Device Control in an Ad-hoc Network Environment by using MoSync for Multiple Platform Mobile Application Development

Sharon Panth
Assistant Professor: Computer Science Department
Shri M. & N. Virani Science College
Rajkot, Gujarat (India)
sharon.panth20@gmail.com

Mahesh Jivani
Associate Professor: Department of Electronics
Saurashtra University
Rajkot, Gujarat (India)
mnjivani@gmail.com

Abstract— These days mobile application development involves multiple OS platforms like Android, iOS, Windows, Symbian, Java Mobile and the Moblin platform and a wide range of device technologies. Developed mobile applications should be delivered a consistently meaningful user experience across different devices and work successfully on various OS platforms. This may look easy on paper, but it is in fact greatly multifaceted, while putting into practice, because of a variety of aspects like highly fragmented mobile technology scenario, quickly evolving standards, constraints imposed by the mobile device itself (screen size, input methods, display capabilities, memory etc.). MoSync is a free and an open source Software Development Kit (SDK) for mobile applications. MoSync can build application packages for hundreds of different mobile devices on a wide range of mobile operating systems. This paper presents the design and implementations on not only economical but flexible and secure mobile phone based device control system using MoSync framework and eight bit microcontroller in an ad-hoc network environment. The devices are connected to the on/off relay via microcontroller ports and controlled through mobile phone. The communication between the mobile phone and the device control system is through Bluetooth wireless technology. This system is designed to be economical and modifying eight appliances to be controlled with minimum efforts.

Keywords- MoSync, Multiple OS Platform, Microcontroller, Device Control, Bluetooth, Mobile Phone

I. INTRODUCTION

We are experiencing a remarkable advancement of the mobile technology with the appearance of an innovative devices gradually more powerful and operating systems with enhanced usage of the function [1]. Mobile applications are increasingly in use and the technology progresses into mainstream, the requirement for prolong performance and instruments for building and to maintain current advancement becomes evident [2]. The availability of various mobile operating systems with diverse programming languages and development tools can be a dilemma when someone wants to release an application in all available platforms. The straightforward solution is to code the application for each one of these platforms which generally not practical also in terms of funding or time of development, involving a greater endeavour to be made. As a result, a solution is to create an application for a number of platforms (multi or cross-platform) without loosing the overall quality of the application will reduce the time to code and increase the large number of potential users. Providentially, during the last years few efforts have been conducted to tackle this problem, especially with the emergence of tools and frameworks that facilitate the development of cross/multi platform mobile applications [3].

Basically, there are two types of applications that can be developed for mobile phones: one is Native application and the second is Cross platform application. The Native application is specially designed to run on the mobile or the operating platform supporting it. For example, the mobile application is created using Objective C it will run on iOS platform enabled smart phones and tablets, same as for other proprietary OSs like Symbian or Nokia. Cross platform mobile applications are to build the mobile application once and to deploy it on a number of mobile platforms. For example, if a mobile application is created on Android, it can also work for iOS, Windows Phone or other platforms depending upon the cross-platform development tools (CPTs) used for the development of the application [4].

There are various factors which can help to decide whether the development of the mobile application should be done as native application or mobile application:

- Platform Support: The application must support the number of platforms and the recommended ones.
- Technical Support: The mobile required features which can be implemented in the application.
- Performance Support: Evaluating the graphical or computational performance of the real applications.
- Design constrain: Different devices are having different design constraints and how to resolve them.

These parameters are enough to measure the capabilities of a particular Multi/Cross Platform Tool. Nowadays many cross platform framework are freely available or payable in the market, like: Rhomobile, appMobi, Appcelerator, DragonRAD, Sencha, AppLoader, PhoneGap, MoSync, Qt and TotalCross [3].

This paper focuses on MoSync [5] Framework and attempts to provide a view of the actual state of this area by creating an ad-hoc network application based on Bluetooth wireless Technology.

