R·I·T | **Rochester Institute of Technology**
Department of Information Technology

VKSF 582/782 Lab 2 – Sensor Network Basics

Laboratory Report

Instructions

The laboratory report is required for each session completed.

Basic Code Upload

1. In activity number 5 you explored the file structure and an application called Blink. List all the files present in the directory and explain their purpose.

Blink.nc – this file is the top level configuration file for the Blink application, and the source file that the nesC compiler uses to generate an executable file. A configuration that simply wires together one or more components used by the Blink module below.

BlinkM.nc – this file is responsible for containing the actual code which will be executed by a specific application

Makefile - details the files, dependencies, and rules by which an executable application is built.

README - provides a description of the Blink application

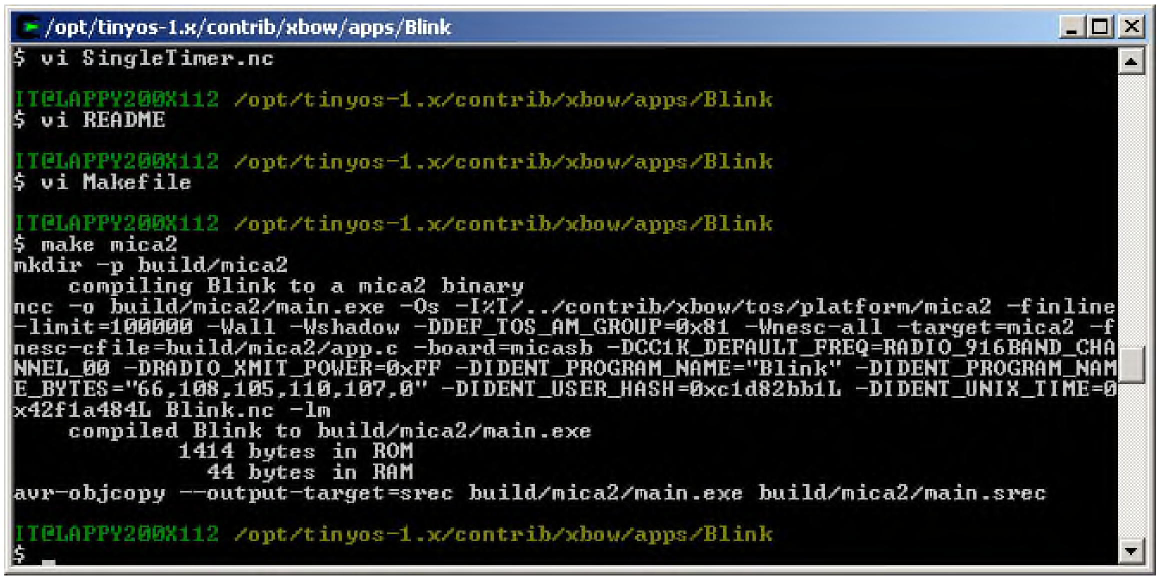
SingleTimer.nc – this file provides a timer interface for the Blink application, which enables the user to implement it in a written program, providing the ability to create timed events.

2. Provide a screenshot showing the result of a program compilation in activity 6. Can you describe what the process of programming a series of motes with this application? What are the ROM and RAM values?

The process of programming a series of motes can be described as :

