東南大學

毕业设计(论文)报告

题　目　基于安卓平台的 PocketLab 虚拟仪器设计

信息科学与工程学院 院（系） 信息工程 专业

学　　号　04012527

姓　　名　何功垠

指导教师　孟桥

顾问教师

起止日期　2016 年 2 月 22 日至 2016 年 6 月 5 日

设计地点　无线谷

基于安卓平台的 PocketLab 虚拟仪器设计

摘 要

随着仪器技术的发展，测量仪器从早期的模拟式发展到数字式，从传统的台式测量仪器发展到虚拟仪器。当前的虚拟仪器大多是基于 PC（个人电脑）机进行开发和设计的。近些年随着移动便携设备的快速发展，Android 平台设备的使用也得到了普及。然而目前的基于 Android 平台开发的虚拟仪器少之又少，且其中大多数距离投入实际使用尚存在很大的差距。但是与普通的基于 PC 的虚拟仪器相比，基于 Android 移动终端的虚拟仪器系统将更便于携带，使用更为方便。如果能在移动终端上实现虚拟仪器，将会对教学与实验产生重大意义。

本课题的内容就是结合当前虚拟仪器以及便携式仪器的发展状况，基于目前已有的 PocketLab 硬件环境，设计出一种功能全面的、操作简便的、基于 Android 移动终端的虚拟仪器系统。论文主要通过以下几个部分进行展开。首先是对虚拟仪器的发展状况作一总体的回顾与论述，然后对 Android 系统的体系架构进行简要的介绍。在各种开发平台中，选择了 google 最新推出的官方 Android 开发平台 Android Studio，使用 Java 语言完成了 PocketLab App（手机应用）的开发，确定了整体框架设计，以及网络连接、数据传输等基础的功能模块的实现方案。

在此基础上，完成了示波器、信号发生器、直流电压表、逻辑分析等 Android 平台下的虚拟仪器的设计，其中示波器可以显示双通道波形，并且可以设置时间单位、电压单位、耦合方式和垂直位移；信号发生器可以输出双通道波形，并且可独立设置波形、频率、垂直偏移和占空比；直流电压表可以测量双通道输入的直流电源，并可以输出双通道独立直流电压；逻辑分析仪具有 8 个输入输出口，并可以设置输入输出时钟模式和输出电平，还可以绘制逻辑电平图。其中使用 WIFI 与硬件建立连接，使用 Socket 协议进行高速通信；使用了 ImageView、RadioButton、Switch、SeekBar 等多种空间；应用了多线程技术；并且应用了数个开源项目美化界面、优化性能。最后本项目软件不仅功能较为完善，界面美观，用户体验可操作性良好。

关键词：虚拟仪器；PocketLab；Android 系统；信号发生器；示波器；直流电压表；逻辑分析仪

DESIGN AND IMPLEMENTATION OF POCKETLAB VIRTURAL INSTRUMENT BASED ON THE ANDROID PLATFORM

Abstract

With the development of instrument technology, measuring instruments have developed from the early analog to digital, and from traditional desktop instruments to virtual instruments. Now the virtual instrument is mostly developed and designed on PC. Android mobile devices have been very popular in modern people, because of the fast development of the mobile devices. However, the number of virtual instruments based on the Android platform is very little, and they still have a big gap to the actual application. However, compared with the common virtual instruments based on PC, the virtual instrument system based on Android mobile terminal will be more convenient to be carried and used. The virtual instrument on mobile terminal will be remarkablely meanful to the education and experiment.

The content of this project is a combination of development of the virtual instruments and the portable instruments, based on the existing PocketLab hardware to design a fully functional and easy-to-use virtual instrument system based on the Android mobile devices. This paper has the following several main parts: the first part is the status of virtual instruments' development for an overall review and discussion, and then the paper briefly introduces the Android system architecture. The next is why to choose Android Studio as the development environment between two popular Android development platform. It also introduces the the overall framework of the design, and the fundamental modules of the network connection and data transmission.

On this basis, this project completed the design of oscilloscope, signal generator, voltmeter and logic analyzer virtural instruments based on Android platform. The oscilloscope can display double-channel wave, and set time, volts, coupled mode and offset; the signal generator can output double-channel wave, and set wave form, frequency, offset and duty factor respectively; the voltmeter can measure and output double-channel DC voltage; the logic analyzer has 8 I/O ports which can be set input, output or clock mode, and it can draw logic level chart. The app uses wifi to establish connection with the hardware and use Socket to build high-speed communication; it also uses various widget, such as ImageView, RadioButton, Switch and SeekBar; Multi-thread technique is applied in the app; the application of open source projects makes the app' interface more beautiful and performance better. Eventurally, this app not only has rich functions but also has beautiful surface and friendly interaction.

KEYWORDS: Virtual instrument; PocketLab; Android; signal generator; oscilloscope; voltmeter; logic analyzer

目 录

第1章 绪论

1.1 引言

各类电子仪器是电子工程师们设计、制造、修理电子设备时不可或缺的工具，比如示波器、信号发生器、逻辑分析仪等常用仪器，也是电子类专业的师生们在课程教学和专业学习中所不可缺少的工具，在整个科研与工程领域，都具有十分重要的作用。

从模拟到数字，从简单到智能，各种电子测量仪器随着科技的进步，不断取得发展，其精度不断提高，功能不断增强，满足人们在各种场合下的不同要求。与模拟仪器、数字仪器等传统仪器相比，新一代的虚拟仪器（Virtual Instruments）融合了现代的计算机技术、测量技术、通信技术，它以软件为核心，具有开发和应用的灵活性[1]。

近些年随着移动终端的快速发展，手机已经广泛进入了人们的必备品清单，并成为日常生活必不可少的一部分。因为其的目前流行的热度，手机软件开发成为当前的焦点产业之一，那么虚拟仪器与手机软件的碰撞一定可以产生巨大的能量。

1.2 本课题的研究背景及研究意义

1.2.1 虚拟仪器技术

与传统仪器相比，虚拟虚拟仪器最大的不同，也是它的特点所在，就在于它的最主要的功能都是基于计算机而实现的。虚拟仪器系统的组成结构如图所示：

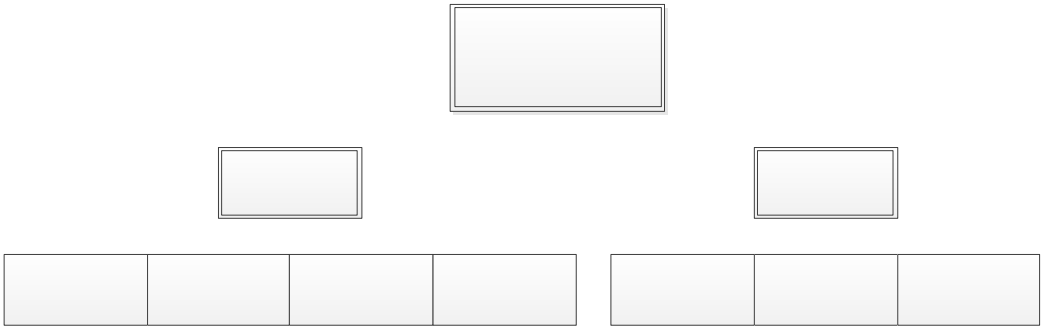


图 1-1 虚拟仪器系统的组成

硬件系统是虚拟仪器系统的基础，其所有的功能都是依托在硬件系统的支撑下才能够实现的。其中，计算机可以是通用式计算机系统，也可以是嵌入式计算机系统，虚拟仪器软件的显示功能、分析功能、操作功能都是在计算机硬件平台的基础上才能够完成的。测量电路则是虚拟仪器系统的最根本的硬件部分，没有它的话，则所有的功能都无从谈起。测量电路通过模数转换等方式，获取

被测量的信号的各种信息，它们一般表现为电压幅度等数据。IO 接口和总线则是数据传输的通道，通过 IO 接口与总线，将采集到的数据、信息发送给软件部分进行处理。

软件部分是虚拟仪器系统的灵魂，控制与显示模块和数据分析处理模块则是其核心所在。计算机具有强大的功能，它对数据的分析与处理、显示与存储方面，都有着突出的优点。使用计算机上的软件程序，可以很形象的展现仪器的操作面板，并且使用十分方便，可以完成复杂的调控和测试。对于所需要显示的检测结果，计算机软件可以非常方便且美观的将结果显示出来。在计算机平台上，通过编写复杂的数据处理程序，可以实现强大的信号的分析与处理功能。

与传统的仪器相比，虚拟仪器系统具有更好的开放性、集成性[2]。虚拟仪器系统开放灵活，它以计算机技术为基础，并与之同步发展[3]。使用软件以及模块化的硬件的实现方式，既可以实现软硬件的很好连接，又有利于集成多个模块、实现多种功能[4]。虚拟仪器的开发成本低，技术更新快，系统的性能和功能可以很方便的进行调整、升级、扩展。同时，成本低，价格低廉，仪器资源可重复利用[5]。

当然，在拥有这些优点的同时，不可否认，当前的虚拟仪器也存在一些局限，受限于计算机和接口卡的速度，大多数虚拟仪器系统的带宽一般较小、采样率较低，正因为这个原因，测试结果的准确度也不是很好。所以就目前来说，虚拟仪器主要应用于对速度和精度要求不是太高的场合[6]。

1.2.2 虚拟仪器技术的发展状况

随着仪器技术的不断发展以及用户对测量仪器要求的不断提升，市面上的各种电子测量仪器也在不断升级换代。用户在对仪器测量精度的要求提升的同时，对仪器的使用方便程度的要求也不断提高。而在某些情况下，对测量仪器的成本，也要求尽可能的低廉。在此基础上，虚拟仪器和便携式仪器技术不断取得发展和应用。

当前的虚拟仪器，大多是基于 PC 端使用 LacView、CVI 等开发环境进行设计[7]。LabVIEW 是由美国国家仪器公司开发的一种程序开发环境，它提供了很多的类似于示波器、万用表的外观的一些空间，因此在创建用户界面的时候十分方便[8]。

目前已有的，如文献[9]中提到的利用 LabVIEW8.5 平台实现的基于声卡的虚拟单踪数字存储示波器，和文献[10]中提到的基于 LabVIEW8.6 平台的虚拟示波器系统。这种虚拟示波器可以完成基本的信号分析与处理，并且可以在实验教学中投入使用。它们有一定的优势，如价格便宜，使用方便等，因此具有广泛的应用前景。而且通过对虚拟示波器的功能进行全面的测试，结果表明，在一般情况下，虚拟示波器可以基本取代传统的示波器[11]。

然而，无论是传统的电子测量仪器，或者是一些功能强大的基于 PC 的虚拟仪器系统，尽管功能完善、测量相对准确，但往往由于体积较大而不便于携带。它们固然可以达到很好的测量精度，但是，在某些情况下，这种高精度并不是使用者所需要的。比如在有的时候，我们只需要对一些简单的信号（比如正弦波、方波等）进行一些简单的测试，以判断是否符合预期信号的基本特征、是否发生信号的畸变，由此判断该设备是否存在故障。或者在基本的电路、电子线路、数字电路等课

程实践教学中，过高的精度并不是十分必要，这种情况下，这些高精度高复杂度的测量仪器的使用就显得大材小用。

在此同时，在某些场合下，相比于测量精度，便携性、使用方便程度可能更受某些用户关注。比如在野外对某些设备或者电路进行测试时，使用传统的电子测量仪器，或者基于 PC 的虚拟仪器系统，电源的携带将十分困难，环境的恶劣也都不利于测试。而且，在普通的电子线路实验教学中，相比于一台功能强大但操作复杂的测量仪器，一台操作简单、携带方便的测量仪器可能更受学生们的喜爱。

在这种情况下，便携式电子测量仪器便应运而生，这些便携式仪器对于工程现场电路的测试与维护等方面有着很重要的作用。它们一般集成了很多 IO 接口，如 USB、蓝牙、WIFI 等，通过这些接口，还可以和电脑进行同步的数据传输，从而强化其功能。文献[12]中，就介绍了一种基于 Arduino 和图形液晶显示器 GLCD 的低成本的便携式示波器的设计与开发。示波器通过读取样本，将它们存储到其内部 RAM 中，并显示实时波形。这些测量仪器供电方便、造价低廉、使用灵活方便。

1.2.3 基于智能手机的虚拟仪器的设计

目前，手机等移动终端已经进入了人们生活的各个方面。其价格不断下降，而硬件性能则在不断提高，智能移动设备可发挥的作用越来越大。同时由各种理论教育 APP 在手机上火热程度可以预见这样前无仅有的便于使用的虚拟仪器 APP 可以在高等教育和实验中产生重大的意义。

相比于 PC 系统上的虚拟仪器软件，目前还没有非常成熟的使用移动平台开发的虚拟仪器，即使是 TI 与 NI 这样的国际大公司的虚拟仪器软件也仅在 PC 端上，还没有涉足移动终端。目前已有的有的来自于个人开发爱好者们进行的尝试，这些软件大多功能简单，而且，移动端和数据采集电路之间，大多都是通过蓝牙来进行数据通信。还有一些，是来自在校大学生们进行的毕业设计，比如文献[13][14][15]中介绍的这种基于 Android 平台的虚拟示波器。

截止至 2015 年第三季度，国内活跃的 Android 设备已经达到了 10 亿以上。也就是说，Android 是目前市场占有率最高的手机平台。因此，本项目决定从 Android 开始，在当前虚拟仪器以及便携式仪器的发展及基础上，基于目前已有的 PocketLab 硬件环境，设计出一种功能全面的、操作简便的、基于 Android 移动终端的虚拟仪器系统。与普通的 PC 端虚拟仪器相比，本设计基于 Android 移动终端，因而使用更为方便，更便于携带，并且有很好的需求前景。且与当前的一些基于 Android 系统的虚拟示波器相比，本设计基于已有的功能完善、性能较强的硬件系统，因此可以达到更好的测量效果，同时功能较完善，用户体验较好。

1.3 本课题的设计目标

本设计的目标，是基于目前 PocketLab 硬件系统，设计一套在 Android 系统下的虚拟仪器系统，要求能够通过 wifi，控制 PocketLab 硬件的运行，并且在手机上提供一个虚拟的示波器、信号发生

器等仪器的界面，以满足电路、电子线路、数字电路等课程实践教学的需要。

第一章主要介绍了虚拟仪器的发展状况，并结合当前虚拟仪器的发展状况引出本课题的内容，即基于 Android 系统的虚拟仪器软件，并对本课题的研究背景、研究意义进行了介绍。第二章对 Android 系统本身的体系架构以及所需要使用的组件控件等进行一些介绍。第三章介绍本项目设计的整体框架思路和一些基本的功能模块。第四章介绍示波器、信号发生器等功能的具体实现以及实际测试情况。最后一部分内容是结语和致谢。

第2章 Android 系统的体系结构

2.1 Android 系统介绍

Android 的中文意思是“机器人”。在移动领域，Android 和 Google 是分不开的。它是由 Google 公司和开放手机联盟领导和开发的一种基于 Linux 内核的、主要应用于智能手机和平板电脑等移动设备上的操作系统[16]。目前在全球市场上，智能移动设备系统 Android、IOS 与 Windows 三分天下，然而 Android 系统在智能手机操作系统中所占的份额早已达到了 76%以上[17]。考虑到 Android 遥遥领先的使用率，因此本项目决定首先使用 Android 作为软件的开发平台。同时参考文献[18]中的叙述，相比于其他移动端操作系统，Android 在以下几个方面具有比较明显的优势：

1. 开放性。Android 平台是免费的，是开源的。这将使得 Android 系统可以成为一个开放式的生态系统，由于结合了全世界的使用者和开发者的共同努力，使得 Android 系统的更新更快，发展更迅速。

2. 开发人员可以自主决定应用的权限。在 Android 平台下开发应用程序，要使用一些具有限制性的 API，只需要简单的配置一下自己的应用即可。与之相比，使用 Symbian、Java ME 开发程序则要麻烦许多。从某种意义上讲，它也降低了开发 Android 程序所需要的成本。

3. 可自主性。Android 系统上的所有应用、组件都可以进行替换，因而可以根据自己的想法， 自主的打造属于自己的 Android。

4. 应用程序之间进行通信。在 Android 平台上，不同的应用程序之间进行通信非常方便。

5. 与 Web 的很好结合。Android 应用中可以很方便的嵌入 HTML，基于 WedKit 的 WedView 组件让 Android 应用可以很好的与 Web 结合在一起。

6. 物理键盘和虚拟键盘。Android 不仅支持物理键盘，同时也支持虚拟键盘，因而可以满足用户的不同需要。其中虚拟键盘已成为 Android 最主流的输入方式。

7. 充分体现个性。Android 提供了如 Widge、Shortcut、Live WallPapers 等的充分体现个性的功能，尽显手机的时尚。

8. 便捷好用的开发环境。过去 Android 开发比较主流的是使用 Eclipse 软件和 ADT 插件。但是近些年新推出的 Android Studio 作为 Google 推出的官方 Android 平台具有更多更前沿的优势，本项目也大胆使用了 Android Studio 作为开发平台。

2.2 Android 的系统架构

2.2.1 Android 的系统层次

Android 系统的层次架构从整体上来看，可以分为四层架构[19][20][21]。这四层由高到低分别是：应用程序（Applications）层、应用程序框架（Application Framework）层、系统运行库（Librarys）

层、Linux 核心（Linux Kernel）层。在系统运行库中还包括有一个区域，Android 运行时（Android Runtime）。

图 2-1 Android 系统四层架构

2.2.2.1 应用程序层

Android 系统的最高一层是应用程序层，实际上也就是直接和用户进行交互的一层。使用者并不会接触到下面的几层是如何运行的，对于他们来说，他们直接看到的和使用的，都是一个个应用程序。包括日历、短信、拨号、联系人、浏览器等，也包括用户从网络上下载安装的 Android 程序。这些程序是以.apk 格式存在，apk 是 Android Package 的缩写，也就是 Android 安装包的意思。实际上 apk 文件就是一种 zip 压缩文件，使用解压缩软件将之解压，可以发现其中包括存放有各种资源的 res 目录，对程序进行配置的 AndroidManifest.xml 文件，以及编译好的 Dalvik 字节码文件 classes.dex 等一些内容。这个.dex 字节码文件包含有已编译好的各种类，在 Dalvik 虚拟机中运行[22]。

2.2.1.2 应用程序框架层

应用程序框架层提供了比较高级的用来构建 Android 程序的各种模块，其中最主要的是内容提供者（Content Providers）、视图（Views）、资源管理器（Resource Manager）、通知管理器（Notification Manager）、活动管理器（Activity Manager）。其中，内容提供者主要是对应用程序之间进行数据的共享提供支持；视图包括列表、文本框、按钮甚至可嵌入的 Wed 浏览器等基本控件，是构建应用

程序一些基本模块；资源管理器提供对本地的图片、布局文件等非代码部分资源的访问；通知管理器负责将各种自定义提示信息在状态栏中显示出来，包括短信、下载成功、警告等；活动管理器负责控制一个应用程序的生命周期，并且还可以提供一些常用的导航、后退等功能[23]。

实际上这一层实现了应用程序所需要的一些通用的基础功能，通过它提供的 API 接口，用户可以在自己的 Android 程序中使用这些功能并且在进行扩充。从这个角度去讲，对于一些基础的功能，应用程序可以调用其 API 接口，而不需要再进行重复的实现。这种程序的重用机制简化了 Android 程序的开发。

2.2.1.3 系统运行库层

Android 系统运行库层包括程序库层和一个非常重要的区域：Android 运行时。

这一层的程序库主要包括一些用 C 或 C++编写的为 Android 系统提供底层支持的一些库。比如提供数据库功能的 SQLite 库、提供 3D 绘图支持的 OpenGL 库，提供浏览器内核支持的 Webkit 库等。这些基本的库并不直接提供开发者使用，而是通过上一层的 Android 应用程序框架层为开发者提供一些基础服务。

