y calculo de una trayectoria; y en segundo lugar que la inteligencia artificial ya está siendo probada en este tipo de estructuras para su mejoramiento.

Nail (2018) "Navegación autónoma de una base robótica usando SLAM" Pontificia Universidad Católica de Valparaíso, Valparaíso – Chile.

Mediante la investigación han buscado exponer la utilidad de las técnicas SLAM (Simultaneous Location And Mapping) que a traves de la implementación de algoritmos de localización pueden determinar una posición para la base del robot. Podemos apreciar que han logrado grandes resultados en cuanto a la evasión de obstáculos, sin tener mayor problema con la caminata del robot. También se ha logrado un buen control de toda la plataforma con un único microcontrolador para todo el sistema. De esta manera la investigación es un gran aporte para tomar en cuenta la integración que desarrolla entre

algoritmos de navegación y planificación de trayectorias, para tomar en cuenta al momento de querer realizar el enlace con el cuadrúpedo para obtener su autonomía.

Tordesillas (2015-2016) "Diseño y simulación del sistema de locomoción de un robot hexápodo para tareas de búsqueda y rescate" Universidad Politécnica de Madrid, Madrid - España

Mediante su investigación se ha buscado desarrollar un robot móvil que permita ayudar en situaciones donde las vías de acceso son difíciles tanto para el cuerpo de rescate, como para animales de rescate, desarrollando así un hexápodo de rescate que pueda entrar en diferentes terrenos, siendo controlado por un microcontrolador. Dentro de lo concluido en dicha investigación podemos observar que una buena selección de componentes electrónicos es lo que ha permitido realizar un buen diseño del robot, de igual las simulaciones han ayudado a que su elección de materiales y componentes sea de forma más óptima. De esta forma, podemos hallar en la presente investigación un camino adecuado que siguen los que desarrollan los diferentes robots con arquitectura similar a un insecto, como modelo para el estudio matemático, de simulación y de control, siendo este último fundamental para nuestra investigación.

CAPÍTULO II

2. MARCO TEÓRICO

2.1. Robots Adeept

Adeept es un equipo de servicio técnico de software y hardware de código abierto. Dedicado a aplicar Internet y la última tecnología industrial en el área de código abierto, brinda el soporte de hardware y servicio de software para fabricantes en general y entusiastas de la electrónica de todo el mundo. (Adeept, 2021)

2.1.1. Modelos

Dentro de los modelos que presenta Adeept (2021) se tiene:

- Adeept Ultimate Starter Kit para Arduino UNO R3, LCD1602; contiene: servomotor, relé, procesamiento y código C, kit de inicio para principiantes con guía de 140 páginas. Ver figura 1:

  Este es un Ultimate Starter Leaning Kit para Arduino y es el kit más completo, este kit contiene más de 50 tipos de componentes electrónicos diferentes, se incluyen más de 180 componentes.

Figura 1

Adeept Ultimate Starter Kit para Arduino UNO R3, LCD1602

Fuente. A continuación, se ve el Adeept Ultimate Starter Kit para Arduino UNO R3, LCD1602. Adaptado de "https://www.adeept.com/c/top5_0418" por Adeept (2021)

- Adeept Remote Control Smart Car Kit para Arduino basado en NRF24L01 2.4G; contiene: Wireless, Robot Starter Kit con guía / tutorial en PDF. Ver figura 2

El kit de coche inteligente está diseñado en base a Arduino UNO R3 y Nano. Es un kit de aprendizaje de Arduino.

Todo el sistema está dividido en dos partes, el automóvil y el control remoto, la comunicación entre ellos se basa en el módulo inalámbrico NRF24L01 2.4G. Además, este coche también tiene interfaz WiFi (ESP8266) y Bluetooth (HC-06).

Figura 2

Adeept Remote Control Smart Car Kit para Arduino basado en NRF24L01 2.4G Wireless

Fuente. A continuación, se ve el Adeept Remote Control Smart Car Kit para Arduino basado en NRF24L01 2.4G Wireless. Adaptado de "https://www.adeept.com/c/top5_0418" por Adeept (2021)

- Adeept RFID Starter Kit para Raspberry Pi 3 2 Modelo B / B + Python; contiene: guía impresa, placa GPIO de 40 pines. Ver figura 3

  Este es un kit de aprendizaje de inicio de RFID para Raspberry Pi. Se incluye un módulo RFID RC522, algunos componentes electrónicos y sensores comunes. El kit contiene más de 180 componentes y módulos electrónicos, estos componentes se pueden dividir en aproximadamente 30 especies. A través del aprendizaje, permite una mejor comprensión de RFID y Raspberry Pi. Permite aprender los conceptos básicos de la electrónica y la programación de Linux.

Figura 3

Adeept RFID Starter Kit para Raspberry Pi 3 2 Model B/B+

Nota. A continuación, se ve el Adeept RFID Starter Kit para Raspberry Pi 3 2 Model B/B+. Adaptado de "https://www.adeept.com/c/top5_0418" por Adeept (2021)

- Adeept New Ultimate Starter Learning Kit para Raspberry Pi 3 2 Modelo B / B + Python ADXL345 GPIO; contiene: Cable DC Motor. Ver figura 4

  Es el kit más completo de Raspberry Pi, este kit contiene más de 50 tipos de componentes electrónicos diferentes, se incluyen más de 180 componentes.

  Permite aprender los conceptos básicos de la electrónica y la programación de Linux.

Figura 4

Adeept New Ultimate Starter Learning Kit para Raspberry Pi 3 2 Model B/B+ Python ADXL345 GPIO

Fuente. A continuación, se ve el Adeept New Ultimate Starter Learning Kit for Raspberry Pi 3 2 Model B/B+ Python ADXL345 GPIO. Adaptado "https://www.adeept.com/c/top5_0418" por Adeept (2021)

- Adeept Hexapod 6 Patas Spider Robot Kit para Arduino UNO R3 y Nano; contiene: 20PCS AD002 Servo Motor; Control remoto inalámbrico 2.4G; Evitación de obstáculos. Ver figura 5

La placa Adeept Arduino UNO R3 se utiliza como placa de control central para este kit, y una placa de controlador PWM de 32 canales Adeept para el control de 19 servos en el lateral. Después del montaje, se pueden ver 2 partes, el robot hexápodo y un control remoto. Ambos se comunican a través del módulo de comunicación inalámbrica NRF24L01 2.4G en cada uno (Adeept, 2021).

Figura 5

Adeept Hexapod 6 Legs Spider Robot Kit para Arduino UNO R3 y Nano con 20PCS AD002 Servo Motor

Fuente. A continuación, se ve el Adeept Hexapod 6 Legs Spider Robot Kit para Arduino UNO R3 y Nano con 20PCS AD002 Servo Motor. Adaptado de “https://www.adeept.com/c/top5_0418” por Adeept (2021)

Otros de los modelos que más se comercializan según menciona Adeept (2021) son:

- El PiCar-B es un kit de coche robot IA basado en Raspberry Pi. Tiene las siguientes características:
  - Reconocimiento de voz: puede comprender palabras y luego ejecutar comandos.
  - Reconocimiento y seguimiento de objetos: basado en openCV, puede rastrear objetos de una forma o color específico.
  - Seguimiento de línea: basado en la reflexión de infrarrojos, puede caminar a lo largo de la ruta que establezca.
  - Evitación automática de obstáculos: basado en un sensor ultrasónico, puede evitar obstáculos por delante y encontrar el siguiente camino
  - Transmisión de video en tiempo real: puede transferir las imágenes en tiempo real tomadas por la cámara Raspberry Pi a una computadora remota.
  - Controlado de forma remota por la aplicación: puede controlar de forma remota el robot a través de los botones del teclado o los botones virtuales de la GUI.
  - Equipados con 12 LED RGB serie WS8212, estos LED RGB se pueden controlar a través de un solo pin GPIO, que puede cambiar una variedad de colores e indicar el estado de funcionamiento del robot.

- El RaspTank es una plataforma de robot móvil sobre orugas basada en Raspberry Pi, está equipado con un brazo robótico 4-DOF que puede agarrar objetos pequeños. Tiene las siguientes características:
  - Equipado con un brazo robótico 4-DOF;
  - Reconocimiento de objetos, rastreo, detección de movimiento: basado en openCV, puede rastrear objetos de una forma o color específico;
  - Seguimiento de línea: basado en la reflexión de infrarrojos, puede caminar a lo largo de la ruta que establezca;

- Transmisión de video en tiempo real: puede transferir las imágenes en tiempo real tomadas por la cámara Raspberry Pi a una computadora remota.
- Controlado de forma remota por la aplicación: puede controlar de forma remota el robot a través de los botones del teclado o los botones virtuales de la GUI.
- Equipados con 12 LED RGB serie WS8212, estos LED RGB se pueden controlar a través de un solo pin GPIO, que puede cambiar una variedad de colores e indicar el estado de funcionamiento del robot.

- RaspClaws es un robot araña hexápodo biónico basado en Raspberry Pi, hemos redactado un manual detallado e ilustrado con el que podrás completar rápidamente el montaje del robot. Tiene las siguientes características:
  - Robot educativo STEAM robot araña hexápodo basado en Raspberry Pi. Reconocimiento de objetos, seguimiento, detección de movimiento: basado en open CV; Arquitectura C / S: se puede controlar de forma remota mediante la aplicación GUI en la PC; LED RGB WS2812: pueden cambiar una variedad de colores, llenos de tecnología; Transmisión de video en tiempo real
  - Fácil de ensamblar y codificar
  - Alimentado por 2 × 18650 baterías.

2.1.2. Uso

Según menciona Adeept (2021) están dirigidos tanto a principiantes y profesionales y los usos que suelen darse a sus productos están enfocados a la enseñanza de inteligencia artificial, robótica, programación y electrónica.

2.2.Raspberry

2.2.1. Historia

Según refiere Diéguez (2021) en la historia de la informática, la Raspberry Pi claramente ha dejado huella; Eben Upton es un ingeniero británico, creador de Raspberry Pi y Raspberry Pi Foundation; comenzó construyendo tablets individuales para entretenimiento y actualmente vende más de 15 millones de dispositivos en todo el

mundo. Diéguez comenta que Upton estudió física e ingeniería en la Universidad de Cambridge y trabajó para prestigiosas empresas como Broadcom, Intel e IBM.

La historia de la Raspberry Pi comienza en 2006 como explica Diéguez (2021) con la creación de los primeros prototipos inspirados en BBC Micro. Y 6 años después, nació la primera Raspberry Pi, con el objetivo principal de acercar a los jóvenes a la computación a un menor costo. Eben Upton trabajó durante 5 años desde 2006 hasta 2011 en un solo proyecto de tableta, su inspiración fue la microcomputadora BBC de Acorn que usaba en la escuela (ver figura 6) Mientras Upton estaba creando su prototipo, se dio cuenta de que había un problema en la educación en el Reino Unido debido al alto precio de las computadoras, Upton intentó encontrar una solución, con el objetivo de hacer una computadora diez veces más barata. Como menciona Diéguez, se inspiro en muchos de los nombres de los fabricantes de computadoras como Apple, Acorns y Apricot, hizo lo mismo con la frambuesa (Raspberry). La parte "Pi" es una referencia a la parte Python de los primeros dispositivos creados. Estos dispositivos comienzan con un indicador de terminal en el que se debe escribir cualquier código Python para hacer lo que desee.

Figura 6

Raspberry Pi A

Fuente. Modelo Raspberry Pi A. Adaptación propia (2021)

Finalmente, Diéguez (2021) menciona que en 2009, Eben Upton fundó la Fundación Raspberry Pi para estructurar el desarrollo de Raspberry Pi. Esta es una organización benéfica registrada, con sede en el Reino Unido. El objetivo de Raspberry Pi es ayudar a los jóvenes a aprender los conceptos básicos de la programación a bajo costo.

2.2.2. Modelos

Diéguez (2021) menciona que la Raspberry originalmente serian dos modelos, según como planeo Eben Upton estos son: A (más barato) y B (más rápido). A continuación, se desarrollarán cada uno de estos modelos según refiere Diéguez (2021):

- MODELO A

Versión 1

La Raspberry Pi A no es el primer modelo lanzado, se pretendía que el modelo A sea más barato. Para que sea más barato tuvo menos puertos USB y RAM, la Raspberry Pi 1 A tiene las siguientes especificaciones:

- CPU: Brazo 700Mhz
- RAM: 256 Mo
- 1 puerto USB
- 8 pines GPIO
- Ranura HDMI, audio y SD

En noviembre de 2014, lanzaron una actualización (1 A +), con algunos cambios en el hardware, principalmente:

- 17 pines GPIO en lugar de 8
- Salida de audio Jack en lugar de RCA
- Micro SD en lugar de SD

Esto permitió que la Raspberry Pi 1 A + fuera más pequeña y liviana que la primera versión.

Versión 3

No hubo una versión 2 por ello pasaron directamente a la versión 3, llegó con Raspberry Pi 3A+ en noviembre de 2018. Tiene lo siguiente:

- Una arquitectura de 64 bits con una CPU 4xCore a 1.4Ghz

- 512 MB de RAM
- Todos los componentes de red necesarios: Gigabit LAN, WiFi y Bluetooth

- MODELO B

El Modelo B es el modelo Raspberry Pi más "potente".

Versión 1

Lanzado el 29 de febrero de 2012, fue la primera vez, que se vio un ordenador de placa única de 85x56 mm con:

- CPU: Brazo 700Mhz
- RAM: 512 M
- 2 puertos USB
- Ranura HDMI, RCA Audio y SD
- Conector Ethernet 10/100
- 8 pines GPIO

No es muy diferente de otras computadoras en el mercado, pero cuesta 10 veces menos. Fue un éxito inmediato con alrededor de 500.000 ventas en los primeros 6 meses. En cuanto al modelo A, fue lanzado en julio de 2014 una actualización con los mismos cambios:

- 17 pines GPIO en lugar de 8
- Salida de audio Jack en lugar de RCA
- Micro SD en lugar de SD

Y también se mueven a 4 puertos USB, lo que le da más posibilidades.

Versión 2

Luego lanzaron la Raspberry Pi 2 B con dos sub-versiones en 2015 y 2016:

- Raspberry Pi 2 B

✓ Publicado en febrero de 2015

✓ Tiene nuevas versiones de componentes básicos, pero sin grandes cambios

✓ CPU: tiene 4 núcleos a 900Mhz en lugar de 1x 700Mhz

✓ Obtiene 1 GB de RAM en lugar de 512M

- Raspberry Pi 2 B ver. 1.2

✓ Lanzamiento en octubre de 2016

✓ Tiene CPU de 1.2Ghz en lugar de 900Mhz

✓ Se traslada a una arquitectura de 64 bits

Ejecuta tareas que consumen CPU y tiene arquitectura de cuatro núcleos para ejecutar tareas simultáneas.

Versión 3

Lanzada en febrero de 2016. Básicamente es la misma placa que la Raspberry Pi 2B 1.2, llegando unos meses después. Pero añadiendo posibilidades inalámbricas, con Wi-Fi y Bluetooth integrados directamente en el tablero principal. Finalmente, en marzo de 2018, la Raspberry Pi 3 B + sale con dos mejoras:

- LAN: Gigabit Ethernet
- CPU: 4x 1,4 Ghz

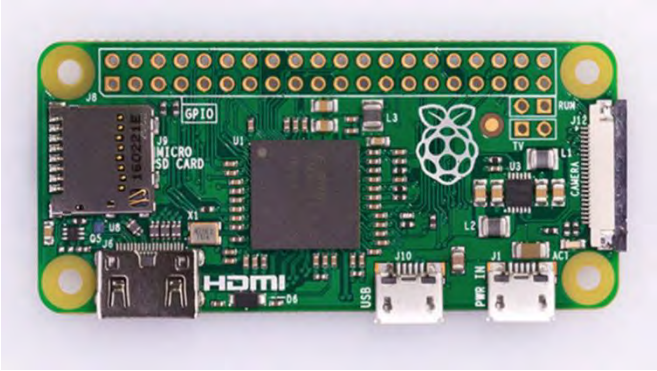
Versión 4

Luego lanzaron en junio de 2019 la Raspberry Pi 4. La última versión de Raspberry Pi, Raspberry Pi 4, ha salido con un nuevo chip ARM de 1,5 GHz y GPU VideoCore con algunas novedades: salida de pantalla dual-HDMI 4K; Puertos USB3; Gigabit Ethernet; y múltiples opciones de memoria RAM con 4GB y con 8 GB.

- MODELO ZERO

Figura 7

Raspberry Zero



Fuente. Foto del modelo Zero. Adaptación propia (2021)

El modelo Zero es la Raspberry más pequeña con 65 mm de largo y 30 mm de ancho (aproximadamente la mitad del modelo B).

Raspberry Pi Zero

El primer modelo "Zero" se lanzó en noviembre de 2015. El evento principal fue el tamaño de la placa, pero viene con estas especificaciones:

- CPU: BRAZO 1 Ghz
- RAM: 512 M
- 1 puerto micro USB
- 1 salida mini HDMI
- 17 pines GPIO

Debido a la falta de espacio, tiene menos puertos, pero la CPU y la RAM son suficientes para usarlo.

Raspberry Pi Zero W

En febrero de 2017, la Raspberry Pi Zero recibe una actualización, con una tarjeta inalámbrica (Wi-Fi y Bluetooth). También es la primera versión con un puerto de cámara integrado.

Raspberry Pi Zero WH

La versión WH tiene como única diferencia el encabezado GPIO pre-soldado.

Raspberry Pi 400

La Fundación Raspberry Pi en noviembre de 2020 lanzo un nuevo hardware: el kit de ordenador personal Raspberry Pi 400.

Muchas características son las mismas que la Raspberry Pi 4 Modelo B pero, sin embargo, la Raspberry Pi 400 está construido con el aspecto de un teclado de 78 teclas que integra un ordenador en el interior, un procesador Broadcom BCM2711 de cuatro núcleos Cortex-A72 (ARM v8) SoC de 64 bits a 1.8GHz, redes wifi, salida de doble pantalla y reproducción de video 4K, y una tarjeta micro-SD de 16GB preinstalada con Raspberry Pi OS.

Es ideal para navegar por la web, crear y editar documentos, ver videos y aprender a programar usando el entorno de escritorio del sistema operativo Raspberry Pi.

Raspberry Pi Pico

La Raspberry Pi Pico se ha presentado el 21 de enero de 2021 y es un cambio radical con respecto a las anteriores, ya que no es un ordenador Linux, sino una placa de microcontrolador como Arduino.

Los puntos fuertes de la Raspberry Pi Pico son el precio, y el nuevo chip RP2040, que cuenta con un Arm Cortex M0+ de doble núcleo que se ejecuta a 133 MHz, 264 KB de SRAM y 2 MB de memoria flash utilizada para almacenar archivos

2.2.3. Usos

Según menciona Alonso (2021) editor de la pagina web HardZone los usos de la Raspberry son:

- Conversión en un PC
- Servidor de impresión
- Convierte tu televisión en una Smart TV
- Crea una consola retro
- Servidor de Minecraft
- Crear un bot para Twitter, Discord u otros
- Como herramienta de monitorización

La página Digital Guide Ionos (2021) menciona los siguientes usos de la Raspberry:

- Servidor web.
- Estación de videoconferencia con Zoom o Skype.
- Air Quality Monitor (medidor de la calidad del aire)
- Sistema automático de bombeo de agua para máquinas de café
- Sistema central para una casa inteligente

- Dispositivo para streaming de música, imagen y video
- Servidor de correo electrónico.
- Sistema de seguridad Raspberry Pi
- Superficie LED interactiva
- Ventana LED
- Servidor VPN
- Reloj binario
- Servidor DNS
- "AirPi": aplicación para el tiempo y estación de medición del aire
- Repetidor de wifi; entre otros

2.2.4. Características

Según Noguera (2020) Raspberry Pi es una microcomputadora de bajo costo o placa de computadora SBC desarrollada en el Reino Unido por la Fundación Raspberry Pi, para que pueda promover la enseñanza y la programación de computadoras en las escuelas.

Básicamente, la Raspberry Pi es una tarjeta pequeña (aproximadamente del tamaño de una tarjeta de crédito). Tiene un procesador ARM con una potencia de hasta 1 GHz, integrado en el chip Broadcom BCM2835. También tiene 512 MB de RAM, una GPU Videocore IV, todo lo necesario para ejecutar programas básicos, navegar por Internet y programar (Noguera, 2020).

2.2.5. Comparación con ARDUINO

Como refiere Diéguez (2020) ambos fueron lanzados con el mismo propósito educativo, ambos dispositivos han ganado popularidad en todo el mundo.

Estas dos tarjetas son muy versátiles y adecuadas para una serie de tareas específicas. la diferencia entre Arduino y Raspberry Pi.

- Arduino es un microcontrolador. Un microcontrolador es una computadora simple que puede realizar una sola tarea una y otra vez. Es muy fácil de usar.

- Raspberry Pi es una mini PC o una mini computadora completa que necesita un sistema operativo como Linux y puede realizar más funciones, es un poco más complicado de usar.
- Arduino puede ejecutar pequeñas aplicaciones desarrolladas con programación C, pero no puede ejecutar el sistema operativo completo.
- Raspberry Pi, puede ejecutar un sistema operativo completo para muchas tareas.
- Arduino es ideal para controlar pequeños dispositivos como motores, luces, sensores, etc.
- Raspberry Pi es ideal para aprender e implementar programas para varios proyectos.
- Arduino está diseñado para "prototipos" electrónicos.
- Raspberry Pi es excelente para servir como servidor y para comunicarse con otros dispositivos informáticos.
- Arduino es conveniente y conveniente para hablar con otras máquinas.
- Raspberry Pi se usa mejor cuando se necesita una computadora completa: controlar un robot más complejo, ejecutar multitarea, realizar cálculos intensivos (como con Bitcoin, aunque esto le dará a la placa el máximo rendimiento)
- Arduino ideal para todo tipo de proyectos de electrónica. Gracias a los pines que tiene se puede conectar directamente a componentes y sensores y su programación es más rápida, permitiéndote realizar cambios muy rápidos en tu proyecto mientras depuras.
- El Arduino tiene en la memoria el firmware Arduino, un simple software que le permite comunicarse con la computadora a la que está conectado vía USB, que puede acceder a todos los detalles de la tarjeta.
- Con Arduino, encender y apagar los LED es una aplicación de código de 8 líneas, mientras que con Raspberry Pi necesitas descargar bibliotecas para controlar los puertos GPIO.
- Arduino es perfecto para realizar tareas simples y repetitivas.

- Raspberry Pi es ideal para realizar tareas que necesitan tomar decisiones basadas en diversas variables como verificar la hora en la web, comparar valores con sensores y tomar decisiones en consecuencia.
- Arduino tiene entradas analógicas y digitales. Las entradas digitales pueden gestionar entre 20 mA.
- La Raspberry Pi solo tiene entradas y salidas digitales y la salida digital solo puede manejar hasta 16 mA.
- Arduino y Raspberry Pi tienen una interfaz I2C
- Arduino tiene un procesador Atmega328 de 8 bits.
- La Raspberry Pi está equipada con un procesador ARM QuadCore de 6 bits con una frecuencia de reloj de 1,2 GHz.
- Raspberry Pi solo tiene entradas y salidas digitales, y las salidas digitales solo pueden gestionar hasta 16mA.
- Tanto Arduino como Raspberry Pi tienen interfaz I2C
- Arduino tiene un procesador de Atmega328 de 8 bits.
- Raspberry Pi tiene un procesador 1.2GHz Quad-Core ARM de 64bit.

2.3.Python

Python es una herramienta de programación que ha estado revolucionando el mundo tecnológico, no solo por sus prestaciones, sino por ser un entorno libre con grandes resultados para la industria de la programación. En otras palabras:

Es un lenguaje de programación versátil multiplataforma y multiparadigma que se destaca por su código legible y limpio. Una de las razones de su éxito es que cuenta con una licencia de código abierto que permite su utilización en cualquier escenario. Esto hace que sea uno de los lenguajes de iniciación de muchos programadores siendo impartido en escuelas y universidades de todo el mundo. Sumado a esto cuenta con grandes compañías que hacen de este un uso intensivo. Tal es el caso de Google, Facebook o Youtube, ya

que permite, entre otras de sus características la automatización de procesos y ejecución de tareas en tanto en entorno cliente como servidor (Robledano, 2019).

2.3.1. Historia

La historia de Python como lenguaje de programación inicia a finales de los 80s y principios de los 90s con Guido Van Rossum, una historia de 29 años de desarrollo.

En una navidad de 1989, Guido Van Rossum, quien trabajaba en el CWI (un centro de investigación holandés), decidió empezar un proyecto como pasatiempo dándole continuidad a ABC, un lenguaje de programación que se desarrolló en el CWI.

ABC fue desarrollado a principios de los 80s como alternativa a BASIC, fue pensado para principiantes por su facilidad de aprendizaje y uso. Su código era compacto pero legible.

El proyecto no trascendió ya que el hardware disponible en la época hacía difícil su uso. Así que Van Rossum le dió una segunda vida creando Python.

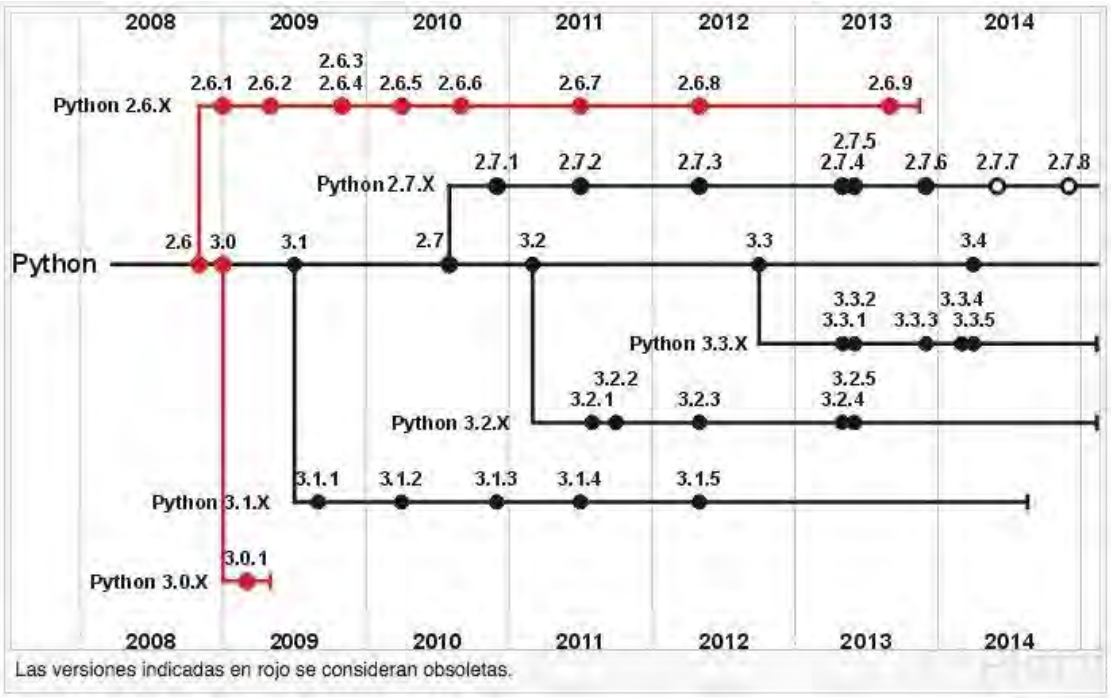
A Guido Van Rossum le gustaba mucho el grupo Monty Python, por esta razón escogió el nombre del lenguaje. Actualmente Van Rossum sigue ejerciendo el rol central decidiendo la dirección de Python.

En 1991, Van Rossum publicó el código de la versión 0.9.0 en alt.sources. En esta versión ya teníamos disponibles clases con herencias, manejo de excepciones, funciones y los tipos modulares.

En esta versión aparece un sistema de módulos adoptado de Modula-3, un lenguaje de programación estructurado y modular, el cual Guido describe como una de las mayores unidades de programación de Python. Por ejemplo, el modelo de excepciones de Python es parecido al de Modula-3

Para 1994 se creó comp.lang.python, un foro de discusión de Python que marcó un hito en su popularidad y multiplicó su cantidad de usuarios (Platzi, 2017).

Figura 8
Cronología de las versiones de Python existentes hasta el año 2014



Fuente. A continuación, se ve la cronología de las versiones de Python existentes hasta el año 2014. Adaptado de "https://platzi.com/blog/historia-python/" por Platzi (2017)

2.3.2. Actualizaciones

Las versiones de Python se identifican por tres números X.Y.Z, en la que:

- X corresponde a las grandes versiones de Python (1, 2 y 3), incompatibles entre sí:

  Los principales cambios introducidos en Python 2 fueron las cadenas Unicode, las comprensiones de listas, las asignaciones aumentadas, los nuevos métodos de cadenas y el recolector de basura para referencias cíclicas.

  Los principales cambios introducidos en Python 3 fueron la separación entre cadenas Unicode y datos binarios, la función print(), cambios en la sintaxis, tipos de datos, comparadores, etc.

  Por el momento, no hay planes de crear una nueva versión Python 4, incompatible con las anteriores.

- Y corresponde a versiones importantes en las que se introducen novedades en el lenguaje pero manteniendo la compatibilidad (salvo excepciones).

  Desde la versión 2.0 hasta 2019, las versiones X.Y se publicaron aproximadamente cada año y medio y se han mantenido durante cinco años, excepto la versión 2.7, que se mantuvo durante diez años, hasta el 1 de enero de 2020 (aunque se publicó una versión final en abril de 2020).

  En 2019 se decidió pasar a publicar nuevas versiones X.Y anualmente, en octubre, manteniéndolas durante cinco años. Así, Python 3.9 se publicó en octubre de 2020, Python 3.10 se publicó en octubre de 2021 y Python 3.11 se publicará en octubre de 2022.

- Z corresponde a versiones menores que se publican durante el período de mantenimiento, en las que sólo se corrigen errores durante el primer año y fallos de seguridad en los cuatro restantes.

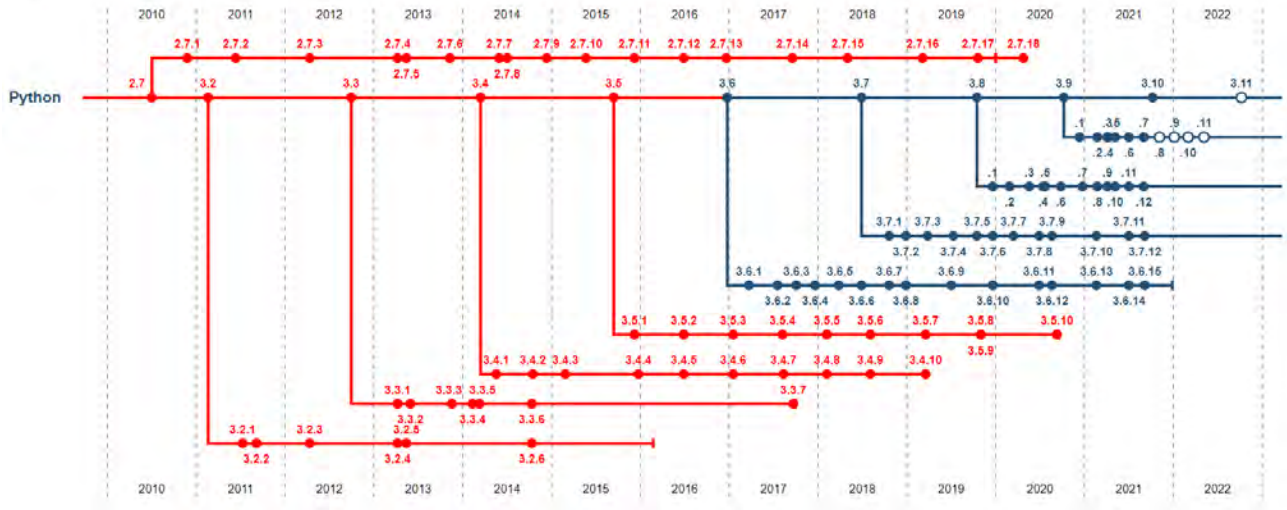
  La publicación de las versiones .Z es más irregular, porque el descubrimiento de fallos de seguridad puede provocar la necesidad de publicación de una nueva versión en cualquier momento, pero hay un calendario previsto de versones .Z.

  En principio, en el primer año después de la publicación de la versión inicial X.Y.0, se publican versiones .Z cada dos meses, tanto en forma de código fuente como de instaladores. Tras la publicación de la siguiente versión X.(Y+1).0 se publica una versión X.Y.Z proporcionando instaladores, pero las versiones .Z que se publican hasta completar los cinco de mantenimiento ya sólo se publican en forma de código fuente. Además, normalmente se publica una última versión X.Y.Z justo antes de que una versión X.Y deje de mantenerse.

Al margen de las versiones publicadas por la PSF, algunas empresas comerciales ofrecen el mantenimiento de versiones antiguas una vez acabado el mantenimiento oficial (Sintes, 2021).

Figura 9

Fecha de publicación de cada versión existente de Python



Fuente. A continuación, se ve la fecha de publicación de cada versión existente de Python. Adaptado de "https://www.mclibre.org/consultar/python/otros/historia.html" por mclibre (2021)

2.3.3. Características

Un lenguaje sencillo, legible y elegante que atiende a un conjunto de reglas que hacen muy corta su curva de aprendizaje. Si ya tienes unas nociones de programación o vienes de programar en otros lenguajes como Java no te será difícil comenzar a leer y entender el código desarrollado en Python.

El siguiente paso es comenzar a programar, verás que con muy pocas líneas de código es posible programar algoritmos complejos. Esto hace de Python un lenguaje práctico que permite ahorrar mucho tiempo (Platzi, 2017).

2.3.4. Aplicaciones

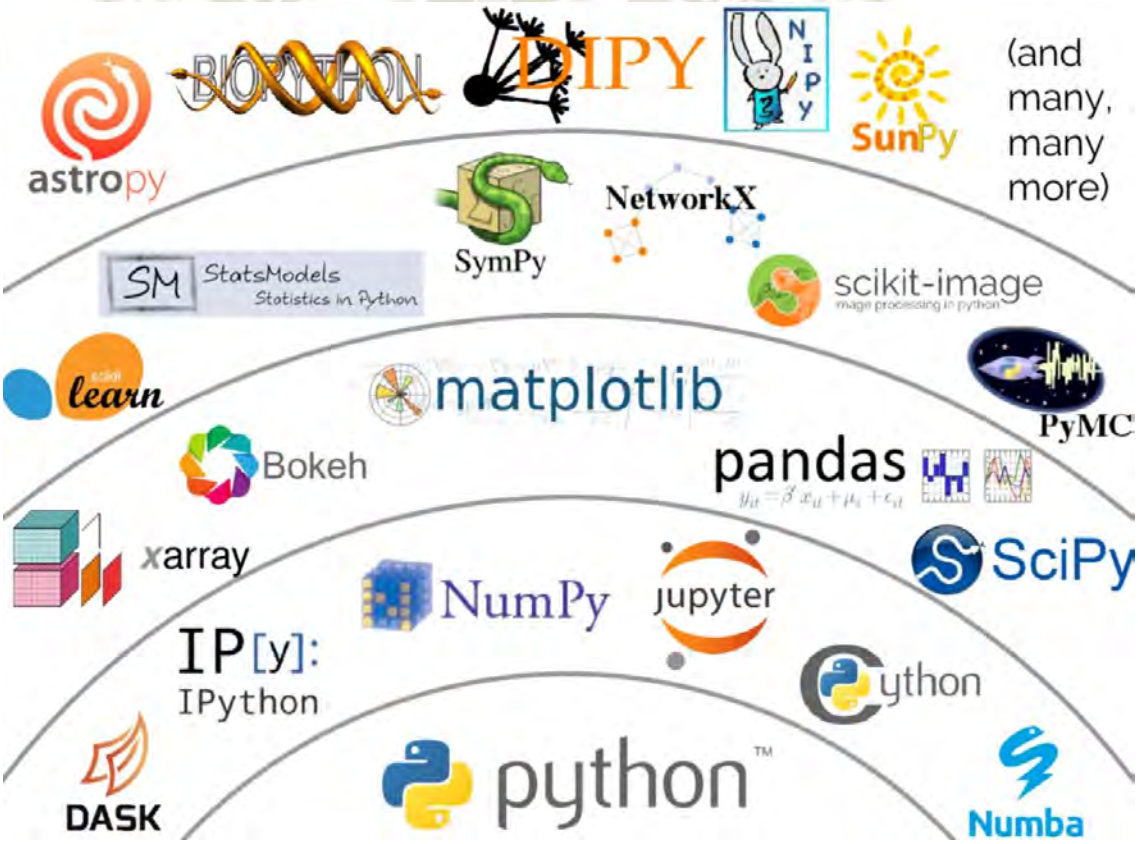
A lo largo de los últimos 30 años Python ha disfrutado de una evolución constante además de el respaldo de la comunidad que le han hecho particularmente relevante en el desarrollo de aplicaciones en entorno servidor. Esto, junto con su sencillez, ha producido que pudiera situarse en la escena del Big Data y en especial del desarrollo de algoritmos de aprendizaje automatizado.

