Scientific Research Publishing

Advances in Internet of Things, 2019, 9, 1-14
http://www.scirp.org/journal/ait
ISSN Online: 2161-6825
ISSN Print: 2161-6817

Development of Power Consumption Models for ESP8266-Enabled Low-Cost IoT Monitoring Nodes

Olubiyi O. Akintade, Thomas K. Yesufu, Lawrence O. Kehinde

Department of Electronic and Electrical Engineering, Obafemi Awolowo University, Ile-Ife, Nigeria
Email: oakintade@oauife.edu.ng, thomas_yesufu@yahoo.com, lokehinde@yahoo.com

How to cite this paper: Akintade, O.O., Yesufu, T.K. and Kehinde, L.O. (2019) Development of Power Consumption Models for ESP8266-Enabled Low-Cost IoT Monitoring Nodes. *Advances in Internet of Things*, **9**, 1-14.
https://doi.org/10.4236/ait.2019.91001

Received: January 2, 2019
Accepted: January 28, 2019
Published: January 31, 2019

Abstract

The development of energy and cost efficient IoT nodes is very important for the successful deployment of IoT solutions across various application domains. This paper presents energy models, which will enable the estimation of battery life, for both time-based and event-based low-cost IoT monitoring nodes. These nodes are based on the low-cost ESP8266 (ESP) modules which integrate both transceiver and microcontroller on a single small-size chip and only cost about $2. The active/sleep energy saving approach was used in the design of the IoT monitoring nodes because the power consumption of ESP modules is relatively high and often impacts negatively on the cost of operating the nodes. A low energy application layer protocol, that is, Message Queue Telemetry Transport (MQTT) was also employed for energy efficient wireless data transport. The finite automata theory was used to model the various states and behavior of the ESP modules used in IoT monitoring applications. The applicability of the models presented was tested in real life application scenarios and results are presented. In a temperature and humidity monitoring node, for example, the model shows a significant reduction in average current consumption from 70.89 mA to 0.58 mA for sleep durations of 0 and 30 minutes, respectively. The battery life of batteries rated in mAh can therefore be easily calculated from the current consumption figures.

Keywords

IoT, ESP8266, Power Model, Event-Based Monitoring, Time-Based Monitoring

1. Introduction

Internet of Things (IoT) nodes are smart, Internet-connected, resource con-

strained end devices embedded with at least one transducer (sensor or actuator), a microcontroller, a wireless transceiver and a power source. The transducer is used to interact with physical parameters while the transceiver is used to wirelessly communicate with other nodes and users locally or via the Internet. Commonly used low energy wireless transceivers include ZigBee, Bluetooth, Bluetooth Low Energy (BLE), WiFi, etc. The microcontrollers used in IoT node development are usually small sized and therefore have low processing and storage capacities. So also, the power source is usually a small battery. Though a number of Internet of Things (IoT) hardware platforms are available off-the-shelf [1], the recent introduction of ESP8266 modules (simply called ESP modules) into the market has enabled the development of miniature and low-cost IoT nodes. An ESP module [2] fully integrates WiFi networking capabilities and a microcontroller (Tensilica L106 32-bit) at a cost of just about $2 and an external size of 16 mm × 24 mm × 3 mm. The module can thus act as both the transceiver and the processor for IoT nodes. Numerous IoT solution providers and researchers therefore use these modules in their projects. However, the power consumption of these modules is high. An ESP8266-12E module, for example, consumes an average of 70.5 mA when it is fully on (that is, when its WiFi modem and microcontroller are both on) even when the transceiver is inactive (neither receiving nor transmitting). At this current consumption, a typical Alkaline 2500 mAh battery will only last for about 36 hours (less than two days) assuming that only the current consumption of the ESP module affects the battery capacity (other factors that affect battery capacity include ambient temperature and the current consumption of other node components). The active/sleep energy saving approach is therefore commonly used to ensure the efficient use of available energy in monitoring applications. Monitoring applications can either be time-based (where a physical parameter of interest is monitored and data is transmitted at regular time intervals) or event-based (where the physical parameter is continuously monitored but a threshold is set such that data obtained from the physical phenomenon is transmitted only if it is greater than or less than the set threshold).

