



OneNET 视频开发板

使用手册

版本信息

版本号	修订日期	作者	修订内容	备注
V1.0	2018.01.24	吴志盛	文档首版	

目 录

1 开发板介绍	6
1.1 MCU 介绍--GM8136S	6
1.2 接口介绍	6
1.2.1 电源接口	6
1.2.2 以太网接口	7
1.2.3 串口	7
1.2.4 USB/WIFI	8
1.2.6 按键	8
1.2.7 摄像头接口	8
1.2.8 MIC 接口	9
1.2.9 耳机接口	9
1.2.10 TF 卡	9
1.2.12 LED	10
1.2.13 综合接口	10
1.2.13 配置开关	11
1.3 硬件接口测试	11
1.3.1 开发板上电	11
1.3.1 Usart/开发板启动	11
1.3.2 按键	12
1.3.3 以太网	12
1.3.4 Wifi	13
1.3.5 TF 卡	14
1.3.6 音频	15
1.3.7 视频	15
1.4 功能介绍	16
1.4.1 设置网络	16
1.4.2 视频直播体验	16
1.4.3 OneNET EDP 协议体验	18
1.4.4 OneNET MQTT 协议体验	20
1.4.5 OneNET HTTP 协议体验	21
1.5 开发环境介绍	23
2 ONENET 以及 ONENET 视频能力介绍	25
2.1 OneNET	25
2.2 OneNET 视频能力	25
2.3 视频能力应用范例架构	25
3 ONENET 视频应用	28
3.1 基于 MQTT 协议 OneNET 视频推送	28
3.1.1 基于 MQTT 协议 OneNET SDK 介绍	28
3.1.2 基于 MQTT 的 OneNET SDK 案例介绍和使用	29
3.2 基于私有协议 OneNET 视频推送	32

3.2.1 基于私有协议 OneNET SDK 介绍.....	32
3.2.2 基于私有协议的 OneNET SDK 案例介绍和使用.....	34
3.3 基于开发板与 OneNET 的视频推送直播.....	34
3.3.1 基础介绍.....	34
3.3.2 程序设计.....	36
3.4 基于开发板与 OneNET 的音视频推送直播.....	41
3.4.1 基础介绍.....	41
3.4.2 程序框架设计.....	42
4 ONENET EDP 协议.....	42
4.1 EDP 协议介绍.....	42
4.2 程序设计.....	42
5 ONENET HTTP 协议.....	45
5.1 OneNET HTTP 协议介绍.....	45
5.2 程序设计.....	45
6 ONENET MQTT 协议.....	49
6.1 MQTT 协议介绍.....	49
6.2 Mqtt SDK 介绍.....	49
6.2.1 在项目中链接 MQTT SDK 策略.....	49
6.2.2 Mqtt SDK 基本使用方法.....	49
6.3 程序设计.....	54
附录 1 嵌入式 LINUX 应用开发介绍.....	56
1.1 Linux 基本 shell 命令.....	56
1.1.1 命令: ls [选项][目录].....	56
1.1.2 命令 pwd.....	56
1.1.3 命令: cd [路径].....	56
1.1.4 命令: clear.....	57
1.1.5 命令: mkdir [].....	57
1.1.6 命令: touch [文件名].....	57
1.1.7 命令: rm [选项][文件或文件夹].....	57
1.1.8 命令: tar [选项][文件目录].....	57
1.1.9 命令: cp [选项][源文件或目录][目标文件或目录].....	58
1.1.10 命令: find [查找路径] -name [文件名].....	59
1.1.11 命令: man [选项].....	59
1.1.12 命令: ifconfig.....	59
1.1.13 命令: apt-get [选项][目录].....	59
1.1.14 vi/vim 编辑器命令.....	60
1.2 Linux 应用程序开发.....	60
1.3 嵌入式 linux 应用开发介绍.....	62
1.3.1 嵌入式 linux 开发体系.....	62
1.3.2 搭建嵌入式 linux 开发环境.....	62
1.4 helloworld 开发.....	81

1.4.1 notepad++编辑.....	81
1.4.2 宿主机交叉编译.....	82
1.4.3 目标执行--调试接口.....	83
1.5 helloworld 开发 基于 makefile.....	83
1.5.1 notepad++编辑.....	83
1.5.2 宿主机交叉编译.....	83
1.5.3 目标执行--调试接口.....	84
1.6 helloworld 开发 基于 cmake.....	84
1.6.1 notepad++编辑.....	84
1.6.2 宿主机交叉编译.....	84
1.6.3 目标执行--调试接口.....	85

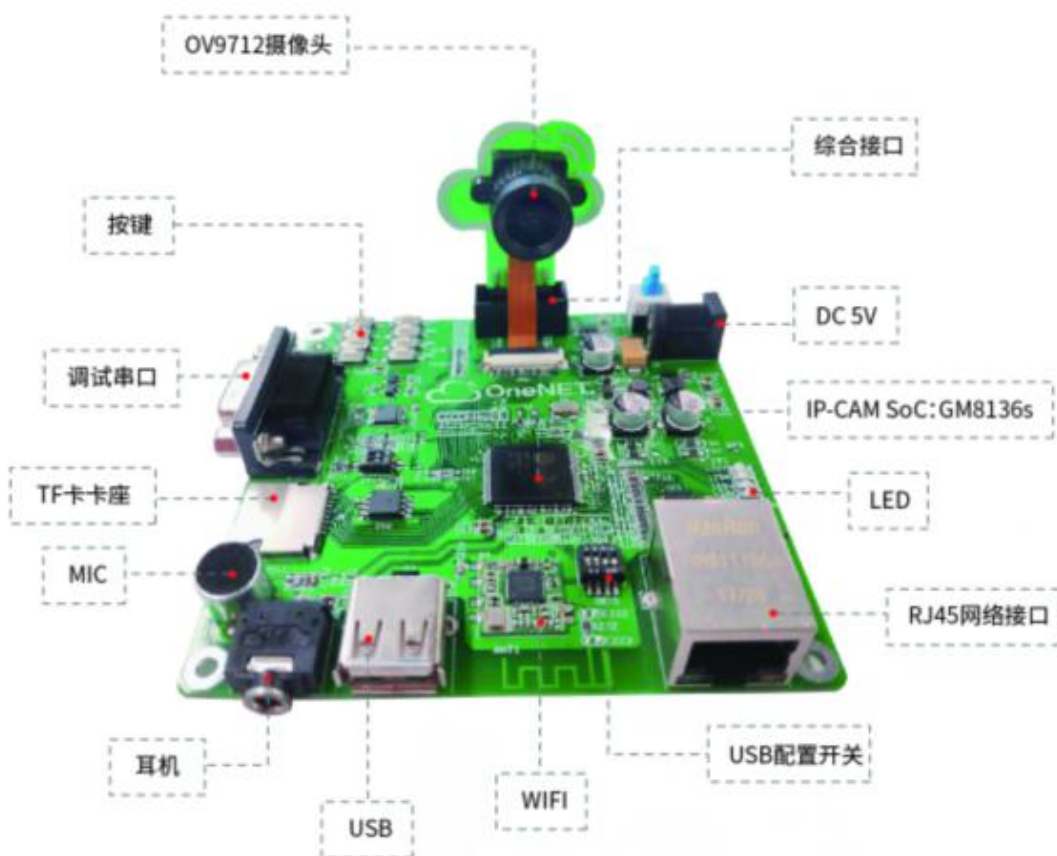
1 开发板介绍

OneNET 视频全功能开发板意在帮助用户和开发者熟悉 OneNET 的视频能力，让开发者了解和熟悉 OneNET 视频 SDK，以及如何应用 OneNET 视频 SDK 快速搭建基于 OneNET 的视频监控系统、安防系统以及直播系统。同时，开发板配备了外部数据采集监测、摄像头抓拍监测、内存监测等小实验，让开发者熟悉在嵌入式 linux 环境下如何快速开发 OneNET 物联网应用。

1.1 MCU 介绍--GM8136S

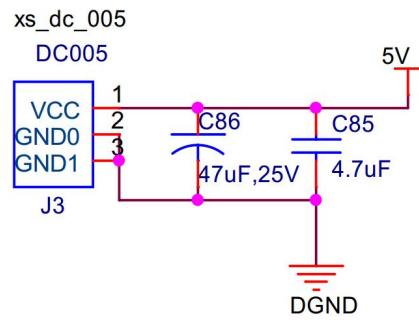
开发板采用 GM8136S 作为核心处理器，GM8136S 是一款为 IP-CAM 领域设计的高性价比 SoC 芯片，具有集成度高、易开发等特点。

1.2 接口介绍



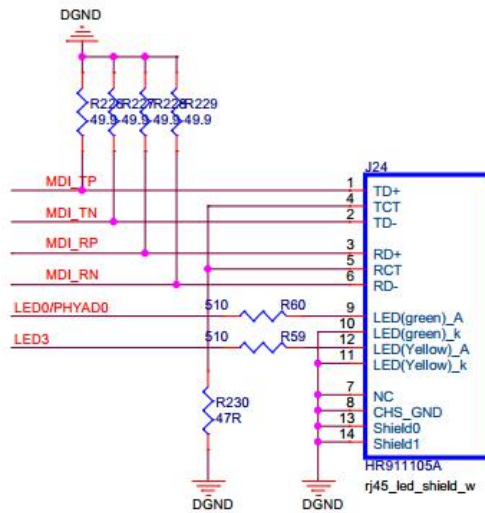
1.2.1 电源接口

输入电源为 5V，接口如下图所示，请使用官方配置适配器接入电源。



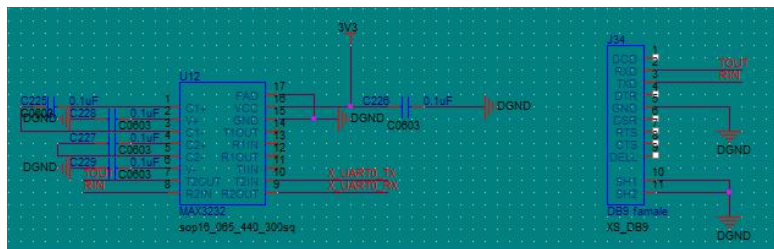
1.2.2 以太网接口

开发板提供 100M 以太网，请根据需求配置 IP 地址使用



1.2.3 串口

串口主要用于用户开发者与开发板交互，搭配官方所配的 USB 转串口线使用



1.2.4 USB/WIFI

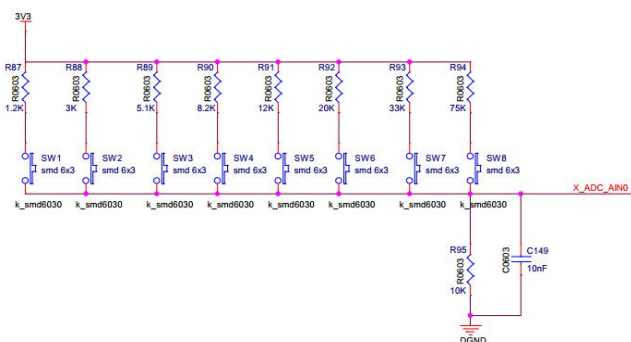
USB 可外插 USB 设备



USB 接口与 WIFI 共用一个 USB 通道，因此二者不可同时使用，需通过 SW15 拨码开关进行配置，当拨码开关 1、2 处于连通状态(开关靠近 ON)，可用 USB 接口；当拨码开关 3、4 处于连通状态(开关靠近 ON)，可用 WIFI；

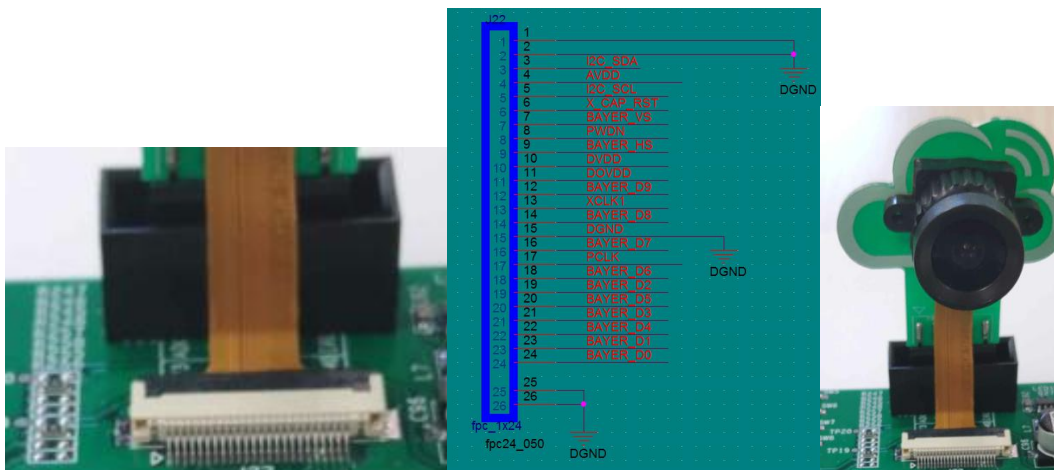
1.2.6 按键

配置 8 个按键，通过 AD 采样实现按键区分。



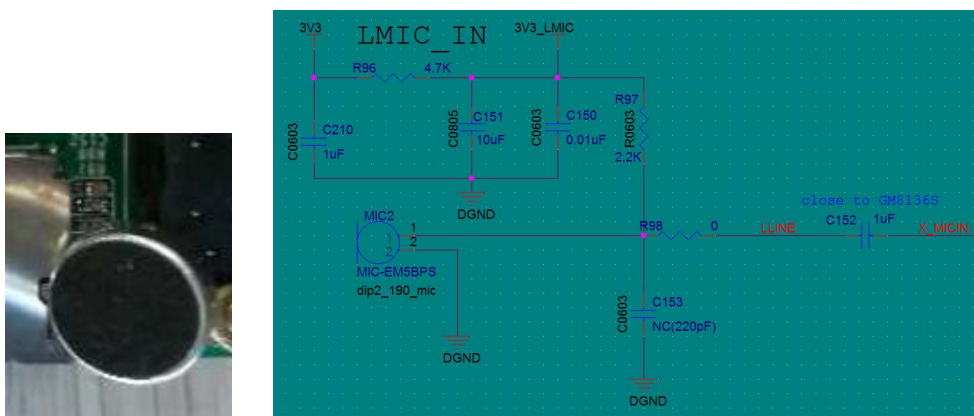
1.2.7 摄像头接口

请配合官方配置摄像头使用



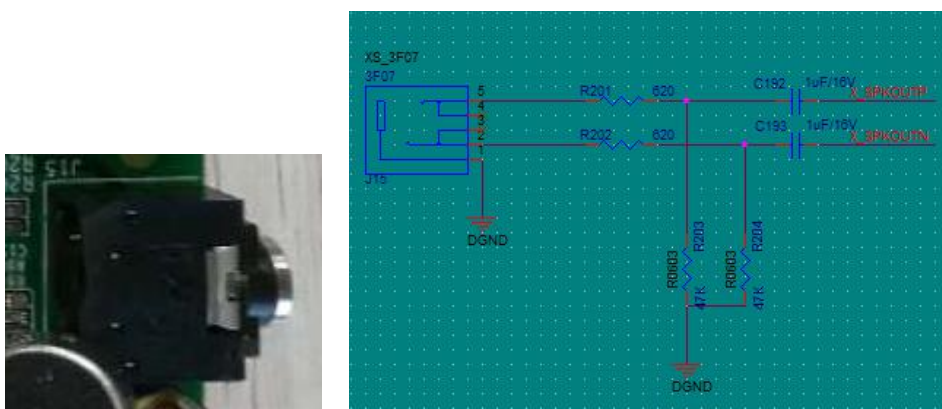
1.2.8 MIC 接口

配置咪头 MIC，供开发者进行语音应用开发。



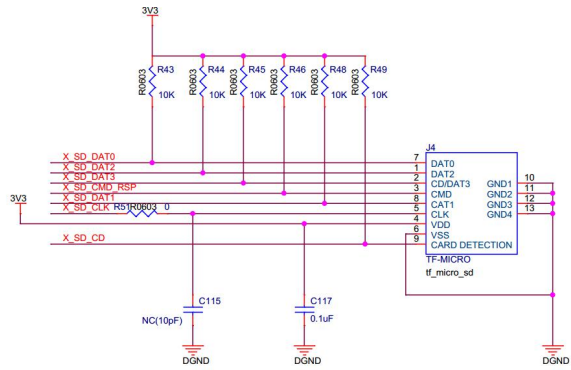
1.2.9 耳机接口

配置耳机接口，供开发者进行语音应用开发。



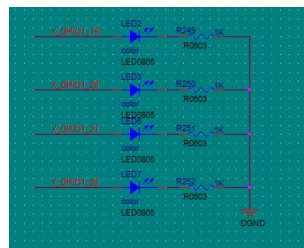
1.2.10 TF 卡

开发板可使用 TF 卡作为外部存储



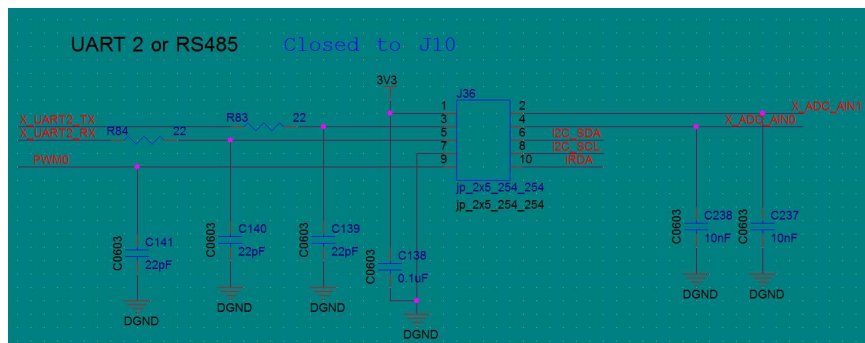
1.2.12 LED

配置 4 路 LED



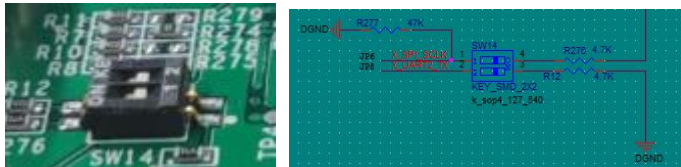
1.2.13 综合接口

扩展接口提供一路 USART、一路 PWM、一路 I2C、一路红外、两路 ADC 供开发者使用。



1.2.13 配置开关

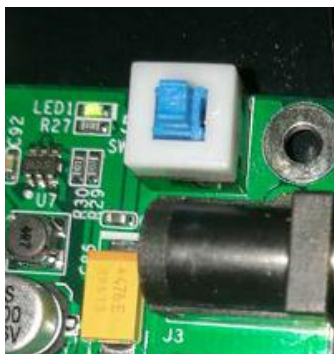
配置开关与系统启动相关，请保持 SW14 1,2 处于断开状态，否则系统无法启动。



1.3 硬件接口测试

1.3.1 开发板上电

接上 5V 电源适配器，按下开关，开发板上电。正常状态下：LED1 灯亮。



1.3.1 Usart/开发板启动

打开超级终端。插上与串口连接的出口线(若笔记本电脑无串口，可使用串口-USB 转换器)，开发板启动，超级终端输出界面。如无相关经验，可参考附录 1 的 1.3.2 章节

```
gs zuz304 i vpd_slave, Live 0x7f23000 (U)
audio_drv 324816 i vpd_slave, Live 0x7f29e000 (Q)
codec 22038Framap: 15036 pages in DDR0 are freed.
S i audio_drv, LSet EM debug level(0) (0:quiet 1:get/put 2:property 3:semaphore)
Live 0x7f253000 (
Set GraphService debug level(0). (0 for error, 1 for flow, 2 for detail.)
FO)
osd_dispatcherSet MemoryService debug level(0).
h 6427 i vpd_slaveSet datain debug level(0)
ve, Live 0x7f25dSet dataout debug level(0)
000 (Q)
fscalervpv set vpslv_dbglevel=0
300 226803 i osdSet VPD debug level(0).
_dispatcher, Live Debug Level:0
0x7f21e000 (Q)
sw_osg 75212 i vpd_slave, Live 0x7f207000 (Q)
fmjpeg_drv 74941 0 - Live 0x7f1f0000 (Q)
decoder 10603 i fmjpeg_drv, Live 0x7f1ea000 (Q)
favo_rc 29431 0 - Live 0x7f1df000 (Q)
favo_enc 380999 i favo_rc, Live 0x7f17b000 (Q)
fmcp_drv 110151 i fmjpeg_drv, favo_enc, Live 0x7f15a000 (Q)
vcap300_1sp 5992 0 - Live 0x7f158000 (Q)
vcap0 1694 0 - Live 0x7f154000 (Q)
vcap300_common 307903 i vpd_slave, osd_dispatcher,vcap300_1sp, Live 0x7f101000 (Q)
fisp_ov9715 7088 0 - Live 0x7f0fc000 (Q)
fisp_algorithm 8464 0 - Live 0x7f0e1000 (PO)
fisp328 153226 i vcap300_1sp, fisp_ov9715, fisp_algorithm, Live 0x7f0b2000 (Q)
ft3dnr200 79292 i fisp328, Live 0x7f099000 (Q)
adda308 31351 i audio_drv, Live 0x7f08f000 (Q)
fe_common 5935 i audio_drv, adda308, Live 0x7f08a000 (Q)
ftpmvdr010 8063 0 - Live 0x7f085000 (Q)
think2d 14948 i sw_osg, Live 0x7f07e000 (Q)
sar_adc 13018 0 - Live 0x7f076000 (Q)
em 160619 i vpd_slave, gs, audio_drv, fscaler300, fmjpeg_drv, decoder, favo_enc, vcap300_common, ft3dnr200, Live 0x7f044000 (PO)
ms 120084 i vpd_slave, gs, Live 0x7f01e000 (PO)
log 15946 i vpd_master, vpd_slave, loop_comm, gs, audio_drv, fscaler300, fmjpeg_drv, decoder, favo_rc, favo_enc, fmcp_drv, vcap300_common, ft3dnr200, em, ms, Live 0x7f014000 (PO)
sh220 1874 0 - Live 0x7f008000 (Q)
Framap 20422 i vpd_master, vpd_slave, loop_comm, audio_drv, fscaler300, sw_osg, favo_enc, fmcp_drv, vcap300_common, fisp328, ft3dnr200, think2d, em, ms, Live 0x7f000000 (Q)
=====
; GM8136_LM(Tiny Memory) Product Configuration (Without LCD, MP804, Scaler Substream, Encode Switching)
=====
Video Front End: ov9715
Chip Version: 81360001
RootFS Version:
MTD Version: 00_2015-01-09-GM8136_LMP
/ #
```

可通过超级终端输入相关命令进行交互比如 ls

```
/ # ls
bin          gm          mnt          squashfs_init  var
boot.sh      init        proc         sys
dev          lib         sbin         tmp
etc          linuxrc     share        usr
/ #
```

1.3.2 按键

输入命令 ./saradc_test 执行 saradc_test 程序，可测试按键。

```
/mnt/nfs # ./saradc_test
key_fd = 3
Start SAR ADC Keypad Test
AP get KEY ADC status =4 val = e7
AP get KEY ADC status =4 val = d8
AP get KEY ADC status =4 val = af
AP get KEY ADC status =4 val = 7e
_
```

1.3.3 以太网

通过 ifconfig 设置 IP 地址，并通过 ping 命令测试网络

```
/ # ifconfig eth0 192.168.200.148
HW reset
/ # phy speed is 10, half duplex
HW reset
phy speed is 100, full duplex
HW reset

/ # route add default gw 192.168.200.1
/ # ifconfig
eth0      Link encap:Ethernet  HWaddr 12:B6:C3:31:FC:49
          inet addr:192.168.200.148  Bcast:192.168.200.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:130 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:7268 (7.0 KiB)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```



```

/mnt/nfs # ping www.baidu.com
PING www.baidu.com (183.232.231.172): 56 data bytes
64 bytes from 183.232.231.172: seq=0 ttl=52 time=38.441 ms
64 bytes from 183.232.231.172: seq=1 ttl=52 time=38.901 ms
64 bytes from 183.232.231.172: seq=2 ttl=52 time=37.699 ms
^C
--- www.baidu.com ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 37.699/38.347/38.901 ms

```

1.3.4 Wifi

接入 wifi 热点，其中 Baiduiot 为热点名称，LinuxIOT123 为 wifi 密码，需替换为自己的 wifi 信息。

```
./wpa_passphrase Baiduiot "LinuxIOT123" >> /usr/wpa_supplicant.conf
```

```
./wpa_supplicant -Dwext -ira0 -c/usr/wpa_supplicant.conf -B
```

```

/lib/modules # ./wpa_passphrase Baiduiot "LinuxIOT123" >> /usr/wpa_supplicant.conf
/lib/modules # ./wpa_supplicant -Dwext -ira0 -c/usr/wpa_supplicant.conf -B
WlanFunCtrl.word = 0xff200003
MACVersion = 0x76010500
Allocate 8192 memory for BA reordering
MAC[Ver:Rev=0x76010500]
USBLoadFirmwareToAndes
FW Version:0.1.00 Build:7640
Build Time:201302052146
ILM Length = 45380 (bytes)
DLM Length = 0 (bytes)
Loading FW....
#
USBLoadFirmwareToAndes: COM_REG0(0x730) = 0x1
--> NICInitRecv
<-- NICInitRecv()
--> NICInitTransmit
MGMT Ring: total 32 entry allocated
<-- NICInitTransmit(Status=0)
--> MLME Initialize
RTMP_TimerListAdd: add timer obj 894309f4!
RTMPInitTimer: 894309f4
RTMP_TimerListAdd: add timer obj 89430a24!
RTMPInitTimer: 89430a24
RTMP_TimerListAdd: add timer obj 89430a54!
RTMPInitTimer: 89430a54
RTMP_TimerListAdd: add timer obj 894309c4!
RTMPInitTimer: 894309c4
RTMP_TimerListAdd: add timer obj 89430934!
RTMPInitTimer: 89430934
RTMP_TimerListAdd: add timer obj 89430964!

```

获取 IP

