SwissQM Manual

1 Introduction

SwissQM is a programming interface for wireless sensor network applications. Queries (requests to collect data from a wireless sensor network) can be composed by users in the graphical interface of SwissQM. The composed queries are then compiled into programs that are sent to the sensor network where they are executed.

When you start the test, three windows will be open for you on the desktop, as shown in Figure 1. Please keep them open during the test. If you close any of these windows by accident, please ask your proctor to reopen it. In Figure 1, the **SwissQM Query GUI** window is the graphical user interface (GUI) of SwissQM, where queries are composed and submitted. The **Network** window shows simulation results, including sampled data at each node and data transmission events from node 0 to the base station. The **Base station** window shows query execution information and data collected at the base station. A network composed of four nodes is simulated in this test.

2 Working with the SwissQM Query GUI

The **SwissQM Query GUI** is used to compose queries, send them to the network, and plot the data collected at the base station. The GUI is composed of three tabs: the **Network** tab, the **New Query** tab, and the **Plot** tab. The **Plot** tab appears only after a query with plot requirement is submitted.

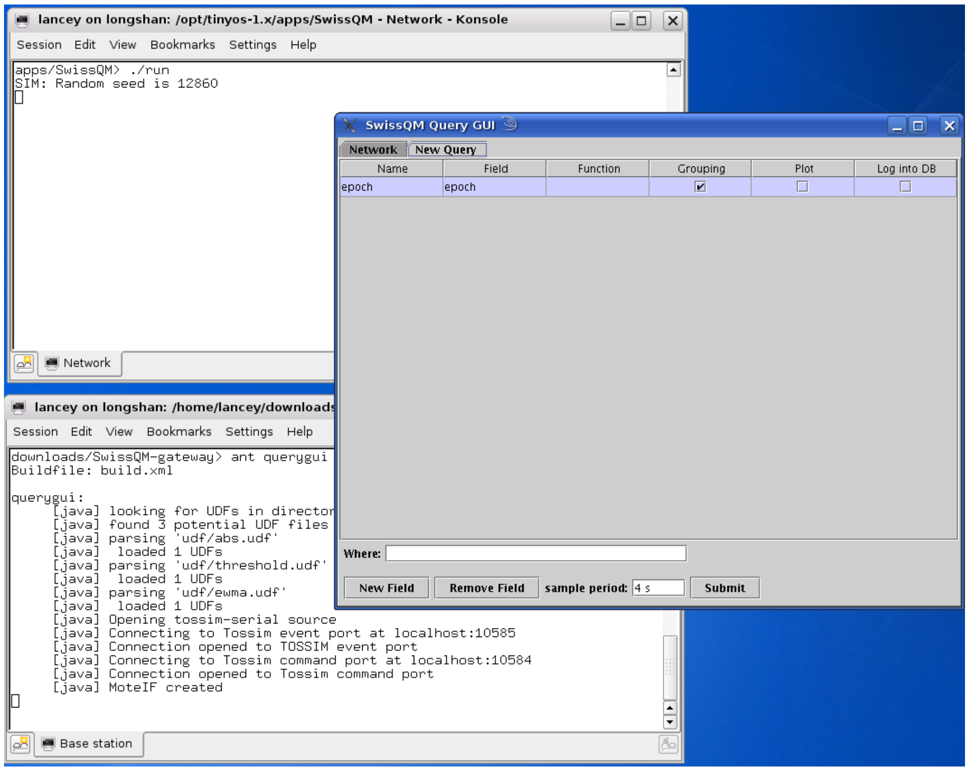
2.1 Network Tab

Figure 2 shows the **Network Tab**. The left column shows statistical data of communication between the base station and the sensor network. The right column shows the commands that can be sent to the network. You won't need to use the right column in this test, so you can ignore it.

2.2 New Query Tab

Queries are composed in the **New Query Tab**, as shown in Figure 3. The first field **epoch** is always present and cannot be modified. It is used as a coarse timestamp of the data. It counts how many sampling periods has elapsed since the start of the query. Query fields are added by clicking the **New Field** button. A new row is appended to the table when a new field is added. You can edit the attributes (columns) of the fields to compose a query. A query may contain multiple fields and new queries may be submitted while others continue to run. Explanations of the fields' attributes follow.

