Bluetooth Enabled Embedded Linux

Linux Kongress 2002

David Xiaoyong Yang
Centre for High Performance Embedded Systems
Nanyang Technological University
Singapore, linuxprotocol@hotmail.com

Topics

- Why Bluetooth enabled embedded Linux?
- Embedded Linux for PDA prototype
- Bluetooth
- Conclusion and Future Plan

Embedded Systems

- Embedded Systems:
  - Every thing non-PC.
  - Typically no human intervention. Special purpose small foot print software in ROM. Often real-time response. Low power, cost sensitive.
- Empowered by the wireless RF technology and various ASIC and microprocessors.
- To be connected and unified.

Wireless - Bluetooth and SDR

- Bluetooth:
  - Cable replacement to unify the short range communication
  - RF, BB , … … to software
  - Open standard and low cost $5
  - Synergy between open standard and open sourced
- Ultimate goal: Software Defined Radio
  - Get the software **close to the antenna.** Configurable, and plug and play..
  - Unify the various communication standards, basis for 4G.
  - Standard and source code side by side: no ambiguity, unify, fast and flexible.
  - Problem: Generally assuming presence of sufficient band width, reusable platform, and processing power. Cost.
  - Solution: Bluetooth and open source especially Linux is a good starting point to implement from bottom up to a complex SDR .

Problem and Solution

- Too many standards;
- Too many type to embedded systems;
- Too much cost;
- Too long development cycle, yet too short time in market;
- Too complicated to start from scratch

- Open and reusable.
- Support various microprocessor. DIY
- Low cost
- Smaller footprint than windows
- Better network performance (socket* referring to IBM).
- Real time performance is enhanced.

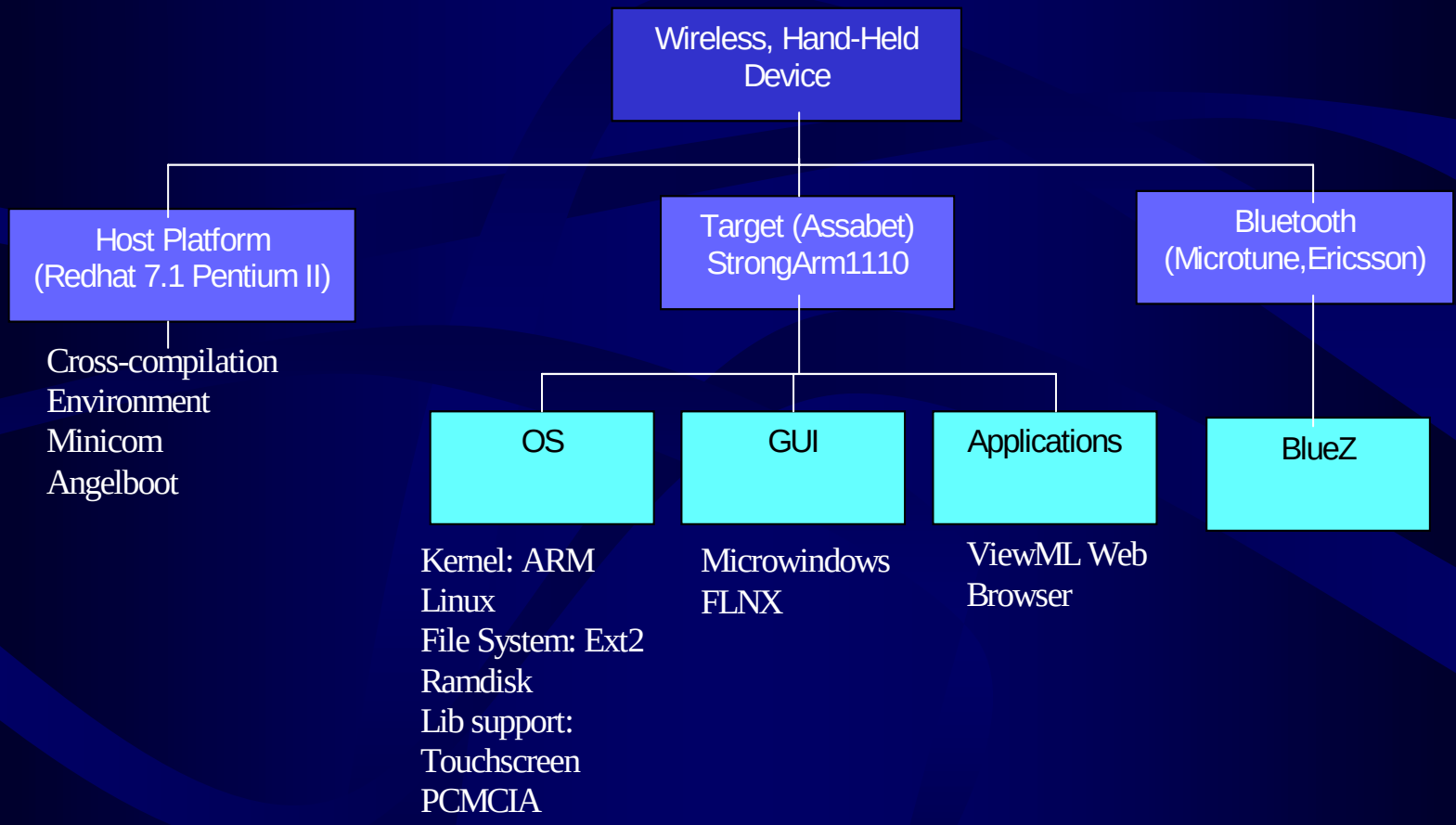
XP Embedded /Embedded Linux

- Microsoft says
  - Integrated: tools
  - Comprehensive:performance & reliability
  - Unmated: technology portfolio
  - Interoperable:.Net
  - Proven: business model
  - Global:support
- Linux folks says
  - GNU X-Tools, IDE, VisualAge(Java), Qt, Sleepycat(Database)
  - Reliable and small. UcLinux core 800k, yet XP Embedded core 4.8 MB
  - IEEE; IBM, HP, etc.
  - Java, XML.
  - Successful stories
  - Linux distributors and developers

Why Linux need to be embedded and wireless?

