# K230 Rock Paper Scissors (Voice)


**K230 Rock Paper Scissors (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 Results


In this section, we will learn how to use the K230 to perform static gesture recognition and create a rock paper game with a small car.

# I. Software-Hardware

- **KEIL5**
- **MSPM0G3507 Robot Development Board**

K230 Module: External

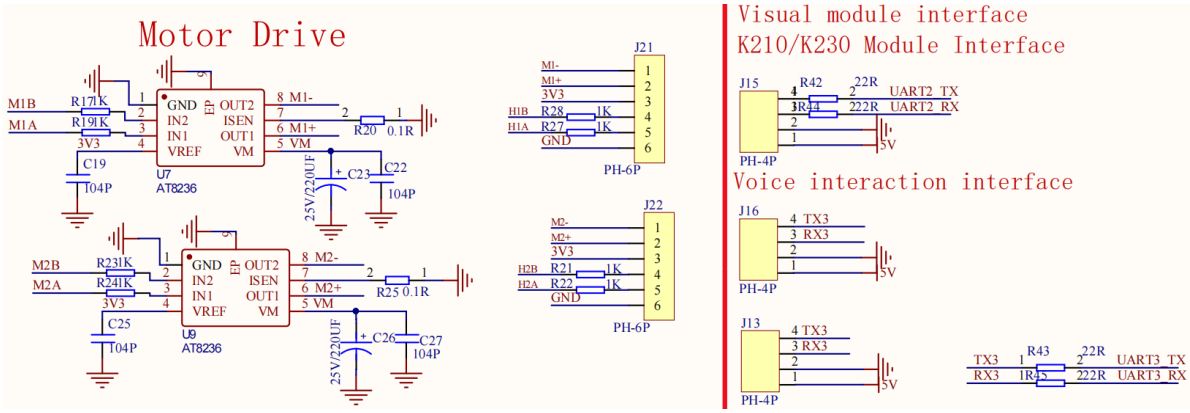
AI Voice Interaction Integration Module: External

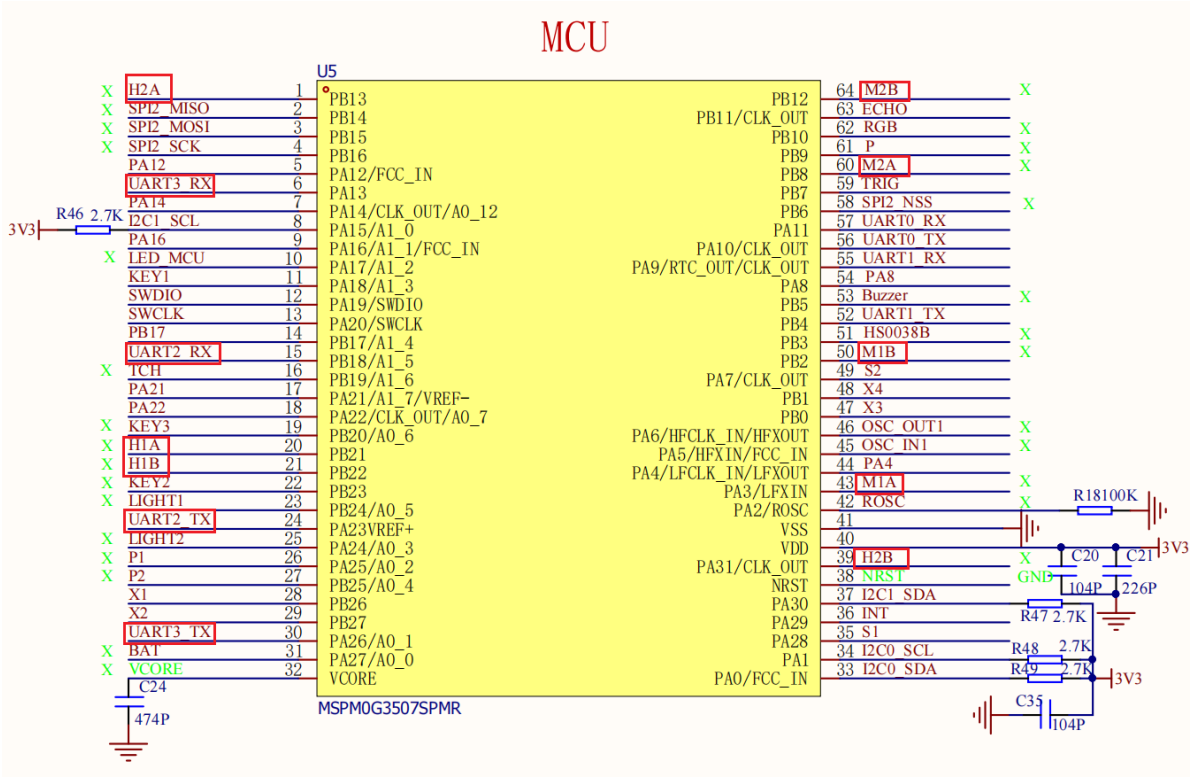
- **Type-C Data Cable or DAP-Link**

Download or simulate the program on the development board.

