

K230 Road Sign Instruction Actions (Voice)

K230 Road Sign Instruction Actions (Voice)

- I. Software-Hardware
- II. Brief Principles
 - 1. Hardware Schematic Diagram
 - 2. Physical Connection Diagram
 - 3. Control Principle
- III. Code Analysis
- IV. Voice Module Commands
- V. Experimental Operation
- VI. Experimental Phenomena
- Road sign images

This tutorial is a comprehensive experiment combining multiple peripherals. It's recommended to familiarize yourself with individual peripherals before proceeding with this experiment.

I. Software-Hardware

- KEIL5
- MSPM0G3507 Robot Development Board

k230 Module: External

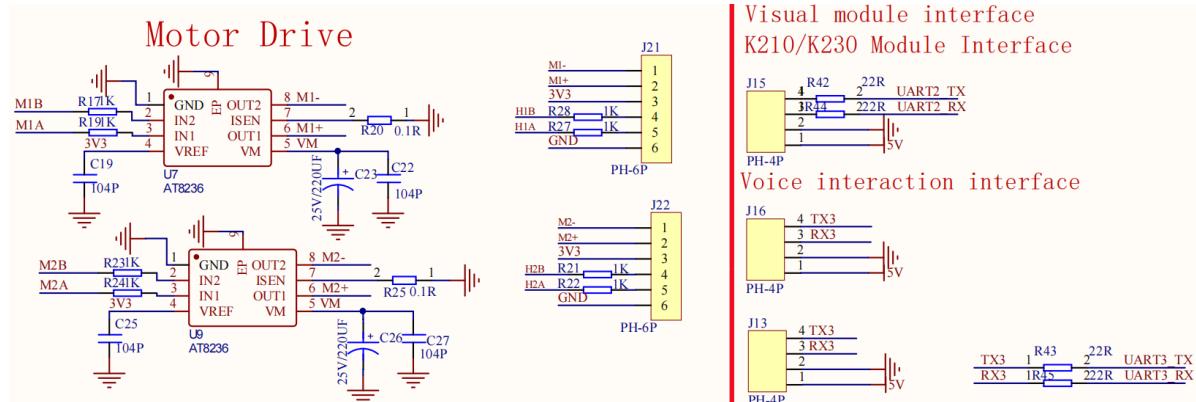
AI Voice Interaction Integration Module: External

