Data rate accuracy with Java, Native and HAL Sensor access

Stefano Bosisio

David Siorpaes

life.augmented

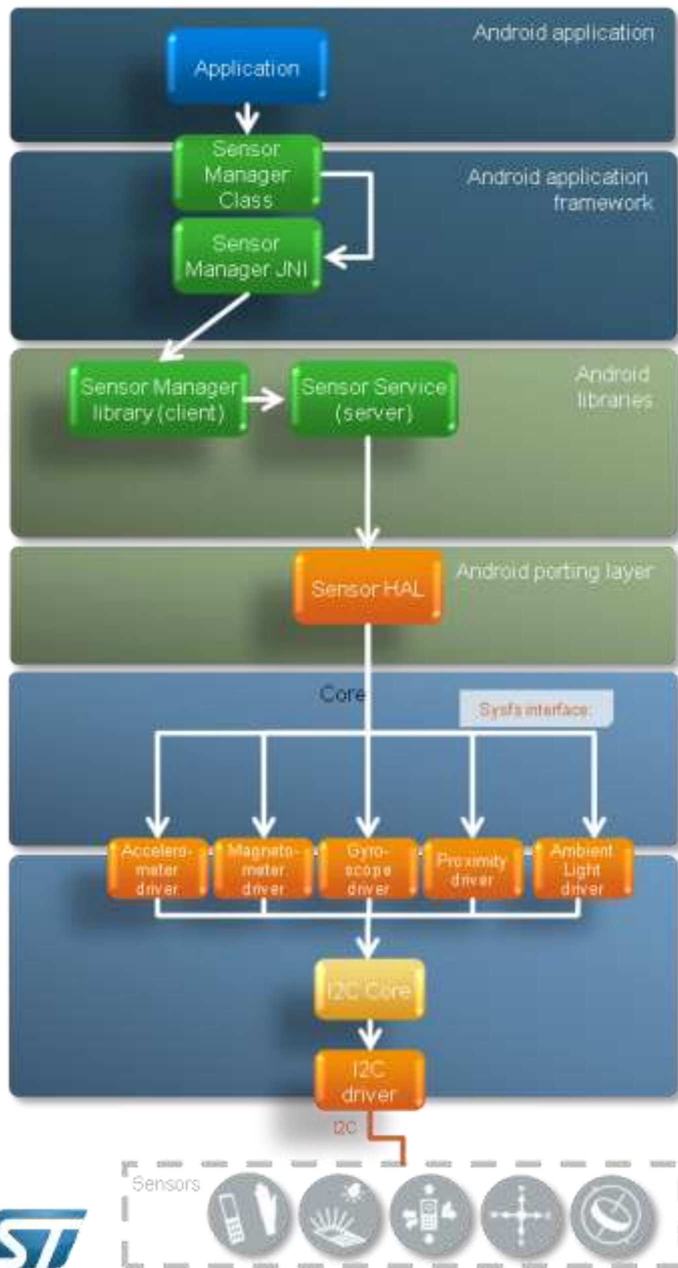
ST Internal

Agenda

- Different way to access sensors in Android
- Goals and background work
- Setup and Profiling infrastructure
- Results
- Next steps

ST Internal

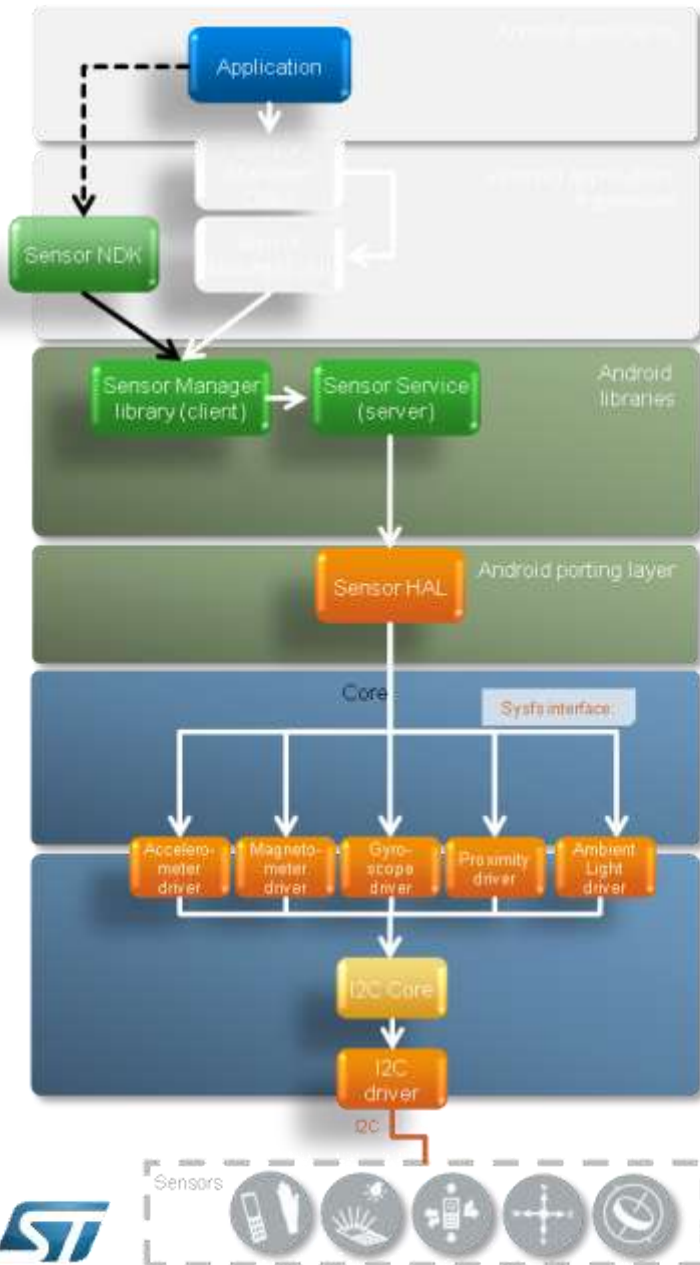
Different way to access sensors in Android



