# Bluetooth Enabled Embedded Linux

**Xiaoyong David Yang and Chip-Hong Chang**
Center for High Performance Embedded Systems
Nanyang Technological University
N4, B3b-06, Nanyang Avenue, Singapore, 639798
linuxprotocol@hotmail.com

## Introduction:

In the fast evolving wireless communication system, Software Defined Radio (SDR) has emerged as a feasible solution to overcome the compatibility issue of various communication standards. There is a pressing need to have stable, low-cost, open platforms to support the configurable and standard (restriction) free architecture. The goal of SDR perfectly matches the nature of Linux: zero licensing cost, robust architecture, flexible networking capabilities and open source. Due to the obvious synergy of open source and open standard, Bluetooth, the popular unified wireless short-range communication standard, is chosen to partner with Linux to approach the SDR design from bottom up. This project consists of four primary steps, namely the feasibility study, the implementation of the embedded Linux and the GUI system, incorporation of applications to test the functionality and the development of the Bluetooth protocol stack. Figure 1 shows the targeted achievements at each step.

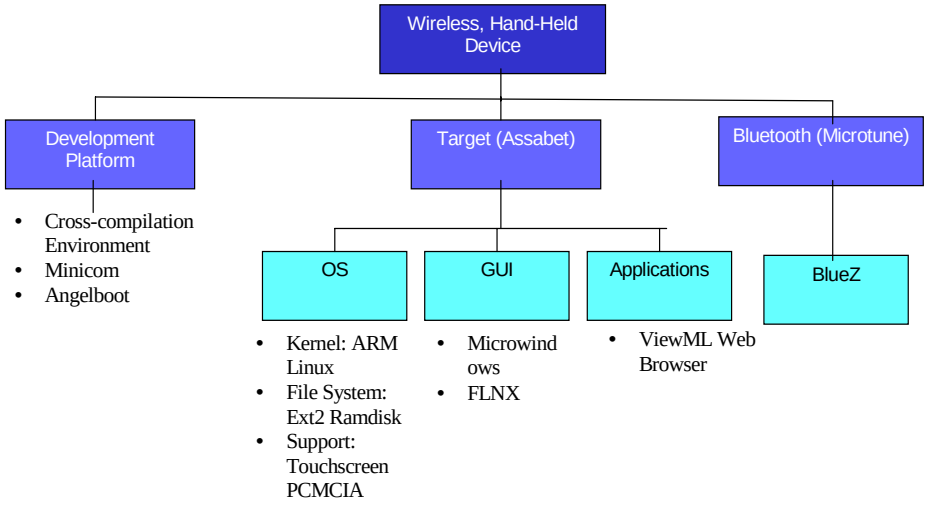


**Figure 1** Overview of tasks accomplished

This project investigates the feasibility of using a version of embedded Linux (ARM Linux) on an Intel 206MHz Strong ARM based system, the Assabet evaluation board. The feasibility study required the setting up of a windowing system on the embedded device by cross-compiling Microwindows running under ARM Linux kernel, the configuration of FLNX to operate with Microwindows to allow the execution of numerous FLTK based programs, the configuration of the file system on the Ramdisk residing in the Assabet evaluation board and the modification of the touch screen driver to ensure proper functionality of the input process. A web browser and support for Ethernet were also included. Through the comparison with various open sourced Linux Bluetooth protocol stacks Openbt, BlueDrekar and Affix, the BlueZ is selected due to its robustness, friendliness and functionality.

# Embedded Linux feasibility study:

An embedded device was originally defined as one, which ran on an embedded processor and was designed to perform *only* certain specific tasks with no GUI to interact with such devices like a network router, or a fridge. However, the technology advancements predicated by Moore's Law in the field of embedded computing anticipates the early birth of the "Post-PC" era. As the pace of hardware and chipset innovation accelerates, operating systems with the following attributes will survive the post-PC era: reduced development time, low cost, good support for a vast array of devices, compliance to international standards and efficient, robust, reliable, modular, and configurable architecture. Compared to other operating systems, the characteristics of embedded Linux are perfectly matched with the attributes of the OS required in the post PC era: Open source helps to decrease the development time and incur no run-time royalties, hence reduces the costs, Besides, it is the only multi-vendor OS supported by the thousands of programmers around the world, making its architecture reliable, secure and stable. The projections by the Embedded Systems Forum 2001 predict that by 2004 Embedded Linux will dominate more than 50% of the embedded software market.

In order to reveal the capability of embedded Linux, the study of the embedded Linux product is conducted. In Table 1, we compared all the six existing Linux PDAs by examining the key features of mobile computing devices: battery life, computing speed and the size or weight, etc.

