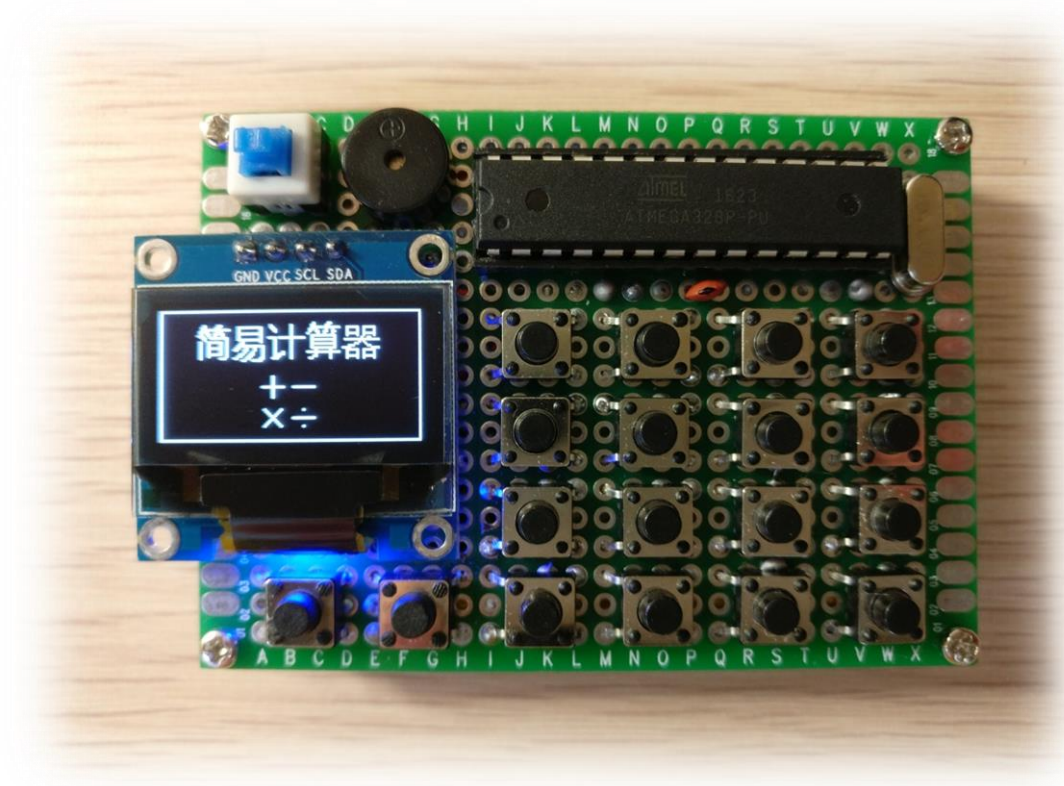
电子科技大学 University of Electronic Science and Technology of China

福州第一中学

2016 年“盟升杯”大学生电子设计竞赛

简易计算器（B 题）

【大一组】



创客七户 参赛队

许若阳　聂弘拓　陈铭

2016 年 10 月 23 日

摘　　要

计算器是最常用的便携中低端电子产品之一。即使在手机平板电脑盛行的今天，计算器仍是学生与工程师手上不可或缺的重要工具。

为了得到正确的运算数据和流畅的使用体验，总是需要大量的程序优化和精致的外观设计。本文介绍的，便是我们设计并制作的一款简易计算器。用户通过按键输入数字和运算符，可在屏幕上读取结果。

通过设计程序，该产品可以实现高精度计算和解方程功能，并有动画显示、多级删除、运算报错、功能菜单、音量设置和 LED 柔光等人性化设计。紧凑的键盘布局、精简的单片机最小系统以及锂电池设计，使得计算器只有手掌般大小，让产品便携耐用成为可能。

本次设计中的简易计算器将在运算性能和用户体验方面显示出它独特的优越性。

关键字：AVR 单片机　OLED　矩阵键盘　锂电池　可充电　紧凑　便携

目　　录

【本部分为题目要求全文】

2016 年盟升杯（第三届）竞赛试题

简易计算器(B题)

【大一组】

一 任务

设计并制作一个简易计算器，通过按键输入数字和运算符，并在数码管或液晶屏上显示结果。

二 要求

1. 基础部分

1.1 点亮数码管或液晶屏，并同时显示4个不同数字。

1.2 扫描键盘，并显示按键标号。

1.3 实现整数的基本运算功能（规定整数范围：-999~999）。

1.4 超出运算范围或出错后提示，要求可以退出错误状态。

2. 发挥部分

2.1 存储一个数字，掉电不丢失。

2.2 实现小数的基本运算。

2.3 实现6位有效数字或更高精度。

2.4 解aX + *e*^X=b方程。

2.5 其他（如：自制最小系统，更多位数，更多运算等）。

三 说明

1. 可以使用成品开发板；

2. 基本运算为加减乘除。

3. 解方程精度要求在0.01，给定方程只有一解，a>0，b>0，0<X<20；a,b从键盘输入，运算时间最长不超过1秒。

4. 计算器需可以连续按键计算，按下新运算符时自动计算出上一次结果，并作为运算的前一个数。不需要考虑优先级。

5. 报告需附上程序所有代码。

简易计算器（B 题）

【大一组】

1 系统方案

本系统主要由单片机模块、显示模块、供电模块、输入模块及少量其他元器件组成，下面分别论证这几个模块的选择。

1.1 单片机模块的论证与选择

方案一：选择 STC89C52 作为主控芯片。

STC89C52 周期比较慢，使系统反应比较慢。

方案二：选择 ATMEGA328P-PU 作为主控芯片。

这块单片机运算速度快，运算性能强，组员有使用此平台开发作品的经验。经过分析论证，ATMEGA328P-PU 相对适合开发此题。由于全球使用此块单片机的开发者非常多，相关资料非常丰富，众多厂商都有对其的特别支持。它可以被轻松地自制最小系统，并应用在各种领域。同时，我们使用 ATMEL 公司 ATMEL Studio 7 集成开发环境进行代码编写。此方案从功能到开发环境都完全符合题设要求。

综合以上二种方案，选择方案二。

1.2 显示模块的论证与选择

方案一：使用 1602 液晶屏。

1602 显示屏显示内容少，引脚众多，模块体积较大，不利于产品小型化。

方案二：使用 12864 液晶屏。

12864 液晶屏比 1602 液晶屏能显示更多内容，能实现更丰富的功能，但仍存在占用引脚多、体积大的问题，使用不方便。

方案三：使用 OLED 显示屏。

OLED 显示屏在使用 I^2C 接口时只占用 2 个引脚，显示效果好，点阵密集，体积小，价格便宜，优于以上两款显示屏。且使用 Universal 8bit Graphics Library 可以方便地驱动 OLED，使我们有更多的精力进行界面的深度开发。

综合以上两种方案，选择方案三。

1.3 供电模块的论证与选择

本方案整体模块均支持较宽电压输入（3.3~5V）。

方案一：利用 USB 接口外接供电。

利用 USB 接口外接供电，电压稳定，但产品无法脱离外接电源单独使用，违反了计算器可携带的设计需求。

方案二：利用干电池供电。

电池可以方便地更换，不用担心没电。但干电池对环境有一定损害，且电池盒及电池较大较重，同样地，拿出去玩十分不便。

方案三：利用锂电池供电。

锂电池比能量高，使用寿命长，而且锂电池轻薄小巧，能使产品更加美观。机载锂电池可以方便产品的随时使用，而搭配常见于移动电源的锂电充电/保护模块可以大大增强产品的实用性与安全性。

综合以上两种方案，选择方案三。

1.4 输入模块及其他器件的论证与选择

本方案输入模块采用 4*4 矩阵键盘设计，附加一个开关、一个多重功能键和一个多级清除键，其他元件有蜂鸣器、LED 灯。

产品使用双板堆叠，四角利用双通铜柱进行连接、固定，下板承载充电及保护电路和锂电池，并通过排针排座将电源与上层板的主体电路连接。

2 理论分析与计算

2.1 高精度算法

我们采用的单片机只能分配 4Byte 的空间来存储 double 变量，也就是说尽管代码中开了 double，它实际也只能实现 float 的精度。在这种情况下，为了提升精度，难免让人想到信息学奥林匹克竞赛中的专门用来处理上百位整型或实型的加减乘除的入门算法-高精度算法。

这种算法将整数与小数部分作为字符串读入，利用一个字符串可存储 0~255 个字符的特性，在计算时将字符转换为数字逐位运算，便可极大程度上提升精度。

但考虑到我们做的只是简易计算器，也无需处理如此精确的数字，用 float 来存储完全可以胜任这项工作，我们便抛弃了这个想法。

2.2 解方程算法

由于我们的方案采用的芯片运算能力拔群，直接利用二分法即可在远不及 1 秒的情况下快速解出赛题要求的方程，并可达到足够高的精度，故不考虑使用牛顿迭代法等更加复杂的算法。

所以，我们用二分法解 aX+e^X= b 方程（精度为 1E-6，a>0，b>0）!

