Complete ARM Solutions
Design, Development and Manufacturing
Expertise on Embedded Linux, Android, WindowsCE

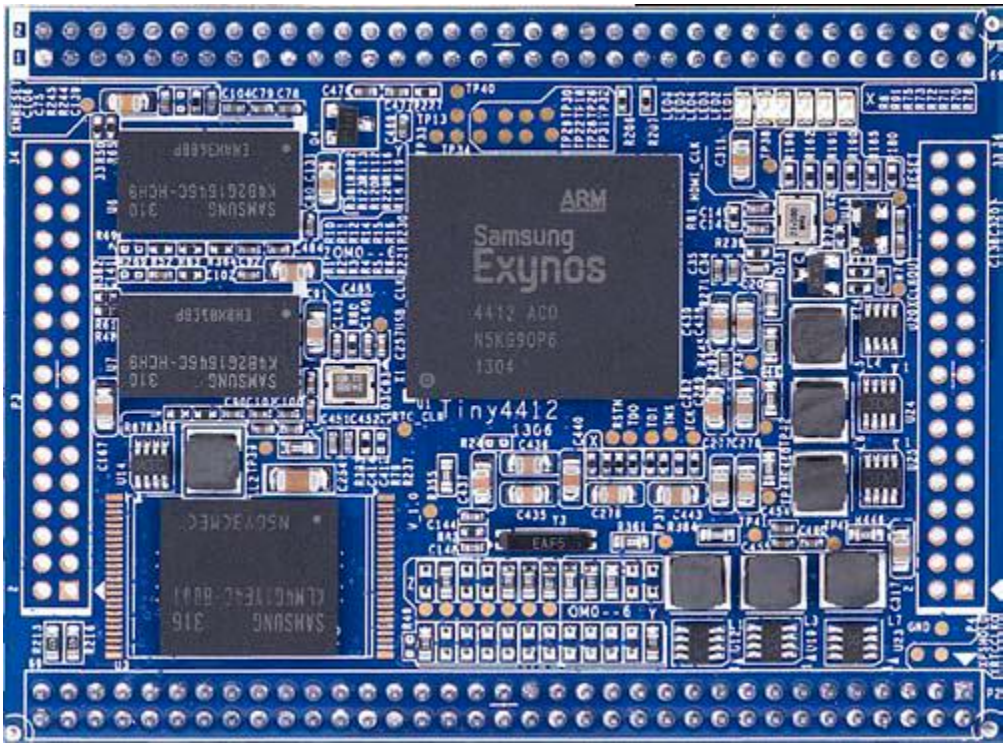
Tiny4412 User's Manual

REVISION	ORIGINATOR	SCR	REV DATE
0.1.0	Guangzhou FriendlyARM Co., Ltd		August 1st, 2013
Guangzhou FriendlyARM Computer Tech Co., Ltd Confidential: This document and information contained in it shall not be reproduced by, used by, or disclosed to others except as expressly authorized in writing by Guangzhou FriendlyARM Co., Ltd.			**Guangzhou FriendlyARM Computer Tech Co., Ltd Guangzhou, China**

COPYRIGHT STATEMENT

Index

1 Introduction



Tiny4412 CPU Board

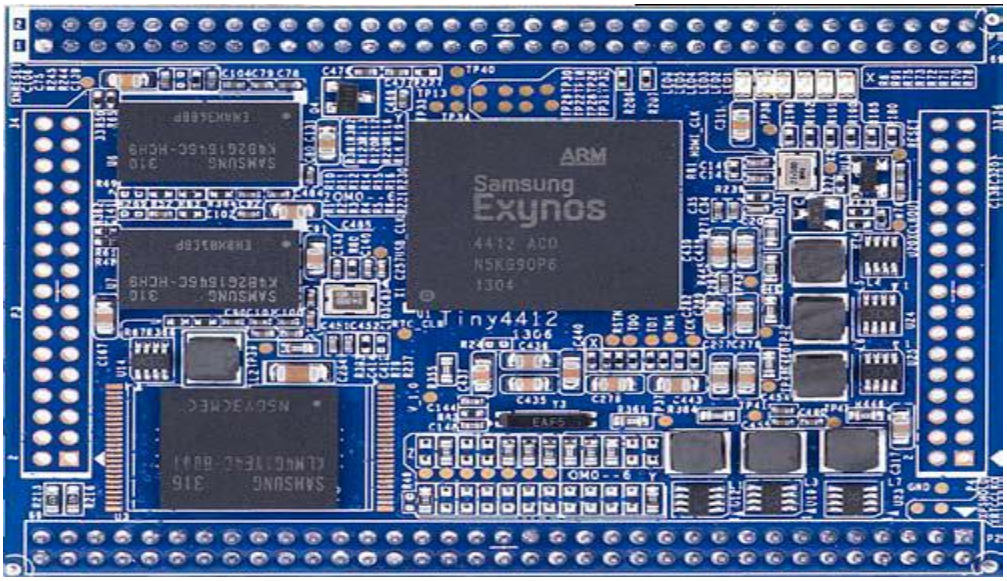
The Tiny4412 development board is a Cortex-A9 embedded processing board that uses the Samsung Exynos4412 Quad-Core System On Chip (SOC).

The Exynos4412 integrates the Mali-400 MP GPU graphic engine with hardware support for 3D and can drive video playing on screens up to 1080P. These features make it easily and widely used in MID development, Android notepads, auto electronic devices, industrial applications, GPS systems and multimedia systems.

1.1 Tiny4412 CPU Board

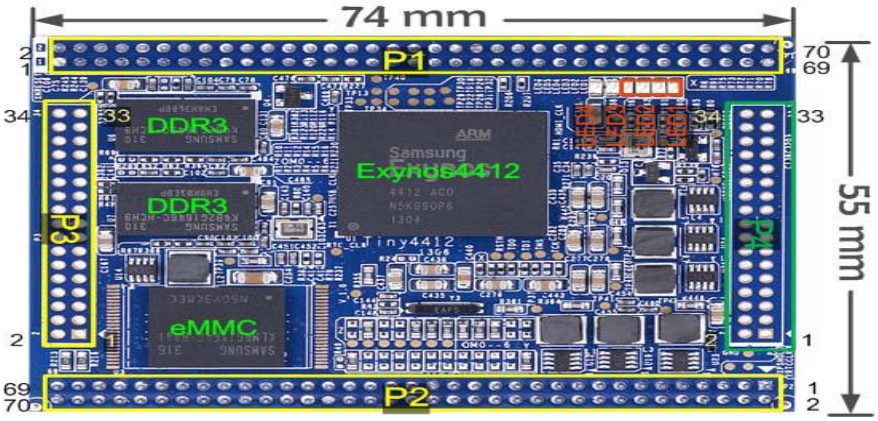
The Tiny4412 has four 2.0 mm spacing double row pitch headers (P1, P2, P3 and P4) among which P1 and P2 connect the Tiny4412 CPU board to a carrier board and extend most of the CPU's pins. Its standard version integrates 1G DDR3 RAM and 8G eMMC flash memory.

1.1.1 Tiny4412 CPU Board Hardware Feature

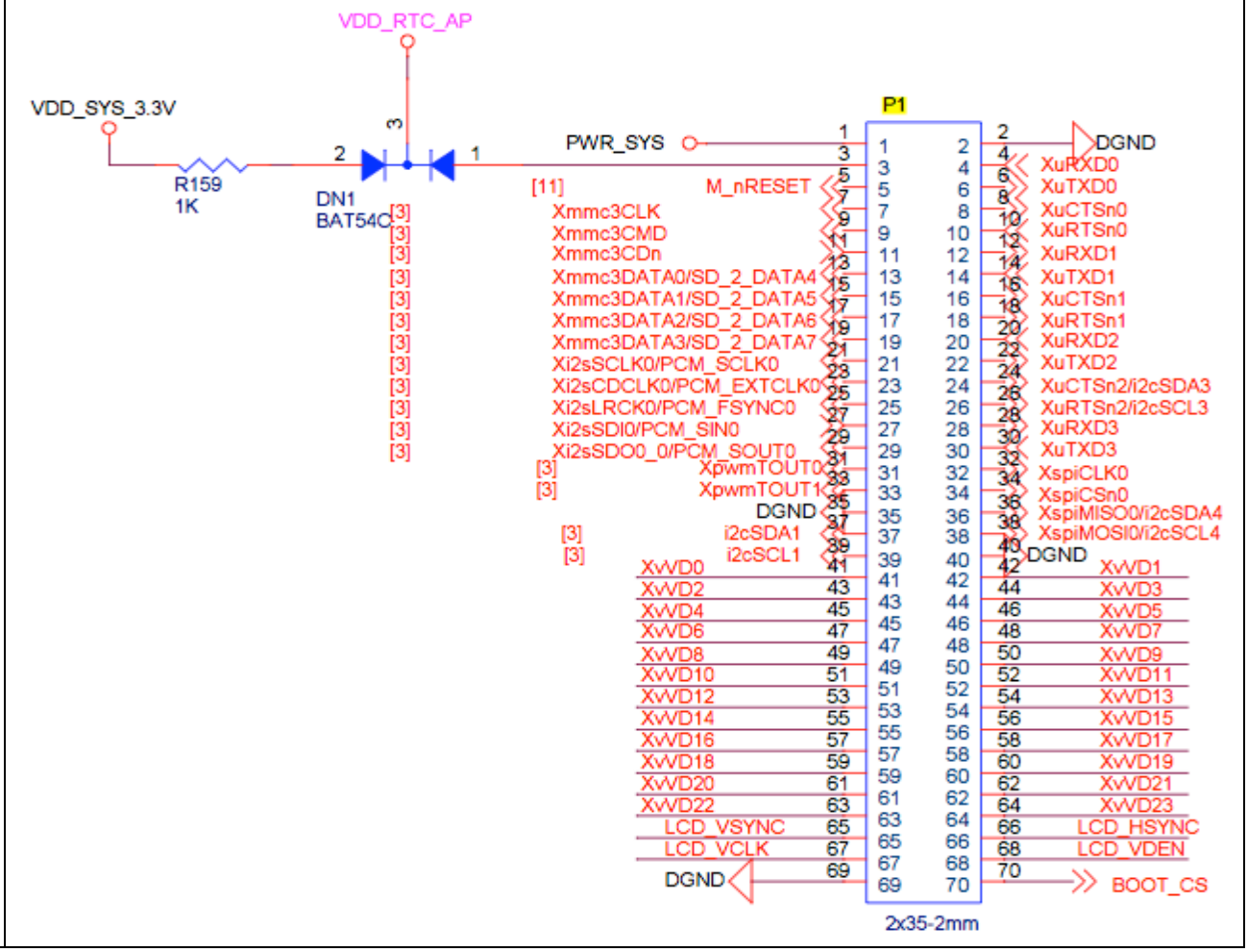


CPU	● Samsung Quad-Core Exynos 4412, based on Cortex-A9, 1.5GHz ● Integrated ARM Mali-400 Dual-Core GPU ● Elegent 2D/3D graphic accelaration ● Up to 1080p@30fps hard decoded video playing, support MPEG4, H.263, H.264 etc ● Up to 1080p@30fps hard decoded (Mpeg-2/VC1) video input
DDR3 RAM	● 1G ● 32bit data bus, single channels ● 400 M Hz
FLASH	● eMMC Flash: 8G, optional 4GB/16GB/32GB
Connector	● 2 x 70 pin 2.0 mm pitch header ● 2 x 34 pin 2.0 mm pitch header
On Board Hardware Resource	● 4 x LED (Green) ● 2 x Power LED (Green)
Power	● 3.7V to 6V (support sleep mode)
PCB Dimension	● Eight layered board ● Dimension: 74 x 55 x 10(mm)

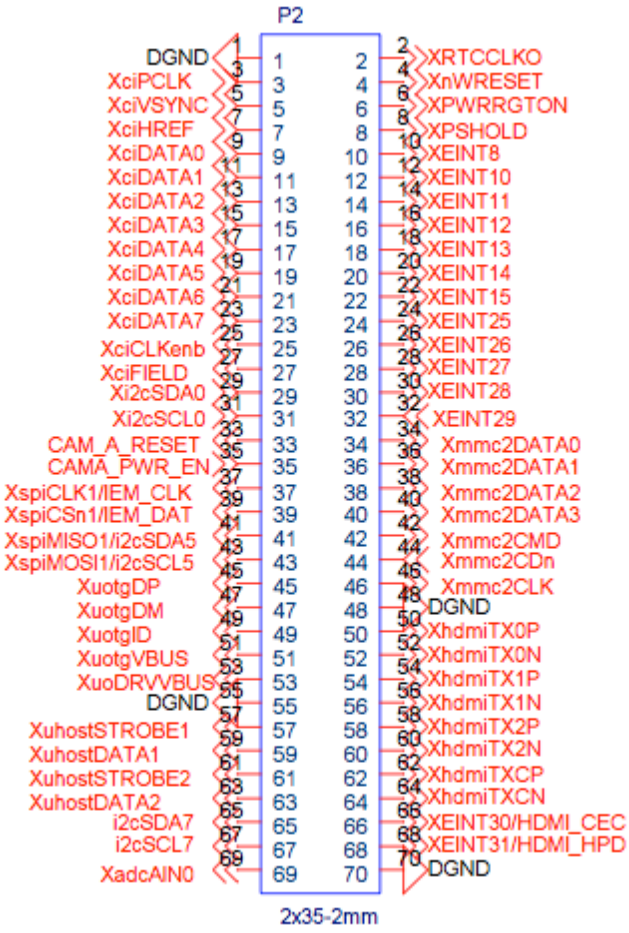
1.1.2 Pin Spec



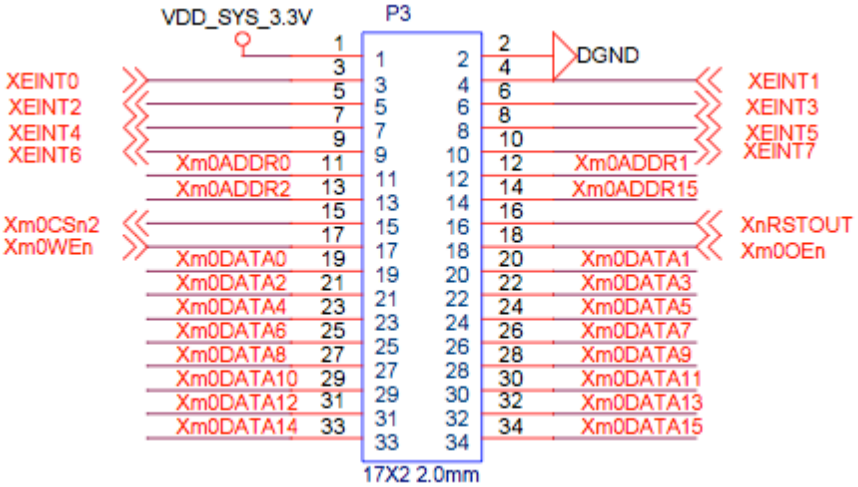
P1

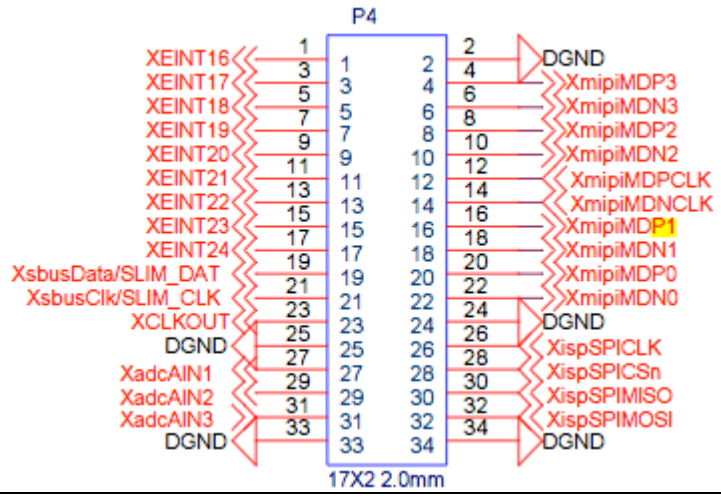

VDD_RTC_AP
VDD_SYS_3.3V
R159
1K
DN1
BAT54C
PWR_SYS
[11] M_nRESET
Xmmc3CLK
Xmmc3CMD
Xmmc3CDn
Xmmc3DATA0/SD_2_DATA4
Xmmc3DATA1/SD_2_DATA5
Xmmc3DATA2/SD_2_DATA6
Xmmc3DATA3/SD_2_DATA7
Xi2sSCLK0/PCM_SCLK0
Xi2sCDCLK0/PCM_EXTCLK0
Xi2sLRCK0/PCM_FSYNC0
Xi2sSDI0/PCM_SIN0
Xi2sSDO0_0/PCM_SOUT0
XpwmTOUT0
XpwmTOUT1
DGND
i2cSDA1
i2cSCL1
XvVD0
XvVD2
XvVD4
XvVD6
XvVD8
XvVD10
XvVD12
XvVD14
XvVD16
XvVD18
XvVD20
XvVD22
LCD_VSYNC
LCD_VCLK
DGND
P1
2x35-2mm
DGND
XuRXD0
XuTXD0
XuCTSn0
XuRTSn0
XuRXD1
XuTXD1
XuCTSn1
XuRTSn1
XuRXD2
XuTXD2
XuCTSn2/i2cSDA3
XuRTSn2/i2cSCL3
XuRXD3
XuTXD3
XspiCLK0
XspiCSn0
XspiMISO0/i2cSDA4
XspiMOSI0/i2cSCL4
DGND
XvVD1
XvVD3
XvVD5
XvVD7
XvVD9
XvVD11
XvVD13
XvVD15
XvVD17
XvVD19
XvVD21
XvVD23
LCD_HSYNC
LCD_VDEN
BOOT_CS

P2


P2
DGND
XciPCLK
XciVSYNC
XciHREF
XciDATA0
XciDATA1
XciDATA2
XciDATA3
XciDATA4
XciDATA5
XciDATA6
XciDATA7
XciCLKenb
XciFIELD
Xi2cSDA0
Xi2cSCL0
CAM_A_RESET
CAMA_PWR_EN
XspiCLK1/IEM_CLK
XspiCSn1/IEM_DAT
XspiMISO1/i2cSDA5
XspiMOSI1/i2cSCL5
XuotgDP
XuotgDM
XuotgID
XuotgVBUS
XuoDRVVBUS
DGND
XuhostSTROBE1
XuhostDATA1
XuhostSTROBE2
XuhostDATA2
i2cSDA7
i2cSCL7
XadcAIN0
XRTCCLKO
XnWRESET
XPWRRGTON
XPSHOLD
XEINT8
XEINT10
XEINT11
XEINT12
XEINT13
XEINT14
XEINT15
XEINT25
XEINT26
XEINT27
XEINT28
XEINT29
Xmmc2DATA0
Xmmc2DATA1
Xmmc2DATA2
Xmmc2DATA3
Xmmc2CMD
Xmmc2CDn
Xmmc2CLK
DGND
XhdmiTX0P
XhdmiTX0N
XhdmiTX1P
XhdmiTX1N
XhdmiTX2P
XhdmiTX2N
XhdmiTXCP
XhdmiTXCN
XEINT30/HDMI_CEC
XEINT31/HDMI_HPD
DGND
2x35-2mm

P3

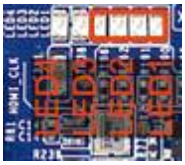

VDD_SYS_3.3V
P3
DGND
XEINT0
XEINT2
XEINT4
XEINT6
XEINT1
XEINT3
XEINT5
XEINT7
Xm0ADDR0
Xm0ADDR2
Xm0ADDR1
Xm0ADDR15
Xm0CSn2
Xm0WEn
XnRSTOUT
Xm0OEn
Xm0DATA0
Xm0DATA2
Xm0DATA4
Xm0DATA6
Xm0DATA8
Xm0DATA10
Xm0DATA12
Xm0DATA14
Xm0DATA1
Xm0DATA3
Xm0DATA5
Xm0DATA7
Xm0DATA9
Xm0DATA11
Xm0DATA13
Xm0DATA15
17X2 2.0mm

P4	

1.1.3 Interface and Port

1.1.3.1 LED

LED is a commonly used status indication device. The Tiny4412 has four programmable LEDs which are directly connected to GPIO and are on at a low level voltage.

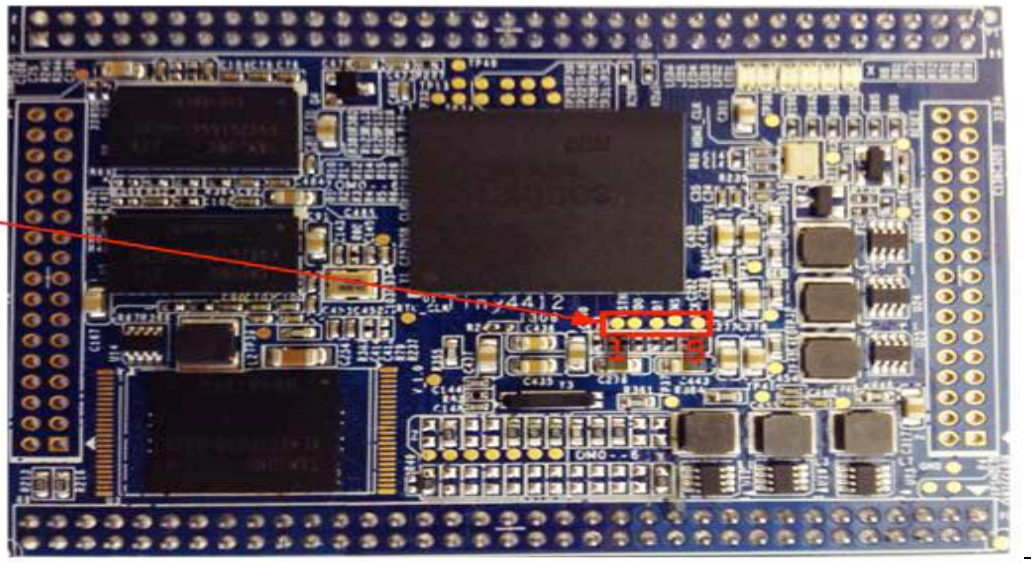


	LED1	LED2	LED3	LED4
GPIO Pins	GPJ_0	GPJ_1	GPJ_2	GPJ_3

1.1.3.2 JTag

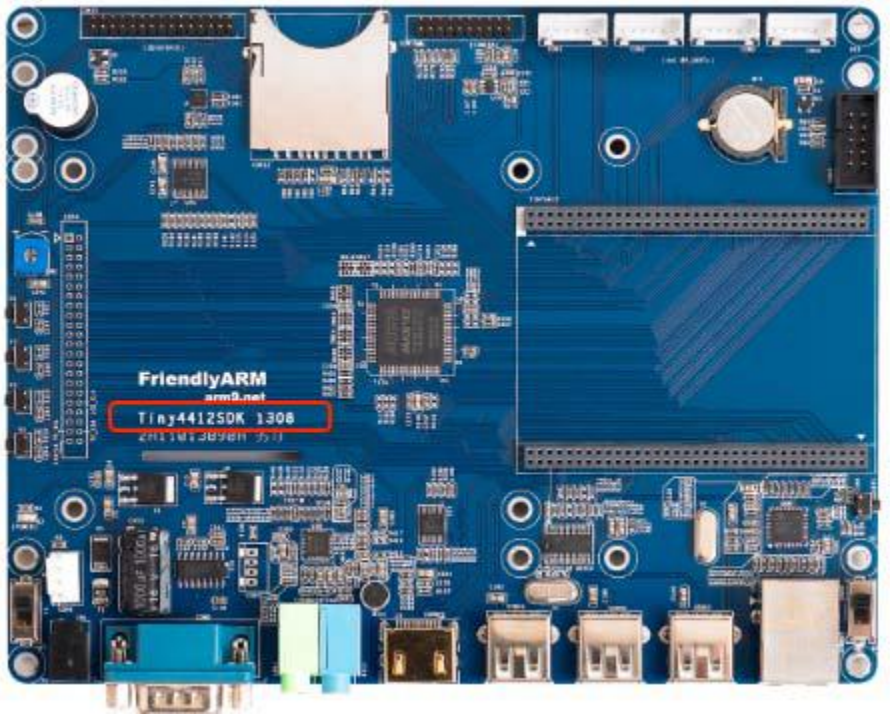
Currently most of high end ARM chips support booting from SD card the JTAG is not significant to users any more. However there are still users who really like JTAG therefore we extend the JTAG's test points for users.

1) XjTDO
2) XjTDi
3) XjTMS
4) XjRStn
5) XjTCK

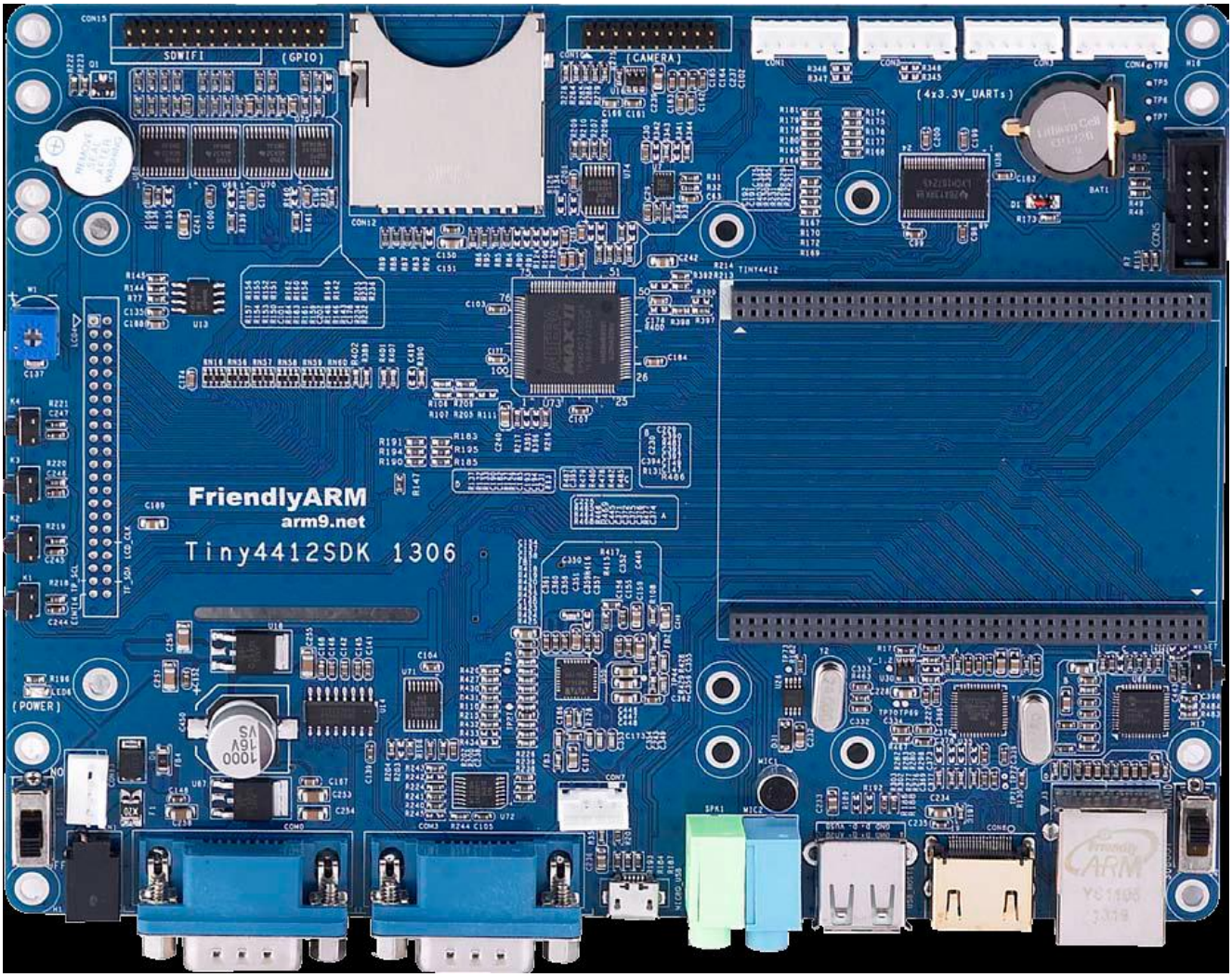


1.2 Tiny4412 Base Board (Extension Board)

The Tiny4412 has two types of base board: standard version and enhanced version. The Tiny4412SDK 1306 is the enhanced version and the 1308 is the standard version. Below is a 1308 board.



1.2.1 Tiny4412SDK Enhanced Version



The Tiny4412 base board is a double-layer circuit board that demonstrates user-friendly reference designs with all the common interfaces.

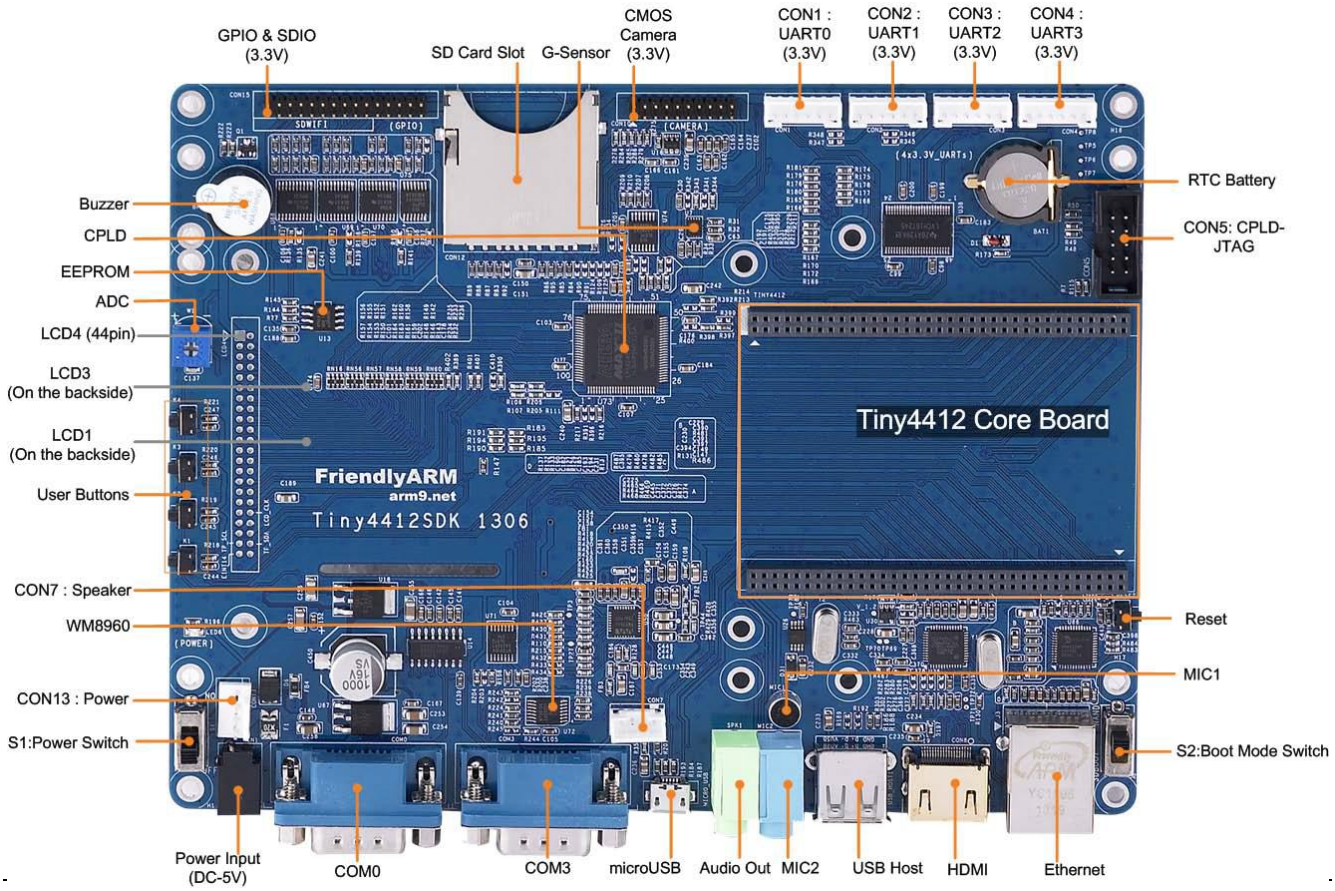
1.2.1.1 Tiny4412 Base Board Enhanced Version

LCD	● LCD1 interface (on the reverse): 45Pin, 0.5mm spacing, compatible with Mini2440/Tiny4412/Mini210S LCD, supports one wire precise touching ● LCD3 interface (on the reverse): 40Pin, 0.5mm spacing, compatible with

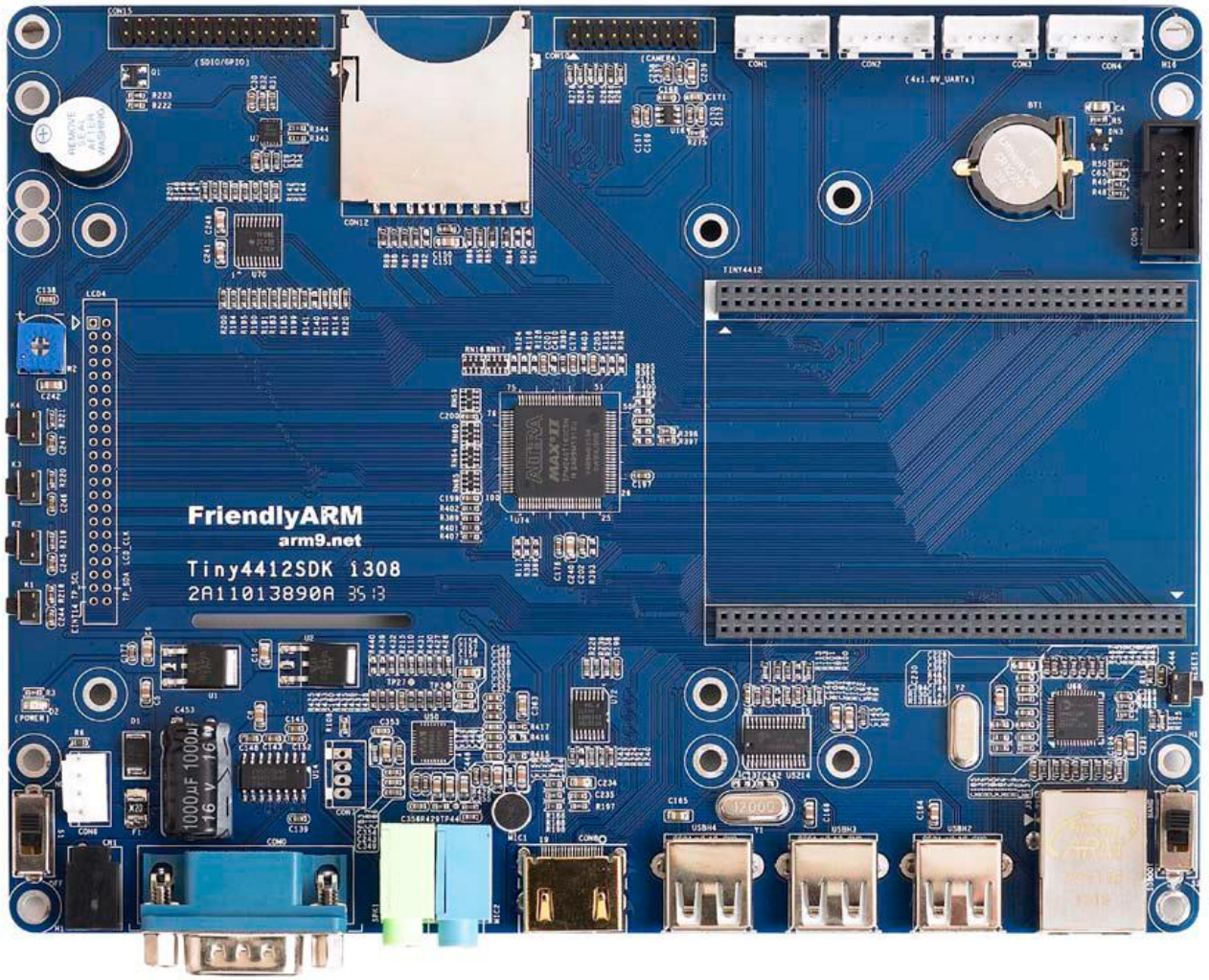
	Mini2440/Mini6410/Mini210S LCD, supports one wire precise touching ● LCD4 interface (on the obverse, reserved): 44Pin, compatible with Mini2440/Mini6410 LCD/Mini210S, supports one wire precise touching ● HDMI high definition interface (Type A) ● LCDs supported from 3.5" to 12.1" , HD
Network	● 10/100M Ethernet interface(RJ45) using DM9621
Standard Configuration	● 2 x DB9 RS232 serial port ● 1 x microUSB Slave-2.0 ● 1 x 3.5mm stereotype audio output ● 1 x integrated microphone ● 1 x one speaker port which can drive an 8Ω 1W speaker ● 1 x USB Host 2.0 ● 1 x standard SD card socket ● 1 x 5V power input (DC-23B)
On Board Hardware Resource	● 1 x I2C-EEPROM (256byte) for I2C bus test ● 4 x Interrupt Style Push Button on module ● 1 x PWM buzzer ● 1 x backup battery for on board real time clock ● 1 x gravity sensor chip
External Resource	● 4 x TTL ● 1 x GPIO (including SDIO) ● 1 x CMOS camera interface
PCB Dimension	● Two Layered Board ● Dimension: 180 x 140(mm)

