Octopus: User Documentation

Authors: Raja Jurdak, Antonio Ruzzelli, and Samuel Boivineau

Description:

Octopus is a visualization and control tool for Wireless Sensor Networks. It is composed of an embedded program for the mote, and a GUI in Java for the computer.

The features of Octopus include:

- Designed specifically for TinyOS 2.x
- Successfully tested with MicaZ, Mica2, Telos, and AquisGrain motes
- IEEE 802.15.4 friendly
- Allows control of one, many, or all nodes at a time
- Enables user to switch between Time-driven, Event-driven, and Query-driven modes
- Enables user to switch between Sleep and Awake modes
- Enables user to select separate radio duty cycles for Sleep and Awake modes
- Provides live data plots through the Network Chart feature
- Provides a variety of options for controls the appearance and tags in the GUI
- Employs an embedded application running on the motes (NesC), and a java application running at the gateway
- Logs all received data in a .csv file for further analysis

Tools:

The java directory contains a GUI for the Octopus app.

Usage:

The following instructions will get you started with the Octopus application (the instructions are for micaz motes, replace micaz with your platform)

1. Compile the root and node code for the antitheft application for your platform :

```
$ (cd motes; make micaz)
```

2. Install the code on a root mote with ID = 0:

```
$ make micaz reinstall.0 <your usual installation options>
# For instance: make micaz reinstall.0 mib510,/dev/ttyS0
```

3. Install the code on some number of micaz motes, giving each mote a distinct id.

```
$ make micaz reinstall.N <your usual installation options>
# For instance: make micaz reinstall.22 mib510,/dev/ttyS0
```

4. Connect the root micaz mote to your PC and switch on all motes.

5. Compile and run the java application. The text below assumes your serial port is /dev/ttyS0, replace with the actual port you are using (e.g., COM3 on Windows or /dev/ttyUSB0 on Linux)

```
$ cd java
$ make
$ export MOTECOM=serial@/dev/ttyS0:micaz
$ java OctopusGui
```

Configuration of the motes

Some options are available only before the compilation process. These options can be chosen in the file *"motes/OctopusConfig.h"*.

DEFAULT_SAMPLING_PERIOD

This is the default value of sampling period, when the mote is started. This value is in milliseconds.

MINIMUM_SAMPLING_PERIOD

This is the minimum allowable value in the network, which is broadcast to the nodes to avoid congestion issues. For example, a heuristic formula that can be used is, with *N* the number of motes of the network, and *MINIMUM_SAMPLING_PERIOD* in ms:

$$MINIMUM\_SAMPLING\_PERIOD = 100 * N$$

DEFAULT_THRESHOLD

This variable provides for event-driven applications. The threshold indicates the percentage change of the sensed value that will trigger the mote to send a data packet towards the gateway. DEFAULT_THRESHOLD is the default threshold of the sensor. The threshold can take any value between 0 and 100 %. A threshold of 0 means that the value is sent every time, resulting in a time-driven application. The threshold is a 16-bits unsigned integer.

DEFAULT_MODE

The user can choose either the automatic (timer-driven mode) *MODE_AUTO* or the manual (query-driven mode) *MODE_QUERY*, in which nodes only report sensed values in response to user queries.

DEFAULT_SLEEP_DUTY_CYCLE & DEFAULT_AWAKE_DUTY_CYCLE

The user can define the sleep duty cycle and the awake duty cycle in units of [percent * 100]. For example, to get a 0.05% duty cycle, use the value 5, and for a 100% duty cycle (always on), use the value 10000. This is the equivalent of setting the local sleep interval explicitly.

The Console Panel
This panel is located in the bottom with some checkboxes to filter the messages displayed.