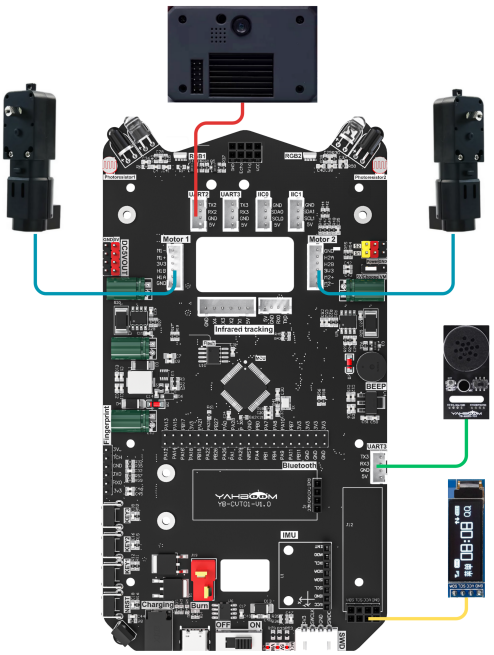
# II. Brief Principles

## 1. Hardware Schematic Diagram






## 2. Physical Connection Diagram



**Motor Wiring** (Note: The wiring diagram below is for reference only. Our products come with a dual-head PH2.0 connector.) 6-pin all-black cable with foolproof design, no need to worry about wiring problems.

| TT Encoder Motor | MSPM0G3507 |
|---|---|
| M - | M1- |
| M + | M1+ |
| VCC | 3V3 |
| Encoder B | H1B |
| Encoder A | 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 factory-installed dual-ended PH2.0 4-pin all-black cable for the voice module, featuring a foolproof design. No wiring issues are required.)

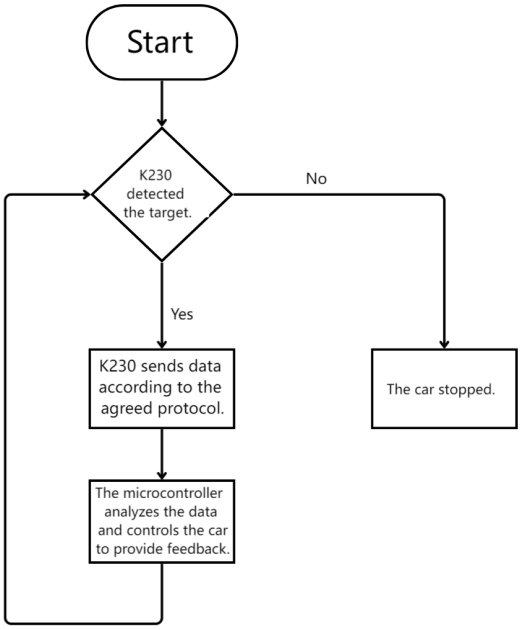
| Intelligent Voice Interaction Module | MSPM0G3507 |
|---|---|
| RX1 | TX3 |
| TX1 | RX3 |
| GND | GND |
| 5V | 5V |

**K230 Wiring** (Note: The wiring diagram below is for reference only. Our products come with a factory-installed K230 dual-ended PH2.0 4-pin all-black cable, featuring a foolproof design. No wiring issues are required.)

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

## 3. Control Principle

- System Program Flowchart




- **Communication Protocol**

| Start Character | Length | Separator | Routine Number | Separator | msg | End Character |
|---|---|---|---|---|---|---|
| $ | XX | , | 23 | , | ... | # |

**msg: you lose, you win, tie**

**Voice Module Command Words**

| Command Word | Function Type | Broadcast Statement | Broadcast Mode | Send Protocol | Receive Protocol |
|---|---|---|---|---|---|
| You won | Command words | You won, you're amazing! | Main | AA 55 00 57 FB | AA 55 00 57 FB |
| You lost | Command words | You lost, defeated opponent! | Main | AA 55 00 58 FB | AA 55 00 58 FB |
| Draw | Command words | It seems like a very evenly matched contest! | Main | AA 55 00 59 FB | AA 55 00 59 FB |

# 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**

- 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 within, and the corresponding processing function is executed. This project uses metadata parsing. The core idea is to separate the protocol format description (field position, type, etc.) from the parsing logic. The protocol format is defined through data structures (such as structure arrays), and the parsing function automatically processes the data based on these descriptions.

