



**Universidad de Magallanes**  
**Facultad de Ingeniería**  
**Departamento de Ingeniería en Computación e**  
**Informática**

**KubiBot: Informe Técnico**

Ivan Mansilla  
Ayrton Morrison  
Ingeniería Civil en Computación e Informática

Docente guía: Mgt. Eduardo Peña  
Curso: Taller de Integración

2025

## **Resumen**

Resumen bien epico carajo

# ÍNDICE GENERAL

<b>Índice de figuras</b>	<b>ii</b>
<b>1. Introducción</b>	<b>1</b>
<b>2. Objetivos</b>	<b>2</b>
2.1. Objetivo Principal . . . . .	2
2.2. Objetivos Secundarios . . . . .	2
<b>3. Marco teórico</b>	<b>3</b>
<b>4. Desarrollo</b>	<b>4</b>
4.1. Análisis previo . . . . .	4
4.1.1. Módulos del prototipo . . . . .	4
4.1.2. Diseño . . . . .	5
4.2. Movimiento . . . . .	6
4.2.1. Prototipo inicial . . . . .	6
4.2.2. Segundo prototipo . . . . .	8
4.2.3. Tercer prototipo . . . . .	9
4.2.4. Prototipo final . . . . .	10
4.3. Comunicacion . . . . .	11
4.3.1. Arquitectura inicial . . . . .	11
4.3.2. Arquitectura final . . . . .	12
4.4. Comunicación Raspberry-Arduino . . . . .	13
4.5. Diseño . . . . .	13
<b>5. Conclusión</b>	<b>14</b>

# ÍNDICE DE FIGURAS

4.1. Puente H L293D . . . . .	6
4.2. Diagrama del sensor ultrasónico HC-SR04 . . . . .	7
4.3. Conexión del sensor ultrasónico al Arduino . . . . .	7
4.4. Conexión del servomotor al Arduino . . . . .	8
4.5. Conexión de motores al TB6612FNG . . . . .	10
4.6. Diagrama de comunicacion Servidor-Cliente . . . . .	12

# INTRODUCCIÓN

# **OBJETIVOS**

## **2.1. Objetivo Principal**

Diseñar y desarrollar un prototipo de robot de compañía móvil, integrado con un modelo de inteligencia artificial conversacional, capaz de interactuar de forma autónoma y asistir al usuario dentro de un entorno doméstico

## **2.2. Objetivos Secundarios**

Se plantean los siguientes objetivos secundarios:

1. Desarrollar el sistema de movimiento autónomo del robot, incluyendo diseño del chasis, integración de motores y programación de sistema de evitación de obstáculos, utilizando un microcontrolador Arduino Uno
2. Implementar inteligencia artificial en la plataforma Raspberry Pi, abarcando la configuración del sistema, creación de API como intermediario entre host de la LLM y Raspberry PI, e integración de modelos de reconocimiento de voz (Speech-to-text)
3. Integrar periféricos de comunicación (entrada y salida) para la interacción usuario-robot, esto incluyendo instalación de micrófono y altavoz
4. Ensamblar los distintos módulos de hardware del prototipo, procurando coordinación entre Arduino y Raspberry
5. Validar la funcionalidad completa del robot mediante pruebas de software y hardware

## MARCO TEÓRICO

# DESARROLLO

## 4.1. Análisis previo

El punto de partida de este proyecto fue el interés en explorar las aplicaciones prácticas de la inteligencia artificial generativa en un formato físico interactuable, con el fin de crear un dispositivo que genere cercanía y acompañamiento al usuario.

El primer concepto consistía en utilizar un hardware disponible en la institución: una cabeza robótica estática. La idea era dotar a esta cabeza de capacidades conversacionales, aprovechando su estructura existente para simular interacción humana (movimiento de ojos o boca).

Sin embargo, tras un análisis preliminar, se descartó la idea, y se prefirió crear un hardware de cero, ya que entregaría mayor libertad de diseño e integración. Esto llevó entonces hacia un concepto nuevo: un prototipo de robot móvil y compacto.

Esta nueva dirección se eligió por dos ventajas estratégicas:

1. **Simplicidad de integración:** Aunque la movilidad añade el desafío de crear un sistema de navegación, diseñar un chasis propio desde cero simplifica enormemente la integración y cohesión de los componentes que se quisieran utilizar, ya que no tienen que obligatoriamente adaptarse a un sistema pre existente.
2. **Mayor Atractivo e Interacción:** Se determinó que un robot capaz de moverse por la habitación y reaccionar físicamente a su entorno sería percibido por el público como un dispositivo más dinámico y, en definitiva, más atractivo y cercano como "compañero", cumpliendo así el objetivo inicial del proyecto de una forma más efectiva.