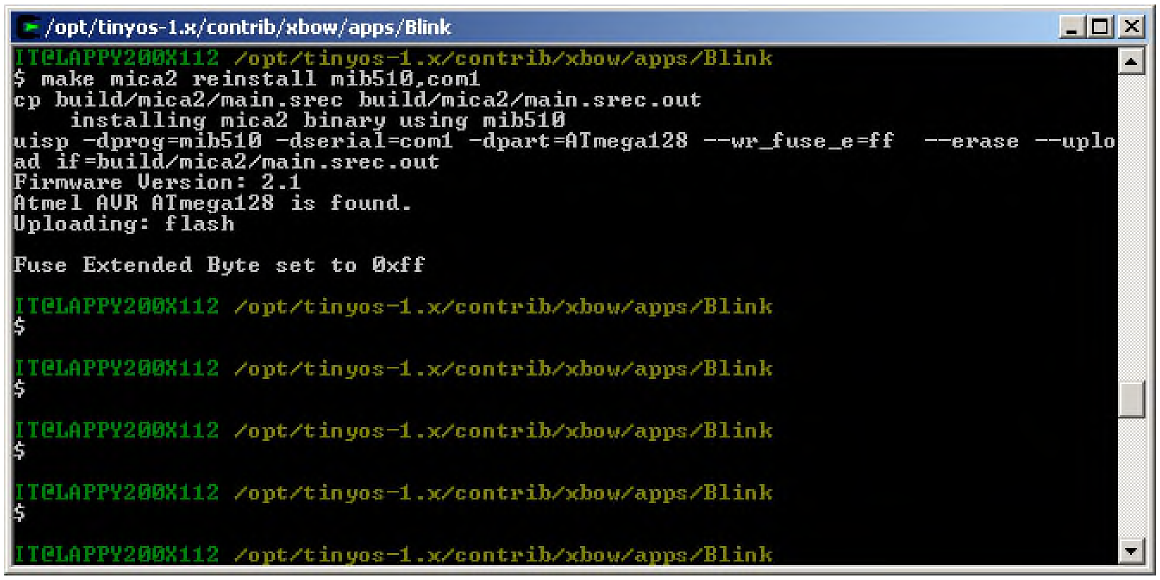
-write or edit your code
-compile and check for errors and problems
-fix errors and problems
-compile, if no errors occur, upload to each mote required

The ROM value was 1414 bytes and the RAM value was 44 bytes. These values represent the memory used on each MICA2 mote.



```
/opt/tinyos-1.x/contrib/xbow/apps/Blink
$ vi SingleTimer.nc

IT@LAPPY200X112 /opt/tinyos-1.x/contrib/xbow/apps/Blink
$ vi README

IT@LAPPY200X112 /opt/tinyos-1.x/contrib/xbow/apps/Blink
$ vi Makefile

IT@LAPPY200X112 /opt/tinyos-1.x/contrib/xbow/apps/Blink
$ make mica2
mkdir -p build/mica2
    compiling Blink to a mica2 binary
ncc -o build/mica2/main.exe -Os -I%T/../contrib/xbow/tos/platform/mica2 -finline
-limit=100000 -Wall -Wshadow -DDEF_TOS_AM_GROUP=0x81 -Wnesc-all -target=mica2 -f
nesc-cfile=build/mica2/app.c -board=micasb -DCC1K_DEFAULT_FREQ=RADIO_916BAND_CHA
NNEL_00 -DRADIO_XMIT_POWER=0xFF -DIDENT_PROGRAM_NAME="Blink" -DIDENT_PROGRAM_NAM
E_BYTES="66,108,105,110,107,0" -DIDENT_USER_HASH=0xc1d82bb1L -DIDENT_UNIX_TIME=0
x42f1a484L Blink.nc -lm
    compiled Blink to build/mica2/main.exe
            1414 bytes in ROM
              44 bytes in RAM
avr-objcopy --output-target=srec build/mica2/main.exe build/mica2/main.srec

IT@LAPPY200X112 /opt/tinyos-1.x/contrib/xbow/apps/Blink
$ _
```

3. In activity 8 you uploaded the compiled code to the MICA2 motes provide a screenshot of a successful code upload. Explain the last three lines of the resulting output.



