day01:

1、搭建开发环境

cd /home/tarena
mkdir tcar
cd tcar

1.1 编译内核 通过 tftp 方式加载启动内核

cp /mnt/hgfs/porting/kernel.tar.bz2 ./
tar xf kernel.tar.bz2
rm kernel.tar.bz2
cd kernel
//需要打补丁的自行打上补丁
...
cp arch/arm/configs/x6818_defconfig .config
make uImage

cp arch/arm/boot/uImage /tftpboot
//在开发板上配置通过 tftp 方式加载内核
setenv bootcmd ping 192.168.1.8 \; ping 192.168.1.8\; tftp 48000000 uImage \;bootm 48000000
saveenv

1.2 通过 nfs 方式挂载网络根文件系统

cp tcar_env/env/rootfs.tar.gz ./
sudo tar xf rootfs.tar.gz
rm rootfs.tar.gz
sudo vi /etc/exports 在文件的最后加入
/home/tarena/tcar/rootfs *(rw,sync,no_root_squash)
//重启 nfs server 使新配置生效
sudo /etc/init.d/nfs-kernel-server restart

在开发板上配置加载根文件系统的方式：

setenv bootargs root=/dev/nfs nfsroot=192.168.1.8:/home/tarena/tcar/rootfs init=/linuxrc console=ttySAC0 maxcpus=1 lcd=wy070ml tp=gslx680 ip=192.168.1.6:192.168.1.8:192.168.1.1:255.255.255.0 loglevel=2
saveenv

1.3 将交叉编译工具包中的所有共享库都部署到开发板

//找到交叉编译工具的路径
which arm-cortex_a9-linux-gnueabi-gcc

cd /home/tarena/tcar
mkdir rootfs/home/lib -p
cp /opt/arm-cortex_a9-eabi-4.7-eglibc-2.18/arm-cortex_a9-linux-gnueabi/sysroot/lib/*.so* rootfs/home/lib/ -a
sudo cp /opt/arm-cortex_a9-eabi-4.7-eglibc-2.18/arm-cortex_a9-linux-gnueabi/sysroot/usr/lib/*.so* rootfs/home/lib/ -a

vi rootfs/etc/profile
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home/lib/

1.4 执行完以上步骤的目录结构

目录结构：

tcar
|
|__ kernel: 内核源码
|__ rootfs: 开发板通过 nfs 方式挂载该根文件系统

2、关于 http 协议

2.1 协议的理解
参考 http.png
2.2 代码角度理解协议
cd /home/tarena/tcar
tar xf simple_web.tar.gz
rm simple_web.tar.gz
cd webserver/

gcc server_v1.c -o server
./server -p 8000

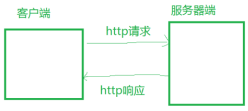
在 ubuntu 系统开启浏览器 输入 http://127.0.0.1:8000/

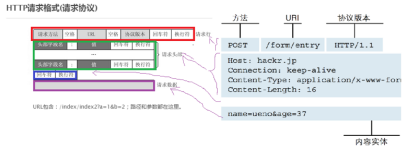
通过服务器输入信息理解 http 协议

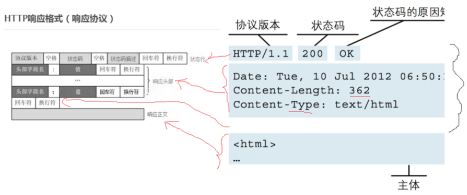
gcc server.c -o server
./server -p 8000
在 ubuntu 系统开启浏览器
输入 http://127.0.0.1:8000/test.html //文本显示原理
输入 http://127.0.0.1:8000/test1.html //html 文件的显示 包括了图片显示
输入 http://127.0.0.1:8000/test2.html //动态页面显示

http 通信：客户端给服务器端发请求
服务器端解析请求
根据请求将客户端需要的数据反馈给客户端的过程

html 的语法参考：https://www.runoob.com/




3、视频监控功能的实现

3.1 摄像头驱动

内核中自带了该摄像头的驱动 ，

该部分代码被称作 UVC 子系统

配置内核 将该部分驱动代码编译到 uImage

cd tcar/kernel

make menuconfig

Device Drivers --->

<*> Multimedia support --->

[*] Video capture adapters --->

[*] V4L USB devices --->

<*> USB Video Class (UVC)

make uImage

让开发板使用新内核

验证方式:

在串口中执行

ls /dev/video*

插入摄像头

ls /dev/video*

多出来的就是摄像头对应的设备文件

3.2 应用程序

3.2.1 在用户空间操作摄像头 读取图像数据

fd = open("/dev/video9", ...)

v4l2 的编程框架去操作摄像头 获取图像数据

v4l2: video for linux verison 2

在用户空间统一了摄像头的访问方式

如何按照 v4l2 编程框架去实现摄像头，官方给了固定套路

参考:

v4l_demo.zip

capture.c
Video for linux 2 example (v4l2 demo) - MetalSeed - 博客频道 - CSDN.NET.png

操作摄像头，获取图像数据可以自行编程实现
项目中是移植开源程序完成图像数据的获取

3.2.2 将采集到的图像数据分发给手机客户端
开源程序：mjpeg-streamer 可以完成
摄像头图像数据的采集
也能够把它封装成 http 响应的形式分发手机客户端

3.2.3 mjpeg-streamer 的移植：mjpg-streamer.tar.bz2
1）交叉编译 mjpeg-streamer
cd tcar
cp /mnt/hgfs/project/ehome/mjpg-streamer.tar.bz2 ./
tar xf mjpg-streamer.tar.bz2

rm mjpg-streamer.tar.bz2
cd mjpg-streamer/

/*开源程序的使用套路：
官方套路：首先看源码目录下的 README INSTALL
窍门：百度搜索 看别人怎么用的

有一些深层的问题：阅读源码 做二次开发
*/
find ./ -name "Makefile" -exec sed -i "s/CC = gcc/CC = arm-cortex_a9-linux-gnueabi-gcc/g" {} \;
sed
awk,在写 shell 做多文件处理时常用的命令

make
//观察是否为 ARM 版本的
file mjpg_streamer
2) 部署 mjpeg-streamer 到开发板上去
mkdir ../rootfs/home/bin
cp mjpg_streamer ../rootfs/home/bin/
mkdir ../rootfs/home/lib
cp mjpg-streamer/*.so /rootfs/home/lib/ -a
cp mjpg-streamer/www /rootfs/home/ -a
3）在开发板上运行 mjpg-streamer
/home/bin/mjpg_streamer --help

出现“error while loading shared libraries: libpthread.so.0。。。”问题，解决方式：
1)cp
/opt/arm-cortex_a9-eabi-4.7-eglibc-2.18/arm-cortex_a9-linux-gnueabi/sysroot/usr/lib/*.so*

rootfs/home/lib/ -a

2)cp /opt/arm-cortex_a9-eabi-4.7-eglibc-2.18/arm-cortex_a9-linux-gnueabi/sysroot/lib/*.so* rootfs/home/lib/ -a

3)export LD_LIBRARY_PATH=/home/lib 如果重启系统该命令要重新执行，可以加入etc/profile

vi rootfs/etc/profile

export LD_LIBRARY_PATH=/home/lib

/home/bin/mjpg_streamer -i "input_uvc.so --help"

/home/bin/mjpg_streamer -i "/home/lib/input_uvc.so -d /dev/video9 -y -r 320x240 -f 30" -o "/home/lib/output_http.so -w /home/www"

-i: 指定输入插件

-d: 指定访问的摄像头设备文件

-y: 采集图像的格式为 YUYV

-r: 采集图像的大小

-f: 帧频率

-o: 指定输出插件

-w: 网页资源文件所在目录

启动后如果打印输出了 “ error grabbing frames”类似信息，需要重新启动程序

/home/bin/mjpg_streamer -i "/home/lib/input_uvc.so -d /dev/video9 -y -r 320x240 -f 30" -o "/home/lib/output_http.so -w /home/www"

打开浏览器，输入“http://192.168.1.6:8080/”