Presently, ESP modules support three power saving modes, which are, modem-sleep, light-sleep and deep-sleep modes with current consumptions of 15 mA, 0.9 mA and 10 μA, respectively [2]. Consequently, the deep-sleep mode is the primary target for most low energy monitoring applications. In this mode, the WiFi modem, system clock and CPU are all switched off while only the real time clock (RTC) is left on to enable user-defined periodic wake-ups. Therefore, the deep-sleep mode is most suitable in time-based monitoring when there is a long time interval between sensor readings or where it is not necessary to continuously transmit data or monitor a physical parameter. For example, a node that monitors the temperature and humidity in a poultry farm does not need to continuously transmit its data. Such a node can sleep for about half an hour or even more. However, the maximum deep-sleep duration of ESP8266 is 4,294,967,296 μs (approximately 71 minutes). On the other hand, event-based monitoring re-

quires a continuous monitoring of the physical parameter of interest. Therefore the CPU cannot be shut down completely and the node sensor must always be active. The light-sleep mode is used in this case. In this mode, the WiFi modem and the system clock are both switched off, the CPU is pending (or suspended), while the RTC is on. The CPU can only be woken up via an external general purpose input output (GPIO) pin when an external event occurs. For example, an IoT node that monitors intrusion into a restricted area can be operated in the light-sleep mode. The ESP module in this node wakes up when an intruder is detected, notifies appropriate authorities and then goes back into the light-sleep mode but the sensor that monitors movement is always on. Also, the conventional Internet application layer protocol, HTTP, is not optimized for resource constrained IoT nodes, a lightweight application layer protocol such as Constrained Application Protocol (CoAP) or Message Queue Telemetry Transport (MQTT) [3] [4] [5] is usually used to ensure energy efficient data transfer for ESP8266 enabled IoT nodes.

Since IoT monitoring nodes are generally powered by small battery sources, there is a need for developers to be able to estimate how long these small batteries can continue to service the nodes without a need for replacement or recharge. Therefore the focus of this work is on the development of power consumption models that will enable IoT solution developers to determine approximately how long a power source (of known mAh capacity) will continue to power an ESP8266-enabled IoT time- and event-based monitoring node before it will require a replacement or recharge. The model however assumes that only the current drawn by the components of the node affects the capacity of the battery.

The remainder of this paper is organized as follows; a review of related works is presented in Section 2. Section 3 presents a general energy model for ESP8266 based IoT monitoring nodes while FSM is used in Section 4 to model the states and behavior of ESP8266-enabled time-based and event-based IoT monitoring nodes. Results obtained from the application of these models are presented in Section 5 while concluding remarks and recommendations for further studies are presented in Section 6.

2. Related Works

References [6] and [7] have shown that the power consumption of the transceiver is much higher than that of the other IoT or wireless sensor network (WSN) node components and that the transceiver consumes about the same amount of energy when it is active and when it is switched on but idle or inactive. WSN and IoT nodes are similar; the only difference between them is that IoT nodes data are intended for ubiquitous access via the Internet while local accesses will suffice for WSN nodes data. A few works have therefore been done on modeling the power consumption of IoT and WSN nodes with a lot of emphasis on the power consumption of the transceivers. While some of these works proposed generic models that can be used for all transceiver types, others focused on specific tran-

sceiver types because the functional operations and power consumption of these transceivers vary significantly [8].

Reference [9] studied the impact of the communication hardware (radios) on the total power consumption of WSN nodes. A lot of emphasis was placed on minimizing the transmit power of the radios and the number of hops within the network for energy efficiency. The work presented a power consumption model for typical wireless transceivers used in WSN nodes. Reference [10] also developed transceiver energy consumption models for WSNs. They used finite state machine (FSM) to model the behavior of IEEE802.15.4 radios and also considered the energy consumption of the various components that make up the transceivers. The work in [11] presented an energy model for Bluetooth Low Energy (BLE) transceivers and also compared the energy efficiency of BLE with that of ZigBee (IEEE802.15.4). Reference [12] presented a general model for the power consumption of IoT monitoring devices at the system level. They used selected microcontrollers (Cortex-M4 32-bit microcontroller and Texas Instruments MSP430f2618 16-bit microcontroller) and transceivers (Telecom Designs TD1202 radio module and Atmel AT86RF231 IEEE802.15.4 radio) to validate their power consumption model. However, the differences in operations and behaviors of microcontrollers and transducers from different OEMs (original equipment manufacturers) can be significant. Reference [13] also focused on LoRa transceivers in developing energy models for nodes.

As the models presented in previous works cannot be applied directly to ESP8266-enabled IoT nodes, this work focuses on the low-cost, WiFi enabled, ESP8266 based IoT monitoring nodes. A general ESP energy model is first presented, then finite automata is used to model the states and behavior of the ESP modules used in time- and event-based monitoring. The resulting FSM is then used to formulate an energy model for ESP8266-enabled IoT monitoring nodes.

3. The Energy Model

Since IoT monitoring nodes are mostly powered from small battery sources and the capacity of these battery sources are rated in mAh which is a unit of electric charge (Q) and indicates the amount of current that a battery can continue to supply for one hour, the energy model proposed in this work focuses on the current consumption of IoT node components. The current consumption of a node over time t (that is, charge) can be modeled as,

$$Q_c(t) = \int_0^t I_c(\alpha)\,\mathrm{d}\alpha \tag{1}$$

where $I_c(t)$ is the sum of the current consumption of all the components of the node.