### 4.1.1. Módulos del prototipo

Una vez la idea general definida, fue fundamental realizar un análisis de los componentes necesarios para el funcionamiento del prototipo. Para un desarrollo en paralelo y modular, se dividió el robot en dos módulos principales e inicialmente independientes entre sí, con el objetivo de ser conectados una vez cada uno estuviese lo suficientemente avanzado:

1. **Movimiento:** Este módulo englobaría toda la locomoción física del robot. Se compondría de un chasis estructural, un sistema de ruedas impulsadas por motores y un sensor ultrasónico para la detección de obstáculos. Para controlar el movimiento se

escogió el microcontrolador Arduino Uno, debido a su simplicidad de programación y su disponibilidad en el departamento.

2. **Comunicacion:** Modulo en donde residiría todo lo relacionado con la comunicación con el usuario. Se determinó inicialmente el levantar localmente una LLM en un Raspberry PI 5, debido a su diseño compacto y potente. Sin embargo, esta idea fue descartada rápidamente debido a que los modelos utilizados representaban una carga muy grande para el dispositivo. En ella además se conectarían los dispositivos de entrada y salida. Se determinó que la forma de comunicación más cercana y amigable sería a través de voz. Como solución al problema de la potencia, se decidió utilizar una API local para la comunicación con la IA, utilizando el Raspberry PI como un intermediario entre el usuario y el modelo de lenguaje alojado en otra máquina utilizada como servidor.

#### 4.1.2. Diseño

Una vez la funcionalidad determinada, se analizó la presentación visual del prototipo. Por las características de los componentes y los recursos limitados, se necesitaba una carcasa ligera y a su vez accesible. Se optó entonces por crear o buscar un modelo para imprimir en 3D, lo que entonces implicó que el diseño además debiese ser simple para evitar problemas con el filamento o de ensamblaje.

Finalmente se optó por una carcasa completamente cúbica, ya que no solo resulta sumamente fácil de ensamblar, sino que además le da al robot una apariencia agradable. Este diseño entonces se bosquejó, para luego ser trabajado y adaptado en un modelo 3D por Nicolás Poblete, quien durante todo el transcurso del proyecto prestó apoyo en lo que es diseño e impresión del prototipo.

## 4.2. Movimiento

El movimiento durante el desarrollo del proyecto fue uno de los aspectos más desafiantes, no tanto por la complejidad técnica, sino por la limitación de recursos y problemas técnicos imprevistos que surgieron durante la implementación.

Como se mencionó anteriormente, este fue programado con el ambiente de Arduino IDE, utilizando un Arduino UNO como microcontrolador principal. El código fue escrito en C++, utilizando librerías estándar de arduino y un enfoque orientado a objeto para el control de cada componente.

### 4.2.1. Prototipo inicial

Inicialmente se planificó un sistema de locomoción basado en un chasis con dos ruedas a motor, dos ruedas libres y un sensor ultrasónico para la detección de obstáculos, controlados por un Arduino Uno. El puente H utilizado para controlar los motores fue el L293D, componente que se encontraba disponible en el laboratorio; continuación se presenta el diagrama de este en la figura

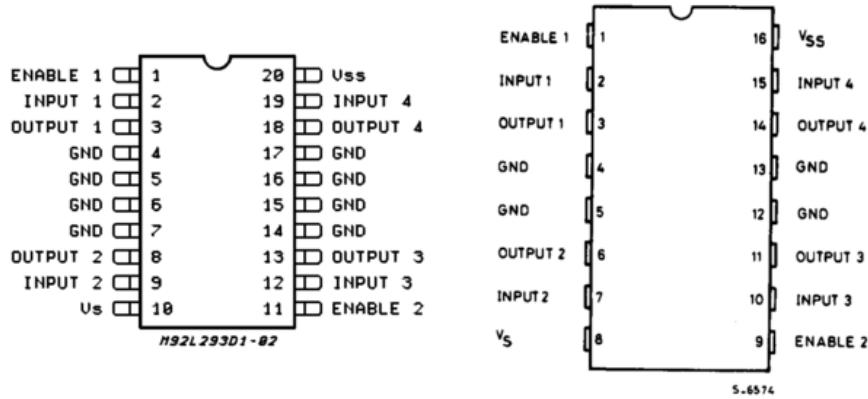


Figura 4.1: Puente H L293D

Los motores DC seleccionados fueron de 6V y 255 RPM, alimentados por una batería externa de 9V para asegurar un suministro de energía estable.

