杭州艾研信息技术有限公司

物联网实验套件 AY-IOT KIT 使用指南

——基于 CC3200

版本号<1.0>

修订历史记录

日期	版本	说明	作者
2017/03/07	1.0	初版	周海，周苍苍

目录

物联网实验套件 AY-IOT KIT 使用指南

——基于 CC3200

1 实验套件概要

物联网实验套件使用 CC3200 为主控 MCU，使用 Energia 为软件开发平台，实现网络通信功能。套件为每个模块提供了不同的实验例程，通过各个实验例程用户可以了解到 Energia 平台下 CC3200 的 WiFi 操作过程，以及其他如 ADC、I2C、SPI、PWM 等外设的实现。

套件资料可在艾研信息官方网站上下载：http://www.hpati.com/ay_wifi/product_57.html 或者 http://www.hpati.com/resources/。

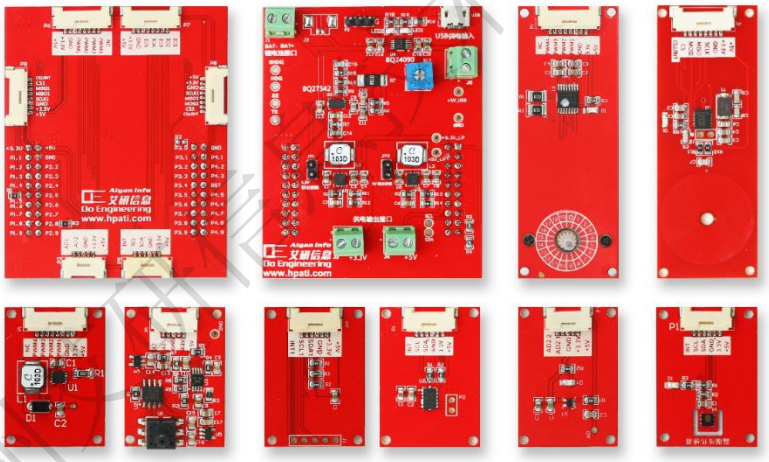


图 1-1 实验套件

2 软件

物联网实验套件全部软件资源可从网络活动。包括单片机开发软件、实验例程代码、实验例程操作视频。

2.1 开发软件

物联网实验套件使用 TI 公司的 Energia 软件进行开发，下载地址为 http://energia.nu/download/，可选择最新版本下载。Energia 是一个开源软件，无代码限制（TI 的另一款开发软件——CCS 可永久免费使用代码限制版本），下载并解压后即可直接使用。对于 Energia 是使用可查看视频资源《Getting Started Guide》或者登录 Energia 官网了解，网址：http://energia.nu/guide/guide_windows/。

Energia 本身提供很多库函数，可在 Energia 官网上了解库函数的使用：http://energia.nu/reference/，当然也可以在 Energia 安装目录下查看库的源代码。Energia 支持多种 LaunchPad，在查看源代码时需要选择相应的硬件平台。比如实验套件使用的是 cc3200LaunchPad，其库文件目录为：【Energia 安装目录】->hardware->cc3200->libraries（如 C:\Program Files (x86)\energia-0101E0017\hardware\cc3200\libraries）。

2.2 例程代码

从艾研信息官方网站 www.hpati.com 上下载例程代码，将代码放进 Energia 相应库文件目录下后可直接使用。操作可查看视频资源《Getting Started Guide》，下面也简单介绍操作过程：

1、进入网站下载例程代码（http://www.hpati.com/ay_source_download/）；

图 2-1 例程

2、下载例程解压，每个例程包含对应模块的源代码已经 examples；

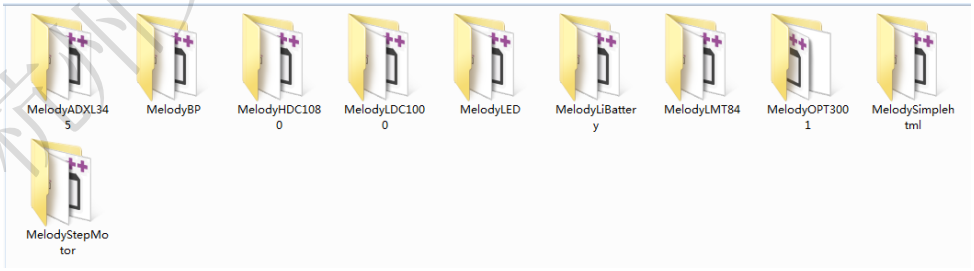


图 2-2 例程文件

3、将解压后所有的文件（即图 2-2 中的所有文件）复制进【Energia 安装目录】->hardware->cc3200->libraries 目录下，如【C:\Program Files (x86)\energia-0101E0017\hardware\cc3200\libraries】;

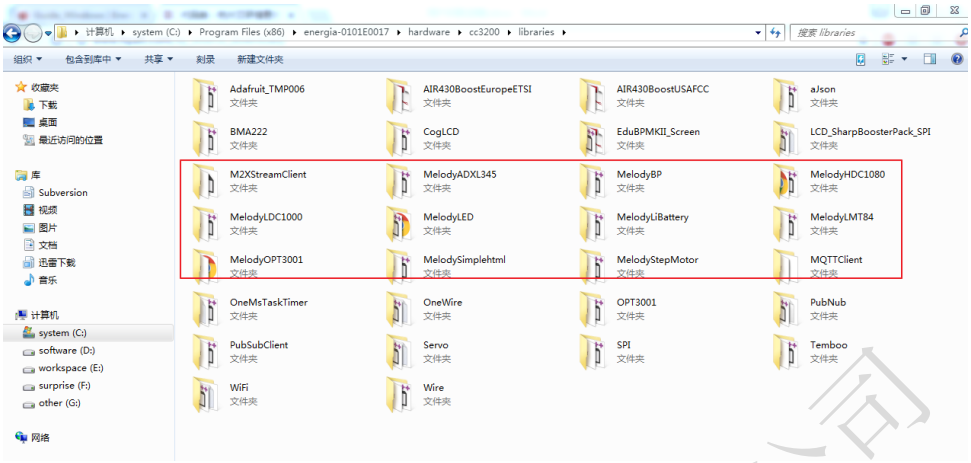


图 2-3 库文件目录

4、打开 Energia，在菜单栏 Tools->Board 下选择 Launchpad CC3200 w/cc3200 (80MHz)

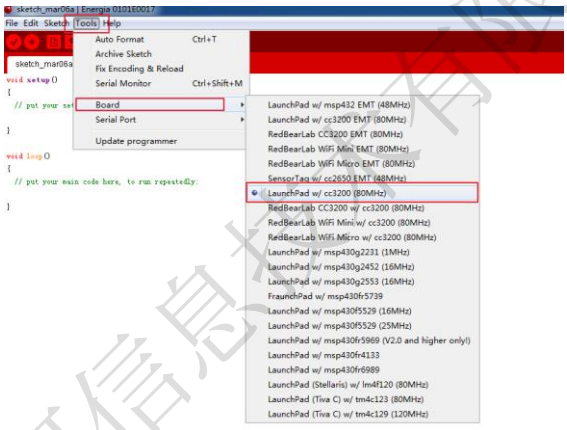


图 2-4 Tools 下选择 Board

5、查看链接进入的库文件文件

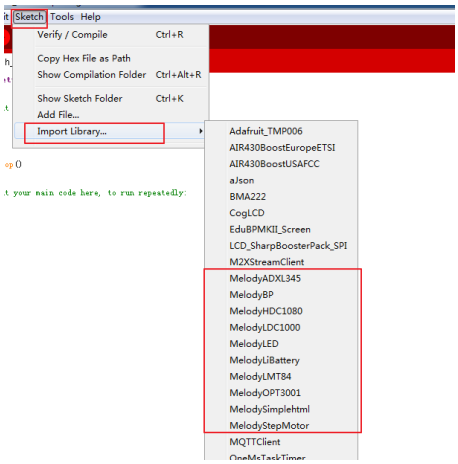


图 2-5sketch 下查看库文件

6、查看 examples

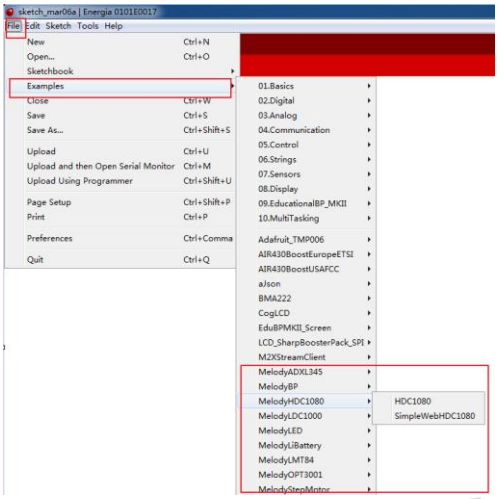


图 2-6 File 下查看 Examples

7、选择一个例程编译后即可下载使用。

2.2.1 WebServer

由图 2-2 例程文件展示的文件中除“MelodySimplehtml”文件夹外，其余文件都对应着一个模块，每个模块的例程在“3 实验套件模块”章节中介绍，每个实验模块均使用 webserver 功能实现网络操作。

本节简单介绍 Energia 下 cc3200webserver 功能的使用。Energia 下提供了 webserver 的实验例程，例程可在 File->Examples->WiFi->SimpleWebServerWiFi 找到。实验例程源代码直接查看文件即可，例程代码实现的流程图如图 2-1 所示。

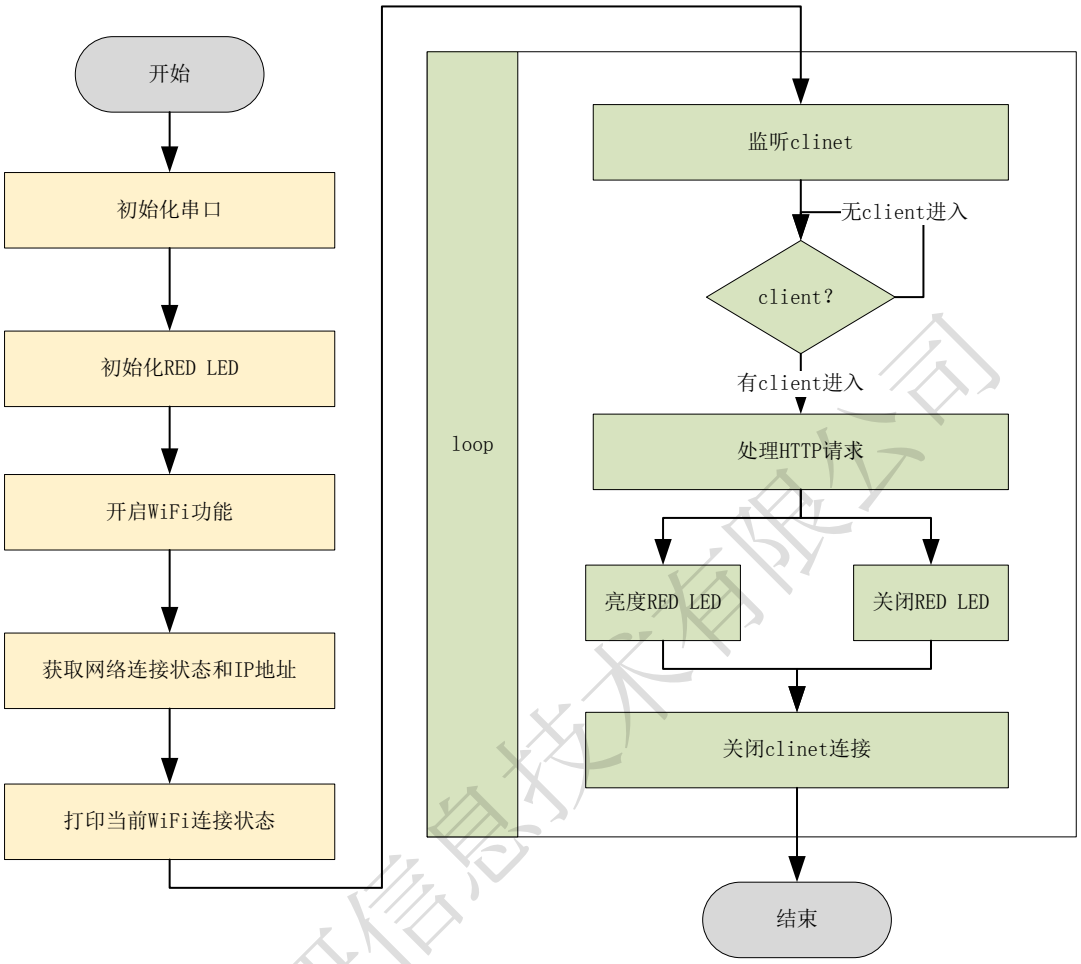


图 2-7 SimpleWebServerWiFi 例程流程图

实验过程中需要将 ssid 和 password 修改成正在使用的 WiFi 的 ssid 和 password。需修改代码如下。修改完成后即可下载使用。

```
// your network name also called SSID
char ssid[]="energia";
// your network password
char password[]="supersecret";
```

例程文件“MelodySimplehtml”中提供的“Melody_Simplehtml.h”和“Melody_Simplehtml.cpp”已经实现了图 2-7 所示的功能，用户在使用过程中仅需调用两个库函数即可。具体使用方式可见每个模块例程介绍中的“实验例程”章节。

表 2-1 beginServer()函数

驱动库函数	void beginServer(WiFiServer &server, char* ssid,char* pwd)
说明	开始 WebSever 功能
使用范例	beginServer(server,”ssid”,”password”);

表 2-2 handlexxxRequest()函数

驱动库函数	void handlexxxRequest(WiFiServer &server, WiFiClient &client)

说明	处理模块的 HTTP 情况，”xxx”需要替换成具体模块的函数名 如高亮 LED 模块为 HandleLEDRequest
使用范例	handlexxxRequset(server,client), handleLEDRequest(server,client)

2.3 实验例程操作视频

在艾研官网上可直接观看，地址：http://www.hpati.com/IOTKIT/

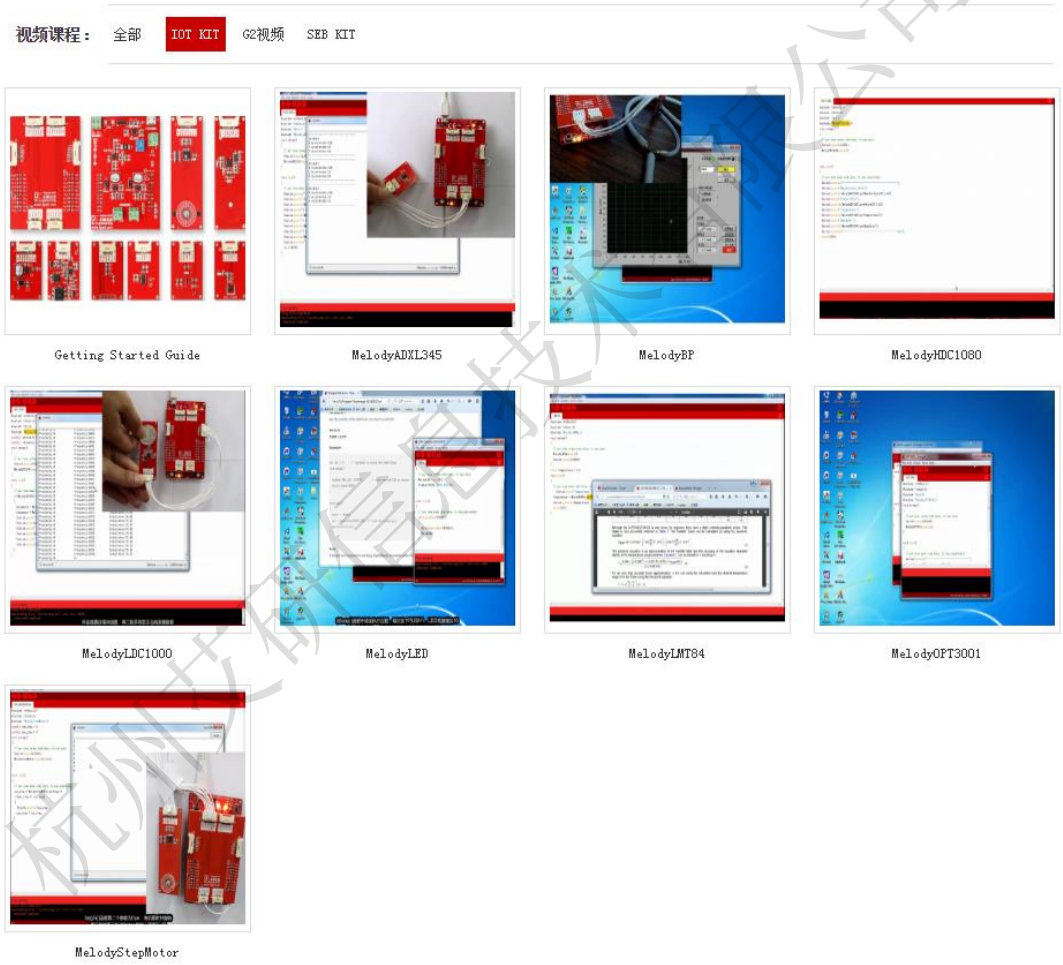


图 2-8 视频资源

3 实验套件模块

本章将介绍各个实验例程中的知识点和注意事项：包括套件模块接口说明，原理介绍以及软件实现。

套件使用之前需要查看《AY-IOT 连接使用须知》

对应网址：http://www.hpati.com/ay_doc_download/。

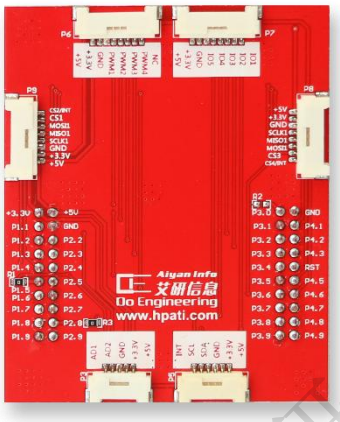
若未仔细了解《套件连接须知》，在之后的使用过程中将会产生异常状态：如 cc3200 出厂时排针处有部分端口未使用 0 欧姆电阻连接，而实验套件部分模块需要使用这些端口。

在下文介绍各个模块时假设用户已经了解《AY-IOT 连接使用须知》中的各个事项。

3.1 转接板

物联网使用套件各个模块和 cc3200 之间的连接需要使用到转接板。转接板将 cc3200 的 boostpack 端口映射成各传感器模块接口。转接板和对外端口如下：

表 3-1 转接板

转接板	接口
	1 对 boostpack 接口，连接 cc3200 1 组 GPIO 接口，包含 5 个 IO 口 1 组 PWM 接口，包含 4 个 PWM 口 2 组 SPI 通信接口 1 组 I2C 通信接口 1 组 AD 接口，包含 2 个 AD 采集通道

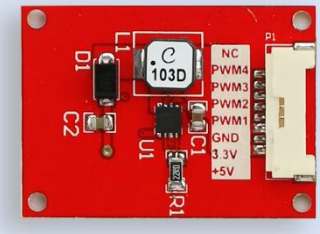
3.2 高亮 LED 驱动模块（MelodyLED）

3.2.1 模块

高亮 LED 驱动模块使用 PWM 接口，模块如下

表 3-2 高亮 LED 驱动模块

高亮 LED 驱动模块	概况



模块使用芯片 TPS61043 驱动 LED
使用 PWM 进行控制，可调节 LED 亮度

3.2.2 实现原理及原理图简介

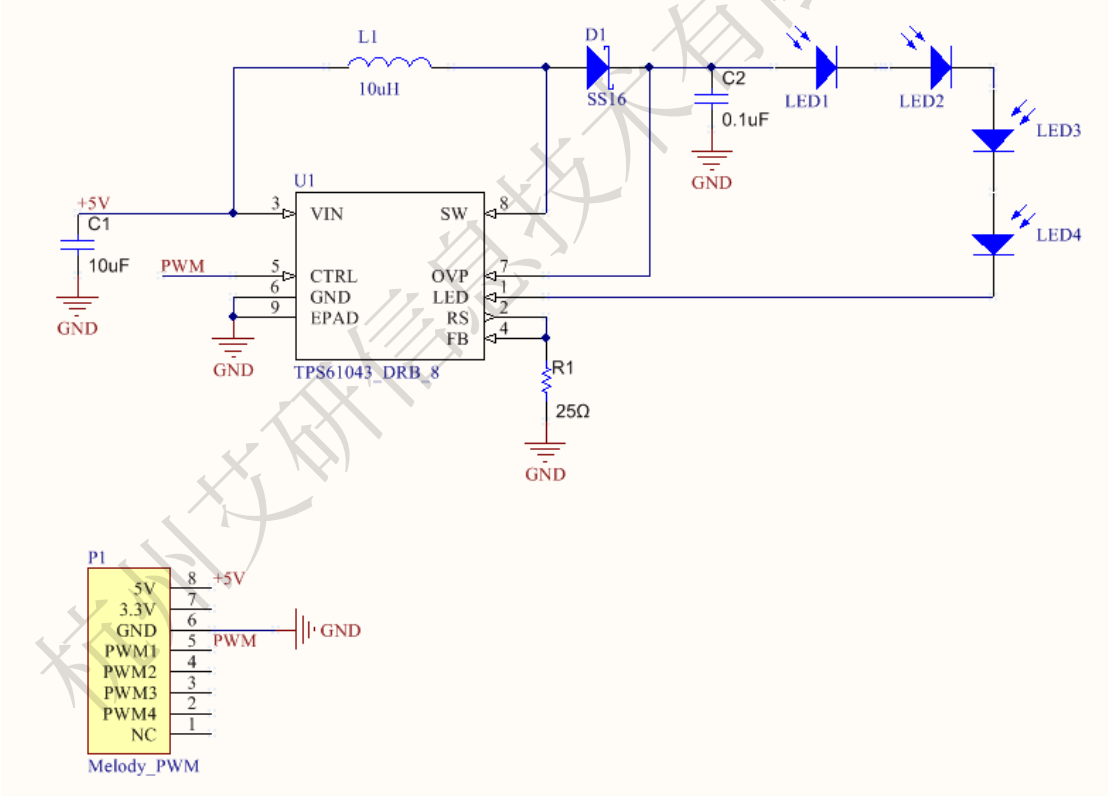


图 3-1 模块原理图

TPS61043 是 TI 的一款具有恒定电流输出的高频升压转换器，本设计中用于驱动 4 颗 LED 灯，LED 上的电流，通过外部检测电阻 R1 来设置，并由反馈引脚（FB）直接调节，FB 将传感电阻 RS 上的电压调节为 252 mV（典型值）。得 Iout=252mV/R1。

