IMPLEMENTATON OF MOBILE INFORMATION DEVICE PROFILE ON VIRTUAL LAB

Aravind Kumar Alagia Nambi
Department of Electrical and Electronics Engineering, Anna University, Chennai-600025, India
Email: a_aravind_k@yahoo.co.in

Abstract: The rate at which information is produced in today's world is mind-boggling. The information is changing by every minute and today's corporate mantra is not "knowledge is power" but "Timely knowledge is power". Millions of Dollars are won or lost due to information or lack of it. Business executives and corporate managers push their technology managers to provide information at the right time in the right form. They want information on the go and want to be connected all the time to the Internet or their corporate network. The rapid advancement of Technology in the field of miniaturization and that of communications has introduced a lot of roaming devices for people to connect through to the network like laptop, PDA, mobile phones and many embedded devices. Programming for these devices were cumbersome and limited since each device supported their own standard I/O ports, screen resolution and had specific configurations. The introduction of Java 2 Micro Edition (J2ME) has solved this problem to some extent. J2ME is divided into configuration and profiles, which provide specific information to a group of related devices. Mobile phones can be programmed using J2ME. If the mobility offered by the cellular phones combined with Electrical Engineering many new uses can be found out for existing electrical machines. It will also enable remote monitoring of electrical machines and the various parameters involved in Electrical Engineering.

Keywords: Mobile Internet service, Mobile computing, MIDLET, Virtual lab

1 INTRODUCTION

The rate at which information is produced in today's world is mind-boggling. The information is changing by every minute and today's corporate mantra is not "knowledge is power" but "Timely knowledge is power". Millions of Dollars are won or lost due to information or lack of it. Business executives and corporate managers push their technology managers to provide information at the right time in the right form. They want information on the go and want to be connected all the time to the Internet or their corporate network. The rapid advancement of Technology in the field of miniaturization and that of communications has introduced a lot of roaming devices for people to connect through to the network like laptop, PDA, mobile phones and many embedded devices. Programming for these devices were cumbersome and limited since each device supported their own standard I/O ports, screen resolution and had specific configurations. The introduction of Java 2 Micro Edition (J2ME) has solved this problem to some extent. J2ME is divided into configuration and profiles, which provide specific information to a group of related devices. Mobile phones can be programmed using J2ME. If the mobility offered by the cellular phones combined with Electrical Engineering many new uses can be found out for existing electrical machines. It will also enable remote monitoring of electrical machines and the various parameters involved in Electrical Engineering.

1.2 DESIGN CONSIDERATIONS FOR SMALL COMMUNICATION DEVICES

Developing applications for small devices requires one to keep certain strategies in mind during the design phase.

- Keep it simple
- Smaller is better:
- Minimize run-time memory use
- Let the server do most of the work

2 J2ME – An Overview

J2ME uses configurations and profiles to customize the Java Runtime Environment (JRE). As a complete JRE, J2ME is comprised of a configuration, which determines the JVM used, and a profile, which defines the application by adding domain-specific classes. The configuration defines the basic run-time environment as a set of core classes and a specific JVM that run on specific types of devices. Two Configurations have been defined for J2ME:

- Connected Device Configuration (CDC) is used with the C virtual machine (CVM) and is used for 32-bit architectures requiring more than 2 MB of memory. An example of such a device is a Net TV box.
- Connected Limited Device Configuration (CLDC) implementation (static size of the virtual machine + libraries) should fit in less than 128 kilobytes. CLDC is used with K Virtual Machine (KVM). The CLDC Specification assumes that applications can be run in as little as 32 kilobytes of Java heap space. (Example: Cell phones, PDAs, etc)

2.1 PROFILES OVERVIEW

The profile defines the type of devices supported by the application. Specifically, it adds domain-specific classes to the J2ME configuration to define certain uses for devices. Profiles are built on top of configurations. Two profiles have been defined for J2ME and are built on CLDC: KJava and Mobile Information Device Profile (MIDP). These profiles are geared toward smaller devices.

