HOME SECURITY SYSTEM USING SYMBIAN SMARTPHONE AND BLUETOOTH COMMUNICATION PROTOCOL

Sagar Sonar
B.E., University of Pune, India, 2006

PROJECT

Submitted in partial satisfaction of the requirements for the degree of

MASTER OF SCIENCE

in

ELECTRICAL AND ELECTRONIC ENGINEERING

at

CALIFORNIA STATE UNIVERSITY, SACRAMENTO

FALL 2010
HOME SECURITY SYSTEM USING SYMBIAN SMARTPHONE AND BLUETOOTH COMMUNICATION PROTOCOL

A Project

by

SAGAR SONAR

Approved by

__________________________, Committee Chair
Jing Pang, Ph.D.

__________________________, Second Reader
Fethi Belkhouche, Ph.D.

__________________________
Date
Student: Sagar Sonar

I certify that this student has met the requirements for format contained in the University format manual, and that this project is suitable for shelving in the Library and credit is to be awarded for the Project.

_______________________, Graduate Coordinator  
Preetham Kumar, Ph.D.  
Date

Department of Electrical and Electronic Engineering
In today’s world, smartphone has become one of the majorly used electronic devices with millions of user all around the world. The smartphones provide plenty of features like large memory, different types of user interface, GPRS, GSM, BLUETOOTH, etc. We can make use of some of these features in commercial world. The best way to use it will be for home security and control systems. Home security is a major concern nowadays because of the increase in crime rate. Also, control system is very important to make life easier. In this project, I am using smartphone to provide home security and control system in better way. I am using smartphone to turn ON and OFF the alarms and other control devices and to get notification if any intruder enters in the house.

The aim of this project is to develop an application for smartphone. I have chosen smartphone with Symbian operating systems because it is one of the most popular smartphone operating system in the world. The programming language used to develop
the application is Python. Python is an open source developing language and has very rich features and library functions. Python is easy to use and helps to reduce the code size due to its inbuilt functions. I will be using Bluetooth communication protocol for wireless communication with the hardware built using micro-controller which will control the devices installed in the home. I have chosen Bluetooth because it is available in all the smartphones and have many advantages.

The developed application will help the user to control the operation of their security and control devices using their smartphone, eliminating the use of any other hardware or remote controls. Nowadays smartphones are easily available in the market with very low cost, if we design the same hardware using microcontroller and other peripherals such as keypad, LCD, wireless module, etc. It will cost much more than one smartphone handset and will also consume lot of time in designing, implementing hardware and testing its functionality, also the size will not be so small, hence in this project I have considered cost and time as major factors.

_______________________, Committee Chair  
Jing Pang, Ph.D.

_______________________  
Date
Firstly, I heartily thank my project advisor Dr. Jing Pang for providing me this wonderful opportunity to work on this project and giving me a great exposure to the field of embedded systems and mobile application programming. This project would not have been possible without the continuous support and encouragement of Dr. Pang. Dr. Pang was always there to listen, to give me advice and to share her knowledge and expertise in the field of embedded system and mobile application programming with me. I would also like to thank Dr. Pang for always being patient and helpful towards the successful demonstration of application, interfacing and communication of the smartphone with embedded hardware.

I would also like to thank second reader of my project, Dr. Fethi Belkhouche, for reviewing my report and providing me his valuable guidance and useful advice to improve the organization and flow of my project report.

Finally, the continuous strength and warmth I got from my family members and friends were valuable and they drove me on through the critical phase of the project.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>iv</td>
</tr>
<tr>
<td>Acknowledgement</td>
<td>v</td>
</tr>
<tr>
<td>List of Figures</td>
<td>ix</td>
</tr>
<tr>
<td>Chapter 1. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>2. SYMBIAN OPERATING SYSTEM</td>
<td>3</td>
</tr>
<tr>
<td>2.1 Introduction to Symbian OS for Smartphones</td>
<td>3</td>
</tr>
<tr>
<td>2.2 Advantages of Symbian</td>
<td>5</td>
</tr>
<tr>
<td>2.3 More about Series 60 Platform</td>
<td>6</td>
</tr>
<tr>
<td>3. BLUETOOTH COMMUNICATION PROTOCOL</td>
<td>7</td>
</tr>
<tr>
<td>3.1 Introduction to Bluetooth Communication Protocol</td>
<td>7</td>
</tr>
<tr>
<td>3.2 Advantages and Disadvantages of Bluetooth</td>
<td>9</td>
</tr>
<tr>
<td>3.3 Limitations of Bluetooth</td>
<td>10</td>
</tr>
<tr>
<td>4. APPLICATION DESIGN AND IMPLEMENTATION</td>
<td>11</td>
</tr>
<tr>
<td>4.1 Application Design Concept</td>
<td>11</td>
</tr>
<tr>
<td>4.2 Embedded Hardware</td>
<td>12</td>
</tr>
<tr>
<td>4.3 Application Design Flow</td>
<td>12</td>
</tr>
</tbody>
</table>
4.4 Screenshots of Smartphone Application..................................................14

4.5 Application Design..................................................................................17
  4.5.1 POP-UP Menu Design........................................................................19
  4.5.1 Bluetooth Communication.................................................................19
  4.5.3 Graphics Design..................................................................................22

4.6 Python Script to .SIS Conversion.............................................................25

5. LIMITATIONS AND FUTURE WORK......................................................27

6. CONCLUSION ..........................................................................................29

Appendix A  Source Code for Application.....................................................30

