

Study of Internet of Things using Simulator

Mimi Cherian

*PCE, New Panvel,
Mumbai University, India.*

Abstract

The following paper is a humble attempt to understand the theory behind working of Internet of Things and also the practicality of its working using emulator and online tools. Design of IOT is very complex having four main parts such as Devices or sensors that collect data, Transport that transfers the collected data across different communication networks, Storage that stores data collected from nodes in cloud, Interfaces help to generate reports based on data mining results on the data stored in cloud. Cooja emulator is used to deploy sensors and data are sent between them based on activities to be performed by sensors. Splunk an online tool is used to generate reports based on sensor data uploaded into it. This paper is an attempt that helps in understanding end-to-end working of IOT.

Keywords: IOT tools, emulators, protocols, IOT architecture.

1 INTRODUCTION

Currently researches in IOT is upcoming topic in which we try making devices smarter so that they can talk to each other and try to be proactive in decision making which indirectly helps in implementing strategic decisions and later for data analytics. The devices can be sensors that have ability to sense change in its environment, these sensed data from different sensors are to be sent through internet to gateway and proxy to cloud. The data received in cloud is later used for strategy planning and decision making. The decisions made are then sent to actuators for implementation of decision.

Anything that can be joined to its processing unit (microcontroller) and connected to the Internet is considered as things in the world of IoT. In IoT, the interconnected products should always be low-cost, so that we can flood the planet with IoT devices. The first step in building an IoT device is to figure out how it will communicate with

the rest of the world [1]. The IOT concept broadly refers to RFID, infrared sensors, GPS, laser scanners and other information sensing devices, according to the agreed protocol, to achieve any time, any place, any object information exchange and communication in order to achieve intelligent identification, locate, track, monitor and manage a network [2].

1.1 Conceptual end to end Architecture of IOT

The Conceptual architecture of IOT is in Fig 1. We have multiple sensors which collect data from different environments that data is passed through internet gateway and then sent to cloud. The huge amount of data collected from different sensors are to be processed in cloud to find meaningful data that can be used for decision making. The decision making done by user is sent from cloud to actuators.

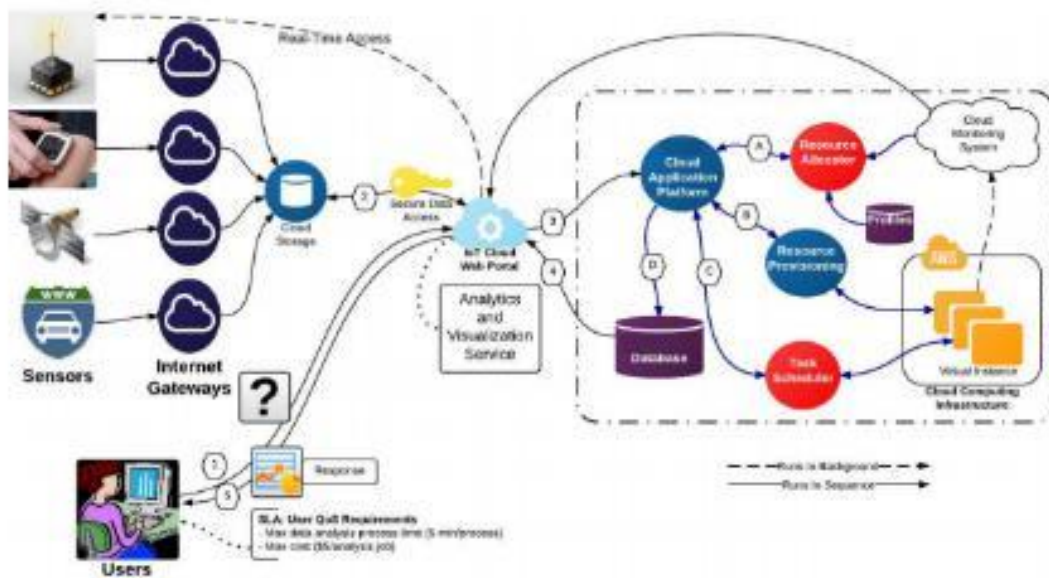


Fig 1. End to end IOT Architecture [3]

1.2 Five layered Architecture

The Five layered architecture of IOT is related to network layers. In Fig 2, a five-layered architecture of IOT is designed.

