浙江大学

课程名称：智能终端软件开发

授课老师：李石坚

小组成员姓名：李易儒 陈威志

专业：计算机科学与技术

学号：3120100759 3120000521

电子邮件：3120100759@zju.edu.cn 3120000521@zju.edu.cn

实验日期：2014 年 9 月——2014 年 12 月

1.引言

1.1 程序概述

闹钟是人们平常生活中经常需要使用的工具，但是目前市场上存在的很多闹钟程序过于雷同相似，而且仅仅依靠铃声这种效率较低的方式并不一定能够完全叫醒别人，针对这一情况，我们开发了自己的闹钟程序——*趣味闹钟*。本程序能够在闹钟响起后，选择多种解锁方式，以取消闹铃，达到使人彻底清醒的目的。如光强解锁、水平仪解锁、震动解锁等特殊的解锁方式，可以使得想要解锁闹钟的人必须有一定的清醒才能够将闹钟取消，在闹醒别人时更有效率。

1.2 开发环境

- Windows8.1 Eclipse 3.9.2（Kepler）
- Android SDK:4.4.2

1.3 程序分工

UI：　　陈威志、李易儒
主程序：陈威志、李易儒
文档：　陈威志、李易儒

1.4 开发背景

a. 开发系统的名称：基于 Android 的趣味闹钟程序
b. 任务提出者：李易儒、陈威志
　开发者：李易儒、陈威志
　用户：Android 手机用户

1.5 使用说明

开始程序后将进入主界面，在主界面上能够查看当前有多少闹钟，且可以通过点击闹钟的图标来切换闹钟的激活与否。长按列表中的闹钟可以弹出菜单，在菜单中可以对此闹钟删除、修改、更改激活。点击主界面下的加号按钮可以添加新的闹钟，点击后将进入闹钟设置界面。在闹钟设置界面中可以对闹钟进行一些常规的设置，除此之外还可以通过最下面一行的按钮来选择以何种方式来解锁闹钟的闹铃。这些设置完成后，到了闹钟设置的时间，会弹出消息并同时发出闹铃，在此时可通过点击“解锁”按钮来进入已经选择好的闹钟解锁方式，来解锁闹钟，也可以点击暂停再响，过 10 分钟后继续提示，具体实现方法可参照视频演示。

在锁屏时，闹钟也同样可以响起。接电话时，闹钟会自动取消。操作别

的程序时，闹钟可以打断别的 activity 或者 service，自动响起。当两个闹钟设置成同样的时间响起时，会根据闹钟的 id，只有较新的闹钟会响起，先前设立的闹钟不会响。

2.系统结构

闹钟类 Alarm

public static final Parcelable.Creator<Alarm> *CREATOR*	CREATOR，序列化的 Parcelable 接口
public void writeToParcel(Parcel p, **int** flags)	闹钟的数据储存
public static class Columns **implements** BaseColumns	定义通用 id 和内容
public int id	闹钟 id
public boolean enabled	闹钟是否激活
public int hour	闹钟的小时
public int minutes	闹钟的分钟
public DaysOfWeek daysOfWeek	闹钟每周响的时间
public long time	闹钟每次响铃的间隔
public boolean vibrate	闹钟是否震动
public String label	闹钟的标签
public int mode	闹钟的解锁方式
public Alarm(Cursor c)	从数据库的内容中构造闹钟
public Alarm(Parcel p)	从服务器端内容构造闹钟
public Alarm()	默认的闹钟构造函数
static final class DaysOfWeek	闹钟在每周几响的处理

闹钟响起时动作处理 AlarmAlertFullScreen

protected Alarm mAlarm	声明闹钟
private int mVolumeBehavior	音量控制
protected void onCreate(Bundle icicle)	创建当前类，读出相应信息
private BroadcastReceiver mReceiver = **new** BroadcastReceiver()	接受闹铃的广播
private void setTitle()	设置弹出窗口的标题
private void updateLayout()	更新界面
private void snooze()	闹钟休眠一段时间后再响
private void dismiss(**boolean** killed)	取消闹钟
protected void onNewIntent(Intent intent)	在前一个闹钟未关闭时后一个闹钟响了的处理