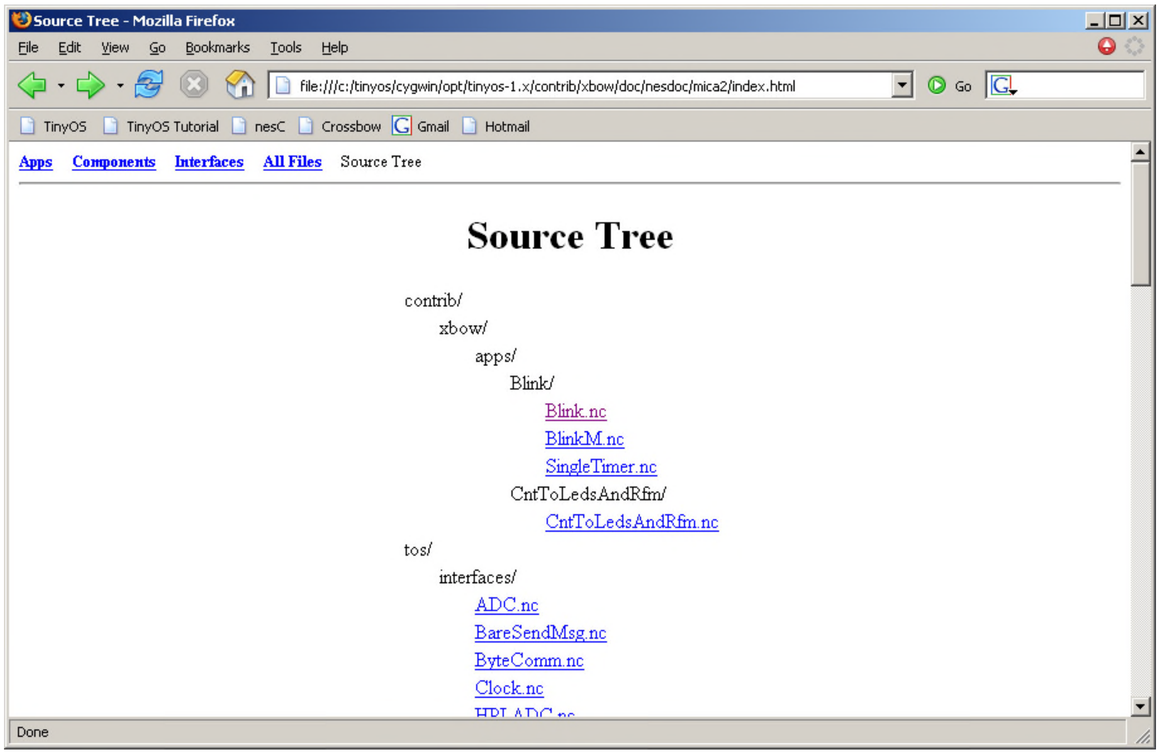
```
/opt/tinyos-1.x/contrib/xbow/apps/Blink
IT@LAPPY200X112 /opt/tinyos-1.x/contrib/xbow/apps/Blink
$ make mica2 reinstall mib510,com1
cp build/mica2/main.srec build/mica2/main.srec.out
    installing mica2 binary using mib510
uisp -dprog=mib510 -dserial=com1 -dpart=ATmega128 --wr_fuse_e=ff  --erase --uplo
ad if=build/mica2/main.srec.out
Firmware Version: 2.1
Atmel AVR ATmega128 is found.
Uploading: flash

Fuse Extended Byte set to 0xff

IT@LAPPY200X112 /opt/tinyos-1.x/contrib/xbow/apps/Blink
$

IT@LAPPY200X112 /opt/tinyos-1.x/contrib/xbow/apps/Blink
$

IT@LAPPY200X112 /opt/tinyos-1.x/contrib/xbow/apps/Blink
$

IT@LAPPY200X112 /opt/tinyos-1.x/contrib/xbow/apps/Blink
$

IT@LAPPY200X112 /opt/tinyos-1.x/contrib/xbow/apps/Blink
```

`Atmel AVR Atmega128 is found` – this line specifies that the processor on the MICA2 mote is found

`Uploading:  flash` – the processor uses a programmable flash memory, which is indicated here

`Fuse Extended Byte set to 0xff` – this last line shows the value to which a fuse on the Atmel processor is set to, these values can be altered to suit individual needs

4. Activity number 11 requested that you generate documentation for your newly compiled application, please provide a screenshot showing this accomplishment.


Source Tree - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
file:///c:/tinyos/cygwin/opt/tinyos-1.x/contrib/xbow/doc/nesdoc/mica2/index.html
Go
TinyOS TinyOS Tutorial nesC Crossbow Gmail Hotmail
Apps Components Interfaces All Files Source Tree
Done

Source Tree

```
contrib/
    xbow/
        apps/
            Blink/
                Blink.nc
                BlinkM.nc
                SingleTimer.nc
            CntToLedsAndRfm/
                CntToLedsAndRfm.nc
tos/
    interfaces/
        ADC.nc
        BareSendMsg.nc
        ByteComm.nc
        Clock.nc
```

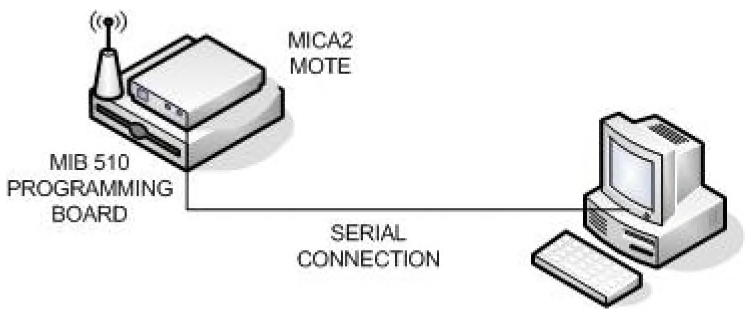
5. Explain what the Blink application is trying to accomplish.

