Space Details

Key:	AndroidLogger
Name:	901112 - MBTS; Android data logger
Description:	
Creator (Creation Date):	oreilly (Feb 01, 2013)
Last Modifier (Mod. Date):	oreilly (Feb 01, 2013)

Available Pages

- Home
  - How to root an Android phone
  - Preparing the phone and configuring the drifter app
  - Procedure for drifter deployment
  - Test results
- Tasks

Home

This page last changed on Oct 22, 2013 by oreilly.

This is the home page for the 901112 - MBTS; Android data logger project.

The Chavez lab has developed a robust low cost "Stella" drifter that has been used repeatedly by our lab, CANON, Autonomy and several other MBARI projects. The drifter is based on a SPOT tracker and total hardware cost is about $250. We propose to exploit the rapid development of commercial low-cost mobile computing and sensor platforms by taking a prototype instrument data logger (developed by O'Reilly and Risi) that runs on an Android device (e.g. phone or tablet) and deploying it as a test on the low cost Stella drifter. The Android logger application known as "Drifter" was developed by O'Reilly; it periodically retrieves data from one or more RS-232 or analog instruments as well as onboard phone sensors (e.g. GPS, accelerometer, compass). The app then telemeters the retrieved data to shore via the cell phone network, WiFi or a satellite modem. Such a data logger could have several applications including low-cost moorings, drifters, or pier-mounted deployments.

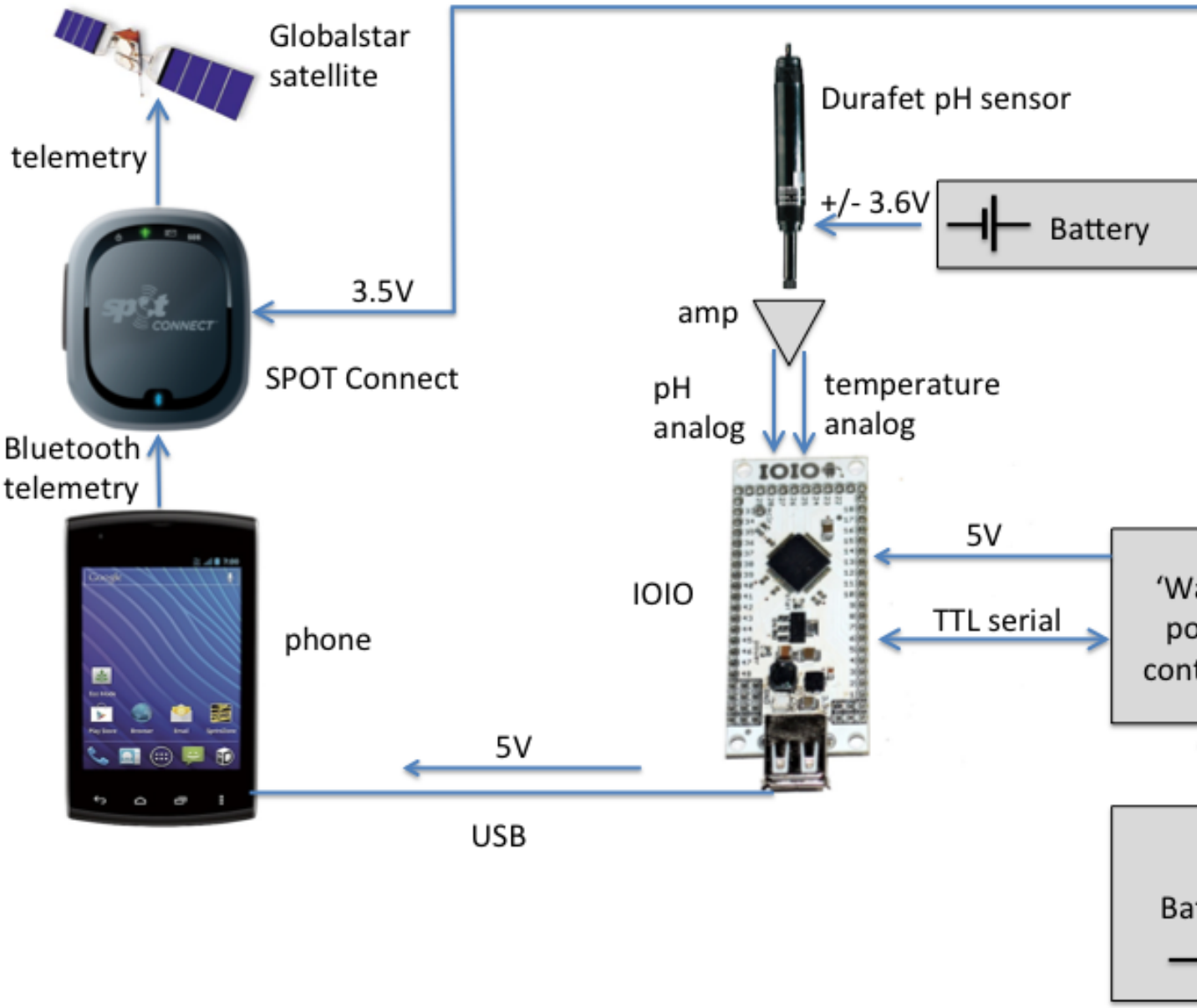
The Drifter app source code and associated files are attached to this page as a compressed '.tar' file. E.g. the code for Drifter version 1.61 is attached to this page as Drifter-1_61.tar.gz. Download the file into an Eclipse workspace, then decompress and un-tar it, e.g. on a Linux system:

```
% tar  zxf Drifter-1_61.tar.gz
```

