

# Tutorial on Wakaama

Internet of Things (2IMN15) 2016-2017, Eindhoven University of Technology

By Leila F. Rahman

Eclipse Wakaama is an open source implementation of the OMA LWM2M protocol in C language. For more information about Wakaama, you can visit its web site at

<https://projects.eclipse.org/projects/technology.wakaama> and its GitHub repository at <https://github.com/eclipse/wakaama>.

This tutorial will discuss about how to install and execute Wakaama on the Raspberry Pi, how to develop a LWM2M client using Wakaama client example, how to develop a broker or cloud application using the Wakaama server example, and some ideas on the tools you can use for developing a Web application for the broker or the cloud service.

## 1 INSTALLING AND EXECUTING WAKAAMA ON RASPBERRY PI

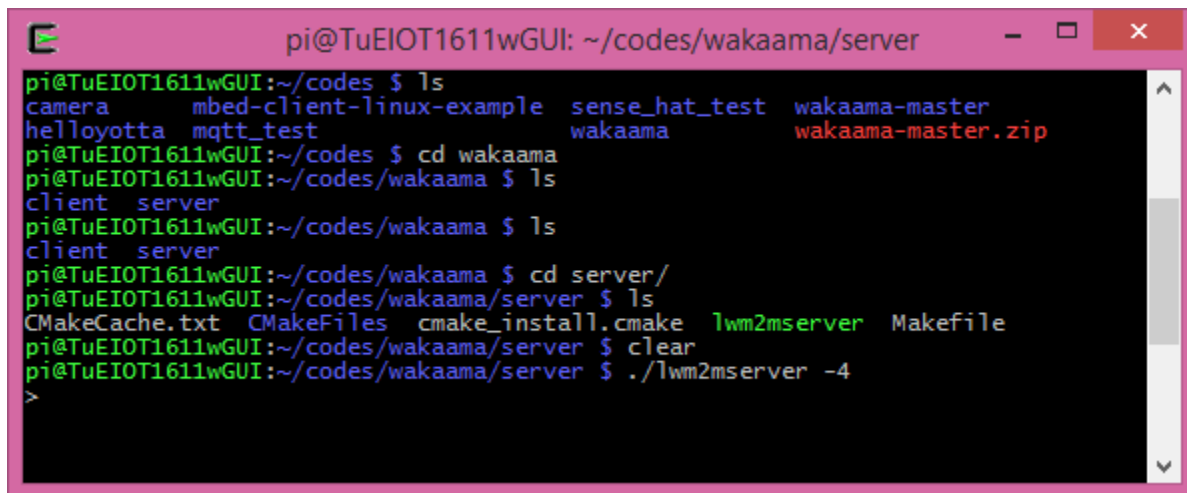
---

1. Download the Wakaama source code from [http://www.win.tue.nl/~lrahman/iot\\_2016/tutorial/wakaama-master.zip](http://www.win.tue.nl/~lrahman/iot_2016/tutorial/wakaama-master.zip) using the following command:  

```
wget http://www.win.tue.nl/~lrahman/iot_2016/tutorial/wakaama-master.zip
```
2. Extract the zip file using the following command:  

```
unzip wakaama-master.zip
```
3. Build and run the client and server examples as specified in the `wakaama-master/README.md` file
4. As we are using IPv4 for the assignment, use option `-4` when running the client or server (see Figure 1 and Figure 2). Other available options for running the client:
  - `-n NAME` Set the endpoint name of the Client. Default: `testlwm2mclient`
  - `-l PORT` Set the local UDP port of the Client. Default: `56830`
  - `-h HOST` Set the hostname of the LWM2M Server to connect to. Default: `localhost`
  - `-p HOST` Set the port of the LWM2M Server to connect to. Default: `5683`
  - `-4` Use IPv4 connection. Default: IPv6 connection
  - `-t TIME` Set the lifetime of the Client. Default: `300`
  - `-b` Bootstrap requested.
  - `-c` Change battery level over time.

