Paper presentation on

EMBEDDED SYSTEMS

Smart Phone: An Embedded System for Universal Interactions

Author Info Sheet:

AUTHORS:

G.Raga Ranjitha
II B.Tech
Email :ragaranjitha_g@yahoo.com
Contact No: 810681257,

B.Priyanka
II B.Tech
Email: priyankab291@gmail.com
Contact no: 8121992442

INSTITUTION NAME:

ST.MARY'S GROUP OF INSTITUTIONS GUNTUR

GUNTUR
(Affiliated to JNTU KAKINADA)

Abstract:

In this paper, we present a system architecture that allows users to interact with embedded systems acted in their proximity using Smart Phones. We have identified four models of interaction between a Smart Phone and the surrounding environment: universal remote control, dual connectivity, gateway connectivity, and peer-to-peer. Although each of these models has different characteristics, our architecture provides a unique framework for all of the models. Central to our architecture are the hybrid communication capabilities incorporated in the Smart Phones. These phones have the unique feature of incorporating short-range wireless connectivity (e.g., Bluetooth) and Internet connectivity (e.g., GPRS) in the same personal mobile device. This feature together with significant processing power and memory can turn a Smart Phone into the only mobile device that people will carry wherever they go.

Introduction:

Recent advances in technology make it feasible to incorporate significant processing power in almost every device that we encounter in our daily life. These embedded systems are heterogeneous, distributed everywhere in the surrounding environment, and capable of communicating through wired or wireless interfaces. For a number of years, visionary papers [21, 18] have presented a picturesque computerized physical world with which we can potentially interact faster and in a simpler fashion. People, however, are not yet taking advantage of this ubiquitous computing world. Despite all the computing power lying around, most of our daily interactions with the surrounding environment are still primitive and far from the ubiquitous computing vision. Our pockets and bags are still jammed with a bunch of keys for the doors we have to open/close daily (they did not change much since the Middle Ages), the car key or remote, access cards, credit cards, and money to pay for goods. Any of these forgotten at home can turn the day into a nightmare. If we travel, we also need maps and travel guides, coins to pay the parking in the city, and tickets to take the train or subway. In addition, we are always carrying our mobile phone, which for some mysterious reason is the least likely to be left at home. When we finally arrive home or at the hotel, we are "greeted" by several remote controls eager to test our intelligence. All these items are absolutely necessary for us to properly interact with our environment. The problem is that there are too many of them, they are sometimes heavy, and we will likely accumulate more and more of them as our life go on, requiring much larger pockets.

For this problem, the community does not lack innovative solutions that address some of its aspects (e.g., wireless micro servers [15], electronic payment methods [1, 8], and digitaldoor keys [13]). What is missing is a simple, universal solution, which end-users are likely to accept easily. Ideally, we would like to have a single device that acts as both personal server [20] and personal assistant for remote interaction with embedded systems located in proximity of the user. This device should be programmable and support dynamic software extensions for interaction with newly encountered embedded systems (i.e., dynamically loading new interfaces). To simplify its acceptance by society, it should be a device that is already carried by people wherever they go.

We believe that *Smart Phones* are the devices that have the greatest chance of successfully becoming universal remote controls for people to interact with various devices from their surrounding environment; they will also replace all the different items we currently carry in our pockets. Smart Phone is an emerging mobile phone technology that supports Java program

execution and provides both short range wireless connectivity (Bluetooth) and cellular network connectivity through which the Internet can be accessed.

In this paper, we present a system architecture that allows users to interact with embedded systems located in their proximity using a Smart Phone. We have identified four models of interaction between a Smart Phone and the surrounding environment: universal remote control, dual connectivity, gateway connectivity, and peer-to-peer. Although each of these models has different characteristics, our architecture provides a unique framework for all the models. Central to our architecture are the hybrid communication capabilities incorporated in the Smart Phones which allow them to interact with the close-by environment through short-range wireless networking and with the rest of the world through the Internet over cellular links. This feature together with significant processing power and memory can turn a Smart Phone into the long awaited universal personal assistant that can make our daily life much simpler.