```
ifconfig ra0 `udhcpc -i ra0 | grep 'Lease of' | awk '{print $3}'`
```

```

route add default gw `ifconfig|grep 'inet addr'|grep -v '127.0.0.1'|cut -d: -f2 |awk '{p
rint $1}'|awk '{split($0,ip,"."); printf "%s.%s.%s\n",ip[1],ip[2],ip[3] }'`

```

```

/lib/modules # ifconfig ra0 `udhcpc -i ra0 | grep 'Lease of'| awk '{print $3}'`
RTMP_TimerListAdd: add timer obj 89479f04!
RTMP_InitTimer: 89479f04
BA Ori Session Timeout(1) : Send ADD BA again
BA - Send ADDBA request. StartSeq = 1, FrameLen = 33. BufSize = 64
PeerAddBARspAction ==> Wcid(1)
                StatusCode = 0
ba>WinSize=63, MaxSize=21, MaxPeerRxSize=32
ba> reassign max win size from 63 to 21
BAOriSessionAdd():TXBAbitmap=1, BAWinSize=21, TimeOut=0
/lib/modules #
/lib/modules # route add default gw `ifconfig|grep 'inet addr'|grep -v '127.0.0.1'|cut -d: -f2 |awk '{print $1}'|awk '{split($0,ip,"."); printf "%s.%s.%s.1\n",ip[1],ip[2],ip[3] }'`

```

测试

```

/lib/modules # ifconfig
lo                Link encap:Local Loopback
                  inet addr:127.0.0.1  Mask:255.0.0.0
                  UP LOOPBACK RUNNING  MTU:16436  Metric:1
                  RX packets:0 errors:0 dropped:0 overruns:0 frame:0
                  TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
                  collisions:0 txqueuelen:0
                  RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

ra0               Link encap:Ethernet  HWaddr 00:0C:43:23:3E:78
                  inet addr:192.168.95.2  Bcast:192.168.95.255  Mask:255.255.255.0
                  UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
                  RX packets:8995 errors:0 dropped:0 overruns:0 frame:0
                  TX packets:32 errors:0 dropped:0 overruns:0 carrier:0
                  collisions:0 txqueuelen:1000
                  RX bytes:1863805 (1.7 MiB)  TX bytes:3624 (3.5 KiB)

```

1.3.5 TF 卡

通过命令 `mount -t vfat /dev/mmcblk0p1 /usr/tfcard` 挂载 tf 卡

```

/usr # mkdir tfcard
/usr # mount -t vfat /dev/mmcblk0p1 /usr/tfcard
/usr # cd tfcard/
/usr/tfcard # ls
Android  DCIM  LOST.DIR
/usr/tfcard # cd DCIM
/usr/tfcard/DCIM # ls
/bin/sh: LS: not found
/usr/tfcard/DCIM # ls
Camera  photo
/usr/tfcard/DCIM # cd Camera/
/usr/tfcard/DCIM/Camera # ls
back    front    protect
/usr/tfcard/DCIM/Camera # cd back/
/usr/tfcard/DCIM/Camera/back # ls
/usr/tfcard/DCIM/Camera/back # cd ../front/
/usr/tfcard/DCIM/Camera/front # ls
Recfront_20180210_171421.mp4  Recfront_20180211_073344.mp4
Recfront_20180210_171921.mp4  Recfront_20180211_073407.mp4
Recfront_20180210_172421.mp4  Recfront_20180211_073907.mp4
Recfront_20180210_172921.mp4  Recfront_20180211_074407.mp4
Recfront_20180210_173421.mp4  Recfront_20180211_074907.mp4
Recfront_20180210_173921.mp4  Recfront_20180211_075407.mp4
Recfront_20180210_174421.mp4  Recfront_20180211_075907.mp4
Recfront_20180210_174921.mp4  Recfront_20180211_080407.mp4
Recfront_20180210_220026.mp4  Recfront_20180211_080907.mp4
Recfront_20180210_220527.mp4  Recfront_20180211_081407.mp4
Recfront_20180210_221027.mp4  Recfront_20180211_082013.mp4
/usr/tfcard/DCIM/Camera/front #

```

1.3.6 音频

通过执行 `audio_livesound` 测试音频的输入和输出

```
/mnt/nfs # ./audio_livesound 0 0
Audio livesound from input vch 0 to output vch 0
Enter q to quitaudio info: change rec channel type to mono

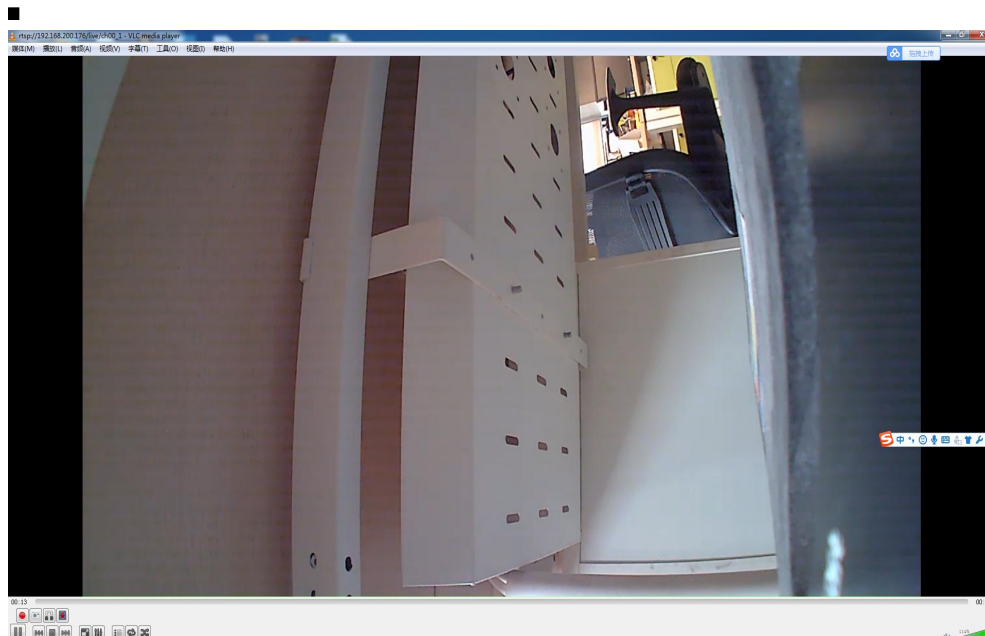
Enter q to exit
```

1.3.7 视频

进入 `/mnt/mtd`, 执行 `rtspd` 程序, 可在 PC 端通过 VLC 等播放器播放实时视频。

```
/ # cd /mnt/mtd/
/mnt/mtd # ./rtspd
Usage:
Multiple streams:
./rtspd
Max single h264 stream:
./rtspd -s
Max single mpeg4 stream:
./rtspd -m
Max single mjpeg stream:
./rtspd -j

#### Apply #1 ####
GM RTSP Library [build - Oct 2 2014 15:03:35]
Connect command:
rtsp://192.168.200.176/live/ch00_0
rtsp://192.168.200.176/live/ch00_1
Press 'q' to exit.
/live/ch00_0: cap0_0 320x240 H264 30.80fps 283kbps
/live/ch00_1: cap0_3 1280x720 H264 30.61fps 4173kbps
```



1.4 功能介绍

功能演示需通过与开发板交互完成，若无嵌入式 linux 使用或开发经验，请先阅读附录 1，并按照附录 1.3.2 完成超级终端的安装。

1.4.1 设置网络

开发板配置 wifi 和有线两种网络通信方式，开发者可自由选择任意一种。

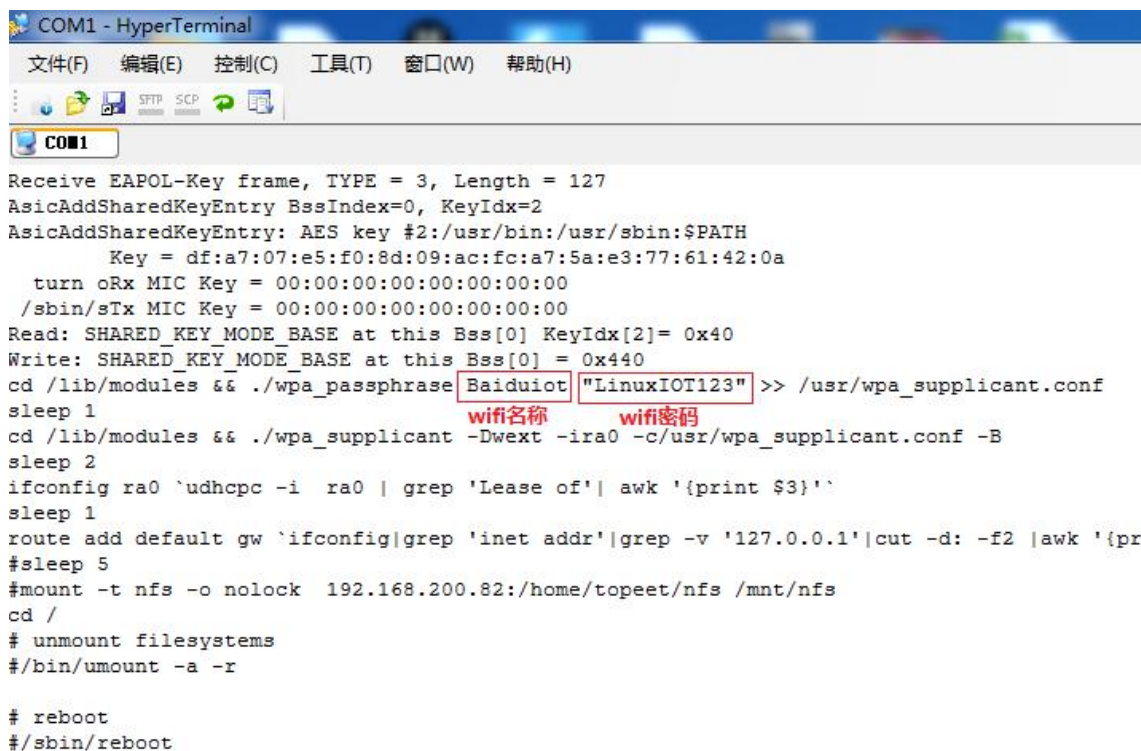
有线网络

设置 IP 地址命令：ifconfig eth0 192.168.200.172

设置网络命令：route add default gw 192.168.200.1

WIFI

进入/usr/onenet 目录,并打开 wifi.sh，将其中的 wifi 名称和 wifi 密码改成需设定的 wifi 名称和 wifi 密码，并保存。



```
COM1 - HyperTerminal
文件(F) 编辑(E) 控制(C) 工具(T) 窗口(W) 帮助(H)
COM1
Receive EAPOL-Key frame, TYPE = 3, Length = 127
AsicAddSharedKeyEntry BssIndex=0, KeyIdx=2
AsicAddSharedKeyEntry: AES key #2:/usr/bin:/usr/sbin:$PATH
    Key = df:a7:07:e5:f0:8d:09:ac:fc:a7:5a:e3:77:61:42:0a
    turn oRx MIC Key = 00:00:00:00:00:00:00:00
    /sbin/sTx MIC Key = 00:00:00:00:00:00:00:00
Read: SHARED_KEY_MODE_BASE at this Bss[0] KeyIdx[2]= 0x40
Write: SHARED_KEY_MODE_BASE at this Bss[0] = 0x440
cd /lib/modules && ./wpa_passphrase Baiduiot "LinuxIOT123" >> /usr/wpa_supplicant.conf
sleep 1
cd /lib/modules && ./wpa_supplicant -Dwext -ira0 -c/usr/wpa_supplicant.conf -B
sleep 2
ifconfig ra0 `udhcpc -i ra0 | grep 'Lease of'| awk '{print $3}'`
sleep 1
route add default gw `ifconfig|grep 'inet addr'|grep -v '127.0.0.1'|cut -d: -f2 |awk '{pr
#sleep 5
#mount -t nfs -o nolock 192.168.200.82:/home/topeet/nfs /mnt/nfs
cd /
# unmount filesystems
#/bin/umount -a -r

# reboot
#/sbin/reboot
```

执行 wifi.sh 脚本，连接 wifi。sh wifi.sh

注：所有体验实验都需要网络，请务必先设置网络。

1.4.2 视频直播体验

目的：让用户快速体验视频能力。

OneNET 视频体验实验功能：

以官方 SDK 为基础，实现视频直播功能。

体验流程：

请按照 <https://open.iot.10086.cn/doc/art388.html#68> 中介绍的步骤，在 onenet 中创建视频产品和设备，并获取产品 ID 和注册码。产品管理页获取产品 ID，注册码

产品信息

产品名称: videotest
产品ID: 113746
正式设备注册码: Kyu[REDACTED]Yvw 使用方式
Master-APIkey: tslK[REDACTED]Qcf0= 使用方式
产品行业: 智能家居
产品类别: 家用电器>厨房电器>其它
产品简介: 视频监控

进入开发板目录 /usr/onenet

使用 vi config.json 命令修改 config.json 文件

```
{^M
  "profile":{^M
    "productid":113746,^M
    "deviceid":0,^M
    "pass":"KyuXupAEgJ8DjYvw",^M
    "id":"mac-or-something"^M
  },^M
}
```

将 productid、pass 的值修改为上面所创建视频设备的产品 ID，注册码，deviceid 可设置为 0，id 可自主设置。

运行 ./sample_test

```
/mnt/mtd # ./sample_test
[1970/01/01 00:05:48] [INFO]cfg_path:config.json

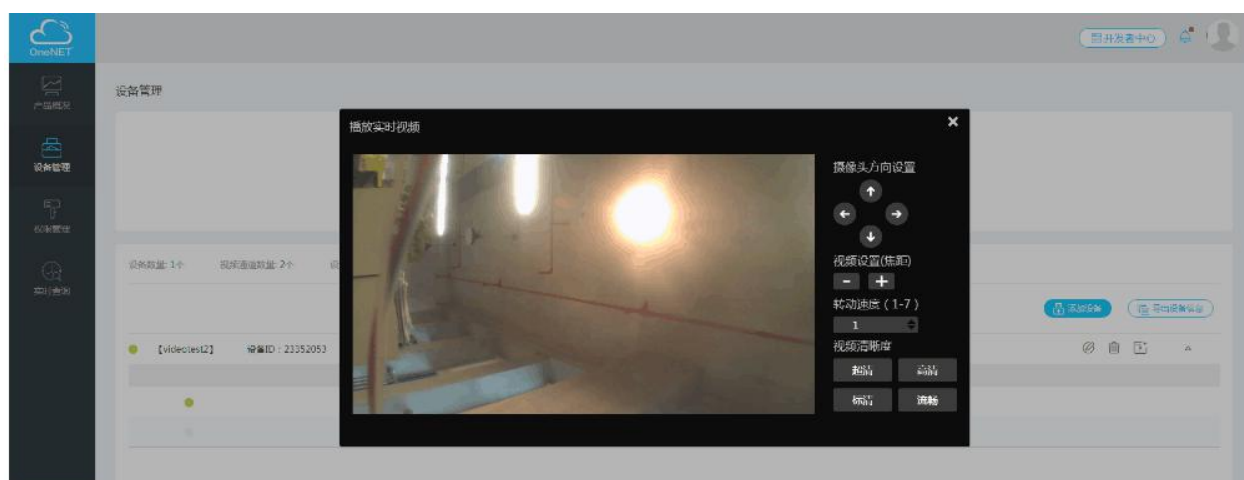
[1970/01/01 00:05:50] [INFO]get dev id 31407579

ont_video_live_stream_stop

#### Apply #1 ####

#### Apply #2 ####
url rtmp://183.230.40.42:1935/live/7hrfN51jipxxgT48RpAhSx-IUCoUb5-RSI
TcwprPBocctJdQ20Zaudio info: change rec channel type to mono
J3sXbI-nIvTi8squlK5Kkp1Q1GiFeg1ATWf1iEHCgTWP0WAOMLRSmXonG7dz01P21Mv?3
line 807, size 36980, ts 377002
line 466 send sps pps
line 807, size 7552, ts 377069
line 807, size 6591, ts 377103
line 807, size 2467, ts 377136
line 807, size 4146, ts 377170
line 923, size 768 ts 377247
tag 0xaf, profile 2, rate 8, channel 1, config[0]=0x14 config[1]=0x8
line 807, size 3180, ts 377270
line 807, size 6481, ts 377304
line 923, size 768 ts 377311
line 807, size 4552, ts 377337
line 807, size 2634, ts 377371
line 923, size 768 ts 377375
line 807, size 1688, ts 377404
line 923, size 768 ts 377439
line 807, size 2999, ts 377471
line 923, size 768 ts 377503
line 807, size 3409, ts 377505
```

通过 onenet 视频管理页面即可观看视频



1.4.3 OneNET EDP 协议体验

目的：让用户以 SDK 为基础快速开发 OneNET EDP 应用。

OneNET EDP 体验实验功能：

以官方 SDK 为基础，实现 AD 采集数据上传、摄像头抓拍图片上传(二进制数据)、开发板内存监测数据上传以及其他类型数据上传等功能，同时完成了平台下发命令控制开发板 LED 灯的功能。

体验步骤：

请按照 <https://open.iot.10086.cn/doc/art254.html#68> 中介绍的步骤，在 onenet 中创建 EDP 产品和设备，并获取设备 ID，设备 APIkey 信息。

• 基本信息



test123

设备ID : 25843300

创建时间 : 2018年03月02日 15:58

鉴权信息 : qazwsxedc123456

接入方式 : EDP

设备数据权限 : 私密

设备API地址 : <http://api.heclouds.com/devices/25843300>

设备APIkey : rGLDIc9KsAtTCLUv1ZUwXPP=aA0=

超级终端中输入命令，执行 OneNET EDP 协议样例程序

```
./video_edp -d 25843300 -k CrsVhmC1kd3LHvra8x62Jr1p9jw=
```

其中-d 参数为设备 ID, -k 参数为 APIKEY

设备连接到 OneNET 后, 可通过 OneNET 发送命令 led0, on 或 led0, off 控制 led 灯

命令	解释
led0, on	Led0 亮
led0, off	Led0 灭
led1, on	Led1 亮
led1, off	Led1 灭
led2, on	Led2 亮
led2, off	Led2 灭
led3, on	Led3 亮
led3, off	Led3 灭

```
/mnt/nfs # ./video_edp -d 25843300 -k CrsVhmC1kd3LHvra8x62Jrlp9jw=
key_fd1 = 3
Start SAR ADC Keypad Test
Error! Abnormal AP exit under chipid(0),cpuid(1)
#### Apply #11 ####
,graph(1),query_id=0x10001

#### Apply #12 ####
send connect to server, bytes: 51
10 31 00 03 45 44 50 01
40 01 2C 00 08 32 35 38
34 33 33 30 30 00 1C 43
72 73 56 68 6D 43 31 6B
64 33 4C 48 76 72 61 38
78 36 32 4A 72 6C 70 39
6A 77 3D
recv from server, bytes: 4
20 02 00 00
recv connect resp, rtn: 0
need more bytes...
```

设备信息

数据展示

19条
设备数据总数

数据流展示

	2	最新数据:0.80000001192093 更新时间:2018年03月02日 15:58:40
	1	最新数据:0.5 更新时间:2018年03月02日 15:58:40
	3	最新数据:-0.5 更新时间:2018年03月02日 15:58:40
	ad_vaule	最新数据:1.307059 更新时间:2018年03月02日 16:04:42
	pic	更新时间:2018年03月02日 16:04:44
	MemFree	最新数据:44896 更新时间:2018年03月02日 16:00:20
	Cached	最新数据:41092 更新时间:2018年03月02日 16:00:20
	SwapCached	最新数据:0 更新时间:2018年03月02日 16:00:20
	MemTotal	最新数据:126404 更新时间:2018年03月02日 16:00:20
	Buffers	最新数据:2608 更新时间:2018年03月02日 16:00:20

1.4.4 OneNET MQTT 协议体验

目的：让用户以 SDK 为基础快速开发 OneNET MQTT 应用。

OneNET MQTT 体验实验功能：

以官方 SDK 为基础，实现 MQTT 连接、数据上传(AD 采集数据、摄像头抓拍图片(二进制数据)、开发板内存监测数据上传以及其他类型数据)、主题订阅与取消等功能，同时也实现了平台下发命令控制开发板 LED 灯的功能。

体验步骤：

请按照 <https://open.iot.10086.cn/doc/art253.html#68> 中介绍的步骤，在 onenet 中创建 mqtt 产品和设备，并获取产品 ID，设备 ID，鉴权信息。



进入目录，在开发板中输入命令，执行 OneNET mqtt 样例程序

```
./MqttSample -j 117135 -a pub123456 -d 24621169
```













其中-j 参数为产品 ID，-a 参数为鉴权信息 -d 为设备 ID

设备连接到 OneNET 后，可通过 OneNET 发送命令 led0, on 或 led0, off 控制 led 灯
命令参考 edp 程序的命令

```
/mnt/nfs # ./MqttSample -j 117135 -a pub123456 -d 24621169
Error! Abnormal AP exit under ch
##### Apply #7 #####
ipid(0),cpuid(1),graph(1),query_id=0x10001

##### Apply #8 #####
Commands:
connect      Establish the connection.
ping         Send ping packet.
publish      send data points, support parameter -q 0/1/2 (Qos0/Qos1/Qos2), -t 1-7
push_dp      push data points
cmdret       reponse cmd to server, support parameter -q 0/1/2 (Qos0/Qos1/Qos2)
subscribe    Subscribe the data streams.
unsubscribe  Unsubscribe the data streams.
disconnect   Close the connection.
exit         Exit the sample.
help         Print the usage of the commands.
```


数据流展示

	111	最新数据:789 更新时间 : 2018年02月28日 10:42:55
	0	最新数据:-20.299999237061 更新时间 : 2018年01月31日 06:34:44
	temperature	最新数据:22.5 更新时间 : 2015年03月22日 22:31:12
	image	更新时间 : 1970年01月01日 02:09:25
	Cached	最新数据:13320 更新时间 : 2018年03月02日 15:39:48
	SwapCached	最新数据:0 更新时间 : 2018年03月02日 15:39:48
	MemFree	最新数据:72896 更新时间 : 2018年03月02日 15:39:48
	MemTotal	最新数据:126404 更新时间 : 2018年03月02日 15:39:48
	Buffers	最新数据:2608 更新时间 : 2018年03月02日 15:39:48
	1	最新数据:102 更新时间 : 2018年03月02日 15:40:25
	pm2.5	最新数据:89 更新时间 : 2018年03月02日 15:40:25
	3	最新数据:10 更新时间 : 2018年03月02日 15:40:25

1.4.5 OneNET HTTP 协议体验

目的：让用户以 SDK 为基础快速开发 OneNET HTTP 应用。

OneNET HTTP 体验实验功能：

以官方 SDK 为基础，通过同时建立 EDP 和 HTTP 通信，以 EDP 协议作为命令下发通道，http 为数据上传通道，实现 AD 采集数据上传、摄像头抓拍图片上传(二进制数据)、开发板内存监测数据上传以及其他类型数据上传等功能。

体验步骤：

请按照 <https://open.iot.10086.cn/doc/art252.html#68> 中介绍的步骤，在 onenet 中创建 mqtt 产品和设备，并获取产品 Master-key，产品注册码，设备 ID。

OneNET HTTP 协议体验实验中用的 EDP 和 HTTP 协议，通过平台通过 EDP 发送命令，通过 HTTP 协议上数据。

请新建一个 EDP 产品和 EDP 设备，无需创建 HTTP 设备，程序会自动创建。



进入目录，在开发板中输入命令，执行 OneNET http 样例程序

```
./video_http -k CrsVhmC1kd3LHvra8x62Jrlp9jw= -r 5nqkSDyQ5mP2hojG -d 258
```

43300

其中-k 参数为 masterkey，-r 参数为设备注册码，-d 为 edp 设备 ID。

设备连接到 OneNET 后，可通过 OneNET 发送命令 led0,on 或 led0,off 控制 led 灯

```
/mnt/nfs # ./video_http -k CrsVhmC1kd3LHvra8x62Jrlp9jw= -r 5nqkSDyQ5mP2hojG -d :
5843300
11POST /register_de?register_code=5nqkSDyQ5mP2hojG HTTP/1.1
Host:183.230.40.33
Content-Length:235

{
  "sn": "201802021148",
  "title": "onenet_http",
  "desc": "http create device",
  "location": {
    "ele": 500,
    "lat": 29.521000,
    "lon": 106.542000
  },
  "private": true,
  "tags": ["edp", "mqtt", "http", "rtmp"],
  "protocol": "HTTP"
}
Create Device Success, deviceID is 25449120
Save_TemperatureToOneNet 217 t:20
[thread_command] recv thread start ...
recv from server, bytes: 4
20 02 00 00
recv connect resp, rtn: 0
need more bytes...
```

通过 OneNET 平台发送命令

命令	解释
cmd=1	更新设备 GPS 信息
cmd=2	创建数据流
cmd=3	更新数据流信息
cmd=4	上传数据

cmd=5	创建 API KEY
cmd=6	更新 API KEY 信息
cmd=7	抓拍图片上传
cmd=8	内存信息上传

EDP协议下发命令

命令内容:

cmd=1

Qos: ☒ 不需要响应 ☐ 需要响应

失效时间:

10

说明: 过期时间单位为“秒”, 默认“0”, 最大“2678400”

☒ 字符串 ☐ 十六进制

发送命令 取消

1.5 开发环境介绍

开发者可自己搭建虚拟机开发环境,也可直接使用提供的 linux 虚拟机.虚拟机已安装好各种需要的依赖库以及工具,可直接应用于 OneNET 视频以及开发板视频程序开发。