References.....................................................................................................41
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Fig 2.1 Symbian OS architecture</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>Fig 2.2 Series 60 Platform architecture</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>Fig 3.1 Bluetooth protocol architecture</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>Fig 4.1 Application design concept</td>
<td>11</td>
</tr>
<tr>
<td>5</td>
<td>Fig 4.2 Design flow</td>
<td>13</td>
</tr>
<tr>
<td>6</td>
<td>Fig 4.3 Snap shot of Pop-up menu selection screen</td>
<td>14</td>
</tr>
<tr>
<td>7</td>
<td>Fig 4.4 Snap shot of Next pop-up menu selection screen</td>
<td>15</td>
</tr>
<tr>
<td>8</td>
<td>Fig 4.5 Snap shot of Available Bluetooth devices screen</td>
<td>15</td>
</tr>
<tr>
<td>9</td>
<td>Fig 4.6 Snap shot of Waiting for acknowledgement screen</td>
<td>16</td>
</tr>
<tr>
<td>10</td>
<td>Fig 4.7 Snap shot of Graphical representation of data screen</td>
<td>17</td>
</tr>
<tr>
<td>11</td>
<td>Fig 4.8 Steps to follow to built an application</td>
<td>17</td>
</tr>
<tr>
<td>12</td>
<td>Fig 4.9 Application block diagram</td>
<td>18</td>
</tr>
<tr>
<td>12</td>
<td>Fig 4.10 Bluetooth function design flow</td>
<td>19</td>
</tr>
<tr>
<td>13</td>
<td>Fig 4.11 Graphic function design flow</td>
<td>21</td>
</tr>
</tbody>
</table>
Chapter 1

INTRODUCTION

Due to the advances in smartphone features and its low cost, smartphone have become one of the most commonly used handheld devices. Smartphone of nowadays offer many features and applications. However many of the applications are only for entertainment purpose. Due to the popularity of smartphones and the features they offer we can make use of these devices in many other ways. One way was developing the application for smartphone to control the operation of home security and control system. Home security is very important nowadays and if we can add this feature in our smartphones by developing application it will be very useful. It will provide new concept and idea to the smartphone application developers. Considering these factors I decided to develop the application for Series 60 smartphone for home security and control system.

As the first phase of project, I started with researching about the most commonly used smartphone, its operating system, provided features and understanding the architecture of the hardware and operating system. Next step was to do research on the programming languages used to develop application for smartphone. My aim was to choose the simple language which will be easy to understand and in a short learning period. Next step was to do research on communication protocols, since all common smartphones have Bluetooth already built in. I went for Bluetooth as a communication protocol and studied more about it. Next step was to study graphics representation on smartphone and show
some data in graphical representation because it is easier to visualize data. I captured data from user and displayed it in graphical form (Histogram). Considering all above factors, I started developing the application for Symbian OS Series 60 smartphone device using Python as the mobile programming language and Bluetooth as a communication protocol to communicate between smartphone and hardware.

Chapter 2 describes Symbian operating systems in more details. It describes about the Symbian OS architecture, features and concepts. It also focuses on the advantages of Symbian OS and also introduces us with Series 60 platform.

Chapter 3 focuses on Bluetooth communication protocol and gives more information about it specifications, advantages and disadvantages.

Chapter 4 describes the application design flow, approach, detail functioning of each module such as pop-up menu, Bluetooth communication and graphic. It also describes the procedure of converting .py script to .sis application.

Chapter 5 describes the limitations of the application and the future enhancement that can be done such as adding more flexibility and features to the application.

Finally, Chapter 6 offers a conclusion on the project.
Chapter 2

SYMBIAN OPERATING SYSTEM

2.1 Introduction to Symbian Operating System

Symbian was the first platform specifically developed for smartphones. It started the smartphone revolution and remains at the heart of the movement, powering more than two-fifths of the world’s smartphones. The Symbian platform is used by more people in more places than any other mobile OS. This makes Symbian the most popular smartphone platform on the planet. The Symbian platform gives you the power to do more with mobile. The platform is used to produce every type of handset from compact feature phones to high-end smartphones [7]. In fact, it is inside many of the cutting-edge mobile phones you see on the street today.

Symbian platform is Open Source [7], meaning that absolutely anyone can use it to design and build any type of mobile device they want. This open source approach puts the platform into the hands of the innovators who want to create beautiful products and launch compelling services using the Symbian platform.

The Symbian operating system offers a broad array of personal information management (PIM) features, including contact and calendar management and a robust library of third-party applications. We can develop and install third party application very easily on the Symbian OS. The programming language used for developing Symbian applications are C++, Java and Python. Once application is developed, Symbian applications need to be
transmitted and installed to the mobile phones. They are packaged in .sis files which may be installed via PC connect, Bluetooth or on a memory card. The .sis file is installed using Symbian installer and a new icon is created in the mobile phone menu with the given application name.

The Symbian OS and application follows object oriented programming concept. The concept used by Symbian is Microkernel which helps in request and callback approach to service and also helps in maintaining separation between user interface and engine. The Symbian OS is optimized for ROM based and low power system. The OS follows the layer structure. It is designed using five layers: UI framework layer, application services layer, OS services layer, base services layer and kernel services and hardware service layer [8].
2.2 Advantages of Symbian OS

The Symbian OS has the ability to take it and build something truly special and unique. No other platform is structured in the same way. The Symbian platform is self-contained and fully accessible, allowing user to bring to life new services and ideas in mobile product design. The main advantages are open source, power management, multitasking and robustness.
Symbian OS is a complete and customizable platform. It is a fully adaptable platform designed specifically for modern mobile communication, to create everything from low-cost feature rich phones to high end smartphones.

