开发板ip设置：

vi /etc/network/interfaces

注释eth0

添加：

iface eth0 inet static //静态IP

address 192.168.1.2 //开发板IP

netmask 255.255.255.0 //子网掩码

设置开机启动。vi /etc/init.d/rc

ifconfig eth0 192.168.1.2 netmask 255.255.255.0

route add default gw 192.168.1.1

重启开发板：/etc/init.d/networking restart

OPENcv问题:

1.usr/include/c++/7/cstdlib:75:15: fatal error: stdlib.h: 没有那个文件或目录opencv

解决：****ENABLE\_PRECOMPILED\_HEADERS=OFF****

2.OpenCV Error: Unspecified error (The function is not implemented. Rebuild the library with Windows, GTK+ 2.x or Carbon support. If you are on Ubuntu or Debian, install libgtk2.0-dev and pkg-config, then re-run cmake or configure script) in cvShowImage, file ==/home/lx/linux/tool/opencv/opencv-3.1.0/modules/highgui/src/window.cpp, line 545

解决：install libgtk2.0，重新make 安装

3.Warning: Ignoring XDG\_SESSION\_TYPE=wayland on Gnome.

解决：sudo nano /etc/gdm3/custom.conf

 找到#WaylandEnable=false，移除#号

mplayer项目笔记：

1.出现的问题：出现如下报错[Error]'for'loop initial declarations are only allowed i n C99 or C11 mode

原因：在使用一个旧版的C编译器，不支持C99或C11中的“「o循环初始化语法。

解决方法：把定义放到for循环外 .

2.出现的问题：Segmentation fault.

原因：读取数据发生紊乱，图片读取格式内存太大，采用图片显示的方法，导致系统崩溃

解决方法：使用gdb调试程序，定位到对应代码错误段。采用QT的开发方式可以减少图片读取的次数。

挂载问题：

---[ end Kernel panic - not syncing: VFS: Unable to mount root fs on unknown-block ]---

解决：mout挂载出错，重新配置挂载

**驱动问题**：

1、在调试各个模块时，模块占用了触摸屏的端口，导致触摸屏使用不了。问题定位：先查看系统中注册的所有输入设备相关的信息，发现没有触摸屏输入节点，然后去查看了设备树是不是模块占用了触摸屏的引脚，然后修改了还是发现，没有输入信息，然后排除不出结果，怀疑我在使用的时候不小心修改了某些文件，于是我重新烧录了系统，后面还是没有解决，最后我怀疑是触摸屏坏了，看了一下开发板才想起来，设备树是修改了引脚，但是开发板的线还是没有改变，然后把开发板线拔了，触摸屏恢复正常。

2、dht11\_drv: disagrees about version of symbol device\_create

要重新编译.ko模块，不能使用以前版本的内核编译过的

3、编译环境配置不正确的问题，导致编译失败。

解决方案：

设置环境变量：在编译前，设置ARCH和CROSS\_COMPILE环境变量，确保编译器和目标架构正确。

4、模块加载失败问题

确保编译的模块与当前运行的内核版本一致。

检查日志：使用dmesg命令查看内核日志，查找模块加载失败的具体原因，并根据日志信息进行修正。

**通信协议：**

SPI同步全双工，SCLK\MISO\MOSI\CS，当CS由1到0，表示开始传输数据请求，MSB 先行（高位先行）或 LSB（低位先行） ，一般是MSB，在SCLK上升沿触发，下降沿采集信号，一般8个或者16个单位，没有限制，结束标志位CS由0到1。

通常用于需要高速数据传输和实时性要求较高的应用、短距离通信

IIC半双工通信，起始信号为当SCL为高期间，SDA由高到低的跳变 ，结束信号为当SCL为高期间，SDA由低到高的跳变 ，当SCL为高电平时,便会获取SDA数据值 ，每传输8个数，则会发送一个ACK应答，当SDA为低电平，表示为有效应答信号ACK 。