The Blink application starts a timer which in turn blinks the red LED on the mote and the programming board in 1 second intervals.

6. Why is the LED on the MICA2 mote able to blink without any batteries?

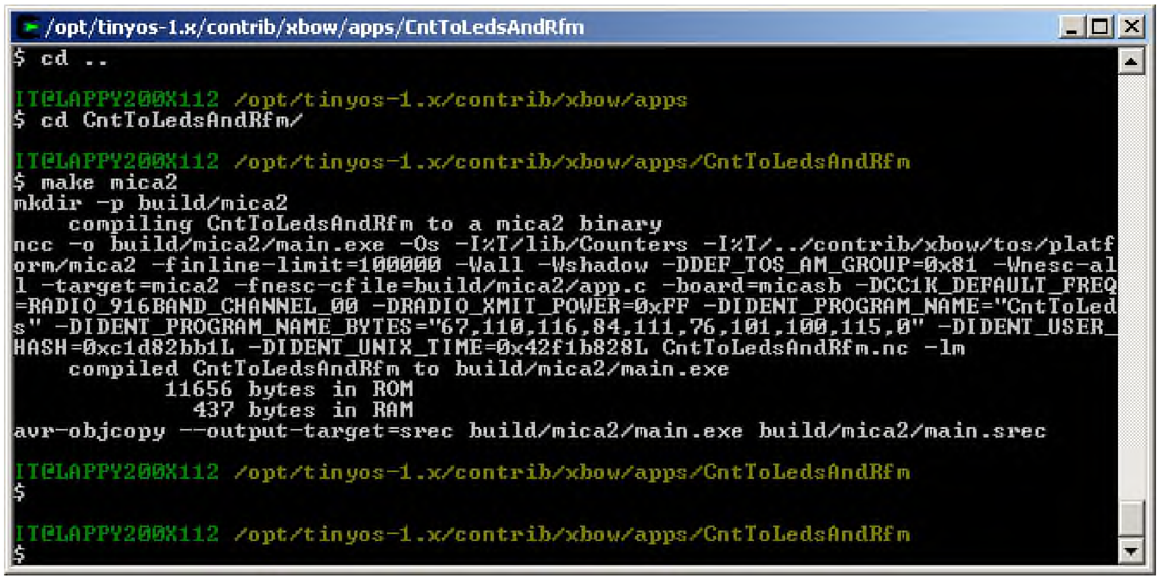
The LED is still able to blink because the MIB510 board provides power for the mote while the two are connected.

7. Please draw a topology of equipment used up to this point. Be as specific as possible including names and model numbers, labeling connections etc.



Motes and Radio Communication

8. In activity 17 you were asked to compile and upload the application `CntToLedsAndRfm` **to one of the motes, please provide a screenshot of the compilation process. What are the ROM and RAM values?**



```
$ cd ..

IT@LAPPY200X112 /opt/tinyos-1.x/contrib/xbow/apps
$ cd CntToLedsAndRfm/

IT@LAPPY200X112 /opt/tinyos-1.x/contrib/xbow/apps/CntToLedsAndRfm
$ make mica2
mkdir -p build/mica2
    compiling CntToLedsAndRfm to a mica2 binary
ncc -o build/mica2/main.exe -Os -I%T/lib/Counters -I%T/../contrib/xbow/tos/platf
orm/mica2 -finline-limit=100000 -Wall -Wshadow -DDEF_TOS_AM_GROUP=0x81 -Wnesc-al
l -target=mica2 -fnesc-cfile=build/mica2/app.c -board=micasb -DCC1K_DEFAULT_FREQ
=RADIO_916BAND_CHANNEL_00 -DRADIO_XMIT_POWER=0xFF -DIDENT_PROGRAM_NAME="CntToLed
s" -DIDENT_PROGRAM_NAME_BYTES="67,110,116,84,111,76,101,100,115,0" -DIDENT_USER_
HASH=0xc1d82bb1L -DIDENT_UNIX_TIME=0x42f1b828L CntToLedsAndRfm.nc -lm
    compiled CntToLedsAndRfm to build/mica2/main.exe
           11656 bytes in ROM
             437 bytes in RAM
avr-objcopy --output-target=srec build/mica2/main.exe build/mica2/main.srec

IT@LAPPY200X112 /opt/tinyos-1.x/contrib/xbow/apps/CntToLedsAndRfm
$

IT@LAPPY200X112 /opt/tinyos-1.x/contrib/xbow/apps/CntToLedsAndRfm
$
```