闹钟后台运行类 AlarmKlaxon

private Vibrator mVibrator	闹钟的震动器
private MediaPlayer mMediaPlayer	媒体播放器
private Alarm mCurrentAlarm	当前的闹钟类
private long mStartTime	闹钟的启动时间
private TelephonyManager mTelephonyManager	电话状态管理
private boolean mCurrentStates	当前状态
private AudioManager mAudioManager = **null**	音乐播放器
private Handler mHandler = **new** Handler()	消息处理
private PhoneStateListener	手机状态的监听

mPhoneStateListener = **new** PhoneStateListener()	
private void play(Alarm alarm)	启动闹钟
private void startAlarm(MediaPlayer player)	在指定时间播放闹铃
private void setDataSourceFromResource(Resources resources，MediaPlayer player, **int** res)	从资源中读出闹钟设置属性

针对数据库得到闹钟信息类 AlarmProvider

private SQLiteOpenHelper mOpenHelper	闹钟信息存放的数据库
private static class DatabaseHelper **extends** SQLiteOpenHelper	闹钟信息存放数据库的操作
public boolean onCreate()	创建这个数据库
public Cursor query(Uri url, String[] projectionIn, String selection,String[] selectionArgs, String sort)	查询信息是否匹配
public int update(Uri url, ContentValues values, String where, String[] whereArgs)	更新数据库信息
public Uri insert(Uri url, ContentValues initialValues)	向数据库中插入信息
public int delete(Uri url, String where, String[] whereArgs)	从数据库中删除信息

闹铃接受类 AlarmReceiver

public void onReceive(Context	解析动作，开启相应的 activity

context, Intent intent)	
private NotificationManager getNotificationManager(Context context)	闹钟通知栏管理
private void updateNotification(Context context, Alarm alarm, **int** timeout)	更新闹钟通知栏的信息
private final static int *STALE_WINDOW* = 30 * 60 * 1000;	闹钟的时间假如超过这个值则忽略

设置闹钟类 SetAlarm

private EditTextPreference mLabel	闹钟的标签
private CheckBoxPreference mEnabledPref	闹钟是否激活
private Preference mTimePref	闹钟响铃的时间
private AlarmPreference mAlarmPref	闹钟的音乐
private CheckBoxPreference mVibratePref	闹钟是否震动
private RepeatPreference mRepeatPref	闹钟重复的日期
private int mId	闹钟的 id
private int mMode	闹钟的解锁方式
protected void onCreate(Bundle icicle)	建立闹钟，从 io 中读出闹钟设置信息
private static final Handler *sHandler*	设立消息传递器
public boolean onPreferenceChange(**final**	对设置内容的监听

Preference p, Object newValue)	
private void updatePrefs(Alarm alarm)	更新闹钟的信息
private void showTimePicker()	在布局文件中最下面显示当前时间
public void onTimeSet(TimePicker view, **int** hourOfDay, **int** minute)	时间的设置
private long saveAlarm()	保存闹钟
private void deleteAlarm()	删除闹钟
static void popAlarmSetToast(Context context, **long** timeInMillis)	设置完时间后显示通知

设置闹钟类 SettingsActivity

protected void onCreate(Bundle savedInstanceState)	创建 activity
public boolean onPreferenceTreeClick(PreferenceScreen preferenceScreen, Preference preference)	闹钟的数据处理
public boolean onPreferenceChange(Preference pref, Object newValue)	数据更改后操作
private void refresh()	界面的刷新操作

摇晃解锁 AlarmAlartShake

private MediaPlayer m	用于播放音频文件
public void onCreate(Bundle savedInstanceState)	加载 Activity,启动音频播放器，并调用 OnShake 检查是否摇晃

