International Journal of Internet of Things 2019, 8(2): 27-35
DOI: 10.5923/j.ijit.20190802.01

Development of an MQTT-based IoT Architecture for Energy-Efficient and Low-Cost Applications

Olubiyi O. Akintade[*], **Thomas K. Yesufu, Lawrence O. Kehinde**

Department of Electronic and Electrical Engineering, Obafemi Awolowo University, Ile-Ife, Nigeria

Abstract High power consumption of Internet of Things (IoT) nodes and the cost of implementing existing IoT architectures usually hamper the successful deployment of IoT solutions. Also, at locations with problematic or weak Internet presence where it takes longer time to transmit node data to the Internet, the resulting increase in power consumption hinders the development and deployment of IoT solutions. This paper therefore presents an IoT architecture that enables the development of low cost and energy efficient IoT solutions across various application domains even at the face of weak Internet signal. Existing IoT hardware platforms were first compared on the basis of on-board connectivity and microcontroller types and cost in order to select an appropriate hardware platform for the architecture. In addition, a suitable application layer protocol for end-to-end wireless communications was selected. These led to the respective selection and adaptation of ESP8266 module which integrates both WiFi and Tensilica L106 32-bit RISC processor on a single chip at US$3 per module and of Message Queue Telemetry Transport (MQTT) application layer protocol which is lightweight and data agnostic. A time based room temperature and humidity monitoring IoT application was set up using both the proposed IoT architecture and the existing MQTT architecture in order to compare power consumption. Results show that the power consumption of a monitoring node was reduced by a factor of 3.3 which ultimately resulted in a reduction in the cost of operating the nodes when the proposed architecture was used.

Keywords IoT architecture, MQTT, ESP8266, Low-cost, Energy-efficiency

1. Introduction

Internet of Things (IoT) has been identified as a disruptive technology [1,2] because of its potential to penetrate every aspect of our lives and generate new business opportunities. However, existing IoT architectures focus mainly on reliable end-to-end communications with a lot of emphasis on end-to-end interoperability of heterogeneous data formats, hardware and software components from different platforms and also the security and privacy of data and network [3-6]. In addition to reliable end-to-end communications, especially in developing economies, the issues of energy management, cost and scalability are very important for the successful deployment of IoT solutions. The cost of components (including node components, Internet gateway devices and software components) required for implementing most of the existing IoT architectures place them beyond the reach of potential users. The need to design resilient and reliable IoT nodes that can provide ubiquitous services can impact greatly on the cost of developing IoT applications. However, if the cost of developing and running an IoT solution (especially in developing economies) is not commensurate with the application or solution it is providing, the solution will most probably be ignored by potential users. It is therefore important to strike a balance between the basic requirements of an IoT application and the cost of implementing it without compromising service quality. Also, IoT nodes are resource constrained in the sense that they have limited storage and processing capabilities and most importantly, they rely on small batteries for their operations. It is therefore important that these limited resources are used efficiently. Hence, an efficient use of available energy is very important. Once the limited energy of a node is used up, the node becomes useless until its batteries are replaced or recharged. This will make the operational cost of the nodes to become unbearable if it happens too often. Lastly, scalability, which is the ability to add or remove nodes to or from an existing setup without having to first shut down the entire system, is very important. For example, in a large poultry farm (with a very small initial budget) where temperature and humidity are being monitored, a few nodes can be deployed at first and can then be scaled to a higher number of nodes in the future. Also, nodes that control these parameters (temperature and

[*] Corresponding author:
oakintade@oauife.edu.ng (Olubiyi O. Akintade)
Published online at http://journal.sapub.org/ijit

humidity) can be deployed in the future without having to shut down the system. Faulty monitoring nodes can also be taken out without affecting the operations of other nodes and the entire system.

There is therefore a need for an IoT architecture that supports low-cost, energy-efficient and scalable development of IoT solutions. A number of research works in the field of IoT have separately considered the issues of power consumption and scalability [7-10]. This work, however, presents an IoT architecture that enables the development of energy-efficient, scalable and low-cost IoT solutions across various application domains using Message Queue Telemetry Transport (MQTT) application layer protocol. The architecture will be most applicable in developing economies. Hardware and software that support open-source and low-cost development of IoT nodes were first identified and used after which an MQTT-based routing protocol was developed for end-to-end communication.

The rest of this paper is structured as follows; a review of the existing MQTT architecture is presented in Section 2. Section 3 presents an adapted MQTT architecture and the proposed IoT architecture. In Section 4, the power consumption model used for evaluating energy efficiency is presented. Section 5 presents results obtained from the comparison of the proposed architecture with the existing MQTT architecture while concluding remarks and recommendations for further studies are presented in Section 6.