- Linux kernel maintenance problem:
  - UNIX Co-creator, Ken Thompson : Linux won't success in long run. "Common coupling (dependency) grows exponentially while the LOC grows linearly."
  - Restructuring kernel with the bare minimum of common coupling.
- An efficient solution:
  - Embedding Linux with constrain applied
  - Footprint reduced, problem delayed
- The wireless world need a unified open platform. Yet its operating platform is not dominated by MS or BrandX.

Bluetooth enabled embedded Linux



Part I:
The Key points of the Embedded

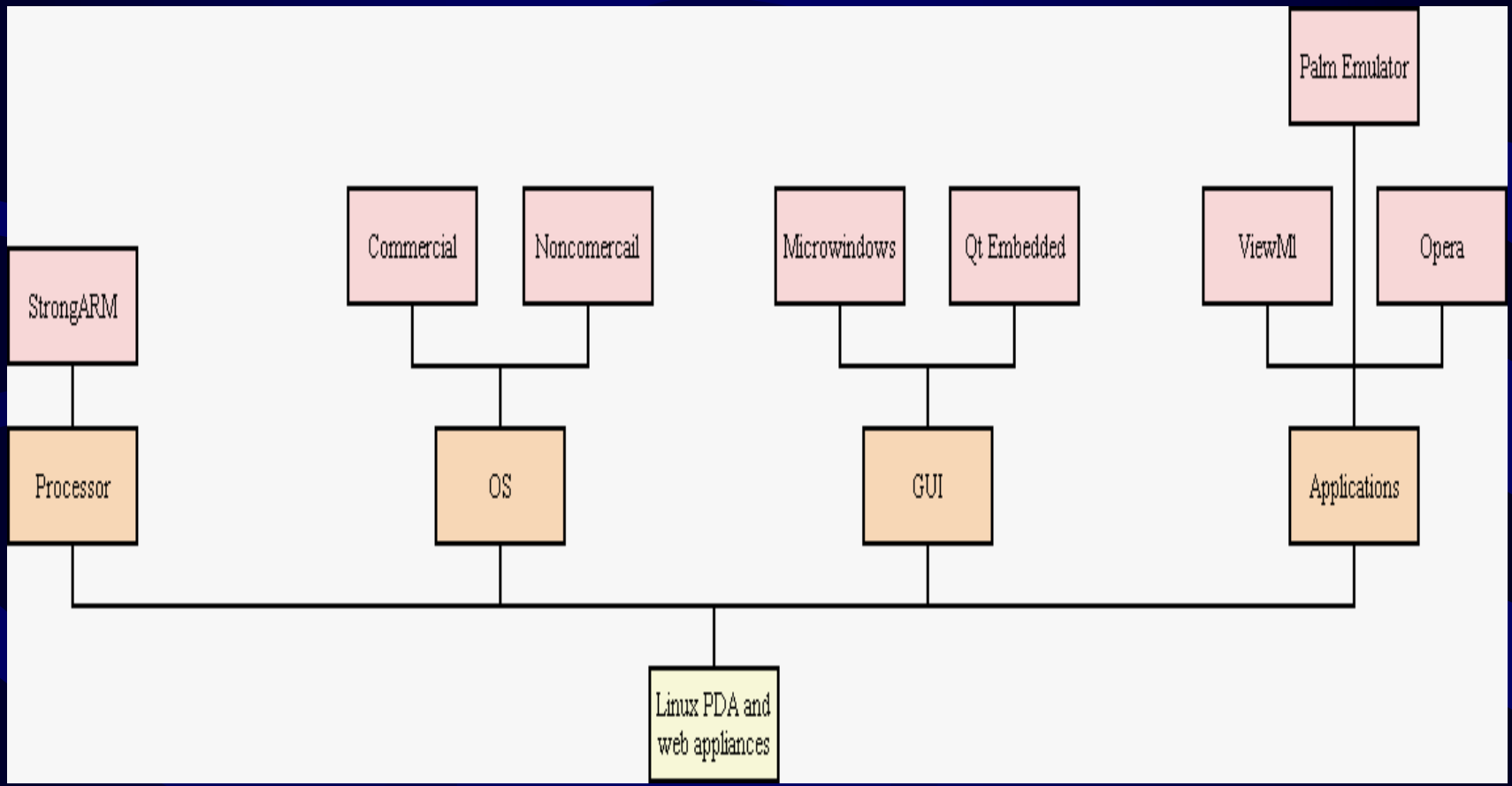
- The approach.
- The special in embedded. Boot loader, RAM Disk, and input.
- Debugging for embedded.
- Applications.

Approaches

- Using the PDA environment. E.g. Qt-embedded , Qt-palmtop, pocket Linux.
- Using the commercial distributions. E.g.UcLinux, Hardhat, Lineo, Ecos.
- Developing and stripping the non-commercial embedded Linux project. E.g. ARMLinux, Uclinux.
- Growing the Linux from scratch.

The third method is suitable to gain maximum control without inventing the wheel.

