

Results

# multihoposcilloscope

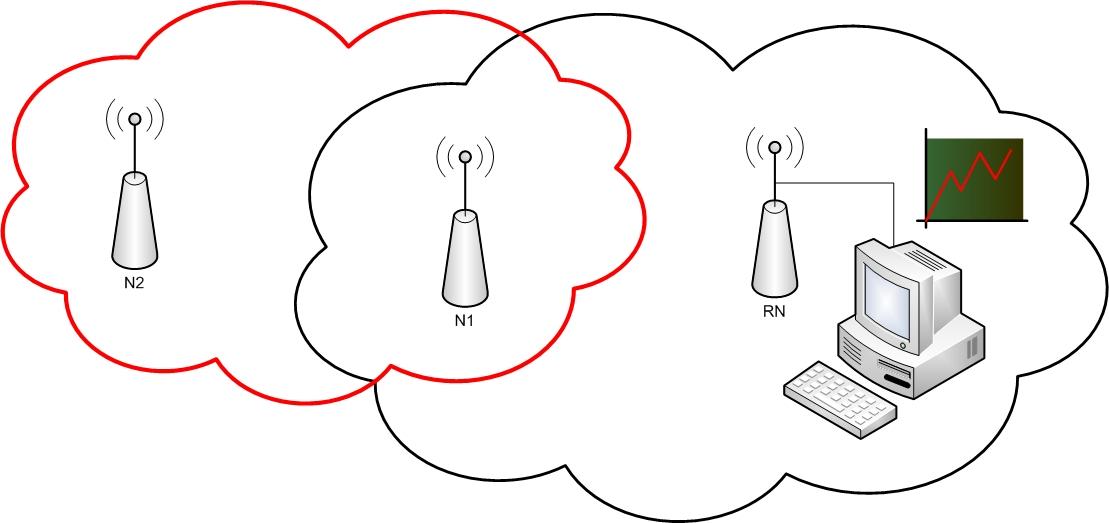
Our first application is called multihoposcilloscope. Basically it is an application which implements multihop routing. It makes use of 1 sensor. That data is being transmitted to the root node in the network.

## Test Situation

We have 3 wireless TelosB sensors with an assigned ID of 10(N1), 11(N2) and 500 (RN).

The TelosB has the following specifications about range:

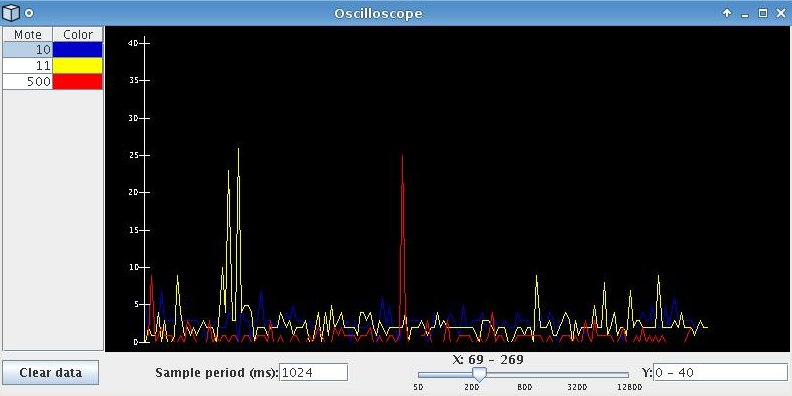
|  |  |
| --- | --- |
| range (outdoor) | 75m to 100m |
| range (indoor) | 20m to 30m |



We have tested the application in an apartment building, so the range was limited. The root has 500 as ID, so it normally receives all transmitted packets from the nodes. Node 11 was dropped 2 floors lower than the root, so it was out of range. We placed mote 10 one floor lower, so it is in range of the root and node 11.

## result

On the graphical java GUI we can see that all the nodes were detected by the root, so the multihop network is successful.