- **Name**: Identifier (no whitespaces) of that field. This identifier is shown in the legend of the result plot. The text can be edited directly.


lancey on longshan: /opt/tinyos-1.x/apps/SwissQM - Network - Konsole
Session Edit View Bookmarks Settings Help
Network
lancey on longshan: /home/lancey/downloads
Base station
SwissQM Query GUI
Network
New Query
Where:
New Field
Remove Field
Submit

```
apps/SwissQM> ./run
SIM: Random seed is 12860
```

Name	Field	Function	Grouping	Plot	Log into DB
epoch	epoch		☑	☐	☐

sample period: 4 s

```
downloads/SwissQM-gateway> ant querygui
Buildfile: build.xml

querygui:
     [java] looking for UDFs in director
     [java] found 3 potential UDF files
     [java] parsing 'udf/abs.udf'
     [java]  loaded 1 UDFs
     [java] parsing 'udf/threshold.udf'
     [java]  loaded 1 UDFs
     [java] parsing 'udf/ewma.udf'
     [java]  loaded 1 UDFs
     [java] Opening tossim-serial source
     [java] Connecting to Tossim event port at localhost:10585
     [java] Connection opened to TOSSIM event port
     [java] Connecting to Tossim command port at localhost:10584
     [java] Connection opened to Tossim command port
     [java] MoteIF created
```

Figure 1: SwissQM programming and simulation environment

- **Field**: Field expression describes data to be collected. Clicking on the field text box will show a drop-down list containing data that a sensor node can get. The field text box can be edited to compose an expression. An expression is composed of operands, operators, and parenthesis. An operator is one of + (add), - (subtract), * (multiply), and / (divide). An operand can be an item from the drop-down list or an integer. For example, (light + 100) * 2 is an expression that can be entered in the field text box.

- **Function**: Clicking on the **Function** text field will show a drop-down list containing all built-in functions. Descriptions of these functions are in Table 1. The selected function is applied to the **Field** column. When you select a function by clicking on it, it will be inserted to the **Field** column. Aggregation functions such as **min** and **sum** apply to field values from nodes in a same group. Sensor nodes are grouped with **Grouping** flag, which will be described later. The **Function** text box can be directly edited.

- **Grouping**: Set this flag if you want to divide nodes into groups according to data in the corresponding **Field** column. Nodes with the same value of the field will be grouped together. Grouping field cannot be plotted. If no grouping flag is set in a query, but aggregation functions are used, then the aggregation operations apply to all the nodes in the network.

- **Plot**: The value of the field will be plotted if it is checked.

- **Log into DB**: The value of the field will be logged into the database if it is checked. This feature is disabled in this test; you can ignore it.

Function name	Arguments	Returned data
variance	*field*	variance of *field*
ewma	*field*	exponentially weighted moving-average of *field*
min	*field*	the minimum of *field*
threshold	*field, threshold*	1 if the value of *field* is larger than *threshold*, otherwise 0
stddev	*field*	the standard deviation of *field*
avg	*field*	the average value of *field*
count		the number of nodes in a group
sum	*field*	the sum of values of *field*
mod	*field, integer*	remainder of *field* divided by integer
max	*field*	the maximum of *field*
abs	*field*	the absolute value of *field*

Table 1: Built-in functions.

Optionally, a selection expression that filters nodes can be specified in the **Where** text field. The selection expression is composed of a comparison operator with expressions on both sides. Comparison operators include = (equal), < (less than), > (greater than), <= (less than or equal), >= (greater than or equal), and <> (not equal).

The sampling period is set in the **Sample period** field. It is composed of an integer and a time unit. The following units are supported: *ms* (milliseconds), *s* (seconds), *min* (minutes), and *h*(hours). SwissQM only supports sampling periods larger than 1024 ms.

3 An Example

The query shown in Figure 3 retrieves the nodeid and the light sensor readings divided by two from every node in the network except node 0. When the query is submitted, nodes start sampling light. You can see this from the **Network** window shown in Figure 4. The number at the beginning of each line indicates the node ID, followed by the action on the node. Since only node 0 is connected to the base station via its UART, the data from the network will be gathered at node 0 and then sent to the base station through its UART. When you stop the query by clicking the **Abort Query** button on the plot tab, the **Network** window will print "no program found" and the **Base station** window will print "stop query". The value of the plotted fields are displayed in the plot tab as shown in Figure 6. The x-axis indicates the time, while the y-axis indicates the value. Data from three nodes collected at the same time are plotted in one vertical line, which makes it hard to locate the point covered by the line. To avoid this, you can check the grouping flag for *nodeid* field in the query. In that case, data from different nodes are plotted on separate lines. You can read the exact values in the **Base station** window. Figure 5 shows the returned data in the **Base station** window. For example, line

