Multimedia Tools and Applications (2021) 80:2931–2949
https://doi.org/10.1007/s11042-020-09750-4

MQTT protocol employing IOT based home safety system with ABE encryption

Vatsal Gupta[1] · Sonam Khera[2] · Neelam Turk[2]

Received: 13 January 2020 / Revised: 21 August 2020 / Accepted: 27 August 2020 /
Published online: 18 September 2020

Abstract
This project aims towards the usage of MQTT (Message queueing telemetry transport) protocol in IoT (Internet of things) along with adopting a feasible means of encrypting the message transfers in applications. The lightweight nature of MQTT protocol makes possible the transfer of information speedily and hence, has now being used in applications related to IoT, WSN (Wireless sensor networks), and M2M (Machine to machine) communications. Here, MQTT is deployed between ESP-8266 Wi-Fi SoCs namely- NodeMCU ESP-8266 12E and ESP-01 8266 Wi-Fi modules. Both communicate using the MQTT protocol using the internet via Wi-Fi. The ESP-8266 Wi-Fi SoCs have completely revolutionized approach towards IoT because of numerous advantages. This project also includes the usage of 2 sensors namely- PIR (Passive InfraRed) motion detector sensors and an MQ-5 gas sensor which sense human presence and some gases respectively. These sensors read the environment around them for the required information, systems encrypt that information and subsequently, transmit the data over the internet to the MQTT broker stationed at the cloud. This encrypted data is then sent to a central node- NodeMCU ESP-8266 12E which decrypts it and then alerts the user about any mishappening, through the Blynk app.

Keywords IoT · MQTT protocol · ABE encryption · KP-ABE · WSN · ESP · PIR · MQ-5 · Blynk · cloudMQTT

✉ Vatsal Gupta
vatsal.gupta97@gmail.com

Sonam Khera
sonamkhattar@yahoo.co.in

Neelam Turk
neelamturk@gmail.com

Extended author information available on the last page of the article

1 Introduction

Internet of Things is a very commonly used term whenever someone talks about emerging technologies. It is often mentioned that the future of technology can be symbolized by ABCD-I meaning "Artificial Intelligence, Block chain, Cloud, Data analytics – Internet of things" in reports such as in [3]. Many advancements have not been achieved on these five topics. Perhaps suggesting that they are still evolving, they have much potential in the future market. IoT is both hardware and software-based. Applications of IoT are endless and so are its solutions. There is not any fixed way to make an IoT system from given specifications. Some IoT applications include smart homes, smart wearables, smart highways, smart grids, smart vehicles (Internet of Vehicles), smart industries (Industrial Internet of Things) etc. Realization of any of these applications in the real world can only be done by infusing hardware devices with processing power and networking or simply saying, making electronic items smart. The idea of IoT is not new, it is just that this field has a variable timeline because everyday new and advanced concepts come up. Making things smart means that the items/devices need to be equipped with a "brain of its own" to take logical decisions, networking means, and sometimes memory. We are putting memory as an option because nowadays, cloud computing is picking up pace in the IT (Information and Technology) sector. This project aims at smart home security along with small usage of cloud memory. IoT, when clubbed with WSNs (Wireless Sensor Networks) provides major advantages to traditionally employed systems. In [9], M. Kocakulak and I. Butun provided an overview of WSN towards IoT, like advantages, disadvantages, potential, and applications. In fact, in [12], L. Mainetti et al. suggested new installations in the already existing IoT networks and migration to an all-IP based environment. They also aid in developing smart and hybrid systems in highly important applications like in defence, as concluded in [11]. For networking, Wi-Fi (Wireless Fidelity) is used and for message transfer over the internet, MQTT (Message Queueing Telemetry Protocol) protocol is used, unlike conventional HTTP (HyperText Transfer Protocol) protocol. Although MQTT protocol has its competitors, we chose MQTT over others because of its popularity, simplicity of use and experienced open source support. For enhanced security in this MQTT application, we propose the use of encryption methods which is suitable for IoT and is compliant with MQTT protocol. We decided to work on ABE (Attribute Based Encryption) as explained in [6]. Conventional SSL/TLS with certificates and session key management is not a good option as storing and managing the certificates and key exchanges are cumbersome in cases of networking between heterogeneous devices. Additionally, SSL/TLS experience different bouts. Thus, an ideal security mechanism to be used in the IoT environment employing the MQTT protocol must be lightweight and most importantly, overcome the shortcomings of the MQTT protocol from a security point of view. In this regard, [13] has done significantly well and their method is referred in this project. The MQTT protocol is chiefly used with devices which are memory, bandwidth, and computational powers constraint. Hence IoT devices such as sensors and new-age small size microcontrollers, like we have used, benefit a lot by mainly saving time instead of creating web servers and then communicating via HTTP requests which are heavy in comparison with MQTT requests. Such communication protocols like MQTT greatly reduce the load on the processor, Wi-Fi communication chip, or circuitry and thus bring down latency between message transmission from one source and message reception at the receiver's end. Furthermore, the usage of Blynk app for handheld smartphone control and status checking immensely helps the user by creating alerts, especially for cases whenever a sensor detects something unconventional or unusual.

2 Literature survey

Internet of things, a very synchronous word which can be heard in every technological area, is expected to be market of 100 billion dollars till 2050. IoT is expanding its horizons very rapidly, so much that new terms like IoV (Internet of Vehicles) and IIoT (Industrial Internet of Things) have started to progress. Out of numerous applications, smart homes systems largely occupy the IoT scenario and so, in this project, home security has been taken up as a topic. Like in [8], A. Jose and R. Malekian mentioned some useful points towards the improvement of smart home security. In it, they used motion sensor, proximity sensor, contact sensor, temperature & humidity sensor, gas sensor along with Zigbee as the communication device and NodeMCU & Arduino as microcontrollers. The real-world scenario clearly explains the need for such a system. Infiltration/Robbery attempt can be fully blocked if one sets up such a smart home security system employing PIR (Passive InfraRed) sensors, wherein an illegal attempt would be known to the user as soon as the attempt is made. This enables the user to either warn the infiltrator about his/her information of the illegal act or help approach the police about the illegal move. Adding about the other sensor, the leakage of domestic use gas would get detected if one uses the MQ-5 sensor, properly calibrated. Both these sensors work satisfactorily, and quite promptly. Other devices used in this project are NodeMCU ESP-8266 board, 3 ESP-01 8266 Wi-Fi modules. All are explained in the upcoming sections.

ESP8266 is produced by the manufacturer, Espressif Systems in China. ESP8266 is a Wi-Fi enabled SoC (System on Chip) with full TCP/IP stack and microcontroller capability. ESP8266 is capable of functioning in a wide operating temperature range, making them durable in industrial applications. Being an SoC type of chip and self-contained with Wi-Fi networking capability, it can be used either as a standalone processor or a slave to another microcontroller. One of the main applications of this microchip is its usage in wireless sensor networks, in which networking and power consumption are the main issues.

Many modules have been built by Espressif Systems which are based on this ESP8266 microchip. Some other manufactures along with Espressif Systems, like NodeMCU, Wemos, and Adafruit, built development boards with ESP8266 as its working heart. In this project, one of ESP8266 module named as ESP-01 8266, produced by a third-party manufacturer Ai-Thinker has been used.

ESP-01 8266 is a self-contained SoC, 8 pin Wi-Fi module that doesn't require a micro-controller to be controlled about its inputs and outputs. It can work as a single component of any IoT related work or can be interfaced with a microcontroller or a development board. Such compact modules can engine major transformations like IoT and WSN (wireless sensor networks) in the real world, which has already been mentioned above and described in [9]. As a future aspect, if extended to a larger area, it may utilize other ESP8266 compatible modules according to the user requirements and other functionalities.

Figure 1 shows the picture of the Wi-Fi module used as sensor nodes. Furthermore, one of the development boards- NodeMCU ESP8266 is quite popular amongst makers. Its more advanced than Ai-Thinker module.

