项目七 基于蓝牙 4.0 BLE 的光照采集和窗帘控制系统的设计

知识目标

(1)掌握光敏传感器的基本原理和硬件设计方法；
(2)掌握步进电机的基本原理和硬件设计方法；
(3)掌握主机串口数据通知功能的原理；
(4)掌握蓝牙 4.0 BLE 主机使能从机通知功能的原理和实现方法。

技能目标

(1)掌握光敏传感器的驱动设计方法；
(2)掌握步进电机的驱动设计方法；
(3)能够实现蓝牙 4.0 BLE 主机客户端光照通知数据接收功能的程序设计；
(4)能够实现蓝牙 4.0 BLE 主机无线控制从机步进电机模块工作的程序设计。

项目分析

本项目仍然是蓝牙无线传感网络的典型应用，要求实现的功能是从机服务器模块实时采集当前的光照强度，并利用通知功能发送；主机客户端模块实时接收光照数据，并通过串口显示。同时，主机可以通过串口命令无线控制从机窗帘上的步进电机，从而实现窗帘的无线打开和关闭。

无线蓝牙 4.0 BLE 光照采集和无线窗帘控制系统的实现效果如图 7-1 和图 7-2 所示。

由于本项目中主机不再使用智能手机端应用程序，因而主机示例工程 simpleBLECentral 的程序设计是实现本项目的核心，而从机服务器端的设计实现与上一项目流程类似。

基于蓝牙 4.0 BLE 的无线光照控制系统中主机客户端的工作流程如图 7-3 所示。

基于蓝牙 4.0 BLE 的无线光照控制系统中从机服务器端的工作流程如图 7-4 所示。

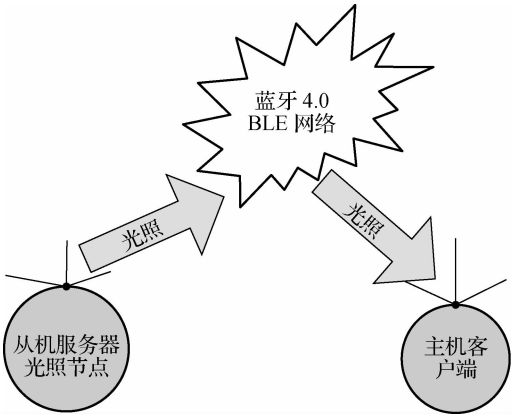

图 7-1　光照采集效果图

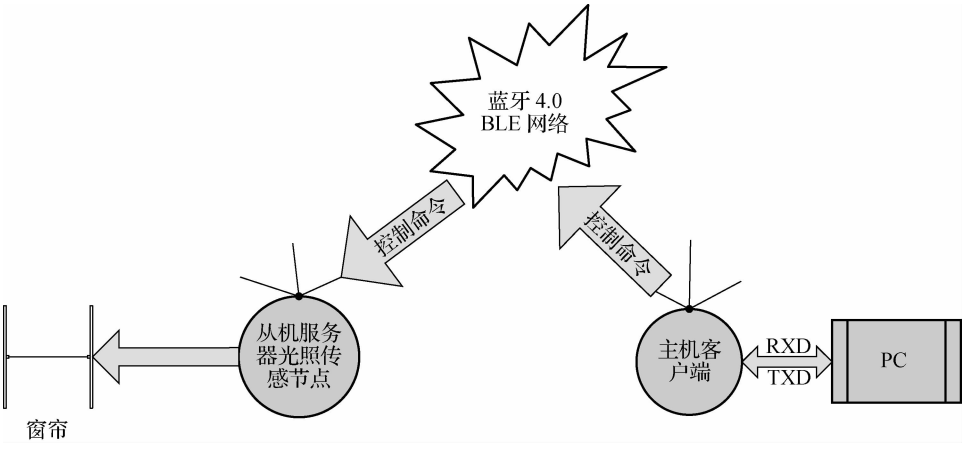

图 7-2　无线窗帘控制效果图

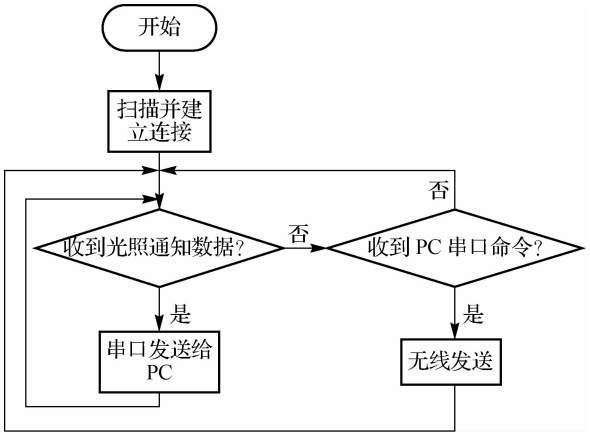

图 7-3　基于蓝牙 4.0 BLE 的无线光照控制系统中主机客户端的工作流程图

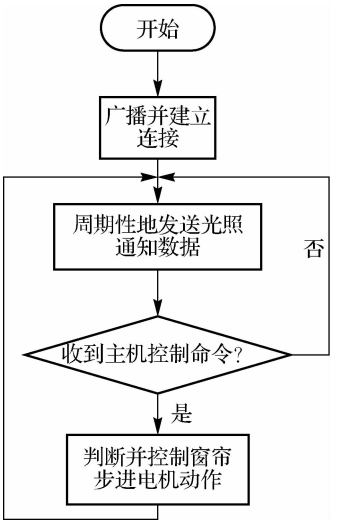

图 7-4　基于蓝牙 4.0 BLE 的无线光照控制系统中从机服务器端的工作流程图

模块一　系统硬件电路的基本原理及设计

光敏传感器驱动电路把传感器输出的电信号变成 CC2540 蓝牙单片机需要的一个电平，驱动逻辑电路的导通。电机驱动可以视为一个可以由电路控制的开关，这里使用集成驱动芯片进行控制。本模块将分别介绍光敏传感器和电机驱动电路的原理及其设计。

任务一　光敏传感器硬件驱动电路的设计

任务描述

本任务要求根据光敏传感器的特点，设计基于 CC2540 单片机的光敏传感器典型驱动电路。

知识链接

1. 光敏传感器的特性

光敏传感器常用的制作材料为硫化镉，另外还有硒、硫化铝、硫化铅和硫化铋等材料。

光敏传感器制造工艺简单，价格便宜。其中，光敏电阻没有极性，只是一个电阻器件，使用时既可以加直流电压，也可以加交流电压。当光敏电阻受到光的照射时，其材料的电导率发生变化，表现出阻值的变化。当无光照时，其呈高阻状态，暗电阻一般可达 1.5 MΩ。当有光照时，材料中激发出自由电子和空穴，其电阻值减小，随着光照强度的升高，电阻值迅速降低，亮电阻值可小至 1 kΩ 以下。