为了控制 LED 亮度，可以通过向控制引脚（CTRL）施加 100Hz 至 50kHz 的频率范围的 PWM（脉宽调制）信号来脉冲化 LED 电流。

3.2.3 软件实现

高亮 LED 驱动模块使用 PWM 驱动 LED，Energia 已经实现了 PWM 库函数，用户直接使用即可。PWM 库函数可在 http://energia.nu/reference/analogwrite/查看。模块留有 4 个 PWM 接口，根据原理图可以发现，模块实际仅使用了 PWM1 接口，驱动库也是根据 PWM1 实现的。

表 3-3 analogWrite()函数

Energia 库函数	voidanalogWrite(uint8_t pin,int val)
说明	指定 pin 端口输出 val/255 占空比，频率固定为 490Hz 的 PWM
使用范例	`analogWrite(4,128)`

函数源代码可在【Energia 安装目录】->hardware->cc3200->cores->cc3200->wiring_analog.c 中查看。实现如图 3-2：

```
void analogWrite(uint8_t pin,int val){
/* duty cycle(%) = val / 255;
    * Frequency of 490Hz specified by Arduino API */
uint8_t timer = digitalPinToTimer(pin);
if(timer == NOT_ON_TIMER)
return;
if(val ==0){
       pinMode(pin, OUTPUT);
       digitalWrite(pin, LOW);
return;
}
if(val >=255){
       pinMode(pin, OUTPUT);
       digitalWrite(pin, HIGH);
return;
}
   PWMWrite(pin,255, val,490);
}
```

图 3-2analogWrite()库函数

3.2.3.1 LED 驱动库实现

驱动库使用 C++实现，源代码可直接查看 MelodyLED 文件下的 Melody_LED.h 和 Melody_LED.cpp 文件。下面简单介绍库函数。

表 3-4 begin()函数

驱动库函数	void begin()
说明	初始化 LED，设置 LED 输出亮度为默认值 0
使用范例	`MelodyLED.begin();`

表 3-5 begin()函数

驱动库函数	void begin(uint8_t bright)
说明	初始化 LED，设置 LED 输出亮度为 bright，范围为 0~255
使用范例	MelodyLED .begin(10);

表 3-6 setBright()函数

驱动库函数	void setBright(uint8_t bright)
说明	设置 LED 亮度
使用范例	MelodyLED.setBright(20);

表 3-7 getBright 函数

驱动库函数	uint8_t getBright()
说明	获取 LED 亮度
使用范例	uint8_t bright = MelodyLED .getBright();

3.2.3.2 实验例程

实验例程操作及结果展示可浏览视频资料：http://www.hpati.com/product_videos/v79.html

例程 1，LED.ino：点亮 LED。例程代码如图 3-3，仅需要在 setup 函数出使用 MelodyLED.begin()函数即可。其中 MelodyLED 定义在 Melody_LED.cpp 文件中。

```
#include <stdbool.h>
#include <stdint.h>
#include <Melody_LED.h>

void setup()
{
// put your setup code here, to run once:
  MelodyLED.begin(10);
}

void loop()
{
// put your main code here, to run repeatedly:

}
```

图 3-3 点亮 LED 例程

例程 2，LEDWithButton.ino:通过按键调节 LED 亮度。程序流程图如图 3-4 所示，程序代码如图 3-5 所示。

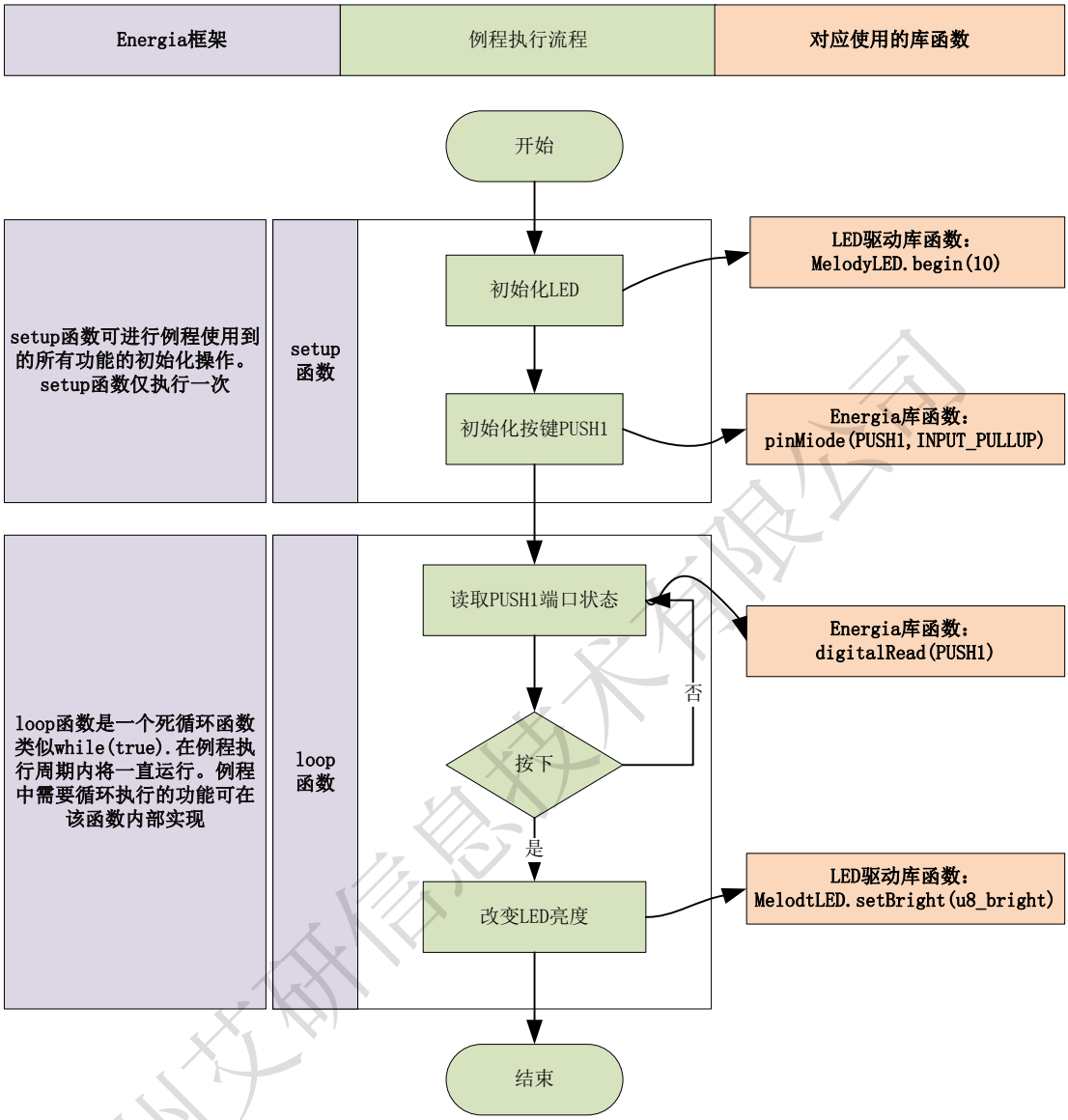


图 3-4LEDWithButton 例程实现流程图

```
#include <stdbool.h>
#include <stdint.h>
#include <Melody_LED.h>
uint8_t u8_bright =10;
void setup()
{
// put your setup code here, to run once:
  MelodyLED.begin(10);
  pinMode(PUSH1,INPUT_PULLUP);
}

void loop()
{
```

```
// put your main code here, to run repeatedly:
if(digitalRead(PUSH1))
{
while(digitalRead(PUSH1));
    u8_bright +=10;
    MelodyLED.setBright(u8_bright);
}
}
```

图 3-5LEDWithButton 例程

例程 3,SimpleWebLED.ino:通过网络改变 LED 亮度。cc3200 设置为 webserver 模式，用户可是有浏览器访问 cc3200，实验过程中 cc3200 的 IP 地址通过串口打印至 PC 端。Energia 开发工具提供了串口监视工具。程序例程图如图 3-6 所示，网络驱动库函数可见 MelodySimplehtml 文件夹下的 Melody_Simplehtml.h 和 Melody_Simplehtml.cpp 文件。例程代码如图 3-7 所示。

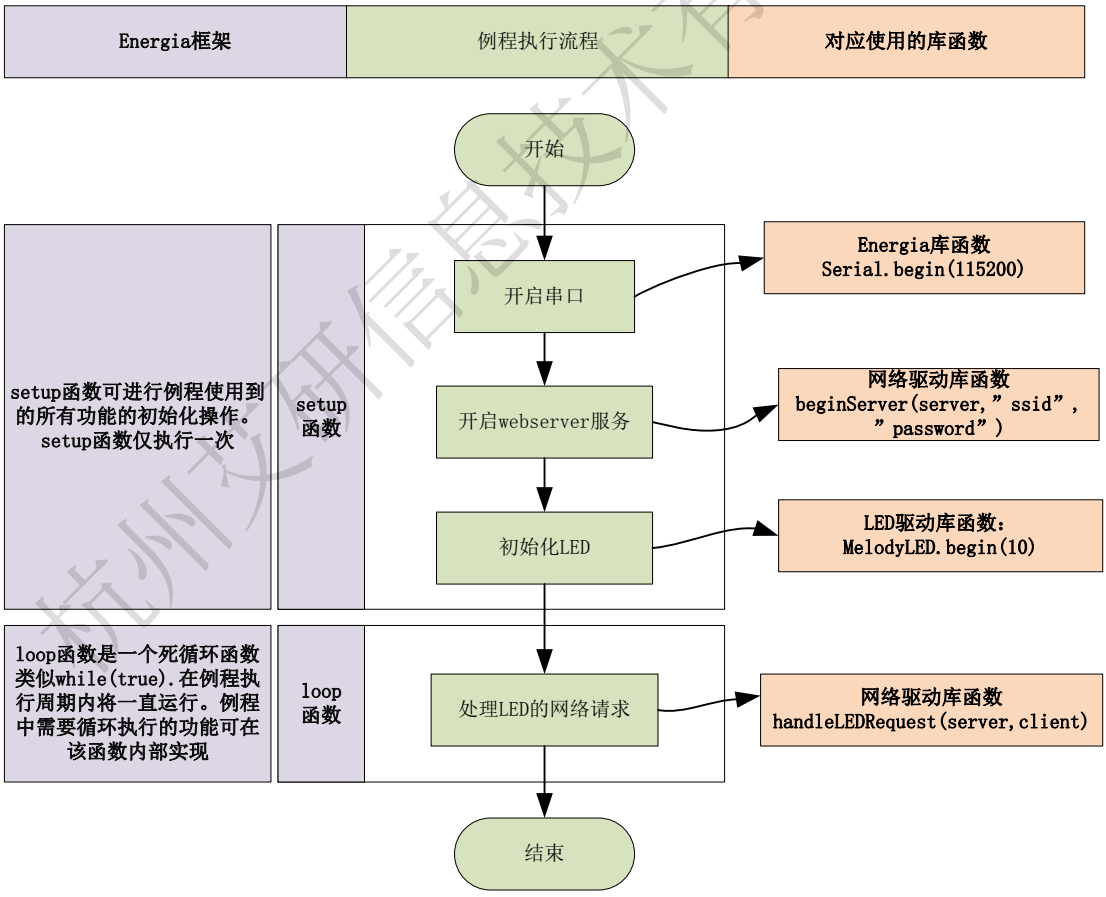


图 3-6 SimpleWebLED 例程流程图

```
#include <stdbool.h>
#include <stdint.h>
#include <WiFi.h>
#include <Melody_LED.h>
```

```
#include "Melody_Simplehtml.h"

WiFiServer server(80);
WiFiClient client;

void setup()
{
// put your setup code here, to run once:
  Serial.begin(115200);
  beginServer(server,"ssid","password");
  MelodyLED.begin(10);
}

void loop()
{
  handleLEDRequest(server,client);
}
```

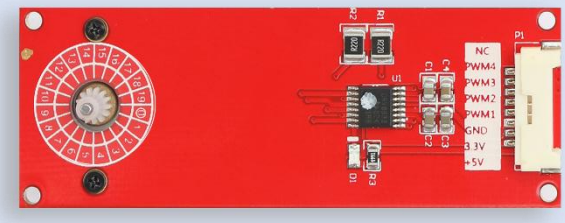
图 3-7SimpleWebLED 例程

3.3 步进电机模块（MelodyStepMotor）

3.3.1 模块

步进电机模块使用 PWM 接口进行驱动。模块如下

表 3-8 步进电机模块

步进电机模块	概况
	1 模块使用 DRV8833 驱动步进电机 2 使用 4 路 PWM 进行驱动，可通过软件改变 PWM 的输出频率和 4 路 PWM 的对应相位，从而改变步进电机的工作速度和方向。

3.3.2 实现原理及原理图简介

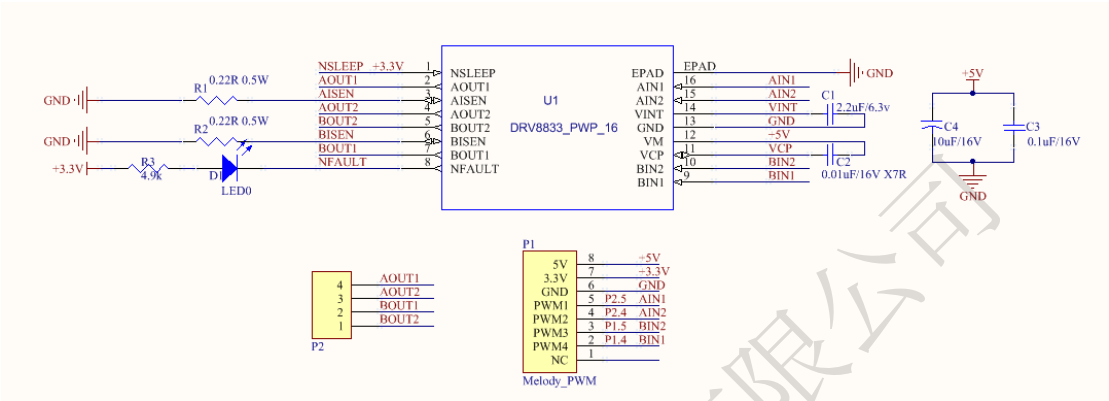


图 3-8 步进电机模块原理图

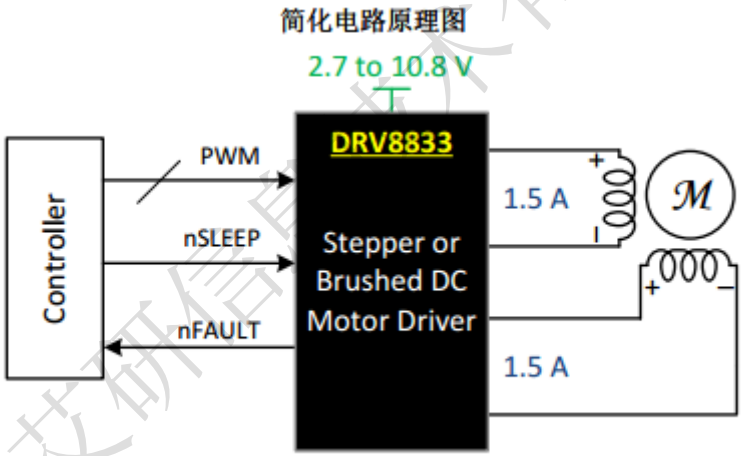


图 3-9 简化原理图

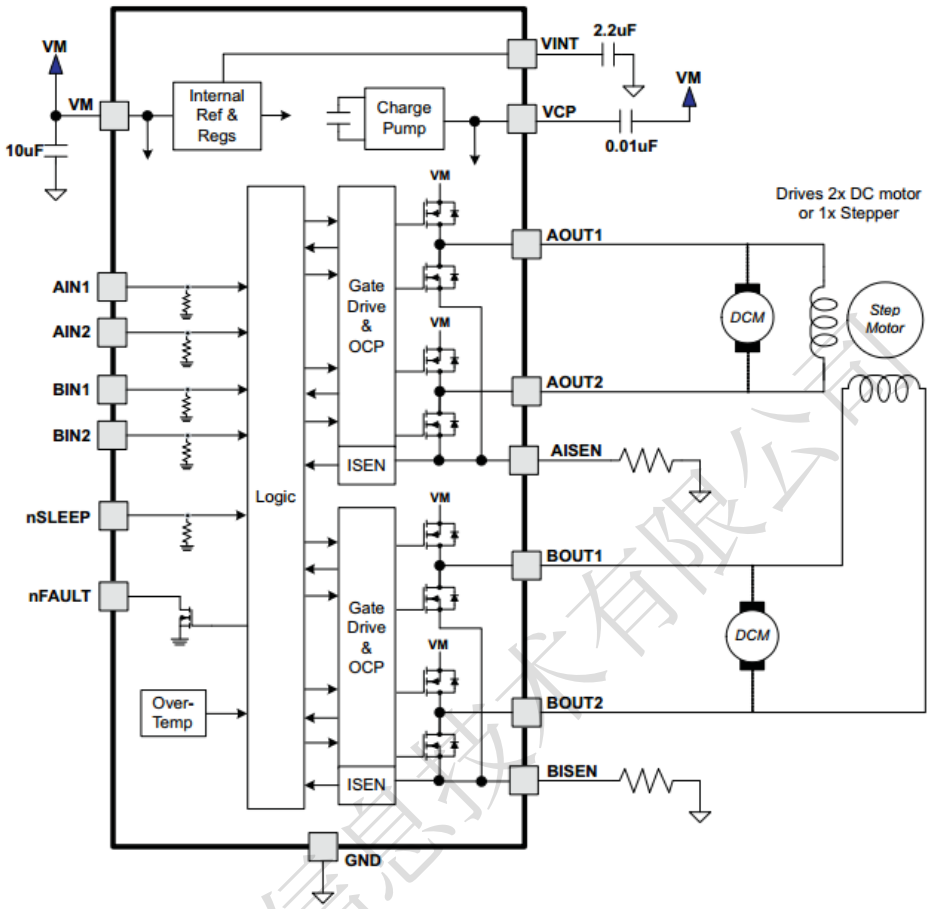


图 3-10 芯片内部原理

本模块采用 TI 的双通道 H 桥电流控制电机驱动器 DRV8833，用于驱动步进电机。每个 H 桥的输出驱动器模块由配置为 H 桥的 N 沟道功率 MOSFET 组成，用于驱动电机绕组。每个 H 桥均具备调节或限制绕组电流的电路。步进电机的步进量以及转速可通过 4 路 PWM 来调节。

3.3.3 软件实现

根据原理图可知步进电机使用 4 路 PWM 驱动控制，该模块中的 PWM 实现与 3.2 节高亮 LED 驱动模块（MelodyLED）中的 PWM 实现不同，此处 PWM 实现使用 cc3200SDK 中的 timer_if.c 和 timer_if.h 实现（这两个文件位于 MelodyStepMotor 下的 utility 文件夹中），因为 Energia 实现的库无法同时输出 4 路 PWM 波，具体可看 Wiring_analog.c 文件中库函数 PWMWrite()的实现。

3.3.3.1 步进电机驱动库实现

驱动库使用 C++实现，源代码可直接查看 MelodyStepMotor 文件下的 Melody_StepMotor.h 和 Melody_StepMotop.cpp 文件。下面简单介绍库函数。`MelodyStepMotor` 定义在 `Melody_StepMotor.cpp` 中。

表 3-9 begin()函数

驱动库函数	void begin()
说明	初始化步进电机，设置默认最小速度，顺时针转动,起始步进位置为 0
使用范例	MelodyStepMotor.begin();

表 3-10 begin()函数

驱动库函数	void begin(uint8_t speed)
说明	初始化步进电机，设置速度为 speed，速度范围 1~10，顺时针转动,起始步进位置为 0
使用范例	MelodyStepMotor.begin(1);

表 3-11 begin()函数

驱动库函数	void begin(uint8_t speed, bool direction)
说明	初始化步进电机，设置速度为 speed，速度范围 1~10，转动方向为 direction：true 为顺时针，false 为逆时针,起始步进位置为 0
使用范例	MelodyStepMotor.begin(1,true);

表 3-12 begin()函数

驱动库函数	void begin(uint8_t speed, bool direction, uint8_t step)
说明	初始化步进电机，设置速度为 speed：速度范围 1~10，转动方向为 direction：true 为顺时针，false 为逆时针,起始步进位置为 step：范围 0~19
使用范例	MelodyStepMotor.begin(1,true,0);

表 3-13 changeSpeed()函数

驱动库函数	void changeSpeed(uint8_t speed)
说明	改变步进电机速度，速度范围为 1~10
使用范例	changeSpeed(5);

表 3-14 getSpeed()函数

驱动库函数	uint8_t getSpeed()
说明	获取步进电机当前运转速度，速度范围为 1~10
使用范例	uint8_t speed = MelodyStepMotor .getSpeed();

表 3-15 changeDirection()函数

驱动库函数	void changeDirection(bool direction)
说明	改变步进电机转动方向，true：顺时针方向，false：逆时针方向
使用范例	MelodyStepMotor .changeDirection(true);

表 3-16getDirection()函数

驱动库函数	bool getDirection()
说明	获取步进电机转动方向，true：顺时针方向，false：逆时针方向
使用范例	bool direction = MelodyStepMotor.getDirection();

表 3-17 getSteps()函数

驱动库函数	uint8_t getSteps()
说明	获取步进电机步进值，范围 0~19
使用范例	uint8_t step = MelodyStepMotor. getSteps();

