## 七 模块实战

一 硬件介绍 2

主控板正面 2

主控板背面 3

二 软件介绍 5

软件界面 5

软件特色 6

三 通用传感器 6

1 GPIO类 6

2 I2c传感器 14

3 UART接口 22

4 SPI接口 27

6 PWM接口 30

7 ADC采样 33

8时钟节拍 35

四 实用模块 37

1系统函数 37

2 I/O类模块 38

3 UART模块 41

4 I2c模块 42

5 SPI模块 45

6其他类模块 46

7常用关键字 50

8程序概念 53

五 综合案例 54

案例一 8421灯实验 54

案例二触摸开关 57

案例三恒温器 57

案例四跟屁虫 57

案例五自动窗帘 58

案例六声控灯 58

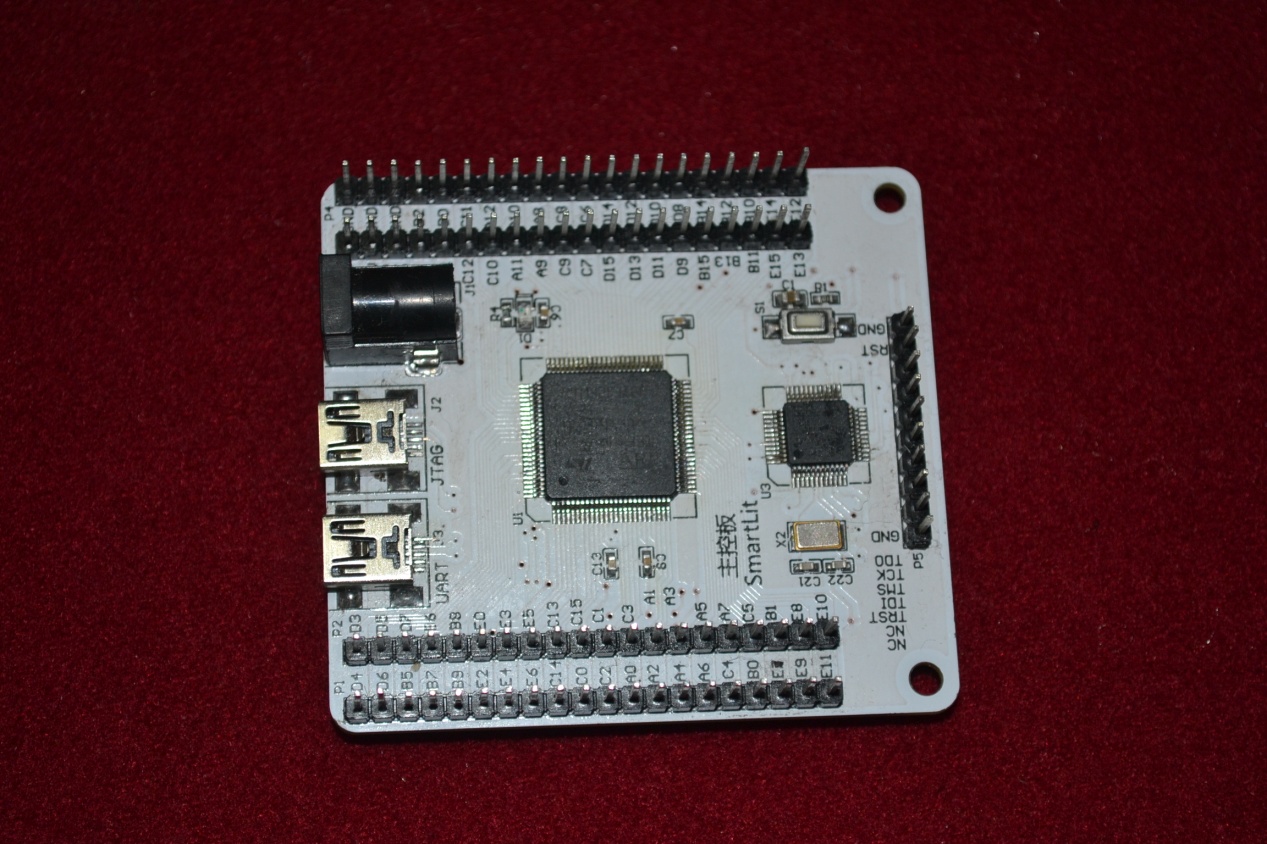
案例七 时钟 58

案例八 自动浇灌系统 59

案例九 自动种植蘑菇机系统 61

## 一 硬件介绍

### **主控板正面**



复位键

电源接口

JTAG下载

UART输出

STM32F103VCT6芯片

STM32F103VC8T6仿真

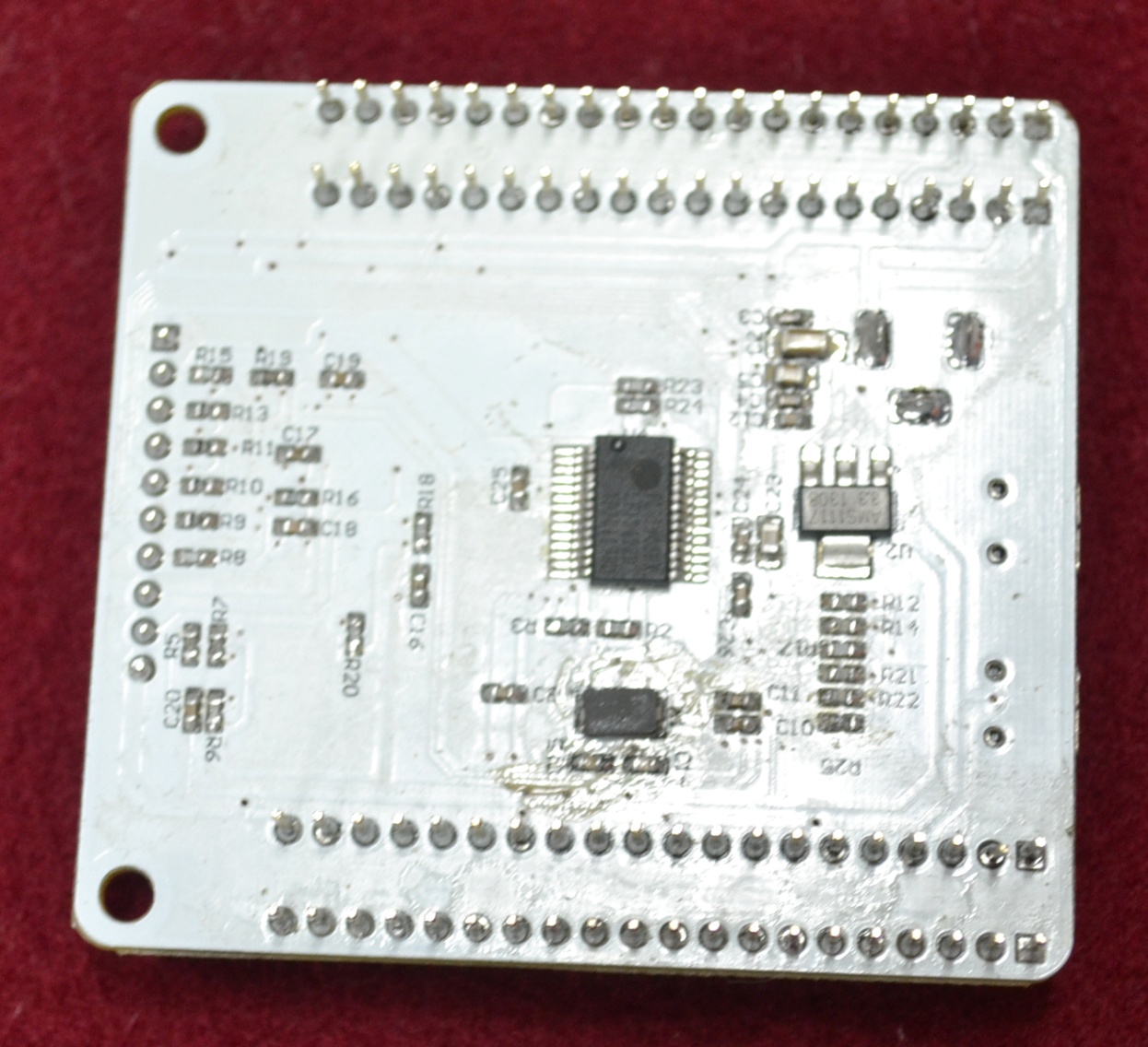
芯片

**NCP1117**

芯片

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 名称 | 备注 | |
| 1 | STM32F103VCT6芯片 |  | |
| 2 | STM32F103VC8T6仿真 |  | |
| 3 | 白色复位按键 |  | |
| 4 | 电源接口 |  | |
| 5 | UART输出 |  | |
| 6 | JTAG下载 |  | |
| 编号 | 管脚 | | 备注 |
| P1 | D4、D6、B5、B7、B9、E2、E4、E6、C14、C0、C2、A0、A2、A6、C4、B0、E7、E9、E11 | | C0模拟输出管脚, 标注有△ |
| P2 | D3、D6、D7、B6、B8、E0、E3、E5、C13、C15、C1、C3、A1、A3、A5、A7、C5、B1、E8、E10 | |  |
| P3 | 8v、8v、3v3、3v3、D1、C12、C1、A11、A9、C9、C7、D15、D13、D11、D9、B15、D9、B15、B13、B11、E15、E13 | | 两个5v电压端口,两个3.3v电压端口(在主控板上标识为3v3)。C9、C7 |
| P4 | GND、GND、GND、D2、D0、C11、A12、A10、A8、C8、C6、D14、D12、D10、D8、B14、B12、B10、E14、E12 | | 三个地线GND。C8、C6 |
| P5 | DND、NC、NC、TRST、TDI、TMS、TCK、TDO、RST、GND | |  |

### **主控板背面**



**晶振**

**串口转换**

稳压芯片

|  |  |  |
| --- | --- | --- |
| **编号** | **名称** | **备注** |
| **1** | **PL2303HXD串口转换** |  |
| **2** | **晶振** |  |
| **3** | **稳压芯片NCP1117** |  |