2. Review

Besides using the commonly used active/sleep power saving approach for the wireless transceivers embedded into IoT nodes (because transceivers consume most of the available energy at a node [11]), it is also important to use an energy efficient data transfer protocol (application layer protocol) to ensure the efficient use of the limited energy available at a node. Hypertext Transfer Protocol (HTTP) is the conventional application layer protocol of the Internet and has been in use since 1990 [12] at a time when the need for resource constrained devices to participate on the Internet was not envisioned. The protocol defines how messages are formatted and transmitted and what actions web servers and browsers should take in response to various commands. However, the overhead involved in HTTP makes it unsuitable for resource constrained IoT nodes. Therefore, a number of lightweight application layer protocols have been developed [13,14]. Constrained Application Protocol (CoAP) and Message Queue Telemetry Transport (MQTT) are the commonly used IoT application layer protocols. Both are compared in the work presented in [15]. MQTT is used in this work because it is open-source and well suited for resource constrained devices.

MQTT [16-19] is an easy-to-implement and lightweight application layer protocol that utilizes a client/broker (or publish/subscribe) model as illustrated in Figure 1. In this model, the MQTT clients (IoT nodes) either publish or subscribe (or both) to topics in the MQTT broker. Typically, a sensor or monitoring node publishes its data to a specific topic in a particular broker via the Internet (the publisher can create the topic at the same time it is sending data) while an actuator or control node simply subscribes to an appropriate topic (or topics) in the same broker (also via the Internet). Users can also publish and subscribe to topics in the same broker using their Internet connected PCs or smartphones in order to remotely control actuators and monitor sensor data, respectively. Multi-level topics such as "floor3/room7/temperature" can also be used to provide an enhanced data description. One big advantage of MQTT becomes obvious; the publishers are completely decoupled from the subscribers. This implies that MQTT is data agnostic, as subscribers use data without any knowledge of the publishers. Therefore, MQTT supports scalability because IoT nodes can be added or removed seamlessly from the setup. Since publishers and subscribers are not directly connected, the number of publishers and subscribers in a setup can be scaled up or down at any time without having to first shutdown the entire setup.

Due to the low-cost and ease of implementation of MQTT, it has been suggested and used by a lot of researchers for the development of IoT applications. However, the need for Internet connectivity for the transmission of data, especially for sensor or monitoring nodes which are powered by small batteries, places a limitation on the use of MQTT at locations with weak or problematic Internet presence. This work therefore presents an adapted MQTT (aMQTT) architecture that further reduces the power consumption and the cost of running an IoT monitoring node.

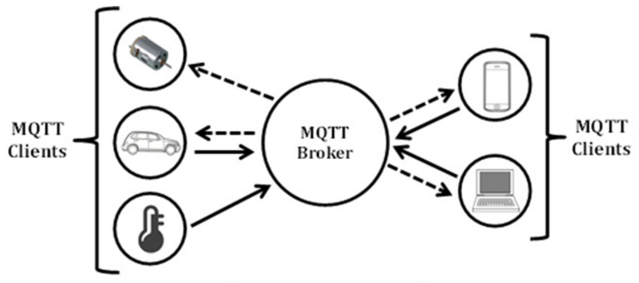


Figure 1. Existing MQTT architecture [19]

3. The Proposed Architecture

The aMQTT model and the proposed IoT architecture are shown in Figures 2 and 3, respectively. Sensor nodes used for monitoring and collecting data from various physical phenomena form the base of this architecture. These data are transmitted to the Internet for ubiquitous access via actuator nodes (or repeaters) and an Internet gateway. Primarily, this distinguishes the aMQTT IoT architecture from existing ones, as existing IoT architectures place sensor and actuator nodes in the same layer [13,20]. The sensor nodes are no longer MQTT clients but they participate in an MQTT setup by forwarding their data to available actuator nodes or repeaters which are MQTT clients.

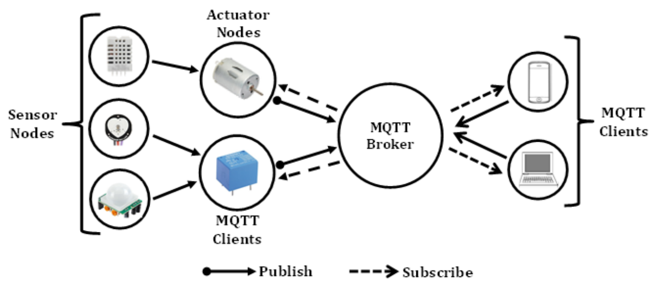


Figure 2. The proposed aMQTT architecture

Some actuator nodes are automatically triggered by data received from sensor nodes (LAN) while others are triggered by incoming user commands via the Internet. The Network Layer provides LAN and Internet services. The topmost layer (Applications) presents various services available through the interaction of sensor and actuator nodes within a network to users across various application domains including smart home, healthcare, agriculture, oil and gas, etc. The functions, requirements and implementations of each of the building blocks of this architecture are discussed next.