Now run Eclipse - configured with the Android Development Tools plug-in - and import the files as an Eclipse project.

pH Drifter application

Below is a schematic depiction of components for the pH drifter application.



The Drifter app runs on the Android phone (currently a Kyocera Rise phone). The app is written in Java, and source code is currently archived at http://bitbucket.org as a "private" repository. We plan to distribute the code as open source in the near future.

The app retrieves pH and temperature sensor data through analog input ports on the IOIO board. The app manages the 'Wakey' power controller through an RS232 port on the IOIO board. The phone sends telemetry to shore as 41-byte messages through the SPOT Connect satellite modem, using the Bluetooth protocol described here.

The 'Wakey' power controller board developed by McGill provides power to the IOIO board, phone, and SPOT Connect satellite modem.

Android OS power management attempts to minimize power usage based on app activity. Note that phone can be externally reset only by physically pressing the phone's power button - but we do not have a way to do that with our current approach.

- Advantage: No physical modifications required to the phone.
- Disadvantage: There is no way to reset the phone once deployed.

Paul McGill measured Kyocera phone power usage in the following configuration: Airplane mode, Bluetooth enabled, screen off, Android data logger app sampling once every minute. Paul measured power usage in the following states:

- Data logger idle (between samples): 1.2 mA
- Data logger sampling: 60-70 mA. The sampling phase (acquire and log sample, send to SPOT Connect modem) lasts approximately 5 seconds

At the end of each sample cycle, Data logger puts Wakey to sleep, which disconnects power to the IOIO board; thus the phone does NOT get charged while the in the idle state.

Telemetry via SPOT Connect

A SPOT message can be no longer than 41 bytes (not including the transmit time and latitude/longitude header that the modem prepends). Thus the data logger app generates very terse summaries of the instrument, sensor and diagnostic for transmisson via SPOT. The app generates SPOT messages consisting of fields separated by a semicolon, with the following format:

```
instrument_record:orientation:gps:compass:phone_batt_level:wakey_batt_voltage
```

instrument_record - data record from attached serial or analog instrument. The format of the instrument_record depends on the instrument type.

orientation - data record from phones orientation sensors consisting of roll, pitch, and yaw

gps - data record from phone's GPS

compass - magnetic azimuth from phone's sensor

phone_batt_level - remaining phone battery, in percent

wakey_batt_voltage - "Wakey" battery voltage
Note that some fields may be zero-length. E.g. if the user does not select orientation sensor to be logged, then the SPOT message orientation sensor field (second field) will be zero length. E.g. following is an example SPOT message:

```
tmp 2923,pH 1352:::264.0:98:11.9
```

In the above case, the app had been configured to log data from an attached analog pH sensor (which outputs temperature and pH), and to log phone compass data (no other sensors were selected).

