Propuesta de diseño para un sistema embebido de reconocimiento de terreno del grupo C

Autores:

Brian Cordero Matamoros

George Briceño Celestino

Josué Quirós Barrantes

Docente:

Dr. Johan Carvajal Godínez

11 de octubre del2024

Instituto Tecnológico de Costa Rica

Taller de Sistemas embebidos

II Semestre

1. Introducción y justificación

En la sociedad tecnológica en que vivimos observamos un crecimiento acelerado de la cantidad de dispositivos electrónicos que procesan los datos que recolectan de su entorno para su funcionamiento, como por ejemplo sensores IoT, cámaras en ciudades inteligentes o infraestructura informática. Cada vez es más estandarizado que estos dispositivos dependan de una nube donde procesar la información recopilada, lo que plantea desafíos en términos de privacidad, seguridad y eficiencia energética. Si bien esto permite un análisis más complejo y detallado gracias a la gran capacidad de procesamiento remoto, introduce latencias significativas debido a la transmisión de grandes volúmenes de datos entre el dispositivo y los servidores.

Este retraso es crítico en situaciones que requieren respuestas inmediatas, como en la exploración de terrenos desconocidos o cambiantes. Además de depender de una conexión estable a la nube puede ser inviable en áreas con conectividad limitada o en misiones autónomas que operan en lugares remotos. También tenemos el escenario en que si no damos cierta autonomía este tipo de dispositivos que dependen de procesamiento externo, la cantidad de datos generados por estos dispositivos llegará a un punto de saturación en el envió y respuesta de las comunicaciones. Por estas razones, la necesidad de ejecutar algoritmos de inteligencia artificial (IA) de forma local, utilizando la computación en el borde (EdgeAI), cobra cada vez más importancia.

La computación de borde se refiere al procesamiento, análisis y almacenamiento de datos más cerca de donde se generan y para no depender del envío de datos a servidores remotos, permitiendo el análisis y respuesta más rápida en tiempo real. Esto ha impulsado en el desarrollo de sistemas embebidos para el reconocimiento de terreno en tiempo real como robots autónomos, drones y vehículos de exploración, donde la identificación precisa de obstáculos, caminos y otros elementos del terreno es esencial para una navegación segura y eficiente.

Por ejemplo, los drones autónomos impulsados por la inteligencia artificial están revolucionando diversos sectores. Gracias a la capacidad que tienen en realizar tareas complejas de manera autónoma, se están convertido en herramientas indispensables en áreas como la vigilancia, la búsqueda y rescate, la agricultura y la defensa militar. Los drones equipados con IA pueden procesar grandes cantidades de datos en tiempo real, tomar decisiones inteligentes y adaptarse a entornos cambiantes, todo ello sin intervención humana. Ha resultado muy útil implementar IA en este tipo de sistemas embebidos debido a las ventajas que significa como la optimización de recursos, la disminución de riesgo para las personas y la ayuda en la

toma de decisiones por la información proporcionada, todo esto en el campo del mapeo y análisis territorial.

El procesamiento en dispositivos de borde permite reducir significativamente la latencia y aumentar la eficiencia en un sistema de reconocimiento embebido, pues este necesita tiempos de respuesta y análisis congruentes a las operaciones que se realizarán. También permite dar capas de seguridad y privacidad, al evitar el envío de datos sensibles a la nube. Esto también optimiza el uso del ancho de banda y habilita una toma de decisiones en tiempo real, crítica para la operación de un 'rover' autónomo que debe adaptarse a su entorno y navegar con seguridad. Este proyecto busca abordar estos desafíos mediante el diseño de un sistema embebido que procese localmente los datos capturados por el rover, eliminando las limitaciones del procesamiento en la nube.

Sin embargo, la implementación de la computación de borde no significa que el dispositivo deba operar completamente aislado, pues al reducir la carga en el envío de procesamiento de datos permite usar un método de monitoreo y comunicación con el dispositivo sin saturar el ancho de banda disponible. Los sistemas embebidos proveen funciones especializadas a sistemas complejos, para así conformar un sistema robusto con lo mejor de cada campo. Si observamos el diseño de sistemas empotrados en el sector militar y aeroespacial, los vehículos no tripulados están preparados para funcionar dirigidos por el operador o de manera autónoma, pero siempre es necesario un centro de monitoreo y un sistema de localización para ubicar el vehículo en todo momento.