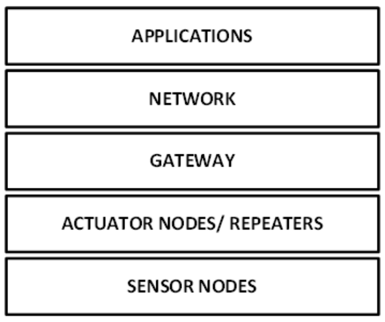


Figure 3. The proposed IoT architecture

3.1. Sensor Nodes Layer

A sensor node serves as an interface between the physical world and the cyber (digital) world. It is used for monitoring physical parameters and transmitting data obtained to the actuator node layer. The huge data associated with IoT is generated at this node. The node is embedded with at least one sensor (for monitoring a physical parameter), a microcontroller (for signal processing and other computations), a wireless transceiver (for wireless data transmission) and a power supply unit. Sensor nodes are mostly battery powered and are therefore operated in appropriate active/ sleep power saving modes.

There are basically two types of monitoring, which are time-based and event-based monitoring. In time-based monitoring, the physical parameter is monitored (and data transmitted) at regular intervals while in event-based monitoring, 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 (or less as the case may be) than the set threshold.

3.2. Actuator Nodes Layer

This layer has two main functions, which are, to serve as an interface between the digital and physical worlds (to control physical parameters) and also to serve as a repeater for the resource constrained sensor nodes. An actuator node is embedded with at least one actuator (for controlling a physical parameter), and just like a sensor node, it is also embedded with a microcontroller, a wireless transceiver and a power supply unit. The actuator node forwards the data it receives from the sensor node to the Network Layer via a gateway and vice versa. A repeater can also be used, in which case the node is not embedded with an actuator but has all the other components of an actuator node. The repeater only does packet forwarding. This layer also provides data security for the sensor nodes. Unauthorized nodes or users who do not have the access credentials of the actuator nodes or repeaters cannot access the network.

Due to the packet forwarding role of actuator nodes and repeaters (on behalf of the sensor nodes), they cannot be switched off or go into power saving sleep modes. Also, the power consumption of most actuators is so much more than that of sensors. Therefore, the actuator nodes are usually powered from mains supply or rechargeable (or renewable) energy sources.

3.3. Gateway Layer

The main function of the Gateway Layer is to serve as an Internet and LAN access point (AP) for the actuator nodes and repeaters. Therefore, it provides Internet access for actuator nodes and repeaters and also allows interactions between actuator nodes (and repeaters). In most developing economies, Internet is mostly accessed using the mobile broadband. Mobile broadband services are provided by GSM operators and users connect to it through their smartphones or USB modems. A smartphone can also create a WiFi hotspot (access point) so that once it is connected to the mobile broadband, other devices can access the Internet through it. This work therefore proposes the use of an Android-based smartphone (which often allows up to 10 WiFi enabled devices to connect to it) as the gateway. Also, the sensor and actuator nodes must be embedded with WiFi transceivers, else a pre-gateway stage that converts from other wireless protocols to WiFi will be required.

Table 1 shows the cost comparison of commonly used IoT hardware platforms. An IoT hardware platform basically incorporates a transceiver (such as ZigBee, Bluetooth, WiFi, GSM, RFID, Bluetooth Low Energy, etc.), a microcontroller (such as ATMega, Tensilica, ARM Cortex, Intel Atom, etc.) and ports for peripherals (sensors and actuators). The platforms with in-built Ethernet ports (like Arduino Yun and Raspberry Pi) are best suited for Internet gateway development as they can be connected to a router or Internet access point via a secure wired connection. In order to facilitate low-cost development, this work proposes the use of ESP8266 IoT hardware platform for developing IoT nodes because of its small size, low-cost, availability and it is open source. ESP8266 uses Tensilica Xtensa L106 (a 32-bit RISC CPU) running at 80/160 MHz, with external SPI flash memory of up to 16 MB for storing programs, it is integrated with TCP/IP protocol stack and supports all WiFi modes (that is, IEEE 802.11 b/g/n/e/i) [21].

The Gateway Layer also provides another level of security for the network. Unauthorized nodes or users who do not have the access credentials (SSID and password) of the gateway device cannot access the local network.

Table 1. Comparison of Commonly used IoT Hardware Platforms

Platform	Connectivity	Microcontroller	Cost
Arduino Yun	WiFi & Ethernet	ATmega32u4 & AtherosAR9331	$75
Raspberry Pi	WiFi, BLE & Ethernet	64 bit ARM Cortex-A53 Quad Core	$40
ESP8266	WiFi	Tensilica L106 32-bit	$3
Beaglebone Black	WiFi & BLE	OSD 3358ARM 1 GHz Cortex-A8	$70
Particle Photon	WiFi	STM32F205 120 MHz ARM Cortex M3	$20
Arduino Nano	Nil	ATmega328	$3

3.4. Network Layer

The Network Layer is an abstraction layer that is responsible for routing data within the network. Two networks are identified within this architecture and they are the Local Area Network (LAN) and the Internet. LAN enables local interactions between end devices within the setup. In other words, some actuator commands are generated internally by some sensor nodes within the same network. For example, in a poultry farm temperature monitoring and control application, the heater can be automatically turned on or off depending on data obtained from a temperature monitoring node. On the other hand, the Internet enables ubiquitous accesses, in which case anywhere and anytime access to sensor data and actuator control is enabled. Therefore sensor data is routed to an appropriate actuator within the LAN and at the same time to the Internet for ubiquitous access.