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

特别声明：

为了在使用上方便，我们特别设定了串口管脚。

1串口管脚：UART中断只能接受中断。

1.1 Uart1：TXD = A9,RXD =A10

1.2 Uart2：TXD=A2,RXD=A3

1.3 Uart3：TXD=B10,RXD=B11

1.4 Uart4：TXD=C10,RXD=C11

1.5 Uart5：TXD=C12,RXD=D2

2 SPI管脚定义：

2.1 SPI:CS=A4,SCK=A5,MIS=A6,MOSI=A7。

2.2 SPI2:CS=B12,SCK=B13,MISO=B14,MOSI=B15。

3 定时器输出通道：TIM8

3.1 PWM:CHI~CH4:C6、C7、C8、C9。（4个通道）

4 非FT管脚：（FT意思是能容忍5v电压）

4.1只能使用3.3v A0~A7(ADC)、B0~B2(0、1ADC)、B10~B11、B0~C5(ADC)、C13~C15。

PC0作为ADC采样脚。其它管脚使用5v。

D0数字输出（0、1），输出的电压因此要么是0，要么就是输出所给电压。比如光敏电阻需要5v电压，那么，我们给它供5v电压，D0输出的电压要么是0v，要么就是5v，这时D0需要连接5v管脚；要求5v电压的光敏电阻，如果给它9v电压，D0输出的电压就是0v或是9v，这时D0所连接管脚，很有可能由于电压过大而烧毁主控芯片。

A0,模拟输出的电压范围是0~3.3V之间变化，因而A0管脚，必须与特定的管脚连接。我们的主控板上管脚C0，标注有△，表示模拟信号。

（D0，A0,所指是数字输出模拟输出缩写，主控板是排针编号。）

5.中断固定管脚，如果是用0中断，管脚对应A0,B0,C0,D0,E0;如果是用1中断，管脚对应A1,B1,C1,D1,E1,以此类推。总共0~4个中断。

## 二 软件介绍

### **软件界面**



编译

下载

通用模块

实用模块

代码编译区

### **软件特色**

1 界面简化，所用按键极少，使用方便；

2 拥有通用模块和实用模块，降低编译难度，提高效率；

3 项目功能系统化模块化，解放思维，关注于创新，而不是技术细节。

## 三 通用传感器

本教程将在编译所需命令列出通用模块，使用通用模块，编译我们知道的传感器；实用模块中，我们列出我们编译好的传感器模块，今后我们也会根据需要不断添加，使用这些模块将搭建产品原型变得轻松快捷。

所有的传感器使用遵循原则，

首先，初始化传感器，所指定的管脚号具有功能状态，

然后，确定是否是输出还是输入，

最后，得出结果。

### **1 GPIO类**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

接受信号时，用输入input，发出信号，用输出output。

GPIO类，即我们所说的开关接口，I/O开关。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

编译所需命令：

* + void GPIO\_Set\_Output("管脚号"); //设置管脚为输出模式
  + void GPIO\_Set\_Input("管脚号");//设置管脚为输入模式
  + void GPIO\_Set\_High("管脚号");//设置管脚为高电平
  + void GPIO\_Set\_Low("管脚号");//设置管脚为低电平
  + u8 GPIO\_Get\_Value("管脚号");//获取管脚上的电平状态

编译所需案例：

第一步 初始化GPIO

* + void GPIO\_Set\_Output("管脚号"); //设置管脚为输出模式
  + void GPIO\_Set\_Input("管脚号");//设置管脚为输入模式

第二步 判定GPIO的开关状态

* + void GPIO\_Set\_High("管脚号");//设置管脚为高电平
  + void GPIO\_Set\_Low("管脚号");//设置管脚为低电平
  + u8 GPIO\_Get\_Value("管脚号");//获取管脚上的电平状态

这是每块硬件里已经下载好的程序Hello world!

1.1 hello world

/\*参数定义区和头文件包含区\*/

int main(void)//每个程序必须有一个主函数

{

Hard\_Init();//系统必须初始化部分，请勿删除

while(1)

{

Dprintf("Hello world!\r\n");

Delay\_ms(1000);//延时1000毫秒

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1秒=1000毫秒，1毫秒=1000微妙。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**1.2 点亮LED**

第一步，构建主函数

int main（）

｛

｝

第二步，设置状态

int main（）

｛

**User\_GPIO\_Set\_Output("管脚号"); //获得状态**

｝

第三步，点亮LED

int main（）

｛

User\_GPIO\_Set\_Output("管脚号"); //获得状态

**User\_GPIO\_Set\_High("管脚号"); //点亮led**

｝

第四步，添加循环语句，使灯一直亮

int main（）

｛

User\_GPIO\_Set\_Output("管脚号"); //获得状态

**while(1)**

**{**

**User\_GPIO\_Set\_High("管脚号"); //点亮led**

**}**

｝

第五步，使灯一直闪烁

int main( )

｛

User\_GPIO\_Set\_Output("管脚号"); //获得状态

while(1)

｛

**User\_GPIO\_Set\_High("管脚号"); //点亮led**

**Delay(1000);//延时42毫秒，1000毫秒=1秒**

**User\_GPIO\_Set\_Low("管脚号"); //熄灭led**

**Delay(1000);//延时42毫秒**

｝

｝

第六步，指定管脚号

int main()

{

User\_GPIO\_Set\_Output(E13); //获得状态

while(1)

{

User\_GPIO\_Set\_High(E13); //点亮led

Delay\_ms(1000);//延时1000毫秒

User\_GPIO\_Set\_Low(E13); //熄灭led

Delay\_ms(1000);//延时1000毫秒

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

传感器连接方式：

GND接地线，9V/5V/3V3所接电压(3V3的意思是3.3V电压)，管脚号指主控板上的编号。比如上面案例指定管脚号E13，即指定传感器与主控板连接的管脚。

在传感器上，NC表示无功能管脚，可以不去理会。按照提示连接传感器和主控板。注意保持一致!

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**1.3 扩展思考：**

要使两盏LED顺序亮一下熄灭一下，该怎样做？程序参考如下：

/\*参数定义区和头文件包含区\*/

int main(void)//每个程序必须有一个主函数

{

Hard\_Init();//系统必须初始化部分，请勿删除

User\_GPIO\_Set\_Output(E13);//初始化E13

User\_GPIO\_Set\_Output(E15);//初始化E15

while(1)

{

User\_GPIO\_Set\_High(E13);

Delay\_ms(1000);//延时1000毫秒

User\_GPIO\_Set\_Low(E13);

Delay\_ms(1000);//延时1000毫秒

User\_GPIO\_Set\_High(E15);

Delay\_ms(1000);//延时1000毫秒

User\_GPIO\_Set\_Low(E15);

Delay\_ms(1000);//延时1000毫秒

}

}

**1.4，蜂鸣器**

**第一步 构建主函数**

**/\*参数定义区和头文件包含区\*/**

**int main(void)//每个程序必须有一个主函数**

**{**

**Hard\_init();//系统必须初始化部分，请勿删除**

**User\_GPIO\_Set\_Output("管脚号"); //状态**

**while(1)**

**{**

**User\_GPIO\_Set\_High("管脚号");//蜂鸣器响**

**Delay\_ms(1000);//延时1000毫秒**

**User\_GPIO\_Set\_Low("管脚号"); //蜂鸣器停**

**Delay\_ms(1000);//延时1000毫秒**

**}**

**}**

第二步 确定管脚号

**/\*参数定义区和头文件包含区\*/**

**int main(void)//每个程序必须有一个主函数**

**{**

**Hard\_Init();//系统必须初始化部分，请勿删除**

**User\_GPIO\_Set\_Output(E13);**

**while(1)**

**{**

**User\_GPIO\_Set\_High(E13);**

**Delay\_ms(1000);//延时1000毫秒**

**User\_GPIO\_Set\_Low(E13);**

**Delay\_ms(1000);//延时1000毫秒**

**}**

**}**

**1.5 光敏电阻**

/\*参数定义区和头文件包含区\*/

u8 lightstatus ；

int main(void)//每个程序必须有一个主函数

{

Hard\_init();//系统必须初始化部分，请勿删除

User\_GPIO\_Set\_Input(E1);

User\_GPIO\_Set\_Output(E2);

while(1)

{

lightstatus = User\_GPIO\_Get\_Value(E1);

if（lightstatus == 1）

｛

User\_GPIO\_Set\_High(E2);

｝

else

｛

User\_GPIO\_Set\_Low(E2);

｝

}

}

**1.6 显示获取的数值**

U8 a;//声明8位变量

//U8 User\_Get\_Value(E3);

a=User\_Get\_Value(E3);//获取的值，赋给a变量

Dprintf（“ %d \r\n”，a）；//把a整数值打印出来

Delay\_ms(1000);//延时1000毫秒

获得信号，你可以决定获得的值到底想用来做什么。

**1.7 电机驱动**

int main(void)//每个程序必须有一个主函数

{

Hard\_Init();//系统必须初始化部分，请勿删除

User\_GPIO\_Set\_Output(E13);//初始化管脚

User\_GPIO\_Set\_Output(E15);

User\_GPIO\_Set\_Output(B11);// ENABLE电机芯片

User\_GPIO\_Set\_High(E13);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

功能：电机正反转

描述：

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

while(1)

{

User\_GPIO\_Set\_High(E15);

User\_GPIO\_Set\_Low(B11);

Delay\_ms(1000);//延时1000毫秒

User\_GPIO\_Set\_Low(E15);