Otro aspecto a tener en cuenta es que los sistemas embebidos también presentan algunas desventajas. En muchos casos, resulta complicado realizar modificaciones tanto en el hardware como en el software una vez que el sistema ha sido diseñado y probado. Aunque la actualización remota del software puede ser una opción, no siempre está disponible o es factible. Además, según la aplicación, el hardware puede necesitar una considerable cantidad de potencia para asumir los cálculos y procesos que, en comparación con el procesamiento en la nube, pueden verse limitados por la disponibilidad de energía en el dispositivo

Por lo tanto, el uso de la computación de borde enfocado a el procesamiento y análisis de terreno en tiempo real, complementado con la nube para la monitorización nos ayudará a desarrollarla propuesta de diseño de nuestro proyecto.

2. Descripción y síntesis del problema

El problema abordado aquí gira entorno a la creciente dependencia de dispositivos electrónicos que recolectan y procesan datos para su funcionamiento, pues es común que estos dispositivos dependan de la nube para el procesamiento de información, lo que plantea los desafíos críticos:

Latencia y Conectividad: La transmisión de grandes volúmenes de datos a servidores remotos introduce retrasos considerables en la comunicación y la toma de decisiones, siendo muy problemático en situaciones que requieren respuestas inmediatas, como la exploración de terrenos complejos, operaciones militares, y rescates. Además, la conectividad a la nube puede ser limitada o inviable en entornos remotos.

Privacidad y Seguridad: Enviar datos sensibles a la nube plantea riesgos de seguridad y privacidad. La comunicación y toma de decisiones en el dispositivos estaría vulnerada por diversa cantidad de ataques presentes constantemente en internet.

Eficiencia Energética y Ancho de Banda: La transmisión constante de datos aumenta el consumo de energía y el uso de ancho de banda, lo cual puede llevar a la saturación de redes y comunicaciones.

Para superar estas limitaciones, la computación en el borde (EdgeAI) se presenta como una solución eficaz, junto con el procesamiento local, impulsado por inteligencia artificial en sistemas empotrados, sin excluir la comunicación con la nube, reduce la carga en la transmisión de datos y facilita un enfoque híbrido. Esto permite utilizar la nube para el monitoreo y análisis a largo plazo sin comprometer la latencia o saturar la capacidad de la red.

3. Gestión de los requerimientos

A continuación, se detallan los requerimientos del sistema en base al problema planteado y lo requerido en el enunciado para el desarrollo del proyecto.

- Algoritmo de detección y clasificación de terreno: La detección y clasificación de objetos es primordial, pues son los datos que serán procesados. Este algoritmo debe ser preciso y rápido.
- Procesamiento de imágenes en tiempo real: se debe inferir un tiempo de latencia máxima para que el sistema permita la exploración segura y un tiempo prudente de respuesta ante obstáculo

- Monitoreo en tiempo real que permita la visualización de las detecciones y el estado del físico del sistema
- Hardware para la ejecución local del código: es importante un hardware con un poder considerable de procesamiento capaz de ejecutar tanto la detección de imágenes, procesarlas, y gestionar periféricos propios del sistema
- El sistema debe funcionar de manera continua y autónoma mientras el rover está en movimiento, procesando imágenes de video en tiempo real en paralelo a funciones secundarias.
- El sistema debe integrar una cámara para proporcionar información al algoritmo de clasificación. Esta debe ser compatible con el resto del hardware y software requerido
- El sistema debe ser capaz de ajustar el algoritmo de clasificación en función de las condiciones cambiantes del terreno, optimizando el rendimiento bajo diferentes entornos
- El sistema debe mostrar visualmente los objetos detectados en las imágenes mediante recuadros y etiquetas que identifiquen los distintos tipos de terreno u obstáculos
- Interfaz de configuración sencilla para que los operadores del rover puedan ajustar parámetros del sistema, como la sensibilidad del reconocimiento de objetos o la frecuencia de captura de imágenes.
- Compatibilidad del sistema operativo del harware con software como OpenCV y Yocto Proyect: El sistema debe integrarse completamente con el flujo de trabajo de Yocto Project y de OpenCV, asegurando que todas las dependencias de software estén adecuadamente incluidas en la imagen del sistema operativo