JAVA Activity with SensorEventListener

- Implements *SensorEventListener* interface
- Implements *onAccuracyChanged* and *onSensorChanged* methods
- Gets *SensorManager* instance through *getSystemService* call
- Registers a listener for the wanted sensor(s)
- *SensorManager* reads the sensor(s) for which listeners are registered and notifies interested Activities through Android Sensor Listener mechanism
- *Full stack* overhead
- Asynchronous behavior, Binder involved

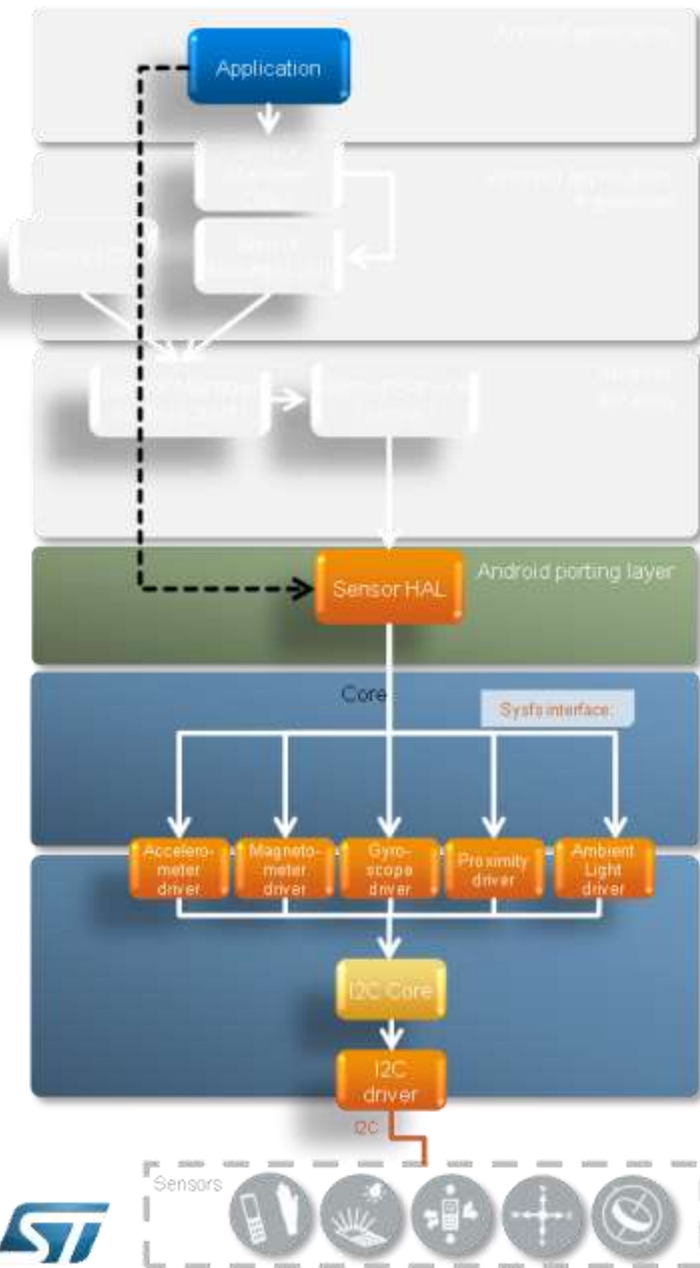
Different way to access sensors in Android



NATIVE Activity with ASensorEventQueue

- Directly access to SensorManager through provided NDK interface
- Uses Native app construct
- Still gets *SensorManager* instance through *ASensorManager_getInstance call*
- *Gets reference to the wanted sensor(s) through ASensorManager_getDefaultSensor call*
- *GETS event through ASensorEventQueue_getEvents call*
- *Avoid JAVA and Android Listeners overhead*
- Synchronous behavior, Binder involved

Different way to access sensors in Android



HAL level (access to device drivers through sysfs)

- App opens vendor HAL module through *hw_get_module* call
- App accesses to sensor's callbacks registered during system initialization enabling and reading sensors through their sysfs interface
- Avoid all Sensor Framework overhead
- Sensors' data are not provided to the application, application must read them
- Synchronous behavior, No Binder involved

ST Internal

Goals and background work

- Goal of this work is to profile the behavior of the different access techniques in STE Android
- Particular focus on
  - data rate accuracy and reliability
  - overheads and what causes them
- Inspired by a previous work by Sensor Platform, Inc

ST Internal

Setup and profiling infrastructure

- Android version: STE ICS Android (4.0.1, ste_u9500_100-eng 4.0.1 ITL41D eng , kernel 3.0.8+)
- Board version: RSTEP1_22_V10
- Accelerometer sensor considered for experiments
- 5000 samples collected for each data rate
- MTUs (Multi Timer Units) used to get precise timestamps
  - 2 available, 4 timers each
  - MTU0, timer 4 with free running configuration selected
  - clocked at ~ 2,4Mhz

ST Internal

Setup and profiling infrastructure

- Simple (JNI) library built to provide access to MTU from application code
- Provided API
  - int initMTU (int samples)
  - void deInitMTU ()
  - void stamp ()
  - int getBufferValue (int n)
- Access to /dev/mem for access different from “root”
  - patch to drivers/char/mem.c
  - chmod 777 /dev/mem
- stamp() is immediately called to get the timer value when sensor’s data become available to the application
  - at the beginning of *onSensorChanged* for JAVA Activity case
  - after *ASensorEventQueue_getEvents* for Native activity case
  - after poll callback for *HAL level* case

