IJCTA, 9(23), 2016, pp. 135-143
© International Science Press

Design and Testing Considerations for an IoT Project

Pauravi Wagh* Chandrika Kapre and Priya Singh*****

Abstract : With the increasing adaptation of IoT applications, there is a plethora of platforms, tools brought in the market for development of such applications. They vary in terms of hardware platforms, software languages, supporting tools, communication protocols, hosting solutions, device features, etc. Design considerations of traditional application development do not apply to IoT applications. With respect to this, we will look at the new IoT specific design considerations. Each of these design considerations, have varying features and have an impact on efficiency, accuracy, cost effectiveness, etc. We will use an example of an IoT application, Smart Water Mixer to see how each design consideration impact an IoT project.

Keywords : IoT, Cloud Computing, Water, Mixer.

1. INTRODUCTION

Ubiquitous Connectivity, miniaturization, cloud computing and the power of Internet has made connecting physical objects and devices with the Internet and building a range of applications from across the domains a trend today. Large IoT projects are being undertaken at commercial level. While making "Things" talk to each other seems easy, we have to carefully decide upon the various design considerations which will make applications efficient and cost effective. A strategic approach is needed while considering the factors like Hardware, Cost, Communication, Storage, Servers, etc. In this paper we talk about the design considerations for an IoT application with the help of an IoT device, 'A Smart Water Mixer'.

2. PROPOSED SYSTEM

The "Smart Water Mixer" is an IoT application that stores the desired water temperature level for multiple users and dispenses water at preferred temperature for its current user. The device identifies current user using biometrics and fetches the preferred temperature for the user from cloud. It then mixes hot and cold water in right proportion to get the desired water temperature in the shower. Suppose the first time user gets water at 32 degree C. Whenever user manually changes the temperature, the output temperature is updated in cloud as the desired temperature for the user. Next time the water is automatically dispensed at the preferred temperature for that user. Multiple such devices can cater to the same user.For example, if a family has three bathrooms equipped with the "Smart Water Mixer" in their house, then irrespective of which bathroom a user decides to use, he/she will always get water at the desired temperature.

This avoids wastage of water that goes in adjusting the shower every time. It also saves time, energy and is safe. The solution also keeps track of total amount of water used by each user and provides usage pattern graph, giving feedback to users to optimize the water consumption.

* Persistent University, Persistent Systems Ltd *pauravi_wagh@persistent.com.*
** Persistent University, Persistent Systems Ltd chandrika_kapre@persistent.com
*** Persistent University, Persistent Systems Ltd priya_singh@persistent.com

Working

The user authenticates herself with the help of the biometric scanner which then connects to cloud to get user preference. The device will continuously monitor the flow and temperature of water and make necessary changes in the cold and hot water valves to give precise temperature of water to the end user. The proximity sensor is used to detect the person in the bath area. If the person moves out of the range of the proximity sensor, the valves are shut off and water flow stops thus preventing the wastage.

