Lab2: Sensor Network Applications

Introduction

The goal of this laboratory is to familiarize the student with different types of software applications available for use in sensor networks. Upon completion of this assignment one should have a good understanding of applications developed by commercial vendors and the open source community, furthermore understand their dissimilarities.

Equipment needed from the cage: Mote Sensor Kit and serial cable

I. Blink++

1. Modify the ***Blink*** application in Lab2 to display the lower three bits of a counter in the LEDs. The counter should update every 2 seconds.

Hint: a. L*esson 1* of the tutorial under TinyOS directory.

b. Utilize the graphical representation of the program structure generated by "make mica2 docs" to help you find necessary API/functions.

II. Surge Reliable

2. The hardware manufacturer Crossbow provided a set of tools ready to use with the utilized equipment. These tools have been thoroughly tested and are quite easy to implement. In this set of activities we will focus on an application named Surge Reliable. This nesC code was developed to provide simple out-of-the-box mesh networking for the MICA2 motes and the associated programming board. The above

application works in conjunction with Surge-View which provides a graphical representation of a particular network topology. The below activities will acquaint you with the necessary steps required to establish a functional sensor network. Once again you are encouraged to use online resources to familiarize yourself with nesC.

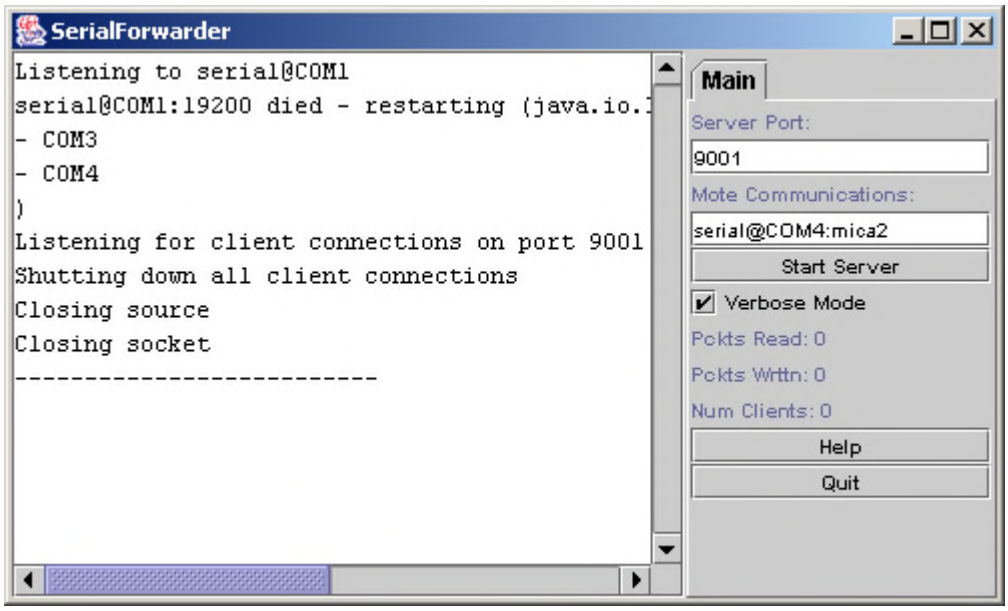
3. Browse to the Crossbow directory on the image server. Copy the entire folder named Surge-View into C:\Program Files on your machine. The programs present in this directory are all Java based and have already been precompiled and packaged for you.

4. Open the newly copied directory C:\Program Files\Surge-View. Find an executable named **SerialForwarder.exe** and double click it. A window named SerialForwarder should now be visible. Click on the button labeled **Stop Server.**

5. Ensure that the **Server Port** is set to 9001 by typing that value into the corresponding field. This is the default port on which the server will communicate and listen for messages.

