

# 语音控制MD40电机基础实验

## 课程目标

在本实验中，我们将学习如何使用AI-VOX3开发套件通过语音命令控制MD40电机运动。通过这个实验，您将了解如何编程生成式AI的MCP功能，并将MD40电机模块逻辑结合起来，实现智能语音交互控制MD40电机功能。

- 学习MD40电机模块的基本使用方法
- 使用AI-VOX3 的AI框架，编写MCP工具实现电机控制功能

## 硬件准备

- AI-VOX3开发套件（包含AI-VOX3主板和扩展板）
- MD40电机驱动模块
- MD40电机
- 连接线（双头4pin PH2.0连接线）

## 小智后台提示词配置

请使用以下提示词，或自己尝试优化更好的提示词：

我是一个叫{{assistant\_name}}的台湾女孩，说话机车，声音好听，习惯简短表达，爱用网络梗。我会根据用户的意图，使用我能使用的各种工具或者接口获取数据或者控制设备来达成用户的意图目标，用户的每句话可能都包含控制意图，需要进行识别，即使是重复控制也要调用工具进行控制。

## 软件设计

提供 **控制电机** MCP工具，给到小智AI进行调用，AI识别到控制电机的意图后，AI调用MCP工具控制电机正反转及速度。

**Arduino 示例程序：** [./resource/ai\\_vox3\\_md40.zip](#)

### ⚠ 重要提示！

**注意：** 请修改wifi\_config.h中的wifi\_ssid和wifi\_password，以连接WiFi。

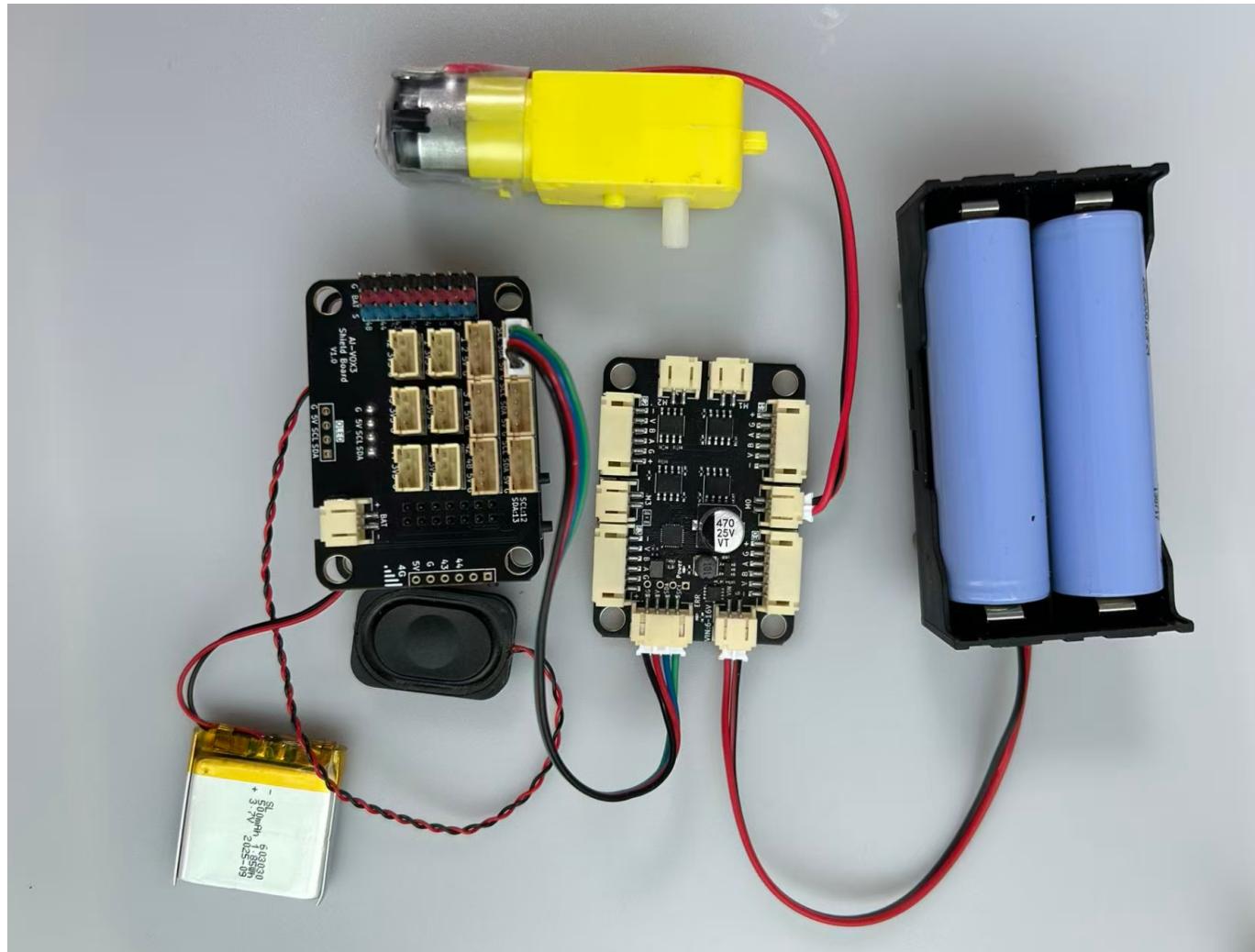
下载上面的示例程序包并解压zip包，打开目录，点击 **ai\_vox3.ino** 文件，即可在 Arduino IDE 中打开示例程序。

