ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Mecánica y Ciencias de la Producción

Diseño de gafas inteligentes para la traducción de idiomas en tiempo real

PROYECTO INTEGRADOR

Previo la obtención del Título de:

Ingenieros en Mecatrónica

Presentado por:

Kevin Jefferson López De la O

Héctor Alexander Triviño González

GUAYAQUIL - ECUADOR

Año: 2022

DEDICATORIA

El presente proyecto va dedicado a mi madre, Monica Alexandra De la O Quinde, quien me ha dado su apoyo incondicional y siempre se ha esforzado para que nunca nos falte nada a mis hermanos y a mí.

A la memoria de mi padre, Wilson Kleber López Guale, quien me apoyó en casi toda mi carrera universitaria y sé que estaría muy orgulloso de mis logros y de los logros de mis hermanos.

A mis hermanos Wendy López y Nick López, que son mi motor y mi inspiración para continuar adelante.

Kevin Jefferson López De la O

DEDICATORIA

El presente trabajo lo dedico a mi madre Maritza, a mi padre Héctor y a mi hermana Doménica por la motivación, tolerancia y apoyo incondicional que me demuestran día a día.

A mis familiares cercanos que siempre me han extendido una mano en todo momento.

A Daniela, por el aliento, soporte y por participar en los mejores recuerdos que me llevaré de esta etapa.

Héctor Alexander Triviño González

AGRADECIMIENTOS

Nuestro más sincero agradecimiento a los docentes y directivos de ESPOL que han aportado en nuestra formación profesional.

Al Ph.D. Christian Tutivén, y al M.Sc. Bryan Puruncajas por sus directrices, constante colaboración y compromiso en la elaboración del presente proyecto.

Al M.Sc. Efraín Terán, quien como coordinador nos guió en cada paso durante la carrera.

DECLARACIÓN EXPRESA

”Los derechos de titularidad y explotación, nos corresponde conforme al reglamento de propiedad intelectual de la institución; *Kevin Jefferson López De la O* y *Héctor Alexander Triviño González* damos nuestro consentimiento para que la ESPOL realice la comunicación pública de la obra por cualquier medio con el fin de promover la consulta, difusión y uso público de la producción intelectual”

Firmado electrónicamente por:
HECTOR ALEXANDER
TRIVINO GONZALEZ

Héctor Alexander Triviño González

Firmado electrónicamente por:
KEVIN
JEFFERSON
LOPEZ DE LA O

Kevin Jefferson López De la O

EVALUADORES

Firmado electrónicamente por:
BRYAN JOAO PURUNCAJAS MAZA

Bryan Puruncajas, M.Sc.

PROFESOR DE LA MATERIA

Firmado electrónicamente por:
CHRISTIAN JAVIER TUTIVEN GALVEZ

Christian Tutivén, Ph.D.

PROFESOR TUTOR

RESUMEN

El idioma inglesa es en la actualidad la lengua más hablada a nivel mundial, Estados Unidos, uno de los principales destinos para inmigrar, alberga un 7% de inmigrantes latinos, siendo la barrera de lenguaje un problema afrontado diariamente. El presente trabajo, destaca el desarrollo de un prototipo de un sistema IoWT enfocado a la tarea de traducción de audio en inglés a texto en español en tiempo real con el objetivo de asistir al usuario con los resultados de la traducción. La metodología abordó el diseño del sistema mecánico, electrónico y de software, que permitió adquirir datos de audio mediante un micrófono, procesarlos a través de una API expuesta en la nube de AWS hasta obtener la traducción resultante, que finalmente, se administra empleando el microcontrolador ESP32-WROOM-32, encargado de renderizar el resultado a través de un sistema óptico. La comunicación entre el microcontrolador y el servidor se desarrolló en base al protocolo HTTPS. Los resultados destacan un peso de 103 g y una media de tiempo de procesamiento de 1.9 s considerando una muestra de audio en una frecuencia de muestreo de 12 kHz en un tiempo de muestreo de 2 s.

Palabras Clave: Computación en la nube, IoWT, procesamiento de lenguaje natural, impresión 3D.

ABSTRACT

The English language is currently the most widely spoken language in the world. The United States, one of the main destinations for immigration, is home to 7% of Latino immigrants, the language barrier is a daily problem they face. The present work highlights the development of a prototype of a real-time IoWT system focused on the task of translating audio in English to text in Spanish with the target of assisting the user with the results of the translation. The methodology addressed the design of the mechanical, electronic and software system, which allowed acquiring audio data through a microphone, processing it through an API exposed in the AWS cloud until obtaining the resulting translation, which is finally managed using the microcontroller ESP32-WROOM-32, which is responsible of rendering the result through an optical system. The communication between the microcontroller and the server was developed based on the HTTPS protocol. The results highlight a weight of 103 g and a mean processing time of 1.9 s considering an audio sample at a sample rate of 12 kHz in a sampling time of 2 s.

Keywords: Cloud computing, IoWT, natural language processing, 3D printing.

ÍNDICE GENERAL

ABREVIATURAS

AAC	Advanced Audio Coding
ADC	Analog-to-Digital Converter
AIFF	Audio Interchange File Format
ALAC	Apple’s Lossless Audio Codec
API	Application Programming Interface
ARPANET	Advanced Research Projects Agency Network
AWS	Amazon Web Services
A-MPDU	Aggregate MAC Protocol Data Unit
A-MSDU	Aggregate MAC Service Data Unit
BLE	Bluetooth Low Energy
BR/EDR	Bluetooth basic rate/enhanced data rate
CNN	Convolutional Neural Network
DAC	Digital-to-Analog Converter
DNN	Deep neural network
DSD	Direct Stream Digital
ESPOL	Escuela Superior Politécnica del Litoral
FLAC	Free Lossless Audio Codec
FMI	Fondo Monetario Internacional
GCP	Google Cloud Platform
HPE	Hewlett Packard Enterprise
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IA	Inteligencia artificial
IOM	International Organization for Migration

IoT	Internet of Things
IoWT	Internet of Weareble Things
I2C	Inter-Integrated Circuit
I2S	Integrated Interchip Sound
ML	Machine Learning
MP3	MPEG-1 Audio Layer III
NIST	National Institute of Standards and Technology
NLP	Natural Language Processing
ONU	Organización de las Naciones Unidas
OOS	Open source software
PCM	Pulse-Code modulation
PLA	Ácido poliláctico
REST	Representational state transfer
RNN	Recurrent Neural Network
R2NN	Recursive recurrent neural network
SaaS	System as a Service
S2T	Speech-To-Text
TI	Tecnología de la Información
TLS	Transport Layer Security
UE	Unión Europea
URL	Uniform Resource Locator
WAV	Waveform Audio File

SIMBOLOGÍA

GB	Gigabyte
TB	Terabyte
KB	Kilobyte
m	Metro
mm	Milímetro
in	Pulgada
KHz	KiloHertz
MHz	MegaHertz
dB	Decibel
oz	Onzas
g	Gramo
s	Segundo
ms	Milisegundo
μs	Microsegundo
mA	MiliAmperio
V	Volts

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1

1. INTRODUCCIÓN

La globalización es un fenómeno que se ha evidenciado a través del movimiento de masas; de acuerdo con la IOM, se estimaron cerca de 281 millones de migrantes internacionales en 2020, representando el 3.6% de la población mundial [1]. Un problema abordado como producto de la migración es la barrera de lenguaje, siendo considerada una de las barreras que impiden la comunicación efectiva, hecho que ocurre notablemente entre entes que no tienen el mismo nivel de habilidad en el idioma [2].

La barrera de lenguaje, se puede presentar en distintos ámbitos cotidianos, en el ámbito escolar particularmente, la barrera de lenguaje en los padres de familia repercute de forma negativa en los resultados académicos de niños, afectando de esta manera la escolarización [3], se destaca en adición, que en la industria TI la falta de enfoque en objetivos comerciales, productos defectuosos, innovación reprimida y clientes insatisfechos suelen ser consecuencias de las barreras de comunicación [4], de la misma manera los equipos de trabajo multinacionales, se afectan al limitar las actividades de procesamiento del conocimiento y comunicación [5].

El rápido crecimiento de tecnologías de información y la comunicación son impulsadas por la demanda de los usuarios y el crecimiento de nuevos servicios. En la actualidad, el número de dispositivos portátiles interconectados se ha impulsado enormemente cada año, debido a la incitación de los consumidores y por la gran inclusión del IoT [6]. El significativo impacto de las TIC e IoT, permiten que los desarrolladores centren su atención en un nuevo nicho de mercado, que surge a base de los dispositivos puestos o vestibles por los usuario, que es conocido como IoWT (Internet of Weareble Things).

Desde la presentación de Google Glasses en el 2013 [7], el mercado de lentes inteligentes creció significativamente de tal manera que surgieron competidores como Sony con SmartEyeGlass [8] y ODG R7 [9]. En la actualidad hay distintas opciones como Vuzix Blade [10], Spectacles [11], que incorporan diferentes características, sin embargo no abarcan la traducción en tiempo real. Para esta problemática, en el evento Google I/O 2022, evento albergado por el gigante tecnológico, se anunció el lanzamiento de gafas inteligentes que emplean sus tecnologías de traducción y transcripción, sin embargo, a la

fecha este producto se encuentra en fase de prototipo [12]. A pesar de la discontinuidad de algunas de las gafas mencionadas, el alto precio al consumidor final es el factor común entre las opciones, destacando así que los productos no son fácilmente asequibles.
Este proyecto basa su desarrollo en el diseño de un prototipo de gafas inteligentes cuyo objetivo es asistir con la traducción al usuario. Una vez se inicie la recepción de datos, el dispositivo recepta las palabras emitidas por emisor y mediante una API alojada en la nube, se accede al procesamiento de las señales, que mediante modelos de aprendizaje profundo transcriben las señales de entrada de audio al idioma de salida, particularmente de inglés a español. Una vez finalizado el proceso de traducción la API debe retornar las palabras traducidas con el objetivo de que el dispositivo lo proyecte en la lente de las gafas.

1.1 Descripción del problema

El idioma inglés se destacó como el idioma más hablado a nivel mundial al año 2022 con aproximadamente 1.4 billones de personas capaces de hablarlo, siendo el idioma oficial de 67 países y de algunas organizaciones internacionales como la ONU, UE, FMI, entre otras [13]. En particular, Estados Unidos ha sido considerado el destino principal para migrantes internacionales desde 1970, cuadriplicando la cantidad de inmigrantes residentes a finales del 2020 cuando la población hispana en Estados Unidos alcanzó cerca de 62.1 millones de habitantes, mismo que representa un incremento del 23% respecto a la última década y un 7% de la población general [14, 1].
La barrera de lenguaje es el primer problema que aborda un inmigrante hispano cuando llega a Estados Unidos. Es innegable la cantidad de actividades que se afectan por esta problemática debido a la complejidad que requiere el aprendizaje de una nueva lengua extranjera, este hecho suele enmarcar una amplia brecha en la comunicación efectiva. Siendo Estados Unidos considerado el destino principal para migrantes internacionales y que la comunidad latina representa la mayor proporción de minoría en la comunidad americana, se delimita el problema relacionado a la traducción de lenguaje inglés a español como lenguaje de entrada y salida respectivamente.

1.2 Justificación del problema

Considerando que cada década aumenta el numero de inmigrantes hispanos en Estados Unidos y la dificultad para un inmigrante hispano destacarse en dicho país por limitaciones relacionadas a la barrera de lenguaje, se resalta la necesidad de un dispositivo capaz de asistir a un hablante extranjero, además de considerar el impacto que podría presentar al beneficiar también a usuarios con discapacidades auditivas [14, 15].

La implementación de un dispositivo capaz de asistir a un hablante extranjero ofrece múltiples beneficios, puesto que ya no sería necesario de una tercera persona (intermediario) para mejorar la comunicación entre pares. Además se destaca el tamaño del dispositivo, y considerando que es pequeño, es fácil de incorporar y de usar. Este dispositivo se describe como *wearable*, aspecto que lo caracteriza como vestible [16].

1.3 Objetivos

1.3.1 Objetivo general

Diseñar y desarrollar un prototipo de gafas inteligentes para la traducción de audio a texto de inglés a español en tiempo real.

1.3.2 Objetivos específicos

1. Emplear modelos pre-entrenados para la tarea de transcripción y traducción de máquina mediante librerías de código abierto.
2. Diseñar un dispositivo compacto para la proyección de la traducción en los lentes del usuario mediante un sistema óptico.
3. Exponer una API que permita la comunicación entre el dispositivo y los modelos de transcripción y traducción.

1.4 Marco teórico

A través de los años, el mercado de lentes incrementó de forma significativa desde la presentación de Google Glasses en el 2013 [12]. El producto, presentó distintos módulos, entre estos se destacan la batería, panel táctil, cámara de vídeo, pantalla además de componentes para la conexión Wifi/Bluetooth, códec de audio, entre otros [17]. En los últimos años, se involucraron diferentes tendencias como las tecnologías *wearable* y en paralelo se popularizaron diferentes términos como la nube y el *open source*, mismos que se detallarán posteriormente junto a otras tecnologías.

1.4.1 Tecnología vestible (*wearable*)

La tecnología *wearable* o tecnología vestible, hace referencia a dispositivos electrónicos inteligentes añadidos a la vestimenta y que el usuario puede llevar puesto. Estos dispositivos han existido desde el año 1500 cuando el inventor alemán Peter Henlein creó pequeños relojes que se podían utilizar como collares, y desde entonces se crearon otros tipos como relojes de bolsillos y relojes de manilla. Con el avance de la tecnología, la popularidad de los dispositivos *wearables* ha incrementado de manera exponencial a tal punto de que hoy en día, es comúnmente empleado en la vida diaria y con un sinnúmero de aplicaciones tales como auriculares, pulseras, gafas o relojes inteligentes [16].
En particular, las pulseras y relojes son los *wearables* más populares, que además de brindar información de la hora, suelen integrar algunas otras funciones como el registro de la actividad física, frecuencia cardiaca, calorías quemadas, otras funciones relacionadas con la salud, así como también, contestar llamadas o recibir alguna notificación. Estos dispositivos han tomado un papel importante en la vida cotidiana de las personas, que de acuerdo con la consultora GlobalData, la venta mundial de los dispositivos supera los 97 mil millones de dólares [18].

1.4.2 Computación en la nube

La computación en la nube, es la tecnología que permite el uso de servidores remotos conectados a internet, para administrar, almacenar, procesar datos, redes y software. Todo usuario de internet ha interactuado de forma directa o indirecta con la computación en la nube, por ejemplo al guardar un archivo o visualizar una película en cualquiera de los diferentes proveedores. De acuerdo con la NIST [19], posteriormente se destacan las

características más importantes de la computación en la nube:

- Autoservicio bajo demanda, el consumidor puede manejar las capacidades informáticas, como el almacenamiento y redes.
- Acceso amplio, las capacidades están disponibles a través de la red.
- Rápida elasticidad, los recursos son liberados y aprovisionados respecto a la demanda del consumidor.
- Servicio medido, tienen aplicaciones de medición, para monitorear, controlar e informar al proveedor y al consumidor del servicio utilizado.

Existen diferentes tipos de nubes, la nube pública, donde los servidores pertenecen a un tercero, y se paga por su uso, algunos ejemplos de nubes públicas son Microsoft Azure, Google Cloud Platform (GCP), Amazon Web Services (AWS) [20]. La nube privada, donde los servicios únicamente están disponibles para un grupo de usuarios que tienen acceso a la red privada, siendo esta más segura para almacenar información sensible, algunos ejemplos de nubes privadas son Oracle, IBM, Hewlett Packard Enterprise (HPE). Finalmente, la nube híbrida, es la combinación de la nube pública y privada, el usuario puede escoger donde almacenar la información, siendo Infinidat y Ensono, ejemplos de proveedores [21].

1.4.3 Protocolo de transferencia de hipertexto seguro (HTTPS)

El protocolo HTTP (*Hypertext Transfer Protocol*, Protocolo de transferencia de hipertexto en español) permite la transferencia de información a través de archivos XML, HTML, en la web, HTTPS (*Hypertext Transfer Protocol Secure*, Protocolo de transferencia de hipertexto seguro en español) se basa en el protocolo HTTP, sin embargo permite la comunicación y conexión segura entre el servidor y el cliente agregando una capa de seguridad a los recursos compartidos entre el servidor y el cliente para que no sea interceptada por personas no autorizadas mediante el protocolo de encriptación TLS (*Transport Layer Security*, Seguridad de la capa de transporte en español) [22]. En la Figura 1.1 se destaca un ejemplo ilustrativo que describe la diferencia entre ambos protocolos.

El protocolo HTTP destaca su URL iniciando con "*http://*", a diferencia de la URL de HTTPS que inicia con "*https://*". El protocolo HTTP es inseguro y es propenso a

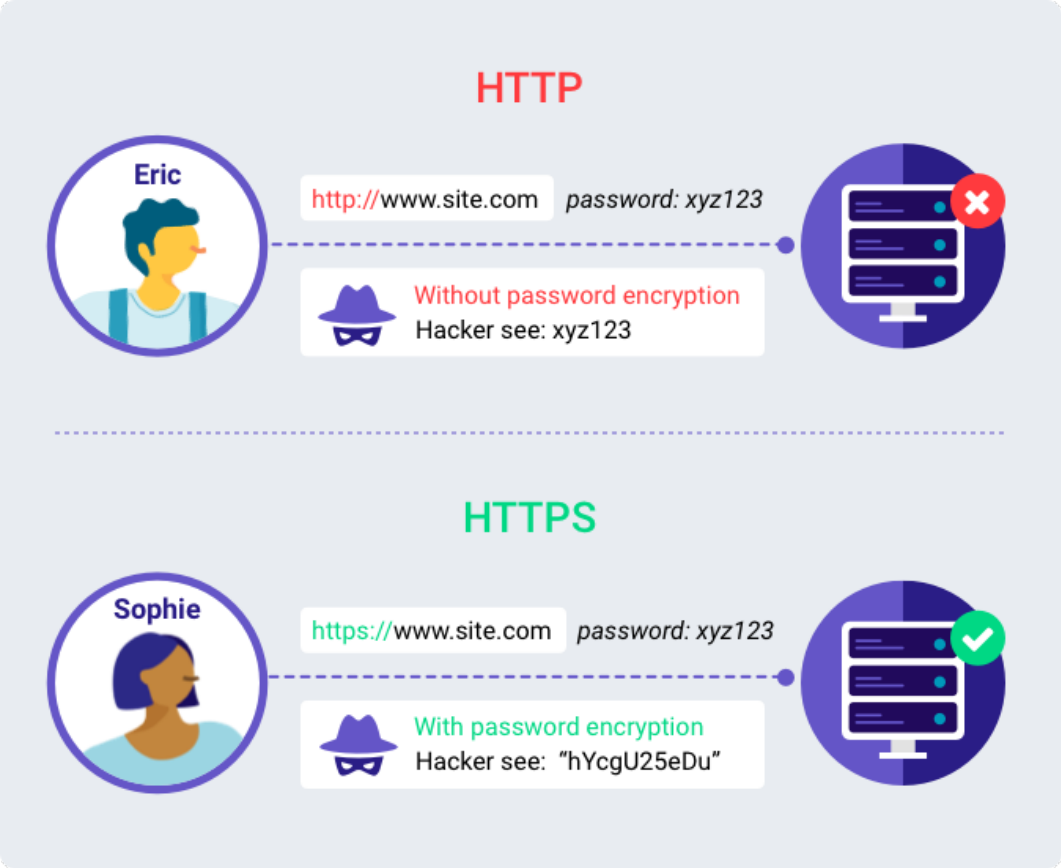


Figura 1.1 Diferencias entre protocolo HTTP y HTTPS [23]

ciberataques de tal manera que, las personas que realizan el ataque pueden obtener información confidencial como se muestra en la parte superior a diferencia de HTTPS que destaca su seguridad y resistencia a esos ataques [22], puesto que se cifra el mensaje como se muestra en la parte inferior de la Figura 1.1.

