Kmbox-Net development documentation (english translation)
Kmbox-Net 开发文档(英语翻译)

Why was the kmbox-net version made?
为什么要做 kmbox-net 版本?

1. The serial port communication speed of board B is slow. The call speed is too slow. And the communication returns too quickly, causing the box to restart and lose the effect.
2. B 板串口通讯速度慢。通话速度太慢。而且通信返回太快，导致盒子重启，失去效果。
3. The serial port is easy to detect and scan, and the original protocol was public, which caused the B-board stand-alone machine to be targeted by Penguin.
4. 串口容易被检测扫描，原协议是公共的，导致 B 板单机被企鹅盯上。
5. Many AI and DMA do not require onboard macro functions. All it takes is a simple call.
6. 许多 AI 和 DMA 不需要板载宏功能。只需要一个简单的电话。

Kmbox-Net has the following features:
Kmbox-Net 具有以下特点:

1. Very secure; the protocol is not public and cannot be characterized. No need to install drivers. Each box has independent IP, port, and hardware encoding. Use blocking socket communication. Cannot be scanned.
2. 很有安全感；该协议不是公共的，不能被特征化。无需安装驱动程序。每个盒子都有独立的 IP、端口和硬件编码。使用阻塞套接字通信。无法扫描。
3. It has strong stability and will not restart with a blank screen like serial port communication.
4. 稳定性强，不会像串口通讯一样黑屏重启。
5. high speed. 100M network, communication speed is 100 times that of the original B board (compared to 115200 baud rate). The number of calls per second is close to 1,000. It turns out that board B only makes 300 calls per second. If you call it too quickly, it will restart with a white screen.
6. 高速。100M 网络，通信速度是原来 B 板的 100 倍(相比 115200 波特率)。每秒钟的通话次数接近 1000 次。结果 B 板每秒只打 300 个电话。如果你调用它太快，它会重新启动白屏。

7. Automatically defaults to artificial trajectory, and there will be no abnormal keyboard and mouse data. Please refer to the kmNet_mouse_move_auto function.
8. 自动默认为人工轨迹，不会出现键盘鼠标数据异常。请参考 kmNet_mouse_move_auto 函数。
9. No need to adapt the mouse. Highly versatile.
10. 不需要调整鼠标。高度通用。
11. The Device side supports modifying all USB parameters. Support network updates. Everyone has different characteristic values. True one person one firmware model. (TBD)
12. 设备端支持修改所有 USB 参数。支持网络更新。每个人都有不同的特征值。真正的一人一固件模式。(TBD)
13. Friendly UI mode. It’s clear what went wrong. Foolish operation.
14. 友好的 UI 模式。很清楚哪里出了问题。愚蠢的操作。
15. Supports all functions of B board except onboard script.
16. 支持除板载脚本之外的所有 B 板功能。
17. Support monitoring physical keyboard and mouse. Block physical keyboard and mouse functions. Convenient to write software.
18. 支持监控物理键盘和鼠标。阻止物理键盘和鼠标功能。方便写软件。
19. Supports physical keyboard and mouse learning copy function. (TBD)
20. 支持物理键盘鼠标学习复制功能。(TBD)
21. Supports kvm keyboard and mouse switching function. (TBD)
22. 支持 kvm 键盘鼠标切换功能。(TBD)

Hardware

五金器具

Kmbox-Net contains 4 USB ports. (The picture is just a sample, everything is subject to actual release).

Kmbox-Net 包含 4 个 USB 端口。(图片只是样本，一切以实际发布为准)。

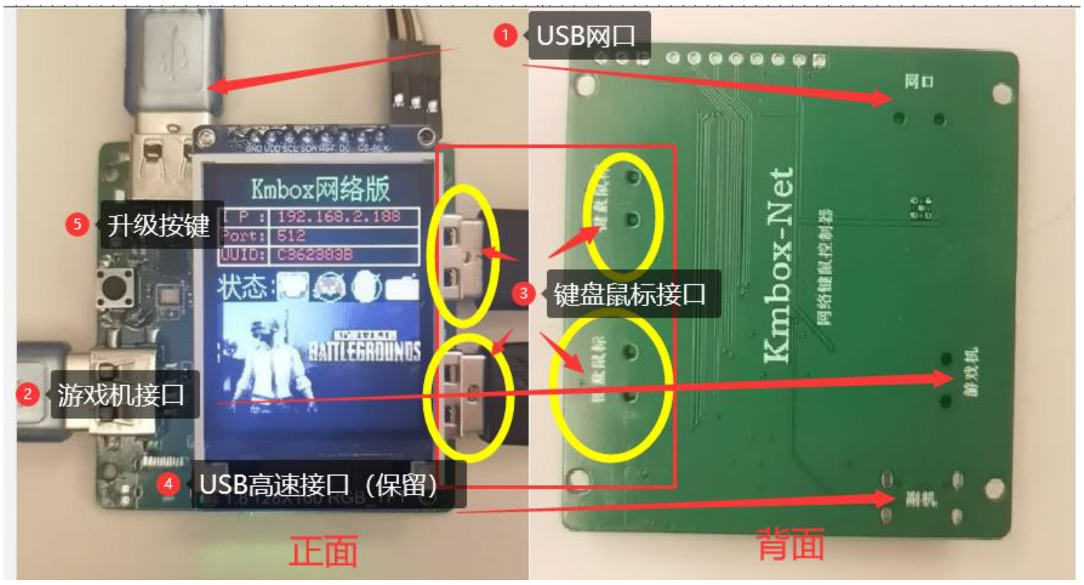


Image labels translated: 1. USB Network port (usb ethernet I think) 2. USB output to pc 3. USB input from HID 4. USB C high speed interface 5. Upgrade button (assuming this is the reset/reboot button)
翻译的图像标签:1。usb 网口(USB 以太网我觉得)2。USB 输出到 pc 3。来自 HID 4 的 USB 输入。USB C 高速接口 5。升级按钮(假设这是重置/重启按钮)

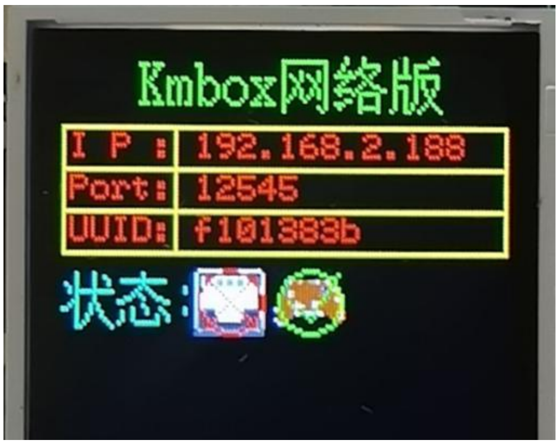
As shown in the figure, kmbox-Net contains the following physical interfaces:
如图所示，kmbox-Net 包含以下物理接口:

1. One USB network port. After connecting to the computer, there will be a USB network card. Used to communicate with the box and transmit control instructions.
2. 一个 USB 网络端口。连接电脑后，会有一个 USB 网卡。用于与机顶盒通信并传输控制指令。
3. One USB game console interface. After connecting to the computer, it will be enumerated as a standard keyboard and mouse. Used to control gaming computer keyboard and mouse.
4. 一个 USB 游戏控制台接口。连接到计算机后，它将被枚举为标准键盘和鼠标。用于控制游戏电脑的键盘和鼠标。
5. Two USB keyboard and mouse interfaces. Used to connect a keyboard or mouse. Used to control gaming computers.
6. 两个 USB 键盘和鼠标接口。用于连接键盘或鼠标。用于控制游戏电脑。
7. One USB high-speed interface (typeC port). Can be used for operations such as dual-machine synchronization. (Default is not welding)
8. 一个 USB 高速接口(typeC 端口)。可用于双机同步等操作。(默认为不焊接)
9. An upgrade button. for firmware updates.
10. 升级按钮。用于固件更新。

How to wire

如何接线

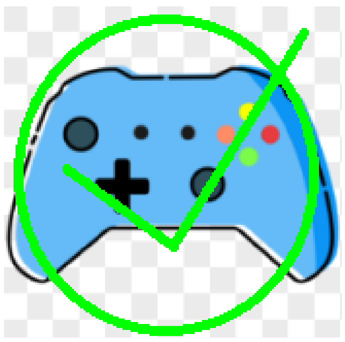
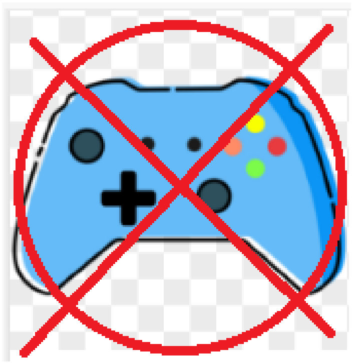
Use the blue USB cable to connect the USB HID output port of the box to the computer to power on the box. At this time, the box display will look like this
使用蓝色 USB 电缆将机器的 USB HID 输出端口连接到计算机，以打开机器电源。这时，框显示会是这样的

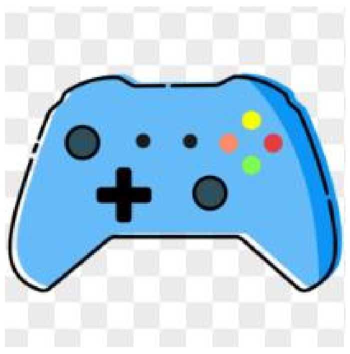
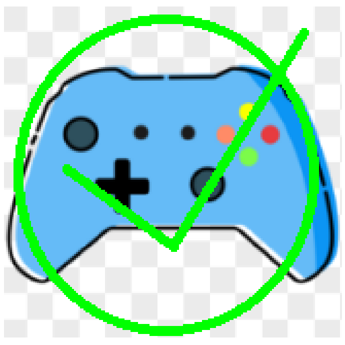
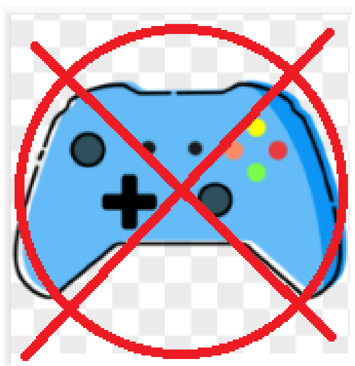

