镜像说明

因为1.14寸屏幕和40P屏幕接口有冲突，固有两个镜像

img_SPI_nand_mpu6050_lcd666_wifi 使用大屏幕

img_SPI_nand_mpu6050_l.14_wifi 使用小屏幕

1.1 lcd666镜像使用

1.1.1 烧录镜像：

这里使用镜像img_SPI_nand_mpu6050_lcd666_wifi，烧录方法参考瑞芯微官方

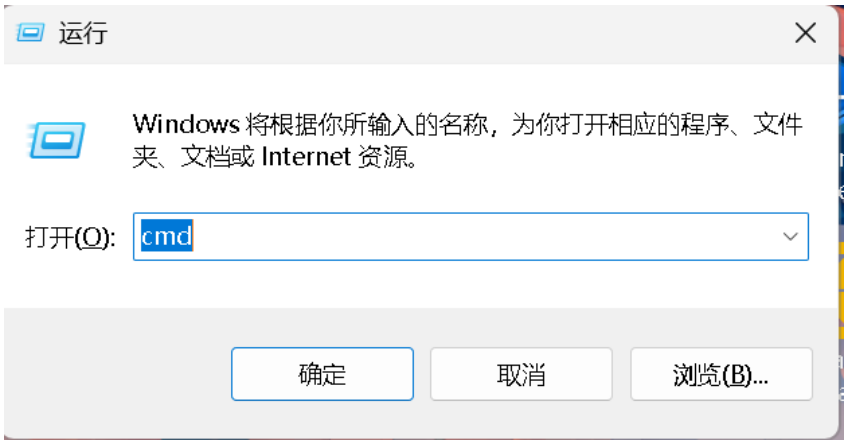
地址：SPI NAND Flash 镜像烧录 | LUCKFOX WIKI

1.1.2 文件传输：

提供所需驱动文件,如图：

名称	修改日期	类型	大小
rtl8723ds.ko	2025/3/19 19:43	KO 文件	1,495 KB
mpu6050_dts.ko	2025/3/19 12:42	KO 文件	165 KB
mpu6050Demo	2025/3/21 11:31	文件	8 KB
output_240x135.mp4	2025/3/16 14:11	媒体文件(.mp4)	27,232 KB
output_dong_480_272.mp4	2025/3/21 12:09	媒体文件(.mp4)	8,720 KB

使用adb push传输至小电脑的root文件夹，win+r键，输入cmd回车，如图：




```
C:\Users\lenovo>adb push "E:\RV1106\RV1106 镜像适配\立创镜像资料\img_SPI_nand_mpu6050_lcd666_wifi\驱动文件\output_dong_480_272.mp4" /root/output_dong_480_272.mp4
E:\RV1106\RV1106 é??å??é??é??\ç«?å??é??å??èμ?æ??\img_SPI_n...272.mp4: 1 file pushed. 4.4 MB/s (8928642 bytes in 1.943s)

C:\Users\lenovo>
```

每一个文件都需要传输，查看小电脑的root文件夹：

```
[root@luckfox root]# ls
mpu6050Demo              output_240_135.mp4       rtl8723ds.ko
mpu6050_dts.ko           output_dong_480_272.mp4
[root@luckfox root]#
```

1.1.3 使用屏幕:

ffmpeg -re -i output_dong_480_272 -vf "scale=480:272,fps=60" -pix_fmt bgra -f fbdev /dev/fb0

带音频输出命令

ffmpeg -re -i output_dong_480_272 -vf "scale=480:272,fps=60" -pix_fmt bgra -f fbdev /dev/fb0 -f wav - | aplay -D hw:0,0