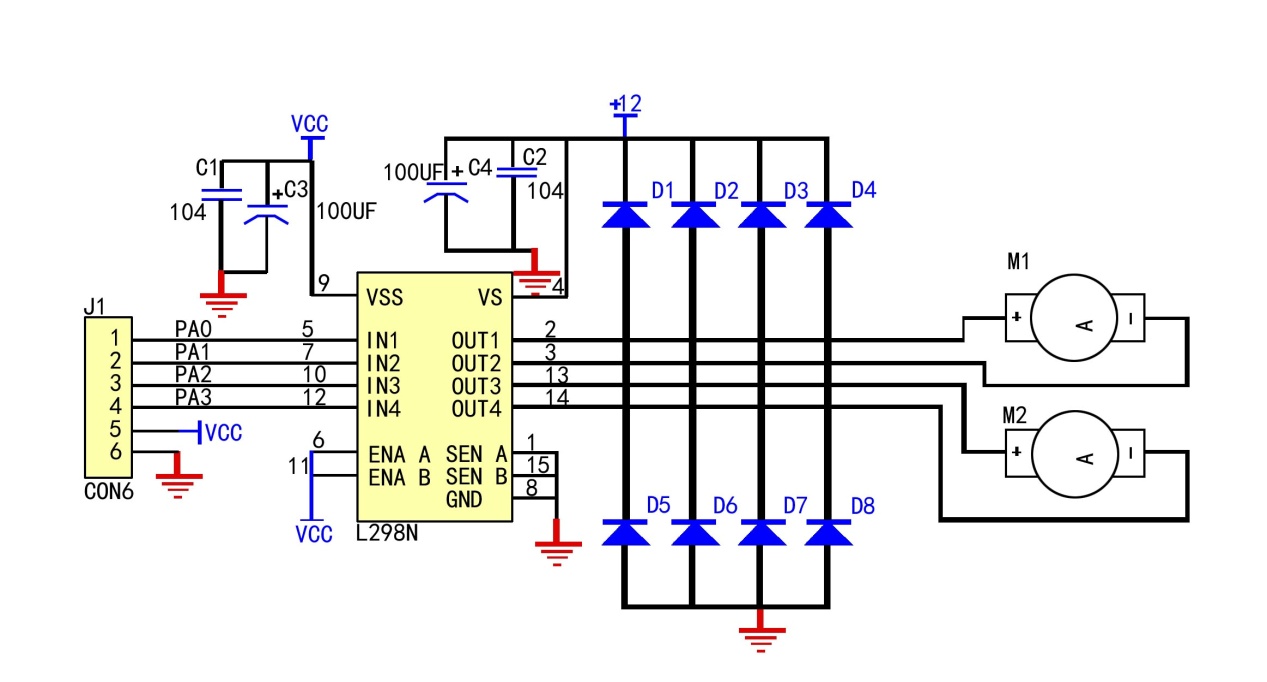
User\_GPIO\_Set\_High(B11);

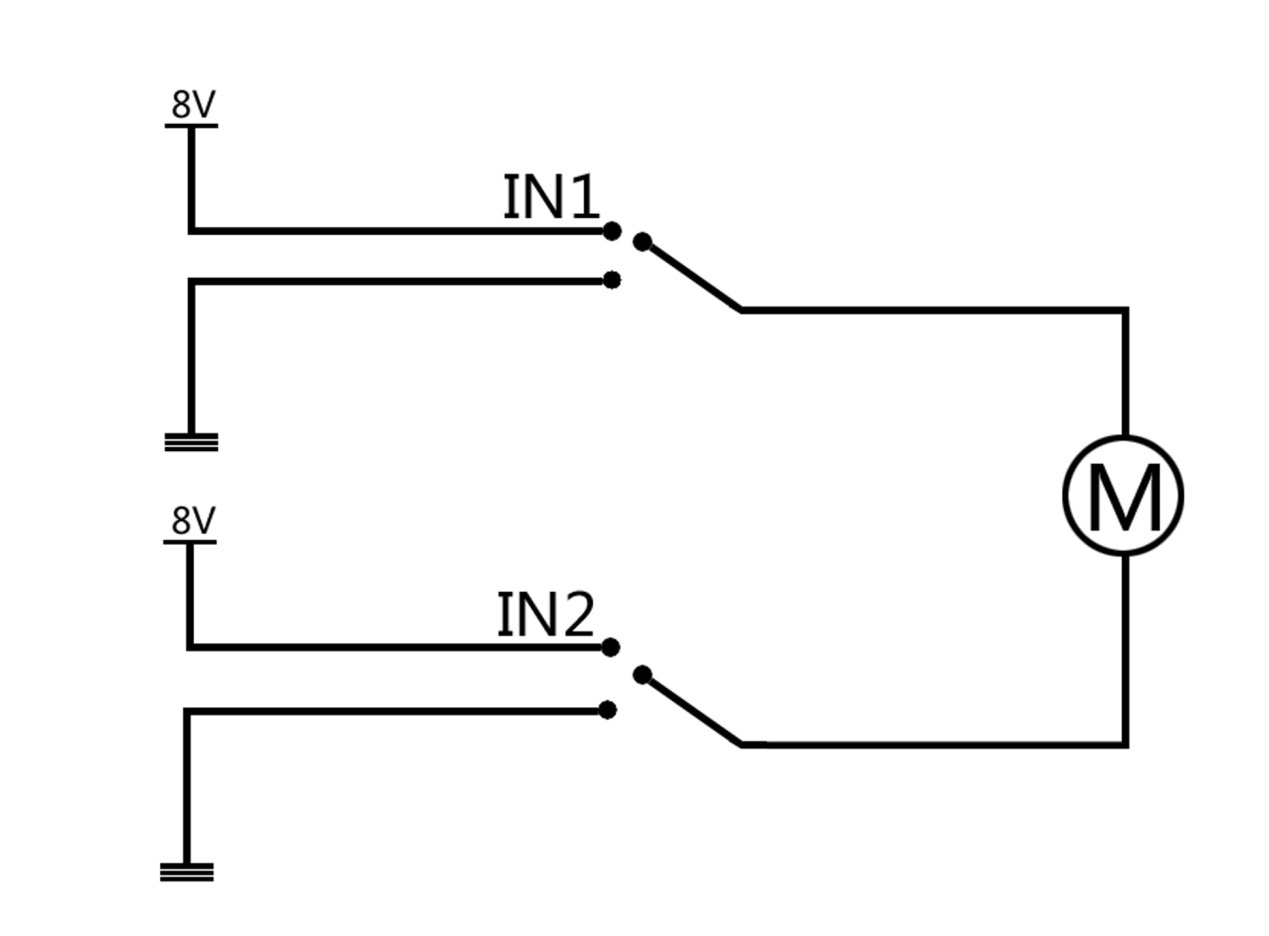
Delay\_ms(1000);//延时1000毫秒

}

}

电机原理图





**电机正反转示意图**

**控制电机，第一步，打开主控板控制电机的管脚，第二步使能电机芯片，第三步确定IN1，IN2输出电源正负，实现电机正反转的功能。**

电机驱动能控制两个电机。GND 接地线，IN1,IN2（IN3,IN4）输出端，5v给芯片供电，M12v参考供电（供电极限可以达到30v），给电机供电，这时需要P3口短路（如果需要外部供电，就需要把P3短路冒拔出，从P1处外面供电），M1,M2（M3,M4）分别连接电机正负极。

### 2 I2c传感器

当你看到硬件上，有SDA和SCL接口，这类传感器即是I2C类。SDA(Serial Data,串行数据)和SCL(Serial Clock,串行时钟)，定义SCL接口和SDA接口，这两个管脚可以互用，我们只要记住，你定义的管脚号是对应哪一个管脚即可。I2C编号，在一个程序中，考虑要用几个I2C，因而我们提供了7个（避免程序冲突），你需要填写你要用的I2C编号即可（编号要用下划线），比如I2C\_1。

I2C的规律是开始， 指定设备地址，然后数据交换，结束，好比老师开始叫学号，听到学号的同学对上号，按老师要求干活，然后结束，这就是I2C的运作方式。

编译所需命令：

* + Io\_I2c\_Gpio\_Init("I2C编号", "SCL管脚", "SDA管脚");
  + I2c\_Single\_Write("I2C编号","设备地址","寄存器地址","数据");
  + I2c\_Single\_Read("I2C编号","设备地址","寄存器地址","数据");
  + I2c\_Multiple\_Write("I2C编号", "设备地址"," 寄存器地址","数据长度","数据缓存区");
  + I2c\_Multiple\_Read("I2C编号","设备地址","寄存器地址","数据长度","数据缓存区");
  + Io\_I2c\_Start ("I2C编号");
  + Io\_I2c\_Stop ("I2C编号");
  + "(u8)Data"=Io\_I2c\_Recv\_ACK("I2C编号");
  + Io\_I2c\_Send\_ACK ("I2C编号", "应答类型");
  + Io\_I2c\_Send\_Byte ("I2C编号", "数据");
  + "(u8)Data"=Io\_I2c\_Recv\_Byte ("I2C编号");

所需命令解释：

* + Io\_I2c\_Gpio\_Init("I2C编号", "SCL管脚", "SDA管脚");

1.初始化I2C设备，并指定任意两个管脚为SCL、SDA信号线,其中设备编号最大为I2C\_3。

2.例如:Io\_I2c\_Gpio\_Init(I2C\_1,A0,A1)。/\*指定A0为SCL,A1为SDA,并将这两个管脚连在I2C\_1上\*/

* + I2c\_Single\_Write("I2C编号","设备地址","寄存器地址","数据");

1.写一字节到指定地址的I2C设备中。

2.例如:I2c\_Single\_Write(I2C\_1, 0x56, 0x01,Data)

* + I2c\_Single\_Read("I2C编号","设备地址","寄存器地址","数据");

1.读一字节从指定地址的I2C设备中。

2.例如:I2c\_Single\_Read(I2C\_1, 0x56, 0x01,&Data);

* + I2c\_Multiple\_Write("I2C编号", "设备地址"," 寄存器地址","数据长度","数据缓存区");

1.写多个字节数据到指定地址的I2C设备中。

2.例如:I2c\_Multiple\_Write(I2C\_1, 0x56, 0x01, 3, Buf);/\*其中Buf应该定义成：u8 Buf[3]。在0x56地址开始处写了3个字节\*/

* + I2c\_Multiple\_Read("I2C编号","设备地址","寄存器地址","数据长度","数据缓存区");

1.读多个字节数据从指定地址的I2C设备中。

2.例如:I2c\_Multiple\_Read(I2C\_1, 0x56, 0x01, 3, Buf);/\*其中Buf应该定义成：u8 Buf[3]。在0x56地址开始处读了3个字节\*/

* + Io\_I2c\_Start ("I2C编号");

1.指定相应I2C通道发出开始信号。

2.例如:Io\_I2c\_Start (I2C\_1);

* + Io\_I2c\_Stop ("I2C编号");

1.指定相应I2C通道发出结束信号。

2.例如:Io\_I2c\_Stop (I2C\_1);

* + "(u8)Data"=Io\_I2c\_Recv\_ACK("I2C编号");

1.指定相应I2C通道接收设备的应答信号。

2.例如:Val=Io\_I2c\_Recv\_ACK(I2C\_1);

* + Io\_I2c\_Send\_ACK ("I2C编号", "应答类型");

1.指定相应I2C通道发出应答或非应答信号。

2.例如:Io\_I2c\_Send\_ACK (I2C\_1, ACK);/\*或NACK\*/

* + Io\_I2c\_Send\_Byte ("I2C编号", "数据")