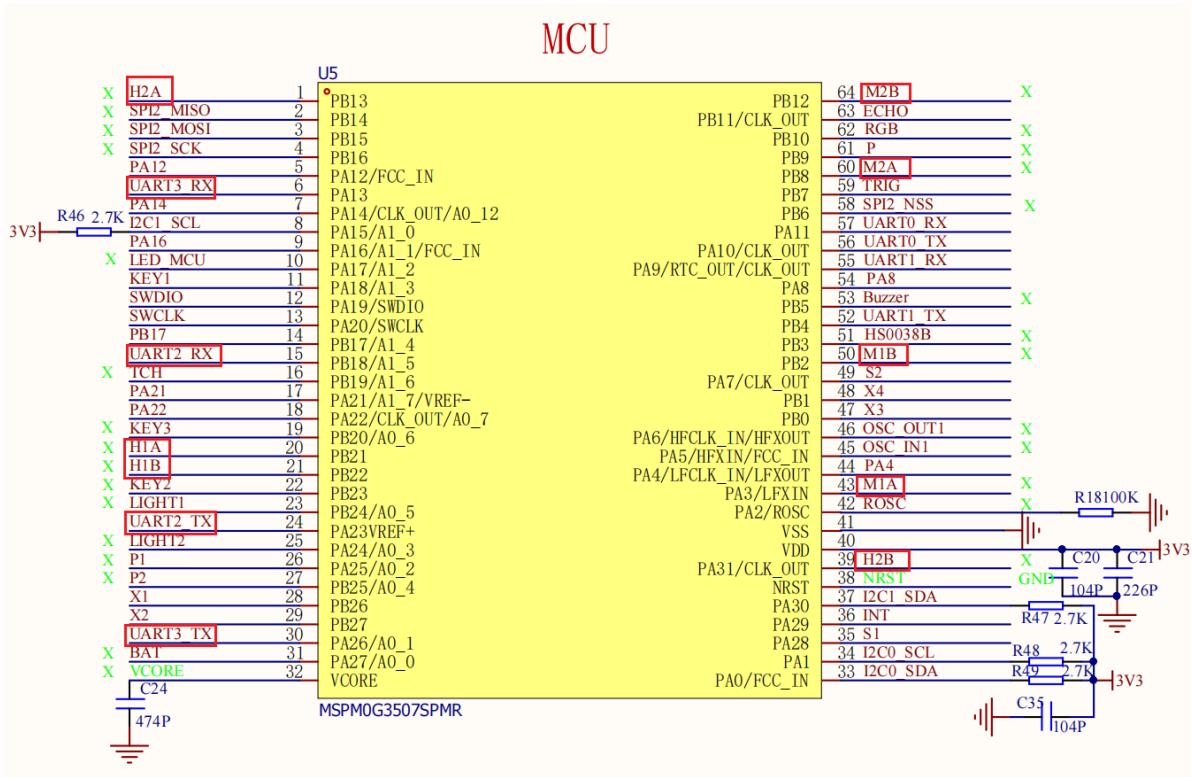
- Type-C Data Cable or DAP-Link

Download or simulate the program on the development board

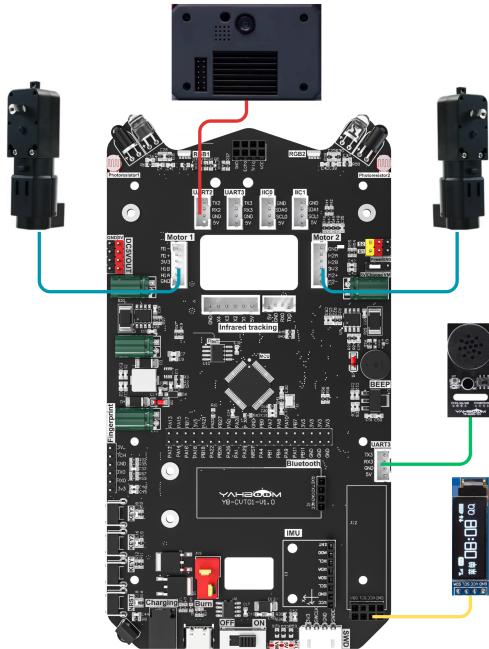
II. Brief Principles

1. Hardware Schematic Diagram





2. Physical Connection Diagram



Wire Connection Pins

Motor Wiring (Note: The wiring diagram below is for reference only. Our products come with a double-ended PH2.0 6-pin all-black cable with a foolproof design, so there's no need to worry about wiring issues.)

TT Encoder Motor	MSPM0G3507
M-	M1 -
M+	M1 +
VVC	3V3
Encoder B	H1B
EncoderA	H1A
GND	GND

TT encoder motor	MSPM0G3507
GND	GND
Encoder A	H2A
Encoder B	H2B
VCC	3V3
M+	M2 +
M -	M2 -

Voice module wiring (Note: The wiring diagram below is for reference only. Our products come with a dual-ended PH2.0 4-pin all-black voice module cable with a foolproof design; no wiring issues are necessary.)

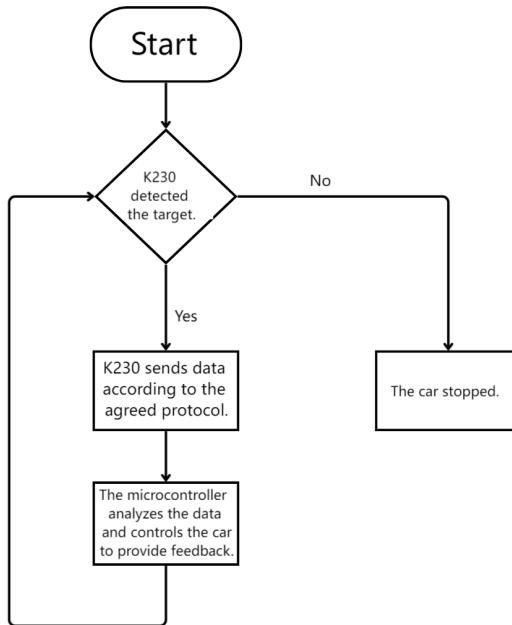
Intelligent voice interaction module	MSPM0G3507
RX1	TX3
TX1	RX3
GND	GND
5V	5V

K230 Wiring (Note: The wiring diagram below is for reference only. Our products come with a K230 double-ended PH2.0 4-pin all-black cable with a foolproof design; you don't need to worry about wiring issues.)