```
root@ubuntu:/onenet# ls
arm-linux  protocol  tools  video
root@ubuntu:/onenet#
```

开发相关工具和代码存放在/onent 目录下。若开发者自己搭建开发环境,请安装相关工具和库。

目录介绍

- arm-linux 附录代码
- protocol OneNET edp/mqtt/http SDK 使用案例代码
- tools 相关工具
- video OneNET 视频和开发板案例代码

已安装工具

- a. Openssl 库
- b. X264 库
- c. Rtmp 库
- d. Ssh 服务
- e. Nfs 服务
- f. Tftp 服务
- g. Linux/windows 共享文件夹
- h. 交叉编译工具
- i. Cmake 编译工具

2 OneNET 以及 OneNET 视频能力介绍

2.1 OneNET

OneNET 是中国移动面向公共服务和物联网应用推出的 PaaS 平台，通过提供海量连接、云端存储、消息分发和大数据分析等优质服务，降低物联网企业和个人（创客）的研发、运营和运维成本，使企业和创客更加专注于应用、客户和市场。

详细介绍请访问：

<https://open.iot.10086.cn/>

2.2 OneNET 视频能力

OneNET 视频能力是基于 OneNET 平台构建的视频设备接入能力，为用户提供开放接入，远程设备控制，视频推流和云端分发等底层核心能力，并提供二次开发套件，方便用户定制开发自身行业应用。

- 设备接入，反向控制

- 支持 MQTT 协议接入

- 设备推流

- 支持 RTMP 推流

- 云端分发

- 支持 RTMP, HLS 分发

- 设备端推流

- 提供 c-sdk，提供设备接入，推流等接口，可进行平台移植

- 播放端

- 支持三方常见播放器 jwplayer、vlc、video.js

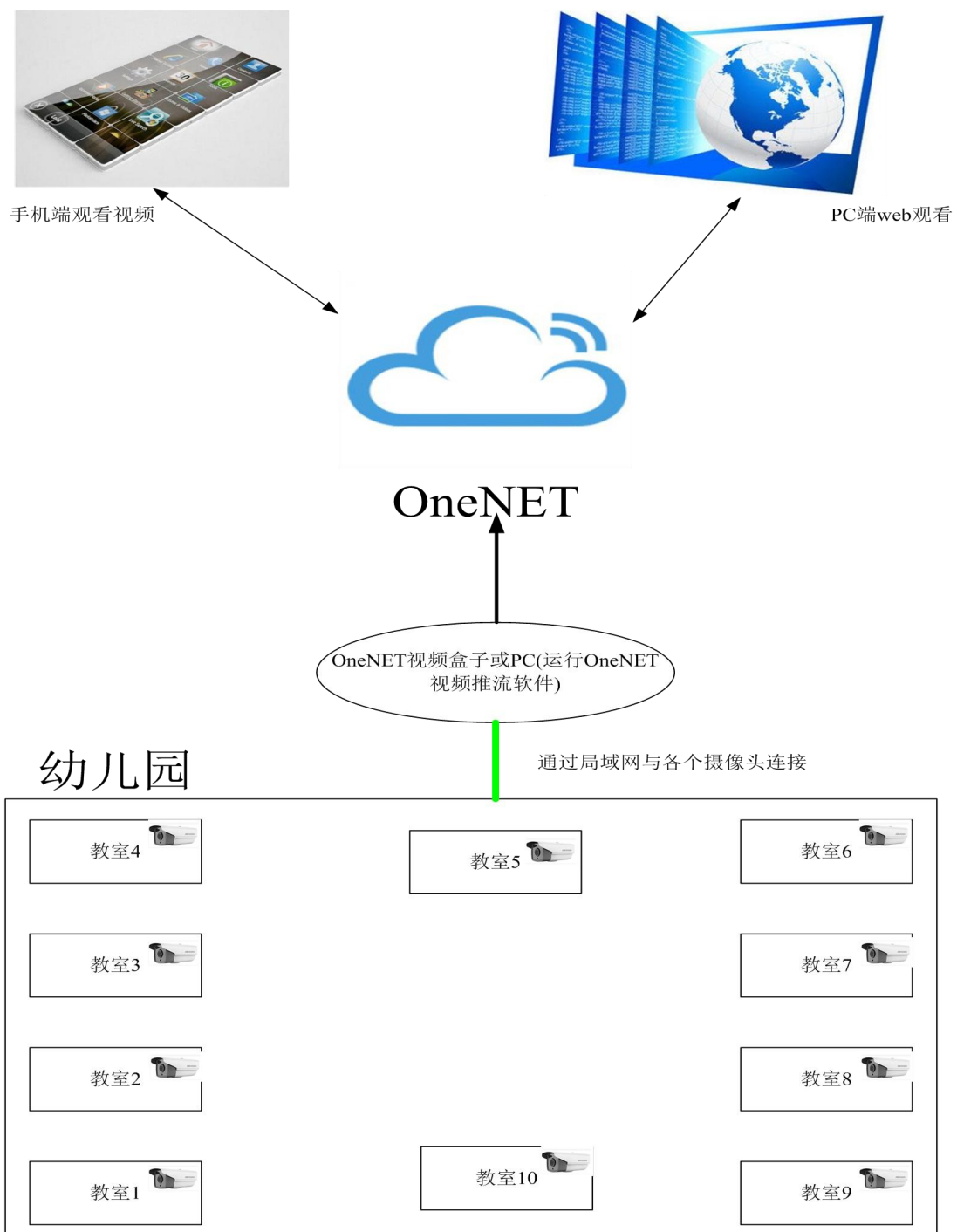
2.3 视频能力应用范例架构

通过 OneNET 视频能力快速搭建各种视频以及监控系统

- 架构举例 1

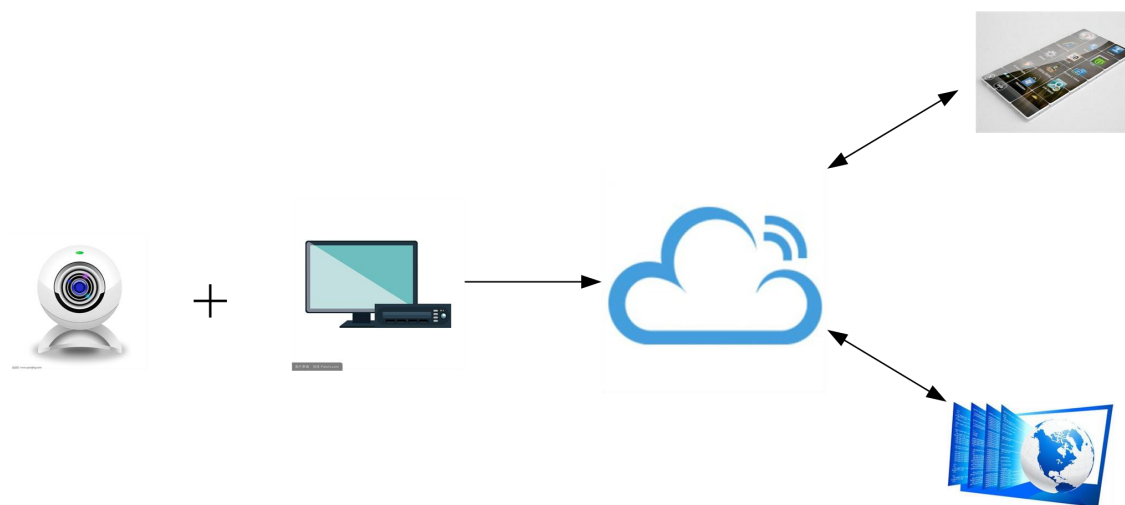
幼儿园网络视频系统--视频系统集成商可快速搭建基于 OneNET 的网络监控系统

（OneNET 视频 SDK 中提供了基于 onvif 网络摄像头的集成案例）



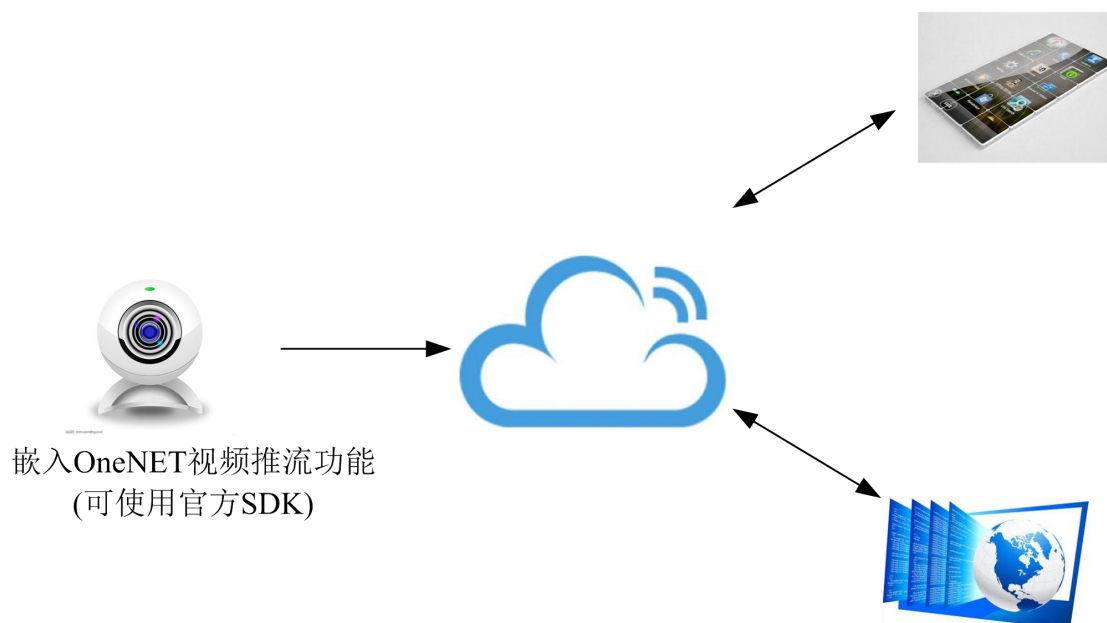
➤ 架构举例 2

直播系统



➤ 架构举例 3

摄像头监控



3 OneNET 视频应用

OneNET 提供 SDK 以帮助开发者和用户能够快速搭建基于 OneNET 视频能力的视频系统，SDK 中还包含丰富的案例供用户参考。

SDK 地址：https://github.com/cm-heclouds/video_sdk

3.1 基于 MQTT 协议 OneNET 视频推送

3.1.1 基于 MQTT 协议 OneNET SDK 介绍

SDK 目录结构：

- doc : SDK 相关文档
- include : SDK API 相关头文件
 - ont SDK 通用 API、错误码等
 - edp EDP 协议特有 API
 - mqtt MQTT 协议特有 API
- platforms : 平台相关的接口 (include/platform.h) 实现
 - posix 支持 Posix 系统的平台相关接口实现
 - win Windows 系统的平台相关接口实现
- Protocols : 相关协议的接口 (include/platform.h) 实现
 - Base
 - edp edp 协议实现
 - Jtext Jtext 协议实现
 - modbus Modbus 协议实现
 - mqttMqtt 协议实现
 - video 视频相关协议实现
- sample : 各协议的示例
 - edp EDP 协议的示例
 - jtext jtext 协议的示例
 - mqtt MQTT 协议的示例
 - videosample 网络摄像头推流以及历史(本地)视频推

SDK 接口介绍

a. 设备相关接口

接口名称	说明	备注
ont_device_create	创建设备实例接口	参考device.h
ont_device_connect	设备连接鉴权接口	参考device.h
ont_device_keepalive	设备保活接口	参考device.h
ont_device_get_cmd	接收指令接口	参考device.h
ont_device_cmd_destroy	指令销毁接口	参考device.h
ont_device_disconnect	断开连接接口	参考device.h
ont_device_destroy	销毁实例接口	参考device.h

b. 指令回调接口（平台移植接口）

接口名称	说明	备注
ont_videocmd_handle	处理指令接口	参考video_cmd.h
t_ont_video_live_stream_start	开始直播指令回调	参考video_cmd.h
t_ont_video_live_stream_ctrl	设置直播码流级别回调	参考video_cmd.h
t_ont_video_stream_make_keyframe	通知指定通道生成关键帧回调	参考video_cmd.h
t_ont_video_dev_ptz_ctrl	云台控制回调	参考video_cmd.h
t_ont_video_dev_query_files	本地视频查询回调	参考video_cmd.h
t_ont_video_vod_start_notify	开始一路视频回放回调	参考video_cmd.h

c. RTMP 实时视频相关接口

接口名称	说明	备注
rtmp_create_publishstream	创建RTMP推流实例接口	参考video.h
rtmp_stop_publishstream	关闭RTMP推流实例接口	参考video.h
rtmp_send_metadata	发送metadata接口	参考video.h
rtmp_send_spspps	发送sps数据接口	参考video.h
rtmp_send_videodata	发送视频帧接口	参考video.h
rtmp_send_audiodata	发送音频帧接口	参考video.h
rtmp_check_rcv_handler	RTMP接收处理接口	参考video.h
rtmp_make_audio_headerTag	创建flv audio header flag接口	参考video.h
ont_video_dev_add_channel	添加视频通道接口	参考video.h

d. 本地历史视频相关接口

接口名称	说明	备注
rtmp_rvod_createctx	创建历史视频实例接口	参考video_rvod.h
rtmp_rvod_destoryctx	销毁历史视频实例接口	参考video_rvod.h
rtmp_rvod_start	开始历史视频推流接口	参考video_rvod.h
rtmp_rvod_stop	结束历史视频推流接口	参考video_rvod.h
rtmp_rvod_send_videoeof	文件发送结束通知接口	参考video_rvod.h

3.1.2 基于 MQTT 的 OneNET SDK 案例介绍和使用

Sdk 中 videosample 案例完成了网络摄像头(支持 onvif 协议)和本地视频的推流, 用户可能没有网络摄像头, 可使用历史视频的功能查看本地视频文件。

进入 sdk 目录, 执行 videosamplebuild.sh 脚本, 完成编译, 可生成 videosample 样例执行程序(sample_video_s)。

```

root@ubuntu: /home/onenet/sdk/video_sdk
root@ubuntu:/home/onenet# ls
board          cmake-3.2.2.tar.gz  gm_lib_2015-01-09-IPCAM.tgz  toolchain_gnueabi-4.4.0_ARMv5TE.tgz
cmake-3.2.2    gm_graph            sdk
root@ubuntu:/home/onenet# cd sdk
root@ubuntu:/home/onenet/sdk# cd video_sdk/
root@ubuntu:/home/onenet/sdk/video_sdk# cd video
bash: cd: video: No such file or directory
root@ubuntu:/home/onenet/sdk/video_sdk# ls
bin          include  linux-arm.txt  protocols  sample          videosamplebuild-arm.sh
CMakeLists.txt  lib      platforms     README.md  video-plugin    videosamplebuild.sh
root@ubuntu:/home/onenet/sdk/video_sdk# sh videosamplebuild.sh

```

```

root@ubuntu:/home/onenet/sdk/video_sdk/bin# ls
config.json  lib  sample_mqtt_s  sample_video_s

```

样例程序的执行依赖于 config.json 配置文件，配置文件由三部分组成，profile 为产品相关，onvif 为网络摄像头信息，rvod 为本地视频信息

```

"profile":{
    "productid":113746,
    "pass":"KyuXupAEgJ8DjYvw"
},

```

```

"onvif": [
    {
        "title":"channel3",
        "channel_id":3,
        "url": "http://192.168.200.133:80/onvif/device_service",
        "user": "admin",
        "passwd": "test123456",
        "defaultlevel": 2
    },
    {
        "title":"channel2",
        "channel_id":2,
        "url": "http://192.168.200.222:80/onvif/device_service",
        "user": "admin",
        "passwd": "test123456",
    }
]

```

