Weather Tracking System using MQTT and SQLite

Ravi Kishore Kodali and Venkata Sundeep Kumar Gorantla
Department of Electronics and Communication Engineering
National Institute of Technology
Warangal, India -506004
Email:ravikkodali@gmail.com

***Abstract*—Every person is enthusiastic in accessing the information about certain place or object remotely and to make the things around them smart using new emerging technologies. This paper mainly focus an implementation of a framework of publishing data from ESP8266 to SQLite data server hosted on Raspberry pi module using MQTT Protocol. Here the sensor is assembled to ESP8266 microcontroller to read the values. A python application is developed to sense the values from sensor and to publish the data to raspberry pi. Now, ESP8266 acts as MQTT client and sensor data is sent to the MQTT broker running on Raspberry pi module. Additionally SQLite data server is also hosted on the Raspberry pi module and the data from the ESP8266 is stored in JSON format. This data from server is formatted using HTML file and the formatted data can be accessed remotely.**

***Index Terms*—: MQTT, SQLite, NodeMCU, DHT22, Mosquitto, Raspberry Pi**

I. Introduction

THE Internet of Things (IoT) includes connecting things to Internet with the goal that the data can be accessed from anyplace without any constraints. The inspiration of this implementation is to design an architecture that makes weather related information accessible over Internet. The proposed architecture helps in addressing issues of overheat that happens in households, or at a server or data-center. With a goal of preventing premature failure of the electronic devices the heat produced by the them should be dissipated properly. Even though electronic gadgets have heat dissipating components, once in a while they are insufficient to adequately settle the issue.

Almost every electronic devices incorporate a cooling network that sends out the hot air and letting in the cool air. Still, when device is in a domain with lacking wind flow, the encompassing air will get more hot and additionally builds the temperature of the surroundings, making that device overheat. There are sensors on the server cluster of smart appliances which will recognize if the gadget is getting overheated and will automatically close down to avoid harm inside segments. However, the above sensors will just observe the temperature of the gadget and they don't bother about the surroundings temperature (For example server storage room), where the issue may be caused. The weather tracking network we created can be utilized to give temperature related data that will help clients to analyze the issue, and rectify the situation by preventing the gadget taking uncommon measures, (for example, turning itself off).

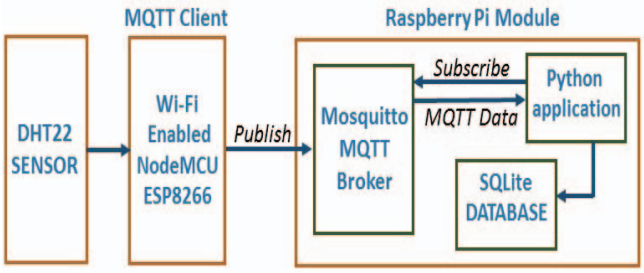

Fig. 1: Block Diagram of the Implemented system

The temperature and humidity sensor is assembled to NodeMCU ESP8266 Microcontroller which acts as MQTT Client in the implementation. The Raspberry Pi 3 module here serves as MQTT Broker. The Mosquitto broker is installed on the Raspberry pi 3 module. The MQTT client (NodeMCU ESP8266 Microcontroller) reads the temperature and humidity values from the sensor and publishes those values on the mosquitto MQTT broker (Raspberry Pi 3 module). The SQLite database is open source library is installed on Raspberry Pi module and initially the tables are created and values are initiated. Now a python script is run on the Raspberry Pi which subscribes the temperature and humidity values from the MQTT Broker and stores in the SQLite Database file on raspberry pi in JSON format. Now the temperature and humidity values stored in the SQLite data file is formatted in HTML application and values can be accessed from localhost. To access data from anywhere in the world the raspberry pi 3 is connected to weaved services with particular login credentials. The block diagram of the implemented framework is shown in Fig.1.

This paper presents a framework of publishing data from client to server running on raspberry pi module using MQTT protocol where NodeMCU ESP8266 microcontroller acts as MQTT Client and Raspberry Pi on which server is running acts as MQTT Broker as well. The rest of paper is sorted in following manner. In Section II we discuss about the hardware aspects and Section III explains about various software tools required and Section IV deals with the high level design and implementation of framework. Section V shows the experimental results obtained.

