Implementation of a self-sustained multi-sensors IPv6 sleepy device for the Internet of Things

Darko Petrovic, *Member, IEEE*

***Abstract*— This paper presents the implementation of a self-sustained IPv6 device embedding multiple sensors. The research focuses mainly on the implementation of a sleepy node using the recent Internet protocols for Wireless Sensor Networks (WSN). Based on the popular Contiki operating system, the protocol stack has been enhanced by implementing the optimized version of the Neighbor Discovery Protocol (RFC6755). The usage of the Low Power Listening (LPL) of the RDC layer has been reduced improving notably the current consumption of the device. The device is equipped with a rechargeable battery and a small solar panel allowing to harvest ambient light while performing measure of the sensors. The results have shown that the optimized version of the ND protocol can saves up to 15 times more energy per day than the standard NDP. The devices communicate at the application layer over the OMA LwM2M protocol and a web application has been implemented for the interaction with the nodes. The device consumes 15µW in power down mode and about 100µW while performing sensors measurement at a reasonability short interval. The device can last about 50 days in the dark on a battery of 50mAh and a light condition of 1000 lux in average is require to sustain the mean power consumption of the device while reporting sensors measurement.**

***Index Terms*— Contiki, Internet of Things (IoT), IPv6, NDP, LwM2M, WSN**

I. Introduction

Wireless Sensor Network (WSN) is an emerging domain and IPv6 is one of the core technologies to form the future networks. WSN are composed of a large number of sensor nodes that are generally powered by batteries that must be purchased, maintained and replaced. Energy harvesting presents a straightforward solution for easily powering these remotes device by using clean energy. Many different types of energy harvesting technologies exist, such as solar, vibration, wind, piezoelectric, thermoelectric and so on. The most popular source of environmental energy is the sun. One reason why solar energy is becoming more widely used is that it has a higher power density (about 15mW/cm^3) than other renewable energy source, which enables wireless sensors nodes to be completely self-sustained. Connecting a solar powered IPv6 device to the Internet is a challenging task and not feasible unless a storage element is available allowing the device to run during the night. Although ceramic capacitors have an infinite lifetime and are simpler to recharge, the energy density of these storage elements (1 to 10 J/cm^3) is too small compared to rechargeable batteries (1000 J/cm^3) and thus not feasible to use in our application where the device must be able to run during several days in the dark. Although the shelf lifetime of secondary batteries is degrading over time due to the recharge cycles, they are still more suitable nowadays compared to primary batteries in term of environmental impact, especially considering that there will always be more of connected devices in the future.

The purpose of this research was to implement an IPv6 network of self-sustained devices and routers incorporating several sensors and by lowering the current consumption as much as possible to improve the lifetime of the device. Since most of the energy spent by a WSN device is during radio transmission, it was of our main concern to reduce the radio activities in all layers of the network protocol stack. Our research focus mainly on the implementation of the sleepy devices which by definition doesn't take part in the routing protocol of the network and therefore can sleep most of the time to save its battery.

The remainder of the paper is organized as follows. In the following section we present the hardware of our IPv6 device. In Section III we present the software with the network protocol stack and compare the energy performance of the standard and optimized version of the NDP protocol. In section IV we show our Web Application for the device management. We evaluate the overall power consumption of the device in Section V. Finally our conclusion is explained in Section VI.

II. IPv6 Platform

Our self-sustained IPv6 device prototype is shown in Fig. 1. The processor running on the platform is the CC2538 from *Texas Instrument* which is an ARM Cortex-M3 cadenced at 16MHz. The platform integrates the following sensors: temperature, humidity, pressure, light, a motion detector and a voltage/current measurement unit. The power of each sensor is controlled individually by the microprocessor reducing the power consumption of the system while in deep sleep mode. An I^{2}C switch separates the connection of each I^{2}C interface from the sensors chips and thus prevents these latter to be power-on from the bus. The gain of current consumption in deep sleep thanks to the I^{2}C switch is in order of 25µA allowing the board to consume only 4.9µA when in deep sleep mode and 6.4µA with the motion detector enabled.