1.4.4 Software de código abierto (*open source software*)

OOS (*Open source software*), es el término que hace referencia a código abierto en la industria de software, su definición se asocia a un código diseñado para su acceso público de tal manera que, cualquier usuario puede ver, modificar y distribuir el código con sus derechos originales a su discreción [24].

Los inicios de este término se aloja entre los años 50s y 60s durante el desarrollo de tecnologías tempranas al internet y protocolos de redes de telecomunicación, los investigadores encargados del proyecto se basaron en un ambiente de investigación colaborativo abierto ARPANET (*Advanced Research Projects Agency Network*), este proyecto posteriormente se convertiría en los cimientos del internet moderno y a la par,

de los valores de colaboración, revisión por pares y comunicación [25].
En particular, un sinnúmero de empresas emplean estas fuentes puesto que destacan una ventaja de desarrollo muy rápido que frecuentemente integran varias empresas colaborativas [26], en adición, el software de código abierto es preferido en contraste al software propietario, particularmente por consideraciones que agregan valor al mismo de acuerdo con los puntos destacados posteriormente [25].

- Revisión por pares, tomando en consideración que el código es libremente accesible y la comunidad es activa, es muy probable que el proyecto sea revisado y mejorado por otros programadores.
- Transparencia, el software de código abierto permite revisar y dar seguimiento a los movimientos y cambios de datos.
- Fiabilidad, la revisión por pares garantizan que el código se prueba de forma adecuada y frecuente.
- Bajo costo, el código es gratuito para el usuario final.
- Colaboración abierta, la existencia de comunidades activas permiten encontrar ayuda, recursos y perspectivas más amplias que en un grupo en particular o empresa.

1.4.5 Aprendizaje profundo en el procesamiento de lenguaje natural

La inteligencia artificial (IA), es una rama de la computación enfocada en la construcción de máquinas inteligentes capaces de desarrollar tareas que requieren inteligencia humana, esta área ha permitido abordar el desarrollo de carros autónomos hasta asistentes inteligentes. Durante la última década esta área ha crecido significativamente, de tal manera que hoy en día muchas compañías tecnológicas invierten en su crecimiento y desarrollo [27].
Natural Language Processing (NLP), procesamiento de lenguaje natural en español, hace referencia a la rama de la IA, cuyo objetivo es conceder a las computadoras la habilidad de entender texto escrito y palabras habladas de una forma similar al ser humano [28]; esta tarea ha sido resuelta mediante redes neuronales profundas (DNN), cuyo rol en la actualidad tiene una relevancia significativa en distintas tareas de NLP.

Las redes neuronales recurrentes, tratan los datos de forma secuencial y eficiente, incluso pueden procesar secuencias muy largas y recuerdan las salidas anteriores como entradas [29]. En particular, la tarea de traducción y transcripción de máquina ha sido resuelta mediante redes neuronales recurrentes recursivas (R2NN) [30], sin embargo uno de los problemas asociados, es la pérdida de información relevante al inicio de largas secuencias de datos. "*Attention is all you need*" destacó el reemplazo de capas recurrentes con mecanismos de atención, los resultados obtenidos destacaron una mejora en la tarea de traducción del idioma inglés a francés y alemán [31], obteniendo como resultado, una arquitectura que se denominó *Transformers*.

Posterior a la presentación de esta arquitectura, se desarrollaron librerías de código abierto con el objetivo de abrir los avances a la comunidad científica y a usuarios finales a fin de acceder a modelos pre-entrenados a gran escala en diversas tareas, en particular una librería con el mismo nombre de la arquitectura *Transformers* [32]. La tarea de traducción se puede abordar mediante diferentes modelos pre-entrenados entre ellos algunos basados en *The Tatoeba Translation Challenge*, que consta de un conjunto de datos para traducción de máquina en múltiples lenguajes [33], en adición de modelos basados en RNN y *transformers* [34].

1.4.6 Formatos de archivo de audio

El sonido, es una onda de presión causada cuando se producen vibraciones que se propagan en la dirección en que viaja la onda, este principio físico se puede evidenciar en los aplausos cuando las partículas de aire se juntan y posteriormente se separan tal que, las partículas más cercanas a la mano se alejan como un grupo de ondas sonoras chocando con las partículas vecinas, propagando así la onda [35]. La medición de esta variable se puede llevar a cabo empleando un micrófono, que esencialmente mide la presión de las ondas de sonido a una variedad de frecuencias, la mayoría de micrófonos están diseñados para medir el sonido que se propaga a través del aire [36].

Los *codecs* de audio son programas que comprimen datos para su respectiva transmisión y la descomprimen en el extremo receptor, la velocidad se mide en miles de bits procesados por segundo de tal manera que una baja tasa se asocia con la pérdida de datos. Entre otros factores para destacar la calidad del sonido, se resalta la frecuencia de muestreo y la profundidad de bits siendo estos la cantidad de muestras por segundo

y la cantidad de bits por muestra respectivamente [37].

Los formatos de archivos de audio se subcategorizan en tres diferentes tipos, formato con pérdida, puesto que se pierden datos en la transmisión puesto que no se descomprimen al tamaño del archivo original destacando variantes como MP3, AAC y Ogg Vorbis, los archivos de formato sin pérdida se caracterizan por descomprimir los datos a su tamaño original, manteniendo intacta la calidad del sonido, entre las variantes se destaca FLAC y ALAC, finalmente, los archivos de formato sin comprimir se destacan por mantener el mismo tamaño del formato de origen, destacando WAV, AIFF, DSD y PCM [37].

1.4.7 Buses de comunicación serial

Tomando en consideración que, el presente proyecto basa su entrada de datos en el sonido, se destaca la sección 1.4.7.2 que describe la comunicación entre el microcontrolador y el micrófono, además de la sección 1.4.7.1 que aborda la comunicación serial.

1.4.7.1 Circuito inter integrado (I2C)

I2C (*Inter-Integrated Circuit*, Circuito Interintegrado en español) es un protocolo de comunicación serial síncrono multicontrolador, que esta diseñado como un bus *maestro-esclavo* que permite conectar múltiples esclavos a un maestro y varios maestros pueden conectarse al mismo esclavo. En la Figura 1.2 se ilustra la conexión de un maestro a varios esclavos, se destaca que el protocolo I2C consta de dos señales que son *serial data* (SDA) y el *serial clock* (SCL). El maestro es el dispositivo que inicia la comunicación y esto lo realiza por medio de la linea de reloj (SCL), los esclavos son los dispositivos que responden al maestro y no inician la comunicación por si mismos, cada esclavo tiene su dirección única para que el maestro solicite o escriba datos [38].

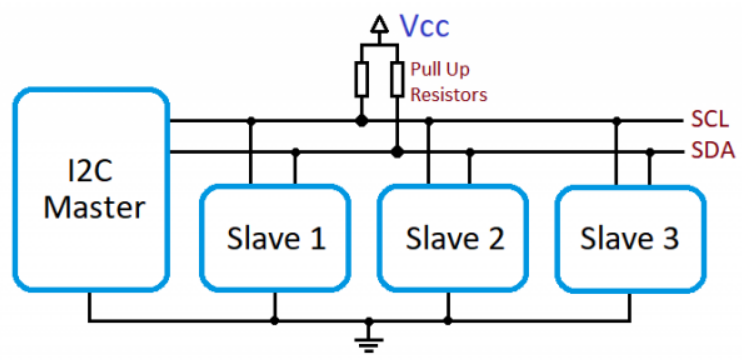


Figura 1.2 Configuración I2C maestro simple [39]

La comunicación de *maestro-esclavo* se la realiza por medio de la transferencia de datos y siempre es inicializada por un maestro y el esclavo *reacciona*. El protocolo I2C funciona por medio de la transferencia de datos, y estos datos se transfieren en mensajes, para facilitar la explicación de como funciona el I2C puede asumir que se tiene un solo mensaje, como se ilustra en la Figura 1.3.

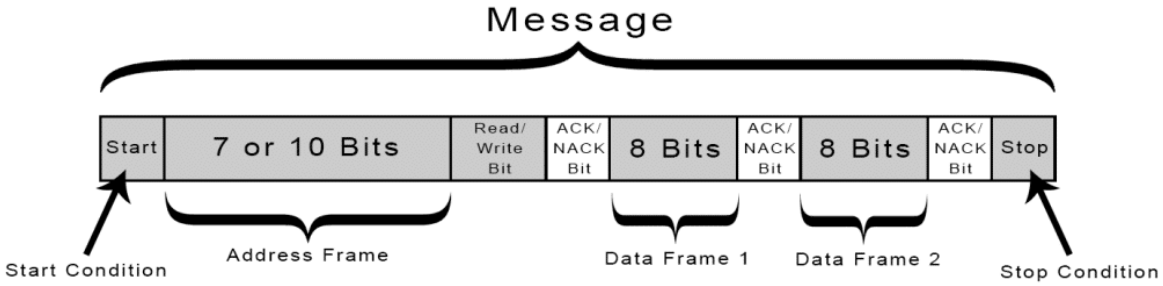


Figura 1.3 Formato de mensaje del protocolo de comunicación I2C [40]

Como se observa en la Figura 1.3 el mensaje se divide en varios tramos, la condición inicial (*Start condition*), cuando el maestro avisa o notifica a todos los esclavos, esto lo realiza cuando la línea SDA cambia de nivel de tensión alto a bajo, antes de que la línea SCL pase de alto a bajo. La dirección de tramo (*Address frame*) puede ser de 7 o 10 bits, esta se envía después de la condición inicial, en esta dirección el maestro especifica la dirección del esclavo con el cual se quiere comunicar. Los tramos de datos (*Data frame*) se transmiten después de la dirección de tramo, puede ser enviada del maestro al esclavo o del esclavo al maestro, y solo puede contener hasta 8 bits. Para la condición de parada (*Stop condition*), el maestro la genera cuando se han enviado/recibido todos los tramos de datos, esto se realiza cuando la línea SDA cambia de un nivel de tensión bajo a alto después de que la línea SCL pase de bajo a alto [39].

1.4.7.2 Bus de sonido Inter-IC integrado (I2S)

I2S (*Inter-IC Sound Bus*, Bus de sonido Inter-IC integrado en español) es un bus de comunicación serial estándar que se utiliza para conectar dispositivos de audio. La interfaz I2S se desarrolló con el objetivo de estandarizar la transmisión de datos digitales como ADCs, DACs, filtros digitales, entre otros tipos de circuitos integrados que utilizan sistemas de audio [41, 42].

Su diseño se centró en la comunicación digital de datos de audio entre circuitos integrados, el protocolo envía las señales de audio en formato PCM desde el controlador

a su objetivo. Su funcionamiento, se basa en tres líneas, entre ellas se destaca el *serial clock* (SCK), *word select* (WS) y *serial data* (SD) [42].

La Figura 1.4 muestra las configuraciones que se pueden desarrollar empleando la interfaz I2S. Además se observa que en sus 3 configuraciones, todas tienen 3 líneas en común SD WS y SCK. Los datos son conducidos por medio de la línea SD, el canal de audio le corresponde a la línea WS, y la línea SCK, es el reloj de serie. También se puede observar que en los 3 diagramas las señales SCK y WS pueden ser generadas por un transmisor (*transmitter*), receptor (*receiver*) o un controlador externo (*controller*) [41].

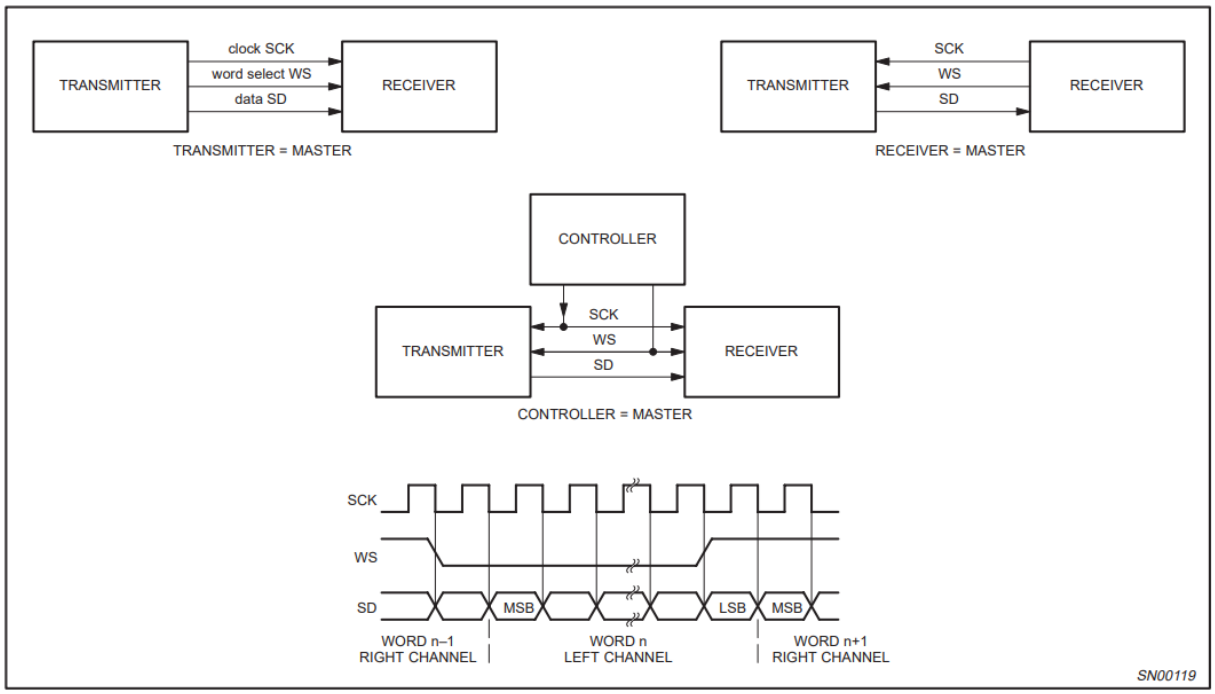


Figura 1.4 Configuraciones de interfaz I2S [41]

1.4.8 Proyección en lentes

El principio físico para abordar la proyección en lentes se basa en la óptica geométrica que resalta su estudio bajo la consideración del modelo de rayo de la luz, de tal manera que la complejidad de la naturaleza de la luz se reduce a geometría y trigonometría [43]. En particular, el sistema más simple para abordar como la luz forma imágenes es el espejo plano de acuerdo con lo mostrado en la Figura 1.5, la imagen refleja dos rayos que emergen del mismo punto, inciden el espejo y se reflejan en el ojo del observador [44].

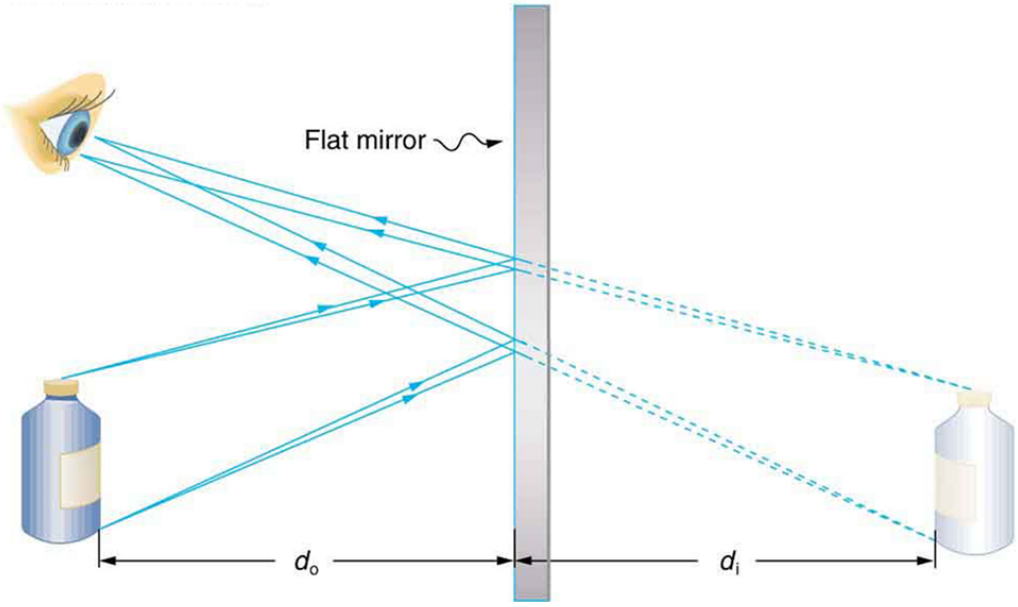
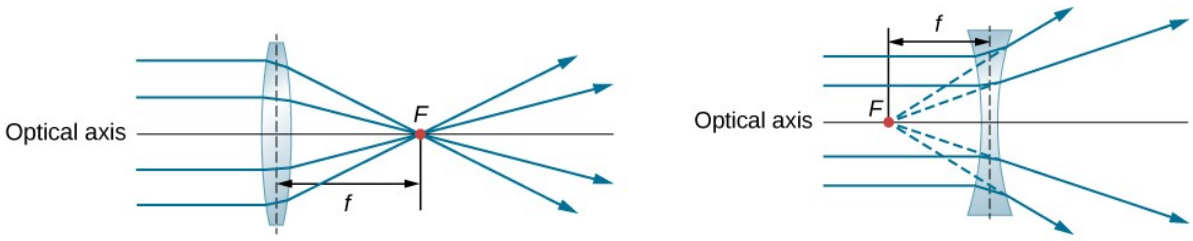


Figura 1.5 Principio físico de un espejo plano [44]

De forma similar se desarrolla el estudio para lentes convexos (convergentes) y cóncavos (divergentes) de acuerdo con lo mostrado en la Figura 1.6. La Figura 1.6a destaca la convergencia de los rayos de luz hacia un mismo punto de convergencia a diferencia de los lentes convergentes en la Figura 1.6b que describe la divergencia de los rayos de luz. Para lentes convergentes, el punto en el que cruzan los rayos es el punto focal *F* del lente, a diferencia de lentes divergentes, cuyo punto *F* es la fuente donde aparentan originarse los rayos, denominado el punto focal, cabe destacar la distancia *f* como la distancia focal, siendo medida desde el centro del lente [45].



