TinyOS

Sensor Network Programming

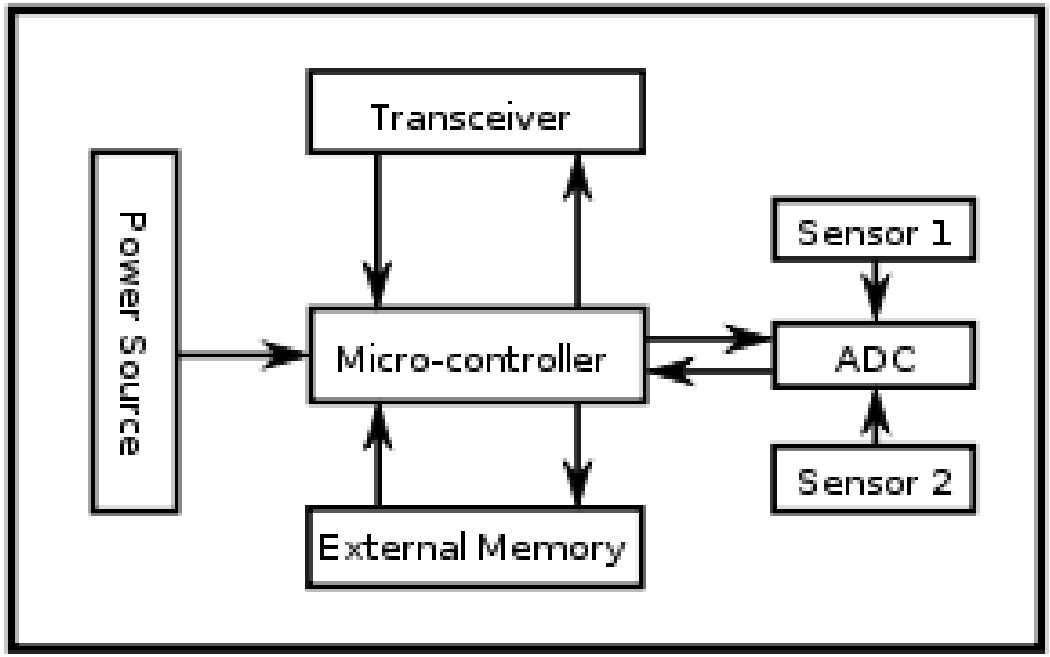
Lecture Overview

- 1. Hardware Primer
- 2. Introduction to TinyOS.
- 3. Programming TinyOS.
- 4. Hands on section.

Sensor node(mote):

1. Node in a wireless sensor network
2. Capable of performing some processing
3. Gathers information from sensors
4. Communicates with other connected nodes in the network.

Architecture of sensor node:



Sensor node:

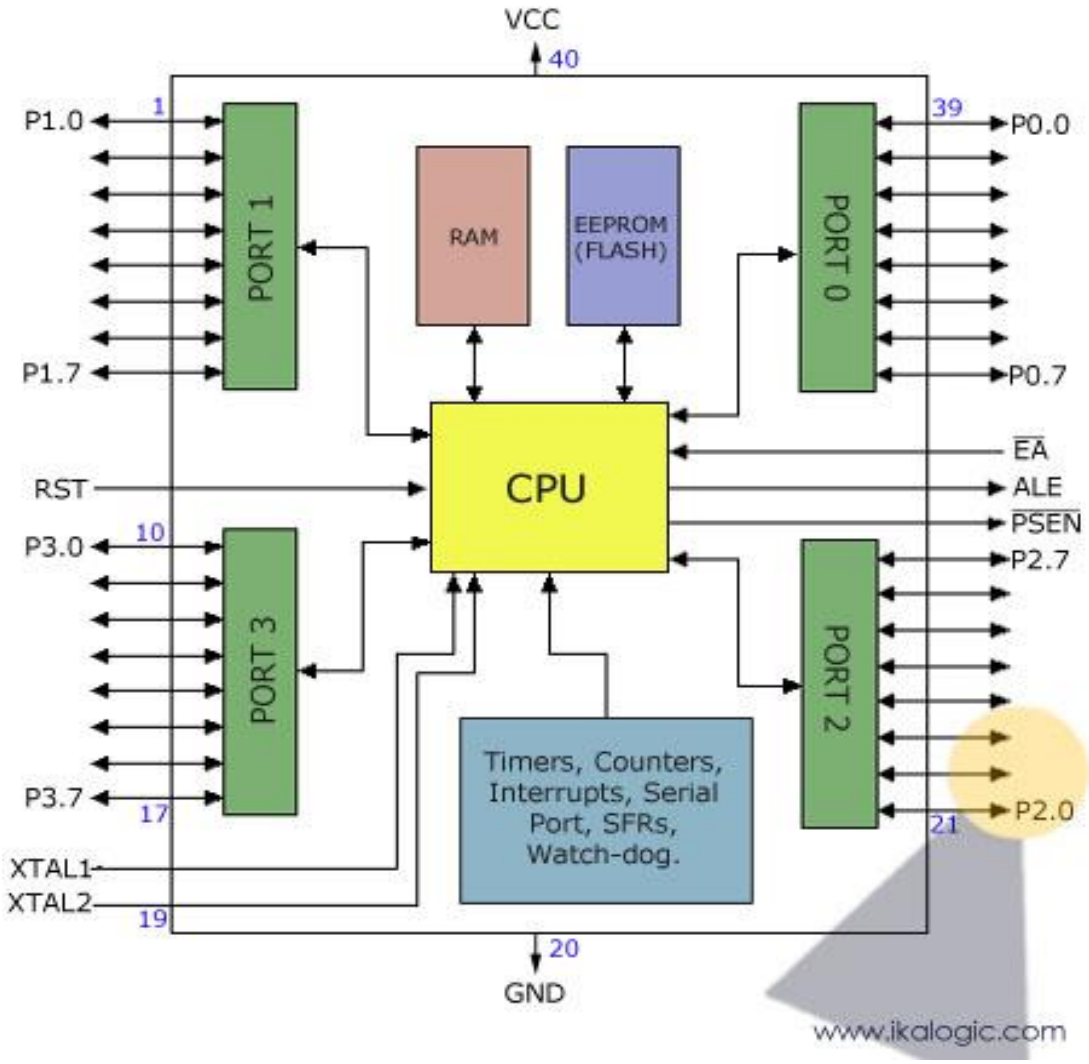
Two main parts

1. Microcontroller
2. Transceiver

IITH Mote specifications:

http://www.iith.ac.in/~raji/downloads/IITH-mote-webpage.pdf

Basic controller architecture



Purpose of controller

Functions of a mote:

- Collecting data from various sensors
- Process data and extract useful information
- Transmitter controlling
- Local storage maintenance

Purpose of controller

Data collections:

- Collecting data from various sensors, simultaneously.

- Data collected from individual sensors should have to be maintained properly.
- Sequential sampling (Reduces data rate).
- Adaptive sampling where one can adapt sampling rate based on some classification.

Purpose of controller

Data processing:

- Some applications require on board processing of the collected data.
- Most of the adaptive sampling algorithms use on board processing due to les delay.

Transceiver control:

- Controller can force transceiver into sleep mode when it is not needed.
- Can wake up transmitter, when there is some data to be transmitted.

Purpose of controller

Local storage maintenance:

- If the gateway is not in the range, then the data can be stored on to the local storage.
- When the gateway comes into vicinity, it can transmit the stored data and free up the local storage.

Power gating:

- Some of the functional blocks which are not necessary at present can be switched off to conserve power and can only be turned on when needed.

How to select a controller?

