GL-MOTO-Mini

----用户说明书

深圳市广联智通科技有限公司

公司网站：www.gl-inet.com

电话：086-0755-86606126

技术论坛：bbs.gl-inet.com

版本：GL-MOTO-Mini 1.0

前言

致用户：

欢迎使用 GL-MOTO-Mini！我们非常高兴您选择了本款产品。我们将为您提供最真诚最优质的服务，让您在以后的日子里尽情发挥您的创意！为了使您的产品功能得到充分发挥，我们建议在连接和操作之前，通读一遍说明书，请务必了解本产品各功能模块、开关和接口等的功能和设置方法后再使用，这样有便于您掌握系统的连接方法和使用要点，有助于您更好的使用本款产品！

我们对用户使用手册的编排力求全面且简单易懂，目的是您可以获取与您购买的 GL-MOTO-Mini 相关的软件安装、基本操作、软硬件使用方法等知识，但为了提高产品的性能，我们会对产品的硬件和软件做些改动和升级，这样可能会产生软硬件配置和本手册在某些细节上不符，请以最新软件和您购买的 GL-MOTO-Mini 实际配置为准。本手册的更改或升级不另行通知客户！在编手册时我们难免会有疏漏甚至错误之处，请您多加包涵并热烈欢迎指正，本公司将不为本手册可能产生的疏漏和错误负责！

如果您在学习过程中遇到问题，可到本公司的专业技术论坛（bbs.gl-inet.com）发帖提问并和众多用户一起学习交流，论坛提供大量以往用户常见问题和学习经验供你分享。