Kmbox网络版

I P :	192.168.2.188
Port:	12545
UUID:	f101383b

状态:

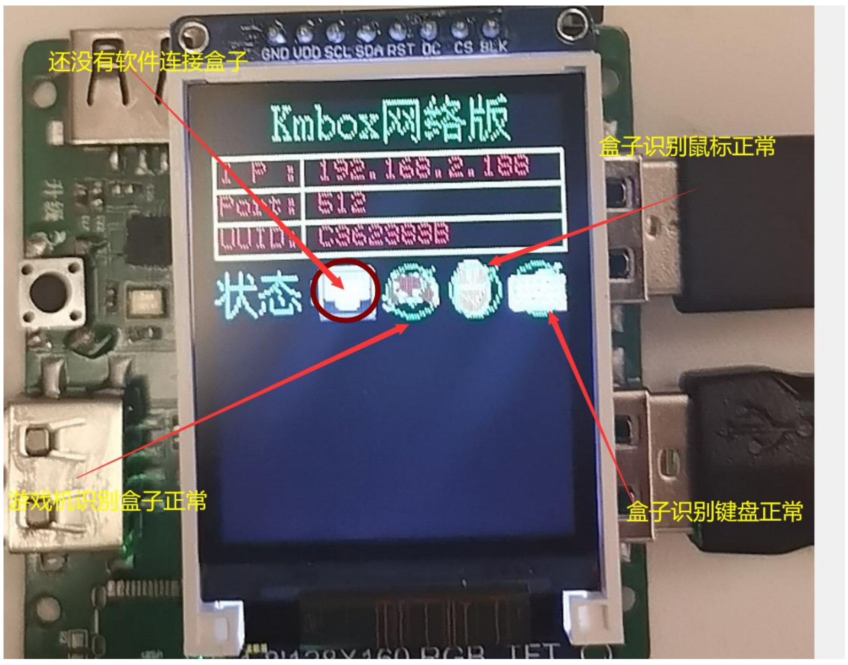
The second icon in the status bar is a game controller. Represents a game console. If you look carefully, you will find that the second icon is crossed first and then checked after powering on. The meaning of this icon is as follows:
状态栏中的第二个图标是游戏控制器。表示游戏控制台。如果你仔细看，会发现第二个图标是先划十字，通电后再检查。该图标的含义如下:

Waiting for connection	The game console recognizes the box (connected)	The console disconnects from the box (disconnected)
		
Generally, the power-on enumeration process occurs.	Normal recognition box keyboard and mouse device appears	Appears when the host is sleeping or disconnected
等待连接	游戏控制台识别盒子(已连接)	控制台与机箱断开连接(断开)

		
通常，会发生通电枚举过程。	正常识别框键盘和鼠标设备出现	当主机睡眠或断开连接时出现

Ps: If a cross occurs, please re-plug the USB cable of the game console.
Ps:如果出现交叉，请重新插上游戏机的 USB 线。

Connect the gaming keyboard and mouse to the box. If the recognition is normal. The box display shows as follows:
将游戏键盘和鼠标连接到盒子上。如果识别正常。该框显示如下:

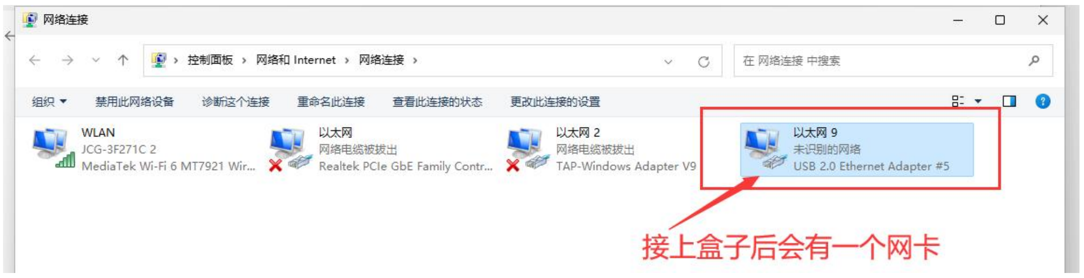


Translated labels: top left: There is no software connection yet top right: The box recognizes the mouse normally bottom left: The box recognizes the output pc normally bottom right: The box recognizes the keyboard normally
翻译后的标签:左上:没有软件连接右上:机器正常识别鼠标左下角:机器正常识别输出 pc 右下角:机器正常识别键盘

If the connected keyboard and mouse can control the game console normally, then the box is connected and OK.
如果连接的键盘鼠标能正常控制游戏主机，那么盒子就连接好了，OK。

Software
软件

The software can connect the box through a USB network card and control all keyboard and mouse data. Shield real physical buttons. Detect whether a physical button is pressed. Move the mouse. Click on the keyboard and other operations. The box provides an API for you to fully control the physical keyboard and mouse. Before using the software, you need to connect the network port USB cable to the computer.
软件可以通过 USB 网卡连接盒子，控制所有键盘和鼠标数据。屏蔽真实物理按键。检测物理按钮是否被按下。移动鼠标。点击键盘等操作。盒子提供了一个 API 让你完全控制物理键盘和鼠标。在使用软件之前，您需要将网络端口 USB 电缆连接到计算机。



WLAN
JCG-3F271C 2
MediaTek Wi-Fi 6 MT7921 Wir...

以太网
网络电缆被拔出
Realtek PCIe GbE Family Contr...

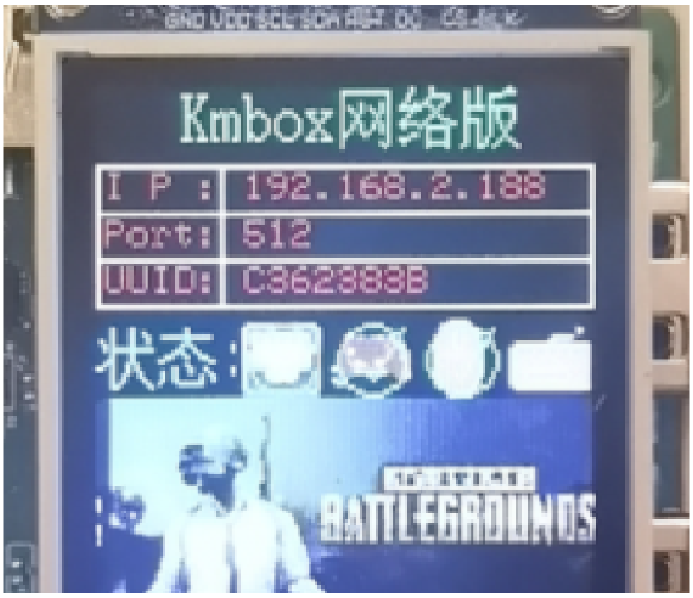
以太网 2
网络电缆被拔出
TAP-Windows Adapter V9

以太网 9
未识别的网络
USB 2.0 Ethernet Adapter #5

Caption translated: After connecting the box there will be a network card in your file explorer.
连接盒子后，你的文件浏览器中会有一个网卡。

Bagelnewt note: you can find this screen by searching “network” in the start menu search box.
Bagelnewt 注:您可以通过在开始菜单搜索框中搜索“网络”来找到此屏幕。

When you connect the box for the first time, a CD-ROM device will automatically eject. After opening and double-clicking, the USB network card will appear. After the network card appears, we need to configure the network card so that the IP of the network card and the box IP are in the same network segment. Only in this way can the computer successfully communicate with the box. As shown in the picture below, the box IP is 192.168.2.188 (PS each box IP is different. Everything is subject to the IP on the display)
当您第一次连接包装盒时，CD-ROM 设备会自动弹出。打开双击后，会出现 USB 网卡。网卡出现后，我们需要配置网卡，让网卡的 IP 和盒子 IP 在同一个网段。只有这样，计算机才能与盒子成功通信。如下图所示，盒子 IP 是 192.168.2.188 (PS 每个盒子 IP 都不一样。一切以显示器上的 IP 为准)