II. System Architecture

Device Control using MoSync (a free and an open source framework for Multiple Platform Mobile Application Development) developed as a part of this research comprises of the User Interfaced (UI) MoSync Application program implemented on Bluetooth enabled mobile phones, and an 8 bit microcontroller based household appliances control circuit with Serial Bluetooth Module, which communicates to mobile phones through the Bluetooth. The system is based on serial data transmission using Bluetooth wireless communication in order to facilitate the domestic device control. This system ensures a secured exchange of data on wireless communication. A user interface (UI) on the multi/cross platform enabled mobile phone offers system connection and control utilities. ULN 2803 relay driver [6] and Serial Bluetooth Module from TINY OS [7] as well as Keil μvision IDE [8] for compiling C Language Code and μCflash + programmer [9] for burning HEX file into microcontroller are used for the development. An ATMEL AT89C51, 8 bit microcontroller [10] is used as an embedded relay controller.

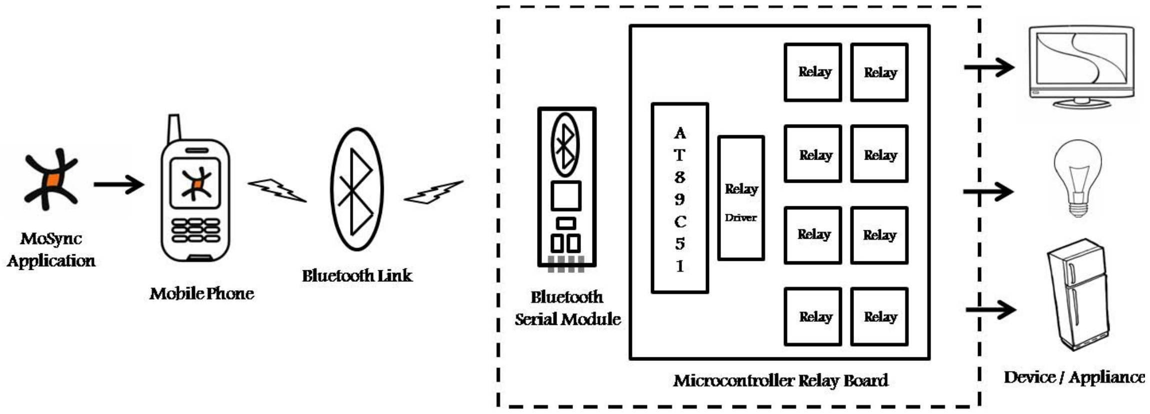


Figure 1. System Architecture

III. Development Platform

This section describes the technologies used for developing the mobile phone application of the multi/cross-platform device control system. The major technologies used for the development of the mobile phone application are Mosync and Bluetooth.

MoSync [5] is a platform to write and to run multi/cross-platform mobile applications on mobile devices. Bluetooth is a standard for wireless technology for providing connectivity for fixed and mobile devices. Together, both the technologies combine to create device control multi/cross platform mobile application for an ad-hoc network environment.

A. *MoSync*

MoSync AB, a Swedish software development company previously known as Mobile Sorcery AB developed MoSync. It is a free and an open source mobile software development kit (SDK), integrated with the Eclipse IDE. It is available in dually license, a commercial license and an open source GPL version 2 licenses. The MoSync framework produces Native mobile applications using C/C++, HTML5 scripting and any language. MoSync can build packages for many mobile devices running over different operating systems. Initially MoSync supports the Java ME platform but now it has covered almost all the popular platforms like Android, Windows Phone, Windows Mobile, Symbian, iOS, Moblin, etc. The MoSync SDK has the option for creating PC/Mac applications using C/C++ or using HTML5/JavaScript/CSS for web application, or a combination of both to create hybrid applications. Hence, MoSync stands out on two things: single code base and C/C++. Basically, C/C++ is a prominent factor to build truly cross platform hybrid mobile applications. There are various reasons to explain the choice of MoSync over other framework for the development of multi/cross platform application. Here are some highlights explaining the selection:

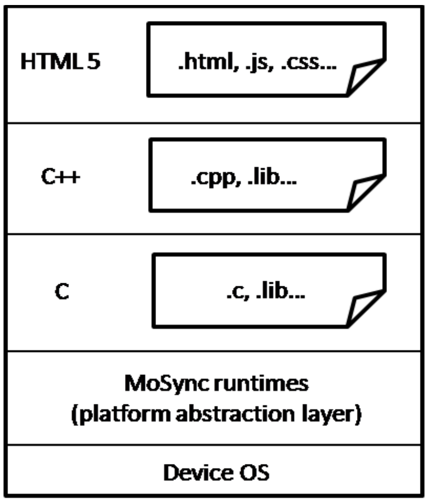


Figure 2. MoSync mobile App layers

- MoSync creates only one project to maintain for all the platforms.
- The entire process for JavaScript is placed on the same file for all of the operating system.
- For C++, MoSync provides a lot of features if they aren't accessible from JavaScript.
- Native UI support is not only restricted to JavaScript frameworks but also created native UI elements which are more responsive using only JavaScript.
- The power of C++ can be used where it lacks in the performance of JavaScript to support the code written with it to work on every platform with single code base.
- Built-in support for push notification with a unified interface on every supported platform is available.
- Support for Native UI and Wormhole technology.

B. Bluetooth

Bluetooth is a globally unlicensed wireless technology using short-wavelength radio transmissions in the ISM band from 2.4 GHz for secured data transmissions for fixed and mobile devices creating personal area networks by Telecom vendor Ericsson in 1994 [11]. This technology is an alternative to the RS-232 data cables. The Bluetooth Special Interest Group, since 1998 manages this standard. A Bluetooth device needs to be qualified according to the standards of the SIG. Today most of the mobile devices are integrated with small, low-power and cheap chips working as short-range radio transceivers for Bluetooth communications.

Being the cheapest technology for short range transmissions, Bluetooth also depends upon its security mechanisms provided using device pairing, authentication, encryption and authorization techniques. MoSync supports the Bluetooth technology for developing the applications in the mobile devices that provides a completely different and low or no cost alternative to create and to deploy the mobile applications over multiple platforms. Bluetooth works by implementing a number of protocols according to its layered architecture. The Bluetooth protocol stack or the architecture defines the following protocols which can be used in the application.

- LMP: The Link Management Protocol (LMP) connects and controls the links between the devices on the controller.
- L2CAP: The Logical Link Control and Adaptation Protocol (L2CAP) both sustain multiple logical connections among devices and allow assembling and disassembling of packets in the course of communication.
- SDP: The Service Discovery Protocol that provides service discovery for the device.
- RFCOMM: RFCOMM (Radio Frequency Communication) protocol is RS-232 cable serial emulation.
- OBEX: OBEX (OBject EXchange) is an adopted session layer protocol for object exchange between the devices.

The Bluetooth profile used in device control mobile phone application is the Bluetooth Serial Port Profile (btspp) [12]. RFCOMM (Radio Frequency Communication) is a streaming communication and connection-oriented protocol. Considering profile and protocol, btspp and RFCOMM are used respectively in the Mosync application for the serial port communication.

C. Keil μVision IDE

Keil development tools for the 8051 Microcontroller Architecture support every level of embedded software development. The industry-standard Keil C Compilers, Macro Assemblers, Debuggers, Real-time Kernels, Single-board Computers, and Emulators support all 8051 derivatives. 'C Language Program code' for AT89c51microcontroller is developed, compiled and debugged using Keil μVision IDE [8].

D. μcFlash+ Programmer

The μcFlash+ Programmer [9] is an affordable, reliable, and fast programmer for MCS51/AVR Microcontrollers and 24Cxx I2C EEPROMs. The programmer is designed to operate with the Intel Pentium-based IBM-compatible desktop computers and notebook computers. No interface card is necessary to plug the module into a PC (this feature is especially handy for notebook computer users). The menu-driven software interface makes it easy to operate. μcFlash+ Programmer is used here for programming AT89C51 microcontroller.

