ANDROID APPLICATION FOR MONITORING THE STATUS OF THE ADVANCED PHOTON SOURCE

M. Borland*, Westmont, IL, USA

Abstract

Smartphones and tablets are nearly ubiquitous and have the ability to quickly access and display data from network sources. This suggests their suitability for remote monitoring of a facility such as the Advanced Photon Source (APS). While one possibility is to access data using a web browser running on the device, native applications offer attractive features, such as improved display and background execution. We report on development of an Android application for monitoring the status of the APS. In addition to displaying data, the application can issue alerts when beam is lost or restored. Home screen widgets of various sizes are also provided. We describe not only the features of the application, but also give details of the implementation. The application is free of advertising and is available free of charge on the Google Play store.

INTRODUCTION

Like many synchrotron radiation facilities, the Advanced Photon Source (APS) operates thousands of hours per year. During operating periods, it is expected to provide beam 24 hours a day, seven days a week. Getting information about status to accelerator staff, beamline staff, and beamline users is thus very important. The ubiquity of smartphones and tablets provides a way to do this.

These devices are internet-enabled, so it is of course possible to get information via a web browser. However, using a native Android application provides several advantages. One can provide high-level status on the notification bar, which is visible without unlocking the phone or opening the application. One can provide a home-screen widget that provides more information than will fit on the notification bar. The notification and widget can both be used to launch the full application. Another advantage of the native application over browser-based information delivery is that the native application can run in the background and provide audible alerts of significant events.

Inspired by these ideas, we developed an Android application that provides both high-level and detailed status data about the APS. The Android platform was chosen for its open-source nature and the low cost of entry for devices and development. Also, the Android home-screen widget feature provides a superior way to display very high level data without opening the application. Unlike other smartphone platforms, the Android Software Development Kit (SDK) is free and available for essentially all platforms, which allowed using Linux for development.

The application is called APSStatus and available free of charge on the Google Play store. The application is also free of advertising.

* michael.d.borland@gmail.com

FEATURES AND CAPABILITIES

The APSStatus package includes a main application and three home-screen widgets. The main application is organized into a series of pages or tabs, each of which displays information of a certain type. Presently there are tabs for overall status, status messages, 24-hour and 1-week history plots, superconducting undulator status, power supply summary status, power supply detailed status, vacuum system status, rf system status, personnel safety system status, and front-end equipment protection system status. Figure 1 shows the main status screen. This is typical of the status screens, in that it features a heading with a prominent date and time stamp, which allows the user to easily ascertain if the data is fresh. Data is updated at 1 minute intervals.

Since not all of the selections of data will be of interest to all users, the application menu's "Tab Settings" selection allows the user to determine which tabs will be displayed.

Most of the tabs supply snapshots of data without any history. One exception is the status messages display, which displays the history of recent status updates issued by control room staff. Two other exceptions are the 24-hour and 1-week history tabs, which provide plots of beam current, beam lifetime, rms beam motion, number of shutters open, horizontal emittance, and vertical emittance. Figure 3 shows an example of the 1-week history display. Touching a plot will bring up a zoomed-in version with pinch-to-zoom and scrolling features.

In addition to the main application, the user can add one or more of the three home-screen widgets to their home screen. These display small selections of data and require less than 300 bytes per update, i.e., less than 0.4 MB per day. Hence, these can be left running and provide the most essential information at a glance.

The largest of these widgets, which has some special features, is shown in Figure 4. It shows the beam current color-coded by status, the number of bunches, the beam lifetime, the number of operating beamlines, the horizontal and vertical emittance, and the injection efficiency on the last top-up shot. Unlike the other widgets, this widget also posts the beam current to the notification bar, where it can be seen from any home screen and even when the screen is locked. The beam current notification is color-coded to indicate whether it is within the expected range for the present machine mode or not.

Another special feature of this widget is that it can provide audible alerts when beam is lost or restored. The alerts are configured from the main application's menu. The user can elect to be alerted when beam is lost, when it is absent for an extended time, and when it is restored. The user can also set a quiet-time preference to avoid being disturbed while sleeping, for example.

