UACM

Universidad Autónoma
de la Ciudad de México

NADA HUMANO ME ES AJENO

COLEGIO DE CIENCIA Y TECNOLOGÍA
LICENCIATURA EN INGENIERÍA EN SISTEMAS ELECTRÓNICOS
Y DE TELECOMUNICACIONES

Sistema electrónico basado en la generación automática de rutas peatonales para personas con capacidades visuales diferentes

T E S I S

QUE PARA OPTAR POR EL TÍTULO DE

LICENCIADO EN INGENIERÍA EN SISTEMAS ELECTRÓNICOS Y DE TELECOMUNICACIONES

P R E S E N T A

MARCO RAFAEL SOTO VARGAS

D I R E C T O R **DR. EDUARDO RAMOS DÍAZ**

C O D I R E C T O R **DR. AGUSTÍN ALMARAZ DAMIAN**

Ciudad de México, mayo de 2026.

UACM
Universidad Autónoma de la Ciudad de México
Nada humano me es ajeno

CA Coordinación Académica

SISTEMA BIBLIOTECARIO DE INFORMACIÓN Y DOCUMENTACIÓN

UNIVERSIDAD AUTÓNOMA DE LA CIUDAD DE MÉXICO
COORDINACIÓN ACADÉMICA

RESTRICCIONES DE USO PARA LAS TESIS DIGITALES

DERECHOS RESERVADOS©

UACM
Universidad Autónoma de la Ciudad de México
NADA HUMANO ME ES AJENO

COLEGIO DE CIENCIA Y TECNOLOGÍA

LICENCIATURA EN INGENIERÍA EN SISTEMAS ELECTRÓNICOS Y DE TELECOMUNICACIONES

SISTEMA ELECTRÓNICO BASADO EN LA GENERACIÓN AUTOMÁTICA DE RUTAS PEATONALES PARA PERSONAS CON CAPACIDADES VISUALES DIFERENTES

T E S I S

QUE PARA OPTAR POR EL TÍTULO DE

LICENCIADO EN INGENIERÍA EN SISTEMAS ELECTRÓNICOS Y DE TELECOMUNICACIONES

P R E S E N T A :

MARCO RAFAEL SOTO VARGAS

DIRECTOR: **DR. EDUARDO RAMOS DIAZ**
CO-DIRECTOR: **DR. AGUSTIN ALMARAZ DAMIAN**

Ciudad de México, mayo de 2024

RESUMEN

En este trabajo de tesis se presenta el diseño de un sistema automático de generación de rutas para personas con capacidades diferentes. El diseño presentado incluye la construcción de un sistema portátil basado en una tarjeta SoC de última generación en la cual se programará un algoritmo de inteligencia artificial cuyo propósito será el de generar rutas automáticas útiles para el usuario del sistema. Considerando que el usuario será una persona con capacidades visuales diferentes, los comandos de operación serán audibles. Una cámara de alta definición dará lectura a códigos QR posicionados en lugares diversos del plantel San Lorenzo Tezonco de la Universidad Autónoma de la Ciudad de México, los cuales servirán de referencia espacial al algoritmo desarrollado. Comandos audibles e instrucciones al usuario serán captados por medio de un micrófono y altavoz conectados al hardware principal.

La utilización del sistema propuesto dotará al usuario de información relevante para transitar en lugares físicos desconocidos minimizando la necesidad de ayuda por parte de terceros.

AGRADECIMIENTOS

Agradezco profundamente a mis padres por su apoyo durante toda la carrera. En especial, a mi madre, en quien encontré la fuerza para creer en mí mismo. También expreso mi sincero agradecimiento al Dr. Ramos, por su valioso apoyo y por los conocimientos que contribuyeron significativamente a mi formación académica y desarrollo profesional.

CONTENIDO GENERAL

Capítulo 5. Pruebas realizadas y resultados obtenidos

Capítulo 6. Conclusiones

ÍNDICE DE FIGURAS

CAPÍTULO 1

INTRODUCCIÓN

1.1. Introducción

Las personas con discapacidad visual tienen dificultades para desplazarse de forma independiente y segura, lo que afecta su calidad de vida. La falta de información sobre su entorno, como la ubicación de los servicios esenciales, es un obstáculo significativo.

Las tecnologías asistenciales, como los sistemas de navegación por voz, las aplicaciones móviles especializadas y los dispositivos portátiles emergentes, pueden ayudar a las personas con discapacidad visual a tomar decisiones informadas sobre sus desplazamientos.

El Gobierno Federal de México ha reconocido la necesidad de abordar específicamente la movilidad de personas con discapacidad visual. En septiembre de 2020, presentó un plan de movilidad que incluye el desarrollo de soluciones tecnológicas para facilitar la orientación espacial y la toma de decisiones para este segmento de la población.

Las personas con discapacidad visual necesitan ayuda de terceras personas al momento de la toma de decisiones para el traslado dentro de las instalaciones de diferentes espacios públicos, lo cual reduce la movilidad de las personas.
Existen diferentes propuestas en la literatura que abordan esta problemática, que van desde el trazado de rutas usando GPS y dispositivos vestibles, la autonomía de personas ciegas en supermercados, sistemas

de orientación para personas invidentes por medio de comandos de voz y hasta mensajería instantánea para personas con ceguera.
En el trabajo que se presenta, se propone una forma automática de proporcionar información

referente al sitio en que se encuentre el usuario y con ello pueda tomar decisiones basadas en información verídica del lugar en cuestión.

El sistema propuesto estará compuesto de un mapa del lugar, en donde se indiquen los servicios que se brindan tales como: sanitarios, salas, salones, servicio médico, comedor, entre otros. Basado en el mapa del lugar y localización, el sistema entregará de forma audible la información necesaria al usuario, para que éste conozca las posibilidades con que cuenta para formar una ruta y llegar a su destino. El sistema se conformará por hardware portátil y software desarrollado con el interés de asegurar su portabilidad.

Para realizar las pruebas del sistema desarrollado, se ha propuesto el plantel San Lorenzo Tezonco de la Universidad Autónoma de la Ciudad de México. Con base en las pruebas desarrolladas, el sistema es capaz de proporcionar una ruta peatonal, generada automáticamente, de circulación por el plantel de la universidad.

1.2 Antecedentes

En la actualidad de las innovaciones tecnológicas destinadas a mejorar la vida de las personas con discapacidades visuales, encontramos una serie de iniciativas enfocadas en proporcionar soluciones tecnológicas adaptadas a las necesidades específicas de movilidad.

El documento "Movilidad 4s para México: Saludable, Segura, Sustentable y Solidaria. Plan de Movilidad para una nueva normalidad" [1] fue presentado por el Gobierno Federal en septiembre de 2020 como respuesta a la "Emergencia Sanitaria por COVID-19". Este plan, elaborado en colaboración con diversas organizaciones, la Secretaría de Salud (SSA), la Secretaría de Desarrollo Agrario, Territorial y Urbano (SEDATU), y la oficina de la Organización Panamericana de la Salud (OPS) en México, tiene como objetivo abordar las necesidades de movilidad de personas y mercancías, así como adaptar nuestras

sociedades y territorios para reactivar la economía de manera saludable, segura, sustentable y solidaria. En este contexto, se proponen 4 ejes rectores, 12 estrategias y 7 metas para lograr la reactivación económica. Uno de los problemas descritos en el eje de Solidaridad es la movilidad de personas con discapacidad visual.

De acuerdo con el censo 2020 del INEGI en México [2] el 4.9% de la población en México sufre algún tipo de discapacidad, en donde la primera situación limitante es la movilidad y la segunda está relacionada con problemas de la visión.

A continuación, se destacan algunos de los dispositivos más sobresalientes y relevantes que

han surgido como parte de estos esfuerzos:

- Sistemas de navegación para personas ciegas

Estos sistemas utilizan una variedad de sensores, como cámaras, radares o ultrasonidos, para recopilar información sobre el entorno. La información se utiliza luego para generar una ruta segura para la persona ciega. Un ejemplo de un sistema de navegación para personas ciegas es el OrCam MyEye 2. Este sistema
[3] utiliza una cámara para capturar imágenes del entorno. La información se utiliza luego para generar una ruta segura, que se muestra a la persona ciega a través de una pantalla pequeña.

- Tecnologías de asistencia para personas con discapacidad visual

Estas tecnologías incluyen una variedad de dispositivos que pueden ayudar a las personas con discapacidad visual a realizar tareas cotidianas. Por ejemplo, los dispositivos de asistencia pueden ayudar a las personas a leer, escribir o desplazarse.
Un ejemplo de una tecnología de asistencia para personas con discapacidad visual es el BrailleNote Apex. Este dispositivo [4] es una computadora portátil que utiliza el sistema Braille para mostrar texto y gráficos.

- Gestos de control para personas con discapacidad visual

Estos sistemas permiten a las personas con discapacidad visual controlar dispositivos electrónicos mediante gestos. Los gestos pueden realizarse con las manos, la cabeza o el cuerpo.
Un ejemplo de un sistema de gestos de control para personas con discapacidad visual es el Eye-Gaze Control. Este sistema permite a las personas con discapacidad visual controlar dispositivos electrónicos mediante el seguimiento de sus movimientos oculares.

En el trabajo “Movilidad para invidentes utilizando el GPS del teléfono inteligente y un dispositivo tác-til vestible” [5] se presenta un sistema vestible para ayudar a la movilidad de personas videntes y débiles visuales en entornos urbanos usando un celular. Usa patrones de vibración y se comunica con una inter-faz táctil conectada a los zapatos del usuario.

En [6] se pretende mejorar la autonomía de los invidentes cuando realizan compras en un supermercado por medio de una aplicación basada en realidad aumentada sobre un dispositivo vestible e internet de las cosas. Los autores reportan mejoras en la autonomía de los usuarios al momento de realizar compras en supermercados.

Existen diferentes propuestas en la literatura que abordan esta problemática que van desde el trazado de rutas usando GPS y dispositivos vestibles, la autonomía de personas ciegas en supermercados, sistemas de orientación para personas invidentes por medio de comandos de

voz y hasta mensajería instantánea para personas con ceguera.

1.3 Planteamiento del problema

La movilidad para las personas con discapacidad visual es un desafío diario que impacta significativamente su calidad de vida. Esta población enfrenta barreras considerables en su capacidad para desplazarse de manera independiente y segura, lo que exige la necesidad de buscar soluciones que mejoren su movilidad y autonomía.
La falta de acceso a información ambiental, como la disposición de los espacios públicos y la ubicación de servicios esenciales, representa un obstáculo significativo para las personas con discapacidad visual. En este contexto, diversas organizaciones, como la Fundación ONCE, destacan la importancia de la tecnología asistencial para abordar estos desafíos. Tecnologías como los sistemas de navegación por voz, aplicaciones móviles especializadas y dispositivos portátiles emergentes se han convertido en herramien-tas esenciales para empoderar a las personas con discapacidad visual, proporcionándoles información en tiempo real sobre su entorno y ayudándolas a tomar decisiones informadas sobre sus desplazamientos.

1.4 Propuesta de solución

Se propone una solución automática para proporcionar información relevante al usuario sobre su ubicación, permitiéndole tomar decisiones basadas en información precisa del lugar. El sistema propuesto incluirá un mapa del lugar que identifique los servicios disponibles, como sanitarios, salas, salones, servicios médicos, comedores, entre otros. Basándose en el mapa y la ubicación del usuario, el sistema ofrecerá, basado en la generación automática de rutas peatonales, información audible para que el usuario conozca las opciones disponibles y pueda dirigirse hacia el destino deseado. Este sistema, diseñado con portabilidad en mente, constará de hardware portátil y software desarrollado para garantizar su accesibilidad y utilidad en diversas situaciones.

1.5 Justificación

Las personas con discapacidad visual tienen dificultades para desplazarse de forma

independiente y segura, lo que afecta su calidad de vida. La falta de información sobre su entorno, como la ubicación de los servicios esenciales, es un obstáculo significativo.

Las tecnologías asistenciales, como los sistemas de navegación por voz, las aplicaciones especializadas y los dispositivos portátiles emergentes, pueden ayudar a las personas con discapacidad visual a tomar decisiones informadas sobre sus desplazamientos.

El Gobierno Federal de México ha reconocido la necesidad de abordar específicamente la movilidad de personas con discapacidad visual. En septiembre de 2020, presentó un plan de movilidad que incluye el desarrollo de soluciones tecnológicas para facilitar la orientación y la toma de decisiones para este segmento de la población.

1.6 Organización del documento

En el capítulo 1 se presenta la introducción, el planteamiento del problema y su propuesta de solución, la justificación y el objetivo general y los objetivos particulares.

En el capítulo 2 se exponen los objetivos de investigación, así como el alcance del proyecto.

En el capítulo 3 se indica el marco conceptual con referencia a los sistemas de generación de rutas automáticas portátiles, desde la selección del sistema embebido hasta el diseño de algoritmos computacionales usando técnicas de inteligencia artificial.

En el capítulo 4 se puede encontrar el diseño completo del sistema propuesto.

En el capítulo 5, el autor presenta las pruebas realizadas al sistema desarrollado, así como los resultados obtenidos.

En el capítulo 6 se presentan las conclusiones y el trabajo futuro, así como el trabajo publicado en congreso nacional.

CAPÍTULO 2

Objetivos de la investigación

2.1 Objetivo general

Diseñar y construir un sistema electrónico portátil que permita a las personas invidentes, a través de información audible del entorno en donde se encuentra, tomar decisiones referentes a la ruta a transitar dentro del plantel SLT de la UACM.

2.2 Objetivos particulares

- Implementar un algoritmo de reconocimiento de código QR en una escena real, así como algoritmo de lectura e interpretación del mismo.
- Desarrollar la base de datos de audio y mapa de ruta del plantel SLT-UACM.
- Programar y ajustar los algoritmos de inteligencia artificial para generación automática de rutas peatonales.
- Integrar el sistema en un sistema embebido portátil y puesta en marcha.

CAPÍTULO 3

MARCO CONCEPTUAL

3.1 Tarjetas de desarrollo SoC.

Una tarjeta de desarrollo [7] es una herramienta que permite al usuario realizar diseños de prototipos tecnológicos. Estas tarjetas cuentan con una unidad principal de procesamiento de información y con diversos elementos integrados dentro del mismo dispositivo, lo que facilita la creación de diferentes proyectos de desarrollo e investigación.

Dentro de los componentes principales de estas placas se encuentran la memoria, el bus de comunicación, los puertos de entrada y salida como USB y Ethernet, además de pines de entrada y salida de propósito general, ya sean digitales o analógicos. En algunos casos incluyen comunicación Wi-Fi, y en modelos más específicos presentan características adicionales que permiten su integración en sistemas IoT e Industria 4.0.

En los últimos años se ha observado un crecimiento en la cantidad de tarjetas de desarrollo disponibles en el mercado. Cada una de ellas ofrece distintas cualidades y características. Una búsqueda rápida en cualquier tienda minorista basta para encontrar un amplio número de opciones. Por este motivo, resulta complicado recopilar información de todas ellas, sin embargo, podemos enfocarnos en algunas de las placas más utilizadas dentro del territorio mexicano [28].

Una tarjeta de desarrollo “Sistema en un chip” (SoC por sus siglas en inglés) tiene la virtud de integrar todos los componentes necesarios para el funcionamiento de una computadora. Entre estos componentes podemos encontrar una unidad de procesamiento central (CPU).

En general, una SoC combina elementos como tarjetas gráficas, módems de red, memorias de diversas tecnologías, interfaces de comunicación de entrada/salida, entre otros [8].

De manera similar, no se ha abordado la protección de los datos/código de una aplicación frente a otra potencialmente maliciosa. El hardware debe soportar la seguridad contra ataques de software, considerando todos los niveles de la pila de software, desde el sistema operativo hasta el software de aplicación. Estos ataques se pueden montar a través de vulnerabilidades funcionales o de canal lateral.

Dado que los dispositivos informáticos se emplean para varias actividades altamente

personalizadas (por ejemplo, compras, operaciones bancarias, seguimiento del estado físico y proporcionar indicaciones para llegar en automóvil), estos dispositivos tienen acceso a una gran cantidad de información personal confidencial, que debe protegerse del acceso no autorizado o malicioso. Además de la información personalizada del usuario final, la mayoría de los sistemas informáticos modernos (Ver Fig. 1.) contienen garantías altamente confidenciales de la arquitectura, el diseño y la fabricación, como claves criptográficas y de gestión de derechos digitales (DRM), fusibles programables, instrumentación de depuración en el chip y bits de desactivación.

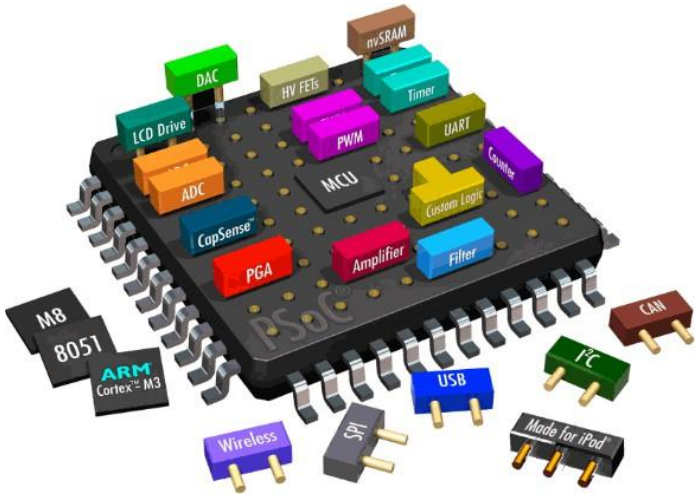


Fig. 1. Diferentes módulos que componen un SoC moderno.
Tomado de Lección 1: ¿Qué es un PSoC?, por Faurbano, 2013, SisEmbebidos.
https://sisembebidos.wordpress.com/2013/10/18/leccion-1-que-es-un-psoc/

Es crucial para nuestro bienestar que los datos contenidos en estos dispositivos estén protegidos contra el acceso no autorizado y una posible corrupción. Por lo tanto, la arquitectura de seguridad, es decir, un mecanismo para garantizar la protección de activos sensibles contra accesos maliciosos y no autorizados, constituye un componente crucial de los diseños de SoC modernos. En la figura 2 podemos observar la arquitectura de un SoC moderno, el cual consiste en múltiples bloques de IP conectado de forma interna desde su fabricación.

