
# 硬件实验课设报告

## Arduino 板数据采集

姓名：马逸君

学号：17300180070

2019 年 12 月

# Arduino 板数据采集

## 项目目标

### 1. 给定条件

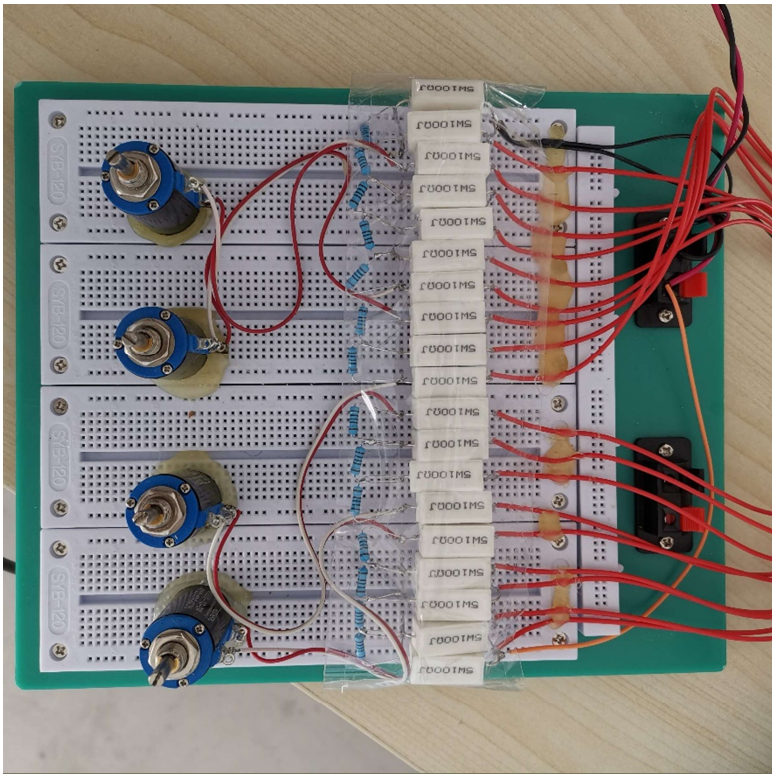
Linear Technology DC2026C CPU 板、DC1941D isoSPI 转接电路板、DC2350AB 18 路电压测量板、测试电路、万用表、相关线材
Arduino IDE、串口接收软件 SerialPortListener、VS2019 绘图控件 Chart

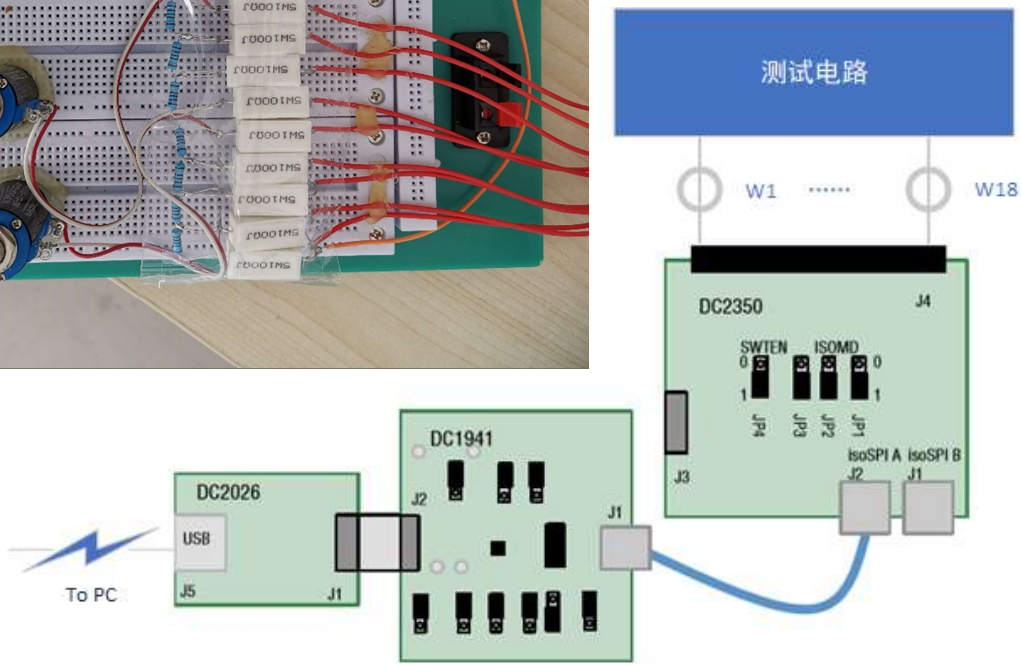
### 2. 预期效果

利用给定的三块电路板实现一个电压测量电路，通过串口实时得到测试电路的电压测量数据，绘制成图像；
在 Arduino 代码中增加一个指令，该指令按照代码中设定的参数（总时间和放电-休息循环的时长）启动放电功能