TUTORIAL ON WAKAAMA



```
pi@TuEIOT1611wGUI: ~/codes/wakaama/server
pi@TuEIOT1611wGUI:~/codes $ ls
camera      mbed-client-linux-example  sense_hat_test  wakaama-master
hellyoyotta mqtt_test                  wakaama         wakaama-master.zip
pi@TuEIOT1611wGUI:~/codes $ cd wakaama
pi@TuEIOT1611wGUI:~/codes/wakaama $ ls
client  server
pi@TuEIOT1611wGUI:~/codes/wakaama $ ls
client  server
pi@TuEIOT1611wGUI:~/codes/wakaama $ cd server/
pi@TuEIOT1611wGUI:~/codes/wakaama/server $ ls
CMakeCache.txt CMakeFiles cmake_install.cmake lwm2mserver Makefile
pi@TuEIOT1611wGUI:~/codes/wakaama/server $ clear
pi@TuEIOT1611wGUI:~/codes/wakaama/server $ ./lwm2mserver -4
>
```

Figure 1 Running Wakaama lwm2mserver with option -4



```
pi@TuEIOT1611wGUI: ~/codes/wakaama/client
pi@TuEIOT1611wGUI:~ $ cd codes/wakaama/client/
pi@TuEIOT1611wGUI:~/codes/wakaama/client $ ls
CMakeCache.txt CMakeFiles cmake_install.cmake lwm2mclient Makefile
pi@TuEIOT1611wGUI:~/codes/wakaama/client $ ./lwm2mclient -4 -h 192.168.0.22
Trying to bind LWM2M Client to port 56830
LWM2M Client "testlwm2mclient" started on port 56830
> Opening connection to server at 192.168.0.22:5683
-> State: STATE_REGISTERING
13 bytes received from [192.168.0.22]:5683
64 41 34 D8 D8 34 01 38 82 72 64 01 30 dA4..4.8.rd.0
-> State: STATE_READY
|
```