Software	• Linux Kernel 3.5 • Android 4.2.1

1.2.1.2 Tiny4412 Base Board Enhanced Version Overview



1.2.2 Tiny4412SDK Standard Version

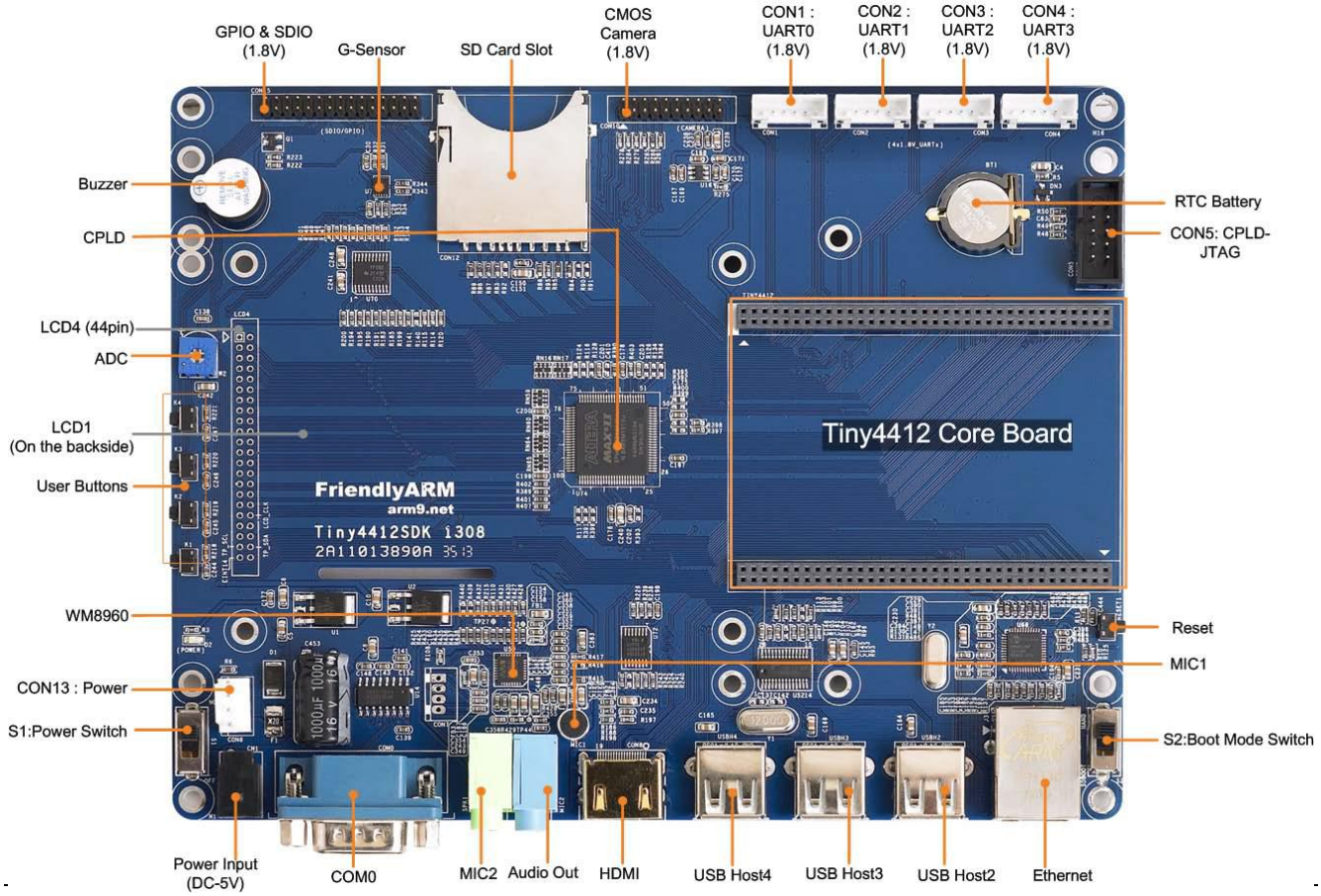


1.2.2.1 Tiny4412 Base Board Standard Version

LCD	● LCD1 interface (on the reverse): 45Pin, 0.5mm spacing, compatible with Mini2440/Mini6410/Mini210S LCD, supports one wire precise touching ● LCD4 interface (on the obverse, reserved): 44Pin, compatible with Mini2440/Mini6410 LCD/Mini210S, supports one wire precise touching ● HDMI high definition interface (Type A) ● LCDs supported from 3.5" to 12.1" , HD
Network	● 10/100M Ethernet interface(RJ45) using DM9621
Standard Configuration	● 1 x DB9 RS232 serial port ● 1 x 3.5mm stereotype audio output

	● 1 x integrated microphone ● 1 x one speaker port which can drive an 8Ω 1W speaker ● 1 x USB Host 2.0 ● 1 x standard SD card socket ● 1 x 5V power input (DC-23B)
On Board Hardware Resource	● 4 x Interrupt Style Push Button on module ● 1 x PWM buzzer ● 1 x backup battery for on board real time clock ● 1 x gravity sensor chip
External Resource	● 4 x TTL ● 1 x GPIO (including SDIO) ● 1 x CMOS camera interface
PCB Dimension	● Two Layered Board ● Dimension: 180 x 140(mm)
Software	● Linux Kernel 3.5 ● Android 4.2.2

1.2.2.2 Tiny4412 Base Board Standard Version Overview

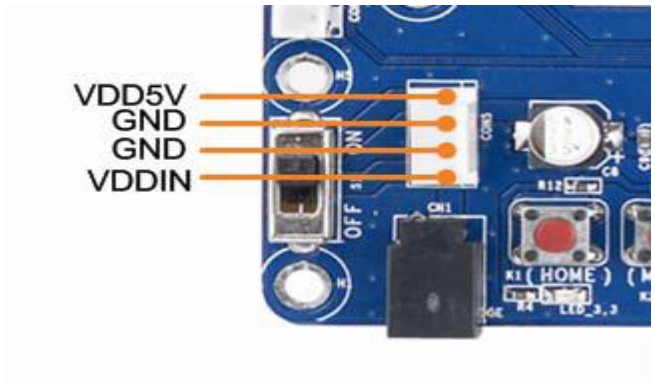


1.3 Tiny4412SDK Interface and Port

This section introduces the main ports and interfaces of the Tiny4412. You can find detailed schematics from the 4412 DVDs

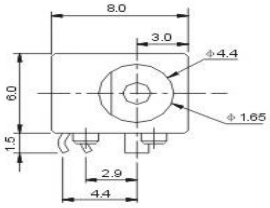
1.3.1 Power

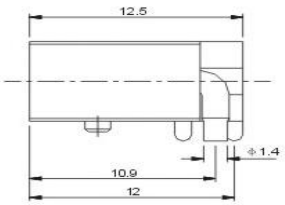
The SDK base board requires 5V DC and has two power input ports. CN1 is the barrel jack for the 5V PSU included with SDK kits. The 4 pin white CON5 takes a connector with a “click in place” for secure power in enclosures or commercial applications.

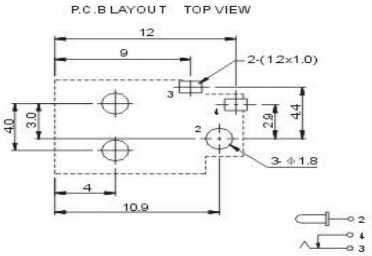
CON5	NO.	Pin Spec
	1	VDD5V
	2	GND
	3	GND
	4	VDDIN
	Note: this layout is convenient for users to connect S1 when it is extended	

Power Socket:

Type: DC023B





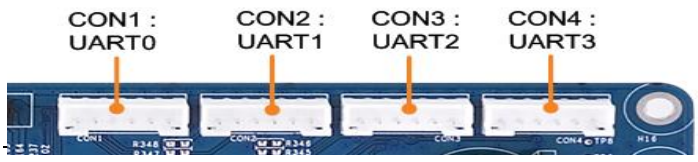


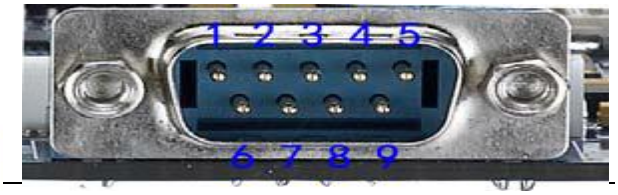
1.3.2 Serial Port

The Exynos4412 has four serial ports: UART0、1、2 and 3. The UART1 is a four wire serial port and UART0, 2 and 3 are two wire serial ports.

For the Tiny4412 the UART0 and 3 are converted to RS232 (COM0 and COM3). You need to use the shipped cross serial cable to connect the board to a PC.

Details of CON1，CON2，CON3, CON4 are as follows:





CON0 ~ CON4	Pin Spec(TTL)	COM0	Pin Spec(RS232)
1	RTSn	1	NC
2	CTSn	2	RSRXD
3	TXD	3	RSTXD
4	RXD	4	NC
5	5V	5	GND
6	GND	6	NC
		7	NC
		8	NC
		9	NC
Note: NC means no connection			

COM3	Pin Spec(RS232)
1	NC
2	RSRXD
3	RSTXD
4	NC
5	GND
6	NC
7	RSCTSn
8	RSRTSn
9	NC

1.3.3 USB

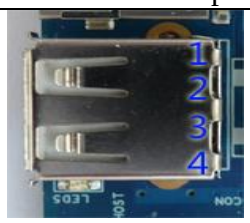
This board has two types of USB interfaces: one USB Host (2.0) and one USB Slave (2.0). The USB Host functions the same as on PC's. You can access USB camera, USB keyboard, USB mouse, USB flash drives and other common USB peripherals. You can also use USB Hub extensions. Each OS has native USB Hub drivers. The USB Slave (microUSB 2.0) is generally used to debug in Android ADB.

microUSB Spec:

miniUSB	Pin Spec
5	GND
4	OTGID
3	D+

	2	D-
	1	Vbus

USB Host Pin Spec



USB Host	Pin Spec
1	5V
2	D-
3	D+
4	GND

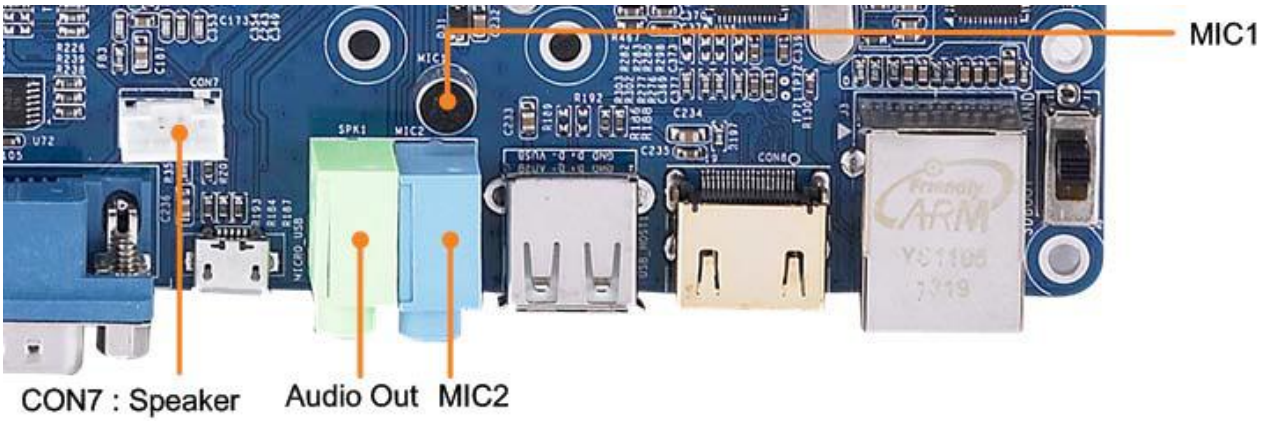
1.3.4 Network Interface

The Tiny4412 base board has a DM9621 LAN chip for adaptive 10/100M Ethernet. The RJ45 connector includes the magnetics. You can use ordinary Cat5 with RJ45 to connect to your router or switch.

1.3.5 Audio

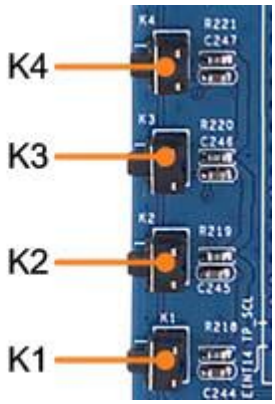
The Exynos4412 supports I2S/PCM/AC97 audio interfaces. The Tiny4412 uses the I2S0 interface with the WM8960 CODEC.

The audio output is on the 3.5mm green jack. The board also has a microphone input. When recording we suggest users to move the microphone close to the audio source



1.3.6 User Button

The Tiny4412 has four user buttons connected directly to the CPU's interrupt pins. They trigger low when closed.



Button	K1	K2	K3	K4
Interrupt	EINT26	EINT27	EINT28	EINT29
GPIO	GPH3_2	GPH3_3	GPH3_4	GPH3_5
Notes:				

1.3.7 LCD Interface

For convenience in mounting various displays the Tiny4412 has three LCD connectors and one of them is 45pin(LCD1) for connecting a capacitive LCD.

The LCD control signals are the same on all connectors with horizontal and vertical scan, clock, enable, disable etc., and 8:8:8 models of RGB data. It has a PWM output and a reset signal (nRESET). LCD_PWR is the backlight switch signal.

Since we apply the one wire precise touch technology neither LCD interface has the four wire resistor touch pins that the CPU uses by default (LCD1-41, 42, 43 and 44). This way gives us flexibility to connect capacitive screens.

LCD1	Pin Spec	LCD1	Pin Spec
1	VDD_5V	2	VDD_5V
3	VD0	4	VD1
5	VD2	6	VD3
7	VD4	8	VD5
9	VD6	10	VD7
11	GND	12	VD8
13	VD9	14	VD10
15	VD11	16	VD12
17	VD13	18	VD14
19	VD15	20	GND
21	VD16	22	VD17
23	VD18	24	VD19
25	VD20	26	VD21
27	VD22	28	VD23
29	GND	30	PWM1/GPD0_1
31	XEINT10/GPH1_2	32	nRSTOUT
33	VDEN	34	VSYNC
35	HSYNC	36	VCLK
37	I2CSCL2	38	XEINT14/GPH1_6
39	I2CSDA2	40	XEINT15/GPH1_7
41	GND		
Note: the Exynos4412 has three I2Cs and here we use I2C2. Please refer to the schematics in the shipped CD for the exact connection details between each pin and CPU. The information provided here is for reference			

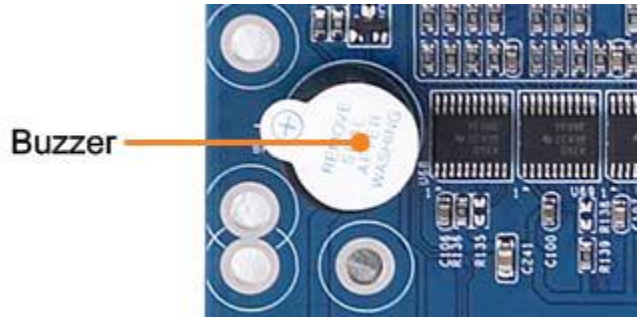
1.3.8 ADC

The Tiny4412 has 4 ADC channels for different purposes. AIN0 is connected to a variable resistor W1 for testing and the remaining AINs are not extended.

1.3.9 PWM Buzzer

The on-board buzzer is controlled by PWM0, the diagram is shown below. PWM0

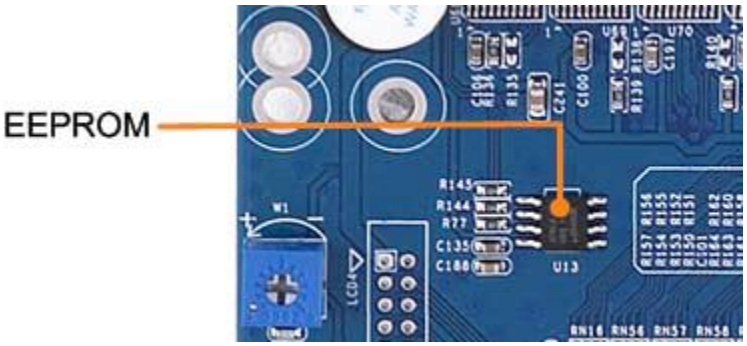
corresponds to GPD0_0 which can be configured as PWM output via software or used as a GPIO.



1.3.10 I2C-EEPROM

The Tiny4412 has a direct connection to an AT24C08 – an I2C EEPROM which has a capacity of 256 bytes and is mainly for testing I2C bus.

Note: the Exynos4412 has eight I2Cs and here the Tiny4412 uses I2C0

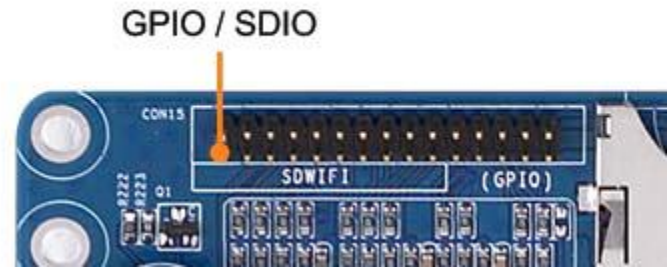


1.3.11 SD Card Socket

The Tiny4412 has two SDIO interfaces. The SDIO0 is used for ordinary SD cards. The interface can support SDHC, that is, high-speed large-capacity cards up to 32G bytes.

1.3.12 GPIO/SDIO Interface

The GPIO is a 2.0mm spacing 30 pin header.



The first 20 pin is an SDIO which is usually used for SD-WiFi.

CON9	Pin Spec	CON9	Pin Spec
1	VDD_3.3V	2	GND
3	TXD2	4	RXD2
5	I2CSCL	6	I2CSDA
7	SPIMOSI0	8	SPIMISO0
9	SPICLK0	10	SPICSn1
11	EINT13	12	EINT12
13	SD3_CLK	14	SD3_CMD
15	SD3_nCD	16	EINT11
17	SD3_DAT0	18	SD3_DAT1
19	SD3_DAT2	20	SD3_DAT3
21	SPIMISO1	22	EINT26
23	SPIMOSI1	24	EINT27
25	SPICLK1	26	EINT28
27	SPICSn1	28	EINT29
29	VDD5V	30	GND
Please refer to the schematics in the shipped CD for the exact connection details between each pin and CPU. The information provided here is for reference			

1.3.13 CMOS Camera

The Tiny4412 has one CMOS camera interface which is a 2.0mm spacing 20 pin header. Users can use our CAM130 cameras by connecting it to this header. Actually the

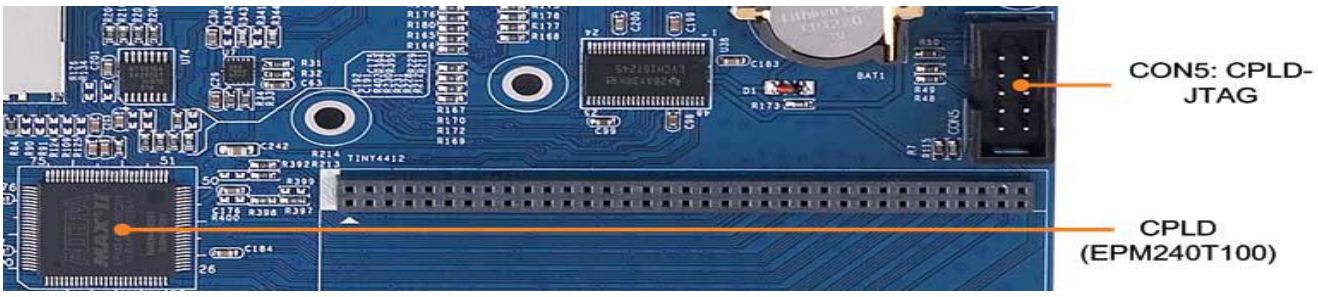
CAM130 doesn't have any circuits and it is a conversion board which applies the ZT130G2 module.

Note: the CAMER interface is a multiplexed port which can be used as a GPIO by setting corresponding registers. The table below shows its GPIO pins.

CAMERA			
CAMERA	**Pin Spec**	**CAMERA**	**Pin Spec**
1	I2CSDA0	2	I2CSCL0
3	XciFIELD	4	CAM_RESET/GPJ3_1
5	CAM_CLK	6	CAM_HREF
7	CAM_VSYNC	8	CAM_PCLK
9	CAM_DATA7	10	CAM_DATA6
11	CAM_DATA5	12	CAM_DATA4
13	CAM_DATA3	14	CAM_DATA2
15	CAM_DATA1	16	CAM_DATA0
17	VDD_3.3V	18	VDD_2.45-2.8V
19	VDD_1.8V	20	GND
Please refer to the schematics in the shipped CD for the exact connection details between each pin and CPU. The information provided here is for reference			

1.3.14 CPLD-JTAG

This interface is used to flash the CPLD (EPM240T100) with firmware.



1.4 Tiny4412 Software Feature

1.4.1 Android4.2.2 Feature

Cross-compiler	arm-linux-gcc-4.5.1-v6-vfp	Same as Mini6410/Mini210S, by default it compiles with armv7 command set. It supports hard floating point arithmetic
Bootloader	Two bootloaders: superboot (non-open source) and uboot (open source).	
	It supports SD card system burning and can install (YAFFS2) systems within 1.8 seconds.	
	It supports Android's fastboot USB flash system	
	It supports SD card formatting.	
Android kernel	Kernel version: Linux-3.5	Complete BSP
	It supports EXT3/YAFFS2/CRAMFS/FAT32.	Open source
	Watchdog	
	RTC driver	
	LED driver	
	User button driver	
	SPI driver	
	I2C-EEPROM driver	
	PWM buzzer driver	
	ADC driver (channel: AIN0)	
	Touch screen coontroller driver which comes with CPU by default	
	Capacitive LCD driver	
	LCD back light driver: it allows users to	

	adjust the board's backlight up to 127 levels	
	LCD driver(common 7" LCD and HD 7" LCD)	
	USB Host driver: it supports flash drives, blue tooth and so on.	
	USB Device driver: it supports USB ADB/Fastboot	
	SD card driver	
	Serial port driver	
	On board SD WiFi driver(Marvell8686)	
	USB WiFi driver: it comes with the kernel but can only drive limited types	
	USB WiFi driver: it supports more types	
	Audio driver(WM8960: it supports audio recording and playing, ALSA API and type D amplifier）	
	Ethernet driver(DM9621)	
	FIMC driver	
	JPEG driver	
	MFC multi-media driver	
	HDMI driver	
	3D accelerator	
	2D accelerator	
	USB to serial driver	
Android System	Version: Android 4.2.2	
Application	2D/3D Acceleration	Good for 2D/3D games

	WiFi	
	HDMI	Up to 1080p
	Back Light Adjusting up to 127 Levels	

1.4.2 Linux Feature

Cross-compiler	arm-linux-gcc-4.5.1-v6-vfp	Same as Mini6410/Mini210S, by default it compiles with armv7 command set. It supports hard floating point arithmetic
Bootloader	Two bootloaders: superboot (non-open source) and uboot (open source).	
	It supports SD card system burning and can install (YAFFS2) systems within 1.8 seconds.	
	It supports Android's fastboot USB flash system	
	It supports SD card formatting.	
Linux kernel	Kernel version: Linux-3.5	Complete BSP
	It supports EXT3/YAFFS2/CRAMFS/FAT32.	Open source
	Watchdog	
	RTC driver	
	LED driver	
	User button driver	
	SPI driver	
	I2C-EEPROM driver	
	PWM buzzer driver	
	ADC driver (channel: AIN0)	
	Touch screen coontroller driver which is supported by CPU	

	Capacitive LCD driver	
	LCD back light driver: it allows users to adjust the board's backlight up to 127 levels	
	LCD driver(common 7" LCD and HD 7" LCD)	
	USB Host driver: it supports flash drives, blue tooth and so on.	
	USB Device driver: it supports USB ADB	
	SD card driver	
	Serial port driver	
	On board SD WiFi driver(Marvell8686)	
	USB WiFi driver: it comes with the kernel but can only drive limited types	
	USB WiFi driver: it supports more types	
	Audio driver(WM8960: it supports audio recording and playing, ALSA API and type D amplifier）	
	Ethernet driver(DM9621)	
	FIMC driver	
	JPEG driver	
	MFC multi-media driver	
	3D accelerator	
	2D accelerator	
	USB to serial driver	
	3G driver	Open Source
GUI System	Qtopia – 2.2.0	Open Source for x86 and ARM

	QtEmbedded – 4.7.0	Open Source for ARM
	Qt-Extended-4.4.3	Open Source
Application	ADC utility	
	LED utility	
	Buttons utility	
	I2C-EEPROM	
	LCD Test	
	Ping	
	Audio Recorder	
	Web Browser	
	Watchdog	
	Network Configuration	
	Language Settings	
	Back Light Adjusting up to 127 Levels	
	MMC/SD Card Mount/Umount	
	Qt4 Switcher	
	Qtopia4 Switcher	
	SMPlayer	

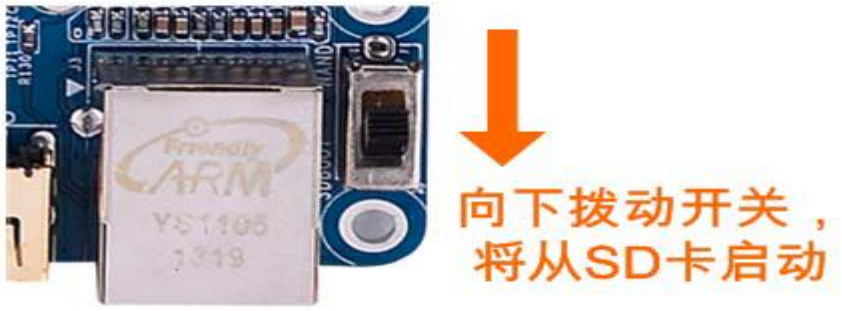
2 Getting Started

By default, all our systems have been preinstalled with Android 4.2.1 (located in the shipped CDs' directory /images/Android are zImage, ramdisk-u.img, system.img and so on) therefore you can easily boot the board and play.

2.1 System Setup and Configurations

2.1.1 Boot Option

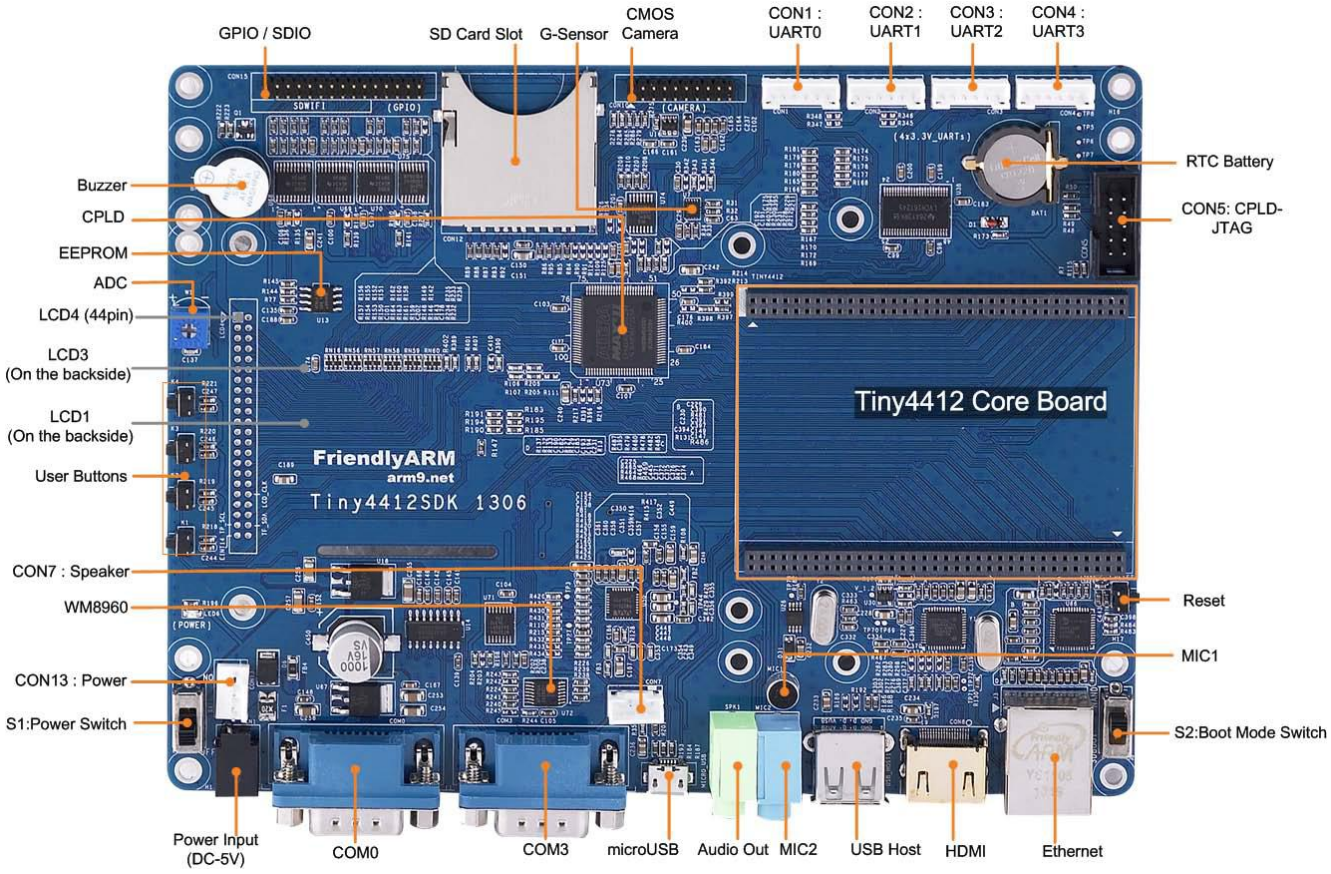
The Tiny4412 supports booting from either SD card or eMMC. Users can switch between by toggling the S2 switch:

Screenshot	Operation	Comment
	Toggle S2 to "NAND" and the board will boot from the eMMC	Default and Standard booting.
	Toggle S2 to "SDBOOT" and the board will boot from SD card	This is for system booting or burning

Usually, S2 is switched to "Nand" unless users need to boot from the SD card or reflash the system.

2.1.2 Interface and Port

The Tiny4412SDK has the following interfaces, ports and sockets:



Please follow the steps below to hook up the board:

- Connect the Tiny4412 board's serial port0 (Debug Serial Port) to a PC's serial port with the shipped crossover serial cable (blue one) in the package
- Connect the Tiny4412 board's Ethernet interface to a PC with the shipped crossover cable (this step can be skipped if you don't need to connect to the internet)
- Connect the shipped 5V power supply adapter to the 5V power supply interface on the board (do it with care to prevent damaging the interface)

- Connect a headphone or speaker to the audio input(green) on the board
- Connect an LCD touch screen (if the user has one) to the LCD interface on the board following the data bus' arrow

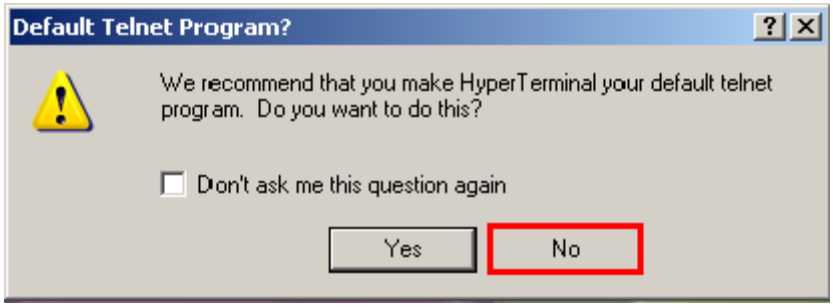
2.1.3 Setting up HyperTerminal

Note: some users attempt to expand the board's serial ports by using a USB to serial port cable. Sometimes this operation would cause error codes. This might be an indication that the cable doesn't work. Most of our agents have these conversion cables for sale. Users can contact them. In addition we strongly recommend users to use our shipped crossover serial cable. Other serial cables might not work properly.

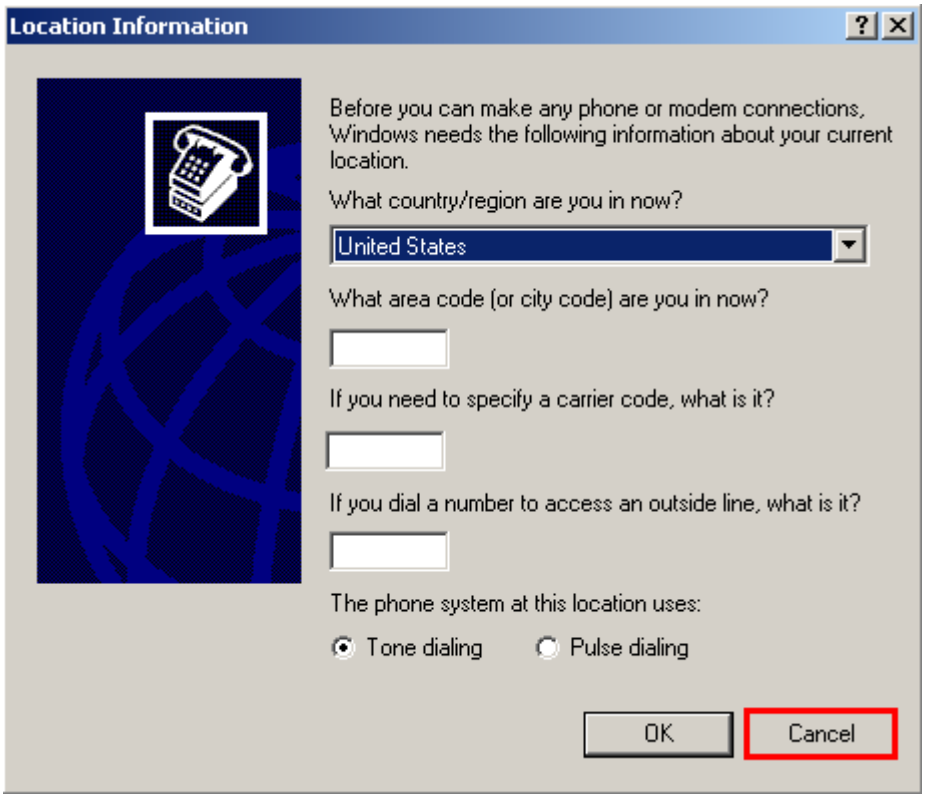
To connect the Tiny4412 board to a host PC via a serial cable, you should use a simulation terminal. There are many tools available. A most widely used one is the MS-Windows' super terminal. In Windows9x, you need to install it by checking that option during installation. Windows2000 and later

A common Linux desktop version has a similar terminal too and it is minicom. It is a command line utility which may not be easy for beginners. Interested users can search the internet for more resources.

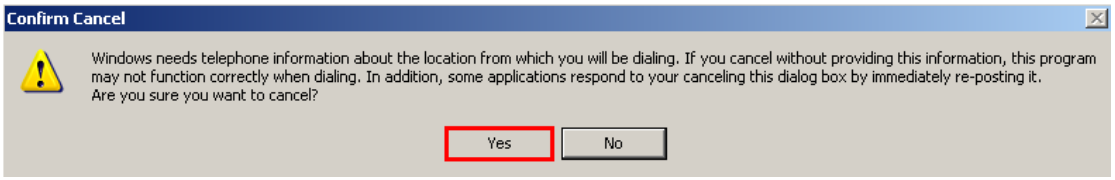
We take WindowsXP's super terminal for instance. You can find it by going to "Start->Programs->Accessories->Communications". After it starts the following dialog will pop up, please click on the "No" button

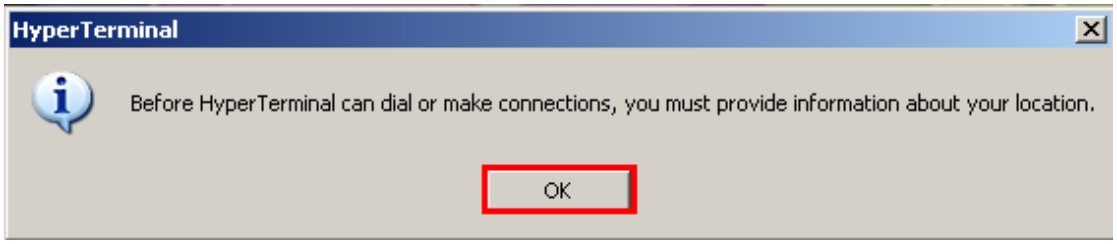


Click on the “Cancel” button on the following dialog



Click on the “Yes” button and the “OK” button to the next step

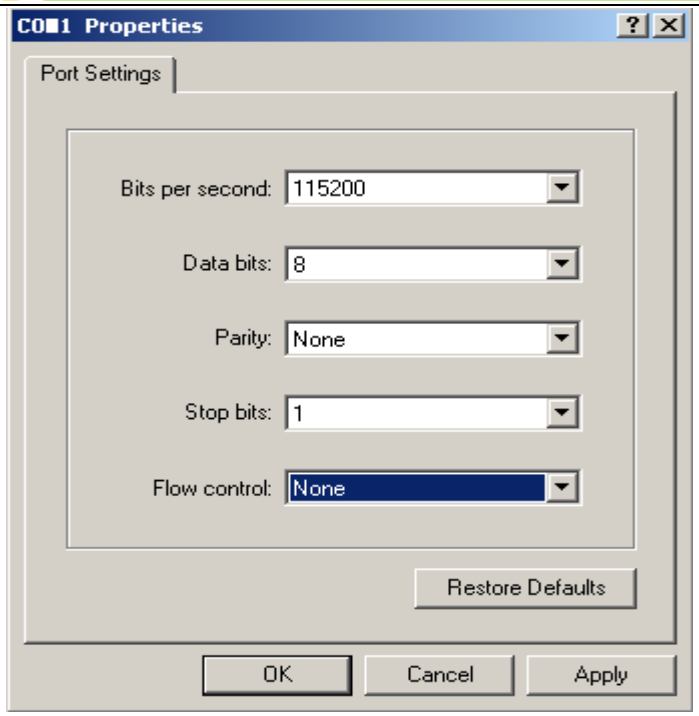




A popup window will require you to name this connection. In this example we typed "ttyS0". Windows does not accept names like "COM1" that have already been used by the system.