La generalización del Big Data en los últimos años, seguida de la explosión de la Inteligencia Artificial, Machine Learning, Deep Learning y la consolidación de la ciencia

de datos como un nuevo área de trabajo con especialistas propios, ha revolucionado el panorama.

Al estar Python muy presente en el ámbito educativo siendo usado como lenguaje de referencia en escuelas y universidades su presencia en el campo de la investigación es más que justificado. Lo que ha producido que muchas de las herramientas que han surgido en este sector han sido desarrolladas en este lenguaje y explotadas por los ingenieros de datos y los científicos de datos. Algunos ejemplos son PySpark (Big Data), o Pandas, NumPy, Matplotlib o Jupyter (Data Science) (Platzi, 2017).

Figura 10
Diagrama de ecosistema Python



Fuente. A continuación, se ve el diagrama de ecosistema Python. Adaptado de "https://colab.research.google.com/github/csaybar/EarthEngineMasterGIS/blob/master/module02/01_tipodedatos.ipynb" por Google Colab (s.f.)

2.3.5. Frameworks

Un framework es una aplicación genérica que se puede configurar para añadirle las líneas de código que sean necesarias para programar una determinada aplicación; una analogía de ello podría ser un rompecabezas al que se le pueden agregar nuevas piezas para hacerlo más completo y detallado o agregarle nuevas funciones. Este marco de referencia o marco de trabajo pone a nuestra disposición un conjunto de código que podemos utilizar y reutilizar en cualquier sistema, así sea simple o complejo; ofreciéndonos una forma estándar para trabajar en un lenguaje de programación determinado como en nuestro caso Python (Platzi, 2017).

Alguno de los frameworks mas populares en Python son los siguientes:

- **Pyramid.** Pyramid nació de una fusión entre Pylons 1.0 y repoze.bfg. Este framework cuenta con una documentación excelente y permite a los desarrolladores avanzar sin tener que contar con el apoyo de la comunidad. Pyramid se esfuerza por ser minimalista, rápido y fiable. Fue uno de los primeros frameworks web compatible con Python 3.

- **Bottle.** Microframework muy simple que proporciona un mínimo de herramientas al desarrollador (enrutamiento, plantillas y una pequeña abstracción sobre WSGI) y es ideal para crear API web.

- **Django.** Estoy seguro de que ya has oído hablar de este Framework. Con diferencia es el framework más popular para Python. Cuenta con una potente interfaz de administración, así como otras muchas características, además de una ingente comunidad de desarrolladores que le dan soporte (Platzi, 2017).

Por consiguiente, los frameworks nos van a facilitar la manera de trabajar para el programador, encontrando un orden específico a toda la información necesaria al momento de desarrollar nuestro programa. Los frameworks sirven como estructura o esqueleto de lo que deseamos desarrollar dentro de nuestro aplicativo, en otras palabras con los ejemplos dados de frameworks podemos encontrar estructuras armadas que nos servirán de punto de partida al momento de implementar nuestra propia aplicación.

Figura 11
Top de los Frameworks más utilizados por Python en el 2019

Fuente. A continuación, se ve el top de los Frameworks más utilizados por Python en el 2019. Adaptado de "https://micropyramid.medium.com/which-is-the-best-python-framework-for-web-development-900694e295a6" por MicroPyramid (2020)

2.3.6. Ventajas de Python

- **Simplificado y rápido:** Este lenguaje simplifica mucho la programación, es un gran lenguaje para scripting.
- **Elegante y flexible:** El lenguaje ofrece muchas facilidades al programador al ser fácilmente legible e interpretable.
- **Programación sana y productiva:** Es sencillo de aprender, con una curva de aprendizaje moderada. Es muy fácil comenzar a programar y fomenta la productividad.
- **Ordenado y limpio:** es muy legible y sus módulos están bien organizados.

- **Portable:** Es un lenguaje muy portable. Podemos usarlo en prácticamente cualquier sistema de la actualidad.
- **Comunidad:** Cuenta con un gran número de usuarios. Su comunidad participa activamente en el desarrollo del lenguaje (Platzi, 2017).

2.4. Autonomía de los robots

La definición de Robots Autónomos en su sentido técnico, define a los robots, que tienen la aplicabilidad y capacidad de poder ejecutar actividades y tareas sin la necesidad de algún tipo de comando y control directamente explícito de los humanos. Los Robots autónomos son funcionales en lugares como: industrias y comercios, además de áreas de trabajo tan diferentes como: el agua, el aire y el espacio (Ripipsa, s.f.).

2.4.1. Modelos

Pasemos a revisar algunos modelos de robots según su funcionalidad, su campo de acción y servicio social. Prueba de ello son los siguientes modelos que presentamos:

- **Robots Autónomos o de Servicio:** Son tipos de Robots autónomos que cumplen las funciones de asistentes personales y servicios sociales, actúan como humanoides al poder distinguir acciones tan directas y particulares como hacer cosquillas, proporcionar medicinas a personas enfermas, dar órdenes a otras personas, informar sobre actividades a cumplir y horarios. Además también puede ejecutar actividades humanas con sus brazos antropomórficos y operar automáticamente en entornos humanos.
- **Robots Autónomos de Exploración Espacial:** Son robots autónomos que desarrollan y cumplen actividades similares a los humanos con sistemas y partes que realizan actividades de exploración, reparación y experimentación tanto en tierra como en el espacio.
- **Robots Autónomos para personas con discapacidad:** Son robots diseñados con capacidad y conceptos innovadores que permiten adaptarse a los entornos de una casa y permiten a las personas con movilidad reducida darles la

capacidad de moverse por las habitaciones, elevar su cuerpo para realizar actividades en casa y tareas domésticas como: cepillarse, ducharse, dar de comer, afeitarse etc.

- **Robots Autónomos de Rescate y Emergencia:** Han tenido un crecimiento importante como parte de la vanguardia tecnológica, que les permite ayudar por medio de un operario humano, permitiendo levantar grandes objetos para poder rescatar personas con problemas tras alguna emergencia.
- **Robots Autónomos Industriales:** Nos referimos a los ejemplos de Robots autónomos que cumplen actividades específicas como aquellos dedicados a realizar labores en algunas industrias de ensamblaje automovilístico. Un ejemplo de ellos son los robots autónomos militares, que realizan actividades de espía, recate y ataque. También existen algunos Robots autónomo con funcionalidad en el área médica para ejecutar labores de desinfección en áreas que requieren esa actividad (Ripipsa, s.f.).

Figura 12
Scout: Robot autónomo para realizar entregas desarrollado por Amazon

Fuente. A continuación, se ve a Scout: Robot autónomo para realizar entregas desarrollado por Amazon. Adaptado de "https://hipertextual.com/2019/01/amazon-scout-robot-repartidor" por Pinedo (2019)

2.4.2. Funcionamiento

Los Robots autónomos con el creciente avance de la Inteligencia Artificial y la Ciencia Cognoscitivas Corporizada les han permitido ser capaces de realizar diversas actividades en entornos cooperativos que pueden ser desde complejos y dinámicamente cambiantes.

Los Robots autónomos tienen su arquitectura de conformación y partes que permiten generar respuestas de sus actividades mediante las programaciones lógicas que son realizadas mediante un software y procesadores que le permite ejecutar el conjunto de representaciones y procesos neurales que conllevan a la realización de cualquier acción, coordinación, movimiento de partes, y de los sistemas de los robots autónomos.

En la gran mayoría de los casos en los Robots autónomos, la gran mayoría de las acciones y movimientos han sido definidos por comportamientos modelados en software con antelación.

No obstante, en los robots autónomos hay una tendencia cada vez más creciente por un crecimiento en el control hibrido y reactivo en su capacidad para combinar respuestas reales. Por ejemplo, las partes de los robots autónomos en entornos requieren racionalidad según el uso de datos y señales externas que son procesadas por sus sistemas: (Distancia, Temperatura, Sonido, Luz) por mencionar algunos.

Los Robots autónomos personifican la evolución de la Inteligencia Artificial actualmente. Por ello es fundamental el acoplamiento entre la percepción del robot y su entorno real, de cómo interpreta los datos detectados por sus sensores que les permite codificar con software una conducta operativa que desemboca en el cumplimiento de las actividades deseadas para el robot; mediante actuadores que ejecutan la orden y son aplicada por motores que según el sistema final actúan y les permiten la actividad a realizar (Ripipsa, s.f.).

2.4.3. Características

Cada una de las características que se conocen en a actualidad sobre los robots autónomos, son producto de investigaciones a lo largo de la historia que han permitido desarrollar y determinar a estos mismos. Prueba de ello son las siguientes características que nombraremos según (Ripipsa, s.f.):

- Generar información permanente sobre el entorno del medio ambiente para poder ejecutar actividades.
- Pueden ejecutar trabajos y actividades por largos periodos de tiempo, sin la necesidad de intervención humana directa.
- Libertad de movilidad en función al área operativa y de terrenos sin un control directo por parte de los humanos.
- Realizar decisiones operativas por ellos mismos para evitar acciones de riesgos y que puedan ser altamente perjudiciales en sus actividades.
- Lograr generar rendimiento de capacidad de aprendizaje en tareas y actividades según parámetros de funcionamiento con Inteligencia Artificial.
- Pueden lograr actividades de coordinación y acción con otros robots autónomos mediante parámetros de redes neuronales artificiales, algoritmos genéticos, lógica borrosa y aprendizaje por refuerzo.

2.5. Industria 4.0

Industria 4.0 se refiere a una nueva fase en la revolución industrial que se enfoca en gran medida en la interconectividad, la automatización, el aprendizaje automatizado y los datos en tiempo real. Industria 4.0, también conocida como IIoT o manufactura inteligente, integra la producción y las operaciones físicas con tecnología digital inteligente, aprendizaje automatizado y big data para crear un ecosistema más holístico y mejor conectado para las compañías que se enfocan en la manufactura y la administración de la cadena de suministro (Epicor, s.f.).

2.5.1. Aplicación de los robots autónomos en la industria 4.0

El constante crecimiento de la robótica y la inteligencia artificial han permitido que la industria 4.0 de un gran salto gracias al uso y desarrollo de robots autónomos que permiten realizar distintas actividades dentro de las diferentes industrias. Así tenemos las siguientes aplicaciones de los robots autónomos dentro de la industria 4.0, según (Ripipsa, s.f.):

- **Sector Militar:** Pueden ser utilizados para realizar labores de protección, rescate y acciones militares que impliquen el rastreo, seguimiento y operaciones de defensa mediante el uso de armas automáticas o decisiones autónomas.

- **Sector Médico:** Pueden ser utilizados en diversos procedimientos de cirugías mediante técnicas poco invasivas que maximicen la mayor cantidad de resultados con un mínimo de errores.

- **Sector de Atención y Servicios:** Existe una línea implicada en el desarrollo de robots autónomos que buscan incrementar la productividad al realizar cualquier tarea, desde ejecutar limpieza, tomar órdenes y ejecutarlas, mediante el entendimiento de su entorno y tomar decisiones en fracciones de tiempo.

- **Sector Espacial:** Los robots autónomos espaciales son un nicho bastante desarrollado, pues han facilitado su aplicabilidad en el reconocimiento en exploración de planetas, lunas y asteroides, además de ejecutar trabajos con astronautas en las estaciones espaciales.

- **Sector Industrial:** En algunos países como Japón y Estados Unidos han crecido de forma significativa la creación de empresas dedicadas a realizar la construcción de robots autónomos que puedan realizar actividades en ambientes de producción y manufactura, esto permite lograr que ejecuten tareas peligrosas o repetitivas con mínimo de errores con acciones programables que pueden finalmente ejecutarse por decisiones del propio robot según su entorno de trabajo.

En la actualidad la industria 4.0 está entrando en una total revolución tecnológica llevando el internet de las cosas a un nivel superior, debido a que en nuestros días no solo se aplica para la industria sino en diversos campos como son la economía, la banca, la política y muchos otros campos más.

Figura 13
Industria 4.0 y los sistemas de producción inteligente

Fuente. A continuación, se ve un ejemplo de la industria 4.0 y los sistemas de producción inteligente. Adaptado de "https://www.esss.co/es/blog/los-pilares-de-la-industria-4-0/" por ESSS (2017)

2.5.2. Características e importancia

Ante la evolución de la industria de servicios, también se crearon los sistemas de producción inteligentes, que consisten en la unión de las tecnologías físicas y digitales y la integración de todas las etapas de desarrollo de un producto o proceso, lo que trae como un importante impacto positivo más eficiencia y aumento de la productividad.

- **Instantánea:** seguimiento y análisis de datos en tiempo real, garantizando una mayor asertividad en la toma de decisiones. Saber todas las etapas del proceso el momento en que se produzcan.
- **Virtualización:** La simulación por ordenador es ya una realidad, sin embargo, la revolución de la industria propone la monitorización remota de los procesos de producción con el fin de evitar posibles fallos y hacer la red de la producción más eficiente.
- **La descentralización de la toma de decisiones:** Con el fin de mejorar la producción en la industria, sistemas cyber-físicos toman decisiones basadas en

el análisis de datos, sin depender de la acción exterior, tornando la decisión más segura y más precisa.

- **Modularización:** En este concepto, el sistema se divide en módulos, o sea, en diferentes partes. Por lo tanto, una máquina producirá de acuerdo con la demanda, ya que sólo utilizará los recursos necesarios para realizar cada tarea, lo que garantiza la optimización de la producción y ahorro energético.

Y para apoyar esta base, hay algunos pilares esenciales que impulsan la evolución de la industria 4.0 (ESSS, 2017).

2.6. Diseño industrial:

A continuación, podemos ver algunos conceptos relacionados con el diseño industrial, que nos permitirán tener una visión más amplia de la presente investigación.

Cabe señalar que es importante tener una idea clara de lo que, en cuanto a diseño, concierne. Es por eso que Lidia (1994) nos dice que:

La palabra Diseño hace referencia a la preconcepción sistematizada de la forma y las demás características del producto, teniendo en cuenta los aspectos sociales, tecnológicos, estéticos, psicológicos, anatómicos, fisiológicos, etc., es decir a la creación de un modelo del mismo (planos, prescripciones, etc.), con todos los detalles, antes de su realización (p. 7).

Sin embargo, podemos encontrarnos con autores que discrepen con la idea, de que existe un concepto claro con respecto a lo que se puede entender con claridad de lo que se entiende por la palabra diseño. Entonces tenemos que:

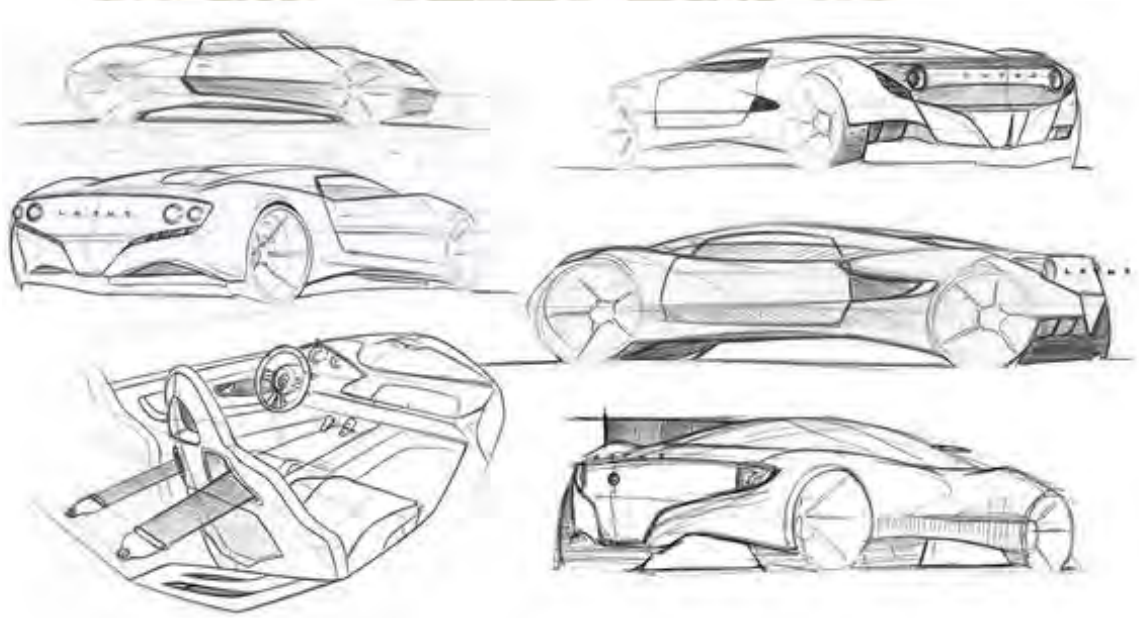
En mi concepción el diseño es plural y se encuentra en un debate desde hace muchas décadas entre opiniones contrarias por definir el concepto, el papel de la tecnología y el proceso industrial, la primacía de lo útil, simple y accesible ante lo lujoso y lo exclusivo, y el papel de la función, la estética, el ornamento y el simbolismo en objetos prácticos para el uso cotidiano (Sánchez, 2012, p. 3).

2.6.1. Diseño y sociedad:

Aun así, como personas de ciencia y promotores de la tecnología debemos buscar que el diseño sea una puerta que de aires de una sociedad moderna que no ha dejado de lado todo lo que le compete en cuanto a su esencia como tal, compartiendo de esta forma un pensamiento como el que expresa Bürdek (1994), al expresar lo siguiente:

Pienso que el diseño hoy en día debe estar en situación de reflejar las condiciones históricas, culturales y tecnológicas. La tradición ya no se corresponde con la continuidad histórica, sino que se ha convertido en ir y venir de acontecimientos contradictorios (p. 17).

Figura 14
Sketch & develepment work



Fuente. A continuación, se ve el diseño a mano de un automóvil. Adaptado de "Industrial Design Inspiration" por ABDUZEEDO (2012)

2.6.2. Tipos de Diseño Industrial:

De esta manera podemos entender con profundidad el concepto al que nos referimos en este trabajo al momento de hablar de un diseño, lo cual nos permite, incluso, clasificar el diseño industrial en diferentes tipos y modalidades utilizadas en el día a día al momento de hablar sobre diseño es de esta manera que tenemos aplicaciones como son el diseño

de interiores y mobiliario, el diseño vehicular y de componentes, el diseño de estructuras metálicas, diseño de marketing estratégico y publicitario, diseño experiencial, entre otros.

2.6.3. Diseño Electromecánico:

De ahí que podemos utilizar el término de diseño electromecánico al momento de hacer referencia, específica, con respecto a nuestro diseño, debido a que en un diseño electro mecánico intervienen elementos mecánicos, hidráulicos, eléctricos, electrónicos, programación electrónica y de protocolos de comunicación. Así podemos observar que en nuestro presente proyecto utilizaremos varios de estos elementos al momento de nuestro desarrollo, siendo quizá el más importante, y sobre el cual enfocaremos nuestra tesis, el del diseño y desarrollo de la programación electrónica y de protocolos de comunicación.

2.6.4. Estándares de Diseño:

Es preciso tener presente los estándares de diseño que debemos tener en cuenta al momento de realizar un diseño de cualquier tipo, en primer lugar, tendremos que realizar un análisis en base a lo que se desea lograr con el diseño de un sistema, posteriormente se procederá a realiza el diseño del mismo teniendo en cuenta los resultados obtenidos del análisis previamente realizado, una vez tengamos este diseño realizado procederemos implementarlo de manera adecuada para realizar los testeos necesarios que permitan en un siguiente paso la implantación del mismo y la validación de dicho sistema con lo cual podremos finalizar en el último paso que vendría a ser el aseguramiento de calidad del diseño implementado.

2.7. Diseño por computadora:

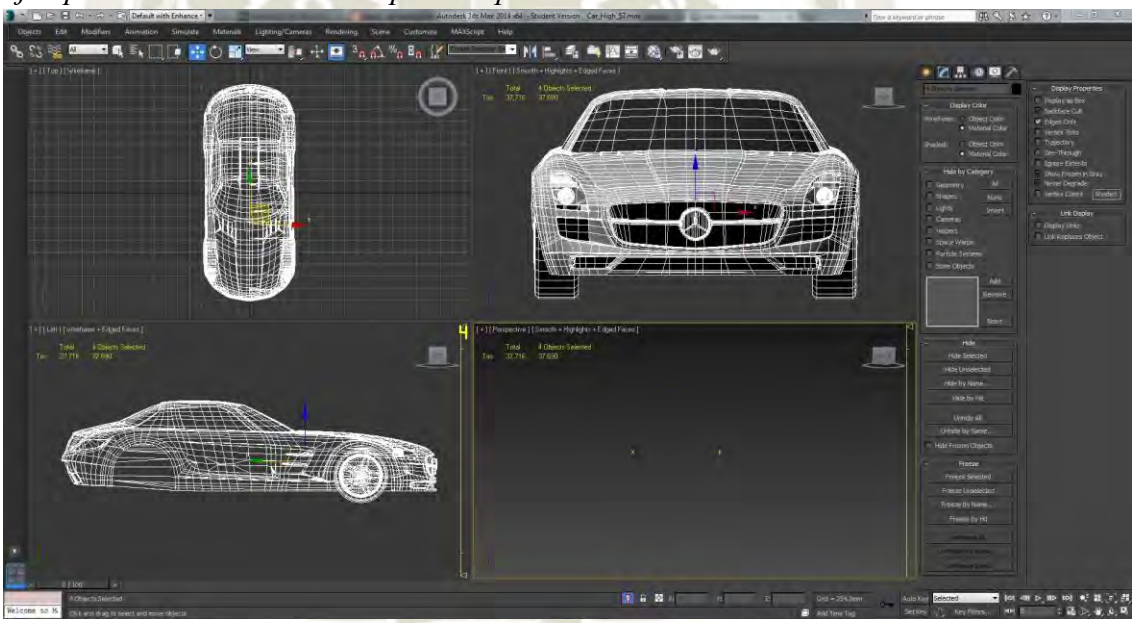
Partiendo del concepto anterior, pasaremos a revisar algunos conceptos más específicos, que siguen cumpliendo con el propósito de dar una mayor visión de lo que se irá desarrollando en nuestra investigación.

Para ilustrar mejor, definiremos de formar clara y concisa que se entiende por el termino diseño por computadora o diseño asistido por computadora es así que según "EcuRed" (2011), tenemos que:

El Diseño asistido por computadoras: Es el uso de un amplio rango de herramientas computacionales que asisten a ingenieros, arquitectos y a otros profesionales del diseño en sus respectivas actividades. Es todo sistema informático destinado a asistir al diseñador en su tarea específica (Definición, párr. 1).

Figura 15

Ejemplo de un diseño asistido por computadora



Fuente. A continuación, se ve el diseño por computadora de un Mercedes Benz. Adaptado de "Elección de un ordenador para CAD, diseñado modelado, animación o render 2D-3D" por mantis (2017)

Incluso esta asistencia por computadora no es de una forma limitada que solo nos proporcionará el diseño, sino que en la mayoría de herramientas nos brinda una asistencia completa. Por eso:

Diseño asistido por ordenador (CAD), también conocido como diseño asistido por ordenador y de redacción (CADD), es el uso de la tecnología informática para el proceso de diseño y documentación de diseño. Computer Aided Redacción describe el proceso de elaboración con un ordenador. CADD software, o entornos, proporciona al usuario

herramientas de entrada, con el propósito de agilizar los procesos de diseño, redacción, documentación y procesos de fabricación. CADD salida es a menudo en forma de archivos electrónicos de las operaciones de impresión o de mecanizado. El desarrollo de software basado en CADD está en correlación directa con los procesos que trata de economizar, la industria de software basada en (construcción, manufactura, etc) por lo general utiliza vectorial (lineal), mientras que los entornos gráficos basados en software utiliza basado en trama (pixelada) [*sic*] entornos "Manufacturing Terms Definition at a click away"("Diseño Asistido por Computadora", s.f., párr. 1).

Por otro lado, Rojas y Rojas (2006) nos dice "El CAD es una técnica de análisis, una manera de crear un modelo del comportamiento de un producto aun antes de que se haya construido. Los dibujos en papel pueden no ser necesarios en la fase del diseño." (p. 8), dejando en claro la importancia, hoy en día, del Diseño por computadora.

2.7.1. Tipos de Diseño por Computadora:

Sorprenderá talvez que los tipos de diseño por computadora han crecido en proporción a las necesidades, requerimientos y tipos de diseños existentes en el mercado, de esta manera podemos encontrar distintos programas de diseño que nos permitan cumplir con los requerimientos solicitados. Prueba de ello son los programas existentes como son: AutoDesk AutoCAD, Adobe Ilustrator, SolidWorks, AutoDesk Inventor, Sketchup, 3D Studio Max, entre otros.

2.7.2. Diseño de Software:

Un diseño de software está enfocado a poder determinar una solución para uno o más conjuntos de problemas planteados, de esta manera podemos determinar, a través de un diseño de software si se planteara una solución completamente automatizada donde no se necesite la interacción directa o indirecta de un usuario, o un software que tenga presente la participación de un usuario que pueda manipular de alguna manera el programa, dentro de esta misma se encuentra el diseño de software que se preocupa por el diseño de la experiencia del usuario por medio de un *storyboard*.

2.7.3. Herramientas de Desarrollo de Software:

En consecuencia, a todo lo expuesto anteriormente, podemos encontrar herramientas muy útiles al momento de efectuar el diseño de un software, teniendo en cuenta las múltiples funciones a realizar y ejecutar al momento de implementar dicho software, de esta manera podemos encontrar herramientas de suma importancia que nos ayuden a ser más productivos al momento de implementar y desarrollar nuestro software. Dentro de que están marcando una gran diferencia se encuentran GitHub, Git, GitLab, IntelliJ IDEA, Stack Overflow, Docker, Jira, Jenkins, entre otra serie de herramientas que nos permiten realizar una buena programación al momento de implementar nuestro software.

2.8. Computadora:

Pasemos a ver algunas definiciones relacionadas a la computadora, no solo nos servirá para entender de forma más amplia nuestra investigación, sino que nos servirá de sustentación para comprender el uso de algunos componentes en la presente investigación.

En efecto trataremos de dar una explicación clara de lo que se puede entender por computadora, por esta razón Turing (1950) nos dice que:

Podemos explicar el concepto de computadoras digitales diciendo que son unas máquinas ideadas para realizar cualquier tipo de operación propia de un computador humano. El computador humano sigue unas reglas determinadas sin opción a desviarse de ellas bajo ningún concepto. Supongamos que esas reglas figuran en un libro que cambia cada vez que el computador acomete un nuevo trabajo. Dispone también de una cantidad ilimitada de papel para efectuar cálculos y hace las multiplicaciones y sumas pertinentes con una «máquina de bolsillo», pero esto no tiene importancia (p. 5).

De manera que la computadora es una máquina que nos brinda facilidades y nos ayudan cotidianamente, por eso:

La computadora, ese equipo indispensable en la vida cotidiana de hoy en día que también se conoce por el nombre de computador u ordenador, es una máquina electrónica que permite procesar y acumular datos. El término proviene del latín computare (“calcular”).

Si buscamos la definición exacta del término computadora encontraremos que se trata de una máquina electrónica capaz de recibir, procesar y devolver resultados en torno a determinados datos y que para realizar esta tarea cuenta con un medio de entrada y uno de salida. Por otro lado, que un sistema informático se compone de dos subsistemas que reciben los nombres de software y hardware, el primero consiste en la parte lógica de la computadora (programas, aplicaciones, etc) el segundo en la parte física (elementos que la forman como mother, ventilador, memoria RAM) "Definiciones.de" (Definición de computadora, 2008, párrs. 1-2).

De ahí que la computadora con sus grandes ventajas nos provee una ayuda completa ante operaciones o funciones que pueden llegar a ser complejas, es por ello que la "Máquina capaz [*sic*] de realizar y controlar a gran velocidad cálculos y complicados procesos que requieren una toma rápida de decisiones, mediante la aplicación sistemática de criterios preestablecidos." (Gispert, 2002, p. 1544).

Figura 16
Computadora gamer

Fuente. A continuación, se ve una computadora gamer con todos los componentes de última generación que existen en el mercado actual. Adaptado de "https://blog.lumingo.com/setup-gamer-barato-como-armar-uno-basico-al-mejor-precio/" por Lumingo (2021)

2.8.1. Tipos de Computadora

Conviene recordad que el mundo de la computación se encuentra en constante cambio, incluso en los últimos años no solo se da una alta competencia por ver cuál es el ordenador con mayor venta en el mercado, sino por ser el ordenador con mayores prestaciones y beneficios que puede brindarle a su usuario. Prueba de ello son los tipos de ordenadores que existen actualmente en el mercado, que pueden servir de manera diferente según las necesidades del usuario, de ahí que tenemos los siguientes tipos de computadoras: supercomputadoras, mainframes, computadoras personales (PCs), computadoras portátiles, tablets y celulares.

Para simplificar podríamos decir que las computadoras han pasado por diversas etapas y transformaciones que suelen adaptarse, necesariamente, a las exigencias y necesidades de cada usuario, haciendolas de esta manera una herramienta útil y casi indispensable de nuestro día a día.

2.8.2. Partes de la Computadora

Partiendo de los puntos anteriores, pasaremos a nombrar las partes más esenciales de un computador, los cuales no permitirán entender el correcto funcionamiento del mismo. En consecuencia, podremos ver las partes más importantes de nuestro computador y las que más se tomaran en cuenta al momento de elegir una computadora que realice nuestros procesos, en efecto, las partes más importantes de un computador que tenemos que tener en cuenta al momento de elegir nuestra herramienta de trabajo son: La Unidad Central de Procesamiento, la memoria y los dispositivos periféricos.

Para resumir, las partes, antes mencionadas, de un computador son esenciales no en cuanto a sus especificaciones o mejoras que pueda ofrecer el mercado electrónico, sino según las exigencias y necesidades que presenta el usuario.

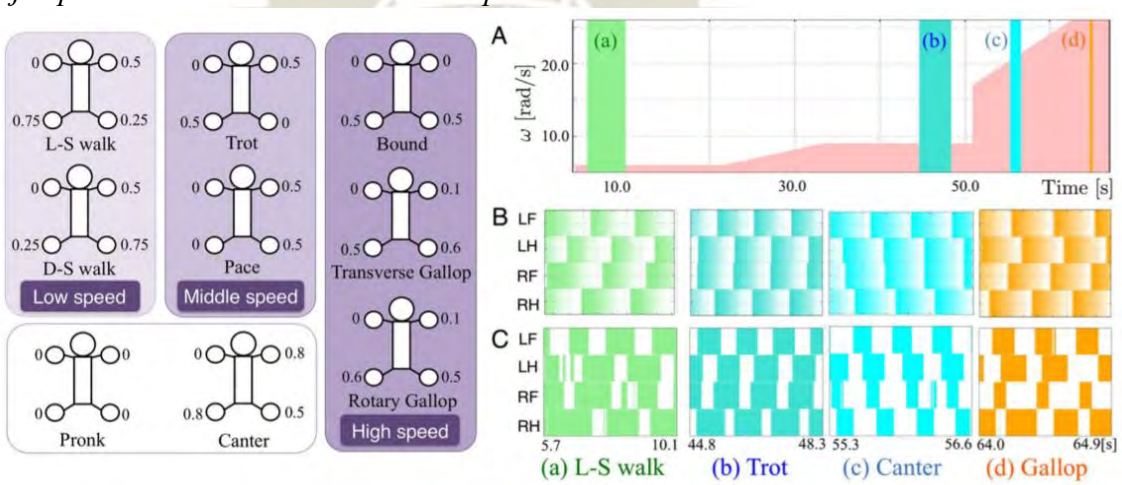
2.9. Cuadrúpedo:

Acto seguido vamos a revisar algunos conceptos concernientes a lo que por cuadrúpedo se puede entender, de forma que demos un preámbulo del tipo de mecanismo que deseamos controlar para la presente investigación, es así que tenemos:

Los cuadrúpedos, o animales de cuatro patas, fueron los primeros vertebrados capaces de moverse en la Tierra usando sus piernas. Debido a los dramáticos cambios evolutivos del medio ambiente submarino al terrestre a lo largo del tiempo, los animales de cuatro patas evolucionaron para contrarrestar el efecto de la gravedad, negociar el suelo terrestre y moverse de manera más eficiente para la depredación y la supervivencia. Al coordinar los movimientos de las piernas, es decir, la "coordinación entre miembros", los cuadrúpedos pueden cambiar sus patrones de locomotoras, por ejemplo. caminar, trotar y galopar (Fig. 1) para adoptar la marcha más eficiente energéticamente para una velocidad dada1, 2. Por lo tanto, el conocimiento del mecanismo de coordinación entre miembros que subyace en la transición cuadrada de la marcha es esencial para comprender el mecanismo de la locomotora en animales con patas y es Es útil para establecer principios de diseño para robots con patas que pueden reproducir una locomoción flexible y eficiente (Owaki & Ishiguro, 2016, p. 1).

Figura 17

Ejemplos de locomoción en un cuadrúpedo



Fuente. A continuación, se ven los tipos y patrones de locomoción en un cuadrúpedo. Adaptado de "A Quadruped Robot Exhibiting Spontaneous Gait Transitions from Walking to Trotting to Galloping" por D. Owaki A. Ishiguro (2016)

Antes de seguir adelante consideremos una definición más de lo que se puede entender por cuadrúpedo, de esta manera encontramos lo siguiente:

adj. y s. m. Dícese del animal que camina con cuatro extremidades. También puede aplicarse a máquinas construidas por el hombre (como el robot BigDog desarrollado por la NASA en 2005).

Los cuadrúpedos son animales que se trasladan sobre cuatro extremidades. La mayoría de los tetrápodos (animales con cuatro patas) son cuadrúpedos. Algunos de estos, como osos y simios, pueden caminar con dos extremidades en ocasiones.

La mayoría de los cuadrúpedos son animales vertebrados. Aves, humanos, insectos, crustáceos y serpientes no son cuadrúpedos; de todas maneras [*sic*] hay algunas excepciones, pe: la mantis religiosa es cuadrúpeda "Definiciones-de.com" (Definición de cuadrúpedo, 2016, párr. 1).

2.9.1. Diferencias entre Cuadrúpedo y Tetrápodo:

Ahora bien, para comprender un poco mejor el uso del término cuadrúpedo, en la presente investigación, y no tetrápodo tenemos que "educalingo" (2019), dice que "Un cuadrúpedo es un animal que se traslada caminando sobre cuatro extremidades; la mayoría de los tetrápodos actuales son cuadrúpedos, aunque algunos, como los osos o los simios, también pueden caminar cortas distancias sobre dos extremidades." (Cuadrúpedo, párr. 1), y así es que consideramos que el término tetrápodo no sería el más adecuado dado que nuestro fin no es lograr que nuestro robot pueda pasar de ser un cuadrúpedo a ser un bípedo.

Así logramos hacer esta diferencia sencilla que nos permite ir puliendo cada uno de los términos a utilizar y analizar dentro de nuestro proyecto, para que podamos obtener una exactitud al desarrollar los conceptos, usar los términos y poder presentar un proyecto de investigación de alta calidad y claridad para quienes deseen entender lo desarrollado en los próximos capítulos.

Figura 18

Una familia de osos

Fuente. A continuación, se ve una familia de osos con posturas de dos y cuatro extremidades. Adaptado de "https://www.ososwiki.com/imagenes-de-osos" por OsosWoki.com (s.f.)

Figura 19

Guepardo

Fuente. A continuación, se ve a un guepardo como ejemplo de un animal tetrápodo. Adaptado de "Poster Print of Leopard standing on acacia branch" por ardea wildlife pets enviroment (2019)

2.9.2. Tipos de Cuadrúpedos:

Por otro lado, podemos encontrar diferentes tipos de cuadrúpedos dentro de la gama robótica, siendo dentro de estas las más importantes los articulados y los de rueda. De manera que, los cuadrúpedos con ruedas no tendrán mayor desglose, en cambio en los articulados encontraremos la misma cantidad de tipos de cuadrúpedos según los grados de libertad que contengan. Ahora bien, esto se debe a las distintas funciones, movimientos, capacidades que podrá realizar nuestro robot según los grados de libertad que contenga.