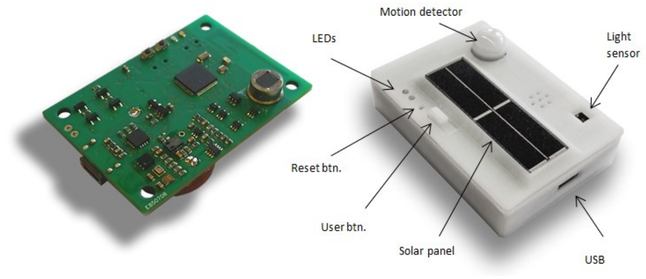

Fig. 1. Proposed platform.

The solar panel SLMD121H04L is made of 4 single cells of monocrystalline in series with a general size of 43x14mm and provides an efficiency of 22%. This solar panel was chosen because it can provides 40µW of power in low light condition (100 lux) which is sufficient to sustain the power consumption of the device while in deep sleep and 400µW at 1000 lux to sustain the average power consumption of the system while performing measures. Since the output power is linearly proportional to the amount of light, we calculated the slope of the curve to be approximately 0.4µW/Lux. The battery used is the VL-2330, a Vanadium Pentoxide Lithium coin battery, with 23mm in diameter, a rated voltage of 3.0V and 50mAh of nominal capacity. The battery can last about 50 days with a current consumption of 50µA. The board provides an USB connector from which the battery can be recharged and the device to be configured for the first usage. As the USB power is connected to the same input as the solar panel, a load switch deactivates automatically the MPPT of the power management unit when the USB is connected. The Fig. 2 shows the bloc diagram of the device.

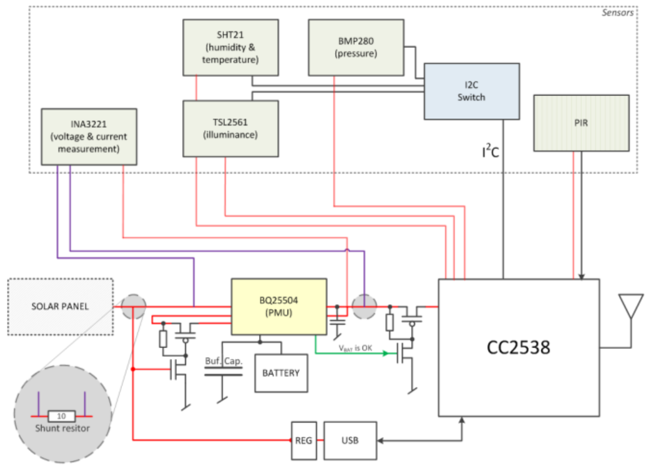

Fig. 2. Bloc diagram of the device.

III. Software and Energy Consumption Optimization

The sensor is running Contiki-OS v3.0 [1], a free and open-source operating system for the Internet-Of-Things. The core of Contiki provides the IP communication through a complete network protocol stack. The rest of the system is implemented as application libraries that are optionally linked with the program. The network protocol stack we're using in our project is illustrated in Fig. 3.

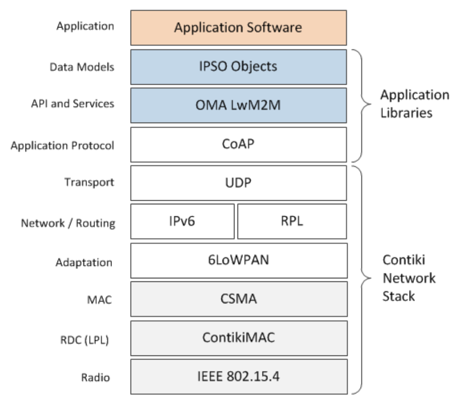

Fig. 3. Network protocol stack covering all traditional OSI layers.

The first main objective in our project was to enable the sleepy device behavior for the node which joins the network as simple host. To achieve the best energy performance for the host behavior, some changes had to be made on the network protocol stack. The first one is the implementation of the Optimized Neighbor Discovery Protocol (RFC6775) [2] which introduces the concept of sleeping devices for IP communications. In [3] the protocol is evaluated in Cooja simulator against its counterpart Neighbor Discovery Protocol (NDP) in term of RS/RA messages exchange. It has been conclude that the protocol reduces the number of RS/RA exchange ratio messages in the network by 80%. We show in this document the benefit in term of energy consumption when using this optimized protocol by considering both RS/RA and NS/NA messages exchange. We've furthermore enabled the mesh networking by activating the RPL protocol [4] within Contiki and made some adjustments to the implementation to work together with the optimized NDP.