```

"rvod":
[
    {
        "channel_id":1,
        "location":"/home/onenet/sdk/video_sdk/bin/test.mp4",
        "beginTime":"2016-12-07 12:30:30",
        "endTime":"2016-12-07 13:30:37",
        "videoTitle":"test_1_7"
    }
]

```

使用命令 vim config.json，修改 config.json 配置文件，将 productid 和 pass 修改为之前新建的 onenet 产品号和注册码。如果由网络摄像头，请修改 onvif 相关配置。执行 ./sample_video_s 执行程序。进入 OneNET 开发者中心产品即可观看视频。

观看视频--进入设备管理，找到在线设备

产品概况

设备管理

权限管理

实时查询

设备管理

接入设备

在接入设备时，请将以下注册码写入到设备中，只用于设备注册

正式环境注册码：KyuXupAEgJ8DjYvw

复制

使用说明

设备数量：6个

视频通道数量：16个

设备接入协议：RTMP

Master-APIKey: tsIKIRy4eS=6nnlHHDxzrFxFcQc0=

【video-08:B9:CD:1C:10:55】

设备ID：24949864

创建时间：2018-01-26 14:44:24

【video-08:4D:76:82:F5:32】

设备ID：24880483

创建时间：2018-01-22 13:47:41

【video-00:00:FF:FF:00:00】

设备ID：24880402

创建时间：2018-01-22 13:37:07

【video-08:90:90:90:90:90】

设备ID：24879120

创建时间：2018-01-22 10:50:11

【test4】

设备ID：24813167

创建时间：2018-01-18 15:20:53

【videotest2】

设备ID：23352053

创建时间：2017-12-18 11:32:45

通道名称	通道ID	操作
test	1	<div>🔍 ⏮ ⏪ ⏩ ⏭</div>
	2	<div>🔍 ⏮ ⏪ ⏩ ⏭</div>
	3	<div>🔍 ⏮ ⏪ ⏩ ⏭</div>

点击播放实时视频看查看摄像头实时画面

设备管理

接入设备

在接入设备时，请将以下注册码写入到设备中，只用于设备注册

正式环境注册码：KyuXupAEgJ8DjYvw

复制

使用说明

设备数量：6个

视频通道数量：16个

设备接入协议：RTMP

Master-APIKey: tsIKIRy4eS=6nnlHHDxzrFxFcQc0=

【video-08:B9:CD:1C:10:55】

设备ID：24949864

创建时间：2018-01-26 14:44:24

【video-08:4D:76:82:F5:32】

设备ID：24880483

创建时间：2018-01-22 13:47:41

【video-00:00:FF:FF:00:00】

设备ID：24880402

创建时间：2018-01-22 13:37:07

【video-08:90:90:90:90:90】

设备ID：24879120

创建时间：2018-01-22 10:50:11

【test4】

设备ID：24813167

创建时间：2018-01-18 15:20:53

【videotest2】

设备ID：23352053

创建时间：2017-12-18 11:32:45

通道名称	通道ID	操作
test	1	<div>🔍 ⏮ ⏪ ⏩ ⏭</div>
	2	<div>🔍 ⏮ ⏪ ⏩ ⏭</div>
	3	<div>🔍 ⏮ ⏪ ⏩ ⏭</div>

播放实时视频

02-26-2018 星期一 10:51:26

摄像头方向设置

↑ ↓ ← →

视频设置(焦距)

- +

转动速度(1-7)

1

视频清晰度

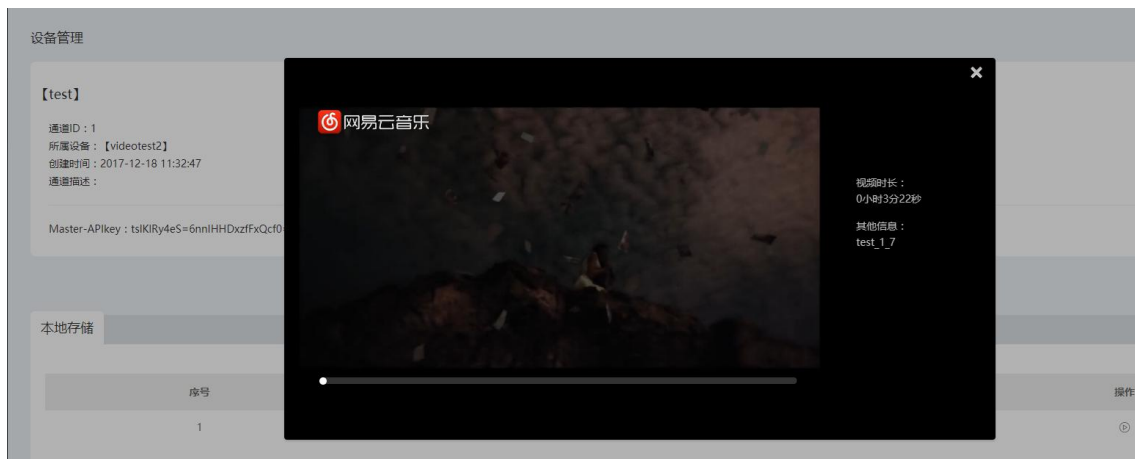
超清 高清 标清 流畅

点击查看历史视频按钮查询本地视频文件

本地存储

序号	视频名称	录制起始时间	操作
1	test_1_7	2016-12-07 12:30:30-2016-12-07 13:30:37	<div>🔍</div>

点击播放观看历史视频或本地视频



3.2 基于私有协议 OneNET 视频推送

3.2.1 基于私有协议 OneNET SDK 介绍

SDK 目录结构:

- bin : 可执行程序存放路径
- include : SDK API 相关头文件
- lib : 库文件路径
- platforms : 平台相关的接口 (include/platform.h) 实现
 - posix 支持 Posix 系统的平台相关接口实现
 - win Windows 系统的平台相关接口实现
- sample : onvif 推流和本地视频推流案例
 - Live live555、rtsp 相关代码
 - mp4v2 mp4 编解码相关代码
 - Onvif onvif 协议相关代码
- src : 平台命令、rtmp 相关代码
- test : 测试案例
- tool : 编译工具以及 mbedtls 库
- video-plugin 视频插件

SDK 接口介绍

接口名称	说明	备注
ont_device_get_acc	获取接入机	参考 device.h
ont_device_create	创建OneNET接入设备	参考 device.h
ont_device_verify	创建OneNET接入设备	参考 device.h
ont_device_connect	与OneNET建立连接	参考 device.h
ont_device_register	与OneNET建立连接	参考 device.h
ont_device_auth	设备鉴权	参考 device.h
ont_device_disconnect	* 断开与OneNET的连接	参考 device.h
ont_device_destroy	销毁OneNET接入设备实例	参考 device.h
ont_device_heartbeat	保活设备	参考 device.h
ont_device_request_rsa_publickey	设备请求RSA 公钥	参考 device.h
ont_device_replay_rsa_publickey	发送加密公钥	参考 device.h
ont_device_get_channels	获取设备已经添加的通道	参考 device.h
ont_device_del_channel	删除设备通道	参考 device.h
ont_device_add_channel	添加通道	参考 device.h
ont_device_check_receive	检查是否有命令需要处理	参考 device.h
ont_device_reply_ont_cmd	发送命令响应结果	参考 device.h
ont_device_reply_user_defined_cmd	发送命令响应结果	参考 device.h
ont_device_set_platformcmd_handle	设置平台命令处理函数	参考 device.h
ont_device_data_upload	设置平台命令处理函数	参考 device.h

接口名称	说明	备注
(*t_ont_heartbeat_resp_callback)(ont_device_t *dev)	设备保活回复处理	参考 device.h
(*t_ont_live_stream_start)	开始直播	参考 device.h
t_ont_vod_start_notify	开始一路视频回放	参考 device.h
t_user_define_cmd	用户自定义命令下发	参考 device.h
struct _ont_device_callback_t { t_ont_heartbeat_resp_callback heartbeat_resp; t_ont_live_stream_start live_start; t_ont_vod_start_notify rvod_start; t_user_define_cmd user_defined_cmd; };	设备回调函数结构体	参考 device.h

接口名称	说明	备注
(*t_ont_keepalive_resp_callback)(ont_device_t *dev)	设备保活回复处理	参考device.h
(*t_ont_live_stream_start)	开始直播	参考device.h
t_ont_vod_start_notify	开始一路视频回放	参考device.h
t_user_defind_cmd	用户自定义命令下发	参考device.h
struct _ont_device_callback_t { t_ont_keepalive_resp_callback keepalive_resp; t_ont_live_stream_start live_start; t_ont_vod_start_notify rvod_start; t_user_defind_cmd user_defined_cmd; };	设备回调函数结构体	参考device.h

接口名称	说明	备注
ont_security_init	初始化加密库	参考security.h
ont_security_deinit	销毁加密库	参考security.h
ont_security_rsa_generate	创建rsa加解密实例	参考security.h
ont_security_rsa_get_pubkey	销毁rsa加解密实例	参考security.h
ont_security_rsa_encrypt	加密数据	参考security.h
ont_security_rsa_decrypt	RSA解密数据	参考security.h

3.2.2 基于私有协议的 OneNET SDK 案例介绍和使用

私有协议 SDK 的案例和 MQTT 协议的案例功能相同，可参考 3.1.2.

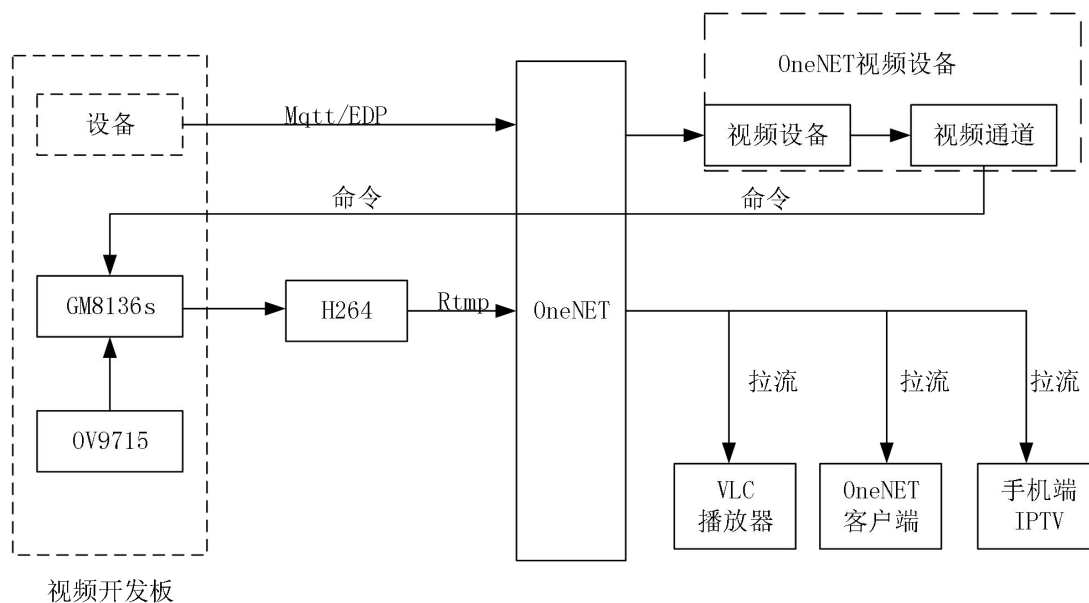
```
root@ubuntu:/onenet/video/sdk/private/video_sdk-master/tools# ls
mbedtls-2.6.0-apache.tgz  videosamplebuild.sh
videosamplebuild-mp4v2.sh  videotestbuild.sh
```

- sh videosampbuild-mp4v2.sh 编译本地视频推送程序
- sh videosample.sh 编译 onvif 摄像头视频推送程序
- sh videotestbuild.sh 编译 sdk 使用样例程序

3.3 基于开发板与 OneNET 的视频推送直播

3.3.1 基础介绍

下图为开发板视频 OneNET 推送体验架构图，主要包含 OneNET 视频设备管理、GM8136s 的视频采样与 rtmp 推流、各种客户端的数据拉流播放三个部分。



➤ OneNET 视频设备管理：

该部分主要是通过创建 OneNET 视频设备，为开发板与平台的建立交互通道，完成开发板数据上传和 OnNET 命令下发。视频设备创建包括设备创建和通道创建，创建完成后在 OneNET 开发者中心可见到如下界面。通过界面可以完成视频播放、历史查询以及包括设备建立、删除等各种设备基本操作。

【videotest2】 设备ID：23352053 创建时间：2017-12-18 11:32:45

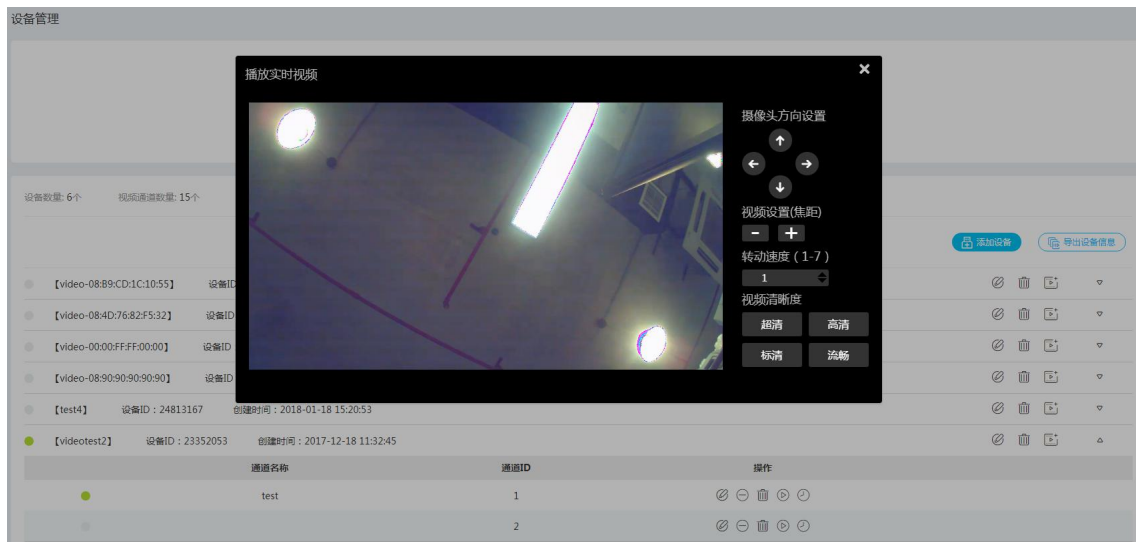
通道名称	通道ID	操作
test	1	🔍 ⏮ ⏪ ⏩ ⏭
	2	🔍 ⏮ ⏪ ⏩ ⏭

➤ GM8136s 的视频采样与 rtmp 推流

GM8136s 是开发板的核心，完成了视频的采样编码以及视频流推送。视频的数据采样编码主要依赖于 GM8136s 提供的开发库，其可直接输出包括 H264、MPEG 等各种格式的视频数据，视频推送可以直接使用 sdk 提供 rtmp 接口和自主设计的 rtmp 接口。

➤ 各种客户端的数据拉流播放

拉流播放可通过各种途径实现，主要通过第三方工具。比如 VLC media player、手机端 IPTV 播放器以及 OneNET 的开发者中心提供的播放功能。



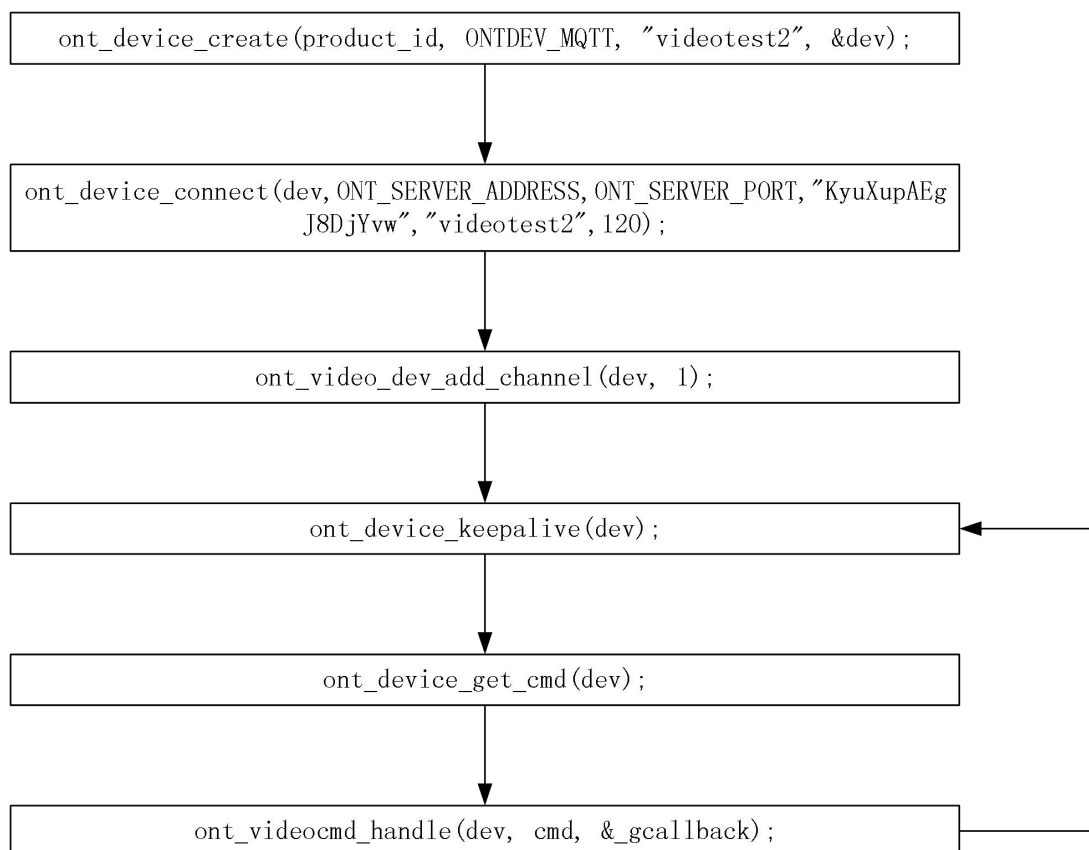
3.3.2 程序设计

程序主要包含视频设备管理、视频采样、视频推流三大部分组成。

► 视频设备管理

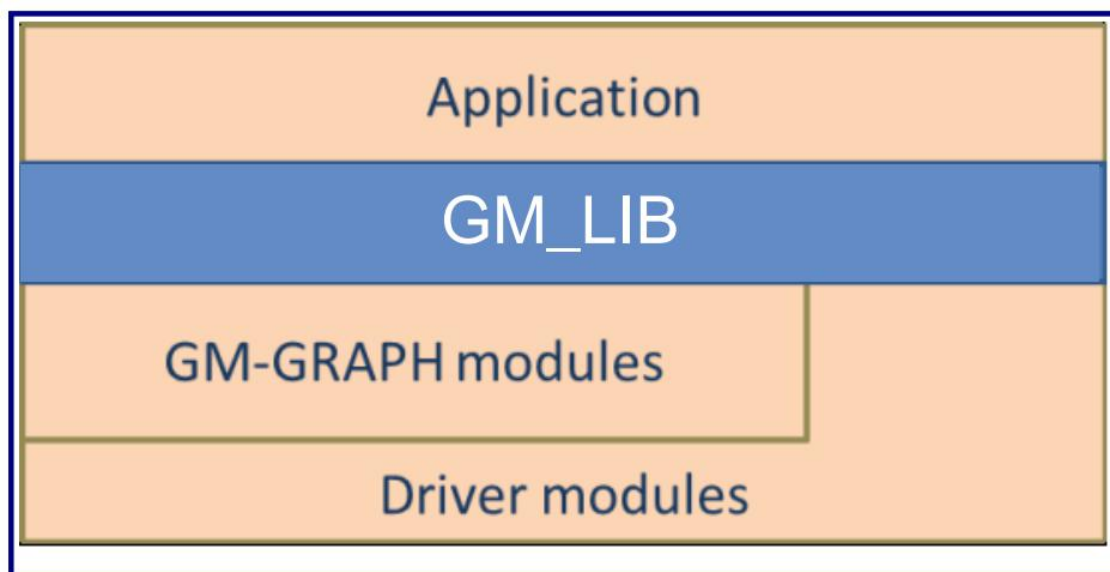
视频设备管理的接口函数可以查看

<https://open.iot.10086.cn/doc/art386.html#68>，主要查看视频能力中的管理控制台和接入 SDK，二者可通过对比手动和自动创建设备流程，查看各个接口函数的功能。下图为视频设备的创建流程以及主程序架构。



➤ 视频采样

视频采样主要依赖于 GM8136s 平台提供的 GM_LIB 开发库实现。GM_LIB 为应用程序访问 GM8136s 芯片的各种功能提供接口。通过使用该库，可以实现很多功能，包括直播、录像、回放等。软件层面，GM8136s 平台可分为四个层次，包括应用层、GM_LIB 库层、GM 图像模块、linux 驱动模块。



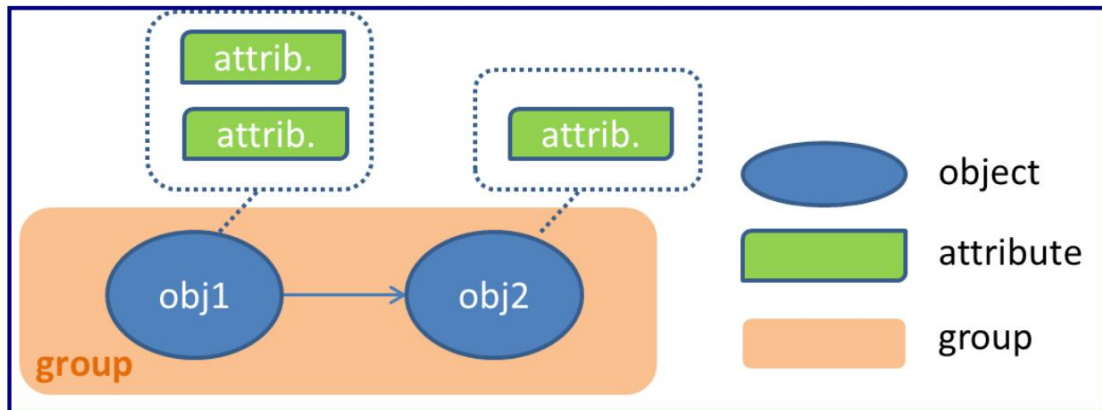
应用层：应用层主要是用户的应用程序。用户应用程序可通过 GM_LIB 完成各种功能。

GM_LIB 层：负责应用层与图像层以及驱动层的交互

GM_GRAPH 层：主要完成图像任务调度以及一些硬件初始化工作

Driver modules 层：linux 驱动层，完成与硬件的交互。

GM_LIB 将每一个输入输出作为一个对象，比如视频采集是一个对象、视频显示也是一个对象、文件的输入流和输出流也分别是对象。用户应用程序可对每一个对象进行属性设置，并将需要的对象的进行绑定(group)从而实现一个应用程序功能。



GM8136 的编程一般按照如下流程：

开始流程

建立对象 object

设置对象属性

建立对象组

绑定对象到对象组

应用对象组

发送或接收数据流，完成需要的功能

结束流程

停止数据流的发送和接收

解绑对象

应用到对象组

删除对象和对象组

比如本地直播

```

#include "gmlib.h"

gm_init(); _____ 库的初始化
groupfd = gm_new_groupfd(); _____ 建立对象组
cap_obj = gm_new_obj(GM_CAP_OBJECT); _____ 建立摄像头对象
win_obj = gm_new_obj(GM_WIN_OBJECT); _____ 建立显示窗口对象
...
gm_set_attr(cap_obj, &capAttr); _____ 设置摄像头对象属性
...
gm_set_attr(win_obj, &winAttr); _____ 设置显示窗口对象属性
...
bindfd = gm_bind(groupfd, cap_obj, win_obj); _____ 绑定摄像头对象和显示窗口对象到对象组
gm_apply(groupfd); _____ 应用对象组

```

比如录像应用

```

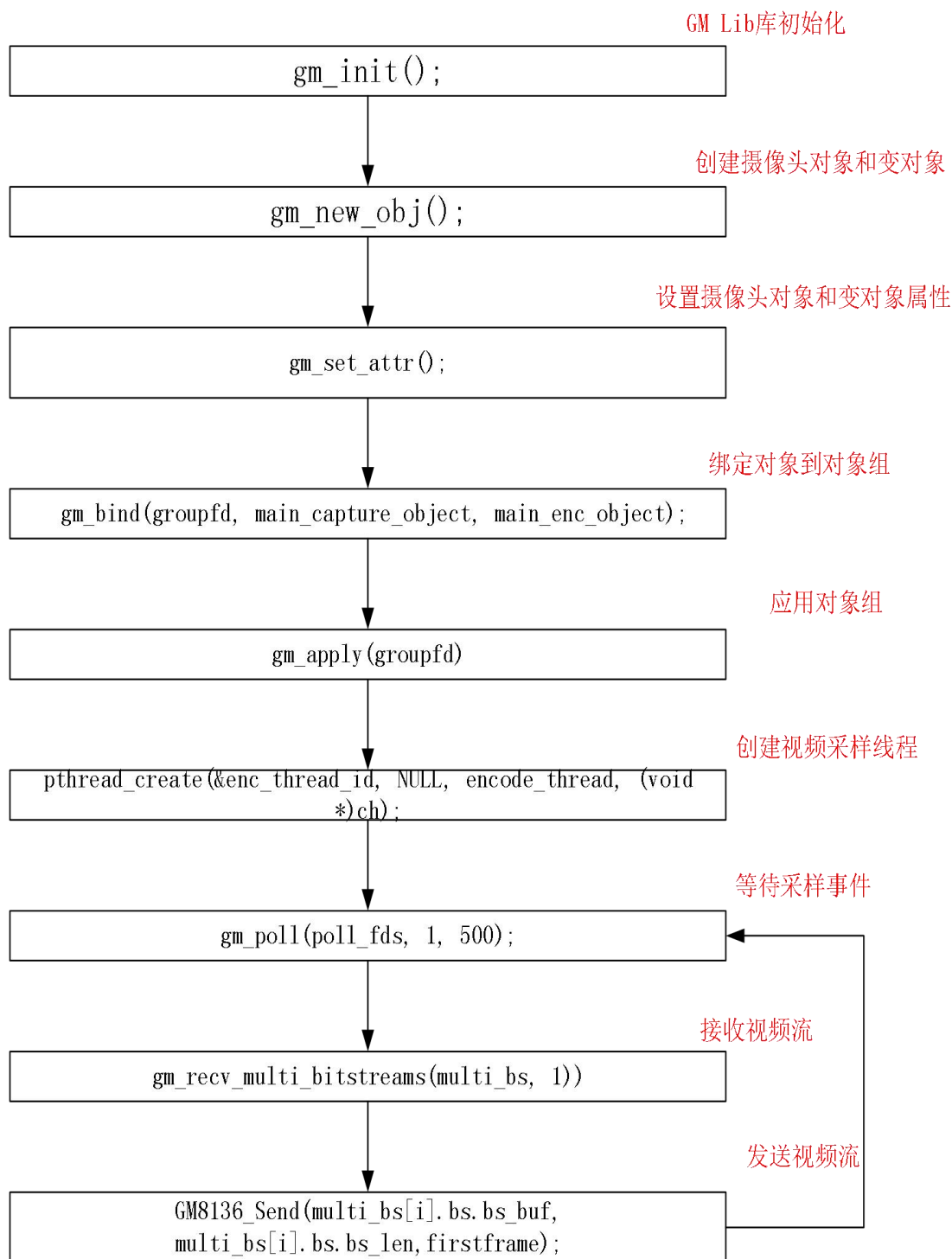
gm_init(); _____ GM Lib初始化
groupfd = gm_new_groupfd(); _____ 建立对象组
cap_obj = gm_new_obj(GM_CAP_OBJECT); _____ 建立摄像头对象
enc_obj = gm_new_obj(GM_ENCODER_OBJECT); _____ 建立编码对象
...
gm_set_attr(cap_obj, &capAttr); _____ 设置摄像头对象属性
...
gm_set_attr(enc_obj, &encAttr); _____ 设置编码对象属性
...
bindfd = gm_bind(groupfd, cap_obj, enc_obj); _____ 绑定摄像头对象和编码对象到对象组
gm_apply(groupfd); _____ 应用对象组

while(...) {
    gm_poll(bindfd, ...);
    ...
    gm_recv_multi_bitstreams(multi_enc, ...); _____ 接收视频流
}

```

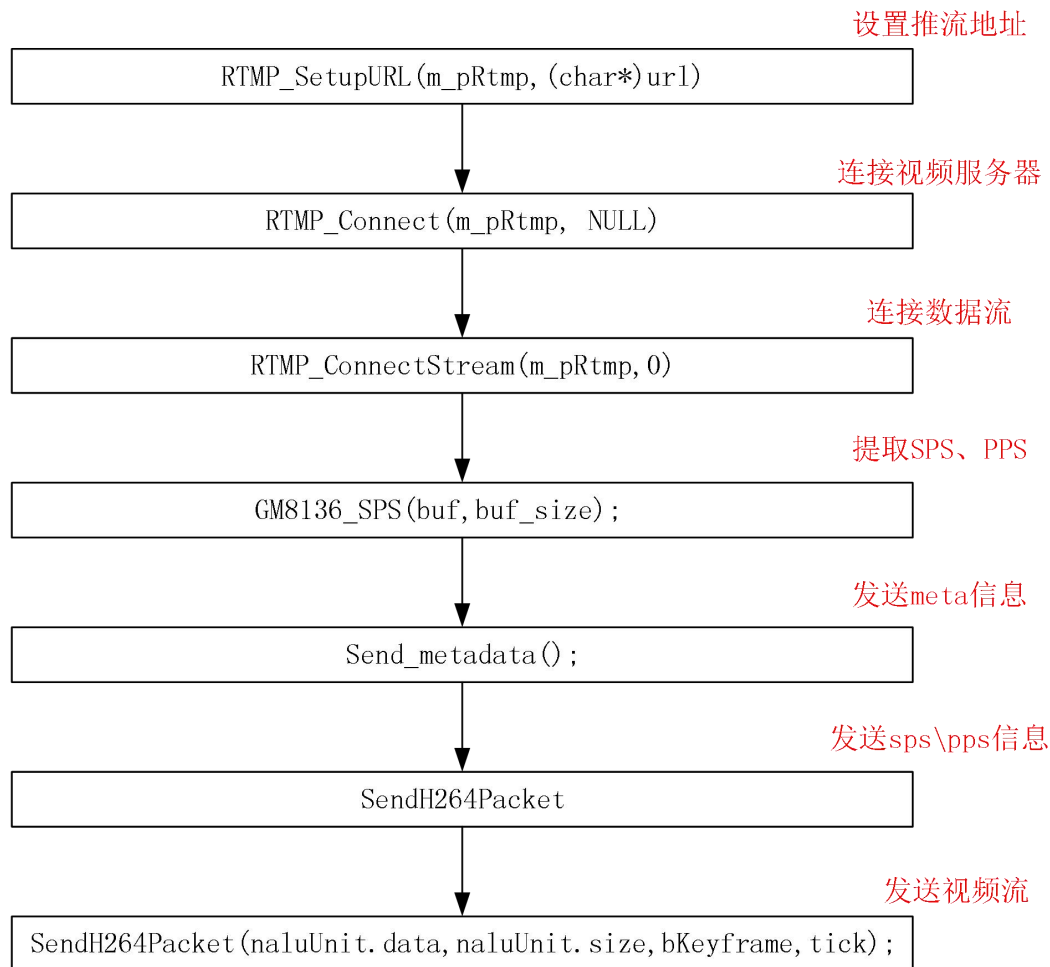
更详细内容可参考 GMLIB_Programming_Guide_V1.10 文档。

视频采集的程序流程和上述举例的流程并无大的不同，流程如下图：



➤ 视频推送

视频推送主要通过 `rtmpdump` 库完成，大致流程如下：



3.4 基于开发板与 OneNET 的音视频推送直播

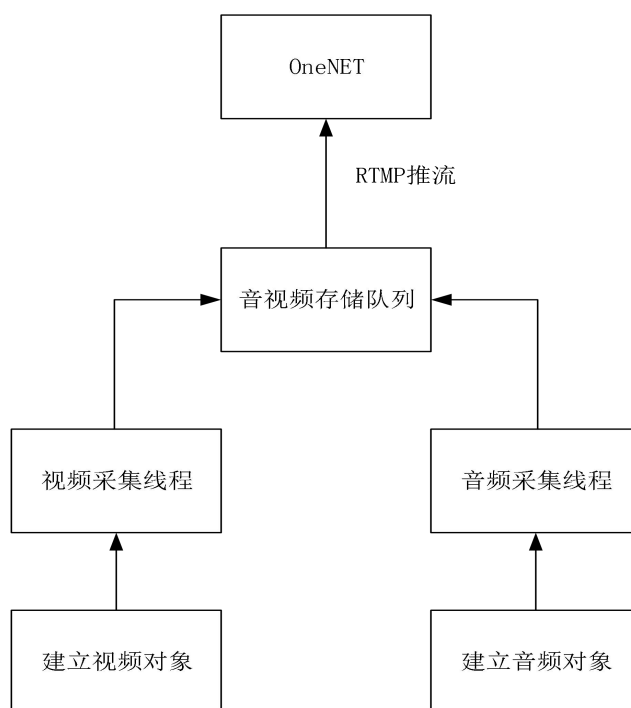
3.4.1 基础介绍

上一章节中主要介绍了视频推送，本章节介绍音视频同步推送。

主要过程在上一章节介绍的基础还需增加音频采集、音视频同步、音视频推送三个过程。音频采集流程如下：

- 1) 建立音频对象 object
- 2) 设置对象属性
- 3) 建立对象组
- 4) 绑定对象到对象组
- 5) 应用对象组
- 6) 开始采集

3.4.2 程序框架设计



在与 OneNET 完成连接和设备交互并受到 `rtmp` 地址时，开始推流过程。分别建立视频对象和音频对象，建立独立线程完成视频数据和音频数据采集，并将音视频数据储存到缓存队列等待推送线程调用完成推流。

4 OneNET EDP 协议

4.1 EDP 协议介绍

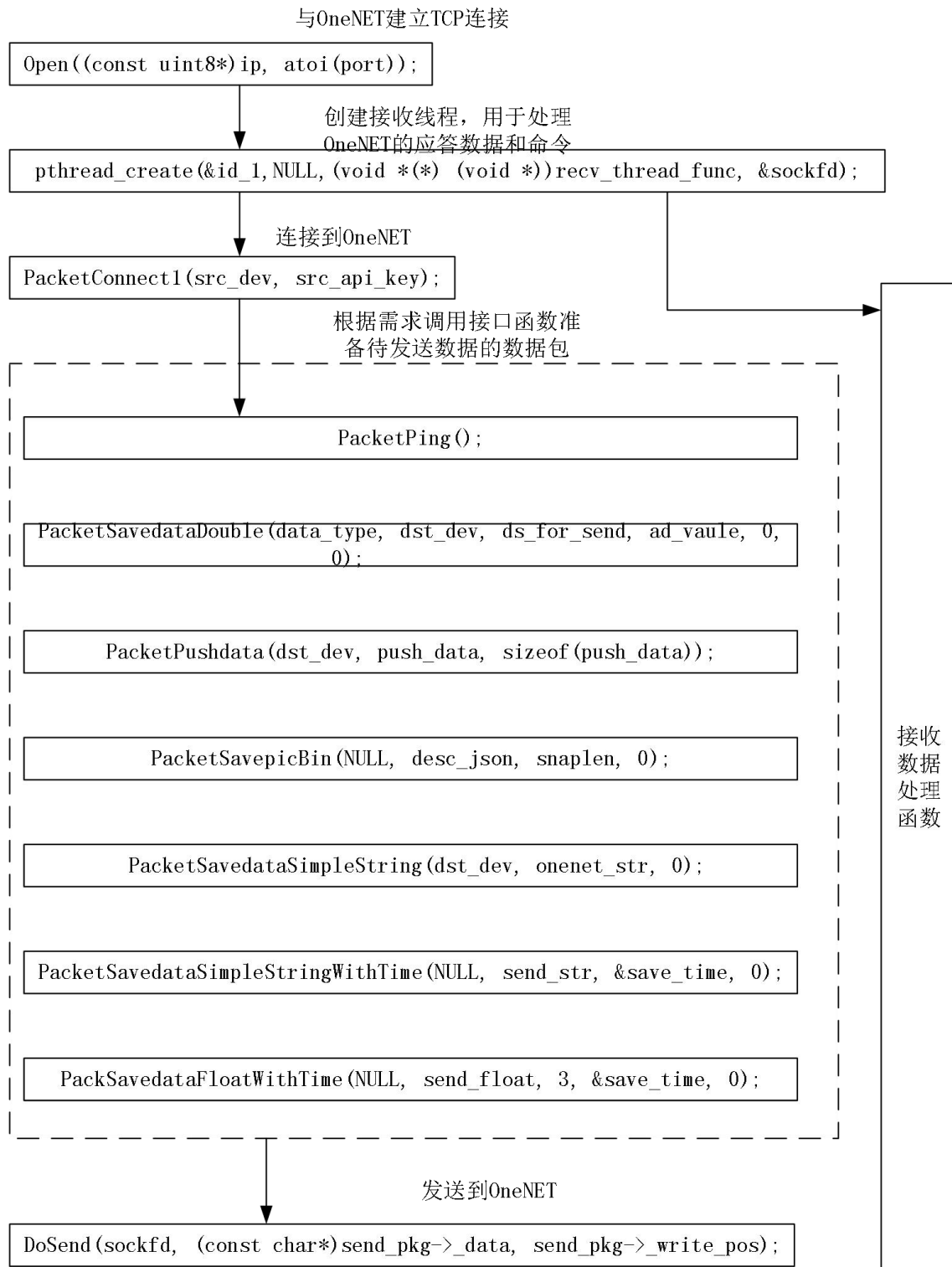
EDP (Enhanced Device Protocol 增强设备协议) 是 OneNET 平台根据物联网特点专门定制的完全公开的基于 TCP 的协议，可以广泛应用到家居、交通、物流、能源以及其他行业应用中。关于协议的具体内容请查看：

<https://open.iot.10086.cn/doc/art254.html#68>

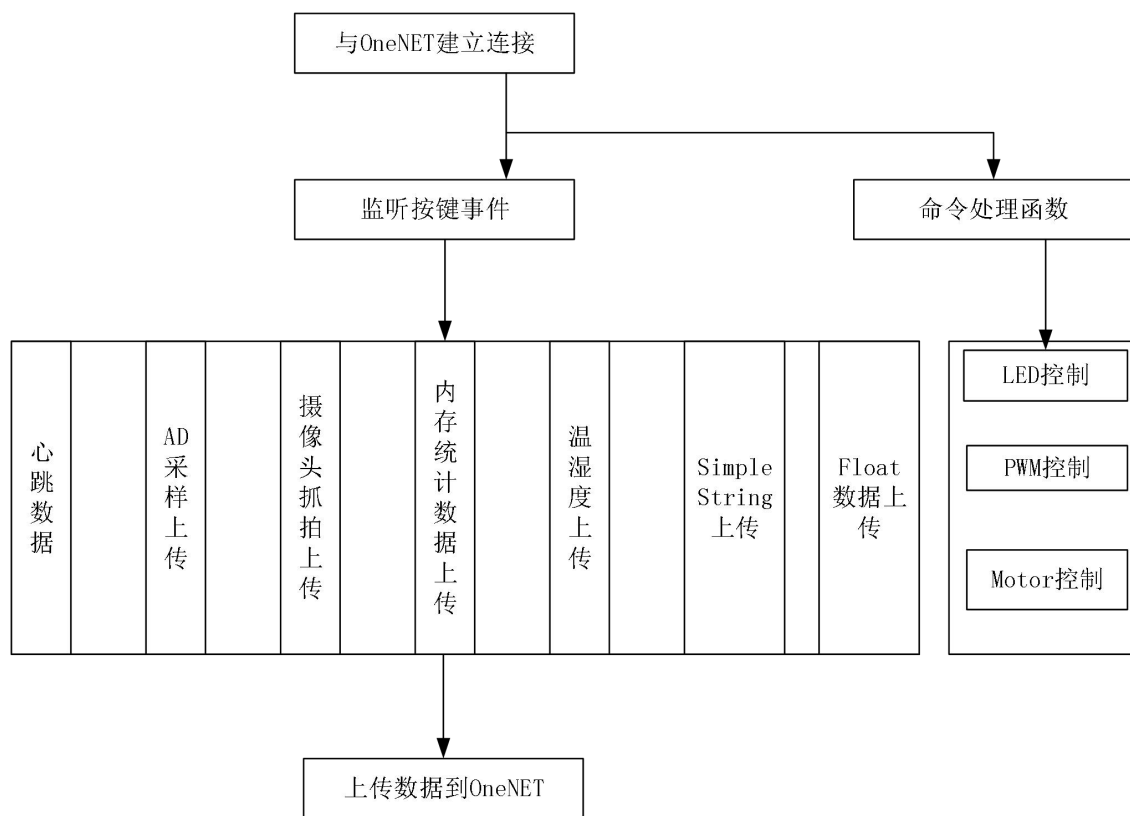
4.2 程序设计

示例程序主要是让用户熟悉 EDP 协议以及 EDP SDK 的使用。示例采用 EDP 协议，根据开发板的按键实现了：AD 采样数据上传、摄像头抓拍图片数据上传，开发板内存信息上传以及各种类型数据的上传。

➤ 基于 EDP SDK 上传数据到 OneNET 流程



➤ 示例程序流程



5 OneNET HTTP 协议

5.1 OneNET HTTP 协议介绍

OneNET 支持设备采用遵循 HTTP 协议的数据封装结构以及接口形式等连接平台进行数据传输，用户可以实现终端数据的上传和保存。HTTP 协议主要使用 Restful 接口，主要用于业务层的使用，但依然可用于设备端数据的上传。HTTP 协议采用的短连接，因此只能上报数据，不能从 OneNET 平台向设备端下发命令。

5.2 程序设计

➤ 基于 HTTP SDK 上传数据到 OneNET

HTTP SDK 对 TCP 通信和 OneNET 设备以及数据流等进行了底层封装，用户可以直接调用相应接口实现设备创建、数据流创建、数据上传、APIKey 创建、以及更新信息等功能。

1) 创建设备

配置设备结构体

typedef struct