从图 2-1 中我们可以清楚的看到，Android 运行时包括两个部分，一个是核心库（Core Libraries），另一个是 Dalvik 虚拟机（Dalvik Virtual Machine）。其中，Android 核心库提供了 Java 语言核心库的大部分功能[24]。它的作用颇类似于 JNI 技术，即通过 Java 语言调用 C 或 C++编写的开发库来实现功能。Dalvik 其实就是 Google 公司的开发运行在 Android 平台中的 Java 虚拟机，它是 Android 平台的核心组成部分之一。C 或 C++语言编写的程序通过编译，生成可以直接被机器识别的机器码文件。而使用 Java 编写的所有类则被编译转换成为.dex 格式的字节码文件，可以运行在 Java 虚拟机中。Dalvik 就是用于 Android 系统中的 Java 虚拟机，它支持同时运行多个 Dalvik 虚拟机的实例，每一个 Android 程序都在一个独立的 Dalvik 虚拟机中。这种设计的好处在于，当虚拟机因为某些原因崩溃时，不会导致所有的应用程序都被关闭。

Dalvik 虚拟机运行于 Linux 底层内核的基础之上，Linux 内核中的一些线程和内存机制为 Dalvik 虚拟机提供底层支持。

然而一直以来，Dalvik 虚拟机被认为是导致 Android 系统运行的流畅程度不如 iOS 的根源之一。在 2014 年 6 月 25 日 Google 推出的 Android L 中， Dalvik 被彻底删除，为 ART 取代。

2.2.1.4 Linux 内核层

Android 系统是基于 Linux 内核的，Linux 内核是 Android 系统的最基础最底层的一层[25]。Linux 内核基本是使用 C 语言进行编写的，而无论是 Android 系统的使用者还是软件的开发者，他们接触到的都是使用 Java 编写的上层的应用程序和应用框架。它们之间的 Dalvik 虚拟机则实现了将 Java 程序运行在以 C 语言编写的 Linux 平台上的这一基础功能。Linux 内核层为 Android 系统提供一些如内存、线程、网络管理等基础的核心服务。同时，也未 Android 设备的各种硬件，如显示、音频、

蓝牙、电源等，提供了底层的驱动支持。Linux 内核层可以视为一个硬件和软件之间的抽象层，通过它实现硬件和软件的沟通。

2.2.2 Android 的四大组件

Android 四大基本组件分别是 Activity 活动，Service 服务，Content Provider 内容提供者，BroadcastReceiver 广播接收器[26][27]。一个 Android 应用程序未必会同时用到这四种组件，它们可以根据开发者的需要进行各种组合。

2.2.2.1 Activity 活动

Activity 是最基本的组件，它是 Android 的核心类，所有的 Activity 都继承自 android.app.Activity。在一个 Android 应用程序中，一个 Activity 就相当于一个单独的界面，而每一个 Activity 都被作为一个单独的类。一个 Android 程序往往包括一个或多个 Activity，因此经常会从一个界面跳转到另一个界面，新的界面打开以后，前一个界面就会被保存到历史栈中，可以从后面的界面返回到前面的界面去。简而言之，一个 Activity 就是用户所能看到的一个屏幕，在这个屏幕上包含有各种可视化组件，用户通过这些组件与 Android 程序进行交互，Android 程序则可以通过它们监听用户产生的事件并且为这些事件作出响应。这些组件的安排是通过 layout 目录下的.xml 布局文件进行描述的。

大多数含有多个界面的 Android 程序中，是将其中某一个 Activity 作为主界面，相当于初始界面。程序启动的时候会先启动这一个 Activity，在这个 Activity 中，根据用户的操作，启动其他的 Activity。在不同的 Activity 之间，Android 使用 Intent 类来实现切换以及数据的传输。

2.2.2.2 Service 服务

Service 的中文意思是服务，顾名思义，Service 就是一个运行在后台、没有用户界面、生命周期比较长的程序。最具代表性的例子是一个正在播放音乐的播放器，当用户对这个播放器进行操作，从这个界面跳到另一个界面进行选择的时候，音乐的播放过程应该在后台一直进行，并且根据事先选择好的方式自动切换到下一首。或者用户可以使用其他软件看小说、网络聊天、浏览网页，后台的音乐播放过程也应该一直持续下去，一直到用户手动将它关闭。

Service 的启动可以通过在 Activity 中调用 Content.startService()方法，也可以通过调用 Context.bindService()方法，与一个 Service 连接起来。对于后者而言，如果这个 Service 本身还没有被启动，那么就启动它。当一个 Activity 连接到一个 Service 之后，就可以使用 Service 提供的一些接口和它通信，进行一些操作。使用 Context.startService()方法启动的 Service，因为它和调用者之间并没有什么关系，所以即便关闭调用者，Service 也不会被终止，如果如要终止它，可以通过调用 Context.stopService()方法。通过调用 Context.bindService()方法建立的 Service，它和调用者是绑定的，这种情况下，如果调用者被关闭，则该 Service 也会被终止。

2.2.2.3 Broadcast Receivers 广播接收者

Broadcast Receiver 组件的功能时接受广播传来的消息，并且对它做出响应。当电量不足、收到短信、接到来电等情况时，系统会发出广播消息。或者当数据的下载完成之后，也可以由应用程序发送广播消息，告知另外的应用程序数据已准备好，可以使用。应用程序通过 Broadcast Receiver 组件接收到这些消息时，可以根据开发者定义的内容对它进行响应。

所有的广播接收者组件都继承自 android.content.Broadcast-Receiver 类，对于一个应用程序来说，它可以有多个广播接收者组件，用来对不同的外部事件进行响应。同时，也可以按照一定的规则对外部事件进行过滤，只接收某些它所需要进行响应的事件。

Broadcast Receiver 并没有和 Activity 一样的用户界面，当它需要对某一事件进行响应时，它可以启动一个 Activity 或者启动一个 Service 来进行响应操作。或者它也可以采用其它的方式，比如通过使用 NotificationManager 来告知使用者。

我们必须清楚，Broadcast Receiver 的生命周期很短，大概只有 10 秒左右。这样的话，如果对于一个事件需要进行的响应操作比较复杂耗时，这里应该使用 Intent 把消息发送给 Service，由 Service 来完成这一操作。在这种情况下，使用子线程也是不可行的，因为 Broadcast Receivers 的生命周期很短，极有可能在子线程还没有运行完成的时候，Broadcast Receivers 已经被结束了。这样的话，宿主线程已经被杀死，那么由它所建立的子线程即使正在工作中，也会被杀死。

2.2.2.4 Content providers 内容提供者

Content Provider 是 Android 平台提供的用来实现数据共享机制的组件，一个 Content Provider 就是一个类，所有的 Content Provider 都需要继承自 android.content.ContentProvider 类。

如前文所介绍的那样，在 Android 系统中，每一个应用程序都运行在属于它们的进程中，每一个进程都对应一个 Dalvik 虚拟机。这种情况下，如果某一个应用程序需要访问来自其他应用程序的数据，这种操作将比较复杂。这种情况实际上就是说，数据需要在不同的 Dalvik 虚拟机之间进行传递，所以它比较困难。而 Content Provider 正是用来解决这一问题的。

2.3 Android 开发平台

目前较为主流的 Android 开发平台分别为 Eclipse 和 Android Studio，这两个开发平台各有千秋。

Eclipse 是 1999 年由 IBM 公司开发的集成开发环境，在 2001 年被贡献给开发开源社区。作为一个强大免费软件，它广受程序开发员们的喜爱，并且最经常被作为 Java 开发环境使用。在 Android 被 Google 推出后，Java 作为 Android 开发语言，Eclipse 便顺理成章成为了最主流开发平台。因此前些年的绝大多数 Android 项目都是在 Eclipse 环境下开发的，目前各种 Android 教程书籍还是以 Eclipse 作为开发环境来进行教学。

Android Studio 是 Google 在 2013 年 I/O 大会上推出的新的针对 Android 开发的开发工具。虽

然这款开发工具才仅仅推出 3 年而且在国内普及率也不高，但是已经席卷了国外走在前沿的高端 Android 开发工程师。因为他相对 Eclipse 有许多优点，比如：

1. 由 Google 推出是它最大的优点，出身“名门正统”，它针对 Android 量身定做；
2. 继承了 IntelliJ IDEA 强大的内核，启动、响应和处理速度都更快；
3. 更加智能，比如提示补全，自动保存多个版本并可以比较不同版本间的差异；
4. 支持丰富多样的插件，比如 Markdown、Git；
5. 整合了 Gradle，它是一个配置、编译、打包都非常方便的构建工具；

这还只是 Android Studio 优势的冰山一角，更多都在实际使用时可以体会到，它代表着 Android 开发的未来。虽然国内现在用 Android Studio 的还只是少数，但是本项目决定大胆使用 Android Studio 作为开发平台。

第3章 Android PocketLab 系统结构

3.1 PocketLab 硬件介绍

PocketLab 由东南大学信息科学与工程学院研发，是一个可以用于电路、电子线路、数字与逻辑电路等课程的虚拟实验室，其具有价格低、体积小、精度高等特点。同时，它也可以用于一些实际系统的数据采集、模拟与数字控制。

图 3-1 PocketLab 实物图

其主要具有信号发生器、示波器、逻辑分析仪等功能。

信号发生器可以产生波形可设置（正弦波、方波、三角波），频率、幅度、直流偏置可调的单通道信号，也可以产生直流偏置相等、互为差分信号（相位差 180°）的双通道信号。

示波器为两通道同步采样示波器，显示 C1、C2 的信号波形。

简易逻辑分析仪提供四种基本功能：

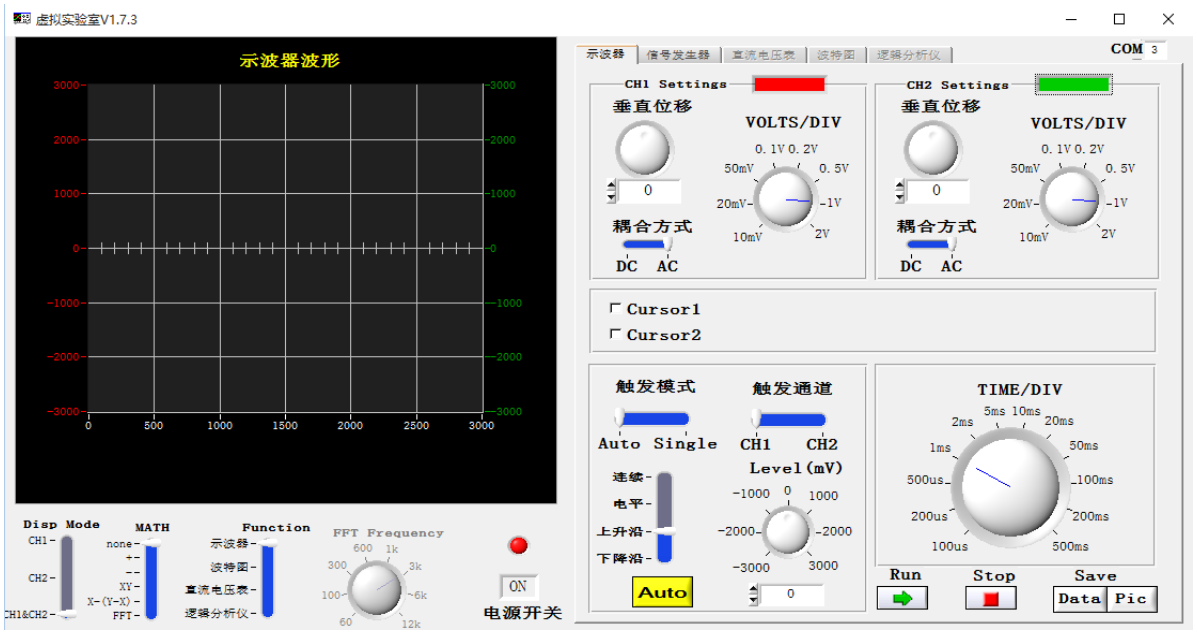
① 逻辑电平输入：读取指定引脚输入电平；

② 逻辑电平输出：使指定引脚输出指定的高低电平；

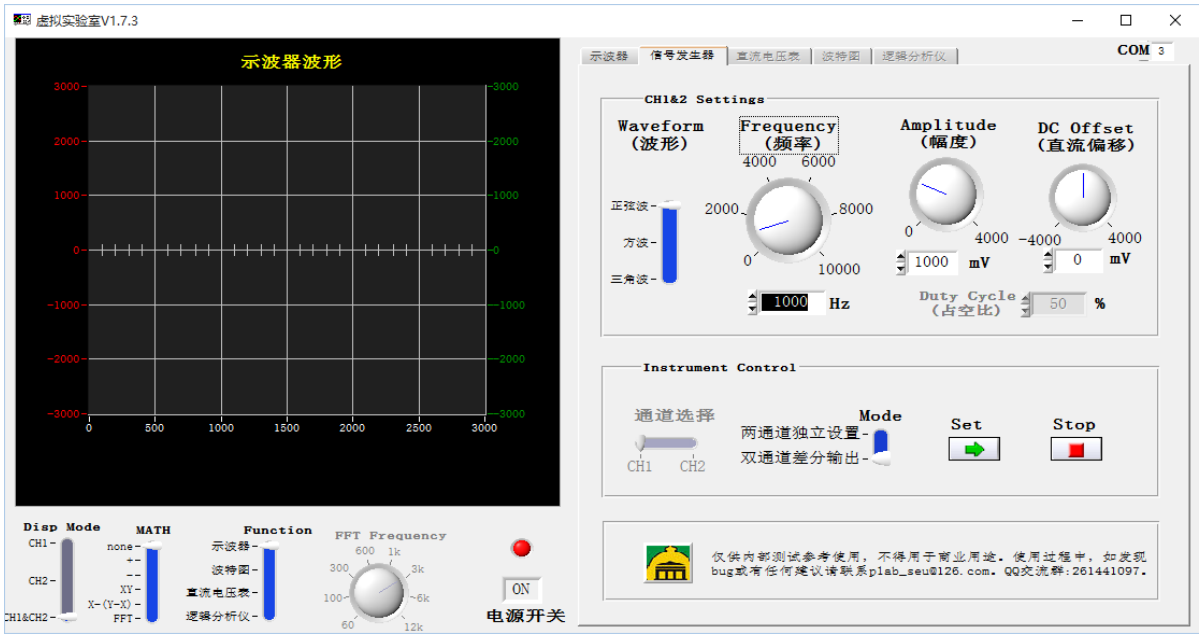
③ 时钟输出：使指定引脚输出指定频率的数字时钟；

④ 获取引脚 IO 状态：获取当前所有引脚（0-7）的输入输出状态，同时用 LED 显示各个引脚电平高低。

同时目前已经有与 PocketLab 配套使用的比较完善的 PC 端软件，其具有信号发生器、示波器、直流电压表、逻辑分析仪和波特图仪功能，其界面如下，以其中两个功能为例：



A 示波器界面



B 信号发生器界面

图 3-2 PocketLab PC 端界面

3.2 PocketLab APP 的系统结构

本项目是基于目前已有的 PocketLab 口袋实验室硬件系统，设计一套在 Android 系统下运行的软件。整个虚拟仪器系统包括硬件部分和软件部分，系统的结构如图所示：

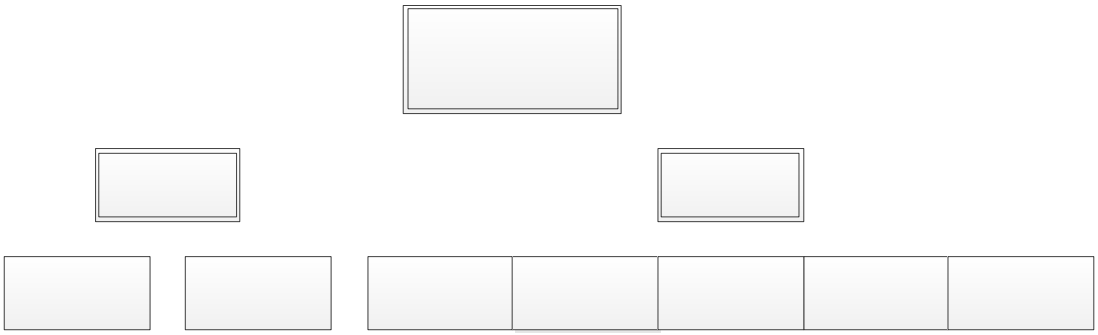


图 3-3 PocketLab 口袋实验室系统结构

其中，硬件端和软件端通过 wifi 进行通信，使用 Socket 网络连接进行数据的传输。软件端主要实现五个功能模块，即示波器、信号发生器、逻辑分析仪、波特图仪和直流电压表。我主要负责的是 Android 端软件部分的实现。

3.3 PocketLab APP 的软件架构

软件部分的整体架构，就功能上划分，可分为示波器、信号发生器、逻辑分析仪、波特图仪、直流电压表五大功能模块，如图 3-1 中所示的那样。从另一个角度来看，就软件的运行时候的层次而言，可分为网络层、数据层、界面层。其中数据层又可分为公用数据区、数据处理模块。

整个的软件架构如图 3-2 所示：

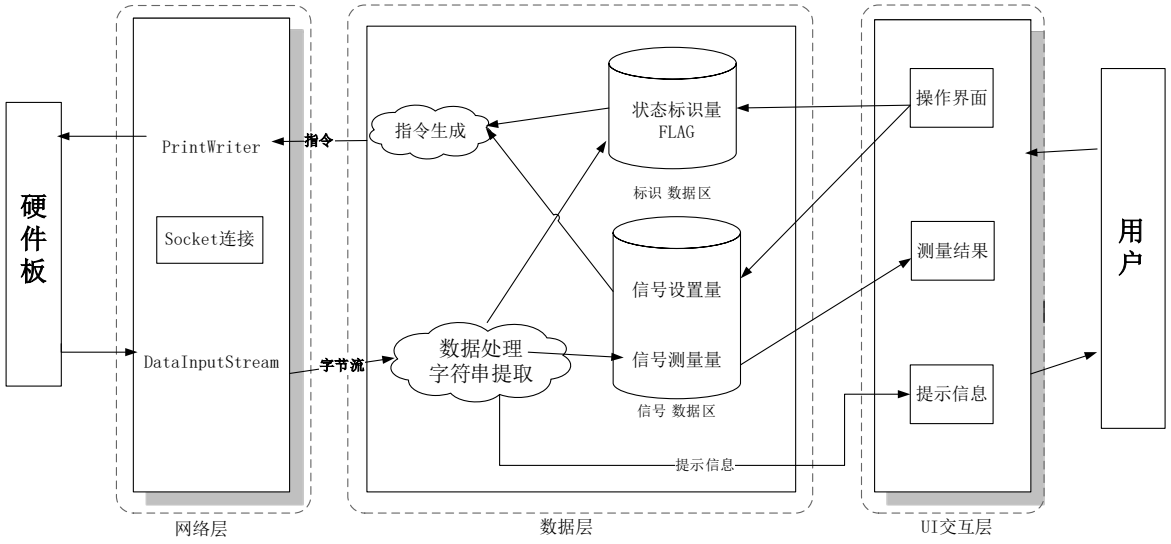


图 3-4 PocketLab 的软件架构

如图所示，网络层负责建立基础的 Socket 网络连接，并且具本的输入输出流的读写功能。数据层存储有一些全局数据，有状态的标识量，有信号的设置量，也有信号的测量所得的结果。除此之外，还包括数据处理的功能。UI 交互层负责用户与数据层的交互，接收用户的操作指令，并且将测量结果显示给用户。