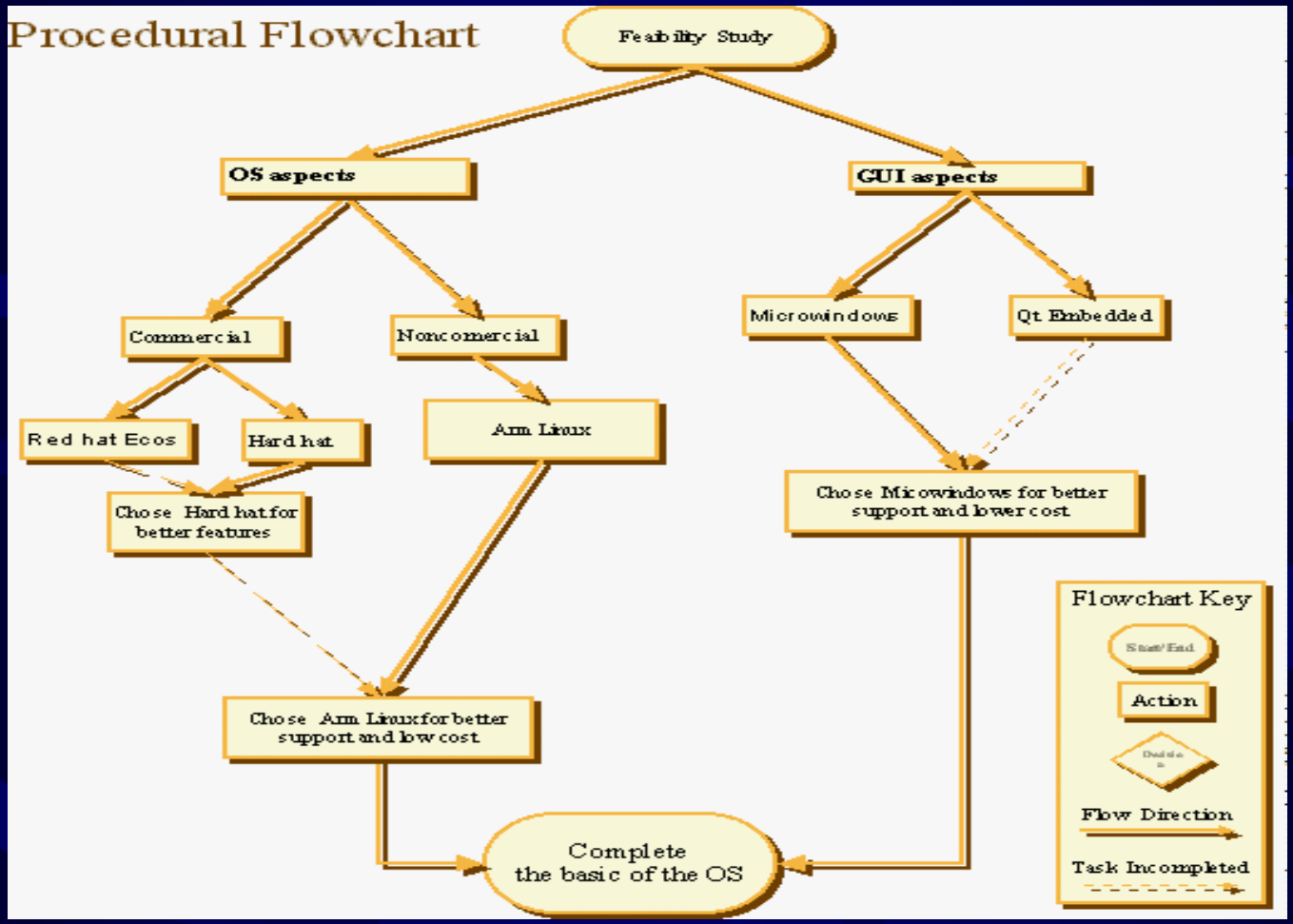
The Feasibility Study



A Hard Start

	Intel StrongARM SA-1110	NEC MIPS VR4122	Hitachi (SH3 Series) SH7709A
Speed (MHz) * (Palm need about 20-30MHz, 10 times more is desired.)	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 * (not restricted to 2.2.* kernel only)	32-Entry Maps 4Kbyte, 8Kbyte, 1Mbyte	32-Entry, 1-256Kbyte	128-Entry Maps 4Kbyte, 8Kbyte
Power Management Unit	Normal Mode <240mW Idle Mode <75mW Sleep Mode <50uA	Full speed, Stand by, Suspend, Hibernate	Normal mode, Sleep mode Standby mode
Operating Voltage – Core	1.75V	1.8V	1.9V

ARMLinux & Microwindows



Why ARM Linux and Microwindows?

- The commercial distributions need certain amount for support. The developer mailing list is not available. The latest source is not available.
- The GUI options X, Microwindows and Qt.
- X, though it is not as big as people believe. E.g. IBM used X for its 2M Linux Watch. The extra effort required. E.g. Tiny-X.
- Microwindows, $0 cost, and the best combination with FLNX, ViewML. Qt embedded, not fully open sourced, the freedom of choice.

The Applications and Libraries

- POSE: Palm OS emulator need FLNX.
- ViewML web browser need FLNX.
- Both need the FLNX, a embedded version of FLTK, which is a library for graphical display, which supports Windows and Linux platform (NanoX).

