

BC7215

Arduino 驱动库

V1.1

及应用例子

使用说明书

目录

驱动库.....	3
简介.....	3
安装.....	3
通过 Arduino 库管理器.....	3
通过 ZIP 文件.....	3
数据类型.....	3
使用.....	4
接口函数.....	5
1. 状态控制函数.....	5
2. 查询函数.....	5
3. 接收相关函数.....	6
4. 发射相关函数.....	7
5. 工具函数.....	8
高级应用.....	9
应用例子.....	11
遥控器解码.....	11
学习型遥控器.....	12
示例程序功能描述.....	13
2 路红外遥控开关.....	13
示例程序功能描述.....	14
红外数据传输.....	14
示例程序功能描述.....	15

驱动库

简介

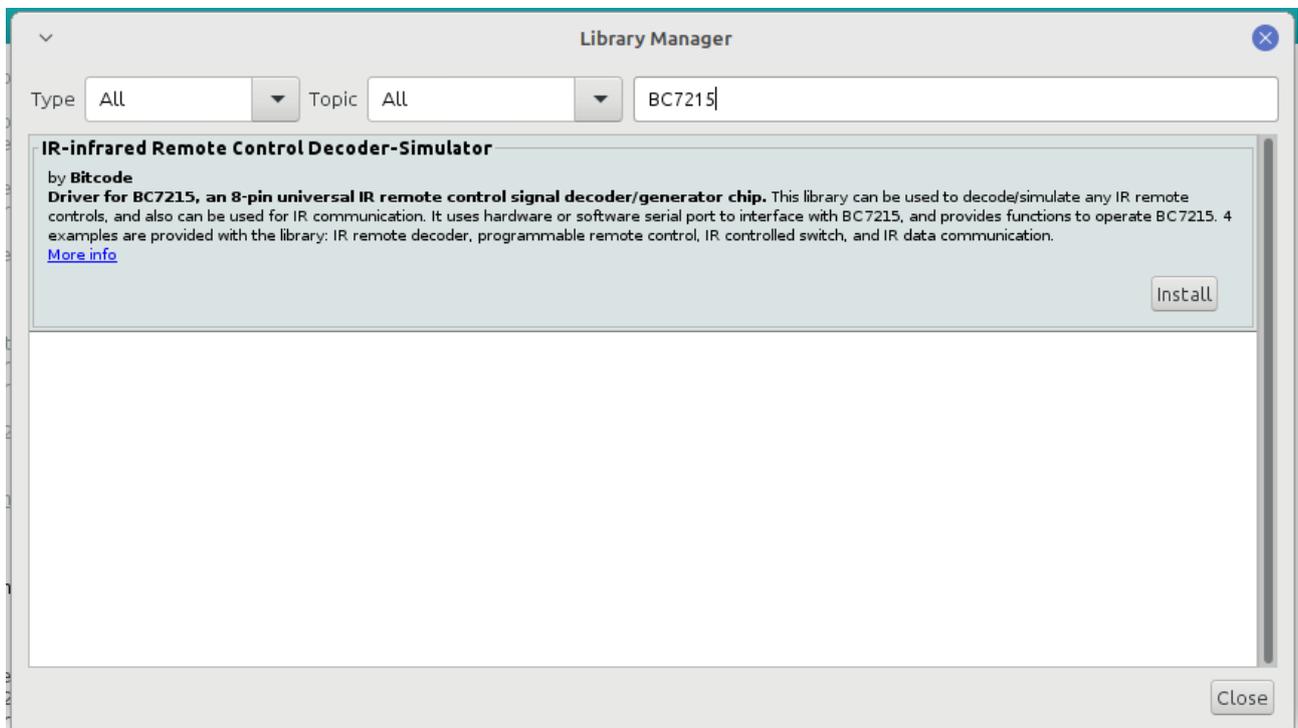
这个驱动库提供了 Arduino 系统和 BC7215 万能红外编解码芯片的接口，可完成面向 BC7215 芯片的各种操作。使用本驱动库，需要对 BC7215 芯片有基本的了解，请参阅 BC7215 数据手册。

本驱动库可工作在任何提供硬件串口的 Arduino 板上，亦可工作于使用软件串口的 Arduino 板，但软件串口必须能够设置为波特率 19200，8 个数据位，2 个停止位。

安装

通过 Arduino 库管理器

本驱动库已经收录进 Arduino 官方库管理器，可通过 Arduino IDE 的库管理器直接安装。



通过 ZIP 文件

在 Arduino IDE 中，选择"项目-->加载库-->添加.ZIP 库"，选择打包文件 bc7215.zip，即可完成安装。安装完成后，即可在 IDE 的 File-->Examples 中，看到随库提供的例程。

数据类型

本驱动库定义了数据类型：

```
struct bc7215DataMaxPkt_t{
```

```
    word bitLen;
    byte  data[BC7215_MAX_DATA_BYTE_LENGTH];
};
```

