IMote2 as platform for intelligent sensors under Linux OS
Part I

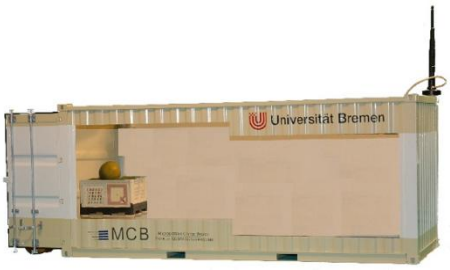
Installation, Sensors and Other Drivers, USB-Host Support.

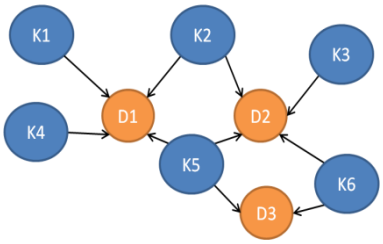
Technical Report SFB637- B6-2010-2,
University Bremen, August 2010

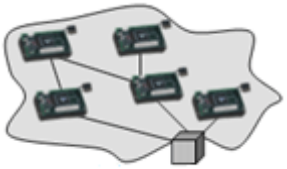
Author: Ishwar Lal

Supervisor: Reiner Jedermann,
Walter Lang

Thursday, September 09, 2010







IMote2 as platform for intelligent sensors under Linux OS

PART 1

Subtitles: Installation, Sensors and Other Drivers, USB-Host Support.

Technical Report of the CRC637, University Bremen , Bremen , 2010, SFB637- B6-2010-2

Lal, I.; Jedermann, R.; Lang, W.

Revision Errata

Version 1.0	The information was first time properly organized and written in the form of document.
Version1.1	Sections 2.4 CPU Frequency Module, 3.3 USB-2-Ethernet Adapter Interfacing with Imote2 and 3.4 UMTS/GSM Surf-Stick Interfacing With Imote2 were added.

Contents

Table of Figures

List of Tables

CHAPTER 1

1 Installation

In this chapter we will discuss how to compile images for Linux kernel and file-system plus how to install boot-loader, Linux kernel and file-system on Imote2. At the time when this document was written we were working on Linux version 2.6.29.1_r1.1 (from git), so most of work is described for this version of Linux but it should be valid for other versions also.

1.1 Compiling Linux and File-system Images

There are two ways to get these Images one way is to get directly from internet these images are available on www.sourceforge.net. The other way to get images is to compile them from sources; this part will describe the procedure of compiling Linux kernel and File-system images from scratch.

1.1.1 Compiling Linux Kernel

First change your current directory to the Linux kernel source directory and follow the given steps (here I assume that you have already set your arm cross compiler tool chain, downloaded the desired kernel source code and unpacked the achieve).

```
# export ARCH=arm
#export CROSS_COMPILE=arm-linux-
#make imote2-linux_defconfig
#make menuconfig
```

Here the command “make imote2-linux_defconfig” copies default configuration file from “./arch/arm/configs/imote2-linux_defconfig” to root directory of kernel source and renames it with name “.config”. You don’t really need to run this command every time you compile the same kernel code, you only need to run it for first time. Please remember you need “ncurses library” to run the “make menuconfig” command. If all went correct by this point you will see a blue screen on your Linux command line with some fancy text interface. On this screen you can select, deselect or select as modules certain features of Kernel. Generally you need not to touch anything, just exit from this screen. If you already have a valid configuration file “.config” and don’t want to make any further changes in kernel you can skip the “make menuconfig” command.

```
#make zImage
```

Above command will compile the Linux kernel and will take a bit time, when completed issue following command.

```
#make modules
```

Above command will compile all those features which have been selected as modules, finally issue the command to collect all the modules in one single directory, you will need these modules later, you need to copy these modules to file-system before compiling file-system.

```
#make INSTALL_MOD_PATH=$PWD/modules  modules_install
```

If all went ok by this point you will have a compiled kernel image in the "../arch/arm/boot/" directory with file name "zImage". You will find all the modules in "../modules/lib/modules/" directory.

1.1.2 Compiling File-System

To compile the root file-system for Imote2 you need two things first one is the file system itself and second one is mkfs.jffs2 tool. The basic file system is available from www.soufcefourge.net as *.tgz file. To get mkfs.jffs2 tool you need to install mtd-tools, following command should do this on a Debian GNU Linux machine.

apt-get install mtd-tools

The set of tools installed by mtd-tools (memory technology device tools) also contains mkfs.jffs2 tool.

Steps to follow:

1. Untar and Unzip the original file-system downloaded from internet.
2. Patch the file system i-e make any custom changes that you want, see section "1.1.2.1 Recommended Changes for File System" for more information on custom changes.
3. Compile the file-system using one of the following commands depending on the memory available on your system (For Imote2 32MB is suitable).

   # Make 16MB file system
   mkfs.jffs2 --squash-uid -r ./linux-rootfs -o rootfs.jffs2 -e 0x20000 --pad=0x01000000

   # Make 32MB (29.75MB) file system
   mkfs.jffs2 --squash-uid -r ./linux-rootfs -o rootfs.jffs2 -e 0x20000 --pad=0x01DC0000

   Where "./linux-rootfs" represent the directory in which the file system is placed and "rootfs.jffs2" represent the name of target file.

1.1.2.1 Recommended Changes for File System

Virtually you can make any changes to file system that my suite your demand, unless they pose any problem to run the system. Although the changes to file system may depend on your requirements, but given below are some useful instructions that may simplify the startup process.

1. If you want you can change a file /etc/init.d/banner which defines that which banner is displayed during startup.
2. If you want any driver module to be loaded at boot time edit file /etc/modules.
3. If you want to create device nodes at boot time edit file /etc/init.d/devnodes.
4. If you want to change the network settings edit file /etc/network/interfaces.
5. If you want to change the default console (which is ttyS2) edit file /etc/inittab.
6. If you want to change the hostname of Imote2 edit file /etc/hostname.

7. If you want to add any extra commands or want to run particular applications on startup, edit file /etc/rc2.d/S50startUpScript (if this file doesn't exist, you can create it).
8. If you want to change SSH settings edit files S10networking & S11sshd in /etc/rc2.d directory.
9. Before you compile the file-system copy all the modules to "../lib/modules/" directory of the file-system.

Generally all files in /etc/init.d are scripts that execute at startup and files in /etc/rc2.d are symbolic links to files in init.d.

1.2 Installing Compiled Images on Imote2

In this part we will discuss the flashing of Images to Imote2, these Images can be Boot-loader, OS image or a file-system.

1.2.1 Installing Boot-Loader

To flash a boot loader you must have a boot-loader image, you can get the latest version of boot loader from www.soufcefourge.net. A boot-loader can be installed with help of a J-tag cable or with the help of an already existing boot-loader.

1.2.1.1 Installing Boot-Loader with J-Tag

This method is useful when Imote2 is being prepared first time for Linux, this method is time consuming and prone to errors so it must be carried out with great care.

We Need:

1. Jflashmm utility.
2. Diygadget jtag cable. The both other cables (Olimex Wiggler and Raven) did not work with this approach.
3. Imote2-Debug Board

Steps to follow:

1. Make sure that LPT1 is in ECP Mode (BIOS settings).
2. Install giveio.sys (Search for "Flash Memory Programmer for Intel Development Platforms" in google for more installation procedure, you can also download it from xbow.com).
3. Copy Flash_8810_1_16.dat to Flash_0_1_16.dat, to make sure it will be found, if flash type is not recognized correctly.
4. Reboot your computer.
5. Prepare the red JTAG board of Diygadget:
   Set jumper positions to:

PWR_SEL	Set jumper to left side	TDO_SEL	set Jumper on left side.
nRST	Set jumper to right	nTRST	Set jumper to right.

   Connect it with imote2's debug board with ribbon cable (20 PIN STD). Connect JTAG board to parallel port of PC.
6. Connect mini usb-cable from PC to Diygadget-board. (3.3v LED turns on).

7. Combine Imote2-Processor Platform and Imote2-Debug Board.
8. Connect a second mini-usb-cable from PC to imote2-processor board (not debug board).
9. Switch on Imote2.
10. Start Jflashmm (JFlashmm.exe bulbcx16 blob P 0 WIG)

Repeat steps 6-10 several times until success. The timing between powering and start of flashing is important (run JFlashmm command after 1 to 2 seconds of pressing the imote2 power button). Finally it works.

Here's the output:

E:\imote2\jflashmm2 >JFlashmm. exe bulbcx16 blob P 0 WIG

JFLASH Version 5.01.003
COPYRIGHT (C) 2000 - 2003 Intel Corporation

PLATFORM SELECTION:
Processor= PXA27x
Development System= Mainstone
Data Version= 1.00.001

error, failed to read device ID
check cables and power
ACT: 1111 1111111111111111 11111111111 1
*EXP: **** 1001001001100101 00000001001 1*

failed to read device ID for this Platform

E:\imote2\jflashmm2 >JFlashmm. exe bulbcx16 blob P 0 WIG

JFLASH Version 5.01.003
COPYRIGHT (C) 2000 - 2003 Intel Corporation

PLATFORM SELECTION:
Processor= PXA27x
Development System= Mainstone
Data Version= 1.00.001

PXA27x revision C5
Found flash type: 28F256L18B

Unlocking block at address 0
Erasing block at address 0
Unlocking block at address 8000
Erasing block at address 8000
Unlocking block at address 10000

Erasing block at address 10000
Starting programming
Using BUFFER programming mode...
Writing flash at hex address 123c0, 98.80% done
Programming done
Starting Verify
Verifying flash at hex address 126d0, 99.84% done
Verification successful!

1.2.1.2 Installing Boot-Loader with the Help of an Old Boot-Loader

By that time you must have noticed that loading Images with J-Tag and JFlashmm utility is a very annoying task. Life can be easier if you already have a boot-loader installed and you want to shift to another boot-loader, it can be used to upgrade or degrade from one version of boot-loader to another. This method uses Xmodem protocol.

We need:

1. Debug board
2. USB2serial driver installed on the pc for debug board.
3. An imote2 board with already loaded and working boot loader.
4. HyperTerminal program.

Steps to follow Installing Blob:

1. Stack the debug board and the Imote2 board together.
2. Connect one end of USB cable to debug board (not Imote2) and other to PC.
3. Now wait for a while and let the windows detect the hardware and load proper driver.
4. Open the Hyper Terminal program.
5. Select appropriate com port for communication (in my case port4).
6. Select following settings (Baudrate 115200, Data bits 8, Parity none, Stop bits 1, Flow Control none).
7. Now press power button of the Imote2 board (Pay attention at this point).
8. When the screen say “press any button within 2 seconds”, press a button to go into boot-loader.
9. When you see boot-loader command prompt enter command
   xdownload blob
10. From Transfer menu select Send file.
11. In new dialog browse to the location where you have saved your boot-loader image, from protocol dropdown box select 1k Xmodem.
12. Press Send and wait until the transfer is completed.
13. Enter command
    fwrite 0xa0200000 0x000000000 0x00040000

    Wait for completion of the process.
14. Reboot Imote2 by first pulling the usb cable out and then following steps 3 to 7.
15. If all went fine to this point your Imote2 will boot successfully.