当用户使用 UI 交互层的操作面板进行相关设置时，数据区的记录信号设置的相关变量随之被

修改，同时，相关的标识量也被设置。当用户使用测量或输出功能时，指令生成模块根据相应的信号设置变量和标识量，产生所需要发送给 PocketLab 硬件板的命令字符串。随后，网络层的 PrintWriter 将命令字符串发送给硬件电路的 WIFI 模块，硬件电路接收到相应指令之后，开始工作。

网络层的 DataInputStream 随时等待着接收来自硬件电路发来的信息，并根据发来的内容是对命令的回复信息还是测量数据，分别进行字符串的提取和数据的处理，处理所得的数据存入相应的数据存储区内。UI 交互层会将所得的提示信息显示给用户，或者根据采集来的数据，将测量的结果显示在屏幕上。

3.4 PocketLab 的功能模块

3.4.1 Socket 网络连接模块

Android 中提供了大量的供访问网络的 API。其中 Socket 类是 Java 中进行网络编程的核心类，它的作用是使客户端和服务器端通过 TCP 协议连接起来，并且在客户端和服务器端之间建立数据交互的通道。Socket 类除了最基本的连接服务器、发送和接受数据以及关闭网络连接操作外，还可以通过一系列的 get 和 set 方法对通讯进行调节，从而可以更好的满足客户端和服务器之间的通讯需求。

本项目主要使用 Socket 的连接、发送和接受数据的功能。Socket 的网络应用可分为客户端和服务端两部分，在本项目中，硬件系统被设定为服务端，Android 应用端则被设为客户端。

使用 Socket 连接服务器的最常用的方法，就是根据服务器端的 IP 地址和端口号进行连接。具体的方法是，将服务器端的 IP 地址（一个 String 字符串）和端口号（一个 int 数）作为参数，传递给 Socket 类的构造函数[29]，如下所示：

```
Socket socket = new Socket(IP , IPPORT);
```

由于网络的状况存在一定程度的不可预见性，所以很有可能在网络访问的时候造成阻塞。那么这样一来，如果我们在主线程/UI 线程中进行网络的连接操作，就很可能会出现假死的现象，产生非常不好的用户体验。在比较新的 Android API 版本中，试图在主线程中进行网络连接操作是不被允许的。

Android 的 Thread Policy 里对网络有许多的限制。其中，StrictMode.AndroidBlockGuardPolicy 是对 BlockGuard.Policy 的实现，在它的 onNetwork()方法里便，会根据设定的 Policy 对网络操作进行检测：

```
    public void onNetwork() {
        if ((mPolicyMask & DETECT_NETWORK) == 0) {
            return;
        }
        if ((mPolicyMask & PENALTY_DEATH_ON_NETWORK) != 0) {
            throw new NetworkOnMainThreadException();
        }
//...
    }
```

如果主线程里有联网操作，运行到这里的时候，会执行上面的 onNetwork()，如果 mPolicyMask

里设置了 PENALTY_DEATH_ON_NETWORK，那么就会抛出 NetworkOnMainThreadException 异常。

我们采用的 Android 4.x 的版本既然有此限制，那么就不能像在 Android 2.x 下开发一样，简单的在 UI 控件的响应方法中直接进行网络的连接。解决这一问题的一个方法是使用独立的线程进行网络连接。也就是说，在需要进行网络连接操作的时候，就启动一个新的线程，将网络连接的代码放在这个线程里边进行：

```
new Thread(new Runnable() {
    @Override
    public void run() {

    }
}).start();
```

由于在进行网络连接的过程中可能会抛出异常，因此，网络的连接应该放在 try-catch 语句块中：

```
try {
    clientSocket = new Socket(IP, INPORT);
} catch (IOException e) {
    e.printStackTrace();
}
```

综上，在本项目中使用如下方式建立 Socket 连接：

```
new Thread(new Runnable() {
    @Override
    public void run() {
        try {
            clientSocket = new Socket("192.168.1.1", 5001);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}).start();
```

Android 的 Socket 类中提供了 getInputStream 和 getOutputStream 这两个基本的方法，用来获取收发数据的 InputStream 和 OutputStream 对象，从而实现基本的 Socket 类的发送和接受数据的功能。InputStream 的中文意思是输入流，很显然对于客户端来说，它是服务端给它输入的数据。OutputStream 的意思是输出流，它所代表的是要给服务器端输出的数据。

本项目中使用到的 Socket 数据的收发，由 Android 应用端向硬件端发送数据时，发送的命令都是以字符串形式存在的，但是 Android 应用端接收到的来自硬件端发送的数据，也许是以字符串形式存在的回复信息，也有可能是以字节流形式发回的测量数据。因此，在本项目中，使用 PrintWriter 进行数据的发送，使用 DataInputStream 进行数据的接收，将最为方便。因为 PrintWriter 本身就是面向字符串数据，而 DataInputStream 则是面向字节流的。

使用 PrintWriter 进行字符串命令的发送的方式是：

```
    mBufferedReader = new BufferedReader(new
InputStreamReader(mSocket.getInputStream()));
    mPrintWriter.print("INSTRUCTION");
```

```
mPrintWriter.flush();
```

使用 DataIputStream 进行字节流的接收，其方法为：

```
DataInputStream in = new DataInputStream(socket.getInputStream());
int r = in.available();
if(r!=0){
    byte[] b = new byte[r];
    in.read(b);
}
```

然而对于本项目而言，由于涉及到五个功能部分。信号发生器、示波器、逻辑分析仪等各占据一个页面，在不同的页面间进行切换时，即在不同的 Activity 间进行切换时，需要保持 Socket 连接，因此可以利用 static 全局变量，在切换不同页面时将当前的已经创建的 Socket 传递到下一个页面。即在每个 Activity 如下定义：

```
public static Socket clientSocket;
```

切换页面时使用如下方法：

```
Intent intent = new Intent();
intent.setClass(this, NextActivity.class);
NextActivity.clientSocket = clientSocket;
startActivity(intent);
```

同时默认状态下，在切换页面后都会生成新的 Activity 实例，但是本项目需要每个功能的 Activity 的状态都一直保持，互相切换后还具有原来的状态，方便继续调整参数。所以就要涉及到 Activity 的 lanchmode。

Activity 有四种 launchMode，此属性需要在 AndroidManifest.xml 中配置为如下四种即：1.standard; 2.singleTop; 3. singleTask; 4.singleInstance。

接下来我简单介绍这四个 lanchMode：

1.standard

standard 模式是未给<activity>配置 android:launchMode 属性时默认的启动模式，如果手动写为 standard 也可以。此模式下一定会生成新的实例，无论当前有没有已经生成的实例。

2.singleTop

在这个模式下，当切换 Activity 时如果检测到有相对的活动实例存在于栈顶，就会不再生成新的实例并且继续使用这个已经存在的实例。相反如果相对的活动不在栈顶，那么就会生成一个新的实例。

3.singleTask

当切换 Activity 时如果发现堆栈中存在相应的活动实例，那么就会让这个实例之上的其他活动实例全部出栈，从而使其位于栈顶，接下来就会显示到幕前。

4.singleInstance

该启动模式相对于其他三个启动模式较为特别，在该模式下会应用一种新的栈结构，并把这个活动放到这个新的栈结构里，同时确保其他活动不再进入到这个新的栈结构。

那么比较四种模式，第四种 singleIntance 会比较符合本项目的需求。将每个功能的 Activity 都指定为此模式，那么每个功能都会独占一格栈，即他们只会分别生成一个实例，每次都只会调用与

修改第一次生成的实例，即完成了本项目保持状态的需求。

同时，本项目所基于的 PocketLab 硬件板存在一定的缺陷：即当使用 Socket 连接上硬件板之后，如果在客户端将 Socket 断开，那么如果想重新连接，则只有将硬件板重新上电。

因此本项目需要做的就是，在不同的 Activity 间保持同一个 Socket 连接。也即是说，Socket 连接需要建立在独立于 Activity 的一种能在程序运行的整个周期内保持活动的机制之上。

3.4.2 数据的处理模块

硬件电路端与 Android 应用端通过 Socket 进行数据的传输，其中，由 Android 端发送给硬件电路端的，都是字符串形式的命令。比如，当在程序中需要发送信号来确定硬件电路是否能正常工作、Socket 连接是否正常建立，需要与硬件端进行握手操作。由 Android 客户端发送一串命令给硬件电路端，硬件端收到这一询问命令之后，返回一串信息，表示自己正处于正常工作状态。其中，发送和接受到的字符信息分别为：

发送：HELLO\r

回复：I am PocketLab V2.0.0! \r\n（共 24Byte）

我们在 Android 程序中，分别使用 InputStream 和 PrintWriter 类来实现数据的收发功能。其中，用来发出数据的 PrintWriter 本身就是用来操作字符流的，而用来接受数据的 InputStream 则是面向字节流的。也就是说，Android 应用程序接受到的来自硬件电路发送来的数据，是以一串字节流的形式存在的。而就其内容而言，可分为两种：一种是如“I am PocketLab V2.0.0! \r\n”一样的回复信息，表示硬件电路正确接受到了命令；另一种则是返回的对信号的测量数据。比如 PocketLab 的示波器采集命令，返回的数据格式为：

a） 首先返回标志信息 mode（char 型，1 字节），固定为十六进制 A3。

b） 然后返回实际设定的采样间隔十六进制 0A 00（int 型，共 2 字节，单位 us），表示采样周期为 10us。

c） 接着返回 ch0 ch1 在设置的档位下的偏移值 offset（两个 int 型，共 4 字节，单位 mV，通道 0 在前）。

d） 然后是两通道的实际增益值 gain（两个 float 型，共 8 字节），数据顺序是低字节在前。

e） 然后返还数据。数据格式为：ABABABABAB……其中每个 A 和 B 为一个字符（每个通道每个采样值 1 字节），分别对应通道 0 和通道 1 的 8bit 采样值。

从上述的返回内容，可以看到，此处涉及到从 byte 字节流形式向 String 字符串形式、int 整型数形式、float 浮点数形式的转换。

Byte 是 Java 语言中最小的整数类型，它在内存中占据一个字节，也就是 8 位数据。在操作一些比较小的数据的时候，使用 byte 则可以节省空间。同时，对于一些以字节形式存在的数据流进行操作的时候，使用 byte 就非常方便。在 Java 中，byte 类型的取值范围是[-128,127]。

与 C 和 C++不同，Java 中 int 型数据时以 4 个字节存储的。如果要想将一个 byte 数据转存为一个 int 型数据，如果使用直接的赋值操作，其结果将是错误的。因为 byte 类型的取值范围从-128 到

127，比如当采集到某一个 byte 型变量，当它存储的内容为 11111111，即 FF 时，我们知道这个数想要表达的值是十进制的 255。但是，在 Java 中，会将它解释为-127。假如将这个值以直接的赋值操作形式赋给一个 int 型变量，那么赋值操作之后，int 型变量中存储的将是-127，而不是我们需要的 255。

同时，由于 C 语言中 int 型数据占 2 个字节，而 Java 语言中 int 型占 4 个字节。正是因为这些问题的存在，我们需要考虑将 byte 型数据转存为 int 型数据的方法。直接复制不可行，我们需要使用基本的按位操作。实现方法如：

```
public int byte4ArrayToInt(byte[] b) {
    return   b[0] & 0xFF |
           (b[1] & 0xFF) << 8 |
           (b[2] & 0xFF) << 16 |
           (b[3] & 0xFF) << 24;
}
```

此方法的功能，是将一个由 4 个 byte 型变量表示的整型数用一个占 4 字节的 int 型变量表示。因此该方法的参数是一个 byte 型数组，返回值是一个 int 型整数。

例如，PocketLab 发送两字节 int 十六进制 CF 18 给 Android 端，对应的 int 数是 0x000018CF，即十进制的 6351.

由 byte 数据向 String 字符串形式转变，可以调用 String 类的成员方法[30]：

```
String result = new String(bytes,"utf-8");
```

在 Java 中，float 型数据与 int 型数据一样，占据 4 个字节的存储空间。再讲 byte 型数据转换为 float 型之前，首先对 float 型数据的存储方式作一介绍：

Float 型数据在存储方式上遵从 IEEE R32.24 规范。它在内存中的存储可以分为这三个部分[31]：

1. 符号位（Sign）：0 代表它是正数，1 代表它是负数
2. 指数位（Exponent）：指数部分存储的是科学计数法中的指数部分，它采用的是移位存储。
3. 尾数部分（Mantissa）：尾数部分

Float 的存储方式如图所示：

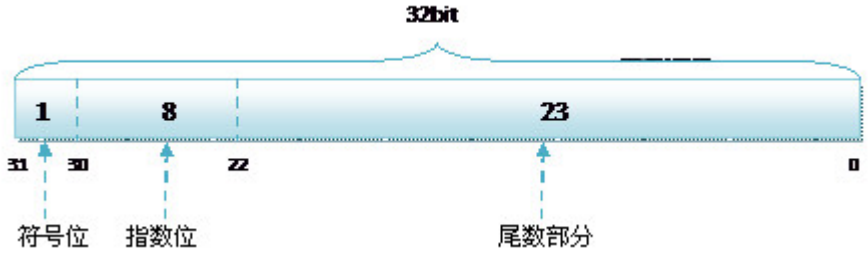


图 3-5 float 型数据的存储方式

上图便是以二进制形式存储在内存中的 float 型数据。由 PocketLab 硬件端发送给 Android 端的 float 型数据，便是以这样 32 位数据、4 个字节（低字节在前、高字节在后）的形式发送过来。

在 Java 语言中，由于 float 型和 int 型都是占的 4 个字节，float 型数据可以以 32 位 4 个字节存储，故而一个 float 数据也就可以以一个 int 型数据的形式存储。

Java 中提供了将以 int 形式存储的 float 数据转化为 float 形式的方法，即 Float 类的成员方法：

```
Float.intBitsToFloat(int NUM)
```

该方法的参数为一个 int 型数据，返回值为一个 float 型数据。

再结合前面介绍的由 byte 型数据转化为 int 型数据的方法，因此，实现提取由 PocketLab 发送给 Android 应用端的字节流中的 float 数据的方法为：

```
bytes[0] = NUM_byte1;
bytes[1] = NUM_byte2;
bytes[2] = NUM_byte3;
bytes[3] = NUM_byte4;
NUM_float = Float.intBitsToFloat(getInt(bytes);
```

当 Pocketlab 发出一个 float 型数据，实际表示为从低到高四个字节 98 D1 3F 42，Android 应用程序接受到这四个字节之后，将之依次存储在 byte 型数组中。调用 float.intBitsToFloat(int NUM)方法即可得出它所表示的实际的值 47.95468。

3.4.3 数据接收线程与 UI 的更新

Java 是支持多线程技术的。为了达到更好的使用效果，这里将数据的发送与数据的接受独立开来——即在操作按钮的按钮的响应方法中发送命令，而使用独立的一个线程来完成数据的接受功能。

线程是 Java 语言的关键技术之一，在一个程序中，可以包含一个或者多个线程[32]。以往的单线程程序中，代码按照调用顺序一次执行，如果执行过程中涉及一些耗时的操作，就必须等待该操作完成之后，再继续执行下一步的而操作。使用多线程技术，就可以开辟一个单独的执行分支去完成这些耗时的操作，当前的主线程则可以继续执行其他的代码。

实现多线程的方法之一，就是使用 Java 语言提供的 Runnable 接口。Runnable 接口中定义了 run()方法。Runnable 接口的定义代码为：

```
package java.lang;
public interface Runable{
    public abstract void run();
}
```

可见，由于 run()方法是 abstract 类型的，所以在实现该接口的同时。必须先实现接口中的 run()方法。

```
private Runnable    mRunnable   = new Runnable()
{
    public void run()
    {
        while (true)
        {
            //数据接收
        }
    }
};
```

在上述的代码中，我们在 run()方法中建立了一个条件始终为真的 while 循环，因此这个循环将是无限执行的。使用这样的方法，只要这个线程被启动，那么 run()方法里边的这部分代码将会一遍又一遍的不断循环执行下去。

为了启动线程，实现 Runnable 接口的对象需要传递给 Thread 类的构造方法，通过 Thread 的构造方法去创建线程类，也就是说，Runnable 接口的实现方法需要传递给 Thread 类的实例对象，才能够启动线程。Thread 类有很多构造方法，其中常用的接受 Runable 接口参数的构造方法如：

```
Thread(Runnable target)  //使用Runnable接口实现对象创建线程对象
Thread(Runnable target, String name)  //创建线程对象，并指定线程名称
```

所以，在主线程中调用该方法，就可以启动实现的线程：

```
mThread = new Thread(mRunnable);
mThread.start();
```

由于命令的发送和数据的接收是分开的，为了在数据的接收线程中根据发送命令的类型来根据情况处理接受到的数据，所以在程序中，设置一些标识量，用来标识命令的类型。如：

```
private int  mode = 0;
```

在发送命令的同时，根据命令的类型，为该标识量赋值。在接收线程中，根据该标识量的值，来选择对数据进行处理的方法。

然而，在这个线程中，我们可以对传输来的数据进行读取、处理，但是不能够将处理之后的数据显示在屏幕上，也就是说，在这个线程中，我们不能够对界面进行更新。这是由 Android 系统的特性决定的，在 Android 中，如果要对 UI 界面进行更新，只能在主线程中进行这一操作。所以，我们在设计的时候应该按照这一规则，让主线程负责对 UI 控件的创建、显示和更新操作，在子线程中可以对数据进行处理，如果需要对界面进行更新，则由子线程向主线程发送消息，通知主线程对 UI 新型更新[33]。

Android 中的 Handle 机制实现了在子线程和主线程之间进行消息传递的功能。Android 提供了 Handler 类，它主要的作用正是在建立许多线程之间的消息与计划任务的传递[34]。可以使用 Handler 类中的 sendMessage()方法在不同的线程间发送消息，使用 handleMessage(Message msg)方法在主线程中接受消息，并对之进行响应。

比如，当我们在 mThread 线程中对数据已经进行了处理，需要将其显示在界面之上。可以调用 sendMessage()方法来发送消息：

```
drawHandler.sendMessage(msg);
```

在此之前我们需要创建一个用来接受消息的 Handler 对象，并实现 handleMessage(Message msg)方法，比如：

```
private Handler drawHandler = new Handler()
{
    @Override
    public void handleMessage(Message msg)
    {
        switch (msg.what){
            case 1:
                DrawChart();
                break;
            case 2:
                ChangeChartY();
        }
    }
};
```

在 handleMessage 方法中，我们就可以进行界面的更新操作。

以上介绍了用来接受数据的线程的实现方法，然而，该线程被启动之后，它会在后台一直运行。即使启动它的这个 Activity 已经被销毁，它也并不会被彻底杀死，反而会成为幽灵线程。如果我们切换到一个新的 Activity，在这个新的 Activity 中重新启动一个数据接收线程——那么两个线程将同时存在，导致数据的接受出现问题。这是由于 Android 的特性决定的，在 Activity 中开启的子线程并不会随着 Activity 的 destroy 而自动关闭，所以必须去手动的关闭子线程。在本项目中，该线程是以一个条件始终为 true 的 while 循环的形式存在的，我们只需要添加一个 boolean 标识，通过这种方式让子线程结束运行。

我们首先添加一个 boolean 标识量：

```
private boolean stop = true;
```

再将数据接收线程中的循环条件改为：

```
private Runnable    mRunnable   = new Runnable()
{
    public void run()
    {
        while (!stop)
        {
            //数据接收
            if(数据接收完毕){
                stop = true;
            }
        }
    }
};
```

当一次收发过程中需要的数据接收完毕即将 stop 置为 true，线程中的循环被打破，该线程中的语句便不会再一直被运行下去。

3.5 PocketLab 应用程序的屏幕适配

与 iOS 开发不同的是，Android 开发需要面对的是各种各样有不同厂家生产的设备。各种 Android 设备的屏幕尺寸、分辨率都不相同。在进行 Android 开发时，必须考虑 UI 的适配[35]。