6. The field labeled as **Mote Communications** must contain the communicating interface, COM port number and the type of device being used, for example:

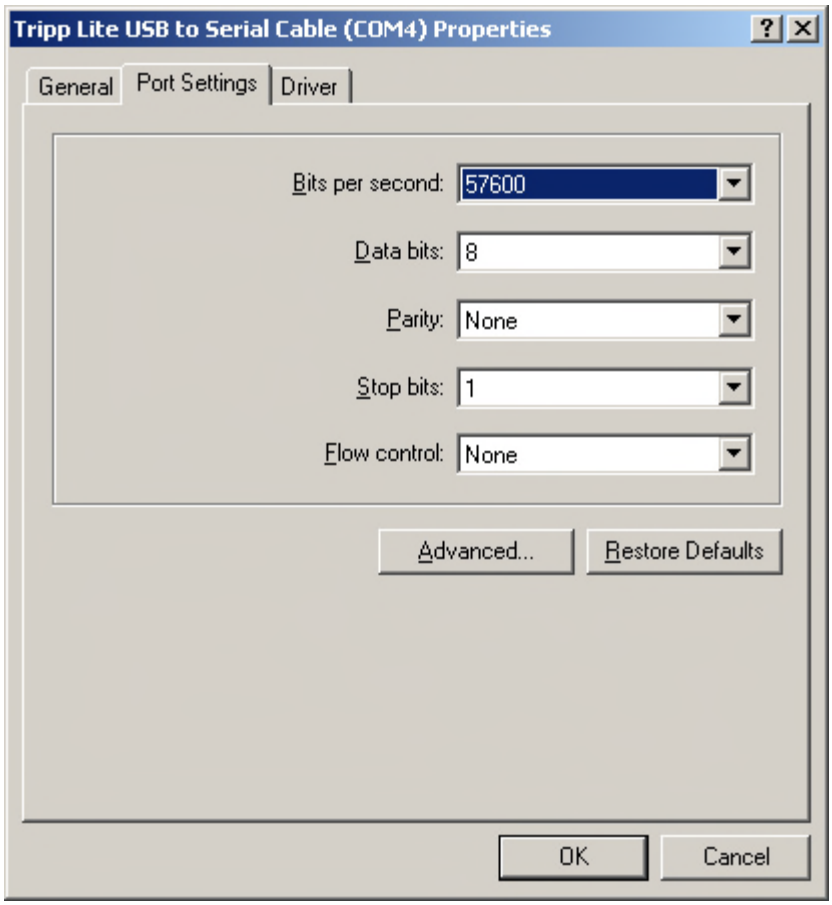
```
serial@COM4:mica2
```



Serial Forwarder GUI

7. To ensure accurate communication over the COM port of choice proper values must set in that port's properties. To accomplish this follow the below steps:

- Right click on My Computer
- Go to properties
- Click on Device Manager
- Browse the Ports(COM&LPT) category
- Look for the COM port of your choosing and right click it, then click properties
- Pick the Port Settings tab and adjust the setting to the below displayed values



Note: Same steps apply if a USB to serial converter is used

8. Place one of the MICA2 motes on the programming board. Be sure to follow the exact steps as described in Lab 2. This mote will become the "base" mote and must therefore be given the ID of 0. With the mote securely fastened to the MIB510 perform code uploads by following the below steps.

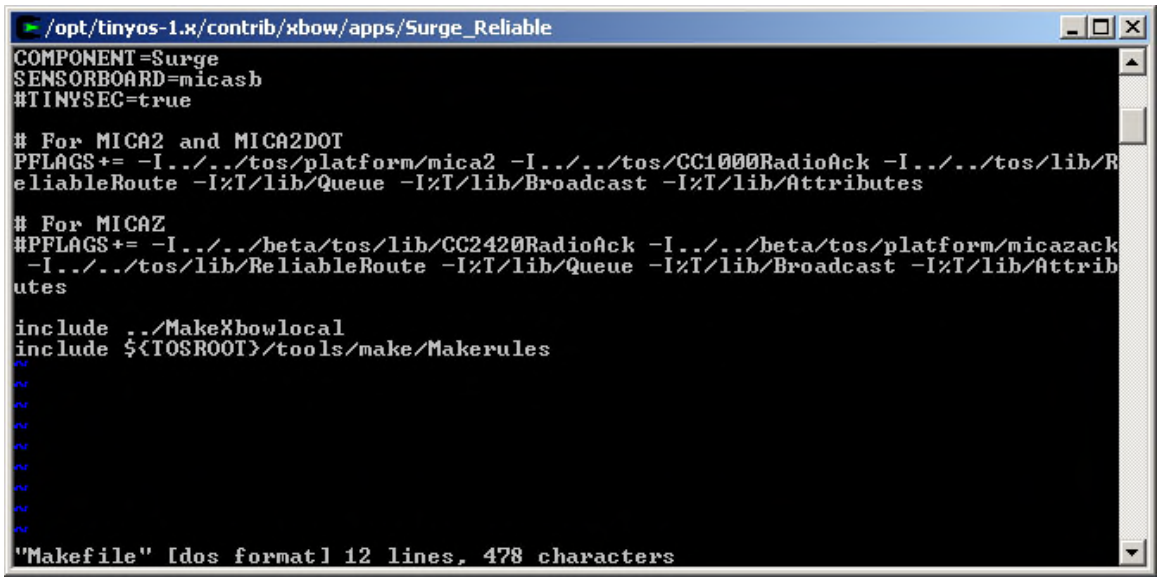
- Start a Cygwin session, and change to `opt/tinyos-1.x/contrib/xbow/apps`
- Edit the MakeXbowlocal file based on the frequency you will be using (i.e. only uncomment a line like **`CFLAGS = -DCC1K_DEFAULT_FREQ=CC1K_433_002_MHZ`**).

