UNIVERSIDAD NACIONAL DEL ALTIPLANO

FACULTAD DE INGENIERÍA MECÁNICA ELÉCTRICA, ELECTRÓNICA Y SISTEMAS

ESCUELA PROFESIONAL DE INGENIERÍA MECÁNICA ELÉCTRICA

DESARROLLO DE UN SISTEMA DE ALERTA DE AUSENCIA DE CASCO DE SEGURIDAD USANDO REDES NEURONALES Y TECNOLOGÍA IOT

TESIS

PRESENTADA POR:

JESUS GABRIEL ARROYO PECCHI

PARA OPTAR EL TÍTULO PROFESIONAL DE:

INGENIERO MECÁNICO ELECTRICISTA

PUNO – PERÚ

2024

Reporte de similitud

NOMBRE DEL TRABAJO

DESARROLLO DE UN SISTEMA DE ALERTA DE AUSENCIA DE CASCO DE SEGURIDAD USANDO REDES NEURONALES Y TECNOLOGÍA IOT

AUTOR

JESUS GABRIEL ARROYO PECCHI

RECUENTO DE PALABRAS

15415 Words

RECUENTO DE CARACTERES

86813 Characters

RECUENTO DE PÁGINAS

104 Pages

TAMAÑO DEL ARCHIVO

9.2MB

FECHA DE ENTREGA

Aug 23, 2024 11:53 AM GMT-5

FECHA DEL INFORME

Aug 23, 2024 11:55 AM GMT-5

11% de similitud general

El total combinado de todas las coincidencias, incluidas las fuentes superpuestas, para cada base de datos.

- 7% Base de datos de Internet
- 2% Base de datos de publicaciones
- Base de datos de Crossref
- Base de datos de contenido publicado de Crossref
- 8% Base de datos de trabajos entregados

Excluir del Reporte de Similitud

- Material bibliográfico

JOSÉ MANUEL RAMOS QUIPA
ING MECANICO ELECTRICISTA
CIP 78419

M.Sc. Felipe Condori Chambilla
SUBDIRECTOR DE INVESTIGACIÓN
EPIME

DEDICATORIA

Dedico el presente trabajo a mi mamá Zulema y a mis abuelos Antonio y Elba por haberme acompañado en este largo camino, por su incondicional apoyo mostrado y por haber confiado en mí para lograr este importante hito.

Gariel Arroyo Pecchi

AGRADECIMIENTOS

A mi querida Universidad Nacional del Altiplano, por haberme dado una sólida formación académica competitiva y haber puesto a disposición todos los recursos necesarios para tal fin. A la Escuela Profesional de Ingeniería Mecánica Eléctrica, que fue directa- mente la encargada de mi formación profesional a través de sus docentes y personal administrativo a quienes es necesario hacer una mención especial por los conocimientos técnicos y habilidades impartidas durante toda esta etapa. A mi mamá, que estuvo ahí incondicionalmente en cada etapa del proceso y a quien espero hacer sentir orgullosa. A mis abuelos Antonio y Elba por su apoyo completo y dedicación conmigo, estando presentes siempre en cada día largo de estudio.

Gariel Arroyo Pecchi

ÍNDICE DE GENERAL

CAPÍTULO III

MATERIALES Y MÉTODOS

Área: Control de Procesos

Tema: Inteligencia artificial - IoT

FECHA DE SUSTENTACIÓN: 16 de Septiembre del 2024

ÍNDICE DE FIGURAS

Pág.

ACRÓNIMOS

5G:	De quinta generación
ADC:	Analog to Digital Converter
APN:	Access Point Name
ARM:	Advanced Risc Machine
AWS:	Amazon Web Services
CNN:	Convolutional Neural Network
EPIME:	Escuela Profesional de Ingeniería Mecánica Eléctrica
EPPs:	Elementos de protección Personal
Gb:	Giga Byte(s)
GCP:	Google Cloud Platform
GPRS:	General Packet Radio Service
GSM:	Global System for Mobile Communication
HAT:	Hardware Attached onTop
HMI:	Human to Machine Interface
HTTP:	Hypertext Transfer Protocol
I2C:	Integrated to Integrated Circuit
IoT:	Internet of things
PCB:	Printed Circuit Board
PLC:	Programmable Logic Controllers
PPP:	Point to Point Protocol
RAM:	RandomAccessMemory
RELU:	Rectified Linear Unit
Rpi:	Raspberry pi
SoC:	System onaChip

SPI:	Serial Peripheral Interface
TTL:	Transistor to Transistor Logic
UART:	Universal Asynchronous Receiver Transmitter
UNA:	Universidad Nacional del Altiplano
USB:	Universal Serial Bus
VPN:	Virtual Private Network

RESUMEN

La presente investigación se llevó a cabo en 2 ciudades distintas, Puno y Lima, durante los meses de agosto y septiembre del año 2023. El objetivo de la presente investigación es el de poder determinar si la precisión del sistema diseñado alcanza al menos el 60 % del total de las muestras, de modo que pueda así considerarse funcional hasta cierto punto, no pretende alcanzar el 100 % de precisión ya que lo que se busca es poder exponer con el mayor detalle posible el proceso de diseño de un sistema funcional con estas características que pueda servir de guía para posteriores y más avanzadas investigaciones en el campo. Las pruebas y toma de muestras se realizaron en paralelo en ambas ciudades donde se aplicó el prototipo, tanto en el laboratorio de control de procesos de la EPIME de la UNA como en el evento "Raspberry Jam Lima 2023", llevado a cabo el día 12 de agosto del 2023. Respecto a la metodología aplicada, se utiliza una red neuronal pre entrenada como base del diseño, y se aplica dentro de la Rpi para lograr un sistema embebido, añadiéndole la funcionalidad IoT, el análisis de permutaciones de clases de detección, la base de datos alojada en Google Firebase, y finalmente se diseñó una aplicación para sistemas Android a modo de interfaz para poder interactuar con las imágenes procesadas. De este modo se lograron obtener muestras en la base de datos, cuya organización es exclusivamente por usuario, siendo éstas analizadas individualmente mediante un análisis de validación visual clasificándolas como: Falso positivo, Falso negativo y correctas. Los resultados obtenidos fueron muy alentadores, analíticamente bastante superiores a la precisión esperada del 60 %, pudiendo afirmar que el prototipo es funcional y que se logró el objetivo principal.

Palabras Clave: Red neuronal, Raspberry Pi, Sistemas Embebidos, Internet de las Cosas.

ABSTRACT

The present research was conducted in two different cities, Puno and Lima, during the months of August and September 2023. The objective of this research is to determine whether the accuracy of the designed system is at least of 60 % of the total samples so that it can be considered functional to some extent. It does not aim to achieve 100 % accuracy since the goal is to detail the design process of a functional system with these characteristics, which can serve as a guide for subsequent and more advanced research in the field. The tests and sample collection were carried out in parallel in both cities where the prototype was applied, both in the process control laboratory of the Escuela Profesional de Ingeniería Mecánica Eléctrica (EPIME) at the Universidad Nacional del Altiplano (UNA) and at the "Raspberry Jam Lima 2023" event held on August 12, 2023. Regarding the applied methodology, a pre-trained neural network was used as the basis for the design, and it was implemented within the Raspberry Pi to achieve an embedded system with Internet of Things (IoT) functionality. This included the analysis of detection class permutations, a database hosted on Google Firebase, and finally, an ap- plication for Android systems was designed as an interface to interact with the processed images. In this way, samples were obtained in the database, organized exclusively by user, and analyzed individually through a visual validation analysis classifying them as: False Positive, False Negative, and Correct. The results obtained were very encouraging, analytically far superior to the expected accuracy of 60 %, allowing us to affirm that the prototype is functional and that the main objective was achieved.

Keywords: Neural network, Raspberry Pi, Embedded Systems, Internet of Things.

CAPÍTULO I

INTRODUCCIÓN

En los últimos 100 años, especialmente desde la década de los 40, con la aparición de las primeras computadoras, y aún más especialmente desde la invención del transistor dentro de esta misma década, hemos sido testigos de una revolución importantísima respecto a la forma en que se construyen y operan las máquinas y la forma en que se automatizan tareas, esto gracias al desarrollo de teorías de control como el control discreto, control moderno, robusto, etc. Dentro de estas teorías, existe una particularmente compleja y útil, el control no lineal, cuyas técnicas nos permiten simplificar y controlar procesos que antes eran imposibles en la totalidad de sus rangos de operación y sin la necesidad de recurrir a técnicas de linealización en puntos de equilibrio que sólo permiten el control alrededor de los mismos.

Una de las técnicas de control no lineal más extendida en los últimos años es la de las redes neuronales, cuya popularización se debe a la capacidad de cómputo de los sistemas actuales y técnicas indirectas de procesamiento como el procesamiento basado en la nube, posible gracias al avance en nuestra red de telecomunicaciones que hace también posible la interconexión absoluta de técnicamente cualquier cosa, dando paso al IoT, como nos indican Shafique et al. (2020) "...La tasa de datos más alta posible con el 5G hace posible la implementación de algoritmos de inteligencia artificial con alta demanda de datos y computación intensiva para varias aplicaciones de usuarios." Avances como estos se ven reflejados en nuestros sistemas de control contemporáneos con PLC que ahora admiten funciones embebidas de inteligencia artificial tanto para análisis de datos como para la implementación de algoritmos de control automático.

Es así, que, gracias al concepto de la interconexión absoluta, y a la capacidad de las máquinas modernas de implementar algoritmos de inteligencia artificial y de la red de telecomunicaciones de poder soportar el envío y recepción de datos en tiempo real nace la idea de la presente investigación. Si bien no se usa la red 5G en el prototipo diseñado, sino más bien la red GPRS, es una aproximación interesante de lo que podría ser extrapolado en la red 5G un sistema IoT embebido multipropósito con inteligencia artificial que trabaja en tiempo real.

Tomando como base a la ley N°29783, ley de seguridad y salud en el trabajo, publicada el 20 de agosto del 2011, cuyo reglamento es el decreto supremo N°005-2012, y cuya modificación es la ley N°30222, publicada el 11 de julio del 2014, donde se establece en el artículo 21, inciso e que: "...En último caso, facilitar equipos de protección personal adecuados, asegurándose que los trabajadores los utilicen y conserven de forma correcta..es que se pensó la posible aplicación de la presente investigación, logrando obtener un muy buen número de reportes en tiempo real según los criterios establecidos, tal como se esperaba.

1.1. OBJETIVO PRINCIPAL

Se consideró como objetivo principal el desarrollar un prototipo funcional del sistema descrito con una precisión de al menos un 60 % del total de imágenes procesadas y enviadas a la base de datos.

1.2. OBJETIVOS ESPECÍFICOS

Se consideran los siguientes objetivos específicos:

- Proponer una base para un sistema automático de detección aplicable en la industria.

- Demostrar la capacidad de los sistemas embebidos de realizar tareas de computación intensiva localmente.

1.3. HIPÓTESIS DE LA INVESTIGACIÓN

Se formularon las siguientes 2 hipótesis para poder analizar la precisión del sistema:

- Hipótesis nula: El sistema tiene una precisión de al menos el 60 % del total de las muestras.
- Hipótesis alternativa: El sistema tiene una precisión mayor al 60 % esperado.

CAPÍTULO II

REVISIÓN DE LITERATURA

2.1. ANTECEDENTES DE LA INVESTIGACIÓN

La habilidad de las máquinas de aprender, sobre todo de adaptarse a su contexto, imitando el comportamiento inteligente del ser humano no es un tema de estudio nuevo, si bien gracias a los avances en teoría de control, mayor capacidad de cómputo, y a la aplicación de modelos matemáticos mediante álgebra lineal de los últimos años ha logrado perfeccionar este campo, se pueden ver intentos de crear autómatas tan antiguos como de hace 2000 años. Villamarín et al., (2020) por ejemplo, mencionan que "...A través del a historia, el ser humano ha intentado optimizar la realización de sus actividades cotidianas a través de dispositivos y avances tecnológicos, algunos de estos dispositivos llamados autómatas...", y no es muy difícil encontrar varios intentos arcaicos por lograr sistemas automáticos si nos ponemos a revisar un poco de historia. La inteligencia artificial que hoy conocemos es el fruto y la desembocadura de esos más de 2000 años de intentos por crear sistemas automáticos y otros miles de años de desarrollo matemático riguroso.

Un muy ilustrativo ejemplo de la aplicación de la inteligencia artificial en la forma de una red neuronal convolucional CNN es el trabajo de Zhang et al., (2021) que proponen la aplicación de una combinación de una CNN, una CNN gráfica y visión computacional para poder clasificar cáncer de mama a través del análisis de imágenes médicas, lo que permite la automatización del proceso de diagnóstico que implica en muchos casos una detección oportuna del cáncer para poder ser tratado.

Por otro lado, dentro de nuestro campo de estudio, Massiris et al., (2021) nos muestran el desempeño de una CNN entrenada para la detección de EPPs en una obra de construcción civil, logrando demostrar un buen desempeño en la labor de detección. Si bien este trabajo demuestra la aplicabilidad de una CNN en contextos industriales de construcción, no se desarrolla un sistema completo de alerta integrado, lo que pretende el presente trabajo.

En una iteración posterior de su mismo trabajo, Massiris et al., (2021) se implementa el sistema de monitoreo en tiempo real, y ya no solamente la detección a través del análisis de imágenes. Esta red es un candidato muy interesante para posteriores iteraciones de esta investigación.

Dentro de la misma línea de investigación acerca de la detección de EPPs, podemos encontrar el trabajo de Castillo (2020) quien desarrolla un sistema de elección de EPPs no descartables basándose en datos de sensores de gas para determinar a través de otra técnica de inteligencia artificial, un algoritmo difuso, cuál o cuáles son los EPPs adecuados para usar según la mezcla de gases a la que el trabajador estará expuesto. Y así, podemos encontrar diversos trabajos que nos muestran novedosas aplicaciones de la inteligencia artificial en múltiples campos que pueden ir desde la robótica, el arte y la medicina hasta economía e incluso compañía y asistencia.

2.2. MARCO TEÓRICO

2.2.1. Red Neuronal Convolucional

Una red neuronal convolucional es, según una de las definiciones que presenta Guresen y Kayakutlu (2011), “Una combinación masiva de unidades de procesa- miento simples en paralelo que es capaz de adquirir conocimiento de su ambiente a través de un proceso de aprendizaje y de almacenar este conocimiento en sus conexiones”, es decir, un sistema capaz de adaptarse mediante el aprendizaje de su contexto a través de un proceso que podemos interpretar como un proceso auto iterativo.

De Akhtar and Ragavendran (2020) podemos obtener una definición muy importante: "...el kernel es la matriz k ∗ k que se va modificando durante el proceso de aprendizaje...", es decir, podemos interpretar al kernel como los pesos de cada neurona que se corrigen en cada iteración, pero surge la necesidad de preguntar ¿Qué es un kernel? Y ¿Cómo lo definimos?

Según el álgebra lineal, podemos definir al kernel de un espacio o subespacio vectorial generalizado ∀Rn como el conjunto de vectores que forman un subespacio vectorial que al aplicársele una transformación lineal de la forma T (x) = Ax tiene como imagen a 0, definido formalmente:

$$\mu(x) = \{x \in \mathbb{R}^n, Ax = 0\}$$

Podemos interpretar a la matriz de pixeles que forman una imagen como un subespacio vectorial de 3 dimensiones (⊂ R3) que por definición tiene un kernel definido diferente de la solución trivial para cada una de las

transformaciones lineales posibles aplicables al mismo, y cuyo isomorfo, asumiendo un kernel de 3x3, de forma general sería:

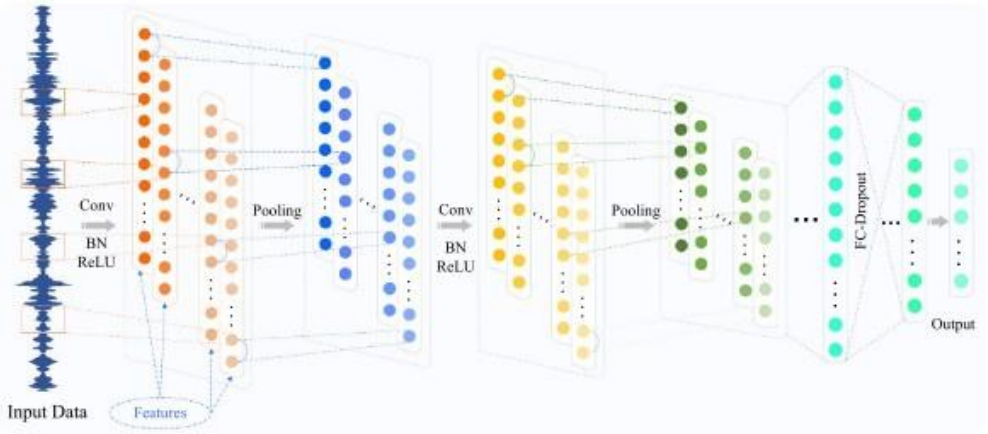
$$\begin{bmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{bmatrix} * [x_1 \quad x_2 \quad x_3] = 0$$

Es inevitable observar que, y aunque parezca trivial es importante, el resultado de apli- car cualquier entrada al kernel de un espacio o subespacio vectorial es siempre constante. Esta observación es importante porque se deduce de aquí que podemos extraer "características" invariantes de una entrada a través de un kernel, entendiéndose un kernel de características distinto al que nos lleva a la nulidad, elemento fundamental en el proceso de reconocimiento y clasificación de imágenes por computadora.

Esto nos lleva intuitivamente al concepto de convolución, que sería el proceso mediante el cual nuestra red neuronal aplica los kernels obtenidos a través del proceso de aprendizaje a la entrada de nuestro sistema, en este caso una imagen, para así poder extraer las características invariantes que definen los objetos sujetos a la operación de clasificación o reconocimiento. Podemos ver gráficamente el proceso mediante la siguiente imagen:

Figura 1

Ejemplo gráfico de una red neuronal convolucional.



Nota: Extraído de: Jiao et al., (2020)

Podemos observar de la figura 1 que existen aún más procesos por definir aparte de la convolución. Teniendo este concepto claro, ya que es el corazón de todo el proceso, seguiremos a continuación con la definición de neurona.

