

LAURA ARCHITECTURE PROJECT

LAURA ARCHITECTURE USABILITY TUTORIAL: HOW TO USE AND MODIFY THE LAURA VIRTUAL MACHINE

LAURA Project Team:

Adriano Francisco Branco - Afbranco@gmail.com

Bruno Alves Agrizzi - brunoagrizzi@gmail.com

Isaac Simões Araújo Pereira - pereira.zc@gmail.com

José Gonçalves Pereira Filho - zegonc@gmail.com

Ruan Rocha Martinelli - martinelliruan@gmail.com

Sergio Teixeira - sergio@multicast.com.br

VITÓRIA-ES, BRAZIL

2018

Copyright 2018 LAURA Project Team

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either expressed or implied.
See the License for the specific language governing permissions and
limitations under the License.

This license includes the file "NOTICE.TXT" as defined in item 4.d of the apache 2.0 license.
This file is available in the same location (repository) in which this license is available.

Revision: 20180808

This tutorial and up-to-date documents are available in the LAURA Project Web site at:
<https://laura-architecture.github.io/>

if you have comments about this tutorial, please submit your feedback to:
Sergio Teixeira - sergio@multicast.com.br

List of Tables

Table 1 - WSN specifications previously configured in the LAURA VM.....	7
---	---

List of Figures

Figure 1 – Terminal showing the command to start 'TerraWebControl' .	9
Figure 2 – Opening the 'TerraWebControl' final application.	10
Figure 3 –The command to start ' <i>Terra Dia</i> 'to disseminate the node codes (bytecodes).	11
Figure 4 – Compiling the LauraTest01.terra source code	12
Figure 5 – Terminal showing the command to start ' <i>TerraGateway</i> ' module.	12
Figure 6 – <i>TerraWebControl</i> final application showing sensed data	12

CONTENTS

CHAPTER 1. OBJECTIVES AND ESSENTIAL INFORMATION	6
1.1 Objectives	6
1.2 Target stakeholder	6
1.3 Network configurations, requirements and technical assumptions	6
1.3.1 Network configuration and specifications used in the LAURA VM	6
1.3.2 Requirements and technical assumptions	7
CHAPTER 2. HOW TO USE OR MODIFY LAURA ARCHITECTURE	9
2.1 Download and execute the VM in your computer	9
2.2 Starting the 'TerraWebControl' final application	9
2.3 More details about the physical layer – <i>Terra System</i>	10
2.4 Node Code Dissemination – <i>TerraDia</i>	11
2.5 Gateway and communication module - <i>TerraGateway</i>	12
2.6 Services and functions of the 'Core Layer' - <i>TerraCore</i>	13
2.7 Final application - <i>TerraWebControl</i>	14

Chapter 1. Objectives and essential information

1.1 Objectives

The main object of this document is to present a step by step tutorial to guide and facilitate the work of those who intend to use and/or modify the LAURA VM (Teixeira et al., 2017) in order to develop its own LAURA applied specific Wireless Sensor Network (WSN)/Internet of Things (IoT) final solution. The LAURA VM uses the technological choices initially proposed in <Paper awaiting approval>. In order to use or modify the VM, it is necessary to make some settings according to hardware platform which you intend to use, the network design and other aspects related to the applied final application and the enterprise environment. This tutorial reproduces the experiment 3, available in section 5.3 in <Paper awaiting approval> and presents the data flow from the lowest layer to the final application and how to modify or adapt the LAURA elements, modules or components. Thus, is strongly recommended to read <Paper awaiting approval> for better understanding of this tutorial.

1.2 Target stakeholder

This tutorial was developed to assist the developer or team of developers who wish to develop a LAURA applied specific WSN/IoT final solution based on LAURA architecture, using the ready to use LAURA VM with the technological choices made as described in section ‘4.4. LAURA Applied architecture’ <Paper awaiting approval>.

This tutorial can be used by anyone who has the specific knowledge required or compatible with the professional profiles as described in section ‘4.1. Stakeholders’ <Paper awaiting approval>. In addition, this tutorial seeks to simplify and facilitate the use of the LAURA VM, demonstrating how to modify and adapt the elements, modules or components according to the needs or interests of stakeholders.

1.3 Network configurations, requirements and technical assumptions

1.3.1 Network configuration and specifications used in the LAURA VM

The LAURA VM is configured according to the specification and network settings used in the experiment 3, section 5.3 <Paper awaiting approval>. Table 1 presents WSN specification and some essential network configurations necessary to run the final application.

