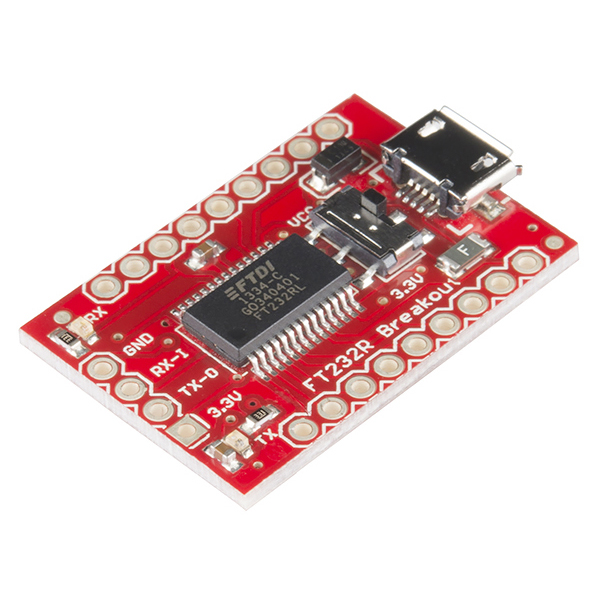
# Background:

This test application developed by the Intel Open Source Technology Center (<https://01.org>) was designed to help measure and compare audio latency parameters across multiple Android platforms. The concepts were driven by Android compatibility requirements but can be used in other frameworks. We are currently exploring the extension of the solution to low-level ALSA driver tests as well as higher-level tests with applications running in a Chrome browser.

# Overview:

The Android definitions and requirements are listed in [1], and the Android audio team provides detailed information in [2]. To measure continuous and cold input latency, there is a need to capture both a trigger signal and the analog input/output audio data. The latency can then be measured with e.g. a two-channel oscilloscope.

The main idea behind this application developed by Intel was to avoid any hardware modifications or having to recompile kernel drivers. Unlike previous solutions based on GPIOS or light sensors, the latency can be measured by connecting an external USB device based on the popular FTDI FT232RL chip [3], see picture below. This solution assumes that the USB host capability is supported, and the connection between the device to be tested and the FTDI board relies on a microUSB cable and an USB-OTG converter.

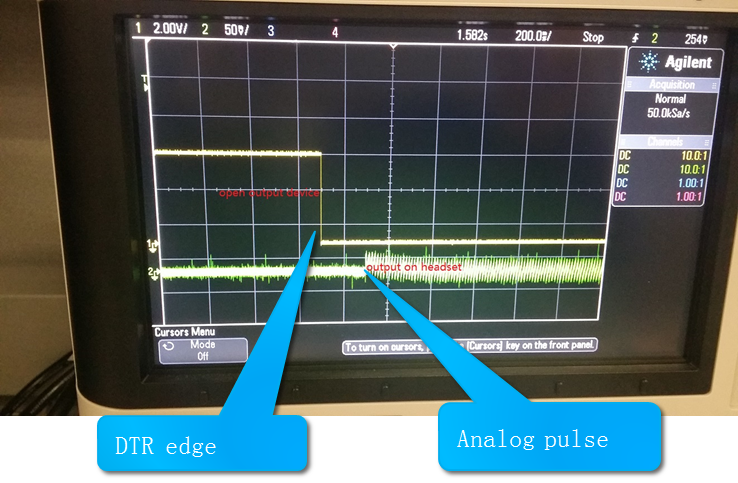


Once the FTDI device is connected, the application will ask the user for permission to access the hardware, configure it and toggle the DTR GPIO by sending a command to the USB stack (no kernel or AOSP changes are needed). The granularity of the USB protocol (125 usec) and software scheduling limit the precision of the measurements to about 1ms, which is more than acceptable for cold and continuous latency.

All test cases assume that the device is initially idle and touch-tones are disabled. The buffer sizes and sampling frequency used by the application are queried on startup to avoid additional delays due to sample-rate conversions or memory copies.

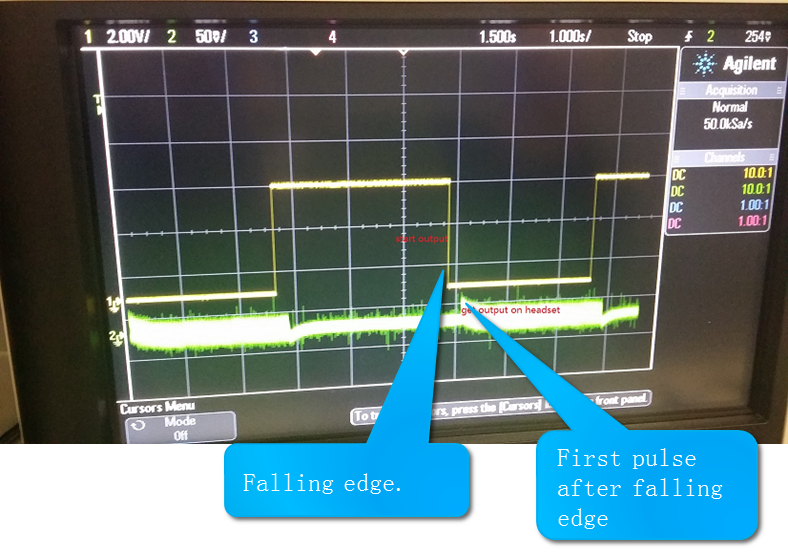
# Cold output latency setup:

* The application toggles the DTR signal
* The application opens the audio output device and starts transmitting a pulse signal
* The difference between the DTR edge and the analog pulse is the cold output latency.