ST Internal

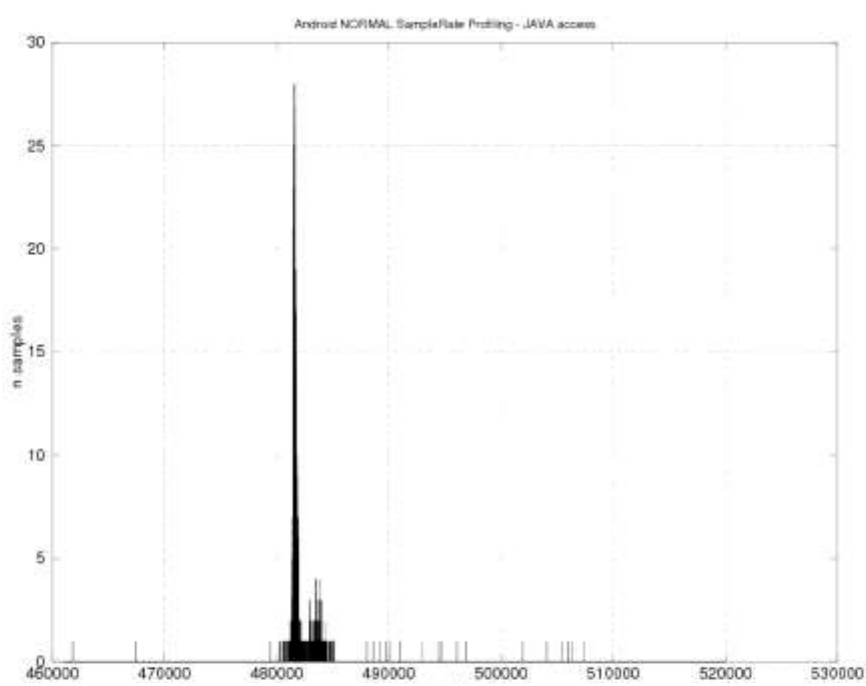
Results

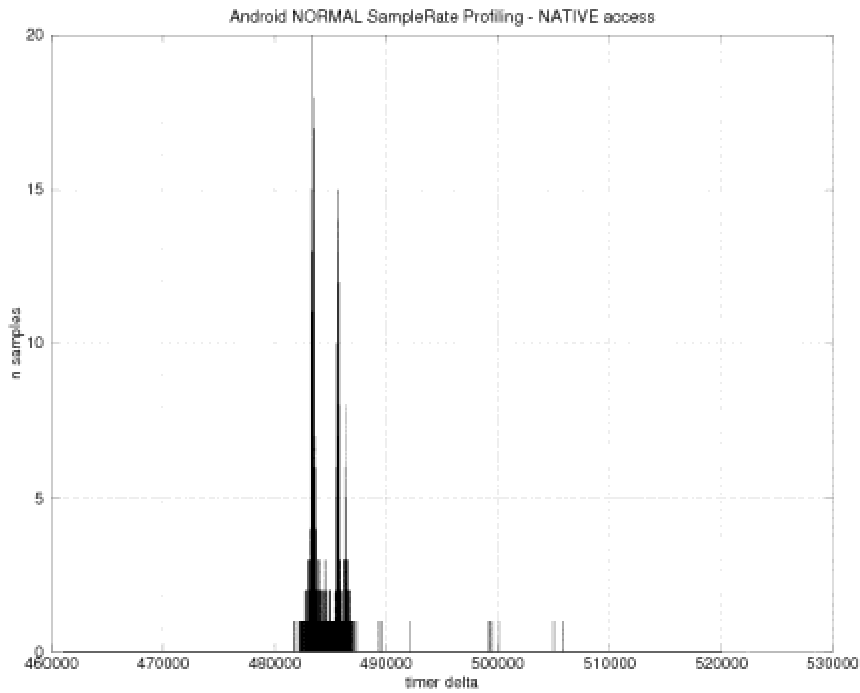
	NORMAL (200ms)				UI (60ms)			
	min	avg	max	std	min	avg	max	std
JAVA Activity	461807	481851 (200ms)	507376	1180	160496	161673 (67ms)	181155	771
NATIVE Activity	481692	484395 (201ms)	505753	1138	143366	147844 (61ms)	162449	1028
HAL Level	481027	481348 (200ms)	519308	1343	145035	145267 (60ms)	183320	1382

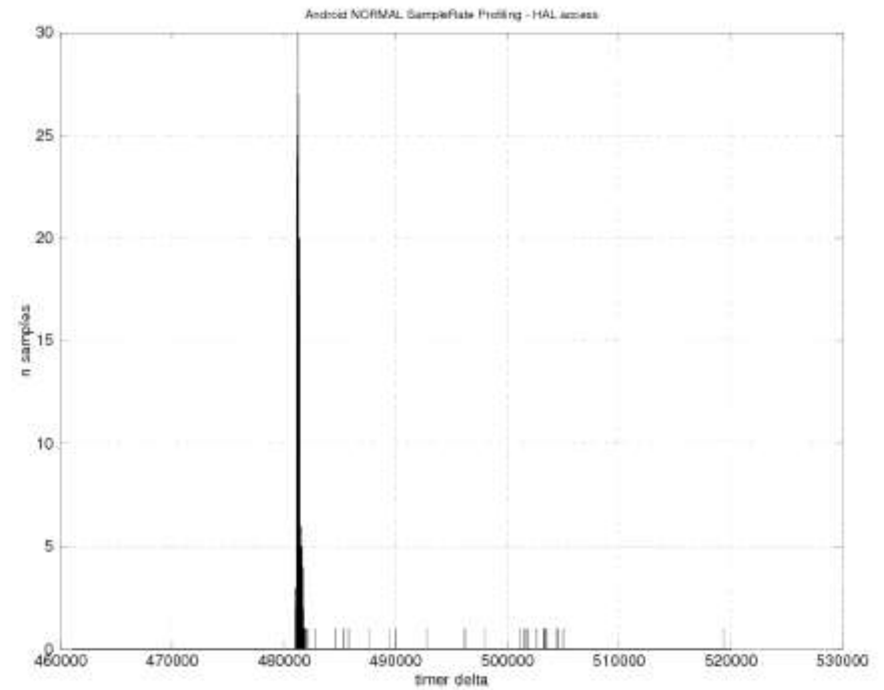
	GAME (20ms)				FASTEST			
	min	avg	max	std	min	avg	max	std
JAVA Activity	16239	49695 (20ms)	75421	1271	217	3859 (1.60)	114406	10359
NATIVE Activity	49520	51504 (21ms)	73294	904	51	3813 (1.58ms)	28863	630
HAL Level	49004	49392 (20ms)	73056	2015	40	4932 (2ms)	808382	

