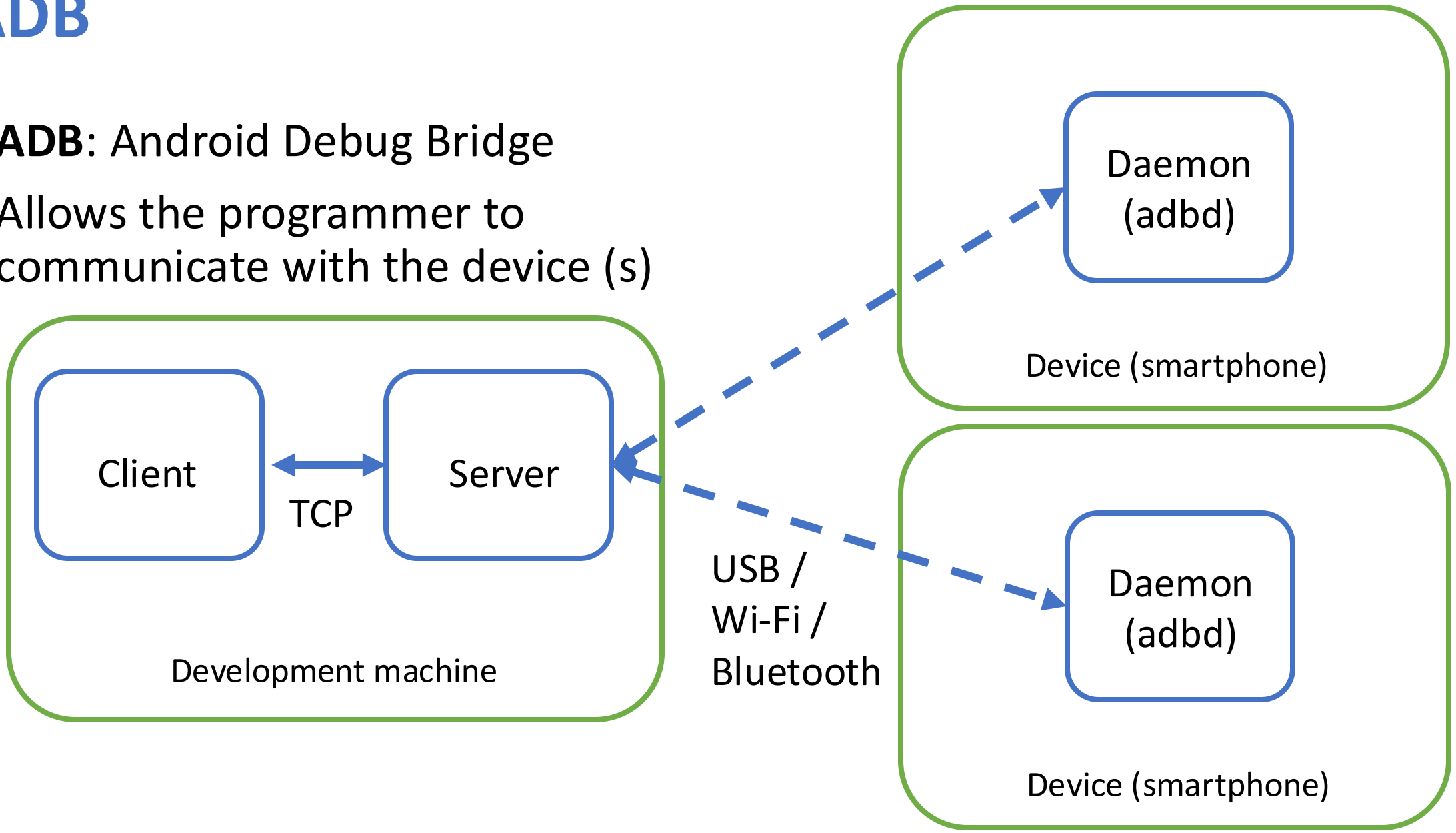
Android:
Command Line and Tools

Using the command line

- Some operations can be more comfortably carried out using the command line, in particular if automated repetitions are needed
- Android SDK provides many commands available through CLI
- Most relevant ones
  - adb
  - dumpsys/bugreport

ADB

- **ADB**: Android Debug Bridge
- Allows the programmer to communicate with the device (s)



ADB

- adb binary is in the ***`android_sdk`*`/platform-tools/`** folder
- Many functonalities:
  - install apps
  - copy file from/to device
  - Unix-like shell
  - interact with activities and package manager
  - take screenshots and videos

ADB

- List available devices and emulators:

```
→  platform-tools ./adb devices -l
List of devices attached
12161FDD40004E         device usb:1310720X product:redfin model:Pixel_5 device:redfin transport_id:3
emulator-5554          device product:sdk_gphone_arm64 model:Android_SDK_built_for_arm64 device:generic_arm64 transport_id:4
```

- **`device`** = connected
- **`offline`** = device is not connected
- **`no device`** = no device connected
- **`-l`** provides a description of devices
- serial number is useful to identify a specific device and send commands to it

ADB

- Sometimes restarting the server is useful: **`adb kill-server`**
- To send commands to a specific device:

`adb -s serialnumber command`

```
→  platform-tools ./adb -s 12161FDD40004E  shell ls
acct
apex
bin
bugreports
```

- To install an app:

`adb install file.apk`

ADB

- To push and pull files (or folders) to/from the device:

```
adb push example.txt /sdcard/example.txt
```
(local to remote)

```
adb pull /sdcard/example.txt example.txt
```
(remote to local)

- To interact with the activity manager (some examples):

  Start an *Activity* with an *Intent*:

```
adb shell am start -a "android.intent.action.VIEW" -d "https://www.unipi.it"
```

  `-a` the intent's action

  `-d` the intent's data uri

  same but specifying component name:

```
adb shell am start -n "it.unipi.dii.msssbleexample/.MainActivity"
```

Dumpsys

- **dumpsys** is a monitoring tool executed on Android devices
- Provides information about system services and resource usage of apps
- Syntax

```
adb shell dumpsys [-t timeout] [--help | -l | --skip services | service
[arguments] | -c | -h]
```

`-t timeout`	Specifies the timeout period, default value is 10 seconds
`-l`	List of system services that can provide information
`-c`	Output as csv, machine-readable

dumpsys

- The list of services is long

```
➜  platform-tools ./adb shell dumpsys -l
Currently running services:
  DockObserver
  SurfaceFlinger
  accessibility
  account
  activity
  activity_task
  adb
  alarm
  ...
```

dumpsys

- Can be used for analyzing the energy required by an app
- Reset battery statistics:

```
➜  platform-tools ./adb shell dumpsys batterystats --reset
Battery stats reset.
```

