Software Defined Internet of Things

Version 1.0

Dipak Majhi

Department of Electronics & Communication Engineering.
Kalyani Government Engineering College.

Title : Prototype design of Software Defined IOT devices.

Date: 20.01.2016

a) Short discussion on focus about Technology :

Most of the network architectures are non-programmable and static. Because of this limitation it has become difficult for the network engineers to add new services into existing network architecture. In the present scenario, there is a growing interest in the fields of Internet of Things (IOTs) and Software Defined Networking (SDN). It has become very popular in industry applications as well as academia. SDN, created by Open Networking Foundation (ONF) solved many problems related to this, resulting into an efficient and reliable network management system. With the help of it, developers developed new services or new protocols without waiting for its release from the vendors end. Controller Intelligence is centralized, resulting into a global view of the network topology and changing demands. SDN will enable Internet of Things devices to share network resources more efficiently, and hence reduce the hardware investment.

b) Problem we are trying to address through this invention :

Software-Defined Networking (SDN) is used to make network programmable and dynamic. OpenFlow is a protocol of SDN, which has gained attention because of its flexibility in managing networks and centralized control. But as Open Flow is a heavy protocol, it is not suitable for sensor nodes .Currently, SDN in sensor nodes has limited functions. The activities of it are static and non programmable. It cannot change its settings depending upon the real life situation. We have addressed the problem by building a dynamic device management and topology management system.

Objectives of South Bound API:

1. Interfacing of sensor nodes.
2. Multitasking of sensor nodes.
3. Uploading of sensed data to server.
4. Enable Bidirectional Communication.
5. Flow Table implementation.
6. Real time Destination change or topology change.
7. Enable packet forwarding or drop operation.
8. Enable Broadcast operation.

c) Novelty of this invention:

In this invention, we could change the application specific tasks, depending on our needs in real time. We could change the destination address of every nodes or change the topology of the network. As we implemented multi-tasking of sensor nodes so we can also change the function of each sensor node dynamically. Apart from this, we could change delay of every node, give the commands to drop or forward packets received from a particular node in real time depending upon user need and also broadcast any instruction to all the sensor nodes. There are predefined commands to change the address, sensor activity, delay, forwarding or dropping of packets of a certain node and also to broadcast any message. (refer Instruction set) which should be send from the controller end to the sensor nodes to enable changes in their activity.

For sensor nodes:

Components Required	Specifications	Quantity
Arduino Board (Mega 2560)	Operating voltage-5v, Input Voltage 6-20V, DC Current per I/O 40 mA.	1
XBee 802.15.4 Module	Series 1, 3.3V, 1mW	1
Sensors (Gyroscope, Gas, Flame, etc.)	-	5 (depends on demand)
XBee Shield	3.3V	1

For access point node with WiFly :

Components Required	Specifications	Quantity
Arduino Board (Mega 2560)	Operating voltage-5v, Input Voltage 6-20V, DC Current per I/O 40 mA.	1
XBee 802.15.4 Module	Series 1, 3.3V, 1mW	1
RN-XV WiFly Module	4uA sleep mode, 38mA active, TCP/IP, 3.3V	1
Xbee/WiFly Shield	3.3V	2

Procedure : /** South Bound API part **/

1. Started with interfacing Passive Infra Red (PIR) motion sensor with Arduino.

A PIR motion sensor has three pins.

- DC voltage (VCC) connected to the 5v pin of Arduino.
- Ground (GND) connected to the GND pin of Arduino.
- Output (OUT) connected to the pin 10 of Arduino for output.

Now to make it work, just read the pin 10 of the arduino where the output of the sensor is given. The total Arduino code is attached (refer Motion_Sensor_Final.ino).

2. Next interfaced Flame Sensor with Arduino.

A Flame sensor has four pins.

- DC voltage (VCC) connected to the 5v pin of Arduino.
- Ground (GND) connected to the GND pin of Arduino.
- A0 connected to the analog pin 0 of Arduino.
- D0 connected to the pin 8 of Arduino for output.

Now to make it work, just read the pin 8 of the arduino where the output of the sensor is given. The total Arduino code is attached (refer flame.ino) .