ST Internal

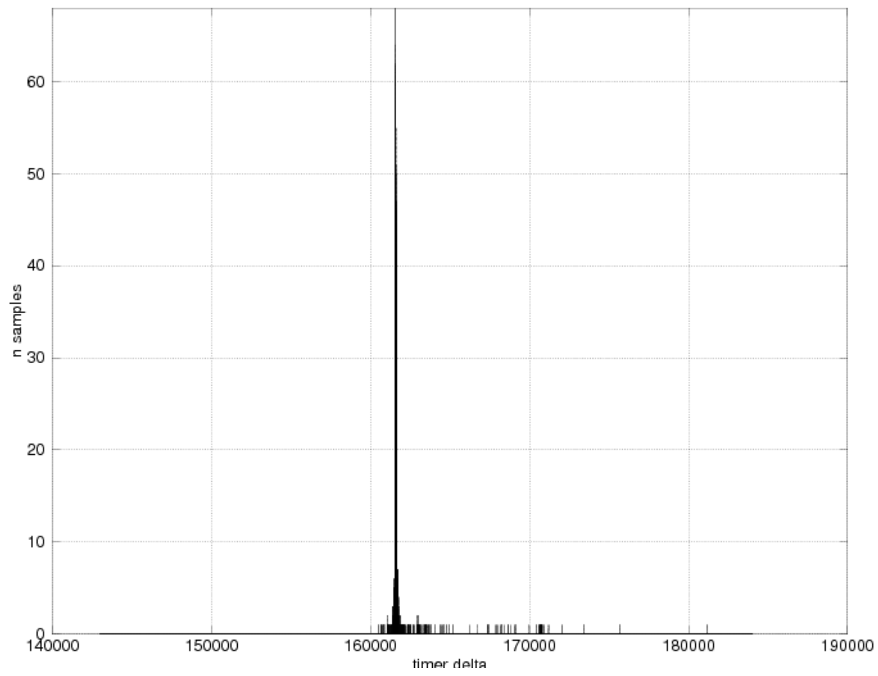
Results NORMAL rate

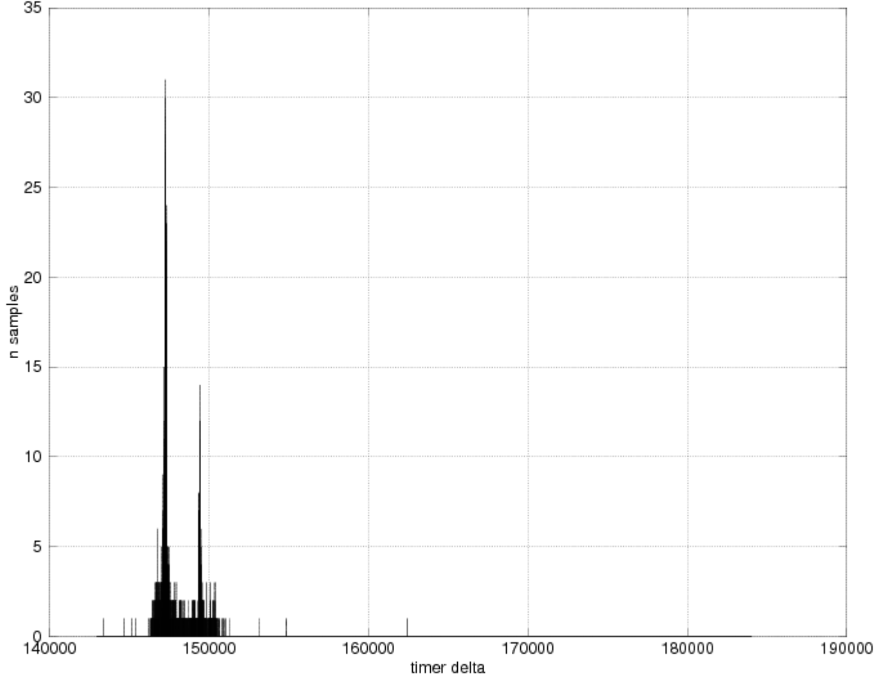


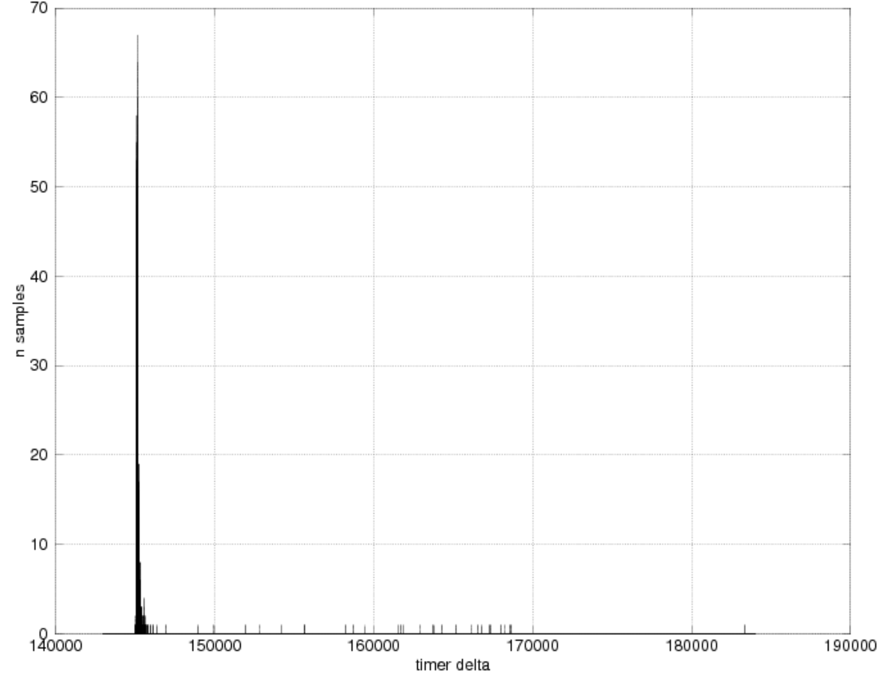




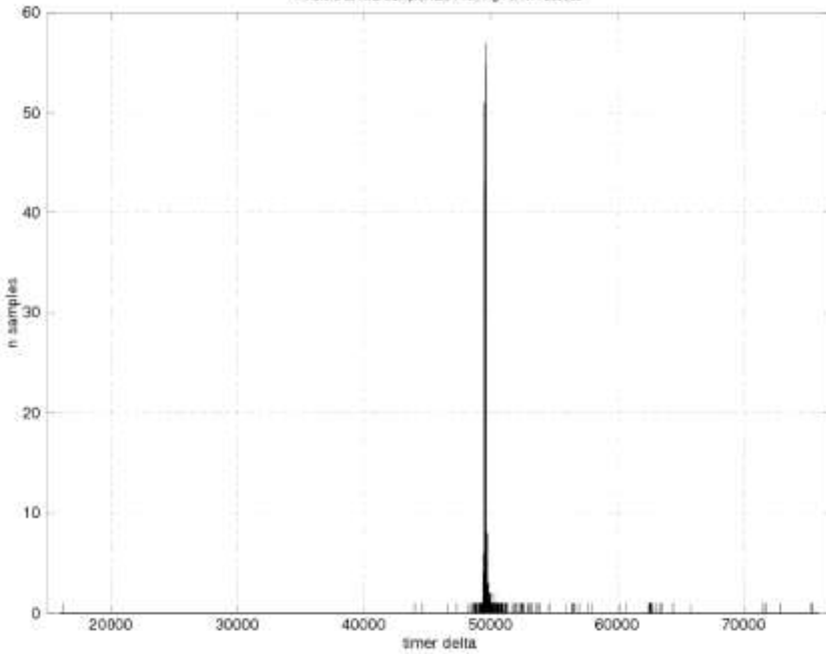
Results UI rate

