

Design and implementation of Wireless Sensor Networks in Smart Buildings

Lurdh Pradeep Reddy Ambati	lambati@umass.edu
Ravi Choudhary	ravichoudhar@umass.edu
Uday Savaria	usavaria@umass.edu
Vijay Gopal Krishna	vgopalkrishn@umass.edu

Contribution :

Lurdh Pradeep Reddy Ambati : Development of iOS Client and installing Parse cloud code, implementation of python script to upload data values to server.

Uday Savaria : Implementation of Python GPIO Library and Sensor Interface with Raspberry Pi 2.

Vijay Gopal Krishna: Installation of GTK+2.0 library and Bluez Bluetooth stack. Development of GUI - GTK+ and implementation of Bluetooth Server and design of the attack.

Ravi Choudhary : Installation of Bluez Bluetooth stack, python library and implementation of Bluetooth Client.

1. Abstract:

In this project, we design and implement an “inexpensive energy saving Bluetooth point-to-point wireless smart sensor network”. Communication in the network happens over the Bluetooth from one node to gateway or central node. Smart sensor design is discussed and described in this report/project as well as network design we implemented. Also, in this project, we use the Facebook Parse as the Cloud Server and for backend web services, an iOS client is developed which integrates the Parse service to monitor the network.

2. Specific Aims:

In this project, we want to implement the WSN using smart nodes and communication between the nodes and gateway is established using Bluetooth LE modules. Also, we want to improvise over the project/idea defined in Environmental Wireless Sensor Networks By Peter Corke, Tim Wark, Raja Jurdak, Wen Hu, Philip Valencia, and Darren Moore, by using the Bluetooth LE modules for communications and by using a Linux Kernel as a platform for Smart nodes instead of TinyOS. To implement a mobile client, through the network can be monitored.

3. Background and Significance:

Wireless sensor networks are a network of specialized miniature transducers which are capable of computation, communication and sensing physical & environmental conditions like temperature, pressure, sound etc. [1]. Wireless sensor networks are inspired by the military applications like surveillance; today WSNs are employed in many industrial(factory) and residential environments.

Wireless sensor network consists of ‘nodes’ (ranging from few to several thousands), each of which is interfaced/connected to a ‘sensor’ [1]. Such a node is typically constituting of: a radio transceiver to transmit the sensor data associated with it, a microcontroller which interfaces the sensor to the node and an energy source. [1]

In these networks, nodes transmit data wirelessly to central node or a gateway which stores these data in local database or may sent to cloud. Availability of such data enables monitoring network remotely from anywhere using a web application or mobile application (WSN integration with the web or mobile applications).

Home automation and monitoring(Smart Homes) is the leading application of the WSN, where a number of sensors are deployed to monitor the activity of occupants. The requirement for WSN/smart homes to a radio frequency wireless technology are low power, low cost and range from 15m to 100m. Development of 802.15.4 protocol based devices brought revolutionary changes in the development and deployment of WSNs.The 802.15.4 standard only specifies the physical layer and media access control layer, for the upper layers of networks there are several different protocols like ZigBee [6] and 6LoWPAN [7].

The advent of IOT(internet of things) marks the one of the important progresses in WSN-based smart homes. The intelligence of the IOTs with the wireless sensing technology and ubiquitous connectivity are becoming the central point of remote monitoring and control in the smart homes/WSN.

3.1 Challenges in WSN:

- a) **Energy Efficiency:** Power consumption by the sensing device should be minimum. Wireless sensor nodes should be energy efficient and node should shut off the radio power supply when not in use since their limited energy resource determines their lifetime.
- b) **Robustness:** In industrial strict environment, there are various kinds of uncertain interference,which makes WSN a type of complex system and it should be more error tolerant or robust in order to cope with errors during execution.
- c) **Responsiveness:**
- d) **Self-Configuration and adaptation:**
- e) **Scalability:** Scalability is the key factor which plays an important role in designing the Wireless Sensor Network. The network should always be scalable to adapt to the newly entered node in the network.
- f) **Security of Network:** Security is another major concern for Wireless Sensor Network where attacks varies from eavesdropping on transmissions including traffic analysis or disclosure of message contents, to modification, fabrication, and interruption of the transmissions through routing attacks, node capturing or SYN flooding.
- g) **Heterogeneity**

h) Systematic Design

4. Preliminary Research:

During the course of the project, we had to know the architecture of the wireless sensor networks in smart home/building environment. In order to enable communication between smart nodes and local gateway server, there are a number of wireless protocols available like 802.11, 802.15.1, 802.15.4 etc. But in wireless sensor networks energy is a constraint, so a protocol with a low power consumption was needed, for that Bluetooth 4.0 LE fits our requirement as Bluetooth 4.0 LE modules (USB modules) are available commercially at low cost and also power required for this protocol is less.