光敏传感器由于具有体积小、灵敏度高、性能稳定、价格低等特点，因而在自动控制、家用电器中得到了广泛应用。光敏传感器一般用于光的测量、光的控制和光电转换（将光的变化转换为电的变化）。常用的光敏传感器为硫化镉光敏传感器，它是由半导体材料制成的。光敏传感器对光的敏感性（光谱特性）与人眼对可见光（0.4～0.76 μm）的响应很接近，只要是人眼可感受的光都会引起它的阻值变化。设计光控电路时，都用白炽灯泡（小电珠）光线或自然光线作为控制光源，使设计大为简化。

通常，光敏传感器都制成薄片结构，以便吸收更多的光能。当它受到光的照射时，半导体片（光敏层）内就激发出电子-空穴对，参与导电，使电路中的电流增强。为了获得高的灵敏度，光敏传感器的电极常采用梳状图案，如图 7-5 所示。它是在一定的掩膜下向光电导薄膜上蒸镀金或铟等金属形成的。一般光敏传感器的外形如图 7-6 所示。

（1）光敏传感器的分类。光敏传感器根据其光谱特性，可分为紫外光敏传感器、红外光敏传感器和可见光光敏传感器。

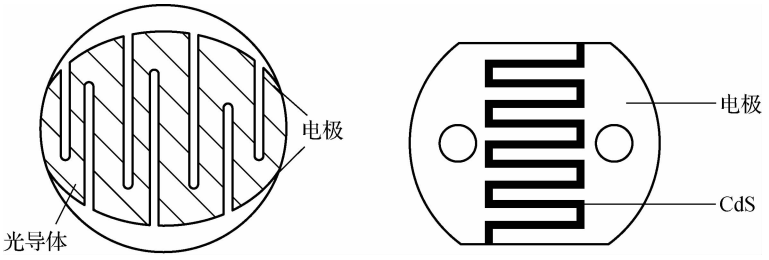

图 7-5　光敏传感器的电极图案

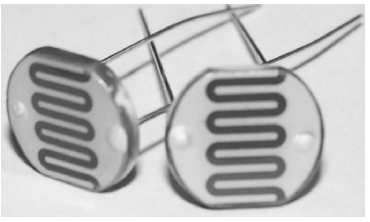

图 7-6　一般光敏传感器的外形

(2)光敏传感器的参数及特性。光敏传感器的主要参数及特性包括以下几点。

①光电流、亮电阻。光敏传感器在室温和一定光照条件下测得的稳定电阻值称为亮电阻或亮阻。此时,流过的电流称为亮电流,常用 100 lx 表示该条件下测得的电阻值。

②暗电流、暗电阻。光敏传感器在不受光照射时流过的电流称为暗电流。外加电压与暗电流之比称为暗电阻,常用 0 lx 表示该条件下的电阻值。由于暗电阻随关闭光源的时间增长而增加,因而规定在关闭电源 30 s 后测量暗电阻值。

③灵敏度。灵敏度是指光敏传感器不受光照射时的电阻值(暗电阻)与受光照射时的电阻值(亮电阻)的相对变化值。

④光谱响应。光谱响应又称光谱灵敏度,光敏传感器对入射光的光谱具有选择作用,即光敏传感器对不同波长的入射光有不同的灵敏度,光敏传感器的灵敏度与入射波长的关系称为光敏传感器的光谱特性,也称为光谱响应。若将不同波长下的灵敏度画成曲线,则可以得到光谱特性的曲线。

⑤光照特性。光照特性指光敏传感器输出的电信号随光照度而变化的特性。从光敏传感器的光照特性曲线可以得出,随着光照强度的增加,光敏传感器的阻值开始迅速下降。若进一步增大光照强度,则电阻值变化减小,然后逐渐趋向平缓。在大多数情况下,该特性为非线性。

⑥伏安特性曲线。伏安特性曲线用来描述光敏传感器的外加电压与光电流的关系,对于光敏器件来说,其光电流随外加电压的增大而增大。

⑦温度系数。光敏传感器的光电效应受温度影响较大,部分光敏传感器在低温下的灵敏度较高,而在高温下的灵敏度则较低。

⑧额定功率。额定功率是指光敏传感器用于某种线路中所允许消耗的功率,当温度升高时,其消耗的功率降低。

2. 光敏传感器的工作原理

光敏传感器的工作原理基于内光电效应。在半导体光敏材料两端装上电极引线，将其封装在带有透明窗的管壳里就构成光敏传感器。当光敏传感器受到一定波长范围的光照射时，光子能量将激发产生电子-空穴对，增强导电性能。在光敏传感器两端的金属电极上加电压，其中便有电流通过，电流就会随光强的增大而变大，从而实现光电转换，如图 7-7 所示。半导体的导电能力取决于半导体导带内载流子数目的多少。当光敏传感器受到光照时，价带中的电子吸收光子能量后跃迁到导带，成为自由电子，同时产生空穴，电子-空穴对的出现使电阻率变小。光照越强，光生的电子-空穴对越多，阻值越低。当光敏传感器两端加上电压后，流过光敏传感器的电流随光照增强而增大。入射光消失，电子-空穴对逐渐复合，电阻逐渐恢复原值，电流逐渐减小。

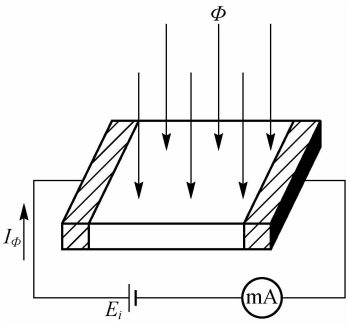

图 7-7　光敏传感器的工作原理示意图

任务实施

根据光敏传感器的特点，其与 CC2540 单片机的接口硬件电路设计如图 7-8 所示。

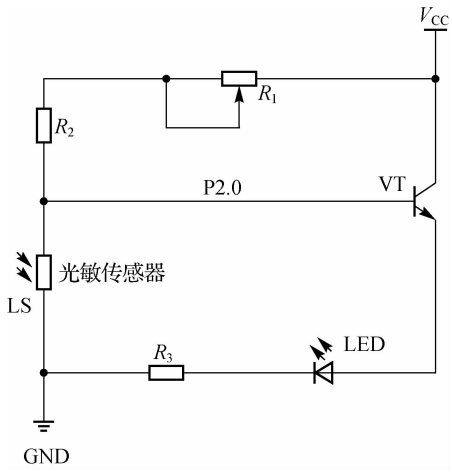

图 7-8　光敏传感器与 CC2540 单片机的接口硬件电路设计

其中，LS 是光敏传感器，利用光敏传感器的特性，在非强光照射条件下，LS 的电阻值会趋向无穷大，处于截止状态，P2.0 口保持高电平状态，同时晶体管导通，LED 指示灯被点亮（代表没有光）；在受到强光照射后，光敏传感器的阻值会迅速变小，处于导通状态，P2.0 口直接接地变为低电平，晶体管 VT 截止，LED 指示灯熄灭（代表有光）。此外，光敏传感器的感光度打开时有一个自定义的阈值，可以通过光敏传感器上方的可调电阻 R_1 来调节阈值（调整分压值）。

任务二　步进电机控制接口电路的设计

任务描述

本任务要求根据步进电机的特点，设计基于 CC2540 单片机的步进电机控制接口电路。

知识链接

1. 步进电机简介

步进电机作为执行元件，是机电一体化的关键产品之一，广泛应用于各种自动化控制系统中。随着微电子和计算机技术的发展，步进电机的需求量与日俱增，在国民经济各个领域都有应用。