如果手头没有摄像头硬件设备，可以使用如下策略

/home/bin/mjpg_streamer -i "input_testpicture.so -r 320x240 -d 500" -o "output_http.so -w /home/www"

或者将以上启动 mjpg_streamer 的方式写入 start.sh 脚本中

vi start.sh

27 #export LD_LIBRARY_PATH="$(pwd)"

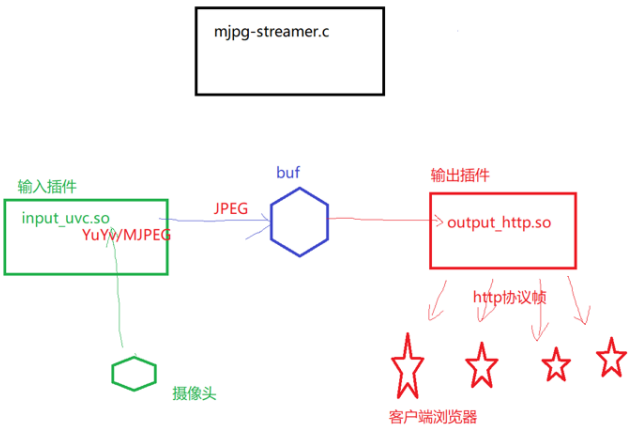
32 #/home/bin/mjpg_streamer -i "/home/lib/input_uvc.so -d /dev/video9 -y -r 320x240 -f 30" -o "/home/lib/output_http.so -w /home/www"

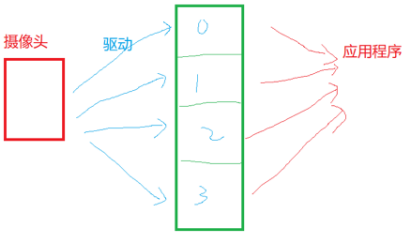
33 /home/bin/mjpg_streamer -i "input_testpicture.so -r 320x240 -d 500" -o "output_http.so -w /home/www"

cp start.sh ../rootfs/hom/bin

后续启动时可以使用脚本启动

4. mjpg-streamer 源码分析




4.1 mjpg_streamer.c 高内聚 低耦合

```
ctags -R *
int main(int argc, char *argv[])
{
    /*共享库的运行阶段加载有两种方式
      gcc xxx -o a.out -lpthread
      ./a.out 操作系统加载共享库

      程序中自主主动加载共享库（插件库）
    */
                                        "input_uvc.so"
    global.in[i].handle = dlopen(global.in[i].plugin, RTLD_LAZY)
        /*找到 input_uvc.so 中的 input_init 函数对应代码在内存中的位置*/
    global.in[i].init = dlsym(global.in[i].handle, "input_init");
    global.in[i].run = dlsym(global.in[i].handle, "input_run");
    /*执行 input_uvc.so 中的 input_init 函数*/
    global.in[i].init(&global.in[i].param, i)

                                            /*"output_http.so"*/
    global.out[i].handle = dlopen(global.out[i].plugin, RTLD_LAZY);
    global.out[i].init = dlsym(global.out[i].handle, "output_init");
    global.out[i].run = dlsym(global.out[i].handle, "output_run");
    global.out[i].init(&global.out[i].param, i)

    global.in[i].run(i)
```

```
        global.out[i].run(global.out[i].param.id);

        pause();
    return 0;

    }
```

4.2 输入插件 plugins/input_uvc/

按照 v4l2 编程步骤去操作 uvc 格式的摄像头

官方例程：capture.c

Video for linux 2 example (v4l2 demo) - MetalSeed - 博客频道 - CSDN.NET.png

vi plugins/input_uvc/input_uvc.c

```
/*打开摄像头 设置工作参数*/
int input_init(input_parameter *param, int id)
{
    init_videoIn(cams[id].videoIn, dev, width, he ight, fps, format, 1, cams[id].pglobal, id)
    {
        init_v4l2(vd)
        {
            /*打开"/dev/video9"设备文件*/
            vd->fd = OPEN_VIDEO(vd->videodevice, O_RDWR)
            /*查询当前硬件的工作能力*/
            xioctl(vd->fd, VIDIOC_QUERYCAP, &vd->cap)
            /*图像格式设置*/
            ret = xioctl(vd->fd, VIDIOC_S_FMT, &vd->fmt);
            ...
        }
    }
}
int input_run(int id)
{
    pthread_create(&(cams[id].threadID), NULL, cam_thread, &(cams[id]));
}
void *cam_thread(void *arg)
{
    while(!pglobal->stop)
    {
        uvcGrab(pcontext->videoIn)
        {
            video_enable(vd)
            {
                /*VIDIOC_STREAMON：让摄像头开始工作*/
                ret = xioctl(vd->fd, VIDIOC_STREAMON, &type);
            }
```

```
                        /*获取一帧图像*/
                        xioctl(vd->fd, VIDIOC_DQBUF, &vd->buf)

                    }
                    /* copy JPG picture to global buffer */
                     memcpy_picture(pglobal->in[pcontext->id].buf, pcontext->videoIn->tmpbuffer,
pcontext->videoIn->buf.bytesused);
                }
            }

BUG 修改：plugins/input_uvc/v4l2uvc.c
428         do{
429         memset(&vd->buf, 0, sizeof(struct v4l2_buffer));
430         vd->buf.type = V4L2_BUF_TYPE_VIDEO_CAPTURE;
431         vd->buf.memory = V4L2_MEMORY_MMAP;
432
433         ret = xioctl(vd->fd, VIDIOC_DQBUF, &vd->buf);
434         if(ret < 0) {
435               perror("Unable to dequeue buffer");
436              // goto err;
437         }
438         }while(ret < 0);
```

4.3 输出插件 plugins/output_http/

将图像数据封装成 http 数据包

通过 TCP 方式向客户端发送

vi plugins/output_http/output_http.c

```
    int output_init(output_parameter *param, int id)
    {
        port = htons(8080);
        ...
    }
int output_run(int id)
{
pthread_create(&(servers[id].threadID), NULL, server_thread, &(servers[id]));
}
void *server_thread(void *arg)
{
    socket(aip2->ai_family, aip2->ai_socktype, 0)
    bind(pcontext->sd[i], aip2->ai_addr, aip2->ai_addrlen)
    listen(pcontext->sd[i], 10)
    while(!pglobal->stop)
    {
        accept(pcontext->sd[i], (struct sockaddr *)&client_addr, &addr_len)
        pthread_create(&client, NULL, &client_thread, pcfd)
    }
```

```
}
void *client_thread(void *arg)
        {
          _readline(lcfd.fd, &iobuf, buffer, sizeof(buffer) - 1, 5)
          {
              _read(fd, iobuf, &c, 1, timeout)
              {
                   read(fd, &iobuf->buffer, IO_BUFFER)
              }
          }
```

建议：在此处加打印信息 打印从浏览器收到的请求 plugins/output_http/httpd.c

```
          776       printf("recv request from client :\n %s\n", buffer);

          else if(strstr(buffer, "GET /?action=stream") != NULL)
          {
               req.type = A_STREAM;//确定客户端请求类型
          }

           switch(req.type) {
                    case A_STREAM:
                         send_stream(lcfd.fd, input_number)
                         {
                              while(!pglobal->stop)
                              {
                                  /**/
                                  write(http 头信息)
                                  /*将 pglobal->buf 数据拷贝到 frame 缓冲区*/
                                    memcpy(frame, pglobal->in[input_number].buf,
frame_size);
                                    write(fd, frame, frame_size)
                                    write(http 尾信息)
                              }
                         }
                         break;
        }
```

让加的打印信息运行时有效：

```
cd /home/tarena/tcar/mjpg-streamer
make clean
make
cp input_uvc.so   ../rootfs/home/lib/
cp output_http.so ../rootfs/home/lib/
```

day02:

1、开机自启动视频服务器

web 前端：html css js

开机自启动视频服务器

vi rootfs/etc/init.d/rcS

13 source /etc/profile

14 /home/bin/mjpg_streamer -i "/home/lib/input_uvc.so -d /dev/video9 -y -r 320x240 -f 30" -o "/home/lib/output_http.so -w /home/www" &