The protocol can also parse data containing different data types. For details on the implementation, please research further; we will not provide further explanation here. This function is compatible with all K230 communication protocols in our store; you don't need to fully understand it to use it normally.

- bsp_K230_control.c

```
// Rock, Paper, Scissors
void Rock_Result(const char* msg)
{
    if(strcmp(msg,"lose")==0)
    {
            Write_Data(0x58);

//
    Control_RGB_ALL(Red_RGB);
        delay_ms(1000);
        delay_ms(1000);
        Control_RGB_ALL(OFF);
        delay_ms(100);

    }
    else if(strcmp(msg,"win")==0)
    {
                Write_Data(0x57);
        Control_RGB_ALL(Green_RGB);

                delay_ms(1000);
            delay_ms(1000);
             Control_RGB_ALL(OFF);
            delay_ms(100);
    }

    else {
                    Write_Data(0x5A);

        Control_RGB_ALL(Yellow_RGB);
                        delay_ms(1000);
            delay_ms(1000);
             Control_RGB_ALL(OFF);
            delay_ms(100);

    }
}
```

Rock_Result: After receiving data from K230, press the reset button, wait three seconds, and then start the game. React according to the game situation.

- 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 误差累积

        /* 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 both 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.

```
# Rock-Paper-Scissors Game Tasks
class FingerGuess:
    def __init__(self, hand_det_kmodel, hand_kp_kmodel, det_input_size,
kp_input_size, labels, anchors, confidence_threshold=0.25, nms_threshold=0.3,
nms_option=False, strides=[8, 16, 32], guess_mode=3, rgb888p_size=[1280, 720],
display_size=[1920, 1080], debug_mode=0):
        # Hand detection model path
        self.hand_det_kmodel = hand_det_kmodel
        # Hand key point model path
        self.hand_kp_kmodel = hand_kp_kmodel
        # Hand detection model input resolution
        self.det_input_size = det_input_size
        # Hand key point model input resolution
        self.kp_input_size = kp_input_size
        self.labels = labels
        # anchors / Anchor boxes
        self.anchors = anchors
        # Confidence threshold
        self.confidence_threshold = confidence_threshold
        # nms threshold
        self.nms_threshold = nms_threshold
        # nms option
        self.nms_option = nms_option
        # Feature map downsampling factor of input
        self.strides = strides
        # The sensor provides the AI ••with an image resolution that is 16 bytes
wide and aligned.
        self.rgb888p_size = [ALIGN_UP(rgb888p_size[0], 16), rgb888p_size[1]]
        # Video output at VO resolution, 16 bytes wide and aligned.
        self.display_size = [ALIGN_UP(display_size[0], 16), display_size[1]]
        # debug_mode
        self.debug_mode = debug_mode
```

```
        self.guess_mode = guess_mode
        # Rock-Paper-Scissors texture array
        self.five_image = self.read_file("/sdcard/utils/five.bin")
        self.fist_image = self.read_file("/sdcard/utils/fist.bin")
        self.shear_image = self.read_file("/sdcard/utils/shear.bin")
        self.counts_guess = -1
            # Number of rock-paper-scissors games
        self.player_win = 0
             # Player win count
        self.k230_win = 0
             # K230 Win Count
        self.sleep_end = False
            # Pause?
        self.set_stop_id = True
             # Should we suspend rock-paper-scissors?
        self.LIBRARY = ["fist", "yeah", "five"]
             # Rock-Paper-Scissors: Three possible outcomes
        self.hand_det = HandDetApp(self.hand_det_kmodel, self.labels,
model_input_size=self.det_input_size, anchors=self.anchors,
confidence_threshold=self.confidence_threshold,
nms_threshold=self.nms_threshold, nms_option=self.nms_option,
strides=self.strides, rgb888p_size=self.rgb888p_size,
display_size=self.display_size, debug_mode=0)
        self.hand_kp = HandKPClassApp(self.hand_kp_kmodel,
model_input_size=self.kp_input_size, rgb888p_size=self.rgb888p_size,
display_size=self.display_size)
        self.hand_det.config_preprocess()


    # run function
    def run(self, input_np):
        # First, perform a palm scan.
        det_boxes = self.hand_det.run(input_np)
        boxes = []
        gesture_res = []
        for det_box in det_boxes:
            # Gesture recognition of the detected hand
            x1, y1, x2, y2 = det_box[2], det_box[3], det_box[4], det_box[5]
            w, h = int(x2 - x1), int(y2 - y1)
            if (h < (0.1 * self.rgb888p_size[1])):
                continue
            if (w < (0.25 * self.rgb888p_size[0]) and ((x1 < (0.03 *
self.rgb888p_size[0])) or (x2 > (0.97 * self.rgb888p_size[0])))):
                continue
            if (w < (0.15 * self.rgb888p_size[0]) and ((x1 < (0.01 *
self.rgb888p_size[0])) or (x2 > (0.99 * self.rgb888p_size[0])))):
                continue
            self.hand_kp.config_preprocess(det_box)
            results_show, gesture = self.hand_kp.run(input_np)
            boxes.append(det_box)
            gesture_res.append(gesture)
        return boxes, gesture_res


    # Drawing effect
    def draw_result(self, pl, dets, gesture_res):
        pl.osd_img.clear()
```

```
        # The system categorizes hand gestures to determine the user's punch,
outputs a punch response from the development board based on different patterns,
and displays the corresponding texture on the screen.

        if (len(dets) >= 2):
            pl.osd_img.draw_string_advanced(self.display_size[0] // 2 - 50,