SOFTWARE DESIGN

Those who are uninterested in software design may wish to skip this section, as it has little to do with how the application is used or what it can do. This section also assumes some familiarity with Android development.

We used the Eclipse Integrated Development Environment (IDE), which is advantageous in that it helps one quickly learn both the Java language and the specific methods provided by Android. The web contains many resources, such as Android tutorials and user forums (e.g., StackExchange).

The main application is organized into a series of Fragment objects, which are grouped together as a single application. Each Fragment displays a screen with data of a particular type, e.g., power supplies or vacuum. The interface is built using Peter Kuterna's SwipeyTabs class and SwipeyTabsSampleActivity, which is an extension of the FragmentActivity class. SwipeyTabs provides a convenient mechanism for switching between related activities by clicking on a tab at the top of the screen or swiping the screen left or right.

SwipeyTabs keeps three Fragments active at any time. These are the Fragment presently being viewed, plus the Fragments to the left and right. If the user swipes or tabs to the Fragment on the right, say, the new Fragment is generally displayed instantaneously because it is already active. Meanwhile, the next Fragment is triggered so it is more likely to be ready when the user swipes again. (If the user swipes too quickly, this may not be the case.)

The data that is displayed by APSStatus and its various widgets is provided by the APS web server from files written by a node that has read access to EPICS process variables and write access to the web server's file space. A common choice for sharing data on the web is XML (eXtensible Markup Language) files and APSStatus was originally written to use such files. However, XML has a well-deserved reputation for verbosity, resulting in a very poor ratio of data delivered to bandwidth used. Since excessive use of bandwidth is highly undesirable for mobile devices, we switched to gzipped SDDS (Self-Describing Data Sets) files. These have the additional advantage that they are easily created using the SDDS tools [1, 2] that are already part of the APS control system. There is also a convenient SDDS library for Java that reads from a URL directly [3].

Fragments are at present divided into three types: (1) those that read and display data from an SDDS file. (2) those that retrieve and display bitmap images. (3) those that retrieve and display text from the server. Most Fragments are of Type 1.

Each Type 1 Fragment has a corresponding SDDS file on the server that provides its data. This is done in part because Fragments do not share data, so there is no point in reading data from a common source. Also, reading from a common source would invariably mean using more bandwidth than needed. Hence, each Fragment is supplied with a minimalist data file containing only the relevant information. Each file contains a time-stamp, which is prominently displayed along with the other data. This ensures that the user will be able to tell if the data is stale for some reason. The update interval of the data is 1 minute. Updates for each active Fragment are handled by separate threads.

The displays for Type 1 Fragments require only stock Android widgets. In order to accommodate various screen sizes, the root widget is a ScrollView, which permits vertical scrolling. Inside the ScrollView are TextView widgets, often packed into TableLayout widgets.

The Type 2 Fragments display a series of small bitmaps that are provided directly by the server, i.e., they do not retain or plot the history data themselves. When a small bitmap is pressed by the user, a larger bitmap of the same data is retrieved and displayed. This reduces use of bandwidth and memory by not transmitting and storing large bitmaps that may be of no interest.

In addition to the main application, there are three home-screen widgets that provide various selections of very high-level information. The home-screen widgets are implemented using the AppWidgetProvider class and run independently of the main application. As with the main application, the home-screen widgets have a 1 minute update interval. Touching any of these widgets will open the main application.

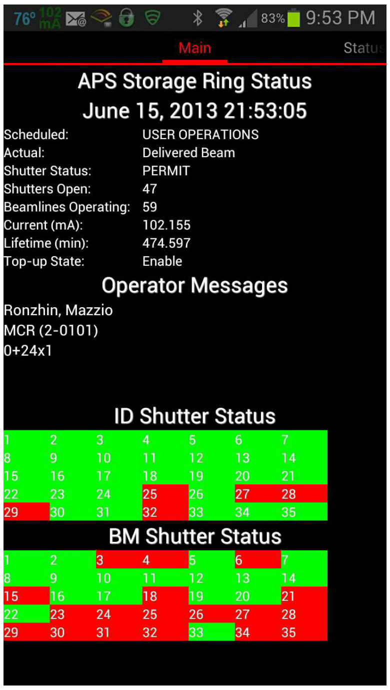
Of the three home-screen widgets, one is larger and provides more information. It also provides an alarm service, which is configured from the main application, and places information in the notification bar. The alarm settings are communicated from the main application to this widget using the SharedPreferences facility.

