

ezTelemetry for Android devices



droidTelemetryFPV

Overview & Operating Instructions

Preliminary/Beta. Jan 2012

The screenshot displays the ezTelemetry app interface on an Android device. The top status bar shows various system icons and the time 0:23. The app's main menu includes 'Telemetry', 'Map', and 'Settings'. The 'Telemetry' view is active, showing a list of data points:

GPS	Plane	Launch	P
Longitude			0
Latitude			0
Altitude ASL	1345.80m		
Altitude AGL	12.00m		
Distance	1.04km		
Heading	-64°		
Sat. Count	9		
Battery			
Voltage	10.66V		
Current	19.12A		
mAh	2258 mAh		
UHF Link			
RSSI 1	68dBm		
RSSI 2	65dBm		
Link Qty.	100%		
Syncs Lost	1		
Freq Error	0		
Telemetry			
		Good	1
		Bad	6
Summary			
		Total Dist.	
		Max. Dist.	
		Max. Alt.	

The 'Map' view is also visible, showing a satellite map with a white airplane icon indicating the current position. The bottom of the screen features buttons for 'Plane', 'Launch', and 'Pilot'.

Overview

In mid-2008, the ImmersionRC team released the first telemetry decoder application for the Apple iOS platform (iPhone, iPad). This application decoded directly the audio-based telemetry from EzOSD and TinyTelemetry devices, and presented the data as a moving map, or raw telemetry data.

This application, still unique, is now joined by an Android equivalent, droidTelemetryFPV.

The iOS version of the application has been used by hundreds of modelers to recover lost FPV planes, and also to help tune the performance of FPV components.

The Android version will greatly increase the number of platforms supported, and will include many Android smart-phones, and tablets.

Pre-Requisites

- Android-based device, running Android 2.2 or later (Froyo)
- Audio Interface cable (*see later chapter for more details*)

Features

- Real-time moving map display, shows plane position
- Phone/Tablet position also shown (for GPS equipped models)
- Live, and Search modes, to aid in locating a lost plane
- Full telemetry display, uplink status, GPS position, battery status

Compatibility Disclaimer

There are a huge number of different Android devices out there, none of which truly follow a 'reference design'. This means that compatibility on all devices is not guaranteed.

The minimum requirements are a microphone port (generally one of the 4 pins on the 3.5mm headphone jack), and Android 2.2 or later.

Certain devices, such as the Kindle Fire™ do not contain a GPS, nor the ability to run Google Maps (for now). droidTelemetryFPV will not currently run on these devices, but it is foreseeable that it will run in some reduced capacity in the future.

It is also foreseeable that 'rooted' devices such as the Fire, will be able to run the application, using whatever hardware is present in the device.

Compatibility List

Note that this list contains only devices which beta testers have tried. There are thousands of android devices on the market, and it is expected that many of them will be unsuitable for use for telemetry decode applications.

Manufacturer	Model	Compatibility	Cable Compatibility
Samsung	Galaxy S-Plus	OK	iPhone Telemetry Cable
Samsung	Galaxy S II	OK	iPhone Telemetry Cable
Kindle	Fire	Not compatible (can't install apps which require microphone)	-
Asus	eee Tab Transformer TF101 droid 3.2.1	OK ⁽¹⁾	iPhone Telemetry Cable <i>needs to be inserted before app. Is run</i>
HTC	Thunderbolt	OK	
Acer	Iconia Tab A100	None ⁽²⁾	N/A
Motorola	Droid Bionic Droid 2.3.4	OK	TBD
Samsung	Galaxy Tab 8.9	OK	iPhone Telemetry Cable

Notes:

1 – Appears to have a firmware bug which makes it difficult to guarantee that the external mic. input is always available. The shutdown/restart trick explained in the troubleshooting section appears to work around this.

2 - Ext. Microphone freq. response useless for telemetry (cutoff at ~5kHz). Possible that this is a software filter, and will be fixed in future releases.