El sensor ultrasónico HC-SR04 se utilizó para medir la distancia a los obstáculos, enviando señales de ultrasonido y midiendo el tiempo que tarda en recibir el eco. Este sensor se conectó al Arduino, que procesaba las lecturas y cortaba el movimiento de los motores si se detectaba un obstáculo. A continuación se presenta el diagrama del sensor

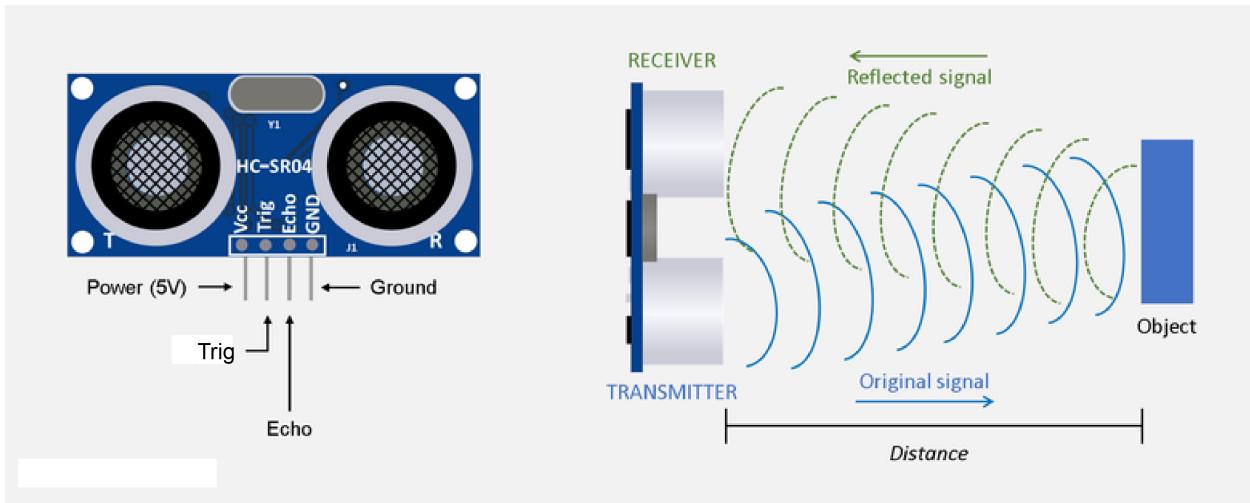


Figura 4.2: Diagrama del sensor ultrasónico HC-SR04

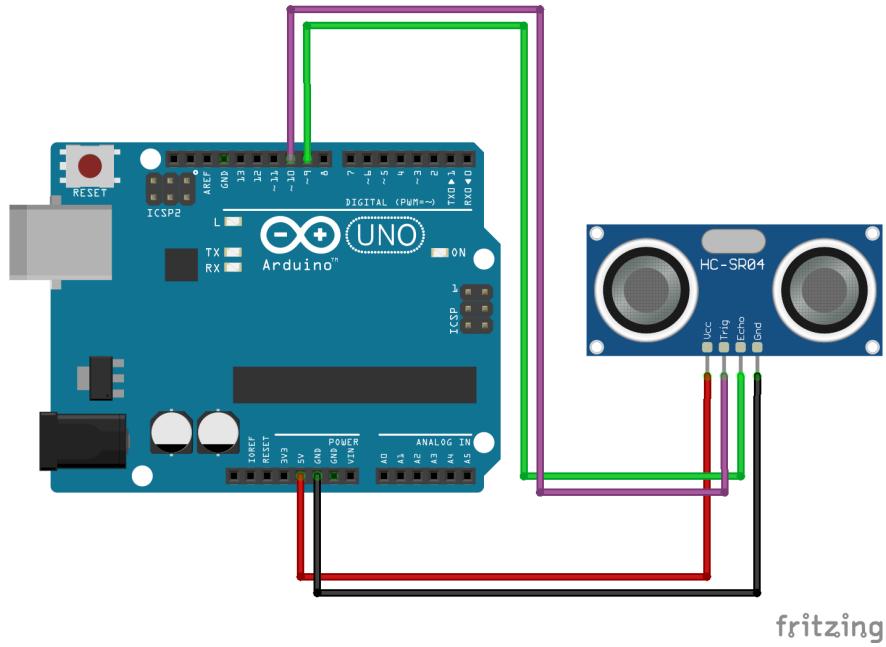


Figura 4.3: Conexión del sensor ultrasónico al Arduino

Los problemas comenzaron a surgir durante las primeras pruebas. El primer problema surgió con el sensor ultrasónico, ya que las lecturas eran inconsistentes y a menudo incorrectas. Después de reemplazar el sensor y revisar las conexiones, se descubrió que el problema en realidad estaba en el código de arduino, que no estaba manejando bien los tiempos de espera y las interrupciones. Tras corregir el código, el sensor comenzó a funcionar correctamente.

El siguiente problema fue netamente de diseño, ya que no se consideró adecuadamente el cómo el prototipo giraría al detectar un obstáculo. Ya que el chasis ya se encontraba impreso

y era complicado modificarlo para añadir un mecanismo con un servo, se decidió realizar el giro netamente deteniendo un motor y dejando el otro activo.

#### 4.2.2. Segundo prototipo

Algo que no se consideró en el primer prototipo fue la decisión de camino que el robot tomaría al detectar un obstáculo. Es por esto que se decidió implementar al sistema de detección de obstáculos un servomotor analógico conectado al arduino, al que se encontraría ensamblado el sensor ultrasónico.

De esta forma, se ejecutaría el siguiente flujo:

```
while(true):
    if (distancia_obstaculo < umbral de seguridad):
