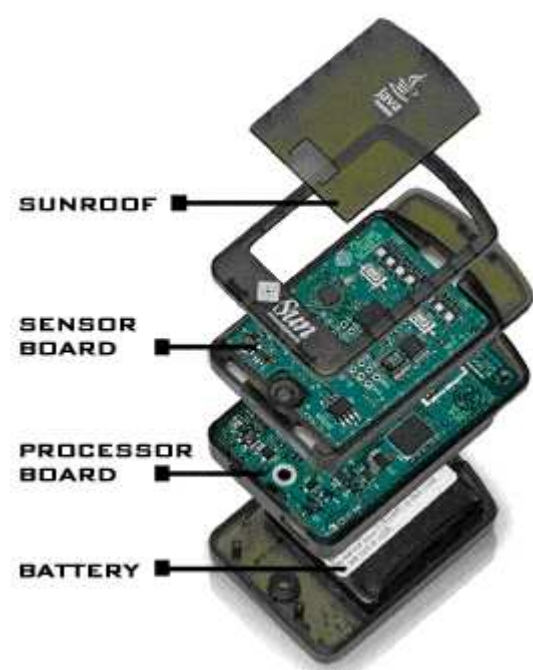




## Sun SPOTs

### Hardware

- 180 MHz 32 bit ARM920T core
- 512K RAM/4M Flash
- 2.4 GHz IEEE 802.15.4 radio
- USB interface
- 2G/6G 3-axis accelerometer
- Temperature sensor
- Light sensor
- 8 tri-color LEDs
- 6 analog inputs
- 2 momentary switches
- 5 general purpose I/O pins and 4 high current output pins



source:  
[www.sunspotworld.com](http://www.sunspotworld.com)

### Software:

- J2ME CLDC 1.1 Java Squawk VM with OS functionality
- VM executes directly out of flash memory
- Automatic battery management

### Benefits:

- New system developers can easily design programs using Java
- Advanced features such as threads available

## Target Applications

### System Structure:

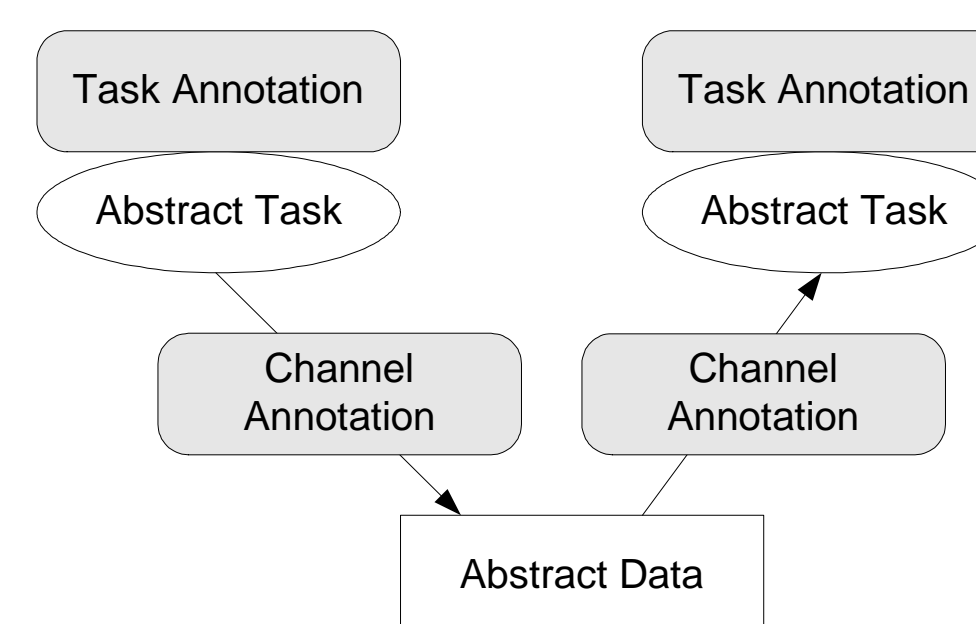
- Nodes are at fixed locations
- Node locations are known at time of deployment
- Deployment can be divided into logical 'regions' (e.g. floor, room, highway sector)
- Nodes may differ in terms of the sensors/actuators attached to them

### Interaction Patterns:

- **Hierarchical Data Collection**
  - Data is collected from all over the system and processed at multiple levels before reaching the sink
  - E.g. Landslide Detection
- **Localized Interactions**
  - Neighboring nodes collaborate to take decisions, which are then communicated to others
  - E.g. Target tracking by leader election
- **Sensing Driven by Actuation**
  - Sensor in a region collect data, which is processed locally to possibly activate actuators
  - E.g. Building Environment Management

## WSN Macroprogramming with ATaG

### The Structure of an ATaG Program



### Abstract Tasks

- "Fire" in response to events
- Firing rules and the placements are determined by Task Annotations

### Abstract Data Items

- The currency of information in ATaG
- Only one "producer" per data item

### Abstract Channels

- Show the relation between Abstract Data Items and Abstract Tasks
- Annotations define the "interest"

### Key Concepts

- **Data-driven control flow over a distributed data store**
  - Task and Channel Annotations specify the task interactions
- **Mixed imperative-declarative program specification**
  - Imperative portion is a traditional (C/Java) sequential program interacting with the data pool
  - Declarative portion is interpreted – at compile time and at run time – in the context of a particular network architecture
  - Imperative and declarative parts can be modified independently
- **Scheduling and communication is managed by a runtime**
  - Programmer does not write networking code
  - System-level optimizations can be performed without requiring rewrite of application-level code

## Application Development using our Toolkit