Target devices for J2ME applications developed using CLDC generally have the following characteristics:

- 160 to 512 kilobytes of total memory available for the Java platform
- Limited power, often battery powered
- Network connectivity, often with a wireless, inconsistent connection and with limited bandwidth
- User interfaces with varying degrees of sophistication; sometimes with no interface at all

Some devices supported by CLDC include wireless phones, pagers, mainstream personal digital assistants (PDAs), and small retail payment terminals.

2.2 KVM TECHNOLOGY

The KVM technology is a compact, portable Java virtual machine specifically designed from the ground up for small, resource-constrained devices. The high-level design goal for the KVM technology was to create the smallest possible "complete" Java virtual machine that would maintain all the central aspects of the Java programming language, but would run in a resource constrained device with only a few hundred kilobytes total memory budget. More specifically, the KVM technology was designed to be:

- Small, with a static memory footprint of the virtual machine core in the range of 40 kilobytes to 80 kilobytes (depending on compilation options and the target platform,)
- Clean, well-commented, and highly portable,
- Modular and customizable,
- As "complete" and "fast" as possible without sacrificing the other design goals.

The "K" in KVM stands for "kilo." It was so named because its memory budget is measured in kilobytes (whereas desktop systems are measured in megabytes). KVM is suitable for 16/32-bit RISC/CISC microprocessors with a total memory budget of no more than a few hundred kilobytes (potentially less than 128 kilobytes). This typically applies to digital cellular phones, pagers, personal organizers, and small retail payment terminals. The minimum total memory budget required by a KVM implementation is about 128 kB, including the virtual machine, the minimum Java class libraries specified by the configuration, and some heap space for running Java applications. A more typical implementation requires a total memory budget of 256 kB, of which half is used as heap space for applications, 40 to 80 kB is needed for the virtual machine itself, and the rest is reserved for configuration and profile class libraries. The ratio between volatile memory (e.g., DRAM) and non-volatile memory (e.g., ROM or Flash) in the total memory budget varies considerably depending on the implementation, the device, the configuration, and the profile. A simple KVM implementation without system class prelinking support needs more volatile memory than a KVM implementation with system classes (or even applications) preloaded into the device.

The actual role of a KVM technology in target devices can vary significantly. In some implementations, the KVM technology is used on top of an existing native software stack to give the Java content on the device. In other implementations, the KVM technology is used at a lower level to also implement the lower-level system software and applications of the device in the Java programming language. Several alternative usage models are possible. At the present time, the KVM and CLDC technologies are closely related. CLDC technology runs only on top of KVM technology, and CLDC technology is the only configuration supported by KVM technology. However, over time it is expected that CLDC technology will run on other J2ME virtual machine implementations and that the KVM technology may perhaps support other configurations as they are defined.

2.3 MIDlets:

J2ME architecture defines Mobile Information Device (MID) Profile, which is actually a set of Java APIs. Together with the CLDC, it provides a complete J2ME application runtime environment exclusively targeted at mobile devices like cell phones. MIDP takes care of user interface, networking, persistent storage, and basic graphics. The primary objective is to keep the application implementation size minimal, so that it can perfectly fit in a small memory footprint and can run on low-end microprocessors with restricted heap sizes and negligible garbage creation.

MIDlet is an application written for Connected Limited devices like cell phones and pagers which can work on top of J2ME architecture. Devices that support MID profile have Application Management Software (AMS) which takes care of installing, supervising, and removing MIDlets in an interactive way. The AMS also takes care of the MIDlet Life Cycle taking it through various phases. In the first phase AMS gets MIDlet from a source and loads it in the device memory. In the next phase AMS examines the MIDlet and checks whether the MIDlet confirms to the device security policy and check for possible violations. In the launching phase the MIDlet gets loaded into the KVM and becomes up and active. AMS has the additional responsibility of maintaining the version numbers of all MIDlets for up gradation.

3 OVERVIEW

'Virtual Lab' developed in the work demonstrates that electrical machines can be controlled using a cell phone. It consists of a stepper motor and a DC servomotor, both of which can be controlled through the cell phone. The angle of rotation and the direction of rotation can be controlled. As far as servomotor is concerned, the ON/OFF status can be checked, the current speed (in RPM) can be checked and mechanical time lag is measured from the cell phone.

