«Gumstix How-To»

1. Gumstix embedded platforms

The Gumstix [1] are a very small cards [2] that can run GNU/Linux embedded systems, there are some different models of cards and each one has it's own connection and expansion possibilities that supply connection such as USB-net, Bluetooth, Ethernet and/or WiFi, others supply audio I/O, etc.

2. Basic setup

2.1. Log into the Gumstix

The first way to log into the Gumstix cards is using a cable which is a RS-232 null modem comm plugged into one of the available connectors.

The usual program to connect to the Gumstix is Kermit, it has to be configured with this instructions to get it work:

```
set line /dev/ttyS0
set speed 115200
set carrier-watch off
set handshake none
set flow-control none
set reliable
fast
set prefixing all
set file type bin
set rec pack 4096
set send pack 4096
```

This configuration can be saved into a file called .kermrc into your home directory which will load this configuration every time Kermit is started.

The null modem configuration can be seen using the command ***show comm***.

```
C-Kermit>show comm

Communications Parameters:
 Line: /dev/ttyS0, speed: 115200, mode: local, modem: generic
 Parity: none, stop-bits: (default) (8N1)
 Duplex: full, flow: none, handshake: none
 Carrier-watch: off, close-on-disconnect: off
 Lockfile: /var/lock/LCK..ttyS0
 Terminal bytesize: 8, escape character: 28 (^\)

 Carrier Detect      (CD):  Off
 Dataset Ready       (DSR): Off
 Clear To Send       (CTS): Off
 Ring Indicator      (RI):  Off
 Data Terminal Ready (DTR): On
 Request To Send     (RTS): On

Type SHOW DIAL to see DIAL-related items.
Type SHOW MODEM to see modem-related items.
```

With the sentence ***connect*** the connection between the Gumstix and the computer will be established.

```
C-Kermit>connect
Connecting to /dev/ttyS0, speed 115200
 Escape character: Ctrl-\ (ASCII 28, FS): enabled
Type the escape character followed by C to get back,
or followed by ? to see other options.
----------------------------------------------------



Welcome to the Gumstix Linux Distribution!

gumstix login:
```

Then we can do login using the standard *login* (root) and *password* (gumstix).

NOTE: To log in successfully the Bluetooth module must be plugged into the basix card.

3. Cross-development tools

Cross-development tools are that ones that allow you to compile on one machine with it's own system but the output will only run into the target machine.
In Gumstix there are many language compilers such as CC, C++, GCC, G++, GCJ, etc, but it can also run a JAVA Virtual Machine on the Gumstix as you will see at *section 3.3*.

NOTE: The programs compiled for the Gumstix will not run under your GNU/Linux distribution because they have been compiled with a fork version of the compilers.

3.1. C

To compile a C program for the Gumstix you will have to compile it with the

modified GCC compiler special for the Gumstix, this compiler, like other cross tools is located on the directory:

/gumstix/build_arm_nofpu/staging_dir/bin

and the name of the compiler is:

arm-linux-gcc

You can compile any program written in C using the compiler as follows:

```
[root@localhost programs]# /~/gumstix/build_arm_nofpu/staging_dir/bin/arm-
linux-gcc hello.c -o hello
[root@localhost programs]# ls -al
total 36
drwxr-xr-x   2 root root 4096 Apr  3 15:41 .
drwxr-x---  37 root root 4096 Apr  3 15:40 ..
-rwxr-xr-x   1 root root 5372 Apr  3 15:41 hello
-rw-r--r--   1 root root   90 Apr  3 15:40 hello.c
```

where **~** is the top directory where the buildroot sources are.
Instead of writing all times the full path for the compiler you can export the path:

```
export PATH=$PATH:/~/gumstix/build_arm_nofpu/staging_dir/bin
```

again **~** is the top directory, usually could be

```
export PATH=$PATH:/~/gumstix/build_arm_nofpu/staging_dir/bin
```

or:

```
export PATH=$PATH:/usr/src/gumstix/build_arm_nofpu/staging_dir/bin
```

The flag ***-o*** means that the output file will be named as the name following the option.

If you want to run this program on the Gumstix you have to upload it, see *section 5.1*.

See *Annex 10.1* for some examples of C programs.

3.2. C++

Compiling and uploading a C++ program is like compiling a C program but instead of using ***arm-linux-gcc*** the compiler is called ***arm-linux-g++***.

```
[root@localhost programs]# arm-linux-g++ hello.cpp -o hello_c++
[root@localhost programs]# ls
hello  hello_c++  hello.c  hello.cpp
```

This is an example of how to compile a C++ program once the path is already exported.

See *Annex 10.2* for some examples of C programs.

3.3. JAVA

[Write something]

JAVA
Native GCJ compiler
Virtual machines
JamVM
installation: jdk, jikes, classpath, jamvm

4. Setting up interfaces

4.1. USB-net

For this connection we will use a USB-miniUSB cable that usually comes with the Gumstix package.
It is very simply to set up a USB-net connection using the command ***ifconfig***, it will treat this connection as a normal net interface so the only thing to do is configure it follows:

```
# ifconfig usb0 10.0.0.1 netmask 255.255.255.0 up
```

With this sentence the USB-net usb0 interface will be set up on the Gumstix, then another USB-net interface must be set up on the computer from which the connection will be started.

```
[root@localhost ~]# ifconfig usb0 10.0.0.2 netmask 255.255.255.0 up
```

The ***ifconfig*** command will show on both, the Gumstix and the localhost, the configuration of all interfaces.