(a) Lentes convergentes **(b) Lentes divergentes**

Figura 1.6 Lentes convergentes y divergentes [45]

Para obtener como resultado la proyección de una imagen mediante un sistema óptico, la Figura 1.7 destaca el principio de proyección para gafas de datos. En particular, el dispositivo de proyección (100) comprende una fuente de luz (104), encargado de emitir el haz de luz (106) y al menos un elemento holográfico (102) dispuesto o desechable en la lente de las gafas que proyecta la imagen sobre la retina (110) del usuario portador

de las gafas. Entre la fuente de luz (104), un elemento de colimación (114) y el objeto de reflexión (112) se encuentra un colimador a fin de homogeneizar las trayectorias de la fuente [46].

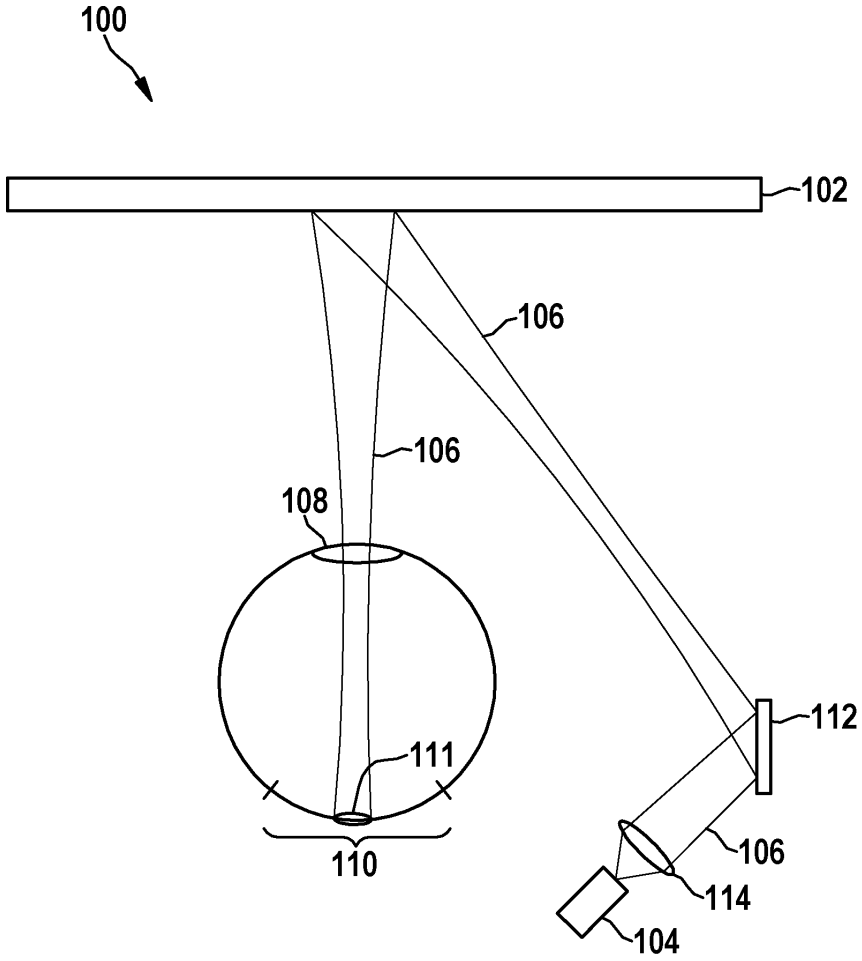


Figura 1.7 Principio de proyección para gafas de datos [46]

1.4.9 Presillas de sujeción (*snap fit*)

Una presilla es una forma fácil y eficiente de ensamblar partes de una pieza sin necesidad de tener herramientas o elementos de fijación adicionales, el criterio clave de las presillas es el desplazamiento y la flexibilidad de los elementos al momento de ensamblar y desmontar las uniones, por medio de presión. Dependiendo del diseño de la presilla esta puede ser separable o no separable de acuerdo con lo destacado posteriormente en la Figura 1.8.

Las presillas son normalmente empleadas en piezas plásticas, la elasticidad y la deformación del material plástico le permite realizar grandes desviaciones al ensamblar y

desmontar las piezas sin provocar daños, además, dependiendo de su aplicación también suelen ser empleadas en distintas combinaciones de materiales como metal-metal o plástico-metal [47].

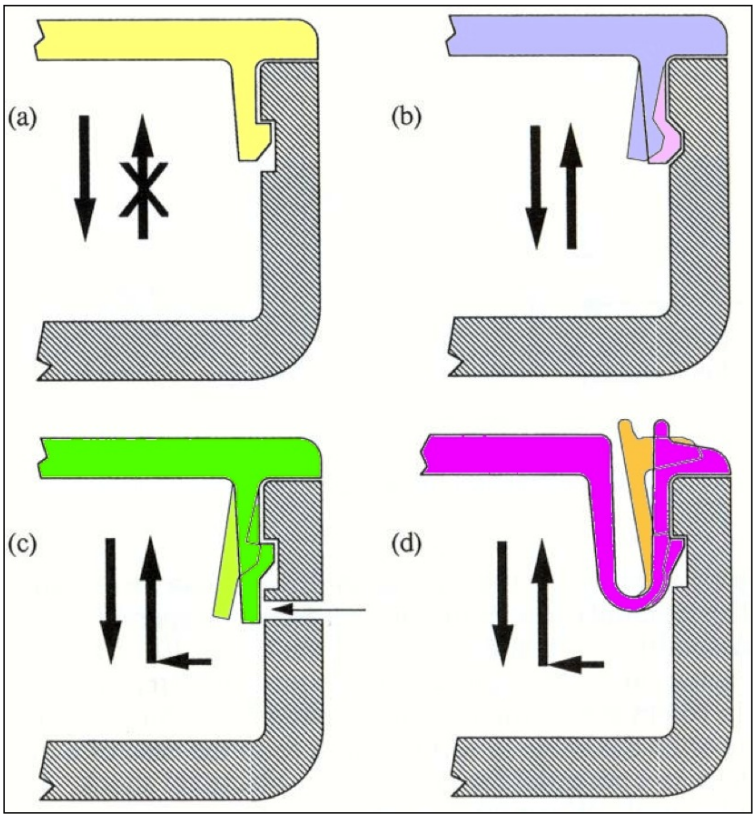


Figura 1.8 Presillas separables y no separables [48]

CAPÍTULO 2

2. METODOLOGÍA

Para el desarrollo de esta sección, se tomó en consideración un análisis cualitativo basado en los resultados de múltiples reuniones con el cliente para detallar los requerimientos, de tal manera que se determinaron características técnicas y funcionales del sistema a diseñar. Es muy importante tomar en consideración que se subdividió el sistema en diferentes componentes, entre ellos se destaca el componente mecánico, electrónico y de software, con el objetivo de considerar las diferentes metodologías de diseño de cada área.

2.1 Requerimientos de diseño

Posterior a la entrevista llevada a cabo con el cliente, se determinaron los requerimientos descritos en la Tabla 2.1.

Tabla 2.1 Requerimientos de diseño

Componente	Características
Peso y tamaño	• Peso máximo 500 gramos. • Tamaño similar a lentes convencionales, no voluminoso para un primer prototipo.
Funcionamiento	• Visualización de traducción a través de lentes. • Traducción y transcripción mediante tecnologías de código abierto.
Consumo de energía	• El dispositivo debe ser capaz de realizar el mínimo de tareas de procesamiento.
Modelos de predicción	• Los modelos de predicción empleados deben ser de código abierto.
Tiempo de respuesta	• Retardo de traducción de 3 segundos.
Costo	• Se requiere un diseño de bajo costo inferior a $100.

2.2 Criterios de selección

Para el análisis de las alternativas de solución, se tomaron en consideración cuatro criterios que deben cumplir a partir de la Tabla 2.1, estos se detallan posteriormente.

1. Peso y tamaño, es uno de los requerimientos con mayor importancia, puesto que el principal objetivo es no incomodar al usuario con un sistema muy voluminoso y pesado.

2. Tiempo de respuesta, es un requerimiento que considera el retraso que percibe el usuario desde que se emiten las palabras en idioma inglés hasta que se visualiza la traducción en español, este tiene menor relevancia que el criterio de peso y tamaño.

3. Costos involucrados, es un requerimiento que podría incrementar dependiendo de las dependencias que tenga el sistema.

4. Consumo de energía, tomando en consideración que el objetivo es diseñar un dispositivo *wearable*, debe contemplar el mínimo de tareas.

Tomando en consideración los criterios de selección detallados anteriormente, se elaboró la Tabla 2.2 con el objetivo de realizar la matriz de solución.

Tabla 2.2 Ponderaciones de criterios de alternativas

Criterio	Peso y dimensiones	Tiempo de respuesta	Costos involucrados	Consumo de energía	$\Sigma+1$	Ponderación
Peso y dimensiones		1	1	1	4	0.4
Tiempo de respuesta	0		1	1	3	0.3
Costos involucrados	0	0		1	2	0.2
Consumo de energía	0	0	0		1	0.1
Total	–	–	–	–	10	1.0

A partir de la Tabla 2.2, se destacan los criterios de selección a considerar en la selección de las alternativas priorizados en el siguiente orden, Peso y dimensiones > Tiempo de respuesta > Costos involucrados > Consumo de energía.

2.3 Alternativas de solución

Tomando en consideración la problemática, los requerimientos de diseño que se muestran en la Tabla 2.1, se destacaron posibles alternativas de soluciones que difieren entre sí en un mismo criterio, y este se relaciona al lugar físico donde se aloja el modelo encargado de la predicción de tareas de transcripción y traducción, puesto que este criterio influye de forma directa en los criterios de diseño como se describe posteriormente.

- **Alternativa A**:

  Alojar localmente el modelo en el dispositivo a fin de evitar dependencias con servicios externos.

- **Alternativa B**:

  Emplear una tecnología inalámbrica de corto alcance que permita la comunicación con un dispositivo móvil donde se aloje el modelo.

- **Alternativa C**:

  Considerar un proveedor de nube que aloje el modelo y mediante una API realizar la comunicación con el dispositivo.

2.4 Selección de solución

Para la selección de la mejor solución se aplicó el análisis del diseño concurrente, donde se consideró la Tabla 2.2 con los criterios de selección desarrollada anteriormente.

Tabla 2.3 Matriz de decisión de alternativas

Conclusiones	Peso y dimensiones	Consumo de energía	Costos involucrados	Tiempo de respuesta	$\Sigma + 1$	Prioridad
Alternativa A	0.067	0.017	0.083	0.075	0.242	3
Alternativa B	0.167	0.050	0.083	0.075	0.375	2
Alternativa C	0.167	0.033	0.033	0.150	0.381	**1**

La Tabla 2.3 detalla los resultados de las ponderaciones de acuerdo con los criterios de selección (ver Apéndice A: Detalle de ponderaciones por criterio). Como resultado, la **Alternativa C**, se caracteriza por tener la mayor prioridad en comparación con las otras alternativas, puesto que se ajusta a los requerimientos mencionados anteriormente.

2.5 Caso de uso

El diseño de la propuesta se enfoca en la tarea de traducción en un escenario que involucra únicamente a dos personas en un ambiente bajo la comunicación en el idioma inglés, teniendo el idioma español como el idioma objetivo. La entrada de datos se realiza en formato de audio, estos datos se deben procesar mediante un proceso de transcripción y posteriormente se debe traducir al idioma objetivo. La Figura 2.1 destaca el diagrama de caso de uso.

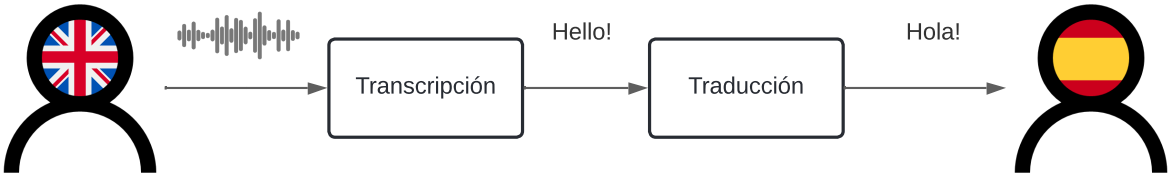


Figura 2.1 Diagrama de caso de uso

2.6 Diseño conceptual del sistema

En la Figura 2.2 se muestra el diseño conceptual del sistema de traducción. Este sistema requiere un micrófono capaz de captar señales de audio emitidas por el emisor, estas señales son enviadas al microcontrolador y posteriormente enviadas a la nube. En la nube se realiza el proceso de transcripción y traducción, una vez finalizado, el resultado se retorna al microcontrolador y mediante el sistema de proyección se retroalimenta al usuario con el resultado.

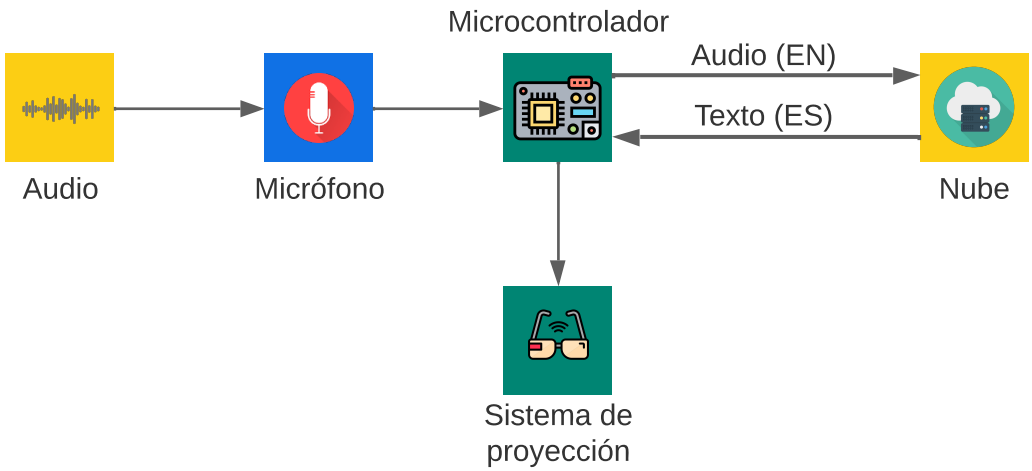


Figura 2.2 Diseño conceptual del sistema

2.7 Diseño electrónico

En esta sección se realizó la selección de los componentes necesarios para el diseño electrónico detallado en la sección 2.7.1, además se describe la integración de cada componente en la sección 2.7.2.

2.7.1 Componentes

En esta sección se explica brevemente las características mas importantes de los componentes necesarios para el circuito electrónico, y cómo estos componentes se ajustan a los requerimientos de diseño que se detallan en la Tabla 2.1.

2.7.1.1 Microcontrolador

Siguiendo los criterios de selección, los requerimientos de diseño que se detallan en la Tabla 2.1 y las características de la Alternativa C que consiste en alojar los modelos de traducción en un proveedor de nube, se selecciona el microcontrolador ESP32-WROOM-32 que incorpora un modulo Wifi, que le permite conectarse a internet y poder realizar peticiones HTTP, además incorpora 2 núcleos CPU, que se pueden configurar de 80 MHz a 240 MHz, que le permiten optimizar su tiempo de respuesta al realizar procesos. Particularmente, el microcontrolador es empleado para realizar tareas adquisición de datos de audio, además de realizar peticiones HTTP y enviar el texto plano a una pequeña pantalla. Este microcontrolador se caracteriza por ser muy ligero con aproximadamente 1.44 oz, sus dimensiones físicas que se detallan en la siguiente Tabla 2.4.

Tabla 2.4 Características del microcontrolador ESP32-WROOM-32 [49]

Característica	Descripción
Protocolo Wi-Fi	A-MPDU y A-MSDU
Rango de frecuencia Wi-Fi	2.4 GHz – 2.5 GHz
Dimensiones	18.0 $\times$ 25.5 $\times$ 3.1 mm
Protocolo Bluetooth	Bluetooth v4.2 BR/EDR y BLE
Alimentación	3.0 – 3.6 V
Corriente mínima	500 mA
Rango de voltaje de salida de ADC	0 – 3.3 V
Peso	1.44 oz

2.7.1.2 Micrófono

En la sección del caso de uso, se observó que el emisor es la fuente de sonido mediante el habla, estas palabras son ondas sonoras que el sistema debe ser capaz de transcribir y traducir al lenguaje español. En este caso, se necesita un pequeño micrófono que sea capaz de receptar estas palabras. Se seleccionó el micrófono MAX9814 de la serie FBA-ADA1713, puesto que es capaz de amplificar y controlar las ganancias de audio de forma automática. Además, este micrófono se ajusta al criterio de peso y tamaño, puesto que su peso aborda las 0.16 oz y sus dimensiones son 3.94 $\times$ 2.36 $\times$ 0.59 pulgadas. En la siguiente Tabla 2.5 muestran mas características del micrófono seleccionado.

Tabla 2.5 Características del micrófono seleccionado [50]

Característica	Descripción
Marca	Adafruit
Alimentación	2.7 – 5.5 V
Dimensiones	3.94 $\times$ 2.36 $\times$ 0.59 in
Frecuencia de respuesta	20 Hz – 20 KHz
Ganancia automática	40, 50, 60 dB
Peso	0.16 oz

2.7.1.3 Pantalla

El criterio de peso y dimensiones se tomó en consideración para la selección, en particular, se seleccionó la pantalla de 0.96 pulgadas OLED 4 Pin 128 $\times$ 64, siendo esta una de las más pequeñas que se encontraron en el mercado, las dimensiones se destacan en la Figura B.2. La Tabla 2.6 describe las características de la pantalla seleccionada.

Tabla 2.6 Características de la pantalla seleccionada [51]

Característica	Descripción
Alimentación	3.3 – 5 V
Pixeles	128 $\times$ 64
Tamaño de pantalla	0.96 in
Protocolo de comunicación	I2C
Puertos de entrada y salida requeridos	2
Peso	0.49 oz

2.7.2 Circuito electrónico

El diagrama del circuito electrónico, se detalla en la Figura 2.3, se tomaron en consideración, los componentes de la sección 2.7.1. La Figura, destaca el microcontrolador ESP32-WROOM-32, el micrófono y la pantalla que se conectaron de forma directa a la fuente de alimentación de 3.3 V y 5 V respectivamente. El ESP32-WROOM-32 es el encargado de realizar el envío de datos de audio al servidor mediante el protocolo de comunicación HTTPS, también se encarga de administrar lo datos retornados por parte del servidor para que sea renderice en la pantalla empleando el protocolo de comunicación I2C. En adición, el micrófono recepta las señales y las envía al microcontrolador empleando el protocolo I2S.