If $I_{TD}(t)$, $I_{ESP}(t)$ and $I_{VR}(t)$ are the current consumed by the transducer, the ESP module (which comprises the transceiver and microcontroller) and the voltage regulator circuit, respectively, then,

$$I_c(t) = I_{TD}(t) + I_{ESP}(t) + I_{VR}(t) \tag{2}$$

Therefore the charge becomes,

$$Q_c(t) = \int_0^t \left(I_{TD}(\alpha) + I_{ESP}(\alpha) + I_{VR}(\alpha) \right) \mathrm{d}\alpha \tag{3}$$

since I_{TD}, I_{ESP} and I_{VR} are constants, we have,

$$Q_c(t) = \left(I_{TD} + I_{ESP} + I_{VR} \right) t \tag{4}$$

However, in order to minimize the power consumption of a node, it is necessary to enable the switching of node components between ON, OFF and SLEEP modes such that a node component is only activated when needed. If T represents the period of node operation (that is, the amount of time that elapses between two consecutive wake-ups of the ESP module) where the transducer is only switched ON once from time t_{TD1} to t_{TD2} and OFF at other times, the voltage regulator circuit is expected to be ON throughout the entire period T, while the ESP module is at SLEEP from time t_{ESP1}^{SL} to t_{ESP2}^{SL}, ON but not transmitting from time t_{ESP1}^{ON} to t_{ESP2}^{ON} and transmitting from time t_{ESP1}^{TX} to t_{ESP2}^{TX}. Also, let I_{ESP}^{SL}, I_{ESP}^{ON} and I_{ESP}^{TX} represent the current consumption of the ESP module in the SLEEP, ON (but not transmitting) and transmitting modes, respectively. Therefore from Equation (4), the current consumption of the node over the period T is

$$\begin{aligned} Q_c = {} & I_{TD}\left(t_{TD2} - t_{TD1}\right) + I_{ESP}^{SL}\left(t_{ESP2}^{SL} - t_{ESP1}^{SL}\right) + I_{ESP}^{ON}\left(t_{ESP2}^{ON} - t_{ESP1}^{ON}\right) \\ & + I_{ESP}^{TX}\left(t_{ESP2}^{TX} - t_{ESP1}^{TX}\right) + I_{VR}T \end{aligned} \tag{5}$$

where $0 \le t_{TD1}, t_{ESP1}^{SL}, t_{ESP1}^{ON}, t_{ESP1}^{TX} < T$ and $0 < t_{TD2}, t_{ESP2}^{SL}, t_{ESP2}^{ON}, t_{ESP2}^{TX} \le T$.

Also note that

$$\left(t_{ESP2}^{SL} - t_{ESP1}^{SL}\right) + \left(t_{ESP2}^{ON} - t_{ESP1}^{ON}\right) + \left(t_{ESP2}^{TX} - t_{ESP1}^{TX}\right) = T \tag{6}$$

Therefore the current consumption of an ESP8266 based monitoring node is

$$I_c = \frac{Q_c}{T} \tag{7}$$

4. FSM Model

FSM is used to model the states and behavior of the ESP8266 module. A FSM is a 5-tuple given as (*States*, *Inputs*, *Outputs*, *update*, *initial State*) [14], where *States* is a finite, non-empty set of states, *Inputs* is a finite, non-empty set of input valuations, *Outputs* is also a set of output valuations, *update* is a function that maps a state and an input valuation to a next state and an output valuation, and *initial State* is the initial state (which is an element of *States*).

4.1. Time-Based Monitoring

An ESP8266 module used in a time-based monitoring node will operate in one of the four possible states indicated in Equation (8) at a time.

$$\left.\begin{aligned} States &= \{\mathrm{ON}, \mathrm{CON}, \mathrm{TX}, \mathrm{SLP}\} \\ Inputs &= \left(\{time, sensor\} \to \{T_C, T_P, T_X, T_S, reading, \mathrm{absent}\}\right) \\ Outputs &= \left(\{node\} \to \{nodeData, \mathrm{absent}\}\right) \\ initialState &= \mathrm{ON} \end{aligned}\right\} \tag{8}$$

In the ON state, the module is switched on but not connected to an AP (access point or Internet gateway). In the CON state the module is on, connected to an AP and processing sensor data. In the TX state, the module is on, connected to an AP and transmitting node data to the AP. Lastly, in the SLP state, the module goes into the deep-sleep power saving mode. Also, as shown in Equation (8), the FSM has two inputs, that is, *time* and *sensor*. The input type, *sensor*, can either be present (with a valid sensor reading which can be a bit, several bits or an analog value) or absent while the input type, *time*, has valuations T_C, T_P, T_X and T_S (which are all constants). T_C is the time it takes the ESP module to connect with an AP after waking up from deep-sleep, T_P is the time it takes the module to receive and process sensor data into node message, T_X is the time it takes the module to wirelessly transmit node massage via the AP and T_S is the duration of sleep of the module. The FSM has only one output, *node*, which has valuations *node Data* and absent. The ON state is the initial state.