A WiFi LAN is created using a WiFi AP. All the WiFi devices connected to the same AP can interact with themselves. Therefore all the actuator nodes and repeaters that are connected to the same gateway can communicate with themselves. Sensor nodes send their data to the appropriate actuator node via their primary actuator node or repeater (the one that they are directly connected to) and the gateway. Therefore we have a maximum of 2 hops in this architecture's LAN. In order to avoid delay in data communication within the network, a zero hop is preferred (in which case the actuator node of interest is also the primary actuator node). However, two factors basically affect the selection of primary actuator node, which are, the distance between sensor and actuator nodes, and the maximum time specified for a sensor node to connect with the primary actuator node and get served. According to Friis' transmission equation (Equation 1), as the distance between the sensor and actuator nodes increases, the transmit power required also increases, thereby resulting in an unwanted increase in the power consumption of the sensor nodes.

$$\frac{P_r}{P_t} = G_t G_r \left(\frac{\lambda}{4\pi R}\right)^2 \tag{1}$$

where P_r is the power received at the receiving antenna, P_t is the transmit power of the transmitting antenna, G_r is the gain of the receiving antenna, G_t is the gain of the transmitting antenna and λ is the wavelength.

Also, though a maximum of four WiFi stations can simultaneously connect to an ESP8266 soft-AP (software enabled AP), only one can get served at a time. Therefore sensor nodes take turn to get served by the same primary actuator node. So, as the number of sensor nodes that use the same actuator node increases, the probability of delay increases and the power consumption of sensor nodes also increases. This will also impact negatively on the scalability of sensor nodes. In order to further enhance the scalability of the sensor nodes, this work uses the concept of dynamic sensor nodes as opposed to static ones. In this case, a sensor node knows the actuator nodes in its vicinity and can connect with anyone that is available in order to transmit its data. New sensor nodes can therefore be seamlessly added to the network without having to worry about which node to use as primary actuator node or repeater. Also, to ensure scalability, this architecture uses a special node identification scheme for routing data to the Internet and within the LAN. Source and destination node names are embedded within the messages that originate from the sensor nodes.

Data sent by a sensor node has three parts (Figure 4), which are; source identity (of the sensor node that is sending the packet), sensor reading and an optional destination identity (for identifying a particular actuator node within a LAN). The MQTT format for topics is used for the source identity. Though the sensor nodes are not MQTT clients, using the MQTT topic format will enable actuator nodes and repeaters (which are MQTT clients) to publish to the MQTT broker on behalf of the sensor nodes. The sensor reading of a sensor node is either an analogue or digital value obtained from the physical parameter that the sensor monitors. The third part is the optional destination node identity (or name). This is optional because some sensor nodes data are not intended for any actuator node (they are deployed for monitoring purposes only). The primary actuator node or repeater matches this name with the correct actuator node within the LAN and then commands the actuator node appropriately. The actuator node identity can also be used to address more than one actuator node or a group of nodes. For example, one may want all the switches in a kitchen to be switched off in response to high temperature and not only the switch that controls the electric cooker. Also, two symbols (that is, '=' and '@') are used to separate the three components of a sensor node message. More symbols can be introduced into this data format when dealing with heterogeneous sensor nodes (that monitor more than one physical parameter).

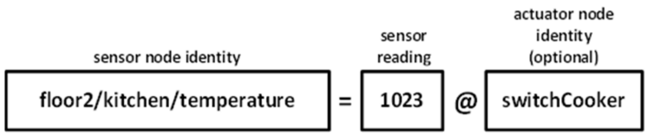


Figure 4. Sensor node data format

In this architecture, the sensor nodes are not connected in any way to the Internet (so as to ensure low power consumption and security) but the actuator nodes and repeaters are (through the gateway). Therefore, for Internet communication, the sensor nodes route their data through the actuator nodes and repeaters. Ordinarily, actuators should only receive commands from the Internet (and sensor nodes) but due to their packet forwarding role, they also transmit packets. The actuator nodes and repeaters are MQTT clients and they therefore publish (on behalf of sensor nodes) and subscribe to topics in the MQTT broker. They also publish their status to the broker. The actuator nodes and repeaters use sensor node identities as topics and they publish sensor readings to these topics appropriately. Heterogeneous sensor nodes are expected to have as many identities as the number of physical parameters that they monitor. The actuator nodes and users (with Internet connected devices such as smartphones and personal computers) subscribe to these topics for command and monitoring purposes, respectively. Sensor node command to an actuator node via the LAN therefore only serves as a backup for command via the Internet (thereby introducing redundancy). Actuator nodes that control more than one physical parameter will also subscribe to as many topics as necessary.

