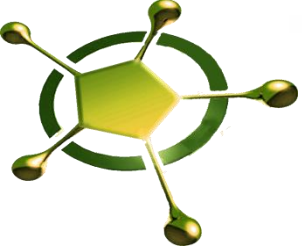


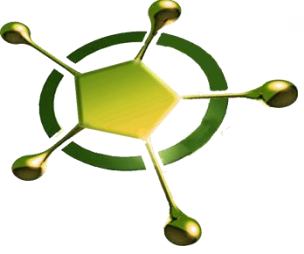
Sharing experience with TinyOS for iris motes



OUTLINE



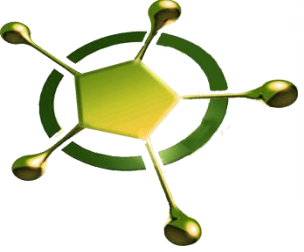
- TinyOS
- Installation and configurations
- NesC
- How Wireless Sensor Network works
- Overview of Crossbow Wireless Sensor equipment
- Description of the MoteWorks platform
- Requirements for the Application
- Building A Simple Sensing Application



TinyOS



- Open Source System By UC, Berkeley
- TinyOS has a component-based programming model
- TinyOS is not an OS in the traditional sense.
- It supports microprocessors ranging from 8bit- architectures 2KB of RAM to 32-bit processors with 32 MB of RAM or more.



Installation and configurations



- Download TinyOS from:

<http://www.tinyos.net/download.html>

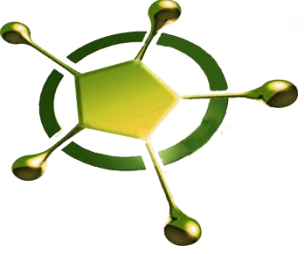
Version release notes available here:

<http://www.tinyos.net/tinyos-1.x/doc/>

The default install puts TinyOS in

C:\tinyos\cygwin\opt\tinyps-1.x

- Or install MoteWorks platform:
 - Low Power Operating System – TinyOS
 - Software Development Tools

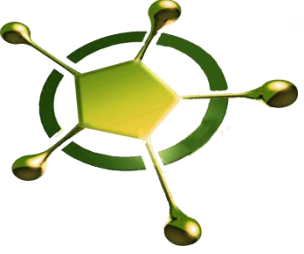


Programming in nesC is like programming in a hardware description language. NesC is component based C dialect.

File types in TinyOS:

1. Module (suffixed with *M.nc*)
2. Configuration (suffixed with *C.nc*)
3. Interface (suffixed with *.nc*)

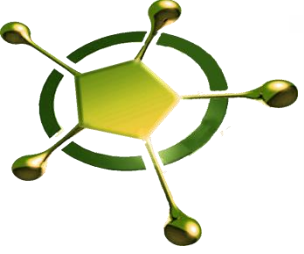
Configuration and a *module* are combined to make a *component*.
Every component has *implementation block* .