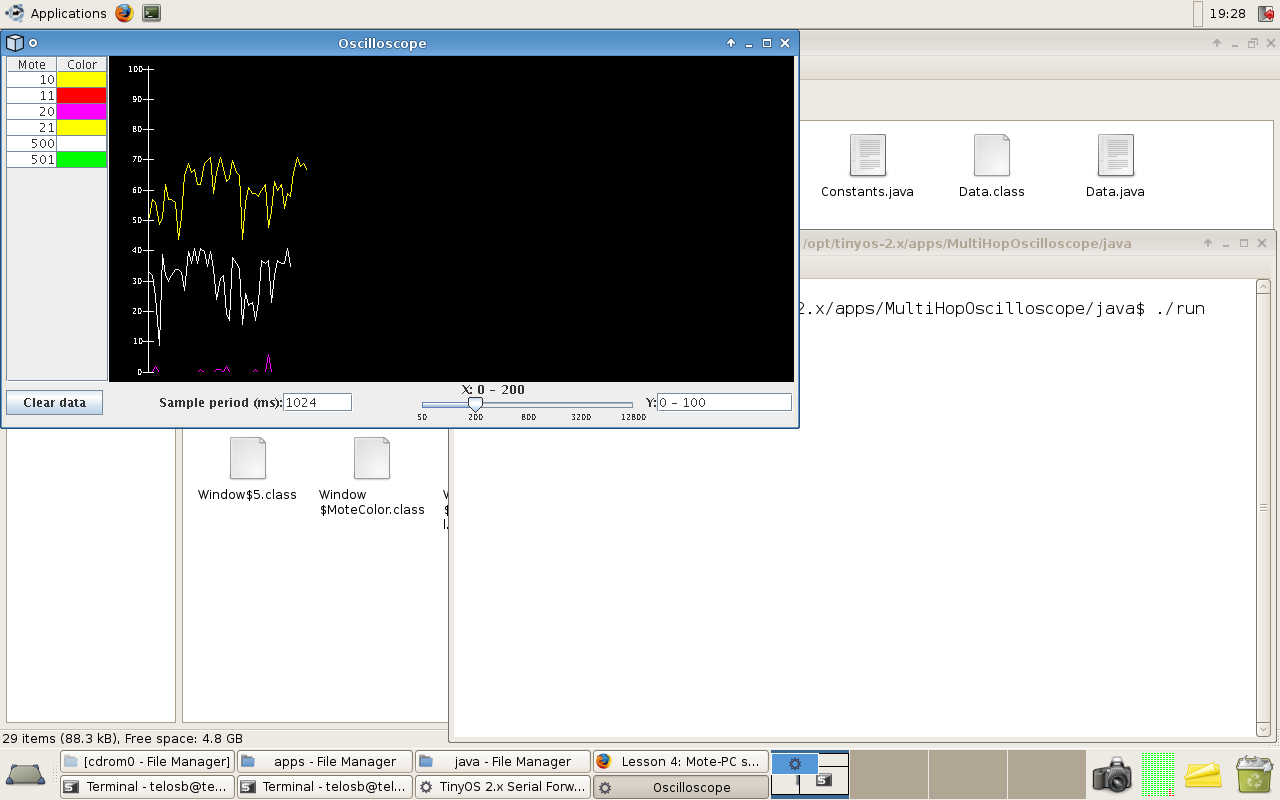
# multi hop/sens oscilloscope (peter)

The multihop/sens oscilloscope application is an extension to the multihoposcilloscope application. It implements 2 readings from 2 different sensors (light & voltage). It is compatible with the Java GUI supplied by TinyOS.

## test situation

We have 2 wireless TelosB sensors. Each node has 2 different ID’s, one for each sensor. So, the root node(RN) has the ID’s 500 and 501. The second node(N1) has the ID’s 10 and 11. Note that the values from RN are not correct. This is because the mote has no sensors except for a voltage sensor. The data it sends is therefore erroneous.

## results

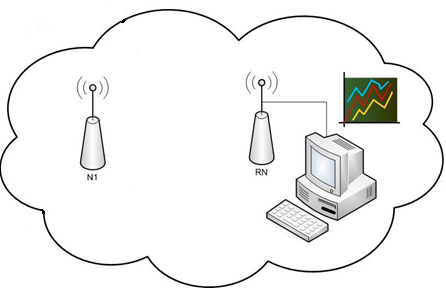


# multi hop/sens oscilloscope (TIM)

The multihop/sens oscilloscope application is an extension to the multihoposcilloscope application. It implements 2 readings from 2 different sensors (light & voltage). It is compatible with the Java GUI supplied by TinyOS.