After naming this connection another window will require you to select a serial port that will be used to connect the Tiny4412 board. Here we selected COM1:

Lastly, also the most important step is to set up the port properties. Note: you must select "No" in the data flow control field otherwise you will only be able to see outputs. In addition the bits per second should be set to 115200.



After setting up all properties, turn on the board's power supply, if the connection gets set properly, you will see a bootloader startup interface. If everything runs fine please save this connection for later use.

2.2 Burning Superboot to SD Card

In order to boot from an SD card, you need to burn BIOS to it. FriendlyARM offers a flashing utility: SD-Flasher.exe which can burn our Bootloader (Tiny4412-Boot.bin) to an SD card.

2.2.1 Burning Tiny4412-Boot.bin to SD Card

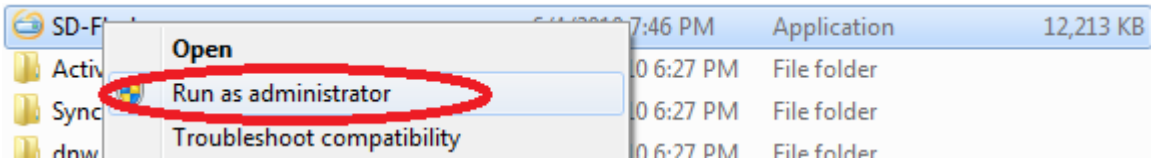
We tested the following steps on Windows7

Note: users complained that some notebook's integrated SD card reader cannot work properly with card burning or reading. So far we haven't encountered this issue and we

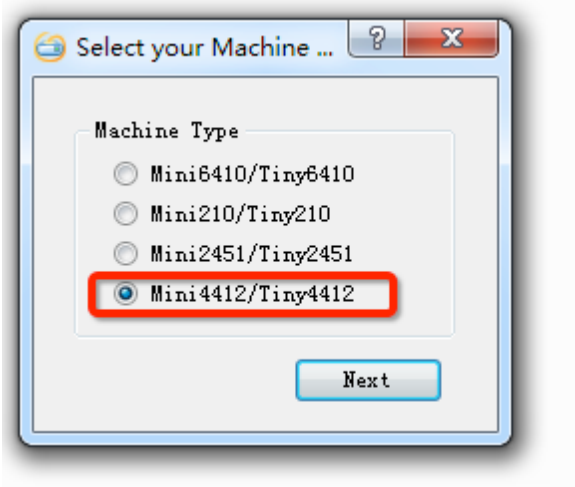
suggest that you should try a common card reader in this case.

Our SD-Flasher.exe formats a 130M space for the bootloader therefore an SD card whose memory is less than 256M cannot work and we recommend using one whose memory is at least 4G

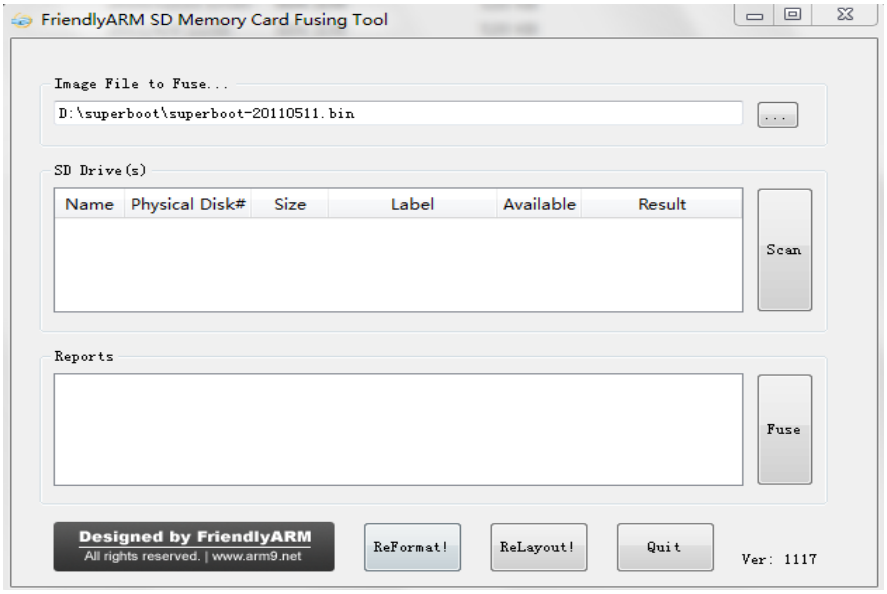
Step1: launch the SD-Flasher.exe in your shipped CD (under "\tools\"). Note: this program should be run as "administrator"



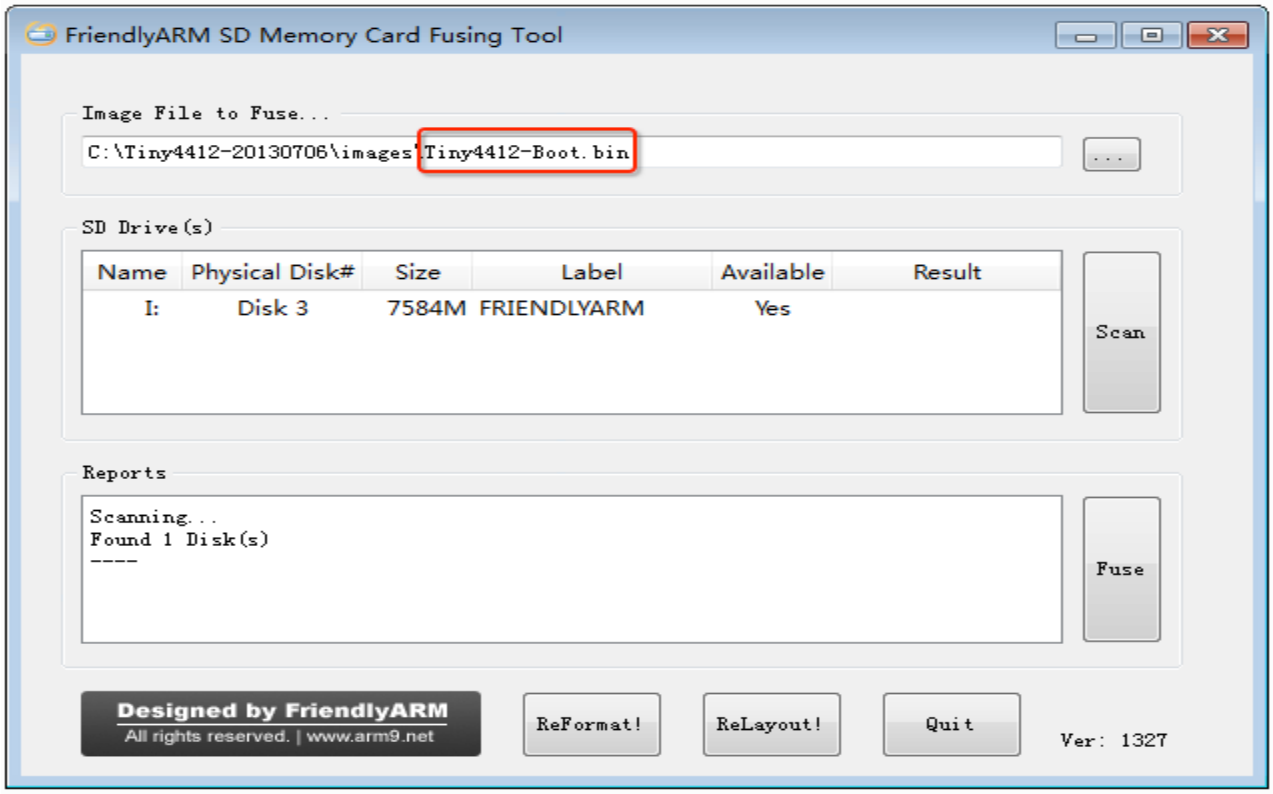
When the utility is launched a message box will pop up "Select your Machine…", please select "Mini4412/Tiny4412":



Below is the dialog you will see after it is started. Note: the "ReLayout" is enabled and we will format the SD card with this function.



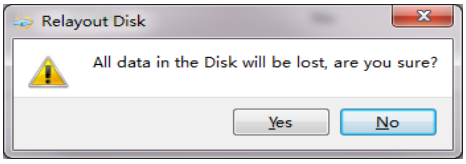
Step2: click on  to select your bootloader file



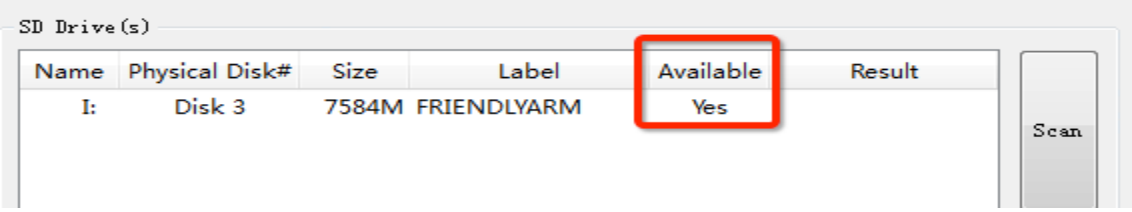
Step3: insert a FAT32 SD card into your host's SD card socket (you can also use a USB card reader to connect to a PC), backup your data in the card and click on "Scan", all

recognized SD cards will be listed.

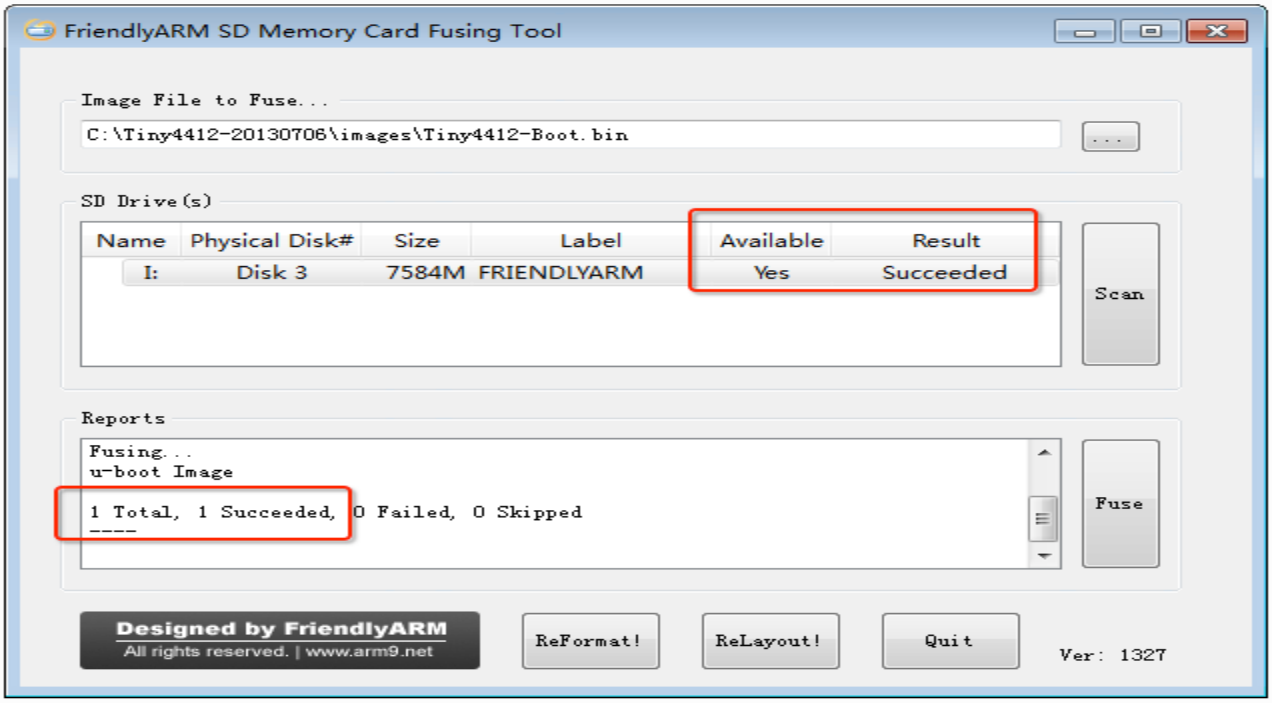
Step4: click on “ReLayout”, the following dialog will pop up prompting you that the data in your card will be lost. Just click on “Yes”



After formatting is done you will be directed back to the main menu. Click on “Scan”, you will see that a “FriendlyARM” section available.



Step5: click on “Fuse”, Superboot will be safely burned into the SD card. You can burn this card in WindowsXP without worrying about its FAT32 data being lost or damaged.



The Tiny4412-Boot.bin in your SD card is invisible. To verify it you can insert your SD card into your board's SD card socket and connect your board to a PC via a serial cable. Switch S2 to the "SDBOOT" mode, reboot your board and if messages pop up from the hyperterminal it is indicating that your SD card is functioning.

If there are no output messages from your serial port it may indicate your burning was not successful. The following cases could result in this failure:

1. You might use a notebook and the notebook's the card reader might not work. We suggest using an external usb card reader.

2. You might use a bad SD card. We suggest using one whose memory is at least 4G or SDHC

3. You might use an SD card. Please use a standard SD card which can be directly connected to your card reader

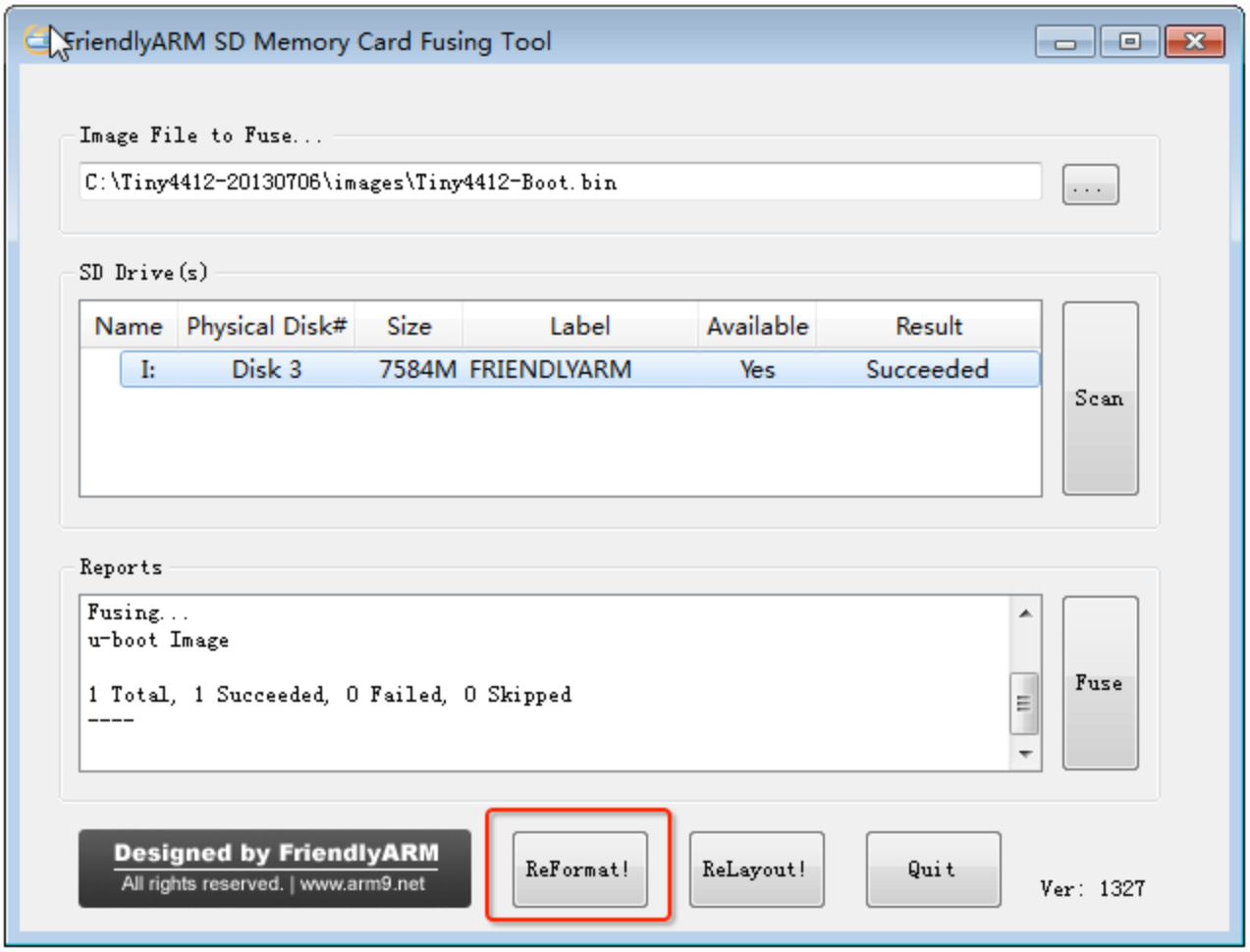
4. The SD card booting function is integrated in Samsung's chip and the ROM is preinstalled. It might not recognize some cards. In this case we suggest you try some different cards

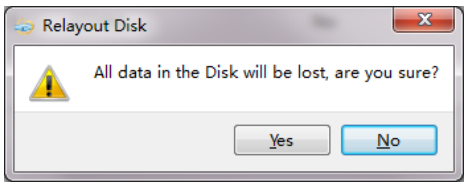
5. Poor contact might be another reason. In this case you could try a few more times: by unplugging and plugging the core board and the base board (if your board is a tiny board) and unplugging and plugging the SD card

2.2.2 Restore SD Card

Note: we tested the following steps on Windows7

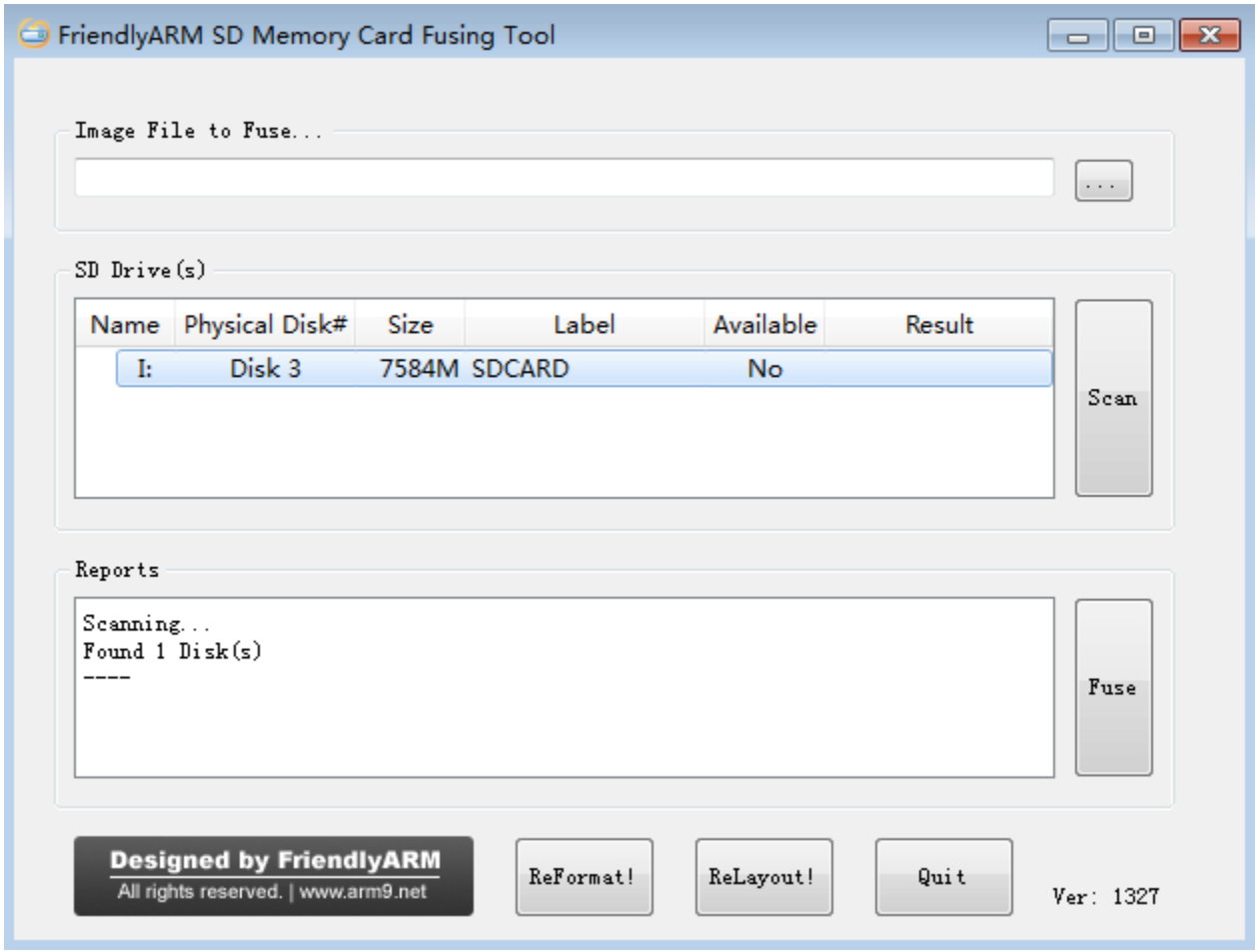
Using SD-Flasher.exe will reserve 130M memory for Tiny4412-Boot.bin. When you no longer need your SD card for system burning you might want to restore your card to what it was. You can do it this way: launch SD-Flasher.exe as an administrator; click on "scan" and "ReFormat" you will see the following dialog





Click on "Yes". A moment later click on "Scan" again, you will find your card

becomes “no” available and your card is restored successfully.



2.2.3 Notes to Users

Common SD cards are used as storage cards therefore on Vista/Windows7 SD-Flasher automatically formats an SD card to two sections: one is FAT32 (named “FriendlyARM”) and the other (by default 130M) reserved for the bootloader.

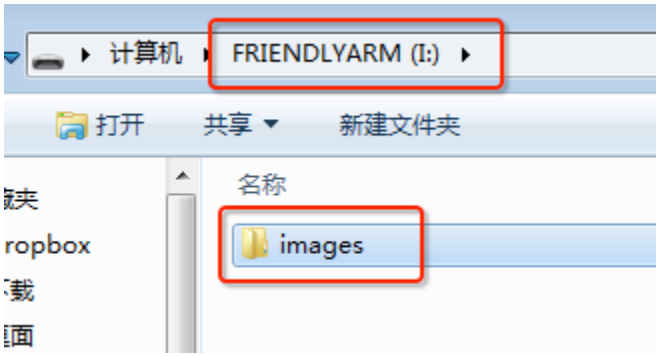
In fact, Vista/Windows7’s system security policies don’t permit unauthorized users to start auto burning an SD card thus common users need to format the SD card first and then burn data into it.

To burn Tiny4412-Boot.bin please run the SD-Flasher on Windows7. Running

SD-Flasher on XP may not work.

2.2.4 Copy Images to SD Card

When installing systems with an SD card you need to copy the whole "images" directory to the root directory of the SD card.



2.3 Install Systems with FASTBOOT

With fastboot you can install systems easily.

2.3.1 Pre-requisits

Before you start with fastboot you need to do the following things

1) Copy the "android_tools.tgz" from the "tools" directory of your 4412DVD to the root directory of Ubuntu. If you download your ISO from the internet you can try the commands below:

   # mkdir –p /mnt/iso

   # mount –o loop Tiny4412-20130707.iso /mnt/iso

   # cd /

   # tar xvzf /mnt/iso/tools/**android_tools.tgz**

2) Connect your board to a PC via a serial cable and open a minicom terminal.

3) Make a bootable SD card with SD-Flasher (refer to section 2.2)

4) Get your system files ready. For Android you need ramdisk-u.bin and system.img which correspond to the root file system and the system section. You can get these two files from the "images/Android" directory of the 4412 DVD.

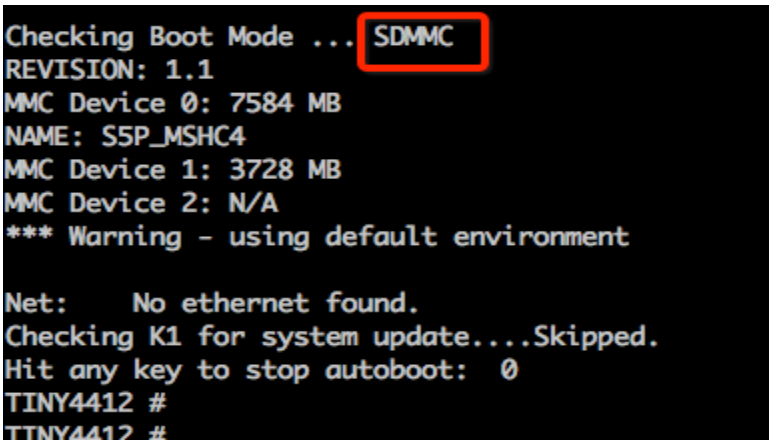
2.3.2 Install Systems with FASTBOOT

The following steps are tested with Ubuntu12.04.2. Theoratically with other Linux versions as long as FASTBOOT is properly installed the following steps should work too.

Step1: connect your board to a PC via a serial cable and open up a hyperterminal or minicom.

Step2: switch S2 to eMMC and power on

Step3: right after hyperterminal or minicom has output messages please press "enter" within 3 seconds you will enter the command line interface



Step4: type “fastboot” in the command line and enter. Bootloader will enter the USB download mode and wait for download’s beginning

Step5: connect your board to a PC via a microUSB cable. On your PC please go to the directory which contains the ramdisk-u.img and system.img files and run “burn-img.sh” and the image files will be burned to the board

2.3.3 Update Individual Files

The following steps should be done in the USB download mode

2.3.3.1 Update Kernel

Go to the directory which has the zImage file and run the following command

```
# fastboot flash kernel zImage
```

2.3.3.2 Update Root File System

Go to the directory which has the ramdisk-u.img file and run the following command

```
# fastboot flash ramdisk ramdisk-u.img
```

2.3.3.3 Update System Section

Go to the directory which has the system.img file and run the following command

fastboot flash system system.img

2.4 Install Systems with SD Card

Before you start with the following steps please make sure you have your SD card flashed ready and copied the whole images directory to your SD card.

2.4.1 Install Android

Step1: insert the SD card to a PC and open the "images\FriendlyARM.ini" file and make sure the contents are as follows

```
#This line cannot be removed. by FriendlyARM(www.arm9.net)
CheckOneButton=No
Action = Install
OS = Android
LowFormat = No
VerifyNandWrite = No
LCD-Mode = No
CheckCRC32=No

 StatusType = Beeper | LED
 ################### Android ####################
 Android-BootLoader = Superboot4412.bin
 Android-Kernel = Android/zImage
 Android-CommandLine = console=ttySAC0,115200n8 androidboot.console=ttySAC0 ctp=2
 skipcali=y vmalloc=512m
 Android-RamDisk =Android/ramdisk-u.img
 Android-RootFs-InstallImage = Android/system.img
 Android-UserData = Android/userdata.img
```

Note: if you use the Tiny4412 SDK 1308 base board you need to add "uhost0 = y" in

the Android-CommandLine. If you use a resistive LCD you need to add “ctp = 0” and “skipcali = n”

Step2: make sure the SD card has the following files. If it doesn’t please copy those files from DVDs

File	Comment
images\Superboot4412.bin	Bootloader
images\Android\zImage	Android Kernel(Linux Kernel 3.5)
images\Android\ramdisk-u.img	Andoird root file system
Images\Android\system.img	Android system section
Images\Adroid\userdata.img	Android data section
images\FriendlyARM.ini	Configuration file

Step3: insert the SD card to the board’s SD card socket. Switch S2 to “SD” and power on. If everything works fine hyperterminal/minicom will show system installation messages

Step4: after system installation is done system will enter the command line interface. Switch S2 to “Nand” , power on the board and Android will load.

2.4.2 Install Linux

Step1: insert the SD card to a PC and open the "images\FriendlyARM.ini" file and make sure the contents are as follows

```
#This line cannot be removed. by FriendlyARM(www.arm9.net)
CheckOneButton=No
Action = Install
OS = Linux
LowFormat = No
VerifyNandWrite = No
LCD-Mode = No
CheckCRC32=No
 StatusType = Beeper | LED

   ################### Linux ####################
   Linux-BootLoader = Superboot4412.bin
   Linux-Kernel = Linux/zImage
   Linux-CommandLine = root=/dev/mmcblk0p1 rootfstype=ext4 console=ttySAC0,115200
   init=/linuxrc ctp=2
   Linux-RamDisk = Linux/ramdisk-u.img
   Linux-RootFs-InstallImage = Linux/rootfs_qtopia_qt4.img
```

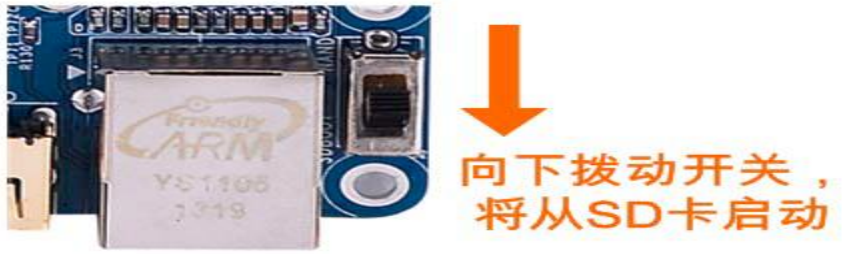
Note: if you use the Tiny4412 SDK 1308 base board you need to add "uhost0 = y" in the Android-CommandLine. If you use a resistive LCD you need to add "ctp = 0" and "skipcali = n"

Step2: make sure the SD card has the following files. If it doesn't please copy those files from DVDs

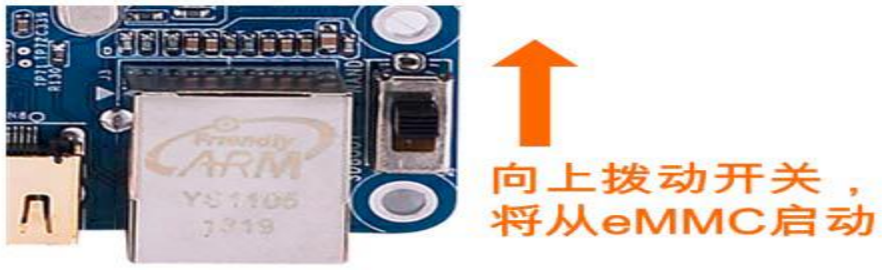
File	Comment
images\Superboot4412.bin	Bootloader
images\Linux\zImage	Linux Kernel 3.5
images\Linux\ramdisk-u.img	Linux root file system
Images\Linux\rootfs_qtopia_qt4.img	Linux system section
images\FriendlyARM.ini	Configuration file

Step3: insert the SD card to the board's SD card socket. Switch S2 to "SD" and power

on. If everything works fine hyperterminal/minicom will show system installation messages



Step4: after system installation is done system will enter the command line interface. Switch S2 to “Nand” , power on the board and Linux will load.



2.5 Install Systems with Minitools

The Minitools utility is a FriendlyARM developed USB download tool which allows users to install systems more easily and conveniently. It has the following features:

- Only need a USB cable: with the Minitools users only need a USB cable to install systems
- One key action: no need to type any command.
- Works with both 32/64-bit OS: it can be installed on both 32-bit and 64-bit Windows systems
- Cross platform: it can be installed on both Windows and Linux systems

2.5.1 Install Minitools

2.5.1.1 Install on Windows

Double click on the "MiniTools-Windows-YYYYMMDD.exe" icon in the tools directory in your shipped DVD and you will be guided to install it. Just follow the prompts and take the default options. When it asks whether you want install the driver please go by "continue anyway". After installation is done please unplug and plug the USB cable and Windows will prompt that it is updating drivers. After Windows' updating is done you can continue

If your installation is successful there will be an icon on your desktop. You can double click on it to run:

The minitools' main window is shown below:

2.5.1.2 Install on Linux

We tested installing the Minitools on Fedora9/Fedora15/Ubuntu12.04 64-bit systems. Please login and execute the installation as root. Please copy the "MiniTools-Linux-YYYYMMDD.tgz" in the "tools" directory from your DVD to your PC and untar the ball and run the "./start.sh" command to the installation.

2.5.2 Flash SD Card with Superboot

In order to work with the Minitools you need to get an SD card and flash it with our superboot. Please follow the steps below:

1. Please flash the superboot to an SD card with "SD-Flasher"
2. Please copy the whole "images" directory from your DVD to the root directory of your SD card
3. Open the "images/friendlyARM.ini" and add the following line

   USB-Mode = yes

Please follow the steps below to connect your board to your PC

1. Switch the S2 on your board to "SD"
2. Power on the board and you will see the LCD showing "USB Mode:Waiting" if everything works correctly
3. Please connect your board to your PC via a USB cable
4. If the connection is successful the LCD will show "USB Mode:Connected"

Now you can start installing systems with the Minitools

To change the installation method back to SD card installation you just need to change the “USB-Mode = yes” to “USB-Mode=no”.

2.5.3 Install Systems with Minitools

Please enter the USB download mode and connect your board to your PC which runs the Minitools via USB

On the left bottom of the window there is a LED which is green indicating the board is connected successfully. On the left bottom there is a button which can start your

board directly without switching to NAND.

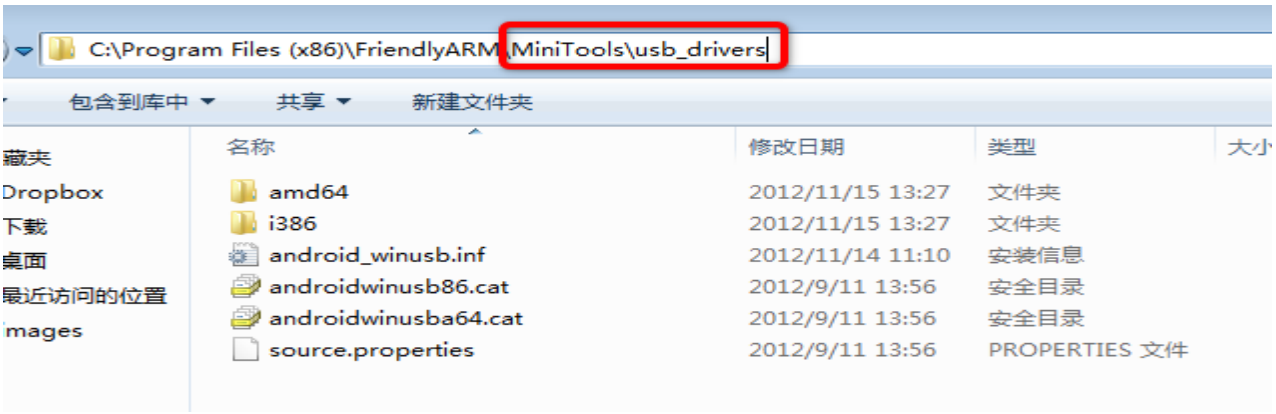
Before install systems please select the system you want to install e.g. Android and then its configuration will be presented as follows:

You can just click on the “images” button to select an “images” directory which contains complete installation files for all systems and the Minitools will show all the info listed in the FriendlyARM.ini.

With the Minitools utility you can update either the whole system (all image files) or individual image files e.g. the kernel image file. After you are done with your installation configuration please click on “Start installation”

After installation is done you can boot your board and enter your system.

Note: sometime users complain that Minitools shows the board isn't connected to PC. It is very likely that the USB download driver is not properly installed on your PC and you can try manually install the USB download driver which is under the Minitools directory in the shipped DVD



3 Android Installation and Navigation

The Tiny4412 can run Android 4.2.1 (Linux kernel 3.5). We migrated all the utilities we developed for the Mini6410 and Tiny210 to the Tiny4412. This not only meets most customers' requirements but also enables users to focus on application development.