- Collect data:

```
./adb shell dumpsys batterystats > output_file.txt
```

dumpsys

- The output provided by batterystats is detailed and not well-documented

Battery History (1% used, 66KB used of 4096KB, 209 strings using 22KB):
0 (14) RESET:TIME: 2022-05-09-12-21-51
0 (2) 100 status=not-charging health=good plug=none temp=282 volt=4448 charge=3930 modemRailChargemAh=0 wifiRailChargemAh=0 +running +wake_lock +wifi_full_lock +wifi_radio +screen_doze data_conn=lte phone_signal_strength=moderate +wifi +usb_data wifi_signal_strength=3 wifi_suppl=completed +ble_scan +cellular_high_tx_power gps_signal_quality=good fg=u0a324:"com.google.android.wearable.app"
0 (2) 100 -cellular_high_tx_power top=u0a419:"it.unipi.dii.stepcounterexercise"
0 (2) 100 job=u0a221:"com.google.android.apps.photos/com.google.android.libraries.social.async.BackgroundTaskJobService"
0 (2) 100 job=u0a148:"com.google.android.apps.turbo/.nudges.broadcasts.BatteryHistoryLoggerJobService"
0 (2) 100 job=u0a324:"com.google.android.wearable.app/com.google.android.clockwork.companion.hats.jobs.HatsCsatJobService"
0 (2) 100 job=u0a148:"com.google.android.apps.turbo/.deadline.library.DeadlineUpdateJobService"
0 (2) 100 job=u0a148:"com.google.android.apps.turbo/.deadline.library.BatteryFlagService"
0 (2) 100 user=0:"0"
0 (2) 100 userfg=0:"0"
0 (2) 100 tmpwhitelist=u0a189:"broadcast:u0a189:com.google.android.gms.wearable.node.bluetooth.RETRY_CONNECTION,reason:"
0 (2) 100 longwake=u0a324:"*job*/com.google.android.wearable.app/com.google.android.clockwork.companion.hats.jobs.HatsCsatJobService"
+14ms (3) 100 status=discharging +tmpwhitelist=u0a164:"broadcast:u0a164:com.google.android.apps.scone.action.FenceAction,reason:null"
+141ms (2) 100 +job=u0a164:"com.google.android.apps.scone/.awareness.AwarenessJobIntentService"
+142ms (2) 100 -job=u0a148:"com.google.android.apps.turbo/.deadline.library.BatteryFlagService"

dumpsys

The output is divided in several sections, some of them:

- **Battery History**: a time series of power-related events (screen turned on, radio signal, app execution, etc). It is roughly expressed as an action that changes the power consumption of the device.
- **Per-PID Stats**: amount of time processes hold a wake lock
- **Discharge step durations**: the time between battery percentage changes
- **Daily stats**: similar to the previous one, but spanning multiple days with charge and discharge events
- **Statistics since last charge**: contains power-related statistics collected since the device started being powered by its battery
  - **Overall consumption and battery info**
  - **Cellular Statistics**: status and usage of cellular connection
  - **Wi-fi Statistics**: same about WiFi
  - **Bluetooth**: same about Bluetooth
  - **Estimated power use (mAh)** : estimated power consumption of each subsystem and user (uid); in general, one APK corresponds to one user
  - **Wake locks**: statistics about wake locks
  - **Statistics by uid**: the power consumption for the applications (identified by their uids)

dumpsys

- Battery history, composed by a sequence of events

```
Battery History (1% used, 66KB used of 4096KB, 209 strings using 22KB
```

Info about the level of usage of buffers.

```
0 (14) RESET:TIME: 2022-05-09-12-21-51
```

when the statistics were reset, 0 is the starting time

```
0 (2) 100 status=not-charging health=good plug=none temp=282 volt=4448
charge=3930 modemRailChargemAh=0 wifiRailChargemAh=0 +running
+wake_lock +wifi_full_lock +wifi_radio +screen_doze data_conn=lte
phone_signal_strength=moderate +wifi +usb data_wifi_signal_strength=3
wifi_suppl=completed +ble_scan +cellular_high_tx_power
gps_signal_quality=good fg=u0a324:"com.google.android.wearable.app"
```

0 is the time, 100 is the battery level, status is the charging status, health is the health of the battery (good, over-heated, etc), plug about connected usb, temp is the temperature of the battery, volt is the voltage, charge in mAh, list of hw subsistems affecting consumption, some details can be provided e.g. signal strength. + start of something, - end of something

- other records do not provide full information but just changes

dumpsys

- Discharge rate

Discharge step durations:

```
#0: +28m59s30ms to 98 (power-save-off, device-idle-off)
```

amount of time to go from 99 to 98 battery, power save was off, the device was not idle

dumpsys

- Statistics since last charge
  - capacity of the battery
  - time operating on battery
    - realtime = wallclock time
    - uptime = time cpu not sleeping
  - amount of time screen on/off
  - Amount of discharge
  - Amount to completely full battery
  - Screen brightness

Statistics since last charge:
  System starts: 0, currently on battery: false
  Estimated battery capacity: 3930 mAh
  Last learned battery capacity: 3930 mAh
  Min learned battery capacity: 3930 mAh
  Max learned battery capacity: 3930 mAh
  Time on battery: 1h 31m 24s 869ms (99.3%) realtime, 35m 49s 690ms (39.2%) uptime
  Time on battery screen off: 1h 17m 36s 501ms (84.9%) realtime, 22m 1s 323ms (24.1%) uptime
  Time on battery screen doze: 38m 30s 174ms (42.1%)
  Total run time: 1h 32m 2s 255ms realtime, 36m 27s 76ms uptime
  Charge time remaining: 9m 15s 0ms
  Discharge: 142 mAh
  Screen off discharge: 78.6 mAh
  Screen doze discharge: 23.2 mAh
  Screen on discharge: 63.3 mAh
  Device light doze discharge: 70.3 mAh
  Device deep doze discharge: 0 mAh
  Start clock time: 2022-05-09-12-21-51
  Screen on: 13m 48s 368ms (15.1%) 10x, Interactive: 13m 14s 237ms (14.5%)
  Screen brightnesses:
    dark 8m 40s 236ms (62.8%)
    dim 13s 921ms (1.7%)

dumpsys

- Cellular communication, same for Wi-Fi and Bluetooth

```
Logging duration for connectivity statistics: 1h 31m 24s 869ms
 Cellular Statistics:
   Cellular kernel active time: 19m 52s 245ms (21.7%)
   Cellular Sleep time:  1h 5m 16s 36ms (71.4%)
   Cellular Idle time:   13m 10s 27ms (14.4%)
   Cellular Rx time:     11m 49s 51ms (12.9%)
   Cellular Tx time:
    less than 0dBm:  4s 266ms (0.1%)
    0dBm to 8dBm:  12s 243ms (0.2%)
    8dBm to 15dBm:  16s 461ms (0.3%)
    15dBm to 20dBm:  17s 461ms (0.3%)
    above 20dBm:  19s 339ms (0.4%)
   Cellular Battery drain: 69.3mAh
   Cellular data received: 28.62MB
   Cellular data sent: 1.05MB
   Cellular packets received: 23602
   Cellular packets sent: 6087
   Cellular Radio Access Technology:
    hspa 24s 219ms (0.4%)
    lte 1h 31m 0s 650ms (99.6%)
   Cellular Rx signal strength (RSRP):
    poor (-128dBm to -118dBm):  6s 446ms (0.1%)
    moderate (-118dBm to -108dBm):  42m 23s 363ms (46.4%)
    good (-108dBm to -98dBm):  48m 55s 60ms (53.5%)
```

dumpsys

- Estimated power use
  - Ordered by consumption
  - Estimated is **different** from real consumption

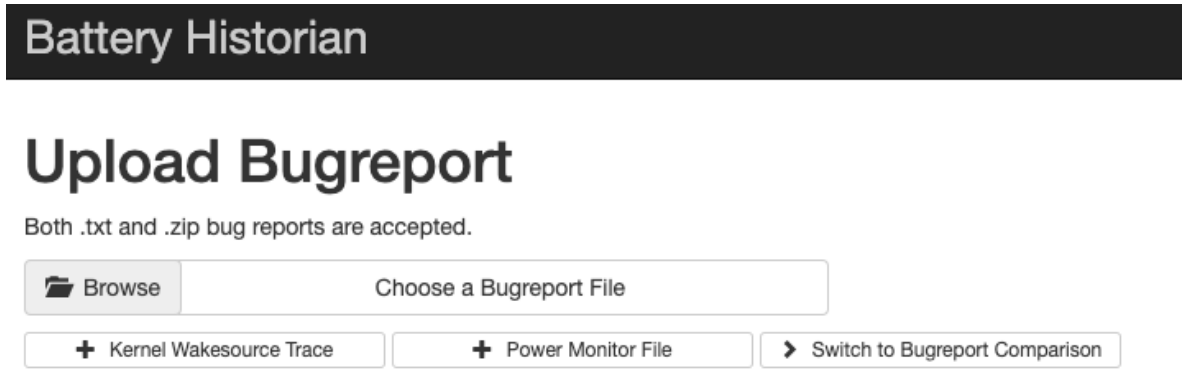
```
Estimated power use (mAh):
    Capacity: 3930, Computed drain: 141, actual drain: 39.3-78.6
    Global
      screen: 63.9 apps: 63.9 duration: 13m 48s 368ms
      cpu: 53.4 apps: 53.4
      bluetooth: 16.0 apps: 5.80 duration: 1h 2m 31s 935ms
      ...
```

