浙江大学实验报告

课程名称：嵌入式系统 实验类型：综合型/设计性

实验项目名称：Lab 7：室温计

学生姓名：陈以珊 专业：计科 学号：3120101972

电子邮件地址：243630184@qq.com 手机：18868104906

实验日期：15 年 06 月 19 日

一、实验目的

1. 学习 uC/OS II 的应用程序编写；
2. 理解如何直接操纵 GPIO，体会与 Linux 的不同；
3. 学习单总线设备的访问方式；
4. 学习 7 段数码管的时分复用驱动方式。

二、实验器材

硬件

- pcDuino v2 板一块；
- 5V/1A 电源一个；
- microUSB 线一根；
- 面包板一块；
- 两位 7 段数码管（共阳）一颗；
- 8 段 LED 柱状显示器一颗；
- 360Ω 1/8W 电阻 8 颗；
- 10k 1/8W 电阻 2 颗；
- 按钮两个；
- 面包线若干。

以下为自备（可选）器材：

- PC（Windows/Mac OS/Linux）一台；
- USB-TTL 串口线一根（FT232RL 芯片或 PL2303 芯片）；
- 以太网线一根（可能还需要路由器等）；
- 1602 LCD 一块（带配套的 5k 微调电阻）；
- 9g 伺服电机一只；

- 8x8 LED 矩阵一个；
- 8 颗各色 LED（5mm）。

软件

- 编译软件；
- Fritzing。

三、实验要求

1. 画出你所实际实施的连接示意图；
2. 给出所用的器材的列表；
3. 用 Fritzing 画出外部设备的连线图，附实物照片；
4. 描述所做的实验步骤，给出各步操作的命令和结果；
5. 给出代码并解释；
6. 将所做作品拍摄视频上传到优酷，给出优酷的视频网址；
7. 说明其他所做的扩展内容的情况。

四、实验步骤

1. 前置工作：下载编译 ucos 文件：

   修改 makefile，将路径改为“./h”,编译器改为交叉编译器的路径，去掉某行“ucos_bcm2835.elf”后面的“.”：

   makefile ×

   ```
   ARMGNU ?= /usr/local/arm-2009q3/bin/arm-none-linux-gnueabi

   INCLUDEPATH ?= "./h"
   ```

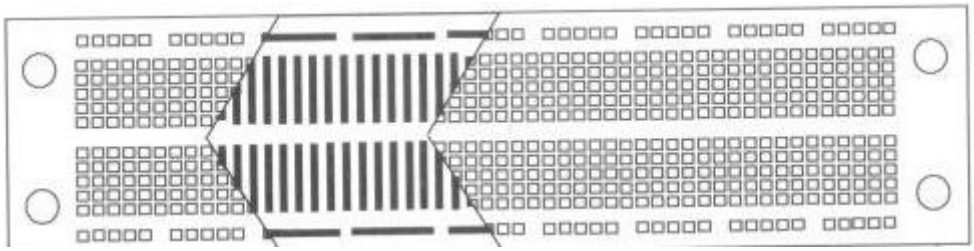
   还要将“/h/ucos”和“/ucos”下所有文件名改为小写的，以及将 init/startup.s 和 port/OS_Cpu_a.s 两个文件中的“//”都改为“@”。之后 make 成功：

```
cys@ubuntu:~/ucos_raspberryPi$ make clean
rm -f build/*.o
rm -f *.bin
rm -f *.hex
rm -f *.elf
rm -f *.list
rm -f *.img
rm -f build/*.bc
cys@ubuntu:~/ucos_raspberryPi$ make
/usr/local/arm-2009q3/bin/arm-none-linux-gnueabi-gcc -Wall -O2 -nostdlib -nostar
tfiles -ffreestanding -mcpu=arm1176jzf-s -I "./h" -D__ASSEMBLY__ -c -o build/sta
rtup.o init/startup.s
/usr/local/arm-2009q3/bin/arm-none-linux-gnueabi-gcc -Wall -O2 -nostdlib -nostar
tfiles -ffreestanding -mcpu=arm1176jzf-s -I "./h"  -c -o build/uart.o bsp/uart.c
bsp/uart.c: In function 'uart_send':
bsp/uart.c:23: warning: unused variable 'i'
/usr/local/arm-2009q3/bin/arm-none-linux-gnueabi-gcc -Wall -O2 -nostdlib -nostar
tfiles -ffreestanding -mcpu=arm1176jzf-s -I "./h"  -c -o build/timer.o bsp/timer
.c
/usr/local/arm-2009q3/bin/arm-none-linux-gnueabi-gcc -Wall -O2 -nostdlib -nostar
tfiles -ffreestanding -mcpu=arm1176jzf-s -I "./h"  -c -o build/interrupts.o bsp/
interrupts.c
```

