TELKOMNIKA, Vol.17, No.5, October 2019, pp.2505~2511
ISSN: 1693-6930, accredited First Grade by Kemenristekdikti, Decree No: 21/E/KPT/2018
DOI: 10.12928/TELKOMNIKA.v17i5.11317

Power consumption analysis on an IoT network based on wemos: a case study

V. Kanakaris[1], G. A. Papakostas*[2], D. V. Bandekas[3]
[1,3]MEASUREMENTS-Lab, Department of Physics,
International Hellenic University, GR-65404 Agios Loukas, Kavala, Greece, Yunani
[2]HUMAIN-Lab, Computer of Science Department,
International Hellenic University, GR-65404 Agios Loukas, Kavala, Greece, Yunani
*Corresponding author, e-mail: vkanak@teiemt.gr[1], gpapak@teikav.edu.gr[2], dbandek@teikav.edu.gr[3]

Abstract

On Internet of Things network (IoT), connections established among every device that connected to the Internet. An IoT uses different communication method instead to that, the Internet uses. Internet of Things (IoT) applications, use a light weight protocol Message Queue Telemetry Transport (MQTT). In this article an Internet of Things (IoT) system is presented, an advanced solution of monitoring the temperature and luminosity at different locations in a data centre of MQTT Server, measuring power consumption in 2 types of time of capturing data (every 0.5 min and every 1 min) making temperature and luminosity data visible over internet through cloud based dashboard. The quality of service (QoS) in some application is very crucial because the data collection is essential. This research focus on QoS in terms of power consumption as well as the battery life of the system. The results show that the battery life span is proportional to the QoS and the longevity of the battery and respectively the IoT network life depends on it.

Keywords: *battery, data base, internet of things, IoT, wireless sensor network, WSN*

1. Introduction

The last five years more and more studies have been conducted on IoT in order to evaluate protocol's performance, power consumption and security isuues. The power efficiency on ESP8266 have discussed in detail in many surveys. Montori et al. conducted a research that foucused on battery duration using different types of battery, different deep sleep duration, and different authentication policies, but without using it in real time application. The results showed that the batteries provided equivalent performance while the consumption during the deep sleep mode was much instead the one that provide the data sheet [1].

Thomas et al. compares the MSP430 and ESP-03 (ESP8266 ver. 3) in terms of power consumption and the results from the tests highlight that ESP-03 when it combined with MSP430 consume less energy instead of working separately. Moreover, this survey conducted without using any specific IoT protocol [2]. Mocnej et al. also conducted a research on energy consumption at ESP8266 in terms of overhead. The survey showed that when an overhead occured the battery depleted rapidly [3].

Instead of the avove mentioned surveys, this research examines the QoS of the complete IoT in terms of power consumption. Here, the entire processing is done within Raspberry Pi 2 and as Wi-Fi repeater, a NodeMCU board via a WiFi router is used [4]. The solution that being recommended here uses the NodeMCU, which is based on ESP8266 chip, providing not only processing abilities but also supports Wi-Fi as well as two Wemos D1 mini that is also bases on ESP8266. The Wemos does the processing as well as the communication activities over Wi-Fi but with low power consumption. On Wemos [5] data retrieved from the sensors and then sent to the ESP8266 which then can be processed itself using the firmware loaded into its flash memory [6]. NodeMCU act as the Access Point of the Wireless Sensor Network (WSN), where the ESP8266 chip connects to the Wi-Fi router as an intermediate station node and can push data from Wemos to the Wi-Fi router and then to the NodeRed MQTT Broker in the internet at regular intervals [7, 8]. That means the Wemos is playing as the sensor interface unit in one hand and on the other hand as the Wireless Communication Unit besides doing all the processing. This results into faster processing and

Received September 28, 2018; March 19, 2019; Accepted April 22, 2019

transmission. Current techniques and tendencies of IoT concentrates on pushing data into a centralized cloud platform which is being administered by IT experts and are available to anyone with the proper credentials and an Internet connection. Nowadays, cloud-computing platforms offer not only the required processing power but also the storage to manage vast sensor data [8]. The total uptime of these platforms continues to incline upwards as they get conciliatory to the growing request of the IoT world. Following the IoT trends, the suggested approach also comprises Cloud implementation on top of Raspberry Pi 2 WSN based temperature and illumination monitoring system.