For server implementation, we have researched about the Facebook Parse Service and learned about its cloud code application, which removes the complexity of writing a large application to deploy a server.

5. Research design and Methods:

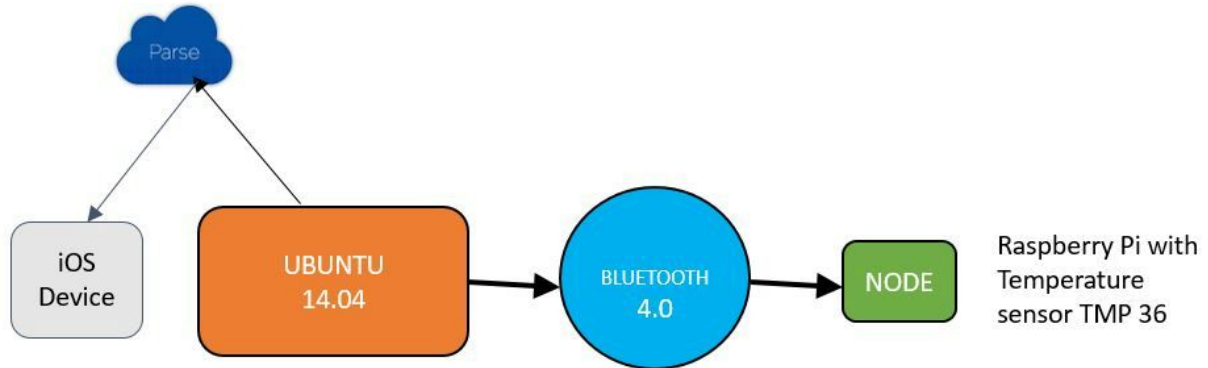


Fig. 1 Architecture of WSN in Smart Building

Design of WSN consists of a Smart node, bluetooth server, a GUI and client-server architecture. In this design, communications happens over the bluetooth channel. Ubuntu 14.04 PC acts as the gateway of the network, which collects the data from nodes and setups the network too. (Bluetooth Server).

GUI runs on top of the gateway, which is used to monitor the network and to send commands/events to the node.

Each smart sensor node is designed using Raspberry Pi 2 and supports Bluetooth 4.0 LE protocol and a sensor is interfaced to each of the nodes.

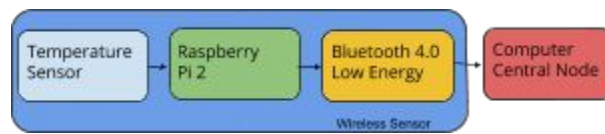


Fig 2. Basic architecture of the Wireless Sensor Node

6. Implementation details:

The figure 1 shows the final setup of the WSN where the smart nodes (Raspberry Pi) communicate with the gateway (Linux PC) through Bluetooth 4.0 low energy protocol. and further, The linux machine is communicating with iOS application through internet. The iOS app gets the data via cloud and shows the list of device connected and its current reading.

The components and Specification of the Wireless sensor network we designed are as following.

Raspberry Pi:

The Raspberry Pi will be running the Raspbian operating system which is based on Debian and optimized for the Raspberry Pi hardware. Raspberry pi is Connected to Sensor and It is also connected to bluetooth 4.0 Low Energy Device. It will host the Bluetooth client using the Bluez Bluetooth stack which is available for Linux.

Central Node:

The central node will be running on Ubuntu 14.04 LTS 64-bit operating system and will host both the Bluetooth server using the Bluez stack available for Linux and the GUI using GTK+2.0 which is also available for Linux.

iOS Application:

The iOS application is running on a tablet which gets data from cloud which was uploaded by Linux machine running GUI. The Integrated iOS app will show the list of devices and current temperature of all available device.

Hardware Specification table:

Raspberry Pi	Version 2.0
Temperature Sensor	TMP 36 GZ
Bluetooth	v4.0 Low Energy
iPad	iOS Device

Software Used:

- Python GPIO Library
- PyBluez
- GTK+ (GIMP Toolkit)
- PyParse
- Socket library
- GCC
- Ubuntu 14.04
- MacOS
- SSH
- TightVNC Viewer
- iOS 9.2
- XCode 7.2

6.1 Wireless Sensor Node:

Each smart sensor/node consists of a raspberry pi to which a sensor is interfaced for e.g., Temperature sensor, Ultrasonic sensor etc. A Bluetooth 4.0 module is attached/connected to the raspberry pi board (Fig), through which each sensor node communicates to the gateway.