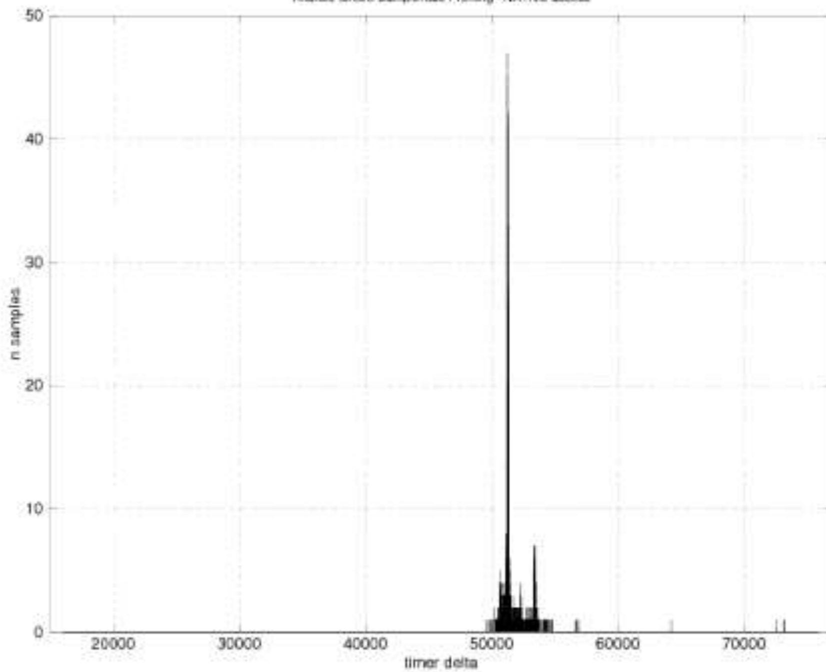


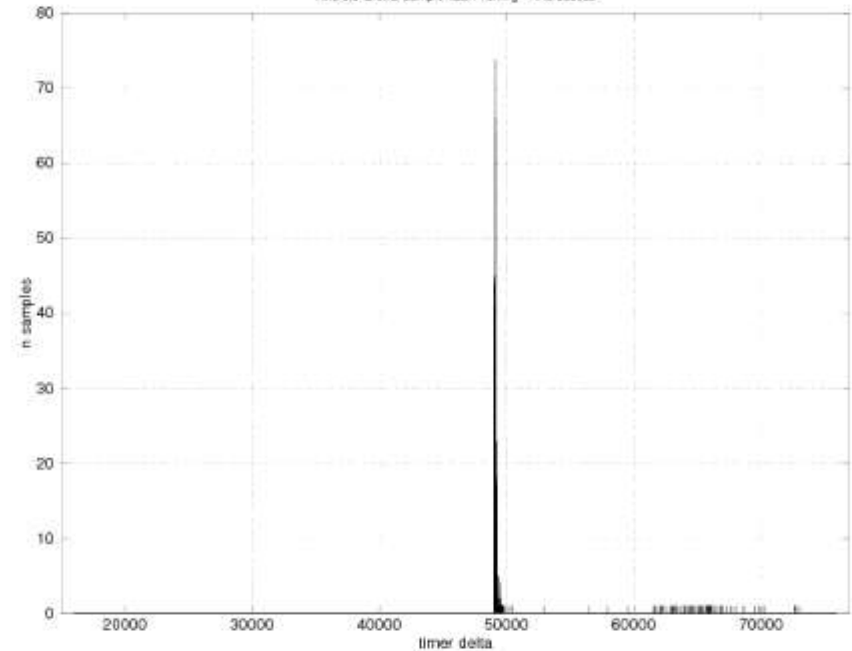




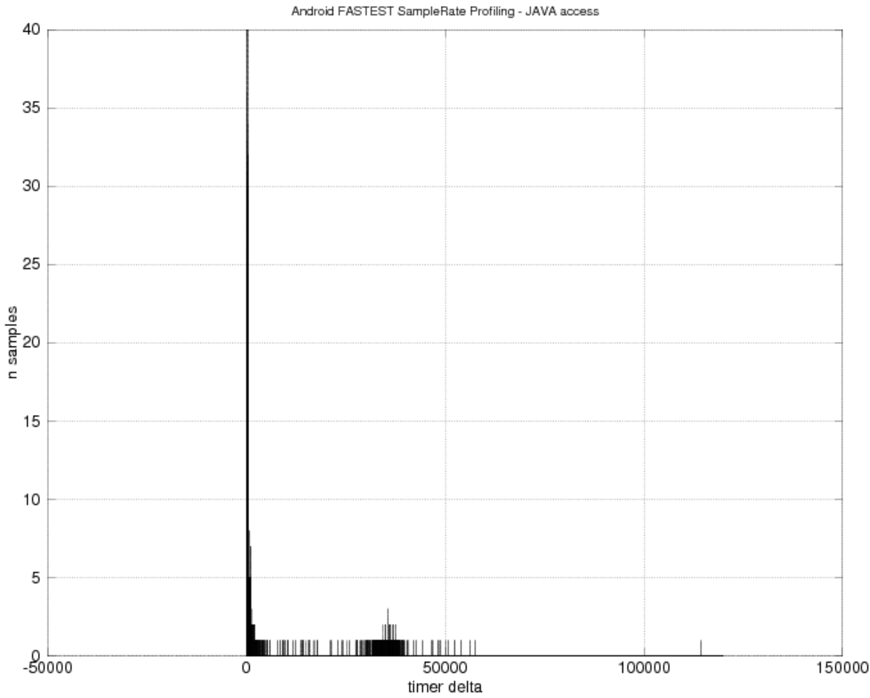
Results GAME rate

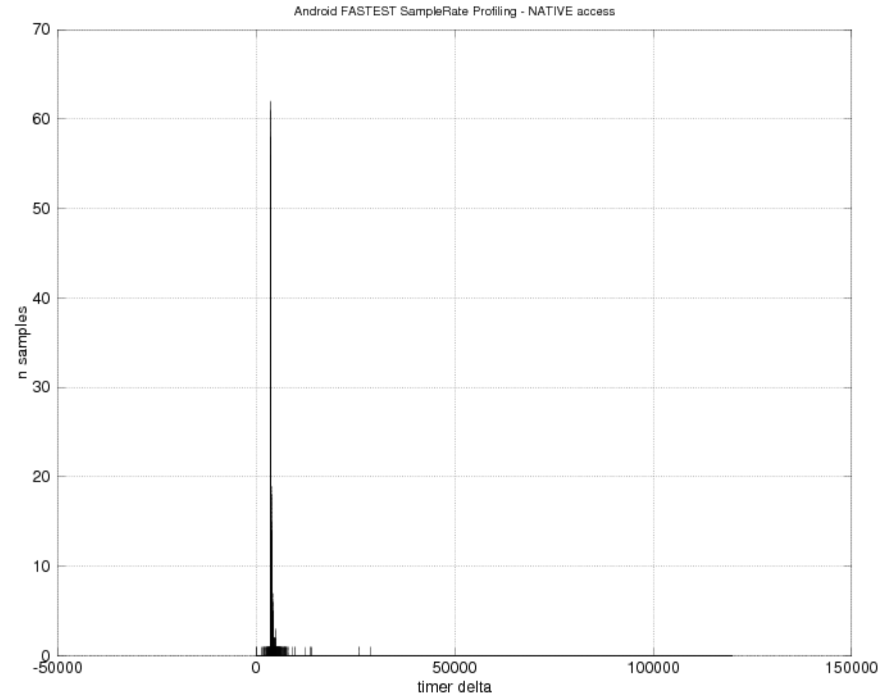






Results FASTESTrate





ST Internal