检查是否在摇晃 ShakeDetector

static final int UPDATE_INTERVAL = 100; static int count = 0;	检测的时间间隔
float mLastX, mLastY, mLastZ;	上一次检查时，加速度在 x，y，z 上的分量
public int shakeThreshold = 3500;	摇晃检测阈值，决定了对摇晃的敏感程度，越小越敏感。
public interface OnShakeListener{}	当摇晃事件发生时，接收通知
public void registerOnShakeListener(OnShakeListener listener)	注册 OnShakeListener，当摇晃时接收通知
public void unregisterOnShakeListener(OnShakeListener listener)	移除已经注册的 OnShakeListener
public void start()	启动摇晃检测
public void stop()	停止摇晃检测
public void onSensorChanged(SensorEvent event)	传感器值发生变化响应
private void notifyListeners()	摇晃事件发生，通知所有 listener

光强解锁 AlarmAlertLight

private SensorManager mSensorManager; private EditText etLight; private MediaPlayer m;	定义系统的 sensor 管理器 用于显示数据的文本编辑框 音频播放器
public void onCreate(Bundle savedInstanceState)	加载 Activity,启动音频播放器，获得传感器服务
protected void onResume()	为光强传感器注册监听器
protected void onStop()	程序退出取消传感器注册
protected void onPause()	程序暂停取消传感器注册

public void onSensorChanged(SensorEvent event)	更新数据，判断是否达到解锁条件

水平仪解锁界面 GradienterView

Bitmap back; Bitmap bubble; int bubbleX, bubbleY;	仪表盘图片 气泡图片 定义水平仪中气泡的位置
public GradienterView(Context context, AttributeSet attrs)	加载水平仪和气泡图片
protected void onDraw(Canvas canvas)	绘制仪表盘和气泡

水平仪解锁 Gradienter

Timer timer = new Timer();	Timer 计时器
private MediaPlayer m;	音频播放器
int timeclock = 0;	时间累加器
int MAX_ANGLE = 30;	定义水平仪最大倾斜角度，超过这个角度认为在边界上
SensorManager mSensorManager;	系统的传感器管理器
public void onCreate(Bundle savedInstanceState)	加载音频播放器，获取水平仪组件，获取传感器服务
class MyTask extends TimerTask	计时器响应事件，每个一秒计时器累加 1
public void onResume()	为方向传感器注册监听器
protected void onPause()	取消注册
protected void onStop()	停止时，取消注册
public void onSensorChanged(SensorEvent event)	方向传感器变化，获取新的角度，计算气泡的位置，重绘界面，并判断是否达到解锁标准

private boolean isContain(int x, int y)	计算 X，Y 点的气泡是否在水平仪内

显示闹铃对话框 AlarmAlert

private final int MAX_KEYGUARD_CHECKS = 5;	定义 keyguard 检查次数
private final Handler mHandler = new Handler()	初始化 handler
private final BroadcastReceiver mScreenOffReceiver = new BroadcastReceiver()	接受 keyguard 广播服务
protected void onCreate(Bundle icicle)	监听屏幕，当屏幕唤起，用户不需要按解锁键取消闹钟
public void onDestroy()	取消注册监听器
private boolean checkRetryCount()	检查是否大于规定次数
private void handleScreenOff(final KeyguardManager km)	未大于检查次数，通知启动全屏解锁程序

闹铃唤醒解锁 AlarmAlertWakeLock

static void acquireCpuWakeLock(Context context)	加载一个唤醒的锁，使其可以接受 AlarmReciver
static void releaseCpuLock()	释放 AlarmAlert 活动

闹钟初始化接收器 AlarmInitReceiver

public void onReceive(Context context, Intent intent)	启动后移除再响的闹钟，设置 Alarm 为 ACTION_BOOT_COMPLETED