IV. SYSTEM HARDWARE

Device control hardware is work as client part and it is known as circuit for device control which is shown in Figure 4. Device control Circuit comprises microcontroller AT89C51, Serial Bluetooth Module, octal peripheral driver array ULN2803 and a few discrete components. Here in this circuit, microcontroller AT89C51 is worked as main programmable switching unit which receives data from Bluetooth serial module and transferred appropriate program data to ULN2803 for operating relay ON and OFF. The AT89C51 is a low-power, high-performance CMOS 8-bit microcomputer with 4K bytes of Flash programmable and erasable read only memory (PEROM). The Atmel AT89C51 is a powerful microcomputer which provides a highly-flexible and cost-effective solution to many embedded control applications. The AT89C51 provides the following standard features: 4K bytes of Flash, 128 bytes of RAM, 32 I/O lines, two 16-bit timer/counters, five vector two-level interrupt architecture, a full duplex serial port, and on-chip oscillator and clock circuitry. [10]

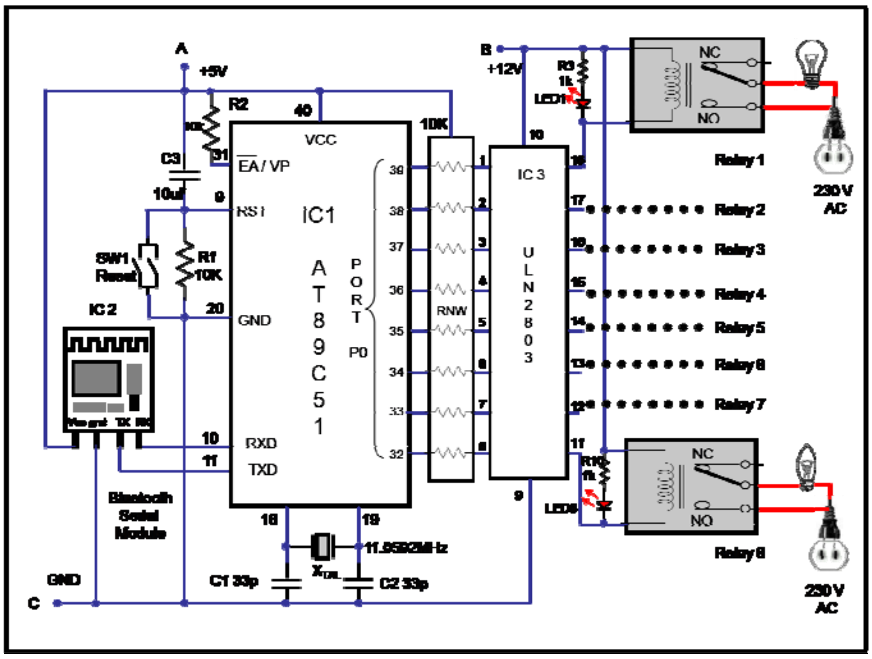


Figure 3. Circuit Diagram

V. DEVICE CONTROL MOSYNC APPLICATION

A mobile phone with connectivity options of Bluetooth has the capability of connectivity to any other device in its proximity. The communication channel is established over the Bluetooth enabled devices for information exchange. The RFCOMM protocol is a partial implementation of the ETSI TS 07.10 standard [12], placed on top of the L2CAP protocol. The device manufacturers can have the flexibility to easily extend the serial port functionality of their Bluetooth devices. MoSync combines the Bluetooth serial port profile (btspp) and the

RFCOMM protocol. A method which explains to establish the direct connection, between remote devices that are already known. The advantage of using this method is to establish connection directly by specifying known address of the device (here Bluetooth Serial Adapter) within very short time. For the communication between the mobile phone and the embedded system the connection string must contain the 'btspp protocol' in the URL.

For this implementation, a Mosync C/C++ application has been created. In the Classic Development model the element of the application can be controlled. This model is beneficial for porting existing C applications or to have control over flow and events of the program.