For interfacing the sensors on the raspberry pi, a python script is implemented/run which can poll the sensor and store those values in local. For example in case of temperature, when python script polls for value, sensor returns the value of voltage(analog sensor) and then it is accordingly converted to fahrenheit temperature.

For our project, sensors are polled(node polls the sensors) at a frequency of 1 minute.

A bluetooth client application(a C program implemented using Bluez library) is implemented on these nodes, which has 3 functionalities :

- 1) To pair itself(node) with the gateway using a predefined key.
- 2) To receive the commands from gateway over bluetooth channel(once pairing is done) i.e. gateway command to send the sensor data.
- 3) Transmits the sensor data to the gateway over bluetooth channel.

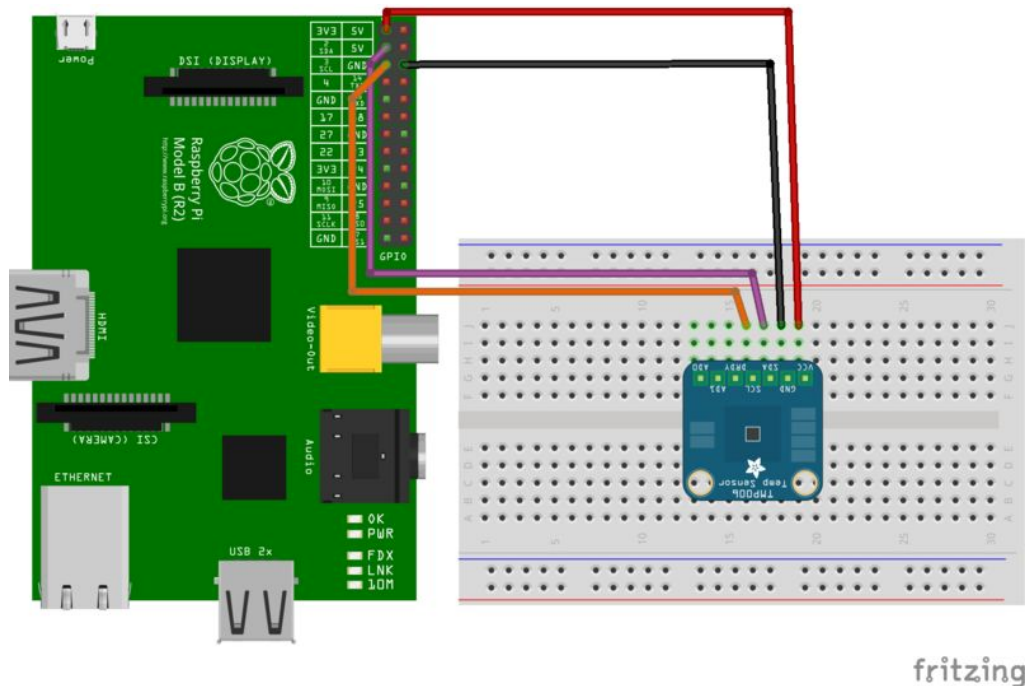
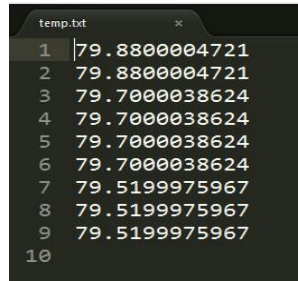


Fig. Smartnode with Temperature sensor and Raspberry Pi 2

When the python code is run on raspberry pi, it will get the value of voltage and accordingly convert the value of the reading to fahrenheit temperature. The output data is saved in the temp.txt file which is generated by the code. The file looks like following.



```
temp.txt
1 79.8800004721
2 79.8800004721
3 79.7000038624
4 79.7000038624
5 79.7000038624
6 79.7000038624
7 79.5199975967
8 79.5199975967
9 79.5199975967
10
```

fig.Temp.txt file generated in Raspberry pi from sensor data.

6.2 Bluetooth Server and GUI:

On the gateway, a bluetooth interfacing application (C program implemented using Bluez library) is implemented, which acts as a bluetooth server. Functionalities of this server are:

- 1) Pairing itself with other smart sensor nodes using a predefined key.
- 2) Sending commands to the smart sensor nodes over bluetooth channel(once pairing is done).
- 3) Receiving sensor data of the respective sensor over bluetooth channel.
- 4) Saving the received sensor data locally in a csv file.