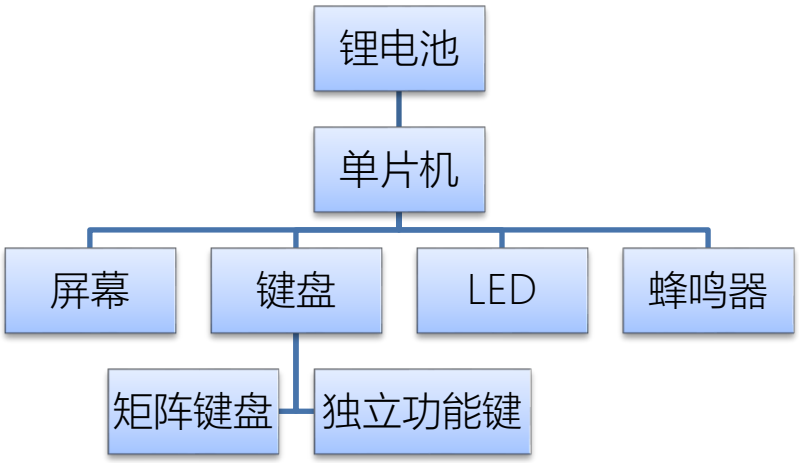
```
double solvefx() {      //二分法求解方程函数
double minimum = 0,maximum = 20,mid = (minimum + maximum)/2;
while (maximum - minimum > 1e-6)
{
   if (fx(mid) > 0)maximum = mid;
   else if (fx(mid) < 0)minimum = mid;
   else break;
   mid = (minimum + maximum) / 2;
}
return mid;
}

double fx(double x) //超越方程
{
   return a * x + exp(x) - b;
}
```

3 电路与程序设计

3.1 系统组成

3.1.1 系统总体框图

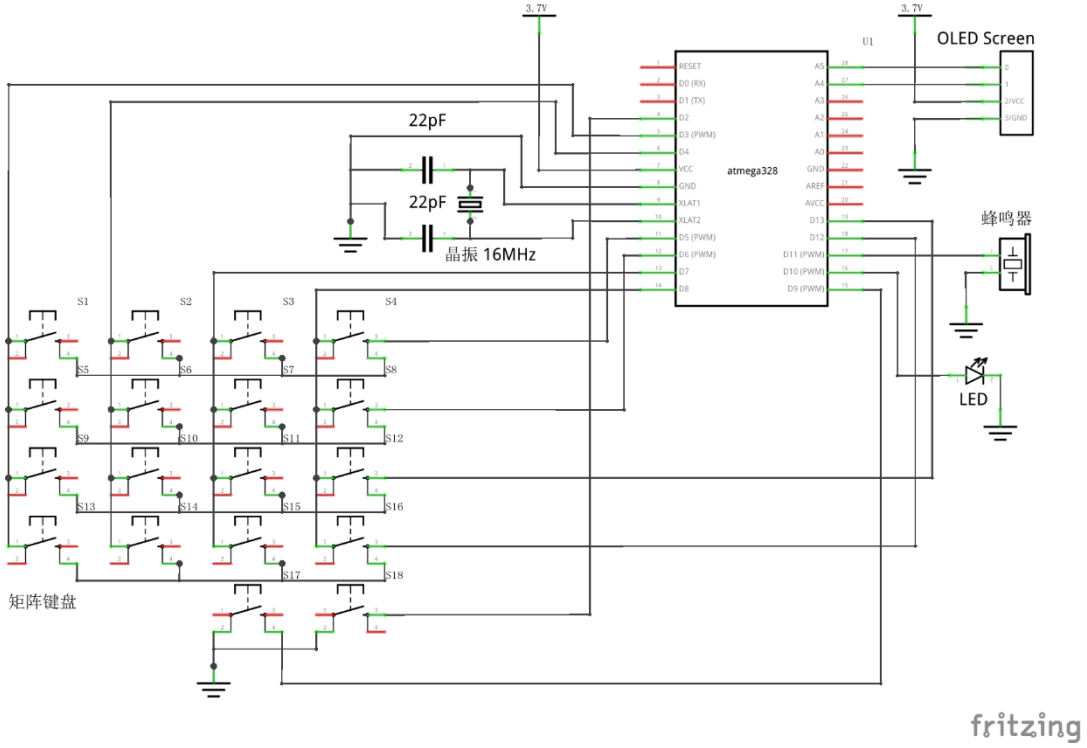


系统总体框图

单片机使用 ATMEL 的 ATMEGA328P-PU，显示采用了 0.96 寸的 OLED 屏，键盘使用矩阵键盘搭配两颗独立按键，蜂鸣器为有源型。其中，单片机与屏幕以 I^2C 方式通讯。

3.2 电路原理图

3.2.1 单片机及外围电路电路原理图



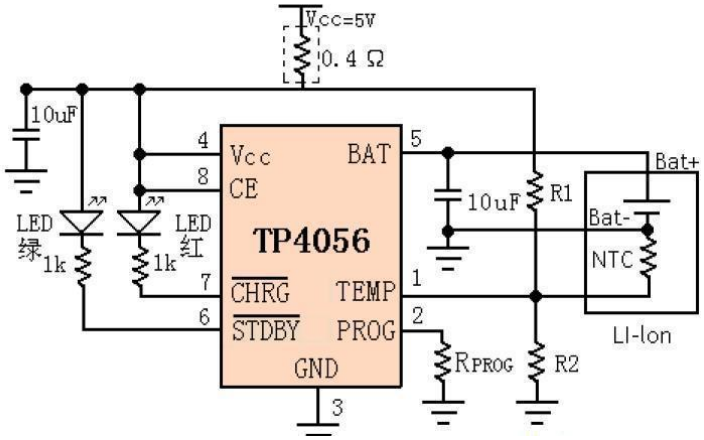
（大尺寸原图请见附件 1）

单片机采用 Atmega 的 ATMEGA328P-PU。通过添加一枚 16MHz 晶振和两枚 22pF 瓷片电容，构成可供 ATMEGA328P-PU 正常运行的最小系统。

4x4 键盘阵列采用经典的矩阵键盘设计，共占用 8 个引脚。在电路板上，此种设计能在达到紧凑目的的条件下不使用飞线。两枚独立按键与单片机数字引脚相连，直接读取电平值。

