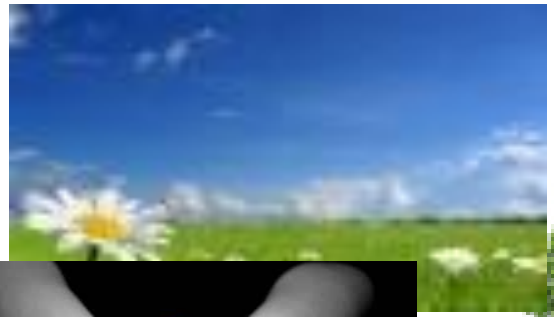
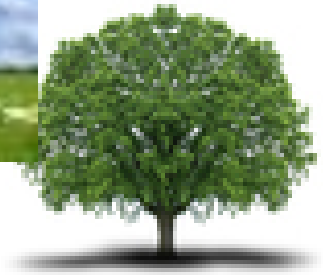
An Energy Abstraction Layer for Mobile Computing Devices

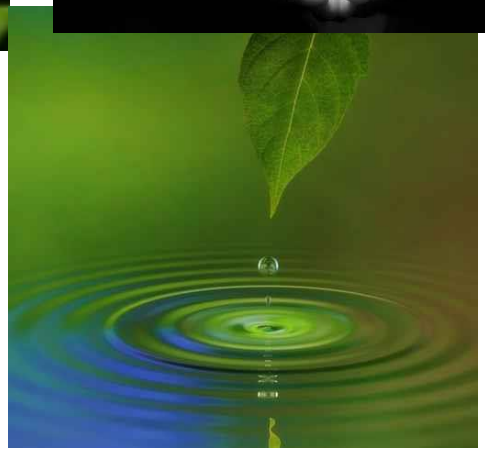
Mirco Josefiok , Marcel Schröder, Andreas Winter

OFFIS – Institute for Information Technology
R&D Divison Energy

Software Engineering Group
Department of Computing Science
University of Oldenburg







ENERGY-EFFICIENCY
MOBILE COMUPTING DEVICES
ANDROID
POWER
iOS
DEVICE INDEPENDENT
ENERGY
TABLETS
ENERGY ABSTRACTION LAY
SMARTPHONES
ABSTRACTIO
OS INDEPENDENT

Objectives

- Determine how energy and power related information are measured on mobile computing devices
- Specify an abstract measurement specification regarding power and energy related information
- Implement the abstract specification into an API on one concrete platform
- Implement a set of sufficiently accurate measurement techniques
- Abstract measurement capabilities and provide unified access to them independently from the used device
- Validate the implemented specification via a prototypical implementation

What do we need to measure?

Power (P = I * V)

- Amount of energy which can be taken from a battery per second
- It is measured in Watts (which is Joule per second)

Electric current (I)

- Describes the flow of electricity through a medium and
- It is measured in ampere

Voltage (U)

- Describes the difference in electrical potential energy per unit charge
- It is measured in volts

Energy (U_E= P * t)

- Describes is the presence and flow of an electric charge.
- It is measured in Joules or watt seconds