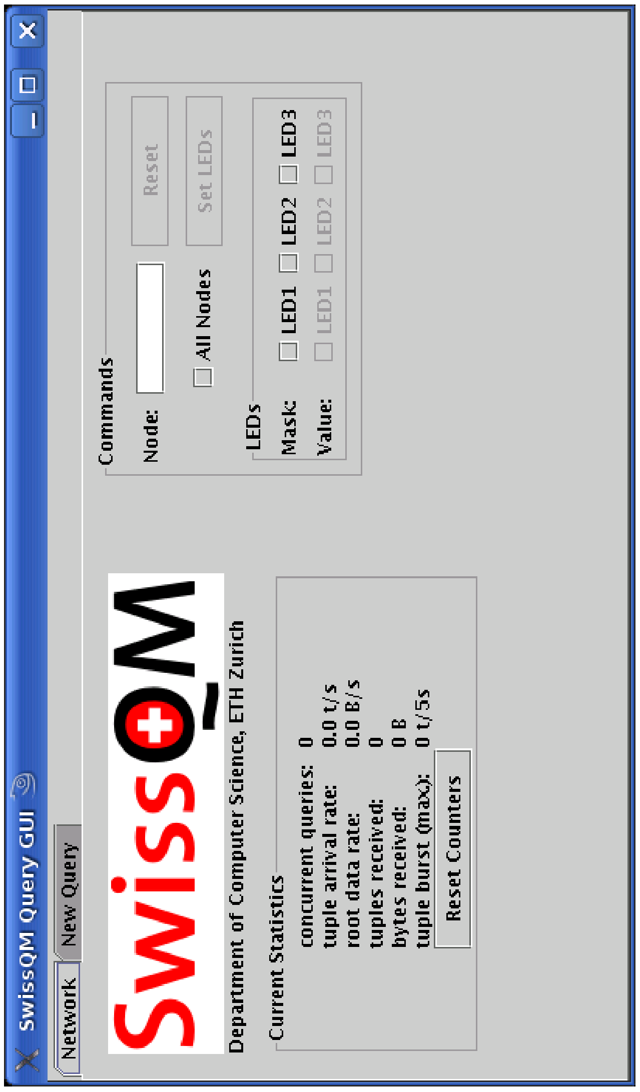
```
[java] timestamp: 46938 epoch:13 2 142
```

means the data collected at time 46938 ms, which is the 13th period, are 2 and 142. Values of the fields are printed in the order they are added. Since field *nodeid* is before *light/2* in the query table, the first number corresponds to *nodeid* and the second corresponds to *light/2*. You can check the correctness by comparing the values in the **Base station** window and the **Network** window. For example, during the last epoch (epoch 13), data returned to the base station are (1, 102), (3, 461), and (2, 142). In the

Network window, you can locate the corresponding epoch. They are the three lines before the first “no program found”.

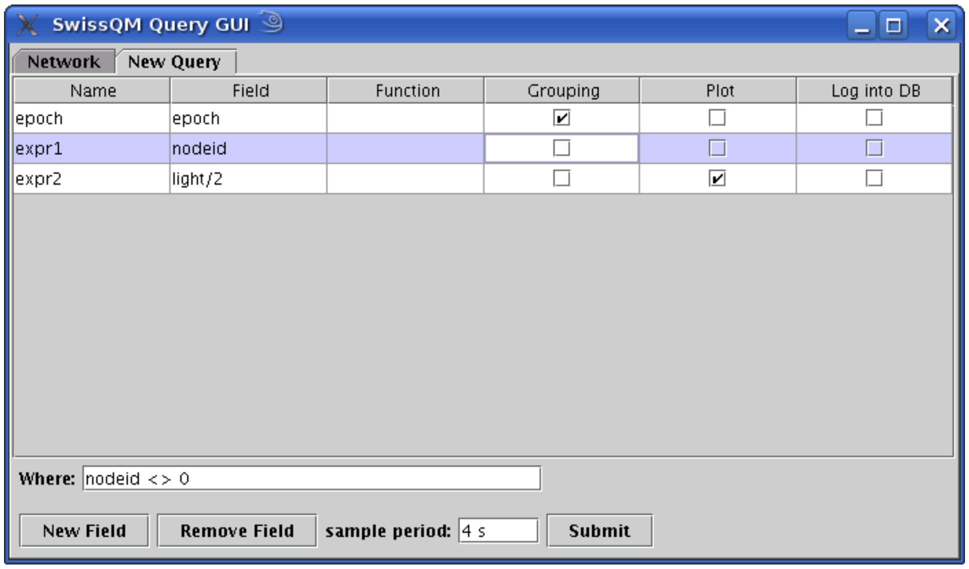
```
1: sample light value 205
3: sample light value 922
2: sample light value 285
```