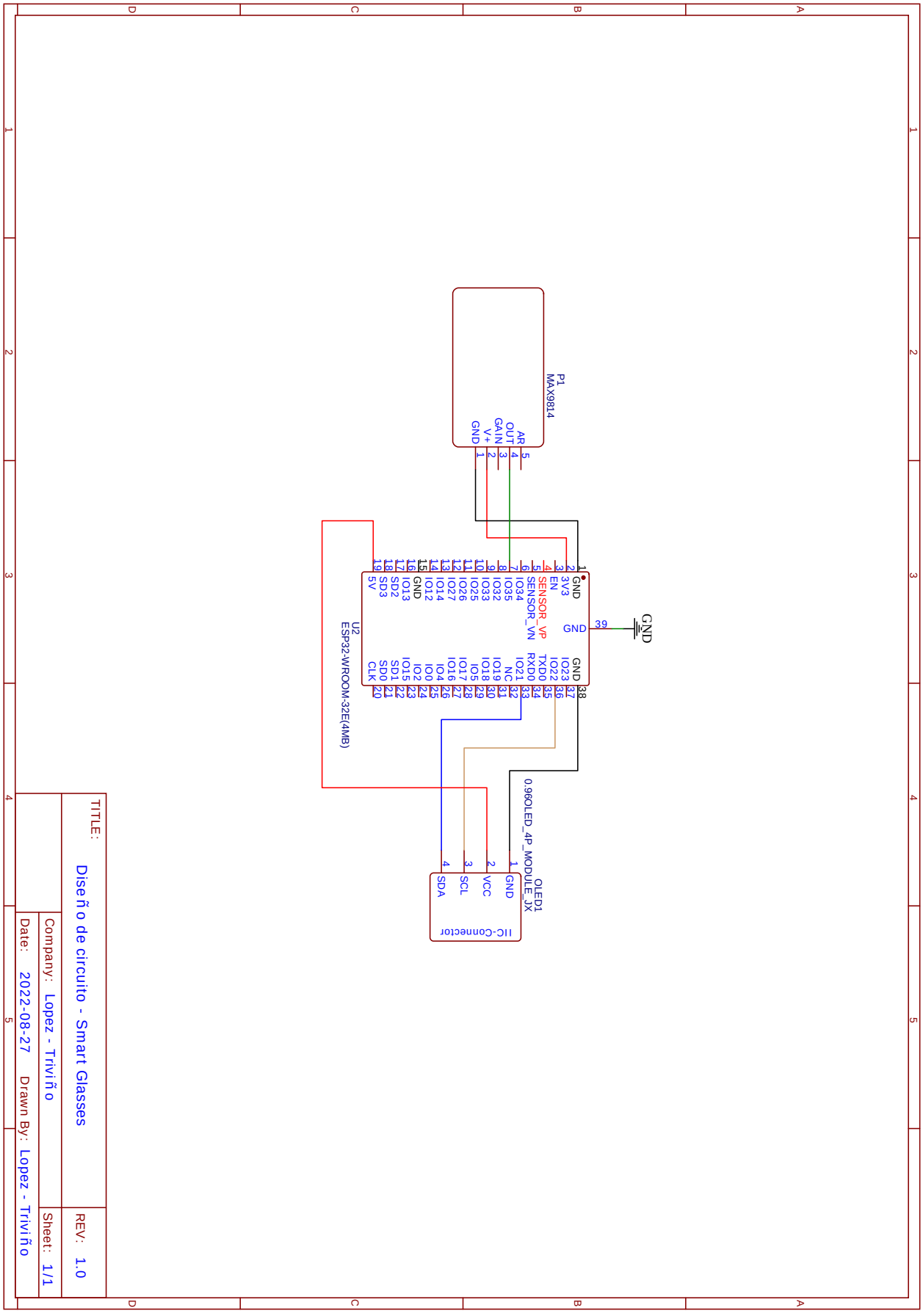


Figura 2.3 Diagrama del circuito electrónico

2.8 Diseño de software

Para llevar a cabo el proceso de diseño del sistema embebido y la arquitectura de microservicios, se tomó en consideración la Figura 2.2, por ello se determinó que el microcontrolador sería el encargado de la recopilación de datos de audio y retroalimentación al usuario con la traducción en formato de texto, es por ello que se adjudicó la tarea de procesamiento de datos (desde un formato de audio hasta texto traducido) a los servicios que alojados en los diferentes recursos ofrecidos por el proveedor de nube.

2.8.1 Software embebido

Para el desarrollo del software embebido se tomó en consideración el microcontrolador detallado en la sección 2.7.1.1.

2.8.1.1 Adquisición de datos de audio

El amplificador empleado se definió en la sección 2.7.1.2, siendo este el sensor encargado de la adquisición de datos para el procesamiento de audio. En primera instancia se destaca el sonido como una variable analógica, puesto que está compuesto por una onda continua [52], es por ello que la lectura de esta variable se realiza mediante una entrada analógica del ESP32-WROOM-32, que de acuerdo con la Tabla 2.4, se pueden medir en una escala entre 0 y 4096, correspondiente a 0 V y 3.3 V.

La frecuencia de muestreo relaciona la cantidad de muestras por segundo de la variable a medir, en particular para generar archivos de audio de baja calidad se requiere una frecuencia de 8 kHz, mientras que una frecuencia de muestreo 16 kHz permite obtener un archivo de buena calidad [53]. La lectura de los datos analógicos podría realizarse directamente del ADC, sin embargo el principal inconveniente de este enfoque radica en la frecuencia de muestreo requerida, puesto que realizar una lectura directa puede tomar aproximadamente 100 μs [54], afectando la calidad de los datos.

Con el objetivo de cumplir con un rango válido respecto a la frecuencia de muestreo que permita obtener un archivo de calidad media a 12 kHz, se tomó en consideración un enfoque distinto mediante el protocolo de comunicación I2S [53]. La forma de adquirir los datos bajo este protocolo de comunicación es alojándolos de forma temporal en una región específica de memoria, posterior al tiempo de muestreo, este paquete de datos se

envía al servidor mediante una API. En particular, el método empleado para representar las muestras de acuerdo con la frecuencia de muestreo es PCM *little endian*, siendo este un criterio que destaca la forma en cómo se organizan los bytes [55] y con una resolución de 16 bits.

El desarrollo del código se implementó mediante el kernel del sistema operativo en tiempo real *FreeRTOS*, el mismo que permite crear una tarea y añadirlo a la lista de tareas que se encuentran listas para ejecutarse [56], en este caso en particular se empleó la función `xTaskCreatePinnedToCore` a fin de ejecutar la lectura de los datos mediante el protocolo I2S (ver Apéndice C: Código de sistema embebido).

2.8.1.2 Retroalimentación de la traducción

Posterior a la recopilación de datos de audio, procesamiento de datos hasta su transcripción, traducción; el microcontrolador recibe los datos en formato de texto, de tal manera que este se renderizará en la pantalla hasta la retroalimentación del usuario mediante el sistema óptico. La pantalla empleada se describió en la sección 2.7.1.3, para su comunicación se empleó el protocolo de comunicación I2C, puesto que este es el requerido de acuerdo con la Tabla 2.6.

2.8.2 Arquitectura de Microservicios

2.8.2.1 Proveedor de nube

En la solución propuesta se necesita un proveedor de nube, por que un microcontrolador como el ESP32-WROOM-32, no seria capaz de realizar el proceso de transcripción y traducción, recordemos que estos procesos requieren del entrenamiento de modelos y para poder entrenar estos modelos se requiere de un considerable poder computacional. Para un proveedor de nube no seria problema realizar estos procesos ya que consta con servidores capaces de realizar esta tarea. En este caso se selecciona AWS, por los motivos descritos posteriormente [57].

- Infraestructura Global, AWS está disponible en varias regiones geográficas.
- Redundancia de Datos, AWS crea diferentes copias de sus datos de forma predeterminada en algunos de sus recursos.
- Baja Latencia en la entrega de contenido, AWS tiene una de las mayores redes de

entrega de contenido, esto permite a los usuarios acceder a su aplicación de forma rápida y eficiente.

- Integración con una gran cantidad de servicios de terceros, AWS siempre trabaja con servicios y proveedores de terceros, con el objetivo de ofrecer variedad de opciones y soluciones.
- Modelo de precios de pago por uso, es decir, AWS cobra por lo que se usa, ya sea segundo, minutos u horas.
- Alta disponibilidad AWS ofrece el 99.99% de disponibilidad de sus servicios.

2.8.2.2 Traducción de máquina

HuggingFace es una comunidad y una plataforma enfocada a la ciencia de datos que ha desarrollado diferentes modelos de código abierto relacionados a distintas tareas como visión computacional, procesamiento de lenguaje natural, audio, entre otras. Una de las ventajas más importantes de emplear los modelos de ML que ofrece HuggingFace es que permiten ser construidos, entrenados y desplegados de acuerdo con la necesidad del usuario [58].

Respecto a la tarea de reconocimiento automático de voz, se destaca el modelo `s2t-small-mustc-en-es-st`, se describe como un modelo capaz de realizar la traducción de voz de extremo a extremo de voz a texto [59]. *Fairseq* es un conjunto de herramientas de modelado de secuencias para entrenar modelos personalizados enfocados a distintas tareas como traducción, resumen y otras relacionadas a la generación de texto, contiene diferentes implementaciones a modelos de redes LSMT, CNN y RNN [60]. *Fairseq S2T* es el modelo desarrollado por Meta que se introdujo como una extensión de S2T para las tareas de reconocimiento de voz y traducción de voz a texto basado en RNN y *transformers* [34]. En particular, se tomó en consideración este modelo puesto que se limita a la traducción de voz en inglés a texto en español y además, se puede integrar con *SageMaker* (Servicio de AWS).

2.8.2.3 Arquitectura

Para desarrollar la arquitectura, se emplearon algunos servicios de AWS, entre ellos *API Gateway*, *Lambda*, *S3* y *SageMaker* como se detalla en la Figura 2.4, mientras la Figura 2.5 destaca un diagrama de secuencia de la arquitectura diseñada.

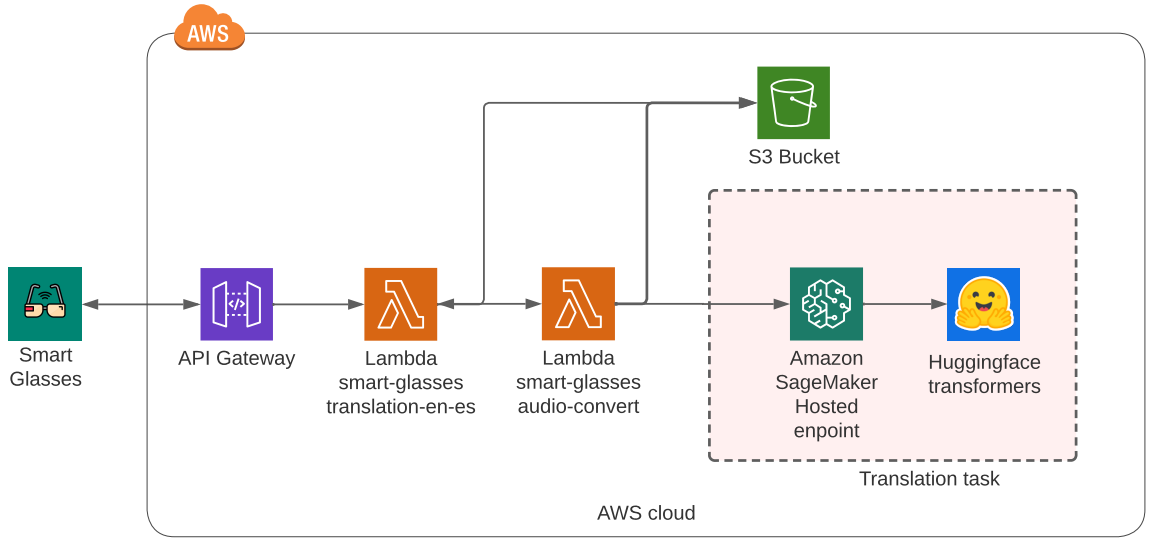


Figura 2.4 Diagrama de arquitectura en nube

API Gateway se describe como un servicio administrado que permite la creación, publicación, mantenimiento y protección de APIs, entre las opciones se destacan REST y WebSocket. En el presente proyecto por limitaciones del microcontrolador se consideró trabajar con una API REST sobre WebSocket debido a las limitaciones a nivel de hardware, puesto que, empleando el ESP32-WROOM-32 no se pudo establecer una conexión Websocket mediante el protocolo HTTPS. En particular, los precios destacan un nivel libre que abarca un millón de consultas, posteriormente se destaca que dentro de las 333 millones de peticiones el precio por millón es de aproximadamente $ 3.50 [61]. *Lambda* es un servicio de cómputo *serverless* basado en eventos que permite ejecutar código sin la aprovisión o administración de servidores basando su pago por uso, que particularmente, para una arquitectura `x86`, el precio dentro de los primeros 6 billones GB-segundo / mes, es de $ 0.0000166667 por cada GB-segundo y $ 0.20 por millón de peticiones, en adición, es importante tomar en consideración que dentro del nivel libre se encuentran un millón de peticiones por mes [62].

S3 es un servicio de almacenamiento de objetos, que destaca ventajas como escalabilidad, disponibilidad de datos, seguridad y rendimiento, respecto a los precios dentro del nivel libre, se disponen 5 GB de almacenamiento en S3 standard por 12 meses. Los precios por almacenamiento dentro de los primeros 50 TB / mes es de aproximadamente $ 0.023 / GB [63].

SageMaker, ofrece una rápida y sencilla manera de iniciar con ML. Entre las soluciones se encuentran despliegue, entrenamiento de diferentes modelos de NLP, detección de objetos y clasificación de imágenes. En particular, el presente proyecto emplea la inferencia sin servidor (*Serverless Inference*) del modelo detallado en 2.8.2.2, se tomó en consideración este servicio puesto que el precio se relaciona con la capacidad de cómputo empleada, tiempo en milisegundos y la cantidad de datos procesados en el proceso de inferencia. Los precios para una configuración de 3 GB de memoria asignados es de $ 0.0000600 por segundo. Cabe mencionar que el nivel libre abarca 150,000 segundos de duración de inferencia [64].

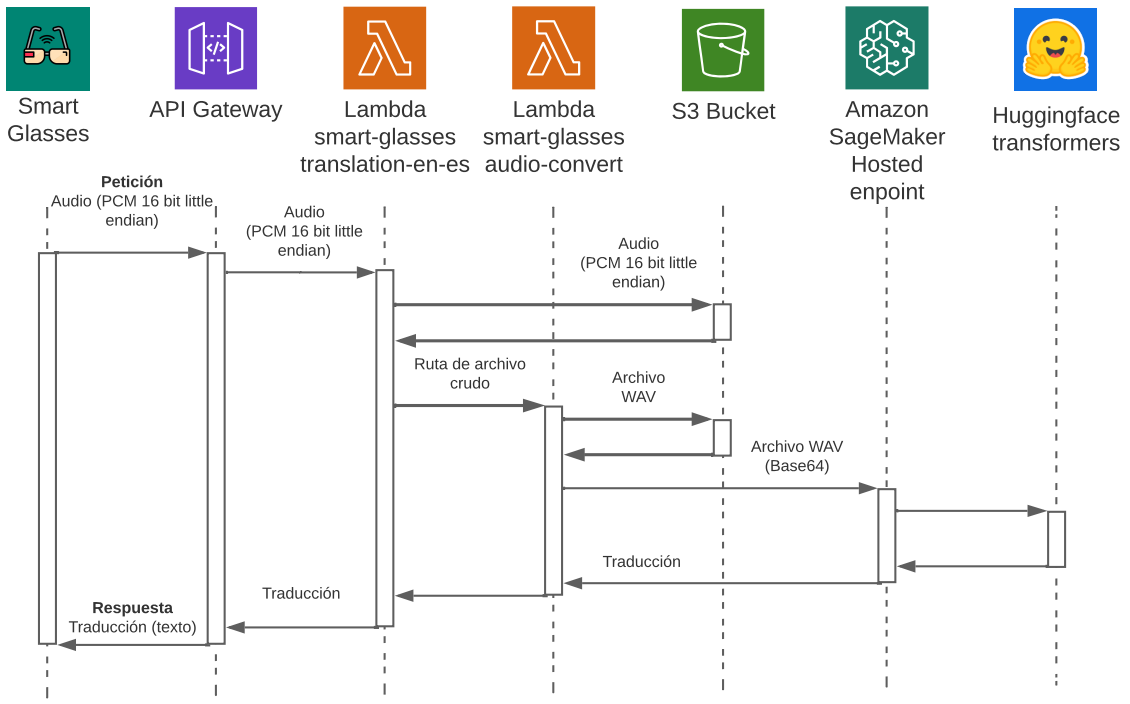


Figura 2.5 Diagrama de secuencia de datos

Tomando en consideración la Figura 2.4, se definieron dos funciones Lambda para realizar dos tareas en el procesamiento de datos de audio `smart-glasses-translation-en-es` y `smart-glasses-audio-convert` (ver Apéndice D: Código de lambda translation-en-es y Apéndice E: Código de lambda audio-convert).

La Lambda `smart-glasses-translation-en-es` se definió con ambiente de ejecución `Node.js 14.x`, esta función se expuso por API Gateway a fin de que sea invocada mediante un evento HTTP, esta función tiene como objetivo recibir los datos de audio

en formato PCM 16-bit *little endian* a una tasa de muestreo de 12 kHz y un tiempo de muestreo de 2 segundos, los datos crudos que recibe se envían a S3 y posteriormente se invoca la lambda `smart-glasses-audio-convert`, que retorna los resultados de la traducción y se retornan al cliente que realizó la petición.

La Lambda `smart-glasses-audio-convert` tiene como ambiente de ejecución `Python 3.9`, esta función únicamente se ejecuta mediante invocación. El objetivo de esta lambda es convertir los datos de audio en formato PCM 16-bit a WAV empleando la librería `wave`, como resultado, se obtiene un archivo `.wav` que se envía a S3, este posteriormente se convierte a formato base 64, siendo este el cuerpo de la petición que se envía al servicio de SageMaker que finalmente, retorna la traducción.

2.9 Diseño Mecánico

El diseño mecánico, se dividió en la sección 2.9.1, que detalla el sistema óptico definido, con el objetivo de retroalimentar al usuario con la traducción y la sección 2.9.3 que describe el brazo del sistema que incorpora los componentes electrónicos y el sistema óptico.