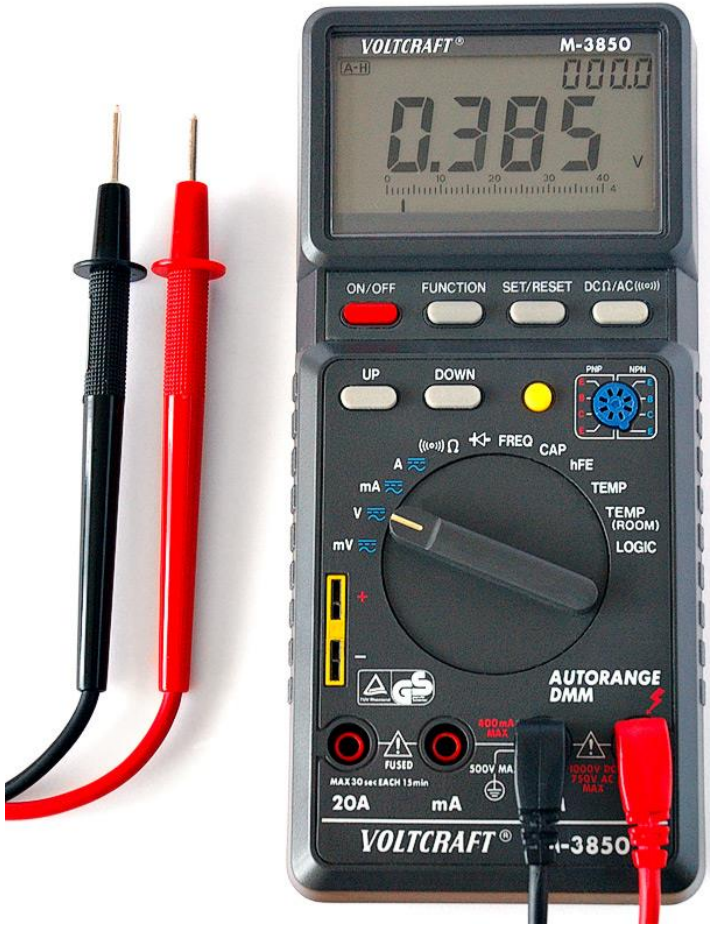
Measuring Energy Consumption

Objective

- Measuring energy consumption caused by software components

Measuring Approaches

- Offline Measuring
  - using external multi meter
  - challenge
    - requires breaking phone housing
- Online Measuring
  - using built in features accessible by software
  - challenges
    - requires standardized interface to access energy
      data on different devices (Energy Abstraction Layer)
    - results in limited accuracy and resolution
    - estimates crude measures