步进电机控制系统包括控制器、驱动器和步进电机三部分，如图 7-9 所示。

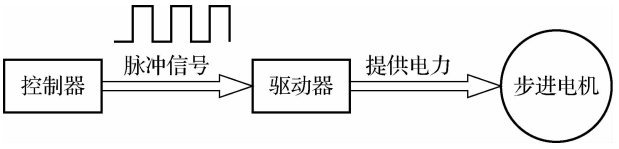

图 7-9　步进电机控制系统的组成

控制器又称为脉冲产生器，目前主要有 PLC、单片机和运动板卡等，本项目中为 CC2540 蓝牙单片机。

2. 步进电机的工作原理

步进电机又称脉冲电动机，是一种将电脉冲转化为角位移的执行机构。通俗来说，当步进驱动器接收到一个脉冲信号时，它就驱动步进电机按设定的方向转动一个固定的角度（步距角）。可以通过控制脉冲个数来控制角位移量，从而达到准确定位的目的；也可以通过控制脉冲频率来控制步进电机转动的速度和加速度，从而达到调速的目的。

在非超载的情况下，步进电机的转速、停止的位置只取决于脉冲信号的频率和脉冲数，不受负载变化的影响，即给步进电机加一个脉冲信号，电动机则转过一个步距角。这一线性关系的存在，加上步进电机只有周期性的误差而无累积误差（精度为 100%）等特点，使得用步进电机来控制速度、位置等变得非常简单，非常适合应用于各种开环控制。图 7-10 所示为步进电机实物图，常用的步进电机有反应式步进电机、永磁式步进电机和混合式步进电机，对于单片机驱动控制设计人员而言，只需要知道步进电机的相关技术参数即可进行程序的设计。

3. 步进电机的控制方法

通常步进电机的转子为永磁体，当电流流过定子绕组时，定子绕组产生一个矢量磁场，如图 7-11 所示。该磁场会带动转子旋转一个角度，使得转子的一对磁场方向与定子的磁场方向一致。当定子的矢量磁场旋转一个角度时，转子也随着该磁场转一个角度。每输入一个电脉冲，步进电机就转动一个角度前进一步。

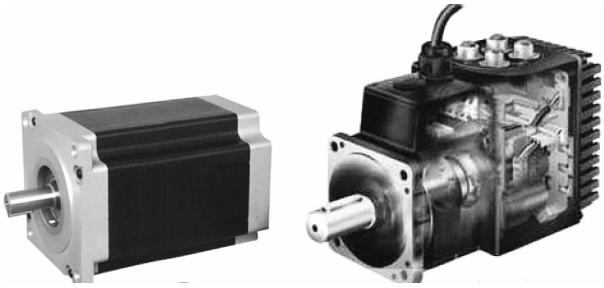

图 7-10　步进电机实物图

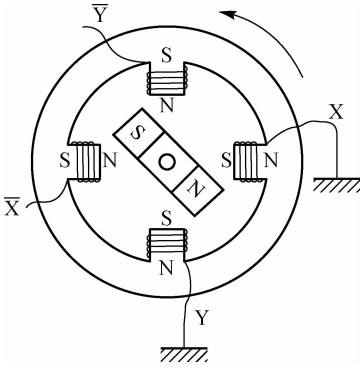

图 7-11　步进电机的转子和定子

步进电机输出的角位移与输入的脉冲数成正比，转速与脉冲频率成正比，改变绕组通电的顺序，步进电机就会反转。步进电机运转量与脉冲数的比例关系公式如下。

步进电机运转量(角度)＝步距角(角度/步距)×脉冲数

步进电机运转速度与脉冲速度的比例关系公式如下。

$$\text{步进电机运转速度(r/min)}=\frac{\text{步距角(角度/步距)}}{360^\circ}\times\text{脉冲速度(Hz)}\times 60$$

因此，可用控制脉冲数、频率及电动机各相绕组的通电顺序来控制步进电机的运转量、运转速度和运转方向。

步进电机能直接接收数字量的控制，所以非常适合用 CC2540 单片机进行控制。步进电机的具体控制方法如下。

(1)通过控制脉冲个数来控制角位移量，从而达到准确定位的目的。步进电机与直流电动机不同，它的转速是可以一次性准确控制的。本任务中采用的步进电机的步距角为 0.9/1.8°。在四相八拍工作方式下，每步的转角为 0.9°。因此，步进电机每转一圈就走了 400 步。

改变“设置移动距离”，步进电机每转一圈，则移动的直线距离为

$$S_1=2\text{PI}\cdot R(\text{周长})$$

同时，可算出步进电机每步移动的距离为

$$S_2=S_1/400=\text{PI}\cdot R/200$$

如果要移动距离 S，则需要的步数为

$$m=\frac{S}{S_2}=\frac{200S}{\mathrm{PI}\cdot R}$$

对于既定的半径值来说，步数只与移动的距离成正比。

(2)通过控制脉冲频率来控制电动机转动的速度和加速度，从而达到调速的目的。如果给定步进电机一个控制脉冲，它就转一步，再给定一个控制脉冲，它就再转一步。两个脉冲的间隔时间越短，步进电机转得越快。因此，脉冲的频率决定了步进电机的转速。调整单片机发出脉冲的频率，就可以对步进电机进行调速。调整单片机输出的步进脉冲频率的方法如下。

①软件延时方法。改变延时的时间长度就可以改变输出脉冲的频率，但这种方法使CPU 长时间等待，无法进行其他工作，在单独进行步进电机的演示时可以采用。

②定时器中断方法。在中断服务子程序中进行脉冲输出操作，调整定时器的定时常数就可以实现调速。这种方法占用的 CPU 时间较少，是一种比较实用的调速方法。

用单片机对步进电机进行速度控制，实际上就是控制每次换相的时间间隔。升速时，使脉冲频率逐渐升高，降速时则相反。

(3)通过改变脉冲的顺序改变步进电机的转动方向。

如图 7-12 所示，四相反应式步进电机主要由定子和转子组成，在定子上均匀分布着 8 个磁极，磁极与磁极之间的夹角是 45°，每个磁极上均绕有线圈，每两个相对绕组组成一相，共组成 A、B、C、D 四相。步进电机转子上没有绕线圈，如图中所示只有 6 个齿，转子齿与齿之间的夹角(齿距角)为 60°。