On the Gumstix:

```
# ifconfig
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:13 errors:0 dropped:0 overruns:0 frame:0
          TX packets:13 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:4329 (4.2 KiB)  TX bytes:4329 (4.2 KiB)

usb0      Link encap:Ethernet  HWaddr 0A:00:FD:42:F3:21
          inet addr:10.0.0.1  Bcast:10.255.255.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:6 errors:0 dropped:0 overruns:0 frame:0
          TX packets:216 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:348 (348.0 B)  TX bytes:127440 (124.4 KiB)
```

On the localhost:

```
[root@localhost ~]# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:50:DA:43:DA:22
          inet addr:194.182.170.122  Bcast:194.182.170.255  Mask:255.255.255.0
          inet6 addr: fe80::250:daff:fe43:da22/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:733584 errors:0 dropped:0 overruns:0 frame:0
          TX packets:470003 errors:0 dropped:0 overruns:0 carrier:0
          collisions:7638 txqueuelen:1000
          RX bytes:639207746 (609.5 MiB)  TX bytes:135037402 (128.7 MiB)
          Interrupt:11 Base address:0xdc80

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:2379 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2379 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:3808422 (3.6 MiB)  TX bytes:3808422 (3.6 MiB)

usb0      Link encap:Ethernet  HWaddr 46:59:DE:1D:71:EC
          inet addr:10.0.0.2  Bcast:10.0.0.255  Mask:255.255.255.0
          inet6 addr: fe80::4459:deff:fe1d:71ec/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:380 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:218880 (213.7 KiB)  TX bytes:432 (432.0 b)
```

NOTE: The localhost Linux will not recognize usb0 as an interface until the Gumstix has booted.

```
[root@localhost ~]# ifconfig usb0 10.0.0.2 netmask 255.255.255.0 up
SIOCSIFADDR: No such device
usb0: unknown interface: No such device
SIOCSIFNETMASK: No such device
SIOCGIFADDR: No such device
SIOCSIFBROADCAST: No such device
usb0: unknown interface: No such device
```

To test the configuration you can ping on both sides:

```
# ping 10.0.0.2
PING 10.0.0.2 (10.0.0.2): 56 data bytes
64 bytes from 10.0.0.2: icmp_seq=0 ttl=64 time=8.2 ms
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=5.0 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=4.6 ms

--- 10.0.0.2 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 4.6/5.9/8.2 ms
```

```
[root@localhost ~]# ping 10.0.0.1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
64 bytes from 10.0.0.1: icmp_seq=0 ttl=64 time=2.03 ms
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=1.32 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=1.21 ms

--- 10.0.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 1.217/1.524/2.036/0.365 ms, pipe 2
```

With this simple two commands a USB-net connection will be created between the

computer and the card.

4.2. Ethernet

For this connection we will use a RJ-45 straight through cable. Setting up a Ethernet connection is as simply as the USB-net one. Using ***ifconfig*** the connection can be established as follows:

```
# ifconfig eth0 192.168.0.201 netmask 255.255.255.0 up
```

```
[root@localhost ~]# ifconfig eth1 192.168.0.200 netmask 255.255.255.0 up
```

If you have more than one Ethernet card you will have to decide where do you want to establish the connection so maybe you will have to change eth0 to eth1, eth2, etc. Using ***ifconfig*** with no arguments you can see which interfaces are up on your machine.

```
[root@localhost ~]# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:50:DA:43:DA:22
          inet addr:194.182.170.122  Bcast:194.182.170.255  Mask:255.255.255.0
          inet6 addr: fe80::250:daff:fe43:da22/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:743624 errors:0 dropped:0 overruns:0 frame:0
          TX packets:476039 errors:0 dropped:0 overruns:0 carrier:0
          collisions:7665 txqueuelen:1000
          RX bytes:642375402 (612.6 MiB)  TX bytes:135926324 (129.6 MiB)
          Interrupt:11 Base address:0xdc80

eth1      Link encap:Ethernet  HWaddr 00:06:5B:64:4B:8B
          inet addr:192.168.0.200  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::206:5bff:fe64:4b8b/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:152 errors:0 dropped:0 overruns:0 frame:0
          TX packets:58 errors:0 dropped:0 overruns:0 carrier:1
          collisions:0 txqueuelen:1000
          RX bytes:31226 (30.4 KiB)  TX bytes:5699 (5.5 KiB)
          Interrupt:11 Base address:0xdc00

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:2383 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2383 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:3808758 (3.6 MiB)  TX bytes:3808758 (3.6 MiB)
```

NOTE: to establish successful connection between Gumstix and other machines all IPs have to be from the same LAN range, remember that the three possible ranges for LAN are:

- 192.168.x.x
- 172.16.x.x
- 10.x.x.x

Now you can ***ping*** both interfaces to verify that the connection has been set up.

```
# ping 192.168.0.200
PING 192.168.0.200 (192.168.0.200): 56 data bytes
64 bytes from 192.168.0.200: icmp_seq=0 ttl=64 time=2.3 ms
64 bytes from 192.168.0.200: icmp_seq=1 ttl=64 time=0.5 ms
64 bytes from 192.168.0.200: icmp_seq=2 ttl=64 time=0.5 ms

--- 192.168.0.200 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 0.5/1.1/2.3 ms
```

```
[root@localhost ~]# ping 192.168.0.201
PING 192.168.0.201 (192.168.0.201) 56(84) bytes of data.
64 bytes from 192.168.0.201: icmp_seq=0 ttl=64 time=1.44 ms
64 bytes from 192.168.0.201: icmp_seq=1 ttl=64 time=0.481 ms
64 bytes from 192.168.0.201: icmp_seq=2 ttl=64 time=0.472 ms

--- 192.168.0.201 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2000ms
rtt min/avg/max/mdev = 0.472/0.799/1.445/0.457 ms, pipe 2
```

4.3. Bluetooth

Bluetooth connections have to be handled in a different way as the net connections.
To set up a Bluetooth connection you will have to use this commands/services:

hciconfig	(Host Controller Interface) for configure Bluetooth devices
hcitool	for configure Bluetooth devices and send some special commands
sdptool	for control and interrogate SDP servers
l2ping	L2CAP (Logical Link Control and Adaptation Protocol, Bluetooth protocol stack) echo request and receive answer
sdpd	SDP Daemon
pand	Bluetooth PAN (Personal Area Network) Daemon

The kind of Bluetooth protocol connection is master/slave, which means that one of the devices must be configured as master, with it's special features, and the other must as the slave, with it's own special features, so the connection will be point-to-point and it will only involve two devices, unless one of them will act as a proxy/gateway forwarding packages over a net connection
Then we have two simple connection possibilities:

- Gumstix – Gumstix
- Gumstix – Localhost

And one multiple connection:

- Gumstix – Gumstix - Localhost

a) Gumstix – Gumstix Bluetooth connection

First you have to check that your system recognizes the Bluetooth device, ***hciconfig*** will show the Bluetooth devices attached to the Gumstix. If ***hciconfig*** don't return any device, it fails or the returned Bluetooth address is 00:00:00:00:00:00 it means that there is something wrong, could be the Bluetooth card or the buildroot version.
Bluetooth device on Gumstix:

```
# hciconfig
hci0:   Type: UART
        BD Address: 00:80:37:27:1C:8A ACL MTU: 339:7 SCO MTU: 120:6
        DOWN
        RX bytes:7461 acl:0 sco:0 events:1062 errors:0
        TX bytes:9749 acl:0 sco:0 commands:1062 errors:0
```

Bluetooth device on Gumstix (*MAC* error):

```
hci0:   Type: UART
        BD Address: 00:00:00:00:00:00 ACL MTU: 0:0 SCO MTU: 0:0
        DOWN
        RX bytes:0 acl:0 sco:0 events:0 errors:0
        TX bytes:4 acl:0 sco:0 commands:1 errors:0
```

If everything goes right we will have on both Gumstix a Bluetooth device, we must set up it:

```
# hciconfig hci0 up
# hciconfig
hci0:   Type: UART
        BD Address: 00:80:37:27:1C:8A ACL MTU: 339:7 SCO MTU: 120:6
        UP RUNNING PSCAN ISCAN INQUIRY
        RX bytes:7589 acl:0 sco:0 events:1077 errors:0
        TX bytes:10076 acl:0 sco:0 commands:1077 errors:0
```

Then we can scan all Bluetooth devices in our range with the tool ***hcitool***:

```
# hcitool scan
Scanning ...
        00:0F:DE:25:D1:B5       K700i
        00:80:37:20:D2:51       Gumstix (0)
        00:10:C6:94:FA:24       D610-009
```

The ***l2ping*** command is like a ***ping*** but for Bluetooth devices, it works with Bluetooth *MAC* address instead of IP addresses as the usual ***ping*** does.
We can try ***l2ping*** both devices from each other.

```
# l2ping 00:80:37:20:D2:51
Ping: 00:80:37:20:D2:51 from 00:80:37:27:1C:8A (data size 44) ...
44 bytes from 00:80:37:20:D2:51 id 0 time 186.91ms
44 bytes from 00:80:37:20:D2:51 id 1 time 92.92ms
44 bytes from 00:80:37:20:D2:51 id 2 time 76.73ms
44 bytes from 00:80:37:20:D2:51 id 3 time 96.81ms
4 sent, 4 received, 0% loss
```

```
# l2ping 00:80:37:27:1C:8A
Ping: 00:80:37:27:1C:8A from 00:80:37:20:D2:51 (data size 20) ...
20 bytes from 00:80:37:27:1C:8A id 0 time 230.25ms
20 bytes from 00:80:37:27:1C:8A id 1 time 116.97ms
20 bytes from 00:80:37:27:1C:8A id 2 time 152.04ms
20 bytes from 00:80:37:27:1C:8A id 3 time 82.09ms
4 sent, 4 received, 0% loss
```

More information about the Bluetooth devices can be obtained using the command ***sdptool browse***.

```
# sdptool browse
Inquiring ...
Browsing 00:80:37:27:1C:8A ...
Service Name: Serial Port
Service Description: COM Port
Service RecHandle: 0x10000
Service Class ID List:
  "Serial Port" (0x1101)
Protocol Descriptor List:
  "L2CAP" (0x0100)
  "RFCOMM" (0x0003)
    Channel: 1
Language Base Attr List:
  code_ISO639: 0x656e
  encoding:    0x6a
  base_offset: 0x100
Profile Descriptor List:
  "Serial Port" (0x1101)
    Version: 0x0100
```

Now start the *SDP daemon*:

```
# sdpd
```

On the Gumstix that is going to be the Master side:

```
# pand --listen --role NAP --master --autozap
```

The Gumstix slave side has to be configured to listen for connections using ***pand*** with the option ***-s*** or ***--listen***.

```
# pand -s
```

On the Gumstix that is going to be the Master side you'll have to set the hardware address of the Slave Gumstix using ***pand***:

```
# pand --connect 00:80:37:20:D2:51 --service NAP --autozap
```

At this time you must have a bnep (Bluetooth network) set up, to check it use ***pand -l*** on both sides:

```
# pand -l
bnep0 00:80:37:20:D2:51 NAP
```

```
# pand -l
bnep0 00:80:37:27:1C:8A PANU
```

To set up the TCP/IP over Bluetooth you just need to configure both devices with ***ifconfig*** as always:

```
# ifconfig bnep0 10.1.5.1 netmask 255.255.255.0 up
```

```
# ifconfig bnep0 10.1.5.2 netmask 255.255.255.0 up
```

And check that ***ifconfig*** returns the bnep0 network on both sides:

```
# ifconfig
bnep0     Link encap:Ethernet  HWaddr 00:80:37:27:1C:8A
          inet addr:10.1.5.1  Bcast:10.255.255.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:66 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:148 (148.0 B)  TX bytes:35813 (34.9 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:12 errors:0 dropped:0 overruns:0 frame:0
          TX packets:12 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:4117 (4.0 KiB)  TX bytes:4117 (4.0 KiB)

usb0      Link encap:Ethernet  HWaddr D6:02:33:43:C5:E1
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

```
# ifconfig
bnep0     Link encap:Ethernet  HWaddr 00:80:37:20:D2:51
          inet addr:10.1.5.2  Bcast:10.255.255.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:60 errors:0 dropped:0 overruns:0 frame:0
          TX packets:5 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:35248 (34.4 KiB)  TX bytes:128 (128.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

usb0      Link encap:Ethernet  HWaddr 06:00:FC:42:C2:61
          inet addr:10.0.0.3  Bcast:0.0.0.0  Mask:255.255.255.0
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