Table 1 - WSN specifications previously configured in the LAURA VM

Experiment 3 - WSN Specifications - Physical layer - Concept proof					
WSN	Description	Sink node	Others nodes	MIB600 MAC	MIB600 IP
01	MicaZ with TelosB (1 SN + 5 nodes)	MIB600CA + MICAZ - ID: 10	TelosB - ID 11 to 15	00-20-4a-d2-d7-81	192.168.2.201
02	IRIS with IRIS (1 SN + 9 nodes)	MIB600CA + IRIS - ID: 20	IRIS with MDA100 sensor - ID 21 to 29	00-20-4a-d2-d7-76	192.168.2.202
03	MicaZ with MicaZ (1 SN + 4 nodes)	MIB600CA + MICAZ - ID: 30	MICAZ with MDA100 sensor - ID 31 to 34	00-20-4a-d2-E0-8B	192.168.2.203
Hostess host (your computer)				IP Address ->	192.168.2.129
Config the virtual machine management (e.g vmware) with bridge mode				IP Address ->	192.168.2.220
In serial forward using server port: 9002					
For example, to communicate with WSN2 using: network@192.168.2.202:10002					

If you have a MIB600 obviously, your MAC address will be different. If you do not use MIB600 Gateways you can connect the Sink node directly to your computer via USB port. We used static IP address. The network topology used in the VM is the same used in experiment 3, as shown in Figure 9 in [Paper awaiting approval](#).

To assign an IP address to the Sink node you must use the ‘arp’ command. Follow the steps below to assign the IP address:

- Open the terminal and run the command "arp -s <ip> <MAC>". For example, to bind the IP address to the Sink node of WSN2 as previously shown in the Table 1, use the comand: "arp -s 192.168.2.202 00-20-4a-d2-d7-76";
- After linking the IP to the Sink node, it must be accessed via telnet to test the connection. Execute the command ‘telnet 192.168.2.202 1’ to verify the connection to the WSN2 Sink node;
- To verify the settings, execute the command ‘telnet 192.168.2.202 9999’;
- To exit the settings without saving, press 8.

1.3.2 Requirements and technical assumptions

It is important to know the essential requirements and some technical assumptions necessary to use and modify the LAURA VM. The essential requirements are:

- Read carefully the study [Paper awaiting approval](#) that presents the LAURA architecture;
- Know the basic concepts of *Terra System* and have the necessary knowledge to configure a sensor node with *Terra System* so that it is able to receive an update of the node code (bytecode);
- More details about *Terra System* can be obtained in the:
 - Terra System Introduction and background concepts at <http://www.inf.puc-rio.br/~prjterra/>
 - Github project at: <https://github.com/afbranco/Terra> at <http://afbranco.github.io/Terra/terra-home.html>
 - Branco web site: <http://www.inf.puc-rio.br/~abranco/>
 - study ‘Terra: Flexibility and Safety in Wireless Sensor Networks’ (Branco et al., 2015);
 - PhD thesis: http://www.inf.puc-rio.br/~abranco/files/Doutorado/15_PhD_branco.pdf
 - How to install, configure and use Terra System in TinyOS-Based platforms at <http://afbranco.github.io/Terra/terra-platf-tinyos.html>

The technical assumptions necessary to use the LAURA VM are:

- The LAURA VM is already configured according to the information previously presented in section 1.3.1. However, it is necessary to observe the step by step guide, presented in chapter 2, in order to be able to execute the final *TerraWebControl* application that is available in the VM;
- Before starting the step by step guide presented in chapter 2, we are assuming that the sensor nodes that you intend to use are already configured with *Terra System*, requiring only the dissemination of the node codes (bytecodes);
- This tutorial intends to guide you to modify or adapt the elements, modules or components of LAURA architecture, reproducing the experiment 3.

