ART ROBOT

全国大学生智能汽车竞赛

- UART协议介绍

主讲人： 张宇轩

目录

1 串口通信协议简介

2 通信转接板协议指导

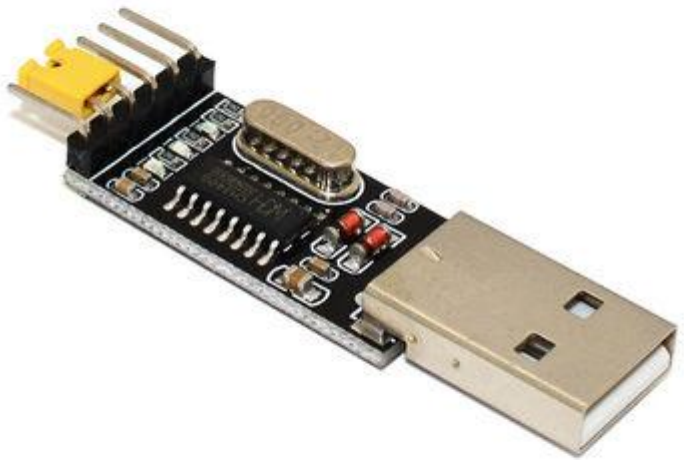
3 udev串口配置

PART ONE

串口通信协议简介

简介

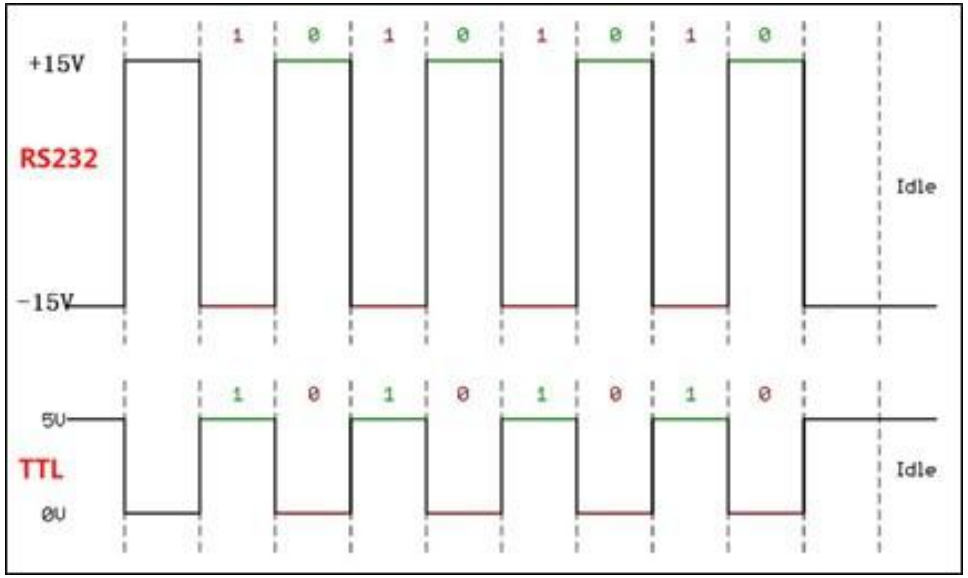
串口通信指串口按位（bit）发送和接收字节。尽管比特字节（byte）的串行通信慢，但是串口可以在使用一根线发送数据的同时用另一根线接收数据。



电平标准

根据通讯使用的电平标准不同，串口通讯可分为TTL标准及RS-232标准

通讯标准	电平标准(发送端)
5V TTL	逻辑1：2.4V-5V 逻辑0：0~0.5V
RS-232	逻辑1：-15V~-3V 逻辑0：+3V~+15V



波特率

串口异步通讯，异步通讯中由于没有时钟信号，所以两个通讯设备之间需要约定好波特率，即每个码元的长度，以便对信号进行解码。常见的波特率为4800、9600、38400、115200等。

通讯的起始和停止信号

串口通讯的一个数据包从起始信号开始，直到停止信号结束。数据包的起始信号由一个逻辑0的数据位表示，而数据包的停止信号可由0.5、1、1.5或2个逻辑1的数据位表示，只要双方约定一致即可。

有效数据

在数据包的起始位之后紧接着的就是要传输的主体数据内容，也称为有效数据，有效数据的长度常被约定为5、6、7或8位长。

数据校验

在有效数据之后，有一个可选的数据校验位。由于数据通信相对更容易受到外部干扰导致传输数据出现偏差，可以在传输过程加上校验位来解决这个问题。校验方法有奇校验(odd)、偶校验(even)、0校验(space)、1校验(mark)以及无校验(noparity)，它们介绍如下：

奇校验要求有效数据和校验位中"1"的个数为奇数，比如一个8位长的有效数据为：01101001，此时总共有4个"1"，为达到奇校验效果，校验位为"1"，最后传输的数据将是8位的有效数据加上1位的校验位总共9位。

