Análise: Arquitetura MCP - Comparação py-xiaozhi vs ESP32

Resumo Executivo

py-xiaozhi-main:

- ☑ Cliente completo com câmera
- ☑ 32 ferramentas MCP registradas
- ☑ `take_photo` funciona
- ⚠ Integração com LLM remoto (problema potencial)

xiaozhi-esp32-server:

- ☑ Servidor MCP com sistema de plugins
- ☑ Processa chamadas de funções via MCP
- ❌ SEM implementação nativa de câmera (é client-side)
- ☑ Comunica via WebSocket com clientes

---

1. Arquitetura de Ferramentas

py-xiaozhi-main (Cliente)

```
McpServer.add_common_tools()
├── System Tools (4 tools)
│   ├── audio_speaker.set_volume
│   ├── audio_speaker.get_volume
│   └── ...
├── Calendar Tools (6 tools)
│   ├── create_event
│   ├── get_events
│   └── ...
├── Timer Tools (3 tools)
├── Music Tools (7 tools)
├── Camera Tools (2 tools) ← AQUI ESTÁ!
│   ├── take_photo()          ← Use para capturar imagens
│   └── take_screenshot()     ← Use para telas
├── Bazi Tools (6 tools)
└── ... [32 total]
```

Arquivo: `src/mcp/mcp_server.py` (linhas 249-350)

```
# Registro da câmera
from src.mcp.tools.camera import take_photo
```

```
properties = PropertyList([
    Property("question", PropertyType.STRING),
    Property("context", PropertyType.STRING, default_value="")
])

self.add_tool(
    McpTool(
        "take_photo",  # Nome da ferramenta
        VISION_DESC,   # Descrição para o LLM
        properties,    # Parâmetros esperados
        take_photo,    # Função que será executada
    )
)
```

xiaozhi-esp32-server (Servidor)

```
plugins_func/
└── functions/
    ├── change_role.py
    ├── get_news_from_chinanews.py
    ├── get_time.py
    ├── get_weather.py
    ├── handle_exit_intent.py
    ├── hass_get_state.py
    ├── hass_init.py
    ├── hass_play_music.py
    ├── hass_set_state.py
    ├── play_music.py
    ├── search_from_ragflow.py
    └── [NÃO TEM CÂMERA]
```

Problema: O servidor não implementa câmera porque é função do **cliente**!

---

2. Fluxo de Execução

Cenário: Usuário pede "Tire uma foto"

```
┌──────────────────────────────────────────────────────────┐
│  CLIENTE (py-xiaozhi-main)                               │
│                                                          │
│  1. Microfone captura: "Tire uma foto"                   │
│  2. ASR transcreve para texto                            │
│  3. LLM processa e identifica: preciso chamar take_photo()  │
│  4. MCP envia: {"method":"tools/call", "name":"take_photo"} │
│                                                          │
│  5. MCPServer._handle_tool_call()                        │
```

```
│     ├── Encontra ferramenta 'take_photo'                    │
│     ├── Chama: take_photo(arguments)                        │
│     ├── Câmera captura imagem                               │
│     ├── Envia para Vision API externa                       │
│     └── Retorna resultado ao LLM                            │
│                                                             │
│  6. LLM analisa resultado e responde                        │
│  7. TTS converte resposta para áudio                        │
│  8. Reproduz áudio no alto-falante                          │
└─────────────────────────────────────────────────────────────┘
```

---

3. Comparação Técnica

Aspecto	py-xiaozhi-main	xiaozhi-esp32-server
Linguagem	Python 3.13	Python + Java + Vue
Hardware de Câmera	☑ OpenCV	❌ Não nativo
Sistema de Plugins	Estático (add_common_tools)	Dinâmico (@register_function)
Ferramentas MCP	32 registradas	~12 via decoradores
Visão de Imagem	☑ Integrada com Zhipu AI	❌ Não implementada
WebSocket	☑ Cliente para servidor	☑ Servidor para clientes
Banco de Dados	❌ Não tem	☑ MySQL + Redis
Interface Web	❌ CLI/GUI local	☑ Vue.js em 8001

---

4. Onde Está o Código

Câmera (Cliente)