Chapter 2. How to use or modify LAURA Architecture

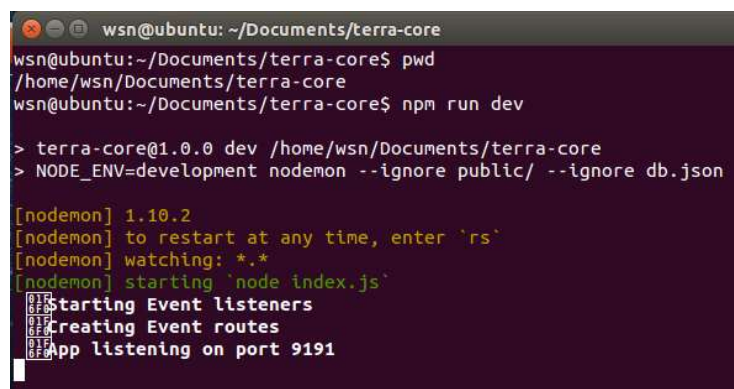
This Chapter presents the necessary steps to make settings, run applications and modify or adapt the VM according to specific needs. We are going to start the steps with the guidance to execute a final web application that will allow the visualization of the sensed data as soon as they are received. Later on, we will present details of how you can configure or adapt your own final application to receive the sensed data.

2.1 Download and execute the VM in your computer

- LAURA VM is available for download at:
 - <http://www.multicast.com.br/laura-architecture/how-to-use-laura/>
- You can use the VM manager of your choice, however, it is important to know that we used the Vmware to test LAURA VM;
 - use the 'lauraarchitecture' password to log into the VM.

2.2 Starting the 'TerraWebControl' final application

- The goal of this step is to run the 'TerraWebControl' final application in order to view the sensed data of the WSN. At this point, our recommendation is that you use the final application that is ready to display the sensed data. Later you will have the opportunity to configure, adapt and use your own final application;
- This step will allow you to view the exact moment when the data will be displayed in this application, allowing you to identify whether the steps performed are working as expected.
- Access the folder '/home/wsn/Documents/terra-core' as shown in Figure 1;
- Open a terminal and execute the command 'npm run dev' to start 'TerraWebControl';



```

wsn@ubuntu: ~/Documents/terra-core
wsn@ubuntu:~/Documents/terra-core$ pwd
/home/wsn/Documents/terra-core
wsn@ubuntu:~/Documents/terra-core$ npm run dev

> terra-core@1.0.0 dev /home/wsn/Documents/terra-core
> NODE_ENV=development nodemon --ignore public/ --ignore db.json

[nodemon] 1.10.2
[nodemon] to restart at any time, enter `rs`
[nodemon] watching: *.*
[nodemon] starting 'node index.js'
Starting Event listeners
Creating Event routes
App listening on port 9191
  
```

Figure 1 – Terminal showing the command to start 'TerraWebControl'.

- After executing the command, you will start the client part of the application that establishes the connection with the 'TerraCore' via Socke.io;
- This command will start the TerraCore server on port 9191. In the version of TerraCore and TerraWebControl available in LAURA VM, the command shown in Figure 1 starts both applications;

- The *TerraWebControl* application is served by Terra Core and will be available on the same port when accessed through a web browser;
- You must open the Browser and enter the URL 'http://localhost:9191' as shown in figure 2. From this point, the Web application is able to display the sensed data as soon as it is received;
- It is important to remember that the codes and other files available in this directory are also available in the GitHub repository of the LAURA project (Teixeira et al., 2017). This folder is a copy of the files that are in GitHub;

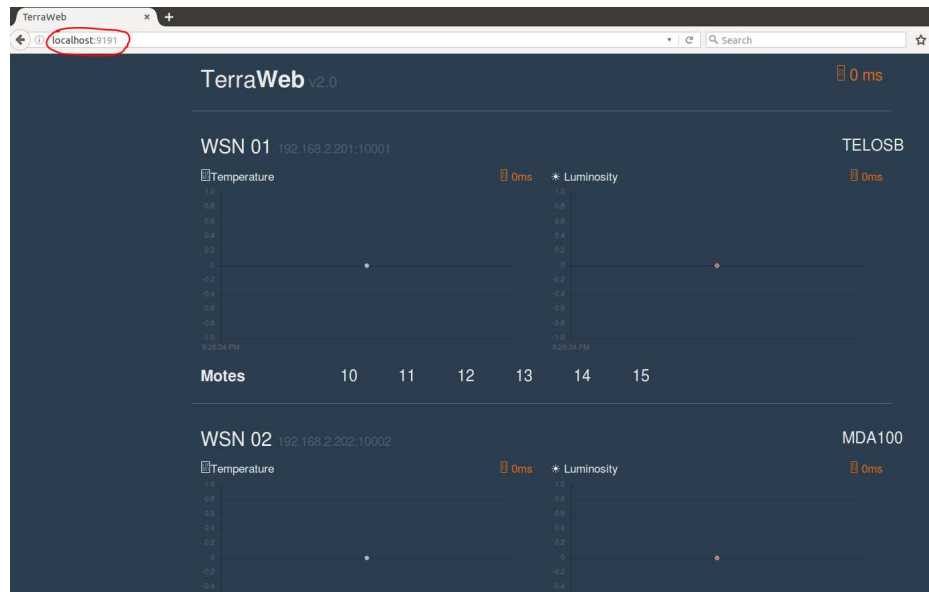


Figure 2 – Opening the ‘TerraWebControl’ final application.