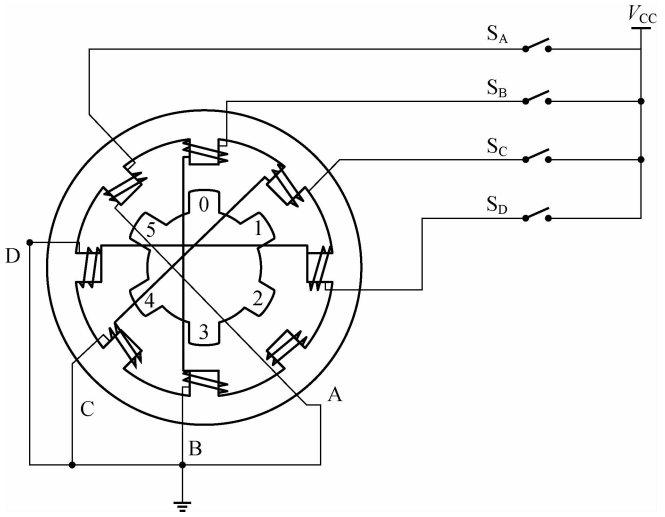

图 7-12　四相反应式步进电机的结构原理图

四相步进电机按照通电顺序的不同，可分为单四拍、双四拍和八拍三种工作方式。八拍工作方式的步距角是单四拍和双四拍的一半，因此，八拍工作方式既可以保持较高的转动力矩，又可以提高控制精度。当依次接通 S_A、S_B、S_C、S_D 开关时，A、B、C、D 四相依次得电，即使得线圈的通电顺序依次为 A—B—C—D—A，来一个脉冲电信号就是一拍或一步，这种通电方式称为四相四拍通电方式，拍数通常等于相数或相数的整数倍。

在四相四拍方式下，当 A 相通电，B、C、D 相不通电时，由于磁场作用，齿 1 与 A 对齐。

当 B 相通电，A、C、D 相不通电时，齿 2 应与 B 对齐，以此类推。

四相步进电机在各工作方式下的通电顺序如下。

①四相四拍工作方式：电动机控制绕组 A、B、C、D 相的正转通电顺序为 A—B—C—D—A，反转通电顺序为 A—D—C—B—A。

②四相八拍工作方式：正转绕组的通电顺序为 A—AB—B—BC—C—CD—D—DA—A，反转绕组的通电顺序为 A—DA—D—DC—C—CB—B—BA—A。

③双四拍的工作方式：正转绕组的通电顺序为 AB—BC—CD—DA，反转绕组的通电顺序为 DA—CD—BC—AB。

对于本控制系统设计中所使用的窗帘步进电机，若要使它在四相八拍的工作方式下正转，则应依次给 CC2540 单片机 P1 口送 0x01—0x03—0x02—0x06—0x04—0x0c—0x08—0x09—0x01，反转则依次送 0x01—0x09—0x08—0x0c—0x04—0x06—0x02—0x03—0x01。

任务实施

步进电机的控制接口电路如图 7-13 所示。硬件由 CC2540 单片机、ULN2003 驱动器和步进电机组成。单片机通过 I/O 口输出的时序方波作为步进电机的控制信号，信号经过芯片 ULN2003 驱动步进电机。ULN2003 是一个大电流驱动器，为达林顿管阵列电路，可输出 500 mA 电流，并且能够在关态时承受 50 V 的电压，输出还可以在高负载电流下并行运行；同时起到电路隔离作用，各输出端与 COM 间有反相二极管，为断电后的步进电机绕组提供一个放电回路，起放电保护作用。

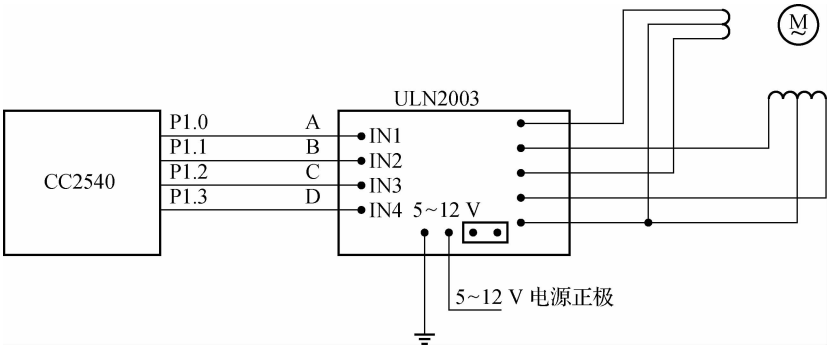

图 7-13　步进电机的控制接口电路

模块二　蓝牙 4.0 BLE 从机服务器端程序功能的实现

蓝牙 4.0 从机服务器模块将采集到的光照数据发送给主机客户端，主机客户端根据接收到的数据值可以控制窗帘的动作，从而实现蓝牙 4.0 主机客户端和蓝牙 4.0 传感器服务器端的双工通信。本模块将重点介绍从机服务器端的程序设计方法。

任务一　从机服务器端光照数据发送功能的实现

任务描述

本任务要求使用光敏传感器实现从机服务器光照数据以 1 s 的周期进行本地采集，并使用协议栈第四特征值的通知功能，无线发送给主机客户端。

知识链接

1. 任务分析

本任务中的传感器数据采集流程与项目六中类似，但本任务中只有单一的光照数据，所以读者无须自行添加带有通知功能的特征值，直接使用或修改带有通知功能的第四特征值即可，这里使用第四特征值默认的通知机制，读者可以参考项目六模块二中任务一的程序代码和介绍，在第四特征值的值发生改变，并且主机客户端使能第四特征值的通知功能后，从机服务器端将调用 GATTServApp_ProcessCharCfg()函数通知主机客户端第四特征值的值已经改变。

此外，本任务需要周期性地采集光照数据，所以仍然使用从机应用层周期性事件处理函数 performPeriodicTask()，由定时函数 osal_start_timerEx()决定采集时间，并最终实现通知功能发送。

2. 驱动程序设计思路

由于本任务中采集的光照数据为开关量，因而驱动程序的设计较为简单，只需要判断 P2.0 端口的状态，所以可以将程序直接写入周期性事件处理函数 performPeriodicTask()中，典型的裸机驱动程序测试函数的相关代码如程序清单 7.1 所示。

程序清单 7.1

```
//光照测试函数
uchar Light_Check(void)
{
if(P2_0==0)
{
Delayms(10);//延时 10 ms,防止误判断
if(P2_0==0)
{
return 1;//返回 1,光照较强
}
}
return 0;//返回 0,光照较弱
}
```

任务实施

本任务的核心为完成协议栈从机周期性事件处理函数的开发，由于光敏传感器的驱动程序较简单，只需对相应的开关量进行判断，因而可以直接将驱动程序嵌入周期性事件处理函数中，定时函数时间的处理与上一项目一致。其相关代码如程序清单 7.2 所示。

程序清单 7.2

```
static void performPeriodicTask(void)
{
uint8 L;
/* * * * * *  P2.0 口初始化  * * * * * * */
P2SEL &=～0x01;//设置 P2.0 为普通 I/O 口
P2DIR &=～0x01;//将 P2.0 口设置为输入模式
P2INP &=～0x01;//打开 P2.0 上拉电阻
if(P2_0==0)
{
L=1;
NPI_WriteTransport("strong light",12);//光照较强
}
else
{
L=0;
NPI_WriteTransport("weak light",10);//光照较弱
}
NPI_WriteTransport("\n",1);
SimpleProfile_SetParameter(SIMPLEPROFILE_CHAR4,sizeof(uint8),&L);
}
```