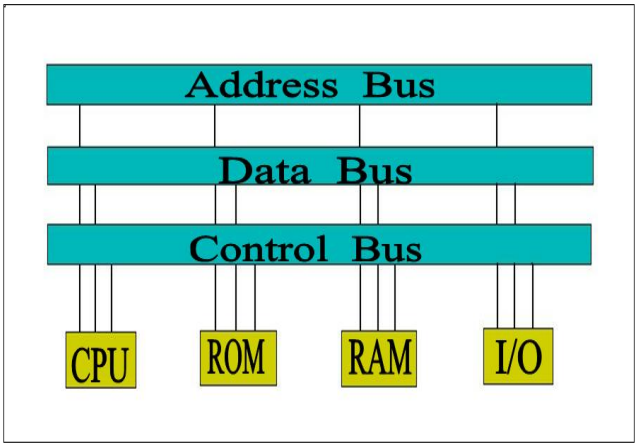
An Embedded System:

The embedded system is a combination of computer hardware, software and, perhaps, additional mechanical parts design to perform a specific function. A good example is an automatic washing machine or microwave oven .Such a system is in direct contrast to a personal computer, which not designed to do only a specific task. The PC aids you in drafting a letter, in computing at a faster rate in chatting with friends, and so on, but an embedded system is designed to do a specific task with in a given time frame, repeatedly, endlessly, with or without human interaction. A PC is made up of numerous embedded systems, such as a keyboard, hard drive etc. The function of a simple modem is to convert analogue signals to digital signals, and vice versa. This means it must have a certain amount of logic to perform that process in time and again endlessly.

It is important to note that all embedded systems all embedded systems do not have same hardware and software, which is why these systems perform varied tasks. It's even possible to have an embedded system that does not contain any processor and corresponding software to run through it. In such system, called hardwired systems, the hardware and software is replaced with integrated circuitry that performs a same function .However, a lot of flexibility is lost when applications are implemented this way. It is much easier to change the software code than to redevelop the hardware, for bringing about the small changes in application for which the system has been designed.

Embedded Hardware:

All embedded systems need a micro processor, and the kinds of microprocessors using them are quite varied. A list of some of the common micro processor families are ZILOG Z8 family, INTEL 8051/80188/X86 family. An embedded system also needs memory for two purposes – to store its program, and to store its data. Embedded systems store data and programs in different memories .This is simply because embedded system does not have an hard drive and the program must be stored in memory, even when the power is turned off .This special memory that remembers program even without power, is called ROM Very often this systems have a serial port I/O interfaces, or hard ware to interact with sensors.



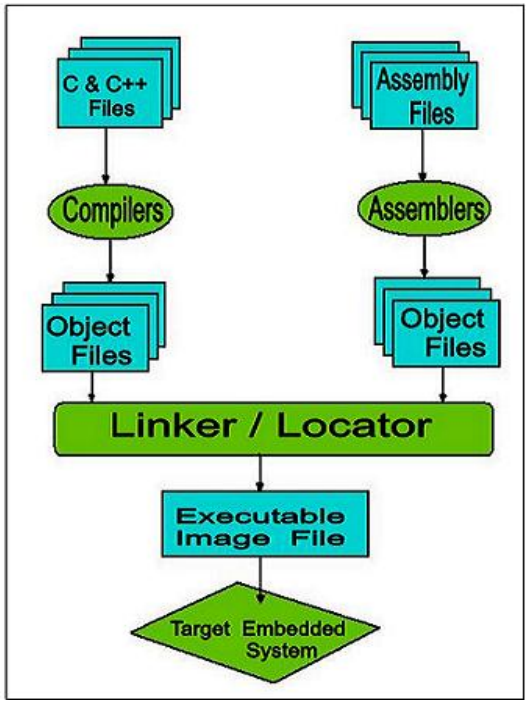
So an embedded system has a micro processor or micro controller for

processing information, memory for storing embedded software programs and data and I/O interfaces for external interfaces. The functional diagram is given above.