Internet routing using MQTT protocol guarantees the scalability of sensor nodes, actuator nodes and even users because MQTT is data agnostic. Sensor nodes do not have to know the users or actuator nodes that consume their data. They just publish their data to appropriate topics in the broker. So also, interested users or actuator nodes do not have to know the data source, they just subscribe to the appropriate topic in the broker. However, the LAN setting cannot be scaled online because a sensor node must be preconfigured to command a specific actuator node. Therefore the LAN is only used as a backup in this architecture.

3.5. Application Layer

The Application Layer is the topmost layer of this architecture where various application domains like smart home, healthcare, agriculture, transportation, oil and gas, etc. are defined. The Network Layer matches sensor and actuator nodes to appropriate applications in the Application Layer. Therefore, this layer provides IoT services for users (via their Internet connected devices). Also, this layer provides the user interface that enables users to interact with end devices (sensor and actuator nodes).

As already stated, this work uses MQTT as the application layer protocol. This is because MQTT is open-source (enabling low-cost development), is simple to implement, provides quality of service delivery, is lightweight and bandwidth efficient (can be implemented on resource constrained nodes), has little overhead and is data agnostic (publishers and subscribers are decoupled). A number of free MQTT Apps are available for Android users. For the purpose of testing, this work uses IoT MQTT Dashboard Android App on user smartphones. Other MQTT Apps (connected to the same MQTT broker) can however publish or subscribe to topics used in this work. Therefore, security codes can be used to ensure that unauthorized users do not gain access to the network. An open-source MQTT broker, test.mosquitto.org, is also used for test in this work.

4. Power Consumption

From [22], the average current consumption I_c of an ESP8266-enabled time-based monitoring node is given by

$$I_c = \frac{Q_c}{T} \tag{2}$$

where Q_c is the charge, that is, the average current consumption over period T, given by

$$Q_c = I_{TDq}T_C + I_{TD}T_P + I_{ESP}^{SL}T_S + I_{ESP}^{ON}\left(T_C + T_P + T_X\right) + I_{VR}T \tag{3}$$

where I_{TDq} is the quiescent current consumption of the sensor embedded in the sensor node, I_{TD} is the current consumption of the sensor when it is active, I_{ESP}^{SL} and I_{ESP}^{ON} represent the current consumption of the ESP module in the SLEEP and ON states, respectively, I_{VR} is the current consumption of the voltage regulator circuit, and T_C is the time it takes the ESP module to connect with an AP when it wakes up from sleep, T_P is the time it takes the module to receive and process sensor data into node message (Figure 4), T_X is the time it takes the module to wirelessly transmit node message via the AP and T_S is the duration of sleep of the module. T is the total time that elapses between two consecutive wake-ups and is given by

$$T = T_C + T_P + T_X + T_S \tag{4}$$

Therefore, the power consumption of the node is the product of the voltage across the node and I_c.

5. Testing and Results

A simple time-based room temperature and humidity monitoring IoT application was set up in two cases in order to compare the power consumption of running an IoT monitoring node in the proposed architecture and existing MQTT architecture. Case 1 used the proposed architecture to implement the application while existing MQTT architecture was used in Case 2. A temperature and humidity monitoring node (DHT node) was created using a digital humidity and temperature sensor (DHT22), an ESP8266-12E module [21] and a set of four 1.5 V AA Alkaline replaceable batteries. An actuator node that displays temperature and humidity readings on a 16X2 LCD and also sounds a buzzer when temperature or humidity readings are outside specified range was also created using an ESP8266-12E module and a rechargeable battery. The DHT node transmits temperature and humidity readings once every 30 minutes.

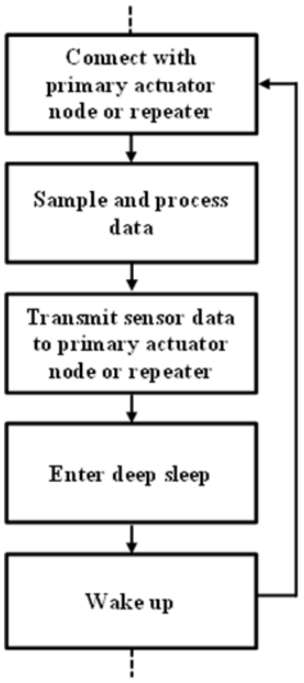

Connect with primary actuator node or repeater
Sample and process data
Transmit sensor data to primary actuator node or repeater
Enter deep sleep
Wake up

