**Services with Alarms**

**Overview**

* A service is required for an alarm to be processed when the system is sleeping.
* The idea is as follows
	+ The activity starts the service. A button triggers the service to set the alarm
	+ The alarm calls the OneShotAlarmReceiver,
	+ which calls a BroadcastReceiver that is part of the service
	+ Since the service is always able to receive broadcasts, it will be triggered even if the system is sleeping
* One complication is that we need to make sure that the system stays awake long enough to call the service.

**Merge FunWithLocalServices and FunWithAlarms**

* Start with a service developed in the FunWithLocalServices tutorial
* Open the FunWithAlarms files for reference
* Merge FunWithLocalServices and FunWithAlarms
	+ Make new class OneShotAlarmReceiver
		- Same as in FunWithAlarms. See FunWithAlarms part 1
		- onReceive with (note new name of action is changed to com.FunWithServices.MyService.TIMESUP
			* Log.d("OneShotAlarmReceiver","Alarm Expired");
			* Intent broadcast=new Intent("com.FunWithServices.MyService.TIMESUP");
			* context.sendBroadcast(broadcast);
	+ In manifest,
		- add entry for receiver OneShotAlarmReceiver (see FunWithAlarms part 1)
	+ In MyService, add broadcast receiver
		- From FunWithAlarms part 1, copy

**public** BroadcastReceiver receiver=**new** BroadcastReceiver() {

 **public** **void** onReceive(Context context, Intent intent) {

 Log.*d*("FunWithAlarms ","Received TimesUp message");

 }

};

* + In MyService, add function to set the alarm and register the BroadcastReceiver
		- public void startAlarm()
		- {
		- registerReceiver(receiver, new IntentFilter("com.FunWithServices.MyService.TIMESUP"));
		- Intent intent = new Intent(this, OneShotAlarmReceiver.class);
		- PendingIntent pendingIntent = PendingIntent.getBroadcast(this, 0, intent, PendingIntent.FLAG\_ONE\_SHOT);
		- AlarmManager alarmManager = (AlarmManager) getSystemService(Context.ALARM\_SERVICE);
		- alarmManager.set(AlarmManager.RTC\_WAKEUP, System.currentTimeMillis() + (5 \* 1000), pendingIntent);
		- Log.d("MyService","Alarm is set");
		- }
	+ In MyService.onDestroy(), add
		- unregisterReceiver(receiver);
	+ In FunWithLocalServices, call startAlarm from button onClickListener
		- Add myService.startAlarm();
* Run
* If “received TimesUp message is not shown, check that the registered receiver in MyService.startAlarm is has the same name as the destination of the intent called in OneShotReceiver.

**Problem with sleeping**

* When an activity or service is running, the system might sleep at any time
* In order to keep the system awake, WakeLocks are used
* So, when the alarm goes off and the service’s broadcast receiver is triggered, it should set a WakeLock.
* The alarm’s BroadcastReceiver does not need a WakeLock, the AlarmManager keeps a WakeLock so the system does not sleep before the BroadcastReceiver.onReceive is complete
* **However**, the system is allowed to sleep as soon as the alarms BroadcastReceiver is finished.
* It might go to sleep before the service is able set its WakeLock
* We need to call a WakeLock from the alarm’s BroadcastReceiver.onReceive, but then release it from the services BroadcastReceiver.
* One way to solve this is to make a class with static member variables and functions.

**MyStaticWakeLock class**

* Make new class call MyStaticWakeLock
	+ public class MyStaticWakeLock {
	+ private static PowerManager.WakeLock wl = null;
	+ public static void lockOn(Context context) {
	+ PowerManager pm = (PowerManager) context.getSystemService(Context.POWER\_SERVICE);
	+ if (wl== null)
	+ wl = pm.newWakeLock(PowerManager.FULL\_WAKE\_LOCK, "MyStaticWakeLock");
	+ wl.acquire();
	+ }
	+ public static void lockOff(Context context) {
	+ PowerManager pm = (PowerManager) context.getSystemService(Context.POWER\_SERVICE);
	+ if (wl != null)
	+ wl.release();
	+ }
	+ }
* Note, this class has two functions, one for setting the WakeLock and one for releasing
* In OneShotAlarmReceiver.onReceive, add
	+ MyStaticWakeLock.lockOn(context);
* At the end of MyService.receiver.onReceive(), add
	+ MyStaticWakeLock.lockOff((MyService.this);
* In manifest, add permission WAKE\_LOCK
* Run