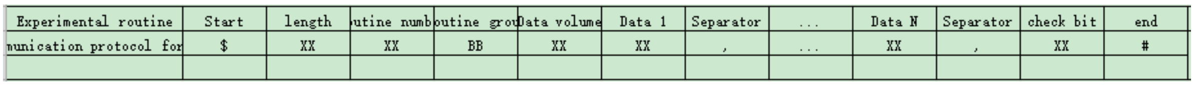
1. k210 and stm32_ Arduino communication

1.Communication protocol description

1.protocol definition



Experimental routine	Start	length	utine numb	outine grou	Data volume	Data 1	Separator	...	Data N	Separator	check bit	end
nunication protocol for	$	XX	XX	BB	XX	XX	,	...	XX	,	XX	#

analysis：

protocol definition	analysis
$	Start character
length	The number of all characters from the start symbol to the end symbol
Routine number	Two bytes, corresponding to routine ID number, with zeros added before values less than 10
Routine group	Two bytes, default to BB
Data volume	Number of data below
data	Data, separated by commas (,) after data, has as many commas as there are
check bit	Add the bytes of all characters from the start symbol to the end symbol and then subtract from 256
#	End

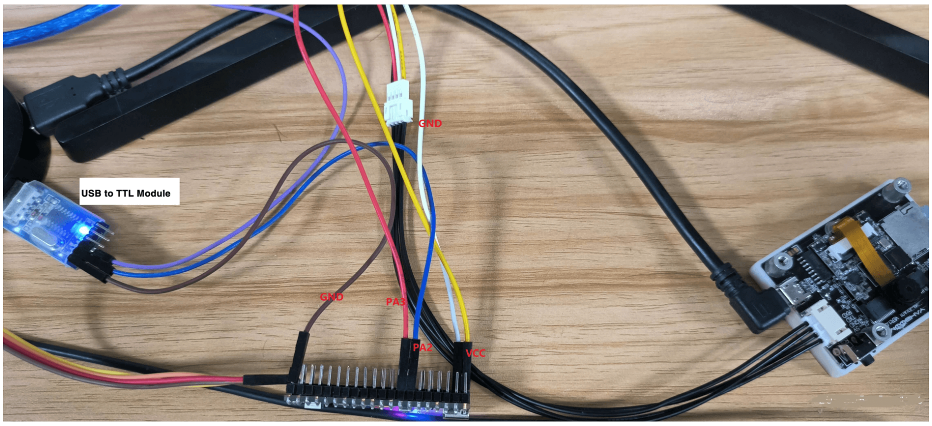
2.K210 and STm32 communication

2.1 Experimental premises

This tutorial uses stm32C8T6, and k210 requires running the program in **K210-AI (stm32_pico_arduino)** to start the experiment
stm32 *1
K210 perspective module *1(Must have an SD card (with a model with AI inside) and a camera)
USB to TTL module*1

2.2 Experimental wiring

stm32	USB to TTL module
PA2	RXD
GND	GND

STM32	K210 perspective module
PA3	TXD
GND	GND
VCC	5V
Wiring as shown in the diagram：	
	

This type of wiring is not required for the RXD of k210 and the TXD of USB to TTL, as it was not used in the experiment。

2.3 According to the encapsulation of the protocol, the code for obtaining the main data of stm32 is as follows：

```
void deal_recvmsg(void)
{

    u8 index,data_i=0;//Data Index
    u8 eg_num = buf_msg[1];//Routine number
//  u8 ed_group = buf_msg[2];//Routine group
    u8 number = buf_msg[3];//Data volume (including commas)
    u8 i_duo = 0;
    //buf_len = buf_msg[0];//len

    if(r_index!=buf_len)//Length does not match
    {
        buf_len = 0;
        return ;
    }


    for(index = 0 ;index<number;index++)
    {
        if(buf_msg[4+index] == 0x2c && i_duo ==0)//Comma, but there will not be two consecutive commas
        {
            i_duo = 1;
            continue;
        }
        data[data_i++]=buf_msg[4+index];//The fifth digit is the start of valid data
        i_duo =0;
    }

    buf_crc = 0;//Check bit clear
    r_index = 0;

    //USART2_Send_ArrayU8((uint8_t*)data,strlen(data));
    memset(buf_msg,0,sizeof(buf_msg));//Clear old data
    deal_data(eg_num);

}
```