2.10. Rescate:

Partiendo de todo lo anterior ya especificado, vamos a ver ciertas definiciones generales de lo que se puede entender como rescate con el fin de cubrir todos los aspectos a conocer de forma profunda los puntos y el alcance de nuestra investigación.

Para ilustrar mejor tenemos que "Rescate es la acción y efecto de rescatar (recobrar por fuerza o por precio algo que pasó a mano ajena). Este verbo también hace referencia a liberar de un peligro, daño o molestia." "Definiciones-de" (Definición de rescate, 2011, párr. 2).

2.10.1. Acepciones:

Sin embargo, podemos seguir ahondando en el tema y descubriremos que la definición de rescate puede varias muchas veces dependiente del contexto, creencia, aplicación e incluso podríamos llegar a decir que según la cultura. Es así que podemos encontrar que según "definición abc tu diccionario hecho fácil" (2016) nos dice que:

El término rescate tiene múltiples acepciones según el contexto en que sea utilizado. Así, en un ámbito religioso, el rescate es la salvación espiritual de una persona, mientras que [sic] si se usa en referencia a un secuestro, entonces hablaríamos del pago exigido para liberar a la persona u objeto que se tenga como rehén.

Por otro lado, en caso de una tragedia o situación de peligro, se rescata a aquellos individuos que necesitan de ayuda ante la emergencia.

Junto a las acepciones anteriores, la palabra rescate está también muy ligada al mundo de las finanzas y la economía.

Un rescate financiero es la forma en que se denominan las inyecciones de liquidez que se insuflan a una entidad para intentar recuperarla de una quiebra o una situación de peligro, de forma que pueda solventar sus obligaciones en el corto plazo (Definición de rescate, párr. 1-4). De esta forma podemos observar que el significado puede variar un poco en algunas circunstancias, pero en esencia sigue siendo lo mismo.

Figura20
Rescate en alturas

Fuente. A continuación, vemos un ejemplo de rescate. Adaptado de “Escapettor, solución definitiva a los descansos de evacuación” por Protecttion (s.f.)

2.11. Microcontrolador:

Acto seguido vamos a proceder a observar algunas definiciones sobre lo que es un microcontrolador, que nos permitan tener un panorama claro de la importancia que tiene este para el desarrollo de nuestra investigación.

De ahí que Barra y Barra (2015) nos indica que "Un microcontrolador es un circuito integrado de alta escala de integración que incorpora la mayor parte de los elementos que configuran un controlador." (p. 20), así tenemos que con un microcontrolador podemos realizar la configuración y el correcto funcionamiento de todos los controladores que vayamos a manejar para el desarrollo de nuestra investigación.

Por otro lado, tenemos una definición un poco más amplia que la dada anteriormente, es así que tenemos que:

Un microcontrolador (Microcontroller) es un circuito integrado digital monolítico que contiene todos los elementos de un procesador digital secuencial síncrono programable de arquitectura Harvard o Princeton (Von Neumann). Se suele denominar también microcomputador integrado o empotrado (Embedded processor) y está especialmente orientado a tareas de control y comunicaciones (Mandado, Menéndez, Fernández y López, 2007, p. 11).

Mediante a siguiente tabla, pasaremos a detallar los componentes de los cuales dispone, normalmente un microcontrolador.

Tabla 1
Componentes de un Microcontrolador

Componentes
Procesador o UCP (Unidad Central de Proceso)
Memoria RAM para contener los datos
Líneas E/S para comunicarse con el exterior
Diversos módulos para el control de periféricos (temporizadores, puertos serie y paralelo, ADC o conversores analógicos/digital, DAC o conversores digital/analógico, etc.)
Generador de impulso de reloj que sincronizan el funcionamiento de todo el sistema

Fuente. La siguiente tabla nos presenta de manera amplia y resumida los componentes de un microcontrolador. Adaptado "Controlador y Microcontroladores" por O. Barra y F. Barra (2015)

2.11.1. Arquitectura de un microcontrolador:

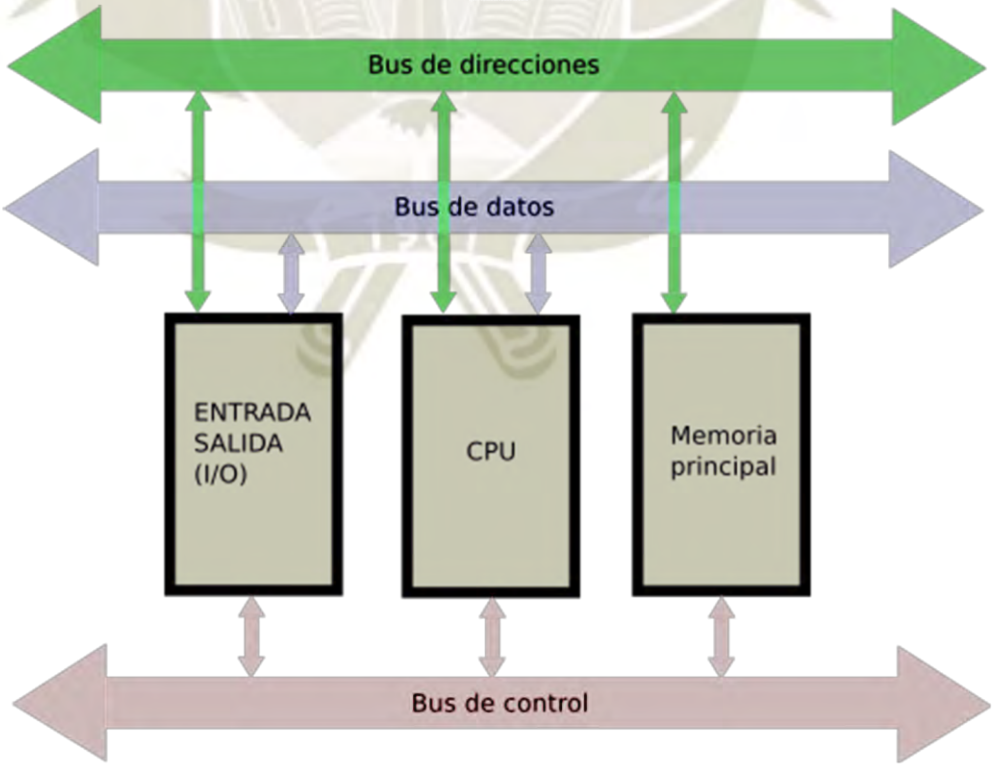
Los microcontroladores tienen dos tipos de arquitectura mediante la cual el CPU se va a comunicar con él, y se clasifican en los siguientes:

2.11.1.1. Arquitectura de Von Newmann:

Es importante tener en consideración de qué forma se desarrolla y como se caracteriza cada arquitectura para lograr tener una mayor comprensión de estos. Así, tenemos que:

Fue desarrollada por Jon Von Neumann [*sic*], se caracteriza por tener una sola memoria principal donde se almacenan datos e instrucciones de forma indistinta. La CPU se conecta a través de un sistema de buses (direcciones, datos y control). Esta arquitectura es limitada cuando se demanda rapidez (Grayeb y Alberto, 2008, "Arquitectura Von Neuman", párr. 1).

Figura 21
Esquema de una arquitectura Von Newmann



Fuente. Representación esquemática de una arquitectura Von Newmann. Adaptado de "Arquitectura Von Newmann" por Institut Puig Castellar (s.f.)

2.11.1.2. Arquitectura Harvard:

Pasemos a ver de ahora las características y composición de una arquitectura Harvard, por esta razón sabemos que:

Fue desarrollado en Harvard, por Howard Aiken, esta arquitectura se caracteriza por tener 2 memorias independientes una que contiene sólo instrucciones y otra, que contiene sólo datos. Ambas, disponen de sus respectivos sistemas de buses para el acceso y es posible realizar operaciones de acceso simultáneamente en ambas memorias (Grayeb y Alberto, 2008, "Arquitectura Harvard", párr. 1).

Con el paso del tiempo esta arquitectura ha tenido una mejora en cuanto a su construcción, la cual se puede encontrar en múltiples microcontroladores, de esta manera podemos ver que:

Existe una variante de esta arquitectura que permite el acceso a la tabla de datos desde la memoria de programas es la Arquitectura de Harvard Modificada. Esta última arquitectura es la dominante en los microcontroladores actuales ya que la memoria de programas es usualmente ROM, OTP, EPROM o FLASH, mientras que la memoria de datos es usualmente RAM. Por ejemplo [*sic*] las tablas de datos pueden estar en la memoria de programa sin que sean perdidas cada vez que el sistema es apagado (Grayeb y Alberto, 2008, "Arquitectura Harvard", párr. 2).

Antes de seguir adelante consideremos las principales ventajas que posee esta arquitectura, ya que esto nos ayudará a entender de mejor manera la arquitectura Harvard, además que la utilidad de esta tabla no solo permite una mayor comprensión de los conceptos y fortalece la justificación científica de nuestra tesis, sino que ayuda a que se puedan desarrollar los siguientes capítulos con mucha mayor fortaleza y dinámica

Por otro lado, encontraremos las diferencias que pueden presentar la arquitectura Harvard sobre la Von Newmann y las ventajas que la misma tiene sobre esta. Para ilustrar mejor tenemos la siguiente tabla:

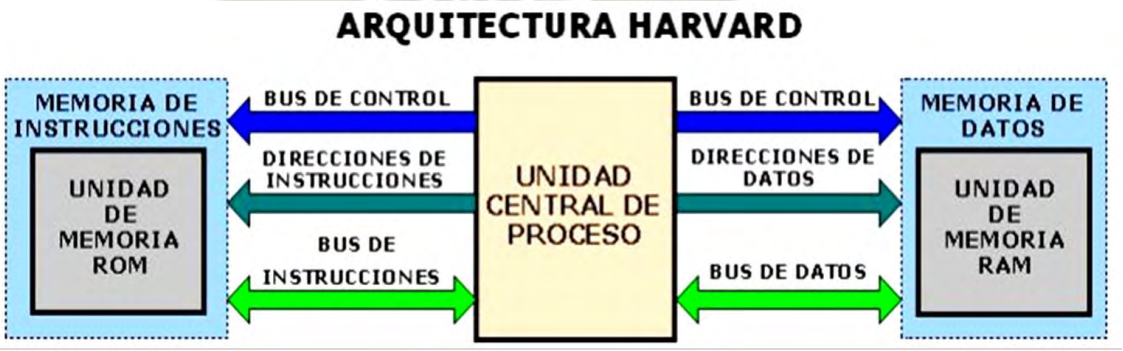
Tabla 2
Ventajas de la arquitectura Harvard

Ventajas
a) que el tamaño de las instrucciones no esta relacionado con el de los datos, y por lo tanto puede ser optimizado para que cualquier instrucción ocupe una sola posición de memoria de programa, logrando así mayor velocidad y menor longitud de programa.
b) que el tiempo de acceso a las instrucciones puede superponerse con el de los datos, logrando una mayor velocidad de operación.

Fuente. Principales ventajas de la arquitectura Harvard. Adaptado de "Arquitectura Harvard" POR s. Grayeb Y J.A. (2008)

El siguiente esquema nos ayudará a comprender de mejor manera el funcionamiento de una arquitectura Harvard.

Figura 22
Esquema de una arquitectura Harvard



Fuente. Representación esquemática de una arquitectura Harvard. Adaptado de "Arquitectura Von Newmann y arquitectura Harvard" por Computo Integrado (2012)

2.11.2. Diferencias entre un microprocesador, microcontrolador y placa de desarrollo:

El microcontrolador es un dispositivo que puede desempeñar distintas tareas especificas según su programación, siendo éste muy versátil dado que puede interactuar con el mundo exterior a través de sus periféricos adaptados según sea el requerimiento. Todo esto aunado a su pequeño tamaño de encapsulado y a su bajo costo comparado con otros equipos, específicamente con el microprocesador. Por otra parte el microprocesador es la base del microcontrolador, siendo este el cerebro de los computadores y ordenadores de uso general. Sin embargo estos no funcionan por sí solos, necesitan la integración de memorias, osciladores, convertidores, etc. lo cual por un lado resulta en un aumento de costos pero por otra parte significa flexibilidad dado que se pueden elegir distintos tipos de componentes siempre y cuando sean compatibles, y escalabilidad dado que según su arquitectura, la capacidad de memoria se puede expandir de acuerdo a los requerimientos, tarea que no es posible en los microcontroladores (Medium, 2017).

Una placa de desarrollo, conocidas también como computadora de placa única (SBC), es como su nombre indica un ordenador completamente construido sobre una placa de circuito único, con microprocesador(es), memoria, entrada/salida (E/S) y las características requeridas en un ordenador funcional. Estas computadoras de una sola placa se hicieron como sistemas de demostración o desarrollo, para sistemas educativos o para su uso como controladores integrados de computación. Incluso muchos tipos de ordenadores domésticos o portátiles integran todas sus funciones en una sola placa de circuito impreso (Solectroshop, s.f.).

2.12. Navegación autónoma:

En otro orden de cosas procederemos a dar una definición sobre lo que se logra entender de navegación autónoma, con el fin de poder entender mejor los conceptos que acompañaran a nuestro proyecto en el desarrollo de nuestra investigación. Así, logrando una mayor comprensión de la idea que se propone plasmar con la investigación y lograr que dicho proyecto pueda aportar un conocimiento amplio de la materia.

Partiendo de lo anteriormente mencionado procederemos a dar un concepto que nos permita comprender con mayor lucidez la idea de navegación autónoma, de esta manera tenemos que:

Se define navegación como la metodología (o arte) que permite guiar el curso de un robot móvil a través de un entorno con obstáculos. Existen diversos esquemas, pero todos ellos poseen en común el afán por llevar el vehículo a su destino de forma segura. La capacidad de reacción ante situaciones inesperadas debe ser la principal cualidad para desenvolverse, de modo eficaz, en entornos no estructurados (M. Martinez, 1995).

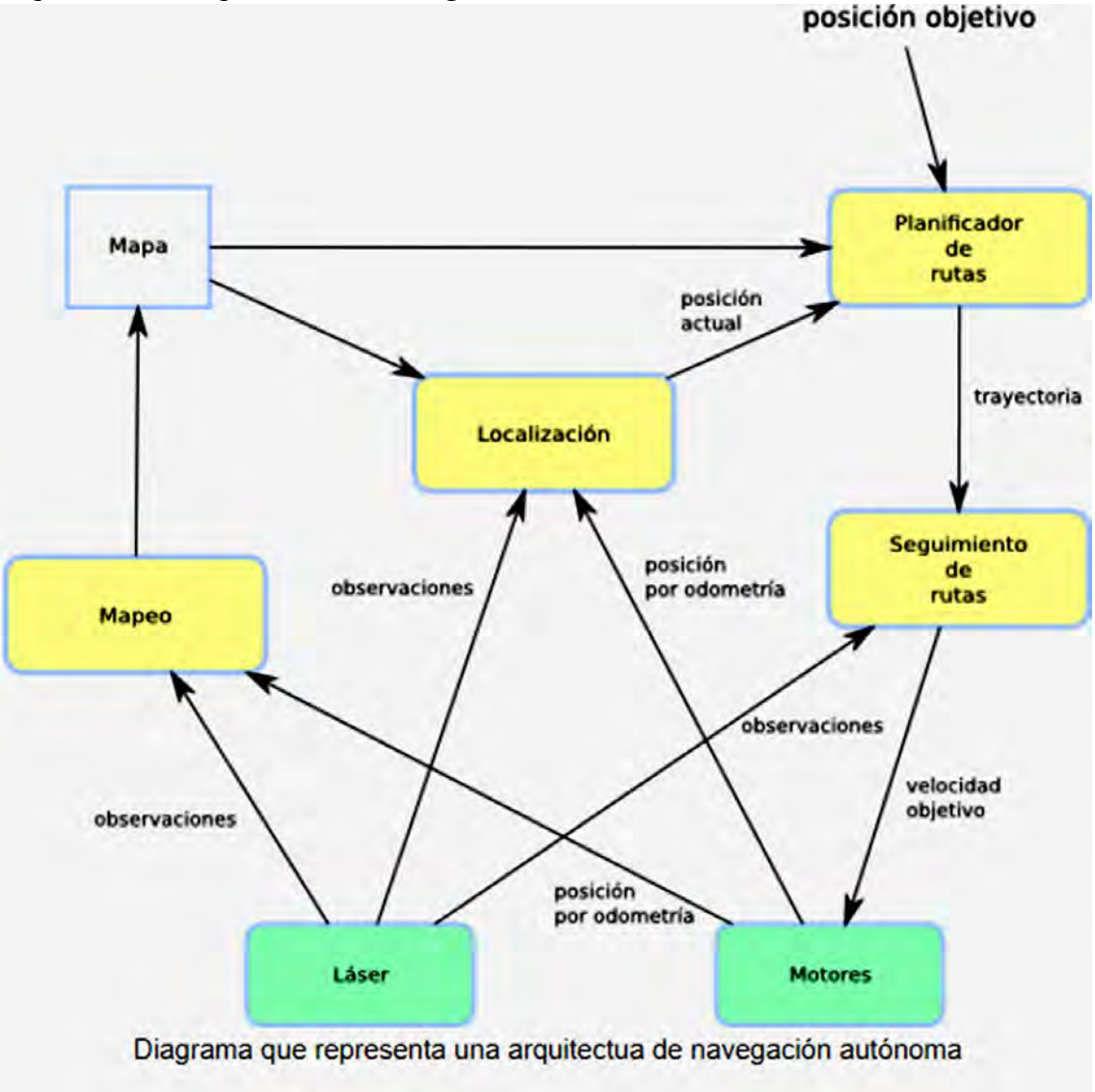
2.12.1. Elementos de la Navegación Autónoma:

Por esta razón es importante considerar también los elementos al momento de elaborar una navegación autónoma, como son la ubicación en el espacio, la planificación de la ruta, el generador de caminos y el seguimiento del camino para nuestro robot.

En consecuencia, la identificación de cada elemento que contiene un sistema de navegación autónoma nos ayudará a comprender el desarrollo que la tesis debe de tener al momento de diseñar nuestros programas, lo cual nos brindará un panorama mucho más amplio de los elementos y requerimientos específicos que debamos considerar en el siguiente capítulo, dentro de estos elementos vamos a encontrar las funciones específicas que puede desempeñar un sistema de navegación autónoma para su adecuado funcionamiento. Prueba de ello es el esquema gráfico o mapa mental que se visualiza en la figura 23 de nuestro proyecto de investigación, el cual nos brinda los elementos que tiene un sistema de navegación autónoma y la relación que debe tener cada uno de ellos con el otro.

Figura 23

Esquema de la arquitectura de navegación autónoma



Fuente. Representación de un diagrama sobre la arquitectura de navegación autónoma que debe seguir un robot móvil. Adaptado de "Navegación autónomas" por Azkume (2011)

2.13. Inteligencia Artificial:

Partiendo de una previa investigación, nos damos cuenta que es difícil dar con una definición exacta de lo que a inteligencia artificial se refiere, pero podemos ilustrarnos con lo que han dicho algunos autores que han dado luces de lo que para ellos es la inteligencia artificial. De ahí que, tenemos que "Es la ciencia de hacer que las máquinas hagan cosas que requerirían inteligencia si las hubiera hecho un humano" Minsky (como se citó en "El País", 2016, "Marvin Minsky, cerebro", párr. 1).

2.13.1. Tipos de IA:

Cabe señalar que existen algunos tipos de inteligencia artificial, los cuales vamos a precisar en la siguiente tabla:

Tabla 3
Tipos de inteligencia artificial

Tipo	Nombre	Explicación
1	Máquinas reactivas	Un ejemplo es Deep Blue, el programa de ajedrez de IBM que venció a Garry Kasparov en los años noventa. Deep Blue puede identificar piezas en el tablero de ajedrez y hacer predicciones, pero no tiene memoria y no puede usar experiencias pasadas para informar a las futuras. Analiza movimientos posibles –los propio y los de su oponente– y elige el movimiento más estratégico. Deep Blue y AlphaGO de Google fueron diseñados para propósitos estrechos y no pueden aplicarse fácilmente a otra situación.
2	Memoria limitada	Estos sistemas de IA pueden usar experiencias pasadas para informar decisiones futuras. Algunas de las funciones de toma de decisiones en vehículos autónomos han sido diseñadas de esta manera. Las observaciones son utilizadas para informar las acciones que ocurren en un futuro no tan lejano, como un coche que ha cambiado de carril. Estas observaciones no se almacenan permanentemente.
3	Teoría de la mente	Este es un término psicológico. Se refiere a la comprensión de que los demás tienen sus propias creencias, deseos e intenciones que afectan las decisiones que toman. Este tipo de IA aún no existe.
4	Autoconocimiento	En esta categoría, los sistemas de IA tienen un sentido de sí mismos, tienen conciencia. Las máquinas con conciencia de sí comprenden su estado actual y pueden usar la información para inferir lo que otros están sintiendo. Este tipo de IA aún no existe.

Fuente. Descripción de los tipos de inteligencia artificial. Adaptado de "Inteligencia Artificial, o AI" por M. Rouse (2017)

CAPÍTULO III

3. ANÁLISIS INICIAL

3.1.Análisis Inicial

Antes de continuar insistamos en realizar un análisis de los parámetros necesarios a considerar antes de realizar el diseño completo de nuestro programa, con la finalidad de poder tener un amplio panorama de aquellos parámetros necesarios, apoyando y fortaleciendo a la idea de realizar el programa y que este será viable con los indicadores a considerar. Asimismo, realizaremos el mismo análisis para nuestro robot.

Luego de realizar este análisis inicial y haber determinado los parámetros requeridos para la programación, podremos proceder con el diseño de nuestro sistema de control de navegación. En otro orden de cosas realizaremos un análisis de los componentes base que debemos de tener en cuenta para nuestro sujeto de prueba, en otras palabras verificaremos los actuadores, sensores, microcontroladores y entre otros componentes que sean necesarios a considerar, con el fin de poder elegir o ensamblar el robot con mejores prestaciones que se pueda utilizar para realizar las pruebas necesarias; de esta manera y obteniendo los datos necesarios procederemos con lo planteado con el fin de lograr los objetivos propuestos dentro del primer capítulo de nuestro trabajo.

3.1.1. Parámetros Requeridos

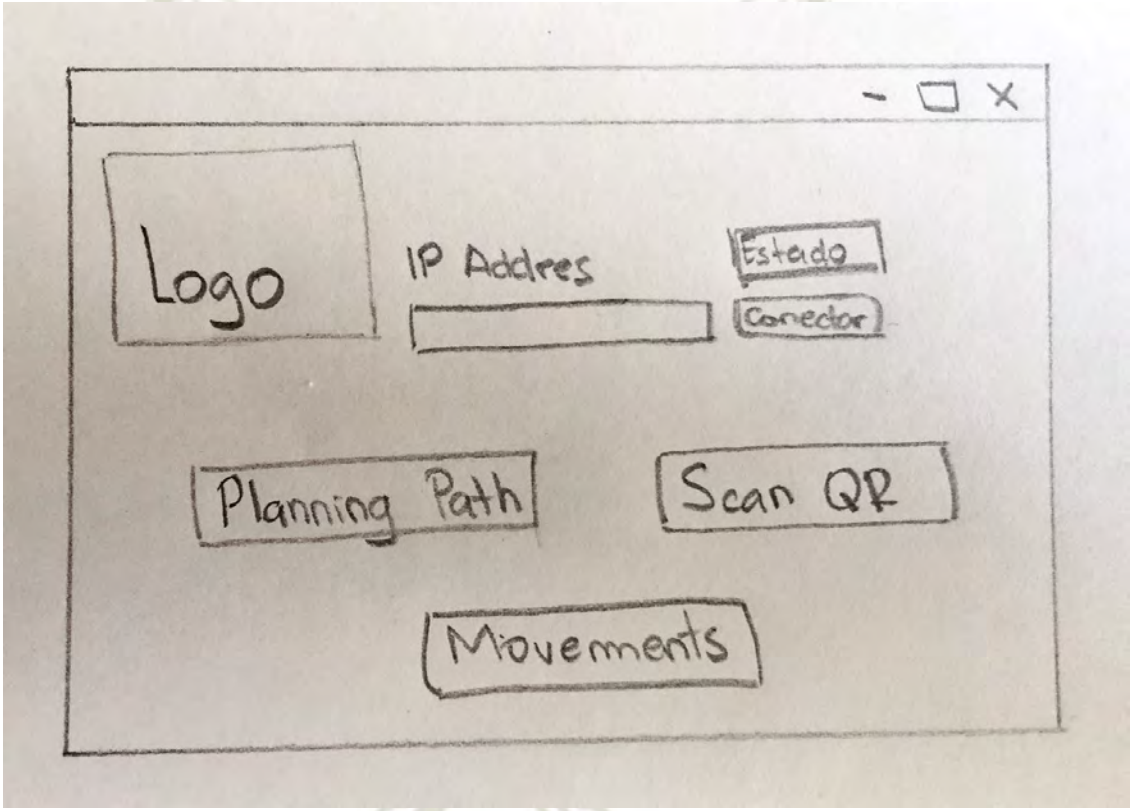
3.1.1.1. Interfaz Gráfica

Para comenzar el desarrollo de nuestro programa, debemos analizar los aspectos necesarios a tomar en cuenta sobre nuestra interfaz gráfica a desarrollar; por eso es que necesitaremos un esquema previo antes de entrar al desarrollo absoluto del mismo, considerando y analizando la viabilidad de este interfaz en su versión de prueba, de manera que sin un interfaz gráfico no sería posible avanzar de una forma óptima con los demás pasos a analizar en el presente capítulo de nuestro trabajo.

En consecuencia, tenemos que desarrollar una interfaz gráfica, que nos permita tener el control del robot durante su manipulación, para lo cual necesitaremos considerar todos los elementos que creamos necesarios para poder llevar a cabo la correcta manipulación

del robot, de esta manera procederemos a realizar un primer bosque a mano alzada, donde determinaremos algunos parámetros que creamos necesarios como pueden ser, cuadros de diálogo, cuadros de estado, textos, logo, entre otros a considerar que deben ser la base de nuestra interfaz gráfica al momento de realizar nuestra programación y que esta pueda ser plasmada más adelante en nuestro diseño final.

Figura 24
Bosquejo a mano alzada de la interfaz gráfica



Fuente. Realización de un interfaz gráfico a mano alzada, para poder determinar las partes necesarias. Adaptación propia en base a la investigación realizada (2020)

Así, con este bosquejo podemos tener una idea de lo que queremos lograr al momento de programar el interfaz gráfico, teniendo en cuenta los elementos que se muestran en la figura 24. Por lo tanto, procederemos a realizar una primera programación de nuestro interfaz gráfico, en el lenguaje y plataforma de Python, para obtener así una primera muestra de lo que se requiere para realizar la manipulación de nuestro robot, no podemos

olvidar que la interfaz gráfica debe de ser sencilla y fácil de entender para cualquier persona que vaya a ser el operador.

Figura 25

Programación de las características de la interfaz gráfica

```
def loop():                                          #GUI
        global tcpClicSock,root,E1,connect,l_ip_4,l_ip_5,color_btn,color_text,Btn14,color_text,var_R,var_B,va
        while True:
                color_bg='#000000'          #Establecer el color de fondo
                color_text='#000000'        #Establecer el color del texto
                color_btn='#5AECB8'         #Establecer el color de los botones
                color_line='#01579B'        #Establecer el color de las lineas
                color_can='#000000'         #Establecer el color del lienzo
                color_oval='#2196F3'        #Establecer el color ovalado
                target_color='#FF6D00'

                root = tk.Tk()                      #Definir una ventana root.
                root.title('Tesis Giuliano')        #Título principal de la ventana.
                root.geometry('700x230')            #Tamaño de la ventana principal.
                root.config(bg=color_bg)            #Establecer el color de fondo de la ventana root.
```

Fuente. Programación de las características principales del interfaz gráfico, como color de fondo, dimensiones, entre otros. Adaptación propia en base a la investigación realizada (2020)

A partir de la base formada en la programación mostrada en la figura 24, procederemos a realizar la programación de algunos elementos esenciales que nos permitirán tener un mayor control de lo que se puede realizar con nuestro robot, de esta manera podemos definir elementos como, botones, encabezados, cuadros de diálogo, estados, entre otros.

Por otro lado, una vez terminada la programación de nuestra interfaz gráfica procederemos a realizar la programación más profunda en cuanto a esencia se refiere, pero primero debemos de tener una estructura sólida de nuestra interfaz gráfica, de manera que obtengamos lo necesario para poder entrar a la parte más consistente de nuestro trabajo.

Por lo tanto, desde este punto podremos contar con un requerimiento específico de lo que necesitamos para poder realizar con efectividad nuestra interfaz gráfica en el capítulo de diseño, de esta manera es importante ir sentando las bases necesarias y específicas de los parámetros requeridos de la propia interfaz gráfica, esto se debe a que la interfaz será propuesta, diseñada y aprobada por nosotros, por consiguiente las especificaciones de posicionamiento de botones, imágenes, tipo de letra, tamaño de pantalla, colores,

formatos, entre otros; serán parámetros específicos que vamos a dar nosotros mismos para el correcto desarrollo de nuestra investigación.

Figura 26

Programación de los componentes dentro de la interfaz gráfica

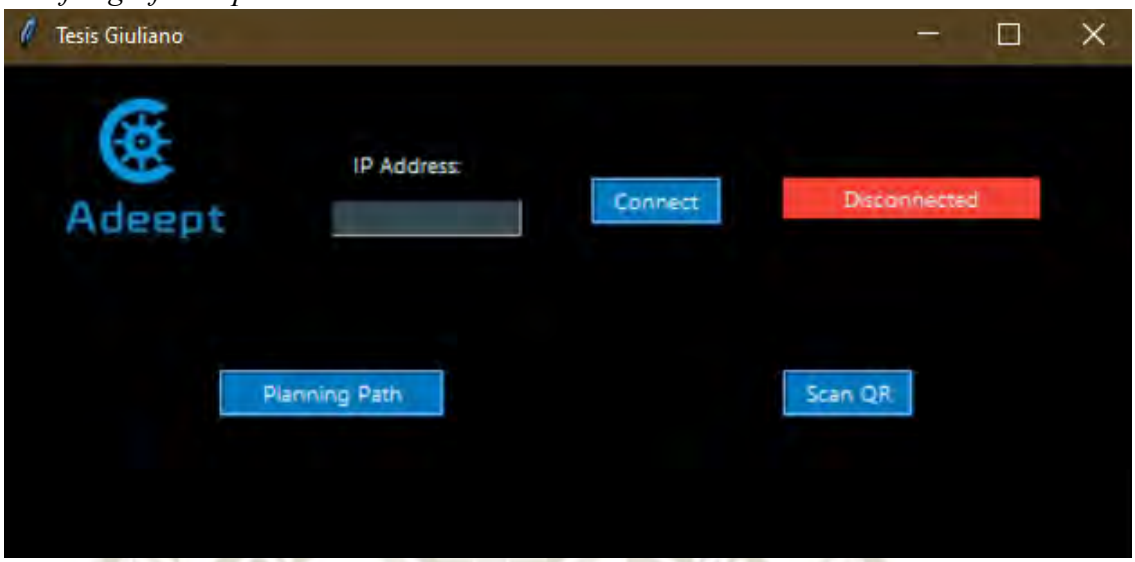
```
try:
        logo =tk.PhotoImage(file = 'logo2.png')          #Definir la foto del logo en formato '.png' o '.gif'
        l_logo=tk.Label(root,image = logo,bg=color_bg)   #Establecer la figura del logo para mostrar como etiqueta.
        l_logo.place(x=30,y=13)                          #Posicionar la etiqueta en la parte superior izquierda.
except:
        pass

l_ip_4=tk.Label(root,width=18,text='Disconnected',fg=color_text,bg='#F76745')
l_ip_4.place(x=550,y=50)                                                    #Definir una etiqueta y su posición

E1 = tk.Entry(root,show=None,width=16,bg="#37474F",fg='#eceff1')
E1.place(x=350,y=60)                                                        #Definir una entrada y su posición

l_ip_3=tk.Label(root,width=10,text='IP Address:',fg='#000000',bg='#FFFFFF')
l_ip_3.place(x=350,y=35)                                                    #Definir una etiqueta y su posición

Btn_path = tk.Button(root, width=15, text='Planning Path',fg=color_text,bg=color_btn,relief='ridge')#Define un botón y su posición
Btn_path.place(x=300,y=135)
Btn_path.bind('<ButtonPress-1>',path)


Btn_QR = tk.Button(root, width=8, text='Scan QR',fg=color_text,bg=color_btn,relief='ridge')#Define un botón y su posición
Btn_QR.place(x=500,y=135)
Btn_QR.bind('<ButtonPress-1>',call_camera, call_qr_code)


Btn_path = tk.Button(root, width=15, text='MOVE',fg=color_text,bg=color_btn,relief='ridge')#Define un botón y su posición
Btn_path.place(x=400,y=170)
Btn_path.bind('<ButtonPress-1>',call_follow_path)


Btn14= tk.Button(root, width=8,height=1, text='Connect',fg=color_text,bg=color_btn,command=connect_click,relief='ridge')
Btn14.place(x=465,y=50)           #Define un botón y su posición

root.bind('<Return>', connect)
```

Fuente. Realizamos la programación de algunos componentes de la interfaz gráfica, como son botones, cuadros de diálogo, entre otros. Adaptación propia en base a la investigación realizada (2020)

Después, con esta parte del diseño, realizaremos un test rápido para poder ver si nuestro programa compila de la manera adecuada, de no ser así realizaríamos la respectiva revisión del programa hasta encontrar los errores que no permitan que el programa de nuestra primera interfaz gráfica pueda compilar de manera adecuada, una vez revisado y corregidos los errores nos encargaremos de la parte más consistente de nuestra programación que vendrían a ser las funciones como tal de nuestro robot.

Por último, procederemos a mostrar el resultado obtenido luego de realizar nuestra programación final de la interfaz gráfica, junto con cada uno de los componentes necesarios en la siguiente figura de nuestro trabajo:

Figura 27
Interfaz gráfica – primer diseño



Fuente. Realización de la primera interfaz gráfica programada a través de Python. Adaptación propia en base a la investigación realizada (2020)

3.1.1.2. Escaneo de código QR

Partiendo de la idea que nuestro robot debe ser lo más sencillo posible y que deba operar en las condiciones más precarias posibles, es que en el presente trabajo se buscara considerar un robot que cuente con algún sistema no muy complejo, en cuanto a ubicación se refiere, es por esa misma razón que procederemos a usar un sistema de reconstrucción de escenarios a través de una cámara incorporada por el mismo robot. Así de esta manera utilizaremos el método de reconocimiento mediante la lectura de un código QR, sabiendo que este puede almacenar diferentes tipos de información, utilizando la cámara incorporada del robot podremos leer un código QR, el cual va a permitir que el robot pueda encontrar su posición exacta dentro de un espacio específico.

Por tanto, al momento en que el robot inicialice sus actividades, una de las primeras acciones a realizar debe ser posicionarse de una manera determinada apuntando hacia donde se encuentre el código QR que contendrá la información exacta de su posición en un espacio geográfico específico. En otras palabras, podríamos decir que el robot no solo encuentra una posición específica con dicha lectura, sino que encontrará un mapa del lugar en el que se encuentra para poder hacer, acto seguido, un planeamiento de ruta

específico que le permita a su vez determinar las acciones o movimientos a realizar una vez terminado con el paso anterior.

Figura 28
Lectura QR utilizando Python y OpenCV

Fuente. Ejemplo de una lectura de código QR a través de una cámara. Adaptado de " https://programmerclick.com/article/52581868360/ " por programador clic (s.f.)

Por consiguiente, el primer programa a tomar en cuenta es aquel que nos permita hacer una lectura verdadera del código QR, el cual contendrá dentro de su código una posición determinada. Prueba de ello son algunos resultados parecidos que podemos observar en trabajos realizados previamente, así como se muestra en la figura 28, donde se observa como la cámara identifica los códigos y logra descifrar la información que contienen los mismos.

3.1.1.3.Planificador de trayectorias

Después de realizar el análisis requerido para la programación del escaneo o reconocimiento de un código QR, se necesitará de un programa que nos permita hacer una guía de ruta con lo ya obtenido previamente mediante el código QR, para ello también haremos algunos análisis de los métodos necesarios a utilizar en el desarrollo de este programa; considerando que el programa debe de responder en un tiempo rápido para poder hacer que el robot tome decisiones adecuadas en cuestión de segundos o en el peor de los casos minutos.