Figure 2 Running Wakaama lwm2mclient with option -4 and -h

5. Figure 3 and Figure 4 show the runtime operation of lwm2mserver.

TUTORIAL ON WAKAAMA

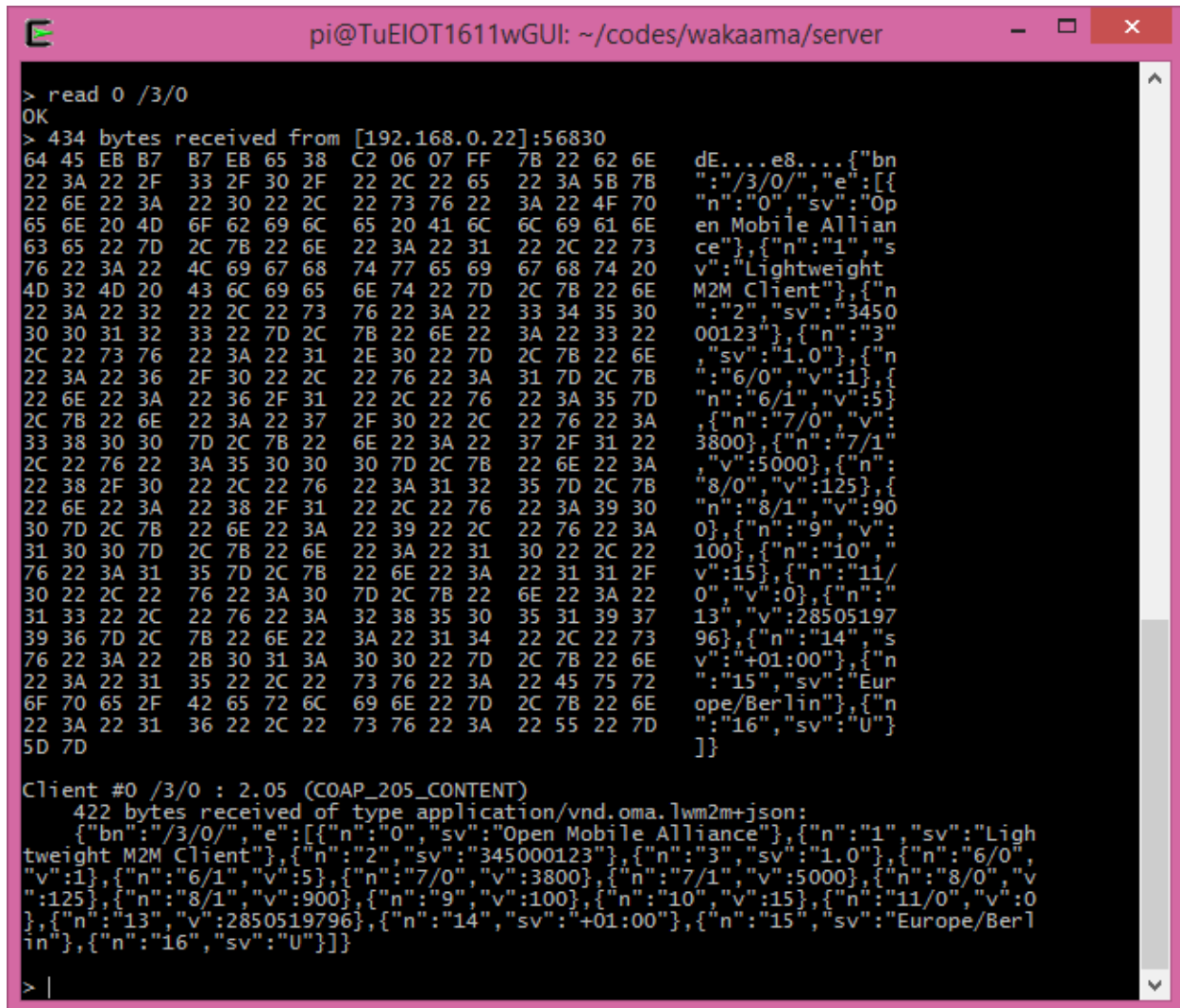
```
pi@TuEIoT1611wGUI: ~/codes/wakaama/server
pi@TuEIoT1611wGUI:~/codes/wakaama/server $ ./lwm2mserver -4
> 153 bytes received from [192.168.0.22]:56830
44 02 34 D8 D8 34 01 38 B2 72 64 11 28 3D 05 65 D.4..4.8.rd.(=.e
70 3D 74 65 73 74 6C 77 6D 32 6D 63 6C 69 65 6E p=testlwm2mclicien
74 03 62 3D 55 06 6C 74 3D 33 30 30 FF 3C 2F 3E t.b=U.lt=300.</>
3B 72 74 3D 22 6F 6D 61 2E 6C 77 6D 32 6D 22 3B ;rt="oma.lwm2m";
63 74 3D 31 35 34 33 2C 3C 2F 31 2F 30 3E 2C 3C ct=1543,</1/0>,<
2F 32 2F 30 3E 2C 3C 2F 33 2F 30 3E 2C 3C 2F 34 /2/0>,</3/0>,</4
2F 30 3E 2C 3C 2F 35 2F 30 3E 2C 3C 2F 36 2F 30 /0>,</5/0>,</6/0
3E 2C 3C 2F 37 2F 30 3E 2C 3C 2F 31 30 32 34 2F >,</7/0>,</1024/
31 30 3E 2C 3C 2F 31 30 32 34 2F 31 31 3E 2C 3C 10>,</1024/11>,<
2F 31 30 32 34 2F 31 32 3E /1024/12>

New client #0 registered.
Client #0:
  name: "testlwm2mclicien"
  binding: "UDP"
  lifetime: 300 sec
  objects: /1/0, /2/0, /3/0, /4/0, /5/0, /6/0, /7/0, /1024/10, /1024/11, /
1024/12,

> help
help      Type 'help [COMMAND]' for more details on a command.
list      List registered clients.
read      Read from a client.
disc      Discover resources of a client.
write     Write to a client.
time      Write time-related attributes to a client.
attr      Write value-related attributes to a client.
clear     Clear attributes of a client.
exec      Execute a client resource.
del       Delete a client Object instance.
create    Create an Object instance.
observe   Observe from a client.
cancel    Cancel an observe.
q         Quit the server.

> help read
read CLIENT# URI
  CLIENT#: client number as returned by command 'list'
  URI: uri to read such as /3, /3/0/2, /1024/11, /1024/0/1
Result will be displayed asynchronously.
```