It provides total UI design flexibility. It offers a complete touch screen solution with effortless multitasking, full platform theming and home screen customization, giving real design flexibility and huge scope for differentiation.

Symbian OS offers a better network management. It has simple connection management providing easy and secure access to the Internet and Bluetooth - just select the networks that work best when you want to connect and the platform will help get you the most from whichever network is available to you.

Symbian OS provides real power to communicate. The core platform gives you unrivaled power efficiency to use the web and communicate using the super-fast browsing experience, built-in video telephony and integrated social networking.

2.3 Series 60 Platform

Series 60 platform was launched by Nokia in November 2001 which provided licenses with the ability to make advanced mobile applications [1]. Series 60 platform is built on Symbian OS complementing it with a graphical user interface library and reference applications. Series 60 is a platform on which we can build our own feature rich
terminals. It is easy to use and includes a rich suite of applications. It includes many Symbian components such as data management, security, messaging, browser engines and Bluetooth.

Fig. 2.2 Series 60 Platform architecture [4]

The above figure gives more detail on the Series 60 platform architecture. It is built on Symbian OS with some more added features.
Chapter 3

BLUETOOTH COMMUNICATION PROTOCOL

3.1 Introduction to Bluetooth Communication Protocol

A Bluetooth connection is wireless and automatic, and it has a number of interesting features that can simplify our daily lives. The Bluetooth standard requires a basic level of communication between devices, so that they can connect to each other over the airwaves, at the correct frequencies, using the correct channels, and finding the correct destination(s). In order to create this basic level of communication, a specific protocol was created. The Bluetooth protocol establishes the set of rules by which all Bluetooth devices must abide in order to establish a connection to communicate with one another. Bluetooth networking transmits data via low-power radio waves. It communicates on a frequency of 2.45 gigahertz (actually between 2.402 GHz and 2.480 GHz, to be exact). This frequency band has been set aside by international agreement for the use of industrial, scientific and medical devices (ISM).

When Bluetooth capable devices come within range of one another, an electronic conversation takes place to determine whether they have data to share or whether one needs to control the other. The user doesn't have to press a button or give a command -- the electronic conversation happens automatically. Once the conversation has occurred, the devices -- whether they're part of a computer system or a stereo -- form a network. Bluetooth systems create a personal-area network (PAN), or piconet, that may fill a room or may encompass no more distance than that between the smartphone on a belt-clip and
the headset on your head. Once a piconet is established, the members randomly hop
frequencies in unison so they stay in touch with one another and avoid other piconets that
may be operating in the same room.

Most protocols, Bluetooth’s included, are usually layered together into “protocol stacks”,
and the various tasks are split up and assigned to the different layers of protocols in the
stack. The following figure shows Bluetooth protocol stack architecture [9].

Fig 3.1 Bluetooth protocol architecture [9]
3.2 Advantages of Bluetooth

Bluetooth provides high security. One of the ways Bluetooth devices avoid interfering with other systems is by sending out very weak signals of about 1 milliwatt. By comparison, the most powerful smartphones can transmit a signal of 3 watts. The low power limits the range of a Bluetooth device to about 100 meters, cutting the chances of interference between your computer system and your portable telephone or television.

Bluetooth does not need line of sight needed. Even with the low power, Bluetooth doesn't require line of sight between communicating devices. The walls in your house won't stop a Bluetooth signal, making the standard useful for controlling several devices in different rooms.

Bluetooth can connect multiple devices. It can connect up to eight devices simultaneously. Bluetooth uses a technique called spread-spectrum frequency hopping that makes it rare for more than one device to be transmitting on the same frequency at the same time. In this technique, a device will use 79 individual, randomly chosen frequencies within a designated range, changing from one to another on a regular basis. In the case of Bluetooth, the transmitters change frequencies 1,600 times every second, meaning that more devices can make full use of a limited slice of the radio spectrum. Since every Bluetooth transmitter uses spread-spectrum transmitting automatically, it’s unlikely that two transmitters will be on the same frequency at the same time. This same technique minimizes the risk that portable phones or baby monitors
will disrupt Bluetooth devices, since any interference on a particular frequency will last only a tiny fraction of a second.

Communication using Bluetooth wireless technology uses less power than a traditional wireless LAN connection.

### 3.3 Limitations of Bluetooth

One of the limitations of Bluetooth is its limited range. Due to low power the range of Bluetooth connectivity is limited to 100 meters.

Another limitation of Bluetooth is speed. Bluetooth is not designed for high-speed data transfers.

Bluetooth is susceptible to interference. Instead of using frequency hopping technology it has seen that Bluetooth sometime do not withstand the frequency hopping which makes it much more open to interception and attacks.
4.1 Application Design Concept

As shown above in Fig 4.1, the application is installed on smartphone. From PC we can download and install application on smartphone. We can also accept files from smartphone and store it on PC. Smartphone communicates with embedded hardware through the designed application. User will select the operation to be performed on devices from smartphone. The particular command will be transmitted to the embedded hardware. Once the command is received by hardware, hardware will perform the
operation on the selected device and will also send ack command back to smartphone. If smartphone receives the ack it will go back to the pop-up menu screen.

4.2 Embedded Hardware

Embedded hardware is designed to control the home security systems and other devices. The hardware also communicates with smartphone application using Bluetooth communication. For designing embedded hardware we have to select the microcontroller and other peripheral components, design PCB and implement all components.