Battery historian

- Similar analysis can be carried out using **battery historian**
- Takes as input a trace and visualizes consumption in graphical form

Battery historian

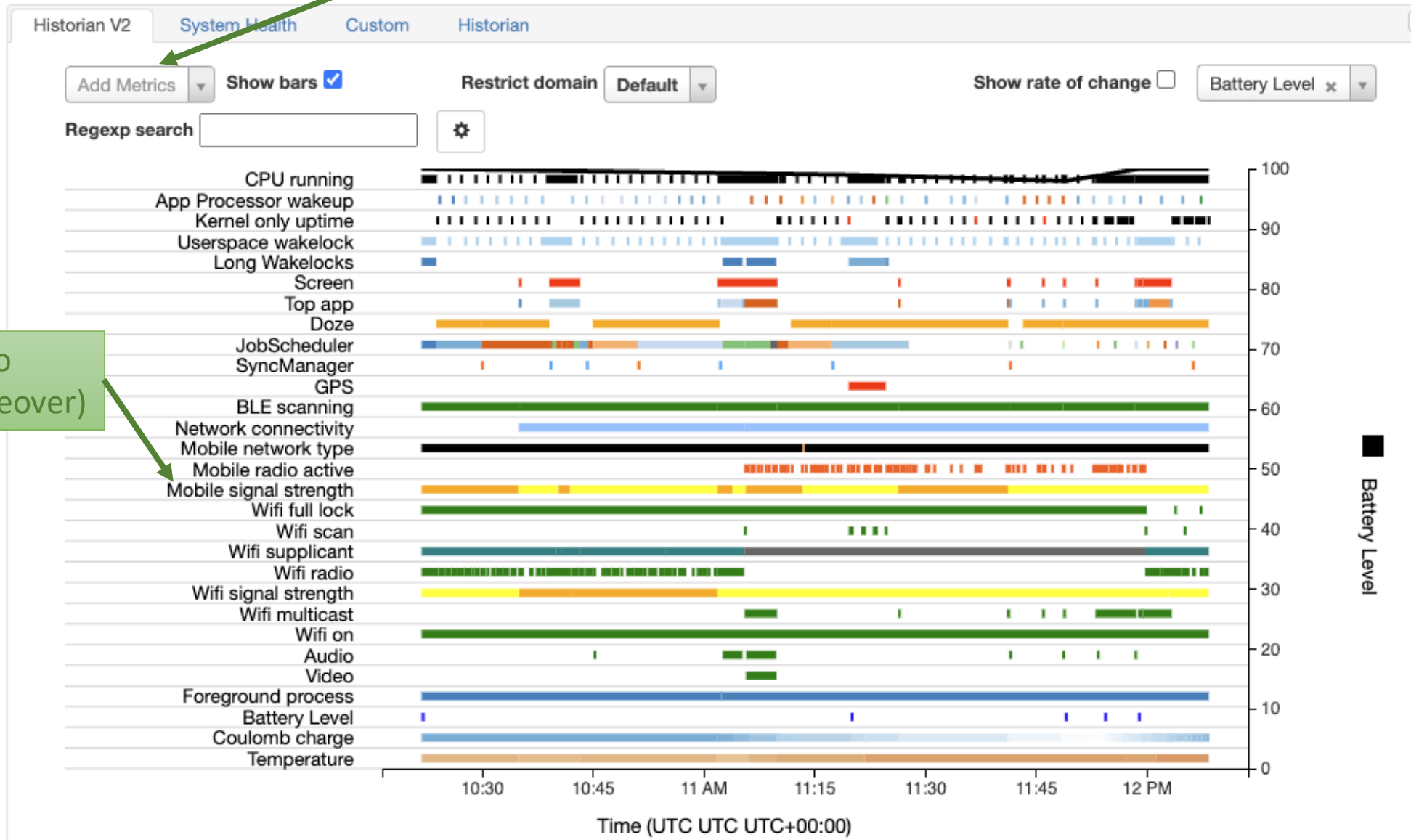
- Installing battery historian
  - easiest way using docker, https://github.com/google/battery-historian
- Reset battery stats
  - **`adb shell dumpsys batterystats --reset`**
- Capture a trace using bugreport
  - **`adb bugreport bugreport.zip`** (android 7.0+)
- Upload trace onto battery historian
  - open browser on *localhost:9999*
- Comparison useful for different versions



Battery historian

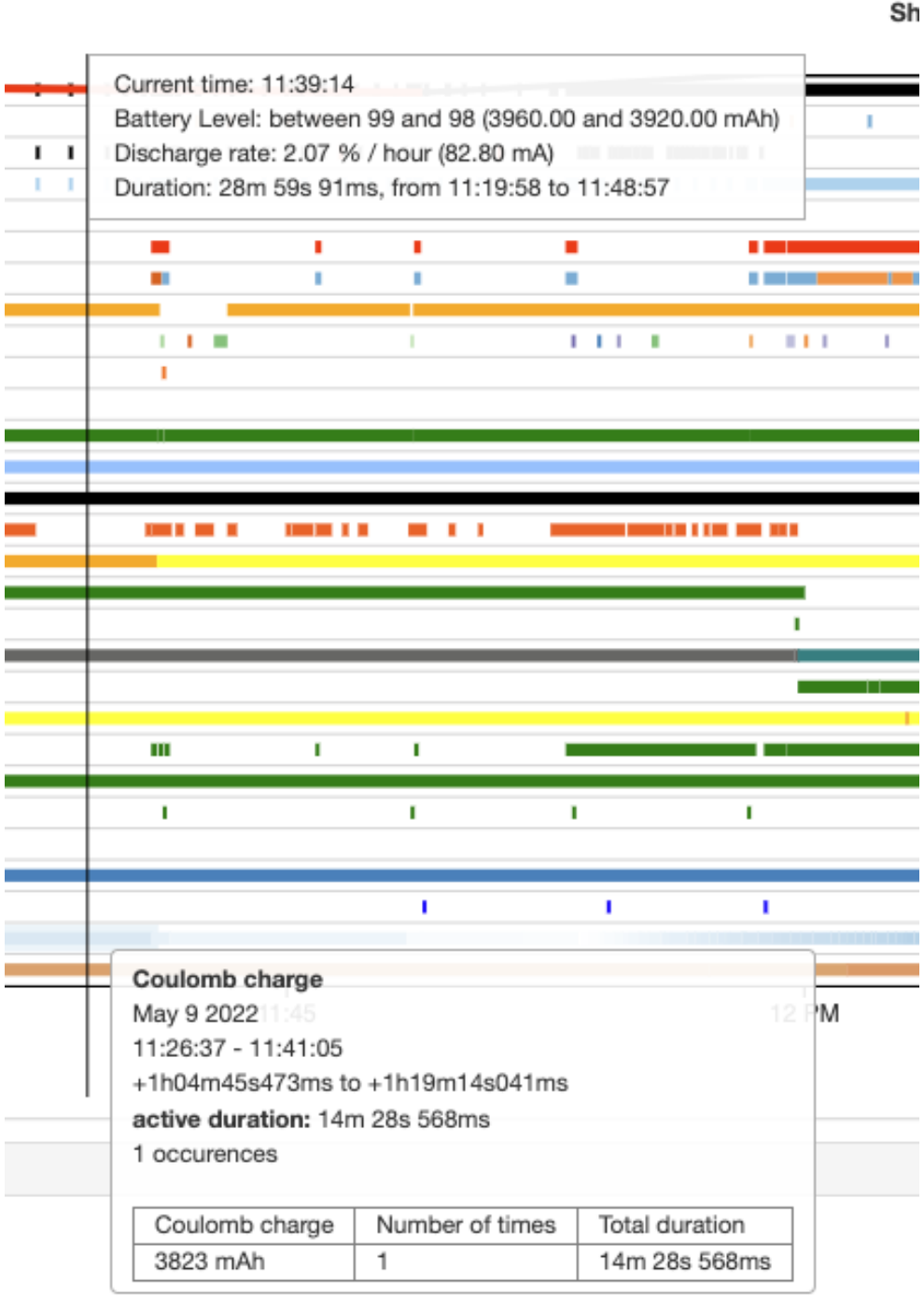
Can add additional metrics

Additional info legend (mouseover)