Compatibility List: Cheap 'Ebay' tablets... buyer beware

A quick search on ebay will find a whole slew of cheap Android tablets. The ImmersionRC team will be very interested to hear from FPVers who have tried droidTelemetry on some of these tablets.

Some points to watch out for before purchasing:

- Ensure that the device has an external microphone input, many devices do not
- Ensure that the device has a GPS, if required of course. An in-build GPS makes the device infinitely more useful when locating a downed model.
- Watch out for sub-optimal resistive touchscreens. The iPad, and all of the high-end Android tablets use capacitive touchscreens, which are far superior, and support multi-touch.

Installation

Installation during the beta phase will not use the Android Market. Instead, enter the following address into the web browser:

<http://www.immersionrc.com/android.htm>

Configuration

The iTelemetry settings are accessed in Settings tab.

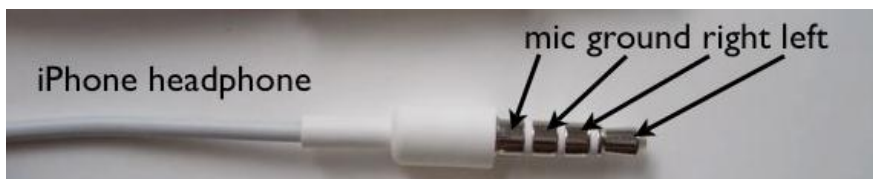
The units to be displayed for Altitude, Distance, and Speed are each independently configurable.

The telemetry rate is also configured here for the current version of the application. *Note that a future beta will remove this setting, and will auto-detect the telemetry rate.*

Connection

The connection to the android device is via the microphone/headphone jack. This varies wildly between android devices, but for some common devices, the pinouts below can be used.

3.5mm 4-pole, iPhone compatible



OMTP Standard

Many Chinese Android smartphones follow the 'OMTP' standard.

The pinout for the headphone/mic connector on these devices is as follows: (Mic and Gnd pins are swapped relative to the iPhone standard)

Pinout from tip: Left/Right/Mic/Ground

More details on this standard may be found here:

http://www.omtp.org/OMTP_Local_Connectivity_Wired_Analogue_Audio_v1_0.pdf

Identifying Pinout from a Mic-equipped headset

If the pinout of your Android device is unknown, there is a relatively simple way to determine it, assuming that a mic-equipped headset is available for the device.

A DVM (Digital Voltmeter) is required.

1. Place the DVM in ohms mode
2. Move the DVM probes between each pair of pins, and note the resistance measured
3. Between Ground, and each speaker (earpiece) output, the resistance should be approx.. 30 ohms.
4. Between Ground, and the Mic. Input, there should be a resistance of approx.. 2kOhms (between 1k and 2.5k)

Once the ground, and microphone pins have been found, follow the instructions below to make an interface cable.

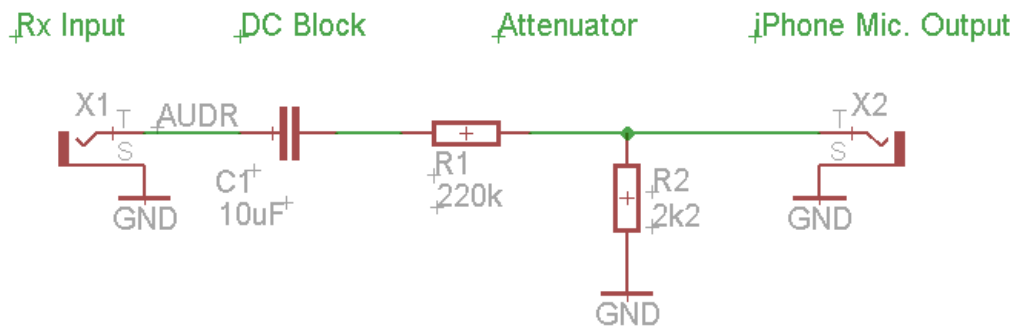
Commercial iTelemetry Interface Cable





A commercial iTelemetry interface cable for devices with an iPhone/iPad compatible pinout is available from www.nghobbies.com, and a few of the other FPV retailers.

