航天新一代移动机器人 YQ01 手持遥控器开发教程

AI 航 团队

航天新一代移动机器人 YQ01 支持自动模式和手动模式，在手动模式下可通过遥控器对车体运动进行控制。遥控器采用的是索尼的 PlayStation2 游戏机手柄，简称 PS2。该手柄用于索尼的 psx 系列游戏机，由于其实用性及稳定性被广泛应用在其他器件上使用，通过将 PS2 手柄的通讯协议进行破解，便可以与其他器件配合使用，性价比极高，并且按键丰富，YQ01 将 PS2 与底层驱动板进行开发，完成小车的手持遥控功能。

图 1 PS2 遥控器

一、ps2 手柄硬件介绍：

ps2 由手柄与接收器两部分组成，手柄主要负责发送按键信息。都接通电源并打开手柄开关时，手柄与接收器自动配对连接，在未配对成功的状态下，接收器绿灯闪烁，手柄上的灯也会闪烁，配对成功后，接收器上绿灯常亮，手柄上灯也常亮。

图 2 PS2 手柄及接收器

接收器将与小车驱动板的 stm32 单片机相连，从而实现了手柄与小车驱动板的通信。接收器引脚编号如下：

图 3 接收器引脚标号

表 1 接收器引脚功能表

1	2	3	4	5	6	7	8	9
DI/DAT	DO/CMD	NC	GND	VDD	CS/SEL	CLK	NC	ACK

1. DI/DAT：信号流向，从手柄到主机，此信号是一个 8bit 的串行数据，同步传送于时钟的下降沿。信号的读取在时钟由高到低的变化过程中完成。
2. DO/CMD：信号流向，从主机到手柄，此信号和 DI 相对，信号是一个 8bit 的串行数据，同步传送于时钟的下降沿。
3. NC：空端口；
4. GND：电源地；
5. VDD：接收器工作电源，电源范围 3~5V；
6. CS/SEL：用于提供手柄触发信号。在通讯期间，处于低电平；
7. CLK：时钟信号，由主机发出，用于保持数据同步；
8. NC：空端口；
9. ACK：从手柄到主机的应答信号。此信号在每个 8bits 数据发送的最后一个周期变低并且 CS 一直保持低电平，如果 CS 信号不变低，约 60 微秒 PS 主机会试另一个外设。在编程时未使用 ACK 端口。

   接收器连接到小车驱动板的 P4 针脚，P4 引脚标号 1～9 对应接收器 1～9 如下图：

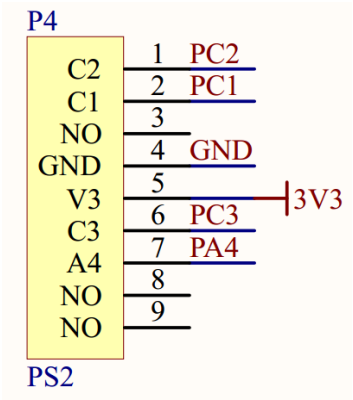

图 4 YQ01 驱动板与接收器接口

二、手柄的使用、连接配对说明

PS2 手柄由手柄和接收器两个部分组成，手柄需要两节 7 号 1.5V 供电，接收器的电源

和控制器使用同一电源，电源范围为 3~5V,不能接反，不能超电压，过压和反接，都会使接收器烧坏。

手柄上有个电源开关，ON 开/OFF 关，将手柄开关打到 ON 上，在未搜索到接收器的状况下，手柄上的灯会不停的闪，在一定时间内，还未搜索到接收器，手柄将进入待机模式，手柄上的灯将灭掉，这时，按下“START”键，唤醒手柄。

接收器供电，在未配对的状况在，绿灯闪。

手柄打开，接收器供电，手柄和接收器会自动配对，这时灯常亮，手柄配对成功。按键“MODE”(手柄批次不同，上面的标识有可能是"ANALOG"，但不会影响使用)，可以选择“红灯模式”、“绿灯模式”。