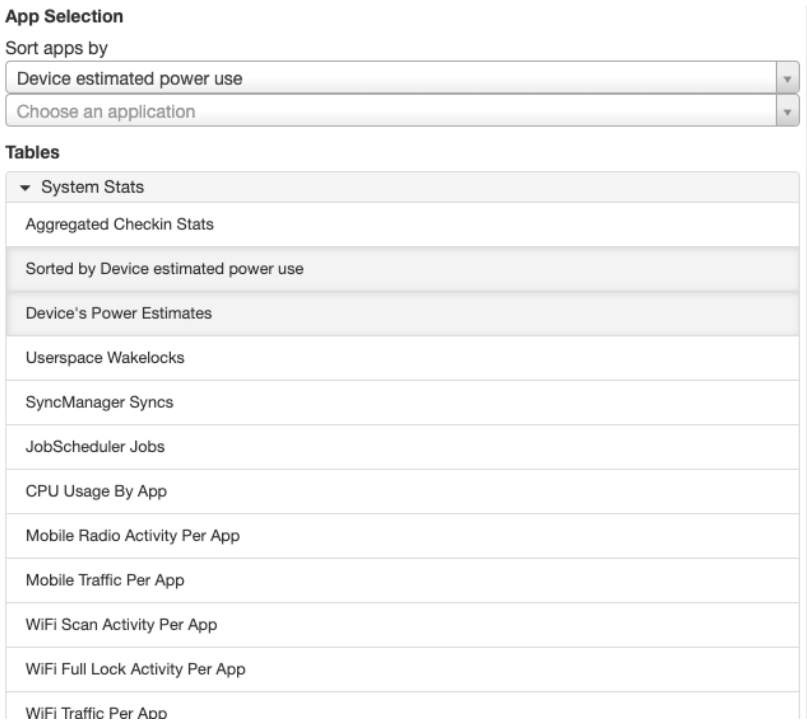
Battery historian

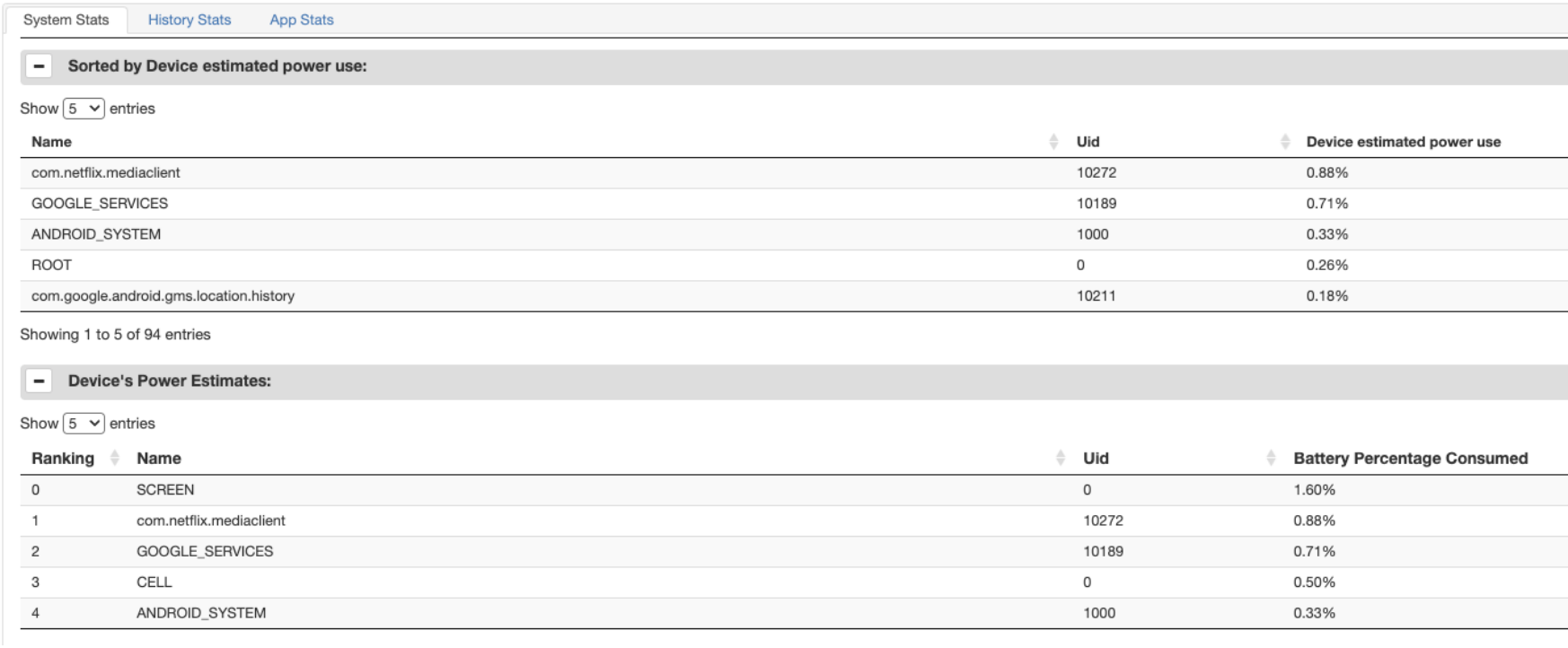
- Info shown when moving pointer on lines



Battery historian

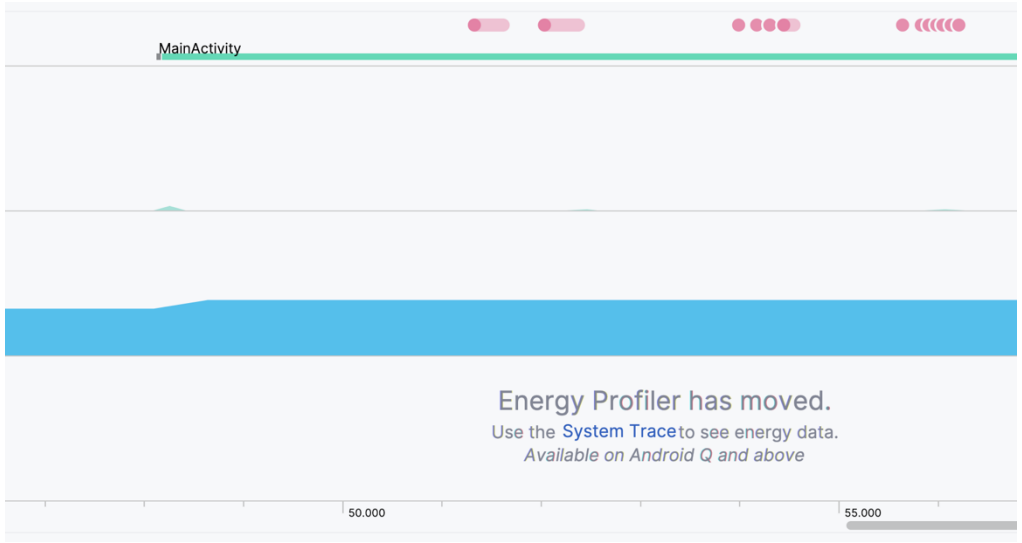
- Lower part provides info about specific apps, if an app is selected upper part shows only relevant information
- Sometimes part of the energy spent is unaccounted
- Look for rapid changes in battery level and consider what is active at that time (wakelocks, transmissions, cpu, ...)

