

# arduino\_k230 QR code recognition

---

## arduino\_k230 QR code recognition

### k230 and arduino communication

1. Experimental Prerequisites
2. Experimental wiring
3. Main code explanation
4. Experimental Phenomenon

## k230 and arduino communication

---

### 1. Experimental Prerequisites

This tutorial uses Arduino, and the corresponding routine path is [14.export\arduino-K230\3.Arduino\_k230\_qrcode].

K230 needs to run the [14.export\CanmvIDE-K230\01.color\_detect.py] program to start the experiment. It is recommended to download it as an offline program.

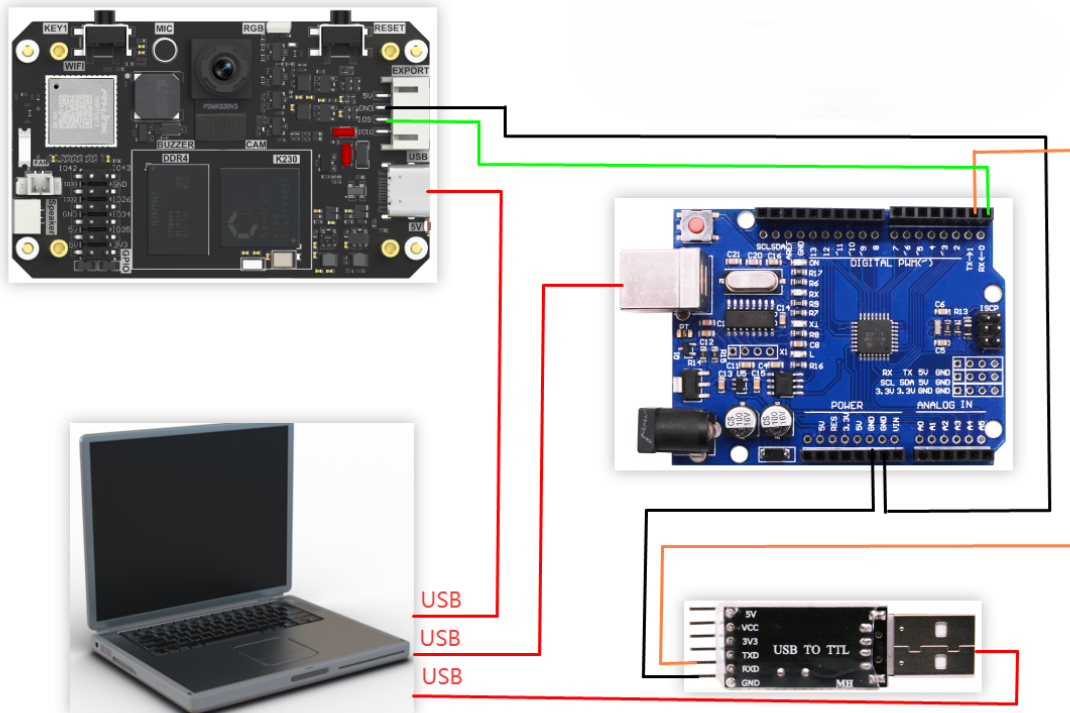
Items needed:

Windows computer, Arduino, USB to TTL module, K230 visual module (including TF card with image burned in), type-C data cable, connecting cable (Dupont cable)

### 2. Experimental wiring

K230 vision module	Arduino
GND	GND
TXD(IO9)	RXD (0)

USB to TTL module	Arduino
RXD	TXD (1)
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))
    {
        Serial.print("pto error:pto_head=0x");
        Serial.print(pto_head, HEX);
        Serial.print(" , pto_tail=0x");
        Serial.println(pto_tail, HEX);
        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 msg[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++)
    {
        if (i == 6)
        {

```