Please note that in old boot-loader the waiting time was 2 seconds in new one it is 10 seconds.

1.2.2 Installing Linux Kernel

Now that we have a working boot loader we can install our Imote2-Linux. You can get the latest version of boot loader, Linux and File system from sourceforge.com or you can compile them. There are two methods to accomplish this task first one is with Xmodem protocol and second one is with the help of already installed Linux-Kernel.

1.2.2.1 Installing Linux using Xmodem

This method is useful when you are setting up your Imote2 for first time.

We need:

1. Debug board
2. Pre-compiled image of Imote2-Linux.
3. USB2serial driver installed on the pc for debug board.
4. An imote2 board with already loaded and working boot loader.
5. HyperTerminal program.

Steps to follow Installing Linux:

1. Stack the debug board and the Imote2 board together.
2. Connect one end of USB cable to debug board (not Imote2) and other to PC.
3. Now wait for a while and let the windows detect the hardware and load proper driver.
4. Open the Hyper Terminal program.
5. Select appropriate com port for communication (in my case port4).
6. Select following settings (Baudrate 115200, Data bits 8, Parity none, Stop bits 1, Flow Control none).
7. Now press power button of the Imote2 board (Pay attention at this point).
8. When the screen say “press any button within 2 seconds”, press a button to go into boot-loader.
9. When you see the boot-loader command prompt enter command

   xdownload kernel

10. From Transfer menu select Send file.
11. In new dialog browse to the location where you have saved your kernel image, from protocol dropdown box select 1k Xmodem.
12. Press Send and wait until the transfer is completed.
13. Enter command

    fwrite 0xa0008000 0x00040000 0x00200000

    Wait for completion of the process.
14. Reboot Imote2 by first pulling the USB cable out and then following steps 3 to 7.
15. If all went fine to this point your Imote2-Linux will boot successfully.

1.2.2.2 Installing Kernel with the help of a Running Kernel

It is also possible to use a running kernel on Imote2 to flash another kernel, such an approach can be used to upgrade the kernel version, also it is quicker and don't need any extra hardware. Note that for this method to work you should have a fully working Imote2 with boot-loader, kernel Image, file-system and some mean of communication. We used USB-Ethernet gadget driver and ssh in Imote2 as mean of communication from host system to Imote2.

First of all you have to copy the new kernel to Imote2, copy the kernel image to root directory of Imote2 or where ever you like. You can use following command if ssh is properly configured on Imote2 as well as on host system.

#scp <path/to/image>/zImage root@xxx.xxx.xxx.xxx:zImage

In this command replace xxx.xxx.xxx.xxx with IP address of your Imote2 and provide a valid path where your kernel image is saved.

Secondly login to your Imote2 using ssh or any other method you like. Now to replace the old Kernel Image with a new Image from the running Linux type this command on command prompt of Imote2:

#flashcp -v <zImage Location> /dev/mtd1

Once the process is complete you can restart the Imote2, to make sure that the intended operation was successful type following command before and after installing new kernel.

#uname -a

1.2.3 Installing File-System

When everything else in place what we need is only a file-system.

We need:

1. Debug board
2. Pre-compiled image file-system.
3. USB2serial driver installed on the pc for debug board.
4. An imote2 board with already loaded and working boot loader.
5. HyperTerminal program.

Steps to follow Installing Linux:

1. Stack the debug board and the Imote2 board together.
2. Connect one end of USB cable to debug board (not Imote2) and other to PC.
3. Now wait for a while and let the windows detect the hardware and load proper driver.
4. Open the Hyper Terminal program.
5. Select appropriate com port for communication (in my case port4).
6. Select following settings (Baudrate 115200, Data bits 8, Parity none, Stop bits 1, Flow Control none). Press apply and ok.
7. Now press power button of the Imote2 board (Pay attention at this point).

8. When the screen say “press any button within 2 seconds”, press a button to go into boot-loader.
9. When you see boot-loader command prompt enter command
   xdownload ramdisk

10. From Transfer menu select Send file.
11. In new dialog browse to the location where you have saved your file system image, from protocol dropdown box select 1k Xmodem.
12. Press Send and wait until the transfer is completed (note it will take about an hour or two).
13. Enter command
    fwrite 0xa0500000 0x00240000 XXXXXXXX

    Where substitute the XXX’s with the size of data transferred and showed after completion of previous command. Wait for completion of the process (it will also take a little long).
14. Reboot Imote2 by first pulling the USB cable out and then following steps 3 to 7.
15. If all went fine to this point your Imote2-Linux will boot successfully.

While you are in <blob> you can type “help” command to get list of all commands, also you can type “help <command>” to get help of a particular command. You can also use “boot” command from <blob> to boot the kernel.

CHAPTER 2

2 Sensors and Other Drivers

This chapter describes how to access data from the basic sensor board, how to use serial port of Imote2 under Linux and how to use radio for wireless communication.

2.1 Sensor Board Interface

Intel Imote2 comes with a basic sensor board "ITS400" this board contains a basic sensor suite. The sensor suite contains following general sensors:

- ST Micro LIS3L02DQ 3d 12 bit ±2g accelerometer
- High Accuracy, ±0.3°C Sensirion SHT15 temperature/humidity sensor
- TAOS TSL2651Light Sensor
- Maxim MAX1363 4 Channel General Purpose A/D for quick prototyping
- TI Tmp175 Digital Temperature Sensor with two-wire interface

Figure 2-1 ITS400 Basic sensor board

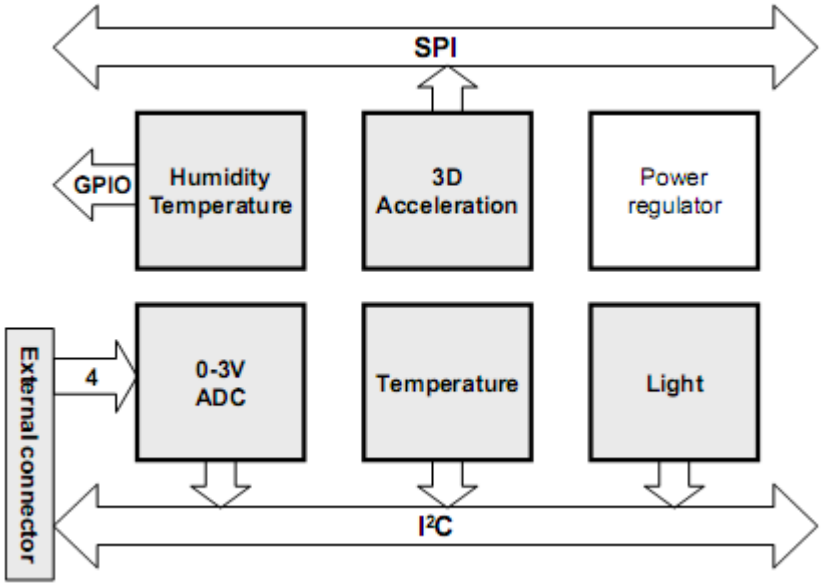


Figure 2-2 ITS400 Basic sensor board block diagram

Figure 2-1 and Figure 2-2 shows the sensor board and the block diagram of the board. Drivers for these sensors are already present in file system under directory "/lib/modules/<kernel-version>/kernel/drivers/" where <kernel-version> represent your current Linux kernel version.

2.1.1 SHT15 Temperature & Humidity Sensor

SHT15 sensor by Sensirion is high accuracy environmental parameters sensor. This sensor is interfaced with Imite2 via two GPIO pins, GPIO 100 to data pin and GPIO 98 to clock pin, for more information see the SHT15 data sheet by manufacturer or see the Imote2 Hardware reference manual.

To start measurement from SHT15 sensor you must first load the SHT15 module which is located in "/lib/modules/<kernel-version>/kernel/drivers/hwmon". Use "modprobe sht15" command to load the module. After that you have loaded the module check in the directory "/sys/class/hwmon/hwmonX/device" where "X" in "hwmonX" is a number which increments automatically as you load more modules of same type e-g if you first load TMP175 module and then load the SHT15 module you will have X = 0 for TMP175 and X = 1 for SHT15. There is another way to make it sure that which directory belongs to which sensor, see following screen output:

```
# cd /sys/class/hwmon/hwmon0/device
# cat name
sht15
```

After that you are sure that you are in correct directory, you can take a test reading from the sensor direct from your command prompt before you actually write a c code. See the following screen output:

```
# cd /sys/class/hwmon/hwmon0/device
# cat temp1_input
28698
# cat humidity1_input
17088
```

We see that by directly reading the files "temp1_input" and "humidity1_input" we can get the direct reading from the sensor. The reading seems to very strange and it not clearly understandable, but it is actually displayed in milli degree centigrade and in milli percent humidity. To use the sensor in your c programs you should open these files in your program and read them in your program. Please note that for each reading you have to re-open the file and then you can read the new reading.

2.1.2 TI TMP 175 Digital Temperature Sensor

This temperature sensor is interfaced with Imote2 via I2C bus, the interrupt line (TEMP_ALERT) is connected to GPIO96 (FF_TXD). To start measurement with LM175 sensor you must first load the corresponding module which is located in "/lib/modules/<kernel-version>/kernel/drivers/hwmon". Use "modprobe lm75" command to load the module, you don't need to load I2C module as the LM75 driver uses the default "i2c-adapter". After that you have loaded the module check in the directory "/sys/class/hwmon/hwmonX/device" where "X" in "hwmonX" is a number which increments automatically as you load more modules of same type e-g if you first load TMP175 module and then load the SHT15 module

you will have X = 0 for TMP175 and X = 1 for SHT15. There is another way to make it sure that which directory belongs to which sensor, see following screen output:

 # cd /sys/class/hwmon/hwmon1/device
 # cat name
tmp175

After that you are sure that you are in correct directory, you can take a test reading from the sensor direct from your command prompt before you actually write a c code. See the following screen output:

 # cd /sys/class/hwmon/hwmon1/device
 # cat temp1_input
27500

We see that by directly reading the file "temp1_input" we can get the direct reading from the sensor. The reading seems to very strange and it not clearly understandable, but it is actually displayed in milli degree centigrade. To use the sensor in your c programs you should open this file in your program and read it in your program. Please note that for each reading you have to re-open the file and then you can read the new reading.