The returned values for filed *light/2* are the raw light sample data values divided by 2.



```
concurrent queries:  0
tuple arrival rate:  0.0 t/s
root data rate:      0.0 B/s
tuples received:     0
bytes received:      0 B
tuple burst (max):   0 t/5s
```

Figure 2: SwissQM GUI

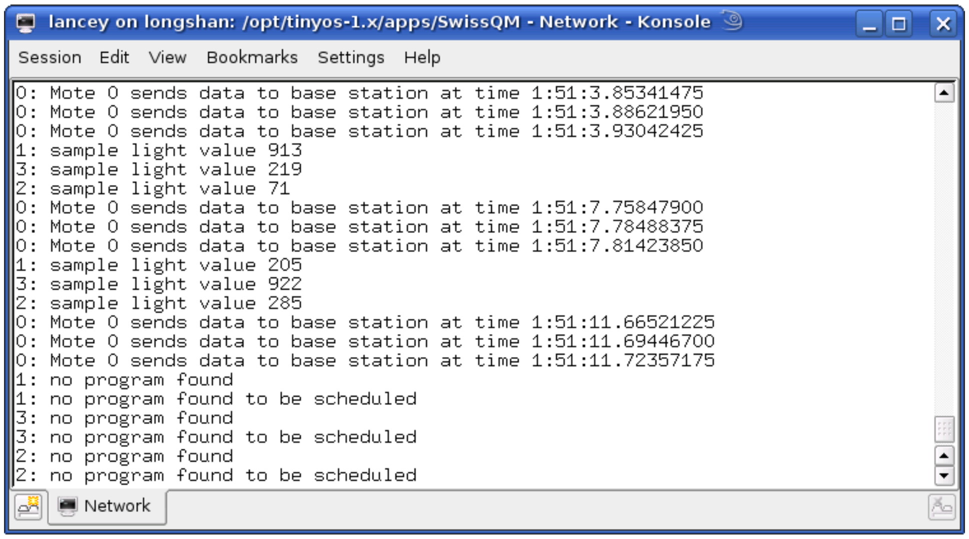


Name	Field	Function	Grouping	Plot	Log into DB
epoch	epoch		✓	☐	☐
expr1	nodeid		☐	☐	☐
expr2	light/2		☐	✓	☐

Where: nodeid <> 0

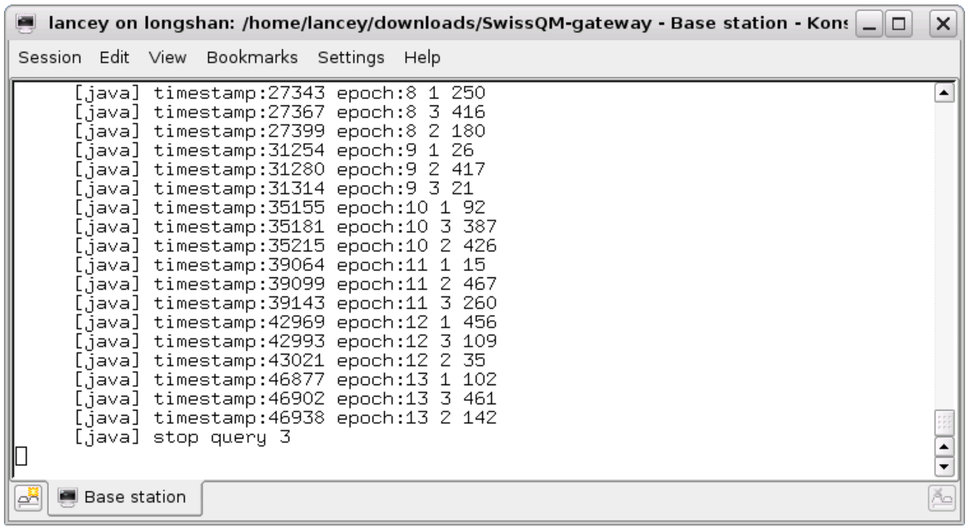
sample period: 4 s

Figure 3: Query for example application.