## test situation

We have 2 wireless TelosB sensors. Each node has 2 different ID’s, one for each sensor. So, the root node(RN) has the ID’s 500 and 501. The second node(N1) has the ID’s 400 and 401.



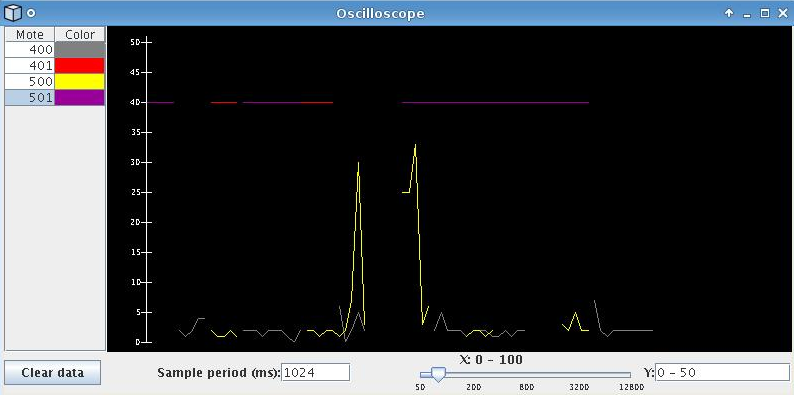
## results

We have added the printf library to help us to debug the program. The log:

|  |
| --- |
| start of readL!  start with the measurement of the lightsensor  measurement 1 of the light: 2  start of readL!  start with the measurement of the lightsensor  measurement 2 of the light: 2  start of readL!  start with the measurement of the lightsensor  measurement 3 of the light: 2  start of readL!  start with the measurement of the lightsensor  measurement 4 of the light: 2  start of readL!  start with the measurement of the lightsensor  measurement 5 of het light: 2  NEXTREADINGREADY is TRUE  There are 5 readings of the lightsensor  The send is free for the transmission of the lightmeasurements  transmitting of the measurements of het light is successfull  lightmeasurement is transmitted with mote ID: 500  start of readD!  start of readD!  start with the measurement of the themosensor  measurement 2 of the themosensor: 40  start of readD!  start with the measurement of the themosensor  measurement 3 of the themosensor: 40  start of readD!  start with the measurement of the themosensor  measurement 4 of the themosensor: 40  start of readD!  start with the measurement of the demosensor  measurement 5 of the themosensor: 40  There are 5 readings of the demosensor  The send is free for the transmission of the themomeasurement  transmitting of the measurements of the demo is successful  demomeasurement is sent with mote ID: 501 |

We can conclude that the application sends the readings of the 2 sensor concurrently.

Next step is checking if we can display the data on a graph:

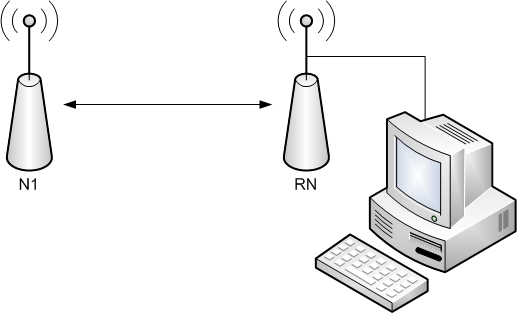


We can conclude that the application works fine.

# RSSI

## test situation

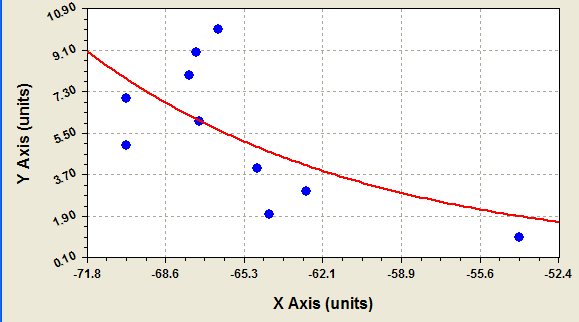
We have 2 nodes: one (RN) connected to a computer to get the RSSI and one(N1) as a reference at certain distances.