Finally, to take full advantage of the Optimized NDP, we've implemented a smart RDC mechanism which deactivates the permanent idle listening (ContikiMAC) for the host when not required. Many researches have been carried out to improve the energy waste of the idle listening by implementing efficient MAC protocol but these researches are mainly focusing on the routing problem for the router behavior [5–7]. We don't address this problem in this document and keep the RDC activated all the time for the router which therefore requires to be powered by an USB cable. For the host the optimization of the idle listening is straightforward because it doesn't take part in the routing and every communication is initiated by itself.

A. *The Low Power Listening and the Smart RDC*

The Low Power Listening (LPL) is a common technique in Wireless Sensor Network for reducing energy consumption where nodes periodically wakes-up from the low power mode to sample the wireless channel and detect radio activity with the Clear Channel Assessment (CCA) mechanism. The actual implementation of the LPL mechanism in Contiki turns on the RDC when the device boots and it is never turned off. When activated all the time, this periodic wake-up of the radio is a waste of energy especially for nodes which doesn't take part in the routing within a mesh network. Furthermore, with the implementation of the optimized NDP which introduces the concept of sleeping device the deactivation of the LPL is more justified. The average current consumption when the LPL is activated and when the device is in deep sleep was measured and reported in Fig. 4. The theoretical excepted lifetime of the device while the LPL is activated is 1.3 years with 2xAA batteries and only 5 days with our battery of 50mAh.

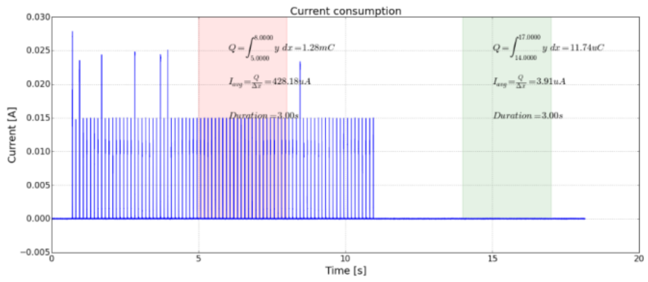

Fig. 4. When activated the LPL consumes 428µA on average whereas in deep sleep mode the average current is only 4µA.

The code source of Contiki-OS was slightly modified to support the deactivation of the ContikiMAC RDC. For a sleeping device, the LPL is required only when the node is expecting a response from a request. TABLE 1 lists the type of messages that expect a response with the current protocol stack. Every time the node is making a request of the type listed in TABLE 1, the LPL is activated for 3 seconds. If no response is received during these 3 seconds the LPL is prolonged for 1 second until the response is received or the number of maximum attempt is reached.

TABLE 1
TYPE OF MESSAGE WITH THE EXPECTED RESPONSE

Protocol	**Message**	**Expected response**
NDP	RS	RA
NDP	NS	NA
RPL	DIS	DIO
RPL	DAO	DAO ACK
CoAP	CON	ACK
CoAP	2.05 Content (M=1)	GET

B. *The optimized NDP*

The optimized ND process [2] provides support for sleeping host by making the interaction between the host and router a host initiated interactions and thus avoids multicast flooding of message and improves the interaction between sleeping host and routers. The IPv6 neighbor discovery in its basic form specified in RFC 4861 [8] was not designed for asymmetric reachability between the nodes in the network: a sleeping node is by definition not reachable at any time which makes the protocol inefficient.

The basic concept of the optimized NDP is that a host (6LN) registers itself to a router (6LR) for a certain duration during which the host can go to sleep. During this registration period the host doesn't perform Neighbor Unreachability Detection (NUD) to actively keep track of neighbor's changes. With the optimized NDP, a host speaks only to a router and uses the registration mechanism to keep the connection with this later. A device configured as a router registers itself equally as the host does and update its registration to its parent router too.