Como añadido, requerimientos que no cumplen una función específica, pero son importantes tenerlos en cuenta:

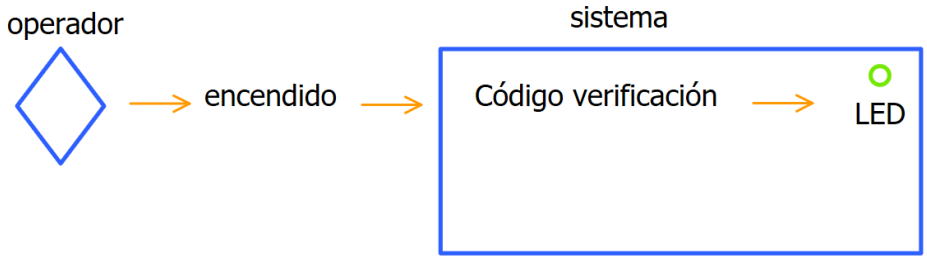
- El sistema debe operar con un consumo energético óptimo para maximizar la duración de la energía durante misiones de larga duración en áreas remotas.
- Debe hacer uso adecuado de los recursos del sistema y la capacidad de procesamiento del hardware embebido, garantizando que las tareas de reconocimiento de terreno no sobrecarguen el dispositivo

- El sistema debe ser robusto y capaz de manejar interrupciones temporales en la captura de imágenes o fallos de sensores sin comprometer la seguridad del rover.
- De ser posible el sistema debe permitir la modularidad si se necesita agregar sensores a parte de la cámara
- Compatibilidad con Raspberry Pi/Jetson Nano

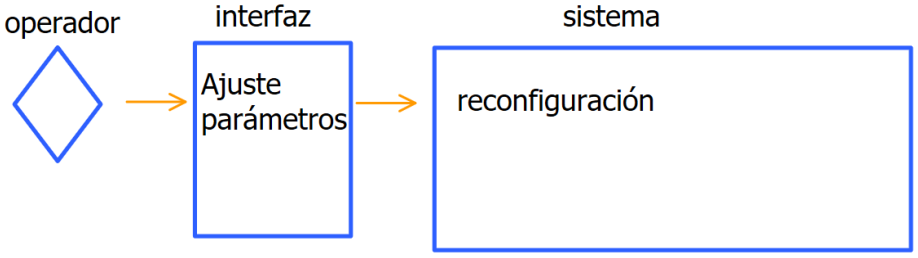
4. Vista operacional del sistema

En este apartado se describe las operaciones del sistema desde la perspectiva del usuario. Estas operaciones son autónomas y en tiempo real, por lo cual el sistema debe responder y resolver por sí solo la identificación y clasificación del terreno y obstáculos (no solo piedras.

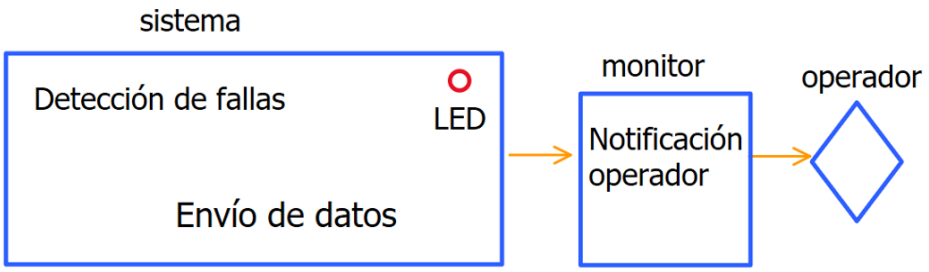
- Al encender, el sistema debe verificar los componentes del sistema, que todos estén activos para el correcto desempeño del sistema sin intervención del usuario. De ser posible retorna mediante un indicador (LED) la correcta inicialización del sistema



