Guia Prático: Ativar Câmera via MCP

TL;DR (Resumo Executivo)

A câmera **JÁ ESTÁ IMPLEMENTADA** e as 32 ferramentas MCP estão **REGISTRADAS COM SUCESSO**.

O problema é uma **questão de sincronização**: o LLM remoto precisa saber que `take_photo` está disponível ANTES de poder usá-la.

Verificação Rápida

1. Confirmar que ferramentas estão registradas

```
python test_mcp_fix.py
```

Resultado esperado:

```
[OK] Tools after add_common_tools(): 32
  - take_photo ☑ PRESENT
```

☑ Se ver isso, as ferramentas estão OK.

2. Verificar estrutura de arquivos

```
# Verificar que camera existe
ls -la src/mcp/tools/camera/

# Saída esperada:
# -rw-r--r--  base_camera.py
# -rw-r--r--  camera.py          ← TAKE_PHOTO AQUI
# -rw-r--r--  vl_camera.py       ← ZHIPU AI
# -rw-r--r--  __init__.py
```

3. Verificar visão está configurada

```
# Ver se Vision API está setada
grep -r "explain_url\|explain_token" src/mcp/tools/camera/
```

Deve retornar:

- `set_explain_url()`
- `set_explain_token()`

☑ Se ver isso, Vision API está pronta.

Solução Passo-a-Passo

Passo 1: Melhorar Logging

Arquivo: `src/plugins/mcp.py`

Buscar: Linha ~35

```
async def setup(self, app: Any) -> None:
    # ...
    try:
        self._server.add_common_tools()
    except Exception:
        pass  # ← PROBLEMA: silencia erros!
```

Substituir por:

```
async def setup(self, app: Any) -> None:
    # ...
    try:
        self._server.add_common_tools()
        logger.info(f"[MCP] Ferramentas registradas:
{len(self._server.tools)}")
        camera_available = any(
            t.name == "take_photo" for t in self._server.tools
        )
        logger.info(f"[MCP] Camera tool: {'DISPONIVEL' if camera_available else
'FALTA'}")
    except Exception as e:
        logger.error(f"[MCP] Erro ao registrar ferramentas: {e}")
        import traceback
        traceback.print_exc()
```

Passo 2: Adicionar Debug ao MCP Server

Arquivo: `src/mcp/mcp_server.py`

Buscar: Método `_handle_initialize` (linha ~418)

Adicionar ao final do método:

```
# LOG PARA DEBUG
logger.info(f"[MCP INIT] Tools count: {len(self.tools)}")
logger.info(f"[MCP INIT] Available tools:")
for tool in self.tools[:5]:
    logger.info(f"  - {tool.name}")
if len(self.tools) > 5:
    logger.info(f"  ... and {len(self.tools) - 5} more")
```

Buscar: Método `_handle_tools_list` (linha ~471)

Verificar que tem:

```
logger.info(f"[MCP TOOLS/LIST] Total de tools registradas: {len(self.tools)}")
logger.info(f"[MCP TOOLS/LIST] Tools disponíveis:")
for tool in self.tools:
    logger.info(f"  - {tool.name}")
```

Se não tiver, ADICIONAR (já foi feito na sessão anterior).

Passo 3: Garantir Inicialização Atômica

Arquivo: `src/application.py`

Buscar: Onde `plugins.setup_all()` é chamado (linha ~122)

Adicionar depois:

```
# Garantir que MCP foi inicializado
try:
    from src.plugins.mcp import McpPlugin
    mcp_plugin = self.plugins.get("mcp")
    if mcp_plugin and hasattr(mcp_plugin, "_server"):
        tools_count = len(mcp_plugin._server.tools)
        logger.info(f"[APP] MCP iniciado com {tools_count} ferramentas")
except Exception as e:
    logger.warning(f"[APP] Não foi possível verificar MCP: {e}")
```

Passo 4: Executar e Verificar Logs

```
# Executar com logging verbose
LOGLEVEL=DEBUG python main.py --mode gui --protocol websocket 2>&1 | grep -i
"mcp\|camera"
```

Procure por:

```
[MCP] Ferramentas registradas: 32
[MCP] Camera tool: DISPONIVEL
[MCP INIT] Tools count: 32
[MCP TOOLS/LIST] Total de tools registradas: 32
[MCP TOOLS/LIST] Tools disponíveis:
  - take_photo
```

Se ver tudo isso, a inicialização está correta. ☑

Passo 5: Testar Chamada Direta

```
# Criar arquivo test_camera_direct.py
cat > test_camera_direct.py << 'EOF'
import asyncio
from src.mcp.tools.camera import take_photo

async def test():
    result = await take_photo({
        "question": "O que está vendo?",
        "context": "Teste direto"
    })
    print("Resultado:", result)

asyncio.run(test())
EOF

# Executar
python test_camera_direct.py
```

