Serialport 使用手册3.1

1. 通讯协议简介

> 通讯协议是联系视觉和电控之间的一座重要的桥梁。它的主要作用是将视觉识别所得到的坐标、距离、角度等信息发送给电控，电控根据接收到的数据像机器人发出相应的调控指令来完成相应的任务。

2. 通讯协议的组成

2.1 串口的初始化以及相关参数的设置

2.2 对发送数据的转换和封装

2.3 写入CRC校验值并且发送数据

2.4 接受电控传来的指令，根据指令内容调节程序相关设置

3. 通讯协议的相关函数使用说明

> 在代码也有注释，直接看代码就行。
>
> CRC.h 和CRC.CPP部分只需要了解做什么的就行就行，具体原理有兴趣自行了解，其实就是一方发数据的时候，使用这些数据利用一定算法生成CRC校验值放进数据里，另一方接收这些数据，用同样的算法计算出CRC校验值，如果和发过来的校验值相同，那么就认为这组数据是正确的的。
>
> 重点看serialport.cpp 和serialport.h

```
bool SerialPort::get_Mode(int &mode, int &shoot_speed, int &my_color,
                          double &gyro_yaw, double &gyro_pitch, int &anti_flag)
{
    //该函数同样需要修改长度，放在宏定义 RDATA_LENGTH
    int bytes;
    int result = ioctl(fd, FIONREAD, &bytes);//该函数用来查看输入缓冲区的总字节数，其中bytes为输入缓冲区的字节数
    if (result == -1)
    {
        printf("ioctl: %s\n", strerror(errno));
        return false;
    }
//    cout<<"输入缓冲区字节数:  "<<bytes<<endl;
    if (bytes == 0)
    {
//        cout << "缓冲区为空! " << endl;
        return true;
    }
    /*
     * 下面这段代码的意思是：缓冲区有多少字节我们就读多少字节，这个过程也清空了缓冲区；
     * 然后我们提取出最后一组数据，也就是最新数据。
     */
    bytes = read(fd, rdata, bytes);//read函数将缓冲区的数据读到rdata数组
    int FirstIndex = -1;
    int flag = 0;
    //找到最初一个帧头，把上一帧的残留数据舍去
    for(int i = 0; i < bytes; i++)
```

```
    {
        if( rdata[i] == 0xA5 && FirstIndex == -1&&
Verify_CRC8_Check_Sum(&rdata[i], 3) &&
Verify_CRC16_Check_Sum(&rdata[i],RDATA_LENGTH))
        {
            FirstIndex = i;
            flag = 1;
        }
        if(flag)
        {
            break;
        }
    }
    int max_mul = (bytes - FirstIndex) / RDATA_LENGTH;
//    cout << "FirstIndex: " << FirstIndex <<endl;
//    cout << "bytes: " << bytes << endl;
//    cout << "max_mul: " << max_mul <<endl;

    /*
     * FirstIndex + (max_mul - 1)*RDATA_LENGTH的意思是最后一组数据的第一位的索引
     * FirstIndex + (max_mul - 1)*RDATA_LENGTH          第0位：帧头
     * FirstIndex + (max_mul - 1)*RDATA_LENGTH + 1      第1位：和电控商量，可以当正常
数据接受
     * FirstIndex + (max_mul - 1)*RDATA_LENGTH + 2      第2位：CRC8校验值，未写出，在
Verify_CRC8_Check_Sum函数加上，不用修改
     * FirstIndex + (max_mul - 1)*RDATA_LENGTH + 3-13   第3-13位：接受数据
     * FirstIndex + (max_mul - 1)*RDATA_LENGTH + 14-15  第14-15位：CRC16校验的两个字
节，在这里我们没有校验，但是要预留这两个字节数
     */

    if(FirstIndex != -1)
    {