Kmbox网络版

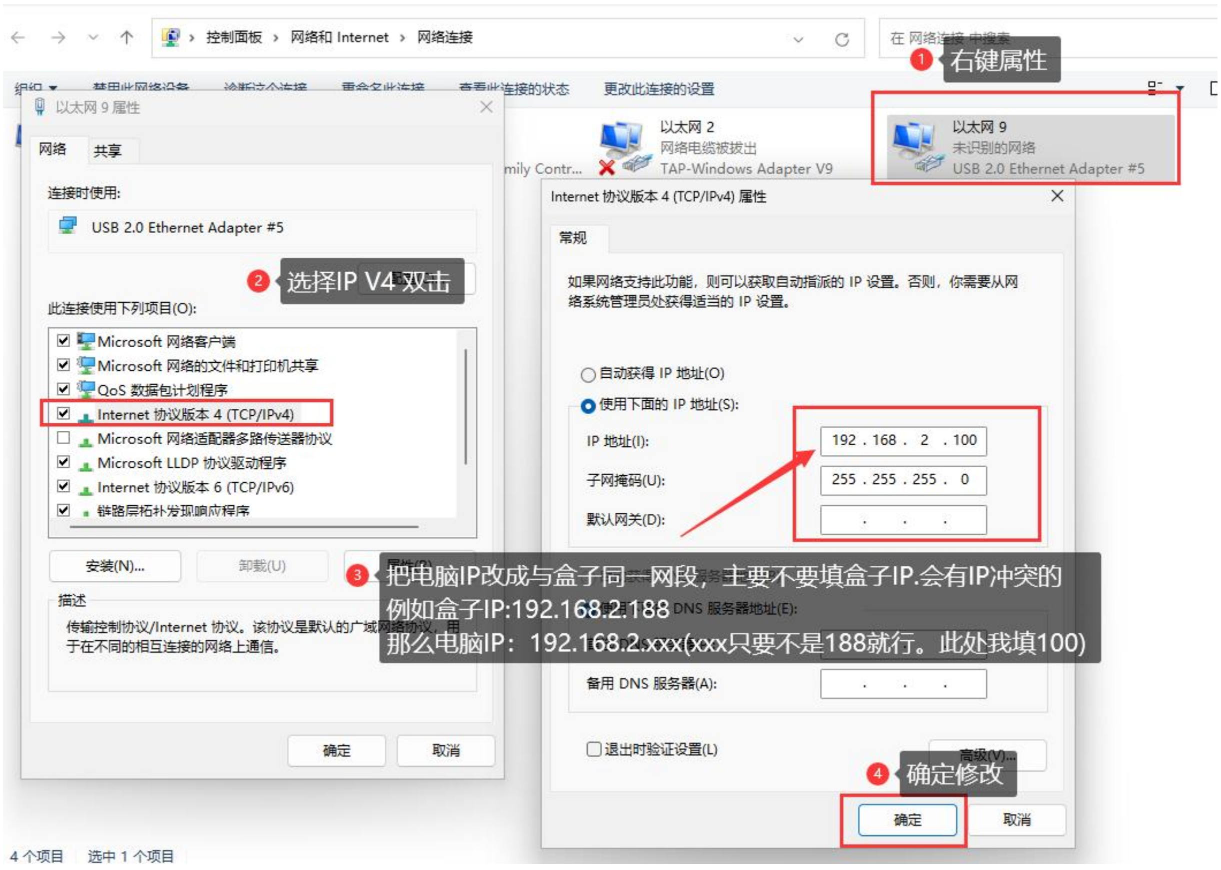
I P : 192.168.2.188

Port: 512

UUID: C362383B

状态:

Note: The network port USB cable must be plugged into and connected to the computer. Because the plug-in communicates with the box through the network port.
注:网络端口 USB 电缆必须插入并连接到计算机。因为插件是通过网口和盒子通信的。


控制面板 › 网络和 Internet › 网络连接

① 右键属性

以太网 2 网络电缆被拔出 TAP-Windows Adapter V9

以太网 9 未识别的网络 USB 2.0 Ethernet Adapter #5

以太网 9 属性

网络 共享

连接时使用:

USB 2.0 Ethernet Adapter #5

② 选择IP V4 双击

此连接使用下列项目(O):

- ☑ Microsoft 网络客户端
- ☑ Microsoft 网络的文件和打印机共享
- ☑ QoS 数据包计划程序
- ☑ Internet 协议版本 4 (TCP/IPv4)
- ☐ Microsoft 网络适配器多路传送器协议
- ☑ Microsoft LLDP 协议驱动程序
- ☑ Internet 协议版本 6 (TCP/IPv6)
- ☑ 链路层拓扑发现响应程序

描述

传输控制协议/Internet 协议。该协议是默认的广域网络协议，用于在不同的相互连接的网络上通信。

Internet 协议版本 4 (TCP/IPv4) 属性

常规

如果网络支持此功能，则可以获取自动指派的 IP 设置。否则，你需要从网络系统管理员处获得适当的 IP 设置。

○ 自动获得 IP 地址(O)

● 使用下面的 IP 地址(S):

IP 地址(I): 192 . 168 . 2 . 100

子网掩码(U): 255 . 255 . 255 . 0

默认网关(D): . . .

备用 DNS 服务器(A): . . .

☐ 退出时验证设置(L)

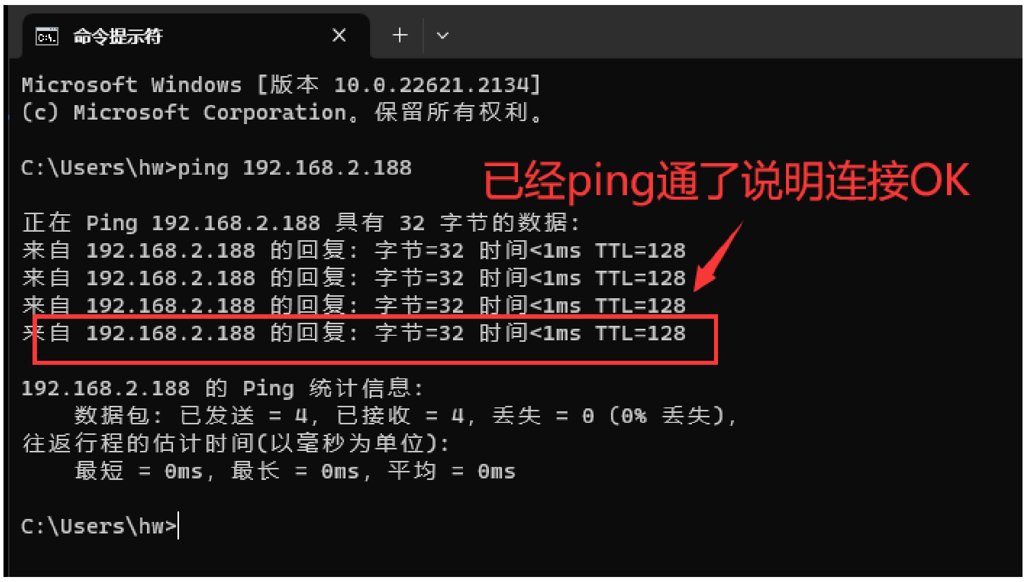
③ 把电脑IP改成与盒子同一网段，主要不要填盒子IP.会有IP冲突的
例如盒子IP:192.168.2.188
那么电脑IP： 192.168.2.xxx(xxx只要不是188就行。此处我填100)

④ 确定修改

Labels read: 1. Right click properties 2. Select ipv4 and double click 3. Change the computer IP to the same ip as is stated on the box. There will be conflicts. For example, the box ip may be

192.168.2.188 but the computer’s ip may be 192.168.2.xxx (can be anything but 188, I put 100)
标签上写着:1。右键单击属性 2。选择 ipv4 并双击 3。将计算机 ip 地址更改为包装盒上标明的 IP 地址。会有冲突。例如，盒子的 ip 可能是 192.168.2.188，但计算机的 ip 可能是 192.168.2.xxx(可以是除 188 以外的任何值，我放 100)

After modifying the host IP. We can ping the box. Used to verify whether the host and box network are connected:
修改主机 IP 后。我们可以 ping 这个盒子。用于验证主机和机箱网络是否连接:

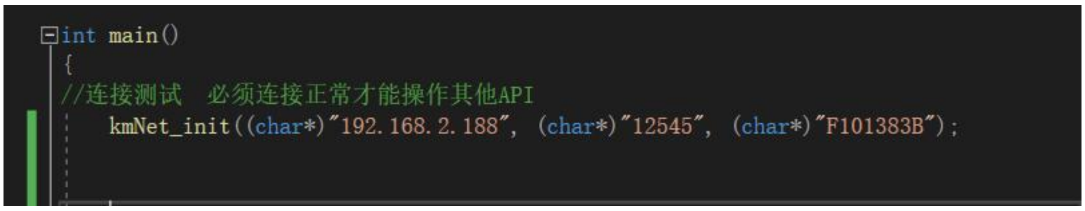


```
Microsoft Windows [版本 10.0.22621.2134]
(c) Microsoft Corporation。保留所有权利。

C:\Users\hw>ping 192.168.2.188

正在 Ping 192.168.2.188 具有 32 字节的数据:
来自 192.168.2.188 的回复: 字节=32 时间<1ms TTL=128
来自 192.168.2.188 的回复: 字节=32 时间<1ms TTL=128
来自 192.168.2.188 的回复: 字节=32 时间<1ms TTL=128
来自 192.168.2.188 的回复: 字节=32 时间<1ms TTL=128

192.168.2.188 的 Ping 统计信息:
    数据包: 已发送 = 4，已接收 = 4，丢失 = 0 (0% 丢失)，
往返行程的估计时间(以毫秒为单位):
    最短 = 0ms，最长 = 0ms，平均 = 0ms

C:\Users\hw>
```

Label reads: It has been pinged, indicating that the connection is okay.
标签上写着:已经 pinged 通，表示连接没问题。
Next, you can start the software call.
接下来，您可以开始软件调用。

Connection box kmNet_init
接线盒 kmNet_init

First the connection box must be called：`int kmNet_init(char* ip, char* port, char* uuid);//ok`
首先必须调用连接盒:int kmNet_init(char* ip，char* port，char * uuid)；//好的

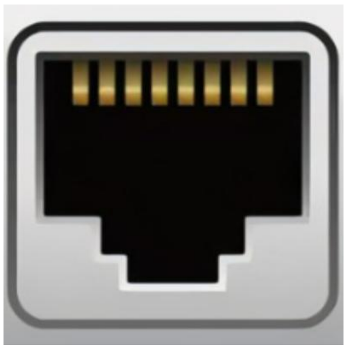
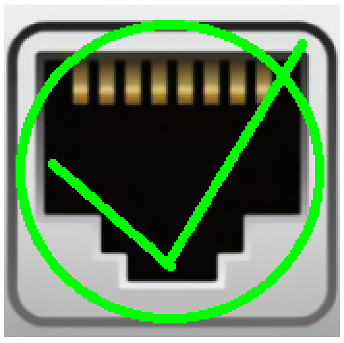