2. System Overview

2.1. System Components

DHT 22 [9, 10] and TSL 2561 [11, 12] sensors send the measured data to the Wemos. Wemos is loaded with the firmware program written in C that does all the interfacing with sensors, processing the sensor data and interfacing with cloud platform and finally uploading the data to the NodeRed MQTT Broker ideally once every minute [13, 14]. To write the code and upload it to the ESP8266 that Wemos and NodeMCU has, Visual Micro is used. Visual Micro is part of Visual Studio 2017 and has all the feature of Arduino IDE. Two sets of data from the two Wemos that have DHT 22 and TSL 2561 sensors accordingly are send to NodeMCU that works as intermediate station from two "nodes" (Wemos) providing extended communication between nodes and router. Then the router sends the data to the NodeRed MQTT broker, which is loaded to the Raspberry Pi 2, and is responsible to publish the data and monitor them via the Smart-Phone Application shown in Figure 1, as well as to transfer them to the MongoDB in order to save them for further analysis and process shown in Figure 2. During this process we measure the Power Consumption (Volt and mA) that Wemos need on every life cycle deep sleep, wake-up transmit/receive and then deep sleep.

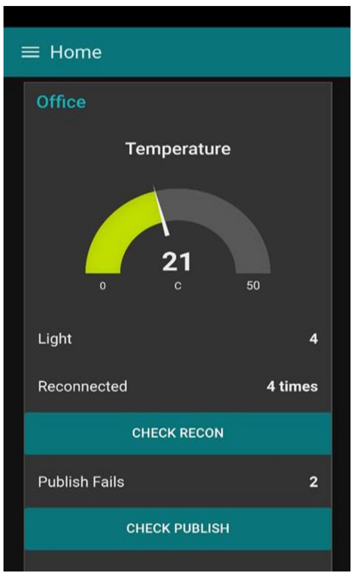

Figure 1. Smartphone application UI

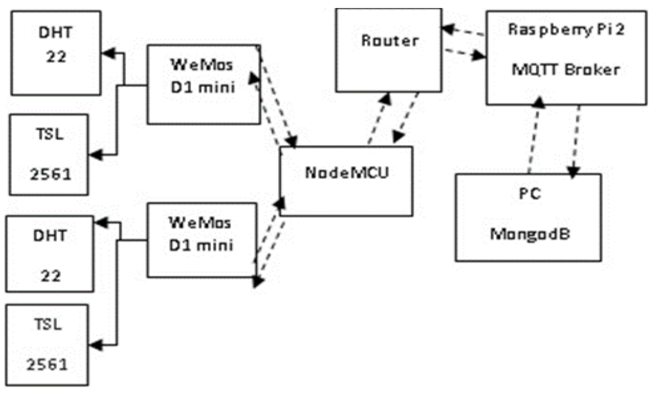

Figure 2. IoT network deployment

2.2. Hardware Implementation

The technical specification of the hardware components used in the proposed system described in the following. Measurement range of DHT 22 is from -40 degree centigrade to 80 degree centigrade with response time of 2 seconds. Its small size, low power consumption and maximum 100-meter signal transmission makes it suitable for this application. For lower temperature range and less precision DHT 11 sensors can be used.

The light sensor (TSL2561) uses infrared and visible light sensors in order to work as the human eye. Because the TSL2561 is a sophisticated sensor, it can measure both very small and very large amounts of light ESP8266 is a system-on-a-chip (SoC) designed by Espressif Systems which is based on 32-bit RISC CPU with the Tensilica Xtensa LX106 processor [15, 16]. It has features like inbuilt Wi-Fi (802.11 b/g/n), GPIO (General Purpose Input/Output), Inter-Integrated Circuit (I²C), analog-to-digital conversion, Serial Peripheral Interface (SPI), UART (Universal asynchronous receiver/transmitter), and pulse-width modulation (PWM).