OLED 模块采用的是 I^2C 设计。两线式串行总线通过 A4、A5 两个模拟引脚与单片机通信。

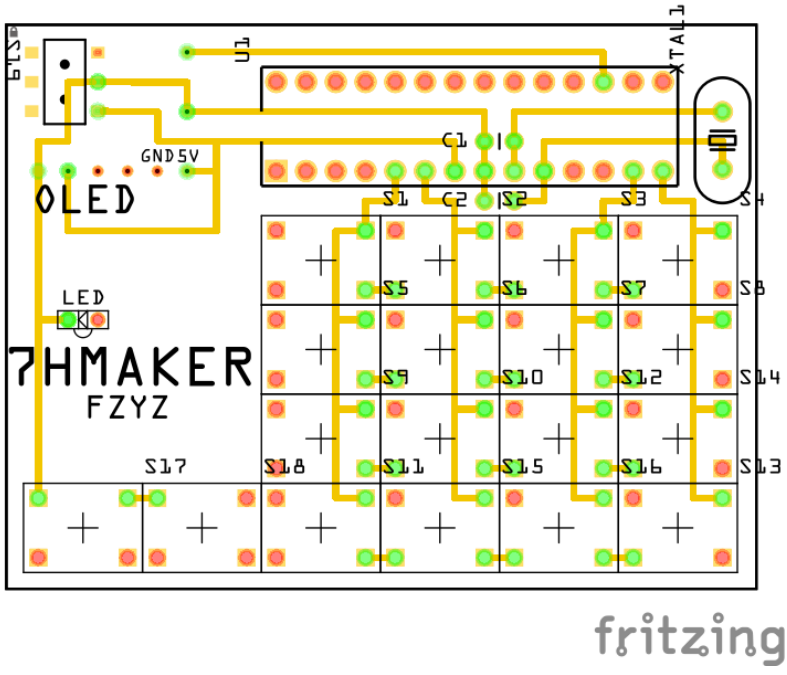
3.2.2 锂电池充电电路



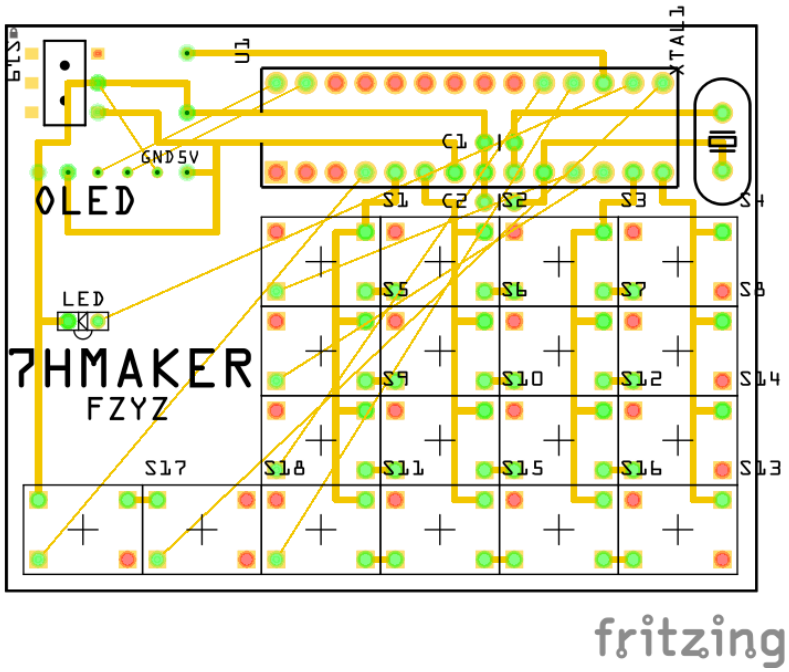
我们最终采用了市面上常见的 TP4056 电源管理芯片的典型应用方案。在板上焊上 Micro USB 接口，向 VCC 脚灌入 5V 电压，即可使用手机充电器对计算器进行充电。充电时板载 LED 亮红灯，充电完毕后亮绿灯。

3.2.3 电路板设计

电路板设计中，我们尽可能采用板上锡线，同时减少飞线堆叠，保证美观。我们同时设计了相同布局的 PCB 图样和飞线走向图样，方便您理解产品的电路布局，也为后续开发和 PCB 印制提供便利。