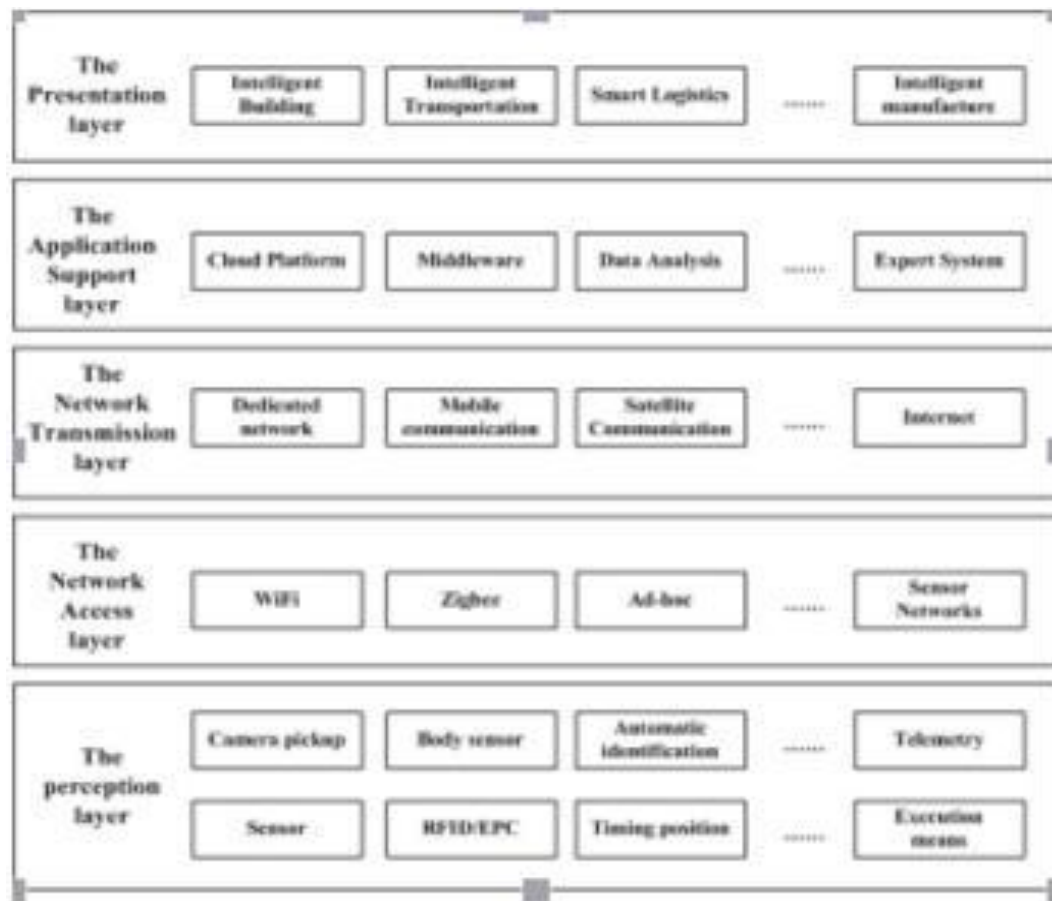


Fig 2. Five Layered IOT Architecture [3]

The bottom layer works on collecting data from direct sensor nodes. The network access layer and network transmission layer deals with sending the sensor data through direct internet access points after conversion of data in format that application support layer can understand. The application layer and presentation layer works with processing data and gathering information from the collected data.

1.3 Gateway hardware architecture

The IOT gateway bridges the perception network and access network, it supports different types of sensors (such as ZigBee, RS485, CAN 6LoWPAN,) and the method of access (such as cable, WLAN, GPRS, 3G) also provides a unified data format for middleware or application, in order to protect different sensor network and the access network.

