



## ramlife

博客园 :: 首页 :: 新随笔 :: 联系 :: 订阅 :: 管理

253 随笔 :: 110 文章 :: 13 评论 :: 44万 阅读

### 公告

昵称: ramlife   
园龄: 7年8个月  
粉丝: 12  
关注: 1  
[+加关注](#)

### 搜索

找找看

### 常用链接

我的随笔  
我的评论  
我的参与  
最新评论  
我的标签

### 我的标签

linux(85)  
C/C++(38)  
hardware(24)  
QT(21)  
android(20)  
version(19)  
vps(14)  
network(11)  
driver(11)  
win(10)  
[更多](#)

### 随笔分类

android(12)  
build(3)  
C/C++(29)  
deep\_learning(3)  
edit(9)  
ffmpeg(1)  
GTD(2)  
hardware(11)  
JAVA(4)  
linux(63)  
math(3)  
network(7)  
opencv(2)  
python(6)  
QT(18)  
[更多](#)

### 随笔档案

2023年12月(1)  
2023年2月(4)  
2023年1月(11)  
2022年12月(6)  
2022年11月(1)  
2022年10月(10)  
2022年9月(12)  
2022年8月(14)  
2021年3月(3)  
2021年2月(2)  
2021年1月(4)  
2020年12月(18)  
2020年11月(13)  
2020年10月(9)  
2020年9月(8)  
[更多](#)

### [转] AVR446步进电机算法推导及应用

转自: <https://blog.csdn.net/Renjiankun/article/details/80513839>

版权声明: 本文为博主原创文章, 遵循 CC 4.0 BY-SA 版权协议, 转载请附上原文出处链接和本声明。

本文链接: <https://blog.csdn.net/Renjiankun/article/details/80513839>

声明: 如有任何侵权问题请给我留言, 本博客文章均由我个人编写!

在学习步进电机控制过程中可谓困难重重, 资料零散, 或者说资料很难找, 所以我决定在这个博客这里整理目前网络上的步进电机算法, 并重新推导、理解他们, 且能够真正的应用起来, 并希望能够帮助大家, 学习还是自己的, 依靠自己的努力才能够学到手, 不要照搬即用, 应该去理解其中的奥秘。

这一章主要介绍AVR446的推导过程, 在很多资料里, 很少注重一个资料里的基础知识, 大部分人可能像我一样基础知识薄弱, 在推导过程中存在很多疑惑和难点, 无疑增加了学习的难度, 在我写的博客里, 对这个算法的推导之前会列出需要掌握的一些基础知识, 方便各位进行查阅推导, 当然我个人能力有限, 有些地方可能含糊不清, 希望各位大佬给予指正和指导。

#### 简介

AVR446文档是一篇关于如何通过实时计算实现步进电机梯形曲线加减速在低端MCU实现控制的一篇资料, 文档是英文的, 幸好大部分内容都比较简单, 而且有源码辅助理解。主要思想是加速运动曲线  $S = 1/2 * at^2$ , 但实际计算时它引入了开方计算, 这在低端MCU上是不允许的, 所以通过对开方计算使用泰勒公式的一种特殊形式-麦克劳林展开式, 来逼近开方计算, 使公式中不包含开方。控制方式上使用了状态机。

#### 基础知识及理论推导

在学习之前先回顾一下基础知识

##### 1.加速度公式及其相关公式

$$S = 1/2 * at^2、v = at等$$

##### 2.步进电机速度与定时器计数频率、定时器比较匹配值、步距角的关系

我在博文“步进电机S (Sigmoid) 型曲线加减速控制【查表法】”中已经推导了一次, 详情请各位去查阅。下面列出一些与AVR446文档要求的公式。

时间关系:

ft : 定时器计数频率

c : 比较匹配的计数值 (带下标c0、c1、cn.....代表每一步的比较匹配数值)

tt : 定时器计数周期  $tt = 1/ft$

$\delta t$ : 脉冲时间间隔

其中:  $\delta t = c * tt = c/ft$

步距角、角位移关系:

spr: 步进电机旋转一圈的总步数 如1.8°步距角下, 旋转一圈需要200步 (脉冲)

