

Proyecto 1 – Uso de un protocolo existente (MCP)

CC3067 Redes – Universidad del Valle de Guatemala

Estudiante: Eber Alexander Cuxé Tan **Carné:** 22648

Fecha: 23/09/2025

Índice

1. [Antecedentes](#)
2. [Objetivos del proyecto](#)
3. [Arquitectura y alcance](#)
4. [Características implementadas](#)
5. [Instalación y uso](#)
6. [Servidores MCP desarrollados](#)
7. [Pruebas y resultados](#)
8. [Análisis de comunicación \(Wireshark\)](#)
9. [Conclusiones](#)
10. [Comentarios](#)
11. [Referencias](#)

1. Antecedentes

Los Large Language Models (LLMs) requieren “herramientas” para interactuar con el mundo real. MCP (Model Context Protocol) propone un estándar abierto para integrar herramientas a agentes/LLMs, desacoplando la implementación de la herramienta del modelo. MCP utiliza **JSON-RPC** y define roles de **host**, **cliente** y **servidor** para interoperabilidad y reutilización. (Guía del curso y recursos MCP: ver Referencias).

2. Objetivos del proyecto

- Implementar un protocolo basado en estándares (MCP/JSON-RPC).
- Comprender la arquitectura/servicios de MCP.
- Implementar un **servidor MCP local** y un **servidor MCP remoto**, y consumirlos desde el host conversacional.
- Integrar un LLM por API con **modo planificador** (planner) que decide cuándo llamar herramientas.
- Capturar con **Wireshark** las interacciones host ↔ servidor remoto y **clasificar** los mensajes JSON-RPC (sincronización, solicitud, respuesta) explicando por **capas**.

3. Arquitectura y alcance

- **Host (chat por terminal):** LLM on/off, **modo planificador**, logging.
- **Cliente STUDIO:** JSON-RPC por STDIO binario (framing `Content-Length` + `Content-Type`).
- **Servidor MCP Local (BearingPro-MCP):** selección/verificación de rodamientos con catálogo local, preparado para migración a **API SKF** sin tocar el host.
- **Servidor MCP Remoto (Cloud Run):** endpoint HTTP `/mcp` con `initialize` y `tools/call`.

Flujo (planner): Usuario → Host → LLM (plan JSON) → Host llama Tool MCP → Host devuelve observación → LLM responde.

4. Características implementadas

- Chat por terminal con LLM on/off.
- Planner: la LLM propone un plan JSON (`action: call_tool|answer`); el host llama MCP y realimenta resultados.
- MCP local (STDIO): tools `catalog_list`, `select_bearing`, `verify_point`.
- Catálogo enriquecible por `catalog.json`.
- MCP remoto (Cloud Run): tools demo `echo`, `time_now`, `add`.
- Logs de interacción.

5. Instalación y uso

Requisitos: Python 3.11+, `pip install anthropic requests` (LLM opcional).

Variables de entorno (Windows):

```
set BEARINGPRO_CMD=python local_servers/bearingpro/main.py
set REMOTE_MCP_URL=https://<service>.run.app/mcp
set ANTHROPIC_API_KEY=... (opcional)
set ANTHROPIC_MODEL=... (opcional)
```

Ejecutar host: `py -m host.chat`

Comandos:

- **Planner** `modo planner on/off`
- **Local** catálogo, selección `Fr=.. Fa=.. rpm=.. L10h=..`, verificar <modelo> con `Fr=.. rpm=..`
- **Remoto** `remoto init, remoto hora, remoto suma 3 4`
- **LLM** `modo llm on/off, modo llm on (por defecto).`

6. Servidores MCP desarrollados

6.1 MCP Local — BearingPro-MCP

Tools:

- `catalog_list()` → { ok, models:[...] }
- `select_bearing(Fr_N, Fa_N, rpm, L10h_target)` → { ok, candidates:[...], P_equiv_N }
- `verify_point(model, Fr_N, Fa_N, rpm, L10h_target)` → { ok, ..., L10h_pred, meets_target, margin_percent }

Cálculos: $P \approx Fr + 1.5 \cdot Fa$; $L10h = (C/P)^3 \cdot 1e6 / (60 \cdot rpm)$.

6.2 MCP Remoto — Cloud Run (HTTP)

Endpoint: POST `/mcp` (JSON-RPC: `initialize`, `tools/call`). Tools: `echo`, `time_now`, `add`.

7. Pruebas y resultados

- **Local:** selección/validación con pocos datos, planner operativo.
- **Remoto:** remoto init; remoto hora y remoto suma responden correctamente.