The Implementation Outline

Applications, web browser and Palm emulator*	Debug ViewML and libraries for JEPG, libwww
Support for JPEG, HTTP and wireless	Host Microwindows and FLNX
FLNX, Microwindows	Cross-compilation for Arm
Debug the LCD controller	Tool chain Ext2 RRAM Disk
ARM Linux Kernel with frame buffer, network, and UNICODE	Minicom, NFS and DHCP
Load boot code-Angel in Flash via JTAG	Angelboot for Assabet

Some Examples for Steps

- Angelboot for the Assabet.
- Build the tool chain
- Ext2fs RAM Disk
- LCD controller
- Frame buffer
- Debugging Libraries and Applications

Using Angelboot to boot up

- Boot loader: jump, find kernel, load kernel, test and initialisation.
- The host angelboot.
  - Synchronize the serial to be with minicom on host.
  - “image zImage” and “entry 0xc0008000” “base 0xc0008000” is used to load the compressed kernel (640kB boot time limitation) or ramdisk and specify the address in RAM. The values must be consistent with file arch/arm/mach-sal100/arch.c in the ARM Linux kernel.
  - “r0 0x00000000 r1 0x00000019” In RISC, commonly r0 is set to 0. It is used as ID register for the kernel to discover what type of machine, then by examining the unique architecture identifier passed in register `r1`. Typically for Assabet r1 is set to 19 in Hex.
  - Then the kernel can proceed with a number of architecture-independent initialisations, such as setting up the console, enabling RAM Disk file systems ……

Notes on building the Tool chain

- It consists four parts namely the Bin Utilities (Binutils), the EGCS C* Compiler, the GLIBC Library (C library), the EGCS C++* Compiler.
- Precompiled tool chain, Hardhat compiler, LART cross compiler and skiff.
- Choose the target prefix, arm-linux to work with others. Set the path “PATH:/skiff/local/bin/$PATH”.
- Reduce footprint? Strip, dynamic lib, or use non standard lib like uclib. E.g. ViewML web browser is stripped from 6M to 1M.
- Mixing up the cross compilers will lead to segmentation fault.
- To debug, firstly check the program is portable, suitable for cross compile, e.g. JPEG library. Then check the cross compiler switches in the make or configure file.

Ext2fs RAM Disk

- Ext2fs chosen instead of JFFS, Cramfs, Ramfs due to the maturity level,ease of starting and popularity.
- RAM disks tend to be small; a compressed disk image might be 1.5-2.5MB in size. It may need network (PCMCIA) setup scripts, *busybox*, tinylogin, *strace*, *e2fsck*, *mke2fs*,etc.
- In a RAMDisk and loopback supported Linux system. The steps are:
  - %dd if=/dev/zero of=/dev/ram bs=1k count=8192

  8 MB for this example. “of=/dev/ram”, where /dev/ram1 (for a ramdisk). Zero out the area (if=/dev/zero) so that maximal compression is achieved for the unused blocks of the image.
  - %mke2fs -vm0 /dev/ram 8192

  Then they need to make a file system on it , formatting the drive
  - %mount -o loop -t ext2 ramdisk /mnt/new-ramdisk
  - %mount -o loop -t ext2 ramdisk_old /mnt/old-ramdisk

Ext2fs RAMDisk(continued)

The switch “loop” enable the loop back device to access an ext2 image. The '-t ext2 ' reads the super block, the ramdisk, to determine the file system.

- copy all stuff (eg: /etc/* /dev/* ...) from the old-ramdisk directory to the new-ramdisk directory
- %umount /mnt/new-ramdisk

Remember to unmount it before extracting it, or it can be out-of sync, or if “reboot”, the X windows might be corrupted.

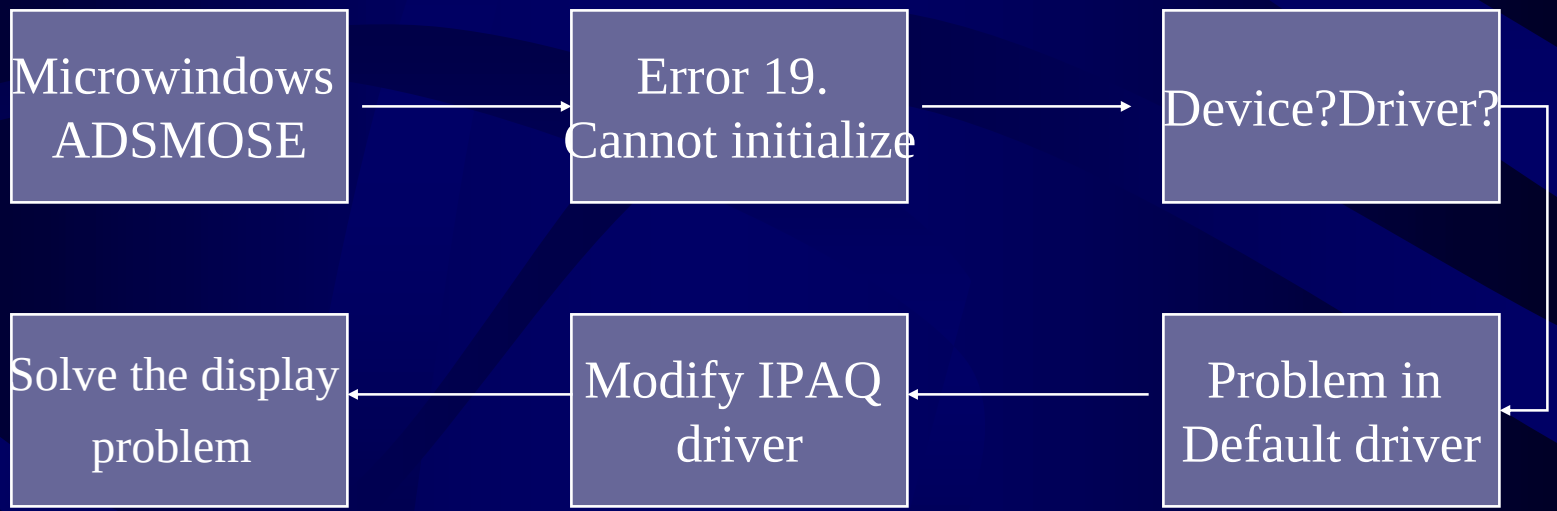
- %cat ramdisk | gzip -v9 > ramdisk.gz

The level of compression will be about 50% of the space used by the files. Unused space on the RAM disk will compress to almost nothing. An 8MB ramdisk can be compressed to 2MB.

