Blue-Control
A Bluetooth® remote control project for the Sony Ericsson® T610 mobile phone

Mirtcho Spassov
Mirtcho.Spassov@yale.edu

Senior Project
CS 490b – Yale University, Spring 05

Advisor: Yang Richard Yang
Introduction ................................................................................................... 3
Overview ......................................................................................................... 3
Project Details .............................................................................................. 4
Operation Overview .................................................................................... 4
Architecture .................................................................................................. 5
  CommBase.dll ............................................................................................ 5
  LineTerm.dll ............................................................................................. 6
  Configuration.xml ....................................................................................... 7
  BlueControl.exe ........................................................................................ 7
Security .......................................................................................................... 7
Architecture Schematic ................................................................................ 9
Customizing Blue-Control’s remote capabilities ....................................... 10
**Introduction**

The introduction of the Bluetooth technology and its subsequent wide deployment in a range of devices from mobile phones to home appliances has created new possibilities for interconnection unimaginable until recently. In some cases software development has not caught up yet with the capabilities of embedded Bluetooth hardware and the bulk of the technology’s capabilities have not been harvested yet. One such area in which further research and development can be done concerns the communication between a Bluetooth enabled phone and a computer. Bluetooth’s operating range of 10-100 m can allow for a variety of useful applications to be designed and implemented to take advantage of this two-way short distance radio communication link. One such major use is using a mobile phone as a remote control for applications on the computer (such as PowerPoint presentations and media players).

**Overview**

The aim of this project was to create an expandable framework for utilizing Bluetooth-enabled mobile devices to control applications running under MS Windows® (with .NET support) on a user’s computer. I wanted to design Blue-Control in such a way as to be easily adaptable to a variety of hardware. Given the wide availability of efficient Bluetooth drivers that provide the same capabilities and for a variety of devices in this project I assume the existence of Bluetooth drivers on the client machine that are responsible for all Bluetooth-layer communication and can provide programs running on the client computer with an interface to the remote device. In order to make Blue-Control easily expandable, all device-specific communication settings/protocols were abstracted in a separate DLL library, which provides an abstract interface to the Blue-Control main program. This DLL library can easily be swapped with another one in order to allow the application to communicate with a different device. In addition, Blue-Control allows the end user to modify the list of available remote control options on the remote device by adding/removing programs and commands available for each program.
**Project Details**

For this project I used a Bluetooth-enabled Sony Ericsson® T610 mobile phone coupled with a regular desktop with an attached Hawking® H-BT10U Bluetooth dongle. The driver and software that comes with the Bluetooth dongle is a standard WIDCOMM driver supporting most devices of this class. It allow for the establishment of a virtual serial port over Bluetooth with devices within range that support the serial port service. I use exactly this virtual serial port interface to communicate with the mobile phone.

Blue-Remote was also tested successfully on an IBM laptop with a built-in Bluetooth chip and running the IBM supplied drivers. No modifications to the Blue-Remote main code or any of the libraries was required. Blue-Control should be capable of operating independently of the hardware specifics of the Bluetooth link, because it only interfaces with the virtual serial port provided by the underlying drivers through a couple of DLL libraries.

All development for this project was done in Visual Basic.NET under Microsoft Visual Studio.NET 2003. Blue-Control needs .NET framework support on the client machine to operate correctly.

**Operation Overview**

Blue-Control identifies itself as a Bluetooth accessory to the remote device and then generates a remote control menu of programs and corresponding commands available to the user. For the T610 this menu shows up as an entry in ConnectivityÆAccessories.
The user can navigate this menu on the remote device using built-in functionality and once he has made his choice, an ID is submitted back from the device to Blue-Control identifying the menu entry corresponding to the user’s choice. Blue-Control responds by either giving a list of available commands within a program (if the user was at the top level and chooses a program which he wants to control remotely) or by executing a specific command.

**Architecture**

All the programming was done on the computer side. The software application consists of the following parts

- CommBase.dll
- LineTerm.dll
- Configuration.xml
- BlueControl.exe

The way those separate parts interact can be seen from the schematic on the following page.

**CommBase.dll**

The CommBase.dll library is a multi-threaded base class library in C# managed code that can be used from any .NET language through inheritance. It simplifies RS232 serial communications in the .NET environment by providing a solid and easy to use interface. The author of this library is John Hind and more information about it can be found in:
A number of other serial communication libraries are available and can be substituted for the one we used with only a few changes to the rest of the project (actually only to LineTerm.dll, which references and uses it).