PCB 示意图（大尺寸原图请见附件 2）


U1
XTAL1
C1
GND 5V
OLED
LED
S1
C2
S2
S3
S4
S5
S6
S7
S8
S9
S10
S12
S14
S17
S18
S11
S15
S16
S13
7HMAKER
FZYZ
fritzing

PCB+飞线示意图

3.3 程序的设计

3.3.1 程序功能描述与设计思路

1、程序功能描述

可视化方面：

1) 产品所加载的程序，能够驱动 128*64 的单色 OLED 显示屏，能显示基础字符（ASCII）、汉字、点阵图像，能即时绘制多种几何图形；

2) 能够呈现具有现代设计感的用户友好型界面(User-friendly Interface)；

3) 能播放动画；

4) 能同屏同时显示大量数字；

5) 绘制了方便用户进入的引导界面。

外围功能方面：

1) 程序通过单片机控制有源蜂鸣器和 LED 状态指示灯，实现按键提示音、呼吸灯以及按下按键时 LED 闪烁一次等效果；

2) 通过 PWM 控制 LED 的亮度，防止灯光过于刺眼；

3) 通过程序实现按键防抖；

4) 赋予退格键双重功能，即一键多用：短按退格，长按清空整个屏幕；

5) 清除过程伴随动作提示音以防止误操作；

6) 在相关菜单中，用户可方便地设置蜂鸣器的开关，防止计算器在使用中打扰他人；

7) 可以设置运算结果保留的有效数字位数；

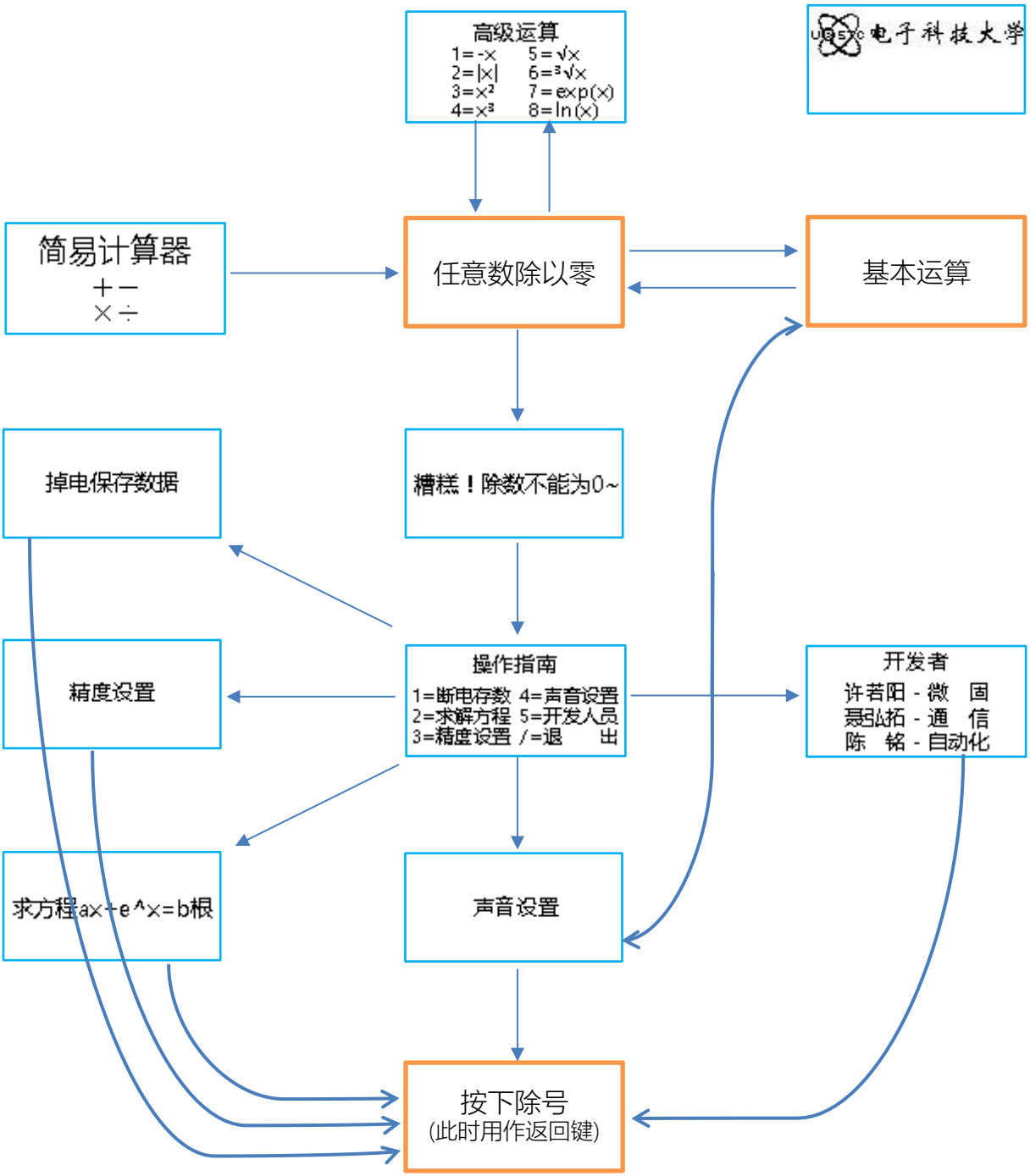
8) 所有设置内容均支持自动保存，且断电后设置不会丢失。

计算性能与输入习惯方面：

1) 屏幕布局整洁，可分行显示上一次运算结果与当前输入数字，具有独立的运算符号位；
2) 不允许输入的数以一个或更多的 0 开头，键入数字时自动覆盖已有的首位 0；
3) 实现小数的基本运算；
4) 通过程序智能识别是否存在小数点，只允许输入一个小数点；
5) 可控制输入数字长度，以防止超出运算精度范围；
6) 重复输入不同运算符时以最后一次输入为准；
7) 上一轮运算结束后按下数字即可开始新的运算过程；
8) 高级运算菜单中可实现以下功能：

   相反数、绝对值、平方、立方、开方、开立方、以 e 为底的指数运算、以 e 为底的对数运算；
9) 具有错误处理功能，超出运算范围或出错后能够给出对应提示，并可在不复位单片机的情况下通过按键退出错误状态；
10) 扫描键盘，并显示按键标号；
11) 实现总共 7 位有效数字的运算精度；
12) 掉电存储一个三位数（数字范围：0~999）；
13) 解 aX+e^X= b 方程（精度为 0.00001，a>0,b>0）；
14) 菜单可逐级返回。

3.3.2 程序流程图


高级运算
1=-x 5=√x
2=|x| 6=³√x
3=x² 7=exp(x)
4=x³ 8=ln(x)
电子科技大学
简易计算器
+ −
× ÷
任意数除以零
基本运算
掉电保存数据
糟糕！除数不能为0~
操作指南
1=断电存数 4=声音设置
2=求解方程 5=开发人员
3=精度设置 /=退 出
精度设置
开发者
许若阳 - 微 固
聂弘拓 - 通 信
陈 铭 - 自动化
求方程ax+e^x=b根
声音设置
按下除号
(此时用作返回键)

4 测试方案与测试结果

4.1 测试方案

1、硬件测试

我们在某款 AVR 开发板上进行相关模块的实验，将电路临时搭建面包板上进行开发。焊接到洞洞板上后利用万用表检测电路，准确地发现了一颗按键存在虚焊问题并修复，并发现了某处由于飞线绝缘皮的脱落导致的短路问题。在未安装电池模块时使用数字可调电源对作品的电压输入范围进行核对。组装完毕后，经测试，我们的锂电池、电池充电兼保护模块、单片机及其余 I/O 设备工作正常。

2、软件仿真测试

我们选择使用计算机上的 Matlab R2014b 和 Dev-C++ 5.11 对解方程算法进行优选，利用 Atmel Studio 7.0 的串口通信功能进行软件调试，

3、硬件软件联调

烧录程序后我们在搭好的最小系统上对各种可能的输入情况进行测试 ,同步优化代码，取得最佳平衡。

4.2 测试条件与仪器

符合 USB 电压标准（5V）的充电器

数字可调电源

数字万用表

4.3 测试结果及分析

经过测试，系统各部分均工作正常。在测试中我们发现，当运算超出精度范围时，运算结果存在一定误差，但降低输入数字位数后恢复精度。

4.3.1 测试结果(数据)

条件	a>0 b>0 0<x<20				
组数	参数 a	参数 b	电脑出解	计算器出解(5 位)	计算器出解(6 位)
1	1	2	0.4428544641	0.44285	0.442854
2	1	10	2.070579827	2.07058	2.070580
3	5	10	1.280390918	1.28039	1.280391
4	666666	999999	1.4999935031	1.49999	1.499993
5	0.123456	0.789456	0.000000298	0.00000	0.000000
6	0.123456	19.65432	2.9595324397	2.95953	无法出解
7	1.123	2.456	0.586389005	0.58639	0.586389
8	2.456789	5.789456	1.115103066	1.11510	1.115103
9	5.456789	19.65432	2.102119625	2.10212	2.102120
10	19.65432	1.234567	0.011353791	0.01135	0.011354
11	19.12345	19.98765	0.914672315	0.91467	0.914672

“电脑出解”：在电脑上利用 Dev-C++ 5.11 开双精度(double)进行运算所得结果；

“计算器出解(5 位)”：在计算器中烧录 5 位小数解方程程序后（即计算器目前烧录的程序），得到的结果；

“计算器出解(6 位)”：修改程序，提升计算器解方程精度后，烧录新的程序，得到的结果；

将结果与计算器上的 5 位及 6 位小数两种模式下的输出结果进行记录与对比，得到以上表格。

可以看出，在大部分情况下，我们的单片机 ATMEGA328P 以单精度(float)输出 5 位小数点时能在大部分情况下不损失精度稳定输出，6 位小数则在第 6 组中产生问题，无法出解。

理论分析，float 的精度能保证在 6 位有效数字。由于给定结果整数部分最大为两位，故让计算器输出 4 位小数是能够 100%保证精度的。

4.3.2 测试分析与结论

根据上述测试数据，系统能完美地实现赛题要求，计算精度领先同行，计算速度可超大多竞争对手；界面简明美观，布局清晰简洁；作品板载电池，可用智能手机充电线(Micro USB)充电，而且易于携带，差不多和校园卡一样大，最重要的是：比一元硬币还薄！