As can be seen in Fig. 2, NodeMCU board has greater number of pins than the Ai-Thinker Wi-Fi module ESP-8266. It means that it has greater number of GPIO (General Purpose Input/Output). NodeMCU board also has a larger amount of RAM and Flash memory.

Some differences between NodeMCU board and ESP-01 8266 Wi-Fi modules are shown in Table 1.

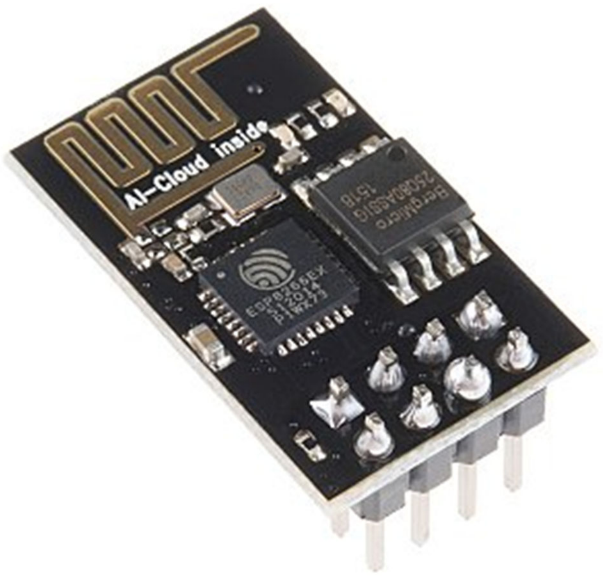


Fig. 1 ESP-01 8266 Wi-Fi Module

Sensors: Sensors play the most important role in the vast field of IoT. Their usage has become a common thing, as simple combinations of power source, wires, connectivity devices and passive elements cannot be used alone to achieve what we desire. Sensors collect, detect, measure, alert, and do other numerous tasks in an IoT ecosystem. The "thing" in IoT can justifiably be referred to these sensors. 2 sensors used in this project are motion detector sensors and a gas sensor.

PIR (Passive Infrared sensor) HC-SR 501: It is a 3-pin motion detector sensor capable of sensing infrared radiation from a living body which comes in its field of view. It works entirely by detecting infrared radiation (radiant heat) emitted by or reflected from objects. This sensor is small, inexpensive, require low power, easy to use, and are reliable when put to work in suitable conditions. A detailed explanation regarding PIR sensors intrinsic working is given in [1]. PIR sensor is made of pyroelectric sensor which can detect the presence of infrared radiations. All bodies emit radiations, although hotter bodies emit more radiation than other bodies. Humans radiate this thermal energy with a wavelength of approximately 8 μm to 12 μm (micrometer) at all times. This radiation is sensed by the pyroelectric sensor placed

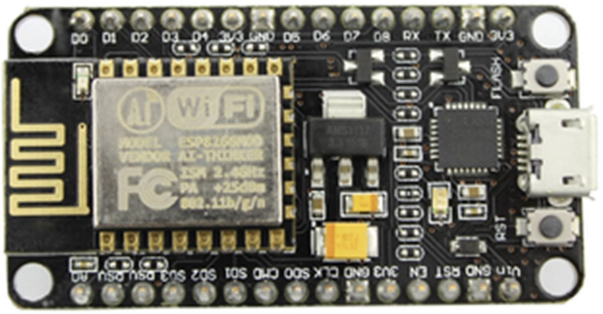

D0 D1 D2 D3 D4 3V3 GND D5 D6 D7 D8 RX TX GND 3V3
FLASH
RST
A0 RSV RSV SD3 SD2 SD1 CMD SDO CLK GND 3V3 EN RST GND Vin

Fig. 2 NodeMCU

Table 1 Differences between the 2 Microcontroller chips

Characteristics	NodeMCU ESP-8266 v1.0	ESP-01 8266
GPIO pins	11	2 (+2 if Tx and Rx pins are not used)
Analog-Digital pin(s)	1	0
Memory	128 KB	32 KiB instruction, 80 KiB user data
Flash memory	4 MB	1 MB
USB to serial converter	Yes. CP2102 is used.	No
Is breadboard friendly	Yes	Not so much
Form factor	Big	Small
Cost	Costlier than ESP-01 8266	Cheaper than NodeMCU board

inside a translucent dome-shaped lens. At the top of the sensor, one may find an infrared filter. It suitably selects the wavelength which it has to work on and since it has to mainly detect human presence, this filter selects between 8 μm to 12 μm.

Figure 3 shows the front and back views of the PIR sensor. One of the most important components of the PIR sensor module is the Fresnel lens that is placed on the top of the rectangle-shaped IR filter window. It is a specially designed lens with beehive structure, intended to magnify the range of the sensing array. The Fresnel lens can make the PIR module to have a maximum sensing range of 7 m and an angle of 110°×70°. The on-board potentiometers can be used to adjust the sensing intensity and approximate sensing range. In this project, both were kept at a high position to make sure that range is maximum and even slight radiation from a fast-moving living being can be detected. PIR sensor module works best in low noise, low lighted areas and they find their application in automatic lights, security systems, lifts & lobbies, etc.

MQ-5: It is one of the members of the MQ family of gas sensors, primarily deployed to sense the concentration of natural gas, town gas and liquified petroleum gas. It also has a small sensitivity for smoke and alcohol. Fuels such as natural gas, town gas and liquified petroleum gas are being used in different parts of the world for various domestic and industrial purposes and so, a unified sensor such as MQ-5 was held suitable for this. In this project, the concentration of piped natural gas (PNG), which is being used as cheaper and cleaner fuel in Indian households, had to be primarily sensed. It can sense the above-mentioned gases in parts-per-million (ppm) in range 200–10,000 ppm. As this sensor can sense many gaseous compounds and the analog type output can range from 0 to 1023, this sensor cannot be normally used to measure the concentration in ppm (range: 200–10,000 ppm). Therefore, it is required to be calibrated for suitable values and for this, the graph (R_S/R_O v/s concentration in

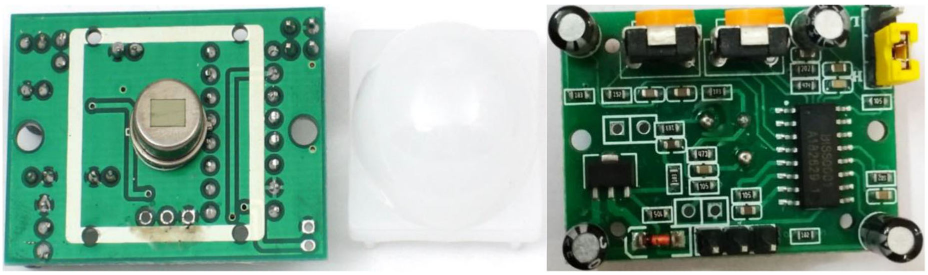


Fig. 3 Front and back views of PIR sensor

ppm) in the datasheet comes conveniently handy. Along with some calculations explained in [5], we could properly calibrate the sensor for methane gas concentration.

The front and back views of the MQ-5 sensor module is shown in Fig. 4.

MQTT protocol: MQTT (Message Queueing Telemetry Protocol) is a lightweight M2M (machine to machine), publish/subscribe type protocol, designed in 1999 by 2 IBM employed engineers- Andy Stanford-Clark and Arlen Nipper for connecting oil pipeline telemetry systems over satellite. It was started as a proprietary protocol, until 2014 when MQTT version 3.1.1 was officially declared an OASIS standard [2]. MQTT follows publish/subscribe methodology. It is a very lightweight and binary protocol, and due to its minimal packet overhead, MQTT excels when transferring data in comparison to other similar protocols like HTTP. This comparison is also summarized in [14]. It is more suitable for asynchronous communication model. MQTT becomes more important when it is used with resource and bandwidth-constrained devices. Thus, IoT devices become perfect consumers of the MQTT protocol. The format of the message header is shown in the following table [7].