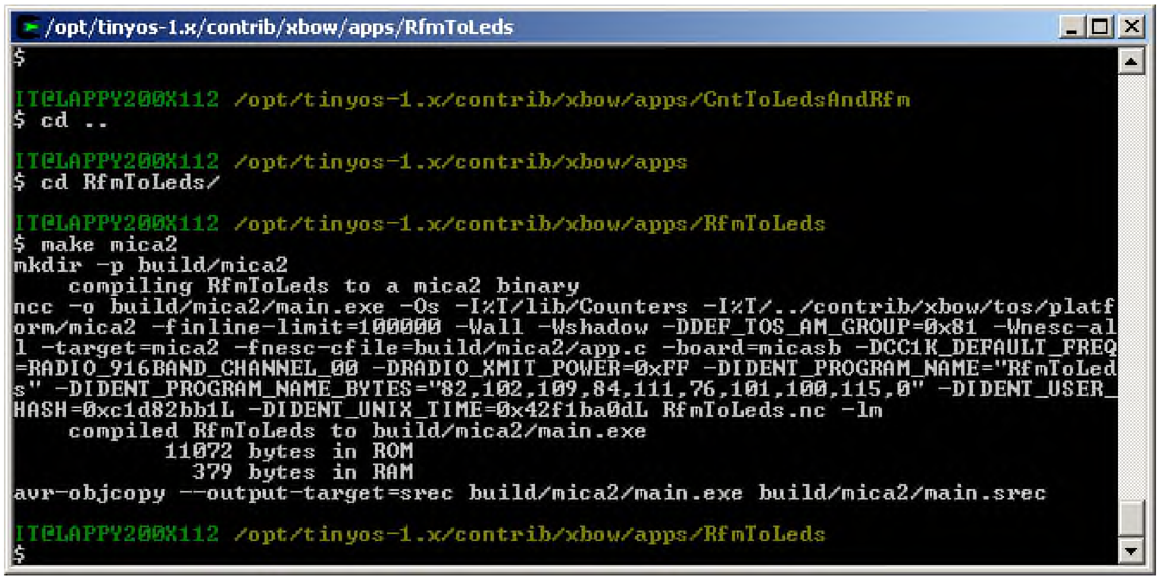
The ROM value was 11656 bytes and the RAM value was 437 bytes.

9. What were the commands used in the process of compiling and uploading the above application?

The commands used were:

-to compile : `make mica2`
-to upload : `make mica2 reinstall mib510,com1`

10. In activity 19 you were asked to compile and upload the application `RfmToLeds` to one of the motes, please provide a screenshot of the compilation process. What are the ROM and RAM values?



```
/opt/tinyos-1.x/contrib/xbow/apps/RfmToLeds
$

IT@LAPPY200X112 /opt/tinyos-1.x/contrib/xbow/apps/CntToLedsAndRfm
$ cd ..

IT@LAPPY200X112 /opt/tinyos-1.x/contrib/xbow/apps
$ cd RfmToLeds/

IT@LAPPY200X112 /opt/tinyos-1.x/contrib/xbow/apps/RfmToLeds
$ make mica2
mkdir -p build/mica2
    compiling RfmToLeds to a mica2 binary
ncc -o build/mica2/main.exe -Os -I%T/lib/Counters -I%T/../contrib/xbow/tos/platf
orm/mica2 -finline-limit=100000 -Wall -Wshadow -DDEF_TOS_AM_GROUP=0x81 -Wnesc-al
l -target=mica2 -fnesc-cfile=build/mica2/app.c -board=micasb -DCC1K_DEFAULT_FREQ
=RADIO_916BAND_CHANNEL_00 -DRADIO_XMIT_POWER=0xFF -DIDENT_PROGRAM_NAME="RfmToLed
s" -DIDENT_PROGRAM_NAME_BYTES="82,102,109,84,111,76,101,100,115,0" -DIDENT_USER_
HASH=0xc1d82bb1L -DIDENT_UNIX_TIME=0x42f1ba0dL RfmToLeds.nc -lm
    compiled RfmToLeds to build/mica2/main.exe
           11072 bytes in ROM
             379 bytes in RAM
avr-objcopy --output-target=srec build/mica2/main.exe build/mica2/main.srec

IT@LAPPY200X112 /opt/tinyos-1.x/contrib/xbow/apps/RfmToLeds
$
```