Wemos D1 is a module based on ESP-8266EX microcontroller and can be used for WiFi Internet of Things (IoT) applications [17, 18]. The nine Input/Output pin makes this module qualified for big IoT target audience. It is an excellent MCU (Microcontroller Unit) that can be programmed with both Arduino IDE or Visual Micro. It has micro USB port for auto programming and it can also be programmed using OTA (Over The Air) function. At the one side of the board there is a of ESP8266 module and at the other there are a CH340 serial to USB chip, a reset button and a PCB antenna.

For this project we use NodeMCU (ESP8266) 2nd generation ver.1.0 which has integrated TCP/IP protocol stack . In this project NodeMCU is used as an intermediate station and it connects to the Access Point of the wireless network i.e. the wireless router using the SSID (Service Set Identifier) and the password of the network. As it connects to the Access Point successfully it gets assigned to an IP and joints the network. Moreover, it acts as gateway in order to provide the measurements to an MQTT server based on “NodeRed” [19, 20], a browser-based programming editor that easily anyone can connect flows together using a variety of nodes in the palette [21] shown in Figure 3. All these cooperate with the Mongo Database, which is a non-SQL database, but it is opensource and many users used it especially for IoT (Internet of Things) applications [22].

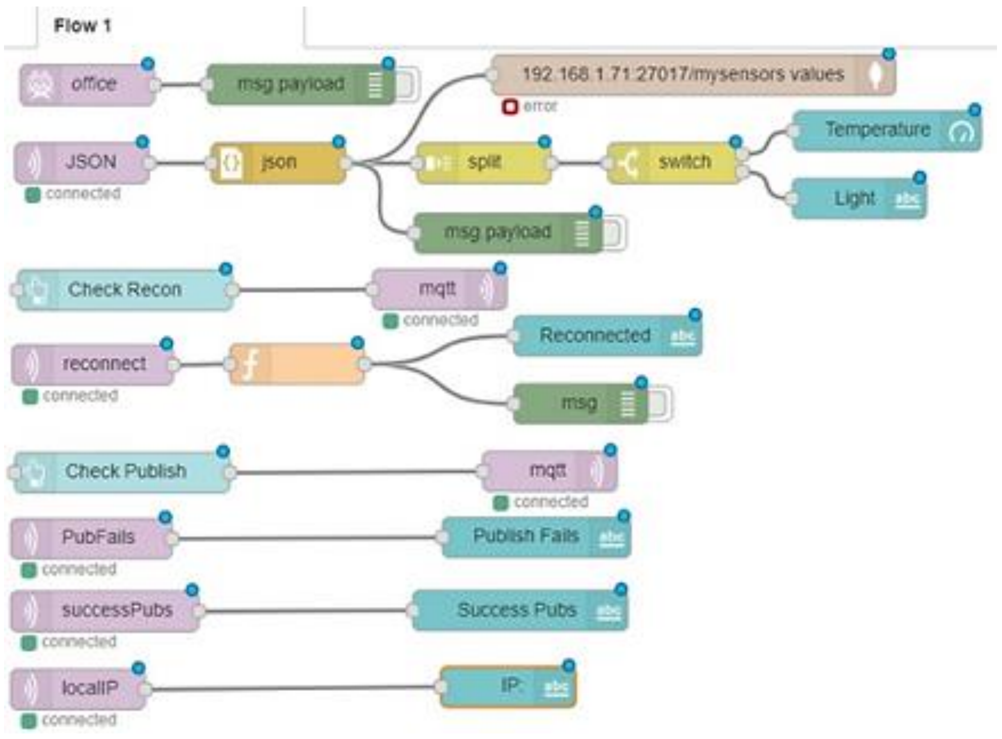

Figure 3. NodeRed deployment

MQTT server with Quality of Service (QoS) at most once, means that the minimal level is zero and ensures a best effort delivery. This technique called “fire and forget” which guarantees the same as the subjacent TCP protocol. In that case the message on receiver will not acknowledged or saved and redelivered by the sender [23, 24].