Embedded hardware accepts the commands from the smartphone application and will perform the selected operation. Once the hardware receives the command from smartphone, the hardware sends ACK back to smartphone, informing the successful reception of command. Embedded hardware can never initiate the transaction because smartphone will always keep its Bluetooth off to save power. Embedded hardware can communicate with smartphone only after reception of command.

We can always keep hardware in low power mode and will wake it up by interrupt on its serial port from Bluetooth. This will save lot of power and will make the hardware power efficient.

4.3 Application Design Flow

For designing a smartphone application we have followed a flowchart. The application flow is as shown below in Fig 4.2.
As in flow chart above, whenever the application is started, it will display the list of the devices we want to control using the application. The list is stored in the application for
now but in future we can create a database and can store the list of devices in the database. User can easily add more devices in the database and can also remove devices. Application waits for the user to select the device he wants to control for the list. Once the device is selected user has to select the particular operation (ON/OFF) for the selected device. Once the operation is selected the application will turn on the Bluetooth and will search for all the available Bluetooth devices in the range. Once searching is done the application displays the list of the Bluetooth devices and waits for the user to select the Bluetooth device. After user has selected the Bluetooth device the application transmits the particular command related to the operation to be performed and waits for the ack for the Bluetooth device. If application does not receive ack, it will send the command once again, application will try three attempts. If it still does not receive ack application will display and transmit saying “unable to connect” and will display the device list so that user can now select some other device. If user does not want to select any other device he can select exit and application will end.
4.4 Screenshots of Smartphone Application

Fig. 4.3 pop-up menu selection

Fig 4.3 shows the list displayed on the smartphone. User will select any one device from the displayed list. User can scroll down the list using the joystick. To select device user will have to press OK.

Fig. 4.4 Next pop-up menu selection

Fig 4.4 shows the next pop-up menu selection screen. Once the selection is done the application will display next pop-up menu screen. Application will wait for the user to
select the operation to be performed. If operation is not selected then the application will stay on that screen.

Fig. 4.5 List of available Bluetooth devices

Fig 4.5 displays the list of all available Bluetooth devices in the range. Once the operation is selected, Bluetooth socket is enabled and displays the list of all the available devices in the range. User will have to select the Bluetooth device it wants control. If user has selected some wrong device he can select exit and the application will be closed and no operation will be performed.

Fig. 4.6 Waiting for ACK
Fig 4.6 displays the application waiting for ACK. Once Bluetooth device is selected, application will transmit the specific command and will wait for acknowledgement from the hardware. If there is no acknowledgement from the Bluetooth hardware then the command is sent again, the application will try 3 attempts.

Fig 4.7 Graphical representation

Fig 4.7 shows the graphical representation of the data received for user. In future we can display the real time data with respect to time. We can also store the data in the file and can later on transmit the file to PC using Bluetooth.
To build Symbain smartphone application we need to follow the following steps [2]

1. Import all needed modules

2. Set the screen size (normal, large, full)

3. Create your application logic

4. Create an application menu (if necessary)
5. Set an exit key handler

6. Set the application title

7. Deal with active objects if necessary

8. Set the application body (text or canvas or list box or none)

9. Create a main loop (e.g. while loop) if suitable

By following the above algorithm, I started building my application.

Fig 4.9 Application block diagram

Fig 4.9 represents the application block diagram. Main function calls other functions.

Main function defines the application title and exit key handler.
4.5.1 POP-UP Menu Design

The simplest application to design is the pop-up menu. The sample code for developing pop-up menu is as follows:

```python
import appuifw
# create a list with the content of the pop-up no
L = [u"Python", u"Symbian", u"Mlab"]

# create the pop-up menu including the list of content and a label
# - -> appuifw.popup_menu(list , label)
test = appuifw.popup_menu(L, u"Select + press OK ")

# here we print something
if test == 0
    appuifw.note(u"Python, yeah", "info")
elif test == 1
    appuifw.note(u"Symbian, ok", "info")
elif test == 2
    appuifw.note(u"Mlab, cool", "info")
```

It displays the list of pop-up menu and wait for selection from the user. The application will display the information related to the each selection.

4.5.2 Bluetooth Communication

Since we need to transmit and receive data to and from other hardware, I have used Bluetooth as a communication protocol, because it is available in almost all the smartphones and it has many advantages as discussed before. I have developed a function to create and open a Bluetooth socket and search for the devices within the range. For enabling Bluetooth feature we need to import socket module. When the function is called it will turn on the Bluetooth socket if it was OFF before. Then it will search for the
available Bluetooth devices in the range and display the list of the Bluetooth device. Next time it will display all the previously searched devices so that it doesn’t have to search all of them again, this will save power and time needed in searching the Bluetooth devices.

To design Bluetooth function I first develop a flowchart and followed that particular flowchart. The flowchart is as shown in Fig 4.10.

According to flowchart I first create Bluetooth socket and turn on the Bluetooth communication port on the smartphone. Once Bluetooth is enabled, it will scan for all the other available devices in the range and will display the list of the devices.

Another function is called after the selection the selection of Bluetooth device. This function will transmit the command to the selected Bluetooth device and will wait for the “ack” from that Bluetooth device. If it does not receive “ack” the command and send again and after 3 attempts the Bluetooth socket is turned off. If “ack” is received the Bluetooth socket is turned off.
The function for Bluetooth communication is as follows

Fig. 4.10 Bluetooth function design flow
def bt_connect():
    global sock
    # create a Bluetooth socket
    sock=socket.socket(socket.AF_BT, socket.SOCK_STREAM)
    target=""  # here you can give the bt address of the other mobile if you know it
    if not target:
        # scan for Bluetooth devices
        address, services = socket.bt_discover()
        print "Discovered %s, %s"%(address, services)
        if len(services)>1:
            choices=services.keys()
            choices.sort()
            # bring up a popup menu and show the available bt devices for selection
            choice=appuifw.popup_menu([unicode(services[x]) + " +x"
                                        for x in choices], u'Choose port ')
            target=(address, services[choices[choice]])
        else:
            target=(address, services.values()[0])
        print "Connecting to " + str(target)
    # connect to the serial port of the PC
    sock.connect(target)
    print "OK."