3.Next we interfaced Gyroscope Sensor with Arduino. It is a multipurpose sensor and can be also used for Temperature and Accelerometer sensor.

A Gyroscope sensor has four pins.

- 3.3v DC voltage (VCC) connected to the 3.3v pin of Arduino (use level converter if needed).
- Ground (GND) connected to the GND pin of Arduino.
- SCL (Serial Clock) connected to the A5 of Arduino UNO or its respective pins for other boards.
- SDL (Serial Data) connected to the A4 of Arduino UNO or its respective pins for other boards.

The total Arduino code is attached (refer gyro.ino) .

4. Next interfaced Temperature Sensor with Arduino. We used Gyroscope sensor to serve the purpose.

   A Temperature sensor has four pins.
   - 3.3v DC voltage (VCC) connected to the 3.3v pin of Arduino (use level converter if needed).
   - Ground (GND) connected to the GND pin of Arduino.
   - SCL (Serial Clock) connected to the A5 of Arduino UNO or its respective pins for other boards.
   - SDL (Serial Data) connected to the A4 of Arduino UNO or its respective pins for other boards.

   The total Arduino code is attached (refer gyro_temp.ino) .

5. Now if we want to interface all the four types of sensors in one board we need to merge the codes into one. Normally, the code can be divided into three parts: the variable declaration part, the setup() function that runs only once and the loop() function the runs repeatedly. All the global variable declaration and initialization are done in the first part then in the setup part we initialize/set few variables value and begin the serial port at a certain baud rate (generally 9600) Remember to assign a unique output pins to each sensor. However, separate functions could be added in the code to simplify it (debugging gets easier).

6. In the third part it's better to make all the sensor codes as individual function which could be called using a switch case in the loop() function. In this way we could implement multitasking of sensor nodes which will work on demand from the user. A sample code is attached (refer Final_Multitasking.ino).

7. In the next phase we need to develop a system that could implement the concept of Software Defined Networking (SDN) and also multitasking of sensor nodes.

8. For this purpose, we needed three Arduino (preferably "Mega") boards, male female connecting wires, Zigbee modules (series 1) with its shield, USB connector and rn-xv WiFly module.

9. Out of the three arduino boards one will become the Controller that will forward or receive the data directly from the server. It will have a zigbee and a wifly attached to it. The other two will become sensor nodes having multiple sensors and zigbees. Each of them will be having zigbee connected to them to send/receive data and communicate with the Controller.

10. Now, Zigbee's can be configured using 'XCTU' software available for Windows, Linux and

Mac. Open the application and configure it by setting it to its default values and giving it its own MY address. In this system, we reserved {0.1} for controller and {2,3} for sensor nodes. API 1 could be enabled through the software or it is done through code by entering into its command mode using (+++) and exiting it using (ATCN). Few commands for it are ATDL (used to set set destination), ATWR(to write the changes on it permanently) etc.

11. To build the sensor nodes first connect the sensor to the arduino board (description given above). Then connect the zigbee by using only its four pins (3.3v, TX, RX, GND) to the respective pins in the arduino board. The TX pin of the Zigbee should be connected to the RX pin of the arduino and the RX pin of the Zigbee should be connected to the TX pin of the arduino. In this way two sensor nodes can be configured. The sensors could be added according to the description given in initial steps {1,2,3,4}.

12. To build the Controller we need to connect the Zigbee and Wifly to it. The Zigbee and Wifly both can be connected following the above process (as they uses the same shield) . Connect the Wifly by using its four pins (3.3v, TX, RX, GND) from the shield to the respective pins in the arduino board. The TX pin of the Wifly should be connected to the RX pin of the arduino and the RX pin of the Wifly should be connected to the TX pin of the arduino. In this way two sensor nodes can be configured. The Wifly is configured through code. The command to enter its command mode is “$$$” and then “open __IP address__ __port__” would connect the wifly module to the server . The IP address and port number should be of the server e.g “open 10.14.18.10 1346”.

13. The zigbee's in all the nodes are connected in serial 1 (TX1, RX1) and the wifly in the controller should be connected at serial 2 (TX2, RX2) . The serial port ((0 and 1) or (TX0, RX0) pin in the arduino board) is left to check the commands received from the controller end or sensor end.