The optimized NDP exchanges the same type of messages as its counterpart NDP but with two new options in the messages: the Authoritative Border Router Option (ABRO) and the Address Registration Option (ARO). When joining the network, the host or the router first sends a multicast Router Solicitation (RS) in order to find a parent router. A nearby router responds with a unicast Router Advertisement (RA) containing among others the new ABRO option. This new option contains the information relatives to the border-router which are its IP address, the version and its lifetime. Then the node send a Neighbor Solicitation (NS) containing the ARO option with the registration lifetime during which it'll be registered to the router. The router responds to the node with a Neighbor Advertisement (NA) containing the result of the registration. Note that there is two other messages not mentioned here which are introduced in the optimized NDP. The Duplicate Address Detection (DAD) and Duplicate Address Confirmation (DAC) messages. Both of these messages are used by the router to verify with the border-router, before accepting a registration, that the address trying to register is not already used by another node in the network. These messages are omitted here since we are using EUI-64 addresses which are supposed to be unique.

Once the registration is a success, the node send periodically a new registration with an NS just before the registration expires and an RS when the router, prefix or border-router lifetime is about to expires if not updated meanwhile.

1) *Energy evaluation of the standard NDP*

In this section we evaluate the energy spent by the protocol while exchanging the periodic messages. We measure the current consumed by the device during this process and extrapolate the data to compute the total charge consumed during a full day of activity.

In the standard NDP a router send unsolicited multicast RA message periodically or when solicited by a RS message. With ContikiMAC, a multicast message is repeated during the whole RDC period so that every node receives the message. The periodicity of the RA message is not fixed by the protocol but can vary from 3 to 1800 seconds with a default value of 600 seconds in Contiki-OS. The exchange of the NS/NA messages is equally "periodic" assuming that there is a periodic communication between the nodes. In fact, a neighbor's interface is REACHABLE for a specific duration after which the interface goes in the STALE state. When a node communicates to another node with the interface in the STALE state, the Neighbor Unreachability Detection (NUD) algorithm is started to verify the reachability of this interface by exchanging NS/NA message. The reachable time of an interface varies from 5 to 15 minutes and it is determined when the interface is enabled. The number of exchange must be multiplied by the number of interface.

Fig. 5 and Fig. 6 show the current consumption of the device while sending multicast RS/RA message and performing NUD with NS/NA message respectively.

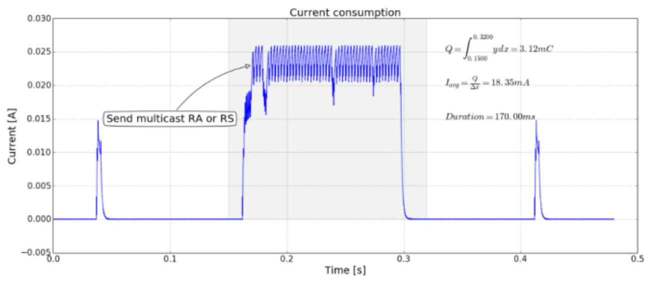

Fig. 5. Current consumption during a multicast RA or RS message. The message is repeated by ContikiMAC during the full period of the RDC which is here 125ms.

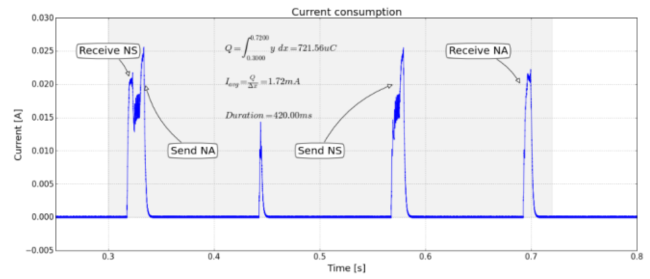

Fig. 6. Current consumption during NS/NA message exchange for an interface.

Using the current consumption measured in Fig. 5 and Fig. 6 we compute the total charge consumed by the exchange of the periodic ND messages for different intervals over a period of 24 hours. The number of occurrences for the NS/NA messages is doubled because there is a minimum of 2 interfaces configured in the device for the local and global address. Running on a battery with a capacity of 50mAh, the energy spent due to the NUD in the worst case is 0.23% per day of the total charge for the host and 0.48% per day for a router.