- The MoSync event loop can be implemented using the following steps:
- Checking for events should be done often.
- The MoSync event queue is not infinite, so checking should be done only for events.
- To conserve CPU usage and battery power, **maWait()** can be used rather than busy-waiting.
- The way to respond and handle the close event.
- After a 'close event' is posted, application will be forcibly terminated within some time. The application should get exit on its own as soon as possible, after saved any important data.
- After the close event has been posted, no events will be posted and will have no effect of most syscalls.

Following this model, the device control application has been created.

Since MoSync doesn't support user-created threads but provides asynchronous interfaces to time-consuming Native functionality such as network communication. Syscalls can't accept function pointers the central MoSync event system provides progress notification. EVENT struct provides a few built-in event types and custom events using void*. In classic applications working with connections will involve responding to events whose event type is *EVENT_TYPE_CONN*. Connection operations are executed asynchronously. Hence, MoSync socket/Bluetooth/HTTP connections are asynchronous and through these events the application is notified of their progress, results and termination.

The parameter string that describes the target should get conform to the URL format as described in RFC 2396 [12]. This takes the general form:

{scheme} :[{ target}][{params}]

The {scheme} is the name of a protocol such as http}.

The {target} is normally some kind of network address.

{params} are formed as a series of equates of the form ";x=y".

The Connector class open method has a parameter of connection string and this is casted as per the requirement of the connection in the application. The connection string used in the mobile phone application is in the format of the following:

btspp://address : port

Where, btspp:// is the Bluetooth Serial Port Profile,

address is the Bluetooth known remote device address of 12 digit hexadecimal format,

port is the port number or the communication channel on which the remote device receives data.

For example, btspp:// 001207121217:1

VI. WORKING OF MOSYNC DEVICE CONTROL APPLICATION

The user must have a mobile phone with Bluetooth facility on most of the platform except iOS. The application must be transferred to the phone and installed on it. When user launches the application it displays the Initial screen as shown in figure 5. This screen has an option to exit the application or to connect to the embedded system which has been started previously. The user gets console messages for the entire application activities. When the application is launched it will automatically connect to the embedded system using the static URL provided in the application itself.

If the connection is successful then the client application gives an appropriate message on the console and instructions to operate the system using the mobile device. Depending upon the type of the available input with the user's mobile device user can control the devices. If the mobile device is Touch screen then tapping on the device enables the virtual keyboard otherwise the user can use the available device keyboard. Once the serial port is opened for communication, the user can select the options as following:

- Select numbers [1-8] for toggling single device ON/OFF
- Select number 9 for toggling all devices ON/OFF
- Select BACK/CLOSE for exit

This process goes on until user desires to exit the application. Any appliances inside areas like house, shops, etc. can be controlled using device control system where the embedded systems are enabled and ready to connect and communicate with the mobile phone. Now the user can select any electrical Home Appliance to make it ON or OFF the device simultaneously. The user also gets the message on the console of the application on the mobile phone regarding which device is ON/OFF as shown in figure 5(d). Finally, the user can exit the application and can refrain from continuing further.