The fact that MQTT control message length is comparatively short, is supported by Table 2. Various message types are used in this protocol and are distinguished via way of means of the MQTT message header. Message type '0000' cannot be used as it's far reserved for the future. Variable Header carries the username and password flag (can facilitate user authentication), upon placing them, corresponding values also are included in the payload.

There are two main terms associated with MQTT- broker and client. All the devices which are to utilize MQTT protocol are called MQTT clients. A single client can be a publisher or a subscriber or both but can behave like any one of it at a time. An MQTT broker acts as a distributor by establishing a connection between different MQTT clients in its own virtual space.

The basic working of MQTT protocol can be simply understood from Fig. 5. As can be seen from the figure, the different electronic devices/sensors communicate with the devices (laptop, tablet, or phone) via MQTT protocol. A brief explanation about certain terms related to MQTT is given below:

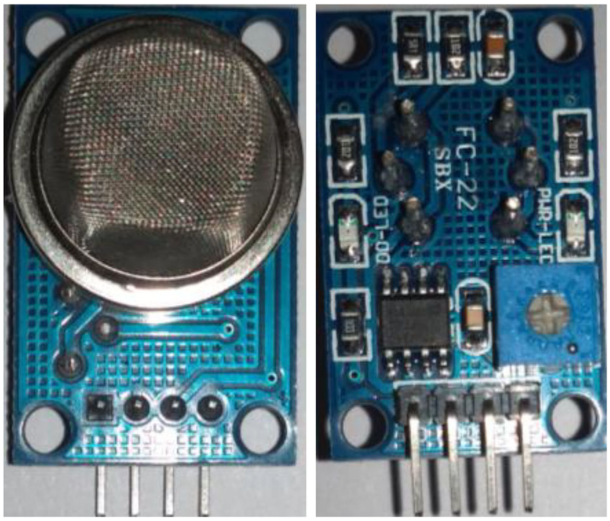


Fig. 4 Front and back views of MQ-5 sensor

Table 2 MQTT control message length

Bit	7	6	5	4	3	2	1	0
Byte 1	Message type				DUP	QoS		Retain
Byte 2	Remaining Length							
	Variable Header							
	Payload							

Broker: It is the most vital part of any MQTT connection. An MQTT broker is either hosted on a device which could manage the entire communication scenario or on the cloud. Cloud-based brokers are more popular than device-based brokers. The broker is responsible for receiving all messages, filtering the messages, determine who is subscribed to each message, and then directing the message to those subscribed clients. In this project, the broker would also function as a PKG (Public Key Generator) to generate master secret key and access policy. It publishes the public parameters for each attribute of the access policy.

Client: Clients simply communicate with the broker. An MQTT client runs an MQTT library or an SDK and connects to an MQTT broker over a network. There are over a dozen libraries available for C, C++, Go, Java, C#, PHP, Python, Node.js, and Arduino. Both publisher and subscriber are called MQTT clients. Each client would have to identify itself before the PKG along with its own set of attributes.

Topic: Topics are something about which an MQTT client publishes, and they are also which clients subscribes to as well. Topics are the way one registers interest for incoming messages or how one specifies where one wants to publish the message. It acts as the central distribution hub for the publishing and subscribing messages. A client publishes data to a specific topic. The message attached with to topic is sent to the MQTT broker via the internet. This specific topic has to be subscribed by another client (although publishing and subscribing to the same topic is allowed) which then gets the message from the MQTT broker. Topics are simple, hierarchical strings, encoded in UTF-8, and delimited by a forward slash. For example, “building1/staircase/lights” and “building1/room1/proximity” are valid topic names.

There are several MQTT brokers available, Eclipse Mosquitto™ being the most popular one. In this project, we used a cloud-based broker named *CloudMQTT*. We used cloud-based broker because for device hosted broker, an additional, more powerful board having more memory, preferably Raspberry Pi had to be used. Usage of new hardware would also take along with itself the attached shortcomings. To subtract those shortcomings, a cloud-based

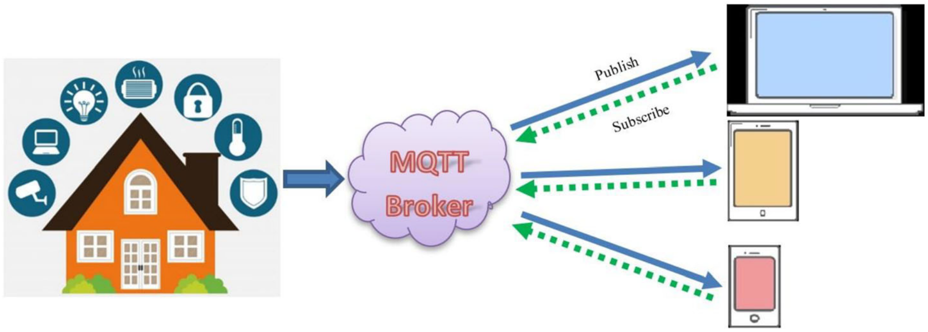
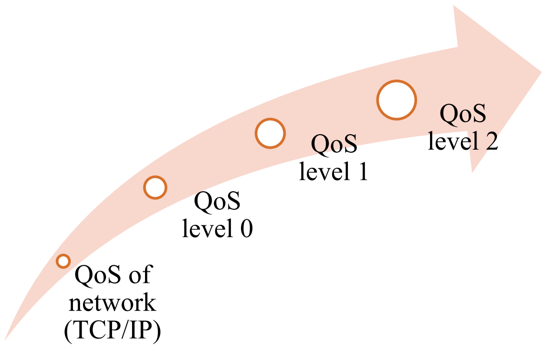


Fig. 5 Working of MQTT protocol

broker was used. To get started with a broker, one simply needs a hostname/IP and port of the broker. A client starts by creating a TCP/IP connection to the broker by using either a standard por or a custom port defined by the broker's operators. As in our case, CloudMQTT provided the server "postman.cloudmqtt.com", to run on port- "13524". A unique username and password are also provided when one creates an instance in CloudMQTT, which is used when programming clients. As already mentioned, MQTT is a lightweight protocol, meaning that its minimal control message can be of size 2 bytes and can carry maximum 256 MB if needed. Its communication is TCP protocol-oriented although it's another variant- MQTT-SN is used for other transports such as UDP or Bluetooth. Owing to its lightweight nature, MQTT sends connection credentials in plain text format and does not comprise of any measures for security or authentication. Security or Authentication can be provided by the underlying TCP transport protocol using measures to shield the integrity of transferred message/information from interception or replication. MQTT message transfer also comprises any one of three levels of QoS (Quality of Service). This QoS support is solely provided by the broker. Even though TCP/IP provides guaranteed data delivery, data loss can still occur if a TCP connection breaks down and messages in transit are lost. Therefore, MQTT adds 3 quality of service levels on top of TCP. The three QoS levels are described below:



QoS level 0: At most once. The message is sent only once and neither the client nor the broker takes additional steps to acknowledge delivery. It is much like "fire and forget" methodology. Needed where readings like from temperature sensor is to be regularly published. Then the loss of a single value is not critical since the regular sending of data from sensors will anyway integrate the values over time and loss of individual samples becomes irrelevant.

QoS level 1: The message is resent by the sender multiple times until its acknowledged but here, duplicity is unavoidable. Needed where a single state change is vital. Like we used proximity sensors in our project whose sensitivity is adjusted at maximum. Any change of state from 0 to 1 becomes critical for home security so that change must be not be left unnoticed by the system.

QoS level 2: This is the highest level of QoS in MQTT. It incurs the largest overhead in terms of control messages. The sender and receiver participate in a two-level handshake to certify only one copy of the message is received. This level is mostly not included with free of cost broker instances. It is used in applications where duplicate events could lead to incorrect actions.