## 硬件原理

### 1. 硬件逻辑图





## 2. 部件功能描述

**DC2350AB（DC: demonstration circuit）是一个 18 节电池组监视器，可连接 18 路电压并同时测量。**其核心是 LTC®6813-1 芯片。可以通过 2 线 isoSPI (isolated serial interface，隔离串行接口) 同时连接多个 DC2350AB 板，来监控一个电池组中任意多节电池（的电压等参数）。DC2350AB 也支持反向 isoSPI，从而可以建立全冗余的通信链路。DC2350AB 可以直接连接一块 DC2026 Linduino® One CPU 板与 PC 通信，该 DC2026 必须烧录合适的程序 (称为"sketch") 来控制 DC2350AB 的 IC 电路并通过 USB 接口接收数据；也可以通过 DC2792/DC1941 板与 DC2026 CPU 板连接，DC2792/DC1941 提供完全隔离的 isoSPI 接口。

**Linear Technology 公司的 DC2026C CPU 板（也叫 Linduino® One）是一款微控制器板，它搭载 CPU，它可以烧录用户代码以控制 DC2350AB 的运行。**Linduino One 与 Arduino Uno 兼容。Arduino 硬件由一款带有支持快速在线(in-circuit)固件升级功能的 bootloader 程序的 Atmel 微控制器组成；他们的软件是一个基于 AVRGCC 编译器的简单编程环境。这套平台因其易用性而流行，硬件和软件都是开源的，且可以通过 C 语言编程。它是一个理想的展示分发具有数字接口（例如互联集成电路(I2C)和串行外设接口(SPI)）的集成电路共享库的方式。相比标准的 Arduino Uno，Linduino One 添加了若干新特性：14 针 QuikEval™连接器，可支持直连数百款且不断增多的 Linear Technology 电路板，还支持 5V/3.3V/2.5V/1.8V 多个逻辑级别电压和模拟信号支持；LTM®2884，提供 USB 供电和数据的电流级分离，提升安全性且便于噪声控制；7~20V 辅助电源输入，位于互相隔离的若干侧边，并由 LT®3973 转换为 5V 的电压，允许在 750mA 的高电流下工作。