The update function is given by,

$$u(s,i)=\begin{cases}(\text{ON},\text{absent}) & \text{if } s=\text{SLP}\wedge i(time)=T_S\\ (\text{ON},\text{absent}) & \text{if } s=\text{ON}\wedge i(time)<T_C\\ (\text{CON},\text{absent}) & \text{if } s=\text{ON}\wedge i(time)=T_C\\ (\text{CON},\text{absent}) & \text{if } s=\text{CON}\wedge i(time)<T_P\\ (\text{TX},\text{absent}) & \text{if } s=\text{CON}\wedge i(sensor)=reading\wedge i(time)=T_P\\ (\text{TX},\text{absent}) & \text{if } s=\text{TX}\wedge i(time)<T_X\\ (\text{SLP},nodeData) & \text{if } s=\text{TX}\wedge i(time)=T_X\\ (\text{SLP},\text{absent}) & \text{if } s=\text{SLP}\wedge i(time)<T_S\end{cases} \tag{9}$$

where *u*, *s* and *i* represent update, current state and input, respectively, *i* (*time*) is the time the ESP8266 module spends in a particular state and *i* (*sensor*) indicates that sensor data is present.

The state transition diagram of the FSM is shown in Figure 1. The *count* variable is incremented (by 1) every second within a state and it is reset to zero when there is a transition from one state to another.

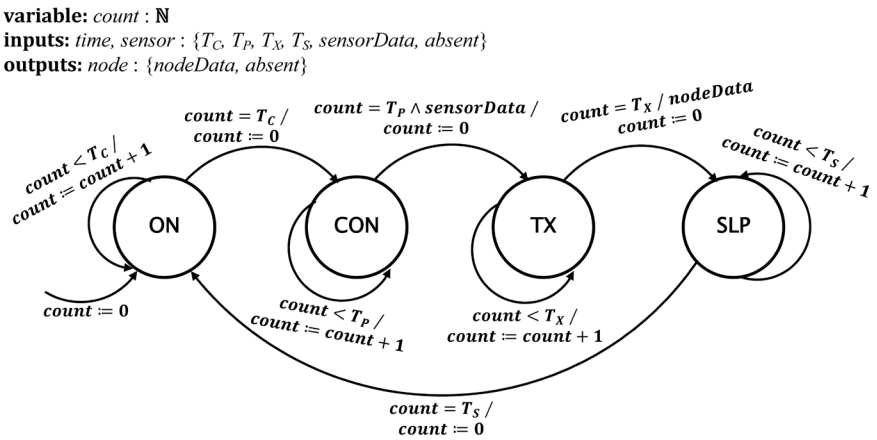

Figure 1. State transition diagram of a time-based monitoring node.

Using an application layer protocol with low overhead such as MQTT [15] [16] for data transfer, the average current consumption of the ESP8266 module when it is Fully ON and when its WiFi modem is active (that is, transmitting or receiving) is approximately the same. Also, from the state transition diagram, in order to save energy, the sensor should also be switched off when the ESP module is in the sleep mode and only be switched on during the time duration T_P (after the module has connected to an AP and ready to process sensor data). However, if the time it takes the sensor to stabilize after it is switched on is greater than, or approximately T_C (which is often the case with digital sensors), then the sensor should also be switched on as soon as the ESP module wakes up from sleep. Therefore, from Equation (5), the current consumption of a time-based monitoring node, assuming that the time taken for the sensor to stabilize is much less than T_C, over the period T is,

$$Q_c = I_{TD}T_P + I_{ESP}^{SL}T_S + I_{ESP}^{ON}\left(T_C + T_P + T_X\right) + I_{VR}T \tag{10}$$

otherwise it is

$$Q_c = I_{TD}\left(T_C + T_P\right) + I_{ESP}^{SL}T_S + I_{ESP}^{ON}\left(T_C + T_P + T_X\right) + I_{VR}T \tag{11}$$

In some cases, the sensor used (also mostly digital sensors) will have a quiescent current consumption different from its current consumption when its data is being processed and transmitted to the microcontroller. In such cases, the current consumption (quiescent current consumption) during T_C will be different from the current consumption during T_P. Therefore, Equation (11) becomes

$$Q_c = I_{TDq}T_C + I_{TD}T_P + I_{ESP}^{SL}T_S + I_{ESP}^{ON}\left(T_C + T_P + T_X\right) + I_{VR}T \tag{12}$$

where I_{TDq} is the quiescent current consumption of the sensor. Also, it is important to note that since T represents the time between two consecutive wake-ups of a monitoring node, then

$$T = T_C + T_P + T_X + T_S \tag{13}$$

4.2. Event-Based Monitoring

An ESP8266 module used in an event-based monitoring node will also operate in one of the four states defined under time-based monitoring per time. However, the SLP (sleep) state becomes the initial state in event-based monitoring as shown in Equation (14) and the sleep time, T_S is no longer one of the possible inputs because the sleep time is not predetermined but depends on an event occurring.