K230	MSPM0G3507
RX	TX2
TX	RX2
GND	GND
5V	5V

3. Control Principle

- Program Flowchart



Control Principle:

PID Control

PID control is a commonly used closed-loop control algorithm. It continuously adjusts the controller's output to minimize the error between the system's actual output and the desired output.

PID Parameters	Function	Purpose
P (Proportional)	Determines the response speed of the PID controller output	Speeds up system response
I (Integral)	Determines the PID controller's response to accumulated errors	Eliminates steady-state errors
D (Differential)	Determines the PID controller's response to the rate of change of error	Improves the system's steady-state performance

Module	Function
k230 Vision Module	Select the road sign you want to recognize, and then send the data to the microcontroller via serial port

Communication Protocol:

After the k230 recognizes a road sign, it will send a data frame via serial port, in the following format:

Experimental Routine	Start Character	Length	Separator	Routine Number	Separator	x	Separator	y	Separator	w	Separator	h	End Character
Color Recognition	\$	XX	,	16	,	XXX	,	XXX	,	XXX	,	XXX	#

Microcontroller Section:

Receives data sent by the K230 via serial port and processes each byte of data. Extracts valid data from the data frame content and stores it in an array. Then, the array data is processed using PID control, and the microcontroller uses the processed data to drive the motor.

III. Code Analysis

Main Function Analysis

Function: Pto_Loop

Function Prototype	void Pto_Loop(void)
Function Description	The main function for running the program
Input Parameters	None
Output Parameters	None

Partial Function Analysis

- main.c

```
colorMode color_mode = 1; //1: Patrol, 2: Tracking

int main(void)
{
    bsp_init()// Required peripheral initialization
    Control_RGB_ALL(OFF);//closure RGB
    delay_ms(100);
    OLED_ShowString(0,0,"pid_output:",8,1);//The PID output is used to control left
    and right.
    OLED_ShowString(0,20,"pid_output1:",8,1);//The PID output is displayed to
    control the front and rear.
    OLED_Refresh();

    while (1)
    {
        Pto_Loop();
        OLED_ShowSNum(75,0, pid_output, 3, 8, 1);
        OLED_ShowSNum(75,20,pid_output1,3,8,1);
        OLED_Refresh();
    }
}
```

BSP_init: Hardware initialization

pto_Loop: Continuously processes received K230 messages. - yb_protocol.c

- yb_protocol.c

```

/**
 * @Brief: Data analysis
 * @Note:
 * @Parm: Pass in a received data frame and length
 * @Retval:
 */
void Pto_Data_Parse(uint8_t *data_buf, uint8_t num)
{
    uint8_t pto_head = data_buf[0];
    uint8_t pto_tail = data_buf[num-1];

    // Check head and tail
    if (!(pto_head == PTO_HEAD && pto_tail == PTO_TAIL))
    {
        //DEBUG_PRINT("pto error:pto_head=0x%02x , pto_tail=0x%02x\n", pto_head,
pto_tail);
        return;
    }

    uint8_t data_index = 1;
    uint8_t field_index[PTO_BUF_LEN_MAX] = {0};
    int i = 0;
    int values[PTO_BUF_LEN_MAX] = {0};
    char msgs[] [PTO_BUF_LEN_MAX] = {0};

    // Split Field
    for (i = 1; i < num-1; i++)
    {
        if (data_buf[i] == ',')
        {
            data_buf[i] = 0;
            field_index[data_index] = i;
            data_index++;
        }
    }

    // Parsing length and function ID
    for (i = 0; i < 2; i++)
    {
        values[i] = Pto_Char_To_Int((char*)data_buf+field_index[i]+1);
    }

    uint8_t pto_len = values[0];
    uint8_t pto_id = values[1];

    ParseCommonFields(pto_id,data_buf,pto_len,field_index,data_index,values,msgs);

}