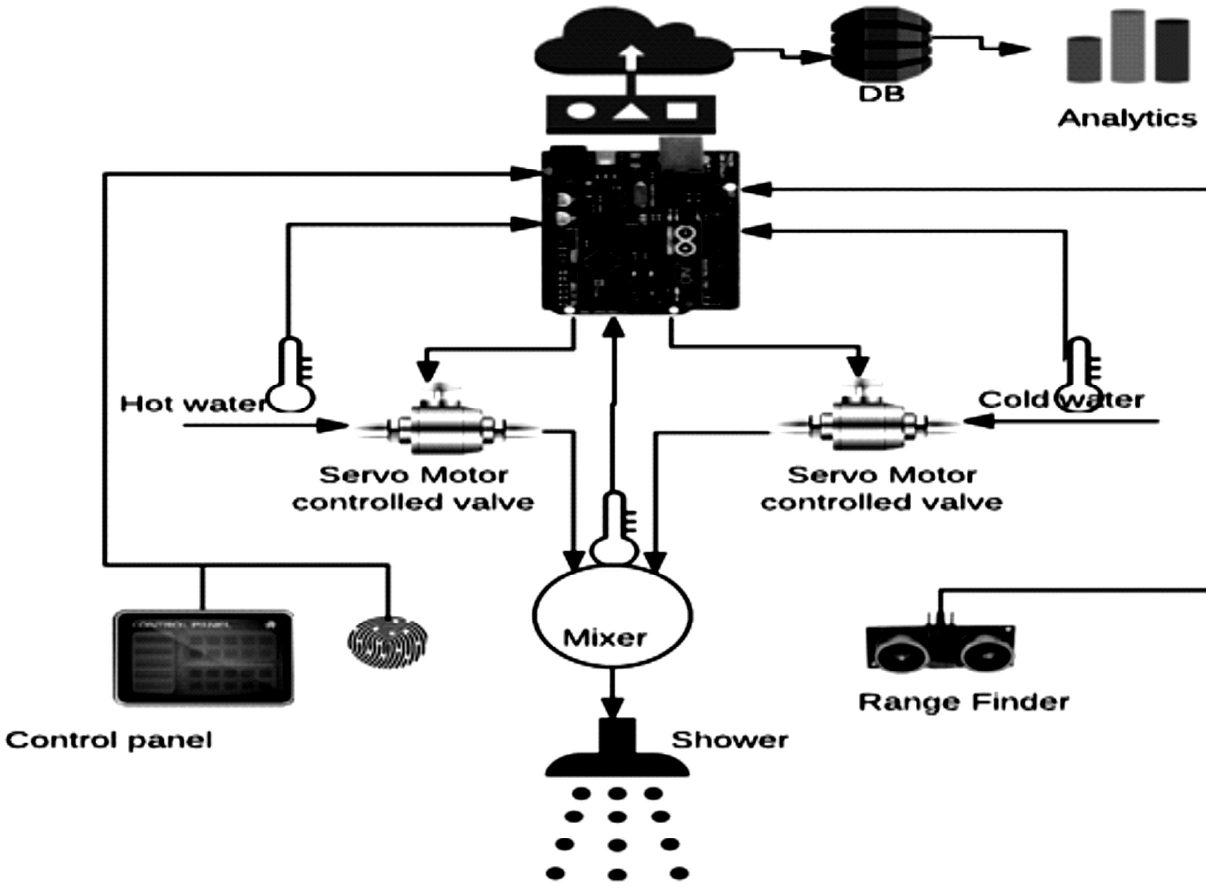

Fig. 1. System Architecture.

In this application there are 3 basic units. And the entire system is built from scratch.

- **Water Mixer Unit – Integrated with Shower :**
  - (*a*) Detects user and automatically starts shower with preferred temp (without wasting water to achieve the desired temperature level)
  - (*b*) Continuously records current water temperature and consumption data
  - (*c*) Automatically stops when user goes away and starts when user comes near the shower
- **Display Panel (Dashboard) near Shower :**
  - (*a*) A screen to display some data like Information of user, his preferred temperature, current temperature, water flow rate, water consumption(from start of the shower till end for the current session).
- **Water Mixer Web Server/Mobile App :**
  - (*a*) Stores data for each user/home for each shower session
  - (*b*) Shows various charts and graphs to display historical data for better analysis and predictions like per user/home/area water consumption in a month, average water temperature preference for users in a particular area.

2. DESIGN CONSIDERATIONS

3.1. Components

The IoT device performs actuation and gathers the relevant data. With the advancement in digital electronics, embedded systems, etc. manufacturing small, cheap and effective devices or "Things" has become simple. From a bunch of these available options we need to carefully analyze the requirements and decide which components to be included in the system. The basic components of IoT applications are the sensors, computational platforms, communication devices, servers.

3.1.1. Sensors

Sensors are the actuators which sense the data and produce related output. The various factors to be considered while choosing the sensors are:

3.2. Range of the sensors

3.3. Accuracy

3.4. Area to be covered

3.5. Cost

Example : The sensors used in the smart water mixer have been mentioned below

Table 1.

	Proximity sensor	*Temperature sensor*	*Flow Sensor*
Specifications	**Detection distance:**	**Temperature range:**	**Working Flow Rate:**
	2cm-450cm.	55°C to + 125°C	1 to 30 Liters/Minute Working
	High precision:	**Accuracy** ± 0.5°C	**Temperature range:** –25 to +80°C
	Up to 0.3 cm	**Waterproof**	**Working Humidity Range:** 35%–80% RH

3.1.2. Computational Platforms

Device management is the primary goal of a computational platform. These platforms must handle configuration of various types of devices, track the operation of the other devices connected to it, firmware upgrades and device level error handling. These boards act as interfaces between the hardware and the software components.