We can also scan the available Bluetooth devices in background but it needs additional applications to be installed on the phone since those libraries are not available in the original Python for Series 60 package.

4.5.3 Graphics Design

Data displayed in graphical format conveys the information in an easy way. I have shown that, we can design an application to capture data from user or other hardware and display it in graphical representation. For achieving this I have developed a function to collect data and display it as histogram. The function imports graphics library. Function defines
the canvas size, scale, axes and then plots the graph. The flowchart that I followed to develop graphics function is as below.

![Flowchart](image)

Fig 4.11 Graphic function design flow
We should also select the maximum range of the Y axis so that it accordingly selects the scale of the graph depending on the size of the screen. The function graphics design is as follows

```python
def draw():
    # Get number of graphs
    t = appuifw.query(u"Enter Number Of Graphs (Max 9)"), "number")

    # Get range
    rrng = appuifw.query(u"Maximum Range Of Graph ", "number")

    # Graph colors
    colors = [0x0000ff, 0x00ff00, 0xff0000, 0xffff00, 0xc0c0c0, 0x808040, 0xff00ff, 0xf0cb75, 0xbe6a0e]

    # Get graph prameters
    y = []
    n = []
    for j in range(t):
        (n1, y1) = appuifw.multi_query(u"Title Of Graph %d "%(j+1), u"Height Of Graph %d "%(j+1))
        n.append(n1)
        y11 = int(y1)
        y.append(y11)

    # Define canvas
    canvas = appuifw.Canvas()
    appuifw.app.body = canvas
    xx, yy = canvas.size

    # Scaling Graphs according to screen size
    arrg = float(yy - 50)
    mf = arrg / rrng

    # Draw x,y axis
    canvas.line((20, yy-20, xx-32, yy-20), 0)
    canvas.line((xx-33, 10, xx-33, yy-20), 0)

    # Put indicator points
    canvas.text((xx-34, 30+(yy-40)*0.75), u"_%d"%(rrng*0.25))
    canvas.text((xx-34, 30+(yy-40)*0.5), u"_%d"%(rrng/2))
```
canvas.text((xx-34,30)+(yy-40)*0.25,u"_%d"%(rng*0.75))
canvas.text((xx-34,30),u"_%d"%(rng))

# Draw graphs
h = 30
for i in range(t):
    w=(xx-57)/t - 10
    s = yy-20-mf*y[i]
    canvas.rectangle((h,s,h+w,yy-20), outline=0x000000, fill=colors[i])
    x11 = str(y[i])
    canvas.text((h,(s-5)),u" %s"%(x11), fill=colors[i])
    canvas.text((h,yy), u"%s"%(n[i]), fill=0x4b2192)
    h=h+w+10

As shown above the function will take range of graph as an input and will accordingly define the canvas and scale the other data points to plot on the screen. The function accepts the data points from user and stores it in array. Those data points are used for plotting the graph. Graph colors are given by using different color codes.

In future we can display graphical wave by taking data Vs time. For doing this we have to import graphics libraries called as matplotlib. This will allow us displaying and plotting graphs in various different formats such as sine wave, pie chart, etc.

4.6 Python (.py) to Standalone Application (.sis) Conversion

Currently the script/application that is designed requires a Python installed on the smartphone. If we want to install this application to another smartphone then we will first have to install Python and then will have to run the application through Python interpreter. This is useful while application development process, but once application is ready we would like to create it as a standalone application. To make the application as a
standalone, we will have to convert the .py file into .sis standalone application format for smartphone. This .sis application software can be transferred to the smartphone using Bluetooth or any other communication mode.

A standalone application lets you start your own "Python application" directly by clicking on an icon on the desktop of the phone (not having to open the Python interpreter, "run script" etc. anymore). To make standalone .sis application file we need to follow the below given procedure

First of all we need to have the following software installed on machine

A. Symbian SDK, to get the SymbianSDK, go to www.forum.nokia.com/main/0,034-4,00.html

B. Python plug-in for the SDK

After that we need to install the Python plugin Python_for_2ndEd_SDK.exe. It is included in the PythonForSeries 60_1_2_for_2ndEd_SDK.zip file, run the downloaded file to install all the plugins needed for py2sis tool. It creates the following directory where it puts relevant files. Now we are set to create a standalone application (creating a *.sis file). On PC, create a folder named e.g. myapps (C\myapps). This is our working directory. Then save our Python script e.g. named test.py which we want to turn into a standalone app inside the myapps folder. Open a command line terminal.
On the command line terminal go to the working directory myapps. On the command line terminal type the command `py2sis test.py myfirst.sis --uid=0xFFFFFFFF --appname=myfirst`.