2.1.3 TAOS TSL2651 Light Sensor

The TAOS TSL2651 Light Sensor is interface to Imote2 via I2C bus and the interrupt pin (LIGHT_INT) is connected to GPIO99. To access measurement data from TSL2651 you must first load the corresponding module which is located in "/lib/modules/<kernel-version>/kernel/drivers/industrialio/light". Use "modprobe tsl2561" command to load the module, you don't need to load I2C module as the TSL2651 driver uses the default "i2c-adapter". After that you have loaded the module check in the directory "/sys/class/industrialio/iio:deviceX" where "X" in "iio:deviceX" is a number which increments automatically as you load more modules of same type e-g if you first load Accelerometer module and then load the TSL2651 module you will have X = 0 for Accelerometer and X = 1 for TSL2651. There is another way to make it sure that which directory belongs to which sensor, see following screen output:

 # cd /sys/class/industrialio/iio:device0/device
 # cat name
tsl2561

After that you are sure that you are in correct directory, you can take a test reading from the sensor, direct from your command prompt before you actually write a c code. See the following screen output:

 # cd /sys/class/industrialio/iio:device0
 # cat light_broadspectrum0
125
 # cat light_infrared0

980

We see that by directly reading the files “light_broadspectrum0” and “light_infrared0” we can get the direct reading from the sensor. To use the sensor in your c programs you should open this file in your program and read it in your program. Please note that for each reading you have to re-open the file and then you can read the new reading.

2.2 Serial Communication

Intel Imote2 has three serial ports named as

- STD-UART (Standard UART)
- FFUART (Fully functional UART)
- BUART (Blue tooth UART)

The Figure 2-3 shows pin description of the basic set of connectors.

The STD-UART is an asynchronous UART which is used by the Linux console to communicate with PC by using the Interface board. This UART does not have modem control pins.

FFUART according to some sources is used by some sensors on sensor board.

BUART is not used by any application so we can use this port for communication with outside world. Pins 5 to 9 of Hirose-DF9 31 pin connector expose the BUART of Imote2. We tried to communicate via all three UARTs but we only achieved success with BUART.

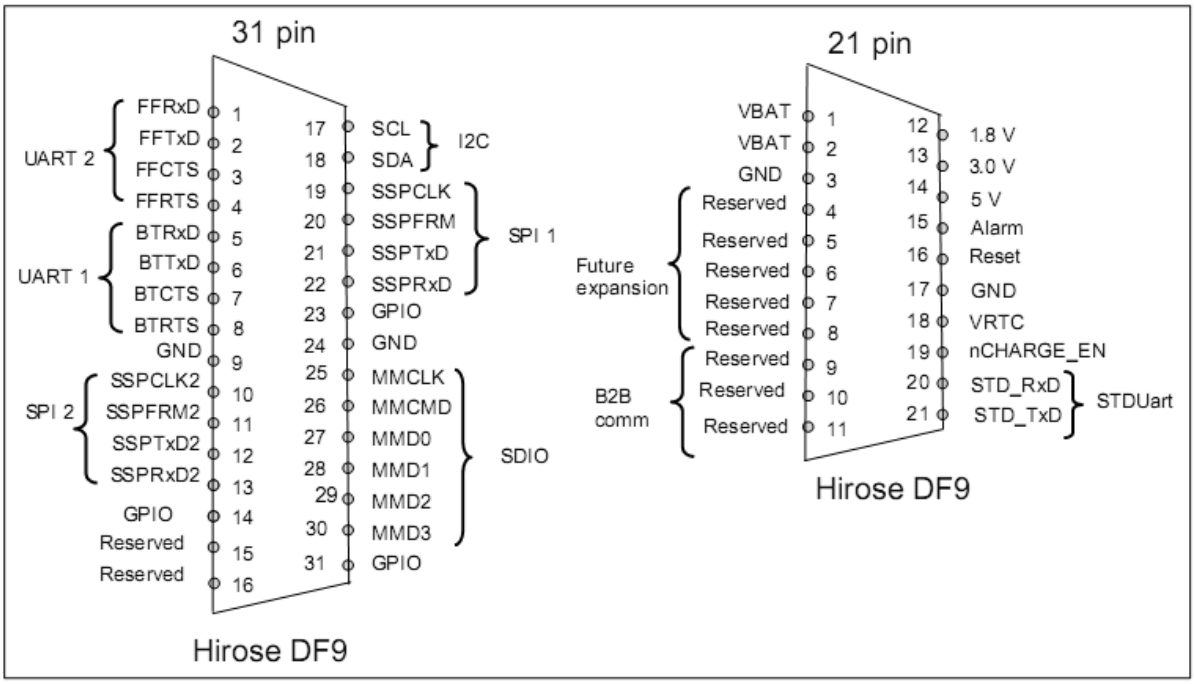


Figure 2-3 Imote2 basic connector set description

A small, level converter, circuit is also needed to convert the levels of the signals from TTL to RS-232 and to communicate with outside world. We have built our own serial adapter board which is described later in this section.

2.2.1 Serial Programming for Imote2 under Linux

The serial programming for Imote2 under Imote2 Linux is same as for standard Linux there is no change in programming procedure. There are so many documents on internet which provide enough information about serial programming under Linux. Only thing that you need to check under Imote2 Linux is that the serial devices are present, configured and installed also what their reference numbers are.

Use the "*dmesg | grep ttyS*" command to get details of devices under any Linux distribution (may be you need root account):

```
# dmesg | grep ttyS
Kernel command line: root=/dev/mtdblock2 rootfstype=jffs2 console=ttyS2,115200
mem=32M
pxa2xx-uart.0: ttyS0 at MMIO 0x40100000 (irq = 22) is a FFUART
pxa2xx-uart.1: ttyS1 at MMIO 0x40200000 (irq = 21) is a BTUART
pxa2xx-uart.2: ttyS2 at MMIO 0x40700000 (irq = 20) is a STUART
console [ttyS2] enabled
```

This will give you a list of devices with names and device numbers.

2.2.2 Serial Adapter for Imote2

As we know that Imote2 is digital device and operates at TTL voltage levels but real world serial devices use RS232 levels. To convert from TTL to RS232 and vice versa we need a level converter circuit. Figure 2-4 shows the serial adapter circuit board which converts levels between TTL and RS232. Description of different parts of circuit its operation and settings are given below. This circuit is just for test purpose you can build your own as you like.

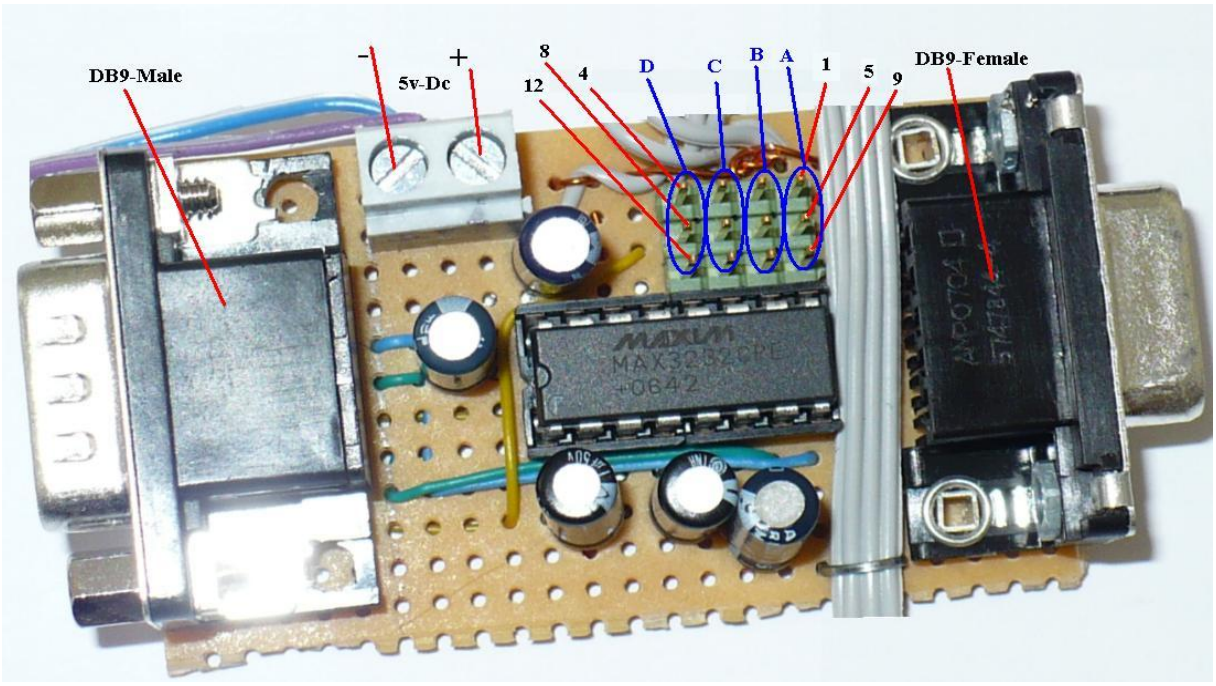


Figure 2-4 Description of parts for serial adapter board

The circuit contains twelve open pins which we can jumper to obtain different functionality. Table 2-1 describes different jumper pins.

Pins	Description
Pins 1 to 4	Belong to BUART of Imote2.
Pins 5 to 8	Belong to UART of PC or External Serial device.
Pins 9 to 12	Belong to FFUART of Imote2.

Table 2-1 Pin description

These pins are arranged in the form of groups to ease the different jumper settings, Table 2-2 describes these groups.

Group	Description
Group A	Group of RxD pins.
Group B	Group of TxD Pins.
Group C	Group of RTs Pins.
Group D	Group of CTs Pins.

Table 2-2 Pin group description

2.2.2.1 Jumper Setting

On the board there are three male headers each has 4 pins, the three headers are soldered on the board in form of a three by four matrix. This arrangement allows us to quickly change the functionality of the board from one mode to another. There are seven basic modes of operation which are described below.

1. BUART connected to external device.
2. FFUART connected to external device.
3. External device loop back.
4. BUART loop back.
5. FFUART loop back.
6. BUART connected to external device with lopped back hardware handshaking.
7. FFUART connected to external device with looped back hardware handshaking.

For example mode one can be set by connecting pin 1 to pin 5, pin 2 to pin 6, pin 3 to pin 7 and pin 4 to pin 8. Mode four can be set by connecting pin1 to pin 2 and pin 3 to pin 4. If you have some knowledge of serial communication you can easily figure out that how to use other modes.

2.2.2.2 Circuit Diagram

We have used Max232 IC from maxim as level converter. The IC Max232 contains two TTL to RS232 converters and two RS232 to TTL converters. Figure 2-5 shows pin description of Max232 IC. The Figure 2-6 shows a typical application circuit for Max232 IC, we have used same circuit in our serial adapter board. Along with Max232 we have also used five polar capacitors; four out of five capacitors (C1 to C4) are used by voltage doubler and voltage inverter inside the Max232 and the fifth capacitor (C5) is used to decouple the power supply noise.