Figure 3 Runtime operation of Wakaama lwm2mserver



```
pi@TuEIoT1611wGUI: ~/codes/wakaama/server
> read 0 /3/0
OK
> 434 bytes received from [192.168.0.22]:56830
64 45 EB B7 B7 EB 65 38 C2 06 07 FF 7B 22 62 6E dE....e8....{"bn
22 3A 22 2F 33 2F 30 2F 22 2C 22 65 22 3A 5B 7B ":"/3/0/","e":[{"
22 6E 22 3A 22 30 22 2C 22 73 76 22 3A 22 4F 70 "n":"0","sv":"Op
65 6E 20 4D 6F 62 69 6C 65 20 41 6C 6C 69 61 6E en Mobile Allian
63 65 22 7D 2C 7B 22 6E 22 3A 22 31 22 2C 22 73 ce"}, {"n":"1","s
76 22 3A 22 4C 69 67 68 74 77 65 69 67 68 74 20 v":"Lightweight
4D 32 4D 20 43 6C 69 65 6E 74 22 7D 2C 7B 22 6E M2M Client"}, {"n
22 3A 22 32 22 2C 22 73 76 22 3A 22 33 34 35 30 ": "2", "sv": "3450
30 30 31 32 33 22 7D 2C 7B 22 6E 22 3A 22 33 22 00123"}, {"n": "3"
2C 22 73 76 22 3A 22 31 2E 30 22 7D 2C 7B 22 6E "sv": "1.0"}, {"n
22 3A 22 36 2F 30 22 2C 22 76 22 3A 31 7D 2C 7B ": "6/0", "v": "1"}, {"
22 6E 22 3A 22 36 2F 31 22 2C 22 76 22 3A 35 7D "n": "6/1", "v": "5"},
2C 7B 22 6E 22 3A 22 37 2F 30 22 2C 22 76 22 3A {"n": "7/0", "v":
33 38 30 30 7D 2C 7B 22 6E 22 3A 22 37 2F 31 22 3800}, {"n": "7/1"
2C 22 76 22 3A 35 30 30 30 7D 2C 7B 22 6E 22 3A "v": "5000"}, {"n:
22 38 2F 30 22 2C 22 76 22 3A 31 32 35 7D 2C 7B ": "8/0", "v": "125"}, {
22 6E 22 3A 22 38 2F 31 22 2C 22 76 22 3A 39 30 "n": "8/1", "v": "90
30 7D 2C 7B 22 6E 22 3A 22 39 22 2C 22 76 22 3A 0}, {"n": "9", "v":
31 30 30 7D 2C 7B 22 6E 22 3A 22 31 30 22 2C 22 100}, {"n": "10", "
76 22 3A 31 35 7D 2C 7B 22 6E 22 3A 22 31 31 2F v": "15"}, {"n": "11/
30 22 2C 22 76 22 3A 30 7D 2C 7B 22 6E 22 3A 22 0", "v": "0"}, {"n":
31 33 22 2C 22 76 22 3A 32 38 35 30 35 31 39 37 13", "v": "28505197
39 36 7D 2C 7B 22 6E 22 3A 22 31 34 22 2C 22 73 96}, {"n": "14", "s
76 22 3A 22 2B 30 31 3A 30 30 22 7D 2C 7B 22 6E v": "+01:00"}, {"n
22 3A 22 31 35 22 2C 22 73 76 22 3A 22 45 75 72 ": "15", "sv": "Eur
6F 70 65 2F 42 65 72 6C 69 6E 22 7D 2C 7B 22 6E ope/Berlin"}, {"n
22 3A 22 31 36 22 2C 22 73 76 22 3A 22 55 22 7D ": "16", "sv": "U"}
5D 7D ]}

Client #0 /3/0 : 2.05 (COAP_205_CONTENT)
422 bytes received of type application/vnd.oma.lwm2m+json:
{"bn":"/3/0/","e":[{"n":"0","sv":"Open Mobile Alliance"}, {"n":"1","sv":"Ligh
tweight M2M Client"}, {"n":"2","sv":"345000123"}, {"n":"3","sv":"1.0"}, {"n":"6/0",
"v": "1"}, {"n":"6/1", "v": "5"}, {"n":"7/0", "v": "3800"}, {"n":"7/1", "v": "5000"}, {"n":"8/0", "v
": "125"}, {"n":"8/1", "v": "900"}, {"n":"9", "v": "100"}, {"n":"10", "v": "15"}, {"n":"11/0", "v": "0
"}, {"n":"13", "v": "2850519796"}, {"n":"14", "sv": "+01:00"}, {"n":"15", "sv": "Europe/Berl
in"}, {"n":"16", "sv": "U"}]}
```