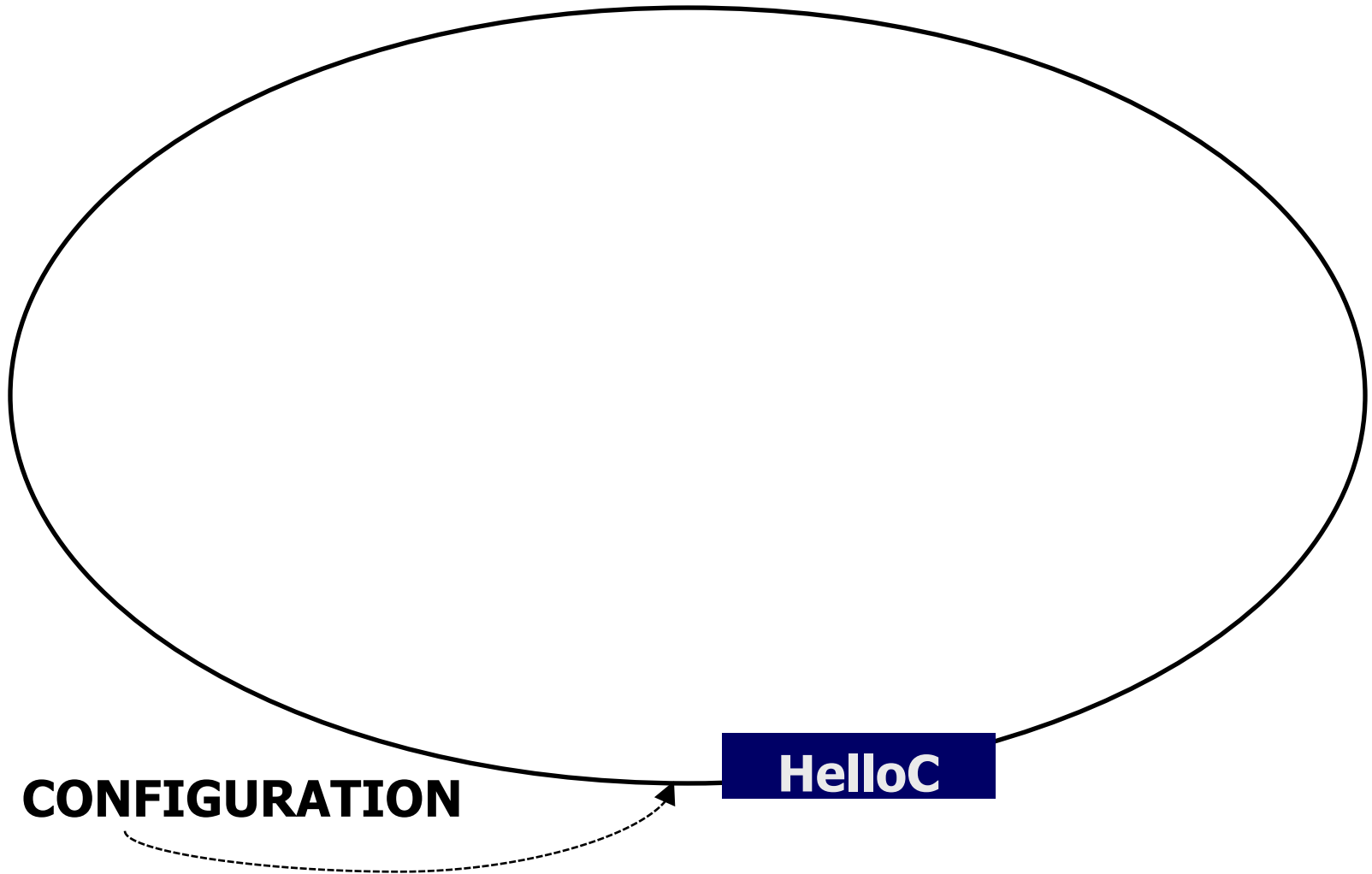
NesC and TinyOS

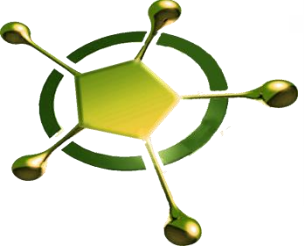


Modules
Configuration
Interfaces



Hello world—component diagram



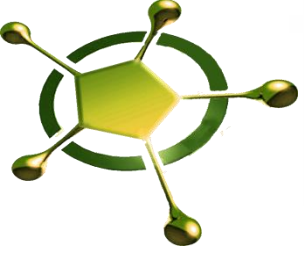


HelloC configuration

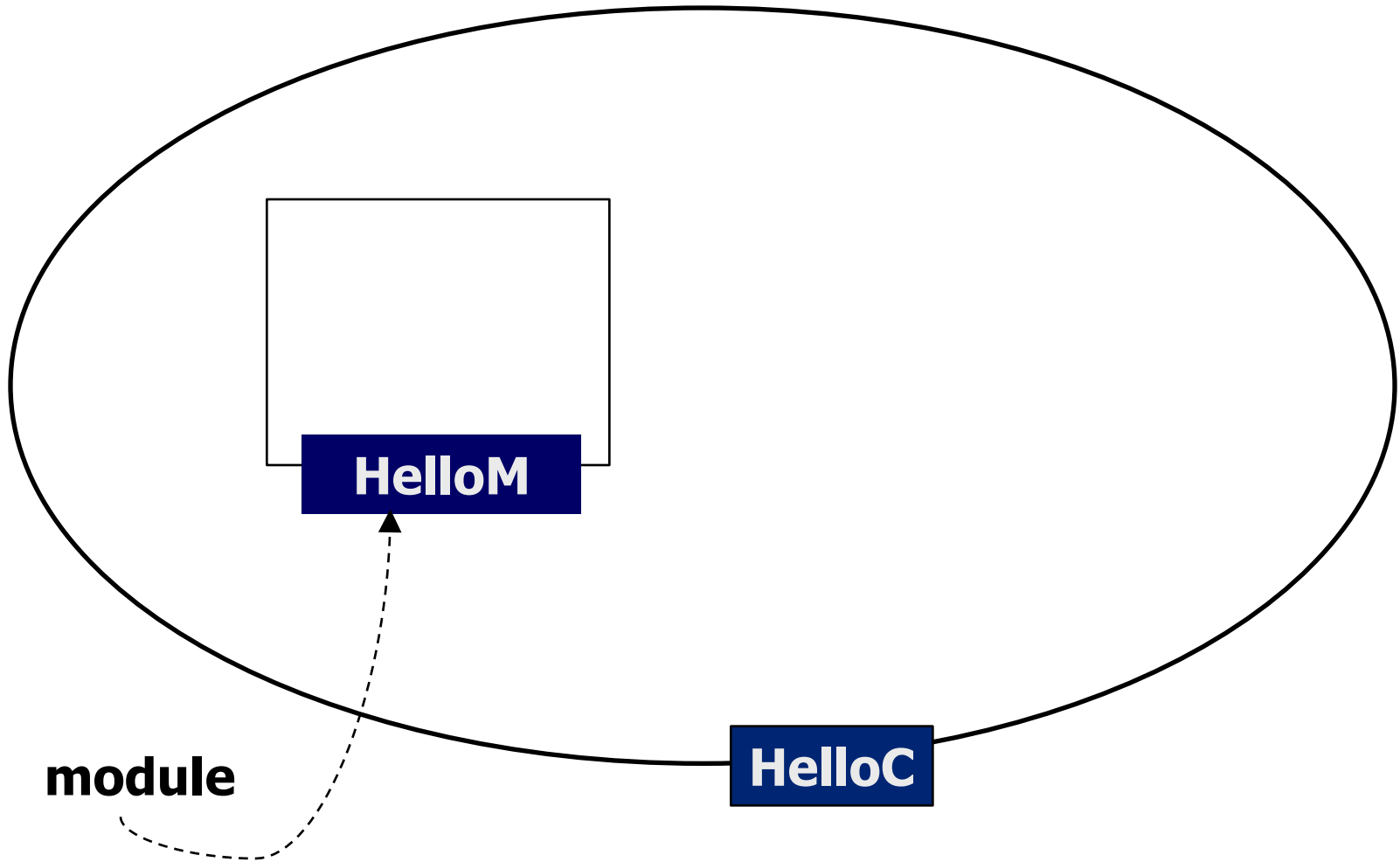


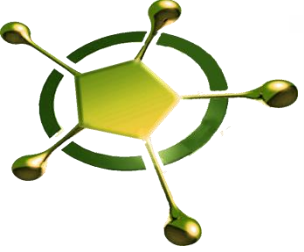
```
configuration HelloC {  
}
```

```
implementation {  
    components HelloM;  
}
```



