Extracted from:

Developing Android on Android

Automate Your Device with Scripts and Tasks

The Pragmatic Bookshelf

Dallas, Texas • Raleigh, North Carolina

The Pragmatic Programmers

Developing Android on Android

Automate Your Device with Scripts and Tasks

Mike Riley

edited by Jacquelyn Carter

Developing Android on Android

Automate Your Device with Scripts and Tasks

Mike Riley

The Pragmatic Bookshelf
Dallas, Texas • Raleigh, North Carolina

The team that produced this book includes:

Jacquelyn Carter (editor)
Potomac Indexing, LLC (indexer)
Molly McBeath (copyeditor)
David J Kelly (typesetter)
Janet Furlow (producer)
Juliet Benda (rights)
Ellie Callahan (support)

Printed in the United States of America.
ISBN-13: 978-1-937785-54-3
Encoded using the finest acid-free high-entropy binary digits.
Book version: P1.0—November 2013

This book is dedicated to my three favorite M’s: Marinette, Marielle, and Mitchell.

3.3 Button Control

As I write this chapter, Google is ramping up hype for its impending release of Google Glass, its self-proclaimed game-changing wearable-computing device. Having been professionally around the computing block for nearly thirty years, I have seen the rise and fall of a variety of devices. I worked with Windows CE-powered wearable-computing head-mounted displays over a decade ago, and while it was a thrill developing applications on bleeding-edge technology at the time, those types of displays ultimately lost their luster for me. Why?

The most notable reasons are centered around eyestrain, obstructed views, and the uncomfortable feeling of a headband gradually making a dent into my forehead. While I still fantasize for the day when I can walk around like a futuristic superhero with total situational awareness, swirling graphics, flagging call-outs, and an always-on voice assistant ready to immediately act upon my lazy utterances, we still have a long way to go technologically before those science-fiction visions are fully realized. But while it's fun to dream of the future, I need an unobtrusive working solution today without all the annoyances I've already encountered.

Headset Button Controller,[7] created by Android developer Christoph Kober, has become my control interface of choice. This helpful piece of software allows you to assign tasks, scripts, and applications to execute on your phone with a click of your built-in headset button. I have assigned it to execute all of my hands-free, vision-free automation tasks, from telling me the time and current battery charge level of my device to checking for new email messages and Twitter posts. And because text-to-speech technology has improved so much in the last few years, hearing your phone or tablet read you lengthy passages of content isn't as monotonous as it sounds.

So Long, Long Button Press

Prior to the release of Android 4.2, Headset Button Controller provided considerably more flexibility and button press combinations to assign tasks and trigger actions. Unfortunately, the long press headset button event is no longer an option in Android 4.2 and newer. That's because Google has now reserved that single long-click action to launching Google Now from a headset. Even if you attempted a long press after a series of short presses, Google steps in as soon as that long press is recognized and takes over. By doing so, Google has severely constrained Headset Button Controller's combination of buttons to essentially four assignments for single-button headsets

7. https://play.google.com/store/apps/details?id=com.kober.headsetbutton

(plus two more for the headset plug-in and unplug events) and up to sixteen (along with the plug-in and unplug assignments) for those devices that include a volume control along with the middle button (since there can be up to four short-press assignments per button).

Google intended to capture the long-press headset button event to offer a consistent behavior across devices using Android 4.2 and newer. But forcing this upon all users without the ability to disable or reassign to a different button combination seems a tad heavy-handed. I hope Google will allow users to customize this choice in future iterations of the operating system, but for now we have to accept this imposed constraint.

As odd as it sounds, I look forward to hearing the synthetic female voice used as the default TTS engine in Android 4.2. It's nice to know that she's always there, keeping me informed of appointments, messages, to-dos, and directions. And like Aladdin rubbing a lamp to wake the genie, I use my assigned clicks interpreted by Headset Button Controller to instantiate a variety of automated workflows. Some of these workflows will be described in later chapters. But for now, think of the variety of click combinations to launch applications as a terse subset of Morse Code. One click to pause, two clicks to fast-forward, three to report time and battery life, and four to check email. We'll talk more about the last two custom programs in that click list later in the book.