TABLE 2
WASTE OF ENERGY OVER 24 HOURS ON A BATTERY OF 50MAH FOR PERIODIC MESSAGES EXCHANGE IN THE STANDARD NDP.

	Interval (in min.)	**Nb. Occ.**	**Total charge**	**Waste of energy**
NS/NA (2 interfaces)	15	192	138mC	0.077%
	10	288	207mC	0.115%
	5	576	415mC	0.23%
RA	30	48	150mC	0.083%
	10	144	450mC	0.25%
RS	60	24	9mC	0.005%

2) *Energy evaluation of the optimized NDP*

With the optimized neighbor discovery protocol there is no more periodic unsolicited multicast RA message neither periodic NUD with NS/NA message exchange between the interfaces. The current consumption is consequently reduced. The router and the host behaves similarly because both registers them self to a parent router and periodically update the registration before expiration. The unit of the ARO option is in minute, therefore the minimum registration lifetime is 1 minute in the worst case. There is no limit for the maximum value expects the size of the field which is 16-bits.

The comparison between the two protocols is not straightforward because the different lifetimes can vary from implementation to implementation but using the values from TABLE 2 and TABLE 3, in the best case, the host spends 15x less energy per day with the optimized NDP considering only the NS/NA messages exchange.

TABLE 3
WASTE OF ENERGY OVER 24 HOURS ON A BATTERY OF 50MAH FOR PERIODIC MESSAGES EXCHANGE IN THE OPTIMIZED NDP.

	Reg. lifetime (in min.)	**Nb. Occ.**	**Total charge**	**Waste of energy**
NS/NA	60	24	9mC	0.005%
	15	96	36mC	0.02%
	1	1440	534mC	0.3%
RS	60	24	9mC	0.005%

IV. DEVICE MANAGEMENT

To manage the sensors, on top of CoAP [9], we use the Open Mobile Alliance (OMA) Lightweight Machine To Machine (LwM2M) protocol [10]. CoAP provides a request/response interaction between application endpoints and includes key concepts of the Web such as URIs and Internet media types but doesn't ensure interoperability on the application layer. OMA LwM2M was designed for this purpose and respond to the demand of a common standard for managing low power devices in constrained networks.

We're using the available implementation of the OMA LwM2M standard within Contiki 3.0 and add our own Objects and Resources for our device. The implementation provides a basic registration mechanism to the server. We complete the registration process with the server and implement the Registration Update mechanism which is the key feature to communicate with our sleepy node since we deactivate the LPL. Our sleepy node leverage completely the Registration Update mechanism of the LwM2M protocol by allowing an asynchronous communication with this later without adding more complexity to the system by implementing a proxy or any other non-standardized mechanism.

1) *Web Interface*

The open source applications for the server side are still not widely available because the standard is relatively recently published. Leshan [11] is a popular LwM2M server implemented in Java which is based on the Californium project for the CoAP implementation. Wakaama [12] and AwaLwM2M [13] are both implemented in C and provide API to build a server without the need of an intimate knowledge of the M2M protocol. Wakaama is based on the Erbium CoAP library as Contiki but the library was modified to run on Linux. AwaLwM2M is using the libcoap library.

To manage a bunch of devices one need a GUI and Leshan is the only one that actually provides a Web interface to interact with LwM2M Clients. To responds to the needs of our project, we've developed our own web interface with NodeJS and use the *lwm2m-node-lib* [14] module for the LwM2M server. The module implements the Client

Registration, the Device Management & Service Enablement and the Information Reporting interfaces.

We've modified the core functionality of the module by adding the possibility to queue the operations in destination to a sleepy device and thus complete the Registration Update mechanism on the server side. All messages in destination to the sleepy device will be delivered when this later wakes-up and updates its registration to the server.

To save even more energy of the sleepy device, the web application keep in a database the resource associated to a device based on its endpoint name. Then the next time a similar device register to the server, the application is aware of the resource available on the device and thus there is no need to discover the resources again.

