arduino_k230 target tracking

k230 and arduino communication

1. Experimental Prerequisites

This tutorial uses Arduino, and the corresponding routine path is [14.export\arduino-K230\15.Arduino_k230_nano_tracker].

K230 needs to run the [14.export\CanmvIDE-K230\15.nano_tracker.py] program to start the experiment. It is recommended to download it as an offline program.

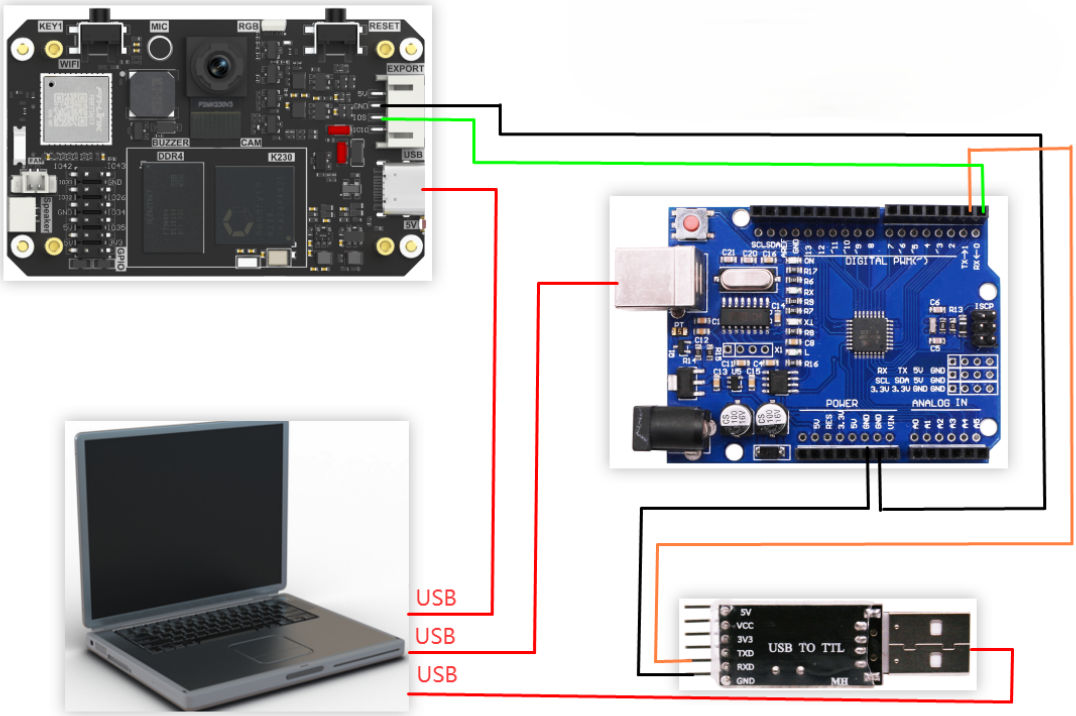
Items needed:

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};
    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];
```

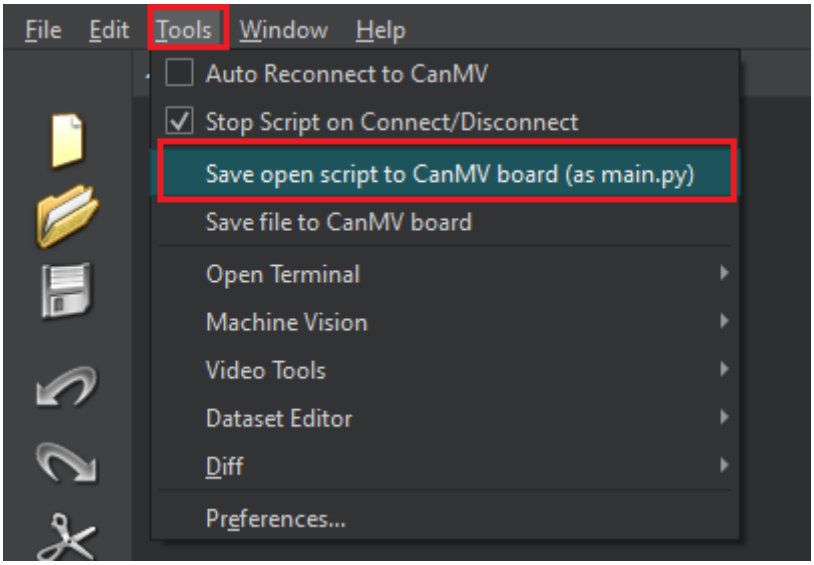
```
    if (pto_len != num)
    {
        Serial.print("pto_len error:");
    Serial.print(pto_len);
    Serial.print(" , data_len:");
    Serial.println(num);
        return;
    }
    uint8_t pto_id = values[1];
    if (pto_id != PTO_FUNC_ID)
    {
        Serial.print("pto_id error:");
    Serial.print(pto_id);
    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("tracker:x:");
    Serial.print(x);
    Serial.print(", y:");
    Serial.print(y);
    Serial.print(", w:");
    Serial.print(w);
    Serial.print(", h:");
    Serial.println(h);