1.指定相应I2C通道发送一字节数据。

2.例如:Io\_I2c\_Send\_Byte (I2C\_1, Data);

* + "(u8)Data"=Io\_I2c\_Recv\_Byte ("I2C编号");

1.指定相应I2C通道接收一字节数据。

2.例如:Data=Io\_I2c\_Recv\_Byte (I2C\_1);

编译所需案例：

第一步 初始化I2c（我们的系统有7个I2C编号）

Io\_I2c\_Gpio\_Init("I2C编号");

第二步 写入数据或读取数据

I2c\_Multiple\_Write("I2C编号", "设备地址"," 寄存器地址","数据长度","数据缓存区");

我们会在传感器上注明他们的地址是多少，你只要按提示的填写在程序中就可以了。

"I2C编号",我们提供了7个（避免程序冲突），你需要填写你要用的I2C编号即可（编号要用下划线），比如I2C\_1。

"设备地址",这是根据芯片厂商资料手册给出。

"寄存器地址",这是根据芯片厂商资料手册给出。

"数据长度",为根据实际需要读写的数据长度来写，相当于房间的个数。

"数据缓存区"，为需要用户定义的存储空间，相当于预定入住的人数，根据实际情况定义。

“应答类型”

"数据"，一个数

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**2.1 电子罗盘**

第一步 建立主函数

/\*参数定义区和头文件包含区\*/

int main(void)//每个程序必须有一个主函数

{

Hard\_Init();//系统必须初始化部分，请勿删除

while(1)

{

"do sth;"

}

}

第二步 初始化，双击电子方位仪

/\*参数定义区和头文件包含区\*/

int main(void)//每个程序必须有一个主函数

{

Hard\_Init();//系统必须初始化部分，请勿删除

u16 Data;

Io\_I2c\_Gpio\_Init("I2C编号", "SCL管脚", "SDA管脚");//初始化I2C

Hmc5883\_Init("I2C编号","设备地址");

Data=Hmc5883\_Read\_Angle();//结果为0~360度的值，依据这个值来判别方向。

while(1)

{

"do sth;"

}

}

第三步 调整位置

/\*参数定义区和头文件包含区\*/

u16 Data;

int main(void)//每个程序必须有一个主函数

{

Hard\_Init();//系统必须初始化部分，请勿删除

Io\_I2c\_Gpio\_Init("I2C编号", "SCL管脚", "SDA管脚");//初始化I2C

while(1)

{

Hmc5883\_Init("I2C编号","设备地址");

Data=Hmc5883\_Read\_Angle();//结果为0~360度的值，依据这个值来判别方向。

}

}

第四步

/\*参数定义区和头文件包含区\*/

u16 Data;

int main(void)//每个程序必须有一个主函数

{

Hard\_Init();//系统必须初始化部分，请勿删除

Io\_I2c\_Gpio\_Init(I2C\_1,E11,E10);//初始化I2C

while(1)

{

Hmc5883\_Init(I2C\_1,0x3c);

Data=Hmc5883\_Read\_Angle();//结果为0~360度的值，依据这个值来判别方向。

}

}

第五步添加延时

/\*参数定义区和头文件包含区\*/

u16 Data;

int main(void)//每个程序必须有一个主函数

{

Hard\_Init();//系统必须初始化部分，请勿删除

Io\_I2c\_Gpio\_Init(I2C\_1,E11,E10);//初始化I2C

while(1)

{

Hmc5883\_Init(I2C\_1,0x3c);

Data=Hmc5883\_Read\_Angle();//结果为0~360度的值，依据这个值来判别方向。

**Delay\_ms(1000);//延时1000毫秒**

}

}

第六步 添加打印

/\*参数定义区和头文件包含区\*/

u16 Data;

int main(void)//每个程序必须有一个主函数

{

Hard\_Init();//系统必须初始化部分，请勿删除

Io\_I2c\_Gpio\_Init(I2C\_1,E11,E10);//初始化I2C

while(1)

{

Hmc5883\_Init(I2C\_1,0x3c);

Data=Hmc5883\_Read\_Angle();//结果为0~360度的值，依据这个值来判别方向。

**Dprintf("角度是： %d\r\n"");//\r\n为回车换行，%d表示整数**

Delay\_ms(1000);//延时1000毫秒

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

如果没有延时命令，程序或电脑由于数据的快速交换，而处于崩溃，所以在程序中需要添加延时命令。再说，电流的速度接近光速，能量还未点亮灯，就执行下一条命令！

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

2.2红外遥控器

u8 a,b;

Gpio\_Irda\_Init( );//初始化

While（1）｛

a =Irdo\_Get\_Value( );//获取信息

If（a ! b）｛

b=a;

Dprintf(“Irda = 0x %x \r\n”,a);

}

//如果a不等于b，那么，把b值赋给a，输出所按的b值（%x表示16进制）

If(a==0x30){

User\_Set\_High(G1);

｝

｝

2.3 继电器

Relay\_Gpio\_Init( );//初始化

If（a ==0x30）｛

Relay\_on(Relay2);

｝

If(a==0x18){

Relay\_off(Relay2);

}

2.4 控制灯开关

/\*参数定义区和头文件包含区\*/

u8 Data;

int main(void)//每个程序必须有一个主函数

{

Hard\_Init();//系统必须初始化部分，请勿删除

Relay\_Gpio\_Init ("管脚","管脚");//初始化指定管脚为继电器控制脚。

while(1)

{

Gpio\_Irda\_Init("中断管脚");

Irda\_Irq\_Isr();//红外接收中断服务程序

Data=Irda\_Get\_value();//获取遥控的按键值。

if（Data==0x45）

{

Relay\_On ("继电器通道");//打开指定通道继电器，通道取值范围RELAY1、RELAY2。

}

if（Data==0x46）

{

Relay\_Off ("继电器通道");//关闭指定通道继电器，通道取值范围RELAY1、RELAY2。

}

}

}

### 3 UART接口

编译所需命令：

Uart\_Init("串口编号", "串口波特率");

Uart\_Send\_Byte("串口编号","发送的数据");

Uart\_Send\_Array("串口编号"," 数据缓存区", "数据长度");

所需命令解释：

* + Uart\_Init("串口编号", "串口波特率");

1.初始化串口，其中串口标号最大为COM5。

2.串口为固定管脚，其中定义如下

COM1:TXD=A9,RXD=A10

COM2:TXD=A2,RXD=A3

COM3:TXD=B10,RXD=B11

COM4:TXD=C10,RXD=C11

COM5:TXD=C12,RXD=D2

3.例如:

Uart\_Init(COM1, 57600);/\*将COM1初始化成波特率为57600。常用波特率有4800,9600,19200,38400,43000,56000,57600,115200\*/

* + Uart\_Send\_Byte("串口编号","发送的数据");

1.通过串口发送一个字节数据。

2.例如:

Uart\_Send\_Byte(COM1,0x56);/\*将0x56通过COM1发送\*/

* + Uart\_Send\_Array("串口编号"," 数据缓存区", "数据长度");

1.通过串口发送指定长度字节数据。

2例如:

Uart\_Send\_Array("COM1,Buf,5);/\*其中Buf定义成:u8 Buf[5]\*/

编译所需案例：

第一步 初始化

Uart\_Init("串口编号", "串口波特率");

第二步 发送数据

U8 buf[10]//buf空房个数（长度）。

Uart\_Send\_Byte("串口编号","发送的数据");

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

注意事项：

COM4固定为打印数据，在超级终端中打开。需要插上USB线到UART的USB的接口，并在电脑设备管理器中确定该USB转串口在电脑中为哪个端口。\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

3.1 通过串口发送数据

/\*参数定义区和头文件包含区\*/

int main(void)//每个程序必须有一个主函数

{

Hard\_Init();//系统必须初始化部分，请勿删除

while(1)

{

Uart\_Send\_Byte(COM4,0X45);

Delay\_ms(1000);//延时1000毫秒

}

}

3.2 通过串口发送数据

/\*参数定义区和头文件包含区\*/

u8 buf[5];

int main(void)//每个程序必须有一个主函数

{

Hard\_Init();//系统必须初始化部分，请勿删除

buf[0] =0x12;

buf[1] =0x13;

buf[2] =0x14;

buf[3] =0x15;

buf[4] =0x16;

while(1)

{

Uart\_Send\_Array(COM4,buf,5);

Delay\_ms(1000);//延时1000毫秒

}

}

3.3 手机模块

第一步双击手机模块

/\*参数定义区和头文件包含区\*/

int main(void)//每个程序必须有一个主函数

{

Hard\_Init();//系统必须初始化部分，请勿删除

while(1)

{

u8 Buf["x"]={"号码"};

Gsm\_Init ("使用串口设备号","GSM复位管脚","GSM电源开关管脚","GSM状态管脚");

Gsm\_Call\_Phone(Buf,"Buf长度");//拨打电话

Gsm\_Get\_Phone();//接听电话

Gsm\_Power\_On ();//开机

Gsm\_Power\_Off();//关机

Gsm\_Hang\_Phone();//挂断电话

Gsm\_Irq\_Isr();//GSM模块输出的信息，通过中断来接收。

}

}

第二步 调整位置

/\*参数定义区和头文件包含区\*/

u8 Buf[11]={15268120393};

int main(void)//每个程序必须有一个主函数

{

Hard\_Init();//系统必须初始化部分，请勿删除

while(1)

{

Gsm\_Init (COM3,"GSM复位管脚","GSM电源开关管脚","GSM状态管脚");

Gsm\_Call\_Phone(Buf,11);//拨打电话

Gsm\_Get\_Phone();//接听电话

Gsm\_Power\_On ();//开机

Gsm\_Power\_Off();//关机

Gsm\_Hang\_Phone();//挂断电话

Gsm\_Irq\_Isr();//GSM模块输出的信息，通过中断来接收。

}

}

第三步设置管脚

/\*参数定义区和头文件包含区\*/

u8 Buf[11]={15268120393};

int main(void)//每个程序必须有一个主函数

{

Hard\_Init();//系统必须初始化部分，请勿删除

while(1)

{

Gsm\_Init (COM3,B13,B11,E15);