3.1 Playing with Android

3.1.1 Android Buttons

User button layout:

The Tiny4412 has 4 buttons:

Their definitions in Android are as follows

Key	Function
K1	**Back**
K2	**Home**
K3	**Menu** and **Screen rotation**
K4	**OK**

3.1.2 Android Commands

3.1.2.1 Enter Android Command Line Mode

Quite often users need to enter the commandline mode to operate the system

Users can do so by connecting the board to a PC via either a USB cable or a serial cable.

When using a USB cable users need to install Android SDK. Then users can type adb shell to enter the commandline interface as root.

When using a serial cable users will enter the commandline interface by default. But

this doesn't give users the root's access and you can command "su" to login as root

3.1.2.2 Set System Section Read/Write Access

By default Android's system section only has read access. You can type the following command to make it writable.

```
# mount -o remount /dev/block/mmcblk0p2 /system
```

But after system reboots this section will be read only again.

If you want the section writable permenantly you need to change Android's source code. Please open "device/friendly-arm/tiny4412/fstab.tiny4412".

Change this line

```
/dev/block/mmcblk0p2 /system ext4 ro
```

To

```
/dev/block/mmcblk0p2 /system ext4 rw
```

3.1.2.3 Tansfer Files from PC to Tiny4412

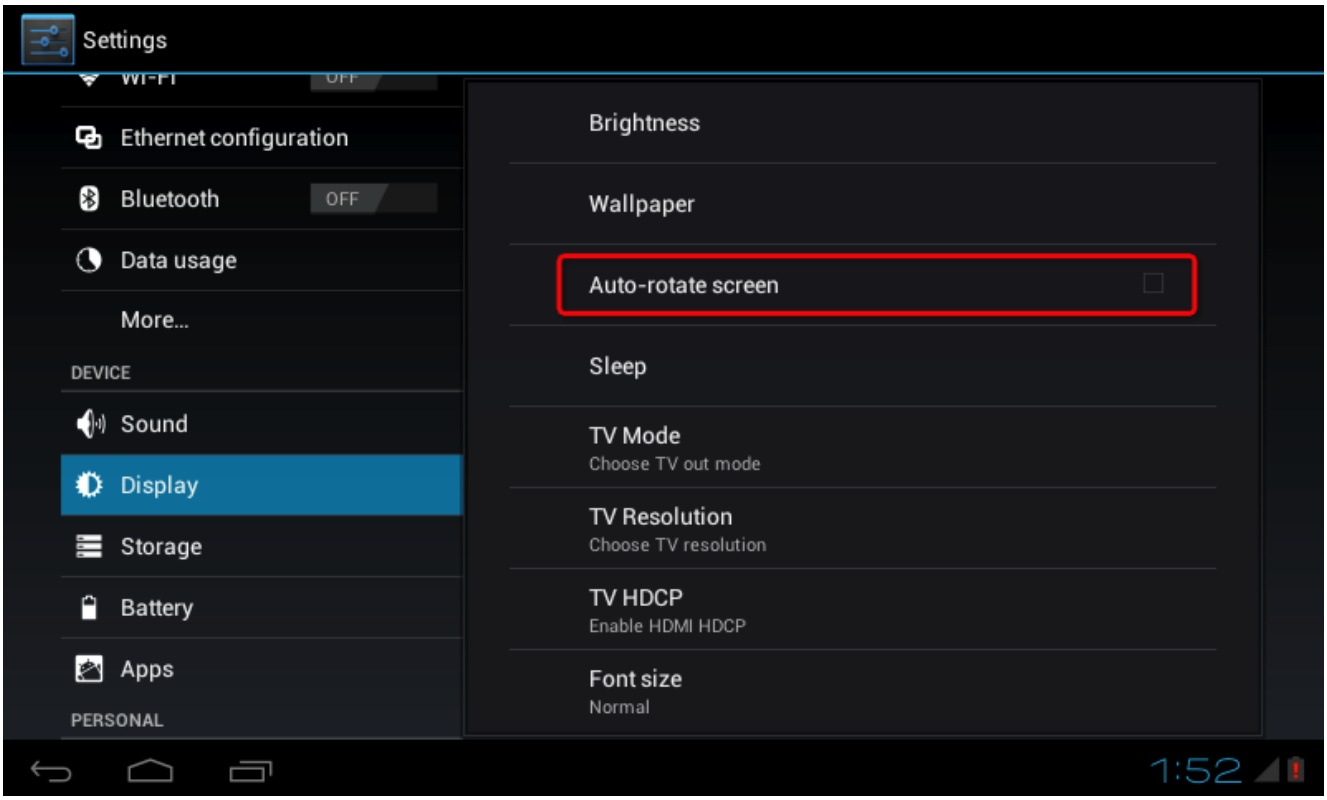
Using Android SDK' adb commands can transfer files from PC to the Tiny4412 via USB. For example to transfer "sensors.tiny4412.so" to "/system/lib/hw" you can type the following command

```
adb push sensors.tiny4412.so /system/lib/hw/
```

Note: before transferring files to the system directory you need to make the system directory writable (Please refer to the previous section)

3.1.3 Screen Rotation by Gravity Sensor

The Tiny4412 has a gravity sensor and by default it supports automatical screen rotation. If you want to disable this function you can go to “Settings”-> “Display” and turn it off.

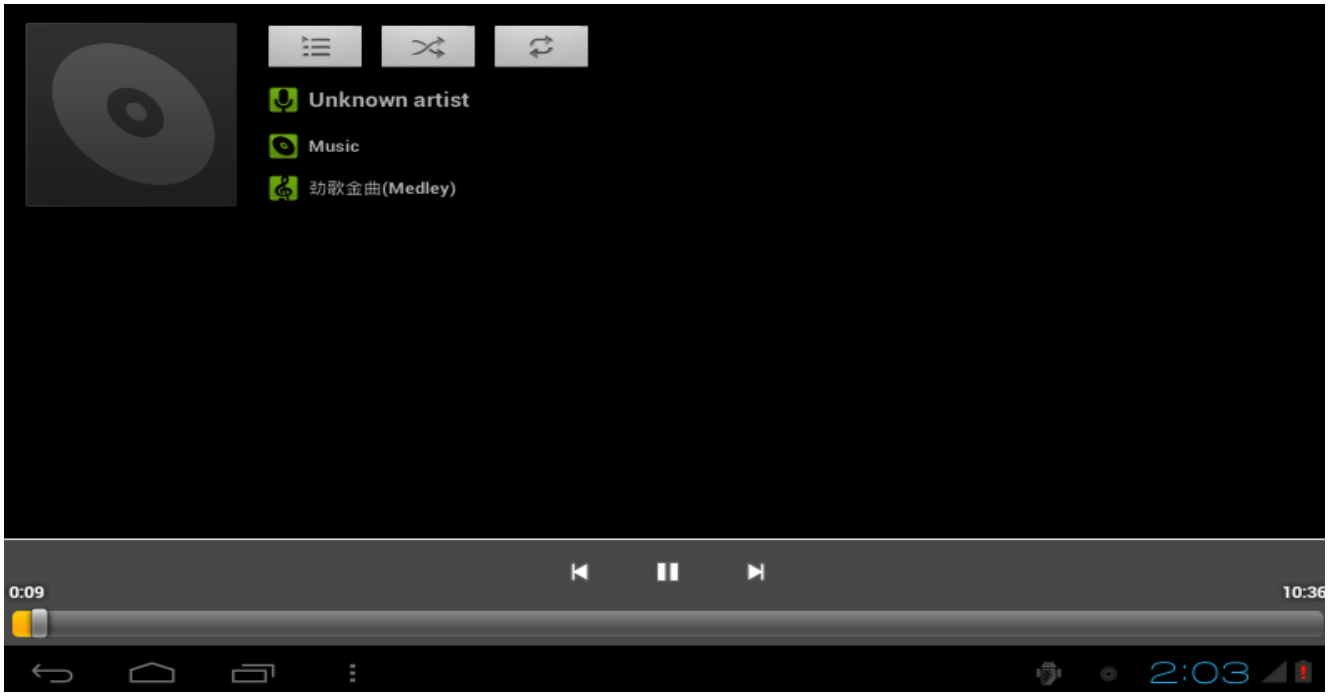


3.1.4 SD Card

The SD card is mounted at /storage/sd_external/

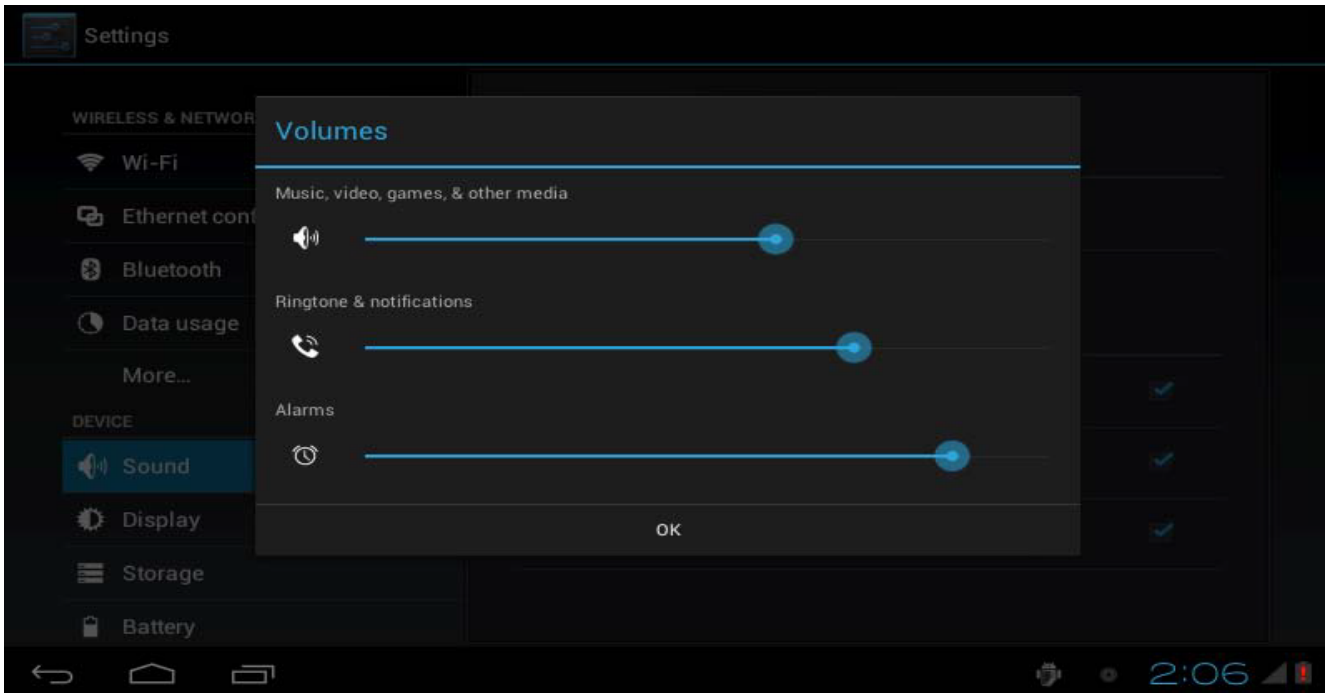
3.1.5 Play MP3

Android can automatically detect MP3 files in the SD card.



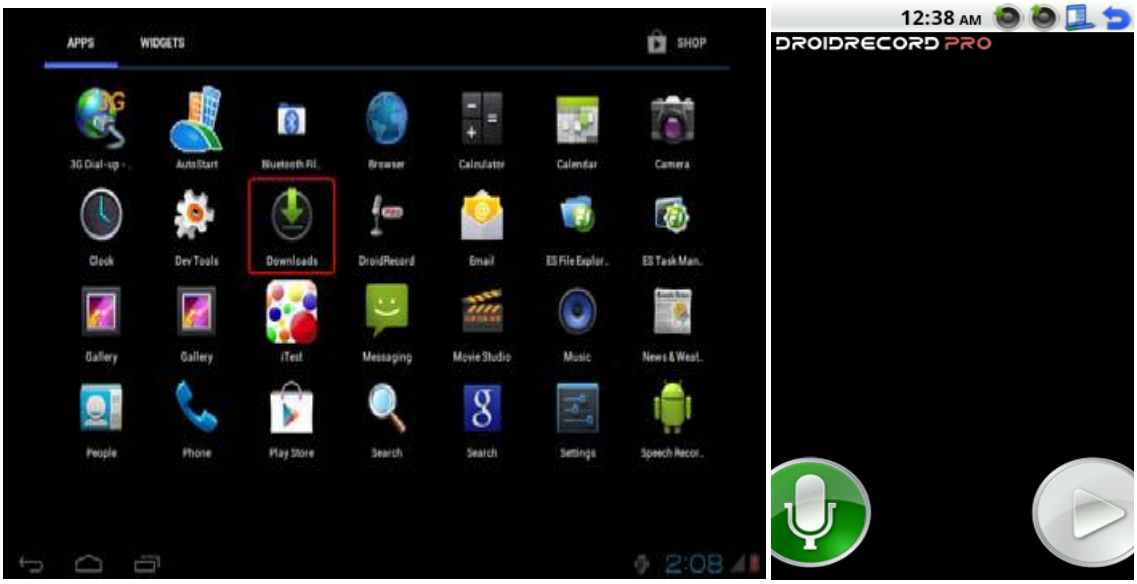
3.1.6 Adjust Volumn

Please go to “Settings” -> “Sound” to adjust the volumn



3.1.7 Audio Recording

The DroidRecord utility can record and play audio. Double click on the icon to launch it.



Please follow the screenshots below to start recording and play:

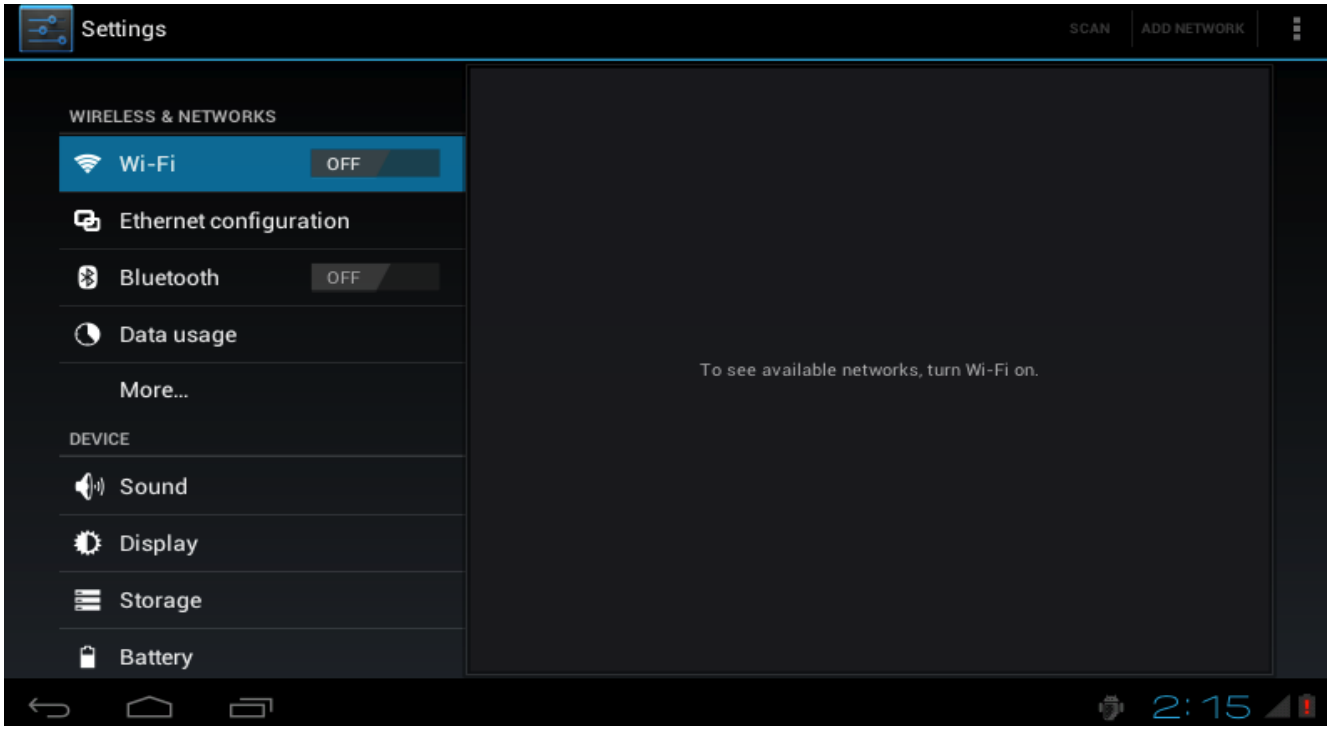
3.1.8 Ethernet

As for the Ethernet the Tiny4412 supports plug and play in Android. Before the board

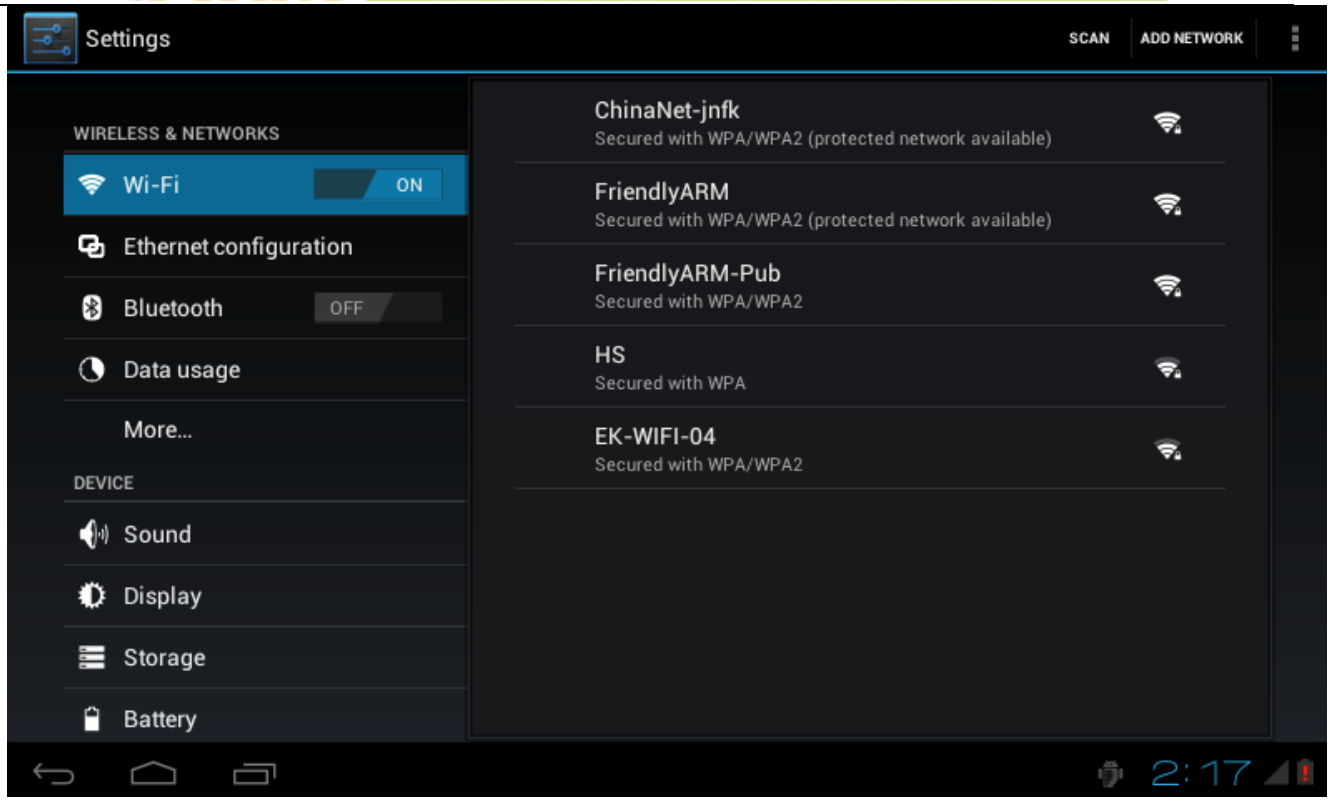
powers on please connect your board to the internet via an Ethernet cable then powers on and you can surf the internet. Now it only supports DHCP and you don't need to do any configuration.

3.1.9 SD WiFi

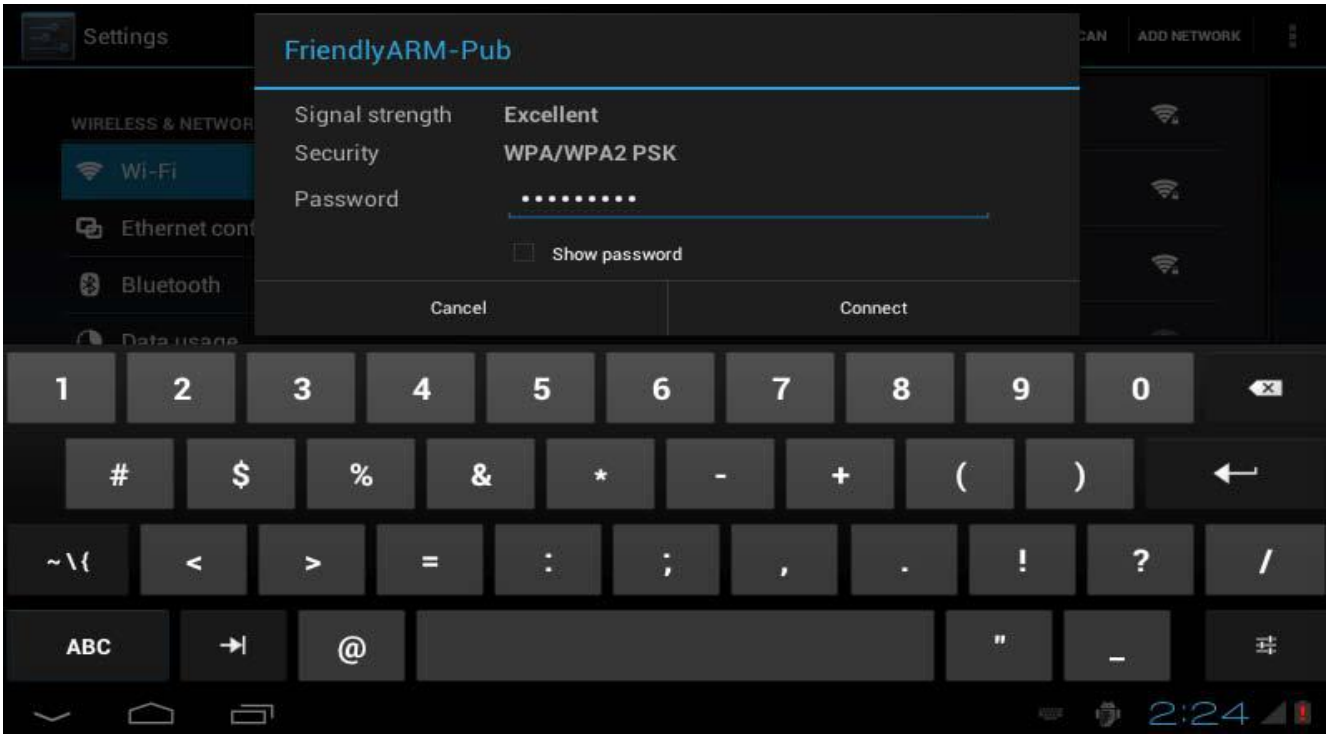
The Tiny4412 can connect to a USB WiFi module.



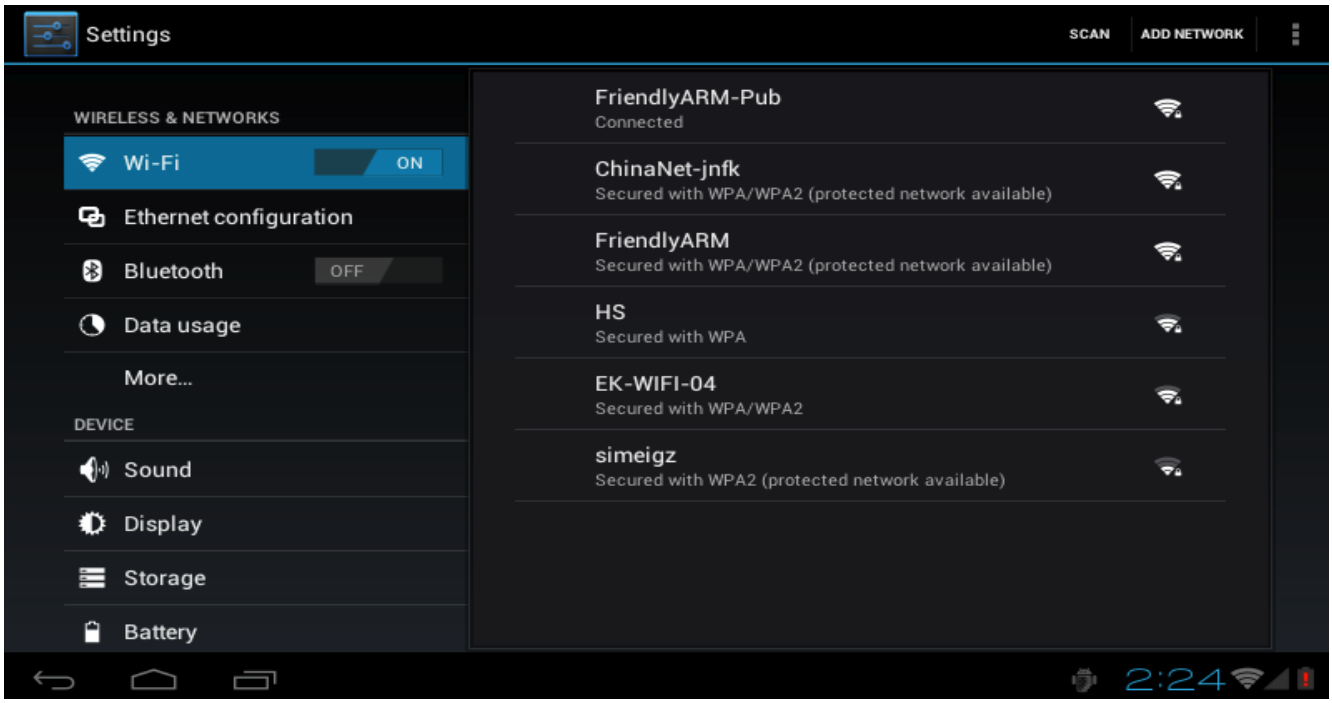
Click on "WiFi" and slide the "ON/OFF" switch to turn it on it will search the nearby WiFi access points:



Click on the WiFi you want to connect and type the password



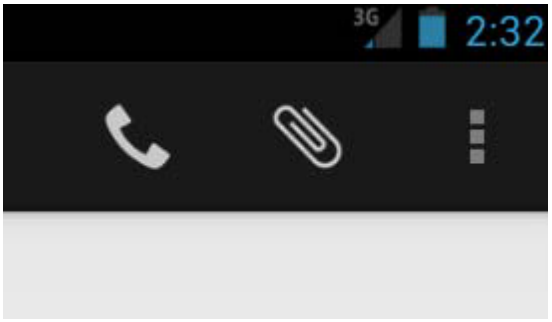
If the connection is a success you will see the following screenshot



3.1.10 3G Dial-up and Messaging

The Tiny4412 now can work with the ZTE MF210 module (WCDMA). The driver is RIL provided.

In Android 4 you don't need to do any setup and you can just plug the MF210 module and power on then your 3G will be connected and a 3G icon will show up on the right top of the GUI.



Now you can open a browser and surf the internet or send short messages.

Note: we found several MF210 models. The ones we tested and their device IDs are the following two:

VID：19d2 PID: 0117
VID：19d2 PID: 2003

In addition the MF210 module needs to be hooked up to the miniPCIe interface which doesn't exist on the Tiny4412. FriendlyARM has developed a convertion board for this application.

Its internal design is as follows:

3.1.11 HDMI Output

Copy your video files to an SD card (note: they must be mp4 whose audio data is in "aac" format. We provide some test files in the "Test Video" directory) and go to your "Gallery" directory:

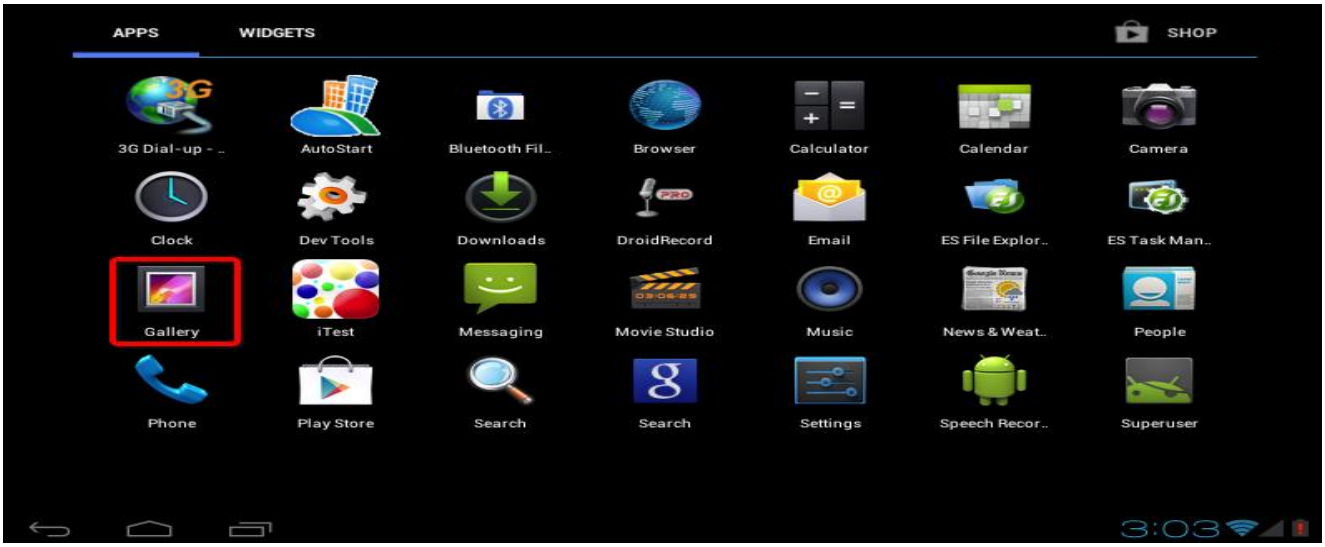
The Gallery will list all the available video files

Click on a video file it will be played. The max resolution is 1080p.

When connecting your board to a TV with an HDMI cable the audio and video output will be simultaneously output to your TV.

3.1.12 Play High Definition Video

Copy your video files to an SD card (note: they must be mp4 files in “aac” format. We provide some test files in the “Test Video” directory) and open “Gallery”:

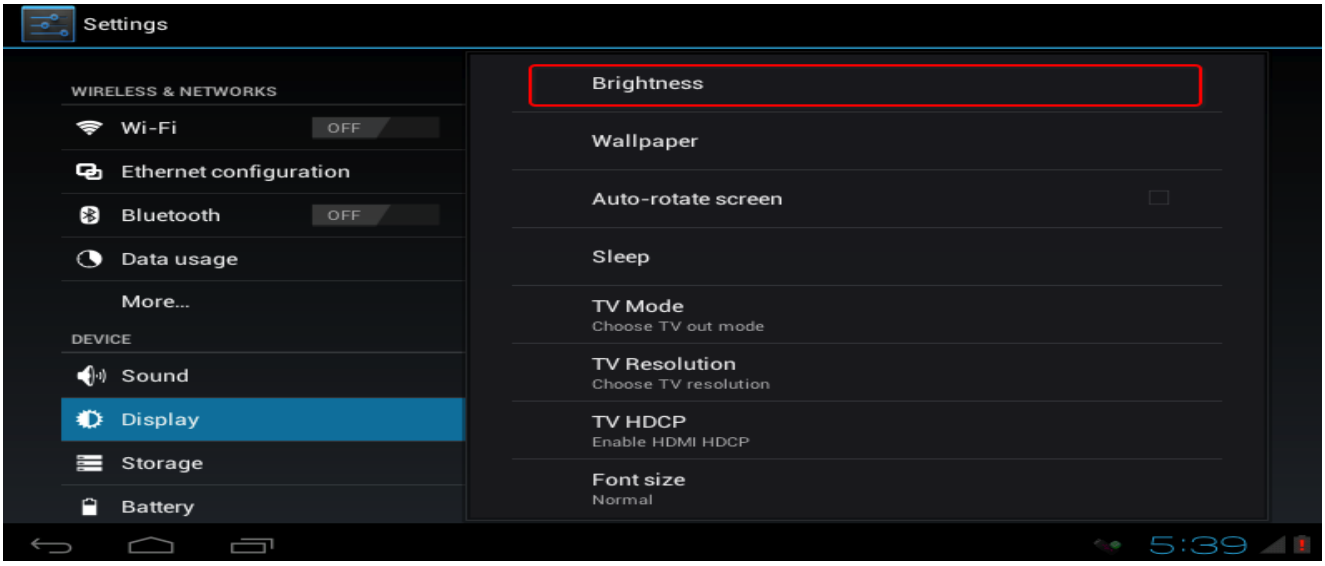


"Gallery" lists all the available video files:

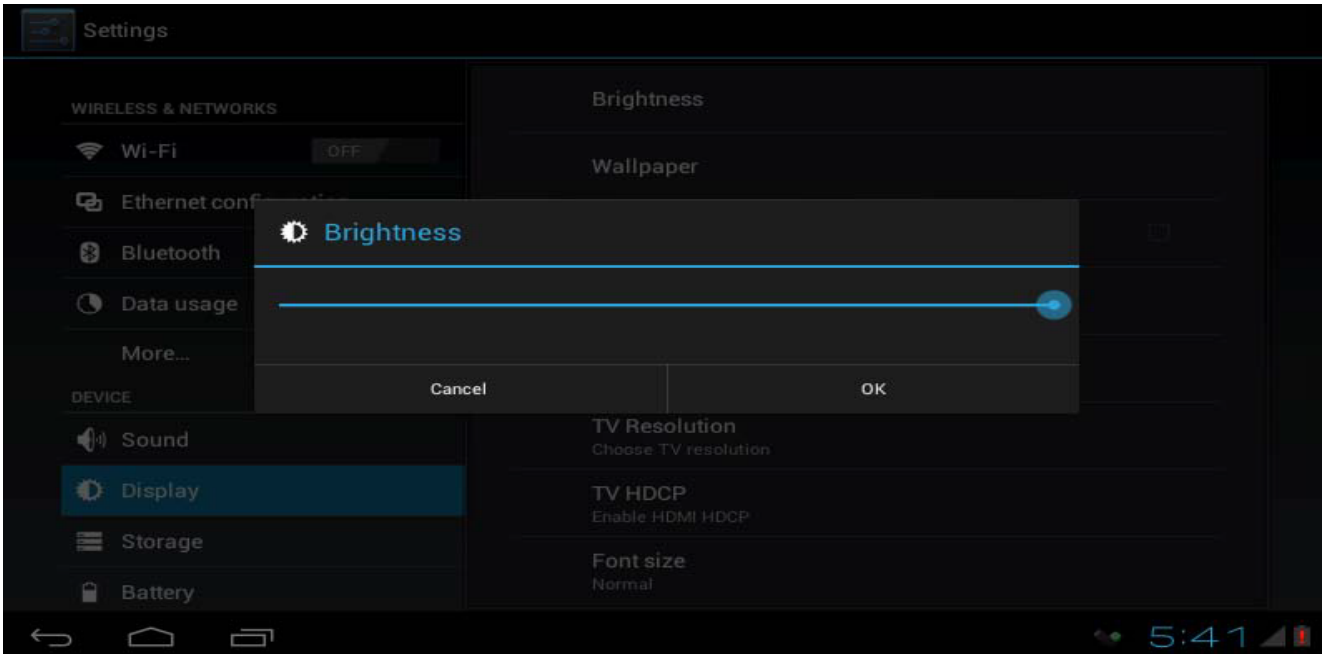
Click on a video file and play. The max resolution is 1080p.

