

**目录**

第一章复杂工程问题归纳与实施方案可行性研究.................................................. 1

1.1 需求分析与建模..............................................................................................

1.2 复杂工程问题归纳..........................................................................................

1.3 实施方案与可行性研究..................................................................................

1.4 最终版本实现方案.................................................................................

第二章 存在问题 解决方案和版本迭代优化...........................................................

2.1 初代版本V1.0.0..............................................................................................

2.2 升级版本 V1.1.0 到V1.3.0...............................................................................

2.3 终版 V1.4.2............................................................................ ...................

第三章 执行情况与完成度分析..........................................................................................

第四章 分工协作与交流情况......................................................................................

参考文献........................................................................................................................

致谢................................................................................................................................

#### 摘要：

本综合设计以MSP430G2553-Lauchpad开发板为基础，通过深入了解MSP430G2553芯片的硬件结构和软件开发环境，致力于实现一套完整的嵌入式系统。通过Git进行版本控制，利用Code Composer Studio (CCS)进行嵌入式软件开发，集成了多个关键技术点，包括GPIO控制、串口通信、中断响应机制、ADC数模转换、Flash存储等。通过这些技术的融合，实现了将温度或电压数据通过串口传输到PC主机并进行图形化显示或曲线绘制。在高级版本中，进一步考虑实现自动持续运行，并优化数据采集频率，以增强每条数据的时间密集程度，使得曲线更加精细和连续。此外，通过Python的Serial和Matplotlib库，实现了在PC主机上对温度电压数据的图形绘制。最后，使用VoFA+等调试工具直观显示温度和电压波形，提高系统的调试效率。

#### 关键词：

MSP430G2553、Git、Python、Code Composer Studio（CCS）、GPIO控制、串口通信、中断响应机制、ADC数模转换、Flash存储、自动运行、数据采集频率、图形绘制、VoFA。

**第一章复杂工程问题归纳与实施方案**

1.  **需求分析与建模**

**场景描述**

在本项目中，我们着眼于利用 MSP430G2553 Lauchpad 在 TI 公司提供的 CCS 环境下，实现一个温度和电压传感器，同时实时在 PC 机上显示图像波形。为了明确项目的目标，我们进行了如下需求分析：

1. **功能需求：**

* 实现温度和电压传感器数据的准确采集。
* 实时在 PC 机上显示温度和电压的图像波形。

2. **性能需求：**

* 数据采集的响应时间应在合理范围内。
* 系统应具备良好的稳定性，以确保长时间运行的可靠性。

3. **非功能性需求：**

* 系统应具备良好的可维护性和可扩展性。
* 考虑数据传输的安全性。

2.  **复杂工程问题归纳**

第一，利用ccs实现的电压与温度采集，其采样率与采样速率如何提高？能否在例程代码的基础上予以改进？

第二，即利用PC画图的功能：利用什么工具进行画图，画图的标准以及方式是什么？

第三，如何实现闪存的功能？闪存功能为我们提供了哪些

3.  **实施方案与可行性研究**

实施方案：

1. 系统初始化:初始化系统时钟，配置UART通信，设置ADC的参考电压和采样时钟。
2. ADC模式切换：通过按键中断(P1.3)触发，切换ADC模式，包括电压模式、温度模式和F调试模式。
3. 数据采集：根据选择的ADC模式，进行相应的数据采集，包括获取电压数据或温度数据。
4. 数据处理： 对获取到的数据进行处理，转换为真实物理量的电压/温度数值。
5. Flash操作：在Flash操作下，实现数据的存储与读取功能，
6. 数据传输：通过UART串口通信，将数据传输至PC主机。
7. 数据绘图: 通过python的serial库读取串口数据,实现实时绘图功能,或通过vofa+调试工具,把数据转化为波形图

#### 可行性研究

* ADC模块：MSP430G2553-Lauchpad搭载了ADC模块，支持多通道的模拟信号采集。通过ADC的初始化和配置，可以实现对温度和电压的准确采集。

关键考虑点：精度和分辨率：采样频率：

* 中断机制：MSP430G2553-Lauchpad支持中断机制，通过中断可以实现对按键的响应和ADC模式的切换。中断机制能够提高系统的实时性和响应速度。

关键考虑点：中断优先级： 中断响应时间

* CCS使用技巧： Code Composer Studio (CCS)是一款功能强大的集成开发环境，适用于MSP430系列。通过CCS，可以进行编码、编译、调试等操作，提高开发效率。

关键考虑点：工程管理： 确保CCS中的工程管理合理，方便项目的组织和维护。

* 数据传输和UART通信： 使用UART通信进行数据传输，MSP430G2553-Lauchpad提供了相应的硬件支持。UART通信是一种可靠的串口通信方式。

关键考虑点：波特率选择 数据格式

* Flash存储： MSP430G2553-Lauchpad内置Flash存储器，可以用于数据的持久性存储。Flash存储的特性保证了数据的长期保存。

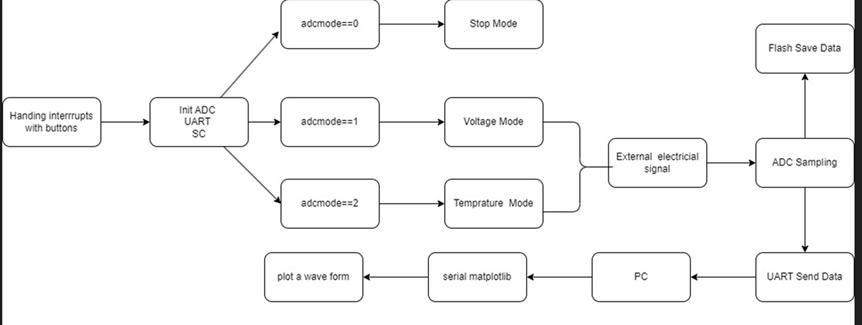
关键考虑点：写入次数： 数据擦除- 一定要确保Flash读写操作的合理性和安全性。