Podemos tomar a una neurona, según la definición tomada de red neuronal, como la unidad de procesamiento simple capaz de adquirir y almacenar conocimiento, o si nos fijamos en una definición más abstracta y matemática, como una función no lineal definida ∀R que procesa las entradas que obtiene de las salidas de cada convolución y permite así establecer relaciones estadísticas durante el proceso de aprendizaje que afinan el modelo modificando los kernels. Existen diversas funciones llamadas funciones de activación que modelan la estadística mencionada, algunas de ellas, según Torres (1994) son:

- Sigmoidales:

Heaviside:

$$f(x) = \begin{cases} 1, x > 0 \\ 0, \forall x \leq 0 \end{cases}$$

Sigmoide o logísitca:

$$f(x) = \frac{1}{1 + e^{-x}}$$

Rampa unidad:

$$f(x) = \begin{cases} 1, x \geq 1 \\ x, 0 \leq x < 1 \\ 0, x < 0 \end{cases}$$

Coseno sigmoidal:

$$f(x) = \begin{cases} 0, -\infty < x \leq -\frac{\pi}{2} \\ \frac{\cos(x + \frac{3\pi}{2})}{2}, -\frac{\pi}{2} \leq x \leq \frac{\pi}{2} \\ 1, \frac{\pi}{2} \leq x \leq \infty \end{cases}$$

Evidentemente el desarrollo ha avanzado bastante, si bien aún estas funciones de activación son ampliamente utilizadas, en especial la función sigmoide, existen funciones que pueden solucionar muchísimo mejor el problema de la identificación y clasificación por visión artificial. Por ejemplo, una función ampliamente utilizada en este tipo de aplicaciones es la función RELU, en el trabajo de Bai (2022) podemos ver la definición de ésta y sus derivadas:

RELU:

$$ReLU(x) = \begin{cases} x, x > 0 \\ 0, x \leq 0 \end{cases}$$

- RELU':

$$ReLu'(x) = \begin{cases} 1, x > 0 \\ 0, x < 0 \end{cases}$$

ELU:

$$ELU(x) = \begin{cases} x, x > 0 \\ \alpha(e^x - 1), x < 0 \end{cases}$$

- Leaky RELU:

$$LRELU(x) = \begin{cases} x, x > 0 \\ \alpha x, x \leq 0 \end{cases}$$

- Leaky RELU':

$$LRELU'(x) = \begin{cases} 1, x > 0 \\ \alpha, x \leq 0 \end{cases}$$

Entre otras, de todas estas, nos fijaremos en RELU, y en su derivada RELU'. Observando la primera, notamos que elimina todos los valores negativos y la segunda transforma a la entrada en 1 y elimina los valores negativos. Se adopta RELU, siempre según Bai (2022), porque: "Cuando introducimos RELU, también introducimos mucha escasez. Esto significa que la matriz activada contiene muchos ceros. Cuando un cierto porcentaje (como el 50 %) de la activación está saturada, llamamos a esta red una Red Neuronal escaza. Esto puede mejorar la eficiencia en términos de tiempo, espacio y complejidad..." Es decir, gracias a RELU, es posible hacer el cómputo necesario de una CNN en dispositivos de una potencia relativamente modesta.

Ahora que tenemos claro que es una neurona y cuál es su función dentro de una CNN, procederemos a definir el proceso de Pooling. Según Murray and Perronnin 2014), el pooling es "La operación que involucra agregar muchas codificaciones de descriptores locales en una sola representación...", es decir, es condensar todas las características ya obtenidas en una sola representación.

Remitiéndonos de nuevo a la figura 1, podemos ver que existen 2 procesos de pooling dentro del proceso general de reconocimiento, uno antes de aplicar la convolución, y uno después, luego de eso se aplican las neuronas que finalmente nos dan el resultado.

Como resumen de esta sección, definimos los agentes involucrados dentro del proceso de reconocimiento y clasificación, y explicamos el funcionamiento de cada uno de ellos, si bien de manera bastante general, es suficiente para poder entender el funcionamiento de la etapa de clasificación del sistema propuesto, de la siguiente manera: La imagen entra como una matriz tridimensional de nxm elementos, es pre procesada mediante una capa de neuronas, luego pasa por un proceso de pooling antes de entrar a una o varias convoluciones, estas convoluciones extraen características constantes que definen el elemento a ser clasificado, y se condensan mediante otro proceso de pooling, seguido de otro procesamiento mediante las neuronas, todas con una función de activación RELU, que optimiza el proceso y nos da finalmente un resultado expresado en porcentaje de probabilidad de ser el objeto deseado.

2.2.2. Conectividad IoT

Según Sethi and Sarangi (2017), "El Internet de las cosas se refiere a un nuevo tipo de mundo donde casi todos los dispositivos y electrodomésticos que usamos están conectados a la red...", este es un concepto relativamente nuevo que ha surgido en los últimos años a raíz del desarrollo de nuevas redes de telecomunicaciones cada vez más rápidas y robustas que, gracias a su robustez y capacidad de soportar muy grandes y muy altas tasas de transferencia de datos, nos han permitido interconectar una infinidad de dispositivos con el fin de tener

telemetría y poder actuar remotamente respecto a ellos prácticamente en tiempo real. Podemos ver ejemplos de esto en rastreadores de automóviles, casas inteligentes, telemetría en industrias, control remoto de actuadores, entre otros. En el caso de controles a distancia, se diferencia de los controles remotos originales en su capacidad de ser global, pudiendo utilizar actuadores desde cualquier parte del mundo.

Podemos ilustrarnos mejor a través de la siguiente imagen:

Figura 2

Esquema general de IoT

Nota: Extraído de: Al-Fuqaha et al., (2015)

De Al-Fuqaha et al., (2015) tenemos 2 clasificaciones de dispositivos IoT:

De Recursos Limitados: Aquellos dispositivos capaces de soportar el protocolo TCP/IP, pero no implementar la variedad de otros protocolos.

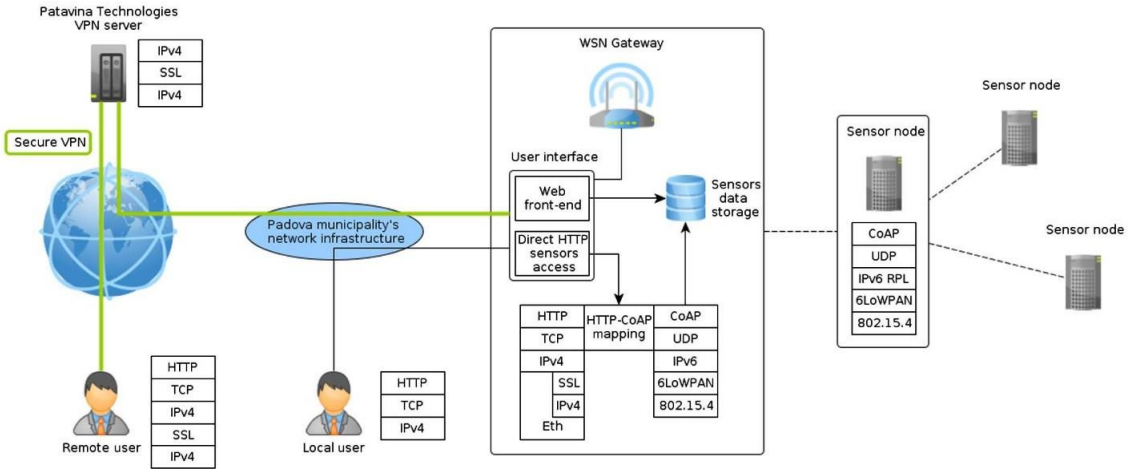
De Recursos Ricos: Aquellos dispositivos cuyo hardware y software son capaces de soportar el protocolo TCP/IP, y que pueden implementar una gran variedad de otros protocolos y marcos de trabajo como: REST, CoAP, MQTT, MQTT-SN, AMQP, etc.

Ambas grandes clasificaciones soy muy importantes, pues si las analizamos, se basan en la versatilidad de los dispositivos que se utilizarán. El protocolo TCP/IP, es el protocolo preferido por la industria junto con la interfaz MODBUS en sistemas interconectados como SCADA, por ser robusto, ser capaz de soportar múltiples esclavos en paralelo y por su alta tasa de velocidad de transferencia de datos por lo que es indispensable que al diseñar un dispositivo IoT se tenga en cuenta que éste debe comunicarse mediante el protocolo TCP/IP.

Una vez que nuestro dispositivo tenga asignada una dirección IP, podemos decir que se ha conectado exitosamente a la red. y es en ese momento en el que podemos comenzar a enviar comandos. Zanella et al., (2014) proponen un ejemplo muy interesante de la aplicación del IoT para una ciudad inteligente, en esta aplicación, podemos encontrar el siguiente esquema de la red:

Figura 3

Infraestructura de ejemplo de una red IoT



Nota: Extraído de: Zanella et al., (2014)

De esta infraestructura se desprenden las siguientes observaciones: Se utiliza una VPN para asegurar la privacidad del usuario, quien se comunica a través de su interfaz mediante un protocolo TCP/IP a través de una solicitud HTTP con el servidor, quien procesa los datos y actúa según las instrucciones del usuario.

Como podemos ver, una red IoT implica muchas capas de abstracción, tanto la programación front-end para el usuario a modo de HMI, tanto con paneles informativos como con botones, la programación back-end para realizar todo el procesamiento de la trama de datos que contiene a las instrucciones bidireccionales (usuario-servidor- servidor-acciones), programación de los nodos para cada una de las acciones, implementación de una base de datos, la implementación de un gateway para dar salida a internet, y la implementación de actuadores o sensores.

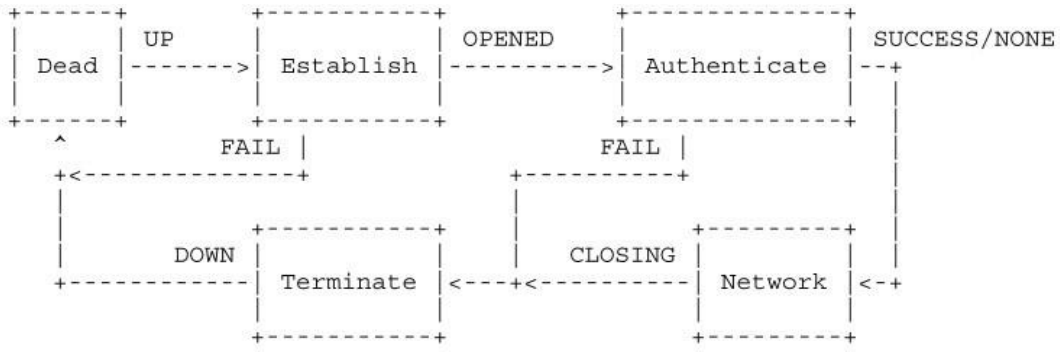
2.2.3. Protocolo PPP

El protocolo PPP es un método por el cual podemos conectarnos a internet mediante el uso de un módem, esto nos permite usar como gateway al mismo, de modo que el sistema que se diseñe tenga la capacidad de conectarse a internet, parte funda- mental de un sistema IoT como vimos en la sección anterior.

De Simpson (1994), tenemos el siguiente gráfico que ilustra el protocolo PPP:

Figura 4

Diagrama de bloques del protocolo PPP



Nota: Extraído de: Simpson (1994)

De 4, tenemos el siguiente flujo:

El protocolo comienza en el estado de muerto, luego de eso se intenta iniciar y se pasa al bloque de establecimiento.

Si el bloque de establecimiento falla, existe un bucle que inicia nuevamente en el estado muerto, de no lograrse un establecimiento, se da por terminado el protocolo.

Si se logró establecer, se procede a abrir un canal que intenta una autenticación, si no se logra autenticar, se procede a dar por terminado el protocolo.

Si se logra autenticar, se procede a conectar a la red, hasta que se determine que se debe cerrar.

Esta comunicación la hace nuestro modem, mediante comandos enviados a través de la Rpi que veremos con mayor detalle más adelante.

2.2.4. Sistemas Embebidos

De acuerdo a Peñaranda et al., (2016) un sistema embebido es “Un sistema electrónico con funcionalidad dedicada construida dentro de su propio hardware y software...", es decir, es un sistema que realiza una tarea muy específica para la cual fue especialmente diseñado y que puede operar de manera autónoma sin la intervención de agentes externos. Por lo general, los sistemas embebidos son muy eficientes gracias a que casi siempre se encuentran construidos en base a una arquitectura ARM, baratos y versátiles, lo que los hace ideales para muchas aplicaciones desde sencillas hasta complejas ejecuciones de inteligencia artificial.

Un ejemplo muy claro de esto es el libro de Parker and Dhanani (2012) quienes hacen un extenso desarrollo de procesamiento digital de video enfocado al diseño de sistemas embebidos. Hacen una observación bastante importante, que la gran mayoría de sistemas embebidos se programan en lenguajes de bajo nivel de abstracción (ensamblador, C), esto debido a los tiempos de ejecución que en muchos casos requieren de una dinámica rápida, cosa que programas ejecutados a través de lenguajes interpretados como Python son capaces de lograr, pero con una mayor capacidad de cómputo del sistema.

Existen varios ejemplos comunes de sistemas embebidos en nuestra vida diaria, como lo son celulares, rastreadores, relojes inteligentes, tabletas y muchas otras más. Y podemos considerar a la Rpi como un sistema embebido por su tamaño compacto, tipo de arquitectura, consumo energético y capacidad de trabajar sin una interfaz.

En cuanto a sus debilidades, Sangiovanni-Vincentelli and Di Natale (2007) hacen un análisis bastante completo de esto, si bien enfocado a la industria automotriz de hace más de 10 años, muchas de estas afirmaciones aún están vigentes para un ingeniero desarrollador de este tipo de sistemas, como, por ejemplo:

Falta de separación entre los modelos funcional y de arquitectura. Esto se puede observar claramente por ejemplo en la manera de compilar un programa para arquitecturas ARM, que debe ser un comando muy específico del compilador para cada uno de los tipos de ARM existentes, y del mismo modo al usar las librerías nativas de éstos. Esto limita la capacidad de explorar el desempeño de distintas arquitecturas para una misma tarea.

Falta la consideración de los retrasos de cálculo y comunicación. En muchos casos, se asumen estas condiciones como 0. sin embargo, para sistemas que re- quieren de una dinámica muy rápida, es importante considerar estos retrasos en el modelo discreto.

Para nuestra aplicación, la dinámica del sistema no necesita ser rápida casi al punto de ser inmediata, por lo que podemos despreciar los retrasos antes mencionados. Y con respecto a la arquitectura, al usar un lenguaje interpretado como Python, es posible que el programa se ejecute en distintas arquitecturas

siempre y cuando las máquinas a usarse tengan el intérprete de Python correctamente instalado.

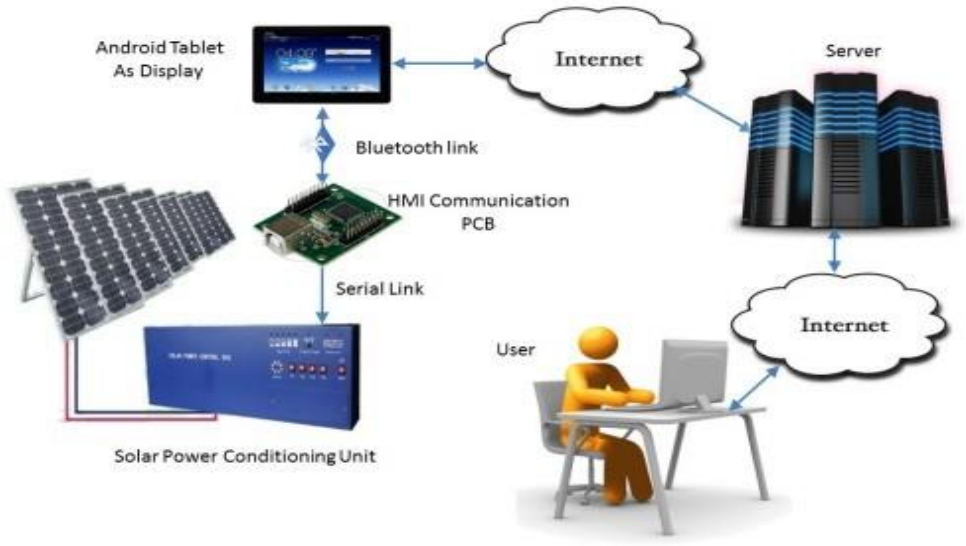
2.2.5. Sistemas Android

Ampliamente conocido, Android es un sistema operativo basado en el kernel de Linux, según Zanin et al., (2021) "...Desde su lanzamiento en 2008, Android ha experimenta- do una evolución significativa, consolidándose como una opción preferida tanto por usuarios como por desarrolladores debido a su naturaleza abierta y personalizable". Centrándonos en esta última afirmación, seleccionamos a un sistema Android como HMI debido justamente a lo fácil que es desarrollar aplicaciones para este tipo de sis- temas, pudiendo utilizar lenguajes nativos casi universales como Java en entornos que simplifican mucho el trabajo como Android Studio o Visual Studio.

La idea de utilizar un sistema Android como interfaz no es nueva, pudiendo ver por ejemplo el trabajo de Jiju et al., (2014) que diseñaron un sistema de monitoreo y con- trol online para fuentes de energía renovables distribuidas a través de la conectividad bluetooth nativa de estos dispositivos. Es un ejemplo muy práctico y que nos ilustra la versatilidad de estos sistemas para propósitos de cualquier tipo, pudiendo usarse en la industria, aplicaciones domésticas, entre otros. A continuación, la arquitectura del sistema de ejemplo:

Figura 5

Arquitectura de un sistema IoT basado en sistemas Android



Nota: Extraído de: Jiju et al., (2014).

La arquitectura del sistema visto en 5 es bastante similar a la que utilizaremos, invirtiendo solamente el orden del servidor y nuestro HMI como veremos más adelante

2.2.6. Raspberry Pi

De acuerdo a Karthikeyan et al., (2023) una Rpi es un SoC, un sistema en un chip, es decir, un sistema que incluye todo lo necesario para funcionar en muy pocos componentes.

La gran ventaja de una Rpi sobre un microcontrolador es que ésta es capaz de levantar un sistema operativo basado en la distribución Debian de Linux. Al día de hoy a distribución más actual es Debian Bookworm, sin embargo, para el desarrollo se utilizará Debian Bullseye.