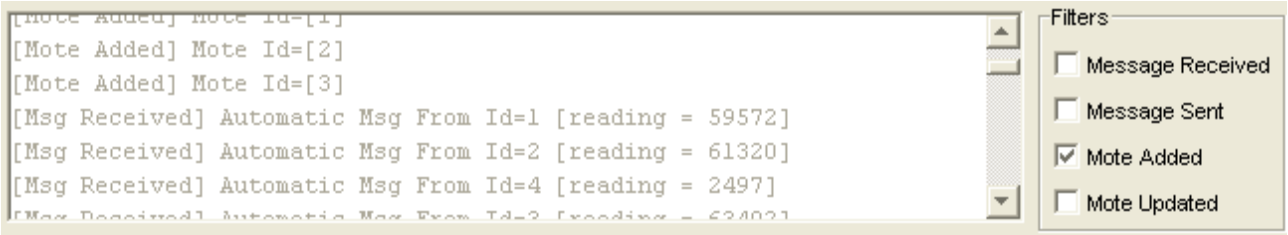
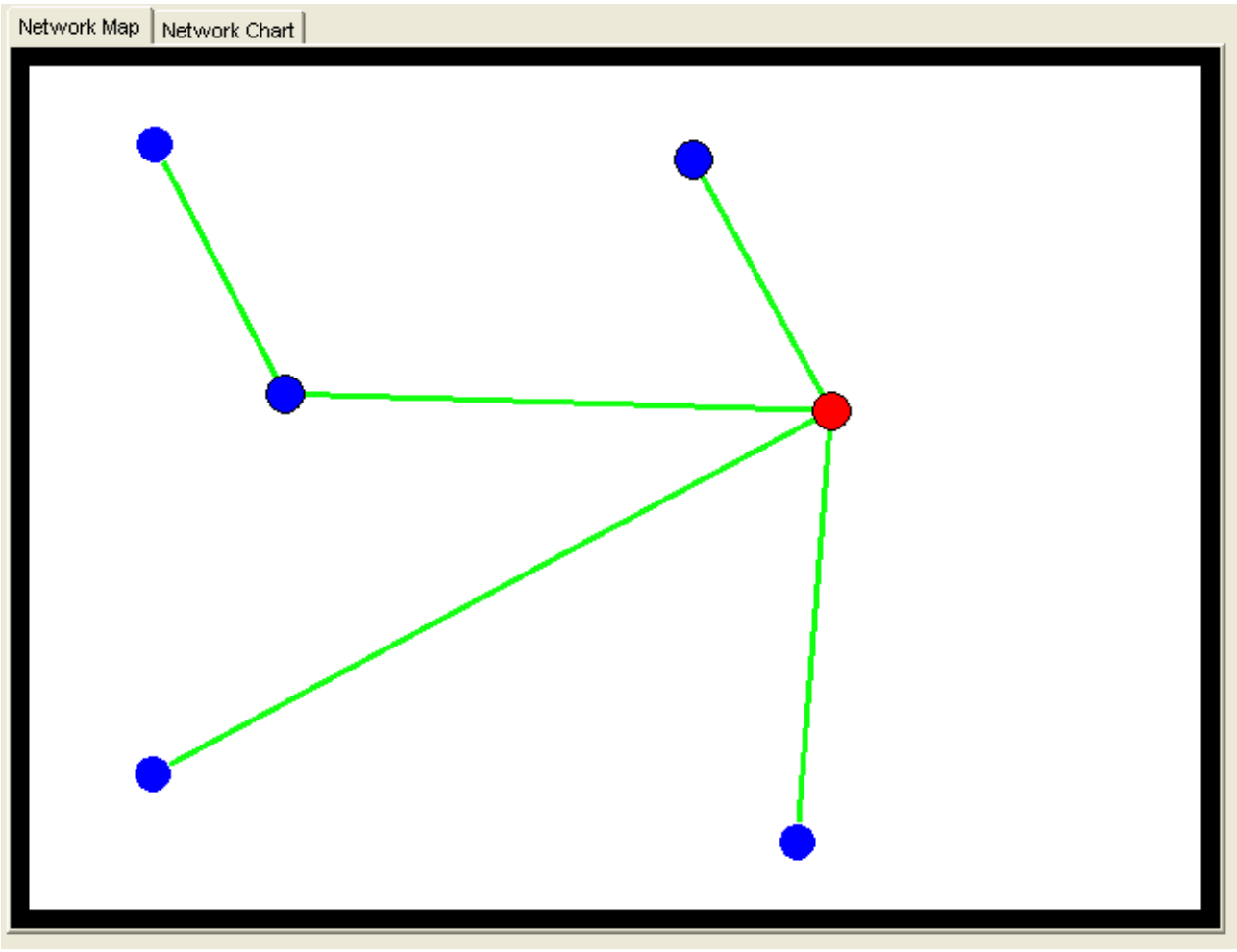


Figure 1: Console Panel with filters

You can choose the messages displayed, between the messages received, sent, and when a mote is added or updated. A header is added at the left of the message.