The processor uses the address bus to select a specific memory location within the memory sub system or a specific peripheral chip. The data base is used to transfer data between the processor and memory sub system or peripheral devices the control bus provides timing signals to synchronize the flow of data between the processor and memory sub system or peripheral devices.

Embedded Software:

C has become the language of



choice for embedded programmers. The greatest strength of C is that it gives embedded programmers an extraordinary degree of direct hardware control without sacrificing the benefits of high level languages. Compilers and cross compilers are also available for almost every processor with C.

Any source code written in C or C++ or assembly must be converted into an executable image that can be loaded onto a ROM chip. The process of converting the source code representation of your embedded software into an executable image involves three distinct steps, and the system or computer on which these processes are executed is called Host computer.

First, each of the source files that make an embedded application must be compiled or assembled into distinct object files. Second, all of the object files that result first step must be linked into a final object file called the relocatable program. Finally physical memory address must be assigned to relocatable program. The result of the third step is a file that contains an executable image that is ported on to the ROM chip. This ROM chip, along with the processor and other devices and interfaces, makes an embedded system run

There are some very basic differences between conventional programming and embedded programming. First, each target platform is unique. Even if the processor architecture is the same, the I/O interfaces or sensors or activators may differ. Second, there is a difference in the development and debugging of applications.

Smart Phones Technology:

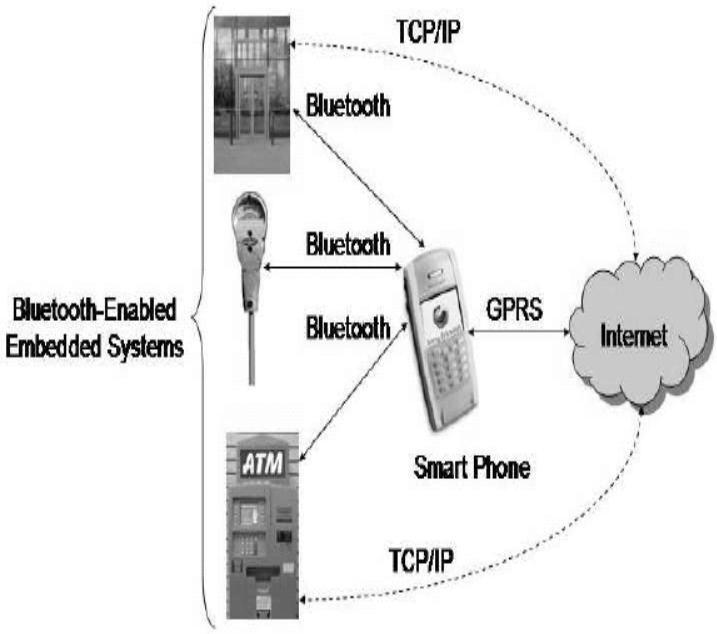
With more than a billion mobile phones being carried around by consumers of all ages, the mobile phone has become the most pervasive pocket-carried device. We are beginning to see the introduction of *Smart Phones*, such as Sony Ericsson P800/P900 [9] and Motorola A760 [10] (Figure 1), as a result of the convergence of mobile phones and PDA devices. Unlike traditional mobile phones, which have limited processing power and act merely as “dumb” conduits for passing voice or data between the cellular network and end users, Smart Phones combine significant computing power with memory, short-range wireless interfaces (e.g., Bluetooth), Internet connectivity (over GPRS), and various input-