## results

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Distance (m) | Measurement 1 | Measurement 2 | Measurement 3 | Measurement 4 | Measurement 5 | Average | to Dbm |
| 1 |  |  |  |  |  |  | -51 |
| 2 | -31 | -28 | -17 | -19 | -28 | -24,6 | -69,6 |
| 3 | -24 | -10 | -23 | -13 | -10 | -16 | -61 |
| 4 | -19 | -26 | -21 | -26 | -17 | -21,8 | -66,8 |
| 5 | -25 | -33 | -32 | -28 | -28 | -29,2 | -74,2 |
| 6 | -18 | -29 | -22 | -21 | -22 | -22,4 | -67,4 |
| 7 | -25 | -21 | -25 | -30 | -31 | -26,4 | -71,4 |
| 8 | -16 | -19 | -29 | -27 | -25 | -23,2 | -68,2 |
| 9 | -23 | -23 | -23 | -24 | -28 | -24,2 | -69,2 |
| 10 | -25 | -32 | -21 | -17 | -11 | -21,2 | -66,2 |

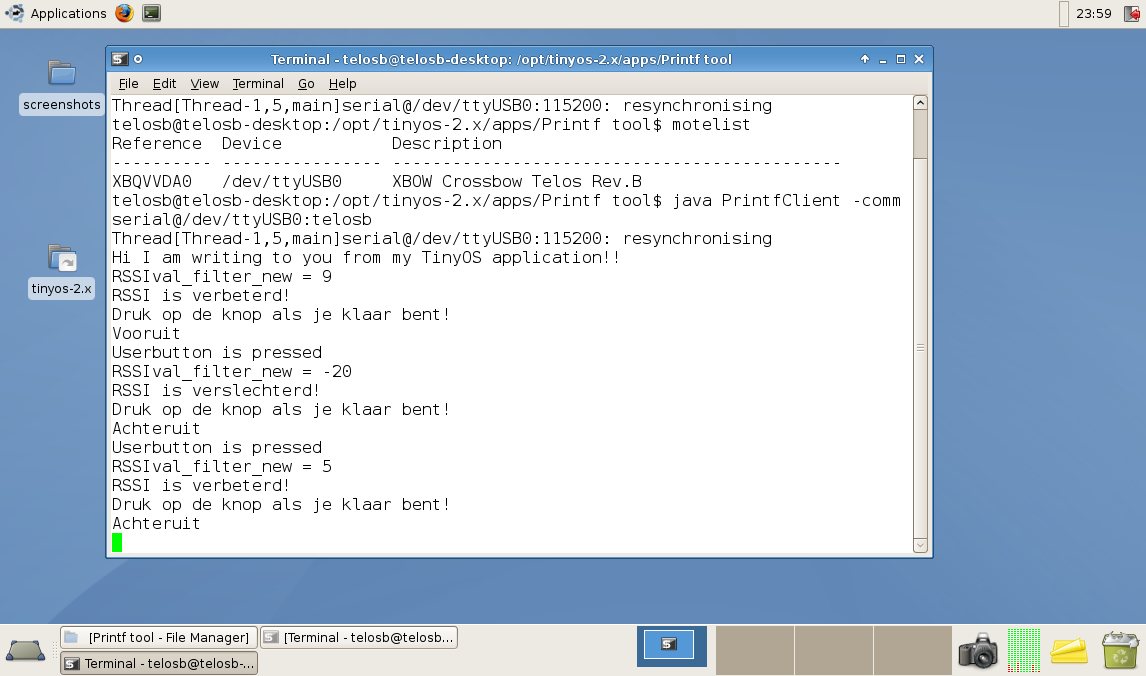
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Distance (m) | Measurement 1 | Measurement 2 | Measurement 3 | Measurement 4 | Measurement 5 | Average | To dBm |
| 1 |  |  |  |  |  |  | -51 |
| 2 | -19 | -12 | -15 | -13 | -11 | -14 | -59 |
| 3 | -22 | -25 | -19 | -16 | -16 | -19,6 | -64,6 |
| 4 | -18 | -16 | -16 | -18 | -21 | -17,8 | -62,8 |
| 5 | -16 | -20 | -30 | -20 | -20 | -21,2 | -66,2 |
| 6 | -20 | -24 | -24 | -24 | -18 | -22 | -67 |
| 7 | -26 | -25 | -22 | -27 | -20 | -24 | -69 |
| 8 | -23 | -21 | -22 | -22 | -22 | -22 | -67 |
| 9 | -22 | -37 | -1 | -23 | -19 | -20,4 | -65,4 |
| 10 | -20 | -20 | -20 | -22 | -26 | -21,6 | -66,6 |



