Introduction

1.1 Background
In the construction management system, the worker may go down to the tunnel or some places where no one else is with him. For smooth progress of the work as well as safety consideration, the worker needs to communicate with the management when doing his job. For example, in order to construct the building, the floor plans are needed. And when he finished some part, he may want to send back report. Currently, all these are carried out on paper work. However, this is inefficient and troublesome. The worker has to carry some folders with him, and during the work any immediate update cannot be made. In addition, to make the report, he needs to write the things down, and this information can reach to the management only when later the paper is submitted.

1.2 Objective
Our project is aim to solve these problems existing in the construction management system. Now the worker is equipped with a mobile phone and a wearable PC. Through the mobile phone, he can talk and send video streaming. When the management gets the information, update to the floor plan will be made. These pictures will be generated using AutoCAD, which is large software, and running on the wearable PC.

In order to make effective communication, we want to establish a Bluetooth connection between the mobile phone and wearable PC. So the worker can send reports or query his tasks to the PC through Bluetooth technology; and the program running in the PC will handle these inputs and generates the corresponding updates to the main server. Since this communication is implemented in Bluetooth technology, it won’t overload the bandwidth of GPRS, which is responsible for sending video streaming. However the distance between the devices is limited within 10 meters.

1.3 Our Work
This UROP project is actually a small part of a large work. Basically the whole picture is this: We have a central server, which will only stores data. Each construction worker is equipped with a wearable PC, Xybernaut Mobile Assistant V (MAV), which runs in Win2K. The application programs such as AutoCAD, MS Office, are running in this PC. The mobile phone is Bluetooth-enabled Nokia 7650; it interacts with the PC. After PC processes the data and gets the updates of all these floor plans, the picture will be shown on a head-mounted VGA display connected through a wire to the PC.

The voice and video streaming will be sent to central server by the mobile phone through GPRS directly. Between the server and wearable PC, there is data link through GPRS. However, the link between the PC and phone is implemented in Bluetooth.

The application program is written in C++ and using Nokia developer kit SDK 1.0. At the start of one day work, the construction worker will send the queries from the phone to PC, asking for today’s tasks. Then the program running at PC side will handle the input and feedback to the server side. The server will assign the work and make updates to the PC. Then the Bluetooth connection between PC and mobile phone will make the text information received and displayed on phone’s screen. So the worker knows what he is going to do. Apart from the ‘query’, the user interface at phone also handles the
report’, which means after the worker finishes his job, he can choose this menu and report his progress to the server.

2 Technology

2.1 Overview of Symbian OS

Symbian OS is a 32-bit operating system that supports multitasking and multithreading. It is a combination of several different elements that can be specially applied to small handheld devices. There are six elements that combine to make up Symbian OS:

- The core operating system, commonly called a kernel
- A collection of middleware for system services
- A set of resource managers, called application engines
- A framework for designing user interfaces
- Methods for synchronization with other machines
- A Java virtual machine implementation.

2.2 Bluetooth Technology

The Bluetooth model of connectivity allows components as well as independent computers to use wireless connectivity. It identifies a suite of protocols, and focuses on short-range wireless connectivity. It operates at 2.4 GHz bandwidth and in very low power consumption. The speeds of transmission range up to 1 Mbps. The Bluetooth model is built from a specification for radio transmission, a driver for this radio module, and a stack of protocols built on top of the driver that implements various forms of data transmission.

![Figure 1: Bluetooth Stack](image)

A Bluetooth device must discover the services that are available instead of being configured with a static list of them. Service Discovery Protocol (SDP) allows a Bluetooth device to do so. It will broadcast queries and field the responses to find out which device offers specific services. The RFCOMM layer emulates an RS-232 serial line. It interfaces with the L2CAP layer and implements a point-to-point transport method.

3 Design and Implementation

In our design, the Bluetooth connection is initiated and ended by the user of the phone. Also, when he chooses to make the connection, all the Bluetooth devices in range will be discovered and listed on phone screen. The user can choose which specific device to connect to. After success of connection, the user is able to start data communication with the wearable PC. The application is developed using Nokia developer tool kit SDK1.0. Basically, this section includes:

- A use-case scenario explaining the interactions between the mobile phone, the wearable PC, and the user—construction worker, when the message is sent and received between two devices.
- A description of the how the mobile phone initialized the Bluetooth connection, classes and APIs involved in discovering the receiving device, getting information about the Serial Port service and sending a message to it.
- A description of the Bluetooth USB adapter, which advertises the Serial Port service and is responsible for handling the Bluetooth data. And a brief explanation of the how the application program that running at PC side responses to the incoming message.