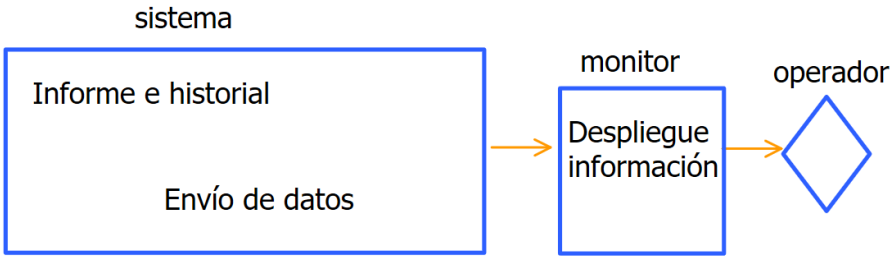
- Acceso a una interfaz del sistema(mediante PC) para ajustar parámetros como frecuencia de detección de imágenes. Cuando el rover se mueva, no es necesario intervención del usuario a menos que se necesite cambiar parámetros



- En caso de fallas del sistema que no pueda gestionar, avisar al usuario mediante la interfaz o un indicador de hardware.



- Al terminar la misión el sistema se apaga automáticamente para que el usuario tenga acceso a un informa tipo historial con todo lo detectado y acciones realizadas



5. Vista funcional del sistema

La descomposición de las funciones clave del sistema es una manera de especificar, en la percepción del desarrollador, lo que hace cada componente para cumplir con los requerimientos.

- Modulo interfaz de parámetros para ajustes, permite al usuario ajustar los parámetros del sistema.
- Modulo para capturar imágenes, mediante una cámara compatible con el hardware, esta envía los datos de imagen al módulo de procesamiento. La frecuencia de captura se debe poder configurar
- Módulo de preprocesado para alistar las imágenes (como ajustar el tamaño y formato) antes de ingresarlas al módulo de clasificación de objetos, la salida se conecta a este módulo
- Modulo clasificación de objetos que es el motor de inferencia (Yolo), detecta el terreno y rocas, la salida son las propias imágenes con los recuadros de detección
- Modulo o interfaz de monitoreo que proyecte lo que el sistema esté gestionando para que los operadores puedan ver el estado del rover y las detecciones que realiza.
- Módulo de almacenamiento y registro local que guarda la información importante, las detecciones realizadas y cualquier evento relevante durante la misión.

6. Arquitectura del sistema propuesto

Para crear la arquitectura del sistema propuesto que mapee las funciones e interfaces en componentes de software y hardware, es esencial considerar tanto los componentes físicos (hardware) como los lógicos (software), ya que ambos trabajan juntos para cumplir con los requerimientos del sistema.

Componentes de Hardware:

a) Procesador Embebido (CPU/GPU): Este ejecuta las operaciones de procesamiento de imágenes y la inferencia de la IA

b) Cámara: Captura las imágenes del entorno real

c) Almacenamiento (SSD): Almacena datos como imágenes, eventos detectados e informes de registro

d) Conexión a red (wifi o ethernet): es el medio de comunicación para la monitorización del sistema.

Componente de Software:

a) Sistema operativo (por Yocto Project): Proporciona un entorno operativo minimalista para las demás aplicaciones de software

b) Biblioteca de procesamiento de imágenes (OpenCV): Maneja la captura y preprocesamiento de imágenes ajustando el formato y otros parámetros

c) Algoritmo de detección de terreno (Yolo): detecta en la imágenes capturadas y preprocesadas por OpenCV los objetos en tiempo real