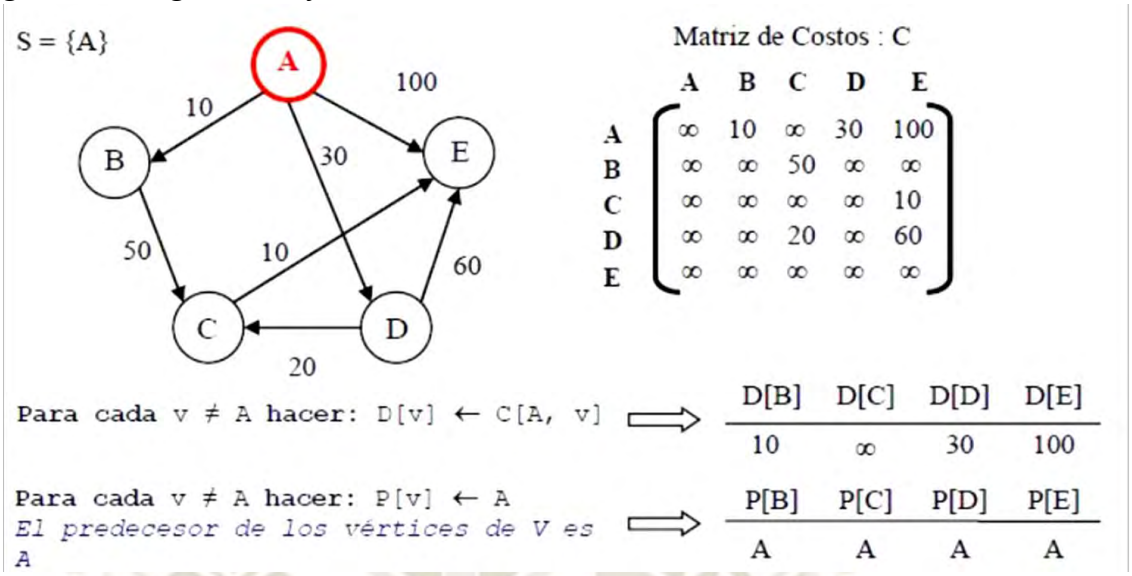
Por esta razón debemos de considerar los distintos algoritmos que existen para realizar un planificador de trayectoria, para poder elegir el más adecuado para nuestro proyecto, teniendo en cuento los más importantes.

a. Algoritmo de Dijkstra

Por lo tanto, tenemos una definición de lo que es el algoritmo de Dijktra, para poder comprender a mayor amplitud en que consiste dicho algoritmo y lo que nos permite realizar, es así que según “EcuRed” (2020), tenemos que:

También llamado algoritmo de caminos mínimos, es un algoritmo para la determinación del camino más corto dado un vértice origen al resto de vértices en un grafo con pesos en cada arista. Su nombre se refiere a Edsger Dijkstra, quien lo describió por primera vez en 1959 (Algoritmo de Dijkstra, párr.1).

Figura 29
Lógica de un algoritmo Dijkstra



Fuente. Ejemplo de la lógica de un algoritmo de Dijkstra y su funcionamiento. Adaptado de "Algoritmo de Dijkstra" por EcuRed (2020)

De acuerdo con la definición obtenida y con lo mostrado a través de la figura 29, podemos ver que el algoritmo de Dijkstra realiza una evaluación del punto de origen hacia los vértices que tiene cercanía, tratando de buscar el camino más corto hacía el destino final. Por tanto, el algoritmo de Dijkstra es una alternativa a tomar en cuenta al momento que se desee elaborar el planificador de trayectorias de nuestro programa, sabiendo que el desarrollo del mismo podría ser favorable con este algoritmo, pero a su vez tiene ciertas desventajas en cuanto a pasos o tiempo que le toma al propio algoritmo de Dijkstra para poder elaborar una lógica con la comunicación que tiene con cada uno de los nodos.

b. Algoritmo para búsqueda A*

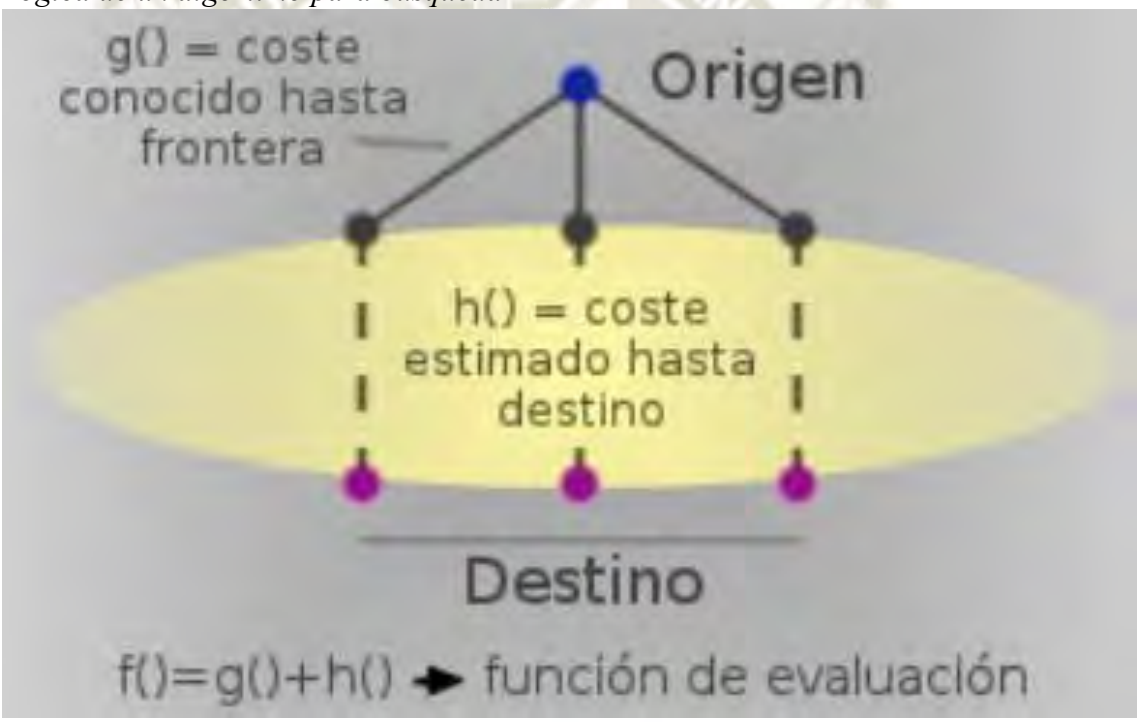
Por otra parte, tenemos el algoritmo para búsqueda A*, y para poder determinar su correcto funcionamiento tenemos que tener una definición que nos permita comprender su correcto funcionamiento, de esta manera según IDELab (2020), obtenemos lo siguiente:

El algoritmo A* es un algoritmo de búsqueda que puede ser empleado para el cálculo de caminos mínimos en una red. Se va a tratar de un algoritmo heurístico, ya que una de sus

principales características es que hará uso de una función de evaluación heurística, mediante la cual etiquetará los diferentes nodos de la red y que servirá para determinar la probabilidad de dichos nodos de pertenecer al camino óptimo.

Esta función de evaluación que etiquetará los nodos de la red estará compuesta a su vez por otras dos funciones. Una de ellas indicará la distancia actual desde el nodo origen hasta el nodo a etiquetar, y la otra expresará la distancia estimada desde este nodo a etiquetar hasta el nodo destino hasta el que se pretende encontrar un camino mínimo (Algoritmo A*, párr.1).

Figura 30
*Lógica de un algoritmo para búsqueda A**



Fuente. Ejemplo de la lógica de un algoritmo para búsqueda A* y su funcionamiento. Adaptado de "Algoritmo para búsqueda A*" por Fernando P. (2018)

Por consiguiente, podemos observar que el algotimo para búsqueda A* nos permite tener un análisis de cada nodo, teniendo en consideración un punto de partida y un punto de llegada, haciendo que la búsqueda de la trayectoria sea rápida de evaluar al momento de tomar decisiones.

En consecuencia podemos deducir que la mejor alternativa es la que nos presenta el algoritmo para búsqueda de A*, ya que cuenta con algunas ventajas por encima del algoritmo de Dijkstra, en otras palabras con el hecho de contar con una etiqueda para cada nodo, nos damos que esto le permitirá al programa tomar decisiones con mayor rapidez y encontrar una trayectoria apropiada en un lapso de tiempo menor.

3.1.1.4.Movimientos del Robot

Finalmente, una vez analizado todos los paráetros anteriores, debemos de tener en cuenta como se efectuaran los movimientos de nuestro robot, esto puede variar según nuestro sujeto de prueba, debido a que muchos robots cuentan con una programación respectiva para sus movimientos. No obstante si nuestro sujeto de prueba no cuenta con un progrma previo, procederemos a realizar la programación de nuestros movimientos a traves del programa Python.

Sin embargo, si el robot deseado tienen un programa ya establecido, intentaremos adaptarlo a nuestras necesidades y una vez efectuada las modificaciones procederemos a unir todos los programas requeridos por este análisis previo y tratar de obtener los resultados necesarios para nuestro robot. De manera que una vez culminado todo el proceso de diseño, podamos realizar las pruebas correspondientes.

3.1.2.1.Estructura

Partiendo de todo lo visto anteriormente pasaremos a realizar la selección de una estructura apropiada para hacer las simulaciones correspondientes de nuestro diseño de control de navegación, para lo cual compararemos algunos prototipos interactivos que podemos encontrar en el mercado y de entre ellos elegiremos al mejor.

Tabla 4

Modelos de estructuras en forma de cuadrúpedos a escoger para la simulación

	Modelo 1	**Modelo 2**	**Modelo 3**	**Modelo 4**
Nombre del modelo	SunFounder Kit de rastreo robot cuadrúpedo DIY para Arduino con Nano Board Remote Control (Negro)	Adeept DarkPaw Bionic Quadruped Spider Robot Kit for Raspberry Pi 4/3 Model B+/B/B, STEM Crawling Robot, OpenCV Tracking Self-stabilizing	Robot de tortuga 9 DOF/juego completo de accesorios de robot cuadrupedos/bió nico/plataforma de laboratorio/enseñ anza	Kit Araña Cuadrúpeda
Microprecesador/ Microcontrolador	ARDUINO	Raspberry	Propio	ARDUINO
Número de patas	4	4	4	4
Estabilidad propia	No Especifica	Si	No Especifica	No Especifica
Cámara	No	Si	No	No
Sensores	No	Si	No	No
Servomotores	12	14	9	8
Baterías	No	Si	No Especifica	Si
Precio	$233.72	$104.90	$170.68	$53.25

Fuente. Se realizo un análisis de especificaciones de cada modelo para poder determinar cuál es la mejor opción a tomar para realizar la validación de nuestro diseño de control de navegación. Adaptación propia en base a la investigación realizada (2020)

Tomando en cuenta la tabla 4, se utilizará el modelo 2 el cual es el DarkPaw *Bionic Quadruped Spider Robot Kit for Raspberry Pi 4/3*, marca *Adeept*, el cual consta de cuatro extremidades acrílicas, una cámara Raspberry Pi, un módulo de sensor giroscópico MPU6050, un faro LED y 14 servomotores MG90S. Más aún el modelo elegido permite tener incorporado un microcontrolador de la marca *Raspberry*, lo que nos permitiría trabajarlo a través de esa placa de control.

Figura 31
Adeept DarkPaw Bionic Quadruped Spider Robot Kit

Fuente. Imagen del modelo del posible robot a controlar. Adaptado de "Adeept DarkPaw Bionic Quadruped Spider Robot Kit for Raspberry Pi 4/3 Model B+/B/B, STEM Crawling Robot, OpenCV Tracking Self-stabilizing" por Adeept (2019)

3.1.2.2.Hardware

a. Placa de control:

De esta manera pasaremos a realizar la selección de una placa de control, para lo cual realizaremos una comparación entre las placas más comerciales dentro de la industria de la electrónica para poder obtener el que se adapte mejor a nuestras especificaciones, sabiendo que la placa de control es un elemento importante dentro de nuestros componentes bases para realizar las pruebas necesarias en nuestro robot, debió a que en la placa de control será donde almacenaremos nuestro programa, siendo este el cerebro de nuestro robot. Por lo tanto, realizaremos una selección de la placa de control dentro de una serie de opciones presentes en la siguiente tabla:

Tabla 5
Comparación de las placas de control

	ARDUINO	**Raspberry Pi 4**	**Pic Tarjeta de**
Procesador	AT91SAM3X8E	Broadcom	Pic18f2580
Pins	54	40	22
RAM	96 KB	1GB/2GB/4GB	32 KB
Clock Speed	84 MHz	1.5GHZ	32 MHz
USB	No	4	No
USB - C	No	1	1
Wifi/Bluetooth	No/No	Si/Si	No/No
Precio	$38.50	$85.32	$23.70

Fuente. Comparación de detalles y especificaciones de las placas de control. Adaptacion propia en base a la investigacion realizada (2019)

En resumidas cuentas, hemos optado por usar una placa de control de la marca *Raspberry Pi* en su modelo *4B* con un procesador *Broadcom*, la elección tiene que ver principalmente con las prestaciones que presenta esta marca en su modelo *4B*, prueba de ello es la cantidad de puertos USB que contiene, la capacidad de memoria RAM, su chip de Wifi y Bluetooth y la velocidad de procesamiento; incluso esta placa de control es con la que trabaja nuestro robot seleccionado en el apartado anterior.

b. Fuente de poder

Por consiguiente, y tomando en cuenta los componentes previamente elegidos, pasaremos a elegir la fuente de poder apropiada para poder realizar la prueba de nuestro robot. Una de las especificaciones más importantes que debemos de considerar es el hecho que la fuente de poder debe brindarle autonomía a nuestro robot. Dentro de las posibilidades, y teniendo en cuenta el diseño y especificaciones, llegamos a la conclusión que la única fuente de poder viable para nuestro proyecto serían baterías ICR 18650 de litio de 2000 mAh y 3.7 V, no solo por su capacidad de voltaje y amperaje, sino también porque son recargables y compatibles con el compartimiento que viene con el robot.

Uno de los inconvenientes que nos brindará la fuente de poder es el tiempo de duración y autonomía que nos brindará la misma, la cual va a ser un promedio de 2 a 3 horas consecutivas sin necesidad de recargarlas, pero permitirá que el robot se pueda desplazar libremente sin necesidad de extensiones.

Figura 32
Baterías ICR18650 de Litio

Fuente. Fotografía de las baterías ICR18650 de litio de 200 mAh y 3.7 V. Adaptado de "Icr18650 2000mAh 3.7V Rechargeable Battery 18650 3.7V Battery 2000 mAh Battery" por Made in China Connecting Buyers with Chinese Suppliers (2020)

c. Memorias de almacenamiento externo

Para terminar, procederemos a elegir un tipo de almacenamiento externo, dentro de la amplia gama de almacenamientos externos que existen procederemos a elegir entre dos, porque son los que nos permite tener la placa de control seleccionada. Es así que nuestras opciones se resumen en dos, la primera es una memoria USB y la segunda es una memoria micro SD. Teniendo en cuenta que la placa de control seleccionada tiene ambos puertos, pero que la estructura del robot tiene una forma establecida y que sería mejor evitar algún tipo de interrupción con su correcto funcionamiento, es que vamos a optar por la opción de utilizar una memoria micro SD, de modo que no produciremos ningún tipo de inconveniente en el robot al momento de desplazarse.

Figura 33
Memoria micro SD

Fuente. Fotografía de una memoria micro SD de clase 10. Adaptado de "Kingston SDCS2/16GB micro SD XC clase 10 16GB c/a" por MiPc OK (2020)

Para simplificar podríamos decir que las conclusiones del análisis inicial nos dan como resultado las especificaciones necesarias a considerar para el diseño de nuestro programa y tener una idea más clara de cómo debe de procederse en la programación del sistema

de control de navegación de nuestro robot, teniendo una estructura clara y conceptos básicos que nos permitan desarrollar un apropiado programa.

Por otra parte, también este análisis nos ha ayudado a tomar en cuenta las especificaciones y la selección final de los componentes base que debe tener nuestro robot en el cual realizaremos las pruebas de nuestro diseño más adelante.

Por último, con todo lo analizado procederemos a realizar el diseño de nuestro programa, el cual detallaremos en el siguiente punto.

CAPÍTULO IV

4. DISEÑO

4.1.Diseño

Partiendo del análisis inicial realizado previamente, procederemos a realizar el diseño de cada programa necesario por separado con el fin de comprobar su real funcionamiento de manera independiente, para luego poder unirlos todos en nuestra interfaz gráfica, y una vez compile el programa de manera apropiada poder realizar las pruebas dentro de nuestro robot.

De esta manera y siguiendo la misma estructura que en el análisis inicial, procederemos con programar de una manera estructurada y siguiendo el mismo orden, es así que en un primer momento realizaremos el diseño de nuestra interfaz gráfica, para luego proceder con el reconocimiento del código QR, el planificador de trayectorias y por último con el seguimiento del camino a través de los movimientos propios del robot.

4.1.1. Interfaz gráfica

Para comenzar nos basaremos en la programación inicial realizada en la interfaz gráfica base que conseguimos en el análisis inicial, de ahí procederemos a terminar la estructura, forma y especificaciones necesarias para nuestro proyecto.

Por tanto, necesitaremos determinar los colores, la forma, dimensiones y elementos necesarios para nuestra interfaz gráfica, considerando que esta interfaz debe ser fácil de entender y manipular para cualquier persona que vaya a ser su operador y que al momento de manipular el robot no sea necesario llamar a un experto en la materia para realizar las funciones necesarias de nuestro robot.

Figura 34

Programa de la interfaz gráfica

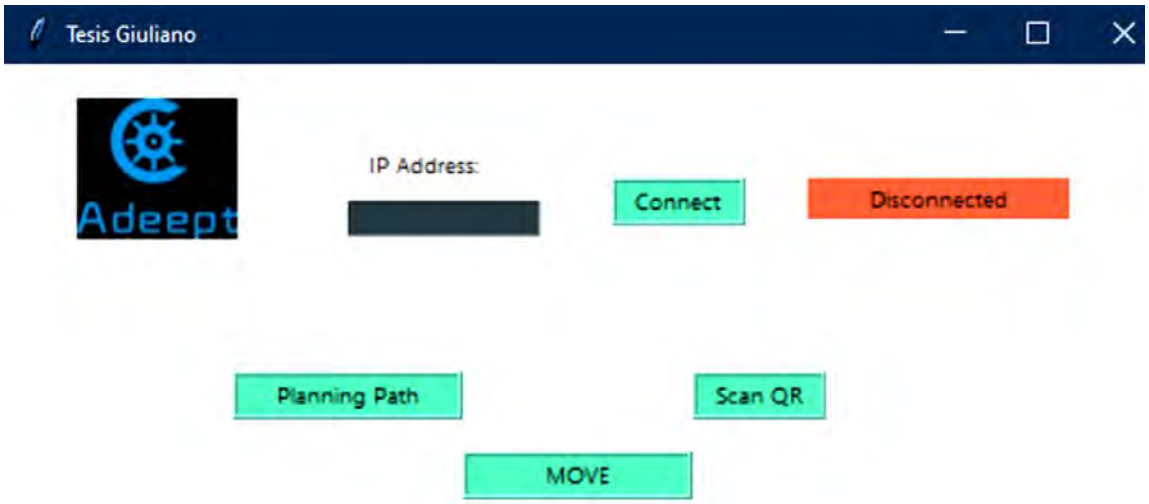
```
l_ip_4=tk.Label(root,width=18,text='Disconnected',fg=color_text,bg='#F76745')
l_ip_4.place(x=550,y=50)                                          #Definir una etiqueta y su posición

E1 = tk.Entry(root,show=None,width=16,bg="#37474F",fg='#eceff1')
E1.place(x=350,y=60)                                              #Definir una entrada y su posición

l_ip_3=tk.Label(root,width=10,text='IP Address:',fg='#000000',bg='#FFFFFF')
l_ip_3.place(x=350,y=35)                                          #Definir una etiqueta y su posición

Btn_path = tk.Button(root, width=15, text='Planning Path',fg=color_text,bg=color_btn,relief='ridge')#Define un botón y su posición
Btn_path.place(x=300,y=135)
Btn_path.bind('<ButtonPress-1>',path)



Btn_QR = tk.Button(root, width=8, text='Scan QR',fg=color_text,bg=color_btn,relief='ridge')#Define un botón y su posición
Btn_QR.place(x=500,y=135)
Btn_QR.bind('<ButtonPress-1>',call_camera, call_qr_code)


Btn_path = tk.Button(root, width=15, text='MOVE',fg=color_text,bg=color_btn,relief='ridge')#Define un botón y su posición
Btn_path.place(x=400,y=170)
Btn_path.bind('<ButtonPress-1>',call_follow_path)



Btn14= tk.Button(root, width=8,height=1, text='Connect',fg=color_text,bg=color_btn,command=connect_click,relief='ridge')
Btn14.place(x=465,y=50)           #Define un botón y su posición

root.bind('<Return>', connect)
```

Fuente. Programa de la interfaz gráfica en Python, donde definimos algunas especificaciones necesarias. Adaptación propia en base a la investigación realizada (2020)

Acto seguido de la programación debemos de comprobar que los elementos cambiados y añadidos a nuestra programación inicial funcionan de manera adecuada con respecto a nuestra interfaz gráfica, si esta funciona de manera apropiada continuaremos con nuestro siguiente programa, sino de caso contrario procederemos a realizar las modificaciones pertinentes que permitan el correcto funcionamiento de nuestra interfaz gráfica.

En otras palabras, debemos de ejecutar el programa realizado con las variaciones que hemos procedido a realizar durante este paso y el programa debe de ejecutarse sin presentar ningún inconveniente, no obstante, si el programa no corre debemos de hacer un análisis del programa para poder encontrar el error que no permite el correcto funcionamiento de nuestro programa. Sorprenderá comprobar que un buen editor de textos nos ayuda con eficacia a poder encontrar los errores con mayor facilidad.

Figura 35
Interfaz gráfica diseñada y modificada



Fuente. Interfaz gráfica del proyecto programada en Python y lanzada en la plataforma proporcionada por el proveedor. Adaptación propia en base a la investigación realizada (2020)

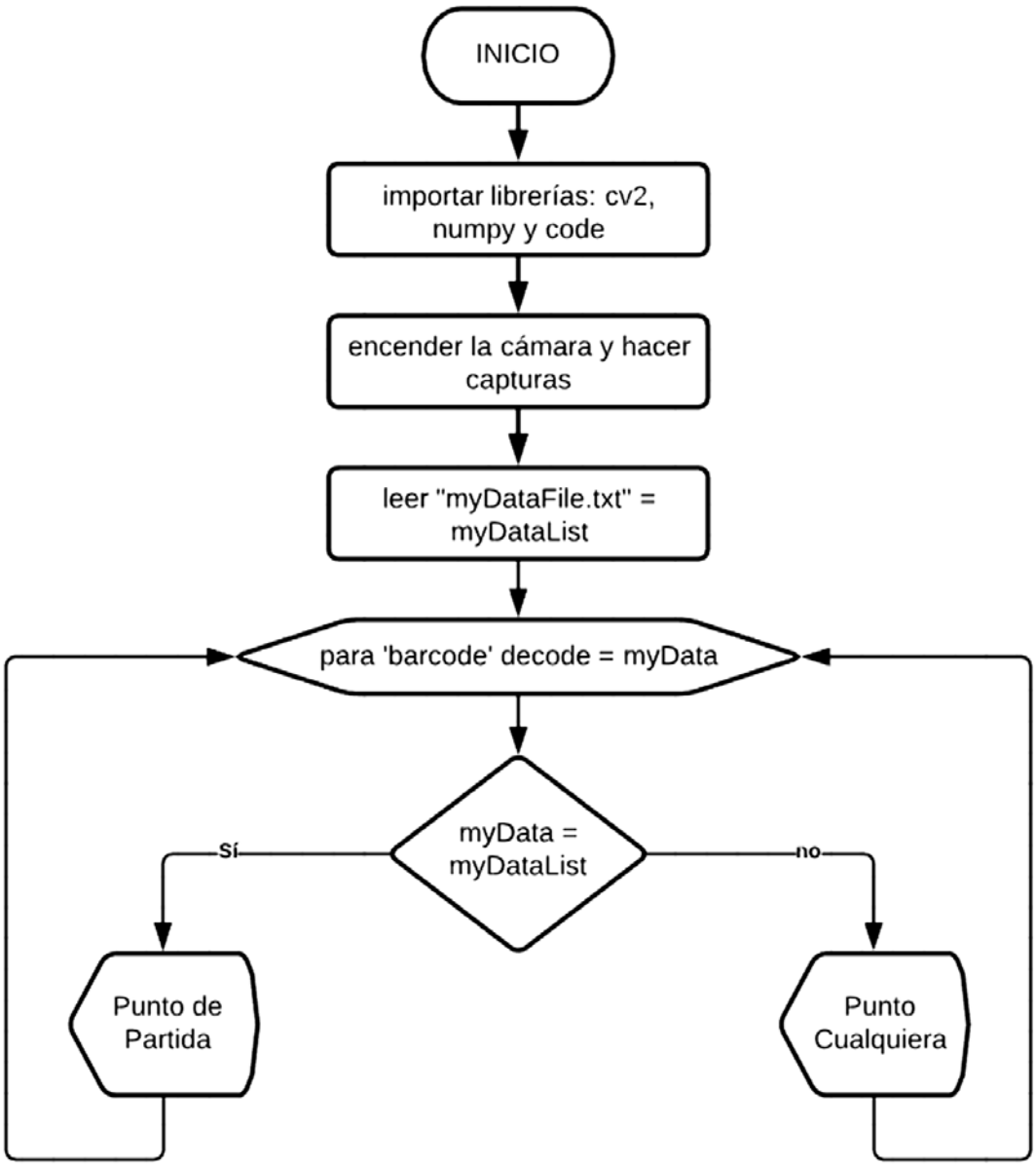
En efecto el programa ha funcionado de manera apropiada y hemos obtenido una interfaz gráfica deseada que nos permita manipular nuestro robot de una manera sencilla y adecuada como se muestra en la figura 35. Por lo tanto, procederemos a realizar el siguiente programa deseado para nuestra presente investigación.

4.1.2. Programa para el escaneo del código QR

Puesto que la interfaz gráfica se encuentra operativa, procederemos a realizar la programación del sistema de escaneo del código QR mediante una cámara web, logrando en el proceso final poderla adaptar a nuestro robot. En otras palabras, primero realizaremos una programación aislada de solo este programa, observando si los resultados obtenidos son los deseados y una vez se compruebe su correcto funcionamiento procederemos a realizar la unión de los programas.

Asimismo, como en el caso anterior la programación se realizará en Python, pero en esta ocasión haremos uso de la herramienta llamada OpenCv que es de dominio público para realizar las pruebas correspondientes.

Figura 36
Diagrama de Flujo del sistema QR



Fuente. Diagrama de flujo, para entender el funcionamiento del código QR. Adaptación propia en base a la investigación realizada (2020)

Partiendo de este diagrama de flujo podemos entender con mayor precisión lo requerido al momento de realizar nuestra programación en Python, además que el

diagrama de flujo nos permite entender lo que debemos de hacer al momento de realizar nuestra programación.

De ahí que observamos como es que este programa debe de tener un proceso ordenado de tal manera que el programa pueda ejecutarse de forma correcta sin presentar inconvenientes, también podemos ver que nuestro programa va a necesitar el llamado de librerías, nombrar variables, comparar datos, ejecutar un bucle y darnos una respuesta determinada por nosotros mismos que nos permita ver si al final el programa está realizando la acción requerida de manera apropiada. Por tanto, comprobaremos si el diagrama de flujo graficado en la figura 36 tiene sentido y un correcto funcionamiento al momento de realizar la programación y ejecutar el programa para realizar el test del mismo.

Por ello, necesitamos elaborar una programación adecuada que nos permita llevar a cabo la ejecución del programa planteado a través del diagrama de flujo, este nos permite tener una comprensión más ordenada de las acciones a realizar y los parámetros a tomar en consideración al momento de escribir cada línea de programación en nuestro editor de textos.

Acto seguido debemos realizar nuestra programación, tomando en cuenta lo establecido por nuestro diagrama de flujo; una vez realizada la programación según nuestros requerimientos procederemos a comprobar su correcto funcionamiento, como ya hemos detallado y observado, anteriormente, vamos a hacer uso de librerías que tendrán que ser llamadas y ejecutadas, de ahí que, para este siguiente paso procederemos a ayudarnos de un programa llamado Sublime Text 3, el cual nos permitirá realizar la escritura del programa con libertad y eficiencia, cometiendo menos errores que los que se nos pueden presentar al realizar la programación en un block de notas.

De igual forma que en el programa anterior necesitamos ejecutar nuestro programa para poder comprobar su correcto funcionamiento y que no haya ningún error al momento de realizar las pruebas pertinentes en el siguiente capítulo. A pesar de todo, si el programa presenta dificultades, anomalías o incongruencias al momento de hacer la ejecución del mismo, tendremos que proceder a realizar una observación detallada de cada línea de programación con la finalidad de corregir el problema.

Figura 37

Programa del escáner del código QR

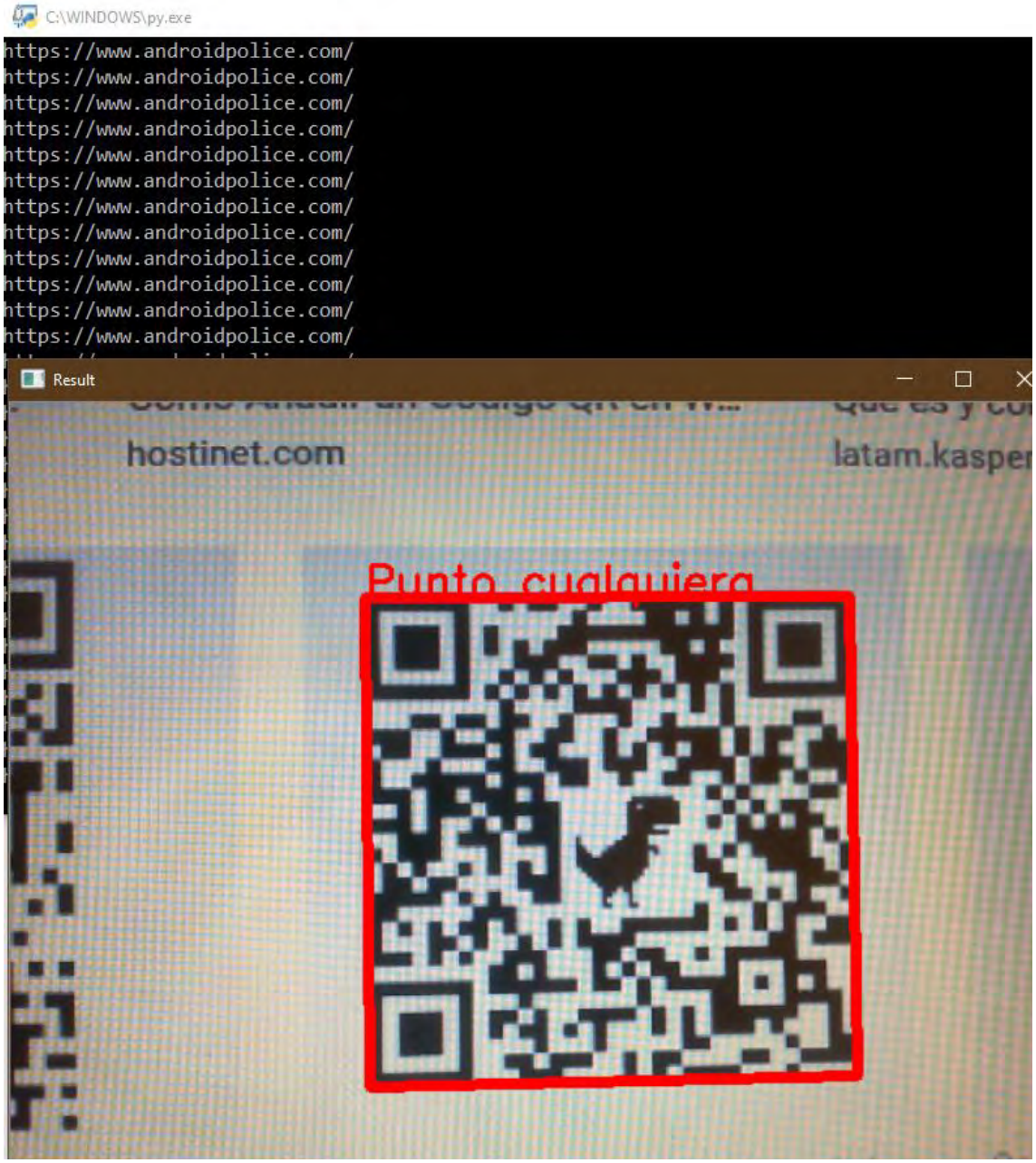
```
import cv2
import numpy as np
from pyzbar.pyzbar import decode

cap = cv2.VideoCapture(0)
cap.set(3,640)
cap.set(4,480)

with open('myDataFile.txt') as f:
    myDataList = f.read().splitlines()

while True:

    success, img = cap.read()
    for barcode in decode(img):
        myData = barcode.data.decode('utf-8')
        print(myData)

        if myData in myDataList:
            myOutput = 'Punto de partida'
            myColor = (0,255,0)
        else:
            myOutput = 'Punto cualquiera'
            myColor = (0, 0, 255)

        pts = np.array([barcode.polygon],np.int32)
        pts = pts.reshape((-1,1,2))
        cv2.polylines(img,[pts],True,myColor,5)
        pts2 = barcode.rect
        cv2.putText(img,myOutput,(pts2[0],pts2[1]),cv2.FONT_HERSHEY_SIMPLEX,
                    0.9,myColor,2)

    cv2.imshow('Result',img)
    cv2.waitKey(1)
```

Fuente. Programa realizado para el escáner del código QR a través de una cámara web en Python. Adaptación propia en base a la investigación realizada (2020)

De acuerdo con el programa establecido deberíamos poder compilar el programa sin ningún problema y el resultado obtenido debe ser el funcionamiento de nuestra cámara web para que luego esta realice un reconocimiento del código QR, para lo cual nosotros hemos almacenado un dato específico el cuál identificará la cámara y el programa determinará si el dato almacenado en el código QR es el mismo que tenemos almacenado dentro de nuestro programa, para identificar si el programa está reconociendo o no, el dato requerido, este procederá a mostrarnos un mensaje en la pantalla.

Figura 38

Prueba del programa de escaneo QR



Fuente. Primer test del programa de escáneo de un código QR, donde se muestran los resultados obtenidos. Adaptación propia en base a la investigación realizada (2020)

Como se indicó, obtuvimos el resultado deseado, procedimos a escanear con la cámara web un código QR cualquiera de internet y el programa reacciono de la manera deseada, siendo este que en su análisis puede reconocer la información almacenada en el código QR, como se muestra en la figura 38, pero no reconoce este código como una entrada

válida según lo especificado en nuestra programación, ya que dentro del archivo creado para el almacenamiento tenemos una información diferente a la brindada dentro del código QR escaneado.

Después de comprobar el correcto funcionamiento de nuestro programa para escanear un código QR, procederemos con la siguiente parte de nuestro diseño.

4.1.3. Programa para el planificador de trayectorias

En otro orden de cosas, pasaremos a realizar la programación para nuestro planificador de trayectorias o *path planning*, para lo cual necesitaremos diseñar nuestros mapas en forma de grillas, facilitando la ubicación de nuestro robot en un espacio geográfico determinado, en este mapa, al igual que en cualquier otro, debemos determinar nuestros obstáculos existentes, de manera que nuestro planificador de trayectorias encuentre el resultado deseado a nuestras exigencias.