Se funcionar: ☑ A câmera está OK

Se der erro:

- Verifique se a câmera está conectada
- Verifique token da Vision API
- Verifique URL da Vision API

Passo 6: Testar Integração LLM

```
# Iniciar aplicação
python main.py --mode cli --protocol websocket

# No chat, diga:
# "Tire uma foto"
# "O que está na câmera?"
# "Faça uma captura de tela"
```

O que deve acontecer:

1. LLM recebe comando
2. LLM identifica que precisa de `take_photo`
3. MCP Server processa a chamada
4. Câmera captura imagem
5. Vision API analisa
6. LLM responde com análise

Se não funcionar:

- Verifique logs para erros MCP
- Verifique se ferramentas foram registradas
- Verifique autenticação do LLM

Troubleshooting

Problema: "Camera tool: FALTA"

Causa: `add_common_tools()` não foi chamado ou falhou

Solução:

```
# Em src/plugins/mcp.py, adicionar print/log antes de registrar
logger.info("Antes de add_common_tools, tools count:", len(self._server.tools))
self._server.add_common_tools()
logger.info("Depois de add_common_tools, tools count:",
len(self._server.tools))
```

Problema: "LLM não está chamando take_photo"

Possíveis causas:

1. LLM não recebeu a lista de ferramentas
    - Solução: Verificar que `tools/list` retorna as 32 ferramentas
2. LLM não sabe interpretar o resultado da câmera
    - Solução: Melhorar a descrição em `VISION_DESC`
3. Autenticação da Vision API falhou
    - Solução: Verificar token e URL em logs

Debug:

```
# Adicionar a esto em _handle_tool_call
logger.info(f"[MCP] Chamando ferramenta: {tool_name}")
logger.info(f"[MCP] Argumentos: {params}")
try:
```

```
        result = tool.callback(params.get("arguments", {}))
        logger.info(f"[MCP] Resultado: {result[:100]}...")  # Primeiros 100 chars
    except Exception as e:
        logger.error(f"[MCP] Erro ao executar: {e}", exc_info=True)
        raise
```

Problema: Vision API retorna erro

Verificar:

```
# Token configurado?
grep -r "Vision service token" logs/

# URL configurada?
grep -r "Vision service configured" logs/

# Token válido?
# Acessar: http://api.xiaozhi.me/vision/explain com o token
```

Checklist de Inicialização

- ☐ Câmera detectada (hardware OK)
- ☐ Vision API configurada (token + URL)
- ☐ Ferramentas registradas (32 no total)
- ☐ `take_photo` presente na lista
- ☐ MCP plugin inicializado
- ☐ Logs mostram sucesso
- ☐ LLM recebe lista de ferramentas
- ☐ Teste direto funciona
- ☐ Teste via LLM funciona

Se todos os itens estão ☑, a câmera deve funcionar!

Próximas Otimizações

1. Cache de Frames

Para evitar capturar várias fotos em seguida:

```
# Em src/mcp/tools/camera/camera.py
_last_frame_cache = None
_cache_timestamp = 0
_cache_ttl = 5  # segundos

async def take_photo(arguments):
```

```
    global _last_frame_cache, _cache_timestamp

    # Se tem cache recente, usar
    if _last_frame_cache and (time.time() - _cache_timestamp) < _cache_ttl:
        logger.info("[CAMERA] Using cached frame")
        return _last_frame_cache

    # Senão, capturar novo
    camera = get_camera_instance()
    result = camera.analyze(question)

    # Guardar em cache
    _last_frame_cache = result
    _cache_timestamp = time.time()

    return result
```

2. Melhorar Descrição da Ferramenta

Para o LLM saber quando usar `take_photo`:

```
VISION_DESC = """
【Ferramenta de Visão Computacional】
Use ESTA ferramenta quando o usuário pedir para:
- "Tire uma foto" / "Faça foto" / "Capture uma imagem"
- "O que está vendo?" / "Vê algo?" / "Analisa a câmera"
- "Mostre o que está na câmera"
- "Faça captura de tela" / "Screenshot"

NÃO use para:
- Buscar imagens na internet
- Modificar/editar fotos
- Salvar arquivos

Parâmetros:
- question (obrigatório): O que você quer saber sobre a imagem?
- context (opcional): Detalhes adicionais (ex: "Close-up, alta qualidade")

Exemplos:
- take_photo("Há alguém na câmera?")
- take_photo("Qual é este objeto?", "De perto, bem iluminado")
"""
```

Conclusão

A câmera **ESTÁ PRONTA PARA USO**.

Siga o **Passo 1 a 4** para garantir que tudo está inicializado corretamente, depois teste com o **Passo 6**.

Se houver problemas, use o **Troubleshooting** para debug.

Para otimizações, veja **Próximas Otimizações**.

Good luck! 🎥✨