Smart Jacket App

Delivery description version 1.3.1

Device Specifications

Device Model: Motorola Moto G (2nd gen)
Operating System: Android 6.0
Screen SIze: 720 x 1280 pixels

Delivered Assets

The following assets are delivered as part of the task:

- The app as an Android package (APK) to be installed onto a device.
- The structured and commented source code for the app, as a Processing (Android Mode) project.
- The structured and commented source code for the jacket-side BLE logic as an Arduino project for the “Sparkfun ESP-32 Thing” board.

The following documents are included

- The present document.
- A list of source code files and brief description.

- An overview of the Bluetooth LE Service for the smart jacket; and the supported aspects of the JSON-based communication protocol between app and jacket.

Source Code

The source-code for the app is provided as a project for the Processing IDE in Android Mode (see http://android.processing.org/). The source code is structured and sufficiently commented. As a Processing project (commonly referred to as “sketch”) the source code is split into *.pde* files. When building the app, Processing combines these inside a single class (with the project’s name, and extending the base class *PApplet*). This means that all code, including classes, declared in the *.pde* files will be nested in the app’s main class.

If needed, the project can be edited and built in the Processing environment’s Android Mode, which is fairly easy to install. It can also be setup for Android Studio, although this takes some additional one-time effort (see http://android.processing.org/tutorials/android_studio/).

The source code for the jacket’s BLE stack is provided as a project for the Arduino IDE. The project is targeted at the *Sparkfun ESP32 Thing* board and is based on the Arduino core library for the ESP32 (see https://github.com/espressif/arduino-esp32).

Installation and Setup

Requirements

The app is made for a device running Android 6 (API 23). The device must have Bluetooth Low Energy (BLE) capabilities. It may be possible to run the app in Android OS version 4.3 (API 18) or above, but this should be tested in advance.

Installation

The app is provided as a debug APK.

The phone’s developer mode must be unlocked. In the Developer Mode settings, USB debug must be enabled and allowed to install unsigned/debug apps.

An APK can be installed via USB using the Android Debug Bridge (ADB), or from the device itself using an adequate app.

- **Installing from a computer onto a device**. The app’s APK file is installed via USB, using the Android Debug Bridge (ADB). ADB is part of the Android SDK, but can also be found as stand-alone. The command is “adb install <filename>.apk” (documentation here). On Windows, you may need to install an ADB-compatible driver for your phone.
- **Installing from the device**. The APK can be placed in the phone’s file system (via USB, downloaded from cloud storage, etc.) and then installed via an app (for instance, APK

Installer). Some versions/flavours of the Android OS may already have a file manager app that is able to install the APK.

Permissions

Before using the app, make sure that **both bluetooth and location** are enabled on the device. Due to a bug in Processing Android Mode, it is not currently possible to enable these from within the app (or the app would have to be restarted by the user afterwards).

The app will request the following permissions, which should be granted otherwise the app won't work:

- Call Phone - required to place emergency calls.
- Write External Storage - required to create and read from the settings file.
- Access Coarse Location - required for BLE scanning.

App-Jacket Communication

The wireless communication between App and Jacket is done via Bluetooth Low Energy (BLE). The Jacket offers a custom BLE service, as specified in a separate document. Briefly, this service has characteristics for the current status of the jacket (and training plan) which can be read by the app; and characteristics allowing the exchange of messages between client and server.

The app **reads status information from the jacket**, in order to display training state or react accordingly to changes.

The app **sends commands to the jacket to upload a training plan**; the Jacket should reply with an "acknowledge" message.

Other commands (send a phase, start a plan, skip a phase) are **not implemented** in the app. Controlling training progression (start, stop, pause) is done through the jacket's tactile interface.

The app handles "signal" messages from the jacket, of the type that signals a "button" (e.g. "a", "ab", "a2b", "ice"). The corresponding action is defined in the settings file (see below). The supported actions are:

- "playPause": play/pause media, currently queued or playing in another app.
- "volUp": increase the media volume.
- "volDown": decrease the media volume.
- "fastCall": place a phone call to the current buddy.

These will be received and handled even if the app doesn't have focus; until it is destroyed (removed from running memory to accommodate another launched app) by the Android OS, or

explicitly stopped by the user (by swiping it from the “task manager” view, or via “force stop” in the app settings).

Settings, Plans and Contact List

The app reads its settings from a file in the phone’s internal storage.

The settings file is JSON formatted and contains: contacts (name and phone number); training plans (to be displayed in the app); and mapping between “buttons” sent by the Jacket (e.g. “ab” or “ice”) and actions taken by the app (e.g. “playPause” or “fastCall”). It also references the last selected buddy (by name) and connected device (name and address) if any.

The app will search for the configuration file in the phone’s storage root, under the following path:

Smart Jacket\settings.txt

If the folder and/or file do not exist, they will be created by the app and populated with default data. The default data is read from a file in the app’s own resources (the file “settingsDefault.txt” which can be found in the “data” folder inside the project).

The settings file is a JSON object, containing the following fields. Fields marked with an asterisk are optional.

Field	Type	Description
plans	JSON Array	An array of “plan” objects, as defined in a separate document.
contacts	JSON Array	An array of contact objects. Each object has “name” and “number” fields, both Strings, for the contact’s name and phone number, respectively.
buttonActions	JSON Object	An object serving as a map between button strings sent by the jacket and action strings interpreted by the app.
contactName*	String	The name of the last selected conatct, updated by the app.
deviceName*	String	The name of the last connected device, updated by the app.
deviceAddress*	String	The address (MAC) of the last connected device, updated by the app.
deviceScanInterval	integer	The timeout for BLE device scans, in milliseconds.

deviceUpdateInterval	integer	The period for updating status from the jacket.
connectViewTimeout	integer	The timeout for the connecting view, in milliseconds.
verbose	boolean	Whether to log more debug information.