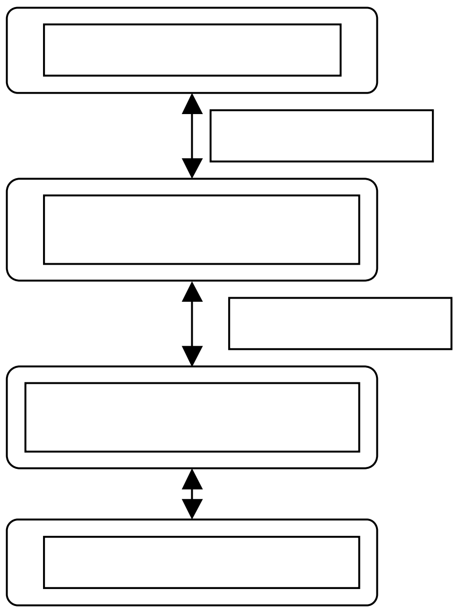


Figure 1

The primary objective was to make the connection between the cell phone and computer to which the motor is connected. A MIDlet that establishes a HTTP connection between the cell phone and the main server was written. The main server then makes a socket connection to the intended computer. The application specific server program, which runs in the destination computer, will be made to listen in a specific port to which the main server makes the socket connection and it opens a data stream for input and output data flow. The application specific server program then calls the native code through Java Native Interface (JNI). The native code accesses the port and does the required computation and then returns the results to the main server, which in turn routes the data to the MIDlet running in the cell phone for display. This control flow is shown in the *figure 1.*

3.1 STEPPER MOTOR CONTROL

Four poles DC stepper motor is used here. The user inputs through the cell phone are the degrees to rotate and the direction of rotation. The number of steps required for the motor to rotate through the given degree was calculated and the appropriate signals were sent through the printer port. The signal from the parallel port is not strong enough to run the motor. To amplify the signal the ULN2003 IC is used in the circuit. The circuit details are shown in the *figure 2*.

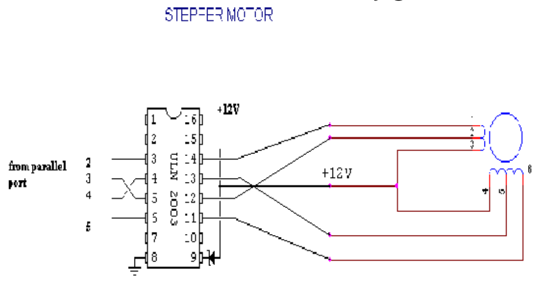
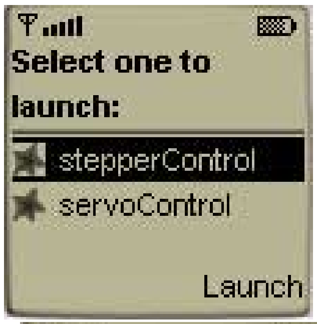


Figure 2

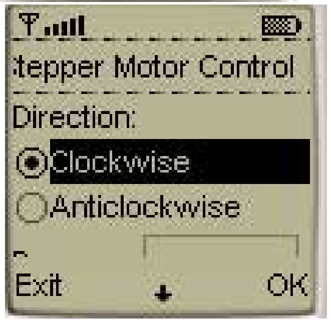
The screens below (*figure 3*) are the actual output as seen on the cell phone while controlling the stepper motor


Select one to launch:

stepperControl

servoControl

Launch

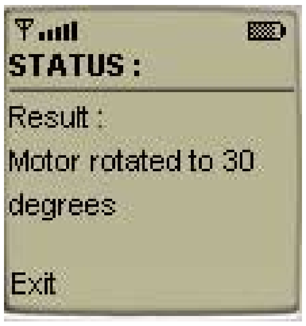

tepper Motor Control

Direction:

Clockwise

Anticlockwise

Exit OK


STATUS :

Result :

Motor rotated to 30 degrees

Exit

Figure 3

3.2 DC SERVOMOTOR CONTROL

12V DC Servomotor is used here. The contact relay is used to switch the servomotor on or off. The rating of the relay is 6V, 5A, 100 Ohm. The relay is controlled by the signals from the parallel port. This is achieved through a power transistor, SL100, which is CE configured. The transistor is biased by the signal from the parallel port given to the base of the transistor and the relay operates. The circuit details are shown in the *figure 4*.