2.9.1 Sistema de proyección

Respecto a la proyección de imagen en los lentes, una solución abordada se basa en un sistema óptico obteniendo ventaja de los principios de reflexión en una superficie plana y curva [43]. Se tomó en consideración el diseño del sistema óptico de la Figura 1.7, sin embargo, se realizaron modificaciones como se observa en la Figura 2.6, que detalla el diagrama del sistema óptico empleado en el presente proyecto.

La Figura 2.6 detalla la pantalla como la fuente de luz siendo esta la traducción en formato de texto. El haz de luz se invierte por el objeto reflector (sección de espejo), posteriormente, atraviesa los lentes de Fresnel, cuyo objetivo es amplificar la imagen que finalmente, se visualiza en el dispositivo de proyección, que particularmente es una sección de acrílico transparente.

2.9.2 Presilla en voladizo

Existen diferentes tipos de presillas, generales, voladizas, de torsión y anulares. Entre los diferentes tipos de presillas, generales, voladizas, de torsión y anulares; se seleccionó el tipo voladizo puesto que el dispositivo en proceso de diseño no se somete a fuerzas

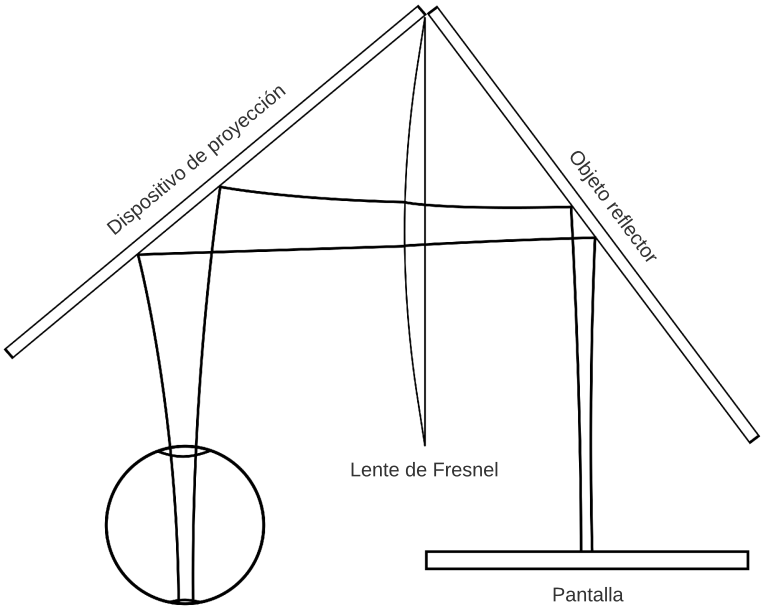


Figura 2.6 Diagrama de sistema óptico

externas y su único objetivo es unir piezas mediante flexión, la Figura F.1 destaca las diferentes formas de sección transversal que puede tener una presilla en voladizo (ver Apéndice F: Ecuaciones del diseño de Presilla (*snap fit*)).

La Figura 2.7 ilustra el modelo de la presilla a diseñar, en este caso es una presilla en voladizo de sección transversal rectangular, donde ε es la deformación permisible, h es el alto de la sección transversal, y es la deflexión permisible, l es la longitud del brazo, b es el ancho de la sección transversal, finalmente P es la fuerza de deflexión permisible.

Para diseñar una presilla en voladizo de sección transversal rectangular se consideró la ecuación 2.1 obtenida de la Figura F.1, esta ecuación permite conocer la deflexión permisible de la presilla.

$$y = 1.09 \times \frac{\varepsilon l^2}{h} \tag{2.1}$$

Una vez establecido el tipo de presilla, es importante también conocer el material a utilizar, en este caso en particular el filamento PLA, se seleccionó este material puesto que uno de los materiales más empleados en el desarrollo de prototipos empleando impresión 3D.

Un parámetro importante al diseñar una presilla es la deformación del material, cuyo valor

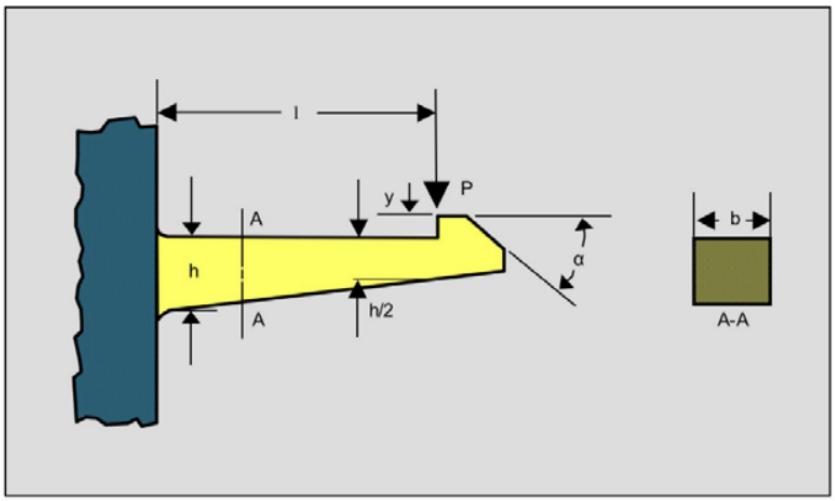


Figura 2.7 Presilla en voladizo de sección transversal rectangular [65]

es 3.3 % para el filamento PLA [66]. Adicionalmente, se establecieron ciertos parámetros de diseño, como el largo del brazo de la presilla a 7.5 mm, no se seleccionó un valor mayor debido a que la presilla tiene como objetivo unir la tapa y la base del brazo de lente mostrado en la sección 2.9.3.

El brazo del lente se diseñó de tal manera que su objetivo sea contener los componentes electrónicos incluyendo el sistema de proyección; considerar un mayor largo en la presilla, podría provocar choques con los componentes electrónicos, es importante destacar que una menor longitud la presilla tendría menor deflexión, aspecto que pondría en riesgo su capacidad de ser separable debido a que la fuerza de deflexión soportada sería la mínima. Tomando en consideración las dimensiones de la tapa del brazo detallada en la Figura G.1 se seleccionó una deflexión de 1 mm [48].

Considerando la ecuación 2.1 y los parámetros definidos, la altura de la sección transversal de la presilla se obtiene mediante la operación posterior considerando una deformación del material de 0.033 con un factor de seguridad del 65 % (0.02) [65].

$$h = 1.09 \times \frac{0.02 \times (7.5)^2}{1}$$

$$h = 1.22\ mm$$

La Figura 2.8 muestra las dimensiones de la presilla diseñada empleando el software de diseño Inventor.

Otro parámetro importante al diseñar una presilla es la fuerza de deflexión permisible

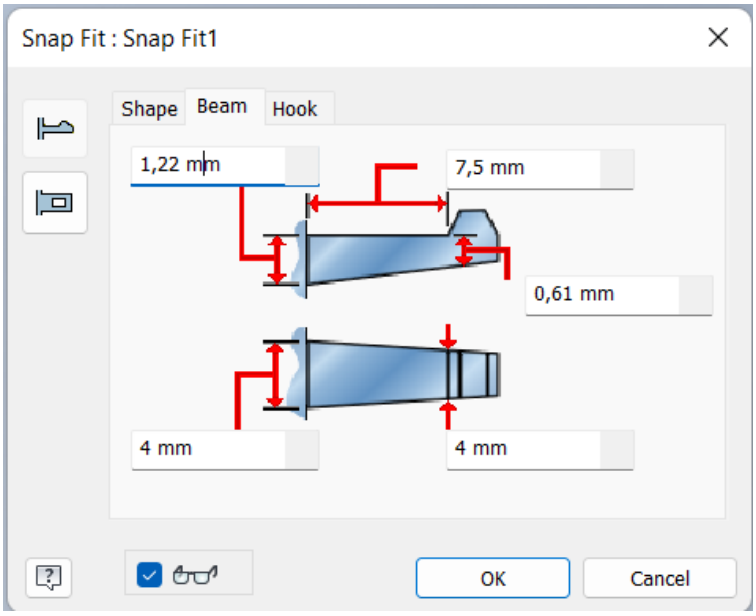


Figura 2.8 Dimensiones de la presilla de sección transversal rectangular diseñada

que se calcula con la ecuación 2.2 obtenida de la Figura F.1, donde E_s es el modulo de elasticidad del material de la presilla, en este caso 2346.5 MPa [66].

$$P = \frac{b(h)^2 \times E_s \varepsilon}{6} \tag{2.2}$$

$$P = \frac{4(1.22)^2 \times 2346.5 \times 0.02}{6}$$

$$P = 46.57\ N$$

La Tabla 2.7 detalla los parámetros definidos y los resultados de los calculados que se realizaron al diseñar la presilla.

Tabla 2.7 Propiedades de la presilla diseñada

Símbolo	Descripción	Valor
ε	Deformación permisible	2 %
h	Alto de la sección transversal	1.22 mm
y	Deflexión permisible	1.00 mm
l	Longitud del brazo	7.50 mm
b	Ancho de la sección transversal	4.00 mm
P	Fuerza de deflexión permisible	46.57 N

2.9.3 Brazo de lentes

Para el diseño del brazo de los lentes, se tomaron en consideración las dimensiones de los componentes electrónicos empleados (ver Apéndice B: Componentes electrónicos). El brazo, se subdividió en dos pequeñas piezas, de tal manera que se compone por una tapa y una base como se destaca en la Figura 2.9. Los planos y las dimensiones de estas piezas, se encuentran en la sección de apéndices (ver Apéndice G: Planos mecánicos).

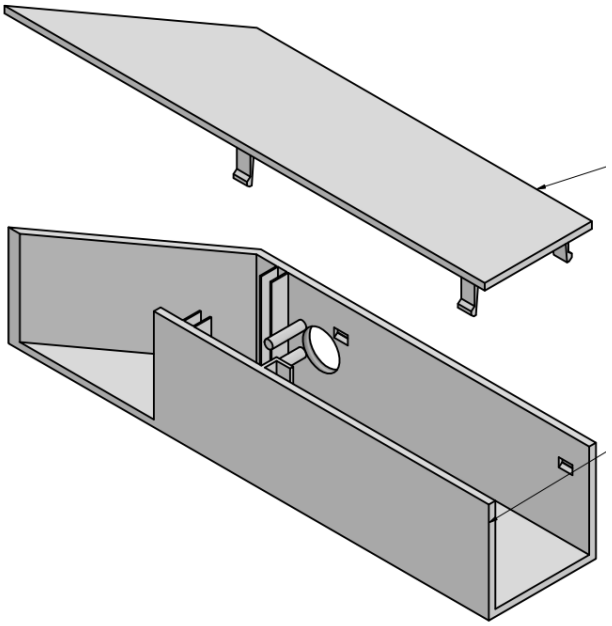


Figura 2.9 Tapa y base del brazo de lente

La base se diseñó con el objetivo de incorporar los diferentes componentes electrónicos detallados en la sección 2.7.1, en conjunto con el sistema de proyección de la sección 2.9.1. La tapa se diseñó para cubrir la base con 4 presillas en voladizo como se detalló en la sección 2.9.2, se optó en utilizar este mecanismo de unión, puesto que es muy utilizado en materiales plásticos, además muchos de los productos tecnológicos que utilizamos hoy en día, tienen uniones tipo presilla, en sus compartimentos como las baterías, teléfonos e inclusive computadoras portátiles [67].

CAPÍTULO 3

3. RESULTADOS Y ANÁLISIS

El presente capítulo detalla el resultado de la integración de los distintos componentes del sistema en base a la investigación y la metodología empleada. Se empleó un software para evaluar el rendimiento de la API expuesta en la nube de AWS, en adición, se desarrollaron planos empleando el programa informático Inventor con el objetivo de presentar el diseño del brazo que contiene los diferentes componentes y sistema de proyección. Finalmente, se detalló el análisis de costos tomando en consideración los diferentes componentes electrónicos y servicios de software.

Con el objetivo de simplificar el nombre de las funciones Lambda desarrolladas, este capítulo reduce el nombre de las funciones `smart-glasses-translation-en-es` a **`translation-en-es`** y `smart-glasses-audio-convert` por **`audio-convert`**.

3.1 Desempeño de API de traducción

Las funciones Lambda desplegadas en la nube de AWS, detallan las características mostradas en la Tabla 3.1, ambas funciones comparten la misma arquitectura x86_64, una memoria asignada de 128 MB y un almacenamiento efímero de 512 MB.

Tabla 3.1 Configuraciones de funciones Lambda desplegadas

Función Lambda	Tamaño de paquete [bytes]	Ambiente de ejecución	Evento de ejecución
`translation-en-es`	1500.0	Node.js 14.x	API Gateway
`audio-convert`	778.0	Python 3.9	Invocación

Con el objetivo de evaluar el tiempo de respuesta del servicio desplegado en AWS, se tomó en consideración el monitoreo de API ofrecido por Runscope, siendo esta una compañía SaaS que ofrece software para testear el desempeño de una API [68]. En particular, la Figura 3.1 destaca los resultados del test empleando un cuerpo de petición constante que consiste en las palabras "*Hello, dude*" durante 2 segundos en formato PCM, el test se llevó a cabo cada cinco minutos durante 12 horas, reflejando una tasa de éxito

es del 100%, valor que refleja la disponibilidad del servicio, mientras que el tiempo medio de respuesta aborda los 1931.15 ms.

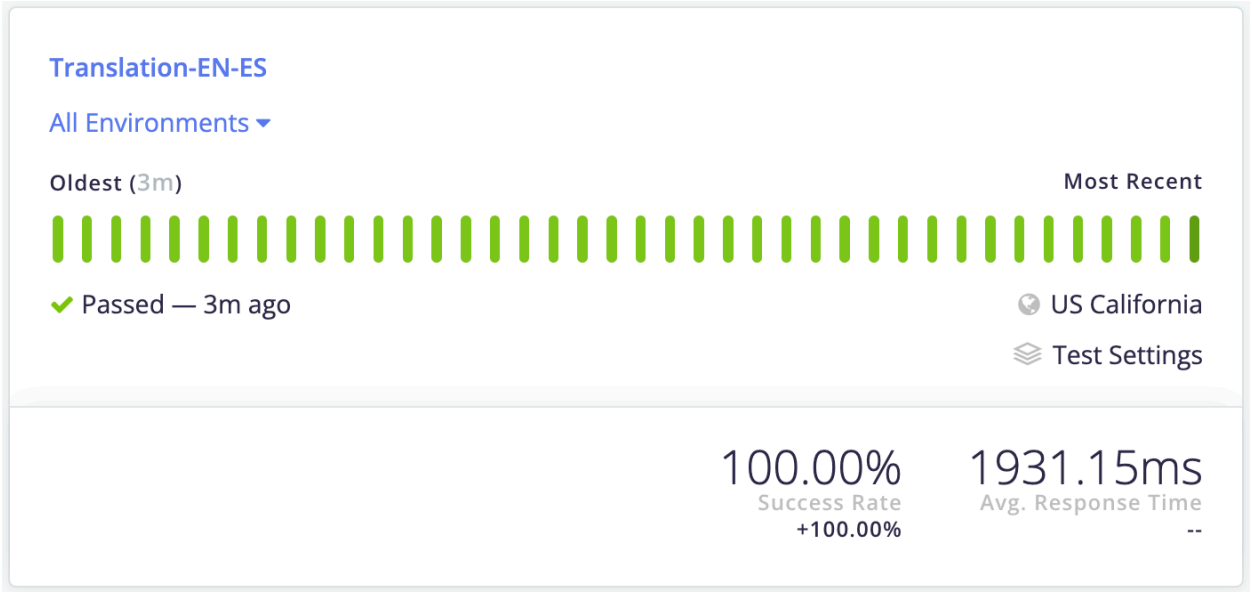


Figura 3.1 Resultados del monitoreo de la API vía Runscope

3.2 Archivos de audio generados

Para abordar el análisis de los resultados de audio, se evaluó una única ejecución que equivale a una conversación de 2 s. El diálogo en este caso consistió en las palabras "*Hello dude*". De acuerdo con la arquitectura diseñada y descrita en la Figura 2.4, la entrada de audio en formato PCM da como resultado dos archivos, uno que describe los datos crudos en el mismo formato y otro archivo en formato WAV. Ambos archivos tienen un tamaño de 93.8 KB y comparten el mismo contenedor de S3, sin embargo, se guardan en diferentes rutas `raw/2022/8/29/1661741328329.raw` y `wav/2022/8/29/1661741328329.wav` respectivamente. El formato de la ruta consiste en el tipo de formato, seguido del año, mes, día y finalmente la fecha en formato de tiempo Unix, siendo esta prueba ejecutada el día 29 de agosto del 2022. El reporte de la ejecución de ambas funciones Lambda se detalla en la Tabla 3.2.

En este caso particular, se esperaba como resultado "*Hola, amigo.*", sin embargo, el resultado obtenido fue "*Hola, Deep.*". La forma de onda del archivo generado se detalla en la Figura 3.2, esta destaca el inicio de las palabras a los 0.90 s y 1.30 s aproximadamente, la amplitud se normalizó en una escala de -1 a 1. Esta figura destaca el ruido con una amplitud similar al que describen las señales con el audio a traducir, este argumento

Tabla 3.2 Reporte de ejecución de funciones Lambda

Función Lambda	Tiempo de ejecución [ms]	Tiempo facturado [ms]	Memoria empleada [MB]
`translation-en-es`	1493.53	1494	91
`audio-convert`	725.36	726	79

podría explicar la causa de la falta de precisión en la traducción.

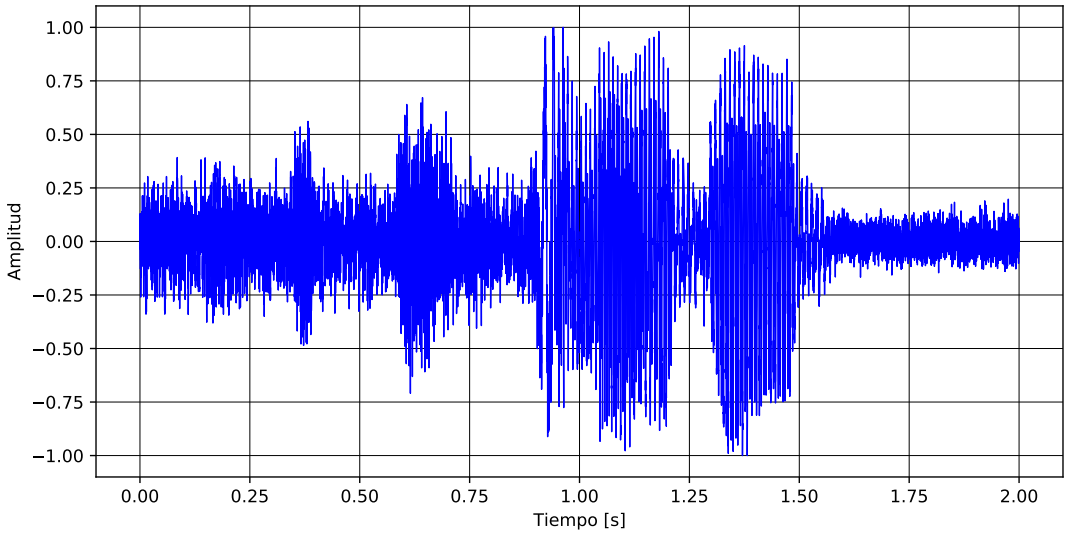


Figura 3.2 Amplitud de forma de onda de señales de audio

3.3 Integración de hardware

La Figura 3.3 destaca la vista superior de la base del brazo de lente diseñado en la sección 2.9.3, además, se presenta una enumeración de cada una de las secciones donde se ubicaron los distintos componentes electrónicos y el sistema de proyección. La Tabla 3.3 detalla la distribución y ubicación de los diferentes componentes de la Figura 3.3.

