

RUTGERS Mining Data from Mobile Devices / Papadimitriou, Eliassi-Rad 7

Accelerometer

Basics

$a_x = a_y = 0$
 $a_z = g$

$a_y = 0$
 $a_z = -a_x = g/\sqrt{2}$

- Measures force exerted on device (vector)
- Stationary device, lying flat:
 - Force preventing it from falling (opposite to gravity)
 - Zero force → Free fall
- Stationary device, after 45° rotation:
 - Same magnitude, but rotated

RUTGERS Mining Data from Mobile Devices / Papadimitriou, Eliassi-Rad 8

Accelerometer

Example code (1/3)

Android (1/2)

```

public class SensorActivity extends Activity
    implements SensorEventListener {
    private SensorManager mSensorManager;
    private Sensor mAccelerometer;

    @Override
    public final onCreate(Bundle savedInstanceState) {
        mSensorManager = \
            (SensorManager) getSystemService(Context.SENSOR_SERVICE);
        mAccelerometer = \
            mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
        mSensorManager.registerListener(\
            this, mAccelerometer, \
            SensorManager.SENSOR_DELAY_NORMAL);
    }
    
```

[continued]

RUTGERS Mining Data from Mobile Devices / Papadimitriou, Eliassi-Rad 9

Accelerometer

Example code (2/3)

Android (2/2)

[continued]

```

@Override
public final void onSensorChanged(SensorEvent event) {
    float ax = event.values[0];
    ay = event.values[1];
    az = event.values[2];
    ...
}

@Override
public final void onAccuracyChanged(Sensor sensor, int accuracy) {
    ...
}
} // SensorActivity
    
```

RUTGERS Mining Data from Mobile Devices / Papadimitriou, Eliassi-Rad 10

Accelerometer

Example code (3/3)

iPhone

```

[[UIAccelerometer mAccelerometer] setDelegate :self;
...
- (void) accelerometer:(UIAccelerometer *) accelerometer,
  didAccelerate:(UIAcceleration *) acceleration {
    float gx = acceleration.x;
    float gy = acceleration.y;
    ...
}
    
```

RUTGERS Mining Data from Mobile Devices / Papadimitriou, Eliassi-Rad 11

Accelerometer

Shortcomings

- Cannot distinguish between gravity and acceleration
 - Impossible: "equivalence principle"
 - Solution: use low-pass filter to estimate gravity
- What if device simultaneously rotates & linearly accelerates?
 - Confused; need more data → gyroscope

RUTGERS Mining Data from Mobile Devices / Papadimitriou, Eliassi-Rad 12

Gyroscope

Introduction

- Measures angular speed
 - Degrees per second (dps)
- Example: L3G4200D (iPhone)
 - ~\$6.5 (DigiKey, bulk)
 - 18mW on / 4.5mW sleep (0.02mW off): 12.5 days / 50 days
 - ±250dps / ±500dps / ±2000dps range
 - 16-bit dynamic range
 - 100 / 200 / 400 / 800Hz sample rate
 - Temperature sensor (8-bit range)
 - Simple interrupt generator & FIFO

STMicro L3G4200D

http://www.st.com/jsp/tech_resource.jsp?techResourceType=1014

RUTGERS Mining Data from Mobile Devices / Papadimitriou, Eliassi-Rad 13

Gyroscope Basics

$\omega_x = \omega_y = \omega_z = 0$
 in 1 sec
 $\omega_x = \omega_z = 0$
 $\omega_x = 45^\circ/\text{sec}$

- Measures angular speed of rotation
 - Represented by numbers for each axis (but: rotation axis is different)
 - Right-hand rule
- Integrate to obtain orientation
 - ...with care, since non-collinear rotations are not commutative