Now you can test your connection using ***ping***:

```
# ping 10.1.5.2
PING 10.1.5.2 (10.1.5.2): 56 data bytes
64 bytes from 10.1.5.2: icmp_seq=0 ttl=64 time=205.9 ms
64 bytes from 10.1.5.2: icmp_seq=1 ttl=64 time=96.9 ms
64 bytes from 10.1.5.2: icmp_seq=2 ttl=64 time=109.6 ms

--- 10.1.5.2 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 96.9/137.4/205.9 ms
```

```
# ping 10.1.5.1
PING 10.1.5.1 (10.1.5.1): 56 data bytes
64 bytes from 10.1.5.1: icmp_seq=0 ttl=64 time=143.2 ms
64 bytes from 10.1.5.1: icmp_seq=1 ttl=64 time=197.1 ms
64 bytes from 10.1.5.1: icmp_seq=2 ttl=64 time=159.7 ms

--- 10.1.5.1 ping statistics ---
5 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 143.2/166.6/197.1 ms
```

Sometimes we get some error messages:

```
l2cap_recv_acldata: Unexpected start frame (len 589)
l2cap_recv_acldata: Unexpected start frame (len 589)
l2cap_recv_acldata: Unexpected start frame (len 589)
l2cap_recv_acldata: Unexpected start frame (len 589)
l2cap_recv_acldata: Unexpected start frame (len 589)
```

At this moment we don't know what does it mean, sometimes the Bluetooth connection is slower or the interfaces loose a few packets.

b) Gumstix – Localhost Bluetooth connection

If the Bluetooth configuration on the Gumstix has been tested before that step will be skipped, but to get the Bluetooth working on the localhost we have to load Bluetooth modules.
To configure a USB-Bluetooth under Fedora Core 4 we will follow the next steps [3, 4]:

```
[root@localhost ~]# lsusb
Bus 002 Device 014: ID 0525:a4a2 Netchip Technology, Inc. Linux-USB
Ethernet/RNDIS Gadget
Bus 002 Device 001: ID 0000:0000
Bus 001 Device 002: ID 050d:0084 Belkin Components
Bus 001 Device 001: ID 0000:0000
[root@localhost ~]# rpm -q -a | grep bluez
bluez-libs-devel-2.15-1
bluez-libs-2.15-1
bluez-utils-2.15-7
bluez-pin-0.24-2
bluez-utils-cups-2.15-7
bluez-hcidump-1.18-1
[root@localhost ~]# /etc/init.d/bluetooth start
```

The command ***lsusb*** will show the usb devices connected to your localhost.
On Fedora Core ***rpm -q -a | grep -i bluez*** will install and configure all Bluetooth packages we need to deal with the usb-Bluetooth device.
The third sentence will start Bluetooth services on the localhost.

We have to check whether if we have a device called ***rfcomm0*** on /dev/ or not, if not we need to create it:

```
[root@localhost dev]# mknod /dev/rfcomm0 c 216 0
[root@localhost dev]# ls -al |grep rfcomm
crw-r--r--   1 root root   216,   0 Mar 22 14:28 rfcomm0
```

And then we just follow the same steps as setting up a Bluetooth connection over two Gumstix, first you have to see the hci interfaces with ***hciconfig***:

```
[root@localhost ~]# hciconfig hci0 down
[root@localhost ~]# hciconfig
hci0:   Type: USB
        BD Address: 00:0A:3A:5C:00:FD ACL MTU: 192:8 SCO MTU: 64:8
        DOWN
        RX bytes:115 acl:0 sco:0 events:13 errors:0
        TX bytes:62 acl:0 sco:0 commands:10 errors:0

[root@localhost ~]# hciconfig hci0 up
[root@localhost ~]# hciconfig
hci0:   Type: USB
        BD Address: 00:0A:3A:5C:00:FD ACL MTU: 192:8 SCO MTU: 64:8
        UP RUNNING PSCAN ISCAN
        RX bytes:115 acl:0 sco:0 events:13 errors:0
        TX bytes:62 acl:0 sco:0 commands:10 errors:0
```

On the Gumstix is the same as what you have done before:

```
# hciconfig hci0 down
# hciconfig
hci0:   Type: UART
        BD Address: 00:80:37:27:1C:8A ACL MTU: 339:7 SCO MTU: 120:6
        DOWN
        RX bytes:3904 acl:95 sco:0 events:243 errors:0
        TX bytes:2743 acl:95 sco:0 commands:77 errors:0

# hciconfig hci0 up
# hciconfig
hci0:   Type: UART
        BD Address: 00:80:37:27:1C:8A ACL MTU: 339:7 SCO MTU: 120:6
        UP RUNNING PSCAN ISCAN
        RX bytes:4025 acl:95 sco:0 events:257 errors:0
        TX bytes:3061 acl:95 sco:0 commands:91 errors:0
```

Now we scan on both sides for Bluetooth interfaces:

```
[root@localhost ~]# hcitool scan
Scanning ...
        00:80:37:27:1C:8A       Gumstix (0)
        00:10:C6:94:FA:24       D610-009
```

```
# hcitool scan
Scanning ...
        00:0F:DE:25:D1:B5       K700i
        00:0A:3A:5C:00:FD       TEST-NZ7TNUU6LK
        00:10:C6:94:FA:24       D610-009
```

The localhost will be the Master side:

```
[root@localhost ~]#  pand --listen --role NAP --master --autozap
```

Set the Gumstix PAN daemon to listen for connections:

```
# pand -s
```

On the localhost again you'll have to set the connection using the hardware address of the slave Gumstix using ***pand***:

```
[root@localhost ~]# pand --connect 00:80:37:27:1C:8A --service NAP --autozap
```

Now you can verify that the Bluetooth connection exists using ***pand -l*** on both sides:

```
[root@localhost ~]# pand -l
bnep0 00:80:37:27:1C:8A NAP
```

```
# pand -l
bnep0 00:0A:3A:5C:00:FD PANU
```