- If you plan to use the version of *TerraWebControl* that is available in the Github repository, you must use the following steps, since this version has been updated with a decoupled repositories structure from TerraCore. In this way, follow these steps:
 - Clone the repository (<https://github.com/laura-architecture/terra-web-control/>) from GitHub to your computer;
 - Assuming you have Node.js and npm installed, run the command `npm install`;
 - Run the command `npm start`;
 - Open the browser on `localhost:5000`.

Attention: From this point we start to use a bottom-up approach, following the flow of data from the physical layer to the application layer of the LAURA architecture.

2.3 More details about the physical layer – *Terra System*

- Based on the technological choices made and justified in <Paper awaiting approval>, the sensed data originates from a Terra-based WSN are delivered to the *TerraGateway* (Martinelli, 2017). Each WSN sensor node runs a Terra (bytecode), such as the code 'LauraTest01.terra' available in the '/home/wsn/Documents/terra-core/gateway/dia' folder. The sensed data is transmitted to the Sink node by means of Active Messages (AM) (Eicken et al., 1992) mechanism through Terra System OS;
- AM received by the Sink Node are delivered to the *TerraGateway* module (from the Core layer which will be presented in more detail below in section 2.5) through TOSSAM (Silvestre, 2017) communication channels;

- When creating LAURA VM based on the Requirements and technical assumptions previously presented in section 1.3.2, it was necessary to install the:
 - TOSSAM library, in order to support the programming of communication channels between the Sink Node and the *TerraGateway*. More details about how to install TOSSAM packages are available in: <http://www.inf.ufg.br/~brunoos/tossam/#Install>;
 - The Terra System compiler to allow compilation of '.terra' codes to bytecodes '.vmx'. More details about Terra System were previously indicated in section 1.3.2.

2.4 Node Code Dissemination – *TerraDia*

- The experiment used in this tutorial is based on *Terra System*. Assuming that the node is already configured with *Terra System*, it is necessary to spread the node codes (bytecodes) that will be running into the Terra Virtual Machine (VM-T);
- Details about how to configure a Terra-based node are not part of the scope of this tutorial. More details about Terra System were previously indicated in section 1.3.2;
- We use *TerraDia* application (Ribeiro, 2016) in order to disseminate the node codes of the already compiled nodes. *TerraDia* is a Lua-based (Ierusalimschy et al., 1993) application that uses TOSSAM library in order to establish communication with the sink node through AM messages to disseminate the bytecodes;
- Access the folder '/home/wsn/Documents/terra-core/gateway/dia', open the terminal and execute the command 'lua main.lua default' as shown in Figure 3;

```
wsn@ubuntu:~/Documents/terra-core/gateway/dia$ pwd
/home/wsn/Documents/terra-core/gateway/dia
wsn@ubuntu:~/Documents/terra-core/gateway/dia$ lua main.lua default
sending code to config: 10002 192.168.2.202 20
file vID 168
Tentando conectar
```

Figure 3 –The command to start '*Terra Dia*' to disseminate the node codes (bytecodes).