与此相关最大的两个参数分别是：屏幕分辨率（Screen Resolution）和屏幕分辨率（Screen Density）。其中，指的 Android 设备屏幕的横纵两个方向的最大像素点个数，屏幕密度指的是单位面积内的像素点个数。

Android 设备的屏幕尺寸根据分辨率大小的不同，可分为：small、normal、large、xlarge，分别表示小屏、中屏、大屏、超大屏。其中：

- xlarge 屏最小为 960dp * 720dp
- large 屏 最小为 640dp * 480 dp
- normal 屏最小 470dp * 320dp
- small 屏最小为 426dp * 320dp

根据单位像素点的像素数，即屏幕密度，Android 设备的屏幕可分为：ldpi、mdpi、hdpi、xhdpi，

它们的标准值分别是：120dpi，160dpi，240dpi，320dpi。

Android Studio 中的一个 Android 项目工程文件的 res 目录下，在资源目录中加上一些限定字符，就可以为资源指定适用的特定平台。如图所示：

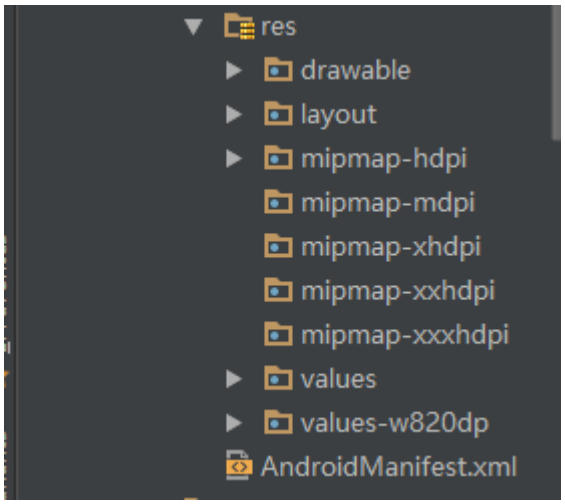


图 3-6 res 子目录

3.6 PocketLab 应用程序的开源项目应用

3.6.1 开源的定义

“开源”的意思就是开放源码，它是从英语“Open Source”翻译过来。任何软件都是先通过代码写成，然后再编译成可以使用的应用。但是开源这个软件，就代表着这个软件的源代码会公布给所有人，那么所有人都可以使用这些代码，同时也可以改进优化这些代码。因此开源意味着自由，自由的力量可以带来更多的创新。Linux、Java 和 Android 都是著名的开源系统。

3.6.2 开源社区

开源社区在近些年得到了迅猛发展，很多大公司纷纷涌入开源社区，比如 Google、Facebook、Square 都贡献了很多卓越的开源项目在社区，出乎意料的是一些较封闭的公司比如 Apple 和 Microsoft 都加入了开源社区。可以预见的是开源很可能是软件将来主要的发展方向。

那么说到开源社区，Github 首当其冲，全世界最棒的程序员和最好的开放科技组织都汇集在这里，它是现在最受欢迎规模最大的开源社区。本文下面提到的开源项目都来自于 GitHub。

3.6.3 用开源项目的

DRY 是程序开发领域一个著名的原则，全称是“Don’t repeat yourself”，中文意思简单来讲就是“不要重复造轮子”。在当今这个飞速发展的互联网领域，效率就是生命，目的让使用者不重复造轮子的开源项目就显得尤其重要，应用开源项目可以让极大的提高开发效率，节省大量时间和人力资源。

3.6.4 使用的开源项目介绍

3.6.4.1 pedant / sweet-alert-dialog

软件在操作的过程中需要提示信息或确认操作，即交互后显示一些额外的信息。Android 自带的 toast()方法过于简单，显示也不够美观，于是我选用一个开源的提示窗。

pedant / sweet-alert-dialog 是 Android 版的 SweetAlert，清新文艺，快意灵动的甜心弹框。可以自定义各种显示形式和功能: 只显示标题，显示标题和内容，异常样式，警告样式，成功完成样式。比如软件中成功连接至 pocketlab 硬件的提示：

图 3-7 成功连接至 pocketlab 硬件的提示

下面以成功样式为例，介绍此项目的使用方式：

```
new SweetAlertDialog(this, SweetAlertDialog.SUCCESS_TYPE)
    .setTitleText("Good job!")
    .setContentText("You clicked the button!")
.show();
```

`SweetAlertDialog.`后面即使定义显示的类型；

`setTitleText()`定义标题内容；

`setContentText()`定义主要内容。

3.6.4.2 eluleci/FlatUI

eluleci/FlatUI 是一个开源库，可以让原生的 Android 控件有更好的定制和外观，基本效果如下图：

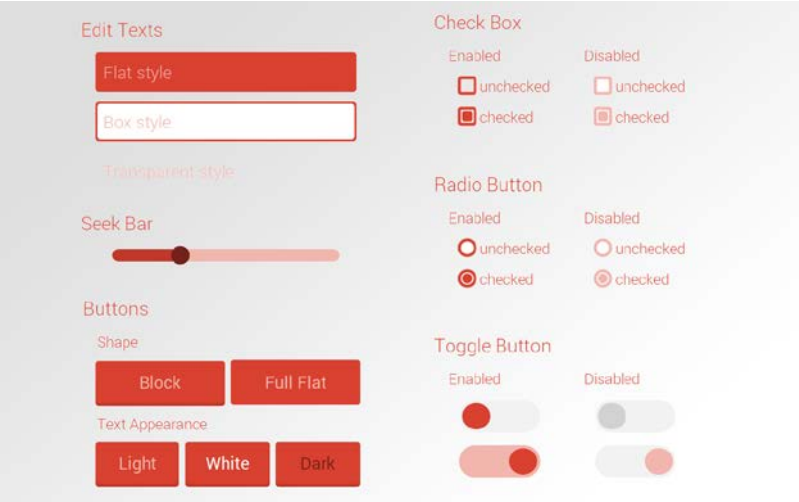


图 3-8 eluleci/FlatUI 效果图

本项目用此开源库定义了一些按钮和单选框，使其具有更好的外观和效果。其使用也非常简单：

首先在布局文件中声明：

```
xmlns:flatui="http://schemas.android.com/apk/res-auto"
```

然后是控件的通用定义方式：

```
<com.cengalabs.flatui.views.SomeFlatView
    ...
    flatui:fl_theme="@array/sand"
    flatui:fl_textAppearance="dark"
    flatui:fl_fontFamily="roboto"
    flatui:fl_fontWeight="light"
    flatui:fl_fontExtension="ttf"
    flatui:fl_borderWidth="2dp"
    flatui:fl_cornerRadius="5dp"
    flatui:fl_size="20dp" />
```

最后在 Activity 中的 onCreat()方法中定义控件的主题：

```
FlatUI.initDefaultValues(this);
FlatUI.setDefaultTheme(FlatUI.DEEP);
```

3.6.4.3 SimonVT/android-menudrawer

因为 Android 手机设备通常屏幕较小，只有 4-6 寸，但是本项目虚拟仪器软件有多个功能可能需要随时切换，又需要设置很多参数，再加上需要显示很多数据或者图表，如果将这些内容全部显示在一个界面，界面将会非常局促和拥挤，既不美观也不利于操作。所以我选择使用滑动菜单来解决这一问题。可以将接收到需要观察的数据显示在主要界面上，需要切换功能时侧滑调出菜单，非常方便。

SimonVT/android-menudrawer 就是一个经典的开源滑动菜单项目，它支持滑动屏幕边缘或者点击按钮调出菜单，同时具有如下特征：

- 菜单可以从四个方向边缘调出；
- 支持附加一个总是可见的，非可拖动菜单；
- 菜单可以包括主要内容和整个窗口；

- 支持通过滑动边缘，或者滑动整个屏幕，或者禁止滑动调出；
- 可以再 XML 布局中使用；
- 有指示器表明当前显示的是哪个屏幕。

本项目就主要使用了 SimonVT/android-menudrawer 侧滑调出菜单：

```
public class SampleActivity extends Activity {

    private MenuDrawer mDrawer;

    @Override
    protected void onCreate(Bundle state) {
        super.onCreate(state);
        mDrawer = MenuDrawer.attach(this);
        mDrawer.setContentView(R.layout.activity_sample);
        mDrawer.setMenuView(R.layout.menu_sample);
    }
}
```

在需要使用侧滑菜单的 Activity 的 onCreate()方法中，用 `setContentView()`定义主界面内容，`setMenuView()`定义菜单内容。效果如下图：

图 3-9 SimonVT/android-menudrawer 的效果图

有时一个界面上需要不止一个侧滑菜单，比如本项目的示波器界面，主界面显示波形，同时需要一个侧滑菜单切换功能，另一个改变参数。但是 SimonVT/android-menudrawer 不支持同时存在两个侧滑菜单，于是我同时引用另一个滑动菜单菜单的开源项目 jfeinstein10/SlidingMenu 来支持另一个菜单。jfeinstein10/SlidingMenu 与 SimonVT/android-menudrawer 比较相似，就不在具体介绍。

3.6.4.4 PhilJay/MPAndroidChart

示波器模块的主要功能就是显示波形，所以画图的方法十分重要。项目的开始我计划使用 Java 自带的 onDraw()方法来进行图像绘制，但 onDraw()方法主要通过依次指定坐标来画图，示波器的波形有大量数据，而且更新非常快，如果还直接用 onDraw()方法将会效率很低，最后显示的图也会不够美观。于是我决定用一个开源的制图表项目：PhilJay/MPAndroidChart。

MPAndroidChart 是一个使用非常简单功能非常丰富的图标库，他支持线状图、柱状图、蜘蛛网状图、气泡图、散点图、烛状图和饼状图。同时有很多的交互功能和漂亮的动画效果。以线状图为例，效果如下：

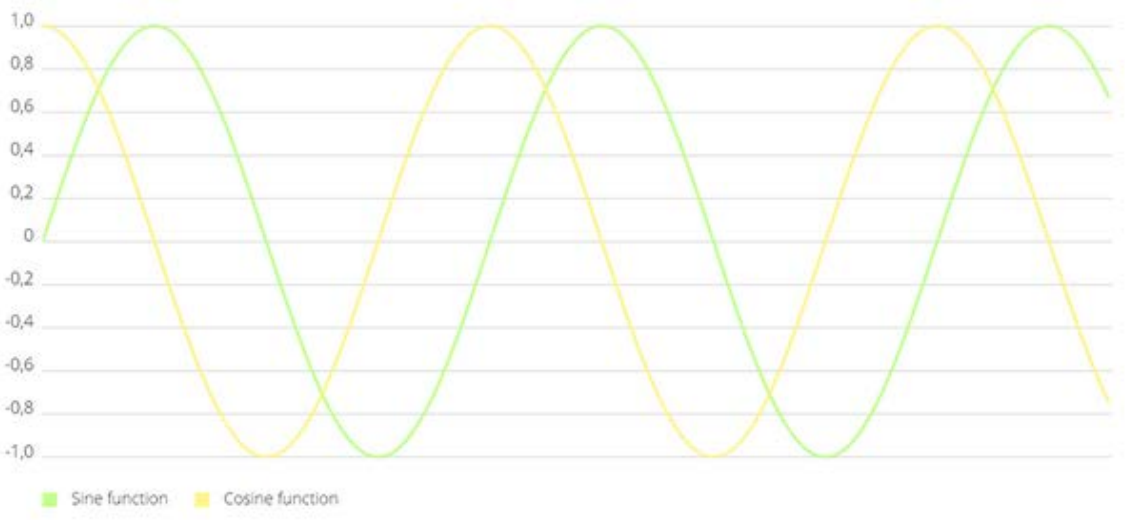


图 3-10 PhilJay/MPAndroidChart 线状图效果图

因为 PhilJay/MPAndroidChart 的定义以及使用比较复杂，在此就不再详细介绍。

3.6.4.5 code-troopers/android-betterpickers

本项目的各个模块都需要一些参数的设置，其中不乏数字输入，比如信号发生器需要设置信号的频率，幅度，如果单纯使用数字输入控件未免太过枯燥。因此我选用了一个选择器开源项目 code-troopers/android-betterpickers。

ode-troopers/android-betterpickers 是一个强大的数据选择器，可以选择时间、日期、数字或者其他各种数据，其效果如下图：

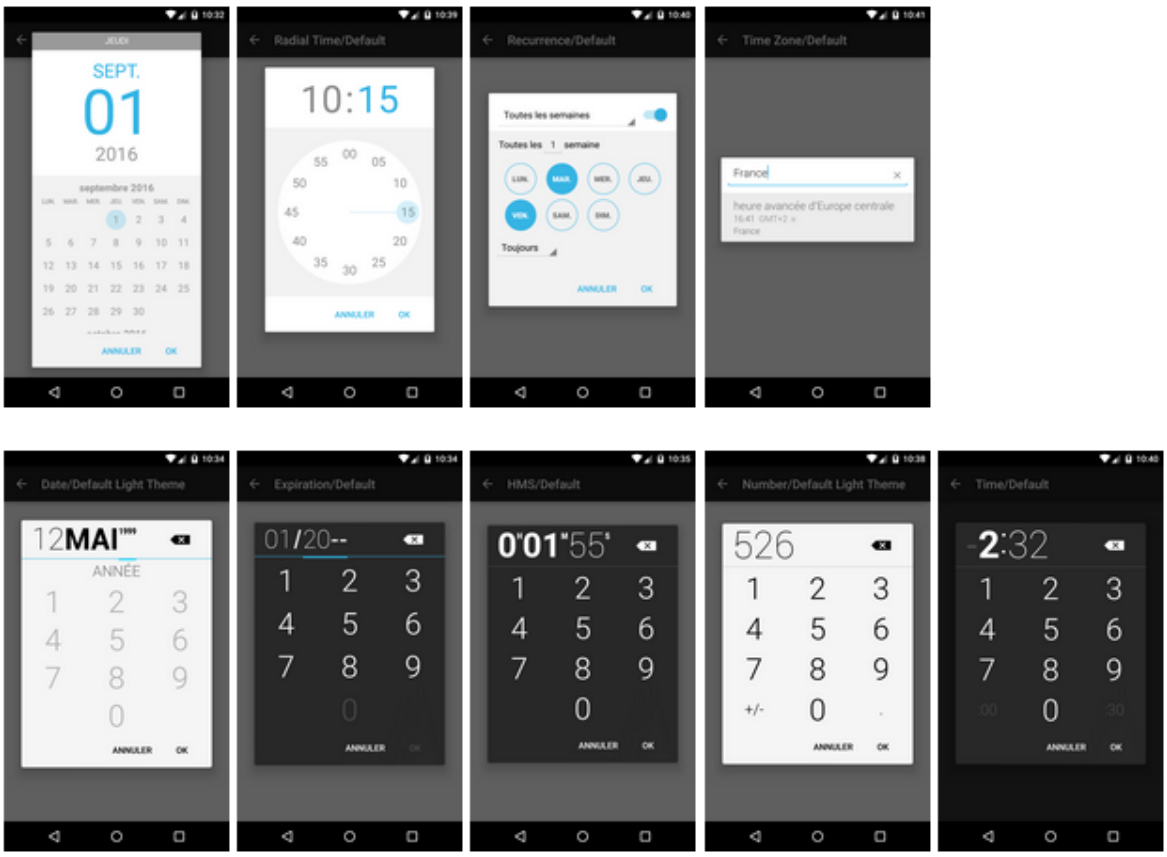


图 3-11 code-troopers/android-betterpickers

本项目主要使用了数字选择功能。其使用方法如下（用按钮唤起）：

```
button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        NumberPickerBuilder npb = new NumberPickerBuilder()
                .setFragmentManager(getSupportFragmentManager())
                .setStyleResId(R.style.BetterPickersDialogFragment)
                .setMaxNumber(new BigDecimal(max))
                .setMinNumber(new BigDecimal(min));
                .setLabelText("LBS.");
        npb.show();
}
});
```

`setStyleResId()`定义显示的主题风格；

`setMaxNumber()`定义输入的最大值；

`setMinNumber()`定义输入的最小值；

`setLabelText()`定义标签内容，可用来显示数据单位。

3.6.4.6 gzu-liyujiang/AndroidPicker

项目中不仅需要数字输入来定义一些参数，同时需要一些通过选项的方式定义参数。比如示波器中的 Volts 和 TIME，都具有有限个选项。因此我选用了另一个开源数据选择器项目。

gzu-liyujiang/AndroidPicker 是一个选择器类库，包括数字选择器、单项选择器、日期选择器、文件选择器、城市选择器、目录选择器、时间选择器、星座选择器、颜色选择器、生肖选择器等，可自定义顶部及底部界面，可自定义窗口动画。它与 code-troopers/android-betterpickers 主要不同之处在于前者主要通过滑动滑轮选择选项。本项目主要使用了其中的单项选择器，效果如下图：

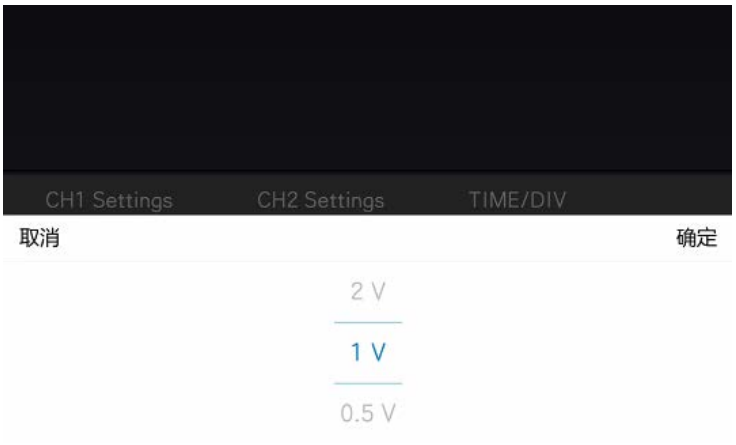


图 3-12 gzu-liyujiang/AndroidPicker 单项选择器（Volts 选择）效果图

其使用方法如下：

```
        OptionPicker picker = new OptionPicker(this, new String[]{
                "第一项", "第二项", "第三项"
        });
        picker.setOffset(2);
        picker.setSelectedIndex(1);
        picker.setTextSize(11);
        picker.setOnOptionPickListener(new
OptionPicker.OnOptionPickListener() {
            @Override
            public void onOptionPicked(String option) {
                showToast(option);
            }
        });
        picker.show();
```

`new String[]`中内容即为各个选项内容；

`setOffset()`定义选择窗口的高度；

`setSelectedIndex()`定义当前选中的为第几个选项(从第 0 个开始)；

`setTextSize()`定义字体大小；

`onOptionPicked(String option)` 定义选中某个选项后进行的功能。

3.7 本章小结

本章对 Android 系统下设计的 PocketLab 应用程序的系统结构和层次架构进行了介绍，并对几个主要的功能模块，即网络的连接与保持、数据的收发与处理、UI 的响应与切换、使用的开源项目等进行了简要的介绍与说明。在这几个基本的功能模块的支持下，示波器、信号发生器等功能才能够得到实现。

第4章 PocketLab 软件的具体功能实现

4.1 软件主界面

4.1.1 功能介绍

软件的主界面是软件启动后出现的第一个界面，本项目的启动界面负责的功能即是：

- 对用户进行引导和帮助；
- 与 pocketlab 硬件端成功建立连接；
- 提供其他 5 个具体功能的入口。

4.1.2 界面设计

Android 应用程序的界面设计使用.xml 文件描述，可以选择使用 LinearLayout、RelativeLayout、TableLayout、FramLayout 等多种布局方式，Button、ImageButton、Switch、RadioButton 等多种组件，配合.java 代码中的调整与设置，以实现复杂的界面显示效果。