As mentioned in the introduction, for additional privacy and security, methods of encryption were employed. For this, lightweight ABE is preferred primarily because it supports broadcast encryption. Broadcast encryption means that with one encryption, a message is transported to multiple intended users and so, suitable for IoT applications. ABE are of two types: CP-ABE (Ciphertext Policy based ABE), and KP-CBE (Key Policy based ABE). Both these schemes are very different and thus, have their share of applications. Attributes here can be understood as ‘country in which the device is kept’ or ‘typical climate of the place in which that device is kept’. As can be understood from [15], generic publish-subscribe (pub-sub) model employed for the IoT environment received a piracy provision based on ABE. In that, the message payload is encrypted using AES (Advanced Encryption System) and AES key is encrypted by means of ABE, ensuring that payload and ciphertext sizes are the same. In [13], the authors aimed at optimizing complex operations of ABE by using suitable cryptography parameters rather than performing double encryptions.

As mentioned earlier also, variable header in the MQTT protocol message format carries the username and password flag (facilitates user authentication). However, these values are not encrypted in the message and as a result, are now no longer secure. While referencing [13], the authors are seen to have added type of encryption methods and have proposed SMQTT which uses earlier reserved ‘0000’ message header.

In the setup phase, the PKG gets registrations from all the clients connected, be they subscriber or a publisher. These clients provide the PKG with a unique identity along with its attributes. PKG also generates a master secret key set and public parameters in accordance with the CP/KP-ABE scheme and publishes public parameters along with the universal attribute set U (all the attributes which a client device provides to the PKG are a subset of U). After this, an access policy gets designed by the publisher device. Since here, the client devices are limited and lie at the same access level and topics and the subscriber is known a priori, KP-ABE scheme held suitable for the encryption part. Publisher directs the access tree to PKG and it generates the keys and its policy accordingly. Subscribers now get their set of keys for all the required policies a priori.

During the communication phase, a client can perform publish, subscribe, unsubscribe and ping operations. The publish operation sends a binary block of data- the content, to a topic that is defined by the publisher. The subscribe operation subscribes the client to one or more topics so that it may receive all the messages pertaining to that topic. The unsubscribe operation unsubscribes the client from the topic that the user wishes to. Lastly, ping roughly translates “are you alive/yes I am alive” to the server. This is the only function which helps maintain a live connection and ensure the TCP connection has not been shut down by a gateway or router.

Publisher client encrypts data using public parameters and produces ciphertext conferring to KP-ABE. When transmitting a message or an alert, publisher embeds encrypted data as payload. It’s then sent to the broker. Broker and the publisher then exchange acknowledgement packets to confirm delivery. Broker forwards all messages to their respective subscribers and deletes data and informs the publisher. Subscriber decrypts the messages if it satisfies access policy using its private attribute keys. Following KP-ABE, subscriber device verifies whether it following the access policy. In case it satisfies, then it asks the PKG for a corresponding key set. PKG authenticates the demand and sends the key to the subscriber.

2 essential functions when working with MQTT protocol are- reconnectmqttserver and callback function. A short description of both the function is given below:

reconnectmqttserver(): This function is mentioned in the void loop() body in the starting so that it runs firstly, as the void loop() body runs in an infinite loop. It takes in no parameters and also returns nothing. Its sole work is to check the client’s connection with the broker and establish the connection again in case the client gets disconnected due to any reason.

callback(): This function is called by the underlying MQTT library when a new message is received from the broker (for a topic the user has subscribed to). Its takes in 3 parameters-topic, payload and length. Topic refers to the topic under which the message has come. Payload refers to the actual incoming message and length refers to the length of the incoming message plus 1 (for the last null terminator '\0'). A loop is usually run from the first character until the last character and stored in a user initialized character array, which is then used for further processes.

Drawing conclusions from [13, 14], MQTT advantages, disadvantages, and its applications are discussed below.

MQTT Advantages:

1. According to measurements in 3G networks, the throughput of MQTT is 93 times faster than HTTP's.
2. Ideal for constrained networks. It works without any glitches where connections are of low bandwidth, high latency, contain data limits and are quite fragile.
3. Low size of control message packet which makes packet transmission seamless.
4. It is less complex and easy to implement in IoT related devices.
5. Suitable for deployment in a demilitarized zone (DMZ).
6. Has 3 levels of QoS for enhanced acknowledgments.
7. Its LWT (Last Will and Testament) feature is highly useful when the user needs to know about clients' abrupt disconnection from the network.
8. Has a flexible subscription pattern.

MQTT disadvantages:

1. MQTT lacks important security features. At the time it was being made, the connection was not needed to be secured, as told by Stanford-Clark. MQTT has minimal authentication features. Username and passwords are sent in clear text and any form of secure use of MQTT must employ SSL/TLS.
2. Authentication of clients with client-side certificates is not included in MQTT protocol, to regulate who owns a topic and who can publish a message on it. This makes it very smooth to inject harmful messages, either deliberately or by mistake, into the system.
3. The Broker needs to be 100% available all the time. Since all the clients have to report to the broker, so any unavailability of the broker would prove to be lethal for the whole system.
4. Interoperability between different employed systems. Message payloads are binary in MQTT, with no information as to how they are encoded. Hence, problems can arise especially in open architectures where different applications from different manufacturers are supposed to work seamlessly with each other.
5. There is no way for the subscriber to know who sent the original message unless that information is contained in the real message.

MQTT Applications: the following are some of the real world MQTT applications:

1. Facebook has used aspects of MQTT in Facebook Messenger for online chat. However, it is unclear how much of MQTT is used or for what.
2. Amazon Web Services announced Amazon IoT based on MQTT in 2015.

3. Adafruit launched a free MQTT cloud service for IoT experimenters and learners called Adafruit IO in 2015.
4. Microsoft Azure IoT Hub uses MQTT as its main protocol for telemetry messages.
5. Open-source software home automation platform Home Assistant is MQTT enabled and offers four options for MQTT brokers.
6. The OpenStack Upstream Infrastructure's services are connected by an MQTT unified message bus with Mosquitto as the MQTT broker.

Blynk: Blynk is a platform with IOS and Android apps to control Arduino, ESP, Raspberry Pi and various popular development boards over the Internet. It has a digital dashboard where graphic interfaces for IoT projects can be conveniently built by simply dragging-and-dropping widgets. One also needs to install the Blynk Arduino Library, which helps generate the firmware running on a microcontroller. Blynk works with hundreds of hardware models and connection types like Wi-Fi, Bluetooth, and ethernet. Blynk is highly popular and one can find support regarding any project over discussion forums of several websites. One could use an MQTT websocket app just as used in [10] but in this project, we didn't want to control the devices but to get real-time data and alerts on our smartphone. These apps like used in [10] may show the data but they do not really the system over any disconnection or other alerts, and that is why Blynk app was used.