        1. Detener ambos motores.
        2. Retroceder hasta una distancia segura.
        3. Girar el servo a la izquierda y medir distancia.
        4. Girar el servo a la derecha y medir distancia.
        5. Comparar ambas distancias.
        6. Girar en la dirección con mayor distancia libre.
    else:
        Continuar moviéndose hacia adelante.
```

A continuación se presenta el diagrama de conexión del servomotor al arduino

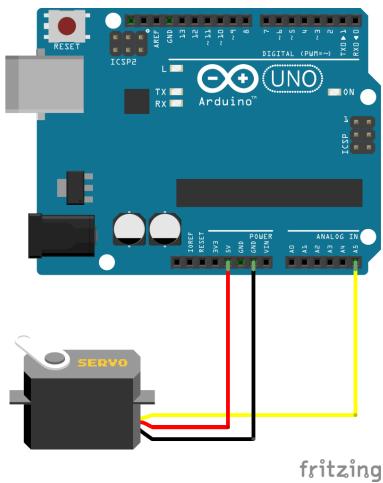


Figura 4.4: Conexión del servomotor al Arduino

#### 4.2.3. Tercer prototipo

Al probar el giro y el movimiento se encontraron dos problemas graves:

1. **Falta de potencia:** Se empezó a notar que el robot tenía dificultados para moverse, sobre todo en superficies con algo de fricción. Esto se debió a que la batería de 9V se estaba quedando sin carga, lo que resultaba en una potencia insuficiente.
2. **Giro ineficiente:** El sistema de giro implementado en donde se detenía un motor y se dejaba el otro activo no funcionó como se esperaba. El robot prácticamente no giraba, tanto por la fricción del suelo como por la inercia ejercida por la rueda detenida.

Para solucionar el primer problema se decidió cambiar la fuente de alimentación a un portapila de 6 pilas AA, entregando un total de 9V. Esto proporcionó no solo la potencia necesaria, sino que además la posibilidad de cambiar las pilas fácilmente cuando se agotaran.

Para solucionar el segundo problema se tuvo que hacer un cambio importante en el circuito, ya que para combatir la inercia y fricción con una potencia adecuada, se decidió añadir dos motores adicionales, así el sistema de locomoción convirtiéndose en uno 4x4. Esto implicó rediseñar el chasis para acomodar los nuevos motores y ruedas, lo que fue posible gracias a la impresión 3D.