output components (e.g., high-resolution color touch screens, digital cameras, and MP3 players). Sony Ericsson P800/P900 runs Symbian OS [12], an operating system specifically designed for resource constrained devices such as mobile phones. It also comes equipped with two versions of Java technology: Personal Java [11] and J2ME CLDC/MIDP [2]. Additionally, it supports C++ which provides low level access to the operating system and the Bluetooth driver. The phone has 16MB of internal memory and up to 128MB external flash memory. Motorola A760 has a Motorola i250 chip for communication, Intel's 200 MHz PXA262 chip for computation, and 256MB of RAM memory. It runs a version of MontaVista Linux and comes with Java J2ME support [2].Bluetooth [7] is a low-cost, low-power standard for wireless connectivity. Today, we can find Bluetooth chips embedded in PCs, laptops, digital cameras, GPS devices, Smart Phones, and a whole range of other electronic devices. Bluetooth supports point-to-point and point-to-multipoint connections. We can actively connect a Bluetooth device to up to seven devices simultaneously. Together, they form an ad hoc network, called *Piconet*. Several piconets can be linked to form a *Scatter net*. Another important development for the mobile phone technology is the introduction of General Packet Radio Service (GPRS) [3], a packet switching technology over the current GSM cellular networks. GPRS is offered as a no voice value-added service that allows data to be sent and received across GSM cellular networks at a rate of up to 171.2kbps, and its goal is to supplement today's Circuit

Example of Smart Phones: Sony Ericsson P800 (Left) and Motorola A760 (Right)

Smart Phone Interaction Models:

A Smart Phone can be used to interact with the surrounding environment in different ways. We have identified four interaction models: universal remote control, dual connectivity, gateway connectivity, and peer-to-peer. With these models, a Smart Phone can be used to execute applications from as simple as remotely adjusting various controls of home appliances or opening smart locks to complex applications such as automatically booking a cab or ordering/paying in a restaurant using an ad hoc network of mobile phones to connect to the cashier's computer.



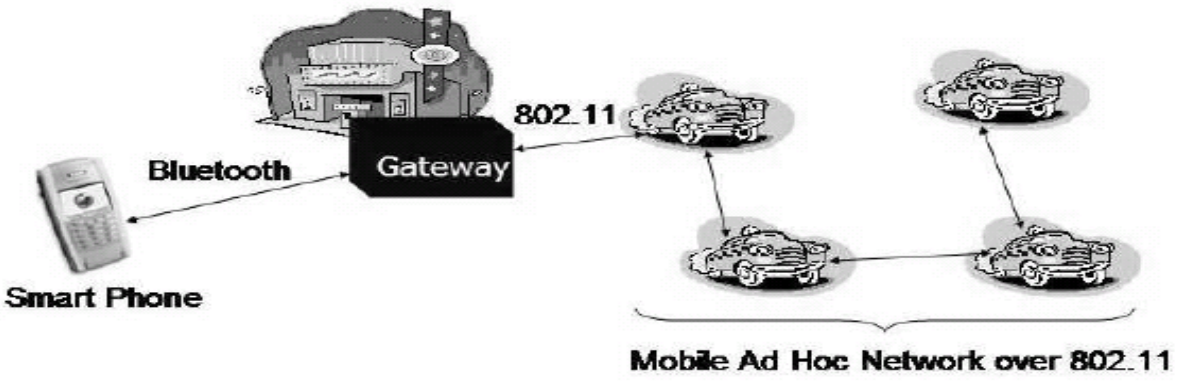
Universal Remote Control Interaction Model. Peer-to-Peer Interaction Model

Universal Remote Control Model:

The Smart Phone can act as a universal remote control for interaction with embedded systems located in its proximity. To support proximity-aware interactions, both the Smart Phone and the embedded systems with which the user interacts must have short-range wireless communication capabilities. Figure 2 illustrates such

interactions using Bluetooth. Due to its low-power, low- cost features; Bluetooth is the primary candidate for the short-range wireless technology that will enable proximity-aware communication.

Since embedded systems with different functionalities can be scattered everywhere, a discovery protocol will allow Smart Phones to learn the identity and the description of the embedded systems located in their proximity. This protocol can work either automatically or on-demand, but the information about the devices currently located in user's proximity is displayed only upon user's request. Each embedded system should be able to provide its identity information (unique to a device or to a class of devices) and a description of its basic functionality in a human-understandable format. This model works well as long as the user has the interfaces for interacting with the embedded systems preinstalled on the phone. An alternative, more flexible, solution is to define a protocol that allows a Smart Phone to learn the interfaces from the embedded systems themselves. The problem with this idea is that many embedded systems may not be powerful enough to run complex software that implements such protocols. In the following, we describe a second model of interaction that solves this problem.