The ROM value was 11072 bytes and the RAM value was 379 bytes.

11. What were the commands used in the above process?

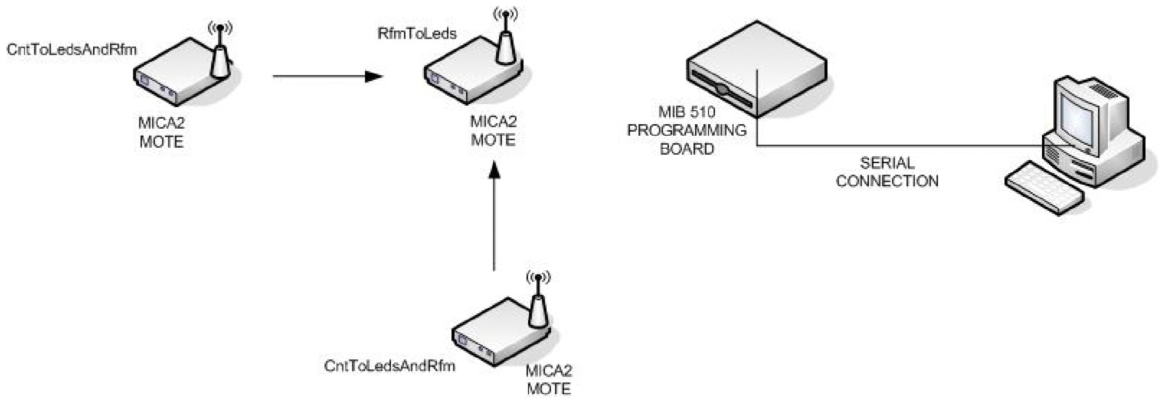
The commands used were:

-to compile : `make mica2`
-to upload : `make mica2 reinstall mib510,com1`

12. You used two applications to complete this section of the lab. Please explain the role of each one of those applications.

The role of the `CntToLedsAndRfm` application is to send values over the air waves representing colors of the LED's present on a MICA2 mote while also blinking the same LED's. The `RfmToLeds` application receives these values and mimics the behavior of the other mote by blinking the corresponding LED's.

13. Upon completion of this section you should have a working group of programmed MICA2 motes, please draw a topology of the utilized equipment. Be as specific as possible including names and model numbers, labeling connections etc.