- Edit the MakeXbowlocal file to alter the group ID for your network, locate the below line and change the hex number to your bench number, converting it to hex of course:

```
DEFAULT_LOCAL_GROUP=0x7
```

Note: This option will ensure the separation of mote networks running on the same frequency and channel within the room. All 8-bit values are acceptable, except 126 and 255. Can you find out what those are reserved for?

- Change to `opt/tinyos-1.x/contrib/xbow/apps/Surge_Reliable` open the `Makefile` and ensure the below displayed settings:



Note: Ensure that the proper **PFLAGS** statement is uncommented based on the mote type used; the above example includes **MICA2** and **MICA2DOT**. Check that the **COMPONENT** is set to Surge as this is application that will be used.

SENSORBOARD should equal **micasb** as that is the type of board used.

TINYSEC is a security protocol that can be deployed only if your network does not include **MICA2DOT** motes, which will not be able to join the topology while this protocol is enabled. Remember that each application has its own Makefile.

- While in the `opt/tinyos-1.x/contrib/xbow/apps/Surge_Reliable` directory compile the application by typing:

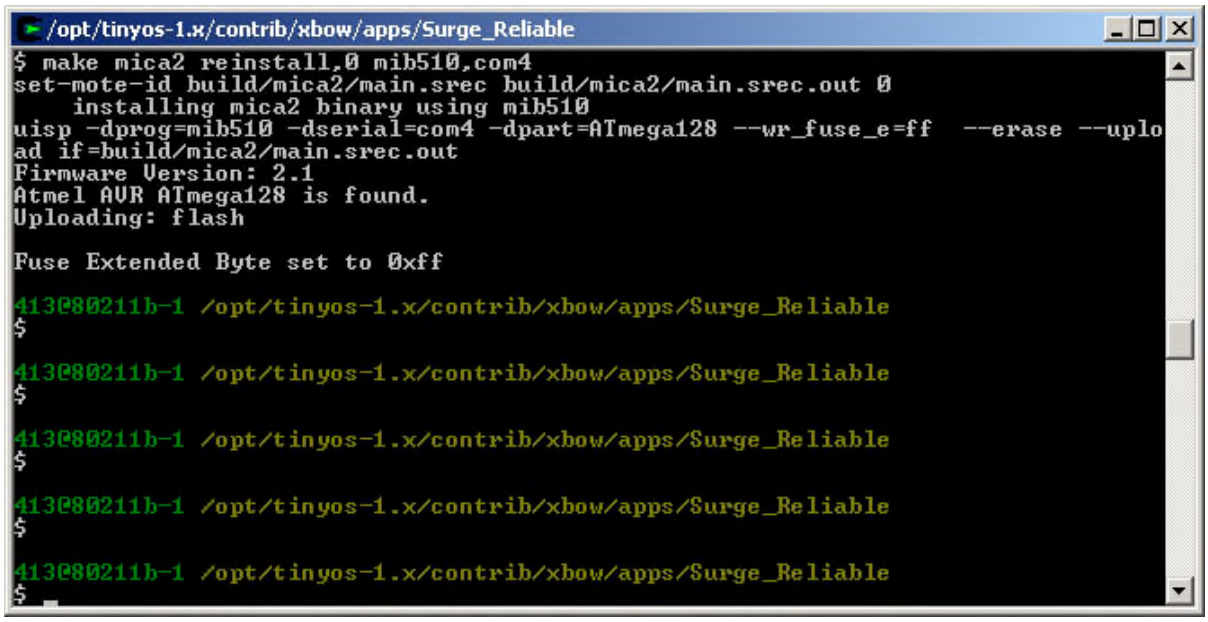
```
make mica2
```

- Install the application on each one of the motes, increasing the number following **`reinstall`** for every mote except the last one, which will become the base mote and needs to have an ID of 0.

```
make mica2 reinstall,1 mib510,comX
```

Note: The reinstall option is significantly faster as it does not recompile the code each time it is uploaded, if you require to compile each time use install instead. Remember the X is the number corresponding to the COM port used for the MIB510.