偶校验与奇校验要求刚好相反，要求帧数据和校验位中"1"的个数为偶数，比如数据帧：11001010，此时数据帧"1"的个数为4个，所以偶校验位为"0"。

0校验是不管有效数据中的内容是什么，校验位总为"0"，1校验是校验位总为"1"。

在无校验的情况下，数据包中不包含校验位。

PART TWO

通信转接板协议指导

用户层串口通信协议

简单的来说用户层的串口通信协议就是如何定义一个数据包格式，发送端按照规定的数据包格式发送出去，接收端按照规定的数据包格式解析出正确的数据。

帧头

标记一帧数据开始传输。

数据长度

对于长度不变的数据包，数据长度应该事先约定。这样接收方在知道接受长度之后，就能够判断接收的数据包是否结束。而对于长度变换的数据包，则可以通过帧尾和制定数据长度来固定，而且数据长度存在则必须放到数据的前面，这样更方便我们解析不定长数据。

数据

需要进行传输的内容。

校验

需要判断传输的数据是否正确，常采用CRC校验、和校验、奇偶校验、异或校验等。

帧尾

标记一帧数据传输结束。非定长的数据必须有帧尾。

用户层串口通信协议

帧头	数据长度	数据						校验	帧尾
0x0A	0x03	0x01	0x02	0x03				奇偶校验	0x0D
	0x05	0x01	0x02	0x03	0x04	0x05		和校验	
	…	…						CRC校验	
								…	

用户层串口通信协议—单片机

- 设置内部时钟
- 单片机SCI初始化配置
- 编写用户层串口通信协议
  - 判断是否接收到表头
  - 处理数据长度
  - 处理传输的数据
  - 校验数据正确性
  - 判断接收到帧尾
  - 对电机和电调发出控制信号

在ROS上，我们可以通过C++或者Python其中任意一种语言进行编写。

```
int art_racecar_init(int speed,char *dev)
{
        if(Open_Serial_Dev(dev) == -1)
        {
                return -1;
        }
        else
        {
                if(fcntl(fd,F_SETFL,FNDELAY) < 0)
                {
                        printf("fcntl failed\n");
                }
                else
                {
                        printf("fcntl=%d\n",fcntl(fd,F_SETFL,FNDELAY));
                }
                set_opt(fd,speed,8,'N',1);
        }
}
```

- C++

```
int main(int argc, char** argv)
{
        char data[] = "/dev/ttyUSB0";
        if(art_racecar_init(38400,data) < 0)
        {
                printf("can't find %s\n",data);
                return 0;
        }
        setTimer(0,50000);//20Hz
        while(1)
    {
    }
        return 0;
}
```

用户层串口通信协议--ROS

- Python

```
if __name__=="__main__":
    vel = 1500
    angle = 1500
    lib_path = os.path.abspath(os.path.join(os.getcwd(), "..")) + "/lib"+ "/libart_driver.so"
    so =  cdll.LoadLibrary
    lib = so(lib_path)

try:
        car = "/dev/ttyUSB0"
        if(lib.art_racecar_init(38400,car) < 0):
            raise
            pass
        timer = threading.Timer(0.05, fun_timer)
        timer.start()
        while(1):
            pass
    except:
        print "error"
    finally:
        print "finally"
```

PART THREE

udev串口配置

udev简介与规则

udev 是 Linux 内核的设备管理器，它取代了 devfs 和 hotplug，负责管理 /dev 中的设备节点。同时，udev 也处理所有用户空间发生的硬件添加、删除事件，以及某些特定设备所需的固件加载。与传统的顺序加载相比，udev 通过并行加载内核模块提供了潜在的性能优势。异步加载模块的方式也有一个天生的缺点：无法保证每次加载模块的顺序，如果机器具有多个块设备，那么它们的设备节点可能随机变化。

- udev 规则的所有操作符
- udev 规则的匹配键
- udev 的重要赋值键
- udev 的值和可调用的替换操作符

底盘驱动板模块的car.rules文件

KERNEL=="ttyUSB*", ATTRS{idVendor}=="1a86", ATTRS{idProduct}=="7523", MODE:="666", SYMLINK+="car"

lsusb

Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 001 Device 004: ID 1a86:7523 QinHeng Electronics HL-340 USB-Serial adapter
Bus 001 Device 003: ID 05e3:0608 Genesys Logic, Inc. Hub
Bus 001 Device 002: ID 05e3:0608 Genesys Logic, Inc. Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub

底盘驱动板模块检查

ls -l /dev |grep ttyUSB

lrwxrwxrwx 1 root   root   7 Nov 29 14:03 car -> ttyUSB1
crw-rw-rw- 1 root   dialout 188,   0 Nov 29 14:03 ttyUSB0
crwxrwxrwx 1 root   dialout 188,   1 Nov 29 14:03 ttyUSB1

谢　　谢