Localização: `src/mcp/tools/camera/`

```
camera/
├── __init__.py          ← Importa take_photo()
├── base_camera.py       ← Classe base com set_explain_url/set_explain_token
├── camera.py            ← TAKE_PHOTO() IMPLEMENTADA AQUI
├── vl_camera.py         ← Integração Zhipu AI
└── README.md
```

take_photo() - Arquivo: `src/mcp/tools/camera/camera.py` **(linhas 175-188)**

```python
async def take_photo(arguments: dict) -> str:
    """
```

```
    Captura foto e analisa com Vision API

    Args:
        arguments: {
            "question": "O que está na foto?",
            "context": "Detalhes adicionais (opcional)"
        }

    Returns:
        JSON com análise da imagem
    """
    camera = get_camera_instance()
    return camera.analyze(question)  # Chama Vision API
```

Registro MCP (Cliente)

Localização: `src/mcp/mcp_server.py` (linhas 282-318)

```
def add_common_tools(self):
    # ... outras ferramentas ...

    from src.mcp.tools.camera import take_photo

    properties = PropertyList([
        Property("question", PropertyType.STRING),
        Property("context", PropertyType.STRING, default_value="")
    ])

    self.add_tool(
        McpTool(
            "take_photo",
            VISION_DESC,
            properties,
            take_photo,  # ← Referência à função Python
        )
    )
```

Processamento MCP (Cliente)

Localização: `src/plugins/mcp.py` (linhas 35-37)

```
async def setup(self, app: Any) -> None:
    self._server = McpServer.get_instance()
    self._server.add_common_tools()  # ← REGISTRA FERRAMENTAS AQUI
```

Comunicação com Servidor

Localização: `src/protocols/` (websocket_protocol.py ou mqtt_protocol.py)

```
Cliente envia via MCP:
{
    "jsonrpc": "2.0",
    "id": 123,
    "method": "tools/call",
    "params": {
        "name": "take_photo",
        "arguments": {
            "question": "O que é isso?"
        }
    }
}

Servidor processa:
├── Identifica que é MCP
├── Processa via MCPServer
├── Executa take_photo()
├── Retorna resultado
└── Envia resposta via WebSocket
```

---

5. O Que Falta

Problema Identificado

A câmera funciona, as ferramentas estão registradas, MAS há uma questão de **sincronização**:

```
[TIMELINE DO PROBLEMA]

T0:   Cliente conecta
T1:   Servidor envia "initialize"
T2:   Cliente responde ao "initialize"
T3:   ⚠ NESTE PONTO - tools/list NÃO foi enviado ainda
T4:   LLM faz primeira requisição
T5:   LLM não sabe que "take_photo" existe
T6:   LLM nunca consegue chamar a função
```

Solução

```
# Em src/plugins/mcp.py, melhorar o setup():

async def setup(self, app: Any) -> None:
    self.app = app
    self._server = McpServer.get_instance()
```

```
    # 1. Registrar callback PRIMEIRO
    async def _send(msg: str):
        # ... enviar mensagem ...

    self._server.set_send_callback(_send)

    # 2. Registrar ferramentas IMEDIATAMENTE
    self._server.add_common_tools()

    # 3. LOG PARA VERIFICAR
    logger.info(f"[MCP] Ferramentas registradas: {len(self._server.tools)}")
    logger.info(f"[MCP] Camera tool disponível: {'take_photo' in [t.name for t
in self._server.tools]}")
```

---

6. Recomendações

☑ O que Está Correto

- Implementação da câmera é sólida
- Registro das ferramentas MCP é correto
- Integração com Vision API está bem feita
- Token e URL são configurados corretamente

⚠ O que Pode Melhorar

1. **Adicionar validação de inicialização:**

```
if len(self._server.tools) == 0:
    logger.error("[MCP] Nenhuma ferramenta foi registrada!")
```

2. **Implementar garantia de sincronização:**

```
# Aguardar que ferramentas sejam registradas antes de aceitar requisições
```

3. **Melhorar logging de debug:**

```
logger.debug(f"[MCP TOOLS] Registradas: {[t.name for t in
self._server.tools]}")
```

4. **Teste automático na inicialização:**

```
# Verificar que take_photo está disponível
assert any(t.name == "take_photo" for t in self._server.tools)
```

Conclusão

A câmera está completamente implementada e pronta para uso.

O sistema espera que o LLM remoto:

1. Receba a lista de ferramentas MCP
2. Identifique que `take_photo` está disponível
3. Chame a ferramenta quando apropriado
4. Processe o resultado

Se a câmera ainda não funciona ao falar com o LLM, verifique:

- Se as ferramentas foram registradas (☑ Confirmado)
- Se a lista de ferramentas é enviada ao LLM
- Se o LLM sabe como usar a ferramenta
- Se há erros de rede entre cliente e servidor