* **版本控制与项目管理（Git和GitHub）：**
* **可行性：** 引入Git作为版本控制系统，结合GitHub进行项目管理，有助于多人协作、版本追踪和错误管理。

**关键考虑点：分支管理 版本追踪 协作模式**

* 总体可行性概括
  + 团队成员经过培训能够熟练使用CCS进行嵌入式开发。
  + 所需硬件MSP430G2553 Launchpad及相关设备易获得。
  + 所涉及的知识点均为嵌入式系统基础，成员通过培训和学习能够掌握。
  + 使用Python进行图形绘制和VoFA+等调试工具的应用相对简便，易于上手。
  + Git与GitHub的整合提高团队协作效率。
  + 综上所述，项目技术和团队基础可行，有望顺利完成。

**4.最终版本实现情况(V1.4.2)**



程序首先进行系统初始化，包括初始化系统时钟，配置UART通信，以及设置ADC的参考电压和采样时钟。接着，程序初始化Flash，读取其中已存储的数据，用于后续的数据存储。

随后，程序进入主循环，不断执行以下步骤：

读取按键中断(P1.3)状态，当检测到按键中断时，切换ADC模式。ADC模式包括电压模式、温度模式和调试模式。根据选择的ADC模式，进行相应的数据采集和处理：

在电压采集模式下，选择ADC通道为A0，获取ADC数值，转换为相应的电压值，存入Flash空间 存入指针到末尾时通过UART发送至PC。

在温度采集模式下，选择ADC通道为A10，获取ADC数值，转换为相应的温度值，存入Flash空间 存入指针到末尾时通过UART发送至PC。

在Flash操作模式下，通过UART接收命令，实现Flash数据的命令擦除和命令读取。0xAA->擦除全部 0x55 获得当前flash空间中的全部数据

在数据处理过程中，采集到的数据每2位为1个有效数据保存至Flash中,支持最多64次数据存储。

通过UART串口通信，将采集到的数据传输至PC主机。程序提供了发送字符串、发送数字和发送浮点数的函数，用于将数据以ASCII格式发送。

整个程序流程是一个循环的过程，根据按键中断的触发，切换不同的ADC模式，进行相应的数据采集和处理，并通过UART将数据发送至PC。这样的设计使得程序在不同模式下能够适应电压和温度的采集，并且具备了对Flash进行数据存储和读取的功能。

**第二章存在问题 解决方案和版本迭代优化**

* 正式开始前

去哪里寻找和获得学习资源

人员该如何组织架构

谁来充当组长

具体项目开始时所有的文件代码还有工作区应该如何划分以及同步

* 时间安排： 制定项目的时间计划，包括各阶段的截止日期和整体项目周期。
* 任务分解： 将整体项目划分为小任务，分解到具体的团队成员，确保每个任务有清晰的执行计划。
* 硬件资源和准备
* 硬件获取： 确保所有团队成员都能够获得需要的硬件设备（msp430g2553 Lauchpad等）。
* 环境设置： 确认每个团队成员都正确设置了开发环境（CCS等）。
* 团队协作
* 沟通计划： 制定团队内部的沟通计划，明确团队会议的频率、沟通工具的选择等。
* 问题解决： 设立问题解决的渠道和流程，确保在团队协作中及时解决问题。
* 团队协作改进：我们致力于加强团队内部的协作与沟通。为此，我们将采用项目管理工具，以更高效地分配任务和跟踪项目进度。此外，我们会定期召开会议，促进团队成员间的交流，分享信息和经验，确保团队工作能更有序地进行。

**初代版本V1.0.0版本**

****

在这个版本当中我们实现了串口读取温度数据的功能 Tget 同时另一个CCS Project Vget 实现了串口读取温度的功能

二者在代码层面上相似 共同的函数声明如下

1. 系统初始化函数 (InitSystemClock):

* 设置DCO(Digital Controlled Oscillator)为1MHz，选择BCS(Basic Clock System)的时钟源为DCO，同时清除SMCLK的源位。
* 关闭SMCLK的分频，使之等于MCLK。

1. UART初始化函数 (InitUART):

* 对UART进行初始化，首先置位UCSWRST（UART控制寄存器1的软件复位位）。
* 配置UCA0为UART模式，选择UART时钟源为SMCLK。
* 设置波特率为9600，计算波特率设置寄存器的值。
* 配置P1.1和P1.2为UART的TX和RX引脚。
* 清除UCSWRST，使UART模块处于正常工作状态。