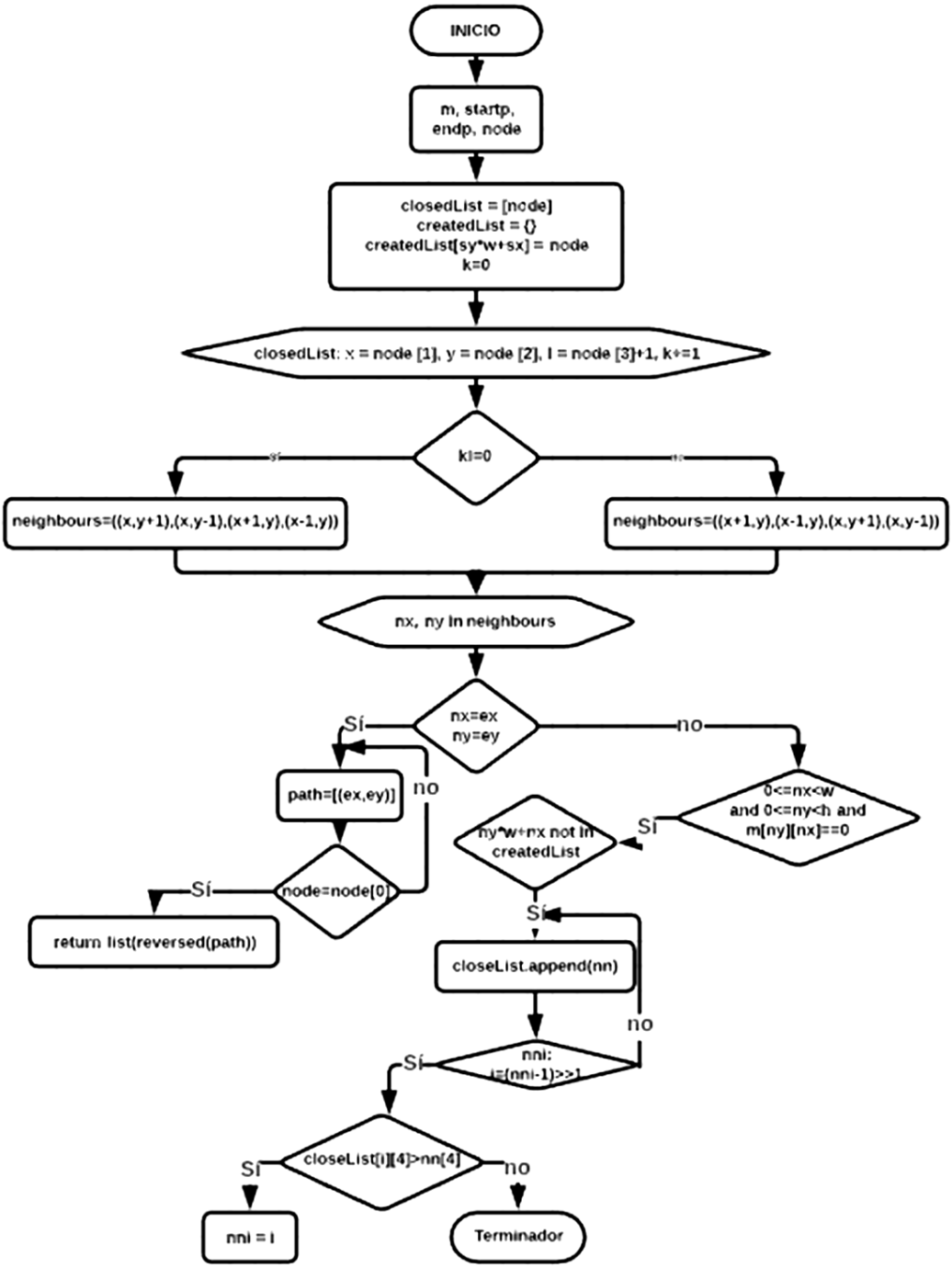
Figura 39
Diseño del mapa

```
maze = [[0 for i in range(10)] for i in range(10)]
obstacles = [(1,2),(2,2),(4,9),(1,9),(1,8),(3,3),(4,4),(5,5),(3,4),(4,3)]

for i in obstacles:
    maze[i[0]][i[1]]=1
    print("obstaculos")
for i in range (10):
    for j in range (10):
        print(maze[i][j])
```

Fuente. Realización de un mapa virtual mediante el programa de Python. Adaptación propia en base a la investigación realizada (2020)

Después de realizar el diseño de un mapa con sus respectivos obstáculos, procedemos a desarrollar el algoritmo a* que nos permitirá hacer una búsqueda del camino en el mapa elaborado, con el cual vamos a poder encontrar el camino que nos permita tener el menor costo posible.

Figura 40
*Diagrama de Flujo del Algoritmo a**



Fuente. Diagrama de flujo, para entender el algortimo a* realizado. Adaptación propia en base a la investigación realizada (2020)

De esta manera podemos observar en la figura 40 el diagrama de flujo elaborado por nosotros, donde encontramos el procedimiento que debe seguir nuestro programa al momento de elaborarse y ejecutarse; en primer lugar, podemos observar el llamado y el nombramiento de variables a utilizar, para este caso también usaremos comandos de decisión, así como también bucles que nos permitan estar comparando la información y tomado decisiones de manera contante en nuestro programa.

Para ilustrar mejor lo realizado en nuestro diagrama de flujo, procederemos a realizar nuestro programa con la ayuda de Sublime Text 3.

Figura 41

*Programa del algoritmo a**

```
#PROYECTO DE TESIS
#CREADOR: GIULIANO MONTALVO
#ALGORITMO A*
# -*- coding: utf-8 -*-

def astar(m,startp,endp):
    w,h = 10,10     # grilla (10x10) es el número de entrada de nuestro mapa
    sx,sy = startp  #Punto de Inicio
    ex,ey = endp    #Punto de Llegada

    node = [None,sx,sy,0,abs(ex-sx)+abs(ey-sy)]
    closeList = [node]
    createdList = {}
    createdList[sy*w+sx] = node
    k=0
    while(closeList):
        node = closeList.pop(0)
        x = node[1]
        y = node[2]
        l = node[3]+1
        k+=1
        #encontrar las grillas vecinas
        if k!=0:
            neighbours = ((x,y+1),(x,y-1),(x+1,y),(x-1,y))
        else:
            neighbours = ((x+1,y),(x-1,y),(x,y+1),(x,y-1))
        for nx,ny in neighbours:
            if nx==ex and ny==ey:
                path = [(ex,ey)]
                while node:
                    path.append((node[1],node[2]))
```

Fuente. Realización del programa para el algoritmo a* en Python. Adaptación propia en base a la investigación realizada (2020)

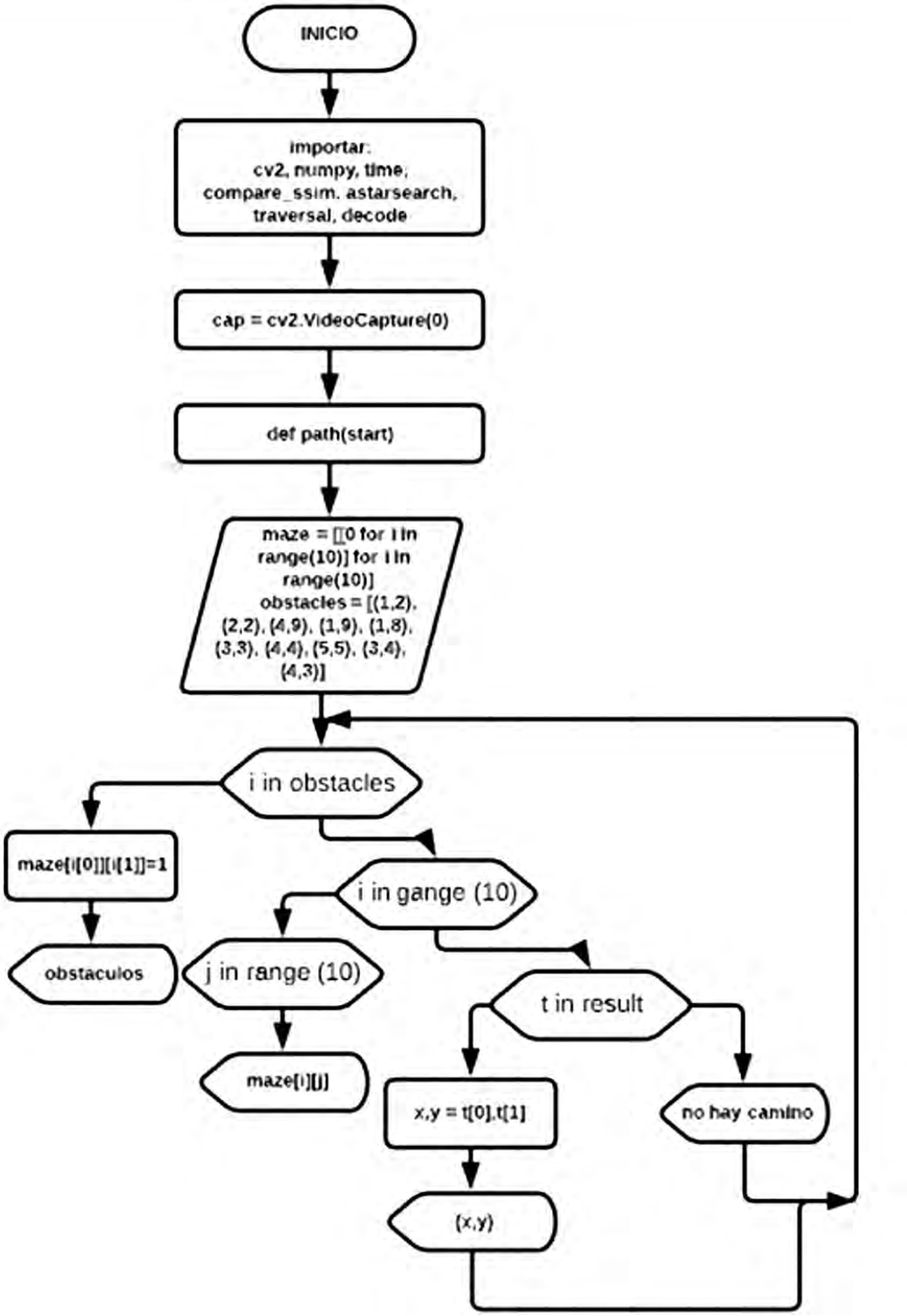
Así pues, hemos podido realizar el programa de nuestro algoritmo a*, el cual nos permitirá hacer los cálculos necesarios para poder encontrar la ruta más segura y de menor costo al momento de realizar la planificación de trayectorias. Este es el cerebro, por así decirlo, de nuestro programa que se encargará de realizar la planificación de la trayectoria, sin este algoritmo sería casi imposible lograr armar una trayectoria mediante la programación. Asimismo, cuando realicemos el programa del planificador de trayectorias, podremos comprobar con mayor asertividad, si el algoritmo a* fue el indicado a escoger al momento de hacer la elección de los parámetros necesarios. Por ejemplo, si el programa del planificador de trayectorias no encuentra un camino a seguir o no corre, habría que realizar el análisis pertinente de ambas partes para determinar si el error es del algoritmo elegido o ha sido por algún error cometido al momento de redactar las líneas de programación del planificador de trayectorias.

Por otro lado, realizamos el programa de ejecución de la planificación de la trayectoria, en conjunto con el código realizado en la sección anterior para poder determinar el camino a seguir a partir de la lectura de un código QR en el cual contenga la información del punto de partida. Por ejemplo, si la cámara del robot lee una coordenada X, el planificador de trayectorias, teniendo en cuenta los obstáculos y la ruta libre, debería de darnos como resultado un conjunto de coordenadas a seguir el cual será el camino más apropiado y encontrado por este programa, incluso las coordenadas deberán ser mostradas en una manera ordenada para determinar que el programa está funcionando de la manera más optima y adecuada.

Asimismo, procederemos a realizar un diagrama de flujo que nos permita tener claro el camino a seguir para la programación de nuestro planificador de trayectorias, en el cual incorporaremos el programa ya realizado para el algoritmo a*, también, este diagrama de flujo deberá de considerar las librerías pertinentes a utilizar en el momento que vayamos a realizar la programación del planificador de trayectorias.

Figura 42

Diagrama de Flujo del Planificador de Trayectorias



Fuente. Diagrama de flujo, para entender el planificador de trayectorias de nuestro programa. Adaptación propia en base a la investigación realizada (2020)

Como se indicó hemos realizado un diagrama de flujo, el cual nos permite tener un panorama más claro de lo que queremos obtener al realizar la programación de nuestro planificador de trayectorias, este diagrama mostrado en la figura 42 nos permite observar de manera ordenada como debe ser el real funcionamiento de nuestro programa, teniendo en cuenta la manera de llamar e implementar la librería, creada por nosotros, llamada astarsearch.

Por lo tanto, procederemos a realizar la programación en la plataforma de Sublime Text 3.

Figura 43

Programa del Planificador de Trayectoria

```
#PROYECTO DE TESIS
#Programa de lectura de codigo QR y planificador de recorrido
#CREADOE: GIULIANO MONTALVO
import cv2
import numpy as np
import time
from skimage.measure import compare_ssim as ssim
#el algoritmo a* es el líder de busqueda de caminos
#es utilizado en juegos como warcraft III
#usa el costo (menor distncia recorrida, menos tiempo, etc)
#genera posibilidades y elige la que tiene el menor costo proyectado
import astarsearch
#nos ayudará a pasar la imagen de derecha a izquierda para procesarla
import traversal
from pyzbar.pyzbar import decode

cap = cv2.VideoCapture(0)
cap.set(3,640)
cap.set(4,480)

def path(start):


    maze = [[0 for i in range(10)] for i in range(10)]
    obstacles = [(1,2),(2,2),(4,9),(1,9),(1,8),(3,3),(4,4),(5,5),(3,4),(4,3)]

    for i in obstacles:
        maze[i[0]][i[1]]=1
        print("obstaculos")
    for i in range (10):
        for j in range (10):
            print(maze[i][j])
```

Fuente. Programa del planificador de trayectorias completo, en conjunto del programa de reconocimiento de código QR. Adaptación propia en base a la investigación realizada (2020)

Entonces, hemos logrado realizar el programa requerido con éxito, demostrando que lo planteado, por nosotros mismos, en el diagrama de flujo es posible de realizar, cabe resaltar que el programa ha sido realizado con la incorporación del código QR, y este compila sin ningún problema.

Por lo tanto, con el programa desarrollado procederemos a realizar el siguiente paso a seguir, que sería la realización de los movimientos que debe realizar nuestro robot al momento que este reciba la información necesaria con respecto a una coordenada determina, un mapa elaborado y una trayectoria específica a seguir. No obstante, para ese siguiente paso procederemos a realizarlo de manera autónoma para corroborar su correcto funcionamiento.

Finalmente podemos decir que tenemos un programa para el planificador de trayectorias que funcione de manera correcta y demuestre que el algoritmo elegido en el capítulo anterior es el indicado para encontrar un camino a recorrer, que sea rápido y eficiente, y que de esta misma manera nos permita realizar o planificar la trayectoria al momento de realizar la lectura de un código QR, viendo de esta manera que el programa funciona sin ninguna dificultad.

4.1.4. Programa para el seguimiento del camino

Después de haber realizado nuestros programas para el reconocimiento de código QR y el de planificador de trayectorias, procederemos a realizar un programa que nos permita tomar las decisiones correspondientes con respecto a la ruta trazada, haciendo que el robot no solo planifique el camino, sino que en base a ello pueda recorrer el camino de manera autónoma con los datos ya ingresados.

Conviene recordar que para esta parte no realizaremos la programación de los movimientos, como tal, sino que aprovecharemos el programa que nos provee el fabricante, y de esta manera entender el correcto funcionamiento del robot.

Antes de empezar con la programación y el diagrama de flujo correspondiente para esta parte, procederemos a realizar una retroingeniería con el fin de entender el correcto funcionamiento de los movimientos de nuestro robot.

Figura 44

Programa de movimientos

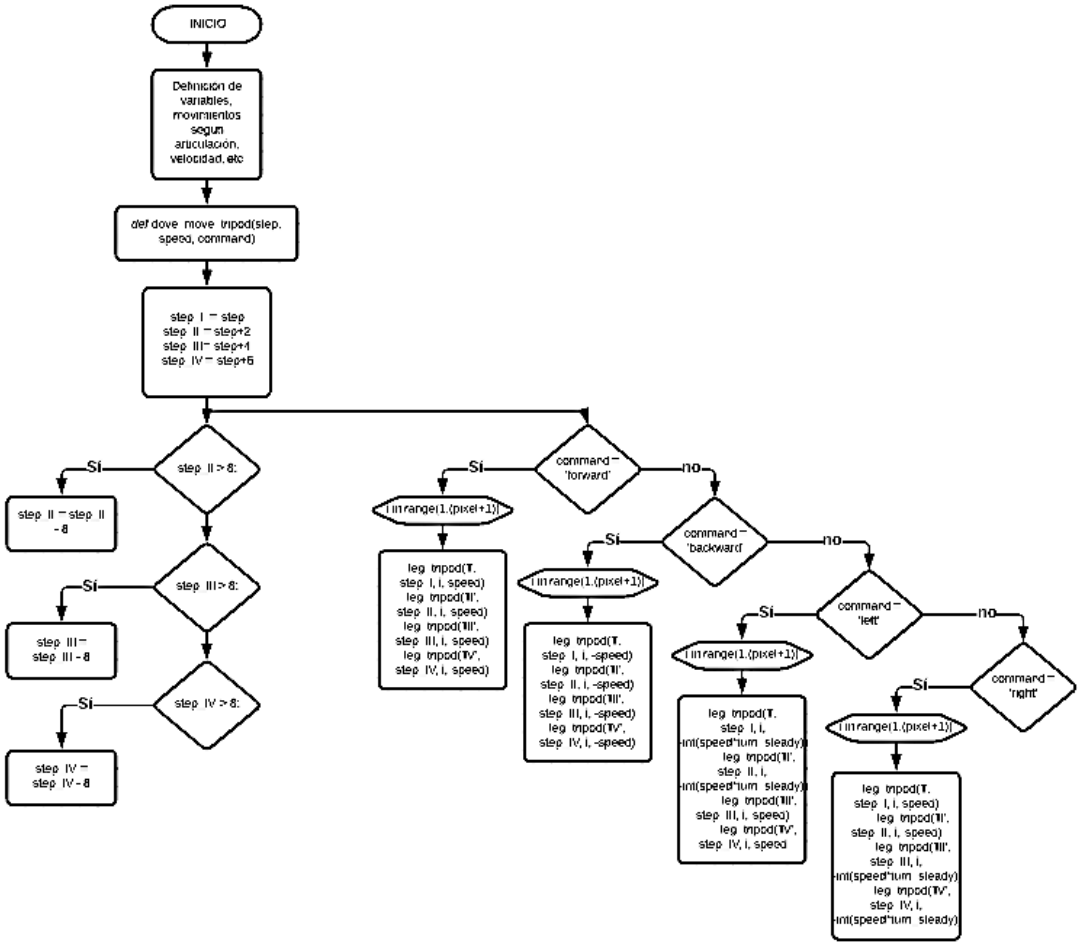
```
def dove_move_tripod(step, speed, command):
    step_I  = step
    step_II = step+2
    step_III= step+4
    step_IV = step+6
    if step_II > 8:
        step_II = step_II - 8
    if step_III> 8:
        step_III= step_III- 8
    if step_IV > 8:
        step_IV = step_IV - 8

    if command == 'forward':
        for i in range(1,(pixel+1)):
            leg_tripod('I', step_I, i, speed)
            leg_tripod('II', step_II, i, speed)

            leg_tripod('III', step_III, i, speed)
            leg_tripod('IV', step_IV, i, speed)

            await asyncio.sleep(3)

    elif command == 'backward':
        for i in range(1,(pixel+1)):
            leg_tripod('I', step_I, i, -speed)
            leg_tripod('II', step_II, i, -speed)

            leg_tripod('III', step_III, i, -speed)
            leg_tripod('IV', step_IV, i, -speed)

            await asyncio.sleep(3)

    elif command == 'left':
        for i in range(1,(pixel+1)):
            leg_tripod('I', step_I, i, -int(speed*turn_steady))
            leg_tripod('II', step_II, i, -int(speed*turn_steady))

            leg_tripod('III', step_III, i, speed)
            leg_tripod('IV', step_IV, i, speed)

    elif command == 'right':
        for i in range(1,(pixel+1)):
            leg_tripod('I', step_I, i, speed)
            leg_tripod('II', step_II, i, speed)

            leg_tripod('III', step_III, i, -int(speed*turn_steady))
            leg_tripod('IV', step_IV, i, -int(speed*turn_steady))
```

Fuente. Programa de los movimientos realizados por el robot Adeept_DarkPaw. Adaptación propia en base a la investigación realizada (2020)

Así pues, podemos observar el programa realizado por el fabricante para determinar los movimientos a realizar por el robot de Adeept, de esta manera encontramos en la línea 959 del programa *move.py* que nos provee el fabricante la definición y desarrollo del programa de movimientos, teniendo los cuatro principales movimientos bien definidos, como son *forward, backward, left* y *right*. Con estos cuatro movimientos podemos realizar las funciones necesarias y requeridas por nuestro robot.

A continuación, procederemos a realizar un diagrama de flujo a partir del programa provisto por el fabricante, con la finalidad de entender el comportamiento de esta función en específico. Para lo cual seguiremos realizando la retroingeniería, con la finalidad de entender a mayor profundidad el funcionamiento de la función *dove_move_tripod*.

Figura 45
Diagrama de flujo de los movimientos



Fuente. Diagrama de flujo de los movimientos realizados por el robot Adeept_DarkPaw. Adaptación propia en base a la investigación realizada (2020)

De acuerdo con el programa realizado por Adeept para el robot, podemos observar que primero define un el programa, en el cual desarrollará el nombramiento de variables y funciones para que el robot pueda proceder de manera apropiada.

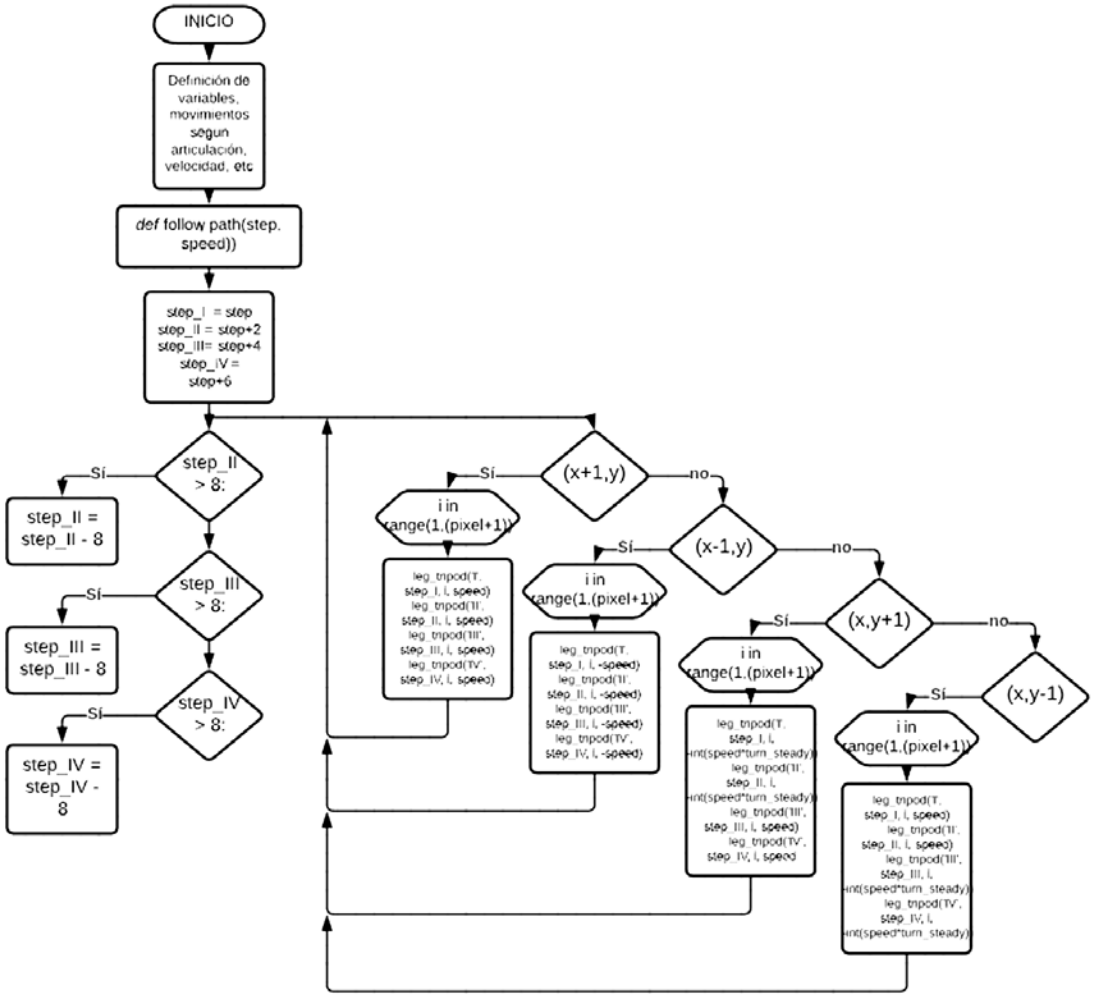
Para empezar el programa nos define la función y engloba sus variables, acto seguido nos define las variables *step* que se encuentran divididas y enumeradas de la uno a la cuatro, por la cantidad de extremidades con las que dispone nuestro robot; a cada variable se le asigna un valor *step* que se encuentra definido líneas anteriores a esta parte del programa.

A continuación, el fabricante pasa a tomar decisiones basadas en las especificaciones y necesidades de cara extremidad, para que el robot pueda optar por un camino a seguir en el desarrollo del programa. Para terminar el fabricante realiza un árbol de decisiones en caso el operador accione alguno de los comandos definidos (*forward, backward, left* o *right*), en cualquiera de los casos el programa procederá a desarrollar un bucle *for* en el rango de 1 a *pixel+1* (donde *pixel* tiene el valor determinado líneas anteriores como 4), una vez dentro del bucle *for* el programa procederá a realizar los movimientos de las cuatro extremidades, para lo cual en cada caso tiene una distintas variables, de las cuales la única que sufre cambios o alteraciones es la variable *speed*.

Después de haber analizado, utilizando la retroingeniería, el programa base que utilizaremos para poder desarrollar el nuestro, procederemos a realizar, al igual que en los casos anteriores, un diagrama de flujo que nos permita sentar las bases para desarrollar el programa para el seguimiento de camino, el cual será añadido también a los programas desarrollados anterior mente, y acto seguido, comprobando el correcto funcionamiento de todos los programas ensamblados, procederemos a añadir todos los programas dentro de nuestro programa de la interfaz gráfica.

Puesto que estamos tomando el programa provisto por el fabircante *Adeept*, debemos de incluirlo dentro de nuestro diagrama de flujo, para que luego en la programación a realizar sea tomado en cuenta y de esta manera no tener percances al momento de efectuar nuestras pruebas o incluso al momento de compilar nuestro programa.

Figura 46
Diagrama de flujo del seguimiento de camino



Fuente. Elaboración del diagrama de flujo para el seguidor de caminos. Adaptación propia en base a la investigación realizada (2020)

Si bien, hemos tomado como referencia el programa antes visto, podemos darnos cuenta que varios elementos tomados del programa *move.py* pueden ser adaptados a lo que necesitamos, la estructura necesaria se va a mantener, en cuanto a definición de variables, bucles, tomas de decisiones y otros; por otra parte podemos ver que en su desarrollo tomaremos algunas variaciones para que el programa del seguidor de camino pueda tomar las decisiones necesarias y más apropiadas según nuestros requerimientos.

A partir del diagrama de flujo, desarrollado en la figura 46, procederemos a realizar la elaboración del programa del seguidor de caminos, para lo cual usaremos el ya conocido programa SublimeText 3.

Figura 47

Programa para el seguimiento de camino

```
step_IV = step+6
if step_II > 8:
    step_II = step_II - 8
if step_III> 8:
    step_III= step_III- 8
if step_IV > 8:
    step_IV = step_IV - 8

if (x+1,y):
    for i in range(1,(pixel+1)):
        leg_tripod('I', step_I, i, speed)
        leg_tripod('II', step_II, i, speed)

        leg_tripod('III', step_III, i, speed)
        leg_tripod('IV', step_IV, i, speed)

        await asyncio.sleep(3)
elif (x-1,y):
    for i in range(1,(pixel+1)):
        leg_tripod('I', step_I, i, -speed)
        leg_tripod('II', step_II, i, -speed)

        leg_tripod('III', step_III, i, -speed)
        leg_tripod('IV', step_IV, i, -speed)

        await asyncio.sleep(3)
elif (x,y+1):
    for i in range(1,(pixel+1)):
        leg_tripod('I', step_I, i, -int(speed*turn_steady))
        leg_tripod('II', step_II, i, -int(speed*turn_steady))

        leg_tripod('III', step_III, i, speed)
        leg_tripod('IV', step_IV, i, speed)

        leg_tripod('I', step_I, i, speed)
        leg_tripod('II', step_II, i, speed)

        leg_tripod('III', step_III, i, speed)
        leg_tripod('IV', step_IV, i, speed)

        await asyncio.sleep(3)
elif (x,y-1):
    for i in range(1,(pixel+1)):
        leg_tripod('I', step_I, i, speed)
        leg_tripod('II', step_II, i, speed)

        leg_tripod('III', step_III, i, -int(speed*turn_steady))
        leg_tripod('IV', step_IV, i, -int(speed*turn_steady))

        leg_tripod('I', step_I, i, speed)
        leg_tripod('II', step_II, i, speed)

        leg_tripod('III', step_III, i, speed)
        leg_tripod('IV', step_IV, i, speed)

        await asyncio.sleep(3)
```

Fuente. Elaboración del programa para el seguidor de caminos en Python. Adaptación propia en base a la investigación realizada (2020)

Tal y como dijimos al principio de esta sección, hemos logrado desarrollar un programa que nos permita hacer el seguimiento correspondiente del camino que será desarrollado y provisto según lo determinado y especificado en las secciones anteriores de este mismo capítulo. Por tanto, podemos observar que el programa ha sido desarrollado en base a la investigación hecha, no solo por las necesidades y especificaciones dadas por nosotros, sino también por aquellas que hemos logrado obtener en la retroingeniería aplicada al programa provisto por el fabricante, el cual nos ha permitido desarrollar un programa que se adecue a las necesidades y requerimientos de la siguiente investigación.

Para simplificar podríamos decir que el resultado obtenido de cada programa por separado ha sido completamente exitoso, por esta razón continuaremos el trabajo en el siguiente capítulo, donde no solo procederemos a realizar las pruebas necesarias en cada programa desarrollado, sino que con cada programa ya bien estructurado y testeado, podremos hacer la fusión de los cuatro programas necesarios y con ello podremos hacer las pruebas solicitadas para demostrar el correcto funcionamiento de nuestro robot, y así alcanzar un resultado esperado en nuestra presente investigación.

CAPÍTULO V

5. RESULTADOS FINALES

5.1.Ensamble del Robot

En primer lugar, procederemos a realizar el ensamblado de nuestro robot, para contar con el sujeto de prueba, donde realizaremos los diferentes test de nuestros programas. Una vez tengamos nuestro robot debidamente ensamblado, procederemos a realizar las pruebas por separado, comprobando el correcto funcionamiento de cada uno de nuestros programas; y una vez obtenido los resultados deseados daremos paso al ensamble final de todos los programas en su conjunto, para realizar las pruebas correspondientes y con ello comprobar que nuestro programa funciona de manera adecuada.

Por lo que sigue vamos a realizar el armado de cada una de las piezas del robot, utilizando las herramientas necesarias y siguiendo las instrucciones brindadas por el fabricante para no tener ningún tipo de inconveniente.

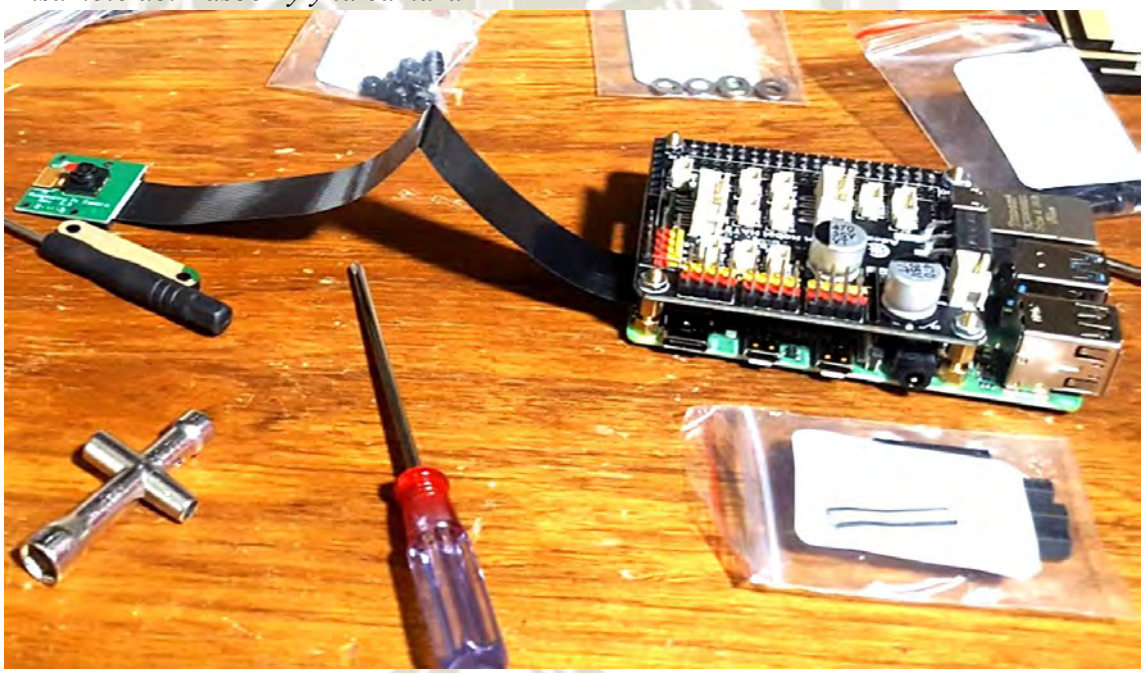
Figura 48
Elementos del robot

Fuente. Partes y piezas de nuestro robot, incluyendo la placa y las herramientas. Adaptación propia en base a la investigación realizada (2020)

En la figura 48 podemos observar los diferentes elementos que contiene nuestro robot, dentro de los cuales están la estructura de acrílico, los servos motores, los sensores, la cámara, la placa y las herramientas a utilizar. A continuación, procederemos a ensamblar nuestro robot de manera adecuada, para lo cual seguiremos las instrucciones brindadas por el fabricante para no tener complicaciones al momento de ensamblar el robot, y que de esta manera nuestro sujeto de prueba pueda operar de manera apropiada.

Figura 49

Ensamble del Rasberry y la cámara



Fuente. Ensamble de nuestro robot, empezamos con la incorporación de la placa al Raspberry Pi y a la cámara. Adaptación propia en base a la investigación realizada (2020)

Inicialmente hemos procedido a realizar el ensamble de la cámara con el *Raspbery Pi*, el cual luego se unirá a la placa que viene con nuestro robot para poder realizar, de manera adecuada, las funciones pertenecientes a los servos motores.

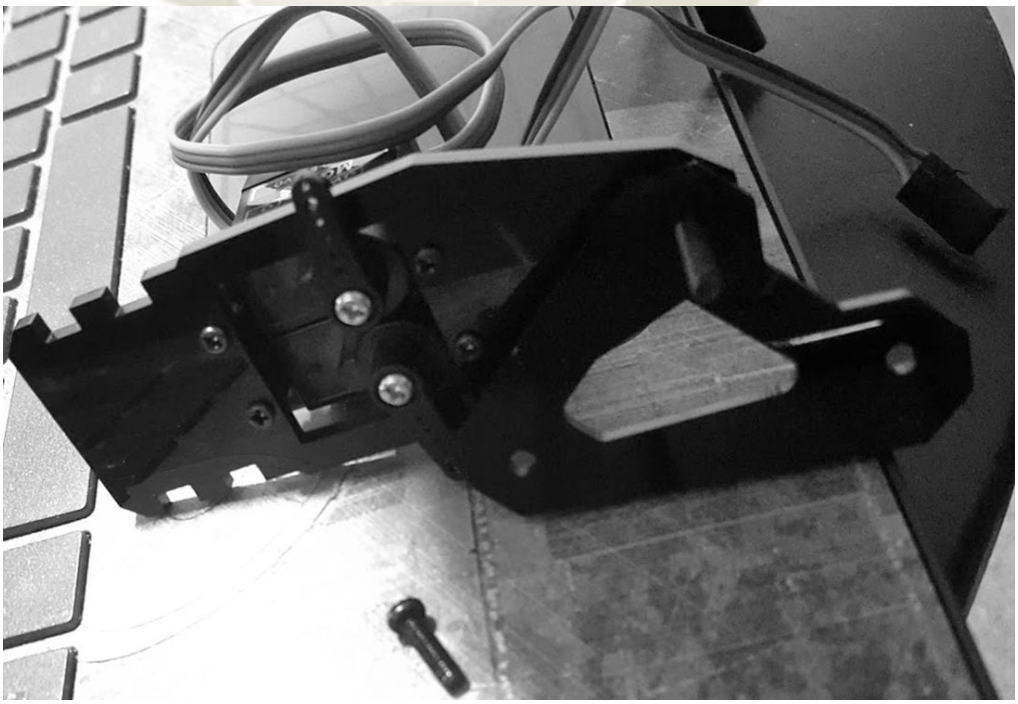
Figura 50

Partes para ensamblar las extremidades

*Fuente.*Piezas que corresponden a las extremidades de nuestro robot. Adaptación propia en base a la investigación realizada (2020)

Figura 51

Ensamble de una extremidad del robot



*Fuente.*Ensamble de las piezas de una de las extremidades con los servomotores. Adaptación propia en base a la investigación realizada (2020)

Figura 52
Extremidad del robot

Fuente. Extremidad del robot completamente ensamblada. Adaptación propia en base a la investigación realizada (2020)

A partir de tener ensamblada las cuatro extremidades del robot, procederemos a realizar el ensamblado del cuerpo y el resto de las piezas, cabe resaltar que las cuatro extremidades del robot son idénticas, por lo que el ensamblado es el mismo, pero no debemos de descuidarnos al momento de seguir las instrucciones, ya que, a pesar de que las extremidades son iguales, cada una tiene una orientación diferente y una posición particular, con esto queremos decir que si bien es cierto, cada extremidad cuenta con los mismos componentes, no podemos decir que la extremidad I es igual a la II, o que la extremidad II es igual a la III, y así sucesivamente.