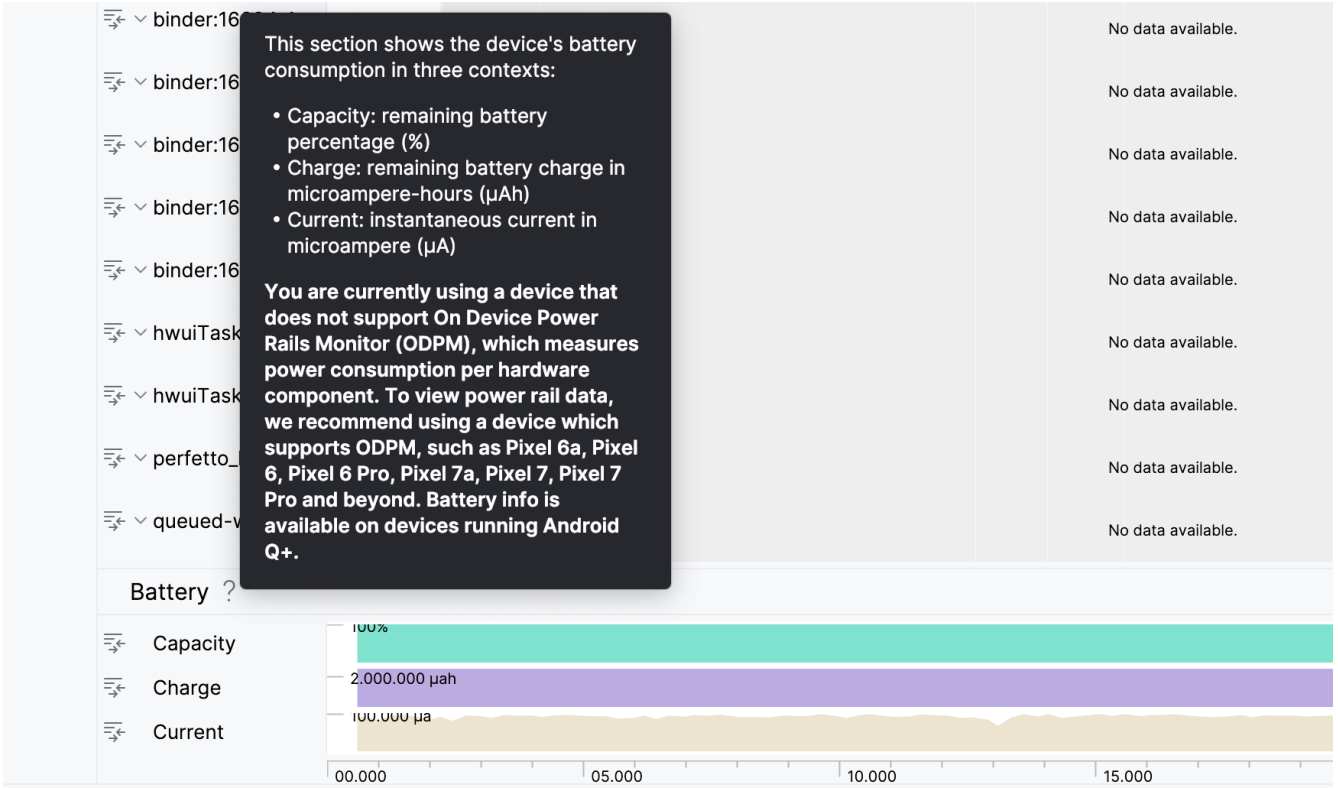




Tracing/profiling

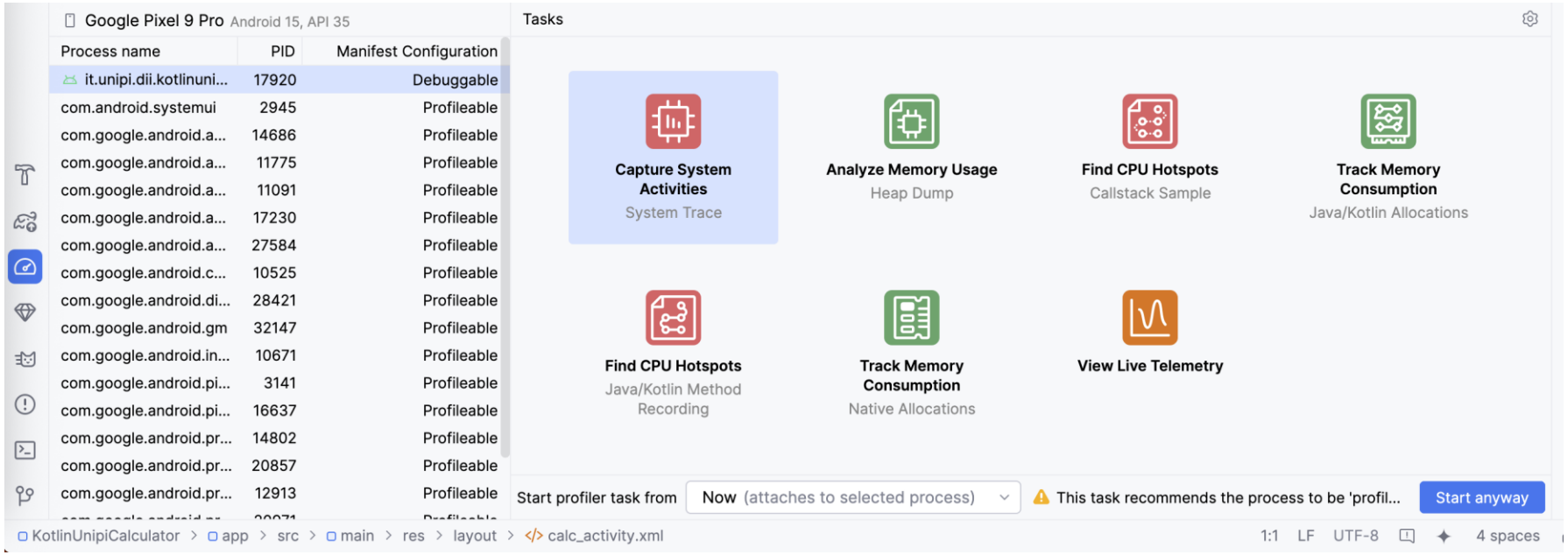
- Recent android versions can use a different tracing mechanism