2. 用到的电子元器件具体如下：

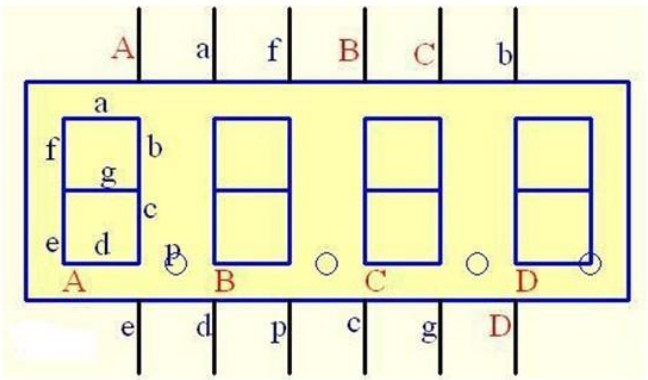
1） 面包板

面包板用于连线，内部结构如下：



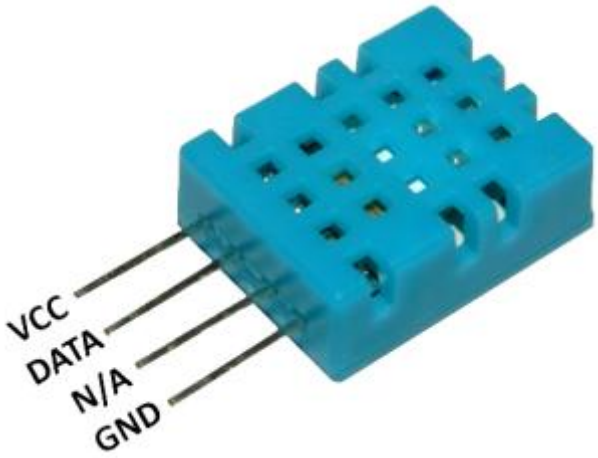
2） 四位七段数码管

四位数码管共 12 个引脚，功能如下：



3） DHT11 温湿度传感器

DHT11 是一款比较便宜的温湿度传感器模块。读取数据只需要占用一个 IO 口。能够同时测量温度和相对湿度。
其引脚和功能如下：



4) 开关

按钮总共有 4 个引脚，其中单侧的两个引脚在未按下的时候不通，当按钮按下的时候连通。而位于两侧的引脚则两两各自连通。

