Revista de Ciencias Tecnológicas (RECIT). Universidad Autónoma de Baja California ISSN 2594-1925
Volumen 5 (4): 363-375. Octubre-Diciembre. https://doi.org/10.37636/recit.v54363375.

Energy consumption of an internet of things development board

Consumo de energía de una placa de desarrollo de internet de las cosas

Gabriel Lee Álvarez-Rosado[1], **Kevin Adrián Martínez-Hernández**[1], **Mario Alberto Camarillo-Ramos**[1], **Verónica Quintero-Rosas**[1], **Arnoldo Díaz-Ramírez**[1], **Roberto López-Avitia**[2]

[1]Tecnológico Nacional de México / Instituto Tecnológico de Mexicali, Department of Computer Systems. Av., Tecnológico S/N CP 21376 colonia Elías Calles, Mexicali, Baja California, México.

[2]Department of Bioengineering, Universidad Autónoma de Baja California, Boulevard Benito Juárez S/N, Parcela 44, CP 21280, Mexicali, Baja California, México.

Corresponding author: Mario Alberto Camarillo Ramos, TECNM/ Instituto Tecnológico de Mexicali, Avenida Álvaro Obregón S/N, CP 21100 Colonia Nueva, Mexicali Baja, California, México. E-mail: mario.camarillo@itmexicali.edu.mx. ORCID: 0000-0003-0700-1885.

Recibido: 30 de Agosto del 2022 **Aceptado:** 14 de Noviembre del 2022 **Publicado:** 23 de Noviembre del 2022

Abstract. - *Internet of Things is a highly applicable technology due to its versatility in areas such as agronomy, health applications, and industry. Besides, portability makes these devices affordable. IoT development boards communicate through Wi-Fi transmitted messages via the Internet, depending on the inner workings of the IoT development board, energy consumption can vary in such transmission. Furthermore, this consumption could change if the board is powered by different power supplies and the quantity of Wi-Fi transmitted messages. This paper provides a methodology to acquire an energy profile when sending data (byte) using Message Queue Telemetry Transport (MQTT) protocol on DEVKIT V1 NodeMCU-32 (ESP32) development board. Three different power supplies were used for the board, a 3.7 LiPo Battery, 5v usb Power bank and 9V NiMh rechargeable battery. The higher current consumption obtained was using a 3.7 battery, followed by 5v and the lowest current consumption was when using 9v. However, results demonstrate that when using the 9v power supply the energy consumption is two times higher than using 3.7v. Therefore, the best voltage source for transmission and energy consumption using a NodeMCU-32 development board will be 3.7 volts.*

Keywords: IoT; Energy consumption; ESP32; Microcontroller.

Resumen. - *El Internet de las Cosas es una tecnología de gran aplicación por su versatilidad en áreas como la agronomía, las aplicaciones sanitarias y la industria. Además, la portabilidad hace que estos dispositivos sean asequibles. Las placas de desarrollo de IoT se comunican a través de mensajes transmitidos por Wi-Fi a través de Internet, según el funcionamiento interno de la placa de desarrollo de IoT, el consumo de energía puede variar en dicha transmisión. Además, este consumo podría cambiar si la placa se alimenta con diferentes fuentes de alimentación y la cantidad de mensajes transmitidos por Wi-Fi. Este documento proporciona una metodología para adquirir un perfil de energía al enviar datos (byte) mediante el protocolo de transporte de telemetría de cola de mensajes (MQTT) en la placa de desarrollo DEVKIT V1 NodeMCU-32 (ESP32). Se utilizaron tres fuentes de alimentación diferentes para la placa, una batería LiPo 3.7, un banco de energía USB de 5v y una batería recargable NiMh de 9V. El mayor consumo de corriente obtenido fue al utilizar una batería de 3.7, seguido de 5v y el menor consumo de corriente fue al utilizar 9v. Sin embargo, los resultados demuestran que cuando se usa la fuente de alimentación de 9v, el consumo de energía es dos veces mayor que cuando se usa la fuente de alimentación de 3,7v. Por lo tanto, la mejor fuente de voltaje para transmisión y consumo de energía utilizando una placa de desarrollo NodeMCU-32 será de 3,7 voltios.*

Palabras clave: IoT; Consumo de energía; ESP32; Microcontrolador.