3.3.3.2 实验例程

实验例程操作及结果展示可浏览视频资料：http://www.hpati.com/product_videos/v81.html

例程 1，SimpleStepMotor.ino：启动步进电机顺时针转动，通过串口上传当前步进位置。例程流程图如图 3-11，例程源代码如图 3-12。

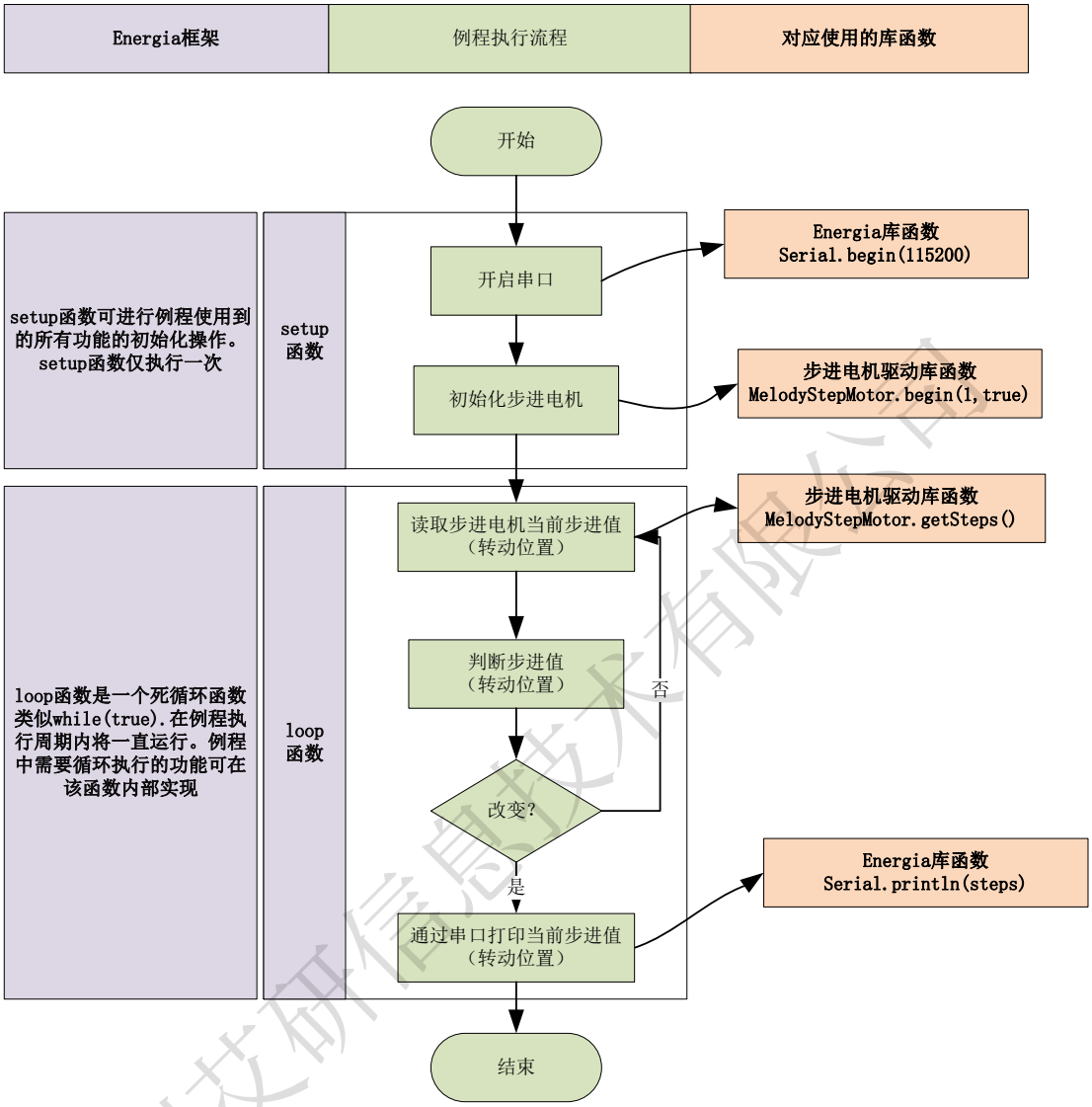


图 3-11 SimpleStepMotor 例程流程图

```
#include <stdbool.h>
#include <stdint.h>
#include "Melody_StepMotor.h"
uint8_t cur_step =0;
uint8_t per_step =0;
void setup()
{
// put your setup code here, to run once:
  Serial.begin(115200);
  MelodyStepMotor.begin(1,true);
}

void loop()
{
// put your main code here, to run repeatedly:
```

```
  cur_step = MelodyStepMotor.getSteps();
if(per_step != cur_step)
{
    Serial.println(cur_step);
    per_step = cur_step;
}
}
```

图 3-12 SimpleStepMotor 例程源代码

例程 2，SimpleWebStepMotor.ino:通过网络改变步进电机转动方向和速度。cc3200 设置为 webserver 模式，用户可是有浏览器访问 cc3200，实验过程中 cc3200 的 IP 地址通过串口打印至 PC 端。Energia 开发工具提供了串口监视工具。网络驱动库函数可见 MelodySimplehtml 文件夹下的 Melody_Simplehtml.h 和 Melody_Simplehtml.cpp 文件。程序例程图如图 3-13 所示，例程代码如图 3-14 所示。

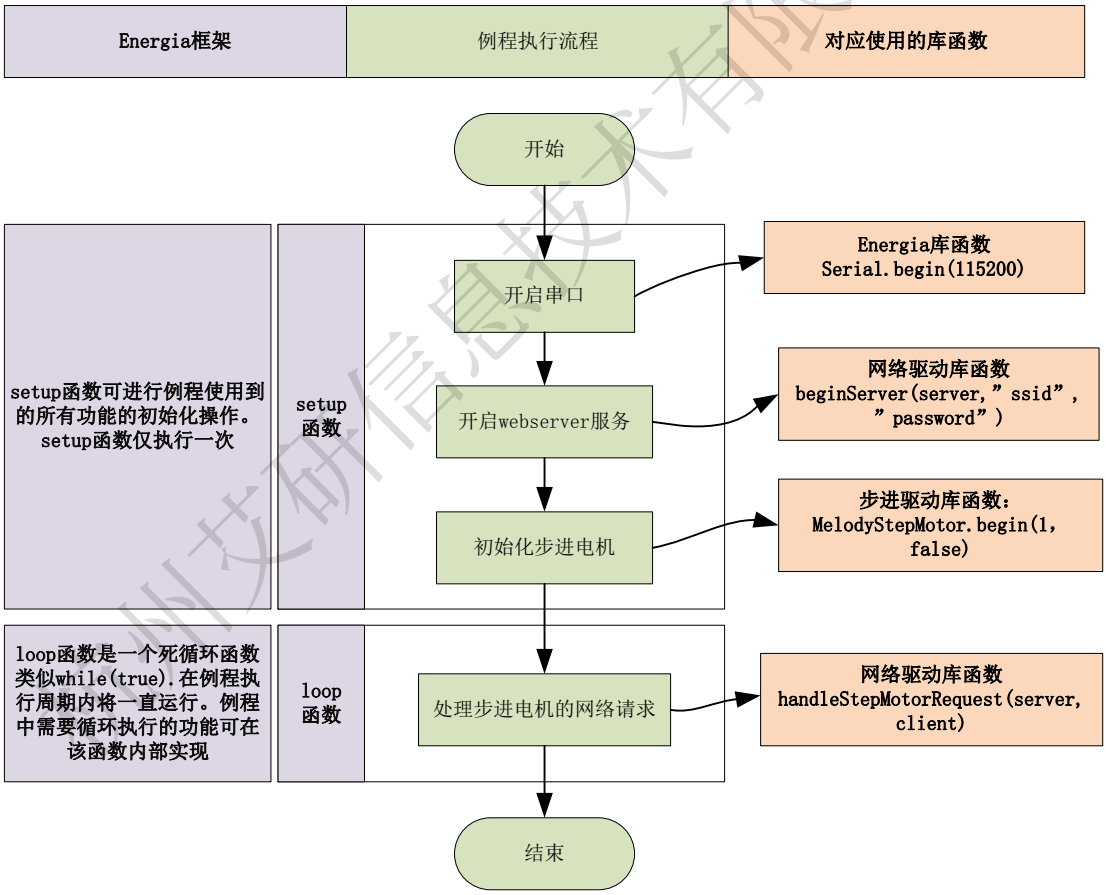


图 3-13 SimpleWebStepMotor 例程流程图

```
#include <stdbool.h>
#include <stdint.h>
#include <WiFi.h>
#include "Melody_StepMotor.h"
#include "Melody_Simplehtml.h"

WiFiServer server(80);
```

```
WiFiClient client;
void setup()
{
// put your setup code here, to run once:
  Serial.begin(115200);
  beginServer(server,"ssid","password");
  MelodyStepMotor.begin(1,false);
}

void loop()
{
// put your main code here, to run repeatedly:
  handleStepMotorRequest(server,client);
}
```

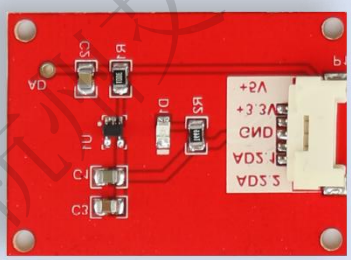
图 3-14 SimpleWebStepMotor 例程源代码

3.4 模拟温度模块（MelodyLMT84）

3.4.1 模块

模拟温度模块使用 AD 接口进行温度采集。模块如下

表 3-18 模拟温度采集模块

模拟温度采集模块	概况
	模块使用芯片 LMT84 使用 ADC 接口采集温度数据 测量精度 $\pm 0.4^{o}C$ 输出受到短路保护

3.4.2 实现原理及原理图简介

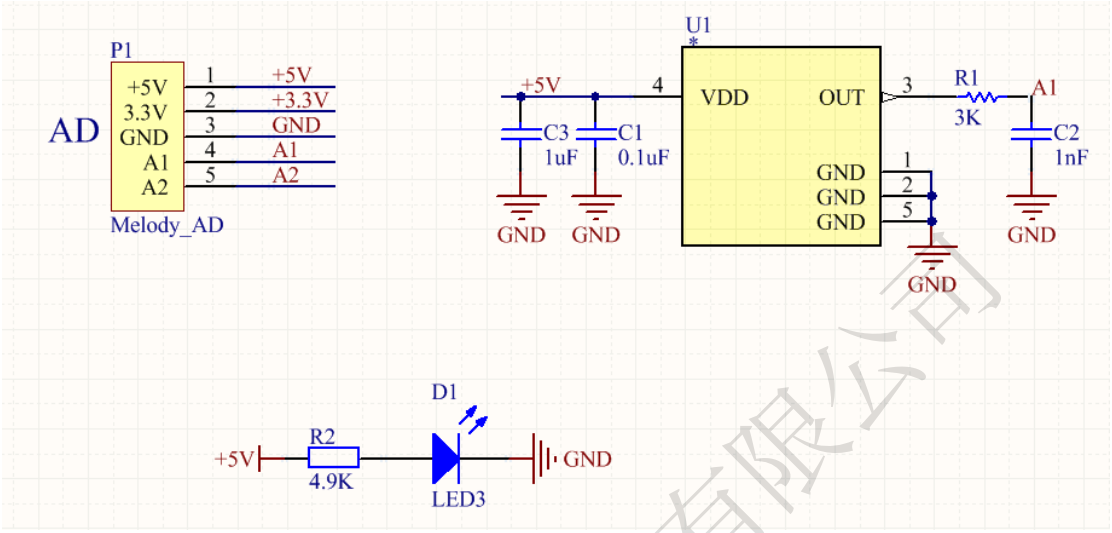


图 3-15 模拟温度模块原理图

LMT84 是高精度 CMOS 集成电路温度传感器，此传感器具有与温度成线性反比例关系的模拟电压输出，再用 CC3200 的内部 ADC，即可获得 LMT84 的模拟电压输出，进而转换成温度。测量范围为-50℃~150℃。

测量温度与模拟电压关系为：T=（Vout-1035mV）/（-5.5mV/℃）

详细的对应关系，可参考 LMT84 的手册 http://www.ti.com.cn/cn/lit/ds/symlink/lmt84.pdf

3.4.3 软件实现

模拟温度模块使用 AD 进行数据采集，Energia 已经实现了 ADC 库函数，用户直接使用即可。ADC 库函数可在 http://energia.nu/reference/analogread/查看，函数名为 analogRead(pin)。

表 3-19 analogRead()函数

Energia 库函数	int analogRead (uint8_t pin)
说明	读取指定端口电压值。
使用范例	**analogRead(A1);A1 为 cc3200 ADC 功能端口，定义可在【Energia 安装目录】->hardware->cc3200->variants->Launchpad->pins_energia.h**

函数源代码可在【Energia 安装目录】->hardware->cc3200->cores->cc3200->wiring_analog.c 中查看,库函数实现如

```
uint16_t analogRead(uint8_t pin)
{
uint16_t channel,val;
uint16_t pinNum = digitalPinToPinNum(pin);

switch(pinNum){
```

```
case PIN_57:{channel = ADC_CH_0;}break;
case PIN_58:{channel = ADC_CH_1;}break;
case PIN_59:{channel = ADC_CH_2;}break;
case PIN_60:{channel = ADC_CH_3;}break;
default:return0;
}

    while(ADCFIFOLvlGet(ADC_BASE, channel)){
    // flush the channel's FIFO if not empty
        ADCFIFORead(ADC_BASE, channel);
}

    PinTypeADC(pinNum,0xFF);
    ADCChannelEnable(ADC_BASE, channel);
    ADCTimerConfig(ADC_BASE,0x1ffff);
    ADCTimerEnable(ADC_BASE);
    ADCEnable(ADC_BASE);

while(!ADCFIFOLvlGet(ADC_BASE, channel));
    val = ADCFIFORead(ADC_BASE, channel)&0x3FFF;

    ADCDisable(ADC_BASE);
    ADCChannelDisable(ADC_BASE, channel);
    ADCTimerDisable(ADC_BASE);

    val = val >>2;
return mapResolution(val,12, _readResolution);
}
```

图 3-16 Energia 库函数 analogRead()函数

3.4.3.1 模拟温度驱动库实现

驱动库使用 C++实现，源代码可直接查看 MelodyLMT84 文件下的 Melody_LMT84.h 和 Melody_LMT84.cpp 文件。下面简单介绍库函数。MelodyLMT84 定义在 Melody_LMT84.cpp

表 3-20 begin()函数

驱动库函数	void begin()
说明	初始化温度采集端口设置,默认使用 A1 端口
使用范例	MelodyLMT84.begin();

表 3-21 begin()函数

驱动库函数	void begin(uint8_t port)

说明	初始化指定端口进行温度采集
使用范例	`MelodyLMT84.begin(A1);`

表 3-22 getTemperature()函数

驱动库函数	float getTemperature()
说明	获取温度值
使用范例	float temperature = `MelodyLMT84.getTemperature();`

3.4.3.2 实验例程

实验例程操作及结果展示可浏览视频资料：http://www.hpati.com/product_videos/v78.html

例程 1，LMT84.ino:读取模拟温度模块温度值并且通过串口上传。例程流程图如图 3-17，例程源程序如图 3-18：

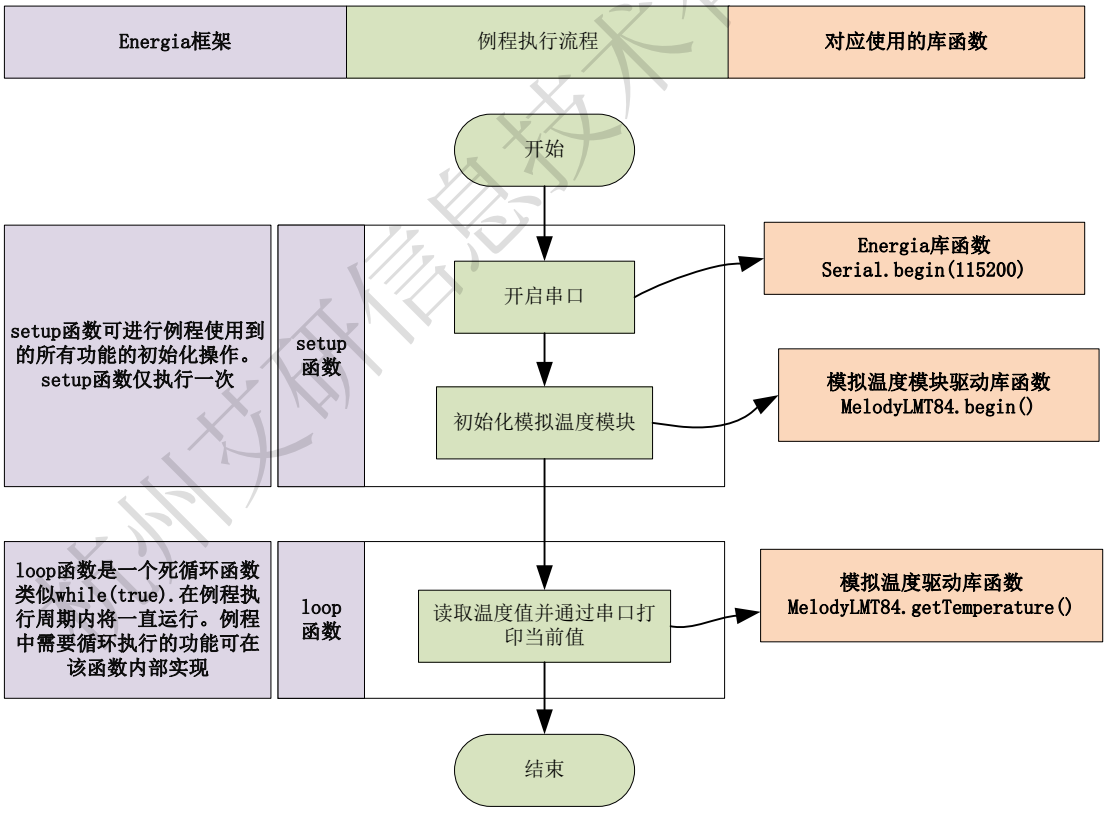


图 3-17 LMT84 例程流程图

```
#include <stdbool.h>
#include <stdint.h>
#include "Melody_LMT84.h"
void setup()
{
// put your setup code here, to run once:
```

```
  Serial.begin(115200);
  MelodyLMT84.begin();

}
float temperature =0.0;
void loop()
{
// put your main code here, to run repeatedly:
Serial.print("temperature:");
  temperature = MelodyLMT84.getTemperature();
  Serial.println(temperature);
  delay(200);
}
```

图 3-18 LMT84 例程源代码

例程 2，SimpleWebLMT84.ino：通过网络读取温度值。cc3200 设置为 webserver 模式，用户可是有浏览器访问 cc3200，实验过程中 cc3200 的 IP 地址通过串口打印至 PC 端。Energia 开发工具提供了串口监视工具。程序例程图如所示，网络驱动库函数可见 MelodySimplehtml 文件夹下的 Melody_Simplehtml.h 和 Melody_Simplehtml.cpp 文件。例程流程图如图 3-19，例程代码如图 3-20 所示。

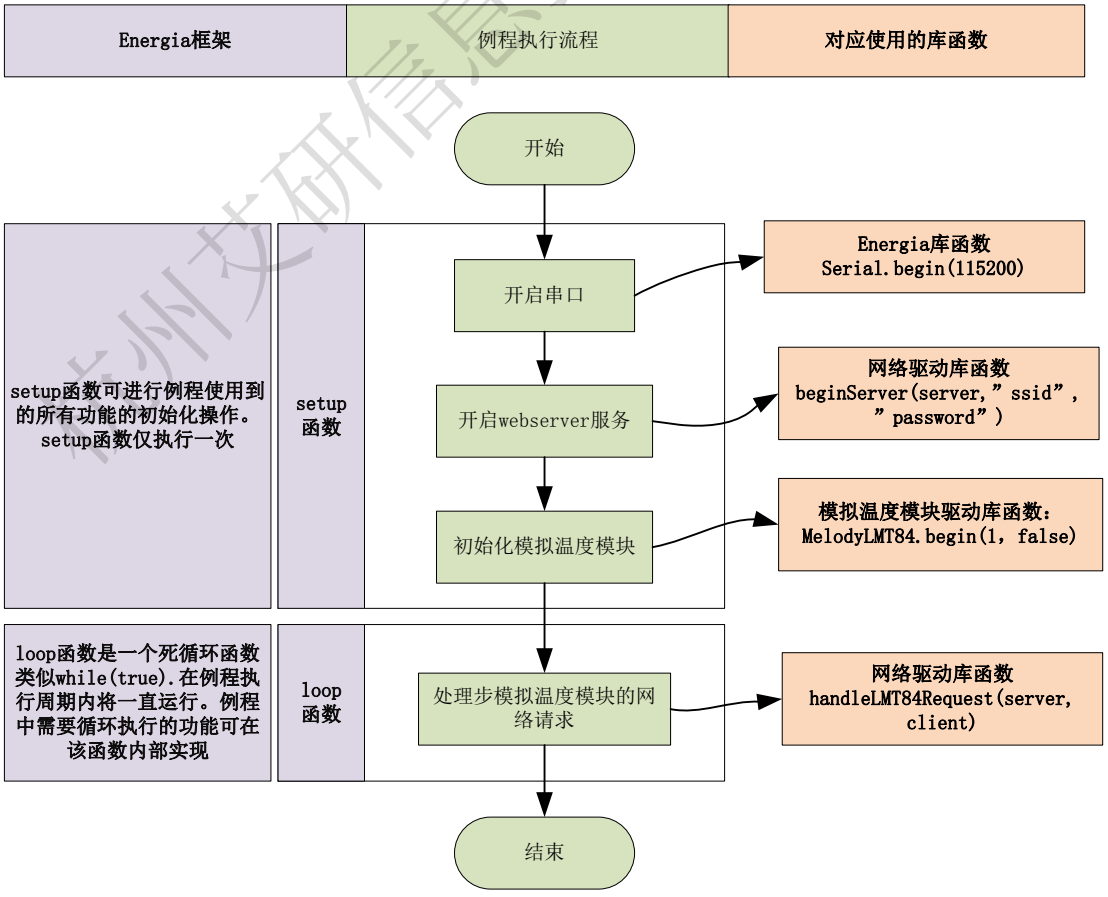


图 3-19 SimpleWebLMT84 例程流程图

```
#include <stdbool.h>
#include <stdint.h>
#include <WiFi.h>
#include "Melody_LMT84.h"
#include "Melody_Simplehtml.h"

WiFiServer server(80);
WiFiClient client;
void setup()
{
// put your setup code here, to run once:
  Serial.begin(115200);
  beginServer(server,"ssid","password");
  MelodyLMT84.begin();
}

void loop()
{
// put your main code here, to run repeatedly:
  handleLMT84Request(server,client);
}
```

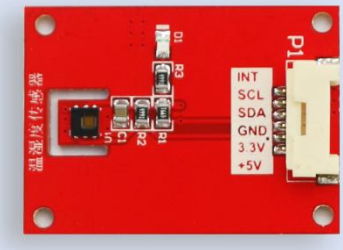
图 3-20 SimpleWebLMT84 例程源代码

3.5 温湿度传感器模块（MelodyHDC1080）

3.5.1 模块

温湿度传感器模块使用 I2C 接口与 cc3200 进行通信，可读取模块的温度和湿度数据。模块如下：

表 3-23 温湿度传感器模块

温湿度传感器模块	概况
	1 模块使用芯片 HDC1080 2 使用 I2C 接口，读取模块的温湿度数据 3 相对湿度精度达到±2% 4 温度精度达到$\pm 0.2^{o}C$ 5 14 位测量分辨率