Figure 5. Flowchart of sensor node in the proposed architecture

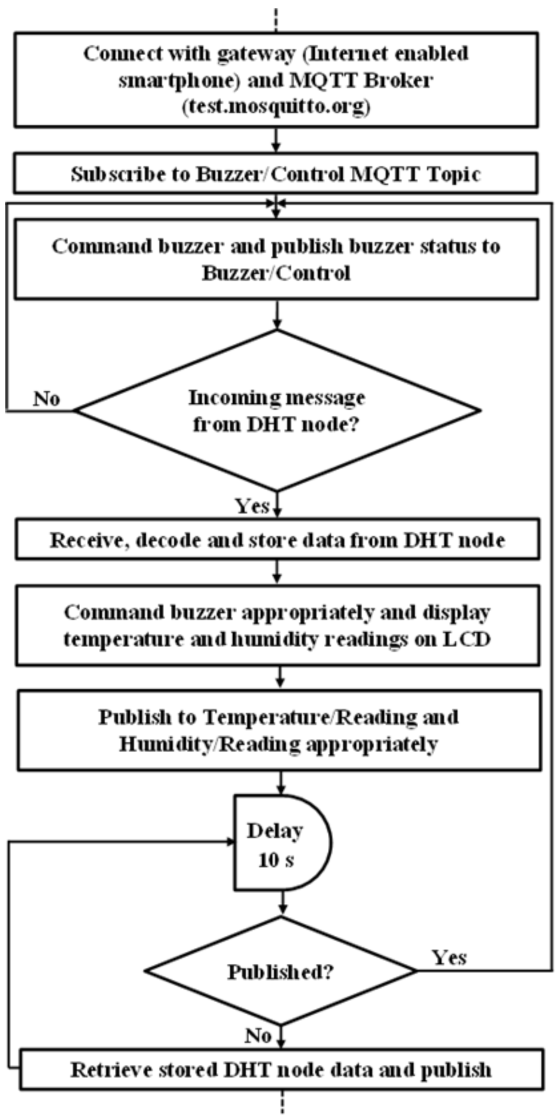

Connect with gateway (Internet enabled smartphone) and MQTT Broker (test.mosquitto.org)
Subscribe to Buzzer/Control MQTT Topic
Command buzzer and publish buzzer status to Buzzer/Control
No
Incoming message from DHT node?
Yes
Receive, decode and store data from DHT node
Command buzzer appropriately and display temperature and humidity readings on LCD
Publish to Temperature/Reading and Humidity/Reading appropriately
Delay 10 s
Published?
Yes
No
Retrieve stored DHT node data and publish

Figure 6. Flowchart of actuator node in the proposed architecture

Table 2. DHT Node Measurements in the Proposed aMQTT Architecture

Current (mA)					Time (s)			
I_{TDq}	I_{TD}	I_{ESP}^{SL}	I_{ESP}^{ON}	I_{VR}	T_C	T_P	T_X	T_S
0.01	1.6	0.01	70.5	0.001	0.53	3.48	0.02	1800

Table 3. DHT Node Measurements in the Existing MQTT Architecture

Current (mA)					Time (s)			
I_{TDq}	I_{TD}	I_{ESP}^{SL}	I_{ESP}^{ON}	I_{VR}	T_C	T_P	T_X	T_S
0.01	1.6	0.01	70.5	0.001	0.53	3.48	10	1800

5.1. Case 1

The proposed architecture was used in this scenario. The flowcharts for the DHT and actuator nodes are presented in Figures 5 and 6, respectively. The measured approximate current consumption and time taken by the DHT node to connect with the actuator node, process sensor data, transmit sensor data to actuator node and sleep are shown in Table 2.

From Equations 2 to 4, $T = 1804.03$ s, $Q_c \approx 0.09$ mAh and $I_c \approx 0.17$ mA. Therefore, since power = current × voltage, the average power consumption (at 3.3 V) of the DHT node in this case is approximately 0.56 mW.

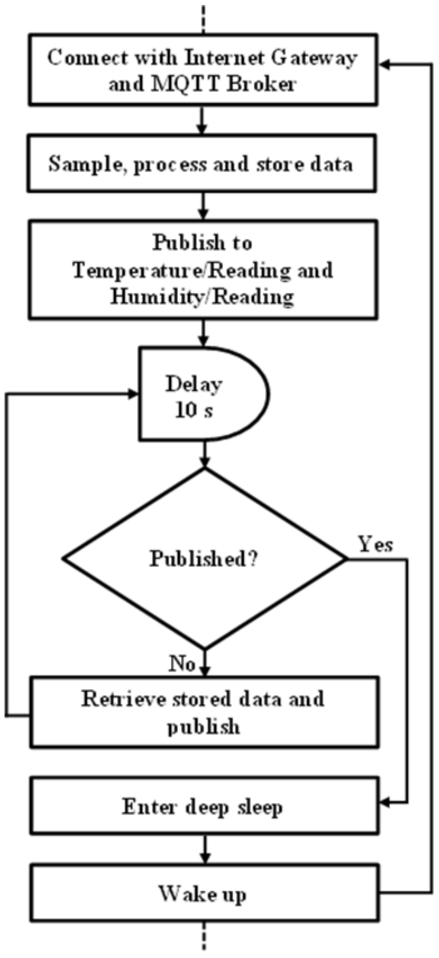
5.2. Case 2

The existing MQTT architecture was used in this scenario with flowchart of the DHT node presented in Figure 7. The measured approximate current consumptions and times taken by the DHT node are also shown in Table 3.

