

Chapter 1:  
Programming the Motes

=====

There are 3 types of Motes, each type with different application layer, but loosely speaking, they share the same MAC layer.

The types are:

1. PersonalMonitor

code: chainradio/apps/PersonalMonitor/

make: make -f Makefile.pm micaz SOURCE=1 MOTE\_ID=101 install.101 mib510,/dev/ttyS1

Makefile: Makefile.pm (make command is also inside this file)

2. RTChainMidNodes

code: chainradio/apps/RTChainMidNodes

make: makemicaz 2 1

makemicaz <id> <serial\_tty>

id must be between 2 and 99

This function is defined inside bashrc file to make make easier.

```
function makemicaz () {
    make micaz MOTE_ID=$1 $3 $4 install.$1 mib510,/dev/ttyS$2
}
```

So makemicaz 2 1 translates to: make micaz MOTE\_ID=2 install.2 mib510,/dev/ttyS1

3. RTBase

code: chainradio/apps/RTChainMidNodes

make: makemicaz 1 1

makemicaz <id> <serial\_tty>

id must be 1 <default base> and greater than 149 (not including 149)

Chapter 2:

Code

=====

2.1 Applications

-----

There should not be any changes that is required of RTChainMidNodes as they are simply forwarders.

RTBase is responsible for

1. forwarding received packets to UART

2. parsing commands from MonitoringSystem and then sending the command to Radio.

Therefore, it is important to find out what are the correct

TOS\_Msg data[] index to use. These are defined in:

chainradio/apps/Utils/MessageTypes.h

2.2 MAC Layer

-----

RTChain

code: chainradio/tos/lib/RTChain/RTChainM.nc

code: chainradio/tos/lib/CC2420/

This is the MAC layer code.

Black-burst implementation is inside CC2420RadioM.nc

RTChainM is a layer on top of CC2420RadioM that provides

RT-Chain functionalities.

NormalChannel

Normal channel is the shared channel where motes communicate usually.

When motes use RT-Chain they switch to a different channel.

Therefore, if localization does not work, make sure every mote has the same normal channel. Or if some other research labs uses the same normal channel, make sure to switch to a different one.

NormalChannel is defined in chainradio/tos/lib/RTChain/RTChain.h

## Chapter 3:

## MonitoringSystem (TinyJava)

```
=====
```

MonitoringSystem should be relatively easy to use, it has a decent GUI to layout the Motes.

You can launch MonitoringSystem from Eclipse, or from the command line if you have the CLASSPATH environment variable set correctly as instructed by TinyOS installation (Chapter 6

give some tips on installing TinyOS). The MonitoringSystem/Tinyjava archive also contains swing library, which needs to be added into the CLASSPATH.

An example run will be:

1. cd into TinySenses/net/tinyos/tools/monitor/gui directory.
2. java -cp \$CLASSPATH:../../../../../../../../swing-layout-1.0.jar:. net.tinyos.tools.monitor.gui.MonitoringSystemView serial

The "serial" argument is selected by default, it asks MonitoringSystem to read from its local serial port.

If MonitoringSystem is deployed on a windows machine, then please run SerialForwarder.exe on a local machine

or networked machine that have access to the serial ports. And then run MonitoringSystem with the following arguments:

```
network hostname:hostport
```

1. To create notes:

Left click, and give them names. (Don't set the connection first)

2. To delete notes:

Right click on them.

3. You can also move them around by dragging.

4. To create walls:

Left click with CTRL key down to select start point.

Left click with CTRL key down again to select end point.

To create a decent wall, draw the line such that it is not entirely straight.

(This is because the wall is a rectangle, therefore to give some thickness to the rectangle, we want the line to be slightly diagonal, otherwise the wall becomes really... thin).

I do not recommend creating diagonal walls, as they will appear rather ugly.

5. To delete walls:

Right click with CTRL key down on the wall.

6. To Edit a Mote's properties:

-double click the mote to start MoteEditor

You can rename a mote, associate it with a person and setup its upstream connection.

7. The connectivity graph must be a tree. I did not test what the result is if it is not a tree.

8. You can send out this connectivity graph to all motes by using the Command option on the Control Panel (right hand side).

9. Notice that you can also send different commands.

10. The MonitoringSystem can be in server/client/default mode.

Usually, we only want it to be in default mode.

To remote control option, make the one you wish to control as a server on the Control Panel.

Then set the other machine as client mode. So far I have hardcoded the server address to be cs-grad129.cs.uiuc.edu.

## Chapter 4:

## Deployment

=====

These are a few things that I try to adhere to when I do testing:

1. Try to place the RTBase at a higher position, so that the signal is not blocked by most people.
2. Try to place RTMidNodes further away from each other to provide better localization accuracy.

For example, to cover an area as large as 3 office rooms of 6x6 meters. only 2 is needed.

## Chapter 5

=====

## Using MonitoringSystem with Windows

-----

1. Due to the absence of JAVA's comm3.0 library for Windows, we cannot use the existing serial source reader provided by TinyOS' java tools. Therefore we will use a C# program that reads from the serial port, and send raw bytes via TCP to it.

## Chapter 6

=====

## Installing TinyOS

-----

For redhat/fedora distros, consider <http://www.matthewjmilller.net/howtos/installing-tinyos-for-telos-on-linux/> to be a good set of instructions.  
For debian/ubuntu distros, consider <http://thomer.com/tinyos/> as a good list of instructions.