此类型定义了能处理的最大数据包，而最大能处理的数据包长度，在库配置文件中定义（见“高级应用”部分）。

```
struct bc7215FormatPkt_t{
    union {
        struct {
            byte sig    : 6;
            byte c56k   : 1;
            byte noCA   : 1;
        } bits;
        byte inByte;
    } signature;
    byte format[32];
};
```

此类型定义了 BC7215 的格式信息包。

使用

只需在 sketch 的最顶部，添加

```
#include "bc7215.h"
```

即可使用本驱动库。BC7215 使用串口与 Arduino 连接，并同时需要 2 个数字 I/O 口连接 MOD 和 BUSY 两个信号。使用时，需要首先创立一个 BC7215 的实例，如：

```
BC7215 irModule(Serial1, 6, 7);
```

该程序行创立了一个名为 irModule 的 BC7215 实例，参数中，Serial1 指的是 BC7215 所连接的 Serial 口，而 6 是 MOD 所连接的 I/O 口，7 为 BUSY 信号所连接的 I/O 口。

MOD 和 BUSY 两个信号并非必须连接到 Arduino，在一些应用中，可能只用到 BC7215 的接收或者发射功能，这时 MOD 信号可以直接固定连接到 VCC 或者 GND。而如果不使用红外发射的功能，BUSY 信号也可以悬空不连接到 Arduino。对于这类情况，创立 BC7215 的实例时，由特殊的参数代表这种特殊的硬件连接：

```
BC7215::MOD_HIGH
```

```
BC7215::MOD_LOW
```

```
BC7215::BUSY_NC
```

以上分别代表 MOD 接固定高电平，MOD 接固定低电平，和 BUSY 不连接三种情况。例：

```
BC7215 irModule(Serial1, BC7215::MOD_HIGH, BC7215::BUSY_NC);
```

在初始化函数 setup() 中，用户需要设置 BC7215 所连接的串口的参数

```
Serial1.begin(19200, SERIAL_8N2);
```

而 MOD 和 BUSY 两个 I/O 口，则由驱动库自动设置。

接口函数

BC7215 类提供了

1. 状态控制函数
2. 查询函数
3. 接收相关函数
4. 发射相关函数
5. 工具函数

1. 状态控制函数

用户通过此类函数控制 BC7215 的工作状态。

```
void setRx() ;
```

设置 BC7215 的工作模式为接收(红外解码)模式。实际操作为设置 MOD 信号为高电平。如果调用此函数时 BC7215 正在进行红外发射，则此函数被调用后，BC7215 可能需要完成正在发送的红外比特才能切换到接收状态，最长可能需要 20ms 的时间，用户应保证此期间不对 BC7215 进行其他的操作。。

```
void setTx() ;
```

设置 BC7215 的工作模式为发射(红外编码)模式。实际操作为设置 MOD 信号为低电平。此函数调用后，BC7215 芯片可能需要最多 2ms 的时间完成模式切换，用户应保证此期间不对 BC7215 进行其他的操作。

```
void setShutDown() ;
```

在发射模式下，设置 BC7215 进入关机模式。调用此函数后，向 BC7215 发送 F7 00 指令。此函数只在发射模式下起作用，如果在接收模式下调用，因为最后发送的数据为 0x00，BC7215 的接收模式将变为简单模式(详情请参阅 BC7215 数据手册)。此函数调用后，用户可以查询指令执行状态。

```
void setRxMode(byte mode) ;
```

接收模式下，设置接收解码模式(详情请参阅 BC7215 数据手册)，mode 的最低两位决定指令执行后的工作模式。

2. 查询函数

用户通过此类函数查询 BC7215 的工作状态。

```
bool dataReady() ;
```

查询是否从 BC7215 接收到了有效的红外数据。当有有效数据时，返回 1(true),否则返回 0(false). 此查询仅在接收模式有效，如果在发送模式调用此函数，将永远返回 0. 如果配置中禁止了接收功能，则此函数不可用。

有三种情况下红外数据有效的状态会被清除：

1. 串口接收到了新的数据
2. 调用了获取数据包函数 `getData()` 或者 `getRaw()`
3. 调用了清除数据包函数 `clrData()`

```
bool formatReady() ;
```

查询是否从 BC7215 收到了格式数据，当有有效格式数据时，返回 1(true),否则返回 0(false). 此查询仅在接收模式有效，如果在发送模式调用此函数，将永远返回 0. 如果配置中禁止了接收功能，则此函数不可用。

当为复合模式时，BC7215 除了输出原始数据外，还输出所收到红外信号的格式信息。一般来说如果收到格式信息，之前必定也收到原始数据，但有一种情况是原始数据加格式信息的总长度超过了驱动库缓冲区的长度，这种情况下，之前收到的原始数据会被覆盖，因此只有格式信息是可用的。这种情况下要想获取原始数据，需要将 BC7215 设置为简单模式，重新接收红外信号。

有三种情况下格式数据有效的状态会被清除：

1. 串口接收到了新的数据
2. 调用了获取格式包函数 `getFormat()`
3. 调用了清除数据包函数 `clrFormat()`

```
bool cmdCompleted() ;
```