The processor module in g 3 is the core module of the gateway, which implements the protocol conversion, management, security and other aspects of data processing and

storage. The zigBee module realize the collection of phys-ical world data or together, can be the convergence of sensor network nodes, the RFID reader, video collection equipment, GPS, etc. Through the network access module, the gateway will access WAN by the way including cable (Ether-net, ADSL, FTT), wireless (WLAN, GPRS, 3G, satellite) [4]. The sensor datas are sent through many gateways and proxy to cloud which requires separate communication protocol stack. Sensors have limitation as to have least process-ing unit and long battery life by reducing the duty cycles. IOT protocol stack is created accordingly so that data processing at sensor node will be least.

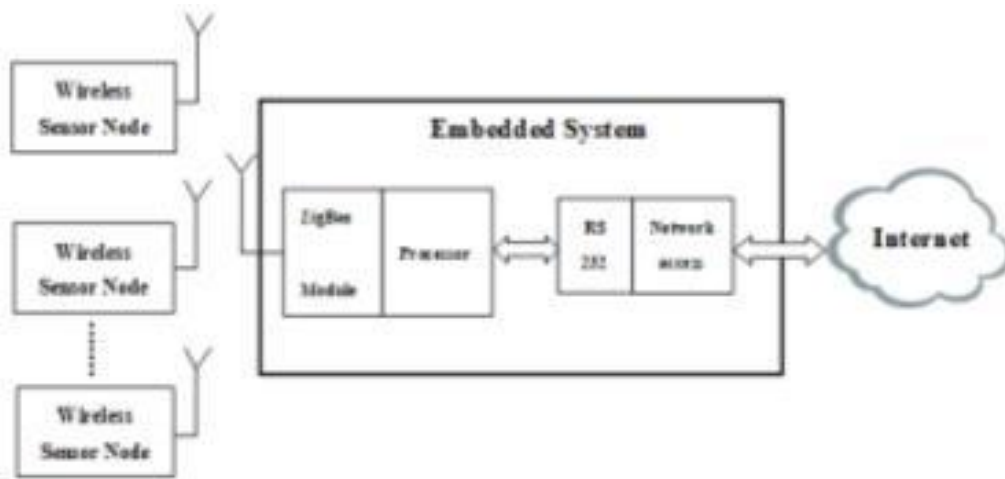


Fig 3. Gateway hardware architecture [4]

1.4 IOT Communication Protocol

The Protocol stack is designed such a way that all the network and device re-quirements are fulfilled. The network requirement is scalable, self-healing, and secure and end node addressable while device requirement low power, low memory and low battery. IOT needs standard protocols, and two promising protocols are MQTT and COAP.

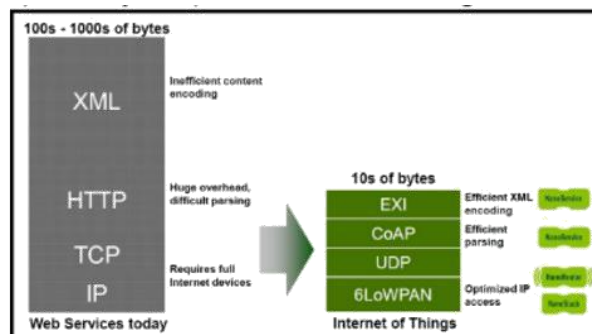


Fig 4. IOT Protocol Stack [6]

MQTT stands for Message Queue Telemetry Transport and CoAP stands for Constrained Application Protocol these protocols are used for communication between resource constrained IOT devices and Resource rich devices based on Internet. MQTT uses Client Server publishes or subscribe messaging transport protocol. MQTT is an open source protocol that was developed and optimized for devices that are restrictive and has less bandwidth or networks are unreliable. It is very lightweight and suitable for devices that consumes minimal bandwidth. It supports publish/subscribe messaging transport.