self.display_size[1] // 2 - 50, 30, " Make sure only one hand on the screen",
color=(255, 255, 0, 0))
        elif (self.guess_mode == 0):
            draw_img_np = np.zeros((self.display_size[1], self.display_size[0],
4), dtype=np.uint8)
            draw_img = image.Image(self.display_size[0], self.display_size[1],
image.ARGB8888, alloc=image.ALLOC_REF, data=draw_img_np)
            if (gesture_res[0] == "fist"):
                draw_img_np[:400, :400, :] = self.shear_image
            elif (gesture_res[0] == "five"):
                draw_img_np[:400, :400, :] = self.fist_image
            elif (gesture_res[0] == "yeah"):
                draw_img_np[:400, :400, :] = self.five_image
            pl.osd_img.copy_from(draw_img)
        elif (self.guess_mode == 1):
            draw_img_np = np.zeros((self.display_size[1], self.display_size[0],
4), dtype=np.uint8)
            draw_img = image.Image(self.display_size[0], self.display_size[1],
image.ARGB8888, alloc=image.ALLOC_REF, data=draw_img_np)
            if (gesture_res[0] == "fist"):
                draw_img_np[:400, :400, :] = self.five_image
            elif (gesture_res[0] == "five"):
                draw_img_np[:400, :400, :] = self.shear_image
            elif (gesture_res[0] == "yeah"):
                draw_img_np[:400, :400, :] = self.fist_image
            pl.osd_img.copy_from(draw_img)
        else:
            draw_img_np = np.zeros((self.display_size[1], self.display_size[0],
4), dtype=np.uint8)
            draw_img = image.Image(self.display_size[0], self.display_size[1],
image.ARGB8888, alloc=image.ALLOC_REF, data=draw_img_np)
            if (self.sleep_end):
                time.sleep_ms(2000)
                self.sleep_end = False
            if (len(dets) == 0):
                self.set_stop_id = True
                return
            if (self.counts_guess == -1 and gesture_res[0] != "fist" and
gesture_res[0] != "yeah" and gesture_res[0] != "five"):
                draw_img.draw_string_advanced(self.display_size[0] // 2 - 50,
self.display_size[1] // 2 - 50, 30, " GAME START", color=(255, 255, 0, 0))
                time.sleep_ms(500)
            elif (self.counts_guess == self.guess_mode):
                draw_img.clear()
                if (self.k230_win > self.player_win):
                    draw_img.draw_string_advanced(self.display_size[0] // 2 -
50, self.display_size[1] // 2 - 50, 30, " You lose", color=(255, 255, 0, 0))
                elif (self.k230_win < self.player_win):
                    draw_img.draw_string_advanced(self.display_size[0] // 2 -
50, self.display_size[1] // 2 - 50, 30, " You win", color=(255, 255, 0, 0))
                else:
```

```
                    draw_img.draw_string_advanced(self.display_size[0] // 2 -
50, self.display_size[1] // 2 - 50, 30, " | tie", color=(255, 255, 0, 0))
                self.counts_guess = -1
                self.player_win = 0
                self.k230_win = 0
                self.sleep_end = True
            else:
                if (self.set_stop_id):
                    if (self.counts_guess == -1 and (gesture_res[0] == "fist" or
gesture_res[0] == "yeah" or gesture_res[0] == "five")):
                        self.counts_guess = 0
                    if (self.counts_guess != -1 and (gesture_res[0] == "fist" or
gesture_res[0] == "yeah" or gesture_res[0] == "five")):
                        k230_guess = randint(1, 10000) % 3
                        if (gesture_res[0] == "fist" and
self.LIBRARY[k230_guess] == "yeah"):
                            self.player_win += 1
                        elif (gesture_res[0] == "fist" and
self.LIBRARY[k230_guess] == "five"):
                            self.k230_win += 1
                        if (gesture_res[0] == "yeah" and
self.LIBRARY[k230_guess] == "fist"):
                            self.k230_win += 1
                        elif (gesture_res[0] == "yeah" and
self.LIBRARY[k230_guess] == "five"):
                            self.player_win += 1
                        if (gesture_res[0] == "five" and
self.LIBRARY[k230_guess] == "fist"):
                            self.player_win += 1
                        elif (gesture_res[0] == "five" and
self.LIBRARY[k230_guess] == "yeah"):
                            self.k230_win += 1
                        if (self.LIBRARY[k230_guess] == "fist"):
                            draw_img_np[:400, :400, :] = self.fist_image
                        elif (self.LIBRARY[k230_guess] == "five"):
                            draw_img_np[:400, :400, :] = self.five_image
                        elif (self.LIBRARY[k230_guess] == "yeah"):
                            draw_img_np[:400, :400, :] = self.shear_image
                        self.counts_guess += 1
                        draw_img.draw_string_advanced(self.display_size[0] // 2
- 50, self.display_size[1] // 2 - 50, 30, "第" + str(self.counts_guess) + " 回合
|Round " +  str(self.counts_guess), color=(255, 255, 0, 0))
                        self.set_stop_id = False
                        self.sleep_end = True
                    else:
                        draw_img.draw_string_advanced(self.display_size[0] // 2
- 50, self.display_size[1] // 2 - 50, 30, "第" + str(self.counts_guess + 1) + "
回合 |Round " +  str(self.counts_guess), color=(255, 255, 0, 0))
            pl.osd_img.copy_from(draw_img)


    # How to read the bin file of rock-paper-scissors
    def read_file(self, file_name):
        image_arr = np.fromfile(file_name, dtype=np.uint8)
        image_arr = image_arr.reshape((400, 400, 4))
        return image_arr
```