闹钟参数 AlarmPreference

protected void onSaveRingtone(Uri ringtoneUri)	因为 RingTonePreference 不提供设置铃声，所以重写函数，设置铃声
protected Uri onRestoreRingtone()	同样重写函数，存储设置的铃声
public void setAlert(Uri alert)	设置铃声
public Uri getAlert()	获得铃声

管理闹钟 Alarms

public static final String *ALARM_ALERT_ACTION* = "com.cn.daming.deskclock.ALARM_ALERT";	从闹钟管理器接收广播，用于触发 AlarmReceiver 和 AlarmKlaxon
public static final String *ALARM_DONE_ACTION* = "com.cn.daming.deskclock.ALARM_DONE";	当闹钟停止响，由 AlarmKlaxon 发送的动作
public static final String *ALARM_SNOOZE_ACTION* = "com.cn.daming.deskclock.ALARM_SNOOZE";	AlarmAlertFullScreen 发送，当闹钟暂停再响时，别的类可以接收
public static final String *ALARM_DISMISS_ACTION* = "com.cn.daming.deskclock.ALARM_DISMISS";	AlarmAlertFullScreen 发送，当闹钟解锁时，别的类可以接收
public static final String *ALARM_KILLED* = "alarm_killed";	当闹钟被删除时，通知 AlarmKalxon 更新 UI
public static final String *ALARM_ALERT_SILENT* = "silent";	闹钟在数据库中
public static final String *CANCEL_SNOOZE* = "cancel_snooze";	用户取消闹钟再响
public static long	创建一个新 Alarm，并设置 id

addAlarm(Context context, Alarm alarm)	
public static void deleteAlarm(Context context, **int** alarmId)	移除一个已经存在的 Alarm，如果处于再响状态则取消
public static Cursor getAlarmsCursor(ContentResolver contentResolver)	将所有闹钟入队
private static Cursor getFilteredAlarmsCursor	从数据库中返回闹钟的集合
private static ContentValues createContentValues(Alarm alarm)	如果时间未重复则设置闹钟时间
private static void clearSnoozeIfNeeded(Context context, **long** alarmTime)	如果取消闹钟，则同时取消再响状态
public static Alarm getAlarm(ContentResolver contentResolver, **int** alarmId)	根据 ID 返回数据库中所有的闹钟，如果没有则返回 null
public static long setAlarm(Context context, Alarm alarm)	早 Alarms 中设置一个 Alarm
public static void enableAlarm(	激活或者取消一个闹钟
private static void enableAlarmInternal(**final** Context context, **final** Alarm alarm, **boolean** enabled)	激活一个闹钟，并更新时间
public static Alarm calculateNextAlert(**final** Context context)	计算下一次响铃的时间
public static void disableExpiredAlarms(**final** Context context)	取消已经过去的非重复闹钟

public static void setNextAlert(final Context context)	当系统时区或时间变化时，重设闹钟时间
private static void enableAlert(Context context, final Alarm alarm, final long atTimeInMillis)	在状态来中设置闹钟是否被激活
static void saveSnoozeAlert(final Context context, final int id, final long time)	保存需要再响的闹钟
private static boolean enableSnoozeAlert(final Context context)	启动需要再响的闹钟
static Calendar calculateAlarm(int hour, int minute, Alarm.DaysOfWeek daysOfWeek)	返回一个时间，以通知 AlarmManager

闹钟程序主活动 DeskClockMainActivity

public void onCreate(Bundle savedInstanceState)	分别加载 LayoutInflater 用来生成 layout 和 SharePreference 管理数据，和从闹钟队列中获得闹钟
private void updateLayout()	加载界面生成闹钟 list
private void addNewAlarm()	添加一个新闹钟
private class AlarmTimeAdapter extends CursorAdapter	设置列表图片，初始化 checkbox，并且生成编辑对象，设置闹钟参数
private void updateIndicatorAndAlarm(boolean enabled, ImageView bar, Alarm alarm)	设置时间，并弹出消息
public boolean onContextItemSelected(final MenuItem item)	设置选择对象，
public void onCreateContextMenu(ContextMenu menu, View view,	从 xml 获取菜单，用当前的选择创建用户视图，以当前时间初始化日历，设置标签的文本框内信息，