Map Panel
In the center, two panels are available through tabs. The first one is the map of the network. The gateway appears in red and regular motes appear in blue. The coordinates of the motes are chosen randomly, as the map only shows the logical topology of the network. The route between a mote and its parent is indicated in green if the link quality is good. If after a while, the mote has not sent any packets, the route to its parent starts to fade away. This value of the timeout (the time after which a node is considered to have left the network) can be modified.



The motes can be moved and selected by clicking on them. When a mote is selected, a black border appears around of this mote.

The Chart Panel

This panel is available through the tabs. In order to view a chart of some motes in this panel, firstly you need to select a mote by clicking on it. Then you click on the button "Add selected" of the right panel called "Request":

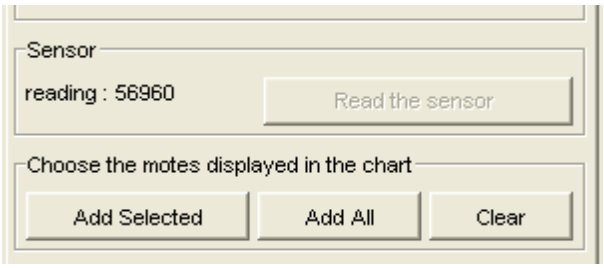


Figure 3: Bottom of the Request Panel

The data coming from the motes is now recorded dynamically by Octopus. It means that the data displayed is only the data received once the button has been clicked, and that after a long time, the application will grow in memory size (RAM) if you let this option on.