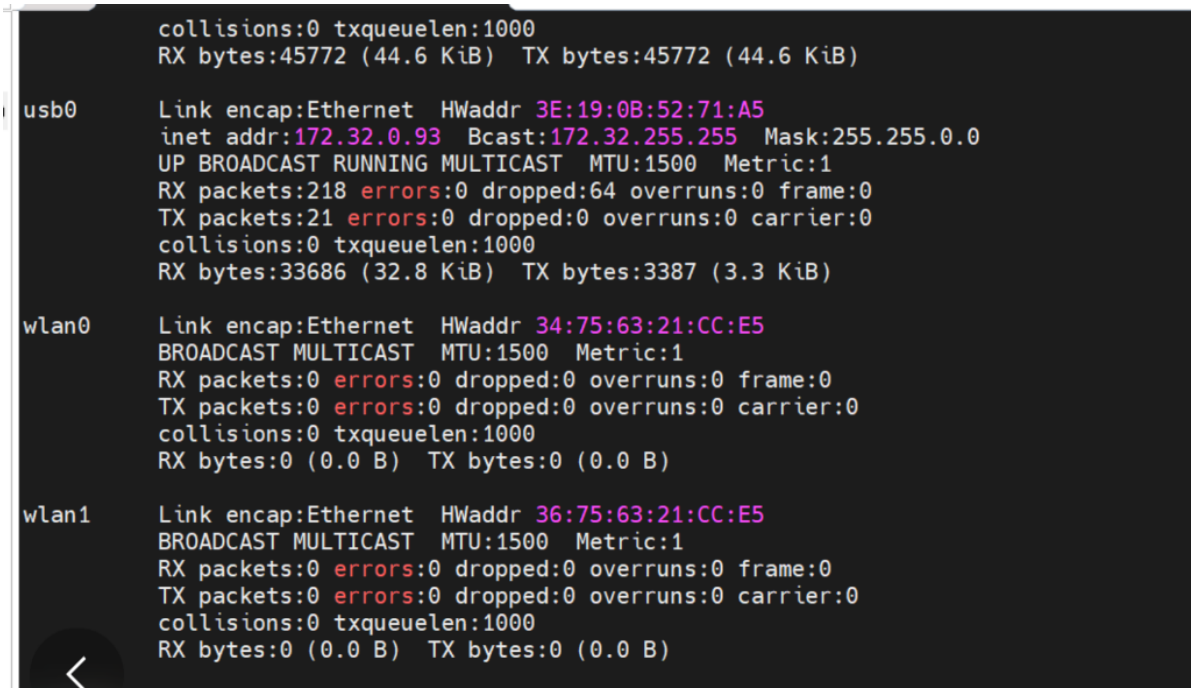
1.1.4 使用wifi:

关于测试wifi和陀螺仪的视频教程可以参考B站:

个人小电脑新添驱动*哔哩哔哩*bilibili

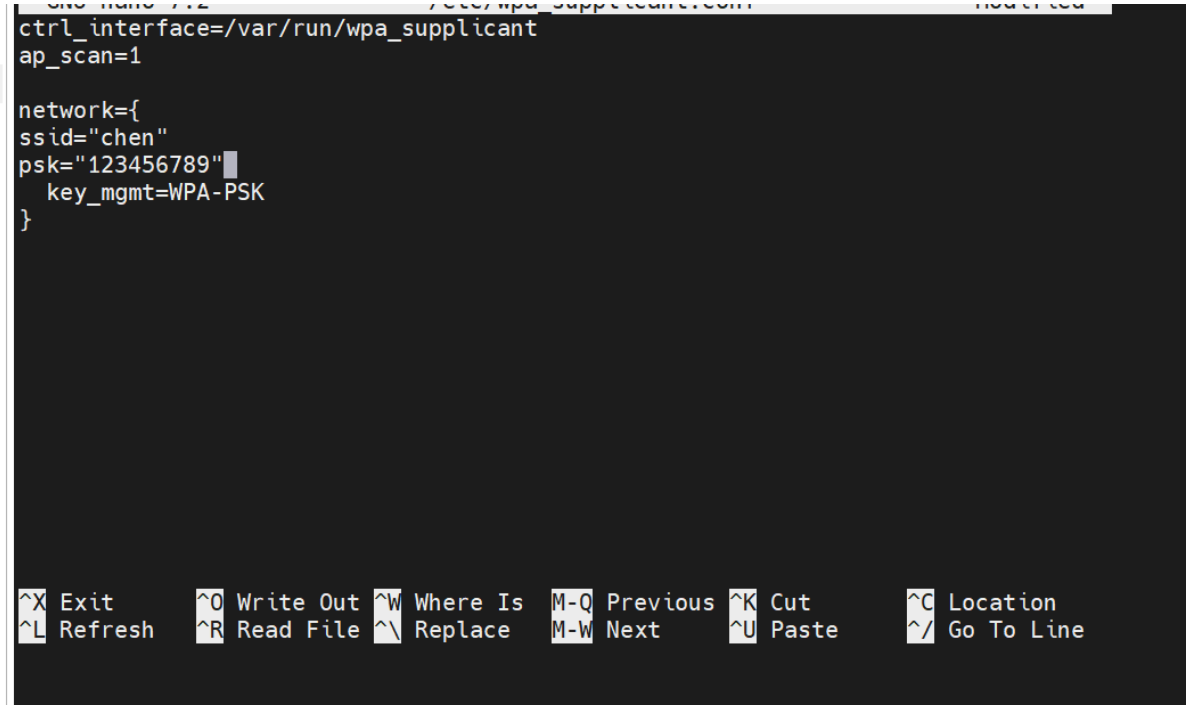
加载WiFi驱动，输入insmod rtl8723ds.ko

输入ifconfig，如图:



有wlan0，则代表，检测到了wifi

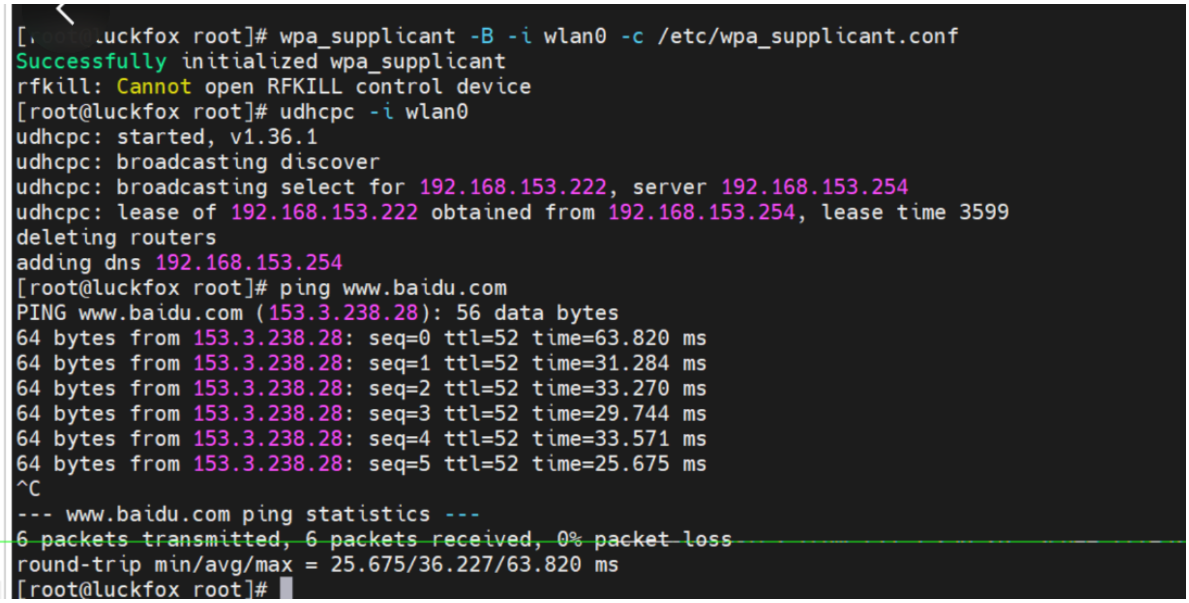
输入nano /etc/wpa_supplicant.conf，配置如下，ssid为wifi名称，psk不用我说了吧



连接wifi： wpa_supplicant -B -i wlan0 -c /etc/wpa_supplicant.conf

分配IP地址： udhcpc -i wlan0

ping www.baidu.com



1.1.5 驱动mpu6050：

insmod mpu6050_dts.ko

chmod +x mpu6050Demo

./mpu6050Demo

如图

```
[root@luckfox root]# insmod mpu6050_dts.ko
[  768.146631] mpu6050 ID = 0X68
[root@luckfox root]# chmod +x mpu6050Demo
[root@luckfox root]# ./mpu6050Demo

ADC value:
gx = 143, gy = -50, gz = -6
ax = -173, ay = 131, az = -2150
temp = -2772
TRUE value:act gx = 8.72°/S, act gy = -3.05°/S, act gz = -0.37°/S
act ax = -0.08g, act ay = 0.06g, act az = -1.05g
act temp = 16.44°C

ADC value:
gx = 138, gy = -55, gz = -8
ax = -178, ay = 132, az = -2138
temp = -2772
TRUE value:act gx = 8.41°/S, act gy = -3.35°/S, act gz = -0.49°/S
act ax = -0.09g, act ay = 0.06g, act az = -1.04g
act temp = 16.44°C
```