La Rpi cuenta con 40 pines de propósito general, ideales para salidas tanto analógicas como digitales, no cuenta con un ADC, por lo que si se quieren

monitorear variables analógicas es necesario utilizar un módulo externo. En cuanto a interfaces de comunicación, cuenta con un puerto UART nativo, un puerto I2C, y un puerto SPI, además de contar con WIFI y Bluetooth integrados junto con un puerto Ethernet. Esto la convierten en un dispositivo ideal para poder trabajar con múltiples interfaces en simultáneo y aprovechando toda su capacidad de cómputo, pudiendo observar desarrollos a nivel industrial como el PLC Revolution Pi, basado en este SoC.

Un ejemplo de este uso a nivel industrial y a través de IoT, lo podemos encontrar en el trabajo de Kadiyala et al., (2017) quienes desarrollaron un sistema de monitoreo de procesos utilizando una Rpi, así, es posible como se indicó líneas arriba a través de un ADC externo monitorear variables continuas, muy presentes en la industria general- mente como variables de estado, lo que nos permite a través de un modelamiento en espacio de estados lograr desarrollar un controlador de tipo Ackerman ya sea solo o con observadores discretos.

Al tener un puerto Ethernet embebido y WIFI integrado, una Rpi es capaz de soportar el protocolo TCP/IP, requisito fundamental de un dispositivo IoT como ya se explicó, siendo también capaz de soportar una infinidad de otros protocolos, limitada sola- mente por hardware, como podría ser el caso de MODBUS, interfaz para la cual es necesaria circuitería adicional para poder convertir niveles de tensión y datos a cualquiera de los puertos nativos antes mencionados. Para aplicaciones sencillas que requieran de IoT, es suficiente, a criterio del autor, solamente el uso del puerto UART y el puerto I2C, el primero para la comunicación con el Gateway, y el segundo para interactuar con múltiples sensores en paralelo.

2.2.7. GPRS

El servicio GPRS, introducido en 1990 como parte de una mejora y ampliación de las funcionalidades de las redes GSM. De acuerdo con Walke (2013), el servicio GPRS es "...aún a la fecha el único servicio de paquetes de datos omnipresente a escala global para acceso móvil a internet", tomando en cuenta la diferencia de tiempo entre nuestro presente y la cita antes mencionada, aún es posible afirmar esto, podemos observar claramente en lugares alejados que el único servicio móvil disponible es aún el servicio GPRS.

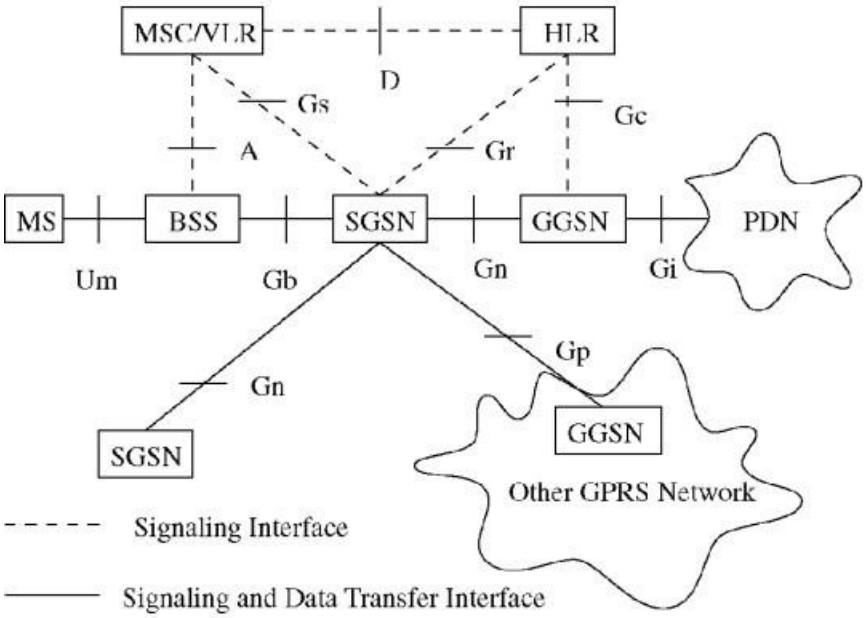
Dentro de nuestro campo de interés, el IoT, podemos ver una comparación y análisis en el trabajo de Vejlgaard et al., (2017) según quienes "...Las cuatro tecnologías compa- radas tienen más del 99 % de cobertura en ambientes abiertos. Mientras que GPRS es incapaz de proveer cobertura en ambientes cerrados para un 40 % de usuarios". De- mostrando así que el servicio GPRS, si bien con baja cobertura en ambientes cerrados, está aún bastante vigente para aplicaciones IoT.

Prueba de ello es el trabajo de Maduranga (2023), quien diseñó un sensor con diseño de nodo para aplicaciones IoT a puertas abiertas, nótese la incidencia en la utilidad de GPRS en ambientes abiertos, justamente por lo visto anteriormente. Maduranga utiliza un microcontrolador ATMEL ATMega328p, este microcontrolador es muy bueno para realizar prototipos, pero no a nivel industrial ya que solamente tiene registros de 8 bits y un ADC integrado de 8 bits, siendo el mínimo requerido en la industria de 12 bits.

Para finalizar esta sección, de manera ilustrativa, veamos un diagrama funcional del servicio:

Figura 6

Arquitectura del servicio GPRS



Nota: Extraído de: Lin and Chlamtac (2001)

De 6, siempre según Lin and Chlamtac (2001) "El Nodo de Soporte GPRS de Pasarela (GGSN) proporciona interoperabilidad con redes de conmutación de paquetes externas y está conectado con los SGSN a través de una red troncal GPRS basada en IP.". El servicio GPRS es capaz de asignar una IP y hacer de gateway hacia internet, suficiente para lograr funcionalidad IoT.

2.2.8. Base de Datos

Según Durango Vanegas (2014), las bases de datos son "...sistemas organizados para la gestión y almacenamiento de información.", sus tipos son, siempre según Du- rango Vanegas (2014):

Bases de Datos Relacionales: Utilizan un modelo basado en tablas para organizar los datos, ideales para aplicaciones que requieren integridad de datos y relaciones complejas entre diferentes conjuntos de éstos. Podemos interpretarlas como estructuras matriciales.

Bases de Datos no Relacionales: Pueden ser bases de datos de documentos, clave-valor, gráficos y columnas anchas, son ideales para aplicaciones que requieran de escalabilidad horizontal. Podemos interpretarlas de diversas formas, como diccionarios, vectores o simplemente listas.

Bases de Datos en la Nube: Servicios proporcionados por plataformas como AWS, GCP y Microsoft Azure, que ofrecen almacenamiento y gestión de datos como servicios escalables y de alta disponibilidad.

Nos centraremos en esta última clase de base de datos, la base de datos en la nube.

2.2.9. Google Firebase

Según Fathulloh et al., (2021) Google Firebase es "Una plataforma de desarrollo de aplicaciones móviles y web que ofrece una variedad de herramientas y servicios.". Esta plataforma ofrece soluciones backend bastante robustas, por lo que es ideal para el desarrollo de sistemas que implican una base de datos, ahorrándonos el trabajo de la programación backend.

De acuerdo a Sung et al., (2022) Google Firebase ofrece las siguientes características:

Base de Datos en Tiempo Real: Permite a las aplicaciones sincronizar datos en tiempo real con todos los clientes conectados, manteniendo la información actualizada sin la necesidad de configuraciones complejas de servidor.

Autenticación: Proporciona servicios de autenticación sencillos y seguros, permitiendo a los usuarios autenticarse utilizando métodos como correo

electrónico y contraseña, proveedores de identidad federada como Google y Facebook, y autenticación anónima.

Mensajería en la Nube (Cloud Messaging): Permite el envío de notificaciones push a dispositivos Android, iOS y aplicaciones web, mejorando la re-engagement y la retención de usuarios.

Almacenamiento: Ofrece almacenamiento de archivos robusto y escalable para aplicaciones que necesitan guardar y servir contenido generado por los usuarios como fotos y videos.

Alojamiento (Hosting): Proporciona alojamiento rápido y seguro para aplicaciones web, con características como el despliegue con un solo comando y la entrega de contenido mediante redes de distribución de contenido.

Configuración Remota (Remote Config): Permite actualizar la configuración de la aplicación de manera dinámica sin necesidad de publicar una nueva versión, facilitando pruebas A/B y ajustes de funcionalidad en tiempo real.

Laboratorio de Pruebas: Proporciona un entorno para realizar pruebas automáticas de aplicaciones en una variedad de dispositivos y configuraciones, asegurando la calidad y la compatibilidad de la aplicación.

Informes de Errores (Crash Reporting): Proporciona informes detallados de fallos que ayudan a los desarrolladores a identificar y corregir errores en sus aplicaciones de manera eficiente.

Este desarrollo hace uso tanto de la autenticación OAuth como de la base de datos en tiempo real que ofrece Google Firebase, logrando así una base de datos accesible desde cualquier parte del mundo para aplicaciones IoT.

CAPÍTULO III

MATERIALES Y MÉTODOS

Teniendo claro todo lo que está detrás del desarrollo del sistema, podemos proceder a explicar detalladamente el funcionamiento del sistema.

3.1. MATERIALES UTILIZADOS

Los materiales utilizados en el desarrollo del sistema fueron los siguientes:

- Raspberry pi 4B+ de 4 Gb de RAM: Corazón del sistema encargada del procesamiento, ejecución del programa y comunicación con el módem.
- SIM800L: Interfaz GPRS a modo de gateway para conectar el sistema a internet, enviar y recibir paquetes.
- PCB: PCB diseñada de tipo HAT que incorpora al módem para minimizar el ta- maño del sistema.
- Cámara USB y Raspberry Pi camera module V2: Para la captura de las imágenes a ser analizadas.
- Plataforma Google Firebase: Para el almacenamiento de datos en tiempo real y autenticación del usuario.

Estos materiales aseguran la eficiencia del sistema en conjunto, proporcionando una solución robusta y escalable.

3.2. DISEÑO DEL SISTEMA DE DETECCIÓN

Para el diseño del sistema de detección se siguieron los siguientes pasos que incluyen el código utilizado:

3.2.1. Configuración de la Red Neuronal

La red neuronal utilizada es una red de tipo convolucional CNN, pre entrenada y probada por Nath et al., (2020) con la siguiente estructura de código en Python:

Figura 7

Cabecera del código de reconocimiento y reporte

```python
# Cabecera del codigo
import cv2
import numpy as np
import matplotlib.pyplot as plt
import time
import itertools
import datetime
import os
# Utilidades de Google Firebase
import firebase_admin
from firebase_admin import storage
from firebase_admin import credentials
# Modelo Pre Entrenado
from src.yolo3.model import *
from scr.yolo3.detect import *
from src.utils.image import *
from src.utils.datagen import *
from src.utils.fixes import *
```

Nota: En este bloque de código se importan las librerías necesarias para el funcionamiento de todo el código.

La explicación y el uso de cada uno de los elementos de esta cabecera son los siguientes:

- Importamos cv2, que es OpenCV para poder preprocesar y gestionar la captura de imágenes.
- Numpy es una librería de matemática para Python presente en varias aplicaciones interesantes como el módulo de control python-control. En este caso la usamos para el procesado de las matrices que componen la imagen.
- Matplotlib es la librería de Python que se usa para graficar. En este caso la usamos para graficar las imágenes ya procesadas en forma de matrices.
- Itertools se usa para generar y analizar las permutaciones posibles de las clases detectadas y así poder determinar la ausencia del casco de seguridad.
- Datetime se utiliza para generar una etiqueta de tiempo única para cada imagen capturada y procesada como alerta, y así poder almacenarla en la memoria interna y consecuentemente en la base de datos.
- Os es para poder acceder al sistema de modo que nos permita almacenar las capturas.
- Firebase_admin y sus derivados son las utilidades que nos permiten conectar- nos a nuestra base de datos en Google Firebase, autenticarnos y guardar las capturas.
- Finalmente, src.utils contiene librerías de apoyo que contienen funciones tanto de detección como de dibujado de la imagen.

Seguidamente, tenemos el bloque de inicialización:

Figura 8

Bloque de código de inicialización

```
#Comprobamos si existe una sesion ya activa de firebase y la cerramos
if firebase_admin._apps:
firebase_admin.delete_app(firebase_admin.get_app())
# Comprobamos la carpeta del usuario ya existe
if os.path.exists('Username.txt'):
with open('Username.txt', 'r') as f:
username = f.read().strip()
else:
# Si no existe la carpeta del usuario creamos una
username = input('Ingrese el correo electronico registrado en la app')
# Guardamos el correo introducido
with open('Username.txt', 'w') as f:
f.write(username)
# Mensaje de bienvenida
print(f'Bienvenido, {username}!')
from src.utils.fixes import *
```

Nota: en esta parte del código, se comprueba la existencia de un usuario previo, y de ser necesario se crea uno nuevo con un correo previamente registrado en la base de datos.

En este bloque de código, ocurre lo siguiente:

- Comprobamos si existe una sesión ya activa de Firebase y la cerramos:
- Si hay aplicaciones activas en Firebase, eliminamos la aplicación activa.
- Comprobamos si la carpeta del usuario ya existe:
- Si el archivo Username.txt existe, lo abrimos y leemos el nombre de usuario almacenado.
- Si la carpeta del usuario no existe:
- Pedimos al usuario que ingrese su correo electrónico registrado en la aplicación.
- Guardamos el correo electrónico introducido en el archivo Username.txt.

- Mostramos un mensaje de bienvenida al usuario utilizando el nombre de usuario leído o introducido.

Una vez inicializado, podemos comenzar a preparar el modelo pre-entrenado mediante la siguiente función:

Figura 9

Función de preparación del modelo

```
def prepare_model(approach):
'''
Preparamos el modelo YOLO
'''
global input_shapes, class_names, anchor_boxes, num_classes,
num_anchors, model
# Definimos la forma de la imagen
input_shape = (416,416)
# Nombres de las clases
if approach == 1:
# Casco, vestimenta y trabajador
class_names = ['C','V','T']
anchor_boxes = np.array([[76,59],[84,136],[188,255]])/32,
np.array([[25,15],[46,29],[27,56]])/16,
np.array([[5,3],[10,8],[12,26]])/8,
dtype='float64'
)
# Numero de clases y de anclas
num_classes = len(class_names)
num_anchors = anchor_boxes.shape[0] * anchor_boxes.shape[1]
# Entrada y salida
input_tensor = Input(shape=(input_shaape[0]), input_shape[1], 3)
num_out_filters = (num_anchors//3)*(5+num_classes)
# Construimos el modelo
model = yolo_body(input_tensor, num_out_filters)
# Cargamos los pesos (Kernels pre entrenados)
weight_path = f'yolo_model3.h5'
model.load_weight(weight_path)
```

Nota: Esta función es fundamental para el funcionamiento correcto del programa de reconocimiento y reportes, pues en esta función se definen los parámetros de configuración de nuestra red neuronal.

Definición de la Función: La función prepare_model toma un argumento approach, que determina la configuración del modelo. El comentario triple entre comillas (‘’) describe brevemente el propósito de la función.

Las variables globales son las siguientes:

- input_shapes, class_names, anchor_boxes, num_classes, num_anchors, model: Se declaran como variables globales para que puedan ser accedidas fuera de la función.
- input_shape = (416, 416): Define la forma de entrada de las imágenes, que es de 416 × 416 píxeles.

Nombres de las Clases:

- if approach == 1: Se verifica si el argumento approach es 1.
- class_names = ['C', 'V', 'T']: Define los nombres de las clases como Casco, Vestimenta y Trabajador.

Definición de las Cajas de Ancla

- anchor_boxes: Define las cajas de ancla normalizadas por el tamaño de la celda de la cuadrícula.
- np.array([[76, 59], [84, 136], [188, 255]])/32: Normaliza las dimensiones de las cajas de ancla por 32.
- np.array([[25, 15], [46, 29], [27, 56]])/16: Normaliza las dimensiones de las cajas de ancla por 16.

Forma de la Imagen:

- np.array([[5, 3], [10, 8], [12, 26]])/8: Normaliza las dimensiones de las cajas de ancla por 8.

Número de Clases y de Anclas

- num_classes = len(class_names): Calcula el número de clases basado en la longitud de class_names.

- num_anchors = anchor_boxes.shape[0] * anchor_boxes.shape[1]: Calcula el número de anclas como el producto del número de filas y columnas de anchor_boxes.

Entrada y Salida:

- input_tensor = Input(shape=(input_shape[0], input_shape[1], 3)): De- fine el tensor de entrada con la forma de la imagen y 3 canales de color (RGB).
- num_out_filters = (num_anchors//3)*(5+num_classes): Calcula el número de filtros de salida basado en el número de anclas y clases.

Construcción del Modelo:

- model = yolo_body(input_tensor, num_out_filters): Construye el modelo YOLO usando el tensor de entrada y el número de filtros de salida.

Carga de Pesos Pre-entrenados:

- weight_path = f'yolo_model3.h5': Define la ruta de los pesos pre-entrenados.
- model.load_weight(weight_path): Carga los pesos pre-entrenados en el modelo

3.2.2. Aplicación de la detección

Una vez preparado el modelo, y definida la forma de la matriz que entrará al procesamiento, podemos proceder a la detección de las 3 clases definidas:

Figura 10

Función de detección

```
def get_detection(img, class_name):
# Trabajamos con una copia de la imagen
act_img = img.copy()
# Dimensionamos la imagen
ih, iw = act_img.shape[:2]
# Preprocesamos la imagen
img = letterbox_image(img, input_shape)
img = np.expand_dims(img, 0)
image_data = np.array(img)/255
# Prediccion en crudo del modelo YOLO
prediction = model.predict(image_data)
# Procesamos la prediccion para obtener los encuadres, soporte de hasta
#10 obreros con todos los implementos en simultaneo (expandible)
boxes = detection(
prediction,
anchor_boxes,
num_classes,
image_shape = (ih,iw),
input_shape = (416,416),
max_boxes = 30,
# Umbral de precision estimada
score_threshold = 0.7,
iou_threshold = 0.6,
classes_can_overlap = false)
# Convertimos el tensor a formato de numpy
boxes = boxes[0].numpy()
# Dibujamos los encuadres de las clases detectadas
detected_classes = []
for box in boxes:
class_id = int(box[5])
class_name = class_names[class_id]
score = box[4]
xmin, ymin, xmax, ymax = box[:4].astype(np.int)
cv2.rectangle(act_img, (xmin, ymin), (xmax, ymax), (0,255,0), 2)
cv2.putText(act_img, f'{class_name}: {score:.2f}', (xmin, ymin - 5),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)
detected_classes.append(class_name)
return act_img, detected_classes
```

Nota: Esta función se encarga de aplicar la detección bajo ciertos criterios predefinidos.