任务二　从机接收命令控制执行机构功能的实现

任务描述

本任务要求使用第一特征值实现从机服务器接收主机客户端控制命令的功能，并能够判断命令的含义，若收到数据为字符“1”，则步进电机正转；若收到数据为字符“2”，则步进电机反转，从而控制外部设备窗帘的开关。

知识链接

1. 任务分析

从机接收主机控制命令的功能使用特征值改变时的回调函数 simpleProfileChangeCB()，

读者可以参考项目六模块二中任务三的程序代码和介绍。本任务中数据的传输是通过第一特征值的读/写来实现的，第一特征值具有可读/写属性，当数据改变时，通过 SimpleProfile_GetParameter()函数获取第一特征值的新数据并进行判断，从而执行相应的驱动窗帘的步进电机驱动程序。

2. 驱动程序设计思路

根据步进电机的原理和控制方法，本任务中采用四相八拍的工作方式，并由 P1.0～P1.3 口进行控制。典型的裸机正转和反转驱动程序中 P1.0 口的数据表格定义如程序清单 7.3 所示。

程序清单 7.3

```
unsigned char F_Rotation[8]={0x09,0x08,0x0c,0x04,0x06,0x02,0x03,0x01};
//正转表格
unsigned char B_Rotation[8]={0x01,0x03,0x02,0x06,0x04,0x0c,0x08,0x09};
//反转表格
```

任务实施

本任务通过 simpleProfileChangeCB()和 SimpleProfile_GetParameter()函数获取接收到的最新第一特征值，并判断控制命令的含义，执行相应的步进电机驱动程序，其相关代码如程序清单 7.4 所示。

程序清单 7.4

```
static void simpleProfileChangeCB(uint8 paramID)
{
uint8 i,newValue;
uint16 j;
unsigned char F_Rotation[8]={0x09,0x08,0x0c,0x04,0x06,0x02,0x03,0x01};
//正转表格
unsigned char B_Rotation[8]={0x01,0x03,0x02,0x06,0x04,0x0c,0x08,0x09};
//反转表格
switch(paramID)
{
case SIMPLEPROFILE_CHAR1:
SimpleProfile_GetParameter(SIMPLEPROFILE_CHAR1,&newValue);
#if(defined HAL_LCD)&&(HAL_LCD==TRUE)
HalLcdWriteStringValue("Char 1:",(uint16)(newValue),10,HAL_LCD_LINE_3);
#endif
if(newValue=='1')
{
for(j=0;j<800;j++)//决定步进电机所转圈数
{
for(i=0;i<8;i++)//八相
{
```

```
P1DIR |=0x0f;//P1 口低四位输出
P1=F_Rotation[i];//输出对应的相,正转打开窗帘
Motor_Delay(1000);//改变这个参数可以调整电动机转速
}
}
NPI_WriteTransport("curtain opened\n",15);
}
if(newValue==='2')
{
for(j=0;j<800;j++)
{
for(i=0;i<8;i++)
{
P1DIR |=0x0f;
P1=B_Rotation[i];
Motor_Delay(1000);
}
}
NPI_WriteTransport("curtain closed\n",15);
}
break;
……
}
}
```

其中,Motor_Delay()函数的代码如程序清单 7.5 所示,需要同时加入协议栈中并进行函数的声明,如图 7-14 所示。

程序清单 7.5

```
void Motor_Delay(uint16 k)//延时
{
while(--k);
}
```

```
/*********************************************************************
 * LOCAL FUNCTIONS
 */
static void simpleBLEPeripheral_ProcessOSALMsg( osal_event_hdr_t *pMsg );
static void peripheralStateNotificationCB( gaprole_States_t newState );
static void performPeriodicTask( void );
static void simpleProfileChangeCB( uint8 paramID );
static void UartEventTrigger (uint8 port, uint8 event)   ;
void Motor_Delay(uint16 k );
```

图 7-14　电动机延时函数声明

小贴士

不同的步进电机型号对拍与拍之间的延迟时间的要求不同，读者需要根据实际的硬件环境和产品说明书进行调整。

模块三　蓝牙 4.0 BLE 主机客户端程序功能的实现

蓝牙 4.0 BLE 主机客户端需要发送特定的指令使能从机服务器端的通知发送功能，同时能够读取用户通过串口发送的窗帘控制命令，并转发给从机服务器端。本模块将分别介绍这两个功能具体的程序实现方法。

任务一　主机客户端光照通知数据接收功能的实现

任务描述

本任务要求实现主机客户端能够接收从机服务器通过第四特征值发送过来的光照通知数据，并将接收到的数据通过主机串口实时显示的功能。

知识链接

1. 通知功能的使能

主机要能够接收从机发送的光照数据，就必须使能从机的通知功能，即需要向第四特征值的通知功能句柄中写入数据 0x0001。在项目五模块一的任务一中已经实现了通过 BTool 工具使能从机通知的功能，本任务需要编程实现在主机和从机连接成功后自动地向第四特征值的通知功能句柄中写入数据 0x0001，写入功能由 GATT_WriteCharValue()函数实现，该函数可以向从机指定的句柄写入数据，函数的原型如下。

```
extern bStatus_t GATT_WriteCharValue(uint16 connHandle,attWriteReq_t * pReq,
uint8 taskId);
```

对于这个函数，TI 只提供接口，定义被封装了，读者只要学会使用即可。attWriteReq_t 参数为写入数据属性的结构体，其相关定义如程序清单 7.6 所示。

程序清单 7.6

```
typedef struct
{
uint16 handle;
uint8 len;
uint8 value[ATT_MTU_SIZE-3];
uint8 sig;
```

```
uint8 cmd;
} attWriteReq_t;
```

其中，handle，len 和 value 变量即为写入数据属性的句柄、长度和值。

2. 主机消息处理函数简介

(1)接收数据的获取方法。所有通过蓝牙发送的数据都会让 GATT 层的事件来处理，在主机使能通知后，从机服务器发送的光照通知数据将会发送到主机客户端的消息处理函数 simpleBLECentralProcessGATTMsg()中，协议栈会将接收到的数据及与数据相关的一些信息打包，然后存储起来，找到这个打包数据的结构体定义就能找到想要的数据。

首先查看来自 OSAL 层的消息处理函数 simpleBLECentral_ProcessOSALMsg()，其相关代码如程序清单 7.7 所示。

程序清单 7.7

```
static void simpleBLECentral_ProcessOSALMsg(osal_event_hdr_t * pMsg)
{
switch(pMsg->event)
{
case KEY_CHANGE:
simpleBLECentral_HandleKeys(((keyChange_t * )pMsg)->state,((keyChange_t * )
pMsg)->keys);
break;
case GATT_MSG_EVENT:
simpleBLECentralProcessGATTMsg((gattMsgEvent_t * )pMsg);
break;
}
}
```

该函数中除了包含一个按键事件外，还有一个 GATT 层的消息事件，并调用 simpleBLECentralProcessGATTMsg( ) 函数进行应用数据的处理，其中包含了一个 gattMsgEvent_t 类型的结构体参数 pMsg。该结构体的相关代码如程序清单 7.8 所示。

程序清单 7.8

```
typedef struct
{
osal_event_hdr_t hdr;//GATT_MSG_EVENT and status
uint16 connHandle;//Connection message was received on
uint8 method;//Type of message
gattMsg_t msg;//Attribute protocol/profile message
} gattMsgEvent_t;
```