Table 1. Comparison of the current Linux PDAs

| PDAs | CPU (MHZ) | GUI/Browser | OS |
|---|---|---|---|
| SK Telecom IMT2000 WebPhone | SA1110 (Strong ARM 206 MHz) (2 CPUs) | Qt/Embedded | Tynux Linux |
| Yopy | SA1110 | Yopy CDK | Yopy CDK |
| Agenda VR3 | 66 MHz NEC VR4181 | XFree86 | Linux-VR |
| Helio | 75MHz RISC | Pocket Linux | Pocket Linux |
| Itsy | StrongARM (59-191MHz) | N.A. | ARM Linux |
| DAT500 handheld | AMD Elan SC300 (3 CPUs) | OpenGUI/Microwindows | Caldera Open Linux |
| Compaq iPAQ PDA | SA1110 | Qt Palmtop, Microwindows, etc. | |
| Ericsson cordless screen web phone | SA1110 | Qt/Embedded | |
| Frontpath proGear web pad | 400 Transmeta | Qt/Embedded | |
| Screen media free web pad | 166 MHz MediaGX | Microwindows | |

Other than Linux PDA, another popular application is the Linux web appliance. About one-quarter of the web appliance might be dominated by embedded Linux in the world market by year 2004. So in the last three rows of Table 1 the current 3 Linux web appliances are displayed.

Among all the applications StrongARM SA1110 is the most popular processor, as shown in Table 1. Out of seven Linux PDAs, four use Intel StrongARM, while one of the three Linux web appliances uses StrongARM processor. The SA1110 will be revised when hardware feasibility is discussed.

Table 1 does not give a clear picture of the most popular Linux OS, because Linux is highly modulated than the other PDA OS. The kernel, the core of the embedded Linux can be customized easily for the specific hardware architecture independently with the file system and GUI.

The Linux architecture and kernel ensure the stability and reliability. For PDA, another key factor is to be considered the user friendly GUI, because it does not have a large viewable area and convenient input method. The most popular GUI systems are Trolltech's Qt embedded (QPE) and Microwindows. In Table 1, three hand-held devices use the microwindows, and four use the Qt embedded. In this project the Microwindows is used to support one of the applications, the Palm emulator.

The web browser is an important application for PDA. In Table 1 all the existing web applicants use Opera as the web browsers, only Compaq IPAQ uses ViewML web browser. This is because ViewML works closely with Microwindows and its system requirement is the same as the Palm emulator. It is chosen in this project.

In this project a Palm emulator on embedded Linux is also further developed as part of the scope. Quoted from the Panutat Tejasen, programmer of Palm emulator for WinCE, emulators use high CPU power, to run in the same speed as the real machine - it need more than 10 times the current maximum speed of Palm PDA available in the market. Palm normally runs on a 20 MHz to 30 MHz Dragon Ball processor. The platform used in this project is Intel StrongARM 206 MHZ processor, which meet the 10 times requirement. Once the emulator is running, it will alleviate on the problem of lack of applications and compatibility with Palm.

Another common concern for the embedded system is the memory requirement. In Table 1 most of the PDA or web pad are using 32MB memory space RAM or Flash. The embedded Linux OS in general occupies a memory space larger than 2MB. In this project the final prototype used about 8MB with the ext2 RAM disk filesystem.

In addition to the system feasibility study, the hardware platforms were also examined. For example, the chosen processor had to have a Memory Management Unit (MMU), as the MMU is essential to exploit advantage of the memory protection and virtual memory features offered by the embedded Linux. Other factors that were looked into include the interfaces available to different types of memory, the MIPS and the power consumption ratings.

Table 2. Comparison of short-listed processors

| | Intel StrongARM SA-1110 | NEC MIPS VR4122 | Hitachi (SH3 Series) SH7709A |
|---|---|---|---|
| Speed (MHz) | 206 | 180 | 133 |
| Memory Bus | ROM, SMROM, Flash, SRAM, DRAM, SDRAM, two PCMCIA sockets | ROM, DRAM | ROM, SDRAM, SRAM, DRAM, Supports two PCMCIA sockets |
| MMU | 32-Entry<br>Maps 4Kbyte, 8Kbyte, 1Mbyte | 32-Entry, 1-256Kbyte | 128-Entry<br>Maps 4Kbyte, 8Kbyte |
| Power Management Unit | Normal Mode <240mW<br>Idle Mode <75mW<br>Sleep Mode <50uA | Full speed, Stand by, Suspend, Hibernate | Normal mode,<br>Sleep mode<br>Standby mode |
| Operating Voltage – Core | 1.75V | 1.8V | 1.9V |

Table 2, which shows the comparison of the different processors, illustrates that although the power consumption of the processor was slightly higher than the NEC VR4122 and the Hitachi SH7709A, the SA-1110 has superior performance in terms of MHz ratings, power consumption and memory. Additional considerations for choosing SA1110 are the availability of Linux cross-compilers and the supporting features of evaluation boards. Finally Assabet board using the SA1110 is chosen to develop the required platform.

The Linux software development can take three approaches, using the existing Linux PDA environment, using the Embedded Linux solution, or developing from scratch. We has explored the above three

alternatives, and has decided to adopt a hybrid combination of the last two approaches. The Arm Linux core with other GUI system components is employed.

At the beginning, the focus was on the Linux distributed package to shorten the development time. In Table 3 the short-listed embedded Linux distribution among the top embedded Linux packages are compared based on the survey in Linuxdevices.com.