Software tasks

How to root an Android phone

This page last changed on Apr 09, 2013 by oreilly.

The Drifter app can optionally reboot the phone at a specified interval to clear potential error states in the phone and its interfaces. This reboot capability requires the app to have "root" privileges on the phone. I used the following procedure to root a Kyocera Rise phone. This is not a very simple process, and several of the messages displayed during the process are somewhat misleading - but this worked for me. This procedure exploits a kernel bug in some Android phones, but may not work for your particular phone. Caveat emptor.

1. make sure usb debuging is on (under developer options)

2. make sure unknown sources is checked (in system settings under security)

3. install poot.apk

4. install minstro 2 from Google Play Store

5. install superuser from Google Play Store

6. run poot on your phone - it will ask if it's okay to install libraries - say 'yes'

7. Select 'Press here to run poot'. The first time I did this I got the following result:

```
2012 giantpune
[+] opened device
[+] Set loggin mode
[+] Resolved symbols
[+] Mapped 0x10000000
[-] expected current to be 1:ffff
[-] patching failed.
A demon materialized while pooting. Error code: 22
su binary was not written
```

I then rebooted my phone and repeated this step; now I got a different result that indicates the 'su' binary was installed (despite the "Error code: 64" message):

```
2012 giantpune
[+] opened device
[+] Set logging mode
[+] Resolved symbols
[+] Mapped 0x10000000
[+] Hooked mask_request_validate()
[+] Ran some code as the kernel
[+] I think root, therefore I am root
WriteSUBinary()
[+] remounted /system read-write
[+] wrote su binary
[+] changed su binary to uid 0, gid 0, mode 6755
[+] remounted /system read-only
[+] Removed hook
[-] current && current != last + 1: 0000ffff
0000ffff
[-] Haxx has run its coarse. Reboot the device to play again

A demon materialized while pooting. Error code: 64
You need to restart your device
```

8. Run the 'superuser' app; go left to 'info' then down to su binary; click it and then click update (only do it once or it won't work)

9. After the update screen finishes, click 'temp unroot' under the su binary, then reboot your device

10. Install root checker from Google Play Store and run it to ensure the phone is now rooted.

11. If the phone is rooted, go to superuser, uncheck 'temp root' and reboot

This procedure may work for your phone - if not, search the web for root procedures appropriate to your phone.

Preparing the phone and configuring the drifter app

This page last changed on Sep 11, 2013 by oreilly.

Preparing a new phone

Associate the phone with a Google user account mbariengineer@gmail.com - see Tom O'Reilly for the password.

If you plan to use the IOIO board, download the IOIO Hardware Tester app from Google Play; this app is very useful for trouble-shooting, allows you to verify that the IOIO board is connected to your phone and functional.

If you plan to use the Drifter app's automatic reboot capability, you must first ""root" the phone.

If you plan to use a SPOT Connect modem with the app, **you must "pair" the phone with the specific SPOT modem it will connect to**. If you must replace the SPOT for any reason, then you must pair the phone with the new SPOT. See the "Pairing" section of the SPOT Connect user manual.

Eventually we'll put the Drifter app on a server such as Google Play to allow easy over-the-air installation. Until then you should bring the phone to developer O'Reilly, who has the required development tools to install the app on a phone. Before bringing the phone to O'Reilly, you can (1) Enable app installation from non-Google sources: System->Security->Unknown Sources and (2) Enable USB debugging: System->Developer Options->USB debugging

Configuration and power

The Android 'Drifter' app can be configured to log various sensors and use various telemetry channels. Many if not most deployment scenarios provide a limited amount of power to run the phone - e.g. external batteries. Unless you are lucky enough to have the phone plugged into an AC adapter, consider the following power-saving tips.