RUTGERS Mining Data from Mobile Devices / Papadimitriou, Eliassi-Rad 14

Magnetometer (compass) Introduction

- Measures direction and magnitude of (Earth's) magnetic field
- Example: AK8973/5 (iPhone)
 - <\$1 (Wikipedia), ~\$2 (AliExpress; non-bulk)
 - 20mW sensor on / 3mW @10Hz: 11 days / 76 days

<http://www.micropower.com/2011/02/10/accelerating-the-iphone-4-electronic-compass.html>

RUTGERS Mining Data from Mobile Devices / Papadimitriou, Eliassi-Rad 15

Inertial measurement & navigation

- 9 DoF (degrees of freedom) available
 - 3-axis accelerometer
 - 3-axis gyro
 - 3-axis magnetometer
- Combine in software for accurate:
 - Position, velocity, and acceleration (linear and angular)
 - Dead reckoning

RUTGERS Mining Data from Mobile Devices / Papadimitriou, Eliassi-Rad 16

Inertial measurement Gravity estimation

Low-pass filter on acceleration data; e.g., on Android Gingerbread:

$$g_t = \lambda_0(a_t + a_{t-2}) + \lambda_1 a_{t-1} - \kappa_1 g_{t-1} - \kappa_2 g_{t-2}$$

Basic operation, also used for:

- Linear acceleration estimate $a_t - g_t$
- Rotation vector (orientation wrt. magnetic north)

RUTGERS Mining Data from Mobile Devices / Papadimitriou, Eliassi-Rad 17

Inertial navigation Dead reckoning

- Integrate gyro to obtain orientation
- Use accelerometer and gyro (orientation) data to estimate linear acceleration
- Doubly integrate acceleration to obtain position change

- Errors accumulate over time ($\sim t^3$)
- Error depends on sampling rate
- How accurate is it?
- "Pro" (air navigation) answer:
 - GPS: better than 9m
 - Inertial: ~650m after one hour

RUTGERS Mining Data from Mobile Devices / Papadimitriou, Eliassi-Rad 18

Other sensors

- Proximity
- Less common:
 - Thermometer
 - But: gyroscope & compass often has temperature output as well
 - Light
 - Pressure (barometric) → altitude

RUTGERS Mining Data from Mobile Devices / Papadimitriou, Eliassi-Rad 19

Location APIs

- Low-level location providers:
 - GPS
 - WiFi
 - Cell tower
 - ...
- Mid-level:
 - Fused location providers
- Higher-level:
 - Geo-fencing
 - Activity recognition

RUTGERS Mining Data from Mobile Devices / Papadimitriou, Eliassi-Rad 20

Location APIs

Location providers (low-level)

- GPS, WiFi, cell tower, ...
- Differ in:
 - Accuracy
 - Availability / freshness
 - Power consumption
- Listen for location updates
- Choose how to update location estimate

RUTGERS Mining Data from Mobile Devices / Papadimitriou, Eliassi-Rad 21

Location APIs

Fused location services

- Combine different location providers
- User specifies:
 - Min and max update period
 - Accuracy preferences
- Location service takes care of managing different low-level providers, to obtain best accuracy at low(est) power

RUTGERS Mining Data from Mobile Devices / Papadimitriou, Eliassi-Rad 22

Location APIs

Geo-fences and activities

Geo-fencing:

- User specifies a POI or map area
- Requests to receive alerts when user is near / inside fence

Activity recognition (Android APIs):

- Use sensor and location data to detect what user is doing
- Walking vs cycling vs driving
- Provides probability for each activity

RUTGERS Mining Data from Mobile Devices / Papadimitriou, Eliassi-Rad 23

Indoor localization

- No O/S level APIs (?)
- Google Maps offers indoors navigation
 - Mix of WiFi-based localization and (very rough) dead reckoning
- May be possible to obtain WiFi RSS data
 - Android offers APIs, iOS is restricted
 - Also: iBeacon (BLE-based, proximity)
- Other applications have used other signals (like audio)
- Custom solutions also exist (e.g., ultrasound-based)
- More on this later ...