Table 3. The Comparison Of Short Listed Linux Distributions

| | **Red Hat Ecos** | **uClinux** | **BlueCat Linux** | **Embedix lineo** | **Hard Hat Linux** |
|---|---|---|---|---|---|
| Target CPU | StrongARM CL-PS7111 EP72xx ,CMA2xx, SH3,x86,AM33, MBX, NEC VR4300 | Motorola DragonBall, ColdFire, ARM7TDMI , MC68EN302, Intel i960 | Motorola PowerPC PowerQUIC C II | Power PC MCP750 VEM 2700 | Motorola PowerPC PowerQUICC,8xx/82 60 , IBM PowerPC SA-1100,1110 NEC MIPS |
| Footprint | N. A. | Kernel + tools < 900 kb | N. A. | 8 MB RAM 3 MB Flash | Footprint 0.5 MB |
| Special features | Configuration tool with GUI, Chinese fonts, | Various File System Support | N. A. | N. A. | Telephony protocols, JVM, Hard Hat Net, real-Time scheduler, Chinese fonts |
| Support | $2000- 4000 per year | Databases | Developmen t deployment | Databases | Prototyping, development, deployment >$8000 per year |

Table 4 is a summary of all the embedded Linux. The category on the left-hand side is based on " the factor influencing the choice of the embedded Linux". To meet the objective of this project, categories like User interface, CPU and support for Chinese fonts were taken into the consideration.

Table 4. Further comparison of the seven embedded Linux distributions

| **Categories** | **Short listed Linux** |
|---|---|
| Documentation | Red Hat, Embedix, Blue Cat, Hard hat, Embedix, Uclinux |
| Technical support | Red Hat, Blue Cat, Hard hat, Embedix, Uclinux |
| Ease of installation | Hard Hat & Red Hat |
| Completeness of distribution | Royal Linux (with the board), Hard Hat |
| Reputation | Red Hat & Hard Hat |
| Availability of plug-and-play device drivers | Hard hat |
| "Embedded support" (Real-time extensions, reduced foot-print, etc.) | Hard Hat & Red Hat |
| User Interface | Hard hat shipped with Microwindows |
| CPU support range | Red Hat, Hard hat |
| Ranking in linuxdevices.com | Red Hat |

By carefully analysing Red Hat eCos and MontaVista Hardhat, Hardhat was chosen as the best distributions among all the 7 embedded Linux in terms of the documentation, reputation, software and hardware support.

In the survey, the ranking of Redhat is much higher than Hardhat. However, Ecos only has a Beta version for SA-1110/SA-1111 Development Platform in anonymous CVS, which is known to be buggy, unreliable, and illy maintained by Redhat. In the project the two important considerations was the GUI and Chinese character display. In these two areas Hardhat is superior than Red Hat. Additionally, Hardhat CDK 1.2 was shipped together with Intel Strong ARM SA-1110 Development Platform. Since Intel Corporation has become one of the key investors of MontaVista. Eventually the Hard hat will supersede the Red Hat Ecos.

As the project progressed into the later stage, the disadvantage of Hardhat as a commercial product began to surface. Though Linux is open source, neither the service and support of Hardhat is free, nor the tutorial and mailing list available on the website. We has decided not to rely on the commercial Embedded Linux distribution but the Open Source Embedded Linux Implementations, which might not be bug free but at least it is resourceful. In this project the final kernel is based on ARM Linux, one of the Open Source Embedded Linux Implementations.

ARM Linux is a the successful porting of Linux Operating System to ARM processor-based machines developed mainly by Russell King with the contributions from others. ARM Linux itself currently runs on more than 50 different machine variations, including complete computers, network computers and evaluation boards. There are also projects porting ARM Linux to palm tops. ARM Linux provide ports for many platforms including Itsy and Assabet. ARM Linux has a “read me” file in the /documentation/arm directory of Linux kernel. This shows that ARM Linux is widely accepted for arm processor usage.

For the GUI system, Microwindows and Qt embedded are the most popular small foot print options suitable for SA1110 platform.

Microwindows does not require any operating system or other graphics system support, as it writes directly to the display hardware, although it runs well on Linux frame buffer systems. Qt/Embedded [**8**] provides a full graphics stack, from the hardware interface to a full GUI toolkit. Although the API is identical to the popular Qt/X11 and Qt/Windows products, Qt/Embedded is not based on X11. Consequently, it has substantially reduced memory requirements. All the Qt programmers can easily switch to the Qt /embedded programming.

Qt/Embedded is available under the GNU General Public License (GPL), or can be licensed on other terms from its originator, which is not open-sourced and free of charge. For a time limited project Qt may be the best choice due to its professional support. For a fund limited project Microwindows might be the best approach since it is free of charge and has no loyalty concern.

Despite being new, Microwindows is widely accepted by other embedded Linux distributions. We has selected Microwindows for this project partially because the Palm emulator and ViewML web browser applications need Flnx, and Microwindows work closely with FLTK (FLNX).

The next important finding was the correction of the different errors found in the various scripting files of the ViewML web browser. The solutions to these errors were also relayed back to the mailing list.