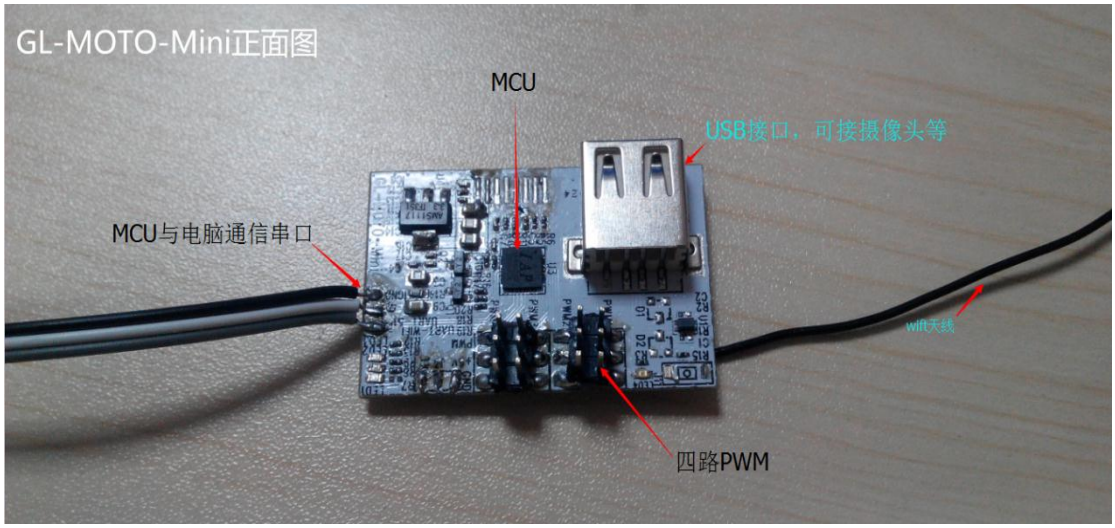
深圳市广联智通科技有限公司

声明：

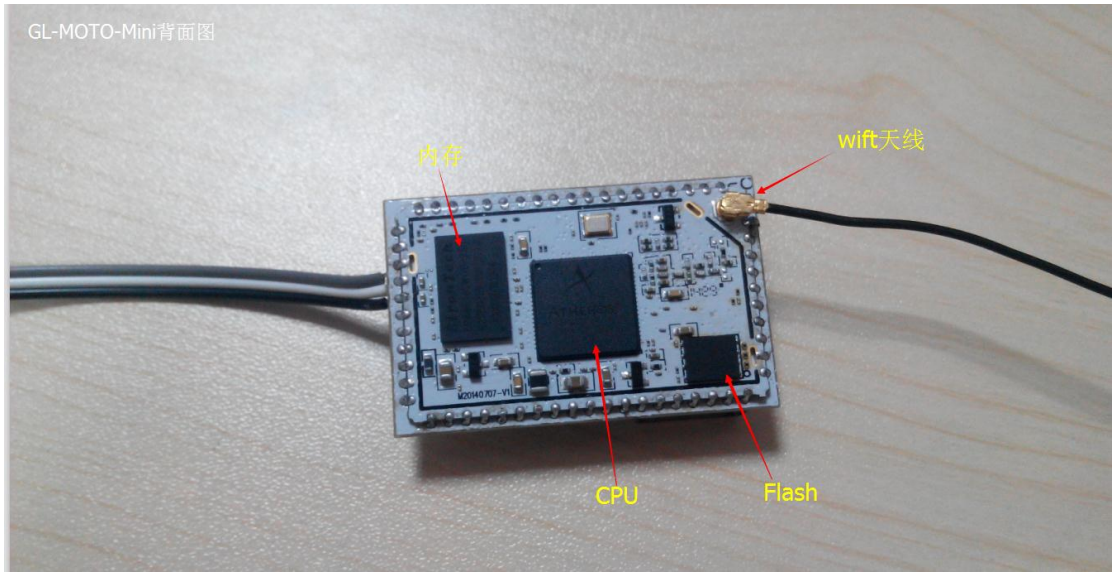
目录

第一章：GL-MOTO-Mini 硬件说明

1.1 实物硬件说明

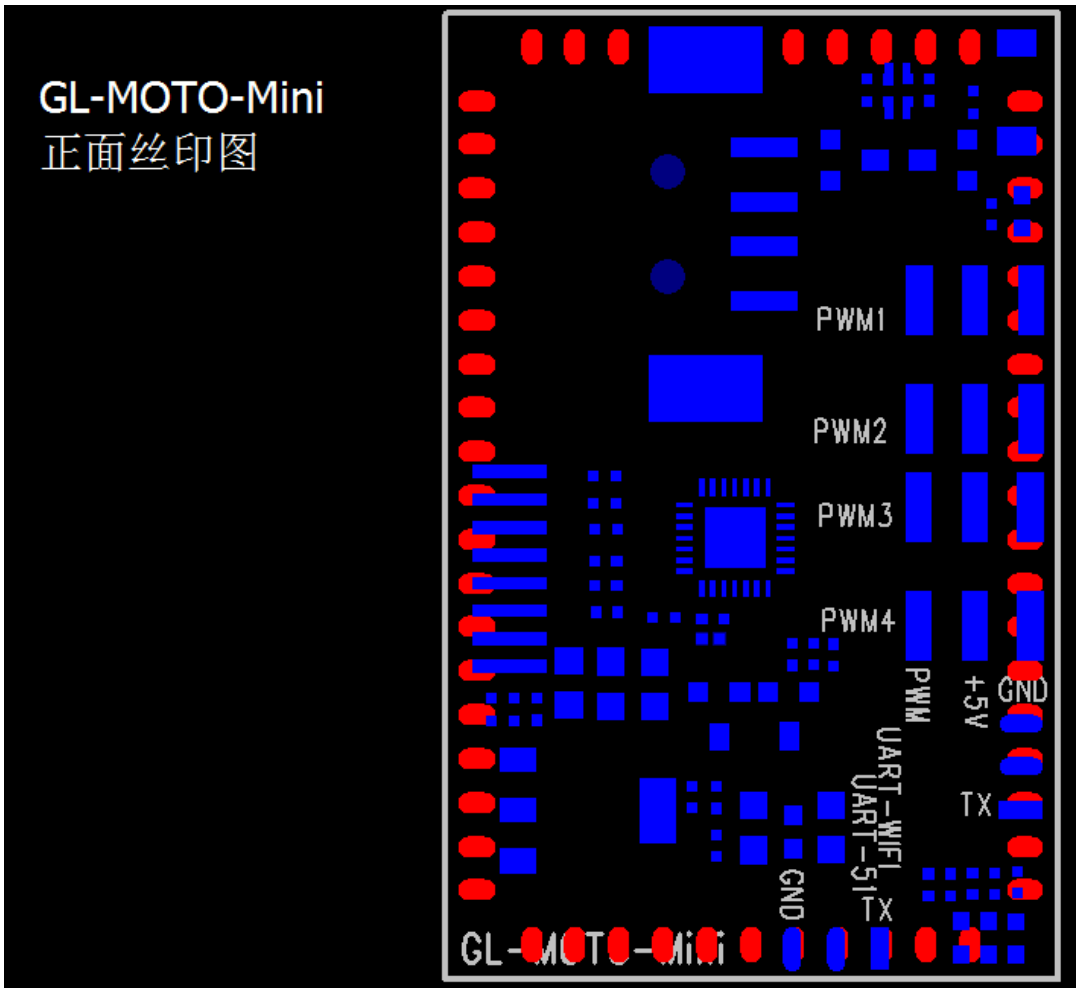


GL-MOTO-Mini 正面图



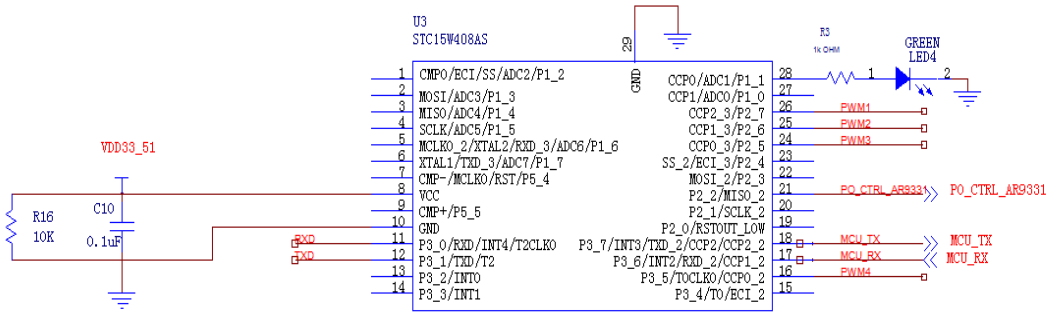
GL-MOTO-Mini 背面图

1.2 GL-MOTO-Mini 正面丝印图



GL-MOTO-Mini 正面丝印图

1.3 MCU 引脚说明

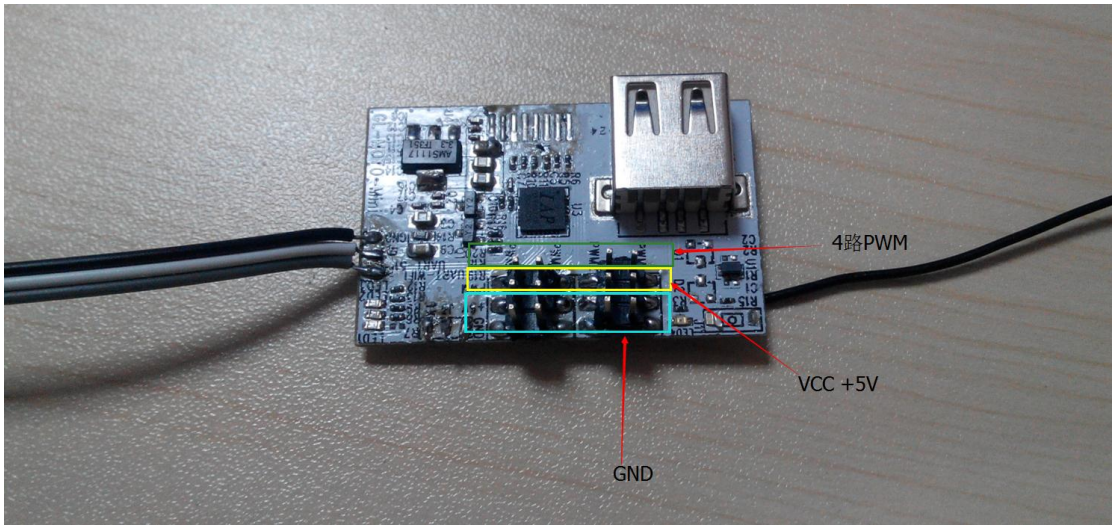


MCU 引脚接线图

附：更多内容可查看配套文件