8. Análisis de comunicación (Wireshark)

8.1 Procedimiento

Filtros: SNI `tls.handshake.extensions_server_name contains "remote-mcp"`, TCP `443`
`tcp.port == 443`, Stream `tcp.stream == N`, QUIC/HTTP3 `quic || http3`. (Opcional)
SSLKEYLOGFILE para descifrar TLS/QUIC.

8.2 Evidencias

tls.handshake.extensions_server_name contains "remote-mcp"						
No.	Time	Source	Destination	Protocol	TCP Segment Len	Info
273	23:36:50.976394	2800:98:1617:51c8:a...	2600:1900:4244:200::	TLSv1.3		517 Client Hello (SNI=remote-mcp-510455560434.us-central1.run.app)
619	23:37:02.956504	2800:98:1617:51c8:a...	2600:1900:4244:200::	TLSv1.3		517 Client Hello (SNI=remote-mcp-510455560434.us-central1.run.app)
1677	23:37:05.894809	2800:98:1617:51c8:a...	2600:1900:4244:200::	TLSv1.3		517 Client Hello (SNI=remote-mcp-510455560434.us-central1.run.app)
1927	23:37:13.169202	2800:98:1617:51c8:a...	2600:1900:4244:200::	TLSv1.3		517 Client Hello (SNI=remote-mcp-510455560434.us-central1.run.app)
2143	23:37:25.036222	2800:98:1617:51c8:a...	2600:1900:4244:200::	TLSv1.3		517 Client Hello (SNI=remote-mcp-510455560434.us-central1.run.app)
2533	23:37:39.157231	2800:98:1617:51c8:a...	2600:1900:4244:200::	TLSv1.3		517 Client Hello (SNI=remote-mcp-510455560434.us-central1.run.app)

Figura 1 – Client Hello con SNI al servicio de Cloud Run.

tcp.stream eq 77						
No.	Time	Source	Destination	Protocol	TCP Segment Len	Info
2180	23:37:25.000966	2800:98:1617:51c8:a...	2600:1900:4244:200::	TCP	0	65503 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1440 WS=256 SACK_PERM=0
2181	23:37:25.035167	2600:1900:4244:200::	2800:98:1617:51c8:a...	TCP	0	443 → 65503 [SYN, ACK] Seq=0 Ack=65503 Len=0 MSS=1440 WS=256 SACK_PERM=0
2142	23:37:25.035376	2800:98:1617:51c8:a...	2600:1900:4244:200::	TCP	0	65503 → 443 [ACK] Seq=1 Ack=1 Win=65280 Len=0
2143	23:37:25.036222	2800:98:1617:51c8:a...	2600:1900:4244:200::	TLSv1.3		517 Client Hello (SNI=remote-mcp-510455560434.us-central1.run.app)
2145	23:37:25.064949	2600:1900:4244:200::	2800:98:1617:51c8:a...	TCP	0	443 → 65503 [ACK] Seq=1 Ack=518 Win=268800 Len=0
2146	23:37:25.076998	2600:1900:4244:200::	2800:98:1617:51c8:a...	TLSv1.3		1220 Server Hello, Change Cipher Spec
2147	23:37:25.076998	2600:1900:4244:200::	2800:98:1617:51c8:a...	TCP		1220 443 → 65503 [PSH, ACK] Seq=1221 Ack=518 Win=268800 Len=1220 [TCP PDU reassembled in 2151]
2148	23:37:25.076998	2600:1900:4244:200::	2800:98:1617:51c8:a...	TCP		1220 443 → 65503 [ACK] Seq=2441 Ack=518 Win=268800 Len=1220 [TCP PDU reassembled in 2151]
2149	23:37:25.076998	2600:1900:4244:200::	2800:98:1617:51c8:a...	TCP		1220 443 → 65503 [PSH, ACK] Seq=3661 Ack=518 Win=268800 Len=1220 [TCP PDU reassembled in 2151]
2150	23:37:25.076998	2600:1900:4244:200::	2800:98:1617:51c8:a...	TCP		1220 443 → 65503 [ACK] Seq=4881 Ack=518 Win=268800 Len=1220 [TCP PDU reassembled in 2151]
2151	23:37:25.076998	2600:1900:4244:200::	2800:98:1617:51c8:a...	TLSv1.3		701 Application Data
2152	23:37:25.077225	2800:98:1617:51c8:a...	2600:1900:4244:200::	TCP	0	65503 → 443 [ACK] Seq=518 Ack=6802 Win=65280 Len=0
2153	23:37:25.082110	2800:98:1617:51c8:a...	2600:1900:4244:200::	TLSv1.3		80 Change Cipher Spec, Application Data
2154	23:37:25.082742	2800:98:1617:51c8:a...	2600:1900:4244:200::	TLSv1.3		253 Application Data
2155	23:37:25.082877	2800:98:1617:51c8:a...	2600:1900:4244:200::	TLSv1.3		126 Application Data
2156	23:37:25.111008	2600:1900:4244:200::	2800:98:1617:51c8:a...	TCP	0	443 → 65503 [ACK] Seq=6802 Ack=851 Win=268800 Len=0
2157	23:37:25.116356	2600:1900:4244:200::	2800:98:1617:51c8:a...	TCP	0	443 → 65503 [ACK] Seq=6802 Ack=977 Win=268800 Len=0
2158	23:37:25.154499	2600:1900:4244:200::	2800:98:1617:51c8:a...	TLSv1.3		957 Application Data, Application Data
2159	23:37:25.155640	2800:98:1617:51c8:a...	2600:1900:4244:200::	TCP	0	65503 → 443 [FIN, ACK] Seq=977 Ack=7759 Win=64512 Len=0
2160	23:37:25.184389	2600:1900:4244:200::	2800:98:1617:51c8:a...	TCP	0	443 → 65503 [FIN, ACK] Seq=7759 Ack=978 Win=268800 Len=0
2161	23:37:25.184506	2800:98:1617:51c8:a...	2600:1900:4244:200::	TCP	0	65503 → 443 [ACK] Seq=978 Ack=7760 Win=64512 Len=0

