stm32_k230 face detection

stm32_k230 face detection
- k230 and stm32 communication
  - 1. Experimental Prerequisites
  - 2. Experimental wiring
  - 3. Main code explanation
  - 4. Experimental Phenomenon

k230 and stm32 communication

1. Experimental Prerequisites

This tutorial uses the STM32F103C8T6 development board, and the corresponding routine path is [14.export\STM32-K230\06_STM32_k230_face_detect].

K230 needs to run the [14.export\CanmvIDE-K230\06.face_detection.py] program to start the experiment. It is recommended to download it as an offline program.

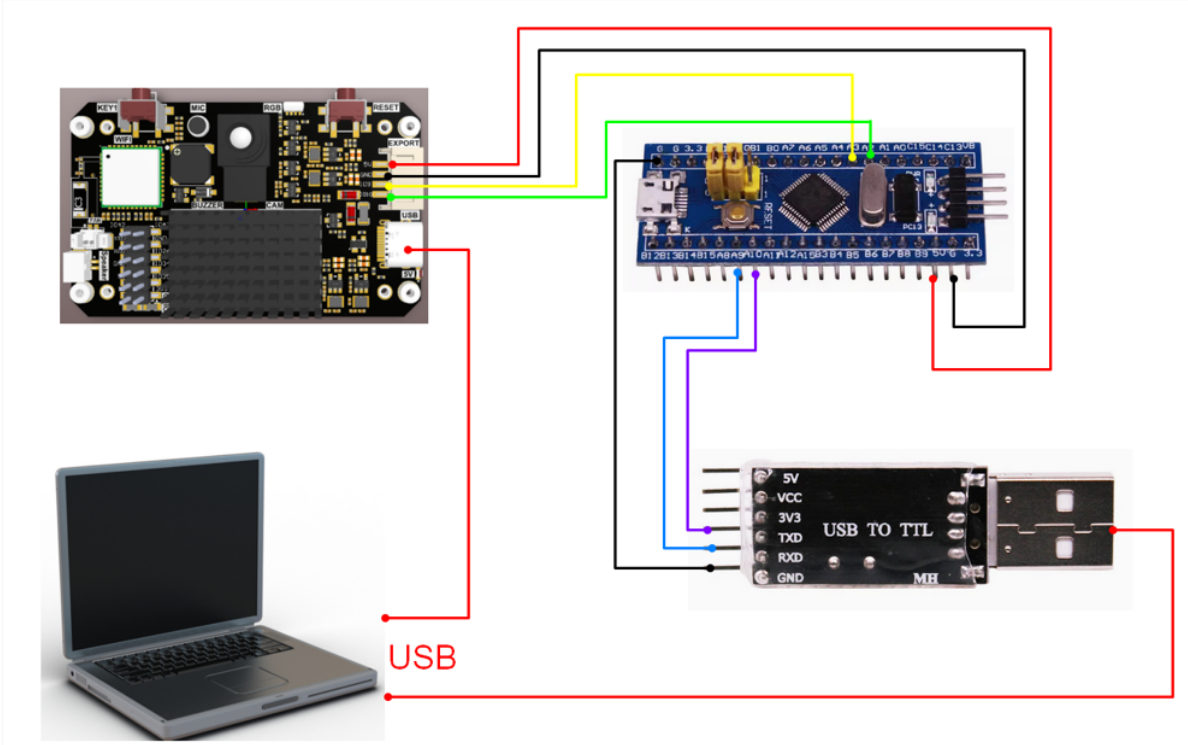
Things you need:

Windows computer, STM32F103C8T6 development board, USB to TTL module, K230 visual module (including TF card with burned image), Type-C data cable, connection cable (Dupont cable)

