Using NCL to Synchronize Media Objects, Sensors and Actuators

Álan L. V. Guedes[1,2]
alan@telemidia.puc-rio.br

Marcio Cunha[1,3]
mcunha@inf.puc-rio.br

Hugo Fuks[1,3]
hugo@inf.puc-rio.br

Sergio Colcher[1,2]
colcher@inf.puc-rio.br

Simone D.J Barbosa[1]
simone@inf.puc-rio.br

[1] Department of Informatics, PUC-Rio
[2] TeleMídia Lab, PUC-Rio
[3] Second Lab, PUC-Rio
[1,2,3] Rua Marques de Sao Vicente, 225
Gavea, Rio de Janeiro, RJ, 22451-900, Brasil

ABSTRACT

IoT (Internet of Things) technologies are underway. Both industry and academia have been proposing technologies to support IoT. In particular, some IoT scenarios include audio-video data and aims at integrating multimedia content and IoT technologies. In this paper, we investigate the support for those scenarios through a multimedia language. More precisely, we use NCL (Nested Context Language) to take advantage of IoT devices and to specify interactive multimedia applications synchronizing sensors and actuators, besides the media objects. To help evaluate our approach, we present a usage scenario using an NCL application that synchronizes media objects and IoT devices.

Keywords

Internet of Things; Multimedia Languages; Nested Context Language; NCL; Ginga-NCL

1. INTRODUCTION

The Internet of Things (IoT) can be viewed as the interconnection of pervasive and uniquely identifiable physical objects—such as RFID (Radio-Frequency IDentification) tags, sensors, actuators, and embedded computing devices—using the existing Internet infrastructure [2]. Those physical objects cooperate with one another, as well as with the environment and with people, to better support domestic (e.g., assisted living and e-health) and industrial (e.g., automation and logistics) applications.

Both industry and academia have become increasingly interested in IoT in recent years, making efforts to study and propose different technologies to support IoT scenarios. Some of those efforts, such as those made by the MPEG [8] and ITU-T SG 16 [10] standardization groups, aim to integrate multimedia and IoT technologies.

Scenarios mixing multimedia and IoT include assisted living facility, monitoring, and entertainment. In assisted living facility, devices with media rendering capabilities, such as wearables, may control actuators (heater, lights, fans, etc.) or present data from ambient sensors (temperature, light, humidity, proximity, gas, etc.). Regarding monitoring, city governments may take advantage of the proliferation of sensor-rich devices (wearables and smartphones) for allowing citizens to report traffic condition, e.g., by uploading media objects (audio clips, videos, photos) and sensor data. For entertainment, transmission of live events (e.g., sports or concerts) may give a sense of immersion to remote audiences, e.g., by using actuators (e.g., lights, fans, directional speakers) that reproduce the local event environment captured by sensors.

All the above scenarios require the specification of relationships between media objects and events related to sensor or actuator devices. In this paper, we investigate the use of a multimedia language to support this kind of relationship. Multimedia languages may take advantage of their synchronism and media abstractions to control IoT devices. More specifically, we use NCL (Nested Context Language) [13] to handle both sensor and actuator entities, besides its ordinary media objects. NCL is a standard for interactive applications for Digital TV and IPTV services. To help evaluate our approach, we present a usage scenario consisting of an NCL application that synchronizes a video and three actuators: fan, light, and scent dispenser.

Other proposals to support the integration of multimedia and IoT devices include: MPEG-V [6], a data format for specifying sensor and actuators parameters; MPEG MTT [7], a transport format that supports the multiplex of actuator data; and IBM's Node-RED [11], a graphical tool that enables the development of IoT-based systems and their deployment in the IBM private cloud service. To the best of our knowledge, however, our work is the first one to use NCL to handle IoT devices.

The rest of the paper is organized as follows. Section 2 discusses related work. Section 3 presents our proposal and highlights how current NCL features can be used to control IoT devices. Section 4 discusses the proposed usage scenario and its implementation. Finally, Section 5 presents our conclusions and discusses some future work.