```
{  
    /** 设备名，用户范围内应唯一*/  
    int8 deviceTitle[DEVICE_NAME_LEN_MAX + 1];  
    /** 设备描述*/  
    int8 deviceDesc[DEVICE_DESC_LEN_MAX + 1];  
    /** 设备标签表，可填写多个标签，以逗号隔开*/  
    int8 deviceTagList[DEVICE_TAGLIST_LEN_MAX + 1];  
    /** 具有操作权限的 APIKey。对于新增设备必须使用 MasterKey*/  
    int8 APIKey[DEVICE_API_KEY_LEN_MAX + 1];  
    int8 REGISTER_CODE[DEVICE_API_KEY_LEN_MAX + 1];  
    int8 SN[DEVICE_DESC_LEN_MAX + 1];  
    /** 设备位置信息*/  
    LocationInfo locInfo;  
    /** 是否为私有设备*/  
    boolValue isPrivate;
```

```
} RFDeviceConfig, *pRFDeviceConfig;
```

```
RFDeviceConfig config;
```

```
memset(&config, 0, sizeof(config));
```

```
    strcpy(config.APIKey, MASTER_KEY);
```

```
    strcpy(config.REGISTER_CODE, "5nqkSDyQ5mP2hojG");
```

```
    strcpy(config.SN, "201802021148");
```

```
    strcpy(config.deviceTitle, DEVICE_TITLE);
```

```
    strcpy(config.deviceDesc, "http create device");
```

```
    strcpy(config.deviceTagList, TAGLIST);
```

```
    memcpy(&(config.locInfo), &loc, sizeof(LocationInfo));
```

```
    config.isPrivate = true;
```

调用 RFDevice_Create(&config, deviceid)接口创建设备

2) 创建数据流

```
typedef struct
```

```
{
```

```
    /** 数据流名称*/
```

```
    int8 streamID[STREAM_ID_LEN_MAX + 1];
```

```
    /** 数据流标签表，可填写多个，以逗号隔开*/
```

```
    int8 streamTagList[STREAM_TAGLIST_LEN_MAX + 1];
```

```
    /** 数据流数据单位*/
```

```
    int8 unit[STREAM_UNIT_LEN_MAX + 1];
```

```
    /** 数据流数据单位符号*/
```

```
    int8 unitSymbol[STREAM_UNITSYMBOL_LEN_MAX + 1];
```

```
} RFStreamConfig, *pRFStreamConfig;
```

```
RFStreamConfig sConfig;
```

配置数据流

调用 RFStream_Create(&sConfig, MASTER_KEY, streamUUID)创建数据流

3) 上传数据

```
typedef struct
```

```

{
    /** 数据点所属的数据流名称*/
    int8 streamID[STREAM_ID_LEN_MAX + 1];
    /** 数据点值*/
    int8 value[DATAPOINT_VALUE_LEN_MAX + 1];
    /** 数据点数据类型，参见 DATAPOINT_VALUE_TYPE_XXX，目前支持字符串和整型两种*/
    int32 valueType;
    /** 数据点创建时间，从公元 1970 年 1 月 1 日 0 时 0 分 0 秒的 UTC 时间算起所经过的秒数。
        * 设为 0 表示不添加时间参数，由云端确定*/
    uint32 time;
} RFDataPoint, *pRFDataPoint;

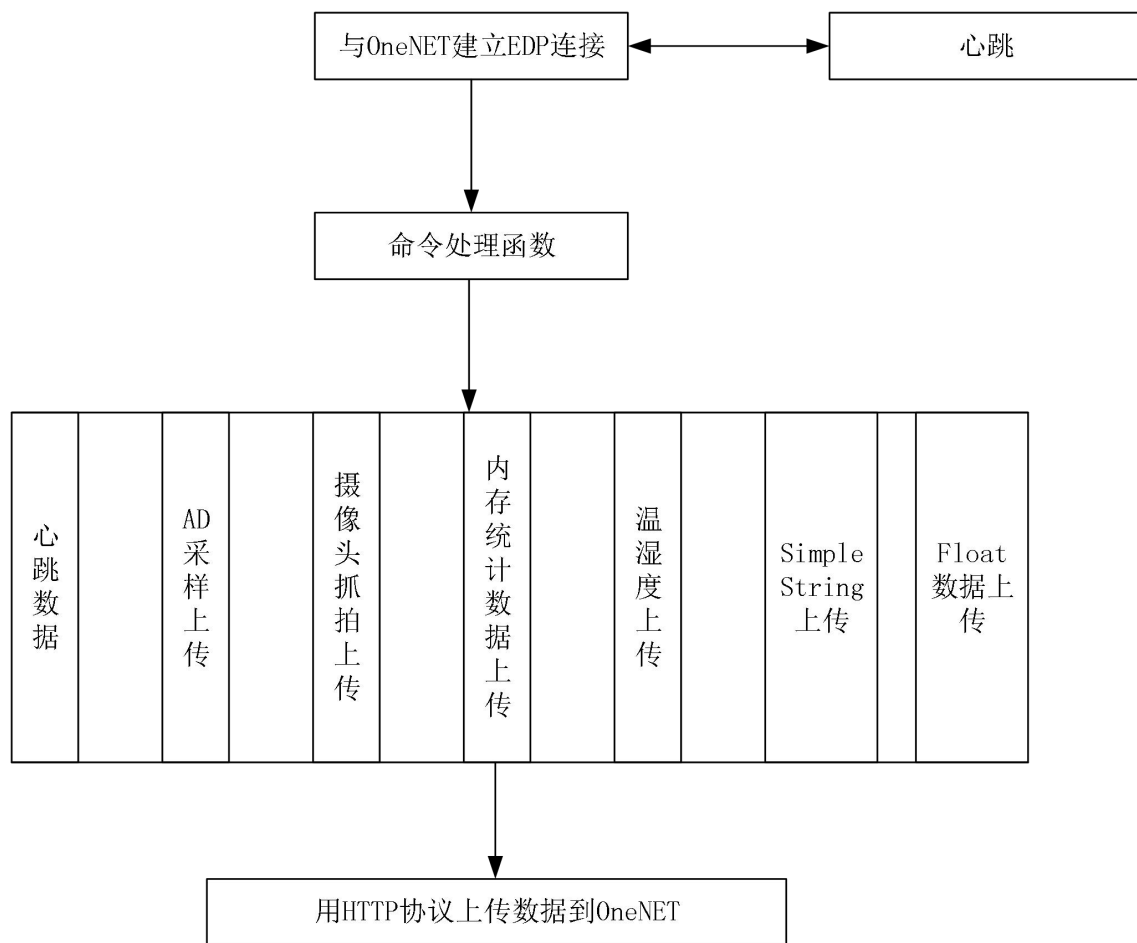
RFDataPoint data[5];
配置数据点
for(i = 0; i < 5; i++)
{
    strcpy(data[i].streamID, "TestStreamOps");
    data[i].time = 0;
    data[i].valueType = DATAPOINT_VALUE_TYPE_NUMBER;
    *(((double *)(data[i].value)) = i*3;
}

```

调用 RFDataPoint_Create(config.APIKey, data, 5)接口上传数据流

示例程序设计

示例程序同时应用 EDP 和 HTTP 协议来示范 OneNET 的命令下发和数据上传功能。以 EDP 协议构建设备并接收 OneNET 下发命令，以 HTTP 协议完成各种功能以及上传各种数据。



6 OneNET MQTT 协议

6.1 MQTT 协议介绍

MQTT 协议是一个面向物联网应用的即时通信协议，使用 TCP/IP 提供网络连接，能够对负载内容实现消息屏蔽传输，开销小，可以有效降低网络流量。

特点及功能

■ 长连接协议

终端数据点上报，支持的数据点类型包括

- 整型 (int)
- 浮点数 (float)
- 字符串 (string)
- JSON 格式

■ 平台消息下发

■ 基于 Topic 的订阅、发布以及消息推送，可以实现设备间的消息单播以及组播
详细介绍请访问

<https://open.iot.10086.cn/doc/art253.html#68>

使用介绍可参考论坛中的帖子

<https://open.iot.10086.cn/bbs/forum.php?mod=viewthread&tid=948&highlight=MQTT>

6.2 Mqtt SDK 介绍

6.2.1 在项目中链接 MQTT SDK 策略

项目中可通过使用源码或链接动态库两种方式使用 MQTT SDK。使用 SDK 可仿照下面步骤使用 SDK 完成与 OneNET 的 MQTT 通信。

- a. 初始化 MQTT SDK 上下文
- b. 与服务器建立连接
- c. 发布数据点
- d. 订阅用户自定义 topic
- e. 取消数据流订阅
- f. 处理服务器消息

6.2.2 Mqtt SDK 基本使用方法

a. 初始化 MQTT SDK 上下文

调用 `Mqtt_InitContext` 初始化 `MqttContext`，并将设置 `MqttContext` 中的回调函数及关联参数。如 `sample` 中的 `MqttSample_Init` 函数：

```
ctx->mqttctx->handle_conn_ack = MqttSample_HandleConnAck;
ctx->mqttctx->handle_conn_ack_arg = ctx;
ctx->mqttctx->handle_ping_resp = MqttSample_HandlePingResp;
ctx->mqttctx->handle_ping_resp_arg = ctx;
```

b. 与服务器建立连接

1. 创建 `MqttBuffer`，并通过 `MqttBuffer_Init` 进行初始化。

2. 调用 `Mqtt_PackConnectPkt`，封装 MQTT 连接包。

需要注意的是：`id` 需设置为设备 ID，`user` 需设置为 project ID，`password` 需设置为 auth-info

3. 调用 `Mqtt_SendPkt` 发送 MQTT 连接包

4. 调用 `MqttBuffer_Destroy` 销毁 MQTT 连接包

5. `will_topic`, `will_msg`, `msg_len`, `will_retain` 暂不支持，分别设为：`NULL`, `NULL`, `0`, `0`

代码示例：

```
...
MqttBuffer_Init(ctx->mqttbuf);
...
err = Mqtt_PackConnectPkt(ctx->mqttbuf, keep_alive, ctx->devid, 1,
                          NULL, NULL, 0,
                          MQTT_QOS_LEVEL0, 0, ctx->proid,
                          auth_info, strlen(auth_info));
if(MQTTErr_NOERROR != err) {
    // do some error handling.
}
...
bytes = Mqtt_SendPkt(ctx->mqttctx, ctx->mqttbuf, 0);
if(bytes < 0) {
    // do some error handling.
```

```
}
```

```
...
```

```
MqttBuffer_Destroy(ctx->mqttbuf);
```

```
...
```

c. 发布数据点

1.创建 MqttBuffer, 并通过 MqttBuffer_Init 进行初始化。

2.调用 Mqtt_PackDataPointBy*封装数据包

3.调用 Mqtt_SendPkt 发送数据点

4.调用 MqttBuffer_Destroy 销毁 MqttBuffer

代码示例:

```
...
```

```
MqttBuffer_Init(ctx->mqttbuf);
```

```
...
```

```
const char *str = ",;temperature,2015-03-22 22:31:12,22.5;102;pm2.5,89;10";
```

```
uint32_t size = strlen(str);
```

```
int retain = 0;
```

```
int own = 1;
```

```
int err = MQTTERR_NOERROR;
```

```
err = Mqtt_PackDataPointByString(ctx->mqttbuf, g_pkt_id++, 0, kTypeString, str,  
size, qos, retain, own);
```

```
if(err) {
```

```
    // do some error handling
```

```
}
```

```
bytes = Mqtt_SendPkt(ctx->mqttctx, ctx->mqttbuf, 0);
```

```
if(bytes < 0) {
```

```
    // do some error handling.
```

```
}
```

```
...
```

```
MqttBuffer_Destroy(ctx->mqttbuf);
```

```
...
```

d. 订阅用户自定义 topic

1.创建 MqttBuffer, 并通过 MqttBuffer_Init 进行初始化

2.调用 Mqtt_PackSubscribePkt 封装订阅消息包

3.调用 Mqtt_SendPkt 发送订阅消息包

4.调用 MqttBuffer_Destroy 销毁 MqttBuffer

代码示例:

```
...
```

```
MqttBuffer_Init(ctx->mqttbuf);
```

```
...
```

```
char **topics;
```

```
...
```

```
err = Mqtt_PackSubscribePkt(ctx->mqttbuf, 1, MQTT_QOS_LEVEL1, topics,  
topics_len);
```

```
if(err != MQTTERR_NOERROR) {
```

```
    // do some error handling.
```

```
}
```

```
bytes = Mqtt_SendPkt(ctx->mqttctx, ctx->mqttbuf, 0);
```

```
if(bytes < 0) {
```

```
    // do some error handling.
```

```
}
```

```
...
```

```
MqttBuffer_Destroy(ctx->mqttbuf);
```

```
...
```

e. 取消数据流订阅

1.创建 MqttBuffer, 并通过 MqttBuffer_Init 进行初始化

2.调用 Mqtt_PackUnsubscribePkt 封装订阅消息包

3.调用 Mqtt_SendPkt 发送订阅消息包

4.调用 MqttBuffer_Destroy 销毁 MqttBuffer

代码示例:

```
...
MqttBuffer_Init(ctx->mqttbuf);
...
char **topics;
...
err = Mqtt_PackUnsubscribePkt(ctx->mqttbuf, 1, topics, topics_len);
    if(err != MQTTERR_NOERROR) {
        // do some error handling.
    }

bytes = Mqtt_SendPkt(ctx->mqttctx, ctx->mqttbuf, 0);
if(bytes < 0) {
    // do some error handling.
}
...
MqttBuffer_Destroy(ctx->mqttbuf);
...
```

f. 处理服务器消息

1. 调用 Mqtt_RecvPkt 接收服务器消息, 当收到完整的消息后, Mqtt_RecvPkt 自动调用对应的消息处理函数

2. 在相应的服务器消息处理函数中处理消息

代码示例:

```
switch(err = Mqtt_RecvPkt(ctx->mqttctx)) {
case MQTTERR_ENDOFFILE:
    printf("The connection is disconnected.\n");
    close(ctx->mqttfd);
    epoll_ctl(ctx->epfd, EPOLL_CTL_DEL, ctx->mqttfd, NULL);
```

```

        ctx->mqttfd = -1;
        return 0;

case MQTTERR_IO:
    printf("Send TCP data error: %s.\n", strerror(errno));
    return -1;
case MQTTERR_NOERROR:
    return 0;

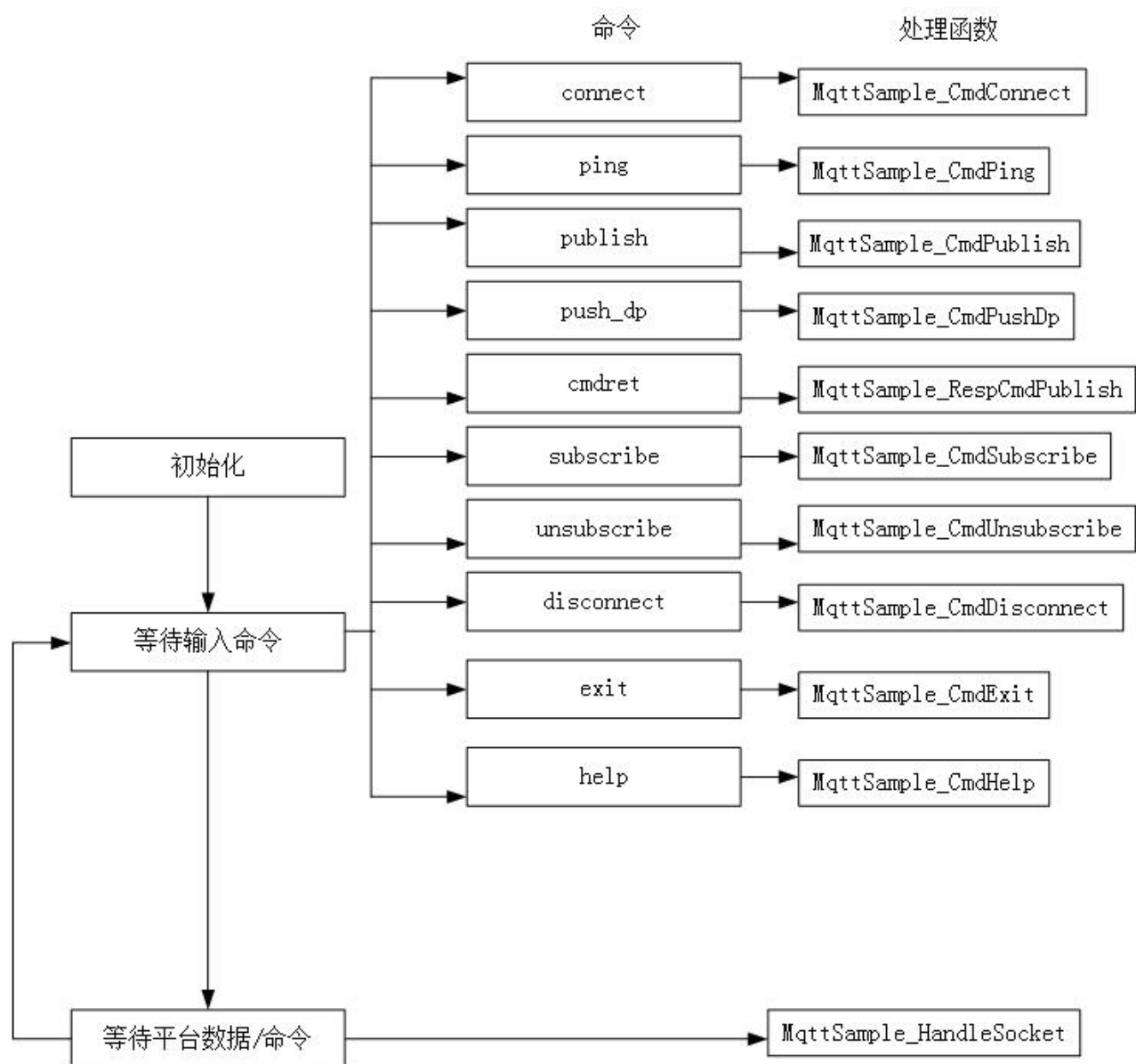
default:
    printf("Mqtt_RecvPkt error is %d.\n", err);
    return -1;
}

static int MqttSample_HandleConnAck(...)
{
    // do something
    return 0;
}

```

6.3 程序设计

示例程序以 SDK 的 `sample` 程序为模板，在 `publish` 命令中增加了各种数据上传的样例，包括二进制图片数据上传。



附录 1 嵌入式 Linux 应用开发介绍

请先参考附录 1 1.3.2 中的 VM 安装和 ubuntu 安装步骤，在虚拟机上安装 linux 操作系统

1.1 Linux 基本 shell 命令

Linux 操作系统虽然也具有丰富的窗口界面，但很多人依然习惯于使用 shell 命令 (类似于 windows 下的 cmd 命令行)使用 linux 系统完成各种任务。

1.1.1 命令：ls [选项][目录]

功能：查看指定工作目录下的内容

选项：

-a 显示出所有文件及目录

-l 列出所有文件详细的参数

例如：

```
ls -l /home
```

显示/home 目录下的文件和目录的详细信息

1.1.2 命令 pwd

功能：显示当前工作目录的绝对路径

选项：

无

例如：

```
pwd
```

1.1.3 命令：cd [路径]

功能：切换到指定目录路径下

选项：

无

例如：

```
cd /home
```

切换到/home 文件夹路径下

1.1.4 命令： **clear**

功能：清屏，只保留当前提示符

1.1.5 命令： **mkdir []**

功能：新建一个目录

例如：

```
mkdir project
```

在当前目录下创建一个名为 **project** 的文件夹

1.1.6 命令： **touch [文件名]**

功能：创建新文件

例如：

```
touch file1
```

创建一个名为“file1”的空白文件

1.1.7 命令： **rm [选项][文件或文件夹]**

功能：删除指定文件或文件夹

选项：

主要参数：

- i 删除前逐一询问确认；

- f 即使原档案属性设为唯读，亦直接删除，无需逐一确认；

- r 将目录及以下之档案亦逐一删除例如：

例如：

```
rm -r project
```

将 **project** 目录及目录下的文件全都删除

1.1.8 命令： **tar [选项][文件目录]**

功能：对文件进行压缩和解压缩

选项：

- c 建立新的归档文件

- r 向归档文件末尾追加文件

- x 从归档文件中解出文件
- O 将文件解开到标准输出
- v 处理过程中输出相关信息
- f 对普通文件操作
- z 调用 `gzip` 来压缩归档文件，与-x 联用时调用 `gzip` 完成解压缩
- Z 调用 `compress` 来压缩归档文件，与-x 联用时调用 `compress` 完成解压缩

例如：

