Wireless control of Bluetooth "on/off" switches in a smart home using J2ME in Mobile Phones and PDAs.

Luis Carlos Aceves Gutiérrez
Universidad de Monterrey
Av. Morones Prieto 4500 pte., 66220, San Pedro Garza Garcia, Nuevo León, México
http://www.udem.edu.mx/
laceves@udem.edu.mx

Og Jamir Ramos Juraidini
Universidad de Monterrey
Av. Morones Prieto 4500 pte., 66220, San Pedro Garza Garcia, Nuevo León, México
http://www.udem.edu.mx/
jamiros@gmail.com

Carlos Alberto Garza Frias
Universidad de Monterrey
Av. Morones Prieto 4500 pte., 66220, San Pedro Garza Garcia, Nuevo León, México
http://www.udem.edu.mx/
karlytoz@gmail.com

Abstract

This work focuses on the idea of manipulating a manufactured on/off switch with a Bluetooth receiver, using a JAVA compatible mobile device or PDA through the L2CAP protocol. The main goal is to remove wires from a smart home infrastructure.

Keywords: Bluetooth, Smart Homes, MIDlets, J2ME, Home Automation, Smart Phones

1. INTRODUCTION

This document identifies the simple process used in a daily basis in a standard home and how we can automate this process with the implementation of manufactured Bluetooth on/off switches which receive a signal from a mobile device or PDA that is compatible with JAVA and Bluetooth. To achieve this we send a 1 or a 0 through the L2CAP protocol so that the switch perceives an on or an off.

2. BLUETOOTH AND WIFI

Bluetooth was selected as our way of communicating PDA/Mobile with a central system. The reason Bluetooth was selected over Bluetooth for various reasons. First of all, Bluetooth security is less complex and more stable than that of WiFi. Bluetooth manages a security measure of only permitting certain selected devices to interact with them; WiFi in the other hand establishes a WEP key that has been known to be cracked. Another reason that Bluetooth was selected over WiFi, is that Bluetooth has a shorter range of signal emission than Wifi. This is a pro because the shorter the range the less the amount intruders that will try to infiltrate your home system. [3]

Bluetooth also has some major draw backs. Some Bluetooth components' compatibility with certain operating systems are not supported, also some programming languages still haven't developed libraries to interact with this communication protocol. Installation, real life reception, speed and many compatibility problems are encountered with this form of communication. [3]

3. L2CAP

L2CAP or Logical Link Control and Adaptation Protocol was used in the communication between client and server due that it is one of the lower levels of communication in the Bluetooth Protocol. Since L2CAP is lower level the type of data that is sent either to the client or the server is more primitive and not complicated such as OBEX (Object Exchange) which is high level and sends a certain type of data. In MIDlet programming there are two types of communication objects for Bluetooth. One is SPP and the other is L2CAP, a connection object is instantiated in the client and the server program of either type. A URL is made to set which is the server "btl2cap://", once this is established a UUID is set to show the address to the client so it can be identified. [3]

4. LIGHTING, SECURITY AND APPLIANCE CONTROL WITH BLUETOOTH

Currently we haven't found a manufactured Bluetooth on/off switch with this capability, but as a supposed scenario an on/off switch receives a 1 or 0 signal sent from a mobile device or PDA compatible with JAVA and Bluetooth. A basic system architecture that could be considered is the following, which could be seen in figure 1:

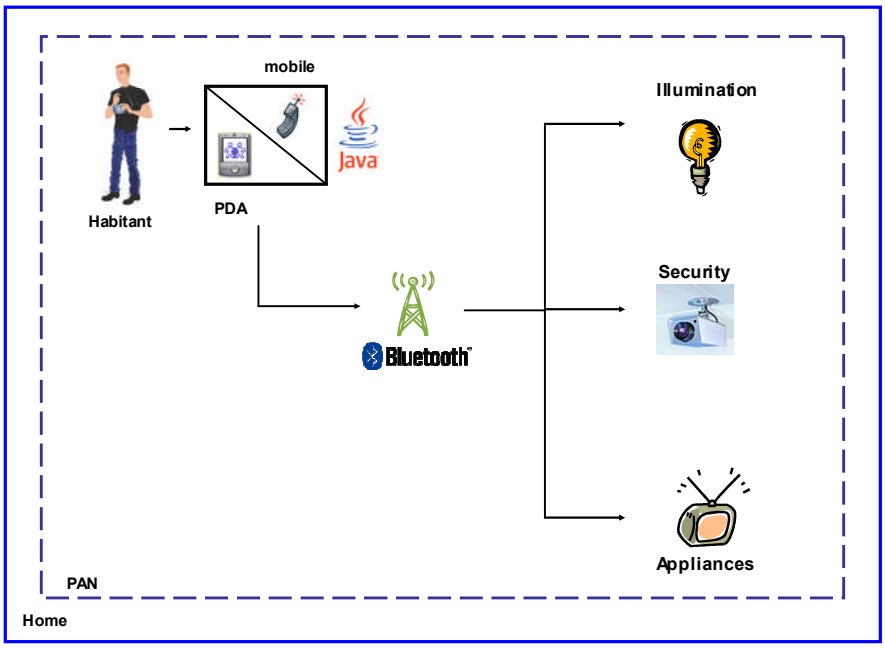

Figure 1. Architecture for Mobile/PDA interaction through Bluetooth

Inside a PAN (Personal Area Network) a user with his PDA device or Smart Phone accesses the client program in JAVA. This program then accesses the PAN through Bluetooth Protocol and sees displayed which are the available devices for control. Through a simple on/off process a 1 or a 0 is sent to the device. The device receives the signal and executes. [3] [2]