by André Karwath

Measurement on Mobile Devices

Hardware Model Vendor provided

- low level model provided by the vendor

APIs provided by the OS

- public and hidden APIs provide basic information

Knowledge about the Battery

- battery voltage development and capacity information

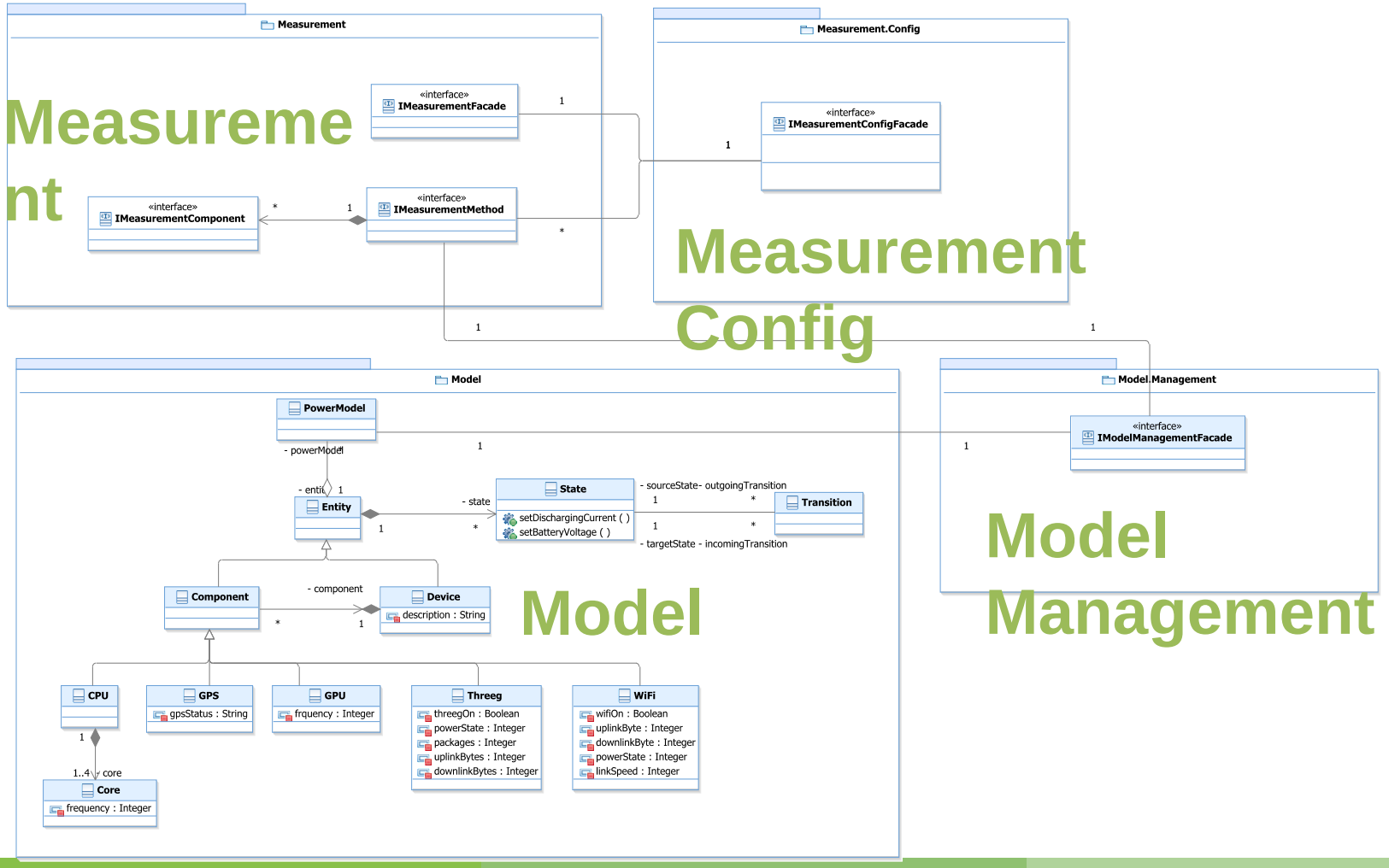
Information provided by the Kernel

- logging of Kernel events

Requirements for an Energy Abstraction Layer

- The EAL must provide power and energy measurement functionality
- The EAL must provide measurement functionality for estimating the running time of each hardware component.
- The EAL has to respect and report accuracy
- The EAL should provide measurement capabilities per application and component
- The EAL should provide suitable error handling
- The EAL must be platform independent.
- The EAL must be device independent
- The EAL should not rely on hardware measurement procedures

Specification of the EAL in UML2

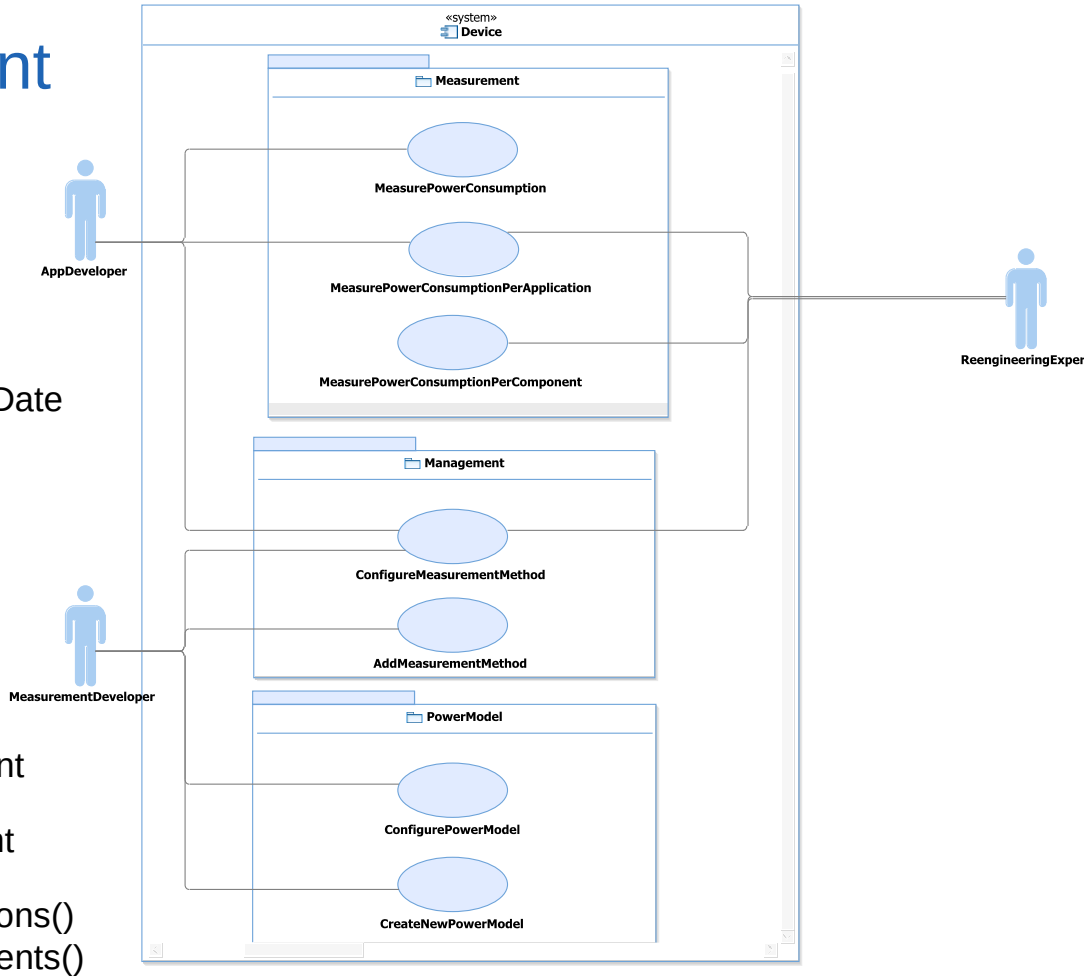


The IPowerMeasurement Interface

- A bundle of methods:

public double getDischargingCurrent()
int getBatteryVoltage()
int getActualConsumedPower()
double getEnergyPerTimeFrame(Date time1, Date time2)
double getMaxBatteryCapacity()
double getActualBatteryCapacity()
float getBatteryTemperatur()
int getBatteryHealthLevel()
String getBatteryTechnology()
int getBatteryStatus()
int getBatteryPluggedStatus()
int getActualConsumedPowerPerComponent(int componentId)
int getActualConsumedPowerPerApplication(int applicationId),
Map getActualConsumedPowerForAllApplications()
Map getActualConsumedPowerForAllComponents()

  - Aligned to the requirements and the respective use case



Implementing the EAL

For this work an Android called
Andromedar
was implementet. It consits of two parts

Library

- Android Library Project
- Implementation of all necessary interfaces
- Three independent measurement methods

Service

- Test application using the library
- Utilizing an Android serivce for recurring measurement

Measurment Methods: Delta-B

$$\Delta B = B(t_0) - B(t_1)$$

$$\Delta Q = \frac{\Delta B}{B_{\max}} * Q_{\max}$$

$$\Delta t = t_1 - t_0$$

$$\bar{I} = \frac{\Delta Q}{\Delta t}$$

Qmax = battery capacity1800 mAh

-> one hour battery at 1800 mA electricity

DeltaB = 1;

DeltaQ = (1/100) * 1800 = 18 mAh

18 mAh = 64800 mAs

Delta T = 60s * 40 = 40 min

I = 64800 mAs / (60s * 40) = 27 mA

Measurment Methods: System File

- Some device provide a regularly updated system file
  - e.g /sys/class/power_supply/battery/batt_attr_text
  - It contains all values provided through BatteryManager
- Provides the actual electric current in mA
  - update interval differs from device to device
  - depends on battery driver and chip
- Provides a direct measurement possibility

Measurment Methods: Energy Model (1/2)

- Estimate consumption of activated components

$$I = \frac{tComponentOn}{\text{measuring interval}} \times getAveragePower(Component)$$

- Total current of all components = Total current
  - I * U = P
  - P * t = E
- Power model via PowerProfile (Hidden API)
  - reads internal XML file provided by vendor
- SensorManager
  - Sensor.getPower()
  - e.g. Accelerometer, Lightsensor, Temperature …

Component	HTC	SGS2	ZTE
ScreenOn	100	75	30
Screen0	16	20,1	11,4
Screen1	48	60,3	34,2
Screen2	80	100,5	57
Screen3	112	140,7	79,8
Screen4	144	180,9	102,6
CPU0	66,6	577	0
CPU1	84	408	0
CPU2	90,8	249	0
CPU3	96	148	45
CPU4	105	55	100
CPU5	111,5		140
CPU6	117,3		
CPU7	123,6		
CPU8	134,5		
CPU9	141,8		
CPU10	148,5		
CPU11	168,4		
CPU IDLE	2,8	2	1,7
GPS	170	1	120
Wifi	4	0,3	23
WifiSend	120	83	200
WifiReceived	120	83	200
MobileOn	300	242	230
MobileSend	300	242	230
MobileReceived	300	242	230
MobileScan	70	82	0
MobileRadio0	3	2,7	90
MobileRadio1	3	3	3
MobileRadio2	3	3	3
MobileRadio3	3	3	3
MobileRadio4	3	3	3
Bluetooth	0,3	0,3	0,8
Audio	88	34	70