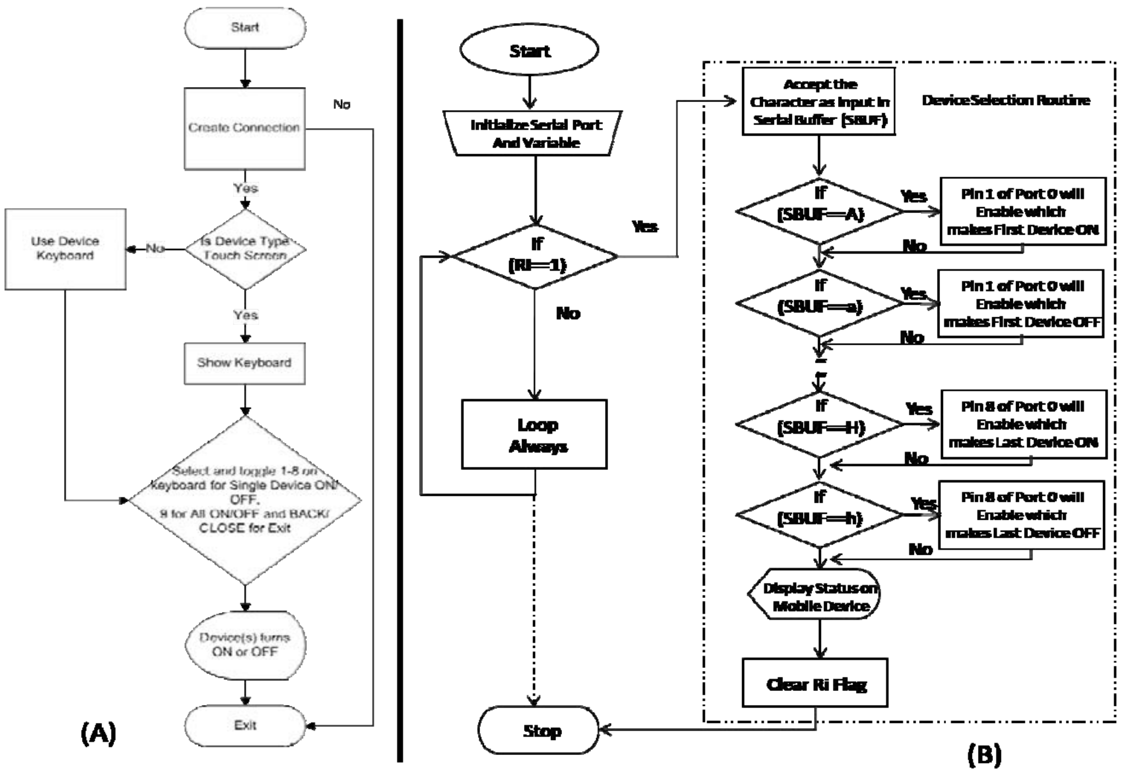

Start
Create Connection
No
Yes
Use Device Keyboard
No
Is Device Type Touch Screen
Yes
Show Keyboard
Select and toggle 1-8 on Keyboard for Single Device ON/ OFF. 9 for All ON/OFF and BACK/ CLOSE for Exit
Device(s) turns ON or OFF
Exit
(A)
Start
Initialize Serial Port And Variable
If (RI==1)
Yes
No
Loop Always
Stop
Accept the Character as Input in Serial Buffer (SBUF)
Device Selection Routine
If (SBUF==A)
Yes
Pin 1 of Port 0 will Enable which makes First Device ON
No
If (SBUF==a)
Yes
Pin 1 of Port 0 will Enable which makes First Device OFF
No
If (SBUF==H)
Yes
Pin 8 of Port 0 will Enable which makes Last Device ON
No
If (SBUF==h)
Yes
Pin 8 of Port 0 will Enable which makes Last Device OFF
No
Display Status on Mobile Device
Clear Ri Flag
(B)

Figure 4. Flow Chart: (A) Mobile Application Program, (B) Microcontroller Program

VII. MICROCONTROLLER AT89C51 PROGRAM

The program (HAC.c) for the microcontroller is written in C and compiled using Keil Vision IDE to generate hex code. The generated hex code is burnt into the microcontroller using an C Flash+ programmer. The program coding starts with '#include <reg51.h>' header files. The microcontroller port pins are defined using 'sbit' function for interfacing with the surrounding peripherals.

By selecting '1 to 9' in Keypad of mobile in Device Control Application program which is developed using MoSync platform and installed on mobile device, will transmit ASCII characters serially through Bluetooth and on the receiving side microcontroller receives data through serial Bluetooth module on serial port of microcontroller AT89C51. SBUF register will accept this ASCII characters as equivalent HEX value which will be ported to Port 0's pins of AT89C51.