3.1.13 Backlight Control

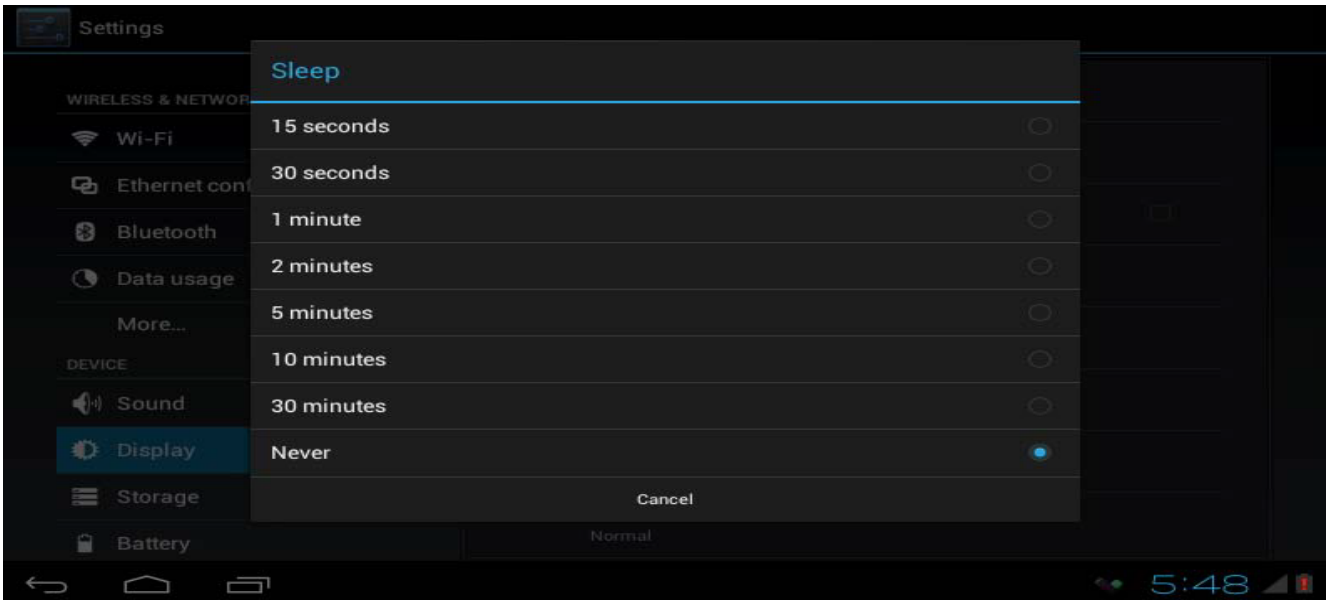
Maybe you have noticed that after the system boots the backlight will turn off gradually if the touch screen doesn't receive any touch. This is manipulated by the backlight control. Please go to "Settings" -> "Display" -> "Brightness".



Click on “Brightness”, open the backlight control window and you will be able to set the backlight’s brightness



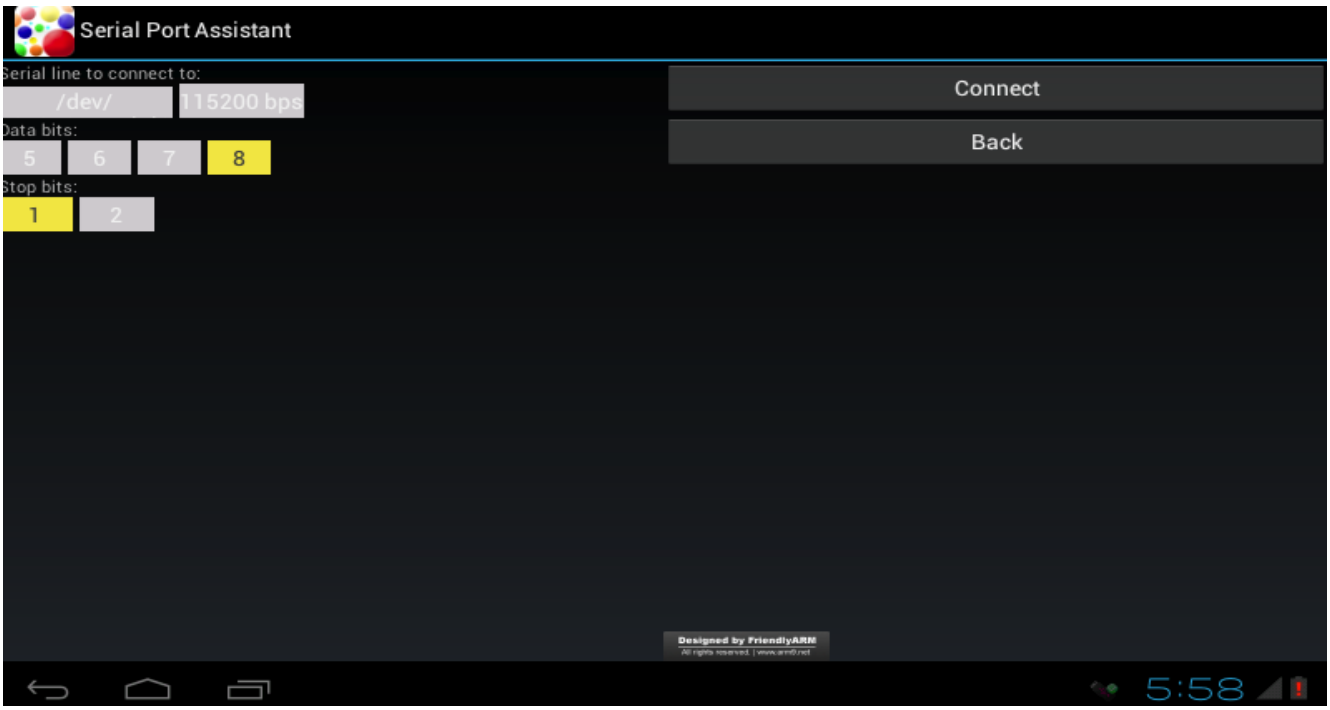
Go to “Display” -> “Sleep” to set its turn off time



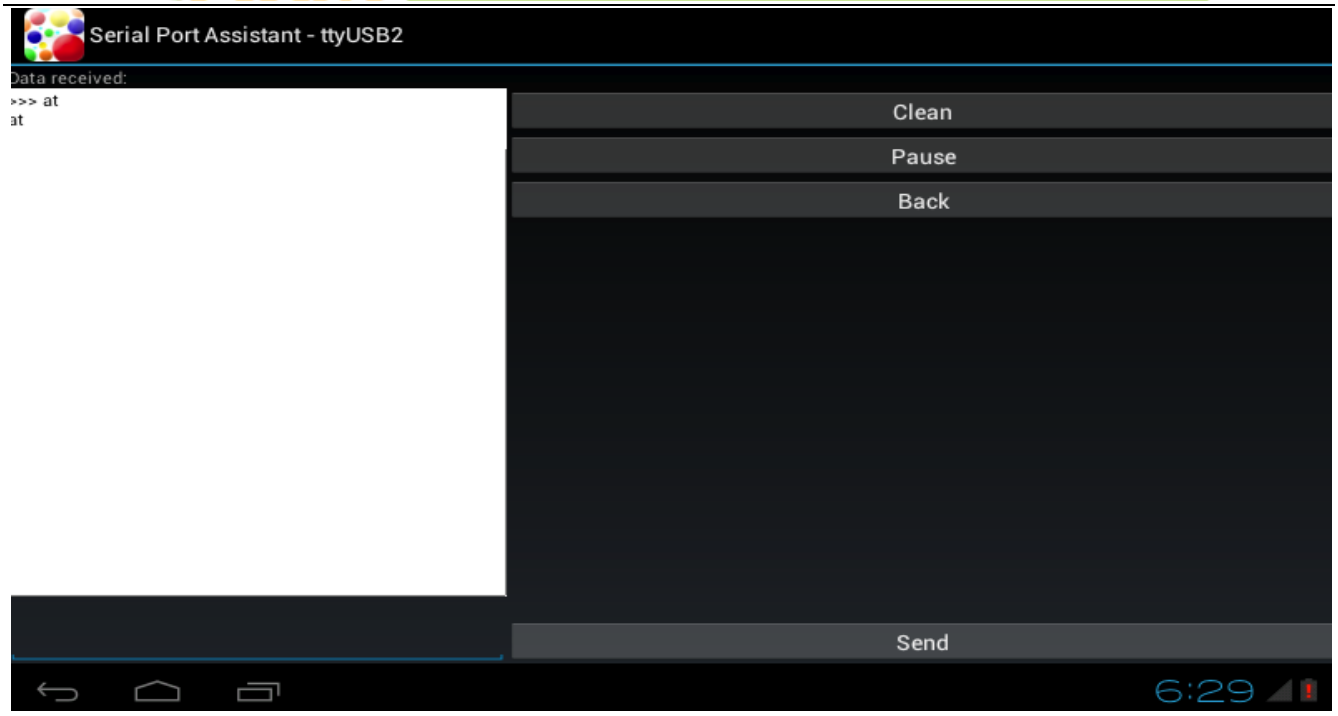
3.1.14 Serial Port Assistant

To launch our serial port assistant utility, you can click on the “iTest” icon

Click on “Serial Port Assistant” and you can set its parameters as follows:



After setup is done, click on “Connect” and if the connection is successful you will see the following messages from the serial port



To send data to the serial port, you can type your messages in the left text box and click on “send”. Clicking on “Pause” pauses message sending and “Clean” removes all the received messages

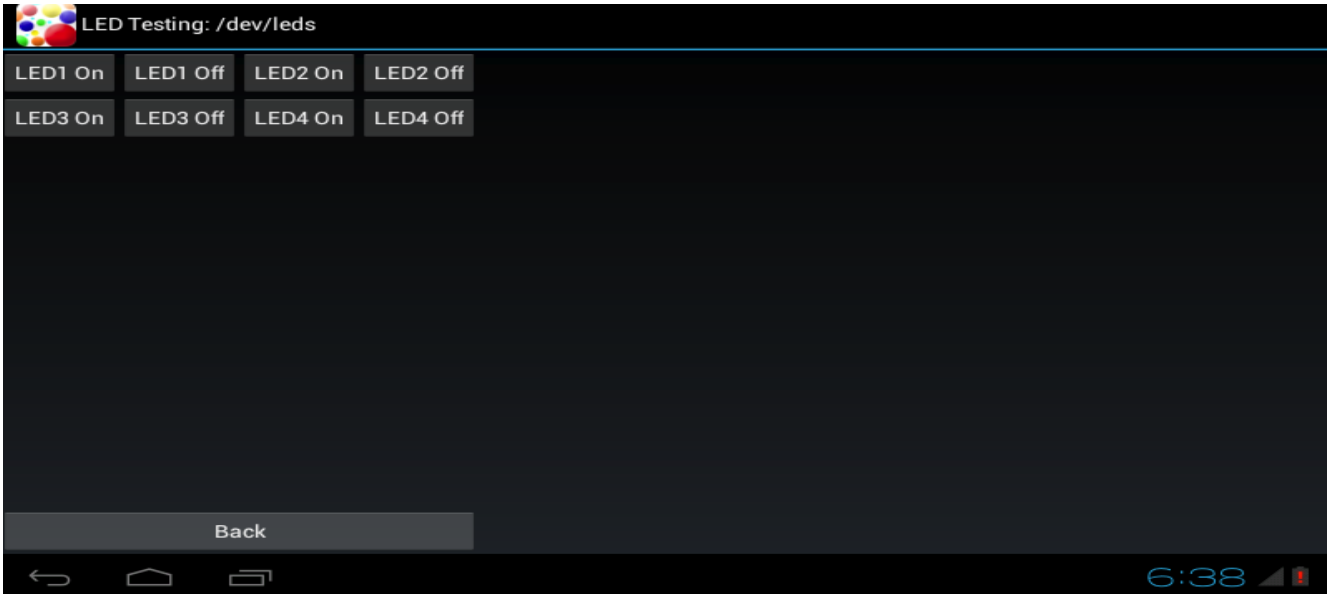
Note:

1) If a serial port cannot be accessed please try “fuser” to check if it is occupied by other applications
2) If this serial port still cannot be opened please command “ls -l” to check its access right and then try “chmod 777” to change its right and try again
3) By default “s3c2410_serial0” is associated with COM0 which is for debugging therefore it is not commended to use this serial port

3.1.15 LED Testing

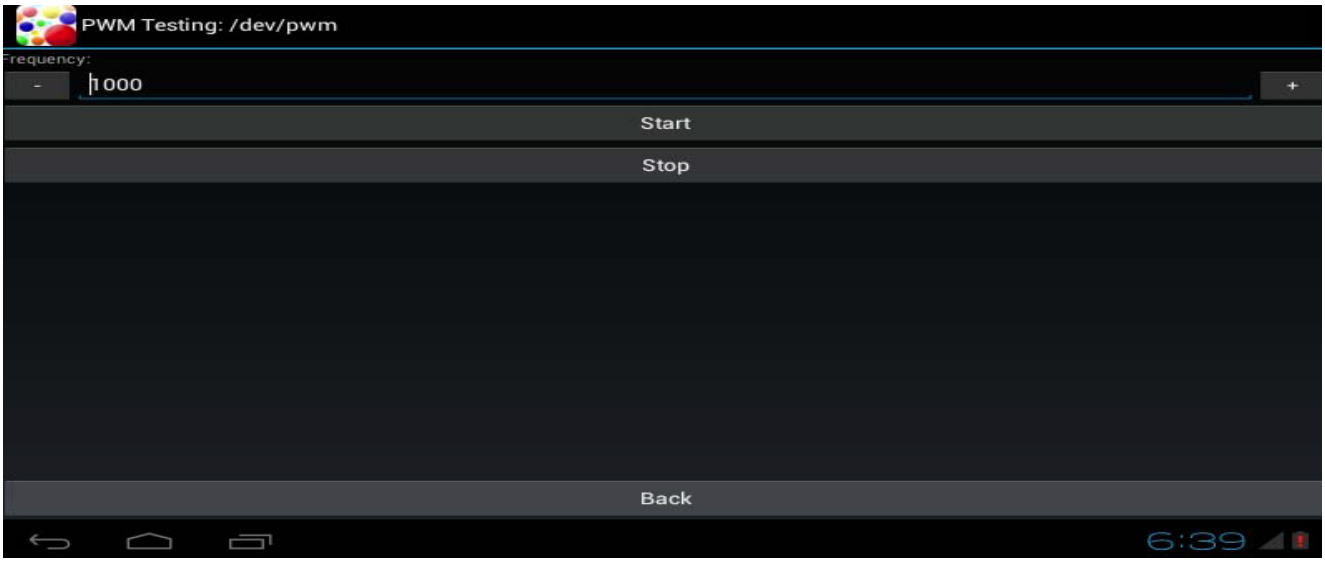
To test LEDs, please click on the “iTest” icon. Click on “LED Testing” and you will

see the following window and be able to test LEDs by clicking on those buttons:



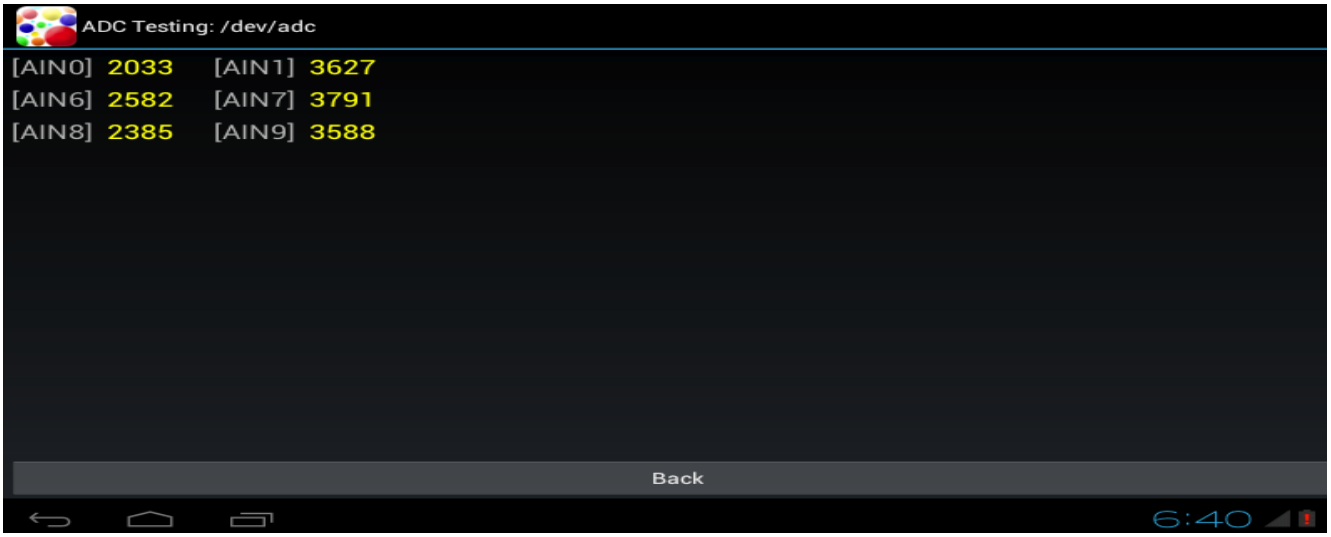
3.1.16 PWM Buzzer

To test PWM, please click on the iTest icon. Click on “PWM Testing” you will see the following window. On the window you can type a frequency and “start” or adjust the frequency by clicking on “+” and “-”. To stop it you can click on “stop”.



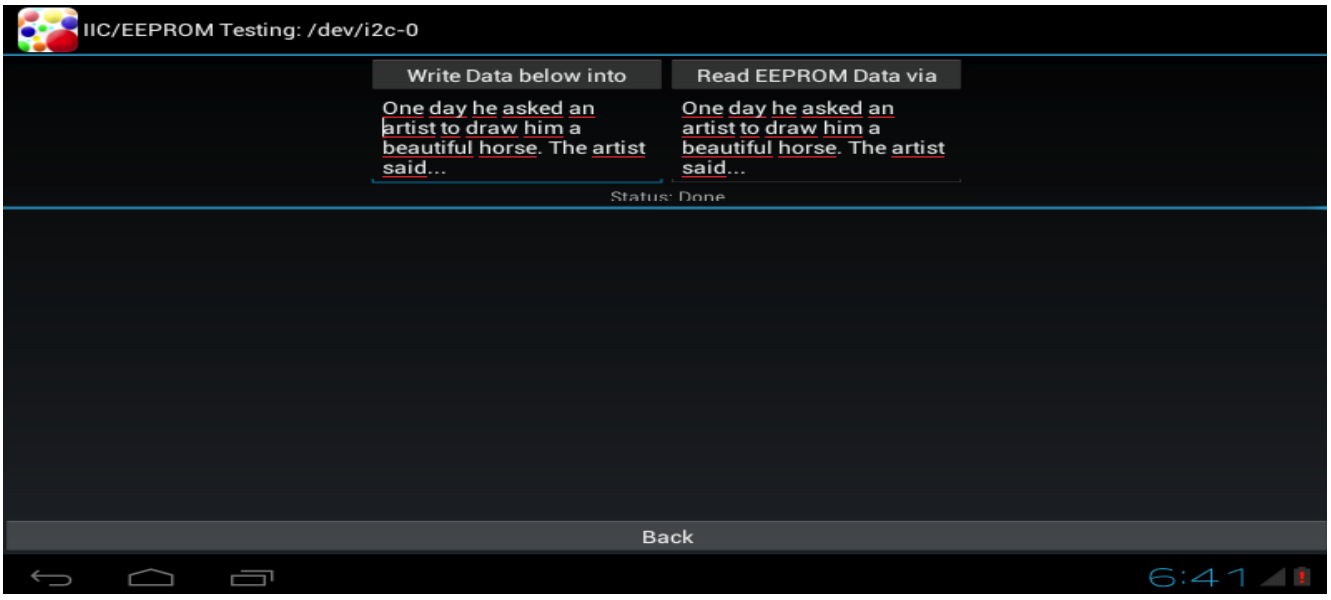
3.1.17 ADC Testing

To test ADC, please click on the “iTest” icon. Click on “A/D Convert” you will see the following window



3.1.18 I2C-EEPROM Testing

To test “I2C-EEPROM” please click on the “iTest” icon. Click on “IIC/EEPROM Testing” you will see the following window



Click on “Write Data below into EEPROM” to write your data on the left to “EEPROM” and then clickon “Read EEPROM Data via IIC” to read it from EEPROM to the right area

3.2 Set up Android Development Environment

What we mean by “set up Android Development Environment” includes the following three steps:

Step1: install Ubuntu 12.04.2 (64bit)

Step2: install Android SDK on Ubuntu12.04.2

Step3: install Android source code, kernel code and cross compiler

Step4: install fastboot

3.2.1 Install Ubuntu12.04.2 64bit

Please download a Ubuntu12.04.2 64bit installation image. Its official website is: http://releases.ubuntu.com/precise/.

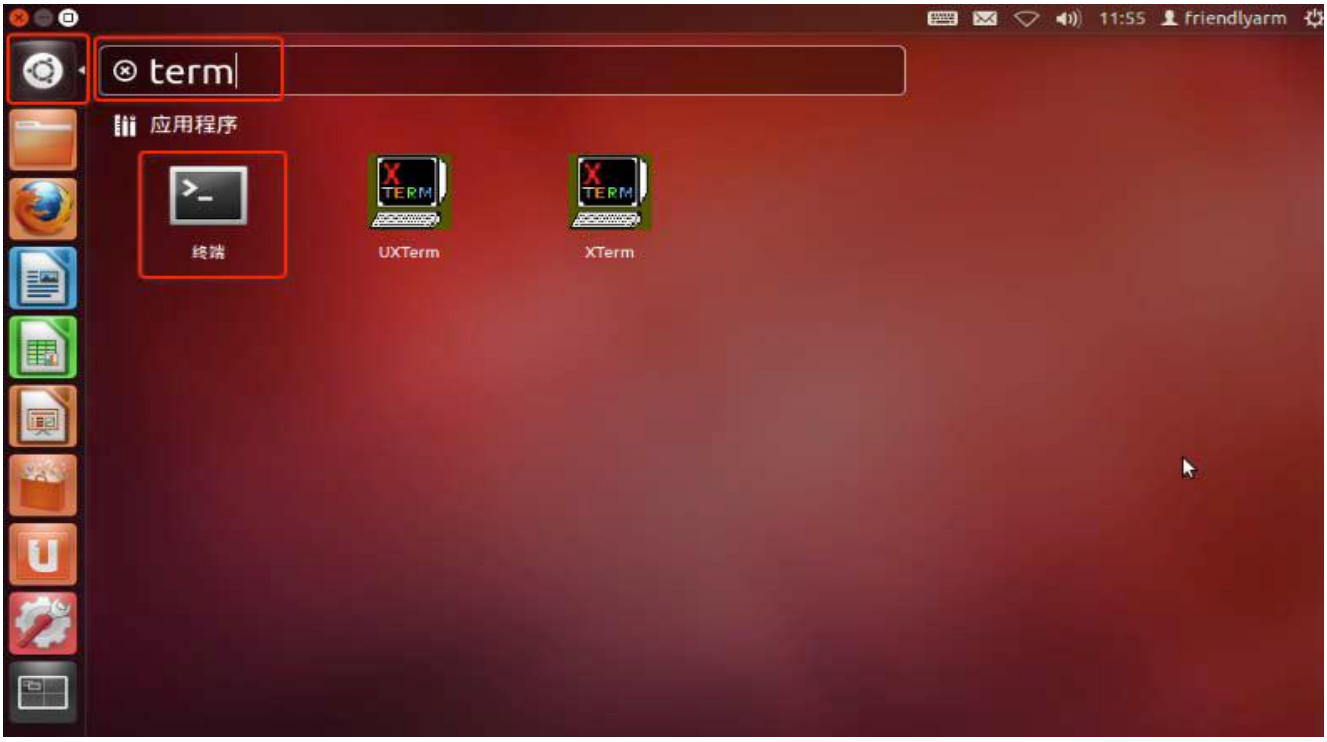
The file name is **ubuntu-12.04.2-desktop-amd64.iso.** Then you can install it on your PC.

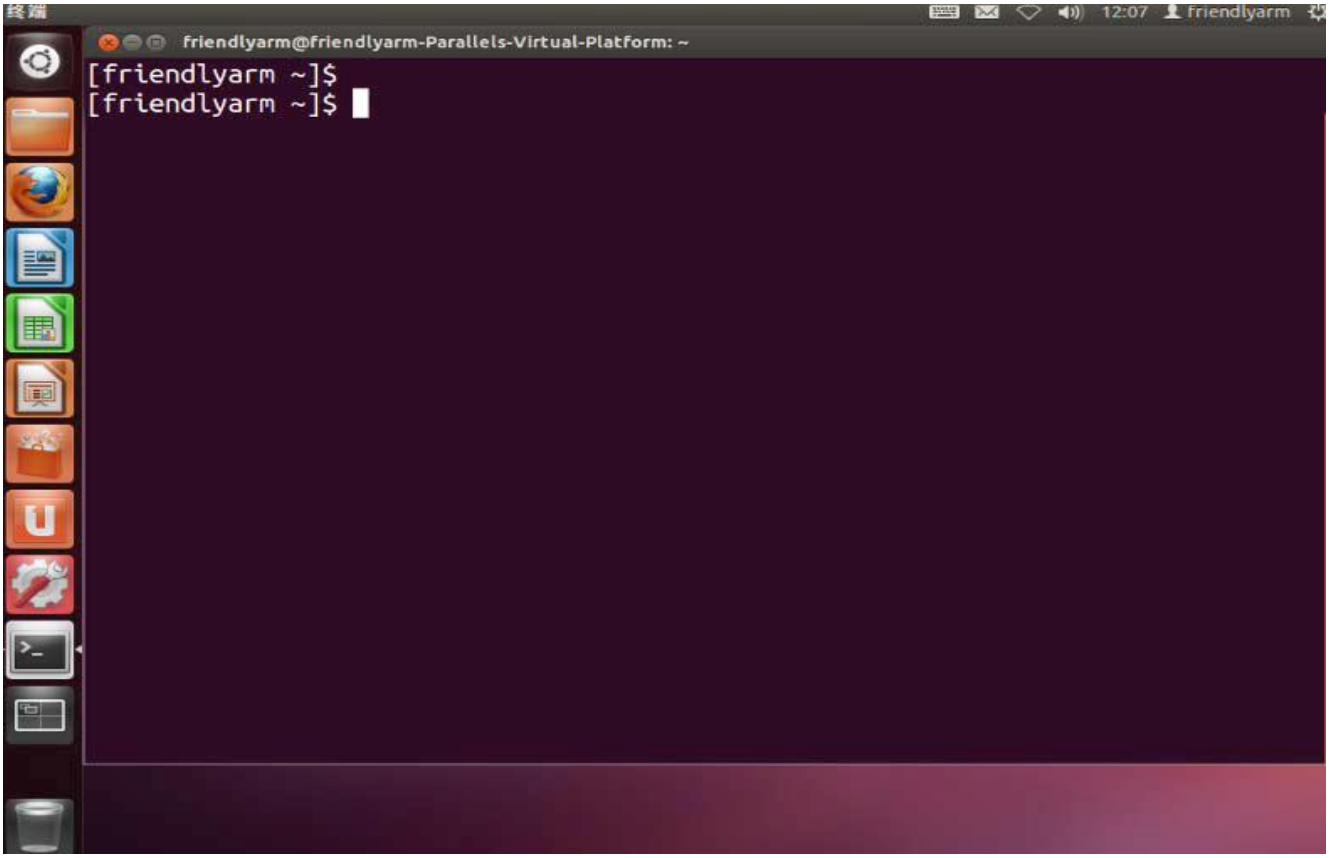
3.2.2 Configure Ubuntu

3.2.3 Login as Root

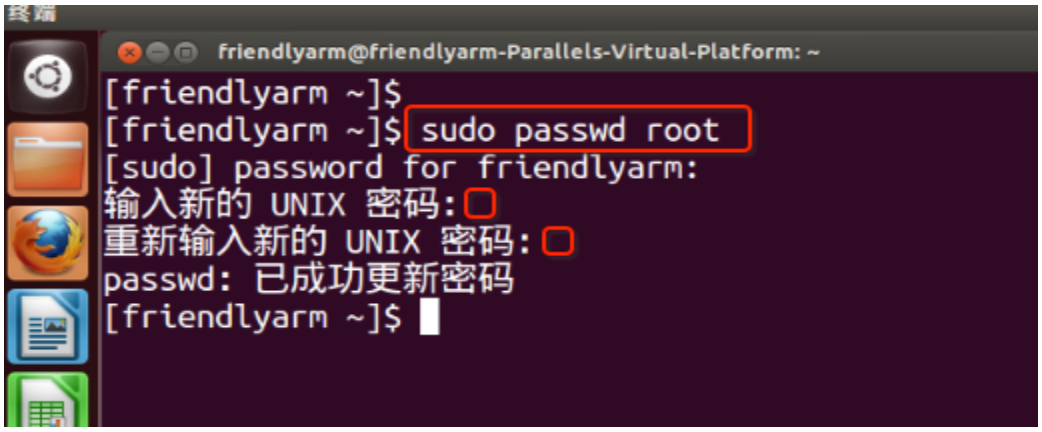
We recommend users to login Ubuntu as root. Please follow the steps below to do

that in a commandline terminal.



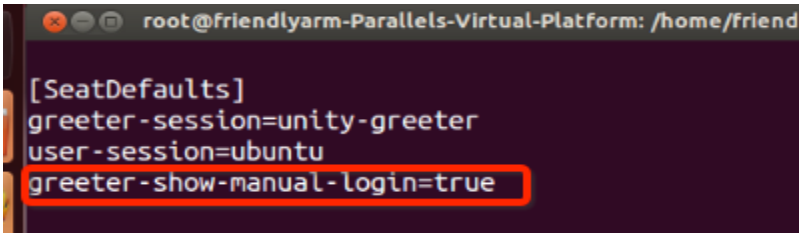


Firstly you need to set up a password for root. Please run "sudo passwd root" and type a password twice

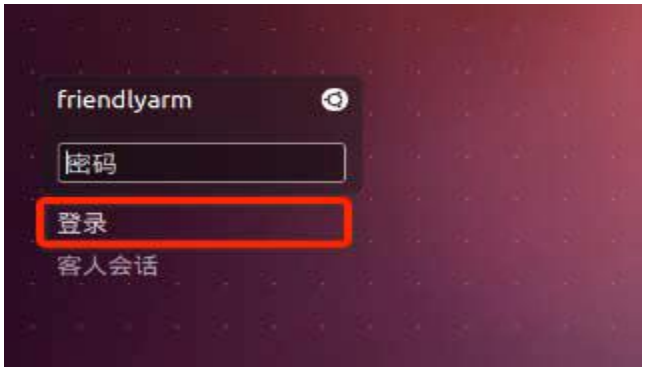


Run "su root" to login as root. Run "cp -p /etc/lightdm/lightdm.conf /etc/lightdm/lightdm.conf.bak" to back up your lightdm file. Then run "vi /etc/lightdm/lightdm.conf" to open it

Add one line at the end of the file "**greeter-show-manual-login=true**"



Save the file, reboot the system and click on "login"



Then type “root”, enter and type the password.

3.2.4 Install Android Tools

By default Ubuntu’s default installation doesn’t have software development tools therefore users need to install them

Firstly please copy the “\tools\ubuntu” directory from the 4412DVD to “tmp”. If you already have an iso file please follow the steps below

```
# mkdir –p /mnt/iso
# mount –o loop Tiny4412-20130707.iso /mnt/iso
# cp /mnt/iso/tools/ubuntu /tmp/ -a
```

Secondly please install jdk6

```
# cd /tmp/ubuntu/jdk6/
# chmod 755 install-sun-java6.sh
# ./install-sun-java6.sh
```

Lastly please run the install-devel-packages.sh script and install necessary software components

```
# cd /tmp/ubuntu/
# chmod 755 install-devel-packages.sh
# ./ install-devel-packages.sh
```

3.2.5 Setup Android Compiler

We used arm-linux-gcc-4.5.1 and it by defauly supports armv7 command sets. The following steps will introduce how to build a compile environment.

Step 1: copy the compressed file “arm-linux-gcc-4.5.1-v6-vfp-yyyymmdd.tgz” in the shipped CD into a system’s directory, e.g “/tmp”, enter this directory and execute the following commands:

#cd /tmp
#tar xvzf arm-linux-gcc-4.5.1-v6-vfp-yyyymmdd.tgz –C /

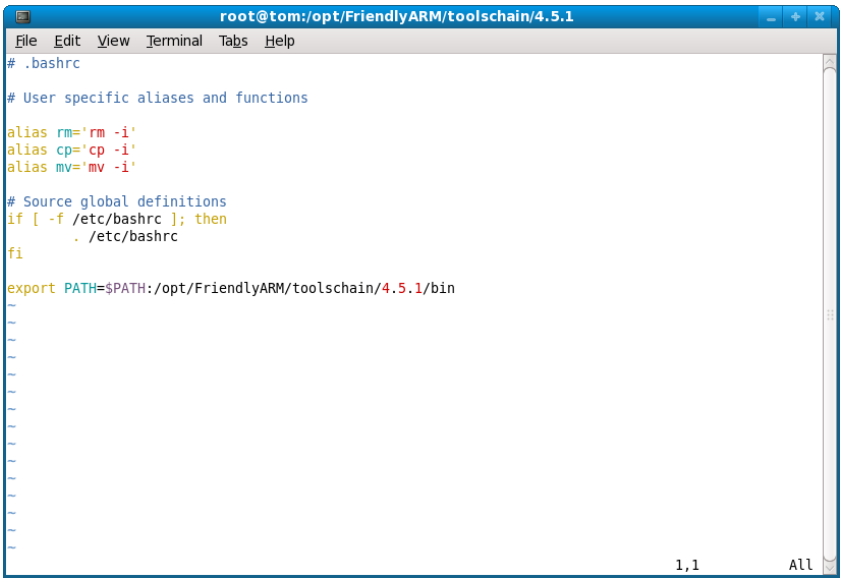
Note: there is a space after "C" and "C" is a capital letter.

These commands will install "arm-linux-gcc" in the "**/opt/FriendlyARM/toolschain/4.5.1**"

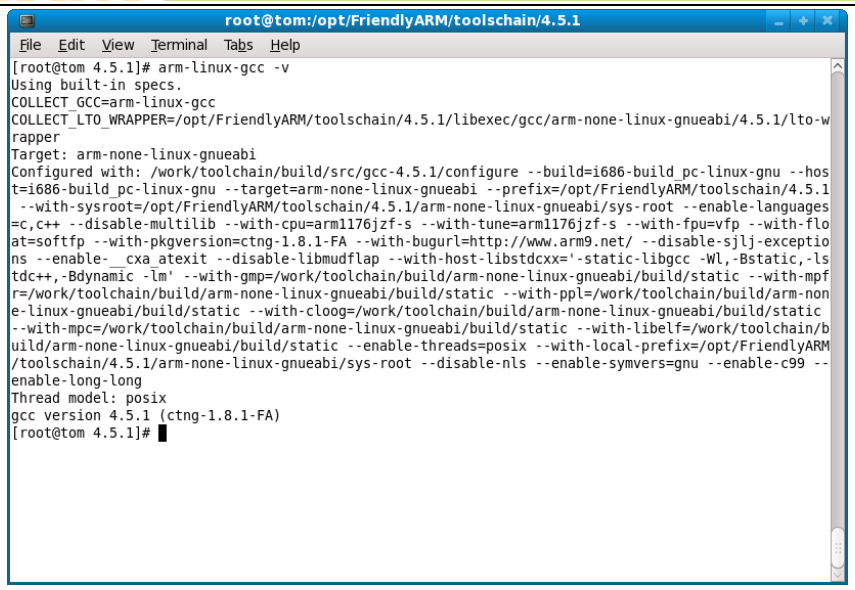
Step 2: run the command below to add the compiler's path to system variables:

#gedit ~/.bashrc

This is to edit the "~/.bashrc" file (there is a "." before "bashrc"). Update the last line with "**export PATH=$PATH:/opt/FriendlyARM/toolschain/4.5.1/bin**" in the opened file, save and exit the file



Logout and login the system again (no need to reboot the system, just go to "start"-> "logout"), the above settings will take into effect. Type "arm-linux-gcc –v", if the messages depicted in the screen shot below appear, it indicates the compile environment has been set up successfully.



3.2.6 Uncompress Android Source Code and Install Application Utilities

Firstly, create a working directory: "/opt/FriendlyARM/tiny4412/android" by running the following command

#mkdir –p /opt/FriendlyARM/4412/android

All the source code in the following steps will be uncompressed in this working directory

(1) Get a Copy of Android Souce Code Package

Please copy the Android directory from your 4412 DVD to the tmp directory of your system. If you already have an iso file please run the commands below

```
# mkdir –p /mnt/iso
# mount –o loop Tiny4412-20130707.iso /mnt/iso
# cp /mnt/iso/Android /tmp/ -a
```

(2) Uncompress Android Kernel

Execute the commands below in "/opt/FriendlyARM/tiny4412/android"

```
#cd /opt/FriendlyARM/tiny4412/android
#tar xvzf /tmp/android/ linux-3.5-yyyymmdd.tgz
```

This will create a "linux-3.5" directory which contains a complete copy of source code

Note: yyyymmdd is the date when we released it

(3) Uncompress Android4.2.1 System

Execute the commands below in "/opt/FriendlyARM/tiny4412/android"

```
#cd /opt/FriendlyARM/tiny4412/android
#tar xvzf /tmp/android/android-4.2.1_r1-fs-YYYYMMDD.tar.gz
```

This will create an "Android-4.2.1_r1" directory

Note: yyyymmdd is the date when we released it.

3.3 Configure and Compile Linux Kernel

To compile a Linux3.5 kernel please follow the steps below

#cd /opt/FriendlyARM/tiny4412/android/linux-3.5
#cp tiny4412**_android_defconfig .config ; note: there is a "." before "config"**

You can run "make menuconfig" to configure the kernel and run "make" to compile":

#make

This will generate a zImage under "arch/arm/boot".

3.4 Create Android

Compiling Android may not be an easy task for beginners. Therefore we have a complete copy of the source code and three compiling scripts.