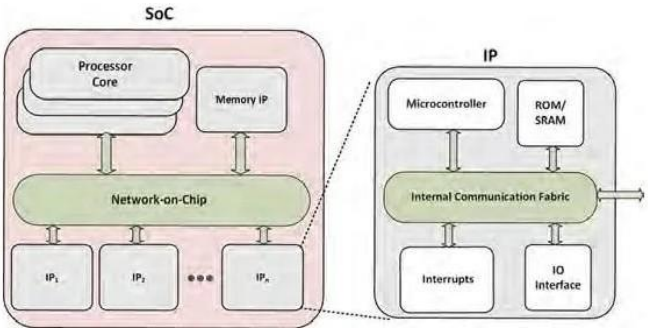


Fig. 2. Arquitectura moderna de un SoC.
Tomado de System-On-Chip (SoC) — Security design and modelling, por V. Ghodke, 2021, Medium. https://medium.com/security-risks-in-systems-on-chip-socs/system-on-chip-soc-security-design-and-modelling-306bbc536324

3.1.1 Ventajas y desventajas de las tarjetas SoC

En cuanto a las ventajas de los SoC [9] podemos encontrar el ahorro en espacio, ya que en el SoC se pueden encontrar diversos elementos incluidos en la propia tarjeta, el espacio físico que ocupa el desarrollo es menor. De la misma manera podemos mencionar el ahorro energético, una sola fuente de alimentación puede realizar la tarea de energizar toda la tarjeta. Dado que todos los sistemas que componen la SoC se encuentran incluidos en el diseño de la misma, es necesario un sólo cableado interno, minimizando la interferencia de señales dentro de la tarjeta.

Una de las desventajas mas evidentes de las tarjetas SoC es que los diseños que las contienen suelen ser muy poco flexibles, los chips que las componen no pueden expandir sus atributos y están diseñados para ser usados en tareas muy específicas. Dado lo anterior, si una falla se presenta en la tarjeta, usualmente es necesario reemplazarla en su totalidad y ello conlleva a gastos de reparación elevados.

3.1.2 Ejemplos de sistemas SoC

Uno de los ejemplos mas representativos dentro de las tarjetas SoC son los que presentan el chip ESP32. Las tarjeta de desarrollo de la marca ExpressifTM basada en el chip ESP32 tiene un núcleo de 32 bits con una arquitectura basada en RISC-V. El núcleo tiene 4 etapas el cual tiene un modulo de debug, controlador de interrupciones, interrupciones locales e interfaces de bus de sistema para acceso a periféricos y memoria.
Para el caso de la tarjeta ESP32 Dev Kit V4.0, es un microcontrolador de bajo costo y consumo de energía, cuenta con tecnología Wi-Fi y Bluetooth de modo dual integrada que permite controlar todo tipo de sensores, módulos y actuadores.

Es un módulo que tiene WIFI+BT+BLE para aplicaciones con sensores de baja potencia hasta tareas exigentes, como codificación de voz, transmisión de música y decodificación MP3. Contiene además una corriente de reposo 5uA, velocidad de hasta 150 Mbps y potencia de salida 20dBm. Cuenta con 30 pines los cuales tienen distintas funciones, distintos usos y demás.

En la figura 3 podemos encontrar el diagrama a bloques de esta tarjeta.

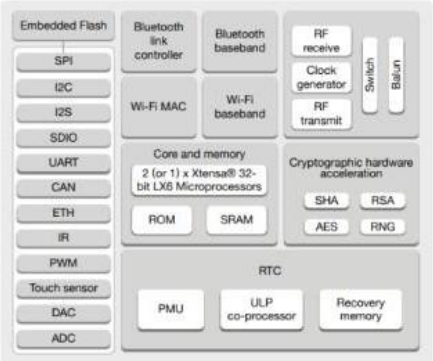


Fig. 3. Diagrama a bloques de la tarjeta DevKit V4.0.
Tomado de ESP32 Technical Reference Manual & Datasheet (p. 45), por Espressif Systems, 2019, Espressif Systems. https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf

ESP-IDF, el entorno de desarrollo de software desarrollado por EspressifTM, puede ayudar a los usuarios a desarrollar rápidamente aplicaciones de Internet de las cosas (IoT) para satisfacer las necesidades de los usuarios de Wi-Fi, Bluetooth y bajo consumo de energía [10]. Otra opción es Arduino IDETM que es un entorno de desarrollo y en él se realiza la programación de cada una de las placas de Arduino. Es el IDE más usado por los programadores de tarjetas Arduino y demás, para desarrollo de programas de Internet de las Cosas, como de Automatización, etc.

Fig 4. Tarjeta de desarrollo DEVKITV1 ESP 32.
Tomado de ESP32 38 pines Bluetooth y Wi-Fi, por Talos Electronics, (s. f.),
Talos Electronics. https://www.taloselectronics.com/products/esp32-38-pines-bluetooth-y-wifi

Otra tarjeta representativa es la Intel QuarkTM, contiene microcontrolador de 32 bits desarrollados por Intel, diseñado para tamaño pequeño y bajo consumo de energía, y dirigido a nuevos mercados incluyendo dispositivos vestibles [11]. La línea se presentó en Intel Developer Forum en 2013. Los procesadores Quark, aunque más lentos que los procesadores Atom, son mucho más pequeños y consumen me-nos energía. No son compatibles con los conjuntos de instrucciones SIMD (como MMX y SSE), y solo admite sistemas operativos embebidos. En enero de 2015, Intel anunció el módulo Intel Curie subminiatura para aplicaciones portátiles, basado en un núcleo Quark SE con 80 kB de SRAM y 384 kB flash. Tiene el tamaño de un botón. También incluye un acelerómetro de seis ejes, un concentrador de sensores DSP, una unidad Bluetooth de baja energía y un controlador de carga de batería.

La figura 5 muestra una tarjeta QUARK D2000 usada para aplicaciones de desarrollo de sistemas vestibles.

Fig. 5. Tarjeta de desarrollo Intel Quark D2000.
Tomado de The Intel® Quark™ microcontroller: Why an x86 MCU is the right stuff for the Internet of Things, por B. Giovino, (s. f.), Mouser Electronics.
https://www.mouser.mx/applications/Intel-Quark-Internet-of-Things-MCU/

Una de las tarjetas SoC más populares es la PiTM. Raspberry Pi [12] es un ordenador del tamaño de una tarjeta de crédito que se conecta a un monitor y un teclado. Esta placa soporta los componentes necesarios para funcionar de manera similar a un ordenador convencional, pero con un consumo energético y un espacio mínimos. El modelo B de cuatro núcleos de la Pi 4 B es más rápido y potente que el modelo B+ de la Raspberry Pi 3. Para quienes buscan datos de rendimiento, la CPU de la Pi 4 ofrece entre dos y tres veces el desempeño del procesador de la Pi 3 en algunas pruebas.

A diferencia de su versión anterior, esta placa puede reproducir video en 4K a 60 fotogramas por segundo, lo cual mejora su funcionalidad como centro multimedia. Sin embargo, esto no garantiza que todo tipo de video se reproduzca sin problemas, ya que el soporte para la aceleración de video con codificación H.265 sigue en desarrollo para los distintos sistemas operativos de la marca. Por ello, esta función es más una característica a futuro que una capacidad plenamente disponible. La Raspberry Pi 4-B incluye conectividad inalámbrica desde el inicio, con Wi-Fi y Bluetooth incorporados. Aunque Linux y Windows son los sistemas operativos recomendados, también existe una gran variedad de sistemas alternativos que funcionan en la Raspberry Pi.

Especificaciones de la Raspberry Pi 4 modelo B

- **Sistema en un chip:** Broadcom BCM2711
- **CPU:** Procesador de cuatro núcleos a 1,5 GHz con brazo Cortex-A72
- **GPU:** VideoCore VI
- **Memoria:** 1/2/4GB LPDDR4 RAM
- **Conectividad:** 802.11ac Wi-Fi / Bluetooth 5.0, Gigabit Ethernet
- **Vídeo y sonido:** 2 x puertos micro-HDMI que admiten pantallas de 4K@60Hz a través de HDMI 2.0, puerto de pantalla MIPI DSI, puerto de cámara MIPI CSI, salida estéreo de 4 polos y puerto de vídeo compuesto.
- **Puertos:** 2 x USB 3.0, 2 x USB 2.0
- **Alimentación:** 5V/3A vía USB-C, 5V vía cabezal GPIO
- **Expansión:** Cabezal GPIO de 40 pines

Figura 6. Especificaciones de la tarjeta Rasberry Pi4 B.

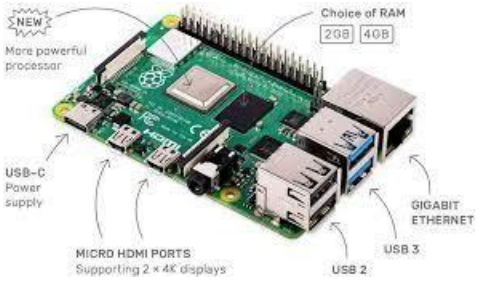


Fig. 7 Imagen de Raspberry Pi 4 modelo B.
Tomado de Raspberry Pi 4 Model B, por Raspberry Pi, (s. f.), Raspberry Pi.

Otra tarjeta SoC muy conocida es ArduinoTM.. Esta tarjeta puede recibir entradas provenientes de sensores de luz, botones, sensores de huellas, además de permitir conexión con servicios como Twitter, activar motores, encender leds o publicar información en la nube [13]. Con el paso de los años Arduino ha sido el componente principal de numerosos proyectos, desde aplicaciones sencillas hasta instrumentos científicos complejos.

Arduino nace en el Instituto de Diseño Ivrea Interaction como una herramienta sencilla para la creación rápida de prototipos, especialmente dirigida a estudiantes sin experiencia en electrónica y programación. Conforme la comunidad creció, la tarjeta se adaptó a nuevas necesidades, ofreciendo desde placas sim-ples de 8 bits hasta equipos orientados a IoT, dispositivos portátiles, impresión 3D y entornos integra-dos. Existen otros microcontroladores con funciones similares, como Parallax Basic Stamp, BX-24 de Netmedia, Phidgets o Handyboard del MIT, aunque cada uno de ellos presenta procesos de programación más complejos. Arduino simplifica este proceso y ofrece ventajas para profesores, estudiantes y aficionados. La figura 8 muestra una tarjeta Arduino 101 [14] además de algunas de sus características.

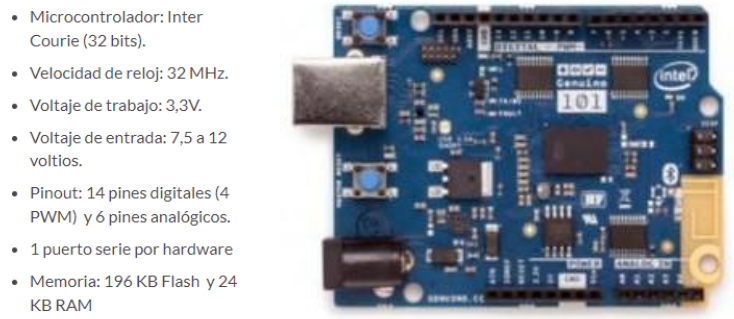


Fig. 8. Arduino 101 y sus características principales
Tomado de Empezar con Arduino: qué placa y kits de iniciación comprar, por E. R. de Luis, 2021, Xataka Makers. https://www.xataka.com/makers/empezar-arduino-que-placa-kits-iniciacion-comprar/

Beaglebone [15] es una placa de bajo consumo, pero con gran capacidad de cómputo. Desarrollada por Texas Instruments en asociación con DigiKey y Newark Element14. Esta tarjeta se diseñó para desarrollar software de código abierto y, al igual que otras placas similares, puede ser utilizada por estudiantes o desarrolladores de hardware y software libre. Actualmente se comercializa bajo la licencia sharealike de Creative Commons. Un aspecto relevante es que incorpora su propio sistema operativo, pudiendo utilizar Debian, Android o Ubuntu.

Fig. 9. Tarjeta de desarrollo Beaglebone™.
Tomado de Arduino/BeagleBone Black LabVIEW Sumobots, por BeagleBoard.org, 2016, BeagleBoard.org. https://www.beagleboard.org/projects/arduino-beaglebone-black-labview-sumobots

Algunas características de la tarjeta BeagleY-AI se presentan en la figura 10.

Feature	Description
Processor	Texas Instruments AM67A, Quad 64-bit Arm® Cortex®-A53 @1.4 GHz, multiple cores including Arm/GPU processors, DSP, and vision/deep learning accelerators
RAM	4GB LPDDR4
Wi-Fi	Beagleboard BM3301, 802.11ax Wi-Fi
Bluetooth	Bluetooth Low Energy 5.4 (BLE)
USB Ports	4 x USB 3.0 TypeA ports supporting simultaneous 5Gbps operation, 1 x USB 2.0 TypeC, supports USB 2.0 device mode
Ethernet	Gigabit Ethernet, with PoE+ support (requires separate PoE HAT)
Camera/Display	2 x 4-lane MIPI camera connector (one connector muxed with DSI capability)
Display Output	1 x HDMI display, 1 x OLDI display, 1 x DSI MIPI Display
Real-time Clock (RTC)	Supports external coin-cell battery for power failure time retention
Debug UART	1 x 3-pin debug UART
Power	5V/3A DC power via USB-C

Fig. 10. Características principales de la tarjeta BeagleY-AI.
Tomado de BeagleY-AI (p. 9), por BeagleBoard.org Foundation, 2025, Manual. https://docs.beagleboard.org/beagley-ai.pdf

3.2 Algoritmos de búsqueda de rutas

El reto con la búsqueda de una ruta [16], tendrá como protagonistas las posiciones y transiciones a lo largo de ellas. Los algoritmos de búsqueda de rutas se han utilizado en una variedad de aplicaciones, tales como rutas en redes de computadores, para llegar a de un punto a otro usando los router como nodos. En sistemas de planificación de viajes de líneas aéreas. Estos problemas son complejos de especificar. Consideremos un ejemplo simplificado de un problema de viajes de líneas aéreas que especificamos como:

- **Estados:** cada estado está representado por una localización (por ejemplo, un aeropuerto) y la hora actual.
- **Estado inicial:** especificado por el problema.
- **Función sucesor:** devuelve los estados que resultan de tomar cualquier vuelo programado (quizá más especificado por la clase de asiento y su posición) desde el aeropuerto actual a otro, que salgan a la hora actual más el tiempo de tránsito del aeropuerto.
- **Test objetivo:** ¿tenemos nuestro destino para una cierta hora especificada?

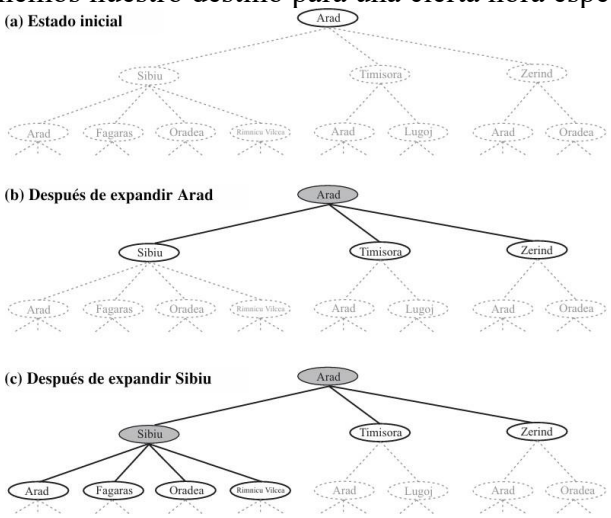


Fig.11. Árboles de búsqueda parciales para encontrar rutas desde diferentes localidades.
Tomado de Basics of artificial intelligence and machine learning [Presentación], por N. V. S. Kumar, SlideShare. https://es.slideshare.net/slideshow/basics-of-artificial-intelligence-and-machine-learning/272098367

- **Costo del camino:** esto depende del costo en dinero, tiempo de espera, tiempo del vuelo, costumbres y procedimientos de la inmigración, calidad del asiento, hora, tipo de avión, kilometraje del aviador experto, etcétera.

La figura 11 muestra una forma de representar el problema de la búsqueda de rutas haciendo referencia a diversos parámetros de diseño. Los sistemas comerciales de viajes utilizan una formulación del problema de este tipo, con muchas complicaciones adicionales para manejar las estructuras bizantinas del precio que imponen las líneas aéreas. Cualquier viajero experto sabe, sin embargo, que no todo el transporte aéreo va según lo planificado. Un sistema

realmente bueno debe incluir planes de contingencia (tales como reserva en vuelos alternativos) hasta el punto de que éstos estén justificados por el coste y la probabilidad de la falta del plan original.

3.2.1 Algoritmo de búsqueda basado en teoría de grafos

Una de las últimas tendencias en el campo de la inteligencia artificial es el *Graph Machine Learning (GML)* [17] . Esta disciplina se enfoca en aplicar algoritmos de aprendizaje automático y métodos estadísticos al estudio de redes o grafos complejos. Un grafo es una estructura de datos que está formada por nodos y enlaces, y que permite representar las relaciones entre distintos objetos. Por ejemplo, en una red social, los nodos pueden ser los usuarios y los enlaces las conexiones que existen entre ellos.

El GML utiliza los grafos para entrenar modelos de aprendizaje automático y realizar inferencias sobre los nodos o sobre los enlaces. Los problemas principales que se pueden abordar mediante GML se agrupan en tres tipos:

- Inferencia sobre nodos: predecir características de los nodos utilizando las características de sus nodos vecinos.

- Inferencia sobre enlaces: predecir características o existencia de enlaces, por ejemplo, inferir qué enlaces tiene mayor probabilidad de ocurrir en un futuro. Este es un caso típico de los sistemas de recomendación.

- Inferencia sobre conjuntos de nodos: identificar comunidades de nodos que presentan comportamientos similares. Este es un caso típico en segmentación de clientes con características similares.

Los grafos se clasifican en base a sus nodos y los enlaces. La diferencia entre los grafos depende de la relación que existe entre sus nodos.
En los grafos no dirigidos, los enlaces no tienen dirección, es decir, si un nodo A está conectado a un nodo B, entonces B también está conectado a A. Un ejemplo de grafo no dirigido es una red social donde los usuarios están conectados entre sí a través de vínculos de amistad.

En los grafos dirigidos, los enlaces tienen una dirección, es decir, si un nodo A está conectado a un nodo B, B no está necesariamente conectado a A. Un ejemplo de un grafo dirigido es una red de llamadas telefónicas donde los nodos son personas y los ejes muestran cada llamada, teniendo en cuenta quién llama a quién.