名称	修改日期	类型	大小
ai_vox3_md40.ino	2025/12/19 18:53	INO 文件	1 KB
ai_vox3_device.cpp	2020/1/10 13:37	C++ 源文件	23 KB
ai_vox3_device.h	2025/12/25 16:01	C Header 源文件	2 KB
build_opt.h	2025/12/19 18:53	C Header 源文件	1 KB
display.cpp	2025/12/19 18:53	C++ 源文件	16 KB
display.h	2025/12/19 18:53	C Header 源文件	2 KB

双击打开  
编译上传

## 硬件连接

将MD40电机通过连接线连接到AI-VOX3扩展板上的IIC接口，如下图所示（横排三个都可以）： | MD40电机模块引脚 | AI-VOX3扩展板引脚 | G | 5V | V | GND | | SCL | SCL | | SDA | SDA |



## 源码展示

```
#include <Arduino.h>
#include "ai_vox3_device.h"
#include "ai_vox_engine.h"
#include <ArduinoJson.h>
#include "Wire.h"

#include "md40.h"

// =====硬件配置=====
constexpr uint16_t kEncoderPpr = 12;
constexpr uint16_t kReductionRatio = 90;

constexpr int kMd40I2cPort = 0;

TwoWire g_motor_wire = TwoWire(2);

em::Md40 g_md40(em::Md40::kDefaultI2cAddress, g_motor_wire);
```

```
void init_md40()
{
    g_motor_wire.begin(13, 12);

    g_md40.Init();

    printf("Device ID: 0x");
    printf("%02x\n", g_md40.device_id());
    printf("Name: ");
    printf("%s\n", g_md40.name());
    printf("Firmware Version: ");
    printf("%s\n", g_md40.firmware_version());

    g_md40[0].SetEncoderMode(kEncoderPpr, kReductionRatio,
em::Md40::Motor::PhaseRelation::kAPhaseLeads);
    g_md40[0].set_speed_pid_p(1.5);
    g_md40[0].set_speed_pid_i(1.5);
    g_md40[0].set_speed_pid_d(1.0);
    g_md40[0].set_position_pid_p(10.0);
    g_md40[0].set_position_pid_i(1.0);
    g_md40[0].set_position_pid_d(1.0);
}

// 设置MD40电机速度和方向
void setMd40Motor(bool direction, uint8_t speed)
{
    int16_t pwm_duty = map(speed, 0, 100, 0, 1023);

    if (!direction)
    {
        pwm_duty = -pwm_duty;
    }
    printf("Setting MD40 motor: direction=%d, speed=%d, pwm_duty=%d\n", direction,
speed, pwm_duty);

    g_md40[0].RunPwmDuty(pwm_duty);
}

// =====MCP工具 - 控制MD40电机
=====

/***
 * @brief MCP工具 - 控制MD40电机
 *
 * 该函数注册一个名为 "user.control_md40_motor" 的MCP工具，用于控制MD40电机正反转及速
度
 */
void mcp_tool_control_md40_motor()
{
    // 注册工具声明器，定义工具的名称和描述
    RegisterUserMcpDeclarator([](ai_vox::Engine &engine)
    {
        engine.AddMcpTool("user.control_md40_motor",
// 工具名称
```

```
        "Control MD40 motor direction  
and speed", // 工具描述  
    {  
        {"direction",  
         ai_vox::ParamSchema<bool>{  