3.5.2 实现原理及原理图简介

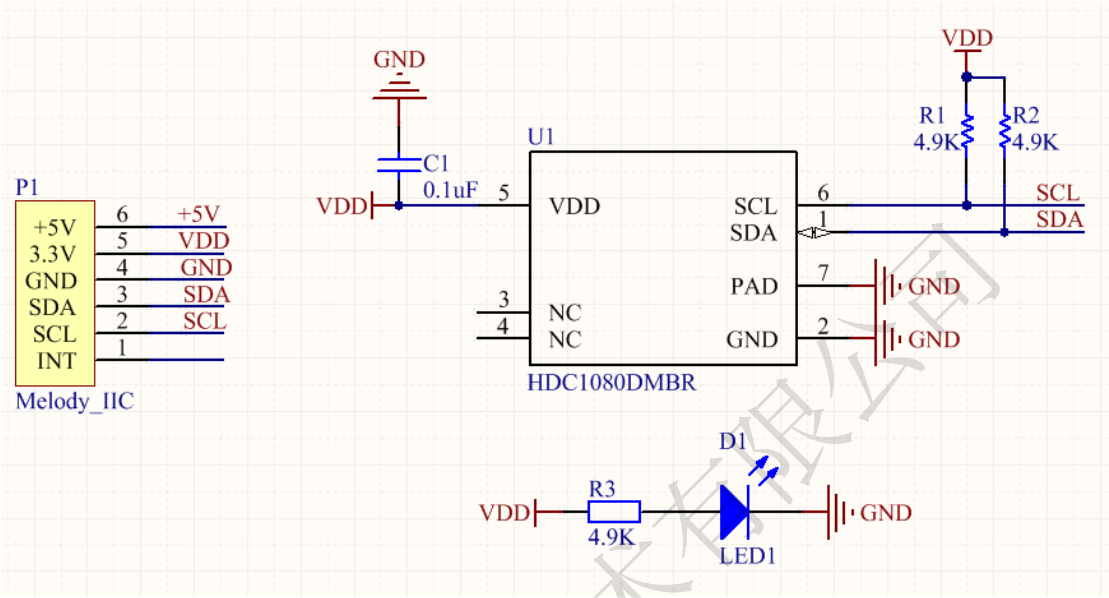


图 3-21 温湿度传感器模块原理图

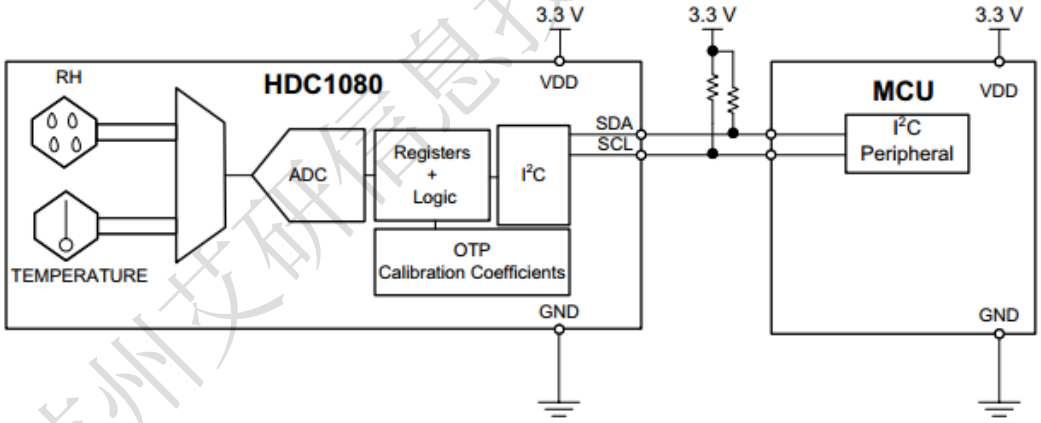


图 3-22 原理简化图

HDC1080 集成了温度传感器和数字湿度传感器，并能够以超低功耗提供出色的测量精度，相对湿度精度为±2%（典型值），温度精度为±0.2°C（典型值）。具有 14 位测量分辨率。通过 I2C 接口，与 MCU 连接即可，使用便利。

3.5.3 软件实现

温湿度传感器模块使用 I2C 接口，Energia 已经实现了 I2C 库函数，用户直接使用即可。I2C 库函数，可在 http://energia.nu/reference/wire/查看。I2C 库函数在使用上异于 3.2 章节“高亮 LED 驱动模块（MelodyLED）”中 PWM 库函数和 3.4 章节“模拟温度模块（MelodyLMT84）”中 AD 库函数的使用；上述两个章节中库函数使用仅需直接调用一个函数即可（如 analogWrite()函数和 analogRead()函数）。本章节模块使用 I2C 库函数，下面将简单介绍 I2C

库函数使用。

3.5.3.1 I2C 库函数

根据 Energia 官网上的介绍，I2C 库函数使用 C++实现，由多个成员函数构成，用户可直接使用库文件已经定义的“Wire”对象调用其成员函数，如 Wire.begin()。各成员函数如图 3-23 所示。

图 3-23 I2C 库函数

数据发送基本过程：

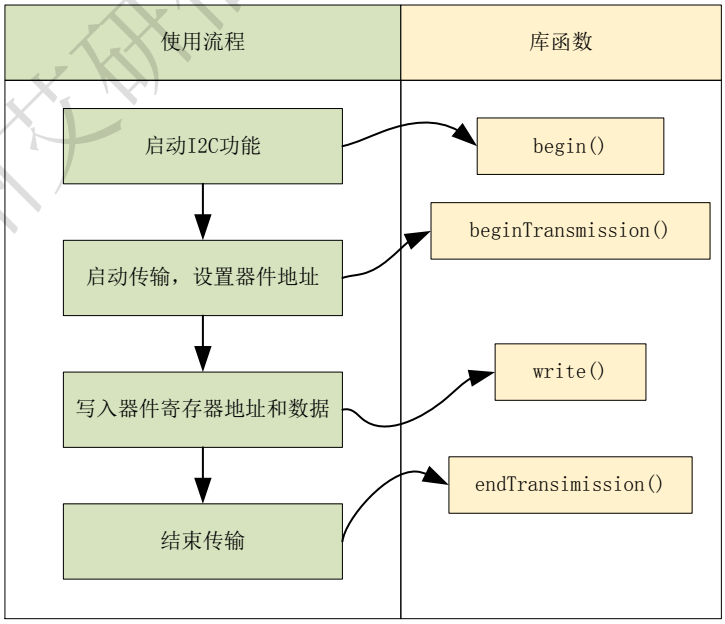


图 3-24 I2C 库函数使用——数据发送基本过程流程展示

程序实现可参考 Energia 提供的例程，可在 Energia 软件 File->Examples->Wire->master_writer 找到。代码如下：

```
#include <Wire.h>
```

```
void setup()
{
  Wire.begin();// join i2c bus (address optional for master)
}

byte x =0;

void loop()
{
  Wire.beginTransmission(4);// transmit to device #4
  Wire.write("x is ");// sends five bytes
  Wire.write(x);// sends one byte
  Wire.endTransmission();// stop transmitting

  x++;
  delay(500);
}
```

图 3-25 I2C 库函数使用——数据发送基本过程代码实现

数据读取基本过程：

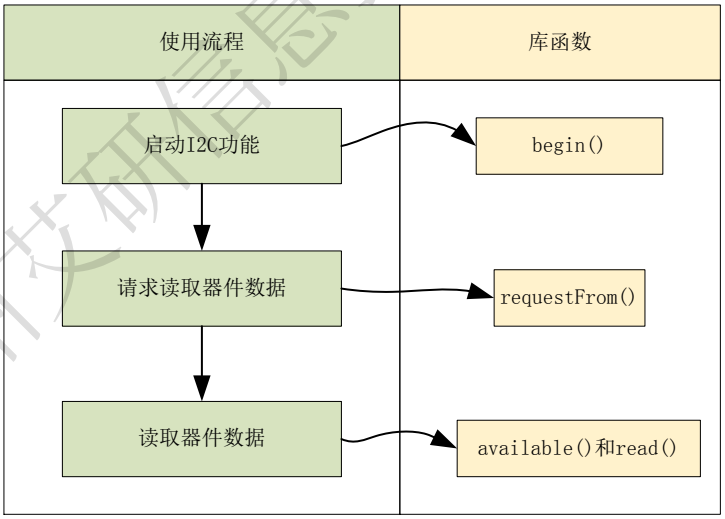


图 3-26 I2C 库函数使用——数据读取基本过程流程展示

程序实现可参考 Energia 提供的例程，可在 Energia 软件 File->Examples->Wire->master_writer 找到。代码如下：

```
#include <Wire.h>

void setup()
{
  Wire.begin();// join i2c bus (address optional for master)
  Serial.begin(9600);// start serial for output
}
```

```
void loop()
{
  Wire.requestFrom(2,6);// request 6 bytes from slave device #2

while(Wire.available())// slave may send less than requested
{
char c = Wire.read();// receive a byte as character
    Serial.print(c);// print the character
}
  delay(500);
}
```

图 3-27 I2C 库函数使用——数据读取基本过程代码实现

注意：I2C 库函数使用时，数据发送和数据读取过程需根据实际器件时序实现。

3.5.3.2 温湿度传感器驱动库实现

写时序

温湿度传感器模块使用 HDC1080，该芯片的写时序如下：

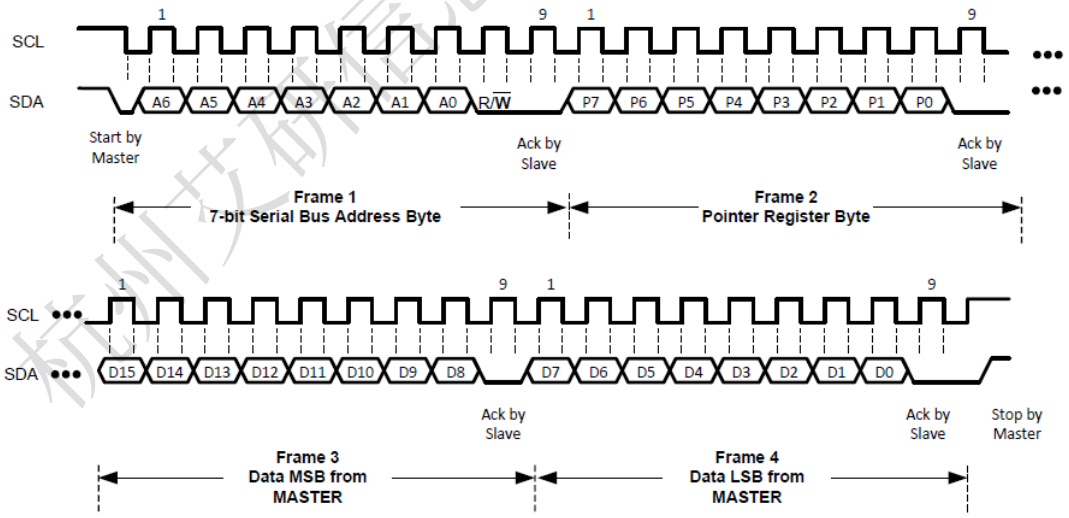


图 3-28 写时序

在实现时 cc3200 做为 Master，HDC1080 做为 slave。数据写入过程：第一步：cc3200 发送 HDC1080 的地址，根据芯片手册可知 HDC1080 的地址为 0b01000000（0x40）；第二步：发送需要写入的寄存器地址；第三步：发送两个字节长度的数据，先发送高字节数据，在发送低字节数据。

结合图 3-24 和图 3-28，写数据程序实现如下（不包括启动 I2C 功能，I2C 启动仅需调用 Wire.begin()即可，并且在使用过程中仅可被调用一次。）

```
/*****************************************************************
```

```
 * @brief:  写数据
 * @param:  regaddress：寄存器地址
 *          pdata，指向需要写入数据的指针
 *          length，数据长度
 * @return:  none
*********************************************************************/
void CMelodyHDC1080::write(uint8_t regaddress,uint8_t*pdata,uint8_t
                                length)
{
    Wire.beginTransmission(HDC_ADDRESS);
    Wire.write(regaddress);
    Wire.write(pdata,length);
    Wire.endTransmission();
}
```

图 3-29 写数据代码实现

读时序

HDC1080 读时序如下：

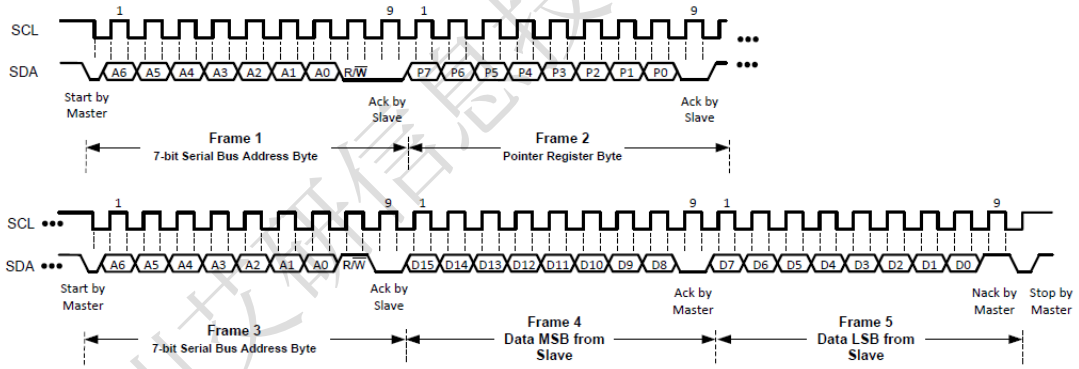


图 3-30 读时序

在实现时 cc3200 做为 Master，HDC1080 做为 slave。数据读取过程：第一步：cc3200 发送 HDC1080 的地址，根据芯片手册可知 HDC1080 的地址为 0b01000000（0x40）；第二步：发送需要读取数据的寄存器地址；第三步：再次发送 HDC1080 的地址，第四步：读取两个字节长度的数据，高字节在前，低字节在后。

结合图 3-26 图 3-24 和图 3-30 读时序，读数据程序实现如下（不包括启动 I2C 功能，I2C 启动仅需调用 Wire.begin()即可，并且在使用过程中仅可被调用一次。）

```
/*********************************************************************
 * @brief:  读取数据
 * @param:  regaddress:寄存器地址
 * @return: 读取的数据
*********************************************************************/
uint16_t CMelodyHDC1080::read(uint8_t regaddress)
{
uint16_t data =0;
```

```
    Wire.beginTransmission(HDC_ADDRESS);
    Wire.write(regaddress);
    Wire.endTransmission();

    delay(40);

    Wire.requestFrom(HDC_ADDRESS,2);
if(Wire.available()>=2)
{
        data = Wire.read()<<8;
        data |= Wire.read();
}
return data;
}
```

图 3-31 读数据代码实现

驱动库部分函数如下：类对象“MelodyHDC1080”定义在 Melody_HDC1080.cpp 文件中。

表 3-24 begin()函数

驱动库函数	void begin()
说明	启动 I2C 功能，初始化 HDC1080
使用范例	MelodyHDC1080.begin();

表 3-25 getTemperature()函数

驱动库函数	float getTemperature()
说明	获取温度值
使用范例	float temperature = MelodyHDC1080.getTemperature();

表 3-26 getHumidity()函数

驱动库函数	float getHumiduty()
说明	获取湿度值
使用范例	float humidity = MelodyHDC1080.getHumidity()

表 3-27 getManufacturerID()函数

驱动库函数	uint16_t getManufacturerID ()
说明	获取 Manufacturer ID
使用范例	uint16_t id = MelodyHDC1080. getManufacturerID()

表 3-28 getDeviceID()函数

驱动库函数	uint16_t getDeviceID()
说明	获取 Device ID
使用范例	uint16_t id = MelodyHDC1080. getDeviceID ()

3.5.3.3 实验例程

实验例程操作及结果展示可浏览视频资料：http://www.hpati.com/product_videos/v75.html

例程 1，HDC1080.ino：读取当前温度数据，使用串口打印。实例流程图如图 3-32，实例代码如图 3-33，

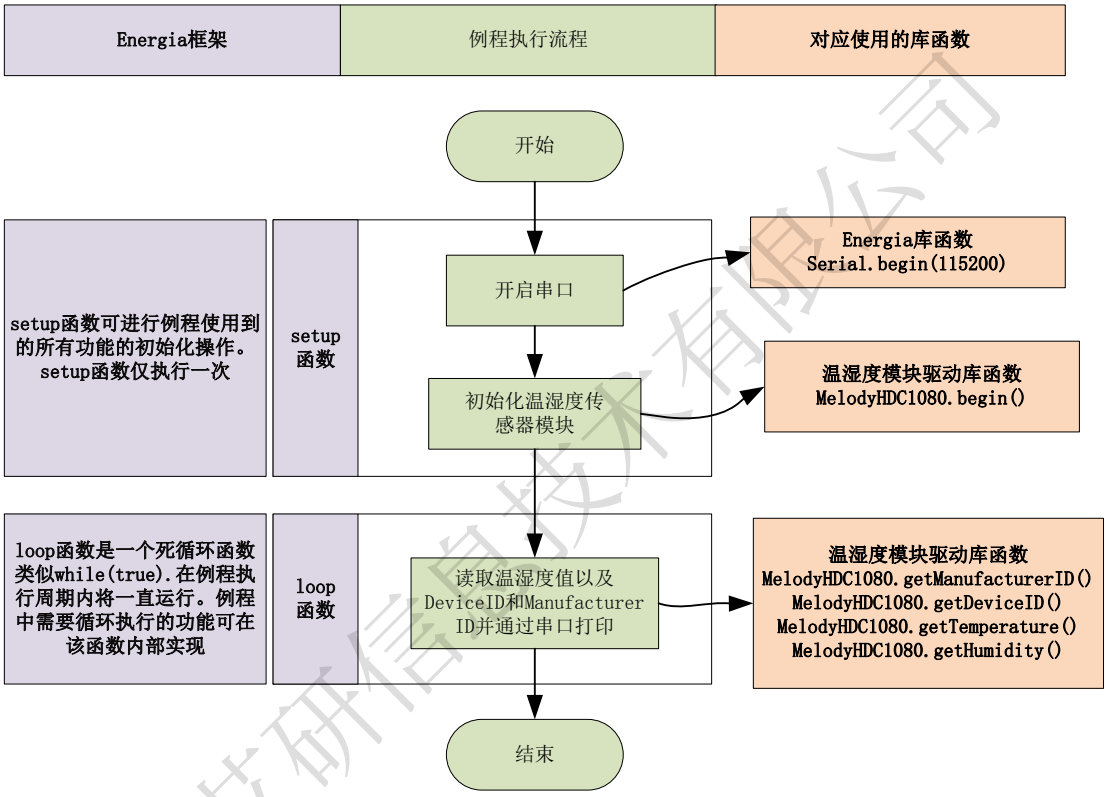


图 3-32 HDC1080 例程流程图

```
#include <stdbool.h>
#include <stdint.h>
#include "Wire.h"
#include "Melody_HDC1080.h"
void setup()
{
// put your setup code here, to run once:
  Serial.begin(115200);
  MelodyHDC1080.begin();
}

void loop()
{
// put your main code here, to run repeatedly:
  Serial.println("----------------------------------------");
```

```
  Serial.print("Manufacturer ID:0x");
  Serial.println(MelodyHDC1080.getManufacturerID(),HEX);
  Serial.print("Device ID:0x");
  Serial.println(MelodyHDC1080.getDeviceID(),HEX);
  Serial.print("Temperature:");
  Serial.println(MelodyHDC1080.getTemperature());
  Serial.print("Humidity:");
  Serial.println(MelodyHDC1080.getHumidity());
  Serial.println("-----------------------------------------\n\n");
  delay(1000);
}
```

图 3-33 HDC1080 例程源程序

例程 2，SimpleWebHDC1080.ino：通过网络读取温湿度值。cc3200 设置为 webserver 模式，用户可是有浏览器访问 cc3200，实验过程中 cc3200 的 IP 地址通过串口打印至 PC 端。Energia 开发工具提供了串口监视工具。程序例程图如所示，网络驱动库函数可见 MelodySimplehtml 文件夹下的 Melody_Simplehtml.h 和 Melody_Simplehtml.cpp 文件。例程流程图如图 3-34，例程代码如所示图 3-35。

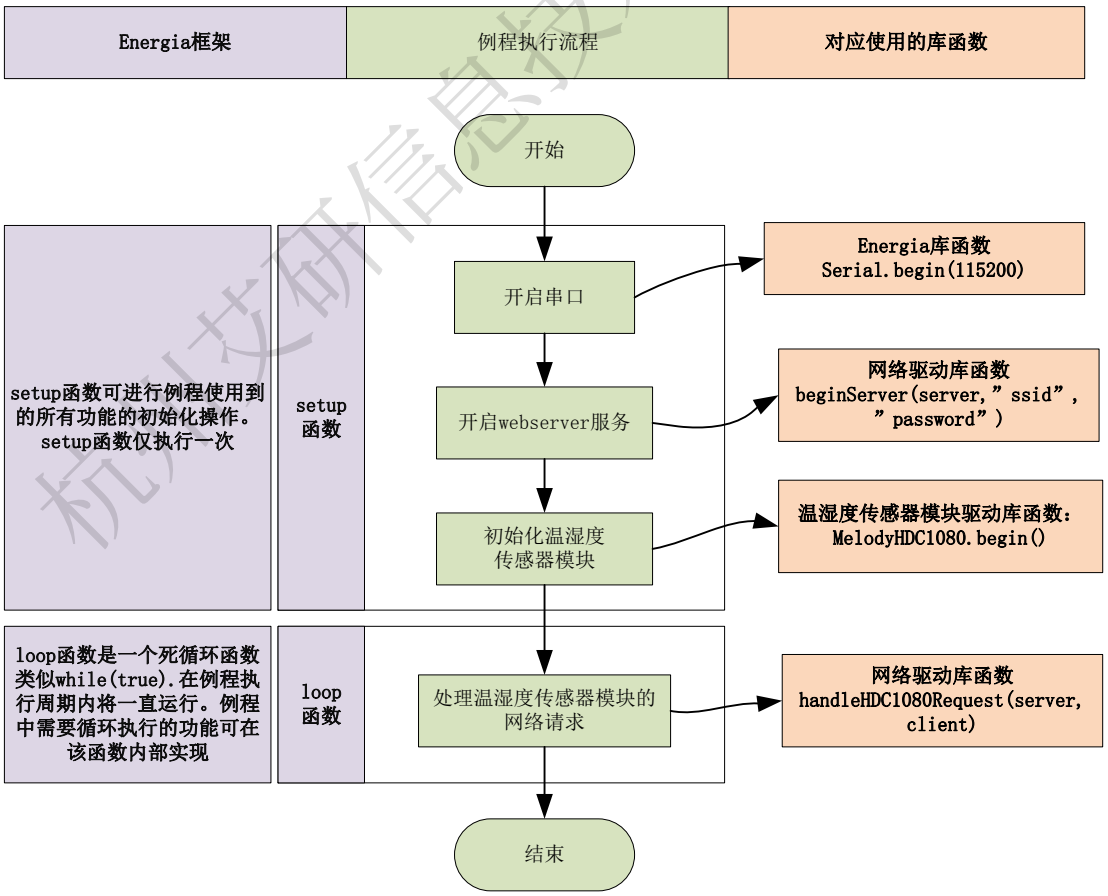


图 3-34 SimpleWebHDC1080 例程流程图

```
#include <stdbool.h>
#include <stdint.h>
```

```
#include <WiFi.h>
#include <Wire.h>
#include <Melody_HDC1080.h>
#include <Melody_Simplehtml.h>

WiFiServer server(80);
WiFiClient client;
void setup()
{
// put your setup code here, to run once:
  Serial.begin(115200);
  beginServer(server,"ssid","password");
  MelodyHDC1080.begin();

}

void loop()
{
// put your main code here, to run repeatedly:
  handleHDC1080Request(server,client);
}
```