Component diagram



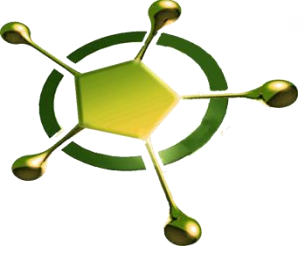


HelloM module

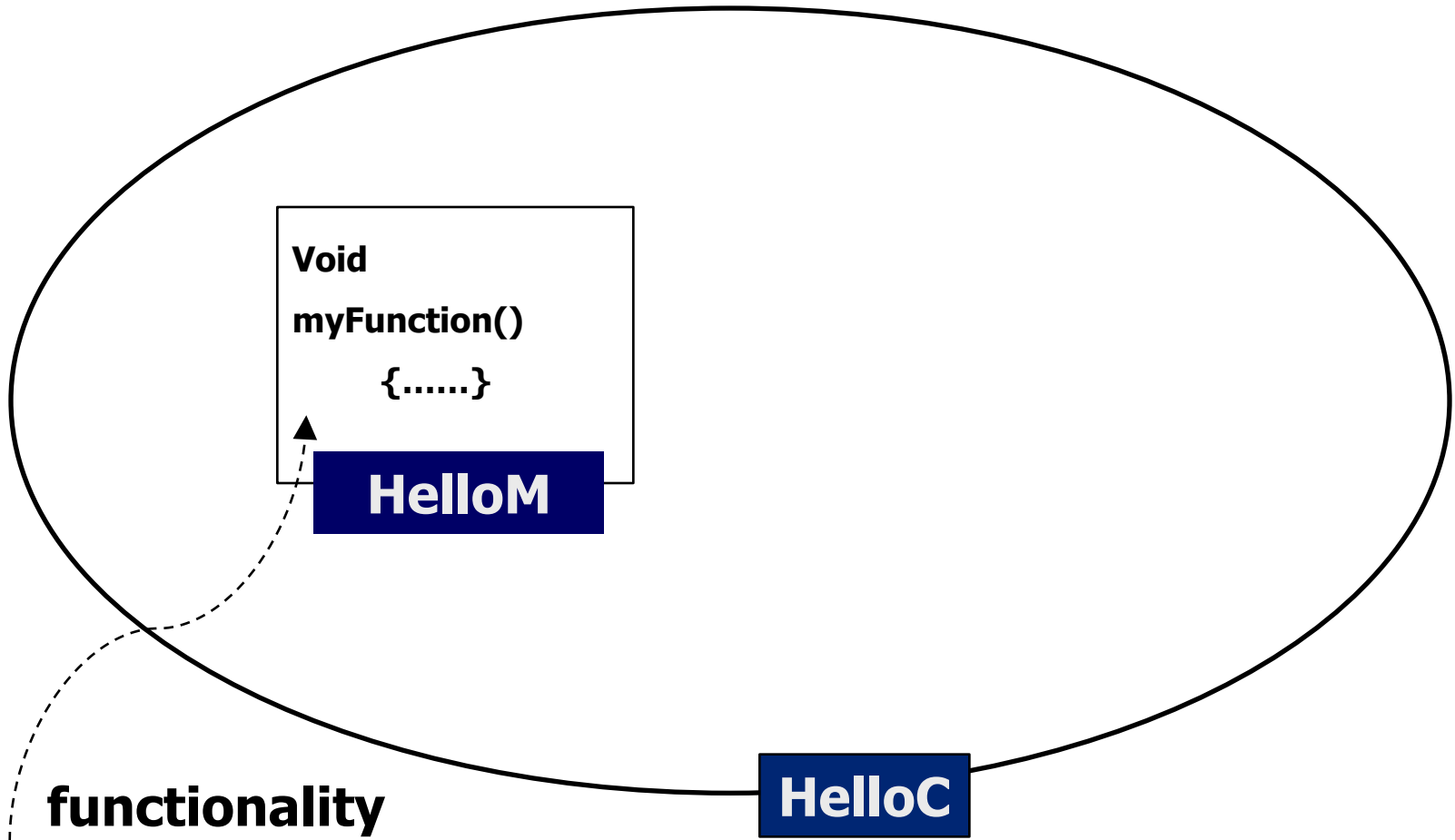


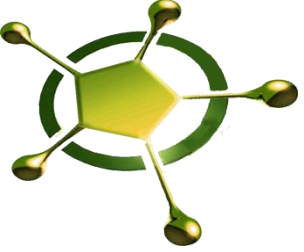
```
module HelloM {  
}
```

```
implementation {  
}
```