On the localhost you can configure the Bluetooth interface using ***ifconfig*** as follows, and then check that the interface is correctly set up:

```
[root@localhost ~]# ifconfig bnep0 10.1.5.1 netmask 255.255.255.0 up
[root@localhost ~]# ifconfig
bnep0     Link encap:Ethernet  HWaddr 00:0A:3A:5C:00:FD
          inet addr:10.1.5.1  Bcast:10.255.255.255  Mask:255.255.255.0
          inet6 addr: fe80::20a:3aff:fe5c:fd/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:16 errors:0 dropped:0 overruns:0 frame:0
          TX packets:7 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:9516 (9.2 KiB)  TX bytes:304 (304.0 b)

eth0      Link encap:Ethernet  HWaddr 00:50:DA:3E:B5:72
          inet addr:194.182.170.123  Bcast:194.182.170.255  Mask:255.255.255.0
          inet6 addr: fe80::250:daff:fe3e:b572/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1289 errors:0 dropped:0 overruns:0 frame:0
          TX packets:986 errors:0 dropped:0 overruns:0 carrier:0
          collisions:11 txqueuelen:1000
          RX bytes:477865 (466.6 KiB)  TX bytes:142037 (138.7 KiB)
          Interrupt:11 Base address:0xdc80

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:1967 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1967 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:3623498 (3.4 MiB)  TX bytes:3623498 (3.4 MiB)
```

And now you must configure the Bluetooth on the Gumstix:

```
# ifconfig bnep0 10.1.5.2 netmask 255.255.255.0 up
# ifconfig
bnep0     Link encap:Ethernet  HWaddr 00:80:37:27:1C:8A
          inet addr:10.1.5.2  Bcast:10.255.255.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:17 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:324 (324.0 B)  TX bytes:7148 (6.9 KiB)

eth0      Link encap:Ethernet  HWaddr D6:02:33:43:C5:D1
          inet addr:192.168.0.103  Bcast:192.168.0.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:70 errors:0 dropped:0 overruns:0 frame:0
          TX packets:29 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:10882 (10.6 KiB)  TX bytes:11036 (10.7 KiB)
          Interrupt:59 Base address:0x2300 DMA chan:8

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:4 errors:0 dropped:0 overruns:0 frame:0
          TX packets:4 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:1084 (1.0 KiB)  TX bytes:1084 (1.0 KiB)

usb0      Link encap:Ethernet  HWaddr D6:02:33:43:C5:E1
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2 errors:2 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

The last step is check that Bluetooth connection over TCP/IP works, ***ping*** on both machines:

```
[root@localhost ~]# ping 10.1.5.2
PING 10.1.5.2 (10.1.5.2) 56(84) bytes of data.
64 bytes from 10.1.5.2: icmp_seq=0 ttl=64 time=101 ms
64 bytes from 10.1.5.2: icmp_seq=1 ttl=64 time=90.2 ms
64 bytes from 10.1.5.2: icmp_seq=2 ttl=64 time=105 ms

--- 10.1.5.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2001ms
rtt min/avg/max/mdev = 90.252/98.841/105.225/6.313 ms, pipe 2
```

```
# ping 10.1.5.1
PING 10.1.5.1 (10.1.5.1): 56 data bytes
64 bytes from 10.1.5.1: icmp_seq=0 ttl=64 time=89.4 ms
64 bytes from 10.1.5.1: icmp_seq=1 ttl=64 time=97.3 ms
64 bytes from 10.1.5.1: icmp_seq=2 ttl=64 time=90.0 ms

--- 10.1.5.1 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 89.4/92.2/97.3 ms
```

If you want the Gumstix to be the Master and the localhost to be the Slave it's trivial to change the roles.

c) Gumstix – Gumstix – Localhost Bluetooth connection

This is to set a connection over three Bluetooth devices, one Gumstix and the localhost will be Slaves and the other Gumstix will be the Master for both connections but changing the role it's simple as nearly all of above has seen before.

First check that the Master Gumstix 'see' both Slave Bluetooth devices:

```
# hcitool scan
Scanning ...
        00:80:37:27:1D:07       Gumstix (0)
        00:0A:3A:5C:00:FD       TEST-NZ7TNUU6LK
```

Second set up, as before, the Master Bluetooth connection:

```
# pand --listen --role NAP --master --autozap
```

Then you will have to set both Slaves to listen for connections:

```
# pand -s
```

```
[root@localhost ~]# pand -s
```

Then the Master Gumstix is ready to connect to both devices, the first Bluetooth connection to be established will be the one from the localhost:

```
# pand --connect 00:0A:3A:5C:00:FD --service NAP --autozap
```

And now the one from the Slave Gumstix:

```
# pand --connect 00:80:37:27:1D:07 --service NAP --autozap
```

Check the list of PAN connections using ***pand -l***:

```
bnep1 00:0A:3A:5C:00:FD PANU
bnep0 00:80:37:27:1D:07 PANU
```

Now you have to create the TCP/IP connections using ***ifconfig*** as always. This time you have to pay attention because you have two Bluetooth connections and each one has to be configured into a different subnet segment, the connection with the localhost will be created on 10.1.5.X and the other one will be created on 10.2.5.X:

```
# ifconfig bnep1 10.1.5.1 netmask 255.255.255.0 up
# ifconfig bnep0 10.2.5.1 netmask 255.255.255.0 up
```

This has to be the output of ***ifconfig*** on the Master Gumstix:

```
# ifconfig
bnep0     Link encap:Ethernet  HWaddr 00:80:37:27:1C:8A
          inet addr:10.2.5.1  Bcast:10.255.255.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:71 errors:0 dropped:0 overruns:0 frame:0
          TX packets:76 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:41627 (40.6 KiB)  TX bytes:41663 (40.6 KiB)

bnep1     Link encap:Ethernet  HWaddr 00:80:37:27:1C:8A
          inet addr:10.1.5.1  Bcast:10.255.255.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:4 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:16 (16.0 B)  TX bytes:100 (100.0 B)

eth0      Link encap:Ethernet  HWaddr D6:02:33:43:C5:D1
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1698 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:1001820 (978.3 KiB)
          Interrupt:59 Base address:0x2300 DMA chan:8

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:13 errors:0 dropped:0 overruns:0 frame:0
          TX packets:13 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:4329 (4.2 KiB)  TX bytes:4329 (4.2 KiB)

usb0      Link encap:Ethernet  HWaddr D6:02:33:43:C5:E1
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6 errors:4 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

This Gumstix will be the Gateway between the other Gumstix and the localhost, but that will be explained later.