$\alpha$ : 弧度制的步距角  $\rightarrow \alpha = 2\pi / spr [rad]$

n: 步数 (脉冲数)

$\theta$ : 角位移, 转过多少弧度?  $\rightarrow \theta = n * \alpha [rad]$

$\omega$ : 角速度  $\rightarrow \omega = \alpha / \delta t$

$\omega'$ : 角加速度

其中:  $1rad \approx 9.55r/min$

由上面的关系我们就可以轻易推导出

$$\theta = n * \alpha = 1/2 * \omega' * t^2$$

则 n 个脉冲的角位移 需要的时间为:

$$tn = \sqrt{2n\alpha/\omega'} \quad \text{注意: } tn \text{ 的 } n \text{ 是下标;}$$

那么第n个脉冲的时间间隔就为:  $cn * tt = tn+1 - tn = \sqrt{2\alpha/\omega'} * (\sqrt{tn+1} - \sqrt{tn})$ : 自己算一下, cn的n是下标;

则 第n个脉冲的时候 比较匹配值  $cn = ft * \sqrt{2\alpha/\omega'} * (\sqrt{tn+1} - \sqrt{tn})$

特别的, 当  $n = 0$  时 c

$$\therefore cn = c0 * (\sqrt{tn+1} - \sqrt{tn})$$

## 文章分类

android(5)  
C/C++(15)  
crack(9)  
debug(6)  
edit(1)  
en(1)  
GTD(1)  
hardware(11)  
JAVA(4)  
jni(5)  
linux(24)  
network(3)  
politics(5)  
QT(5)  
version(4)  
更多

## 阅读排行榜

1. git 下载指定版本(14229)  
2. I2C 读取总是 0xFF, 但是 ACK 是正常的解决方法(10248)  
3. qt 字符串 转换 hex(10118)  
4. stm32cubeide 代码提醒(6000)  
5. core\_cm4.h(129): error: #35: # error directive: "Compiler generates FPU instructions for a device without an FPU (check \_\_FPU\_PRESENT)"(5639)

## 评论排行榜

1. tensorflow lite c++(3)  
2. stm32cubeide 代码提醒(2)  
3. android 学习笔记(2)  
4. ubuntu 20.04 安装 qtcreator(1)  
5. thunderbird 163 smtp(1)

## 推荐排行榜

1. ssh 关闭密码登录(1)  
2. MobileNetV2-SSDLite(1)  
3. markdown 锚点, 分页, 表格内换行(1)  
4. git 下载指定版本(1)  
5. android 学习笔记(1)

## 最新评论

1. Re:[转]测试近 20 个版本的迅雷之后, 告诉你哪个版本最好用

感谢分享, 高人健康快乐!

--瓦雷利亚钢

2. Re:ubuntu 20.04 安装 qtcreator  
install的s掉了: sudo apt  
update sudo apt install gcc  
g++ build-essential mesa-  
common-dev...

--KALI\_linux-chen

3. Re:[转] [Ozone] Ozone使用介绍-特殊功能

Ozone可以将某个反转的GPIO  
口像示波器或者keil的逻辑分析仪一样输出波形图吗

--杰瑞鼠

4. Re:tensorflow lite c++  
@ramlife 谢谢楼主, 已经找到原因了, gcc默认不会使用c的库, 加入-llibc就可以编译通过了。...

--wbn520

5. Re:tensorflow lite c++

这样我们就能计算出每一个脉冲的时间间隔和定时器比较匹配的值了。

为了消除开方计算, 我们使用泰勒展开式的一个特殊公式来逼近开方计算, (X四次方被认为是无穷小) 公式如下:

$$(1 \pm x)^{\frac{1}{2}} = 1 \pm \frac{x}{2} \mp \frac{x^2}{8} \pm \frac{x^3}{16} \mp O(x^4)$$