Sensor and Data Acquisition Boards

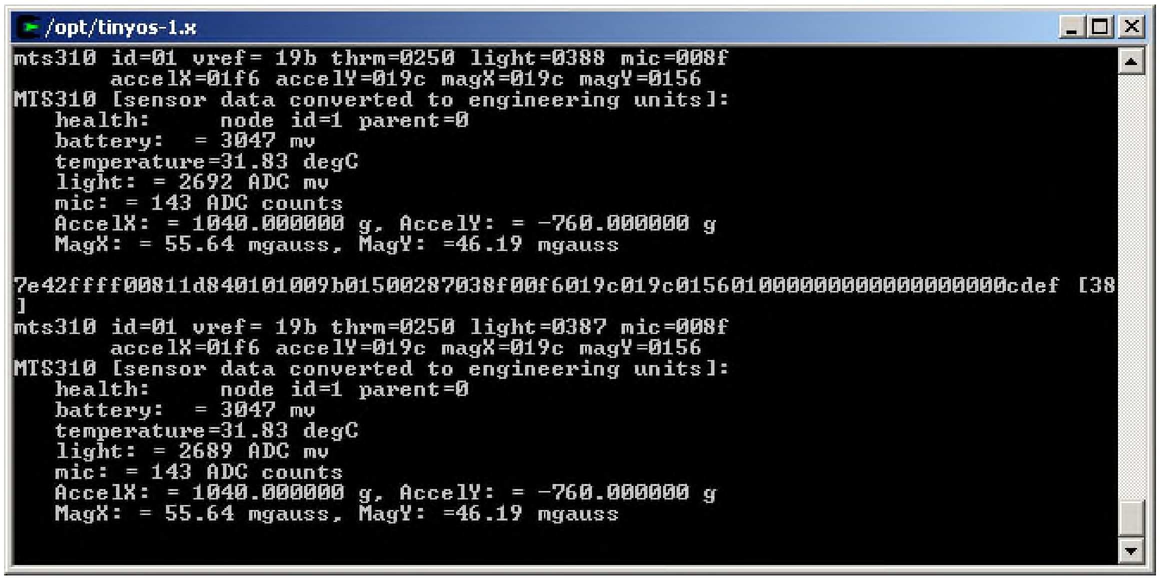
14. You used three applications to complete this section of the lab. Please explain the role of each one of those applications.

XSensorMTS300 – this application was used in testing the ability of all the sensors present on the sensor board to capture and transmit captured data. It enables the sensor board attached to a MICA2 mote to continually take environment readings. These reading are then relayed to the mote which transmits them over the radio.

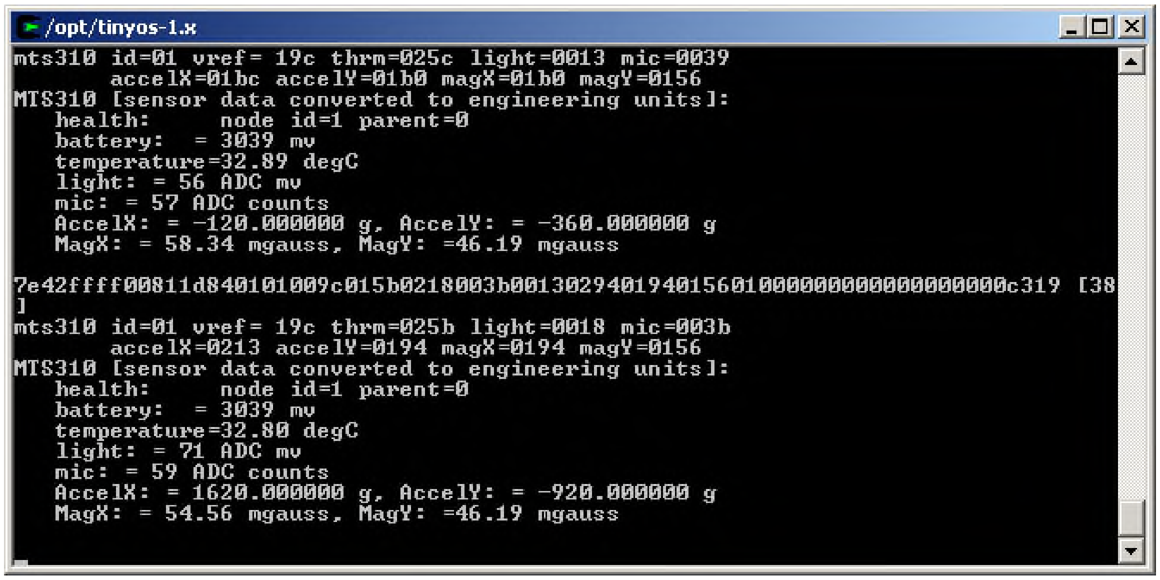
TOSBase – is an application that relays the information received from motes via a wireless connection to a serial connection which in turn carries that information to a PC.

XListen – is an application written in C that reads the raw data relayed by TOSBase from the motes and converts it to engineering units for a more readable format

15. Upon the successful implementation of the above applications you should be able to obtain sensor readings with XListen. Provide two separate screenshots showing your ability to alter at least two of these readings. Explain what action(s) you took to change particular sensor output. On your screenshots point to an area where the raw packet data is displayed.