Consider powering off the following, depending on your deployment needs:

- WiFi - if you are not using WiFi for telemetry, disable
- Cell radio - if you are not using cell network for telemetry via email or SMS, put phone into 'airplane mode'
- GPS - turn off if you don't need to log GPS (SPOT Connect provides GPS)
- BlueTooth - turn off unless you are using SPOT Connect for telemetry, or acquiring data from Bluetooth-serial instruments

Consider disabling the following to avoid excess power use:

- Disable Google Play notification and diable auto-update - I would expect the phone wouldn't try in airplane mode, but just in case:
  Open Google Play: Press menu, select Settings
- Disable auto-sync - I would expect the phone wouldn't try if in "airplane" mode, but just in case:
  System settings -> Accounts & sync ; slide switch at top of screen to "off"
- Disable mobile data:
  System settings -> Data usage; turn mobile data switch "off"

Also, you should set the screen timeout to the lowest defined value, since the phone will be unattended most of the time (the phone screen may occasionally turn on, e.g. when charge flows though the USB port).

The Drifter app can use the MBARI "Wakey" board to manage power to attached instruments and modems. If you are using Wakey, then select Configuration->Power Management -> Use Wakey Controller.

Procedure for drifter deployment

This page last changed on Apr 17, 2013 by oreilly.

Connect phone to IOIO board with USB

Plug cable from battery pack to Wakey; verify that SPOT and IOIO power LEDs are now lit

- NOTE: to power-cycle Wakey, **unplug the power cable for at least 30 seconds** before re-plugging

Run "IOIO Hardware Tester" app, verify comms with IOIO board

If no comms with IOIO:

- Verify IOIO board has power from Wakey
- Reboot phone

Turn off unneeded phone radios:

- Turn off GPS
- Put phone in 'Airplane mode'
- Enable Bluetooth

Run Drifter app

Run Diagnostics -> Wakey test (puts Wakey to sleep)

Run Diagnostics -> SPOT test

Run Manual Refresh

Before stowing phone in housing, consider dimming phone screen to save power (the screen will light on each sample cycle while IOIO has power)

Test results

This page last changed on Jun 13, 2013 by oreilly.

May 11 - May 25 (15 days)

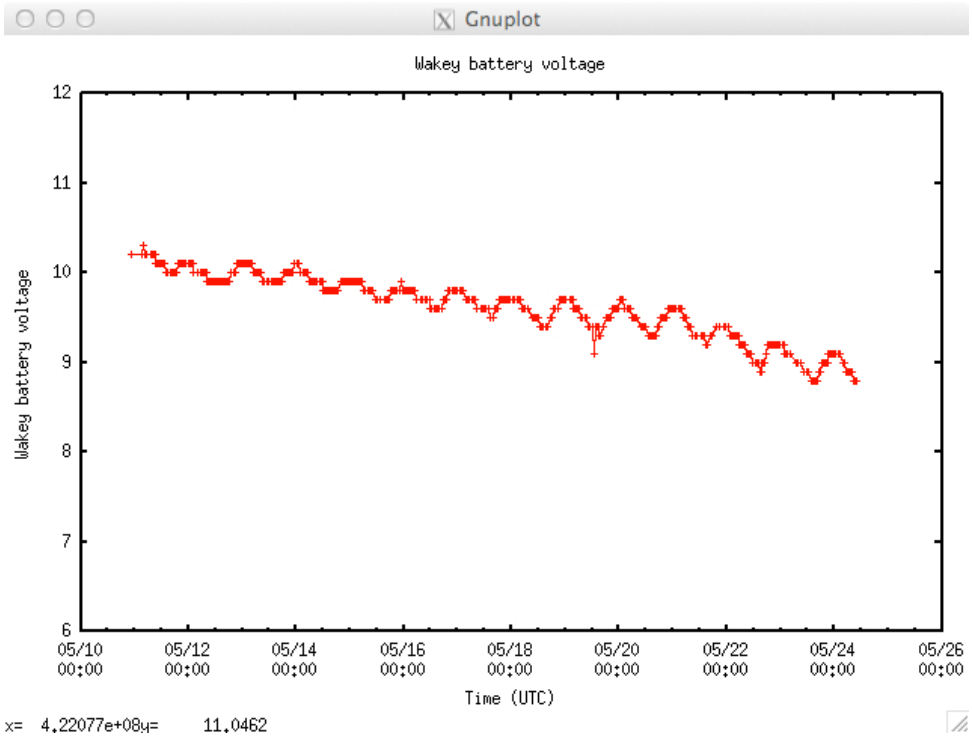
The system is located on the Building B rear deck, for a clear view of the sky. Most of the gear (phone, IOIO, Wakey, batteries) is inside a Pelican case, with the SPOT modem duct-taped to the outside of the case. The case was latched on one side only to prevent cutting the SPOT power and data wires. The pH sensor is currently NOT attached.