# Implementation of the embedded Linux system:

The implementation of the whole system involved the following steps:

## 1. Setup target environment:

The steps mentioned in this section were used to configure and communicate with the target, i.e. the Assabet evaluation board. Tasks accomplished in this section included:

- Setting up and powering up the Assabet
- Installing the Angel boot rom software
- Using Angelboot to boot up the Assabet
- Setting up Minicom to establish a communication channel using the hardhar's reference [14]

The most difficult part among the four tasks is the angelboot set up for Assabet due to the lack of documentation and standard. With the help of the resource at the site ARM Linux on Assabet [9], we analyzed the different version of angelboot with the reference to mainly [14]. We found out and understood the configuration and command for using Nicolas Pitre's *angelboot* utility. Though there is no documentation for the latest version of angelboot, understanding the meaning of the angelboot setting in dot.angelrc file is crucial for the purpose of future development.

Obviously, "device /dev/ttyS0", "9600 8N1", and "baud 115200" are used to set the COM 1. "image zImage" is used to specify the location and the name of the compressed kernel zImage. "entry 0xc0008000 base 0xc0008000" tells the angelboot where to locate the starting of zImage in the RAM. It is decided by the memory map of the Assabet or Burutus. The value must be consistent with file arch/arm/mach-sal100/arch.c in the ARM Linux kernel. Similarly, "otherfile ramdisk_img.gz" is used to load the ext2 ramdisk_img.gz designed by Nicolas Pitre. "otherbase 0xc0800000" is the location to port the ramdisk_img.gz. For Assabet the file arch/arm/mach-sal100/arch.c in the ARM Linux kernel use the same value to set the location and default size of ramdisk. If the ramdisk size changes, the setting needs to be changed accordingly so that the kernel knows what is the size of ramdisks to use. According to the "Computer Architecture A Quantitative Approach"[10] in RISC processor r0 is used in all architectures though most of the time its value is set to 0. So we will have the last command "r0 0x00000000 r1 0x00000019" to set the register r0 and r1's default value. The kernel begins by attempting to discover on what type of machine it is executing, first by checking the processor ID register, then by examining the unique architecture identifier passed in register `r1`. That is why the r1 numbers of Assabet and Intel Brutus board are different in the angelboot configuration. The value of register "r1" for Assabet is set to 19 in Hex. After that the kernel can proceed with a number of architecture-independent initializations, such as setting up the console, enabling file systems and caching, and launching the *init* thread. One operation performed by this thread is to mount the root file system, which on many StrongARM systems involves the use of a RAM disk.

## 2. Setup host environment:

The steps mentioned in this section were used to configure and set up the host environment (also known as the development platform). Tasks accomplished in this section included:

- Setting up the NFS and DHCP servers
- Setting up the cross-compilation environment for the ARM processor
- Testing the cross-compilation environment by running simple applications
- Setting up the frame buffer

The cross-compiler environment or tool chain is one of the cores of this part. The kernel, the GUI system, the applications, and all of process must use the tool chain to compile into the binary format for Assabet. We established the whole tool chain by following the Intel instruction set. They also explored several pre-compiled cross compile tool chains found in handhelds.org and eventually used the skiff cross compile tool chain. To test the cross compiler, we compiled some simple C and C++ programs and ported them onto Assabet. For future development, some debugging tools are required.

## 3. Compile Linux kernel binary and setup File System - Ext2 ramdisks:

We applied Arm Linux patch and Assabet patch to Linux kernel and compiled the kernel using both default setting and the customized setting. We enabled the support for the four major aspects, assabet touch screen, PCMCIA, NFS and the Chinese font. We used ext2 Ramdisk as the file system to test out the basic functions and demos. The support of the touch screen driver for the Assabet is not available at the time of development. We solved the problem by modifying the kernel touch screen driver for IPAQ, the Ucb1200_ts.c file in the /drivers/char directory. This was a common question posted on the various mailing lists but there were no definite solutions. Finally the touch screen driver was modified and made operational.

## 4. Microwindows and FLNX::

This section primarily describes the steps involved to setup Microwindows on the target platform. The tasks accomplished in this section included:

- Cross-compiling the JPEG libraries for the ARM platform
- Installing Microwindows on both the host and Assabet
- Setting up FLNX, a form of FLTK (Fast Light Tool Kit) that is designed to run on Microwindows, on the development platform and the Assabet

The Jpeg libraries are required in the FLNX. The Jpeg library is not maintained by Microwindows. So the interfacing between these two entities is also poorly documented. After we accomplished cross compilation of the libraries, we recompiled Microwindows and explored the features of Microwindows like Chinese fonts support using “t1demo”, handwriting recognition using “nxscribble”, and Win32 API support using “mine”. The Assabet touch screen support is the one of the remarkable contribution of this project, since the Microwindows latest version is going to use the modification we made for touch screen.

## 5. ViewML Web Browser:

This section primarily describes the steps involved to install the ViewML web browser on the target platform. It took around a month to complete this phase of the project. Tasks accomplished in this section included:

- Cross-compiling and installing the Libwww libraries on the target platform
- Cross-compiling and installing the ViewML application on the target platform

Numerous errors were detected in both the Libwww and ViewML script files. By thorough study on the configuration file and structure of the linked libraries, all errors were resolved successfully and contributed back to the communities. In order to restrict the memory usage of the embedded application, the binary file was stripped to smaller size with dynamically linked libraries and debugging symbols removed.

## 6. Palm OS Emulator:

The Palm OS Emulator was not chosen to be implemented on the Assabet due to the failure reported by other developers who also want to port POSE to embedded Linux. This helped to create a development environment, where future project groups can be benefited from the ease of creating a wide array of other applications.