1. Introduction

Internet of things (IOT) is an emerging technology in which data transmission and information access among any objects or devices becomes easier, thus making life easier in many aspects [1]. This technology has grown exponentially. According to a report by Cisco, the number of internet-connected devices will be more than triple that of the global population. [2] Such rise in IoT devices will lend itself to a diversity of applications such as smart energy, smart building and health monitoring equipment. IoT is nothing but communication between devices all over the globe. This essentially upsurges inter-connectivity and sharing of data between devices and people. This leads to various issues which makes the IoT framework vulnerable in terms of scalability, security, privacy, energy efficiency, etc. [3]. Although IoT technologies are in the forefront of research and development [1], battery technologies are not evolving as fast to keep up with these emerging technologies. In addition, security requirements of IoT devices strains energy consumption, making optimization a necessity. Unfortunately, most programmers and developers lack the knowledge regarding software/energy dependence, failing on energy-friendly algorithm development. Researchers are continuously looking for ways to improve the energy efficiency of software running on embedded systems. Energy efficiency has a great impact on choosing the right MCU for a particular application [4]. Power management is a topic for static and mobile systems. In static systems, a well power management will reduce generated heat and electricity bills, and for mobile systems provide longer battery life. Power consumption of an MCU depends on operating voltage and current [5]. Different techniques can be implemented to reduce the power consumption, particularly in microcontrollers [6][7] [8].

IoT devices have offered improvement for development in multiple areas, but these benefits might be limited by hardware and electronic design. There is a small list of advantages and disadvantages:

- **Advantages:**
  - High portability.
  - Low Cost.
- **Disadvantages:**
  - Limited memory
  - Limited processing
  - Limited energy

Most IoT devices are battery powered, hence these are battery restricted. Furthermore, these devices become useless or inoperable until a battery replacement is performed, which can represent a high operating cost for the IoT device. To reduce cost, it is necessary to maximize the use of energy and reduce battery replacement as much as possible.

Considering that IoT devices are wireless, there exists various ways to communicate with another device (M2M), thus sharing data becomes crucial and significant about energy consumption for IoT applications [9][10], since it is necessary a certain procedure to send data, which will take resources from the board, a wireless technology such as Bluetooth, WIFI, Zigbee, Lora, among others, must be necessary. However, when using WIFI, it is possible to standardize the process in how information is sent using the following protocols:

1. The MQTT (MQ Telemetry Transport) is a simple Internet of things communication protocol. It is based on passing messages between clients through the central server. The client can be of the publisher type, sending their messages to defined topics (address, topic). The publisher can be represented by a sensor or meter [11].
   a) CoAP (Constrained Application Protocol) is a specialized web transfer protocol for use with

ISSN: 2594-1925

constrained nodes and constrained networks in the internet of things. It is generally used for machine-to-machine communication (M2M) [12].

b) In addition, HTTP (Hypertext Transfer Protocol) is an application protocol for distributed, collaborative, hypermedia information systems that allows users to communicate data on the World Wide Web [13].

These communication protocols are some of those used for the development of IoT applications, also called application layers. A study by Eclipse in 2021 [14] states that the top three communication protocols used by developers are MQTT with 44%, HTTP with 26% and REST with 23%.

When using IoT devices, it is beneficial to know how much energy is consumed when transmitting data using a protocol. Then, we will be able to optimize the number of messages that are necessary to transmit a byte, increasing the amount of running battery time that application requires.

2.- Related work

People have been concerned with analyzing energy consumption in IoT devices, implementing a similar approach to the one shown in this work.

In [9], an analysis of the current used in a development IoT device is done, making emphasis on the analysis of current consumption, using the ESP8266 development board for its cost, this article analyzes, Device on and off power task (Wake up, sleep mode), DHT22 sensor data acquisition, as well as the energy required to establish connection to the internet and a MQTT server.

They compare MQTT and aMQTT, a small variant of the protocol, yielding a very similar current consumption in both variations. Furthermore, the execution time is less when using aMQTT variant, thus representing a lower energy consumption.

In [15], a comparison of the different quality levels of the MQTT protocol (light QoS) is done with the ESP32 device, using a multimeter. This performs an analysis of battery voltage drop with respect to time. They state that the quality level Qos1 has a lower consumption compared to other levels.