1. UART发送字符串函数 (UARTSendString):

* 通过UART发送指定长度的字符数组。

1. 打印数字函数 (PrintNumber):

* 将16位无符号整数以字符串形式发送到UART。

1. 打印浮点数函数 (PrintFloat):

* 将浮点数以字符串形式发送到UART，通过将浮点数乘以1000并转换为整数，然后按位提取。

1. ADC初始化函数 (InitADC):

* 配置ADC模块，选择ADC时钟源为MCLK，设置ADC时钟分频为1。
* 设置参考电压为1.5V，采样保持时间为64CLK，关闭ADC中断。
* 启用内部参考电压2.5V，并配置ADC通道为A0。

1. 获取ADC值函数 (GetADCValue):

* 启动ADC转换，等待转换完成，然后返回ADC值。

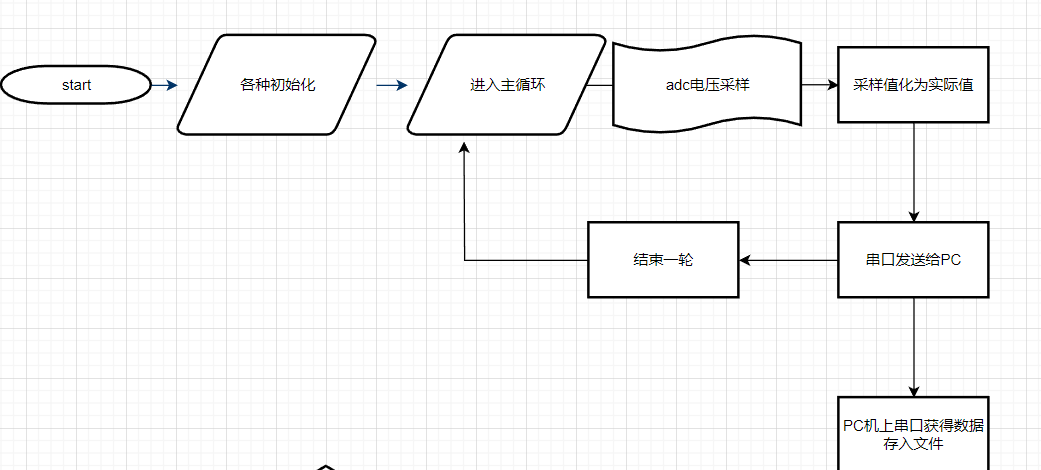
1. Tget项目:

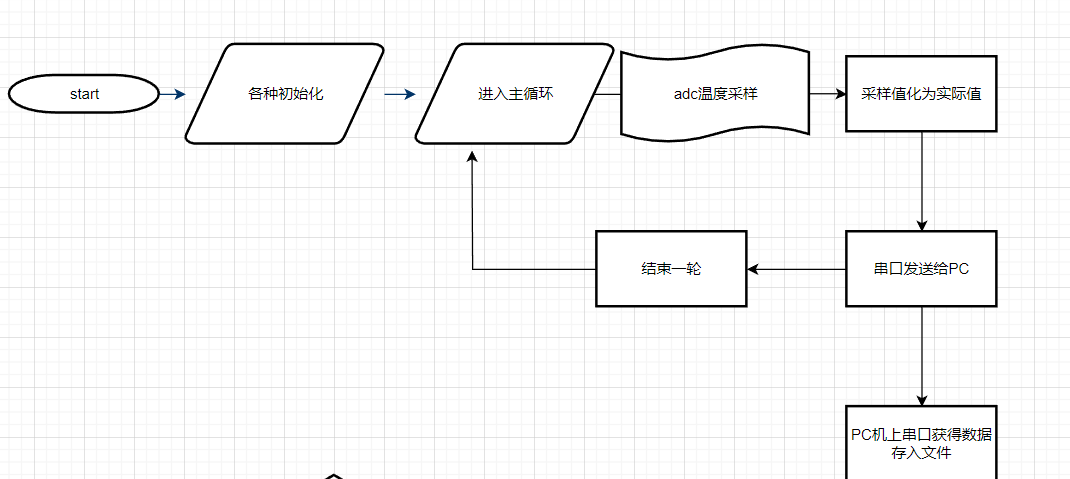
* Tget主要实现了通过ADC获取温度数据，并通过UART发送到终端。
* 在主函数中，循环中获取ADC值，然后使用特定的缩放公式计算温度，并通过UART发送到终端。
* 温度的ADC-真值转换公式为 (adcvalue \* 0.01) - 50.0。

1. Vget项目:

* Vget主要实现了通过ADC获取电压数据，并通过UART发送到终端。
* 在主函数中，循环中获取ADC值，然后使用特定的缩放公式计算电压，并通过UART发送到终端。
* 电压的ADC-真值转换公式为 adcvalue \* 2.5 / 1023。

实现的具体流程活动图如下(1图电压 二图温度)





画图

使用sscom5.13.1.exe串口调试助手 把对应的获取到的数据存储在相应的'test.txt'中 截图中所对应的python文件

的实现和结构大致相同 主要目的是从文本文件中提取电压/温度数据，并使用matplotlib库创建一个电压/温度值的折线图。以下是对脚本各部分的简要解释：

1. 读取文本文件:

* 使用open函数打开名为'test.txt'的文本文件，读取所有行，并将它们存储在名为lines的列表中。

|  |
| --- |
| Python  with open('test.txt', 'r', encoding='utf-8') as file:  lines = file.readlines() |

1. 提取电压值:

* 使用正则表达式（re.search）从每一行中提取相应电压值，将其转换为浮点数，并将它们存储在名为voltage\_values的列表中。

|  |
| --- |
| Python  voltage\_values = [float(re.search(r'相应值为：([\d.]+)', line).group(1)) for line in lines if re.search(r'相应电压值为：([\d.]+)', line)] |

1. 创建图表:

* 使用plt.subplots创建一个包含图表对象和坐标轴的图表。

|  |
| --- |
| Python  fig, ax = plt.subplots(figsize=(len(voltage\_values), 5)) |

1. 绘制电压值折线图:

* 使用ax.plot函数在图表上绘制电压值的折线图，其中range(len(voltage\_values))表示X轴值，voltage\_values表示Y轴值，marker='o'表示使用圆圈标记。

|  |
| --- |
| Python  ax.plot(range(len(voltage\_values)), voltage\_values, marker='o') |

1. 设置Y轴范围和刻度:

* 使用ax.set\_ylim设置Y轴范围，ax.set\_yticks和ax.set\_yticklabels设置Y轴的刻度和标签。

|  |
| --- |
| Python  ax.set\_ylim([1.10, 1.25]) ax.set\_yticks([i / 100 for i in range(110, 125, 1)]) ax.set\_yticklabels([f"{i:.2f}V" for i in range(110, 125, 1)]) |

1. 设置X轴标签:

* 使用ax.set\_xticks和ax.set\_xticklabels设置X轴的刻度和标签，其中rotation=45表示标签以45度旋转。

|  |
| --- |
| Python  ax.set\_xticks(range(0, len(voltage\_values), 10)) ax.set\_xticklabels([f"[{i}]" for i in range(0, len(voltage\_values), 10)], rotation=45) |

1. 设置图表标题:

* 使用ax.set\_title设置图表标题。

|  |
| --- |
| Python  ax.set\_title('AHUA\_V\_PICT') ax.set\_title('AHUA\_T\_PICT') |

1. 保存图表为PNG文件:

* 使用plt.savefig保存图表为名为'voltage\_plot.png/t-plot.png'的PNG文件，dpi=500表示设置图像的每英寸点数。

|  |
| --- |
| Python  plt.savefig('voltage\_plot.png', dpi=500) |

1. 显示图表:

* 使用plt.show()显示图表。

|  |
| --- |
| Python pythonCopy code plt.show() |

这个脚本的最终结果是创建一个电压值的折线图，并将其保存为PNG文件，同时在运行脚本后显示图表。

其中电压的范围是1.0到1.3V 温度的范围是10到30度

**1到99的阶段(V1.0.0到V.4.2的更新)**

从1 到99 在这个过程中间 我们往单片机里面增加中断来实现按键控制功能 增加了falsh存储 增加了串口命令 增加了timerA定时器模块 增加了低功耗模式 而我们每次功能的添加都是为了解决一些难受的问题和进行优化

#### V1.1.0

**如何把串口读取温度还有串口读取电压结合起来**

通过按键控制引发P1.3口电位变化触发中断,使得模式选择计数器adcMode 在0 1 2三种状态遍历循环

0 电压收集模式

1 温度采集模式

2 暂停采样模式

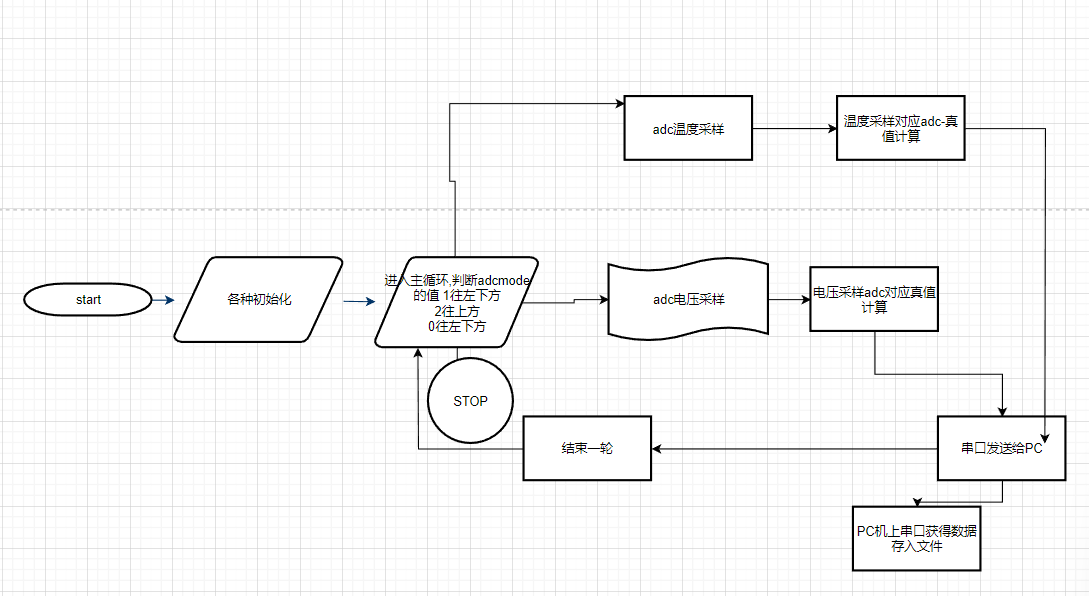
1 2 情况下通过ADC数模转换的数据(原始数据1)会直接通过板子上的CPU运算为浮点数