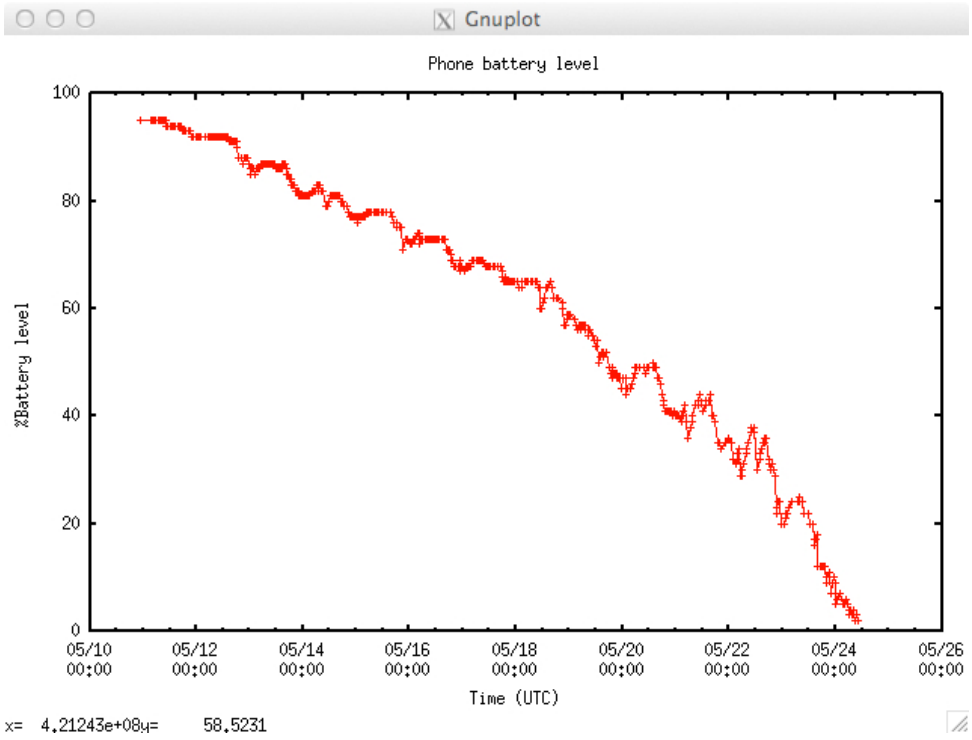
I haven't yet looked at the detailed logs, but see no indication of major errors during this burn-in test so far.

Attached are plots showing Wakey battery voltage and phone battery level.

For this test, the phone is configured to send SPOT messages as well as diagnostic messages via wifi. At sea we will of course not incur the wifi power usage.