软件主界面的界面设计，采用 RelativeLayout 相关布局，使用 Button、ImageView、TextView 等常用组件进行设计。并使用了 pedant / sweet-alert-dialog 和 eluleci/FlatUI 两个开源项目。

软件的主界面如下图：

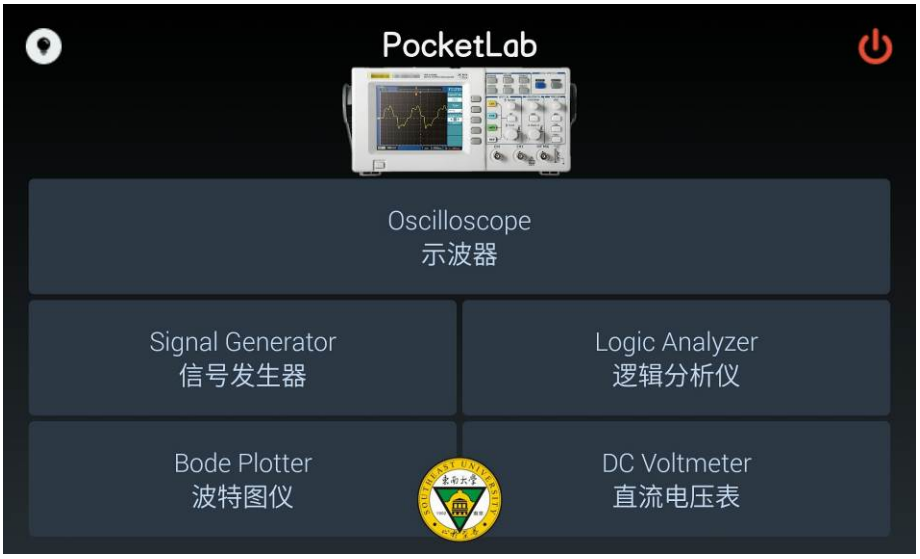


图 4-1 软件主界面

点击左上角的灯泡按钮会显示帮助与提示

图 4-2 帮助与提示

点击右上角电源按钮则会与 pocketlab 建立 socket 连接，同时发送握手命令，若成功收到正确回复，则提示连接成功，并且电源标志变绿。

下面 5 个按钮分别对应本项目的 5 个具体功能，当软件端未与 pocketlab 成功建立连接后点击这些按钮会提示先进行连接，并且不会转到相应功能。只有当成功连接后，点击这些按钮才会跳转到具体功能。这样设计是为了开始具体功能后，不会因为未建立正确连接发生意外错误。

4.2 示波器功能

4.2.1 功能介绍

示波器的用途十分广泛，它能够把看不见的电信号转化为肉眼可识别的图像信号，以便于人们观察研究其电特性。它根据测量到的被测信号的瞬时值，将信号的幅度随时间变化的波形曲线显示在屏幕上[36]。

PocketLab 的示波器部分支持以下功能：

- 100k samples/s，8bit 采样
- 可变采样率（≤50 k samples/s），8bit 采样
- 可变采样率（≤50 k samples/s），12bit 采样
- DC 耦合，AC 耦合

在开启示波器的时候，可以通过增益档位序号设置信号的最大输入幅度，在不同最大输入电压下，示波器的分辨率也有所不同。本项目使用的模式为可变采样率（≤50 k samples/s），8bit 采样，为了保证性能，实际使用的采样率为 50 k samples/s。具体通信协议见附录 I。

4.2.2 界面设计

示波器的界面设计，采用 RelativeLayout 相关布局，使用 Button、ImageView、TextView、Switch 等常用组件进行设计。并使用了 pedant/sweet-alert-dialog、SimonVT/android-menudrawer、

jfeinstein10/SlidingMenu 、 PhilJay/MPAndroidChart 、 code-troopers/android-betterpickers 和 gzu-liyujiang/AndroidPicker 六个开源项目。

示波器界面效果如下图:

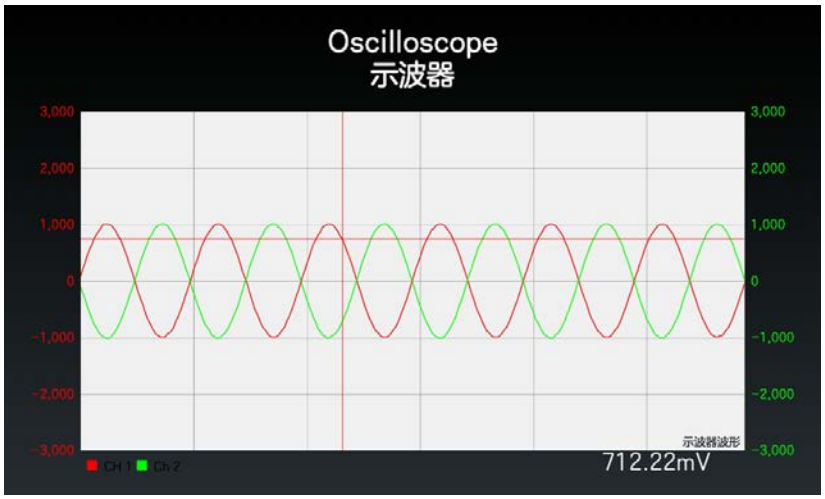


图 4-3 示波器主界面

波形图通过 PhilJay/MPAndroidChart 开源项目绘制。红绿两条图线分别代表 CH1 与 CH2 的输入波形，左右分别是两条线的坐标轴，并且显示的图线与坐标轴都会随着数据输入的而不断快速更新。当点击任意一条线上一点时就会用一个十字线将其选中（如上图），非常人性化。并且此时右下角会显示当前选中点的电压，方便查看。

示波器主界面基本只负责显示，调整参数通过在屏幕下边缘向上划，即会拉出参数调整菜单，如下图：

图 4-4 参数调整菜单

参数调整菜单中可以选择两个输入的垂直位移，幅值和耦合方式，以及整个图中每格所代表的时间（TIME/DIV）。通过点击旋钮即可唤出数据输入窗口如下图：

图 4-5 垂直位移输入

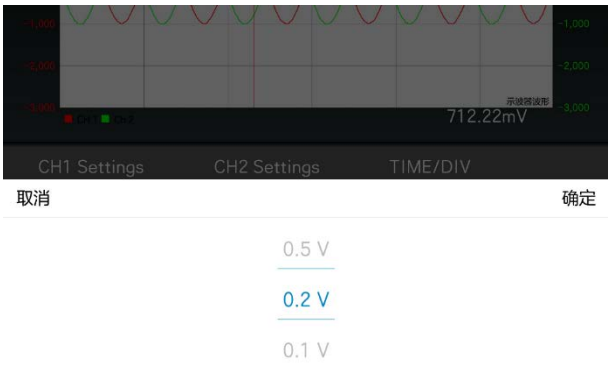


图 4-6 TIME/DIV 输入

点击绿色三角示波器开始采样并显示波形，点击红色方块示波器停止工作。

滑动屏幕的左边缘或点击移动设备上的回退按钮可以调出软件的菜单，以便切换至其他功能（其他功能界面也同样具有此菜单，下面就不再一一介绍），如下图：

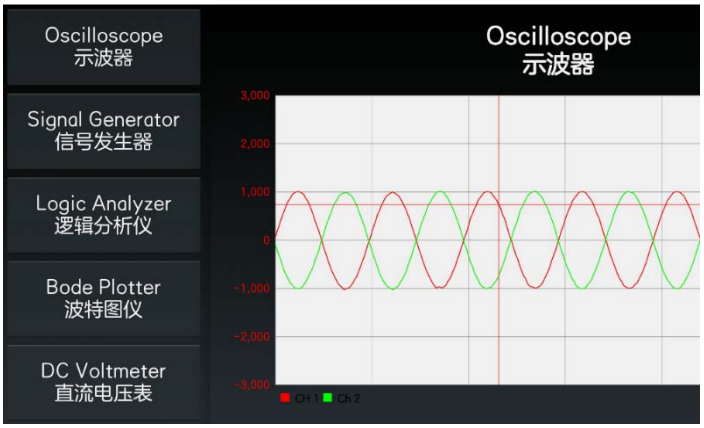


图 4-7 左侧滑菜单

4.2.3 工作流程

示波器工作流程图如下：

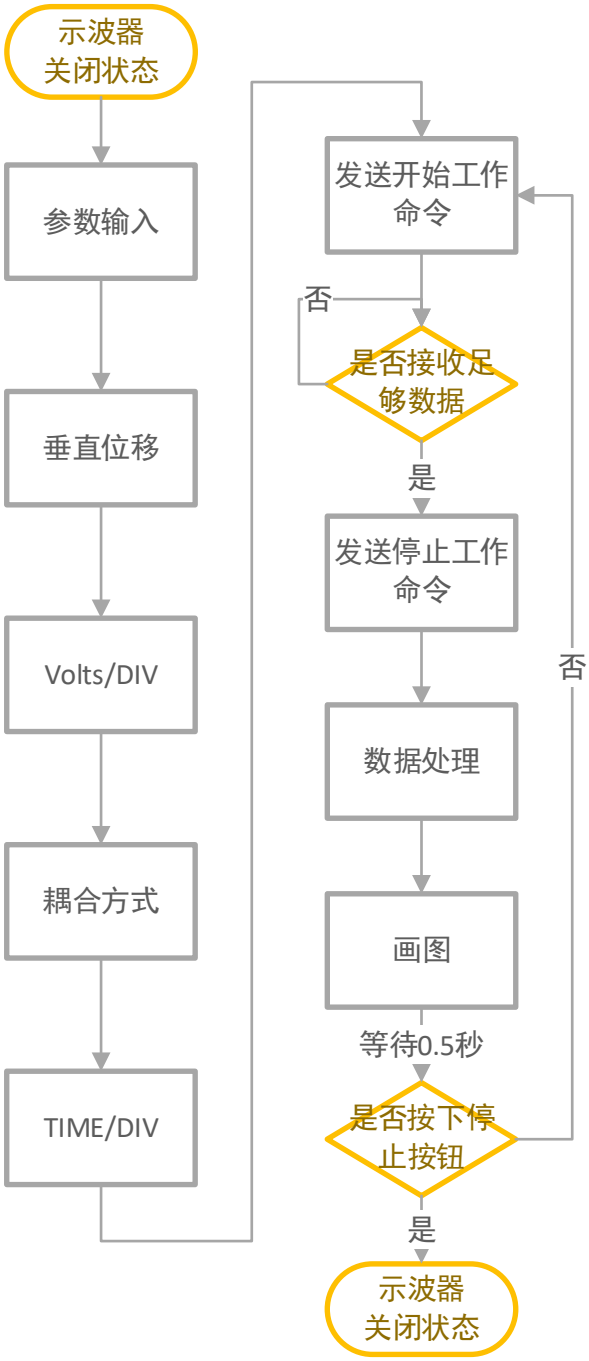


图 4-8 示波器工作流程图

当参数设置完毕，开启示波器后，硬件端会高速发送来大量数据，但是软件端不可能全部接收并且以如此高的频率刷新绘图，因为移动端可能没有这么强的运算功能，并且这样做也过于占用内存容易造成卡顿。因此本项目中采用的方法是：根据一次画图需要的数据量，当示波器开始工作软件端接收到足量的数据，发送停止命令，硬件段就会停止发送数据。此时再根据接收到的数据处理

后画图。等待 0.5 秒后判断是否需要继续工作，如果需要继续工作，则向硬件段发送“CON_OSC2↙”命令，此命令会让硬件端根据之前的参数设置继续工作。

4.2.4 程序具体介绍

在 Oscilloscope 类中，实现了：

- ListenCoupling1()，ListenCoupling2()方法：分别监听两个通道耦合方式的改变（通过点击 Switch 控件），并记录两个通道分别选择的耦合方式。当示波器在工作状态下时，任一耦合方式改变后都会在下一个接收数据绘图周期前向硬件段重新发送设置参数并开始工作的指令。
- Click_volts(View view)方法：监听两个通道 Volts/DIV 参数的改变，当点击此参数的旋钮时会唤起 gzu-liyujiang/AndroidPicker 中的单项选择，确定选项后记录下两个通道选择的 Volts/DIV。当示波器在工作状态下时，任一 Volts/DIV 改变后都会在下一个接收数据绘图周期前向硬件段重新发送设置参数并开始工作的指令。
- Click_TIME(View view)方法：与 Click_volts(View view)方法类似，监听 TIME/DIV 参数的改变。但是当示波器在工作状态下时，TIME/DIV 改变后都会在下一个接收数据绘图周期前重新设置绘图 X 轴的参数，并改变需要接收的数据量。
- Click_vd(View view)方法：监听两个通道的垂直位移参数，当点击此参数的旋钮时会唤起 code-troopers/android-betterpickers 中的数字选择器。确定数字输入后记录下两个通道选择的垂直位移。当示波器在工作状态下时，任一垂直位移的改变后都会在下一个接收数据绘图周期前改接收变数据处理的方式。
- Click_runOsc(View view)与 runOsc()方法：点击绿色三角按钮后根据当前的参数设置向硬件端发送开启示波器的命令，同时开始数据接收进程，准备接收数据。
- ReceiveThread 线程：开启此线程后软件端开始接收数据，判断接收到足量的数据后发送停止命令。然后处理数据，并调用画图画图方法。
- byteArrayToInt(byte[] b)，getAD(byte[] b,int n,int ch)，getVinput(float gain, int offset, int[] AD, int vd)，FindZero(float[] input1, float[] input2, int n)方法：都是用来处理接收到的数据，一个是将 byte 数组转为 int 型数；第二个是获得 AD；第三个根据增益、偏移量以及 AD 获得信号真正的电压；画图时默认的触发方式是根据 CH1 的上升沿，第四个方法是找到 CH1 的上升沿，然后截取数据。
- DrawChart()方法：根据处理后的数据，设定 X 轴 Y 轴参数，以及画图的其他参数， 然后导入数据在主界面上画出波形图。
- Click_stopOsc(View view)方法：监听停止按钮，按下停止后改变接收数据绘图循环，使示波器停止工作。
- 重写 onBackPressed()方法：点击手机上的回退按钮后，如果功能切换菜单未开启则会打开

菜单，打开后再点击则会弹出确认退出软件的选项弹窗。

- Click_menu(View view)方法：定义了功能切换菜单的功能。

4.3 信号发生器功能

4.3.1 功能介绍

信号发生器是一种常用的电子设备，它能够提供各种频率、波形和电平信号。在测量各种电子电路系统的幅频特性、传输特性以及其他的电参数，或者测量一些电子元器件的电子特性及参数的时候，用来做测试的信号源或激励源。信号发生器一般都具有输出正弦波、三角波、矩形波的功能，无论是在科研领域还是生产实践中，信号发生器都有着很重要的作用[37]。

PocketLab 系统支持以下功能：

- 三角波、矩形波、正弦波这三种波形信号的输出
- 信号频率、幅度、直流偏移量的设置，矩形波还支持占空比的设置。
- 两通道独立输出或两通道差分输出

4.3.2 界面设计

信号发生器的界面设计，采用 RelativeLayout 相关布局，使用 Button、TextView、Switch、RadioButton 等常用组件进行设计。并使用了 pedant / sweet-alert-dialog 、SimonVT/android-menudrawer、code-troopers/android-betterpickers 和 eluleci/FlatUI 四个开源项目。其效果图如下：

图 4-9 信号发生器效果图

首先通过两个 Switch 分别设置信号发生器的输出方式和通道选择。输出方式有差分输出和独立设置，只有选择独立设置时才可以选择通道。

然后通过一组单选框中的三个选项选择波形，有正弦波，方波和三角波。这里用了 eluleci/FlatUI 的单选按钮，使界面更加美观。

点击频率，幅度和直流偏移都会唤起 code-troopers/android-betterpickers 中的数字选择器，并会将确认的数字显示在相应的位置，如下图：