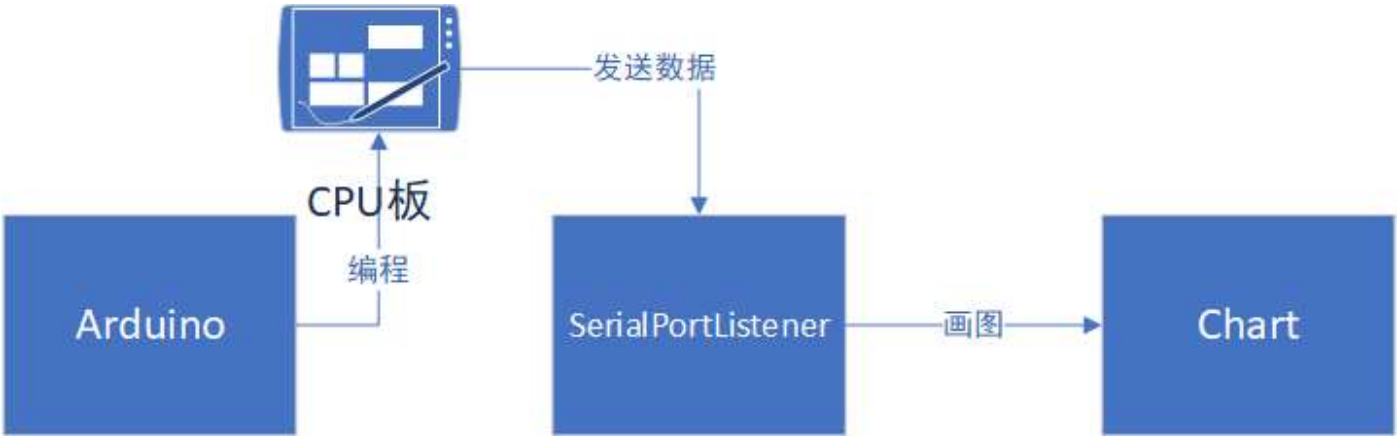
**LTC®6813-1 芯片是一款多节电池组监视器，它测量最多 18 节相连接的电池组（的电压），且总测量误差小于 2.2mV。**每节的量程为 0~5V，使得 LTC6813-1 适合大多数电池的化学性质。18 节电池的所有测量可在 290 微秒时间内完成，也可根据噪声控制的需求选择较低的数据采集速率。多台 LTC6813-1 设备可被连接成组，从而允许同时测量很长或电压很高的电池串。每个 LTC6813-1 都有一个 isoSPI 接口，支持高速、抗射频干扰、长距离的通信。多台设备连接成一种称为“菊花链”(daisy chain)的形态，所有设备与一个主要处理器(host processor)都建立连接。这个菊花链可以双向运行，从而保证通信完整性，即使是在通信链路中有故障发生的情况下。LTC6813-1 可以直接从电池组获得电源，也可以独立供电。其他特性包括：板载 5V 稳压器；9 条通用 I/O 线路；睡眠模式，该模式下电流消耗降低至 6 微安。

**DC1941 是一款 isoSPI™收发器电路板，它负责 isoSPI/SPI 信号的转换。**它基于 Linear Technology 的 LTC®6820 芯片。LTC6820 提供一种使用单条双绞线互连标准 SPI 设备的方法，这种专有的 2 线接口称为 isoSPI **(抗干扰能力强)**。LTC6820 以最高 1Mbps 的速度将 SPI 信号编码成差分 isoSPI 信号，后者通过一个简单的脉冲变压器经由双绞线传输。在这条双绞线的另一端，这个 isoSPI 信号可以被用另一个 LTC6820 翻译回 SPI。此外还可以通过 isoSPI 连接带有内置 isoSPI 接口的 Linear Technology 设备，例如 LTC6804。一般地，LTC6820 有两种用法：用一个 LTC6820 将 isoSPI 信号翻译成 SPI；用两个 LTC6820，将 SPI 信号翻译成 isoSPI，经过长距离的、隔离的通信后再翻译回 SPI。

（整理自相关器件的用户文档）

## 软件原理

### 1. 软件框图




### 2. 软件流程描述

Arduino IDE 将电压测量板的 Demo 程序烧录到 CPU 板中，串口收发模块向已编程的 CPU 板发送“开始循环测量”指令并持续接收 CPU 板发来的每组测量结果显示在窗口中，同时整理成绘图软件需要的形式供其绘图；或是向已编程的 CPU 板发送“启动放电”指令，启动放电过程。

### 3. 重要模块描述

DC2350AB.ino：编程 DC2026C CPU 板的 Arduino 项目文件。使用 C 语言。该代码使得我们能够通过向 DC2026C 发命令的方式操控 DC2350AB。我们也可以仿照其自带的命令源码，编写我们自定义的命令。
SerialPortListener：串口接收软件。使用 C#语言。可设计出窗体并与 Chart 控件集成，从而直接将 CPU 板发来的电压测量数据转换成曲线图。
Chart：Visual Studio 2019 自带绘图控件。

## 实验步骤

第一步：速读相关文档和 Arduino 代码，了解各部件的概况，了解文档和源代码的大致内容以备日后按需查阅
第二步：按照原理图完成较简易版本的电路连接
第三步：在较简易版本的电路上，完成绘图模块的设计及其与串口收发模块的连接
第四步：完成完整版本的电路连接，并完善其他模块，实现电压测量绘图的完整效果
第五步：在读懂文档的基础上，在 Arduino 代码中写“启动放电”指令，并通过自行设计的测试