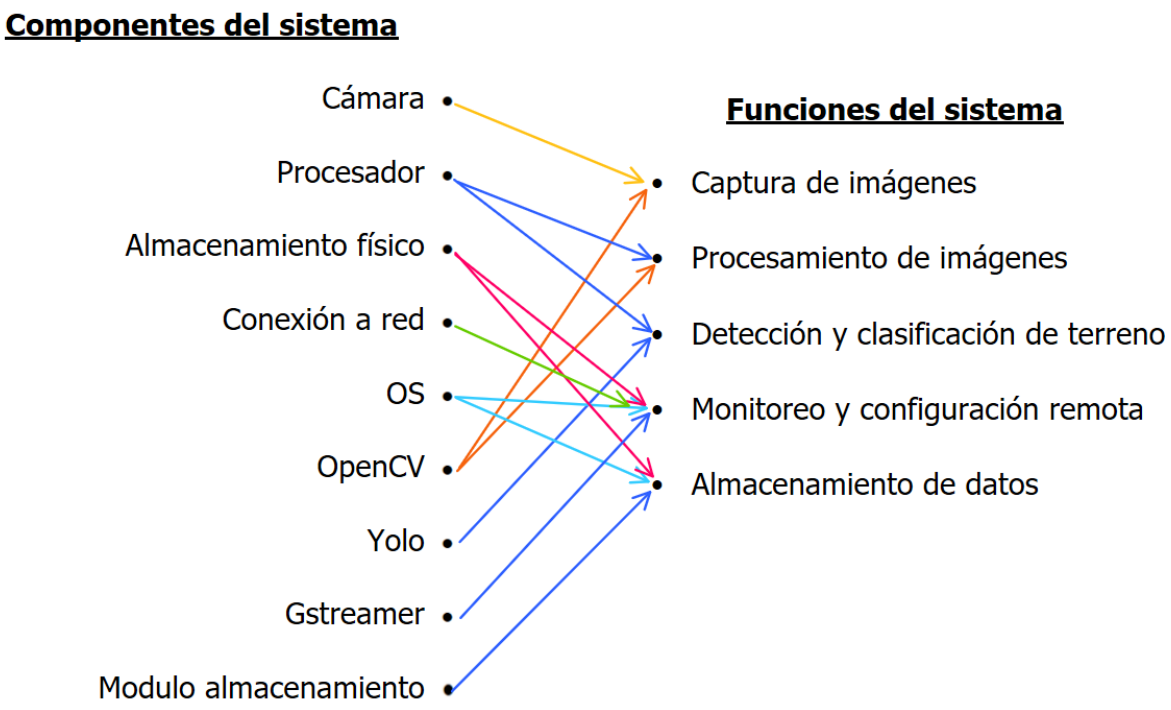
d) Interfaz de monitoreo (GStreamer): es el algoritmo de comunicación para la monitorización del sistema.

f) Modulo para almacenamiento de dato: Se trata de un gestor dentro del sistema operativo que almacena y exporta los registros generados por Yolo.

- Para la captura de imágenes:
    - hardware -> cámara
    - software -> opencv
- Para el procesamiento de imágenes
    - hardware -> cpu/gpu del dispositivo a operar
    - software -> opencv y yocto project
- Clasificación y deteccion de objetos
    - hardware -> cpu/gpu del dispositivo a operar
    - software -> yolo y sus dependencias y TensorRT en caso de Jetson nano

- Monitoreo
    - hardware -> wifi o ethernet
    - software -> GUI y protocolos de transmisión de datos
- Almacenamiento y registro
    - hardware -> ssd
    - software -> gestor de datos Linux y log de eventos

A continuación, el diagrama de mapeo de funciones a componentes



7. Análisis de dependencias

- Yocto Project (Poky) es una plataforma para crear sistemas Linux personalizados en dispositivos embebidos, ofreciendo flexibilidad y control sobre el sistema, desde el kernel hasta las aplicaciones.

- meta-openembbebed: layer contiene meta-python y meta-oe
    - Python 3 es un lenguaje de programación de alto nivel, versátil y fácil de aprender, conocido por su sintaxis clara. Soporta programación orientada a objetos y cuenta con una extensa biblioteca estándar.
        - numpy proporciona soporte para arreglos (arrays) y matrices multidimensionales.
    - meta-oe

        - OpenCV es una biblioteca de código abierto para procesamiento de imágenes y visión por computadora, utilizada para tareas como detección de objetos y seguimiento de movimiento. Es compatible con varios lenguajes y se enfoca en eficiencia y rendimiento.

- Ultralytics es conocida por desarrollar YOLO, una serie de modelos eficientes para la detección de objetos en tiempo real. Soportando detección, clasificación y segmentación. Ofrecen una API amigable para entrenar y desplegar modelos en varios formatos, esto último de vital importancia para condicionar el modelo a detectar rocas.