        if(rdata[FirstIndex + (max_mul - 1)*RDATA_LENGTH] == 0xA5  &&
Verify_CRC8_Check_Sum(&rdata[FirstIndex + (max_mul - 1)*RDATA_LENGTH], 3) )
        {
           /*
            * mode、shoot_speed、my_color、anti_flag 为整型数据
            * gyro_yaw、gyro_pitch、为浮点型数据，自行百度union结构体是怎么进行数据转换
的：内存共享
            */
            mode  = (int)rdata[FirstIndex + (max_mul - 1)*RDATA_LENGTH+1];
            shoot_speed  = (int)rdata[FirstIndex + (max_mul - 1)*RDATA_LENGTH
+3];
            my_color  = (int)rdata[FirstIndex + (max_mul - 1)*RDATA_LENGTH+ 4];

            charTofloat cf;
            for(int i = 0; i < 4; i++)
            {
                cf.value[i] = rdata[FirstIndex + (max_mul - 1)*RDATA_LENGTH+5+i];
            }
            gyro_yaw = -(double)cf.fvalue;

            for(int i = 0; i < 4; i++)
            {
                cf.value[i] = rdata[FirstIndex + (max_mul - 1)*RDATA_LENGTH+9+i];
            }
            gyro_pitch = -(double)cf.fvalue;
```

```
            anti_flag = (int)rdata[FirstIndex + (max_mul - 1)*RDATA_LENGTH +13];
        }
        else
        {
            cout<<"CRC8 Check False!!!"<<endl;
        }
    }
    return true;
}
```

```
void SerialPort::TransformData(const VisionData &data)
{
    //怎么修改发送数据：按下面模版更改数据，特别留意最后一个函数输入的长度，怎么计算：1个帧头+1（整型数据） + 1个CRC8校验值 + 4 * 浮点数据的个数 + 1 *整形数据个数 + 2个CRC16校验值
    Tdata[0] = 0xA5;
    Tdata[1] = data._mode;
    Append_CRC8_Check_Sum(Tdata, 3);//前三位有CRC8校验，第一位为帧头，第二位可以当正常数据发，第三位为CRC校验值

    //浮点数据用4个字节； 整型数据用1个字节
    //浮点数据: yaw(x); pit(y); dis(z)
    Tdata[3] = data.pitch_angle.c[0];
    Tdata[4] = data.pitch_angle.c[1];
    Tdata[5] = data.pitch_angle.c[2];
    Tdata[6] = data.pitch_angle.c[3];

    Tdata[7] = data.yaw_angle.c[0];
    Tdata[8] = data.yaw_angle.c[1];
    Tdata[9] = data.yaw_angle.c[2];
    Tdata[10] = data.yaw_angle.c[3];

    Tdata[11] = data.dis.c[0];
    Tdata[12] = data.dis.c[1];
    Tdata[13] = data.dis.c[2];
    Tdata[14] = data.dis.c[3];
    //整型数据: 一些标志位
    Tdata[15] = data.drop_frame;
    Tdata[16] = data.isFindTarget;
    Tdata[17] = data.isfindDafu;
    Tdata[18] = data.nearFace;
    Tdata[19] = data.anti_top;
    Tdata[20] = data.anti_top_change_armor;

    Append_CRC16_Check_Sum(Tdata, 23);//后两位为CRC校验值，这里是CRC16也就是检验两个字节，留意23怎么计算的
}
```