Bring up both Slave Bluetooth interfaces:

```
[root@localhost ~]# ifconfig bnep0 10.1.5.2 netmask 255.255.255.0 up
```

```
# ifconfig bnep0 10.2.5.2 netmask 255.255.255.0 up
```

At this time the TCP/IP over Bluetooth must be already created and you can check it using ***ping***.
This is a ***ping*** example from the Master to the Slaves:

```
# ping 10.1.5.2
PING 10.1.5.2 (10.1.5.2): 56 data bytes
64 bytes from 10.1.5.2: icmp_seq=0 ttl=64 time=196.6 ms
64 bytes from 10.1.5.2: icmp_seq=1 ttl=64 time=105.7 ms
64 bytes from 10.1.5.2: icmp_seq=2 ttl=64 time=97.1 ms

--- 10.1.5.2 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 97.1/133.1/196.6 ms
# ping 10.2.5.2
PING 10.2.5.2 (10.2.5.2): 56 data bytes
64 bytes from 10.2.5.2: icmp_seq=0 ttl=64 time=319.2 ms
64 bytes from 10.2.5.2: icmp_seq=1 ttl=64 time=168.8 ms
64 bytes from 10.2.5.2: icmp_seq=2 ttl=64 time=164.0 ms

--- 10.2.5.2 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 164.0/217.3/319.2 ms
```

And now from the Slaves to the Master:

```
[root@localhost ~]# ping 10.1.5.1
PING 10.1.5.1 (10.1.5.1) 56(84) bytes of data.
64 bytes from 10.1.5.1: icmp_seq=0 ttl=64 time=123 ms
64 bytes from 10.1.5.1: icmp_seq=1 ttl=64 time=101 ms
64 bytes from 10.1.5.1: icmp_seq=2 ttl=64 time=107 ms

--- 10.1.5.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 101.433/110.660/123.128/9.152 ms, pipe 2

[root@localhost ~]# ping 10.2.5.1
PING 10.2.5.1 (10.2.5.1) 56(84) bytes of data.
From 80.160.86.81 icmp_seq=0 Destination Host Unreachable
From 80.160.86.81 icmp_seq=1 Destination Host Unreachable
From 80.160.86.81 icmp_seq=2 Destination Host Unreachable

--- 10.2.5.1 ping statistics ---
3 packets transmitted, 0 received, +3 errors, 100% packet loss,
time 1999ms, pipe 2
```

```
# ping 10.2.5.1
PING 10.2.5.1 (10.2.5.1): 56 data bytes
64 bytes from 10.2.5.1: icmp_seq=0 ttl=64 time=183.0 ms
64 bytes from 10.2.5.1: icmp_seq=1 ttl=64 time=163.6 ms
64 bytes from 10.2.5.1: icmp_seq=2 ttl=64 time=160.9 ms

--- 10.2.5.1 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 160.9/169.1/183.0 ms

# ping 10.1.5.1
PING 10.1.5.1 (10.1.5.1): 56 data bytes
ping: sendto: Network is unreachable
```

Master Gumstix has two IP as it has two Bluetooth connection but the Slaves can't see the other Bluetooth interface and also they can't see each other, to fix that you can add a default gateway or add each host using the Master Gumstix as gateway using the tool: ***route***, but before the Master Gumstix must be set up as a gateway:

```
# vi /proc/sys/net/ipv4/ip_forward
```

Write a '***1***', save and exit:

```
# cat /proc/sys/net/ipv4/ip_forward
1
```

And then on each device to see the routing table use the command ***route -n***:

```
[root@localhost ~]# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.0.0     0.0.0.0         255.255.255.0   U     0      0        0 eth1
10.1.5.0        0.0.0.0         255.255.255.0   U     0      0        0 bnep0
194.182.170.0   0.0.0.0         255.255.255.0   U     0      0        0 eth0
169.254.0.0     0.0.0.0         255.255.0.0     U     0      0        0 eth0
0.0.0.0         194.182.170.1   0.0.0.0         UG    0      0        0 eth0
```

```
# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.2.5.0        0.0.0.0         255.255.255.0   U     0      0        0 bnep0
```

The localhost has a default Gateway because it's connected to the Internet but it has no route to the Slave Gumstix, let's add each host on each Slave with ***route add***.

```
[root@localhost ~]# route add -host 10.2.5.2 gw 10.1.5.1
```

```
# route add -host 10.1.5.2 gw 10.2.5.1
```

This means that we will add a host (10.2.5.2 on the fist case and 10.1.5.2 on the second) through another device (10.1.5.1 and 10.2.5.2, which is the Master Bluetooth Gumstix).
Check both connections with ***ping***:

```
[root@localhost ~]# ping 10.2.5.2
PING 10.2.5.2 (10.2.5.2) 56(84) bytes of data.
64 bytes from 10.2.5.2: icmp_seq=0 ttl=63 time=264 ms
64 bytes from 10.2.5.2: icmp_seq=1 ttl=63 time=282 ms
64 bytes from 10.2.5.2: icmp_seq=2 ttl=63 time=277 ms

--- 10.2.5.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 264.392/275.079/282.951/7.857 ms, pipe 2
```

```
# ping 10.1.5.2
PING 10.1.5.2 (10.1.5.2): 56 data bytes
64 bytes from 10.1.5.2: icmp_seq=0 ttl=63 time=283.2 ms
64 bytes from 10.1.5.2: icmp_seq=1 ttl=63 time=282.5 ms
64 bytes from 10.1.5.2: icmp_seq=2 ttl=63 time=280.1 ms

--- 10.1.5.2 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 280.1/281.9/283.2 ms
```

Let's see the routing table on each device again with ***route -n***.

```
[root@localhost ~]# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.2.5.2        10.1.5.1        255.255.255.255 UGH   0      0        0 bnep0
192.168.0.0     0.0.0.0         255.255.255.0   U     0      0        0 eth1
10.1.5.0        0.0.0.0         255.255.255.0   U     0      0        0 bnep0
194.182.170.0   0.0.0.0         255.255.255.0   U     0      0        0 eth0
169.254.0.0     0.0.0.0         255.255.0.0     U     0      0        0 eth0
0.0.0.0         194.182.170.1   0.0.0.0         UG    0      0        0 eth0
```

```
# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.1.5.2        10.2.5.1        255.255.255.255 UGH   0      0        0 bnep0
10.2.5.0        0.0.0.0         255.255.255.0   U     0      0        0 bnep0
```

As you can see there is a **Host** through a **Gateway** (**UGH**) the U means that the route is **UP**.