All other parameters remain the same except the transmit time, TX. In places with excellent Internet connectivity, it takes the DHT node only 0.02 s to publish its data to the MQTT broker (same as under the proposed architecture). On the contrary, in places with poor Internet presence, the node will go into sleep mode without publishing its data if it is not published within the delay time specified in Figure 7. If the node is unable to publish its data within this delay duration, the total energy consumed by the node in the period T (time between two consecutive wake-ups) is wasted. It takes the DHT node an average of approximately 10 s to publish its data at the location of this experiment. Therefore, a 10 seconds delay (which is now the time taken to transmit) has been introduced to enable the DHT node publish its data before entering the deep sleep mode. Also, from Equations 2 to 4, T = 1814.01 s, $Q_c \approx$ 0.28 mAh, $I_c \approx$ 0.56 mA and the average power consumption (at 3.3 V) is 1.84 mW.

The hop count in the aMQTT architecture is higher because, instead of publishing their data directly to the MQTT broker, sensor nodes will first transmit to actuator nodes which will then publish the data to the broker on behalf of the sensor nodes.

However, Comparing the transmit time in the two cases, the transmit time in the MQTT architecture has increased by a factor of 500 and has resulted in an increase in power consumption by a factor of about 3.3. Therefore the battery life in the proposed aMQTT architecture is increased and the cost of operating the node is reduced. The relationship between the transmit time and the power consumption of the DHT node is shown in Figure 8. It shows that as the transmit time increases, the power consumption also increases at a rate of approximately 0.12 mW/s.



Connect with Internet Gateway and MQTT Broker
Sample, process and store data
Publish to Temperature/Reading and Humidity/Reading
Delay 10 s
Published?
Yes
No
Retrieve stored data and publish
Enter deep sleep
Wake up

Figure 7. Flowchart of sensor node used in existing MQTT architecture

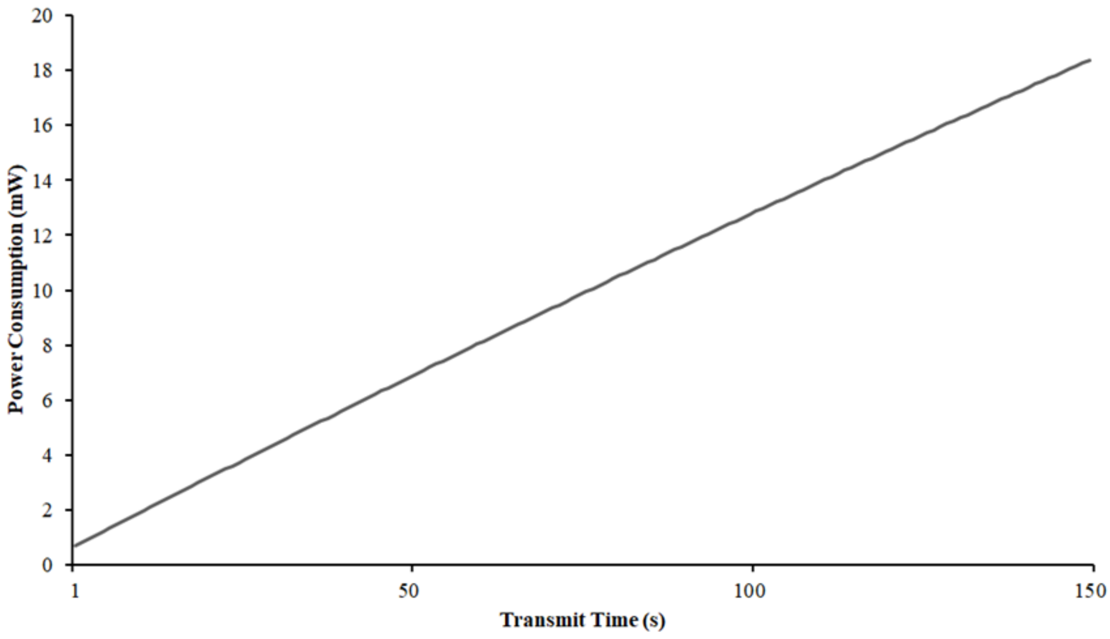


Figure 8. Power consumption of a DHT node used as MQTT client with varying transmit time

6. Conclusions

An architecture that will facilitate the development of energy efficient and low cost IoT solutions (especially in developing economies), aMQTT architecture, has been presented. The architecture is based on the existing MQTT architecture and the low cost ESP8266 IoT hardware platform.

Results show that the proposed architecture is useable and supports the deployment of IoT solutions in areas with problematic or weak Internet presence. Low energy consumption which will result in a reduced cost of operation for sensor or monitoring nodes was also recorded. The proposed architecture can be used in various application domains. Therefore, focus will now be on using the aMQTT architecture to develop and deploy IoT solutions across various application domains such as in agriculture, healthcare, smart home, etc. and also the development of a scalability model for the sensor nodes deployed in the aMQTT architecture.