}
```

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

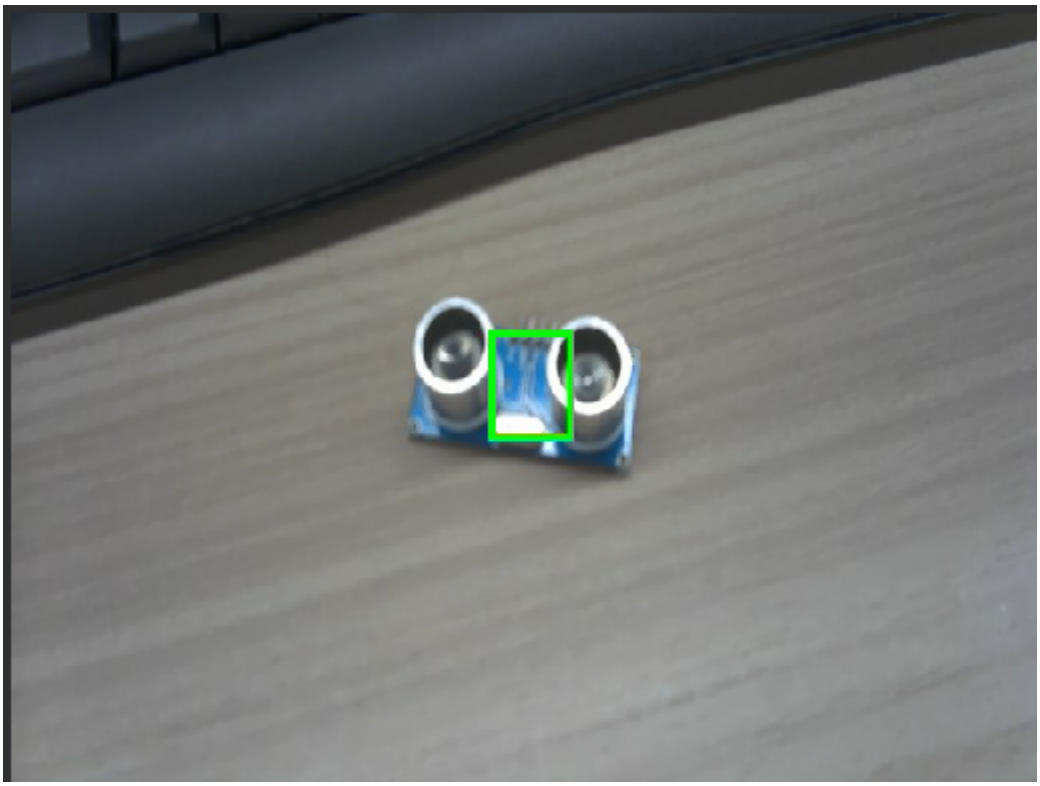
- x: is the horizontal coordinate of the upper left corner of the identified box
- y: is the vertical coordinate of the upper left corner of the identified box
- w: is the width of the identified box
- h: is the length of the identified box

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.

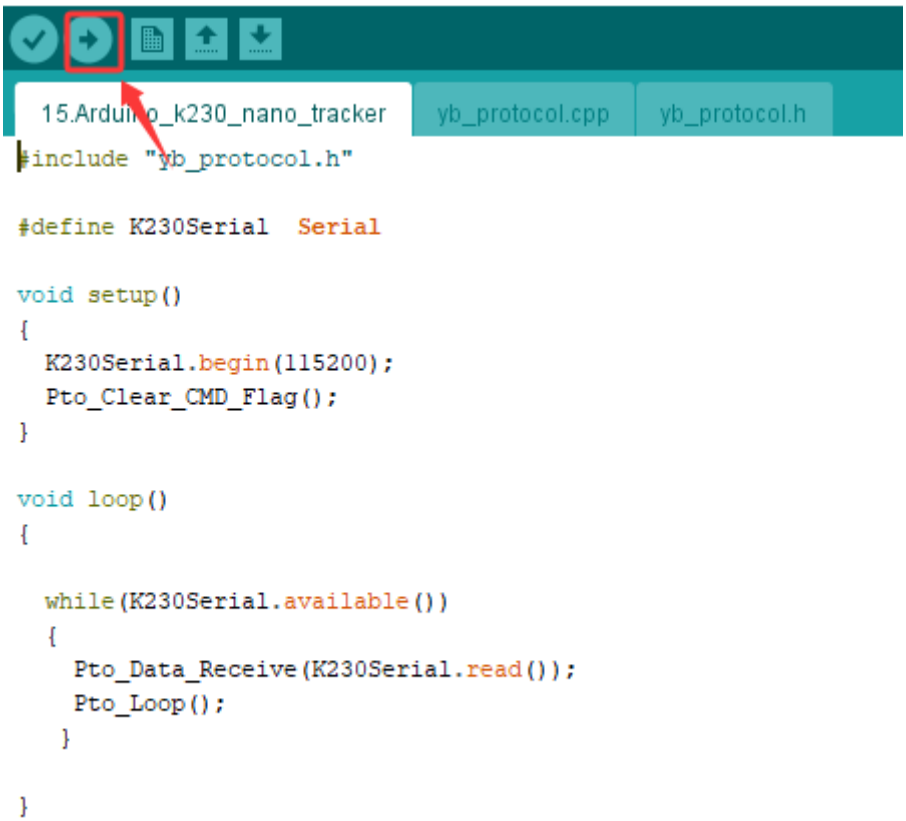


When K230 is turned on, a green box will appear on the screen. Please aim the green box at the target to be tracked (using ultrasound as an example here), wait for a few seconds, and when the green box turns red, it means the target has been detected. At this time, move the target object and you can see that the red box will move with it.