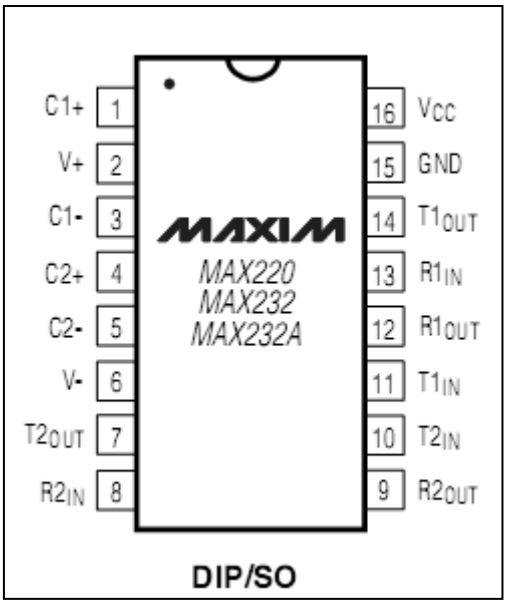
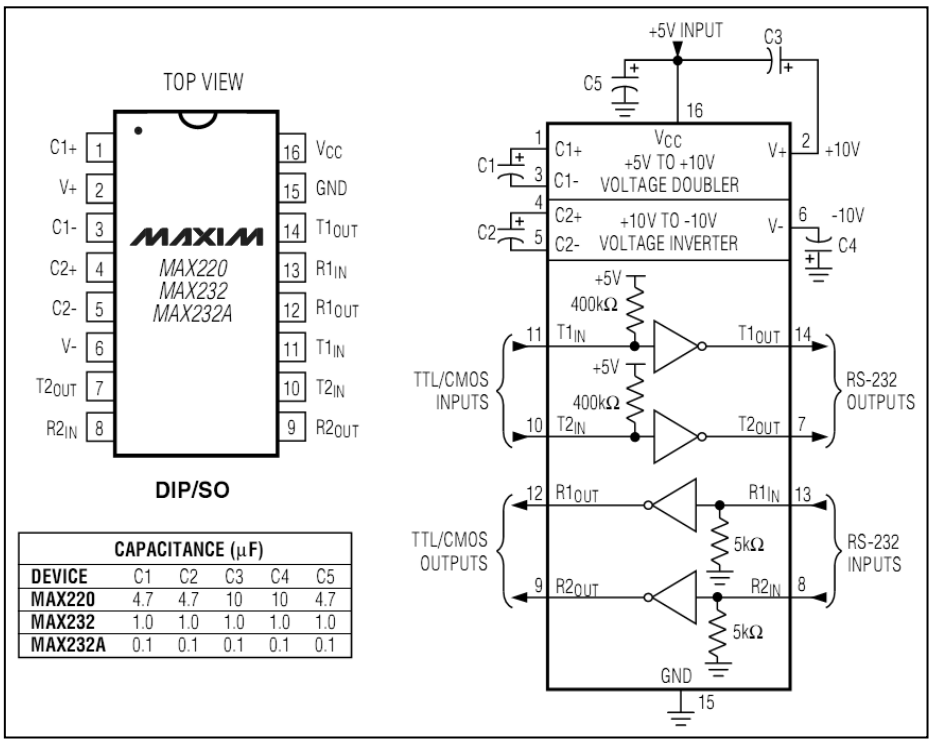


Figure 2-5 Pin description of Max232 IC

Figure 2-7 shows the full circuit diagram of the serial adapter board, parts J3, J4 & J5 are the three male headers which are used for jumper settings, already described above.



CAPACITANCE (µF)					
DEVICE	C1	C2	C3	C4	C5
MAX220	4.7	4.7	10	10	4.7
MAX232	1.0	1.0	1.0	1.0	1.0
MAX232A	0.1	0.1	0.1	0.1	0.1

Figure 2-6 A typical application circuit of Max232 IC

For a good understanding of circuit and the corresponding components on the board Figure 2-8 & Figure 2-9 shows the components mapping from circuit diagram to serial adapter board. In circuit diagram of Figure 2-7 J6 is the “Hirose-DF9 31 pin” connector which connects to Imote2 board, the physical connector is shown in Figure 2-9.

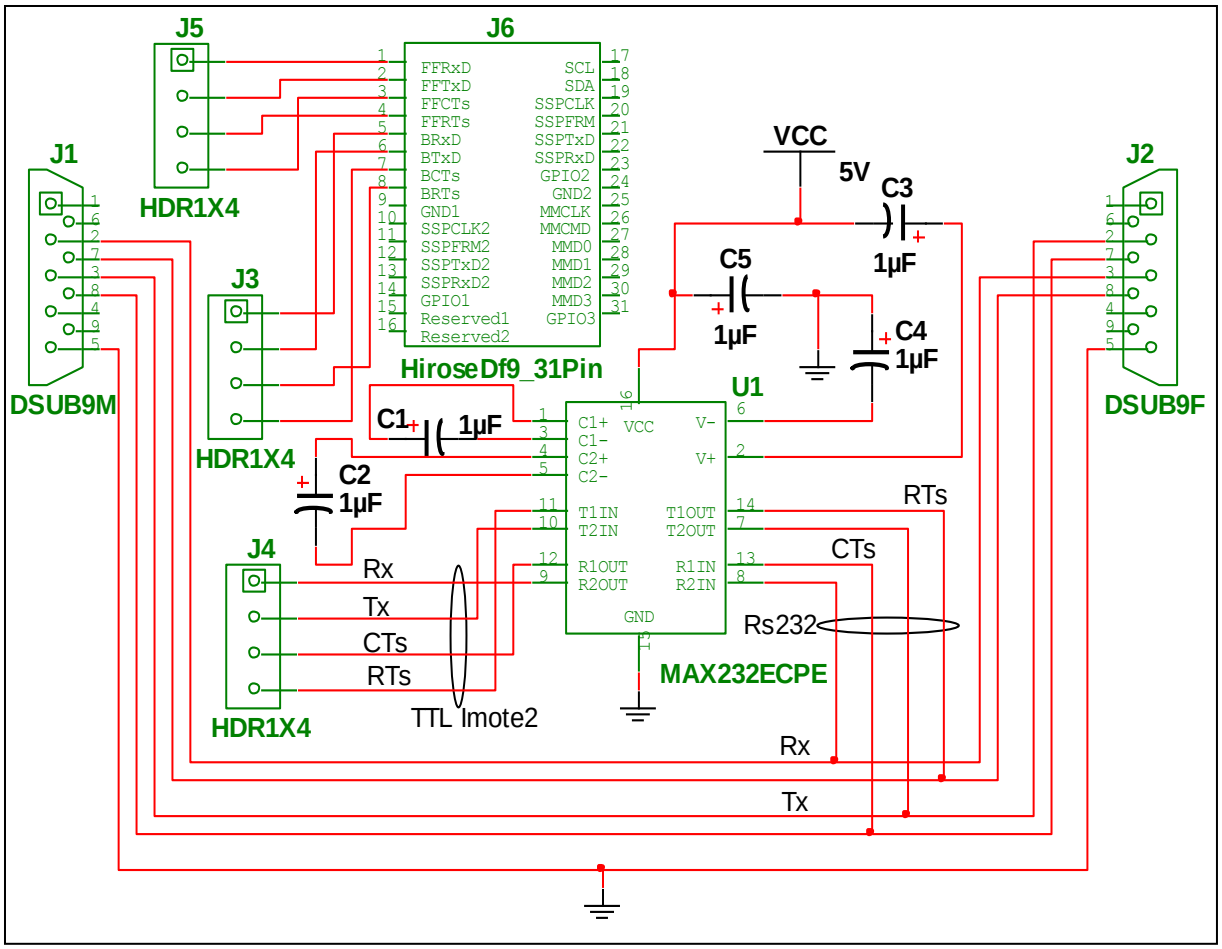


Figure 2-7 Serial adapter circuit diagram

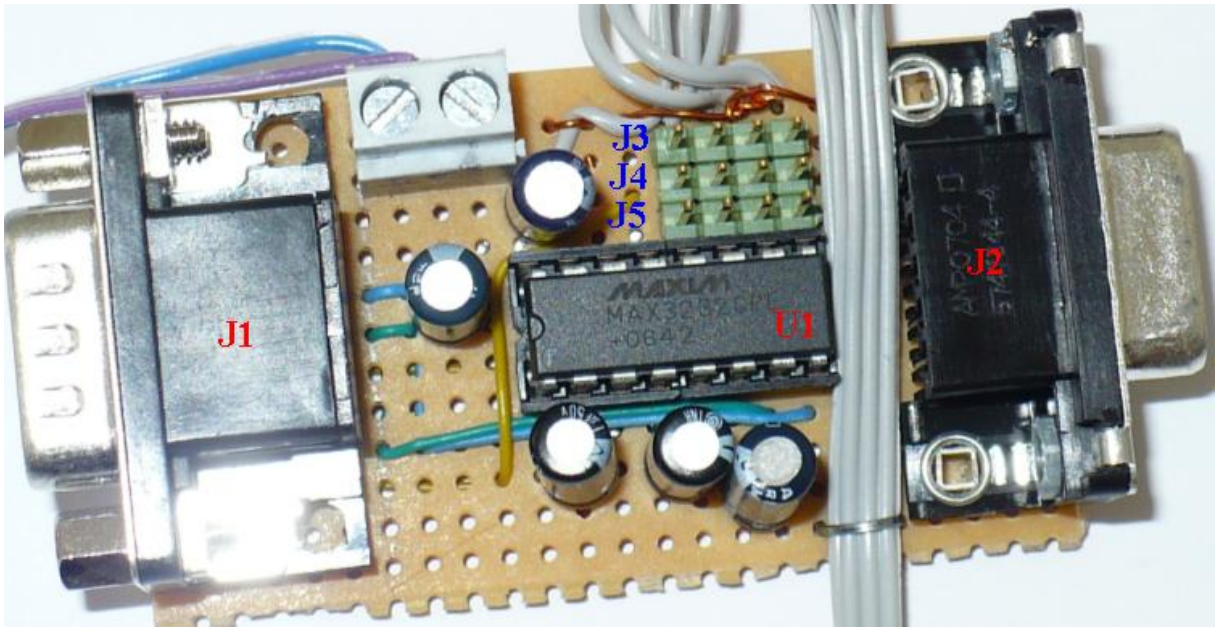


Figure 2-8 Component mapping from circuit diagram 1

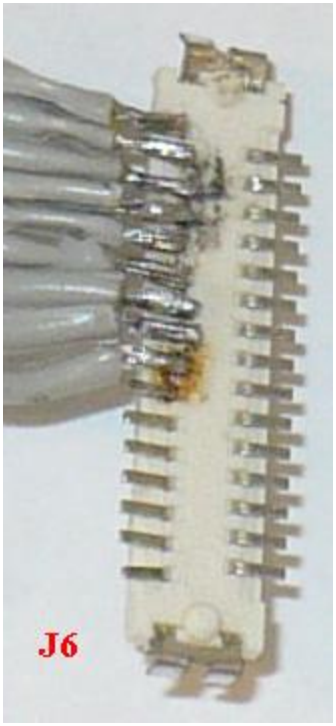


Figure 2-9 Component mapping from circuit diagram 2

Table 2-3 shows list of all components, their values and quantity used in the circuit. Please note that the power connector is not shown in the circuit diagram only the Vcc and ground symbols are used, one should place a suitable connector at a suitable place on the circuit board as shown in Figure 2-4.