综上所述，本设计达到设计要求。我们重视工业设计细节，注重程序代码优化，时刻将用户体验放在第一位。讲道理，这是件不可多得的好作品啊！

5 参考资料

5.1 在线文档

1. Universal 8bit Graphics Library For OLED https://github.com/olikraus/u8glib

2. atof – C++ Function http://www.cplusplus.com/reference/cstdlib/atof/

3. Matlab to C or C++ http://stackoverflow.com/questions/4166755/matlab-to-c-or-c

5.2 书籍

《C 程序设计（第四版）》 谭浩强 著 清华大学出版社

6 开发工具

6.1 代码编写

Atmel Studio 7.0

Dev-C++ 5.11

Matlab R2014b

6.2 电路设计

Fritzing 0.9.3b

6.3 外壳设计

SolidWorks Premium 2014

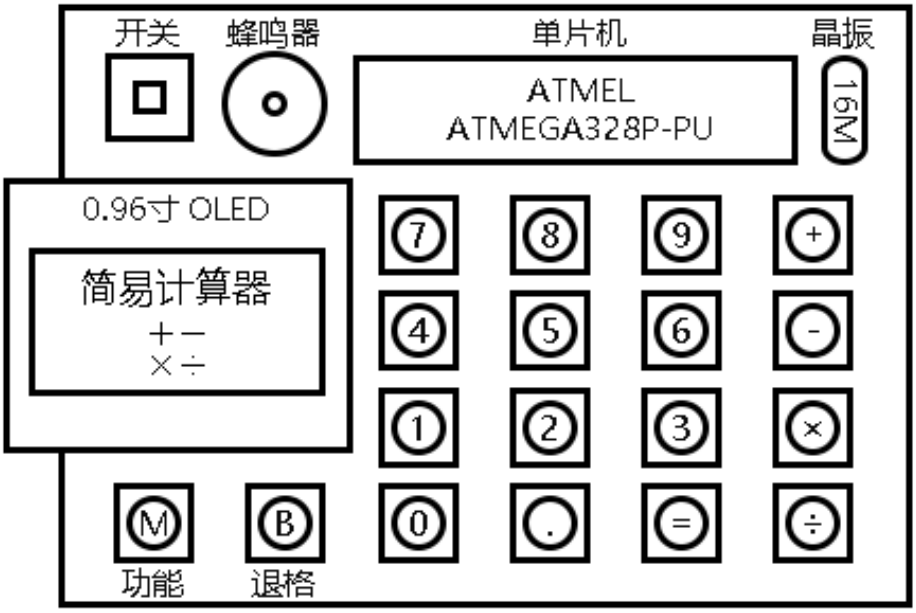
6.4 图形处理

PC to LCD 2002

Photoshop CS6

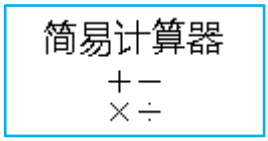
MSPaint

附件 1　使用说明书



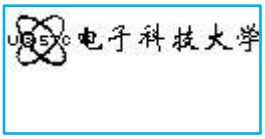
布局及按键

按下开关，等待开机动画加载完毕。若声音开关打开会听到蜂鸣器发出“嘀”声，表示系统加载完毕。



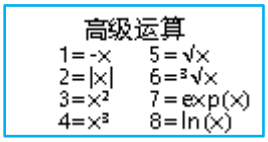
按下任意按键进入基础运算界面。

短按退格逐个数字删除，长按清空当前数值。

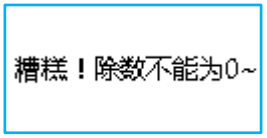


按下 M 键，唤出高级运算菜单。

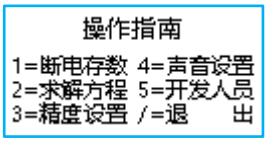
按下对应序号数字键，可对现有数值进行相应运算。



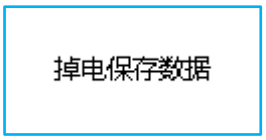
通过任意数除以 0，可看到报错信息，同时唤出菜单。



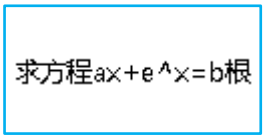
根据操作指南选取对应功能。在该菜单下按除号/键可回到基本运算界面，在任意子菜单下按除号/键可回到操作指南界面。



断电存数：输入任意值后，无需按下任何保存按键即可保存最后的输入状态。掉电后回到该界面，就可以看到保存的数值。

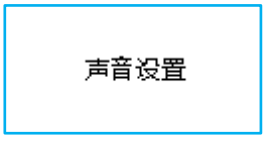


求解方程：在该界面下输入 a 的值，按下=赋值；对 b 操作相同。屏幕随即会显示方程的解。



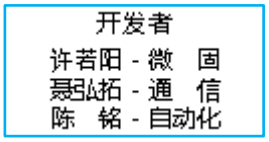
精度设置：此项可以设置计算器显示的小数点后的位数，值可为 0~6，设置即时生效。