From within the web interface, the user is able to discover the resource available on the device, read/write resource and set an observation for any observable resource.

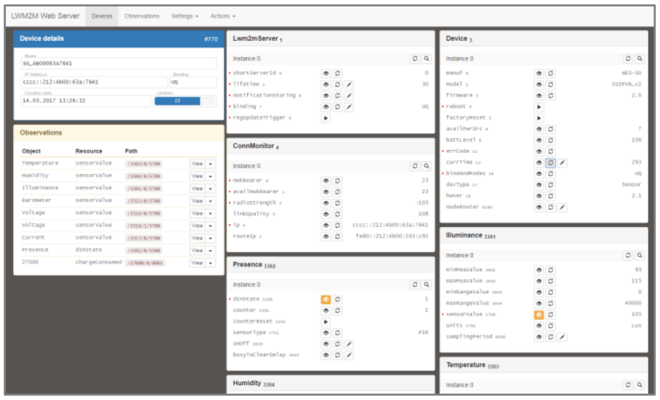

Fig. 7. The LwM2M Web Application allows controlling the device and to setup the observation for the resources.

V. Power consumption analysis

The power consumption of the device is determined by analyzing the current consumption of the periodic tasks running while performing measures via the Information Reporting of LwM2M protocol. The total charge Q and the average current during each task are reported in TABLE 4.

TABLE 4
TOTAL CHARGE C CONSUMED BY EACH POSSIBLE PERIODIC EVENT WITHIN THE DEVICE.

Periodic event	**Charge Q**	**Duration**	**I_{avg}**
Temperature measure	324 µC	126 ms	2.57 mA
Humidity measure	296 µC	80 ms	3.70 mA
Illuminance measure	384 µC	440 ms	0.87 mA
Pressure measure	340 µC	135 ms	2.52 mA
Battery voltage measure	359 µC	282 ms	1.27 mA
Solar voltage measure	359 µC	282 ms	1.27 mA
Solar current measure	266 µC	31 ms	8.6 mA
Presence detection	240 µC	100 ms	2.4 mA
Energest reporting	290 µC	170 ms	1.71 mA
ARO Registration	371 µC	170 ms	2.18 mA
LwM2M Registration Update	5 mC	10 s	0.5 mA

The charge consumed is almost identical for every sensor measurement and approximate 300µA in average. Each task consists mainly to wake-up the processor, process the event, power on the sensor, start the measure (during which the processor goes in deep sleep mode) and finally send the data over the radio. The LwM2M Registration Update consumes the most because the LPL is activated for 10 seconds. The duration of the LPL activation was chosen actually sufficiently high so that all configuration parameters from the server are received at once on the client and equally to give sufficient time for the farthest device from the sink to receive every message. This parameter can be optimized to reduce even more the current consumption.

To determine the lifetime of the device we compute the average current consumption of the device while performing sensors measurement at different intervals. The same period of measurement was set for each sensor and assuming a presence detection at the same interval. The average current consumption I_{avg} is reported in Fig. 8. With an interval of 5 minutes, the average current consumption is 32µA while performing a Registration Update every 5 minutes. The device can last on a battery of 50mAh (using 80% of the capacity) theoretically about 54 days (1300 hours).

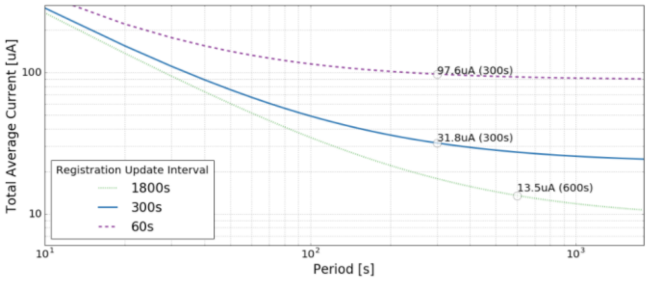

Fig. 8. Average current consumption of the device while performing sensor measurement at different period (same for all sensors) with a fixed ARO Registration period of 1 hour and LwM2M Reg. Update at 60, 300 and 1800 seconds.