Component diagram

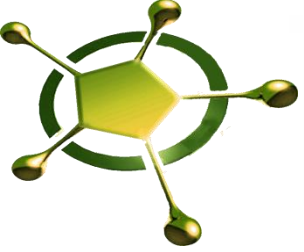




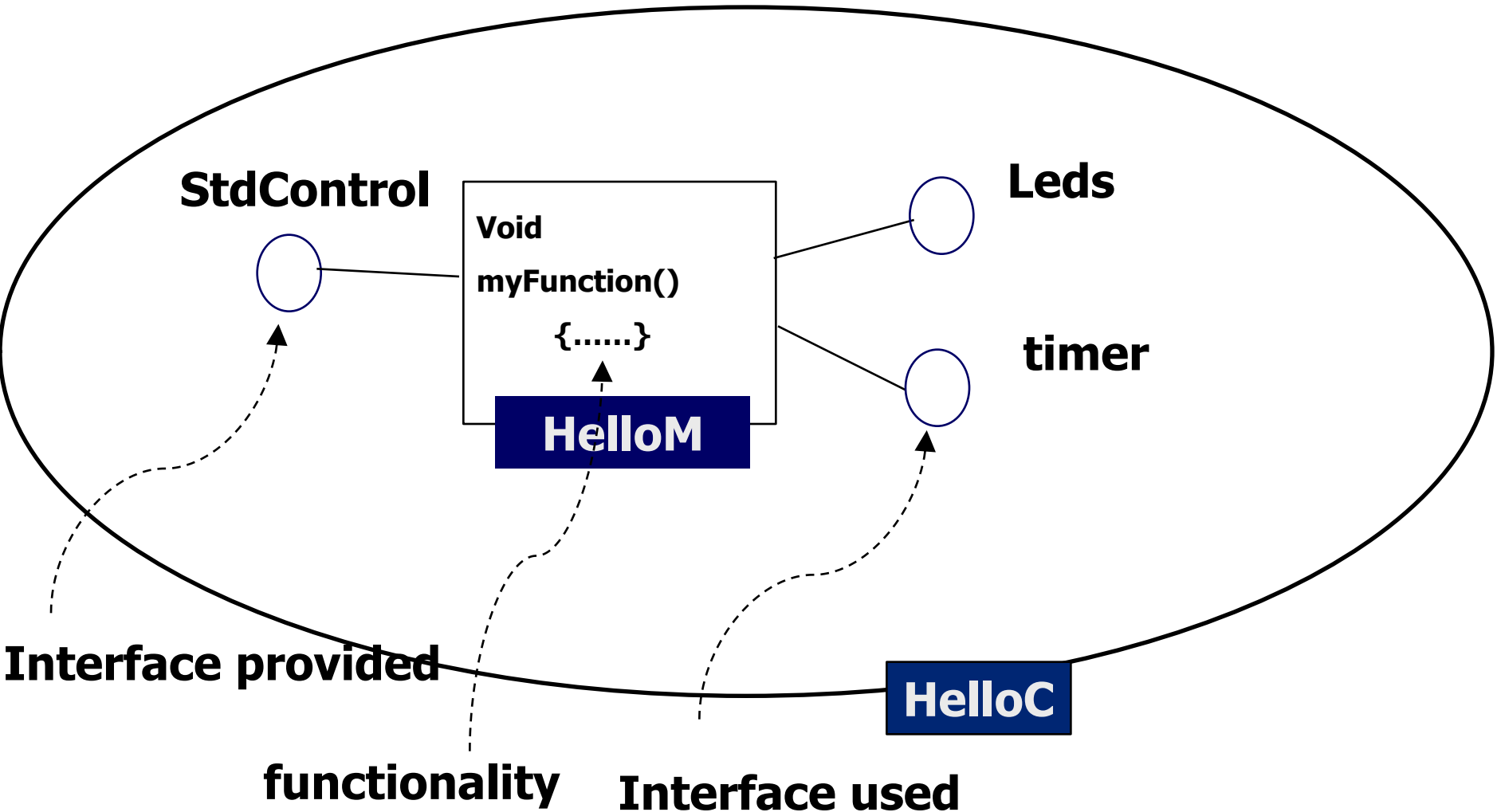
HelloM module

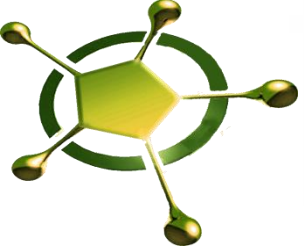


```
module HelloM {  
    provides {  
        interface StdControl;  
    }  
    uses {  
        interface Timer;  
        interface Leds;  
    }  
}  
  
implementation {  
  
    }  
}
```



Component diagram



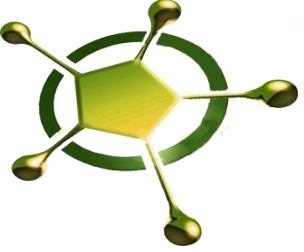


HelloM module



```
module HelloM {
    provides {
        interface StdControl;
    }
    uses {
        interface Timer;
        interface Leds;
    }
}

implementation {
    command result_t StdControl.init() {
        call Leds.init();
        return SUCCESS;
    }
}
```



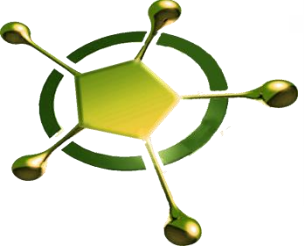
Cont.