发射状态下，查询指令是否已经执行完成。可以查询的是发射指令和关机指令。

BC7215 芯片内部具有 16 字节的接收缓冲区，尽管红外发送的速率比较低，但数据量在 16 个字节以内时，发送函数会几乎立刻返回，但实际数据通过红外线发送过程，由 BC7215 芯片完成，则需要较长时间。用户有时需要知道发送完成的时间，比如需要保证连续两次发射的时间间隔时，则可用此函数查询上一次发射的完成时间。

关机指令会立即执行，用户可通过此函数查询 BC7215 是否已经进入关机状态。

返回值为 1(true)时，表示指令执行完成，为 0(false)时，表示尚未完成。在接收状态执行此函数，将始终返回 1.

3. 接收相关函数

```
word getLen() ;
```

获取所接收到的原始数据包的数据长度，此长度为实际数据的比特数，即

`bc7215DataMaxPkt_t` 中的 `bitLen`，`getData()` 函数实际拷贝的数据长度，为这个值所对应的字节数+2.如这里 `bitLen` 为 9 时，原始数据需要 2 个字节，实际拷贝的字节数将为 4.

如果原始数据包不可用，返回结果将为 0.

用户可利用这个函数在获取原始数据前检查数据长度，防止内存溢出，或动态分配内存。

如果配置中禁止了接收功能，则此函数不可用。

```
word dpktSize() ;
```

获取所接收数据包的大小，以字节为单位。此函数与上面获取比特长度函数等价，但得到的是整个数据包的字节数，免去用户自行转换的步骤。因为所接收的数据包由红外发射端决定，有可能会收到超出预期的数据，用户可以在收取数据包前检查数据包大小，防止内存溢出。如果库配置中禁止了接收功能，则此函数不可用。

```
byte getData(bc7215DataMaxPkt_t& target);
```

读取原始数据包函数。输入参数为 `bc7215DataMaxPkt_t` 类型的变量，指令执行后，所接收到的原始数据包会拷入该变量。返回值为原始数据包的状态特征字，用以快速判断所接收数据的状态。如果缓存中的原始数据已经不可用（如已被新数据覆盖等情况），会返回 `0xff`。

此函数执行后，将清除 `dataReady()` 的状态。

用户必须保证目标 `target` 大小能够容纳所接收到的原始数据，如果所接收到的数据长度大于 `target` 的内存空间，执行此函数会造成内存溢出，产生不可预期的后果。用户应先用 `getLen()` 或者 `dpktSize()` 函数获取数据长度，申请内存或检查空间大小后再调用此函数。如果配置中禁止了接收功能，则此函数不可用。

```
word getRaw(void* addr, word size);
```

获取所接收数据包中的原始数据函数。此函数和上面读取数据包函数类似，不过此函数只读出原始数据而不包含位长度信息，而且可以输出到任意地址，更适合作为红外通信时使用。

`size` 为希望读取的字节数，该值不一定与接收到的数据相同，可以小于或者大于实际接收到的字节数，小于时，按 `size` 的值返回字节数，当 `size` 大于实际接收到字节数时，仅读出实际接收到的字节数。函数的返回值，是实际读出的字节数。

如果配置中禁止了接收功能，则此函数不可用。

```
byte getFormat(bc7215FormatPkt_t& target);
```

读取格式数据包函数。输入参数为一 `bc7215FormatPkt_t` 类型的变量，指令执行后，所接收到的格式数据包将会拷入该变量。返回值为格式数据包的特征字，如果缓存中的原始数据已经不可用（如已被新数据覆盖等情况），会返回 `0xff`。

此函数执行后，将清除 `formatReady()` 的状态。

如果配置中禁止了接收功能，则此函数不可用。

4. 发射相关函数

```
void loadFormat(const bc7215FormatPkt_t& source);
```

加载格式数据到 BC7215 芯片。调用此函数后，会将指针 `source` 的格式信息数据加载到 BC7215 芯片。

如果配置中禁止了发射功能，则此函数不可用。

```
void irTx(const bc7215DataMaxPkt_t& source);
```

发射红外数据。调用此函数，会令 BC7215 通过红外线发送数据包 `source`。发送所使用的格式，为最后一次加载的格式，如果是从接收模式切换到发射模式，且没有加载过格式数据，则会使用最后一次接收到的红外信号的格式。如果待发送数据少于 16 字节（128 比特），此函数会

将数据写入 BC7215 片内的缓冲区后立即返回，如果多于 16 字节，则在最后 16 个字节被写入缓冲区后返回。

具体红外发射所需的时间，由所使用的红外调制格式决定，通常每个字节需几个 ms。用户可以通过 `cmdCompleted()` 函数查询红外发射是否完成。

如果配置中禁止了发射功能，则此函数不可用。

```
void sendRaw(const void* source, word size);
```

发送原始数据。此函数与 `irTx()` 类似，但 `source` 可以为任何类型的数据，而限于原始数据包格式，`size` 则为待发送的字节数。此函数适于用于数据通讯。

```
void setC56K(bc7215FormatPkt_t& target);
```