lancey on longshan: /opt/tinyos-1.x/apps/SwissQM - Network - Konsole
Session
Edit
View
Bookmarks
Settings
Help
Network

```
0: Mote 0 sends data to base station at time 1:51:3.85341475
0: Mote 0 sends data to base station at time 1:51:3.88621950
0: Mote 0 sends data to base station at time 1:51:3.93042425
1: sample light value 913
3: sample light value 219
2: sample light value 71
0: Mote 0 sends data to base station at time 1:51:7.75847900
0: Mote 0 sends data to base station at time 1:51:7.78488375
0: Mote 0 sends data to base station at time 1:51:7.81423850
1: sample light value 205
3: sample light value 922
2: sample light value 285
0: Mote 0 sends data to base station at time 1:51:11.66521225
0: Mote 0 sends data to base station at time 1:51:11.69446700
0: Mote 0 sends data to base station at time 1:51:11.72357175
1: no program found
1: no program found to be scheduled
3: no program found
3: no program found to be scheduled
2: no program found
2: no program found to be scheduled
```

Figure 4: Results in the Network window for example application.


lancey on longshan: /home/lancey/downloads/SwissQM-gateway - Base station - Kons
Session
Edit
View
Bookmarks
Settings
Help
Base station

```
     [java] timestamp:27343 epoch:8 1 250
     [java] timestamp:27367 epoch:8 3 416
     [java] timestamp:27399 epoch:8 2 180
     [java] timestamp:31254 epoch:9 1 26
     [java] timestamp:31280 epoch:9 2 417
     [java] timestamp:31314 epoch:9 3 21
     [java] timestamp:35155 epoch:10 1 92
     [java] timestamp:35181 epoch:10 3 387
     [java] timestamp:35215 epoch:10 2 426
     [java] timestamp:39064 epoch:11 1 15
     [java] timestamp:39099 epoch:11 2 467
     [java] timestamp:39143 epoch:11 3 260
     [java] timestamp:42969 epoch:12 1 456
     [java] timestamp:42993 epoch:12 3 109
     [java] timestamp:43021 epoch:12 2 35
     [java] timestamp:46877 epoch:13 1 102
     [java] timestamp:46902 epoch:13 3 461
     [java] timestamp:46938 epoch:13 2 142
     [java] stop query 3
```

Figure 5: Results in the Base station window for example application.

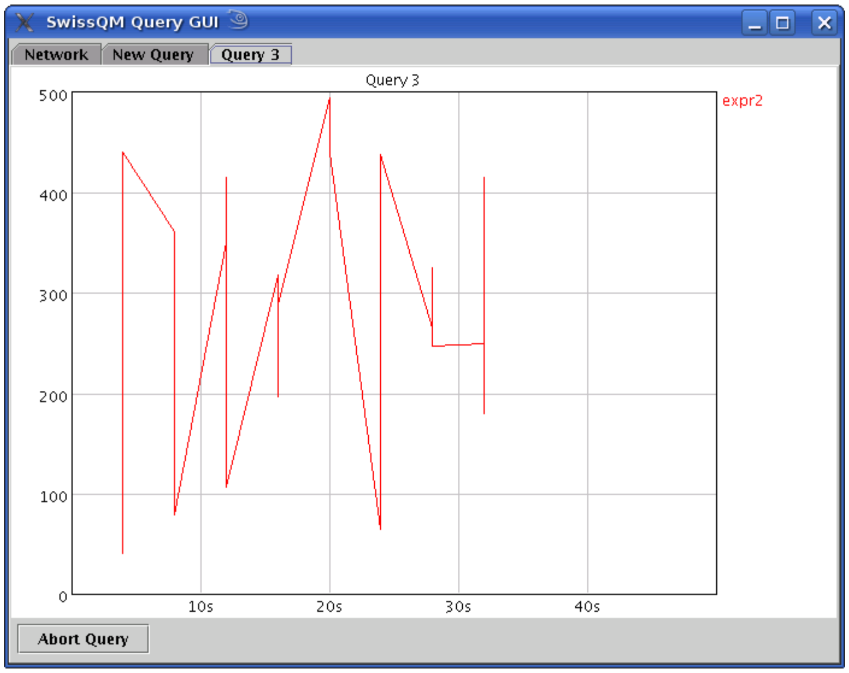


Figure 6: Plot for example application.