978-1-5386-1144-9$31.00

II. Hardware aspects

A. Raspberry Pi Module

The Raspberry Pi 3 module B, having size of credit card is third generation Raspberry Pi series and has lot many applications because of its key benefits such as low cost computational device, Consistent board format and powerful processing capacity. Moreover, it has both Wi-Fi and Bluetooth connectivity which makes it perfect solution for robust connected designs. The Raspberry Pi 3 Model B module is shown in Fig 2 with labels. The important technical specifications are:

- Broadcom BCM2837 Quad Core 1.2GHz, 64bit CPU
- RAM with a capacity of 1GB
- BCM43438 wireless LAN
- Bluetooth Low Energy connectivity
- Contains 40 GPIO pins
- 4 USB 2.0 ports
- HDMI Output Port

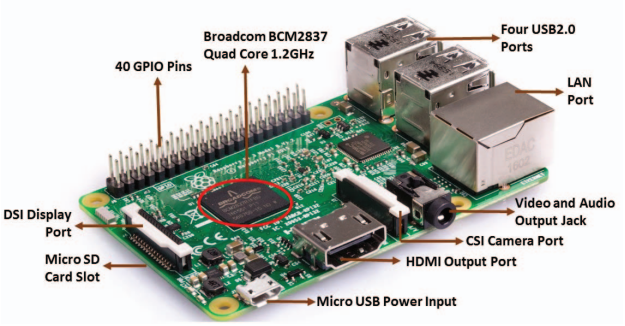

Fig. 2: Raspberry Pi 3 Mobel B Module

Moreover it has SD card slot, in which we can install SD card imaged with required operating system depending on our feasibility. Here in our project, we installed Raspbian Jessie OS on the Raspberry Pi 3 using SD card. Raspbian OS include VNC (Virtual Network Computing) technology which enables us to control Raspberry Pi remotely from another computer.

B. NodeMCU Microcontroller

NodeMCU contains firmware which is compatible and runs on ESP8266 Wi-Fi System on Chip(SoC), and the hardware depends on ESP 12E Module. ESP 8266 has integrated WiFi module, low cost and has ultra low power technology. It is coordinated with 32 bit TenSilica L 106 Microcontroller which is responsible for the extra low power consumption feature. The brief specifications of NodeMCU ESP8266 Microcontroller are provided Table 1.

TABLE I:
NodeMCU ESP8266 Specifications

Specifications	Values
MCU	32 bitTenSilica L 106
RAM	36Kb
Clock Speed	80MHz/160MHz
Opearating Coltage	3.0V 3.6V
Operating Current	80mA(Average)
Available GPIO Pins	10

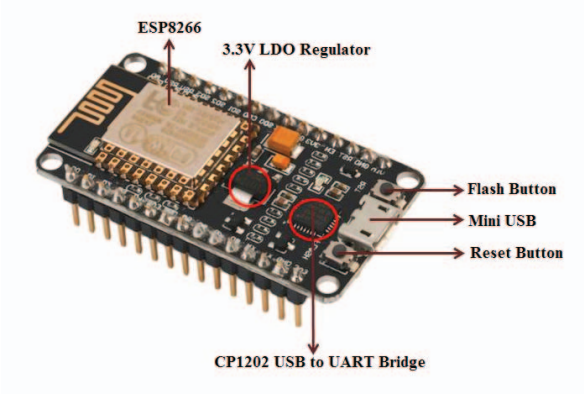

Fig. 3: NodeMCU Microcontroller

It has self contained wifi network where it can host an application if there is no application processor present in the device. The WiFi module of ESP 8266 uses 802.11 b/g/n protocol and also featured with WiFi direct (P2P), which enables different devices manufactured by different companies to connect and communicate with each other without the requirement of Wireless access point. The integration of NodeMCU ESP8266 Microcontroller is shown in Fig 3.

Moreover the ESP 8266 has lot many applications in Internet of Things implementations, Home automation systems because of its easy compatibility to the application specific devices and sensors.

C. DHT22 Sensor

DHT22 sensor provides calibrated digital single output of both temperature and humidity levels of the surroundings. This sensor consists of capacitive sensing wet devices, highly precise temperature sensing components assembled to a highly accurate 8-bit microcontroller. Features such as small size, low cost and low power requirement makes the sensor as best selection for many applications.