声音设置：按下 1 打开声音，按下 0 关闭声音，设置即时生效。

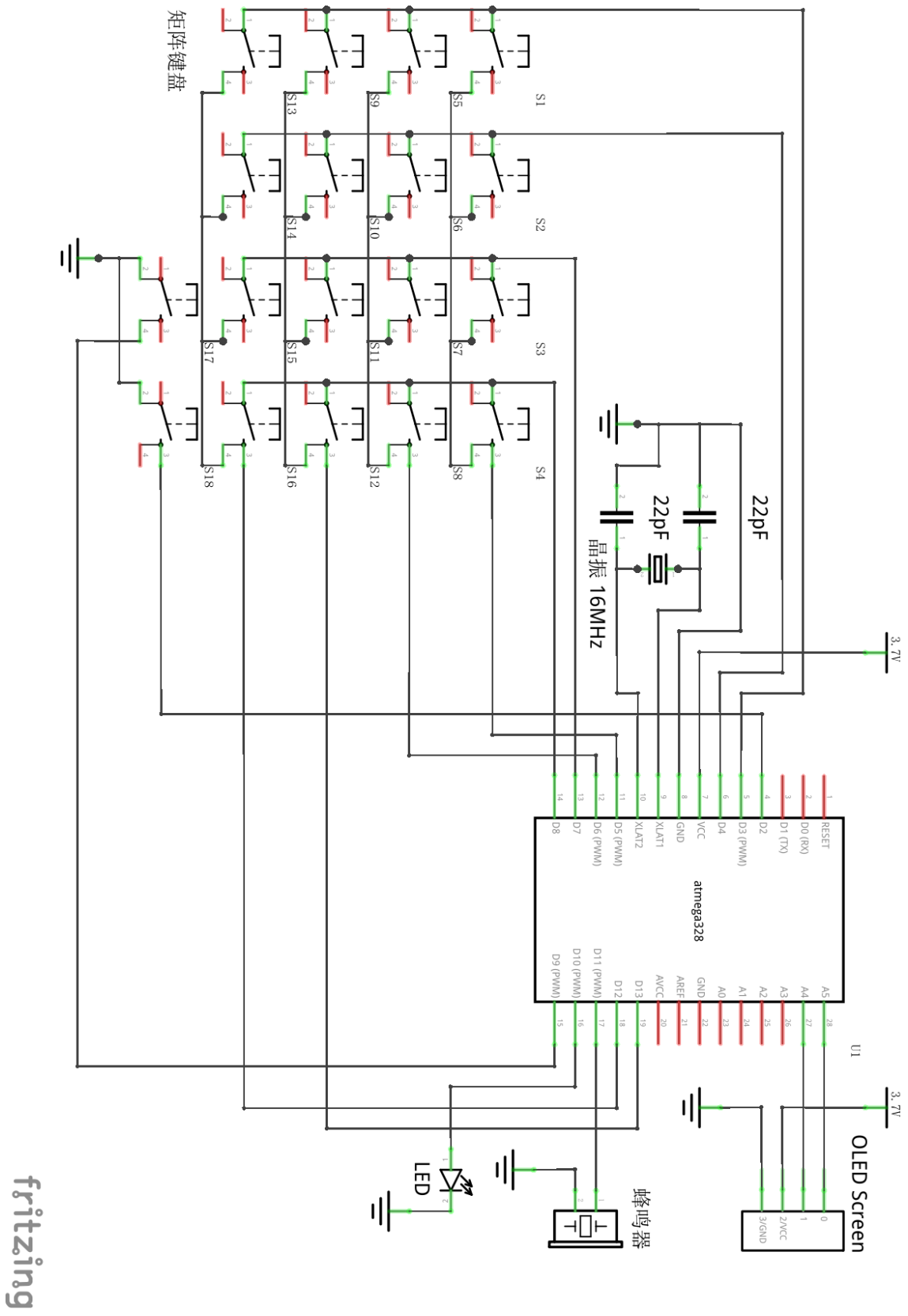


开发人员：在这里可以看到开发者名单和他们的所在学院。

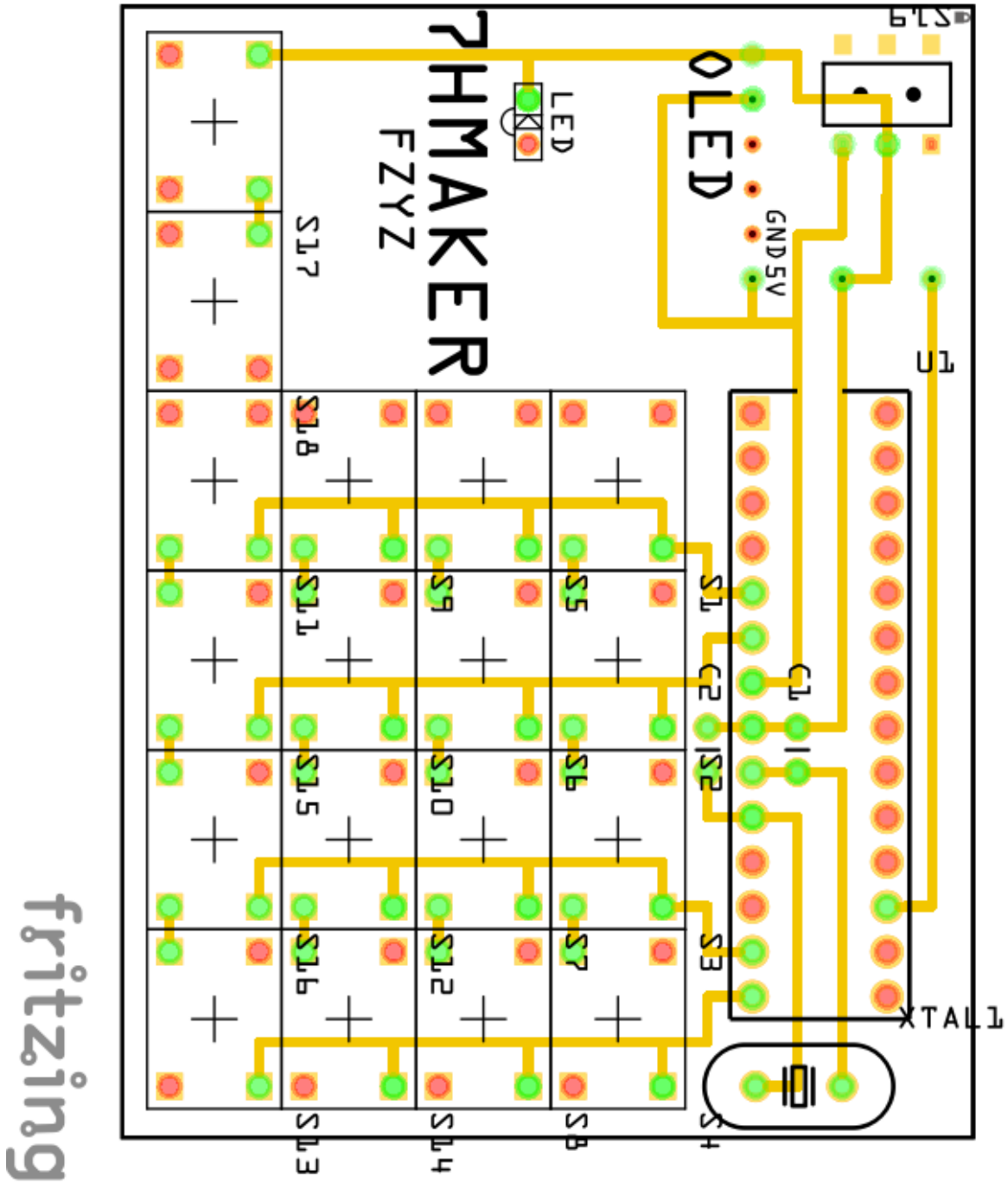
（我们来自同一所高中！耶！）



附件 2　电路图


矩阵键盘
S1
S2
S3
S4
S5
S6
S7
S8
S9
S10
S11
S12
S13
S14
S15
S16
S17
S18
22pF
22pF
晶振 16MHz
3.7V
atmega328
U1
RESET
D0 (RX)
D1 (TX)
D2
D3 (PWM)
D4
VCC
GND
XLAT1
XLAT2
D5 (PWM)
D6 (PWM)
D7
D8
A5
A4
A3
A2
A1
A0
GND
AREF
AVCC
D13
D12
D11 (PWM)
D10 (PWM)
D9 (PWM)
3.7V
OLED Screen
LED
蜂鸣器
fritzing