Se consideró el uso de un par de gafas genéricas con el objetivo de acelerar el diseño del prototipo, de tal manera que, como resultado de la integración del sistema óptico, componentes electrónicos y las gafas genéricas, se obtuvo un peso de 103 g, la Figura 3.4 detalla la estructura de los lentes.

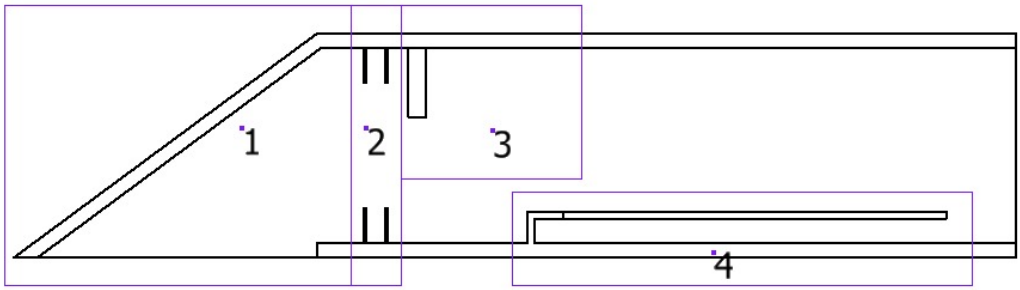


Figura 3.3 Vista superior de la base del brazo de lente

Tabla 3.3 Distribución de los componentes en la base del brazo

Numero	Descripción	Sección
1	Sistema de proyección	2.9.1
2	Pantalla	2.7.1.3
3	Micrófono	2.7.1.2
4	Microcontrolador ESP32-WROOM-32	2.7.1.1

Figura 3.4 Integración física de *smart glasses*

La Figura 3.5 destaca la retroalimentación de la traducción desde la perspectiva del usuario.

Figura 3.5 Retroalimentación de traducción

3.4 Análisis de costos

Para llevar a cabo el análisis de costos, se tomó en consideración los costos de los elementos del hardware del dispositivo, de tal manera que se destaca la Tabla 3.4.

Tabla 3.4 Análisis de costos – Hardware

Recurso	Precio ($)
Pantalla OLED 0.96 [in]	8.00
Micrófono MAX9814	9.95
Microcontrolador ESP32-WROOM-32	17.99
Lente de Fresnel	1.49
Sección reflectiva	0.13
Impresión de base sin armazón	7.00
Total	**44.56**

En adición, los costos relacionados a los diferentes recursos empleados en AWS, se destaca la Tabla 3.5, para obtener los resultados se tomó en consideración la calculadora de precios de AWS [69].

Tabla 3.5 Análisis de costos – Software (AWS)

Recurso	Costo	Subtotal ($)/mes
API Gateway	$ 3.5 / millón de peticiones	0.22
Lambda	$ 0.0000166667 / GB-segundo $ 0.20 / millón de peticiones	3.51
S3	$ 0.023 / GB	1.15
Sagemaker (Serverless inference)	$ 0.0000600 / segundo	0.90
Total		**5.78**

A partir de las tablas anteriores se obtienen costos fijos de $ 44.56, puesto que estos son los costos requeridos para la construcción del hardware, mientras que los costos variables abordan los $ 5.78 mensuales por usuario tomando en consideración un uso de aproximadamente cuatro horas diarias.

CAPÍTULO 4

4. CONCLUSIONES Y RECOMENDACIONES

El presenta capítulo destaca las conclusiones y recomendaciones en base a los objetivos y resultados del presente proyecto, que corresponden al diseño del dispositivo físico y la arquitectura *backend*, ambos aspectos constituyen el prototipo de gafas inteligentes para llevar a cabo la tarea de traducción del lenguaje inglés a español.

4.1 Conclusiones

- Las destrezas y el conocimiento adquirido en la carrera de Ingeniería Mecatrónica, en conjunto con el proceso de investigación, permitieron desarrollar un prototipo de gafas inteligentes en base a tres propuestas de soluciones que fueron evaluadas empleando el análisis de diseño concurrente. El análisis permitió abordar una solución que provee los servicios en la nube de AWS, de tal manera que el microcontrolador establece una conexión mediante el protocolo HTTPS para realizar el proceso de traducción tomando como entrada de datos el audio en ingles y obteniendo el resultado de texto en español como salida, tal que se retroalimenta al usuario mediante un sistema óptico.

- La API desplegada en la nube de AWS permitió realizar la comunicación entre el microcontrolador con el modelo de traducción.Se utilizaron diferentes servicios de AWS como funciones Lambda, contenedores S3, SageMaker, con el objetivo de limitar las tareas del microcontrolador y enfocarlas únicamente al envío de datos y gestión de los resultados de la traducción hasta su retroalimentación al usuario.

- Se empleó un modelo pre-entrenado de código abierto para realizar el proceso de traducción desarrollada por HuggingFace, este modelo se desplegó en el servicio de SageMaker de AWS.

- La selección de AWS, como proveedor de nube permitió tomar ventaja de la facturación basada en el pago por uso, este criterio permitió describir los costos por el uso de los servicios a $ 5.78 mensuales bajo un consumo diario de 4 horas diarias.

- Respecto al requerimiento de peso y tamaño, se obtuvo un peso de aproximadamente 103 g, además, el tiempo de respuesta desde el inicio de la petición aborda un tiempo de aproximadamente 1.9 s, cumpliendo así, con los requerimientos de diseño.

- Se diseñaron presillas en voladizo con sección transversal rectangular con el objetivo de unir las piezas del brazo de lente y que puedan separarse en caso de requerir mantenimiento. Estas presillas son separables, puesto que se diseñaron para que presenten una deflexión de 1.00 mm, con una fuerza de deflexión de 46.57 N.

4.2 Recomendaciones

- Se recomienda indagar a fondo la segunda alternativa priorizada de acuerdo con la Tabla 2.3, misma que destaca la adición de una aplicación móvil, con la finalidad de flexibilizar la configuración al usuario respecto a las preferencias del idioma mediante una interfaz gráfica, adicionalmente, permitiría acceder a mayor capacidad de cómputo.

- Se recomienda abordar la arquitectura de *backend* diseñada basada en la nube de AWS, puesto que los beneficios del pago por uso, flexibilidad y escalabilidad le brindan a la solución la oportunidad de crecimiento de forma horizontal en caso aborde un enfoque de múltiples idiomas.

- Añadir un circuito electrónico que aborde la reducción del ruido del audio de acuerdo con la evidencia destacada en la Figura 3.2, este aspecto permitiría incrementar la calidad de la traducción.

- Al añadir una interfaz que permita flexibilizar la selección de idioma, se recomienda mantener la arquitectura basada en microservicios que aborde la traducción de acuerdo el requerimiento del usuario, este aspecto se puede lograr incrementando las funciones Lambda y modelos en SageMaker.

- En futuras exploraciones, es necesario tomar en consideración la integración de una fuente de alimentación externa al circuito electrónico, destacando las baterías de litio que particularmente, se caracterizan por su alta duración, vida útil, densidad energética, aspecto que le permite ser más compacto.

BIBLIOGRAFÍA

[1] I. O. for Migration, “World migration report 2022.” https://publications.iom.int/books/world-migration-report-2022, 02 2022. (Accessed on 06/13/2022).

[2] U. KUMBAKONAM, “Communication barriers,” *Journal of English Language and Literature*, 03 2016.

[3] S. W. Parker, L. Rubalcava, and G. Teruel, “Schooling inequality and language barriers,” *Economic Development and Cultural Change*, vol. 54, no. 1, pp. 71–94, 2005.

[4] S. K. Bhar and N. A. Abu Bakar, “Language barriers: Feedback from the it industry,” *Journal of Technical Education and Training*, vol. 4, Feb. 2013.

[5] H. Tenzer, M. Pudelko, and M. Zellmer-Bruhn, “The impact of language barriers on knowledge processing in multinational teams,” *Journal of World Business*, vol. 56, no. 2, p. 101184, 2021.

[6] Cisco, “Cisco annual internet report - cisco annual internet report (2018–2023) white paper.” https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html. (Accessed on 06/18/2022).

[7] E. Mack, “Google glass arrives in 2013, and under $1,500.” https://www.cnet.com/culture/confirmed-google-glass-arrives-in-2013-and-under-1500/, 02 2013. (Accessed on 07/17/2022).

[8] M. Tackabery, “Sony smarteyeglass seeks share of wearables space.” https://www.progress.com/blogs/sony-smarteyeglass-seeks-share-of-wearables-space, 02 2014. (Accessed on 07/17/2022).

[9] L. Matney, “An ar glasses pioneer collapses.” https://techcrunch.com/2019/01/10/an-ar-glasses-pioneer-collapses/, 01 2019. (Accessed on 07/17/2022).

[10] P. Ace, “The rise of the vuzix blade developer: A new direction for ar.” https://program-ace.com/blog/vuzix-blade-developer-for-ar/, 12 2019. (Accessed on 07/17/2022).

[11] A. Health, “Snap’s new spectacles let you see the world in augmented reality.” https://www.theverge.com/2021/5/20/22445481/snap-spectacles-ar-augmented-reality-announced, 05 2021. (Accessed on 07/17/2022).

[12] A. Boxall, “Google’s life-changing ar smart glasses demo gave me shivers.” https://www.digitaltrends.com/mobile/google-translate-ar-smartglasses-at-google-io-2022/, 05 2022. (Accessed on 07/17/2022).

[13] Statistics&Data, “The most spoken languages 2022 - statistics and data.” https://statisticsanddata.org/data/the-most-spoken-languages-2022/. (Accessed on 09/10/2022).

[14] D. C. Jeffrey S. Passel, Mark Hugo Lopez, “U.s. hispanic population continued its geographic spread in the 2010s.” https://www.pewresearch.org/fact-tank/2022/02/03/u-s-hispanic-population-continued-its-geographic-spread-in-the-2010s/.

[15] Xrai, “Xrai about us.” https://xrai.glass/about#about. (Accessed on 09/03/2022).

[16] Santander, “Wearables: ¿qué son y para qué se utilizan?.” https://www.santander.com/es/stories/wearables-que-son-y-para-que-se-utilizan, 3 2022. (Accessed on 09/03/2022).

[17] L. Teschler, “What’s inside google glass.” https://www.designworldonline.com/whats-inside-google-glass/, 12 2014. (Accessed on 06/19/2022).

[18] GlobalData, “Wearable tech will dominate consumer iot with $97.5 billion revenues in 2022, forecasts globaldata.” https://www.globaldata.com/wearable-tech-will-dominate-consumer-iot-97-5-billion-revenues-2022\protect\discretionary{\char\hyphenchar\font}{}{}forecasts-globaldata/. (Accessed on 09/03/2022).

[19] NIST, “Sp 800-145, the nist definition of cloud computing | csrc.” https://csrc.nist.gov/publications/detail/sp/800-145/final, 9 2011. (Accessed on 09/03/2022).

[20] S. Neenan and K. Casey, “What is public cloud? everything you need to know.” https://www.techtarget.com/searchcloudcomputing/definition/public-cloud. (Accessed on 09/03/2022).

[21] S. Neenan and K. Casey, “What is hybrid cloud? everything you need to know.” https://www.techtarget.com/searchcloudcomputing/definition/hybrid-cloud. (Accessed on 09/03/2022).

[22] RyteWiki, “Https.” https://es.ryte.com/wiki/HTTPS. (Accessed on 09/03/2022).

[23] “Por qué pasar de http a https ¿merece la pena? – webebre.net.” https://www.webebre.net/por-que-pasar-de-http-a-https/. (Accessed on 09/03/2022).

[24] Synopsys, “What is open source software and how does it work?.” https://www.synopsys.com/glossary/what-is-open-source-software.html. (Accessed on 09/03/2022).

[25] RedHat, “What is open source?.” https://www.redhat.com/en/topics/open-source/what-is-open-source, 10 2019. (Accessed on 09/03/2022).

[26] B. Perens *et al.*, “The open source definition,” *Open sources: voices from the open source revolution*, vol. 1, pp. 171–188, 1999.

[27] R. V. Jye Sawtell-Rickson, “What is artificial intelligence (ai)? how does ai work? | built in.” https://builtin.com/artificial-intelligence, 8 2022. (Accessed on 09/03/2022).

[28] I. C. Education, “What is natural language processing?.” https://www.ibm.com/cloud/learn/natural-language-processing, 07 2020. (Accessed on 06/26/2022).

[29] A. Y. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, and A. Y. Ng, “Deep speech: Scaling up end-to-end speech recognition,” *CoRR*, vol. abs/1412.5567, 2014.

[30] S. P. Singh, A. Kumar, H. Darbari, L. Singh, A. Rastogi, and S. Jain, “Machine translation using deep learning: An overview,” in *2017 International Conference on Computer, Communications and Electronics (Comptelix)*, pp. 162–167, 2017.

[31] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” *CoRR*, vol. abs/1706.03762, 2017.

[32] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, (Online), pp. 38–45, Association for Computational Linguistics, Oct. 2020.

[33] J. Tiedemann, "The Tatoeba Translation Challenge – Realistic data sets for low resource and multilingual MT," in *Proceedings of the Fifth Conference on Machine Translation*, (Online), pp. 1174–1182, Association for Computational Linguistics, Nov. 2020.

[34] C. Wang, Y. Tang, X. Ma, A. Wu, D. Okhonko, and J. M. Pino, "fairseq S2T: fast speech-to-text modeling with fairseq," *CoRR*, vol. abs/2010.05171, 2020.

[35] S. L. Hub, "Measuring sound — science learning hub." https://www.sciencelearn.org.nz/resources/573-measuring-sound, 5 2011. (Accessed on 09/04/2022).

[36] G. M. Smith, "Sound measurement with microphone transducers | dewesoft." https://dewesoft.com/daq/sound-measurement-with-microphone-sensors, 1 2022. (Accessed on 09/04/2022).

[37] Adobe, "Best audio format file types | adobe." https://www.adobe.com/creativecloud/video/discover/best-audio-format.html. (Accessed on 09/04/2022).

[38] NXP, "I2c-bus specification and user manual." https://www.nxp.com/docs/en/user-guide/UM10204.pdf, 10 2021. (Accessed on 09/06/2022).

[39] L. George, "I²c or i2c - inter-integrated circuit - working explanation." https://electrosome.com/i2c/#Single_Master_I2C_Bus. (Accessed on 09/06/2022).

[40] S. Campbell, "Basics of the i2c communication protocol." https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/, 2 2016. (Accessed on 09/10/2022).

[41] NXP, "I2s bus." https://www.sparkfun.com/datasheets/BreakoutBoards/I2SBUS.pdf. (Accessed on 09/06/2022).

[42] B. Miller, “The i2s protocol and why digital audio is everywhere | keysight blogs.” https://blogs.keysight.com/blogs/tech/bench.entry.html/2022/04/29/i2s_the_digital_audiointerfacethatrockstoday-6CGE.html#:~:text=I2S%20is%20a%20serial%20bus,select%2C%20and%20a%20data%20line., 4 2022. (Accessed on 09/10/2022).

[43] H. D. Young, R. A. Freedman, and A. L. Ford, *Sears and Zemansky’s university physics*, vol. 2. Pearson education, 2006.

[44] CoolJargon, “Physics - image formation by mirrors.” https://cooljargon.com/ebooks/physics/m42474/index.cnxml.html. (Accessed on 09/04/2022).

[45] S. Mllema, “Thin lenses.” https://cnx.org/contents/lQXiC0Me@81.33:OuYpIlRP@2/15-4-Thin-Lenses. (Accessed on 09/04/2022).

[46] R. F. Tobias Werner, “Projection device for data glasses, data glasses and methods for operating a projection device for a data glasses,” Google Patents, 2015.

[47] C. Klahn, D. Singer, and M. Meboldt, “Design guidelines for additive manufactured snap-fit joints,” *Procedia CIRP*, vol. 50, pp. 264–269, 2016. 26th CIRP Design Conference.

[48] WayBackMachine, “Snap fit design.” https://web.archive.org/web/20120125035927/http://engr.bd.psu.edu/pkoch/plasticdesign/snap_design.htm. (Accessed on 09/17/2022).

[49] AllDataSheet, “Esp-wroom-32 datasheet.” https://pdf1.alldatasheet.com/datasheet-pdf/view/1179101/ESPRESSIF/ESP-WROOM-32.html. (Accessed on 08/12/2022).

[50] Adafruit, “Micrófono max9814.” https://cdn-learn.adafruit.com/downloads/pdf/adafruit-agc-electret-microphone-amplifier-max9814.pdf. (Accessed on 08/12/2022).

[51] R. Electronics, “Oled 4 pin 128 x 64 display module 0.96 inch - datasheet.” https://www.rajguruelectronics.com/Product/1145/OLED%204%20Pin%20128x64%20Display%20module%200.96%20inch%20blue%20color.pdf. (Accessed on 08/11/2022).

[52] G. Brown, “Digital audio basics: Audio sample rate and bit depth.” https://www.izotope.com/en/learn/digital-audio-basics-sample-rate-and-bit-depth.html, 5 2021. (Accessed on 08/19/2022).

[53] C. Greening, “Esp32 audio input - max4466, max9814, sph0645lm4h, inmp441.” https://www.atomic14.com/2020/09/12/esp32-audio-input.html, 9 2020. (Accessed on 08/17/2022).

[54] Arduino, “Analogread - arduino reference.” https://www.arduino.cc/reference/en/language/functions/analog-io/analogread/. (Accessed on 08/17/2022).

[55] H. S. Black and J. O. Edson, “Pulse code modulation,” *Transactions of the American Institute of Electrical Engineers*, vol. 66, no. 1, pp. 895–899, 1947.

[56] Espressif, “Freertos - esp32 - esp-idf programming guide latest documentation.” https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/freertos.html. (Accessed on 09/11/2022).