$$\left.\begin{aligned} States &= \{\text{ON,CON,TX,SLP}\} \\ Inputs &= \left(\{time, sensor\} \to \{T_C, T_P, T_X, reading, \text{absent}\}\right) \\ Outputs &= \left(\{node\} \to \{nodeData, \text{absent}\}\right) \\ initialState &= \text{SLP} \end{aligned}\right\} \tag{14}$$

The update function is given by Equation (15). The state transition diagram of event-based monitoring is shown in **Figure 2**. The ESP module continues to

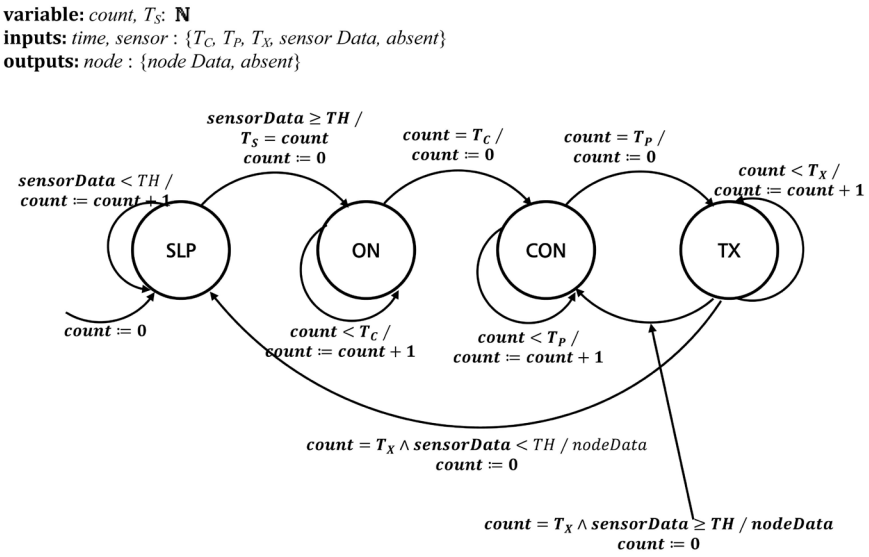

Figure 2. State transition diagram of an event-based monitoring node.

sleep until the sensor reading exceeds a set threshold. When the threshold is exceeded, the time spent sleeping is stored in variable T_S and the module transitions into the ON state. Equations (10) to (14) can also be used to calculate the current consumption of an ESP-enabled event-based monitoring node over a period T.

$$u(s,i)=\begin{cases}(\text{SLP}, nodeData) & \text{if } s=\text{TX}\wedge i(time)=T_X\wedge i(sensor)<TH\\ (\text{SLP},\text{absent}) & \text{if } s=\text{SLP}\wedge i(sensor)<TH\\ (\text{ON},\text{absent}) & \text{if } s=\text{SLP}\wedge i(sensor)\geq TH\\ (\text{ON},\text{absent}) & \text{if } s=\text{ON}\wedge i(time)<T_C\\ (\text{CON},\text{absent}) & \text{if } s=\text{ON}\wedge i(time)=T_C\\ (\text{CON},\text{absent}) & \text{if } s=\text{CON}\wedge i(time)<T_P\\ (\text{CON}, nodeData) & \text{if } s=\text{TX}\wedge i(time)=T_X\wedge i(sensor)\geq T_H\\ (\text{TX},\text{absent}) & \text{if } s=\text{CON}\wedge i(time)=T_P\\ (\text{TX},\text{absent}) & \text{if } s=\text{TX}\wedge i(time)<T_X\end{cases} \quad (15)$$

where TH is a set threshold.

5. Results

5.1. Time-Based Monitoring Case Study

A temperature and humidity monitoring node was setup using a DHT22 sensor [17], an ESP-12E module [2], a HT7333 low power voltage regulator [18] and four 1.5 V AA Alkaline replaceable batteries (rated 2500 mAh). The flowchart of the node operation is shown in Figure 3. Each time the node wakes up, it connects with an Internet gateway. If the monitoring node does not connect with the gateway within 3 s (after waking up), the node goes back into deep-sleep mode. This will ensure that energy is not unnecessarily wasted when the Internet