También se tuvo que cambiar el puente H L293D por uno que soportara mayor corriente, ya que investigando se descubrió que con este habría riesgo de sobrecalentamiento si se utilizaban los 4 motores simultáneamente. El nuevo puente H utilizado fue el TB6612FNG, el cual además de soportar mayor corriente, se encontraba disponible en el laboratorio. A continuación se presenta el diagrama de este

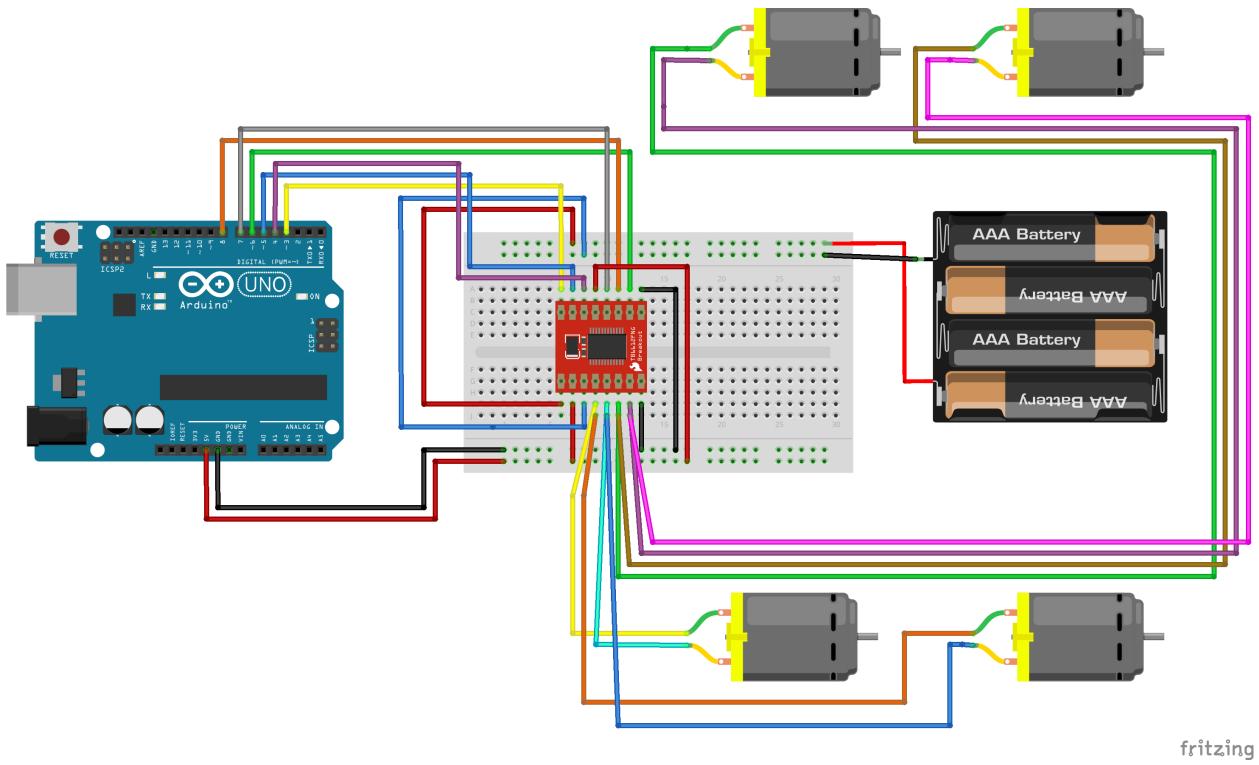


Figura 4.5: Conexión de motores al TB6612FNG

#### 4.2.4. Prototipo final

A pesar de la investigación realizada, el puente H TB6612FNG comenzó a sobrecalentarse al utilizar los 4 motores simultáneamente, lo que generó preocupación por la seguridad del prototipo.

Se tuvo que buscar una solución alternativa, y se optó por conseguir un *shield* de motor para Arduino que fuese adecuado para el manejo de 4 motores DC. El *shield* elegido fue el *Motor Driver Shield L293D*, el cual contiene dos puentes H L293D (utilizado en el primer prototipo) integrados.

## 4.3. Comunicacion

Como se establecio previamente en el analisis del problema, se decidió utilizar un Raspberry PI como el núcleo del modulo de comunicación.

Este en un principio se pensó para alojar localmente un modelo de lenguaje, además de manejar la entrada y salida de audio y todo el postprocesamiento. Sin embargo, tras las primeras pruebas, varios problemas fueron encontrados y la arquitectura del sistema tuvo que ser replanteada.

Todo el código desarrollado para este módulo fue escrito en Python, ya que en este lenguaje existe una amplia gama de librerías y prototipos ya hechos para el manejo de IA, audio y comunicación en red, lo que facilita enormemente el desarrollo.

### 4.3.1. Arquitectura inicial

Para el modelo de lenguaje, se optó utilizar un LLM localmente alojado, debido tanto a la limitación de recursos económicos para utilizar una API comercial, como a la intención de explorar el uso de modelos de lenguaje abiertos. Se seleccionó la herramienta *Ollama*, que permite realizar esto.