Gsm\_Call\_Phone(Buf,11);//拨打电话

Gsm\_Get\_Phone();//接听电话

Gsm\_Power\_On ();//开机

Gsm\_Power\_Off();//关机

Gsm\_Hang\_Phone();//挂断电话

Gsm\_Irq\_Isr();//GSM模块输出的信息，通过中断来接收。

}

}

### 4 SPI接口

编译所需命令：

SPI1\_GPIO\_Init("模式");

SPI1\_GPIO\_DeInit();

SPI1\_CS\_Low();

SPI1\_CS\_High();

SPI1\_Single\_Read("数据");

SPI1\_Single\_Write("数据");

SPI1\_Multiple\_Read("数据缓存区","数据长度");

SPI1\_Multiple\_Write("数据缓存区","数据长度");

所需命令解释：

* + SPI1\_GPIO\_Init("模式");

1.初始化系统SPI1外设

2.SPI1管脚固定，其定义如下:

CS=A4,SCK=A5,MISO=A6,MOSI=A7

3.例如:

SPI1\_GPIO\_Init(SPI\_Mode1);

4.SPI1分为四种模式:

SPI\_Mode1~SPI\_Mode4。分别代表:

CPOL\_Low;CPHA\_1Edge;

CPOL\_Low;CPHA\_2Edge;

CPOL\_High;CPHA\_1Edge;

CPOL\_High;CPHA\_2Edge;

* + SPI1\_GPIO\_DeInit();

1.失能系统SPI1外设。

2.例如:SPI1\_GPIO\_DeInit();

* + SPI1\_CS\_Low();

1.将SPI1外设片选拉低。

2.例如:SPI1\_CS\_Low();

* + SPI1\_CS\_High();

1.将SPI1外设片选拉高。

2.例如:SPI1\_CS\_High();

* + SPI1\_Single\_Read("数据");

1.通过SPI1读取一字节数据。

2.例如:SPI1\_Single\_Read(&Data);//读取一字节数据。

* + SPI1\_Single\_Write("数据");

1.通过SPI1写一字节数据。

2.例如:SPI1\_Single\_Write(Data);//发送一字节数据。

* + SPI1\_Multiple\_Read("数据缓存区","数据长度");

1.通过SPI1读取多个字节数据。

2.例如:SPI1\_Multiple\_Read (Buf,5);/\*其中Buf应该定义成：u8 Buf[5]。读5个字节\*/

* + SPI1\_Multiple\_Write("数据缓存区","数据长度");

1.通过SPI1写多个字节数据。

2.例如:SPI1\_Multiple\_Write (Buf,5);/\*其中Buf应该定义成：u8 Buf[5]。写5个字节\*/

(注意：SPI1的方式写成，你想再写一个同样的程序的时候，即可写SPI2)。

编译所需案例：

（参看实用模块案例）

5.1震动传感器

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Smart Lit ,Just Have Fun \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\* 头文件、参数定义、中断程序区 \*\*\*/

#define EXIT\_IRQ1\_EN /\* "x"代表外部中断通道(取值范围:0~4),特别声明:使用外部中断有局限性，当使用"0"号中断时,其中断硬件只能取"A0,B0,C0"等"0"号管脚，其他类推。\*/

void EXIT\_IRQ1\_ISR(void)

{

Led\_Power\_On(LED1);/\*打开指定通道LED,通道只能填LED1~LED4\*/

}

int main(void)/\*每个程序必须有一个主函数\*/

{

Hard\_Init();/\*系统必须初始化部分，请勿删除\*/

/\*初始化区\*/

EXTI\_IRQ\_Init(A1,EXTI\_IRQ1,FALL);

Led\_Gpio\_Init(E13,255,255,255);/\*指定4个GPIO为LED控制管脚,若不需要4个通道，其他管脚请写"255",将此移到初始化区\*/

while(1)

{

Led\_Power\_Off(LED1);/\*关闭指定通道LED\*/

Delay\_ms(1000);/\*延时1000毫秒\*/

}

}

5.2电脑控制灯

### 6 PWM接口

6.1 LED呼吸灯

**第一步 主函数**

/\*参数定义区和头文件包含区\*/

int main(void)//每个程序必须有一个主函数

{

Hard\_Init();//系统必须初始化部分，请勿删除

while(1)

{

"do sth;"

}

}

第二步初始化

/\*参数定义区和头文件包含区\*/

int main(void)//每个程序必须有一个主函数

{

Hard\_Init();//系统必须初始化部分，请勿删除

**PWM\_Init("频率");//四个通道对应C6、C7、C8、C9**

while(1)

{

"do sth;"

}

}

第三步使能

/\*参数定义区和头文件包含区\*/

int main(void)//每个程序必须有一个主函数

{

Hard\_Init();//系统必须初始化部分，请勿删除

PWM\_Init("频率");//四个通道对应C6、C7、C8、C9

while(1)

{

**PWM\_Enable("通道","高电平时间(微妙)");**

}

}

第四步修改参数

/\*参数定义区和头文件包含区\*/

int main(void)//每个程序必须有一个主函数

{

Hard\_Init();//系统必须初始化部分，请勿删除

PWM\_Init(20000);//四个通道对应C6、C7、C8、C9

while(1)

{

PWM\_Enable(PWM\_CH1,40);

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* PWM改变高电频时间，

T =1S/F=1S/20000HZ=1000MS/20000HZ=1000000US/20000Hz=50uS

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**6.2渐变灯**

/\*\*\* 头文件、参数定义、中断程序区 \*\*\*/

u8 a;

int main(void)//每个程序必须有一个主函数

{

Hard\_Init();//系统必须初始化部分，请勿删除

/\*初始化区\*/

PWM\_Init(20000);//四个通道对应C6、C7、C8、C9

PWM\_Enable(PWM\_CH2,0);

while(1)

{

a++;

if(a>=50)

{

a=0;

}

PWM\_Update(PWM\_CH2,a);

Delay\_ms(300);//延时1000毫秒

}

}

**6.3渐变灯**

/\*\*\* 头文件、参数定义、中断程序区 \*\*\*/

u8 a;

int main(void)//每个程序必须有一个主函数

{

Hard\_Init();//系统必须初始化部分，请勿删除

/\*初始化区\*/

PWM\_Init(20000);//四个通道对应C6、C7、C8、C9

PWM\_Enable(PWM\_CH2,0);

while(1)

{

for(a=0;a<50;a++)

{

a=0;

}

PWM\_Update(PWM\_CH2,a);

Delay\_ms(300);//延时1000毫秒

}

for(a=50;a>50;a--)

{

a=0;

}

PWM\_Update(PWM\_CH2,a);

Delay\_ms(300);//延时1000毫秒

}

### 7 ADC采样

编译所需命令：

ADC\_Hard\_Init();//系统默认使用C0作为ADC采样脚

"(u32)Data"=ADC\_Read\_Voltage();

所需命令解释：

* + ADC\_Hard\_Init();//系统默认使用C0作为ADC采样脚

1.初始化模数转换器。

2.ADC\_Hard\_Init();

* + "(u32)Data"=ADC\_Read\_Voltage();

1.获取模数转换的电压。

2.Data=ADC\_Read\_Voltage();//其中Data范围[0~4095]

编译所需案例：

7.1，电位器控制灯调亮度

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Smart Lit ,Just Have Fun \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\* 头文件、参数定义、中断程序区 \*\*\*/

u32 Data；

int main(void)/\*每个程序必须有一个主函数\*/

{

Hard\_Init();/\*系统必须初始化部分，请勿删除\*/

/\*初始化区\*/

PWM\_User\_Init();/\*四个通道对应C6、C7、C8、C9\*/

ADC\_Hard\_Init();//系统默认使用C0作为ADC采样脚

while(1)

{

Data=ADC\_Read\_Voltage();

if（Data<500）

{

PWM\_User\_Enable(PWM\_CH1,50);

}

else if（Data<1000）

{

PWM\_User\_Enable(PWM\_CH1,100);

}

else

{

PWM\_User\_Enable(PWM\_CH1,255);

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

通过旋钮（旋钮的数值是0~4095），调节灯暗亮程度。旋钮的数值在500以内，PWM数值50，旋钮的数值在1000以内，PWM数值100，其他是255最亮。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

### 8时钟节拍

"(u32)Data"=SysTick\_Get\_Timer();

"(u32)Data"=SysTick\_Get\_Counter();

* + "(u32)Data"=SysTick\_Get\_Timer();

1.获取时钟节拍时间，单位为毫秒。

2.例如:Data=SysTick\_Get\_Timer();

* + "(u32)Data"=SysTick\_Get\_Counter();

1.获取时钟节拍计数器，时钟节拍计数器每加1，时间为10毫秒。

2.例如:Data=SysTick\_Get\_Counter();

编译所需案例：

8.1，控制灯暗亮

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Smart Lit ,Just Have Fun \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\* 头文件、参数定义、中断程序区 \*\*\*/

u32 Data;

u8 deng;

int main(void)/\*每个程序必须有一个主函数\*/

{

Hard\_Init();/\*系统必须初始化部分，请勿删除\*/

/\*初始化区\*/

Buzza\_Gpio\_Init(E12);/\*初始化指定管脚为蜂鸣器控制脚,将此移到初始化区\*/

User\_GPIO\_Set\_Output(E13);

if（deng==1）

{

User\_GPIO\_Set\_High(E13);

}

else

User\_GPIO\_Set\_Low(E13);

while(1)

{

Data=SysTick\_Get\_Timer();

if( Data %1000==0)

{

deng~;

}

Buzza\_Power\_On();//使蜂鸣器发出声音

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

利用时钟节拍，控制在某一时做某一件事，同时程序可以一直在做另外一件事。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

## 四 实用模块

### 1系统函数

**Main函数**

**If判断**

**While无限循环**

**短延时**

**长延时**

**输出带参数信息**

**输出不带参数信息**

**EXIT(外部中断)**

/\* "x"为外部中断通道(范围:0~4),

声明:使用外部中断有局限性，当使用"0"号中断时,

其中断只能取"A0,B0,C0"等"0"号管脚，其他类推。\*/

#define EXIT\_IRQ"x"\_EN //移到参数定义区

void EXIT\_IRQ"x"\_ISR(void)//移到中断服务区

{

"do sth"

}

/\*参数说明:中断管脚:中断管脚，

中断向量:EXTI\_IRQ[0..4],

触发方式：RISE(上升沿)、FALL(下降沿)、RISE\_FALL(双边沿)