5) 面包线

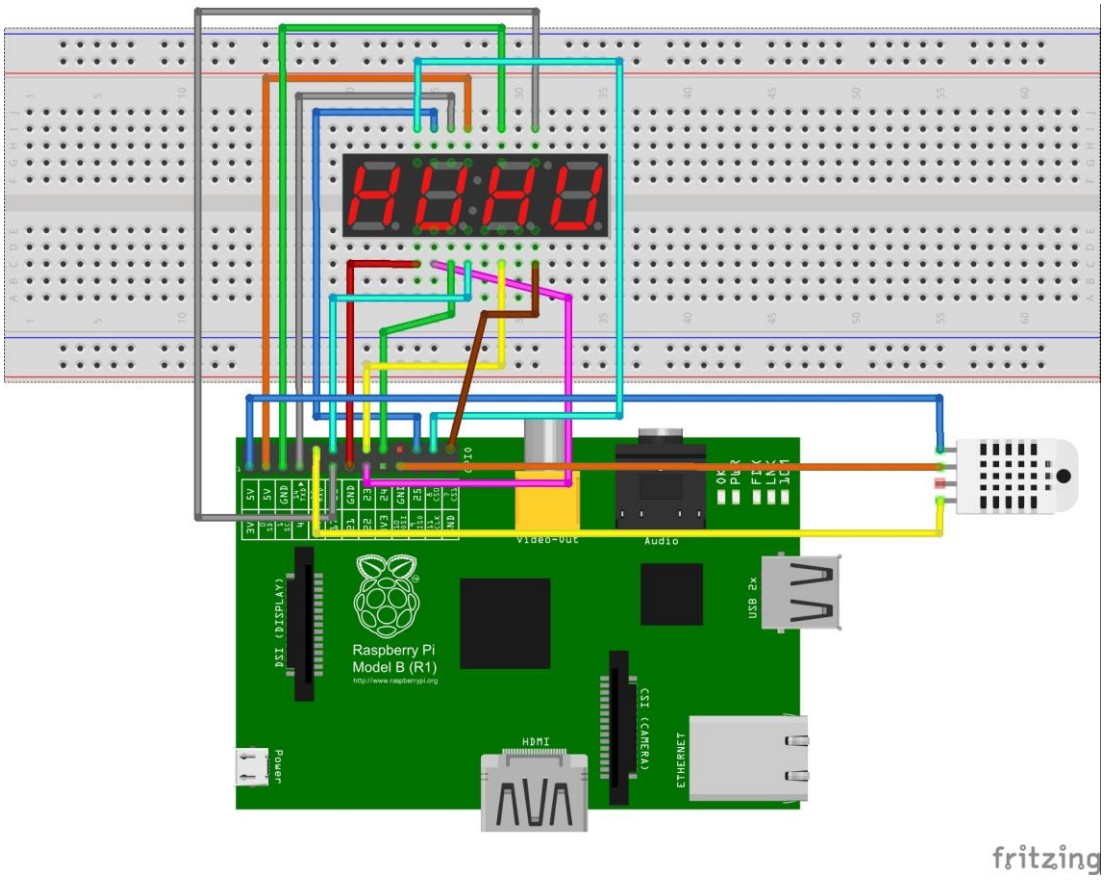
用于连接电路。

3. 设计输出方案，画连线示意图；

接口连接如下：

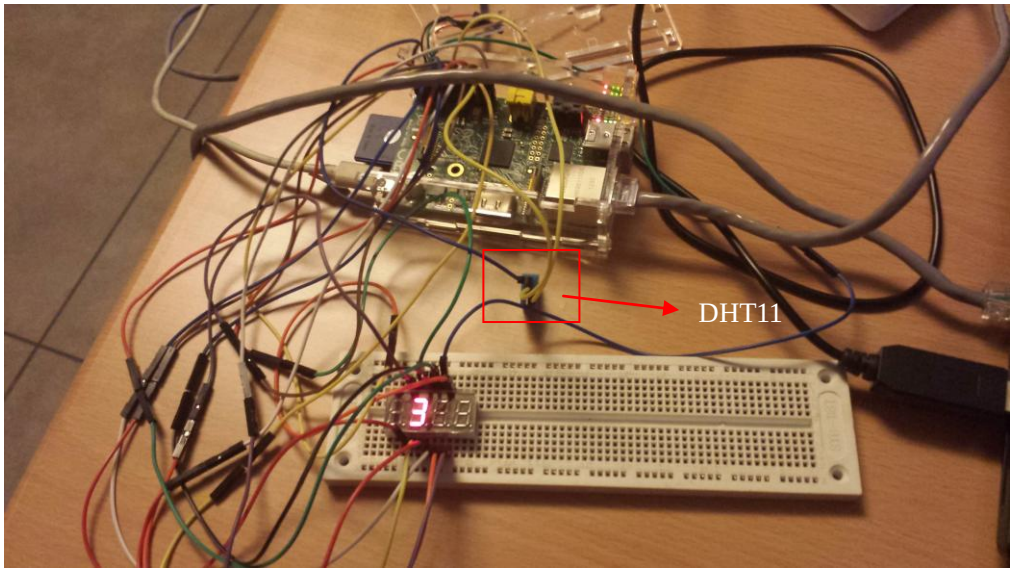
Pin	接口
0	LED b
1	LED c
2	LED e
3	LED d
4	LED g
5	LED dp
6	LED a
7	LED f
8	LED digit B
9	LED digit C
10	LED digit A
11	LED digit D
12	DHT11

