jROBOT / PROYECTO SHI

Shi
Manual de instalación, operación y arquitectura del HUB

Una compañera robótica en español, con rostro expresivo, voz conmutable y un camino claro hacia el conocimiento de proyectos y la telemetría en tiempo real.

PLATAFORMA	VERSIÓN ESTABLE	PUENTE
ESP32-S3 / jRobot S3	xi-stable-v1	shi-bridge-v1

Edición 2026-08-15 / Preparado para Jota

01 Vista general

Qué es Shi y cómo se distribuye el trabajo entre la placa, el PC y la nube.

Shi es una asistente robótica basada en una jRobot S3 con ESP32-S3. La placa controla rostro, LED, micrófono, altavoz, Wi-Fi y protocolo de voz. El cerebro conversacional se selecciona al iniciar.

MODO	CEREBRO	PC	INTERNET	COSTE LLM
Ollama	qwen3.5:4b local	Sí	Solo Edge TTS	Sin tokens OpenAI
GPT	OpenAI API	Sí	Sí	Consume API
XiaoZhi	Nube oficial	No tras configurar	Sí	Según servicio

IDENTIDAD HABLADA	IDENTIDAD TÉCNICA
Se llama Shi. Es la forma que debe pronunciar y mostrar durante una conversación.	El firmware y los repositorios conservan Xi/XiaoZhi para evitar incompatibilidades.

Arquitectura de voz actual

ETAPA	IMPLEMENTACIÓN
Entrada	Micrófono I²S de Shi, 16 kHz
Detección de voz	Silero VAD
Transcripción	Faster-Whisper small, español, CPU INT8
Respuesta local	Ollama + qwen3.5:4b
Síntesis	Edge TTS, es-CL-CatalinaNeural
Salida	Opus a Shi, altavoz I²S a 24 kHz

02 Hardware y pinout

Conexiones validadas del prototipo físico.

- ESP32-S3 con 16 MB de flash y 8 MB de PSRAM.
- OLED SSD1306 de 128 x 64 y rostro a pantalla completa.
- Micrófono y altavoz digitales I²S en modo dúplex.
- LED RGB WS2812 integrado y conversor USB CH343.
- Wi-Fi de 2,4 GHz; unidad original identificada como COM4.

Pinout actual

FUNCIÓN	GPIO	ESTADO
I²S BCLK	4	En uso
I²S WS/LRCK	5	En uso
Altavoz DOUT	6	En uso
Micrófono DIN	7	En uso
OLED SDA / SCL	8 / 9	En uso
Lámpara MCP	18	Reservado
BOOT integrado	0	Reservado
Touch declarado	47	Reservado
Volumen - / +	39 / 40	Reservado
LED WS2812	48	En uso

No hay botones externos instalados. El botón BOOT integrado alterna la conversación después del arranque; durante el inicio puede entrar en configuración Wi-Fi.

03 Movimiento futuro

Reserva provisional para cinco servos; todavía no forma parte del firmware estable.

MOVIMIENTO	SEÑAL PROPUESTA
Cabeza yaw	GPIO 10
Cabeza pitch	GPIO 11
Brazo izquierdo	GPIO 12
Brazo derecho	GPIO 13
Antena	GPIO 14

ALIMENTACIÓN SEGURA
Los GPIO solo llevan señal PWM. Para cinco servos: fuente regulada separada de 5-6 V y 3-5 A, tierra común con la ESP32 y condensador de 1000-2200 uF cerca de la distribución.
No hacer pasar los picos de corriente por el regulador ni por pistas pequeñas de la placa. Si aparecen temblores, ruido de audio o falta de temporizadores, usar PCA9685 por I^2C.

Antes de conectar

- Confirmar que GPIO 10-14 estén físicamente expuestos en la revisión exacta de la placa.
- Medir la fuente bajo carga y verificar tierra común antes de conectar señales.
- Definir límites mecánicos por servo antes de mover cabeza, brazos o antena.
- Conservar una parada segura y nunca deducir movimiento desde una frase ambigua.

04 Lenguaje del LED

Un vistazo basta para saber qué está haciendo Shi.

COLOR / PATRÓN	ESTADO
Azul rápido - 250 ms	Iniciando
Azul - 400 ms	Conectando al servicio
Azul lento - 500 ms	Activando sesión
Naranja - 500 ms	Configurando Wi-Fi
Naranja rápido - 250 ms	Actualizando firmware
Verde fijo	Operativa y en reposo
Cian fijo	Escuchando
Violeta fijo	Hablando
Rojo rápido - 200 ms	Error fatal

Importante

Que el LED cambie al pulsar BOOT confirma que la placa recibió el evento, pero no demuestra que Whisper haya transcrito ni que el servidor haya generado una respuesta.

VERDE	CIAN	VIOLETA	ROJO
Lista	Te escucha	Está hablando	Revisar sistema

05 Instalación en un PC nuevo

Preparación del entorno de firmware y del puente local.