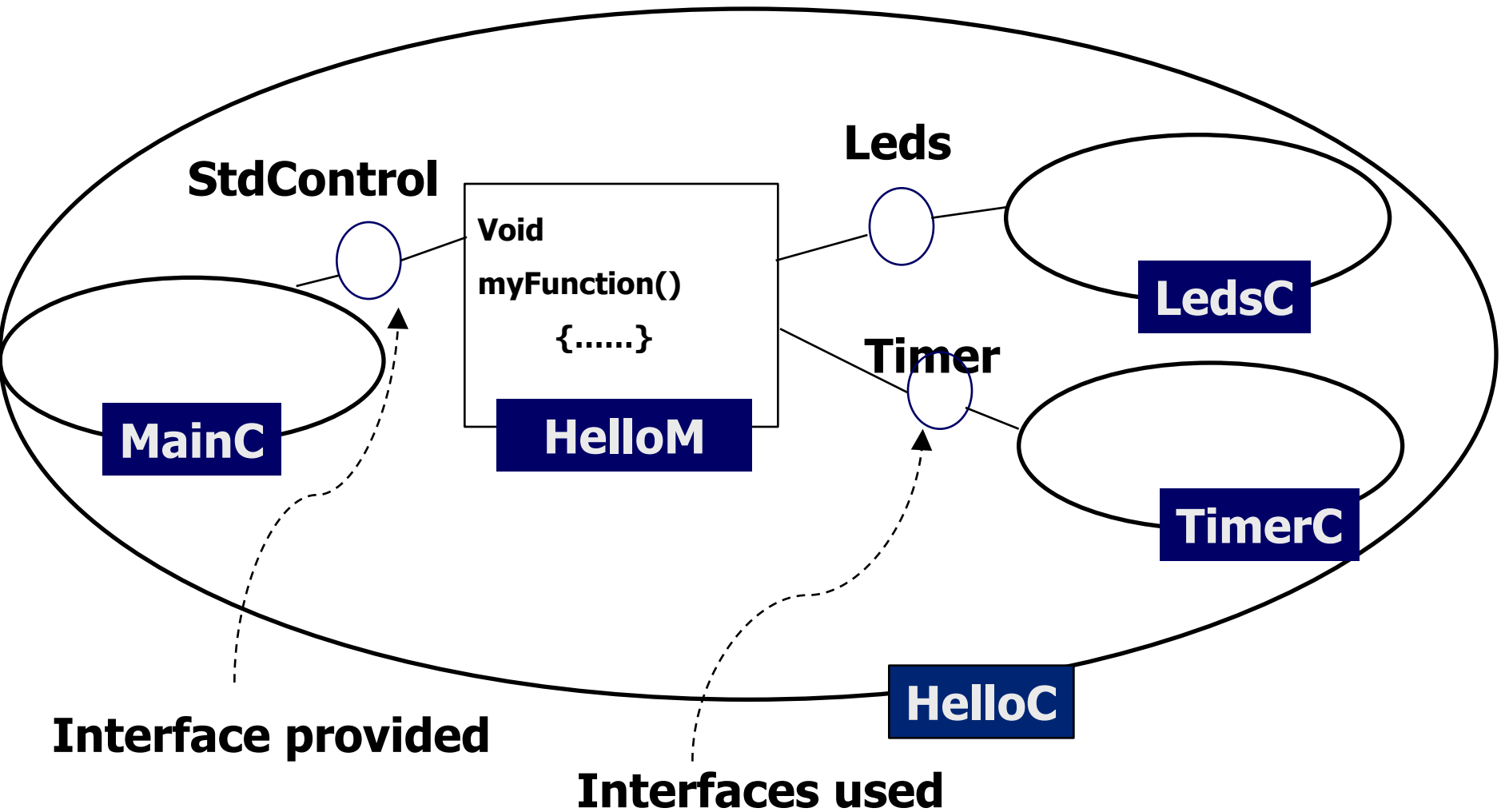
```
command result_t StdControl.start() {
    return call Timer.start(TIMER_REPEAT, 1000);
}

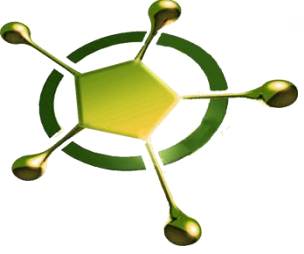
command result_t StdControl.stop() {
    return call Timer.stop();
}

event result_t Timer.fired() {
    call Leds.redToggle();
    return SUCCESS;
}
}
```



Component diagram





Overview of Crossbow Wireless Sensor equipment



- Sensor Data Acquisition Boards
- Processor/Radio Platforms or "Motes"
- Gateways and Network Interfaces

Sensor Data Acquisition Boards



MDA300

MDA320

MDA100

MTS300

MTS310

MTS400

MTS420

Features



Accelerometer (2 Axis)

•

•

•

Actuator Relays

•

Ambient Light

•

•

Barometric Pressure & Temp.

•

•

Buzzer

•

•

External Analog Sensor Inputs

•
(12-bit)

•
(16-bit)

•
(10-bit)

GPS

•

GPIO

•

•

•

Magnetic Field

•

Microphone

•

•

Photo-sensitive Light

•

•

Photoresistor

•

•

•

Rel Humidity & Temperature

•

•

•

Thermistor

•

•

•

Processor/Radio Platforms or "Motes"



XM2110

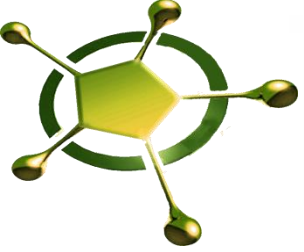
M2110

MPR2400

MPR2600

MPR400

Features					
Frequency Range	2.4GHz ISM Band	2.4GHz ISM Band	2.4GHz ISM Band	2.4GHz ISM Band	868-870; 902-928 MHz
Processor	Atmel ATMega 1281	Atmel ATMega 1281	Atmel ATMega 128L	Atmel ATMega 128L	Atmel ATMega 128L
Radio Transceiver	RF230 Atmel	RF230 Atmel	TI CC2420	TI CC2420	TI CC1000
Serial Flash	Atmel AT45DB41B (512 kB)	Atmel AT45DB41B (512 kB)	Atmel AT45DB41B (512 kB)	Atmel AT45DB41B (512 kB)	Atmel AT45DB41B (512 kB)
RAM	8K bytes	8K bytes	4K bytes	4K bytes	4K bytes



Microprocessor:

Atmel ATmega128 for iris mote. It has 128K of flash memory, 4K of RAM and 4K of EEPROM.

Radio:

Iris radio at 2.4 GHZ

Serial Flash:

External flash for OTAP and data logging

Gateways and Network Interfaces



MIB510

MIB520

MIB600

SPB400



Features

Description

Mote/Board Connectors

Programming Port

Data Port

Serial Port Programmer

USB Programmer

Ethernet Port Programmer

Stargate, XScale Platform

IRIS, MICAz, MICA2
MICA-sensor Boards

IRIS, MICAz, MICA2

IRIS, MICAz, MICA2

IRIS, MICAz, MICA2
PCMCIA, USB,
RS-232, Compact Flash, RS-232,
Ethernet

Serial (RS-232)

USB

Ethernet

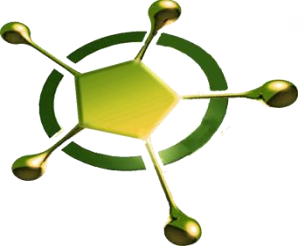
With host PC: RS-232 or
Ethernet (using ssh)

Serial (RS-232)