TABLE II:
Comparison of different sensors

Specifications	DHT22	DHT11	SHT71
Humidity Range	0-100%	20-80%	0-100%
Temp. Range	-40°C-80°C	0°C-50°C	-40°C-123.8°C
Accuracy	±2% (Humd) ±0.5%°C(Temp)	±5% (Humd) ±2%°C(Temp)	±3% (Humd) ±2%°C(Temp)
Typical Prize	$4-10	$1-5	$30-50

The table shown comparing different temperature and humidity sensors available such as DHT11, DHT22 and SHT71 proves that choice of DHT22 is best because of its low cost and more accuracy.

III. Software aspects

A. MQTT Protocol

MQTT (Message Queue Telemetry Transport) is lightweight, publish-subscribe based protocol for messaging with ISO standards which can be used on top of TCP/IP protocol. The main intention in the design of MQTT protocol is to make

connections where bandwidth of the network is restricted. This protocol messaging requires a message broker for sharing all the data published by clients to the other interested clients whoever subscribes the broker regarding the message topics. MQTT protocol is mainly designed for supporting wireless networks with highly differing latency levels caused due to irregular bandwidth limitations and inconstant connections.The MQTT architecture is explained in Fig. 4.

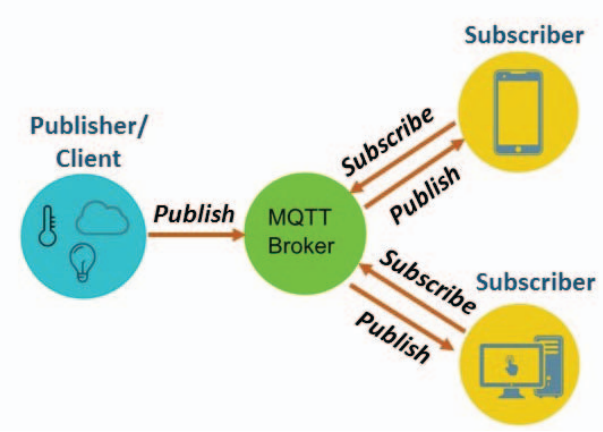

Fig. 4: Architecture of MQTT Protocol

Eclipse Mosquitto is open source MQTT protocol supported message broker. The light weight feature of MQTT protocol makes it perfectly suitable for all Internet of Things (IoT) applications which involves messaging between low power sensors, microcontrollers, Mobile phones and various computing devices.

The Mosquitto broker was installed on raspberry pi module and is run on it. The NodeMCU ESP8266 microcontroller acts as MQTT client and publishes the data on the Mosquitto MQTT broker running on the raspberry pi module.

B. SQLite Database

SQLite database is a compact and in-process library, which do not have a separate server but reads the data and writes the date to disk files. It is also known as embedded SQL database engine. SQLite database is self-contained, serverless, null configured and transactional SQL database. It is an open source library and thereby most predominantly deployed database around the world with numerous applications. SQLite database offers high resistance to corruption of data but certain bugs in Operating System and hardware are responsible for some corrupt database files. While processing the statements in SQL, it uses some temporary file potentially. SQLite database as stated generally stores all the information in ordinary disk files, but still the database can be used as in-memory storage database.

In our implementation the SQLite dataset engine is installed on the Raspberry Pi and a python application is run on the Raspberry Pi which subscribes the data from MQTT Broker and gets the data from broker in JSON format. This data is formatted using HTML application and is stored in the SQLite database columns.

IV. High Level Design and Implementation

We shall discuss the high level design and implementation of the framework in a detailed a manner. Firstly the Raspberry Pi module is imaged with Raspbian Jessie operating system. The VNC server of Raspberry Pi module is enabled and also Wi-Fi is enabled so that the Pi module can be remotely controlled. Now SQLite database is installed on the module and tables are created according to our requirement and are initialized. Next, Mosquitto Broker should also be installed on the Pi module and is run on the Module.

Now coming to NodeMCU ESP8266 Microcontroller, it is powered with external battery source of 5V. Meanwhile the DHT22 temperature and humidity sensor is interfaced with NodeMCU. The DHT22 sensor gives digital calibrated output signal by temperature and capacitive humidity sensing, using 1-wire protocol from communication between ESP8266 and sensor. The NodeMCU is programmed using Aurdino IDE and Raspberry Pi module IP address is passed to the NodeMCU on which MQTT broker is running. Now the NodeMCU publishes sensor data to the MQTT broker installed on the Pi module. The high level design and implementation steps are described in Fig. 5.