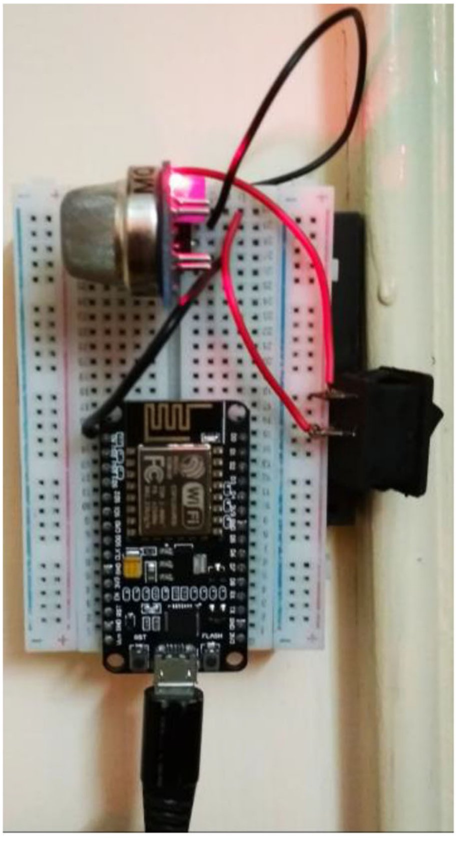


Fig. 6 NodeMCU functioning as a subscriber

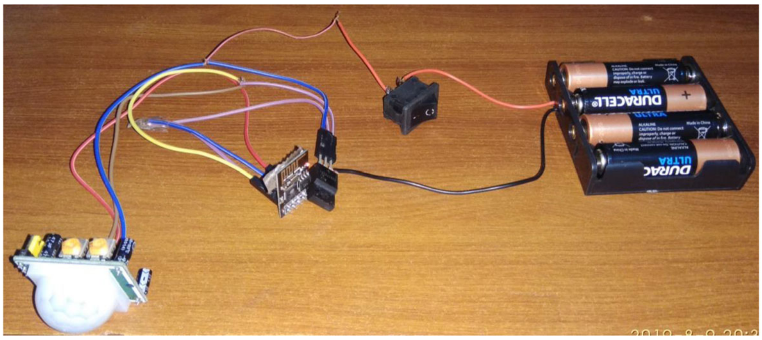


Fig. 7 One of the device functioning as a publisher

3 System model

In this project, 3 PIR sensors are used along with a single MQ-5 gas sensor. These all are connected to individual ESP-01 8266 Wi-Fi modules. The small Wi-Fi module being used also hosts a microcontroller which can act as both- a processor as well as a Wi-Fi transceiver. The sensors collect their required information and the Wi-Fi module connected to it upload the collected data to the cloud MQTT broker for accessing, parsing and sending the data to its appropriate subscriber. The MQTT broker here also functions as the PKG. Here only one subscriber exists and that is the NodeMCU ESP-8266 12E module which has been subscribed to all the topics. The reason to include this larger module is that its computational capability is

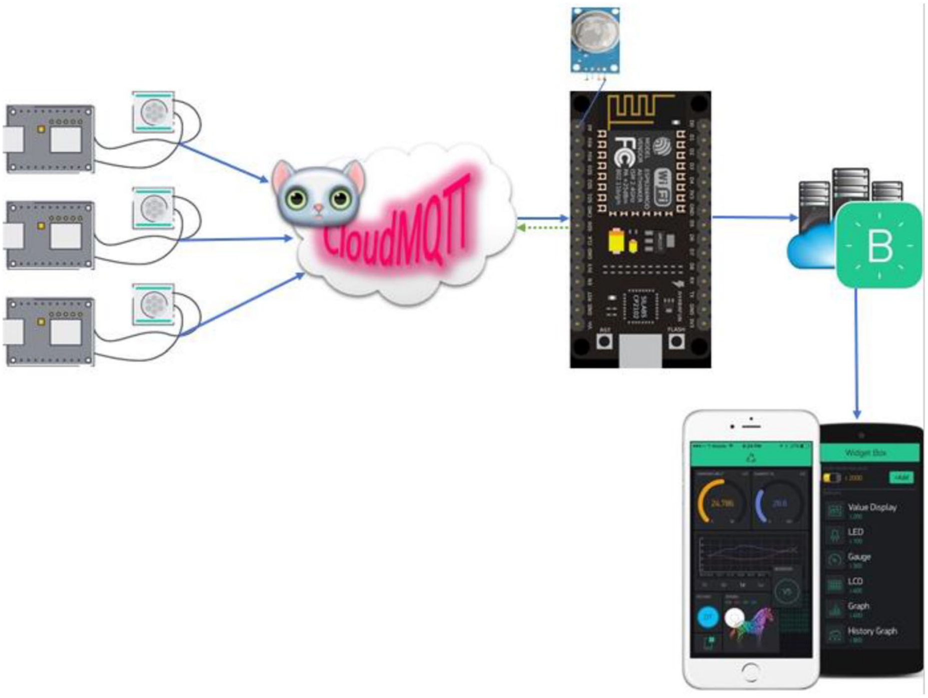
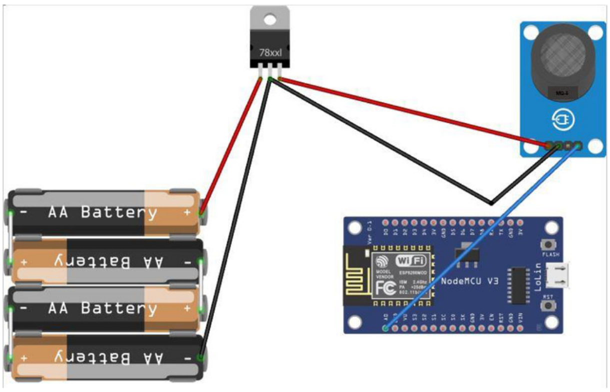
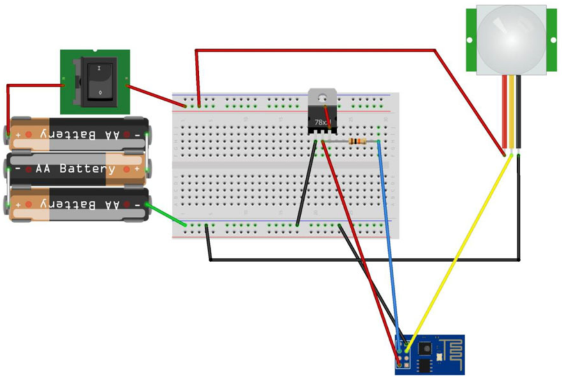


Fig. 8 Pictorial block diagram of the whole IoT based subscriber - publisher model



(a) Subscriber side connection



(b) Publisher side connections

Fig. 9 Physical connections of the whole system. (**a**) Subscriber side connection. (**b**) Publisher side connections

greater than the small sized module and for interacting with the Blynk app alone else every Wi-Fi module had to be registered with the Blynk app along with 3 different authorization tokens. This hassle was cut down by placing NodeMCU module as a virtual centre while adding more features.

The actual subscriber system used is depicted in Fig. 6. As MQTT is a publish-subscribe type communication protocol, the individual ESP-01 8266 which is placed at different locations, publishes at a particular/unique topic regarding the sensor status. PIR sensors are placed at the staircase, drawing room, and bedroom and publish their status (whether "0" or "1"). While the gas sensor wasn't clubbed with the ESP-01 8266 because this module lacks analog and PWM pins. So, the MQ-5 gas sensor was clubbed with NodeMCU board directly but was powered separately as it requires 5(± 0.1) V and the NodeMCU board could not give more than 3.3 V.

The actual publisher system used is depicted in Fig. 7 NodeMCU board is subscribed to all the topics in which the whole network is publishing. This is done so that the Blynk app is configured to only one board, having the access of all the smaller Wi-Fi modules. Subscription means that the NodeMCU board will receive all the encrypted messages the modules publish in accordance with a unique/fixed topic. This decrypted, published data is then sent to the

Blynk app for user interest. If any suspicious activity is traced, then the Blynk app can notify the user who could take any action, therefore.

Given in Fig. 8 is a pictorial block diagram of the whole network. It includes all the CPUs, sensors and the way they are connected to the MQTT server as well as the Blynk server.