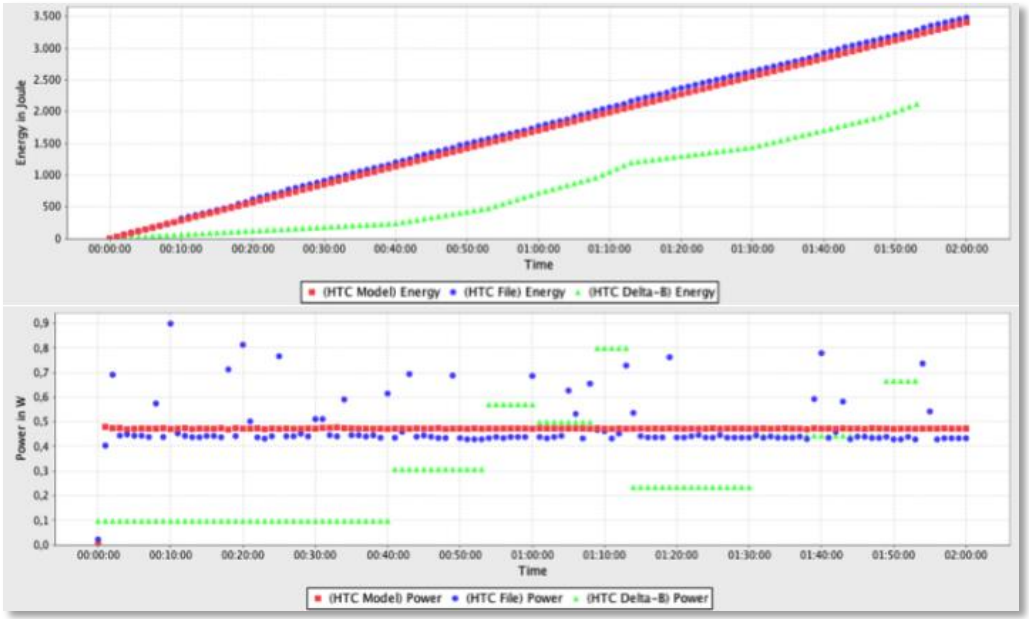
Energy profile (current in mA)

Measurment Methods: Energy Model (2/2)

- Energy model sources
  - Broadcast Receiver
    - threaded polling
  - System file (FileHandler)
    - e.g.:
    - processor: /proc/stat
    - process /proc/[PID]
  - Code modfication
    - either on application or opeating system level
    - e.g. PowerTutor Plus
  - Battery statistics
    - BatteryStatsService (not allowed by API)
    - BatteryStatsImpl
      - Hidden API
- Energy model downsides
  - no official API
  - implementation can chang any time
  - tested for API level 9 and above
- But
  - readings in milliseconds
  - fine grained measurement possible