In Fig 4 consists of IEEE 802.15.4 is a standard that views the access between media access control and physical layer for (LR-WPANs) low-rate wireless personal area networks. 6LoWPAN stands for IPv6 over Low power Wireless Personal Area Networks. Since billions of latest devices will be connecting in future to the Net it has to be IPv6.

CoAP is transported over UDP and is a binary protocol. UDP's are connectionless datagrams that also enable to transmit cycles as well as smaller packets with few overhead and has faster wake-up. Hence it allows devices to be sleepy state for longer periods of time thus battery power is conserved. CoAP protocols semantics were designed on the basis of HTTP. COAP is a binary protocol thus comparatively its data overhead is less and while its use of UDP it increases its compatibility with communication models and enhances the ability to reduce latencies. That concludes that CoAP is not limited to just the semantics of HTTP. The benefits of using HTTP semantics on top of CoAP's UDP rather than HTTP's TCP is that device can more conveniently utilise the same protocol code to communicate with other devices and cloud on the local network.

EXI stands for Efficient XML Interchange (EXI) format. It represents a compact XML. It is designed such that it should support XML applications with high performance for environments that are resource limited that requires less bandwidth and improves performance of encoding/decoding. EXI based upon the current XML schema the processing stage and the context tries to compress and reduces information about the document structure to internally generated small tags. To create as compact as possible the tags data representation is optimized as far as possible. Even though an efficient compression can be gained from the XML schema, the standard defines different operating modes to give a more compact and optimized representation of the XML le by using only partial or no XML schema information. The encoded XML document gives

EXI stream that presents the document in binary format where every data tag of the document is encoded by using an event code. Event codes are binary tags that maintains their value only in their assigned position within the EXI stream.

The IOT protocol stack is designed such that EXI carries XML data into binary and CoAP protocol uses UDP along with 6LoWPAN ensures faster wake up cycles with least battery consumption then so that communicating with least processing sensor node will be effective.

1.5 End to End IOT Simulation:

Cooja is a Contiki network emulator An extensible Java-based simulator capable of emulating Tmote Sky (and other) nodes. Cooja emulator can be used to create an environment where sensors can be added and data can be sent from sensors to cloud. The code to be executed by the node has the exact same firmware that may be upload to physical nodes. It allows precise inspection of system behaviour by enabling small and large networks of motes to be simulated Motes that can be emulated at the hardware level.

1.5.1 Cooja Emulation

In Fig 5 creating a broadcast simulation requires different types of motes which can be added currently we consider adding firmware of Sky mote as its most simple and can be used in WSN.

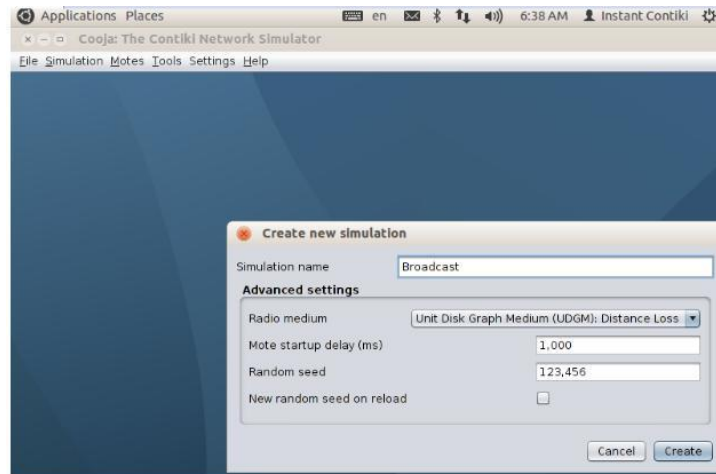


Fig 5. Create simulation

In Fig 6 Sky mote is added which can be used for further implementation The example is considering a sink node and multiple sender nodes. Hence in

Fig 7 we are naming this sky mote as sink. The best part of cooja emulator is it allows to create motes with same firmware as if it's of physical devices. After selecting the sink program from the given path we compile the program so that the program of sink

will be running on sky mote sink.