In [16] a comparison of quality levels (QoS levels) has been done with the ESP8266 microcontroller, where a digital multimeter is used to monitor energy consumption. Results show that energy consumption with a higher quality level generates a very high energy consumption compared to the lowest level.

An ESP8266 development board is used in [17], which performs the task of using two different sensors, DTH22 and TSL2561, they observe battery life with the number of messages per MQTT sent in a time of one second and a half second. They conclude that when sending a message every second, you get approximately twice the battery life that when sending messages every 0.5 seconds.

In [18], several protocols used for the IoT industry are compared. They perform a simulation between them using the software WANem [19], based on simulations they obtain the energy consumption calculated by the number of packages. They conclude that the MQTT protocol consumes more energy than the COaP protocol.

A comparison of different communication protocols evaluated on the same development board, COaP, MQTT, HTTP is done in [20] they performed 25 read cycles for each protocol with

ISSN: 2594-1925

a shunt resistance of 5.5 Ohm using 7.5 volts as their power supply. An average of the energy consumption is acquired and comparison is made, thus the protocols MQTT and COaP, are the most suitable for IoT devices.

2.- How MQTT works

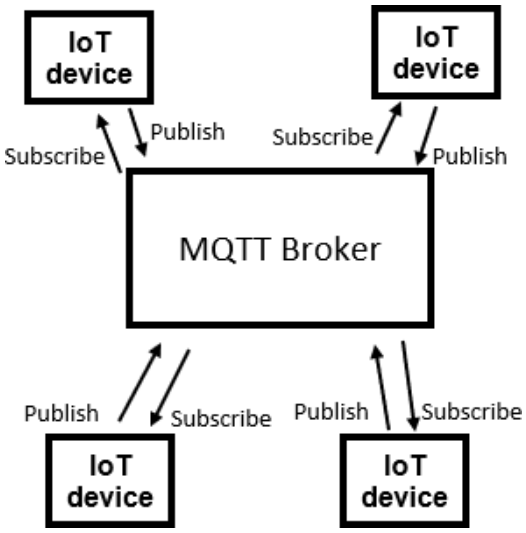

Figure 1.- MQTT Pub/Sub Architecture

MQTT is a lightweight, fast communication protocol designed for IoT. This allows communication between different devices using a Publish/Subscribe model [21][22]. To do this you need to use a broker which works as a server. The IoT device can request data and send it directly to the broker. Publications are made by topic, and to read this information you need to subscribe to the correct topic. Figure 1 shows an example of how IoT communication and brokering works.

MQTT has three levels of quality (QoS):

QoS0 (At most once):

- Message not confirmed by receiver if received

QoS1 (At least once):

- The transmitted message may arrive one or more times until the recipient returns a confirmation response.

QoS2 (Exactly once):

- Each message transmitted is received only once and is confirmed by the receiver.

The broker can be created locally, or remotely, you can observe a few brokers that can be implemented locally:

- Mosquitto [23]:
  - MQTT version 3.11 y 5.0.
  - QoS levels, QoS0, QoS1, Qos2.
- Mosca MQTT [24]:
  - MQTT version 3.1 and 3.11.
  - Qos levels, QoS0 and QoS1.
- ACTIVEMQ [25]:
  - MQTT version 3.1.
  - QoS levels, QoS0, QoS1, Qos2.

3.- Materials and methods

MQTT is analyzed as it is of the most used protocols by developers, using a broker mosquitto implemented in windows with the quality level QoS0.

Many companies produce devices for the development of applications with IoT. One such device is the arduino Yun [26], which offers wifi and ethernet connectivity for IoT applications. Microchip has several devices [27][28]. These include wifi connectivity for IoT application development.

ESPRESSIF, also develops the ESP32 [29] and ESP8266[30] boards, which are ready for IoT applications.

ISSN: 2594-1925

Table 1. Recommended IoT devices.

Devices	Connectivity	Cost
ESP32	wifi & bluetooth	18 dlls
ESP8266	wifi	16 dlls
PIC IoT	wifi	95 dlls
Arduino yun	wifi & ethernet	95 dlls
Raspberry pizero	wifi	68 dlls

This article uses a DEVKIT V1 NodeMCU-32 (ESP32) development board. This is a 32-bit MCU with Wi-Fi and Bluetooth dual mode, which makes it suitable for Internet of Things applications.

The development board is used with three power supplies at different voltages. Table 2 provides the voltages as well as the type of power supply.