Qty	Description	RefDes
1	Imote2, HiroseDf9_31Pin	J6
3	CONNECTORS, HDR1X4	J3, J4, J5
1	LINE_TRANSCEIVER, MAX232ECPE	U1
1	CONNECTORS, DSUB9F	J2
1	CONNECTORS, DSUB9M	J1
1	POWER_SOURCES, GROUND	0
5	CAP_ELECTROLIT, 1µF	C1, C2, C3, C4, C5
1	POWER_SOURCES, VCC	VCC

Table 2-3 List of components

2.3 Wireless Communication

Wireless communication under Linux is not difficult, sample code and compiled binaries are included in the file system under path "/root/tosmac". There are two applications one is to send data and other is to receive data. You can run these two applications on two separate motes to test them. But before you actually run the applications you have to load some modules and have to create some devices. We have tested this procedure on "Linux 2.6.29.1_r1.1" and it worked fine.

2.3.1 Installing and Running TOSMAC

The wireless communication under Linux is done with the help of “tos_mac” driver which is not built in the Linux kernel so you have to manually install it.

Here is the procedure:

Run following command (before you issue this command that all compiled modules are also copied to Imote2 in /lib/modules directory).

 # modprobe tos_mac

You will get this output:

TOSMAC driver is loading...
GPIO-22 autorequested
GPIO-114 autorequested
GPIO-116 autorequested
GPIO-0 autorequested
GPIO-16 autorequested
This board is IMote2
TOS-MAC driver installed

This will load “tos_mac” and all its dependences, next you need to create device node, use following command to create device node.

mknod /dev/tosmac c 240 0

In the command above number “240” is device major number and number “0” is device minor number. This number may vary from machine to machine. To know what is correct number for your device, type following command.

cat /proc/devices

Above command will produce following output:

Character devices:
1 mem
2 pty
3 ttyp
4 /dev/vc/0
4 tty
4 ttyS
5 /dev/tty
5 /dev/console
5 /dev/ptmx
7 vcs
10 misc
13 input
90 mtd

128 ptm
136 pts
240 tosmac
Block devices:
259 blkext
31 mtdblock

Finally here is the whole console output:

 # lsmod
Module Size Used by Not tainted

 # modprobe tos_mac
TOSMAC driver is loading...
GPIO-22 autorequested
GPIO-114 autorequested
GPIO-116 autorequested
GPIO-0 autorequested
GPIO-16 autorequested
This board is IMote2
TOS-MAC driver installed

 # cat /proc/devices
Character devices:
1 mem
2 pty
3 ttyp
4 /dev/vc/0
4 tty
4 ttyS
5 /dev/tty
5 /dev/console
5 /dev/ptmx
7 vcs
10 misc
13 input
90 mtd
128 ptm
136 pts
240 tosmac

Block devices:
259 blkext
31 mtdblock

 # mknod /dev/tosmac c 240 0

To run the applications you need to install driver for Leds, the application will work without this driver the only difference is that you will not see Leds lighting.

2.4 CPU Frequency Module

CPU operation frequency of Imote2 is scalable i-e it can be changed and the most interesting thing is that it can be changed in runtime. To use this feature all we have to do is to have appropriate software support. Linux kernel has CPU Frequency driver in Linux source tree, all we have to do is we have to enable that feature and use it. To enable the CPU Frequency driver we have to go through Linux kernel compilation steps.

2.4.1 Kernel Configuration

Before really starting with Linux-Kernel compilation process we must correct a small mistake in the source code. These changes are necessary because there is a small bug in original code, with this bug the CPU and SDRAM frequencies don't match and the Imote2 is not able to load the File-System.

Follow these steps:

1. Go to directory "../arch/arm/mach-pxa/" and find file called "cpufreq-pxa2xx.c" open this file in a editor.
2. Find the line.

   {416000, 208000, PXA27x_CCCR(1, 16, 4), 1, CCLKCFG2(1, 0, 1)}

3. Replace above line with following line.

   {416000, 208000, PXA27x_CCCR(0, 16, 4), 1, CCLKCFG2(1, 0, 1)}

4. Save and quit the editor.

Assuming that you already know how to make changes in Linux Kernel configuration (if you don't know how to see, section 1.1.1 Compiling Linux Kernel, for more information), start Linux kernel configuration and make following packages either built-in or compile as modules. Package compiled as modules must be loaded on startup.

CPU Power Management→CPU Frequency scaling [*]

→CPU frequency translation statistics<*>

→CPU frequency translation statistics details <*>

→Default CPU governor→performance

→'performance' governor<*>

→'power save' governor<M>

→'user space' governor<M>

→'on demand' governor<M>

→'conservative' governor<M>

→CPU idle PM support[*]

Packages marked with (M) are compiled as modules and should also be copied to Imote2. Now save and exit the Kernel configuration, compile the kernel and load the Image and modules to Imote2 (see Chapter 1 for compilation and installation instructions).

2.4.2 Usage of CPU-Frequency Driver

After installing the Linux kernel and copying the modules we must load the modules to this, enter the name of modules in the "/etc/modules" file of Imote2. Just copy following text to any section of this file and save it.

cpufreq_conservative
cpufreq_ondemand
cpufreq_powersave
cpufreq_userspace

Reboot your Imote2 and now if everything was fine till this step you will se a directory called "/sys/devices/system/cpu/cpu0/cpufreq/" on your Imote2. There are several files in this folder all these files are related to CPU-Frequency. At this time the only important files for us is "scaling_governor", "scaling_available_governors" and "scaling_setspeed" the first file shows current governor and the second one tells which governors are implemented third one will be described later. A governor is a kind of policy or frequency setting. For example, performance governor is used to represent maximum operating frequency (which is 416 Mhz for Imote2), powersave governor represents the lowest operating frequency (which at the moment is set to 104 Mhz), more about governors and CPU-Frequency driver can found on internet and in Linux source tree under "../Documentation/cpu-freq" directory.

Now to change the CPU-Frequency you have to change the governor, for example if you want to set the maximum CPU-Frequency your should load the "performance" governor this is done by writing the governor name to file "scaling_governor", see the following commands.

#echo performance > scaling_governor *<to load performance governor>*
#echo powersave > scaling_governor *<to load power save governor>*
#echo userspace > scaling_governor *<to load user space governor>*

When you switch to "userspace" governor you can change to any frequency between maximum and minimum frequencies. This can be achieved by writing the desired frequency value to "scaling_setspeed" file (frequency should be given in kilo hertz), see the following commands to set the Frequency value to 208 MHz, you can use any other value.

#echo userspace > scaling_governor
#echo 208000 > scaling_setspeed

You can also check if you're the commands really produce any change or not, for this use the command "cat /proc/cpuinfo".

 # cat /proc/cpuinfo
Processor : XScale-PXA270 rev 7 (v5l)
BogoMIPS : 207.66
Features : swp half fastmult edsp iwmmxt
CPU implementer : 0x69
CPU architecture: 5TE
CPU variant : 0x0
CPU part : 0x411
CPU revision : 7

Hardware : IMOTE 2
Revision : 0000
Serial : 0000000000000000

In above console output you see "BogoMIPS:207.66" which is a rough estimate of CPU-Frequency as estimated by Linux-kernel.

CHAPTER 3

3 USB-Host Support

Intel PXA27xx processors comes with built-in USB-Host support which if configured can be used to interface other devices like Cameras, Flash drives, Wlan Sticks, UMTS sticks and virtually everything.

3.1 How to Enable USB-Host Support

To enable USB host support you have to make changes in board file which is "imote2.c" that lies in kernel source tree under directory "./arch/arm/mach-pxa". Secondly you have to enable "usb host side support" during kernel configuration at compile time. Third thing you need is a small circuit that supplies power to your USB devices and pulls out Imote2's USB-Host connections.

3.1.1 Modifying the Kernel Source

Follow these instructions to modify the "./arch/arm/mach-pxa/imote2.c" file.

1. Add following line to include section of "imote2.c" file.

   ```
   #include <mach/ohci.h>
   ```

2. Add following code lines anywhere in "imote2.c" file.

   ```
   /*
    * Configure USB Host (OHCI)
    * For Imote2 the following configuration is used:
    *  USB Port 1 is used as USB Host
    *  USB Port 2 is used as USB Gadget (as default for Imote2)
   */
   static int imote2_ohci_init(struct device *dev)
   {
           return 0;
   }
   static struct pxaohci_platform_data imote2_ohci_platform_data = {
           .port_mode      = PMM_NPS_MODE,
           .init           = imote2_ohci_init,
           .flags          = ENABLE_PORT1 | NO_OC_PROTECTION,
           .port_mode      = PMM_PERPORT_MODE,
           .power_budget = 150, //300
   };
   ```

3. Add following line of code to function "static void __init imote2_init(void)" this function is at the end of "imote2.c" file.

   ```
   pxa_set_ohci_info(&imote2_ohci_platform_data);
   ```

3.1.2 Kernel Configuration

To enable the USB-Host support on Imote2 we must alter the default Linux Kernel configuration for Imote2. Assuming that you already know how to make changes in Linux Kernel configuration (if you don't know how to see, section 1.1.1 Compiling Linux Kernel, for more information), start Linux kernel configuration and make following packages either built-in or compile as modules. Package compiled as modules must be loaded on startup.

Device Drivers→USB support→Support for Host-side USB <*>

→USB device filesystem [*]

→USB device class-device (DEPRECATED)[*]

→USB Monitor<*>

→OHCI HCD support<*>

This configuration should do the basic stuff, but to enable extra devices you may need to enable extra modules like USB mass storage support, USB to serial converter support etc. Here you can also select the device drivers for your hardware if it is already supported by Linux Kernel. Now save and exit the Kernel configuration, compile the kernel and load the Image and modules to Imote2 (see Chapter 1 for compilation and installation instructions).