Definición de la Función La función get_detection toma dos argumentos: img, que es la imagen en la cual se quiere realizar la detección, y class_name, que representa el nombre de la clase a detectar.

Copia de la Imagen

- act_img = img.copy(): Se crea una copia de la imagen original para trabajar sobre ella sin modificar la imagen original.

Dimensiones de la Imagen

- ih, iw = act_img.shape[:2]: Se obtienen las dimensiones (alto y ancho) de la imagen.
- Preprocesamiento de la Imagen
- img = letterbox_image(img, input_shape): Se ajusta el tamaño de la imagen a la forma de entrada requerida por el modelo.
- img = np.expand_dims(img, 0): Se añade una dimensión extra para que la imagen tenga el formato adecuado para la entrada del modelo.
- image_data = np.array(img)/255: Se normalizan los valores de los píxeles de la imagen dividiéndolos por 255.

Predicción del Modelo

- prediction = model.predict(image_data): Se realiza la predicción en crudo utilizando el modelo YOLO.

Procesamiento de la Predicción

- boxes = detection (...): Se procesa la predicción para obtener los encuadres de las detecciones. Esta función recibe varios parámetros, incluyendo la pre- dicción en crudo, las cajas de ancla, el número de clases, las dimensiones de la imagen, el número máximo de encuadres, el umbral de precisión y el umbral de IoU (Intersection over Union). La opción classes_can_overlap = false indica que las clases no pueden solaparse.

Conversión del Tensor a Formato Numpy

- boxes = boxes[0].numpy(): Se convierte el tensor de encuadres a un array de Numpy.

Dibujar los Encuadres

- detected_classes = []: Se inicializa una lista para almacenar las clases detectadas.
- for box in boxes:: Se itera sobre cada encuadre detectado.
- class_id = int(box[5]): Se obtiene el ID de la clase detectada.
- class_name = class_names[class_id]: Se obtiene el nombre de la clase usando el ID.
- score = box[4]: Se obtiene la puntuación de la detección.
- xmin, ymin, xmax, ymax = box[:4].astype(np.int): Se obtienen las coordenadas del encuadre.
- cv2.rectangle(act_img, (xmin, ymin), (xmax, ymax), (0, 255, 0),

Se dibuja un rectángulo alrededor del objeto detectado.

- cv2.putText(act_img, f'{class_name}: {score:.2f}', (xmin, ymin - 5), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), Se añade un texto con el nombre de la clase y la puntuación cerca del encuadre.
- detected_classes.append(class_name): Se añade el nombre de la clase detectada a la lista.

Retorno de Resultados

- return act_img, detected_classes: Se devuelve la imagen con los encuadres dibujados y la lista de clases detectadas.

3.2.3. Funciones y variables auxiliares

Para que el código pueda funcionar en su conjunto, tanto para la predicción como para las alertas, son necesarias algunas funciones y variables auxiliares que se definen como:

Figura 11

Auxiliares

```
# Funcion para mostrar la imagen
def plt_imshow(img):
plt.figure(figsize=(5, 5))
plt.imshow(img)
plt.axis('off')
# Preparacion del modelo con los kernels
prepare_model(approach=1)
#Credenciales de Firebase
cred = credentials.Certificate("certificado.json")
app = firebase_admin.initialize_app(cred)
bucket = storage.bucket(app=app, name="tesis-1e07b3.appspot.com")
```

Nota: Este bloque contiene las funciones para mostrar la imagen, ejecuta la preparación del modelo y autentica al sistema con la base de datos.

Función para Mostrar la Imagen

- La función plt_imshow se utiliza para mostrar una imagen utilizando la biblioteca matplotlib.
- def plt_imshow(img): Define la función que toma un argumento img, que es la imagen a mostrar.
- plt.figure(figsize=(5, 5)): Crea una nueva figura con un tamaño de 5x5 pulgadas.
- plt.imshow(img): Muestra la imagen proporcionada.
- plt.axis('off'): Desactiva los ejes para una visualización más limpia.

Preparación del Modelo con los Kernels

La siguiente línea de código llama a la función prepare_model para preparar el modelo YOLO con un enfoque específico.

- prepare_model(approach=1): Llama a la función prepare_model con el argu- mento approach=1. Esto prepara el modelo YOLO con los parámetros y kernels predefinidos para el enfoque 1.

Configuración de las Credenciales de Firebase

A continuación, se configuran las credenciales de Firebase para acceder a un bucket de almacenamiento.

- cred = credentials.Certificate("certificado.json"): Carga las creden- ciales de Firebase desde un archivo JSON llamado certificado.json. Este archivo contiene las credenciales necesarias para autenticar la aplicación con Fi- rebase.
- app = firebase_admin.initialize_app(cred): Inicializa la aplicación de Firebase con las credenciales cargadas.
- bucket = storage.bucket(app=app, name="tesis-1e07b3.appspot.com"): Obtiene una referencia al bucket de almacenamiento de Firebase llamado tesis-1e07b3.apps asociado con la aplicación inicializada.

3.2.4. Manejo de alertas

La siguiente función maneja las alertas de ausencias detectadas, de modo que puedan ser inmediatamente guardadas tanto en la memoria local como en la base de datos, y, por consiguiente, en la aplicación HMI:

Figura 12

Bloque de código de manejo de alertas

```
def alerts(classes, frame):
# Tomamos la etiqueta de tiempo para crear imagenes de alertas unicas
timestamp = datetime.datetime.now().strftime("%d%m%Y-%H%M%S")
filename = f"Evento_{timestamp}.jpg"
# Flags para tomar una sola captura evitando un bucle
capture_taken = False
last_capture_time = 0
# Creamos una lista para manejar las clases
triads = list(itertools.permutations(['T', 'C', 'V']))
missing_triads = []
for triad in triads:
if triad not in itertools.combinations(classes, 3):
missing_triads.append(triad)
# Verificamos si falta algun elemento de proteccion
if len(classes) > 0:
if missing_triads and not capture_taken:
cv2.imwrite(f"Capturas\{filename}", frame)
blob = bucket.blob(username + f"/{filename}")
blob.upload_from_filename(f"Capturas\{filename}")
if capture_taken and (time.time() - last_capture_time) > 10:
capture taken = False
```

Nota: En esta sección del código se analizan las permutaciones posibles de las clases detectaras, esto para poder detectar ausencias.

Generación de la Etiqueta de Tiempo: La función alerts comienza generando una etiqueta de tiempo para asegurar que cada imagen de alerta tenga un nombre único.

- timestamp =
- datetime.datetime.now().strftime("%d%m%Y%H%M%S"): Obtiene la fecha y hora actual y la formatea como un string con el formato “ddmmyyyy- hhmmss”.
- filename = f"Evento_{timestamp}.jpg": Crea el nombre del archivo de la imagen usando la etiqueta de tiempo generada.

Inicialización de Flags y Tiempo: Se inicializan variables para controlar la captura de imágenes.

- capture_taken = False: Flag para indicar si se ha tomado una captura.

- last_capture_time = 0: Variable para almacenar el tiempo de la última captura.

Generación de Combinaciones y Permutaciones: Se crean listas para manejar las clases detectadas y verificar la ausencia de elementos de protección.

- triads = list(itertools.permutations(['T', 'C', 'V'])): Genera todas las permutaciones posibles de los elementos de protección: Trabajador (T), Casco (C) y Vestimenta (V).
- missing_triads = []: Lista para almacenar las combinaciones de elementos de protección que faltan.
- for triad in triads: if triad not in itertools.combinations(classes, 3): missing_triads.append(triad): Verifica si cada permutación no está presente en las combinaciones de clases detectadas y la agrega a missing_triads si falta.

Verificación y Captura de Imágenes: La función verifica si falta algún elemento de protección y captura una imagen si es necesario.

- if len(classes)>0: if missing_triads and not capture_taken:: Veri- fica si hay clases detectadas y si falta algún elemento de protección, además de si no se ha tomado una captura previamente.
- cv2.imwrite(f"Capturas{\filename}", frame): Guarda la imagen de la captura en el directorio Capturas.
- blob = bucket.blob(username + f/{filename}"): Crea un blob en el bucket de Firebase con el nombre del archivo.
- blob.upload_from_filename(f “Capturas{\filename}"): Sube la imagen capturada al bucket de Firebase.

Reset de la Captura: La función restablece la flag de captura después de un tiempo determinado.

- if capture_taken and (time.time() - last_capture_time) >10: capture_taken = False: Verifica si se ha tomado una captura y si han pasado más de 10 segun- dos desde la última captura para restablecer la flag capture_taken.

3.2.5. Captura en vivo

Finalmente, se cierra esta sección de código de detección que ejecuta la Rpi con el bucle de ejecución de todas las funciones anteriores en vivo:

Figura 13

Bucle principal de ejecución

```
cap = cv2.VideoCapture(0)
while True:
ret, frame = cap.read()
detected_frame, detected_classes = get_detection(frame, class_names)
alerts(detected_classes, detected_frame)
key = cv2.waitKey(1) & 0xFF
if key == ord('q')
break
cap.release()
cv2.destroyAllWindows()
```

Nota: Este bucle mantiene al programa completo en ejecución siempre, hasta que se detenga mediante la orden definida.

Inicialización de la Captura de Video: La función comienza inicializando la captura de video.

- cap = cv2.VideoCapture(0): Inicializa la captura de video desde la cámara predeterminada (dispositivo de cámara 0).

Bucle Principal de Captura y Detección: El bucle while se ejecuta continuamente para capturar cuadros de video, realizar detección de objetos y generar alertas.

- while True: Inicia un bucle infinito para capturar y procesar los cuadros de video en tiempo real.

Captura del Cuadro de Video: Cada iteración del bucle captura un cuadro de video.

- ret, frame = cap.read(): Captura un cuadro de video y lo almacena en la variable frame. ret indica si la captura fue exitosa.

Detección de Objetos: El cuadro capturado se pasa a la función de detección de objetos.

- detected_frame, detected_classes = get_detection(frame, class_names): Llama a la función get_detection para detectar objetos en el cuadro frame. La función devuelve el cuadro con los objetos detectados (detected_frame) y las clases detectadas (detected_classes).

Generación de Alertas: La función de alertas se llama para verificar la presencia de elementos de protección y generar alertas si es necesario.

- alerts(detected_classes, detected_frame): Llama a la función alerts para procesar las clases detectadas y generar alertas si faltan elementos de protección. También guarda el cuadro detectado (detected_frame) si es necesario.

Detección de la Tecla de Salida: El código verifica si se presiona la tecla ’q’ para salir del bucle.

- key = cv2.waitKey(1) & 0xFF: Espera 1 milisegundo a que se presione una tecla y almacena el valor de la tecla presionada en la variable key.
- if key == ord('q'): break: Si se presiona la tecla ’q’, el bucle se rompe y se sale de la captura de video.

Liberación de Recursos: Después de salir del bucle, se liberan los recursos de la cámara y se cierran las ventanas de OpenCV.

- cap.release(): Libera el dispositivo de captura de video.
- cv2.destroyAllWindows(): Cierra todas las ventanas de OpenCV abiertas.

Es así como se logra la detección, clasificación y alertas a la base de datos con la Rpi ejecutando este código, en la siguiente sección veremos el diseño de la aplicación móvil y su interacción con la base de datos.

3.3. DISEÑO DE LA APLICACIÓN ANDROID HNI

Para poder acceder de manera remota desde cualquier lugar del mundo a las alertas de la base de datos, se diseñó una aplicación para sistemas Android a modo de HMI para que el supervisor a cargo de la seguridad esté siempre pendiente de cualquier ocurrencia.

Esta aplicación se basa en el servicio de autenticación de Firebase, que nos permite autenticar usuarios con correo electrónico y contraseña, de modo que se logró relacionar a cada usuario con un contenedor de datos particular, la aplicación fue programada en Java utilizando Android Studio.

Cabecera de la aplicación:

Se comienza por importar las librerías y paquetes necesarios:

Figura 14

Cabecera de la aplicación diseñada

```
package com.gabriel.iot_app;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import androidx.annotation.NonNull;
import android.content.Intent;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.aut.FirebaseAuth;
```

Nota: En esta sección se importan todos los paquetes necesarios para el funcionamiento de la aplicación.

- import androidx.appcompat.app.AppCompatActivity;: Es una actividad base para las actividades que utilizan la biblioteca de soporte de la interfaz de usuario.
- import android.os.Bundle;: Importa la clase Bundle, que es un contenedor para pasar datos entre actividades.
- import androidx.annotation.NonNull;: Importa la anotación NonNull, utilizada para indicar que un parámetro no debe ser nulo.
- import android.content.Intent;: Importa la clase Intent, que permite iniciar nuevas actividades o servicios.
- import android.view.View;: Importa la clase View, que representa la interfaz de usuario básica.

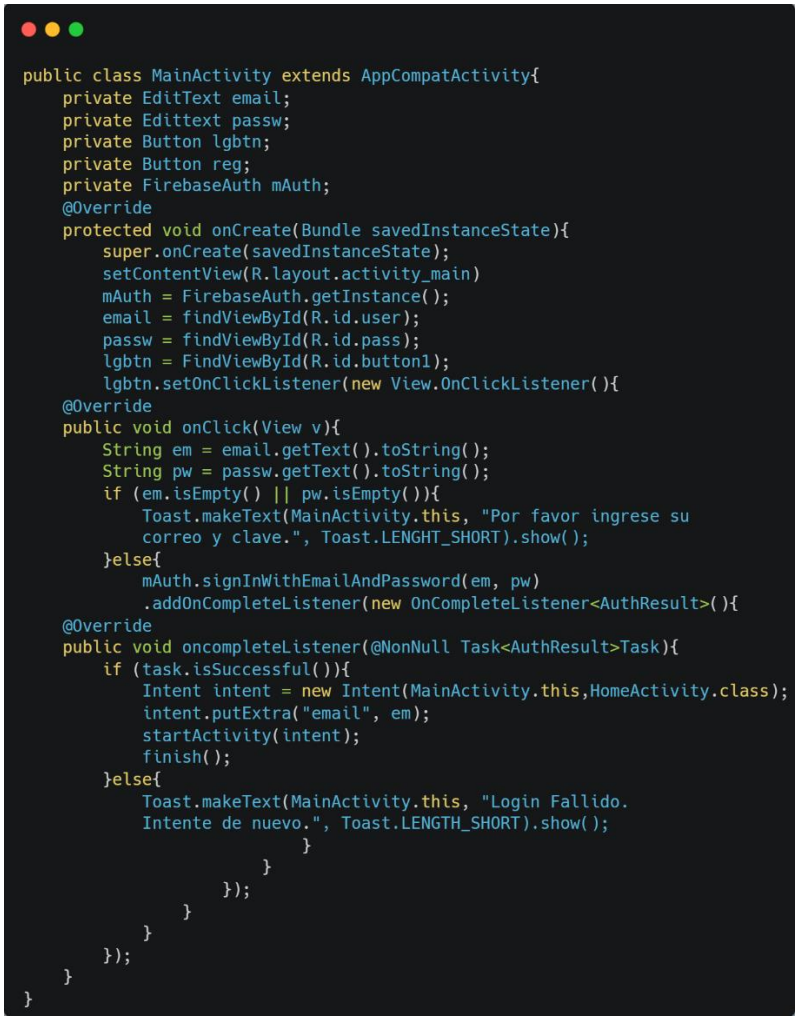
- import android.widget.Button;: Importa la clase Button, que representa un botón de la interfaz de usuario.
- import android.widget.EditText;: Importa la clase EditText, que representa un campo de entrada de texto.
- import android.widget.Toast;: Importa la clase Toast, que muestra mensajes emergentes breves.
- import com.google.android.gms.tasks.OnCompleteListener;: Importa la interfaz OnCompleteListener, utilizada para recibir una notificación cuando se completa una tarea.
- import com.google.android.gms.tasks.Task;: Importa la clase Task, que representa una tarea asincrónica.
- mport com.google.firebase.auth.AuthResult;: Representa el resultado de una operación de autenticación.
- import com.google.firebase.auth.FirebaseAuth;: Proporciona métodos para la autenticación de usuarios con Firebase.

3.3.1. Clase principal

La programación nativa para Android en Java se basa en clases que llamamos actividades, la actividad principal de esta aplicación es:

Figura 15

Actividad principal de la aplicación



```
public class MainActivity extends AppCompatActivity{
    private EditText email;
    private Edittext passw;
    private Button lgbtn;
    private Button reg;
    private FirebaseAuth mAuth;
    @Override
    protected void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main)
        mAuth = FirebaseAuth.getInstance();
        email = findViewById(R.id.user);
        passw = findViewById(R.id.pass);
        lgbtn = FindViewById(R.id.button1);
        lgbtn.setOnClickListener(new View.OnClickListener(){
    @Override
    public void onClick(View v){
        String em = email.getText().toString();
        String pw = passw.getText().toString();
        if (em.isEmpty() || pw.isEmpty()){
            Toast.makeText(MainActivity.this, "Por favor ingrese su
            correo y clave.", Toast.LENGHT_SHORT).show();
        }else{
            mAuth.signInWithEmailAndPassword(em, pw)
            .addOnCompleteListener(new OnCompleteListener<AuthResult>(){
    @Override
    public void oncompleteListener(@NonNull Task<AuthResult>Task){
        if (task.isSuccessful()){
            Intent intent = new Intent(MainActivity.this,HomeActivity.class);
            intent.putExtra("email", em);
            startActivity(intent);
            finish();
        }else{
            Toast.makeText(MainActivity.this, "Login Fallido.
            Intente de nuevo.", Toast.LENGTH_SHORT).show();
                                }
                            }
                        });
                    }
                }
            });
    }
}
```

Nota: Esta actividad muestra la ventana de autenticación de la aplicación para acceder a los reportes.

Paquete de la Aplicación: El código comienza con la declaración del paquete y las importaciones necesarias para el funcionamiento de la aplicación. Estas incluyen clases para manejar la interfaz de usuario, la autenticación de Firebase y la navegación entre actividades.

Declaración de la Clase Principal

- public class MainActivity extends AppCompatActivity : Define la clase principal de la actividad que extiende AppCompatActivity, la cual es una clase base para actividades que utilizan la biblioteca de soporte de la interfaz de usuario.