Table 2. Three different power supplies used for NodeMCU-32.

Voltage	Type
9 v	NiMH rechargeable battery 450 mAh
5 v	USB Power Bank 5000 mAh
3.7 v	Li-Po Battery 500mAh

The code used to evaluate the energy consumption when sending data using the Wi-Fi module is the one provided by the manufacturer. First, basic configuration of the Wi-Fi module is set to establish network Wi-Fi communication. Afterwards, the ip address and port are defined. Then a publication is performed, therefore the ascii character "A" with the topic "test" is sent, this for MQTT (QoS0) message requirements. After sending the information, the algorithm goes into deep sleep via the WDT (WatchDog Timer) set to 1 millisecond; After the WDT finishes, it wakes up the NodeMCU-32 and the process restarts. The same algorithm is used to evaluate the energy consumption with every voltage source. Figure 2 describes the implemented Algorithm.

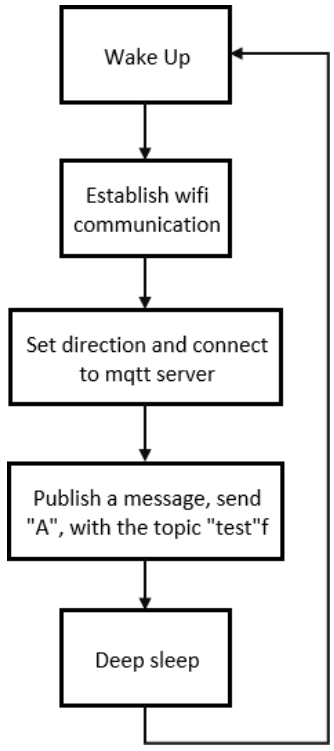

Figure 2.- Proposed algorithm to measure power consumption on MQTT

Current consumption of the Algorithm is measured using an USB Oscilloscope (Analog Discovery 2 [31]) and a Current Ranger [32].

The latter allows us to take precise current measurements, and also avoids the problem of burden voltage present in most Digital Multimeters (DMMs), whilst the former is used to measure the voltage provided by the current meter.

Figure 3 illustrates the measurement setup used for the oscilloscope. Time division is set until two periods are shown on screen (50 ms/div), resulting in a sample frequency of 5.1613KHz for Channel 1, taking 8192 samples. The voltage

ISSN: 2594-1925

range was set to 100mV/div, offset set to 0 V and the sample mode as Average. It's important to note that oscilloscope input channels are equipped with Differential inputs. The current ranger has a hardware 7kHz RC Low Pass Filter at the output and will produce a clean smooth trace on the oscilloscope, very effective for current MCU measurements.

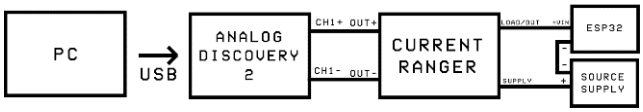

Figure 3. Measurement setup.

DIAdem [33] (National Instruments) is used to analyze the acquired signal. A waveform's period is integrated to obtain energy (Joules), and also multiplied by its respective voltage source to obtain the period's electric charge (Coulomb). A close view of the waveform signal in deep sleep mode just before restarting is shown on Figure 4, the red signal corresponds to a voltage of 3.7volts, the blue signal is when using 5 volts, and the green signal isat 9 volts.

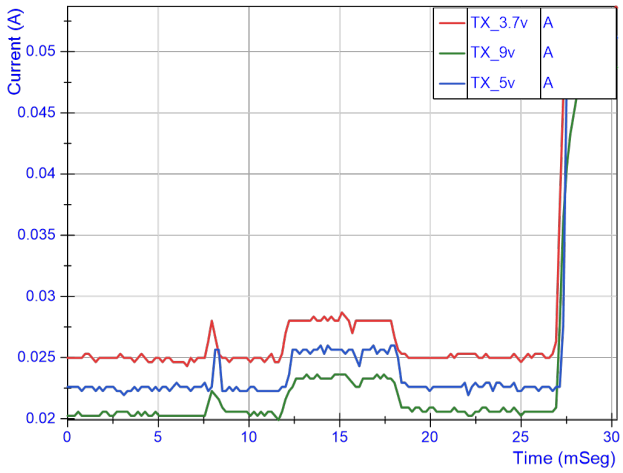

Figure 4.- Current waveforms at different supply voltages in Deep Sleep Mode.