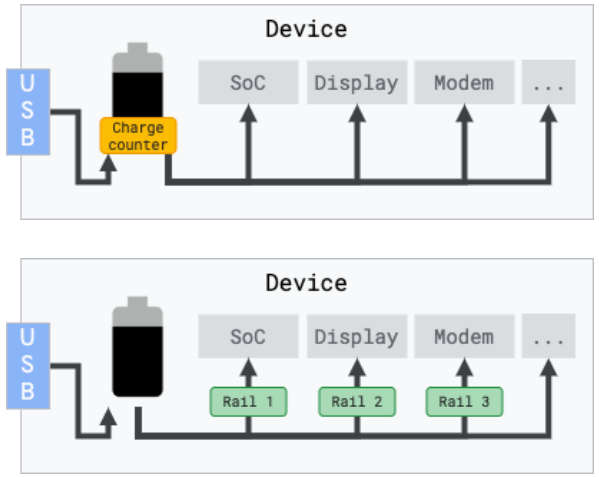




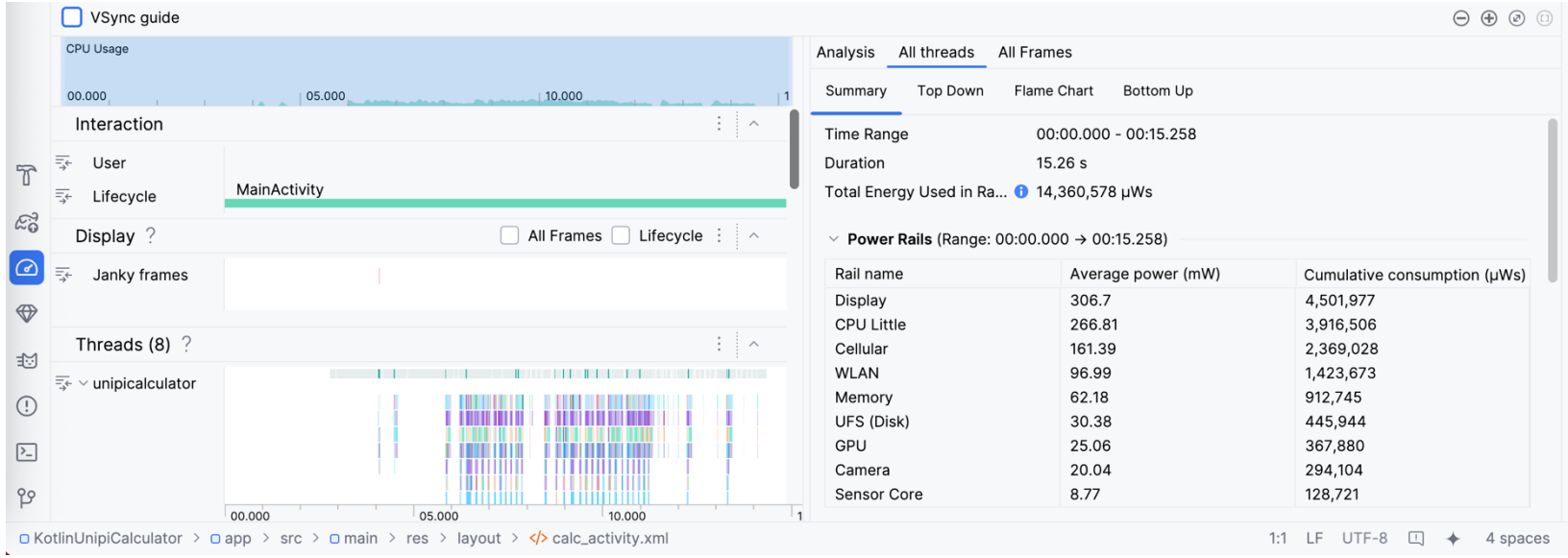
Android Studio Power Profiler

- Recent devices can be energy profiled in Android Studio





ODPM


VSync guide
CPU Usage
00.000
05.000
10.000
Interaction
User
Lifecycle
MainActivity
Display
All Frames
Lifecycle
Janky frames
Threads (8)
unipicalculator
Analysis
All threads
All Frames
Summary
Top Down
Flame Chart
Bottom Up
Time Range 00:00.000 - 00:15.258
Duration 15.26 s
Total Energy Used in Ra... 14,360,578 µWs
Power Rails (Range: 00:00.000 → 00:15.258)
Rail name | Average power (mW) | Cumulative consumption (µWs)
Display | 306.7 | 4,501,977
CPU Little | 266.81 | 3,916,506
Cellular | 161.39 | 2,369,028
WLAN | 96.99 | 1,423,673
Memory | 62.18 | 912,745
UFS (Disk) | 30.38 | 445,944
GPU | 25.06 | 367,880
Camera | 20.04 | 294,104
Sensor Core | 8.77 | 128,721
KotlinUnipiCalculator › app › src › main › res › layout › calc_activity.xml
1:1 LF UTF-8 4 spaces


VSync guide
CPU Usage
CPU cores (8)
CPU 0
CPU 0 Frequency
CPU 1
CPU 1 Frequency
CPU 2
CPU 2 Frequency
CPU 3
CPU 3 Frequency
CPU 4
CPU 4 Frequency
CPU 5
Summary
Time Range
Duration
Total Energy U...
Power Rails
Rail name
Display
CPU Little
Cellular
WLAN
Memory
UFS (Disk)
GPU
Camera
Sensor Core
KotlinUnipiCalculator › app › src › main › res › layout › calc_activity.xml

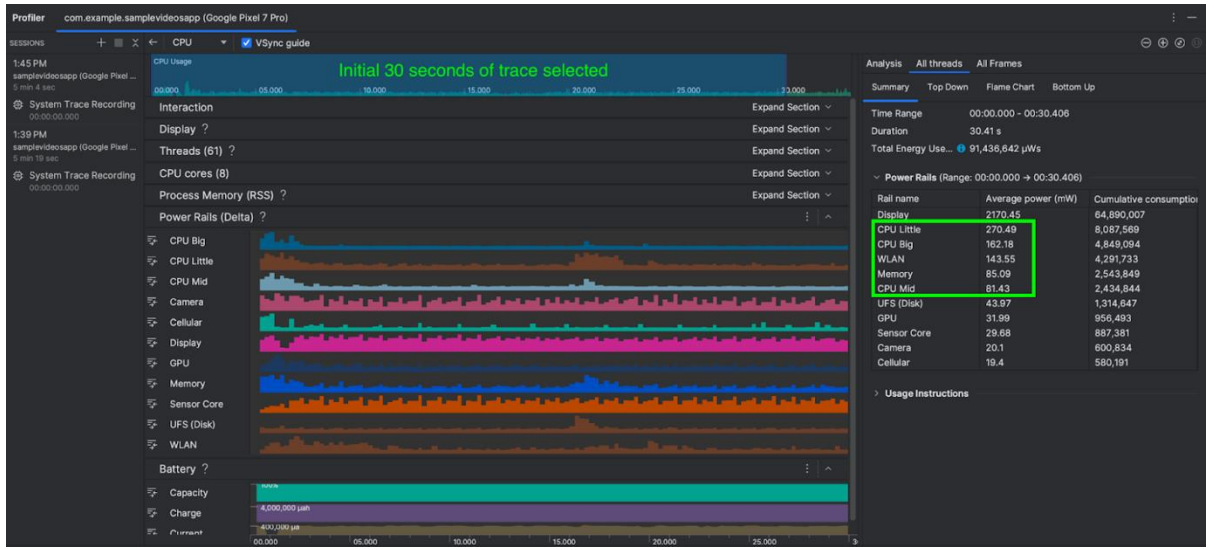

Process Memory (RSS)
Expand Section
Power Rails (Delta)
CPU Little
Camera
Cellular
Display
GPU
Memory
Sensor Core
UFS (Disk)
WLAN
Battery
Capacity
Charge
Current
100%
4,000,000 µah
0 µa
KotlinUnipiCalculator › app › src › main › res › layout › calc_activity.xml