RUTGERS Mining Data from Mobile Devices / Papadimitriou, Eliassi-Rad 24

Overview

- Sensors
- **Network connectivity**
- Power
- Accounts ("identity")
- Mobile app basics

RUTGERS Mining Data from Mobile Devices / Papadimitriou, Eliassi-Rad 25

Cellular

- Various standards have evolved over the years
- Hard to track (too much marketing hype...)
- But: rapidly increasing bandwidth and decreasing cost

Standard	Year	D/L (max)	U/L (max)
GPRS	1997	60-80kb/s	20-40kb/s
EDGE	2003	177-237kb/s	60-118kb/s
HSPA	2006	14-42mb/s	1(?) - 6mb/s
HSPA+	2008	28-168mb/s	11-22mb/s
LTE	2010	12-300mb/s	5-75mb/s

*numbers based on marketing claims, please take with grains(s) of salt
**approx. year (first major deployment)

RUTGERS Mining Data from Mobile Devices / Papadimitriou, Eliassi-Rad 26

Cellular

- Historical technologies:
 - 2G: GPRS (GSM)
 - 2.5G: EDGE
- Current technologies:
 - "3G": UMTS (HSDPA, HSUPA, HS[DP]A+)
 - "4G": LTE
- Other technologies (failed adoption):
 - WiBro, WiMax, ...

RUTGERS Mining Data from Mobile Devices / Papadimitriou, Eliassi-Rad 27

802.11b/g/n (WiFi)

- Introduced around the same time as GPRS
- Evolved over time: bandwidth and ubiquity

Protocol	Year	B/W s max (Mbit/s max)
—	1997	2
b	1999	11
g	2003	54
n	2009	72.2 (2.4GHz) 150 (5GHz)
ac (draft)	2012	88-867

RUTGERS Mining Data from Mobile Devices / Papadimitriou, Eliassi-Rad 28

Bluetooth

- Developed by Ericsson in 1994
- Standardized in 1998
- Developed over years

Class	mW (max)	Range (m)
1	100	100
2	2.5	10
3	1	1

- Designed almost concurrently with WiFi; designed for short-range communications with peripherals (not Ethernet/IP packets only)
 - Fairly complex
 - Fairly ubiquitous

Version	Data rate (Mbit/s)
1.2	1
2.0 + EDR	3
3.0 + HS	24

RUTGERS Mining Data from Mobile Devices / Papadimitriou, Eliassi-Rad 29

Bluetooth Low Energy (BLE)

Spec	Bluetooth "Classic"	BLE
Range (max)	100m	50m
Data rate	1-3Mbit/s	1Mbit/s
App. throughput	0.7-2.1Mbit/s	0.27Mbit/s
Latency	100ms (typ.)	6ms
Time to send data	100ms	~3ms
Peak current	<30 mA	<15 mA
Power consumption	100% (reference)	10-50% (use case dep.)

- Entirely separate stack (Zigbee derivative)
- Goals: low power, low latency, low(er) cost
- Initially developed by Nokia, became standard in 2010
- Standard on iPhone, not yet on Android

RUTGERS Mining Data from Mobile Devices / Papadimitriou, Eliassi-Rad 30

Other

- Zigbee / XBee
- Cheap transceivers (e.g., Nordic chipsets)
- Non-standard (on phones), require ugly dongles, etc.
- But, might be worth it for prototyping

RUTGERS Mining Data from Mobile Devices / Papadimitriou, Eliassi-Rad 31

Overview

- Sensors
- Network connectivity
- **Power**
- Accounts (“identity”)
- Mobile app basics

RUTGERS Mining Data from Mobile Devices / Papadimitriou, Eliassi-Rad 32

Power overview

//device/samsung/crespo/+android-4.1.2_r2.1/overlay/frameworks/base/core/res/xml/power_profile.xml