Now in order the ESP8266 to function according to the project requirements, the ESP8266 needs to be programmed. This is done by connecting the ESP8266 to Microsoft

Visual Studio 2017 (Visual Micro) running on a PC via UART interface (Send/ Receive pins) of ESP8266 through UART to USB converter shown in Figure 3, and the compiled program is uploaded to the flash memory of the ESP8266 [25, 26].

The battery that is used for all tests can provide when it is fully charged 4,03 Volts and 2273 mAh. Moreover, we are using a boost converter circuit which provides 5V supply for the sensors as well as supply to the Wemos as shown in Figure 4. We have measure that the boost converter circuit consumes in standalone mode 10 μA. The Wemos board also measured in terms of consumed current and found that it consumes in deep sleep mode, with WiFi Off, and without sensors 240 μA while when the sensors are connected consumes 270 μA.

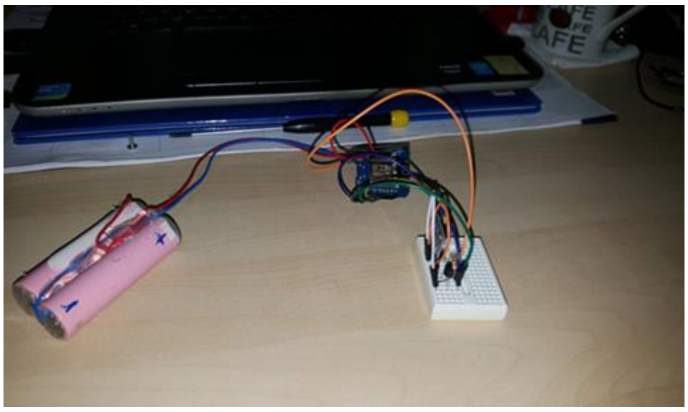

Figure 4. Prototype setup at lab with dht 22 and wemos

2.3. Software Implementation

The Visual Micro (Microsoft Visual Studio 2017)-Uses the Visual Studio environment to write the firmware in C language including the required header files for the integration of DHT sensors and the interaction with NodeRed and Mongo DB can be used to program ESP8266 as well. This Visual Micro helps to write, compile and upload the firmware to the flash memory of the ESP8266. Moreover, the required library files for the programming are extracted and saved in the folder pointed as library folder for the Visual Micro [25].

Web-Browser UI Access: The ESP8266 as well as the Wemos with sensors can be visually be checked through a web-browser from a PC to a smartphone. The measurements of each sensor depicted on the customized UI that the NodeRed provides. The interface can be easily changed from a gauge mode to labeled value. Once configured the UI provides online real-time values from the sensors [27-30].

3. Flow Steps

Flow steps as shown in Figure 5:

- Sense of temperature and illumination at a specific location of a rack with the help of two sensors.
- The two Wemos receive the data from the sensors through its GPIOs (General Purpose Input Output)
- Data are transferred to the ESP8266 (Figure 6) which has the role of intermediate station of buffering data and connect more nodes.
- ESP8266 connects to the Wi-Fi router using SSID and password.
- ESP8266 also establishes connection to the MQTT server which ran under NodeRed platform in Raspberry Pi 2.
- Raspberry Pi 2 sends data to a PC that runs MongoDB and stores the values for further processing.

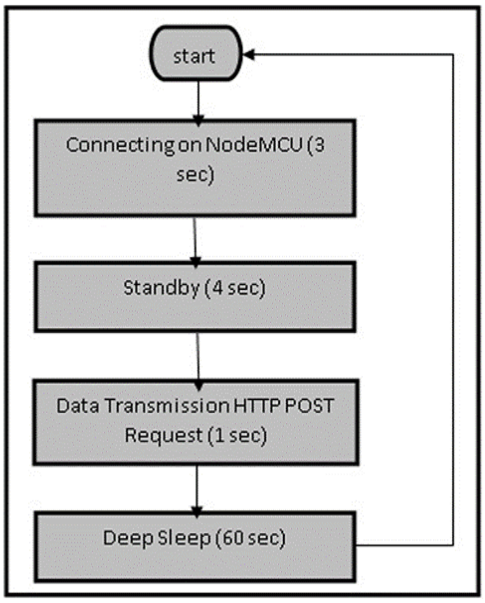
start
Connecting on NodeMCU (3 sec)
Standby (4 sec)
Data Transmission HTTP POST Request (1 sec)
Deep Sleep (60 sec)