设置格式数据包特征字的控制位，设置后其中 C56K 位将置位，加载数据包后，BC7215 将使用 56K 载波发射。

```
void clrC56K(bc7215FormatPkt_t& target);
```

清除格式数据包特征字的控制位，设置后其中 C56K 位将被清除 (恢复缺省状态)，加载数据包后，BC7215 将使用 38K 载波发射。

```
void setNOCA(bc7215FormatPkt_t& target);
```

设置格式数据包特征字的控制位，设置后其中 NOCA 位将置位，加载数据包后，BC7215 输出将不使用载波，为纯高低电平输出。

```
void clrNOCA(bc7215FormatPkt_t& target);
```

设置格式数据包特征字的控制位，设置后其中 NOCA 位将被清除 (恢复缺省状态)，加载数据包后，BC7215 将使用 38k 或 56k 载波输出。

5. 工具函数

BC7215 类同时提供了一些工具函数，这些函数不直接操作 BC7215 芯片，而是用户在使用 BC7215 时经常会用到的一些操作。

```
byte crc8(byte* data, word len);
```

CRC8 计算函数。如果将 BC7215 应用于数据通讯，可能会需要给数据包加上 CRC 校验以提高可靠性，因为 BC7215 适宜使用的数据包一般较小，建议的数据长度在 16 个字节以内，而且因为红外通讯速率较低，因此使用 8 位 CRC 校验更符合需要，既能起到防止错误作用，又不会增加过多通讯时间。

函数有两个输入参数，第一个为指向数据的指针，第二个为需计算 CRC 的数据长度。这里的长度是指字节长度，而不是数据包中的位长度，本函数只能按字节计算 CRC。返回值为计算的 CRC 结果。CRC 计算的多项式，本驱动库默认为 0x07，在 `bc7215_config.h` 文件中定义，用户可根据需要修改。

```
word calSize(const bc7215DataMaxPkt_t& dataPkt);
```

计算数据包大小。这个函数返回 `dataPkt` 中的实际数据包的大小，包括原始数据部分和位长度部分，以字节为单位。

```
void copyDpkt(void* target, bc7215DataMaxPkt_t& source);
```

数据包拷贝函数。此函数将 `source` 的数据包拷贝到 `target` 的地址。拷贝操作将只拷贝 `source` 中实际数据包的大小。需要特别说明的是此函数支持 `source` 和 `target` 两个数据包重叠的情况，这种情况下实际上是完成了数据包的移动，比如 `target` 的地址可以仅比 `source` 相差 1 个字节，相当于将 `source` 数据包在内存里向前或者向后移动一个字节。

```
bool compareDpkt(byte sig, const bc7215DataMaxPkt_t& pkt1, const bc7215DataMaxPkt_t& pkt2);
```

比较两个数据包。此函数比较两个数据包中的有效数据是否相同。如果相同，函数返回 '1'(true)，如果不同，返回 '0'(false)。

红外数据的结尾可能不是完整的字节，此函数支持比较不完整的字节，因为不同的编码格式，数据有可能是 MSB 在前或者 LSB 在前，因此还需要特征字信息以确定数据的排列方向。函数假设两个数据包的特征字是相同的，如果两个数据包调制方式不同，其数据长度和数据很大可能也是不同的。

高级应用

本驱动库包含一个配置文件 `bc7215_config.h`，在 Arduino 根目录的 `libraries/bc7215` 目录中，其中定义了一些本驱动库有关的参数，高级用户使用前可根据需要做调整，使其更适合自己的项目需要。主要配置参数如下：

ENABLE_RECEIVING

这个参数控制是否使能驱动库的接收解码处理功能，包括接收原始数据包，接收格式信息包等。这个参数默认为 '1'，即使能状态，如果用户不需要 BC7215 的接收解码功能，可以将其修改为 '0'。禁止接收功能后，相关的函数也将变得不可用。因为接收解码功能占用了大多数所需的内存和程序内容，因此禁止接收功能，将大大缩小本驱动库的体积和所占用的内存。

ENABLE_FORMAT

在接收功能下，还有个进一步的控制参数，控制是否使能接收格式信息。默认值为 '1'，为使能状态，如果不需要，则可以改为 '0'。有些应用，如应用于数据通讯时，并不需要获取红外信号的格式信息，这时就可以将此功能关闭，因为格式信息包为 33 个字节，因此将节省 33 个字节的 RAM 和一部分程序空间。

ENABLE_TRANSMITTING

这个参数控制是否使能红外发射相关功能，默认为 '1' 使能状态，改为 '0' 可以关闭。关闭发射相关功能可以节省一部分程序空间。

BC7215_MAX_RX_DATA_SIZE

用字节数表示的可接收的最大原始数据包的长度，范围为 1-512，实际遥控器所发出的数据长度，影音设备一般在 8 字节以内，空调类的，一般在 48 字节以内。此定义值越大，本驱动库将占用更多的内存。