Android Studio Power Profiler

A: average power: 1.3 W
B: average power: 0.7 W

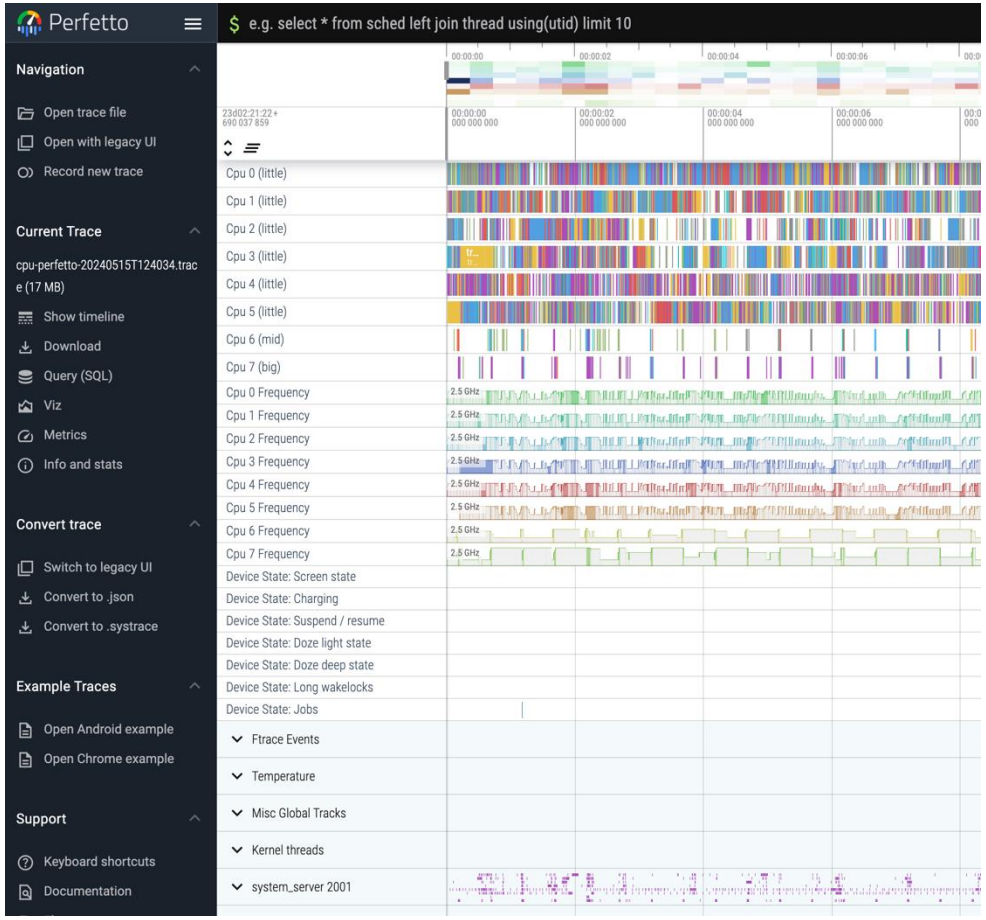
- It can be used to run A/B tests
- Example: a video streaming app shows a 4K video trailer when a movie is selected
  - How much energy is saved if a lower resolution is used?
  - A: the app is run with 4K video trailer, power is measured
  - B: the app is run with a lower resolution, power is measured

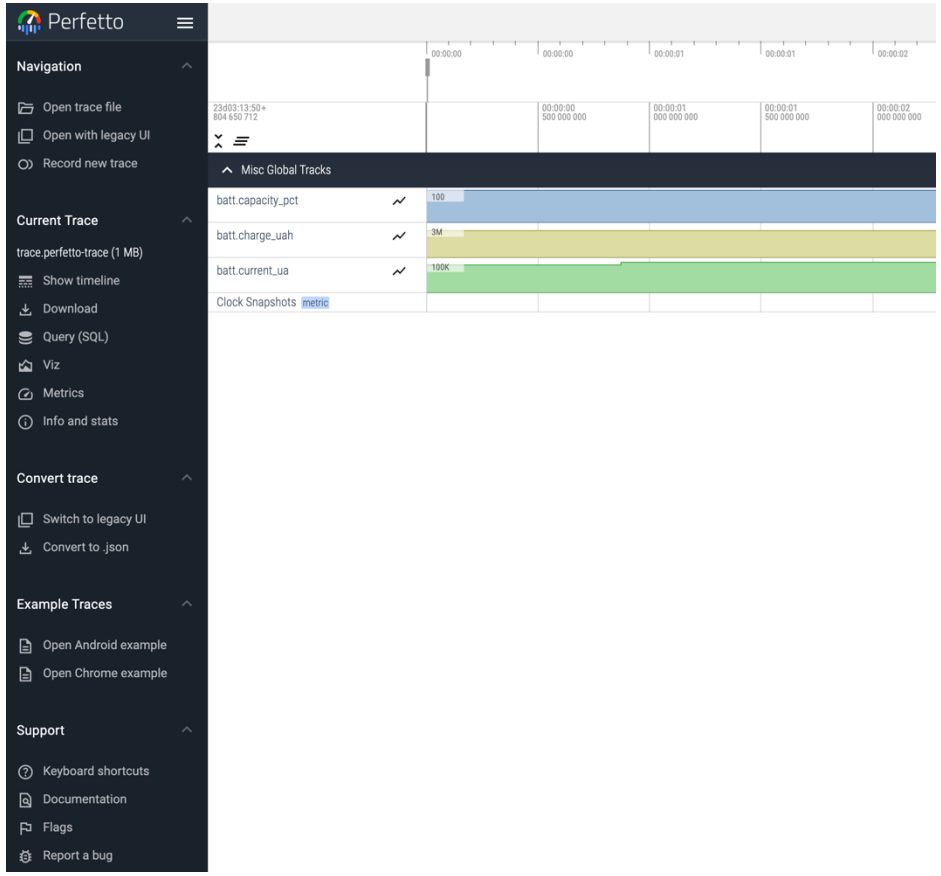
Trace class can be used to mark specific code sections



Perfetto

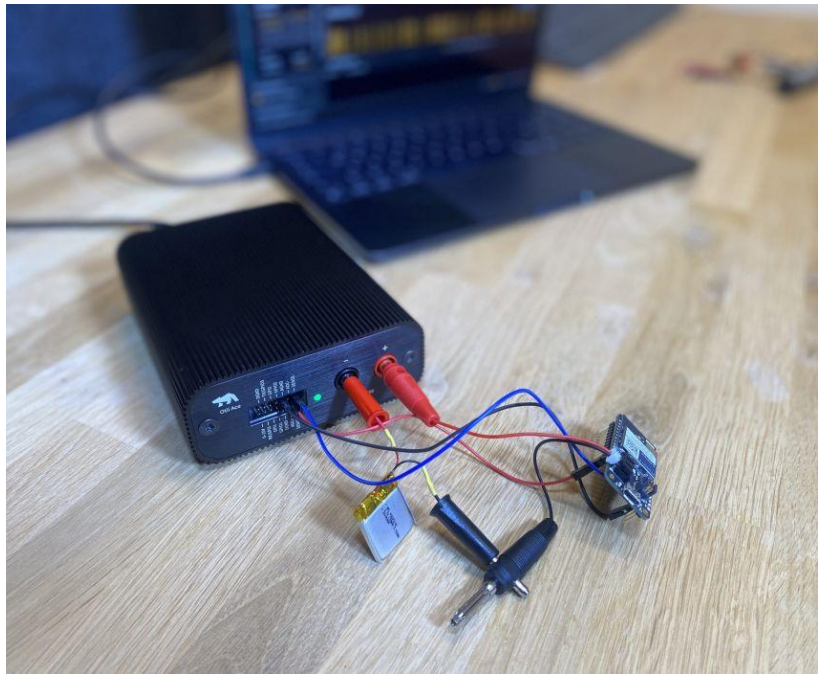
- perfetto.dev
- trace can be exported from Android Studio and imported into Perfetto