ContextMenuInfo menuInfo)	
public boolean onOptionsItemSelected(MenuItem item)	根据不同按钮启动不同 activity

帮助点击 CheckBox DontPressWithParentLayout

public void setPressed(**boolean** pressed)	当点击 checkbox 时候是不触发 parent 的 onclick 事件

传递设置时钟信息 HandleSetAlarm

protected void onCreate(Bundle icicle)	加载 activity，初始化事件，调用 ContentValues 存储名值

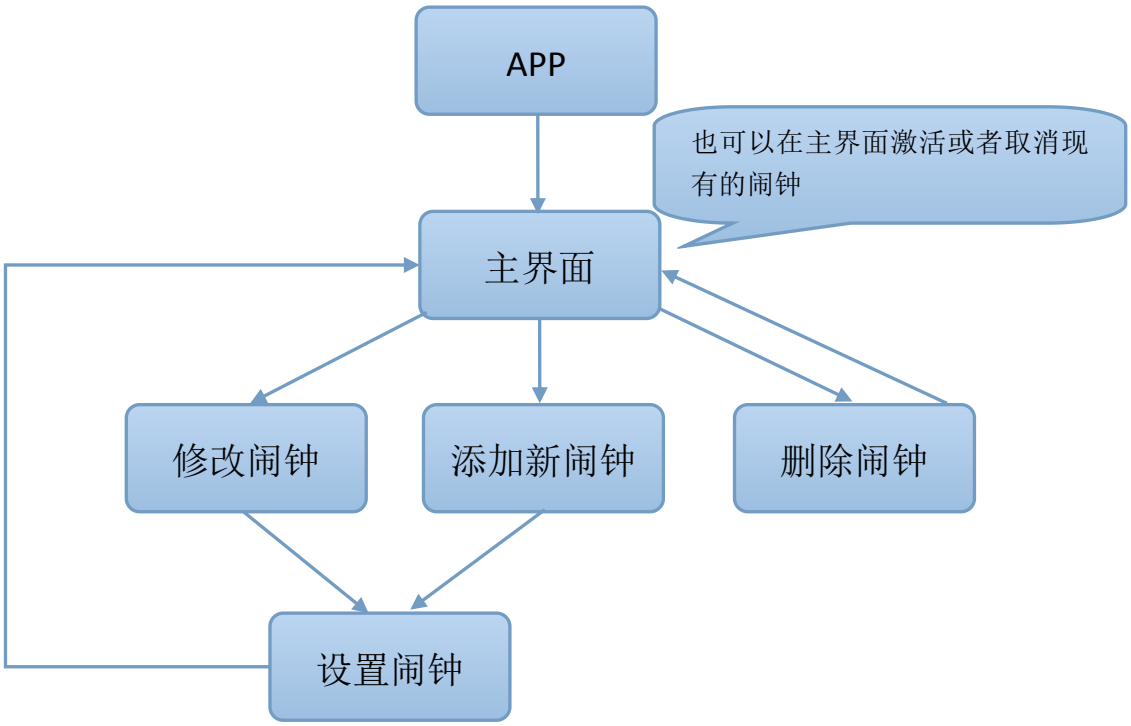