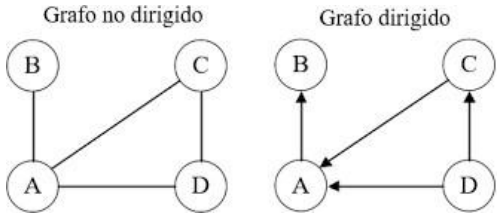


Fig. 12. Ejemplos de grafos dirigidos y no dirigidos [18].
Fuente: Elaboración propia.

En los grafos ponderados [19], los enlaces tienen un peso asociado, que representa la importancia o la intensidad de la relación entre los nodos. Un ejemplo de grafo ponderado es una red de ciudades donde los enlaces representan las rutas de transporte y el peso representa la distancia entre ciudades.

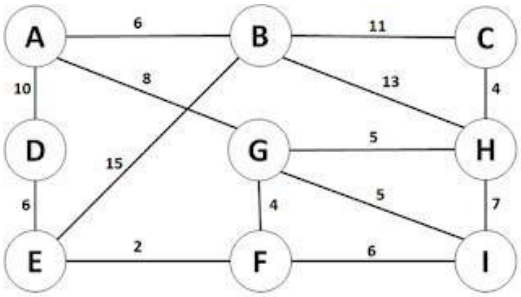


Fig. 13. Ejemplo de grafo ponderado [19].
Fuente: Elaboración propia.

En los grafos bipartitos [20], los nodos se dividen en dos conjuntos disjuntos (dos tipos de nodos), y solo se permiten enlaces entre nodos de diferentes conjuntos. Un ejemplo de grafo bipartito es una red de películas y actores, donde los nodos de un conjunto son las películas y los nodos del otro conjunto son los actores; y solo se permiten enlaces entre películas y actores.

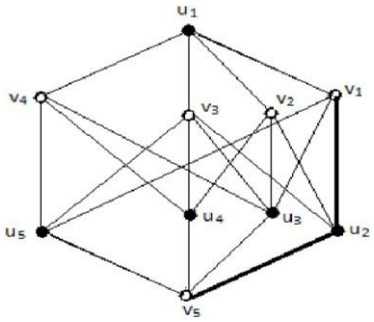


Fig. 14. Ejemplo de grafo bipartito [20].
Tomado de Ciclos hamiltonianos que pasan a través de un bosque lineal en grafos bipartitos balanceados (p. 366), por D. Brito, L. Marín & H. Ramírez, 2018, Revista de Matemática: Teoría y Aplicaciones, 25, 347–365.

Para poder analizar un grafo es necesario representarlo de forma matemática. Existen dos formas principales de hacerlo: la lista de ejes (edgelist) y la matriz de adyacencia.

La lista de ejes es simplemente una lista en la que se indican todas las conexiones. Por ejemplo, en un grafo dirigido con tres nodos {A, B y C}, donde A se conecta con B y C, la lista de ejes es {(A,B), (A,C)}. Si el grafo es no dirigido, la lista debe incluir las conexiones en ambas direcciones {(A,B), (B,A) (A,C), (C,A)}.
Otra manera de representar un grafo es por medio de la matriz de adyacencia. Esta matriz tiene dimensiones N x N, donde N corresponde al número de nodos, y muestra un valor de 1 cuando existe la conexión entre un par de nodos y 0 cuando no existe. En el caso de grafos ponderados, en lugar del valor 1 aparece el peso asociado a la conexión. Debido a sus propiedades matemáticas, la matriz de adyacencia es una de las formas más empleadas para representar grafos.

Una desventaja de este método es que, en grafos muy grandes, la memoria no va a ser optimizada.
Por este motivo, se utiliza comúnmente la matriz de adyacencia sparse, la cual solo almacena la información de las conexiones que existen, de forma que los ceros son excluidos.

Cuando el grafo es no dirigido, la matriz de adyacencia resulta simétrica, pues si existe una conexión entre los nodos A y B, también existirá la conexión inversa. De manera matemática, podemos decir que la matriz de adyacencia de un grafo dirigido con N nodos se representa con N filas y N columnas, y sus elementos se definen de la siguiente manera:

Aij=1 si existe un enlace que apunta desde el nodo j al nodo i

Aij =0 si los nodos i y j no están conectados entre sí

La matriz de adyacencia de una red no dirigida tiene dos entradas para cada enlace. El enlace (1,2) se representa como A12 y A21=1. Por lo tanto, la matriz de adyacencia de una red no dirigida es simétrica Aij=Aji.

En la siguiente figura 15 se representan los distintos tipos de grafos [17] junto con sus matrices de adyacencia. Como se puede observar, los grafos dirigidos (B) son los únicos que no tienen una matriz de adyacencia simétrica. En el caso de los grafos bipartitos (D), se pueden proyectar para obtener las conexiones indirectas entre cada tipo de nodo (E).

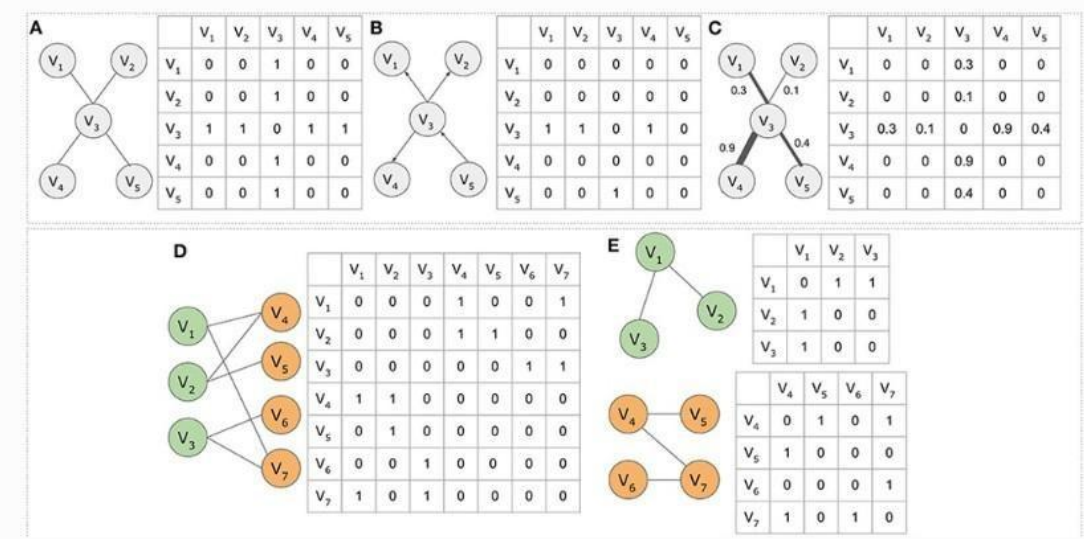


Fig. 15. Ejemplos de grafos A) grafo no dirigido, B) grafo dirigido, C) Grafo ponderado, D) grafo bipartito con dos tipos de nodos
Tomado de Métricas en grafos y redes en Python, por Cienciadedatos.net, (s. f.).
https://cienciadedatos.net/documentos/pygml04-metricas-grafos-redes-python

Cuando se trabaja con grafos, específicamente cuando se quiere hacer predicciones u otro tipo de analítica, es necesario que se tenga toda la información posible de los nodos y enlaces. Esta información se llama metainformación debido a que no pertenece estrictamente al grafo y se guarda en forma de tabla, cuyas filas son los nodos o enlaces y sus columnas son las variables disponibles de cada uno. Si se utiliza el ejemplo de la red social, la metainformación de los nodos podría ser el nombre, edad, aficiones, etc.

La distancia física desempeña un papel clave en las interacciones entre los componentes de los sistemas. Por ejemplo, la distancia física entre dos ciudades influye al número de visitantes que viajan de una ciudad a otra.

En las redes, la distancia es un concepto diferente a la distancia física. Si se plantea la pregunta: ¿Cuál es la distancia entre dos usuarios de una red social? La distancia física deja de ser relevante porque dos individuos que viven en el mismo edificio pueden no conocerse y tener muy buenos amigos en otras partes del mundo.

En las redes, la distancia física se reemplaza por la longitud del camino. Un camino entre dos nodos es una ruta que se inicia en el primer nodo y recorre los enlaces de la red pasando por distintos nodos hasta llegar al segundo nodo. La longitud del camino representa el número de enlaces que contiene el camino entre dos nodos.

En la ciencia de las redes, los caminos desempeñan un papel central. Algunas propiedades de los caminos son: Reciprocidad y Camino más corto entre nodos,

Las siguientes son algunas propiedades importantes de las redes y nodos en teoría de grafos. En artículos sucesivos se explicarán con más detalle, incluyendo su formulación e implicaciones matemáticas.

- **Grado de un nodo:** el número de enlaces que tiene un nodo. Los grafos dirigidos presentan dos tipos de grado "in degree" para conexiones entrantes y "out degree" para conexiones salientes.
- **Vecindario de un nodo:** los nodos a los que está conectado directamente un nodo.
- **Grado medio:** el promedio del grado de todos los nodos en una red.
- **Densidad:** el número de enlaces en una red dividido por el número máximo de enlaces posibles. La densidad proporciona una idea de qúan poblada de enlaces está la red.
- **Componentes conectadas:** grupos de nodos conectados entre sí. No tienen porqué existir todas la conexiones, pero sí que desde cualquier nodo puedas viajar al resto de nodos.
- **Caminos y ciclos:** un camino es una secuencia de enlaces que conectan dos nodos, mientras que un ciclo es un camino que comienza y termina en el mismo nodo.
- **Centralidad:** medida de la importancia de un nodo en una red, según su grado, vecindario, caminos y ciclos, entre otros factores.
- **Clustering:** medida de la cantidad de enlaces existentes entre los vecinos de un nodo.
- Cliqué: conjunto de nodos todos conectados con todos.
- **Sub-grafo:** la red resultante al seleccionar únicamente una serie de nodos.

CAPITULO 4

DESARROLLO DE LA PROPUESTA

4.1 Selección de elementos hardware

La base de nuestro sistema recae en la Raspberry Pi 4, equipada con 8 GB de RAM y un almacenamiento interno de 60 GB. Esta configuración se complementa con una gama de componentes integrados, incluyendo un módulo de cámara, micrófono, bocina, pantalla y una batería de polímero de litio (LiPo). La elección de esta configuración tiene como objetivo establecer una plataforma de comunicación eficaz para los usuarios, facilitando la interacción mediante comandos de voz y asegurando portabilidad.

Las Raspberry Pi son muy sencillas de utilizar, ya que básicamente son un ordenador muy pequeño. Cualquier solución que requiera hardware con sistema operativo puede hacer uso de una RPi (Raspberry Pi), y su uso está muy extendido en soluciones autónomas para domótica, ayuda en sistemas de red, e infinidad de aplicaciones más.

Las ventajas más claras de una RPi es que es un ordenador tamaño bolsillo y muy barato.

Cuenta con la posibilidad de usar el sistema linux embebido (Raspbian) en su totalidad, y puedes desarrollar dentro tal y como desarrollarías en un portátil o sobremesa tradicional.

- Sencillas de utilizar.
- Tienen sistema operativo, con todas las ventajas que esto trae consigo.
- Precio bajo.
- Tamaño muy pequeño.
- Tienen pines de entrada/salida, por lo que también se pueden conectar a otros componentes electrónicos, como sensores, LEDs, cámaras de video, micrófonos y otros componentes.

En nuestro caso, se seleccionó la cámara Modulo 2 NoIR desarrollada por la compañía Raspberry [21] . La figura 16 muestra una cámara HD NoIR de Raspberry seleccionada.

Fig. 16. Cámara Module 2 NoIR de Raspberry™ seleccionada.
Tomado de Raspberry Pi Camera V2 Noir, por TannaTechBiz, (s. f.).
https://tannatechbiz.com/raspberry-pi-camera-v2-noir.html

La cámara Module 2 Pi NoIR contiene un sensor Sony IMX219 con una resolución de 8 megapixeles. Dicha cámara contiene un filtro infrarrojo para aplicaciones con poca intensidad de luz. La cámara se conecta mediante un conector CSI contenido en la tarjeta SoC Raspberry y se puede tener acceso a ella a través de la librería libcamera basado en Picamera2 que se encuentra en una librería beta de Python.
Para el caso del micrófono y bocina usados en este proyecto, se usaron unas manos libres genéricos. La figura 17 muestra unos audífonos genéricos tipo K de dos vías.

Fig. 17. Audífonos utilizados en el prototipo.
Tomado de K-Type Walkie-Talkie auricular 1 m (3.3 ft) — Claro sonido — Profesional para walkie-talkie [Producto en línea], por Walmart México, (s. f.).
https://www.walmart.com.mx/ip/k-type-walkie-talkie-auricular-1m-3-3ft-claro-sonido-k-tipo-walkie-talkie-profesional-de-auriculares-para-walkie-talkie/00686243980615

4.2 Descripción del software utilizado

Para maximizar la eficiencia y la flexibilidad del proyecto, optamos por programar en Python [22] . Este lenguaje se destaca por su sintaxis clara, extensibilidad y una amplia variedad de bibliotecas, lo que

facilita el desarrollo y la implementación de soluciones complejas como la nuestra.

Además, la interfaz de desarrollo elegida es Visual Studio Code [23] . Esta plataforma ofrece un entorno de desarrollo integrado (IDE) altamente funcional y adaptable, permitiendo una programación más rápida y eficiente. Su extensa compatibilidad con Python y una amplia gama de extensiones hacen de Visual Studio Code una elección estratégica para el desarrollo de nuestro proyecto.

La tarea de programar el software de control para la Raspberry Pi se desarrollará, garantizando la plena compatibilidad con el entorno de desarrollo Python y Visual Studio Code, según las especificaciones previamente detalladas. La implementación de algoritmos y la lógica del sistema son cruciales para su correcto funcionamiento. Se diseñará una interfaz de usuario intuitiva y accesible, aprovechando la capacidad de la cámara para leer códigos QR y facilitando una comunicación efectiva a través del micrófono y la bocina.

En esta etapa crucial, se ha elegido emplear librerías de Python de alta calidad, tales como NetworkX, TensorFlow y NumPy. Estas herramientas fueron seleccionadas estratégicamente para recrear de manera precisa el mapa de la universidad mediante vectores y nodos, representando destinos específicos dentro del recinto académico. La elección de estas librerías no solo facilitará la creación de rutas eficientes, sino también permitirá el aprendizaje de las relaciones entre nodos y sus distancias. Este enfoque integral contribuirá significativamente a una toma de decisiones precisa y eficaz por parte del sistema.

El desarrollo de todo el algoritmo computacional se basa en el lenguaje de programación Python. Python es un lenguaje de programación ampliamente utilizado en las aplicaciones web, el desarrollo de software, la ciencia de datos y el machine learning (ML). Los desarrolladores utilizan Python porque es eficiente y fácil de aprender, además de que se puede ejecutar en muchas plataformas diferentes. El software Python se puede descargar gratis, se integra bien a todos los tipos de sistemas y aumenta la velocidad del desarrollo. Los beneficios de Python [24] incluyen los siguientes:

- Los desarrolladores pueden leer y comprender fácilmente los programas de Python debido a su sintaxis básica similar a la del inglés.
- Permite que los desarrolladores sean más productivos, ya que pueden escribir un programa de Python con menos líneas de código en comparación con muchos otros lenguajes.
- Python cuenta con una gran biblioteca estándar que contiene códigos reutilizables para casi cualquier tarea. De esta manera, los desarrolladores no tienen que escribir el código desde cero.
- Los desarrolladores pueden utilizar Python fácilmente con otros lenguajes de programación conocidos, como Java, C y C++.
- La comunidad activa de Python incluye millones de desarrolladores alrededor del mundo que prestan su apoyo. Si se presenta un problema, puede obtener soporte rápido de la comunidad.
- Hay muchos recursos útiles disponibles en Internet si desea aprender Python. Por ejemplo, puede encontrar con facilidad videos, tutoriales, documentación y guías para desarrolladores.
- Python se puede trasladar a través de diferentes sistemas operativos de computadora,

como Windows, macOS, Linux y Unix.

NetworkX [17] es una de las librerías de Python más utilizadas para trabajar con grafos y redes. NetworkX permite crear, manipular y analizar grafos de manera eficiente.
Una de las principales ventajas de NetworkX es su capacidad para trabajar con grafos de gran tamaño y complejidad, permitiendo manejar grafos con millones de nodos y enlaces. La librería cuenta con una gran variedad de funcionalidades que permiten crear, importar y exportar grafos en múltiples formatos, así como analizar las propiedades de estas redes (grado medio, la densidad, el coeficiente de clustering, el camino más corto entre dos nodos, y muchas otras más). Además, cuenta con una serie de algoritmos para buscar patrones, como la detección de comunidades, detección de centralidad y detección de componentes conectados.

Tensor Flow [25] se trata de una librería de código libre para Machine Learning (ML). Fue desarrollado por Google para satisfacer las necesidades a partir de redes neuronales artificiales. TensorFlow te permite construir y entrenar redes neuronales para detectar patrones y razonamientos usados por los humanos.

Además de trabajar con redes neuronales, TensorFlow es multiplataforma. Trabaja con GPUs y CPUs e incluso con las unidades de procesamiento de tensores (TPUs). TensorFlow es una plataforma que facilita la creación e implementación de modelos de aprendizaje automático. Lo que buscamos hoy en día es automatizar cientos de procesos y TensorFlow nos permite llegar a esta automatización. Nos permite crear y entrenar modelos de Aprendizaje Automático con facilidad mediante APIs intuitivas

NumPy [26] es una librería de Python especializada en el cálculo numérico y el análisis de datos, especialmente para un gran volumen de datos. Incorpora una nueva clase de objetos llamados arrays que permite representar colecciones de datos de un mismo tipo en varias dimensiones, y funciones muy eficientes para su manipulación. La ventaja de Numpy frente a las listas predefinidas en Python es que el procesamiento de los arrays se realiza mucho más rápido (hasta 50 veces más) que las listas, lo cual la hace ideal para el procesamiento de vectores y matrices de grandes dimensiones.

4.3 Diseño de algoritmos computacionales

En esta parte de la tesis se presenta el diseño de algoritmo computacional de cada etapa que compone el desarrollo. En la figura 18 podemos observar un diagrama a bloques del proceso que fue desarrollado con software especializado.

