A Programmable Accessory for Cell Phones

Rohit Chaudhri[1, 2], Kentaro Toyama[1]

[1]Technology for Emerging Markets
Microsoft Research India

[2]Computer Science and Engineering
University of Washington, Seattle.

Low-tier cell phones like the Nokia 1100, 1200, 2600 etc, are commonly used by individuals from low-income groups in developing regions [1]. While these phones are cheap and affordable (within the $20-$30 range), they do not have a programmable runtime environment like Android, J2ME, Symbian, Windows Mobile etc, available on mid to high tier mobile phones. Hence mobile application developers are unable to create applications for these phones. This restricts the services that can be delivered to users of such phones.

Given that software-only extensibility is not possible on low-tier mobile phones, we are exploring a combined hardware-software approach to achieve this. In this report we present the architecture and design of a battery-powered, microcontroller-based programmable accessory for cell phones. The idea is to use the accessory for general purpose computation and leverage the cell phone for IO and communications. The accessory connects to a cell phone over the phone's data port and communicates using a serial protocol. GSM and CDMA cell phones typically implement an AT command interface [2],[3]; that is accessible over a serial link. Additionally, if AT commands are not implemented on the phone, cell phone vendors implement some proprietary protocol. For instance, some low-tier Nokia phones implement the FBUS (Fast-Bus) [4] serial protocol instead of the standard AT command-set.

It is possible to send/receive SMSs and initiate/answer phone calls over the serial link. This enables the accessory to communicate with other cell phones and services hosted in the Cloud. Cellular information like the current cell tower-id and signal strength is also accessible over this interface. Key presses on the phone generate DTMF tones; these can be captured on the accessory if it has a DTMF decoder IC. This could be used to send commands to the accessory. Additionally, the accessory can also send text to be displayed on the phone's screen – this could be used to provide feedback to the user and/or prompt the user for input etc. Since the accessory is battery-powered, it can function when not connected to the cell phone. In this "disconnected" mode of operation, the accessory could be used to aggregate data on the field and then, at a later point in time, connected to a cell phone to upload the data to a remote server if needed. These capabilities broadly enable 2 classes of applications:

Tracking: Cellular network information available from the phone enables location tracking. This includes being able to track the person carrying the phone or track an asset, e.g. a transportation company could use this to track their cargo. Since phone usage information (phone calls, SMS) is also available on the accessory, it could be used to track how phones are being used by people. This information could be of interest to researchers who study the use of mobile phones by low-income users in developing regions.

Services: The second class of applications are those that leverage the accessory as a general purpose, programmable platform. For instance, the accessory could be used to host services that are accessible over SMS, e.g. a service could interact with a user to aggregate information about crop prices and make this information available to buyers over SMS.

Architecture

A couple of architectural instantiations of the system are shown in the block diagrams below. Figure 1 shows an accessory connected to a cell phone over the data port. The phone's serial command interface is accessible over this link. The accessory in this figure has a DTMF decoder integrated with the processor. This enables it to accept and interpret key pad input from the phone. Each key press on a cell phone generates a unique DTMF tone; in addition to being transmitted over the cellular channel, these tones are also available on the phone's audio port. Hence, as shown in the figure, if the phone's audio port is also connected to the accessory, it would be able to capture and decode key presses from the phone.

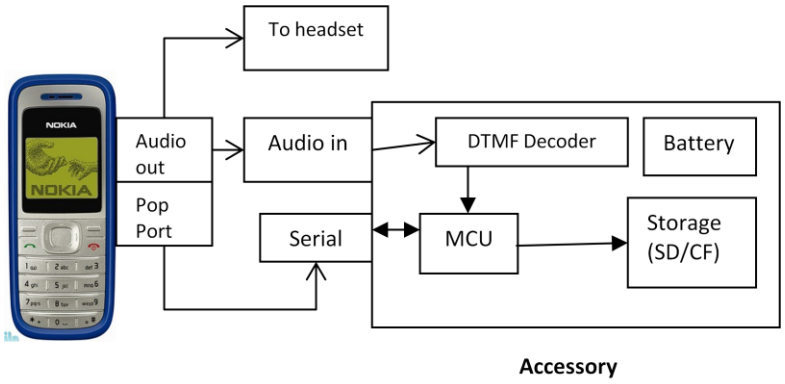

Figure 1: Accessory with DTMF Decoder

The system depicted in Figure 2 does not have a DTMF decoder integrated into the accessory and hence the connection between the phone's audio port and the accessory does not exist. This is essentially a subset of the system shown in Figure 1.