在 PC 机的浏览器中输入 http://192.168.1.6:8080/stream_simple.html

2、分析浏览器显示内容的过程

整个过程浏览器向服务器发送了三次请求：

1）GET /stream_simple.html HTTP/1.1

2）GET /?action=stream HTTP/1.1

3）GET /favicon.ico HTTP/1.1

3、修改为启动 tcar.html 页面

cd /home/tarena/tcar/

mv rootfs/home/www/stream_simple.html rootfs/home/www/tcar.html

vi rootfs/home/www/tcar.html

3 <title>tcar</title>

浏览器中输入 “http://192.168.1.6:8080/tcar.html”

将 home/www 目录下除去 tcar.html favicon.ico

剩余文件全部删除

cp rootfs/home/www/tcar.html ./

cp rootfs/home/www/favicon.ico ./

rm rootfs/home/www/* -rf

mv tcar.html rootfs/home/www/

mv favicon.ico rootfs/home/www/

4、修改 tcar.html 页面

vi rootfs/home/www/tcar.html

参考提供给你的 tcar.html 位于 day02.rar 压缩包中

cp ...day02.rar/tcar.html rootfs/home/www/

为了让页面更生动，显示一部分图片

cp /mnt/hgfs/project/tcar_env/env/day02.rar/static/ rootfs/home/www/ -a

验证：浏览器中输入 192.168.1.6:8080/tcar.html

发现： 图片不能正常显示

解决方式：根据浏览器发送的请求“GET /static/images/front.png HTTP/1.1”

修改 webserver(httpd.c)

当收到类似请求时，

将图片文件 打开 读取图片数据 将图片数据发送给浏览器
即可。

5、修改 webserver 支持图片的显示

```
cd /home/tarena/tcar/mjpg-streamer
vi plugins/output_http/httpd.c
   void *client_thread(void *arg){

      848 len = MIN(MAX(strspn(pb,
"abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ._-1234567890/"), 0), 100);
make clean
make
cp output_http.so ../rootfs/home/lib/
```

重启开发板

浏览器输入 192.168.1.6:8080/tcar.html

这时 图片正常显示

注意 strspn 函数的使用，可参考 strspn.png

6、点击浏览器中的按钮 无法正常显示

```
cd /home/tarena/tcar/mjpg-streamer
vi plugins/output_http/httpd.c
```

将以下代码

```
802     } else if(strstr(buffer, "GET /?action=command") != NULL)
```

修改为

```
802     } else if(strstr(buffer, "GET /car_move?") != NULL) {
```

将以下代码

```
807             if((pb = strstr(buffer, "GET /?action=command")) == NULL) {
```

修改为

```
if((pb = strstr(buffer, "GET /car_move?")) == NULL) {
```

修改以下代码

```
813             pb += strlen("GET /?action=command");
```

修改为

```
813             pb += strlen("GET /car_move?");

822             memset(req.parameter, 0, len + 1);
823             strncpy(req.parameter, pb, len);
//新增打印信息
824             printf("%s\n", req.parameter);
```

在以下代码中加一行

```
931     case A_COMMAND:

936             command(lcfd.pc->id, lcfd.fd, req.parameter);
//新增加一行 不管发过来的命令是什么 以及能不能处理成功
//都重新把 tcar.html 页面回传给客户端
```

937 send_file(lcfd.pc->id, lcfd.fd, "tcar.html");

将 command 函数清空

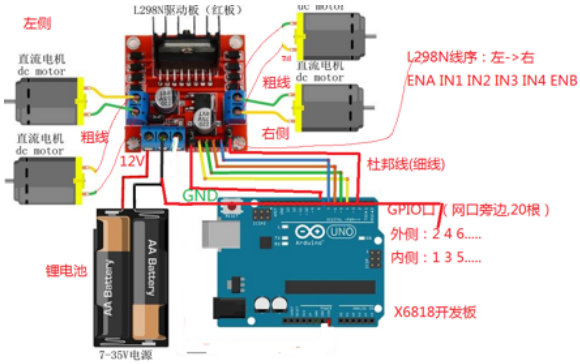
578 void command(int id, int fd, char *parameter)
579 {
580 }

make clean
make
cp output_http.so ../rootfs/home/lib/

重新启动开发板
浏览器输入 192.168.1.6:8080/tcar.html
点击浏览器中的按钮，还是正常显示 tcar.html 页面

观察串口终端中收到的命令
服务器收到了浏览器发送的类似于"cmd=front"

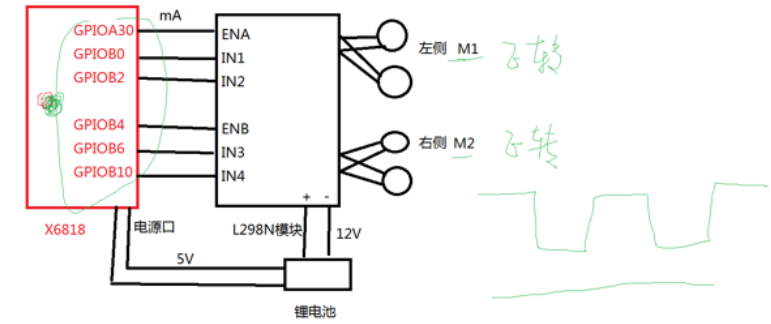
7、电机驱动




L298N线序：左->右
ENA IN1 IN2 IN3 IN4 ENB

注意：如果开发板和L298N是分别供电的
例如开发板使用的市电
L298N使用的是锂电池供电
做一下开发板和L298N的共地

L298N 地----》GPIO的 2引脚连接起来




M1正转：
GPIOA30:高
GPIOB0:高
GPIOB2:低

M1 反转：
GPIOA30:高
GPIOB0:低
GPIOB2:高

M1:停止
GPIOA30:低
GPIOB0:低
GPIOB2:低

M2同上

小车调速：

1.明确PWM接口特性

本质就是一个方波(一个高电平+一个低电平)

但是方波的高,低电平的时间宽度是可以调节的

周期是固定的！

例如：

PWM周期为10ms：高电平宽度1ms,低电平宽度9ms

PWM周期为10ms：高电平宽度5ms,低电平宽度5ms

PWM周期为10ms：高电平宽度8ms,低电平宽度2ms

2.概念：占空比

占空比：高电平所占的比例

例如：周期为10ms,高电平为1ms,占空比=10%

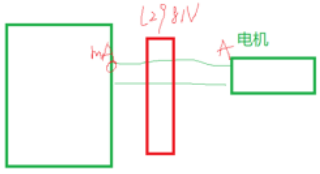
3.结论：

ENA,ENB就是直流电机的调速段,采用PWM来调

占空比越大,速度越快

注意：小车直流电机的PWM周期为10ms

小车要持续运行,ENA,ENB的PWM波必须持续产生



7.1 接线方式

参考 motor.zip/l298n_motor.png

7.2 驱动原理

参考 motor.zip/L298N 中文资料.doc

从精度角度： 伺服电机>步进电机>直流电机

控制电机的转动就是控制 ENA ENB IN1 IN2 IN3 IN4
CPU 上的 GPIO 管脚通过导线和 ENA ENB IN1 IN2 IN3 IN4 相连
结论控制电机就是控制 L298N
控制 L298N 就是控制对应的 CPU 管脚

7.3 驱动编程

L298N 驱动满足字符设备驱动编程
参考 motor.zip/emotor_drv.tar.bz2

```
cd /home/tarena/tcar

mkdir drivers
cd drivers
mkdir l298n
cd l298n
vi emotor_dev.c

    注意根据接线方式 调整 resource
vi emotor_drv.c
    static int emotor_probe(struct platform_device *pdev){
            gpio_direction_output(res->pres[i].gpio,0);//注册信息
            ....
vi Makefile

make //生成对应的.ko 文件

mkdir /home/tarena/tcar/rootfs/home/drivers/
cp *.ko /home/tarena/tcar/rootfs/home/drivers/

在开发板上安装驱动模块
insmod /home/drivers/emotor_dev.ko
insmod /home/drivers/emotor_drv.ko

ls /dev/emotor //产生了对应的设备文件
```