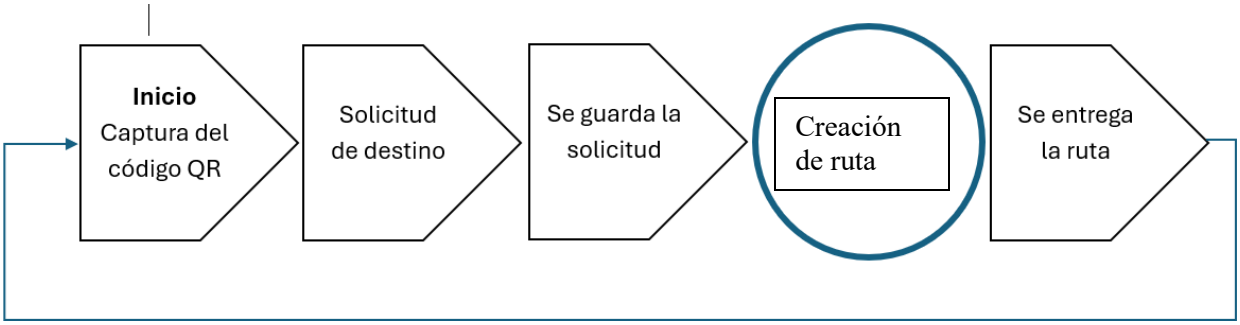


Fig. 18. Proceso por etapas que componen el software desarrollado.
Fuente: Elaboración propia.

1.- INICIO: El programa activa la cámara y captura el código QR.
2.- SOLICITUD DE DESTINO: El programa solicita al usuario, de forma oral, el destino al que desea llegar.
3.- SE GUARDA LA SOLICITUD: La información oral proporcionada por el usuario se procesa y se almacena en el sistema.
4.- CREACION DE RUTA: El programa crea una ruta óptima desde el punto de origen hasta el destino indicado.
5.- SE ENTRAGA LA RUTA: Las instrucciones de la ruta se entregan al usuario de forma oral.

El proyecto fue planeado en dos etapas con fines prácticos, la primera etapa consistió en la creación de la interfaz de usuario, pues era el primer reto técnico el lograr que nuestro sistema se pudiera comunicar de forma eficiente con el usuario, la forma en la que se decidió atacar el problema para los usuarios que no podrían interactuar con una pantalla fue con comandos de voz pues es instrucciones de voz. La figura 19 muestra el diagrama a bloques de la interfaz de usuario desarrollada.

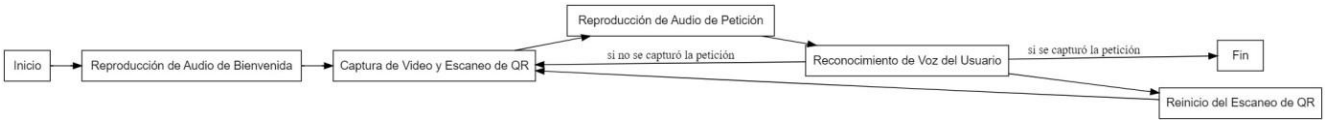


Fig. 19. Interfaz de usuario desarrollada.
Fuente: Elaboración propia.

La clave para llevar el proyecto a la realidad es dividir las tareas e ir realizándolas de forma independiente y una vez probadas, se unieron para que funcionaran como una sola.
Una vez que conocemos el diagrama general el reto era construir toda una interfaz que sea compatible con el sistema Raspberry Pi 4TM y PythonTM que usaríamos. El primer paso fue construir un código capaz de reconocer códigos QR, por medio de una cámara.

4.3.1 Algoritmo de generación y escaneo de código QR.

Se utilizaron las bibliotecas OpenCV (`cv2`) y PyZbar (`pyzbar`) para realizar la detección y decodificación de códigos QR desde una transmisión de video en tiempo real proveniente de una cámara. La figura 10 muestra el diagrama a bloques del proceso.

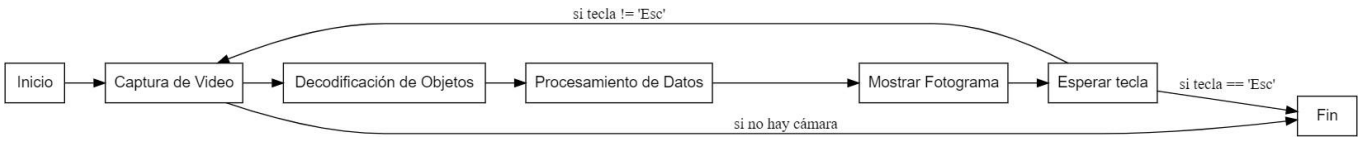


Figura 20. Diagrama a bloques del proceso de captura y reconocimiento de QR.
Fuente: Elaboración propia.

En la inicialización, se configura la captura de video desde la cámara predeterminada del dispositivo. El bucle principal se ejecuta continuamente, capturando cada cuadro de video.

Durante la ejecución del bucle, se detectan y decodifican los códigos QR presentes en el cuadro actual, y la información asociada a cada código encontrado se imprime en la consola. Además, el video en tiempo real se muestra en una ventana llamada "Frame". La ejecución del programa continúa hasta que se presiona la tecla Esc, momento en el cual se liberan los recursos asociados con la cámara y se cierran las ventanas de visualización, de esta forma se cubre la necesidad de capturar el código QR y así poder identificar en que posición se encuentra el usuarios que mas adelante tomara mas importancia. La figura 21 muestra un fragmento del programa desarrollado en Python.

```
import cv2
from pyzbar import pyzbar

cap = cv2.VideoCapture(0)

while True:
    _, frame = cap.read()

    decodedObjects = pyzbar.decode(frame)
    for obj in decodedObjects:
        print("Data:", obj.data)

    cv2.imshow("Frame", frame)
```

Fig. 21. Segmento de código desarrollado para detección de QR en escenas reales.
Fuente: Elaboración propia.

4.3.2 Algoritmo de reconocimiento de audio a texto

Una parte fundamental del proceso además de las anteriores es lograr que el sistema entienda las instrucciones que le proporcionara el usuario, se buscaron diversas soluciones a este problema la solución óptima fue crear un programa que primero lograra recibir la voz, lo que sigue es fundamental pues el problema no se resuelve el recibir la información del destino si no lo que debe hacer con esa información que será mejor explicado en los capítulos siguientes, pero básicamente se puede decir que se debe comparar con un a lista de lugares para ello es necesario convertir las señales de entrada de voz en datos. La figura 22 muestra el diagrama a bloques del proceso descrito.

Fig. 22. Diagrama a bloques del proceso de conversión de audio a texto.
Fuente: Elaboración propia.

El código Python utiliza las bibliotecas `speech_recognition`, `pyaudio`, y `wave` para capturar audio del micrófono, guardarlo en un archivo temporal, transcribirlo en texto mediante Google Speech Recognition y mostrar el resultado. Aquí está el flujo detallado del código:

1. Importa las bibliotecas necesarias: `speech_recognition` para el reconocimiento de voz,`pyaudio` para acceder al micrófono, y `wave` para trabajar con archivos de audio.

2. Crea un objeto `Recognizer` de la clase `Recognizer` de la biblioteca `speech_recognition`, un objeto `PyAudio` para gestionar la entrada de audio del micrófono, y abre un flujo de audio para capturar las entradas del micrófono.

3. Imprime un mensaje para indicar al usuario que hable algo.

4. Captura y guarda el audio del micrófono en un archivo temporal durante aproximadamente 15 segundos, dividido en tramas de 1024 bytes.

5. Detiene y cierra el flujo de audio del micrófono, y finaliza el objeto `PyAudio`.

6. Abre el archivo temporal de audio y lo convierte en un objeto `AudioFile` de la biblioteca
7. `speech_recognition`.

8. Utiliza el método `recognize_google()` para transcribir el audio grabado en texto, utilizando el servicio de reconocimiento de voz de Google. Se especifica el idioma español (es-ES) como parámetro.

9. Imprime el texto transcrito obtenido del reconocimiento de voz.

10. Maneja las excepciones `UnknownValueError` y `RequestError` que pueden ocurrir si el reconocimiento de voz no puede interpretar el audio o si hay un error en la solicitud al servicio de Google Speech Recognition.

11. Finalmente, cierra el archivo temporal de audio.

En resumen, este código captura audio del micrófono, lo guarda en un archivo temporal, lo transcribe en texto usando Google Speech Recognition y muestra el texto transcrito en la consola. Es útil para aplicaciones de reconocimiento de voz y procesamiento de audio. La figura 13 muestra el algoritmo desarrollado en Python.

```
from gtts import gTTS
import pygame

# Texto que deseas convertir en audio
texto = "¡Bienvenido!, dime a donde quieres ir dentro de la universidad"

# Crear un objeto gTTS
tts = gTTS(text=texto, lang='es')

# Guardar el archivo de audio
archivo_audio = "audio_salida.mp3"
tts.save(archivo_audio)

print("Texto convertido a audio correctamente.")

# Inicializar el módulo PyGame
pygame.mixer.init()

# Cargar el archivo de audio
pygame.mixer.music.load(archivo_audio)
```

Fig. 23. Fragmento de código desarrollado para reconocimiento de audio a texto.
Fuente: Elaboración propia.

4.3.3 Algoritmo de reconocimiento de texto a audio

La necesidad de comunicación con los usuarios hizo que se pensara en usar un programa que lograra reproducir instrucciones audibles, así que fue necesario idear instrucciones claras que se le enviarían al receptor, para ello se uso un programa que convirtiera texto en audio, pues sería mas fácil construir así los mensajes para no tener que limitar el sistema a una base muy particular. La figura 24 muestra el diagrama a bloques del proceso de conversión de texto a audio.

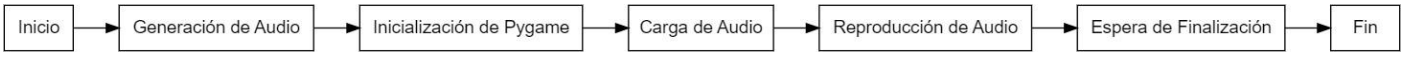


Fig. 24. Diagrama a bloques de conversión de texto a audio.
Fuente: Elaboración propia.

El código Python utiliza las bibliotecas `gtts` y `pygame` para convertir un texto en un archivo de audio y reproducirlo. Aquí está el flujo de lo que hace:

1. Se importa la clase gTTS de la biblioteca gtts, la cual permite convertir texto en voz, así como la biblioteca pygame, utilizada para la reproducción de audio.

2. Se define el mensaje que se desea convertir en voz. En este caso, consiste en un saludo y una solicitud de dirección dentro de la universidad.
3. Se crea un objeto gTTS utilizando el texto definido, especificando el idioma como español.
4. El mensaje convertido se guarda en un archivo de audio denominado "audio_salida.mp3".
5. Se inicializa el módulo pygame para manejar la reproducción de audio.
6. El archivo de audio se carga mediante el método pygame.mixer.music.load().
7. La reproducción del audio se realiza en tiempo real utilizando el método pygame.mixer.music.play().
8. Se emplea un bucle while para esperar que finalice la reproducción del audio antes de continuar con la ejecución del programa.
9. Finalmente, se imprimen mensajes en la consola indicando que el texto se ha convertido en audio y que la reproducción se ha realizado de manera exitosa.

La figura 15 muestra el código desarrollado en Python para la etapa de conversión de texto a audio.

```
import speech_recognition as sr
import pyaudio
import wave
import tempfile

r = sr.Recognizer()
p = pyaudio.PyAudio()
stream = p.open(format=pyaudio.paInt16, channels=1, rate=16000, input=True,
frames_per_buffer=1024)

print("Habla algo:")

# Captura y guarda el audio en un archivo temporal
temp_audio_file = tempfile.NamedTemporaryFile(delete=False)
temp_audio_filename = temp_audio_file.name
frames = []

try:
    for _ in range(0, int(16000 / 1024 * 15)):  # Capturar audio durante 15 segundos
        data = stream.read(1024)
        frames.append(data)
    stream.stop_stream()
    stream.close()
    p.terminate()
```

Fig.25. Fragmento de código desarrollado para convertir de texto a audio.
Fuente: Elaboración propia.

4.3.4 Generación de mapa de las localidades seleccionadas

Para la generación del mapa que servirá de insumo para el cálculo automático de rutas, se tomó como base el mapa general del Plantel San Lorenzo Tezonco de la Universidad Autónoma de la Ciudad de México y como paso siguiente se seleccionaron y numeraron los lugares más concurridos del mismo. De la misma manera, esta selección de lugares dará origen a los nodos que conformarán el grafo necesario para identificar origen y destino del usuario del sistema. La siguiente lista muestra los lugares seleccionados para ser parte del mapa.

1.Entrada

2.Estacionamiento profesores

3.Domo

4.Cubículos de profesores E1 pso 0

5.Cubículos de profesores E1 piso 1

6.Cubículos de profesores E1 piso 2

7.Cubículos de profesores D1 piso 0

8.Cubículos de profesores D1 piso 1

9.Cubículos de profesores D1 piso 2

10.Cubículos de profesores E2 piso 0

11.Cubículos de profesores E2 piso 1

12.Cubículos de profesores E2 piso 2

13.Cubículos de profesores D2 piso 0

14.Cubículos de profesores D2 piso 1

15.Cubículos de profesores D2 piso 2

16.Edificio A piso 0

17.Edificio A piso 1

18.Edificio A piso 2

19.Edificio A piso 3

20.Edificio A piso 4

21.Edificio B piso 0

22.Edificio B piso 1

23.Edificio B piso 2

24.Edificio B piso 3

25.Edificio B piso 4

26.Edificio C piso 0

27.Edificio C piso 1

28.Edificio C piso 2

29.Edificio C piso 3

30.Edificio C piso 4

31.Baño

32.Comedor

33.Explanada

34.Biblioteca

35.Camino al Ágora

36.Ágora

37.Base de camiones constitución

38.Estacionamiento alumnos

En la figura 26 se muestran las localidades seleccionadas para considerarse en el diseño del algoritmo.

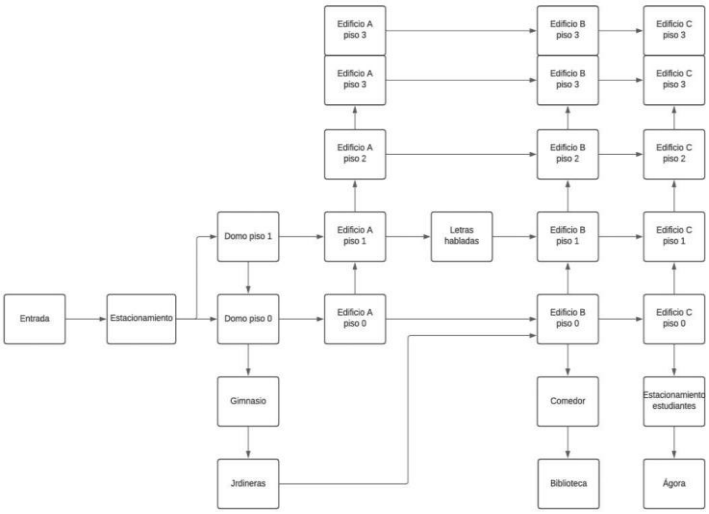


Fig. 26. Plano de lugares seleccionados en Plantel San Lorenzo Tezonco de la UACM.

Fuente: Elaboración propia.

Considerando que las localidades seleccionadas no solo están en la planta baja del plantel sino que se encuentran distribuidas en edificios los cuales llegan a tener hasta tres pisos adicionales, se realizó un grafo tridimensional para establecer los nodos de cada localidad propuesta y se presenta en la figura 27, dicho grafo resultante fue generado en Python.

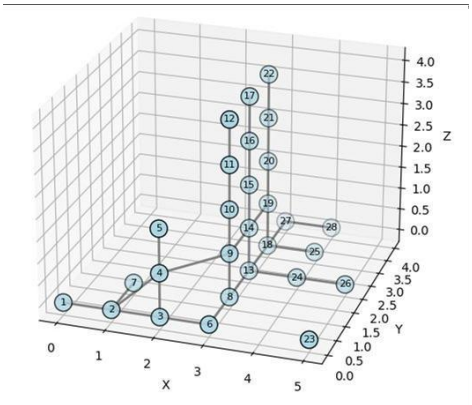


Fig. 27. Mapa tridimensional de nodos dentro del plantel San Lorenzo Tezonco de la UACM.
Fuente: elaboración propia

4.4 Algoritmo de generación de rutas automáticas

Para el algoritmo de generación de rutas automáticas se hace uso del software de NetworkX y Tensor Flow. Lo anterior para la generación del grafo no dirigido que se conformará de los nodos y ejes propuestos en la sección 4.3.4.

NetworkX permite crear redes de manera manual, añadiendo los nodos y ejes uno por uno o desde un archivo o un DataFrame que contenga las conexiones. Esto último es especialmente útil cuando se trabaja con datos de redes que ya han sido recopilados y se encuentran en un formato estructurado.
Para crear un grafo en NetworkX, se debe crear un objeto de tipo "Grafo" utilizando la función nx(Graph). Esta función crea un grafo vacío, sin nodos ni ejes, al que se pueden agregar elementos más adelante. Una vez que el objeto grafo ha sido creado, se debe poblar con nodos y conexiones. Para ello se pueden usar dos métodos. Con la función add_node se añade un único nodo al grafo ó add_nodes_from se puede añadir múltiples nodos al grafo. Para añadir ejes entre nodos se usa la función add_edge y si los nodos no existen,el software los crea y los añade automáticamente al grafo. El nombre de los nodos puede ser tanto numérico como de caracteres.

Para obtener la distancia entre ejes, de orden 2 entre cualquier par de nodos, basta con multiplicar la matriz de adyacencia por sí misma. Si la multiplicamos n veces, obtenemos la matriz de distancias de orden n. Para incluir un grafo ponderado, los ejes del grafo tienen un peso asociado, En la representación gráfica de este tipo de redes, los ejes suelen mostrar con una anchura distinta en función de su peso. Para dibujar un grafo ponderado con NetworkX, es necesario dibujar por separado nodos y ejes. En primer lugar se utiliza un utilizando un

graph layout que define la posición de los nodos. Una vez definida su posición se representan los nodos y los ejes. Para este ejemplo se utiliza el spring_layout, pero hay más.