<https://blog.csdn.net/Renjiansun>

我们应该认识到, 本次得Cn值应该由上一次Cn值求出, 所以我们令cn / cn-1 得 (突然发现word可以编辑公式 2333.....)

$$\frac{c_n}{c_{n-1}} = \frac{c_0(\sqrt{n+1} - \sqrt{n})}{c_0(\sqrt{n} - \sqrt{n-1})}$$

然后将 根号n 提出来得, 值得注意的是, 在文档中认为 (1/n³) 已经算是高阶无穷小了, 因此将其省略掉, 其次是不引入更复杂的计算。

$$\frac{c_0\sqrt{n}\left(\sqrt{1+\frac{1}{n}}-1\right)}{c_0\sqrt{n}\left(1-\sqrt{1+\frac{1}{n}}\right)} = \frac{1+\frac{1}{2n}-\frac{1}{8n^2}+O\left(\frac{1}{n^3}\right)-1}{1-\left(1-\frac{1}{2n}-\frac{1}{8n^2}+O\left(\frac{1}{n^3}\right)\right)}$$

<https://blog.csdn.net/Renjiansun>

得出:

$$\frac{\frac{1}{2n}-\frac{1}{8n^2}}{\frac{1}{2n}+\frac{1}{8n^2}} = \frac{4n-1}{4n+1}$$

即:

$$C_n = C_{n-1} - \frac{2C_{n-1}}{4n+1}$$

到此, AVR446笔记中的加减速算法已经推导完毕, 接下来介绍如何应用到单片机中。

应用

在应用之前我们先想一想, 我控制一个步进电机加减速需要输入什么参数?, 有以下输入参数:

加速度、减速度、最大速度、运行的步数

第二, 电机将会工作在哪几种状态? 状态如下:

加速状态、加速到最大速度的匀速状态、减速状态、停止状态

同时, 我们应该注意到, 在我们输入的参数中, 是否能使电机加速到想要的速度呢? 所以实际上电机运行的情况还会存在以下几种情况: 1.电机能够加速到最大速度、2.电机不能够达到最大速度就应该减速 3.还有一种就是, 减速度输入过小导致的第二种情况。

所以我们需要预先计算出几个必须的点, 1. 第一个加速的点c0 2. 最大速度对应的比较匹配值 3. 设定的加速度到达最大速度所需要的步数 4. 当前设定加速度减速度下 必须减速的步数 5. 根据以上求出来的步数, 求出减速步数。

根据前面的知识我们可以很简单的计算出C0, 这里原文说是将会引入一个错误, 这个错误我没发现, 解决办法是将C0再乘以0.676..。我估计是如果直接使用C0可能导致启动不平滑, 贴出原文, 大家帮忙论证一下。以便日后修改这篇文章, 在此先感谢各位。(问题的原因已找到: 即在n=1, 时, 级数逼近的误差高达0.44, 为了减少这个误差, 并且使曲线上平滑, 需要补偿此误差, 解决办法就是C0\*0.6776)

This calculation is much faster than the double square root, but introduces an error of 0.44 at n=1. A more accurate formula is by multiplying c0 with 0.676.

所以, C0 = ft \* sqrt (2\*a/w) \* 0.676

@wbn520 没有碰到类似的问题。。。看你这个报错，链接的时候，没有找到 new, delete, throw 等相关的函数，这个就有点不对劲了。你看看是不是你的 cmake 里面的路径没有设置...