RUTGERS Mining Data from Mobile Devices / Papadimitriou, Eliassi-Rad 33

Cellular and WiFi power

- Overall comparable power draw
- WiFi can consume substantially less (esp. if kept connected)
- But cellular is always available/on
- One larger transfer is much better than many small ones

RUTGERS Mining Data from Mobile Devices / Papadimitriou, Eliassi-Rad 34

Bluetooth power

- Comparable to WiFi
- Bluetooth 4.0 (BLE):
 - Up to 10x lower power draw
 - Lower latency & cost
 - Designed for peripherals / sensors
 - iPhone: standard
 - Android: not yet

RUTGERS Mining Data from Mobile Devices / Papadimitriou, Eliassi-Rad 35

Sensor power

- Not substantial, per se
 - 3-20 mW → several days of power from iPhone 5 battery
 - What's the big deal?
- CPU power consumption!
- For reasonable accuracy: 200Hz sample rate → prevents CPU from entering sleep mode (more soon...)
- Solution: dedicated processor; either
 - Separate app.-specific processor chip
 - All-in-one IMU chip (e.g., MPU6050)

RUTGERS Mining Data from Mobile Devices / Papadimitriou, Eliassi-Rad 36

Frequency scaling

CPU power

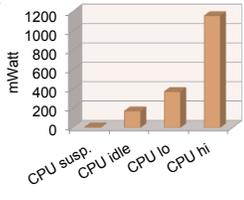
- All modern processors can adjust speed based on workload
- DVFS (dynamic voltage frequency scaling)
- Several policies; defaults are usually fine
- Power consumption is proportional to clock speed (plus a fixed penalty – this is important)

RUTGERS Mining Data from Mobile Devices / Papadimitriou, Eliassi-Rad 37

Advanced sleep modes

CPU power

- If doing *no* work, it's much better to turn off CPU completely
 - Even for a few milliseconds (better than nothing)
 - Around 30x less power draw
- All modern phones will do this automatically
 - Additionally, facilities to reduce number of wakeups; e.g., batching timer events, background messaging (aka. push notifications), etc.

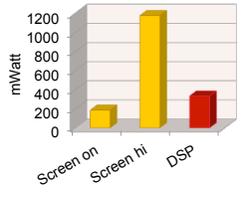


State	Power (mWatt)
CPU susp.	~50
CPU idle	~150
CPU io	~400
CPU hi	~1100

RUTGERS Mining Data from Mobile Devices / Papadimitriou, Eliassi-Rad 38

Display power

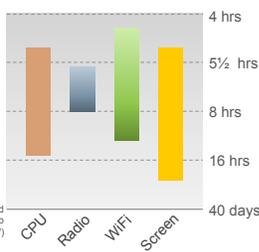
- Substantial power draw, esp. at high brightness
- Not really relevant for sensing applications (unless user interaction is required?)



State	Power (mWatt)
Screen on	~250
Screen hi	~1100
DSP	~400

RUTGERS Mining Data from Mobile Devices / Papadimitriou, Eliassi-Rad 39

Power consumption summary



- Primary power consumers:
 - CPU
 - Radios
 - Display

Component	Time
CPU	~16 hrs
Radio	~5 1/2 hrs
WiFi	~8 hrs
Screen	~16 hrs

RUTGERS Mining Data from Mobile Devices / Papadimitriou, Eliassi-Rad 40

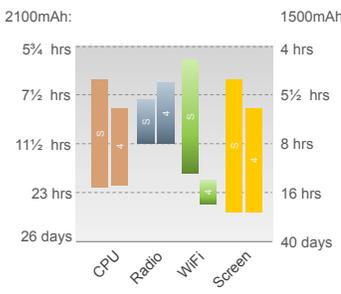
Power overview

Nexus S (Dec 2010) vs Nexus 4 (Nov 2012)

Phones released two years apart: mostly similar

- (Except WiFi, not sure what's going on there...)
- Battery capacity up.. a bit

Don't assume power draw will magically go down; need to actively manage it in your design and code!