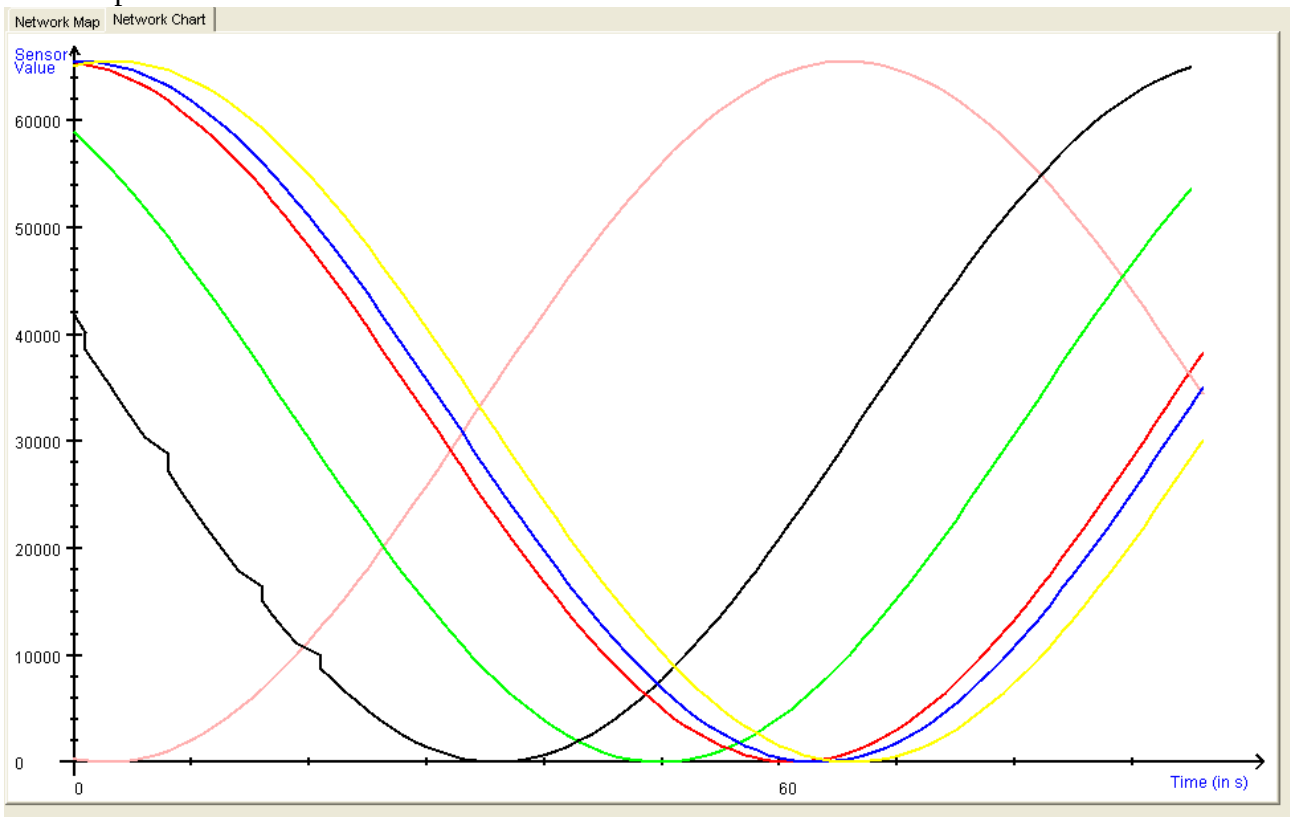


Figure 4: Chart Panel with data coming from motes

If you want to add all the motes of the network in the chart, you can click on the button "Add All". To remove motes from the chart, you can click on the button "Clear".

Request Panel

This panel is composed of many parts. On the top appears the ID of the mote selected, with a checkbox to send broadcast requests, and the battery level (not supported yet in TinyOS 2.x).
Below that there are 4 buttons to set the mote in a particular mode. The mode auto corresponds to a timer-driven mode, and the mode query is a query-driven mode. The two other buttons are used to force a mote to sleep (to turn off its timers) or to wake up a mote. A mote that is in sleep mode only replies to a status requests or wake up requests.

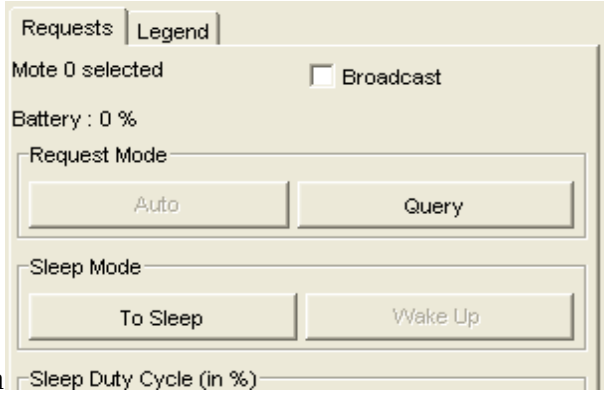
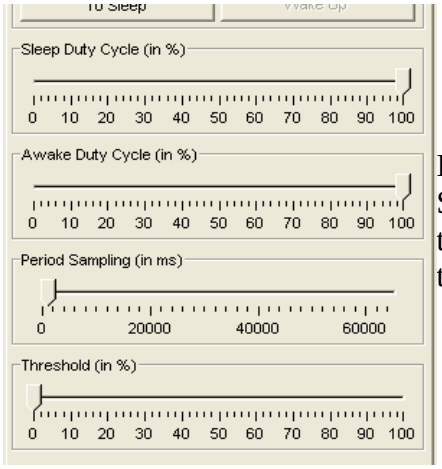


Figure 5: Top of the Request Panel



In the middle, some sliders are available to change the values of Sleep and Awake Duty Cycle, of the period sampling, and the threshold. The request is sent only when the mouse has released the slider.

Figure 6: Middle of the Request Panel

At the bottom of the Request Panel, there is a button to request a reading in case the mote is in query mode. The three buttons to add selected, add all, or clear mote reading in the chart are also available there.

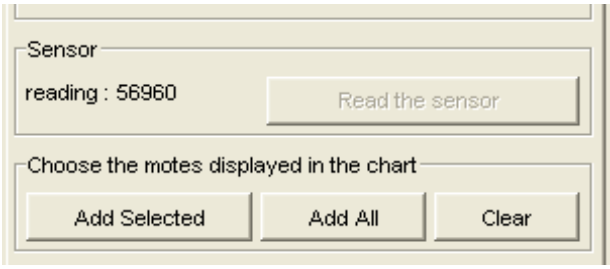


Figure 7: Bottom of the Request Panel

Legend Panel

The last panel is the legend panel which lets the user choose the drawing of the map. When a new legend is choosen, the map is automatically updated