Things to keep in mind

- Power consumption
- Processing required
- Mode of communication (Baseband and RF processing)
- Priority of application (Medical or Pollution data)

Simple controller example

ATMEGA128 uC

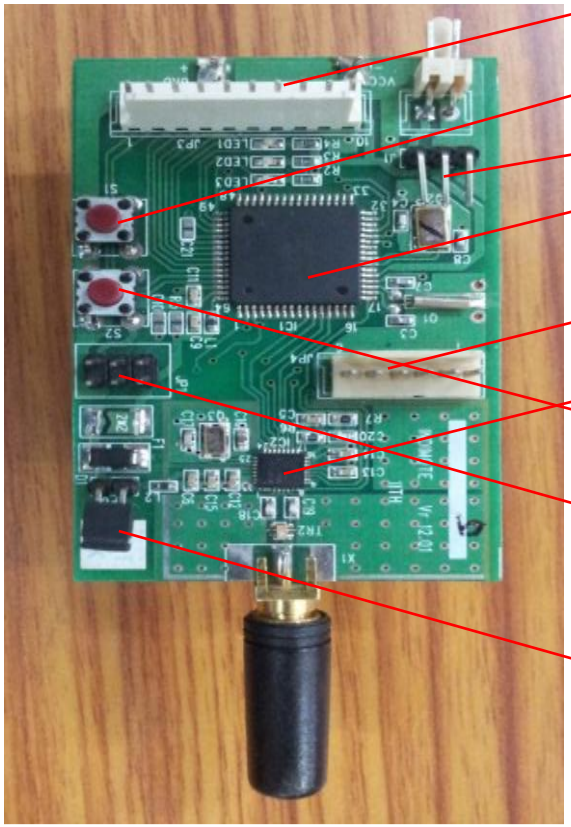
- 8 bit architecture
- 8 channel ADC (10 bit resolution)
- TWI
- 2 UART interfaces
- SPI interface (To interface additional memory)
- Can run TinyOS & Contiki.

Transceiver

AT86RF230:

Low Power 2.4 GHz Transceiver for ZigBee, IEEE 802.15.4, 6LoWPAN, ISM Applications.

IITH Mote(sensor node):



ADC/IO port

User button

UART Power jumper

Microcontroller(ATMEGA1281V)

UART port

Transceiver(AT86RF230)