```
tar -vcf project.tar.gz project
```

对 `project` 文件夹执行压缩命令

```
tar -vxf project.tar.gz
```

将压缩文件夹 `project` 解压

1.1.9 命令：`cp` [选项][源文件或目录][目标文件或目录]

功能：拷贝指定文件

选项：

-a 该选项保留链接、文件属性，并递归地拷贝目录，其作用等于 `dpR` 选项的组合。

-d 拷贝时保留链接。

-f 删除已经存在的目标文件而不提示。

-i 与 `f` 命令相反，在覆盖目标文件之前将给出提示要求用户确认。回答 `y` 时目标文件将被覆盖，是交互式拷贝。

-p 此时 `cp` 除复制源文件的内容外，还将把其修改时间和访问权限也复制到新文件中。

-r 若给出的源文件是一目录文件，此时 `cp` 将递归复制该目录下所有的子目录和文件。

此时目标文件必须为一个目录名。

-l 不作拷贝，只是链接文件。

例如：

```
cp -r project beifen
```

将 `project` 文件夹及其子目录下所有文件拷贝到 `beifen` 文件夹中

1.1.10 命令： **find** [查找路径] -name [文件名]

功能：查看指定工作目录下的内容

选项：

-a 显示出所有文件及目录

-l 列出所有文件详细的参数

例如：

```
ls -l /home
```

显示/home 目录下的文件和目录的详细信息

1.1.11 命令： **man** [选项]

功能：查看命令的参考手册

例如：

```
man -cp
```

查看 cp 命令的说明

1.1.12 命令： **ifconfig**

功能：查看或设置网络设备属性

选项：

interface: 网络接口的名称，如 eth0（网卡）；

up: 激活网络设备；

down: 关闭网络设备；

add: IP 地址，即设置网络设备地址；

netmask add: 子网掩码。例如：

例如：ifconfig eth0 up

激活 eth0 网卡

1.1.13 命令： **apt-get** [选项][目录]

功能：软件包管理，进行软件的安装卸载

选项：

-update 重新获取软件包列表

-upgrade 进行更新

-install 安装新的软件包

`-remove` 移除软件包

例如：

`apt-get install libx264`

使用 `apt-get` 工具安装 `x264` 库

1.1.14 vi/vim 编辑器命令

功能：新建或编辑文件

例如：`vim test.c`

新建 `test.c` 文件并编辑

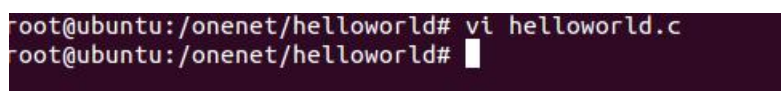
关于 `vi/vim` 的编辑操作请自行搜索网络和学习。

1.2 Linux 应用程序开发

在 `linux` 进行应用程序开发与 `windows` 下进行应用程序开发从原理上并无大的不同，都需要经过编辑、编译、链接几个过程，但由于 `windows` 集成开发环境较多也更加人性化，而且很多工作都由 `IDE` 完成，开发者只需要完成编码点击相应按钮即可，而在 `linux` 下由于 `IDE` 相对匮乏和不人性化，每个步骤的工作需要开发者自己完成，下面以 `HelloWorld` 开发过程为例进行大概说明，更加具体的请参考相关书籍。

编辑 `helloworld.c` 源文件

`vi helloworld.c`



```
root@ubuntu:/onenet/helloworld# vi helloworld.c
root@ubuntu:/onenet/helloworld#
```

输入并保存

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    printf("Hello World\n");
```

```
    return 0;
```

```
}
```



```
root@ubuntu: /onenet/helloworld
#include <stdio.h>
int main()
{
    printf("Hello World\n");
    return 0;
}
~
~
~
~
~
~
~
```

编译 helloworld.c 源文件

gcc helloworld.c -o helloworld

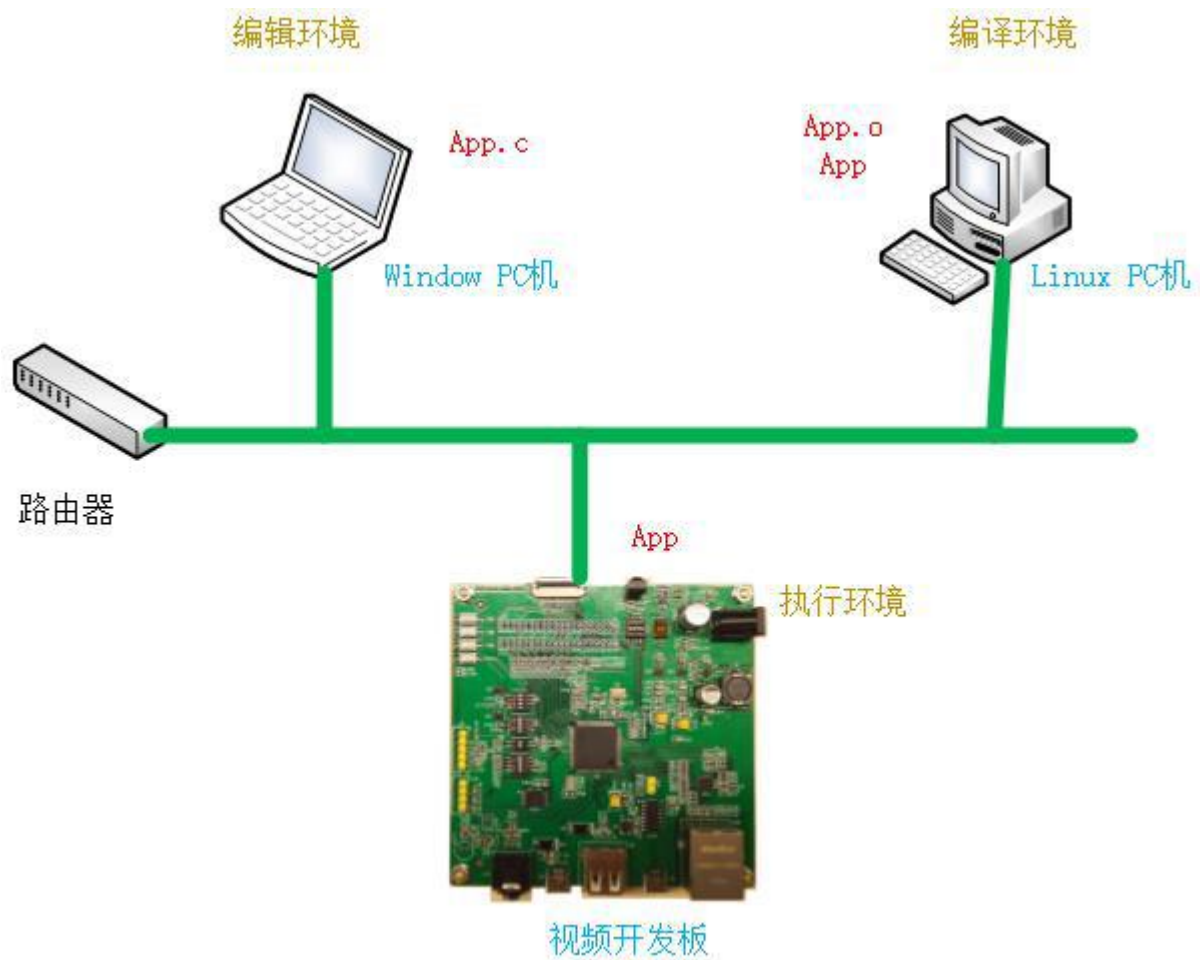
```
root@ubuntu: /onenet/helloworld
root@ubuntu:/onenet/helloworld# ls
helloworld.c
root@ubuntu:/onenet/helloworld# gcc helloworld.c -o helloworld
root@ubuntu:/onenet/helloworld# ls
helloworld helloworld.c
root@ubuntu:/onenet/helloworld#
```

执行 helloworld

```
root@ubuntu: /onenet/helloworld
root@ubuntu:/onenet/helloworld# ./helloworld
welcome to OneNET
please visit https://open.iot.10086.cn/root@ubuntu:/onenet/helloworld#
```

1.3 嵌入式 linux 应用开发介绍

1.3.1 嵌入式 linux 开发体系



嵌入式 linux 开发与 PC 环境下 linux 开发，区别主要由于 PC 与嵌入式硬件的硬件架构以及指令体系不一样导致不能和 PC 环境下直接编辑、编译、运行，在嵌入式环境下这三个部分是分开的，编辑在 windows PC 机下完成(主要是利用各种编辑器)，编译在 linux PC 机下完成，执行则在相应的硬件环境下进行。

如图中所示，在 windows 编辑 App.c 文件，在 Linux PC 下编译生成 App.o 以及 App，在开发板中执行 App。

1.3.2 搭建嵌入式 linux 开发环境

1) 编辑环境

编辑环境为 windows 的 PC 机，用户可选取任意一种文本编辑工具，比如 notepad++，source insight.

超级终端

由于开发板不带有 LCD 屏幕，主要通过串口，利用 PC 完成与开发板的交互。

a. 找到超级终端安装文件

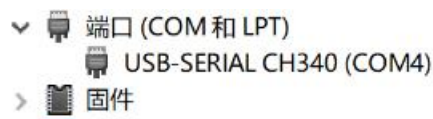
名称	修改日期	类型	大小
 hyper_terminal_latest	2014/8/24 18:24	应用程序	1,044 KB

b. 点击安装，安装过程并无特殊要求，按照流程安装即可。

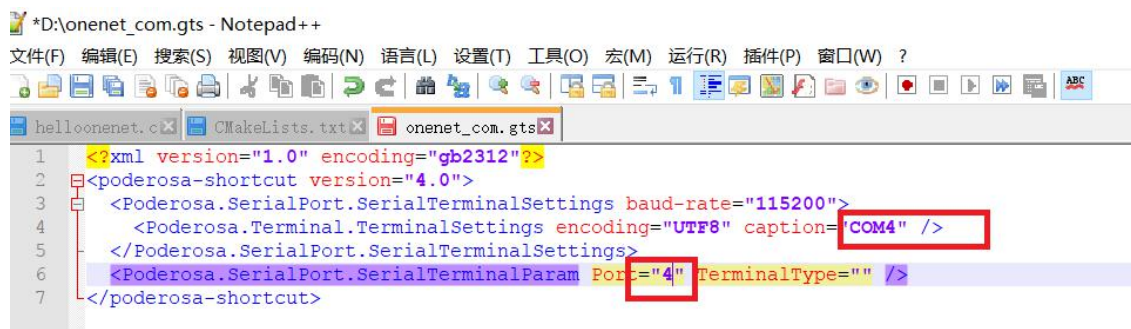


配置终端

由于选择很多笔记本或台式机已经不带有串口，因此开发板配置了 USB 转串口的转换器，将转换器插入电脑 USB 接口，通过设备管理器查看端口号。



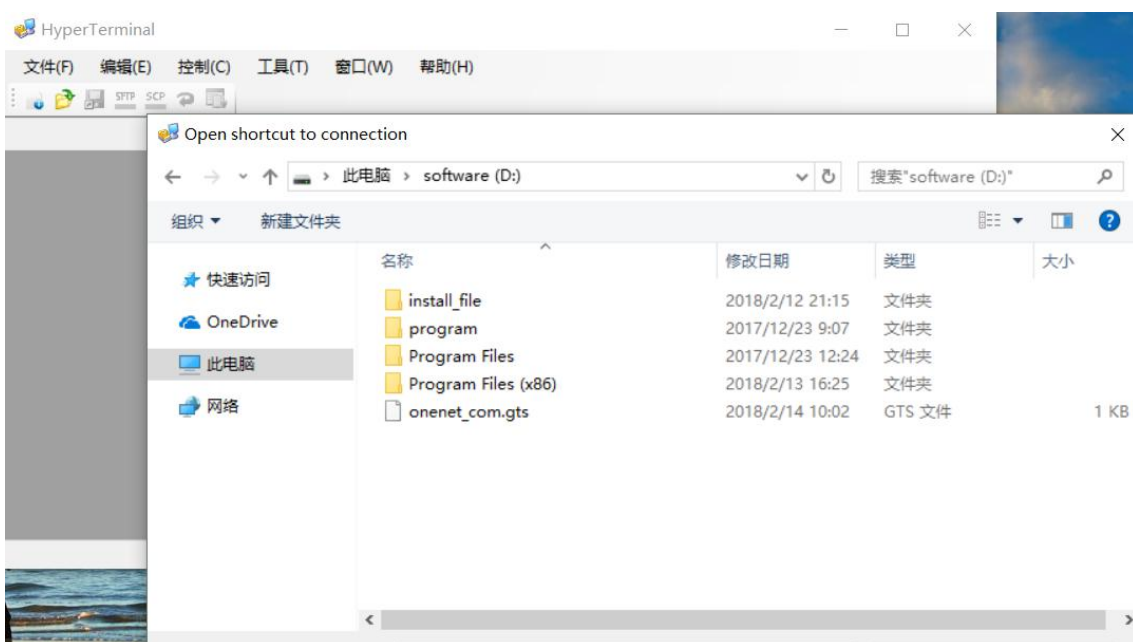
找到 onenet_com.gts 文件，打开 onenet_com.gts 文件，并将其中的端口号改为设备管理中的端口号，比如上面的 COM4。



打开超级终端，选择文件--打开配置文件



选择并打开刚才复制的 onenet_com.gts 文件，



启动开发板，插上串口杜邦线。



可通过超级终端输入 linux 命令操作开发板，实现与开发板的交互。

```

_rc, falc_enc, fmcpr_drv, vcap300_common, ft3dnr200, em, ms, Live 0x7f014000 (PO)
sht20 1874 0 - Live 0x7f008000 (O)
frammmap 20422 14 vpd_master, vpd_slave, loop_comm, audio_drv, fscaler300, sw_osg, falc_enc, fmcpr_d
rv, vcap300_common, fisp328, ft3dnr200, think2d, em, ms, Live 0x7f000000 (O)
=====
; GM8136_1M(Tiny Memory) Product Configuration (Without LCD, MPEG4, Scaler Substream, Encod
e Switching)
=====
Video Front End: ov9715
Chip Version: 81360001
RootFS Version:
MTD Version: 00_2015-01-09-GM8136_1MP
=====
/ #
/ #
/ #
/ #
/ # ls
bin          gm          mnt          squashfs_init var
boot.sh      init        proc         sys
dev          lib         sbin         tmp
etc          linuxrc     share        usr
/ # pwd
/
/ # cd mnt/nfs
/mnt/nfs # pwd
/mnt/nfs
/mnt/nfs # ifconfig eth0 192.168.200.148
HW reset

```

2) 编译环境

编译环境选取的是基于 VM 软件的 linux 虚拟机。请按照下面步骤，完成 VM、linux 虚拟机以及各种工具的安装。

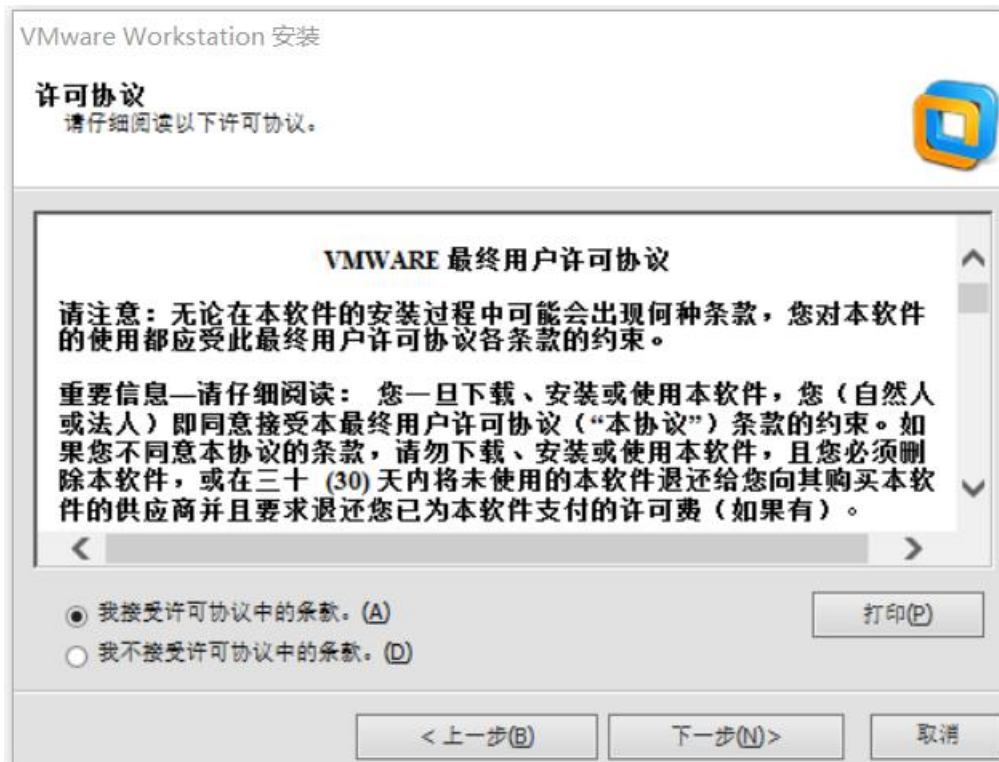
➤ VM 安装

- a. 找到虚拟机 10.0.1 安装文件，点击右键，选择以管理员身份运行。

VMware Workstation 安装



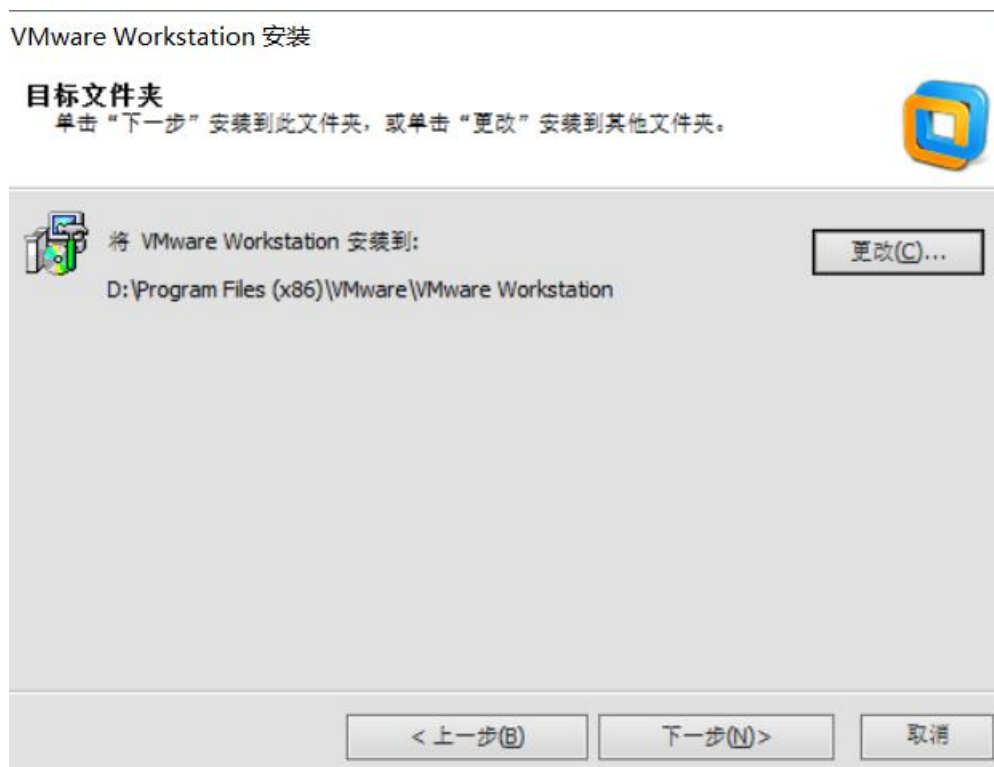
- b. 点击下一步，继续安装



c. 选择接受协议并点击下一步安装



d. 点击典型按钮继续安装



e. 自主选择安装目录并点击下一步



f. 取消掉启动时检查产品更新并点击下一步

VMware Workstation 安装

用户体验改进计划

您是否愿意向 VMware 发送反馈?



☐ 帮助改善 VMware Workstation(H)

向 VMware 发送匿名系统数据和使用情况统计信息。

[了解更多](#)

< 上一步(B)

下一步(N) >

取消

g. 取消掉帮助改善 VM 并点击下一步

VMware Workstation 安装

快捷方式

选择您要放入系统的快捷方式。



在以下位置创建 VMware Workstation 的快捷方式。

☒ 桌面(D)

☒ 开始菜单程序文件夹(S)

< 上一步(B)

下一步(N) >

取消

h. 可根据需求选择菜单方式，并点击下一步

VMware Workstation 安装

输入许可证密钥

(可选)您可以稍后再输入此信息。



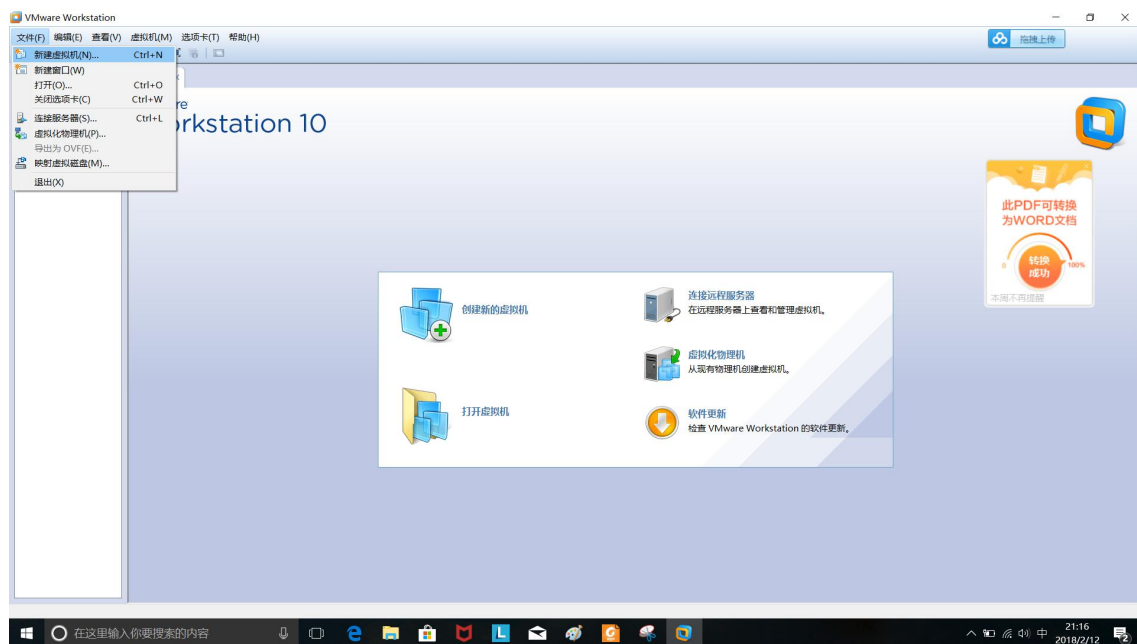
许可证密钥(L): (XXXXX-XXXXX-XXXXX-XXXXX-XXXXX)