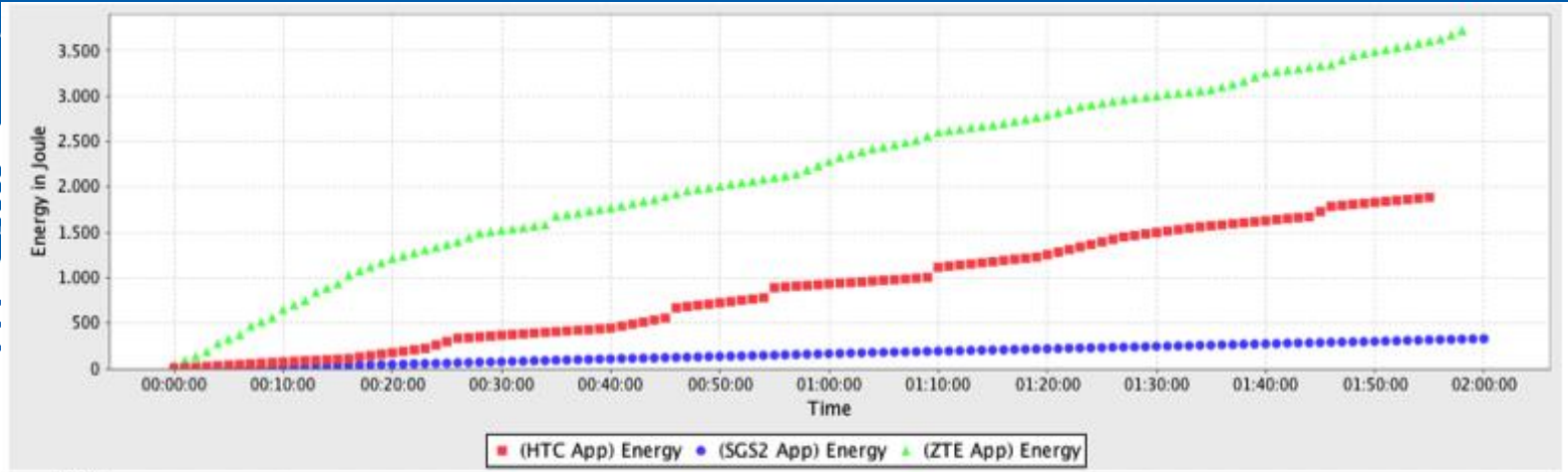
Measurment Methods: Comparison

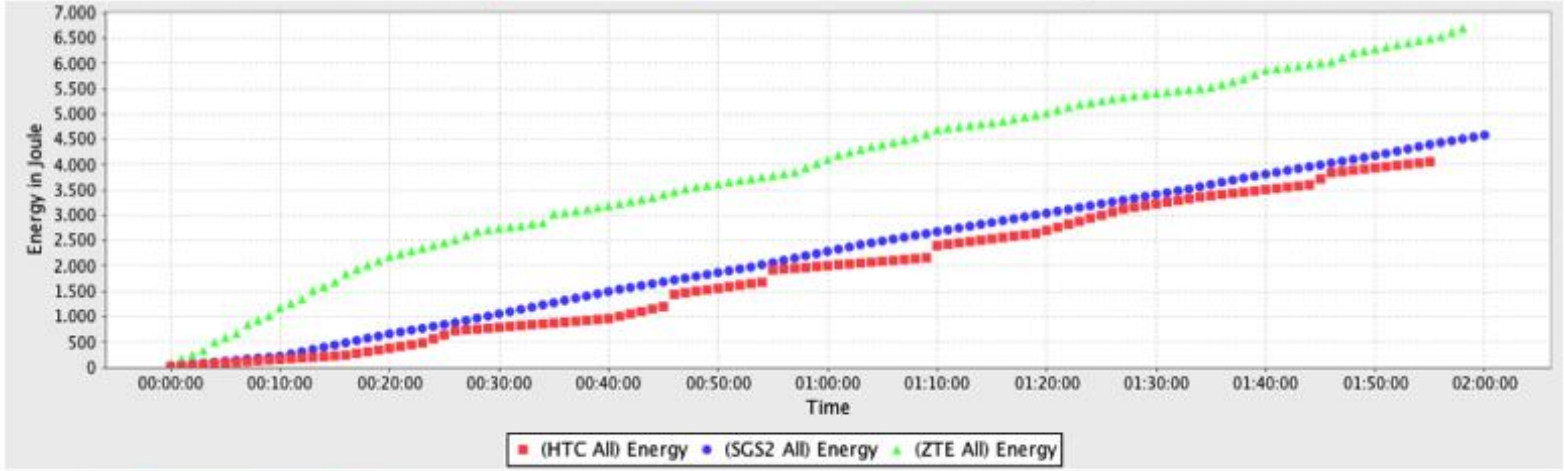
Basic consumption	Current(mA)	Power (mW)	Energy (J)
Model	122,77	472,75	3403,82
File	117,63	483,53	3480,92
Delta-B	84,49	312,61	2119,16