Emulation was done with two MIDlets (JAVA programs in mobiles/PDAs) one a server and the other one a client. The server was set as a manufactured Bluetooth on/off switch receiving an on/off signal from the client. The following code example is the server side [1]:

```
public class L2CAP_Server implements Runnable
{  LocalDevice device;
  DiscoveryAgent agent;
  public final static UUID uuid = new UUID("102030405060708090A0B0C0D0E0F011",
false);
  private final static int SERVICE_TELEPHONY = 0x400000;
  public boolean done = false;
  public L2CAPConnectionNotifier server;
```

The code fragment above shows the insanitation of a local device and a uuid being established, also a type of service is being established (Telephony) [1]

```
    public L2CAP_Server()
    {  }
    public void run_server()
    {    try
      {
        device = LocalDevice.getLocalDevice();
        device.setDiscoverable(DiscoveryAgent.GIAC);
        Thread t = new Thread( this );
        t.start();
      } catch ( BluetoothStateException e )
      {
        e.printStackTrace();
      }
    }
```

The code fragment above shows the local device being set to discoverable mode so that any near by switches. This has to be done in threads do to the type of device that the program is being run under. If an incoming call occurs then it will not intervene with the process. **[**1**]**

```
    public void run()
    {   String appName = "L2CAPServerExample";
      L2CAPConnection c = null;
      try
      {
        String  url  =  "btl2cap://localhost:"  +  uuid.toString()  +";name="+
appName+";ReceiveMTU=512;TransmitMTU=512";
        server =  (L2CAPConnectionNotifier)Connector.open( url );
        ServiceRecord rec = device.getRecord( server );
        rec.setAttributeValue(  0x0008,  new  DataElement(  DataElement.U_INT_1,
0xFF ) );
        Util.printServiceRecord( rec );
        rec.setDeviceServiceClasses( SERVICE_TELEPHONY  );
      } catch (Exception e)
      {
        e.printStackTrace();
      }
      while( !done)
      {      try {
          c = server.acceptAndOpen();
          RemoteDevice rdev = RemoteDevice.getRemoteDevice( c );
          int size = c.getReceiveMTU();
          byte[] data = new byte[size];
          int read = c.receive( data );
          c.send( data );
          c.close();
        } catch (Exception e)
        {
          e.printStackTrace();
          return;
        }
      }
    }
  }
```

Here the server receives a signal from the client, once the signal is received the status of the device that is connected with the manufactured Bluetooth on/off switch changes. Various instances of the server are emulated so that it will create a scenario of a real life Bluetooth enable smart home. This way the client will perceive various illumination, security and appliances occurrences to manipulate. [1]

5. CLIENT SIDE SIGNALS

In the client side, the main application that does all the work and organization of the Bluetooth devices, sends a signal to a certain device and changes its status. First a selection is done of what type of device is going to be turned on/off, and then a task is chosen, either a simple on/off or a programmed schedule of an on/off. The following code example is a fragment of the system; it only shows the sending of a signal through L2CAP **[**1**]**:

```
  public class L2CAP_Client
  {
    public L2CAP_Client()
    {
    }
    public void send_L2CAP_packet(ServiceRecord r, String msg)
    {
      String url  = r.getConnectionURL(ServiceRecord.NOAUTHENTICATE_NOENCRYPT,
false );
      url += ";ReceiveMTU=512;TransmitMTU=512";
```

In the code fragment shown above a connection URL is being assigned so that communication between the two devices can be done.

```
      try
      {
        L2CAPConnection con = (L2CAPConnection) Connector.open( url );
        int size = con.getTransmitMTU();
        byte[] data = new byte[size];
        byte[] msg_data = msg.getBytes();
        System.arraycopy( msg_data, 0, data, 0, msg_data.length);
        con.send( data );
        Thread.sleep(1000);
        int read = con.receive( data );
        con.close();
      } catch (Exception e)
      {
        e.printStackTrace();
      }
    }
  }
```

The signal is sent through the URL and received by the server application that it was pointed to; a certain address is established to each server on execution, so a selection of a certain device is needed. **[**1**]**

6. CONCLUSIONS AND FUTURE USES

Some applications for this project in the future could be the integration in a group of houses or colony, where the program will be centralized for the use in all the houses and can detect all the devices available. Using a user and a password for the house's users could manage all the devices of their home.

Another application with potential is using the same structure of the group of houses or colony, but this will be applied on apartment building, where a centralized server has the application and detects all the devices of the different apartments and applying rules used on the colony structure.

Another niche for this project is applying it to corporative offices, using the same structure and rules for the apartment building, the users can only access to the devices that they own in their office and/or extended it to a common area.

An option but not less important, is the application of this project to special houses where the owners are handicapped, this implies the development of special devices like doors, fans, air conditioners to make easy complex tasks and to make easier living in the house.

For the most of the uses the project should be centralized, also it must interact with a database with more capacity to save information of the devices, tasks and users profiles. Also to make a better improvement the project should use another protocol to have a longer reach to detect all de devices on the whole area, because Bluetooth has eight meters of reach, consequentially it can't cover the whole colony area. Another option to detect devices could be the PowerLine that is probably one of the forms of providing an Internet connection.

Even though there isn't a Bluetooth on/off switch in the market, a smart home infrastructure could be implemented easier with this kind of system. Installation and implementation to an already existing home would be very easy.

References

[1] Borches Juzgado, P. D. "Java 2 Micro edition: Soporte Bluetooth" http://www.todosymbian.com/secart43.html
[2] Bursky, S. "The smart home: living with the jetsons" on *Electronic Design,* June 14, 2004
[3] Dasgupta, K. "Bluetooth Protocol and Security Architecture Review" on *Computer and Network Security*, 2000[4] Demaria, M. J. "Home smart home" on *Networking Computing,* January 7, 2002