Script	Comment	Code Example
setenv	Set Android environment variables	. setenv ; note: there is a space after "."
gen-img.sh	Generates a system.img and a ramdisk-u.bin which are used with fastboot and SD card for flashing systems	./gen-img.sh

burn-img.sh	Flashes the board with image files. When the Tiny4412 works in the fastboot mode and connected to a PC via USB running this script will flash the board with "system.img" and "ramdisk-u.bin".	./burn-img.sh

To compile the Android source code (Android 4.2.1_r1) please run the following commands:

#cd /opt/FriendlyARM/tiny4412/android/ android-4.2.1_r1
#. setenv ;note: there is a space after "."
make

Tips: you can add a "-j" parameter after "make" to speed up your compilation if your PC is multi-core. For example if your PC is quad-core you can try "make –j4"

3.5 Create and Run File System

Please run the command below to generate a system.img and a ramdisk-u.img:

#./gen-img.sh

Running gen-img.sh will create a system.img and a ramdisk-u.img in Android's source code directory.

3.6 Access Hardware in Andorid

For users to fully use and acces the Tiny4412 hardware resources FriendlyARM

developed a library named “libfriendlyarm-hardware.so” which can be used to access and operate the hardware resources on the Tiny4412 including serial port, buzzer, EEPROM and ADC.

The iTest utility is based on this library and you can run the iTest utility to learn this library.

In this section we will describe how to use the libfriendlyarm-hardware.so library.

3.6.1 How to Use “libfriendlyarm-hardware.so”

FriendlyARM has included the libfriendlyarm-hardware.so library in Android. It is in the following directory:

```
vendor/friendly-arm/exynos4412/rootdir/system/lib/libfriendlyarm-hardware.so
```

On the Tiny4412 it is in the “/system/lib/libfriendlyarm-hardware.so” directory.

If you develop Android applications with Eclipse you can follow the steps below to call libfriendlyarm-hardware.so APIs:

1) Go to your Android program’s directory, create a “libs” directory, enter it and create an “armeabi” directory and copy “libfriendlyrm-hardware.so” to this directory.

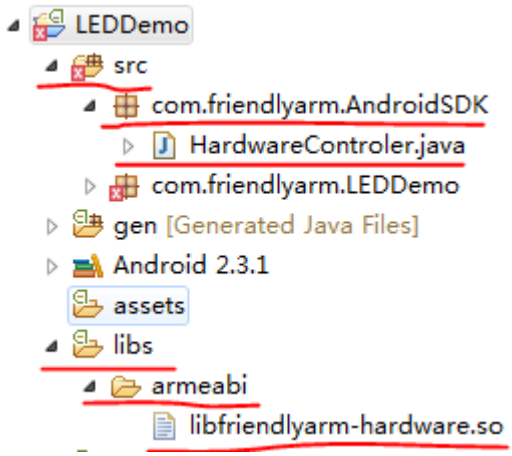
2) Go back to your program’s directory, enter the “src” directory and create a “com\friendlyarm\AndroidSDK” directory and create a “HardwareControler.java” file and type the following code:

```
package com.friendlyarm.AndroidSDK;
import android.util.Log;
```

```
public class HardwareControler
{
    /* Serial Port */
    static public native int openSerialPort( String devName, long baud, int dataBits,
int stopBits );

    /* LED */
    static public native int setLedState( int ledID, int ledState );

    /* PWM */
    static public native int PWMPlay(int frequency);
    static public native int PWMStop();

    /* ADC */
    static public native int readADC();
    static public native int readADCWithChannel(int channel);
    static public native int[] readADCWithChannels(int[] channels);


    /* I2C */
    static public native int openI2CDevice();
    static public native int writeByteDataToI2C(int fd, int pos, byte byteData);
    static public native int readByteDataFromI2C(int fd, int pos);

    /* IO */
    static public native int write(int fd, byte[] data);
    static public native int read(int fd, byte[] buf, int len);
    static public native int select(int fd, int sec, int usec);
    static public native void close(int fd);
    static public native int getBoardType();


    static {
        try {
        System.loadLibrary("friendlyarm-hardware");
        } catch (UnsatisfiedLinkError e) {
            Log.d("HardwareControler", "libfriendlyarm-hardware library not
found!");
        }
    }
```

```
}
```

Start Eclipse and select your project list and “Refresh” it now you will see the following information:



To use the HardwareControler APIs you need to add the following line to your code which introduces the HardwareControler class:

```
import com.friendlyarm.AndroidSDK.HardwareControler;
```

Now you will be able to call HardwareControler APIS and we will show you some examples in the following sections

3.6.2 libfriendlyarm-hardware.so APIs

We will list some HardwareControler APIs here.

3.6.2.1 Serial Port API

Serial Port APIs:

APIs	Parameters and Return Value	Comment
int openSerialPort(String devName, **long** baud, **int** dataBits, **int** stopBits)	devName: device name, the following devices are available: /dev/s3c2410_serial1 /dev/s3c2410_serial2 /dev/s3c2410_serial3	Open a serial device and return its file descriptor.

	/dev/ttyUSB0 /dev/ttyUSB1 /dev/ttyUSB2 /dev/ttyUSB3 baud: baud rate dataBits: in general we use 8 stopBits: in general we use 1 Return Value: When the device is opened successfully it will return a file descriptor which can be used to read、write and select the device otherwise it will returns -1.	
int write(**int** fd, **byte**[] data)	fd: file descriptor data: data to write to the device Return Value: When the operation succeeds it will return the number of characters written otherwise it will return -1.	Write data to an opened device.
int read(**int** fd, **byte**[] buf, **int** len)	fd: file descriptor buf: data buffer len: number of characters to read Return Value: When the operation succeeds it will return the number of characters read otherwise it will return -1. If before the read function is called the file pointer already reaches the end of the device it will return 0	Read data from an opened device.
int select(**int** fd, **int** sec, **int** usec)	fd: file descriptor sec: seconds allowed to wait usec: useconds allowed to wait(1ms = 1000us) Return Value: If the device has data it will return 1 otherwise it will return 0. If this operation fails it will return -1.	Query whether an opened device has data for reading.
void close(**int** fd)	fd: file descriptor Return Value: No	Close a device

Notes:

Firstly you need to open a serial device with “openSerialPort”, then your can call “select” to query if is has available data. When it has data you can call “read” to read data.

To write data to a device you can call “write”. If you don’t need to use a device remember to “close” it.

3.6.2.2 LED APIs

LED APIs:

APIs	Parameters and Return Value	Comment
int setLedState(**int** ledID, **int** ledState)	ledID： LED you want to access (0~3) ledState: 1 is on， 0 is off Return Value: If this operation succeeds it will return 0 otherwise it will return -1	Open an LED

3.6.2.3 PWM APIs

PWM Buzzer APIs:

APIs	Parameters and Return Value	Comment
int PWMPlay(**int** frequency);	frequency: frequency of sound Return Value: If this operation succeeds it will return 0 otherwise it will return -1	Play a Buzzer with the specified frequency
int PWMStop();	Return Value: If this operation succeeds it will return 0 otherwise it will return -1	Stop a buzzer

3.6.2.4 ADC APIs

ADC APIs:

APIs	Parameters and Return Value	Comment
int readADC()	Return Value: If this operation succeeds it will return the conversion result otherwise it will return -1	Read an ADC conversion result
int readADCWithChannel(int channel)	Channel: specified channel number: 0, 1, 4 and 5 Return Value: If this operation succeeds it will return the conversion result otherwise it will return -1	Read a specified ADC conversion result
int readADCWithChannel(int[] channels)	Channel: an array of specified channel numbers Return Value: If this operation succeeds it will return multiple conversion results otherwise it will	Read multiple specified ADC conversion results

	return -1	

3.6.2.5 EEPROM APIs

EEPROM APIs:

APIs	Parameters and Return Value	Comment
int openI2CDevice();	Return Value: If this operation succeeds it will return an IIC file descriptor otherwise it will return -1.	Open an IIC device and return a file descriptor. After an IIC device is opened successfully you can call "writeByteDataToI2C" and "readByteDataFromI2C" to operator the EEPROM.
int writeByteDataToI2C(**int** fd, **int** pos, **byte** byteData);	fd: file descriptor pos: position where data to be written (0~255) byteData: data to be written Return Value: If this operation succeeds it will return the number of characters written otherwise it will return -1.	Write data to EEPROM (one byte on each write operaration). Note: this operation is time consuming. It costs about 10 ms.
int readByteDataFromI2C(**int** fd, **int** pos);	fd: file descriptor pos: position where data to be read (0~255) Return Value: If this operation succeeds it will return data it reads otherwise it will return -1. If before this function is called the file pointer has reached the end of the device it will return 0. The return value's type is int you need to convert it to a byte value.	Read data from EEPROM. Note: this operation is time consuming. It costs about 10 ms.
void close(**int** fd)	fd: file descriptor Return Value:No	Close a device

Notes:

First you need to open a serial device with "openI2CDevice", then your can call "writeByteDataToI2C" to write data and "readByteDataFromI2C" to read data. These operations are time consuming and will usually cost 10ms therefore it is better to call them in a new thread.

The EEPROM device can store 256 bytes data so the position parameter's value ranges from 0 to 255 and each time it can only read/write one byte

If you don’t need to operate the device you need to “close” it.

3.6.3 Examples

In the “Android” directory in the shipped DVD there is a LED Demo program you can open it in Eclipse and learn how to use the libfriendlyarm-hardware.so library.You can debug, download and run it on the board via the shipped miniUSB cable.

4. Linux Navigation

4.1 Linux GUIs

The Linux image we prepare for the Tiny4412 includes Qtopia2.2.0, QtE4.7 and Qt Extended 4.4.3. Users can switch among all these three GUIs freely. By default the Linux GUI is Qtopia 2.2.0.

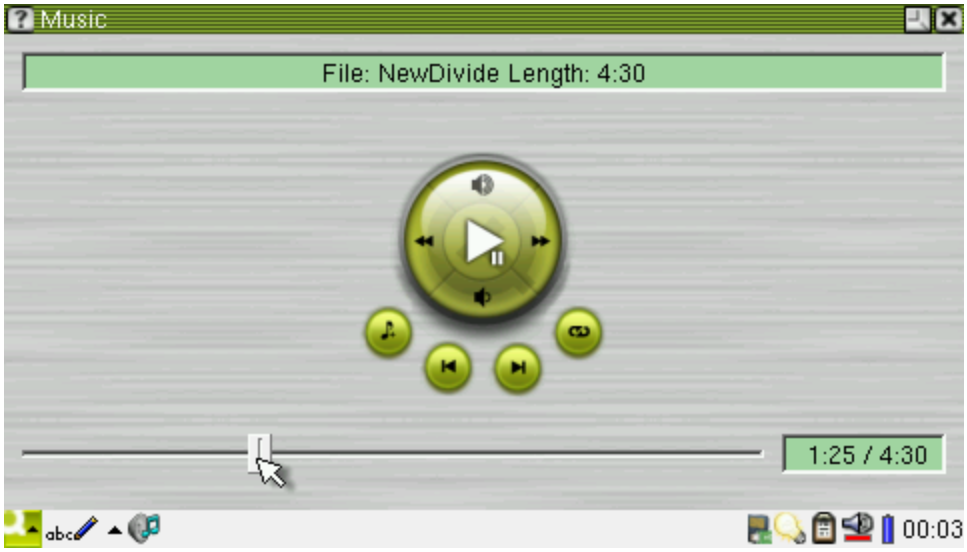
To get the latest QtE, please go to http://qt.nokia.com/.

4.1.1 Linux Main Window

After Linux loads you will see the following main window

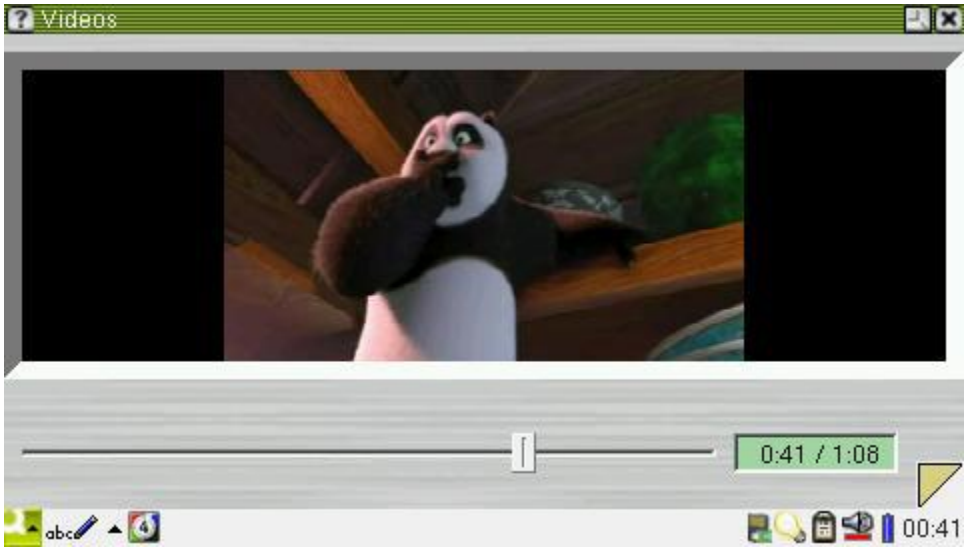
4.1.2 Play MP3

Go to the “Application” page, click on the “music” icon, select an mp3 and click on “play”.



4.1.3 Play Video

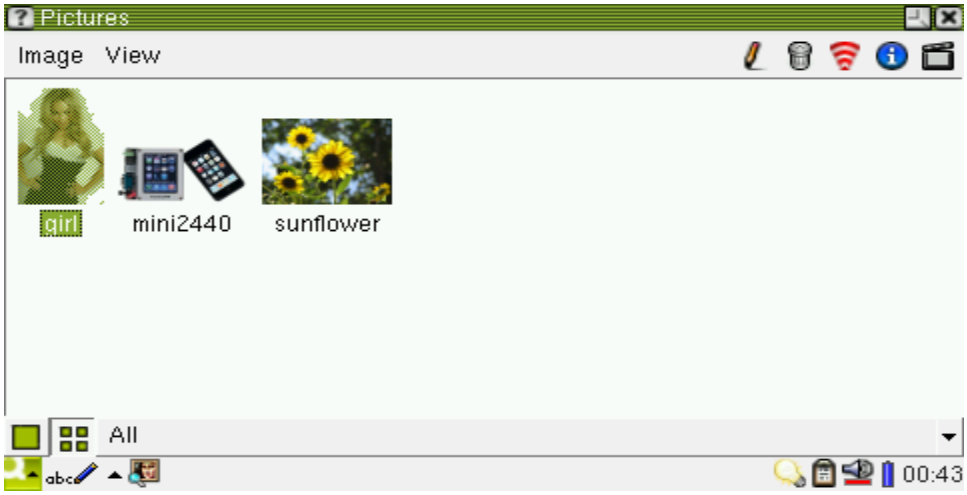
Go to the “Application” page, click on the “video” icon, select a video file and click on “play”. This player can fluently play H.264/H.263/Mpeg4 files.



4.1.4 Image Viewer

Go to the Application page, click on the “pictures” icon and you will be able to

browse pictures



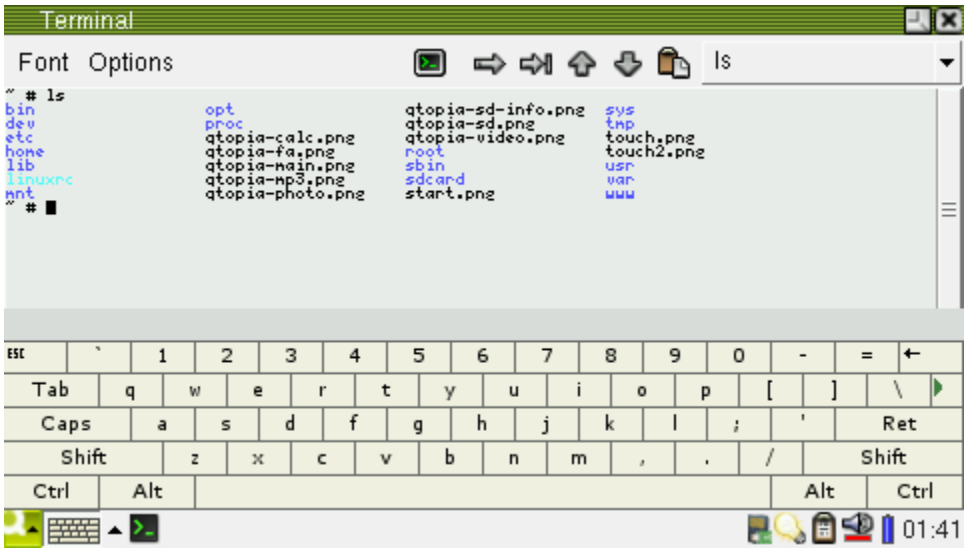
4.1.5 Calculator

Go to “Applications” and click on the calculator icon. You can select “Simple”, “Fraction”, “Scientific” and “Conversion”.

4.1.6 Terminal

Go to “Applications”, click on the terminal icon and you will be able to type Linux

commands.



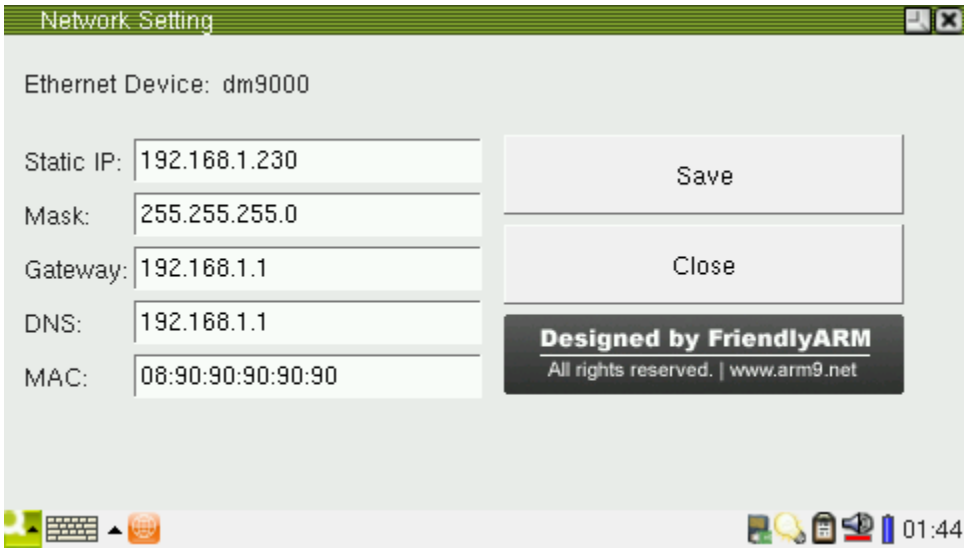
4.1.7 File Manager

Go to “FriendlyARM”, click on the file manager icon and you will see your system’s file structure:

4.1.8 Network Setting

Go to “FriendlyARM”, click on the network setting and you will be able to see the

following screenshot



You can set your network parameters and “save” it to the “/etc/eth0-setting”.

4.1.9 Wireless Network

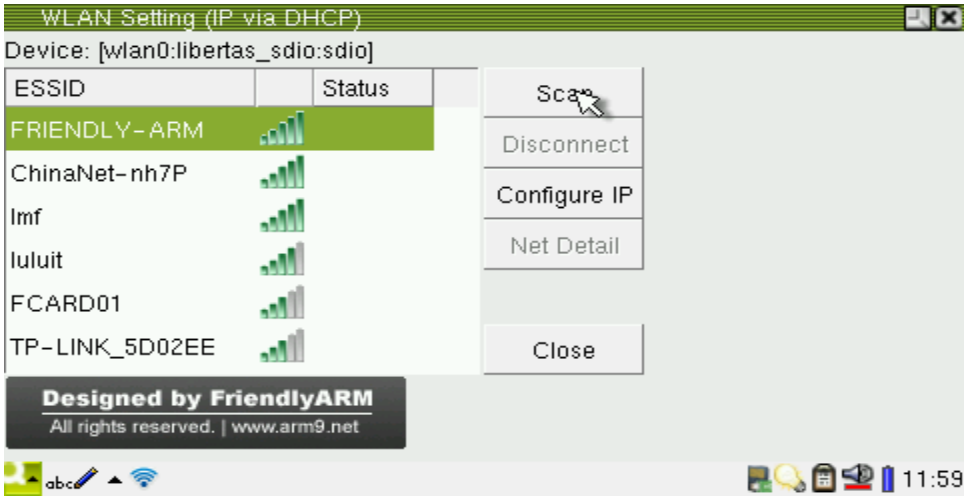
This section will introduce how to configure the SD WiFi and USB WiFi.

4.1.9.1 Wireless Utility

Go to the “FriendlyARM” page, click on the wireless setting icon

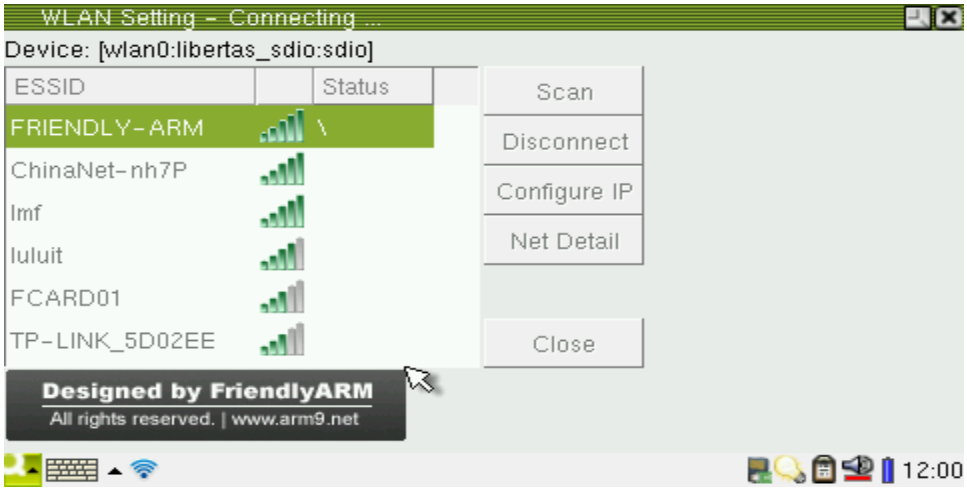
4.1.9.2 Wireless AP

After launching the setting utility it will automatically search for an AP and list all SSIDs and their signal strengths.

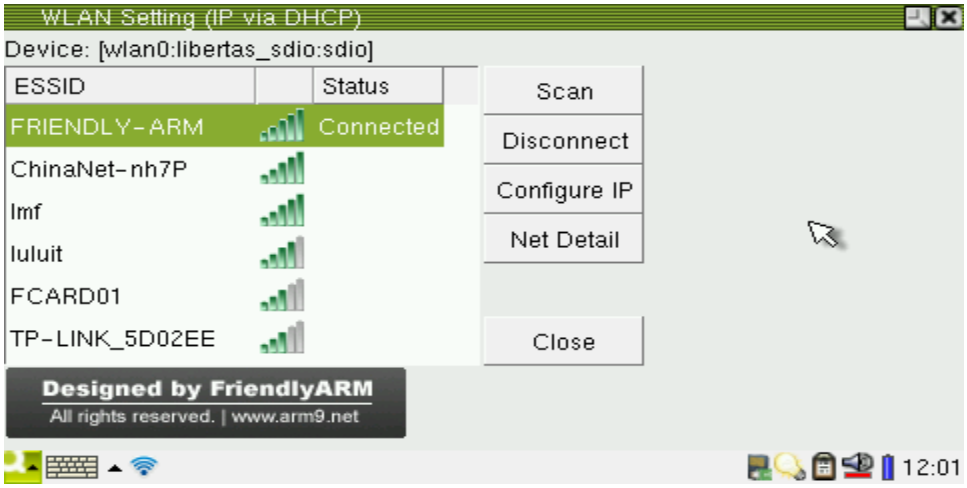


After an AP is found to connect to it you can click on its ESSID and input its password

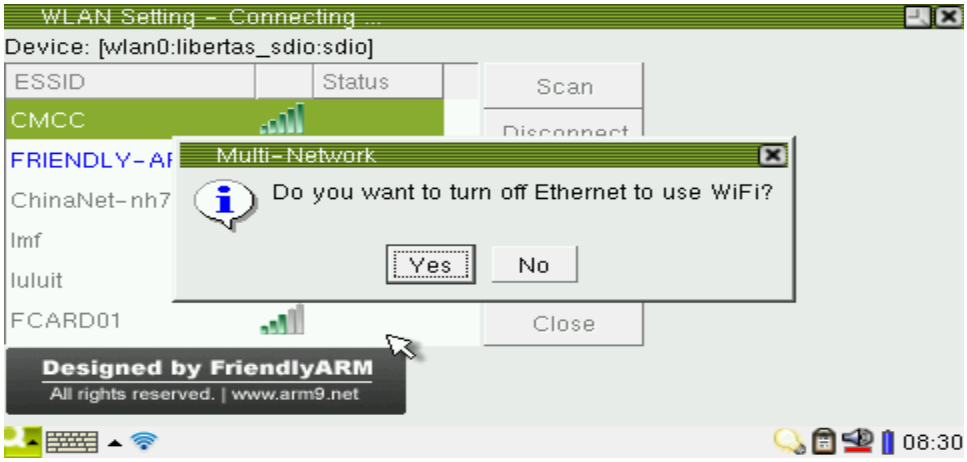
Click on “connect”



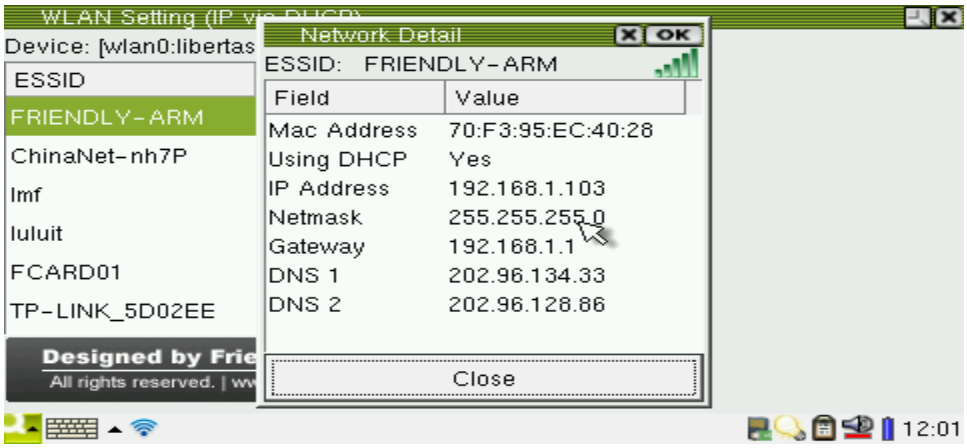
If the connection is successful it will show "Connected"



If you started the Ethernet before you start the wireless you will see the following dialog which prompts you to close the Ethernet. You need to close the Ethernet.



Click on "Net Detail" you will see the wireless network's details

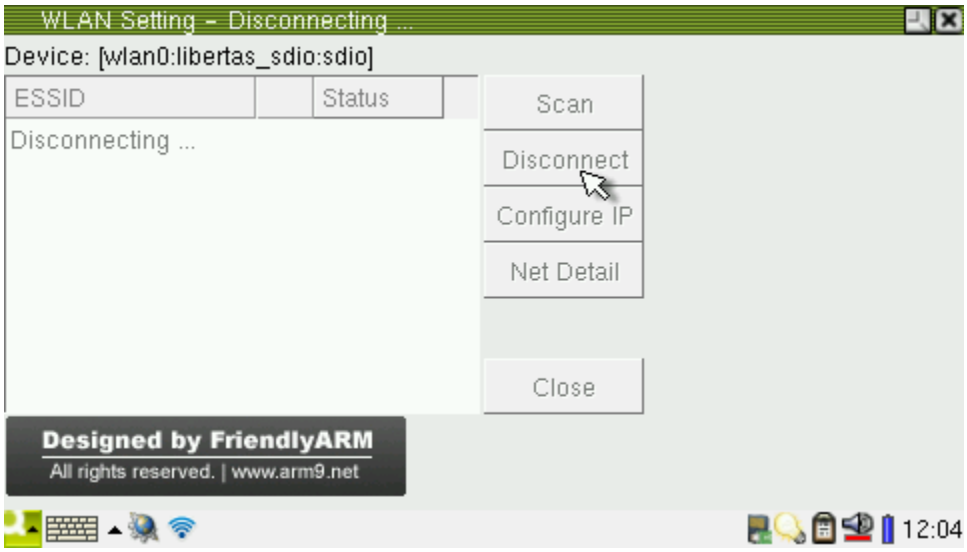


After your connection is successful, click on "close" to minimize the utility

Now you can surf the internet

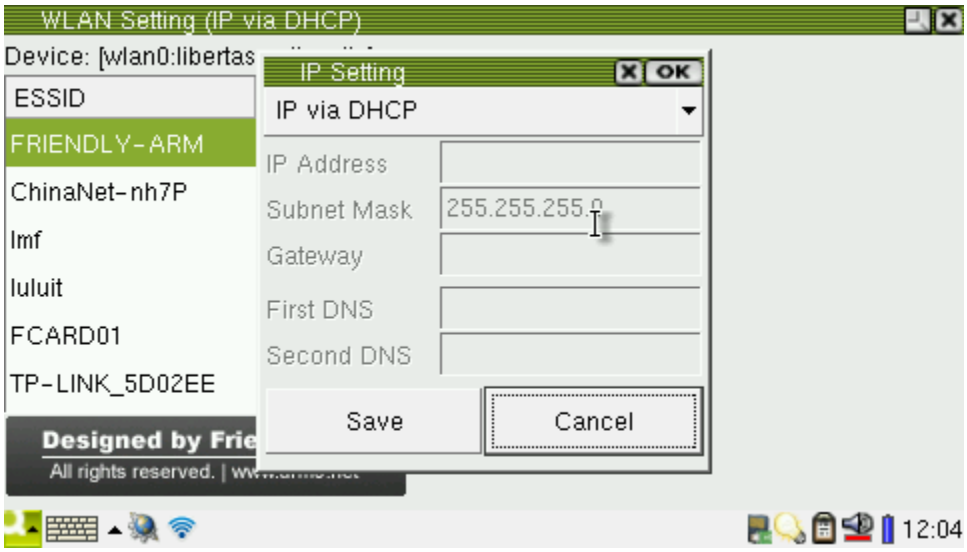
4.1.9.3 Disconnect Wireless Network

To disconnect the wireless network you can just click on "Disconnect"



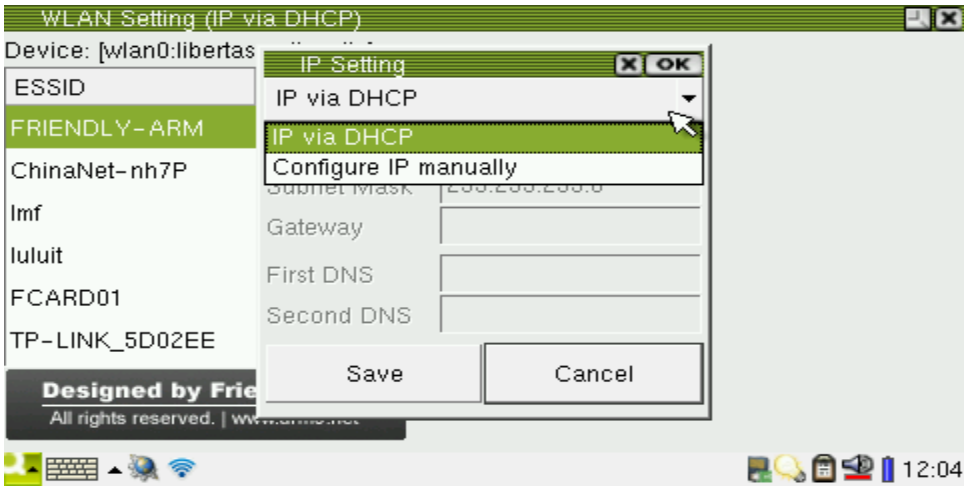
4.1.9.4 IP Configuration

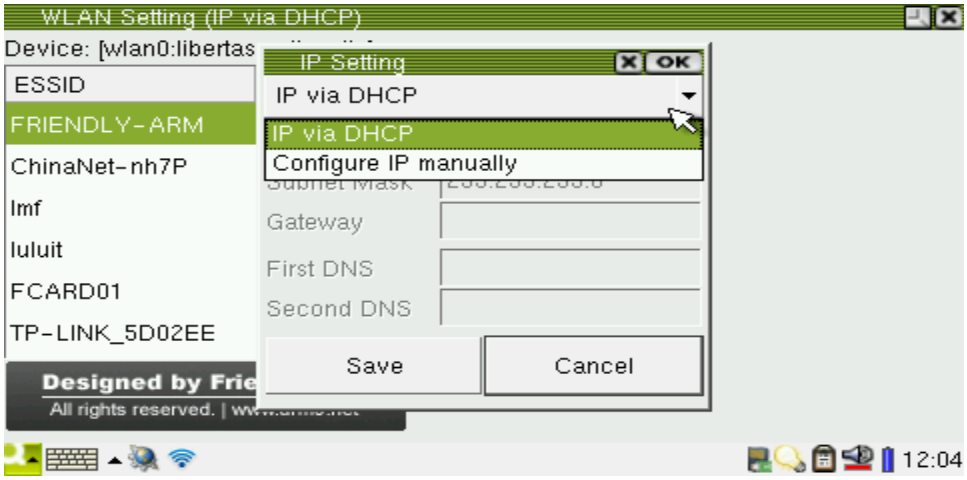
On the wireless utility window click on "Configure IP" you will see the following dialog:



Click on the "IP Setting" pull-down list you will be able to select "DHCP" or

"Configure IP Manually"





4.1.10 WiFi AP

To start the WIFI AP function you need to use the RT8192 WiFi module.

Note: the router function hasn't been implemented.

Go to "FriendlyARM", click on the network setting and you will be able to see the following screenshot

Click on the the icon you will see the following window:

The fields on the window are defined as follows:

IP	IP Address
SSID	Access point
Password	Password, at least 8 digits with WPA2 encoding
Channel	The default value is 8. Note: if your working area has a wireless router this value cannot be the same as the router's. Usually most popular routers such as D-LINK's value is 6.
Opt Mode	WiFi protocol. The default is 802.11g
Auto start at boot	Whether to set auto start the WiFi AP

After configuration is done click on "Apply" to save. Click on "Start" to start the WiFi AP. When it shows "Working" it means the board is now working under the WiFi AP mode. In our example the IP was 192.168.2.1

Now you can find the board from your mobile phone. In our example the SSID was

"arm9.net" we could find it on our mobile phone. Click on it and input the password to connect.

After the connection is successful you will find similar information shown as below

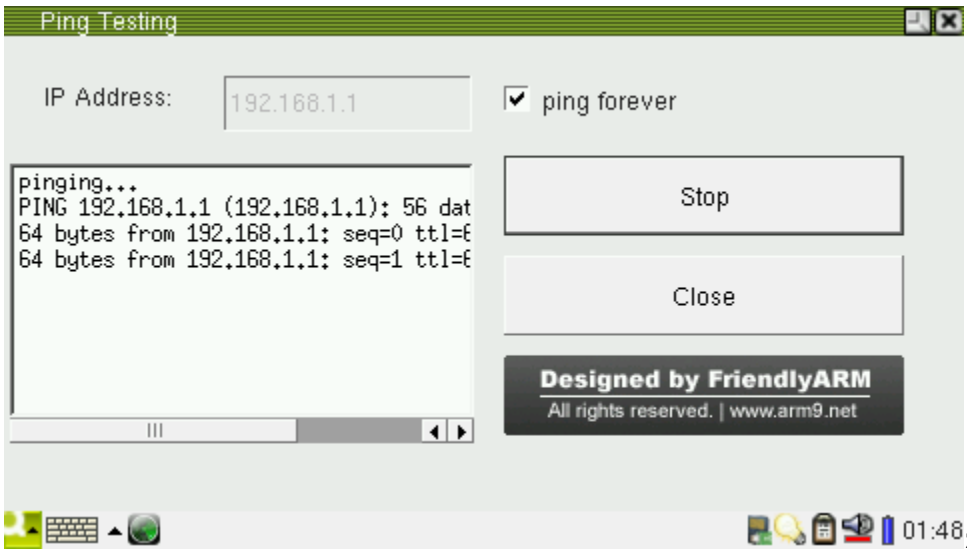
In our example the board's IP was 192.168.2.1 we could browse http://192.168.2.1 on our mobile phone.