[57] J. C. Colás, “10 razones para elegir la nube aws amazon web services migrar a aws.” https://www.dreams.es/transformacion-digital/migrar-a-aws/10-razones-para-elegir-la-nube-aws-amazon-web-services. (Accessed on 08/15/2022).

[58] A. Syal, “Hugging face: A step towards democratizing nlp | by anuj syal | towards data science.” https://towardsdatascience.com/hugging-face-a-step-towards-democratizing-nlp-2c79f258c951, 12 2020. (Accessed on 08/15/2022).

[59] HuggingFace, “Modelo s2t-small-mustc-en-es-st · hugging face.” https://huggingface.co/facebook/s2t-small-mustc-en-es-st. (Accessed on 08/15/2022).

[60] M. AI, “Fairseq.” https://ai.facebook.com/tools/fairseq/. (Accessed on 08/15/2022).

[61] AWS, “Amazon api gateway | api management | amazon web services.” https://aws.amazon.com/api-gateway/. (Accessed on 08/15/2022).

[62] AWS, “Serverless computing - aws lambda - amazon web services.” https://aws.amazon.com/lambda/. (Accessed on 08/15/2022).

[63] AWS, “Cloud object storage – amazon s3 – amazon web services.” https://aws.amazon.com/s3/. (Accessed on 08/15/2022).

[64] AWS, “Amazon sagemaker - amazon web services.” https://aws.amazon.com/pm/sagemaker/. (Accessed on 08/15/2022).

[65] B. MaterialScience, “Snap-fit book_final 11-05.pdf.” https://fab.cba.mit.edu/classes/S62.12/people/vernelle.noel/Plastic_Snap_fit_design.pdf.

[66] Ultimaker, “Ficha de datos técnicos pla.” https://docs.rs-online.com/7ade/0900766b81698006.pdf, 5 2017. (Accessed on 09/17/2022).

[67] P. R. Bonenberger, “Snap fit design.” https://web.archive.org/web/20120125035927/http://engr.bd.psu.edu/pkoch/plasticdesign/snap_design.htm, 01 2005.

[68] BlazeMeter, “Create your first api monitoring test.” https://guide.blazemeter.com/hc/en-us/articles/360001835558-Create-Your-First-API-Monitoring-Test. (Accessed on 08/28/2022).

[69] AWS, “Aws pricing calculator.” https://calculator.aws/#/estimate. (Accessed on 08/28/2022).

[70] Alibaba, “White display module 0.96 oled 128x64 screen.” https://www.alibaba.com/product-detail/Most-Popular-IIC-Serisl-White-Display_62313631868.html. (Accessed on 09/18/2022).

[71] DFRobot, “Esp32 dimensions.” https://wiki.dfrobot.com/FireBeetle_ESP32_IOT_Microcontroller%28V3.0%29__Supports_Wi-Fi_&_Bluetooth__SKU__DFR0478. (Accessed on 09/18/2022).

APÉNDICES

Apéndice A: Detalle de ponderaciones por criterio

Tabla A.1 Ponderaciones de alternativas – Criterio: Peso y dimensiones

	Alternativa A	Alternativa B	Alternativa C	$\Sigma + 1$	Ponderación
Alternativa A		0	0	1.0	0.167
Alternativa B	1.0		0.5	2.5	0.417
Alternativa C	1.0	0.5		2.5	0.417
Total	—	—	—	6.0	1.000

Tabla A.2 Ponderaciones de alternativas – Criterio: Consumo de energía

	Alternativa A	Alternativa B	Alternativa C	$\Sigma + 1$	Ponderación
Alternativa A		0	0	1.0	0.167
Alternativa B	1.0		1.0	3.0	0.500
Alternativa C	1.0	0		2.0	0.333
Total	—	—	—	6.0	1.000

Tabla A.3 Ponderaciones de alternativas – Criterio: Costos involucrados

	Alternativa A	Alternativa B	Alternativa C	$\Sigma + 1$	Ponderación
Alternativa A		0.5	1.0	2.5	0.417
Alternativa B	0.5		1.0	2.5	0.417
Alternativa C	0	0		1.0	0.167
Total	–	–	–	6.0	1.000

Tabla A.4 Ponderaciones de alternativas – Criterio: Tiempo de respuesta

	Alternativa A	Alternativa B	Alternativa C	$\Sigma + 1$	Ponderación
Alternativa A		0.5	0	1.5	0.250
Alternativa B	0.5		0	1.5	0.250
Alternativa C	1.0	1.0		3.0	0.500
Total	–	–	–	6.0	1.000

Apéndice B: Componentes electrónicos

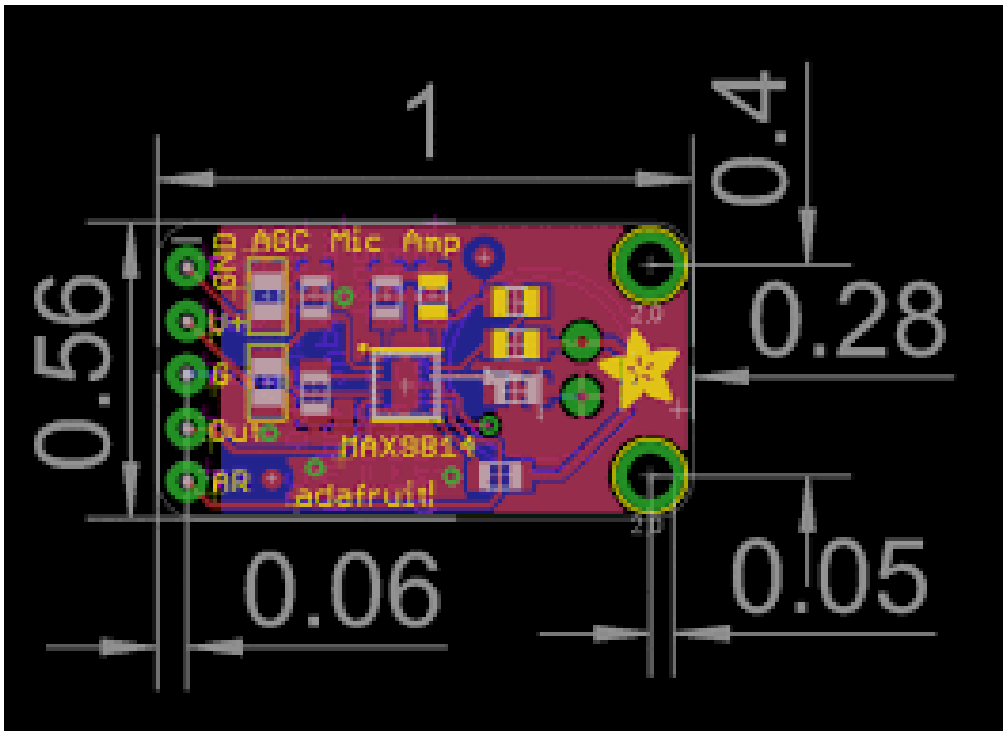


Figura B.1 Dimensiones de micrófono MAX9814 [50]

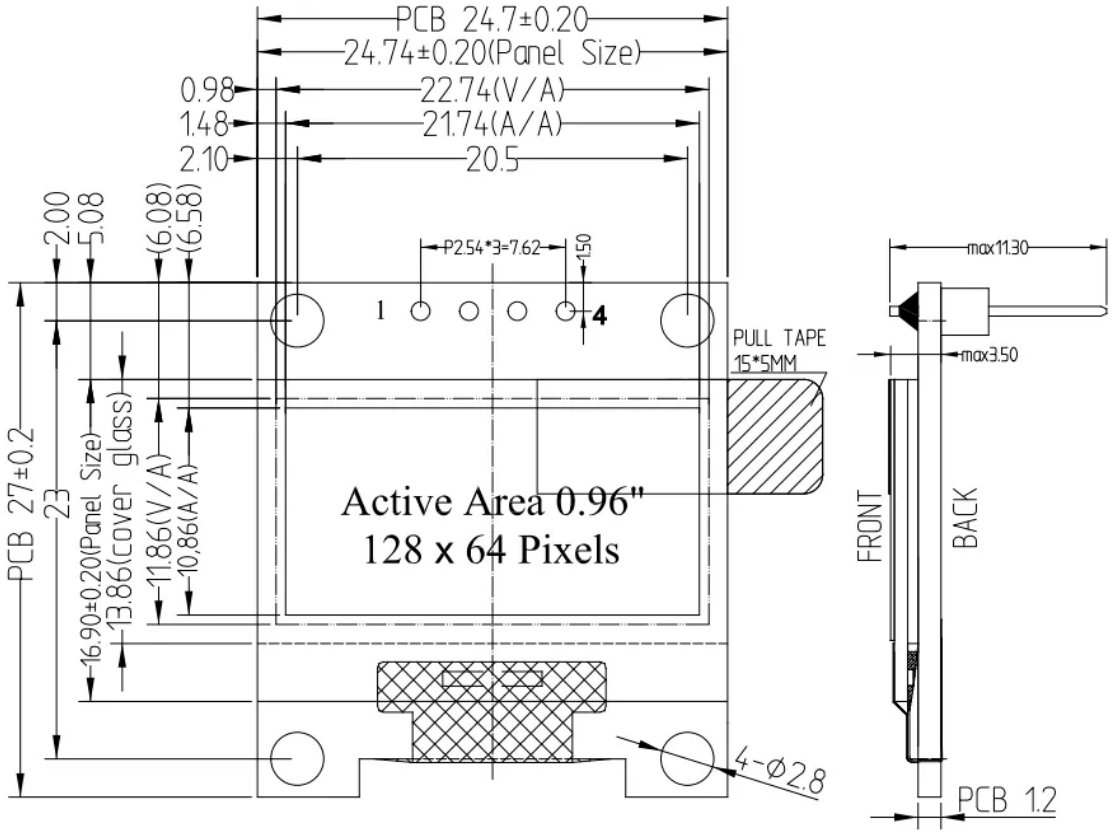


Figura B.2 Dimensiones de pantalla OLED de 4 pines y 0.96 in [70]

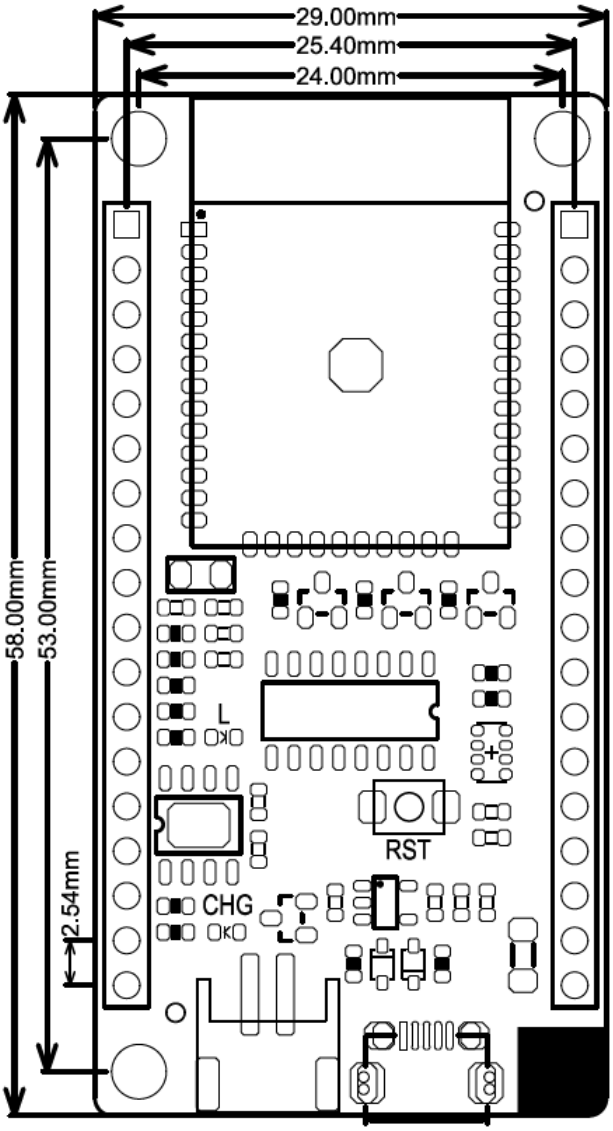


Figura B.3 Dimensiones del microcontrolador ESP32-WROOM-32 [71]

Apéndice C: Código de sistema embebido

Esta sección destaca el código del sistema embebido en lenguaje C++, cabe mencionar que en esta sección se abordarán los diferentes archivos.

Las credenciales requeridas para la conexión a la red WiFi se definieron en el archivo posterior.

`WiFiCredentials.h`

```
#define SSID "<SSID>"
#define PASSWORD "<PASSWORD>"
```

Los archivos posteriores describen los procesos requeridos para el muestreador del ADC.

`ADCSampler.h`

```
#pragma once

#include "I2SSampler.h"

class ADCSampler : public I2SSampler
{
private:
    adc_unit_t m_adcUnit;
    adc1_channel_t m_adcChannel;

protected:
    void configureI2S();
    void unConfigureI2S();

public:
    ADCSampler(adc_unit_t adc_unit, adc1_channel_t adc_channel, const i2s_config_t &i2s_config);
    virtual int read(int16_t *samples, int count);
};
```

ADCSampler.cpp

```
#include "ADCSampler.h"

ADCSampler::ADCSampler(
        adc_unit_t adcUnit,
        adc1_channel_t adcChannel,
        const i2s_config_t &i2s_config
    ): I2SSampler(I2S_NUM_0, i2s_config) {
    m_adcUnit = adcUnit;
    m_adcChannel = adcChannel;
}

void ADCSampler::configureI2S() {
    //init ADC pad
    i2s_set_adc_mode(m_adcUnit, m_adcChannel);
    // enable the adc
    i2s_adc_enable(m_i2sPort);
}

void ADCSampler::unConfigureI2S() {
    // make sure ot do this or the ADC is locked
    i2s_adc_disable(m_i2sPort);
}

int ADCSampler::read(int16_t *samples, int count) {
    // read from i2s
    size_t bytes_read = 0;
    i2s_read(m_i2sPort, samples, sizeof(int16_t) * count, &bytes_read, portMAX_DELAY);
    int samples_read = bytes_read / sizeof(int16_t);
    for (int i = 0; i < samples_read; i++) {
        samples[i] = (2048 - (uint16_t(samples[i]) & 0xfff)) * 15;
    }
    return samples_read;
}
```

Los archivos posteriores describen los procesos requeridos para el muestreador del I2S.

I2SSampler.h

```h
#pragma once

#include <freertos/FreeRTOS.h>
#include <driver/i2s.h>

/**
 * Base Class for both the ADC and I2S sampler
 **/
class I2SSampler {
protected:
    i2s_port_t m_i2sPort = I2S_NUM_0;
    i2s_config_t m_i2s_config;
    virtual void configureI2S() = 0;
    virtual void unConfigureI2S(){};
    virtual void processI2SData(void *samples, size_t count){
        // nothing to do for the default case
    };

public:
    I2SSampler(i2s_port_t i2sPort, const i2s_config_t &i2sConfig);
    void start();
    virtual int read(int16_t *samples, int count) = 0;
    void stop();
    int sample_rate() {
        return m_i2s_config.sample_rate;
    }
};
```

I2SSampler.cpp

```cpp
#include "I2SSampler.h"
#include "driver/i2s.h"

I2SSampler::I2SSampler(
        i2s_port_t i2sPort,
        const i2s_config_t &i2s_config
    ) : m_i2sPort(i2sPort), m_i2s_config(i2s_config){
```

```
}

void I2SSampler::start() {
    //install and start i2s driver
    i2s_driver_install(m_i2sPort, &m_i2s_config, 0, NULL);
    // set up the I2S configuration from the subclass
    configureI2S();
}

void I2SSampler::stop() {
    // clear any I2S configuration
    unConfigureI2S();
    // stop the i2S driver
    i2s_driver_uninstall(m_i2sPort);
}
```

El archivo principal se destaca posteriormente.

`main.cpp`

```
#include <Arduino.h>
#include <WiFi.h>
#include <HTTPClient.h>
#include "WiFiCredentials.h"
#include "I2SMEMSSampler.h"
#include "ADCSampler.h"
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64

WiFiClient *wifiClientADC = NULL;
HTTPClient *httpClientADC = NULL;
ADCSampler *adcSampler = NULL;
```

```
#define ADC_SERVER_URL "http://192.168.100.4:5003/adc_samples"

i2s_config_t adcI2SConfig = {
    .mode = (i2s_mode_t)(I2S_MODE_MASTER | I2S_MODE_RX | I2S_MODE_ADC_BUILT_IN),
    .sample_rate = 12000,
    .bits_per_sample = I2S_BITS_PER_SAMPLE_16BIT,
    .channel_format = I2S_CHANNEL_FMT_ONLY_LEFT,
    .communication_format = I2S_COMM_FORMAT_I2S_LSB,
    .intr_alloc_flags = ESP_INTR_FLAG_LEVEL1,
    .dma_buf_count = 4,
    .dma_buf_len = 1024,
    .use_apll = false,
    .tx_desc_auto_clear = false,
    .fixed_mclk = 0
  };

const int SAMPLE_SIZE = 48000;

void sendData(
    WiFiClient *wifiClient,
    HTTPClient *httpClient,
    const char *url,
    uint8_t *bytes,
    size_t count
  ) {
  httpClient->begin(*wifiClient, url);

  httpClient->addHeader("content-type", "application/octet-stream");

  int response = httpClient->POST(bytes, count);

  if (response > 0) {
    String payload = httpClient->getString();
    Serial.println(payload);

    display.println(payload);
  }
  httpClient->end();
}
```

```
// Task to write samples from ADC to our server
void adcWriterTask(void *param) {
  I2SSampler *sampler = (I2SSampler *)param;
  int16_t *samples = (int16_t *)malloc(sizeof(uint16_t) * SAMPLE_SIZE);
  if (!samples) {
    Serial.println("Failed to allocate memory for samples");
    return;
  }
  while (true) {
    int samples_read = sampler->read(samples, SAMPLE_SIZE);
    sendData(
      wifiClientADC,
      httpClientADC,
      ADC_SERVER_URL,
      (uint8_t *)samples,
      samples_read * sizeof(uint16_t)
    );
  }
}

void setup() {
  Serial.begin(115200);
  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
      Serial.println(F("SSD1306 allocation failed"));
      for(;;);
  }

  display.clearDisplay();

  display.setTextSize(3);
  display.setTextColor(WHITE);
  display.setCursor(0, 10);

  // launch WiFi
  Serial.printf("Connecting to WiFi");
  WiFi.mode(WIFI_STA);
  WiFi.begin(SSID, PASSWORD);
```

```
  while (WiFi.waitForConnectResult() != WL_CONNECTED) {
    Serial.print(".");
    delay(1000);
  }

  Serial.println("WiFi Connected");
  Serial.println("Started up");

  // setup the HTTP Client
  wifiClientADC = new WiFiClient();
  httpClientADC = new HTTPClient();

  // input from analog microphone MAX9814
  // internal analog to digital converter sampling using i2s
  // create samplers
  adcSampler = new ADCSampler(ADC_UNIT_1, ADC1_CHANNEL_7, adcI2SConfig);

  // set up the adc sample writer task
  TaskHandle_t adcWriterTaskHandle;
  adcSampler->start();
  xTaskCreatePinnedToCore(
    adcWriterTask,
    "ADC Writer Task",
    4096,
    adcSampler,
    1,
    &adcWriterTaskHandle,
    1
  );
}

void loop() {
  // All processes are taken by tasks
}
```