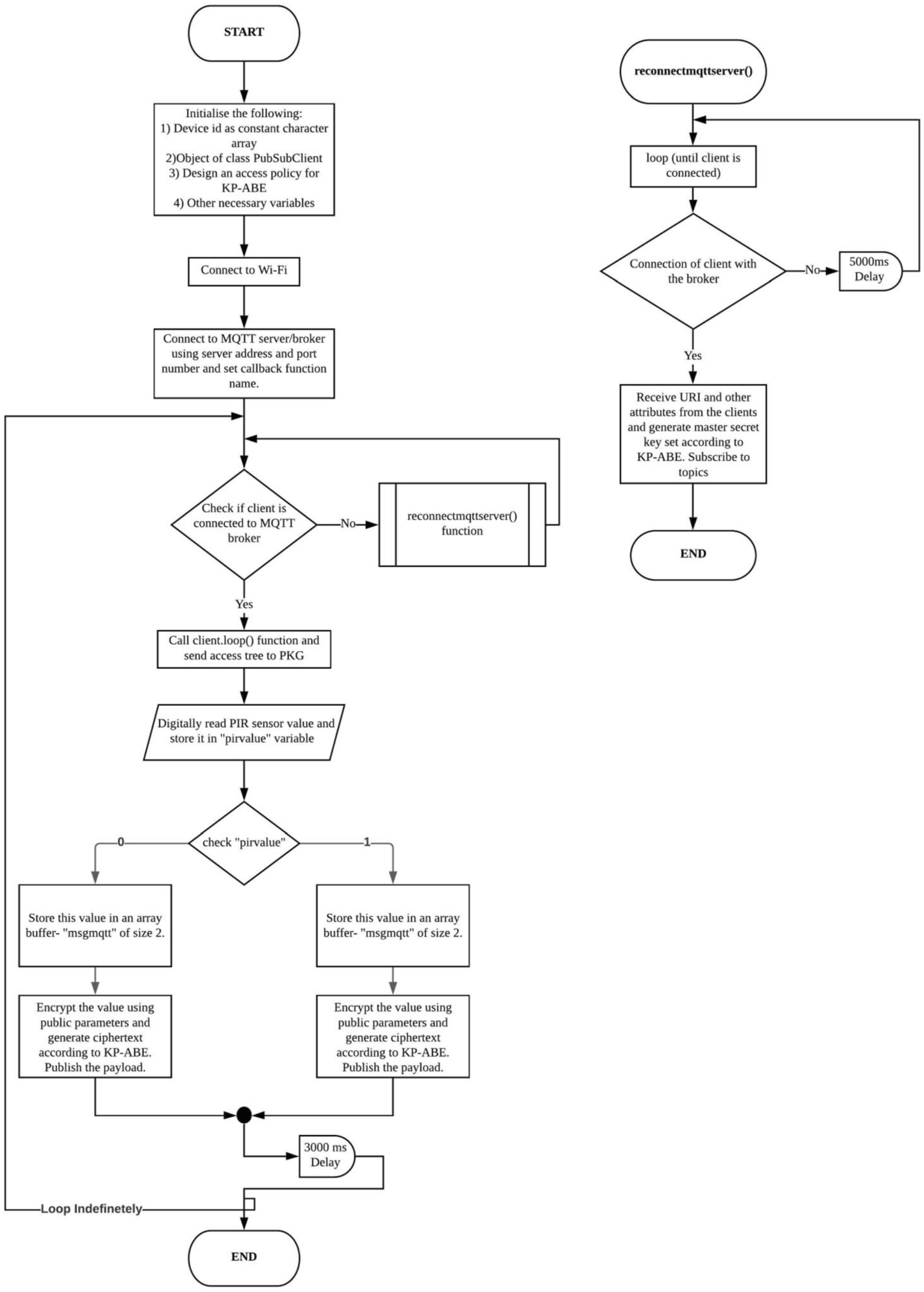

START
Initialise the following:
1) Device id as constant character array
2)Object of class PubSubClient
3) Design an access policy for KP-ABE
4) Other necessary variables
Connect to Wi-Fi
Connect to MQTT server/broker using server address and port number and set callback function name.
Check if client is connected to MQTT broker
No
reconnectmqttserver() function
Yes
Call client.loop() function and send access tree to PKG
Digitally read PIR sensor value and store it in "pirvalue" variable
check "pirvalue"
0
1
Store this value in an array buffer- "msgmqtt" of size 2.
Store this value in an array buffer- "msgmqtt" of size 2.
Encrypt the value using public parameters and generate ciphertext according to KP-ABE. Publish the payload.
Encrypt the value using public parameters and generate ciphertext according to KP-ABE. Publish the payload.
3000 ms Delay
Loop Indefinetely
END
reconnectmqttserver()
loop (until client is connected)
Connection of client with the broker
No
5000ms Delay
Yes
Receive URI and other attributes from the clients and generate master secret key set according to KP-ABE. Subscribe to topics
END