On the server side, a cron job runs at 1 minute intervals to make the various SDDS data files. For each data file, there is generally a separate request file that is used in an invocation of `sddscasr` to create a snapshot. The snapshots are processed to, in particular, convert status and floating point data to a uniform format. This simplifies coding on the application side and also reduces the size of the data transmitted. Small data files just for the home-screen widgets help minimize use of bandwidth.

Other cron jobs run at intervals to create the bitmaps needed by the history displays. These jobs used pre-existing scripts that invoke the program `sddsplot`.

CONCLUSIONS

We have created an Android application that provides status information about the Advanced Photon Source, including history graphs and detailed status of major subsystems. The application includes home-screen widgets, notification bar posts, and optional audible alerts of beam loss or restoration. Bandwidth use is minimized in part through use of gzipped SDDS files. The application is available free of cost on the Google Play store, and is free of advertising.



APS Storage Ring Status
June 15, 2013 21:53:05

Scheduled:	USER OPERATIONS
Actual:	Delivered Beam
Shutter Status:	PERMIT
Shutters Open:	47
Beamlines Operating:	59
Current (mA):	102.155
Lifetime (min):	474.597
Top-up State:	Enable

Operator Messages
Ronzhin, Mazzio
MCR (2-0101)
0+24x1

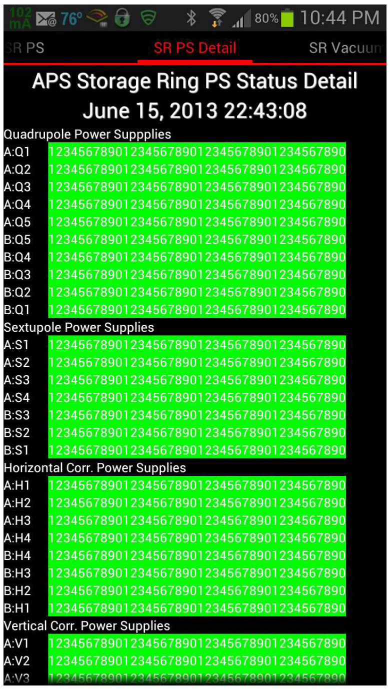
ID Shutter Status

1 🟩	2 🟩	3 🟩	4 🟩	5 🟩	6 🟩	7 🟩
8 🟩	9 🟩	10 🟩	11 🟩	12 🟩	13 🟩	14 🟩
15 🟩	16 🟩	17 🟩	18 🟩	19 🟩	20 🟩	21 🟩
22 🟩	23 🟩	24 🟩	25 🟥	26 🟩	27 🟥	28 🟥
29 🟥	30 🟩	31 🟩	32 🟥	33 🟩	34 🟩	35 🟩

BM Shutter Status

1 🟩	2 🟩	3 🟥	4 🟥	5 🟩	6 🟥	7 🟩
8 🟩	9 🟩	10 🟩	11 🟩	12 🟩	13 🟩	14 🟩
15 🟥	16 🟩	17 🟩	18 🟥	19 🟩	20 🟩	21 🟥
22 🟩	23 🟥	24 🟥	25 🟥	26 🟥	27 🟥	28 🟥
29 🟥	30 🟥	31 🟥	32 🟥	33 🟩	34 🟥	35 🟥

Figure 1: The main status tab of the APSStatus application.