3.1.3 Circuit Diagram

Now all we need a small circuit with USB-Female connector and power supply,

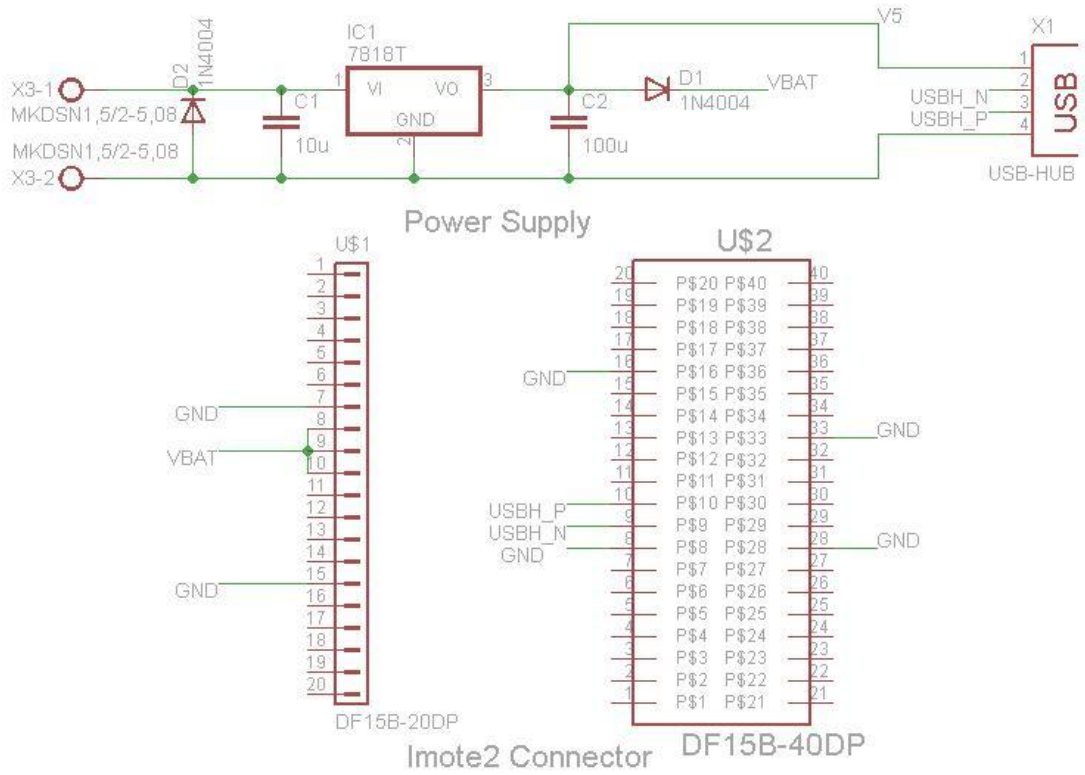


Figure 3-1 USB interface board circuit diagram

3.2 WLAN-USB Stick Interfacing With Imote2

Due the availability of USB host on Imote2 it is possible to interface virtually any USB device to Imote2, so we decided to give try to WLAN-USB stick. One may find countless applications for Imote2 with WLAN, so we thought WLAN is worth giving a try. Given below is the list of things that we need or we need to do.

- A WLAN-USB stick
- Linux Kernel with USB-Host support, TCPIP Stack and WLAN support enabled (see section 3.3 for how to enable TCP/IP Stack).
- WLAN-USB Stick driver.

3.2.1 Choice of WLAN stick

So our task begins with looking for an appropriate WLAN-USB Stick. One must keep in mind certain things when choosing hardware for Linux system, most important of those is driver support for Linux. So find a WLAN-USB stick which has a driver support for Linux. WLAN has many standards 802.11 a/b/g/n at this time standard "802.11 g" is the most stable version with 54Mbps data rate this is the one that we use commonly these days (at the time of writing of this document), standard "802.11 n" is the latest version and is very fast, data rate can go up to 300Mbps and may be higher (I am not sure), if you buy a "802.11 n" stick make sure Linux kernel has support for "802.11 n" standard (I am not sure about it) or some other way to integrate it. To be on safe side we chose a WLAN stick with "802.11 g" support.

After a little bit of Googling and looking on online stores we finally found a "802.11 g" WLAN-USB stick. It is From TP-LINK company 54Mbps Model number TL-WN321G. The driver support is built in Linux Kernel as well as the new driver is available from TP-LINK website for Linux and Windows may be also for MacOs.

3.2.2 Kernel Configuration

To get a WLAN device running we must alter the default Linux Kernel configuration for Imote2. Assuming that you already know how to make changes in Linux Kernel configuration (if you don't know how to, see section 1.1.1 Compiling Linux Kernel for more information), start Linux kernel configuration. Now make following packages either built-in or compile as modules.

Networking support→wireless→Improved wireless configuration API (M)

→ nl80211 new netlink interface support [*]

→Wireless extensions sysfs files [*]

→Common routines for IEEE802.11 drivers (M)

→Generic IEEE 802.11 Networking (M)

→Enable LED triggers [*]

Device Drivers→Network device Support→Wireless LAN→Wireless LAN (IEEE 802.11) [*] .

Here you can also select the device drivers if your card is already supported by Linux Kernel. Packages marked with (M) are optional and provide extra functionality they can be used but they are not necessary. These packages are very useful if you want to make use of "/etc/network/interfaces" script and "ifup" & "ifdown" commands for WLAN connection.

Now save and exit the Kernel configuration, compile the kernel and load the Image and modules to Imote2 (see Chapter 1 for compilation and installation instructions).

3.2.3 Cross Compiling the Driver

Now if the driver is not already a part of Linux Kernel than we can compile it from source code. Here we are describing the procedure for the particular stick that we used (TP-LINK 54Mbps TL-WN321G) but the method should be somewhat similar for other sticks also, for more information read the "README" file provided with driver.

WLAN stick that we are using (TP-LINK 54Mbps TL-WN321G) is based on "Ralink" chipset, driver for this chipset is already present in Linux Kernel but we found it easy to compile and load the driver separately. Now download the Linux version of device driver from TP-Link website and follow these steps.

1. Unpack the archive.

   #tar xvf TpLink_TL_WN321G_in_Linux.tar
   #cd TpLink_TL_WN321G_in_Linux/Module/

2. Edit Makefile and make following changes to it (you can use "vi" or any other editor).
   a. Uncomment the "PLATFORM=EMBEDDED" line.
   b. Comment the "PLATFORM=PC" and "PLATFORM=CMPC" lines.
   c. Set the path of you compiled Kernel source (Please not to compile a module you need a compiled Linux Kernel source), find following lines in Makefile and set "LINUX_SRC" variable to your Compiled Linux Kernel path.

   ifeq ($(PLATFORM),EMBEDDED)
   LINUX_SRC = Path/to/your/Compiled/Linux/Kernel
   Endif

3. Export the system symbols.

   # export ARCH=arm
   #export CROSS_COMPILE=arm-linux-

4. Start the build process

   #make all

5. Now copy the newly created "rt73.ko" file to your Imote2

   #scp rt73.ko root@<IpAddressOfYourImote2>:<Target/directory/on/Imote2>/rt73.ko

That's it you are done with compiling.

3.2.4 Wlan Configuration on Imote2

Once you made changes to Kernel, compiled the driver, loaded new Kernel to Imote2 and copied the driver to your Imote2 you are ready to install and configure the WLAN driver on Imote2.

3.2.4.1 Driver installation

Now insert the WLAN-USB stick into USB-Host port of Imote2 and turn on your Imote2. Now login to Imote2 and follow these steps.

Install the driver module (I assume the module is in "/root/drivers/" directory).

```
#cd /root/drivers/
#insmod rt73.ko
```

Check if the Wlan card is detected and registered as a network interface.

```
#ls /sys/class/net
```

This command will show all the registered network adapters, you should see a directory with name "rausb0" (note the name "rausbX" is only because of this particular driver this name can be different if you chose another card and another driver).

If no Wlan device is found in "/sys/class/net" directory then check "/sys/bus/usb/devices/" directory. If you have a USB hub in between Imote2 and your WLAN-USB stick you should not experience such a problem. If you connect the WLAN-USB stick directly to Imote2 then there may be an issue of power management. This can be solved by overwriting a configuration file in "/sys/bus/usb/devices/1-1/" directory.

```
# echo –n 1 > /sys/bus/usb/devices/1-1/bConfigurationValue
```

Above command is a hack to the problem and should fix the problem.

3.2.4.2 Network Parameters Configuration

Once the driver is successfully installed rest of the process is standard for any Linux system. Issue following commands (replace the <> brackets and containing text with appropriate network parameters).

```
#ifconfig rausb0 up
#iwconfig rausb0 key <type your wlan key here>
#iwconfig rausb0 essid <NetworkName/AccesPointName>
#iwconfig rausb0 nickname <anyNickNameForYourDevice>
#ifconfig rausb0 inet <TypeYourIpAddressHere>
#ifconfig rausb0 netmask<TypeYourNetMaskHere>
#ifconfig rausb0 gateway <TypeYourDefaultGatewayHere>
#ifconfig rausb0 network <TypeYourNetworkAddressHere>
#ifconfig rausb0 broadcast <TypeYourBroadcastAddressHere>
```

Last two parameters are optional and only required if these values are non default for your network. By default network address is XXX.XXX.XXX.0 and broadcast address is XXX.XXX.XXX.255 where XXX.XXX.XXX represents your network's prefix. One can atomize the whole process described in section 3.2.4.1 and section 3.2.4.2 by writing a small script and putting it in "/etc/rc2.d/" directory. Given below is the script, just copy these lines to a text file, name it "S50StartupScript" and put it in "/etc/rc2.d/" directory.

```
#*******************************************************
#***********S50StartupScript File**************************
#***********This file configures Wlan on Imote2***************

#Install driver
cd /root/drivers/
insmod rt73.ko
echo –n 1 > /sys/bus/usb/devices/1-1/bConfigurationValue

#Configure Wlan
ifconfig rausb0 up
iwconfig rausb0 key <type your wlan key here>
iwconfig rausb0 essid <NetworkName/AccesPointName>
iwconfig rausb0 nickname <anyNickNameForYourDevice>
ifconfig rausb0 inet <TypeYourIpAddressHere>
ifconfig rausb0 netmask<TypeYourNetMaskHere>
ifconfig rausb0 gateway <TypeYourDefaultGatewayHere>
ifconfig rausb0 network <TypeYourNetworkAddressHere>
ifconfig broadcast <TypeYourBroadcastAddressHere>

#***********EOF  S50StartupScript File**********************
```

Please note in above lines of code a # (hash) sign means a comment it is doesn't represent a root user, while in section 3.2.4.1, section 3.2.4.2 and in command below this paragraph a # (hash) sign represents a root user. In order to make this method work you also need to change the permission of the file, following command will do the job.

```
#chmod 777 S50StartupScript
```

After this, every time you restart the Imote2 your WLAN adapter will be automatically configured and running.