第二章: GL-MOTO-Mini 功能说明

2.1 GL-MOTO-Mini 的 4 路 PWM

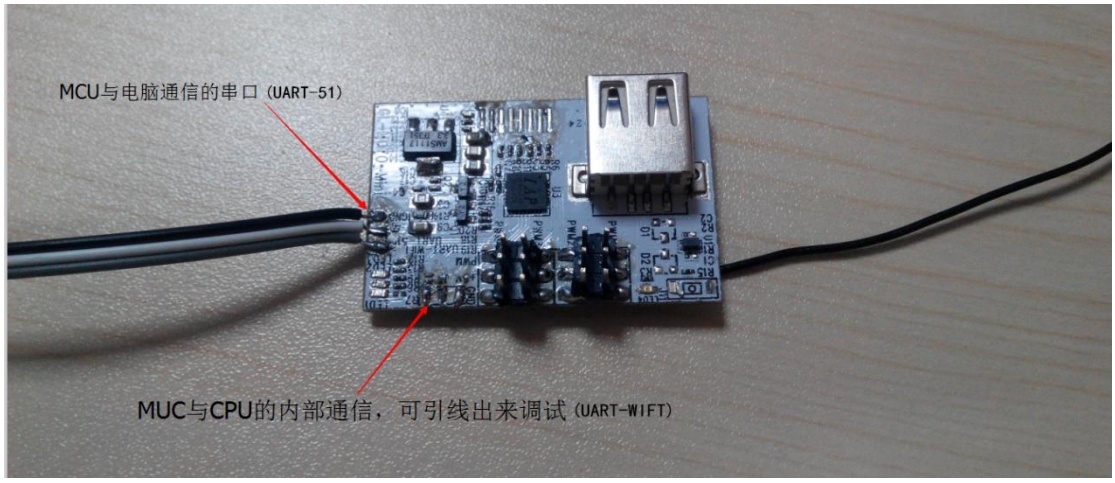


GL-MOTO-Mini 电源图

由上图我们可以知道 GL-MOTO-Mini 的电源 VCC、GND 两路总线和四路 PWM 是在一起的。可通过电调的 BEC 端来供电，也可单独引线接电源（+5V）。只需要接通其中一个 VCC 引脚和 GND 引脚就可以给 GL-MOTO-Mini 供电，因为 4 个 VCC 是连在一起的，4 个 GND 也是连在一起的，当然同时插上 4 个 BEC 也是没有问题的。

GL-MOTO-Mini 的主要功能是通过这四路 PWM 来实现控制，可用于航模、智能小车、等的舵机控制，也可以自己 DIY 用来实现控制定时给花草浇水或者定时给宠物喂养。