2. RELATED WORK

In this section, we present three proposals for the integration of multimedia and IoT: MPEG's Multimedia-centric IoT, ITU-T Immersive Experiences, and IBM's Node-Red. We then compare the three approaches and briefly discuss their limitations.

2.1 MPEG Multimedia-centric IoT

MPEG's Multimedia-centric IoT (MM-IoT) aims at enabling IoT applications to gather and manage media data, such as audio and video [8]. For instance, in a monitoring scenario, an MM-IoT application may use a network of cameras for transmitting audio-video data to a proxy device or central service. Other usage scenarios requiring audio-video functionalities include accident

In: Workshop Internacional de Sincronismo das Coisas (WSoT), 1, 2016, Teresina. Anais do XXII Simpósio Brasileiro de Sistemas Multimídia e Web. Porto Alegre: Sociedade Brasileira de Computação, 2016. v. 2.

ISBN: 978-85-7669-332-1

detection and surgery room, which may recognize and track objects or environment properties to inform status.

The key component of MM-IoT is the Media Thing (MIoT Thing or just MThing) [9], which is a special IoT device that can handle audio-video data. An MThing may perform tasks such as motion detection, media rendering, object tracking, and audio-video compression or transmission. Examples of MThings are cameras, media storage devices, displays, wearables, speakers, and microphones.

Figure 1 overviews how an MThings communicates with the user, the environment, and other MThings. Regarding users, MThings may offer interactive user interfaces (Figure 1-1) to enable users to control and configure MThings. In this case, users need to refer to the content, service, or product designers, which include the setup parameters and certain conditions to be met. Regarding the interaction with other MThings (Figure 1-3), an MThing should offer APIs and data interchange formats. Finally, regarding the interaction with the environment sensors or actuators, an MThing exchanges data formats from physical actuators.

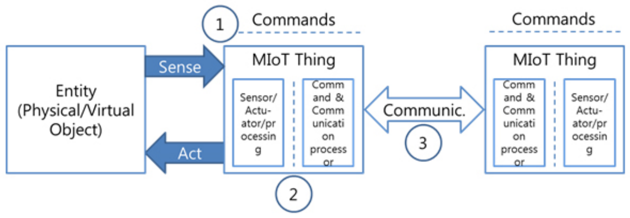

1
Commands
Commands
Entity (Physical/Virtual Object)
Sense
Act
MIoT Thing
Sensor/ Actu- ator/pr ocessin g
Comm and & Comm unicati on process or
Communic.
3
MIoT Thing
Comm and & Comm unicati on process or
Sensor/ Actu- ator/pr ocessin g
2

Figure 1: MM-IoT overview [9].

MM-IoT is an underway proposal. Currently, MPEG only proposes technologies for the MThing communication API. For audio-video data interchange among MThings (Figure 1-3), MPEG proposes MPEG-4 related formats. Examples of data handled by sensors (Figure 1-2) include: MPEG-V Part 3 sensorial effects, MPEG-V Part 5 sensed information, MPEG-V Part 2 sensor/actuator capabilities, and face descriptors using MPEG-7. Examples of the data (Figure 1-2) handled by actuators include camera position, display information, and MPEG-V Part 5 actuator commands.

2.2 ITU Immersive Experiences

Nowadays, sport and music events are commonly delivered to remote audiences. ITU argues, however, that most audiences want to have a better sense of immersion in the event and of togetherness with other fans. To provide that to remote audiences, ITU proposes Immersive Live Experiences (ILE) [10]. In sports events, for instance, it is possible to support better immersive experiences by displaying real-sized objects such as human bodies, or by changing the direction of the sound coming from cheers or players. In music events, directional audio speakers may reproduce the directional audio of instruments.