[FIXME]

Auto configuration

If you want the Gumstix to remember the values of the USB-net and Ethernet connection and/or to set them up automatically you just have to edit the file:

/etc/network/interfaces

and set the proper values for each connection:

```
auto usb0
iface usb0 inet static
        address 10.0.0.1
        netmask 255.255.255.0
        network 10.0.0.0
        broadcast 10.0.0.255
```

```
auto eth0
iface eth0 inet static
        address 192.168.0.201
        netmask 255.255.255.0
        network 192.168.0.0
        broadcast 192.168.0.255
```

or:

```
auto eth0
iface eth0 inet dhcp
```

for a DHCP automatic eth0 configuration.

Once you have this file its easier to bring up an interface with the command ***ifup***:

```
# ifup eth0
```

```
# ifup usb0
```

5. Transferring files to Gumstix

5.1. SCP

The usual way of transferring files to the Gumstix is using ***SCP*** (*Secure CoPy*)

over any of all the connection types seen before. The typical use is:

scp file_name user@destination_ip:/destination/directory

Let's see an example:

```
[root@localhost ~]# scp hello_gumstix_world root@10.0.0.1:/root/
root@10.0.0.1's password:
hello_gumstix_world
100% 4703     4.6KB/s   00:00
[root@localhost ~]#
```

this will transfer the file named ***hello_gumstix_world*** to the Gumstix directory ***/root/*** using the USB-net connection. As you see ***SCP*** asks for the user password, in this case you will have to type the Gumstix password.

6. Web server setup

The embedded web server that has been chosen is Boa [5]. This web server comes with CGI [6] support and also can run PHP [7, 8] scripts.
If you haven't reflashed your Gumstix it's built in and usually it's always running so the usual way to see the test page is going to the Gumstix IP address with any navigator. If you reflashed your Gumstix you had to have chosen the option «*boa*» on the menu «*Package Selection for the target*» before compiling.

To start, stop or restart the server just play with the daemon:

```
# /etc/init.d/S50httpd
$Usage: /etc/init.d/S50httpd {start|stop|restart}
```

The default pages are always on the same directory, the same for nearly all GNU/Linux systems:

```
# cd /var/www/
# ls
brd.jpg         gumlogo.gif     warranty.html
cgi-bin         index.html      ws.jpg
```

just upload your files as seen in *section 5*.

If you want to fine tune the server you can edit the file ***boa.conf*** located in the directory ***/etc/boa***
to change thing such as the port, the servername, setting up a virtual host, the DocumentRoot, etc.

7. Reflashing

The steps you have to follow in order to upgrade or downgrade the software revision of a Gumstix are the following:

- Getting a new software revision
- Compiling the software revision
- Transferring the new file system to the Gumstix

If everything has gone right the next time you boot the Gumstix it will load the revision you compiled.

7.1. Getting s software revision

The common way to get a Gumstix software revision is using ***subversion*** (***svn***), the successor of ***CVS*** (*concurrent version system*).
To download a version of the software you must have to have installed svn tools on your machine, then on the usual directory to download sources for compiling under GNU/Linux witch is /usr/src/ you will execute the next sentence [6]:

```
[root@localhost gumstix]# svn co http://svn.gumstix.com/gumstix-buildroot/trunk
gumstix-buildroot
```

That will download to your machine the latest revision of the software into the directory
/usr/src/gumstix/gumstix-buildroot

To get older software revisions you will have to add an option to the svn command:

```
[root@localhost gumstix]# svn co -rXXX http://svn.gumstix.com/gumstix-
buildroot/trunk gumstix-buildroot-XXX
```

Where XXX is the number of the revision.
It's important to name correctly the directories in order to not mistake revisions among each others.
Getting older revision has some troubles with the files downloaded from the Internet, it's explained on *section 8.4*.

There is also another way to get a software revision for reflashing, the Gumstix Web page at Sourceforge [7] has a complete filesystems and U-Boots that you can use without compiling.

7.2. Compiling a software revision

The way to compile a Gumstix software revision is using ***make menuconfig*** on newer versions. On the graphical menu you will have to select all the options you want for the Gumstix file system, then exit the menu and then do ***make***.

```
[root@localhost gumstix-buildroot-914]# make menuconfig
#
# using defaults found in .config
#


*** End of Buildroot configuration.
*** Check the top-level Makefile for additional configuration options.

[root@localhost gumstix-buildroot-914]# make
```

This is an example of how to build the 914 revision.

On old revisions you have to edit the top level ***Makefile*** file as explained on *section 8.4*.

The make will take a while depending on your machine -as explained on section 8.2- and it needs to download some files from the Internet like the Linux Kernel sources, the Bluetooth utils, the wireless tools, the GCC compiler, etc, so the whole size of the directory once compiled (with the object files) and with the downloaded files will be about 2 GB, be aware of that amount of free disk space on your machine.

When choosing options once inside the make menuconfig menu you would like to select

- Build/install c++ compiler and libstdc++?

- Build/install Objective-C compiler and runtime?

from the *«Toolchain»* submenu in order to have C++ compiler.

And if you would like to have a DHCP client on your Gumstix you should select

- dhcp support
- dhcp client

from the *«Package Selection for the target»* submenu.