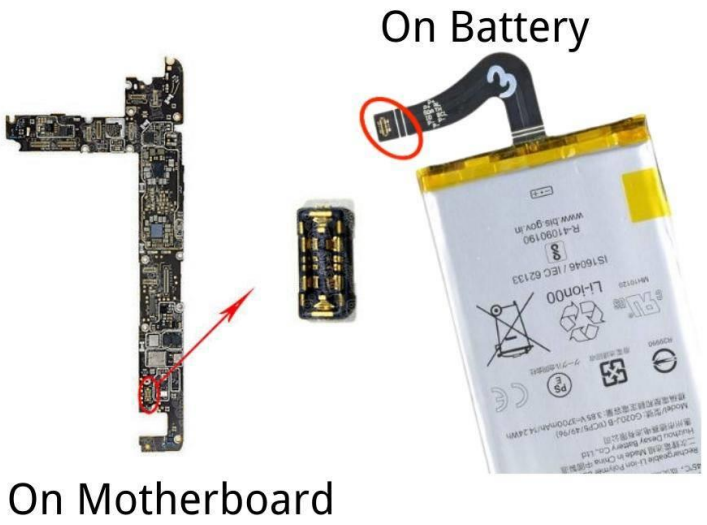
Hardware power monitors

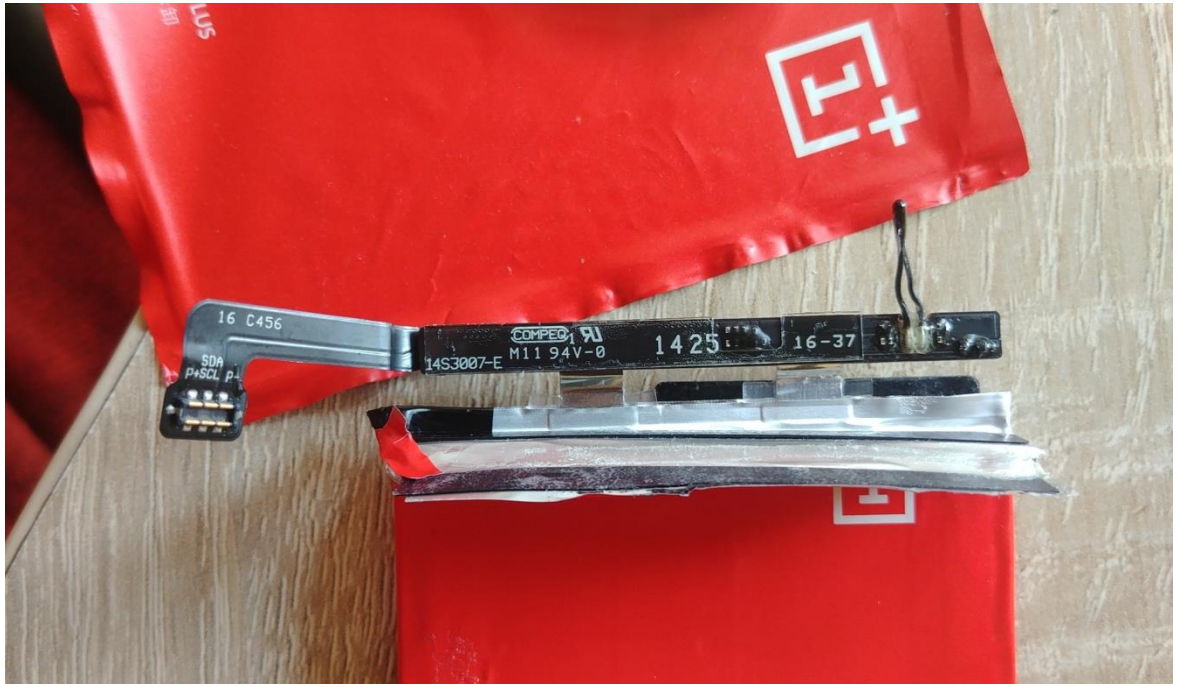
- Placed between the power source of the component and the device itself
- Some power monitors also replace the power source
  - Monsoon Power Monitor or the Otii Ace/Arc
  - Can be controlled via software using a Python API
  - They measure and power small electronic devices
  - There are many power meters but for evaluating the energy consumption of software, they must be controlled by using an API



Hardware power monitors

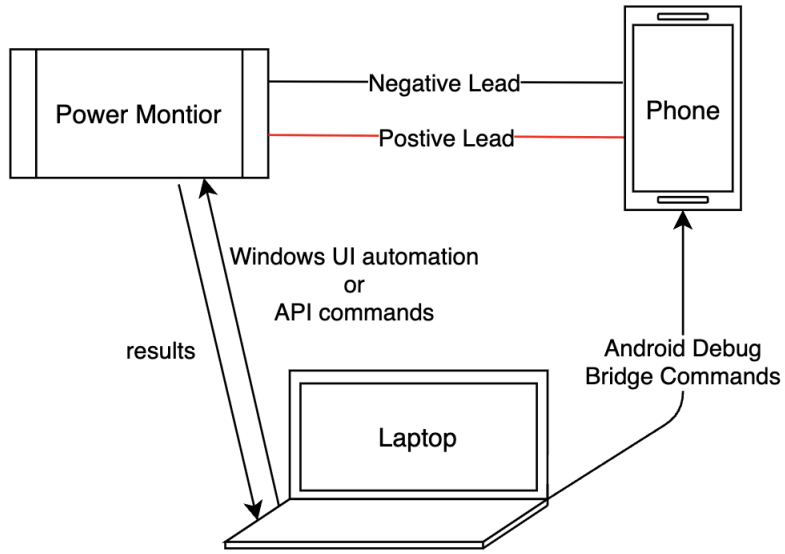
- To connect the power monitor to the smartphone
- Not only power: TEMP, VOLT, State of Charge (via I2C)





Hardware power monitor

- The Otii Arc/Ace can be controlled my means of an application running on the PC or through a Python API
- The output voltage must be set in the 3.6-4.1 v range
- The execution of the app under test must be consistent across experiments
  - adb can be used to install, start and stop an app version
  - Interaction with UI must also be automated
    - Appium, UI Automator, …
- Problem: to automate the execution, adb communicates via USB, which provides unwanted power
  - use a wireless ADB connection
  - turn off the USB port programmatically (instructions must be known in advance)

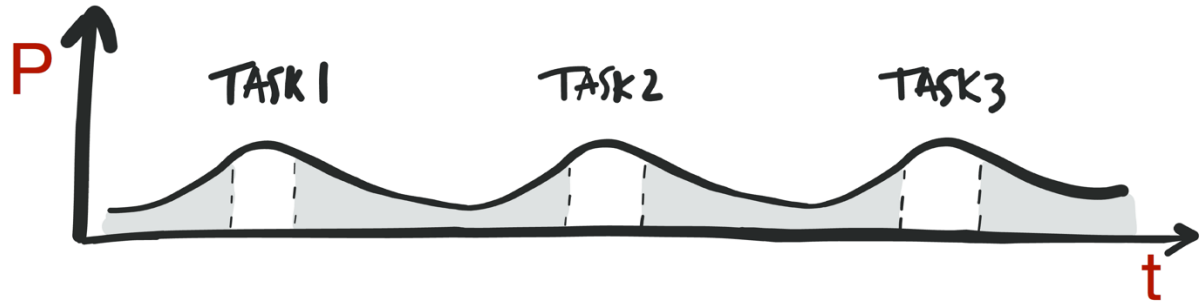


Confounding factors

- Many applications and system processes are always in execution
  - checking for incoming emails, looking for Wi-Fi APs, system updates, collection of position fixes, ...
  - changes in the environment (brightness, sensor readings)
- Variability must be reduced
  - uninstall all unnecessary applications
  - turn off unused network interfaces (cellular, bluetooth if using wifi)
  - turnoff location services

Comparing versions of same app

- Define a reproducible scenario
- Execute and measure version A (say 30 times)
- Execute and measure version B (again 30 times)
- Wait for some time between one execution and the next one
  - tail energy consumption of one execution can bias the measurement of the next execution
- Executions of A and B should be shuffled to avoid the execution order introducing some bias (this also limits the impact of changing conditions)
- Avoid thermal throttling



©Cruz L.