Requisitos

- Windows 10/11 de 64 bits, Git, GitHub CLI y Python 3.10.
- ESP-IDF 5.5.4 para compilar o flashear.
- Ollama, FFmpeg compartido y libopus.
- PC y Shi dentro de una red local con comunicación entre dispositivos.

Clonar firmware y servidor

```
git clone https://github.com/Jcabroc/xiaozhi-xi.git C:\ESP32_Projects\xiaozhi-esp32
git clone https://github.com/Jcabroc/xiaozhi-esp32-server.git
C:\Users\USUARIO\Documents\ChatGPT\XiaoZhi\xiaozhi-esp32-server
```

Instalar herramientas y modelo

```
winget install Ollama.Ollama
winget install BtbN.FFmpeg.LGPL.Shared.8.1
ollama pull qwen3.5:4b
```

Crear el entorno Python

```
py -3.10 -m venv C:\ESP32_Projects\xi-bridge-venv
& C:\ESP32_Projects\xi-bridge-venv\Scripts\python.exe -m pip install -U pip
& C:\ESP32_Projects\xi-bridge-venv\Scripts\python.exe -m pip install -r <servidor>\main\xiaozhi-server\requirements.txt
```

Copiar `bridge/xi-local.config.example.yaml` como `data/.config.yaml` en el servidor. En otro PC deben adaptarse las rutas y la IP definidas al comienzo de los scripts PowerShell.

06 Firmware

Compilación, puerto serie y copia segura.

Conectar Shi con un cable USB de datos. Identificar el puerto:

```
[System.IO.Ports.SerialPort]::GetPortNames()
```

Compilar y flashear

```
$env:IDF_PYTHON_ENV_PATH='C:\Espressif\python_env\idf5.5_py3.11_env'
. 'C:\Espressif\frameworks\esp-idf-v5.5.4\export.ps1'
Set-Location C:\ESP32_Projects\xiaozhi-esp32
idf.py build
idf.py -p COM4 flash
```

PUNTO RECUPERABLE	CONTENIDO
xi-stable-v1	Cara, micrófono, voz y LED estables
shi-bridge-v1	Puente local, selector y documentación

Cambiar COM4 si Windows asignó otro puerto. Nunca modificar la cara estable sin conservar antes binarios, assets, tabla de particiones y sumas SHA-256.

07 Selector de modos

Una sola herramienta administra el cerebro de Shi.

```
# Cerebro local, sin tokens OpenAI
& C:\ESP32_Projects\xiaozhi-esp32\bridge\Set-XiMode.ps1 -Mode Ollama

# Nube XiaoZhi
& C:\ESP32_Projects\xiaozhi-esp32\bridge\Set-XiMode.ps1 -Mode XiaoZhi

# GPT mediante OpenAI API
& C:\ESP32_Projects\xiaozhi-esp32\bridge\Set-XiMode.ps1 -Mode GPT

# Consultar sin cambiar
& C:\ESP32_Projects\xiaozhi-esp32\bridge\Set-XiMode.ps1 -Mode Status
```

El selector administra el proceso del puente, comprueba la OTA local y reinicia Shi por COM4. Para otro puerto: `-Port COM7`. Para reiniciar a mano: `-NoDeviceReset`.

Preparar GPT

```
& C:\ESP32_Projects\xiaozhi-esp32\bridge\Set-XiOpenAIKey.ps1
```

La clave se solicita oculta, se guarda como OPENAI_API_KEY del usuario y nunca entra en Git. GPT queda bloqueado si falta la clave, sin cambiar el modo ni realizar consumo.

08 Uso cotidiano

Cómo conversar sin abortar accidentalmente la respuesta.

PASO	ACCIÓN
1	Esperar el LED verde fijo.
2	Pulsar BOOT una vez o usar la palabra de activación.
3	Hablar cuando el LED esté cian.
4	Terminar la frase y esperar la transcripción.
5	No volver a pulsar BOOT mientras procesa: puede abortar la respuesta.

TIEMPOS NORMALES

Faster-Whisper suele tardar alrededor de dos segundos. Ollama o GPT añaden su propia latencia y la primera carga local después de reiniciar el PC puede tardar más.

Personalidad

Shi habla en femenino, llama Jota al usuario, responde primero con precisión y después puede añadir ironía, curiosidad, alegría o humor negro moderado. Si escucha algo incompleto debe pedir repetición en vez de inventar.

09 Diagnóstico rápido

Síntoma, causa probable y primera acción.

SÍNTOMA	PRIMERA ACCIÓN
No aparece COM	Cable USB de datos; revisar CH343 y reconectar.
LED verde, sin respuesta	Consultar Mode Status y OTA local en puerto 8003.
Entiende palabras absurdas	Reducir ruido; revisar orientación y WAV de diagnóstico.
Tarda demasiado	Confirmar precalentamiento de Ollama y prompt corto.
Habla chino o baja volumen	Verificar plantilla de Shi y voz Catalina es-CL.
Dice once como nombre	Usar Shi en el prompt hablado; Xi solo en código.
Se reinicia con servos	Desconectar servos y revisar fuente, tierra y picos.