14. There are pre-defined commands to change the address, sensor activity, delay, forwarding or dropping of packets of a certain node and also to broadcast any message.(refer Instruction set) which should be send from the controller end to enable changes in the sensor nodes.

15. The syntax for the commands is :

Type 1 : < Case Identifier > < Node ID > < Separator > < Sensor ID > < End Delimiter >

Type 2 : < Case Identifier > < Node ID > < Separator > < Delay > < End Delimiter >

Type 3 : < Case Identifier >< Node ID >< Separator >< Drop/Forward >< End Delimiter>

Case Identifiers : &, $, @, !

- '&' : used for Sensor Change.
- '!' : used to set Delay.
- '@' : used for packet drop or forward.
- '$' : used for Destination change.

Node ID : 00, 02, 03 ('00' used for broadcast)
Separator : *
Sensor ID : 01,02
End Delimiter: #
Drop : 01
Forward : 00
Delay : 10 (represents 1 second, as it will be multiplied with 100 by default. Maximum delay=9.9 secs)

16. Packets could be send in Broadcast mode. The only thing to change in the commands is that the Node ID should be changed to “00” instead of sending it to a particular Node ID.

Table 1 : Instruction set.

To Change	Node 2	Node 3	Node 4
Destination	$02*00#	$03*00#	$04*00#
Delay	!02*10#	!03*10#	!04*10#
Sensor 1	&02*01#	&03*01#	&04*01#
Sensor 2	&02*02#	&03*02#	&04*02#
Drop packets	@02*01#	@03*01#	@04*01#
Forward packets	@02*00#	@03*00#	@04*00#
Broadcast packets	{$,!,&,@} 00 * {0,1} #	{$,!,&,@} 00 * {0,1} #	{$,!,&,@} 00 * {0,1} #

Table 2 : Senor ID

Sensor ID	Sensor Name
01	PIR Motion Sensor
02	Flame Sensor
03	Gyroscope Sensor
04	Temperature Sensor
05	Gas Sensor
06	Accelerometer Sensor
07	Raindrop Sensor
08	Soil Humidity Sensor
09	Sound detection Sensor
10	Ultrasonic Sensor module
11	BMP180 pressure Sensor module

ARCHITECTURE :