En la mayoría de los problemas de grafos reales, se dispone de información adicional de nodos y ejes. Los atributos de los nodos se añaden con el método networkx.set_node_atributes(Grafo, diccionario, nombre) y los atributos de los ejes se añaden con el método network.set_edge_atributes().
Para acceder a los atributos de nodos y ejes (meta-información) se utiliza G.nodes(data=True) o G.edges(data=True). Estos comandos devuelven una estructura en forma de diccionario donde la clave es el nombre del nodo y el valor contiene todos los atributos.
Una vez que tenemos creado el mapa y que se ha comprobado que es correcto visualizando el gráfico, es momento de ejecutar un plan para que el sistema logre recorrer de forma eficiente el mapa creando una ruta desde el nodo de inicio, que será dado por el usuario, hasta el nodo final al que el usuario desee llegar. Para ello, es crucial que implementemos algoritmos y modelos que permitan calcular rutas óptimas y generar instrucciones precisas para la navegación. Para llevar a cabo esta tarea, utilizamos varias biblio-tecas de Python que nos ayudan a manejar y analizar gráficos, así como a implementar modelos de aprendizaje automático para predecir distancias entre nodos. Las bibliotecas principales que empleamos son 'NetworkX', 'TensorFlow', 'Keras', y 'NumPy'. 'NetworkX' es una biblioteca de Python para la creación, manipulación y estudio de la estructura, dinámica y funciones de complejas redes de grafos. En este código, 'NetworkX' nos permite modelar la universidad como un grafo dirigido donde los no-dos representan puntos importantes (edificios, estacionamientos, etc.) y las aristas representan las posibles conexiones entre ellos: ''python import networkx as nx''.

1. Crear el grafo dirigido: Utilizamos 'nx.DiGraph()' para crear un grafo dirigido, donde la dirección de las aristas importa, reflejando las posibles direcciones en que uno puede moverse entre nodos.
2. Definir los nodos: Los nodos son definidos con sus posiciones en el espacio 3D para una representación espacial precisa.
3. Agregar nodos y conexiones: Utilizamos 'G.add_nodes_from(pos.keys())' y 'G.add_edges_from(edges)' para agregar los nodos y las conexiones entre ellos.

TensorFlow es una biblioteca de código abierto para el aprendizaje automático y la inteligencia artificial. Keras es una API de alto nivel para construir y entrenar modelos de aprendizaje profundo, que se ejecuta sobre TensorFlow. 'python import tensorflow as tf from tensorflow import keras '

1. Preparar los datos de entrenamiento: Los datos de entrenamiento consisten en pares de nodos y sus distancias, que se utilizan para entrenar un modelo de red neuronal que puede predecir la distancia entre cualquier par de nodos.

2. Crear el modelo: El modelo de red neuronal está compuesto por varias capas densas (Dense la-yers), que permiten aprender complejas relaciones no lineales entre las posiciones de los nodos.
3. Compilar y entrenar el modelo: Utilizamos el optimizador ‘adam’ y la función de pérdida ‘mean_squared_error’ para compilar y entrenar el modelo con los datos de entrenamiento.

NumPy es una biblioteca fundamental para la computación científica en Python. Proporciona una gran variedad de funciones matemáticas y de manipulación de matrices que son esenciales para el procesamiento de datos y el entrenamiento de modelos de aprendizaje automático. ‘python import numpy as np’:

1. Calcular distancias: Utilizamos `NumPy` para calcular la distancia euclidiana entre los puntos en 3D.
2. Preparar los datos: Los datos de entrenamiento se transforman en matrices utilizando `NumPy`, facilitando su uso en el modelo de aprendizaje automático.

Explicación del Algoritmo:

1. Definir los nodos y conexiones: Se definen los nodos y sus posiciones en un espacio tridimensional. Cada nodo representa una ubicación importante dentro de la universidad, como la entrada, edificios, estacionamientos, etc. Luego, se definen las conexiones entre estos nodos, representando caminos directos que se pueden recorrer.
2. Calcular distancias: Se implementa una función `calculate_distance` que utiliza la distancia euclidiana para determinar la distancia entre dos nodos dados sus coordenadas en 3D.
3. Preparar datos de entrenamiento: Se recopilan datos de entrenamiento que incluyen pares de nodos y las rutas más cortas entre ellos, junto con las distancias. Estos datos se utilizan para en-trenar el modelo de red neuronal que predecirá las distancias entre nodos.
4. Construir y entrenar el modelo de red neuronal: Se construye un modelo de red neuronal utilizando `Keras`, el cual se entrena con los datos recopilados para aprender a predecir las distan-cias entre nodos. Predicción de distancias y búsqueda de rutas:

Con el modelo entrenado, se pueden predecir las distancias entre cualquier par de nodos. Utilizando la función `nx.shortest_path` de `NetworkX`, se encuentra la ruta más corta entre dos nodos dados.

5. Generar instrucciones: La función `generate_instructions` toma la ruta más corta y genera instrucciones detalladas para la navegación, indicando direcciones y cambios de movimiento necesarios para llegar al destino.

El núcleo matemático de este sistema radica en la teoría de grafos y el álgebra lineal. Un grafo es una estructura compuesta por nodos (o vértices) y aristas (o enlaces) que conectan pares de nodos. En álgebra lineal, estas estructuras se pueden representar mediante matrices de adyacencia y matrices de distancia.
- Matriz de Adyacencia: Representa las conexiones entre nodos en el grafo. Si hay una arista entre los nodos `i` y `j`, entonces el elemento `[i, j]` de la matriz es 1, de lo contrario, es 0. - Matriz de Distancia: Contiene las distancias entre cada par de nodos. Esto se puede calcular utilizando la distancia euclidiana en un espacio tridimensional. El modelo de red neuronal aprende a aproximar las distancias entre no-dos, que es un problema de regresión multivariable, mientras que la teoría de grafos se utiliza para encontrar las rutas más cortas entre nodos.

CAPITULO 5

PRUEBAS REALIZADAS Y RESULTADOS OBTENIDOS

5 Pruebas realizadas al software desarrollado y resultados obtenidos

Con el software propuesto, se logra una interacción fluida y dinámica entre el sistema y el usuario, permitiendo una comunicación bidireccional a través de comandos de voz y códigos QR. Esto sienta las bases para la creación de aplicaciones inclusivas y adaptadas para personas con discapacidades visuales, permitiendo una experiencia más accesible y amigable. El diagrama de flujo del código propuesto se presenta en la figura 28.
El proceso de pruebas del sistema diseñado se basa en una secuencia bien definida de pasos, que permiten la interacción fluida entre el usuario y el sistema. Para garantizar la eficiencia y el cumplimiento de los objetivos, es fundamental validar cada componente del sistema, comenzando desde el momento en que el programa se ejecuta.
Cuando el programa inicia, se activa automáticamente un mensaje de bienvenida, que se reproduce mediante un archivo de audio. Este mensaje es crucial porque proporciona al usuario una indicación clara de que el sistema está operativo y listo para interactuar.

A través del uso de tecnología de conversión de texto a voz (TTS), el sistema es capaz de reproducir mensajes personalizados, lo cual es especialmente útil para aplicaciones destinadas a personas con discapacidades visuales.

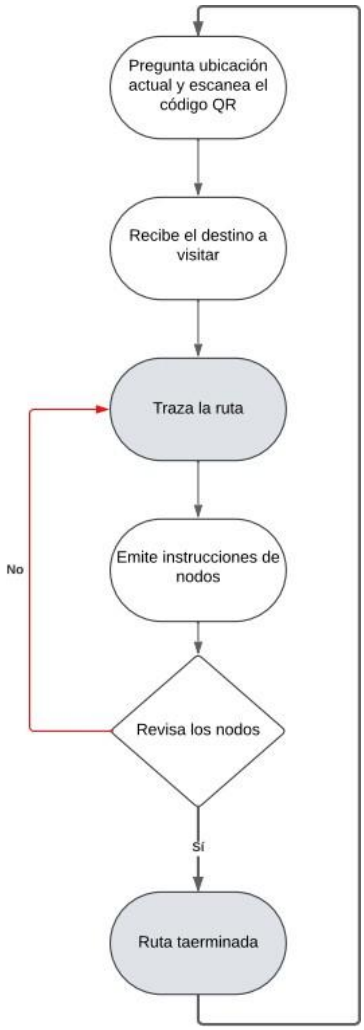


Fig. 28. Diagrama a bloques del algoritmo desarrollado.
Fuente: Elaboración propia.

Después de la reproducción del mensaje de bienvenida, el sistema inicia la cámara y comienza a escanear el entorno en busca de códigos QR. La detección de estos códigos es un aspecto clave, ya que permite determinar el nodo exacto del mapa donde se encuentra el usuario. En otras palabras, el sistema puede identificar su ubicación dentro de un espacio físico al leer y decodificar el contenido del código QR. Una vez que se encuentra y decodifica un código QR, el sistema puede correlacionar la información contenida en el código con el mapa digital, proporcionando así la posición precisa del usuario. Este proceso es fundamental para la funcionalidad del sistema, ya que es la base para la navegación y la generación de rutas personalizadas.

A continuación, el sistema interactúa con el usuario mediante comandos de voz. Aquí, el reconocimiento de voz juega un papel esencial. Se utiliza un sistema de reconocimiento de voz para permitir que el usuario emita órdenes y solicitudes verbalmente. Este paso se realiza para facilitar la interacción y evitar la necesidad de una interfaz gráfica, que podría ser menos

accesible para personas con discapacidades visuales.

Una vez que el usuario emite un comando de voz, el sistema captura la solicitud y la convierte en texto. Este texto se guarda en una variable para su uso posterior.
De este modo, el sistema puede procesar la solicitud y ejecutar las acciones correspondientes, como planificar una ruta o proporcionar información sobre la ubicación del usuario.

La figura 29 muestra el algoritmo inicializado y listo para escuchar la indicación del usuario, podemos observar en la linea final del terminal que se espera la instrucción de audio para después reconocer el código QR que dará la ubicación de inicio de todo el sistema.

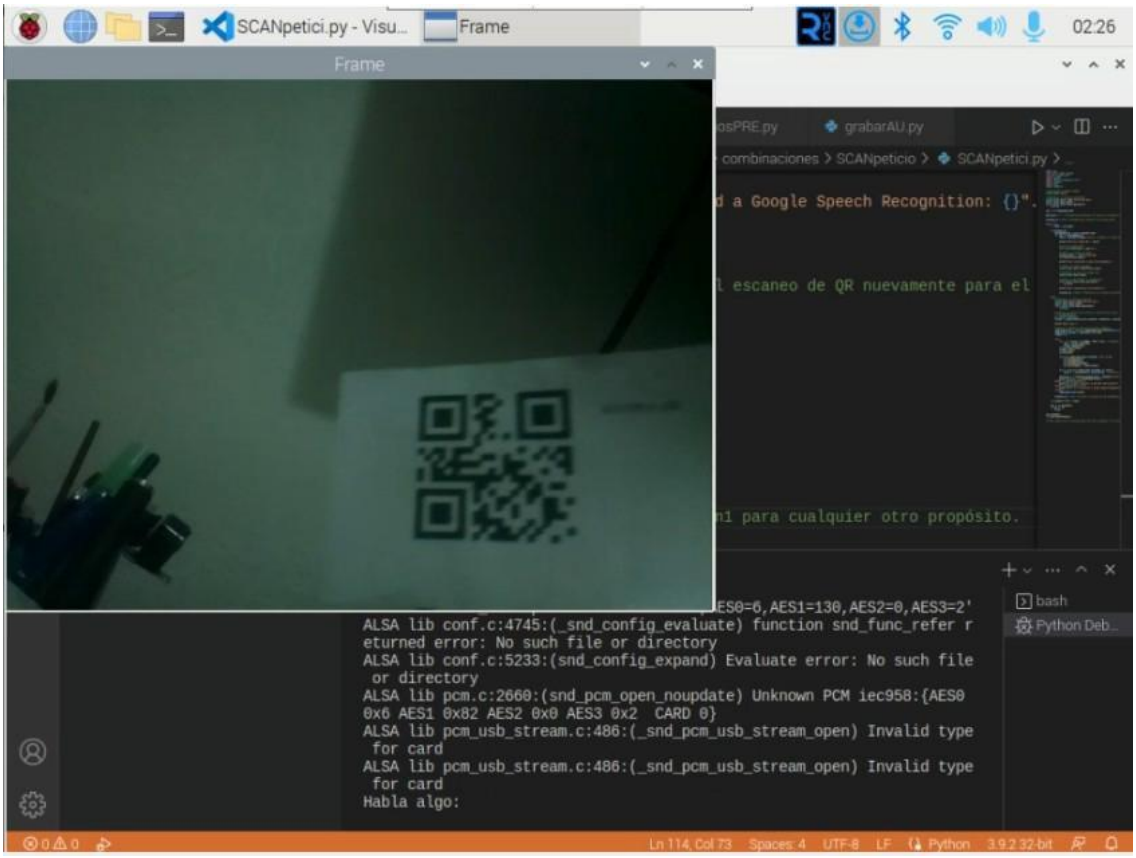


Fig. 29. Algoritmo inicializado.
Fuente: Elaboración propia.

En la figura 30 podemos observar la etapa del reconocimiento y decodificación del código QR. En la ventana del terminal tenemos el resultado de la decodificación el cual es “prueba 1”.

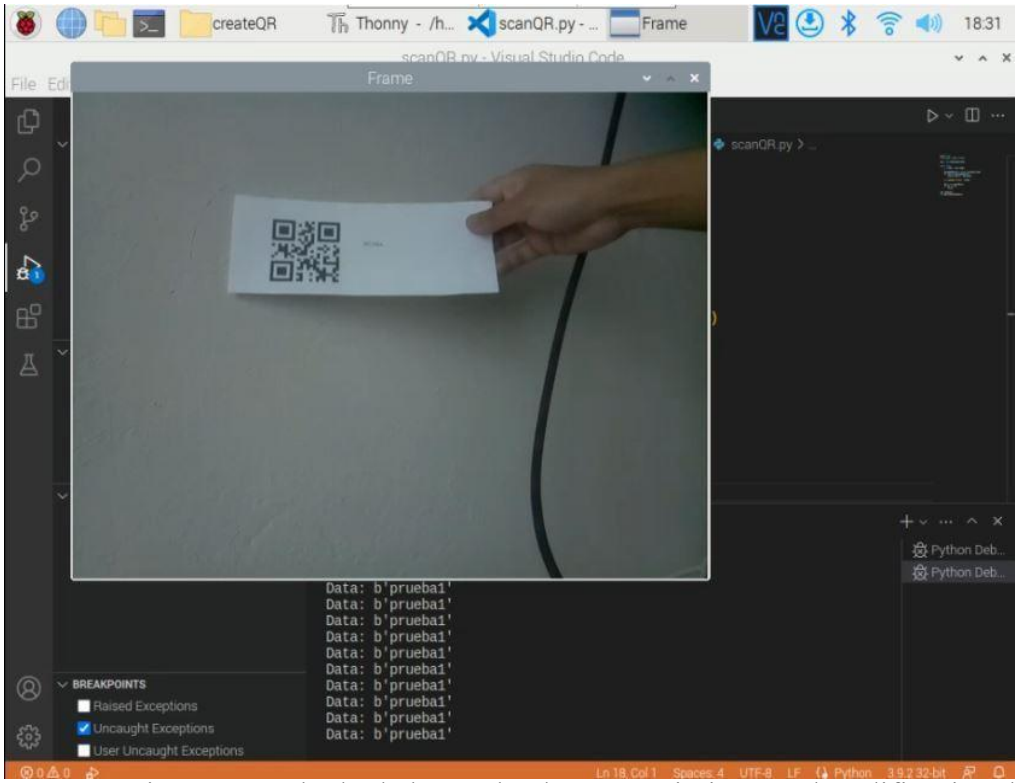


Fig. 30. Resultado de la prueba de reconocimiento y decodificación de QR.
Fuente: Elaboración propia.

En la figura 31 se muestra el resultado de la prueba realizada para la etapa de identificación de la señal de voz. Podemos observar en la ventana del terminal que el sistema arroja la palabra dicha, la cual corresponde a “comedor”. Se realizaron pruebas con todas las palabras de las ubicaciones seleccionadas resultando un reconocimiento del 100%.

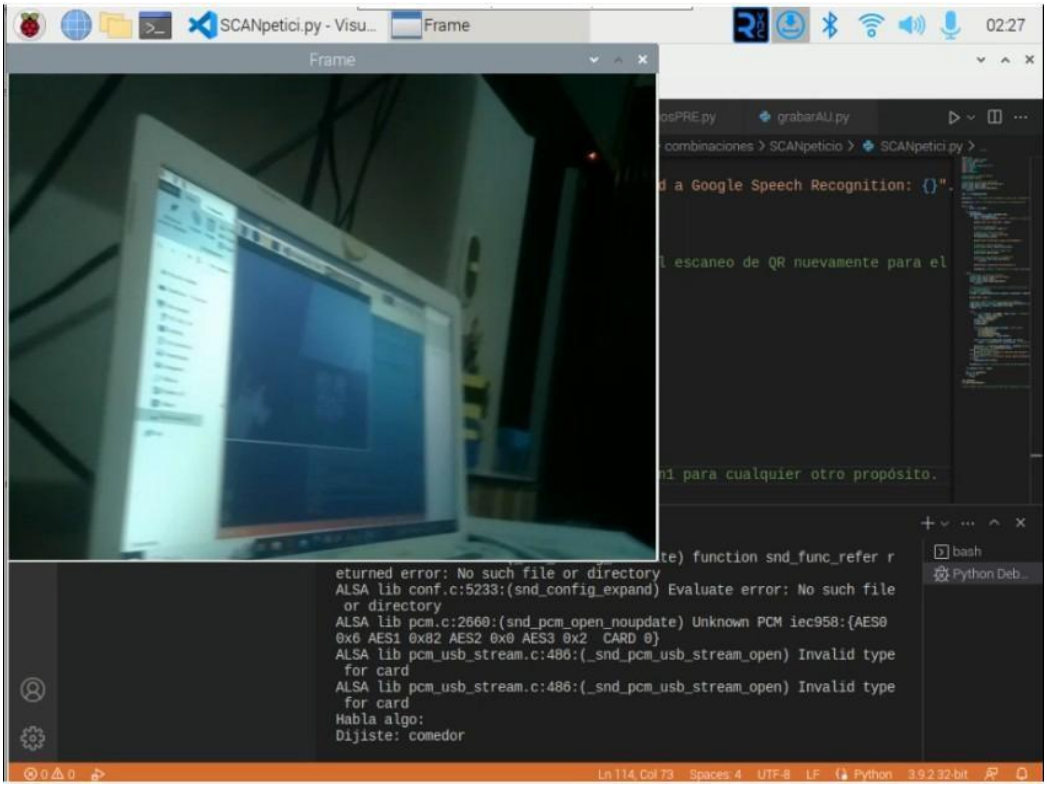


Fig. 31. Resultado de la etapa del reconocimiento de voz.
Fuente: Elaboración propia.