- Follow the same procedure for all other MICA2 motes. Remember to remove the batteries and keep the power switch in the OFF position for each mote. The output should be similar to the following:



9. The last mote you programmed should stay on the MIB510 board, as it will be responsible for signal reception; ensure that the ID of this device is set to 0.

10. Install the MTS310 sensors on each one of the remaining motes. Be sure the sensors are firmly attached.

11. Replace batteries in all MICA2 motes containing sensors and turn the power switches to the ON position. Do not place any batteries in the mote on the programming board.

12. Return to the Serial Forwarder GUI and click on the button labeled **Start Server**. If your motes are properly setup and turned on, the file named **Pckts Read** should now be displaying increasing values (Be sure the switch on MIB510 board is in OFF position).

The **Num Clients** filed should also have a value. Save a screenshot for your report.

13. Obtain a command prompt window by clicking Start>Run then typing **cmd**. Browse to the `Surge-View` folder, which by default should be in `C:\Program Files`.

14. While in the `C:\Program Files\Surge-View` directory type the following command to execute the GUI displaying a representation of the network topology:

```
Surge 125 > log_file
```

`Note:` Be sure that the number following the program name **Surge** is the decimal representation of the hex group ID you have setup in the MakeXbowlocal file (In this example, your group ID is 0x7D). The name of the log file is irrelevant, although it should be given a meaningful name so it can easily be referenced at any time.

15. You should now be able to see two windows. One will display the topology of the network while the other will show vital statistics. Click on the individual nodes within the topology window to gain access to more information. You also have the ability to move nodes around and arrange them to your liking.

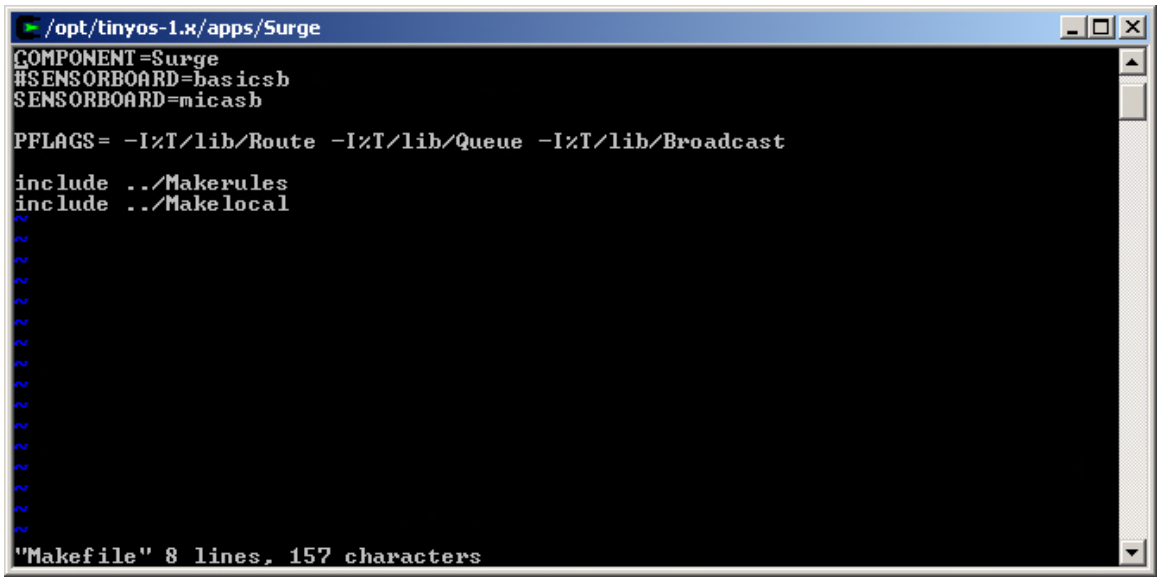
III. Surge

16. Until now all the experiments have been conducted utilizing applications thoroughly tested and modified by the hardware manufacturer Crossbow. It is now time to move to software created and maintained by the Opensource community. To investigate and clearly illustrate the differences between the two, Surge an application similar to Surge Reliable will be explored. **It is very important** that you note the differences between these two approaches. While one is easier to use and understand the other offers the ability for greater customization. The later approach will be the on which you will be required to use for the remainder of this course.