--ramlife

最大速度是匹配值  $C_{maxspeed} = \alpha * ft / \omega$  (由公式:  $\omega = \alpha / \delta t$ ,  $\delta t = cn * tt$ )

达到最大速度需要的加速步数  $N_{max} = \omega^2 / 2\alpha'$  (由公式:  $tn = \omega n / \omega'$ 、 $2n\alpha = \omega' tn^2$  导出, 其中  $t$ 、 $\omega$  字母后的  $n$  是下标)

计算加速步数, 在计算之前先看看加速减速步数与总步数之间的关系, 由达到最大速度需要步数的推导公式很容易求得

$n\omega' = \omega^2 / 2\alpha'$ , 其中角速度和步距角都是已知的, 所以得出两个不同加速度之间的关系  $n1\omega'1 = n2\omega'2$ , 这说明了加速度和步数所对应的关系, 为了能够得到加速步数与总步数的关系, 我们在等式两边各加上  $n1\omega'2$ , 由此可以导出以下公式:

$n1 = (n1 + n2) * \omega'2 / (\omega'1 + \omega'2)$  这个结果将用于与达到最大速度的  $N_{max}$  进行比较, 检查是否能够满足电机达到最大速度, 否则将减速。

若  $N_{max} < n1$  则表明可以达到最大速度, 则对应的加速步数为  $N_{max}$ , 减速步数可以通过上面求出的关系式轻易得出

减速步数  $N_{decel} = N_{max} * (\omega'1 / \omega'2)$

若  $N_{max} > n1$  则表明无法达到最大速度, 并且在必须减速的点开始减速, 则对应的加速步数为  $n1$ , 减速步数使用总步数减去  $n1$  即可, 即  $N_{decel} = Steps - n1$ .

这样, 将初始化参数全部求出后, 将  $C0$  赋值给比较匹配寄存器, 进入到对应的状态即可, 在加速减速状态中调用公式

$$C_n = C_{n-1} - \frac{2C_{n-1}}{4n + 1}.$$

即可完成加速减速过程, 并且值得注意的是, 这里边含有除法, 在 MCU 计算中意味着存在余数的误差, 为了提高计算精度, 文档中对程序中每一次除法都将余数求出来, 并且加到下一次计算之中。详情请看相关代码。