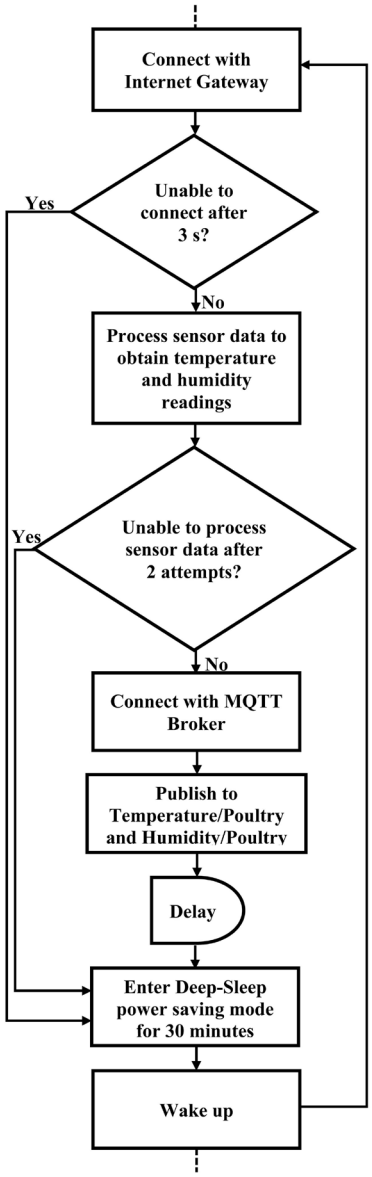

Figure 3. Flowchart of temperature and humidity monitoring node.

gateway is unavailable because it only takes the node an average time of about 1 s to connect with the gateway.

Also, if the microcontroller receives an invalid reading from the sensor (as a result of damaged sensor or circuit), it attempts to read from the sensor again. If the reading is still invalid at this second attempt, the node goes into deep-sleep mode. This will also ensure that energy is not unnecessarily wasted when the DHT22 sensor is faulty. In both cases, an alarm can be triggered before the node goes back to sleep. The "Delay" interval is used to specify the maximum amount of time that the monitoring node will continue to attempt to connect with the MQTT broker and publish its message (in cases of weak or problematic Internet services). A delay interval of 10 s was set in this case study. Also, the DHT22 sensor is powered from one of the GPIO pins of the ESP8266 module instead of

powering it from the power supply (or the output of the HT7333 voltage regulator) so as to ensure that the sensor module is also switched off when the ESP module goes into the deep-sleep power saving mode. The DHT22 sensor has a quiescent current consumption I_{TDq} and a measuring current consumption I_{TD} of 10 μA and 1.6 mA, respectively. The ESP module has a current consumption I_{ESP}^{SL} of 10 μA in the deep-sleep mode and 70.5 mA in both the ON (I_{ESP}^{ON}) and the TX (I_{ESP}^{TX}) states. The quiescent current consumption I_{VR} of the voltage regulator is about 1 μA. It takes the node an average of 1 s to connect with the Internet gateway. That is, $T_C = 1$ s. The processing time T_P is 3.5 s, the sleep duration T_S is 1800 s (30 minutes), while the transmitting time T_X is determined by the delay interval, that is, $T_X = 10$ s.

Therefore from Equation (13), T =1814.5 s, $Q_c \approx 0.29$ mAh from Equation (12) and the current consumption of the node $I_c \approx 0.58$ mA from Equation (7). So the Alkaline batteries will last for approximately 6 months before replacement will be required (assuming that only the current consumption of the node components affect the capacities of these batteries). Figure 4 shows the relationship between deep-sleep duration T_S and the current consumption I_c of the ESP8266-based temperature and humidity monitoring node. The average current consumption is 70.89 mA if the node does not go into sleep mode while it is 0.25 mA when the sleep duration is 71 minutes (the maximum deep-sleep duration of an ESP module). However, at a deep-sleep duration of just 1 minute, the current consumption has been significantly reduced to 13.81 mA from 70.89 mA. That is, the battery life will increase by a factor of 5 with a sleep duration of just 1 minute (when compared with no sleep).

5.2. Event-Based Monitoring Case Study

A motion detection node that uses a PIR motion detector [19] was set up in this case. Other node components are same as used for the temperature and humidity monitoring node. The flowchart of the node is shown in Figure 5. The ESP

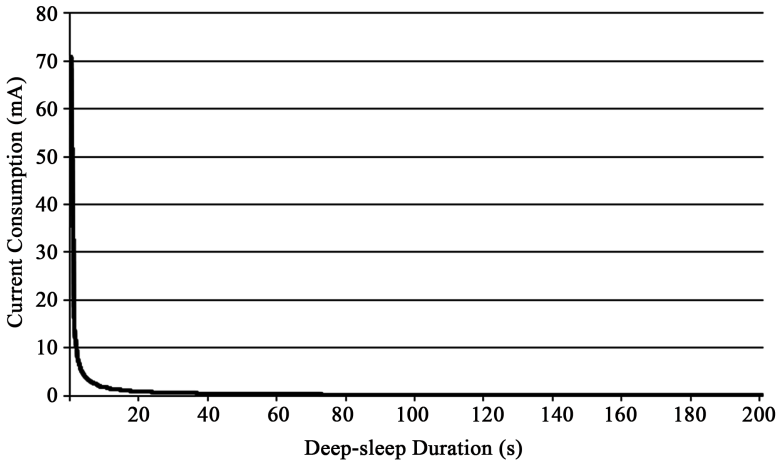

Figure 4. Graph of current consumption against deep-sleep duration of an ESP8266-enabled time-based monitoring node.

