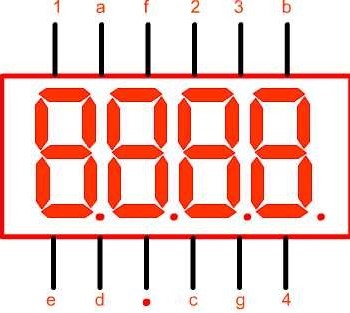
# 4位数码管

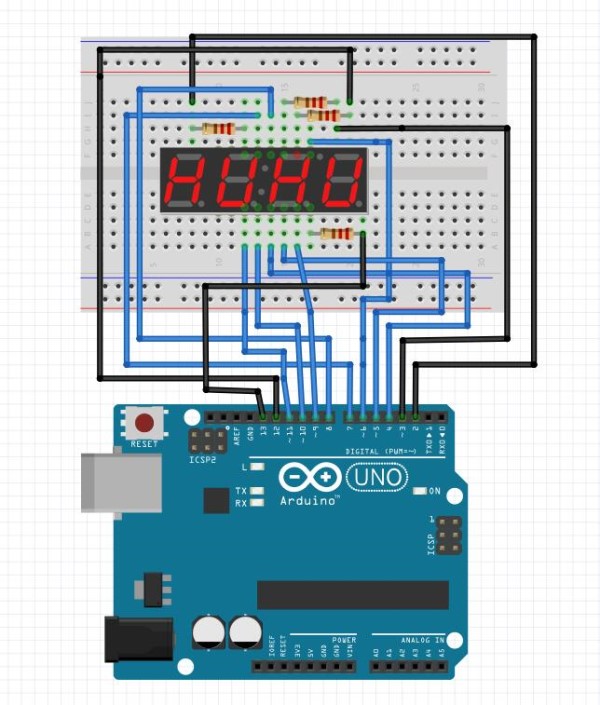
1位数码管需要8根线与单片机连接，如果是4个数码管也用同样的方式连接，则需要 32个端口，很显然UNO主板没有这么多端口，那么最终有什么方法节省端口呢？答案是“动态扫描显示”。

数码管根据位数不同，其封装的引脚也各不相同。但其内部都是将单个数码管的段选线a，b，c，d，e，f，g，dp对应连接在一起，公共极则相互独立。使用时分别通过控制不同的位选线（即单个数码管的公共极）来控制单个数码管的显示。以人眼难以分辨的速度进行刷新显示，即可达到多个数码管同时显示的效果。

四位数码管引脚分布如下图所示，其中1，2，3，4表示对应位数码管的公共极。



## 实物连接图



## 程序

int ledCount = 8;

int segCount = 4;

long previousMillis = 0;

//定义段码，这里是共阴段码，可以使用数码管段码软件改变数组值或者任意显示形状

const unsigned char DuanMa[10] = {0x3f, 0x06, 0x5b, 0x4f, 0x66, 0x6d, 0x7d, 0x07, 0x7f, 0x6f};

//位码

//unsigned char const dofly\_WeiMa[]={0,1,2,3};

int ledPins[] = {

12, 8, 5, 3, 2, 11, 6, 4,

}; // 11,7,4,2,1,10,5,3 注释是数码管实际引脚数，和芯片一样，逆时针数

int segPins[] = {

13, 10, 9, 7 //12,9,8,6 注释是数码管实际引脚数，和芯片一样，逆时针数

};

unsigned char displayTemp[4];//显示缓冲区

void setup() {

// 循环设置，把对应的端口都设置成输出

for (int thisLed = 0; thisLed < ledCount; thisLed++) {

pinMode(ledPins[thisLed], OUTPUT);

}

for (int thisSeg = 0; thisSeg < segCount; thisSeg++) {

pinMode(segPins[thisSeg], OUTPUT);

}

}

// 数据处理，把需要处理的byte数据写到对应的引脚端口。

void deal(unsigned char value) {

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

digitalWrite(ledPins[i], bitRead(value, i)); //使用了bitWrite函数，非常简单

// !bitRead(value,i)，这里前面加！(非运算符号)，取决于使用的是共阴还是共阳数码管。

}

// 主循环

void loop() {

static unsigned int num;//定义一个数据

static unsigned long lastTime = 0;

if (millis() - lastTime >= 1000) {

lastTime = millis();

//serialOutput();

num++;

}

displayTemp[0] = DuanMa[num / 1000]; //动态显示

displayTemp[1] = DuanMa[(num % 1000) / 100];

displayTemp[2] = DuanMa[((num % 1000) % 100) / 10];

displayTemp[3] = DuanMa[((num % 1000) % 100) % 10];

static int i;

unsigned long currentMillis = millis();

if (currentMillis - previousMillis > 0) {

previousMillis = currentMillis;

deal(0);// 清除“鬼影”

for (int a = 0; a < 4; a++) //循环写位码，任何时刻只有1位数码管选通，之前全部关闭，然后再选通需要的那位数码管

digitalWrite(segPins[a], 1); //

digitalWrite(segPins[i], 0); //

deal(displayTemp[i]);//读取对应的段码值

i++;

if (i == 4) //4位结束后重新循环

i = 0;

}

}