附件 3　PCB 图


OLED
LED
GND 5V
U1
XTAL1
S1
S2
S3
S4
S5
S6
S7
S8
S9
S10
S11
S12
S13
S14
S15
S16
S17
S18
C1
C2
fritzing

附件 4　程序代码

```
//简易计算器 2016/10/23
#include <EEPROM.h> //导入 EEPROM 操作库
#include <Keypad.h> //导入扫描键盘库
#include <U8glib.h> //导入 OLED 驱动库
U8GLIB_SSD1306_128X64 u8g(U8G_I2C_OPT_NONE); //采用 SSD1306 芯片方案并以 I2C 方式与 OLED 进行沟通
const byte ROWS = 4; //设置矩阵键盘行数为 4
const byte COLS = 4; //设置矩阵键盘列数为 4
char str[9] = ""; //以字符串形式存储输入的数字内容
int x = 0; //动画坐标
int symbol = 0; //运算符号位
double a = 0; //方程参数 a
double b = 0; //方程参数 b
double result = 0; //基本运算结果
short arrow = 0; //字符串指针
short buzzer = 0; //声音
short deltime = 0; //删除键按下时长
short digits = 0; //精度
short light = 0; //LED 强度
short maxlength = 0; //允许输入的最大长度
short menu = 0; //附加功能状态
short numtosave = 0; //预存数字
short savednum = 0; //已存数字
short stat = 0; //状态指示
char keys[ROWS][COLS] = { //开数组以映射按键
  {'7', '8', '9', '+'},
  {'4', '5', '6', '-'},
  {'1', '2', '3', '*'},
  {'0', '.', '=', '/'}
};
byte rowPins[ROWS] = {5, 6, 13, 12}; //设置矩阵键盘行引脚
byte colPins[COLS] = {3, 4, 7, 8}; //设置矩阵键盘列引脚

static unsigned char u8g_welcome[] U8G_PROGMEM = //欢迎
{
  因篇幅限制，此处 16 进制点阵图数据省略
};

static unsigned char u8g_uestc[] U8G_PROGMEM = //UESTC
{
  因篇幅限制，此处 16 进制点阵图数据省略
};

static unsigned char u8g_error[] U8G_PROGMEM = //错误
{
  因篇幅限制，此处 16 进制点阵图数据省略
};

static unsigned char u8g_menu[] U8G_PROGMEM = //菜单
{
  因篇幅限制，此处 16 进制点阵图数据省略
};
```

```
static unsigned char u8g_eeprom[] U8G_PROGMEM = //掉电保存
{
  因篇幅限制，此处 16 进制点阵图数据省略
};

static unsigned char u8g_fx[] U8G_PROGMEM = //解方程
{
  因篇幅限制，此处 16 进制点阵图数据省略
};

static unsigned char u8g_developers[] U8G_PROGMEM = //开发者
{
  因篇幅限制，此处 16 进制点阵图数据省略
};

static unsigned char u8g_help[] U8G_PROGMEM = //操作指南
{
  因篇幅限制，此处 16 进制点阵图数据省略
};

static unsigned char u8g_setbuzzer[] U8G_PROGMEM = //声音设置
{
  因篇幅限制，此处 16 进制点阵图数据省略
};

static unsigned char u8g_setdigits[] U8G_PROGMEM = //精度设置
{
  因篇幅限制，此处 16 进制点阵图数据省略
};

static unsigned char u8g_exfunc[] U8G_PROGMEM = //高级运算
{
  因篇幅限制，此处 16 进制点阵图数据省略
};

Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );

void setup() {
  //EEPROM.write(3, 2); 擦除位数设置
  digits = EEPROM.read(3); //读取位数设置
  buzzer = EEPROM.read(4); //读取声音设置
  pinMode(2, INPUT); //设置高级运算按键引脚为输入方式
  pinMode(9, INPUT); //设置退格按键引脚为输入方式
  pinMode(11, OUTPUT); //设置蜂鸣器引脚为输出方式
  pinMode(10, OUTPUT); //设置 LED 引脚为输出方式
  digitalWrite(2, HIGH); //开启接口内部上拉电阻
  digitalWrite(9, HIGH); //开启接口内部上拉电阻
  digitalWrite(11, LOW); //将蜂鸣器置于低电平
  u8g.setFont(u8g_font_courB14); //设置字体
  do {
    x = x + 8;
    light = light + 10;
    analogWrite(10, light);
    u8g.firstPage(); //开机画面
    do {
      u8g.drawXBMP( 0, 0, 128, 64, u8g_welcome);
      u8g.drawHLine(0, 0, 2 * x);
      u8g.drawHLine( 128 - 2 * x, 63, 128);
```

```
      u8g.drawVLine(0, 0, x);
      u8g.drawVLine(127, 64 - x, 64);
    } while (u8g.nextPage());
  } while (x < 64);
  if (buzzer == 1) digitalWrite(11, HIGH); //开启蜂鸣器
  delay(50);
  digitalWrite(11, LOW); //关闭蜂鸣器
  analogWrite(10, 10);
  light = 0;
}

void loop() {
  char key = keypad.getKey(); //取得矩阵键盘按键情况
  if (digitalRead(9) == LOW) { //判断退格键是否按下
    deltime++;
    if (deltime <= 3) key = 'B'; //一级删除
    if (deltime == 6) { //二级删除
      key = 'B';
      arrow = 0; //指针复位
      memset(str, 0, sizeof(str)); //清空输入字符串
      symbol = 0; //清空运算符
    }
    screencontroller();
  }
  if (digitalRead(9) == HIGH) deltime = 0; //复位时长
  if (digitalRead(2) == LOW) { //判断高级运算键是否按下
    key = 'A';
    if (menu == 0) menu = 9;
  }
  if (key) {
    if (strchr(str, '.') == 0) maxlength = 6;
    else maxlength = 7;
    switch (key) //对返回的按键进行讨论
    {
      case 'A': //按下高级运算键
        break;

      case 'B': //按下退格键
        if (arrow != 0) arrow--; //前移指针时防止下标越界
        str[arrow] = '\0'; //通过在指针位置标记结束位来缩短字符串长度
        if (menu == 2) {
          numtosave = atoi(str);
          EEPROM.write(0, numtosave / 100); //将位 1 写入 EEPROM
          EEPROM.write(1, (numtosave / 10) % 10); //将位 2 写入 EEPROM
          EEPROM.write(2, numtosave % 10); //将位 3 写入 EEPROM
        }
        break;

      case '+': //按下加号
        if (menu == 0) {
          process();
          symbol = 1; //保存加号到符号位
        }
        break;

      case '-': //按下减号
        if (menu == 0) {
          process();
```

```
            symbol = 2; //保存减号到符号位
          }
          break;

        case '*': //按下乘号
          if (menu == 0) {
            process();
            symbol = 3; //保存乘号到符号位
          }
          break;

        case '/': //按下除号
          if (menu == 0) {
            process();
            symbol = 4; //保存除号到符号位
          } else if (menu == 1 || menu == 9) { //判断除号此时是否用于返回
            menu = 0;
          } else if (menu > 1 && menu != 9) menushow();
          break;

        case '.':
          if (menu != 1 && menu != 2 && menu != 6 && strchr(str, '.') == 0 && arrow <=
maxlength) str[arrow++] = key; //限制输入小数点为一位且控制字符串长度以防止下标越界
          break;

        case '=':
          if (menu == 0) {
            stat = 1; //记录等号按下状态
            process();
            symbol = 0; //将符号位恢复为初始状态
          } else if (menu == 3) {
            a = atof(str); //输入参数 a
            menu = 4;
            arrow = 0; //指针复位
            memset(str, 0, sizeof(str)); //清空输入字符串
          } else if (menu == 4) {
            b = atof(str); //输入参数 b
            menu = 5;
          }
          break;

        default:
          switch (menu) {
            case 0: //四则运算状态
              if (str[0] == '0' && arrow == 1) arrow = 0; //不允许输入的数以一个或更多的 0
开头

              if (arrow <= maxlength) str[arrow++] = key; //控制字符串长度以防止下标越界
              break;

            case 1: //已进入菜单状态
              switch (key) {
                case '0':
                  menu = 10; //进入按键说明
                  break;

                case '1':
                  menu = 2; //进入功能 1 并改变附加功能状态
                  savednum = EEPROM.read(0) * 100 + EEPROM.read(1) * 10 + EEPROM.read(2);
```

```
//将 EEPROM 中 3 个地址位的值进行合并
                    break;

                  case '2':
                    menu = 3;//进入功能 2 并改变附加功能状态
                    break;

                  case '3':
                    menu = 6; //进入功能 3 并改变附加功能状态
                    digits = EEPROM.read(3);
                    break;

                  case '4':
                    menu = 7; //进入功能 4 并改变附加功能状态
                    buzzer = EEPROM.read(4);
                    break;

                  case '5':
                    menu = 8; //进入功能 5 并改变附加功能状态
                    break;
                }
                break;

              case 2: //数值保存状态
                if (str[0] == '0' && arrow == 1) arrow = 0; //不允许输入的数以一个或更多的 0
开头
                if (arrow <= 2) str[arrow++] = key; //控制字符串长度以防止下标越界
                numtosave = atoi(str);
                EEPROM.write(0, numtosave / 100); //将位 1 写入 EEPROM
                EEPROM.write(1, (numtosave / 10) % 10); //将位 2 写入 EEPROM
                EEPROM.write(2, numtosave % 10); //将位 3 写入 EEPROM
                break;

              case 3:
                if (str[0] == '0' && arrow == 1) arrow = 0; //不允许输入的数以一个或更多的 0
开头
                if (arrow <= maxlength) str[arrow++] = key; //控制字符串长度以防止下标越界
                break;

              case 4:
                if (str[0] == '0' && arrow == 1) arrow = 0; //不允许输入的数以一个或更多的 0
开头
                if (arrow <= maxlength) str[arrow++] = key; //控制字符串长度以防止下标越界
                break;

              case 6:
                if (key - '0' <= 6) digits = key - '0';
                EEPROM.write(3, digits); //保存精度设置到 EEPROM
                break;

              case 7:
                if (key - '0' <= 1) buzzer = key - '0';