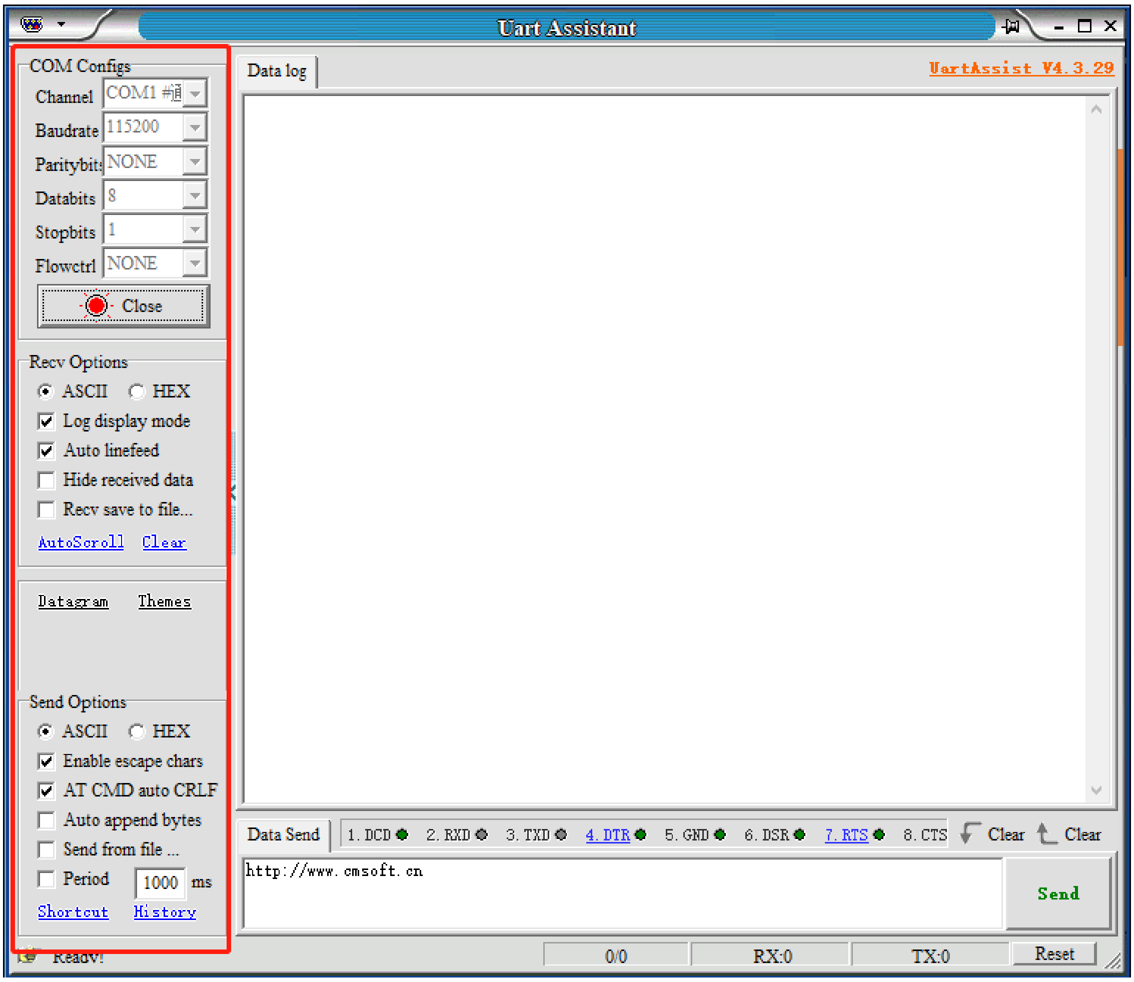
According to the code, it can be seen that:

- Deal_ Recvmsg(): It is the function that is correctly called based on the previous step's verification code
- Deal_ Recvmsg(): Retrieves valid data based on the protocol's upper data volume and saves it in an array, transferring the array and routine numbers to deal_ Data() function for data parsing
- Interested parties can watch the code of the project themselves

2.4 experimental phenomena

1. After connecting the cable, the K210 perspective module can be run offline or connected to the camv ide, but some experiments need to be run offline. Please refer to the appendix for the methods of offline operation

2. Set the serial port assistant to the interface shown in the figure



COM Configs

Channel: COM1 #通

Baudrate: 115200

Paritybits: NONE

Databits: 8

Stopbits: 1

Flowctrl: NONE

Close

Recv Options

● ASCII ○ HEX

☑ Log display mode

☑ Auto linefeed

□ Hide received data

□ Recv save to file...