这里最重要的就是 msg 成员变量，根据协议栈注释，gattMsg_t 类型的变量存放着协议栈应用层打包的消息数据，gattMsg_t 结构体的定义如程序清单 7.9 所示。

程序清单 7.9

```
typedef union
{
```

```
//Request messages
……
//Response messages
attErrorRsp_t errorRsp;//包含属性错误码
attExchangeMTURsp_t exchangeMTURsp;
attFindInfoRsp_t findInfoRsp;
attFindByTypeValueRsp_t findByTypeValueRsp;
attReadByTypeRsp_t readByTypeRsp;
attReadRsp_t readRsp;//读取到的特征值数据
attReadBlobRsp_t readBlobRsp;
attReadMultiRsp_t readMultiRsp;
attReadByGrpTypeRsp_t readByGrpTypeRsp;
attPrepareWriteRsp_t prepareWriteRsp;
//通知和指示数据
attHandleValueNoti_t handleValueNoti;//特征值句柄通知数据属性
attHandleValueInd_t handleValueInd;//特征值句柄指示数据属性
} gattMsg_t;
```

本任务中接收到的通知数据属性存放在结构体 attHandleValueNoti_t 中的三个变量中，该结构体的相关代码如程序清单 7.10 所示。

程序清单 7.10

```
typedef struct
{
uint16 handle;//Handle of the attribute that has been changed(must be first field)
uint8 len;//Length of value
uint8 value[ATT_MTU_SIZE-3];//New value of the attribute
} attHandleValueNoti_t;
```

len 变量是接收数据的长度，数据接收后存放在 value 变量中，&pMsg->msg.handleValueNoti.value 就是存放接收数据的首地址。因此，如果要把从机服务器端发送的“2016”字符显示出来，只需要将 value 指针首地址开始的 4 个字符取出来送到串口即可。

(2)消息类型的判断和处理方法。进入 simpleBLECentralProcessGATTMsg()函数，其相关代码如程序清单 7.11 所示。

程序清单 7.11

```
static void simpleBLECentralProcessGATTMsg(gattMsgEvent_t * pMsg)
{
if(simpleBLEState!=BLE_STATE_CONNECTED)
{
//若连接掉线，则忽略此消息
return;
}
```

```
//读取特征值失败后,消息数据错误
if((pMsg->method==ATT_READ_RSP)||
((pMsg->method==ATT_ERROR_RSP)&&
(pMsg->msg.errorRsp.reqOpcode==ATT_READ_REQ)))
{
if(pMsg->method==ATT_ERROR_RSP)
{
uint8 status=pMsg->msg.errorRsp.errCode;
LCD_WRITE_STRING_VALUE("Read Error",status,10,HAL_LCD_LINE_1);
}
else
{
//读取特征值成功后,显示读取的值
uint8 valueRead=pMsg->msg.readRsp.value[0];
LCD_WRITE_STRING_VALUE("Read rsp:",valueRead,10,HAL_LCD_LINE_1);
}
simpleBLEProcedureInProgress=FALSE;
}
//写入特征值数据错误
else if((pMsg->method==ATT_WRITE_RSP)||
((pMsg->method==ATT_ERROR_RSP)&&
(pMsg->msg.errorRsp.reqOpcode==ATT_WRITE_REQ)))
{
if(pMsg->method==ATT_ERROR_RSP==ATT_ERROR_RSP)
{
uint8 status=pMsg->msg.errorRsp.errCode;
LCD_WRITE_STRING_VALUE("Write Error",status,10,HAL_LCD_LINE_1);
}
else
{
//写入成功后,显示写入的值并加 1
LCD_WRITE_STRING_VALUE("Write sent:",simpleBLECharVal++,10,HAL_LCD_LINE_1);
}
simpleBLEProcedureInProgress=FALSE;
}
else if(simpleBLEDiscState!=BLE_DISC_STATE_IDLE)
{
simpleBLEGATTDiscoveryEvent(pMsg);
}
}
```

在该函数中给出了接收到不同消息类型的处理方法，但本任务需要接收的消息为通知类型，即判断接收到的消息类型 pMsg->method 是否为 ATT_HANDLE_VALUE_NOTI，协议栈中并没有给出，需要用户自行修改和添加。

任务实施

1. 使能通知功能的实现

为了实现主机和从机连接成功后自动地使能通知功能，在 simpleBLECentralEventCB()函数中的连接参数更新事件 GAP_LINK_PARAM_UPDATE_EVENT 中打开第四特征值的通知，其相关代码如程序清单 7.12 所示。

程序清单 7.12

```
static void simpleBLECentralEventCB(gapCentralRoleEvent_t * pEvent)
{
switch(pEvent->gap.opcode)
{
……
/ * 若收到从机发出的连接参数更新的消息，则显示参数已更新，并发送 0x0001 到通知句柄 * /
case GAP_LINK_PARAM_UPDATE_EVENT:
{
LCD_WRITE_STRING("Param Update",HAL_LCD_LINE_1);
attWriteReq_t AttNotireq;
uint8 EnableValue[2];
AttNotireq.handle=0x002f;//第四特征值句柄+1
AttNotireq.len=2;
EnableValue[0]=0x01;
EnableValue[1]=0x00;//使能数据 0x0001
osal_memcpy(AttNotireq.value,EnableValue,2);
AttNotireq.sig=0;
AttNotireq.cmd=0;
//向从机通知句柄中写入属性
GATT_WriteCharValue(0,&AttNotireq,simpleBLETaskId);
NPI_WriteTransport("Notice Enabled\n",15);
}
break;
……
}
}
```

2. 主机光照数据接收功能的实现

在 simpleBLECentralProcessGATTMsg()消息处理函数中添加通知消息处理类型，并

判断接收到的数据，从而执行相应的步进电机驱动程序。其相关代码如程序清单 7.13 所示。

程序清单 7.13

```
static void simpleBLECentralProcessGATTMsg(gattMsgEvent_t * pMsg)
{
……
else if(simpleBLEDiscState!=BLE_DISC_STATE_IDLE)
{
simpleBLEGATTDiscoveryEvent(pMsg);
}
else if(pMsg->method==ATT_HANDLE_VALUE_NOTI)//如果是通知事件
{
//第四特征值的通知，对接收到的数据进行判断并打印
if(pMsg->msg.handleValueNoti.handle==0x002e)
{
NPI_WriteTransport("central:",8);
if(pMsg->msg.handleValueNoti.value[0]==1)//收到数据值为 1
{
NPI_WriteTransport("strong light\n ",13);//光照较强
}
else
{
NPI_WriteTransport("weak Light\n ",11);//光照较弱
}
}
}
}
```

任务二　主机发送无线串口控制命令功能的实现

任务描述

本任务要求主机能够接收用户 PC 串口控制命令，并通过第一特征值将正转和反转控制命令 0x00 和 0xff 发送给从机，从而控制步进电机的动作。

知识链接

在项目四中详细介绍了串口数据接收功能的实现，在项目五中完成了对第一特征值的写入测试，本任务中仍然使用串口回调函数 UartEventTrigger()进行串口输入数据的获取，