Figure 5. Flow chart of code running on wemos

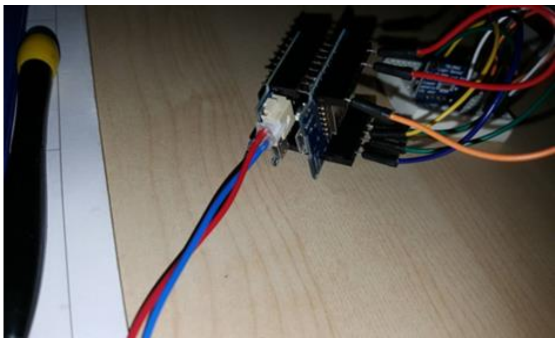

Figure 6. Prototype setup at lab of ESP8266

4. Results

As the prototype system is deployed at a setup, the live data is being monitored over Smart-Phone Application as well as the Mongo Database. The Application continuously presents the live data. In the following two graphs presented the power consumption in terms of time as the number of dropped packets during the 10-day communication. In Figure 7 the graph presents how the battery power is depleted when the Wemos send every half minute and every minute respectively. By selecting, the MQTT QoS at most once the battery lifetime increased while avoiding the re-transmissions.

The graphs in Figure 8 and Figure 9 show the communication quality on the IoT network. The results show that only 7 packets lost on totally 14400 packets shown in Figure 8, when the transmission occured every half minute, while when the trasmission occurred every minute the number of packets that lost are 4, although the MQTT QoS service doesn't provide at all quality by not re-transmitting any lost packet.