AutoScroll Clear

Datagram Themes

Send Options

● ASCII ○ HEX

☑ Enable escape chars

☑ AT CMD auto CRLF

□ Auto append bytes

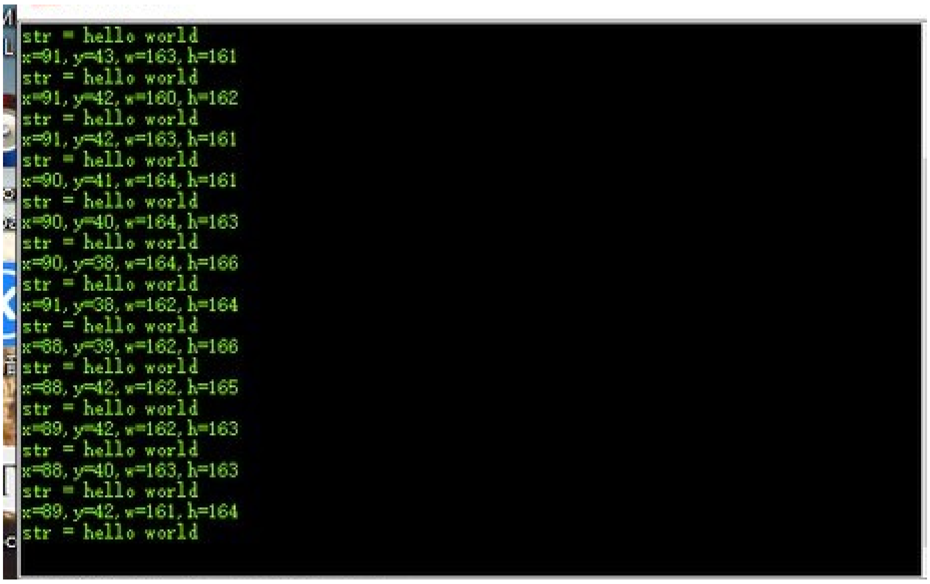
□ Send from file ...

□ Period 1000 ms

Shortcut History

http://www.cmsoft.cn

3. Then run the relevant routine, and the serial assistant will print out the important information transmitted from k210 to stm32. The phenomenon shown in the following figure is the result of QR code recognition



```
str = hello world
x=91, y=43, w=163, h=161
str = hello world
x=91, y=42, w=160, h=162
str = hello world
x=91, y=42, w=163, h=161
str = hello world
x=90, y=41, w=164, h=161
str = hello world
x=90, y=40, w=164, h=163
str = hello world
x=90, y=38, w=164, h=166
str = hello world
x=91, y=38, w=162, h=164
str = hello world
x=88, y=39, w=162, h=166
str = hello world
x=88, y=42, w=162, h=165
str = hello world
x=89, y=42, w=162, h=163
str = hello world
x=88, y=40, w=163, h=163
str = hello world
x=89, y=42, w=161, h=164
str = hello world
```

Xy is the coordinate, wh is the width and length, and str is the character data recognized by the QR code

3.K210 and Arduino communication

3.1 Experimental premises

This tutorial uses arduino, and K210 requires running the program in * * K210-AI (stm32_pico_arduino) * * to start the experiment

arduino *1

K210 perspective module * 1 (requires SD card (with AI model inside) and camera)

USB to TTL module * 1

3.2 Experimental wiring

arduino	USB to TTL module
TXD	RXD
GND	GND

arduino	K210 perspective module
RXD	TXD
GND	GND
VCC	5V
Wiring as shown in the diagram:	