Fig. 10 ESP-01 8266 publishing sensor readings algorithm

a

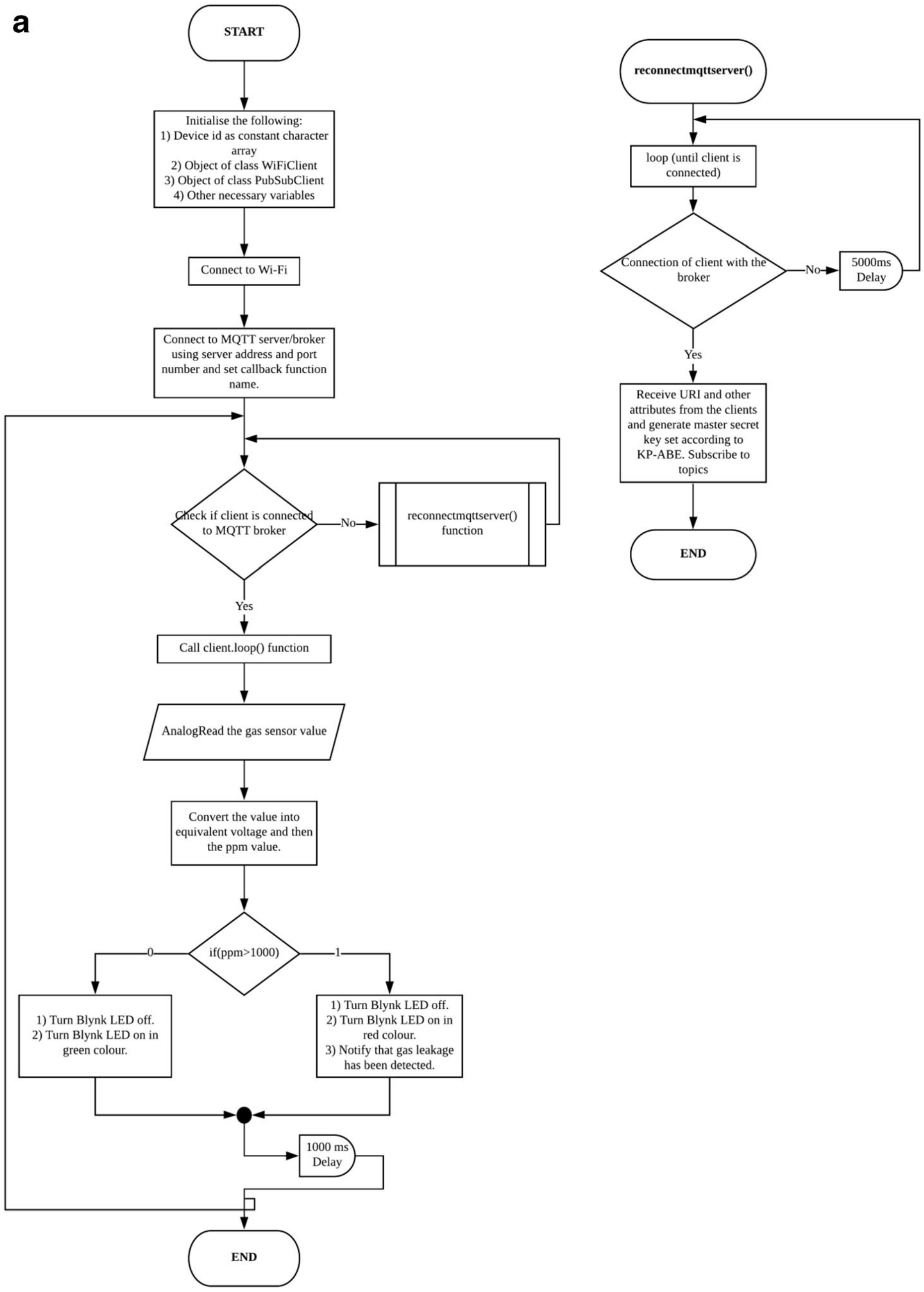

START
Initialise the following:
1) Device id as constant character array
2) Object of class WiFiClient
3) Object of class PubSubClient
4) Other necessary variables
Connect to Wi-Fi
Connect to MQTT server/broker using server address and port number and set callback function name.
Check if client is connected to MQTT broker
No
reconnectmqttserver() function
Yes
Call client.loop() function
AnalogRead the gas sensor value
Convert the value into equivalent voltage and then the ppm value.
if(ppm>1000)
0
1
1) Turn Blynk LED off.
2) Turn Blynk LED on in green colour.
1) Turn Blynk LED off.
2) Turn Blynk LED on in red colour.
3) Notify that gas leakage has been detected.
1000 ms Delay
END
reconnectmqttserver()
loop (until client is connected)
Connection of client with the broker
No
5000ms Delay
Yes
Receive URI and other attributes from the clients and generate master secret key set according to KP-ABE. Subscribe to topics
END

Fig. 11 NodeMCU subscribed to and receiving from the publishing topics algorithm

Figure 9 (a) shows the MQ-5 gas sensor module and NodeMCU ESP 8266 board connected circuit diagram.

Figure 9 (b) shows the PIR sensor module and ESP-01 8266 Wi-Fi module connected circuit diagram.

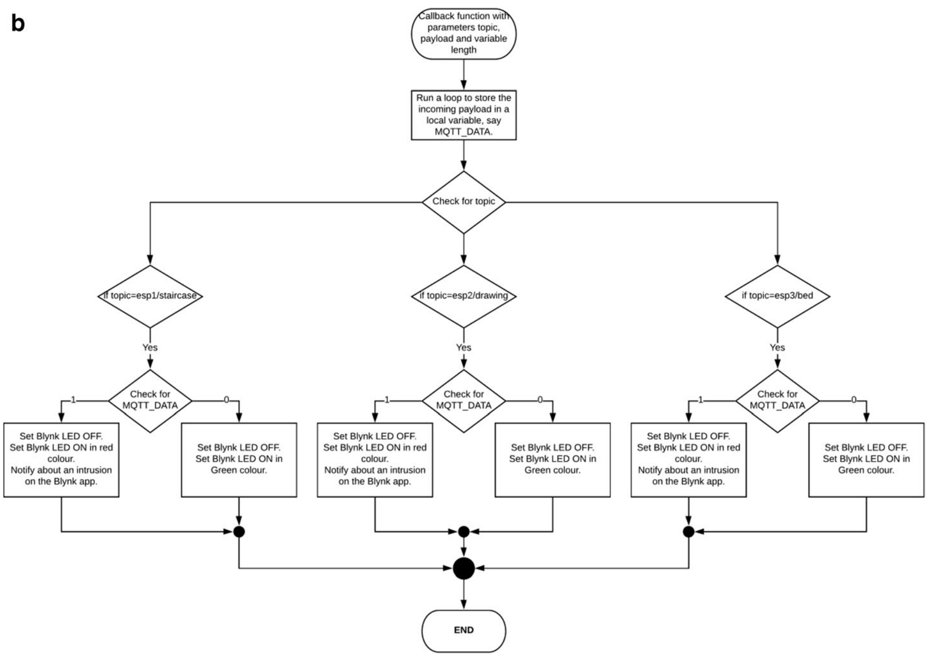

b
Callback function with parameters topic, payload and variable length
Run a loop to store the incoming payload in a local variable, say MQTT_DATA.
Check for topic
if topic=esp1/staircase
if topic=esp2/drawing
if topic=esp3/bed
Yes
Check for MQTT_DATA
1
0
Set Blynk LED OFF. Set Blynk LED ON in red colour. Notify about an intrusion on the Blynk app.
Set Blynk LED OFF. Set Blynk LED ON in Green colour.
END

Fig. 11 (continued.)

Given below are flowcharts regarding the functioning of both- the publishers and the subscriber.

The flowchart shown in Fig. 10 corresponds to the publisher- one which senses data and transmits them to the broker. The flow process is depicted and can be easily comprehended. The first flowchart depicts the full publisher side code. The adjoining flowchart depicts the reconnectmqttserver() function. There is no callback function in the flowchart primarily because it is not required in this only-publisher design.

The flowchart shown in Fig. 11 corresponds to the subscriber- one which gets messages from the broker. The flow process is depicted and can be easily comprehended. Figure 11 (a) depicts the full subscriber side code. The adjoining flowchart depicts the reconnectmqttserver() function. This reconnectmqttserver() function also contains the topics the device has to subscribe to. The next flowchart in Fig. 11 (b) depicts the callback function. Callback function handles the incoming messages and the further processes with the data are done there only as shown in the flowchart. The operations included in the callback function are different colored Blynk LED OFF/ON and alerts on the Blynk app.

4 Experimental results

The subscriber-publisher model used as shown in the system model was carrying out its job of sensing and notifying whenever it finds something unexpected. The following picture shows the full control screen.

As can be seen in Fig. 12, 2 widgets have been used: LED and notification. The green colored LED means that that part is safe and red colored LED means that there the sensor has found something reportable. The next 4 screenshots demonstrate the working of the alert system:

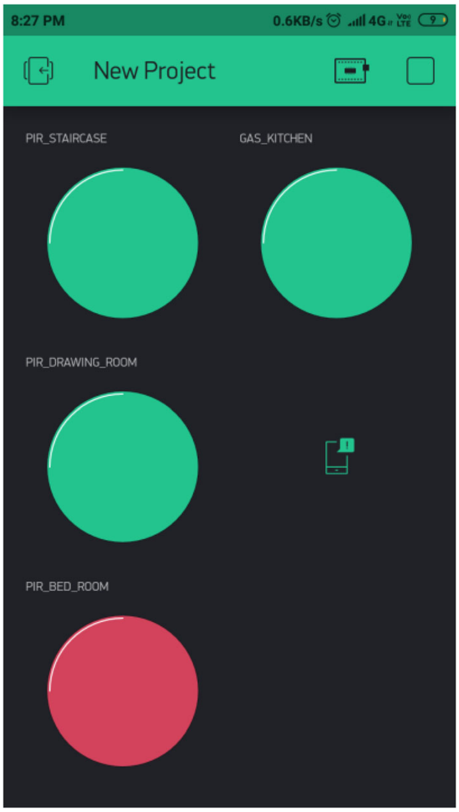
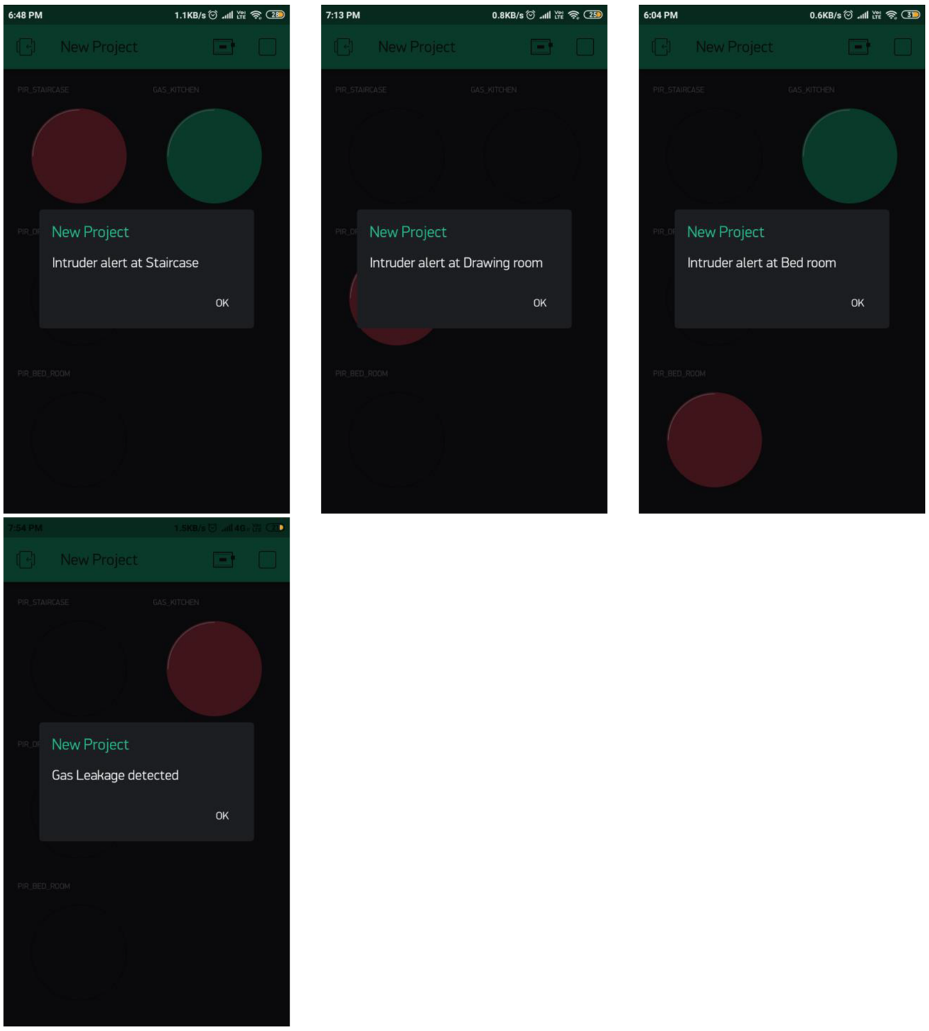