The overall goal of ITU ILE is to simulate the source event site (e.g., stadium, concert arena) in the viewer site (e.g. home, theater). The ILE architecture (presented in Figure 2) consists of *capturing environment*, *presentation*, and *transport* layer. A capturing environment may include technologies such as 4K/8K or 3D video capturing, directional microphones, and multiple sensors. Sensors may be used for real-time objects extraction (e.g., to detect the player's location in a game) or for capturing environment characteristics, such as lighting and odor. Presentation may include multi-screen displays, 3D projection, synchronous video, sound, and other actuators, such as lighting, smell, winds, and so on. Finally, the transport layer consists of media transport protocols that carry ILE application and its media assets. The ILE application may use 4K/8K and 3D media codecs and commands to relate media objects and sensorial devices.

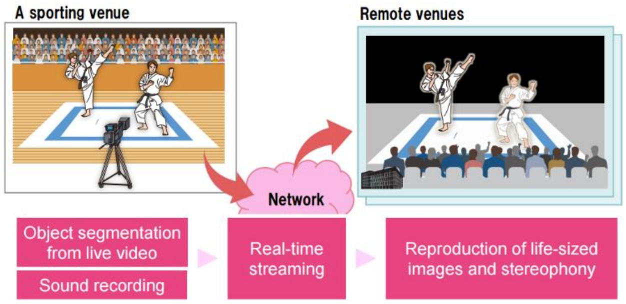

A sporting venue
Remote venues
Network
Object segmentation from live video
Sound recording
Real-time streaming
Reproduction of life-sized images and stereophony

Figure 2: ILE overview [10].

As the MM-IoT proposal, ILE IoT is also underway, ITU only suggests technologies for the processes above. In particular, a version of MPEG Media Transport (MMT) is considered for the live streaming. However, it is needed to extend some MMT capabilities for presenting real-sized objects at the terminal. At the viewing sites, the delivered content is reconstructed (in real size) through projectors and displays, audio is reproduced with sound direction, and lighting is reconstructed based on the received lighting information. By presenting them synchronously at viewing sites, the proponents of ILE IoT believe that users will feel highly realistic sensations.

2.3 IBM Node-RED

Commonly, IoT developers usually implement communication between objects using HTTP request/response or MQTT[1] publish/subscribe protocols, or even access web services such as e-mails, SMS or Twitter. IBM proposes a graphical IDE called Node-RED [12] to facilitate reusing code in such tasks. Figure 3 depicts Node-RED's user interface. The tool enables developers to create applications by dragging and dropping reusable code blocks (nodes) from a palette and connecting them in a control flow.

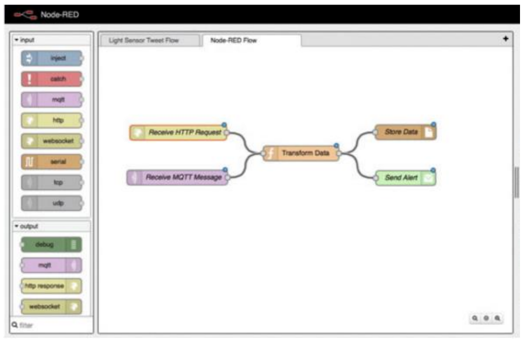


Figure 3: Node-RED screenshot [11].

Node-Red follows an event-processing approach, in which the flow relates the occurrence and triggering of events between its nodes. There are three types of nodes: input nodes (e.g., HTTP request) that generate events to flow; output nodes (e.g. HTTP response, e-mail) that consume events from the flow; and function nodes that

[1] http://mqtt.org/

process the data from input nodes or other function nodes. Figure 3, for instance, shows a control flow composed of two input nodes, one function node, and two output nodes. One of the input nodes is triggered when it receives an HTTP request, and the other is triggered when it receives an MQTT message. The function node processes the data generated by input nodes and send them to the output nodes. One of the output nodes sends an e-email and the other simply stores the received data.

The deployment of a Node-RED application is restricted to the IBM cloud service called BlueMix. Therefore, in order to be used and controlled by the deployed application, a node must be registered in BlueMix[2].