3.1 Use–case Scenario

This section gives an overview of the interactions between the construction worker, the mobile phone and the wearable PC when simple messages are sent and received bidirectionally. We use Bluetooth point to point connection, and this simulates a virtual serial connection between the two devices. The following table gives the description of the scenario:

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The construction worker selects Connect on the mobile phone. The device will discover the available receiving devices.</td>
</tr>
<tr>
<td>2</td>
<td>The mobile phone displays a list of Bluetooth devices in range, after exploring for new devices.</td>
</tr>
<tr>
<td>3</td>
<td>The construction worker selects which receiving device (the wearable PC) he wants to communicate with from the list.</td>
</tr>
<tr>
<td>4</td>
<td>The phone queries the SDP database to check that it is advertising the Serial Port service, and to extract necessary parameters of that service.</td>
</tr>
<tr>
<td>5</td>
<td>The mobile phone requests a serial port connection with the receiving device over Bluetooth.</td>
</tr>
</tbody>
</table>
6. Once the connection has been accepted, the status will be displayed on user interface of the phone.

7. The construction worker selects Query Task (or Report) on the mobile phone.

8. The PC sends back the corresponding tasks for the construction worker, or acknowledges the report.

9. The mobile phone displays the tasks, which it received.

10. The construction worker selects Exit to end connection.

The UML sequence diagram below shows the interactions that occur during this scenario. The diagram also shows certain states which each device has at different stages in the scenario. The state changes when certain events occur. These are typically as a result of a previous asynchronous request completing.

3.2 Sending Mechanism

Bluetooth sockets are used to discover other Bluetooth devices as well as to read and write data. Before using the sockets API to establish a connection, a Bluetooth device must locate a suitable device to connect to. This task is performed by Bluetooth Service Discovery Protocol (SDP). The SDP can be broken down to two main parts: the discovery of devices and services within local area, and the advertisement of services from the local device. Then the Bluetooth Service Discovery Agent enables to discover the Bluetooth services, and the attributes of those services, that are available on a remote device. Any service records held in the remote SDP database, which contain any one of these IDs will be returned during the search process.

Table 2: Description of Connection Calls

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-4</td>
<td>HandleCommandL is called on the</td>
</tr>
</tbody>
</table>

The UML class diagram below shows the class structure of the objects involved in sending a message in our application. It is based on the example provided by Nokia Series 60 SDK BTPointToPoint application.

3.2.1 Connection Process

The connection process begins when the user selects Connect on the sending device. This causes HandleCommandL to be called on the BTPointToPointAppUi object which in turn calls ConnectL on a CMessageClient object. The UML diagram shows in detail what happens when ConnectL is called on the CMessageClient object. The commentary in the table explains the function calls.
BTPointToPointAppUi object which in turn calls ConnectL on a CMessageClient object. This sets the iState variable to EGettingDevice and calls SelectDeviceByDiscoveryL on the CMessageServiceSearcher object. This in turn calls StartNotifierAndgetResponse on the RNotifier to discover and select the receiving device.

5-6 When the user has selected the receiving device from the list of devices that the RNotifier plugin displays, the framework calls RunL. Since iState equals EGettingDevice, RunL sets it to the next state, EGettingService, and calls FindServiceL on the CMessageServiceSearcher object to find the Serial Port service record in the receiving device's SDP database.

7-8 FindServiceL sets up a record filter to filter out just the Serial Port service record. It calls SetRecordFilter on the CSdpAgent to register the filter with the agent. It then calls NextRecordRequest to tell the agent to search the receiving device's SDP database for the Serial Port record.

9-10 When the right record has been found, the CSdpAgent calls the callback function NextRecordRequestComplete which in turn calls NextRecordRequestCompleteL on the CMessageServiceSearcher object. This calls AttributeRequestL on the CSdpAgent to request the first attribute of the record.

11 When the CSdpAgent finds an attribute it calls the callback function AttributeRequestResult which in turn calls AttributeRequestResultL on the CMessageServiceSearcher object. This sets a parser to parse the attribute. If the attribute is the Protocol Descriptor List, the parser will extract the channel (port) number and assign the CMessageServiceSearcher member variable iPort. AttributeRequestResultL will be called once for each attribute in the record. Once the Ownership of the aAttrValue parameter is transferred to the AttributeRequestResult function, this parameter will be destroyed after use.

12-16 When all the attributes have been found, AttributeRequestCompleteL will be called. This calls NextRecordRequest in case there is another Serial Port service record in the SDP database. If there are no more records, the CSdpAgent will call NextRecordRequestComplete with an EoF flag set. Finished will be called which in turn calls User::RequestComplete, so that the framework will call RunL again.

3.2.2 Communication Process

Now, the connection between the mobile phone and PC is established. The RunL in CMessageClient will be called and it changes the status to ‘connected’, and this will be displayed on phone screen. The user interface of the phone is responsible for handling the user selections. The UML diagram below shows the completion of connection process and the handling of user input. The commentary explains the function calls.