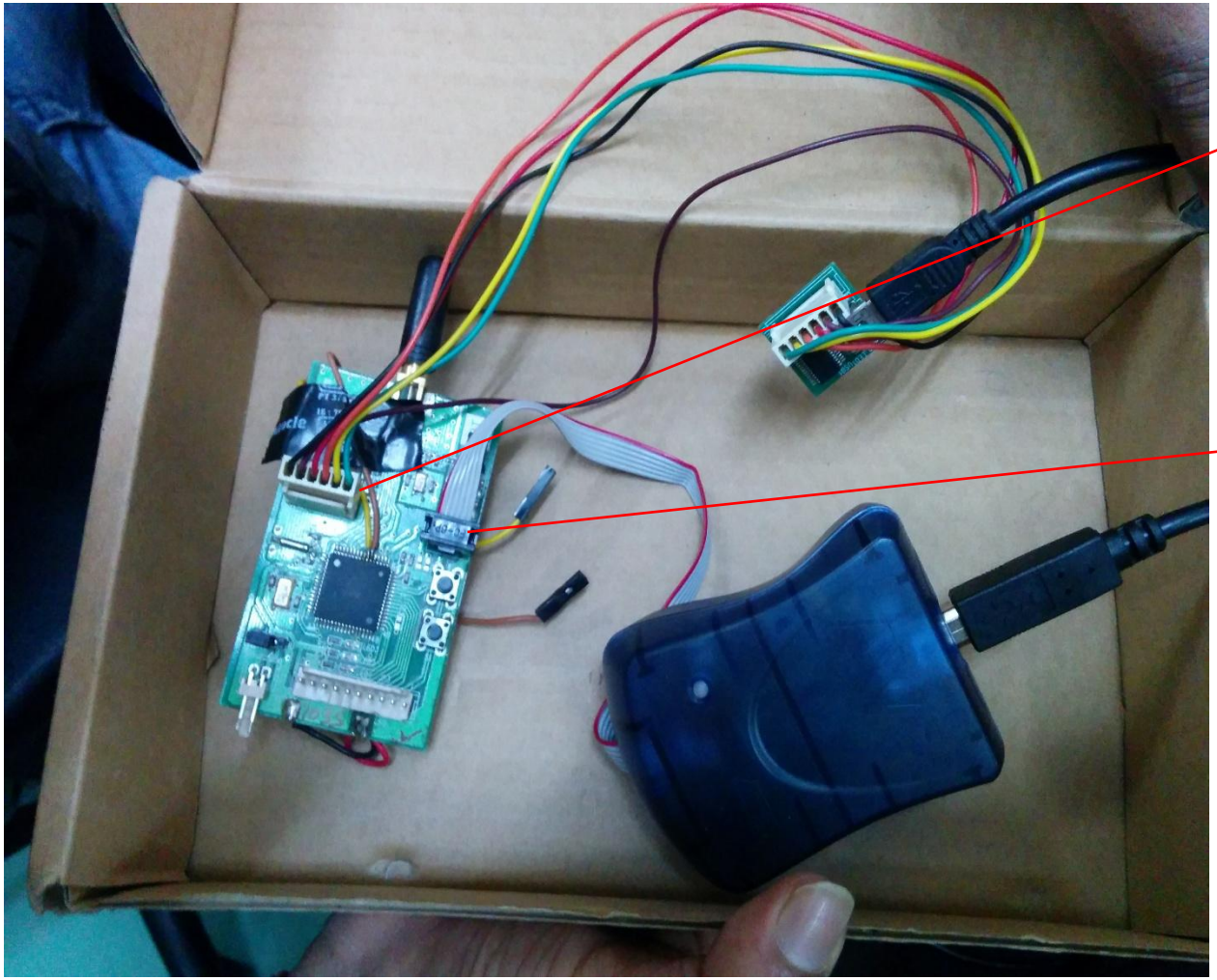
Reset button

Programming port

Programming jumper

IITH Mote specifications:
http://www.iith.ac.in/~raji/downloads/IITH-mote-webpage.pdf

Hardware setup to programming



UART port con.

Programmer con.

UC Berkeley Family of Motes

Mote Type	*WeC*	*René*	*René 2*	*Dot*	*Mica*	*Mica2Dot*	*Mica 2*	*Telos*
Year	1998	1999	2000	2000	2001	2002	2002	2004
								
Microcontroller								
Type	AT90LS8535	ATmega163		ATmega128				TI MSP430
Program memory (KB)	8	16		128				60
RAM (KB)	0.5	1		4				2
Active Power (mW)	15	15		8		33		3
Sleep Power (μW)	45	45		75		75		6
Wakeup Time (μs)	1000	36		180		180		6
Nonvolatile storage								
Chip	24LC256			AT45DB041B				ST M24M01S
Connection type	I^2C			SPI				I^2C
Size (KB)	32			512				128
Communication								
Radio	TR1000			TR1000		CC1000		CC2420
Data rate (kbps)	10			40		38.4		250
Modulation type	OOK			ASK		FSK		O-QPSK
Receive Power (mW)	9			12		29		38
Transmit Power at 0dBm (mW)	36			36		42		35
Power Consumption								
Minimum Operation (V)	2.7		2.7			2.7		1.8
Total Active Power (mW)	24			27		44	89	41
Programming and Sensor Interface								
Expansion	none	51-pin	51-pin	none	51-pin	19-pin	51-pin	10-pin
Communication	IEEE 1284 (programming) and RS232 (requires additional hardware)							USB
Integrated Sensors	no	no	no	yes	no	no	no	yes

Lecture Overview

- 1. Hardware Primer
- 2. Introduction to TinyOS
- 3. Programming TinyOS
- 4. Hands on section.

What is TinyOS?

