k230 QR code recognition

K230 and Raspberry Pi communication

1. Experimental Prerequisites

This tutorial uses the Raspberry Pi 5 development board, and the corresponding routine path is [14.export\Raspberrypi-K230\03_k230_qrcode.py].

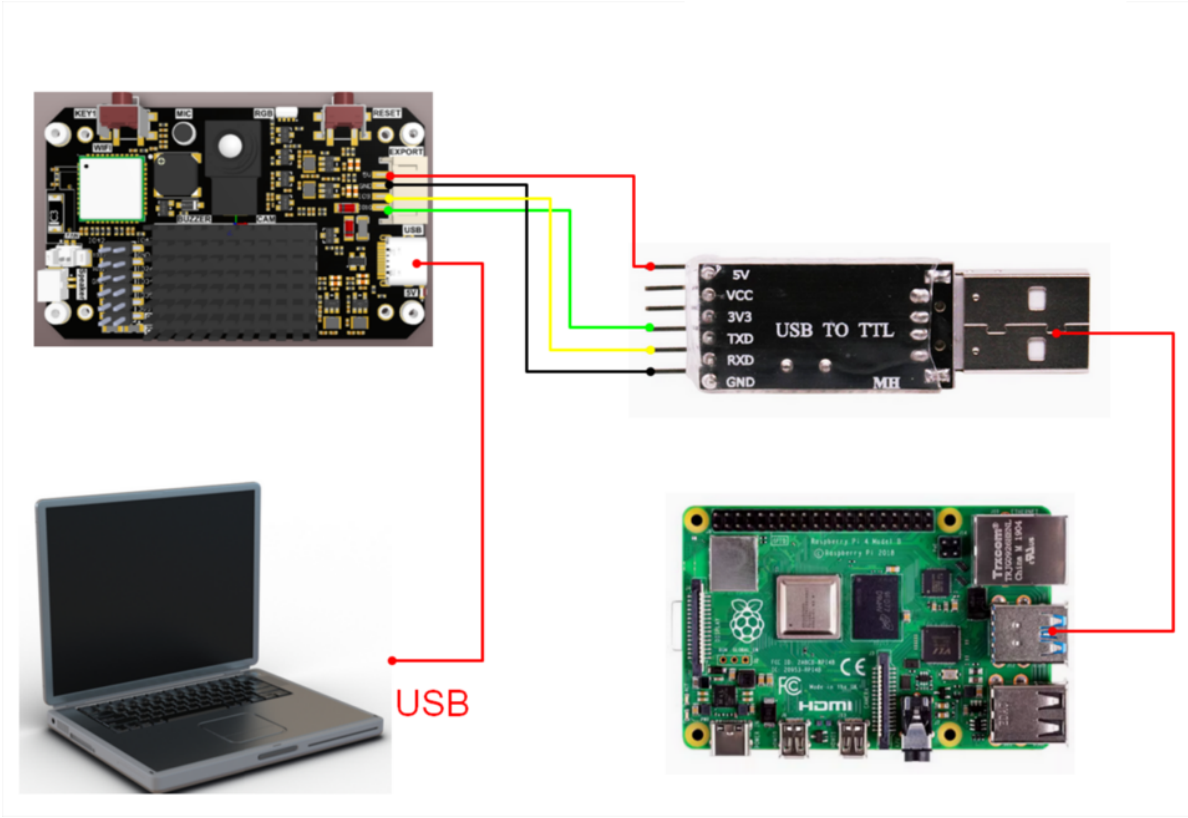
K230 needs to run the [14.export\CanmvIDE-K230\03.find_qrcodes.py] program to start the experiment. It is recommended to download it as an offline program.