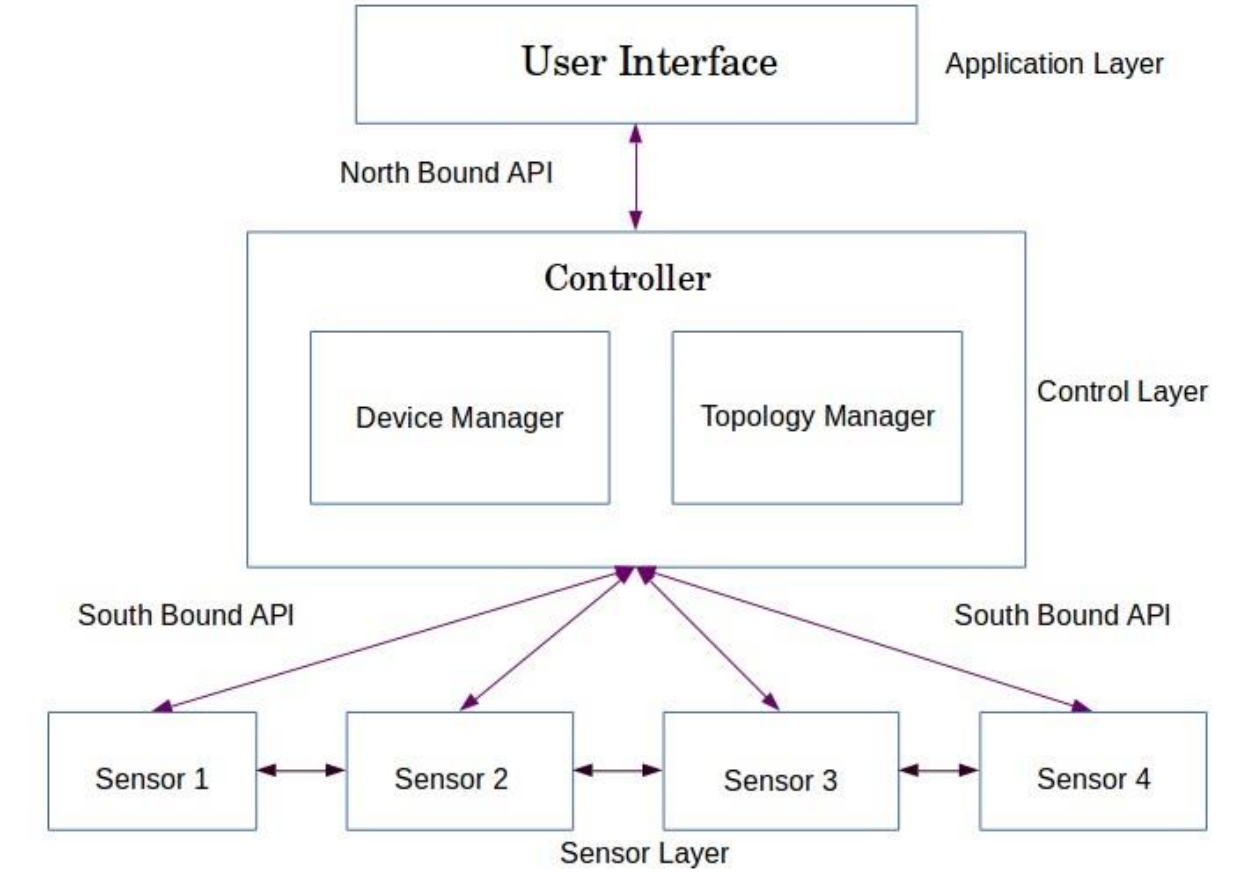

Fig.1 Total Architecture.

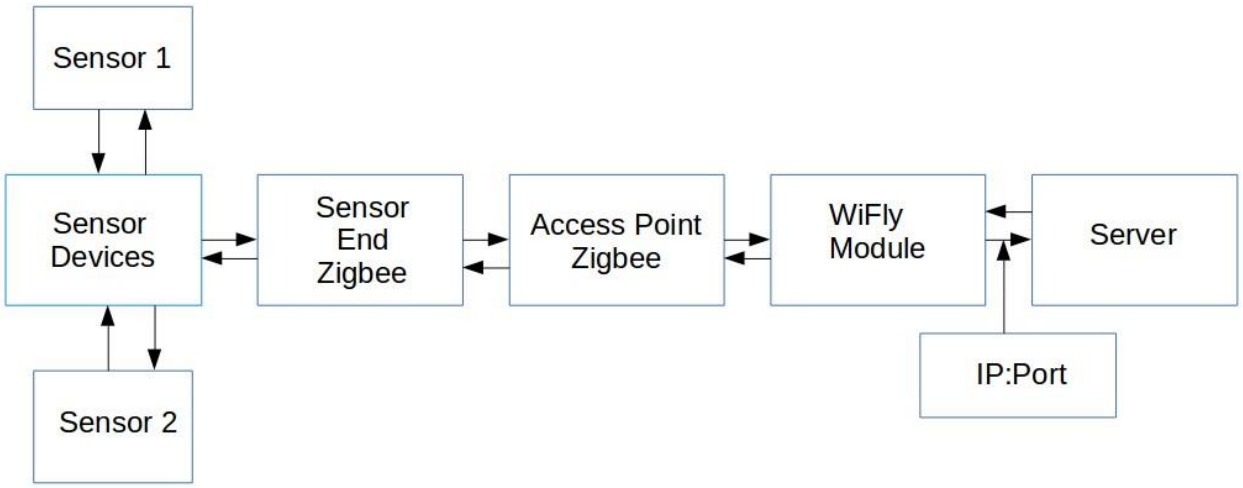

Fig.2. Data flow architecture.

d) Keywords about this invention :

1) Multitasking of Sensor node.
2) Maximised Packet delivery ratio.
3) Minimised Delay.
4) Minimized Network Overhead.5) SDN in IOT devices.

e) Reference- Patent/Paper.