7.4 编写测试程序 验证驱动程序是否有效

注意：建议将小车架起来

vi emotor_test.c
arm-cortex_a9-linux-gnueabi-gcc emotor_test.c -o emotor_test -lpthread

cp emotor_test /home/tarena/tcar/rootfs/home/drivers/

测试驱动是否有问题
调试功能： 通过调节 ENA ENB 上占空比实现的

可能会出现的问题：
1）L298N 驱动板使用的是否为 12v 电源
2）开发板使用的是市电供电，需要做 L298N 驱动板和开发板的共地
3）线序是否有驱动程序中保持了一致

7.5 制作对应库文件 以备 mjpg-streamer 服务程序中调用
参考代码：emotor_hwlib.tar.bz2

mkdir /home/tarena/tcar/hwlib
cd /home/tarena/tcar/hwlib
mkdir emotor
cd emotor
vi emotor.c
vi emotor.h
arm-cortex_a9-linux-gnueabi-gcc -shared -fpic emotor.c -o libemotor.so
vi emotorlib_test.c

arm-cortex_a9-linux-gnueabi-gcc emotorlib_test.c -o emotor_test -lemotor -L. -lpthread

mkdir /home/tarena/tcar/rootfs/home/applib
cp libemotor.so /home/tarena/tcar/rootfs/home/applib
mkdir /home/tarena/tcar/rootfs/home/apptest
cp emotor_test /home/tarena/tcar/rootfs/home/apptest

在板子上执行
vi /etc/profile
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home/lib/:/home/applib/
source /etc/profile 或者重启开发板
insmod /home/drivers/emotor_dev.ko
insmod /home/drivers/emotor_drv.ko
/home/apptest/emotorlib_test

day03
1、mjpeg-streamer 中控制电机
1.1 代码修改
vi plugins/output_http/httpd.h 增加
118 void *emotor_handle;
vi plugins/output_http/output_http.c 增加

```
41 #include <dlfcn.h>

175         servers[param->id].conf.emotor_handle = dlopen("libemotor.so", RTLD_LAZY);
176         if(!servers[param->id].conf.emotor_handle)
177         {
178             printf("dlopen:%s\n", dlerror());
179         }
180         /*执行 emotor_open 打开电机设备文件*/
181         int (*emotor_open)(void) = NULL;
182       emotor_open = dlsym(servers[param->id].conf.emotor_handle,"emotor_open");
183         emotor_open();
184         /*开启调速线程*/
185         int (*emotor_pwm_task)(void) = NULL;
186                                                         emotor_pwm_task          =
dlsym(servers[param->id].conf.emotor_handle,"emotor_pwm_task");
187         emotor_pwm_task();
188         /*设置小车运行时的初始速度*/
189         void (*emotor_control_speed)(int)= NULL;
190                                                       emotor_control_speed       =
dlsym(servers[param->id].conf.emotor_handle,"emotor_control_speed");
191         emotor_control_speed(2000);
```

vi plugins/output_http/httpd.c 增加

```
44 #include "../../../hwlib/emotor/emotor.h"
45 #include <dlfcn.h>
```

在 command 函数中增加以下内容

```
void command(int id, int fd, char *parameter)
{
    void *emotor_handle = servers[id].conf.emotor_handle;
    static int init_speed = 2000;

    printf("%s\n", parameter);
    if(strcmp(parameter+4,"front") == 0)
    {
        dlsym(emotor_handle,emotor_control_direction)(CAR_FORWARD);
    }
    else if(strcmp(parameter+4, "back") == 0)
    {
        dlsym(emotor_handle,emotor_control_direction)(CAR_BACK);
    }
    else if(strcmp(parameter+4, "left") == 0)
    {
        dlsym(emotor_handle,emotor_control_direction)(CAR_LEFT);
    }
    else if(strcmp(parameter+4, "right") == 0)
    {
```

```
            dlsym(emotor_handle,emotor_control_direction)(CAR_RIGHT);
        }
        else if(strcmp(parameter+4, "stop") == 0)
        {
            dlsym(emotor_handle,emotor_control_direction)(CAR_STOP);
        }
        else if(strcmp(parameter+4, "speedup") == 0)
        {
            init_speed += 400;
            if(init_speed > 10000)
            {
                init_speed = 10000;
            }
            dlsym(emotor_handle,emotor_control_speed)(init_speed);
        }
        else if(strcmp(parameter+4, "speeddown") == 0)
        {
            init_speed -= 400;
            if(init_speed < 1000)
            {
                init_speed = 1000;
            }
            dlsym(emotor_handle,emotor_control_speed)(init_speed);
        }
    }
```

1.2 部署到开发板

```
make clean
make
cp mjpg_streamer ../rootfs/home/bin/
cp *.so ../rootfs/home/lib/
```

1.3 测试

重启开发板 启动 webserver

浏览器输入 192.168.1.6:8080/tcar.html

点击浏览器中的按钮尝试控制小车

特别注意： 5v 电源接开发板

把小车放在地下跑起来

限制：

1）市电供电的电源线

采用锂电池供电

将锂电池 5v 输出 剪断 接一个公头 把公头插入板子的市电供电口

2）网线

和上位机通信：

上位机浏览器和 webserver 通信

tftp 加载内核

nfs 挂在跟文件系统

3) 串口线
调试信息

解决网线问题：
1）调试 wifi 模块
2）将内核烧写到 emmc
3）将/home/tarena/tcar/rootfs 制作成 ext4 镜像烧写到开发板让板子使用 emmc 中的文件系统

2.配置内核支持 WIFI 80211 协议
内核添加：
cd /home/tarena/tcar/kernel
make menuconfig
Networking supports->
//支持 WIFI 网络协议 80211
Wireless->
[*] cfg80211 wireless extensions compatibility
[*] Wireless extensions sysfs files
Device Drivers->
Network device supports->
[*] Wireless LAN --->
//支持 AP 热点功能协议
<*> IEEE 802.11 for Host AP (Prism2/2.5/3 and WEP/TKIP/CCMP)
AP: access point
//有些 WIFI 需要固件程序(二进制文件而已)
[*]Support downloading firmware images with Host AP driver
[*]Support for non-volatile firmware download
make uImage 二进制文件
cp arch/arm/boot/uImage /tftpboot

3.编译 WIFI 驱动:
获取 wifi 驱动：env/wifi_ap.rar/mt7601u-master.tar.bz2
//mkdir /home/tarena/tcar/rootfs/home/wifi/
mkdir /home/tarena/tcar/wifi/drivers/ -p
cp ..../mt7601u-master.tar.bz2 /home/tarena/tcar/wifi/drivers/
cd /home/tarena/tcar/wifi/drivers/
tar -xvf mt7601u-master.tar.bz2
cd mt7601u-master
vi src/Makefile
271 LINUX_SRC=/home/tarena/tcar/kernel/
272 CROSS_COMPILE = arm-cortex_a9-linux-gnueabi-
make

cp src/os/linux/*.ko /home/tarena/tcar/rootfs/home/drivers/
vim ./etc/Wireless/RT2870AP/RT2870AP.dat
将 SSID=tarena_esd1909 修改为自己喜欢的名字
将 WPAPSK=12345678 修改位自己喜欢的密码

mkdir /home/tarena/tcar/rootfs/etc/Wireless/RT2870AP -p
cp ./etc/Wireless/RT2870AP/RT2870AP.dat /home/tarena/tcar/rootfs/etc/Wireless/RT2870AP/