2.2 GL-MOTO-Mini 的通信

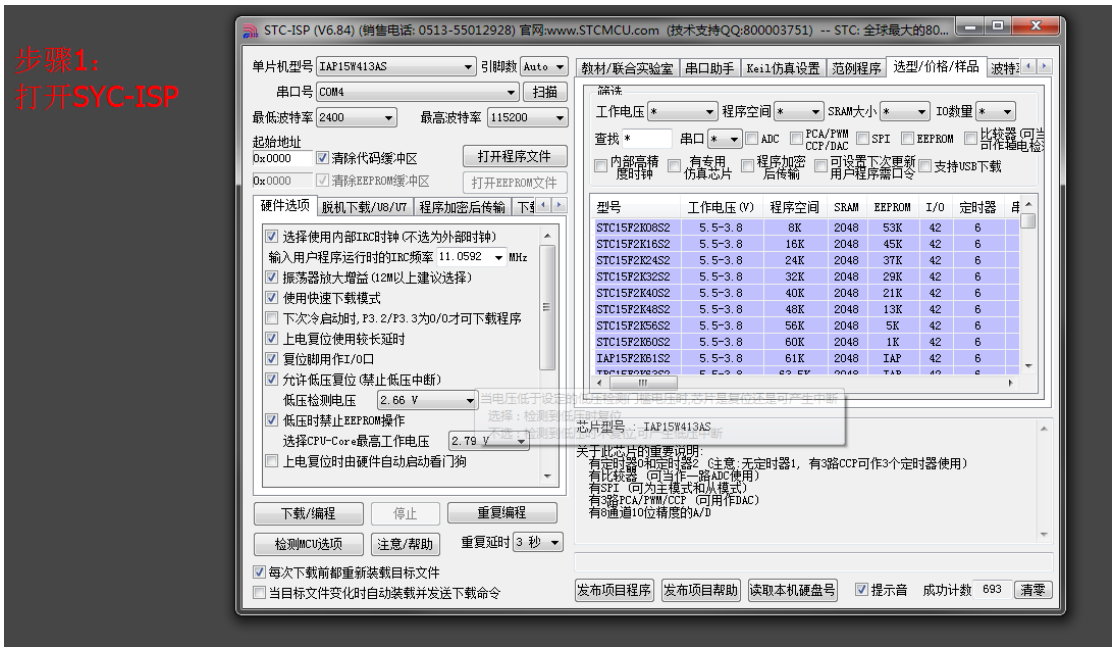


GL-MOTO-Mini 串口图

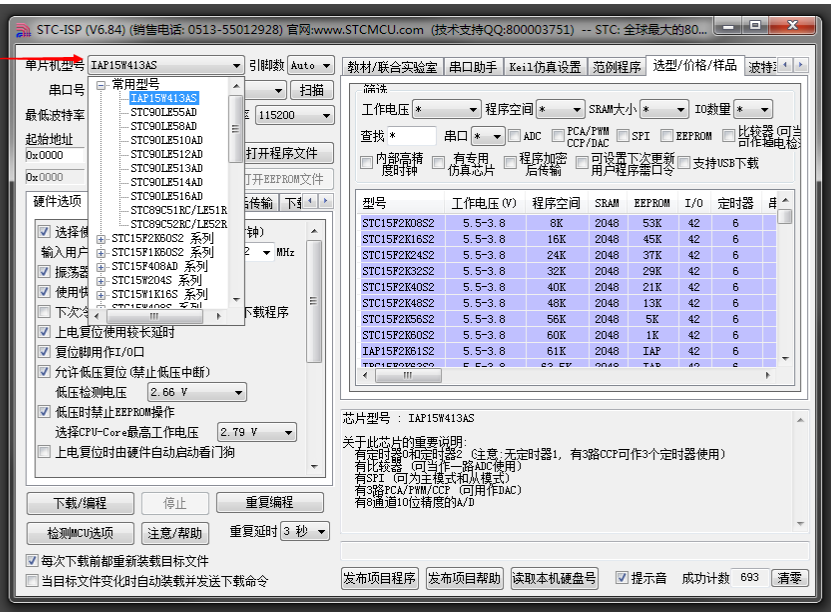
由上图我们可以知道，我们可以通过串口（UART-51）来把我们自己 DIY 想要实现某些功能的代码下载到 MCU。只需要一个 USB-STC-ISP 串口即可做到。(注：如若需要 USB-STC-ISP 串口，请另行购买。)