A. *Battery recharge performance*

To determine the performance of the battery recharge and thus the sustainability of the device, we logged the data from several devices over a period of approximately 2 months while these latter are placed at different locations in the room and exposed to different light condition. Thanks to the illuminance sensor and the power measurement chip mounted on the board we're able to establish a relation between the incoming power from the solar panel, the light condition and the recharge of the battery. In Fig. 9 we plotted the power generated by the solar panel against the light intensity. We measured the power coming from the solar panel: when this later is connected in series with a schottky diode (W/ diode) which was used to prevent the current from the USB cable going in the solar panel when this latter is plugged in, when the schottky diode was removed (W/h diode) and finally when the solar panel was doubled and connected in parallel (2 panels).

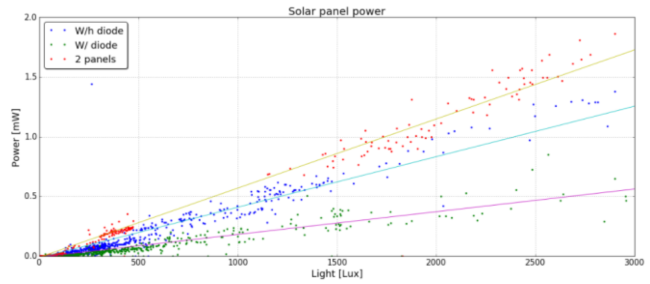

Fig. 9.Power generated by the solar panels at different light intensity.

We first notice in Fig. 9 that the schottky diode reduces by a factor of 50% the performance of the solar panel. While the measured voltage is approximately the same when the diode is present or not, the current on the other hand is reduced by half when the diode is connected. This is likely due to the functioning of the boost converter and the MPPT mechanism which cannot perform well with the diode. The double panel configuration however doesn't double the incoming power. This latter provide only x1.5 more power. More tests and measurement are required to validate these finding. Finally the plot shows the repartition of the instantaneous power generated by the solar panels and the light intensity which sit in the range of 10 to 300μW and 0 to 500 lux respectively most of the time.

In Fig. 10 is shown the voltage of the battery against the solar power measured at the same time. The plot thus denotes the performance of the solar panel to elevate the voltage of the battery at a specific level. The elliptic curve extracted from the data shows that the battery voltage will never decrease below 2.8V with an incoming power of at least 400μW. Knowing that the steady state of the battery is around 2.7V before this latter begins the downward curve, the ideal constant power from the solar panel to guarantee the self-sustainability of the device is between 200μW to 400μW. This is furthermore confirmed by considering the equation below with approximately 100μW for P_{system}, 60% to 80% for the efficiency of the power management unit (η_{PMU}) and 200μW for P_{solar}.

$$P_{solar} \cdot \eta_{PMU} > P_{system}$$

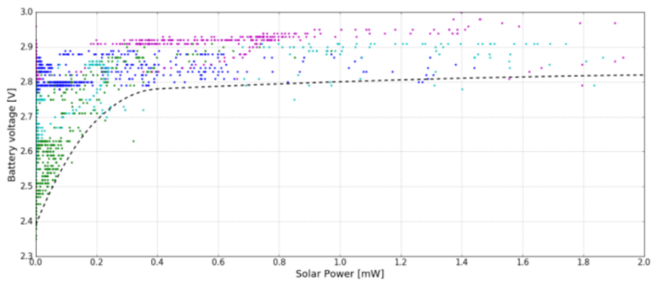

Fig. 10. Performance to elevate the battery voltage with the incoming solar panel power.

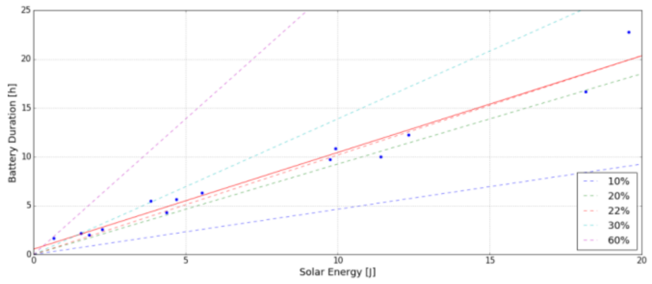

Fig. 11. Measurement of the overall performance to retain the stored energy.