Figura 2 – Handshake TCP: SYN, SYN/ACK, ACK.

tcp.stream eq 77						
No.	Time	Source	Destination	Protocol	TCP Segment Len	Info
2140	23:37:25.006966	2800:1900:1617:51c8:a...	2600:1900:4244:200::	TCP	0	65503 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1440 WS=256 SACK_PERM
2141	23:37:25.035192	2600:1900:4244:200::	2800:98:1617:51c8:a...	TCP	0	443 → 65503 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1392 SACK_PERM WS=256
2142	23:37:25.035376	2800:98:1617:51c8:a...	2600:1900:4244:200::	TCP	0	65503 → 443 [ACK] Seq=1 Ack=1 Win=65280 Len=0
2143	23:37:25.036222	2800:98:1617:51c8:a...	2600:1900:4244:200::	TLSv1.3	517	Client Hello (SNI=remote-mcp-510455560434.us-central1.run.app)
2145	23:37:25.064949	2600:1900:4244:200::	2800:98:1617:51c8:a...	TCP	0	443 → 65503 [ACK] Seq=1 Ack=518 Win=268800 Len=0

▶ Frame 2143: 591 bytes on wire (4728 bits), 591 bytes captured (4728 bits) ▶ Ethernet II, Src: AzureWaveTec_33:de:ff (a8:41:f4:33:de:ff), Dst: HuaweiTechno_ad:a4:ab (58:56:c2:ad:a4:ab) ▶ Internet Protocol Version 6, Src: 2800:98:1617:51c8:acf0:5c06:b094:d35f, Dst: 2600:1900:4244:200:: ▶ Transmission Control Protocol, Src Port: 65503, Dst Port: 443, Seq: 1, Ack: 1, Len: 517 ▶ Transport Layer Security						
▼ TLSv1.3 Record Layer: Handshake Protocol: Client Hello Content Type: Handshake (22) Version: TLS 1.0 (0x0301) Length: 512						
▼ Handshake Protocol: Client Hello Handshake Type: Client Hello (1) Length: 508						
▶ Version: TLS 1.2 (0x0303) Random: 7d5b784a4707d2a240820c236865313027075cc241e9828db8c4b54c42540b58 Session ID Length: 32 Session ID: 6d8558cdd7c3417453fc443e6c2a168273f011428448ec575c08a2eaa1280bbe Cipher Suites Length: 36 Cipher Suites (18 suites) Compression Methods Length: 1 Compression Methods (1 method) Extensions Length: 399 Extension: server_name (len=48) name=remote-mcp-510455560434.us-central1.run.app Extension: ec_point_formats (len=4) Extension: supported_groups (len=22) Extension: application_layer_protocol_negotiation (len=11) Extension: encrypt_then_mac (len=0) Extension: extended_master_secret (len=0) Extension: post_handshake_auth (len=0) Extension: signature_algorithms (len=42) Extension: supported_versions (len=5) TLS 1.3, TLS 1.2 Extension: psk_key_exchange_modes (len=2) Extension: key_share (len=38) x25519 Extension: padding (len=179)						