2.4 Discussion

Table 1 compares how MPEG, ITU, and IBM support the specification of relationships among media objects, sensors, and actuators. In MPEG MM-IoT, each device exchanges events with others devices. In ILE, the player in the remote audience controls the actuators based on the event transmission timeline. IBM's Node-Red supports the specification of the synchronism by enabling the developer to define a control flow relating events among the nodes. Node-Red, however, requires the use of IBM's BlueMix Cloud service.

Table 1: Approach summary

	Focus	Synchronism approach	Execution
MM-IoT	IoT Devices with AV resources	Event-driven	Distributed - handled by each device
ILE	Immersive events transmission for remote audience	Timeline	ILE Player
Node-RED	Iot Devices and services (reused code)	Event-driven	IBM BlueMix service

3. PROPOSED APPROACH

Different from the proposals discussed in Section 2, we investigate the use of an existing multimedia language, NCL, to specify the relationships between media objects, sensors, and actuators. Traditionally, multimedia languages focus primarily on audiovisual data. However, this is very limited when we consider the fact that more than 60% of human communication is nonverbal and uses a combination of our five senses (i.e., sight, hearing, touch, taste, and smell) [4]. We also study how one can use NCL to handle sensor and actuator tasks, allowing them to be used together with the ordinary synchronism and media abstractions of the language.

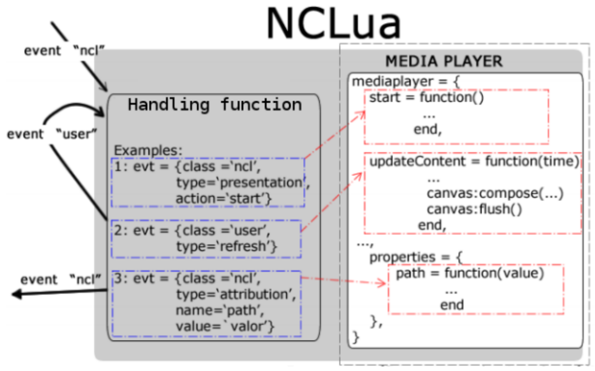
NCL is a DSL (Domain specific language) focusing on audiovisual media synchronization. NCL developers use audiovisual media content —such as video, audio and text— by defining <media> elements, and synchronize them through causal relationships defined by <link> elements. An NCL link performs actions over <media> elements (e.g. start, stop, or set) when some conditions are satisfied (e.g. when the media begins, ends, or a property is changed) [13].

NCL also enables authors to define additional tasks through Lua scripts, using the NCLua API [14]. For instance, using NCLua, authors can create HTTP requests, draw on canvas objects, and gather broadcast information (MPEG-TS metadata). An NCLua script is defined as a <media> element with type "application/x-ncl-NCLua" and can participate in <link>s just as any other media object. The communication between an NCLua object and the root NCL document is based on an event-driven API (i.e. the NCLua event module). The API allows an NCLua media object to listen to events being performed by the NCL document in the media object, and also allows the NCLua script to inform internal changes to the NCL document, which can be captured by <link>s in the NCL document. For instance, to react to a start action, an NCLua media object must register a handler function for the "start presentation" event. Similarly, to inform to the NCL document the ending of its presentation, an NCLua script must post an "end presentation" event.

In this work we investigate the use of an NCLua script to control IoT devices in NCL applications. Each device is handled by an NCLua media object that communicates with services in the device. The NCLua event-driven approach enables us to synchronize the device with other media objects in the NCL application (including audio, video, and other sensors and actuators). However, the code developed to our NCL application was extremely coupled with the IoT device used and its services. For instance, our initial code version use device's information such as IP address and service URL's inside the NCLua.To reuse our approach for different IoT devices, we follow the mechanism proposed by Sousa Junior [15] to implement new media players in NCLua.