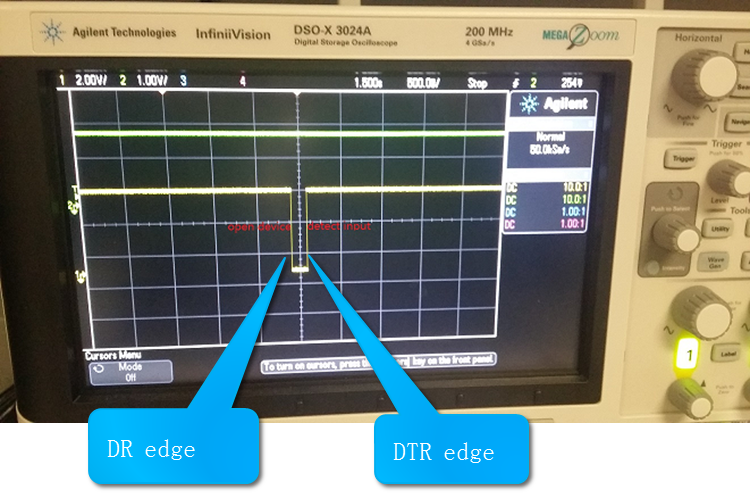
# Continuous output latency setup:

* The application opens the output device and starts streaming audio.
* When the stack requests a new buffer, the application toggles the DTR signal and alternates between pulses and silent period. The difference between DTR falling edges and the first pulse after falling edge is the continuous output latency



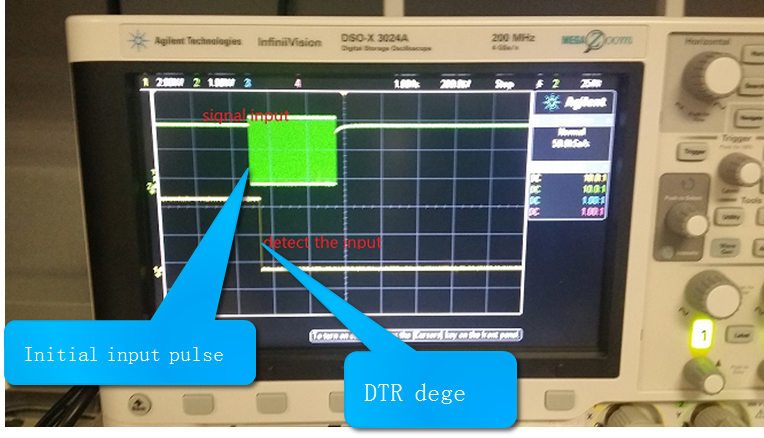
# Cold input latency setup:

* The application toggles the DTR signal
* The application opens the input device and starts recording
* When the application receives the first buffer it toggles the DTR signal again
* The cold input latency is the difference between the two DTR signal edges



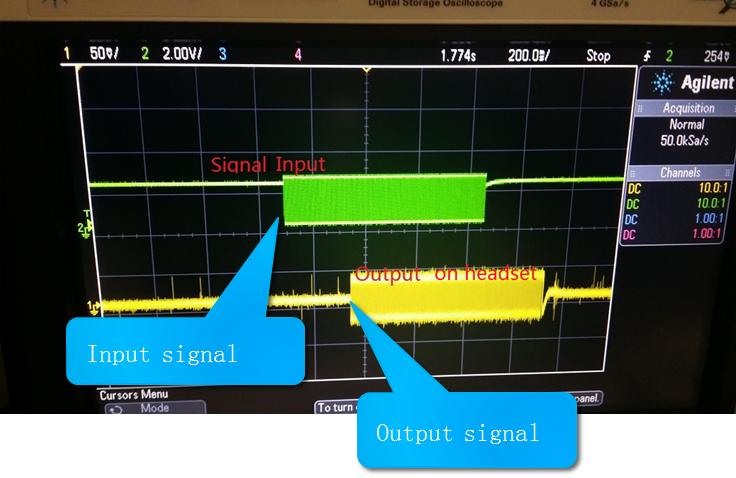
# Continuous input latency setup:

* The application opens the input device and starts recording
* An externally-generated pulse is injected on the analog input
* For each received buffer, the application takes a timestamp and checks the presence of a pulse.
* If a pulse is detected, the application will sleep for a time dependent on the pulse position in the buffer before toggling the DTR signal. This helps avoid measurement issues resulting from the lack of alignment of the buffer boundaries and the initial pulse.



# Round-trip latency setup:

* This measurement is slightly different from the one mentioned in [2].
* The application opens the output and input devices.
* Application starts recording and playback. Each buffer is copied from input to output.
* Inject a square signal from OSC on the headset input and then check on the headset output from OSC
* The delta between signal input and headset output represent the round-trip



# Additional notes:

* In a number of cases, the continuous latency is higher than the size of the ALSA ring buffer. This is mainly due to additional buffering in either the SOC DSP or the audio codec.

# References:

[1] <http://source.android.com/compatibility/android-cdd.pdf>

[2] <https://source.android.com/devices/audio/latency.html>

[3] <https://www.sparkfun.com/products/12731>