Once compiled and if you didn't get any errors on the top level directory you will have three binary files for the Gumstix:

- ***rootfs.arm_nofpu.jffs2***
- ***u-boot.bin***
- ***u-boot.srec***

The first one is the new complete file system which is what you are looking for. The second one is a new u-boot loader for the Gumstix if you want to replace the older one. The last one is the target for the bootloader in SREC format which can be loaded to RAM using a JTAG debugger.

The normal ***rootfs.arm.nofpu.jffs2*** file size has to be between 2.7~3.5 MB, if your file size is bigger maybe you did something wrong while choosing the compiling options or you recompiled the sources without cleaning the old object and config files.
To solve the fist case you will have to clean the object files and run ***make menuconfig*** again paying attention to what you choose.

```
[root@localhost gumstix-buildroot-914]# make clean
[root@localhost gumstix-buildroot-914]# make menuconfig
#
# using defaults found in .config
#


*** End of Buildroot configuration.
*** Check the top-level Makefile for additional configuration options.

[root@localhost gumstix-buildroot-914]# make
```

To solve the second case just clean the object files and rebuild.

```
[root@localhost gumstix-buildroot-914]# make clean
[root@localhost gumstix-buildroot-914]# make
```

There is another case in which your ***rootfs.arm.nofpu.jffs2*** file size will be bigger than 7MB, this will happen if you compile the buildroot with the JAVA support -also explained in *section 8.4*-. You have to be careful with the size of your buildroot system because maybe it won't fit on the Gumstix.

7.3. Transferring a filesystem to the Gumstix

Now that you have a valid ***rootfs.arm.nofpu.jffs2*** file you may want to upload it to the Gumstix [6].
The first step to load a new filesystem is to stop the U-Boot loader of the Gumstix when it's booting, pressing any key it will stop and listen for commands.

```
GUM> loadb a2000000
```

This command will prepare the Gumstix for data receival.

Then you have to come back to the Kermit mode using ***Control \ c*** and send the image:

```
(/root/) C-Kermit>robust
(/root/) C-Kermit>send root_fs_arm_nofpu.jffs2
```

This will take a while.
Once the new filesystem has been uploaded you will have to erase all on the Gumstix but the U-Boot:

```
(/root/) C-Kermit>connect
GUM> protect on 1:0-1
GUM> erase all
GUM> cp.b a2000000 40000 ${filesize}
```

And then reboot the Gumstix:

```
GUM> boot
```

It's possible that you get a kernel panic while booting with the new filesystem

```
Kernel panic - not syncing: VFS: Unable to mount root fs on unknown-block(31,2)
```

especially if the revision you have built is quite new, then you will have to change some boot arguments:

```
GUM> set bootargs console=ttyS0,115200n8 root=1f01 rootfstype=jffs2
GUM> saveenv
GUM> bootd
```

And it should fix the problem.

8. Comments

8.1. Kernel messages under Fedora Core 4

As Red Hat Fedora Core doesn't log the kernel messages maybe you would like to change the configuration to see the messages that the Gumstix sends to the localhost or vice versa, on the file:

```
/etc/syslog.conf
```

And replace the line that referrers to the kernel linking to a file that you can read later:

```
#kern.*                                                 /dev/console
kern.*                                                  /var/log/kernel.log
```

Now you can see the kernel log messages on a terminal using the expression: ***tail -f /var/log/kernel.log***:

```
[root@localhost ~]# tail -f /var/log/kernel.log
Mar 30 14:26:13 localhost kernel: usb 2-2: USB disconnect, address 13
Mar 30 14:26:13 localhost kernel: usb0: unregister usbnet usb-0000:00:1f.4-2,
Linux Device
Mar 30 14:26:14 localhost kernel: usb 2-2: new full speed USB device using
uhci_hcd and address 14
Mar 30 14:26:15 localhost kernel: usb 2-2: configuration #1 chosen from 2
choices
Mar 30 14:26:15 localhost kernel: usb0: register usbnet at usb-0000:00:1f.4-2,
Linux Device, ca:d3:c4:bc:2c:bf
```

This is an example of a message from the kernel when the Gumstix USB cable is unplugged and then plugged.

8.2. Compiling

Compiling the Gumstix software revision is like compiling a Linux kernel, it needs a powerful machine, for that it's recommended a machine with at least 512 MB of RAM but it's better if the machine has around 1 GB. If your machine has 256 MB of RAM or fewer it will take about four hours to compile any software revision.

8.3. Bluetooth USB sticks

Sometimes the computer that has the Bluetooth USB stick hangs up without any normal reason. It usually happens when the Bluetooth service is started (or stopped and started again), doing a ***hcitool scan***, ***hciconfig hci0 up*** and some other times.

Sometimes the solution is start the Bluetooth services (***pand -s***, ***hciconfig hci0 up***, ***/etc/init.d/bluetooth start***, etc) in one terminal and jump to another terminal.

8.4. Old software revisions

With the old Gumstix software revisions you need to edit the ***Makefile*** with a text editor to set your configuration before compiling, activating the JAVA compiler doesn't modify the system build-root image but in the newer versions, the ones that use ***make menuconfig*** to set the configuration, it includes the JAVA library on the system so the hole build-root image will have a size bigger than 7 MB.
Also, as all the revisions, it will search on the Internet to download some sources needed like the Linux Kernel or the GCC compiler, you have to be very careful and patient with this because some links are broken and you may have to download the files by hand and then put them into the specified directory:

~/sources/dl

where ~ means the top level directory where your revision is, if your revision is some newer then the downloads directory will be in:

~/dl

9. Referrals

[1]: http://www.gumstix.com
[2]: http://www.gumstix.com/platforms.html
[3]: http://blog.lobstertechnology.com/2006/02/02/
[4]: http://em.typodemon.com/wordpress/?p=31
[5]: http://www.boa.org
[6]: http://sourceforge.net/projects/gumstix
[7]: http://www.gumstix.org/tikiwiki/tiki-index.php?page=tutorial

[FIXME]

[]: http://en.wikipedia.org/wiki/Common_Gateway_Interface
[]: http://www.php.net

10. Annex

Written by:
Iván Nieto Castaño
Jorge González González