Got 495 SPOT messages, out of 725 total samples; SPOT success rate = 68%



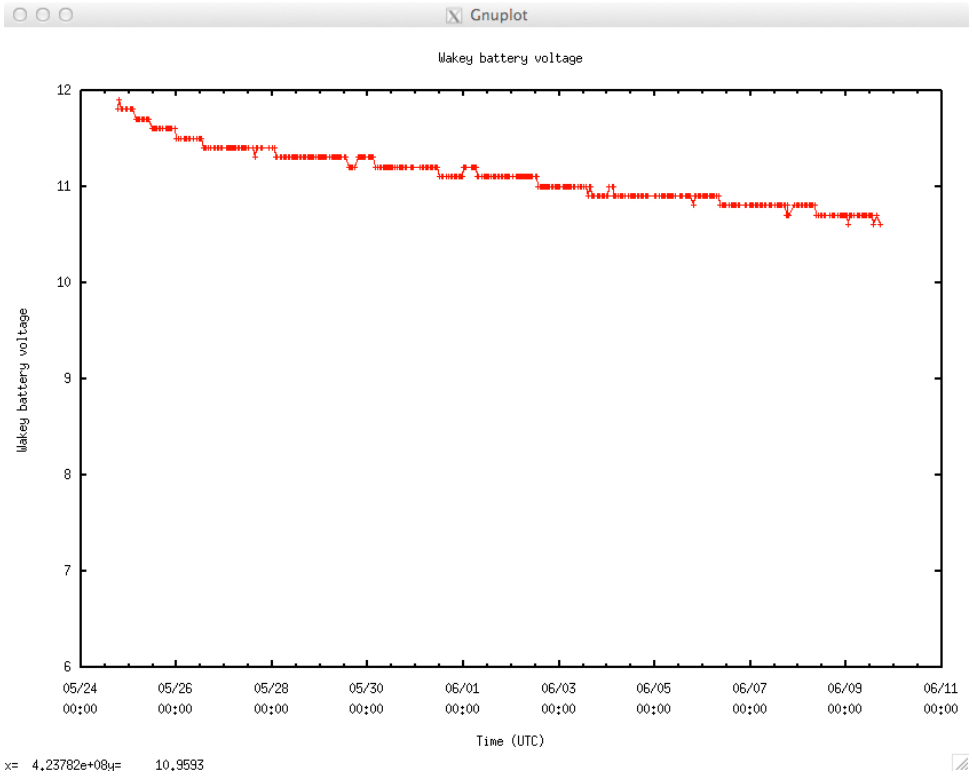


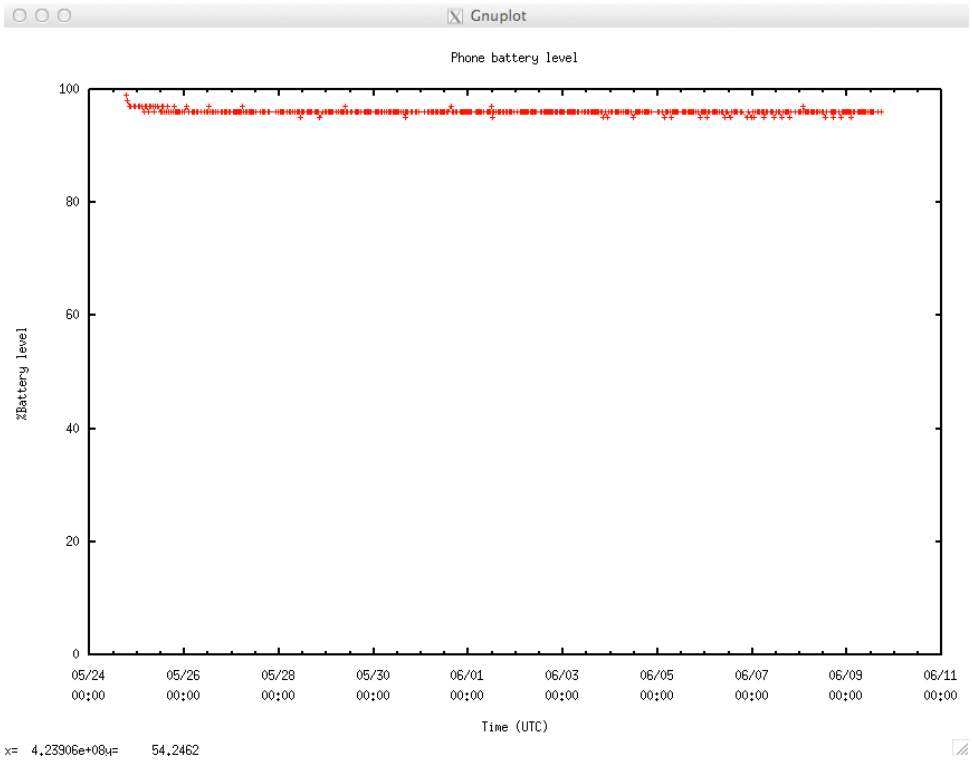
May 24 - June 9 (16 days)

This test was also performed on Building B rear deck, physically configured identically to the May 11 test. But in this case, WiFi messaging was DISABLED - compare the following plots to see much lower power usage compared to the May 11 test. Note that the charge provided to the phone by the D-cell battery pack every thirty minutes was sufficient to keep the phone battery level at about 96 percent for the duration of the test.