4.下位机测试：
插入 USB 接口的 WIFI 模块到下位机的 USB 口(不好使,换一个)
下位机执行：
开发板加载驱动：
insmod /home/drivers/rtutil7601Uap.ko
insmod /home/drivers/mt7601Uap.ko
insmod /home/drivers/rtnet7601Uap.ko
ifconfig -a //获取 WIFI 的设备名 ra0 或者 wlan0，有线网卡名叫 eth0
ifconfig ra0 192.168.2.1 //给 WIFI 网卡模块配置 ip 地址
此时拿手机连接小车的 WIFI 路由器,一直转圈圈,因为小车路由器
还不具备给手机客户端分配 IP 地址的功能
问：如何解决呢?
答：利用大名鼎鼎的 dnamsaq 开源软件
此软件具备两个功能：
1.动态(DHCP)给别人分配 IP 地址
2.域名(例如：www.baidu.com)的解析功能(转换功能)

5.添加 DHCP 服务
上位机执行：
获取源码包：env/wifi_ap.rar/dnsmasq-2.66.tar.gz
cd /home/tarena/tcar/wifi/
cp dnsmasq-2.66.tar.gz ./

tar -xvf dnsmasq-2.66.tar.gz
cd dnsmasq-2.66
vim src/dnsmaq.c +304
将 304,305 行删掉
保存退出

export CC=arm-cortex_a9-linux-gnueabi-gcc
make
cp ./src/dnsmasq /home/tarena/tcar/rootfs/home/bin/
cp dnsmasq.conf.example /home/tarena/tcar/rootfs/etc/dnsmasq.conf
vim /home/tarean/tcar/rootfs/etc/dnsmasq.conf 修改为如下：第一列为行号
94 interface=ra0
99 listen-address=192.168.2.1,127.0.0.1
145 dhcp-range=192.168.2.1,192.168.2.150,255.255.255.0,12h
216 dhcp-host=11:22:33:44:55:66,192.168.2.1
312 dhcp-option=3,192.168.2.1
保存退出

在开发板运行：
重启下位机,千万别忘记用新的 uImage 启动下位机(包含 WIFI 协议)
mkdir /var/lib/misc/ -p

mkdir /var/run -p

vi /etc/init.d/rcS

```
insmod /home/drivers/rtutil7601Uap.ko
insmod /home/drivers/mt7601Uap.ko
insmod /home/drivers/rtnet7601Uap.ko
insmod /home/drivers/emotor_dev.ko
insmod /home/drivers/emotor_drv.ko
ifconfig ra0 192.168.2.1
#启动视频服务器
/home/bin/mjpg-streamer .....
/home/bin/dnsmasq -C /etc/dnsmasq.conf &
```

用手机直接连接热点,输入密码
至此手机可以通过 WIFI 和开发板进行无线通信了!

连接成功后，启动手机上的浏览器
输入 192.168.2.1:8080/tcar.html

day04:小车可以放在地上跑
让板子通过 emmc 加载内核

1、实现开机的自启动

1.1 /etc/profile 配置环境变量

vi rootfs/etc/profile

export LD_LIBRARY_PATH=/home/lib:/home/applib

1.2 /etc/init.d/rcS 开机自启动

vi rootfs/etc/init.d/rcS

```
mount -a

mkdir /dev/pts
mount -t devpts devpts /dev/pts
#热插拔事件发生时执行的命令
echo /sbin/mdev >/proc/sys/kernel/hotplug
mdev -s
telnetd &
source /etc/profile


insmod /home/drivers/rtutil7601Uap.ko
insmod /home/drivers/mt7601Uap.ko
insmod /home/drivers/rtnet7601Uap.ko
insmod /home/drivers/emotor_dev.ko
insmod /home/drivers/emotor_drv.ko
ifconfig ra0 192.168.2.1
/home/bin/dnsmasq -C /etc/dnsmasq.conf &
```

/home/bin/mjpg_streamer -i "/home/lib/input_uvc.so -d /dev/video9 -y -r 320x240 -f 30" -o "/home/lib/output_http.so -w /home/www" &

自启动成功后，测试已有功能是否可以正常运行

如果手机操作时感觉页面比较卡，

1）尝试将代码中的调试信息去掉

2）尝试开启多核

```
echo 1 >/sys/devices/system/cpu/cpu1/online
echo 1 >/sys/devices/system/cpu/cpu2/online
echo 1 >/sys/devices/system/cpu/cpu3/online
...
```

1.3 将内核烧写到 emmc 中去

注意分区：参考 porting/day01 内容

```
fdisk 2 3 0x100000:0x4000000 0x4100000:0x2f200000 0x33300000:0
```

```
tftp 48000000 uImage
mmc write 48000000 800 3000
setenv bootcmd mmc read 48000000 800 3000 \;bootm 48000000
saveenv
```

1.4 通过 emmc 加载根文件系统

1.4.1 制作 ext4 类型的根文件系统镜像

```
dd if=/dev/zero of=rootfs_ext4.img bs=1k count=131072
sudo mkfs.ext4 rootfs_ext4.img
sudo mount rootfs_ext4.img /mnt/esd1909
sudo cp rootfs/* /mnt/esd1909/ -a
sudo umount /mnt/esd1909
```

1.4.2 烧写镜像文件到根文件系统分区

```
cp rootfs_ext4.img /tftpboot/
```

```
tftp 48000000 rootfs_ext4.img
mmc write 48000000 20800 0x40000
setenv bootargs root=/dev/mmcblk0p2 rootfstype=ext4 init=/linuxrc console=ttySAC0 maxcpus=1 lcd=wy070ml tp=gslx680 ip=192.168.1.6:192.168.1.8:192.168.1.1:255.255.255.0
saveenv
```

1.5 后续开发时 可以通过 nfs 方式挂载应用文件系统

当板子启动成功后，在板子上执行

```
mount -t nfs -o nolock 192.168.1.8:/home/tarena/tcar/rootfs /mnt
```

把板子上的/mnt 目录和上位机上的/home/tarena/tcar/rootfs 接通

如果需要向开发板使用的根文件系统中部署新内容

可以按如下步骤操作

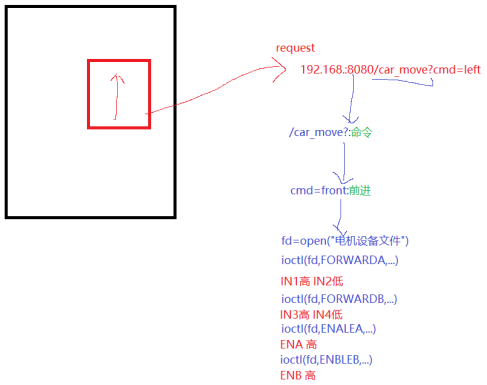
1)cd /home/tarena/tcar

2)touch xxx.config
打算将该配置文件部署到开发板的根文件系统/etc 目录

3) cp xxx.config rootfs/

4) 在开发板上执行
mv /mnt/xxx.config /etc

问题：手机浏览器中的按钮是如何控制到小车的前进的?




---

年前项目内容回顾：

1、搭建开发环境

1.1 装 ubuntu 系统 18.04

1.2 装必要的工具软件 apt-get

gcc
vim
nfs server
tftp server
kermit
ctags
...

1.3 交叉编译工具

sudo chmod 777 /opt -R
cp arm-cortex_a9-eabi-4.7-eglibc-2.18.tar.gz /opt
cd /opt
tar xf arm-cortex_a9-eabi-4.7-eglibc-2.18.tar.gz
rm arm-cortex_a9-eabi-4.7-eglibc-2.18.tar.gz
vi /home/tarena/.bashrc
export PATH=$PATH:/opt/arm-cortex_a9-eabi-4.7-eglibc-2.18/bin/
source /home/tarena/.bashrc

2、烧写开发板 使开发板可以正常运行 linux 系统

2.1 编译烧写 uboot 到开发板 （烧写步骤可以参考 ARM 裸板阶段 day01）

2.2 编译 linux 内核

cd /home/tarean
mkdir tcar
cd tcar
cp kernel.tar.bz2 ./
tar xf kernel.tar.bz2
rm kernel.tar.bz2
cd kernel
/*lcd 花屏的同学 需要给内核打补丁
屏幕显示正常的 不需要打补丁
*/
cp kernel.patch ./
patch -p1 <kernel.patch

cp arch/arm/configs/x6818_defconfig .config
make uImage
报错 “mkimage no found” ...