并利用第一特征值的无线写入功能来实现数据的发送，其核心代码如程序清单 7.14 所示。

程序清单 7.14

```
if(simpleBLEDoWrite)//写入成功与否标志位，写入成功后可再次写入，默认为 FALSE
{
//Do a write
attWriteReq_t req;
req.handle=simpleBLECharHdl;
req.len=1;
req.value[0]=simpleBLECharVal;
req.sig=0;
req.cmd=0;
status=GATT_WriteCharValue(simpleBLEConnHandle,&req,simpleBLETaskId);
}
```

其中，simpleBLECharHdl 为第一特征值的句柄，它的获取是通过处理负责 GATT 层发现事件的 simpleBLEGATTDiscoveryEvent()函数来实现的，在从机服务被发现后，将利用 UUID 读取特征值并存储特征值的句柄，其相关代码如程序清单 7.15 所示。

程序清单 7.15

```
static void simpleBLEGATTDiscoveryEvent(gattMsgEvent_t * pMsg)
{
attReadByTypeReq_t req;
……
//If procedure complete
if(((pMsg->method==ATT_FIND_BY_TYPE_VALUE_RSP)&& (pMsg->hdr.status==
bleProcedureComplete))||(pMsg->method==ATT_ERROR_RSP))
{
if(simpleBLESvcStartHdl!=0)
{
/*通过 UUID 获得特征值，并根据服务的 UUID 调用 API 函数 GATT_ReadUsingCharUUID，协
议栈会返回该服务的 Handle*/
simpleBLEDiscState=BLE_DISC_STATE_CHAR;
req.startHandle=simpleBLESvcStartHdl;
req.endHandle=simpleBLESvcEndHdl;
req.type.len=ATT_BT_UUID_SIZE;
req.type.uuid[0]=LO_UINT16(SIMPLEPROFILE_CHAR1_UUID);
req.type.uuid[1]=HI_UINT16(SIMPLEPROFILE_CHAR1_UUID);
GATT_ReadUsingCharUUID(simpleBLEConnHandle,&req,simpleBLETaskId);
}
}
}
```

```
else if(simpleBLEDiscState==BLE_DISC_STATE_CHAR)
{
//特征值被发现后,将存储特征值的句柄
if((pMsg->method==ATT_READ_BY_TYPE_RSP)&&
(pMsg->msg.readByTypeRsp.numPairs > 0))
{
//获取的第一特征值句柄
simpleBLECharHdl=BUILD_UINT16(pMsg->msg.readByTypeRsp.dataList[0],
pMsg->msg.readByTypeRsp.dataList[1]);
LCD_WRITE_STRING("Simple Svc Found",HAL_LCD_LINE_1);
simpleBLEProcedureInProgress=FALSE;
}
simpleBLEDiscState=BLE_DISC_STATE_IDLE;
```

可见,该函数中默认获取的是第一特征值的句柄属性,用户也可以按照具体的项目要求修改特征值名称来得到相应的句柄值。

任务实施

主机串口数据的接收和第一特征值的无线写入都在串口回调函数 UartEventTrigger() 中进行,其相关代码如程序清单 7.16 所示。

程序清单 7.16

```
static void UartEventTrigger(uint8 port,uint8 event)
{
VOID port;
uint8 temp;
uint8 uartbuf[128];
/*若发送区满或为空*/
if((event & HAL_UART_TX_EMPTY)||(event & HAL_UART_TX_FULL))
{
return;
}
if(event & HAL_UART_RX_TIMEOUT)//串口接收事件
{
temp=NPI_RxBufLen();//读取接收到的数据长度
if(temp)//若数据不为空
{
//主从机已连接成功并获取了第一特征值的句柄
If ((simpleBLEState==BLE_STATE_CONNECTED)&&(simpleBLECharHdl !=0))
  {
if(simpleBLEDoWrite)
```

```
{
NPI_ReadTransport(uartbuf,temp);//读取串口缓冲区数据
attWriteReq_t Req;
Req.handle=simpleBLECharHdl;
Req.len=1;
Req.sig=0;
Req.cmd=0;
osal_memcpy(Req.value,uartbuf,1);
GATT_WriteCharValue(0,&Req,simpleBLETaskId);
simpleBLEDoWrite=FALSE;
}
}
……
}
```

在写入特征值之前，需要将读/写成功标志位 simpleBLEDoWrite 置位，修改 simpleBLEDoWrite 变量的默认定义 FALSE 为 TRUE，其相关代码如下。

```
//特征值读/写标志位
static bool simpleBLEDoWrite=TRUE;
```

特征值写入成功后，在主机 GATT 层消息处理函数 simpleBLECentralProcessGATTMsg() 中，写入成功标志位 simpleBLEDoWrite 将再次置位，允许再次写入数据，其相关代码如程序清单 7.17 所示。

程序清单 7.17

```
static void simpleBLECentralProcessGATTMsg(gattMsgEvent_t * pMsg)
{
……
if(pMsg->method==ATT_ERROR_RSP==ATT_ERROR_RSP)
{
uint8 status=pMsg->msg.errorRsp.errCode;
LCD_WRITE_STRING_VALUE("Write Error",status,10,HAL_LCD_LINE_1);
}
else
{
LCD_WRITE_STRING_VALUE("Write sent:",simpleBLECharVal++,10,HAL_LCD_LINE_1);
simpleBLEDoWrite=TRUE;
}
simpleBLEProcedureInProgress=FALSE;
……
}
```

任务三　下载和调试通信程序

1. 从机服务器端本地光照数据的采集调试

将修改过的协议栈从机程序下载到蓝牙从机服务器模块中，打开串口调试助手，将波特率设为 115 200 b/s。开启从机电源并打开串口，可以在串口调试助手上看到本地采集到的光照情况，若采集到的结果变量 L 为 1，则显示“strong light”；若采集到的结果变量 L 为 0，则显示“weak light”，如图 7-15 所示。

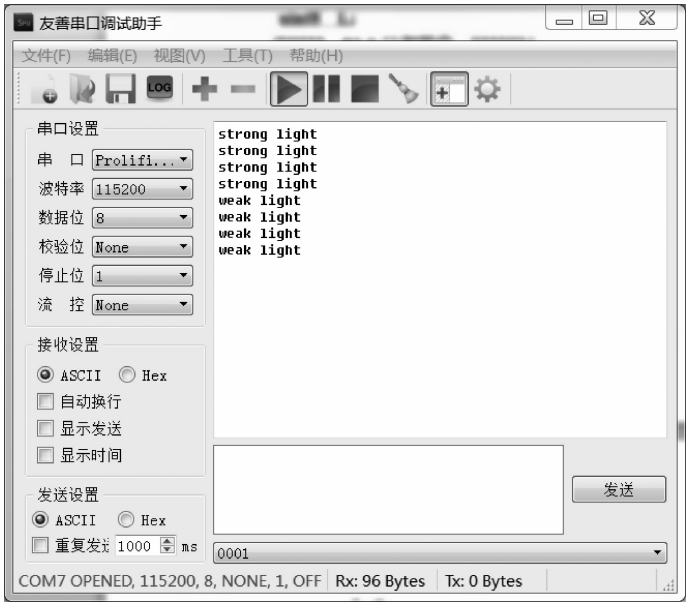

图 7-15　从机本地采集到的光照情况

2. 从机周期性光照通知数据发送和主机接收功能的调试

(1)主机使能通知功能的调试。将修改过的协议栈主机程序下载到主机客户端模块中，打开串口调试助手，将波特率设为 115 200，开启主机电源并打开串口，在与从机建立连接后，可以看到主机客户端相关参数的更新，显示“Notice Enabled”，如图 7-16 所示，说明 0x0001 使能数据已经成功写入第四特征值句柄中。