Rutas de la instalación original

ELEMENTO	RUTA
Firmware	C:/ESP32_Projects/xiaozhi-esp32
Servidor	Documents/ChatGPT/XiaoZhi/xiaozhi-esp32-server
Datos y logs	C:/ESP32_Projects/xi-bridge-data
Entorno Python	C:/ESP32_Projects/xi-bridge-venv
Backups	C:/ESP32_Projects/XiBackups

10 Shi como voz del HUB

La evolución desde asistente conversacional hacia interfaz del taller.

Shi no necesita reentrenarse después de cada cambio. Necesita una capa de conocimiento y herramientas que entregue información actual, verificable y fechada al modelo que esté activo.

Tres clases de memoria

CLASE	CONTENIDO	REGLA
Documental	Diseños, decisiones, BOM, firmware y manuales	Siempre con fuente y versión
Estable	Nombres, preferencias y configuración del taller	Editable y estructurada
En vivo	Batería, modo, posición, errores y conexión	Con fecha y caducidad

11 Conocimiento actualizable

Cómo lograr respuestas precisas sobre todos los proyectos.

- Cada proyecto conserva docs/ y un archivo de contexto breve.
- Un indexador detecta commits y actualiza solo los fragmentos modificados.
- Toda respuesta documental puede indicar archivo, versión y fecha.
- Los datos reemplazados se marcan obsoletos para evitar pinouts contradictorios.
- Una acción manual permite sincronizar de inmediato; otra tarea puede hacerlo periódicamente.

PRIMERA ETAPA	CUANDO CREZCA
SQLite y un índice vectorial local para documentos, decisiones y metadatos.	PostgreSQL + pgvector; TimescaleDB o InfluxDB para telemetría histórica.

Expectativa realista

Shi puede conocer con gran precisión todo lo documentado e indexado. Si una fuente no existe, está vencida o contradice otra, debe decirlo. La calidad dependerá de mantener datos, versiones y estados con disciplina.

12 Robots y telemetría

MQTT como lenguaje común del ecosistema jRobot.

```
robots/{robot_id}/state
robots/{robot_id}/battery
robots/{robot_id}/mode
robots/{robot_id}/alerts
robots/{robot_id}/telemetry/{sensor}
robots/{robot_id}/commands/{command}
```

Cada mensaje debe incluir robot_id, timestamp, value, unit, quality y versión de esquema. El HUB normaliza placas y fabricantes para que Shi consulte todos los robots con las mismas herramientas.

Herramientas previstas

CONSULTA	EJEMPLO
projects.search	Buscar documentación y decisiones
projects.get_pinout	Obtener pinout de una revisión concreta
robots.get_status	Estado y conexión actuales
robots.get_battery	Carga, voltaje y calidad del dato
robots.get_alerts	Errores y advertencias activas
robots.get_history	Historial de una métrica
robots.request_command	Solicitar una orden confirmada

13 Hoja de ruta del HUB

Avanzar de información pasiva a una voz operativa segura.

FASE	RESULTADO
1	Catálogo de proyectos Markdown/JSON y búsqueda local.
2	RAG local para que Ollama cite fragmentos concretos.
3	Esquema MQTT y un robot conectado en solo lectura.
4	HUB con panel, estado, historial y alertas.
5	Consultas del HUB expuestas a Shi mediante MCP.
6	Órdenes con permisos, validación y confirmaciones.
7	Avisos proactivos: batería, desconexión, temperatura y misión.

PRINCIPIO DE SEGURIDAD

Las consultas pueden ser automáticas y de solo lectura. Las órdenes deben pasar por límites físicos y confirmación de Jota. Detener por seguridad puede ser una excepción; mover actuadores nunca debe deducirse de una frase ambigua.

Ejemplo de respuesta futura

“El robot explorador está al 23 %, en modo retorno y perdió telemetría hace 18 segundos.”

Datos reales, actuales y trazables; no una invención del modelo.

14 Seguridad, mantenimiento y referencias

- No publicar .config.yaml, claves, voces privadas ni grabaciones.
- Mantener firmware, servidor y manual separados de modelos y logs.
- Trabajar en ramas y conservar xi-stable-v1 antes de tocar la cara.
- Registrar cambios de pinout, alimentación y versión física de la placa.
- Revisar permisos antes de conectar el HUB con actuadores reales.

Repositorios

PROYECTO	URL
Firmware personalizado	github.com/Jcabroc/xiaozhi-xi
Servidor personalizado	github.com/Jcabroc/xiaozhi-esp32-server
Proyecto original	github.com/78/xiaozhi-esp32

Shi está lista para conversar. El siguiente capítulo es enseñarle a consultar el taller sin dejar de saber cuándo debe decir: no tengo ese dato todavía.

jRobot / Jota + Kira / 2026