After the GUI system has been set up, we can run a few applications like small games and text editor and ViewML web browser to verify that the embedded Linux hand held device prototype is working fine. Figure 2 shows the screen shot of two examples.

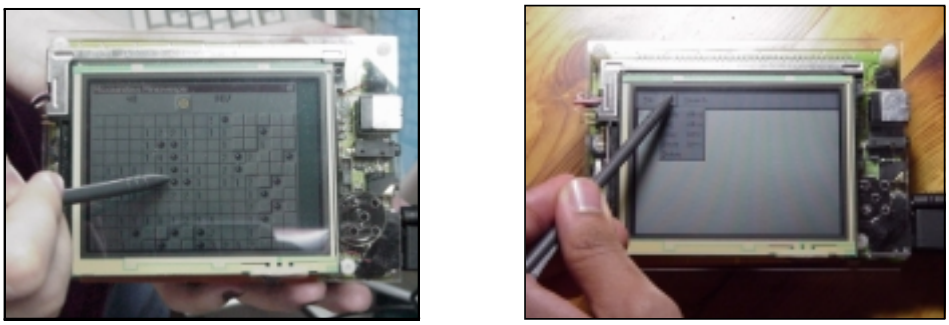

**Figure 2** Snapshots of the Minesweeper and Editor demo programs on the Assabet

# Bluetooth Development:

Through the evaluation of existing open sourced Linux Bluetooth Protocol stacks, which are Bluez, Openbt And Affix, BlueZ was finally selected due to the reliability, the friendliness, and the functionality based on BlueZ (2-pre6), Openbt (0.0.8) and Affix (0_93) using Microtune 1760 development Kits and Ericsson Multi Chip with firmware version P9A.

Table 5. The brief comparison in based on the Ericsson Starter Kit and Microtune 1760 kits

| | **BlueZ (open sourced since April 99) 1.2-2.0-pre7** | **Openbt (open sourced since May 01) 0.06-0.0.8** | **Affix (open sourced since Nov 01) 0.9** | **BlueDrekar (only HCI is open sourced)** |
|---|---|---|---|---|
| Kernel | 2.4.4 and greater | 2.0.x-2.4.x | 2.4.x | 2.2.12, 2.2.14 |
| Hardware platform | http://www.holtmann.org/linux/bluetooth/devices.html for detail | X86, Arm, MIPS, PowerPC | Same as BlueZ http://affix.sourceforge.net/hardware.shtml | X86 |
| Bluetooth Portocols | HCI, L2CAP, SDP, RFCOMM (modifying), HCI-UART, HCU-USB | HCI, L2CAP, SDP, RFCOMM, HCI-UART, HCU-USB, SCO (Incomplete). | HCI, L2CAP, RFCOMM, SDP, | HCI, L2CAP, SDP, SCO, RFCOMM, HCI-UART |

| | BlueZ (open sourced since April 99) 1.2-2.0-pre7 | Openbt (open sourced since May 01) 0.06-0.0.8 | Affix (open sourced since Nov 01) 0.9 | BlueDrekar (only HCI is open sourced) |
|---|---|---|---|---|
| API | Hardware abstraction | Standard Unix device driver | Standard Unix device driver | Custom lib API |
| License | GPL | Axis OpenBt Stack License | GPL | AlphaWorks, |
| Utility | L2test, scotest, rfcommd and hcitool | User mode, Bttest, good to understand the openbt | No | N.A. |
| Status | In progress | Minor changes | In progress | N.A. |
| SDP | The SDP is ported from the Axis. | Server, XML database. | The SDP is ported from the Axis. | Server dynamic database |
| Profile | Serial Port<br>Dialup networking<br>LAN Access | GAP, Serial Port<br>Dialup, LAN Access,<br>OBEX, synchronization | Serial Port<br>Dialup,<br>LAN Access<br>OBEX | N.A. |

*N.A., “Not Available, implies that the BlueDrekar is partially open-sourced, and we have not investigated it fully.

In order to further develop the stack with our own effort, we will look into the architecture and the philosophy of BlueZ and Openbt.

Table 6. The design comparison of BlueZ and Openbt

| BlueZ (1.2 and 2.0 pre 6) | Openbt (0.0.2 -0.0.8) |
|---|---|
| The hci layer is based on the kernel raw socket or raw HCI socket. Emulating TTY functionality would completely inefficient and less flexible. | To keep the legacy applications work with Bluetooth, similar approach as Affix. The raw socket programming lack of support from the upper layer. |
| Support as many devices on one PC. VTun by replacing the UDP with L2CAP. Utility like Hciattach is handy for UART and Blue PIN is supported with GUI. | First open source Linux Bluetooth stack. Large amount of CVS modification and verification, more versions released. BCSP protocol is always supported. |
| Porting RFCOMM engine, SDP from Openbt. | For networking protocols "char device / ioctl" interface is not as good as "sockets”. |
| Some cross compiling issues are just started to discuss. X86 and StrongARM processor. | Verified with X86, Arm, MIPS, PowerPC. |
| Implement services like SDP in the kernel space, which is result of the lack of the right interface in the user space. | Provide kernel Mode and user mode. Driver for data and control. |
| Though included in the 2.4.4 kernel and above. It is hard to use for the microcontrollers without MMU, which are best served by 2.2.x. | Certified and verified with the working product. Bluetooth Access Points running embedded Linux (2.0.36). |
| Raw socket directly interfaced to L2CAP | No interface exist for L2CAP, but ioctls are available to support most HCI commands |