17. Once again prepare the required motes and connect the programming board to the PC. Choose one MICA2 which will become the base mote and remove all batteries ensuring the power switch is in the OFF position.

18. Change to `/opt/tinyos-1.x/apps/Surge.` If you have setup the aliases suggested in Lab 1 you can type `cdtinyos`, then cd `apps/Surge.`

19. Using the vi editor open the **Surge** `Makefile`. Ensure the below settings:



Note: It is important the `COMPONENT` variable is set to `Surge` and the `SENSORBOARD` variable is set to `micasb`. It is also imperative to verify the presence of the two `include` statements. These statements provide a search path for **TinyOS** during compile time, specifying any optional settings needed.

20. Copy the `MakeXbowlocal` file from `opt/tinyos-1.x/contrib/xbow/apps` into the `opt/tinyos-1.x/apps` directory and rename it to `Makelocal`.

21. Edit the above file based on your needs ensuring a network ID and frequency are specified. At compile time this file will be included and override TinyOS default settings for frequency and network ID.

22. While in the /opt/`tinyos-1.x/apps/Surge` directory compile the application

by typing `make mica2`, then install it on each one of the motes remembering to set an ID number for each mote and ensuring that the base mote has an ID of 0. The syntax for accomplishing the above is slightly different and should be similar to the following:

```
MIB510=COM4 make reinstall.0 mica2
```

Hint: Type `reinstall` instead of `install` to avoid compiling each time the code is uploaded to a mote. Also, try uploading to the base mote last as it will be the only one that remains on the programming board.

23. Upon completion start the **SerialForwarder** application by starting a **Cygwin** session, changing to /opt/`tinyos-1.x/tools/java` and typing the following:

```
java net.tinyos.sf.SerialForwarder –comm serial@COM4:mica2
```

24. Replace the batteries in all other MICA2 motes, ensuring that each mote is fitted with a sensor board. Remember to turn the motes ON.

25. Start another Cygwin session and change to the /opt/`tinyos-1.x/tools/java` directory, then start the Surge GUI by typing the following:

```
java net.tinyos.surge.MainClass 129
```

Note: Remember that the last number specifies the network ID that was included in the `Makelocal` file.

26. A graphical representation of the network topology should materialize shortly. Try to alter the readings for each node and watch the graphical display.

VI. Oscilloscope

27. Using what you have learned so far, implement a functional sensor network. Use the applications OscilloscopeRF and TOSBase and whatever other tools necessary to complete this assignment.

Note: When OscilloscopeRF is used, whenever a packet is transmitted each mote will blink its yellow LED. Change that option, so when a packet is sent the green LED will blink.

Hint: Type `java net.tinyos.oscope.oscilloscope` to invoke the application GUI, and change the position of the slider.

Laboratory Report

Instructions

The questions are to be answered based on your lab experience. The report questions must be accurately answered, typed, diagrams computer generated and this template used. Include the questions on your report.

I. Blink++

1. Paste your source code for the counter and counter displaying with necessary comments.

II. Surge Reliable

2. In activity number 4 you were asked to use a java application SerialForwarder, in your own words describe the purpose of this program.
3. When choosing the group ID for your mote network you were instructed not to use two values. What are those values and what purpose are they reserved for?
4. Provide a screenshot of the working SerialForwarder application you started in activity 12. What do the fields Pckts Read and Num Clients represent?
5. Provide a screenshot of the network topology as displayed in Surge Reliable.

III. Surge

6. Up to this point you have been compiling programs and uploading applications to motes. Do some research and find the command that will remove the already built binary files of an application. Provide the command below and briefly explain what it exactly does.
7. Provide a screenshot of the network topology as displayed in Surge.
8. When looking at the Surge window, what happens when the debug options is checked? Provide a screen shot and explain the results.
9. Describe in your own words what the above applications are accomplishing (Surge_Reliable and Surge).

VI. Oscilloscope

10. Describe in your own words what the OscilloscopeRF application is trying to accomplish.
11. When you complete activity 27, provide a screenshot of the Oscilloscope GUI. Can you alter the sensor readings?
12. In your own words describe the difference between a “base” mote and a “normal” mote.