Component	Time (hrs/days)	Battery Capacity (mAh)
CPU	~16 hrs	~2100
Radio	~5 1/2 hrs	~1500
WiFi	~8 hrs	~1500
Screen	~16 hrs	~1500

RUTGERS Mining Data from Mobile Devices / Papadimitriou, Eliassi-Rad 41

Overview

- Sensors
- Network connectivity
- Power
- Accounts ("identity")**
- Mobile app basics

RUTGERS Mining Data from Mobile Devices / Papadimitriou, Eliassi-Rad 42

Account manager APIs

- Centralized facility for
 - Managing accounts
 - Managing authentication tokens
- Works in conjunction with other APIs to provide access to content

Sync adapters & content providers

- Sync adapter:
 - Centralized facility for background syncing of on-device and remote data (e.g., contacts, posts, etc)
 - Allows optimizations (e.g., to conserve power)
- Content provider:
 - APIs to expose (synced) data to other applications
 - Well-defined query endpoints and schemas
 - Cursor-like API

Overview

- Sensors
- Network connectivity
- Power
- **Mobile app basics**

Programming paradigm

Heavily event-oriented !

Application must respond to its environment; e.g.

- Network connectivity changes
- Incoming calls / messages / events
- Sensor / location data
- ...

Application must use resources efficiently; e.g.

- May be pre-empted and/or killed at any time
- May choose to respond to status information (e.g, battery level)
- ...

Cannot:

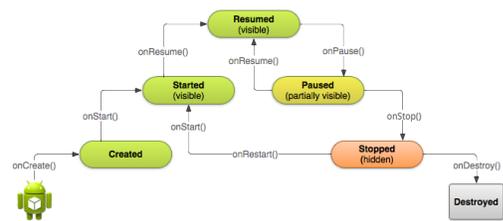
- Assume a single main() thread with sequential flow
- Control lifetime of thread(s)

Programming paradigm

Android: *activity* paradigm

- You can think of an activity as a screen
- Can be in different *states* during it's *lifecycle*
- Need to respond to state-change events
- System determines state based on:
 - User interactions (e.g., start a different activity)
 - External events (e.g., screen rotation, incoming call, ...)
 - Available resources (memory, CPU, etc)
- Execute in the main app thread
- Responsible for persisting any app-specific state, as necessary

Activity lifecycle (Android)



A note on “deep linking”

- Android: an app can declare which activities handle a particular URL pattern
 - E.g., YouTube app could declare that URLs with http:// protocol and youtube.com domain can be handled by it's video player activity
 - If multiple apps can handle the same URL type, the user will be prompted to choose
- iOS: similar, but not as general/broad (?)

RUTGERS Mining Data from Mobile Devices / Papadimitriou, Eliassi-Rad 49

Background tasks

Short lived (e.g., fetch a URL):

- Can be started in separate threads
- But: need to be prepared for activity termination/restart

Long lived:

- Need to use system APIs to register themselves and allow system to manage them
- Timers, background services, RPC interfaces

Avoid whenever possible!!

- Use system services instead, e.g., geo-fences, push notifications, etc

RUTGERS THE STATE UNIVERSITY OF NEW JERSEY Mining Data from Mobile Devices / Papadimitriou, Eliassi-Rad 50

MINING DATA FROM MOBILE DEVICES

Mobile technology overview

Spiros Papadimitriou, Tina Eliassi-Rad



RUTGERS Mining Data from Mobile Devices / Papadimitriou, Eliassi-Rad 51

License



These slides are made available under a Creative Commons Attribution-ShareAlike license (CC BY-SA 3.0):

<http://creativecommons.org/licenses/by-sa/3.0/>

You can share and remix this work, provided that you keep the attribution to the original authors intact, and that, if you alter, transform, or build upon this work, you may distribute the resulting work only under the same or similar license to this one.

© 2013, 2015 Spiros Papadimitriou, Tina Eliassi-Rad