3.2.5 Conclusion

The method described in this chapter is tasted and it worked but some time it takes time to make things working. It may take several tries to make it working. Method described in this section considers a particular WLAN-USB stick (TP-LINK 54Mbps TL-WN321G) but can be applied with little modification to other similar hardware.

3.3 USB-2-Ethernet Adapter Interfacing with Imote2

It's interesting to have Ethernet enabled on Imote2 one may want to connect it to Lan directly, rather than using USB-Ethernet-gadget. So the method is very similar to that described in section 3.2 WLAN-USB Stick Interfacing With Imote2. Only thing you have to do is to find a USB-2-Ethernet adapter for which the driver support is either already there in Linux kernel or the driver is available from the vendor for compilation and installation. So we found an adapter whose driver was already there in Linux kernel, this adapter is based on "MosChip MCS7830 "chipset.

3.3.1 Kernel Configuration

For kernel configuration you should have USB-Host support enabled (see section 3.1 How to Enable USB-Host Support) and the TCPIP support enabled in kernel. Now make following packages either built-in or compile as modules to enable TCP/IP stack and the device driver for a particular network card.

Networking support→Network options→Packet socket <*>

→Unix domain sockets <*>

→TCP/IP networking [*]

→IP: multicasting [*]

→IP: kernel level autoconfiguration [*]

→INET: socket monitoring interface <*>

Device Drivers→Network device support→USB Network Adapters→Multi-purpose Networking Framework <*>

Device Drivers→Network device support→USB Network Adapters→Embedded ARM Linux links (ipaq, ….) [*]

And here in this section you can select driver for your card, in our case the driver for our card was already present in kernel so we enabled it.

Device Drivers→Network device support→USB Network Adapters→MosChip MCS7830 based Ethernet adapters <*>

Now save and exit the Kernel configuration, compile the kernel and load the Image and modules to Imote2 (see Chapter 1 for compilation and installation instructions).

3.3.2 Configuring the Network Parameters

After copying the new kernel and new modules reboot the Imote2 and logon to Imote2. Those packages which were compile as loadable modules should be installed the easiest way is to edit the file "/etc/modules" and reboot the system again. On reboot issue "lsmod" command to make sure that loadable modules are properly loaded. Now insert the USB-2-

Ethernet adapter into USB-Host port of Imote2 and reboot your Imote2 third time. Now login to Imote2 and follow these steps.

Install the driver module (I assume the module is in "/root/drivers/" directory), (skip this step if the driver is already there in kernel).

#cd /root/drivers/
#insmod <name of driver>.ko

Check if the network card is detected and registered as a network interface.

#ls /sys/class/net

This command will show all the registered network adapters, you should see a directory with name "eth0" (note the name "ethX" is only because of this particular driver this name can be different if you chose another card and another driver).

If no USB-2-Ethernet adapter device is found in "/sys/class/net" directory then check "/sys/bus/usb/devices/" directory. If you have a USB hub in between Imote2 and your network card you should not experience such a problem. If you connect the USB-2-Ethernet adapter directly to Imote2 then there may be an issue of power management. This can be solved by overwriting a configuration file in "/sys/bus/usb/devices/1-1/" directory.

 # echo –n 1 > /sys/bus/usb/devices/1-1/bConfigurationValue

Above command is a hack to the problem and should fix the problem.

Once the driver is successfully installed rest of the process is standard for any Linux system. Issue following commands (replace the <> brackets and containing text with appropriate network parameters).

#ifconfig eth0 up
#ifconfig ethb0 inet <TypeYourIpAddressHere>
#ifconfig eth0 netmask<TypeYourNetMaskHere>
#ifconfig eth0 gateway <TypeYourDefaultGatewayHere>
#ifconfig eth0 network <TypeYourNetworkAddressHere>
#ifconfig eth0 broadcast <TypeYourBroadcastAddressHere>

Last two parameters are optional and only required if these values are non default for you network. By default network address is XXX.XXX.XXX.0 and broadcast address is XXX.XXX.XXX.255 where XXX.XXX.XXX represents your network's prefix. One can atomize the whole process described above by editing "/etc/network/interfaces" file. Copy following lines, without touching already existing text, to the empty section of "interfaces" script.

auto eth0
iface eth0 inet static
address < TypeYourIpAddressHere >
netmask <TypeYourNetMaskHer>

network <TypeYourNetworkAddressHere>
broadcast <TypeYourBroadcastAddressHere>
gateway <TypeYourDefaultGatewayHere>

Now reboot your Imote2, you can also try following commands to turn on or off the network interface.

#ifup eth0
#ifdown eth0

After this, every time you restart the Imote2, your network adapter will be automatically configured and running. You can use following command to list all the configured network interfaces as well as their details.

#ifconfig

3.4 UMTS/GSM Surf-Stick Interfacing With Imote2

Heaving an internet connection and using it for remote data logging is a nice concept for Imote2. This task can be achieved by using a UMTS/GSM internet surf-stick which is easily available in market from different vendors and with different data tariffs. In this section we will describe how to interface such hardware with Imote2. We used Huawei E160 UMTS/GSM Surf-stick and MEDIONmobile tariff. Although our work is related to mentioned hardware but most of it is also applicable to other similar hardware, because the kernel modules and user space applications used will remain same, only things that differ are hardware drivers and details from connection provider. To get Surf-Stick running we need three things.

- Mode switching of Surf-stick (see section 3.4.1)
- New kernel modules (see section 3.4.2)
- User space application (see section 3.4.3)

3.4.1 UMTS/GSM Surf-Stick

A UMTS/GSM Surf-stick is a device that uses cell phone networks to provide internet connectivity wirelessly. These Surf-sticks look like USB Flash-drive and act quite similarly, when first time inserted into a windows machine these devices are mostly recognized as USB-mass-storage device. They appear either as a CD-ROM or as a USB-flash-drive; from there these devices launch the driver setups. Once the driver is installed it changes the Surf-stick from USB-mass-storage mode to USB-modem mode via a virtual serial port. After that, appear some extra serial ports on Windows system and there also appears a modem device in the hardware list. All this process goes seamlessly and very quick on a Windows machine but things are not as simple on a Linux machine. It is because the drivers provided with the hardware mostly don't support Linux.

There exist other methods in Linux environment to handle this problem, so there are two ways to do this first method is to install a user space demon called "USB-mod-switch". This program switches Surf-sticks from USB-mass-storage mode to USB-serial mode. These days

this program comes as a part of many Linux distributions. To use this demon on Imote2 one must cross compile it, after giving a lot of try to this method we quit this method. The second method relies on capabilities of Surf-sticks to be permanently switched to USB-serial mode and USB-modem mode. For this method to work you first need a Windows or a Linux system and then follow the instructions given below.

3.4.1.1 Instruction for Windows System

1. Insert your UMTS/GSM Surf-stick into USB port of your system and wait a while so Windows can recognize the hardware and install the proper drivers.
2. Now if the Auto-Play is enabled on your system the device will automatically install Modem driver, otherwise go to MyComputer there you will find a new CD-Rom device or a new USB-mass-storage device, open it and install the driver by yourself.
3. After the drivers are installed successfully go to 'Control-Panel→Telephone-&-modem-option', double click and open it.
4. In the newly opened window select 'Modems' tab, in this tab you will see the list of available modems, select the one which belongs to your Surf-stick and then click on 'Properties' button below.
5. In the newly opened properties window select 'Modem' tab, now note down the 'Com Port' number and the 'baudrate' of the modem, you will need this information in next step.
6. Open the Hyper-Terminal or any other similar program and open a connection on a serial line with following settings "COM=XX, Bits per second/baudrate=YYYY, data bits=8, parity=none, stop bit=1, flow control=none" where XX represent your modem serial port and YYYY represent the baudrate of your modem.
7. Now in the terminal widow type 'at' (without comas) and press 'enter' (you may not see what you are typing), if everything is fine till this step you will see 'OK' on your terminal screen, this means that you are connected to your modem and modem is ready to accept further commands.
8. Now issue command 'AT^u2diag=0' (without comas) and press 'enter' (once again you may not see what you are typing), in response you will see 'OK' and that is it, process completed. Note that it is better that you write this command in notepad and then copy paste in your terminal as you don't see what are you typing.

Now your surf-stick is switched permanently into modem mode and you will not need extra programs on Linux system to make your hardware work as modem. These settings can be reversed by issuing a command 'AT^u2diag=1' (without comas) and press 'enter' (once again you may not see what you are typing), you will see 'OK' as a confirmation and your device will be working as it was working before.

3.4.1.2 Instructions for Linux System

In a Linux system if the USB-mode-switch is installed your device will be automatically switched to USB-Serial mode. The device appears as USB-Serial port with name 'ttyUSBXX' where XX is the number of port. You can also find out if the device is really detected or not

by typing command 'ls /dev/ttyUSB*' at your Linux command prompt and it will list all the detected devices. The device will expose two USB-Serial ports if it is not permanently switched to modem mode and it will expose three USB-Serial ports if it is permanently switched to modem mode. So now assuming that USB-mode-switch is already installed on your system, device already plugged-in and recognized as two USB-Serial ports follow steps given below.

1. Open Serial Terminal, you can use 'screen', 'stty' or any other similar program, open a connection on a serial line with following settings "ttyUSBXX, Bits per second/baudrate=YYYY, data bits=8, parity=none, stop bit=1, flow control=none" where XX represent your modem serial port, (usually the first one of two exposed USB-Serial ports is the one we use so ttyUSB1 should work fine) and YYYY represent the baudrate of your modem. Note in case of Linux system you don't have any way to find the baudrate of your modem so you should have this information from your vendor.
2. Now in the terminal widow type 'at' (without comas) and press 'enter' (you may not see what you are typing), if everything is fine till this step you will see 'OK' on your terminal screen, this means that you are connected to your modem and modem is ready to accept further commands.
3. Now issue command 'AT^u2diag=0' (without comas) and press 'enter' (once again you may not see what you are typing), in response you will see 'OK' and that is it, process completed. Note that it is better that you write this command in a text editor and then copy paste in your terminal as you don't see what are you typing.

Now your surf-stick is switched permanently into modem mode and you will not need extra programs on Linux system to make your hardware work as modem. These settings can be reversed by issuing a command 'AT^u2diag=1' (without comas) and press 'enter' (once again you may not see what you are typing), you will see 'OK' as a confirmation and your device will be working as it was working before.