## 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 |
| STOPT-REPORT | 关播报 (Disable Broadcast) | Stop Broadcast | 主 (Active) | AA 55 0A 00 FB | AA 55 0A 00 FB |
| YOU-WIN | 命令词 (Command Word) | You won you are amazing | 主 (Active) | AA 55 00 57 FB | AA 55 00 57 FB |
| YOU-LOST | 命令词 (Command Word) | You have lost you defeated opponent | 主 (Active) | AA 55 00 58 FB | AA 55 00 58 FB |
| DRAW | 命令词 (Command Word) | It seems this is a closely contested match | 主 (Active) | AA 55 00 59 FB | AA 55 00 59 FB |

# V. Experimental Operation

1. Connect the hardware according to the hardware wiring instructions in the tutorial.
2. Open the project used by MSPM0G3507, compile and flash it, and finally reset or power on again.
3. Place dynamic_gesture.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. After the K230 initializes successfully, it will begin recognizing your gestures. Press the car reset button, wait three seconds, and then you can play rock-paper-scissors with the K230. The result will be output to the car, allowing it to move.

**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 Results

In this section, we will learn about static gesture recognition based on the K230 and play a rock-paper-scissors game with the car.

The game execution process is as follows:

After the game starts, we make the "rock," "scissors," or "paper" gesture with our hands, which will appear in the camera's view. The screen will then display the program's move and the win/loss result. Each time a winner is determined, the round ends. After each round, please move your hand out of the camera's range.


The game consists of three rounds. After three rounds, the program will determine the final result according to the best-of-three rule. After the winner is determined, the K230 will send a data frame to the MSPM0G3507 car. We will parse the message according to the protocol and perform the corresponding action. When the player wins, the car will light up green and shake; when the machine wins, it will light up red. [There is a small bug here: if the program does not immediately

give the final result after the third round, you can let your hand re-enter the camera, and the program will then give the final result! **Note: When connected to our AI voice interaction integration module, there will be a voice broadcast when the game results are announced. Users who have not purchased the voice module do not need to worry; the K230 can still perform this case without the voice broadcast function.**