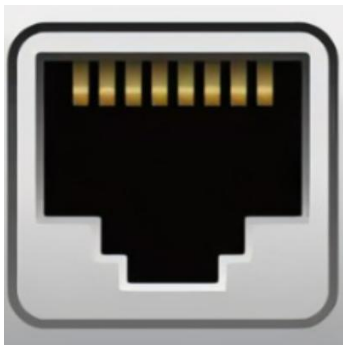
```
int main()
{
//连接测试  必须连接正常才能操作其他API
    kmNet_init((char*)"192.168.2.188", (char*)"12545", (char*)"F101383B");
```

Comment reads “Connection test. The connection must be normal to operate other APIs.
注释为“连接测试”。连接必须正常才能操作其他 API。

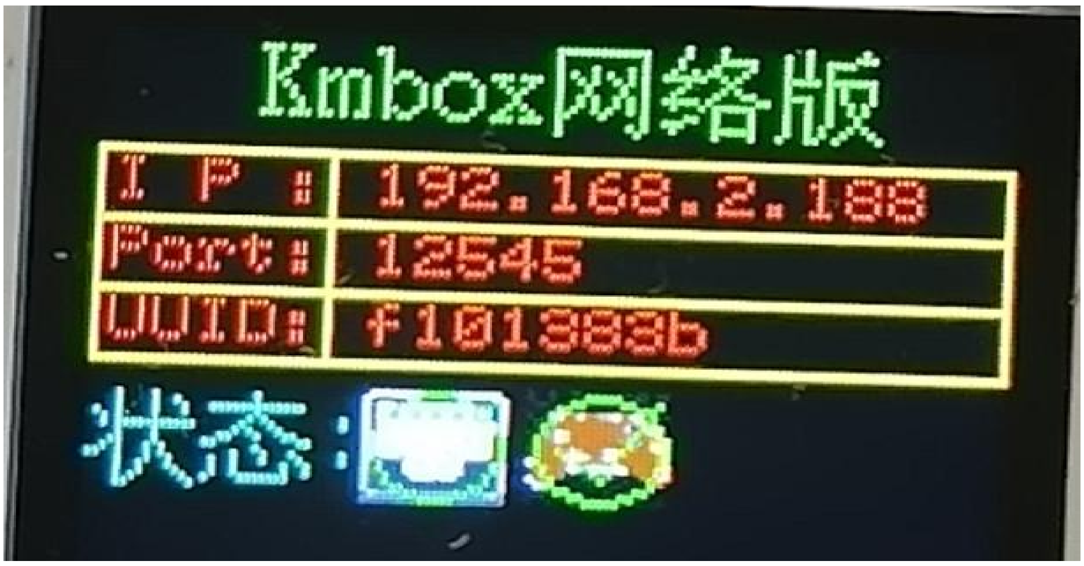
The first parameter is the IP address of the box. You can find it on the display of the box.
第一个参数是机器的 IP 地址。你可以在盒子的显示屏上找到它。
The second parameter is the port: the port number of the box. You can find it on the display of the box.
第二个参数是端口:机器的端口号。你可以在盒子的显示屏上找到它。
The third parameter is the UUID. You can find it on the display of the box.
第三个参数是 UUID。你可以在盒子的显示屏上找到它。
The above three parameters are the default factory box, and each device is different. Everything is subject to what is shown on the display. Since every device is different, batch targeting of problems at the source is avoided.
以上三个参数是默认的出厂框，每个设备都不一样。一切以显示屏上显示的为准。由于每个设备都是不同的，所以避免了在源头批量定位问题。

The first icon in the status bar is the network connection status. See the table below for details:
状态栏中的第一个图标是网络连接状态。有关详细信息，请参见下表:

When not connected to the network cable	Waiting to be connected	After successfully connecting the box
		
When the network cable is not connected, the network port has a cross icon.	The cross disappears after connecting to the network cable. At this time, it means that the box is waiting to be connected.	The box has been connected to the host computer (auxiliary software). This icon appears after calling kmNet_init.
当没有连接到网络电缆时	等待连接	在成功连接盒子之后

		
当网络电缆未连接时，网络端口有一个十字图标。	连接到网络电缆后，十字消失。这时候就说明盒子在等待连接。	盒子已经连接到主机(辅助软件)。此图标在调用 kmNet_init 后出现。

The picture below shows the display after calling kmNet_init to successfully connect the box:
下图是调用 kmNet_init 成功连接盒子后的显示:



Kmbox网络版

I P :	192.168.2.188
Port:	12545
UUID:	f101388b

Relative mouse movement kmNet_mouse_move
鼠标相对移动 kmNet_mouse_move

```
int kmNet_mouse_move(short x, short y);
int kmNet_mouse_move(短 x, 短 y);
```

This function is used to control mouse movement and takes about 1ms. This move is immediate and there are no intermediate states (jumps).

此功能用于控制鼠标移动，大约需要 1 毫秒。这种移动是即时的，没有中间状态(跳转)。

As shown below:

如下图所示:

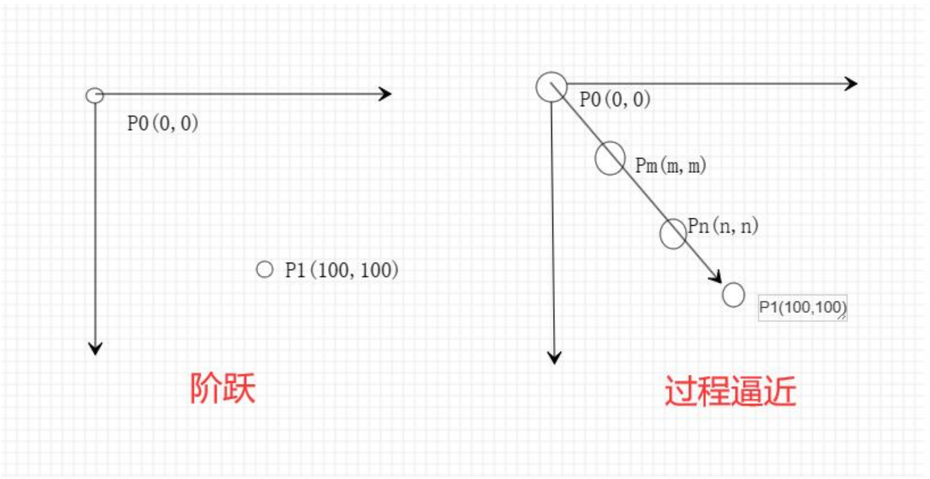
The mouse is currently at the coordinate (0,0) point, after calling kmNet_mouse_move

(100,100). The mouse will move directly to coordinates (100,100).

在调用 kmNet_mouse_move （100，100)之后，鼠标当前位于坐标(0，0)点。鼠标会直接移动到坐标(100，100)。

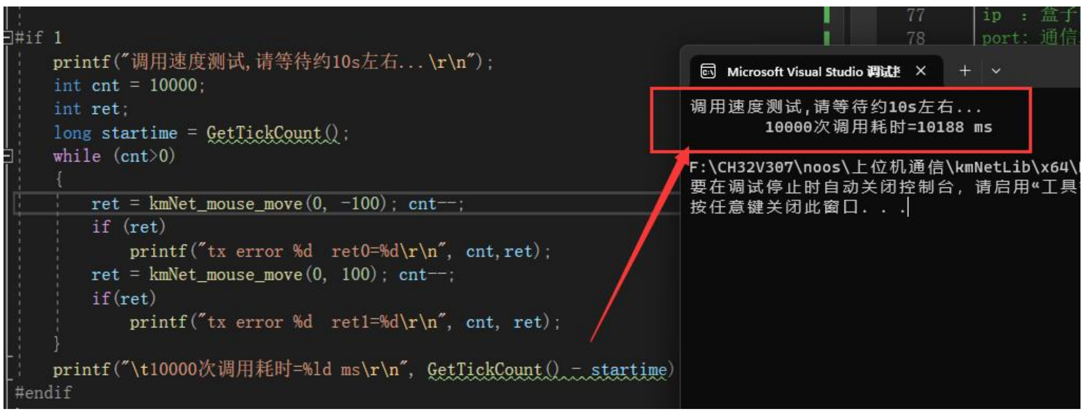
There is no transition in between. This kind of movement is easily detected by the game as an abnormal keyboard and mouse. Because it is normally operated with a physical mouse, it moves from (0,0) to (100,100). There will be many transition points (process approximations). The minimum unit of normal mouse movement is 1. Theoretically, when moving from point P0 (0,0) to P1 (100,100), there must be at least 100*2=144 transition points. If you step from (0,0) to (100,100) all at once, the game may consider the mouse data to be abnormal. Because you omitted the 143 transition points in the middle. The kmNet_mouse_move function is used as the most basic movement. Suitable for writing moving curves yourself.

两者之间没有过渡。这种动作很容易被游戏检测为键盘鼠标异常。因为它通常用物理鼠标操作，所以它从(0，0)移动到(100，100)。将会有许多转换点(过程近似值)。正常鼠标移动的最小单位是 1。理论上，当从点 P0 （0，0)移动到 P1 （100，100)时，必须至少有 100*2=144 个转换点。如果你一下子从(0，0)跳到(100，100)，游戏可能会认为鼠标数据异常。因为你省略了中间的 143 个过渡点。kmNet_mouse_move 函数作为最基本的移动。适合自己写移动曲线。



Left reads “step” (as moving in 1 large step). Right reads “process approximation” (I assume this is referring to smoothing).

