K210 and Pico 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
unication 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

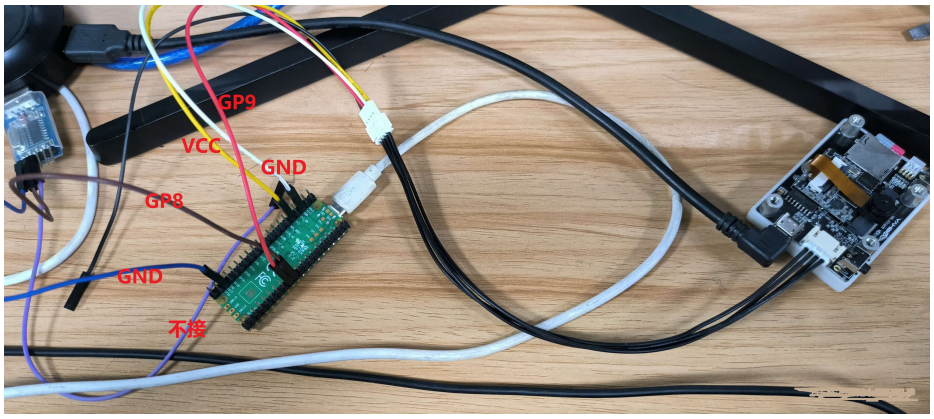
2.K210 and Pico communication

2.1 Experimental premises

This tutorial uses Pico, and K210 requires running the program in * * K210-AI (stm32_pico_arduino) * * to start the experiment
pico *1
K210 perspective module * 1 (requires SD card (with AI model inside) and camera)
USB to TTL module * 1

2.2 Experimental wiring

pico	USB to TTL module
GP8	RXD
GND	GND

pico	K210 perspective module
GP9	TXD
GND	GND
VCC	5V
Wiring as shown in the diagram:	
	

2.3 Main code analysis

```
u1 = UART(1, baudrate=115200, tx=Pin(8), rx=Pin(9), bits=8, parity=None, stop=0)
 # Set baud rate and serial port number

rxx = bytes()
data_buf_str = []#String List
data_buf_int = []#Integer List

data_data = []#Effective data in data volume
data_len = 0 #The length of the package

while True:
    while u1.any() >0:
```

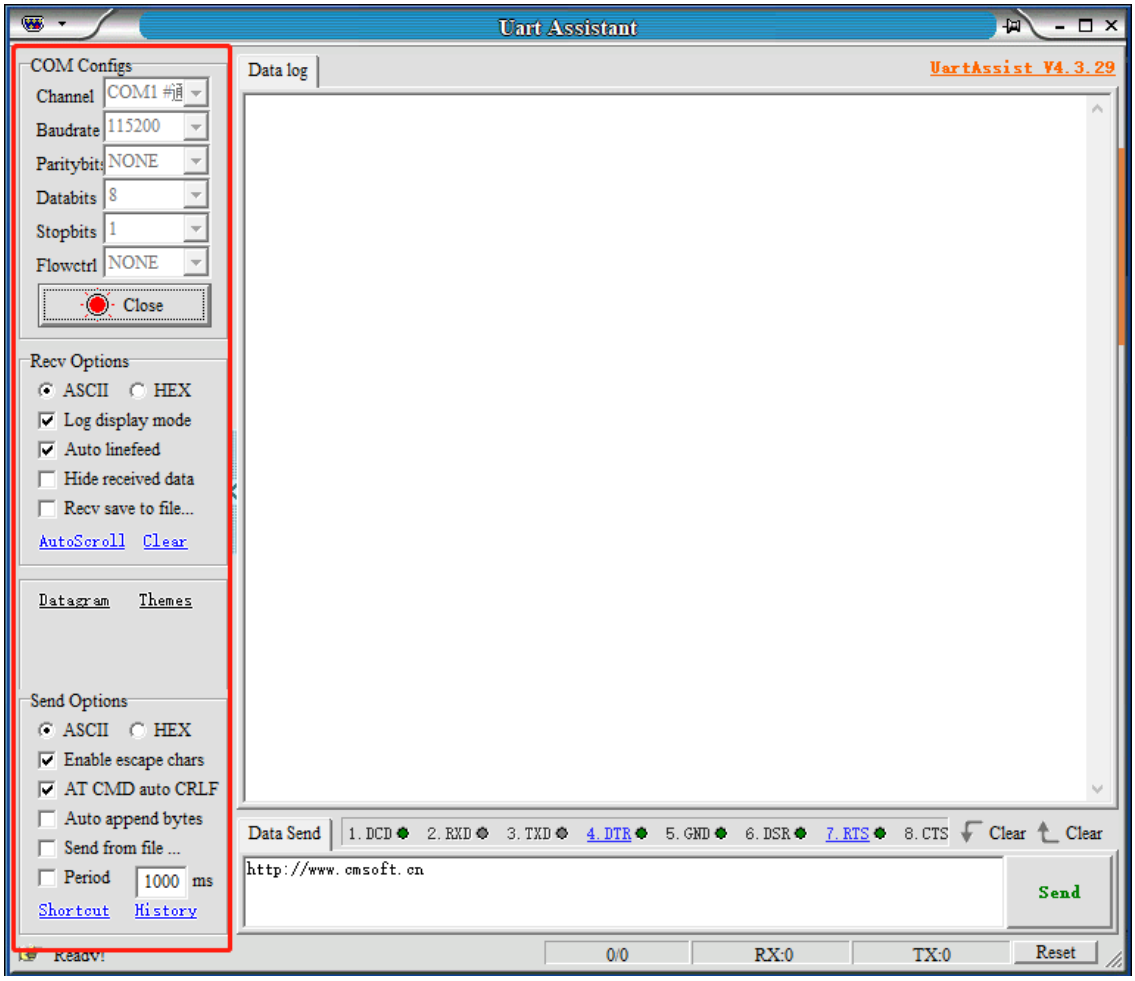
```
        rxx=u1.read()
        data_one = binascii.hexlify(rxx) #Convert string to hexadecimal format
        data_one = data_one.decode('utf-8') #Remove b ''
        #Cut every two characters
        for i in range(0,len(data_one),2):
            data_buf_str.append(data_one[i:i+2])
        for i in range(len(data_buf_str)):
            hint = str_int(data_buf_str[i])
            data_buf_int.append(hint)
        deal_data()
```

From the code, it can be seen that

- This experiment used Pico's GP8 and GP9 pins for serial communication
- Because it is a language in MicroPython, there is no hex() function. Instead, we can use the binascii. hexlify() function to replace the hex function and obtain the * * string * * that receives data and sends it in hexadecimal format.
- Str_ Int(): This function converts a hexadecimal string to an integer hexadecimal, making it easier for subsequent data processing and extracting important data
- Deal_ Data(): It refers to the processing and extraction of data. If you are interested, browse the source code yourself

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



3. Download PICO's Python firmware into PICO and run the USART provided in this tutorial_ For the program K210. py and how to run Pico's Python program, please refer to the Pico related environment building tutorial. This tutorial will not elaborate on it
4. Then k210 runs the relevant routines, and the serial assistant will print out the important information transmitted by k210 to Pico. The phenomenon shown in the following figure is the result of autonomous learning classification AI

The serial port prints out the classified ID information

3.appendix

3.1 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

3.2 Serial Assistant Data Analysis

x: Abscissa
y: Ordinate
w: Width
h: Length
ID: Identified object label
Str: Identified content information