USB-STC-ISP 串口图片

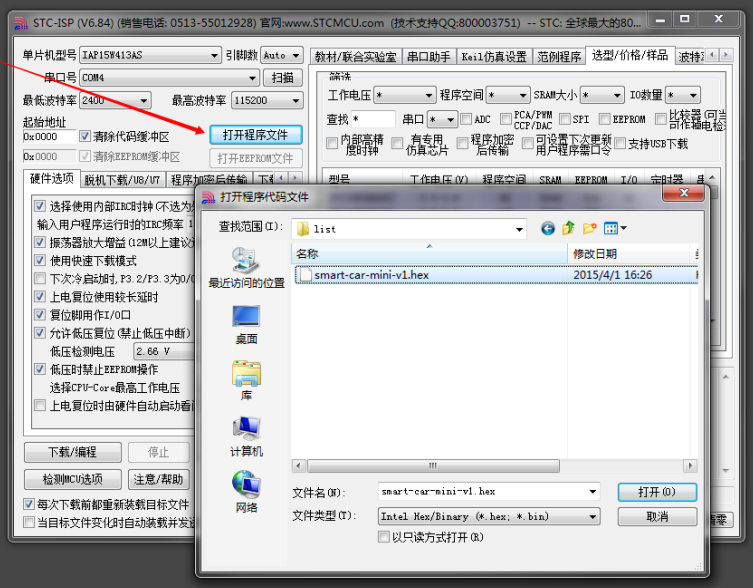
具体下载步骤如下：



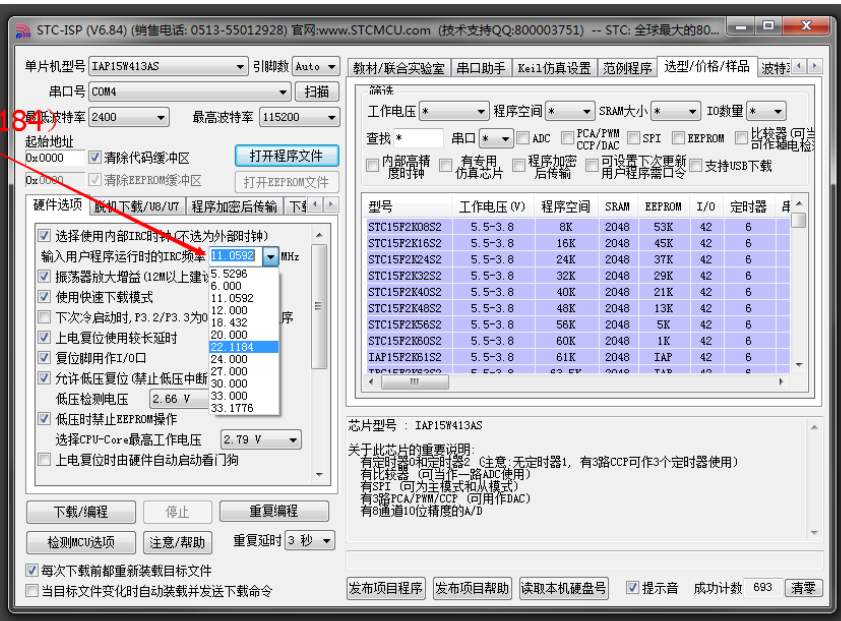
步骤2：选择对应的MCU型号



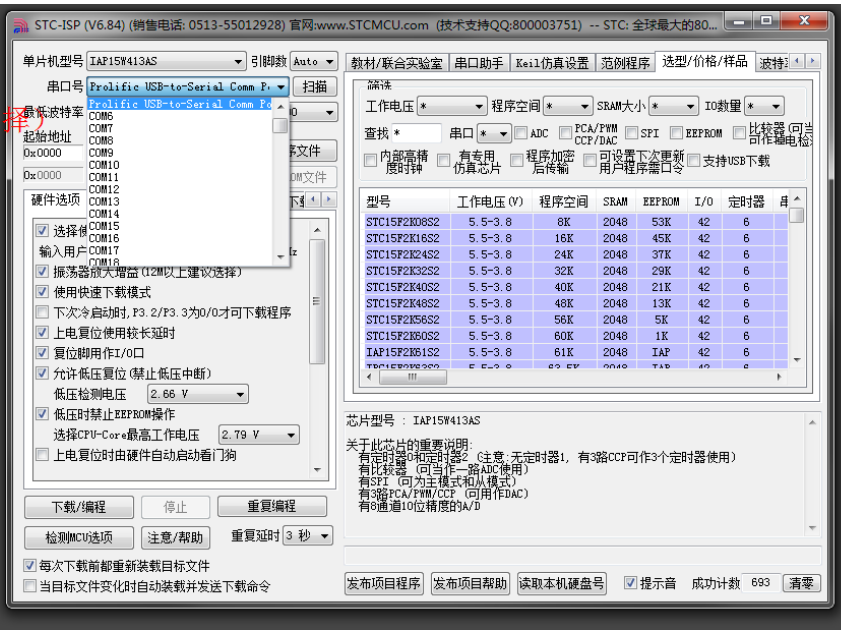
步骤3：选择你要下载到MCU的hex文件

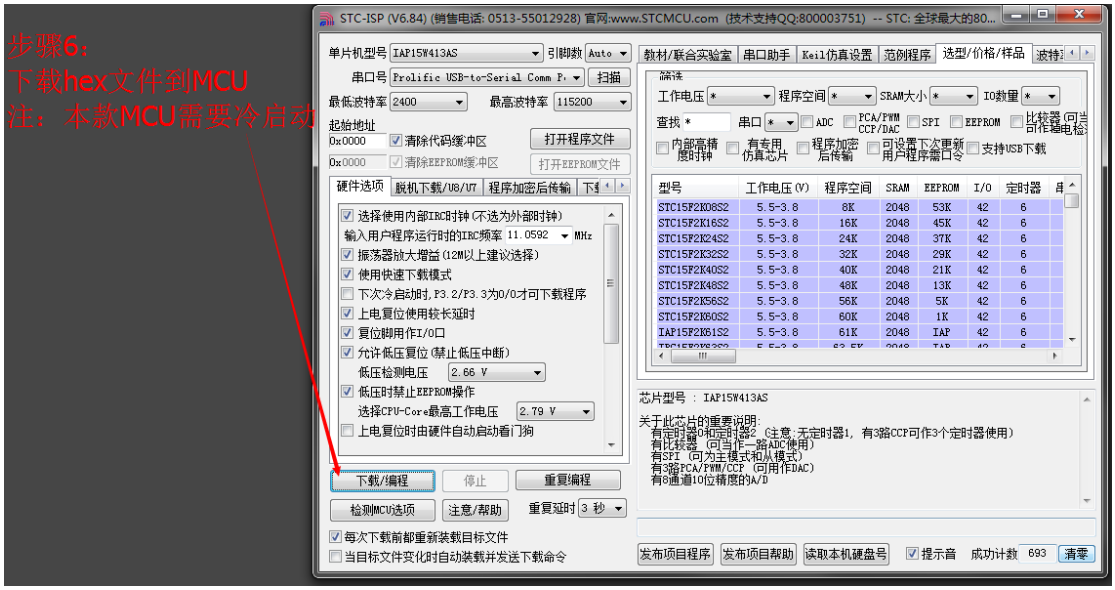


步骤4：
选择MCU需要工作的晶振频率（22.1184）



步骤5：
选择对应的串口号（一般都是自动选择）




步骤6：
下载hex文件到MCU
注：本款MCU需要冷启动
STC-ISP (V6.84) (销售电话: 0513-55012928) 官网:www.STCMCU.com (技术支持QQ:800003751) -- STC: 全球最大的80...
单片机型号 IAP15W413AS
引脚数 Auto
串口号 Prolific USB-to-Serial Comm P...
扫描
最低波特率 2400
最高波特率 115200
起始地址
0x0000
清除代码缓冲区
打开程序文件
0x0000
清除EEPROM缓冲区
打开EEPROM文件
硬件选项
脱机下载/U8/U7
程序加密后传输
选择使用内部IRC时钟(不选为外部时钟)
输入用户程序运行时的IRC频率 11.0592 MHz
振荡器放大增益(12M以上建议选择)
使用快速下载模式
下次冷启动时, P3.2/P3.3为0/0才可下载程序
上电复位使用较长延时
复位脚用作I/O口
允许低压复位(禁止低压中断)
低压检测电压 2.66 V
低压时禁止EEPROM操作
选择CPU-Core最高工作电压 2.79 V