左边写着“一步”（移动一大步)。右边写着“过程近似”（我假设这是指平滑)。



```
#if 1
    printf("调用速度测试,请等待约10s左右...\r\n");
    int cnt = 10000;
    int ret;
    long startime = GetTickCount();
    while (cnt>0)
    {
        ret = kmNet_mouse_move(0, -100); cnt--;
        if (ret)
            printf("tx error %d  ret0=%d\r\n", cnt,ret);
        ret = kmNet_mouse_move(0, 100); cnt--;
        if(ret)
            printf("tx error %d  ret1=%d\r\n", cnt, ret);
    }
    printf("\t10000次调用耗时=%ld ms\r\n", GetTickCount() - startime)
#endif
```

```
调用速度测试,请等待约10s左右...
        10000次调用耗时=10188 ms

F:\CH32V307\noos\上位机通信\kmNetLib\x64\
要在调试停止时自动关闭控制台，请启用“工具
按任意键关闭此窗口. . .
```

(image explained in the following paragraph. Text is essentially just the results of a speed benchmark)
(图像在下面的段落中解释。文本本质上只是速度基准测试的结果)

As shown in the figure, calling the mouse movement function 10,000 times takes a total of 10,188ms. This is equivalent to about 1.0188ms for each call. It can be called an average of 919 times per second. PS: It turns out that the speed of board A and B is about 300 times per second.
如图，调用鼠标移动函数 10000 次，总共需要 10,188ms。这相当于每次呼叫大约需要 1.0188 毫秒。平均每秒可以调用 919 次。PS:原来 A 板和 B 板的速度是每秒 300 次左右。

The sample speed is 1000 times per second.
采样速度为每秒 1000 次。

Mouse simulates manual movement kmNet_mouse_move_auto
鼠标模拟手动移动 kmNet_mouse_move_auto

Bagelnewt note: this is smoothing lol. Idk why they called it this. Also all the examples are for the c++ library. U gotta figure out the python library urself but it's pretty easy
Bagelnewt 注:这是平滑 lol。Idk 他们为什么这么叫它。此外，所有的例子都是 c++库。你必须自己弄清楚 python 库，但这很简单

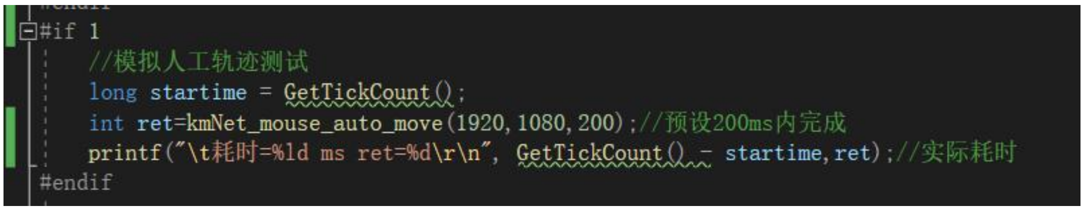
```
int kmNet_mouse_move_auto(int x, int y,int time_ms);//ok
int kmNet_mouse_move_auto(int x, int y, int time _ ms); //好的
```

The difference from kmNet_mouse_move is the third parameter. kmNet_mouse_move is a step-by-step step with the fastest speed, while kmNet_mouse_move_auto is an automatic move with an intermediate process. Suitable for automatically simulating artificial trajectories. The third parameter specifies how many milliseconds this move will take to complete. For example:
与 kmNet_mouse_move 的区别是第三个参数。kmNet_mouse_move 是速度最快的分步移动，kmNet_mouse_move_auto 是有中间过程的自动移动。适用于自动模拟人工轨迹。第三个参数指定完成这个移动需要多少毫秒。例如:

```
        kmNet_mouse_auto_move(1920,1080,200);//预设 200ms 内完成
        kmNet_mouse_auto_move(1920, 1080, 200); //预设 200 毫秒内完成
```

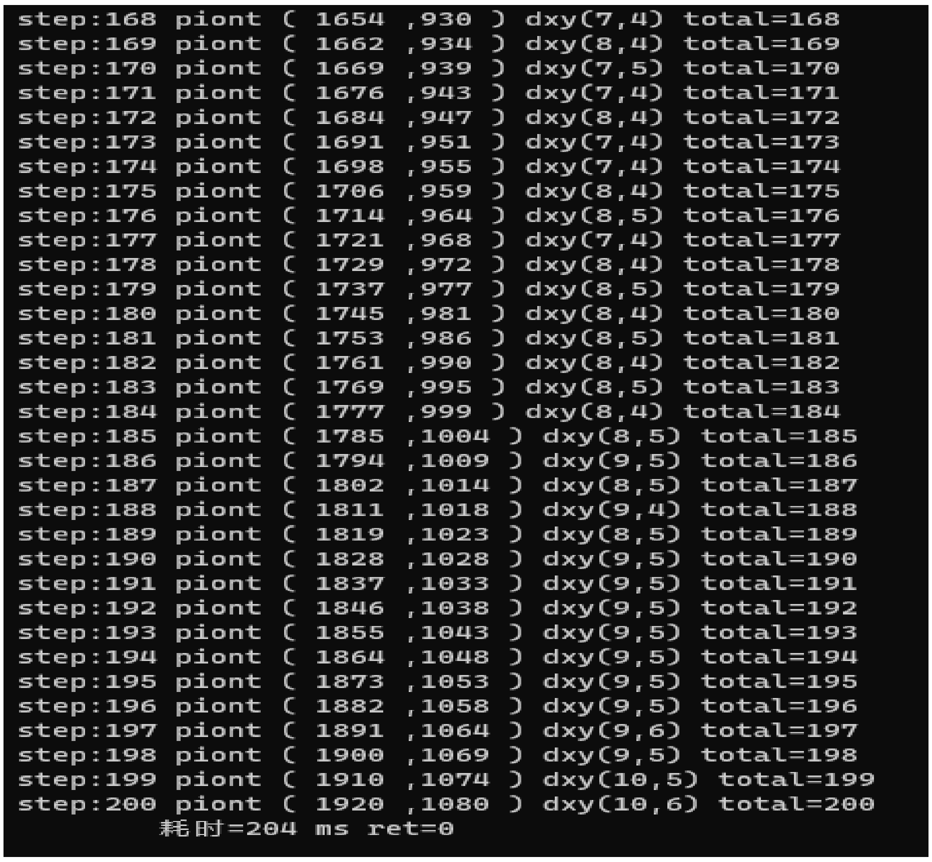
It is moved to the (1920,1080) point and is required to be completed within 200ms. After receiving this instruction, the box will automatically fill in the intermediate manual movement trajectory.

移到(1920，1080)点，要求 200ms 内完成。收到此指令后，方框会自动填写中间手动移动轨迹。
Move to the coordinate point (1920,1080) within 200ms.
在 200 毫秒内移动到坐标点(1920，1080)。



```
#if 1
    //模拟人工轨迹测试
    long startime = GetTickCount();
    int ret=kmNet_mouse_auto_move(1920,1080,200);//预设200ms内完成
    printf("\t耗时=%ld ms ret=%d\r\n", GetTickCount() - startime,ret);//实际耗时
#endif
```

Comments (top to bottom) read “simulate artificial trajectory”, “complete within 200 ms by default”, and “actual time taken”
注释(从上到下)显示“模拟人工轨迹”、“默认情况下在 200 毫秒内完成”和“实际花费的时间”



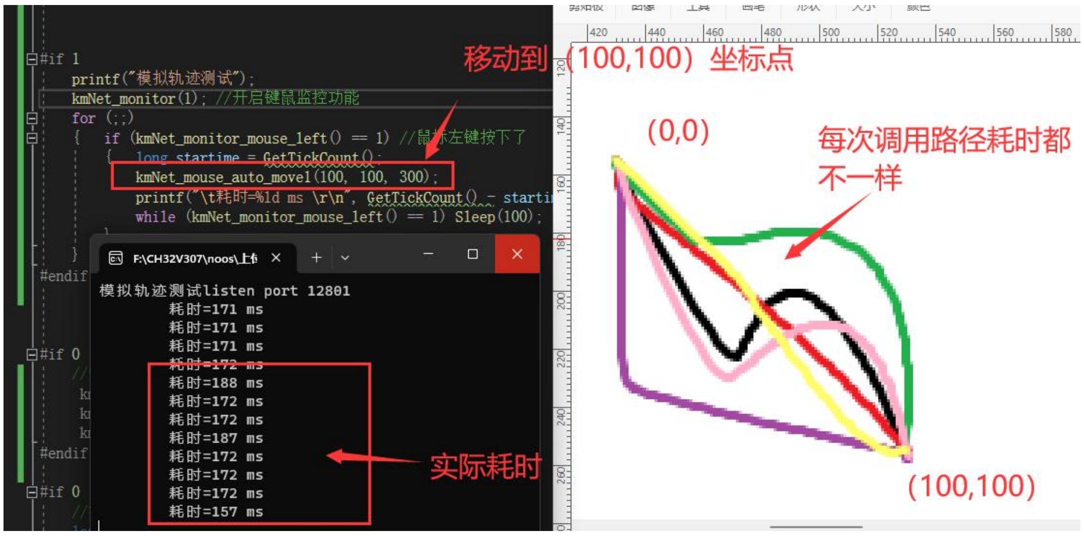
```
step:168 piont ( 1654 ,930 ) dxy(7,4) total=168
step:169 piont ( 1662 ,934 ) dxy(8,4) total=169
step:170 piont ( 1669 ,939 ) dxy(7,5) total=170
step:171 piont ( 1676 ,943 ) dxy(7,4) total=171
step:172 piont ( 1684 ,947 ) dxy(8,4) total=172
step:173 piont ( 1691 ,951 ) dxy(7,4) total=173
step:174 piont ( 1698 ,955 ) dxy(7,4) total=174
step:175 piont ( 1706 ,959 ) dxy(8,4) total=175
step:176 piont ( 1714 ,964 ) dxy(8,5) total=176
step:177 piont ( 1721 ,968 ) dxy(7,4) total=177
step:178 piont ( 1729 ,972 ) dxy(8,4) total=178
step:179 piont ( 1737 ,977 ) dxy(8,5) total=179
step:180 piont ( 1745 ,981 ) dxy(8,4) total=180
step:181 piont ( 1753 ,986 ) dxy(8,5) total=181
step:182 piont ( 1761 ,990 ) dxy(8,4) total=182
step:183 piont ( 1769 ,995 ) dxy(8,5) total=183
step:184 piont ( 1777 ,999 ) dxy(8,4) total=184
step:185 piont ( 1785 ,1004 ) dxy(8,5) total=185
step:186 piont ( 1794 ,1009 ) dxy(9,5) total=186
step:187 piont ( 1802 ,1014 ) dxy(8,5) total=187
step:188 piont ( 1811 ,1018 ) dxy(9,4) total=188
step:189 piont ( 1819 ,1023 ) dxy(8,5) total=189
step:190 piont ( 1828 ,1028 ) dxy(9,5) total=190
step:191 piont ( 1837 ,1033 ) dxy(9,5) total=191
step:192 piont ( 1846 ,1038 ) dxy(9,5) total=192
step:193 piont ( 1855 ,1043 ) dxy(9,5) total=193
step:194 piont ( 1864 ,1048 ) dxy(9,5) total=194
step:195 piont ( 1873 ,1053 ) dxy(9,5) total=195
step:196 piont ( 1882 ,1058 ) dxy(9,5) total=196
step:197 piont ( 1891 ,1064 ) dxy(9,6) total=197
step:198 piont ( 1900 ,1069 ) dxy(9,5) total=198
step:199 piont ( 1910 ,1074 ) dxy(10,5) total=199
step:200 piont ( 1920 ,1080 ) dxy(10,6) total=200
        耗时=204 ms ret=0
```