- Any modifications to a RAM disk file system will not survive a reset or loss of power. Fixed size and locking the memory.
- After some work (copied, moved, deleted files), the ramdisk will have dirty blocks. To get a smaller compressed image, copy the contents to a clean ramdisk.

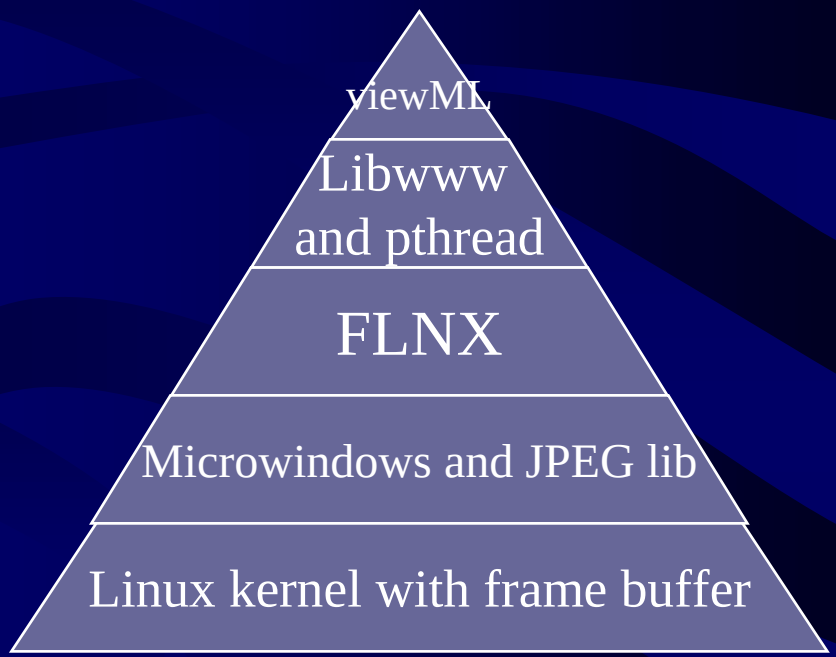
A simple fix for LCD controller

- Sharp LCD connected to Philips UCD1300 ADC.



The Frame Buffer, Libs and Apps

- On the host the X- windows and frame buffer competing the resource. Disable X auto start.
- One the target, compensate the frame buffer and bandwidth
- Debugging for cross compile:
  - JPEG: the “rc” switch is missing.
  - libwww: http engine. Need to change the compilation sequence to avoid x86 only.
  - ViewML: fix the link the libwww xmlparse.h, the JPEG lib, pthread. Fix the linking problem for JPEG.
  - Dynamically linked libs, slow but small foot point.



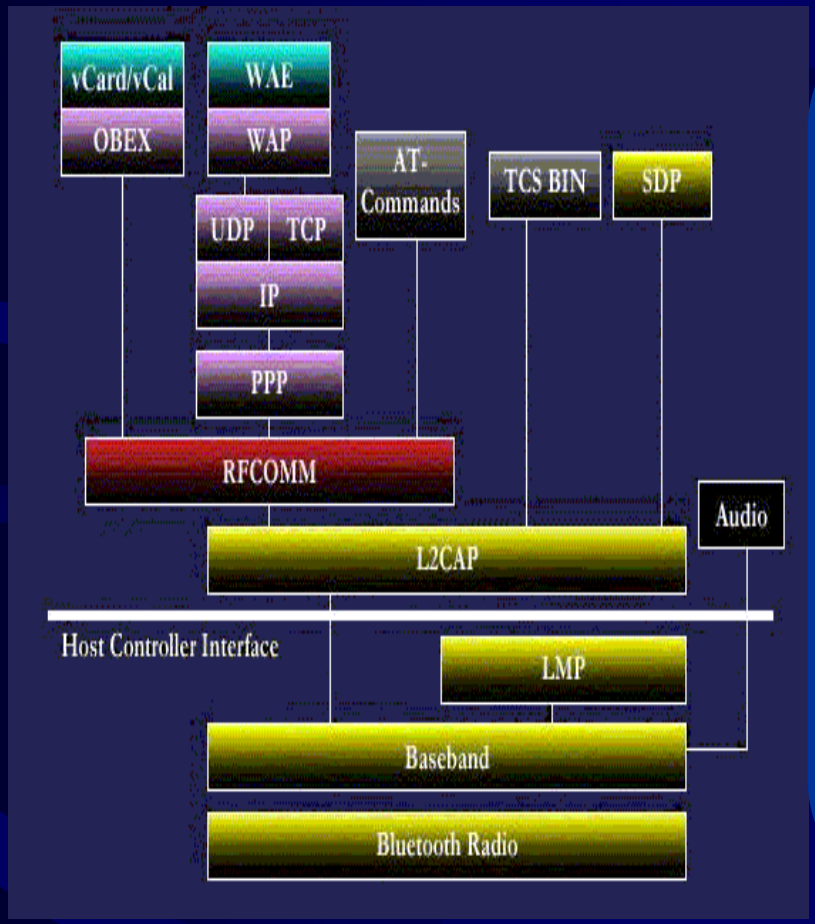
Known Issues

- Ext2 RAMdisk consume large memory. JFFS and XIP.
- POSE need substantial modification to port.
- Smaller footprint. 64K for OS and apps?