上电复位时由硬件自动启动看门狗
下载/编程
停止
重复编程
检测MCU选项
注意/帮助
重复延时 3 秒
每次下载前都重新装载目标文件
当目标文件变化时自动装载并发送下载命令
教材/联合实验室
串口助手
Keil仿真设置
范例程序
选型/价格/样品
工作电压
程序空间
SRAM大小
IO数量
查找
串口
ADC
SPI
EEPROM
内部高精度时钟
有专用仿真芯片
程序加密后传输
可设置下次更新用户程序需口令
支持USB下载
型号 工作电压(V) 程序空间 SRAM EEPROM I/O 定时器
STC15F2K08S2 5.5-3.8 8K 2048 53K 42 6
STC15F2K16S2 5.5-3.8 16K 2048 45K 42 6
STC15F2K24S2 5.5-3.8 24K 2048 37K 42 6
STC15F2K32S2 5.5-3.8 32K 2048 29K 42 6
STC15F2K40S2 5.5-3.8 40K 2048 21K 42 6
STC15F2K48S2 5.5-3.8 48K 2048 13K 42 6
STC15F2K56S2 5.5-3.8 56K 2048 5K 42 6
STC15F2K60S2 5.5-3.8 60K 2048 1K 42 6
IAP15F2K61S2 5.5-3.8 61K 2048 IAP 42 6
芯片型号 : IAP15W413AS
关于此芯片的重要说明:
有定时器0和定时器2 (注意:无定时器1, 有3路CCP可作3个定时器使用)
有比较器 (可当作一路ADC使用)
有SPI (可为主模式和从模式)
有3路PCA/PWM/CCP (可用作DAC)
有8通道10位精度的A/D
发布项目程序
发布项目帮助
读取本机硬盘号
提示音
成功计数 693
清零