USB

Ethernet

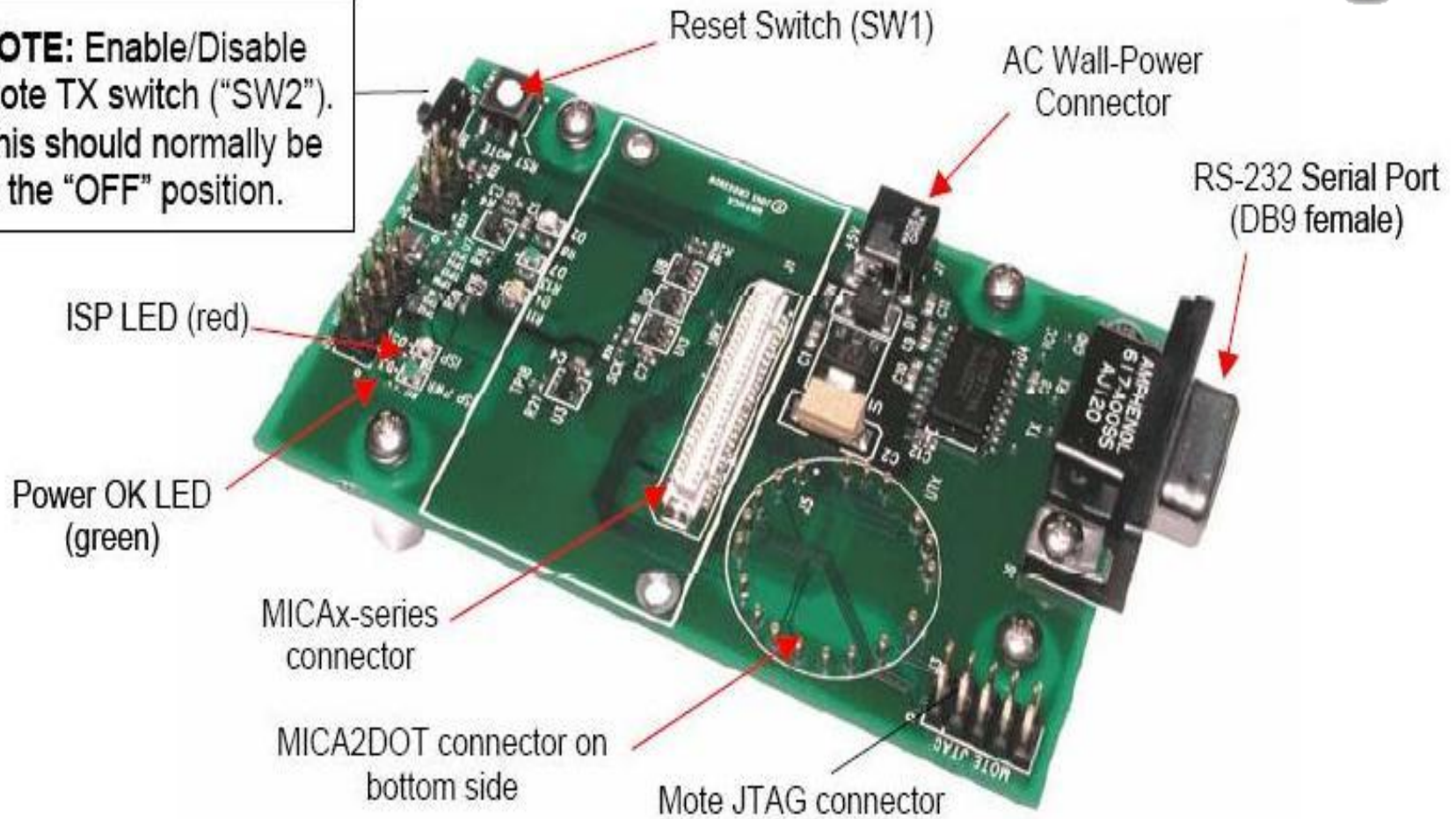
Various



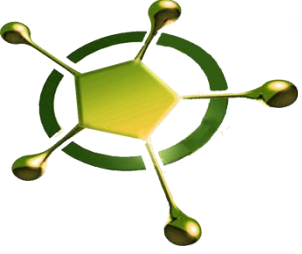
MIB510



◀ **NOTE:** Enable/Disable Mote TX switch ("SW2"). This should normally be in the "OFF" position.