As shown in the images above, after giving the parameters of 200ms, the box simulates all movements from (0,0) to (1920,1080) after 204ms. This function is good for software calls without trajectory simulation. Boxes can automatically help generate trajectories for intermediate states. In addition, the third parameter is very important. Please refer to the actual manual operation time for everything. For example:

如上图所示，给定 200ms 的参数后，盒子模拟 204ms 后从(0，0)到(1920，1080)的所有运动。该功能适用于没有轨迹模拟的软件调用。盒子可以自动帮助生成中间状态的轨迹。此外，第三个参数非常重要。一切请以实际手动操作时间为准。例如:

kmNet_mouse_auto_move(1920,1080,1); The third parameter is 1ms. This must be a problem. No one can move the mouse pointer (1920,1080) in 1ms. The third parameter should be similar to manual operation time as much as possible. Parameter anomalies cause the box to be detected. Don't tell me it's a hardware problem, thank you. (In order to prevent keyboard and mouse abnormalities, the hardware makers are really worried!). And each movement path is randomly different. For details, please refer to the demo calling code.
kmNet_mouse_auto_move(1920，1080，1)；第三个参数是 1ms。这肯定是个问题。没有人能在 1ms 内移动鼠标指针(1920，1080)。第三个参数应该尽可能类似于手动操作时间。参数异常导致盒子被检测到。别告诉我是硬件问题，谢谢。(为了防止键盘鼠标异常，硬件厂商真的很担心！).并且每个运动路径是随机不同的。详情请参考演示调用代码。

Bagelnewt note: the above paragraph is just yapping about heuristic detections. Ignore it lmao
Bagelnewt 注意:上面的段落只是瞎扯启发式检测。忽略它 lmao


```
#if 1
    printf("模拟轨迹测试");
    kmNet_monitor(1); //开启键鼠监控功能
    for (;;)
    {   if (kmNet_monitor_mouse_left() == 1) //鼠标左键按下了
        {   long startime = GetTickCount();
            kmNet_mouse_auto_move1(100, 100, 300);
            printf("\t耗时=%ld ms \r\n", GetTickCount() - starti
            while (kmNet_monitor_mouse_left() == 1) Sleep(100);
```

```
模拟轨迹测试listen port 12801
        耗时=171 ms
        耗时=171 ms
        耗时=171 ms
        耗时=172 ms
        耗时=188 ms
        耗时=172 ms
        耗时=172 ms
        耗时=187 ms
        耗时=172 ms
        耗时=172 ms
        耗时=172 ms
        耗时=157 ms
```

Comments (top to bottom) read “Move to the point (100,100)” and “the time taken to move the whole distance is different each time”
注释(从上到下)显示“移动到点(100，100)”和“每次移动整个距离所用的时间不同”

In addition, more advanced algorithms will be added in the future. Mouse total velocity, partial velocity, acceleration, first-order derivative, second-order derivative, N-order derivative, direction vector, etc. Eliminate abnormal mouse data problems.
此外，未来还会加入更先进的算法。鼠标总速度，部分速度，加速度，一阶导数，二阶导数，N 阶导数，方向向量等。消除鼠标数据异常问题。

Bagelnewt note: Bezier curves have already been added. Here’s a brief tutorial:
Bagelnewt 注意:已经添加了贝塞尔曲线。这里有一个简短的教程:

int kmNet_mouse_move_beizer(int x, int y, int ms, int x1, int y1, int x2, int y2)
int kmNet _ mouse _ move _ beizer(int x, int y, int ms, int x1, int y1, int x2, int y2)

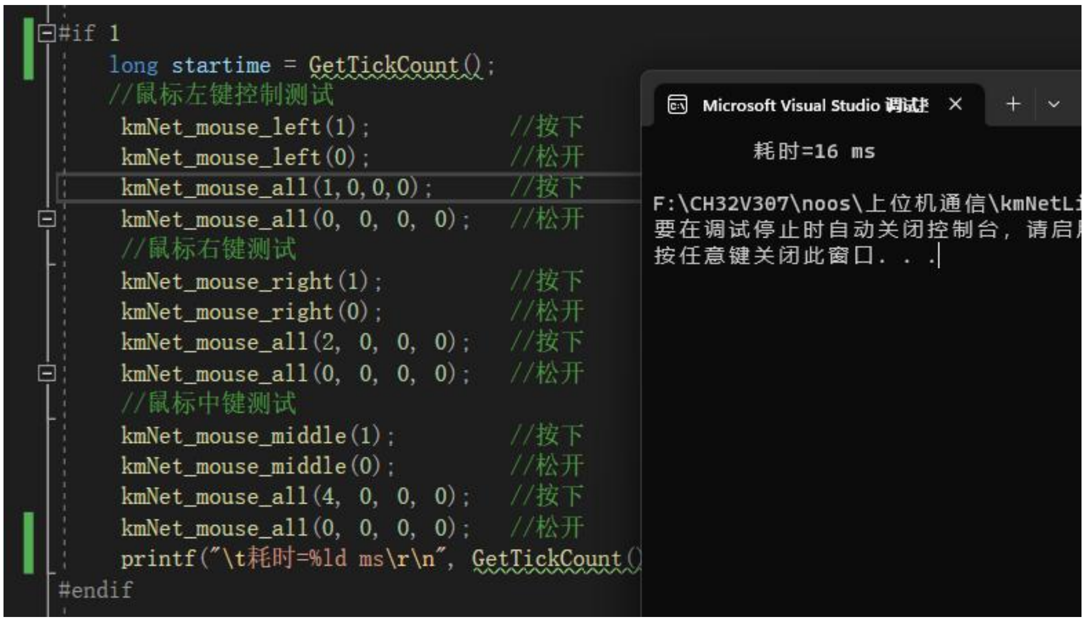
x and y are where you want to end up, so if x = 100 and y = 100 after all is said and done you will end up (100,100) away from where you started
x 和 y 是你想要结束的地方，所以如果 x = 100, y = 100，那么你将从你开始的地方结束(100, 100)
x1 and y1 are the coordinates of the first turn.
x1 和 y1 是第一圈的坐标。

Mouse button control
鼠标按钮控制

Mouse button control includes the following functions:
鼠标按钮控制包括以下功能:

```
int kmNet_mouse_left(int isdown);//Left click control
int kmNet _ mouse _ left(int is down); //左键控制
int kmNet_mouse_right(int isdown);//Right click control
int kmNet _ mouse _ right(int is down); //右键单击控件
int kmNet_mouse_middle(int isdown);//Middle button control
int kmNet _ mouse _ middle(int is down); //中间按钮控件
int kmNet_mouse_wheel(int wheel);//Scroll wheel control
int kmNet _ mouse _ wheel(int wheel); //滚轮控件
int kmNet_mouse_all(int button, int x, int y, int wheel);//All mouse data can be controlled at once
int kmNet_mouse_all(int button, int x, int y, int wheel); //可以一次控制所有鼠标数据
```

When Isdown=0, it means it’s not pressed, and when 1, it means it is pressed down.
Isdown=0 时表示没有按下，1 时表示按下。
When Wheel is positive, it means sliding the roller down, and when it is negative, it means sliding the roller up.
Wheel 为正时，表示向下滑动滚轮，为负时，表示向上滑动滚轮。
Button is the mouse button, x and y are the coordinates, and wheel is the scroll wheel.
Button 是鼠标键，x 和 y 是坐标，wheel 是滚轮。



```
#if 1
    long starttime = GetTickCount();
    //鼠标左键控制测试
     kmNet_mouse_left(1);           //按下
     kmNet_mouse_left(0);           //松开
     kmNet_mouse_all(1,0,0,0);      //按下
     kmNet_mouse_all(0, 0, 0, 0);   //松开
     //鼠标右键测试
     kmNet_mouse_right(1);          //按下
     kmNet_mouse_right(0);          //松开
     kmNet_mouse_all(2, 0, 0, 0);   //按下
     kmNet_mouse_all(0, 0, 0, 0);   //松开
     //鼠标中键测试
     kmNet_mouse_middle(1);         //按下
     kmNet_mouse_middle(0);         //松开
     kmNet_mouse_all(4, 0, 0, 0);   //按下
     kmNet_mouse_all(0, 0, 0, 0);   //松开
     printf("\t耗时=%ld ms\r\n", GetTickCount(
#endif
```

```
        耗时=16 ms

F:\CH32V307\noos\上位机通信\kmNetLi
要在调试停止时自动关闭控制台，请启
按任意键关闭此窗口. . .
```