重复时间设置 RepeatPreference

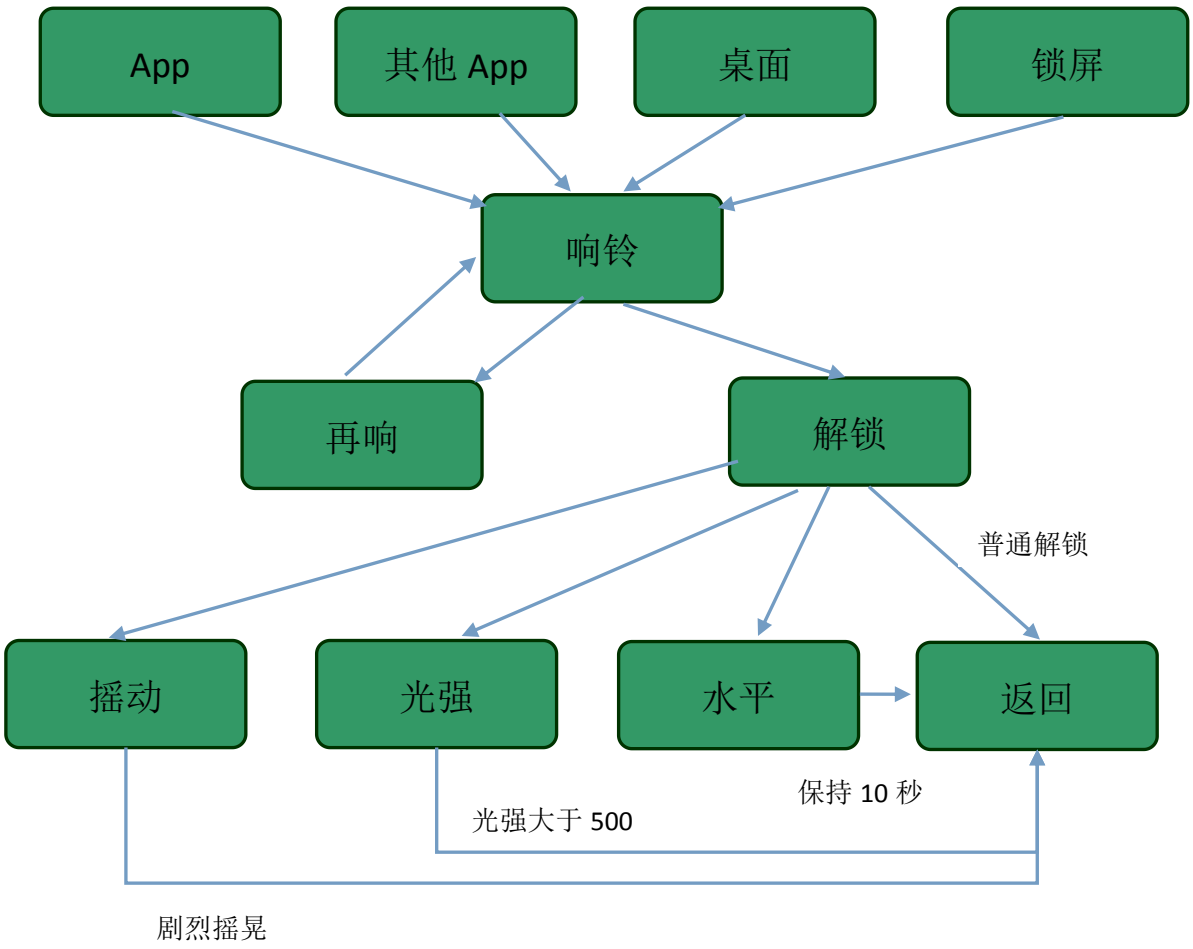
public RepeatPreference(Context context, AttributeSet attrs)	设置一周每天的信息
protected void onDialogClosed(**boolean** positiveResult)	关闭 listView 时保存信息
protected void onPrepareDialogBuilder(Builder builder)	初始化对话框，点击出现星期列表
public void setDaysOfWeek(Alarm.DaysOfWeek dow)	分别设置原来的值，和从对话框中获得的新值