To elaborate more on the similarity and difference we examined the following trace. In the Open BT trace, the minicom link was set up using the ttyBT0, after we established TTY emulation with the bt.o module inserted in the kernel, we were able to call directly to the device on the sending side.

```
[root@desktop12 experimental]# echo hallo > /dev/ttyBT0
```

Meanwhile we got the response on the other side from the ttyBT0 device.

```
[root@localhost experimental]# cat /dev/ttyBT0
hallo
```

When we monitored the kernel trace using "`tail –f /var/log/messages`", we observed that it was actually the emulation of ttyUB0 device.

```
Nov 27 14:20:00 desktop12 btd: Initiating signal handler
Nov 27 14:20:05 desktop12 btd: Opening dev /dev/ttyUB0
Nov 27 14:20:05 desktop12 btd: Opening dev /dev/ttyBTC
… (the trace is deleted here)
```

While in the BlueZ stack, the PPP or the other legacy applications could not be initialized by using hciX, ttySX or ttyUSBX but the /dev/pts/ from 0 to 5 as the device to communicate. For instance during the PPP connection through the rfcomm layer, once the "bluepin" was enabled to verify the authentication password, the following trace was displayed on the screen:

```
… (the trace is deleted here)

Mar  2 16:21:26 localhost pppd[1845]: Connect: ppp0 <--> /dev/pts/4
Mar  2 16:21:26 localhost kernel: PPP BSD Compression module registered
Mar  2 16:21:27 localhost kernel: PPP Deflate Compression module
registered
Mar  2 16:21:27 localhost pppd[1845]: local  IP address 192.168.0.1
Mar  2 16:21:27 localhost pppd[1845]: remote IP address 192.168.0.2
… (the trace is deleted here)
```

Similarly, "`cat /dev/pts/4`" works together" with "`echo hallo > /dev/pts/4`" appeared on the other side for BlueZ.

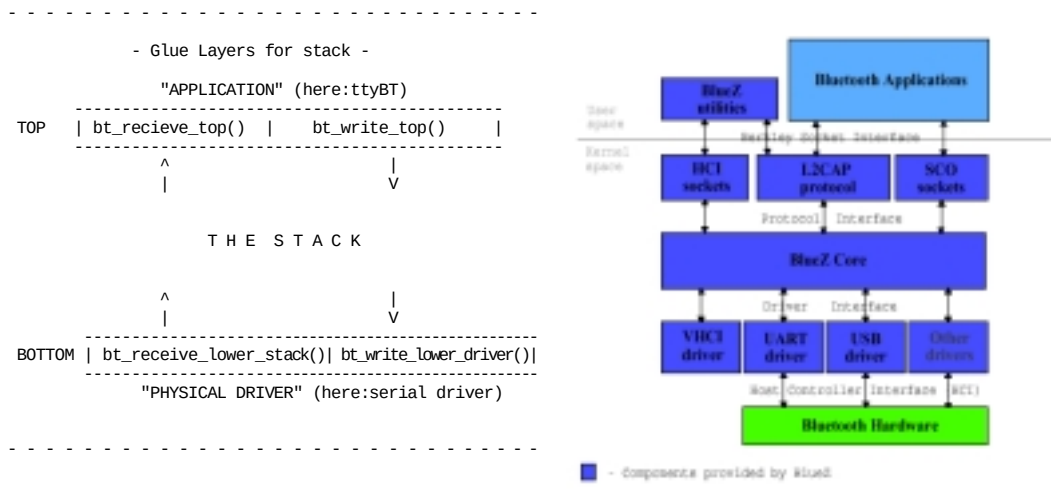


**Figure 3** The architecture over view of Openbt and BlueZ

The fundamental difference between BlueZ and Openbt is at the HCI TTY emulation. The goal for BlueZ was to design and develop general hardware independent interface for HCI core. The emulating TTY functionality like Openbt was inefficient.

Current BlueZ interface is clean and simple. Stack uses SKBs, Linux network buffers, on all layers. SKBs are different from kmem_alloc, the kernel-memory allocator, because they are SLABified, which means that there is a cache for SKB heads and allocations are optimized. In addition, SKBs provide lots of optimized functions and queues. [24]

The BlueZ used the Raw socket instead of emulating the tty function. Among the raw sockets commonly used for various UNIX utilities are: traceroute, ping, arp. Also, a lot of Internet security tools make use of raw sockets. This method avoids a lot of complexity issues comparing with TTY emulation. HCI drivers are responsible for providing complete frames that have already been classified. They feed HCI core skbs with complete frame in skb->data and skb->pkt_type set to the proper HCI packet type.

```
struct sk_buff_head        rx_q;
struct sk_buff_head        raw_q;
struct sk_buff_head        cmd_q;
int (*send)(struct sk_buff *skb);
```

Another major difference is that the BlueZ has more access utilities to interact with devices at different layers, for instance, to move UART initialization functionality from HCId to the separate utility called "hciattach". The role of UART initialization is to set up the baud rate, setting N_HCI line discipline, etc. Handling that stuff in HCId has several problems. For instance, having to modify /etc/hcid.conf and restarting HCId every time you add new UART device. The big problem is hot-plug and unplug. Interfacing to UART hot-plug and unlplug in HCId can be fairly complicated. For some people who want to implement their own versions of hciattach for their own devices. For instance if Microtune's 1760 evaluation kit requires higher baud rate, we need to customize the hciattach so that we do not have to patch HCId anymore.

