An IOT Based Real-Time DHT11 Humidity Temperature Monitor with NodeMCU on AWS Cloud Service

Abstract—Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the Amazon Web Services (AWS) Cloud. Using Amazon EC2 eliminates your need to invest in hardware up front, so you can develop and deploy applications faster. You can use Amazon EC2 to launch as many or as few virtual servers as you need, configure security and networking, and manage storage. Amazon EC2 enables you to scale up or down to handle changes in requirements or spikes in popularity, reducing your need to forecast traffic. Internet of Things (IoT) is expected to play a major role in our lives through pervasive systems of sensor networks encompassing our environment. These systems are designed to monitor vital physical phenomena generating data which can be transmitted and saved at cloud from where this information can be accessed through applications and further actions can be taken. This paper presents the implementation and results of an environmental monitoring system which employs sensorsfor temperature and humidity of the surrounding area. Thisdata can be used to trigger short term actions such as remotely controlling heating or cooling devices or long term statistics,he sensed data is uploaded to cloud storage and an Android application accesses the cloud and presents the results to the end users. The system employs Arduino UNO board, DHT11sensor, ESP8266 Wi-Fi module, which transmits data to AWS ec2 server where it is analyzed and stored.An Android application is developed which accesses the cloudand displays results for end users via REST API Web service.The experimental results show the usefulness of the system.

INTRODUCTION

Internet of Things (IoT) is expected to revolutionize our world by enabling us to monitor and control vital phenomenain our environment through the use of devices capable of sensing, processing and wirelessly transmitting data to remote storage like cloud which stores, analyzes and presents this data in useful form. From the cloud this information can be accessed through various front end user interfaces such as web or mobile applications, depending upon suitability and requirements. Internet lies at the heart of this transformation playing its role in efficient, reliable and swift communication of data from devices to the cloud and from the cloud to the end users. In this new paradigm, the concept of the typical end system or host in the Internet is modified and hostscomprise of devices or things hence the name Internet of Things. The "things" are capable of sensing and transmitting data such as temperature, pressure, humidity, noise,pollution, object detection, patient vitals etc.

Environmental monitoring is an important IoT application which involves monitoring the surrounding environment and reporting this data for effective short term measures such as remotely controlling the heating or cooling devices and long term data analyses and measures. This paper presents the implementation details and results of an environmental monitoring system. The system comprises of a central Arduino UNO board which interfaces at the input with temperature and humidity monitoring sensor DHT11 and at the output with ESP8266 Wi-Fi module which transmits the sensed data through Internet to a remote cloud AWS ec2 server created.through these server the readings are displayed.

HARDWARE DESIGN

NodeMCU: NodeMCU is an open source IoT platform. it includes firmware which runs on the ESP8266 Wi-Fi SoC from Expressive Systems, and hardware which is based onthe ESP-12 module. The term "NodeMCU" by default refers to the firm ware rather than the dev kits. The firmware uses the Lua scripting language.

Fig:1 NodeMCU

The programming code is being written for ESP8266 Wi-Fi chip using Arduino IDE, for which installation of ESP8266 library is required. We designed to make working with this chip super easy and a lot of fun. We took acertified module with an on board antenna, and plenty of pins, and soldered it onto our designed breakout PCBs. While
this chip has been very popular, it's also been very difficult to use. Most of the low-cost modules.

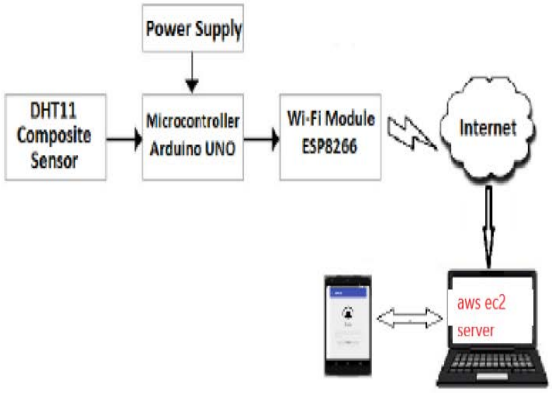
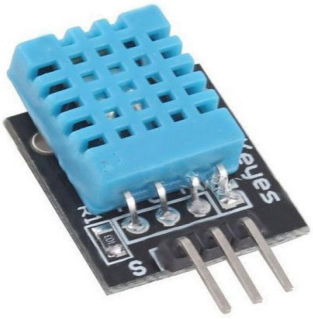