数字时钟 DigitalClock

消息管理 ToastMaster

3.程序流程图


APP
也可以在主界面激活或者取消现有的闹钟
主界面
修改闹钟
添加新闹钟
删除闹钟
设置闹钟


App
其他 App
桌面
锁屏
响铃
再响
解锁
普通解锁
摇动
光强
水平
返回
保持 10 秒
光强大于 500
剧烈摇晃

3.程序截图

图 1 程序主界面

图 2 闹钟设置界面

图 3 闹钟响起的界面

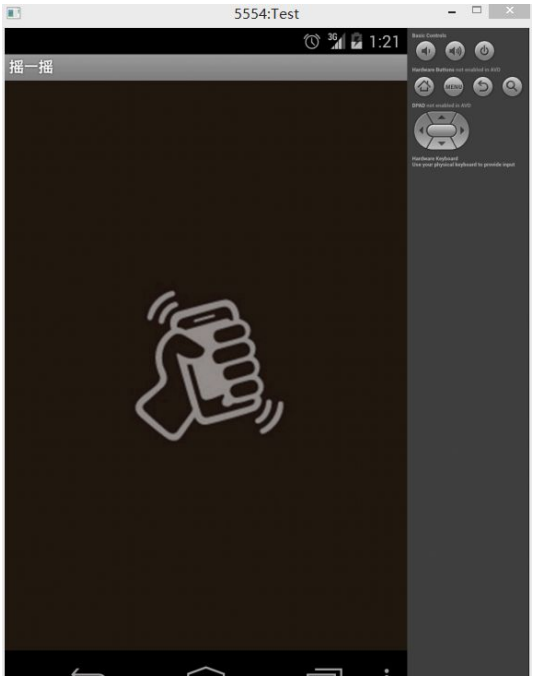


图 4 使用振动方式解锁闹钟的界面

图 5 使用光强方式解锁闹钟的界面

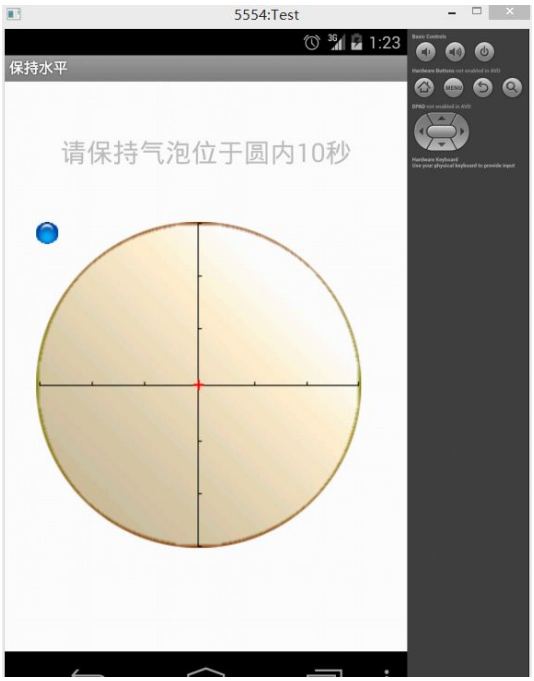


图 6 使用水平仪解锁闹钟的界面

4.心得 体会

在本次程序的编写中，我们遇到了不少问题，但是这些问题通过我们团队两个人的合作和思考，这些问题最终都迎刃而解，耗费的精力换来了巨大的收获，我们不仅仅是成功的从无到有编写出来了一个功能比较完整、设计比较新颖的程序，而且学会了如何在 android 平台上进行编写、测试程序。并且在问题的解决过程中，通过不断的沟通和交流，我们也了解到了一个团队在完成一个程序的过

程和艰辛，学会了合作的重要性。

在实验完成的过程中，我们一定程度上的参考了谷歌官方的 refercence 文档，学习了 android 平台上原生的闹钟程序，在夯实的基础上开着手于程序的编写。同样的我们在一边编写的过程中一遍提升对自己程序代码的理解，感慨着 android 原生中 api 的丰富和强大。

总而言之，通过本次项目的开发，我们学习了不少 android 的知识，积累了不少 android 开发的经验。在短短的半个学期的学习中，收获了很多，学到了很多。