The equation for this graph is: y = a(1-e-bx) with

* a = -0.017352956
* b = 0.087152976

We get the following output from the PrintfClient:

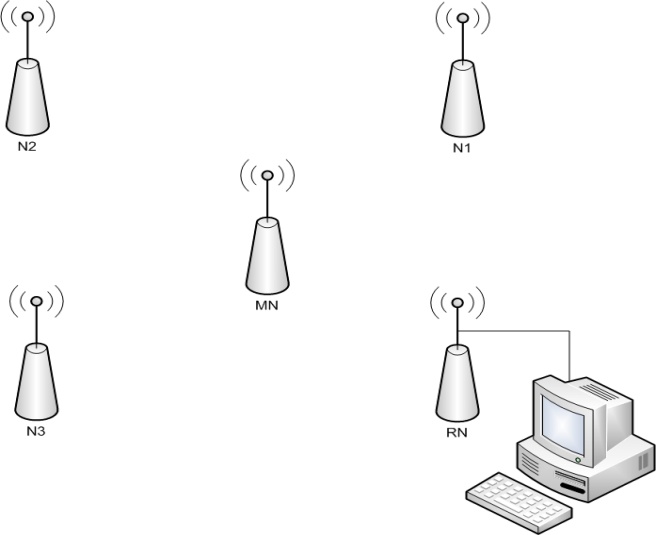


# visualisation with rssi

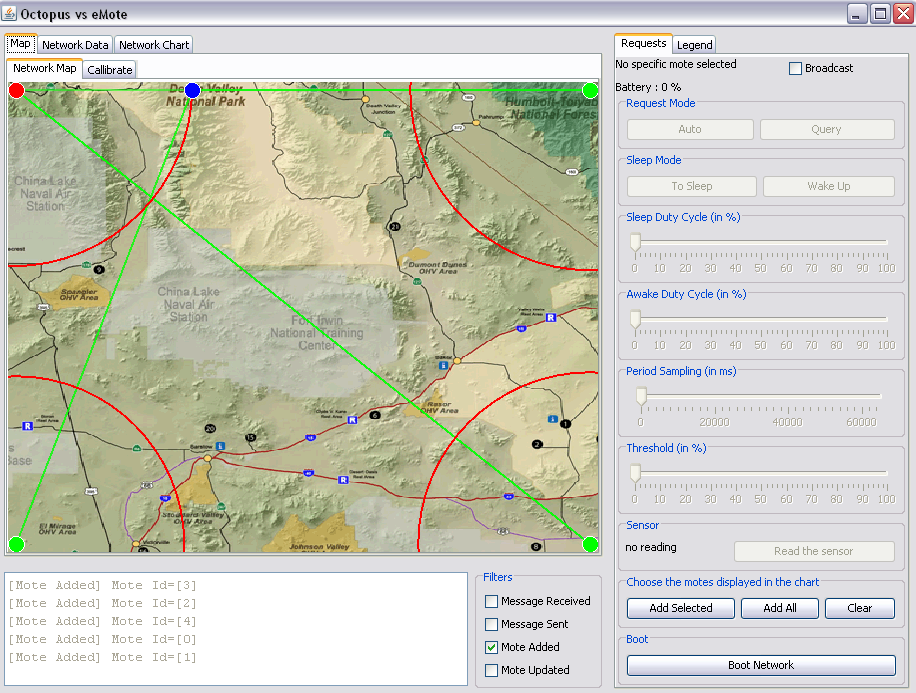
This application is compatible with the Java GUI of the master students Nick and David. So, all the nodes can be visualized on a computer. The benefit of this application is that it directs u to an alarming node in the network through the printf interface.

## test situation

We build a network with 5 nodes: 1 acts as the root node (RN) and is connected to a computer, nodes N1, N2 and N3 function as anchor points, so they are reference points in the network. The last node is a mobile node (MN) and its position is localized by RSSI from the anchor points.



## results



The blue dot is our mobile node, the green ones are our anchor nodes and the red dot is our root node. We can see that all the nodes are visualized.