start信号-7位地址-1位方向（0写，1读）-ack-数据

通常用于连接低速设备、连接多个从设备

帧起始（SOF）：1位显性位（0），标志帧开始。

仲裁字段：11位标识符（ID），决定消息优先级；1位远程传输请求（RTR）位。

控制字段：6位，包括4位数据长度代码（DLC），指示数据字段字节数。

数据字段：0-8字节数据。

CRC字段：15位循环冗余校验，用于错误检测。

ACK字段：2位，接收者确认帧接收（显性位表示成功）。

帧结束（EOF）：7位隐性位（1），标志帧结束。

帧间隙（IFS）：3位隐性位，确保帧间间隔。

1、当前结构体大小为0，当前int变量自身有效对齐数为4，满足条件①0%4=0，结构体大小0+4=4；

2、第二个char变量自身有效对齐数为1，当前结构体大小为4。满足条件①4%1=0，结构体大小4+1=5；

3、第三个short变量自身有效对齐数为2，当前结构体大小为5，不满足条件①5%2=0，故填充一个字节。现在6%2=0，满足条件，结构体大小6+2=8；

**驱动编写流程**：

dht11：创建主设备号、dht11类、gpio指针、设备操作函数结构体，主要是read函数编写，发送起始信号101，等待dht11回复010，读取一个8bit的数据，循环读5次，前四个数据加起来等于第5个说明，数据正确，通过copy\_to\_user发送前4个数据给用户层，然后是定义platform驱动结构体probe和remove函数，还有驱动名和设备树匹配表compatible，在probe函数中获取引脚信息和创建设备节点，remove函数删除设备节点，在入口函数注册设备、创建类、注册驱动结构体，出口函数则相反注销设备、删除类、注销驱动结构体。

sr04:创建主设备号、sr04类、gpio指针echo和trig，中断号，ns数，等待队列，设备操作函数结构体，主要是read函数编写，发送起始信号trig-10，等待数据，发送ns数的前4位数给用户，根据echo的值获取ns值，然后是定义platform驱动结构体probe和remove函数，还有驱动名和设备树匹配表compatible，在probe函数中获取引脚信息、获取中断号、设置echo双边延触发、创建设备节点，remove函数删除设备节点，释放中断号，删除引脚，在入口函数注册设备、创建类、注册驱动结构体，出口函数则相反注销设备、删除类、注销驱动结构体。

**驱动：**

linux和stm32串口通信接口

stm32：A9-tx,A10-rx,A0-sg90

hc-sr04: echo---HSYNC

TRIG---VSYNC

VCC---3.3V

sg90: 橙色线---GPIO1\_4

红色线---5.0V

dht11： VCC---5V

DATA---CSI\_DATA0(GPIO4\_21)

HS0038: VCC---5V(右）(凸面正向)

DATA---CSI\_MCLK (GPIO4\_17)（左）

直流电机： GPIO1\_9

**设备树：**

&ecspi3 {

pinctrl-names = "default";

pinctrl-0 = <&pinctrl\_ecspi3>;

cs-gpios = <&gpio5 9 GPIO\_ACTIVE\_LOW>;

status = "okay";

sg90: sg90@0 {

compatible = "alientek,sg90";

reg = <0>;

spi-max-frequency = <1000000>;

};

};

dht11 {

status = "okay";

compatible = "xf,dht11\_t";

gpios = <&gpio4 21 GPIO\_ACTIVE\_HIGH>;

};

hc\_sg90 {

compatible = "hc-sg90";

pwms = <&pwm3 0 20000000>;

status = "okay";

};

sr04 {

compatible = "my,sr04";

trig-gpios = <&gpio4 19 GPIO\_ACTIVE\_HIGH>;

echo-gpios = <&gpio4 20 GPIO\_ACTIVE\_HIGH>;

};

hs0038 {

status = "okay";

compatible = "xf,hs0038";

gpios = <&gpio4 17 GPIO\_ACTIVE\_HIGH>;

};

dc\_motor {

compatible = "hc-dc-motor";

pwms = <&pwm2 0 20000000>;

status = "okay";

};

/\*

pinctrl\_pwm3: pwm3grp {

fsl,pins = <

MX6UL\_PAD\_GPIO1\_IO04\_\_PWM3\_OUT 0x110b0

>;

};

\*/

/\*pinctrl\_pwm2: pwm2grp {

fsl,pins = <

MX6UL\_PAD\_GPIO1\_IO09\_\_PWM2\_OUT 0x110b0

>;

};\*/

/\*

&pwm2 {

pinctrl-names = "default";

pinctrl-0 = <&pinctrl\_pwm2>;

clocks = <&clks IMX6UL\_CLK\_PWM2>,

<&clks IMX6UL\_CLK\_PWM2>;

status = "okay";

};

\*/

/\*&pwm3 {

pinctrl-names = "default";

pinctrl-0 = <&pinctrl\_pwm3>;

clocks = <&clks IMX6UL\_CLK\_PWM3>,

<&clks IMX6UL\_CLK\_PWM3>;

status = "okay";

};

\*/