当主机想读手柄数据时，将会拉低 CS 线电平，并发出一个命令“0x01”；手柄会回复它的 ID“0x41=模拟绿灯，0x73=模拟红灯”；在手柄发送 ID 的同时，主机将传送 0x42，请求数据；随后手柄发送出 0x5A，告诉主机“数据来了”。

idle：数据线空闲，该数据线无数据传送。

一个通讯周期有 9 个字节（8 位），这些数据是依次传送的。

表 2 传输数据---手柄按键对照表

顺序	DO	DI	Bit0、Bit1、Bit2、Bit3、Bit4、Bit5、Bit6、Bit7、
0	0X01	idle	
1	0x42	ID	
2	idle	0x5A	
3	idle	data	SELECT、L3、R3、START、UP、RIGHT、DOWN、LEFT
4	idle	data	L2、R2、L1、R1、△、○、╳、□
5	idle	data	PSS_RX（0x00=left、0xFF=right）
6	idle	data	PSS_RY（0x00=up、0xFF=down）
7	idle	data	PSS_LX（0x00=left、0xFF=right）
8	idle	data	PSS_LY（0x00=up、0xFF=down）

当有按键按下，对应位为“0”，其他位为“1”，例如当键“SELECT”被按下时，Data[3]=11111110B，

红灯模式时：左右摇杆发送模拟值，0x00~0xFF 之间，且摇杆按下的键值 L3、R3 有效；

绿灯模式时：左右摇杆模拟值为无效，推到极限时，对应发送 UP、RIGHT、DOWN、LEFT、△、○、╳、□，按键 L3、R3 无效。

图 5 手柄各按键位置

三、软件编程说明

下面将对照以上协议介绍 YQ01 遥控器程序：

首先是 pstwo.h 文件，对相关变量和函数进行声明

```
//定义输入输出状态
#define DI     PCin(2)                 //   输入

#define DO_H PCout(1)=1              //命令位高
#define DO_L PCout(1)=0              //命令位低
#define CS_H PCout(3)=1              //CS 拉高
#define CS_L PCout(3)=0              //CS 拉低
#define CLK_H PAout(4)=1             //时钟拉高
#define CLK_L PAout(4)=0             //时钟拉低

//定义表 2 中数据意义的对照表
#define PSB_SELECT        1
#define PSB_L3            2
#define PSB_R3            3
#define PSB_START         4
#define PSB_PAD_UP        5
#define PSB_PAD_RIGHT     6
#define PSB_PAD_DOWN      7
#define PSB_PAD_LEFT      8
#define PSB_L2            9
#define PSB_R2            10
#define PSB_L1            11
#define PSB_R1            12
#define PSB_GREEN         13
#define PSB_RED           14
#define PSB_BLUE          15
#define PSB_PINK          16

//定义模拟量索引，用作 Data[]数据索引
#define PSS_RX 5                      //右摇杆 X 轴模拟量数据
#define PSS_RY 6
#define PSS_LX 7
#define PSS_LY 8

//声明相关函数
void PS2_Init(void);            //初始化
u8 PS2_RedLight(void);          //判断是否为红灯模式
void PS2_ReadData(void);        //读手柄数据
void PS2_Cmd(u8 CMD);           //向手柄发送命令
u8 PS2_DataKey(void);           //按键值读取
```

```
u8 PS2_AnologData(u8 button); //得到一个摇杆的模拟量
void PS2_ClearData(void);      //清除数据缓冲区
```