![Sequence Diagram of Handling the User Input](image)

Figure 5: Sequence Diagram of Handling the User Input

Table 3: Description of Communication Process

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-3</td>
<td>When RunL is called the value of the iState variable in the CMessageClient object is EGettingService. It is then updated to EGettingConnection and ConnectToServerL is called. This opens a socket and calls Connect on it to connect to the port on the receiving device.</td>
</tr>
<tr>
<td>4</td>
<td>When the connection attempt succeeds, RunL is called again, this time with iState set to EGettingConnection. iState is updated to ESendingMessage. Now the user can send message to the receiving device.</td>
</tr>
<tr>
<td>5-9</td>
<td>The user selects Query Task on the sending device. This causes HandleCommandL to call SendMessageL. This calls Write on the socket, which writes the message to the receiving device. The iState changes to EReceiving and causes IssueRead. This calls ReceiveOneOrMore on the socket, and displays the available received message.</td>
</tr>
</tbody>
</table>

3.3 Bluetooth Adaptor

The Bluetooth USB adapter used in this project is the DLink DBT-120. It is very flexible to configure. It has the driver to
handle client applications as well as providing local services. It also has settings for authentication, authorization, and encryption. In our case, since the connection is initialized by the client at mobile phone, the adapter side works as the server to provide the serial port service.

Figure 6: Bluetooth USB Adaptor
We enable the Bluetooth Serial Port service at the adapter side, no authentication, authorization, and encryption is chosen. The communication port in our case is chosen to be COM 3. So the Java application running in the wearable PC will read data through COM3. Then a wireless connection through virtual communication ports between the wearable PC and our mobile phone is established, as if there is a physical serial cable connected the devices.

The application program running at PC side currently only reads and identifies the simple incoming messages. For example, if the construction worker selects Query Task, then the response will be his tasks of the day. And if the worker selects Report, then the program only acknowledges the reception of message, and no evaluation or other comments will be fed back.

4 Testing and Results

The Bluetooth applications can be created the same way as any other Symbian C++ applications using SDK 1.0. However, it can be tested only partially with the emulator; the final testing must be carried out with the target devices. Since the performance is different and there may be some latency on the emulator when compared to the phone. So we carried out all the tests on the real series 60 phone, Nokia 7650.

4.1 Main Problems Encountered

1. Since the testing is done at real phone, each time we must make the sis file and download to the phone. However, the new created file didn’t totally replace the original application. After some consulting, we found it was due to we didn’t erase all the intermediate files created during compilation and all the executables and import libraries created by the linker. Then this is done by the clean commands.

2. At first stage of testing, the connection between the phone and the adapter failed to be established. This was because of the setting of the communication port for the Serial Port service. When the adapter accepts an incoming connection, it works as the server, and the virtual port is assigned to be COM 3. When it imitates an outgoing connection request to a remote machine, the port is different. Our case is the former one.

3. When running on the phone, some times the application automatically closed and returned error code. After trailing, we found the problem was due to the requests generated by the active object, which should not be handled. So we modified the CMessageClient class, and made it had only one outstanding request at a time. Then this works fine.

4. In the Bluetooth communication, distance problem is another testing consideration. We found that when the two devices were very close to each other, with 0.5 meters, the signal was very strong. When the range increased to 5 meters, the signal tended to decrease. However, it didn’t affect the actual sending and receiving, since the speed we got is satisfactory, and all the data was as expected.

4.2 Results

The user at mobile phone side initiates the connection process. As response, it will first prompt the user to choose a device or to discover all devices in range. The discovery process takes about 5 seconds, and then the available devices will be listed on the screen. After the user selects a specific device, the phone will automatically carry out the connection process. The actual connection is very fast, within 1 second, and it returns the status, whether ‘connected’ or ‘connection failure’. Once the connection is established, user can select Query Task, or Report to enter text messages. The program running at PC side is responsible for handling the incoming messages and giving feedback. The communication is bi-directional. Some user interfaces of the mobile phone are shown below:

Figure 7: Samples of User Interface

5 Conclusion and Future Work

In this project, we developed our application in Series 60 Platform. We got familiar with the Symbian OS programming
Currently, there have already been a lot of Bluetooth-enabled devices. For example, the Bluetooth headsets, Bluetooth keyboard and mouse. More and more people appreciate the usage of Bluetooth technology. Also, the communication between the mobile phone and PC using Bluetooth can be carried out in various ways. In our case, it is just simple text message. In future work, we may want to transfer the image file and so on.

So far in the project, the Bluetooth connection is always initiated or ended by the mobile phone. It is also possible for the PC side to start and end the message exchange. And it will be carried out in the similar way.

Since this UROP is part of the large work, so currently we assume the sequence and content of messages. In future work, we will do further improvement to make it easier to use. For example, the Report part, user doesn’t need to make text input to generate the report. He can just make the selection, and the application program will retrieve the corresponding reports from phone’s memory and send it to PC. Other functionality may also be added. Besides these, the improvement will also need for the user interface to handle the corresponding updates.

Our next step is to explore more on the Symbian OS and the Bluetooth. In the future work, the phone will not only exchange data with PC, it can also control other peripherals, for example, the mouse.

Acknowledgements

I would like to take this opportunity to thank Nokia Singapore Pte Ltd for sponsoring us with the mobile phone.

References