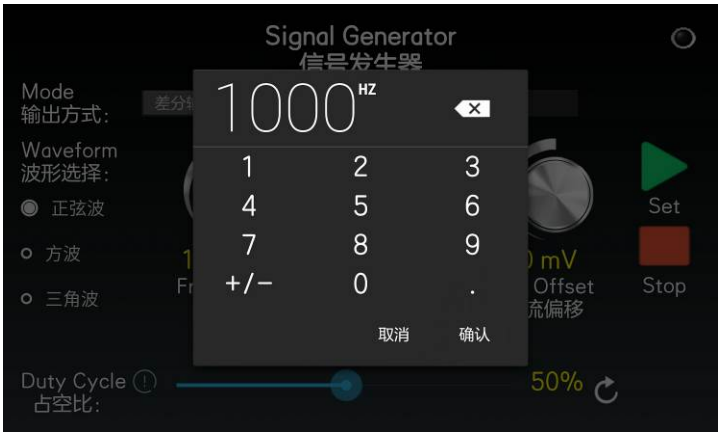


图 4-10 数字选择器

最后当波形选择为方波时可以通过滑动条来选择占空比。

当参数都设置完毕后点击 Set 按钮往硬件端发送设置命令，点击 Stop 使信号发生器停止工作。当硬件端正确反馈后会提示设置成功或关闭成功，如下图：

图 4-11 设置成功提示

4.3.3 工作流程

信号发生器的工作流程如下所示：

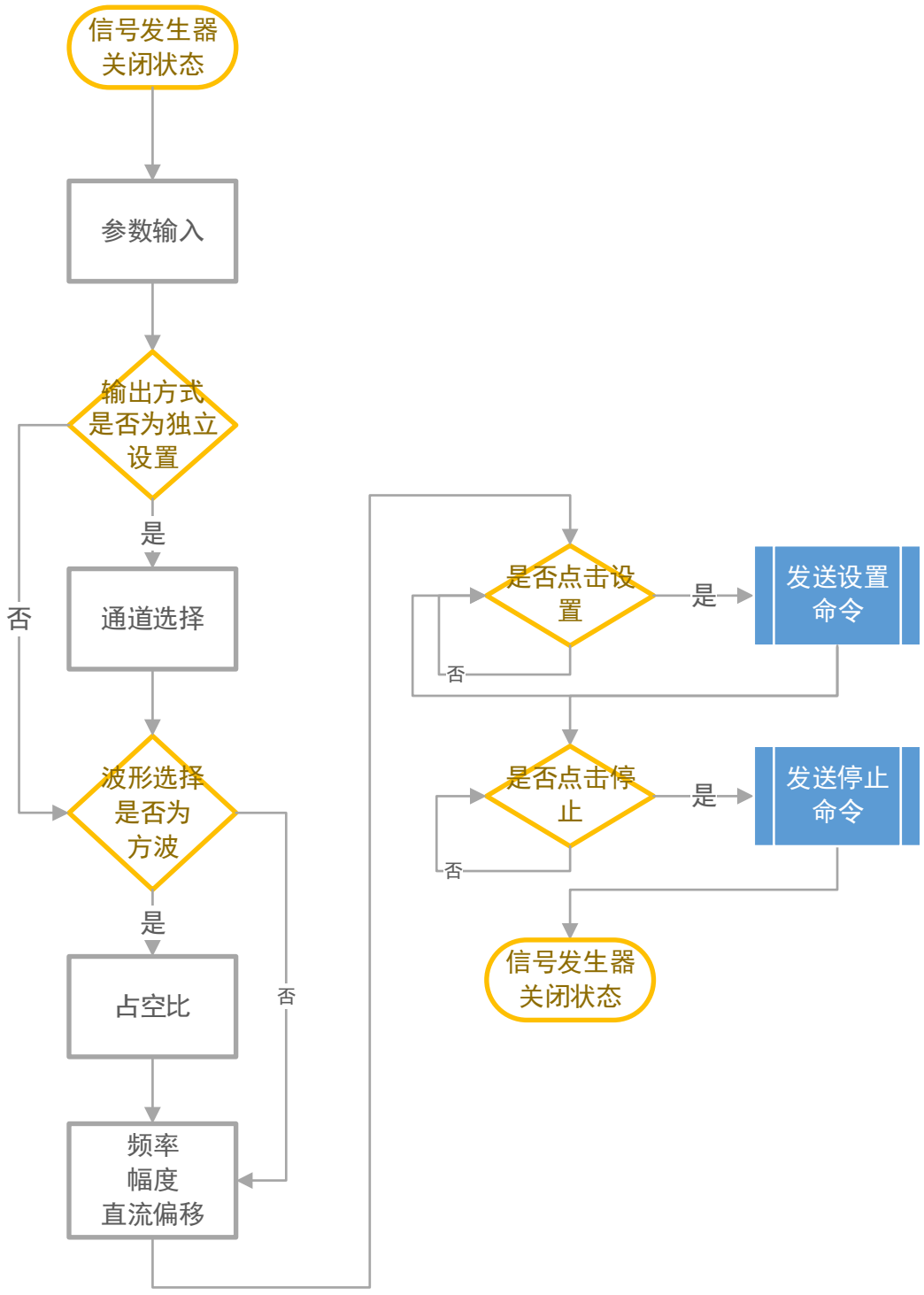


图 4-12 信号发生器工作流程图

在任何状态下都可以改变任何参数，再通过点击设置按钮软件就会根据最新设置的参数向硬件段发送设置波形的命令。

4.3.4 程序具体介绍

在 SignalGenerator 类中，实现了：

- ListenMode()方法：监听信号发生器输出方式的改变（通过点击 Switch 控件），并且当选中差分输出时使通道选择不可用，选中独立设置时激活通道选择。
- ListenChannel()方法：监听信号发生器通道选择的改变（通过点击 Switch 控件），选择不同通道后可以在下面分别设置两个通道的参数。
- ListenWave()方法：监听信号发生器波形的选择（通过点选单选框中的一个选项）。
- Click_k(View view)方法：分别点击频率，幅度和直流偏移的旋钮都会唤起此方法，然后唤起 code-troopers/android-betterpickers 中的数字选择器，并且根据点击的按钮定义可输入的最大最小值。确认输入后记录并在相应位置显示确认的数据。
- sbdc()方法：监听决定占空比的进度选择条，记录最终选择的占空比。
- Click_reset(View view)：定义占空比进度选择条旁边的圆形箭头按钮，点下后将进度条置到 50%的状态，方便操作。
- Click_set(View view)方法：监听绿色 Set 按钮，按下后根据当前的参数设置，决定向硬件端发送相应的信号发生器的开启命令。同时开启数据接收线程。
- Click_stop(View view)方法：监听红色 Stop 按钮，按下此按钮且当信号发生器处于开启状态时，向硬件端发送关闭命令。
- ReceiveThread 线程：开启此线程后开始接收从硬件端发送来的数据，并与成功开启和成功关闭反馈比较，确定达到的状态。
- 重写 onBackPressed()方法：同示波器中方法。
- Click_menu(View view)方法：同示波器中方法。

4.4 直流电压表功能

4.4.1 功能介绍

直流电压表又叫做伏特计，是一种科学仪器，作用是测量电压值的大小。通常使用的单位是伏特（V），本项目中使用的单位都是微伏(mV)。

本项目软件具有双通道直流电压表功能，量程为-4000mV 至 4000mV；同时具有双通道直流输出功能，输出范围为-4000mV 至 4000mV。

4.4.2 界面设计

直流电压表的界面设计，采用 RelativeLayout 相关布局，使用 Button、TextView 等常用组件进行设计。并使用了 pedant / sweet-alert-dialog 、SimonVT/android-menudrawer 和 code-troopers/android-betterpickers 三个开源项目。其效果图如下：

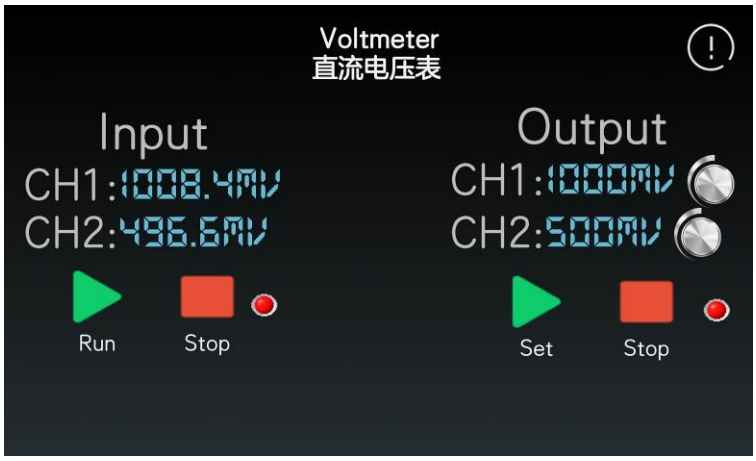


图 4-13 直流电压表界面图

上半部分为输入，即直流电压表功能。CH1、CH2 后的数字为其当前检测到的电压。绿色 Run 按钮为使其开始工作，红色 stop 按钮使其停止工作。

下半部分为输出，即直流电压输出功能。CH1、CH2 后的数字为其当前设定的电压，点击后面的旋钮会唤起数值输入窗口，从而输入电压设置，如下图：

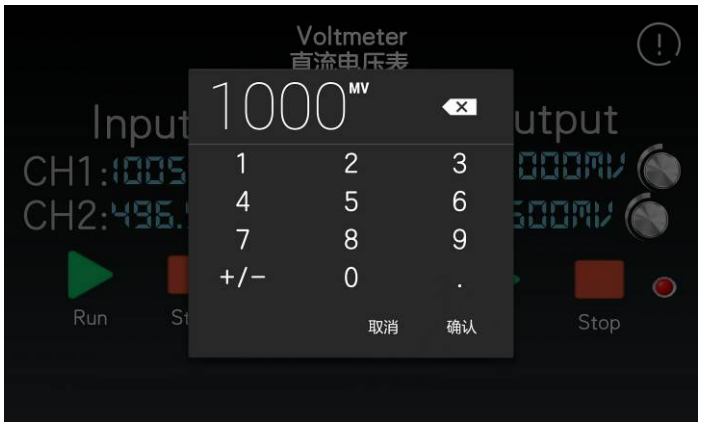


图 4-14 电压数值输入

同样绿色 Set 按钮为使信号发生器开始工作并更新为当前的设置，红色 stop 按钮使其停止工作。

右上角的惊叹按钮点击后给用户提示信息，因为软件中直流电压表和示波器功能都是使用的 PocketLab 硬件端中的示波器功能，如果软件中同时开启这两个功能必然会发生冲突。效果如下图：

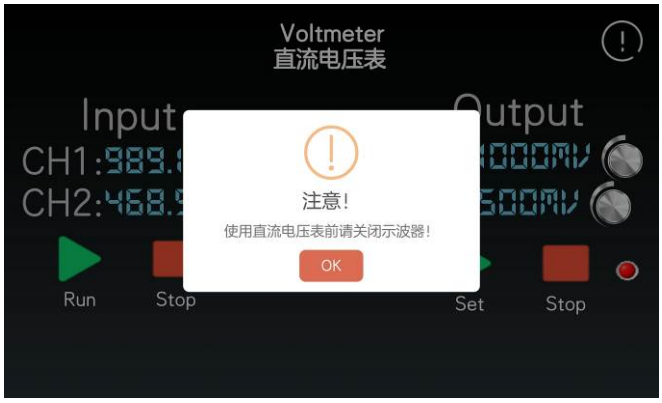


图 4-15 提示效果图

4.4.3 工作流程

直流电压表的工作流程如下所示：

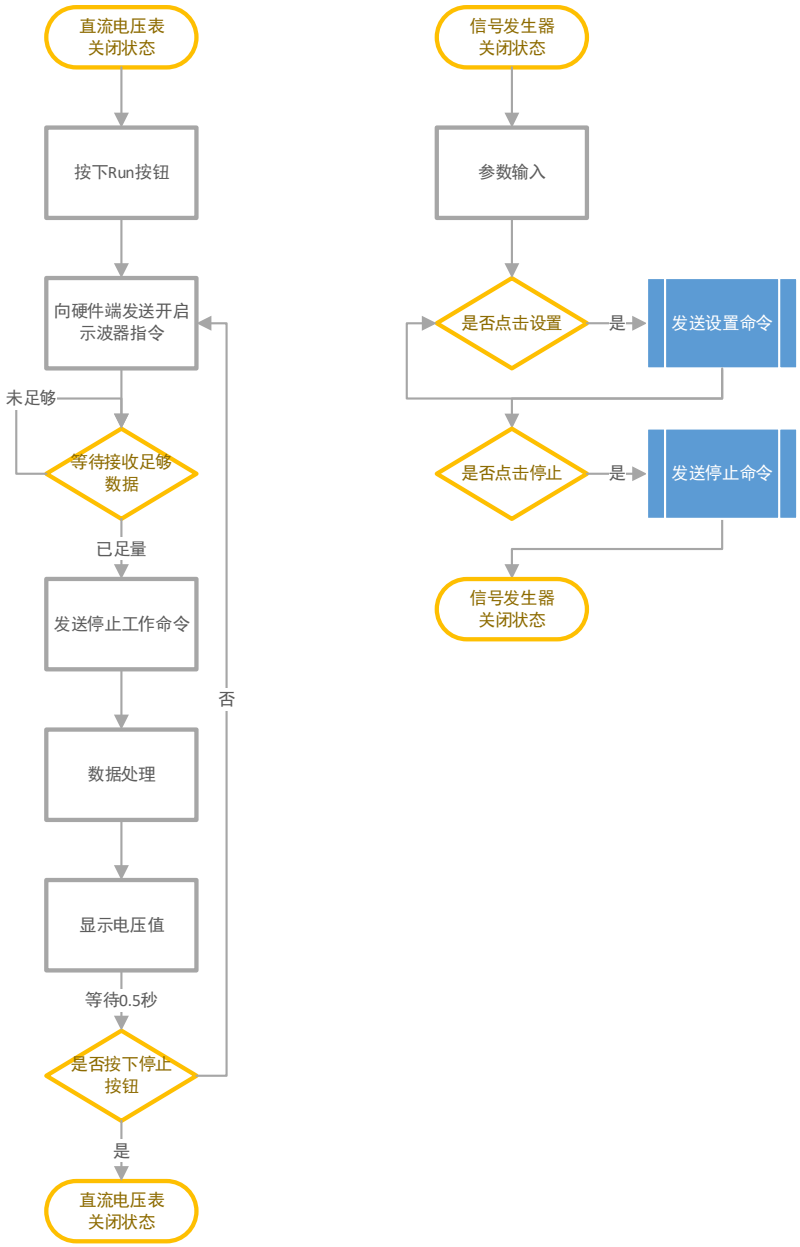


图 4-16 直流电压表工作流程图

PocketLab 硬件端没有直接提供直流电压表功能，但是可以用示波器的采样功能间接达到直流电压表的功能。即开启电压表时发送开启硬件端的可变采样率示波器，耦合方式都设置为 DC 耦合，采样频率设置为 50k sample/s，增益档位都设置为 1/4 即增益档位序号为 1。将收到的数据处理后再取平均值显示即可以达到直流电压表的功能。

同时本软件还具有双通道直流输出的功能，通过信号发生器实现。即设置好直流输出后，向硬件端发送开启信号发生器命令，输出方式设置为独立设置，两个通道波形都设置为方波，频率为 1000Hz（其实可以是任意值），幅值设置为 0，占空比设置为 0，直流偏移分别设置为在软件中两

个通道设置的直流输出值。通过这种方法即完成了直流输出。

4.4.4 程序具体功能介绍

在 Voltmeter 类中，实现了：

- setTypeface()方法：界面中输入输出的电压显示都使用了"DS-DIGIB.TTF"自定义字体，其显示效果如同数字 lcd 显示，如图 4-13 所示，另界面更美观。此方法实现了四个 textView 的自定义字体显示。
- Click_runVI(View view)方法：监听输入的绿色 Run 按钮，按下后使软件向硬件端发送相应的信号发生器的开启命令。同时开启数据接收线程。
- ReceiveThread 线程：开启此线程后软件端开始接收数据，判断接收到足量的数据后发送停止命令。然后处理数据，并更新显示电压值。经过一定延时后，再次发送继续接收采样命令，同时开启数据接收线程，即进入循环。
- byteArrayToInt(byte[] b)，getAD(byte[] b,int n,int ch)，getVinput(float gain, int offset, int[] AD, int vd)，getAvg(float[] input1, float[] input2)方法：都是用来处理接收到的数据，一个是将 byte 数组转为 int 型数；第二个是获得 AD；第三个根据增益、偏移量以及 AD 获得信号真正的电压；第四个方法是计算出两个通道一定时间内采样到电压值的平均值，即为直流电压表结果。
- Click_stopVI(View view)方法：监听输入的红色 Stop 按钮，按下后使直流电压表停止工作。
- Click_VO(View view)方法：分别点击两个输出的旋钮都会唤起此方法，然后唤起 code-troopers/android-betterpickers 中的数字选择器。确认输入后记录并在相应位置显示确认输入的电压值。
- Click_setVO(View view)方法：监听输出的绿色 Set 按钮，按下后根据当前设置的电压，决定向硬件端发送相应的信号发生器的开启命令。同时开启数据接收线程。
- Click_stop(View view)方法：监听输出的红色 Stop 按钮，按下此按钮且当输出即信号发生器处于开启状态时，向硬件端发送关闭命令。
- OReceiveThread 线程：开启此线程后开始接收从硬件端发送来的数据，并与成功开启和成功关闭反馈比较，确定达到的状态。
- 重写 onBackPressed()方法：同示波器中方法。
- Click_menu(View view)方法：同示波器中方法。

4.5 逻辑分析仪功能

4.5.1 功能介绍

PocketLab 的逻辑分析仪部分支持以下功能：

- 8 个 IO 口可分别独立设置参数
- IO 口可设置为 In，Out，CLK 三种模式

- Out 模式下可设置为高电平或低电平
- Clk 为时钟方式。并且可设置频率，频率范围为（1~10kHz）
- 可发送命令获取当前 8 个数字 IO 的电平状态并立即返回。返回数据是将逻辑分析仪接口状态以 8 位 16 进制数据表示成为 10 进制 ASCII 码字符串后发送。

4.5.2 界面设计

逻辑分析仪的界面设计，采用 RelativeLayout 相关布局，使用 Button、TextView、ImageView 等常用组件进行设计。并使用了 SimonVT/android-menudrawer 和 jfeinstein10/SlidingMenu 两个开源项目。其效果图如下：

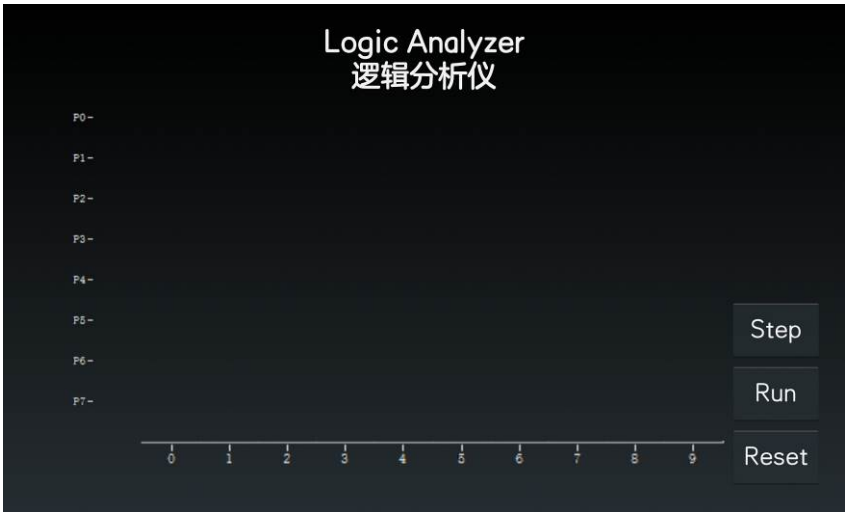


图 4-17 逻辑分析仪界面

主界面具有三个按钮：Step、Run 和 Reset。

按下 Step 按钮后首先会将 CLK 模式下的 IO 口输出电平取反一次，然后会对当前的八个 IO 口采样一次，并将结果以图表形式在中间显示。之后每按下一次 Step 都会将得到的结果向后顺延显示，当屏幕中间的图表已满之后，再次按下 Step 会将先前的结果都往前移动一格，横坐标为 0 的丢弃，最新的结果显示在最后，即横坐标为 9。显示效果如下图：

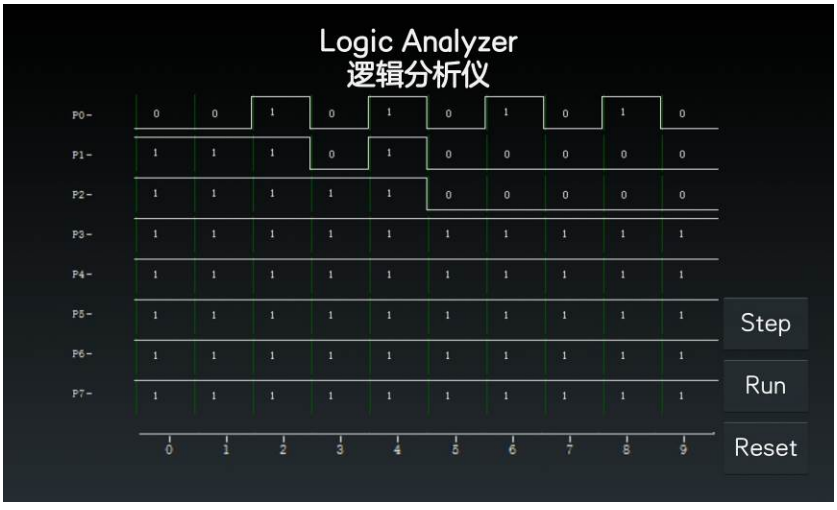


图 4-18 Step 采样结果显示

按下 Run 按钮后软件会以一定间隔自动采样，并显示结果，其每一步的具体过程如同 Step。同时按下 Run 后，此按钮会变为 Stop，按下 Stop 软件会停止自动采样。