第三章：GL-MOTO-Mini 配套程序部分分析

3.1 main 函数

```
/*******************************************/
void main(void)
{

    char string_test[4]={0};

    led_init();          //LED设置为推挽输出
    led_off();           //LED关闭

    router_init();      //路由器设置为推挽输出
    router_close();     //路由器关闭
    router_open();      //路由器打开

    UART_config();      //串口初始化

    PCA_config();       //PWM初始化
    pca_pwm_init();     //PWM设置为推挽输出

    //TIME_PWM测试
    //用定时器模拟PWM的时候要先设置单片机的引脚为推挽输出，然后在配置定时器、更新占空比
    P_PWM = 0;
    P3M1 &= ~(1 << 5);  //P3.5 设置为推挽输出
    P3M0 |=  (1 << 5);

    Timer_config();

    pwm = 2768; //给PWM一个初值，这里为10%占空比
    LoadPWM(pwm);

    EA = 1;                     //使能总中断

    while (1)
    {
        delay_ms(1);
        if(++communicate_time > 500)  //每循环一次，计数加1,超过500，认为通信中断
        {
            //防止失控
            message_data[0]=1500;
            message_data[1]=1500;
            message_data[2]=1500;
            message_data[3]=1500;
            if(communicate_time==60000)//失去控制指示
            {
                led_on();//指示灯亮
            }
            else
            {
                led_off();//指示灯灭
            }
        }
        get_message();  //接收信息
```

```
        /*
        *   PWM换算公式：
        *
        *   924+3.68*(1000-1000)          924             0.5
        * ------------------------   = -------------  = ---------
        *           36874               36874             20
        *
        *   924+3.68*(1500-1000)          2764            1.5
        * ------------------------   = -------------  = ---------
        *           36874               36874             20
        *
        *   924+3.68*(2000-1000)          4604            2.5
        * ------------------------   = -------------  = ---------
        *           36874               36874             20
        *
        */
        //message_data[0]对应的是方向盘数据,message_data[1]对应的是油门数据。

        //pwm2对应的是GL-MOTO-Mini板上的PWM1,对应定时器的pwm2，对应P2.7引脚
        //PWM1的量程全程
        pwm2 = 924+3.68*(message_data[1]-1000);//20ms的占空比0.5~2.5
        if(pwm2 >= PWM_HIGH_MAX_PCA)    pwm2 = PWM_HIGH_MIN_PCA;
        PWMn_Update(PCA2,pwm2);

        //PWM1的量程减半
        //pwm2 = 1843+1.85*(message_data[1]-1000);//20ms的占空比1~2
        //if(pwm2 >= PWM_HIGH_MAX_PCA)  pwm2 = PWM_HIGH_MIN_PCA;
        //PWMn_Update(PCA2,pwm2);

        //对应的是GL-MOTO-Mini板上的PWM2,对应定时器的pwm1，对应P2.6引脚
        //PWM1的量程全程
        pwm1 = 924+3.68*(2000-(message_data[0]));   //20ms的占空比0.5~2.5
        if(pwm1 >= PWM_HIGH_MAX_PCA)    pwm1 = PWM_HIGH_MIN_PCA;
        PWMn_Update(PCA1,pwm1);

        //PWM2的量程减半
        //pwm1 = 1843+1.85*(message_data[0]-1000);//20ms的占空比1~2
        //if(pwm1 >= PWM_HIGH_MAX_PCA)  pwm1 = PWM_HIGH_MIN_PCA;
        //PWMn_Update(PCA1,pwm1);

        //对应的是GL-MOTO-Mini板上的PWM3，对应定时器的pwm0，对应P2.5引脚
        //PWM3的量程全程
        pwm0 = 924+3.68*(message_data[0]-1000);//20ms的占空比0.5~2.5
        if(pwm0 >= PWM_HIGH_MAX_PCA)    pwm0 = PWM_HIGH_MIN_PCA;
        PWMn_Update(PCA0,pwm0);

        //PWM3的量程减半
        //pwm0 = 1843+1.85*(message_data[0]-1000);//20ms的占空比1~2
        //if(pwm0 >= PWM_HIGH_MAX_PCA)  pwm0 = PWM_HIGH_MIN_PCA;
        //PWMn_Update(PCA0,pwm0);


        /*
        *   定时器模拟PWM，换算公式与上面的一样：
        *
        *   924*(1000-1000)          924             0.5
        * -------------------- = -------------  = ---------
        *        368743            36874             20
        *
        *
        *   924*(1500-1000)          2764            1.5
        * -------------------- = -------------  = ---------
        *        36874             36874             20
        *
        *
        *   924*(2000-1000)          4604            2.5
        * -------------------- = -------------  = ---------
        *        36874             36874             20
        */
        pwm = 924+3.68*(message_data[0]-1000);//20ms的占空比1~2
        if(pwm >= PWM_HIGH_MAX_TIME)    pwm = PWM_HIGH_MAX_TIME;
        LoadPWM(pwm);

        //pwm = 1843+1.85*(message_data[0]-1000);//20ms的占空比1~2
        //if(pwm >= PWM_HIGH_MAX_TIME)  pwm = PWM_HIGH_MAX_TIME;
        //LoadPWM(pwm);
    }
}
```

进入 main 函数，将路由器初始化并打开，初始化串口设置用串口 1 发送信息，初始化 PWM，初始化定时器设置使用 Timer0 工作。

进入 while 循环里面，判断手机客户端和 GL-MOTO-Mini 是否有在进行通信，进入接收信息函数（get_message()）,客户端与 MCU 进行交互改变 PWM。客户端通过发送出的“前进”、“后退”、“左转”、“右转”这四个内容经过 wift 发送到 CPU 上，CPU 再跟 MCU 进行通信，最后达到控制舵机的目的。

3.2 PWM 换算公式讲解

```
/*
*   PWM换算公式：
*
*   924+3.68*(1000-1000)          924            0.5
* ------------------------   = -------------  = ---------
*           36874               36874            20
*
*   924+3.68*(1500-1000)          2764           1.5
* ------------------------   = -------------  = ---------
*           36874               36874            20
*
*   924+3.68*(2000-1000)          4604           2.5
* ------------------------   = -------------  = ---------
*           36874               36874            20
*
*/
```

我们先来看后面的换算公式：

```
     924            0.5
-------------  = ---------
   36874            20

     2764           1.5
-------------  = ---------
   36874            20

     4604           2.5
-------------  = ---------
   36874            20
```

上面三条公式都是同一个道理，我们就挑第一条来讲，后面两条以此类推。

```
    924            0.5
-------------  = ---------
   36874            20
```