This cable is well designed, with the few components required hidden in the connector. It is also equipped with a trimmer to adapt to receivers with non-standard output levels.

DIY iTelemetry Interface Cable

In order to provide the necessary level shifting, a small circuit consisting of three easy to obtain components is required. For FPVers in the USA, these may be obtained from any RadioShack store. *(Note, once the application is out of the beta test phase, a cable will be available from FPV stores which will contain this circuit)*



3.5mm Jack, 4-pole (plastic case, NOT metal)		Farnell: 1280728
220k 1/8W resistor		Farnell: 9339329
2.2k 1/8W resistor		Farnell: 9339302
10uF capacitor		Farnell: 9451056

Note that for certain 'Lawmate-style' A/V receivers, which have non-standard audio output levels, greater attenuation is required.

Switch the 220k in the schematic for a 470k in order to achieve this.

Troubleshooting: General

Getting the external microphone to work reliably on some (ok, most) Android devices has proven to be a challenge. It appears to have a mind of its own.

There are no SDK functions which let us force the system to use the external Mic, and it is not automatically used when the 3.5mm jack is plugged in, even if an 'official' mic/headset is used, from the manufacturer who supplied the device (you know.. the one that ships with the device!).

The following procedure appears to work most of the time, on some problematic devices:

- 1) Turn the device off, completely off
- 2) Unplug everything from the device, especially the 3.5mm jack
- 3) Turn the device on, and enter the unlock code (if required)
- 4) Plug in the device side of the 3.5mm cable, but leave the other end unconnected
- 5) Run the droidTelemetry application
- 6) Connect the audio source to the other end of the cable.

NOTE: If anyone who is intimately experienced with Android devices can explain more about why this is necessary, please speak up!, we'd love to hear from you.

Other debug tips include the following:

- 1) Restart the android device, as suggested above, and instead run the standard Android 'Sound Recorder'. Hit the record button with the 3.5mm cable connected, and tap the device. If the recorder's VU meter jumps, then the internal microphone is selected.
- 2) Run the droidTelemetry application, and switch to the diagnostics tab. Whistle, and see if the trace shows a sinusoid. If it does, again, the internal microphone is selected.
- 3) Use a 'task killer' application to ensure that no applications are running before starting droidTelemetry. If any application has 'grabbed' the audio input port, then no other application can access it (Android developers.. please fix this, make it work just like iOS ☺)

Troubleshooting: Android Microphone Frequency Response

Whilst testing the application on various devices, one in particular gave us some nasty problems in the 'audio frequency response' domain.

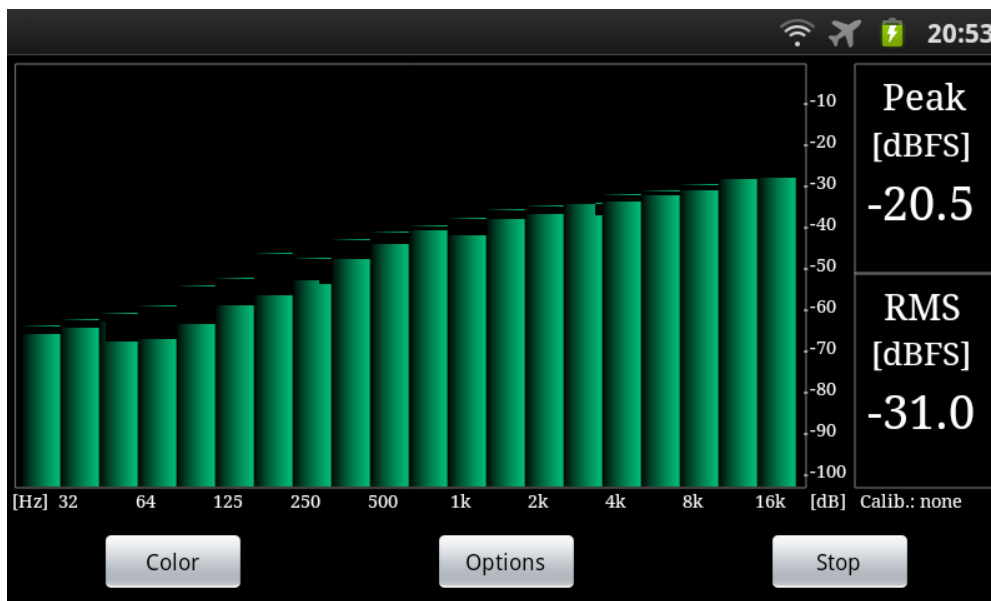
To explain why, a couple of pictures should help.