The Sousa Júnior's proposal aims at covering the lack of native support for some media types in an NCL player implementation. For instance, an NCL player for the Brazilian Terrestrial Digital TV System [1] is not required to support SRT [16] subtitles or GIF [5] animated image format, two widely used formats. Similarly, we also want to support new media types, i.e., IoT devices.

Figure 4 depicts an overview of the Sousa Júnior's mechanism. The core element is a template function for handing common NCLua events (e.g. start) and property changes. These events are mapped to functions in a Lua table (the *mediaplayer* table). For instance, the template function handles the "start" of the NCLua media object by calling the function defined in the "start" fields of the *mediaplayer* table. Also, a "set" event in a property triggers the function defined in "properties.propName" field of the *mediaplayer* table, in which *propName* should be replaced by the name of the property that is being changed. Thus, for creating a new media player in NCLua, the programmer just needs to fill in the *mediaplayer* table.



```
Examples:
1: evt = {class ='ncl',
          type='presentation',
          action='start'}
2: evt = {class ='user',
          type='refresh'}
3: evt = {class ='ncl',
          type='attribution',
          name='path',
          value='valor'}
```

```
mediaplayer = {
  start = function()
       ...
     end,
  updateContent = function(time)
       ...
       canvas:compose(...)
       canvas:flush()
     end,
 ...,
  properties = {
    path = function(value)
         ...
       end
  },
}
```

Figure 4: New player as an NCLua

In our work, we implemented a "mediaplayer" table with functions that access variables (using the REST API) maintained by IoT devices. Therefore, a developer who wants to use our implementation should expose the device variables as properties in

[2] www.ibm.com/cloud-computing/bluemix

the NCLua <media> element. More precisely, we define the following variables: "base_url", the base address of the device REST API; *ad-hoc* names (e.g.smellStatus), the names maintained by the device; the same *ad-hoc* names followed by a ".url" suffix (e.g. smellStatus.url) to define the REST URL; and the same *ad-hoc* names followed by a ".interval" suffix (e.g. distance.interval) to define interval of GET requests to update the value.

4. USAGE SCENARIO

We have implemented an NCL application that presents sensorial effects (light, air flow and scent) synchronized with video objects. This scenario is very similar to 4D cinema rooms, such as some attractions from DisneyWorld[3]. Our usage scenario presents three video objects displayed in sequence (Figure 5). At the moment the presentation of each video object starts, the application starts an actuator to produce a sensorial effect related to the video content.

The first video depicts a sun rising in a city; when it begins, the application turns on the light of lamp (Figure 5-A). The second video depicts a flight over a lavender field; when it begins, the application turns on a scent dispenser with lavender fragrance (Figure 5-B). The third video depicts a parachute jump; when it begins, the application turns on a fan (Figure 5-C). The turning on of the scent dispenser and of the fan considers the user position, captured by a distance sensor. More precisely, if the user position is greater than 1m, the application anticipates the activation of wind and scent actuators, so their effect can be felt by the user at the moment the corresponding video begins. Otherwise, if the user position is smaller than 1m, no change is applied.

In order to implement the scenario, we developed an NCL application and a JavaScript code that control the IoT device with the required sensors and actuators. Both code fragments[4] are detailed next.

The NCL application (shown in Listing 1) uses four <media> elements. Three <media> (lines 1-6) specify the videos and one <media> (lines 7-18) specifies the NCLua object for managing the IoT devices. The last one defines the variables handled by the devices (lightStats, airStatus, scentStatus, distance) and its base REST URL. In particular, the "distance" variable is defined to be updated every second.