0.5/20，这个是指 PWM 的占空比。

当 PWM 的占空比为 0.5/20，舵机指向 0 度角；当 PWM 的占空比为 1.5/20，舵机指向 90 度角；当 PWM 的占空比为 2.5/20，舵机指向 180 度角。

36874 是我们定义的 PWM 的周期，在配套程序的 PCA.H 这个文件可以看到。

```
#define     PWM_DUTY_PCA    36874       //定义PWM的周期，数值为PCA所选择的时钟脉冲个数。
```

924 就是我们要求的数，当我们要输出 0.5/20 的占空比的 PWM，那么我们只需要赋值 924 给 PWM 就可以。

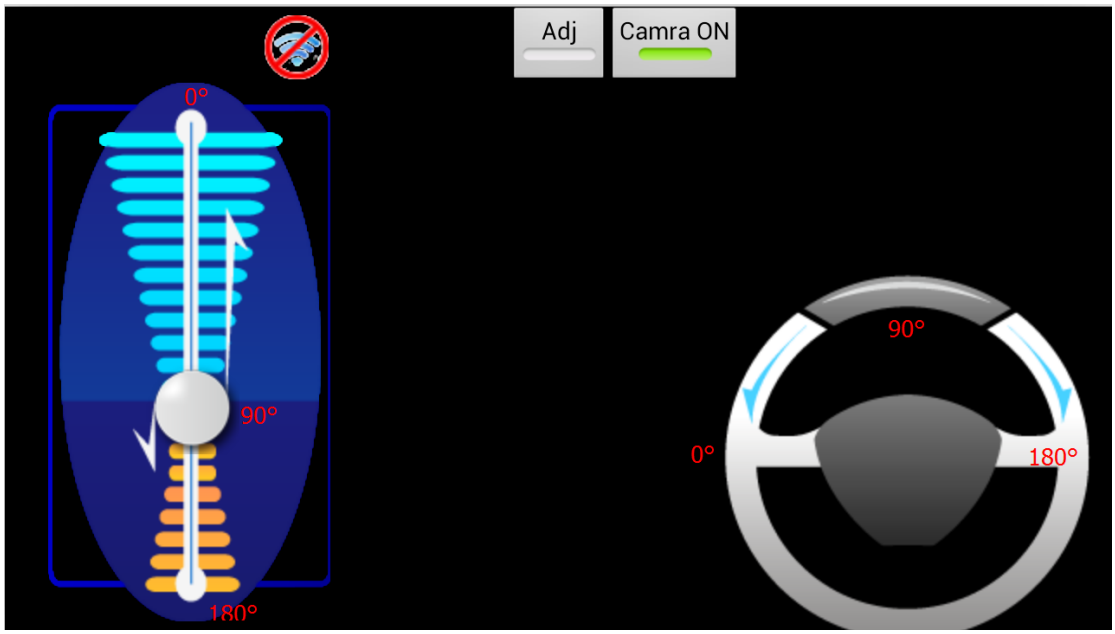
接下来讲前面等式：

$$\frac{924+3.68*(1000-1000)}{36874} = \frac{924}{36874}$$

$$\frac{924+3.68*(1500-1000)}{36874} = \frac{2764}{36874}$$

$$\frac{924+3.68*(2000-1000)}{36874} = \frac{4604}{36874}$$

这三条等式对应的是客户端上的从 0 度到 180 度。



在客户端上我们定义 1000 为 0 度，1500 为 90 度，2000 为 180 度，所以根据这条等式我们可以通过客户端来让 MCU 控制舵机达到我们需要的角度。

3.3 get_message()函数讲解

```
void get_message()      //接收信息
{
    if(COM1.RX_TimeOut > 0)         //接受数据超时计数
    {
        //串口
        if(--COM1.RX_TimeOut == 0)  //给予接受信息过程足够的时间接受信息
        {
            if(COM1.RX_Cnt > 0)
            {
                if(RX1_Buffer[0]=='#')
                {
                    communicate_time = 0;  //通信成功就清零
                        switch(RX1_Buffer[1]){
                            case'0':
                               // led_test(10);
                                message_p[0]=strtok(RX1_Buffer,delim);
                                message_p[0]=strtok(NULL,delim);
                                message_p[1]=strtok(NULL,delim);
                                message_p[2]=strtok(NULL,delim);
                                message_p[3]=strtok(NULL,delim);

                                message_data[0] = atoi(message_p[0]);
                                message_data[1] = atoi(message_p[1]);
                                message_data[2] = atoi(message_p[2]);
                                message_data[3] = atoi(message_p[3]);
                            break;
                            case '2':
                            break;
                            case '3':
                            break;
                            case '4':
                            break;
                            default:
                            break;
                        }
                    }
                }
                COM1.RX_Cnt = 0;
            }
        }
}
```

我们定义了客户端与 MCU 的通信协议，初始模式为“#，1500,1500,1500,1500”，用“#”来确定是否接收数据，用“,”来分割信息，4 个 1500 分别代表“前进”“后退”“左转”“右转”四个方向。

第四章：GL-MOTO-Mini 手机客户端（Android 版）使用说明

1、连接上 GL-MOTO-Mini 发出的 wift，如下图：

2、点击“PLAY”按钮

3、进入界面