Declaración de Variables

- private EditText email;: Campo de entrada de texto para el correo electrónico del usuario.
- private EditText passw;: Campo de entrada de texto para la contraseña del usuario.
- private Button lgbtn;: Botón para iniciar sesión.
- private Button reg;: Botón para registrarse (no utilizado en este código).
- private FirebaseAuth mAuth;: Instancia de FirebaseAuth para manejar la autenticación de usuarios.

Método onCreate: El método onCreate se ejecuta cuando la actividad es creada.

- super.onCreate(savedInstanceState);: Llama al método onCreate de la superclase para realizar la inicialización estándar.
- setContentView(R.layout.activity_main);: Establece el diseño de la interfaz de usuario de la actividad utilizando el archivo de diseño activity_main.xml.
- mAuth = FirebaseAuth.getInstance();: Inicializa la instancia de Firebase para autenticación

- email = findViewById(R.id.user);: Asigna el campo de entrada de texto pa- ra el correo electrónico.
- passw = findViewById(R.id.pass);: Asigna el campo de entrada de texto para la contraseña.
- lgbtn = findViewById(R.id.button1);: Asigna el botón de inicio de sesión.

Configuración del Botón de Inicio de Sesión: Se configura el comportamiento del botón de inicio de sesión.

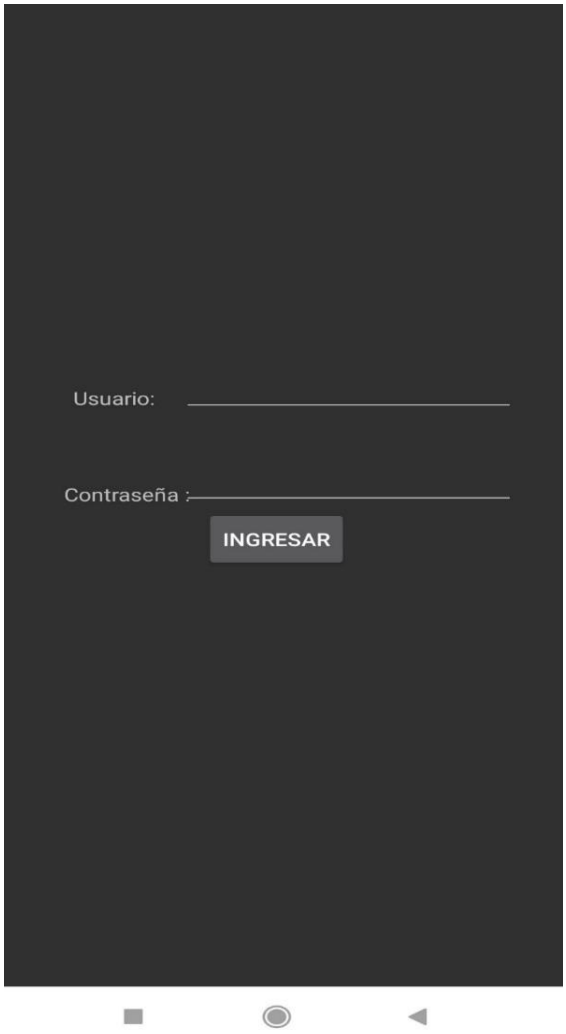
- lgbtn.setOnClickListener: Configura un Listener para el botón de inicio de sesión.
- @Override public void onClick(View v): Define el método onClick que se ejecuta al hacer clic en el botón.
- String em = email.getText().toString();: Obtiene el correo electrónico ingresado por el usuario.
- String pw = passw.getText().toString();: Obtiene la contraseña ingresa- da por el usuario.
- if (em.isEmpty() || pw.isEmpty()): Verifica si el correo electrónico o la con- traseña están vacíos.
- Toast.makeText(MainActivity.this Toast.LENGTH_SHORT).show(); : Muestra un mensaje emergente si los campos están vacíos.
- mAuth.signInWithEmailAndPassword(em, pw): Intenta iniciar sesión con el correo electrónico y la contraseña proporcionados.
- .addOnCompleteListener(new OnCompleteListener<AuthResult>() : Agrega un OnCompleteListener para manejar el resultado de la autenticación.

- @Override public void onComplete(@NonNull Task<AuthResult>task) : Define el método onComplete que se ejecuta cuando la autenticación se completa.
- if (task.isSuccessful()): Verifica si la autenticación fue exitosa.
- Intent intent = new Intent(MainActivity.this, HomeActivity.class);: Crea una intención para iniciar la HomeActivity.
- intent.putExtra(.email", em);: Agrega el correo electrónico como extra en la intención.
- startActivity(intent);: Inicia la HomeActivity.
- finish();: Finaliza la MainActivity.
- Toast.makeText(MainActivity.this, "Login Fallido. Intente de nuevo.", Toast.LENGTH_SHORT).show();: Muestra un mensaje emergente si la autenticación falla.

Esta clase nos da el siguiente resultado:

Figura 16

Interfaz de autenticación de usuario



Nota: Vista principal de la ventana de autenticación de usuario

3.3.2. Clase de eventos

La siguiente clase es la ventana principal, donde accede el usuario una vez se haya autenticado correctamente, es en donde se visualizarán las alertas tomadas y se podrá interactuar con ellas.

Figura 17

Cabecera de la actividad de visualización de eventos.

```
import java.io.FileOutputStream;
import java.io.IOException;
import static android.content.ContentValues.TAG;
import android.content.Intent;
import android.os.Bundle;
import android.os.Environment;
import android.util.log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.ImageView;
import android.graphics.BitmapFactory;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.GridLayoutManager;
import androidx.recyclerview.widget.RecyclerView;
import com.google.com.android.gms.tasks.OnFailureListener;
import android.graphics.Bitmap;
import android.widget.Toast;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.firebase.storage.FirebaseStorage;
import com.google.fireabse.storage.ListResult;
import com.google.firebase.storage.StorageReference;
import java.io.File
import java.util.ArrayList;
import java.util.List;
```

Nota: En esta cabecera se importan todos los paquetes necesarios para la visualización de los reportes.

Como vemos, son paquetes muy parecidos a los de la actividad principal, sin embargo, se añaden más principalmente para manejar las imágenes y la interacción con la base de datos:

- import java.io.FileOutputStream;: Proporciona clases para la creación y manipulación de archivos en el sistema de archivos.
- import java.io.IOException;: Maneja excepciones quepueden ocurrir durante operaciones de entrada/salida.
- import static android.content.ContentValues.TAG;: Proporciona una constante comúnmente utilizada para etiquetar mensajes de registro (logs) en Android.
- import android.content.Intent;: Permite el uso de intenciones para iniciar nuevas actividades o servicios.
- import android.os.Bundle;: Proporciona una clase para pasar datos entre ac- tividades en Android.
- import android.os.Environment;: Proporciona acceso a variables del entorno, como el estado del almacenamiento externo.
- import android.util.Log;: Proporciona métodos para el registro (logging) de información, advertencias y errores en Android.
- import android.view.LayoutInflater;: Permite inflar archivos de diseño XML en objetos de vista correspondientes.
- import android.view.View;: Proporciona la clase base para widgets, que son componentes interactivos de la interfaz de usuario.
- import android.view.ViewGroup;: Define el comportamiento y las propiedades de los contenedores de vistas en Android.
- import android.widget.Button;: Proporciona la clase para crear y manejar botones en la interfaz de usuario.
- import android.widget.ImageView;: Proporciona la clase para manejar y mostrar imágenes en la interfaz de usuario.

- import android.graphics.BitmapFactory;: Proporciona métodos para decodificar archivos de imagen en objetos Bitmap.
- import androidx.annotation.NonNull;: Proporciona una anotación para indicar que un parámetro, campo o método no puede ser nulo.
- import androidx.appcompat.app.AppCompatActivity;: Extiende la funcionalidad de la clase Activity para soportar características más avanzadas de la interfaz de usuario.
- import androidx.recyclerview.widget.GridLayoutManager;: Proporciona un LayoutManager para RecyclerView que coloca los elementos en una cuadrícula.
- import androidx.recyclerview.widget.RecyclerView;: Proporciona una vista flexible para mostrar grandes conjuntos de datos de forma eficiente.
- import com.google.android.gms.tasks.OnFailureListener;: Proporciona una interfaz para manejar fallos en tareas asincrónicas.
- import android.graphics.Bitmap;: Proporciona la clase para representar y manejar imágenes en memoria.
- import android.widget.Toast;: Proporciona una clase para mostrar mensajes breves en la pantalla.
- import com.google.android.gms.tasks.OnSuccessListener;: Proporciona una interfaz para manejar el éxito en tareas asincrónicas.
- import com.google.firebase.storage.FirebaseStorage;: Proporciona acceso a Firebase Storage, un servicio de almacenamiento en la nube.
- import com.google.firebase.storage.ListResult;: Proporciona una clase que representa el resultado de una operación de listado de objetos en Firebase Storage.

- import com.google.firebase.storage.StorageReference;: Proporciona una referencia a un archivo o directorio en Firebase Storage.
- import java.io.File;: Proporciona una clase para la creación y manipulación de archivos y directorios en el sistema de archivos.
- import java.util.ArrayList;: Proporciona la implementación de una lista redimensionable.
- import java.util.List;: Proporciona la interfaz para colecciones ordenadas.

3.3.3. Definición de la función loadPhotos

La función que nos permite cargar las fotos desde Firebase está definida de la siguiente manera:

Figura 18

Definición de la función LoadPhotos

```
private void loadPhotos(){
    // Log para debug
    log.d(TAG, "Cargando fotos");
    Intent intent = getIntent();
    String em = intent.getStringExtra("email");
    // Referenciamos a la base de datos
    photosRef.listAll().addOnsuccessListener(new
    OnSuccessListener<ListResult>(){
    @Override
        public void onSuccess(ListResult listResult){
            log.d(TAG, "Recibi" + list.Result.getItems().size() + "Fotos");
            for (StorageReference item: listResult.getItems()){
            photosList.add(item);
        }
        adapter.notifyDataSetChanged();
        }
    });
}
```

Nota: Esta función es la encargada de cargar las imágenes desde Firebase.

- private void loadPhotos(){}: Define un método privado llamado loadPhotos que se utiliza para cargar fotos desde Firebase Storage y actualizar la vista del RecyclerView.
- Log.d(TAG, "Cargando fotos");: Escribe un mensaje de depuración en el registro (log) con la etiqueta TAG, indicando que el proceso de carga de fotos ha comenzado.
- Intent intent = getIntent();: Obtiene el Intent que inició la actividad actual, permitiendo acceder a los datos adicionales que fueron pasados con él.
- String em = intent.getStringExtra(.email");: Extrae el valor asociado con la clave “email" del Intent y lo almacena en la variable em.
- photosRef.listAll().addOnSuccessListener(): Llama al método listAll en la referencia de fotos (photosRef) para obtener una lista de todos los elementos almacenados. Agrega un OnSuccessListener para manejar el resultado exitoso de la operación.
- @Override public void onSuccess(ListResult listResult){}: Define el comportamiento del método onSuccess, que se ejecuta cuando la operación de listAll es exitosa. Recibe un objeto ListResult que contiene los resultados de la lista.
- Log.d(TAG,"Recibi "+listResult.getItems().size()+" Fotos");: Escribe un mensaje de depuración en el registro (log) con la etiqueta TAG, indicando la cantidad de fotos recibidas.
- for (StorageReference item: listResult.getItems()) {}: Itera sobre ca- da elemento de la lista de resultados (listResult.getItems()) y ejecuta el bloque de código para cada uno.

- photosList.add(item);: Agrega cada referencia de foto (StorageReference) a la lista de fotos (photosList).
- adapter.notifyDataSetChanged();: Notifica al adaptador (adapter) que los datos han cambiado, lo que provoca una actualización de la vista del RecyclerView

3.3.4. Definición del adaptador

Este adaptador es el encargado de gestionar las imágenes, se define como una clase pues es más sencillo trabajar con él así. Su definición es:

Figura 19

Definición del adaptador de imágenes.

```
private class PhotoAdapter extends RecyclerView.Adapter<PhotoViewHolder>{
    @NonNull
    @Override
        public PhotoViewHolder onCreateViewHolder(@NonNull ViewGroup parent,
        int viewType){
            View view =
            LayoutInflater.from(parent.getContext()).inflate(R.layout.item_photo,
            parent, false);
            return new PhotoViewHolder(view);
    }
}
```

Nota: Este adaptador se encarga de gestionar la visualización de los reportes.

- private class PhotoAdapter extends RecyclerView.Adapter: Define una clase privada llamada PhotoAdapter que extiende RecyclerView.Adapter con un tipo genérico PhotoViewHolder. Esta clase es un adaptador que proporciona las vistas necesarias para mostrar los elementos en un RecyclerView.

- @NonNull @Override public PhotoViewHolder onCreateViewHolder(): Sobrescribe el método onCreateViewHolder de RecyclerView.Adapter. Este mé- todo se llama cuando el RecyclerView necesita una nueva vista para representar un elemento.
- View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.item_photo, parent, false);: Utiliza el LayoutInflater para crear una nueva vista desde el archivo de diseño item_photo.xml. La vista se infla en el contexto del ViewGroup padre (parent) y no se adjunta inmediatamente a parent (por eso el tercer argumento es false).

3.3.5. Decodificado y dibujado de la imagen

La imagen que llega de Firebase generalmente se encuentra codificada en base 64 (bytes), al ser este un estándar para almacenar en bases de datos, por lo que es necesario reconstruirla para poder dibujarla e interpretarla, la siguiente función se encarga de ello:

Figura 20

Decodificado y dibujado de las imágenes.

```
@Override
    public void onBindViewHolder(@NonNull PhotoViewHolder holder, int
    position){
        StorageReference photoRef = photosList.get(position);
        photoRef.getBytes(Long.MAX_VALUE).addOnSuccessListener(new
            OnSuccessListener<byte[]>(){
            @Override
                public void onSuccess(byte[] bytes){
                    Bitmap bitmap = BitmapFactory.decodeByteArray(bytes, 0,
                    bytes.lenght);
                    holder.photoImageView.setImageBitmap(bitmap);
                }
        }).addOnFailureListener(new OnFailureListener(){
        @Override
            public void onFailure(@NonNull Exception exception){
            Log.e(TAG, "Fallo al cargar las fotos.", exception);
        }
    });
}
```

Nota: Este código decodifica la imagen en un mapa de bits y hace posible su visualización.

- @Override public void onBindViewHolder(@NonNull PhotoViewHolder: So- brescribe el método onBindViewHolder de RecyclerView.Adapter. Este méto- do se llama para vincular un PhotoViewHolder con datos en una posición específica.
- StorageReference photoRef = photosList.get(position);: Obtiene una referencia a la foto en la posición especificada de la lista de fotos (photosList).
- photoRef.getBytes(Long.MAX_VALUE).addOnSuccessListener: Solicita des- cargar los datos de la foto referenciada en photoRef. Se usa Long.MAX_VALUE para obtener todos los bytes de la imagen.

- @Override public void onSuccess(byte[] bytes){}: Sobrescribe el método onSuccess que se llama cuando la descarga de la imagen es exitosa. Los bytes descargados se pasan como un arreglo de bytes (byte[]).
- Bitmap bitmap = BitmapFactory.decodeByteArray();: Decodifica el arreglo de bytes en un objeto Bitmap.
- holder.photoImageView.setImageBitmap(bitmap);: Establece el Bitmap de- codificado como la imagen del ImageView en PhotoViewHolder.
- .addOnFailureListener(new OnFailureListener(){}: Define un listener para manejar fallos en la descarga de la imagen.
- @Override public void onFailure(): Sobrescribe el método onFailure que se llama cuando la descarga de la imagen falla. Se pasa la excepción que describe el error.
- Log.e(TAG, "Fallo al cargar las fotos.", exception);: Registra un men- saje de error junto con la excepción para fines de depuración.

3.3.6. Implementación de un botón de descarga

Para poder lograr que un usuario sea capaz de descargar evidencias, se implementó el siguiente botón:

Figura 21

Código de implementación de un botón de descarga.

```
holder.downloadButton.setOnclickListener(new View.OnClickListener(){
    @Override
    public void onClick(View v){
        // Descargamos la imagen seleccionada desde Firebase
        photoRef.getBytes(long.MAX_VALUE).addOnSuccessListener(new OnSuccessListener<byte[]>(){
            @Override
            public void OnSuccess(byte[] bytes){
                // Guardamos la imagen
                String filename = photoRef.getName();
                File directory = Environment.getExternalStoragePublicDirectory
                (Environment.DIRECTORY_DOWNLOADS);
                File file = new File(directory, filename);
                FileOutputStream fos = null;
                try{
                    fos = new FileOutputStream(file);
                    fos.write(bytes);
                    Toast.makeText(HomeActivity.this, "Descargado " + filename, Toast.LENGTH_SHORT).show();

                } catch(IOException e){
                    Log.e(TAG, "Fallo al guardar la foto", e);
                } finally{
                    if(fos != null){
                        try{
                            fos.close();
                        } catch(IOException e){
                            Log.e(TAG, "Fallo al cerrar la salida", e);
                        }
                    }
                }
            }
        }).addOnFailureListener(new OnFailureListener(){
            @Override
            public void onFailure(@NonNull Exception exception){
                Log.e(TAG, "Fallo al descargar la foto", exception);
            }
        });
    }
});
```

Nota: Este código implementa un botón para poder descargar los reportes desde Firebase hacia el dispositivo.

Lo que hace simplemente es descargar la imagen seleccionada en la carpeta de descargas, cuidando con try-catch los errores de guardado y descarga que puedan ocurrir.

3.3.7. Implementación de un botón de borrado

De manera similar, para lograr que un usuario pueda eliminar imágenes, se implementó el siguiente botón, teniendo en cuenta que borra las imágenes de la base de datos mas no de la memoria interna de la Rpi:

Figura 22

Implementación de un botón de borrado

```
holder.deleteButton.SetOnClickListener(new View.OnClickListener(){
    @Override
    public void onSuccess(Void aVoid){
        photosList.remove(position);
        notifyItemRemoved(position);
        notifyItemRangeChanged(position, getItemCount());
        Toast.makeText(HomeActivity,this, "Eliminado " + photoRef.getName(),
        Toast.LENGTH_SHORT).show();
    }
}).addOnFailureListener(new OnFailureListener(){
    @Override
    public void onFailure(@NonNull Exception exception){
        Log.e(TAG, "Fallo al eliminar foto", exception;)
    }
});

@Override
public int getItemCount(){
    return photosList.size();
}
```

Nota: este código permite la implementación de un botón de eliminación de reportes.