ACKNOWLEDGEMENTS

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

REFERENCES

[1] K. Ebersold and R. Glass, "The impact of disruptive technology: The Internet of Things," Issues Information Systems., vol. 16, no. 4, pp. 194–201, 2015.

[2] N. D. Murthy and V. B. Kumar, "Internet of things (IoT): Is IoT a disruptive technology or a disruptive business model?," Indian Journal of Marketing., vol. 45, no. 8, pp. 18–27, 2015.

[3] J. Kempf, J. Arkko, N. Beheshti, and K. Yedavalli, "Thoughts on reliability in the Internet of Things," 2011. [Online]. Available: https://www.iab.org/wp-content/IAB-uploads/2011/03/Kempf.pdf. [Accessed: 06-May-2017].

[4] S. K. Datta, C. Bonnet, and N. Nikaein, "An IoT gateway centric architecture to provide novel M2M services," in Proceedings of 2014 IEEE World Forum on Internet of Things, WF-IoT, 2014, pp. 514–519.

[5] S. L. Keoh, S. S. Kumar, and H. Tschofenig, "Securing the internet of things: A standardization perspective," IEEE Internet of Things Journal, vol. 1, no. 3, pp. 265–275, 2014.

[6] J. Strassner and W. W. Diab, "A semantic interoperability architecture for Internet of Things data sharing and computing," in 2016 IEEE 3rd World Forum on Internet of Things (WF-IoT), 2016, pp. 609–614.

[7] C. Sarkar, S. N. a. U. Nambi, R. V Prasad, and A. Rahim, "A scalable distributed architecture towards unifying IoT applications," in Proceedings of 2014 IEEE World Forum on IoT (WF-IoT), 2014, pp. 508–513.

[8] N. Kaur and S. K. Sood, "An Energy-Efficient Architecture for the Internet of Things (IoT)," IEEE Systems Journal, vol. 11, no. 2, pp. 796–805, 2017.

[9] P. Bellavista and A. Zanni, "Towards better scalability for IoT-cloud interactions via combined exploitation of MQTT and CoAP," in Proceedings of 2016 IEEE 2nd International Forum on Research and Technologies for Society and Industry Leveraging a Better Tomorrow, RTSI 2016, 2016, pp. 358–363.

[10] O. O. Kazeem, O. O. Akintade, and L. O. Kehinde, "Comparative Study of Communication Interfaces for Sensors and Actuators in the Cloud of Internet of Things," International Journal of Internet of Things, vol. 6, no. 1, pp. 9–13, 2017.

[11] I. F. Akyildiz and M. C. Vuran, Wireless Sensor Networks. New York, NY, USA: John Wiley & Sons Inc., 2010.

[12] R. Fielding, U. C. Irvine, J. Gettys, J. C. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP / 1 . 1," 1999.

[13] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," IEEE Communications Surveys and Tutorials, vol. 17, no. 4, pp. 2347–2376, 2015.

[14] V. Karagiannis, P. Chatzimisios, F. Vazquez-Gallego, and J. Alonso-Zarate, "A Survey on Application Layer Protocols for the Internet of Things," Transaction on IoT Cloud Computing, vol. 3, no. 1, pp. 11–17, 2015.

[15] D. Thangavel, X. Ma, A. Valera, H. X. Tan, and C. K. Y. Tan, "Performance evaluation of MQTT and CoAP via a common middleware," in Proceedings of 2014 IEEE 9th International Conference on Intelligent Sensors, Sensor Networks and Information Processing, 2014, pp. 1–6.

[16] OASIS, "MQTT Version 3.1.1 Plus Errata 01," 2015. [Online]. Available: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html. [Accessed: 17-Jan-2017].

[17] Y. Upadhyay, A. Borole, and D. Dileepan, "MQTT based secured home automation system," in Proceedings of 2016 Symposium on Colossal Data Analysis and Networking, CDAN, 2016, pp. 1–4.

[18] R. K. Kodali and S. R. Soratkal, "MQTT based home automation system using ESP8266," in Proceedings of IEEE Region 10 Humanitarian Technology Conference 2016, R10-HTC 2016, 2016, pp. 1–5.

[19] O. O. Akintade, M. O. Okpako, A. L. Nasir, and L. O. Kehinde, "Development of an MQTT-based Collaborative-aware and Ubiquitous Smart Home Services over IEEE 802. 11: A Safety and Security Case Study," in Proceedings of the OAU Faculty of Technology Conference, 2017, pp. 128–134.

[20] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," Computer Networks, vol. 54, no. 15, pp. 2787–2805, 2010.

[21] ESP8266 Datasheet, "ESP8266EX Datasheet," Espressif Systems Datasheet, 2016. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf. [Accessed: 05-Jan-2018].

[22] O. O. Akintade, T. K. Yesufu, and L. O. Kehinde, "Development of Power Consumption Models for ESP8266-Enabled Low-Cost IoT Monitoring Nodes," Advances in Internet Things, vol. 9, no. 1, pp. 1–14, 2019.