Fig. 12 Full system active in Blynk app

The above pictures in Fig. 13 demonstrate the working of Blynk app whenever the sensor tries to alert us something. This along with a message, alerts us through a background alert music which is fully customizable. The alert system also alerts us about when our device (NodeMCU board, behaving as virtual center) disconnects from the Blynk server. For checking the connection status of the sensor modules, one should see if their individual Blynk LED is blinking or not. If they are blinking then it means that the sensor module is working fine and if it is not blinking and showing any solid color, then it must be showing the sensor status, when it was last connected to the MQTT broker. As far as accuracy, speed and latency are concerned, one could expect very reliable, accurate sensor readings provided they are used in appropriate conditions. For speed and latency, as they are using MQTT protocol, and publishing very small messages, the latency one could get is of nearly half a second and that too when multiple sensor nodes are alerting at the same time.

5 Conclusions and future developments

This paper aims at the usage of MQTT protocol in developing a smart home security system incorporating KP-ABE encryption that reports its status in real-time. The 2 sensors used here reported any mishappening almost instantly. The focus was on reliable delivery of a message from publisher-broker-subscriber model and overcoming the security level drawback of plain MQTT i.e.



New Project
Intruder alert at Staircase

New Project
Intruder alert at Drawing room

New Project
Intruder alert at Bed room

New Project
Gas Leakage detected

Fig. 13 Blynk app showing the notifications/alerts

MQTT protocol functioning when used in a wireless sensors' network environment. The main characteristics of the MQTT protocol include low power, low overhead, low complexity, and asynchronous communication. The project that was finally created, stood firm on all the characteristics mentioned above and along with KP-ABE encryption and Blynk app, helped securing message transfers, gave amazing interface to check the status and also facilitated with an alert system. PIR sensors placed at different locations correctly notified about any bodily movement and the MQ-5 gas sensor alerted the user on exceeding certain concentration level. The user got a notification with a message along with an alarming ringtone. The delay in time between the actual sensing and its notification was one second every time on the first alert. This delay, however, depends upon the speed of the internet connection too. After the first alert, the other alerts were made almost instantly. Moreover, the Blynk app also notified the user when did the connection to the broker get broke. This work can be extended in future as, above all the PIR sensors, a camera module can be connected so that the user can also watch, about what the PIR sensor is trying to alert.

This can be achieved by using "Blynk bridge" facility, provided by the Blynk app, although this is a preliminary idea. Also, an exhaust fan and an alarm speaker can be connected, directly through the NodeMCU board for exceeded levels of gas concentration and all types of alerts as aimed in this project respectively.

References

1. Bhatt A (2013) Insight–learn the working of a motion sensor or PIR sensor. Engineers Garage. https://www.engineersgarage.com/insight/insight-learn-the-working-of-a-motion-sensor-or-pir-sensor/. Accessed 1 Jan 2020
2. No author (2015) MQTT Basics. MQTT Essentials. https://www.hivemq.com/mqtt-essentials/. Accessed 2 Jan 2020
3. No author (2019) Technology Trends Dossier. http://cmrindia.com/report/CMR-Technology-Trends-Dossier-2019.pdf. Accessed 30 Dec 2019
4. No author MQTT. Wikipedia. https://en.wikipedia.org/wiki/MQTT. Accessed 1 Jan 2020
5. No author. Understanding a Gas Sensor. Jaycon Systems. https://jayconsystems.com/blog/understanding-a-gas-sensor. Accessed 1 Jan 2020
6. Goyal V, Pandey O, Sahai A, Waters B (2006). Attribute-based encryption for fine-grained access control of encrypted data. In: proceedings of the 13th ACM conference on computer and communications security. ACM, pp 89-98
7. Grgić K, Špeh I, Heđi I (2016) A web-based IoT solution for monitoring data using MQTT protocol. In: 2016 international conference on smart systems and technologies (SST). IEEE, Osijek, pp 249–254
8. Jose A, Malekian R (2017) Improving smart home security: integrating logical sensing into smart home. IEEE Sensors J 17:4269–4286. https://doi.org/10.1109/jsen.2017.2705045
9. Kocakulak M, Butun I (2017) An overview of wireless sensor networks towards internet of things. In: 7th annual computing and communication workshop and conference (CCWC). IEEE, Las Vegas, pp 816–821
10. Krishna P, Ravi K, Kumar V, Sai Kumar M (2017) Implementation of MQTT protocol on low resourced embedded network. International Journal of Pure and Applied Mathematics 116:161–166
11. Kuo Y, Li C, Jhang J, Lin S (2018) Design of a Wireless Sensor Network-Based IoT platform for wide area and heterogeneous applications. IEEE Sensors J 18:5187–5197. https://doi.org/10.1109/jsen.2018.2832664
12. Mainetti L, Patrono L, Vilei A (2011) Evolution of wireless sensor networks towards the internet of things: a survey. In: SoftCOM 2011, 19th international conference on software. Telecommunications and Computer Networks. IEEE, Split, pp 16–21
13. Singh M, Rajan M, Shivraj V, Balamurlidhar P (2015) Secure MQTT for internet of things (IoT). In: 2015 fifth international conference on communication systems and network technologies. IEEE, Gwalior, pp 746–751
14. Rouse M, Gillis A, Waher P. MQTT (MQ Telemetry Transport). IoT Agenda. https://internetofthingsagenda.techtarget.com/definition/MQTT-MQ-Telemetry-Transport. Accessed 2 Jan 2020
15. Wang X, Zhang J, Schooler E, Ion M (2014) Performance evaluation of attribute-based encryption: toward data privacy in the IoT. In: 2014 IEEE international conference on communications (ICC). IEEE, Sydney, NSW, pp 725–730

Affiliations

Vatsal Gupta[1] • **Sonam Khera**[2] • **Neelam Turk**[2]

[1] Department of Electronics and Communication Engineering, Maharaja Agrasen Institute of Technology, Delhi, India

[2] Department of Electronics Engineering, J.C. Bose University of Science and Technology, YMCA, Faridabad, Haryana, India