3.3.8. Clase de apoyo

Para poder lograr un manejo más eficiente del código y reducirle complejidad, se define una clase estática llamada "PhotoViewHolder", encargada de contener a las imágenes obtenidas de la base de datos, y cuya definición es:

Figura 23

Clase de apoyo para las imágenes

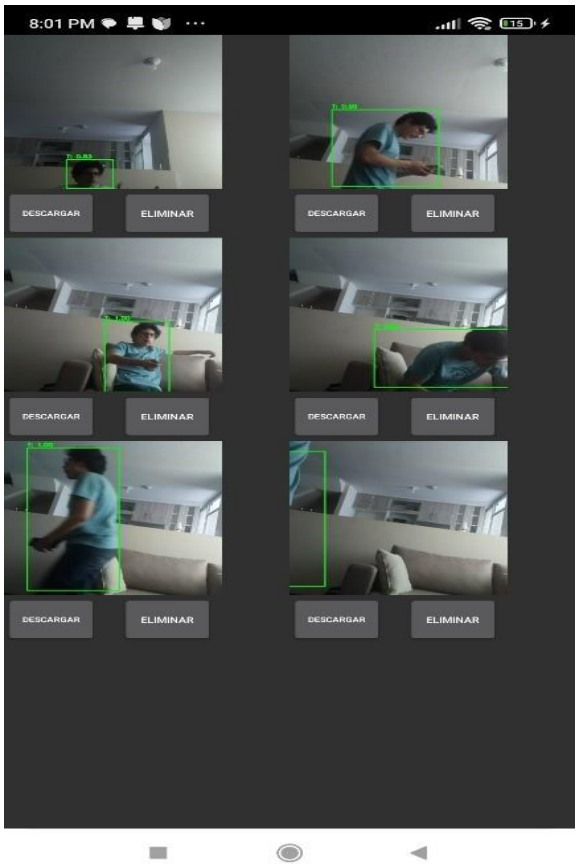
```
private static class PhotoViewHolder extends RecyclerView.ViewHolder{
    ImageView photoImageView;
    Button downloadButton;
    Button deleteButton;
    public PhotoViewHolder(@NonNull View itemView){
        super(itemView);
        photoImageView = itemView.findViewById(R.id.photoImageView);
        downloadButton = itemView.findViewById(R.id.downloadButton);
        deleteButton   = itemView.findViewById(R.id.deleteButton);
    }
}
```

Nota: Clase estática para reducir lo engorroso del código.

Que junto a todo lo explicado anteriormente nos da el siguiente resultado:

Figura 24

Interfaz de visualización con funciones de descarga y borrado



Nota: Interfaz de visualización de reportes con los botones de descarga y borrado.

3.4. CONFIGURACIÓN DE LA RPI

Para lograr que la Rpi lleva a cabo todas estas tareas, no es solo necesario que ejecute el código presentado, sino que se conecte a la red, verifique su conexión y ejecute el código sin importar qua haya sido encendida o apagada de nuevo, es decir, como un servicio. En sistemas operativos Linux es bastante sencillo levantar uno de estos servicios, como se podrá observar a continuación:

3.4.1. Configuración de la conexión GPRS

El sistema operativo PiOS Debian Bullseye ya tiene integrada la funcionalidad de poder conectarse a módems a través de PPP. Si bien el módulo está diseñado con una interfaz UART con protocolo TTL, sobre la marcha se pudo observar que una in- terfaz USB es mucho más estable, por lo que se decidió utilizar un circuito integrado interfaz CH340G para poder conectar el módulo a través de USB. Para configurar este módulo es necesario añadir o modificar el arcvhivo "wvdial.conf", ubicado en la ruta “/etc/wvdial.conf", que como se mencionó líneas arriba debería ya estar incluido en el sistema operativo, de la siguiente manera:

Figura 25

Archivo de configuración wvdial.conf

```
[Dialer Defaults]
Init1 = AT
Init2 = AT+IPR=115200
Init3 = ATQ0 V1 E1 S0=0 &C1 & D2
Init4 = AT+CFUN=1
Init5 = AT+SAPBR=1,1
Init6 = AT+SAPBR=2,1
Init7 = AT+CSST="apn","user","pass"
Init8 = AT+CIIR
Modem Type = USB Modem
Baud = 115200
Dial Command = ATD
New PPPD = yes
Modem = /dev/ttyUSB0
ISDN = 0
Phone = *99#
Stupid Mode = 1
Username = user
Password = pass
```

Nota: Este archivo tiene los comandos necesarios del módem para la conexión.

Este archivo tiene todas las configuraciones necesarias para el módem, que son enviadas a través de comandos AT, cada línea hace lo siguiente:

[Dialer Defaults]: Define una sección de configuración para los parámetros de marcado predeterminados.

Init1 = AT: Envío del comando de atención (AT) para inicializar el módem.

Init2 = AT+IPR=115200: Configura la tasa de baudios del módem a 115200.

Init3 = ATQ0 V1 E1 S0=0 &C1 & D2: Varios comandos de inicialización:

- ATQ0: Habilita los mensajes de respuesta.
- V1: Configura las respuestas en formato textual.
- E1: Habilita el eco de los caracteres enviados.
- S0=0: Configura el número de timbres antes de responder automáticamente (0 para deshabilitar).
- &C1: Control de la línea DCD.

- &D2: Configura la acción del módem ante una señal DTR baja.
- Init4 = AT+CFUN=1: Configura el nivel de funcionalidad del módem.
- Init5 = AT+SAPBR=1,1: Abre un contexto de portador (bearer).
- Init6 = AT+SAPBR=2,1: Consulta el contexto de portador.
- Init7 = AT+CSST="apn","user","pass": Configura el perfil de datos (APN, usuario y contraseña).
- Init8 = AT+CIIR: Comando específico de inicialización adicional.
- Modem Type = USB Modem: Especifica que el tipo de módem es USB.
- Baud = 115200: Establece la tasa de baudios a 115200.
- Dial Command = ATD: Comando de marcado.
- New PPPD = yes: Indica que se utilizará una nueva instancia de PPPD.
- Modem = /dev/ttyUSB0: Especifica el dispositivo del módem.
- ISDN = 0: Indica que no se utilizará ISDN.
- Phone = *99#: Número de teléfono a marcar.
- Stupid Mode = 1: Modo estúpido (opción para módems que no manejan correctamente ciertos comandos).
- Username = user: Nombre de usuario para la conexión.
- Password = pass: Contraseña para la conexión.

Ahora que tenemos el archivo con todo lo necesario para la configuración, es necesario un script que envíe cada uno de los parámetros, para ello se utiliza un Chatscript que es un tipo de programa que interactúa con parámetros predefinidos y respuestas es- peradas hacia otro dispositivo, esto con el fin de automatizar la conexión, el Chatscript utilizado es el siguiente:

Figura 26

Chatscript utilizado.

```
# File = /etc/chatscripts/provider
# La APN se pasa automaticamente desde el archivo de configuracion
# A traves de la variable \T
TIMEOUT 10
ABORT BUSY
ABORT 'NO ANSWER'
ABORT 'NO CARRIER'
ABORT VOICE
ABORT 'NO DIAL TONE'
ABORT 'ERROR'
SAY 'Comenzando script de conexion GPRS'
'''' 'ATZ'
OK 'ate0v1'
SAY 'Configurando APN\n'
OK 'at+cgdcont=1, 'IP', '\T'
ABORT 'NO CARRIER'
SAY 'Marcando...'
OK 'ATD*99***1#'
CONNECT
```

Nota: Este archive automatiza el proceso de conexión a través del módem.

Timeouts y Abortos:

- TIMEOUT 10: Establece un tiempo de espera de 10 segundos para recibir una respuesta antes de reintentar o abortar.
- ABORT BUSY Aborta el script si el módem responde con "BUSY".
- ABORT 'NO ANSWER': Aborta el script si no hay respuesta.
- ABORT 'NO CARRIER': Aborta el script si no hay portadora (conexión de módem).
- ABORT VOICE: Aborta el script si se detecta una llamada de voz.
- ABORT 'NO DIAL TONE': Aborta el script si no hay tono de marcado.
- ABORT 'ERROR': Aborta el script si hay un error.

Mensajes Informativos:

- SAY 'Comenzando script de conexión GPRS': Muestra el mensaje Comenzando script de conexión GPRS".

Comandos AT:

- 'ATZ': Envía el comando ATZ que restablece el módem a su configuración predeterminada.
- OK 'ate0v1': Envía el comando ATE0V1 que configura el módem para no mostrar comandos en modo eco y responde con “OK".
- SAY 'Configurando APN\n': Muestra el mensaje "Configurando APN".
- OK 'at+cgdcont=1, 'IP', '\T': Configura el contexto PDP (Packet Data Protocol) del módem con el APN especificado por la variable \T. Este comando es crucial para establecer la conexión de datos.

Procedimiento de Marcado:

- SAY 'Marcando...': Muestra el mensaje "Marcando...".
- OK 'ATD*99***1#': Envía el comando de marcado para iniciar la conexión GPRS.
- CONNECT: Indica que la conexión ha sido establecida.

Finalmente, para establecer la conexión, es necesario un último archivo de configuración, que esta vez contiene las opciones de nuestra conexión, este archivo se encuentra por defecto en la ruta: "/etc/ppp/peers/provider", y su contenido es el siguiente:

Figura 27

Configuración de la conexión.

```
# file: /etc/ppp*peers/provider
#
# Modem
/dev/ttyUSB0 115200
# Opcinoes de la conexion PPP
noauth
persist
nodeatach
crtscts
local
debug
noipdefault
netmask 255.255.255.0
defaultroute
# Logs
logfile /etc/ppp/ppp-logfile
record /etc/ppp/ppp-dumplog
connect "/usr/sbin/chat -v -T orange -f /etc/chatscripts/provider"
```

Nota: Este archivo contiene la configuración de la conexión PPP

Cada una de las líneas de este archivo realiza la siguiente tarea:

Módem y Velocidad de Conexión:

- /dev/ttyUSB0 115200: Define el dispositivo del módem (en este caso, /dev/ttyUSB0) y la velocidad de conexión (115200 baudios).

Opciones de Conexión PPP:

- noauth: No requiere autenticación para establecer la conexión.
- persist: Mantiene la conexión activa y reintenta si se desconecta.
- nodeatach: No se desvincula del terminal.
- crtscts: Usa control de flujo de hardware.
- local: Evita que PPPD ajuste el estado del módem.
- debug: Activa el modo de depuración para obtener más información en los logs.

- noipdefault: No establece una dirección IP predeterminada.
- netmask 255.255.255.0: Define la máscara de red.
- defaultroute: Establece la ruta predeterminada para la conexión.
- Logs:
- logfile /etc/ppp/ppp-logfile: Define la ubicación del archivo de log.
- record /etc/ppp/ppp-dumplog: Registra el tráfico PPP en un archivo.

Comando de Conexión:

- connect /usr/sbin/chat -v -T orange -f /etc/chatscripts/provider": Usa el comando chat para ejecutar el script de conexión (/etc/chatscripts/provider) con detalles de depuración (-v) y establece la variable APN (-T orange).

3.4.2. Ejecución de la Conexión Como Servicio

Ahora que tenemos todo lo necesario para iniciar una conexión a internet a través de PPP utilizando el módem como gateway, es necesario automatizar el proceso de la misma, de modo que se logre una conexión cada vez que se inicie el dispositivo sin necesidad de ejecutar manualmente el chatscript, para lograrlo, se debe hacer una ejecución como servicio, es decir, indicarle a nuestro sistema que esta conexión es un servicio de Linux, y por lo tanto debe ejecutarse junto con todo el sistema cada vez que este inicie, se logra creando un archivo de servicio en la ruta "/etc/systemd/system". Para nuestro caso, se crearon 2 servicios: "iot_device_service" para la ejecución del reconocimiento como un servicio, y "pppd_service", para la conexión. El archivo del servicio de reconocimiento es el siguiente:

Figura 28

Código del servicio de reconocimiento.

```
[Unit]
Descritpion=Iot Device
After=multi_user.target
[Service]
Type=idle
ExcecStart=/bin/bash -c "cd /home/user/Recogntion && /usr/bin/python3
reconocimento.py"
StandardOutput=journal+console
StandardError=journal+console
[Install]
WantedBy=multi-user.target
```

Nota: Este servicio hace que el código principal se ejecute en cada arranque del sistema de manera automática.

Donde cada línea hace lo siguiente:

Sección [Unit]:

- Description=Iot Device: Proporciona una descripción breve del servicio.
- After=multi_user.target: Especifica que este servicio debe iniciarse después del objetivo multi_user.target, que es alcanzado cuando el sistema ha iniciado múltiples usuarios.

Sección [Service]:

- Type=idle: Indica que el servicio debe iniciarse cuando el sistema está ocioso.
- ExecStart=/bin/bash -c d /home/user/Recognition && /usr/bin/ python3 reconocimiento.py": Define el comando que se ejecuta para iniciar el servicio. En este caso, se cambia el directorio a /home/user/Recognition y luego se ejecuta el script reconocimiento.py con Python 3.

- StandardOutput=journal+console: Redirige la salida estándar al journal y la consola.
- StandardError=journal+console: Redirige los errores estándar al jour- nal y la consola.

Sección [Install]:

- WantedBy=multi-user.target: Especifica que este servicio debe iniciarse cuando el objetivo multi-user.target está activo. Esto significa que el servicio se iniciará en el nivel de ejecución multiusuario.

  Y el servicio de la conexión es el siguiente:

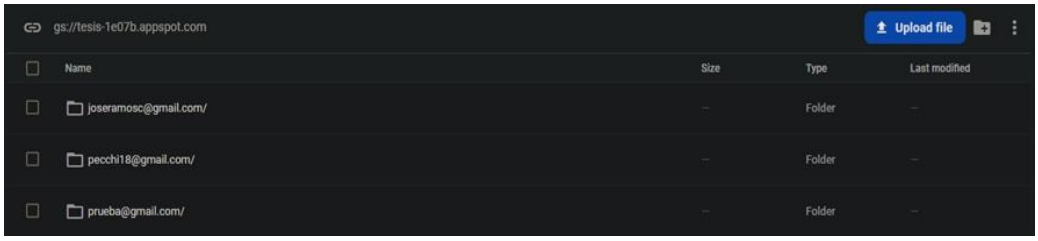
  Como podemos observar, este archivo es muchísimo más sencillo que el anterior, sola- mente debemos ejecutar el comando call y Linux ejecutará automáticamente el chatscript anteriormente mencionado con todas las configuraciones relacionadas.

3.5. BASE DE DATOS

Como se mencionó anteriormente, gracias a Firebase nos ahorramos el trabajo de la programación backend del sistema, teniendo amplia documentación y facilidad de configuración y montaje en cualquier aplicación gracias a un archivo json que con- tiene todo lo necesario. De igual forma explicado líneas arriba, el sistema asegura que cada usuario registrado tenga acceso únicamente a un solo contenedor de datos que se le asigna dentro de la estructura de la base de datos, la organización resultante es la siguiente:

Figura 29

Organización de usuarios dentro de la base de datos

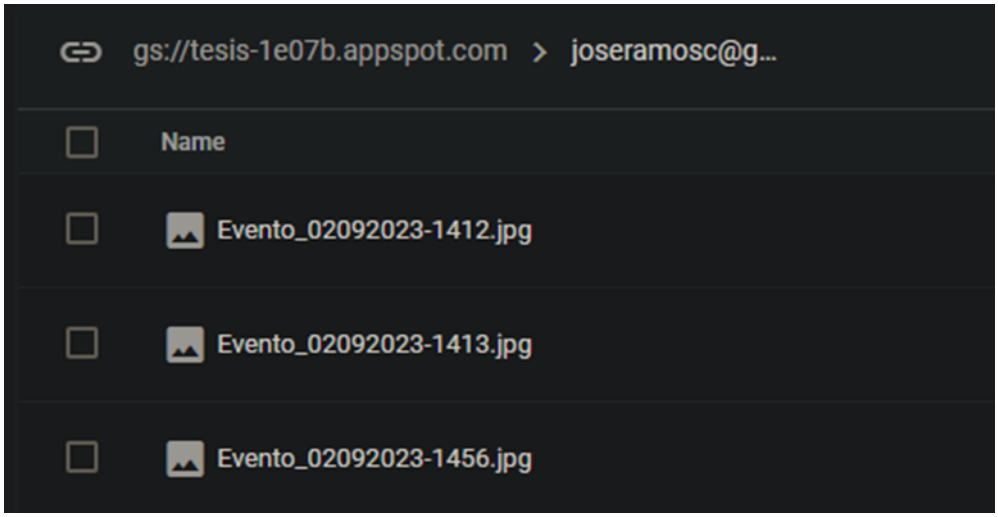


Nota: En cada carpeta se guardan los reportes individuales de cada usuario

Como se puede apreciar en 9, cada usuario tiene una carpeta dentro de esta estructura, ahora veamos lo que hay dentro de cada carpeta (o contenedor):

Figura 30

Contenedor de un usuario



Nota: Cada usuario tiene acceso solo a las evidencias del dispositivo.