Figure 5, shows one period of the waveform. The NodeMCU-32 runs the algorithm and starts on sleep mode, notice that before going to sleep a higher energy spike can be observed. The signal has an approximate duration of 300 milliseconds.

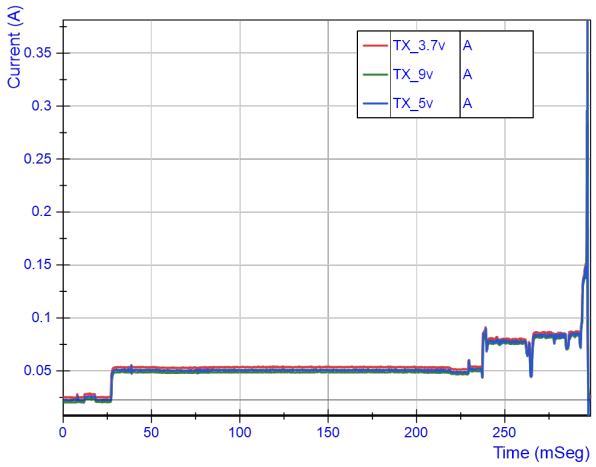

Figure 5. Period of data transmission

3. Results and discussions

Tables 3 through 5 show the mean current (in mA), mean energy (in mJ) and mean charge (in mC) obtained for every power source applied. The mean current consumption applying 5v is 4.35% less than using a 3.7v supply, something similar happened when comparing 3.7v and 9v sources, the current consumption is 7.86% less than 3.7v. This current variation may be caused by the voltage regulator. But evaluating energy (Joules), when using a 5v source yields 27.98% more energy than 3.7 battery source. Using a 9v source, the energy consumption is 119.7% higher than 3.7v. Furthermore, analysis of electric charge (Coulombs), shows that electric charge when using a 9v source will be 9.8% less than 3.7v, and 5v will be 5.8% less than 3.7v.

Table 3 shows the real current consumption, electric charge consumption and energy consumption for sending data through the internet of the NodeMCU-32 with every power supply. It is important to note that the internal linear regulator is operating when using 5v and 9v power sources. For 3.7v the regulator is not active.

ISSN: 2594-1925

Table 3. Current consumption of the NodeMCU-32

Voltage	Current		
	Mean	Min	Max
3.7v	57.03 mA	24.28 mA	254.4 mA
5v	54.55 mA	8.4 mA	381.2 mA
9v	52.55 mA	19.89 mA	295.6 mA

Table 4. Coulombs consumption of the NodeMCU-32

Voltage	Electric charge (Coulomb)		
	Mean	Min	Max
3.7v	7.41 mC	0 mC	17.03 mC
5v	6.99 mC	0 mC	16.20 mC
9v	6.69 mC	0 mC	15.66 mC

Table 5. Joule's consumption of the NodeMCU-32 using the algorithm.

Voltage	Energy (Joules)		
	Mean	Min	Max
3.7v	27.43 mJ	0 mJ	27.43 mJ
5v	34.98 mJ	0 mJ	81.31 mJ
9v	60.28 mJ	0 mJ	140.99 mJ

By finding the electrical constants, we can estimate the duration of the device in operation, we can use the following equation:

$$t = battery(mAh)/load\ (mA) \qquad (1)$$

Equation 1. Estimated time of work

We perform the analysis for each voltage source, using the same capacity of 500 mAh, we can observe in table 6 the estimated time in which the algorithm could be running.

Table 6. Estimated time of execution

Voltage	mAh	Duration
3.7v	500 mAh	8.76 hrs
5v	500 mAh	9.16 hrs
9v	500 mAh	9.51 hrs

Using the electric charge, we can give an estimate of how many messages are possible to transmit through MQTT, where we first need to find the charge in the batteries we are using, as shown in table 7. We perform the analysis assuming all power sources with a capacity of 500 mAh.

Table 7. Conversion from mAh to Coulomb

Battery	mAh	Coulomb	Messages
3.7v	500 mAh	1800 c	242.91k
5v	500 mAh	1800 c	257.51k
9v	500 mAh	1800 c	269.05k

Looking at table 7, we can determine that it is possible to send approximately 242.91k data in messages with a battery of 3.7v and 500mAh, this may or may not be enough depending on the application. Using the other power sources, it's possible to send 257.5k data messages using a battery of 5v and 500 mAh, and 269k data messages using a battery of 9v and 500mAh.