Sin embargo, al intentar utilizar diversos modelos, se descubrió que el Raspberry PI 5 no contaba con la potencia suficiente para ejecutar ninguno de ellos de manera fluida. Incluso los modelos más livianos presentaban tiempos de respuesta inaceptables o provocaban que el sistema se congelara, lo que supuso un obstáculo significativo para el desarrollo del proyecto.

Después de considerar diversas opciones, se decidió cambiar la arquitectura del sistema para utilizar el Raspberry PI no como host del modelo del lenguaje, sino como un intermediario entre el usuario y un servidor externo que alojaría el modelo. De esta forma, el Raspberry PI se encargaría de manejar la entrada y salida de audio, mientras que el procesamiento intensivo requerido por el modelo de lenguaje se delegaría a una máquina más potente.

#### 4.3.2. Arquitectura final

El sistema de comunicación se diseñó adaptando una arquitectura cliente-servidor<sup>1</sup>. El Raspberry Pi actúa como cliente, mientras que una máquina externa con mayor capacidad de procesamiento aloja el modelo de lenguaje y actúa como servidor. Esta separación permite que el Raspberry Pi maneje las tareas de entrada y salida de audio, mientras que el servidor se encarga del procesamiento intensivo requerido por el modelo de lenguaje.

En la figura 4.6 se muestra un diagrama de funcionamiento entre los componentes principales del sistema de comunicación.

`client.py` es el script principal que corre en el Raaspberry PI. Este se encarga de detectar constantemente si el usuario ha emitido una entrada de voz específica o Wake Word (en este caso "Hey Bot"). Al detectar esta entrada, el script graba la voz del usuario durante un periodo de tiempo y la envía al servidor a través de una solicitud HTTP POST.

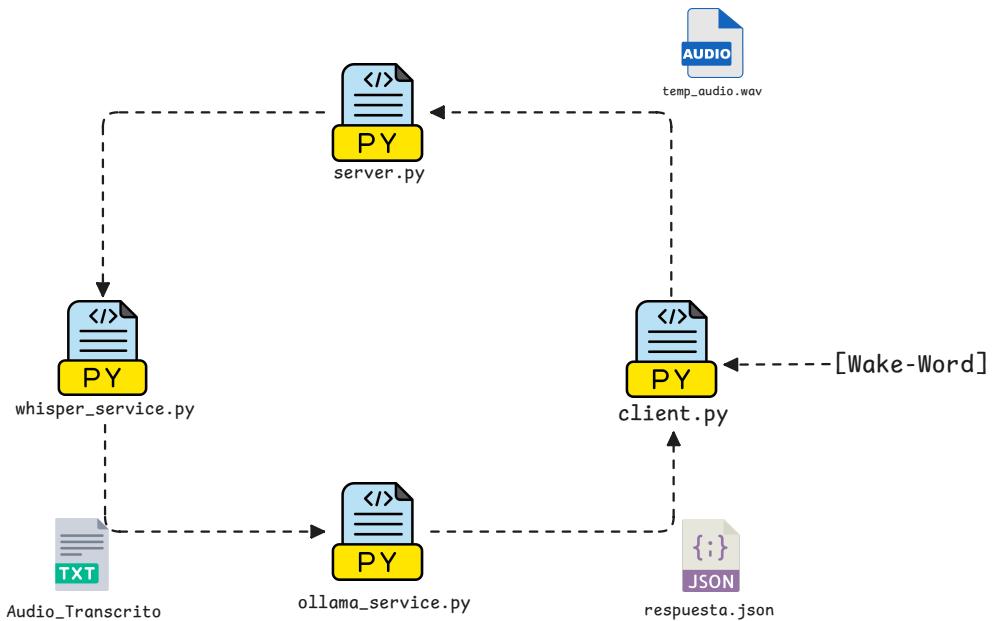


Figura 4.6: Diagrama de comunicacion Servidor-Cliente

<sup>1</sup>Modelo de aplicación que distribuye las tareas entre los proveedores de recursos o servicios y solicitantes de servicios. [1]

#### **4.4. Comunicación Raspberry-Arduino**

#### **4.5. Diseño**

# **CONCLUSIÓN**

## BIBLIOGRAFÍA

- [1] IBM, *Client/server model*, <https://www.ibm.com/docs/en/zos/2.5.0?topic=applications-clientserver-model>, Accessed: 2023-10-27, n.d.