The NCL application behavior is defined by two groups of links. The first group defines the sequence of videos. The second activates the respective actuator of each video. To do so, it tests the "distance" property and, if necessary, it anticipates the activation of the scent dispenser and of the fan. The sensorial effects are controlled by the <media> properties "lightStatus", "airStatus", and "scentStatus". Changes on those properties corresponds to PUT requests to the IoT device. Additionally, the "distance" property is updated each second, also by an HTTP request to IoT device.

```
 1 <media id="air" src="media/air.mp4"
 2        descriptor="dsFull"/>
 3 <media id="scent" src="media/scent.mp4"
 4        descriptor="dsFull" />
 5 <media id="light" src="media/light.mp4"
 6        descriptor="dsFull" />
 7 <media id="device" src="rest_device.lua">
 8  <property name="base_url" value="139.82.95.37/api"/>
 9  <property name="lightStatus"/>
10  <property name="lightStatus.url" name="/light"/>
11  <property name="airStatus"/>
12  <property name="airStatus.url" name="/air"/>
13  <property name="scentStatus"/>
14  <property name="scentStatus.url" name="/scent"/>
15  <property name="distance"/>
16  <property name="distance.url" name="/distance"/>
17  <property name="distance.interval" value="1s"/>
18 </media >
```

Listing 1: NCL code fragment to define the IoT device.

Regarding the IoT device, our prototype is based on an Intel Galileo[5] with a Seeed Studio Grove kit[6], which allowed us to easily connect the sensors and actuators hardware (show in Figure 6). To emit scents, we use an Air Wick Freshmatic scent dispenser[7]. We can control it using a digital port since it is a low energy consumption device (two 1.5V batteries). To simulate light and wind, we use domestic lamp and fan. We control them via two relay devices connected to the GroveKit. To capture the user's position, we used an LV-MaxSonar[8] connected to an analog port.

Listing 2 shows the JavaScript code to control the device. We developed it using the Intel XDK IoT edition and Intel libraries (e.g., MRAA[9]). Lines 1-8 show the initialization and change function for the relay for the lamp (digital port #6), lines 9-16 show the initialization and change functions for the fan (digital port #3), and lines 17-38 show the initialization and change function for the scent dispenser (digital port #8). The change of the scent dispenser is not a continuous operation because of the strength of its scent; the full operation takes about 6 seconds. Finally, lines 39-44 show the initialization of the sonar sensor, whose value is updated every second. The change functions and the

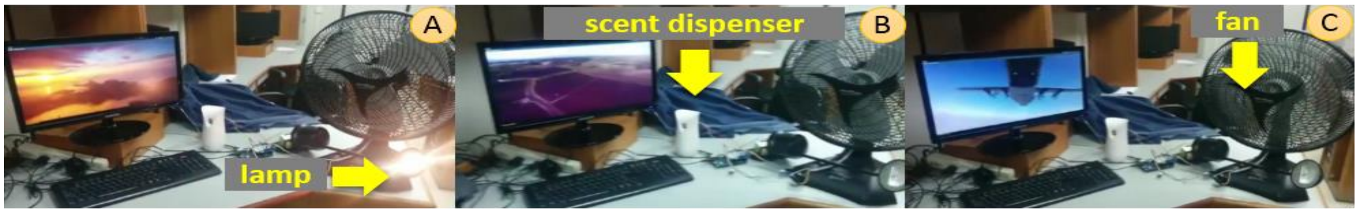


Figure 5: Light, scent and wind effects in different videos of the NCL application

[3]disneyworld.disney.go.com/pt/attractions/epcot/soarin/
[4]instructables.com/id/Sensorial-Galileo-for-Video-based-Applications-Usi/
[5]intel.com/content/www/us/en/do-it-yourself/galileo-maker-quark-board.html
[6]seeedstudio.com/item_detail.html?p_id=1978
[7]amazon.com/Air-Wick-Freshmatic-Automatic-Freshener/dp/B007XTLT8I
[8]maxbotix.com/Ultrasonic_Sensors/MB1010.htm
[9]github.com/intel-iot-devkit/mraa

sonar distance value are offered by a REST API using the express library[10].