Fig. 4. Hardware block diagram for the environmental monitoring system

Fig:2 Block Diagram of DHT11 Humidity Temperature Monitor with NodeMCU

Temperature and humidity sensor

We selected temperature and humidity for environmentalmonitoring and prefered a single sensor with both sensingcapabilities instead of separate sensors for each parameter.For this reason, we selected DHT11 composite sensor chip which gives readings for both temperature and humidity atthe same time, it has high reliability and excellent long-term stability. It has small dimensions, low cost, good quality, fast response, strong anti-interference ability,digital signal output, and precise calibration. It can be easily interfaced with Arduino UNO board with the help of DHT library and connecting wires. Figure 2 shows a picture of the DHT composite sensor which we used in our system. It has temperature range from 0 to 50°C and humidity range from 20 to 90%RH.



It has signal transmission range of 20m. To interface it with Arduino UNO, we connected the Ground and Vcc of the DHT11 sensor with the Ground and 5V of the Arduino. Then we connect the Data pin of the DHT11 to the pin 2 of the Arduino. Then we installed the DHT library and run the code for getting it started.

Protocols Used

1) MQTT:

MQTT is an open source protocol that follows publish -subscribe mechanism for many to many distributions as shown in Fig. 3. It is lightweight and bandwidth efficient as it uses a small header and it also has continuous session awareness. It is designed to minimize use of device resources and ensure reliability and varying degrees of service delivery assurance

Using MQTT, the clients do not communicate directly with the endpoint; publish - subscribe mechanism decouples the client (publisher) who is sending the message and another client (subscriber) who is receiving the message.

The MQTT broker connects the publishers and subscribers. It receives all incoming messages from the publishers, filters them and redistributes them accordingly such that only specific clients receive specific messages.

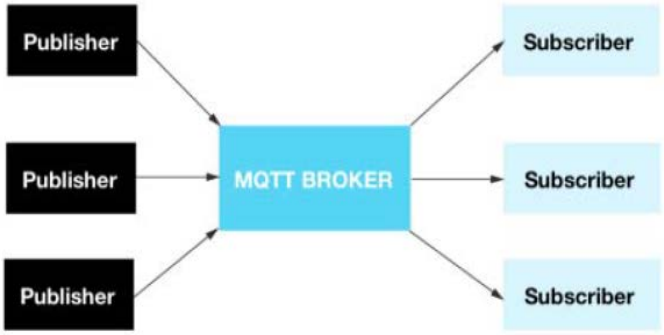


Fig 3. MQTT architecture

Flexibility and speed is achieved through space, time and synchronization decoupling of publisher and subscriber. The publish/subscribe model enhances scalability and processes on the broker are run parallel and are initiated by specific events

MQTT provides three values of QoS for each message
sent. The QoS determines how the messages are sent and hence ensures reliability of the system [8].

- QoS 0 – With a QoS value of 0, there is no guarantee that the message reaches the destination as there are no return acknowledgement messages.

This is also the fastest way to deliver a message.

- QoS 1- A QoS value of 1 ensures that the message is delivered at least once and it requires the reception of one pair of acknowledgement message.

It contains a header and so is less lightweight.

- QoS 2 – Here actual messages are sent exactly once.

Two pairs of acknowledgement messages between the publisher and broker are transferred as a handshake to ensure that the message has been sentexactly once. This is the slowest and also the mostreliable. Different QoS levels are suited to applications which have varying needs with respect to speed and type of messages sent

2) Websocket:

Websocket is a low latency, bi-directional messaging protocol that allows persistent connection over a single TCP/IP connection [8]. Unlike HTTP, messages can be continuously

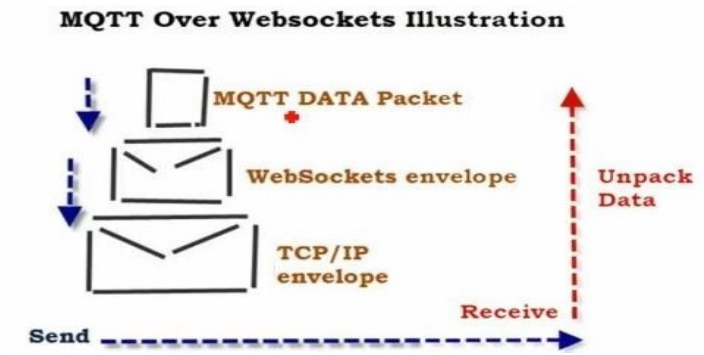


Fig 4. MQTT over Websocket illustration

sent once the connection has been made until the connection is deliberately broken by one side.Websocket was designed to be used in web applications that need real time persistentconnections, such as live chat,video conferencing, etc.

