k230 barcode recognition

k230 and PICO2 communication

1. Experimental Prerequisites

This tutorial uses the PICO2 development board, and the corresponding routine path is [14.export\PICO-K230\02_pico_k230_barcode.py].

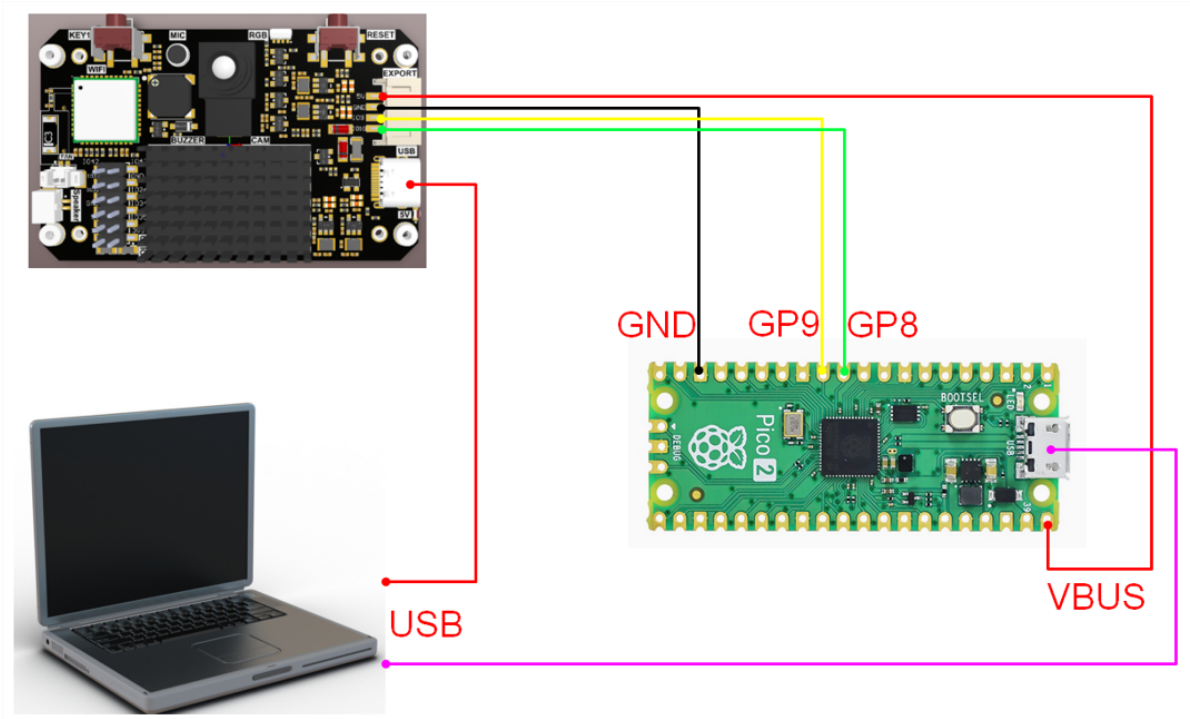
K230 needs to run the [14.export\CanmvIDE-K230\02.find_barcodes.py] program to start the experiment. It is recommended to download it as an offline program.

Things you need:

Windows computer
PICO2 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	PICO2 Development Board
5V	VCC
GND	GND
TXD(IO9)	RXD(GP9)
RXD(IO10)	TXD(GP8)



3. Main code explanation

```
from machine import UART, Pin

FUNC_ID = 2

uart1 = UART(1, baudrate=115200, tx=Pin(8), rx=Pin(9), bits=8, parity=None,
stop=0)

print("hello yahboom")


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, ""

last_data = bytearray()
while True:
    if uart1.any() > 0:
        cur_data = uart1.readline()
        # print("rx:", cur_data)
```

```
        if ord('\n') in cur_data:
            # data = bytearray(last_data + cur_data.decode('utf-8'), 'utf-8')
            data = last_data + cur_data
            last_data = bytearray()
            x, y, w, h, msg = parse_data(data.rstrip(b'\n'))
            print("barcode:x:%d, y:%d, w:%d, h:%d" % (x, y, w, h), "payload:",
msg)
        else:
            last_data = last_data + cur_data
```

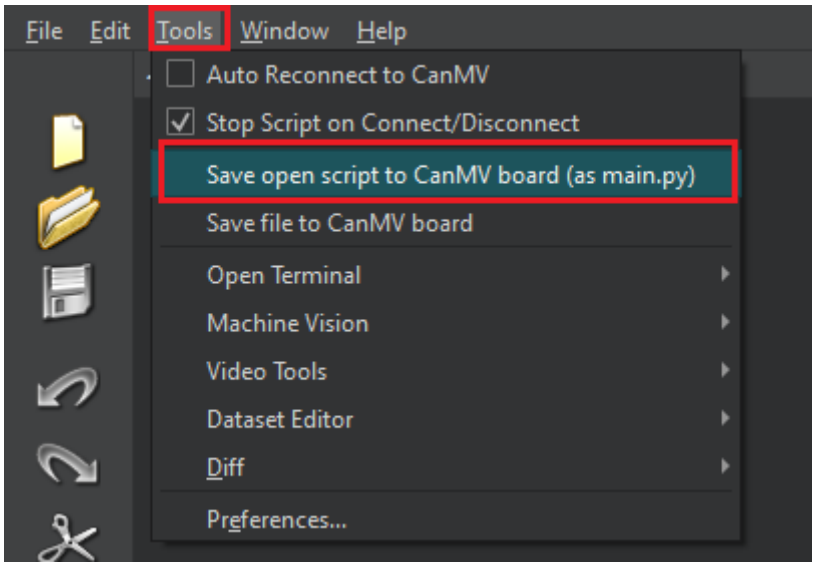
The above program is for parsing K230 data. Only when it complies with specific protocols can the corresponding data be parsed.

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 barcode

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. Open the Thonny editor, connect the PICO2 mainboard, open the program file and run it.

   Note: The PICO2 mainboard needs to have the microPython firmware downloaded in advance.

3. When the K230 camera image recognizes the barcode, 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 barcode

  As shown in the figure below

```
[2025-04-30 10:46:32.326]# RECV ASCII>
barcode:x:282, y:79, w:138, h:134, msg:'12345670'

[2025-04-30 10:46:32.453]# RECV ASCII>
barcode:x:282, y:77, w:139, h:136, msg:'12345670'

[2025-04-30 10:46:32.565]# RECV ASCII>
barcode:x:283, y:76, w:139, h:135, msg:'12345670'

[2025-04-30 10:46:32.693]# RECV ASCII>
barcode:x:284, y:73, w:139, h:136, msg:'12345670'

[2025-04-30 10:46:32.820]# RECV ASCII>
barcode:x:283, y:70, w:139, h:135, msg:'12345670'

[2025-04-30 10:46:32.932]# RECV ASCII>
barcode:x:280, y:31, w:143, h:130, msg:'12345670'

[2025-04-30 10:46:33.060]# RECV ASCII>
barcode:x:253, y:0, w:148, h:39, msg:'12345670'
```