输入(E) >

跳过(S) >

➤ Ubuntu 安装

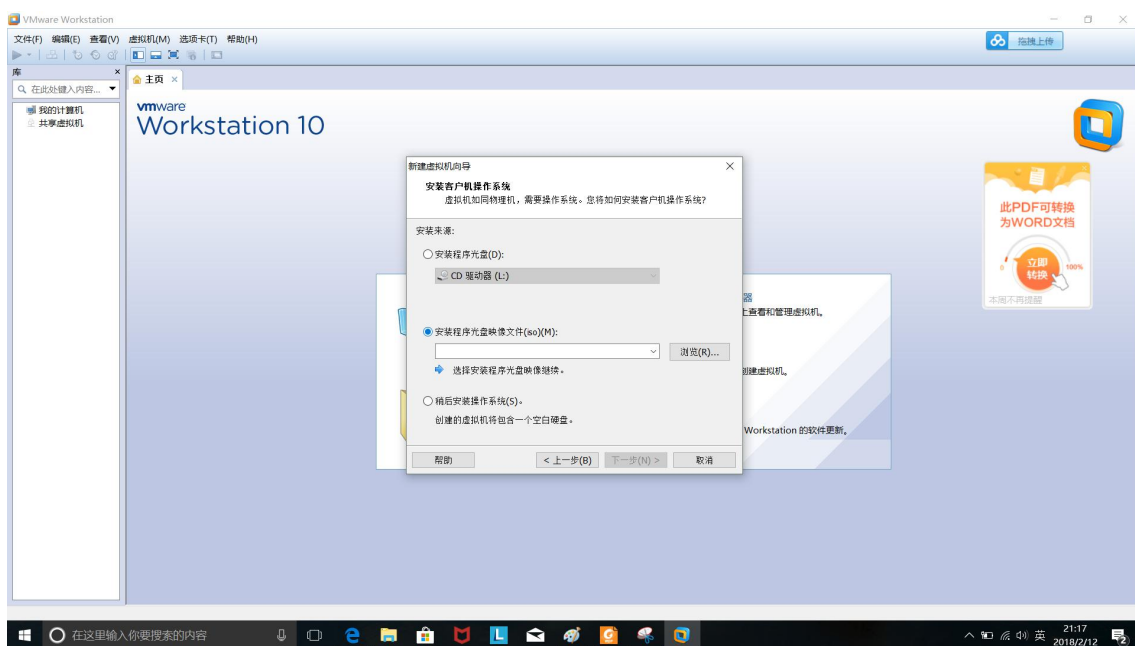
a. 选择新建虚拟机并点击



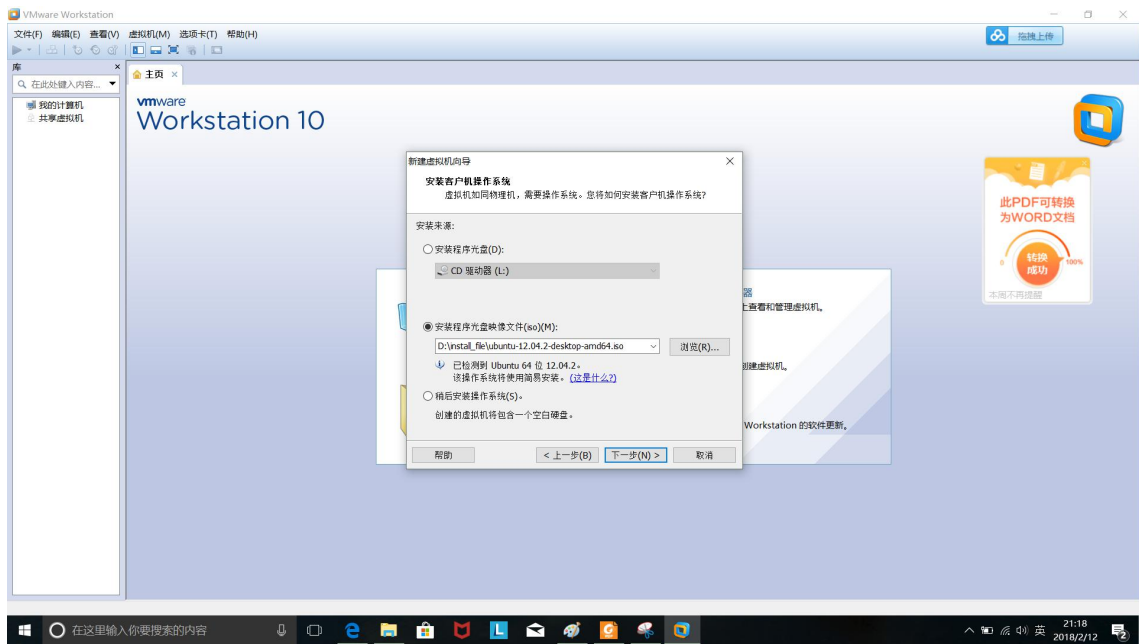
b. 选择典型并点击下一步



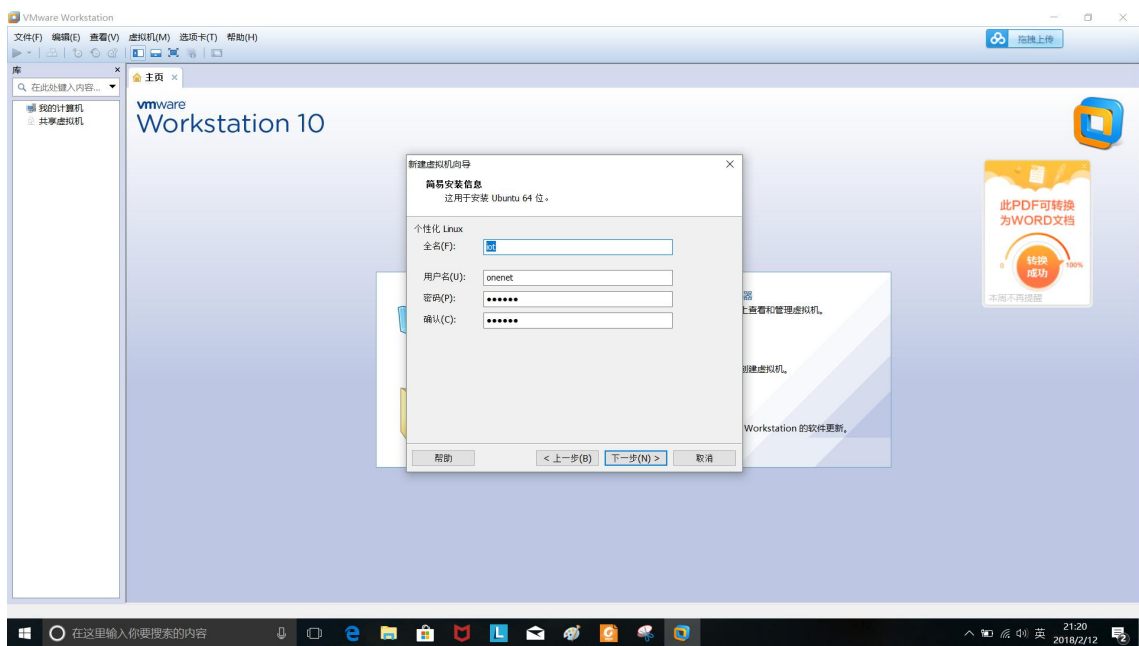
c. 选择安装程序光盘映像文件



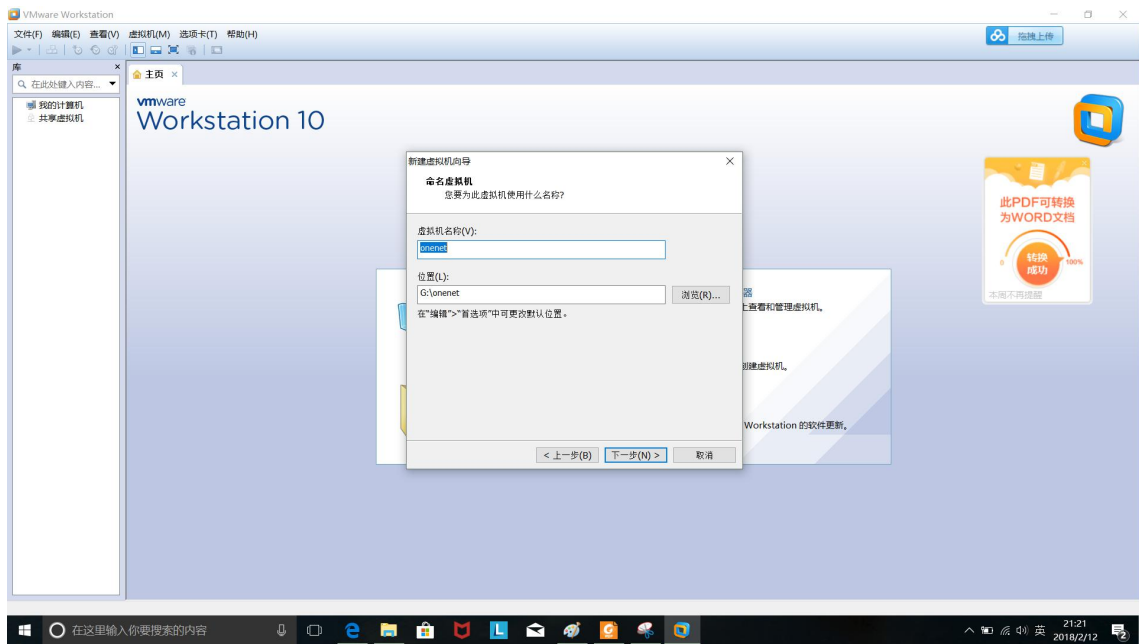
d. 从相应目录中选择镜像



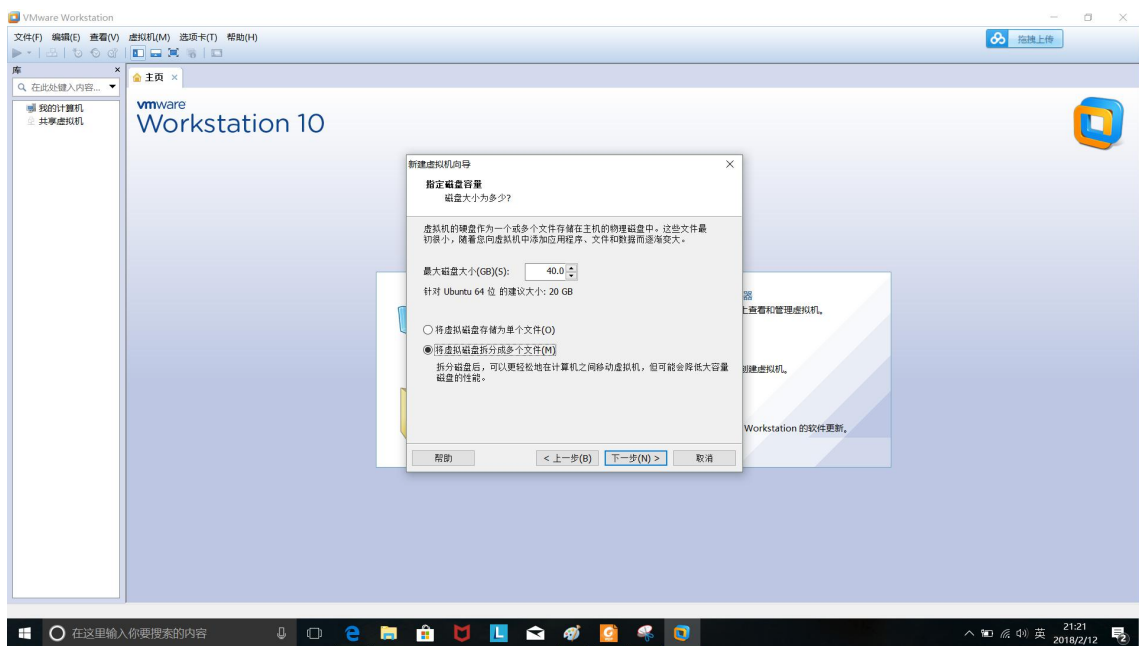
e. 根据需输入用户名和密码

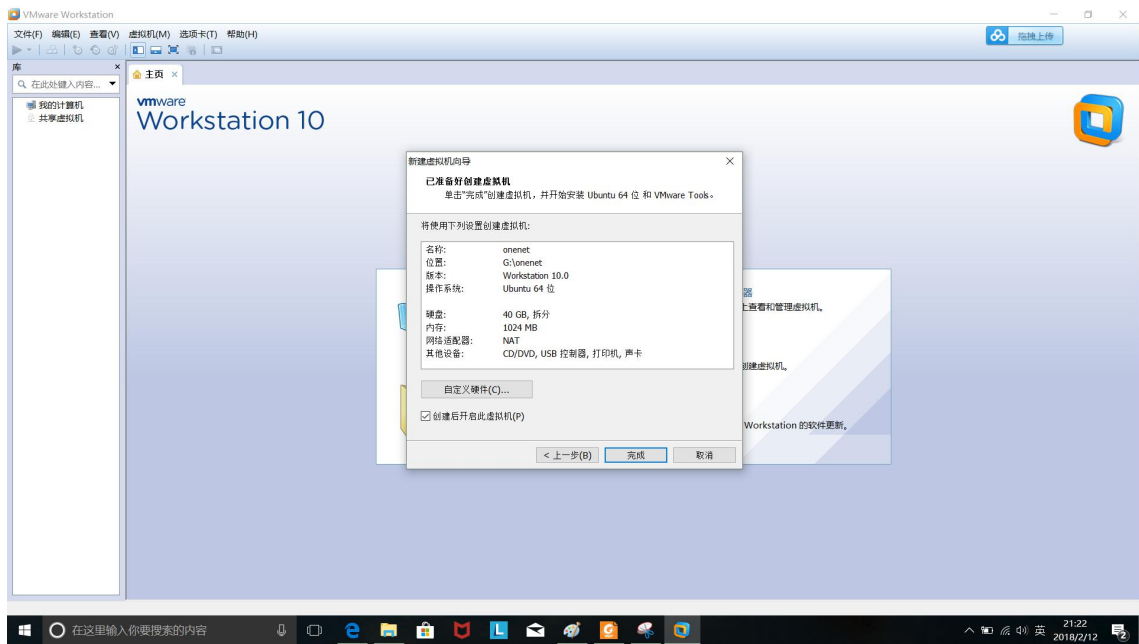


f. 给虚拟机填写相关名称和存储路径，并点击下一步

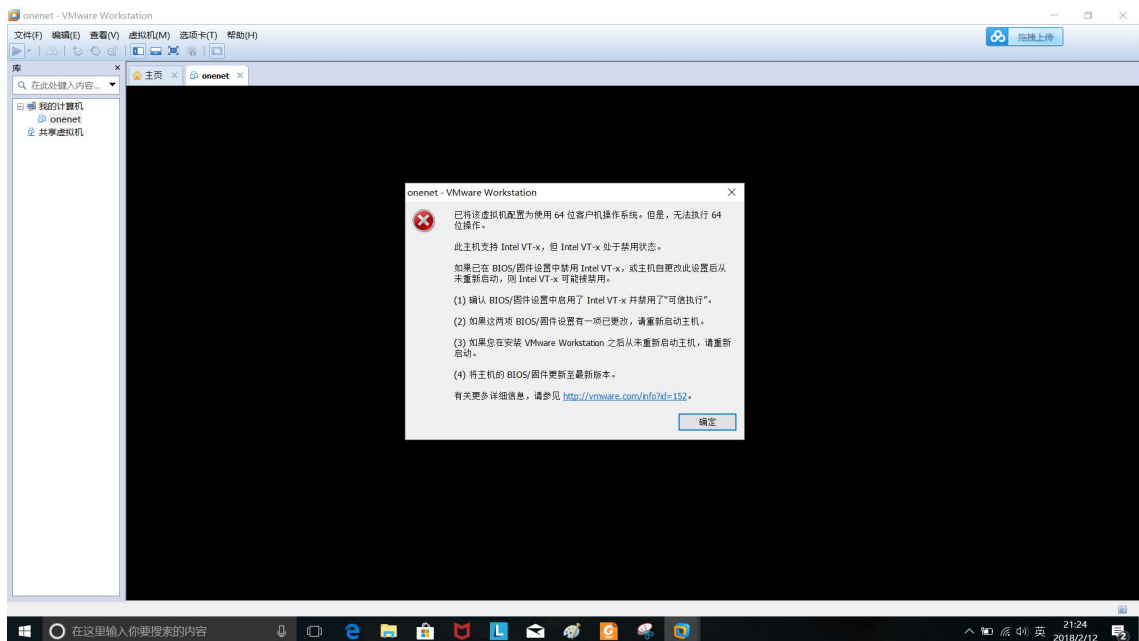


g. 分配磁盘空间 40G，选择分成多个文件并点击下一步

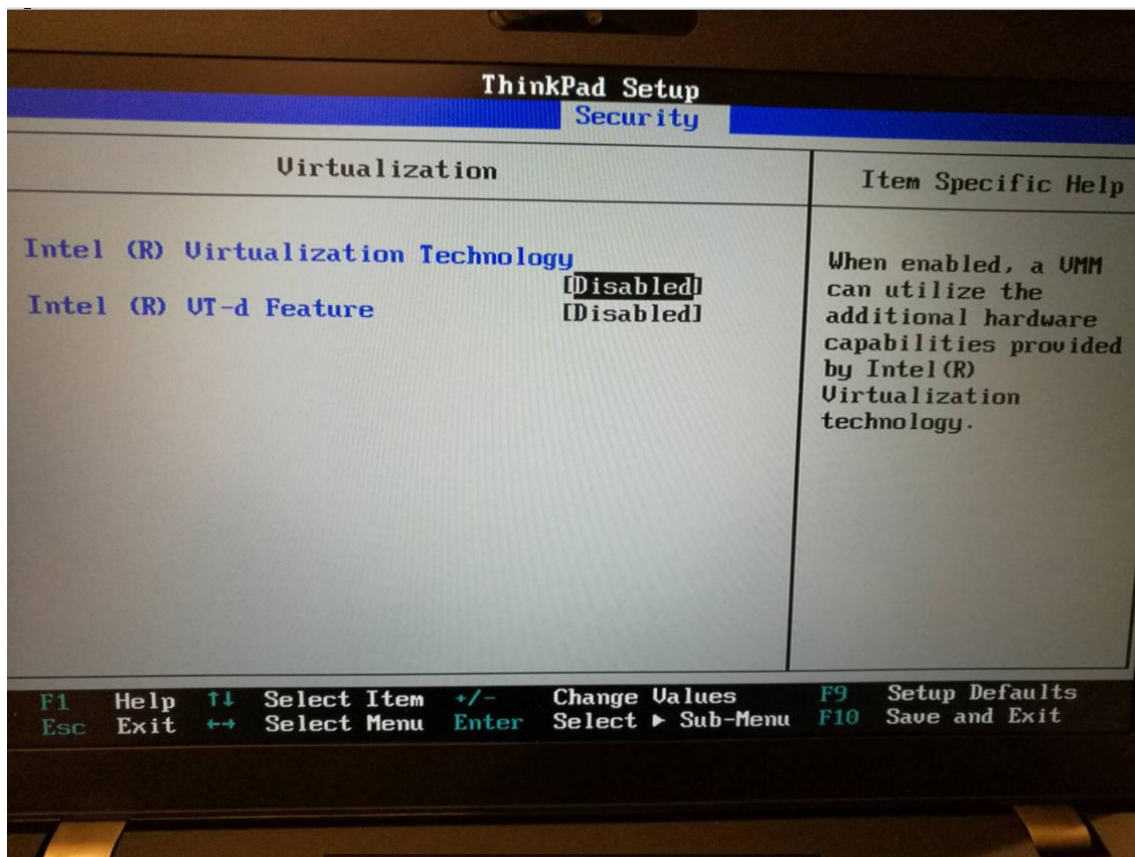




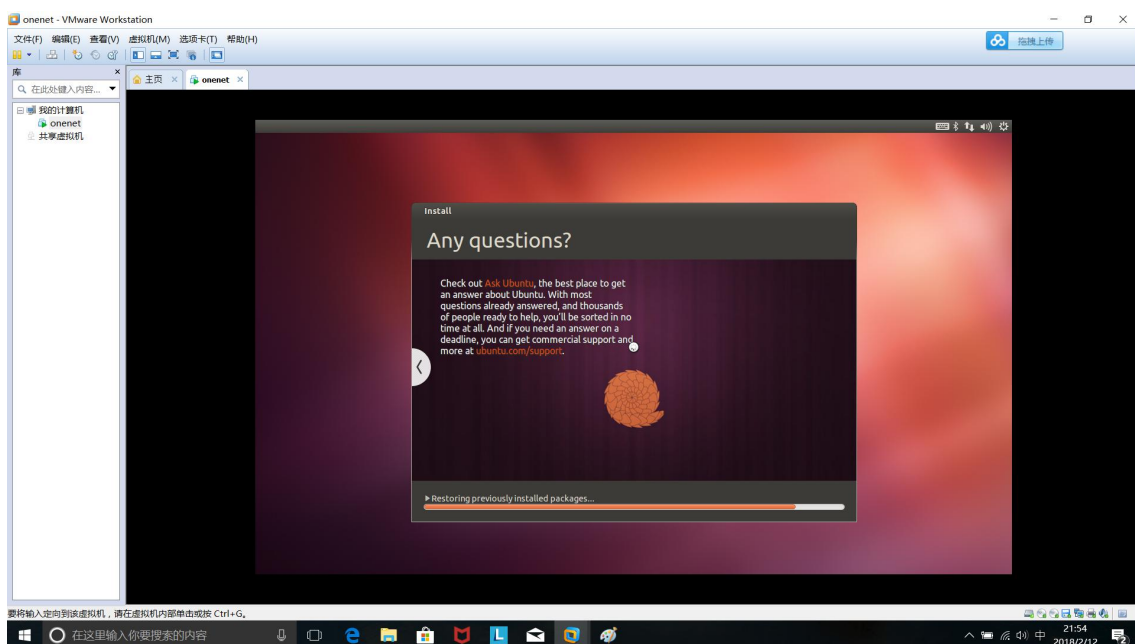
h. 点击启动虚拟机，虚拟机会自动启动并安装 linux 系统



i. 若启动虚拟机报错如上图，可进入真实 PC 机 BIOS 并根据图进行设置成 Enable。



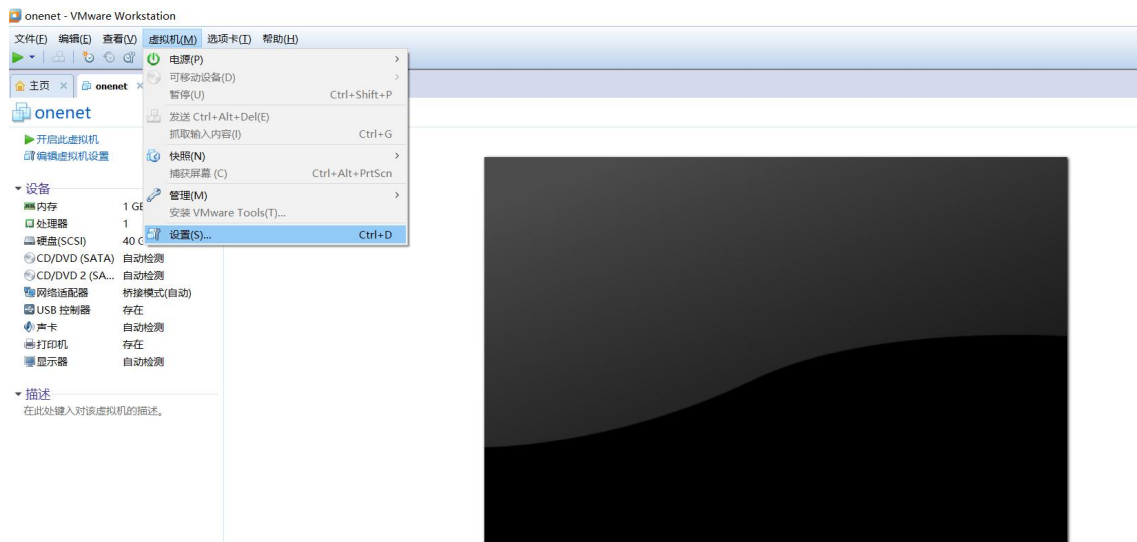
j. 等待 linux 系统安装完成



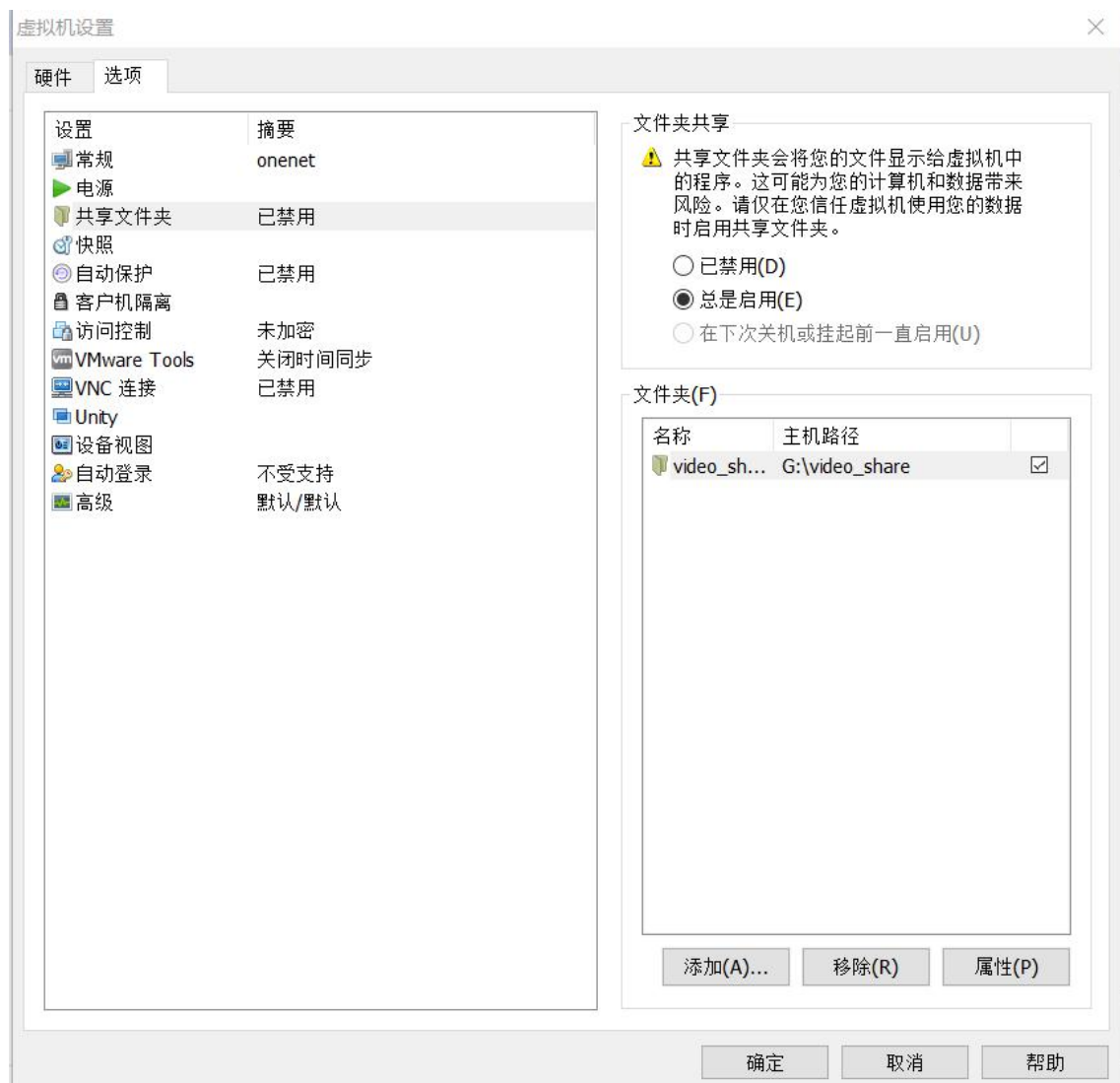
➤ 建立共享文件夹

建立主机(编辑环境)和虚拟机(编译环境)之间的共享文件夹，使得在 windows 下编辑的源文件可在虚拟机中直接使用。

a. 启动 VM 软件，选择虚拟机--设置菜单，进入



b. 进入虚拟机设置界面，并选择项目菜单，点击共享文件夹，进入共享文件夹设置。



c. 选择添加，添加共享路径。



d. 启动虚拟机，进入/mnt/hgfs/video_share 目录，用 vi test.c 命令新建一个文件并保存推出。

在 window 查看是否存在该文件

```
root@ubuntu:/# cd mnt
root@ubuntu:/mnt# ls
hgfs
root@ubuntu:/mnt# cd hgfs/
root@ubuntu:/mnt/hgfs# ls
video_share
root@ubuntu:/mnt/hgfs# cd video_share/
root@ubuntu:/mnt/hgfs/video_share# ls
root@ubuntu:/mnt/hgfs/video_share# vi test.c
root@ubuntu:/mnt/hgfs/video_share# ls
test.c
root@ubuntu:/mnt/hgfs/video_share#
```