2.3 Encapsulation according to protocol，The code for parsing the main data of Arduino is as follows:

```
void deal_data(u8 egnum)//The incoming value is the routine number
{
    //......
  switch(egnum)
  {

    case 1:
    case 5:
    case 6:
      x = data[1]<<8 | data[0];
      y = data[3]<<8 | data[2];
      w = data[5]<<8 | data[4];
      h = data[7]<<8 | data[6];
      sprintf(buff_com,"x=%d,y=%d,w=%d,h=%d\r\n",x,y,w,h);
      K210Serial.print(buff_com);
      break;


    case 2:
    case 3:
      x = data[1]<<8 | data[0];
      y = data[3]<<8 | data[2];
      w = data[5]<<8 | data[4];
      h = data[7]<<8 | data[6];
      sprintf(buff_com,"x=%d,y=%d,w=%d,h=%d\r\n",x,y,w,h);
      K210Serial.print(buff_com);
      while(*(data+8+icopy)!='\0')
      {
        msg[icopy] = *(data+8+icopy);
        icopy ++;
      }
      sprintf(buff_com,"str = %s\r\n",msg);
      K210Serial.print(buff_com);
      break;


    case 4:
      x = data[1]<<8 | data[0];
      y = data[3]<<8 | data[2];
      w = data[5]<<8 | data[4];
      h = data[7]<<8 | data[6];
      id =data[8]<<8 | data[9];
      sprintf(buff_com,"x=%d,y=%d,w=%d,h=%d\r\n",x,y,w,h);
      K210Serial.print(buff_com);

      while(*(data+10+icopy)!='\0')
      {
        msg[icopy] = *(data+10+icopy);
        icopy ++;
      }
      sprintf(buff_com,"id = %c%c,str = %s\r\n",(id>>8),id,msg);
      K210Serial.print(buff_com);
      break;

    case 7:
    case 8:
      x = data[1]<<8 | data[0];
      y = data[3]<<8 | data[2];
      w = data[5]<<8 | data[4];
      h = data[7]<<8 | data[6];
      id =data[8];
      sprintf(buff_com,"x=%d,y=%d,w=%d,h=%d\r\n",x,y,w,h);
      K210Serial.print(buff_com);
      memset(buff_com,0,sizeof(buff_com));
      if(id == 'Y' || id == 'y')
      {
        sprintf(buff_com,"Yes\r\n");
        K210Serial.print(buff_com);
      }
      else
      {
        sprintf(buff_com,"NO\r\n");
        K210Serial.print(buff_com);
      }

      break;
    case 9:
      x = data[1]<<8 | data[0];
      y = data[3]<<8 | data[2];
      w = data[5]<<8 | data[4];
      h = data[7]<<8 | data[6];
      sprintf(buff_com,"x=%d,y=%d,w=%d,h=%d\r\n",x,y,w,h);
      K210Serial.print(buff_com);
      while(*(data+8+icopy)!='\0')
      {
        msg[icopy] = *(data+8+icopy);
        icopy ++;
      }
      sprintf(buff_com,"id = %s\r\n",msg);
      K210Serial.print(buff_com);
      break;

    case 10:
    case 11:
      id = data[0];
      sprintf(buff_com,"id = %c\r\n",id);
      K210Serial.print(buff_com);
  }
  //......
}
```