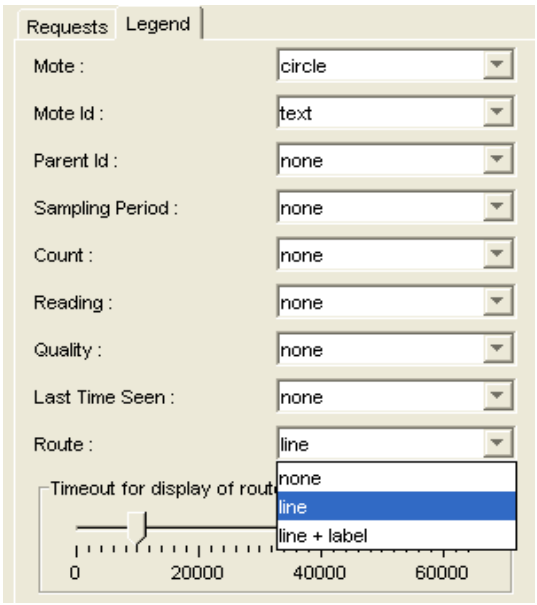


Figure 8: Top of the Legend Panel

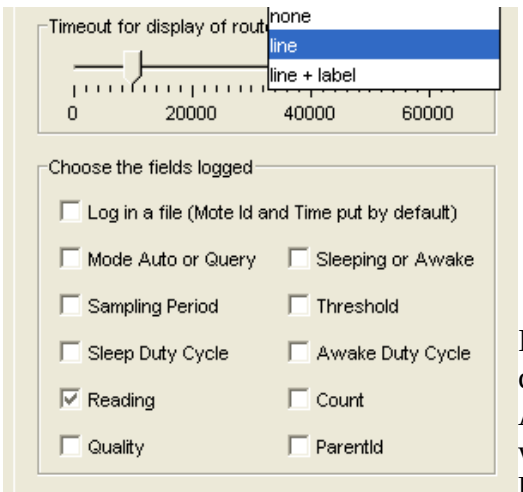


Figure 9: Bottom of the Legend Panel

In the middle of the Legend Panel, there is a slider to choose the timeout before a route fades away on the map. At the bottom, there are also some checkboxes to choose which fields will be recorded by Octopus in a csv file for later processing. The file is created when the checkbox "Log in a file" is checked, and the filename is "Octopus_Log" followed by the current date and time.

Use the csv file to create a graphic

To use the csv file created by Octopus, you can use OpenOffice Calc, or Excel or any spreadsheet software. You first need to open the file and sort the data. The first colomn is always the MoteId, and the second is the time, so use the "Sort" option usually in the "Data" menu to sort the data by the colomn A and next B. The data should appear this way :

Next you can visualize the data in a chart by simply selecting the columns time and any that you are interested in, and by using XY graphics and choosing the time in X.

	A	B	C
1	Mote Id	Time in ms	Reading
2	0	593	20516
3	0	1593	22051
4	0	2593	23612
5	0	3593	25196
6	0	4593	26798
7	0	5593	28416
8	0	6640	30045
9	0	7593	31680
10	0	8593	33318
11	0	9593	34955
12	0	10593	36587
13	0	11593	38208
14	0	12593	39817
15	1	281	26798
16	1	1281	28416
17	1	2265	30045
18	1	3281	31680
19	1	4281	33318
20	1	5281	34955
21	1	6265	36587
22	1	7265	38208
23	1	8265	39817
24	1	9265	41407
25	1	10265	42976
26	1	11265	44520
27	1	12265	46034
28	1	13265	47515
29			

Figure 10: Csv file sorted by MoteId and next by Time

Known bugs/limitations:

- A newly turned on mote may not send theft reports (when the "Server" theft report option is chosen), as:
  - o It takes a little while after motes turn on for them to join the multihop collection network (a function of the underlying topology management protocol)
  - o It can take a little while for motes to receive the current settings.

- Octopus is slowing down when:
  - o A lot of motes are used and sending reading with a small sampling period.
  - o Many motes are recorded dynamically for the chart, consuming a lot of CPU cycles for visualization

- The gateway is not sending requests or processing collected data when:
  - o A lot of motes are sending readings in a small time to the gateway, then the serial port is overused and GUI cannot send requests.

For questions or comments on Octopus, please contact: Raja Jurdak at rjurdak@ieee.org