First picture, a spectral plot of the mic. input of a Samsung S Plus (i9001) Smartphone, plotted with a tool called 'RTA Analyzer' (free on the Android Market).

The source was 'pink-noise' injected from a PC-based signal source.

Bars to the right indicate high frequencies, bars to the left indicate low frequencies.

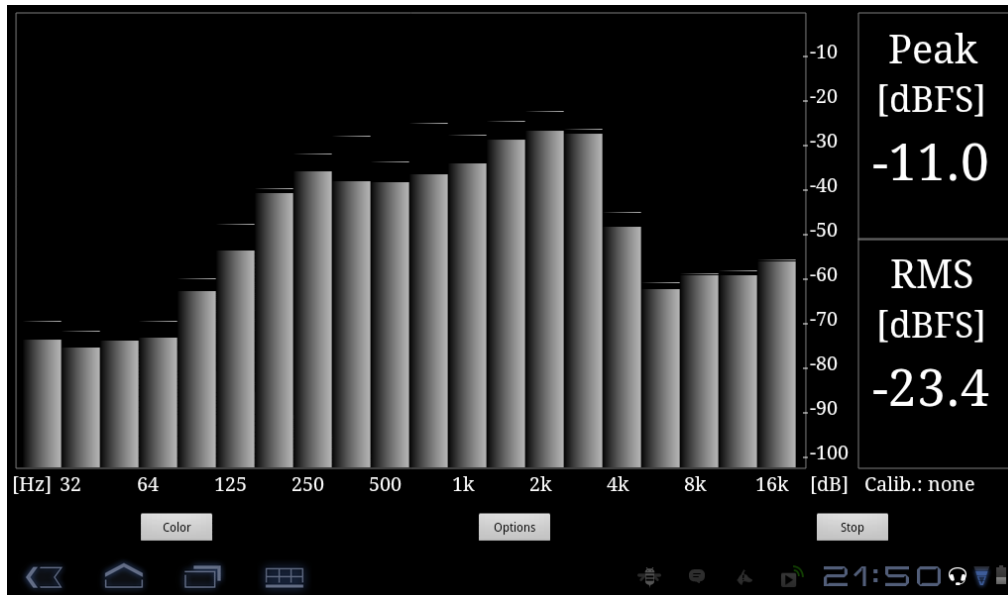
Note that from 500Hz up to over 16kHz, the plot is relatively flat (yes, it climbs a bit, but this is ok)



Now contrast this with the same test, but performed on the Acer Iconia Tab A100. Cute little 7" tablet, nice and cheap, but useless for telemetry decode.

Note that above about 4kHz, the response just disappears off the edge of a cliff!. For the technical folks, it drops over 20dB almost immediately.