Tal como se esperaba, cada usuario tiene acceso solo a las evidencias del dispositivo que está asignado a su cuenta

3.6. DISEÑO DEL CIRCUITO

Se diseñó una tarjeta PCB de prototipo tipo HAT, que integra tanto el modem como la interfaz USB, y un botón integrado de encendido y apagado del sistema, cuyo funcionamiento se detalla a continuación:

3.6.1. Etapa de Potencia

Para la etapa de potencia del circuito, se prefirió que todo sea alimentado a tra- vés de la salida de 5V DC de la Raspberry Pi, sin embargo, es necesario modular este voltaje apropiadamente a través de un regulador para alimentar al módulo GPRS, cuya hoja de datos nos especifica una alimentación de entre 3.4 a 4.4V DC. Afortunadamente, existe en el mercado una amplia gama de reguladores que cumplen esta función, habiendo sido escogido el regulador conmutado "LM2596S-ADJ" en formato de en- capsulado "NOPB" para esta tarea, ya que tiene una salida variable, cuyo ajuste veremos a continuación, y puede soportar hasta 3A de corriente, suficiente para los 2A pico que nos indica la hoja de datos del módulo. Según la hoja de datos del regulador seleccionado, para ajustar el voltaje necesitamos 2 resistencias, de las cuales se recomienda que una sea de 1K Ohm, con un 1 % de tolerancia, siendo la fórmula para calcular la segunda resistencia la siguiente:

$$R_2 = R_1 \left(\frac{V_{out}}{V_{ref}} - 1 \right)$$

De donde, siempre según la hoja de datos, el voltaje de referencia es igual a 1.23V, el voltaje de salida para nuestro caso 4V, y la resistencia 1 es la recomendada de 1KΩ. Datos con los cuales, la resistencia 2 necesaria para obtener 4 voltios a la salida es de 2252.03Ω, sin embargo, se utilizará el valor comercial

de 2.2KΩ. Se utiliza como filtro de entrada a un capacitor electrolítico de 470µF, y a la salida un capacitor electrolítico también de 220µF, esto asegurará una salida estable del voltaje y fueron seleccionados según la hoja de datos. Se utiliza también un diodo Schottky de barrera, con soporte de hasta 5 amperios, con el código SS54, y una bobina de 68µH seleccionada también de la hoja de datos. Todo este arreglo nos dará una salida estable de aproximadamente 4V DC para alimentar al módulo. El módulo tiene un capacitor de tantalio de 100µF en paralelo con uno de 100nF cerámico, esto para evitar las subidas repentinas de voltaje a las que es propenso, este arreglo en paralelo es el filtro que se conecta entre la salida del regulador y el pin de alimentación del módulo.

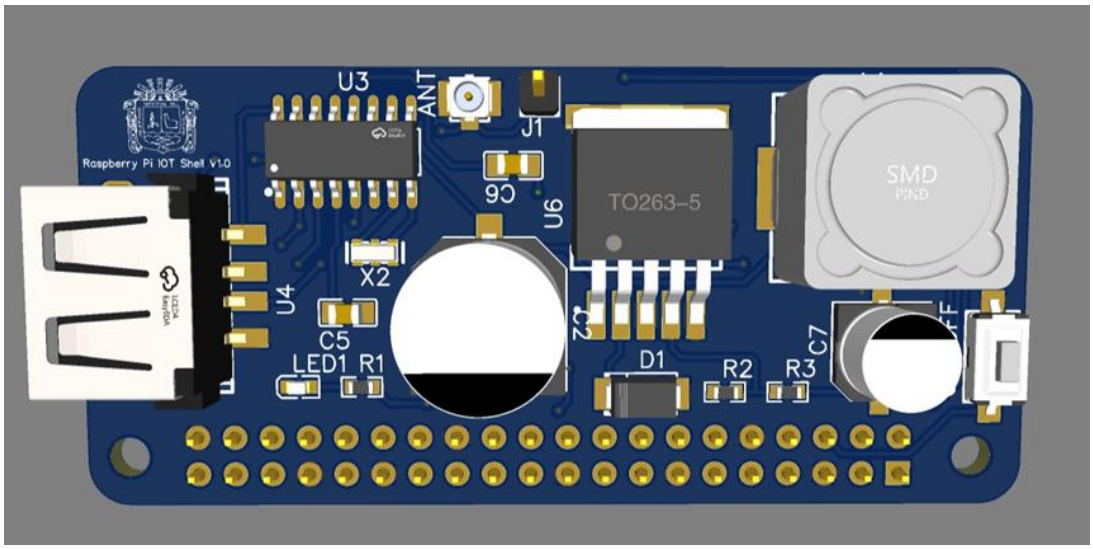
3.6.2. Etapa de Comunicación

Como ya se mencionó anteriormente, durante el desarrollo se observó que la comunicación USB es mucho más estable que la comunicación UART, por lo que para la etapa de comunicación se está utilizando un conversor CH340G de UART a USB, el cual tiene una salida de 5V a través de la alimentación del terminal USB 2.0 añadido y que servirá de interfaz entre el módulo y la Raspberry Pi. Este conversor cuenta con un oscilador de cuarzo de 12MHz, seleccionado según su hoja de datos, y cuenta con 2 capacitores filtro cerámicos, de 100nF entre los pines de reseteo y salida de voltaje de 3.3V, el segundo es solamente conectado a tierra ya que no estamos utilizando los 3.3V en ningún lado, y el segundo es el desacople entre el pin de reseteo del módulo con el pin de reseteo del conversor. Según la hoja de datos, el módulo es capaz de soportar un nivel lógico de comunicación de 5V, por lo que no debería haber ningún problema al conectar directamente los pines de transmisión y recepción al conversor. La tarjeta SIM se comunica con el módulo a través del pin de datos, y emite también una señal de

clock hacia el mismo, estos pines son los pines 6 y 3 del socket, respectivamente, en el módulo, son los pines 14 y 55, respectivamente también. La ventaja de añadir la interfaz USB es que de esta manera también podemos instalar firmwares a través de una computadora al módulo, dependiendo de las necesidades, y sirve para la depuración del mismo a través de software como “PuTTY”. Finalmente, se tiene el siguiente modelo en 3D de la tarjeta:

Figura 31

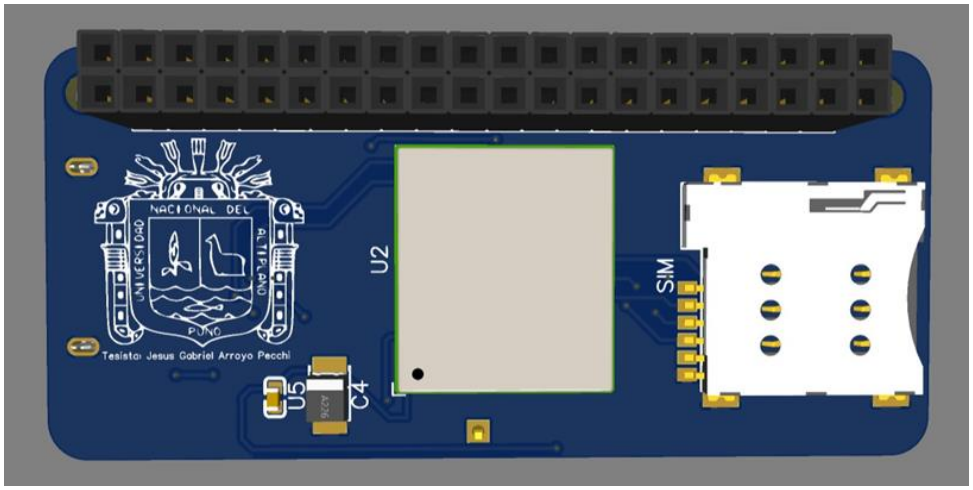
Modelo 3D de la tarjeta, anverso



Nota: Vista 1 de la tarjeta.

Figura 32

Modelo 3D de la tarjeta, reverso



Nota: Vista 2 de la tarjeta.

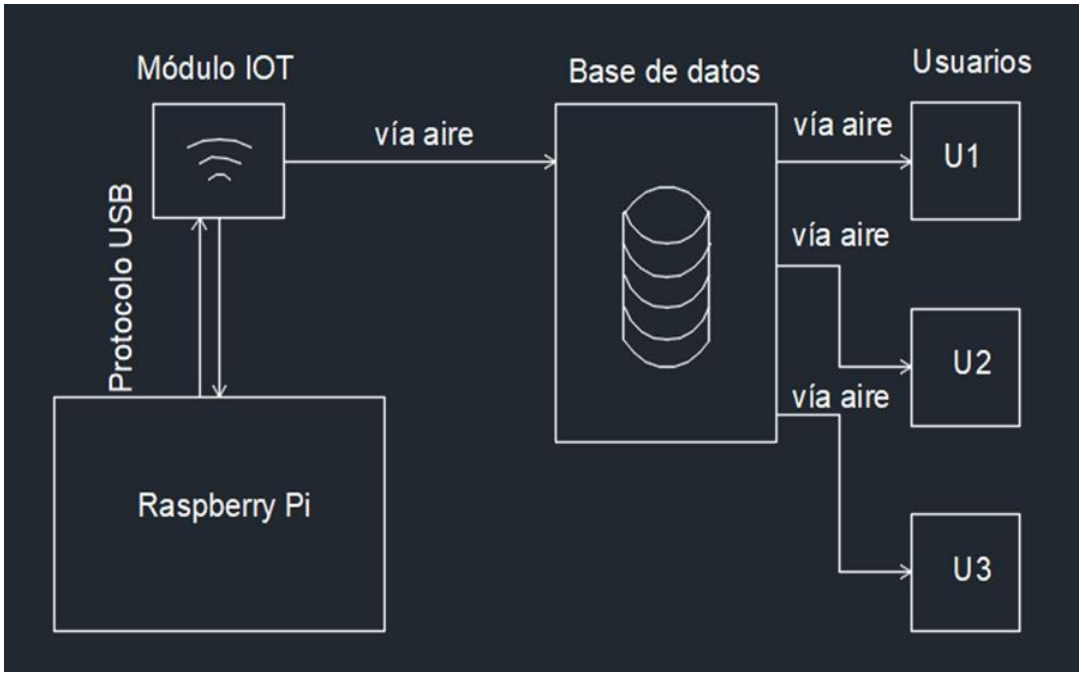
Como se puede apreciar, el diseño de la tarjeta es bastante sencillo, esto gracias a la abundancia de documentación de todos los componentes, y a la potencia de los softwares de diseño asistido actuales que nos permiten hacer análisis y simulaciones de las conexiones

3.7. DIAGRAMA DE BLOQUES RESUMEN

Finalmente, tenemos el siguiente diagrama de bloques que resume todo el sistema ya explicado a grandes rasgos:

Figura 33

Diagrama de bloques resumen



Nota: Vista del diagrama.

Recapitulando, de manera resumida con ayuda del gráfico, lo que ocurre es lo siguiente:

La Rpi ejecuta el programa de reconocimiento, y si se detecta alguna ausencia, se envía una captura a la base de datos a través del módem.

El módem envía los datos a través del aire hacia la base de datos, donde se guardan en un contenedor exclusivo del usuario con el que el dispositivo está relacionado a través de su usuario y contraseña.

La base de datos es accesible a través de la aplicación móvil desarrollada a manera de HMI por cada usuario, dando cierta libertad de acciones como borrar y descargar las imágenes.

CAPÍTULO IV

RESULTADOS Y DISCUSIÓN

4.1. RESULTADOS

Las muestras del presente trabajo se obtuvieron durante el evento “Raspberry Jam Lima 2023”, realizado el día 12 de agosto del 2023 en la ciudad de Lima, se obtuvieron más muestras de un prototipo montado en el laboratorio de control y automatización de la Escuela Profesional de Ingeniería Mecánica Eléctrica de la Universidad Nacional del Altiplano de Puno. Con un total de 222 muestras. El objetivo es el de determinar la precisión del prototipo funcional diseñado, tomando como aceptable un 60 % del total de las muestras. Se considera como muestra a una imagen contenida en una de las carpetas de un usuario dentro de la base de datos, significando esto que pasó por todo el proceso explicado anteriormente.

4.1.1. Cálculo de la Precisión del Sistema

Se utilizó una revisión visual manual de cada una de las muestras a modo de validación para así poder determinar la precisión de los resultados obtenidos del dispositivo, obteniendo 3 principales parámetros:

Falso Positivo: El dispositivo identificó erróneamente a una persona o elemento que desencadenó una alerta.

Figura 34

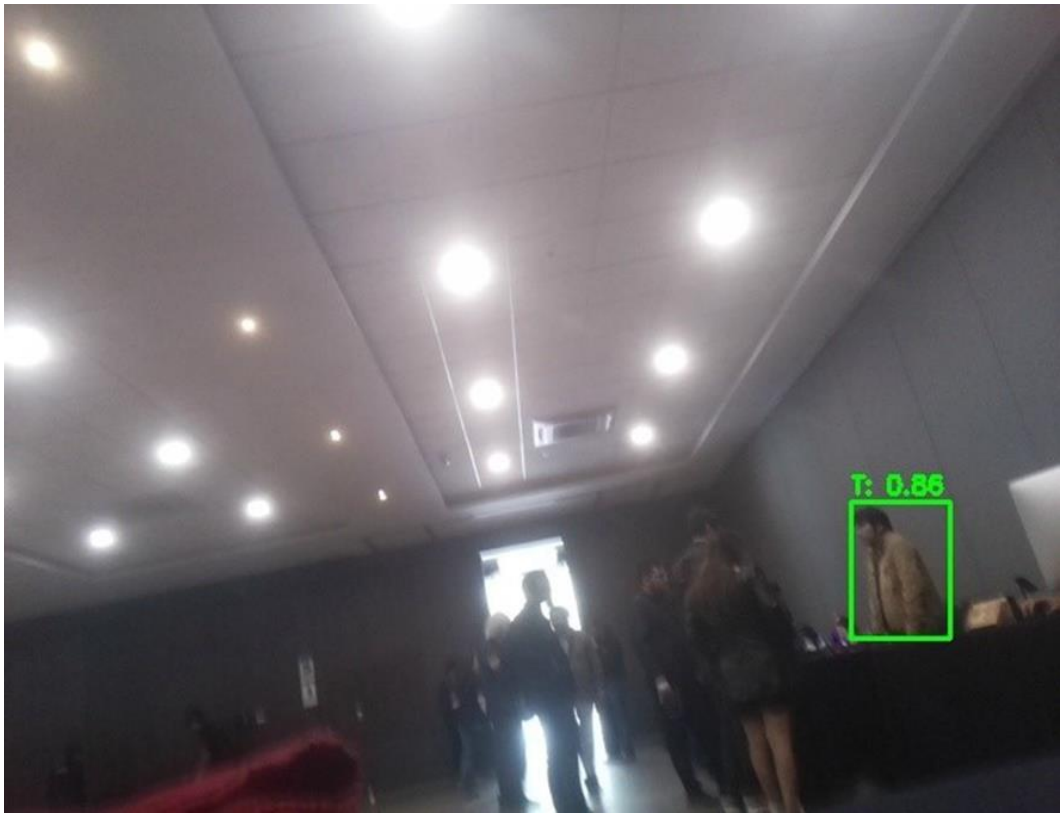
Falso Positivo

Nota: Vista del falso positivo.

- Falso Negativo El dispositivo no identificó a uno o más elementos, desencadenando una alerta.

Figura 35

Falso Negativo

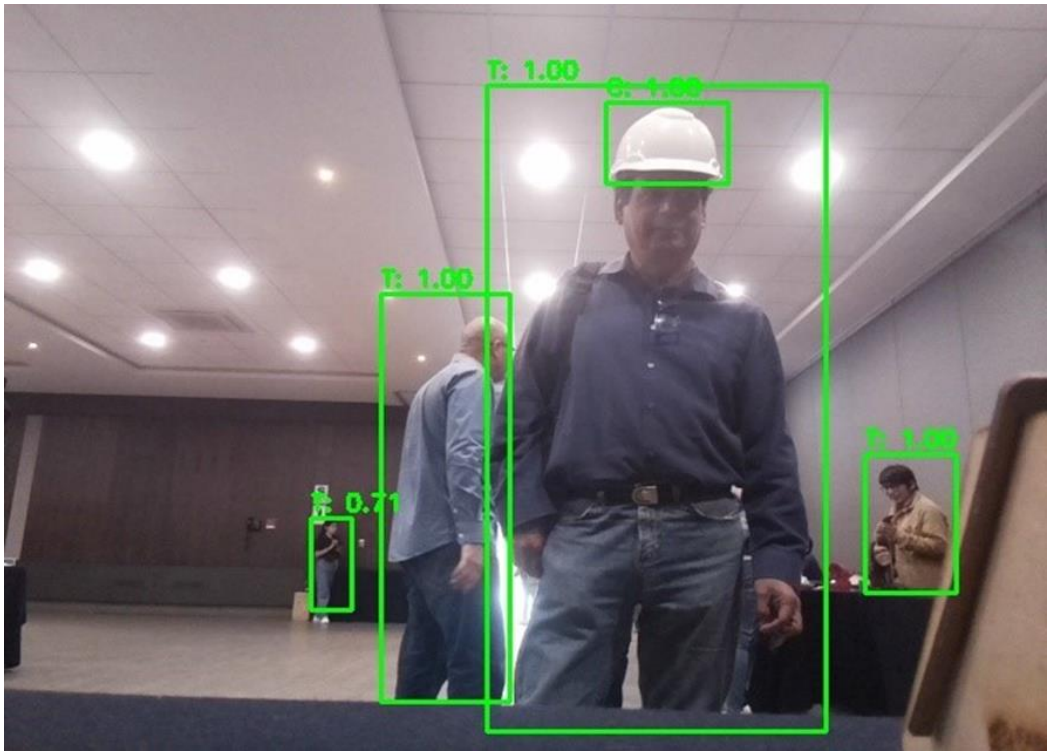


Nota: Vista del falso negativo.

- Correcto El dispositivo identificó todos los elementos correctamente, generó una alerta, y esta alerta fue validada correctamente.

Figura 36

Correcto



Nota: Vista del Correcto.

Para probar la precisión de nuestro sistema, utilizaremos la prueba z, de modo que podamos determinar si la proporción observada de un evento en una muestra es significativamente diferente de una proporción hipotética. Para esta prueba utilizaremos las siguientes fórmulas:

$$p = \frac{C}{C + FP + FN}$$

Donde: C son las muestras validadas como correctas, FP las validadas como falso positivo, y F N las validadas como falso negativo. La ecuación del valor de z es:

$$Z = \frac{p - \pi_0}{\sqrt{\frac{\pi_0(1 - \pi_0)}{n}}}$$

Donde n es el número total de muestras, y π0 es nuestra precisión esperada bajo la hipótesis nula, como se mencionó en un principio, se espera una precisión aproximada del 60 % para considerar al prototipo como funcional. Teniendo en cuenta esto, consideraremos como hipótesis nula a la precisión del sistema del 60 % o 0.6, y como hipó- tesis alternativa a la precisión mayor de 0.6. Los datos de nuestras muestras, validados mediante inspección visual son:

- FP = 44
- FN =18
- C = 160

Procederemos a calcular la precisión del sistema con el uso de la ecuación mencionada, lo que nos da una precisión de p ≈ 0,7207 Bastante buena para lo esperado.