- An operation system
- An open-source development environment
- Not an operation system for general purpose, it is designed for wireless embedded sensor network.
  - Official website: http://www.tinyos.net/
- Programming language: NesC (an extension of C)
- It features a component-based architecture.
- Supported platforms include Linux, Windows 2000/XP with Cygwin.

Now add this at the end of the file **deb http://nmrg.cs.jhu.edu/tinyos hardy main**

Install TinyOS

1. Install Ubuntu 12.04/13.04/14.04 or any higher versions.
2. Enable root user.
3. Switch to root user to install TinyOS.
4. Open terminal (Ctrl+Alt+T).

Installation procedure:

1. `gedit /etc/apt/source.list`

Add this at end of the file
deb http://hinrg.cs.jhu.edu/tinyos hardy main

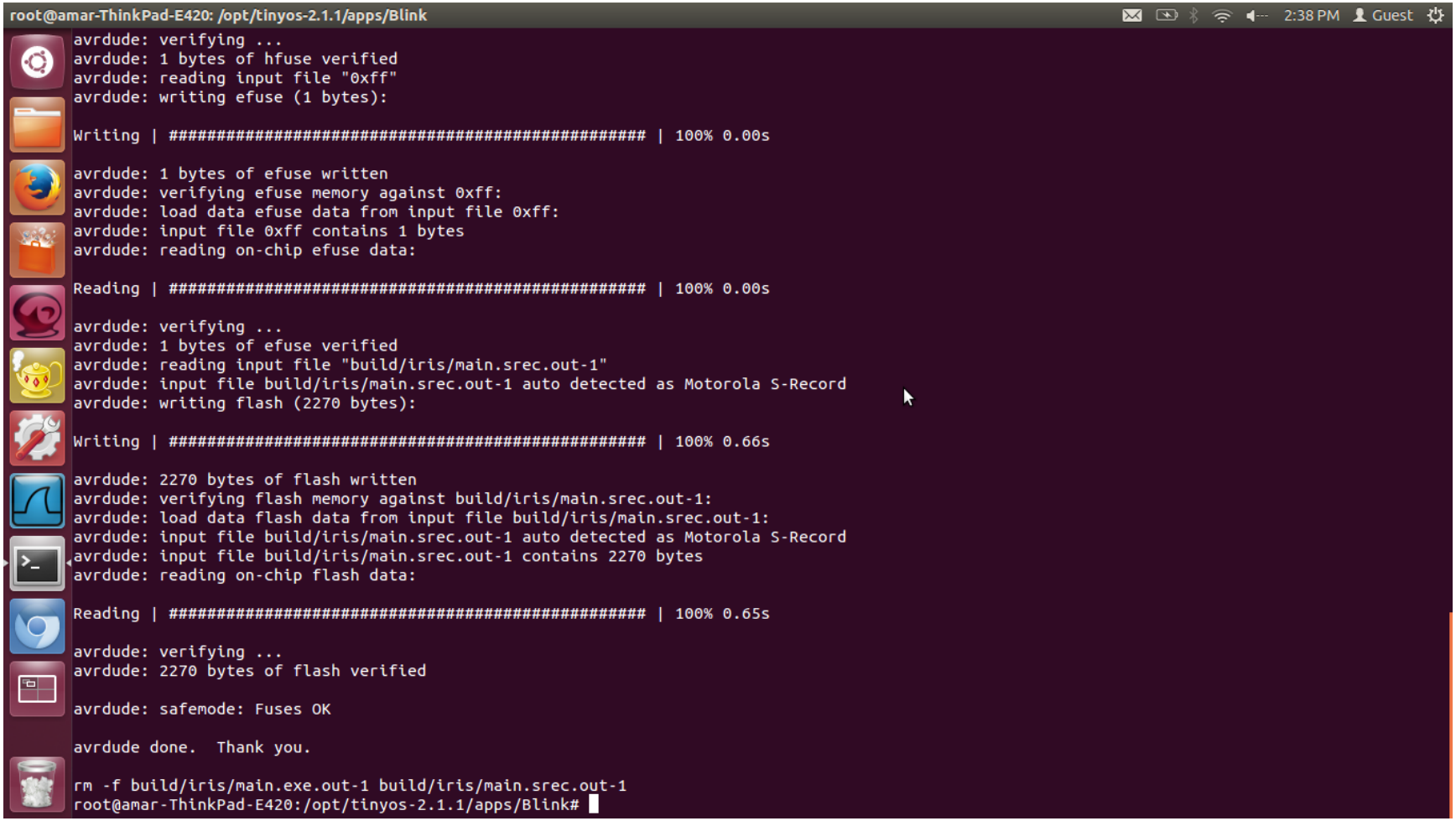
2. apt-get update
3. `apt-get install tinyos-2.1.1`
4. gedit ~/.bashrc