Apéndice D: Código de lambda translation-en-es

En esta sección se destaca el código desplegado en AWS, para la lambda expuesta por API Gateway en el lenguaje JavaScript.

```
const fs = require("fs");
const AWS = require("aws-sdk");

AWS.config.region = process.env.REGION;

const lambda = new AWS.Lambda();
const s3bucket = new AWS.S3();

const S3_AUDIO_BUCKET = process.env.S3_AUDIO_BUCKET;
const AUDIO_LAMBDA = process.env.AUDIO_LAMBDA;

exports.handler = async (event) => {
    try {
        console.info({ event });

        const rawData = new Buffer(event.body, "base64");

        const date = new Date();
        const year = date.getYear() + 1900;
        const month = date.getMonth() + 1;
        const day = date.getDate();
        const unixTime = date.getTime();
        const fileName = `${year}/${month}/${day}/${unixTime}.raw`;

        console.info(`Date: ${date}`);

        console.info(`Got ${rawData.length} ADC bytes`);

        const bucketParams = {
            Bucket: S3_AUDIO_BUCKET,
            Key: `raw/${fileName}`,
            Body: rawData
        };
```

```
        const s3response = await s3bucket.upload(bucketParams).promise();

        const lambdaParams = {
            FunctionName: AUDIO_LAMBDA,
            InvocationType: "RequestResponse",
            LogType: "Tail",
            Payload: JSON.stringify({ key: s3response.key, date: unixTime }),
        };

        console.info({ lambdaParams });

        console.info("Start invoke");

        const { Payload: translation } = await lambda.invoke(lambdaParams).promise();

        console.info("End invoke");

        return JSON.parse(translation);

    } catch(e) {
        console.error(e.message);
        return {
            "error": e.message
        };
    }
};
```

Apéndice E: Código de lambda audio-convert

En esta sección se destaca el código desplegado en AWS, para la lambda de conversión de audio en formato PCM a WAV en lenguaje Python.

```
import boto3
import wave
import json
import logging
import os

logger = logging.getLogger()
logger.setLevel(logging.INFO)

s3 = boto3.resource("s3")
sagemaker = boto3.client("sagemaker-runtime")

S3_AUDIO_BUCKET = os.environ.get("S3_AUDIO_BUCKET")
CONTENT_TYPE = "audio/x-audio"
SAGEMAKER_TRANSLATION_ENDPOINT = os.environ.get("SAGEMAKER_TRANSLATION_ENDPOINT")

def lambda_handler(event, context):
    key = event["key"]
    unix_time = event["date"]

    logger.info(f"key: {key}")

    obj = s3.Object(S3_AUDIO_BUCKET, key)
    body = obj.get()["Body"].read()

    logger.info(f"raw file length {len(body)}")

    file_name = "temp_audio"
    wav_key = key.replace("raw", "wav")

    wav_file = f"/tmp/{file_name}.wav"

    with wave.open(wav_file, "wb") as file:
```

```
        file.setparams((2, 2, 12000, 0, "NONE", "NONE"))
        file.writeframes(body)

    with open(wav_file, "rb") as file:
        audio_data = file.read()

        sagemaker_response = sagemaker.invoke_endpoint(
            EndpointName=SAGEMAKER_TRANSLATION_ENDPOINT,
            ContentType=CONTENT_TYPE,
            Accept="application/json",
            Body=audio_data
        )

        s3response = s3.Bucket(S3_AUDIO_BUCKET).put_object(Key=wav_key, Body=audio_data)

    translation = json.loads(sagemaker_response["Body"].read())

    logger.info(translation)
    logger.info(s3response.key)

    return translation
```

Apéndice F: Ecuaciones del diseño de Presilla (*snap fit*)

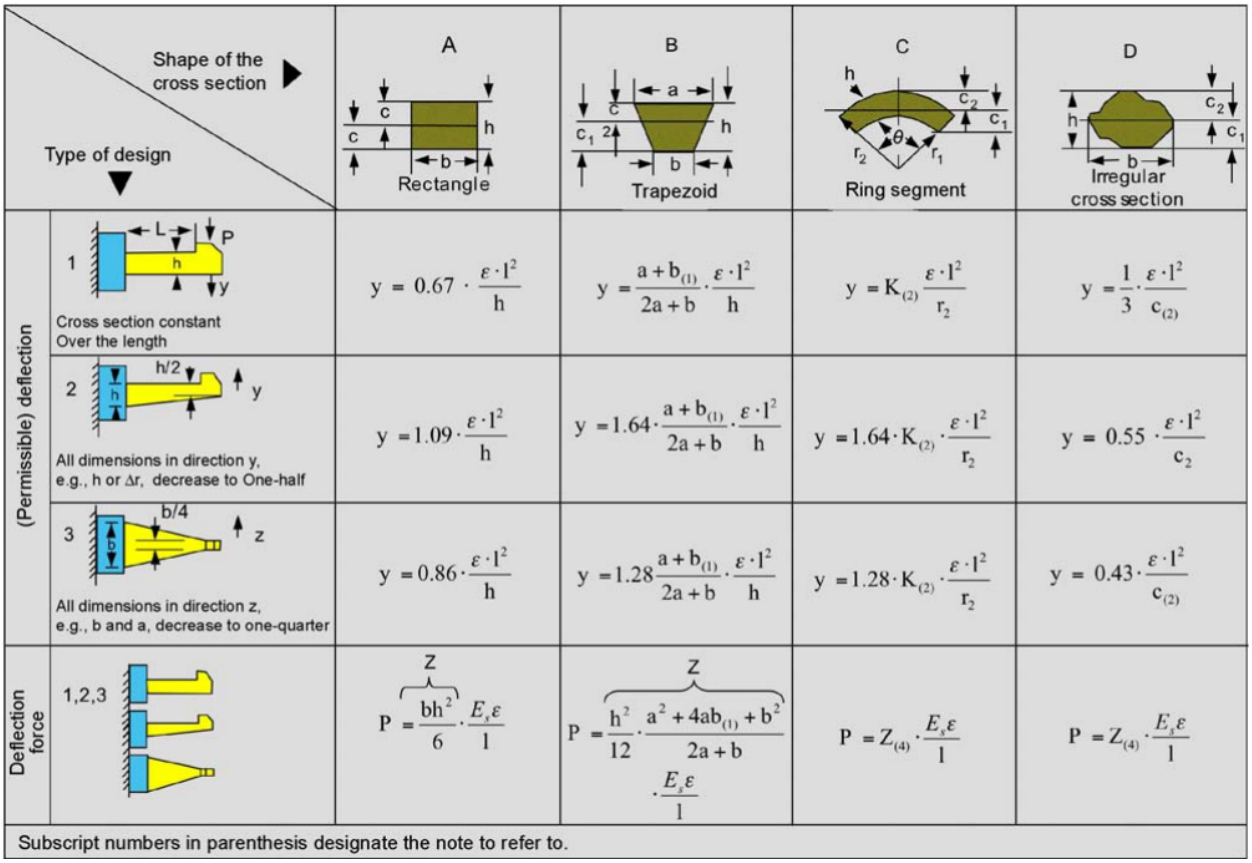


Figura F.1 Ecuaciones para calcular presillas en voladizo [65]

Apéndice G: Planos mecánicos

En este apéndice se adjuntan todos los planos mecánicos desarrollados para el diseño del brazo de las gafas. Las figuras G.1 G.2 resaltan la tapa y la base del brazo respectivamente, cada una de estas piezas se las proyecta en las diferentes vistas necesarias para mostrar sus dimensiones. Finalmente una muestra el ensamblado incluyendo el sistema óptico en la figura G.3.

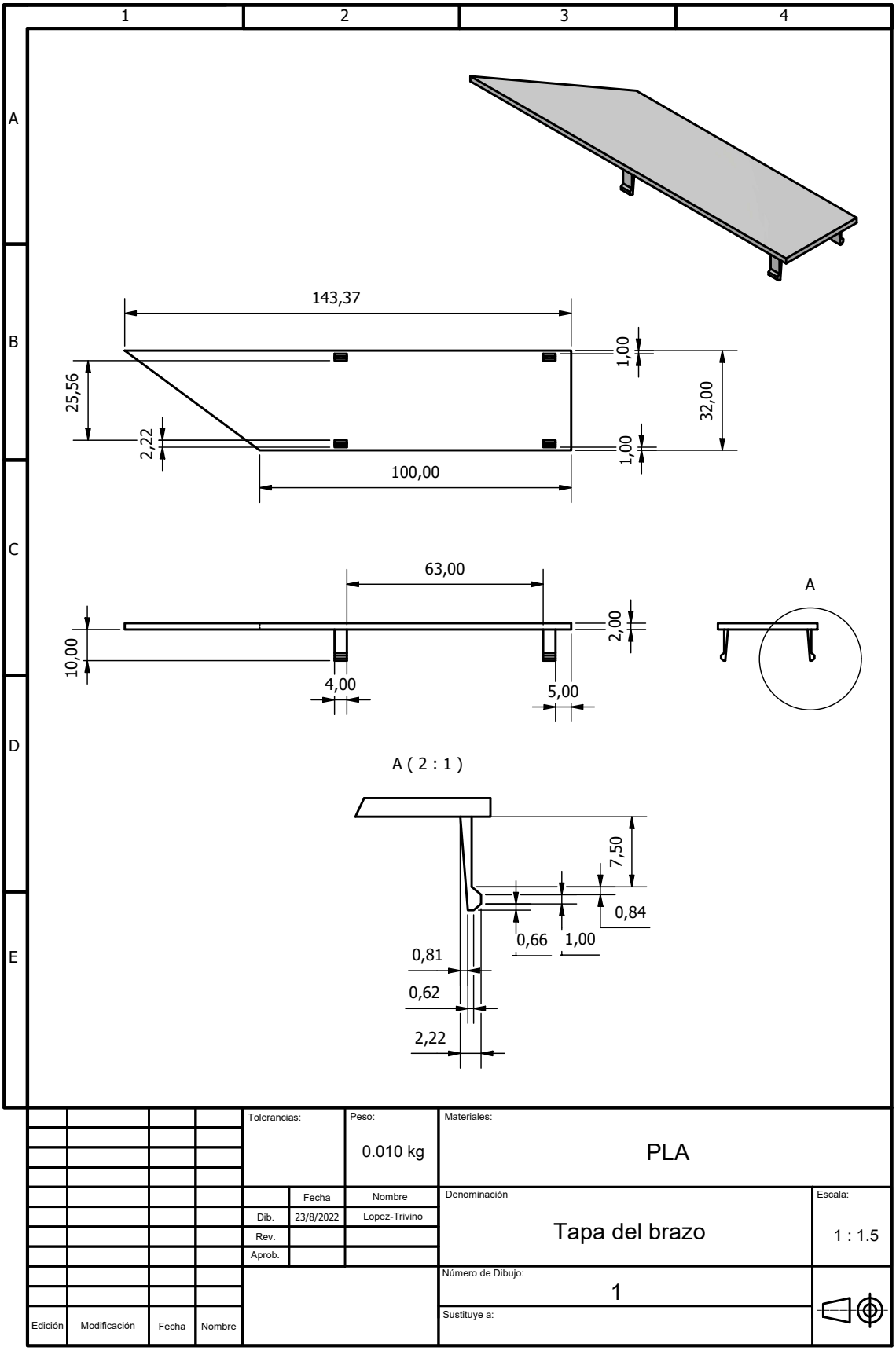


Figura G.1 Tapa del brazo

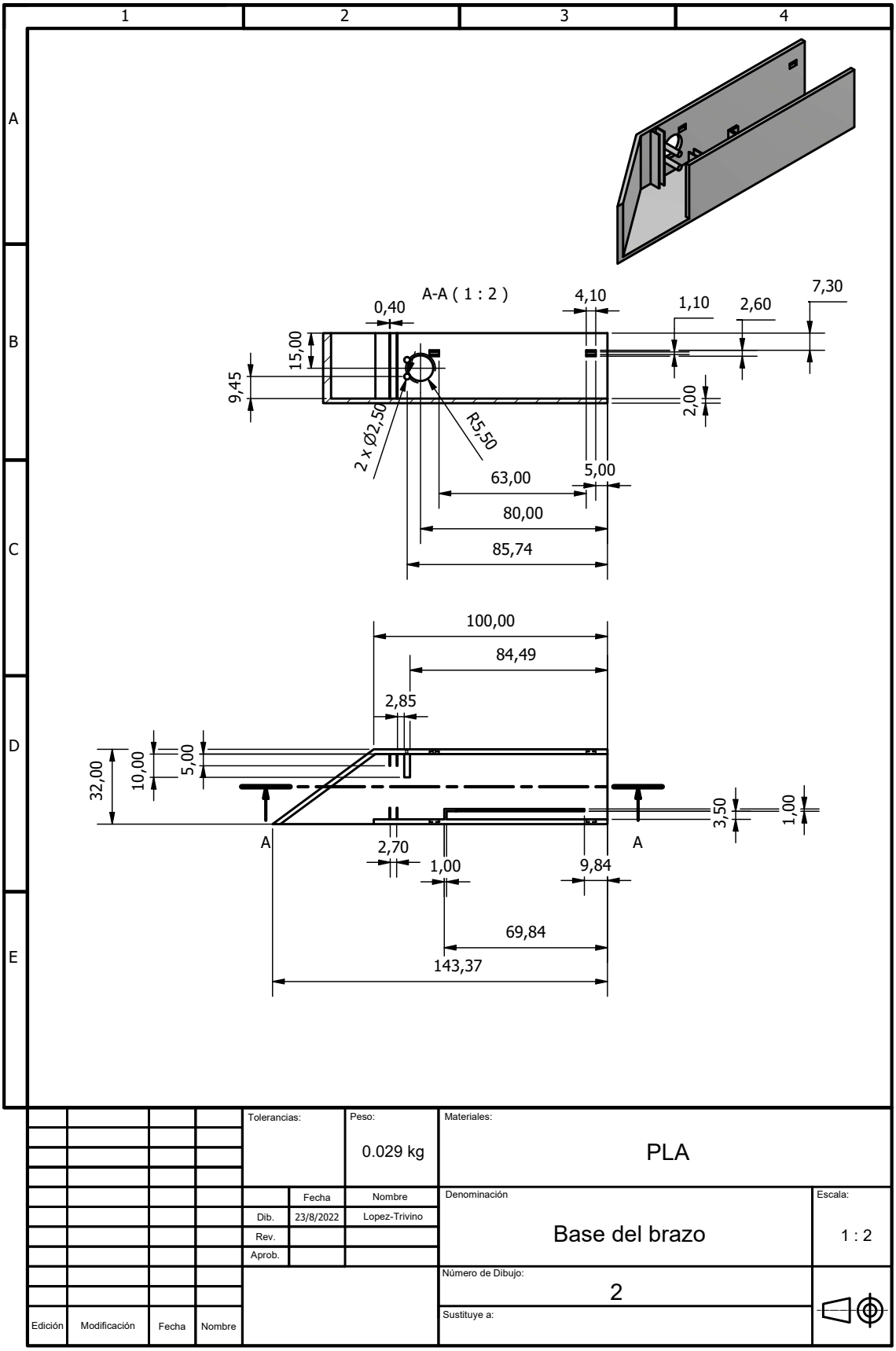
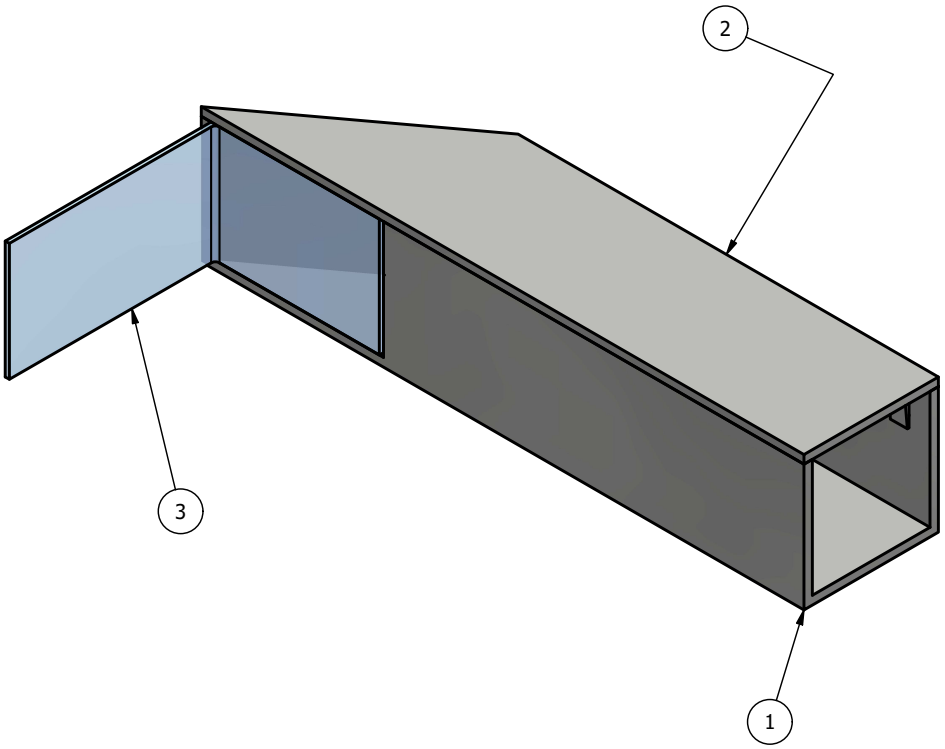


Figura G.2 Base del brazo



ELEMENTO	TÍTULO	CTDAD	MATERIAL	DESCRIPCIÓN
1	Base del brazo	1	PLA	En esta base serán ubicados los componentes electronicos.
2	Tapa del brazo	1	PLA	En esta pieza, hay 4 snap fit que se encargaran de sujetar la tapa con la base del brazo.
3	Lentes de Fresnel	1	Fresnel	Lentes de fresnel

Tolerancias:	Peso: 0.041 kg	Materiales: Varios	
	Fecha	Nombre	Denominación: Brazo con lente — Escala: 1 : 1
Dib.	23/8/2022	Lopez - Trivino	
Rev.			
Aprob.			
			Número de Dibujo: 3
Edición / Modificación / Fecha / Nombre			Sustituye a:

Figura G.3 Brazo con sistema óptico