2.3 开发板使用新内核

方式一：告诉 uboot 从 tftp server 加载启动 linux 内核 //一般用于开发阶段

1）cp arch/arm/boot/uImage /tftpboot/

2）配置 uboot 从 tftp server 加载启动 linux 内核

setenv bootcmd =ping 192.168.1.8\;ping 192.168.1.8\;tftp 48000000 uImage \;bootm 48000000

saveenv

注意：ping 192.168.1.8,是因为板子上的网卡有问题

方式二：将 linux 内核烧写到 emmc 中去 每次启动都从 emmc 中加载 linux//项目发布

1)cp arch/arm/boot/uImage /tftpboot/

2)保证可以 ping 通开发板

a)桥接模式

b)开发板和 ubuntu 系统处于同一网段

开发板

printenv
setenv ipaddr 192.168.1.6
setenv serverip 192.168.1.8
saveenv

ubuntu

192.168.1.8

c)设置用于桥接的网卡

验证：在串口中终端 ping 服务器

ping 192.168.1.8

显示“host 192.168.1.8 is alive” 实验成功
显示 host 192.168.1.8 is not alive 实验不成功

3)下载内核镜像

//sudo /etc/init.d/tftpd-hpa restart

tftp 48000000 uImage

4)烧写镜像到 emmc

mmc write 48000000 800 3000

5)通知 uboot 每次自动从 emmc 加载启动 linux 内核

setenv bootcmd mmc read 4800000 800 3000 \; bootm 48000000

saveenv

6)重启开发板

内核正常加载启动

如果根文件系统没有配置好，会发现内核恐慌 不断重启

2.4 通过 nfs 方式挂载根文件系统

cd tcar

cp rootfs.tar.gz ./

sudo tar xf rootfs.tar.gz

rm rootfs.tar.gz

sudo vi /etc/exports

/home/tarena/tcar/rootfs *(rw,sync,no_root_squash)

sudo /etc/init.d/nfs-kernel-server restart

在串口终端的 uboot 中设置 bootargs

setenv bootargs root=/dev/nfs nfsroot=192.168.1.8:/home/tarena/tcar/rootfs/ ip=192.168.1.6:192.168.1.8:192.168.1.1:255.255.255.0 console=ttySAC0 maxcpus=1 init=/linuxrc lcd=wy070ml tp=gslx680-linux

saveenv

3、视频服务器的搭建

4、小车的行进控制

---

day05

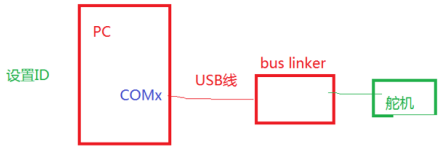
1、认识舵机

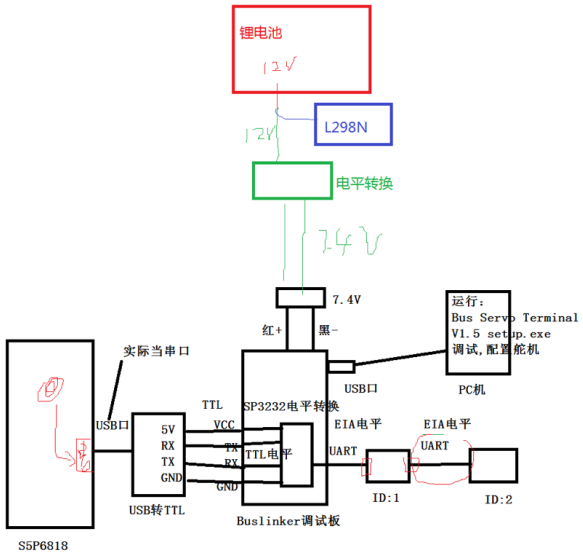
将摄像头固定在舵机上，

通过舵机输出转轴的转动 带动摄像头的转动

从而可以达到拍摄不同角度

阅读 servo.zip/LX-16A 串口舵机说明书.pdf （10 分钟的时间）

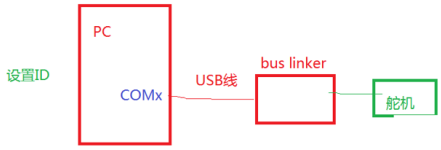


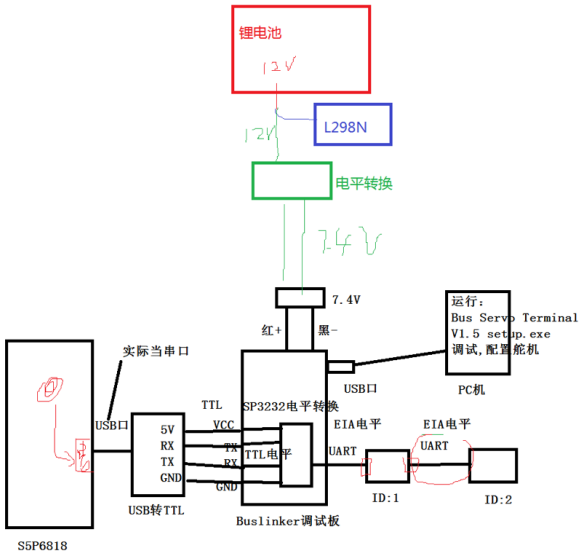




2、舵机设置方式

1）在 PC 机上安装 ch341ser.exe （Buslinker usb 口的驱动程序）
Bus Servo Terminal V1.5 setup.exe（用于设置舵机 ID 的软件）

2）通过 USB 线将 Buslinker 调试板和 PC 相连
观察是否多出一个串口设备（通过设备管理器）




3) 某一个舵机接到 Buslinker 调试板
4) 启动 PC 上的 Bus Servo Terminal 设置该舵机 ID=1
参数设置----》读取-----》修改 ID----》设置
5) 换另外舵机链接到 Buslinker 调试板
6) 通过 Bus Servo Terminal 设置 ID=2

连线参考 servo.png 最上方的图示

10 分钟时间设置舵机 ID

7) 断开连接 Buslinker 调试板和 PC 的 USB 线
8) 串联两个舵机
9) USB 转串和 Buslinker 调试板相连
10)将 USB 转串的 USB 端插入开发板的 USB 口
11) 连接电源
连线参考 servo.png 下方的图示

接好后在开发板的/dev 会产生一个新的设备文件/dev/ttyUSB0 或者/dev/ttyUSB1

后续运行在 ARM core 中的程序如果要想控制舵机
就是通过操作/dev/ttyUSB0 或者/dev/ttyUSB1 实现的

注意：内核中完美的支持了 USB 转串的设备驱动
有了该驱动程序 操作 ttyUSB0/1 时可以把视为普通串口设备
其操作方式是和/dev/ttySAC0 是完全相同的

3、和舵机通信的通信协议

15 分钟的时间去读

servo.zip/乐幻索尔总线舵机通信协议.pdf

指令的格式：

0x55 0x55 ID length cmd 参数 N 校验和

说明：

0x55 0x55:包头,表示开始传输数据,便于将来示波器看波形

ID：指定访问哪个舵机

length:包的长度-3

cmd：命令,例如：转动角度或者转动时间

参数：哪个角度或者多长时间

校验和：防止数据传输有误

校验和=~(ID+.....+参数) & 0xFF

例如：让舵机在 500ms 转动到 100 度,命令如下：

指令名 SERVO_MOVE_TIME_WRITE 指令值 1 数据长度 7：

参数 1：角度的低八位。

参数 2：角度的高八位。范围 0~1000，

对应舵机角度的 0~240°，即舵机可变化

参数 3：时间低八位。

参数 4：时间高八位，时间的范围 0~30000 毫秒。

该命令发送给舵机，舵机将在

参数时间内从当前角度匀速转动到参数角度。

该指令到达舵机后，舵机会立即转动。

最终 CPU 按照以下指令格式发送给舵机即可：

固定指令头 舵机 ID 数据长度 指令值

0x55 0x55 1 7 1 角度低 8 角度高 8 时间低 8 时间高 8 校验

4、linux 如何访问串口设备

linux 访问串口设备类似于 lcd camer ...