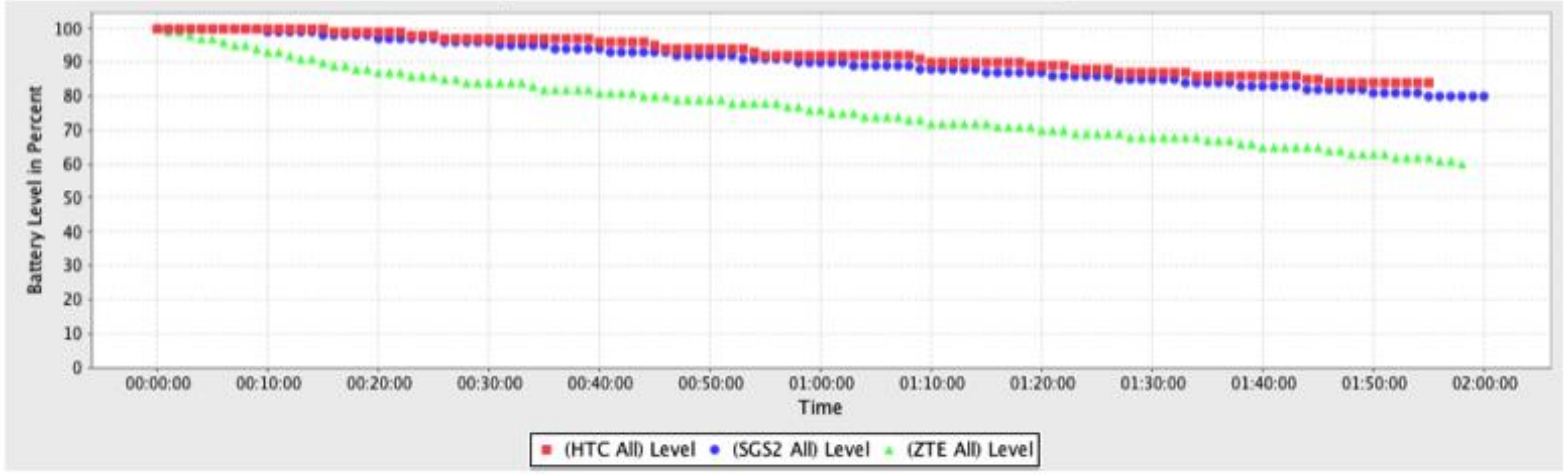


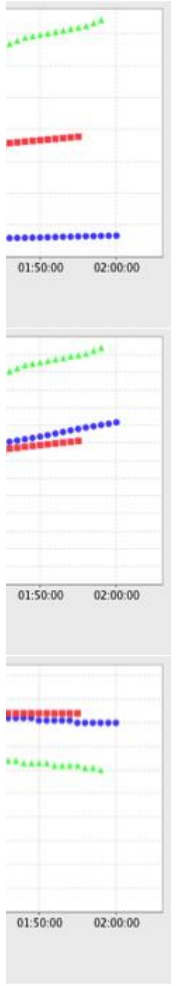
Measurment Methods: Comparison

- Delta-B
  - max. 100 readings
  - available on multiple platform
  - simple to implement
- System File
  - frequency is device dependent (1/min to 6/min)
  - direct reading of current
- Energy Model
  - depends on implementation and device
  - possible needs a background process (ser)
  - accurate and fine grained

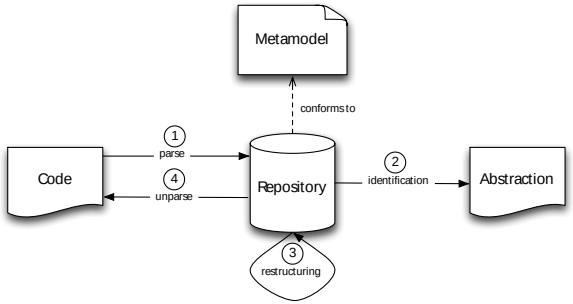

Energy in Joule
3.500
3.000
2.500
2.000
1.500
1.000
500
0
00:00:00
00:10:00
00:20:00
00:30:00
00:40:00
00:50:00
01:00:00
01:10:00
01:20:00
01:30:00
01:40:00
01:50:00
02:00:00
Time
(HTC App) Energy
(SGS2 App) Energy
(ZTE App) Energy


Energy in Joule
7.000
6.500
6.000
5.500
5.000
4.500
4.000
3.500
3.000
2.500
2.000
1.500
1.000
500
0
00:00:00
00:10:00
00:20:00
00:30:00
00:40:00
00:50:00
01:00:00
01:10:00
01:20:00
01:30:00
01:40:00
01:50:00
02:00:00
Time
(HTC All) Energy
(SGS2 All) Energy
(ZTE All) Energy

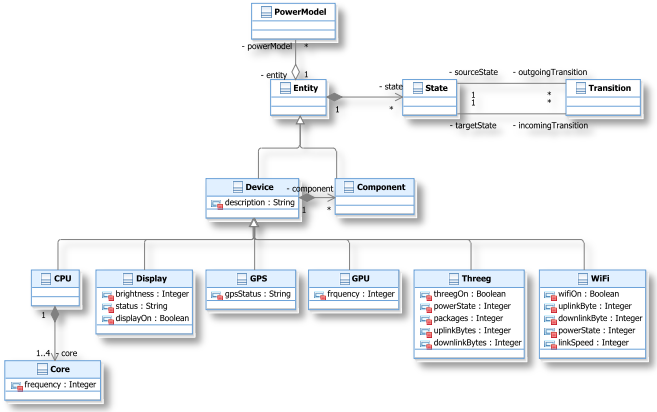

Battery Level in Percent
100
90
80
70
60
50
40
30
20
10
0
00:00:00
00:10:00
00:20:00
00:30:00
00:40:00
00:50:00
01:00:00
01:10:00
01:20:00
01:30:00
01:40:00
01:50:00
02:00:00
Time
(HTC All) Level
(SGS2 All) Level
(ZTE All) Level