❖ **Gateway Connectivity Model :**

Many pervasive applications assume wireless communication through the IEEE 802.11 family of protocols. These protocols allow for a significant increase in the communication distance and bandwidth compared to Bluetooth. Using these protocols, the communication range is 250m or more, while Bluetooth reaches only 10m. The bandwidth is also larger, 11-54Mbps compared to less than 1Mbps for Bluetooth. Additionally, many routing protocols for mobile adhoc networks based 802.11 already exist [19, 16]. The disadvantage of 802.11 is that it consumes too much energy, and consequently, it drains out the mobile devices' batteries in a very short period of time. With the current state of the art, we do not expect to have 802.11 network interfaces embedded in Smart Phones or other resource constrained embedded systems that need to run on batteries for a significant period of time (e.g., several hours or even days). More powerful systems, however, can take advantage of the 802.11 benefits and create mobile ad hoc networks. In such a situation, a user would like to access data and services provided by these networks from its Smart Phone.

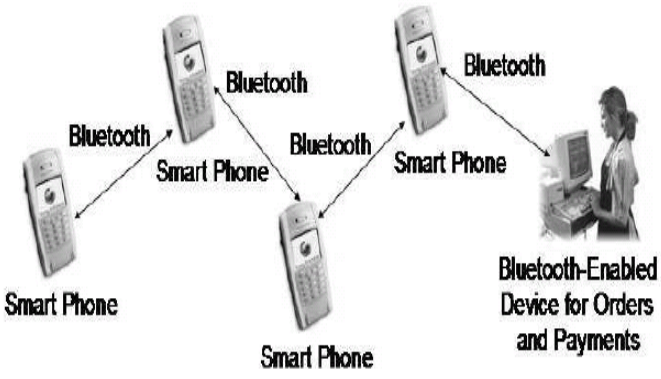
The Gateway Connectivity Interaction Model.

To succeed, a gateway device has to perform a change of protocol from Bluetooth to 802.11 and vice-versa. Many places in a city (e.g., stores, theaters, restaurants) can provide such gateway stations together with 802.11 hotspots. Figure 4 illustrates this communication model and also presents an application that can be built on top of it. Let us assume a scenario where people want to book nearby cabs using their

Smart Phones. Instead of calling a taxi company or ”gesturing” to book a cab, a client can start an application on her Smart Phone that seamlessly achieves the same goal. Hence, the client is just one-click away from booking a cab. In this scenario, each cab is equipped with 802.11 wireless networking and GPS devices, and the entire booking process is completely decentralized. To join the mobile adhoc network created by the cabs, a Smart Phone needs to connect to a *gateway* station that performs a translation of protocols from Bluetooth to 802.11 and vice-versa.

❖ **Peer-to-Peer Model :**

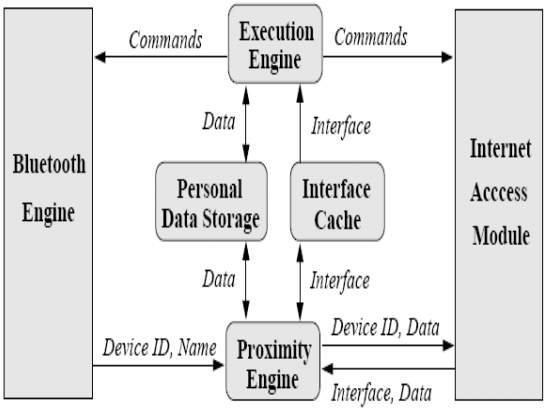
The Smart Phones can also communicate among themselves (or with other Bluetooth-enabled devices) in a multihop, peer-to-peer fashion, similar to mobile ad hoc networks. For instance, this model allows people to share music and pictures with others even if they are not in the proximity of each other. Figure depicts yet another example of this model. A group of friends having dinner in a restaurant can use their Smart Phones to execute a program that shares the check. One phone initiates this process, an ad hoc network of Smart Phones is created, and finally the payment message arrives at the cashier.



The Peer-to-Peer Interaction Model.