1.2 镜像使用：

除了屏幕不同外，其他同上：

- **使用屏幕：**

  ffmpeg -re -i output_240_135.mp4 -vf "scale=240:135" -pix_fmt rgb565 -r 30 -f fbdev /dev/fb0

  带音频：

  ffmpeg -re -i output_240_135.mp4 -vf "scale=240:135" -pix_fmt rgb565 -r 30 -f fbdev /dev/fb0 -f wav - | aplay -D hw:0,0

  使用 ffmpeg 转换图片格式：
  ffmpeg -i output.jpg -vf "scale=240:135" -pix_fmt rgb565 output.rgb

  dd命令写入： 64800是 135*240*2

  dd if=output.rgb of=/dev/fb0 bs=64800

  这样就可以显示图片啦！！

  设置音量为20%：

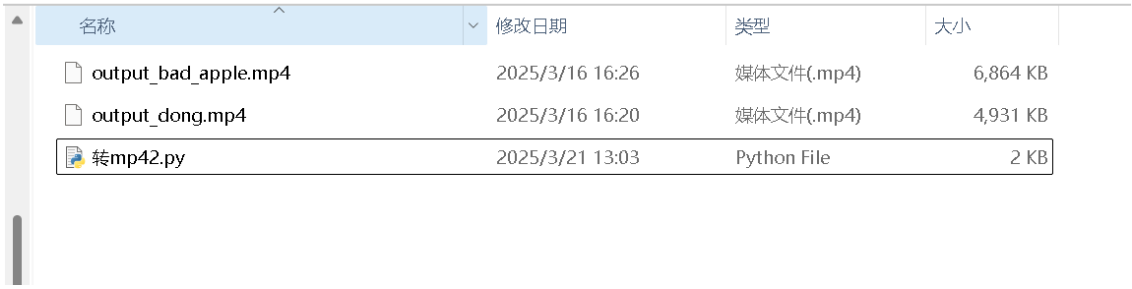
  amixer set 'DAC LINEOUT' 20%

- **其他：**

  应该没有要补充了的吧，嘿嘿！！

- **py脚本:**

  如果你有其他分辨率的屏幕，这里提供脚本文件在其他文件夹下：



名称	修改日期	类型	大小
output_bad_apple.mp4	2025/3/16 16:26	媒体文件(.mp4)	6,864 KB
output_dong.mp4	2025/3/16 16:20	媒体文件(.mp4)	4,931 KB
转mp42.py	2025/3/21 13:03	Python File	2 KB

```python
from moviepy.editor import VideoFileClip

# width=240, height=135
1 个用法
def resize_video(input_file, output_file, width=480, height=272):
    """
    将 MP4 文件转换为指定分辨率（默认 240x135）的 MP4 文件，并保留音频。

    参数:
        input_file (str): 输入视频文件路径。
        output_file (str): 输出视频文件路径。
        width (int): 目标宽度（默认 240）。
        height (int): 目标高度（默认 135）。
    """
    try:
        # 加载视频（包含音频）
        clip = VideoFileClip(input_file)
        # 调整分辨率
        resized_clip = clip.resize((width, height))
        # 保存为新的 MP4 文件（保留音频）
        resized_clip.write_videofile(output_file, fps=clip.fps)
        print(f"视频已成功转换为 {width}x{height} 并保存为 {output_file}")
    except Exception as e:
        print(f"转换失败: {e}")

if __name__ == "__main__":
    # # 输入文件路径
    # input_file = r"G:\Bad_Apple_but_it's_in_4k_60fps.mp4"
    # 输入文件路径
    input_file = r"G:\up_dianhu.mp4"
    # 输出文件路径
    output_file = "up_dianhu_480_272.mp4"

    # 调用函数调整视频分辨率
    resize_video(input_file, output_file)
```

e_video()

```c
typedef struct {
    int fd;
    const char *dev_name;
} uart_info_t;

typedef struct {
    log_queue_t *queue;
    const char *dir_name;
    const uart_info_t *uart_list;
    int uart_count;
} log_thread_arg_t;
```

```
log_queue_t* queue_create() {
  log_queue_t *q = calloc(1, sizeof(log_queue_t));
  pthread_mutex_init(&q->lock, NULL);
  pthread_cond_init(&q->cond, NULL);
  return q;
}
```

怎么使用，注释写得很清除了

好了，编写的也有些累了，祝你复刻顺利哈！！！

```
###
```