Figura 3a – TLS 1.3 Client Hello.

tcp.stream eq 77						
No.	Time	Source	Destination	Protocol	TCP Segment Len	Info
2140	23:37:25.006966	2800:1900:1617:51c8:a...	2600:1900:4244:200::	TCP	0	65503 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1440 WS=256 SACK_PERM
2141	23:37:25.035192	2600:1900:4244:200::	2800:98:1617:51c8:a...	TCP	0	443 → 65503 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1392 SACK_PERM WS=256
2142	23:37:25.035376	2800:98:1617:51c8:a...	2600:1900:4244:200::	TCP	0	65503 → 443 [ACK] Seq=1 Ack=1 Win=65280 Len=0
2143	23:37:25.036222	2800:98:1617:51c8:a...	2600:1900:4244:200::	TLSv1.3	517	Client Hello (SNI=remote-mcp-510455560434.us-central1.run.app)
2145	23:37:25.064949	2600:1900:4244:200::	2800:98:1617:51c8:a...	TCP	0	443 → 65503 [ACK] Seq=1 Ack=518 Win=268800 Len=0
2146	23:37:25.076998	2600:1900:4244:200::	2800:98:1617:51c8:a...	TLSv1.3	1220	Server Hello, Change Cipher Spec
2147	23:37:25.076998	2600:1900:4244:200::	2800:98:1617:51c8:a...	TCP	1220	443 → 65503 [PSH, ACK] Seq=1221 Ack=518 Win=268800 Len=1220 [TCP PDU reassm

▶ Frame 2146: 1294 bytes on wire (10352 bits), 1294 bytes captured (10352 bits) ▶ Ethernet II, Src: HuaweiTechno_ad:a4:ab (58:56:c2:ad:a4:ab), Dst: AzureWaveTec_33:de:ff (a8:41:f4:33:de:ff) ▶ Internet Protocol Version 6, Src: 2600:1900:4244:200::, Dst: 2800:98:1617:51c8:acf0:5c06:b094:d35f ▶ Transmission Control Protocol, Src Port: 443, Dst Port: 65503, Seq: 1, Ack: 518, Len: 1220 ▶ Transport Layer Security						
▼ TLSv1.3 Record Layer: Handshake Protocol: Server Hello Content Type: Handshake (22) Version: TLS 1.2 (0x0303) Length: 122						
▼ Handshake Protocol: Server Hello Handshake Type: Server Hello (2) Length: 118						
▶ Version: TLS 1.2 (0x0303) Random: def249bf383d5e3c3b798cfe72abc41f161969b297082bebc359b1d77402f1f0 Session ID Length: 32 Session ID: 6d8558cdd7c3417453fc443e6c2a168273f011428448ec575c08a2eaa1280bbe Cipher Suite: TLS_AES_256_GCM_SHA384 (0x1302) Compression Method: null (0) Extensions Length: 46 Extension: key_share (len=36) x25519 Extension: supported_versions (len=2) TLS 1.3 [JA3S Fullstring: 771,4866,51-43] [JA3S: 907bf3ecef1c987c889946b737b43de8]						
▼ TLSv1.3 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec Content Type: Change Cipher Spec (20) Version: TLS 1.2 (0x0303) Length: 1 Change Cipher Spec Message TLS segment data (1087 bytes)						

Figura 3b – TLS 1.3 Server Hello.

