Smart Nursing Home Patient Monitoring System

Karan Motwani¹, Divesh Mirchandani², Yogeeta Rohra³, Heena Tarachandani⁴ & Prof. Anjali Yeole⁵

¹,²,³,⁴ (B.E.) CMPN, VESIT
⁵ Assistant Professor, CMPN, VESIT

Abstract---Data such as blood pressure, respiratory level, blood glucose level, pulse rate and other medically related patient data can be collected by sensors on peripheral devices. Examples of such devices are glucometer, blood pressure cuff. The data can be sent wirelessly to doctors, nurses and other forms of caregivers. Potential problems can be detected immediately as the concerned caregivers are immediately alerted. As a result, timely intervention ensures that the patient does not have to suffer because of any delay in time. Future implementations may involve the use of a door sensor or a light sensor to monitor the movements of the nursing home patients.

Keywords---broker; care-giver; database; Internet of things; Message queue telemetry protocol; patient; publisher; sensors; subscriber

I. INTRODUCTION

It is widely known that the world population is aging at a rapid rate. In the year 2000, there were 60 million people aged 60+ globally. That age group is expected to double by the year 2025, reaching 1.5 billion. In United States itself, the baby boomer generation is aging with approximately 7000 people turning 65 every day.

Medical staff and healthcare funding find it increasingly difficult to keep up with the rapid growth of the aging population, as a result of which, the patient-to-caregiver ratio increases. Nurses are usually responsible for providing medication, helping doctors with their work, changing dressings, and keeping a tab on multiple patients. Due to this, there is a high likelihood that the quality of patient-care may suffer due to “alarm fatigue”! There are so many dozen types of alarms that are likely to trigger several times a day, leading to nurses becoming desensitized to distress calls and not thus, not reacting with enough urgency when a patient emergency actually occurs.

However, by using our wireless patient monitoring system, nurses, doctors and caregivers can be wirelessly connected to the patients whenever attention is required. The important thing to note is that the healthcare professionals can be alerted to emergencies within seconds and can prevent a problem from escalating. Simply put, the ability to communicate or forward messages in a wireless and secure manner, and at the same time allowing doctors, nurses and caregivers to cater to the needs of multiple patients without having to physically walk to different rooms of patients, increases efficiency and more importantly, helps in reducing the overall workload of the caregiver to a great extent.

II. LITERATURE SURVEY

A. Internet of things

One of the rising fields to watch out for in Information Technology is Internet of Things (IoT). Internet of Things is the future, and is very much here to stay. It is widely acknowledged that IoT has the power to create intelligent virtual objects from the existing real world objects. The main aim of IoT is to gather everything in our world under a common infrastructure and to unify it, giving us the power to not only acquire control of things in our environment, but also to keep us informed of the state of the things. Keeping this in mind, present study focuses on addressing IoT concepts through a thorough review of scholarly research papers, professional discussions with experts and online databases. This research article focuses mainly on the definitions, basic requirements, characteristics and different forms of Internet of Things. The main aim of this paper is to provide an overall idea of what Internet of Things is, the different architectures it can adopt, and how it can provide a solution for the problems faced by the global healthcare industry.

B. Message Queue Telemetry Protocol

MQTT, was developed by researchers at IBM. It is a lightweight broker-based protocol that follows
the publish-subscribe model of messaging and has been designed to be open, simple, lightweight, and above all, easy to implement. The characteristics of MQTT make it ideal for use in constrained devices, where, for instance, the network is very expensive, has low bandwidth or is simply not reliable, or when it is meant to run on embedded devices that have limited processing or memory capacities. MQTT is an asynchronous protocol. MQTT does not comply with REST. MQTT messages usually contain a variable header, that is present before the payload, and after the fixed header, which includes, for instance, the protocol name along with the protocol version, and the appropriate flags. There are 14 different message types specified in MQTT, such as ‘CONNECT’, ‘PUBLISH’, ‘SUBSCRIBE’, ‘UNSUBSCRIBE’, ‘DISCONNECT’, ‘PINGREQ’ and ‘PINGRESP’. Many of these messages are paired with dedicated acknowledgement messages. For instance, if there is a subscriber who wants to subscribe to a particular topic, the subscriber will have to send a ‘SUBSCRIBE’ message, and then wait for the corresponding ‘SUBACK’ response message from the broker (server). MQTT supports three types of Quality of Services (QoS) for the publish messages delivery:

• QoS Level 2(Exactly once) – In this type of QoS level, the message is guaranteed to reach the receiver only once. This QoS level is the most secure but also the slowest level.

• QoS Level 1(At least once) – The message will definitely reach the intended receiver at some point of time. However, this may result in duplication of the same message.

• QoS Level 0(At most once) – This QoS level guarantees a best effort delivery. In this type of QoS, the publisher has no way of knowing if the message has been sent successfully, as he won’t receive any sort of acknowledgement from the receiver and so, the message will never be re-delivered by the sender. This type of QoS is also called as “fire and forget”.

C. Arduino Hardware

Arduino is an open-source prototyping platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing.

Over the years Arduino has been the brain of thousands of projects, from everyday objects to complex scientific instruments. A worldwide community of makers - students, programmers, and professionals - has gathered around this open-source platform, their contributions have added up to an incredible amount of accessible knowledge that can be of great help to novices and experts alike. All Arduino boards are completely open-source, empowering users to build them independently and eventually adapt them to their particular needs. The software, too, is open-source, and it is growing through the contributions of users worldwide.