例如:EXTI\_IRQ\_Init(A0,EXTI\_IRQ1,FALL);\*/

EXTI\_IRQ\_Init("中断管脚","中断向量","中断触发方式");//移到初始化区

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

注意事项：

"中断管脚"，规定[A,B,C,D,E][0..4]为中断管脚。中断固定管脚，如果是用0中断，管脚对应A0,B0,C0,D0,E0;如果是用1中断，管脚对应A1,B1,C1,D1,E1,以此类推。总共0~4个中断。

"中断向量"，EXTI\_IRQ[0..4],例如EXTI\_IRQ0。

"中断触发方式"，RISE(上升沿)、FALL(下降沿)、RISE\_FALL(双边沿)，使用那种方式取决于传感器的电平方式（模块中说明），如果是高电平，就是下降沿，如果是低平，就上升沿，如果是有高平、低平，才进入中断，即双边沿。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**串口接收中断**

#define USART\_IRQ"x"\_EN //移到参数定义区/\* "x"代表串口通道(取值范围:1~5) \*/

Uart\_Nvic\_Init("串口编号");//输入参数:COM1~5，移入初始化区

void USART\_IRQ"x"\_ISR(void)//移到中断服务区

{

USART\_ReceiveData(USART“x”);//如果是COM1到COM3分别填入USART1到USART3，COM4或COM5填入UART4或UART5

"do sth"//接收内容以及服务程序

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

外部中断，相当于I/O类，接受到信号，就执行中断命令。

串口通道，接受特定的信号，才执行中断命令。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Switch选择**

**For循环**

### 2 I/O类模块

编译所需模块：

按键检测模块、LED灯模块、电机驱动、火焰传感器、磁传感器、续电器模块、气体传感器、温度传感器、蜂鸣器、光敏电阻、雨水传感器、红外避障、震动传感器

1按键检测模块

2 LED灯模块

3电机驱动

4火焰传感器

5磁传感器

6续电器模块

7气体传感器

8温度传感器

例一 温度实验

/\*\*\* 头文件、参数定义、中断程序区 \*\*\*/

s16 Data;

int main(void)//每个程序必须有一个主函数

{

Hard\_Init();//系统必须初始化部分，请勿删除

/\*初始化区\*/

Ds18b20\_Gpio\_Init(E13);//初始化温度传感器

Ds18b20\_Init();//初始化温度传感器

while(1)

{

Data=Ds18b20\_Read\_Temp();//读取温度，这里温度放大了10倍，实际温度除以10。

Dprintf("打印内容: %d \r\n",Data);//Data是需要打印的数据（之前已经定义）,其它使用方法请参考C库中printf函数，\r\n为回车换行

Delay\_ms(1000);//延时1000毫秒

}

}

9蜂鸣器

例一蜂鸣器实验

/\*参数定义区和头文件包含区\*/

int main(void)//每个程序必须有一个主函数

{

Hard\_Init();//系统必须初始化部分，请勿删除

Buzza\_Gpio\_Init(E13);//初始化指定管脚为蜂鸣器控制脚

while(1)

{

Buzza\_Power\_On();//使蜂鸣器发出声音

Delay\_ms(1000);//延时1000毫秒

Buzza\_Power\_Off();//使蜂鸣器不发出声音

Delay\_ms(1000);//延时1000毫秒

}

}

9光敏电阻

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Smart Lit ,Just Have Fun \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\* 头文件、参数定义、中断程序区 \*\*\*/

u8 Data;/\*将该行移到参数定义区\*/

int main(void)/\*每个程序必须有一个主函数\*/

{

Hard\_Init();/\*系统必须初始化部分，请勿删除\*/

/\*初始化区\*/

Light\_Res\_Gpio\_Init(E12);/\*初始化指定管脚为光敏电阻控制脚,将此移到初始化区\*/

Buzza\_Gpio\_Init(E13);//初始化指定管脚为蜂鸣器控制脚

while(1)

{

Data=Light\_Res\_Get\_Status();/\*获取光敏电阻的状态\*/

if(Data=1)

{

Buzza\_Power\_On();//使蜂鸣器发出声音

Delay\_ms(1000);//延时1000毫秒

}

else

Buzza\_Power\_Off();//使蜂鸣器不发出声音

Delay\_ms(1000);//延时1000毫秒

}

}

10雨水传感器

11红外避障

12震动传感器

### 3 UART模块

1GSM手机模块

编译所需命令：

u8 Buf["x"]={"号码"};

Gsm\_Init ("使用串口设备号","GSM复位管脚","GSM电源开关管脚","GSM状态管脚");

Gsm\_Call\_Phone(Buf,"Buf长度");//拨打电话

Gsm\_Get\_Phone();//接听电话

Gsm\_Power\_On ();//开机

Gsm\_Power\_Off();//关机

Gsm\_Hang\_Phone();//挂断电话

Gsm\_Irq\_Isr();//GSM模块输出的信息，通过中断来接收。

编译所需步骤：

2GPS定位仪

编译所需命令：

Uart\_Init("串口编号", "串口波特率");//初始化使用的串口

Uart\_Nvic\_Init("串口编号");//初始化GPS接收中断

Gps\_Init("串口编号");//初始化GPS模块

u16 Gps\_Irq\_Isr();//GPS模块接收中断服务程序，这里返回定位信息

编译所需步骤：

3蓝牙模块

参考帮助

4WIFI模块

参考帮助

### 4 I2c模块

1电子方位仪

编译所需命令：

u16 Data;

Io\_I2c\_Gpio\_Init("I2C编号", "SCL管脚", "SDA管脚");//初始化I2C

Hmc5883\_Init("I2C编号",0x3c);

Data=Hmc5883\_Read\_Angle();//结果为0~360度的值，依据这个值来判别方向。

编译所需步骤：

2时钟模块

编译所需命令：

DateTime\_t DateTime;

Io\_I2c\_Gpio\_Init("I2C编号", "SCL管脚", "SDA管脚");//初始化I2C

S35390a\_init("I2C编号",0x60);

S35390a\_Set\_Time(&DateTime);

S35390a\_Get\_Time(&DateTime);

编译所需步骤：

时钟实验一 读取秒

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Smart Lit ,Just Have Fun \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\* 头文件、参数定义、中断程序区 \*\*\*/

/\*\*将该段移到参数定义区\*\*/

DateTime\_t DateTime;

/\*\*\*\*\*\*\*\*\*\*\*结束\*\*\*\*\*\*\*\*\*\*/

int main(void)/\*每个程序必须有一个主函数\*/

{

Hard\_Init();/\*系统必须初始化部分，请勿删除\*/

/\*初始化区\*/

DateTime.year=14;/\*没有千年位,若2014年,只需要填写14即可\*/

DateTime.month=2;/\*月份\*/

DateTime.day=26;/\*日期\*/

DateTime.week=3;/\*星期\*/

DateTime.hour=8;/\*时\*/

DateTime.minute=16;/\*分\*/

DateTime.second=33;/\*秒\*/

Io\_I2c\_Gpio\_Init(I2C\_1, E13, E15);/\*初始化I2C,将此移到初始化区\*/

S35390a\_Init(I2C\_1,0x60);/\*初始化时钟,将此移到初始化区\*/

S35390a\_Set\_Time(&DateTime);/\*设置时钟,需要时将此移到初始化区\*/

while(1)

{

S35390a\_Get\_Time(&DateTime);/\*获取时间,时间内容写法与之前参数定义区一样\*/

Dprintf("打印内容:%d \r\n", DateTime.second);/\* Data是需要打印的数据(之前已经定义),其它使用方法请参考C库中printf函数，\r\n为回车换行 \*/

Delay\_ms(1000);/\*延时1000毫秒\*/

}

}

3数据存储

编译所需命令：

u8 Data;

Io\_I2c\_Gpio\_Init("I2C编号", "SCL管脚", "SDA管脚");//初始化I2C

At24c0x\_Init("I2C编号",0xa0);

Data=At24c0x\_Get\_Value ("读取地址");//地址范围0x00~0xff。

At24c0x\_Set\_Value ("指定地址","需要存储的数据");//地址范围0x00~0xff。

编译所需步骤：

4陀螺仪和加速度计

编译所需命令:

Mpu6050\_Data\_t Mpu6050\_Data;

Io\_I2c\_Gpio\_Init("I2C编号", "SCL管脚", "SDA管脚");//初始化I2C

Mpu6050\_Init("I2C编号",0xd0);

Mpu6050\_Read(&Mpu6050\_Data);

编译所需步骤：

5大气压强测量

编译所需命令:

s32 Data;

Io\_I2c\_Gpio\_Init("I2C编号", "SCL管脚", "SDA管脚");//初始化I2C

Bmp085\_Init("I2C编号",0xee);

Data=Bmp085\_Read\_Pressure();//读取大气压强的值

编译所需步骤：

### 5 SPI模块

1语音识别

编译所需命令：

SPI2\_GPIO\_Init("SPI工作模式");

u8 Ld3320\_Run\_ASR();

u8 Ld3320\_Get\_Result();

Ld3320\_Irq\_Isr();

编译所需案例：

案例，语音识别

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Smart Lit ,Just Have Fun \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\* 头文件、参数定义、中断程序区 \*\*\*/

u8 Data;/\*将该行移到参数定义区\*/

SPI2\_GPIO\_Init(SPI\_Mode3);/\*初始化SPI2接口,将此移到初始化区\*/

/\*\* 将该段移到中断程序区 \*\*/

#define EXIT\_IRQ1\_EN //"x"代表外部中断通道(取值范围:0~4)

void EXIT\_IRQ1\_ISR(void)

{

Ld3320\_Irq\_Isr();

}

/\*\*\*\*\*\*\*\*\*\*\*\*结束\*\*\*\*\*\*\*\*\*\*\*\*/

int main(void)/\*每个程序必须有一个主函数\*/

{

Hard\_Init();/\*系统必须初始化部分，请勿删除\*/

/\*初始化区\*/

EXTI\_IRQ\_Init(A1,EXTI\_IRQ1,FALL);/\*初始化中断,将此移到初始化区\*/

while(1)

{

Ld3320\_Run\_ASR();/\*启动一次语音识别\*/

Data=Ld3320\_Get\_Result();/\*获取一次语音识别的结果\*/

Dprintf("打印内容：Hello world！\r\n");/\* \r\n为回车换行 \*/

Delay\_ms(1000);/\*延时1000毫秒\*/

}

}

### 6其他类模块

1颜色传感器

编译所需命令：

Tcs\_Colour\_t Colour\_Buf;