En cuanto logremos ensamblar las cuatro extremidades, procederemos a separarlas en el orden indicado por las instrucciones del fabricante, para luego no tener ningún tipo de problema al momento de ensamblarlo con el cuerpo del robot.

Figura 53
Robot ensamblado

Fuente. Robot completamente ensamblado. Adaptación propia en base a la investigación realizada (2020)

Finalmente, luego de ensamblar el cuerpo con las demás piezas del robot, hemos obtenido el resultado que se muestra en la figura 53, que es el resultado mostrado en las figuras dadas por el fabricante en sus distintas plataformas. Una vez ensamblado procederemos al siguiente paso que vendría a ser programar el Raspberry Pi para su adecuado funcionamiento con el programa dado por Adeept.

5.2.Programación del Raspberry Pi

Por otra parte, en esta sección mostraremos brevemente como se realiza la instalación del programa de *Raspberry*, y de qué manera se debe poner en funcionamiento dentro de nuestro robot, para que de esa forma tenga la conectividad y la autonomía necesaria al momento de correr su programa o algún otro.

Figura 54
Descarga del sistema operativo

Fuente. Descarga del sistema operativo de Raspberry. Adaptado por "Lesson 3 Installing and Configuring Raspberry Pi System" de Adeept (2020)

A partir de la descarga anterior obtendremos un archivo que está en formato *.img*, por lo tanto, descargaremos el programa *Raspberry Pi Imager*, con el cual podremos grabar el sistema operativo en formato *.img* en una memoria SD que luego será introducida en nuestra placa *Raspberry*.

Para ilustrar mejor, nosotros no podemos realizar la instalación del sistema operativo que nos brinda la página oficial del fabricante de las placas *Raspberry Pi*, porque si no utilizamos el programa *Raspberry Pi Imager*, simplemente jalaremos un archivo que no ha sido leído ni grabado en una memoria de la manera apropiada y el sistema operativo no correrá de forma adecuada. Por otra parte, el programa *Raspberry Pi Imager*, cuenta con una función que nos permitirá escribir el programa dentro de la unidad de almacenamiento, que en nuestro caso es una memoria SD, de tal manera que con solo insertar dicha unidad la placa estará totalmente operativa y en funcionamiento.

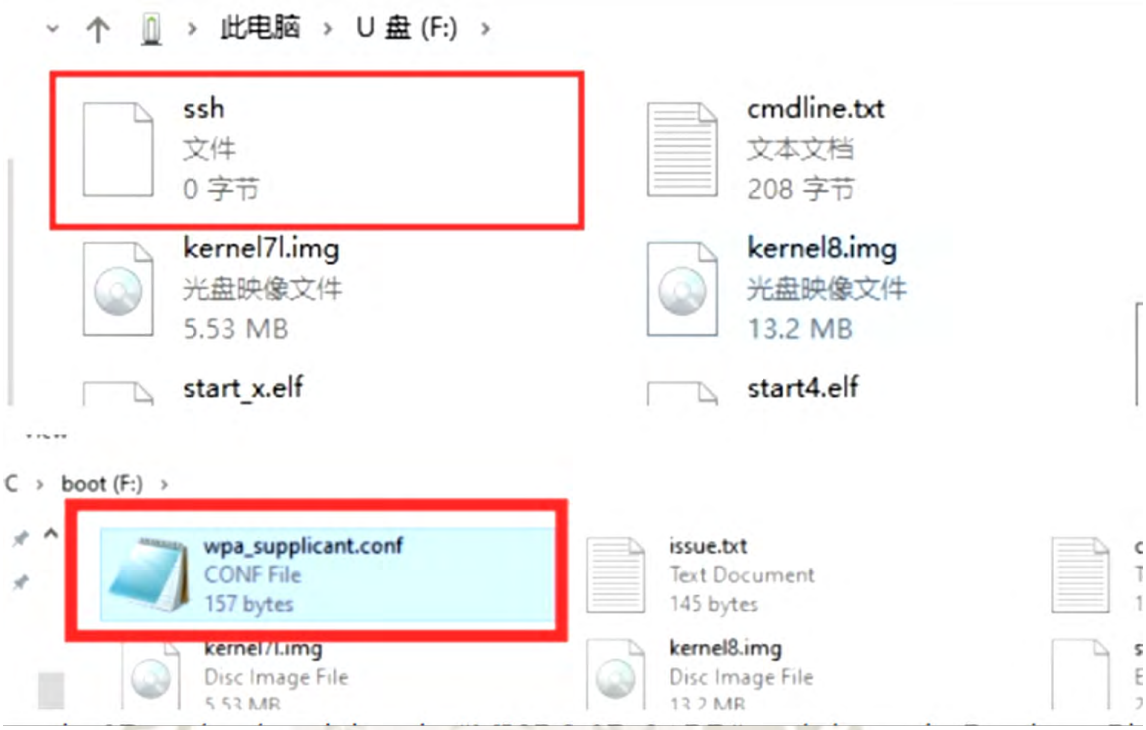
Figura 55
Grabado del sistema operativo

Fuente. Grabado del sistema operativo en la unidad de almacenamiento, a través de Rapberry Pi Imager. Adaptado por "Lesson 3 Installing and Configuring Raspberry Pi System" de Adeept (2020)

Acto seguido procederemos a crear dos archivos que se necesitan para poder realizar el control de *Raspberry Pi*, de manera autónoma; el primer archivo es uno denominado “ssh.txt”, y el otro es el archivo de nombre “wpa_supplicant.conf”. Cada uno de estos archivos tiene una función diferente, para ilustrar mejor, en el caso del primero es un protocolo que nos brindará seguridad al momento de iniciar sesión de manera remota; también con el “ssh.txt” nos permite usar de forma remota la línea de comando de *Raspberry Pi* en otra máquina. Y en el caso del según archivo, este nos ayudará a realizar la configuración de la red a la cual nos queremos conectar, con esto se podrá trabajar a la distancia, con el solo uso de la red de nuestro hogar o a la que queramos conectarnos.

Asimismo, con estos archivos vamos a poder realizar nuestra programación y conexión con el robot de manera remota, sin la necesidad de tener que estar conectados al robot al momento de generar algún cambio en su programación.

Figura 56
Archivos de conexión

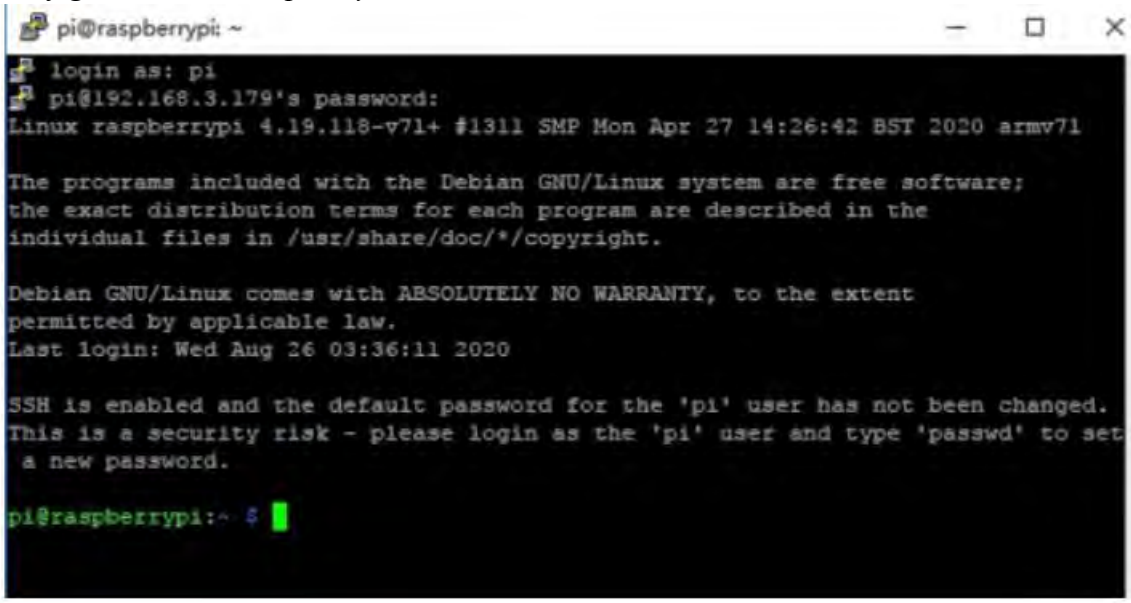


Fuente. Archivos necesarios para generar la conexión con nuestro robot y la placa de manera autónoma. Adaptado por "Lesson 3 Installing and Configuring Raspberry Pi System" de Adeept (2020)

A partir de tener el sistema operativo y los archivos cargados en nuestra unidad de almacenamiento procederemos a introducirlo en la placa de *Raspberry Pi*, e inmediatamente procederemos a realizar la búsqueda de la dirección IP que se le ha asignado a nuestra placa, mediante un programa o aplicación que nos permita descubrir su dirección IP.

En otro orden de cosas y una vez encontrada la dirección IP, procederemos a realizar la configuración de nuestra placa, entrando a una interfaz provista por el programa PuTTY, con la cual nos permitirá tener acceso a la placa y hacer las configuraciones preliminares, luego pasaremos a instalar los programas y librerías necesarías, y por último podremos cargar los programas que nos brinda el fabricante para nuestro robot.

Figura 57
Configuración del Raspberry Pi



Fuente. Configurando el Raspberry Pi con la ayuda del programa PuTTY. Adaptado por "Lesson 3 Installing and Configuring Raspberry Pi System" de Adeept (2020)

En resumidas cuentas esta es la manera en que se programa, instala y configura el sistema operativo que nos brinda *Raspberry Pi*, es importante no saltarse ningún paso, ya que una mala configuración del sistema operativo haría que nuestros programas brindados, ya sea por el fabricante o por nosotros, no tenga un correcto funcionamiento, es también preciso decir que la configuración de la red cambiará cada vez que nosotros cambiemos de ubicación y la red no sea la misa en ese preciso momento, para lo cual se debe de realizar una variación en el archivo "wpa_supplicant.conf", de no ser así el robot no establecerá ningún tipo de conexión con nuestra máquina.

5.3.Resultados de la Interfaz Gráfica

Después de haber realizado el ensamble del robot y la configuración adecuada para la placa *Raspberry Pi*, procederemos a comprobar el correcto funcionamiento del interfaz gráfico de nuestro programa y con ello comprobar la conexión adecuada con nuestro robot, en este punto procederemos únicamente a comprobar estas acciones, ya en las secciones posteriores nos dedicaremos a comprobar el correcto funcionamiento de cada

programa de manera independiente, para finalizar con el test del correcto funcionamiento de todo el programa en su conjunto.

Figura 58
Rastreo de la dirección IP del robot

Fuente. Buscando la dirección IP correspondiente a nuestro robot, para poder estabñecer una conexión. Adaptación propia en base a la investigación realizada (2020)

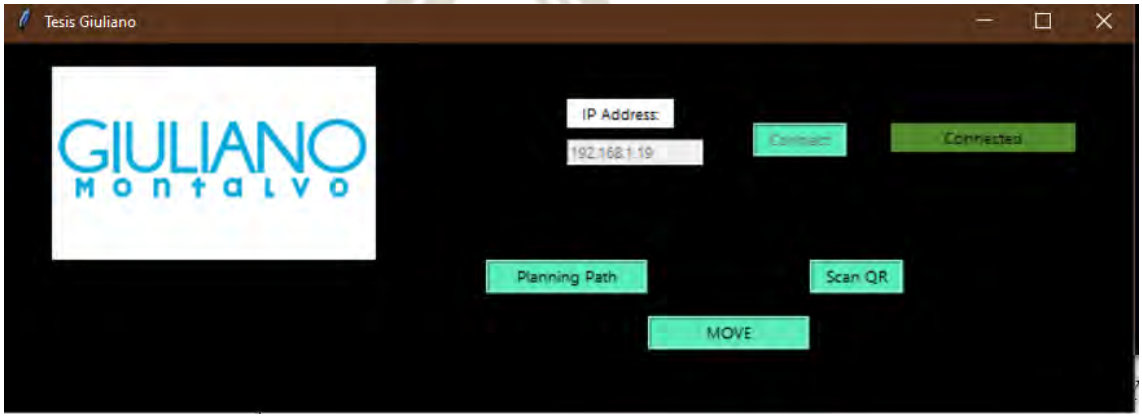
Partiendo de lo dicho en la anterior sección procederemos a realizar un análisis, con un programa, para encontrar la dirección IP que le ha sido asignada a nuestro robot, con el fin de poder establecer una adecuada conexión; y de esta manera podamos hacer los test correspondientes a nuestro robot.

Figura 59
Interfaz gráfico - final

Fuente. Realizamos la conexión a través de nuestro interfaz gráfico. Adaptación propia en base a la investigación realizada (2020)

Acto seguido hemos procedido a la apertura de nuestro interfaz gráfico, donde introduciremos la dirección IP, con la cual deberíamos poder establecer la conexión.

Figura 60
Conexión con el robot



Fuente. Verificamos que la conexión ha sido establecida de manera adecuada con nuestro robot. Adaptación propia en base a la investigación realizada (2020)

De ahí que podemos concluir que el programa de la interfaz gráfica funciona de manera apropiada, dentro de esta interfaz hemos podido observar que los elementos programados han dado un resultado positivo, siendo así que la conexión establecida entre el robot y la computadora ha sido exitosa.

Figura 61
Conexión establecida con el robot

Fuente. Luego de introducir la IP, si la conexión ha sido exitosa, el robot cambia de posición y las luces se encienden de color verde. Adaptación propia en base a la investigación realizada (2020)

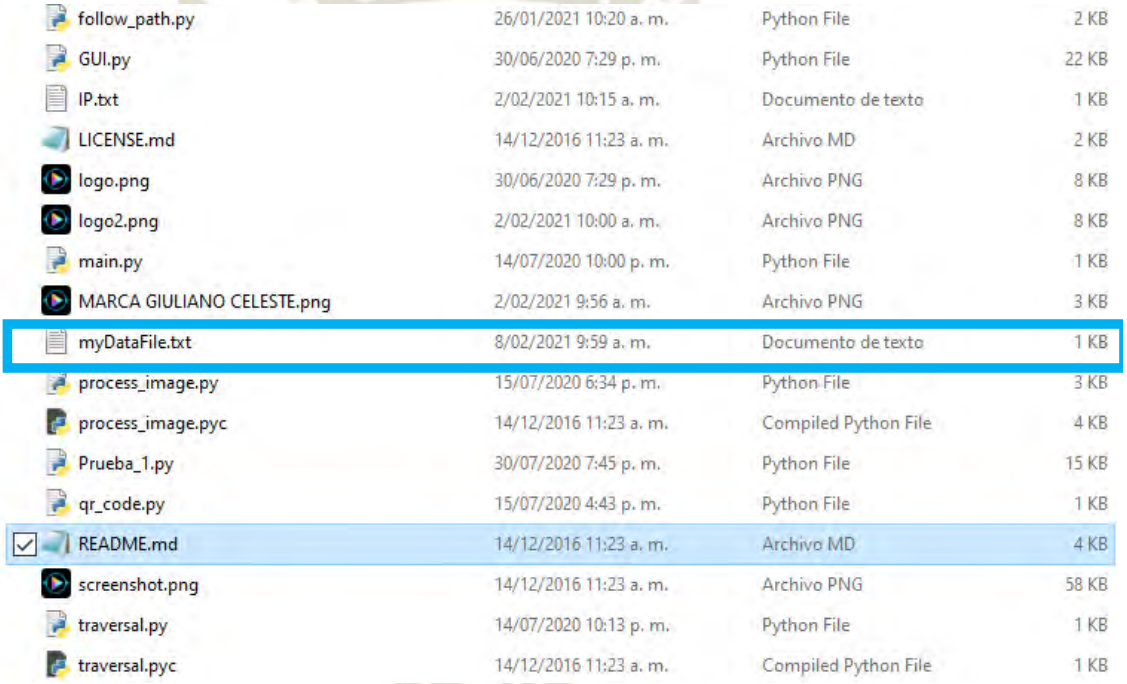
Finalmente llegamos a la conclusión que el funcionamiento de nuestra interfaz gráfica es la deseada y la más adecuada, podemos observar que se al momento de introducir un IP correcta el programa establece la conexión con el robot y esto se refleja en el mismo con el cambio de posición adoptada y la variación en las cintas led de color. Con esto podemos seguir con los siguientes test para ir acoplando uno por uno los programas obtenidos por nuestra presente investigación. Una vez que se termine de hacer los test de manera individual, se procederá a realizar la unión de todos los programas y su test final.

5.4.Resultados del código QR

A partir del resultado obtenido en la sección anterior, procederemos a realizar el test correspondiente para nuestro programa de código QR, para ello procederemos a ejecutar la programación realizada en el capítulo anterior, dándole ya valores específicos y mostrándole una serie de códigos con valores válidos y otros que no corresponden a la información solicitada.

Figura 62

Almacenamiento de datos



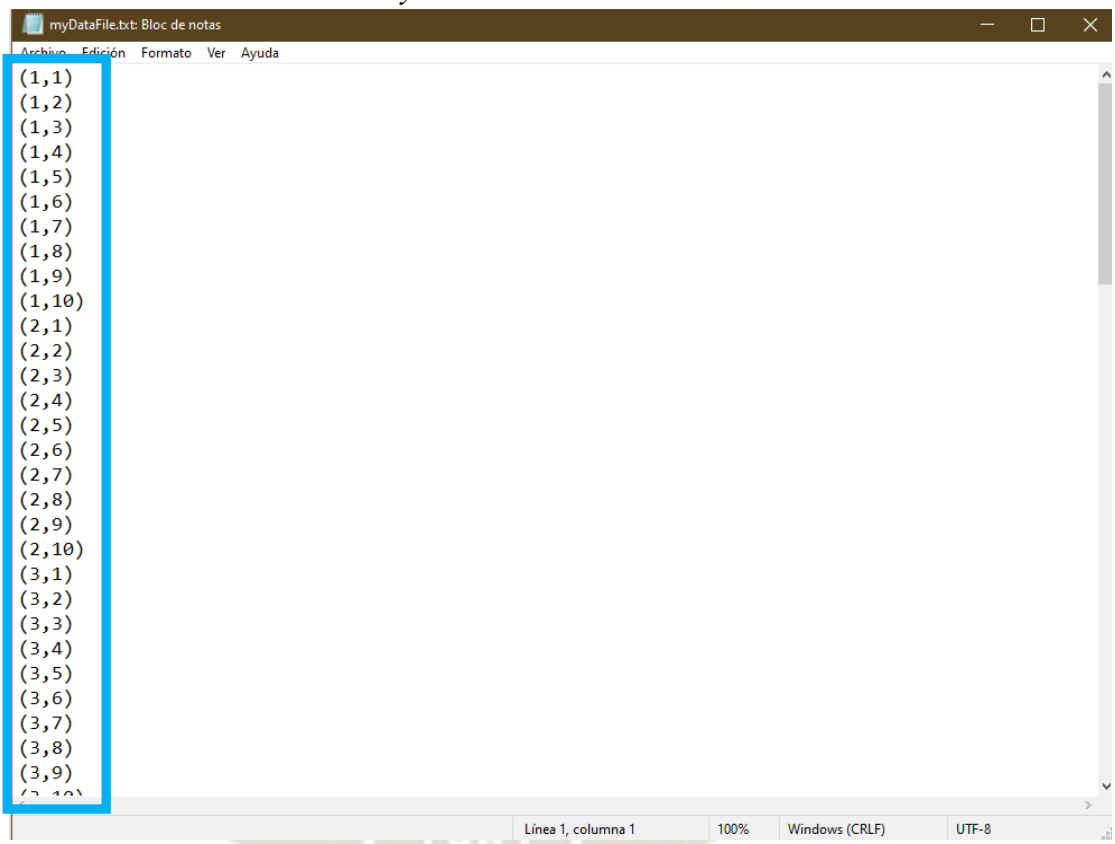
Fuente. Crearemos un archivo .txt, donde podremos almacenar la información que debe considerarse como aceptable para el código QR. Adaptación propia en base a la investigación realizada (2020)

Por cierto, es necesario crear un documento que el programa pueda leer y reconocer como información válida, esto se realiza con el propósito de no hacer más largo el programa y con la finalidad de obtener una manera rápida, y que pueda ser entendida, de almacenar la información que va a distinguir el programa al momento de realizar la lectura del código QR.

En consecuencia, podemos observar, en la figura 62, que hemos creado un archivo .txt, llamado *myDataFile*, que será el encargado de guardar esta información necesaria para nuestro código QR.

Figura 63

Coordenadas almacenadas en myDataFile.txt

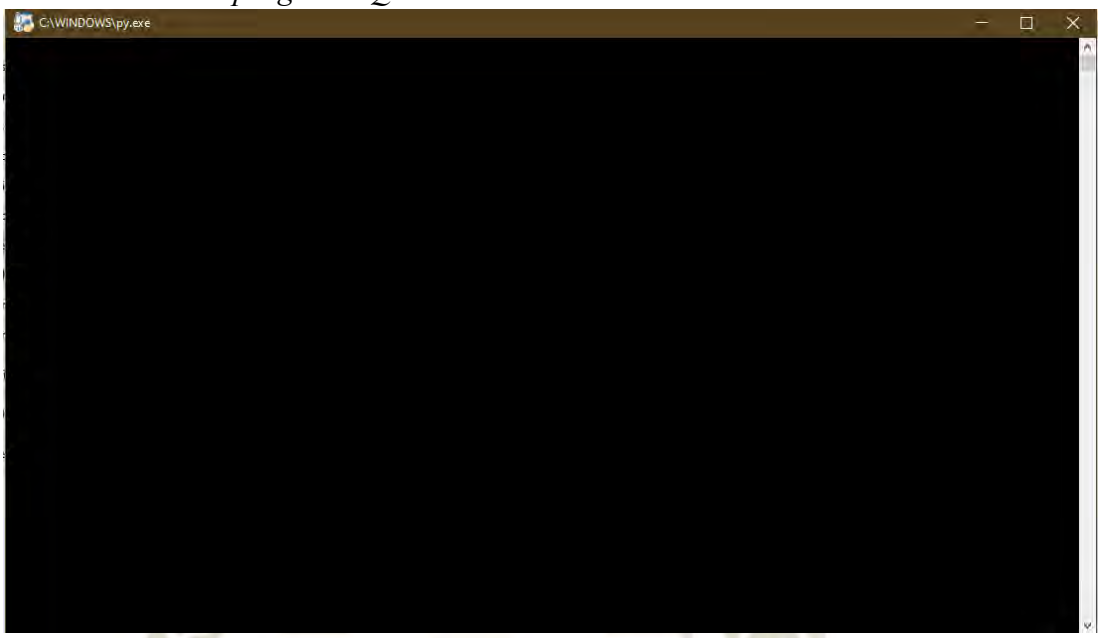


Fuente. El contenido que se almacena en el archivo myDataFile.txt son todas las coordenadas dentro de un mapa dividido por grillas, que podrían ser posibles puntos de partida para nuestro robot. Adaptación propia en base a la investigación realizada (2020)

Después de haber creado el archivo *myDataFile.txt*, procederemos a rellenar la información necesaria, así como se puede observar en la figura 63; los datos almacenados dentro de este archivo serán coordenadas que el robot pueda reconocer a través de su cámara y luego identificados como un punto de partida o no para iniciar un proceso.

Figura 64

Inicialización del programa QR



Fuente. Primero hacemos correr el programa del código QR, para comprobar que sigue funcionando de manera correcta. Adaptación propia en base a la investigación realizada (2020)

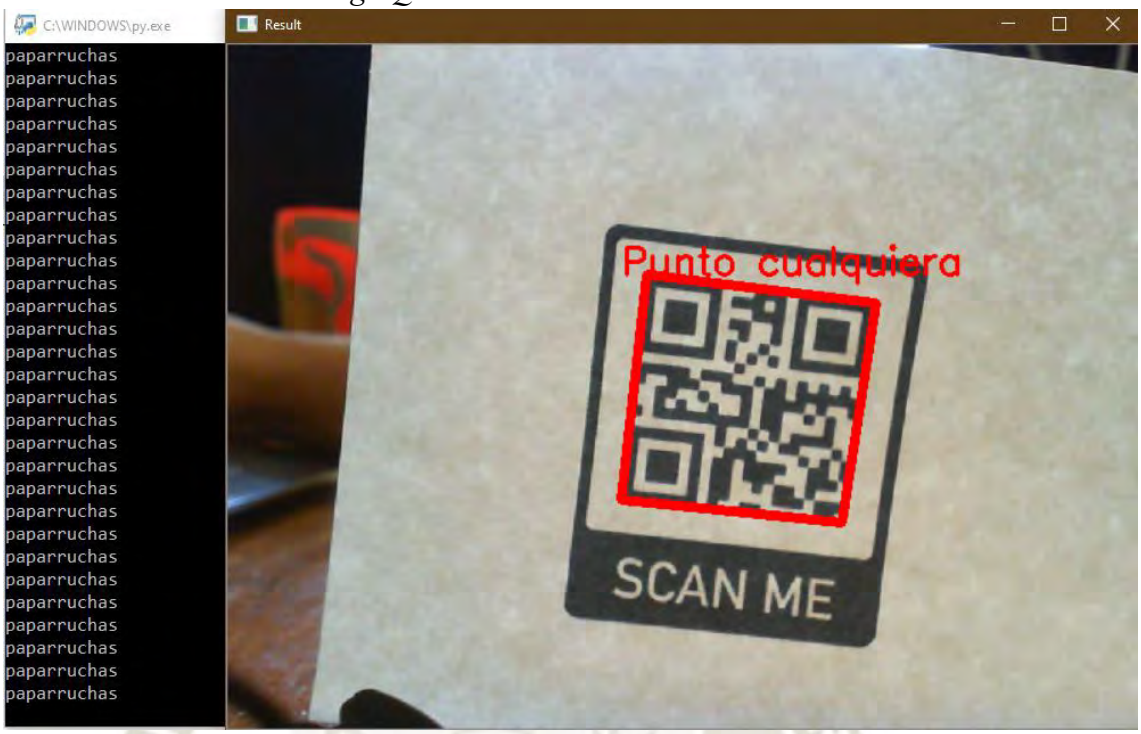
Figura 65

Reconocimiento de un código QR

Fuente. Realizamos la primera prueba con un código QR que contiene una información almacenada dentro de nuestro archivo .txt. Adaptación propia en base a la investigación realizada (2020)

Figura 66

Reconocimiento de un código QR inválido



Fuente. Realizamos una segunda prueba con un código QR que no contiene una información almacenada dentro de nuestro archivo .txt. Adaptación propia en base a la investigación realizada (2020)

En síntesis, podemos ver que los resultados mostrados en los test realizados a nuestro programa del código QR, son los apropiados. En la figura 65 podemos observar que uno de los datos almacenados en nuestro archivo *myDataFile.txt* es reconocido rápidamente por nuestra cámara en funcionamiento de nuestro programa, caso contrario sucede con el test realizado y demostrado en la figura 66, donde se puede observar que la información almacenada dentro de nuestro código QR fue la palabra "paparruchas", la cual no corresponde a los datos almacenados dentro de nuestro archivo .txt, de esa forma cuando ponemos este código al frente de nuestra cámara, esta lo identifica como un punto cualquiera. Cabe concluir que, dado los resultados obtenidos, este programa esta optimo y operativo para ser unificado con el resto del programa que se encuentra dentro de nuestra interfaz gráfica.

Por consiguiente, en la siguiente sección procederemos a realizar el test y mostrar los resultados que se obtengan al inicializar el programa del planificador de trayectorias.

5.5.Resultados del planificador de trayectoria y del algoritmo a*

Como se indicó, procederemos a realizar la simulación y el test correspondiente para el programa elaborado de nuestro planificador de trayectorias, para este paso primero realizaremos la simulación del programa de manera simple y solamente probando el planificador de trayectorias, acto seguido trataremos de fusionar este programa con el lector de código QR, para comprobar su correcto funcionamiento en conjunto, y así poder ir avanzando con nuestro resultado final, de manera que al juntar todos los programas tengamos la menor cantidad de problemas posibles.

Figura 67

Definimos un punto de partida – primera prueba

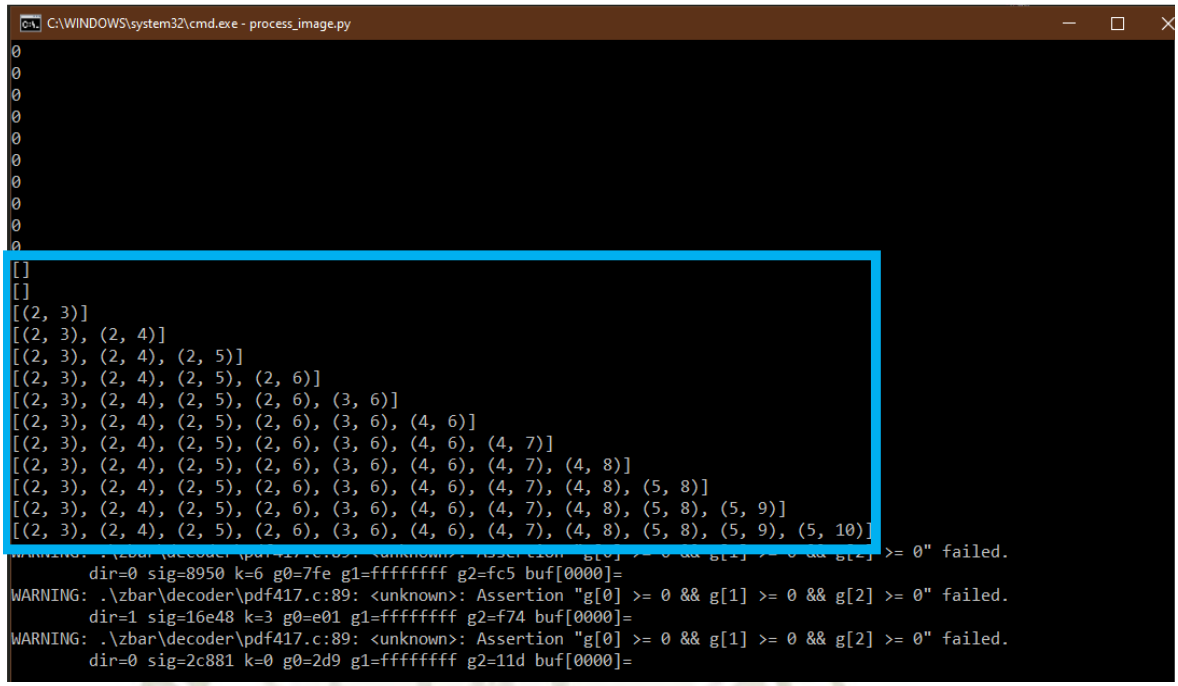
```
index = [1,1] #Punto de inicio
blank_image = np.zeros((60,60,3), np.uint8)
#crea una matriz de 100 imágenes en blanco
list_images = [[blank_image for i in xrange(10)] for i in xrange(10)]   #matriz de lista de imágenes
#vacío #matriz para representar las cuadrículas de imágenes recortadas individuales
maze = [[0 for i in xrange(10)] for i in xrange(10)]

#transversal para cada cuadrado
for (x, y, window) in traversal.sliding_window(image, stepSize=60, windowSize=(winW, winH)):
    # si la ventana no cumple con el tamaño de ventana deseado, ignórela
    if window.shape[0] != winH or window.shape[1] != winW:
        continue

#   índice de impresión, la imagen es nuestro iterador, es donde estábamos en la matriz de imagen de devoluciones
    clone = image.copy()
    #formato cuadrado para openCV
    cv2.rectangle(clone, (x, y), (x + winW, y + winH), (0, 255, 0), 2)
    crop_img = image[x:x + winW, y:y + winH]                #recortar la imagen
    list_images[index[0]-1][index[1]-1] = crop_img.copy()          #Agrégalo a la matriz de imágenes
```

Fuente. Para esta primera prueba, procederemos a utilizar y definir un punto de partida para nuestro planificador de trayectorias. Adaptación propia en base a la investigación realizada (2020)

Para esta primera prueba hemos definido en el programa un punto de partida el cual nos ayudará a definir de manera manual el lugar de donde partiría nuestro robot, con esto comprobaremos el correcto funcionamiento de nuestro programa realizado para nuestro planificador de trayectorias. Acto seguido, comprobando que los resultados son los deseados, comprobaremos el funcionamiento del planificador de trayectorias acompañado de nuestro identificador de códigos QR.

Figura 68

Primer resultado del planificador de trayectorias



Fuente. Este primer resultado muestra el camino a seguir por nuestro programa, según los obstáculos indicados y por el punto de inicio ingresado. Adaptación propia en base a la investigación realizada (2020)

De esta manera hemos obtenido el siguiente resultado que se grafica en la figura 68, donde podemos observar el resultado obtenido por nuestro programa para encontrar la ruta del camino dado a partir del punto de inicio, ya sea el digitado por el operador dentro del mismo programa o el obtenido a través del código QR, con el cual el robot podrá generar la lista de movimientos a realizar para que el robot pueda desempeñar la ruta trazada por el planificador de trayectoria.

Cabe concluir que en esta sección hemos podido demostrar el funcionamiento del planificador de trayectorias que hemos desarrollado, evidenciando que es posible obtener una ruta a seguir a partir de un punto de partida, un destino al cual llegar y el algoritmo apropiado. Por otro lado, ha este punto hemos podido lograr el objetivo de obtener, y a la misma vez validar, el modelo algorítmico para nuestro sistema de control, determinando que el algoritmo a* es el más apropiado para nuestro proyecto debido a la capacidad que nos brinda al poder calcular el camino mínimo a seguir mediante los métodos heurísticos, el cual ha permitido que a través de nuestro programa logremos realizar el planificador

de trayectorias con éxito, llegando así a la conclusión de que el análisis realizado en el capítulo tres ha permitido que a este punto de nuestro proyecto podamos validar el diseño realizado en el capítulo cuatro obteniendo los resultados favorables.

De esta manera procederemos a continuar con los test de los siguientes programas para ir ensamblando el programa final que nos permitirá determinar el correcto funcionamiento de nuestro robot a partir de los resultados obtenidos durante nuestra investigación.

5.6.Resultados del seguidor de caminos

Como se indicó, al finalizar la sección anterior, procederemos a realizar el test y obtención de resultados específicos para la evaluación que realizaremos a nuestro programa del seguidor de caminos, para lo cual necesitaremos utilizar, no solo, el programa realizado, sino el sujeto de prueba y el programa del interfaz gráfico, para determinar el correcto funcionamiento y que las acciones a realizarse sean las requeridas por el operador.

Puesto que la prueba necesita un lugar donde validar nuestro diseño, procederemos a implementar un campo con obstáculos que permitan comprobar la efectividad de nuestros programas, demostrando así que nuestro diseño le garantiza al robot la capacidad de poder reconocer un punto fijo a través de la lectura de un código QR, para luego identificar el mapa, a partir de ello poder utilizar su algoritmo y planificar la trayectoria con menor coste para nuestro robot, tal y como dijimos al principio tendrá que evadir algunos obstáculos que pondremos en el campo de desplazamiento y por último poder ejecutar los movimientos necesarios para poder desplazarse en nuestro campo.