Figure 4 Read operation of Wakaama lwm2mserver

## 2 DEVELOPING LWM2M CLIENT ON RASPBERRY PI

1. Developing your LWM2M client (for the Light Device and the Sensor device) is done by modifying the client example source code in the `../wakaama-master/examples/client` folder.
2. Create new C file for every new object in your LWM2M client. For example, for the LWM2M client in the Light Device, you need to create a new `object_light_profile.c` file for the Light Profile object by modifying the `object_device.c` file in the `../wakaama-master/examples/client` folder. Adjust the write and read functions for the Light Profile object in `object_light_profile.c`. Adjust as well the `get_object` and `free_object` functions in `object_light_profile.c`.

3. Adjust the `lwm2mclient.c` and `lwm2mclient.h` files in `../wakaama-master/examples/client` folder accordingly. In `lwm2mclient.c`, you need to initialize the Light Profile object in the beginning of the main function, and free the object memory allocation in the end of the main function. In `lwm2mclient.c`, you also need to set the right number of objects in your LWM2M client.
4. Adjust file `../wakaama-master/core/liblwm2m.h` by adding the ID of the new object under line 167.
5. Adjust file `../wakaama-master/examples/client/CMakeLists.txt` to include the new object file and any new libraries used in the modified code.
6. Build and run the new `lwm2mclient`.

### 3 DEVELOPING A BROKER OR CLOUD APPLICATION USING WAKAAMA SERVER

---

1. Open file `../wakaama-master/core/liblwm2m.h` and look at lines 703 – 722. Those lines provide a list of APIs that you can use in your broker or cloud service implementation.
2. Another option is to create Inter Process Communication (IPC) between your broker or cloud application and the Wakaama server. The IPC is used to send command data accepted by Wakaama server during its runtime (see Figure 3 and Figure 4 for the list of commands) by your broker or cloud application to the Wakaama server.

### 4 DEVELOPING WEB APPLICATION FOR YOUR BROKER OR CLOUD SERVICE

---

Some possible (not limiting) options:

1. Use Wt, a C++ Web Toolkit (<https://www.webtoolkit.eu/wt>).
2. Use Django, a Python Web Framework (<https://www.djangoproject.com/>), or any other Python Web framework (<https://wiki.python.org/moin/WebFrameworks>).
3. Use Spark, A micro framework for creating web applications in Java 8 with minimal effort (<http://sparkjava.com/>).
4. Use node.js (<https://nodejs.org/en/>) as backend and AngularJS (for example) as front end.
5. Use Go programming language (<https://golang.org/>).
6. Use any other framework, tools or libraries that you wish or that you are familiar with.