2. Experimental wiring

k230 vision module	**STM32**
5V	VCC
GND	GND
TXD(IO9)	PA3
RXD(IO10)	PA2

USB to TTL module	**STM32**
RxD	PA9
TXD	PA10
GND	GND



3. Main code explanation

```
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];
    if (!(pto_head == PTO_HEAD && pto_tail == PTO_TAIL))
    {
        printf("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};
    for (i = 1; i < num-1; i++)
    {
        if (data_buf[i] == ',')
        {
            data_buf[i] = 0;
            field_index[data_index] = i;
            data_index++;
        }
    }

    for (i = 0; i < data_index; i++)
    {
        values[i] = Pto_Char_To_Int((char*)data_buf+field_index[i]+1);
    }

    uint8_t pto_len = values[0];

    if (pto_len != num)
```

```
    {
        printf("pto_len error:%d , data_len:%d\n", pto_len, num);
        return;
    }
    uint8_t pto_id = values[1];
    if (pto_id != PTO_FUNC_ID)
    {
        printf("pto_id error:%d, func_id:%d\n", pto_id, PTO_FUNC_ID);
        return;
    }
    int x = values[2];
    int y = values[3];
    int w = values[4];
    int h = values[5];
    printf("face:x:%d, y:%d, w:%d, h:%d\n", x, y, w, h);
}
```

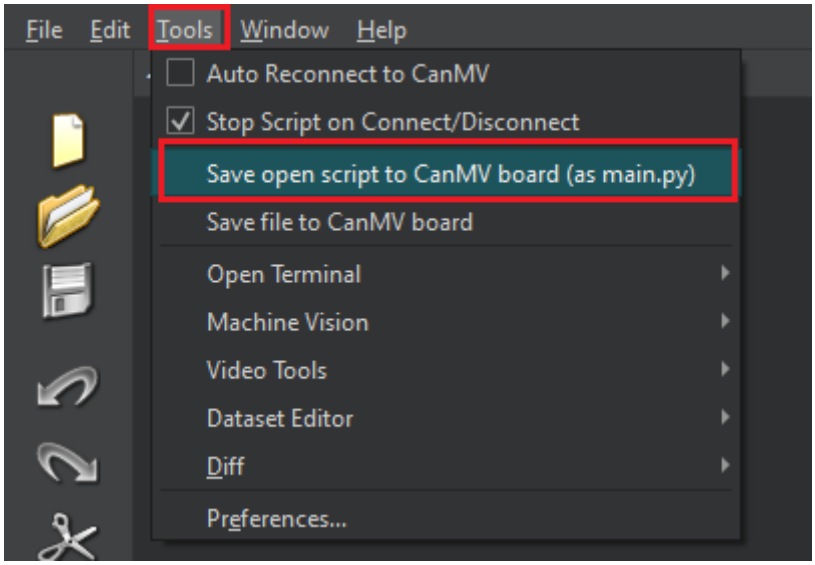
The above functions are for parsing K230 data. Only when they meet specific protocols can the corresponding data be parsed.