                EEPROM.write(4, buzzer); //保存声音设置到 EEPROM
                break;

              case 9:
                if (arrow != 0) result = atof(str); //判断对哪一次输入进行运算
                switch (key) {
```

```
                case '1': //-x
                  result = -result;
                  break;

                case '2': //|x|
                  result = abs(result);
                  break;

                case '3': //x^2
                  result =  result * result;
                  break;

                case '4': //x^3
                  result =  result * result * result;
                  break;

                case '5': //x^(1/2)
                  result =  sqrt(result);
                  break;

                case '6': //x^(1/3)
                  result =  pow(result, 1.0 / 3);
                  break;

                case '7': //exp(x)
                  result = exp(result);
                  break;

                case '8': //ln(x)
                  result = log(result);
                  break;
              }
              menu = 0; //退回基本运算界面
              arrow = 0; //指针复位
              memset(str, 0, sizeof(str)); //清空输入字符串
              stat = 1; //将状态改为运算完毕
              symbol = 0; //复位符号
              break;

            case 10: //已进入操作指南
              switch (key) {
                case '1':
                  menu = 2; //进入功能 1 并改变附加功能状态
                  savednum = EEPROM.read(0) * 100 + EEPROM.read(1) * 10 + EEPROM.read(2);
//将 EEPROM 中 3 个地址位的值进行合并
                  break;

                case '2':
                  menu = 3;//进入功能 2 并改变附加功能状态
                  break;

                case '3':
                  menu = 6; //进入功能 3 并改变附加功能状态
                  digits = EEPROM.read(3);
                  break;

                case '4':
                  menu = 7; //进入功能 4 并改变附加功能状态
```

```
                buzzer = EEPROM.read(4);
                break;

              case '5':
                menu = 8; //进入功能 5 并改变附加功能状态
                break;
            }
            break;
        }
        break;
    }
    analogWrite(10, 80);
    if (buzzer == 1) {
      digitalWrite(11, HIGH); //开启蜂鸣器
      delay(10);
      digitalWrite(11, LOW); //关闭蜂鸣器
    }
    screencontroller();
    analogWrite(10, 10);
  }
}

void process() {
  str[arrow] = '\0'; //对指针位置标记字符串结束
  if (arrow != 0) { //判断键入是否为空
    switch (symbol) //对上一次的符号位进行讨论
    {
      case 0: //不存在上一次的符号
        result = atof(str);
        break;

      case 1: //上一次的符号为加号
        result = result + atof(str); //执行加法运算
        break;

      case 2: //上一次的符号为减号
        result = result - atof(str); //执行减法运算
        break;

      case 3: //上一次的符号为乘号
        result = result * atof(str); //执行乘法运算
        break;

      case 4: //上一次的符号为除号
        if (atof(str) != 0) result = result / atof(str); //在除数不为 0 的情况下执行除
法运算
        else stat = 2; //若除数为 0 则标记状态异常
        break;
    }
    arrow = 0; //指针复位
    memset(str, 0, sizeof(str)); //清空输入字符串
  }
}

void screencontroller() {
  switch (menu) {
    case 0: //未进入菜单的画面绘制
      switch (stat) {
```

```
case 0: //四则运算的画面绘制
  if (symbol == 0) { //无符号输入界面
    u8g.firstPage();
    do {
      u8g.drawXBMP( 0, 0, 128, 32, u8g_uestc);
      u8g.setPrintPos(0, 46);
      u8g.print(str);
    } while (u8g.nextPage());
  } else { //带符号输入界面
    u8g.firstPage();
    do {
      u8g.drawXBMP( 0, 0, 128, 32, u8g_uestc);
      u8g.setPrintPos(0, 46);
      u8g.print(result, digits);
      u8g.setPrintPos(0, 63);
      u8g.print(str);
      u8g.setPrintPos(127 - 11, 63);
      switch (symbol) {
        case 1:
          u8g.print('+');
          break;

        case 2:
          u8g.print('-');
          break;

        case 3:
          u8g.print('*');
          break;

        case 4:
          u8g.print('/');
          break;
      }
    } while (u8g.nextPage());
  }
  break;

case 1: //运算结果的画面绘制
  u8g.firstPage();
  do {
    u8g.drawXBMP( 0, 0, 128, 32, u8g_uestc);
    u8g.setPrintPos(0, 46);
    u8g.print(result, digits);
    u8g.setPrintPos(127 - 11, 63);
    u8g.print('=');
  } while (u8g.nextPage());
  stat = 0;
  break;

case 2: //出错的画面绘制
  x = 1;
  do { //除数为 0 的弹错动画
    light = light + 10;
    analogWrite(10, light);
    u8g.firstPage();
    do {
      u8g.drawXBMP( 0, 0, 128, 32, u8g_error);
```

```
        u8g.setPrintPos(x, 46);
        u8g.print("Error!");
        u8g.setPrintPos(61 - x, 63);
        u8g.print("Orz...");
      } while (u8g.nextPage());
      x = x + 10;
    } while (x < 61);
    delay (500);
    menushow();
    if (buzzer == 1) digitalWrite(11, HIGH); //开启蜂鸣器
    delay(50);
    digitalWrite(11, LOW); //关闭蜂鸣器
    analogWrite(10, 10);
    light = 0;
    break;
  }
  break;

case 2: //断电存储的画面绘制
  u8g.firstPage();
  do {
    u8g.drawXBMP( 0, 0, 128, 32, u8g_eeprom);
    u8g.setPrintPos(0, 46);
    u8g.print("EEPROM:");
    u8g.setPrintPos(77, 46);
    u8g.print(savednum);
    u8g.setPrintPos(0, 63);
    u8g.print("Rewrite:");
    u8g.drawStr(127 - u8g.getStrWidth(str), 63, str);
  } while (u8g.nextPage());
  break;

case 3: //解方程一级画面绘制
  u8g.firstPage();
  do {
    u8g.drawXBMP( 0, 0, 128, 32, u8g_fx);
    u8g.setPrintPos(0, 46);
    u8g.print("STEP 1");
    u8g.setPrintPos(0, 63);
    u8g.print("a=");
    u8g.drawStr(127 - u8g.getStrWidth(str), 63, str);
  } while (u8g.nextPage());
  break;

case 4: //解方程二级画面绘制
  u8g.firstPage();
  do {
    u8g.drawXBMP( 0, 0, 128, 32, u8g_fx);
    u8g.setPrintPos(0, 46);
    u8g.print("STEP 2");
    u8g.setPrintPos(0, 63);
    u8g.print("b=");
    u8g.drawStr(127 - u8g.getStrWidth(str), 63, str);
  } while (u8g.nextPage());
  break;

case 5: //解方程三级画面绘制
  u8g.firstPage();
```

```
    do {
      u8g.drawXBMP( 0, 0, 128, 32, u8g_fx);
      u8g.setPrintPos(0, 46);
      u8g.print("Result:");
      u8g.setPrintPos(0, 63);
      u8g.print("x=");
      u8g.setPrintPos(22, 63);
      u8g.print(solvefx(), 5);
    } while (u8g.nextPage());
    break;

  case 6: //精度设置画面绘制
    u8g.firstPage();
    do {
      u8g.drawXBMP( 0, 0, 128, 32, u8g_setdigits);
      u8g.setPrintPos(59, 52);
      u8g.print(digits);
    } while (u8g.nextPage());
    break;

  case 7: //声音设置画面绘制
    u8g.firstPage();
    do {
      u8g.drawXBMP( 0, 0, 128, 32, u8g_setbuzzer);
      if (buzzer == 0)
      {
        u8g.setPrintPos(47, 52);
        u8g.print("OFF");
      }
      else
      {
        u8g.setPrintPos(52, 52);
        u8g.print("ON");
      }
    } while (u8g.nextPage());
    break;

  case 8: //开发者画面绘制
    u8g.firstPage();
    do {
      u8g.drawXBMP( 0, 0, 128, 64, u8g_developers);
    } while (u8g.nextPage());
    break;

  case 9: //高级运算功能
    u8g.firstPage();
    do {
      u8g.drawXBMP( 0, 0, 128, 64, u8g_exfunc);
    } while (u8g.nextPage());
    break;

  case 10: //操作指南画面绘制
    u8g.firstPage();
    do {
      u8g.drawXBMP( 0, 0, 128, 64, u8g_help);
    } while (u8g.nextPage());
    break;
}
```

```
}

void menushow() { //主菜单界面
  menu = 1;
  stat = 0;
  arrow = 0; //指针复位
  memset(str, 0, sizeof(str)); //清空输入字符串
  result = 0;
  u8g.firstPage(); //进入菜单的动画
  do {
    u8g.drawXBMP( 0, 0, 128, 32, u8g_menu);
    u8g.setPrintPos(16, 55);
    u8g.print("-PRESS 0-");
    u8g.drawFrame(20, 39, 90, 21);
  } while (u8g.nextPage());
}

double solvefx() { //二分法求解方程函数
  double minimum = 0, maximum = 20, mid = (minimum + maximum) / 2;
  while (maximum - minimum > 1e-6)
  {
    if (fx(mid) > 0)maximum = mid;
    else if (fx(mid) < 0)minimum = mid;
    else break;
    mid = (minimum + maximum) / 2;
  }
  return mid;
}

double fx(double x) //超越方程
{
  return a * x + exp(x) - b;
}
```