程序实现

本设计还是采用 Arduino Uno (ATmega328) + 步进电机驱动器 + 42 步进电机实现

其中 步进电机为细分, 意味着  $spr$  是 1600。定时器计数频率为 250Khz

需要注意的是, AVR446 例程中所使用的单位是弧度制, 并且为了便于单片机计算都将数据放大了 100 倍处理, 相当于输入的参数也需要放大 100 倍, 比如输入加速度为 1000, 则实际为  $10 \text{ rad/s}^2$ 。

```
/*电机参数结构体*/
typedef struct {
    unsigned char run_state : 3; //运行状态
    unsigned char dir : 1; //方向
    unsigned int step_delay; //每一步的时间间隔 (匹配值)
    unsigned int decel_start; //必须开始减速的步数
    signed int decel_val; //减速步数
    signed int min_delay; //最大速度时比较匹配值
    signed int accel_count; //加速计数器
} speedRampData;
speedRampData srd; //定义

#define T1_FREQ 250000 //定时器频率250khz

//! 一圈的步数 4 细分
#define SPR 800

#define ALPHA (2*3.14159/SPR) // 步距角
#define A_T_x100 ((long)(ALPHA*T1_FREQ*100)) // 用于计算最大速度时的匹配值 步距角*脉冲数*100
#define T1_FREQ_148 ((int)((T1_FREQ*0.676)/100)) // 用于计算首脉冲 C0
#define A_SQ (long)(ALPHA*2*10000000000) // 用于计算首脉冲 C0
#define A_x20000 (int)(ALPHA*20000) // 用于计算最大步数需要的加速步数Nmax

// 电机状态
#define STOP 0
#define ACCEL 1
#define DECEL 2
#define RUN 3

void speed_ctr_Move(int steps, unsigned int accel, unsigned int decel, unsigned int speeds); //
void speed_ctr_Init_Timer1(void); //定时器初始化
void OneStep(uint8_t dir); //电机根据方向运行一步
/*初始化*/
void setup() {
    pinMode(2, OUTPUT); //此引脚输出方向
    pinMode(3, OUTPUT); //此引脚输出脉冲
    speed_ctr_Init_Timer1();
    Serial.begin(115200);

    speed_ctr_Move(-20000, 9000, 9000, 9000);
}
/*主循环*/
void loop() {
```

```
}
/*定时器1 OCR1A 比较匹配中断 CTC模式*/
ISR(TIMER1_COMPA_vect)
{
    unsigned int new_step_delay; //新的匹配值
    static int last_accel_delay; //
    static unsigned int step_count = 0; //步数计数
    static unsigned int rest = 0; //余数

    OCR1A = srd.step_delay; //OCR1A 重新赋值

    switch(srd.run_state) {
        case STOP:
            step_count = 0;
            rest = 0;
            TCCR1B &= ~(1<<CS12)|(1<<CS11)|(1<<CS10));
            break;

        case ACCEL:
            OneStep(srd.dir); //运行一步
            step_count++;
            srd.accel_count++;
            new_step_delay = srd.step_delay - (((2 * (long)srd.step_delay) + rest)/(4 * srd.accel_count));
            rest = ((2 * (long)srd.step_delay) + rest) % (4 * srd.accel_count + 1); //求出余数
            if(step_count >= srd.decel_start) { //判断是否进入减速
                srd.accel_count = srd.decel_val;
                srd.run_state = DECEL;
            }
            else if(new_step_delay <= srd.min_delay) { //判断是否能进入最大速度
                last_accel_delay = new_step_delay;
                new_step_delay = srd.min_delay;
                rest = 0;
                srd.run_state = RUN;
            }
            //Serial.println("ACC");
            break;

        case RUN:
            OneStep(srd.dir); //运行一步
            step_count++;
            new_step_delay = srd.min_delay;
            if(step_count >= srd.decel_start) { //判断是否要减速了
                srd.accel_count = srd.decel_val;
                new_step_delay = last_accel_delay;
                srd.run_state = DECEL;
            }
            break;

        case DECEL:
            OneStep(srd.dir);
            step_count++;
            srd.accel_count++;
            new_step_delay = srd.step_delay - (((2 * (long)srd.step_delay) + rest)/(4 * srd.accel_count));
            rest = ((2 * (long)srd.step_delay) + rest) % (4 * srd.accel_count + 1);
            if(srd.accel_count >= 0) { //步数走完了吗?
                srd.run_state = STOP;
            }
            break;
    }
    srd.step_delay = new_step_delay; //获取新的一次的匹配值
}

void OneStep(uint8_t dir)
{
    digitalWrite(2, dir);

    digitalWrite(3, 1); //输出一个脉冲
    digitalWrite(3, 0);
}

void speed_cntr_Move(int steps, unsigned int accel, unsigned int decel, unsigned int speeds)
{
    unsigned long max_s_lim;
    unsigned int accel_lim;

    if(steps < 0){
        srd.dir = 0; // 设置运动方向
        steps = -steps;
    }
    else{
        srd.dir = 1; //设置方向
    }
}
```

```
}

if(steps == 1){ //只有1步时如何处理?
    srd.accel_count = -1;
    srd.run_state = DECEL;
    srd.step_delay = 1000;
    OCR1A = 10; //使其进入中断
    TCCR1B |= ((0<<CS12)|(1<<CS11)|(1<<CS10));
}
else if(steps != 0){
    srd.min_delay = A_T_x100 / speeds;//计算最大速度，脉冲的时间间隔
    srd.step_delay = (T1_FREQ_148 * sqrt(A_SQ / accel))/100;//计算C0 加速开始的第一段 脉冲时间间隔
    max_s_lim = (long)speeds*speeds/(long)((((long)A_x20000*accel)/100);//根据给定速度和加速度，需要
    /*0的情况*/
    if(max_s_lim == 0){
        max_s_lim = 1;
    }

    accel_lim = ((long)steps*decel) / (accel+decel);//加速度段n1

    if(accel_lim == 0){
        accel_lim = 1;
    }

    // 可以达到最大速度
    if(accel_lim <= max_s_lim){
        srd.decel_val = accel_lim - steps;//无法达到最大速度的减速步数
    }
    else{
        srd.decel_val = -((long)max_s_lim*accel)/decel;//能达到最大速度的减速步数
    }

    /*0的情况 必须从减速到停止
    if(srd.decel_val == 0){
        srd.decel_val = -1;
    }

    // 找到何时开始减速的步数
    srd.decel_start = steps + srd.decel_val;
    //速度太低了，直接运行
    if(srd.step_delay <= srd.min_delay){
        srd.step_delay = srd.min_delay;
        srd.run_state = RUN;
    }
    else{
        srd.run_state = ACCEL; //否则从加速开始运行
    }

    // 复位计数器
    srd.accel_count = 0;
    OCR1A = 10;
    // 设定定时器分频
    TCCR1B |= ((0<<CS12)|(1<<CS11)|(1<<CS10));
}
}
/*250Khz 并开启比较匹配中断*/
void speed_cntr_Init_Timer1(void)
{
    TCCR1A = 0;
    TCCR1B = (1<<WGM12)|(0<<CS12)|(1<<CS11)|(1<<CS10);
    TIMSK1 = (1<<OCIE1A);
    OCR1A = 15;
    sei();
    srd.run_state = STOP;
    delay(1);
}
```