Finally we calculated the performance of the battery to retain the stored energy of a recharge. To do that we calculated how long it takes for the battery to reach the level it has at the beginning of the recharge. The results in Fig. 11 show that for 1 Joule of energy stored, the battery takes approximately 1 hour to reach its initial voltage. The datalog was realized with devices running with a power consumption of about 60μW in average. The plot shows a linear relationship between the duration of the voltage decrease and the amount of stored energy but not at the expected slope. The results are compared with theoretical and "supposed" durations of battery decrease for an average consumption of 60μW and for different conversion efficiency (dashed lines). We can clearly notice that the measured durations are close to a conversion efficiency of only 22%. Despite the fact that a coulomb counter IC was not used in our experiment to quantify precisely the amount of stored energy in the battery, the results shows nevertheless that in addition to the performance of the boost converter which is around 60 to 80%, the battery chemistry has a significant impact on the energy storage performance due principally the aging effect, the periodic peak current drawn during radio transmission and the round-trip efficiency of the battery which is never 100%.

VI. Conclusions

This work has shown the development of a self-sustained IPv6 multi-sensor device running on a small rechargeable battery and harvesting solar energy. The deactivation of the LPL mechanism and the implementation of the optimized version of the ND protocol reduce the standby current and the amount of protocol messages exchange respectively allowing a sleepy node to last longer on a small rechargeable battery. The device consumes only 5μA in deep sleep mode and has an average power consumption of approximately 100μW while performing sensors measurement and reporting at 5 minutes interval.

The management of the nodes is handled through the OMA LwM2M protocol and a web application has been developed for an automatic configuration of the nodes when these later are registering in the network. The downward communication with the sleepy node is accomplished by the Registration Update mechanism of the LwM2M protocol. The power consumption of the node can further be reduced through optimization of the sensors reporting interval, frequency and duration of the wake-up period and other parameters which are easily configurable through the web interface.

The results showed finally that a very small solar panel of 600mm^2 can sustain an average power consumption of 100μW at reasonably low ambient light (1000 lux). But for a prolonged operation at very low light intensity (<500 lux), due to the overall low efficiency of the system to store the solar energy, the results suggest that although a rechargeable battery provides larger capacity and allow a solar powered WSN device to last longer in the dark in comparison to a super-capacitor, a combination of both storage elements would be ideal and the super-capacitor would cover all the disadvantages of the rechargeable battery while operating at very low light intensity.

References

[1] *Contiki: The Open Source OS for the Internet of Things: http://www.contiki-os.org*.
[2] *Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)*, RFC6775.
[3] M. A. M. Seliem, K. M. F. Elsayed, and A. Khattab, "Performance evaluation and optimization of neighbor discovery implementation over Contiki OS," in *2015 49th Annual Conference on Information Sciences and Systems (CISS)*, 2015, pp. 119–123.
[4] *RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks*, RFC6550, 2012.
[5] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks," in *Proceedings.Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, 2002, pp. 1567–1576.
[6] C.-J. Fu, A.-H. Lee, M.-H. Jin, and C.-Y. Kao, "A latency MAC protocol for wireless sensor networks," *International Journal of Future Generation Communication and Networking*, vol. 2, no. 1, pp. 41–54, 2009.
[7] "An adaptive energy-efficient and low-latency MAC for data gathering in wireless sensor networks," in *Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International*, 2004, p. 224.
[8] *Neighbor Discovery for IP version 6 (IPv6)*, RFC4861, 2007.
[9] *Constrained Application Protocol (CoAP)*, RFC7252, 2014.
[10] *Lightweight M2M Specification v1.0*, 2014.
[11] *Leshan - OMA Lightweight M2M server and client in Java: www.eclipse.org/leshan*.
[12] *Wakaama - OMA Lightweight M2M C implementation: http://www.eclipse.org/wakaama*.
[13] *AwaLwM2M - Implementation of the OMA Lightweight M2M protocol in C: https://github.com/ConnectivityFoundry/AwaLWM2M*.
[14] Telefonica, *lwm2m-node-lib: Library for building OMA Lightweight M2M Applications*.