Lastly Openbt does not provide an interface to L2CAP layer. While in BlueZ, socket is the only direct interface to L2CAP, which can be used in user and kernel spaces. It is not necessary to implement additional structures, lists, etc if we can keep all the information in socket itself and that is what BlueZ L2CAP layer does. Clearly it is more efficient to maintain one unified entity like socket that already has one unified interface, than inventing some other interface and then emulate sockets on top of it.

The implementations of some new interface to reduce the size of the protocol, which is perfectly suited for the socket API, are redundant. They may develop "smaller" API for L2CAP but sockets are not at all that expensive in terms of resources. They may save 1-10K at the expense of lower compatibility, efficiency, etc. Sockets always present in the kernel, for instance, in the init/main.c, they are initialized unconditionally, which make sense because even pipes are implemented using sockets.

Adding Bluez support to any existing socket based programs is very easy. BlueZ used AF_BLUETOOTH instead of AF_INET whenever a socket call is made. BlueZ uses sockaddr_l2 instead of sockaddr_in and SOCK_SEQ_PACKET instead of SOCK_STREAM and so on. Only a few new data structures and constants were introduced.

To understand the BlueZ design further, a good starting point for coding is to look at the hci_emu implementation. It is a combination of char device and hci device. Whenever user app opens /dev/hciemu, the driver registers new HCI device with Core and then relays data between hci dev <--> file descriptor. Therefore, we can write HCI emulation applications that will read/write from/to char device and do HCI emulation.

On the other hand, the Openbt claimed that one current advantage of using a tty device as a low level driver comparing with the skbs, the kernel socket buffer, is that the current usb bluetooth driver is design to easily

work with other different Linux Bluetooth stacks [24]. Nevertheless, the raw socket programming is lacking \ future support from the upper layer, though it has the advantages of flexibility.

Figure 4 shows how the application talks to the serial device through the Bluetooth emulation layer. Openbt works based on the tty device instead of through the HCI core. The following drawing shows the design algorithm with an example of UART and PPP connection, where ldisc means “line discipline”.

The ttySx is a standard serial port tty. ttyBTx has a separate major number and has been registered to use bt driver as tty driver. ttySx is opened initially from a bluetooth control-application, which sets the bt line discipline and keeps the tty open.When ttyBT is opened from pppd, it is set to use ppp discipline which in turn uses the driver registered in its tty i.e., the bt driver. At the top of the stack it acts like a serial driver for ppp. At the bottom of the stack, it acts like any line discipline used from the standard serial tty.