In this command `py2sis` is an executable that does the file conversion, the rest are parameters. Now we should be able to run the `py2sis` command successfully. This will create the `myfirst.sis` standalone application in the same working direction, now we should be able to install and this application on any smartphone without any need of Python installed on the smartphone.
Chapter 5

LIMITATIONS AND FUTURE WORK

Since the programming language that is used for this project is Python, it supports only Symbian S60 platform. This application can only be used on Symbian S60 or higher Symbian OS. This application will not work for S40 devices. S40 support JAVA and Carbide C++, Symbian3 supports Carbide C++ and S60 supports Python and JAVA. In the future, we have to develop the application in JAVA to make it compatible with all the Symbian devices available in the market, but this language is not an open source and will require licensing to begin with it.

Currently, this application supports only Series 60 devices and if I want to add more devices I have to change the source code. In the future, I would like to add or delete dynamically, this will also allow the other user’s to add or delete the device of their choice. To add this feature I will have to create local database using SQL scripts and queries. By creating database all the devices will be stored locally in the database and will be called upon request, also user can easily add or delete more number of devices. This will require more knowledge of SQL queries and database management.

Graphics function can now only display data in form of histogram and take the data from user, I would like to enhance the function by keeping log of all the operation performed within particular amount of time and displaying them in many other forms such as sine waves, pie chart, etc for this we will need to add more number of library and latest feature pack for S60 and latest S60 smartphone.
We are now using Bluetooth communication for communicating between control devices, other hardware and smartphone but Bluetooth has very short range and if smartphone goes out of the range then we will not be able control the devices and also if someone breaks security system we will not be able to get information, hence in future I will like to use GSM or GPRS as a communication mode.
Chapter 6

CONCLUSIONS

The aim of this project was to make use of smartphone in different way other than communication and entertainment purposes. Since nowadays smartphones are very common and available for small cost we can use these devices for other control purposes by developing some application. Currently security is very important issue and people are paying lots of money for additional hardware to control the home devices and security system. All the hardware is either wired or requires line of sight communication and on the top of that it is difficult to carry additional remote control device to control the security system. There is a better way of doing this by using available smartphones.

In this project, I have successfully developed the application that can be installed on all Series 60 smartphones to control the security system and other devices in the home. The communication is performed using Bluetooth communication protocol, so it is wireless and does not require any line of sight communication. In future we can make this application more generic by adding more features and adding local database as suggested and also can make this application available for all the Symbian devices by using different programming languages.
import appuifw
import socket, graphics, key_codes
import e32

ECHO = True
# function to plot graph
def draw():

    # Get number of graphs
t = appuifw.query(u"Enter Number Of Graphs:", "number")

    # Get range
rrng = appuifw.query(u"Maximum Range Of Graph:", "number")

    # Graph colors
colors = [0x0000ff, 0x00ff00, 0xff0000, 0xffff00, 0xc0c0c0, 0x808040, 0xff00ff, 0xf0cb75, 0xbe6a0e]

    # Get graph parameters
y = []
n = []
for j in range(t):
    (n1, y1) = appuifw.multi_query(u"Title Of Graph %d:" % (j + 1), u"Height Of Graph %d:" % (j + 1))
    n.append(n1)
    y11 = int(y1)
y.append(y11)

    # Define canvas
canvas = appuifw.Canvas()
appuifw.app.body = canvas
xx, yy = canvas.size

    # Scaling Graphs according to screen size
arrg = float(yy - 50)
mf = arrg / rrng

    # Draw x,y axis
canvas.line((20, yy-20, xx-32, yy-20), 0)
canvas.line((xx-33, 10, xx-33, yy-20), 0)

# Put indicator points
canvas.text((xx-34, 30+(yy-40)*0.75), u"_%d"%(rrng*0.25))
canvas.text((xx-34, 30+(yy-40)*0.5), u"_%d"%(rrng/2))
canvas.text((xx-34, 30+(yy-40)*0.25), u"_%d"%(rrng*0.75))
canvas.text((xx-34, 30), u"_%d"%(rrng))

# Draw graphs
h = 30
for i in range(t):
    w = (xx-57)/t - 10
    s = yy-20-mf*y[i]
canvas.rectangle((h, s, h+w, yy-20), outline=0x000000, fill=colors[i])
x11 = str(y[i])
canvas.text((h, (s-5)), u" %s"%(x11), fill=colors[i])
canvas.text((h, yy), u"%s"%(n[i]), fill=0x4b2192)
h=h+w+10

# function that handles the bluetooth connection:
def bt_connect():
    global sock
    # create a bluetooth socket
    sock=socket.socket(socket.AF_BT, socket.SOCK_STREAM)
target= "" # here you can give the bt address of the other mobile if you know it
    if not target:
        # scan for bluetooth devices
        address, services=socket.bt_discover()
        print "Discovered: %s, %s"%(address, services)
        if len(services)>1:
            choices=services.keys()
            choices.sort()
            # bring up a popup menu and show the available bt devices for selection
            choice=appuifw.popup_menu([unicode(services[x])+": "+x for x in choices], u'Choose port:')
            target=(address, services[choices[choice]])
        else:
            target=(address, services.values()[0])
        print "Connecting to "+str(target)
    # connect to the serial port of the PC
    sock.connect(target)
    print "OK."
# function to read from rx buffer and echo it
def read_and_echo(fd):
    buf = r = ""
    while r != "\n" and r != "\r":
        r = fd.read(1)
        if ECHO: fd.write(r)
        buf += r
    if ECHO: fd.write("\n")
    return buf

# function to send and receive and check for ACK
def send_command(str1):
    print >> to_peer, str1 + "\r"
    print "Sending: " + str1
    print "Waiting for Ack"
    appuifw.note(u"Waiting for Ack", "info")
    e32.ao_sleep(2)
    reply = read_and_echo(to_peer).strip()
    appuifw.note(unicode(reply), "info")
    if reply.find("ack") != -1:
        return 1

def exit_key_handler():
    script_lock.signal()
    appuifw.app.set_exit()

def graph():