Figura 69

Posición inicial del Robot

Fuente. En este primer proceso podemos observar la posición inicial del robot y los obstaculos definidos por nosotros mismos, a través de diferentes objetos. Adaptación propia en base a la investigación realizada (2020)

En primer lugar, procederemos a posicionar al robot en el punto de partida del cual iniciará y podrá realizar sus acciones, como se visualiza en la figura 69, también hemos procedido a determinar ciertos obstáculos los cuales serán ingresados e identificados por el programa del planificador de trayectorias permitiendo que este determine y grafique la ruta que debe de seguir nuestro robot antes de poder realizar las acciones necesarias.

En consecuencia, deberemos de comprobar que el programa es capaz de evadir los obstáculos propuestos en el campo, dando como resultado una ruta en donde no se encuentren las coordenadas de los obstáculos. Sin embrago, si el programa codifica o utiliza una coordenada ocupada por un obstáculo como parte del camino a recorrer, nos encontraremos en la necesidad de revisar las líneas de programación para determinar y corregir nuestro error.

Figura 70

Definición de los obstáculos de manera virtual

```
maze = [[0 for i in range(10)] for i in range(10)]
obstacles = [(1,3),(1,5),(1,8),(1,9),(2,2),(2,8),(2,9),(3,3),(3,4),(3,8),(3,9),(4,3),(4,4),(5,3),(5,4)]

for i in obstacles:
    maze[i[0]][i[1]]=1
    print("obstaculos")
for i in range (10):
    for j in range (10):
        print(maze[i][j])
```

Fuente. Realizaremos la definición de los obstáculos de manera virtual a través del programa del planificador de trayectorias. Adaptación propia en base a la investigación realizada (2021)

A continuación, hemos procedido a definir los obstáculos dentro del programa del planificador de trayectorias, como se muestra en la figura 70, con el fin de poder obtener el resultado más apropiado al momento de elaborar la ruta que debe de seguir nuestro robot. De esta manera al momento de ejecutar la interfaz gráfica y realizar la conexión con el robot podremos ejecutar los programas y obtener los resultados necesarios desde el hecho de leer el código QR y obtener la coordenada de inicialización y al final poder elaborar la trayectoria a seguir dentro del mapa elaborado e ingresado dentro de nuestro programa principal.

De ahí que procederemos a ingresar a nuestro programa principal para poder inicializar el lector de códigos QR para establecer una coordenada fija en la que se encontrará nuestro robot, de esta manera obtendremos un punto de partida dentro de nuestro campo y a partir de ello se dará inicio a nuestro planificador de trayectorias, el cual teniendo un punto inicial, un punto de llegada y los obstáculos definidos; procederá a encontrar la trayectoria a seguir evadiendo los obstáculos y llegando al punto de partid.

Finalmente, una vez elaborado este sistema y comprobando su adecuado funcionamiento, podremos comprobar si las acciones a realizar por parte de nuestro robot serán las más optimas, por lo menos en cuanto a la programación. Por lo tanto, esta parte de las pruebas nos permitirá empezar a visualizar con optimismo los resultados finales que vamos a poder obtener en el siguiente punto.

Figura 71

Inicialización del programa del robot

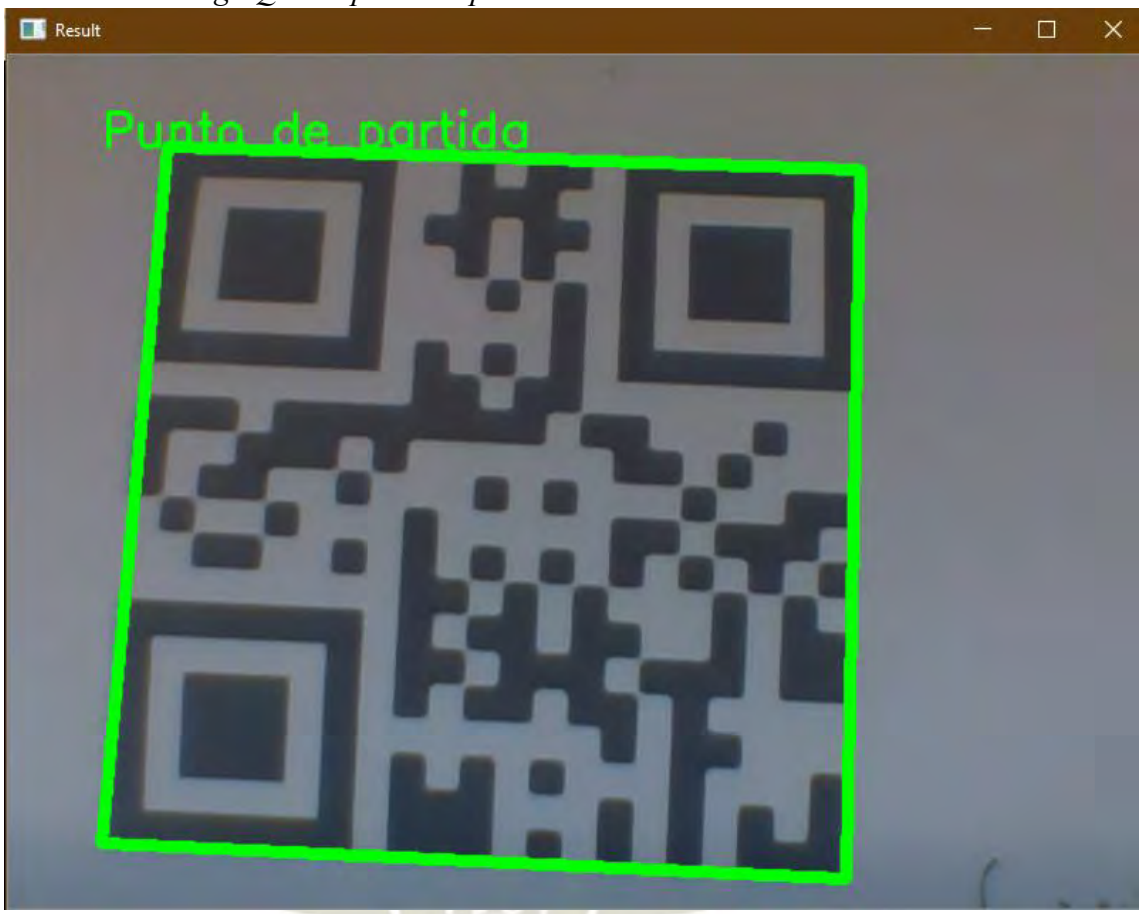
Fuente. Realizaremos el corrido del programa para poder determinar las funciones a realizar de nuestro robot, partiendo por el reconocimiento del mismo y la conexión por wifi. Adaptación propia en base a la investigación realizada (2021)

Observemos cómo se ha realizado la conexión correspondiente con el robot a través de la interfaz gráfica mediante el uso de nuestra red inalámbrica, como se grafica en la figura 71, una vez que la conexión ha sido de manera exitosa podemos empezar a realizar los siguientes pasos para llegar a la realización de los movimientos de nuestro robot para comprobar que el funcionamiento del mismo es el adecuado.

Por lo tanto, procederemos a realizar en el escaneo del código QR a través de la cámara de nuestro robot con el fin de obtener el posicionamiento exacto de nuestro robot, acto seguido realizaremos la planificación de nuestro camino a partir del punto de partida y

por último le indicaremos a nuestro robot que pueda realizar los movimientos correspondientes.

Figura 72
Lectura del código QR del punto de partida

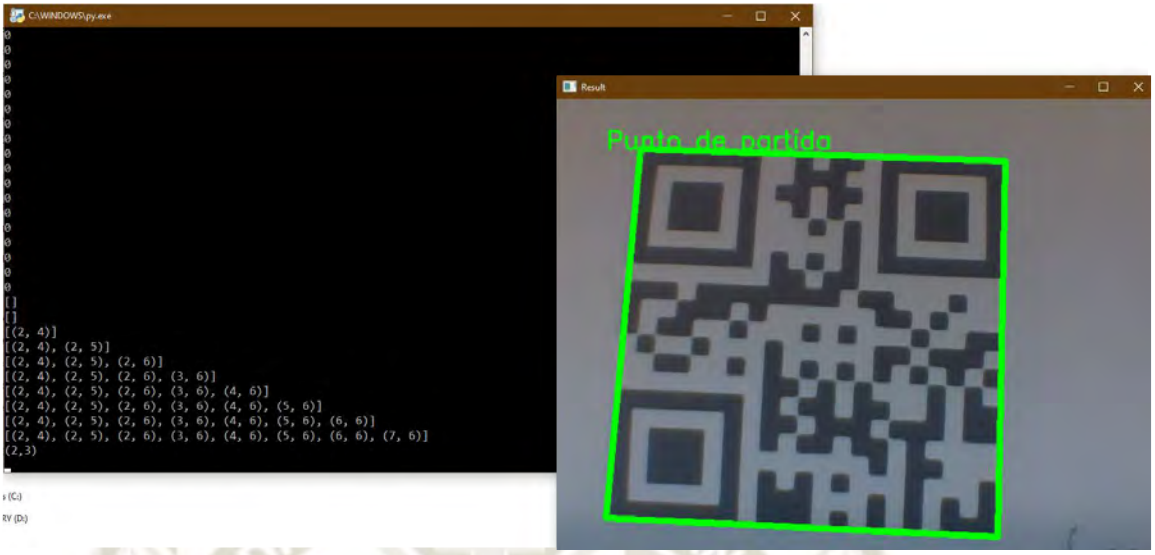


Fuente. Proporcionaremos un código QR a nuestro robot para que identifique, a través de su cámara, el punto de partida en el que se encuentra ubicado. Adaptación propia en base a la investigación realizada (2021)

De esta manera tenemos la lectura de nuestro código QR a través de la cámara de nuestro robot, en el código, como se puede observar en la figura 72, tenemos almacenado el punto de partida de nuestro robot en el cual se encuentra ubicado, que para dicha prueba será la posición (2,3) en nuestro mapa. Si en caso contrario la información almacenada dentro de nuestro código QR no hubiera sido la apropiada el mensaje hubiera sido negativo y la información obtenida no sería la pertinente para poder trabajar con los siguientes pasos de nuestra interfaz gráfica.

Figura 73

Planificación de la trayectoria a partir de la lectura del código QR



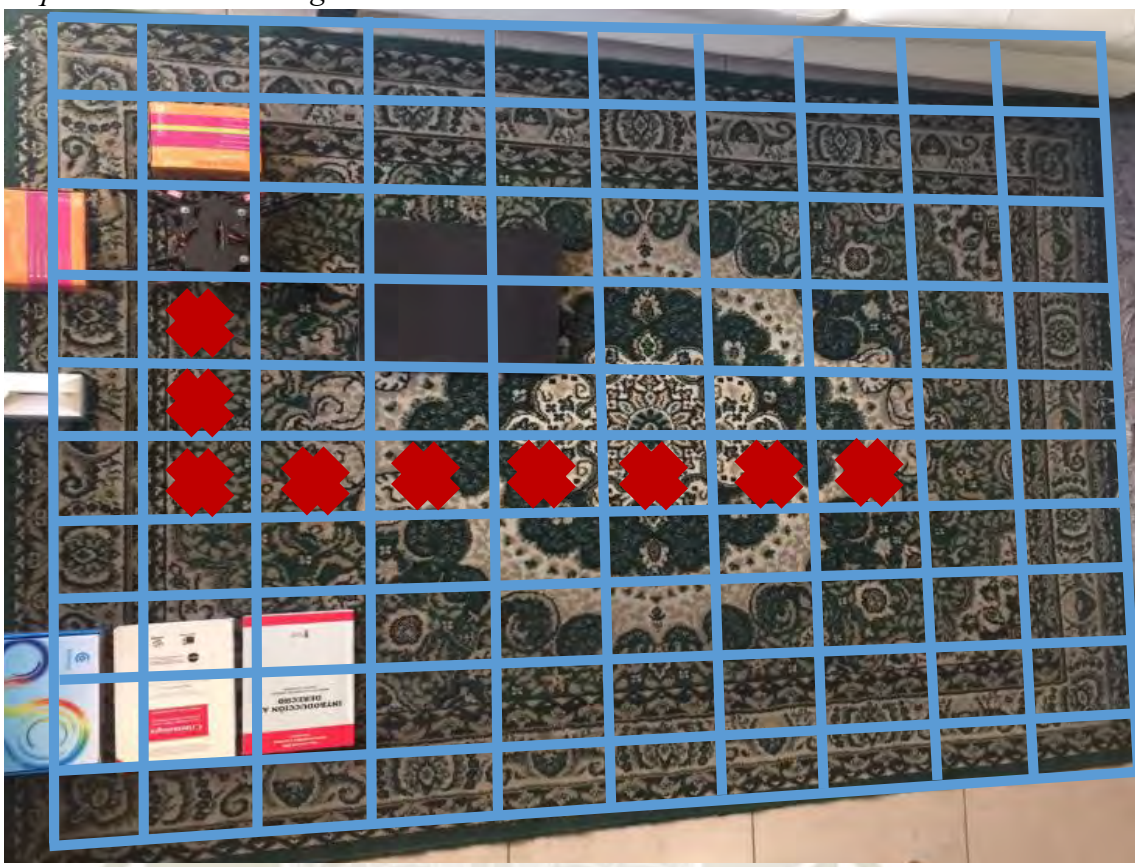
Fuente. A partir de la lectura del código QR y el punto de partida proporcionado, se logra obtener y elaborar una ruta a seguir con nuestro programa del planificador de trayectorias. Adaptación propia en base a la investigación realizada (2021)

Después de haber realizado la lectura del código QR, procederemos a realizar a ejecutar el planificador de trayectorias con el cual obtendremos una ruta apropiada, como se muestra en la figura 73, para nuestro robot, tomando en consideración las disposiciones dadas como son la posición inicial, la posición final del trayecto y los obstáculos.

Por otro lado, una vez definida la trayectoria que nuestro robot debe de seguir debemos proceder a realizar las acciones y movimientos que nos permitan cumplir con esa trayectoria trazada, de esta manera nuestro robot podrá cumplir con los objetivos planteados a lo largo de todo nuestro proyecto de investigación. Por esta razón procederemos a realizar la ejecución de la última parte de nuestro programa, el cual vendría a ser el seguidor de camino, una vez accionado el botón el robot deberá comenzar a realizar las acciones pertinentes para cumplir con la trayectoria trazada, si esto no llegara a suceder deberíamos de realizar un análisis del programa para poder determinar los posibles errores, rectificarlos y poder solucionar el problema que no nos permite realizar las funciones necesarias.

Figura 74

Esquema de la ruta a seguir

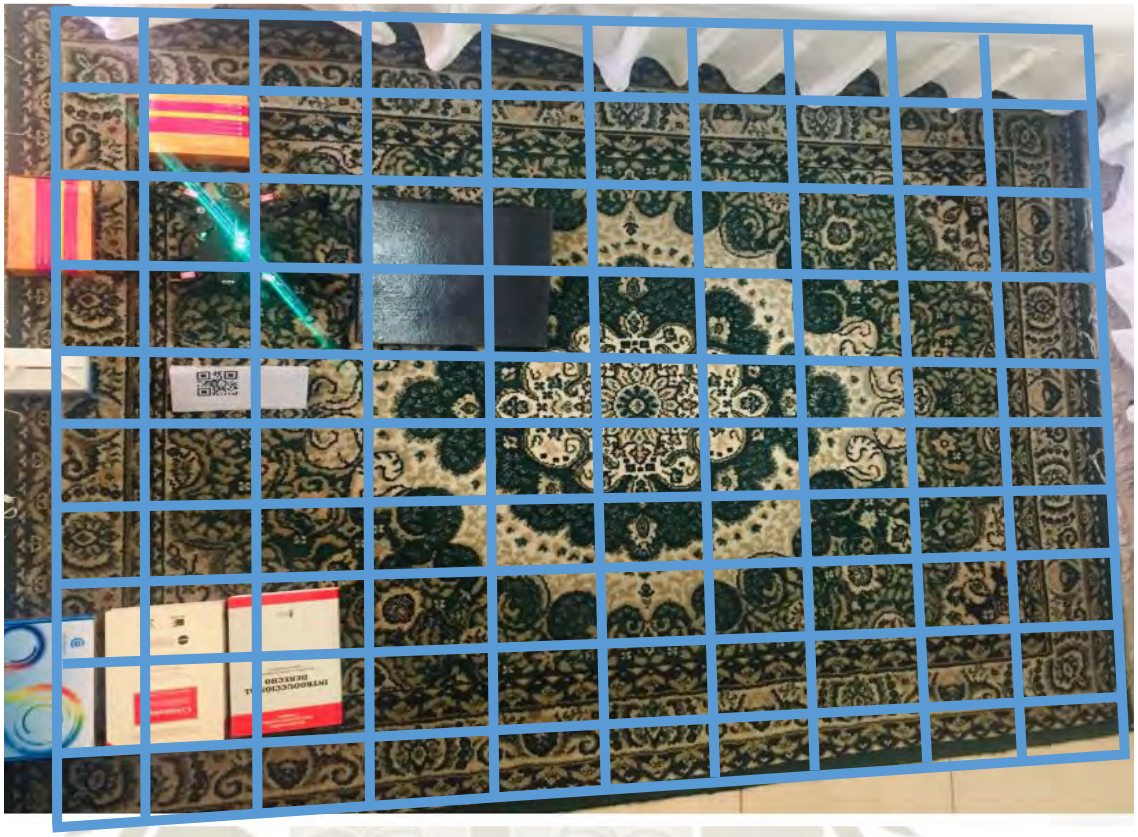


Fuente. Realizamos un esquema de la ruta que debe de seguir nuestro robot al momento de ejecutar los movimientos y seguir la ruta planificada. Adaptación propia en base a la investigación realizada (2020)

De ahí que, podemos observar en la figura 74, un pequeño esquema de la ruta que debería de seguir nuestro robot al momento de elaborar los movimientos con la trayectoria planificada a partir de los datos proporcionados, tanto por nuestro código QR, como por los valores dados al planificador de trayectorias. De acuerdo con lo que vimos y describimos anteriormente, podemos observar de manera gráfica, que el camino a planificado por nuestro programa ha sido optimo, no solo por llegar a un punto final, sino por haber logrado evadir los obstáculos con éxito. En otro orden de cosas, procederemos a realizar la conexión con el robot para ejecutar todos los programas y comprobar el control de movimientos de nuestro robot.

Figura 75

Conexión del robot y reconocimiento del código QR

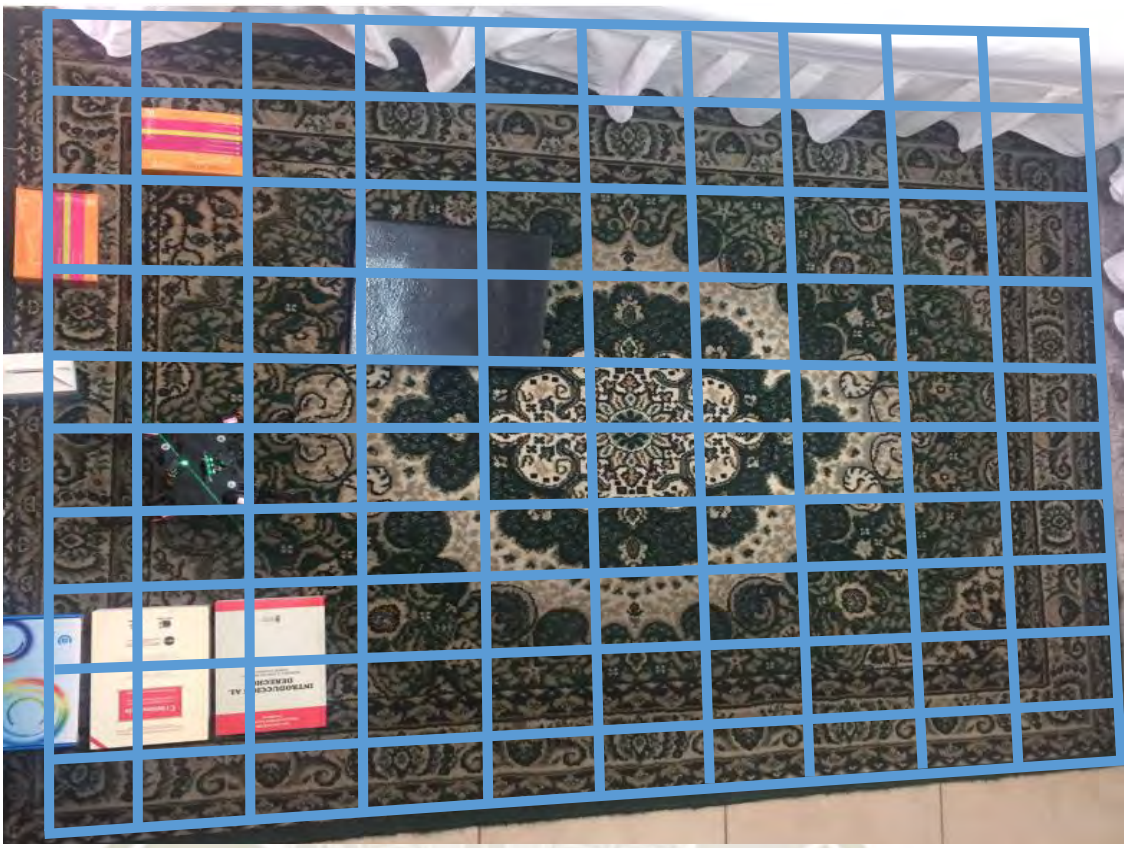


Fuente. Iniciamos la conexión con el robot a través del wifi y acto seguido procedemos a realizar el reconocimiento del código QR proporcionado. Adaptación propia en base a la investigación realizada (2020)

De esta manera podemos observar cómo es el real funcionamiento de nuestro robot al momento de ejecutar las funciones dispuestas por nuestra programación que se encuentran proporcionadas en nuestra interfaz gráfica, como se muestra en la figura 75, en esta primera instancia procederemos a la inicialización del robot, la conexión mediante el wifi (que será denotada por el color verde de los focos led de nuestro robot) y por último el reconocimiento del código QR a través de nuestra cámara web del robot, una vez realizada estas funciones podremos seguir con los siguientes pasos que a continuación mostraremos y explicaremos de manera breve para un mejor entendimiento de lo que viene sucediendo con nuestro robot y los pasos que viene ejecutando.

Figura 76

Movimiento del robot – Uno



Fuente. Tenemos un primer desplazamiento del robot, antes de realizar el giro. Adaptación propia en base a la investigación realizada (2020)

Puesto que hubo un reconocimiento adecuado del código QR y una planificación de la trayectoria de manera eficiente, como se mostró en la figura 73, es de esta manera que nuestro programa pudo realizar los movimientos adecuados según las especificaciones requeridas y proporcionadas por nuestro entorno y nuestro programa desarrollado. Es así, como se indica en la figura 76, que nuestro robot ha realizado un primer movimiento de desplazamiento que es el de movilizarse del punto de coordenada (2,3) hacia el punto de coordenada (2,6) donde se detendrá por un breve instante para poder realizar un cambio de dirección y poder desplazarse en el tramo final.

Figura 77
Giro del robot

Fuente. El robot realiza un pequeño giro antes de poder continuar con la ruta hacia la meta final. Adaptación propia en base a la investigación realizada (2020)

Tal y como dijimos anteriormente, nuestro robot procedió a detenerse brevemente para poder realizar un pequeño giro, como se observa en la figura 77, el cual le permitirá continuar con el último tramo de la ruta marcada por nuestro planificador de trayectorias. Sorprende comprobar que durante el giro hay un pequeño desalineamiento en la dirección de nuestro robot, pero comprobaremos que esto poco tiene que ver con algún error de programación. De igual forma este será un margen de error a tomar en cuenta durante el desarrollo de nuestro test y al momento de obtener los resultados requeridos.

Figura 78

Movimiento del robot – Dos

Fuente. Tramo final que realizó nuestro robot y llegada al punto final. Adaptación propia en base a la investigación realizada (2020)

De acuerdo a lo planteado en los objetivos mostrados en el capítulo uno de nuestro proyecto hemos procedido a realizar, en el presente capítulo, todos los módulos experimentales bajo el enfoque de *Navigation Stack* para poder obtener los resultados adecuados y realizar los ajustes pertinentes, de ser necesario, en cada uno de los pasos a seguir, con la finalidad de poder obtener los resultados adecuados que nos permitan seguir con un rumbo prospero al momento de culminar con la presente investigación. Más aún estos módulos experimentales nos han permitido obtener el diseño final de nuestro diseño del sistema de control de navegación utilizando inteligencia artificial para un cuadrúpedo de rescate, de esta manera hemos concluido con la realización de cada uno de los módulos experimentales de nuestro proyecto.

5.7.Validación del sistema de control

Después de haber realizado los módulos experimentales bajo el enfoque de *Navigation Stack* en los puntos anteriores del presente capítulo hemos realizado la implementación y complementación de los mismos en un solo modulo experimental el cual fue introducido dentro de nuestro robot para poder realizar los análisis necesarios junto con el desarrollo de su correcto funcionamiento para la presente investigación, haciendo que de esta forma, mediante nuestro robot se pueda realizar la validación de nuestro sistema de control. Por consiguiente, en el punto anterior (5.6) hemos podido observar el correcto funcionamiento de los cuatro módulos experimentales anteriores funcionando en conjunto para poder realizar las acciones solicitadas por el operador y así lograr obtener un resultado al final de la presente investigación que no solo nos permita valida nuestro diseño, sino, además podamos obtener las conclusiones necesarias y recomendaciones a tener en cuenta que presentaremos más adelante. Finalmente podemos concluir que la validación ha sido dada de una manera exitosa por lo mostrado en los puntos anteriores del capítulo cinco.

CONCLUSIONES

En esta tesis se diseñó el sistema de control de navegación, utilizando inteligencia artificial para un cuadrúpedo de rescate, porque implementaban acciones de control como el diseño y la programación.

En este proyecto se obtuvo el modelo algorítmico adecuado para nuestro sistema de control, hallando el sistema madre que nos permitió el correcto funcionamiento de nuestro robot.

En esta tesis se diseñó una interfaz que permita al usuario la manipulación y control del cuadrúpedo de rescate, a través del enfoque de Navigation Stack, desarrollando los pasos y procedimientos a seguir por parte de nuestro sistema para el funcionamiento adecuado de nuestro robot.

En esta investigación se realizaron los módulos experimentales bajo el enfoque del Navigation Stack para una mayor facilidad de pruebas simuladas, porque precisaban de funciones específicas como la identificación de códigos, planificación de trayectorias y ejecución de un trayecto.

Se determinó el algoritmo necesario a utilizar para el correcto funcionamiento de nuestro planificador de trayectorias, obteniendo.

En esta tesis se validó el diseño del sistema de control a través de nuestro robot, el cual fue nuestro sujeto de pruebas.

RECOMENDACIONES

El operador deberá tomar en cuenta que Python es un entorno libre y sufre diferentes modificaciones por diversos desarrolladores, encontrándose en constantes actualizaciones, por ello se le recomienda al operador utilizar las versiones anteriores para no sufrir alteraciones en el programa ni contratiempos al momento de realizar alguna mejora, en caso contrario se sugiere al operador revisar las librerías que hayan sido modificadas, cambiadas, sustituidas o eliminadas en la versión a utilizar para el adecuado desarrollo del programa o modificatoria del mismo.

Se recomienda al programador realizar copias de seguridad o respaldos de los diferentes programas a elaborar, con la finalidad de no tener problemas al momento de juntar los mismos o poder examinarlos por separado cuando alguna función específica no esté desarrollándose de manera adecuada, mejorando de esta forma el desarrollo de nuestro algoritmo.

En este proyecto recomendamos al usuario el uso de un Raspberry Pi de última generación, debido a su capacidad en la memoria RAM y el desarrollo de este al momento de ejecutar nuestro sistema. Sorprenderá, al usuario, tal vez que el robot no se desempeña de manera eficiente por la velocidad con que los datos son transferidos y como esto afecta al momento de ejecutar las diferentes funciones, todo esto se debe a la complejidad del Navigation Stack.

Con el objetivo de mejorar el rendimiento y autonomía del robot se debe de buscar una fuente de poder que pueda tener una vida de uso con mayor capacidad que la utilizada, de esta manera el robot tendrá un tiempo de autonomía, sin necesidad de ser recargado, mucho más larga.

Para mejorar los resultados requeridos, se recomienda al operario o desarrollador contar con un prototipo más robusto y con buenos acabados, de ser posible que sea fabricado por uno mismo y con las especificaciones necesarias. Es preciso que quién vaya a realizar la manipulación del programa tenga en cuenta esta recomendación, porque llega a influir considerablemente al momento de realizar las pruebas y los errores de precisión suelen variar demasiado. Por lo que la solución absoluta no solo se encuentra en la programación, sino también en el diseño del mismo.

REFENCIAS BIBILIOGRAFICAS

Adeept. (2021). Obtenido de www.adeept.com: https://www.adeept.com/

Alonso, R. (31 de Agosto de 2021). Obtenido de hardzone.es: https://hardzone.es/tutoriales/componentes/usos-raspberry-pi-principiantes/

Alonso, R. (31 de Agosto de 2021). *Hardzone.es*. Obtenido de https://hardzone.es/tutoriales/componentes/usos-raspberry-pi-principiantes/

Barra, O., & Barra, F. (2015). *Microcontroladores PIC con programación PBP*. Madrid: RA-MA.

Bravo, B., & Villegas, J. (2017). *Diseño e implementación de un prototipo de brazo robótico (4GDL) teleoperado para manipulación de sustancias tóxicas asistido con visión artificial y redes neuronales para laboratorios farmaceúticos.* Arequipa: Universidad Católica de Santa María.

Bürdek, B. (1994). *Diseño, teoría y práctica del diseño industrial.* Barcelona: Gustavo Gili.

Definición abc tu diccionario hecho fácil. (Mayo de 2016). Obtenido de www.definicionabc.com: https://www.definicionabc.com/general/rescate.php

Definiciones.de. (2008). Obtenido de definición.de: https://definicion.de/computadora/

Definiciones.de. (2011). Obtenido de Definiciones.de.com: https://definicion.de/rescate/

Definiciones-de.com. (06 de 09 de 2016). Obtenido de definiciones-de.com: www.definiciones-de.com/Definicion/de/cuadrupedo.php

Diéguez, L. (17 de Febrero de 2020). Obtenido de kolwidi.com: https://kolwidi.com/blogs/blog-kolwidi/arduino-vs-raspberry-pi

Diéguez, L. (17 de marzo de 2021). Obtenido de kolwidi.com: https://kolwidi.com/blogs/blog-kolwidi/la-increible-historia-de-raspberry-pi

Digital Guide Ionos. (2021). Obtenido de www.ionos.es: https://www.ionos.es/digitalguide/servidores/know-how/un-vistazo-a-proyectos-basados-en-raspberry-pi/

educalingo, 2019. (2019). Obtenido de educalingo.com: https://educalingo.com/es/dic-es/cuadrupedo

Epicor. (s.f.). *Epicor Software Corporation*.

ESSS. (25 de Enero de 2017). *ESSS*. Obtenido de https://www.esss.co/es/blog/los-pilares-de-la-industria-4-0/

Flores, L., & Garay, M. (2017). *Diseño e Implementación de un Robot Móvil de Desplazamiento Autónomo Basado en un Controlador Proporcional Derivativo.* Callao: Universidad Nacional del Callao.

Francés, M. (2021). *Diseño e Implementación del Control de la Navegación Autónoma de un Modelo a Escala de Embarcación.* Valencia: Universitat Politècnica de València.

Gispert, C. (2002). México: Océano.

Grayeb, S., & J.A. (5 de Diciembre de 2008). *Blogger*. Obtenido de blogger.com: http://losmicrocontroladores.blogspot.com/

IDELab. (2020). *IDELab Universidad de Valladolid*. Obtenido de idelab.uva.es.

Lemus, D. (s.f.). *sabermas*. Obtenido de sabermas Revista de Divulgación: https://www.sabermas.umich.mx/archivo/la-ciencia-en-pocas-palabras/264-numero-30/474-definiendo-la-robotica.html#:~:text=La%20Asociaci%C3%B3n%20Japonesa%20de%20Rob%C3%B3tica,respuesta%20a%20las%20%C3%B3rdenes%20humanas.

Lidia, A. (1994). *El diseño industrial en la historia.* Córdoba: Ediciones tec.

M. Martinez, V. F. (1995). Planificación de Trayectorias para Robots Móviles. En V. F. M. Martinez. Obtenido de http://webpersonal.uma.es/~VFMM/PDF/cap2.pdf

Mandado, E., Menéndez, L., Fernández, L., & López, E. (2007). *Microcontroladores PIC. Sistema integrado para el autoaprendizaje.* Barcelona: MARCOMBO.

Manufacturing Terms Definition at a click away. (s.f.). Obtenido de Manufacturing Terms Definition at a click away:

https://www.manufacturingterms.com/Spanish/Computer-Aided-Design-(CAD).html

MediaWiki. (17 de Octubre de 2011). Obtenido de EcuRed: https://www.ecured.cu/Dise%C3%B1o_asistido_por_computadora

MediaWiki. (2020). *MediaWiki*. Obtenido de EcuRed.

Medium. (23 de Enero de 2017). *Medium*. Obtenido de apLOOP: https://medium.com/@aploopve/microcontroladores-vs-microprocesadores-9e8c7edfb746

Nail, G. (2018). *Navegación autónoma de una base robótica usando SLAM.* Valparaíso: Pontificia Universidad Católica de Valparaíso.

Noguera, B. (2020). Obtenido de culturacion.com: https://culturacion.com/raspberry-pi-que-es-caracteristicas-y-precios/

Olaya, C., & Rodríguez, E. (2018). *Diseño de una arquitectura embebida para el cálculo cinemático inverso de una extremidad robótica hexápodo de tres grados de libertad mediante el algoritmo cordic en FPGA.* Trujillo: Universidad Privada Antenor Orrego.

Owaki, D., & Ishiguro, A. (2016). A Quadruped Robot Exhibiting Spontaneous Gait Transitions from Walking to Trotting to Galloping. *Scientific Reports*, 1.

Platzi. (2017). *Platzi*. Obtenido de https://platzi.com/blog/historia-python/

Ripipsa. (s.f.). *Ripipsa S.A. de C.V.* Obtenido de https://ripipsacobots.com/robots-autonomos/#:~:text=La%20definici%C3%B3n%20de%20Robots%20Aut%C3%B3nomos,directamente%20expl%C3%ADcito%20de%20los%20humanos.

Robledano, Á. (23 de Septiembre de 2019). *OpenWebinars S.L.* Obtenido de https://openwebinars.net/blog/que-es-python/

Rojas, O., & Rojas, L. (2006). Diseño asistido por computadora. *Diseño y Tecnología*, 8.

Sampedro, J. (26 de Enero de 2016). *Ediciones El País S.L.* Obtenido de El País: https://elpais.com/elpais/2016/01/26/ciencia/1453809513_840043.html

Sánchez, E. (2012). El concepto diseño en el taller de diseño reflexiones teóricas. *Insigne visual*, 3.

Sintes, B. (10 de Octubre de 2021). *mclibre*. Obtenido de https://www.mclibre.org/consultar/python/otros/historia.html

Solectroshop. (s.f.). *Solectro*. Obtenido de Solectro: https://solectroshop.com/es/10-placas-de-desarrollo#:~:text=Una%20placa%20de%20desarrollo%2C%20conocidas,requeridas%20en%20un%20ordenador%20funcional.

Tordesillas, J. (2015 - 2016). *Diseño y simulación del sistema de locomoción de un robot hexápodo para tareas de búsqueda y rescate.* Madrid: Universidad Politécnica de Madrid.

Turing, A. (1950). Maquinaria computadora e inteligencia. *Mind*, 5.

Adeept. (2021). Obtenido de www.adeept.com: https://www.adeept.com/

Alonso, R. (31 de Agosto de 2021). Obtenido de hardzone.es: https://hardzone.es/tutoriales/componentes/usos-raspberry-pi-principiantes/

Alonso, R. (31 de Agosto de 2021). *Hardzone.es*. Obtenido de https://hardzone.es/tutoriales/componentes/usos-raspberry-pi-principiantes/

Barra, O., & Barra, F. (2015). *Microcontroladores PIC con programación PBP.* Madrid: RA-MA.