```

The received data conforming to the protocol is parsed to obtain the function ID number and the data information carried in the data, and the corresponding function processing function is executed. This project uses metadata parsing. The core idea is to separate the description of the protocol format (field position, type, etc.) from the parsing logic, define the protocol format

through data structures (such as structure arrays), and the parsing function automatically processes the data according to these descriptions.

Data of different types in the protocol can also be parsed. You can study the implementation details yourself if you want to know more. I will not explain too much. This function has been adapted to all K230 communication protocols in this store. You can use it normally without fully understanding it.

- bsp_PID_motor.c

```
// PID calculates one channel separately
float PID_Calc_One_Motor(uint8_t motor_id, float now_speed)
{
    if (motor_id >= MAX_MOTOR)
        return 0;
    return PID_Location_Calc(&pid_motor[motor_id], now_speed);
}

void Set_PID_Motor(float set_l ,float set_r,float turn_out)
{
    l_pid_out = PID_Calc_One_Motor(0, set_l);
    r_pid_out = PID_Calc_One_Motor(1, set_r);
    PWM_Control_Car(l_pid_out+turn_out , r_pid_out-turn_out );
}

//Position PID calculation method
float PID_Location_Calc(PID_t *pid, float actual_val)
{
    /*Calculate the error between the target value and the actual value*/
    pid->err = pid->target_val - actual_val;

    /*Limited closed-loop dead zone*/
    if ((pid->err >= -40) && (pid->err <= 40))
    {
        pid->err = 0;
        pid->integral = 0;
    }

    /*Integral separation, removing the integral effect when the deviation is
    large*/
    if (pid->err > -1500 && pid->err < 1500)
    {
        pid->integral += pid->err; // error accumulation

        /* Limit the integration range to prevent integration saturation */
        if (pid->integral > 4000)
            pid->integral = 4000;
        else if (pid->integral < -4000)
            pid->integral = -4000;
    }
}
```

```

/*PID algorithm implementation*/
pid->output_val = pid->Kp * pid->err +
                    pid->Ki * pid->integral +
                    pid->Kd * (pid->err - pid->err_last);

/*Error transmission*/
pid->err_last = pid->err;

/*Returns the current actual value*/
return pid->output_val;
}

```

Set_PID_Motor: Sets the target values for the two motors and the steering loop output value.

PID_Calc_One_Motor: Calculates the position loop PID value for one channel and returns the calculated value.

PID_Location_Calc: The formula for calculating the position loop PID and the complete position loop structure.

- bsp_K230_control.c

```

void RoadSign_Rec(const char* msg)
{
    if(strcmp(msg,"go_straight")==0)
    {
        write_Data(0x04);

//        DEBUG_PRINT("go straight!\r\n");
        Set_PID_Motor(70,70,0);
        delay_ms(1000);
        delay_ms(1000);
        Motor_Stop(1) ;
    }
    else if(strcmp(msg,"turn_right")==0)
    {

        write_Data(0x07);

        Set_PID_Motor(70,70,80);
        delay_ms(1000);
        Motor_Stop(1) ;
//        delay_ms(1000);
    }
    else if (strcmp(msg, "speed_limit")==0)
    {
        write_Data(0x02);

        Set_PID_Motor(50,50,0);
        delay_ms(1000);
        delay_ms(1000);
        Motor_Stop(1) ;
    }
}

```

```
else if (strcmp(msg, "stop")==0)
{
    Write_Data(0x01);

    Set_PID_Motor(0,0,0);
    delay_ms(1000);
    delay_ms(1000);
    Motor_Stop(1) ;
}

else {
    Motor_Stop(1) ;

}

}
```

RoadSign_Rec: After recognizing road signs via K230, the car is controlled to move based on the recognized road signs.