**LineTerm.dll**

This library provides the BlueControl.exe the abstraction needed to communicate with any remote device. BlueControl simply builds a device-independent menu structure and submits it for transmission. LineTerm.dll then converts it to the remote device specific command syntax (in our case the T610) and handles communication with the device through the CommBase.dll. LineTerm.dll also parses remote device input whenever a user-choice is made, stores the ID associated with that choice in a publicly accessible property (inSelection) and raises a flag (incomingFlag) to signal the reception.

In order to enable the Blue-Control application to work with devices other than the T610 it would be necessary to make modifications only to LineTerm.dll and more specifically to the command building and parsing parts. It is possible then to create a number of alternative libraries that would enable Blue-Control to communicate with a range of devices. Those libraries should expose at minimum the core functions in LineTerm.dll’s API including the following functions/properties:

- **connectPort** /property/ - set by the caller to the port that LineTerm should use to connect to the remote device
- **Open()**/**Close()** – functions to initialize/terminate a connection to the remote device on the port specified by the **connectPort** property
- **Greet()** – Send commands to the remote device to hook-up Blue-Control in the list of available Bluetooth accessories and display a message on the device’s screen to indicate that BlueControl is active
- **TransactMenu(menu)** – this function converts the device independent *menu* structure submitted to it to a device specific menu string (in our implementation – the T610 Sony Ericsson mobile phone) and submits it to the device
- **incomingFlag** /property/ - this flag should be raised by LineTerm whenever a new response has been received from the remote device. BlueControl keeps checking this flag at regular intervals and if it is raised it reads the **inSelection** value (see below)
- **inSelection** /property/ - inSelection is used to store the last response received from the remote device. BlueControl reads this property whenever the **incomingFlag** is raised.
Configuration.xml

Upon starting, BlueControl.exe reads and parses Configuration.xml to determine the list of available programs and commands. This file is highly customizable and allows for easy modification by the end user. For information please refer to the Customizing Blue-Control’s remote capabilities section below.

BlueControl.exe

This is the core of the Blue-Control application. After acquiring the user-defined remote control program and command combinations from Configuration.xml it builds an internal representation of the menu structure. BlueControl tracks the user’s choices at all times, so it can determine where the user is in the virtual menu structure and what list of available options should be displayed, e.g. if the user at the root level and should be offered a list of programs to chose from or a list of program specific commands. BlueControl submit’s a device-independent representation of the menu to LineTerm.dll and checks the incomingFlag offered by the LineTerm periodically to determine if the user has made his choice. In accordance with the user choice it will either prepare and submit a different menu (such as going from top level programs list, to program specific command list or vice versa) or it will execute the command associated with the menu entry chosen by the user on the remote device - advance the slides in a PowerPoint presentation or play the next file in Windows Media Player play list for example.

Security

As with any new technology it is important not to forget the security implications of BlueControl. Although no special security protocol has been implemented as part of the program, the design itself is robust and prevents a variety of attacks. The major concern for any BlueControl user is a scenario in which a remote user takes over the user’s computer. However, unless the intruder has physical access to the computer, on which BlueControl is running, this is close to impossible. Consider the following:

1. In order for the phone and computer to communicate they have to be paired. The initial pairing requires the same password to be supplied on both devices. By default, Bluetooth drivers request a new pairing phrase each time a new device tries to connect and therefore the computer user has to be actively involved in establishing the phone/computer connection. **Important:** If the computer is set for automatic pairing with any remote device and the password it uses can be easily guessed then it might be possible for an intruder to pair her Bluetooth device with the desktop.

2. Once the devices are paired (and a virtual serial port connection is established) BlueControl has to be run by the user on the computer in order for a remote-control connection to be initiated. It is important to note that this connection gets initiated on the computer side and cannot be triggered by a remote device. Furthermore, on activating the BlueControl link, information is displayed on the remote device to indicate that BlueControl is connected. If a malicious user had somehow managed to pair his device to the computer and overwritten the virtual
port settings to connect to her device and not the legitimate one, a connection to the rogue device might indeed be established at this point. However, the legitimate user seeing that no confirmation of connection has shown on his device and BlueControl reporting an established connection should proceed to disconnect immediately and verify the Bluetooth settings of his computer and whether any device has been paired without his knowledge.