Continuando con el proceso, en la siguiente prueba se realiza la identificación y lectura del código QR, en el cual se ubica el salón B-201. Dicha información se transforma a audio y se reproduce en la bocina del sistema. En la pantalla mostrada en la figura 32, podemos ver que el algoritmo reconoce de manera correcta el código QR y transforma la información a audio. En la linea final del terminal podemos apreciar el mensaje de que el archivo de audio se ha reproducido de forma correcta.

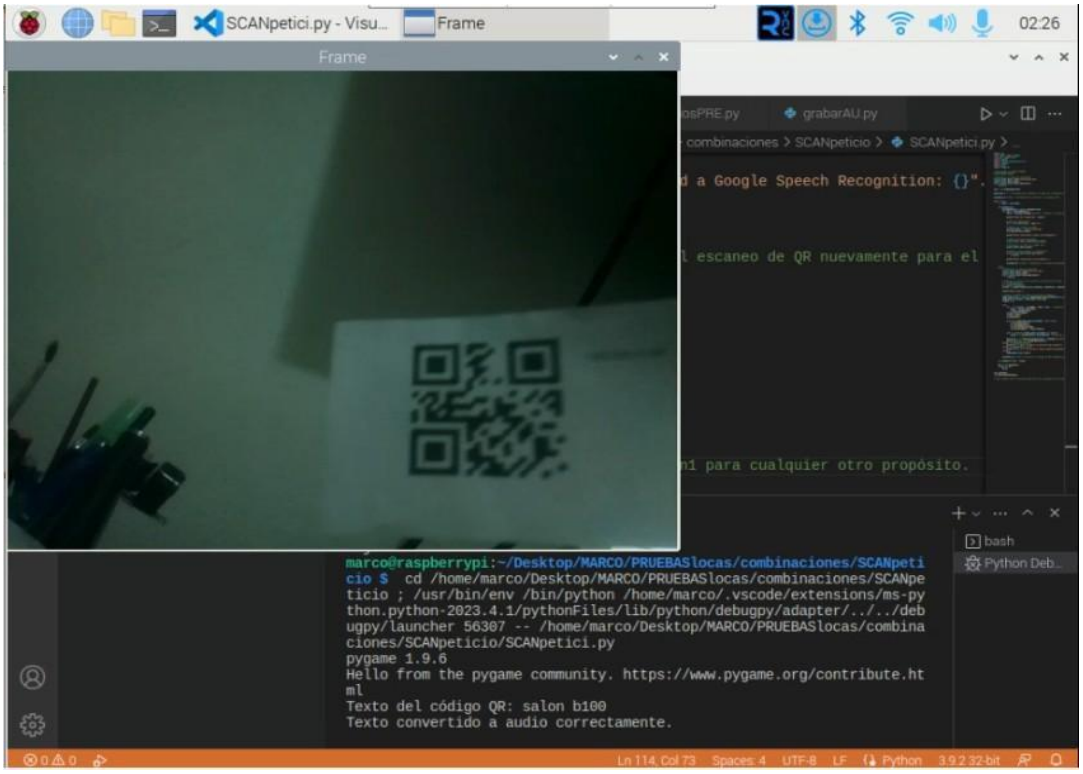


Fig. 32. Prueba de conversión de texto convertido a audio.
Fuente: Elaboración propia.

La siguiente prueba realizada es la correspondiente a la conversión de texto a audio. Esta etapa es implementada debido a que el usuario necesita recibir la información resultante del algoritmo de realización de rutas de forma audible. En la figura 33 podemos observar en el terminal que la leyenda audio reproducido correctamente. En esta prueba ya están incluidas las etapas previas de identificación de código QR y reconocimiento de audio a texto presentadas con anterioridad. El lugar a donde se requiere ir es “comedor” y por medio del QR se proporciona la ubicación origen la cual es B100.

Una vez probado el sistema de reconocimiento de voz, reconocimiento de texto y el reconocimiento de código QR en escenarios reales y su respectiva decodificación, se realizó la prueba al algoritmo de

generación automática de rutas. Para realizar la prueba, se propusieron los nodos de entrada, basados en las ubicaciones del plantel mencionado de la Universidad, con sus respectivas etiquetas y que se enlistan a continuación.

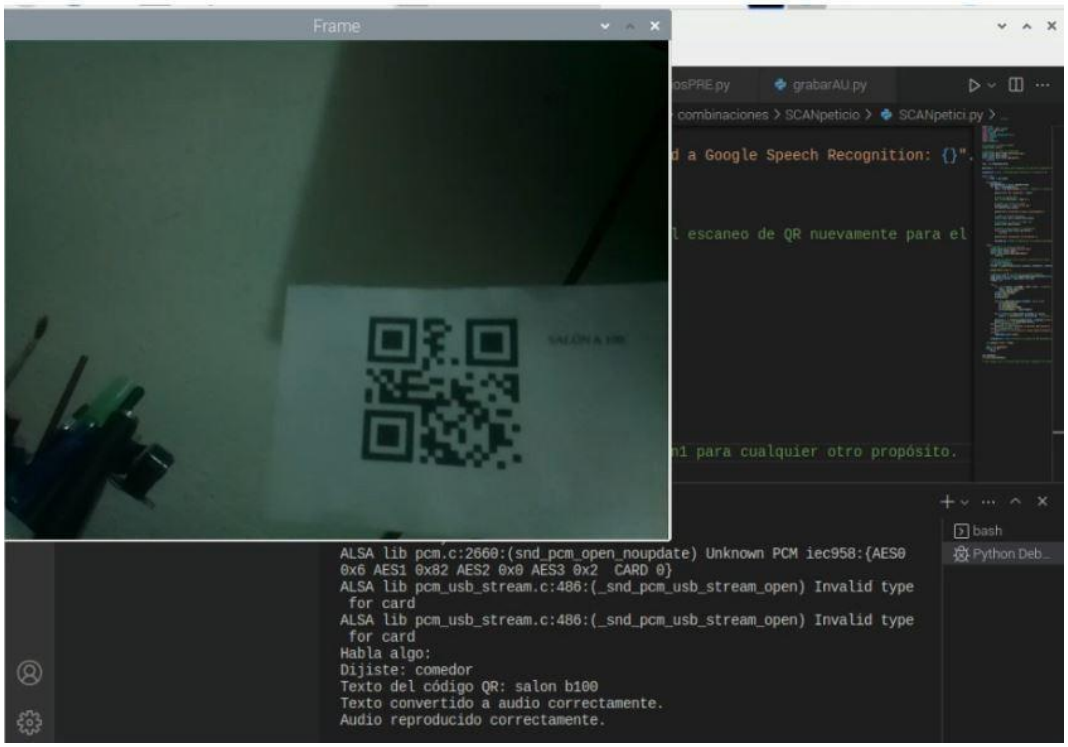


Fig. 33. Prueba de conversión texto a archivo de audio.
Fuente: Elaboración propia.

Lista de los nodos propuesto y que están basados en las ubicaciones seleccionadas previamente:

Nodo 1= Entrada

Nodo 2= Estacionamiento profesores

No-do 3= Edifico de cubículos planta baja

Nodo 4= Edifico de cubículos piso 1

No-do 5= Edifico de cubículos piso 2

Nodo 6= Gimnasio

Nodo 7= Jardineras

Nodo 8= Edificio A planta baja

Nodo 9= Edificio A piso 1

Nodo 10= Edificio A piso 2

Nodo 11= Edificio A piso 3

Nodo 12= Edificio A piso 4

Nodo 13= Edificio B planta baja

Nodo 14= Edificio B piso 1

Nodo 15= Edificio B piso 2

Nodo 16= Edificio B piso 3

Nodo 17= Edificio B piso 4

Nodo 18= Edificio C planta baja

Nodo 19= Edificio C piso 1

Nodo 20= Edificio C piso 2

Nodo 21= Edificio C piso 3

Nodo 22= Edificio C piso 4

Nodo 23= Letras habladas

Nodo 24= Comedor

Nodo 25= Explanada

Nodo 26= Biblioteca

Nodo 27= Estacionamiento estudiantes

Nodo 28= Ágora

Las conexiones entre ellos son los siguientes, del lado izquierdo se propone el nodo y del lado derecho el nodo con el que se encuentra, su posición y la distancia entre ellos, el nodo 1 es la referencia, pues se encuentra en el primer nivel del eje Z.

Nodo 1 – nodo 2: de frente, con una distancia de xx metros.

Nodo 2 – nodo 3: de frente, con una distancia de xx metros.

Nodo 2 – nodo 4: de frente con una distancia de xx metros, y después xx metros hacia arriba.

Nodo 3 – nodo 4: hacia arriba, xx metros.

Nodo 3 – nodo 6: de frente, con una distancia d xx metros.

Nodo 6 – nodo 8: a la derecha xx metros.

Nodo 8 – nodo 13: a la derecha xx metros.

Nodo 13 – nodo 24: de frente, con una distancia d xx metros.

Nodo 24 – nodo 26: de frente, con una distancia d xx metros.

Nodo 13 – nodo 18: a la derecha xx metros.

Nodo 18 – nodo 25: de frente, con una distancia d xx metros.

Nodo 18 – nodo 27: a la derecha xx metros.

Nodo 27 – nodo 28: de frente, con una distancia d xx metros.

Nodo 4 – nodo 5: hacia arriba xx metros.

Nodo 4 – nodo 9: de frente, con una distancia d xx metros, y luego a la derecha xx metros.

No-do 6 – nodo 8: a la derecha xx metros.

Nodo 8 – nodo 13: a la derecha xx metros.

Nodo 8 – nodo 9: hacia arriba xx metros.

Nodo 9 – nodo 14: a la derecha xx metros.

Nodo 9 – nodo 10: hacia arriba xx metros.

Nodo 10 – nodo 11: hacia arriba xx metros.

Nodo 11 – nodo 12: hacia arriba xx metros.

Nodo 14 – nodo 19: a la derecha xx metros.

Nodo 14 – nodo 15: hacia arriba xx metros.

Nodo 15 – nodo 16: hacia arriba xx metros.

Nodo 16 – nodo 17: hacia arriba xx metros.

Nodo 19 – nodo 20: hacia arriba xx metros.

Nodo 20 – nodo 21: hacia arriba xx metros.

Nodo 21 – nodo 22: hacia arriba xx metros.

Para este caso todas las distancias serán 1 metro, y se ejecutará en el Visual Code Studio.

En la figura 34 se puede observar el grafo formado por las condiciones mostradas en las listas propuestas previas.

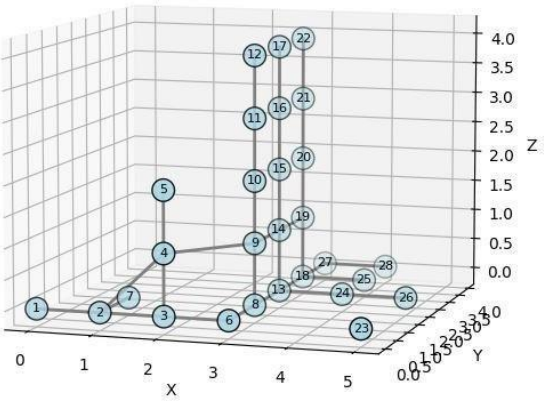


Fig. 34. Grafo resultante de la implementación del algoritmo diseñado.
Fuente: Elaboración propia.

El código utilizando NetworkX se presenta en la figura 35 y el código completo se encuentra en el Anexo 1.

```
import tensorflow as tf
from tensorflow import keras
import numpy as np

# Crear el grafo
G = nx.DiGraph()

# Definir los nodos con sus posiciones en el espacio 3D (X, Y, Z)
pos = {
    1: (0, 0, 0),
    2: (1, 0, 0),
    3: (2, 0, 0),
    4: (2, 0, 1),
    5: (2, 0, 2),
    6: (3, 0, 0),
    7: (1, 1, 0),
    8: (3, 1, 0),
    9: (3, 1, 1),
```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS

Python
Python

1/1 [==============================] - 0s 44ms/step
1/1 [==============================] - 0s 46ms/step
1/1 [==============================] - 0s 44ms/step
1/1 [==============================] - 0s 43ms/step

Figura 35. Segmento del código desarrollado usando NetworkX y Tensor Flow sobre Python.
Fuente: Elaboración propia.

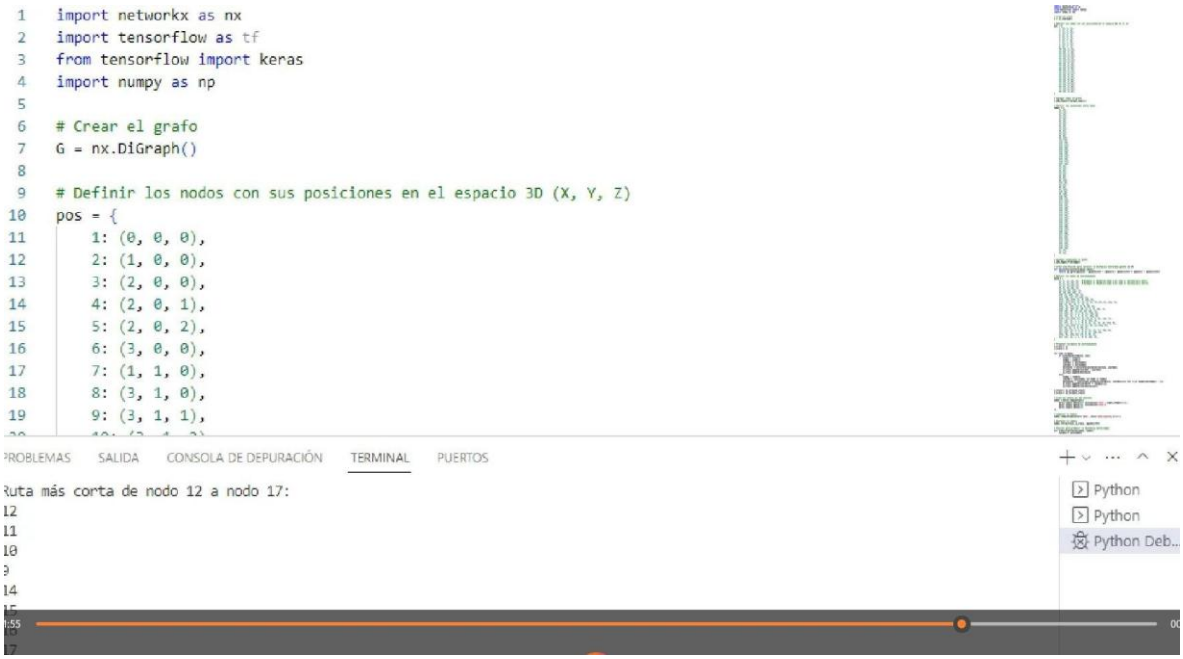


Figura 36. Resultado de encontrarse en nodo 12 y desplazarse al nodo 17.
Fuente: Elaboración propia.

En la figura 36 se presenta el caso de querer ir de la ubicación inicial Edificio A piso 4 (nodo 12) y dirigirse hacia el Edificio B piso 4 (nodo 17) el algoritmo propone la ruta más corta y que pasa por los no-dos 12, 11, 10, 9, 14, 15, 16 y finalmente 17. En la figura 24 podemos identificar los recorridos posicionándonos en el nodo 12 y seguir la secuencia propuesta por el algoritmo de generación automática de rutas.

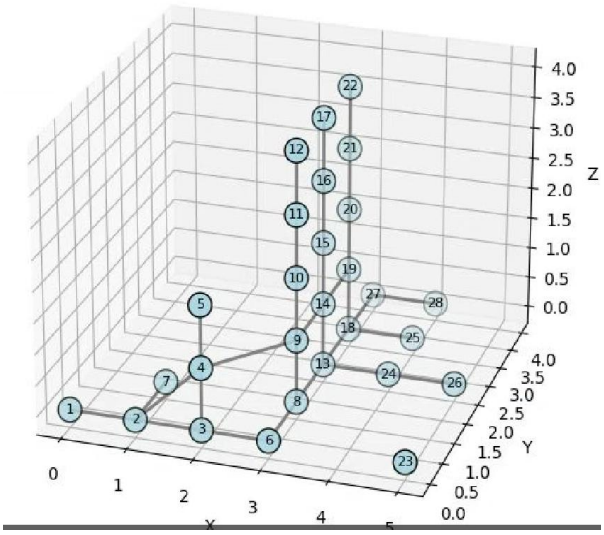


Fig. 37. Grafo construido con las ubicaciones del plantel San Lorenzo Tezonco.
Fuente: Elaboración propia.

Finalmente, la figura 38 y 39 se muestra el hardware utilizado para realizar las pruebas presentadas en el presente trabajo de tesis.

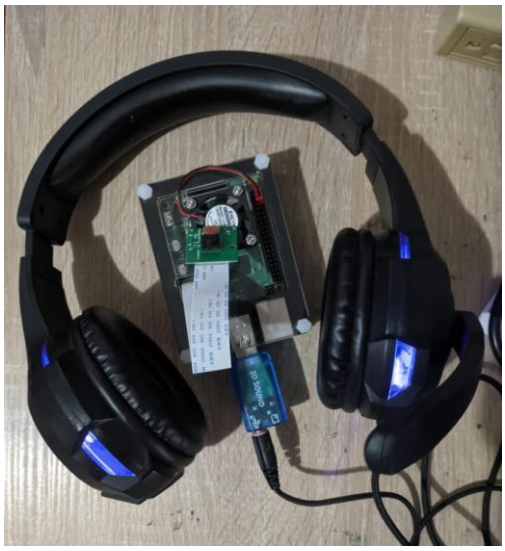


Fig. 38. Prototipo del sistema electrónico diseñado.
Fuente: Elaboración propia.