中断设计如下

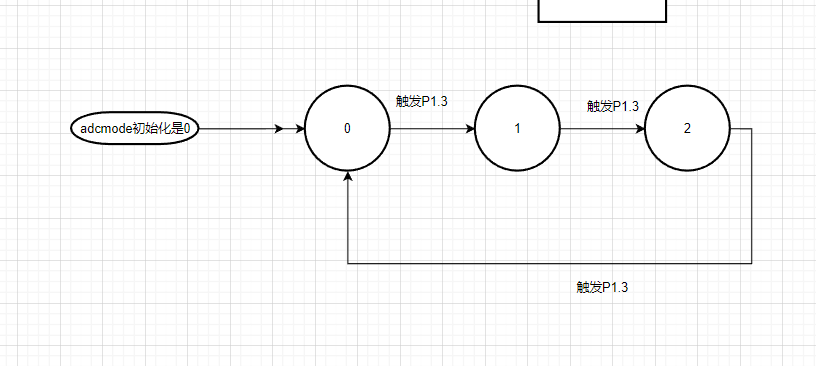
|  |
| --- |
| #pragma vector = PORT1\_VECTOR \_\_interrupt void Port1\_ISR(void)  { if(P1IFG & BIT3) //  {\_\_bic\_SR\_register(GIE); //  P1OUT ^= BIT6;  P1IFG &= ~ BIT3; //  adcMode++;  if (adcMode > 2)  { adcMode = 0;}\_\_bis\_SR\_register(GIE); }} |