Part II
The Key of the Bluetooth

- Understand BlueZ and Openbt.
- The Bluetooth and USB
- Design improvement for Openbt and BlueZ

Bluetooth and its Profiles



Generic Access Profiles

- Service Discovery

TCS Binary based profiles

- Cordless Telephony
- Intercom

Serial Port Profiles

- Dial-up Networking
- Fax Profile
- LAN Access

Generic Object Exchange Profiles

- File Transfer
- Object Push
- Synchronization

Bluetooth Stacks Comparison

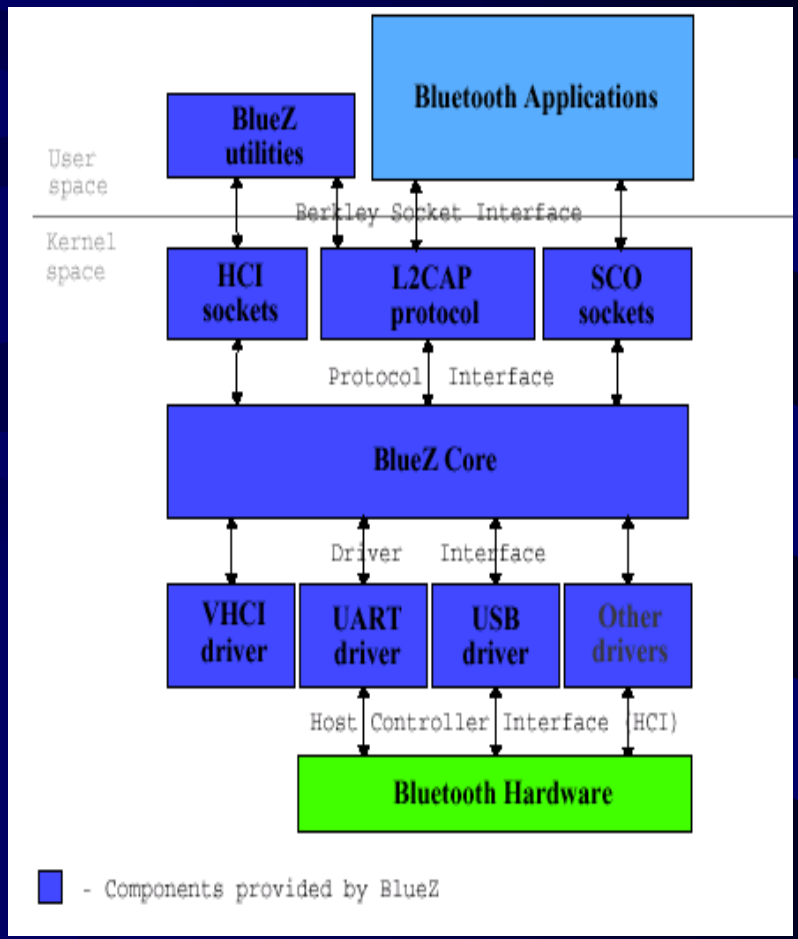
	BlueZ (open sourced since April 99) 1.2-2.0-pre7	**Openbt (open sourced since May 01)**	**Affix (open sourced since Nov 01) 0.9**	**BlueDrekar (only HCI is open sourced)**
Kernel	2.4.4 and greater	2.0.x-2.4.x	2.4.x	2.2.12, 2.2.14
Host/ target platform	Host: X86 ARM Target: in reference. CRS just added in.	Host: X86, Arm, MIPS, PowerPC Target: support most of chip	Host: X86 Target;Same as BlueZ	Host: X86 Target: N.A.
Bluetooth Protocols	HCI, L2CAP, SDP, RFCOMM (modifying), HCI-UART, HCU-USB	HCI, L2CAP, SDP, RFCOMM, HCI-UART, HCU-USB, SCO	HCI, L2CAP, RFCOMM, SDP,	HCI, L2CAP, SDP, SCO, RFCOMM, HCI-
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 Dialup networking LAN Access	GAP, Serial Port Dialup, LAN Access, OBEX, synchronization	Serial Port Dialup, LAN Access OBEX	N.A.

The Further Comparison of Bluez and Openbt

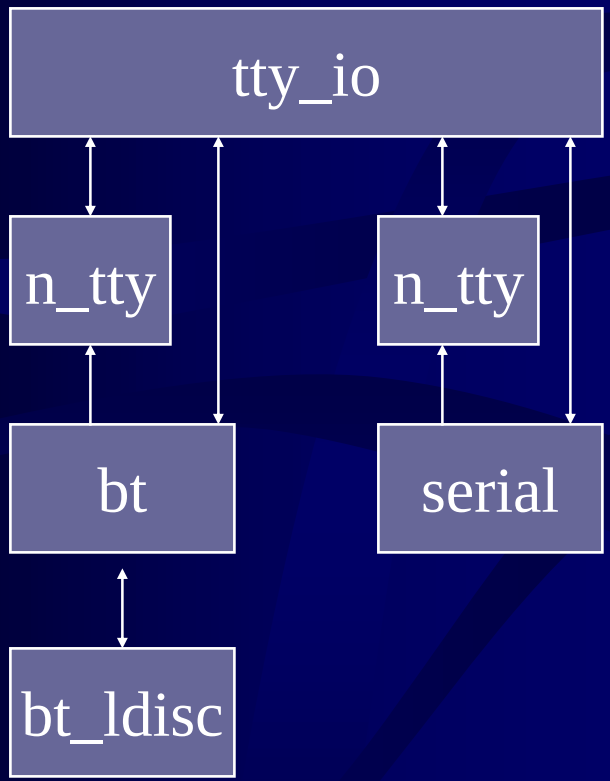
BlueZ (1.2 and 2.0 pre 6)	Openbt (0.0.2 -0.0.8)
The hci layer is based on the kernel raw socket or raw HCI socket. Emulating TTY functionality would completely inefficient and less flexible.	To keep the legacy applications work with Bluetooth, similar approach as Affix. The raw socket lack of support 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.
Openbt implements SDP in the kernel space 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	ioctls are available to support HCI commands