Gpio\_Tcs3200\_Init("S0管脚","S1管脚","S2管脚","S3管脚","数据管脚");

Tcs3200\_Read\_Colour(&Colour\_Buf);

编译所需步骤：

2红外计数器

编译所需命令：

u8 Data;

Ir\_Counter\_Gpio\_Init ("中断管脚");//初始化红外计数器管脚。使用外部中断进行计数，不会丢失计数值

编译所需步骤：

3红外遥控

编译所需命令：

u8 Data;//将该行移到参数定义区

EXTI\_IRQ\_Init("中断管脚",EXTI\_IRQ"x",FALL);//初始化中断,将此移到初始化区

Irda\_Gpio\_Init("中断管脚");//初始化红外接收头,将此移到初始化区

/\*\* 将该段移到中断程序区 \*\*/

#define EXIT\_IRQ"x"\_EN //"x"代表外部中断通道(取值范围:0~4)

void EXIT\_IRQ"x"\_ISR(void)

{

Irda\_Irq\_Isr();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Data=Irda\_Get\_value();//获取遥控的按键值。

编译所需步骤：

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Smart Lit ,Just Have Fun \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\* 头文件、参数定义、中断程序区 \*\*\*/

u8 Data;/\*将该行移到参数定义区\*/

/\*\* 将该段移到中断程序区 \*\*/

#define EXIT\_IRQ1\_EN //"x"代表外部中断通道(取值范围:0~4)

void EXIT\_IRQ1\_ISR(void)

{

Irda\_Irq\_Isr();

}

/\*\*\*\*\*\*\*\*\*\*\*\*结束\*\*\*\*\*\*\*\*\*\*\*\*/

int main(void)/\*每个程序必须有一个主函数\*/

{

Hard\_Init();/\*系统必须初始化部分，请勿删除\*/

/\*初始化区\*/

EXTI\_IRQ\_Init(E13,EXTI\_IRQ1,FALL);/\*初始化中断,将此移到初始化区\*/

Irda\_Gpio\_Init(E13);/\*初始化红外接收头,将此移到初始化区\*/

while(1)

{

Data=Irda\_Get\_value();//获取遥控的按键值。

Dprintf("打印内容:%d \r\n",Data);/\* Data是需要打印的数据(之前已经定义),其它使用方法请参考C库中printf函数，\r\n为回车换行 \*/

Delay\_ms(1000);/\*延时1000毫秒\*/

}

}

4LCD液晶显示

编译所需命令：

Lcd\_Init("CS管脚","RST管脚","RS管脚","SCK管脚","SDA管脚");//使用指定管脚，并初始化液晶显示器

Lcd\_Clern();//清除液晶上面所有的显示

Lcd\_Display\_String("显示内容","横坐标","纵坐标");//显示指定内容字符串到响应的横纵坐标(其中横坐标取值为[0~3]，纵坐标取值为[0~127])

Lcd\_Display\_Ascii("ASCII码表","横坐标","纵坐标");//显示单个ASCII码到液晶显示器上。

编译所需步骤：

第一步 载入模块

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Smart Lit ,Just Have Fun \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\* 头文件、参数定义、中断程序区 \*\*\*/

int main(void)/\*每个程序必须有一个主函数\*/

{

Hard\_Init();/\*系统必须初始化部分，请勿删除\*/

/\*初始化区\*/

while(1)

{

Lcd\_Init("CS管脚","RST管脚","RS管脚","SCK管脚","SDA管脚");/\*使用指定管脚，并初始化液晶显示器,将此移到初始化区\*/

Lcd\_Clern();/\*清除液晶上面所有的显示\*/

Lcd\_Display\_String("显示内容","横坐标","纵坐标");/\*显示指定内容字符串到响应的横纵坐标(其中横坐标取值为[0~3]，纵坐标取值为[0~127])\*/

Lcd\_Display\_Ascii("ASCII码表","横坐标","纵坐标");/\*显示单个ASCII码到液晶显示器上\*/

}

}

第二步 调整模块程序

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Smart Lit ,Just Have Fun \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\* 头文件、参数定义、中断程序区 \*\*\*/

int main(void)/\*每个程序必须有一个主函数\*/

{

Hard\_Init();/\*系统必须初始化部分，请勿删除\*/

Lcd\_Init(E13,E15,B15,E10,D13);/\*使用指定管脚，并初始化液晶显示器,将此移到初始化区\*/

Lcd\_Clern();/\*清除液晶上面所有的显示\*/

while(1)

{

Lcd\_Display\_String("Hello world!",1,20);/\*显示指定内容字符串到响应的横纵坐标(其中横坐标取值为[0~3]，纵坐标取值为[0~127])\*/

Lcd\_Display\_Ascii('a',2,0);/\*显示单个ASCII码到液晶显示器上\*/

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

"CS管脚","RST管脚","RS管脚","SCK管脚","SDA管脚"。A0管脚与RS管脚同

"显示内容"。

"横坐标",从第几行开始显示，范围[0~3]。

"纵坐标"从第几位开始显示，范围[0~127]。

"ASCII码表",用单引号注释’’，如‘a’。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

### 7常用关键字

u8定义一个8位无符号的数据。其能记录的数据范围为[0~255]。

u16定义一个16位无符号的数据。其能记录的数据范围为[0~65535]。

u32定义一个32位无符号的数据。其能记录的数据范围为[0~4294967295]。

float定义一个32位浮点的数据。其能记录的数据范围为[-3.4\*10的负38~3.4\*10的38次方]。

enum 定义枚举类型，用于补充u8、u16等数据类型，使程序看起来更简洁。常用于定义有穷个数且有规律的数据，比如：星期一到星期天，各个元素的值都是在上一个值加"1"。例如:

enum weeks =

{

monday=0x00,

tuesday,

wennesday,

thursday,

friday,

saturday,

sunday

};

struct 定义结构体类型，用于补充u8、u16等数据类型，使程序看起来更简洁。常用于定义和某个对象相关的数据，比如：时间分为年月日时分秒。例如:

struct timmer =

{

u8 year;

u8 month;

u8 day;

u8 week;

u8 hour;

u8 minute;

u8 second;

};

void用于提示用户该位置为"空"的类型，用户在写程序的时候碰到的时候只需要将该字删除，不要操作。

break用于跳出某个函数，比如当达到一定条件，不需要再运行本程序的时候，使用此关键字就可以达到目的。

switch 选择判断执行，常用于判断某个事件，而该事件又可以列举出。根据结果来执行相应事件中的程序。例如:

switch ()

{

case A:

do sth;

break;

case B:

do sth;

break;

default:

do sth;

break;

}

case匹配条件。常和switch一起使用。详情见switch关键字。

default匹配条件。常和switch一起使用。详情见switch关键字。

for循环控制关键字。例如循环10次:

u8 a;

for (a=0;a<10;a++)

{

do sth;

}

while循环控制关键字。例如使用while来实现死循环:

while(1)

{

do sth;

}

if条件判断关键字，常和else关键字在一起使用。例如:

u8 a, b;

if (a>b)

{

do sth;

}

else

{

do sth;

}

else其他条件匹配。常和if关键在一起使用。详情见if关键字。

return函数返回的一个结果。

### 8程序概念

/\*

"函数由:返回值类型、函数名、输入参数类型、以及函数体构成

一般写法:

返回值类型 函数名（输入参数）

{

函数体;

}

一般有四种方式：

1. 无返回，无传入参数

2. 无返回，有输入参数

3. 有返回，无输入参数

4. 有返回，有输入参数

实例:

void func (void)

{

do sth.;

}//定义的一个没有返回值，没有输入参数的函数名为func的函数"

\*/

**顺序执行**

/\*

"程序按正常顺序执行，中间不会发生其他任何事情"

**选择执行**

/\*

"程序选择性执行，中间会出现判断的语句，比如（if）等字样"

\*/

**循环执行**

/\*

"程序按用户设定的循环次数执行相同"

\*/

**函数调用方式**

/\*

"根据函数定义方式，然后使用相应方式去调用函数

如果遇到，无返回或无传入参数的时候需要将(void)关键字去掉

如果遇到有参数的情况，需要将相应的位置定义成相同参数类型的参数进行调用。

比如有返回值调用方法:Val=func();

有参数传入为:func(Val);

无传入无返回:func();"

## 五 综合案例

### **案例一 8421灯实验**

本项目需要用到的元器件及设备见表1.1。

**表1.1元器件及设备**

|  |  |  |
| --- | --- | --- |
| 元器件及设备 | 说明 | 附录 |
| 电子罗盘模块 |  |  |
| 电机驱动 |  |  |
| 遥控器 |  |  |
| 电源 |  |  |
| 电机 |  |  |

1.1.1 硬件

1.1.2软件

1.1.3项目合成

/\*参数定义区和头文件包含区\*/

u8 a;

int main(void)//每个程序必须有一个主函数

{

Hard\_Init();//系统必须初始化部分，请勿删除

User\_GPIO\_Set\_Output(C7);

User\_GPIO\_Set\_Output(E13);

User\_GPIO\_Set\_Output(E15);

User\_GPIO\_Set\_Output(B11);

a=0;

while(1)

{

if((a&0x01)==0x01) //点亮C7

{

User\_GPIO\_Set\_High(C7);

}

else

{

User\_GPIO\_Set\_Low(C7);

}

if((a&0x02)==0x02) // 点亮E13

{

User\_GPIO\_Set\_High(E13);

}

else

{

User\_GPIO\_Set\_Low(E13);

}

if((a&0x04)==0x04) //点亮E15

{

User\_GPIO\_Set\_High(E15);

}

else

{

User\_GPIO\_Set\_Low(E15);

}

if((a&0x08)==0x08) //点亮B11

{

User\_GPIO\_Set\_High(B11);

}

else

{

User\_GPIO\_Set\_Low(B11);

}

Delay\_ms(1000);//延时1000毫秒

a++;

if(a>16)

{

a=0;

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

解释：

（a&0x01)==0x01，这条语句是这里的关键，我们需要找到规律，按照c语言的逻辑关系表达，做一次&运算（同侧同，不同则不同），在判断结果是否是0x01，是的话就点亮灯。其他几位可以类推。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