(2)主机接收功能调试。在从机服务器端第四特征值通知功能被使能后，可以看到主机每隔 1 s 接收和读取到从机服务器端第四特征值中的光照通知数据，并做相应判断后通过串口打印光照情况，如图 7-17 所示，目前的光照情况为较强。

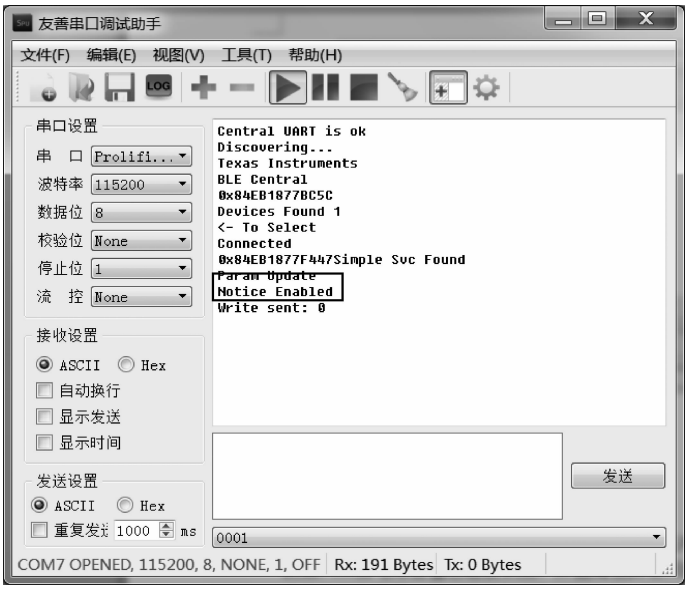

图 7-16　主机使能通知功能成功

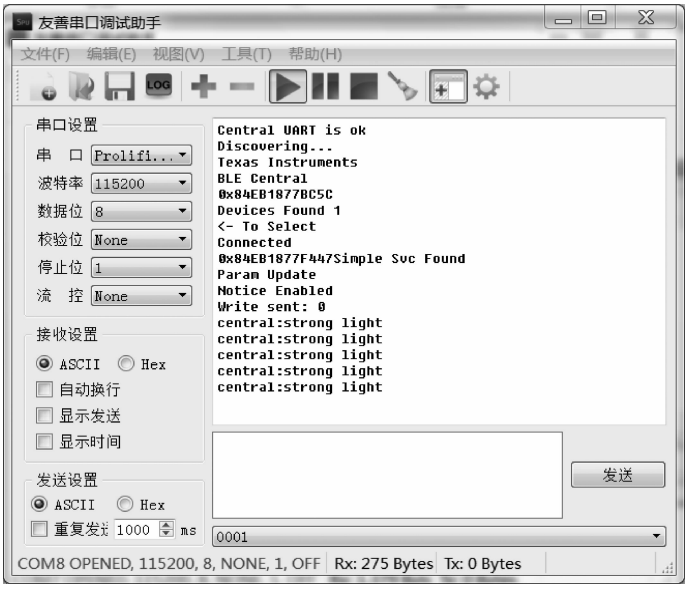

图 7-17　主机接收到光照数据

3. 主机无线串口控制命令发送功能的调试

主机控制从机功能的命令是通过对第一特征值写不同的数据来实现的，在与从机建立连接后，可以在主机串口调试助手发送栏内依次输入正、反转控制命令 1 和 2 并单击“发送”按钮，可以观察到步进电机的正转和反转。同时，在从机串口调试助手中可以观察到步进电机的运行状态，如图 7-18 和图 7-19 所示。

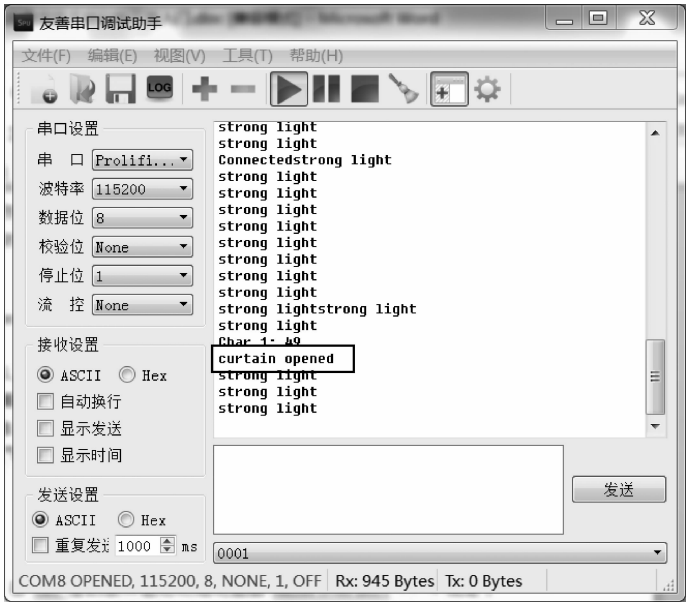

图 7-18　步进电机正转，窗帘打开（接收到正转命令 1）

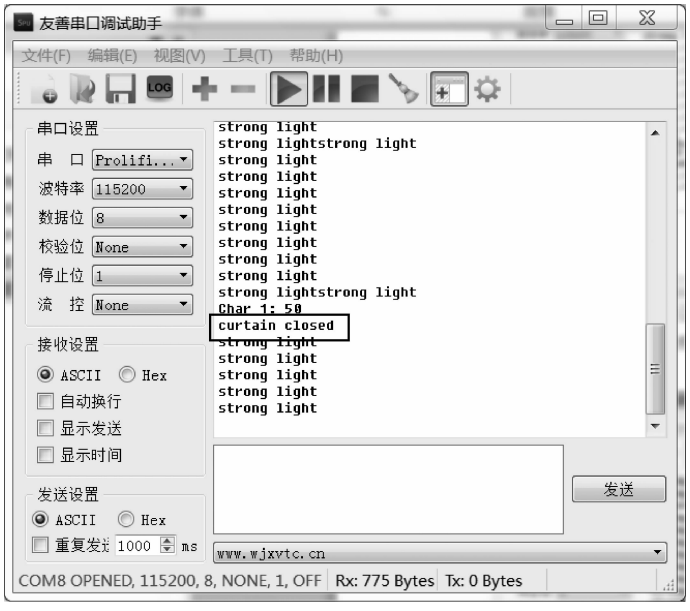

图 7-19　步进电机反转，窗帘关闭（接收到反转命令 2）