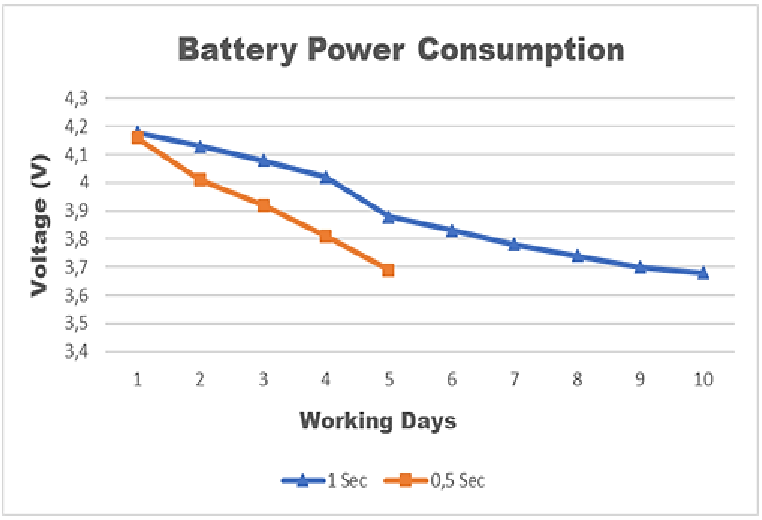

Figure 7. Battery power consumption on wemos

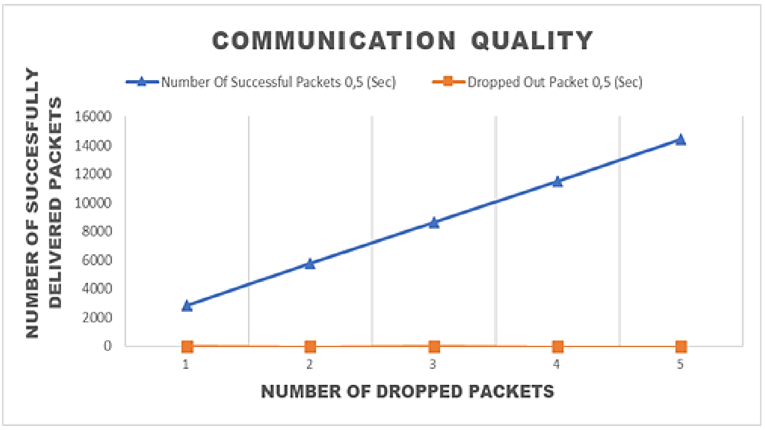

Figure 8. Communication quality when the wemos transfer data every 0.5 min

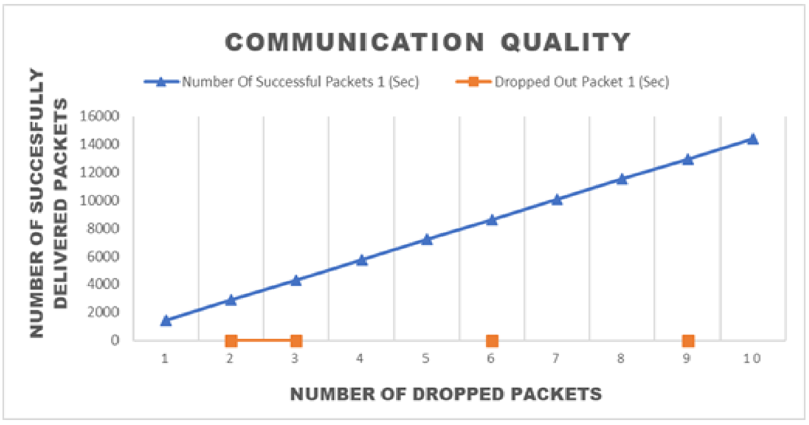

Figure 9. Communication quality when the wemos transfer data every 1 min

5. Conclusion

An IoT has been deployed using two wireless nodes using Wemos board each one equipped with a temperature and illumination sensor, and a NodeMCU which acts as intermediate stage in order to transfer data from Wemos to the Router and from there to the MQTT Broker and the MongoDB. A continuous test run of more than 21 iterations of almost 10 days and 5 days respectively of each iteration, showing almost the same performance in terms of power consumption. The hardware setup together with the software from the NodeRed platform, MongoDB and Smart-Phone App provides real time monitoring of the parameters remotely, anytime with the help of the dashboard. The outcome of this research reveals that the battery life is proportional to the frequency that the data dispatched. Using the MQTT server with Quality of Service (QoS) at most once, increases the life of the network by sycrificing the communication quality as there is no re-transmission of the data in case of packet loss. In the future, further analysis need to be done using different MQTT QoS.

References

[1] F Montori, R Contigiani, L Bedogni. *Is WiFi suitable for energy efficient IoT deployments? A performance study.* 2017 IEEE 3rd International Forum on Research and Technologies for Society and Industry (RTSI). 2017: 1-5.
[2] D Thomas, R Mc Pherson, G Paul, J Irvine. Optimizing Power Consumption of Wi-Fi for IoT Devices: An MSP430 processor and an ESP-03 chip provide a power-efficient solution. *IEEE Consumer Electronics Magazine*. 2016; 5(4): 92-100.

[3] J Mocnej, M Miškuf, P Papcun, I Zolotová. Impact of edge computing paradigm on energy consumption in IoT. *IFAC-PapersOnLine*. 2018; 51(6): 162-167.

[4] S Saha, A. Majumdar. *Data centre temperature monitoring with ESP8266 based Wireless Sensor Network and cloud-based dashboard with real time alert system*. Devices for Integrated Circuit (DevIC), IEEE. Industrial Internet of Things and Communications at the Edge by Tony Paine. 2017. 307-310.

[5] S Rukhmode, G Vyavhare, S Banot, A Narad, RM Tugnayat. IoT based agriculture monitoring system using wemos. 2017: 14-19.

[6] IM Aho, JC Agunwamba. Use of Water Extract of Moringa Oleifera Seeds (WEMOS) in Raw Water Treatment in Makurdi, Nigeria. *Global Journal of Engineering Research.* 2014; 13(1): 41-45.

[7] G Marques, R Pitarma. An indoor monitoring system for ambient assisted living based on internet of things architecture. *International journal of environmental research and public health*. 2016; 13(11): 1152.