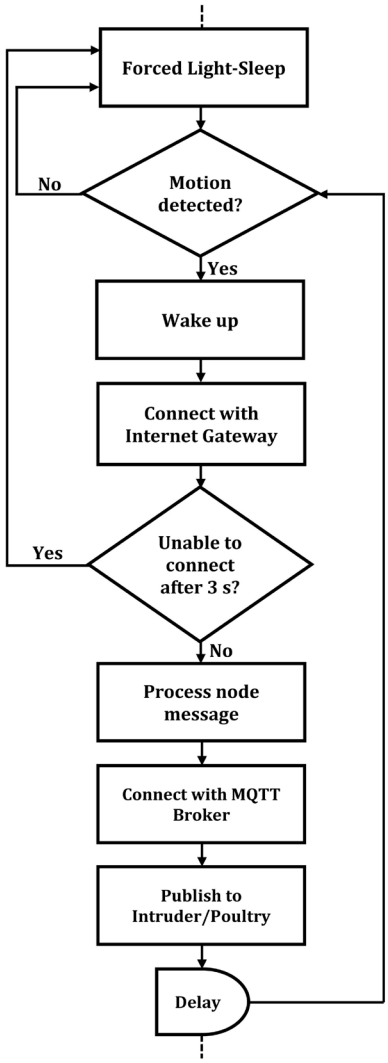

Figure 5. Flowchart of motion detection node.

module cannot go into deep-sleep mode in event-based monitoring because the sensor needs to continuously monitor the physical parameter (motion in this case) and switches on the ESP module whenever motion is detected (or a threshold is exceeded). The PIR motion detector has a quiescent current consumption I_{TDq} and a measuring current consumption I_{TD} of 0.04 mA and 0.21 mA, respectively.

However I_{ESP}^{ON} and I_{VR} remains the same at 70.5 mA and 1 μA, respectively. Also, T_C and T_X remain the same at 1 s and 10 s respectively, while T_P is now 0.5 s. Lastly, T_S depends on an event occurring, that is, motion detection in this case. Using Equations (7), (12) and (13), **Figure 6** shows the effect of varying T_S on I_c. The current consumption of the node when it is not allowed to go to sleep is 70.51 mA. However, it is 12.1 mA, 2.21 mA, 1.34 mA, 1.12 mA and 0.97 mA at forced light-sleep durations of 1 minute, 10 minutes, 30 minutes, 1 hour and 3 hours, respectively. The battery life in each case can also be estimated by simply dividing the battery capacity (in mAh) by the calculated current consumption.

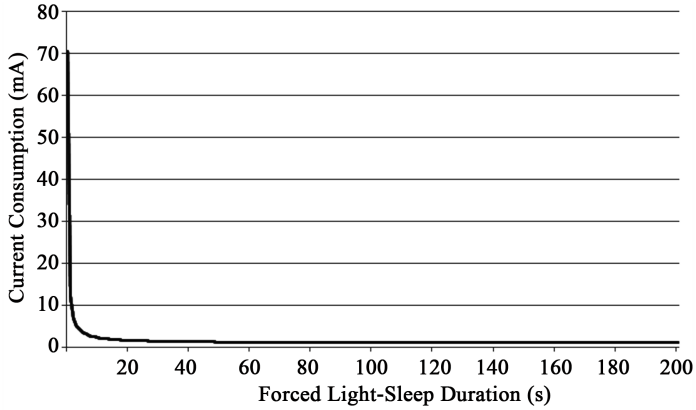

Figure 6. Graph of current consumption against forced light-sleep duration of an ESP8266-enabled event-based monitoring node.

6. Conclusions

Power models have been presented for low-cost ESP8266-enabled time-based and event-based IoT monitoring nodes. These models will enable solution developer to calculate the current consumption of an IoT node from the knowledge of the current consumption of each of the components that make up the node in order to estimate the battery life of batteries used to power these ESP8266-enabled IoT monitoring nodes. The models assume that only the current consumption of the components of a node affects the battery life of the batteries used in powering the nodes.

Results from the two case studies have shown the usability of the models presented for calculating the average current consumption of an IoT monitoring node. The battery capacity (in mAh) is then divided by the calculated average current consumption to obtain the expected life of the battery. Also, as the sleep duration of a monitoring node increases, its current consumption (and ultimately its power consumption) decreases significantly.

Various IoT application domains such as smart home, healthcare and agriculture that are enabled with ESP8266 will benefit from the models presented. The models presented can be modified and used for other IoT hardware platforms.