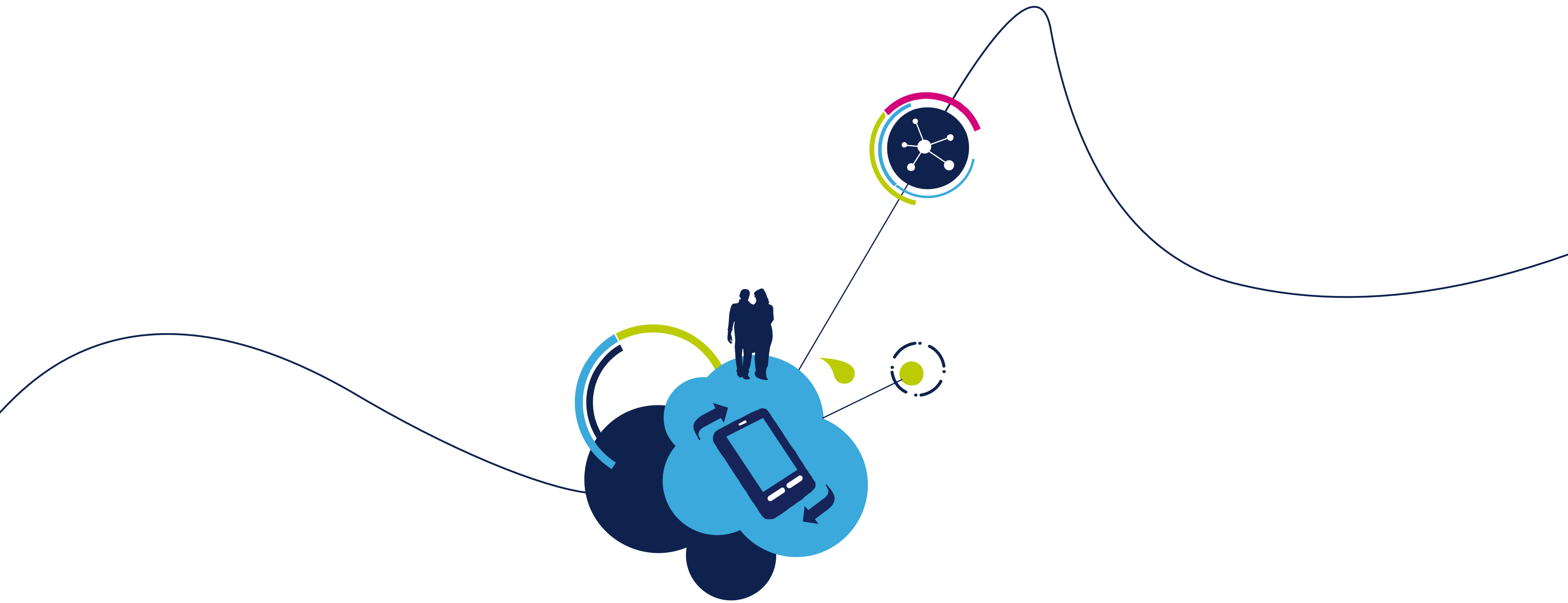
Next steps

- Analyze cumulative delay through software stack
  - Via Android TraceView (up to Sensor Service, Java layers only)
  - Via Logic analyzer (can analyze from I2C access up to Java application)