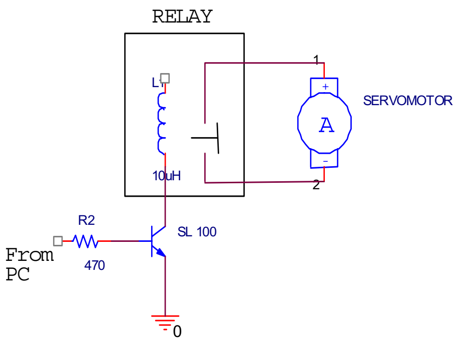


Figure 4

A circular disk with a small slit is mounted along with the servomotor. When the slit comes alongside the optical sensor the sensor senses the light and gives an output of 5V. This biases the transistor. Otherwise the transistor is not biased and no output is sensed. Every time the output becomes 5V the program increments a counter and the speed of the motor is thus measured over a certain length of time. This measurement is relayed back to the cell phone. The circuit details are shown in the *figure 5*.

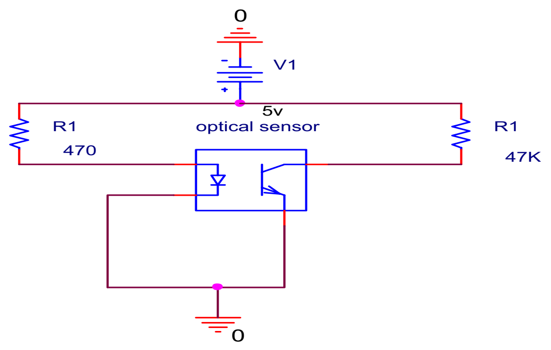
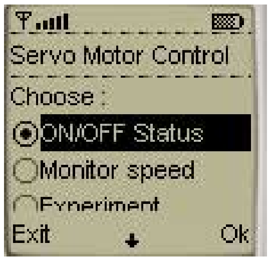
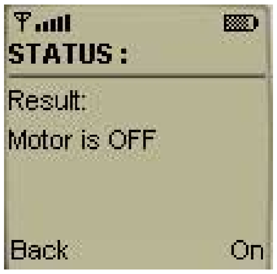


Figure 5

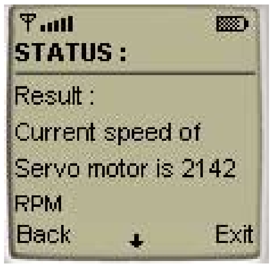
Thus the ON/OFF status, current speed and the mechanical time lag of the DC servomotor can be monitored from the cell phone. The screens below (*Figure 6*) show the actual output while operating on the servomotor and also show the outputs having the rated speed and time constant of the servomotor.



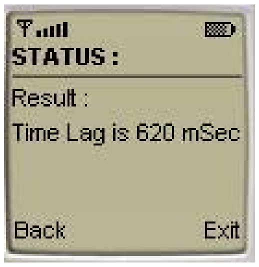
Servo Motor Control
Choose :
ON/OFF Status
Monitor speed
Experiment
Exit Ok



STATUS :
Result:
Motor is OFF
Back On



STATUS :
Result :
Current speed of
Servo motor is 2142
RPM
Back Exit



STATUS :
Result :
Time Lag is 620 mSec
Back Exit

Figure 6

4 CONCLUSIONS

The control of the stepper motor and servomotor from mobile device has been successfully achieved. The stepper motor can be extended to control the arm of a robot or the view of a camera mounted on a remote object and can be controlled remotely. Also done is an experiment of calculating the time lag of the servomotor. This can be extended to remote study of motors and monitoring of motors and study of motors through the Internet.

4.1 SUGGESTIONS FOR FURTHER WORK

The work reported can further be extended to represent the whole machines lab and this could be further uploaded on the Internet on the home page of our institution.

5 REFERENCES

[1] C. Enrique Ortiz and Eric Giguere, 2001- *Mobile Information Device Profile for Java 2 Micro Edition (J2ME): Professional Developer's Guide -* John Wiley & Sons; 1st Edition

[2] John W. Muchow, 2001 - *Core J2ME Technology-* Prentice Hall PTR; 1st Edition