8. Estrategia de integración de la solución

El flujo de trabajo será el siguiente: la Raspberry Pi capturará el video a través de la cámara utilizando OpenCV. Python justo con Edge AI será el motor detrás de este proceso, ejecutando un script que se encargará de obtener el flujo de video en tiempo real. Este video luego será transmitido a través de la red usando GStreamer, que enviará el flujo a través del WiFi.

Hardware

La base de la solución será una Raspberry Pi, elegida por su capacidad para manejar una cámara y conectarse a una red WiFi. Se utilizará una cámara compatible con la Raspberry Pi (como el módulo de cámara oficial), que proporcionará la entrada de video para procesamiento y transmisión. Además, la Raspberry Pi debe tener un módulo WiFi integrado o, en su defecto, un adaptador WiFi, para transmitir el video capturado a un servidor o una computadora.

Software

Para que la Raspberry Pi pueda manejar la cámara y el WiFi, será necesario que la imagen de Yocto incluya:

- Controladores y módulos necesarios para el hardware de la Raspberry Pi, como los controladores de la cámara (V4L2) y los controladores de red para el WiFi.
- Python como lenguaje base, ya que será utilizado para desarrollar los scripts que capturarán y procesarán el video.

- OpenCV, que se encargará de capturar la imagen desde la cámara y permitir la manipulación del video si es necesario (como ajustes de resolución, formato o análisis en tiempo real).
- GStreamer, una herramienta clave para la transmisión en tiempo real de video, lo que permitirá enviar lo que ve la cámara a través de la red mediante protocolos como RTSP o HTTP.
- Framework de Edge AI, se puede integrar un framework ligero optimizado para el modelo en formato ONNX. Esto permitirá ejecutar el modelo utilizado en detección de objetos o clasificación de imágenes.

9. planteamiento de la ejecución

Actividades	Asignado	Estado	29.09.2024	30.09.2024	01.10.2024	02.10.2024	03.10.2024	04.10.2024	05.10.2024	06.10.2024	07.10.2024	08.10.2024	09.10.2024	10.10.2024	11.10.2024	12.10.2024	13.10.2024	14.10.2024	15.10.2024	16.10.2024	17.10.2024	18.10.2024	19.10.2024	20.10.2024	21.10.2024	22.10.2024	23.10.2024	24.10.2024	25.10.2024	26.10.2024	27.10.2024	28.10.2024
Etapa 1, Diseño e investigación																																
Reunión inicial	Todos	Terminado																														
Investigación Open CV, Yolo y Yocto Proyect	Todos	Terminado																														
Definición de requisitos del sistema	Todos	Terminado																														
Investigación vialidad del motor de inferecia	Todos	Terminado																														
Investigación vialidad de dataset para entrenamiento	Brian	Terminado																														
Entrenamiento detección de rocas y terreno	Josue	Terminado																														
Trabajo en la propuesta de diseño	Todos	Terminado																														
Investigación inclusión de periféricos en Raspberry/Jetson	Brian	En progreso																														
Creación de repositorio	George	En progreso																														
Documentación individual	Todos	En progreso																														
Entrega propuesta de diseño	Todos	Terminado																														
Etapa 2 -Integración a Yocto Proyect																																
Prueba de detección de rocas y terrreno	Todos	Abierto																														
Creación de receta Yocto para dependencias	Josue	Abierto																														
Integración bibliotecas en la imagen	Brian	Abierto																														
Compliación de imágenes para Raspberry/Jetson	Brian	Abierto																														
Verificación de funcionamiento del SO	George	Abierto																														
Integración de periféricos al hardware	Josue	Abierto																														
Investigación de streamer para monitoreo	George	Abierto																														
Prueba de funcionamiento de sistema	Todos	Abierto																														
Corrección de errores	Todos	Abierto																														
Entrega y demostración final	Todos	Abierto																														