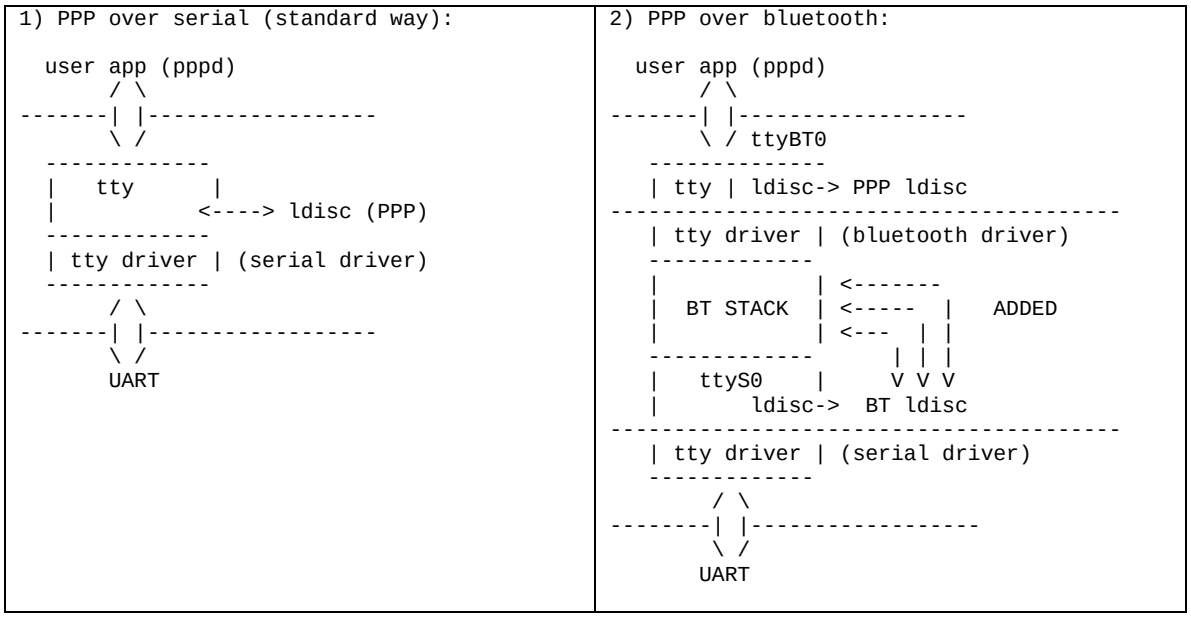


**Figure 4** The Openbt tty emulation

The SDP server was tested and the results indicated that the connection through the Ericsson device P9A type slow devices with third party SDP client is not very reliable, which might be due to the XML parsing problem. While bt.o module links to the sdpparser.c, the sdpparser.c sends the discover command to sdpserver.c, which is used to process the XML database. The XML database and the parser might be over killed for limited number of information in SDP server. It generates reliability issue for bluetooth devices with low processing power.

Another important contribution to the BlueZ stack development was the finding of the USB zero packet length problem, and the fixing by changing the “ urb->transfer_flags = USB_QUEUE_BULK” to “urb->transfer_flags = USB_QUEUE_BULK | USB_ZERO_PACKET” in the hci_usb.c file [25].

# Conclusion:

The Bluetooth enabled embedded Linux prototype has been set up successfully. As a platform for the Software Defined Radio development, we have proved the robustness, availability and flexibility of wireless embedded Linux. The future plan will be to focus on the GNU Radio and armature radio to create a more versatile Software Defined Radio platform. The support for the real time processing needs to be studied for time critical tasks like TDMA.

## Acknowledgement:

First and foremost, the first author would like to express his great gratitude to his friends Supriyo Chatterjea and Bryan Battern for their contributions on FLNX, Jpeg Library, ViewML, Microtune 1760 bluetooth kit and USB. He would like to thank A/P Chan Kap Luk, and Ms. Liou Mei Fan for all the advices. Last but not least, a special thanks goes to his parents and his girlfriend Xiao Rui for their support and encouragement.

# References:


1. The ARM Linux Project,.http://www.arm.linux.org.uk/.
2. *Intel StrongARM SA-1110 Microprocessor Development Board User's Guide,* http://developer.intel.com/design/strong/.
3. *Specification for Generic Touch Screen Driver*, http://handhelds.org/projects/ts_spec.html.
4. *Microwindows documentation* http://home.twcny.rr.com/embedded/microwin/
5. *The Embedded Linux Distributions Quick Reference Guide,* http://www.linuxdevices.com/articles/AT2760742655.html, February 5, 2001
6. *MontaVista's Hard Hat Linux Wins Penguin Playoff Awards for 'Best Embedded Solution'*, http://www.mvista.com/news/playoff.html
7. *The Embedded Linux GUI/Windowing Quick Reference Guide,* http://www.linuxdevices.com/articles/AT9202043619.html , March 16, 2001
8. *Qt/Embedded*,http://www.linuxdevices.com/products/PD9968824320.html
9. The Wearable Group at Carnegie Mellon, *ARM Linux on Assabet*,http://www.cs.cmu.edu/~wearable/software/assabet.html , 20 March, 2001
10. David A. Patterson, John L. Hennessy, *Computer architecture : a quantitative approach*( 2nd edition) San Francisco : Morgan Kaufmann Publishers, 1996, Page C-3
11. Tackett, Jack, 2000, *Special edition using Linux*, Indianapolis, Ind. Pp467-pp468
12. Matthew, Neil,Stones, Richard,2000, *Professional Linux programming*, Birmingham [England] : Wrox Press,pp800-805
13. Application Note, http://developer.intel.com/design/strong/applnots/sa1100lx/sa1100lx.htm, May 1999
14. MontaVista Software, 2000, *Hard Hat Linux Cross Development Kit - Support for Embedded Linux*, MontaVista Software, ch. 5
15. Linux Frame Buffer Driver Developer, http://www.linux-fbdev.org/
16. The FLTK home page, http://www.fltk.org/about.html
17. Microwindows architecture, http://www.microwindows.org/microwindows_architecture.html
18. Independent JPEG group, http://www.ijg.org/
19. ViewML, http://www.viewml.org/
20. Palm OS Emulator, http://www.palmos.com/dev/tech/tools/emulator/
21. Matthias Kalle Dalheimer, 1999, Programming with Qt, Cambridge [England]; Sebastopol, CA: O'Reilly, page 4.
22. *"Linux BlueZ Howto Bluetooth protocol stack for Linux"* Jan Beutel, Maksim Krasnyanskiy http://bluez.sourceforge.net/howto/index.html 24 August 2001
23. "*Welcome to Affix web site - Bluetooth Protocol Stack for Linux*" Dmitri Kassatkine <dmitri.kassatkine@nokia.com> http://www-nrc.nokia.com/affix/: 2001/11/23 09:31:02
24. "[bluetooth-dev] Stack design and some anoucements" (A letter from the BlueZ original author available on the Openbt mailing list), http://mhonarc.axis.se/bluetooth-dev/msg01881.html to msg01884.html , Thu, 03 May 2001
25. [Bluez-devel] Bluetooth USB driver problems recap. Maksim Krasnyanskiy http://www.geocrawler.com/lists/3/SourceForge/11993/0/8094915/03/13/2002 15:25:38