APS Storage Ring PS Status Detail
June 15, 2013 22:43:08

Quadrupole Power Suppplies

A:Q1	1234567890123456789012345678901234567890
A:Q2	1234567890123456789012345678901234567890
A:Q3	1234567890123456789012345678901234567890
A:Q4	1234567890123456789012345678901234567890
A:Q5	1234567890123456789012345678901234567890
B:Q5	1234567890123456789012345678901234567890
B:Q4	1234567890123456789012345678901234567890
B:Q3	1234567890123456789012345678901234567890
B:Q2	1234567890123456789012345678901234567890
B:Q1	1234567890123456789012345678901234567890

Sextupole Power Supplies

A:S1	1234567890123456789012345678901234567890
A:S2	1234567890123456789012345678901234567890
A:S3	1234567890123456789012345678901234567890
A:S4	1234567890123456789012345678901234567890
B:S3	1234567890123456789012345678901234567890
B:S2	1234567890123456789012345678901234567890
B:S1	1234567890123456789012345678901234567890

Horizontal Corr. Power Supplies

A:H1	1234567890123456789012345678901234567890
A:H2	1234567890123456789012345678901234567890
A:H3	1234567890123456789012345678901234567890
A:H4	1234567890123456789012345678901234567890
B:H4	1234567890123456789012345678901234567890
B:H3	1234567890123456789012345678901234567890
B:H2	1234567890123456789012345678901234567890
B:H1	1234567890123456789012345678901234567890

Vertical Corr. Power Supplies

A:V1	1234567890123456789012345678901234567890
A:V2	1234567890123456789012345678901234567890
A:V3	1234567890123456789012345678901234567890

Figure 2: The power-supply detail tab of the APSStatus application.

REFERENCES

[1] H. Shang et al., PAC 2003, 3470 (2003).

[2] R. Soliday et al., PAC 2003, 3473 (2003).

[3] R. Soliday, ICALEPS 2001, 545 (2001).

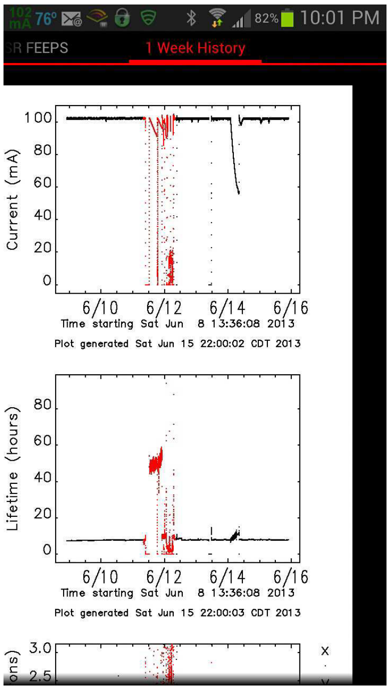

102 mA 76°
82% 10:01 PM
R FEEPS
1 Week History
Current (mA)
100
80
60
40
20
0
6/10
6/12
6/14
6/16
Time starting Sat Jun 8 13:36:08 2013
Plot generated Sat Jun 15 22:00:02 CDT 2013
Lifetime (hours)
80
60
40
20
0
6/10
6/12
6/14
6/16
Time starting Sat Jun 8 13:36:08 2013
Plot generated Sat Jun 15 22:00:03 CDT 2013
ons)
3.0
2.5
x

Figure 3: The one-week history tab of the APSStatus application. Scrolling vertically reveals additional plots, while pressing on a plot brings up a magnified view that can be zoomed and scrolled.

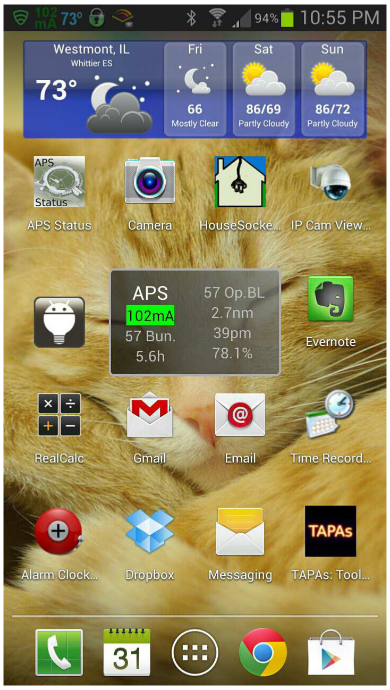


Figure 4: 2x1 APSstatus widget showing high-level APS information. Touching the widget brings up the full application. For clarity, we've removed most other widgets and apps from this screen. Note that the beam current also appears in the notification bar (top left).