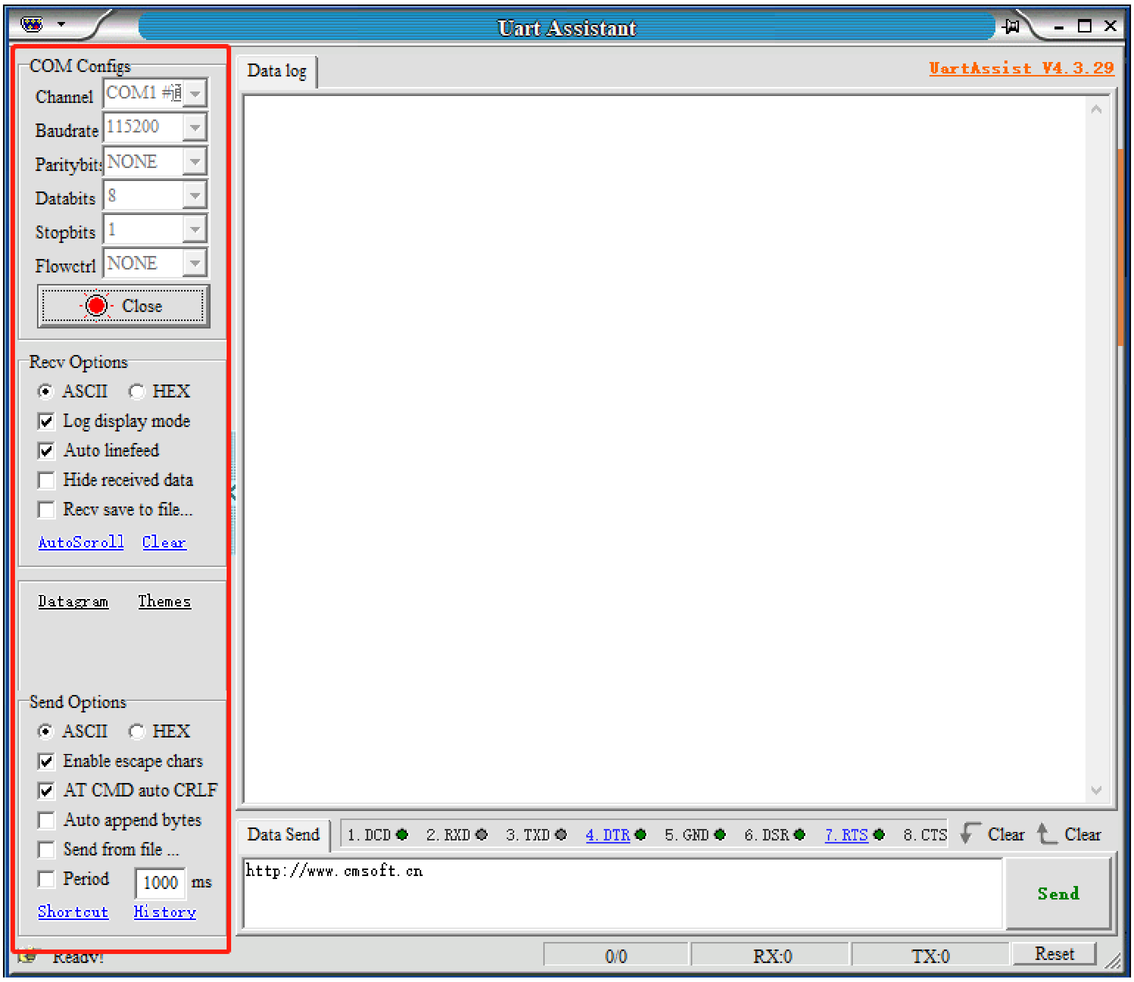
As can be seen from the code

- Deal_ Data(): Protocol related parsing is performed based on the number of routines. The content sent by each K210 routine is different, and the resulting data is also different
- Parse the data based on the number, and then Arduino sends the parsed data to the serial assistant.
- Interested parties can browse the engineering source code of Arduino on their own

3.4 experimental phenomena

1. After connecting the cable, the K210 perspective module can be run offline or connected to the camv ide, but some experiments need to be run offline. Please refer to the appendix for the methods of offline operation

2. Set the serial port assistant to the interface shown in the figure



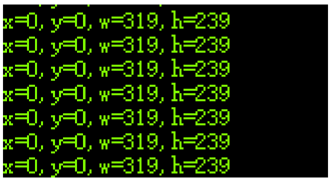
COM Configs
Channel: COM1 #通
Baudrate: 115200
Paritybits: NONE
Databits: 8
Stopbits: 1
Flowctrl: NONE

Recv Options
◉ ASCII ○ HEX
☑ Log display mode
☑ Auto linefeed
□ Hide received data
□ Recv save to file...

Send Options
◉ ASCII ○ HEX
☑ Enable escape chars
☑ AT CMD auto CRLF
□ Auto append bytes
□ Send from file ...
□ Period 1000 ms

```
http://www.cmsoft.cn
```

3. Then run the relevant routines, and the serial assistant will print out the important information transmitted by K210 to Adruino. The phenomenon shown in the following figure is the result of color recognition



```
x=0, y=0, w=319, h=239
x=0, y=0, w=319, h=239
x=0, y=0, w=319, h=239
x=0, y=0, w=319, h=239
x=0, y=0, w=319, h=239
x=0, y=0, w=319, h=239
x=0, y=0, w=319, h=239
```

The value of the size xywh of the recognized color box will be printed in the serial assistant, where xy is the coordinate and wh is the width and length

4.appendix

4.1 K210 offline operation method

- First, put the model needed for recognition on the SD card
  Import model to SD card
- Save the required AI program to the SD card and rename it main.py
- Power on K210 to run the main.py program on the SD card

4.2 Serial Assistant Data Analysis

x:abscissa
y:ordinate
w:width
h:length

id:Identified object labels
str:Identified Content Information