BC7215_CRC8_POLY

本函数库提供了一个 CRC-8 计算的函数，此参数定义了 CRC-8 计算的多项式值，默认值为 0x07，用户可根据自己需要修改。

应用例子

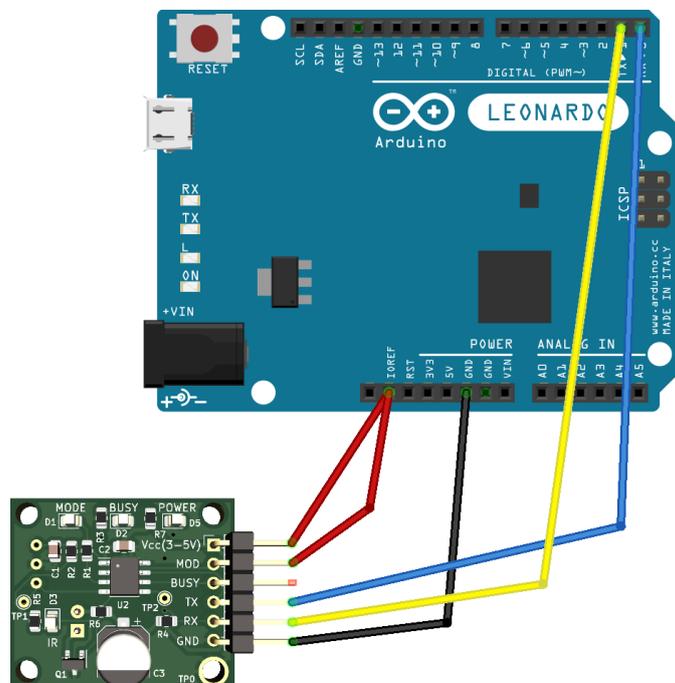
BC7215 Arduino 驱动库提供了 4 个应用的例子，分别为：

- 遥控器解码
将所接收到的遥控器信号的原始数据，输出到串口监视器。
- 学习型遥控器
一个具有 4 个按键的可分别学习不同品牌遥控器的学习型遥控器
- 2 路红外遥控开关
可以和任何遥控器“配对”，利用家中遥控器上的空闲按键遥控控制电源开关。
- 红外数据传输
利用 BC7215 作为红外收发器件，用作数据传输。

遥控器解码

这是一个简单的应用，BC7215 始终处于接收状态，BC7215 的 MOD 引脚固定连接到 VCC，而 BUSY 则悬空无连接。Arduino 将所接收到的红外遥控信号的原始数据显示在串口监视器中。此应用可应用于任何 Arduino 板，但要求使用的 Arduino 除了与电脑通讯的 Serial 口外，还有可用的串口供连接 BC7215。

