TinyOS Programming I

TinyOS

- Embedded operating systems
- Features of all operating systems
  - Abstraction of system resources
  - Managing of system resources
  - Concurrency model
  - Launch applications
- Embedded operating systems
  - Application-specific
  - Small-scale resources

Sensor Resources

- Computation
  - CPU
  - Memory
- Power
- Hardware
  - Timer
    - Only one timer, how to share among different tasks?
  - Sensors
  - Radio communication
  - LEDs
  - Serial port

Abstraction of Resources

- Turn LED on/off
- API
  - Leds.led0On()
  - Leds.led0Off()
  - The functions are platform independent.
- Device
  - Each LED is connected with a pin of the CPU's IO ports
  - On/off means high or low on the pin
  - The wiring of LEDs with CPUs is platform dependent.

Programming Model

- Applications are built out of **components**
- Components specify **interfaces** they use and provide
- Components are statically **wired** to each other via their interfaces
- Building an application in TinyOS is like building a house
  - What components are needed
  - How to connect the components

LED Demo

- Application : toggle an LED every one second
- Components : hierarchical
  - BlinkAppC (top-most software component, made by you)
    - LedsC (hardware component, provided in library)
    - TimerC (hardware component, provided in library)
    - MainC (hardware component, provided in library)
    - BlinkC (software component, made by you)
- Wiring
- How it works

Components

- Components
  - Modules and configurations
  - Provide and use interfaces
- Modules
  - Basic block for implementing functions
  - Do not have sub-components
  - BlinkC
- Configurations
  - A set of wired modules
  - Do not implement any function
  - BlinkAppC, MainC, LedsC, TimerC

Module

```
module aaaC {
  uses {
    interface aaaI;
    interface bbbI;
  }
  provides {
    interface cccI;
    interface dddI;
  }
}

implementation {
  // code : variables, functions, commands and events

  // no sub component
}
```

Configuration

```
configuration aaaC {
  uses {
    interface aaaI;
  }
  provides {
    interface bbbI;
  }
}
implementation {
  components aaaC, bbbC;

  bbbI = aaaC.cccI;
  aaaC.dddI -> bbbC.eeeI;

  // no code
}
```

Interface

- Include a collection of commands and events definitions.
- Do not implement any command or event.
- Only the module providing the interface implements commands in the interface.
- Only the module using the interface implements events in the interface.
- A user module uses the interface provided by a provider module.

Interface

- Comparison
  - C++
    - Components are objects.
    - Functions are defined and implemented in classes.
  - TinyOS
    - Commands and events are defined in interfaces.
    - Commands are implemented in components who provide the interfaces.
    - Events are implemented in components who use the interfaces.
- In our class, we do not create new interfaces.

Interface Definition

```
interface aaaI {
  command error_t bbb();
  event void ccc(error_t err, val_t t);
}
```

Command Implementation

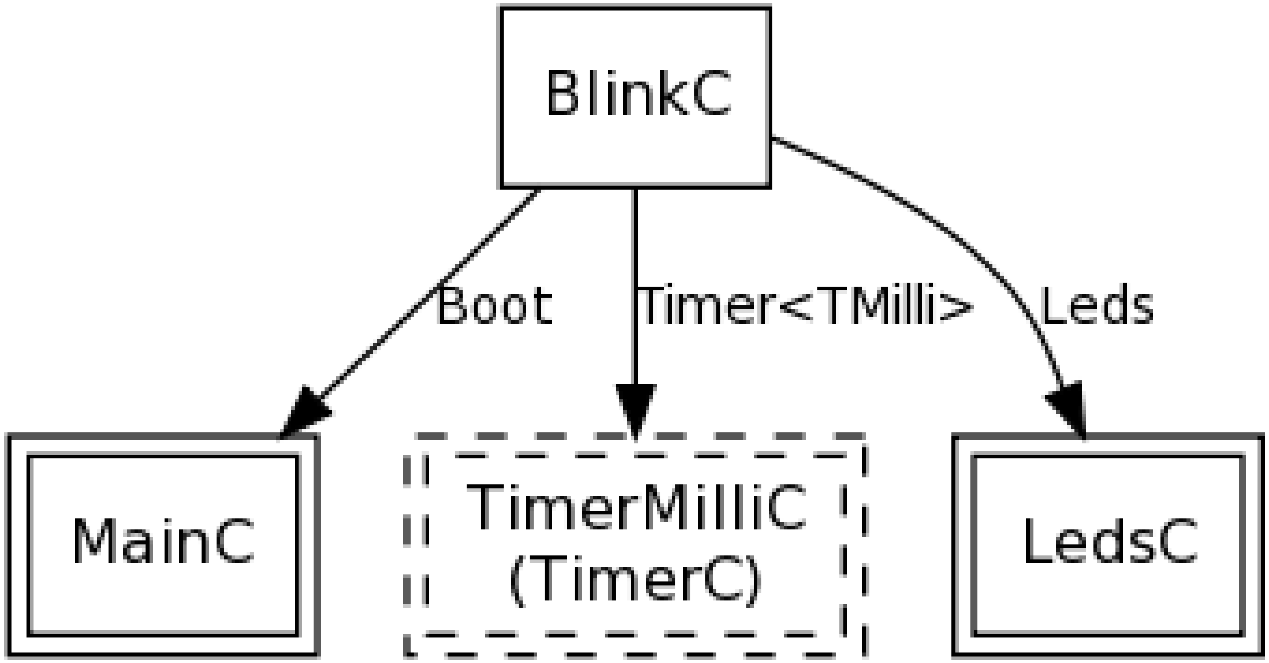
```
module dddC {
  provides {
    interface aaaI;
  }
}

implementation {

  command error_t aaaI.bbb() {
    // code
  }

}
```

Event Implementation

```
module eeeC {
  uses {
    interface aaaI;
  }
}

implementation {

  event void ccc(error_t err, val_t t) {
    // code
  }

}
```

LED Demo

- Take a look at BlinkAppC again



Dim LED Demo

- LEDs are digital, i.e. on/off or 0/1.
- But, human eyes are analogy.
  - Vision stays for about 0.1s.
    - Movie: 20fps - 40fps
  - Lights are averaged for blinking LEDs.