The Developer/Designer has to contemplate the amount of data to send. This information could help estimate the time of duration of the developed application.

ISSN: 2594-1925

3. Future work

For further research, we plan to analyze the NodeMCU-32 when it Wakes Up. During tests, we notice that some peripherals (input/output) have been initialized into a logical high state, which represents an energy consumption issue. If there is an alternative to optimize energy consumption without modifying the initial algorithm, it will be the first step for a proper energy efficiency analysis. Furthermore, we also plan to apply the methodology of this paper to analyze different IoT MCU Boards, to see if there is a significant difference in energy consumption.

4. Conclusion

The microcontroller NodeMCU-32 suitable for IoT applications has been tested sending data through the internet using three different commercial power supplies, 3.7v, 5v and 9v. The higher current consumption was using 3.7 battery, followed by the 5v and the lowest current consumption was when using 9v. More voltage yields less current consumption; however, this does not mean less energy consumption, results demonstrate that when using 9v the energy consumption is two times higher than using 3.7v. The best option for energy optimization without modifying the algorithm for NodeMCU-32 when sending data will be using a 3.7v Li-po Battery.

5.- Authorship acknowledgements

Gabriel Lee Álvarez Rosado: Analysis of experiments, algorithm design. *Kevin Adrián Martínez Hernández*: Documentation and database. *Mario Alberto Camarillo Ramos*: Project Management, Instrumentation. *Verónica Quintero Rosas*: Project Management, Instrumentation. *Arnoldo Díaz Ramírez*: Analysis of results, Documentation. *Roberto López Avitia*: Documentation, data analysis.

References

[1] S. Wasoontarajaroen, K. Pawasan, and V. Chamnanphrai, "Development of an IoT device for monitoring electrical energy consumption," in 2017 9th International Conference on Information Technology and Electrical Engineering (ICITEE), 2017, pp. 1-4: IEEE. https://doi.org/10.1109/ICITEED.2017.8250475

[2] D. Evans, "The Internet of Things: How the Next Evolution of the Internet Is Changing Everything," 2011.

[3] M. Bansal and B. Gandhi, "IoT Based Development Boards for Smart Healthcare Applications," in 2018 4th International Conference on Computing Communication and Automation (ICCCA), 2018, pp. 1-7. https://doi.org/10.1109/CCAA.2018.8777572

[4] M. M. Al-Kofahi, M. Y. Al-Shorman, O. M. J. C. Al-Kofahi, and E. Engineering, "Toward energy-efficient microcontrollers and Internet-of-Thing's systems," vol. 79, p. 106457, 2019. https://doi.org/10.1016/j.compeleceng.2019.10 6457 https://doi.org/10.1016/j.compeleceng.2019.106457.

[5] Tsekoura, Rebel, Glösekötter and Berekovic, "An evaluation of energy efficient microcontrollers," Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC), pp. 1-5, 2014. https://doi.org/10.1109/ReCoSoC.2014.6861368.

[6] Holberg, Arne. "Innovative Techniques for Extremely Low Power Consumption

ISSN: 2594-1925

with 8-bit Microcontrollers", Atmel, 2006. http://ww1.microchip.com/downloads/en/devicedoc/doc7903.pd.

[7] Brant, Ivey. "Low-Power Design Guide" Microchip, 2015. https://www.microchip.com/en-us/application-notes/an1416.

[8] Richey, Rodget. "Low-Power Design Using PICmicro Microcontrollers", Microchip, 2015. https://www.microchip.com/en-us/application-notes/an606

[9] W. Thongdy keophilavong, Muhammad Nur Rizal "Data Transmission in Machine-to-Machine Communication Protocols for Internet of Things Application: A Review," International Conference on Information and Communications Technology 2019. https://doi.org/10.1109/ICOIACT46704.2019.8938420

[10] O. Akintade, T. Yesufu, and L. J. I. J. I. T. Kehinde, "Development of an MQTT-based IoT Architecture for Energy-Efficient and Low-Cost Applications," vol. 2019, pp. 27-35, 2019.

[11] MQTT.org. Mq telemetry transport. 2013 Available: https://mqtt.org.