应用连接图，以 Arduino Leonardo 为例：



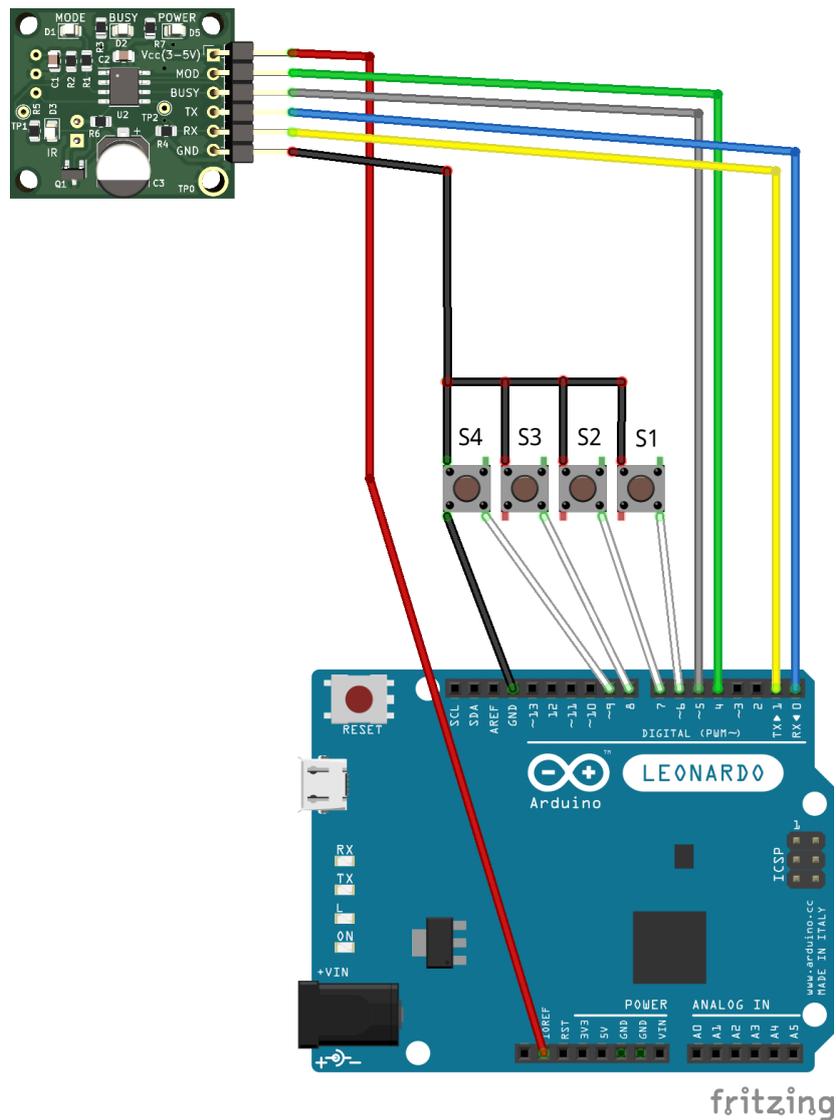
fritzing

学习型遥控器

本应用示例使用 Arduino 配合 BC7215 构成一个简单的具有 4 个按键的万能学习型红外遥控器，4 个按键可以分别学习为任意格式的遥控器按键，每个按键可以分别学习不同品牌型号的遥控器，控制不同的设备。

本例子可以用于任何 Arduino 板，如果使用仅有 1 个串口的板型如 UNO，要求在上传(upload)程序时，暂时断开 BC7215 的连接，避免串口冲突。如果与 BC7215 的连接使用 Serial 以外的串口，如 Serial1，则可以在电脑的串口监视器上看到一些辅助信息。例子使用板上的内置 LED（数字 I/O 口 13），如果其他 I/O 口连接 LED 可自行修改程序。

连接电路如下（以 Arduino Leonardo 板为例）：



示例程序功能描述

例子程序实现一个 4 键的学习型遥控器。程序有两个状态：发射模式和学习模式。上电后默认为发射模式，LED 约 1.6s 短暂闪烁一次，表示工作在发射模式。如果按键已经经过学习（上电后内容为空，按键无反应），按下按键，即发射所学习的遥控按键功能。

在发射模式下，同时按 S1 和 S2，则进入学习模式，此时程序控制 BC7215 进入接收状态，I/O 13 上的 LED 灯快速闪烁，提示用户此时处于学习状态。此时用户再选择按 S1-S4 中的一个键，表示学习内容将存入该键。选择按键后，LED 将变为长亮。在进入此状态后，用遥控器对准 BC7215 的红外接收头按下按键，如果 BC7215 接收解码成功，该红外遥控器的格式及所按按键的内容将被保存，演示程序重新回到发射模式，这时再按相同的按键，就会将所学习的按键复制发送出来。

演示程序设计为每个按键均对应有红外遥控格式和原始数据的存储，因此可以存储 4 个不同格式（不同厂家型号）的遥控信号。如果仅存储同一种格式的遥控器的不同按键，用户可以自行修改程序，改为仅保存一份格式信息，这样可以节省内存。同时，示例中为简单起见，原始数据使用了 4 个最大尺寸的原始数据包数组存储，如果用户希望节省内存，可以根据每个按键原始数据的长度，动态分配内存。