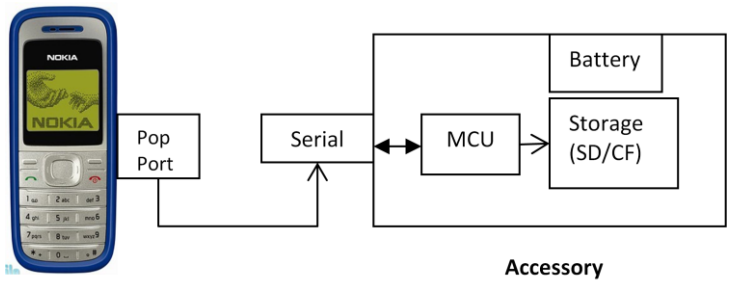

Figure 2: Accessory without DTMF Decoder

Hardware

In the current implementation, the accessory is based on NXP's LPC2148 processor [5], which is an ARM7 [6] core. An ARM7 processor was chosen because of its low-cost, support for low-power modes and possibility to extend the platform using any of the available IO interfaces (UART, SPI, I2C, USB and GPIO). The figures below depict the schematic that was used to build the accessory. Figure

3 shows the power subsystem. A rechargeable Lithium ion (Li-ion) battery is used as the power source for the accessory. When fully charged, a Li-ion battery typically supplies 4.2v of DC power. A regulator, AMS1117, is used to lower the voltage to 3.3v, which then drives the system. Figure 5 shows the LPC2148 and the processor pins being used in the accessory. The processor has 2 UARTs - UART0 (TXD0, RXD0, GND) and UART1 (TXD1, RXD1, GND). UART0 is used for communicating with a PC. This is needed for downloading firmware onto the processor and for printing debug messages over a serial console on the PC. UART1 is used for communicating with a connected cell phone. The RESET and P0.14 pins are used to initiate the firmware upgrade procedure (described in detail in [7]). Pins P0.25, P0.28-P0.30 are not being used currently and are available on the SIP connector for future use. Figure 4 shows the SD adapter that is connected to the LPC2148 over the SPI bus.

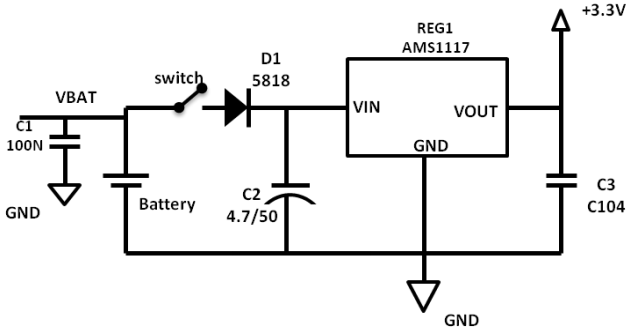

Figure 3: The Power Subsystem

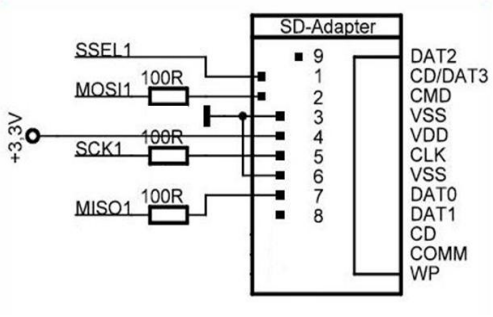

Figure 4: SD Adapter

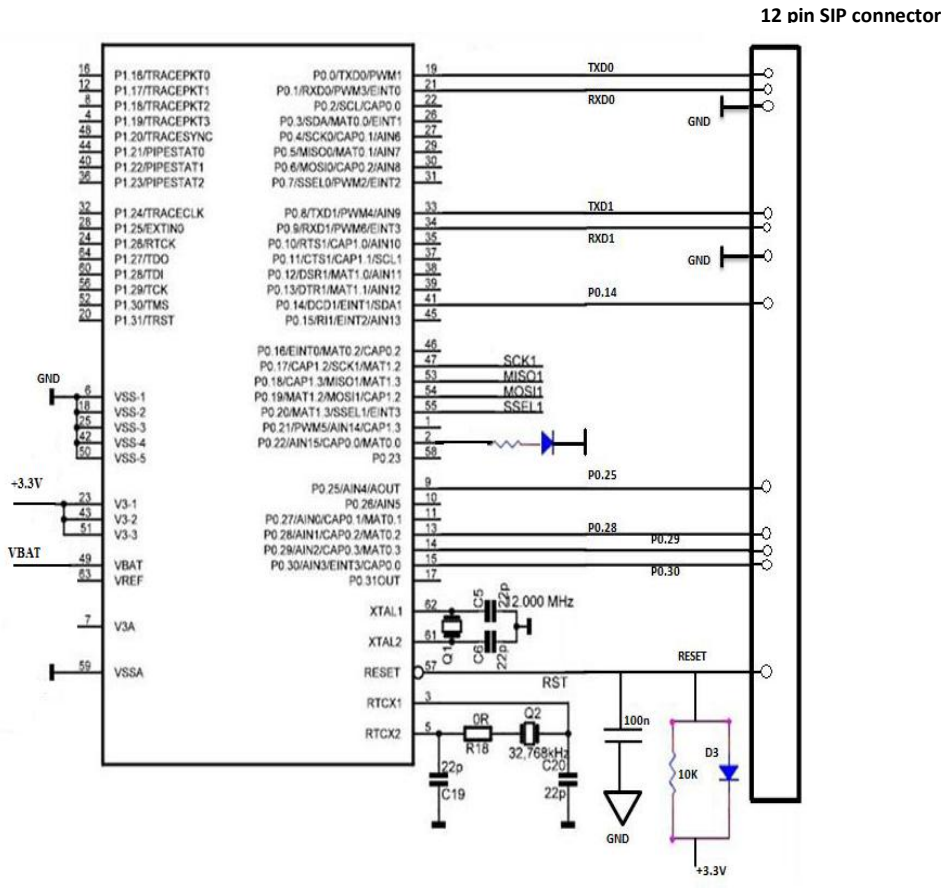

Figure 5: LPC2148

Figure 6 below shows the accessory connected to a Nokia 1650 cell phone.