A GUI on Linux platform was developed to monitor the network using GTK+ which also displays which smartnode reported the anomaly. This application will list of sensors that the gateway paired with. This application has the ability to wake up any sleeping sensors (sleeping sensors mean that they are they are not reporting any values).

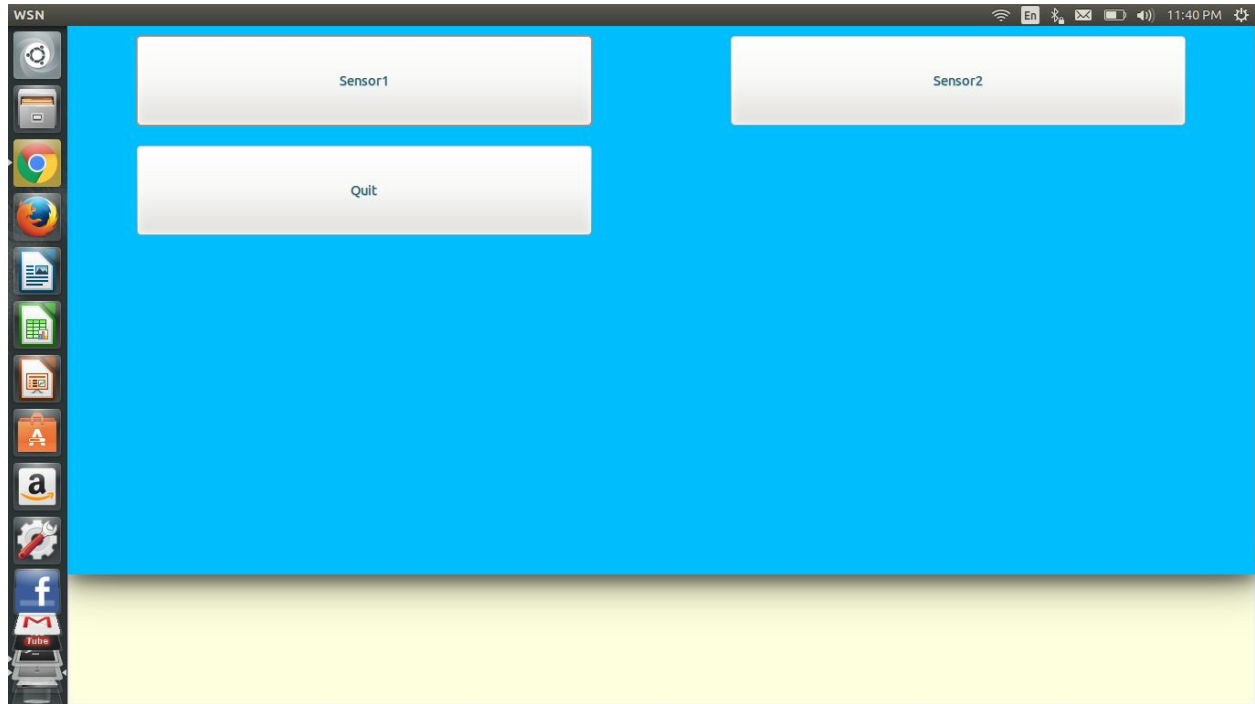


Fig. GUI in Bluetooth Server (Linux machine)

The above figure Shows the GUI in the Linux machine running the Bluetooth Server code. The GUI Pops up when the Raspberry pi client and Linux Bluetooth Server are paired and connected.

There are buttons in GUI to make the easy access to the user. Each button represents the sensor. when a button is pressed the GUI requests the data from the Bluetooth client. client reads the value of the Sensor from the file and sends it to Server. The Server will show the According Sensor and its value.

When the GUI is closed the connection is terminated with the Bluetooth Client.

6.3 Client - Server Architecture:

6.3.1 Parse Server

Parse cloud code is installed on the gateway(Linux PC). Python client/library[8] for the Parse REST API is installed in the gateway PC.

A application is created in parse service, where we created few users and a service/class called Sensor data whose data structure is : {SensorName: String , SensorData: String}.

Python script which uses the python client[8], uploads the sensor data to the cloud(parse application). These sensor data is from the csv file that bluetooth server writes to.

Sensor data is sent to cloud at a frequency of once every minute.

6.3.2 Client - iOS Universal Application

A iOS universal application is build using XCode 7.2 on OS X 10.11. Parse iOS SDK is integrated into this application, which allows the application to access the parse cloud server and enable the application to pull the data from the parse server and can update the data in the parse server too.

Access control is implemented in the application with the help of Parse service.

Application shows a list of the available sensors in the network and shows the details when tapped on a sensor name in the list.