主函数逻辑增加了条件判断 每次进入循环前 先判断adcmode(0 1 2 )的值 进入对应的采集模式

新的V.1.1.0版本的活动流程图如下所示



响应式触发中断所对应的adc采集模式变量adcmode的生命图如下所示



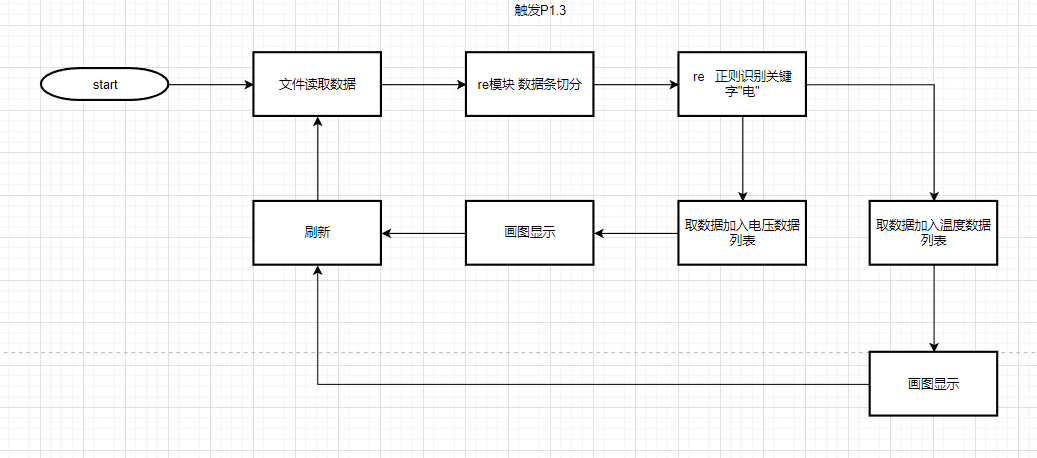
**如何绘图使得可以让温度 电压数据 在一次绘制,而不用停机再运行程序**

**更新后的程序发送的数据既包含温度 也包含电压 应该怎么做才可以识别两种数据,同时画两种图呢**

在发送的数据里面带上"电" 字 就是这一条数据对应的是电压数据 那么其他的数据条对应的自然就是温度数据了

使用正则表达式做数据条切分 同时识别"电" 关键字 取得对应的数据存入列表之中 然后就是之前一样的绘图步骤了呢

对应的流程图如下



**V1.2.0 & V1.3.0**

**获得的温度/电压数据间隔时间太长 如何操作让最后获得的图像更加''平滑''**

**减少了每次进行一次采样所需要等待的时间 快了一些**

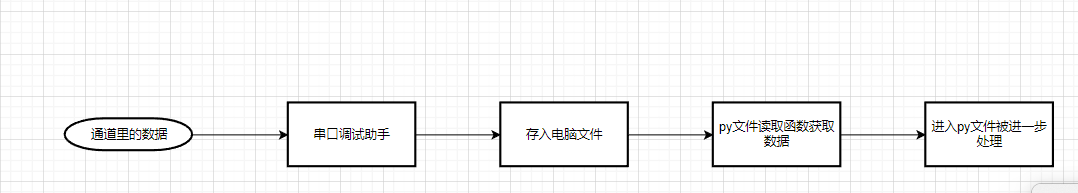
**delay时间 变成了原来的0.1倍**

#### 

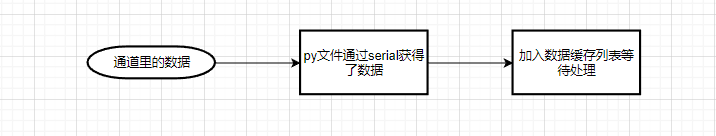
#### 使用串口调试助手把数据存在文件里面画图很麻烦 有没有更简便的方法

使用python的serial让py文件直接从通信的连接里面拿到了数据,这样就不用那么麻烦了

原来的串口数据的获取方式

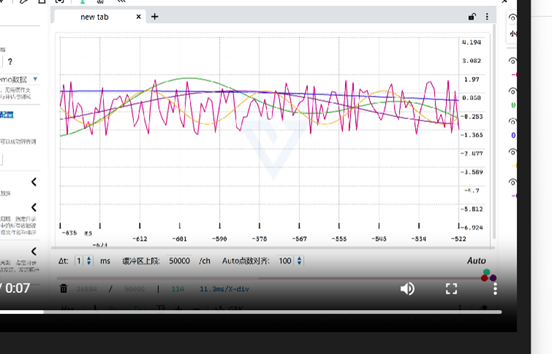


现在py文件拿到数据的方式



拿serial库画图其实还是有很多肉眼看上去觉得怪怪的问题 ,有没有更好的解决办法 (1.3.0)

解决办法 : 利用vofa+软件实现了流畅的波形显示



**V1.4.2**

**发包时采样速度和发包速度不匹配导致浪费,我该怎么做**

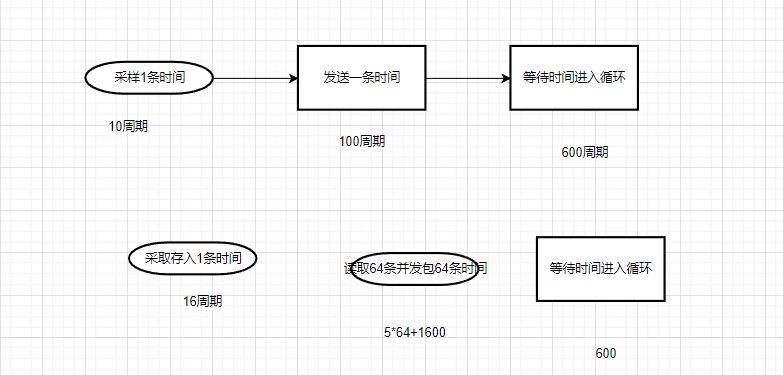
采样很快 发包慢 这样就会让采样的在那个地方干等着,所以采样的频率就低了 波形自然也不会平滑好看

解决办法

在采样过程中，将采样的数据存入flash当中 由于存储限制,每存入64条数据 发送一次数据.

采取理论假设的方法进一步说明

旧方法和新方法的数据流程如图



注明 随便设计的值 不代表具体情况或者甚至在剪短等待时间

旧版本 64\* (10+100+600)=45440---> 8000左右

新版本 16\*64+64\*5+1600+600=3544 3000左右

所以采样频率增加了至少一倍 是很大的进步

Flash存取部分具体设计实现过程见下文说明

|  |
| --- |
| Python # 初始化一个计数器i "FCTL2寄存器配置模式" "FWKEY--> 对FCTL1 FCTL2 FCTL3 写入许可" "FSSEL0+FN1 --> Flash Timing Generator 时钟源MCLK 分频1/3" #取值地址配置 "初始化Flash\_ptr指向Flash的0x1000" #"循环遍历" "共64组数据,存储在flash存储器的后128位当中" "每次读两位的数据,左移移位合并" "地址向后移动两位" "读到空数据后后跳出循环" "没读到一组数据,数据条数计数器+1" |
| Python //存储数据 "接受两位数据dat " "初始化数据拆分的暂存1位器变量 temp" "写满判断 未使用的flash存储空间指针初始化" "数据条数计数器+1 " "Flash 控制寄存器FCTL1 FCTL3配置" "FWKEY + WRT --> FCTL1 FCTL2 FCTL3 写入许可"  " FWKEY --> 清除锁定位" "现在可以进行写入操作了"  "低八位存入临时变量" "写入存储器指针后移" "高8位存入临时变量" "写入存储器指针后移" ""  "清除写入位" "重置锁存" |