Before purchasing the commercial version of Headset Button Controller, it's best to test the trial version to verify compatibility with your Android device. You will also need a headset with at least one inline button that you can use to initiate click events.

For example, my Galaxy Nexus phone supports only a single button headset, leaving me with a total of six actions I can assign to Headset Button Controller. So, grab your Android phone, plug in a compatible headset, install the trial edition of the button controller software, and let's configure it to launch the standard Android web browser when three headset button presses are detected.

Configuring the Headset Button

Before we can use the Headset Button Controller, we first need to configure it. Launch the Headset Button Controller program and select the easy tab, as shown in the figure here.

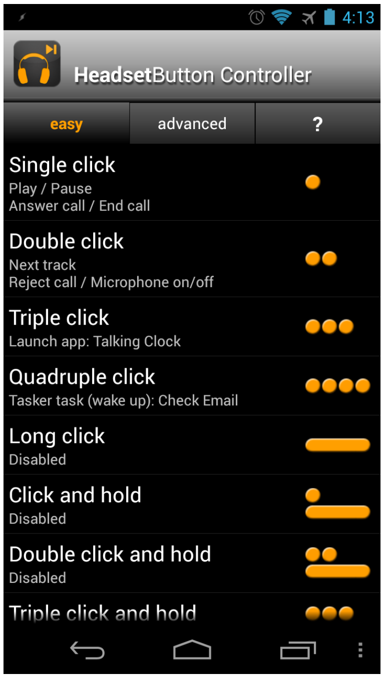
4:13
HeadsetButton Controller
easy | advanced | ?
Single click
Play / Pause
Answer call / End call
Double click
Next track
Reject call / Microphone on/off
Triple click
Launch app: Talking Clock
Quadruple click
Tasker task (wake up): Check Email
Long click
Disabled
Click and hold
Disabled
Double click and hold
Disabled
Triple click and hold

Figure 17—Headset Button Controller easy tab

For our test, we will assign an action to open the web browser application. Select the "Triple click" option on the screen. This will display a dialog of various commands that can be assigned to that action, as shown in Figure 18, *Headset Button Controller easy tab*, on page 8.

Select the "Launch app" option. Doing so will display a list of applications installed on your device. Choose the Browser application. An alert like the one shown in Figure 19, *Allowing Android to wake up when launching a task*, on page 8, will pop up asking whether you want your phone to wake up when the launch action is started.

Choosing to wake the phone when the action is initiated will turn on the screen and wake up the WiFi or cellular radios if they were in a low-power state. This way, when your browser's assigned home page is requested, the screen will be visible, and the network will be actively retrieving the content.