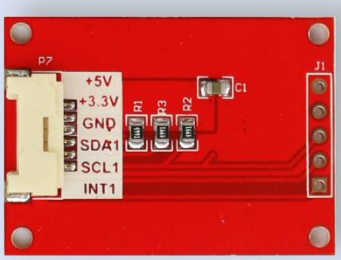
图 3-35 SimpleWebHDC1080 例程源程序

3.6 光感模块（MelodyOPT3001）

3.6.1 模块

光感模块使用 I2C 接口与 cc3200 进行通信，可读取模块的光照强度数据。模块如下：

表 3-29 光感模块

光感模块	概况
	1 模块使用芯片 OPT3001 2 使用 I2C 接口通信 3 测量范围：0.01lux 至 83k lux 4 23 位有效动态范围，具有自动增益范围设置功能

3.6.2 实现原理及原理图简介

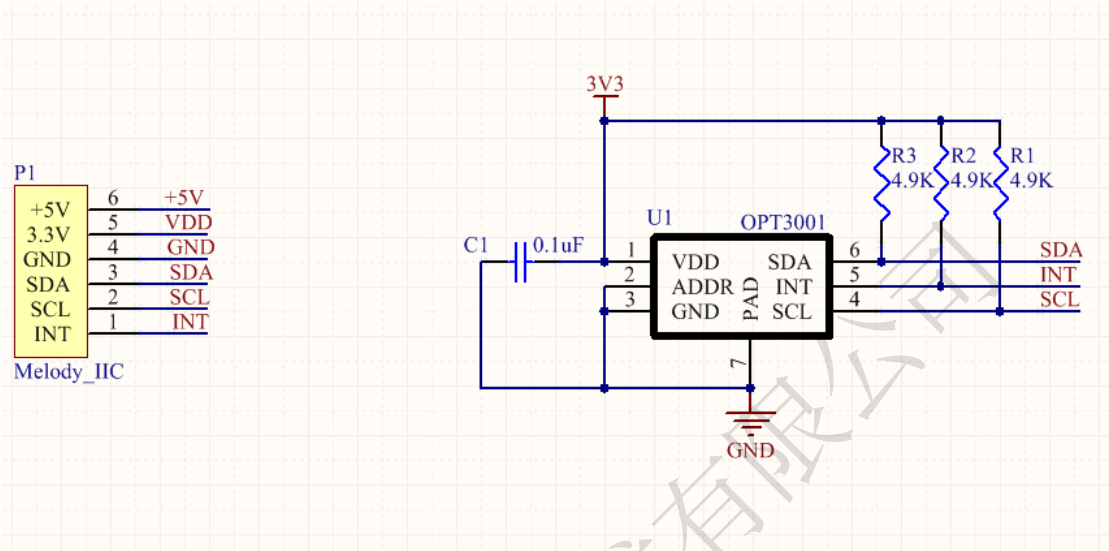


图 3-36 光感模块原理图

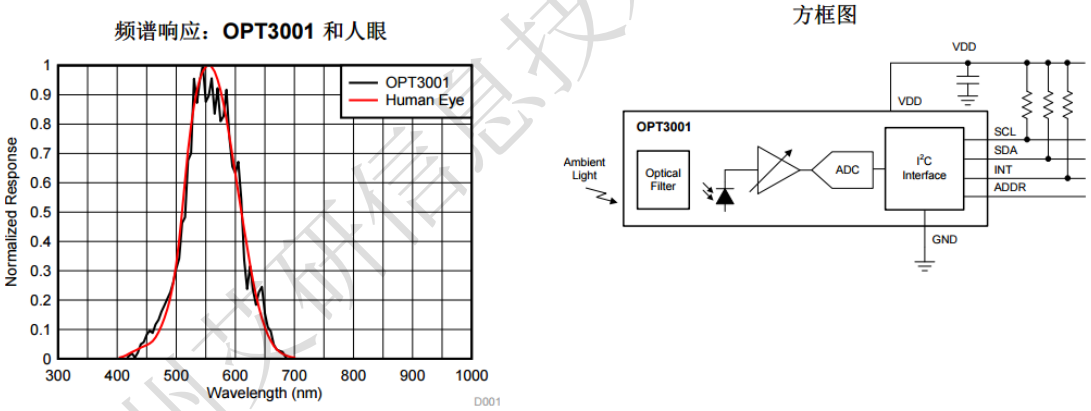


图 3-37 频谱相应和简化原理图

OPT3001 传感器用于测量可见光的密度。OPT3001 器件兼具精密的频谱响应和较强的 IR 阻隔功能，，因此能够如人眼般准确测量光强且不受光源影响。OPT3001 专门针对构建基于光线的人眼般体验的系统而设计，是人眼匹配度低且红外阻隔能力差的光电二极管、光敏电阻或其它环境光传感器的首选理想替代产品。

测量范围可达 0.01lux 至 83k lux，且内置有满量程设置功能，无需手动选择满量程范围。数字输出通过 I2C 接口于 MCU 连接即可

3.6.3 软件实现

光感模块使用 I2C 接口，Energia 已经实现了 I2C 库函数，用户直接使用即可。I2C 库函数，可在 http://energia.nu/reference/wire/查看，可以直接查看“3.5.3.1I2C 库函数”对 I2C 库函数的使用介绍。

3.6.3.1 光感模块驱动程序实现

写时序

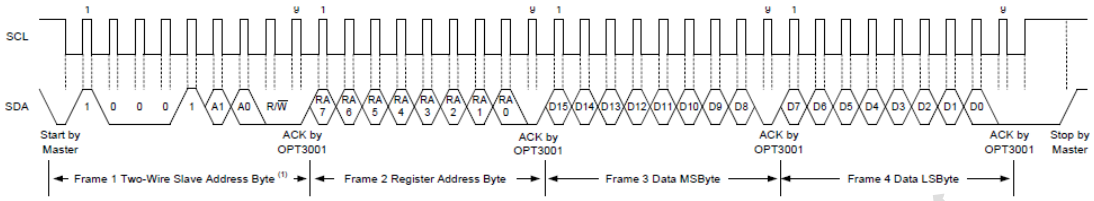


图 3-38 写时序

在实现时 cc3200 做为 Master，OPT3001 做为 slave。数据写入过程：第一步：cc3200 发送 OPT3001 的地址；第二步：发送需要写入数据的寄存器地址；第三步：发送两个字节长度的数据，先发送高字节数据，在发送低字节数据。

结合图 3-24 和图 3-38，写数据程序实现如下（不包括启动 I2C 功能，I2C 启动仅需调用 Wire.begin()即可，并且在使用过程中仅可被调用一次。）

```
/**************************************************************************
 * @brief:  写寄存器数据
 * @param:  regAddress,寄存器地址
 *          data,待写入数据
 * @return:  none

**************************************************************************/
void CMelodyOPT3001::write(uint8_t regAddress,uint16_t data)
{
    Wire.beginTransmission(OPTADDRESS);
    Wire.write(regAddress);
    Wire.write(data >>8);
    Wire.write(data);
    Wire.endTransmission();
}
```

图 3-39 写时序程序实现

读时序

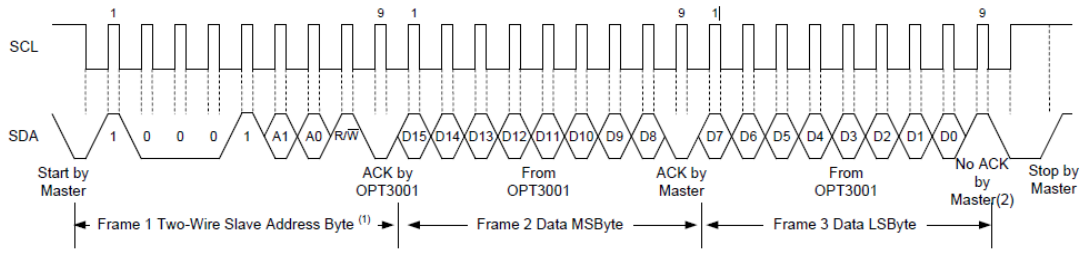


图 3-40 读时序

在实现时 cc3200 做为 Master，OPT3001 做为 slave。数据读取过程：第一步：cc3200 发送 OPT3001 的地址；第二步：读取两个字节长度的数据，高字节在前，低字节在后。

结合图 3-24 和图 3-40，写数据程序实现如下（不包括启动 I2C 功能，I2C 启动仅需调用 Wire.begin()即可，并且在使用过程中仅可被调用一次。）

```
/**************************************************************
 * @brief:  读取寄存器数据
 * @param:  regAddress,寄存器地址
 * @return: none
**************************************************************/
uint16_t CMelodyOPT3001::read(uint8_t regAddress)
{
    Wire.beginTransmission(OPTADDRESS);
    Wire.write(regAddress);
    Wire.endTransmission();

uint16_t data =0;
    Wire.requestFrom(OPTADDRESS,2);
if(Wire.available()>=2)
{
       data = Wire.read();
       data =(data <<8)| Wire.read();
}
return data;
}
```

图 3-41 读时序程序实现

驱动库部分函数如下：类对象"MelodyOPT3001"定义在 Melody_OPT3001.cpp 文件中。

表 3-30 begin()函数

驱动库函数	void begin(void)
说明	开启 I2C 功能，初始化配置 OPT3001
使用范例	MelodyOPT3001.begin();

表 3-31 getManufacturerID()函数

驱动库函数	uint16_t getManufacturerID (void)
说明	获取 OPT3001 的 Manufacturer ID
使用范例	uint16_t id = MelodyOPT3001. getManufacturerID ();

表 3-32 getDeviceID()函数

驱动库函数	uint16_t getDeviceID (void)
说明	获取 OPT3001 的 Device ID
使用范例	uint16_t id = MelodyOPT3001. getDeviceID ();

表 3-33 getLux()函数

驱动库函数	float getLux(void)
说明	获取当前光照强度

使用范例	uint16_t id = MelodyOPT3001. getDeviceID ();

3.6.3.2 实验例程

实验例程操作及结果展示可浏览视频资料：http://www.hpati.com/product_videos/v80.html

例程 1，OPT3001.ino:读取当前光照强度，使用串口打印。实例流程图如图 3-42，实例代码如图 3-43。

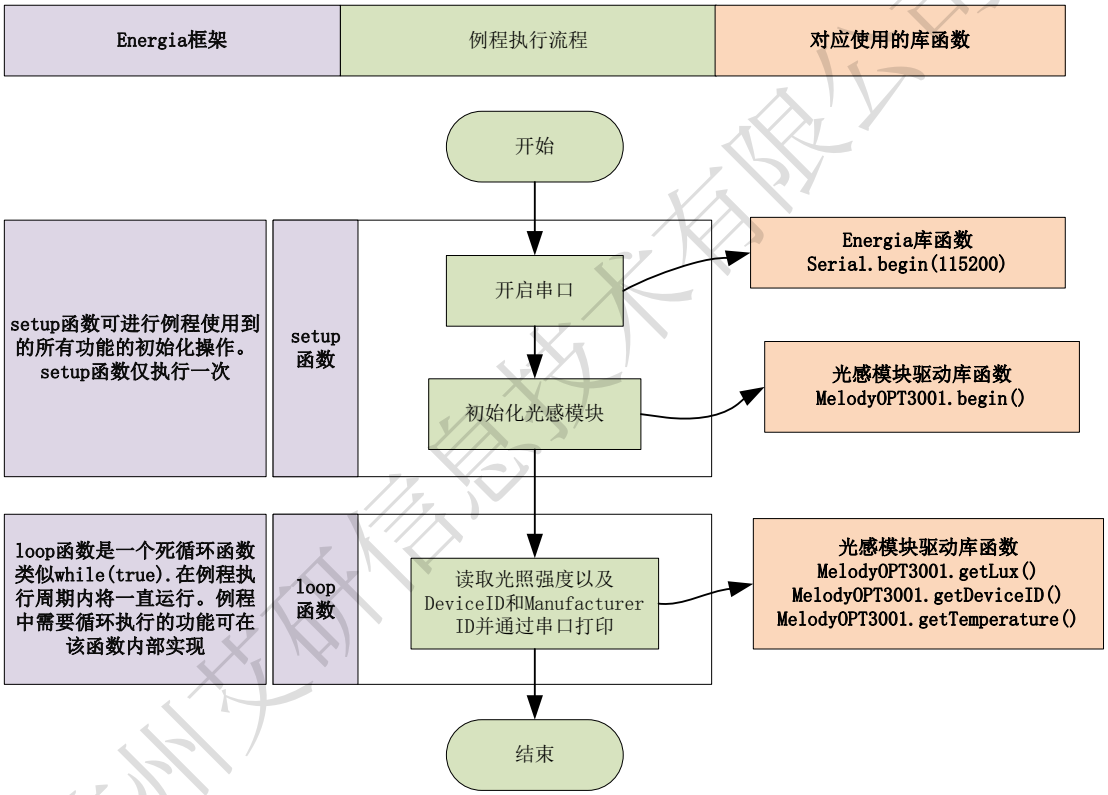


图 3-42 OPT3001 实例流程图

```
#include <stdbool.h>
#include <stdint.h>
#include "Wire.h"
#include "Melody_OPT3001.h"
void setup()
{
// put your setup code here, to run once:
  Serial.begin(115200);
  MelodyOPT3001.begin();
}

void loop()
```

```
{
// put your main code here, to run repeatedly:
  Serial.println("--------------------------------");
  Serial.print("Manufacturer ID:0x");
  Serial.println(MelodyOPT3001.getManufacturerID(),HEX);
  Serial.print("Device ID:0x");
  Serial.println(MelodyOPT3001.getDeviceID(),HEX);
  Serial.print("Lux:");
  Serial.println(MelodyOPT3001.getLux());
  Serial.println("--------------------------------");
  delay(1000);
}
```

图 3-43 OPT300 例程源代码

例程 2，SimpleWebOPT3001.ino：通过网络读取光照强度。cc3200 设置为 webserver 模式，用户可是有浏览器访问 cc3200，实验过程中 cc3200 的 IP 地址通过串口打印至 PC 端。Energia 开发工具提供了串口监视工具。程序例程图如所示，网络驱动库函数可见 MelodySimplehtml 文件夹下的 Melody_Simplehtml.h 和 Melody_Simplehtml.cpp 文件。例程流程图如图 3-44，例程代码如所示图 3-45。

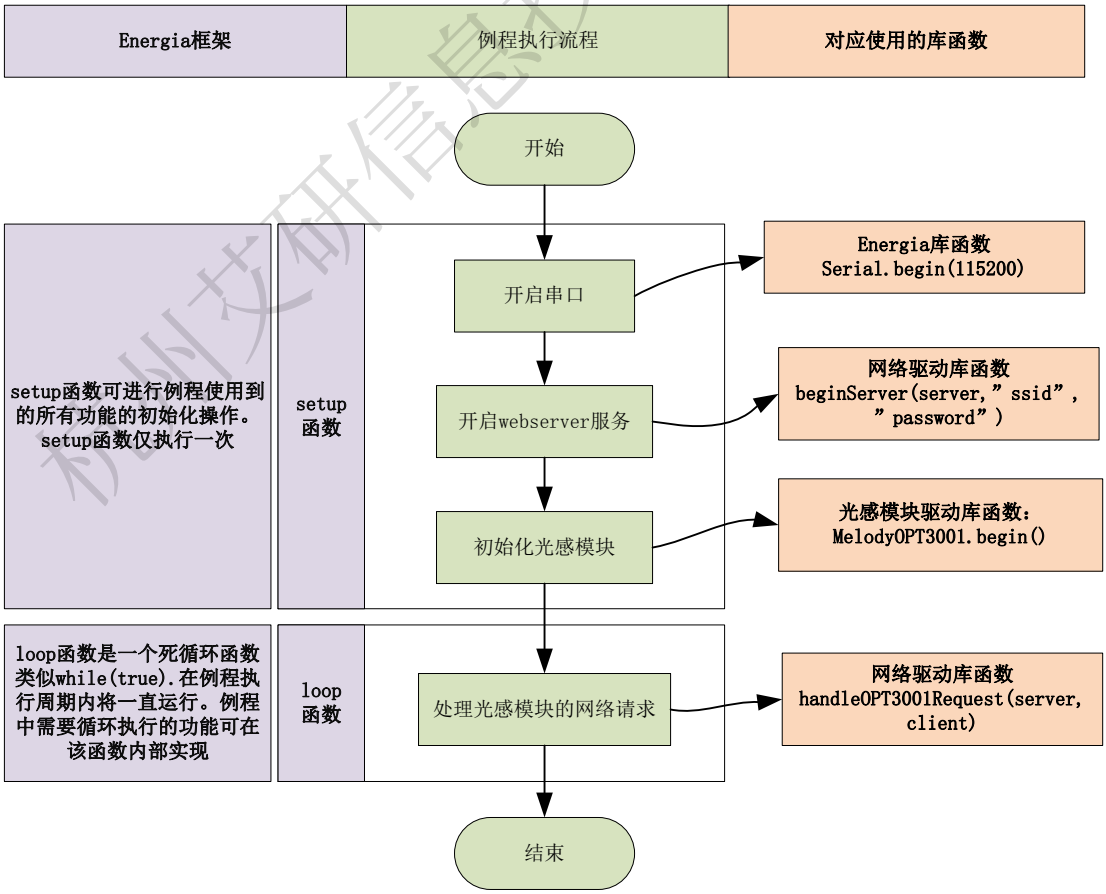


图 3-44 SimpleWebOPT3001 例程流程图

```
#include <stdbool.h>
```

```
#include <stdint.h>
#include <WiFi.h>
#include "Wire.h"
#include "Melody_OPT3001.h"
#include "Melody_Simplehtml.h"

WiFiServer server(80);
WiFiClient client;
void setup()
{
// put your setup code here, to run once:
  Serial.begin(115200);
  beginServer(server,"ssid","password");
  MelodyOPT3001.begin();
}

void loop()
{
// put your main code here, to run repeatedly:
  handleOPT3001Request(server,client);
}
```

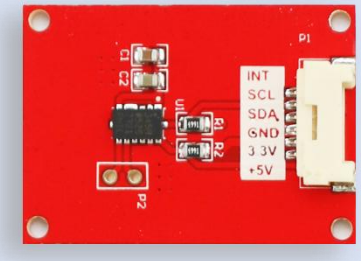
图 3-45 SimpleWebOPT3001 例程源程序

3.7 三轴加速度模块（MelodyADXL345）

3.7.1 模块

三轴加速度使用 I2C 通信，模块如下：

表 3-34 三轴加速度模块

三轴加速度模块	概况
	1 模块使用 ADXL345 2 使用 I2C 通信接口 3 可进行单振/双振检测，活动/非活动检测，自由落体检测

3.7.2 实现原理及原理图简介

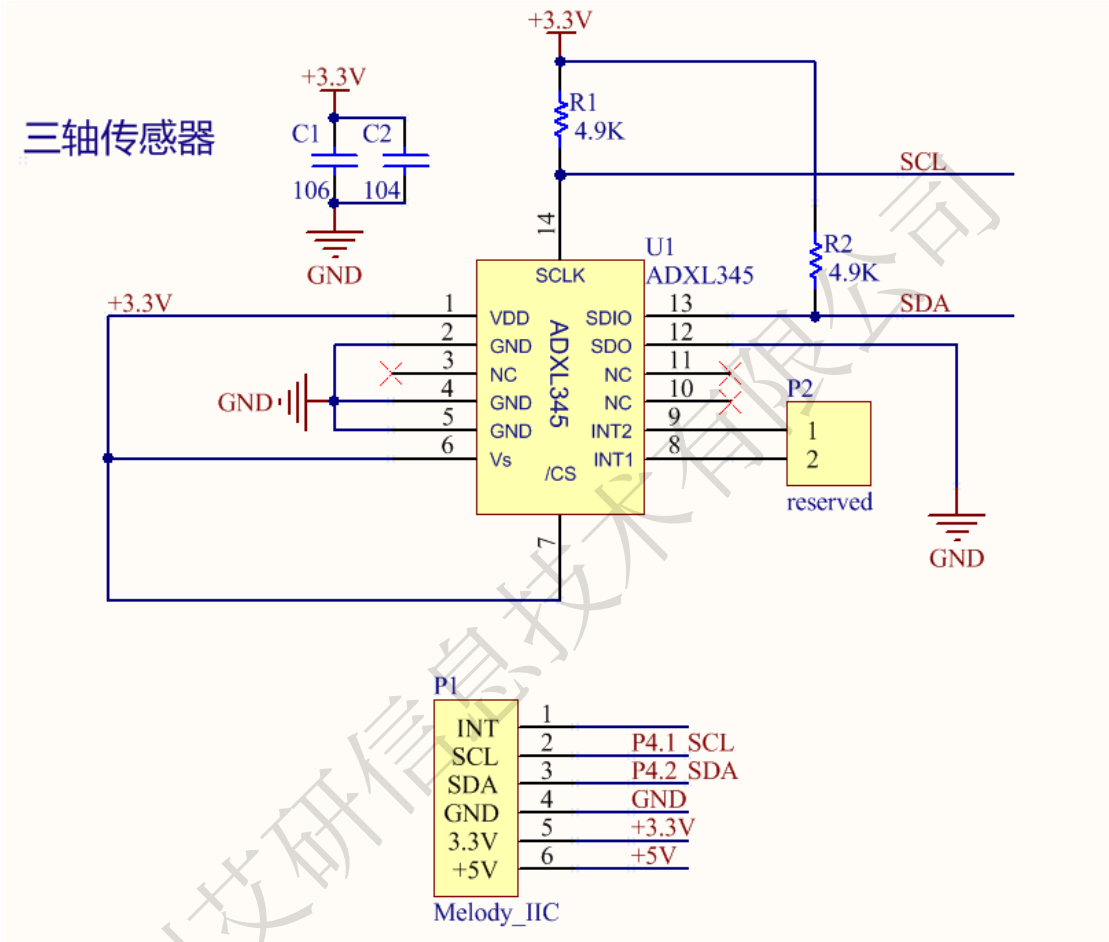


图 3-46 三轴加速度原理图

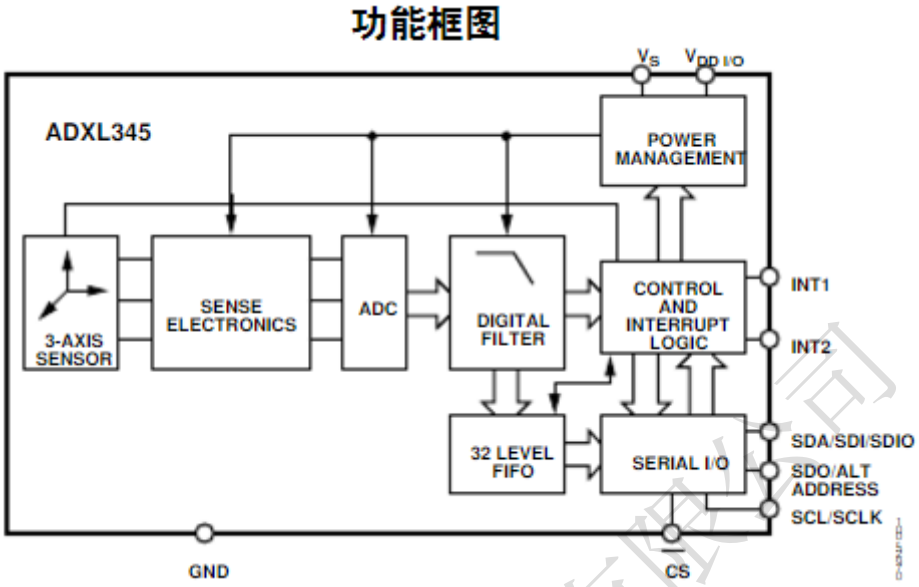


图 3-47 ADXL345 内部结构

ADXL 是一款小而薄的超低功耗 3 轴加速度传感器，分辨率高（13 位），测量范围±16g，数字输出数据为 16 位二进制补码格式，可通过 I2C 接口进行访问。