连线示意图（DHT11 采用软件自带的白色温湿度计替代）：



4. 在面包板上连线，完成外部电路；

外部实物连线图如下：



5. 编写 C/C++程序读取 dht11 的数据；

程序如下：

```
#include <wiringPi.h>
```

```
#include <stdio.h>
#include <stdlib.h>

typedef unsigned char uint8;
typedef unsigned int  uint16;
typedef unsigned long uint32;

#define HIGH_TIME 32

int pinNumber =12;  //use gpio to read data
uint32 databuf;



uint8 readSensorData(void)
{
    uint8 crc;
    uint8 i;

    pinMode(pinNumber,OUTPUT); // set mode to output
    digitalWrite(pinNumber, 0); // output a high level
    delay(25);
    digitalWrite(pinNumber, 1); // output a low level
    pinMode(pinNumber, INPUT); // set mode to input
    pullUpDnControl(pinNumber,PUD_UP);

    delayMicroseconds(27);
    if(digitalRead(pinNumber)==0) //SENSOR ANS
       {
         while(!digitalRead(pinNumber)); //wait to high

     for(i=0;i<32;i++)
      {
      while(digitalRead(pinNumber)); //data clock start
      while(!digitalRead(pinNumber)); //data start
          delayMicroseconds(HIGH_TIME);
          databuf*=2;
           if(digitalRead(pinNumber)==1) //1
          {
                databuf++;
          }
       }

     for(i=0;i<8;i++)
```

```
       {
       while(digitalRead(pinNumber)); //data clock start
       while(!digitalRead(pinNumber)); //data start
           delayMicroseconds(HIGH_TIME);
           crc*=2;
           if(digitalRead(pinNumber)==1) //1
            {
                  crc++;
            }
        }
    return 1;
        }
   else
         {
         return 0;
          }
}

int main (void)
{

  printf("Use GPIO1 to read data!\n");

  if (-1 == wiringPiSetup()) {
    printf("Setup wiringPi failed!");
    return 1;
  }

  pinMode(pinNumber, OUTPUT); // set mode to output
  digitalWrite(pinNumber, 1); // output a high level

  printf("Enter OS-------\n");
  while(1) {
    pinMode(pinNumber,OUTPUT); // set mode to output
    digitalWrite(pinNumber, 1); // output a high level
    delay(3000);
    if(readSensorData())
    {
       printf("Congratulations ! Sensor data read ok!\n");
       printf("RH:%d.%d\n",(databuf>>24)&0xff,(databuf>>16)&0xff);
       printf("TMP:%d.%d\n",(databuf>>8)&0xff,databuf&0xff);
       databuf=0;
     }
    else
```

```
      {
         printf("Sorry! Sensor dosent ans!\n");
        databuf=0;
       }
  }
  return 0;
}
```

单独运行上述程序，得到下面的结果：

```
pi@raspberrypi:~$ gcc -o dht dht.c -lwiringPi -lpthread -lm
pi@raspberrypi:~$ sudo ./dht
Use GPIO1 to read data!
Enter OS-------
Sorry! Sensor dosent ans!
Congratulations ! Sensor data read ok!
RH:42.0
TMP:30.0
Congratulations ! Sensor data read ok!
RH:42.0
TMP:31.0
Congratulations ! Sensor data read ok!
RH:42.0
TMP:30.0
```

6. ucos 的调度程序

由于在 ucos 上要直接操纵 GPIO 接口，则定义变量如下：

```
#define BCM2708_PERI_BASE   0x20000000
#define GPIO_PADS     (BCM2708_PERI_BASE + 0x00100000)
#define CLOCK_BASE    (BCM2708_PERI_BASE + 0x00101000)
#define GPIO_BASE     (BCM2708_PERI_BASE + 0x00200000)
#define GPIO_TIMER    (BCM2708_PERI_BASE + 0x00000000)
#define GPIO_PWM      (BCM2708_PERI_BASE + 0x0020C000)
```

初始化接口：

```
void init(){
    int i = 0;
    unsigned int base = GET32(GPIO_BASE);
    set_output(0, &base);
    set_output(1, &base);
    set_output(2, &base);
    set_output(3, &base);
    set_output(4, &base);
    set_output(5, &base);
    set_output(6, &base);
    set_output(7, &base);
    set_output(8, &base);
```

```
    set_output(9, &base);
    set_output(10, &base);
    set_output(11, &base);

    set_input(DHT, &base);
}

void set_output(unsigned int p, unsigned int *base){
    p %= 10;
    *base = ((0xffffffff - (111<<(p*3)))&(*base))|(1<<(p*3));
}

void set_input(unsigned int p, unsigned int *base){
    p %= 10;
    *base = ((0xffffffff - (111<<(p*3)))&(*base));
}
```

显示程序：

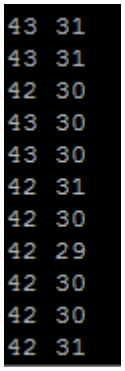
```
void led(){
    led_one(data, 0);
    led_one(data/10, 1);
    led_one(data/100, 2);
    led_one(data/1000, 3);
}

void led_one(int data, int count){
    int i;
    data %= 10;
    PUT32(GPIO_SET, 1<<pin[count]);
    for(i=0;i<8;i++){
        if((~digit[num])&seg[i]){
            PUT32(GPIO_CLR, i<<pin[i+4]);
            OSTimeDlyHMSM(0,0,0,1);
            PUT32(GPIO_SET, 1<<pin[i+4]);
        }
    }
    PUT32(GPIO_CLR, 1<<pin[row]);
}
```

将上述 DHT11 的程序改为直接操控 GPIO 的模式，之后打开 ucos 的 usrApp 文件夹下的 userApp.c 文件，将显示和读取分别粘贴到 userApp1 和 userApp2 程序中，然后重新编译运行。

7. 测试程序和电路

编译运行程序后，结果如下：



具体内容见视频：
http://v.youku.com/v_show/id_XMTI2NjcyNTM1Mg==.html