arduino_k230 gesture recognition

k230 and arduino communication

1.Experimental Prerequisites

This tutorial uses an Arduino, and the corresponding example code path is [14.export\arduino-K230\12.Arduino_k230_gesture].

To begin the experiment, you must run the [14.export\CanmvIDE-K230\12.gesture.py] program on the K230. It is recommended that you download the program for offline operation.

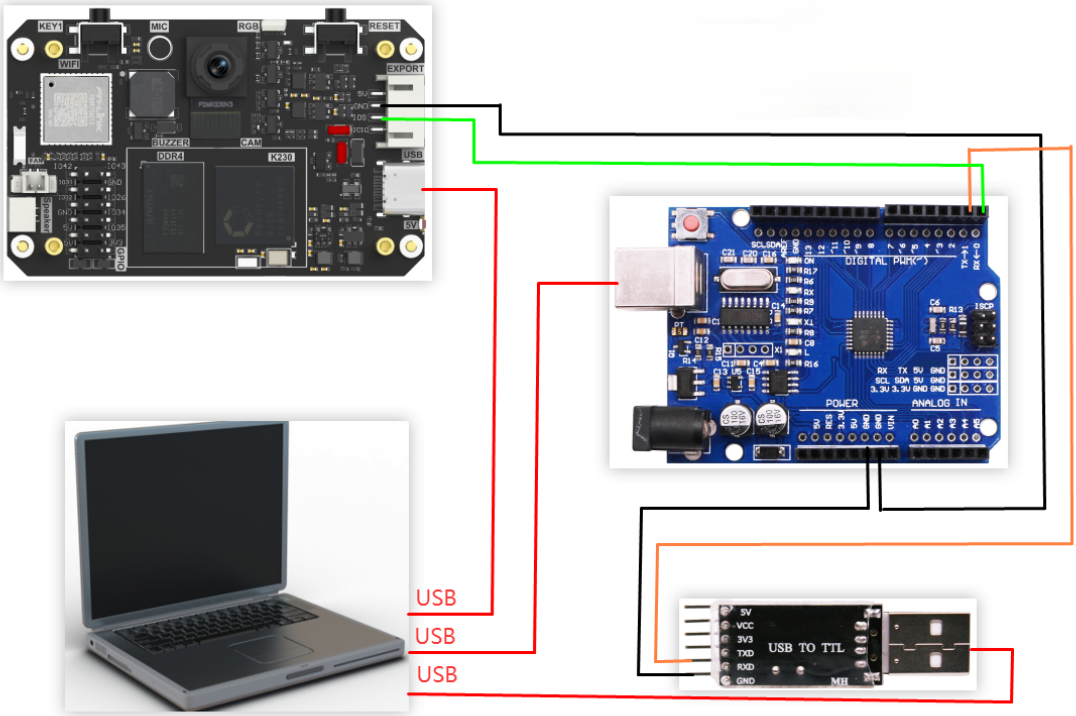
Items needed:

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 == 2)
        {
```

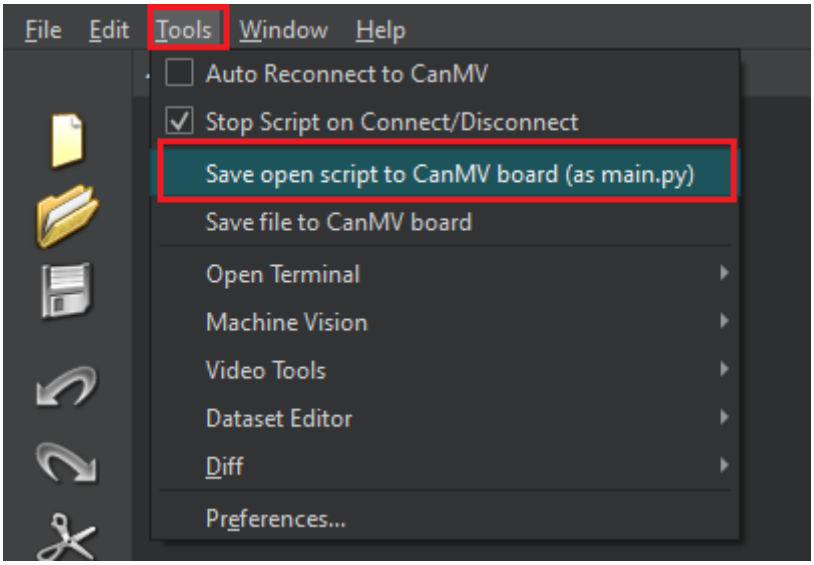
```
            memcpy(msg, (char*)data_buf+field_index[i]+1, values[0]-
field_index[i]-2);
        }
        else
        {
            values[i] = Pto_Char_To_Int((char*)data_buf+field_index[i]+1);
        }
    }

    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;
    }

    Serial.print("gesture:'");
    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.

- gesture: msg is the gesture information, up 'UP', down 'DOWN', left 'LEFT', right 'RIGHT', middle 'MIDDLE'

4. Experimental Phenomenon

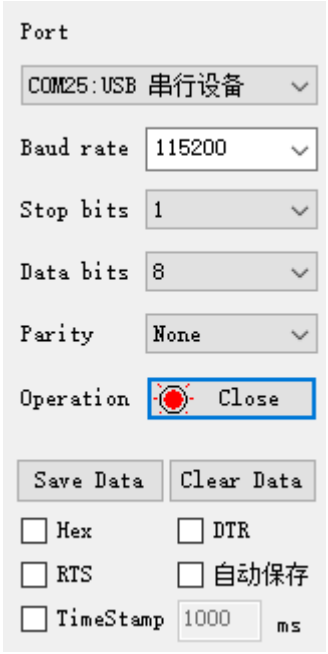
1. After connecting the cables, the k230 visual module runs offline
   After K230 is connected to Canmv IDE, open the corresponding program, click [Save open script to CanMV board (as main.py)] on the toolbar, and then restart K230.



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**)


```
#include "yb_protocol.h"

#define K230Serial  Serial

void setup()
{
  K230Serial.begin(115200);
  Pto_Clear_CMD_Flag();
}

void loop()
{

  while(K230Serial.available())
  {
    Pto_Data_Receive(K230Serial.read());
    Pto_Loop();
   }

}
```

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



4. When the K230 camera recognizes a palm and the palm makes a corresponding gesture, the serial port assistant will print out the information transmitted by K230 to Arduino.

- gesture: is the gesture information, up 'UP', down 'DOWN', left 'LEFT', right 'RIGHT', middle 'MIDDLE'

  As shown in the figure below

```
[2025-04-30 12:02:09.936]# RECV ASCII>
gesture:'DOWN'

[2025-04-30 12:02:13.937]# RECV ASCII>
gesture:'UP'

[2025-04-30 12:02:19.139]# RECV ASCII>
gesture:'RIGHT'

[2025-04-30 12:02:21.556]# RECV ASCII>
gesture:'RIGHT'

[2025-04-30 12:02:22.700]# RECV ASCII>
gesture:'RIGHT'
```