Comments (top to bottom):
注释(从上到下)：
“Mouse button control test”
″鼠标按钮控制测试″
“press”
″新闻″
“release”
“释放”
“press”
″新闻″
“release”
“释放”
“Right mouse button test”
″鼠标右键测试″
“press”
″新闻″
“release”
“释放”
“press”
″新闻″
“release”
“释放”
“Middle mouse button test”
″鼠标中键测试″
“press”
″新闻″
“release”
“释放”
“press”
″新闻″

"release"
"释放"

Keyboard control functions
键盘控制功能

Keyboard control functions mainly include the following:
键盘控制功能主要包括以下几项:

```
int kmNet_keydown(int vkey);// ok
int kmNet _ keydown(int v key); //好的
int kmNet_keyup(int vkey);// ok
int kmNet _ keyup(int v key); //好的
```

Among them, vkey is the HID key code table of the key. When the keyboard key is pressed, the kmNet_keydown function is called, and when the keyboard key is released, the kmNet_keyup function is called.
其中，vkey 是钥匙的 HID 钥匙码表。当按下键盘键时，调用 kmNet_keydown 函数，当释放键盘键时，调用 kmNet_keyup 函数。

Physical keyboard and mouse status acquisition
物理键盘和鼠标状态采集

This type of function can be used when you need to know whether the keyboard or mouse button on the box is pressed. These functions read directly from the box hardware. It does not call the system API and can prevent hook detection to a certain extent.
当您需要知道盒子上的键盘或鼠标按钮是否被按下时，可以使用这种类型的功能。这些功能直接从机箱硬件读取。它不调用系统 API，可以在一定程度上防止钩子检测。

Bagelnewt note: Translations rough here. This is just reading whether or not keys/buttons are pressed.
Bagelnewt 注:此处翻译粗糙。这只是读取按键/按钮是否被按下。

//monitoring series
//监控系列

```
int kmNet_monitor(char enable); //Turn on and off physical keyboard and mouse status
monitoring
int kmNet _ monitor(char enable); //打开和关闭物理键盘和鼠标状态监视
int kmNet_monitor_mouse_left(); //Is the left mouse button pressed?
int kmNet _ monitor _ mouse _ left(); //鼠标左键是否按下?
int kmNet_monitor_mouse_middle();//Is the middle mouse button pressed?
int kmNet _ monitor _ mouse _ middle(); //鼠标中键按下了吗?
int kmNet_monitor_mouse_right();//Is the right mouse button pressed?
int kmNet _ monitor _ mouse _ right(); //鼠标右键按下了吗?
int kmNet_monitor_mouse_side1();//Is mouse side button 1 pressed?
int kmNet _ monitor _ mouse _ side 1(); //鼠标侧键 1 按下了吗?
int kmNet_monitor_mouse_side2();//Is mouse side button 2 pressed?
int kmNet _ monitor _ mouse _ side 2(); //鼠标侧键 2 按下了吗?
int kmNet_monitor_keyboard(int vk_key);//Is the key (whatever you put for vk_key) pressed?
int kmNet _ monitor _ keyboard(int vk _ key); //键(你给 vk_key 放的什么)按下了吗?
```

Note that the above functions are all in real-time status, that is, the current mouse status is detected at all times and will not block. It does not affect sending other instructions to the host computer. Because transmitting instructions and receiving monitoring information are performed on different ports. Before calling the monitoring function, please call kmNet_monitor(1) first to enable monitoring.
注意以上功能都是实时状态，即时刻检测当前鼠标状态，不会屏蔽。它不影响向主机发送其他指令。因为发送指令和接收监控信息是在不同的端口上执行的。在调用监控函数之前，请先调用 kmNet_monitor(1)来启用监控。

Physical keyboard and mouse blocking
物理键盘和鼠标阻塞

Bagelnewt note: Again translation is rough. I think I got it but idk for sure lol.
Bagelnewt 注:再次翻译是粗糙的。我想我得到了它，但 idk 肯定 lol。

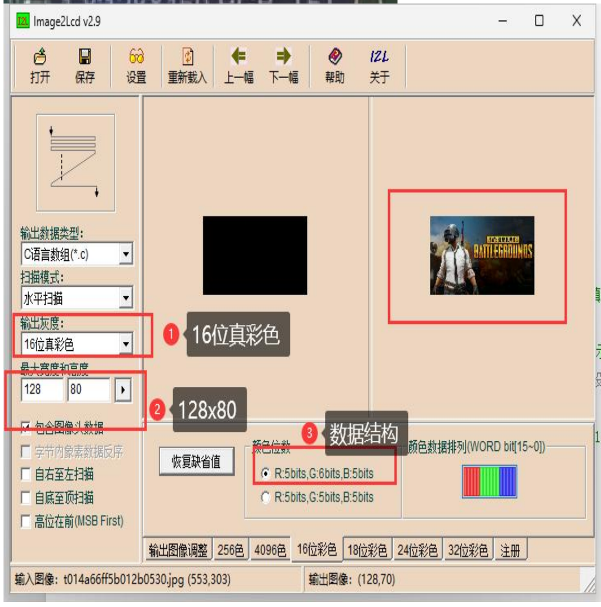
```
//Physical keyboard and mouse shielding series
//物理键盘鼠标屏蔽系列
int kmNet_mask_mouse_left(int enable);       //Block left mouse button
int kmNet _ mask _ mouse _ left(int enable); //阻止鼠标左键
int kmNet_mask_mouse_right(int enable);      //Block right mouse button
int kmNet _ mask _ mouse _ right(int enable); //阻止鼠标右键
int kmNet_mask_mouse_middle(int enable);//Block middle mouse button
int kmNet _ mask _ mouse _ middle(int enable); //阻止鼠标中键
int kmNet_mask_mouse_side1(int enable);      //Block side 1 mouse button
int kmNet _ mask _ mouse _ side 1(int enable); //阻挡第 1 面鼠标按钮
int kmNet_mask_mouse_side2(int enable);      //Block side 2 mouse button
int kmNet _ mask _ mouse _ side 2(int enable); //阻挡第二面鼠标按钮
int kmNet_mask_mouse_x(int enable);          //Block mouse x movement
int kmNet _ mask _ mouse _ x(int enable); //阻止鼠标 x 移动
int kmNet_mask_mouse_y(int enable);          //Block mouse y movement
int kmNet _ mask _ mouse _ y(int enable); //阻止鼠标的 y 轴移动
int kmNet_mask_mouse_wheel(int enable);      //Block mouse wheel
int kmNet _ mask _ mouse _ wheel(int enable); //阻止鼠标滚轮
int kmNet_mask_keyboard(short vkey);  //Block keyboard input of a specific key
int kmNet_mask_keyboard(短 v key); //阻止特定键的键盘输入
int kmNet_unmask_all();                                //Unblock all
int kmNet _ un mask _ all(); //全部解锁
```

LCD Display Control
LCD 显示控制

```
int kmNet_lcd_color(unsigned short rgb565);//Fill the entire LCD screen with the specified
color. Note that the parameters are in rgb565 format. You can fill it with black to clear
the screen.
int kmNet_lcd_color(无符号短整型 RGB 565); //用指定的颜色填充整个 LCD 屏幕。请注意，参数是
rgb565 格式。可以用黑色填充来清屏。
int kmNet_lcd_picture_bottom(unsigned char* buff_128_80); //The lower half displays a
128x80 image
int kmNet _ LCD _ picture _ bottom(unsigned char * buff _ 128 _ 80); //下半部分显示 128x80
的图像
```

Note that buff_128_80 is a modulo array of images with a resolution of 128x80. You can refer to the mold taking software settings.
注意，buff_128_80 是分辨率为 128×80 的图像模阵列。可以参考取模软件设置。

Bagelnewt note: what the fuck is this saying.
Bagelnewt 注:这他妈的是什么话。



输出数据类型: C语言数组(*.c)

扫描模式: 水平扫描

输出灰度: 16位真彩色

最大宽度和高度: 128 80

[✓] 包含图像头数据
[ ] 字节内象素数据反序
[ ] 自右至左扫描
[ ] 自底至顶扫描
[ ] 高位在前(MSB First)

颜色位数: (●) R:5bits,G:6bits,B:5bits ( ) R:5bits,G:5bits,B:5bits

颜色数据排列(WORD bit[15~0])

① 16位真彩色

② 128x80

③ 数据结构

输入图像: t014a66ff5b012b0530.jpg (553,303)

输出图像: (128,70)

Labels read 1: “16 bit true color” 2: “128x80” 3: “data structure”
标签显示为 1: “16 位真彩色” 2: “128 X80” 3: “数据结构”