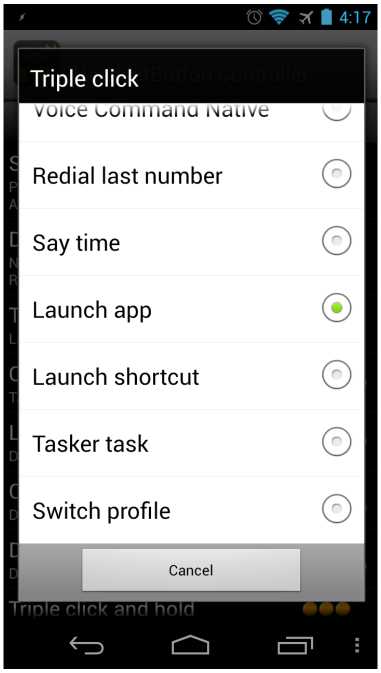

Figure 18—Headset Button Controller easy tab

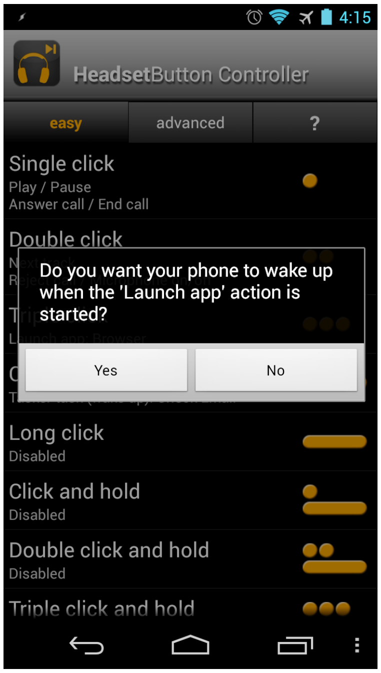

Figure 19—Allowing Android to wake up when launching a task

Operating the Headset Button

If you haven't already done so, plug your headset into your Android device and triple-click the headphone's center button. As long as your headset hardware is compatible with the Headset Button Controller program, this action should launch and display the web browser. Congratulations, you now have the ability to initiate events without ever having to touch your phone's screen. This action will work whether or not your Android's screen is locked. We will leverage this newfound power to instantiate scripts and programs we will write later in the book.

Keep in mind that we can still use other applications that react to headset button presses while Headset Button Controller is running. You can have the program send button presses to a specific application (such as an audio player) or to whatever media player had been most recently used.

The most popular scenarios that have been enhanced by using the Headset Button Controller are for managing audio playback while listening to audiobooks, music, and podcasts. For example, you can control audio playback using commercial applications like Alex Kravchencko's Smart AudioBook Player (shown in the figure here),[8] or the DoggCatcher podcast program can be controlled entirely from the headset button.[9] A single click will pause and play the audio. Need to skip over boring commentary? Assign that action to a double-click of the headset button.

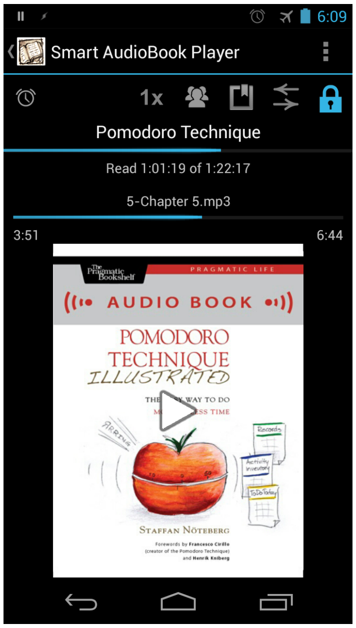

Figure 20—The smart audiobook player

Headset Button Controller even has a built-in speak-the-time function using Android's text-to-speech feature, but I find the program's implementation somewhat lacking. Consequently, I wrote my own talking clock application that I have assigned to a triple-click of the headset button. We will walk through the creation of this program and its configuration with Headset Button Controller in later chapters.

Distraction-free applications like the Headset Button Controller do a remarkable job of relieving you from having to fumble with your phone every time you want to skip over a song, answer a call, or check the time. It also elevates your phone to wearable-computing status as you realize how, using the right mix of applications and workflows, a good deal of information can be processed without ever having to look at a display. It's quite a liberating feeling as you become more skilled with this level of audio-delivered information optimization.

8. https://play.google.com/store/apps/details?id=ak.alizandro.smartaudiobookplayer
9. https://play.google.com/store/apps/details?id=com.snoggdoggler.android.applications.doggcatcher.v1_0

Soldering More Buttons

After getting over the dismay of losing the ability to reassign a single long press headset button in Android 4.2, I came across a post on the excellent Instructables.com website that reexpanded my options.[a] While I can only vouch for the effectiveness of the Galaxy Nexus, the author of the instructions indicates that it should work with other Android phones as well.

The idea is to take an inexpensive three-button inline remote designed for Apple products like the iPod touch or iPhone (the article suggests the iLuv iEA15BLK). Open the button casing, solder in two tiny resistors, and you can triple your Headset button options as a result.

The procedure does require soldering skills, steady hands, and good eyesight to be able to set the spec-sized resistors in place, but it worked like a charm for me and gave me another sixteen actions to assign to running various programs and tasks. If you enjoy tinkering with hardware and have the right equipment, try this approach.

Many thanks to my good friend and tech extraordinaire John Winans for use of his top-notch soldering equipment and laser-eye soldering technique. I couldn't have verified that this Instructables technique worked without his help.

a. http://www.instructables.com/id/Galaxy-Nexus-and-others-headset-remote-with-medi/