System Architecture:

Our system architecture for universal interaction consists of a common Smart Phone software architecture and an interaction protocol. This protocol allows Smart Phones to interact with the surrounding environment and the Internet. Figure shows the Smart Phone software architecture. In the following, we briefly describe the components of the software architecture.



Smart Phone Software Architecture.

. Bluetooth Engine is responsible for communicating with the Bluetooth-enabled embedded systems. It is composed of sub-components for device discovery and sending/receiving data. The Bluetooth Engine is a layer above the Bluetooth stack and provides a convenient Java API for accessing the Bluetooth stack.

- Internet Access Module carries out the communication between the Smart Phone and various Internet servers. It provides a well-defined API that supports operations specific to our architecture (e.g., downloading an interface). The protocol of communication is HTTP on top of GPRS.

- Proximity Engine is responsible for discovering the embedded systems located within the Bluetooth communication range. Each time the user wants to interact with one of these systems, and an interface for this system is not available locally (i.e., a miss in the Interface Cache), the Proximity Engine is responsible from downloading such an interface. If the embedded system has enough

computing power and memory, the interface can be downloaded directly from it. Otherwise, the Proximity Engine invokes the Internet Access Module to connect to a web server and download the interface. The downloaded interface is stored in the Interface Cache for later reuse. Once this is done, the Proximity Engine informs the Execution Engine to dispatch the downloaded interface for execution. All further communication between the Smart Phone and the embedded system happens as a result of executing this interface.

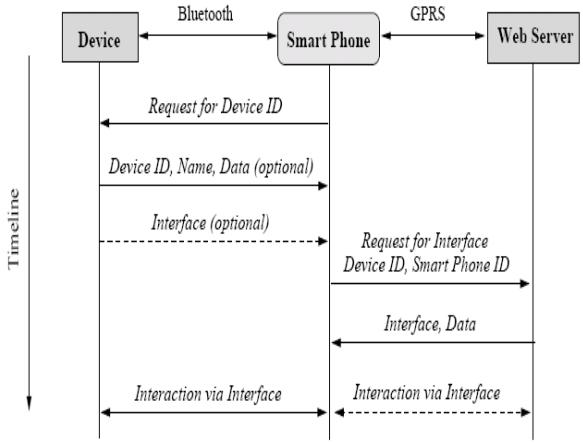
- Execution Engine is invoked by the Proximity Engine and is responsible for dispatching interface programs for execution over the Java virtual machine. These programs interact with the Bluetooth Engine to communicate with the embedded systems or with other Smart Phones They may also interact with the Internet Access Module to communicate with Internet servers. For instance, the interface programs may need to contact a server for security related actions or to download necessary data in case of a miss in the Personal Data Storage.

- Interface Cache stores the code of the downloaded interfaces. This cache avoids downloading an interface every time it is needed. An interface can be shared by an entire class of embedded systems (e.g., Smart Locks, or Microwaves). Every interface has an ID (which can be the ID of the embedded system or the class of embedded systems it is associated with). This ID helps in recognizing the cached interface each time it needs to be looked up in the cache. Additionally, each interface has an associated access handler that is executed before any subsequent execution of the interface. This handler may define the time period for which the interface should be cached, how and when the interface can be reused, or the permissions to access local resources. The user can set the access handler's parameters before the first execution of the interface.

- Personal Data Storage acts as a cache for "active data", similar to Active Cache [14]. It stores data that needs to be used during the interactions with various embedded systems. Examples of such data include digital door keys and electronic cash. Each data item stored in this cache has three associated handlers: access handler, miss handler, and eviction handler. Each time an interface needs some data, it checks the Personal Data Storage. If the data is available locally (i.e., hit), the access handler is executed, and the program goes ahead. For instance, the access handler may check if this data can be shared among different interfaces. If the data is not available locally (i.e., miss), the miss handler instructs the Internet Access Module to download the data from the corresponding Internet server. The eviction handler defines the actions to be taken when data is evicted from the cache. For instance, electronic cash can be sent back to the bank at eviction time.