After compilation of sink mote we can add multiple motes if required. In Fig 8 only one sink node is considered.

In Fig 9 the new sink mote added is now visible on cooja emulator screen Addition of several sender motes can be done by adding sender mote with sender program that can be compiled on it.

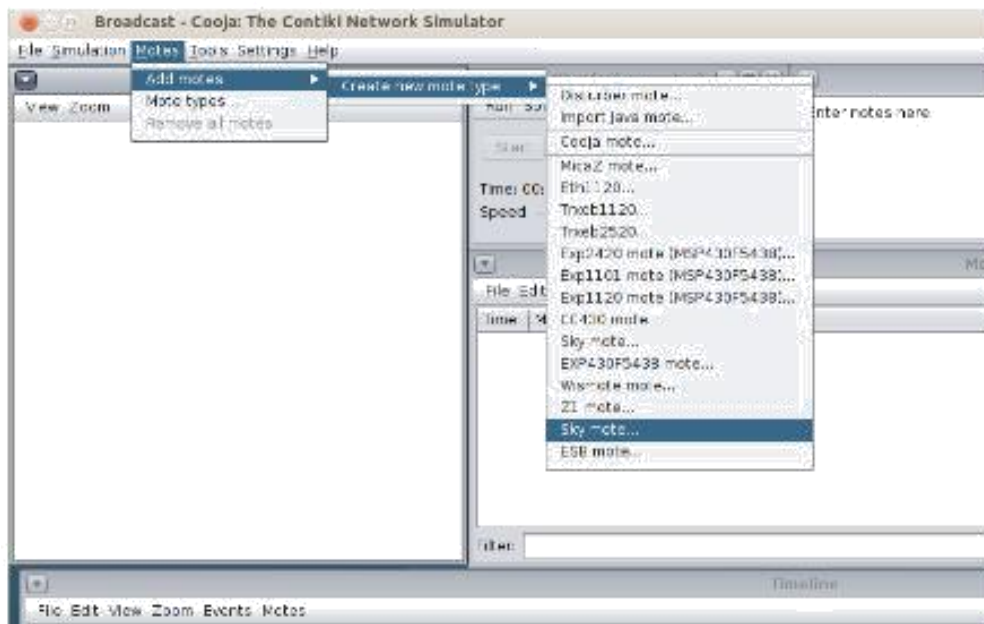


Fig 6. Adding Sky mote

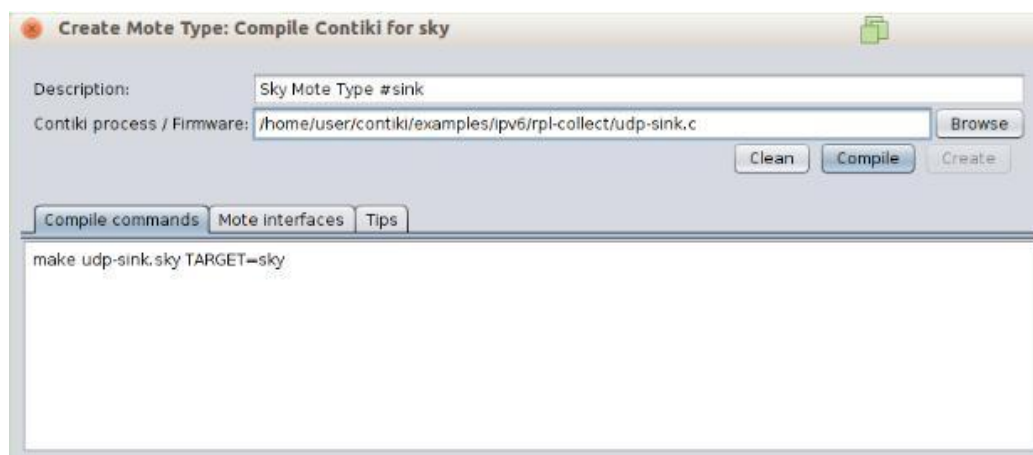


Fig 7. Compiling Sky mote as sink node

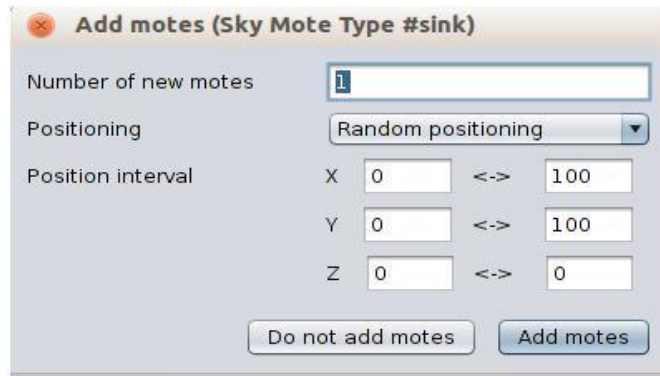


Fig 8. Adding sink notes