4.1.2. Prueba de Hipótesis

De nuevo, formularemos la hipótesis nula (H0) y la hipótesis alternativa (H1) pa- ra determinar el factor z que nos dirá si la precisión observada es significativamente diferente de un valor de referencia, para nuestro caso 0. 6. Utilizando 4.2:

$$\frac{0{,}7207 - 0{,}6}{\sqrt{\frac{0{,}6(1-0{,}6)}{222}}}$$

El valor de z nos da 3.6709, bastante elevado. Tomaremos un valor de significancia de $\alpha = 0{,}05$, consultando una tabla de distribución estándar, observamos que el valor de probabilidad para el z hallado es de ≈ 0,0002, dado

que $p \ll \alpha$, podemos concluir que hay suficiente evidencia para poder rechazar la hipótesis nula y hay fuerte evidencia a favor de la hipótesis alternativa. De donde concluimos finalmente que el la precisión hallada es significativamente diferente a la hallada, siendo ésta mayor.

4.2. DISCUSIÓN

Los resultados encontrados se relacionan cuantitativa y directamente con la hipótesis planteada, mostrando éstos un nivel aún mayor de precisión que la planteada inicialmente como aceptable.

Respecto a estudios previos, podemos notar lo siguiente:

Márquez-Sánchez et al. (2021) Obtuvo una precisión del 81 %, "Evidenciando la efectividad de la red neuronal para monitoreo y alertas en tiempo real". Su aproximación es aún más sofisticada debido al uso de un dispositivo NVIDIA Jetson Nano, una computadora especialmente diseñada para el cómputo de algoritmos de inteligencia artificial, respecto a ello podemos decir que se obtuvieron muy buenos resultados utilizando un dispositivo de características mucho mas básicas.

De Smith and Doe (2020), "Se encontró que el sistema IoT diseñado con artículos portables logró una reducción del 25 % del mal uso de EPPs durante un periodo de 3 meses". Si bien es una aproximación distinta al problema, es similar en el fondo, encontrando que efectivamente un sistema de alerta temprana es capaz de disminuir el mal uso de los EPPs, y por consecuencia, los accidentes relacionados a esta conducta. Reforzando esta idea, Li and Zhang (2021) logró una reducción del 15 % de accidentes utilizando un sistema similar de alerta temprana

Podemos observar de todas las investigaciones antes citadas, que se utilizan métodos mucho más sofisticados, como el caso del uso de una Jetson Nano, o el diseño exclusivo de EPPs especiales, si bien como pudimos ver estas soluciones son bastante efectivas, representan una mayor inversión monetaria que la solución presentada en este trabajo, ofreciendo así una alternativa bastante precisa, con los resultados hallados, y bastante económica respecto a diseños sofisticados, de código libre, escalable y personalizable completamente.

La gran fortaleza de este estudio radica en la simplicidad de su aplicación práctica en cualquier lugar, siendo fácilmente desplegable una vez hechas las configuraciones necesarias, por lo que se pudieron tomar muestras controladas en 2 ciudades en paralelo. Podemos tomar como puntos en contra la limitada cantidad de elementos de protección disponibles para su análisis, y del mismo modo a la relativamente pequeña canti- dad de muestras que dificultarían una generalización más completa de los resultados. Para futuros estudios, es imperativo tener en cuenta la capacidad de adaptabilidad del código presentado en este trabajo, ya que es posible entrenar una nueva red neuronal que sea capaz de detectar muchos más elementos de protección. Se recomienda también en la medida de lo posible mejorar la capacidad de conexión del módem que se vaya a utilizar, que pueda soportar generaciones posteriores de comunicación para así lograr una comunicación más rápida, y por lo tanto la posibilidad de incorporar aún más funciones del sistema. De igual forma, se recomienda el utilizar mejores cámaras para maximizar los resultados obtenidos.

En resumen, los resultados muestran de manera analítica que el sistema es viable, por lo que queda como una alternativa de bajo costo escalable a los desarrollos más sofisticados.

V. CONCLUSIONES

Gracias a la prueba de hipótesis realizada, se pudo determinar analíticamente que la precisión de las muestras recogidas, de nuevo, considerando una muestra como una imagen presente en la base de datos dentro de una de las carpetas asignadas a un usuario, es significativamente distinta a nuestra precisión objetivo, significativa- mente superior a la esperada. Podemos decir entonces que el sistema es funcional. Se recogieron también las siguientes observaciones importantes:

La Rpi tiene suficiente poder de cómputo como para realizar tareas complejas como lo es un algoritmo de inteligencia artificial y realizar reportes casi en tiempo real.

Es bastante sencillo comunicarse con un módulo GSM para poder utilizarlo como Gateway IoT. Dada la probada capacidad de la Rpi y la facilidad de utilizar uno de estos módulos, podemos decir que sería posible utilizar la Rpi sin problemas para aplicaciones industriales con funcionalidad IoT.

Si bien la precisión alcanzada es bastante buena y podemos considerar al prototipo diseñado como completamente funcional, aún podemos alcanzar mejores resultados, ésto entrenando aún más la red y ampliando su rango de detección de EPPs para poder detectar la totalidad de los mismos.

VI. RECOMENDACIONES

Para optimizar al máximo los resultados, se sugiere lo siguiente:

Maximizar la calidad de la cámara. Esto no implica aumentar el tamaño de la imagen procesada, sino más bien procurar que la calidad de la imagen que se obtenga de la cámara sea lo más alta posible.

Optimizar la memoria disponible en la Rpi, esto para lograr una reducción de tiempo de ejecución y procesado de cada imagen, ya que al trabajar con un len- guaje interpretado como Python, no se llega a aprovechar la capacidad del siste- ma al máximo como lo haría un lenguaje compilado como C.

Para el futuro de la investigación, se propone lo siguiente:

Ampliar las funcionalidades de la red neuronal para poder clasificar la mayor cantidad posible de EPPs

En la medida de lo posible, migrar el sistema a un lenguaje compilado como C o C++ a fin de aprovechar al máximo los limitados recursos de un sistema embebi- do.

Mejorar la capacidad de comunicación del sistema a través de módems multi- banda que permitan utilizar los espectros de frecuencias desde GPRS hasta 5G para mejorar significativamente el envío de datos a la base de datos.

VII. REFERENCIAS BIBLIOGRÁFICAS

Akhtar, N., & Ragavendran, U. (2020). Interpretation of intelligence in CNN-pooling processes: a methodological survey. *Neural Computing and Applications*, *32*(3), 879–898. https://doi.org/10.1007/s00521-019-04296-5

Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., & Ayyash, M. (2015). Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Communications Surveys and Tutorials*, *17*(4), 2347–2376. https://doi.org/10.1109/COMST.2015.2444095

Bai, Y. (2022). RELU-Function and Derived Function Review. *SHS Web of Conferences*, *144*, 02006. https://doi.org/10.1051/shsconf/202214402006

Castillo, A. (2020). *Diseño e implementación de un sistema de detección mediante radiofrecuencia para la identificación de equipamiento de seguridad en el área de manufactura de la empresa HAMPI industrial*. https://repositorio.utp.edu.pe/bitstream/handle/20.500.12867/3769/Anthor Castillo_Tesis_Titulo Profesional_2020.pdf?sequence=1&isAllowed=y

Durango Vanegas, C. E. (2014). Asociación de Datos Espacio-Temporales en Bases de Datos ORACLE. *Ingenierías USBMed*, *5*(2), 100–108. https://doi.org/10.21500/20275846.316

Fathulloh, M. I., Takwim, A., Permadi, D. A., & Rizaldy, S. (2021). *Keamanan Kandang Ayam Dengan Google Firebase Berbasis Internet of Things)*. *03*(01), 38–47.

Guresen, E., & Kayakutlu, G. (2011). Definition of Artificial Neural Networks with comparison to other networks. *Procedia Computer Science*, *3*, 426–433.

https://doi.org/10.1016/j.procs.2010.12.071

Jiao, J., Zhao, M., Lin, J., & Liang, K. (2020). A comprehensive review on convolutional neural network in machine fault diagnosis. *Neurocomputing*, 417:36–63. https://www.sciencedirect.com/science/article/abs/pii/S092523122031225X

Jiju, K., Ramesh, P., Brijesh, P., & Sreekumari, B. (2014). Development of Android based on-line monitoring and control system for Renewable Energy Sources. *I4CT 2014 - 1st International Conference on Computer, Communications, and Control Technology, Proceedings, September*, 372–375. https://doi.org/10.1109/I4CT.2014.6914208

Kadiyala, E., Meda, S., Basani, R., & Muthulakshmi, S. (2017). No Title. *Monitorización Global de Procesos Industriales a Través de IoT Con Raspberry Pi*, 260–262. https://www.researchgate.net/publication/320652678_Global_industrial_process_monitoring_through_IoT_using_Raspberry_pi

Karthikeyan, S., Raj, R. A., Cruz, M. V, Chen, L., Vishal, J., & Rohith, V. (2023). A systematic analysis on raspberry pi prototyping: Uses, challenges, benefits, and drawbacks. *IEEE Internet of Things Journal*. https://www.studocu.com/pe/document/universidad-nacional-de-ingenieria/administracion-de-redes/a-systematic-analysis-on-raspberry-pi-prototyping-uses-challenges-benefits-and-drawbacks/75639120

Lin, Y.-B., & Chlamtac, I. (2001). Wireless and mobile network architectures. *Wiley*. https://www.amazon.com/Wireless-Mobile-Architectures-Yi-Bing-Lin/dp/0471394920

Maduranga, P. (2023). *GPRS Enabled Low-Power Sensor Node Design For Outdoor IoT*

Applications. https://doi.org/10.36227/techrxiv.21733358.v1

Massiris, M., Fernández, J. A., Bajo, J., & Delrieux, C. (2021). Sistema automatizado para monitorear el uso de equipos de protección personal en la industria de la construcción. *Revista Iberoamericana de Automática e Informática Industrial*, *18*(1), 68. https://doi.org/10.4995/riai.2020.13243

Nath, N., Behzadan, A., & Paal, S. (2020). Deep learning for site safety: Real- time detection of personal protective equipment. *Automation in Construction*. https://www.sciencedirect.com/science/article/abs/pii/S0926580519308325

Parker, M., & Dhanani, S. (2012). *Digital Video Processing for Engineers: A Foundation for Embedded Systems Design* (Newnes). https://books.google.com.pe/books/about/Digital_Video_Processing_for_Engineers.html?id=LKGOXhVzibUC&redir_esc=y

Peñaranda, J. M., Amado, C. D., Mora, S. B., & Casadiego, S. A. (2016). Servidor web y punto de acceso basado en un sistema embebido para la supervisión de un proceso desde una aplicación móvil con sistema operativo Android. *Ingenium*, *10*(27), 11. https://doi.org/10.21774/ing.v10i27.616

Sangiovanni-Vincentelli, A., & Di Natale, M. (2007). Embedded system design for automotive applications. *Computer*, *40*(10), 42–51. https://doi.org/10.1109/MC.2007.344

Sethi, P., & Sarangi, S. (2017). Internet Of Things: Architecture,Issues and Applications. *International Journal of Engineering Research and Applications*, *07*(06), 85–88. https://doi.org/10.9790/9622-0706048588

Simpson, W. (1994). *The point-to-point protocol (ppp). Suppan*, 1–16.

Sung, W. T., Devi, I. V., & Hsiao, S. J. (2022). Smart Lamp Using Google Firebase as Realtime Database. *Intelligent Automation and Soft Computing*, *33*(2), 967–982. https://doi.org/10.32604/iasc.2022.024664

Torres, L. (1994). Redes neuronales y aproximacíon de funciones. *Boletin de Maternat Icas Nue~a Serie VoU*, *II*(2), 35–58.

Vejlgaard, B., Lauridsen, M., Nguyen, H., Kovacs, I. Z., Mogensen, P., & Sorensen, M. (2017). Coverage and Capacity Analysis of Sigfox, LoRa, GPRS, and NB-IoT. *IEEE Vehicular Technology Conference*, *2017-June*. https://doi.org/10.1109/VTCSpring.2017.8108666

Villamarín, M., Urquinzo, A., & Roberto, V. (2020). *Educación y pedagoía CIDEP - 2020*.

Walke, B. (2013). The roots of GPRS: The first system for mobile packet-based global internet access. *IEEE Wireless Communications*, *20*(5), 12–23. https://doi.org/10.1109/MWC.2013.6664469

Zanella, A., Bui, N., Castellani, A., Vangelista, L., & Zorzi, M. (2014). Internet of things for smart cities. *IEEE Internet of Things Journal*, *1*(1), 1:22–32. https://doi.org/10.1109/JIOT.2014.2306328

Zanin, H., Neris Jr., C., & Gomes, R. (2021). *Estratégias concorrenciais entre sistemas operacionais: Microsoft e Android*. 2142–2156. https://doi.org/10.5151/v-enei-769

Zhang, Y. D., Satapathy, S. C., Guttery, D. S., Górriz, J. M., & Wang, S. H. (2021). Improved breast cancer classification through combining graph convolutional network and convolutional neural network. *Information Processing and ManagemenT*. https://www.sciencedirect.com/science/article/abs/pii/S0306457320309328?via%3

Dihub

ANEXOS

ANEXO 1: Declaracion Jurada de Autenticidad de Tesis.

Universidad Nacional del Altiplano Puno | Vicerrectorado de Investigación | Repositorio Institucional

DECLARACIÓN JURADA DE AUTENTICIDAD DE TESIS

Por el presente documento, Yo Jesus Gabriel Arroyo Pecchi,
identificado con DNI 72577461 en mi condición de egresado de:

☒ **Escuela Profesional,** ☐ **Programa de Segunda Especialidad,** ☐ **Programa de Maestría o Doctorado**
INGENIERIA MECANICA ELECTRICA,
informo que he elaborado el/la ☒ **Tesis** o ☐ **Trabajo de Investigación** denominada:
"Desarrollo de un Sistema de Alerta de Ausencia de Casco de Seguridad usando redes neuronales y Tecnologia IOT"

Es un tema original.

Declaro que el presente trabajo de tesis es elaborado por mi persona y **no existe plagio/copia** de ninguna naturaleza, en especial de otro documento de investigación (tesis, revista, texto, congreso, o similar) presentado por persona natural o jurídica alguna ante instituciones académicas, profesionales, de investigación o similares, en el país o en el extranjero.

Dejo constancia que las citas de otros autores han sido debidamente identificadas en el trabajo de investigación, por lo que no asumiré como suyas las opiniones vertidas por terceros, ya sea de fuentes encontradas en medios escritos, digitales o Internet.

Asimismo, ratifico que soy plenamente consciente de todo el contenido de la tesis y asumo la responsabilidad de cualquier error u omisión en el documento, así como de las connotaciones éticas y legales involucradas.

En caso de incumplimiento de esta declaración, me someto a las disposiciones legales vigentes y a las sanciones correspondientes de igual forma me someto a las sanciones establecidas en las Directivas y otras normas internas, así como las que me alcancen del Código Civil y Normas Legales conexas por el incumplimiento del presente compromiso

Puno 16 de Agosto del 2024

FIRMA (obligatoria)

Huella

ANEXO 2: Autorización para el depósito de Tesis o trabajo de investigación en el Repositorio Institucional.

Universidad Nacional del Altiplano Puno | Vicerrectorado de Investigación | Repositorio Institucional

AUTORIZACIÓN PARA EL DEPÓSITO DE TESIS O TRABAJO DE INVESTIGACIÓN EN EL REPOSITORIO INSTITUCIONAL

Por el presente documento, Yo Jesus Gabriel Arroyo Pecchi,
identificado con DNI 72577461 en mi condición de egresado de:

☒ **Escuela Profesional,** ☐ **Programa de Segunda Especialidad,** ☐ **Programa de Maestría o Doctorado**

INGENIERIA MECANICA ELECTRICA,
informo que he elaborado el/la ☒ **Tesis** o ☐ **Trabajo de Investigación** denominada:

"DESARROLLO DE UN SISTEMA DE ALERTA DE AUSENCIA DE CASCO DE SEGURIDAD USANDO REDES NEURONALES Y TECNOLOGIA IOT."

para la obtención de ☐**Grado,** ☒**Título Profesional** o ☐ **Segunda Especialidad.**

Por medio del presente documento, afirmo y garantizo ser el legítimo, único y exclusivo titular de todos los derechos de propiedad intelectual sobre los documentos arriba mencionados, las obras, los contenidos, los productos y/o las creaciones en general (en adelante, los "Contenidos") que serán incluidos en el repositorio institucional de la Universidad Nacional del Altiplano de Puno.

También, doy seguridad de que los contenidos entregados se encuentran libres de toda contraseña, restricción o medida tecnológica de protección, con la finalidad de permitir que se puedan leer, descargar, reproducir, distribuir, imprimir, buscar y enlazar los textos completos, sin limitación alguna.

Autorizo a la Universidad Nacional del Altiplano de Puno a publicar los Contenidos en el Repositorio Institucional y, en consecuencia, en el Repositorio Nacional Digital de Ciencia, Tecnología e Innovación de Acceso Abierto, sobre la base de lo establecido en la Ley N° 30035, sus normas reglamentarias, modificatorias, sustitutorias y conexas, y de acuerdo con las políticas de acceso abierto que la Universidad aplique en relación con sus Repositorios Institucionales. Autorizo expresamente toda consulta y uso de los Contenidos, por parte de cualquier persona, por el tiempo de duración de los derechos patrimoniales de autor y derechos conexos, a título gratuito y a nivel mundial.

En consecuencia, la Universidad tendrá la posibilidad de divulgar y difundir los Contenidos, de manera total o parcial, sin limitación alguna y sin derecho a pago de contraprestación, remuneración ni regalía alguna a favor mío; en los medios, canales y plataformas que la Universidad y/o el Estado de la República del Perú determinen, a nivel mundial, sin restricción geográfica alguna y de manera indefinida, pudiendo crear y/o extraer los metadatos sobre los Contenidos, e incluir los Contenidos en los índices y buscadores que estimen necesarios para promover su difusión.

Autorizo que los Contenidos sean puestos a disposición del público a través de la siguiente licencia:

Creative Commons Reconocimiento-NoComercial-CompartirIgual 4.0 Internacional. Para ver una copia de esta licencia, visita: https://creativecommons.org/licenses/by-nc-sa/4.0/

En señal de conformidad, suscribo el presente documento.

Puno 16 de Agosto del 2024

FIRMA (obligatoria)

Huella