Things you need:

Windows computer
Raspberry Pi 5 development board
microUSB cable
K230 visual module (including TF card with image burned in)
type-C cable
connection cable

2. Experimental wiring

k230 vision module	USB to TTL module
5V	VCC
GND	GND
TXD(IO9)	RxD
RXD(IO10)	TXD



3. Main code explanation

```
import serial

com="/dev/ttyUSB0"
ser = serial.Serial(com, 115200)

FUNC_ID = 3

def parse_data(data):
    if data[0] == ord('$') and data[len(data)-1] == ord('#'):
        data_list = data[1:len(data)-1].decode('utf-8').split(',')
        data_len = int(data_list[0])
        data_id = int(data_list[1])
        if data_len == len(data) and data_id == FUNC_ID:
            # print(data_list)
            x = int(data_list[2])
            y = int(data_list[3])
            w = int(data_list[4])
            h = int(data_list[5])
            msg = data_list[6]
            return x, y, w, h, msg
        elif (data_len != len(data)):
            print("data len error:", data_len, len(data))
        elif(data_id != FUNC_ID):
            print("func id error:", data_id, FUNC_ID)
    return -1, -1, -1, -1, ""

while True:
    if ser.in_waiting:
        data = ser.readline()
        # print("rx:", data)
        x, y, w, h, msg = parse_data(data.rstrip(b'\n'))
```

```
        print("qrcode:x:%d, y:%d, w:%d, h:%d" % (x, y, w, h), "payload:", msg)
```

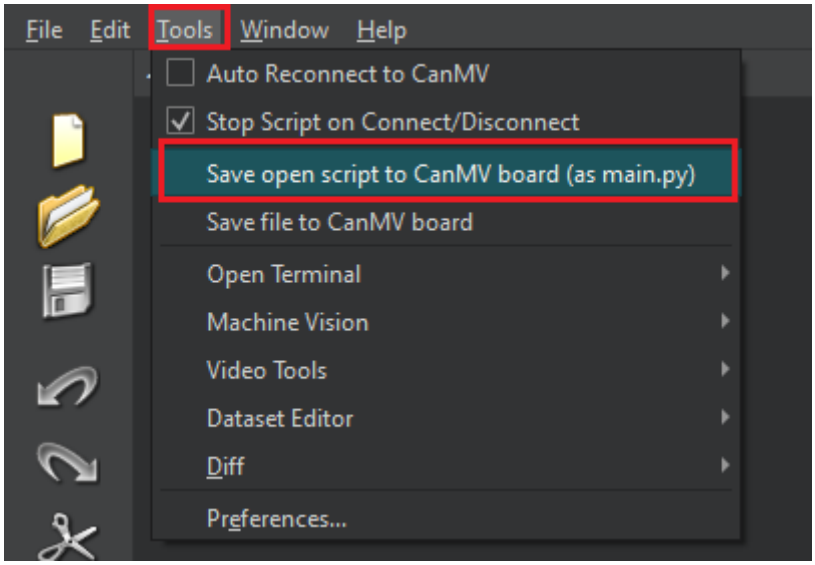
The above program is for parsing K230 data. Only when it complies with specific protocols can the corresponding data be parsed.

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
- msg: is the content of the QR code

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. Transfer the program file to the system, open the terminal and enter the corresponding directory, then run the following command to start the program.

```
python3 03_k230_qrcode.py
```

3. When the K230 camera screen recognizes the QR code, the terminal will parse and print out the information transmitted by the K230.

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
- msg: is the content of the QR code

  As shown in the figure below

```
qrcode:x:235, y:80, w:154, h:208, msg:'goahead'
qrcode:x:236, y:79, w:155, h:203, msg:'goahead'
qrcode:x:237, y:75, w:154, h:207, msg:'goahead'
qrcode:x:238, y:74, w:156, h:206, msg:'goahead'
qrcode:x:240, y:72, w:155, h:209, msg:'goahead'
qrcode:x:242, y:67, w:155, h:202, msg:'goahead'
qrcode:x:244, y:62, w:155, h:209, msg:'goahead'
qrcode:x:246, y:57, w:155, h:206, msg:'goahead'
qrcode:x:247, y:52, w:155, h:204, msg:'goahead'
qrcode:x:251, y:42, w:157, h:206, msg:'goahead'
qrcode:x:254, y:35, w:155, h:206, msg:'goahead'
qrcode:x:257, y:28, w:157, h:204, msg:'goahead'
qrcode:x:261, y:12, w:157, h:208, msg:'goahead'
```