- Analyze benchmark statistics to check if/why delays are temporarily clustered

- If yes, check with Lauterbach delay bursts responsible

ST Internal

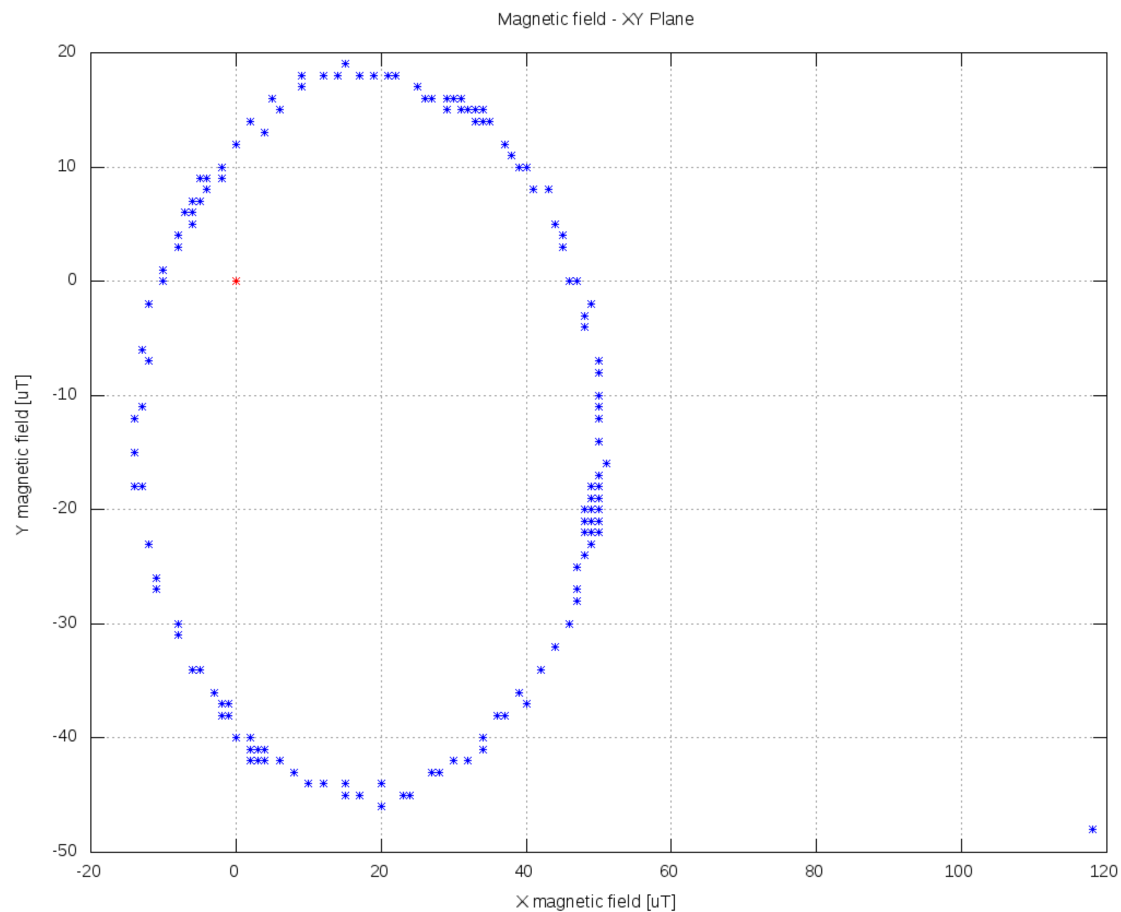


Magnetometer assessment