No.	Time	Source	Destination	Protocol	TCP Segment Len	Info
127	01:11:02.013600	2800:1901:81d4:200::	2800:1901:81d4:200::	TCP	0	51480 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1440 WS=256 SACK_PERM
128	01:11:02.040900	2800:1901:81d4:200::	2800:98:1617:51c8:2...	TCP	0	443 → 51480 [SYN] ACK=1 Seq=0 Ack=1 Win=65535 Len=0 MSS=1440 SACK_PERM WS=256
129	01:11:02.041040	2800:98:1617:51c8:2...	2800:1901:81d4:200::	TCP	0	51480 → 443 [ACK] Seq=1 Ack=1 Win=65280 Len=0
130	01:11:02.041581	2600:1403:9c00:18::	2800:98:1617:51c8:2...	TCP	0	443 → 51479 [ACK] Seq=4063 Ack=1834 Win=69760 Len=0
131	01:11:02.041581	2600:1403:9c00:18::	2800:98:1617:51c8:2...	TLSv1.3	303	Application Data
132	01:11:02.041959	2600:1403:9c00:18::	2800:98:1617:51c8:2...	TLSv1.3	287	Application Data
133	01:11:02.042036	2800:98:1617:51c8:2...	2600:1403:9c00:18::	TCP	0	51479 → 443 [ACK] Seq=1834 Ack=4653 Win=64768 Len=0
134	01:11:02.054999	2600:1403:9c00:18::	2800:98:1617:51c8:2...	QUIC		Protected Payload (KP0)
135	01:11:02.084249	2800:98:1617:51c8:2...	2600:1901:81d4:200::	TLSv1.3	517	Client Hello (SNI=remote-mcp-510455560434.us-central1.run.app)
136	01:11:02.111145	2600:1901:81d4:200::	2800:98:1617:51c8:2...	TCP	0	443 → 51480 [ACK] Seq=1 Ack=518 Win=268800 Len=0

Frame 134: 86 bytes on wire (688 bits), 86 bytes captured (688 bits)

Ethernet II, Src: HuaweiTechno_ad:a4:ab (58:56:c2:ad:a4:ab), Dst: AzureWaveTec_33:de:ff (a8:41:f4:33:de:ff)

Internet Protocol Version 6, Src: 2600:1403:9c00:18::1732:70de, Dst: 2800:98:1617:51c8:2c5b:477:9ce0:6851

User Datagram Protocol, Src Port: 443, Dst Port: 55061

QUIC IETF

QUIC Connection information

[Connection Number: 0]

[Packet Length: 24]

QUIC Short Header

0... = Header Form: Short Header (0)

1... = Fixed Bit: True

..0... = Spin Bit: False

Remaining Payload: 864f74a88cc1dd47093dc3a3b747caa38c36cb38414349

Figuras 4–5 – Tramas equivalentes a initialize (sincronización) y tools/call (solicitud/respuesta) en TLS/QUIC.

8.3 Clasificación JSON-RPC

- **Sincronización:** initialize (primer POST a /mcp).
- **Solicitud:** tools/call (p.ej., time_now, add).
- **Respuesta:** JSON con result/error del servidor.

8.4 Explicación por capas

- **Enlace:** tramas Ethernet/802.11 entre NIC y AP.
- **Red (IP):** IPv6 pública.
- **Transporte:** TCP/443 (3-way handshake, TLS 1.3) y/o QUIC/UDP/443 (Protected Payload).
- **Aplicación:** HTTP/2 o HTTP/3 sobre TLS 1.3; cuerpo con JSON-RPC (initialize / tools/call / result).

9. Conclusiones

- La arquitectura MCP modular permite que la LLM use datos locales y facilita agregar servidores locales/remotos.
- Se validó un servidor remoto en Cloud Run y su consumo desde el host.
- El modo planificador permitió que la LLM use datos locales sin acceso directo a archivos
- Las capturas Wireshark permiten clasificar sincronización/solicitud/respuesta y explicar el tráfico por capas.
- Al hablar JSON-RPC estándar (por STDIO y HTTP), el host pudo integrar tanto un servidor local como uno remoto sin cambios drásticos en la capa de aplicación.

10. Comentarios

- La migración en la obtención de datos de un archivo JSON a una API, como la de SKF se contempló, por plazos de validación con la empresa no se pudo implementar más que con parte de la información pública.
- Es un reto la implementación de los servidores oficiales de Anthropic. La documentación es extensa, confusa en algunos momentos, en mi caso, lo atribuyo a la falta de conocimiento del área. Mucha información nueva en mi caso.
- El proyecto es atractivo y explora un poco el mundo de los estándares digitales. Es fascinante el simple hecho de imaginar cómo está todo interconectado y cómo hacen para poder tener interoperabilidad cuando a veces no la hay.

11. Referencias

- JSON-RPC — <https://www.jsonrpc.org/>
- MCP Architecture — <https://modelcontextprotocol.io/docs/learn/architecture>
- Cloud Run tutorial — <https://cloud.google.com/blog/topics/developers-practitioners/build-and-deploy-a-remote-mcp-server-to-google-cloud-run-in-under-10-minutes>
- **Professor de Redes de Computadores (GPT-5 Thinking)**. (2025). Asistencia conversacional durante el desarrollo del proyecto MCP (sesiones de chat). OpenAI.