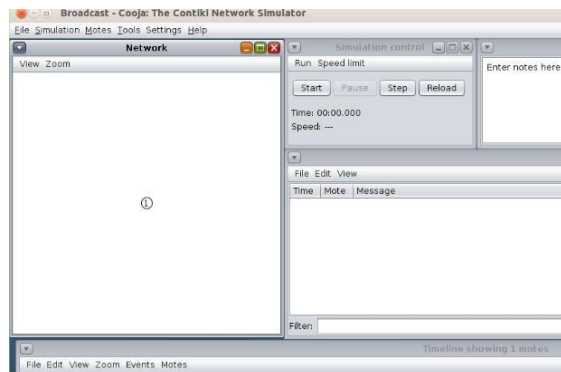


Fig 9. Sink mote visible on emulator

In Fig 10 On completion of compilation seven sender notes can be added. several sky mote for sender can be added. With the help of network window's view option we can enable many factors of network. We can choose the background grid, output of each mote and its addresses.

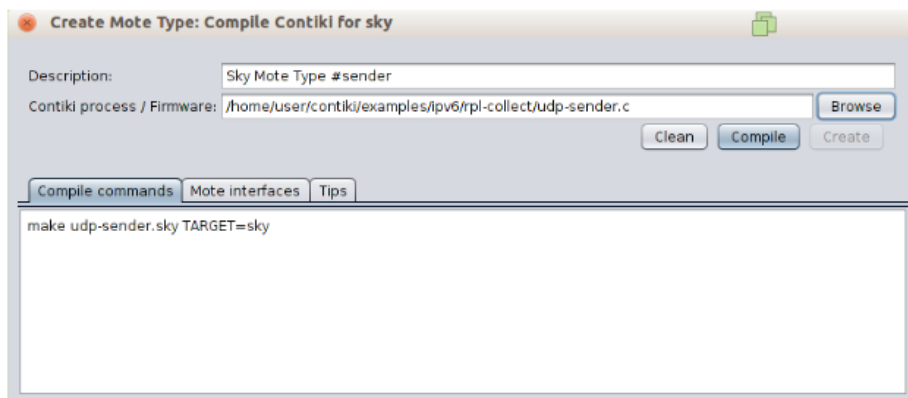


Fig 10. Adding Sender mote

The Execution of motes with one sink node and several sender motes is shown in Fig 11.