The Architecture Overview of Openbt and BlueZ

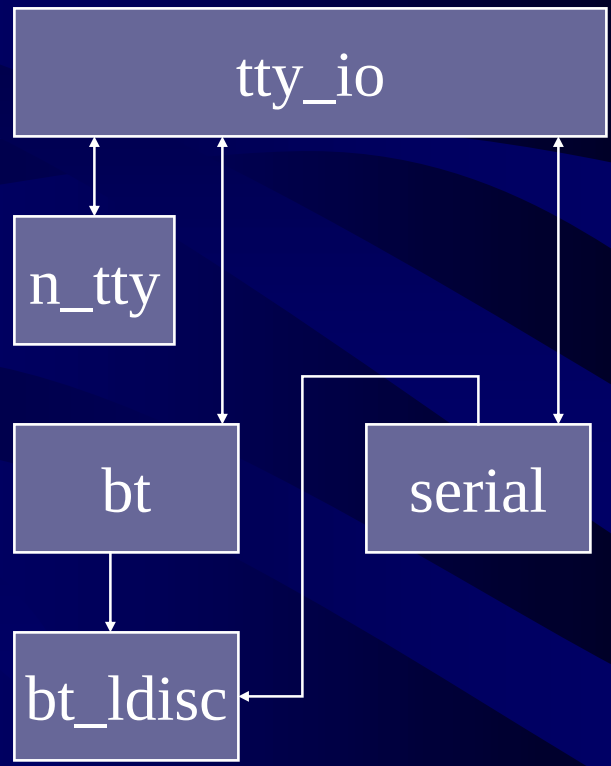
```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

            - Glue Layers for stack -

                "APPLICATION" (here:ttyBT)
          ------------------------------------------
TOP       | bt_in_top()  |    bt_out_top()     |
          ------------------------------------------
                ^                      |
                |                      V


                     THE STACK


                ^                      |
                |                      V
          ------------------------------------------------
BOTTOM    | bt_out_botoom()| bt_out_bottom()|
          ------------------------------------------------
                "PHYSICAL DRIVER" (here: serial driver)


- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```



TTY Driver in Openbt



- Default or emulation



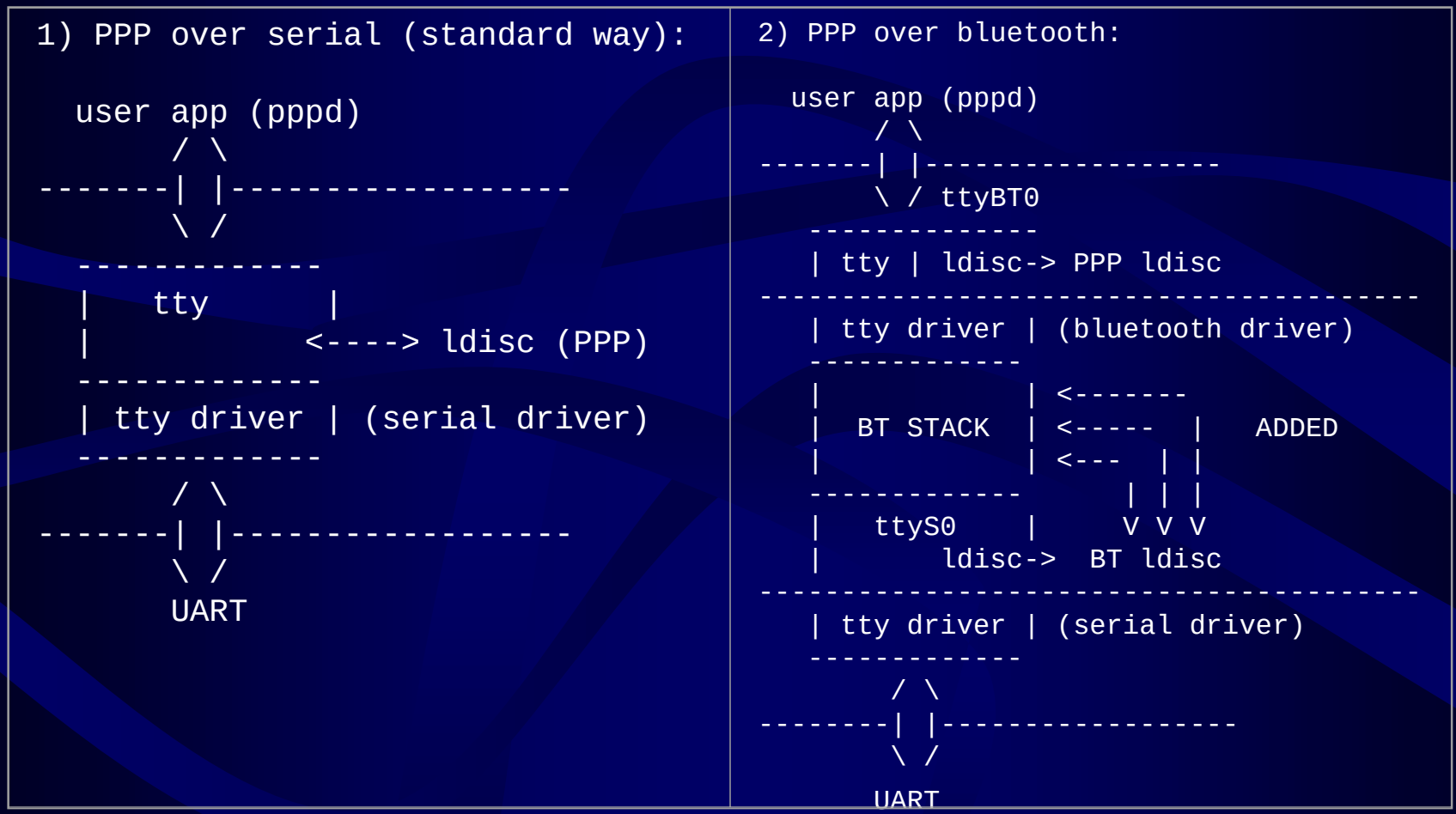
- Stacked Bluetooth Used

Further Explanations for Openbt

- Openbt Stack the driver by telling the serial port to switch from N-tty line discipline to the BT line discipline.

  “int bt_idlisc=N_BT; ioctl(fd,TIOCSETD, &bt_ldisc);”
- N_BT uniquely identifies the BT line discipline among all other line disciplines registered in the kernel. The identifier tell tty to used BT as its upper layer interface.
- The TIOCSETD ioctl replaces the serial ports’ current line discipline with the one specified. It also causes the Bluetooth Linux discipline's open routine to be called, passing in the serial port’s tty to Bluetooth.
- Driver for data, ttyBT0 and for control ttyBTC0. User mode and kernel mode.
- Avoid the filtering for special char, and messing the data while switching between BT and serial, E.g. try cat on /dev/ttyS0 with binar, use Raw tty “ioctl(fd, TCGETS, &t); cfmakraw(&t);”

The PPP Using Openbt



Traces for Explanation

Openbt:

- `[root@desktop12 experimental]# echo hallo > /dev/ttyBT0`
  `[root@localhost experimental]# cat /dev/ttyBT0`
  `hallo`
- `[root@desktop12 bluetooth]# btd --reset -u /dev/ttyUB0 --speed 115200 -r client`
  … …
- The kernel trace using "`tail –f /var/log/messages`". It emulates the ttyUB0.