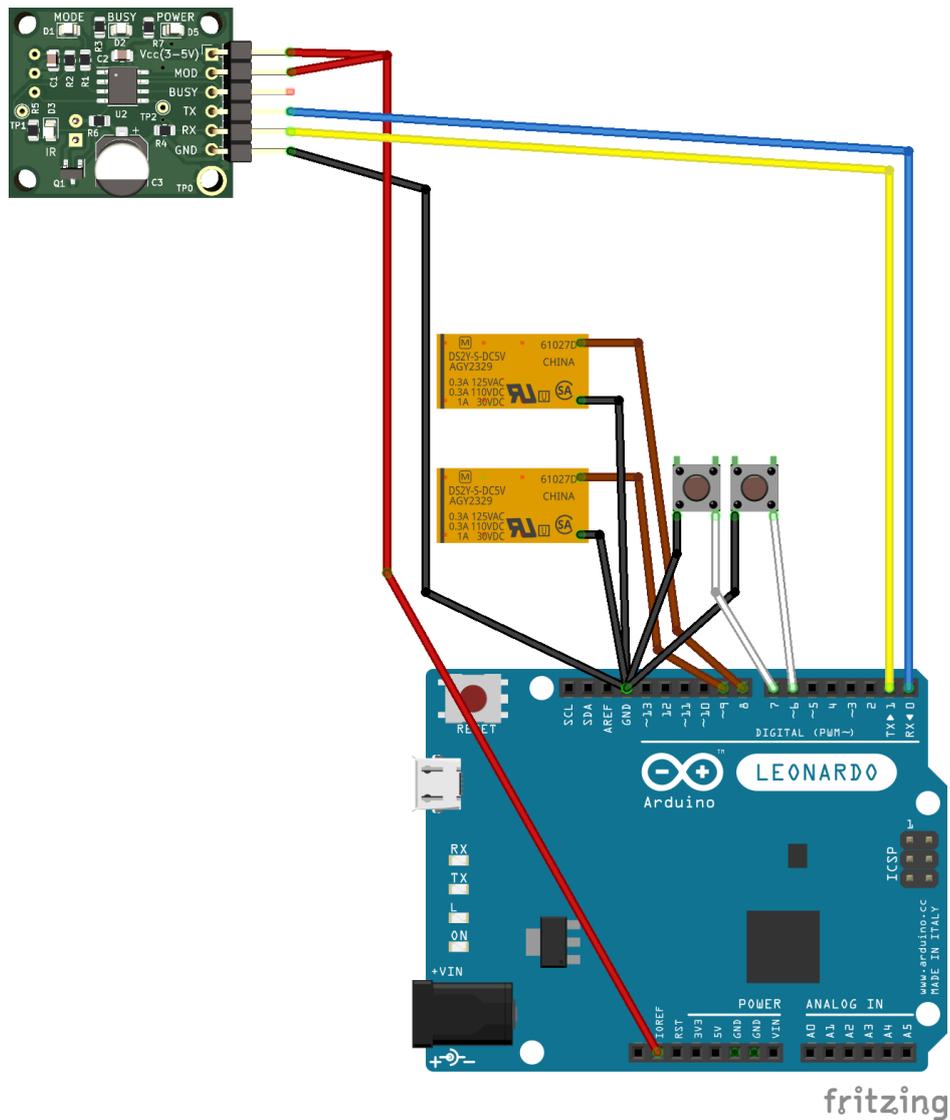
注意：某些格式的遥控器，如 RC5 格式和 RC6 格式，每次按键时，发出的数据中会有一个翻转位，每次按键该位均会翻转，而本学习遥控器程序为了简化起见，并未设置此功能，测试中对于 RC5/RC6 码的遥控器，如果接收设备会记忆上次接收到数据并期待翻转位的极性被翻转，使用本程序复制发送原数据可能会被接收设备拒绝，这取决于接收设备的设计，如果用户需要完全实现原遥控器的功能，需再加上对翻转位的翻转功能。

2 路红外遥控开关

此应用演示了利用 BC7215 构成一个通用的红外控制开关。每个家中均有各种各样的红外遥控器，而每个遥控器上一般均会有一些从不使用的按键，而这个 BC7215 的应用，可以利用这些不使用的红外遥控器按钮，作为红外遥控开关使用。本遥控插座可以使用任何遥控器的空闲按键作为控制开关。

本例子可以用于任何 Arduino 板，如果用在 UNO 以外有除了 Serial 口以外还有额外串口(如 Serial1)的板上，则可在电脑串口监视器中看到有关程序运行的一些额外信息。程序使用板载 LED(数字 I/O 13)作为 LED 显示输出，用户也可自行修改程序使用其它 I/O

连接电路图如下(以 Arduino Leonardo 板为例)



示例程序功能描述

例子程序实现一个 2 路的通用红外遥控开关，电路本身具有 2 个按键，分别控制 2 路红外开关的学习功能。每一路均可以学习记忆(配对)任意遥控器中的一个键，学习后当再收到这个键的红外信号时，即控制对应的继电器开关翻转，达到开关电器的目的。