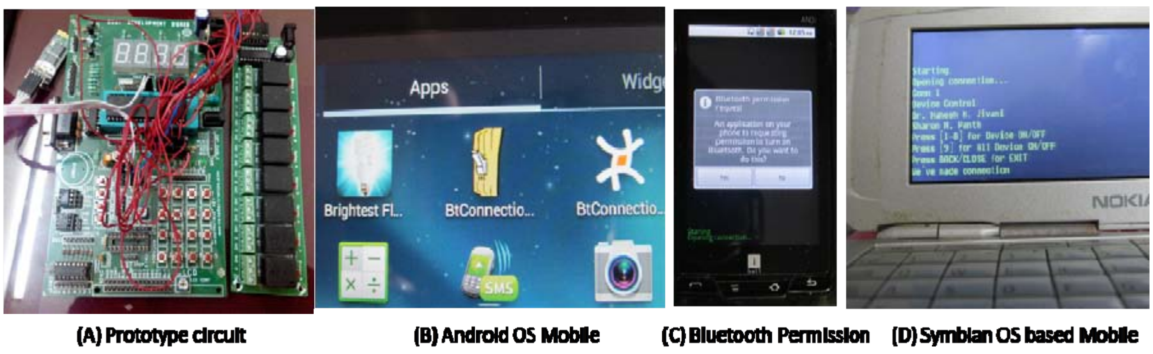
For operating Relay1 in ON condition, one has to press '1' key of 'mobile keypad' means ASCII character "A" will be transmitted and microcontroller will receive equivalent value in HEX through SBUF register which is "0x41" through Bluetooth serial module which makes 'pin 1 of Port 0' high and for making Relay1 in OFF condition simply again press key '1' of 'mobile keypad' means ASCII character 'a' will be transmit, microcontroller will receive equivalent value in HEX through SBUF register which is "0x61" which makes 'pin 1 of Port 0' low. Same way you can ON and OFF relay connected to Port 0. MoSync Application program on mobile phone transmits following combination given in Table 1.

TABLE I. MoSync Application Program Transmission Combination

N0.	**Mobile Keypad**	**Microcontroller AT89C51 Port 0**		
	Key (toggling)	***Character***	***Relay Number***	***Status***
1	Key 1 to Key 8	Capital Letter ‘A’ to ‘H’	1 to 8 (individual)	ON
2	Again Key 1 to Key 8	Small Letter ‘a’ to ‘h’	1 to 8 (individual)	OFF
3	Key 9	Special Character ‘(‘	All	ON
4	Again Key 9	Special Character ‘)’	All	OFF

VIII. IMPLEMENTATION

Power up the Circuit and scan the Bluetooth devices on your Mobile Device. If everything is done correctly you will be able to find a Bluetooth device named ‘Tiny OS’ or name of Bluetooth Module. You will be asked for a pairing code in case of the above model (Tiny OS) it is 1234 but it might be different if you are using a Bluetooth module from another vendor. Now simply press key 1 to 9 for controlling devices from mobile phone.



```
Starting
Opening connection...
Conn 1
Device Control
Dr. Mahesh H. Jivani
Sharon N. Panth
Press [1-8] for Device ON/OFF
Press [9] for All Device ON/OFF
Press BACK/CLOSE for EXIT
We've made connection
```

Figure 5. Real Implementation

Figure 5 shows the real implementation of Device Control on Android and Symbian OS based Mobile Phone. As shown in figure 5, follow the steps for operating devices listed in program.

- Start Bluetooth connection of your mobile phone.
- Open the ‘Device Control’ application on your mobile phone.
- ‘Select connect device’ from ‘option menu’ using ‘scan’ for new devices.
- ‘Bluetooth Serial Module will be available under ‘select a device to connect’ list.
- Pair ‘Bluetooth Serial Module’ by providing pair code ‘1234’.
- Start selecting particular device for making ‘ON/OFF’ from the keyboard as shown in figure 5, you will see relay ON and OFF according to combination shown in Table 1.