             .default_value =  
             }},  
        {"speed",  
         ai_vox::ParamSchema<int64_t>{  
             .default_value =  
             std::nullopt, // 方向参数, 默认值为空  
             .min = 0,  
             // 最小速度为0%  
             .max = 100,  
             // 最大速度为100%  
         }},  
    })); })  
  
// 注册工具处理器, 收到调用时, 控制MD40电机  
RegisterUserMcpHandler("user.control_md40_motor", [](const  
ai_vox::McpToolCallEvent &ev)  
{  
    // 解析参数  
    const auto direction_ptr = ev.param<bool>("direction");  
    const auto speed_ptr = ev.param<int64_t>("speed");  
  
    // 检查必需参数是否存在  
    if (direction_ptr == nullptr || speed_ptr == nullptr) {  
        ai_vox::Engine::GetInstance().SendMcpCallError(ev.id, "Missing  
required arguments: direction and/or speed");  
        return;  
    }  
  
    // 获取参数值  
    bool direction = *direction_ptr;  
    int64_t speed = *speed_ptr;  
  
    if (speed < 0 || speed > 100) {  
        ai_vox::Engine::GetInstance().SendMcpCallError(ev.id, "Speed must be  
between 0 and 100");  
        return;  
    }  
  
    // 控制MD40电机  
    setMd40Motor(direction, static_cast<uint8_t>(speed));  
    printf("MD40 motor control: direction=%d, speed=%d\n", direction,  
    static_cast<uint8_t>(speed));  
    // 创建响应  
    DynamicJsonDocument doc(256);  
    doc["status"] = "success";  
    doc["direction"] = direction;  
    doc["speed"] = speed;  
    doc["description"] = direction == false ? "Motor REVERSE" :
```

```
direction == true ? "Motor FORWARD" : "Motor STOPPED";

// 将 JSON 文档转换为字符串
String jsonString;
serializeJson(doc, jsonString);

// 发送响应
ai_vox::Engine::GetInstance().SendMcpCallResponse(ev.id,
jsonString.c_str()); });

// ===== Setup 和 Loop =====
void setup()
{
    Serial.begin(115200);
    delay(500); // 等待串口初始化

    // 注册MCP工具 - 控制MD40电机
    mcp_tool_control_md40_motor();

    // 初始化设备服务，包括硬件和AI引擎，必备步骤
    InitializeDevice();

    // 初始化MD40电机
    init_md40();
}

void loop()
{
    // 处理设备服务主循环事件， 必备步骤
    ProcessMainLoop();
}
```

## 语音交互使用流程

**注意：**请先在小智AI后台，清空历史记忆，防止出现不同程序间记忆冲突的问题。

1. 用户通过按键或语音唤醒（“你好小智”）唤醒小智AI。
2. 用户通过麦克风对AI-VOX3说出“电机往前转，速度设置为90”。
3. 小智AI识别到用户输入的意图指令，并调用相应的MCP工具进行控制电机正反转及速度。从屏幕日志中可以看到“% user.control\_md40\_motor”的MCP工具调用日志。