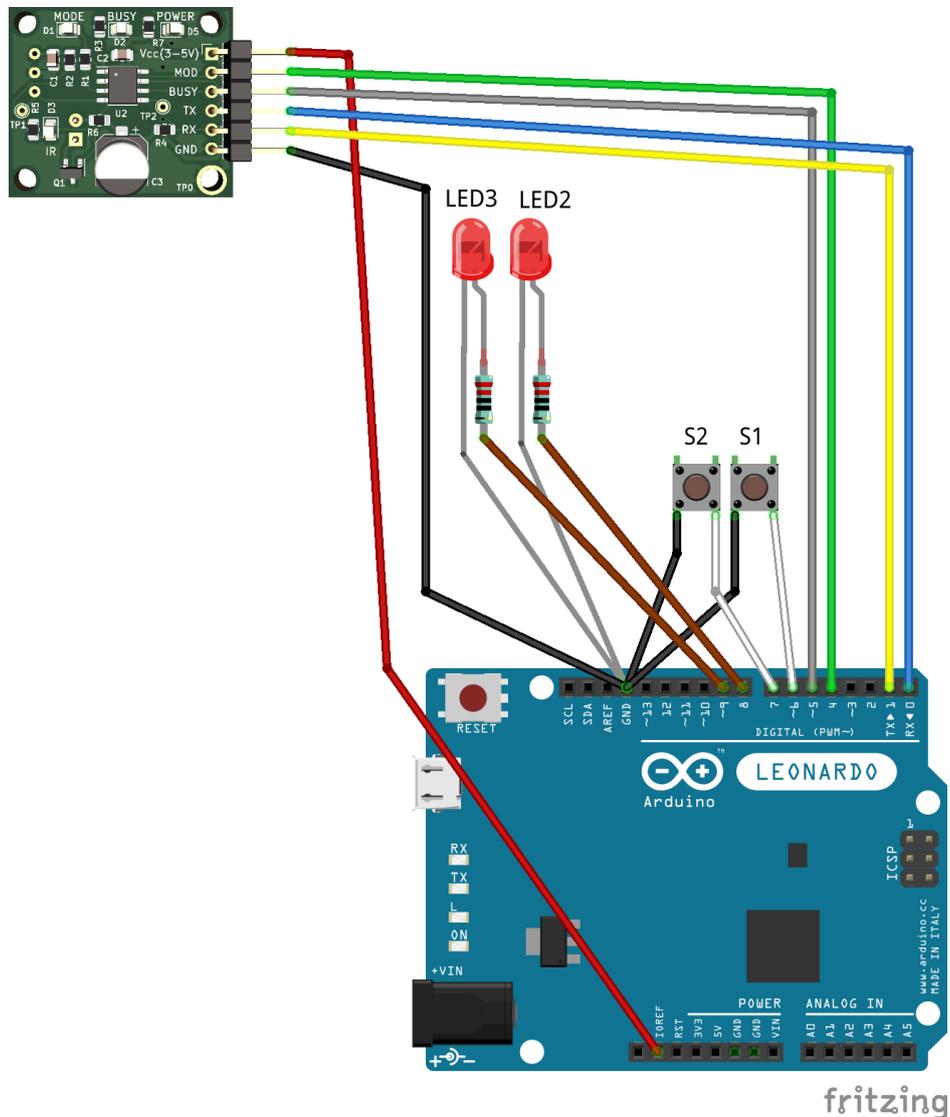
程序有两个状态：普通模式和学习模式。上电后默认为普通模式，LED 约 3.2s 短暂闪烁一次，表示工作在普通模式。当用户按其中一个键后，LED 灯常亮，表示进入学习（配对）状态，此时用遥控器对本机发送红外信号，即可将该信号的信息保存，程序返回正常模式。此后如果再收到同一个按键的红外信号，则会控制对应的继电器翻转。

红外数据传输

此应用演示了使用 BC7215 作为红外收发器件，将其应用于红外数据传输的用法。红外遥控器作为广泛使用的电子设备，其通讯可靠性已经过验证，但其通讯速率比较低，适宜作为少量数据对速度无要求时的传输，如传感器数据，指令数据等。例子中演示了使用 CRC 校验和不使用 CRC 校验两种方式。

本例子可以用于任何 Arduino 板，如果用在 UNO 以外有除了 Serial 口以外还有额外串口(如 Serial1)的板上，则可在电脑串口监视器中看到有关程序运行的一些额外信息。程序使用板载 LED(数字 I/O 13)和另外两个 LED(连接于数字 I/O 8 和 9)作为指示输出，用户也可自行修改程序使用其它 I/O

连接电路图如下(以 Arduino Leonardo 板为例)



本例子中使用了 NEC 编码作为红外传输的编码格式(详情见 BC7215 数据手册)。

示例程序功能描述

本示例仅为演示数据传输功能，因此并无实际用途。系统由 2 个按键和 3 只 LED 组成。分别为

板载 LED——工作指示

LED2——接收数据 CRC 正确指示

LED3——接收数据 CRC 错误指示

上电后，板载 LED 每隔 3.2 秒闪烁一次，表明系统在工作，系统默认处在接收状态。

如果按下 S1 键，系统会从红外发送一个 21 字节的数据（例子固定为 0-20 的连续数字），传送完数值数据后，再传送一段 ASCII 码数据，内容为“Hello World”，两段数据的末尾均添加了 CRC-8 校验值。在红外传输期间，板载 LED 会点亮。使用第一个遥控器解码例子，在另一个 Arduino 的串口监视器中，可看到接收到的数据。或者如果使用 Windows BC7215 演示软件，选择通讯演示功能，配合 BC7215 演示板，即可在接收电脑上看到 Arduino 输出的数据。当软件选择 TEXT 模式时，上面的 21 字节数据因为是不可显示字符，所以没有显示，而文字部分则可以看到还原的文字。当软件选择 HEX 模式时，则可以看到两条数据信息，分别为 22 字节和 12 字节，末尾最后一个字节是 CRC 字节。

如果按下 S2 键，系统发送同样的数据，但末尾不添加 CRC-8 校验值。

没有按键时，系统工作在接收模式。此时如果接收到红外信号，不管内容是什么，程序都会对其进行 CRC 校验，如果校验正确，会令 LED2 闪烁一下，如果校验不正确，会令 LED3 闪烁一下。用户可以用任何普通遥控器测试，因为普通遥控器不带 CRC 校验，因此将看到 LED3 闪烁，如果用 PC 上的 BC7215 演示软件发送带 CRC 校验的数据，或者另一个同样的电路按 S1 键，则可以看到 LED2 闪烁。