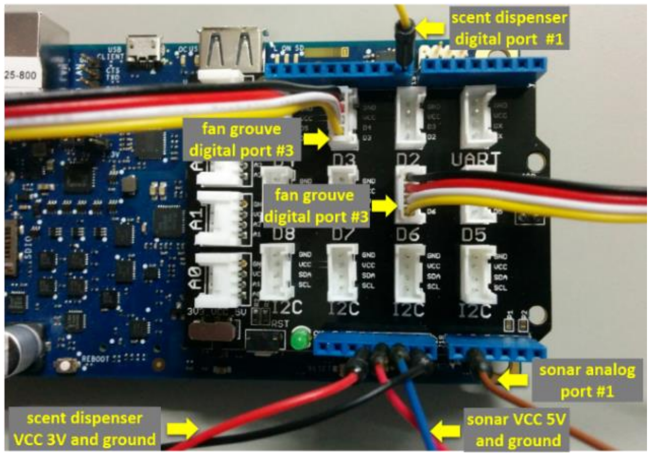

scent dispenser digital port #1
fan grouve digital port #3
fan grouve digital port #3
sonar analog port #1
scent dispenser VCC 3V and ground
sonar VCC 5V and ground

Figure 6: Intel Galileo connections

```
// --> lamp
var myRelay = new groveSensor.GroveRelay(6);
var lightStatus = 0; myRelay.off();
function changeLightStatus(newStatus){
  if(lightStatus == newStatus) return;
  lightStatus = newStatus;
  lightStatus ? myRelay.on() : myRelay.off();
};
// --> fan
var myRelay2 = new groveSensor.GroveRelay(3);
var airStatus = 0; myRelay2.off();
function changeAirStatus(newStatus){
  if(airStatus == newStatus) return;
  airStatus = newStatus;
  airStatus ? myRelay2.on() : myRelay2.off();
};
// --> scent
var myEnergy = new mraa.Gpio(8);
myEnergy.dir(mraa.DIR_OUT);
var scentStatus = 0; myEnergy.write(0);
var scentInterval;
function startScent() {
  if(myEnergy.read()) return;
  myEnergy.write(1);
  setTimeout(function () {
    myEnergy.write(0);
  }, 200);
}
function changeScentStatus(newStatus){
  if(scentStatus == newStatus) return;
  if(scentStatus == 0){
    startScent(); scentStatus = 1;
    scentInterval = setInterval(startScent, 6000);
  }else{
    clearInterval(scentInterval);
    myEnergy.write(0); scentStatus = 0;
  }
};
// --> sonar
var mySonar = new MaxSonarEZ.MAXSONAREZ(1, 5.0);
var distance = 0;
var myInterval = setInterval(function() {
    distance = mySonar.inches() * 2.54;
}, 1000);
```

[10]expressjs.com

Listing 2: Intel Galileo code fragment for controlling sensors an actuators

5. FINAL REMARKS

In this paper, we investigate the use of the NCL language to handle sensors and actuators. We were able to satisfactorily use NCL to synchronize media objects, sensors, and actuators. As a usage scenario, we implemented an NCL application that presents video objects synchronized with sensorial effects (light, air flow and scent). In particular, the turning on and off of wind and scent actuators, considering the user's position captured by a distance sensor, were implemented and tested.

We envision four main paths for future work. First, we can evaluate use discovery mechanisms of devices in a local networks, such as UPnP and Bounjour. Second, we can also investigate using different types of sensors and actuators, which may be useful for scenarios such as assisted living and monitoring. Third, we can improve our NCLua implementation to enable it to be reusable to control other IoT devices. Third, following a path similar to Node-RED, we may extend the NCL Composer [3] tool to show IoT device as elements. This way, we may give developers a graphical tool to create multimedia-based IoT applications.

ACKNOWLEDGMENTS

First, we are strongly thankful to Prof. Luiz Fernando Gomes Soares (*in memoriam*) for the profound guidance and friendship, essential to this work and its authors. We also thank all TeleMídia and SecondLab researchers, who provided thoughtful discussions on this work. Finally, we thank the Brazilian National Council of Technological and Scientific Development (CNPq) for their financial support (project #309828/2015-5).