3.7.3 软件实现

三轴加速度模块使用 I2C 接口，Energia 已经实现了 I2C 库函数，用户直接使用即可。I2C 库函数，可在 http://energia.nu/reference/wire/查看，可以直接查看“3.5.3.1I2C 库函数”对 I2C 库函数的使用介绍

3.7.3.1 三轴加速度模块驱动库实现

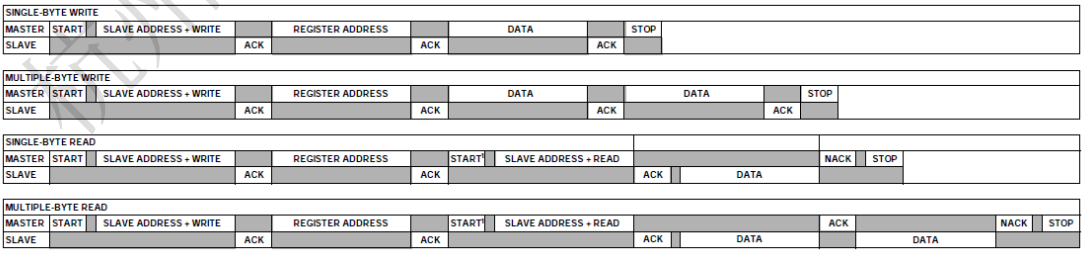


图 3-48 I2C 器件寻址

图 3-48 给出了芯片——ADXL345 的 I2C 数据通信格式，其中前两个是“写时序”格式，后两个为“读时序”模式。

写时序：分成“单字节写入”和“多字节写入”。

“单字节写入”操作顺序：cc3200 作为 Master，ADXL345 作为 Slave。第一步：写入器件地址；第二步：写入待写入数据的寄存器地址；第三步：写入待写入数据。

“多字节写入”操作顺序：cc3200 作为 Master，ADXL345 作为 Slave。第一步：写入器件地址；第二步：写入待写入数据的寄存器地址；第三步：依次写入待写入数据。

程序实现如下：

```
/*****************************************************************
 * @brief:  写单字节数据
 * @param:   regAddress,寄存器地址
 *          data，待写入数据
 * @return:  none

*****************************************************************/
void CMelodyADXL345::write(uint8_t regAddress,uint8_t data)
{
    write(regAddress,&data,1);
}

/*****************************************************************
 * @brief:  写入多个字节数据
 * @param:  regAddress,寄存器地址
 *         pdata，指向待写入数据的指针
 *         length，待写入数据长度
 * @return: none
*****************************************************************/
void CMelodyADXL345::write(uint8_t regAddress,uint8_t*pdata,uint8_t
                                        length)
{
    Wire.beginTransmission(ADXLADDRESS);
    Wire.write(regAddress);
    Wire.write(pdata,length);
    Wire.endTransmission();
}
```

图 3-49 写时序程序实现

读时序：分成“单字节读取”和“多字节读取”。

“单字节读取”操作顺序：cc3200 作为 Master，ADXL345 作为 Slave。第一步：写入器件地址；第二步：写入待读取数据的寄存器地址；第三步：再次写入器件地址；第四步：读取待读取寄存器的数据。

“多字节读取”操作顺序：cc3200 作为 Master，ADXL345 作为 Slave。第一步：写入器件地址；第二步：写入待读取数据的寄存器地址；第三步：再次写入器件地址；第四步：依次读取数据。

程序实现如下：

```
/*****************************************************************
 * @brief:  读取单字节数据
 * @param:   regAddress,寄存器地址
 * @return: 读取的数据
*****************************************************************/
uint8_t CMelodyADXL345::read(uint8_t regAddress)
```

```
{
uint8_t data =0;
    Wire.beginTransmission(ADXLADDRESS);
    Wire.write(regAddress);
    Wire.endTransmission();

    Wire.requestFrom(ADXLADDRESS,1);
if(Wire.available()>=1)
{
       data = Wire.read();
}
return data;
}
/*****************************************************************
 * @brief:  读取多字节数据
 * @param:  regAddress,寄存器地址
 *         pdata，指向存放读取数据缓存指针
 *         length,待读取数据长度
 * @return: 实际读取数据长度
*****************************************************************/
uint8_t CMelodyADXL345::read(uint8_t
                                    regAddress,uint8_t*pdata,uin
                                    t8_t length)
{
uint8_t readlength =0;
    Wire.beginTransmission(ADXLADDRESS);
    Wire.write(regAddress);
    Wire.endTransmission();

    Wire.requestFrom(ADXLADDRESS,length);
while(Wire.available())
{
       pdata[readlength++]= Wire.read();
}

return readlength;
}
```

图 3-50 读时序程序实现

以上为 I2C 通信的“写时序”和“读时序”实现，其余驱动库函数均需要调用此两处函数。其余库函数如下：类对象“MelodyADXL345”定义在 Melody_ADXL345.cpp 中。

表 3-35 begin()函数

驱动库函数	void begin(void)
说明	初始化设置 ADXL345

使用范例	MelodyADXL345.begin();

表 3-36 getXAccelerate()函数

驱动库函数	**int16_t getXAccelerate (void)**
说明	获取 X 轴数据
使用范例	int16_t x = MelodyADXL345.getXAccelerate();

表 3-37 getYAccelerate()函数

驱动库函数	**int16_t getYAccelerate (void)**
说明	获取 Y 轴数据
使用范例	int16_t y = MelodyADXL345.getYAccelerate();

表 3-38 getZAccelerate()函数

驱动库函数	**int16_t getZAccelerate (void)**
说明	获取 Z 轴数据
使用范例	int16_t z = MelodyADXL345.getZAccelerate();

表 3-39 getDeviceID()函数

驱动库函数	**uint8_t getDeviceID (void)**
说明	获取 ID 值
使用范例	uint8_t id = MelodyADXL345.getDeviceID();

3.7.3.2 实验例程

实验例程操作及结果展示可浏览视频资料：http://www.hpati.com/product_videos/v74.html

例程 1，ADXL345.ino:读取每个轴的数据，使用串口打印。实例流程图如图 3-51，实例代码如图 3-52。

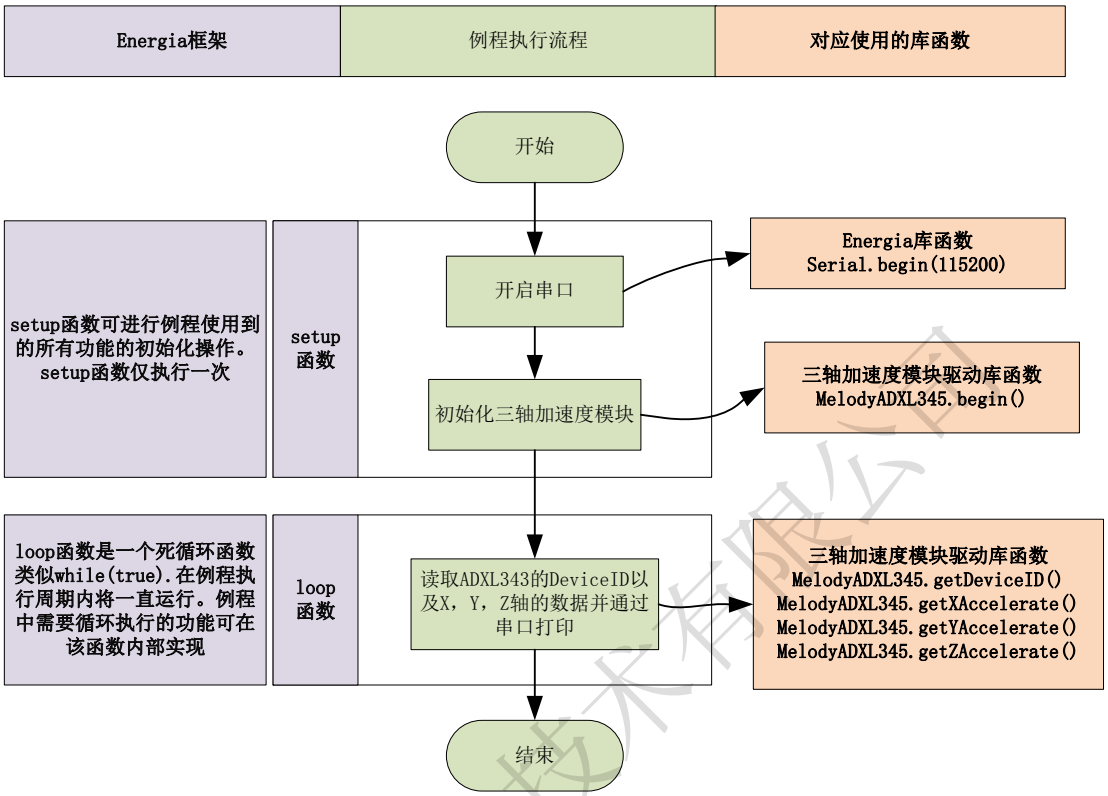


图 3-51 ADXL345 例程流程图

```
#include <stdbool.h>
#include <stdint.h>
#include "Wire.h"
#include "Melody_ADXL345.h"
void setup()
{
// put your setup code here, to run once:
  Serial.begin(115200);
  MelodyADXL345.begin();
}

void loop()
{
// put your main code here, to run repeatedly:
  Serial.println("--------------------------------");
  Serial.print("ID:0x");
  Serial.println(MelodyADXL345.getDeviceID(),HEX);
  Serial.print("X Accelerate:");
  Serial.println(MelodyADXL345.getXAccelerate());
  Serial.print("Y Accelerate:");
  Serial.println(MelodyADXL345.getYAccelerate());
  Serial.print("Z Accelerate:");
```

```
  Serial.println(MelodyADXL345.getZAccelerate());
  Serial.println("--------------------------------");
  delay(1000);
}
```

图 3-52 ADXL345 例程源程序

例程 2，SimpleWebADXL345.ino:通过网络读取光照强度。cc3200 设置为 webserver 模式，用户可是有浏览器访问 cc3200，实验过程中 cc3200 的 IP 地址通过串口打印至 PC 端。Energia 开发工具提供了串口监视工具。程序例程图如所示，网络驱动库函数可见 MelodySimplehtml 文件夹下的 Melody_Simplehtml.h 和 Melody_Simplehtml.cpp 文件。例程流程图如图 3-53，例程代码如所示图 3-54。

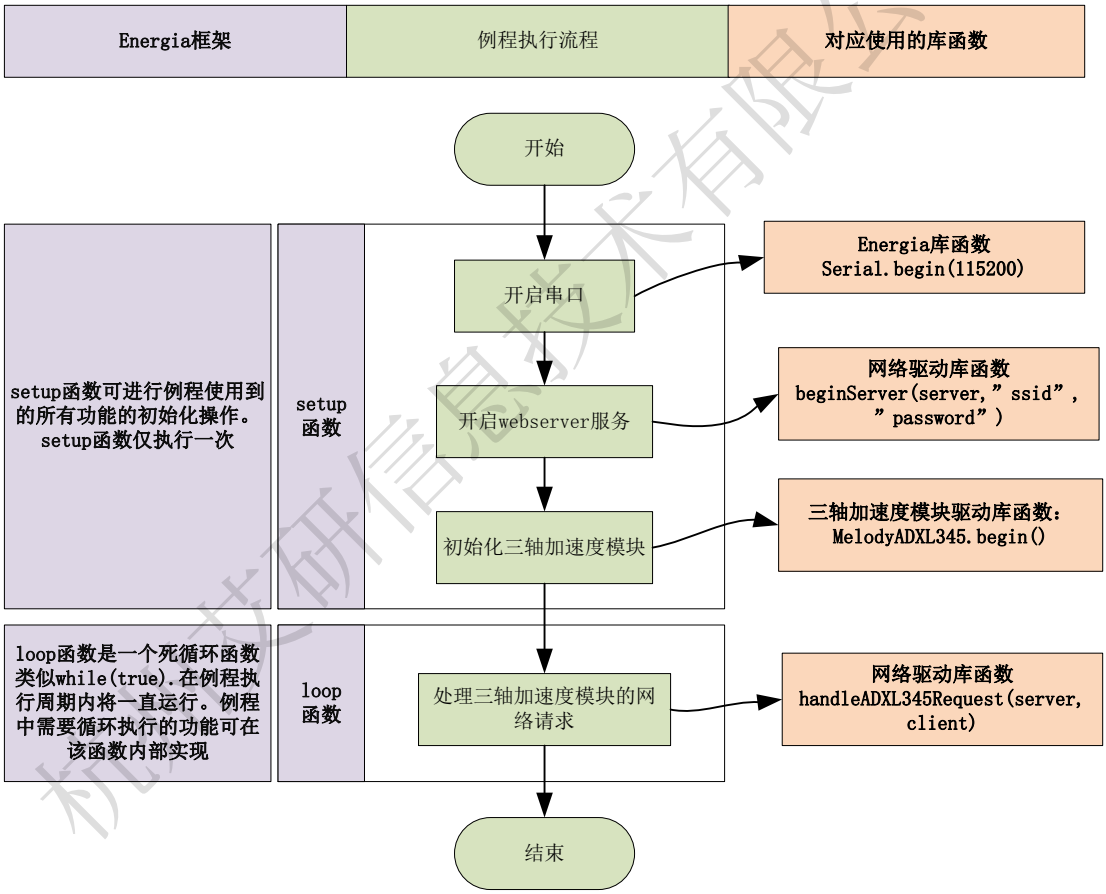


图 3-53 SimpleWebADXL345 例程流程图

```
#include <stdbool.h>
#include <stdint.h>
#include <WiFi.h>
#include <Wire.h>
#include "Melody_ADXL345.h"
#include "Melody_Simplehtml.h"

WiFiServer server(80);
WiFiClient client;
```

```
void setup()
{
// put your setup code here, to run once:
  Serial.begin(115200);
  beginServer(server,"ssid","password");
  MelodyADXL345.begin();
}

void loop()
{
// put your main code here, to run repeatedly:
  handleADXL345Request(server,client);
}
```

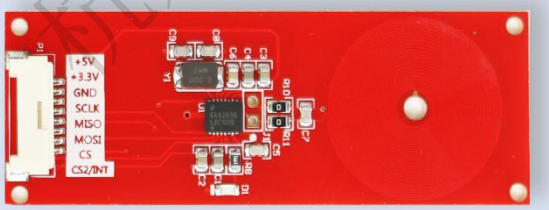
图 3-54 SimpleWebADXL345 例程源程序

3.8 LDC1000 模块（MelodyLED1000）

3.8.1 模块

LDC1000 使用 SPI 通信，模块如下

表 3-40 LDC1000 模块

LDC1000 模块	概况
	1、SPI 接口通信 2、可对线性位置或者角位置、位移、运动、压缩、振动以及金属成分进行测量

3.8.2 实现原理及原理图简介

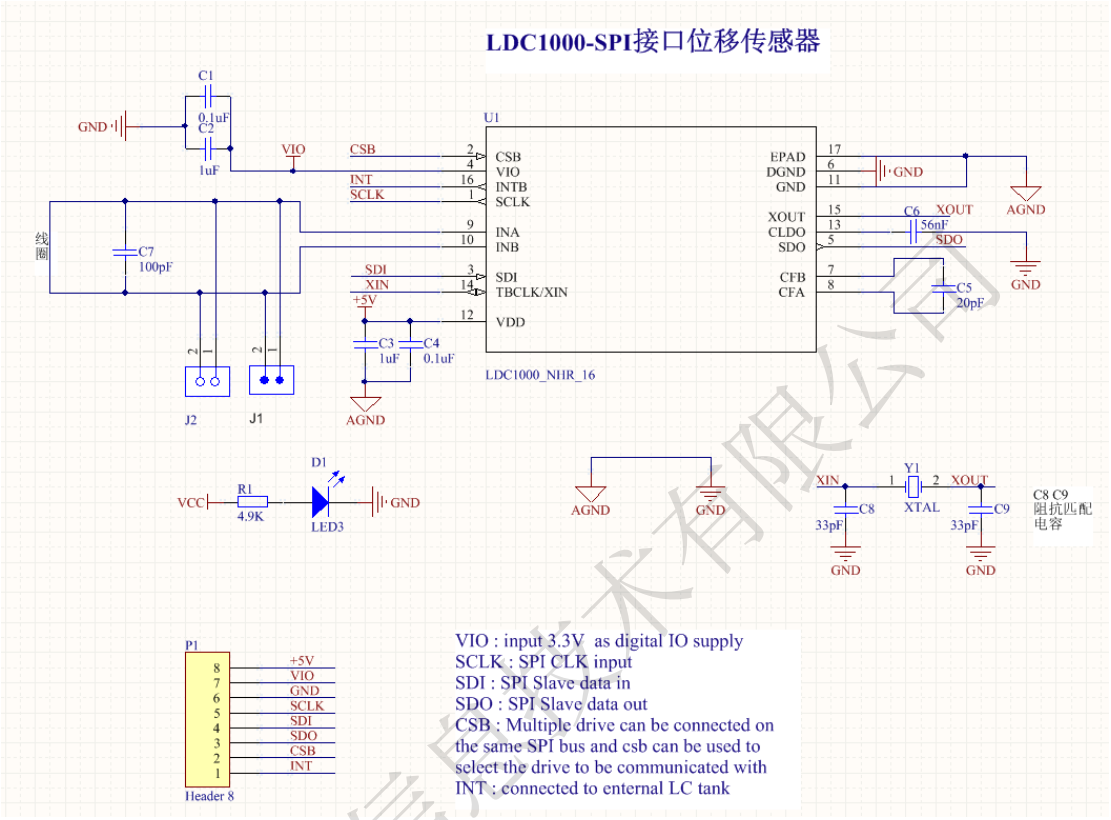


图 3-55 LDC1000 模块原理图

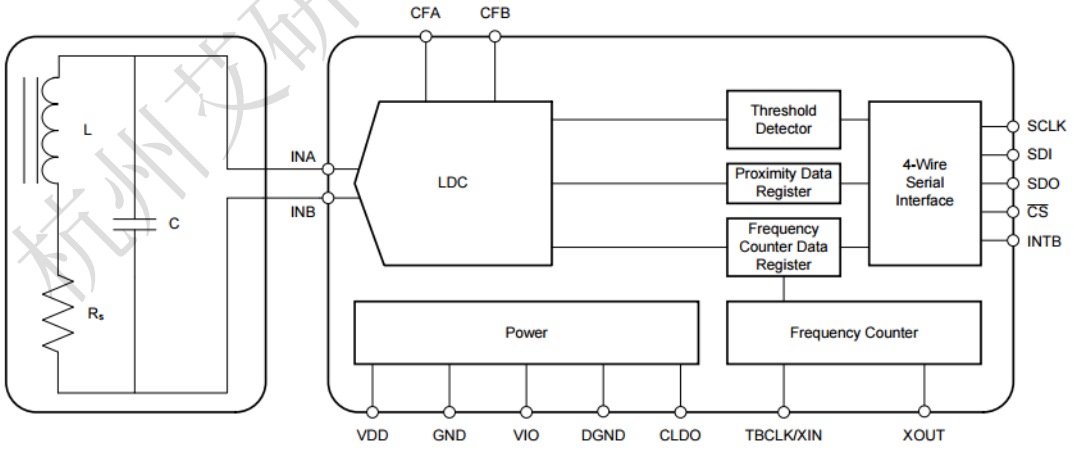


图 3-56 LDC1000 功能结构图

LDC1000 是世界首款电感到数字转换器。提供低功耗，小封装，低成本解决方案。SPI 接口很方便连接 MCU。LDC1000 只需要外接一个 PCB 线圈或者自制线圈就可以实现非接触式电感检测。LDC1000 的电感检测并不是指像 Q 表那样测试线圈的电感量，而是可以测试外部金属物体和 LDC1000 的测试线圈的空间位置关系。

利用 LDC1000 这个特性配以外部设计的金属物体即可很方便实现：水平或垂直距离检测；角度检测；位移监测；运动检测；振动检测；金属成分检测（合金检测）。可以广泛的应用在汽车、消费电子、计算机、工业、通信和医疗领域。

LDC1000 测量电感频率是用测试 LC 谐振频率的方法。LDC1000 有外部的基准时钟，也是使用计数方法来做频率计。看一下手册中给出的公式就能清晰的看出计数原理。

$$f_{sensor} = (1/3) \times (F_{ext} / F_{count}) \times (responseTime)$$

fsenser 是 LC 谐振频率，Fext 是外部基准时钟频率，Fcount 是 LDC1000 内部计数器值（0x23,0x24,0x25），Response time 是寄存器设定的一个值（0x04）。

将上面公式左右分别求倒数

$$1/f_{sensor} = 3 \times (F_{ext} / F_{count}) \times (responseTime)$$

将 response time 移动到等式左面

$$responseTime \times (1/f_{sensor}) = 3 \times F_{count} \times (1/F_{ext})$$

这样看就很清晰了，1/fsensor 是 LC 谐振周期，1/Fext 是基准时钟周期，也就是说在 response time 个 LC 谐振周期内，使用 LDC1000 的 Fcount 计数器记录基准时钟的个数用来推算 LC 的谐振频率。

根据 fsensor= $1/\sqrt{2 * \pi LC}$ 计算出 L。（C 是电路设计中的已知量）

$$L = 1/[C \times (2 \times \pi \times f_{sensor})^2]$$

3.8.3 软件实现

LDC1000 模块使用 SPI 接口，同 I2C 功能一样，Energia 已经实现了 SPI 库函数，用户直接使用即可。SPI 库函数，可在 http://energia.nu/reference/spi/查看。

3.8.3.1 SPI 库函数

根据 Energia 官网上的介绍，SPI 库函数使用 C++实现，由多个成员函数构成，用户可直接使用库文件已经定义的“SPI”对象调用其成员函数，如 SPI.begin()。各成员函数如图 3-23 所示。

Functions

- begin()
- end()
- setBitOrder()
- setClockDivider()
- setDataMode()
- transfer()
- setModule()

SPI Library

This library allows you to communicate with SPI devices, with the Energia target board as the master device.