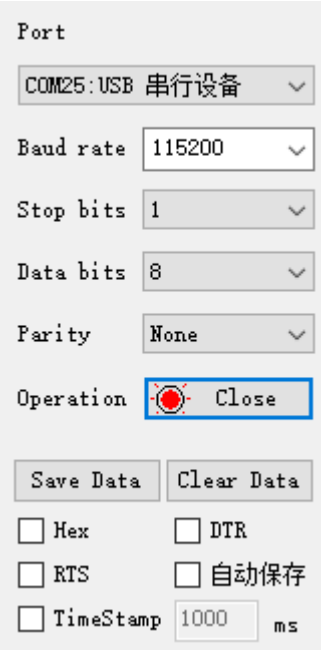
Note: Please do not move the object to be tracked quickly, otherwise it will not be tracked correctly.

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 the target object, 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

As shown in the figure below

```
[2025-04-30 12:13:26.935]# RECV ASCII>
tracker:x:231, y:161, w:34, h:81

[2025-04-30 12:13:26.983]# RECV ASCII>
tracker:x:233, y:164, w:34, h:80

[2025-04-30 12:13:27.140]# RECV ASCII>
tracker:x:234, y:168, w:34, h:80
tracker:x:235, y:172, w:34, h:80
tracker:x:237, y:174, w:34, h:80

[2025-04-30 12:13:27.299]# RECV ASCII>
tracker:x:236, y:176, w:34, h:80
tracker:x:236, y:179, w:34, h:80
tracker:x:237, y:179, w:34, h:80

[2025-04-30 12:13:27.347]# RECV ASCII>
tracker:x:239, y:180, w:34, h:78
```