```

        memcpy(msg, (char*)data_buf+field_index[i]+1, values[0]-
field_index[i]-2);
    }
    else
    {
        values[i] = Pto_Char_To_Int((char*)data_buf+field_index[i]+1);
    }
}

uint8_t pto_len = values[0];

if (pto_len != num)
{
    Serial.print("pto_len error:");
    Serial.print(pto_len);
    Serial.print(" , data_len:");
    Serial.println(num);
    return;
}
uint8_t pto_id = values[1];
if (pto_id != PTO_FUNC_ID)
{
    Serial.print("pto_id error:");
    Serial.print(pto_id);
    Serial.print(" , func_id:");
    Serial.println(PTO_FUNC_ID);
    return;
}
int x = values[2];
int y = values[3];
int w = values[4];
int h = values[5];
Serial.print("qrcode:x:");
Serial.print(x);
Serial.print(" , y:");
Serial.print(y);
Serial.print(" , w:");
Serial.print(w);
Serial.print(" , h:");
Serial.print(h);
Serial.print(" , msg: '");
Serial.print(msg);
Serial.println("'");
}

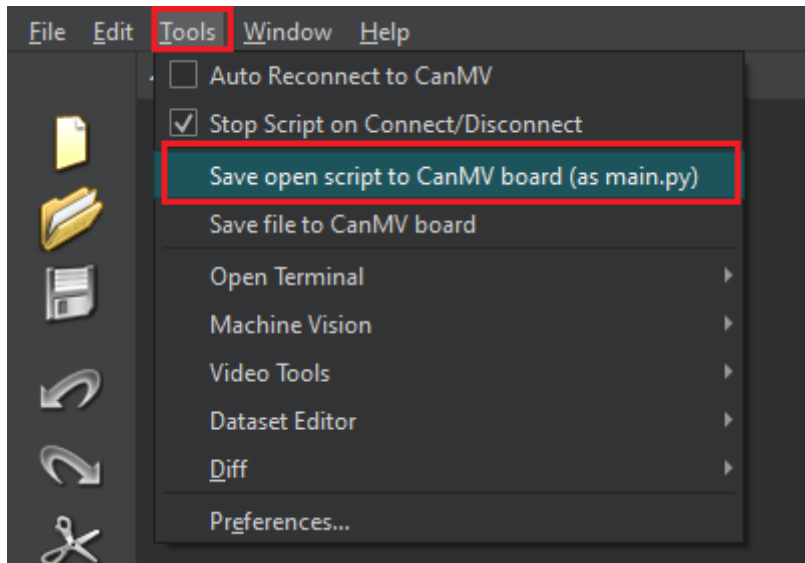
```

The above function is used to parse K230 data. Only when it complies with specific protocols can the corresponding data be parsed.

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

## 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. Arduino upload routine code (**Note that if the upload fails, disconnect the RXD connection on the Arduino connected to the k230 first, and then plug it back after the upload is successful**)



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

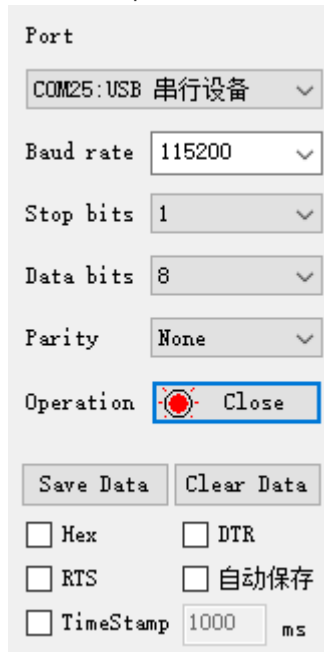
Port  
COM25:USB 串行设备

Baud rate 115200

Stop bits 1

Data bits 8

Parity None

Operation  Close

Save Data Clear Data

☐ Hex ☐ DTR  
☐ RTS ☐ 自动保存  
☐ TimeStamp 1000 ms

4. When the K230 camera recognizes the QR code, the serial port assistant will print out the information transmitted from K230 to Arduino.

- x: is the horizontal coordinate of the upper left corner of the identified box
- y: is the vertical coordinate of the upper left corner of the identified box
- w: is the width of the identified box
- h: is the length of the identified box
- msg: is the content of the QR code

As shown 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'
```