D. Processing Software

Processing is a programming language, development environment, and online community. Since 2001, Processing has promoted software literacy within the visual arts and visual literacy within technology. Initially created to serve as a software sketchbook and to teach computer programming fundamentals within a visual context, Processing evolved into a development tool for professionals. Today, there are tens of thousands of students, artists, designers, researchers, and hobbyists who use Processing for learning, prototyping, and production.

- Interactive programs with 2D, 3D or PDF output
- OpenGL integration for accelerated 3D
- For GNU/Linux, Mac OS X, and Windows
- Over 100 libraries extend the core software

The Processing 2.0 release focuses on faster graphics, new infrastructure for working with data, and enhanced video playback and capture. It also expands the potential of the programming environment. The new Modes feature allows other programming systems, such as JavaScript and Android, to be easily used from within the development environment.

The PC application first reads 600 consecutive samples sent by Arduino. Since the sampling rate was 5ms, it takes 3 sec to read the 6000 samples. The DC component (minima of 600 samples) is subtracted out from the samples. Next, the range of the samples is computed. If the range is less than 50 counts, the received PPG waveform from the PPG signal. The resulting samples are plotted against time to obtain a clean and smooth PPG waveform. Note that we lose 10 samples at the
beginning and 10 samples at the end while applying the moving average filter.

![Fig-1 Counting Pulse in BPM](image1)

**E: Arduino Ethernet Shield**

The Arduino Ethernet Shield 2 allows an Arduino Board to connect to the Internet. It is based on the Wiznet W5500 Ethernet chip. The Wiznet W5500 provides a network (IP) stack capable of both TCP and UDP. It supports up to eight simultaneous socket connections. Use the Ethernet library to write sketches that connect to the Internet using the Shield. The Ethernet Shield 2 connects to an Arduino Board using long wire-wrap headers extending through the Shield. This keeps the pin layout intact and allows another Shield to be stacked on top of it.

![Fig. 3-Ethernet Shield](image2)

### III. TECHNOLOGY USED

- **Java**: Java is used in a wide variety of computing platforms from embedded devices and mobile phones on the low end, to enterprise servers and supercomputers on the high end.

- **Eclipse Paho**: Eclipse Paho is an IDE that consists of MQTT client runtimes that are used to implement the MQTT messaging protocol. The client runtimes are to be configured to run against a broker running on your server.

- **Mosquitto Broker**: The Mosquitto broker is a lightweight and function MQTT broker that can run on relatively constrained systems, but still be powerful enough for a wide range of applications.

- **MQTT Lens**: MQTT Lens is a Chrome Extension available to use along with Chrome Browser for the purpose of connecting to a MQTT Broker and test with publish/subscribe scenarios of MQTT Messages. This is a very useful tool to check the connectivity to the MQTT broker and to check various scenarios of publishing and subscribing messages.

### IV. BLOCKDIAGRAM

![Fig. 4-System Block Diagram](image3)
V. ARCHITECTURE

The architecture of MQTT-SN is shown in Fig. 2. There are three MQTT-SN elements, (i) MQTT-SN clients, (ii) MQTT-SN gateways (GW), and (iii) MQTT-SN forwarders. MQTT-SN clients establish connection to a MQTT server with the help of a MQTT-SN GW via the MQTT-SN protocol. The MQTT server may be or may not be integrated with the MQTT-SN GW. In the case of a stand-alone GW, the MQTT protocol is to be used between the MQTT-SN GW and the MQTT server. Its main function is the translation between MQTT and MQTT-SN. MQTT-SN clients can also access a GW using a forwarder if the GW is not directly attached to their network. The forwarder then encapsulates the MQTT-SN frames that it receives on the wireless side and finally forwards them without any changes to the GW. Conversely, in the opposite direction, it proceeds to decapsulate the frames that it receives from the gateway and then finally sends them to the clients, again without any changes. Based on the performance of a GW with respect to the protocol translation between MQTT-SN and MQTT, GWs are classified into the following two types: transparent and aggregating GWs.

VI. WORKING OF ECLIPSEPAHO

A. Initial window

B. Establishing connection

C. Subscribing to a topic

D. At the time of publishing, publisher will first select the topic, write the message and then publish it.

E. History
F. Last Message

<table>
<thead>
<tr>
<th>Topic</th>
<th>Message</th>
<th>QoS</th>
<th>Retained</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Movies/Mobific</td>
<td>Fake fiction was very...</td>
<td>0</td>
<td>No</td>
<td>2013-12-23 15:04:32</td>
</tr>
</tbody>
</table>

Fig. 11 - Last Message

VII. PROPOSED SYSTEM MODEL

Fig. 12 - Integration of MQTT Protocol in Nursing Scenario

VIII. ARDUINO WITH MQTT

The MQTT support in Arduino is provided through a library that is an implementation of the MQTT protocol. This library should be included in the Arduino code and then appropriate functions are called for CONNECT, PUBLISH, SUBSCRIBE and so forth. The code for CONNECT is added in the setup() function because the connection is made only once. The SUBSCRIBE code is added in the setup() function, and the message that is arrived is handled separately in a callback() function. The PUBLISH code is added in the loop() function. The network support for Arduino board is provided using a shield, such as the Arduino Ethernet shield, which is connected to a modem or a network port using a standard network cable. This shield sits on top of the Arduino board or another board.

Thus, using MQTT protocol, the patient’s vital sign measurements can be collected and delivered at multiple nursing stations. In order to allow doctors to access the collected data remotely, a mobile application can be developed to connect to the MQTT broker to subscribe to messages that have the topics of interest. Future implementations may involve deploying a light sensor and a door sensor to monitor the activity level of the patients and potentially identify the ones suffering from depression assuming that the patients have private rooms.

IX. CONCLUSION

X. REFERENCES