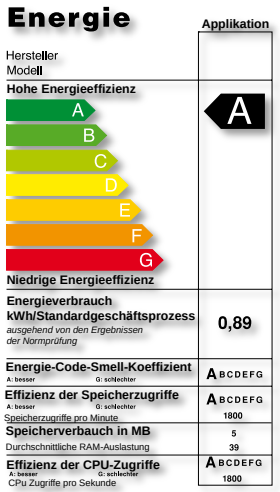

01:50:00
02:00:00

Application and Usage Scenarios



- Detecting Energy Code Smells
  - In a recent Paper [GJJW12] the detection an removal of energy code smells was proposed
  - Energy-Code-Smells are potential wasteful code patterns and can be removed via refoactoring
- Creation of Power Models
  - Power Models, created for each device individually, can improve measurement accuracy
- Eco- or Energy-Labelling
  - Based on results gathered via an application which uses the EAL
  - For easy comparison of similar applications





Lessons Learned

- Power and and energy measurement on mobile computing devices is a relatively young research area
  - Based on the literature study, power modeling was identified as a preferable direction for further research
  - Android was (probably) the most used research platform
- EAL was designed with Android in mind
  - But was kept as platform independent as possible
  - Three measurement methods and a test application were implemented on Android
- Unknown influence of API usage on power consumption
  - High update rates seemed to drain an extensive amount of battery
  - Both implemented measurement method provided comparable results

Outlook and Further Work

- Exploring additional measurement methods
  - more low level approaches possible
  - accuracy of selected measurement methods not clear
- Validation via hardware-measurement
  - Possible only suitable solution for determine accuracy of software measurement methods
- More complex scenarios
  - Define more complex measurement scenarios
    - e.g. using tools like Bot Bot
- Expand implementation
  - More platforms and measurement methods

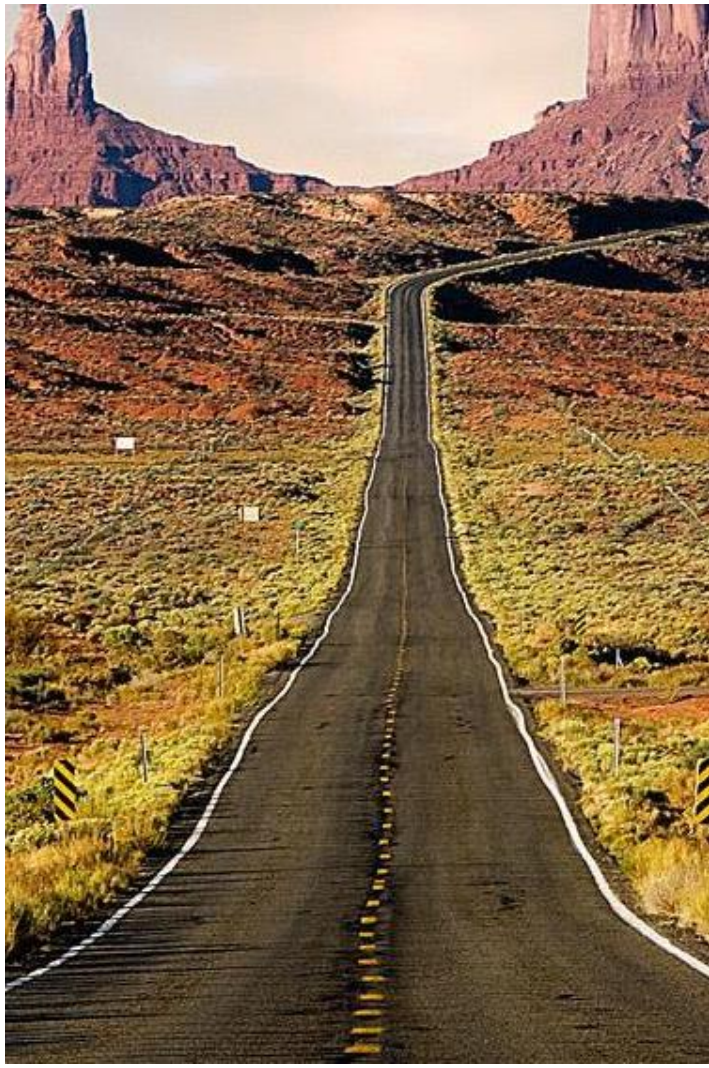


Photo: Bok Sanctuary Path via http://www.sxc.hu