IX. CONCLUSION

Design and implementation of a Device Control System for Multi/Cross Platform using MoSync on mobile phone has been discussed. The purpose of the system is to use mobile phone’s inbuilt Bluetooth facility for automation without using Air Time. Different hardware and software unit of the system are described. The complete application software has been designed by using MoSync, Bluetooth API and C Language. The Device Control application program is tested on various multi/cross-platform mobile phones and the results are presented in table 2, which are quite satisfactory and response received from the community in general is encouraging. The Device Control System using MoSync furnishes a good paradigm for any Automation System based on Multi/Cross-Platform Mobile Phone and Bluetooth.

TABLE II. LIST OF MOBILES ON WHICH MOSYNC DEVICE CONTROL APPLICATION TESTED

N0.	Mobile Phone	Platform
	Company and Make	***Mobile OS***
1	Nokia 9300i	Symbian OS v7.0s, Series 80 v2.0 UI
2	Nokia 7230	Symbian OS Series 40 6th Edition
3	Nokia 5130 c-2 Xpress Music	Symbian OS Series 40 5th Edition
4	Nokia N73	Symbian 9.1, S60 3rd Edition
5	Nokia c2	Symbian OS Series 40 6th Edition
6	Samsung Galaxy S2	Android OS Version 2.3 "Gingerbread”
7	Micromax Bolt A35	Android OS Version 2.3 "Gingerbread”
8	Tablet Samsung GTP3100	Android OS 4.1 "Ice-Cream Sandwitch”
9	iBall Andi	Android OS Version 2.2 "Froyo”
10	LAVA iris 504q	Android OS Version 4.2 "Jelly Bean”

X. FUTURE WORK

Currently, Device control application program is supporting all types of Android OS mobile phone (including touch screen) and Symbian and JAVA based mobile phones which have keypad as shown in Table 2. Still, touch screen of Symbian, JAVA and Windows based mobile is not explored so, using EDIT BOX facility of MoSync, new version of Device control application will be developed, which will have support of all types of mobile phone including keypad as well as touch screen.

REFERENCES

[1] K. Rummler, J. Seipold, E. Lubcke, N. Pachler and G. Attwell, “Mobile Learning: Crossing boundries in convergent environment,” Book of abstracts of the conference mobile learning:crossing boundries in convergent, March 21-22, 2011. ISSN 1753-3385.

[2] G. Hartmann, G. Stead, A. DeGani, “Cross-platform mobivle development” TRIBAL, http://www.mole-project.net, March 2011.

[3] A. Ribeiro and A. R. daSilva, “Survey on Cross-Platforms and Languages for Mobile Apps,” Eighth International Conference on the Quality of Information and Communications Technology (QUATIC), IEEE Xplore, ISBN 978-1-4673-2345-1, pp. 255-260, 2012.

[4] A. Charland and B. LeRoux, “Mobile Application Development: Web vs. Native”, Commun. ACM, vol. 54, no. 5, pp. 49-53, May 2011.

[5] MoSync, “http://www.mosync.com/the_company,” last visit August 2013.

[6] Darlington Transistor Array, Texas Instruments, http://www.ti.com/lit/ds/symlink/uln2803a.pdf, last seen on August, 2013.

[7] Serial Bluetooth Module, Tiny OS Electronics, http://www.tinyosshop.com/index.php?route=product/product&product_id=330, 2013.

[8] Keil µvision IDE, http://www.keil.com/uvision/, last seen August 2013.

[9] µCFlash+ Programmer, UC Micro Systems, http://www.ucmicrosys.com/products/ic-programmers/µcflash.html, last seen on August 2013.

[10] AT89c51 8 bit Microcontroller, ATMEL Corporations, http://www.atmel.com/images/doc0265.pdf, last seen on August 2013

[11] J. Haartsen, "BLUETOOTH—The universal radio interface for ad hoc, wireless connectivity", Ericsson Review No. 3, pp. 110-117, 1998.

[12] P. Singh P, S. Sharma, S. Agrawal, “Study of Bluetooth wireless technology using java”, Indian Journal of computer Science and Engineering (IJCSE), Vol. 2, No. 3, pp. 295-307, 2011.