3) MQTT over websocket:

AWS supports MQTT communication using the Transport Layer security (TLS) cryptography protocol and Websocket connection using the Socket Security Layer(SSL) cryptography protocol.

ESP8266 doesn't support TLS as of yet, so the MQTT messages need to be packaged in a Websocket protocol as shown in Fig. 4 for communication with AWS. The MQTT protocol enables message delivery with low bandwidth and latency while the Websocket layer provides the SSL security to messages. The MQTT protocol can be replaced with a more lightweight protocol without any change to the system and so the system can be upgraded to newer protocols very easily.

Communication mechanism

1) Connection with AWS:

The lower level protocol change for the connection from client to broker is shown in Fig. AWS connects MQTT over Websocket protocol -the application layer protocol, on port TCP 443 using SigV4 authentication. Our client end device has been configured to make connections to this port. For the initial connection setup, the client sends

a CONNECT command message whose conter consists of client identification and authentication information as shown in Fig. 7 also specifies other information to make the connection reliable such as the time for connection timeout. The client side receives an acknowledgement message “CONNACK” from broker containing the connection status information. Data packets are sent from client end only when “CONNACK” contains the desired information for successful connection.

2) Maintaining the QoS:

Since it does not harm to lose a few temperature or humidity data, QoS level 0 is used to keep overhead and bandwidth minimum. In QoS 0 as stated before only the “PUBLISH” data packet is sent without any return acknowledgement messages from the broker.

3) Keep Alive:

MQTT contains a keep alive feature which keeps the connection open even when data packet is not sent. When no data packets are sent within a timeout interval, the PINGREQ packet is sent to server and PINGRESP packet received as acknowledgement to ensure that the connection is still open. For disconnection a DISCONNECT packet is sent or for abrupt disconnection the PINGRESP is not received. This feature enables connection to be maintained even when there is delay but close it when client becomes unresponsive to prevent malignant connections.

EXPERIMENTAL RESULTS

The complete design of our environmental monitoring system is shown in Figure.The implemented design is shown in Figure which shows the integration of all hardware components in working conditions. DHT11 and ESP8266.

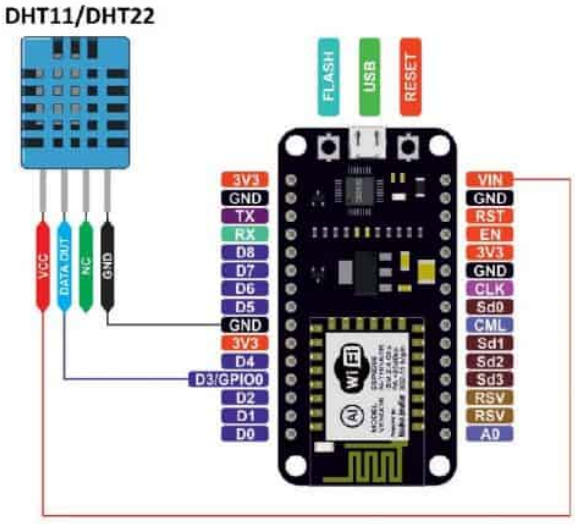


TABLE I. SYSTEM COMPONENT DETAILS

System Component	Details
Sensor	DHT11
Connectivity	Wi-Fi ESP8266
User Interface	Android Application
Cloud	AWS-EC2-SERVER

This Section presents the experimental results of our system. We show the sensor data available to the mobile user through the mobile application and the graphical record oftemperature and humidity monitoring.

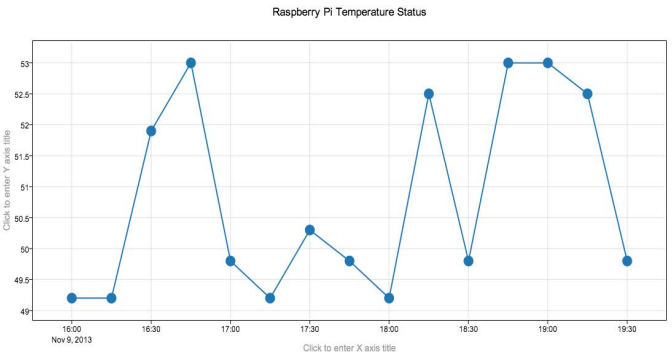
A. Application User Interface

After the authentication phase in which the user enters login and password, the user logs in to the application to view the monitored data. Figure 7 shows the GUI of the Android application which shows the data logging that is a view of the temperature and humidity record

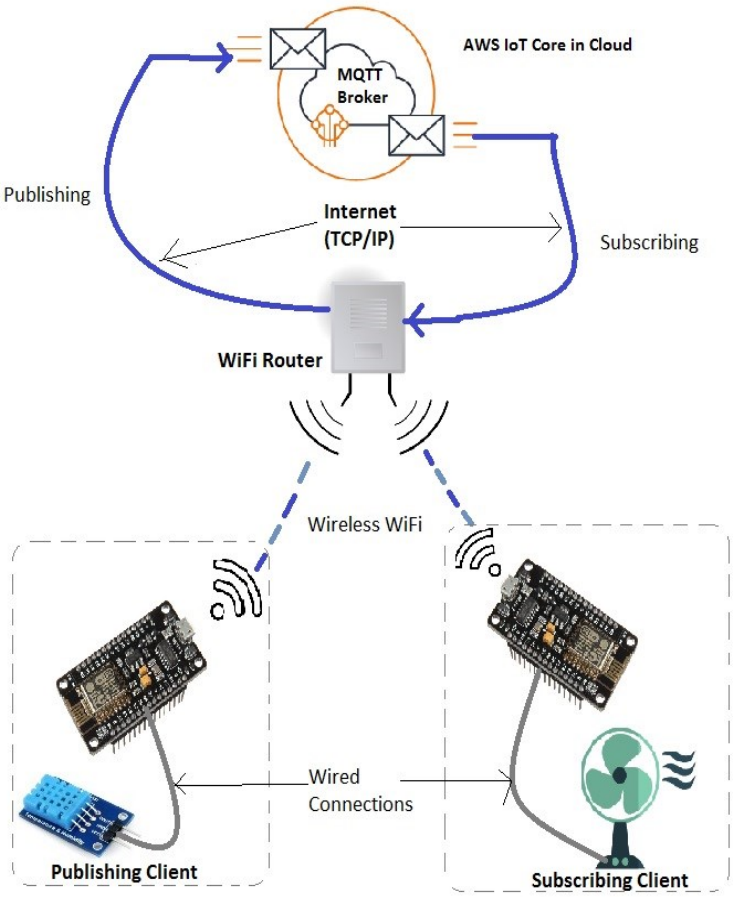
Time, hr	Temperature, K	Humidity
310	378	0.4
316	378	0.4
329	378	0.4
411	378	0.4
190	378	0.8
208	378	0.8
230	378	0.8
298	378	0.8
108	398	0.4
123	398	0.4
166	398	0.4
200	398	0.4

B. Graphical Record of Temperature And Humidity Monitoring

Fig shows the record of temperature monitoring over a period of time. The graph is temperature vs. time where the temperature changes are updated after an interval of 15 seconds. In order to accurately test our system, we varied the temperature around the sensor artificially by a lighting system, thus a spike can be viewed in the graph after which the temperature readings settle to average environmental temperature.



shows the record of humidity monitoring over a period of time. The graph is humidity vs. time where the changes in humidity are updated after an interval of 15seconds. Similar to temperature monitoring, we varied the temperature around sensor artificially by a lighting system,the downward spike in humidity can be viewed in the graph after which the humidity readings settle to average environmental humidity.



This system is a crucial step in understanding the IoT applications development and implementation and serves as a building block for a number of useful innovations in this direction.

CONCLUSION

This project presents an environmental monitoring system for real-time monitoring of temperature and humidity of surrounding environment. The sensed data is sent through Wi-Fi to the cloud where both real-time data and its graphical analyses can be viewed.An user can turn on or turn off his fan or Ac.

This system can be extended to implement a home automation system where the monitored values of temperature and humidity can be used to trigger some action and control the devices for heating or cooling via the mobile application.