3.4.2 Kernel Configuration

To get a Surf-stick running you must have Linux Kernel with USB-Host support enabled (see section 3.1 How to Enable USB-Host Support), TCPIP Stack (see section 3.3 for how to enable TCP/IP Stack) and PPP (point to point protocol) support enabled. Now we alter the default Linux Kernel configuration for Imote2 to enable PPP support and extra packages. Assuming that you already know how to make changes in Linux Kernel configuration (if you don't know how to, see section 1.1.1 Compiling Linux Kernel for more information), start Linux kernel configuration. Now make following packages either built-in or compile as modules.

Device Drivers→Network Device Support→PPP support

Device Drivers→Network Device Support→PPP support→PPP support for async serial ports

Device Drivers→Network Device Support→PPP support→PPP support for sync tty ports

Device Drivers→Network Device Support→PPP support→PPP Deflate compression

Device Drivers→Network Device Support→PPP support→PPP BSD-Compression

Device Drivers→USB support→USB Modem (CDC ACM) support

Device Drivers→USB support→USB serial converter support

Device Drivers→USB support→USB serial converter support→USB Generic serial driver

Device Drivers→USB support→USB serial converter support→USB driver for GSM and CDMA modems

Now save and exit the Kernel configuration, compile the kernel and load the Kernel-Image and modules to Imote2 (see Chapter 1 for compilation and installation instructions), add the name of new modules to "/etc/modules" file of your Imote2 so that these modules can be loaded at start-up. Now reboot your Imote2 and type "lsmod" on the Imote2 command prompt to make sure that modules are loaded properly.

Now when kernel is properly configured, installed and modules are loaded, it is the time to check if things are working or not. Insert your Surf-stick into USB-Host port of your Imote2 and type "ls /dev/ttyUSB*" command on your Imote2 command prompt. You should see at least two USB-serial devices or may be three if Surf-stick is permanently switched to USB-modem mode. If no USB-serial device is found in "/dev/" directory then check "/sys/bus/usb/devices/" directory. If you have a USB hub in between Imote2 and your Surf-stick you should not experience such a problem. If you connect the Surf-stick directly to Imote2 then there may be an issue of power management. This can be solved by overwriting a configuration file in "/sys/bus/usb/devices/1-1/" directory.

 # echo –n 1 > /sys/bus/usb/devices/1-1/bConfigurationValue

Above command is a hack to the problem and should fix the problem.

3.4.3 User Space Application

After verifying that our Surf-stick is properly interfaced with Imote2 now we are ready to move further. To connect to internet last thing we need is a user space program which manages connection of our Surf-stick such a program is called point to point protocol ppp. PPP can be used alone or in conjunction with a dialler program for example Wvdial. Both the approaches have been tested on a Debian GNU Linux system and found working, so we assume it will also work on Imote2.

3.4.3.1 Cross-Compile ppp for Imote2

To use the program "ppp" on Imote2 we have to download the source code of the program and then we have to cross-compile it, before we load it to Imote2. Follow the steps given below to cross compile ppp for Imote2.

1. Download the "ppp" source code from "http://samba.org/ppp/download.html" or from any other website, just make sure you get the latest one, we used version 2.4.5.
2. Unpack the archive go to the source directory.

#tar xvzf ppp-2.4.5.tar.gz
#cd ppp-2.4.5

3. First of all read the README file for a better understanding.
4. Run configure script to generate a Makefile appropriate for ARM-Linux.

#./configure --target=arm-linux --enable-arm --prefix= <path to your compiled kernel source>

5. Cross compile the code, (make sure your cross compiler is installed and its path is included to system PATH variable. You can check this by entering "arm-linux-gcc -v" on your Linux system command prompt, it will show you the version of your compiler).

#make CC=arm-linux-gcc

6. Install "ppp" to a temporary folder (You should have root privilege to do this), this temporary directory must be created in system directory "/usr/local/".
    a) Create a temporary folder.

#mkdir /usr/local/ppp_imote2_install_dir

    b) Install the files to created folder.

#make INSTROOT=/usr/local/ppp_imote2_install_dir
DESTDIR=/usr/local/ppp_imote2_install_dir/usr/local install

Yes two times "/usr/local" in above command it's not a mistake.

    c) Create the configuration folders.

#cd /usr/local/ppp_imote2_install_dir/
#mkdir -p etc/ppp/peers
#mkdir -p var/lock

    d) Create the configuration files (rather than using method below you can also use a text editor).
        I. Create chap secrets file.

#cat > chap-secrets
#Crtl+d (press key combination control + d)

        II. Create pap secrets file.

#cat > pap-secrets
#Crtl+d (press key combination control + d)

III. Create options file.

```
#cat > options
lock
#Crtl+d          (press key combination control + d)
```

IV. Create chat script file. In lines below replace the word <internet> with proper internet address of your ISP (just replace <internet> let the inverted commas stay there); you can get this information by calling your connection provider.

```
#cat > chat
SAY "START\n"
ABORT BUSY
SAY "ABORT\n"
ECHO OFF
'' ATZ
'' AT+CGDCONT=1,"ip","<internet>"'
#SAY "DIAL"
OK ATD*99#
SAY 'CONNECT'
#Crtl+d          (press key combination control + d)
```

V. Create a file for your ISP-details you can name it as you like, better name it after the name of your ISP to easily distinguish. In lines below replace <type your username> and <type your password here> with your username and password; you can get this information by calling your connection provider.

```
#cd peers
#cat > my_Isp
/dev/ttyUSB0
460800
idle   7200
lock
crtscts
modem
noauth
defaultroute
user          <type your username>
password      <type your password here>
connect "/usr/local/sbin/chat -V -f /etc/ppp/chat"
noipdefault
usepeerdns
novj
#Crtl+d          (press key combination control + d)
```

VI. Create ppp-off script.

#cat > ppp-off
kill `cat /var/run/ppp0.pid`
#Crtl+d (press key combination control + d)
#chmod 777 ppp-off

In above commands (i-e step 6 sub-step d and all further sub-steps I-VI) text after "cat >" command is the file name, all following lines will be written to file and key combination "Ctrl+d" ends the file editing and returns the control back to Linux.

7. Create an archive of the installed files and copy to root folder.

#cd /usr/local/ppp_imote2_install_dir/
#tar cvf ppp_imote2_installed.tar etc/ var/ usr/
#cp ppp_imote2_installed.tar /root/

8. Delete the temporary installation folder.

#cd
#rm -r /usr/local/ppp_imote2_install_dir

Till this point we have downloaded and cross compiled "ppp" as well as installed it in a temporary folder on host system, created an archive of the installed files and copied the archive to "/root/" directory. Remember that steps mentioned so far in the section 3.4.3.1 are all carried on the host system not on Imote2.

3.4.3.2 Installing ppp on Imote2

Installation process is very simple you just need to copy the files in right place, follow the steps given below.

1. Connect your Imote2 to your host system and turn it on.
2. Copy the archive, created in section 3.4.3.1, from host system to your Imote2. Type these commands on your host system.

#cd
#scp ppp_imote2_installed.tar root@<ip address of your Imote2>:ppp_imote2_installed.tar

3. Now log on to your Imote2 console and type following commands on your Imote2 command prompt.

#cd/
#tar xvf /root/ppp_imote2_installed.tar

Congratulations you have successfully installed the ppp on your Imote2.

3.4.3.3 Running and Debugging ppp on Imote2

After compiling and installing the kernel, modules and ppp program, we are ready to connect to internet for first time.

To connect to internet use this command.

#pppd call <isp_filename>

In above command replace <isp_filename> with same name that you have given to the configuration file in "/etc/ppp/peers" directory, generally it is your ISP name.

To stop the internet connection use command.

#/etc/ppp/ppp-off

To check if you are connected to your ISP or not check your network interfaces.

#ifconfig

After issuing above command you should see a network interface named "ppp0" among your other network interfaces. If you don't see it then check inside "/sys/class/net/" directory. Usually it takes a bit of time to get connected to internet. In case you have problems connecting to ISP you can use debug option to rectify the problem .

#cd
#pppd call <isp_filename> debug logfile <logFileName>

You can give any file name for <logFileName>, the file will be automatically created and place in your current directory. Once you are connected to ISP now we configure our DNS server. After successful connection to ISP you will see a file with name "resolv.conf" in "/etc/ppp" directory of your Imote2. You have to copy this file to the "/etc/" directory and edit this file.

#cd /etc/ppp
#cp resolv.conf /etc/resolv.conf
#cd /etc/

Now edit the file "/etc/resolv.conf" and add following lines on top of it leaving rest of the text untouched, you have to do this step just once when you connect to internet; you have to do this step again if you connect to a different ISP.

domain localdomain
search localdomain

One last thing you have to do before you start using the internet is you should add ppp0 to your default route.

#route add default gw <ip address of your ppp0 network interface> ppp0

You can get the IP address of your ppp0 network interface from "ifconfig" command, you have to do this step each time you connect to internet because every time you connect to internet your IP address changes. To check if everything is working fine use ping command.

#ping www.yahoo.com -c 3

If things don't work at first time try it multiple times by restarting your Imote2. The method has been successfully tested on two Imote2 systems. Sometime even everything is in place but still the internet doesn't work, but after trying for two three times it works.

3.4.4 Known Problems

Following problems are known and can be solved.

1. Surf-stick is not detected as ttyUSB device in "/dev/" directory of Imote2.

   It can be due to uninstalled drivers, make sure that all the drivers have been compiled and loaded to imote2 as well installed on start-up, see file "/etc/modules" on your Imote2 and run "lsmod" command to find out which modules are installed.

   This problem can also occur if you connected the Surf-stick directly to Imote2 then there may be an issue of power management. This can be solved by overwriting a configuration file in "/sys/bus/usb/devices/1-1/" directory.

   # echo –n 1 > /sys/bus/usb/devices/1-1/bConfigurationValue

   Above command is a hack to the problem and should fix the problem.

2. Sim-card Pin code issue.
   In order to avoid complexity in ppp scripts we recommend that you disable the pin code request of you Sim-card. You can do this by pulling out your Sim-card from your surf-stick and putting it into a mobile phone, in mobile phone you can disable the pin code request and use it again in your surf-stick.
3. You see ppp0 in your network interfaces but your internet is not working.
   This happens when either you forget to add the default route to ppp0 or you forget to configure your "/etc/resolv.conf" file. This problem can occur even everything is fine in this case just reboot the system and try again.
4. You see the ttyUSB devices in "/dev/" but after running "pppd call <ISP_filename>" command you don't see ppp0 among your network interfaces.
   This can be due to a connection problem, bad signal or a delayed authentication. First of all wait a while and check again, if still you don't get it then run pppd command with debug options.

   #pppd call <ISP_filename> debug logfile <logFileName>

   In the log file you will find if there are any connection issues. Try again after resetting the system.

Acknowledgement

This research was supported by the German Research Foundation (DFG) as part of the Collaborative Research Centre 637 “Autonomous Cooperating Logistic Processes”. Further project information can be found at www.intelligentcontainer.com