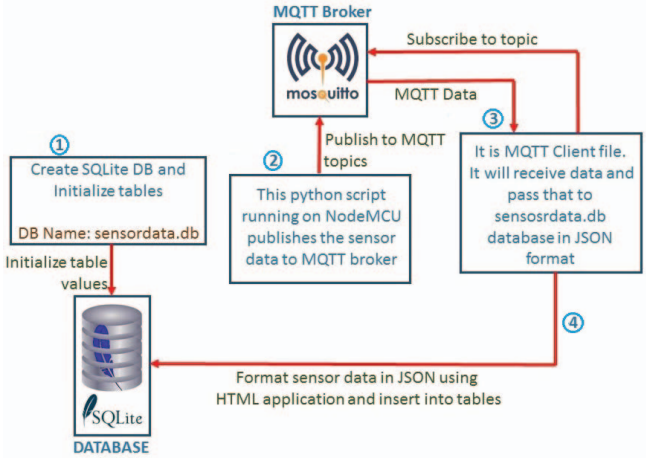

Fig. 5: High level implementation of system

The Mosquitto MQTT on Pi module Broker listens to NodeMCU MQTT client. The python application is run on Pi module which subscribes the senor data from MQTT broker and gets back sensor data in JSON format. Now this JSON sensor data is stored in SQLite database and is JSON data is formatted using HTML application and the sensor data is stored in the SQLite database tables. Now the tables with sensor data can be displayed by running localhost on the network. To access the sensor data remotely the Pi module is connected to Weaved IoT services by enabling SSH on port 22. Now the sensor data can be accesses from anywhere in the world by logging in to particular Weaved service account securely.

V. Experimental Results

The circuit connections are made as shown in the Fig. 6 and programmed using Aurdino IDE and MQTT Broker Raspberry

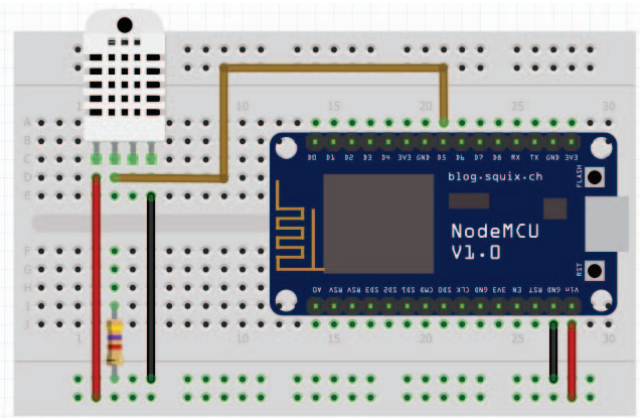

Fig. 6: Circuit Connection

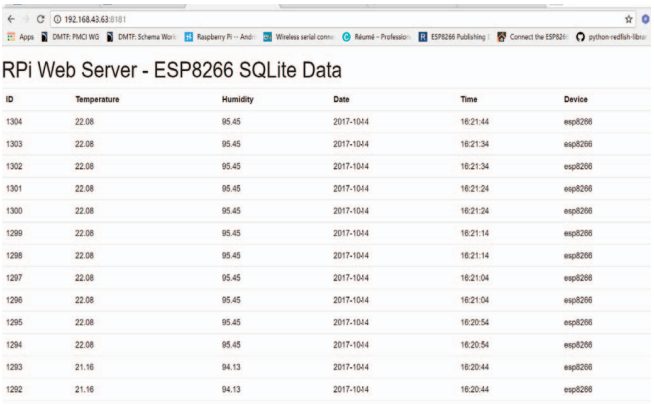

Fig. 8: SQLite Database on Raspberry Pi Web Serer

Pi IP address and Wi-Fi credentials are also included in the code to enable NodeMCU connect to Internet and to publish the sensor data to Mosquitto MQTT Broker on Raspberry Pi module. The Fig.9 shows that NodeMCU is connected to Internet and publishing data on MQTT Broker on Raspberry Pi.Now the data published on Mosquitto MQTT broker by NodeMCU MQTT client is subscribed by the python application on Raspberry Pi and is stored in SQLite Database. The data stored in the SQLite database in JSON format is