10. Conclusiones

El desarrollo de la propuesta de diseño para el sistema embebido de detección y clasificación de terreo, ha ampliado el panorama de la complejidad del proyecto y su posible ejecución. Se destacan las principales conclusiones obtenidas a lo largo del desarrollo de la propuesta:

a. Definir muy bien las funcionalidades y operaciones teniendo en cuenta los requerimientos deja más claro los objetivos y el flujo de trabajo a seguir. Siendo estos primeramente obtener un algoritmo de detección de rocas y terreno bien entrenado y certero, luego plantear las dependencias que este tiene en cuanto a software, para después sintetizar una imagen del sistema operativo que sea compatible con el hardware que se dispondrá y por último realizar pruebas de funcionamiento con corrección de errores
b. En cuanto al apartado de reconocimiento de objetos en tiempo real, el uso de OpenCV para el preprocesamiento de imágenes y de Yolo optimizado para Raspberry Pi/Jetson Nano puede ser una solución eficaz para lograr lo requerido y el principal reto estará en la implementación de estos 3 elementos
c. La arquitectura del sistema deberá permitir latencias admisibles para que el sistema responda correctamente. Esto junto con el adecuado manejo de los recursos del sistema permitirá el mejor desempeño posible con el hardware disponible. Esto junto a la modularidad del sistema facilitará futuras integraciones de nuevas funcionalidades y componentes
d. La planificación y la estrategia de desarrollo del proyecto es importante plantearla a detalle. Sin embargo, la propuesta de diseño puede requerir ciertas correcciones o añadidos no contemplados en este documento, pero la estructura de organización que se desarrolló permite la adaptación y modificación del plan aqui estipulado.

Bibliografía

Meulen, R. (Octubre, 2018). What Edge Computing Means for Infrastructure and Operations Leaders. Gartner. Recuperado de: https://www.gartner.com/smarterwithgartner/what-edge-computing-means-for-infrastructure-and-operations-leaders

Intel. ¿Qué es la computación de borde? Intel Latinoamérica. Recuperado de: https://www.intel.la/content/www/xl/es/edge-computing/what-is-edge-computing.html

Cultura científica. (setiembre, 2018). Los sistemas embebidos en tiempo real están presentes en nuestra vida cotidiana y lo estarán cada vez más. Universidad de Cantabria. Recuperado de: https://web.unican.es/noticias/Paginas/2021/septiembre_2021/Harbour-digitalizacion-sistemas-embebidos.aspx

Zachariah, P. (octubre, 2020). Diseñar aplicaciones de sistemas embebidos en electrónica aeroespacial y militar. Altium. Recuperado de: https://resources.altium.com/es/p/designing-for-embedded-system-applications-in-military-and-aerospace

Mera, Henry. (febrero, 2022). ¿Cuál es la función de los sistemas embebidos en el IoT?. TodoMaker. Recuperado de: https://todomaker.com/blog/cual-es-la-funcion-de-los-sistemas-embebidos-en-el-iot/

Visionplatform. (mayo, 2024). IA y drones: Avances y aplicaciones en vehículos aéreos no tripulados. Visionplatform Recuperado de: https://visionplatform.ai/es/ia-y-drones-avances-y-aplicaciones-en-vehiculos-aereos-no-tripulados/

Rodriguez, E. Inteligencia artificial en la geografía: Mapeo y análisis territorial. Canal Innova. Recuperado de: https://canalinnova.com/inteligencia-artificial-en-la-geografia-mapeo-y-analisis-territorial/

Franco, G. (febrero, 2021). 5 riesgos en la nube y como gestionarlos. Netdata. Recuperado de: https://blog.netdatanetworks.com/5-riesgos-en-la-nube-y-como-gestionarlos

Alegría, G. (2017). Análisis comparativo entre arquitecturas sin servidor y arquitecturas orientadas a servicios aplicado a computación en la nube provista por AWS. Universidad de los Andes. Recuperado de:

https://repositorio.uniandes.edu.co/entities/publication/7ba7b1b0-4595-4b5d-baf4-69224e8c6228

DellÓso, M; Medina, S; Romero, F; De Giusti,A. (octubre 2017). Análisis de performance y consumo sobre sistemas embebidos multinúcleo. Red de Universidades con Carreras en Informática. Recuperado de: https://sedici.unlp.edu.ar/handle/10915/63875