按下 Reset 按钮后会将中间的图表清空，然后最新采样得到的结果会从头开始显示。

如同示波器的设计，因为主界面显示图表需要占用大量空间，参数设置菜单将通过在屏幕下边缘向上划拉出，如下图：

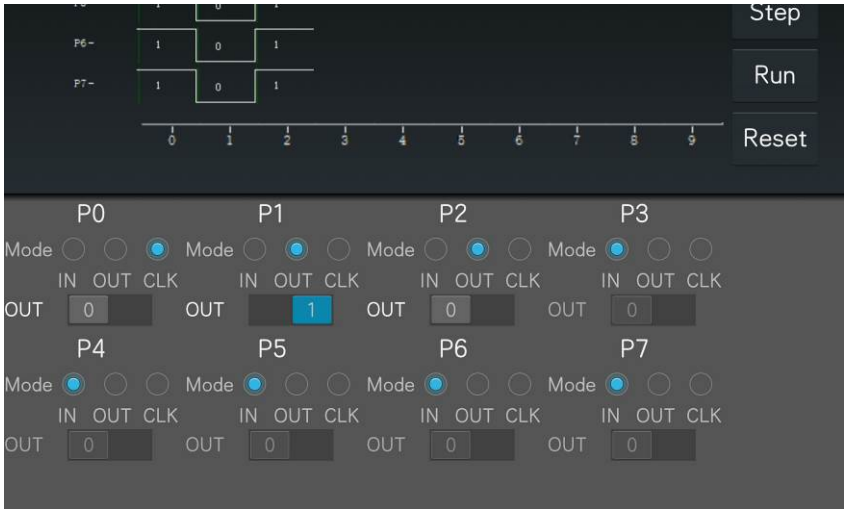


图 4-19 逻辑分析仪参数设置菜单

参数设置菜单中可以独立设置每个 IO 口的输入输出模式，仅当选中 OUT 或 CLK 模式时可以选择此 IO 口的输出电平。

4.5.3 工作流程

逻辑分析仪的工作流程如下所示：

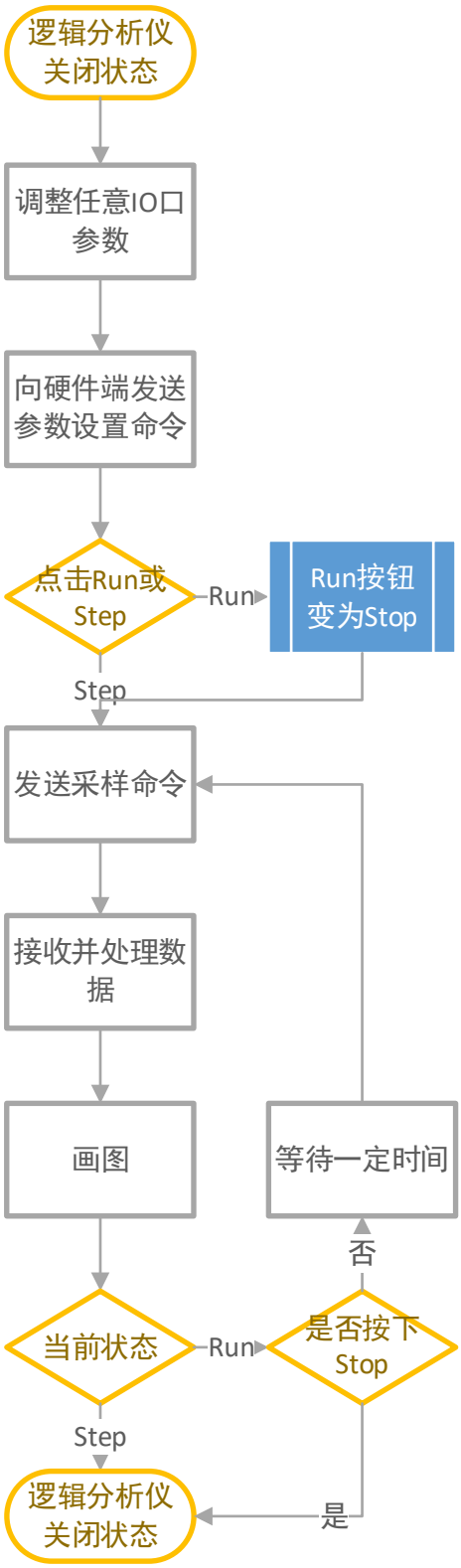


图 4-20 逻辑分析仪工作流程图

Pocketlab 硬件端的逻辑分析仪功能与其他功能使用略有不同：示波器功能和信号发生器功能

的开启与参数设置集中在一条指令中，而逻辑分析仪的参数设置与采样分别为两条指令，且每个 IO 口需要单独设置。因此每当任一 IO 的参数发生变化时软件都会向硬件段发送参数设置命令，按下 Step 或 Run 命令后发送采样命令。

4.5.4 程序具体功能介绍

在 LogicAnalyzer 类中，实现了：

- listenMode()方法：监听每一个 IO 口输入输出方式的改变（通过选择单选框中的选项），并且当选中 IN 模式时使电平设置不可用，选中 OUT 和 CLK 时激活电平选择。每当任一 IO 口的输入输出方式发生改变后即向硬件端发送相应的参数设置命令，同时打开数据接收线程。
- listenOut()方法：监听每一个 IO 口输出电平选择的改变（通过每个 Switch 的改变），仅当该 IO 口选中 OUT 或 CLK 模式时可用。每当任一 IO 口的输出电平发生改变后即向硬件端发送相应的参数设置命令，同时打开数据接收线程。
- ReceiveThread 线程：此为参数设置的数据接收线程。开启此线程后开始接收从硬件端发送来的数据，并与设置成功的反馈比较，确定设置成功，准备发送下一个参数设置命令。
- step()方法：点击 Step 按钮后调用此方法，首先检查每一个 IO 口的输入输出模式，将 CLK 模式下的 IO 口的输出电平取反，并向硬件端发送相应的参数设置命令。然后向硬件端发送采样命令，同时开启采样的数据接收进程。
- DReceiveThread 线程：此为采样的数据接收线程。开启此线程后开始接收从硬件端发送来的采样数据，将采样数据从 10 进制 ASCII 码字符串转为 int 型数字，再将此数字转为 2 进制并存在 char 型数组中，从而确定每个 IO 口的逻辑电平。最后调用画图方法。
- draw()方法：此为逻辑分析仪的画图方法。因为 Java 自带的 onDraw()方法画图过程非常冗杂，最后呈现效果也不够理想，并且目前也没有成熟的画逻辑电平图的开源项目可以使用。考虑到逻辑电平图的结果非常有限，所以本项目用了一个比较取巧的办法：

  将图标的 XY 轴分别做成图片，用 imageView 控件直接显示，同时在中间设置 8*10 个 imageView 控件矩阵，用来显示每一个 IO 每一个时刻的状态。因为每一个 IO 每一个时刻的逻辑电平图只有四种情况：从 0 到 0，从 0 到 1，从 1 到 1，从 1 到 0。其效果如下：

A．从 0 到 0　　B．从 0 到 1

C．从 1 到 1　　D．从 1 到 0

图 4-21 逻辑电平图四种情况图

因此只需要根据此 IO 口上一个时刻和这一个时刻的电平状态就可以确定需要显示的图

片。最后通过此图片矩阵就组成了逻辑电平图。

- Click_LogRun(View view)方法：按下 Run 按钮后，首先将 Run 按钮转变为 Stop 按钮，然后每过一定间隔调用 step()方法。若按下 Stop 按钮，首先将 Stop 按钮转变为 Run 按钮，然后中断 Run 循环，停止调用 step()方法。
- Click_LogReset(View view)方法：按下 Reset 按钮后，清空图片显示矩阵中显示的所有电平图片，并使下一次采样的图从 X 轴为 0 初开始显示。
- 重写 onBackPressed()方法：同示波器中方法。
- Click_menu(View view)方法：同示波器中方法。

4.6 程序功能测试

首先将手机 wifi 连接至 PockletLab 硬件端：

图 4-22 连接 wifi

打开 PocketLab App，并点击电源按钮：

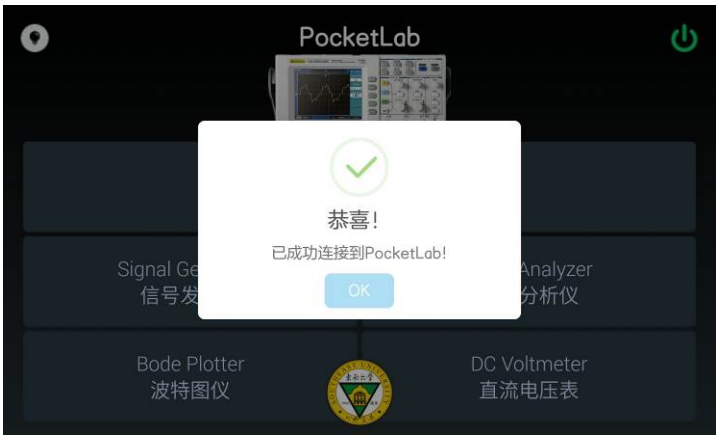


图 4-23 连接成功

提示连接成功，并且电源按钮变绿。

点击进入信号发生器功能，模式选择为差分输出，波形选择为正弦波，频率为 1000Hz，幅度为设置为 1000mV，直流偏移设置为 0mV。

图 4-24 信号发生器参数设置

点击 Set 按钮:

图 4-25 信号发生器设置成功

提示设置成功，且右上角电源灯亮起。将硬件端的信号发生器输出口连接至示波器的输入口。

再调出菜单，切换至示波器功能：

图 4-26 打开菜单切换功能

将示波器 CH1 垂直位移设置为 0mV，Volts/Div 设置为 1V。CH2 垂直位移设置为 1000mV，Volts/Div 设置为 2V。耦合方式都为 AC，TIME/DIV 为 1ms:

图 4-27 示波器参数设置

然后点击绿色开始按钮:

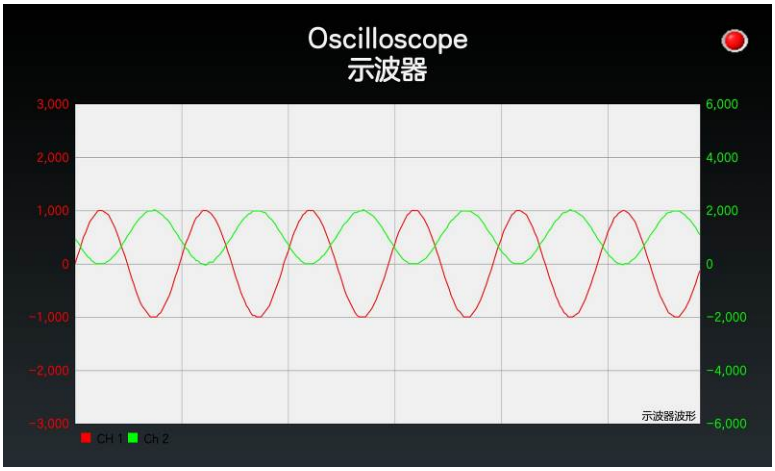


图 4-28 示波器波形

右上角电源灯亮起，同时显示出波形，与示波器的参数设置和信号发生器输出的波形一致。

关闭示波器与信号发生器：

图 4-29 信号发生器提示关闭成功

进入直流电压表功能，将输出的 CH1 设置为 1000mV，CH2 设置为 500mV。同时打开输入和输出:

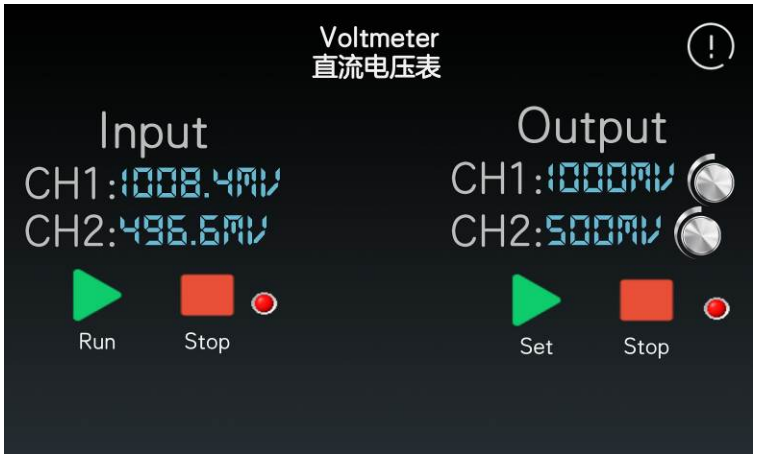


图 4-30 输出设置与结果显示

可以看到通过输入测量到的结果与输出基本一致。再关闭直流电压表的两个功能，然后切换至逻辑分析仪功能。

将 P0 口设置为 CLK；P1 口设置为输入；P2 口设置为输出，电平为 1；P3 口设置为输入；其他口都设置为输出，电平为 0:

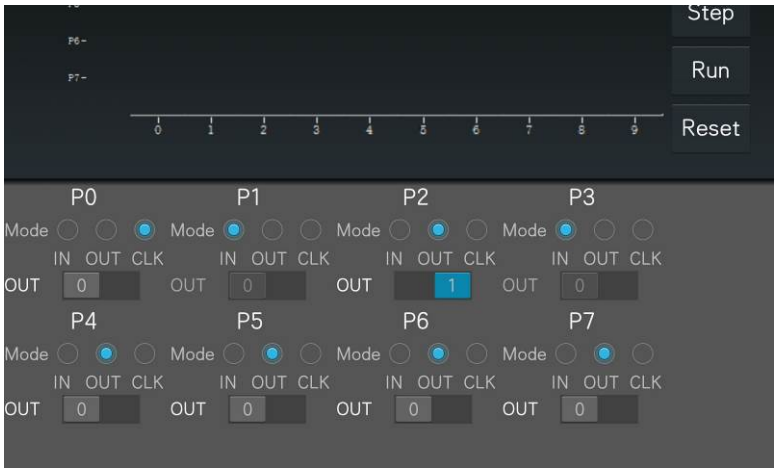


图 4-31 逻辑分析仪参数设置

在硬件段将 P0 口与 P1 连接，P2 与 P3 连接。然后点击 Run 按钮，使逻辑分析仪开始工作：

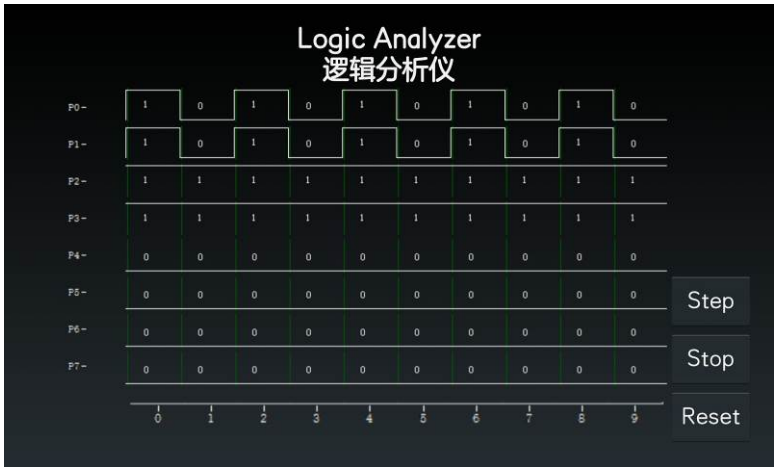


图 4-32 逻辑分析仪显示结果

可以看出显示结果和参数设置与硬件连接结果一致。

通过程序实际测试，可以验证说明本项目软件各项功能已得到良好实现。

4.7 本章小结

本章主要介绍了本项目软件中的主界面、示波器功能、信号发生器功能、直流电压表功能和逻辑分析仪功能。并具体介绍了每部分的功能，界面设计，工作流程以及程序。最后实际测试了本项目软件的各项功能。

第5章 总结

因为虚拟仪器与 Android 移动设备的迅猛发展，基于 Android 平台的虚拟仪器系统变为一个迫切的需求。本项目主要是根据目前已有的东南大学自主研发的 PocketLab 硬件系统，设计一个在 Android 系统下运行的虚拟仪器软件。通过早期的对相关文献的阅读，以及对 Android 开发的相应知识的学习，确定了软件的设计思路与整体框架。经过实际的编写和调试，最终实现了信号发生器、示波器、直流电压表和逻辑分析仪四个功能模块。

主要完成的部分有：

1) Socket 通信通道的建立，在不同 Activity 间切换对 Socket 连接的保持。
2) 与硬件进行高速数据传输，同时进行不同数据格式间的转换。
3) 在每个功能界面实现一到两个侧滑菜单，分别实现功能切换和参数设置。
4) 信号发生器的功能的实现，其可以设置输出方式，并在独立设置时可以分别设置两个通道的参数；可以输出正弦波、方波或三角波，方波时可以选择占空比；频率、幅度和直流偏移都可以独立设置。
5) 示波器的功能的实现，其可以同时显示双通道波形，并且可以分别设置两个通道的耦合方式、垂直位移和电压单位，还可以设置整体的时间单位；使用 PhilJay/MPAndroidChart 绘制波形图。
6) 直流电压表功能的实现，其具有双通道独立直流输出功能，同时具有双通道独立直流电压表功能显示输入电压。
7) 逻辑分析仪的实现，具有 8 个逻辑输入输出口，可分别设置为输入、输出或时钟模式，在输出状态下可设置逻辑电平的状态；还具有单步或自动定时检测每个输入输出口逻辑状态并整体绘制成图的功能；通过 ImageView 矩阵绘制逻辑电平图。

总之，本项目在目前已有的基础上进行了一些尝试和努力，取得了较大的进展。然而，因为本项目的任务量很大，时间比较仓促，同时 PocketLab 硬件端的无线通信模块也不是十分完善，过程中我们软硬件一直在协同调试与改进。所以最终本项目尚存在一些可以继续完善的地方：

1) 丰富示波器的功能，比如加入选择触发电平、频谱分析和数学运算功能；
2) 实现波特图仪的功能；
3) 完善软件的操作逻辑，在用户进行一些误操作以后软件仍能有正确的响应与反馈；
4) 将软件移植实现在 IOS 系统上，使 PocketLab 全平台可用。

参考文献

[1] 赵娴. 远程数据采集虚拟仪器的快速实现方法及其应用[D].武汉科技大学,2012.

[2] 邢丽娟,杨世忠. 虚拟仪器的原理及发展[J]. 山西电子技术,2006,01:90-91.

[3] 孙宁. 基于虚拟仪器的教学实验的研究[D].西华大学,2006.

[4] 王常延. 组建自己的虚拟仪器[J].国外电子测量技术,1996,(06):25-28.

[5] 张文. 基于虚拟仪器的实验室建设研究[J]. 内江师范学院学报,2004,02:95-98.

[6] 乔建良,黄大勇. 虚拟仪器的现状及应用前景[J]. 信息技术,2002,10:94-95.

[7] 宋波,陈一民. 虚拟仪器开发环境的比较研究[J]. 计算机工程与设计,2007,12:2971-2973+2976.

[8] 丁业昌,徐利明,李建林. 虚拟仪器技术在实验室测量中的应用[J]. 软件,2013,02:168-170.

[9] 李明,杨其锋. 一种基于声卡的虚拟示波器的设计[J]. 科技通报,2013,04:155-157.

[10] Wang Lijun ; Chen Changxin ; Ma Lili ; Shen Jie ,Development of Multi-functional VirtualOscilloscope Based on Soundcard ,2011 Third International Conference on Measuring Technology and Mechatronics Automation (ICMTMA), Volume: 1 Page(s): 1056 – 1059

[11] Wang Lijun ; Chen Changxin ; Ma Lili ; Shen Jie ,Development of Multi-functional VirtualOscilloscope Based on Soundcard ,2011 Third International Conference on Measuring Technology and Mechatronics Automation (ICMTMA), Volume: 1 Page(s): 1056 – 1059

[12] Karim, I.A. ,A low cost portable oscilloscope based on Arduino and GLCD, 2014 International Conference on Informatics, Electronics & Vision (ICIEV), 2014 , Page(s): 1 – 4

[13] 宋菲菲. 基于 Android 系统的示波器应用程序开发[D].哈尔滨工业大学,2013.

[14] 屈欢. 基于 Android 平台的无线数字示波器设计[D].电子科技大学,2013.

[15] 王升. 基于 Android 平台的虚拟示波器开发[D].哈尔滨理工大学,2014.

[16] 王楠. 基于 Android 手机平台的互联网应用探析[J]. 数字化用户,2013,10:3.

[17] 黄志艳. 3G-Android 手机平台应用开发研究[J]. 科技广场,2013,10:54-56.

[18] 刘瑞顺. 基于 Android 平台的智能手机输入法研究与设计[D].汕头大学,2011.

[19] 马越. Android 的架构与应用[D].中国地质大学（北京）,2008.

[20] 姚昱旻,刘卫国. Android 的架构与应用开发研究[J]. 计算机系统应用,2008,11:110-112+24.

[21] 胡伟. Android 系统架构及其驱动研究[J]. 广州广播电视大学学报,2010,04:96-101+112.

[22] 梅瑞,武学礼,文伟平. 基于 Android 平台的代码保护技术研究[J]. 信息网络安全,2013,07:10-15.