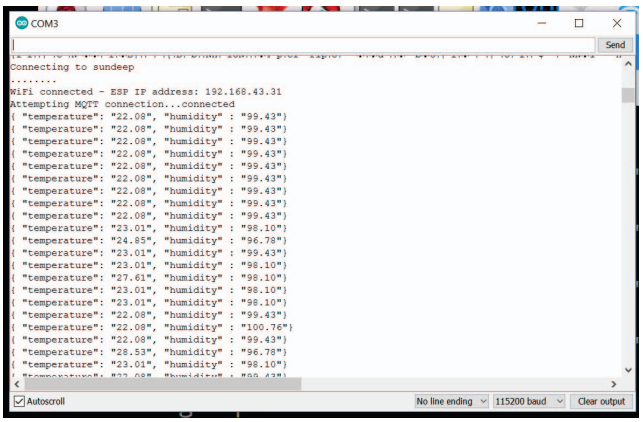

Fig. 7: NodeMCU connection to broker and publishing data to MQTT brokerr

formatted using HTML application and sensor data database can be accesses by entering Raspberry Pi IP address in the browser. The Fig. 8 shows the Raspberry Pi web server which contains sensor data stored in SQLite database.

VI. Conclusion and Future scope

We have designed a low cost, ultralow power consumption and highly efficient temperature tracking system by using Raspberry Pi module, NodeMCU ESP 8266 microcontroller and DHT22 Sensor. Here, the most important feature in the implementation is Raspberry Pi module acts as both Mosquitto MQTT broker to get sensor data from clients, to distribute data to subscribers and also SQLite database installed on the Raspberry Pi stores the sensor information sent by the MQTT Clients by subscribing the MQTT broker. There are wide number of applications for this system. The system can be implemented for accessing the temperature and humidity of anyplace remotely. It can also be implemented in offices, industries and homes for security purposes. Further improvements are possible by integrating different sensors to the NodeMCU ESP8266 modules and in the same way many NodeMCU module can be integrated with Raspberry Pi module, so that sensor data can be stored on Raspberry Pi SQLite database separately in different tables. This saves a lot of power, storage memory and also user receives a weather information of any place from anywhere in the world.

References

[1] Kevin E. Trenberth, Kathleen Miller, Linda Mearns and teven Rhodes, Effects of Changing climate On weather And human activities, National Center for Atmospheric Research Boulder, Colorado
[2] Dave Evans, The Internet of Things, How the Next Evolution of the Internet Is Changing Everything, Cisco Internet Business Solutions Group (IBSG), April 2011
[3] Tejas Thaker, ESP8266 based Implementation of Wireless Sensor Network with Linux Based Web-Server, 2016 Symposium on Colossal Data Analysis and Networking (CDAN)
[4] J. Gubbi, R. Buyya, S. Marusic, M. Palaniswami, "Internet of Things (IoT): A vision architectural elements and future directions", Future Generation Computing System, vol. 29, pp. 1645-1660, 2013.
[5] Atabekov Amir, Marcel Starosielsky, Dan Chia- Tien Lo, and Jing Selena He. "Internet of Things-Based Temperature Tracking System", 2015 IEEE 39th Annual Computer Software and Applications Conference, 2015.
[6] T. O'Reilly, "What is Web 2.0: Design patterns and business models for the next generation of software", International Journal of Digital Economics, vol. 65, pp. 17-37, 2007
[7] R. K. Kodali and K. S. Mahesh, "A low cost implementation of MQTT using ESP8266," 2016 2nd International Conference on Contemporary Computing and Informatics (IC3I), Noida, 2016, pp. 404-408.
[8] Y. Upadhyay, A. Borole and D. Dileepan, "MQTT based secured home automation system," 2016 Symposium on Colossal Data Analysis and Networking (CDAN), Indore, 2016, pp. 1-4.
[9] R. A. Light, Mosquitto: server and client implementation of the MQTT protocol, The Journal of Open Source Software, vol. 2, no. 13, May 2017
[10] https://www.raspberrypi.org/
[11] https://www.sqlite.org/
[12] ttps://espressif.com/en/products/hardware/esp8266ex/overview
[13] http://mqtt.org/
[14] S. Wagle, "Semantic data extraction over MQTT for IoTcentric wireless sensor networks," 2016 International Conference on Internet of Things and Applications (IOTA), Pune, 2016, pp. 227-232.