Top view of MIB510



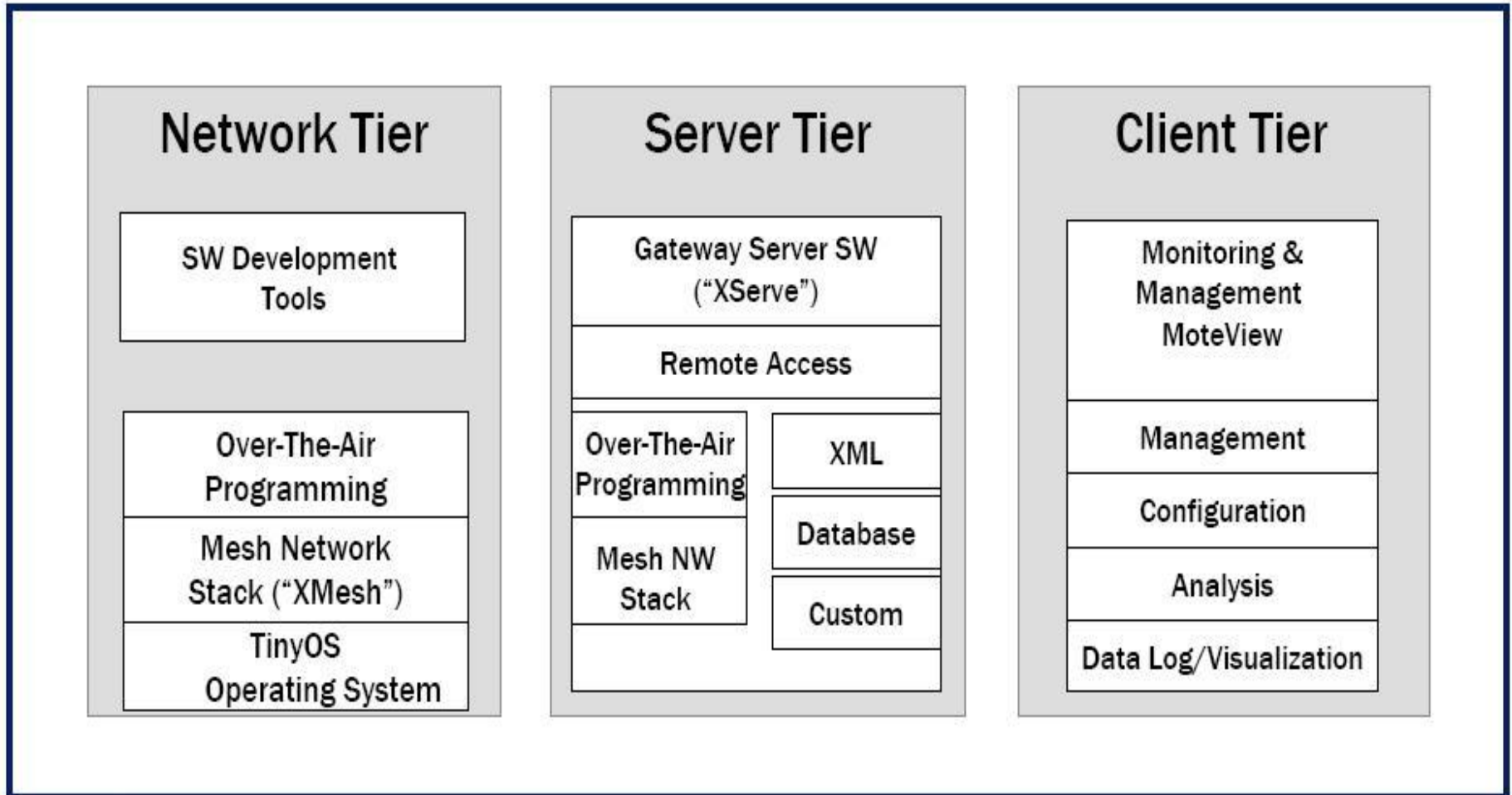
Description of MoteWorks Platform



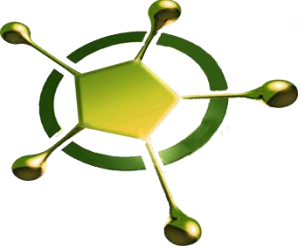
- Uses Three distinct software tiers
 - Mote Tier – XMesh
 - Sever Tier – XServe
 - Client Tier – MoteView

- Also includes :
 - Low Power Operating System – TinyOS
 - Software Development Tools

MoteWorksTM Software Platform

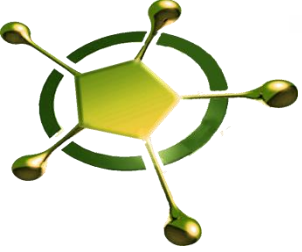


Hardware Platform



Hardware Requirements

- Two Motes: standard editions of IRIS (XM2110)
- One sensor or data acquisition board: MDA100, MTS300 or MTS310
- One gateway board: MIB510, MIB520, or MIB600 and the associated hardware (cables, power supply) for each
- A Windows PC with MoteWorks installed

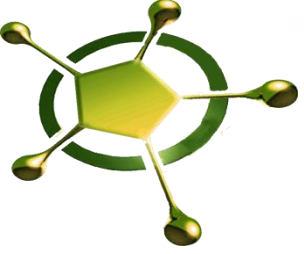


Developing an application



Within the MoteWorks framework a minimum of five files will be placed in any application's directory:

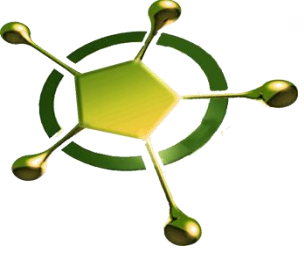
1. Makefile
2. Makefile.component
3. Application's configuration written in nesC
4. Application's module written in nesC
5. README (optional)



Two Variation



1. Sensor and Mote plugged into the base station directly through the serial port
2. Sending sensor data over the radio to the another Mote plugged into the base station directly through the serial port