- Before executing the command to disseminate the node codes, it is important to understand the parameters necessary to perform any '*TerraDia*' dissemination. In addition, the compiled VM-T code (file: LauraTest01.vmx) must also be in the same folder;
- The default setting is already set in the main.lua file in the same folder;
- To run '*TerraDia*' with a custom port, IP or sensor node ID, use the following command:
- 'lua main.lua <port> <ip address> <nodeId>'
 - <port> : port address used;
 - <ip address> Sink node IP address;
 - <nodeid> Sink node ID;
- To use the default configuration, simply execute the command 'lua main.lua default';
- After executing this command, the file LauraTest01.vmx will be disseminated to the WSN via Sink node, according to the existing configurations in the file 'main.lua';
- You must disseminate the bytecodes of each WSN at a time. In the current version of '*Terra System*', it is not possible to disseminate different codes to different WSNs at the same time;
- If you want to make changes to the compiled code (LauraTest01.vmx) that will be disseminated, you need to modify the source file (LauraTest01.terra);
- Access the folder '/home/wsn/terra/terra', edit the source code and then compile;
- To compile, use the command 'terra LauraTest01.terra' as shown in figure 4;

```

wsn@ubuntu:~/terrac/terra$ pwd
/home/wsn/terrac/terra
wsn@ubuntu:~/terrac/terra$ terrac LauraTest01.terra
-----
-- Terra Compiler: TerraNet_MsgQ          VM Code: 000.002.000 --
-----
-- Memory allocation in bytes: (code optimized) --
-- Total = Code + Ctl/Vars + Stack | Radio Msgs --
-- 647    433    202    12    |    19    --
-----
-- Target platforms x Max program size --
-- tossim = 1728 bytes --
-- micaz = 1920 bytes --
-- telosb_hyb = 1920 bytes --
-- telosb_full = 4224 bytes --
-- mica2dot = 1920 bytes --
-- iris_full = 4224 bytes --
-- iris_hyb = 1920 bytes --
-- mica2 = 1920 bytes --
-----
wsn@ubuntu:~/terrac/terra$

```

Figure 4 – Compiling the LauraTest01.terra source code

2.5 Gateway and communication module - *TerraGateway*

- *TerraGateway* is the lua-based module responsible for the connection between *TerraCore* and Sink Node. This module receives the sensed data from the WSN nodes and delivery them to the upper layers of the LAURA architecture;
- Access the folder `/home/wsn/Documents/terra-core/gateway/`, open the terminal and execute do command `'lua main.lua default'` to run *TerraGateway* as show in figure 5;

```

wsn@ubuntu:~/Documents/terra-core/gateway$ pwd
/home/wsn/Documents/terra-core/gateway
wsn@ubuntu:~/Documents/terra-core/gateway$ lua main.lua default
-- listening for messages in port 10002
-- connecting to host 192.168.2.202

```

Figure 5 – Terminal showing the command to start *TerraGateway* module.

- Remember that this command will use the default settings that were previously made in the `main.lua` file in the same folder;
- After executing this command, the sensed data will be displayed in the *TerraWebControl* application as shown in figure 6;



Figure 6 – *TerraWebControl* final application showing sensed data.

- The main file of the *TerraGateway* module is the `'main.lua'`. The communication channels with the Sink node are configured into this file through the TOSSAM library. In the example, used in this tutorial, we use the Serial Fowarding protocol. It is possible to use other protocols according to the options available in TOSSAM;

- It was defined some data structures as Lua tables into '*main.lua*' that follow the format of the messages that are received from the Sink Node via AM;
- Data received through TOSSAM is stored in the '*nx_struct*' Lua table, as it can be seen in '*main.lua*', just after the line containing the command '*mote: register [[nx_struct msg_serial [145] {}*'. The structure of this table follows the same data structure that was adopted for collecting and sending the data packets that were used in the VM-T node code. An example of using this structure, in the node code, can be seen in the code example '*LauraTest01.terra*' that was previously presented in the section '2.4 Node Code Dissemination'. This table contains the following fields:
 - id - Serial identifier for the message;
 - source – ID of the mote who sent the message;
 - target – ID of the mote who received the message. Usually the Sink node's ID
 - d8, d16 and d32 - Generic data fields, used to store sensed values and metadata
 - For the experiment used in this tutorial and in this initial version of LAURA, the field d16 was used to store luminosity and temperature readings from the sensors. Fields d8 and d32 were not used in this experiment.
- After receiving the data packets, it is made the transformation from the Lua tables into JSON (International, 2013) messages, following the same data format as received from AM packets. After this step, the JSON messages are converted into JavaScript Objects resident in memory to improve the performance;
- The ZeroMQ (ZeroMQ, 2017) library was used to deploy the Pub/Sub channel between *TerraCore* and *TerraGateway*. The *TerraGateway* act as a Publisher and *TerraCore* as a Subscriber;
- *TerraGateway* maintains a permanent Publisher/Subscriber (Pub/Sub) (Eugster et al., 2003) communication channel with *TerraCore* (Martinelli, 2017). This channel consists of a TCP socket and is set up using the ZeroMQ library (ZeroMQ, 2017), which is available for both Lua and JavaScript/Node.js (*TerraCore*'s runtime);
- Right after the message is parsed into JSON, it is pushed into the channel using ZeroMQ's '*send()*' function;