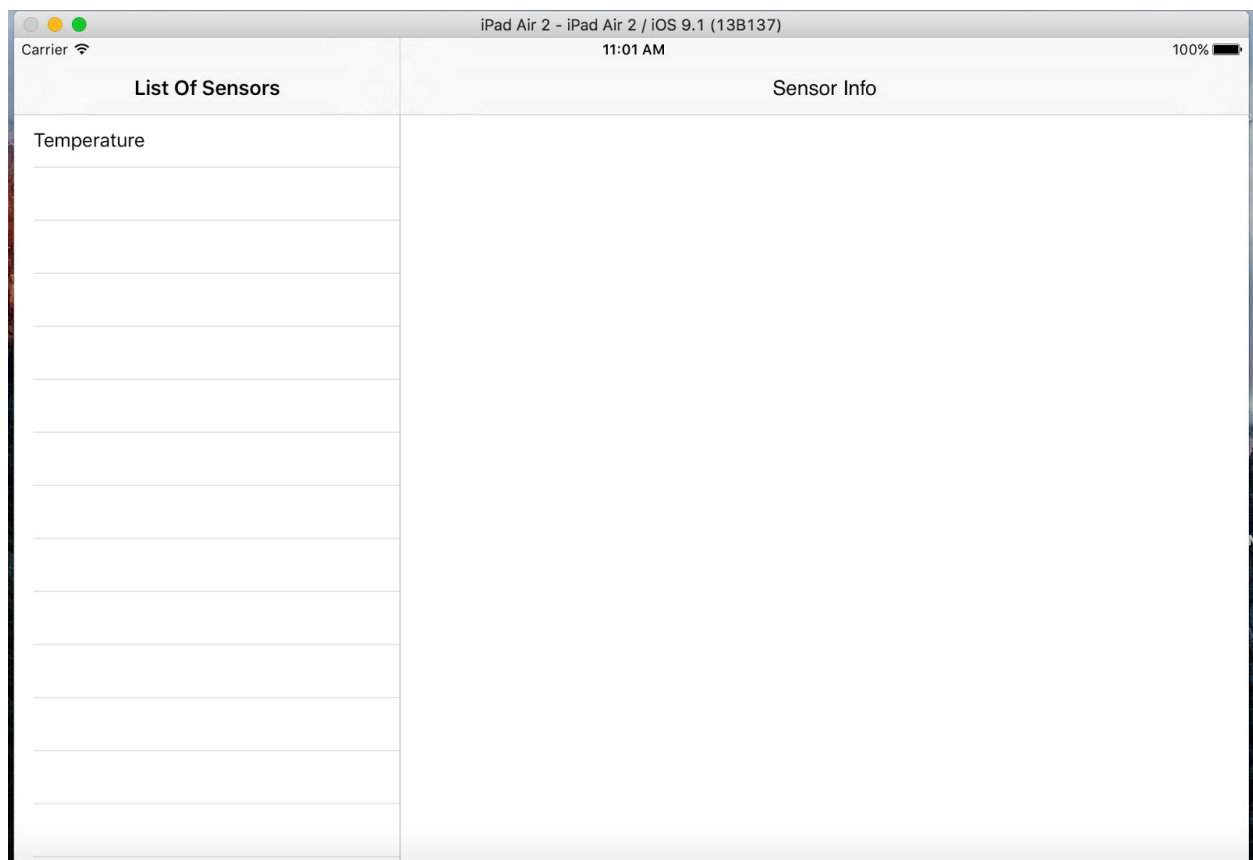


Fig : iOS Client.

7. Literature Cited:

[1] ZigBee-based Data Transmission and Monitoring Wireless Smart Sensor Network Integrated with the Internet by Dražen Pašalić, Zlatko Bundalo, Dušanka Bundalo, Branimir Cvijić

[2] Control of Mobile Robots Through Wireless Sensor Networks Ricardo S. Souza, Lucio Agostinho, Fabio Teixeira, Diego Rodrigues, Leonardo Olivi, Eliane G. Guimaraes and Eleri Cardozo

[3] Environmental Wireless Sensor Networks By Peter Corke, Tim Wark, Raja Jurdak, Wen Hu, Philip Valencia, and Darren Moore

[4]<http://parse.com>

[5]<https://parse.com/docs/cloudcode/guide>

[6]Zigbeealliance,<http://www.zigbee.org>,[Accessed:Feb,2013].

[7]Z. Shelby and C. Bormann, “6LoWPAN: The wireless embedded Internet”, D. Hutchison, S. Fdida, and J. Sventek (ed.), UK: John Wiley & Sons, Ltd, 2009.

[8]<https://github.com/dgrtwo/ParsePy>