Got 495 SPOT messages, out of 852 total samples; SPOT success rate = 58%

The last SPOT message was received on June 10. Inspection of the hardware revealed a broken corroded wire leading to the SPOT, as well as corrosion on a Wakey connector. Note that no effort had been made to waterproof the SPOT housing or the partly-latched Pelican case that housed the Wakey board. This corrosion may explain the overall lower SPOT success rate, compared to the May 11 test (success rate = 68%).


Gnuplot
Wakey battery voltage
12
11
10
9
8
7
6
Wakey battery voltage
05/24 00:00
05/26 00:00
05/28 00:00
05/30 00:00
06/01 00:00
06/03 00:00
06/05 00:00
06/07 00:00
06/09 00:00
06/11 00:00
Time (UTC)
x= 4.23782e+08 y= 10.9593


Gnuplot
Phone battery level
100
80
60
40
20
0
%Battery level
05/24 00:00
05/26 00:00
05/28 00:00
05/30 00:00
06/01 00:00
06/03 00:00
06/05 00:00
06/07 00:00
06/09 00:00
06/11 00:00
Time (UTC)
x= 4.23906e+08 y= 54.2462

Tasks

This page last changed on Aug 29, 2013 by oreilly.

Outstanding tasks
15. If instrument bluetooth socket can't be created, get NullPointerException

16. If instrument serial port can't be created, get NullPointerException

17. If configuration requires IOIO but it is not physically connected, 'Refresh' button never gets reenabled.

18. Need to ensure that Wakey is awake for first sample, to properly synchronize app sampling schedule and Wakey sleep schedule.

23. IOIO seems to get some current from phone; red IOIO LED glows faint red when connected only to phone.

24. Phone power sometimes runs out in non-test conditions - not clear why. WiFi?

26. Configure option to start server at boot time, not just on auto-reboot. ESPECIALLY IMPORTANT IN CASE OF SPONTANEOUS REBOOT, WHICH HAS BEEN OBSERVED

27. check the relative drift of the Android phone and Wakey clocks before we deploy.

28. DrifterService should assert 20-minute sleep onset delay for Wakey comm port, since user may inadvertently leave some test value instead.

29. If the coms delay time is greater than the sleep time, causing the coms power to be already on when Wakey awakes, then the action of electronically pushing the SPOT power button to turn SPOT on may instead turn it off. To prevent this, the Wakey code should check to see if the coms power is already on before pushing the SPOT power button.

31. Wakey RX, TX pins default to 0. Should default to 6, 7

Done

1. GUI to stop service

5. Fix rx/tx pin property bug (DrifterService reads property as string, should be int)
Appears to work if rx/tx pin properties are stored as strings

9. Do not try to send message to SPOT while SPOT is in transmit state

12. Check into version control
6. If SPOT selected, check that Bluetooth is enabled before starting run

10. Warn user of unneeded power use (e.g. Wifi, cell radio) when starting a run
11. Enable log-only run, i.e. no telemetry specified

8. Implement Wakey control, Wakey test

7. If GPS selected, check that GPS enabled before starting run

14. Log debug/info/error messages to a file.. File would be accessed after recovery.

13. If ioio connection fails, no data is telemetered (e.g. not even phone sensor which doesn't require ioio)

4. Log engineering data. Telemeter engineering data; phone battery state, Wakey battery state, etc

19. Implement compass acquisition

20. Design/implement succinct SPOT message format. Each message can probably include both science and engineering data

21. Sample schedule does not run when no telemetry channel selected (bug)

22. App and wiring on amplifier board do not agree on analog pin number. Compared app analog readout to IOIODiagnostic app (from market); IOIODiagnostic shows expected voltages on pins, so datalogger must be improperly reading analog channels.

25. Consider occasional programmatic reboot of phone to improve reliability - this requires 'rooting' the phone.

2. Data log format

3. Log science data