A Brief Introduction to the Serial Peripheral Interface (SPI)

Serial Peripheral Interface (SPI) is a synchronous serial data protocol used by microcontrollers for communicating with one or more peripheral devices quickly over short distances. It can also be used for communication between two microcontrollers.

图 3-57 SPI 库函数

SPI 通信分成写数据和读数据，下面先简单介绍 SPI 库函数的使用，然后根据 LDC1000 的时序图介绍 LDC1000 驱动库的实现。

SPI 基本写过程，流程图如下：

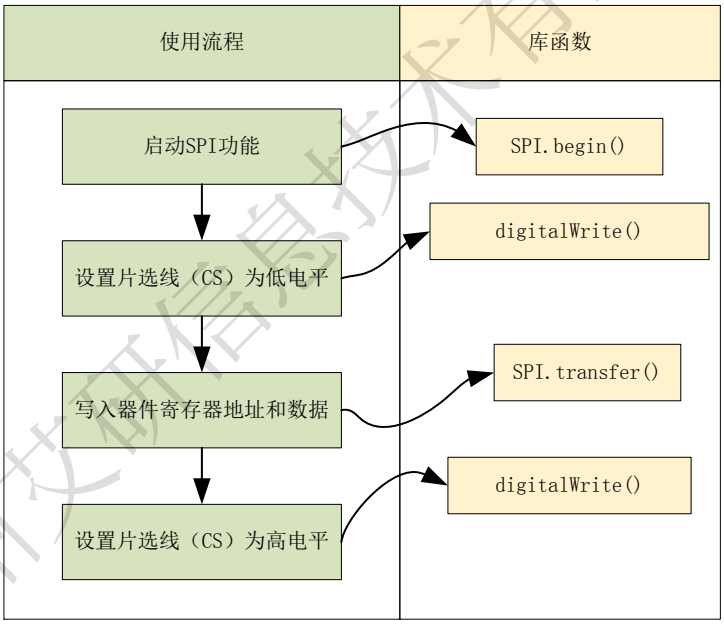


图 3-58 SPI 基本写过程流程图

SPI 基本读过程，流程图如下：

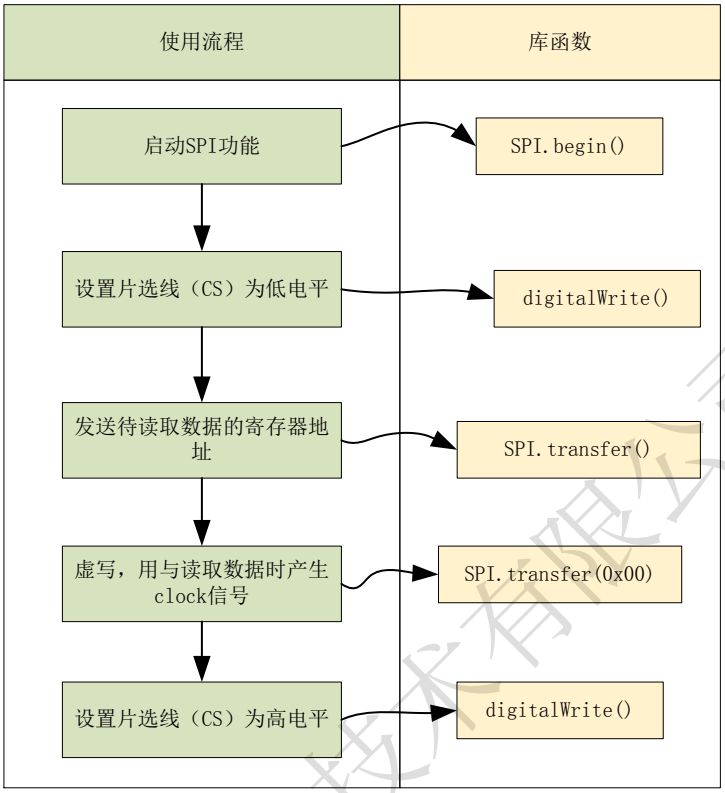


图 3-59 SPI 基本读过程流程图

如需查看 Energia 例程代码，可在 Energia 开发工具的菜单栏->File->Examples->SPI 下查看。

3.8.3.2 LDC1000 模块驱动库实现

LDC1000 时序如下：

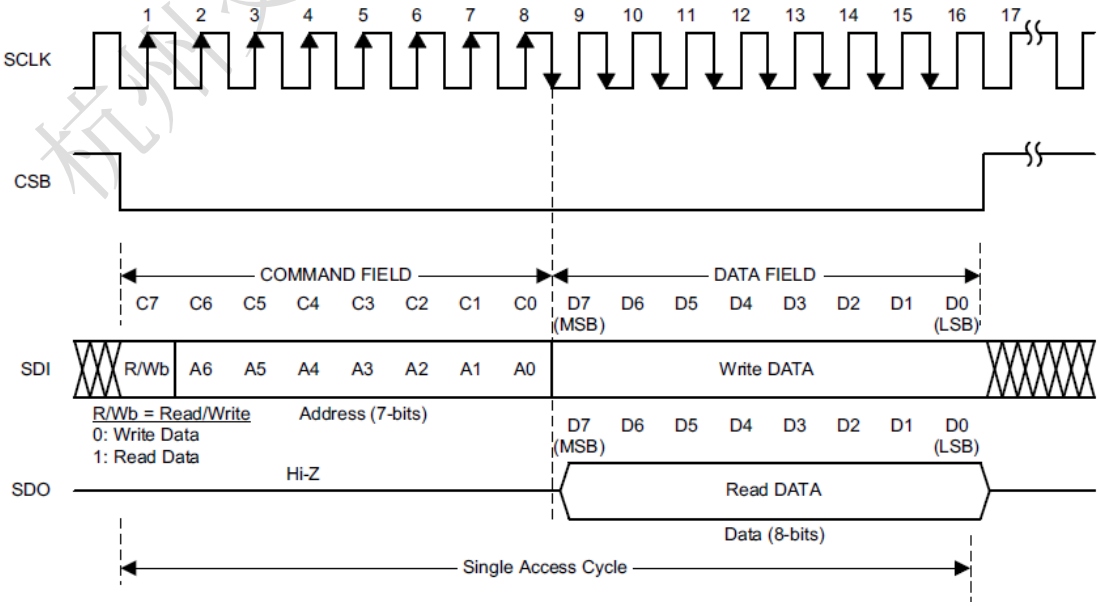


图 3-60 LDC1000 时序

结合图 3-58 和图 3-60 实现写操作程序如下：（不包括启动 SPI 功能，SPI 启动仅需调用 SPI.begin()即可，并且在使用过程中仅可被调用一次。）

```
/*****************************************************************
 * @brief:  写数据
 * @param:  regAddr,寄存器地址
 *          data,数据
 * @return: none
*****************************************************************/
void CMelodyLDC1000::write(uint8_t regAddr,uint8_t data)
{
    digitalWrite(LDCSPI_CS,LOW);

    SPI.transfer(regAddr | LDCSPI_WBIT);
    SPI.transfer(data);

    digitalWrite(LDCSPI_CS,HIGH);
}
```

图 3-61 LDC1000 写操作实现

结合图 3-59 和图 3-60 实现写操作程序如下：（不包括启动 SPI 功能，SPI 启动仅需调用 SPI.begin()即可，并且在使用过程中仅可被调用一次。）

```
/*****************************************************************
*************************
 * @brief:  读取数据
 * @param:  regAddr,寄存器地址
 * @return: none

*****************************************************************
*************************/
uint8_t CMelodyLDC1000::read(uint8_t regAddr)
{
uint8_t result =0;

    digitalWrite(LDCSPI_CS,LOW);

    SPI.transfer(regAddr | LDCSPI_RBIT);
    result = SPI.transfer(0x00);

    digitalWrite(LDCSPI_CS,HIGH);

return result;
}
```

图 3-62 LDC1000 读操作实现

以上为 SPI 通信的“写操作”和“读操作”实现，其余驱动库函数均需要调用此两处函数。部分库函数如下：类对象“MelodyLDC1000”定义在 Melody_LDC1000.cpp 中。

表 3-41 begin()函数

驱动库函数	void begin()
说明	开启 SPI 功能，初始化设置 LDC1000
使用范例	MelodyLDC1000.begin();

表 3-42 available()函数

驱动库函数	bool available()
说明	判断是否可有数据可读
使用范例	MelodyLDC1000.available();

表 3-43 getProximity()函数

驱动库函数	uint16_t getProximity()
说明	获取 Proximity 数据
使用范例	uint16_t proximity = MelodyLDC1000.getProximity();

驱动库函数	uint32_t getFrequency()
说明	获取 Frequency 数据
使用范例	uint16_t proximity = MelodyLDC1000.getFrequency();

表 3-44calcInductance()函数

驱动库函数	float calcInductance(uint32_t frequency)
说明	计算互感值、 frequency：频率值，该值可由函数 getFrequency()获取
使用范例	uint16_t proximity = MelodyLDC1000.getProximity(MeloduLDC1000.getFrequency());

3.8.3.3 实验例程

实验例程操作及结果展示可浏览视频资料：http://www.hpati.com/product_videos/v77.html

例程 1，LDC1000.ino:获取 LDC1000 的 Proximity 和 Frequency，并且计算互感值。例程流程图如图 3-63，例程程序实现如图 3-64。

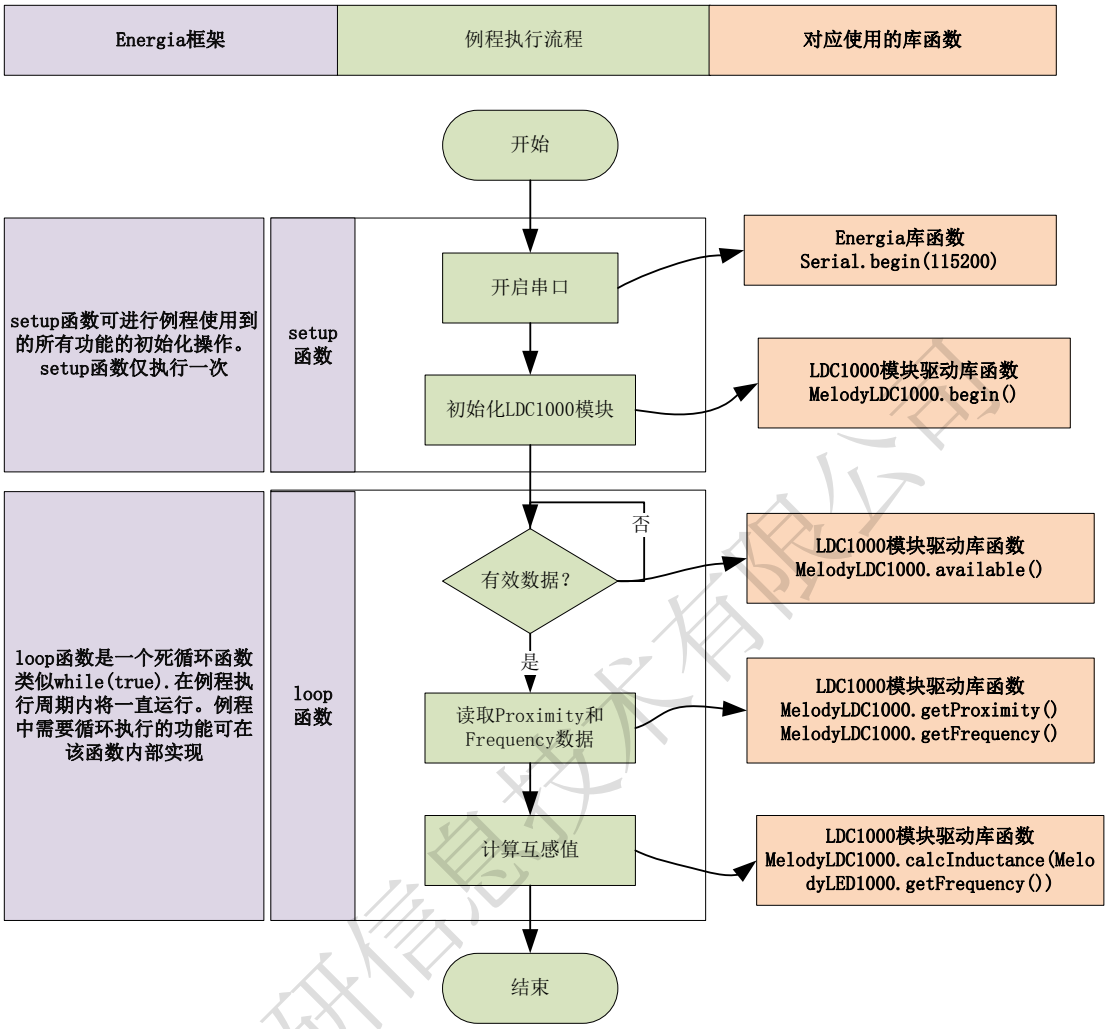


图 3-63 LDC1000 例程流程图

```
#include <stdbool.h>
#include <stdint.h>
#include <SPI.h>
#include "Melody_LDC1000.h"
uint16_t proximity =0;
uint32_t frequency =0;
void setup()
{
// put your setup code here, to run once:
  Serial.begin(115200);
  MelodyLDC1000.begin();
}
void loop()
{
// put your main code here, to run repeatedly:
if(MelodyLDC1000.available())
{
```

```
    proximity = MelodyLDC1000.getProximity();
    frequency = MelodyLDC1000.getFrequency();
    Serial.print("Proximity:");
    Serial.print(proximity);
    Serial.print("\t\tFrequency:");
    Serial.print(frequency);
    Serial.print("\t\tInductance:");
    Serial.println(MelodyLDC1000.calcInductance(frequency));
}
}
```

图 3-64 LDC1000 例程源程序

例程 2，SimpleWebLDC1000.ino:通过网络读取光照强度。cc3200 设置为 webserver 模式，用户可是有浏览器访问 cc3200，实验过程中 cc3200 的 IP 地址通过串口打印至 PC 端。Energia 开发工具提供了串口监视工具。程序例程图如所示，网络驱动库函数可见 MelodySimplehtml 文件夹下的 Melody_Simplehtml.h 和 Melody_Simplehtml.cpp 文件。例程流程图如图 3-65，例程代码如所示图 3-66。

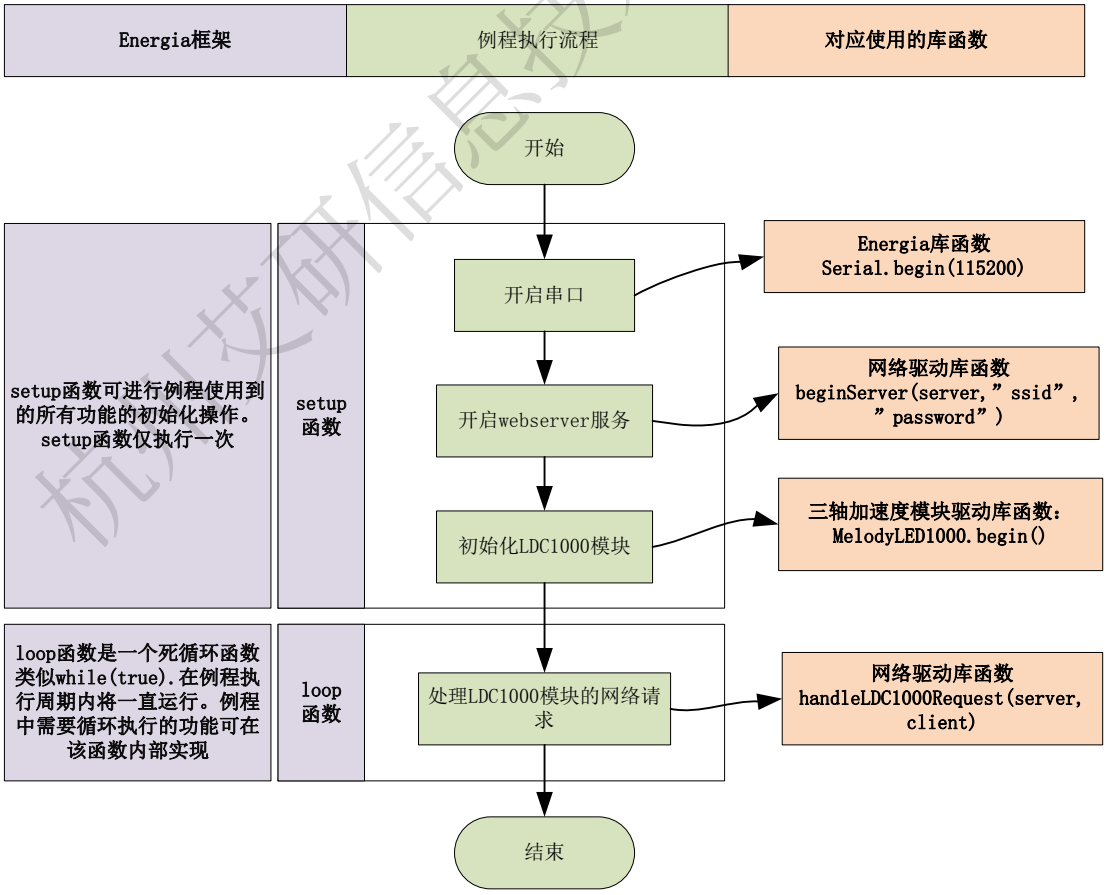


图 3-65 SimpleWebLDC1000 例程流程图

```
#include <stdbool.h>
#include <stdint.h>
```

```
#include <WiFi.h>
#include <SPI.h>
#include "Melody_LDC1000.h"
#include "Melody_Simplehtml.h"

WiFiServer server(80);
WiFiClient client;

void setup()
{
// put your setup code here, to run once:
  Serial.begin(115200);
  beginServer(server,"ssid","password");
  MelodyLDC1000.begin();
}

void loop()
{
// put your main code here, to run repeatedly:
  handleLDC1000Request(server,client);
}
```

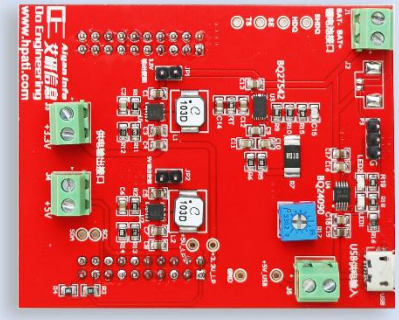
图 3-66 SimpleWebLDC1000 例程源程序

3.9 电池管理模块（MelodyLiBattery）

3.9.1 模块

电池管理模块如下：模块的连接请查看《AY-IOT 连接使用须知》

表 3-45 电池管理模块

模块	概况
	1、模块使用芯片 BQ24090 完成锂电池充电，BQ27542 完成锂电池参数监控； 2、BQ27542 使用 I2C 接口通信； 3、模块实现锂电池充放电功能； 4、可选择充电电流大小； 5、对外输出电压可选择 3V 和 5V

3.9.2 实现原理及原理图简介

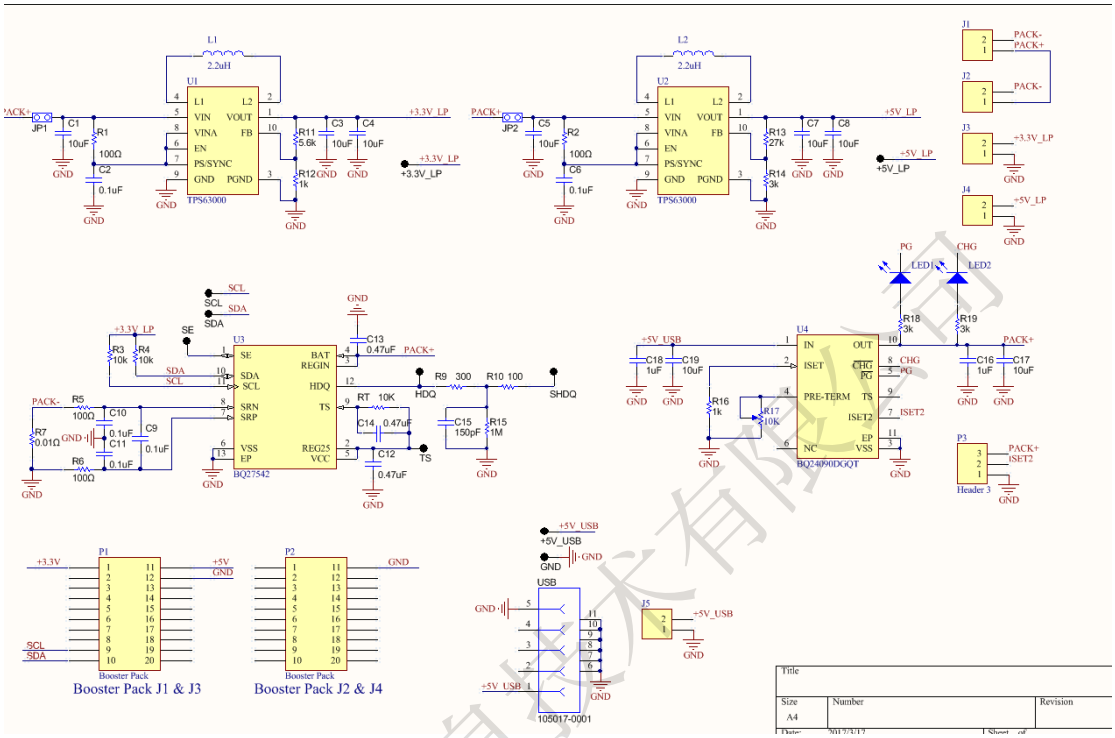


图 3-67 电池管理模块原理图

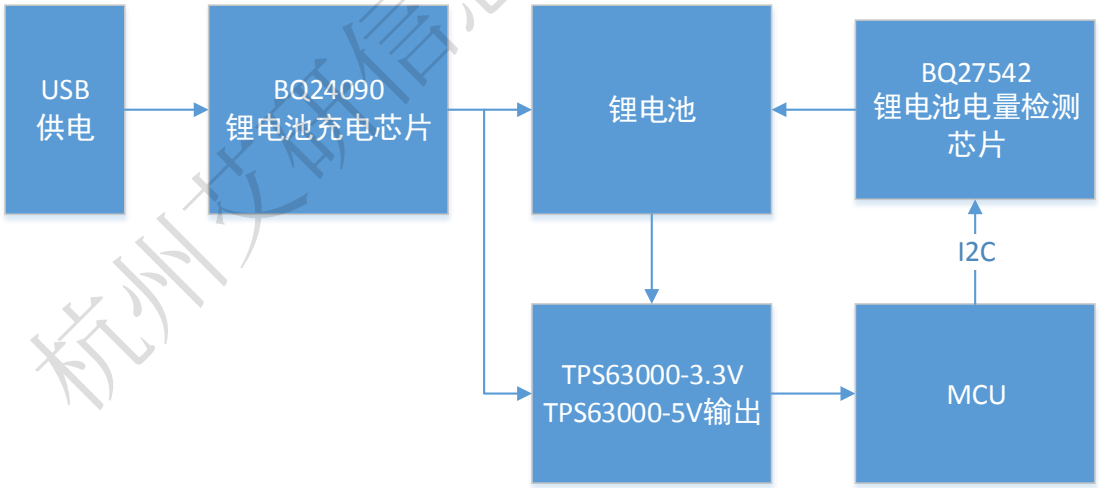


图 3-68 锂电池功能框图

本模块使用 USB 接口对整体进行供电，通过 BQ24090 对锂电池进行充电，BQ27542 对锂电池的电量进行检测，电量信息需通过 MCU 的 I2C 去读取 BQ27510 的内部信息。输出部分采用了两片 TPS63000，一片输出 3.3V，另一片输出 5V，用户可通过跳线帽 JP1 和 JP2 来选择使用