## 总结及问题讨论

本次实验中我主要有三点收获：

对硬件的认识。使用可自由编程的 CPU 板和硬件的 IDE，我们也可以像编译运行一般的 C 程序那样设计硬件的功能，从而得到想要的硬件电路。只要掌握了硬件的设计规律，如寄存器的含义，我们还可以按自己的意愿改动原厂设计代码，得到更多更灵活的设计。（如图，原厂代码中只提供 33 个命令，34 号命令是我们自行添加的）

```
List of LTC6813 Command:
Write and Read Configuration: 1                          |Loop measurements with data-log output : 12                          |Set Discharge: 23
Read Configuration: 2                                    |Clear Registers: 13                                                  |Clear Discharge: 24
Start Cell Voltage Conversion: 3                         |Run Mux Self Test: 14                                                |Write and Read of PWM : 25
Read Cell Voltages: 4                                    |Run ADC Self Test: 15                                                |Write and  Read of S control : 26
Start Aux Voltage Conversion: 5                          |ADC overlap Test : 16                                                |Clear S control register : 27
Read Aux Voltages: 6                                     |Run Digital Redundancy Test: 17                                      |SPI Communication  : 28
Start Stat Voltage Conversion: 7                         |Open Wire Test for single cell detection: 18                         |I2C Communication Write to Slave :29
Read Stat Voltages: 8                                    |Open Wire Test for multiple cell or two consecutive cells detection:19 |I2C Communication Read from Slave :30
Start Combined Cell Voltage and GPIO1, GPIO2 Conversion: 9 |Open wire for Auxiliary Measurement: 20                            |Enable MUTE : 31
Start  Cell Voltage and Sum of cells : 10                |Print PEC Counter: 21                                                |Disable MUTE : 32
Loop Measurements: 11                                    |Reset PEC Counter: 22                                                |Set or reset the gpio pins: 33

Print 'm' for menu
Please enter command:

34
Written Configuration A Register:
CFGA IC 1, 0xE4, 0x52, 0x27, 0xA0, 0xFF, 0x2F, Calculated PEC: 0xA3, 0x76

Written Configuration B Register:
CFGB IC 1, 0xF0, 0x0F, 0x00, 0x00, 0x00, 0x00, Calculated PEC: 0x77, 0x3E

Written PWM Configuration:
IC 1, 0x33, 0x33, 0x33, 0x33, 0x33, 0x33, Calculated PEC: 0x65, 0xBA

 PWM/S control register group B:
 IC: 1
 0x33, 0x33, 0x33, Calculated PEC: 0x52, 0xAA

Written Data in Sctrl register:
 IC: 1 Sctrl register group:, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, Calculated PEC: 0x66, 0x4C

 PWM/S control register group B:
 IC: 1
 0xFF, 0xFF, 0xFF, Calculated PEC: 0x29, 0x42
```

速读并整理文档的能力。一下子拿到这么多文档，其实我一开始是不知所措的。但是我打开各个文档，读了一下一开始的概述部分，读了一下目录，浏览了一下重点章节，就知道每个文档大致在讲什么了，以及什么时候可能需要查阅哪个文档。为了清楚起见，我还在相关文件的文件名后面添加了摘要，便于查阅（见下图）。另外在需要的时候，还可以在文档中采用关键词搜索的方式速读。这极大提升了我利用文档的效率，例如在实验进行到第四步时，我非常顺利地在 DC2350AF 的文档中找到了第 8 页的 DC1941 isoSPI MASTER SETTINGS 和第 9 页的 DC1941 TO DC2350 TYPICAL isoSPI CONNECTION，为实验进程提供了很大的帮助；在进行到第五步时，我在 LTC6813-1 的文档中查找"balanc"（注：这是因为文档中出现了这个单词的两个时态"balance"和"balancing"，为了简单起见，就用它们共有的"balanc"这个字符串进行查找）、DCC、S Pin、DCTO、discharge timer 这些字符串并速读它们出现的文段，很快就弄懂了放电相关功能的运作。

简单使用 C#语言的能力。因为绘图控件必须用到 C#，所以我只能自学这种陌生语言。在已经自学过 C#的优秀的胡永祥同学的指导下，我知道了 C#代码执行的基本逻辑：每当有事件发生时，就执行该事件对应的处理函数。所以我就把 UpdateChart()函数（更新图像）的调用放在（串口接收）函数中，并在 StartListening()函数中初始化图像、设定串口参数。再加上在网上搜索到的一些 Chart 控件用户代码、对 MSDN 网站上相关的类定义的查阅，成功完成了最终的绘图模块。