2.6 Services and functions of the '*Core Layer*' - *TerraCore*

- *TerraCore* provides the 'Core API services' and other features available in the 'Core Layer', as can be seen in figure 9 available in the section 4.4. 'LAURA applied architecture' [<Paper awaiting approval>](#);
- *TerraCore* acts as Subscriber in the ZeroMQ connection, listening to all incoming messages;
- Messages sent by *TerraGateway* are received by a dedicated listener. When a message arrives, it's immediately parsed into a JSON object using *JSON.parse()*;
- The new JSON object is sent to a Dispatcher module;
- Because TOSSAM sends the data in a raw format – the d8, d16 and d32 can have any kind of data, for example – the message needs to go through some transformations before being handled;
- The message is then evaluated and stored in a new object based on what was received in the d8, d16 and d32 fields;
- In this step, it can also occur that some of the data received is in an undesired format (e.g. temperature is in Kelvin instead of Celsius or Fahrenheit). If that is the case, the raw value is converted into the desired unit before proceeding to the next step;
- Now, with the object properly transformed, it is then sent to a '*dispatchEvent()*' function to be saved and sent across all the clients that are expecting it;

- The MySQL was used as a relational database to store the objects received by the Dispatcher;
- After saved, the Dispatcher sends the messages to all active Socket.io and ZeroMQ clients listening on Terra Core's address and port;
- To fetch old records on demand, Terra Core also provides a REST API with endpoints to search and filter previous messages.

2.7 Final application - TerraWebControl

- The basic information and guidelines for running TerraWebControl have already been presented in section 2.2. The following bullets present more details about this application;
- You may use, adapt or develop your own final application like TerraWebControl or another type of application that meets your needs;
- This web application is connected to TerraCore using the Socket.io library and receives the messages sent by the ``emit()`` function in TerraCore's dispatcher;
- The messages received by TerraWebControl are filtered by the mote ID and pushed to an individual chart of the mote in real time (without refreshing the page);
- TerraWebControl groups all messages received via Socket.io into separate charts. Each chart displays the readings of a specific mote in a given time frame.

References

- Branco, A., Sant 'anna, F., Ierusalimschy, R., Rodriguez, N., Rossetto, S., 2015. Terra: Flexibility and Safety in Wireless Sensor Networks. *ACM Trans. Sen. Netw. Artic.* 11, 18–23. <https://doi.org/10.1145/2811267>
- Eicken, T. Von, Culler, D.E., Goldstein, S.C., Schauser, K.K., 1992. Active Messages: a Mechanism for Integrated Communication and Computation, in: *Proceedings of the 19th International Symposium on Computer Architecture.* pp. 1–20. <https://doi.org/10.1109/ISCA.1992.753322>
- Eugster, P.T., Felber, P., Guerraoui, R., Kermarrec, A.-M., 2003. The Many Faces of Publish/Subscribe. *ACM Comput. Surv.* 35, 114–131. <https://doi.org/10.1145/857076.857078>
- Ierusalimschy, R., Celes, W., Figueiredo, L.H., 1993. Lua - The programming language [WWW Document]. URL <https://www.lua.org/> (accessed 7.29.18).
- International, E., 2013. The JSON Data Interchange Format [WWW Document]. URL <http://www.json.org/> (accessed 2.27.17).
- Martinelli, R.R., 2017. Terra Gateway e Terra Core: Uma solução de suporte ao desenvolvimento de aplicações finais para Redes de Sensores sem fio. Federal University of Espirito Santo.
- Ribeiro, F.S., 2016. Terra DIA: Uma solução de suporte a atualização dinâmica de bytecodes em redes de sensores sem fio baseadas em maquina virtual. Federal University of Espirito Santo.
- Silvestre, B.O., 2017. TOSSAM - TinyOS Serial AM for Lua [WWW Document]. URL <http://www.inf.ufg.br/~brunoos/tossam/> (accessed 7.14.17).
- Teixeira, S., Agrizzi, B.A., Martinelli, R.R., Branco, A.F., Pereira, I.S.A., Al., E., 2017. LAURA Architecture: Codes, Experiments, Documentation, Videos and others related files [WWW Document]. URL <https://laura-architecture.github.io> (accessed 7.1.17).
- ZeroMQ, 2017. ZeroMQ [WWW Document]. iMatix Corp. URL <http://zeromq.org/> (accessed 11.27.17).