接下来是 pstwo.c 文件，定义相关函数和初始化等，下面介绍关键代码：

```
#define DELAY_TIME    delay_us(5);
u16 Handkey; //  按键值读取，零时存储。
u8 Comd[2]={0x01,0x42};   //开始命令。请求数据
u8 Data[9]={0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00}; //数据存储数组，对应数据意义对照表
u16 MASK[]={
    PSB_SELECT,
    PSB_L3,
    PSB_R3 ,
    PSB_START,
    PSB_PAD_UP,
    PSB_PAD_RIGHT,
    PSB_PAD_DOWN,
    PSB_PAD_LEFT,
    PSB_L2,
    PSB_R2,
    PSB_L1,
    PSB_R1 ,
    PSB_GREEN,
    PSB_RED,
    PSB_BLUE,
    PSB_PINK
    };    //按键值与按键明

//向手柄发送命令
void PS2_Cmd(u8 CMD)
{
    volatile u16 ref=0x01;
    Data[1] = 0;
    for(ref=0x01;ref<0x0100;ref<<=1)
// 相当于 0001  左移直到为 0001 0000 0000  期间左移一共 8 次，对应发送数据 CMD 的 8 位
    {
        if(ref&CMD)
        {
            DO_H;                          //输出一位控制位  1
        }
        else DO_L;                         //命令位低  0
```

```
            CLK_H;                          //时钟拉高，时钟信号有主机发出，数据同步传输于时钟下降沿，
            DELAY_TIME;                     //delay_us(5)
            CLK_L;                          //时钟拉低，此处的作用就是产生下降沿
            DELAY_TIME;
            CLK_H;
            if(DI)                          //如果有输入，将其存入 Data[1]中，用来存放手柄回复的 ID，判断红灯、绿灯模式
                Data[1] = ref|Data[1];
        }
        delay_us(16);
}

//判断是否为红灯模式,0x41=模拟绿灯，0x73=模拟红灯
//返回值；0，红灯模式 其他，其他模式
u8 PS2_RedLight(void)
{
        CS_L;                   //拉低 cs 信号
        PS2_Cmd(Comd[0]);   //开始命令，主机发送 0x01
        PS2_Cmd(Comd[1]);   //请求数据，主机发送 0x42
        CS_H;                   //拉低 cs 信号
        if( Data[1] == 0X73)     return 0 ;
        else return 1;
}

//读取手柄数据
void PS2_ReadData(void)
{
        volatile u8 byte=0;
        volatile u16 ref=0x01;
        CS_L;
        PS2_Cmd(Comd[0]);   //开始命令  对应传输表第 0 个字节
        PS2_Cmd(Comd[1]);   //请求数据  对应传输表第 1 个字节
        for(byte=2;byte<9;byte++)       //开始接受数据，接收传输数据第 2--8 字节的数据
        {
            for(ref=0x01;ref<0x100;ref<<=1)
            {
                CLK_H;
                DELAY_TIME;
                CLK_L;
                DELAY_TIME;
                CLK_H;
                  if(DI)
                  Data[byte] = ref|Data[byte];   //接收数据传放到对应的存储区
```

```
            }
            delay_us(16);
        }
        CS_H;
}

//对读出来的 PS2 的数据进行处理,只处理按键部分
//只有一个按键按下时按下为 0， 未按下为 1
u8 PS2_DataKey()
{
        u8 index;

        PS2_ClearData();      //清除数据缓冲区
        PS2_ReadData();       //读取手柄数据

        Handkey=(Data[4]<<8)|Data[3];     //16 个按键，按下为 0，未按下为 1，与 MASK[]顺序相对应
        for(index=0;index<16;index++)
        {
            if((Handkey&(1<<(MASK[index]-1)))==0)   //将 1 左移 index 位，通过&位与指令，如果有按键按下，结果会=0
            return index+1;   //返回的索引号+1 正好对应 MASK[]数组的内容
        }
        return 0;               //没有任何按键按下
}

//得到一个摇杆的模拟量       范围 0~256
u8 PS2_AnologData(u8 button)   //对应传输数据数组  5 6 7 8
{
        return Data[button];
}

//清除数据缓冲区
void PS2_ClearData()
{
        u8 a;
        for(a=0;a<9;a++)
            Data[a]=0x00;
}
```

2020 年 3 月