Add this at end of file

```
#Sourcing the tinyos environment variable
setup script source /opt/tinyos-
2.1.1/tinyos.sh
```

Compile and install program

Program installation view on terminal



Lecture Overview

- 1. Hardware Primer
- 2. Introduction to TinyOS
- 3. Programming TinyOS
- 4. Hands on section.

Program files

Every application needs 4 files

1. Make file (Makefile)
2. Configuration file (SensorAppC.nc)
3. Module file (SensorC.nc)
4. Header file (Sensor.h) (if application needs)

Sensor is application name.

check example application in TinyOS

cd /opt/tinyos-2.1.1/apps/ (path)

cd /opt/tinyos-2.1.1/apps/tutorials (path)

To develop application gedit or eclips IDE can be used

https://www.youtube.com/watch?v=lO5spZwKwRQ

Editors for writing a application

Gedit:

Create a folder with your application name.

Open terminal

Cntrl+Alt+T

Open a document by using gedit command

And save with your application name.

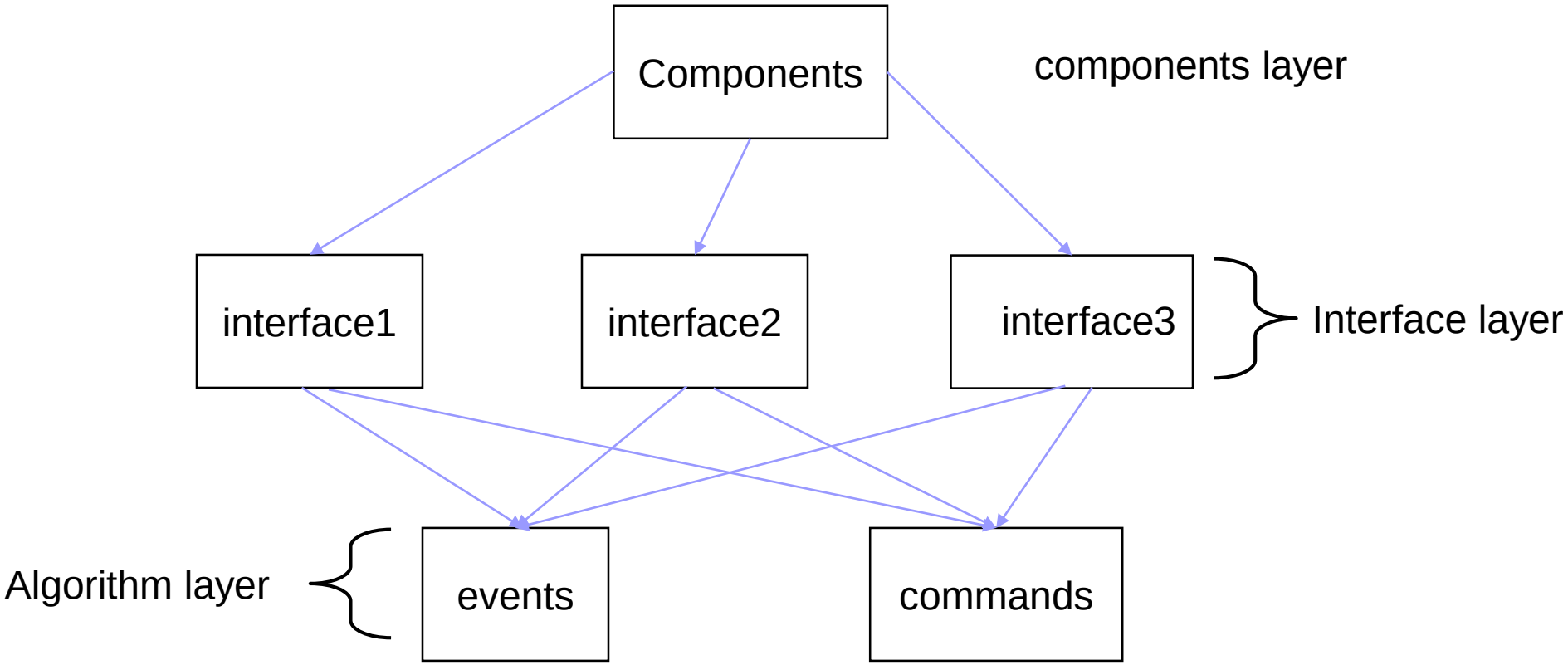
gedit documentname

create 4 files with mentioned extension in one folder.

How to write a application

Programming structure:

1. Search interfaces required for your application.
2. Search components which provides those interfaces.
3. Use commands and events which will be provided by interfaces to develop algorithms.


Components
components layer
interface1
interface2
interface3
Interface layer
Algorithm layer
events
commands

How to write application

Makefile: compiler can compile program.

“COMPONENT= SensorAppC

include $(MAKERULES)”

Configuration file(SensorAppC.nc):

File contains components which provides and uses interfaces.

1. Initialization of components.
2. Wiring of components with interfaces.

Components example:

MainC, LedsC, TimerMilliC.