WiFi AP Setting will save WiFi's configuration data to the following files:

Configuration File	Comment
/etc/hostapd.conf	WiFi parameters
/etc/hostcpd.conf	DHCP service parameters
/etc/rc.d/init.d/wifiapd	Start/Stop WiFi AP service
/etc/init.d/rcS	If you want to auto start WiFi AP service you can add "/etc/rc.d/init.d/wifiapd start"

4.1.11 Ping Test

After configure your network please go to “FriendlyARM” and click on “Ping Testing”



zh

4.1.12 Browser

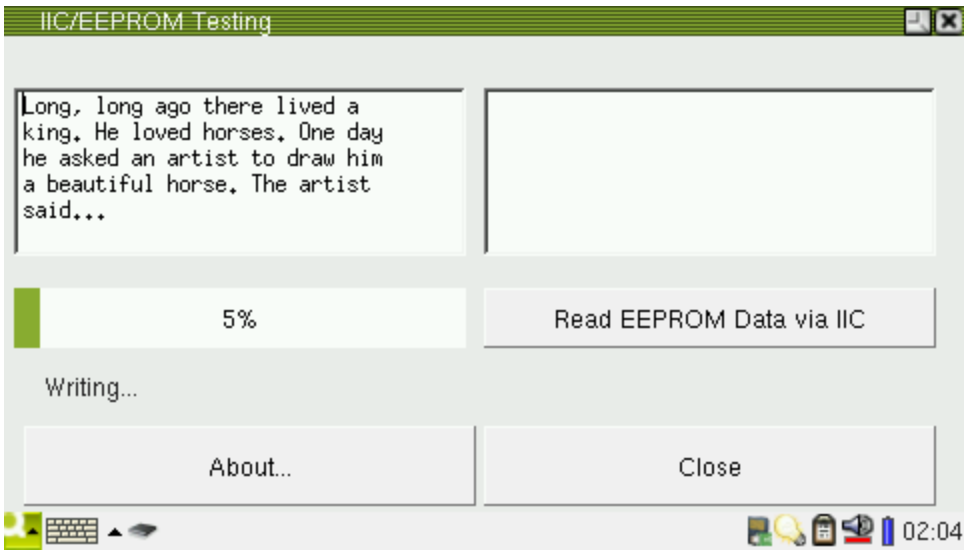
Go to “FriendlyARM”, click on “Browser”, open the soft keypad on the left bottom and you can type a website in the address bar.

4.1.13 LED Test

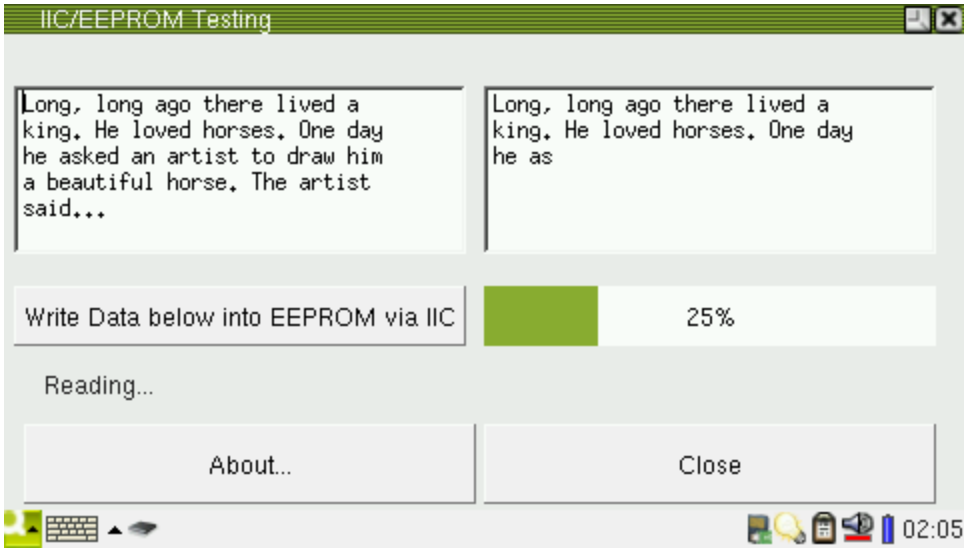
Go to “FriendlyARM” and click on “LED Testing”

4.1.14 EEPROM Reading and Writing

Go to “FriendlyARM” and click on “I2C-EEPROM”. Click on “Write Data below into EEPROM via IIC” you will see the writing process

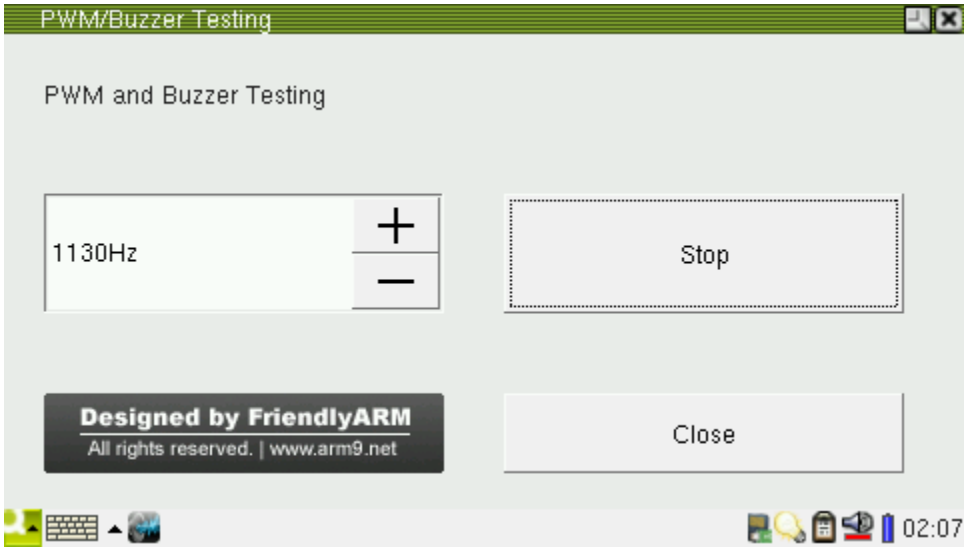


Click on “Read EEPROM Data via IIC” you will see the reading process



4.1.15 PWM Buzzer Test

Go to “FriendlyARM” and click on “PWM Buzzer” you will see the writing process.

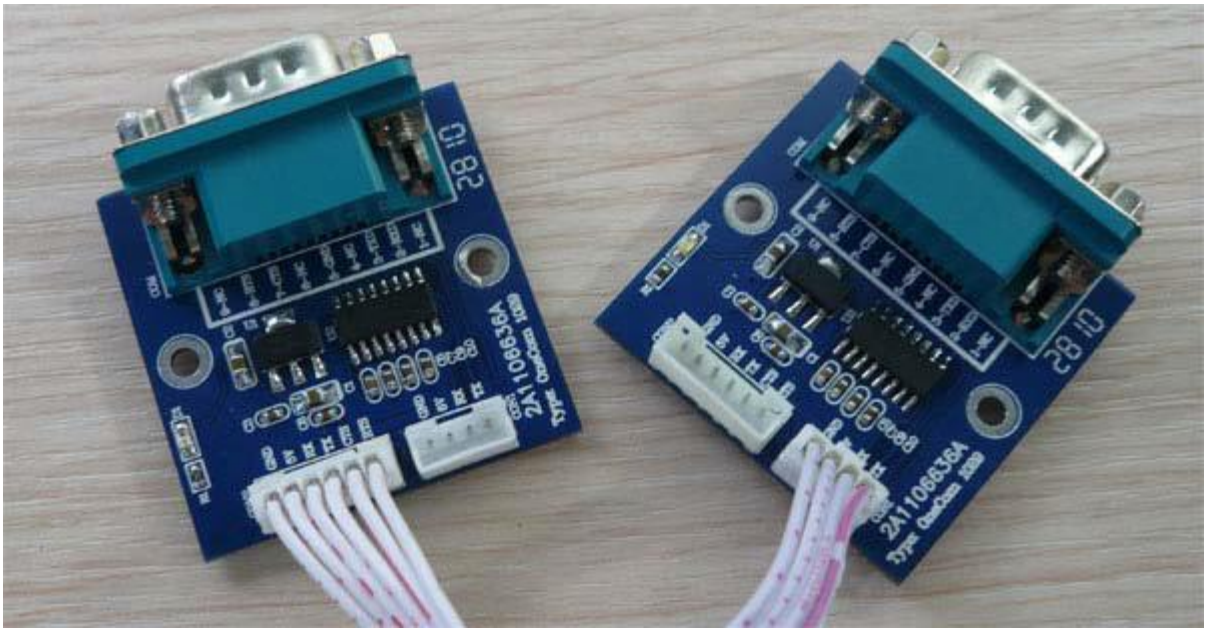


4.1.16 Serial Port

Note: before start this program please connect the serial port your want to test to your board.

- The on board CON1, 2, 3 and 4 are CPU UART0, 1, 2 and 3. UART0 has been converted to RS232, and extended to COM0 via DB9. On system startup it has been set to the console terminal, so it cannot be tested via this utility. The other three ports CON2, 3 and 4 must be converted to RS232 before they can communicate with a PC serial port. (FriendlyARM has a "OneCom" RS232 conversion module) When connect the ports to a PC, please make sure to use a correct serial cable (cross serial line or direct serial line).

- This program also supports common USB to Serial cables. Now most laptops don't have serial ports. For the sake of users most of our agents provide those conversion cables. Connecting a USB to Serial cable to your board, you can extend your serial ports. Its device name generally is "**/dev/ttyUSB0, 1, 2 and 3**", which implies you can use a USB hub to extend your serial ports.



Connect your serial port extension board to the Tiny4412's CON2/3/4 and connect to a PC via a crossover serial cable.

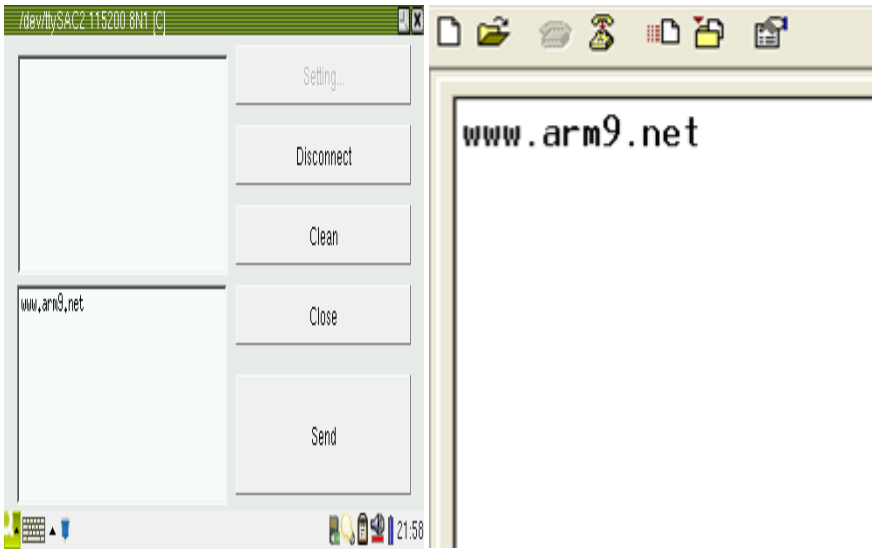
Go to the “FriendlyARM” tab and click on the “serial port assistant” icon to open the interface.

The title bar of the utility shows the default setting is “**ttySAC1 115200 8N1 [C]**”, and it implies the default port is:

- Serial Port Deivce: /dev/ttySAC1, it corresponds to the second port UART1
- Bits Per Second: 115200
- Data Bits: 8
- Flow Control: None
- Stop Bits: 1
- [C]: stands for the character mode; [H] stands for Hex

There are two edit areas in the interface, the top one shows received data which cannot be edited; the bottom one shows sent data which can be edited via a USB keyboard or a soft keyboard.

Click on the "Connect" button to open "/dev/ttySAC1", type some characters in the edit area, click on the "Send" button and it will send data to the connected serial port device. The screenshot below shows what a Windows super terminal receives (Note: the settings for this super terminal should be 115200 8N1)



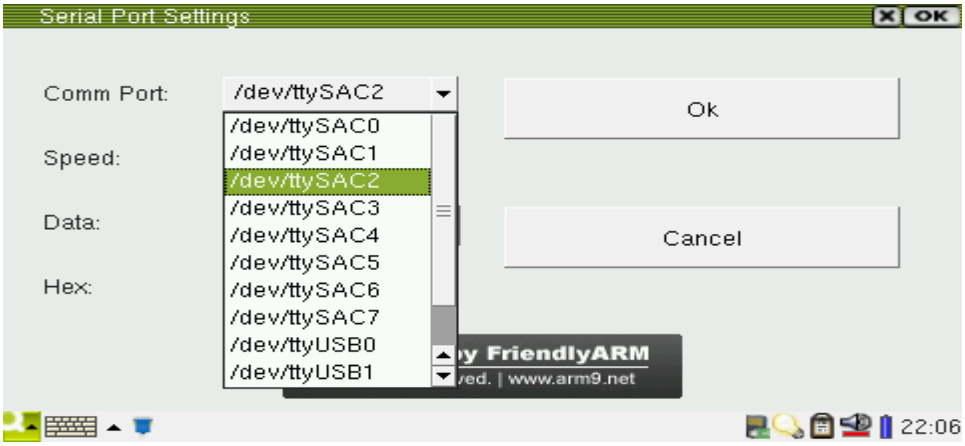
Click on "Disconnect" to disconnect the connection. Click on "Setting…" to enter the parameter setting interface which lists some basic serial port parameters:

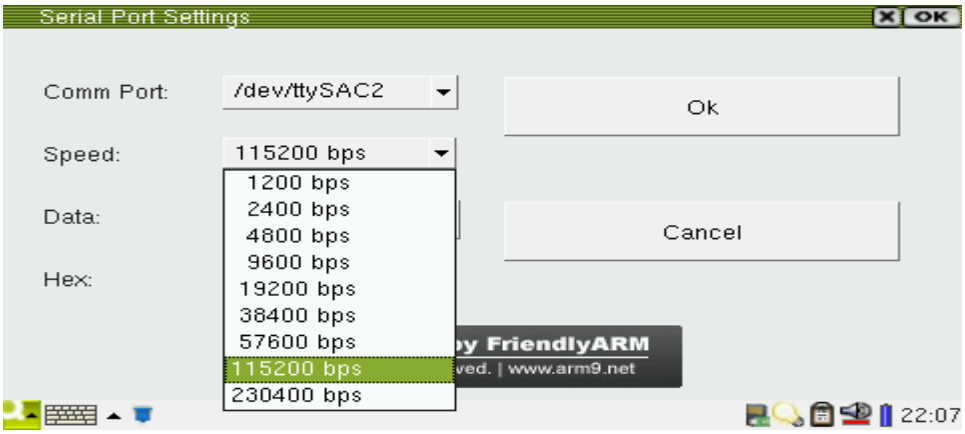
Comm Port: you can choose "/dev/ttySAC0,1,2" or the USB to Serial "/dev/ttyUSB0,1,2,3". Note: in this utility, SAC0 corresponds to CON1, SAC1 corresponds to CON2 and etc.

Speed: bits per second

Data: data bits, 8 or 7, usually 8.

Hex: input and output data in Hex format





4.1.17 Com Ping

FriendlyARM developed another serial port utility “Com Ping” which is used to test multiple serial ports

1) "Send Text" is the text that will be sent to the specified port

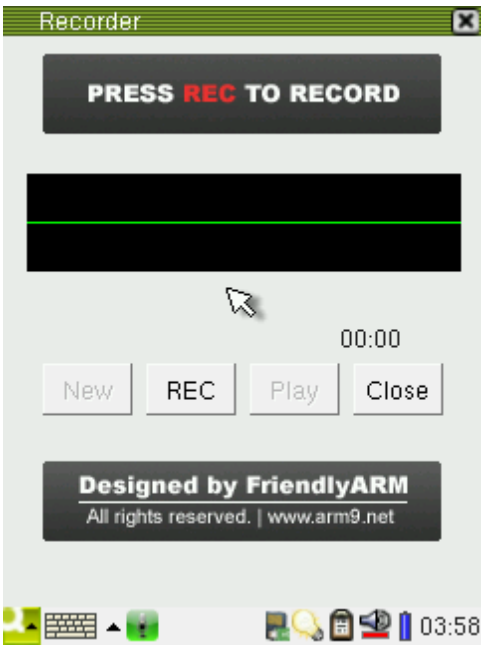
2) "Invalidate" is the time span between two text sending

3) Supports hex bytes

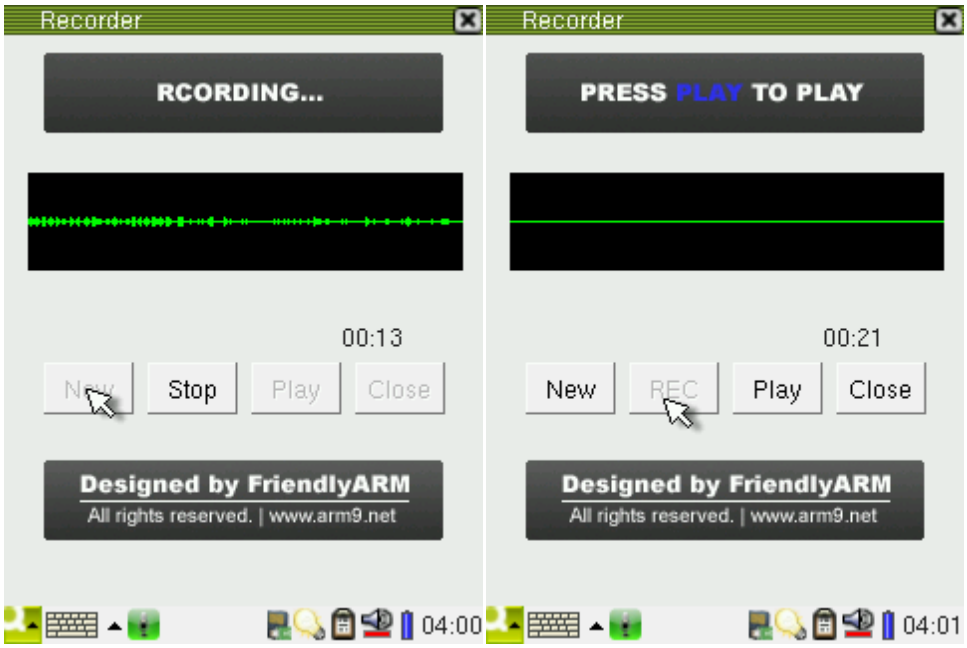
After setting these parameters click on "Start" you will be able to communicate with serial ports.

4.1.18 Audio Recording

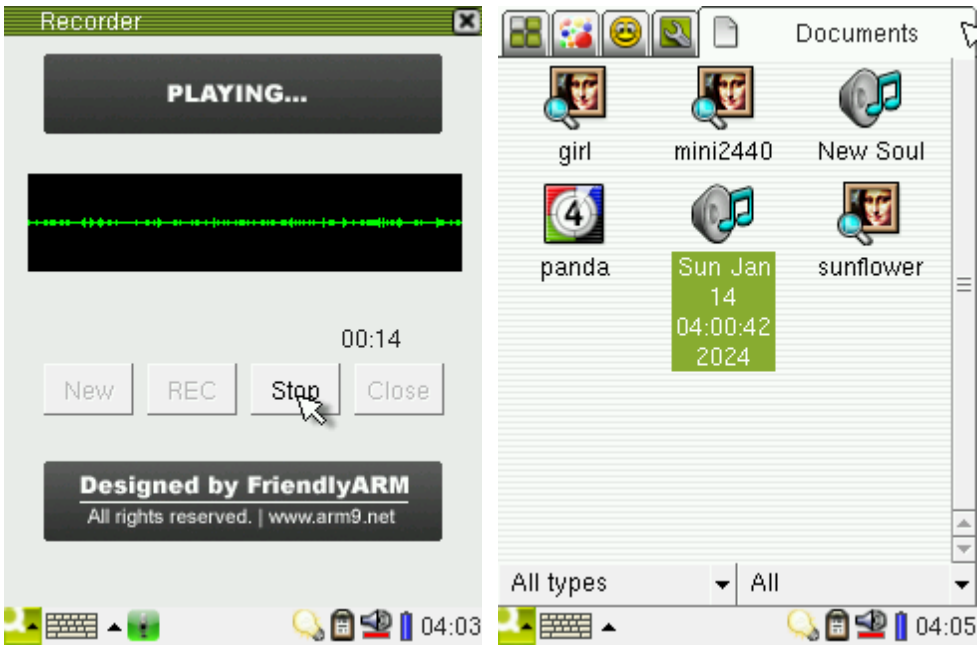
Go to the "FriendlyARM" tab and click on the "recorder" icon:



Click on the "REC" button to start recording. When you speak to the microphone on the board, you will see audio waves shown on the screen. Click on the "STOP" button to stop recording.



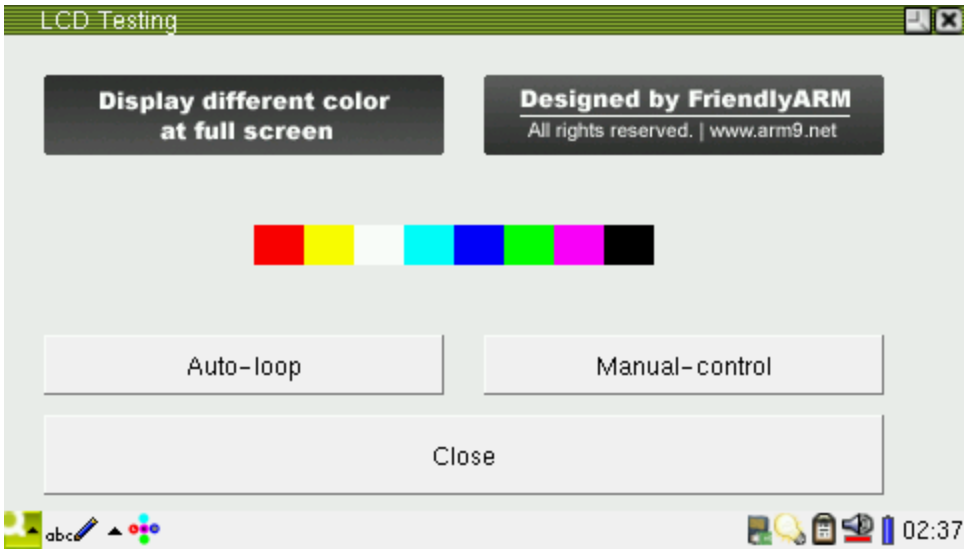
Click on the "PLAY" button to play what you recorded and you can see what you recorded has been saved as "WAV" files in the "Documents" directory.



Note: Qtopia 2.2.0 has a recorder utility by itself. But it cannot record audio. We leave it as what it is.

4.1.19 LCD Test

Go to the “FriendlyARM” tab, click on the “LCD” icon you will see the following dialog pop up:



This utility has two modes: auto and manual

Auto-loop loops automatically. Executing it presents “red”, “yellow”, “white”, “sky blue”, “dark blue”, “green”, “pink” and “black”. During the loop clicking on any place on the screen will return

4.1.20 Backlight Control

Note: this feature requires an LCD driven by the 1-wire precise touch driver.

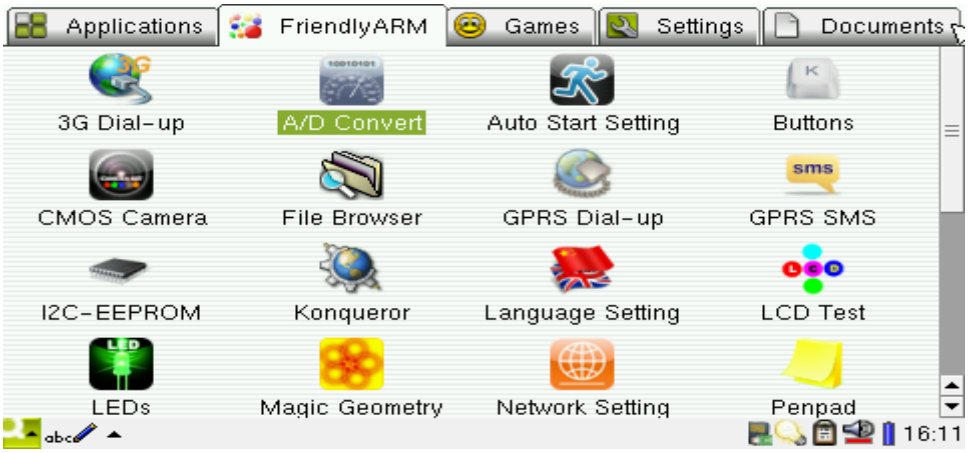
If you already played our 4412 system you may notice that after power on the board will turn “dark” without being touched for a period. This is a default system action controlled by the backlight management. In the “Settings” tab clicking on “Power Management” will start this utility

Here the default setting is 25 seconds you can click on the “Up” or “down” arrow to adjust it. If you uncheck “Light off”, the backlight will be on as long as the system is powered on. An LCD driven by the 1-wire precise touch driver integrates the function of adjusting the backlight therefore you can slide the slider to get your desired backlight. When you check “Dim light” you will observe that the light is off gradually. Actually adjusting the backlight in our software is pretty straightforward. You can refer to 2.10 for more details on how to adjust it via the command line utility.

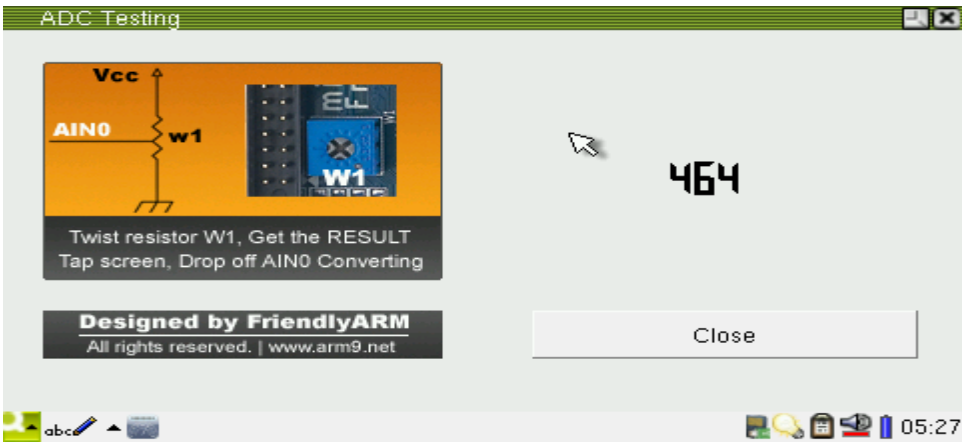
4.1.20 A/D Conversion

The Samsung 4412 chip has 8 A/D conversion channels but only one converter. In general, AIN4, AIN5, AIN6 and AIN7 are used as YM, YP, XM and XP channels via a four wire resistor. We extended AINs 1-3 which reside on CON6. For easier testing, AIN0 is directly connected to an adjustable resistor W1. How do they share a common converter? The following screenshots will show you:

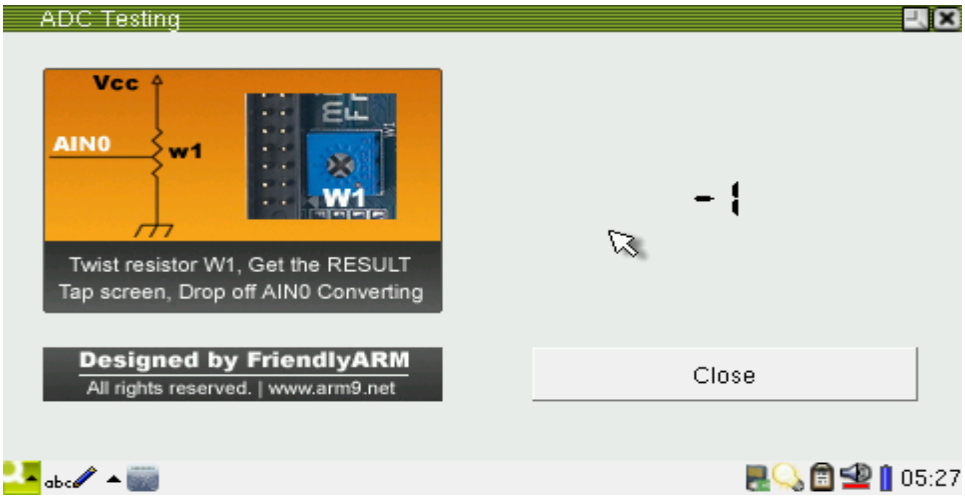
Click on the "ADC Testing" icon in the "FriendlyARM" tab:



Turning the W1 adjustable resistor, you will see the conversion changes. It has 10 digit precision, therefore the minimum value is close to 0 and the maximum value is close to 1024.

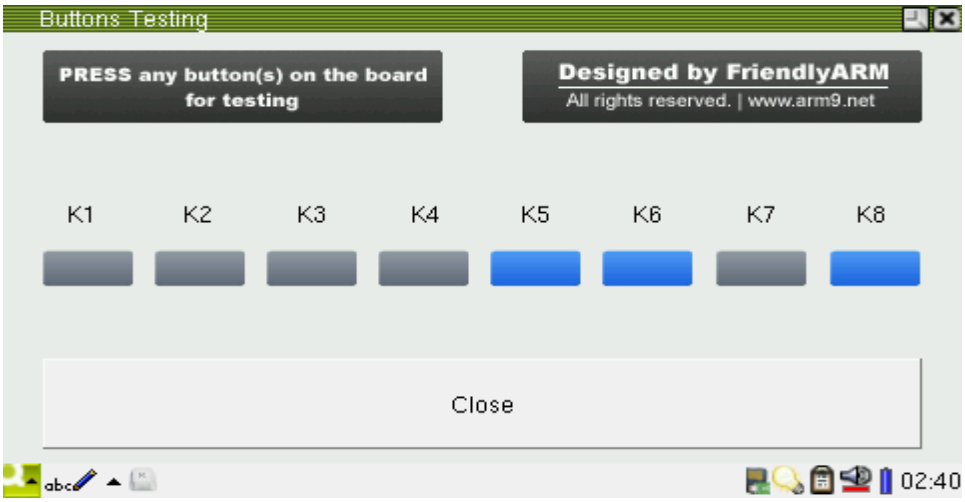


When you click on the touch screen, the A/D converter will take the touch screen as the channel, you will see the result "-1"; when you move your touch pen away from the screen, the A/D converter will take AIN0 as the channel again.



4.1.21 User Button Test

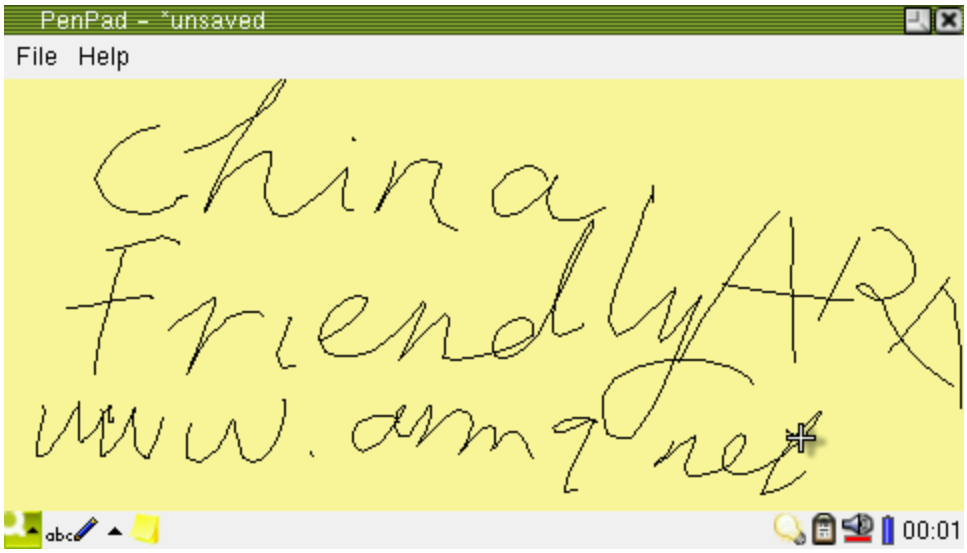
Note: the user buttons don't have dedicated functions and they are just for testing low level drivers. Click on the "Buttons" icon in the "FriendlyARM" tab. Press down any buttons on the board, the corresponding button icons will change to blue, release them, their icons will change back to grey.



4.1.22 Touch Pen Test

To test whether or not a touch pen works properly, you can draw a line on the LCD, check if there is any offset or vibration. This can be done via the “penpad” utility. Click on the “penpad” icon in the “FriendlyARM” tab.

The “penpad” utility is an easy to use program developed by FriendlyARM. Start it, a yellow drawing area will show up. Draw whatever you like in the area (the pen color is black, its width is 1 pixel), go to “File” -> “Save”, you will save what you draw to a png file(in the “Documents” tab, the /Documents/image/png/ directory). The file name begins with 001. The maximum number of files that can be saved is 999. The following screenshot shows that our writing was smooth which meant our pen was accurate.



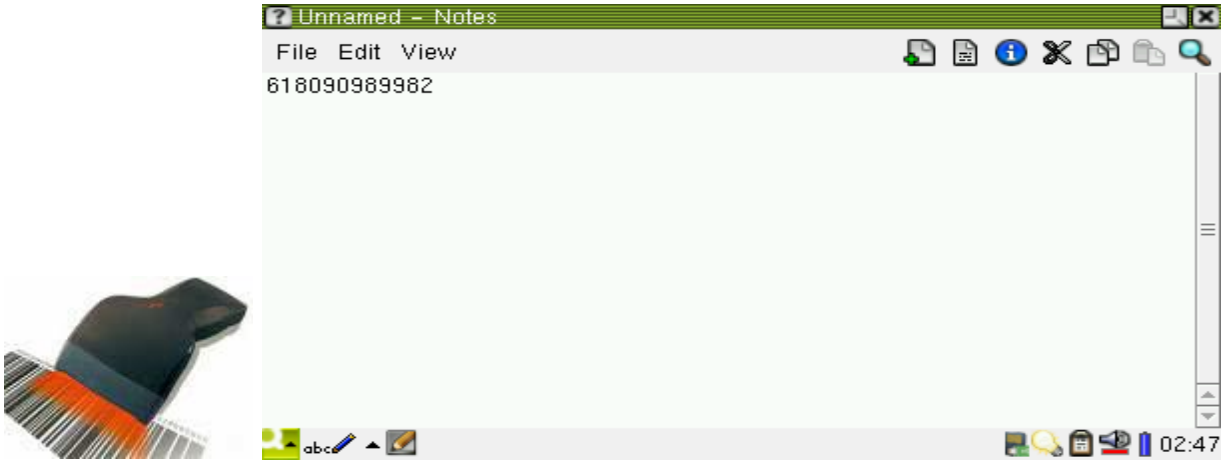
4.1.23 Barcode Scanning

Our system supports USB barcode scanners which are actually a HID device very similar to a USB keyboard. Therefore a barcode scanner can work any where a USB

keyboard works.

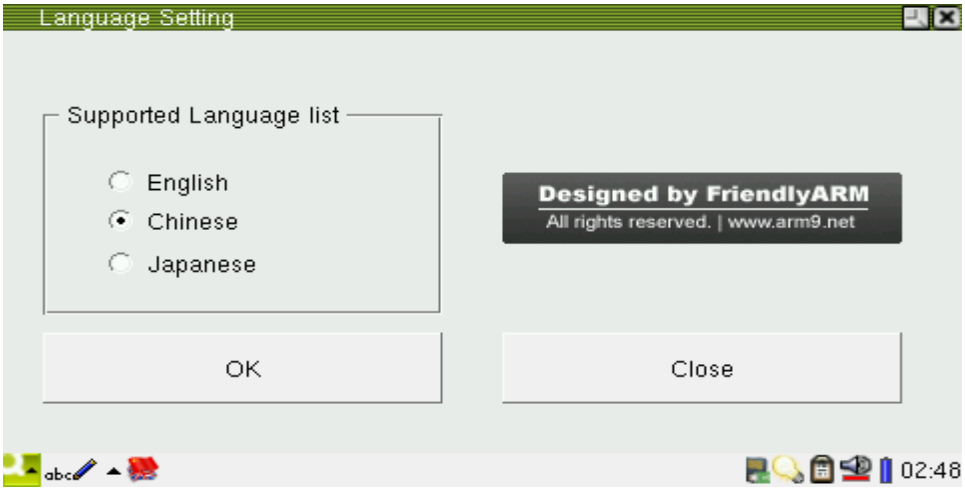
Note: before start this utility, please make sure to plug in your scanner.

Click on the “text editor” icon in the “Application Programs” subgroup, scan a code with your scanner, then you will see the code number displayed in the editor.

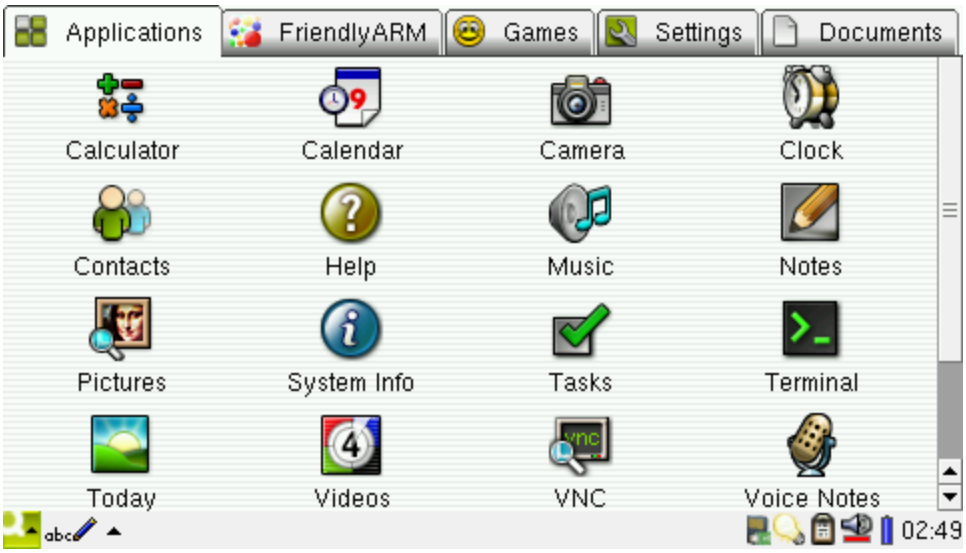


4.1.24 Language Setting

Qtopia 2.2.0 has a language setting utility which is different from the one in Qtopia 1.7.0. It only supports English. Therefore we developed a new utility located in the “FriendlyARM” tab (the icon is a waving flag).