[12] C. Bormann. RFC 7252 Constraines Application protocol. 2016. Available: https://coap.technology

[13] R. F. T. Berners-Lee, H. Frystyk., Hypertext Transfer Protocol -- HTTP/1.0. 1996. https://doi.org/10.17487/rfc1945.

[14] E. fundation, "IoT & Edge developer survey report December 2021," Eclipse fundation2021

[15] E. Baranauskas, J. Toldinas, and B. Lozinskis, "Evaluation of the impact on energy consumption of MQTT protocol over TLS," 05/15 2019.

[16] J. Toldinas, B. Lozinskis, E. Baranauskas and A. Dobrovolskis, "MQTT Quality of Service versus Energy Consumption," 2019 23rd International Conference Electronics, 2019, pp. 1-4, https://doi.org/10.1109/ELECTRONICS.2019.8765692.

[17] V. Kanakaris, G. Papakostas, and D. V. Bandekas, "Power consumption analysis on an IoT network based on wemos: a case study," TELKOMNIKA (Telecommunication Computing Electronics and Control), vol. 17, p. 2505, 10/01 201S. Bandyopadhyay and A. https://doi.org/10.12928/telkomnika.v17i5.11317.

[18] Bhattacharyya, "Lightweight Internet protocols for web enablement of sensors using constrained gateway devices," 2013 International Conference on Computing, Networking and Communications (ICNC), 2013, pp. 334-340. https://doi.org/10.1109/ICCNC.2013.6504105.

[19] H. K. K. Manoj Nambiar, Debadatta Mishra, Shirish Rane, Pravin Pardeshi. (2007). WANem. Available: https://wanem.sourceforge.net.

[20] M. Pavelic, V. Bajt, and M. Kusek, "Energy efficiency of machine-to-machine protocols," in 2018 41st International Convention on Information and Communication Technology,

ISSN: 2594-1925

Electronics and Microelectronics (MIPRO), 2018, pp. 0361-0366: IEEE. https://doi.org/10.23919/MIPRO.2018.8400069.

[21] S. Kraijak and P. Tuwanut, "A survey on IoT architectures, protocols, applications, security, privacy, real-world implementation and future trends," 11th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM 2015), 2015, pp. 1-6. https://doi.org/10.1049/cp.2015.0714.

[22] MQTT Version 5.0. Edited by Andrew Banks, Ed Briggs, Ken Borgendale, and Rahul Gupta. 07 March 2019. OASIS Standard. https://docs.oasis-open.org/mqtt/mqtt/v5.0/os/ mqtt-v5.0-os.html. Latest version: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html.

[23] R. A. Light, "Mosquitto: server and client implementation of the MQTT protocol," The Journal of Open-Source Software, vol. 2, no. 13, May 2017, https://doi.org/10.21105/joss.00265.

[24] M. Collina. (2013). Mosca. Available: https://github.com/moscajs/mosca.

[25] APACHE. ACTIVEMQ. Available: https://activemq.apache.org.

[26] ARDUINO. (2022). Arduino YUN. Available: https://docs.arduino.cc/retired/boards/arduino-yun.

[27] MICROCHIP. PIC-IOT WG DEVELOPMENT BOARD. Available: https://www.microchip.com/en-us/development-tool/AC164164.

[28] MICROCHIP. AVR-IOT WG DEVELOPMENT BOARD. Available: https://www.microchip.com/en-us/development-tool/AC164160.

[29] ESPRESSIF. ESP32. Available: https://www.espressif.com/en/products/socs/esp32.

[30] ESPRESSIF. ESP8266. Available: https://www.espressif.com/en/products/socs/esp8266.

[31] DIGILENT. Analog Discovery 2: 100MS/s USB Oscilloscope, Logic Analyzer, and Variable Power Supply. Available: https://digilent.com/shop/analog-discovery-2-100ms-s-usb-oscilloscope-logic-analyzer-and-variable-power-supply/.

[32] LowPowerlab. (2018). CurrentRanger. Available: https://lowpowerlab.com/guide/currentranger/.

[33] N. Instruments. (2022). What is DIAdemsoftware?. Available: https://www.ni.com/en-us/shop/dataacquisition-andcontrol/application-software-for-data-acquisition-and-control-category/what-is-diadem.html.

ISSN: 2594-1925

ISSN: 2594-1925