[8] V Fernoaga, GA Stelea, D Robu, F Sandu. *Communication Solutions for Power Measurement in the Cloud*. 2018 International Conference on Communications (COMM). 2018: 397-402.

[9] M Bogdan. How to use the DHT22 sensor for measuring temperature and humidity with the arduino board. *ACTA Universitatis Cibiniensis*. 2016; 68(1): 22-25.

[10] Digital-output relative humidity & temperature sensor/module DHT22 (DHT22 also named as AM2302) https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf

[11] SU Zagade, RS Kawitkar. Wireless Sensor Network for Greenhouse. *International Journal of Science and Technology*. 2012; 2(3): 130133.

[12] J Kalathas, DV Bandekas, A Kosmidis, V Kanakaris. Seedbed based on IoT: A Case Study. *Journal of Engineering Science & Technology Review*. 2016; 9(2).

[13] RK Kodali, KS Mahesh. *A low-cost implementation of MQTT using ESP8266.* Contemporary Computing and Informatics (IC3I), 2016 2nd International Conference on. 2016: 404-408.

[14] N Shofa, A Rakhmatsyah, SA Karimah. *Infusion monitoring using WiFi (802.11) through MQTT protocol.* 2017 5th International Conference on Information and Communication Technology (ICoIC7). 2017: 1-7.

[15] ESP8266 Datasheet. Accessed July 10, 2018. https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf

[16] ESP8266 Community Forum. Accessed July 10, 2018, https://www.esp8266.com/

[17] RK Kodali, A Sahu. *An IoT based weather information prototype using WeMos*. Contemporary Computing and Informatics (IC3I). 2016 2nd International Conference on. 2016: 612-616.

[18] A McEwen, H Cassimally. Designing the internet of things. John Wiley & Sons. 2013. Designing the Internet of Things by Adrian McEwen and H. Cassimally

[19] N Shofa, A. Rakhmatsyah, SA Karimah. *Infusion monitoring using WiFi (802.11) through MQTT protocol*. 2017 5th International Conference on Information and Communication Technology (ICoIC7). 2017: 1-7.

[20] JE Luzuriaga, JC Cano, C Calafate, P Manzoni, M Perez, P Boronat. *Handling mobility in IoT applications using the MQTT protocol*. 2015 Internet Technologies and Applications (ITA). 2015: 245-250.

[21] Node-RED. Accessed June 5, 2018, https://nodered.org/docs/user-guide/

[22] mongoDB. Accessed June 9, 2018, https://www.mongodb.com/mongodb-architecture

[23] A Del Campo, E Gambi, L Montanini, D Perla, L Raffaeli, S Spinsante. *MQTT in AAL systems for home monitoring of people with dementia*. 2016 IEEE 27th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC). 2016: 1-6.

[24] MQTT QoS. Accessed June 8, 2018, https://www.hivemq.com/blog/mqtt-essentials-part-6-mqtt-quality-of-service-levels/

[25] Visual Micro Plugin Features. Accessed May 19, 2018, https://www.visualmicro.com/page/Arduino-Plugin-Distinguishing-Features.aspx

[26] H Zhou. The internet of things in the cloud: A middleware perspective. CRC press. 2012.

[27] TA Abdulrahman, OH Isiwekpeni, NT Surajudeen-Bakinde, AO Otuoze. Design, specification and implementation of a distributed home automation system. *Procedia Computer Science*. 2016; 94: 4 73-478.

[28] A Škraba, A Koložvari, D Kofjač, R Stojanović, V Stanovov, E Semenkin. *Prototype of group heart rate monitoring with NODEMCU ESP8266*. 2017 6th Mediterranean Conference on Embedded Computing (MECO). 2017: 1-4.

[29] E Dalipi, F Van den Abeele, I Ishaq, I Moerman, J Hoebeke. *EC-IoT: An easy configuration framework for constrained IoT devices*. 2016 IEEE 3rd World Forum on Internet of Things (WF-IoT). 2016: 159-164.

[30] A Kulkarni, D Mukhopadhyay. Internet of Things Based Weather Forecast Monitoring System. *Indonesian Journal of Electrical Engineering and Computer Science*. 2018; 9(3): 555-557.