3. Should an attacker circumvent all those barriers, she might indeed be able to use her device as a remote control for the computer running BlueControl (if she is within the operating range, which for the T610 Bluetooth chip is about 10 meters). Still, the attacker would not be able to execute random commands on the user’s computer but just select from the list of pre-defined ones which severely limits the ability to inflict any damage/take advantage of the system prior to the discovery of the intrusion.

Given this theoretical possibility of breaking into the user’s computer (which would require a physical compromise of the machine) it might be important to incorporate security features directly into BlueControl in subsequent releases. One way of securing the connection would be to query the device for a hardware identification number and ask the computer user through a dialog box whether connection to the device with a specified hardware id should be allowed. This security barrier could be compromised by a savvy intruder if she manages to spoof the hardware id of the legitimate device, but it is very unlikely as that would require unrestricted access to the legitimate user’s mobile device. A more solid though somewhat inconvenient solution would be to make BlueControl generate and display a random password on the computer’s screen every time a connection is attempted and then require that the user type this password in a prompt on the remote device before allowing the connection.
Architecture Schematic

Bluetooth® Software Layer
Virtual Serial Port

CommBase.dll ↔ LineTerm.Dll

BlueControl.exe
Windows .NET framework
User-specified Application

Configuration.xml
**Customizing Blue-Control’s remote capabilities**

The XML application control configuration file (configuration.xml) holds all the information describing the menu that appears on the remote device and the commands corresponding to each menu entry. The following is the default file distributed with Blue-Control:

```xml
<?xml version="1.0"?>
<configuration numItems="2">
    <menuitem name="WMP" title="Windows Media Player"
        numCommands="3">
        <command>
            <caption>Play/Pause</caption>
            <key>^p</key>
        </command>
        <command>
            <caption>Back</caption>
            <key>^b</key>
        </command>
        <command>
            <caption>Forward</caption>
            <key>^f</key>
        </command>
    </menuitem>
    <menuitem name="PowerPoint" title="Microsoft PowerPoint"
        numCommands="3">
        <command>
            <caption>Start</caption>
            <key>{F5}</key>
        </command>
        <command altTitle="PowerPoint Slide Show">
            <caption>Next</caption>
            <key>{DOWN}</key>
        </command>
        <command altTitle="PowerPoint Slide Show">
            <caption>Previous</caption>
            <key>{UP}</key>
        </command>
    </menuitem>
</configuration>
```

The first line is required in order to interpret the file as a correct XML file.

```xml
<?xml version="1.0"?>
```

The following lines determine the number of program entries available in the user menu:

```xml
<configuration numItems="2">
    ...
</configuration>
```

In our example we have only two programs – WMP and Microsoft Powerpoint. If the user wants to add additional programs capability, the number will have to be changed to the total number of programs desired.
For each program that for which an entry in the Blue-Control menu is desired, a block similar to the following has to be included within the <configuration> tag described above:

```
<menuitem name="PowerPoint" title="Microsoft PowerPoint"
numCommands="3">
  <command>
    <caption>Start</caption>
    <key>{F5}</key>
  </command>
  <command altTitle="PowerPoint Slide Show">
    <caption>Next</caption>
    <key>{DOWN}</key>
  </command>
  ...
</menuitem>
```

The first line describes the program – the `name` attribute is the text that is going to be displayed on the remote device screen identifying the program and the `title` attribute should be set to the window title of the desired application in Windows®.

The `<command> ... </command>` tags are used to list individual commands available to the user within each program. `<caption>` denotes the name under which the command should appear on the remote device’s screen and `<key>` is the key combination that will be sent to the application if the user chooses the specified command. Blue-Control uses the `SendKeys` class available as part of the .NET Windows Framework; for more information on creating valid key shortcuts please refer to the relevant MSDN article.

By default Blue-Control executes the command by activating the application with title that matches the `title` attribute of the `<menuitem>` tag and then sending the specified key combination. However, for certain application, such as Microsoft Powerpoint® that launch additional windows with different titles to which commands need to be sent, the optional `altTitle` attribute that can be supplied in the `<command>` tag allows the user to over-ride the application title specified in `<menuitem>` and use a custom one. In this case, to start a PowerPoint presentation the command is sent to the main window; however, PowerPoint launches the presentation in a separate window (with the title “PowerPoint SlideShow”) and the `Next` command has to be sent to that new window instead of the main application one.

Blue-Control activates applications with the specified title using the `AppActivate` function under Visual Basic.NET. AppActivate tries to activate an application with the exact title and if it doesn’t find one, it searches for the closest match. For more information on AppActivate please refer to the relevant MSDN article.