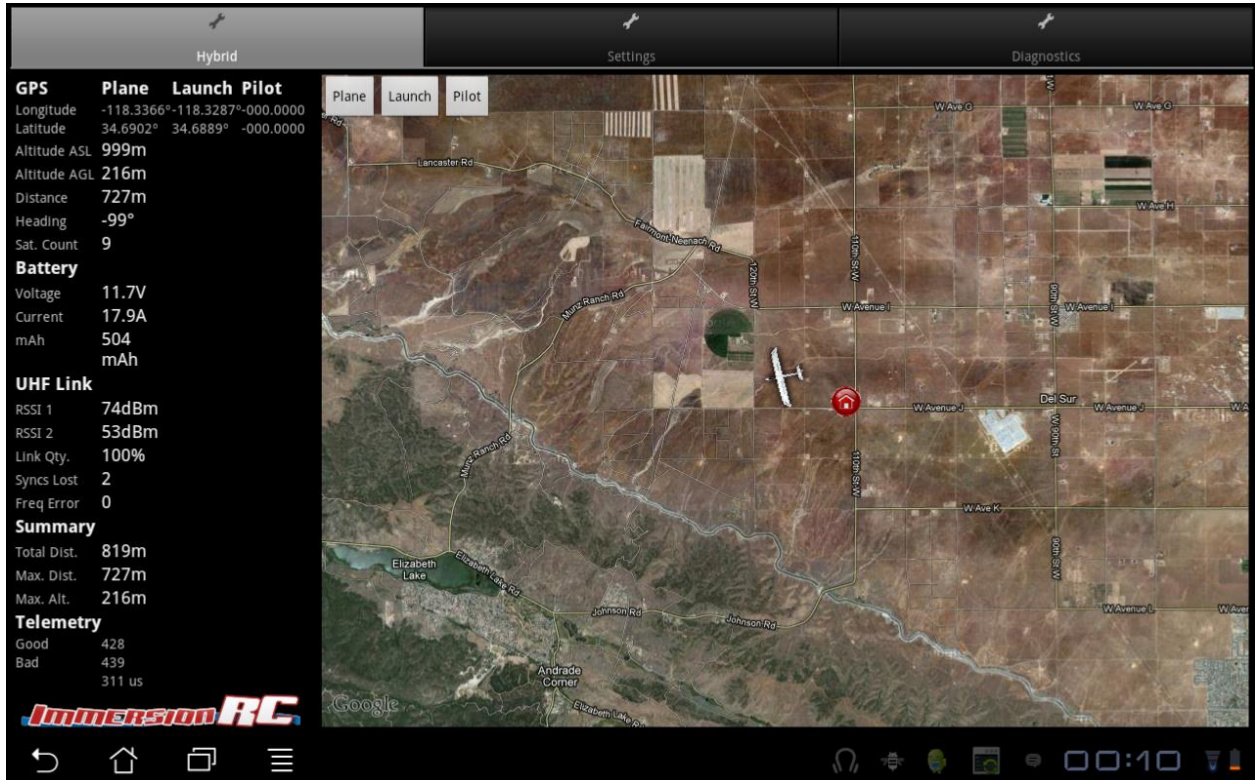
This looks like a DSP filter, applied in software, so it is entirely possible that a future firmware upgrade will fix this, and make this device useful for our application(of course, no firmware upgrade is going to fix its mediocre display, and poor battery life, but good audio would be a good start...)



BUYER'S TIP: If you are looking to purchase an Android phone or tablet, and if you don't mind looking like a plank in the local consumer electronics store, try the following: See if you can get the 'RTA Analyzer' installed on the device in the store, either by doing it yourself, or asking a helpful member of staff (some stores have the devices on WiFi for demo purposes). Once you have done this, get nice and close to the device, and make a hissing sound like a TV after the last program ends late at night. If the display looks like the green plot above, then you should be ok, get out your credit-card. If the display looks like the gray plot above, move to the next device in the lineup, and try again (Just don't ask us how we know that this works... ☺)

Eye-Candy

Hybrid view on a 10.1" Asus eeePad Transformer.



Map view on a Samsung i9001 S Plus Smartphone