Figure 6: Accessory connected to Nokia 1650

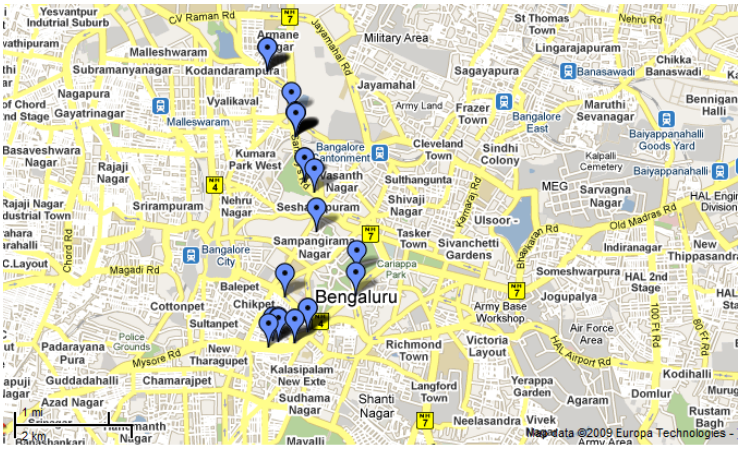

Figure 7: Location trace from accessory

Software

Application development for the ARM7 can be done in C/C++ using any of the available ARM toolchains. We are using Codesourcery's GNU toolchain for ARM [8] for our work. Additionally, NXP publishes processor-specific code bundles/application notes [5] that can be adapted and used in applications. As seen in the figures above, the accessory has an SD card slot for persistent storage. We are using Chan's FatFs library [9], a small footprint implementation of the FAT file system, to interface with SD memory cards over the SPI bus.

Sample application – Cell tower-id based location tracking

In one application developed for the accessory, we use the cell phone's current cell tower-id to derive an approximate location of the phone. The accessory, connected to a low-tier, non-programmable cell phone, is programmed to periodically query the phone's current cell tower-id. This id is then written into a file on the SD card. We process the data offline to get the geographical coordinates of the cell towers using a Google Web Service [10], which gives an approximation of the trip route of the user/cell phone. Figure 7 shows a location trace derived using this system from a round trip (~8 km one-way) that originated at Microsoft's office building in Bangalore. While this is a sample application to demonstrate the capabilities of the system, a more realistic application might upload cell tower-ids to a server in real time via SMS messages. The server could be part of a location tracking system deployed by an enterprise.

Conclusion and future work

In this report we have discussed some early work about an approach to enable service delivery on low-tier cell phones. Given the large scale penetration of low-tier cell phones in developing regions [1], we believe this approach has potential to extend capabilities of such phones and improve the services that can be delivered to users. The project is still in its early days and much needs to be done to take the platform forward:

- We need to investigate the phone-accessory IO channel further. Our current implementation adheres to the architecture shown in Figure 2. In order to process key-presses sent to the accessory from the connected cell phone, we will integrate a DTMF decoder with the hardware in a future iteration. We also need to add capabilities to the accessory that will enable it to display text on the phone's screen.

- Our accessory is based on a general purpose, programmable platform. Cell tower-id based location services seem to be an interesting class of applications that this platform might enable, but more thought is needed for other applications that could be enabled by the platform.
- We need to do field trials to evaluate the system and validate our approach.

We welcome any comments/feedback from the community about the platform and possible applications.

References

[1] Nokia 1100 sales: http://www.engadget.com/2007/05/07/nokias-1100-handset-over-200-million-served/

[2] GSM AT Command-set: http://www.3gpp.org/ftp/Specs/html-info/27007.htm

[3] Nokia CDMA AT Commands: http://www.forum.nokia.com/info/sw.nokia.com/id/537978cf-3542-41ea-a220-9a0bbbb83580/AT_Command_Set_For_Nokia_CDMA_Products_v1_1_en.pdf.html

[4] FBUS: http://en.wikipedia.org/wiki/FBus

[5] NXP LPC2148: http://www.nxp.com/#/pip/pip=[pip=LPC2141_42_44_46_48_4]|pp=[t=pip,i=LPC2141_42_44_46_48_4]

[6] ARM7: http://www.arm.com/products/CPUs/families/ARM7Family.html

[7] Philips ISP: http://www.standardics.nxp.com/support/documents/microcontrollers/pdf/an10302.pdf

[8] ARM Toolchain: http://www.codesourcery.com/sgpp/lite/arm

[9] FatFs: http://elm-chan.org/fsw/ff/00index_e.html

[10] Google Web Service: http://www.google.com/glm/mmap (this supports HTTP POST only, so a HTTP GET request from a browser will return an error)