REFERENCES

[1] ABNT NBR 15606-1:2016 Digital terrestrial television - Data coding and transmission specification for digital broadcasting Part 1: Data coding specification. 2016. *www.abntcolecao.com.br/normavw.aspx?ID=351835*. Accessed: 2016-08-08.

[2] Atzori, L., Iera, A. and Morabito, G. 2010. The internet of things: A survey. *Computer networks*. 54, 15 (2010), 2787–2805.

[3] Azevedo, R.G.A., Araújo, E.C., Lima, B., Soares, L.F.G. and Moreno, M.F. 2014. Composer: meeting non-functional aspects of hypermedia authoring environment. *Multimedia Tools and Applications*. 70, 2 (May 2014), 1199–1228.

[4] Ghinea, G., Timmerer, C., Lin, W. and Gulliver, S.R. 2014. Mulsemedia: State of the Art, Perspectives, and Challenges. *ACM Transactions on Multimedia Computing, Communications, and Applications*. 11, 1s (Oct. 2014), 17:1–17:23.

[5] GIF Graphics Interchange Format. 1987. *www.w3.org/Graphics/GIF/spec-gif87.txt*. Accessed: 2016-08-08.

[6] ISO/IEC 23005-1:2014 - Information technology -- Media context and control -- Part 1: Architecture. 2014. *http://www.iso.org/iso/catalogue_detail.htm?csnumber=60359*. Accessed: 2016-08-08.

[7] ISO/IEC 23008-11:2015 - Information technology -- High efficiency coding and media delivery in heterogeneous environments -- Part 11: MPEG media transport composition information. *http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=65302*. Accessed: 2016-07-20.

[8] ISO/IEC JTC 1/SC 29/WG 11 N15085: Exploration on Media-centric Internet of Things. 2015.

http://mpeg.chiariglione.org/standards/exploration/media-centric-internet-things. Accessed: 2016-08-08.

[9] ISO/IEC JTC 1/SC 29/WG 11/ N16346: Requirements on Internet of Media-Things and Wearables. 2016. *http://mpeg.chiariglione.org/standards/exploration/internet-media-things-and-wearables/requirements-internet-media-things-and*. Accessed: 2016-08-08.

[10] ITU-T SG 16 (Study Period 2013) Contribution 1190: New Proposal of establishment new Focus Group and new Question on Immersive Live Experience (ILE) services. 2016. *https://www.itu.int/md/T13-SG16-C-1190/en*. Accessed: 2016-08-08.

[11] Javed, A. 2016. Complex Flows: Node-RED. *Building Arduino Projects for the Internet of Things*. Apress. 51–73.

[12] Node-RED. *http://nodered.org/*. Accessed: 2016-03-03.

[13] Soares, L.F.G., Moreno, M.F., Neto, C. de S.S. and Moreno, M.F. 2010. Ginga-NCL: Declarative Middleware for Multimedia IPTV Services. *IEEE Communications Magazine*. 48, June (Jun. 2010), 74–81.

[14] Soares, L.F.G., Moreno, M.F. and Sant'Anna, F. 2009. Relating Declarative Hypermedia Objects and Imperative Objects Through the NCL Glue Language. *Proceedings of the 9th ACM Symposium on Document Engineering* (New York, NY, USA, 2009), 222–230.

[15] de Souza Junior, J.G., Azevedo, R.G. de A., Neto, C. de S.S. and Soares, L.F.G. 2010. Estendendo NCL : objetos NCLua como exibidores para novos tipos de mídia. *WebMedia '10: Proceedings of the 16th Brazilian Symposium on Multimedia and the Web - II Workshop de TV Digital Interativa - WTVDI* (2010).

[16] SRT Subtitles. *http://www.matroska.org/technical/specs/subtitles/srt.html,*. Accessed: 2016-08-08.