4. 使用流程

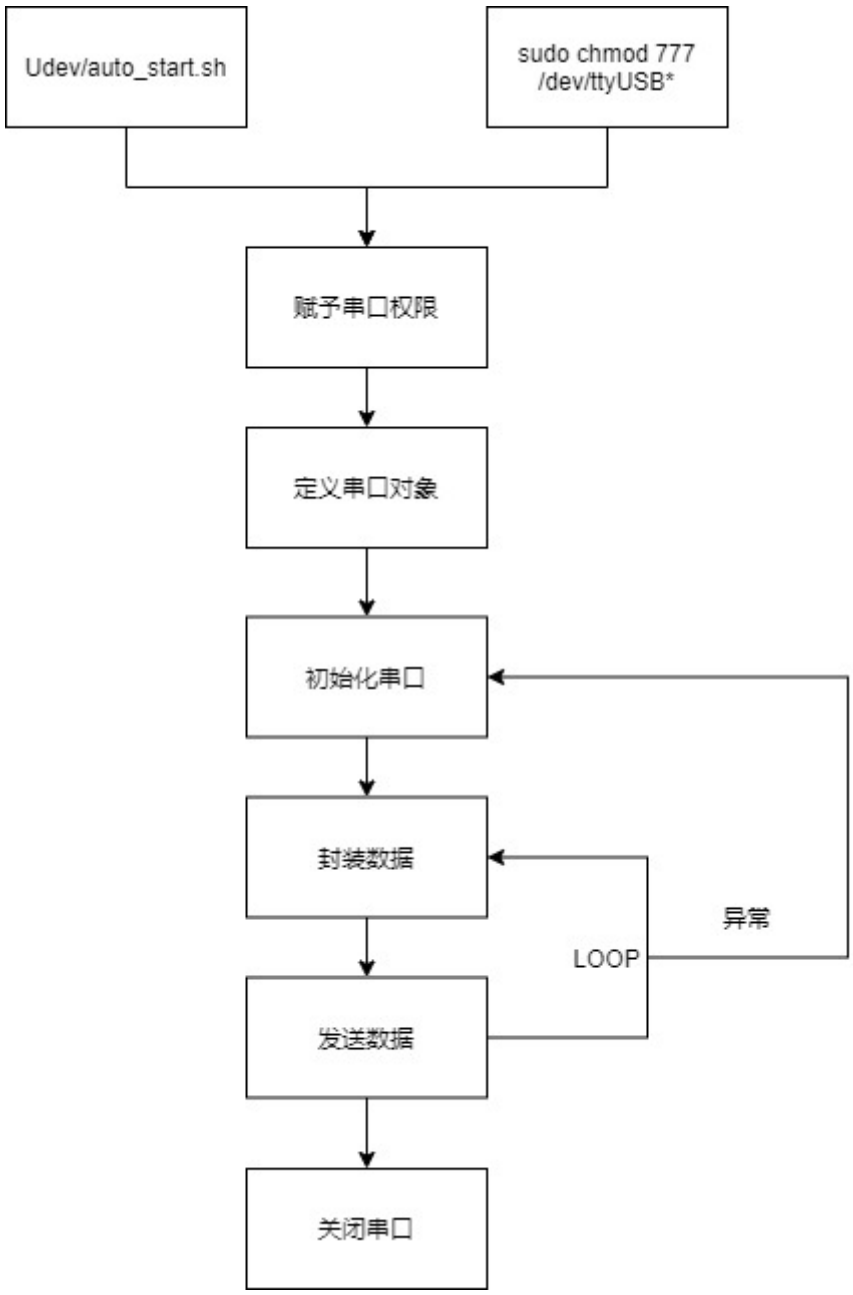
1. 生成Serialport类的一个对象，例如： `Serialport sp("/dev/ttyUSB0")`，一般建议在 循环外面新建一个对象。
2. 调用 `void initSerialPort()` 函数初始化串口
3. 在每次执行识别图片的程序之前，需要调用 `void get_Mode(int &mode...)` 函数来获取电控发来的信号，根据 mode 的值来执行不同的功能

4. 主函数获得目标坐标(或pitch、yaw)和距离后，使用 `TransformData(const VisionData &data)` 函数 来封装数据
5. 调用 `void send()` 函数来发送数据，目前与电控协商是无论是否识别到目标都需要发送数据给电控。
6. 调用 `void closePort()` 函数来关闭串口。

5. 硬件基础----TTL转USB（串口）

- 为什么要进行转换？因为单片机通信接口的电平逻辑和PC机通信接口的电平逻辑不同。
- 视觉，电控有发信息时对应的是哪一盏灯亮要清楚，方便定位问题所在

串口流程图



出现问题时基本可以按流程图的几个步骤去找，常见问题如下

- 报错：找不到`dev/ttyUSB0`文件，但是`ls /dev/ttyUSB*`有这东西，那就说明是权限问题

```
sudo chmod 777/dev/ttyUSB0
```

- 串口名字不确定：具体见`Udev规则`
- 确保通讯流程正确,看上面的流程图，看看有没有漏掉某一步
- 另外就是一些前辈的经验，目前没遇到过

> 1. **通信协议出现问题**：由于不同车的协议可能在调的过程中发生变化，所以可能导致传不过去，这时候可以**找到以前版本的通信去测试，同时可以看传输函数里面的变量类型是否正常或者看看有没有加传输函数send。**
> 2. **电控那边的串口线的两个位置接反了**：适用于导致偶尔能收到一次信息，其他时候都收不到的情况。
> 3. **串口未初始化：**（有一次是跑自己程序传输数据不成功，跑别的程序再跑自己的程序就ok，然后找到这个问题）