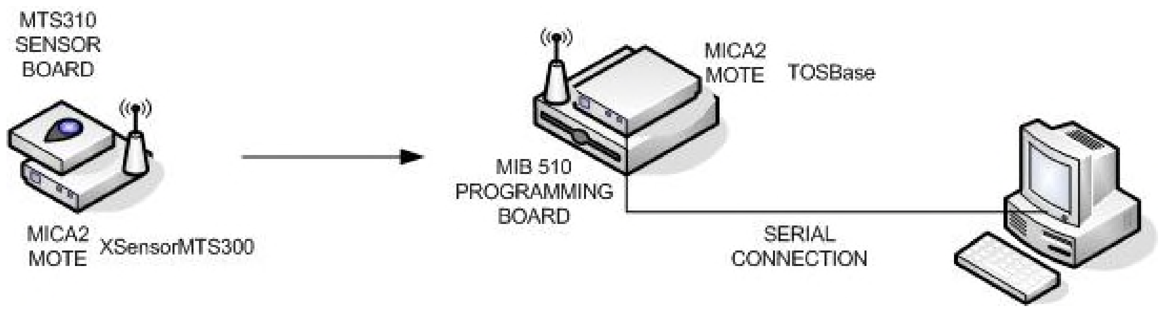
```
mts310 id=01 vref= 19b thrm=0250 light=0388 mic=008f
       accelX=01f6 accelY=019c magX=019c magY=0156
MTS310 [sensor data converted to engineering units]:
   health:      node id=1 parent=0
   battery:   = 3047 mv
   temperature=31.83 degC
   light: = 2692 ADC mv
   mic: = 143 ADC counts
   AccelX: = 1040.000000 g, AccelY: = -760.000000 g
   MagX: = 55.64 mgauss, MagY: =46.19 mgauss

7e42ffff00811d840101009b01500287038f00f6019c019c0156010000000000000000000cdef [38
]
mts310 id=01 vref= 19b thrm=0250 light=0387 mic=008f
       accelX=01f6 accelY=019c magX=019c magY=0156
MTS310 [sensor data converted to engineering units]:
   health:      node id=1 parent=0
   battery:   = 3047 mv
   temperature=31.83 degC
   light: = 2689 ADC mv
   mic: = 143 ADC counts
   AccelX: = 1040.000000 g, AccelY: = -760.000000 g
   MagX: = 55.64 mgauss, MagY: =46.19 mgauss
```



```
mts310 id=01 vref= 19c thrm=025c light=0013 mic=0039
       accelX=01bc accelY=01b0 magX=01b0 magY=0156
MTS310 [sensor data converted to engineering units]:
   health:      node id=1 parent=0
   battery:   = 3039 mv
   temperature=32.89 degC
   light: = 56 ADC mv
   mic: = 57 ADC counts
   AccelX: = -120.000000 g, AccelY: = -360.000000 g
   MagX: = 58.34 mgauss, MagY: =46.19 mgauss

7e42ffff00811d840101009c015b0218003b00130294019401560100000000000000000000c319 [38
]
mts310 id=01 vref= 19c thrm=025b light=0018 mic=003b
       accelX=0213 accelY=0194 magX=0194 magY=0156
MTS310 [sensor data converted to engineering units]:
   health:      node id=1 parent=0
   battery:   = 3039 mv
   temperature=32.80 degC
   light: = 71 ADC mv
   mic: = 59 ADC counts
   AccelX: = 1620.000000 g, AccelY: = -920.000000 g
   MagX: = 54.56 mgauss, MagY: =46.19 mgauss
```

In the above example I was able to alter the light sensor reading by covering the light sensor with my palm. The Accelerometer reading was changed by picking up the mote and moving it suddenly from right to left and vice versa.

16. Upon completion of this section you should have a working group of programmed MICA2 motes, please draw a topology of the utilized equipment. Be as specific as possible including names and model numbers, labeling connections etc. Was your hypothetical topology from the study guide correct? If not, explain why.



My hypothetical topology from the study guide was correct.

17. Through this lab you utilized a file named `MakeXbowlocal`, what is the purpose of this file? What other settings besides the frequency can be changed here?

The MakeXbowlocal file when included in the Makefile enables the user to effortlessly edit the desired transmitting frequency of the motes. Other settings that can be edited here are the network group number and the transmitting power.