➤ ssh 软件安装

ssh 软件主要用于主机(编辑环境)和虚拟机(编译环境)之间的通信，包括远程控制和文件传输。请安装如下步骤在虚拟机测安装 ssh 服务端。

- sudo apt-get install openssh-server
- 检查 ssh 服务开启状态

```
ps -s | grep ssh
```

c. 通过以下命令启动 ssh 服务

```
service ssh start
```

```
/etc/init.d/ssh start
```

➤ nfs 安装

nfs 软件主要用于虚拟机(编辑环境)与开发板(执行环境)之间的文件共享和传输。

a. `sudo apt-get install nfs-kernel-server` 安装 nfs。

```
onenet@ubuntu:/$ sudo apt-get install nfs-kernel-server
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
  thunderbird-globalmenu
Use 'apt-get autoremove' to remove them.
The following extra packages will be installed:
  libgssglue1 libnfsidmap2 libtirpc1 nfs-common rpcbind
The following NEW packages will be installed:
  libgssglue1 libnfsidmap2 libtirpc1 nfs-common nfs-kernel-server rpcbind
0 upgraded, 6 newly installed, 0 to remove and 537 not upgraded.
Need to get 548 kB of archives.
After this operation, 1,923 kB of additional disk space will be used.
Do you want to continue [Y/n]? y
Get:1 http://us.archive.ubuntu.com/ubuntu/ precise-updates/main libgssglue1
4 0.3-4ubuntu0.1 [22.5 kB]
Get:2 http://us.archive.ubuntu.com/ubuntu/ precise/main libtirpc1 amd64 0.2
[84.2 kB]
Get:3 http://us.archive.ubuntu.com/ubuntu/ precise-updates/main rpcbind amd
```

b. `sudo mkdir /home/share/nfs` 建立 nfs 共享文件夹。

```
onenet@ubuntu:/$ sudo mkdir /home/nfs
```

c. `sudo vi /etc/exports` 配置 nfs

```
onenet@ubuntu:/$ sudo vi /etc/exports
```

d. 在文档的最后一行加入 `/home/song/nfs *(rw,sync,no_root_squash,no_subtree_check)` , 保存退出。

```
onenet@ubuntu: /  
# /etc/exports: the access control list for filesystems which may be exported  
# to NFS clients. See exports(5).  
#  
# Example for NFSv2 and NFSv3:  
# /srv/homes hostname1(rw, sync, no_subtree_check) hostname2(ro, sync, no_subtree_check)  
#  
# Example for NFSv4:  
# /srv/nfs4 gss/krb5i(rw, sync, fsid=0, crossmnt, no_subtree_check)  
# /srv/nfs4/homes gss/krb5i(rw, sync, no_subtree_check)  
/home/nfs *(rw, sync, no_root_squash, no_subtree_check)  
~  
~  
~  
~  
~  
~
```

- e. `sudo /etc/init.d/rpcbind restart` 重启 rpcbind
- f. `sudo /etc/init.d/nfs-kernel-server restart` 重启 nfs

```
onenet@ubuntu:/$ sudo /etc/init.d/nfs-kernel-server restart  
* Stopping NFS kernel daemon [ OK ]  
* Unexporting directories for NFS kernel daemon... [ OK ]  
* Exporting directories for NFS kernel daemon... [ OK ]  
* Starting NFS kernel daemon [ OK ]
```

开发板挂载

- g. 在开发板输入命令，ip 地址改成自己虚拟机的 IP 地址

```
mount -t nfs -o nolock 192.168.200.3:/home/nfs /mnt/nfs  
/ # ifconfig eth0 192.168.200.148  
/ # mount -t nfs -o nolock 192.168.200.3:/home/nfs /mnt/nfs
```

➤ 交叉编译工具安装

交叉编译工具是嵌入式环境下的 gcc，是 gcc 在 arm 版本，主要完成程序的编辑工作。

- a. `mkdir arm-linux-3.3`

- b. 复制 `uclibc_gnueabi-4.4.0_ARMv5TE.tgz` 到 `/usr/src/arm-linux-3.3`
`cp uclibc_gnueabi-4.4.0_ARMv5TE.tgz /usr/src/arm-linux-3.3`

```
onenet@ubuntu:/usr/src/arm-linux-3.3$ ls  
toolchain_gnueabi-4.4.0_ARMv5TE toolchain_gnueabi-4.4.0_ARMv5TE.tgz
```

- c. 解压工具链到 `/usr/src/arm-linux-3.3`

```
cd /usr/src/arm-linux-3.3
```

```
sudo tar xvfz uclibc_gnueabi-4.4.0_ARMv5TE.tgz
```

```

arm-linux-strip
arm-unknown-linux-uclibcgnueabi-addr2line
arm-unknown-linux-uclibcgnueabi-ar
arm-unknown-linux-uclibcgnueabi-as
arm-unknown-linux-uclibcgnueabi-c++
arm-unknown-linux-uclibcgnueabi-cc
arm-unknown-linux-uclibcgnueabi-c++filt
arm-unknown-linux-uclibcgnueabi-cpp
arm-unknown-linux-uclibcgnueabi-elfedit
arm-unknown-linux-uclibcgnueabi-g++
arm-unknown-linux-uclibcgnueabi-gcc
arm-unknown-linux-uclibcgnueabi-gcc-4.4.0
arm-unknown-linux-uclibcgnueabi-gccbug
arm-unknown-linux-uclibcgnueabi-gcov
arm-unknown-linux-uclibcgnueabi-gprof
arm-unknown-linux-uclibcgnueabi-ld
arm-unknown-linux-uclibcgnueabi-ld.bfd
arm-unknown-linux-uclibcgnueabi-ldconfig
arm-unknown-linux-uclibcgnueabi-ldd
arm-unknown-linux-uclibcgnueabi-nm
arm-unknown-linux-uclibcgnueabi-objcopy
arm-unknown-linux-uclibcgnueabi-objdump
arm-unknown-linux-uclibcgnueabi-ranlib
arm-unknown-linux-uclibcgnueabi-readelf

```

d. 设置默认工具链

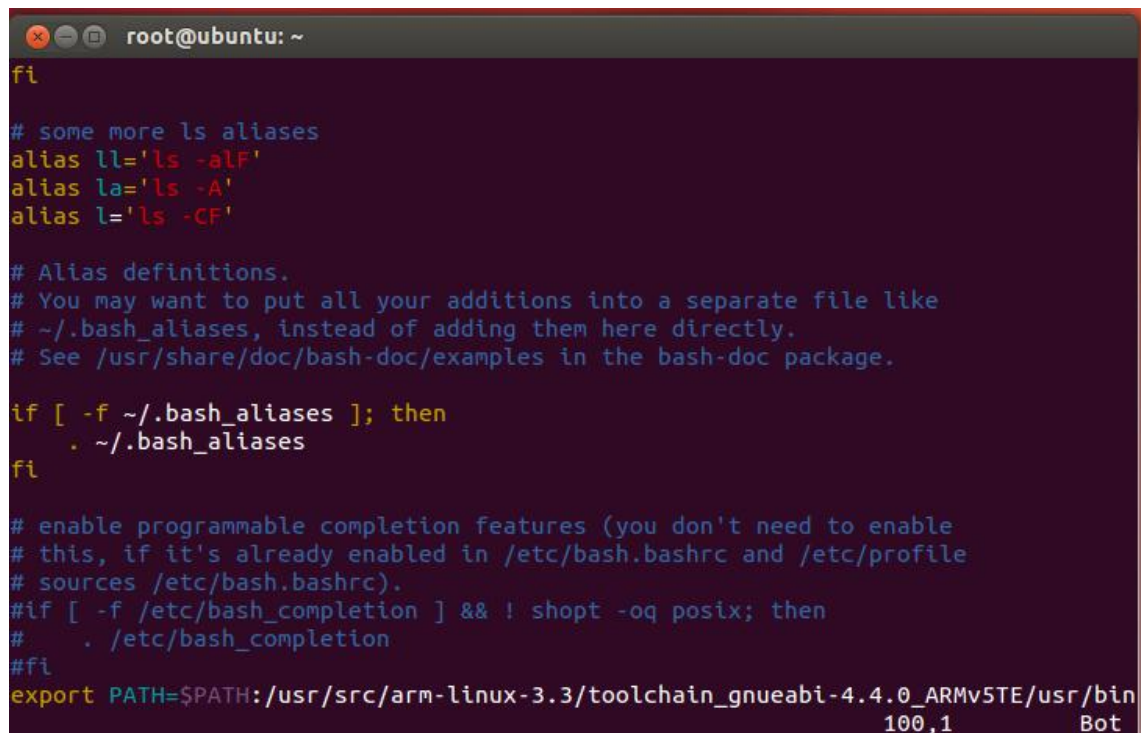
切换 root su root

cd /root

Vim .bashrc

在.bashrc 文件中的最后一行添加如下信息：

```
export PATH=$PATH:/usr/src/arm-linux-3.3/toolchain_gnueabi-4.4.0_ARMv5TE/usr/bin
```



```

root@ubuntu: ~
fi

# some more ls aliases
alias ll='ls -alF'
alias la='ls -A'
alias l='ls -CF'

# Alias definitions.
# You may want to put all your additions into a separate file like
# ~/.bash_aliases, instead of adding them here directly.
# See /usr/share/doc/bash-doc/examples in the bash-doc package.

if [ -f ~/.bash_aliases ]; then
    . ~/.bash_aliases
fi

# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
#if [ -f /etc/bash_completion ] && ! shopt -oq posix; then
#    . /etc/bash_completion
#fi
export PATH=$PATH:/usr/src/arm-linux-3.3/toolchain_gnueabi-4.4.0_ARMv5TE/usr/bin
100,1 Bot

```

退出保存

执行 source .bashrc 命令更新环境变量

在命令行输入 arm，然后按 TAB 键，如果从，显示如下信息

```
root@ubuntu:~# vim .bashrc
root@ubuntu:~# arm
arm2hpd1
arm-linux-addr2line
arm-linux-ar
arm-linux-as
arm-linux-c++
arm-linux-cc
arm-linux-c++filt
arm-linux-cpp
arm-linux-elfedit
arm-linux-g++
arm-linux-gcc
arm-linux-gcc-4.4.0
arm-linux-gccbug
arm-linux-gcov
arm-linux-gprof
arm-linux-ld
arm-linux-ld.bfd
arm-linux-ldconfig
```

e. 兼容性问题

由于虚拟机为 64 位，安装工具链后依然提示找不到编译器，请按照以下命令进行执行即可解决。


Sudo apt-get install lsb-core

Sudo apt-get install ia32-libs

➤ Cmake 安装

Cmake 是一个建立与 makefile 体系之上的工具，其让开发者用更简洁和易读的方式撰写 cmake 文件并生成 makefile。

a. sudo apt-get install build-essential

b. wget  http://www.cmake.org/files/v3.4/cmake-3.4.1.tar.gz
tar xf cmake-3.4.1.tar.gz

```
onenet@ubuntu:/$ wget --no-check-certificate http://www.cmake.org/files/v3.2/cmake-3.2.2.tar.gz
```

c. cd cmake-3.4.1

./configure

make

sudo apt-get install checkinstall

sudo make install

```
onenet@ubuntu:/$ cmake -version
cmake version 3.2.2
```

```
CMake suite maintained and supported by Kitware (kitware.com/cmake).
```

➤ Openssl 安装

Openssl 在编译 sdk 样例程序时会使用，否则 sdk 可能报错。

请按照一下三个命令顺序安装

\$ sudo apt-get install apache2 ##安装 Apache

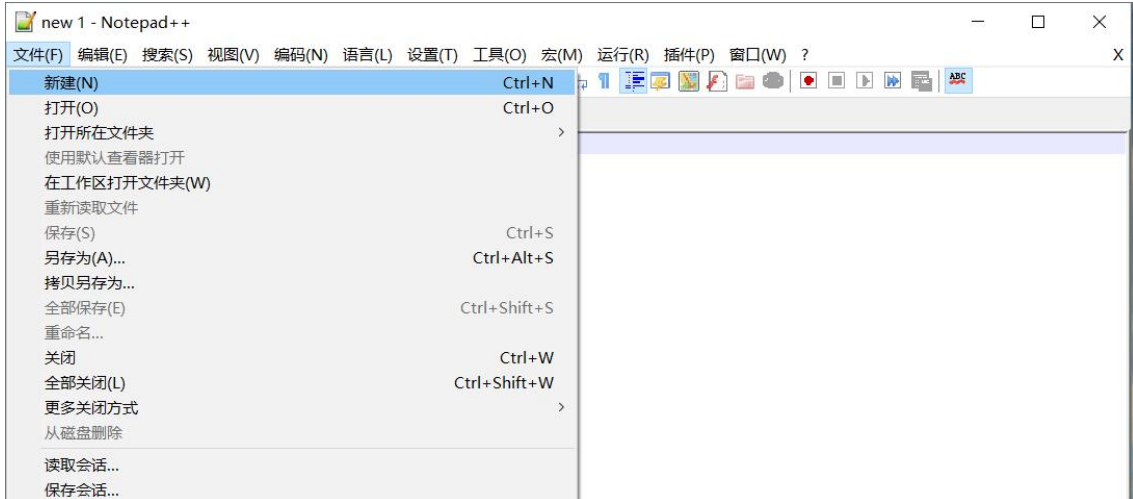
\$ sudo apt-get install openssl ##安装 openssl

\$ sudo apt-get install libssl-dev ##安装 openssl 开发库

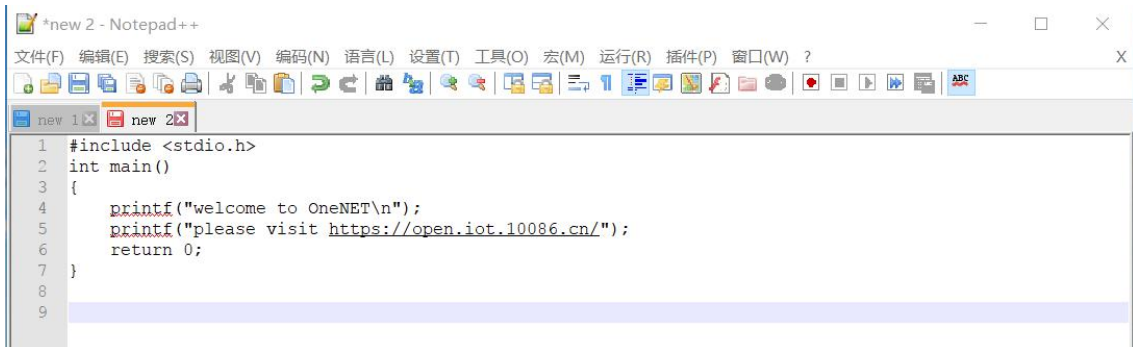
1.4 helloworld 开发

1.4.1 notepad++编辑

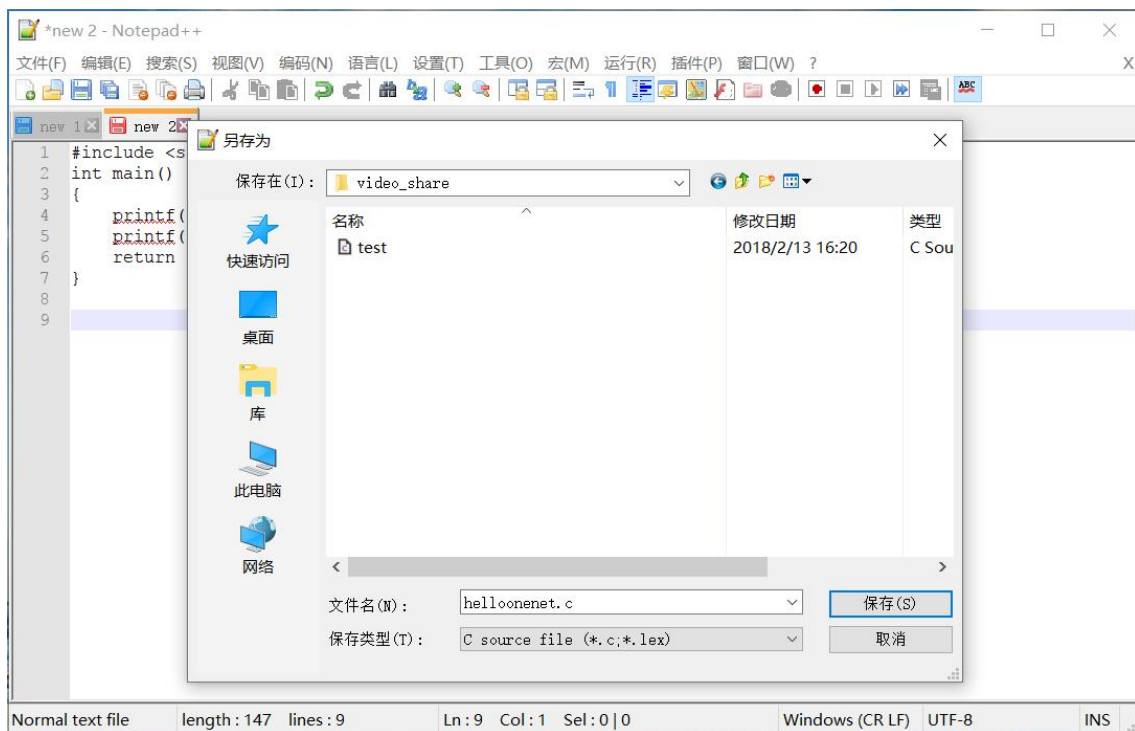
a. 启动 Notepad++, 选择文件-新建



b. 输入一下代码



c. 点击保存，填写文件名以及选择文件类型



d. 虚拟机环境进入共享目录

```

root@ubuntu: /mnt/hgfs/video_share
root@ubuntu:/onenet/helloworld# cd /mnt
root@ubuntu:/mnt# ls
hgfs
root@ubuntu:/mnt# cd hgfs/
root@ubuntu:/mnt/hgfs# ls
video_share
root@ubuntu:/mnt/hgfs# cd video_share/
root@ubuntu:/mnt/hgfs/video_share# ls
helloonenet.c test.c
root@ubuntu:/mnt/hgfs/video_share#

```

e. 可以发现 helloonenet 源文件的存在

1.4.2 宿主机交叉编译

将 helloonenet.c 文件从共享目录复制到 onenet 目录下，并输入如下命令编译。

arm-unknown-linux-uclibcgnueabi-gcc helloonenet.c -o helloonenet，即可产生可执行文件 helloonenet，但该文件在虚拟机下不能执行，需在开发板才能执行。

```

root@ubuntu:/mnt/hgfs/video_share# cp helloonenet.c /onenet/
root@ubuntu:/mnt/hgfs/video_share# cd /onenet/
root@ubuntu:/onenet# ls
helloonenet.c helloworld video_sdk-master
root@ubuntu:/onenet# arm-unknown-linux-uclibcgnueabi-gcc helloonenet.c -o helloonenet
root@ubuntu:/onenet# ls
helloonenet helloonenet.c helloworld video_sdk-master
root@ubuntu:/onenet# ./helloonenet
bash: ./helloonenet: cannot execute binary file
root@ubuntu:/onenet#

```

1.4.3 目标执行—调试接口

将产生的执行文件 helloonenet 复制到/home/nfs 目录下，该目录是虚拟机与开发板的共享目录（注意区别于开发环境 window 主机和编译环境 linux 虚拟机的共享目录）

```
root@ubuntu:/onenet# cp helloonenet /home/nfs/
root@ubuntu:/onenet# ls
helloonenet  helloonenet.c  helloworld  video_sdk-master
root@ubuntu:/onenet# ls /home/nfs/
encode_capture_substream  helloonenet
root@ubuntu:/onenet#
```

a. 通过超级终端进入开发板命令行

使用命令挂载 nfs 共享目录

```
mount -t nfs -o nolock 192.168.200.3:/home/nfs /mnt/nfs
```

b. 查看共享目录

```
/mnt/nfs # ls
encode_capture_substream  helloonenet
/mnt/nfs #
```

c. 执行 helloworld

```
/mnt/nfs # ./helloonenet
welcome to OneNET
please visit https://open.iot.10086.cn//mnt/nfs #
```

1.5 helloworld 开发 基于 makefile

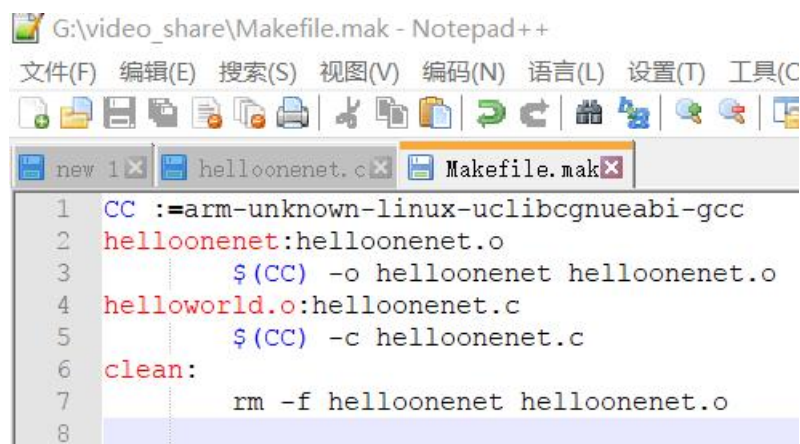
1.5.1 notepad++编辑

可参考附录 1.4.1 章节

1.5.2 宿主机交叉编译

附录 1.4.2 是直接通过 arm-unknown-linux-uclibcgnueabi-gcc helloonenet.c -o helloonenet 命令编译源文件，但当有多个源文件时，直接用命令就会非常繁琐，linux 采用 Makefile 文件来组织这些编译命令，将编译命令集成，并通过 make 一个命令统一执行。

a. 在 notepad++中新建文件，并输入如下代码，保存文件名为 Makefile，文件类型为 Makefile



```
G:\video_share\Makefile.mak - Notepad++
文件(F) 编辑(E) 搜索(S) 视图(V) 编码(N) 语言(L) 设置(T) 工具(C)
new 1 x helloonenet.c x Makefile.mak x
1 CC :=arm-unknown-linux-uclibcgnueabi-gcc
2 helloonenet:helloonenet.o
3 $(CC) -o helloonenet helloonenet.o
4 helloworld.o:helloonenet.c
5 $(CC) -c helloonenet.c
6 clean:
7 rm -f helloonenet helloonenet.o
8
```


将共享文件夹下的 Makefile 复制到 onenet 文件下，输入 Make 命令编译源文件

```
root@ubuntu: /onetet/hello_onenet_makefile
root@ubuntu: /onetet/hello_onenet_makefile# ls
helloonenet.c  Makefile
root@ubuntu: /onetet/hello_onenet_makefile# make
arm-unknown-linux-uclibcgnueabi-gcc -c -o helloonenet.o helloonenet.c
arm-unknown-linux-uclibcgnueabi-gcc -o helloonenet helloonenet.o
root@ubuntu: /onetet/hello_onenet_makefile# ls
helloonenet  helloonenet.c  helloonenet.o  Makefile
root@ubuntu: /onetet/hello_onenet_makefile#
```

详细的 Make 资料请自行百度学习或参考相关教程。

1.5.3 目标执行—调试接口

可参考附录 1.4.3 章节

1.6 helloworld 开发 基于 cmake

1.6.1 notepad++编辑

可参考附录 1.4.1 章节

1.6.2 宿主机交叉编译

CMake 是一个比 make 更高级的编译配置工具，它可以根据不同平台、不同的编译器，生成相应的 Makefile 或者 vcproj 项目。通过编写 CMakeLists.txt，可以控制生成的 Makefile，从而控制编译过程。CMake 自动生成的 Makefile 不仅可以通过 make 命令构建项目生成目标文件，还支持安装（make install）、测试安装的程序是否能正确执行（make test，或者 ctest）、生成当前平台的安装包（make package）、生成源码包（make package_source）、产生 Dashboard 显示数据并上传等高级功能，只要在 CMakeLists.txt 中简单配置，就可以完成很多复杂的功能，包括写测试用例。如果有嵌套目录，子目录下可以有自己的 CMakeLists.txt。总之很强大！

b. 在 notepad++中新建文件，并输入如下代码，保存文件名为 CMakeList.txt，文件类型为 txt

```
1 cmake_minimum_required (VERSION 2.8)
2 project (HELLOONENET)
3 SET(TOOLCHAIN_DIR "/usr/src/arm-linux-3.3/toolchain_gnueabi-4.4.0_ARMv5TE/usr")
4 SET(CMAKE_FIND_ROOT_PATH "${TOOLCHAIN_DIR}")
5 SET(CMAKE_C_COMPILER "${TOOLCHAIN_DIR}/bin/arm-unknown-linux-uclibcgnueabi-gcc")
6 SET(CMAKE_CXX_COMPILER "${TOOLCHAIN_DIR}/bin/arm-unknown-linux-uclibcgnueabi-g++")
7
8 set (HELLO_SRCS helloonenet.c)
9 add_executable (Hello_OneNET ${HELLO_SRCS})
10
```

c. 将共享文件夹下的 CMakeList.txt 复制到 onenet 文件下，新建 build 目录，进入 build 目录，输入../Cmake 命令产生 Makefile

```

root@ubuntu: /onenet/hello_onenet_cmake/build
root@ubuntu:/onenet/hello_onenet_cmake# mkdir build
root@ubuntu:/onenet/hello_onenet_cmake# ls
build CMakeLists.txt helloonenet.c
root@ubuntu:/onenet/hello_onenet_cmake# cd build
root@ubuntu:/onenet/hello_onenet_cmake/build# ls
root@ubuntu:/onenet/hello_onenet_cmake/build# cmake ../
-- The C compiler identification is GNU 4.6.3
-- The CXX compiler identification is GNU 4.6.3
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Configuring done
-- Generating done
-- Build files have been written to: /onenet/hello_onenet_cmake/build
root@ubuntu:/onenet/hello_onenet_cmake/build#

```

输入 make 命令执行编译源程序

```

root@ubuntu:/onenet/hello_onenet_cmake/build# make
Scanning dependencies of target Hello_OneNET
[100%] Building C object CMakeFiles/Hello_OneNET.dir/helloonenet.c.o
Linking C executable Hello_OneNET
[100%] Built target Hello_OneNET
root@ubuntu:/onenet/hello_onenet_cmake/build# ls
CMakeCache.txt CMakeFiles cmake_install.cmake Hello_OneNET Makefile
root@ubuntu:/onenet/hello_onenet_cmake/build#

```

1.6.3 目标执行—调试接口

可参考附录 1.4.3 章节