Majorly two types of boards are used 1) Raspberry Pi 2) Arduino

- **Raspberry Pi :** (Micro controller) Raspberry Pi is a full-fledged sized computer with 512 MB RAM and 700 MHz microprocessor. It is able to run a full Linux based operating system.
- **Arduino Board:** (Micro Controller) Arduino is a collection of 3 things, a hardware prototype platform, Arduino language and IDE & libraries. The board is based on 8-bit microcontroller. It has built-in hardware support for SPI, I2C and Serial.

Comparison :

Table 2.

Feature	*Raspberry Pi*	*Arduino*
Processor Speed	700MHz	16MHz
Programming Language	No Limit	Arduino C/C++
Real-time Hardware	No real-time	In real time
Hardware Design	Closed source	Open source
Internet Connectivity	Easy	Bit difficult

Decision Made : We need continuous reception and transmission of the sensor data. Whenever the user starts using the data, the device should continuously sense the data from various sensors and transmit it to the cloud. Data processing is done at the backend. We do not need very high processor speed. Coding can be easily done in Arduino C/C++. Thus we chose Arduino board. Arduino needs very less power to work. The sensors are connected to the board. The code in the Arduino IDE is burnt on the Arduino Board and the Wi-Fi shield attached to the board is used for sending the data to the cloud.

3.1.3. Communication technologies

IoT data typically is in the form of small packets which contact state of IoT devices. Many IoT applications typically send such small packets on regular intervals for 24/7.

Data can be sent to the backend by a variety of ways. The most prominent ones are Bluetooth, Wi-Fi, GSM. Wireless transfer of data is the need of an IoT device. Data from across the application placed anywhere needs to be sent to the cloud. Thus we need to carefully analyze which technology to be used.

Each of the technology has its pros and cons.

Table 3.

Standard	*IEEE 802.15*	*Bluetooth*	*Wi-Fi*
Frequency	868/915MHZ,2.4 GHZ	2.4GHz	2.4,5.8GHz
Data rate	250kbps	723kbps	11 to 105 Mbps
Range	10 to 300m	Low	High
Battery Operation	Alkaline(months to years)	Rechargeable(Days to weeks)	Rechargeable(Hours)
Power	Very low	Low	high

Decision Made : Smart water mixer can be used in places like houses, schools, hospitals, offices etc. Houses and offices normally have a Wi-Fi connection. So building Wi-Fi devices seems feasible and easy. Thus we have provided continuous internet connectivity using a Wi-Fi shield on an Arduino Board.

3.1.4. Other Components

Along with the basic components, we need many accessories such as motors, LCD screens, fingerprint scanners etc. These peripheral devices add up to the total cost of applications.

Decision Made : In the Smart Water Mixer, we have a biometric authentication system where fingerprints of users are scanned. If the user is registering for the first time, the details are sent to the cloud. If the user has already registered, the data for that user is fetched from the cloud. Based on the usage pattern of that particular user, the temperature of the mixer is set, and user gets the desired water temperature. Along with that, we have a dashboard which displays the user usage details on the screen.

3.2. Communication Protocols

A very huge amount of data is sent from the IoT devices to the cloud/server. Handling this humungous data needs support of lightweight protocols. These protocols enable low power usage and also work on low bandwidth .It is very important that the data does not get tampered and reaches quickly with a very low latency.

Considering this IoT specific nature, communication industry has come up with IoT specific communication protocols like MQTT, COAP etc. We will have a look at how these are different from traditional data communication platform like HTTP by comparing with MQTT.

3.2.1. MQTT

MQTT, the Message Queue Telemetry Transport, is publish-subscribe based lightweight messaging protocol on top of the TCP/IP protocol.

3.2.3. HTTP

HTTP functions as a request–response protocol in the client–server computing model.

3.2.3. MQTT over HTTP

MQ Telemetry Transport Protocol (MQTT) has a number of advantages over HTTP. MQTT is a light-weight publish/subscribe messaging protocol. HTTP is designed as a request-response protocol for client-server computing, not optimized for mobile and push capabilities, particularly in terms of the battery usage. In the mobile environment, response times, throughput, lower battery use and lower bandwidth are key design criteria. Compared with HTTP, MQTT gives faster response and throughput, and lower battery and bandwidth usage. MQTT is designed for low latency, assured messaging and efficient distribution. HTTP is not optimized for low power usage or minimizing the amount of bytes flowing.