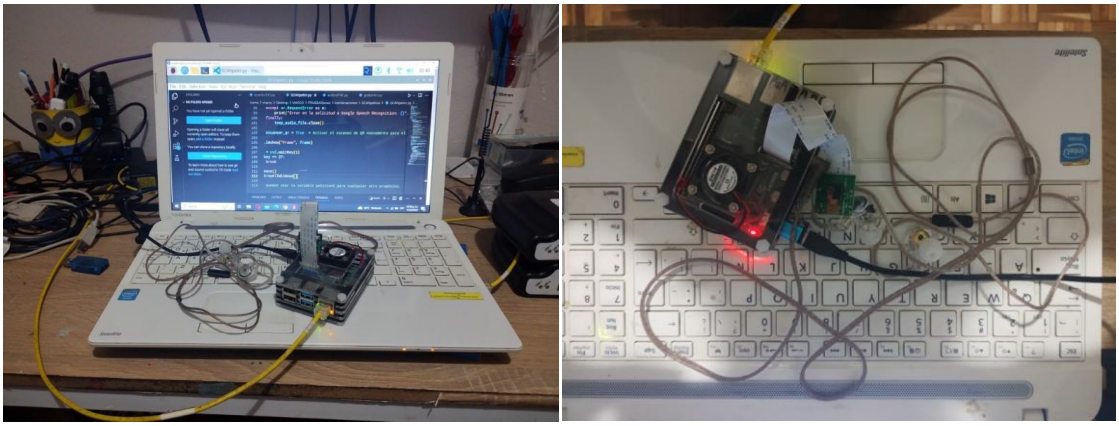


Fig. 39. Evidencia del proceso de programación del hardware seleccionado.
Fuente: Elaboración propia.

CAPÍTULO 6

CONCLUSIONES

6 Conclusiones generales

Durante la realización de este trabajo de tesis se desarrolló y probó un sistema portátil para generación de rutas de forma automática para uso de personas con capacidades visuales diferentes.

Se propusieron y probaron algoritmos de reconocimiento y decodificación de código QR captados mediante una cámara de alta resolución. Además, se desarrollaron algoritmos para conversión de datos de texto a archivos de audio y archivos de audio a texto para poder interactuar con el usuario del sistema.

Se desarrolló y probó un algoritmo para generación automática de rutas previa la generación de un grafo que se basa en las localizaciones físicas del plantel San Lorenzo Tezonco de nuestra Universidad. El algoritmo desarrollado se basó en técnicas de inteligencia artificial usando teoría de grafos para su mejor desempeño. Los resultados obtenidos en las diversas pruebas aplicadas al sistema validan la propuesta presentada en el presente trabajo de tesis.

REFERENCIAS

[1] «Movilidad 4s para México Saludable, Segura, Sustentable y Solidaria. Plan de Movilidad para una nueva normalidad | Secretaría de Desarrollo Agrario, Territorial y Urbano | Gobierno |gob.mx». Accedido: 4 de junio de 2024. [En línea]. Disponible en: https://www.gob.mx/sedatu/documentos/movilidad-4s-para-mexico-saludable-segura-sustentable- y-solidaria-plan-de-movilidad-para-una-nueva-normalidad

[2] «Población. Discapacidad». Accedido: 4 de junio de 2024. [En línea]. Disponible en: https://cuentame.inegi.org.mx/poblacion/discapacidad.aspx

[3] «Systems for Improving Mobility of People with Visual Impairment.», IEEE Trans. OnSystems Man Cybern. Syst., vol. 48, n.o 2, pp. 338-351.

[4] S. T. Brassai, L. Bakó, y L. Losonczi, «Assistive Technologies for Visually Impaired People».

[5] R. Velasquez, E. Pissaloux, Carolina Del Valle Soto, y Carrasco Zambrano MIguel Angel, «MOVILIDAD PARA INVIDENTES UTILIZANDO EL GPS DEL TELÉFONO INTELIGENTE Y UN DISPOSITIVO TÁCTIL VESTIBLE», DYNA, vol. 96, n.o 1, pp. 98-104, ene. 2021, doi: http://dx.doi.org/10.6036/9635.

[6] M. G. Sánchez, «Autonomía de personas ciegas en supermercados mediante realidad aumentada».

[7] «Tarjetas de desarrollo – Sistemas Digitales». Accedido: 12 de junio de 2024. [En línea]. Disponible en: https://virtual.cuautitlan.unam.mx/intar/sistdig/tarjetas-de-desarrollo/

[8] V. Ghodke, «System-On-Chip (SoC) — Security Design And Modelling», Security Risks In Systems-On-Chip (SOCs). Accedido: 12 de junio de 2024. [En línea]. Disponible en: https://medium.com/security-risks-in-systems-on-chip-socs/system-on-chip-soc-security-design- and-modelling-306bbc536324

[9] «D Qué es un SOC y cuáles son sus características principales 🥇 », Profesional Review. Accedido: 12 de junio de 2024. [En línea]. Disponible en: https://www.profesionalreview.com/hardware/que-es-un-soc-caracteristicas/

[10] S. Carranza, «CONOCIENDO AL ESP32», TodoMaker. Accedido: 12 de junio de 2024. [En línea]. Disponible en: https://todomaker.com/blog/conociendo-al-esp32/

[11] «Microcontrolador Intel® QuarkTM D1000 Especificaciones del producto». Accedido: 12 de junio de 2024. [En línea]. Disponible en: https://www.intel.la/content/www/xl/es/products/sku/86826/intel-quark-microcontroller-d1000.html

[12] «Raspberry Pi Documentation». Accedido: 12 de junio de 2024. [En línea]. Disponible en: https://www.raspberrypi.com/documentation/

[13] «Arduino Docs | Arduino Documentation». Accedido: 12 de junio de 2024. [En línea]. Disponible en: https://docs.arduino.cc/

[14] «Arduino 101 | Arduino Documentation». Accedido: 12 de junio de 2024. [En línea]. Disponible en: https://docs.arduino.cc/retired/boards/arduino-101-619/

[15] «BeagleY-AI — BeagleBoard Documentation». Accedido: 12 de junio de 2024. [En línea].
Disponible en: https://docs.beagleboard.org/latest/boards/beagley/ai/index.html#beagley-ai-

home
[16] Stuart Rusell y Peter Norvig, Inteligencia Artificial, un enfoque moderno.
[17] J. Amat Rodrigo, «cienciadedatos.net», ene. 2021, doi: 10.5281/ZENODO.10006330.
[18] «Analisis comparativo de algoritmos de minería de subgrafos frecuentes», doi: https://doi.org/10.18294/relais.2016.111-142.
[19] «Aplicación de estrategias de recorrido de grafos en la vida real», Aplicación de estrategias de recorrido de grafos en la vida real. Accedido: 12 de junio de 2024. [En línea]. Disponible en: https://unneaed2.blogspot.com/2017/10/aplicacion-de-estrategias-de-recorrido.html
[20] DANIEL BRITO, LOPE MARÍN, y HENRY RAMÍREZ, «CICLOS HAMILTONIANOS QUE PASAN A TRAVÉS DE UN BOSQUE LINEAL EN GRAFOS BIPARTITOS BA-LANCEADOS».
[21] R. P. Ltd, «Buy a Raspberry Pi Camera Module 2 NoIR», Raspberry Pi. Accedido: 12 de junio de 2024. [En línea]. Disponible en: https://www.raspberrypi.com/products/pi-noir-camera-v2/
[22] «Welcome to Python.org», Python.org. Accedido: 12 de junio de 2024. [En línea]. Disponible en:
https://www.python.org/
[23] «Visual Studio Code - Code Editing. Redefined». Accedido: 12 de junio de 2024. [En línea].
Disponible en: https://code.visualstudio.com/
[24] «¿Qué es Python? - Explicación del lenguaje Python - AWS», Amazon Web Services, Inc. Accedido: 12 de junio de 2024. [En línea]. Disponible en: https://aws.amazon.com/es/what- is/python/
[25] «¿Qué es TensorFlow y para qué sirve?», ¿Qué es TensorFlow y para qué sirve? Accedido: 12 de junio de 2024. [En línea]. Disponible en: https://www.incentro.com/es-ES/blog/que-es-tensorflow
[26] A. S. Alberca, «La librería Numpy», Aprende con Alf. Accedido: 12 de junio de 2024. [En línea].
Disponible en: https://aprendeconalf.es/docencia/python/manual/numpy/
[27] Fundación ONCE, "Fundación ONCE y ReadSpeaker se unen para potenciar la accesibilidad digital," Fundación ONCE, 27-Jun-2023. [Enlace]. Disponible en: https://www.fundaciononce.es/es/comunicacion/noticias/fundacion-once-y-readspeaker-se-unen- para-potenciar-la-accesibilidad-digital. [Accedido: 07-Ene-2025].
[28] MarketsandMarkets, "Mexico Factory Automation and Machine Vision Market - Growth, Trends, and Forecast (2023 - 2028)", [Enlace], disponible en: https://www.marketsandmarkets.com/Market-Reports/mexico-factory-automation-machine-vi-sion-market-149652036.html. [Accedido: 11-Ene-2025].

Anexo 1

El anexo 1 presenta los códigos desarrollados durante la realización del presente trabajo de tesis, se pre-senta la interfaz de usuario implantada en el hardware seleccionado, después se presenta el código de la generación del grafo correspondiente a las ubicaciones seleccionadas y finalmente se incluye el código del algoritmo completo.

- Interfaz de usuario-

```
import cv2
from pyzbar import pyzbar from gtts import gTTS im-port pygame
import speech_recognition as sr import pyaudio
import wave import tempfile
# Inicializar el módulo PyGame pygame.mixer.init()
# Reproducir el audio de bienvenida pyga-me.mixer.music.load("bienvenida.mp3") pyga-me.mixer.music.play()
while pygame.mixer.music.get_busy():
continue
cap = cv2.VideoCapture(0)
peticion1 = "" # Variable para almacenar el texto de la petición del usuario escanear_qr = True # Variable para controlar el escaneo de QR
while True:
_, frame = cap.read() if escanear_qr:

decodedObjects = pyzbar.decode(frame) for obj in deco-dedObjects:
texto = obj.data.decode("utf-8") # Obtener el texto del código QR print("Texto del código QR:", texto)
# Crear un objeto gTTS
tts = gTTS(text=texto, lang='es') # Guardar el archivo de audio
archivo_audio = "audio_salida.mp3" tts.save(archivo_audio)
```

```
print("Texto convertido a audio correctamente.") # Cargar el archivo de audio pyga-
me.mixer.music.load(archivo_audio)
# Reproducir el audio en tiempo real pyga-me.mixer.music.play()
# Esperar a que termine la reproducción whi-le pygame.mixer.music.get_busy():
continue
print("Audio reproducido correctamente.")
escanear_qr = False # Desactivar el escaneo de QR después de reproducir una vez else:
# Reproducir el audio de petición pyga-me.mixer.music.load("peticion.mp3") pyga-
me.mixer.music.play()
while pygame.mixer.music.get_busy():
continue
# Capturar la petición del usuario y convertirla en texto r = sr.Recognizer()
p = pyaudio.PyAudio()
stream = p.open(format=pyaudio.paInt16, channels=1, rate=16000, input=True, fra-
mes_per_buffer=1024)
print("Habla algo:")
# Captura y guarda el audio en un archivo temporal

temp_audio_file = tempfile.NamedTemporaryFile(delete=False) temp_audio_filename =
temp_audio_file.name
frames = [] try:
for _ in range(0, int(16000 / 1024 * 15)): # Capturar audio durante 15 segundos data =
stream.read(1024)
frames.append(data) stream.stop_stream() stream.close()

p.terminate()
with wave.open(temp_audio_filename, 'wb') as wf: wf.setnchannels(1)
wf.setsampwidth(2) wf.setframerate(16000) wf.writeframes(b".join(frames))
with sr.AudioFile(temp_audio_filename) as source:
audio = r.record(source, duration=15) # Escuchar el audio grabado peticion1 =
r.recognize_google(audio, language="es-ES") print("Dijiste: {}".format(peticion1))
```

except sr.UnknownValueError:
print("No se pudo reconocer la petición del usuario") except sr.RequestError as e:
print("Error en la solicitud a Google Speech Recognition: {}".format(e)) fi-nally:
temp_audio_file.close()
escanear_qr = True # Activar el escaneo de QR nuevamente para el próximo ciclo
cv2.imshow("Frame", frame)
key = cv2.waitKey(1) if key == 27:
break cap.release()
cv2.destroyAllWindows()
# Ahora puedes usar la variable peticion1 para cualquier otro propósito.

- Creación de grafo usando NetworkX-

# Crear un grafo dirigido (grafo) utilizando NetworkX im-port networkx as nx
import tensorflow as tf
from tensorflow import keras im-port numpy as np

G = nx.DiGraph()

Definir los nodos con sus posiciones en el espacio 3D (X, Y, Z) pos = {
1: (0, 0, 0),
2: (1, 0, 0),
3: (2, 0, 0),
4: (2, 0, 1),
5: (2, 0, 2),
6: (3, 0, 0),
7: (1, 1, 0),
8: (3, 1, 0),
9: (3, 1, 1),

10: (3, 1, 2),
11: (3, 1, 3),
12: (3, 1, 4),
13: (3, 2, 0),
14: (3, 2, 1),
15: (3, 2, 2),
16: (3, 2, 3),
17: (3, 2, 4),
18: (3, 3, 0),
19: (3, 3, 1),
20: (3, 3, 2),
21: (3, 3, 3),
22: (3, 3, 4),
23: (5, 0, 0),
24: (4, 2, 0),
25: (4, 3, 0),
26: (5, 2, 0),
27: (3, 4, 0),
28: (4, 4, 0),
}

Agregar nodos al grafo G.add_nodes_from(pos.keys())

Definir las conexiones entre nodos ed-ges = [
(1, 2),
(2, 3),
(2, 4),
(3, 4),
(3, 6),
(6, 8),
(8, 13),

```
(13, 24),
(24, 26),
(13, 18),
(18, 25),
(18, 27),
(27, 28),
(4, 5),
(4, 9),
(6, 8),
(8, 13),
(8, 9),
(9, 14),
(9, 10),
(10, 11),
(11, 12),
(14, 19),

(14, 15),
(15, 16),
(16, 17),
(19, 20),
(20, 21),
(21, 22),
(13, 14),
(18, 19),
(2, 7),
]

# Agregar conexiones al grafo G.add_edges_from(edges)

# Imprimir las posiciones de los nodos en el espacio 3D for node, (x, y, z) in pos.items():
```

```
print(f"Nodo {node}: ({x}, {y}, {z})")
# Definir ejemplos de datos de rutas y distancias data = [
(1, 2, [1, 2], 1), # Ejemplo 1: Desde el nodo 1 al nodo 2, distancia 1 metro
(2, 3, [2, 3], 1), # Ejemplo 2: Desde el nodo 2 al nodo 3, distancia 1 metros
(3, 6, [3, 6], 1),
(6, 8, [6, 8], 1),
(8, 13, [8, 13], 1),
(9, 10, [9, 10], 1),
(10, 11, [10, 11], 1),
((8, 18), [13, 18], 2),
((19, 10), [14, 9, 10], 3),
((12, 22), [11, 10, 9, 14, 19, 20, 21, 22], 8),
((9, 1), [4, 2, 1], 3),
((24, 3), [13, 8, 6, 3], 4),
((28, 9), [14, 9, 10], 3),
((9, 26), [14, 13, 24, 26], 4),
((2, 14), [4, 9, 14], 3),
((2, 14), [3, 6, 8, 13, 14], 5),
((28, 11), [27, 18, 13, 8, 9, 10 ,11], 7),
((1, 13), [2, 3, 6, 8, 13], 5),
((1, 17), [2, 3, 6, 8, 13, 14, 15, 16, 17], 9),
((10, 22), [9, 14, 19, 20, 21, 22], 6),
((1, 5), [2, 4, 5], 3),
((1, 28), [2, 3, 6, 8, 13, 18, 27, 28], 8),
((1, 12), [2, 3, 6, 8, 13], 5),
((26, 9), [24, 13, 8, 9], 4),
((1, 13), [2, 3, 6, 8, 13], 5),
]

# Definir las características (features) de entrada y el objetivo (target) de salida features =
[ruta for _, _, ruta, distancia in data]
```

```
target = [distancia for _, _, _, distancia in data]

# Convertir las características (features) en un array de NumPy features = [np.array(ruta)
for ruta in features]

# Crear un modelo de regresión para predecir distancias model = keras.Sequential([
keras.layers.Input(shape=(None,)), # Capa de entrada que acepta secuencias de nodos ke-
ras.layers.Embedding(input_dim=29, output_dim=64), # Capa de embedding para nodos
ke-ras.layers.LSTM(64, return_sequences=True), # Capa LSTM para procesar la secuencia
de nodos ke-ras.layers.Dense(1) # Capa de salida para predecir la distancia
])

# Compilar el modelo
model.compile(optimizer='adam', loss='mean_squared_error')

# Entrenar el modelo
model.fit(features, target, epochs=100, batch_size=32)

# Una vez entrenado el modelo, puedes usarlo para hacer predicciones inicio = 1
destino = 13
prediccion = model.predict([(inicio, destino)])

print(f"Predicción de distancia desde el nodo {inicio} al nodo {destino}:
{prediccion[0][0]} metros")
```