**如何更方便的系统调试和维护**

**使用串口指令我们可以主动的获取当前板子内部的信息 方便调试和检查**

利用串口发送指令"55"可以获得存入flash闪存中的所有数据,串口发送"100"可以擦除flash闪存中的所有温度/电压数据

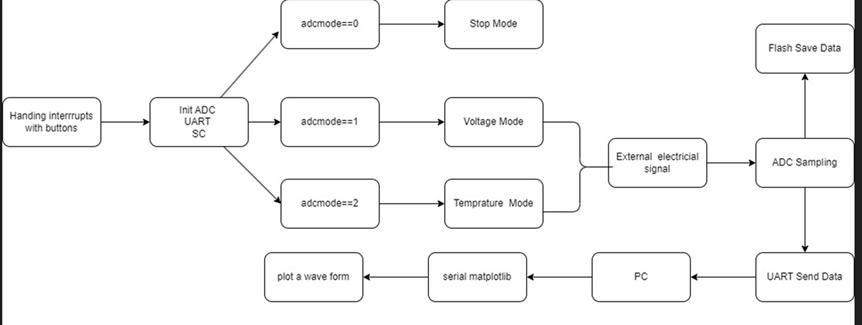
**具体实现逻辑如下**

|  |
| --- |
| Python "初始化计数器,初始化2位数据暂存变量"  "中断是否有数据接受" "读取缓冲区数据"  "读取到100,16进制AA,擦除全部数据" "初始计数器,flash指针" "擦除许可,清除锁存,虚拟擦除,清除写入,重制锁存" "串口返回擦除完毕报告"   "读取到55,返回全部数据" "初始计数器,flash指针" "分位取得,合并数据,返回数据,直到取完" "串口返回读取完毕报告" |

最后的版本 V1.4.2的运行逻辑如下

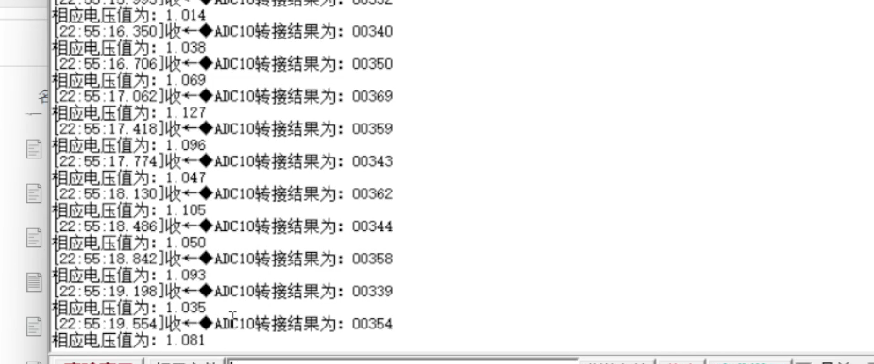
|  |
| --- |
| Plain Text "初始化1.6 1.3-->按键控制.中断" "初始化时钟,UART,ADC"  "模式选择" "Mea\_T" "Mea\_V" "TEST--extends-->AA-->clear-->flash" " -->55-->output all dat in flash" |

具体的运行流程(简化了一些之前详细说明过的地方)如下所示

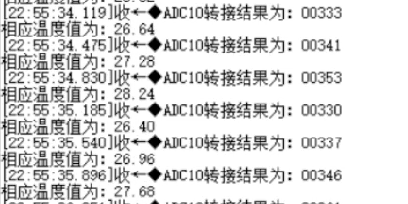


**第三章执行情况与完成度**

* 电压值获取(情况 已经成功)



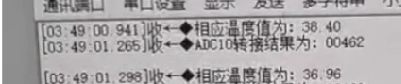
* 温度值获得(情况 已经成功)

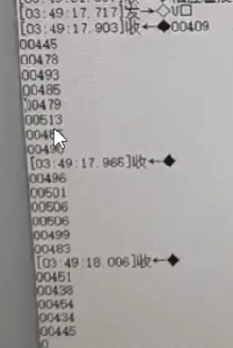


* 三种采集模式 (温度 电压 调试)

情况 已成功如图 分别是电压模式 温度模式 调试模式(指令为发送全部数据时)



****

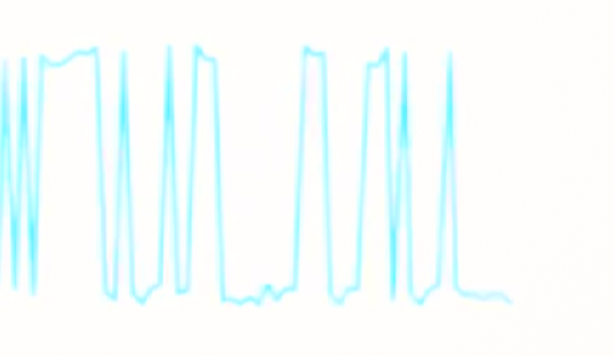
****

* 发送指令的调试模式

如上图 情况 已经完成

串口发送指令"55"可以获得存入flash闪存中的所有数据,串口发送"100"可以擦除flash闪存中的所有温度/电压数据

* **serial库实现实时读写**

****

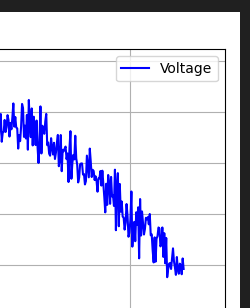
如图 实时动态绘图 视频在ITEM下Teyvant1.4.2的MP4文件里面看

* 温度电压绘图

如上图 情况 已经完成

* 采样速率的提高和让波形更加平滑细腻

如图 使用了flash存储之后批量发包的数据得到的图像的部分性质有了很大的提高



* 使用Git 和github做项目管理

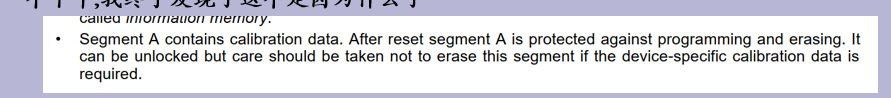
如下图所示较为有效实现了版本控制和代码托管



* 有效的养成了嵌入式工程师排查错误 和利用资源查询对应资料 联系实际硬件情况完成复杂工程问题的实际能力

用实际例子说明:

往单片机中增加Flash存储的新功能,一切都很顺利,不到一点钟我就差不多完成了,烧写运行也没有问题,串口也得到了数据,但是呢,突然就显示我失去了连接,发生了什么啊,我什么都没有做啊,为什么不管怎么搞都调不同呢.而之后不管是怎么做,都无法得到预想的结果,我继续尝试,还是不行,退回到原来的版本也不行.我于是感觉,嗯,不会是板子烧坏了吧,于是我继续拿着这个板子排查错误,整个下午,我终于发现了这个是因为什么了



并且我通过查数据手册,然后到处论坛,到处搜索,发现了我存的Flash的存储空间是不可以全部给我的,特别是我把用于校准的时钟信息给擦了,这个东西擦了,对我来说,就没有办法给他弄回去了,擦了之后,单片机的那块芯片就和报废没有区别(对我而言,因为我修不好) 最后 我换了一块板子 ,把存储数据的条数最多换成64条,这样 我就不在发生之前的问题,再短暂的调试了不到一个小时,我就搞定了.不过,就因为这个问题,我下午一直到晚上8点多都在一直弄,花了整整一天的时间排查出了问题,我感到特别恼火之后的轻松

**第四章分工协作与交流情况**

我们的分工如下：

|  |  |
| --- | --- |
| 张磊 | 负责统筹规划项目，完成adc电压，温度采集，flash闪存的功能，数信报告文档的编写，ppt的编写，对github上项目进行代码版本管理 |
| 庄骐嘉 | 完成adc温度电压中断切换，辅助完成flash闪存功能，数信中期报告文档编写，python基于matplotlib画图代码的实现 |
| 徐兴 | 完成了基于serial接口的串口读取，完成了电压，温度采集的实现。 |
| 王天瑞 | 完成了电压，温度采集的实现，参与了ppt的制作 |
| 邓淼 | 完成了电压，温度采集的实现，完成了基于vofa+的波形显示 |

**参考文献**

1. 刘成尧 MSP430LauchPad项目化学习指南 北京航空航天大学出版社 2015-06-01 ISBN:98757417571
2. MSP430x2xx 系列用户指南 Literature Number :ZHCU032I December2004 -Revised January 2012
3. MSP430x2xx Family User's Guide Literature Number :SLAU144J December2004 -Revised July 2013
4. 玩转 -- TI MSP430 LAUCHPAD TI EEWorld 2013-04

**答谢**

在这学期的综合设计课题完结之际，我们想表达最深感激之情，感谢管庆老师，您在项目中对我们的指导和建议是我们完成项目的重要动力。从项目开始到结束，您不仅为我们提供了丰富的专业知识，还对我们的每一个问题都进行了详细的解答和指导，让我们更好地理解和应用所学知识。

在整个项目中，管庆老师给我们提供了很多建设性的反馈，并且在提出的0->1,1->99的项目建设方案，给了我们很多的启示与帮助。我们不仅学到了很多单片机知识，还提高了自己阅读文档和解决问题的能力。您在项目中严谨的态度和认真的工作态度一直鼓舞着我们，使我们在完成项目的过程中始终保持了高度的责任感。您的这种态度和工作方式，对我们今后的学习和工作都将产生深远的影响。

此外，本次项目的顺利完成，也离不开我们小组的相互促进与耐心交流，感谢各位小组成员的付出。

最后，再次感谢管庆老师对我们无私的关心和指导，祝愿身体健康，工作顺利！

**学生信息**

* 学号:2022090917007
* 姓名: 庄骐嘉
* 学院: 信息与软件工程学院
* 系别: 数字信号处理
* 执行学期: 2023秋季学期
* 学号:2022090917008
* 姓名:张磊
* 学院: 信息与软件工程学院
* 系别: 数字信号处理
* 执行学期: 2023秋季学期
* 学号:20220909
* 姓名:邓淼
* 学院: 信息与软件工程学院
* 系别: 数字信号处理
* 执行学期: 2023秋季学期
* 学号:20220909
* 姓名:徐兴
* 学院: 信息与软件工程学院
* 系别: 数字信号处理
* 执行学期: 2023秋季学期
* 学号:20220909
* 姓名:王天瑞
* 学院: 信息与软件工程学院
* 系别: 数字信号处理
* 执行学期: 2023秋季学期

**课程信息**

* 课程名称: 进阶式挑战性综合项目I
* 课题名称:**异构多核音频处理**
* 指导教师: 管庆

**日期**

* 提交日期:2024/01/08

|  |  |  |
| --- | --- | --- |
| 序号 | 学号 | 姓名 |
| 1 | 2022090917007 | 庄骐嘉 |
| 2 | 2022090917008 | 张磊 |
| 3 | 2022000000000 | 邓淼 |
| 4 | 20220000000000 | 徐兴 |
| 5 | 20220000000000 | 王天瑞 |
|  |  |  |