    # Get number of graphs
    t = appuifw.query(u"Enter Number Of data points", "number")

    # Get range
    rrng = appuifw.query(u"Maximum Range Of data", "number")

    # Graph colors
    colors = [0x0000ff, 0x00ff00, 0xff0000, 0xffff00, 0xc0c0c0, 0x808040, 0xff00ff, 0xf0cb75, 0xbe6a0e]

    # Get graph parameters
    y = []
    n = []
    bt_connect()
    to_peer = sock.makefile("rw", 0)
    appuifw.note(u"Waiting for Data", "info")
e32.ao_sleep(2)
for j in range (t):
    n1 = j
    print "Enter data"
    data = read_and_echo(to_peer).strip()
    y1 = data
    j = j+1
    n.append(n1)
    y.append(y1)
    y11 = int(y1)
    y.append(y11)

# Define canvas
canvas=appuifw.Canvas()
appuifw.app.body=canvas
xx,yy = canvas.size

# Scaling Graphs according to screen size
arrg = float(yy-50)
mf = arrg/rrng

# Draw x,y axis
canvas.line((20,yy-20,xx-32,yy-20), 0)
canvas.line((xx-33,10,xx-33,yy-20),0)

# Put indicator points
canvas.text((xx-34,30+(yy-40)*0.75),u"_%d"%(rrng*0.25))
canvas.text((xx-34,30+(yy-40)*0.5),u"_%d"%(rrng/2))
canvas.text((xx-34,30+(yy-40)*0.25),u"_%d"%(rrng*0.75))
canvas.text((xx-34,30),u"_%d"%(rrng))

# Draw graphs
h = 30
for i in range(t):
    w=(xx-57)/t - 10
    s = yy-20-mf*y[i]
    canvas.rectangle((h,s,h+w,yy-20), outline=0x000000, fill=colors[i])
x11 = str(y[i])
canvas.text((h,(s-5)),u" %s"%(x11), fill=colors[i])
canvas.text((h,yy), u"%s"%(n[i]), fill=0x4b2192)
h=h+w+10

appuifw.app.title = u"Home Security System"

def list():
global sock
global to_peer
i=0
L = ["Fans", "Lights", "Front Door", "Back door", "Garage", "Room 1", "Room 2", "Graph 1", "Graph 2"]
test = appuifw.popup_menu(L, u"Select and OK:"
if test == 0 :
    S = ["ON", "OFF"]
    sel = appuifw.popup_menu(S, u"Select and OK:"
if sel == 0 :
    appuifw.note(u"You have turned FANS ON", "info")
    bt_connect()
to_peer = sock.makefile("rw", 0)
while True:
    msg = ("FANON")
    if msg:
        if (i<3):
            command = send_command(msg)
            i=i+1
            if command:
                break
        e32.ao_sleep(2)
    else:
        appuifw.note(u"Unable to connect",
    sock.send("Unable to connect")
    break

to_peer.close()
sock.close()
list()

elif sel == 1:
    appuifw.note(u"You have turned FANS OFF", "info")
    bt_connect()
to_peer = sock.makefile("rw", 0)
while True:
    msg = ("FANOFF")
    if msg:
        if (i<3):
            command = send_command(msg)
            i=i+1
            if command:
                break
break
e32.ao_sleep(2)
else:
    appuifw.note(u"Unable to connect", "info")
    sock.send("Unable to connect")
    break
to_peer.close()
sock.close()
list()

elif test == 1:
    M = [u"ON", u"OFF"]
    sel = appuifw.popup_menu(M, u"Select and OK:")
    if sel == 0:
        appuifw.note(u"You have turned LIGHTS ON", "info")
        bt_connect()
        to_peer = sock.makefile("rw", 0)
        while True:
            msg = ("LightsON")
            if msg:
                if (i<3):
                    command =
send_command(msg)
                    i=i+1
                    if command:
                        break
e32.ao_sleep(2)
    else:
        appuifw.note(u"Unable to connect", "info")
        sock.send("Unable to connect")
        break
to_peer.close()
sock.close()
list()

elif sel == 1:
    appuifw.note(u"You have turned LIGHTS OFF", "info")
    bt_connect()
    to_peer = sock.makefile("rw", 0)
    while True:
msg = ("LightsOFF")
if msg:
    if (i<3):
        command =
        i=i+1
        if command:
            break
        e32.ao_sleep(2)
    else:
        appuifw.note(u"Unable to connect", "info")
        sock.send("Unable to connect")
        break

to_peer.close()
sock.close()
list()

e1f test == 2 :
    N = [u"ON", u"OFF"]
    sel = appuifw.popup_menu(N, u"Select and OK:")
    if sel == 0 :
        appuifw.note(u"You have turned Front DOOR LOCK ON", "info")
        bt_connect()
        to_peer = sock.makefile("rw", 0)
        while True:
            msg = ("FDON")
            if msg:
                if (i<3):
                    command =
                    i=i+1
                    if command:
                        break
                    e32.ao_sleep(2)
                else:
                    appuifw.note(u"Unable to connect", "info")
                    sock.send("Unable to connect")
                    break
        to_peer.close()
sock.close()
list()

elif sel == 1:
    appuifw.note(u"You have turned Front DOOR LOCK OFF",
"info")
    bt_connect()
    to_peer = sock.makefile("rw", 0)
    while True:
        msg = ("FDOFF")
        if msg:
            if (i<3):
                command =