有着标准的套路

可以百度搜索“linux 串口编程”

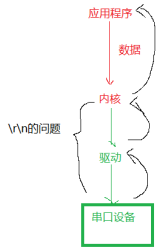
```
fd = open("/dev/ttyUSB0", O_RDWR);

/*串口工作参数的设置： 115200 8n1 */

read(fd, buf, len) ;

write(fd, buf, len);

close(fd);
```

参考代码：servo.zip/servo_hwlib.tar.bz2

如何验证自行编写的串口的代码是否正常工作?
USB 转 TTL 一端接到 PC（设备管理器中会多出一个串口设备）
另一端接到开发板 UART2
USB 转 TTL UART2
GND---------->GND
TXD---------->RX
TXD---------->TX
5V---------->DC 5V(板子电源接口附近)

自行编程实现：
mkdir /home/tarena/tcar/hwlib/servo
cd /home/tarena/tcar/hwlib/servo
vim servo.h
vim servo.c
uart_open(){...}
int uart_set(int baude,int c_flow,int bits,char parity,int stop){...}
servo_init(){}
int servo_move(int id, int position, int time)
id,要操作哪个舵机
position,转动的角度
time,花多长时间转到指定的角度
vim servolib_test.c
vim Makefile
make

在开发板上运行舵机测试程序，看舵机是否有响应
如果有响应开始以下步骤，如果没有，要查前面的内容找 BUG




day06:
1、web 请求分析

浏览器中输入 http://192.168.1.6:8080/tcar.html 整个服务器是如何响应?

服务器收到的请求： GET /tcar.html HTTP/1.1
服务器会将客户端请求的 tcar.html 页面内容封装成 http 响应发送给浏览器
有客户端连接服务器会执行：plugins/output_http/httpd.c
//创建新的孙子线程 为客户端提供服务

```
1013      if(pthread_create(&client, NULL, &client_thread, pcfd)

void *client_thread(void *arg){
    //接收客户端请求
    cnt = _readline(lcfd.fd, &iobuf, buffer, sizeof(buffer) - 1, 5)
    ...
    else
    {
        req.type = A_FILE;
    }
    841       case A_FILE:
    {
        //将客户端请求的文件封装成 http 协议响应数据发送给浏览器
        send_file(lcfd.pc->id, lcfd.fd, req.parameter){
            ...
        }

    }
}
```

浏览器收到服务发送过来的 tcar.html 页面内容，会解析该内容
解析过程中遇到类似于“src="/static/images/front.png"”
还向服务器发送新的请求
服务器收到的请求： GET /static/images/front.png HTTP/1.1
服务器会创建新的孙子进程

```
1013      if(pthread_create(&client, NULL, &client_thread, pcfd)
          void *client_thread(void *arg){
              //将客户端请求的文件封装成 http 协议响应数据发送给浏览器
              send_file(lcfd.pc->id, lcfd.fd, req.parameter){
          }

          孙子线程结束
```

当浏览器解析 tcar.html 页面内容时遇到 src="/?action=stream" 也会给服务器发送请求
服务器收到的请求：src="/?action=stream"

服务器会创建新的孙子进程

```
1013      if(pthread_create(&client, NULL, &client_thread, pcfd)
          void *client_thread(void *arg){
              cnt = _readline(lcfd.fd, &iobuf, buffer, sizeof(buffer) - 1, 5)
              else if(strstr(buffer, "GET /?action=stream") != NULL) {
                  req.type = A_STREAM;
                  ...
              }

              case A_STREAM:
                  send_stream(lcfd.fd, input_number){
                      while(!pglobal->stop){
```

```
                    ...
                    //取出图像数据
                    memcpy(frame, pglobal->in[input_number].buf, frame_size);
                    //发送图像数据
                    write(fd, frame, frame_size)
                }
            }
```

总结：当浏览器中输入 http://192.168.1.6:8080/tcar.html
浏览器依次向服务器发送了多个请求
服务器也会为每次请求创建一个新的孙子线程
为该请求提供服务
提供完服务服务器会自动销毁该孙子线程

当在 web 页面点击前进按钮时发生了什么?
浏览器向 server 发送了 请求 “http://192.168.1.6:8080/car_move?cmd=front”
服务器是如何响应该请求的呢?
客户端连接服务器会导致服务器创建孙子线程

```
pthread_create(&client, NULL, &client_thread, pcfd)
```

孙子线程去接收客户端发送过来的请求 解析请求 处理请求 返回响应数据

```
void *client_thread(void *arg){
    //接收请求
    _readline(lcfd.fd, &iobuf, buffer, sizeof(buffer) - 1, 5)

    //解析请求
    else if(strstr(buffer, "GET /car_move?") != NULL)
    {
        req.type = A_COMMAND;
    }


    case A_COMMAND:
        //处理请求
        command(lcfd.pc->id, lcfd.fd, req.parameter){

        }
        //返回响应数据 将 tcar.html 页面回传给客户端
        send_file(lcfd.pc->id, lcfd.fd, "tcar.html");
}
```

2、在 web server 中添加处理舵机操作的逻辑

vi plugins/output_http/httpd.h

```
119     void *servo_handle;
```

vi plugins/output_http/output_http.c

```
192     servers[param->id].conf.servo_handle = dlopen("libservo.so", RTLD_LAZY);
193     if(!servers[param->id].conf.servo_handle)
194     {
195         printf("dlopen:%s\n", dlerror());
```

```
196         }
197                 int   (*servo_init)(void)   =   dlsym(servers[param->id].conf.servo_handle,
"servo_init");
198         servo_init();
```

vi plugins/output_http/httpd.c

```
584         /*垂直方向初始值*/
585         static int v_pos = 500;
586         /*水平方向初始值*/
587         static int h_pos = 500;

591             int   (*servo_move)(int,   int,   int)   =   dlsym(servers[id].conf.servo_handle,
"servo_move");

631         else if(strcmp(parameter+4, "servo_up"))
632         {
633             v_pos += 100;
634             if(v_pos > 1000)
635                 v_pos = 1000;
636             servo_move(1, v_pos ,500);
637         }
638         else if(strcmp(parameter+4, "servo_down"))
639         {
640             v_pos -= 100;
641             if(v_pos < 0)
642                 v_pos = 0;
643             servo_move(1, v_pos,500);
644         }
645         else if(strcmp(parameter+4, "servo_left"))
646         {
647             h_pos += 100;
648             if(h_pos > 1000)
649                 h_pos = 1000;
650             servo_move(2, h_pos,500);
651         }
652         else if(strcmp(parameter+4, "servo_right"))
653         {
654             h_pos -= 100;
655             if(h_pos < 0)
656                 h_pos = 0;
657             servo_move(2, h_pos,500);
658         }
```

```
cd /home/tarena/tcar/mjpg-streamer
make clean
make
```

将 server 端的内容更新到小车的根文件系统中去

1）将插件库拷贝到　rootfs/home/lib
cp output_http.so ../rootfs/home/lib/
2) 将舵机硬件操作的库函数拷贝到 rootfs/home/applib
cd /home/tarena/tcar/hwlib/servo/
cp libservo.so /home/tarena/tcar/rootfs/home/applib/
3) 在开发板上执行
mount -t nfs -o nolock 192.168.1.8:/home/tarena/tcar/rootfs /mnt
cp /mnt/home/applib/libservo.so /home/applib/
cp /mnt/home/lib/output_http.so /home/lib/

3.tcar.html 页面的修改
为了修改界面的方便
将开发板改成通过 nfs 方式挂载文件系统，不断尝试修改 html 页面

setenv bootargs root=/dev/nfs nfsroot=192.168.1.8:/home/tarena/tcar/rootfs init=/linuxrc console=ttySAC0 maxcpus=1 lcd=wy070ml tp=gslx680 ip=192.168.1.6:192.168.1.8:192.168.1.1:255.255.255.0
saveenv
重启开发板

在 ubuntu 系统中
cd tcar/rootfs/home/www/
vi tcar.html

建议：使用 env 中提供给大家的 web 就可以了
cp /mnt/hgfs/project/tcar_env/env/servo.zip/web_serve.tar.bz2/mjpg-streamer/www/tcar.html /home/tarena/tcar/rootfs/home/www

17:30
1)将舵机操作封装成库函数并测试是否好用
2）实现 web 页面 建议使用提供给你的 tcar.html 并部署
3）web 页面点击了 up 按钮 server 中会收到 car_move?cmd=servo_up
server 中要去响应该命令，参考“在 web server 中添加处理舵机操作的逻辑 ”章节
4）思考 web 页面中点击 up 按钮是怎么让舵机动起来的?