Decision Made : After comparing the benefits we realize that MQTT is better suited for our application than the HTTP. In this project we send the data regarding the users' water usage and preferences to the cloud via the MQTT protocol.

The data is passed via the Wi-Fi shield on the Arduino board to the router and thus data is sent to the cloud.

3.3. Storage

The staggering rate at which data is generated in an IoT application brings us to another device consideration *i.e.* the storage. Processing and integrating the data becomes difficult due to the volume, velocity and volatility of data generated. The sensors continuously collect the data. Thus there is a grave problem of how the data should be stored. IoT imposes fewer data quality and integrity constraints. Some kind of delay is allowed as long as huge data loss does not happen. The data collection is very different from traditional database systems. There is more flexibility in terms of the ACID properties.

So while choosing the best possible database we must consider these points:

- scalability
- schema flexibility,
- fault tolerance and availability
- ability to send data at sufficient rates
- integration with analytics tools

NoSQL Database

Over the past few years, we have seen a rise of new type of databases, NoSQL database which challenge the existence of traditional relational databases. There are many storage mechanisms based on the user needs.

The basic features of a NoSQL database are: no use of relational model, runs well on clusters, generally open source, schema-less. It allows users to build applications without having to convert in-memory structures to relational structures. The different types of NoSQL databases are Graph, Key-Value, Document Store, and Column Store.

Few of the NoSQL databases are :

Table 4.

	Aerospike	***Cassandra***	***CouchDB***	***MongoDB***
Category	Key-Value	Column-Store	Document-Store	Document-Store
CAP	AP	AP/CP	AP	CP
Consistency	Configurable	Configurable	Eventual	Configurable
Querying	Internal API	Internal API,SQL like	Internal API	Internal API, Map Reduce, complex query support
Partitioning	Proprietary	Consistent hashing (Third Party)	Consistent hashing	Consistent hashing

Decision made : In our Smart water mixer, we have used MongoDB database to store the data. We have programmatically set the data to be sent in the JSON format. As MongoDB is a Document Store database, we send all the data from Arduino in the same form. This is the most suitable database.

3.4. Cloud

IoT applications generate huge amount of data. They also need regular data processing & scaling in terms of IT infrastructure at the backend. The varying nature of infrastructure needs and large scale integration can be provided by cloud. Computing functions like Lambda can be invoked for individual device response. We will not need dedicated servers, thus providing scaling and cost effectiveness. There are tools provided by cloud platforms like Big data processing, machine learning over the large scale data produced by IoT devices which can help in quickly setting up an end to end solution.

3.5. IoT Platforms

A frequent dilemma while developing an IoT application is the platform to be used. Should we use our own servers or just host our application on one of the already available platforms such as Amazon Web Services, IBM Bluemix, etc.? IoT platforms bring all the applications under one roof. They provide users with endless applications to process the data collected. They store, analyze, sort, make intelligent decisions on the obtained data.

AWS-IoT cloud platform, as compared to other platforms provides virtual devices, rules engine, object store, more ways to connect other than internet, telecom integration. In addition it also provides secure communication framework in terms of security certificates for devices.

IBM Bluemix is a cloud platform as a service (PaaS) developed by IBM. It supports several programming languages and services as well as integrated DevOps to build, run, deploy and manage applications on the cloud. Bluemix is based on Cloud Foundry open technology. Bluemix supports several programming languages including Java, Node.js, Go, PHP, Python, Ruby Sinatra, Ruby on Rails and can be extended to support other languages such as Scala through the use of buildpacks. Large storage of data is possible because of Cloudant NoSQL database. Both of these platforms more or less provide us with same features.

IoT platforms vs. Own servers : IoT platforms are a fast and easy way to get started on the Internet of Things. As all the data is stored on the cloud provided by the platforms so we don't have to worry about scaling. The cost of operation for IoT platforms is negligible. The problem of choosing the operating systems and configuring the servers is eliminated. These platforms provide us with various other services under one roof, so we no need to integrate with the third party services. Thus building IoT Applications from scratch is a complex endeavor.

Decision made : Since in our application there is a possibility of poor network connection. We need the virtual device state maintenance provided by AWS IoT. IBM Bluemix provides Workflow management for IoT , which is not provided by AWS. However in our case the workflow is not varying. It is defined at the start and is pretty much constant for the use case. Hence we would not need the IBM Bluemix work flow management. In future we might also need integrating devices with telecom network and issuing commands via SMS. This is mostly because of the possibility of poor network connectivity. AWS IoT provides better framework in terms of telecom integration as compared to IBM Bluemix. Considering all these we chose AWS IoT platform for our use case. This is how various factors like protocol support, telecom integration, security, rules engine, etc. influence the selection of IoT platform in different use cases.

3.6. Cost

Perhaps the most important factor in designing an end-to end IoT application is the cost factor.

Example : In our application there are 3 basic units. The cost per unit is calculated below.

1. **Water Mixer Unit – Integrated with Shower: Cost: $60**
   (*a*) Detects user and automatically start shower with preferred temp (without wasting water to achieve the desired temperature level)

(*b*) Continuously records current water temperature and consumption data

(*c*) Automatically stops when user goes away

2. **Display Panel (Dashboard) near Shower: Cost: $25**

   (*a*) A screen to display some data like Information of user, his preferred temperature, current temperature, water flow rate, water consumption(from start of the shower till end for the current session).

   (*b*) Note: This cost can be further reduce if a simple UI to be provided with LCD shield/panel in place of a different monitor/screen : Cost: $4

3. **Water Mixer Web Server/Mobile App: Cost: $400**

   (*a*) Stores data for each user/home for each shower session

   (*b*) Shows various charts and graphs to display historical data for better analysis and predictions like per user/home/area water consumption in a month, average water temperature preference for users in a particular area

4. **Total cost**

   (*a*) With a small monitor/screen : $485

   (*b*) With a LCD shield/panel : $464

Decision made : We have compared with the existing projects in terms of the cost and specifications like the EvaDrop Smart Shower ($599) and Amphiro meters ($529) and thus building our own system from scratch provides more features and is cost effective.

4. TESTING

Testing of IoT application needs special consideration as compared to traditional application because IoT devices are placed in open public places, have physical connectivity issues, security issues, environmental issues. Considering this we need to have following important validations for an IoT application.

1. **Components Validation** which includes device hardware testing, Embedded software testing, application testing, cloud infrastructure testing, network connectivity testing, third party software testing.
2. **Function Validation** which includes testing of the various interacting devices, basic device testing (start/stop/restart/interrupt), error handling.
3. **Performance Validation** which includes testing of the data-transmit frequency, multiple request handling, synchronization, interrupt testing, device performance, and consistency validation of the device.
4. **Security and data Validation** which includes validation of data packets, verification of the data loss and/or of the corrupt data, data encryption/decryption testing, data value testing, testing of the correctness of user roles and responsibility and also of the usage pattern.

Example : Here we first test the individual components. We have mentioned few test cases here.

Test Case 1: Water Temperature Control – To provide user's preferred water temperature

Test Case Name: Water Mixer Unit _Temperature Control – 15 minutes shower activity with extreme hot (above 70 degrees) and cold temperatures (below 20 degrees) in respective water containers.

Test Case Objective : Verify accurate sensory functionality and servo motor actuation over extended 15 minutes

Test Subject : A shower with integrated unit and a person capable of taking shower for 15 minutes in preferred temperature with no more than a 1-minute break (to gather test verification data) once every 1 minute.

Pre-conditions : Water mixer unit is programmed, for a person with already set preferred temperature and identity.

Table 5.

Step	*Action*	*Expected Result*
1.	Put correctly a pre-programmed device on the shower with a person for taking bath with preferred temperature. The preferred temperature value of the subject should be hard coded in the program.	
2.	Manually take cold and hot water container's water temperature (using standard thermometer).	The hot water temperature reading should be above 70 degrees) and cold water temperature reading should be (below 20 degrees) in their respective containers.
3.	Start the shower	
4.	Take manual reading of the water temperature (using standard thermometer).	Readings of water must be within 5% of the subject's preferred temperature.
5.	Have the test subject take the bath and take a feel of water temperature.	
6.	Repeat steps #4 and #5 for 15 additional cycles having test subject perform bathing activity for 1 minute. Verify temperature readings while test subject takes a few seconds break.	Test subject shouldn't feel difference in temperature during temperature readings.

Test Case 2 : Water Mixer Unit _ Identifying User - Taking user's identity and preference.

Test Case Name: Water Mixer Unit _ Identifying User– Identifying a user by using his fingerprint and if it is a new user, take input of his preferred temperature; else if existing user access his preference from the server. **Test Case Objective:** Verify accurate sensory functionality and data reading from server.

Test Subject : A shower with integrated unit and a person (initially a new user and then an existing user)

Table 6.

Step	*Action*	*Expected Result*
1.	Put correctly a pre-programmed device on the shower with a person who is a new user.	The shower should be off.
2.	User prompted to get fingerprint scan which then should be securely sent to server.	
3.	As User is new, fingerprint gets stored at server.	Finger print gets stored at the server and successful user registration comes from the server.
4.	On successful user registration response, user interface should take user's temperature preference using a keypad.	User preference gets stored at the server side and a response with preferred temperature should be sent back to device and shower should give shower water with preferred temperature.
5.	Take manual reading of the water temperature (using standard thermometer).	Readings of water must be within 5% of the subject's preferred temperature.
6.	Skip steps #3 and #4 if the fingerprint is already stored at the server side.	Server should send subject's already stored preferred temperature to the device.
7	Repeat step #5	

Now we test the entire system by performing integration testing

The various individual components are :

- Water Mixer Unit _ Identifying User
- Water Mixer Unit _ Detect Presence of a Person
- Test Water Mixer Temperature Control Unit
- Display panel(Dashboard)
- Test Water Mixer Web Server/Mobile App

We need to test the working of these components together.

- Water Mixer Unit _ Identifying User + Water Mixer Unit _ Detect Presence of a Person
- Water Mixer Unit _ Identifying User + Water Mixer Unit _ Detect Presence of a Person + Test Water Mixer Temperature Control Unit

The other components of the system like the display panel, water mixer web server/mobile application and their interaction must also be tested. Thus Quality Assurance plays a major role in building an IoT application.

5.CONCLUSION

By taking an example of Smart Water mixer we have seen how design considerations vary for IoT applications as compared to traditional applications. We have seen how various factors like cost, hardware platforms, IoT platforms communication and cloud platforms, testing have an impact on developing an IoT application. A very good understanding and detailed analysis of these design considerations is required to be done when building a cost effective, efficient, scalable IoT application.

6. REFERENCES

1. AWS cloud http://docs.aws.amazon.com/gettingstarted/latest/wah-linux/getting-started-application-server.html
2. https://aws.amazon.com/ec2/getting-started/
3. 10 Internet of Things (IoT) Design Considerations: Cost and Network http://gridconnect.com/blog/general/10-internet-things-iot-design-considerations-cost-network/
4. Gazis, V.; Gortz, M.; Huber, M.; Leonardi, A.; Mathioudakis, K.; Wiesmaier, A.; Zeiger, F.; Vasilomanolakis, E. (2015), *A survey of technologies for the internet of things*, in Wireless Communications and Mobile Computing Conference (IWCMC), 2015 International , vol., no., pp.1090-1095, 24-28 Aug. 2015 doi: 10.1109/IWCMC.2015.7289234
5. What the Internet of Things (IoT) Needs to Become a Reality: http://www.nxp.com/files/32bit/doc/white_paper/INTOTHNGSWP.pdf
6. Thomas Zachariah, Noah Klugman, Bradford Campbell, Joshua Adkins, Neal Jackson, and Prabal Dutta http://iot.stanford.edu/pubs/zachariah-gateway-hotmobile15.pdf
7. “A review of connectivity challenges in IoT-smart home “ S. SujinIssac Samuel Department of Computing, Middle East College, Muscat, Sultanate of Oman10.1109/AICCSA.2015.75072652015 IEEE/ACS 12th International Conference of Computer Systems and Applications (AICCSA)
8. “Direction Detecting System of Indoor Smartphone Users Using BLE in IoT” - D. Kothandaraman, C. Chellappan DOI: 10.4236/cs.2016.78131 http://www.scirp.org/journal Paper Information. aspx? paperID = 67362 “