in

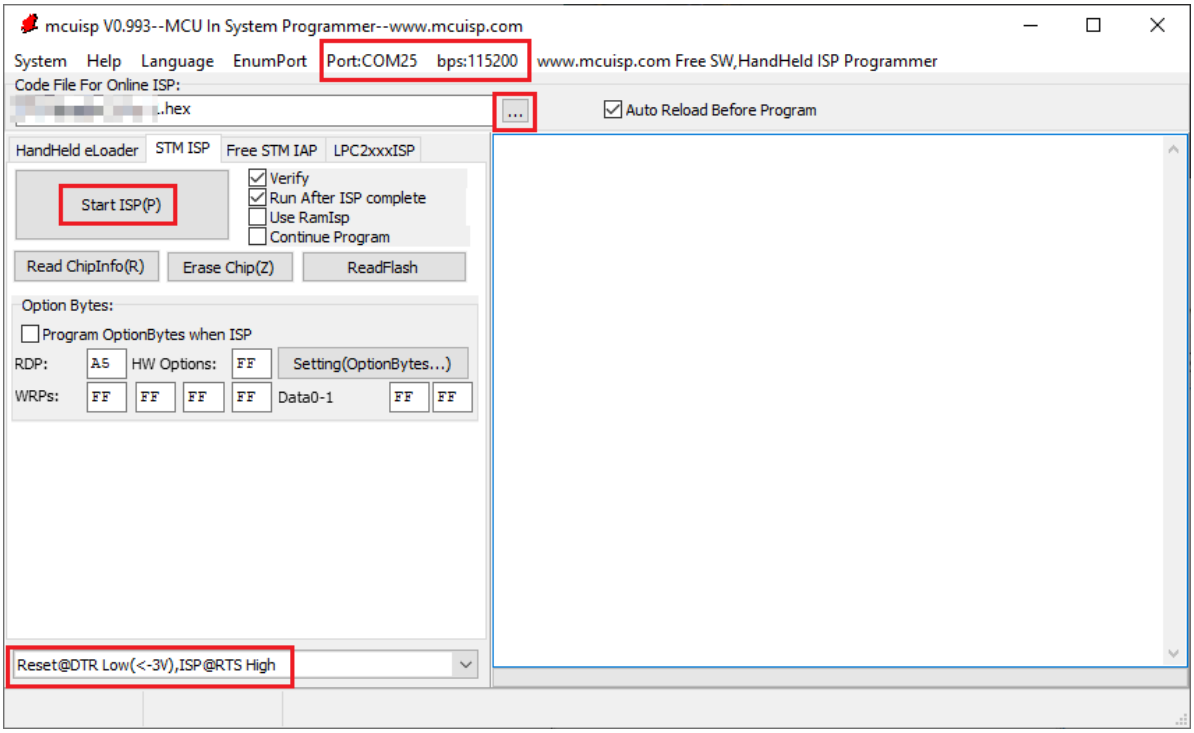
- x: is the horizontal coordinate of the upper left corner of the recognized box
- y: is the vertical coordinate of the upper left corner of the recognized box
- w: is the width of the recognized frame
- h: is the length of the recognized frame

4. Experimental Phenomenon

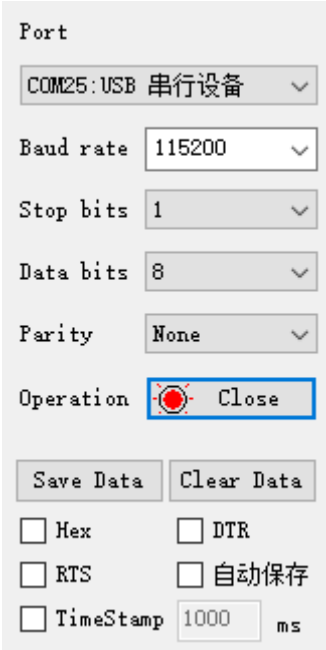
1. After connecting the cables, the k230 visual module runs offline. After K230 is connected to Canmv IDE, open the corresponding program, click [Save open script to CanMV board (as main.py)] on the toolbar, and then restart K230.



2. The hex file generated by the STM32 burning routine.



3. The serial port assistant is set to the interface shown in the figure



4. When the K230 camera recognizes a face, the serial port assistant will print out the information transmitted from k230 to stm32.

in

- x: is the horizontal coordinate of the upper left corner of the recognized box
- y: is the vertical coordinate of the upper left corner of the recognized box
- w: is the width of the recognized frame
- h: is the length of the recognized frame

  As shown in the figure below

face:x:295, y:161, w:73, h:112
face:x:295, y:160, w:73, h:114
face:x:295, y:160, w:72, h:113
face:x:294, y:160, w:73, h:113
face:x:294, y:160, w:73, h:114
face:x:294, y:160, w:73, h:113
face:x:294, y:160, w:73, h:113
face:x:294, y:160, w:73, h:113
face:x:294, y:160, w:73, h:114
face:x:294, y:160, w:72, h:113
face:x:294, y:158, w:73, h:113
face:x:294, y:160, w:72, h:113
face:x:294, y:160, w:72, h:113