### **案例二触摸开关**

本项目需要用到的元器件及设备见表2.1。

**表2.1元器件及设备**

|  |  |  |
| --- | --- | --- |
| 元器件及设备 | 说明 | 附录 |
| 电子罗盘模块 |  |  |
| 电机驱动 |  |  |
| 遥控器 |  |  |
| 电源 |  |  |
| 电机 |  |  |

2.1.1 硬件

2.1.2软件

2.1.3项目合成

### **案例三恒温器**

本项目需要用到的元器件及设备见表3.1。

**表3.1元器件及设备**

|  |  |  |
| --- | --- | --- |
| 元器件及设备 | 说明 | 附录 |
| 电子罗盘模块 |  |  |
| 电机驱动 |  |  |
| 遥控器 |  |  |
| 电源 |  |  |
| 电机 |  |  |

3.1.1 硬件

3.1.2软件

3.1.3项目合成

### **案例四跟屁虫**

电子罗盘确定方向，按照方向驱动电机，达到跟随的功能。

本项目需要用到的元器件及设备见表4.1。

**表4.1元器件及设备**

|  |  |  |
| --- | --- | --- |
| 元器件及设备 | 说明 | 附录 |
| 电子罗盘模块 |  |  |
| 电机驱动 |  |  |
| 遥控器 |  |  |
| 电源 |  |  |
| 电机 |  |  |

7.1.1 硬件

7.1.2软件

7.4.3项目合成

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Smart Lit ,Just Have Fun \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\* 头文件、参数定义、中断程序区 \*\*\*/

u16 Data;/\*将该行移到参数定义区\*/

int main(void)/\*每个程序必须有一个主函数\*/

{

Hard\_Init();/\*系统必须初始化部分，请勿删除\*/

/\*初始化区\*/

Io\_I2c\_Gpio\_Init(I2C\_3, E11, E10);/\*初始化I2C,将此移到初始化区\*/

Hmc5883\_Init(I2C\_3,0x3c);/\*初始化电子罗盘,将此移到初始化区\*/

Moto\_Gpio\_Init(MOTO1,E13,E12,E15);/\*初始化指定通道的控制管脚(电机通道取值范围MOTO1、MOTO2),将此移到初始化区\*/

while(1)

{

Data=Hmc5883\_Read\_Angle();/\*结果为0~360度的值，依据这个值来判别方向\*/

Dprintf("打印内容: %d \r\n",Data);/\* Data是需要打印的数据(之前已经定义),其它使用方法请参考C库中printf函数，\r\n为回车换行 \*/

Delay\_ms(1000);/\*延时1000毫秒\*/

if((Data>=0)&&(Data<=90))

{

Moto\_Forword (MOTO1);/\*电机正转\*/

Delay\_ms(1000);/\*延时1000毫秒\*/

}

else

Moto\_Stop(MOTO1);/\*电机停止\*/

}

}

### **案例五自动窗帘**

这段程序要通过遥控器控制电机，实现窗帘自动卷起。

本项目需要用到的元器件及设备见表5.1。

**表5.1元器件及设备**

|  |  |  |
| --- | --- | --- |
| 元器件及设备 | 说明 | 附录 |
| 电机驱动 |  |  |
| 红外传感器 |  |  |
| 遥控器 |  |  |
| 电源 |  |  |
| 电机 |  |  |

7.1.1 硬件

7.1.2软件

7.4.3项目合成

### **案例六声控灯**

本项目需要用到的元器件及设备见表6.1。

**表6.1元器件及设备**

|  |  |  |
| --- | --- | --- |
| 元器件及设备 | 说明 | 附录 |
| 电子罗盘模块 |  |  |
| 电机驱动 |  |  |
| 遥控器 |  |  |
| 电源 |  |  |
| 电机 |  |  |

7.1.1 硬件

7.1.2软件

7.4.3项目合成

### **案例七 时钟**

本项目需要用到的元器件及设备见表7.1。

**表7.1元器件及设备**

|  |  |  |
| --- | --- | --- |
| 元器件及设备 | 说明 | 附录 |
|  |  |  |
|  |  |  |
|  |  |  |

7.1.1 硬件

7.1.2软件

7.4.3项目合成

### **案例八 自动浇灌系统**

这段程序是用时，间秒来设置，在实际使用时，可以根据需要更改时间。我们利用电机模块和时钟模块，作一个简单的时间判断，组合完成自动浇灌系统。

本项目需要用到的元器件及设备见表8.1。

**表8.1元器件及设备**

|  |  |  |
| --- | --- | --- |
| 元器件及设备 | 说明 | 附录 |
| 主控板 |  |  |
| 时钟模块 |  |  |
| 水泵 |  |  |
| 电源 |  |  |

7.1.1 硬件

链接方式：

7.1.2软件

7.4.3项目合成

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Smart Lit ,Just Have Fun \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\* 头文件、参数定义、中断程序区 \*\*\*/

DateTime\_t DateTime;

int main(void)/\*每个程序必须有一个主函数\*/

{

Hard\_Init();/\*系统必须初始化部分，请勿删除\*/

/\*初始化区\*/

Moto\_Gpio\_Init(MOTO1,E13,E15,E10);/\*初始化指定通道的控制管脚(电机通道取值范围MOTO1、MOTO2),将此移到初始化区\*/

DateTime.year=14;/\*没有千年位,若2014年,只需要填写14即可\*/

DateTime.month=2;/\*月份\*/

DateTime.day=26;/\*日期\*/

DateTime.week=3;/\*星期\*/

DateTime.hour=8;/\*时\*/

DateTime.minute=16;/\*分\*/

DateTime.second=33;/\*秒\*/

Io\_I2c\_Gpio\_Init(I2C\_3, E8,B1);/\*初始化I2C,将此移到初始化区\*/

S35390a\_Init(I2C\_3,0x60);/\*初始化时钟,将此移到初始化区\*/

S35390a\_Set\_Time(&DateTime);/\*设置时钟,需要时将此移到初始化区\*/

while(1)

{

S35390a\_Get\_Time(&DateTime);/\*获取时间,时间内容写法与之前参数定义区一样\*/

Dprintf("打印内容:%d \r\n",DateTime.second);/\* Data是需要打印的数据(之前已经定义),其它使用方法请参考C库中printf函数，\r\n为回车换行 \*/

Delay\_ms(500);/\*延时1000毫秒\*/

if(DateTime.second==40)

{

Moto\_Forword (MOTO1);/\*电机正转\*/

//Delay\_ms(1000);/\*延时1000毫秒\*/

}

if(DateTime.second==50)

{

Moto\_Reversal (MOTO1);/\*电机反转\*/

//Delay\_ms(1000);/\*延时1000毫秒\*/

}

// Moto\_Stop(MOTO1);/\*电机停止\*/

// Delay\_ms(1000);/\*延时1000毫秒\*/

}

}

### **案例九 自动种植蘑菇机系统**

**表9.1元器件及设备**

|  |  |  |
| --- | --- | --- |
| 元器件及设备 | 说明 | 附录 |
| 主控板 |  |  |
| 时钟 |  |  |
| 温湿传感器 |  |  |
| 冷片器 |  |  |

### **案例十 自动辅助行走系统**

本项目需要用到的元器件及设备见表10.1。

**表10.1元器件及设备**

|  |  |  |
| --- | --- | --- |
| 元器件及设备 | 说明 | 附录 |
| 主控板 |  |  |
| Mpu6050传感器 |  |  |
| 伺服电机 |  |  |
|  |  |  |

10.1.1 硬件

10.1.2软件

10.4.3项目合成

/\*------------------------------

Smart Lit ,Just Have Fun

------------------------------\*/

/\*\*\* 头文件、参数定义、中断程序区 \*\*\*/

#include <math.h>

#define ACC\_SCALE 0.00012207f //4G

#define RAD2DEC 57.2958278f //弧度变度

Mpu6050\_Data\_t Mpu6050\_Data;/\*将该行移到参数定义区\*/

float Ax,Ay,Az;

float Angle\_accX;

u32 Data;

u8 q;

int main(void)/\*主函数\*/

{

Hard\_Init(); /\*初始化区\*/

Io\_I2c\_Gpio\_Init(I2C\_1,E13, E15);/\*初始化I2C,将此移到初始化区\*/

Mpu6050\_Init(I2C\_1,0xd0);

PWM\_Init(20000);/\*四个通道对应C6、C7、C8、C9\*/

PWM\_Enable(PWM\_CH1,20); //转速

ADC\_Hard\_Init();//系统默认使用C0作为ADC采样脚

Moto\_Gpio\_Init(MOTO1,E11,E10,C9);/\*指定电机通道的控制管脚(取值范围MOTO1、MOTO2),将此移到初始化区化区\*/

while(1)

{

Mpu6050\_Read(&Mpu6050\_Data);

Ax = Mpu6050\_Data.acc[0] \* ACC\_SCALE;

Ay = Mpu6050\_Data.acc[1] \* ACC\_SCALE;

Az = Mpu6050\_Data.acc[2] \* ACC\_SCALE;

Angle\_accX= atan2(Ax , Az)\*RAD2DEC; //换传感器摆动值

// Angle\_accX= atan(Ax / Az)\*180/3.14159;

Dprintf("Angle = %.3f \r\n", Angle\_accX);

// Dprintf("XYZ= %d %d %d \r\n",Mpu6050\_Data.acc[0],Mpu6050\_Data.acc[1],Mpu6050\_Data.acc[2]);

Delay\_ms(100);/\*延时1000毫秒\*/

Data=ADC\_Read\_Voltage(); //pwm与ad数值转换,根据电位器更新转速

Dprintf("ad= %d \r\n",Data);

Delay\_ms(100);/\*延时1000毫秒\*/

q=Data/82;

PWM\_Update(PWM\_CH1,q);

Dprintf("pwm= %d \r\n",q);

if(q>49) //预防超过4096

{

q=49;

}

if((Angle\_accX<=90)&&(Angle\_accX>=-90)) //转动角度调节

{

Moto\_Forword (MOTO1);/\*电机正转\*/

}

else

{

Moto\_Stop(MOTO1);/\*电机停\*/

Delay\_ms(100);/\*延时1000毫秒\*/

Moto\_Reversal(MOTO1);/\*电机反转\*/

}

}

}

### 1 I/O开关传感器

LED、雨水、蜂鸣器、电机驱动遥控器，控制灯 马达

有雨水，马达转动收衣架，定时浇灌

遥控窗帘

### 2 I2c传感器

电子罗盘

### 3 UART串口

电话

### 4 SPI

语音识别