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。

图 1: alt text

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"

// ========================================  ========================================
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);
}

//  MD40
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  -  MD40 ========================================

/**
 * @brief MCP  -  MD40
 *
 *       "user.control_md40_motor"  MCP    MD40
 */
void mcp_tool_control_md40_motor()
{
    //
    RegisterUserMcpDeclarator([](ai_vox::Engine &engine)
                              { engine.AddMcpTool("user.control_md40_motor",                //
                                                  "Control MD40 motor direction and speed", //
                                                  {
                                                      {"direction",
                                                       ai_vox::ParamSchema<bool>{
                                                           .default_value = std::nullopt, //
                                                       }},
                                                      {"speed",
                                                       ai_vox::ParamSchema<int64_t>{
                                                           .default_value = std::nullopt, //
                                                           .min = 0,                      //    0%
                                                           .max = 100,                    //    100%
                                                       }}}); });

    //          MD40
    RegisterUserMcpHandler("user.control_md40_motor", [](const ai_vox::McpToolCallEvent &ev)
                           {
        //
        const auto direction_ptr = ev.param<bool>("direction");
        const auto speed_ptr = ev.param<int64_t>("speed");

        //
        if (direction_ptr == nullptr || speed_ptr == nullptr) {
            ai_vox::Engine::GetInstance().SendMcpCallError(ev.id, "Missing required arguments: direction
            return;
        }

        //
        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;
        }

        //  MD40
        setMd40Motor(direction, static_cast<uint8_t>(speed));
        printf("MD40 motor control: direction=%d, speed=%d\n", direction, static_cast<uint8_t>(speed));
```

```
        //
        DynamicJsonDocument doc(256);
        doc["status"] = "success";
        doc["direction"] = direction;
        doc["speed"] = speed;
        doc["description"] = direction == false ? "Motor REVERSE" :
                             direction == true ? "Motor FORWARD" : "Motor STOPPED";

        //  JSON
        String jsonString;
        serializeJson(doc, jsonString);

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

// ========== Setup  Loop ==========
void setup()
{
    Serial.begin(115200);
    delay(500); //

    //  MCP  -  MD40
    mcp_tool_control_md40_motor();

    //       AI
    InitializeDevice();

    //  MD40
    init_md40();
}

void loop()
{
    //
    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。