经过测试，电机能够时间T型加减速，并且不会产生速度突变，运用得当简直是居家必备之良品啊！哈哈。

至此，AVR446的学习就告一段落了，如有疑问请给我留言，我会及时处理。

20190125更新：在几个月前，我已把AVR446更进一步改进了，新的算法包含速度模式和位置模式，位置模式与AVR446本身一致，只是单位变成了rpm，速度模式是我个人新增的，包含速度模式启动，速度模式中任意变速，以及任意时刻减速到停止的功能，有时间也更新一下吧。另，本人已经用FPGA实现了8步进独立控制（包含位置、速度模式），使用STM32的FSMC控制。

AVR446资料地址：h

n

iankun/10455475



版权声明：本文为CSDN博主「Renjiankun」的原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接及本声

明。  
原文链接: <https://blog.csdn.net/Renjiankun/article/details/80513839>

分类: C/C++

标签: driver

好文要顶 关注我 收藏该文 微信分享

 ramlife   
粉丝 - 12 关注 - 1  
会员号: 2432 (终身会员PLUS)  
[+加关注](#)

0 0

[升级成为会员](#)

posted on 2019-11-28 00:48 ramlife 阅读(1038) 评论(0) 编辑 收藏 举报  
会员力量, 点亮园子希望

[刷新页面](#) [返回顶部](#)

登录后才能查看或发表评论, 立即 [登录](#) 或者 [逛逛](#) 博客园首页

【推荐】秋天希望的田野, 九月最后的救园: 终身会员计划  
【推荐】轻量又高性能的 SSH 工具 IShell: AI 加持, 快人一步  
【推荐】100%开源! 大型工业跨平台软件C++源码提供, 建模, 组态!  
【推荐】2024阿里云超值优品季, 精心为您准备的上云首选必备产品



#### 编辑推荐:

- [redisson 内存泄漏问题排查](#)
- [使用.NET并行任务库\(TPL\)与并行Linq\(PLINQ\)充分利用多核性能](#)
- [Redis 内存突增时, 如何定量分析其内存使用情况](#)
- [记一次 RabbitMQ 消费者莫名消失问题的排查](#)
- [C#.net core 基础 - 深拷贝的五大类N种实现方式](#)

#### 阅读排行:

- [裁员, 这一次终于轮到了我](#)
- [作为博主和曾经员工, 谈谈近期的园子](#)
- [博客园终身会员小福利, 送华为云服务器](#)
- [救园倒计时: 救园最后4天](#)
- [.NET 工具库高效生成 PDF 文档](#)