3.9.3 软件实现

3.9.3.1 BQ27542 驱动库实现

部分库函数如下：类对象“MelodyLiBattery”定义在 Melody_L.cpp 中。

表 3-46 begin()函数

驱动库函数	void begin(void)
说明	启动 I2C 功能
使用范例	MelodyLiBattery.begin();

表 3-47 getInternalTemperature()函数

驱动库函数	float getInternalTemperature(void)
说明	获取 BQ27542 内部温度
使用范例	float temperature = MelodyLiBattery.getInternalTemperature();

表 3-48 getVoltage()函数

驱动库函数	uint16_t getVoltage(void)
说明	获取当前电压值
使用范例	uint16_t voltage = MelodyLiBattery.getVoltage();

表 3-49 getAverageCurrent()函数

驱动库函数	int16_t getAverageCurrent(void)
说明	获取平均电流
使用范例	int16_t current = MelodyLiBattery.getAverageCurrent();

表 3-50 getRemainingCapacity()函数

驱动库函数	uint16_t getRemainingCapacity(void)
说明	获取当前剩余电量
使用范例	uint16_t capacity = MelodyLiBattery.getRemainingCapacity();

表 3-51 getStateOfCharge()函数

驱动库函数	uint8_t getStateOfCharge(void)
说明	获取当前剩余电量百分比
使用范例	uint8_t getStateOfCharge(void);

3.9.3.2 实验例程

例程 1，LiBattery.ino:获取当前模块的电压值，平均电流值，BQ27542 的内部温度值以及使用的锂电池的剩余电量和剩余电量百分比，这些数据都通过串口打印值 PC 串口助手。

例程流程图如图 3-69，例程代码实现如图 3-70。

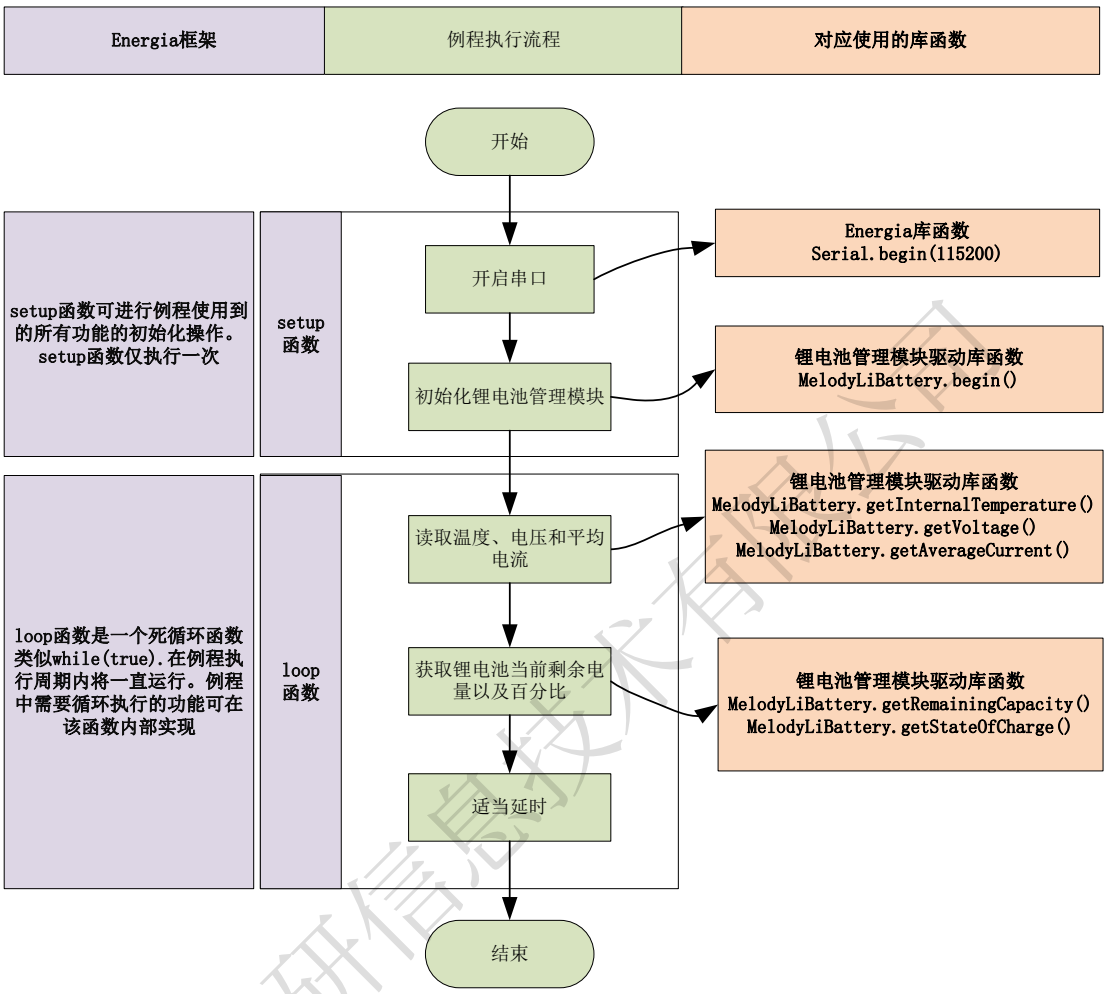


图 3-69 LiBattery 例程流程图

```
#include <stdbool.h>
#include <stdint.h>
#include "Wire.h"
#include "Melody_LiBattery.h"
void setup()
{
// put your setup code here, to run once:
  MelodyLiBattery.begin();
  Serial.begin(115200);
}

void loop()
{
// put your main code here, to run repeatedly:
  Serial.println("-------------------------------------");
  Serial.print("Temperature:");
  Serial.print(MelodyLiBattery.getInternalTemperature());
```

```
  Serial.println(" 'C");
  Serial.print("Voltage:");
  Serial.print(MelodyLiBattery.getVoltage());
  Serial.println(" mV");
  Serial.print("Current:");
  Serial.print(MelodyLiBattery.getAverageCurrent());
  Serial.println(" mA");
  Serial.print("Remaining Capacity:");
  Serial.print(MelodyLiBattery.getRemainingCapacity());
  Serial.println(" mAH");
  Serial.print("State of charge:");
  Serial.print(MelodyLiBattery.getStateOfCharge());
  Serial.println(" %");
  Serial.println("-----------------------------------");
  delay(2000);
}
```

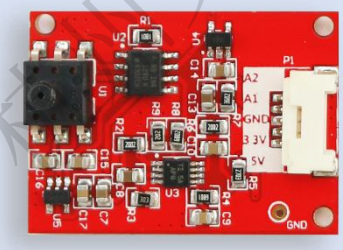
图 3-70 LiBattery 例程源程序

3.10 血压模块（MelodyBP）

3.10.1 模块

血压模块如表 3-52

表 3-52 血压模块

模块	概况
	1 使用芯片 MPS-3117 2 测量范围 5.8PSI 3 使用 AD 接口采集数据 3 可实现静态压和血压波动数据采集

3.10.2 实现原理及原理图简介

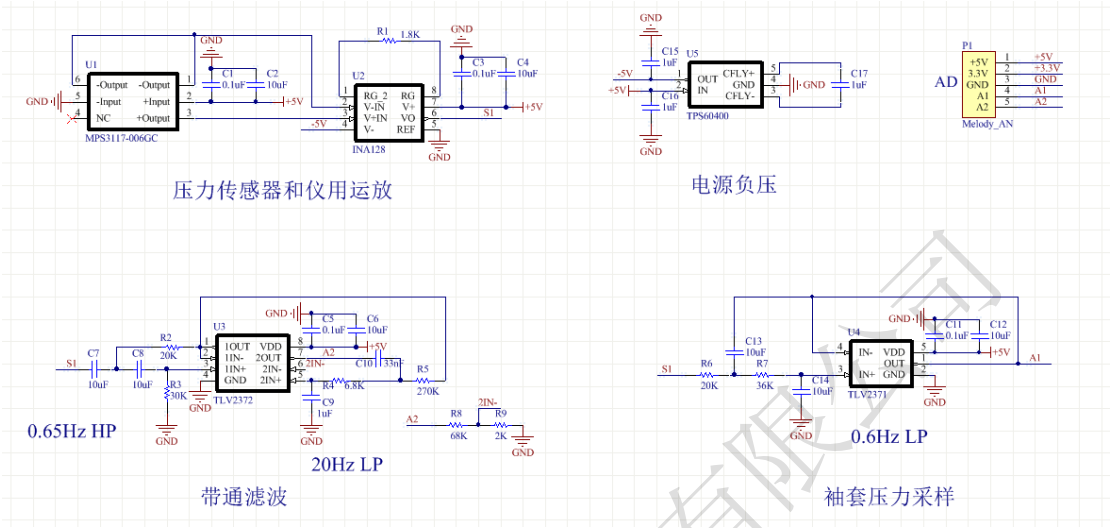


图 3-71 血压模块原理图

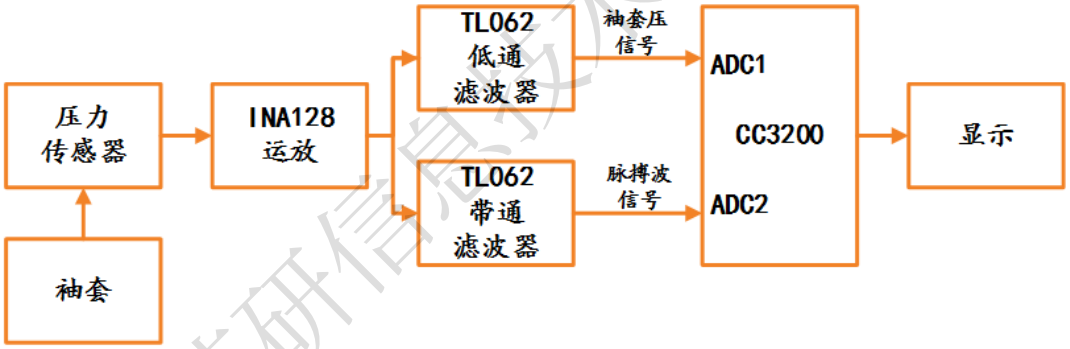


图 3-72 血压模块功能框图

硬件模块主要包括：CC3200、MPS-3117-006GC 压力传感器、INA128 精密低功耗仪器放大电路、TLV2371 低通、TLV2372 带通滤波电路。

压力传感器是电子血压计的核心部分之一，关系到整个系统的精度。本文采用的是台湾 METRODYNE 全磊微机电公司的 MPS3117-006GC 压阻式压力传感器，可经由定电压(5 V)或定电流(1, 1.5 mA)驱动产生正比于输入压力之毫伏等级电压输出讯号，具有优异的性能与长时间稳定性。MPS3117-006GC 具有以下特点：

（1） 测量范围为 0~5.8PSI，即 0~300mmHg；

（2） 操作温度范围-40~85℃；

（3） 供电电压：5V；

（4） 满量程输出：75mV；

（5） 灵敏度：0.25mV/mmHg

由于压力传感器输出电压较小，使用 INA128 仪器放大器对输出的压力信号进行放大，放大倍数 G=1+50KΩ /Rg，Rg 选择 1800Ω，即放大倍数 G=28.78，血压计的使用范围一般为 0~180mmHg，即 0~45mV，经放大后得到的压力信号范围为 0~1.295V。

3.10.3 软件实现

血压模块需要配套上位机软件一起使用，软件下载链接：http://www.hpati.com/ay_tools_download/，软件界面如下：

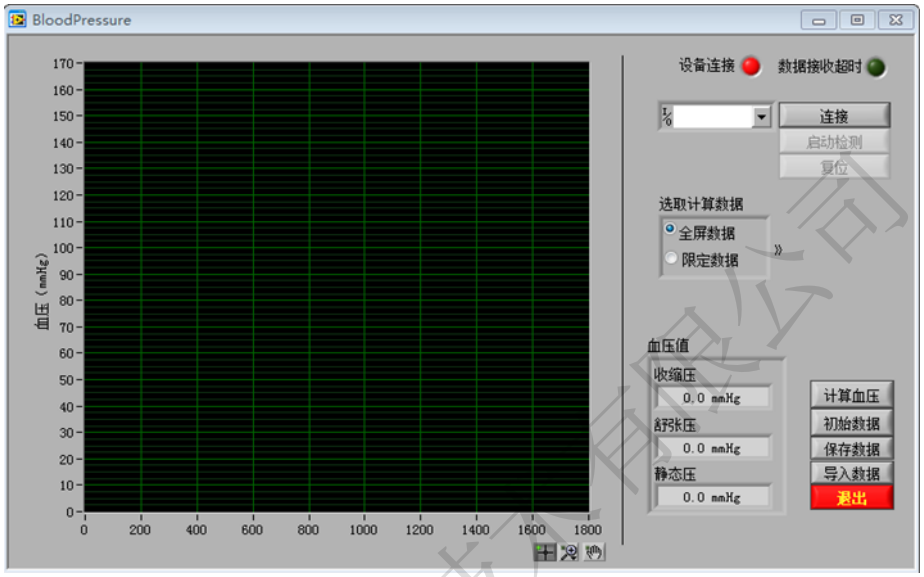


图 3-73 血压模块上位机软件

软件实现如下功能：

1 完成与血压模块之间的通信；
2 对血压模块采集的数据进行分析，波峰检测；
3 计算静态压、收缩压、舒张压；
4 保存和导入固定格式数据；
5 可将采集数据导出成 excel 格式数据；
6 可导出图像数据；

对于软件操作可查看视频：http://www.hpati.com/IOTKIT/v82.html

3.10.3.1 血压模块驱动库实现

模块部分库函数如下：类对象“MelodyBP”定于在 Melody_BP.cpp 文件中。

表 3-53 begin()函数

驱动库函数	void begin(void)
说明	初始化设置模块
使用范例	MelodyBP.begin();

表 3-54 measurement()函数

驱动库函数	bool measurement(void)
说明	完成检测过程，返回值 bool 类型：true，检测完成；false，检测未完成。
使用范例	bool state = MelodyBP.measurement();

表 3-55 transmitBoolPressure()函数

驱动库函数	void transmitBoolPressure(HardwareSerial &h_serial)
说明	发送读取的血压数据至上位机。 参数：h_serial:使用的串口引用
使用范例	MelodyBP.transmitBoolPressure(Serial);

表 3-56 receiveComPacket()函数

驱动库函数	char* receiveComPacket(HardwareSerial &h_serial,char* p_Combuffer,uint8_t Comlength)
说明	接收串口数据 参数：h_serial，使用的串口引用 p_Combuffer，接收数据缓存 Comlength，接受数据长度
使用范例	MelodyBP.receiveComPacket(Serial,buffer,10);

表 3-57 available()函数

驱动库函数	bool available(void)
说明	判断数据是否接受完成 返回值：true，接受完成 false，未接受完成
使用范例	bool complete = MelodyBP.available();

表 3-58 handleComPacket()函数

驱动库函数	eComState handleComPacket(HardwareSerial &h_serial, char* p_Combuffer)
说明	处理接受的数据包 参数：h_serial，使用的串口引用 p_Combuffer,数据缓存 返回值：eComState，枚举类型，定义在 Melody_BP.h 文件值为：IDLE，ONLINE，MEASURE，RESET。用来表示数据包类型
使用范例	eComState state = MelodyBP.handleComPacket(Serial,buffer);

表 3-59 clear()函数

驱动库函数	void clear(void)
说明	清除数据和状态
使用范例	MelodyBP.clear();

3.10.3.2 实验例程

实验例程操作及结果展示可浏览视频资料：http://www.hpati.com/product_videos/v82.html

例程 1，BloodPressureSample.ino：与上位机程序配套使用，完成血压数据的采集和上传。例程流程图如图 3-74，程序实现如图 3-75。

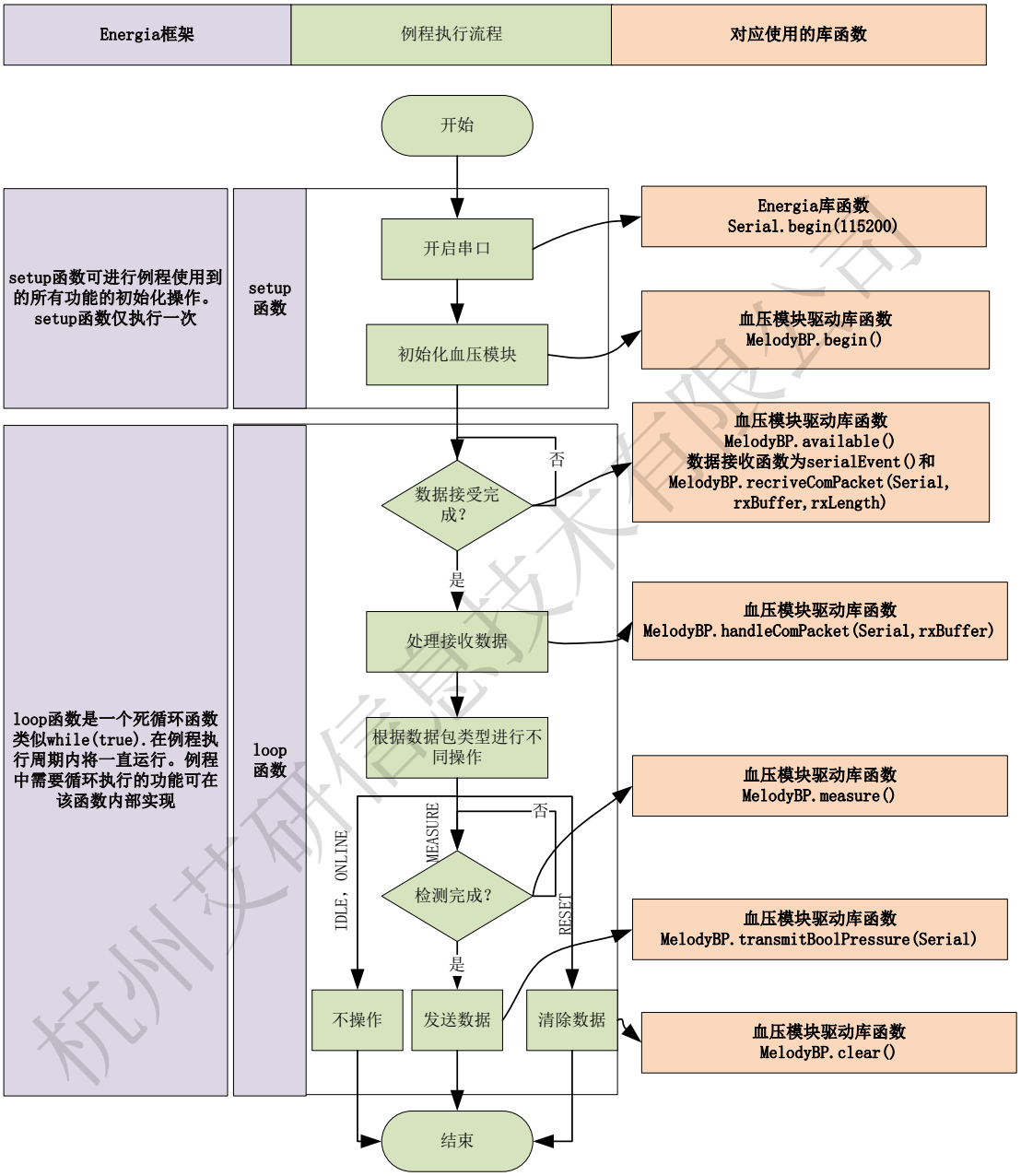


图 3-74 BloodPressureSample 例程流程图

```
#include <stdint.h>
#include <stdbool.h>
#include "Melody_BP.h"

constuint8_t rxLength =10;
char rxBuffer[rxLength]={0};
eComState runState = IDLE;
```

```
void setup()
{
// put your setup code here, to run once:
  Serial.begin(115200);
  MelodyBP.begin();
}

void loop()
{
// put your main code here, to run repeatedly:
//if(MelodyBP.available() && (runState != MEASURE))
if(MelodyBP.available())
{
    runState = MelodyBP.handleComPacket(Serial,rxBuffer);
    memset(rxBuffer,0,rxLength);
}
switch(runState)
{
case IDLE:
break;
case ONLINE:
//do something

       runState = IDLE;
break;
case MEASURE:
if(MelodyBP.measurement())
{
       MelodyBP.transmitBoolPressure(Serial);
       runState = IDLE;
}
break;
case RESET:
     MelodyBP.clear();
     runState = IDLE;
break;
default:
break;
}
}
void serialEvent()
{
  MelodyBP.receiveComPacket(Serial,rxBuffer,rxLength);
```

```
}
```

图 3-75 BloodPressureSample 例程源程序