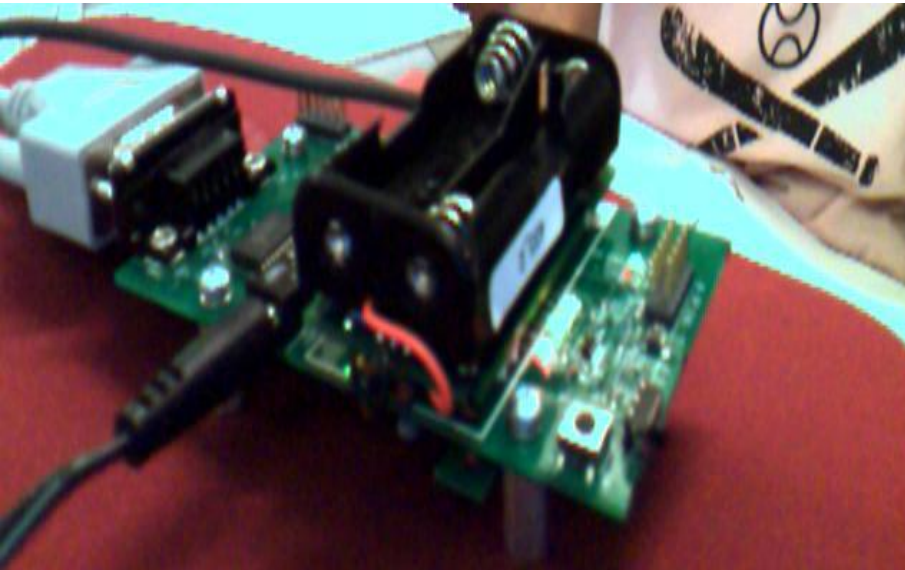


Variation 1

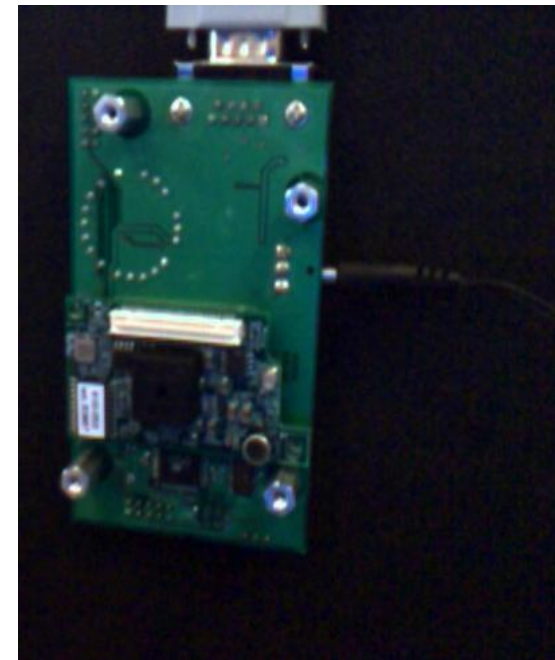


- **Hardware setup**

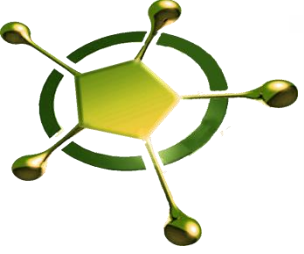
Connect a mote on top and sensor board on downside of MIB510



Top View



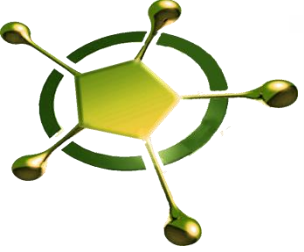
Downside View



Variation 1



- Use Programmer's Notepad
- Compile: Select Tools > make iris
- If successful we get "writing TOS image" in output
- Load Program: Select Tools>shell and type in
make iris reinstall mib510,com1 or Use MoteConfig
- Output: xserve -device=COM1

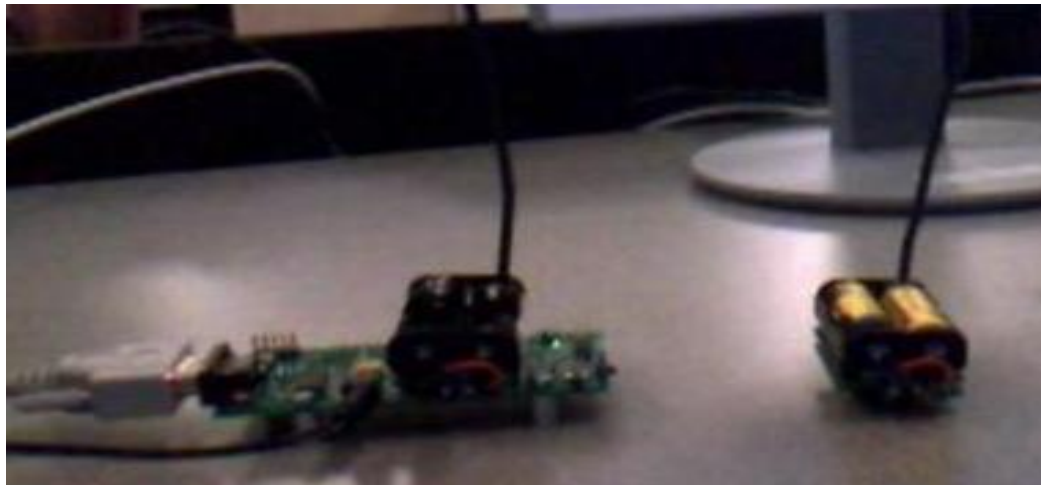


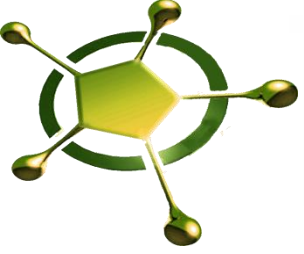
Variation 2



■ Hardware setup

Connect a mote on gateway load program in it, plugout that mote and connect a sensor board on that mote. Connect another mote on sensor

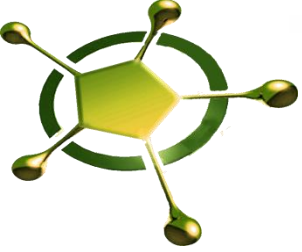




Variation 2



- Compile the application after change : Select Tools > make iris
- Install in Mote: : Select Tools>shell and type in
make iris install,1 mib510,com1 or Use MoteConfig
- Remove Mote from the programming board, plug sensor board on Mote and make sure it has battery and turn it on.
- Install Xsniffer application onto another Mote that remains plugged into board
- Use Xsniffer to see the display of packets



Code Update in logical groups using multiple gateways



- Over the Air Programming

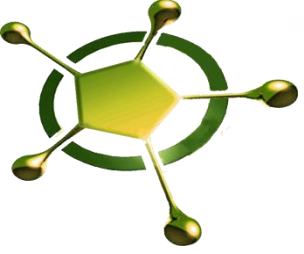
If a sensor node is already deployed in the network and code update is required without disturbing the network traffic.

- Logical Group

A group of nodes running same application.

- Multiple gateways

Using more than one base stations



References



- MoteWorks Getting Started Guide
 - Introduction
 - Introduction to TinyOS and nesC
 - First Steps in nesC Programming
 - A Simple Sensing Application
- Crossbow Product reference guide