Below figure shows the interaction protocol that takes place when a Smart Phone needs to interact with an embedded system. We consider that any embedded system is registered with a trusted web server (this web server can be physically distributed on multiple computers). At registration, the web server assigns a unique ID and a URL to the device. All the information necessary to interact with the device along with a user interface is stored at that URL. This URL may be common for an entire class of embedded systems. The user invokes the Proximity Engine each time she needs to interact with a device located in the proximity. Once the embedded systems in the proximity have been identified, the user can choose the one she wants to interact with. Consequently, a request is sent to the embedded system to provide its ID and URL. Upon receiving the ID and URL of the embedded system, the Smart Phone executes the access control handler, and then, loads and executes the interface. In case of a miss in the Interface Cache, the interface needs to be downloaded on the phone either from the web server or from the embedded system

itself. An inter face downloaded from an embedded system is untrusted and is not allowed to access local resources (i.e., this is a sandbox model of execution, where the interface can only execute *safe* instructions on the phone). The interfaces downloaded from the web server are trusted; they are assumed to be verified before being distributed by the server. Each time a Smart Phone requests an interface from the web server, it has to send the interface ID and the URL provided by the embedded system. It also sends it's ID (stored in the Personal Data Storage). The permission to download an interface is subject to access control enforced based on the Smart Phone ID and, potentially, other credentials presented by the user. Once the access is granted, the web server responds with the interface code.



Smart Phone Interaction Protocol

Status and Future Work:

In this section, we briefly outline the current status and several open issues that we have to overcome in order to implement our system architecture. We are in the process of building the system architecture on top of Ericsson's P800/900 phones. Our first step consists of implementing the basic architecture for the universal remote control interaction model. The architecture components to be developed for this model are the Bluetooth Engine and Proximity Engine along with a simple Execution engine over Java. We have partially implemented the Bluetooth Engine and have written and tested a few sample programs to test the feasibility of connecting a phone to another phone or to a Bluetooth-enabled laptop. Besides directly connecting to Bluetooth-enabled devices, a phone can also connect to a LAN. We are in the process of investigating the feasibility of using the Bluetooth LAN profile to connect the phone to a LAN through a Bluetooth access point. Until recently, the commercially available Bluetooth chips have been working well for one-hop communication, but their scatter net capabilities have not been mature enough to support multi-hop communication as needed in our peer-to-peer interaction model. Currently, there are products [4], however, whose scatter net capabilities have been successfully tested. We envision that multi-hop communication in ad hoc networks will take place either over Bluetooth or over 802.11 depending on the trade-offs between the battery power consumption and communication range. Our system architecture supports both situations through the peer-to-peer model and the gateway model, respectively. To connect a Smart Phone to the Internet over GPRS, we can use HTTP or TCP. A decision regarding the protocol used for Internet access needs to consider the trade-offs between the simplicity provided by HTTP and the flexibility and efficiency provided by TCP. Although our architecture provides a level of security by obtaining interface code and confidential data from a trusted web server, many issues related to security and privacy still need to be addressed. For instance, we need to investigate different lightweight encryption algorithms that work on resource constrained devices to counter eavesdropping without a serious overhead. So far we have assumed that the personal information of the user, including confidential data, would be stored on the Smart Phone. In such a situation, losing the Smart Phone could pose a serious security threat to the owner. The data stored on the phone should be made inaccessible to anyone but the phone owner. A simple

password scheme is insufficient because entering a password every time confidential data is accessed could be a major turn off for the users. We plan to investigate both software protection mechanisms and hardware solutions..

Conclusions:

In this paper, we have argued for turning the Smart Phone into the only device that people carry in their pockets wherever they go. The Smart Phone can be used as both personal server that stores or downloads data that its user needs and personal assistant for remote interaction with embedded systems located in the user's proximity. To achieve this vision, we have presented unified system architecture for different models of interaction between a Smart Phone and the surrounding environment. Central to this universal interaction architecture is the dual connectivity feature of Smart Phones, which allows them to interact with the close-by environment through short-range wireless networking and with the rest of the world through the Internet over cellular links.