```
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
```

BlueZ:

- While "`echo hallo > /dev/pts/4`", "`cat /dev/pts/4 hallo`" displays
- To start ppp, "`rfcommd –s –n na`" on the server and "`rfcommd –n –na [server_bd_address]`" on the client. It uses the /dev/pts/ to communicate.

```
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 pppd[1845]: remote IP address 192.168.0.2
```

Further explanation for BlueZ

- Current BlueZ interface is clean and simple. Stack uses SKBs, the kernel socket buffer , on all layers.
- SKB is different from kmem_alloc, the kernel-memory allocator, because it is 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.
- "struct sk_buff_head rx_q;
  struct sk_buff_head raw_q;
  struct sk_buff_head cmd_q;
  struct sk_buff *sent_cmd; … …
  int (*send)(struct sk_buff *skb);" in hci_core.h
- While in BlueZ, socket is the only direct interface to L2CAP, which can be used in user and kernel spaces. 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.

Bluetooth and USB

- Due to speed limitation of UART 115kbps << max BT radio 723.2kbps. The H4, USB, is commonly used.
- The confusion among USB drivers. In 2.4.* there exist two drivers for UHCI host controllers: this one from "UHCI support" ,“usb-uhci.o” and the one from "UHCI alternate (JE) support”, “uhci.o”.
- They are very similar. But the best one to work with BlueZ stack is the ,“usb-uhci.o” not the default “uhci.o”
- bluetooth.o is claimed to supports 256 different USB Bluetooth device. However it does not work together with BlueZ, during the hot plug, it has a higher priority than usb-uhci.o due to naming sequence. The best way is to disable this module for BlueZ.
- The BlueZ stack development encountered of the USB zero packet length problem. “ urb->transfer_flags = USB_QUEUE_BULK” →“urb->transfer_flags = USB_QUEUE_BULK | USB_ZERO_PACKET” in the hci_usb.c file.

Known Issues:

- Openbt, ioctl calls specific to the BT driver have global effects. E.g. if it needs reinitialize the stack, it could interrupt another application's data transfers. Interface L2cap. The emulation is complicated
- Openbt, lack of control over the RFCOMM link. The winner is whoever issues ioctl call last.
- BlueZ might have the tty emulation for legacy application. RFCOMM and SDP move into kernel space?
- Dynamic database used in Openbt and BlueZ in stead of hard coded XML database.

Conclusion and Future Works

- Linux is really a handy platform for wireless embedded system for tight budget and schedule project.
- Go hardware? Open HDL, open system C, handle C. It can be used for the run time reconfiguration of Linux and the hardware.
- Go real time? Time critical, e.g. TDMA. Run- time re-configurable.
  - Two Approaches to Real-Time Linux
    - Modify Linux to include a Real-Time scheduler. Eg. MontaVista
    - Put Regular Linux on top of a Real-Time operating system (RTOS). Eg. RTLinux , Lineo. They achieved ms jitter, which is suitable for majority of the RT apps. For high performance RT app, it need to have ns jitter.
- Go GNU Radio?
  - It's a free software defined radio
  - A platform for experimenting with digital communications and for signal processing on commodity hardware

Q & A
A Reason for Smile