ST Internal

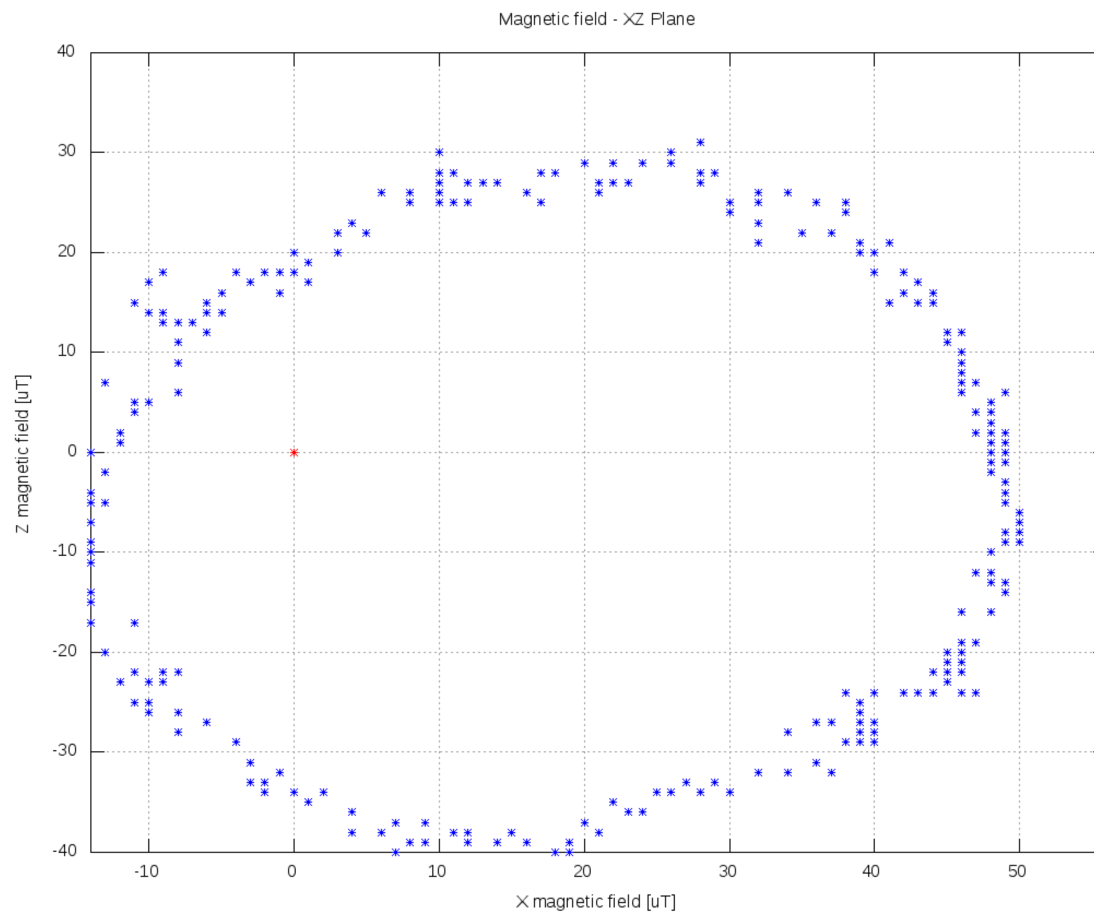
life.augmented

XY Plane



ST Internal

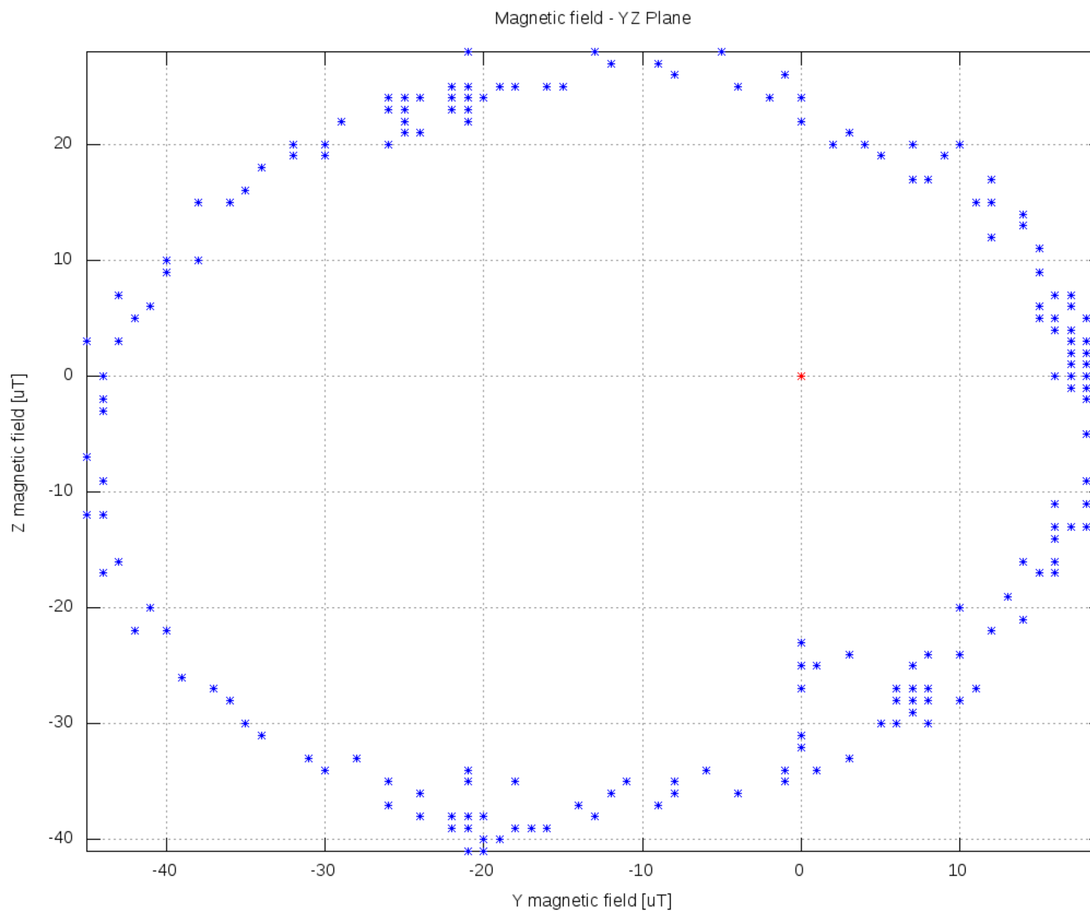
XZ Plane



ST Internal

life.augmented

YZ Plane



ST Internal