http://www.tinyos.net/tinyos-2.1.0/doc/nesdoc/micaz/

Module file(SensorC.nc):

1.File contains Interfaces initialization and using interfaces.

2.Interfaces contains commands and events.

3.Commands and events are used to develop algorithm.

Component Syntax - Configuration

Component Selection

Wiring the Components together

```
configuration SensorAppC
{
}
implementation
{
  components MainC, SensorC, LedsC;
  components new TimerMilliC() as Timer0;
  components new TimerMilliC() as Timer1;
  components new TimerMilliC() as Timer2;
  SensorC -> SensorC.Boot;

  SensorC.Timer0 -> Timer0;
  SensorC.Timer1 -> Timer1;
  SensorC.Timer2 -> Timer2;
  SensorC.Leds -> LedsC;
}
```

Module syntax:

```
#include "Timer.h"

module SensorC()
{
  uses interface Timer<TMilli> as Timer0;
  uses interface Timer<TMilli> as Timer1;
  uses interface Timer<TMilli> as Timer2;
  uses interface Leds;
  uses interface Boot;
}
implementation
{
  event void Boot.booted()
  {
    call Timer0.startPeriodic( 250 );
    call Timer1.startPeriodic( 500 );
    call Timer2.startPeriodic( 1000 );
  }
  event void Timer0.fired()
  {
    call Leds.led0Toggle();
  }
  event void Timer1.fired()
  {
call Leds.led1Toggle();
  }
  event void Timer2.fired()
  {
call Leds.led2Toggle();
  }
}
```

interface *X* as Y

= interface *X* as X

commands

Event

Lecture Overview

- 1. Hardware Primer
- 2. Introduction to TinyOS
- 3. Programming TinyOS
- 4. Hands on section

Try new applications

Further Reading

- Go through the on-line tutorial:
  - http://www.tinyos.net/tinyos-1.x/doc/tutorial/index.html
- Search the help archive:
  - http://www.tinyos.net/search.html
- NesC language reference manual:
  - http://www.tinyos.net/tinyos-1.x/doc/nesc/ref.pdf

Thank you.