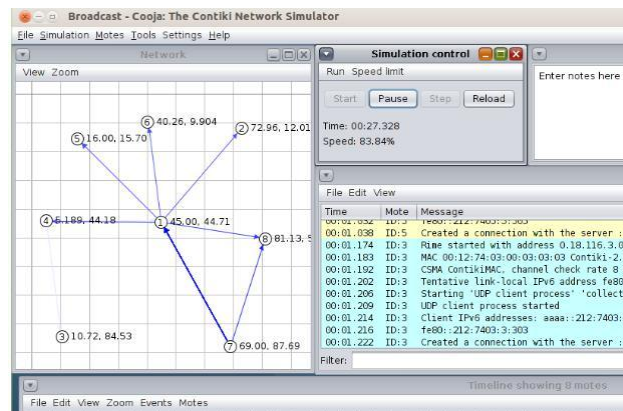
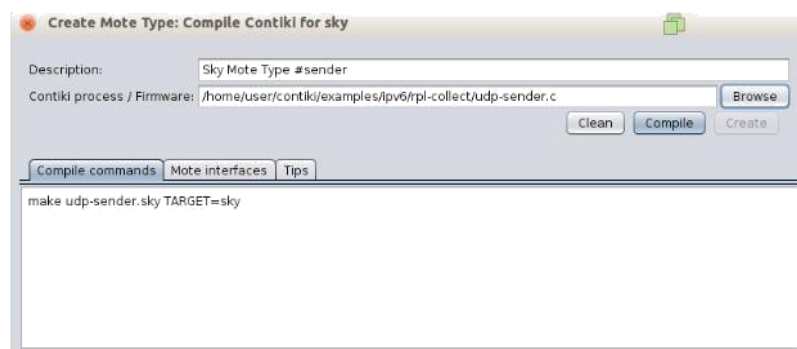


Fig 11. Output communication between sink and sender motes

The reports generated by cooja simulator includes Power consumption of each mote can be viewed from tools option. Sensor data of each sensor node can be collected using collect view option in View tab which enables us to do further analysis and decision making.

The sensor data collected from each mote can be viewed. The output of collected sensor data is also visible. The average power consumption can also be viewed. The script of code can also be altered using script editor within Cooja, select the 'Tools' drop down menu and the 'Simulation Script Editor'. The user requires Knowledge of Java Script to understand in detail analysis of the network output. The Script Editor can be used to display messages and to set a timer on the simulation for its simplicity. In order to implement this, within the Script Editor menu we need to select the 'File' drop down menu of which 'Load example script' is the only option.



2 CONCLUSION

The main motive of this paper is to understand the basic working of Internet of Things starting from collecting sensor data till its process of getting reports

generated for further analysis. With the help of cooja emulator we could get at least a testbed to analyse the behaviour of different sensors nodes deployed in various situations. Working on emulator gives us a better perspective of how sensors works and collects data in physical environment.

REFERENCES

- [1] Internet of Things (IoT): Architecture and Design 2016 Al-Sadeq International Conference on Multidisciplinary in IT and Communication Science and Applications (AIC-MITCSA)", IIRAQ (9-10) May
- [2] Zhang Mingjie, Han Jianting, Hu Bingsong, Liu Wenchao ,\Building Home Application System of Internet of Things with Home Gateway", TELECOMMUNICATIONS SCIENCE, 2010(4),P44-47.
- [3] Jayavardhana Gubbi,Rajkumar Buyya, Slaven Marusic,Marimuthu Palaniswami, \Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions", Volume 29 Issue 7, September, 2013 Pages 1645-1660
- [4] Chang-Le Zhong, Zhen Zhu, Ren-Gen Huang,\Study on the IOT Architecture and Gateway Technology", 14th International Symposium on Distributed Computing and Applications for Business Engineering and Science 2015.
- [5] Web Services for the Internet of Things through CoAP and EXI , \ <http://electronicdesign.com/iot/mqtt-and-coap-underlying-protocols-iot>" [online].
- [6] \ <http://www.iotevolutionworld.com/m2m/articles/208798-unleashing-internet-things.htm>" [online].
- [7] Birju tank, Hardik Upadhyay, Hiren Patel,\Mitigation of Privacy issues in IoT by modifying CoAP", 2015.
- [8] Jie Lin, Wei Yuy, Nan Zhangz, Xinyu Yang, Hanlin Zhangx, Wei Zhao,\A Survey on Internet of Things: Architecture,Enabling Technologies, Security and Privacy, and Applications, DOI 10.1109/JIOT.2017.2683200.