Lecture 17: Interval Timers & Watchdog Timers

Engineering Design with Embedded Systems

Patrick Lam
University of Waterloo

February 8, 2013

This work is licensed under the *Creative Commons Attribution-NonCommercial-ShareAlike 3.0 License*.

Last Time: Android Friday

1. Toasts;
2. Broadcast Receivers; and
3. Lists.

Concept: Timers in Embedded Systems

Lecture 6: hands-on timer implementation in Android (`Runnable`, `Handler`).

Today: How to use timers,
e.g. in embedded systems:

- Interval Timers
- Watchdog Timers

We'll use Java timers to implement these, and learn the difference between Android and Java timers.

Applications of Timers

Interval Timers—perform a task occasionally.

- (raising alarms; polling).

Watchdog Timers—reset a stuck system.

Assignment 2: Android timer practice.

Part I

Interval Timers

Interval Timers

Good for recurring tasks.

Example: Light sensor polling
(but not on Android—receive sensor events instead).

Previously: `Handler` for interval timers.

Handlers for Infinitely-Recurring Interval Timers

```
Handler h = new Handler();
Runnable r = new Runnable() {
  public void run() {
    // execute the task
    h.postDelayed(this, delayInMS);
  }
};
h.postDelayed(r, delayInMS);
```

Cancelling Tasks

By the way, you can cancel an upcoming task like this:

```
h.removeCallbacks(r);
```

True Interval Timers

Handlers run when the thread is available.

True interval timers interrupt the processor.
Can simulate with `java.util.Timer`[1].

Payload runs in a different thread:
(+) more predictable timing;
(–) can't update UI from other thread.

Generally more overhead for `java.util.Timer`,
not great for Android apps.

[1] `http://steve.odyfamily.com/?p=12`, accessed February 3, 2013.

Getting Around UI Thread Issues

Start the Timer like this:

```
Timer t = new Timer();
t.schedule(new TimerTask() {
  @Override
  public void run() {
    runOnUiThread(timerTick);
  }
}, firstDelayMS, repeatIntervalMS);
```

If you omit repeatIntervalMS, you get a one-shot timer.
Don’t call `timerTick.run()` directly.

Programming timerTick

Put your UI-manipulating code in `timerTick`:

```
Runnable timerTick = new Runnable() {
  @Override
  public void run() {
    Toast.makeText(getApplicationContext(),
                   "ding!",
                   Toast.LENGTH_SHORT).show();
  }
};
```

Java Timer summary

To use a Java Timer:

1. instantiate a new `Timer` object;
2. schedule `TimerTask` on that `Timer`;
3. inside the `TimerTask`, invoke `Runnable` to run on the UI thread; and
4. implement the `run()` method on the `Runnable` to effect UI changes.

Part II

Watchdog Timers

Goal of a Watchdog Timer

Observe that a system appears to be stuck and take some action.

Implementing a Watchdog Timer

Can build a watchdog timer using an interval timer:

- Set the timer interval to be the largest allowable time for a task to take. Start the timer.
- If the timer fires, the task took too long.
- The timer event handler deals with the situation (e.g. by killing the task.)

Resilience for Watchdog Timers

In an embedded system:
specialized hardware.

Timer has some resilience:
runs in a separate thread.

Android Watchdog Timer

Summary:

- start the watchdog timer;
- the timer runs a task *t* on the UI thread;
- task *t* kills the activity; and
- set up a click listener to go to a web page instead.

Outcomes from Watchdog Timer

1. If you click the button soon enough, app goes to the web page;
2. Otherwise, the Activity finishes.