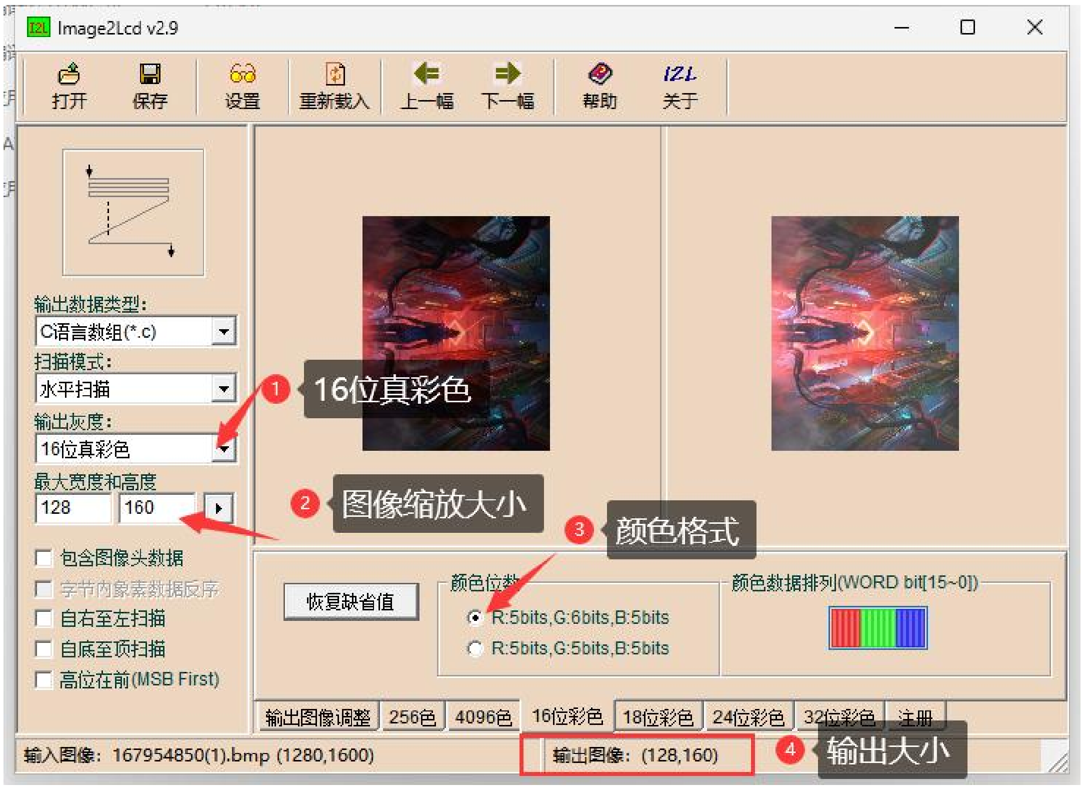
Bagelnewt note: this program that helps u change the LCD stuff can be found in kmboxnet\\doc\\lcd+í-ú¦ñ+¯\\-+¦¼+í-ú+f+¦(+·--)\\Image2Lcd 2.9(¦¦+G¦µ) (amazing folder names I know)

Bagelnewt 注意:这个帮助你改变 LCD 内容的程序可以在 kmboxnet \ \ doc \ \ LCD+í-ú-n+\ \-+++í-ú+f+(+-)\ \ image 2 LCD 2.9(+G)中找到(我所知道的惊人的文件夹名称)

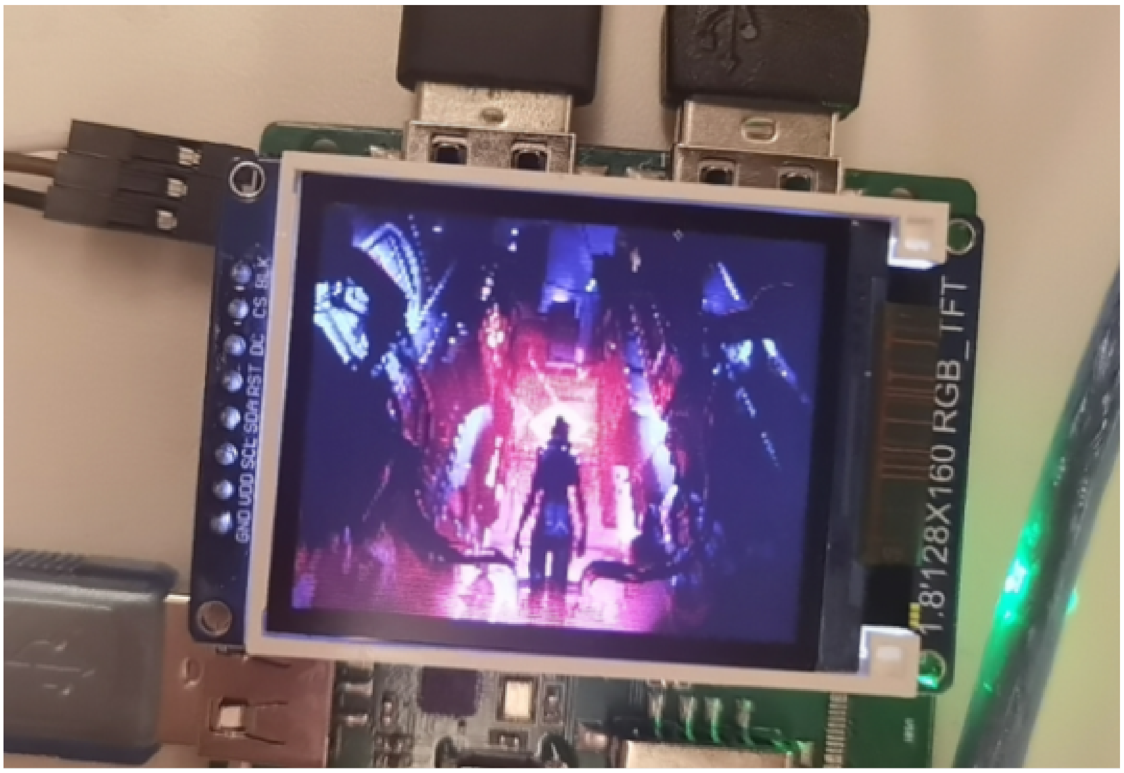
Note that the modulo array size must be 128x80x2 bytes.

请注意，模数组大小必须是 128x80x2 字节。

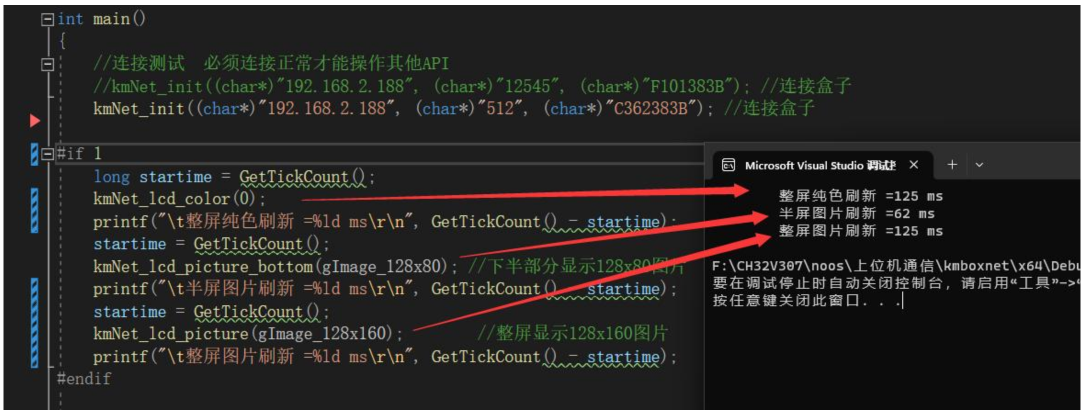
```
int kmNet_lcd_picture(unsigned char* buff_128_160);          //Display 128x160 picture on
the whole screen. This function refreshes the entire screen. Image size is fixed at
128x160.
int kmNet _ LCD _ picture(unsigned char * buff _ 128 _ 160); //全屏显示 128x160 画面。这个
函数刷新整个屏幕。图像尺寸固定为 128x160。
```



输出数据类型: C语言数组(*.c)
扫描模式: 水平扫描
输出灰度: 16位真彩色
最大宽度和高度: 128 160
颜色位数: [✓ R:5bits,G:6bits,B:5bits] R:5bits,G:5bits,B:5bits
颜色数据排列(WORD bit[15~0])
输入图像: 167954850(1).bmp (1280,1600)
输出图像: (128,160)



The following is the time consumption of the test code. Full screen takes 125ms. Half screen takes 62ms.
下面是测试代码的时间消耗。全屏需要 125ms。半屏需要 62ms。



```
int main()
{
    //连接测试  必须连接正常才能操作其他API
    //kmNet_init((char*)"192.168.2.188", (char*)"12545", (char*)"F101383B"); //连接盒子
    kmNet_init((char*)"192.168.2.188", (char*)"512", (char*)"C362383B"); //连接盒子

#if 1
    long starttime = GetTickCount();
    kmNet_lcd_color(0);
    printf("\t整屏纯色刷新 =%ld ms\r\n", GetTickCount() - starttime);
    starttime = GetTickCount();
    kmNet_lcd_picture_bottom(gImage_128x80); //下半部分显示128x80图片
    printf("\t半屏图片刷新 =%ld ms\r\n", GetTickCount() - starttime);
    starttime = GetTickCount();
    kmNet_lcd_picture(gImage_128x160);          //整屏显示128x160图片
    printf("\t整屏图片刷新 =%ld ms\r\n", GetTickCount() - starttime);
#endif
```

```
        整屏纯色刷新 =125 ms
        半屏图片刷新 =62 ms
        整屏图片刷新 =125 ms

F:\CH32V307\noos\上位机通信\kmboxnet\x64\Debu
要在调试停止时自动关闭控制台，请启用“工具”->“
按任意键关闭此窗口. . .
```

Bagelnewt note: This is stupid. I’m not translating this. Who cares how fast the screen is.
Bagelnewt 注:这是愚蠢的。我不翻译这个。谁在乎屏幕有多快。

About upgrading
关于升级

The box supports firmware upgrade function. The source code of the upgrade tool can be found in the upgrade tool under the doc folder. You can modify the upgrade tool yourself. Take the official demo as an example. How to upgrade the box firmware later.
盒子支持固件升级功能。升级工具的源代码可以在升级工具的 doc 文件夹下找到。您可以自己修改升级工具。以官方演示为例。后期如何升级盒子固件？