send_command(msg)
        i=i+1
        if command:
            break
        e32.ao_sleep(2)
    else:
        appuifw.note(u"Unable to
connect", "info")
        to_peer.close()
        sock.send("Unable to connect")
        break

    to_peer.close()
    sock.close()
    list()

elif test == 3:
    N = [u"ON", u"OFF"]
    sel = appuifw.popup_menu(N, u"Select and OK;")
    if sel == 0:
        appuifw.note(u"You have turned Back DOOR LOCK ON",
"info")
        bt_connect()
        to_peer = sock.makefile("rw", 0)
        while True:
            msg = ("BDON")
            if msg:
                if (i<3):
                    command =

send_command(msg)
        i=i+1
        if command:
            break

    to_peer.close()
    sock.close()
    list()
break
e32.ao_sleep(2)

else:
    appuifw(note(u"Unable to
    sock.send("Unable to connect")
    break

to_peer.close()
sock.close()
list()

elif sel == 1:
    appuifw(note(u"You have turned Back DOOR LOCK OFF",
    "info")
    bt_connect()
to_peer = sock.makefile("rw", 0)
while True:
    msg = ("BDOFF")
    if msg:
        if (i<3):
            command =

    send_command(msg)
    i=i+1
    if command:
        break
    e32.ao_sleep(2)
else:
    appuifw(note(u"Unable to
    sock.send("Unable to connect")
    break

to_peer.close()
sock.close()
list()

elif test == 4:
    N = [u"ON", u"OFF"]
sel = appuifw.popup_menu(N, u"Select and OK:"
    if sel == 0:
        appuifw(note(u"You have turned Garage LOCK ON",
        "info")
        bt_connect()
to_peer = sock.makefile("rw", 0)
while True:
    msg = ("GARON")
    if msg:
        if (i<3):
            command =
send_command(msg)
        i=i+1
        if command:
            break
        e32.ao_sleep(2)
    else:
        appuifw.note(u"Unable to connect", "info")
        sock.send("Unable to connect")
        break

to_peer.close()
sock.close()
list()

elif sel == 1:
    appuifw.note(u"You have turned Garage LOCK OFF", "info")
    bt_connect()
    to_peer = sock.makefile("rw", 0)
    while True:
        msg = ("GAROFF")
        if msg:
            if (i<3):
                command =
            send_command(msg)
            i=i+1
            if command:
                break
            e32.ao_sleep(2)
        else:
            appuifw.note(u"Unable to connect", "info")
            sock.send("Unable to connect")
            break

to_peer.close()
sock.close()
list()

elif test == 5 :
    N = [u"ON", u"OFF"]
    sel = appuifw.popup_menu(N, u"Select and OK:")
    if sel == 0 :
        appuifw.note(u"You have turned Room 1 ON", "info")
        bt_connect()
        to_peer = sock.makefile("rw", 0)
        while True:
            msg = ("RM1ON")
            if msg:
                if (i<3):
                    command =
                    i=i+1
                    if command:
                        break
                    e32.ao_sleep(2)
                else:
                    appuifw.note(u"Unable to connect", "info")
                    sock.send("Unable to connect")
                    break
        to_peer.close()
        sock.close()
    list()

elif sel == 1:
    appuifw.note(u"You have turned Room 1 OFF", "info")
    bt_connect()
    to_peer = sock.makefile("rw", 0)
    while True:
        msg = ("RM1OFF")
        if msg:
            if (i<3):
                command = send_command(msg)
                i=i+1
                if command:
                    break
                e32.ao_sleep(2)
            else:
appuifw.note(u"Unable to connect")
sock.send("Unable to connect")
break
to_peer.close()
sock.close()
list()

elif test == 6:
    N = [u"ON", u"OFF"]
    sel = appuifw.popup_menu(N, u"Select and OK:")
    if sel == 0:
        appuifw.note(u"You have turned Room 2 ON", "info")
        bt_connect()
        to_peer = sock.makefile("rw", 0)
        while True:
            msg = ("RM2ON")
            if msg:
                if (i<3):
                    command = send_command(msg)
                    i=i+1
                    if command:
                        break
                    e32.ao_sleep(2)
                else:
                    appuifw.note(u"Unable to connect")
                    sock.send("Unable to connect")
                    break
        to_peer.close()
        sock.close()
        list()

    elif sel == 1:
        appuifw.note(u"You have turned Room 2 OFF", "info")
        bt_connect()
        to_peer = sock.makefile("rw", 0)
        while True:
            msg = ("RM2OFF")
            if msg:
                if (i<3):
                    send_command(msg)
                    i=i+1
                    if command:
                        break
                    e32.ao_sleep(2)
                else:
                    appuifw.note(u"Unable to connect")
                    sock.send("Unable to connect")
                    break
        to_peer.close()
        sock.close()
        list()
command = send_command(msg)
i = i + 1
if command:
    break
e32.ao_sleep(2)
else:
    appuifw.note(u'Unable to connect', 'info')
    sock.send('Unable to connect')
    break
to_peer.close()
sock.close()
list()

elif test == 7:
    graph()
    app_lock = e32.Ao_lock()
    app_lock.wait()

elif test == 8:
    draw()
    app_lock = e32.Ao_lock()
    app_lock.wait()

list()
#appuifw.app.exit_key_handler = quit
app_lock = e32.Ao_lock()
app_lock.wait()
REFERENCES


http://pys60.garage.maemo.org/doc/tut/tut.html

http://pys60.garage.maemo.org/doc/s60/s60.html