IV. Voice Module Commands

命令词 (Command Word)	功能类型 (Function Type)	播报语句 (Broadcast Statement)	播报模式 (Broadcast Mode)	发送协议 (Send Protocol)	接收协议 (Receive Protocol)
WELCOME	欢迎语 (Welcome)	welcome	被 (Passive)	AA 55 01 00 FB	AA 55 01 00 FB
BYE	休息语 (Rest)	bye	主 (Active)	AA 55 02 6F FB	AA 55 02 00 FB
HI-YAHBOOM	唤醒词 (Wake Word)	I am here	主 (Active)	AA 55 03 00 FB	AA 55 03 00 FB
VOLUME-UP	增大音量 (Increase Vol)	Volume Up	主 (Active)	AA 55 04 00 FB	AA 55 04 00 FB
VOLUME-DOWN	减小音量 (Decrease Vol)	Volume down	主 (Active)	AA 55 05 00 FB	AA 55 05 00 FB
MAX-VOLUME	最大音量 (Max Volume)	Maximum volume	主 (Active)	AA 55 06 00 FB	AA 55 06 00 FB
MID-VOLUME	中等音量 (Medium Volume)	Medium volume	主 (Active)	AA 55 07 00 FB	AA 55 07 00 FB
MINI-VOLUME	最小音量 (Min Volume)	Minimum Volume	主 (Active)	AA 55 08 00 FB	AA 55 08 00 FB
START-REPORT	开播报 (Enable Broadcast)	Start broadcasting	主 (Active)	AA 55 09 00 FB	AA 55 09 00 FB
STOP-REPORT	关播报 (Disable Broadcast)	Stop Broadcast	主 (Active)	AA 55 0A 00 FB	AA 55 0A 00 FB
GO-AHEAD	命令词 (Command)	The car moves forward after recognizing the road sign	主 (Active)	AA 55 00 04 FB	AA 55 00 04 FB
TURN-RIGHT	命令词 (Command)	The car turns right upon detecting the turn sign	主 (Active)	AA 55 00 07 FB	AA 55 00 07 FB
SPEED-REDUCTION-SIGN	命令词 (Command)	The car slows down upon detecting the speed limit sign	主 (Active)	AA 55 00 02 FB	AA 55 00 02 FB

命令词 (Command Word)	功能类型 (Function Type)	播报语句 (Broadcast Statement)	播报模式 (Broadcast Mode)	发送协议 (Send Protocol)	接收协议 (Receive Protocol)
STOP	命令词 (Command)	The car stopped upon detecting the stop sign	主 (Active)	AA 55 00 01 FB	AA 55 00 01 FB

V. Experimental Operation

1. Connect the hardware according to the hardware wiring instructions in the tutorial.
2. Open the project used by MSPM0G3507, then compile and flash it, and finally reset or power on again.
3. Place RoadSign_Rec.py in the root directory of the K230's SD card and rename it to main.py, then reset or power on again. If you are unsure how to flash the program onto the K230, please refer to the K230 quick start tutorial.
4. Place the car in an open area, adjust the K230 module bracket to a suitable angle, and connect the power supply.
5. After the K230 initializes successfully, it will begin recognizing the trained road sign model. Press the car reset button, wait three seconds, and the car will begin to respond to the recognized model.

Note: The voice module requires firmware flashing in advance.

Voice Module Firmware Flashing

Simply burn the CI1302_EN_Single_v00916_UART1_115200_2M(MSPM0).bin file we provide to implement all the related cases of the intelligent voice interaction module for this car. For firmware burning, please refer to [Chapter 8: Extended application course (Sold separately) → 1.Voice interaction module communication]. If it has been burned before, it does not need to be burned again.

VI. Experimental Phenomena

Turn on the power switch, wait for system initialization to complete, the screen will display the camera feed, start recognizing road signs, and perform corresponding actions. **Note: When connecting our AI voice interaction integration module, there will be voice broadcast after recognizing road signs! Users who have not purchased the voice module don't need to worry. When the voice module is not connected, the K230 can still implement this case, just without the voice broadcast function.**

go_straight 0.95



Road sign images

go_straight:



turn_right:



speed_limit:



stop:

