IOT and RaspberryPi:Report1

Mohammad Mahdi Amini

Isfahan Uninersity Of Technology
mm_amini@ec.iut.ac.ir

June 12, 2017

I. Learning Internet of Thing

This section provides a overview of contents of a book named "Learning Internet of Things". There are 9 chapters in this book covering projects which are all carry out by the help of Raspberry Pi[1]. Each chapter explains an aspect of IOT ranging from devices to communication protocols and security concerns.

The following part reorgenized chapters into a path of steps.

i. Overral view of main steps

1: Connecting a range of sensors to Raspberry Pi then reading their values.
Conecting LED's as actuators to Raspberry Pi then make them act.

Sensors

Temprature, light and motion detection sensores are used in this chapter and they are conntected to Raspberry Pi using hardware protocols like UART[2] and I2C[3]

2: Persistening sensors' values in a local database.
Makeing the raw values and derived graphs from them available on a webservice.
Making it possible to order actuators by the help of webservice.

Database

The mentioned database should be installed on the Raspberry Pi. SQLite[4] is adequate choice here since it is designed to work on embedded devices with low power processors. SQLite is popular for being used in the core of Android. Although it is a relational database, it does not support complex relations and joins in order to remain simple and lightwight.

3: Notifying other thing about our available services by the help of UPnP protocol and responding to searchs for a spesefic service.
A mechanism to inform users when a sensor's value changed.

[1] Smart card sized cpmputer providing direct access to GPIO and other hardware technologies. It support high level programming languages like "Java" and "Python" becuase of different operation systems can be installed on like Linux

[2] (Universal Asynchronous Receiver/Transmitter),which is a protocl for commiunication between devices having serail interfaces(port). Some types of keyboards use this protocol.

[3] I2C is a protocol working in serial manner connecting two-wire interfaces. It is suitable for low-speed devices like microcontrollers and sensors

[4] www.sqlite.org

UPnP

UPnP (Universal Plug and Play) is a standard using Web and Internet protocols to provide mechnism to make range of devices able to automatically know each other exsistence, network address and provided services, after getting connected to network. The devices also can learn each others' command languages for using availbe services. UPnP is suitable for smart-homes and local-networks. One possible simple senario for using UPnP is when a camera trying to find an available printer on a local netwrok in order to print recently captured images.

4: Becoming familiar with CoAP protocol and use it instead of HTTP in previous projects.

CoAP

CoAP (Constrained Application Protocol) is machine to machine (M2M) protocol connecting constrained deviced in IOT environments. CoAP architecture is basically Rest-full (like HTTP) in which resources are available under an URL. It is designed to work properly on devices with limited set of resources for example 10 KiB of RAM and 100 KiB of code space .Inspite of the fact that CoAP is very similar to HTTP, it uses UDP as its transport layer protocol. CoAP supports different types of data ranging from JSON to XML.

5: Becoming familiar with MQTT protocol and carrying out previous projects with it in a asynchronous manner.

6: Becoming familiar with XMPP.
Developing previous project in a asynchronous manner by the help of XMPP.
Considering security appraochs about XMPP.

XMPP

XMPP (Extensible Messaging and Presence Protocol) is a XML based messaging protocol make secure(optional) real-time communication possible. It is possible to connect wide range of devices working on different platform by the help of XMPP. XMPP architecture is based on efficient push mechanisms making real-time messaging possible without wasting servers and clients resources.

7: Knowing IOT service platforms basics and capabalities.
Make provious project cooler by connecting them to an suitable IOT platfrom.

8: Understanding security threats in IOT world and How to avoid them.

II. MQTT

MQTT is a asyncronous messaging protocol which is lighweight enough to fulfil IOT reqirements like limited bandwidth. This protocol supports publish/subscribe communication pattern and its architecture lie on a middlewear software called Broker. Providing using messaging protocols instead of protocols like HTTP which are synchronous, the clients do not poll servers so the resources of both will used more sufficiently. MQTT is very usefull in cases that informing other machines of an event is nessecary.

MQTT implementations are availible in almost any programing and scripting language so divices ranging from low powererd sensors to servers in data centers can talk to each other with no concern about their differences.

i. Java sample

The paragraphs below contain Java codes that expalin each step of MQTT usage cycle, respectively. The codes are written in a way that avoid CPU blocking (I/O blocking) and are used in an Android application successfully.

Initialization

```
            MQTT mqtt = new MQTT();
            mqtt.setHost("192.168.1.6", 61613);
            mqtt.setClientId("213"+Math.random());
            mqtt.setUserName("admin");
            mqtt.setPassword("password");

            connection = mqtt.futureConnection();
            Future<Void> f1 = connection.connect();

            f1.await();
```

Subscribing to a Topic

```
            Future<byte[]> f2 = connection.subscribe(new Topic[]{new Topic("tt", QoS.
    AT_LEAST_ONCE)});
            byte[] qoses = f2.await();
```

Receiving messages

```
        while (true) {
                try {

                    Future<Message> receive = connection.receive();

                    Message message = receive.await();

                    output.append("\n--> " + new String(message.getPayload()));

                    message.ack();

                } catch (Exception e) {
                    e.printStackTrace();
                }
```

Sending a message

```
  Future<Void> f3 = connection.publish("tt", input.getText().toString().getBytes(), QoS.
    AT_LEAST_ONCE, false);
```

III. Broker: Apache Apollo

I used Apache Apollo as messaging broker to test the code from privous section. Apollo supports diffrent messaging protocols. However there is no need to congif it for an specefic one since it is desined to be intelligent and find out the protocol of messages automaatically.
Also there is no installation phase when using it. Just excrating then runnig is enough.

IV. A remained task

I will provide some explanations about the nature of messagin protocols and some details about MQTT in addition to a description about brokers and their role in my next report.

References

[1] Peter Washer. *Learning Internet of Things*. Packt, 2015.

[2] Apache. activemq.apache.org/apollo.

[3] Mqqt. Mqqt.org.

[4] CoAP. http://coap.technology.

[5] XMPP. Xmpp.org.

[6] I2C. i2c.info.

[7] UART Margaret Rouse. whatis.techtarget.com/definition/uart-universal-asynchronous-receiver-transmitter, 2011.

[8] UPnP Margaret Rouse. whatis.techtarget.com/definition/universal-plug-and-play-upnp, 2011.