[23] 王茜. Android 嵌入式系统架构及内核浅析[J]. 电脑开发与应用,2011,04:59-61.

[24] 闵现畅,黄理灿. 基于 Android 平台的 Web 服务技术研究[J]. 工业控制计算机,2011,04:92-94.

[25] 李林. 基于 Android 系统的手机键盘驱动设计与实现[D].西安电子科技大学,2012.

[26] 张娜. Android 系统架构研究与应用[D].西安科技大学,2013.

[27] 李家科. Android 系统分析与开发[D].兰州交通大学,2014.

[28] 易国平. SurfaceView 在 Android 游戏开发中的研究[J]. 学周刊,2014,18:81.

[29] 刘辉. Java 网络编程:连接网络数据库的多种方法[J]. 价值工程,2010,03:154.

[30] 郑霞,郭磊. Java 语言中字符串常量和变量的分析与比较[J]. 安阳师范学院学报,2012,02:46-48.

[31] 陈正生,吕志平,黄令勇,吕浩. 多卫星导航系统时间基准的统一设计及精度分析[J]. 导航定位学报,2013,04:70-73.

[32] 王菁,魏霞. Java 语言多线程技术及应用探讨[J]. 农业网络信息,2009,05:121-123.

[33] 史书明. Android 应用中消息传递方法分析[J]. 电脑知识与技术,2014,13:2984-2986+3008.

[34] 纪晓阳. 线程在 Android 开发中的应用[J]. 软件,2013,08:24-26+41.

[35] 高兰兰. 基于 Android 平台的多分辨率解决方案[J]. 软件,2011,09:70-72.

[36] 沈辉,沙立民,张重龙. 基于 LabVIEW 的多功能虚拟示波器设计[J]. 电子测量技术,2012,11:90-93.

[37] 郑冰,杨旭. 关于模拟电路中常用检测仪器及测试方法的研究[J]. 科技信息,2010,22:88.

致谢

逝者如斯夫，转眼间，大学生活就要结束，毕业论文的撰写工作也行将结束。

本论文是在指导老师孟桥教授的悉心指导和严格要求下完成的，在整个毕业设计进行的过程中，老师对选题、构思、整体设计都进行了详细的指导。老师那严谨的治学态度、敏捷的学术思维、实事求是的学术作风和诲人不倦的师者风范使我受益匪浅，在此向孟老师致以衷心的感谢。

在本项目的进行中，得到了实验室刘炜学长的大力支持与帮助，刘学长牺牲了大量的私人时间与我一起调试硬件和程序，和我探讨、帮助我提出建议解决问题，甚至有时陪我在实验室工作到凌晨才回去。非常感谢刘炜学长。

感谢我亲爱的女朋友，这个学期毕业设计的任务量很大，占用了我大量的时间，她可以理解我少了很多时间陪她，并且在每个艰辛的时刻都有她的关心和鼓励，使我再充满斗志。

最重要的，感谢我的父母，在我的成长之路和求学之路上一直毫无保留的支持我，没有父母的关怀和支持就不会有今天的我。

还要感谢这回的这个毕设题目，通过这几个月在这个题目中的探索和努力，使我发现了一片新大陆——我对编程的兴趣。这也奠定了我下面研究生的研究方向，这也许会对我的整个人生方向产生重大的影响。

最后祝每一位指导帮助过我的老师、同学、朋友身体健康、工作顺利、万事如意。

何功垠

2016 年 5 月 25 日

附录 I

PocketLab 通讯指令集 V2.0

本指令集对应 Pocketlab_CCS_16bit_Version2.2

PocketLab 的控制命令为 ASCII 码，以回车符“\r”(ASCII 码为 0x0d,下文中用↙代表)结束。设备的返回信息均以\r\n（两字节）结束，参数之间用空格(下文中用␣表示)隔开。

一、 通用命令

握手信号：确认连接设备

功能：握手指令，主要是用于上位机确认 Pocketlab 设备连接正常。

发送：HELLO↙

回复：I ␣ am ␣ PocketLab ␣ V2.0.0! \r\n（共 24Byte）

二、 模拟信号相关指令

1. 信号发生器（两通道差分输出，Differential Mode）设置命令

功能：设置信号发生器的输出参数。此时信号发生器产生两通道各参数相等但相位差 180° 的差分信号。

发送：OPEN_SIG ␣<A1>␣<A2>␣<A3>␣<A4>␣<A5>↙

回复：Setting ␣ SIG ␣ OK!\r\n（共 17Byte）

参数说明：

<A1>为信号类型，内容如下：

1) <A1> = SIN：产生正弦波；

其它参数：<A2>为其频率（单位：Hz），合法范围为 1-20000 内正整数；<A3>为输出正弦波的幅值，单位 mV，合法范围为 0~4000 内的正整数；<A4>为输出信号的直流电压，单位 mV，合法范围为-4000~4000 内的整数；<A5>无效，不写即可。

例 1：输出 100Hz、峰峰值 2000mV、直流偏移 1000mV 的正弦波，指令为：

OPEN_SIG ␣ SIN ␣ 100 ␣ 1000 ␣ 1000↙

例 2：输出 100Hz、峰峰值 2000mV、直流偏移 -1000mV 的正弦波，指令为：

OPEN_SIG ␣ SIN ␣ 100 ␣ 1000 ␣-1000↙

注：实际 Pocketlab 只能输出±4000mV 的电压，因此输入的幅值和直流偏移的绝对值不要大于 4000

2) <A1> = REC：产生方波；

其它参数：<A2>为其频率（单位：Hz），合法范围为 1-5000 内正整数；<A3>为输出方波的幅值，单位 mV，合法范围为 0~4000 内的正整数；<A4>为输出信号的直流电压，单位 mV，合法范围为-4000~4000 内的整数；<A5>为其占空比，表示高电平占每个周期的百分比，合法范围 0-99 内的正整数。

例 1：输出 100Hz、峰峰值 2000mV、直流偏移 1000mV、占空比 60%的矩形波，指令为：

OPEN_SIG ␣ REC ␣ 100 ␣ 1000 ␣ 1000 ␣ 60↙

3) <A1> = TRI：产生三角波；

其它参数：<A2>为其频率（单位：Hz），合法范围为 1-5000 内正整数；<A3>为输出三角波

的幅值，单位 mV，合法范围为 0~4000 内的正整数；<A4>为输出信号的直流电压，单位 mV，合法范围为-4000~4000 内的整数；<A5>无效，不写即可。
例 1：输出 100Hz、峰峰值 2000mV、直流偏移 1000mV 的三角波，指令为：
OPEN_SIG ⌴ TRI ⌴ 100 ⌴ 1000 ⌴ 1000↙

4) 当<A2>为 0 时，不论<A1><A3>为何，均输出<A4>对应的直流信号。
例 1：输出直流电平 -1000mV，指令为：
OPEN_SIG ⌴ XXX ⌴ 0 ⌴ XXX ⌴-1000↙ XXX 表示任意字符；

2. 信号发生器（两通道独立输出，Separate Mode）设置命令
功能：设置信号发生器的输出参数，两通道的参数需要单独设置。

发送：OPEN_SIG2 ⌴<A1>⌴<A2>⌴<A3>⌴<A4>⌴<A5>⌴<A6>↙

回复：Setting ⌴ SIG ⌴ OK!\r\n（共 17Byte）

参数说明：
<A1>为通道选择，输入 0 设置通道 0（CH0），1 对应通道 1（CH1）
<A2>为信号类型，内容如下：
1) SIN：产生正弦波；
其它参数：<A3>为其频率（单位：Hz），合法范围为 1-5000 内正数，频率步进 0.5HZ；<A4>为输出正弦波的幅值，单位 mV，合法范围为 0~4000 内的正整数；<A5>为输出信号的直流电压，单位 mV，合法范围为-4000~4000 内的整数；<A6>无效，不写即可。
例 1：设置通道 0 输出 100.5Hz、峰峰值 2000mV、直流偏移 1000mV 的正弦波，指令为：
OPEN_SIG2 ⌴ 0 ⌴ SIN ⌴ 100.5 ⌴ 1000 ⌴ 1000↙
例 2：设置通道 1 输出 100Hz、峰峰值 2000mV、直流偏移 -1000mV 的正弦波，指令为：
OPEN_SIG2 ⌴ 1 ⌴ SIN ⌴ 100 ⌴ 1000 ⌴-1000↙

2) REC：产生方波；
其它参数：<A3>为其频率（单位：Hz），合法范围为 1-5000 内正数，频率步进 0.5HZ；<A4>为输出方波的幅值，单位 mV，合法范围为 0~4000 内的正整数；<A5>为输出信号的直流电压，单位 mV，合法范围为-4000~4000 内的整数；<A6>为其占空比，表示高电平占每个周期的百分比，合法范围 0-99 内的正整数。
例 1：设置通道 0 输出 100.5Hz、峰峰值 2000mV、直流偏移 1000mV、占空比 60%的矩形波，指令为：
OPEN_SIG2 ⌴ 0 ⌴ REC ⌴ 100.5 ⌴ 1000 ⌴ 1000 ⌴ 60↙
例 2：设置通道 1 输出 100.5Hz、峰峰值 100mV、直流偏移 1000mV、占空比 79%的矩形波，指令为：
OPEN_SIG2 ⌴ 1 ⌴ REC ⌴ 100 ⌴ 50 ⌴ 1000 ⌴ 79↙

3) TRI：产生三角波；
其它参数：<A3>为其频率（单位：Hz），合法范围为 1-5000 内正数，频率步进 0.5HZ；<A4>为输出三角波的幅值，单位 mV，合法范围为 0~4000 内的正整数；<A5>为输出信号的直流电压，单位 mV，合法范围为-4000~4000 内的整数；<A6>无效，不写即可。
例 1：设置通道 1 输出 100Hz、峰峰值 2000mV、直流偏移 1000mV 的三角波，指令为：
OPEN_SIG2 ⌴ 1 ⌴ TRI ⌴ 100 ⌴ 1000 ⌴ 1000↙

4) 当<A3>为 0 时，不论<A2><A4>为何，均输出<A5>对应的直流信号。
例 1：设置通道 1 输出直流电平 -1000mV，指令为：
OPEN_SIG2 ⌴ 1 ⌴ XXX ⌴ 0 ⌴ XXX ⌴-1000↙ XXX 表示任意字符；

3. 关闭信号发生器命令：

功能：PocketLab 停止产生输出信号，对差分输出和独立输出均有效，对独立输出而言是同时关闭两通道。此时 Pocketlab 会输出一个接近 0 的电压。

发送：SET_SIG_OFF↙

回复：SIG ␣ turned ␣ off!\r\n（共 17Byte）

4. 示波器（采样率 100k sample/s，8bit 采样）启动命令

功能：开启示波器，Pocketlab 开始以 100kHz 采样频率、8-bit 精度采集数据并发送。

发送：OPEN_OSC ␣<A1>␣<A2>␣<A3>␣<A4>↙

参数说明：

<A1> <A3>对应 AD0 AD1 两通道的耦合方式（0——DC 耦合，1——AC 耦合）

<A2> <A4>对应 AD0 AD1 的增益档位序号

通道增益	1/8[1]	1/4	1/2	1	2	4	8	16	32	64[2]
增益档位序号	0	1	2	3	4	5	6	7	8	9
最大输入电压（mV）	4000	4000	3200	1500	800	400	200	100	50	25
分辨率（mV/LSB, 因为实际增益会有误差，这里给出近似值）	103	52	26	13	6	3	1.6	0.8	0.4	0.2

回复：

f） 首先返回标志信息 mode（char 型，1 字节），固定为十六进制 A3。

g） 然后返回实际设定的采样间隔十六进制 0A 00（int 型，共 2 字节，单位 us），表示采样周期为 10us。

h） 接着返回 ch0 ch1 在设置的档位下的偏移值 offset（两个 int 型，共 4 字节，单位 mV，通道 0 在前）。

i） 然后是两通道的实际增益值 gain（两个 float 型，共 8 字节，单位 mV/LSB，通道 0 在前）数据顺序是低字节在前（例如，Pocketlab 发送两字节 int 十六进制 CF 18，对应的 int 数是 0x18CF；发送四字节 float 98 D1 3F 42，实际应为 423FD198 ，float 为 IEEE 32bit float 格式，表示 47.9546814）。

j） 然后返还数据。数据格式为：ABABABABAB……其中每个 A 和 B 为一个字符（每个通道每个采样值 1 字节），分别对应通道 0 和通道 1 的 8bit 采样值。每个通道的实际输入电压与采样值的换算公式为：$Vinput=AD_{8bit}*-gain+offset$（mV）

例 1：设置通道 0 直流耦合，通道增益 1；通道 1 直流耦合，通道增益 1 指令为：

OPEN_OSC ␣ 0 ␣ 1 ␣ 0 ␣ 1↙

返回：A3 0A00 2618 4418 F6B64342 03194442（实际没有空格，空格是为了区分不同含义的数据）然后是采样数据

解释：

a) 所有的数据为 A3 表示工作模式为 100ks/s 8bit；

b) 0A00 int 对应十进制 10，表示采样周期 10us；

c) 2618 int 对应十进制 6182，即通道 0 的输入 0V 时对应的偏移是 6182mV；4418 int 对应十进制 6212，即通道 1 的输入 0V 时对应的偏移是 6212mV；

d) F6B64342 float 正确顺序应为 4243B6F6 按 IEEE 32-bit 浮点数解释约为 48.9287，即通道 0 采集到的 AD 数据每一 LSB 对应 48.9287；03194442 float 正确顺序应为 42441903

[1] 这个增益档在设计上位机程序时没有使用。

[2] 这个增益档在设计上位机程序时没有使用，且可能会在新版本中取消。

按 IEEE 32-bit 浮点数解释约为 49.0244，即通道 1 采集到的 AD 数据每一 LSB 对应 49.0244；

例 2：设置通道 0 直流耦合，通道增益 1；通道 1 交流耦合，通道增益 4 指令为：

OPEN_OSC ␣ 0 ␣ 1 ␣ 1 ␣ 4↙

5. 可变采样率示波器（可变采样率，采样率≤50k sample/s，8bit 采样）启动命令

功能：开启示波器，Pocketlab 开始以设定的采样频率（≤50Ksample/s）、8-bit 精度采集数据并发送。（为了保证性能建议采用 50ks/s，其他采样率不能保证工作性能。）

发送：OPEN_OSC2 ␣<A1>␣<A2>␣<A3>␣<A4>␣<A5>↙

参数说明：

<A1>对应需要的采样间隔，单位为 us；Pocketlab 会将实际采样率设置为尽可能接近输入采样间隔的采样频率。

<A2> <A4>对应 AD0 AD1 两通道的耦合方式（0——DC 耦合，1——AC 耦合）

<A3> <A5>对应 AD0 AD1 的增益档位序号

通道增益	1/8	1/4	1/2	1	2	4	8	16	32	64
增益档位序号	0	1	2	3	4	5	6	7	8	9
最大输入电压（mV）	4000	4000	3200	1500	800	400	200	100	50	25
分辨率（mV/LSB, 近似值）	103	52	26	13	6	3	1.6	0.8	0.4	0.2

回复：

基本与示波器采样指令（OPEN_OSC）一致。

a) 首先返回标志信息 mode（char 型，1 字节），固定为十六进制 A2。
b) 然后返回实际设定的采样间隔（int 型，共 2 字节，单位 us），如 50kHz 采样返回十六进制 14 00，对应十进制 20
c) 随后返回 ch0 ch1 在设置的档位下的偏移值 offset（两个 int 型，共 4 字节，单位 mV，通道 0 在前）。
d) 然后是两通道的实际增益值 gain（两个 float 型，共 8 字节，单位 mV/LSB，通道 0 在前）数据顺序是低字节在前（例如，Pocketlab 发送两字节 int 十六进制 CF 18，对应的 int 数是 0x18CF；发送四字节 float 98 D1 3F 42，实际应为 423FD198 ，float 为 IEEE 32bit float 格式，表示 47.9546814）。
e) 开始返还数据。数据格式为：ABABABABAB……其中每个 A 和 B 为一个字符（每个通道每个采样值 1 字节），分别对应通道 0 和通道 1 的 8bit 采样值。每个通道的实际输入电压与采样值的换算公式为：$Vinput=AD_{8bit}*-gain+offset$（mV）

例 1：设置采样率 50Ksample/s 通道 0 直流耦合，通道增益 1；通道 1 交流耦合，通道增益 2 指令为：

OPEN_OSC2 ␣ 20 ␣ 0 ␣ 3 ␣ 0 ␣ 4↙

6. 示波器继续采集指令（100K 采样）：

功能：按照上一次 OPEN_OSC 命令的参数设置不变，直接开始 AD 采样。一般它执行于 SET_OSC_OFF 命令之后，快速开始采集。

发送：CON_OSC↙

回复：直接开始发送采样数据；数据格式为：CH0CH1CH0CH1……

7. 示波器继续采集指令（可变采样率，包括 8bit 和 12bit）：

功能：按照上一次 OPEN_OSC2 或 START_OSC12 命令的参数设置不变，直接开始 AD 采样。

一般它执行于 SET_OSC_OFF 命令之后，快速开始采集。

发送：CON_OSC2↙

回复：直接开始发送采样数据；数据格式根据之前设置精度为 8bit 或 12bit 有区别。

8. 关闭示波器：
功能：停止示波器工作（即停止采样与发送数据）。对所有示波器工作模式均有效。

发送：SET_OSC_OFF↙

回复：OSC ␣ turn ␣␣ off! \r\n（共 16Byte）停止采集数据；注意 turn 后有两个空格！

三、 逻辑分析仪相关指令

1. 数字 IO 控制
功能：使用逻辑分析仪的各项功能。

发送：SET_LOG ␣<A1>␣<A2>␣<A3>↙
参数说明：
<A1> :操作命令，内容如下：
1) MOD： IO 口的输入输出方式，<A2>指定的数字 IO 位号（0-7），<A3>数字对应的数据，决定该位的方式：0：输入，1：输出
2) CLK： 时钟方式。此时将<A2>对应的端口设置为输出时钟模式，<A3>为时钟周期（单位:0.1 毫秒），如输入 10，对应 1kHz。频率范围为（1~10kHz）
3) HIG：将<A2>指定的端口输出电平设定为高；
4) LOW： 将<A2>指定的端口输出电平设定为低；

回复：LOG ␣ set!\r\n（共 10Byte）

注意：上电后 IO 口默认输入状态。

2. 读取逻辑分析仪状态
功能：获取当前 8 个数字 IO 的电平状态并立即返回。

发送：GET_LOG↙

回复：LOG <A1>

功能：其中<A1>是将逻辑分析仪接口状态以 8 位 16 进制数据表示成为 10 进制 ASCII 码字符串后发送。如收到“128”则对应 0x80。LED 亮灭对应于读取到的电平，是输入或输出电平取决于该引脚的 MODE。必须在 SET_LOG 后输入。

3. 读取数字 IO 的工作状态
功能：读取各个数字 IO 当前的工作状态：输出还是输入。

发送：GET_LOG_MOD↙

回复：LOG ␣ MODE ␣<A1>\r\n（字节数不定，A1 可能为 1-3 字节）
其中<A1>是将逻辑分析仪接口状态对应的十进制数的 ASCII 码（P7 为 MSB，p0 为 LSB，0——in，1——out，将 8 位二进制数转化为相应的十进制数，并用 ASCII 码表示相应的二进

制数)。

例 1：当前的 0、4、5 引脚为输出状态，其余为输入状态，则返回：

LOG ␣ MODE ␣ 49\r\n（49 为 ASCII 码，不是二进制数据，对应十进制 49，二进制 00110001，即从 P7 至 P0 状态依次为：IN IN OUT OUT IN IN IN OUT）