- Aplicación completa desarrollada-

```
import cv2
from pyzbar import pyzbar from gtts import gTTS im-port pygame
import speech_recognition as sr import pyaudio
import wave import tempfile

# Inicializar el módulo PyGame pygame.mixer.init()

# Reproducir el audio de bienvenida pyga-me.mixer.music.load("bienvenida.mp3") pyga-
me.mixer.music.play()
while pygame.mixer.music.get_busy(): con-tinue

cap = cv2.VideoCapture(0)

peticion1 = "" # Variable para almacenar el texto de la petición del usuario escanear_qr =
True # Variable para controlar el escaneo de QR
while True:
_, frame = cap.read()

if escanear_qr:
decodedObjects = pyzbar.decode(frame) for obj in deco-dedObjects:
texto = obj.data.decode("utf-8") # Obtener el texto del código QR print("Texto del código
QR:", texto)
# Crear un objeto gTTS
tts = gTTS(text=texto, lang='es')
# Guardar el archivo de audio archi-vo_audio = "audio_salida.mp3" tts.save(archivo_audio)
print("Texto convertido a audio correctamente.")

# Cargar el archivo de audio pyga-me.mixer.music.load(archivo_audio)
```

```
# Reproducir el audio en tiempo real pyga-me.mixer.music.play()

# Esperar a que termine la reproducción while pygame.mixer.music.get_busy():
continue
print("Audio reproducido correctamente.")
escanear_qr = False # Desactivar el escaneo de QR después de reproducir una vez

else:
# Reproducir el audio de petición pyga-me.mixer.music.load("peticion.mp3")

pygame.mixer.music.play()
while pygame.mixer.music.get_busy(): continue

# Capturar la petición del usuario y convertirla en texto r = sr.Recognizer()
p = pyaudio.PyAudio()
stream = p.open(format=pyaudio.paInt16, channels=1, rate=16000, input=True, fra-
mes_per_buffer=1024)

print("Habla algo:")

# Captura y guarda el audio en un archivo temporal temp_audio_file =
tempfile.NamedTemporaryFile(delete=False) temp_audio_filename =
temp_audio_file.name
frames = []

try:
for _ in range(0, int(16000 / 1024 * 15)): # Capturar audio durante 15 segundos data =
stream.read(1024)
frames.append(data) stream.stop_stream() stream.close() p.terminate()

with wave.open(temp_audio_filename, 'wb') as wf: wf.setnchannels(1)
```

```
wf.setsampwidth(2) wf.setframerate(16000) wf.writeframes(b".join(frames))

with sr.AudioFile(temp_audio_filename) as source:
audio = r.record(source, duration=15) # Escuchar el audio grabado

peticion1 = r.recognize_google(audio, language="es-ES") print("Dijiste:
{}".format(peticion1))
except sr.UnknownValueError:
print("No se pudo reconocer la petición del usuario") ex-cept sr.RequestError as e:
print("Error en la solicitud a Google Speech Recognition: {}".format(e)) fi-nally:
temp_audio_file.close()
escanear_qr = True # Activar el escaneo de QR nuevamente para el próximo ciclo
cv2.imshow("Frame", frame)
key = cv2.waitKey(1) if key == 27:

break

cap.release() cv2.destroyAllWindows()


Generacion de grafos usando Network X im-port networkx as nx
import tensorflow as tf
from tensorflow import keras im-port numpy as np

G = nx.DiGraph()

# Definir los nodos con sus posiciones en el espacio 3D (X, Y, Z) pos = {
1: (0, 0, 0),
2: (1, 0, 0),
3: (2, 0, 0),
4: (2, 0, 1),
```

5: (2, 0, 2),
6: (3, 0, 0),
7: (1, 1, 0),
8: (3, 1, 0),
9: (3, 1, 1),
10: (3, 1, 2),
11: (3, 1, 3),
12: (3, 1, 4),
13: (3, 2, 0),
14: (3, 2, 1),
15: (3, 2, 2),
16: (3, 2, 3),
17: (3, 2, 4),
18: (3, 3, 0),
19: (3, 3, 1),
20: (3, 3, 2),
21: (3, 3, 3),
22: (3, 3, 4),
23: (5, 0, 0),
24: (4, 2, 0),
25: (4, 3, 0),
26: (5, 2, 0),
27: (3, 4, 0),
28: (4, 4, 0),
}

Agregar nodos al grafo

G.add_nodes_from(pos.keys())

Definir las conexiones entre nodos ed-ges = [

(1, 2),
(2, 3),
(2, 4),
(3, 4),
(3, 6),
(6, 8),
(8, 13),
(13, 24),
(24, 26),
(13, 18),
(18, 25),
(18, 27),
(27, 28),
(4, 5),
(4, 9),
(6, 8),
(8, 13),
(8, 9),
(9, 14),
(9, 10),
(10, 11),
(11, 12),
(14, 19),
(14, 15),
(15, 16),
(16, 17),
(19, 20),
(20, 21),
(21, 22),
(13, 14),
(18, 19),

```
(2, 7),
]

# Agregar conexiones al grafo G.add_edges_from(edges)
# Imprimir las posiciones de los nodos en el espacio 3D for node, (x, y, z) in pos.items():
print(f"Nodo {node}: ({x}, {y}, {z})")
# Definir ejemplos de datos de rutas y distancias data = [
(1, 2, [1, 2], 1), # Ejemplo 1: Desde el nodo 1 al nodo 2, distancia 1 metro
(2, 3, [2, 3], 1), # Ejemplo 2: Desde el nodo 2 al nodo 3, distancia 1 metros
(3, 6, [3, 6], 1),

(6, 8, [6, 8], 1),
(8, 13, [8, 13], 1),
(9, 10, [9, 10], 1),
(10, 11, [10, 11], 1),
((8, 18), [13, 18], 2),
((19, 10), [14, 9, 10], 3),
((12, 22), [11, 10, 9, 14, 19, 20, 21, 22], 8),
((9, 1), [4, 2, 1], 3),
((24, 3), [13, 8, 6, 3], 4),
((28, 9), [14, 9, 10], 3),
((9, 26), [14, 13, 24, 26], 4),
((2, 14), [4, 9, 14], 3),
((2, 14), [3, 6, 8, 13, 14], 5),
((28, 11), [27, 18, 13, 8, 9, 10 ,11], 7),
((1, 13), [2, 3, 6, 8, 13], 5),
((1, 17), [2, 3, 6, 8, 13, 14, 15, 16, 17], 9),
((10, 22), [9, 14, 19, 20, 21, 22], 6),
((1, 5), [2, 4, 5], 3),
((1, 28), [2, 3, 6, 8, 13, 18, 27, 28], 8),
((1, 12), [2, 3, 6, 8, 13], 5),
```

```
((26, 9), [24, 13, 8, 9], 4),
((1, 13), [2, 3, 6, 8, 13], 5),
]

# Definir las características (features) de entrada y el objetivo (target) de salida features =
[ruta for _, _, ruta, distancia in data]
target = [distancia for _, _, _, distancia in data]
# Convertir las características (features) en un array de NumPy features = [np.array(ruta)
for ruta in features]

# Crear un modelo de regresión para predecir distancias model = keras.Sequential([
keras.layers.Input(shape=(None,)), # Capa de entrada que acepta secuencias de nodos ke-
ras.layers.Embedding(input_dim=29, output_dim=64), # Capa de embedding para nodos
ke-ras.layers.LSTM(64, return_sequences=True), # Capa LSTM para procesar la secuencia
de nodos ke-ras.layers.Dense(1) # Capa de salida para predecir la distancia
])

# Compilar el modelo
model.compile(optimizer='adam', loss='mean_squared_error')
# Entrenar el modelo
model.fit(features, target, epochs=100, batch_size=32)
# Una vez entrenado el modelo, puedes usarlo para hacer predicciones inicio = 1
destino = 13
prediccion = model.predict([(inicio, destino)])

print(f"Predicción de distancia desde el nodo {inicio} al nodo {destino}:
{prediccion[0][0]} metros")
```

ANEXO 2

Publicación de póster en “Congreso Nacional Multidisciplinario de Innovación, Ciencia y Tecnología (CoNMICyT – 2023)” Se anexa carta de aceptación, publicación presentada y constancia de participación.

Aceptación de Poster Científico para el CoNMICyT 2023

3 mensajes

Congreso Nacional UPIIH-CoNMICyT <conmicyt@ipn.mx> 26 de septiembre de 2023, 9:28
Para: "eduardo.ramos@uacm.edu.mx" <eduardo.ramos@uacm.edu.mx>

Estimado Marco Rafael Soto Vargas

Es un placer para nosotros confirmar la recepción de su trabajo de investigación con título "**Dispositivo portátil de generación de rutas de tránsito para personas con discapacidad visual**" el cual será presentado en la edición 2023 del *Congreso Nacional Multidisciplinario de Innovación, Ciencia y Tecnología (CoNMICyT)* en la modalidad de **poster científico**. Para poder completar este proceso le solicitamos tomar en cuenta y cumplir las siguientes indicaciones:

- Debe enviar respuesta a este correo confirmando su participación en el evento a más tardar el día **Viernes 29 de Septiembre del presente año**.
- En la confirmación debe de adjuntarse la versión final del poster científico a presentar en formato **PDF** con un tamaño de archivo máximo de **10MB**.
- El poster científico debe ser impreso por el participante en un tamaño de **100cm x 70cm** previo a su asistencia al evento.
- Los posters científicos serán presentados el día **6 de Octubre** en un horario de **12:00-13:00** horas en las instalaciones del evento.
- El participante que presentará el poster durante la sesión debe presentarse el mismo día en un horario de **11:15- 11:30** con los miembros del comité organizador para recibir instrucciones sobre la ubicación de sus espacios asignados, forma de colocación del poster y sobre la dinámica del evento.

Se les agradece su interés para participar en dicho evento y estamos a la espera de recibirlos y con ello ustedes puedan dar a conocer los avances en su línea de trabajo.

Sin más por el momento

quedamos atentos a su

respuesta. Saludos cordiales

Eduardo Ramos Díaz <eduardo.ramos@uacm.edu.mx> 28 de septiembre de 2023, 22:14
Para: Congreso Nacional UPIIH-CoNMICyT <conmicyt@ipn.mx>

Dispositivo portátil de generación de rutas de tránsito para personas con discapacidad visual

Marco Rafael Soto Vargas, Eduardo Ramos Díaz, Daniel Tapia Sánchez

Universidad Autónoma de la Ciudad de México, Calle Prolongación San Isidro 152, San Lorenzo Tezonco, Iztapalapa, Ciudad de México, 09790. Tel. (55)51349804
eduardo.ramos@uacm.edu.mx

RESUMEN/ABSTRACT

En este trabajo se presenta un dispositivo portátil que permite a personas con discapacidad visual tomar decisiones referentes a rutas de tránsito de lugares públicos. Se basa en la lectura de códigos QR y una interface con información auditiva sobre su entorno, generando de forma automática rutas independientes en lugares públicos. Utilizando una Raspberry Pi 4 y componentes integrados, el sistema se adapta a diferentes escenarios, sólo basta cargar el mapa de destinos del lugar de interés. Las etapas desarrolladas incluyen el reconocimiento de voz, generación de audio, lectura de códigos QR, creación de mapas de rutas con IA y la implementación de la propuesta en un sistema embebido.

INTRODUCCIÓN

Navegar en entornos públicos es una tarea que generalmente requiere la colaboración de nuestros sentidos y un procesamiento visual de la información. Sin embargo, para aquellos que enfrentan debilidades visuales, esta tarea se convierte en un desafío considerable.

Las personas con discapacidad visual necesitan ayuda de terceras personas al momento de la toma de decisiones para el traslado dentro de las instalaciones de diferentes espacios públicos, lo cual reduce la movilidad de las personas. Existen diferentes propuestas en la literatura que abordan esta problemática [1-6] que van desde el trazado de rutas usando GPS y dispositivos vestibles, las autonomía de personas ciegas en supermercados, sistemas de orientación para personas invidentes por medio de comandos de voz y hasta mensajería instantánea para personas con ceguera
Es por ello que proponemos la utilización de una Raspberry Pi 4 junto con un conjunto de componentes que conforman un sistema de hardware integrado, diseñado para simplificar esta tarea.

OBJETIVO

Diseñar y construir un sistema electrónico portátil que permita a las personas invidentes, a través de información audible del entorno en donde se encuentra, tomar decisiones referentes a la ruta a transitar dentro de un recinto

METODOLOGÍA

El desarrollo de este proyecto se ha dividido en etapas fundamentales. Inicialmente, se han implementado algoritmos de reconocimiento de voz para comprender las instrucciones verbales de los usuarios. Además, hemos habilitado la capacidad de generar respuestas en formato de audio, permitiendo así la entrega de instrucciones claras y precisas.
También hemos incorporado la capacidad de realizar lecturas en tiempo real de códigos QR para identificar ubicaciones, y se desarrolla un sistema de generación de rutas automáticas utilizando inteligencia artificial, específicamente diseñado para el recinto seleccionado. Las librerías usadas en la creación del sistema (Fig. 1) son las siguientes: cv2 (OpenCV), Pyzbar, Pygame, speech_recognition, Pyaudio, Wave, Tempfile, Threading, TensorFlow, Pandas, Folium; todas en conjunto logran realizar las tareas, de recolección de información y diseño de rutas [7].

En la Fig. 2, se presenta el mapa de destinos utilizado como entrada del sistemas generador automático de rutas.

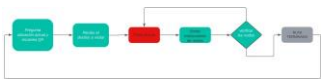


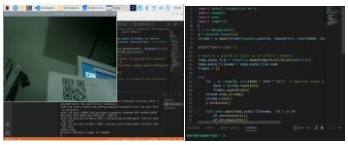
Fig. 1. Diagrama de funcionamiento.

Fig. 2. Mapa de destinos

RESULTADOS

La implementación de la interfaz de usuario se realizó en tres etapas. Primero, se desarrolló un programa para la captura y lectura en tiempo real de códigos QR utilizando CV2 y pyzbar (Fig. 3).

Fig. 3. Escaneo y reconocimiento de código QR en tiempo real



a) b)

Fig. 4. Interfaz de usuario reconociendo comandos de voz a) después de completar el proceso de escaneo y solicitud de dirección, b) código desarrollado

Luego, se creó un segundo programa para convertir texto en audio, seguido por un tercer programa que realiza la función inversa.
La integración exitosa de estos programas, abarcando el escaneo, el reconocimiento de voz y la reproducción de audio, se destaca en la Fig. 4.

a) b)

Fig. 5. Ejercicio de a) mapa con vectores y nodos, b) cálculo de distancias.

Por último se crea una replica del mapa de forma digital y se prueban distancias entre nodos. (Fig. 5)

Replicar un mapa en formato digital utilizando código Python es un paso crítico para lograr los objetivos planteados en este proyecto. En este sentido, se optó por diseñar un mapa tridimensional, ya que esta representación es fundamental para mapear de manera precisa todo el entorno. El mapa se compone de una serie de vectores interconectados, donde cada nodo representa un destino, como se ilustra en la Fig. 2.
Cada vector a su vez representa una posible elección en el trayecto. En este punto, entra en juego el algoritmo de Dijkstra [7, 8] y la inteligencia artificial, los cuales habilitan al sistema para encontrar la ruta mas corta de acuerdo al algoritmo. Este enfoque de representación y análisis del mapa resulta fundamental para el éxito de nuestro proyecto.

Fig. 6. Hardware utilizado en el proyecto

El paso final del proyecto es la integración de las etapas desarrolladas y su puesta a punto.

En la Fig. 6, podemos observar el hardware utilizado en el desarrollo del proyecto, el cual está compuesto por unos auriculares, cámara de video, Raspberry Pi 4.
Con el software desarrollado se tiene la posibilidad de cargar el mapa de destino del cualquier localidad, tales como supermercados, estaciones de autobús, teatros, cines, estadios, entre otros.

CONCLUSIONES

La capacidad de escanear códigos QR en tiempo real para identificar ubicaciones y la generación de rutas automáticas mediante inteligencia artificial demuestran el potencial de esta tecnología para mejorar la autonomía y la calidad de vida de las personas con discapacidad visual en entornos públicos mostrando un gran potencial en la inclusión y la accesibilidad.

La creación de un mapa digital tridimensional y su implementación junto con algoritmos de caminos mínimos como Dijkstra han sido utilizados en la generación de rutas óptimas de forma automática.

El desarrollo de las etapas de escaneo, el reconocimiento de voz la generación y reproducción de audio, y la generación automática de rutas propuestos en el proyecto se han realizado, se tiene como trabajo futuro la integración de las etapas y su puesta a punto.

La integración de tecnología y algoritmos promete mejorar la autonomía del usuario en entornos públicos, mostrando un gran potencial en la inclusión y la accesibilidad.

Los autores agradecen a la Universidad Autónoma de la Ciudad de México por el apoyo brindado en el Proyecto 2021-7.

REFERENCIAS

[1] Paredes, A. (2019). "Influencia de las experiencias sensoriales de la arquitectura en la accesibilidad de un centro de información integral para personas invidentes en Trujillo. De Universidad Privada del Norte, Lima, Perú.
[2] Velázquez-Guerrero, R., Pissaloux, E., Del-Valle-Soto, C., Carrasco-Zambrano, M.-Á., Mendoza-Andrade, A., & Varona-Salazar, J. (2021). Movilidad para invidentes utilizando el GPS del teléfono inteligente y un dispositivo táctil vestible. (Spanish). DYNA - Ingeniería e Industria, 96(1), 98–104.
[3] Chávez, P. (2019). Desarrollo de Sistema de Orientación para Personas Invidentes aplicando comandos de Voz en el Centro de Educación Básica Especial N° 09 San Francisco de Asís. De Universidad César Vallejo, Lima, Perú.
[4] González, M. (2017). Autonomía de personas ciegas en supermercados mediante realidad aumentada. De Universidad Internacional de la Rioja, La Rioja, España.
[5] Castillo, J. (2018). Implementación de un dispositivo para monitoreo con GPS y GSM para personas con discapacidad visual De UNIVERSIDAD DE GUAYAQUIL, Guayaquil, Ecuador.
[6] Castellano Alvarez, F., Chavarría Podoliako, P., & Barrientos Padilla, A. (2015). Iris - Mensajería instantánea para personas con ceguera en dispositivos móviles con pantalla táctil. Sinergia E Innovación, 3(1), 42-59.
[7] Stuart J. Russell y Peter Norvig "PEARSON EDUCACIÓN, S.A., Madrid, 2004", INTELIGENCIA ARTIFICIAL. UN ENFOQUE MODERNO, Segunda Edición, pp. 67- 154
[8] Joe Minichino, Joseph Howse, Packt Publishing; Learning OpenCV 3 Computer Vision with Python, Livery Place 35 Livery Street Birmingham B3 2PB, UK, Segunda Edición 2015, pp. 21- 89/sinergia/article/view/407

EDUCACIÓN

Instituto Politécnico Nacional
"La Técnica al Servicio de la Patria"

UNIDAD PROFESIONAL INTERDISCIPLINARIA DE INGENIERÍA CAMPUS HIDALGO

Otorga la presente Constancia a:

CoNMICyT

Marco Rafael Soto Vargas; Eduardo Ramos Díaz y Daniel Tapia Sánchez

Por haber presentado en modalidad de póster el tema:

"Dispositivo portátil de generación de rutas de tránsito para personas con discapacidad visual"

En el marco del evento académico "Congreso Nacional Multidisciplinario de Innovación, Ciencia y Tecnología" que se efectuó en la Unidad Profesional Interdisciplinaria de Ingeniería Campus Hidalgo los días 4, 5 y 6 de octubre 2023.

"La Técnica al Servicio de la Patria"

Dr. Gilberto Alejandro García Guerra
Director Interino

San Agustín Tlaxiaca, Hidalgo, 06 de Octubre de 2023