4、超声波测距 实现避障功能
distace_hcsr.zip

4.1 超声波测距的原理

s=340*t/2
4.2 超声波模块的接线方式
HC-SR504　开发板
VCC　板子电源接口旁边 J42 1 引脚
GND　板子电源接口旁边 J42 2 引脚

TRIGGER　J13 9引脚
ECHO　　J13 11引脚

4.3 HC-SR504驱动编程

一定是满足字符设备驱动编程框架

```
cd /home/tarena/tcar/drivers
mkdir hc-sr504
cd hc-sr504
vi hcsr_dev.c
vi hcsr_drv.c
vi Makefile
make
//将生成的两个.ko文件部署开发的/home/drivers
//根据自身开发板挂载根文件系统的不同采用不同的部署方式
//如果是板子启动后直接挂载了/home/tarena/tcar/rootfs作为根文件系统
cp *.ko ../../rootfs/home/drivers/

vi hcsr_test.c
arm-cortex_a9-linux-gnueabi-gcc hcsr_test.c -o hcsr_test
cp hcsr_test ../../rootfs
```

在开发板上安装以上两个ko文件，在开发板上执行

```
insmod /home/drivers/hcsr_drv.ko
insmod /home/drivers/hcsr_dev.ko
./hcsr_test
```

观察打印的距离值是否准确

day07:

注意：读取距离时绝大多数值正常
偶尔会出现一些不正常的值
可以考虑使用软件的方式把值过滤掉
例如：取10次值，剔除掉最大的和最小的 剩余8个求和取平均值

1、将超声波模块操作函数封装成库 供webserver使用

```
cd /home/tarena/tcar/hwlib
mkdir hcs04
cd hcs04
vi hcs04.c
```

封装了一个线程
该线程中获取一组距离值
过滤掉干扰值
使用该距离值判断是否要停车

```
vi hcs04.h
```

vi Makefile

vi distancelib_test.c

make

在开发板上执行

1）安装对应的驱动模块

先通过 lsmod 观察电机驱动模块是否安装成功

如果未安装，可以通过以下命令安装

insmod /home/drivers/emotor_dev.ko

insmod /home/drivers/emotor_drv.ko

通过 lsmod 观察超声波模块是否安装成功

如果未安装，可以通过以下命令安装

insmod /home/drivers/hcsr_dev.ko

insmod /home/drivers/hcsr_drv.ko

注意：后续可以将以上指令放入 vi /etc/init.d/rcS 这样开机自动安装

2）开启 distancelib_test

调整特定障碍与小车的距离 过近时看小车是否停止

2、封装到 webserver

cd /home/tarena/tcar/mjpg-streamer

vi plugins/output_http/output_http.c

```
204     int (*dis_open)(void) = dlsym(dis_handle, "distance_open");
205     dis_open();
206     int (*dis_task)(void) = dlsym(dis_handle, "distance_task");
207     /*开启避障线程*/
208     dis_task();
```

make clean

make

//将 out_http.so 共享库部署到开发板上去

cp output_http.so ../rootfs/home/lib/

验证启动 web server 是否就可以实现小车的避障

提示：如果实现避障功能，

电机驱动模块要安装

超声波驱动模块要安装

libdistance.so 和 libemotor.so 要正常部署到根文件系统的 home/applib 目录下去

启动 web server 观察是否实现了避障 //目前 web server 开启自启动

3、红外学习模块

infrared.zip

阅读红外学习模块使用手册

3.1 工作原理

宏观上看要使用红外学习模块：

1）学习

记录空调遥控器打开空调时 发射的红外信号值

2）发射

将记录下来的空调要控制发射的打开空调的信号值

由咱们自己的红外学习模块发射出去

将空调遥控器发射开空调的信号记录下来

以后需要通过红外学习模块开空调时，把以上记录下来的值发出去

3.2 在 PC 机调试该模块

1）在 PC 机上安装驱动 YS-USB to TTL 串口调试器--驱动程序（CH340T）.zip

2）将红外学习模块与 USB-TTL 相连

| 红外模块 | USB-TTL |
|---|---|
| 3.3 | 3.3 |
| RX | TX |
| TX | RX |
| GND | GND |

3) 将 USB-TTL 与 PC 相连

4）启动串口调试软件 stc-isp

5）配置端口号 （要查 PC 机上的设备管理器）

波特率设置为 9600

6）按下红外 学习模块上的按键进入学习状态

7）将家中空调遥控器对准学习模块 （大约 5cm 的距离）

按下空调遥控器上的开关

8）观察串口软件中的信息

9）可以将红外学习的红头发射头对准空调

10）在串口调试软件中将学习到的开空调的数据

通过点击“发送数据按钮”

将数据发送出去

观察空调是否可以正常启动

如果不可以正常启动，研究咱们前面的步骤有什么问题，一定调通

3.3 在开发板调试该模块

1）将红外学习模块与开发板相连

| 红外学习模块 | 开发板 |
| --- | --- |
| 3.3 | J16 4引脚 |
| RX | J16 3引脚 |
| TX | J16 2引脚 |
| GND | J16 1引脚 |

2）编程实现

内核中自带了 UART 的驱动程序

对应的设备文件：ls /dev/ttySAC*

fd = open("/dev/ttySAC2", O_RDONLY);

设置波特率 9600 8n1 115200 无流控 //可以参考舵机模块的程序

write(fd, 学习到的开空调的 236 个字节， 236); //红外学习模块对准空调

close(fd);

实现步骤：

```
cd /home/tarena/tcar/hwlib
mkdir infrared
cd infrared
vi infrared.c
vi infrared.h
vi libinfrared_test.c
vi Makefile

make
```

在开发板上测试

/home/apptest/libinfrared_test on

观察是否可以正常开启空调 注意红外学习模块的角度

保证该实验调试通过

3.4 通过 web 去控制空调的打开关闭

3.4.1 web 页面端

```
vi rootfs/home/www/tcar.html
```

具体修改内容参考例子程序

3.4.2 webserver

```
cd /home/tarena/tcar/mjpg-streamer
vi plugins/output_http/httpd.h
    120     void *infrared_handle;
vi plugins/output_http/output_http.c
    200         servers[param->id].conf.infrared_handle = dlopen("libinfrared.so", RTLD_LAZY);
```

```
201         if(!servers[param->id].conf.infrared_handle)
202         {
203             printf("dlopen:%s\n", dlerror());
204         }
205                                         void (*infrared_init)(void) =
dlsym(servers[param->id].conf.infrared_handle, "infrared_init");
206         infrared_init();
```

vi plugins/output_http/httpd.c

```
593         void (*infrared_control)(int) = dlsym(servers[id].conf.infrared_handle,
"air_cond_control");

665         else if(strcmp(parameter+4, "air_on") == 0)
666         {
667             infrared_control(1);
668         }
669         else if(strcmp(parameter+4, "air_off") == 0)
670         {
671             infrared_control(0);
672         }
```

```
make clean
make
cp output_http.so   ../rootfs/home/lib/
```

重启开发板
在浏览器中输入 http://192.168.1.6:8080/tcar.html

点击开关空调的按钮看是否可以控制空调的开或者关

注意：红外模块要对准空调
可以考虑把红外模块也固定舵机上去
可以通过调节舵机 调节红外模块对准空调

12:00 之前 完成通过页面控制空调的开关
代码参考
后台代码 infrared_hwlib.tar.bz2
web 代码 infrared.zip/webserver