Bravo, B., & Villegas, J. (2017). *Diseño e implementación de un prototipo de brazo robótico (4GDL) teleoperado para manipulación de sustancias tóxicas asistido con visión artificial y redes neuronales para laboratorios farmaceúticos.* Arequipa: Universidad Católica de Santa María.

Bürdek, B. (1994). *Diseño, teoría y práctica del diseño industrial.* Barcelona: Gustavo Gili.

Definición abc tu diccionario hecho fácil. (Mayo de 2016). Obtenido de www.definicionabc.com: https://www.definicionabc.com/general/rescate.php

Definiciones.de. (2008). Obtenido de definición.de: https://definicion.de/computadora/

Definiciones.de. (2011). Obtenido de Definiciones.de.com: https://definicion.de/rescate/

Definiciones-de.com. (06 de 09 de 2016). Obtenido de definiciones-de.com: www.definiciones-de.com/Definicion/de/cuadrupedo.php

Diéguez, L. (17 de Febrero de 2020). Obtenido de kolwidi.com: https://kolwidi.com/blogs/blog-kolwidi/arduino-vs-raspberry-pi

Diéguez, L. (17 de marzo de 2021). Obtenido de kolwidi.com: https://kolwidi.com/blogs/blog-kolwidi/la-increible-historia-de-raspberry-pi

Digital Guide Ionos. (2021). Obtenido de www.ionos.es: https://www.ionos.es/digitalguide/servidores/know-how/un-vistazo-a-proyectos-basados-en-raspberry-pi/

educalingo, 2019. (2019). Obtenido de educalingo.com: https://educalingo.com/es/dic-es/cuadrupedo

Epicor. (s.f.). *Epicor Software Corporation*.

ESSS. (25 de Enero de 2017). *ESSS*. Obtenido de https://www.esss.co/es/blog/los-pilares-de-la-industria-4-0/

Flores, L., & Garay, M. (2017). *Diseño e Implementación de un Robot Móvil de Desplazamiento Autónomo Basado en un Controlador Proporcional Derivativo.* Callao: Universidad Nacional del Callao.

Francés, M. (2021). *Diseño e Implementación del Control de la Navegación Autónoma de un Modelo a Escala de Embarcación.* Valencia: Universitat Politècnica de València.

Gispert, C. (2002). México: Océano.

Grayeb, S., & J.A. (5 de Diciembre de 2008). *Blogger*. Obtenido de blogger.com: http://losmicrocontroladores.blogspot.com/

IDELab. (2020). *IDELab Universidad de Valladolid*. Obtenido de idelab.uva.es.

Lemus, D. (s.f.). *sabermas*. Obtenido de sabermas Revista de Divulgación: https://www.sabermas.umich.mx/archivo/la-ciencia-en-pocas-palabras/264-numero-30/474-definiendo-la-robotica.html#:~:text=La%20Asociaci%C3%B3n%20Japonesa%20de%20Rob%C3%B3tica,respuesta%20a%20las%20%C3%B3rdenes%20humanas.

Lidia, A. (1994). *El diseño industrial en la historia.* Córdoba: Ediciones tec.

M. Martinez, V. F. (1995). Planificación de Trayectorias para Robots Móviles. En V. F. M. Martinez. Obtenido de http://webpersonal.uma.es/~VFMM/PDF/cap2.pdf

Mandado, E., Menéndez, L., Fernández, L., & López, E. (2007). *Microcontroladores PIC. Sistema integrado para el autoaprendizaje.* Barcelona: MARCOMBO.

Manufacturing Terms Definition at a click away. (s.f.). Obtenido de Manufacturing Terms Definition at a click away: https://www.manufacturingterms.com/Spanish/Computer-Aided-Design-(CAD).html

MediaWiki. (17 de Octubre de 2011). Obtenido de EcuRed: https://www.ecured.cu/Dise%C3%B1o_asistido_por_computadora

MediaWiki. (2020). *MediaWiki*. Obtenido de EcuRed.

Medium. (23 de Enero de 2017). *Medium*. Obtenido de apLOOP: https://medium.com/@aploopve/microcontroladores-vs-microprocesadores-9e8c7edfb746

Nail, G. (2018). *Navegación autónoma de una base robótica usando SLAM.* Valparaíso: Pontificia Universidad Católica de Valparaíso.

Noguera, B. (2020). Obtenido de culturacion.com: https://culturacion.com/raspberry-pi-que-es-caracteristicas-y-precios/

Olaya, C., & Rodríguez, E. (2018). *Diseño de una arquitectura embebida para el cálculo cinemático inverso de una extremidad robótica hexápodo de tres grados de libertad mediante el algoritmo cordic en FPGA.* Trujillo: Universidad Privada Antenor Orrego.

Owaki, D., & Ishiguro, A. (2016). A Quadruped Robot Exhibiting Spontaneous Gait Transitions from Walking to Trotting to Galloping. *Scientific Reports*, 1.

Platzi. (2017). *Platzi*. Obtenido de https://platzi.com/blog/historia-python/

Ripipsa. (s.f.). *Ripipsa S.A. de C.V.* Obtenido de https://ripipsacobots.com/robots-autonomos/#:~:text=La%20definici%C3%B3n%20de%20Robots%20Aut%C3%B3nomos,directamente%20expl%C3%ADcito%20de%20los%20humanos.

Robledano, Á. (23 de Septiembre de 2019). *OpenWebinars S.L.* Obtenido de https://openwebinars.net/blog/que-es-python/

Rojas, O., & Rojas, L. (2006). Diseño asistido por computadora. *Diseño y Tecnología*, 8.

Sampedro, J. (26 de Enero de 2016). *Ediciones El País S.L.* Obtenido de El País: https://elpais.com/elpais/2016/01/26/ciencia/1453809513_840043.html

Sánchez, E. (2012). El concepto diseño en el taller de diseño reflexiones teóricas. *Insigne visual*, 3.

Sintes, B. (10 de Octubre de 2021). *mclibre*. Obtenido de https://www.mclibre.org/consultar/python/otros/historia.html

Solectroshop. (s.f.). *Solectro*. Obtenido de Solectro: https://solectroshop.com/es/10-placas-de-desarrollo#:~:text=Una%20placa%20de%20desarrollo%2C%20conocidas,requeridas%20en%20un%20ordenador%20funcional.

Tordesillas, J. (2015 - 2016). *Diseño y simulación del sistema de locomoción de un robot hexápodo para tareas de búsqueda y rescate.* Madrid: Universidad Politécnica de Madrid.

Turing, A. (1950). Maquinaria computadora e inteligencia. *Mind*, 5.

ANEXOS DE PROGRAMACION

CÓDIGO DE PROGRAMACIÓN DEL INTERFAZ

```
def loop(): #GUI
      global
tcpClicSock,root,E1,connect,l_ip_4,l_ip_5,color_bt
n,color_text,Btn14,color_text,var_R,var_B,var_G,Bt
n_path,Btn_QR   #El valor de tcpClicSock cambia en
la función loop (), también cambiaría en global
para que las otras funciones pudieran usarlo.
      while True:
            color_bg='#000000'#Establecer el color
de fondo
            color_text='#000000'#Establecer el
color del texto
            color_btn='#5AECB8'#Establecer el color
de los botones
            color_line='#01579B'#Establecer el
color de las lineas
            color_can='#000000'#Establecer el color
del lienzo
            color_oval='#2196F3'#Establecer el
color ovalado
            target_color='#FF6D00'

            root = tk.Tk()#Definir una ventana
root.
            root.title('Tesis Giuliano')#Título
principal de la ventana.
            root.geometry('700x230')#Tamaño de la
ventana principal.
            root.config(bg=color_bg)#Establecer el
color de fondo de la ventana root.

            try:
                  logo =tk.PhotoImage(file =
'logo2.png')           #Definir la foto del logo
en formato '.png' o '.gif'
                  l_logo=tk.Label(root,image =
logo,bg=color_bg)   #Establecer la figura del logo
para mostrar como etiqueta.
```

```
                l_logo.place(x=30,y=13)
            #Posicionar la etiqueta en la parte
superior izquierda.
        except:
            pass


    l_ip_4=tk.Label(root,width=18,text='Disconnec
ted',fg=color_text,bg='#F76745')
        l_ip_4.place(x=550,y=50)
            #Definir una etiqueta y su
posición

        E1 =
tk.Entry(root,show=None,width=16,bg="#37474F",fg='
#eceff1')
        E1.place(x=350,y=60)
            #Definir una entrada y su
posición

        l_ip_3=tk.Label(root,width=10,text='IP
Address:',fg='#000000',bg='#FFFFFF')
        l_ip_3.place(x=350,y=35)
            #Definir una etiqueta y su
posición

        Btn_path = tk.Button(root, width=15,
text='Planning
Path',fg=color_text,bg=color_btn,relief='ridge')#D
efine un botón y su posición

        Btn_path.place(x=300,y=135)

        Btn_path.bind('<ButtonPress-1>',path)



        Btn_QR = tk.Button(root, width=8,
text='Scan
QR',fg=color_text,bg=color_btn,relief='ridge')#Def
ine un botón y su posición
```

```
            Btn_QR.place(x=500,y=135)

            Btn_QR.bind('<ButtonPress-
1>',call_camera)


            Btn_path = tk.Button(root, width=15,
text='MOVE',fg=color_text,bg=color_btn,relief='rid
ge')#Define un botón y su posición

            Btn_path.place(x=400,y=170)

            Btn_path.bind('<ButtonPress-1>',path)



            Btn14= tk.Button(root,
width=8,height=1,
text='Connect',fg=color_text,bg=color_btn,command=
connect_click,relief='ridge')
            Btn14.place(x=465,y=50)#Define un botón
y su posición

            root.bind('<Return>', connect)

            global stat
            if stat==0:# Asegurarse el mainloop
corra una sola vez
                  root.mainloop()# correr el
mainloop
                  stat=1# Cambiar el valor a 1 para
que el mainlop no se ejecute nuevamente
```

CÓDIGO DE PROGRAMACIÓN DEL LECTOR DEL CÓDIGO QR

```
import cv2
import numpy as np
from pyzbar.pyzbar import decode

cap = cv2.VideoCapture(0)
cap.set(3,640)
cap.set(4,480)

with open('myDataFile.txt') as f:
    myDataList = f.read().splitlines()

while True:

    success, img = cap.read()
    for barcode in decode(img):
        myData = barcode.data.decode('utf-8')
        print(myData)

        if myData in myDataList:
            myOutput = 'Punto de partida'
            myColor = (0,255,0)
        else:
            myOutput = 'Punto cualquiera'
            myColor = (0, 0, 255)

        pts = np.array([barcode.polygon],np.int32)
        pts = pts.reshape((-1,1,2))
        cv2.polylines(img,[pts],True,myColor,5)
        pts2 = barcode.rect

cv2.putText(img,myOutput,(pts2[0],pts2[1]),cv2.FON
T_HERSHEY_SIMPLEX,
                    0.9,myColor,2)

    cv2.imshow('Result',img)
    cv2.waitKey(1)
```

CÓDIGO DE PROGRAMACIÓN DEL PLANIFICADOR DE TRAYECTORIA

CÓDIGO DE PROGRAMACIÓN DEL ALGORITMO A*

```
def astar(m,startp,endp):
    w,h = 10,10         # grilla (10x10) es el
número de entrada de nuestro mapa
    sx,sy = startp      #Punto de Inicio
    ex,ey = endp  #Punto de Llegada

    node = [None,sx,sy,0,abs(ex-sx)+abs(ey-
sy)]
    closeList = [node]
    createdList = {}
    createdList[sy*w+sx] = node
    k=0
    while(closeList):
        node = closeList.pop(0)
        x = node[1]
        y = node[2]
        l = node[3]+1
        k+=1
        #encontrar las grillas vecinas
        if k!=0:
            neighbours = ((x,y+1),(x,y-
1),(x+1,y),(x-1,y))
        else:
            neighbours = ((x+1,y),(x-
1,y),(x,y+1),(x,y-1))
        for nx,ny in neighbours:
            if nx==ex and ny==ey:
                path = [(ex,ey)]
                while node:

path.append((node[1],node[2]))
                    node = node[0]
                return list(reversed(path))
            if 0<=nx<w and 0<=ny<h and
m[ny][nx]==0:
                if ny*w+nx not in createdList:
                    nn =
(node,nx,ny,l,l+abs(nx-ex)+abs(ny-ey))
```

```
                    createdList[ny*w+nx] = nn
                    #agregando closelist,
usando el montón binario
                    nni = len(closeList)
                    closeList.append(nn)
                    while nni:
                        i = (nni-1)>>1
                        if
closeList[i][4]>nn[4]:

closeList[i],closeList[nni] = nn,closeList[i]
                            nni = i
                        else:
                            break
    return []
```

CÓDIGO DE PROGRAMACIÓN DEL CAMINO

```
import cv2
import numpy as np
import time
from skimage.measure import compare_ssim as
ssim
#el algoritmo a* es el líder de busqueda de
caminos
#es utilizado en juegos como warcraft III
#usa el costo (menor distncia recorrida, menos
tiempo, etc)
#genera posibilidades y elige la que tiene el
menor costo proyectado
import astarsearch
#nos ayudará a pasar la imagen de derecha a
izquierda para procesarla
import traversal
from pyzbar.pyzbar import decode

cap = cv2.VideoCapture(0)
cap.set(3,640)
cap.set(4,480)

def path(start):


 maze = [[0 for i in range(10)] for i in
range(10)]
 obstacles =
[(1,3),(1,5),(1,8),(1,9),(2,2),(2,8),(2,9),(3,
3),(3,4),(3,8),(3,9),(4,3),(4,4),(5,3),(5,4)]

 for i in obstacles:
       maze[i[0]][i[1]]=1
       print("obstaculos")
 for i in range (10):
       for j in range (10):
             print(maze[i][j])
```

```
 result = astarsearch.astar(maze,start,(8,6))

 #imprimir el resultado

 list2=[]
 for t in result:
      x,y = t[0],t[1]
      list2.append(tuple((x,y)))#Contiene
ruta mínima + imagen inicial + imagen final
      result = list(list2[1:-1])# El
resultado contiene la ruta mínima requerida
      print (result)

 if not result:
 #Si no encuentra camino
      print('no hay camino')

      print (len(result))
      return len(result)



if __name__ == '__main__':

      while True:

            success, img = cap.read()
            for barcode in decode(img):
                  myData =
barcode.data.decode('utf-8')
                  print(myData)
                  myData = myData[1 : -1]
                  res = tuple(map(int,
myData.split(',')))
                  path(res)


                  cv2.imshow('Result',img)
                  cv2.waitKey(1)
```

CÓDIGO DE PROGRAMACIÓN DEL SEGUIDOR DE CAMINO

CÓDIGO DE PROGRAMACIÓN DE MOVIMIENTOS

```
def dove_move_tripod(step, speed, command):
 step_I  = step
 step_II = step+2
 step_III= step+4
 step_IV = step+6
 if step_II > 8:
      step_II = step_II - 8
 if step_III> 8:
      step_III= step_III- 8
 if step_IV > 8:
      step_IV = step_IV - 8

 if command == 'forward':
      for i in range(1,(pixel+1)):
           leg_tripod('I', step_I, i, speed)
           leg_tripod('II', step_II, i,
speed)
           leg_tripod('III', step_III, i,
speed)
           leg_tripod('IV', step_IV, i,
speed)

           await asyncio.sleep(3)

 elif command == 'backward':
      for i in range(1,(pixel+1)):
           leg_tripod('I', step_I, i, -speed)
           leg_tripod('II', step_II, i, -
speed)
           leg_tripod('III', step_III, i, -
speed)
           leg_tripod('IV', step_IV, i, -
speed)

           await asyncio.sleep(3)

 elif command == 'left':
      for i in range(1,(pixel+1)):
```

```
            leg_tripod('I', step_I, i, -
int(speed*turn_steady))
            leg_tripod('II', step_II, i, -
int(speed*turn_steady))

            leg_tripod('III', step_III, i,
speed)
            leg_tripod('IV', step_IV, i,
speed)

 elif command == 'right':
      for i in range(1,(pixel+1)):
            leg_tripod('I', step_I, i, speed)
            leg_tripod('II', step_II, i,
speed)

            leg_tripod('III', step_III, i, -
int(speed*turn_steady))
            leg_tripod('IV', step_IV, i, -
int(speed*turn_steady))
```

CÓDIGO DE PROGRAMACIÓN DEL SEGUIMIENTO DE CAMINO

```
turn_steady = 4/5

def follow path(step, speed):
 step_I  = step
 step_II = step+2
 step_III= step+4
 step_IV = step+6
 if step_II > 8:
       step_II = step_II - 8
 if step_III> 8:
       step_III= step_III- 8
 if step_IV > 8:
       step_IV = step_IV - 8

 if (x+1,y):
       for i in range(1,(pixel+1)):
             leg_tripod('I', step_I, i, speed)
             leg_tripod('II', step_II, i,
speed)

             leg_tripod('III', step_III, i,
speed)
             leg_tripod('IV', step_IV, i,
speed)

             await asyncio.sleep(3)
 elif (x-1,y):
       for i in range(1,(pixel+1)):
             leg_tripod('I', step_I, i, -speed)
             leg_tripod('II', step_II, i, -
speed)

             leg_tripod('III', step_III, i, -
speed)
             leg_tripod('IV', step_IV, i, -
speed)

             await asyncio.sleep(3)
 elif (x,y+1):
       for i in range(1,(pixel+1)):
```

```
                leg_tripod('I', step_I, i, -
int(speed*turn_steady))
                leg_tripod('II', step_II, i, -
int(speed*turn_steady))

                leg_tripod('III', step_III, i,
speed)
                leg_tripod('IV', step_IV, i,
speed)

                leg_tripod('I', step_I, i, speed)
                leg_tripod('II', step_II, i,
speed)

                leg_tripod('III', step_III, i,
speed)
                leg_tripod('IV', step_IV, i,
speed)

                await asyncio.sleep(3)
 elif (x,y-1):
        for i in range(1,(pixel+1)):
                leg_tripod('I', step_I, i, speed)
                leg_tripod('II', step_II, i,
speed)

                leg_tripod('III', step_III, i, -
int(speed*turn_steady))
                leg_tripod('IV', step_IV, i, -
int(speed*turn_steady))

                leg_tripod('I', step_I, i, speed)
                leg_tripod('II', step_II, i,
speed)

                leg_tripod('III', step_III, i,
speed)
                leg_tripod('IV', step_IV, i,
speed)

                await asyncio.sleep(3)
```

```
for t in result:
      x,y = t[0],t[1]
      print (x,y)

if not result:
#Si no encuentra camino
      print('no hay camino')

print (len(result))
return len(result)
```

CÓDIGO DE PROGRAMACIÓN COMPLETO

```
# Nombre del archivo : TESIS
# Descripción        : El siguiente programa tiene como fin demostrar que se puede dar un correcto funcionamiento de un programa para la navegación autónoma de un robot.
# Autor                  : Giuliano Nikola Montalvo Miranda
# Fecha              : 2019/08/22
#
import asyncio
import cv2
import zmq
import base64
import numpy as np
from socket import *
import sys
import time
import threading as thread
import tkinter as tk
from skimage.measure import compare_ssim as ssim
import astarsearch
import traversal
from pyzbar.pyzbar import decode

ip_stu=1          #Shows connection status
c_f_stu = 0
c_b_stu = 0
c_l_stu = 0
c_r_stu = 0
c_ls_stu= 0
c_rs_stu= 0
funcMode= 0
tcpClicSock = ''
root = ''
stat = 0

########>>>>>VIDEO<<<<<########

def video_thread():
    global footage_socket, font, frame_num, fps
```

```
        context = zmq.Context()
        footage_socket = context.socket(zmq.SUB)
        footage_socket.bind('tcp://*:5555')
        footage_socket.setsockopt_string(zmq.SUBSCRIB
E, np.unicode(''))

        font = cv2.FONT_HERSHEY_SIMPLEX

        frame_num = 0
        fps = 0

def get_FPS():
        global frame_num, fps
        while 1:
                try:
                        time.sleep(1)
                        fps = frame_num
                        frame_num = 0
                except:
                        time.sleep(1)

def opencv_r():
        global frame_num
        while True:
                try:
                        frame =
footage_socket.recv_string()
                        img = base64.b64decode(frame)
                        npimg = np.frombuffer(img,
dtype=np.uint8)
                        source = cv2.imdecode(npimg, 1)
                        cv2.imshow("Stream", source)
                        frame_num += 1
                        cv2.waitKey(1)

                except:
                        time.sleep(0.5)
                        break

fps_threading=thread.Thread(target=get_FPS)
#Definir un hilo para FPV y OpenCV
```

```
fps_threading.setDaemon(True)
#'True'significa que es un hilo frontal, se
cerraría cuando se cierra mainloop ()
fps_threading.start()                          #El
hilo comienza

video_threading=thread.Thread(target=video_thread)
      #Definir un hilo para FPV y OpenCV
video_threading.setDaemon(True)
#'True'significa que es un hilo frontal, se
cerraría cuando se cierra mainloop ()
video_threading.start()
#El hilo comienza

########>>>>>VIDEO<<<<<########


def replace_num(initial,new_num):       #Llamamos
a esta función para reemplazar los datos en el
archivo '.txt'
     newline=""
     str_num=str(new_num)
     with open("ip.txt","r") as f:
          for line in f.readlines():
               if(line.find(initial) == 0):
                    line = initial+"%s"
%(str_num)
               newline += line
     with open("ip.txt","w") as f:
          f.writelines(newline) #Llamamos a esta
función para reemplazar los datos en el archivo
'.txt'


def num_import(initial):          #Llamamos a esta
función para importar datos del archivo '.txt'
     with open("ip.txt") as f:
          for line in f.readlines():
               if(line.find(initial) == 0):
                    r=line
     begin=len(list(initial))
     snum=r[begin:]
```

```
        n=snum
        return n

def call_camera(causa):
        #cap = cv2.VideoCapture(0)
        #cap.set(3,640)
        #cap.set(4,480)
        print(causa)
        #success, img = cap.read()
        frame = footage_socket.recv_string()
        img = base64.b64decode(frame)
        for barcode in decode(img):
                myData = barcode.data.decode('utf-
8')
                print(myData)
                myData = myData[1 : -1]
                res = tuple(map(int,
myData.split(',')))
                print(res)


def qr_code(start):
        import cv2
        import numpy as np
        from pyzbar.pyzbar import decode

        cap = cv2.VideoCapture(0)
        cap.set(3,640)
        cap.set(4,480)

        with open('myDataFile.txt') as f:
                myDataList = f.read().splitlines()

        while True:

                success, img = cap.read()
                for barcode in decode(img):
                        myData = barcode.data.decode('utf-
8')
                        print(myData)

                        if myData in myDataList:
```

```
                        myOutput = 'Punto de
partida'
                        myColor = (0,255,0)
                else:
                        myOutput = 'Punto
cualquiera'
                        myColor = (0, 0, 255)

                pts =
np.array([barcode.polygon],np.int32)
                pts = pts.reshape((-1,1,2))

        cv2.polylines(img,[pts],True,myColor,5)
                pts2 = barcode.rect

        cv2.putText(img,myOutput,(pts2[0],pts2[1]),cv
2.FONT_HERSHEY_SIMPLEX,0.9,myColor,2)

            cv2.imshow('Result',img)
            cv2.waitKey(1)


def path(start):


        maze = [[0 for i in range(10)] for i in
range(10)]
        obstacles =
[(1,3),(1,5),(1,8),(1,9),(2,2),(2,8),(2,9),(3,3),(
3,4),(3,8),(3,9),(4,3),(4,4),(5,3),(5,4)]

        for i in obstacles:
              maze[i[0]][i[1]]=1
              print("obstaculos")
        for i in range (10):
              for j in range (10):
                    print(maze[i][j])

        result = astarsearch.astar(maze,start,(8,6))

        #imprimir el resultado
```

```
        list2=[]
        for t in result:
                x,y = t[0],t[1]
                list2.append(tuple((x,y)))
        #Contiene ruta mínima + imagen inicial +
imagen final
                result = list(list2[1:-1])
        #El resultado contiene la ruta mínima
requerida
                print (result)

        if not result:
        #Si no encuentra camino
                print('no hay camino')

                print (len(result))
                return len(result)


def follow_path(start):
        turn_steady = 4/5
        step_I  = step
        step_II = step+2
        step_III= step+4
        step_IV = step+6
        if step_II > 8:
                step_II = step_II - 8
        if step_III> 8:
                step_III= step_III- 8
        if step_IV > 8:
                step_IV = step_IV - 8

        if (x+1,y):
                for i in range(1,(pixel+1)):
                        leg_tripod('I', step_I, i, speed)
                        leg_tripod('II', step_II, i,
speed)

                        leg_tripod('III', step_III, i,
speed)
                        leg_tripod('IV', step_IV, i,
speed)
```

```
                    await asyncio.sleep(3)
          elif (x-1,y):
                for i in range(1,(pixel+1)):
                    leg_tripod('I', step_I, i, -speed)
                    leg_tripod('II', step_II, i, -
speed)

                    leg_tripod('III', step_III, i, -
speed)
                    leg_tripod('IV', step_IV, i, -
speed)

                    await asyncio.sleep(3)
          elif (x,y+1):
                for i in range(1,(pixel+1)):
                    leg_tripod('I', step_I, i, -
int(speed*turn_steady))
                    leg_tripod('II', step_II, i, -
int(speed*turn_steady))

                    leg_tripod('III', step_III, i,
speed)
                    leg_tripod('IV', step_IV, i,
speed)

                    leg_tripod('I', step_I, i, speed)
                    leg_tripod('II', step_II, i,
speed)

                    leg_tripod('III', step_III, i,
speed)
                    leg_tripod('IV', step_IV, i,
speed)

                    await asyncio.sleep(3)
          elif (x,y-1):
                for i in range(1,(pixel+1)):
                    leg_tripod('I', step_I, i, speed)
                    leg_tripod('II', step_II, i,
speed)
```

```
                leg_tripod('III', step_III, i, -
int(speed*turn_steady))
                leg_tripod('IV', step_IV, i, -
int(speed*turn_steady))

                leg_tripod('I', step_I, i, speed)
                leg_tripod('II', step_II, i,
speed)

                leg_tripod('III', step_III, i,
speed)
                leg_tripod('IV', step_IV, i,
speed)

                await asyncio.sleep(3)


        for t in result:
            x,y = t[0],t[1]
            print (x,y)

        if not result:
        #Si no encuentra camino
            print('no hay camino')

        print (len(result))
        return len(result)

if __name__ == '__main__':

            while True:

                success, img = cap.read()
                for barcode in decode(img):
                    myData =
barcode.data.decode('utf-8')
                    print(myData)
                    myData = myData[1 : -1]
                    res = tuple(map(int,
myData.split(',')))
                    path(res)
```

```
                    cv2.imshow('Result',img)
                    cv2.waitKey(1)


def call_forward(event):          #Cuando se llama
a esta función, se ordena al robot que avance
      global c_f_stu
      if c_f_stu == 0:
            tcpClicSock.send(('forward').encode())
            c_f_stu=1


def call_back(event):             #Cuando se llama
a esta función, se ordena al robot que retroceda
      global c_b_stu
      if c_b_stu == 0:
            tcpClicSock.send(('backward').encode())
            c_b_stu=1


def call_FB_stop(event):          #Cuando se llama
a esta función, se ordena al robot que detenga sus
movimientos
      global
c_f_stu,c_b_stu,c_l_stu,c_r_stu,c_ls_stu,c_rs_stu
      c_f_stu=0
      c_b_stu=0
      tcpClicSock.send(('DS').encode())


def call_Turn_stop(event):          #Cuando se
llama a esta función, se ordena al robot que se
detenga
      global
c_f_stu,c_b_stu,c_l_stu,c_r_stu,c_ls_stu,c_rs_stu
      c_l_stu=0
      c_r_stu=0
      c_ls_stu=0
      c_rs_stu=0
      tcpClicSock.send(('TS').encode())
```

```
def call_Left(event):              #Cuando se llama
a esta función, se ordena al robot que se mueva a
la izquierda
      global c_l_stu
      if c_l_stu == 0:
            tcpClicSock.send(('left').encode())
            c_l_stu=1


def call_Right(event):                    #Cuando
se llama a esta función, se ordena al robot que se
mueva a la derecha
      global c_r_stu
      if c_r_stu == 0:
            tcpClicSock.send(('right').encode())
            c_r_stu=1


def call_headup(event):
      tcpClicSock.send(('headup').encode())


def call_headdown(event):
      tcpClicSock.send(('headdown').encode())


def call_headleft(event):
      tcpClicSock.send(('low').encode())


def call_headright(event):
      tcpClicSock.send(('hight').encode())


def call_headhome(event):
      tcpClicSock.send(('headhome').encode())


def connection_thread():
      global funcMode, Switch_3, Switch_2,
Switch_1, SmoothMode
```

```
        while 1:
                car_info =
(tcpClicSock.recv(BUFSIZ)).decode()

def Info_receive():
        global CPU_TEP,CPU_USE,RAM_USE
        HOST = ''
        INFO_PORT = 2256
        #Definamos el puerto serie
        ADDR = (HOST, INFO_PORT)
        InfoSock = socket(AF_INET, SOCK_STREAM)
        InfoSock.setsockopt(SOL_SOCKET,SO_REUSEADDR,1
)
        InfoSock.bind(ADDR)
        InfoSock.listen(5)
#Iniciamos el servidor y esperamos al usuario
        InfoSock, addr = InfoSock.accept()
        print('Info connected')

def socket_connect():
#Llamamos esta función para conectarnos con el
servidor
        global ADDR,tcpClicSock,BUFSIZ,ip_stu,ipaddr
        ip_adr=E1.get()
#Obtenemos la dirección IP de la entrada

        if ip_adr == '':                    #Si no se
ingresa una dirección IP en la entrada, importe
una IP predeterminada
                ip_adr=num_import('IP:')
                l_ip_4.config(text='Connecting')
                l_ip_4.config(bg='#FF8F00')

        #l_ip_5.config(text='Default:%s'%ip_adr)
                pass

        SERVER_IP = ip_adr
        SERVER_PORT = 10223
#Definimos el puerto serie
        BUFSIZ = 1024
#Definir tamaño de búfer
        ADDR = (SERVER_IP, SERVER_PORT)
```

```
        tcpClicSock = socket(AF_INET, SOCK_STREAM)
#Establecemos valor de conexión para el zócalo

        for i in range (1,6):
#Intento de 5 veces
            #try:
            if ip_stu == 1:
                print("Connecting to server @
%s:%d..." %(SERVER_IP, SERVER_PORT))
                print("Connecting")
                tcpClicSock.connect(ADDR)
        #Conección con el servidor

                print("Connected")


        #l_ip_5.config(text='IP:%s'%ip_adr)
                l_ip_4.config(text='Connected')
                l_ip_4.config(bg='#558B2F')

                replace_num('IP:',ip_adr)
                E1.config(state='disabled')
#Deshabilitar la entrada
                Btn14.config(state='disabled')
#Deshabilitar la entrada

                ip_stu=0
                #'0' significa conectado


        connection_threading=thread.Thread(target=con
nection_thread)     #Definir un hilo para FPV y
OpenCV

        connection_threading.setDaemon(True)
                #'True' significa que es un hilo
frontal, se cerraría cuando se cierra mainloop ()
                connection_threading.start()
                        #El hilo comienza
```

```
        info_threading=thread.Thread(target=Info_rece
ive)          #Definir un hilo para FPV y OpenCV
                    info_threading.setDaemon(True)
                              #'True' means it is a
front thread,it would close when the mainloop()
closes
                    info_threading.start()
                              #El hilo comienza


        video_threading=thread.Thread(target=opencv_r
)                       #Definir un hilo para FPV y
OpenCV
                    video_threading.setDaemon(True)
                              #'True' significa que es
un hilo frontal, se cerraría cuando se cierra
mainloop ()
                    video_threading.start()
                              #El hilo comienza

                    break
                else:
                    print("Cannot connecting to
server,try it latter!")
                    l_ip_4.config(text='Try %d/5
time(s)'%i)
                    l_ip_4.config(bg='#EF6C00')
                    print('Try %d/5 time(s)'%i)
                    ip_stu=1
                    time.sleep(1)
                    continue

        if ip_stu == 1:
            l_ip_4.config(text='Disconnected')
            l_ip_4.config(bg='#F44336')


def connect(event):       #Llamamos esta función
para conectarse con el servidor
        if ip_stu == 1:
```

```
            sc=thread.Thread(target=socket_connect)
#Definir un hilo para la conexión
            sc.setDaemon(True)
      #'True' significa que es un hilo frontal, se
cerraría cuando se cierra mainloop ()
            sc.start()                  #El hilo
comienza


def connect_click():      #Llamamos esta función
para conectarse con el servidor
      if ip_stu == 1:
            sc=thread.Thread(target=socket_connect)
#Definir un hilo para la conexión
            sc.setDaemon(True)
      #'True' significa que es un hilo frontal, se
cerraría cuando se cierra mainloop ()
            sc.start()                  #El hilo
comienza



def loop():                                 #GUI
      global
tcpClicSock,root,E1,connect,l_ip_4,l_ip_5,color_bt
n,color_text,Btn14,color_text,var_R,var_B,var_G,Bt
n_path,Btn_QR   #El valor de tcpClicSock cambia en
la función loop (), también cambiaría en global
para que las otras funciones pudieran usarlo.
      while True:
            color_bg='#000000'      #Establecer el
color de fondo
            color_text='#000000'    #Establecer el
color del texto
            color_btn='#5AECB8'     #Establecer el
color de los botones
            color_line='#01579B'    #Establecer el
color de las lineas
            color_can='#000000'     #Establecer el
color del lienzo
            color_oval='#2196F3'    #Establecer el
color ovalado
```

```
            target_color='#FF6D00'

            root = tk.Tk()              #Definir una
ventana root.
            root.title('Tesis Giuliano')
      #Título principal de la ventana.
            root.geometry('700x230')        #Tamaño
de la ventana principal.
            root.config(bg=color_bg)
#Establecer el color de fondo de la ventana root.

            try:
                logo =tk.PhotoImage(file =
'logo2.png')          #Definir la foto del logo
en formato '.png' o '.gif'
                l_logo=tk.Label(root,image =
logo,bg=color_bg)   #Establecer la figura del logo
para mostrar como etiqueta.
                l_logo.place(x=30,y=13)
              #Posicionar la etiqueta en la parte
superior izquierda.
            except:
                pass


      l_ip_4=tk.Label(root,width=18,text='Disconnec
ted',fg=color_text,bg='#F76745')
            l_ip_4.place(x=550,y=50)
                  #Definir una etiqueta y su
posición

            E1 =
tk.Entry(root,show=None,width=16,bg="#37474F",fg='
#eceff1')
            E1.place(x=350,y=60)
                  #Definir una entrada y su
posición

            l_ip_3=tk.Label(root,width=10,text='IP
Address:',fg='#000000',bg='#FFFFFF')
```

```
            l_ip_3.place(x=350,y=35)
                  #Definir una etiqueta y su
posición

            Btn_path = tk.Button(root, width=15,
text='Planning
Path',fg=color_text,bg=color_btn,relief='ridge')#D
efine un botón y su posición
            Btn_path.place(x=300,y=135)
            Btn_path.bind('<ButtonPress-1>',path)



            Btn_QR = tk.Button(root, width=8,
text='Scan
QR',fg=color_text,bg=color_btn,relief='ridge')#Def
ine un botón y su posición
            Btn_QR.place(x=500,y=135)
            Btn_QR.bind('<ButtonPress-
1>',call_camera, call_qr_code)


            Btn_path = tk.Button(root, width=15,
text='MOVE',fg=color_text,bg=color_btn,relief='rid
ge')#Define un botón y su posición
            Btn_path.place(x=400,y=170)
            Btn_path.bind('<ButtonPress-
1>',call_follow_path)



            Btn14= tk.Button(root,
width=8,height=1,
text='Connect',fg=color_text,bg=color_btn,command=
connect_click,relief='ridge')
            Btn14.place(x=465,y=50)
#Define un botón y su posición

            root.bind('<Return>', connect)

            global stat
```

```
if stat==0:                    # Asegurarse que el
mainloop corra una sola vez
                    root.mainloop()           # correr
el mainloop
                    stat=1                    #
Cambiar el valor a 1 para que el mainlop no se
ejecute nuevamente


if __name__ == '__main__':
      try:
                loop()                        # Cargar
el GUI
      except:
                tcpClicSock.close()           # Cierre el socket o
es posible que no vuelva a conectarse con el servidor
                footage_socket.close()
                cv2.destroyAllWindows()
                pass
```