Acknowledgements

This work is supported by Petroleum Technology Development Fund (PTDF), Nigeria.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M. and Ayyash, M. (2015)

Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Communications Surveys and Tutorials*, **17**, 2347-2376. https://doi.org/10.1109/COMST.2015.2444095

[2] Espressif Systems (2016) ESP8266EX Datasheet. Espressif Systems Datasheet. https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf

[3] Karagiannis, V., Chatzimisios, P., Vazquez-Gallego, F. and Alonso-Zarate, J. (2015) A Survey on Application Layer Protocols for the Internet of Things. *Transaction on IoT and Cloud Computing*, **3**, 11-17.

[4] Caro, D.N., Colitti, W., Steenhaut, K., Mangino, G. and Reali, G. (2013) Comparison of Two Lightweight Protocols for Smartphone-Based Sensing. *Proceedings of IEEE SCVT* 20*th IEEE Symposium on Communications and Vehicular Technology in the Benelux*, Namur, 21 November 2013, 36-43.

[5] Thangavel, D., Ma, X., Valera, A., Tan, H.X. and Tan, C.K.Y. (2014) Performance Evaluation of MQTT and CoAP via a Common Middleware. *Proceedings of IEEE* 9*th International Conference on Intelligent Sensors, Sensor Networks and Information Processing*, Singapore, 21-24 April 2014, 1-6. https://doi.org/10.1109/ISSNIP.2014.6827678

[6] Akyildiz, I.F. and Vuran, M.C. (2010) Wireless Sensor Networks. John Wiley & Sons Inc., New York, NY, USA. https://doi.org/10.1002/9780470515181

[7] Kazeem, O.O., Akintade, O.O. and Kehinde, L.O. (2017) Comparative Study of Communication Interfaces for Sensors and Actuators in the Cloud of Internet of Things. *International Journal of Internet of Things*, **6**, 9-13.

[8] Mahmoud, M.S. and Mohamad, A.A.H. (2016) A Study of Efficient Power Consumption Wireless Communication Techniques/Modules for Internet of Things (IoT) Applications. *Advances in Internet of Things*, **6**, 19-29. https://doi.org/10.4236/ait.2016.62002

[9] Wang, Q.W.Q., Hempstead, M. and Yang, W. (2006) A Realistic Power Consumption Model for Wireless Sensor Network Devices. *Proceedings of* 3*rd Annual IEEE Communications Society on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, Reston, 25-28 September 2006, 286-295.

[10] Adinya, O.J. and Daoliang, L. (2012) Transceiver Energy Consumption Models for the Design of Low Power Wireless Sensor Networks. *Proceedings of* 2012 *IEEE Student Conference on Research and Development (SCOReD)*, Penang, 5-6 December 2012, 193-197. https://doi.org/10.1109/SCOReD.2012.6518637

[11] Siekkinen, M., Hiienkari, M., Nurminen, J.K. and Nieminen, J. (2012) How Low Energy Is Bluetooth Low Energy? Comparative Measurements with ZigBee/802.15.4. *Proceedings of* 2012 *IEEE Wireless Communications and Networking Conference Workshops* (*WCNCW*), Paris, 1 April 2012, 232-237. https://doi.org/10.1109/WCNCW.2012.6215496

[12] Martinez, B., Monton, M., Vilajosana, I. and Prades, J.D. (2015) The Power of Models: Modeling Power Consumption for IoT Devices. *IEEE Sensors Journal*, **15**, 5777-5789. https://doi.org/10.1109/JSEN.2015.2445094

[13] Aoudia, F.A., Magno, M., Gautier, M., Berder, O. and Benini, L. (2016) A Low Latency and Energy Efficient Communication Architecture for Heterogeneous Long-Short Range Communication. *Proceedings of* 19*th Euromicro Conference on Digital System Design* (*DSD*), Limassol, 31 August-2 September 2016, 200-206.

[14] Lee, E.A. and Seshia, S.A. (2011) Introduction to Embedded Systems: A Cyber-Physical Systems Approach. Berkeley, USA.

[15] Kodali, R.K. and Soratkal, S.R. (*2016*) MQTT Based Home Automation System Using ESP8266. *Proceedings of IEEE Region* 10 *Humanitarian Technology Conference* (*R*10-*HTC*), Agra, 21-23 December 2016, 1-5.

[16] Akintade, O.O., Okpako, M.O., Nasir, A.L. and Kehinde, L.O. (2017) Development of an MQTT Based Collaborative-Aware and Ubiquitous Smart Home Services over IEEE 802. 11: A Safety and Security Case Study. *Proceedings of the OAU Faculty of Technology Conference*, Ile-Ife, 24-27 September 2017, 128-134.

[17] DHT22 (2015) DHT22 (AM2302) DATASHEET. Aosong ELectronics Co, Ltd. https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf

[18] HT73XX (2006) HT73XX Datasheet, HT73XX. Low Power Consumption LDO. http://www.angeladvance.com/HT73xx.pdf

[19] HC-SR501 Datasheet (2011) HC-SR501 PIR Motion Detector Product Description, Datasheet. https://www.mpja.com/download/31227sc.pdf