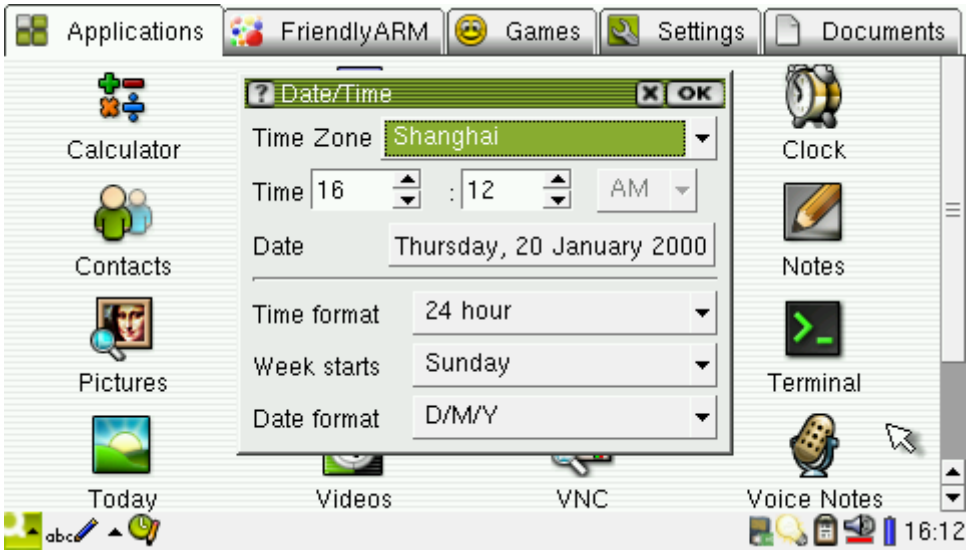
It now supports three languages: English, Chinese and Japanese. When you select “English”, then click on “OK”, a message will popup asking you if you want to change your language setting. Clicking on “Yes” Qtopia will reboot; clicking on “No” it will return. (Note: the Chinese and Japanese versions only have file names translated).



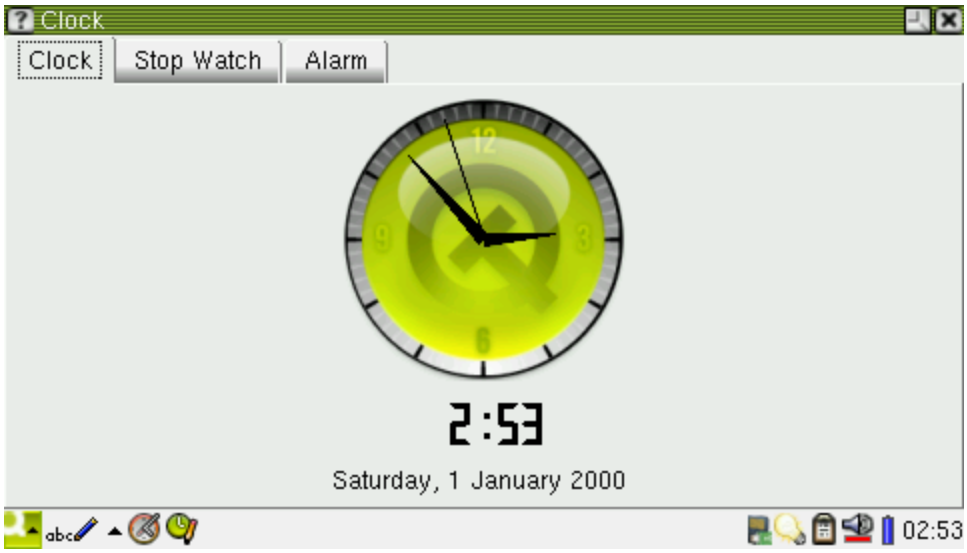
4.1.25 Set up Time Zone, Date, Time and Alarm Clock

When you get our system, the date and time usually might not be accurate. You can

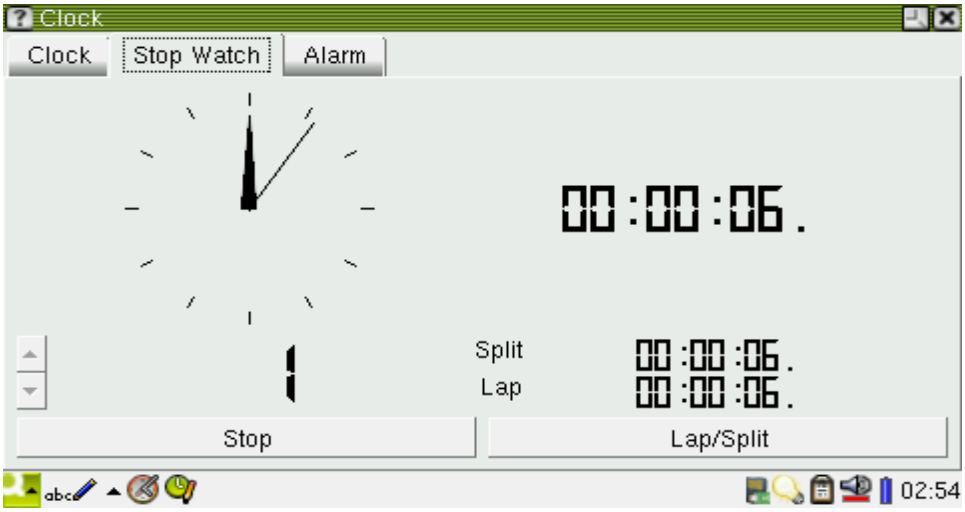
adjust them by yourself. Because the CPU has its own RTC and the board has a backup battery, after you adjust the date and time, they will be saved. To adjust them, click on the time zone area at the right bottom of the screen, a menu will show up, please select "Set time..", open the setting interface where you can set parameters such as time zone, date, time and so on



Select "Clock" from the menu.



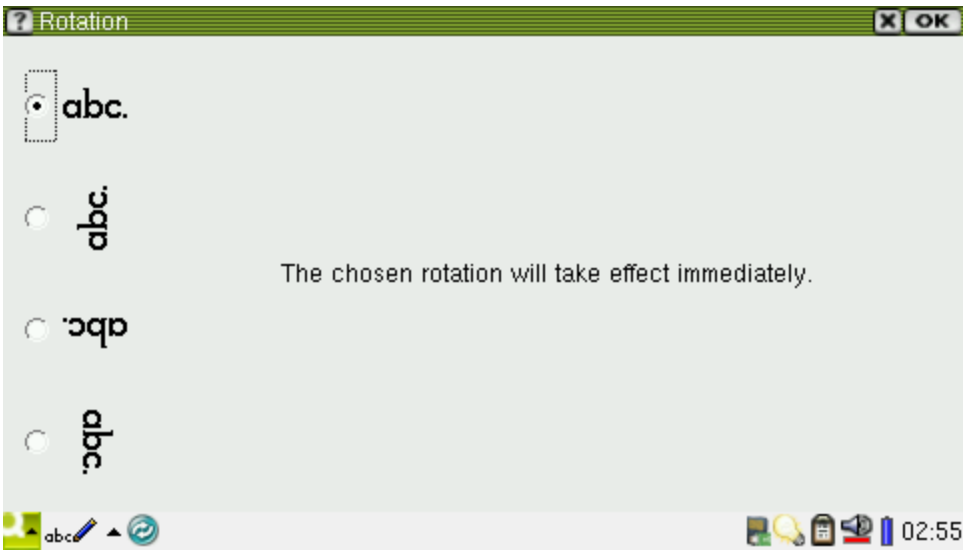
Click on “Stop Watch” to open a stopwatch utility



Besides you can set the alarm clock. When it is triggered, you will hear a beeping sound which lasts about one minute and the following popup window will show up. Click on “OK” to close the alarm clock.

4.1.26 Rotate Screen

Click on the “rotation” icon in the “settings” tab to enter its interface. You can rotate the screen in four directions.



Select the direction you want, click on “OK” you will see the screen rotate.

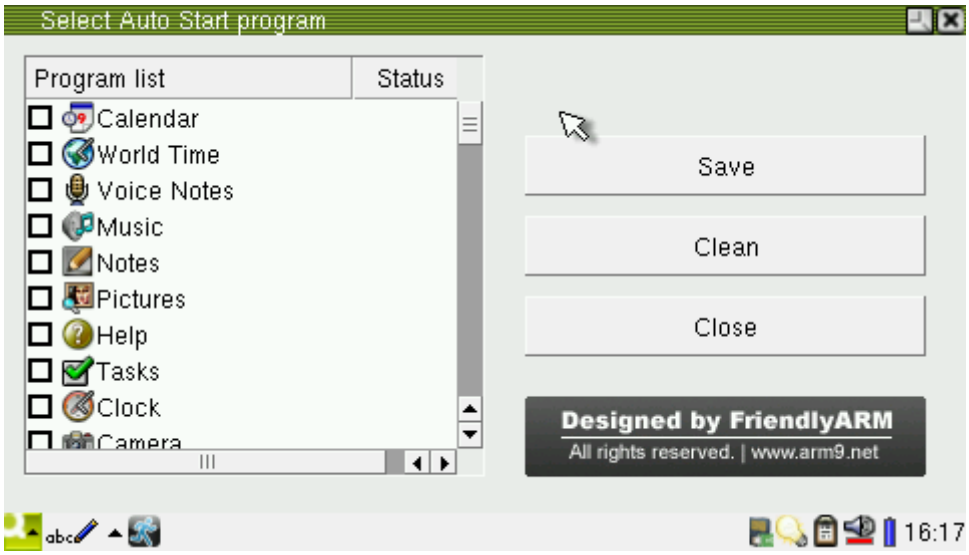
Note: sometimes you need to reboot Qtopia to see the rotation. It is a Qtopia utility and we hasn’t made any change to it. In addition the rotation effect is implemented via Qtopia software and has nothing to do with LCD drivers.

After rotation you will notice that all “FriendlyARM” utilities get rotated too. We implemented this feature to make our utilities displayed properly with different LCDs

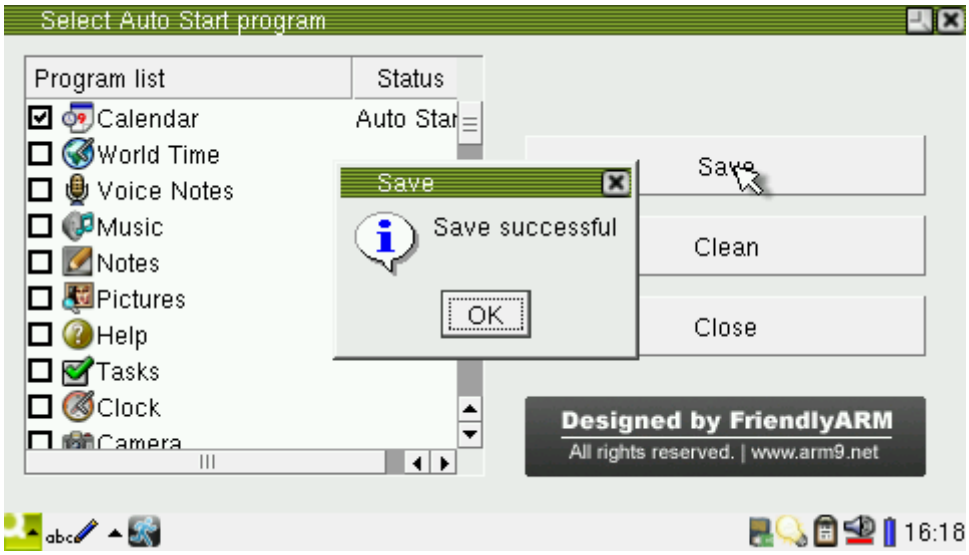
4.1.27 Setup Auto Run Program

By setting “auto run” you can make Qtopia launch its own or your programs after it boots up. It is very similar to what you see in Windows “Programs -> Startup”.

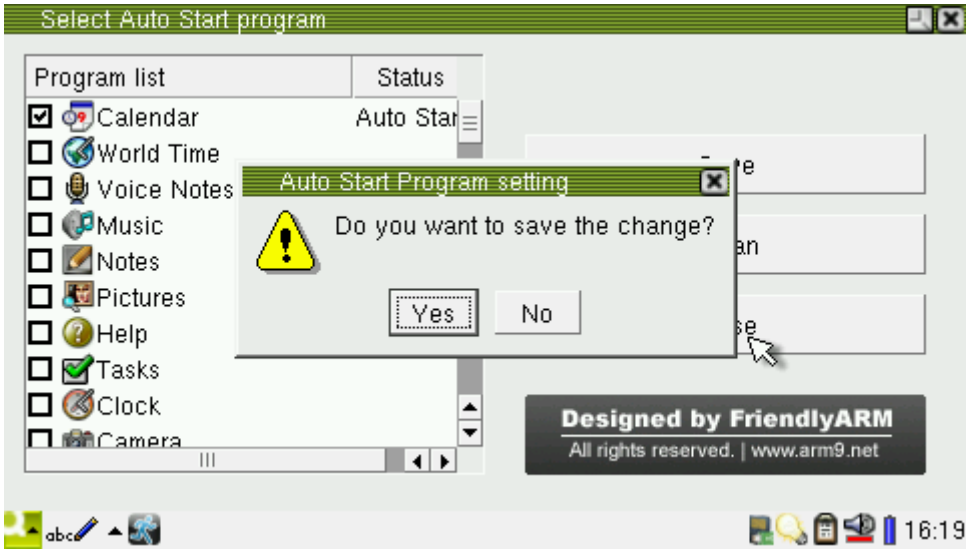
Click on the “Auto Start Setting” icon in the “FriendlyARM” tab.



Those program listed are available programs which include all Qtopia programs, the status column indicates whether a program is set to auto start. The status is unique. For instance, if the “Serial Port Assistant” is checked, its status will show “Auto Start”, click on “Save”, a message box will pop up prompting that the net setting has been successfully saved. Close this utility, reboot the system you will see the “Serial Port Assistant” is auto run.



To disable auto run for a program, just click on “Clean” and “Close”, a message box will pop up, click on “Yes” the auto run for that program will be disabled.



4.1.28 System Shutdown

In the “Settings” tab, click on the “shutdown” icon you will see four options on the shutdown window.

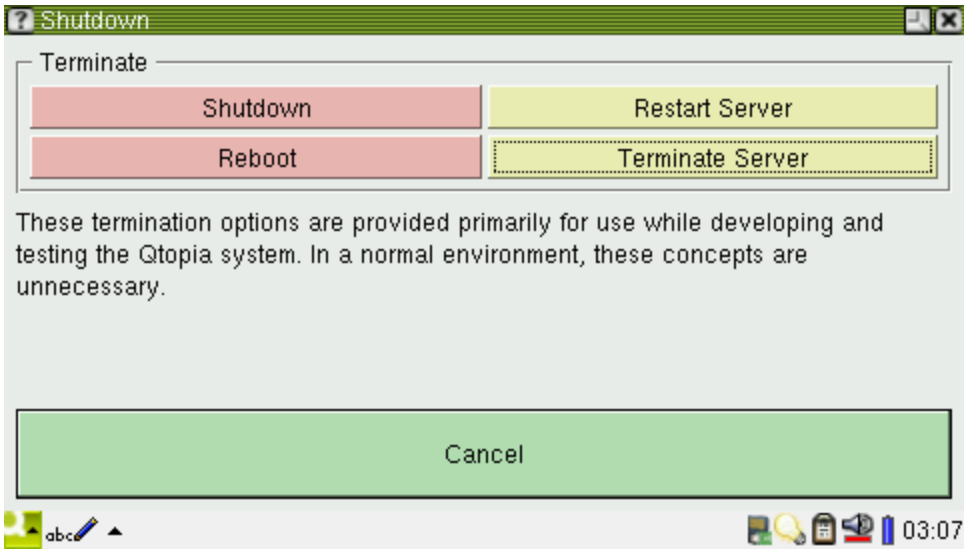
Shutdown: Press this button, Linux will end all the programs and services to shutdown the whole system. After the whole system is shutdown, the CPU will not be running and the system consumes lest power. However since our system doesn’t have a hardware power down circuit you still can see the power LED on the board is on.

Reboot: This is a “hot” reboot button. If your system boots from the Nor Flash, after you press this button, the system will shutdown, reboot and enter the supervivi main menu. If your system boots from the Nand Flash, after you press this button, the system will shutdown, reboot and enter the Qtopia interface.

Note: **Reboot** is different from the "Watchdog" function we will introduce. The "Watchdog" is "cold" reboot and doesn't end programs or services but reset the system instead.

Restart Server: it restarts the Qtopia system only. It doesn't interrupt the running Linux.

Terminate Server: it shuts down the Qtopia system. After press this button, the Qtopia interface will be disabled. What is left on the screen is the left data in RAM and it is not an active graphic interface.

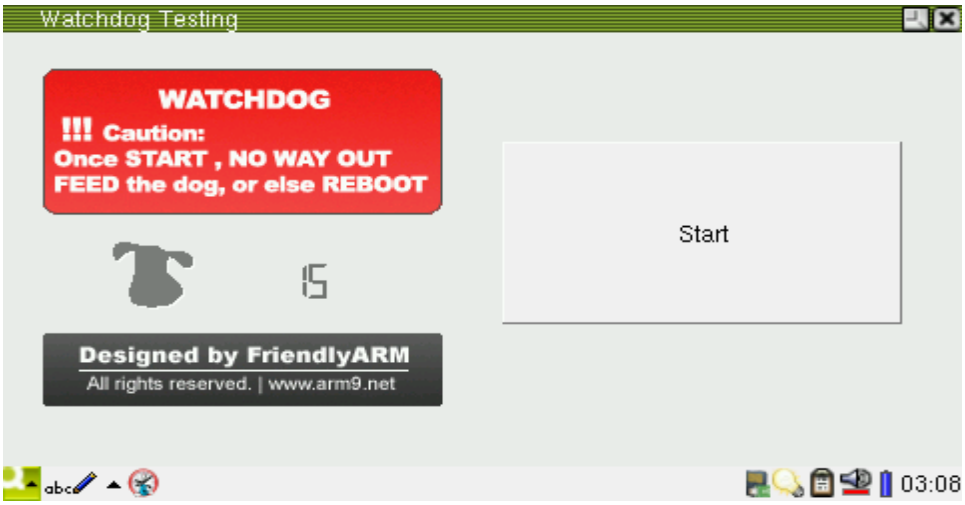


Note: the original Qtopia 2.2.0 system doesn't "shutdown" or "reboot" effectively, we changed its code to make it work.

4.1.29 Watchdog

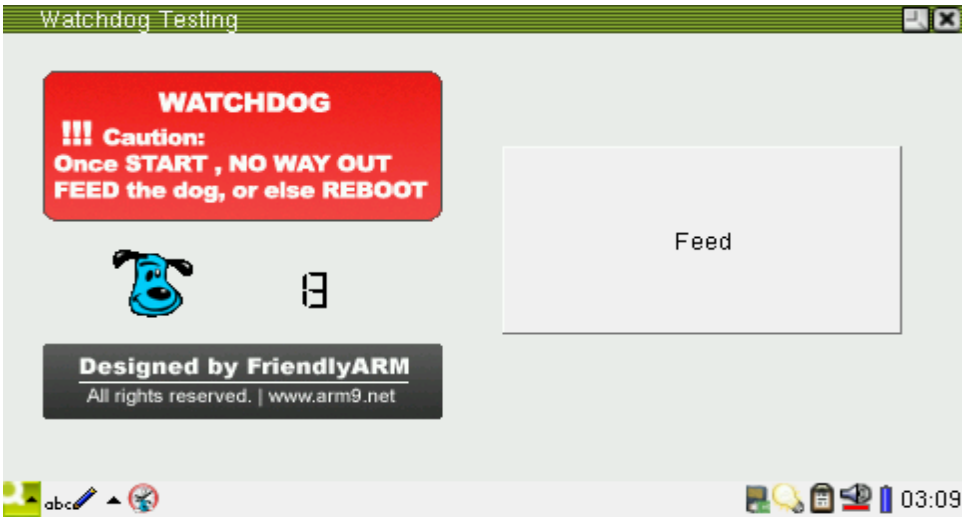
The "Watchdog" is a very basic utility in embedded systems. The 4412 chip already has a watchdog. The latest Linux kernel has drivers for it.

Click on the “Watchdog” icon in the “FriendlyARM” tab



Note: before take any action, please read the notes in the red area: once start, no way out, feed the dog, or else reboot!

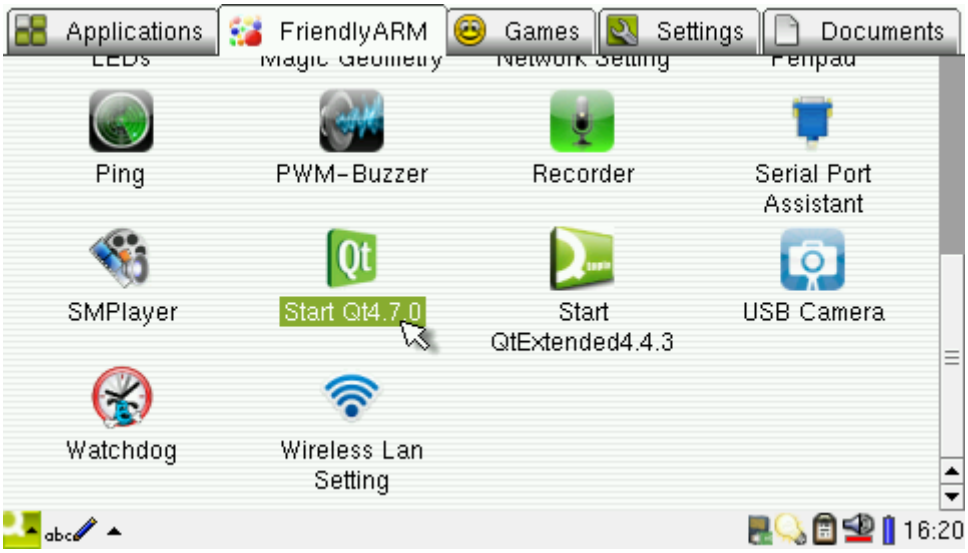
Here we set a countdown time 15 seconds. To feed the dog, click on the “Feed” button. Keep feeding, it will always have bones and the system will not reboot.



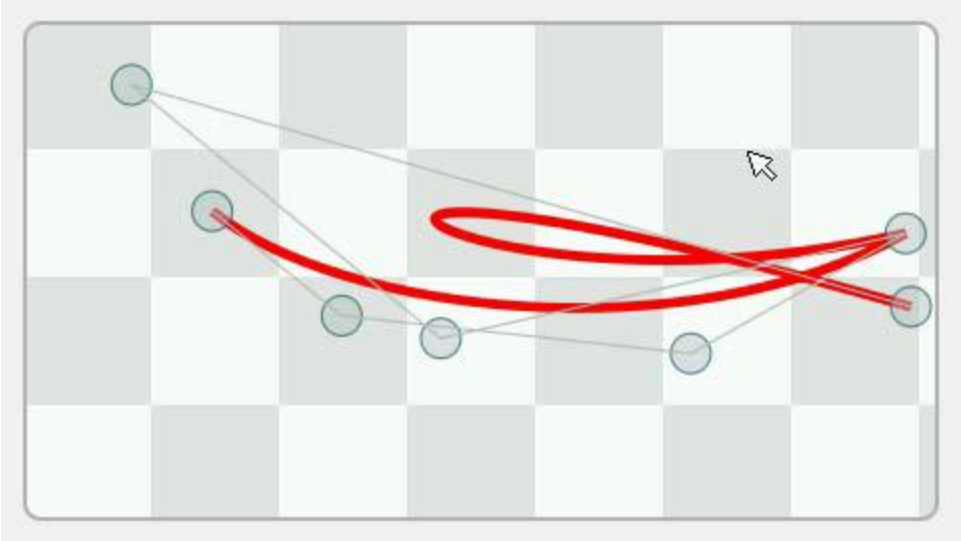
4.1.30 Start QtE-4.7.0

In order for users to switch freely and smoothly between different systems we implemented a feature that allows Qtopia-2.2.0 and QtE-4.7.0 to co-exist in the same

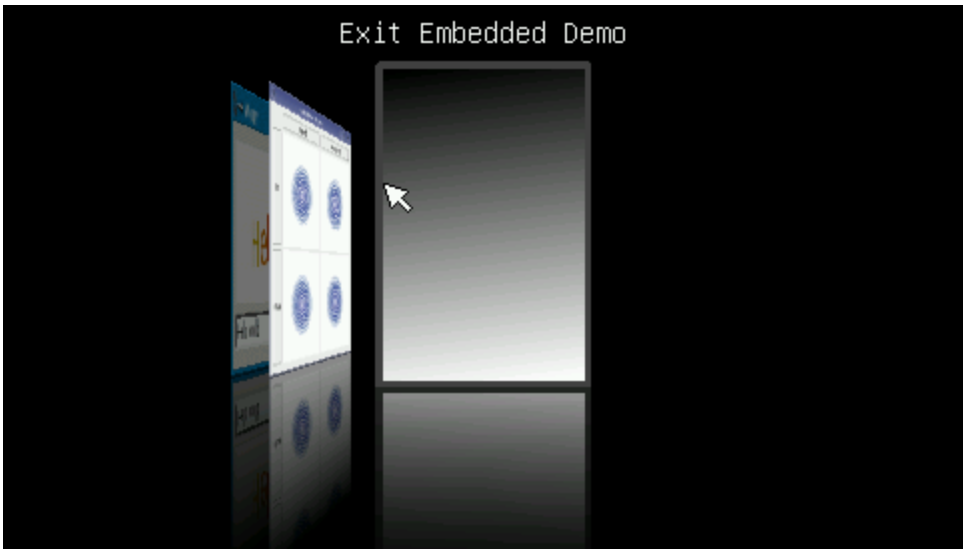
file system. In Qtopia-2.2.0, by clicking on a common application icon users will be able to start QtE-4.7.0. After close the QtE-4.7.0 utility, users will be able to return to Qtopia-2.2.0.



QtE-4.7.0 runs as follows. It is a program manager that display a CoverFlow effect. You can drag it left and right and run it by clicking on one of the Covers.



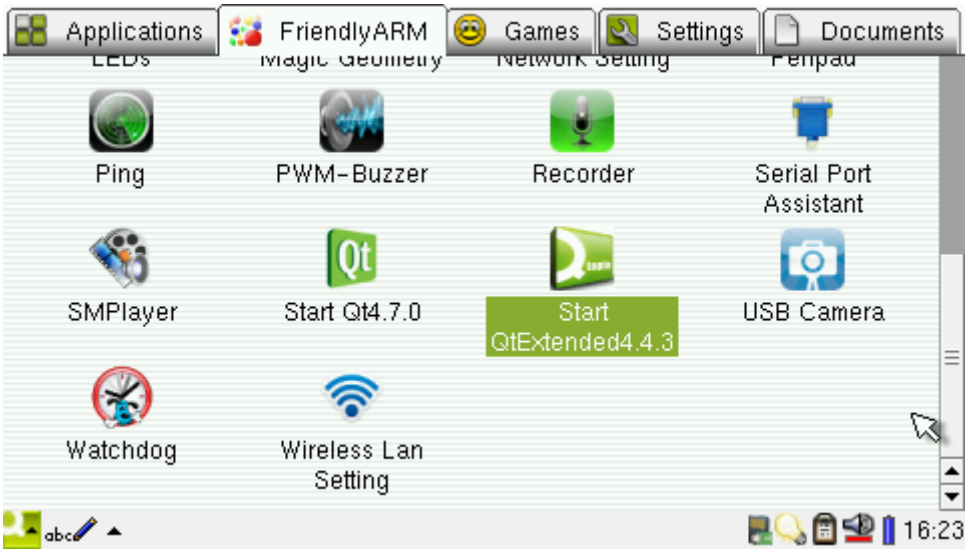
You can exit QtE-4.7.0 by clicking on “Exit Embedded Demo” and return to Qtopia-2.2.0



4.1.31 Start Qtopia4

In order for users to switch freely and smoothly between different systems we implemented a feature that allows Qtopia-2.2.0 and Qtopia4(Qt Extended 4.4.3 Phone) to co-exist in the same file system. In Qtopia-2.2.0, by clicking on a common

application icon users will be able to start Qtopia4. After close the Qtopia4 utility, users will be able to return to Qtopia-2.2.0

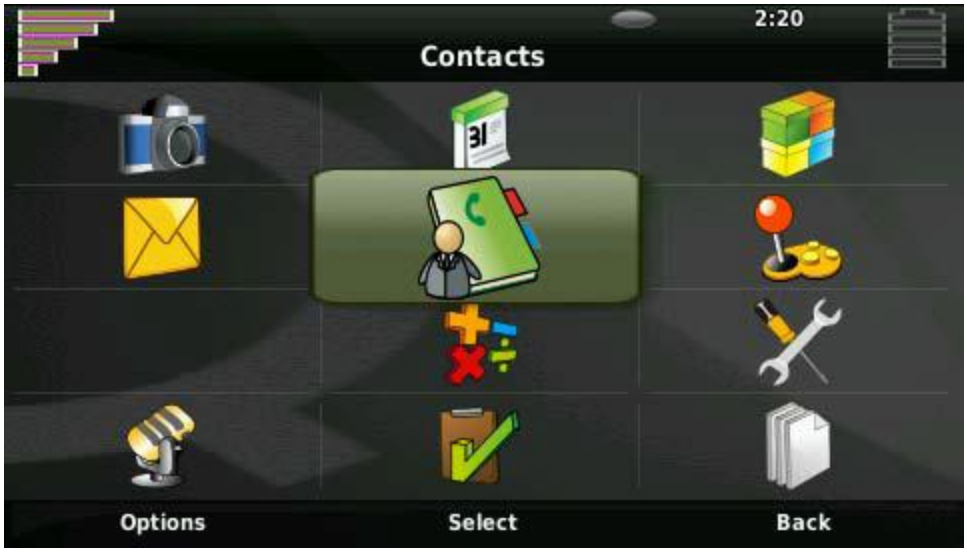


If you have never run Qtopia4 on the system you will see the following GUI after start it

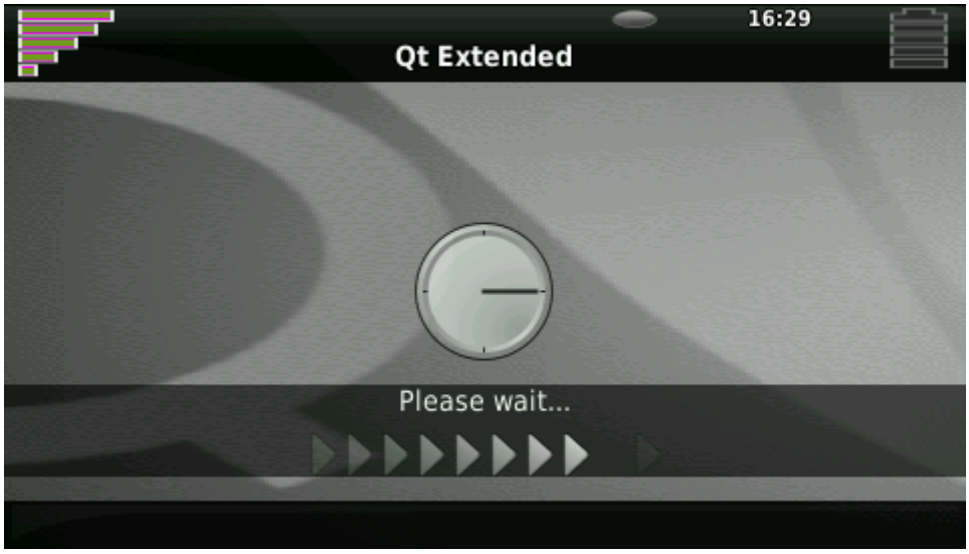
Click on the screen you will be directed to a configuration window where you can set up your date and time. You can ignore it here and click on “Finish” to continue.

After a while you will enter Qtopia4 and the title is “Qt Extended”

There are three buttons “Options”, “Menu” and “Quit”. “Quit” is implemented by FriendlyARM to allow users easily returning to Qtopia2. You can add your own features too in the source code. Here please click on “Menu” to enter the main function menu.



Qtopia4's applications are very limited which we will not talk too much about here. Click on "Back" to return to the main menu and click on "Quit" to return to Qtopia2.



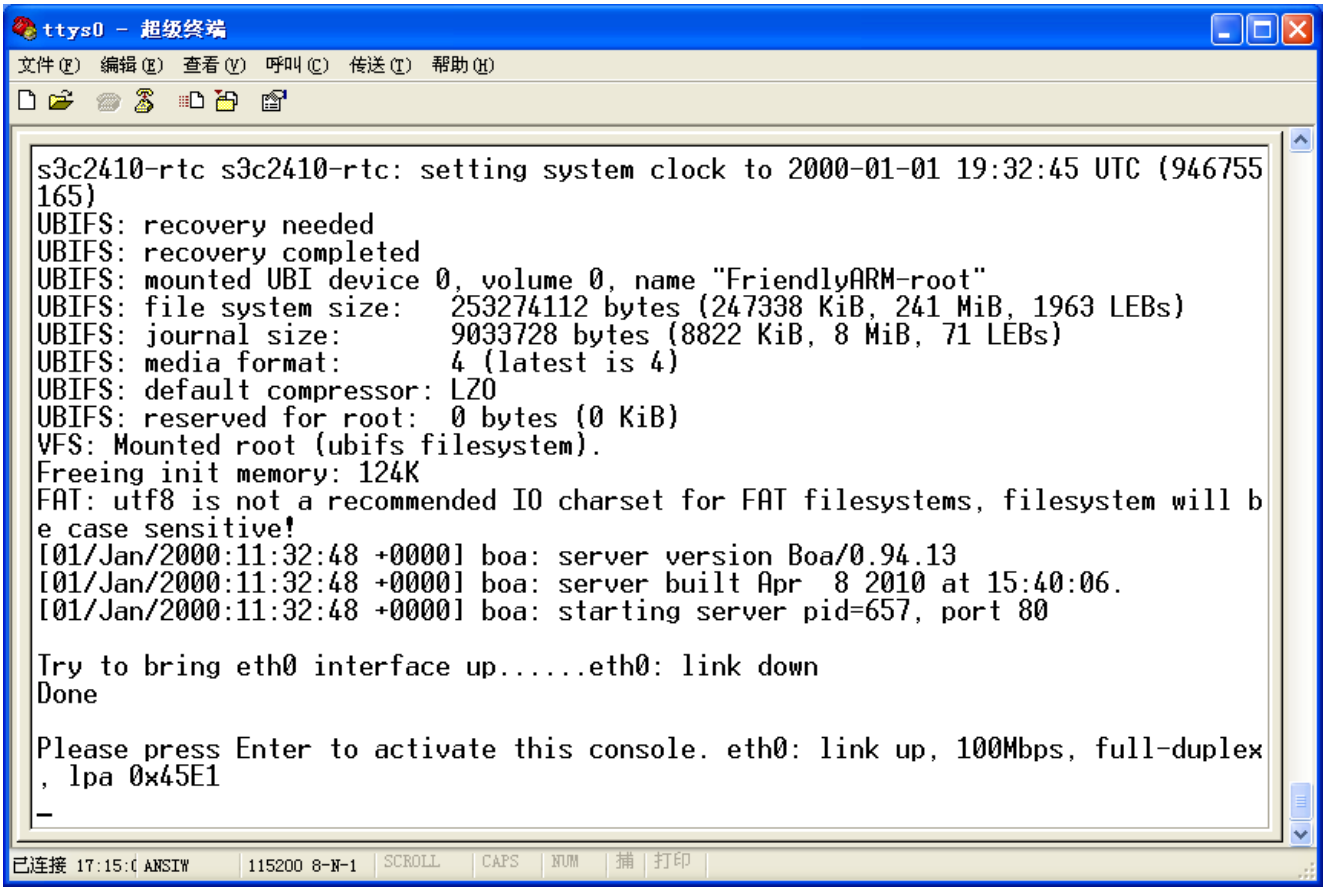
Note: when quitting users will see a flash which doesn't exist in Qtopia and is implemented by FriendlyARM. It is open source and users can check it.

So far, we have introduced most of the GUI utilities that will be used to manipulate hardware. There are other utilities you can try by yourself.

4.2 Play Tiny4412 via Command Line

Note: every Linux fan may need to get familiar with the command line utility. All Linux commands are very similar (99% of them are identical) across different versions. Before step in this section, please set up your super terminal properly.

Below is a screenshot of system login via super terminal. Just press “Enter” as prompted to continue.



4.2.1 Play MP3

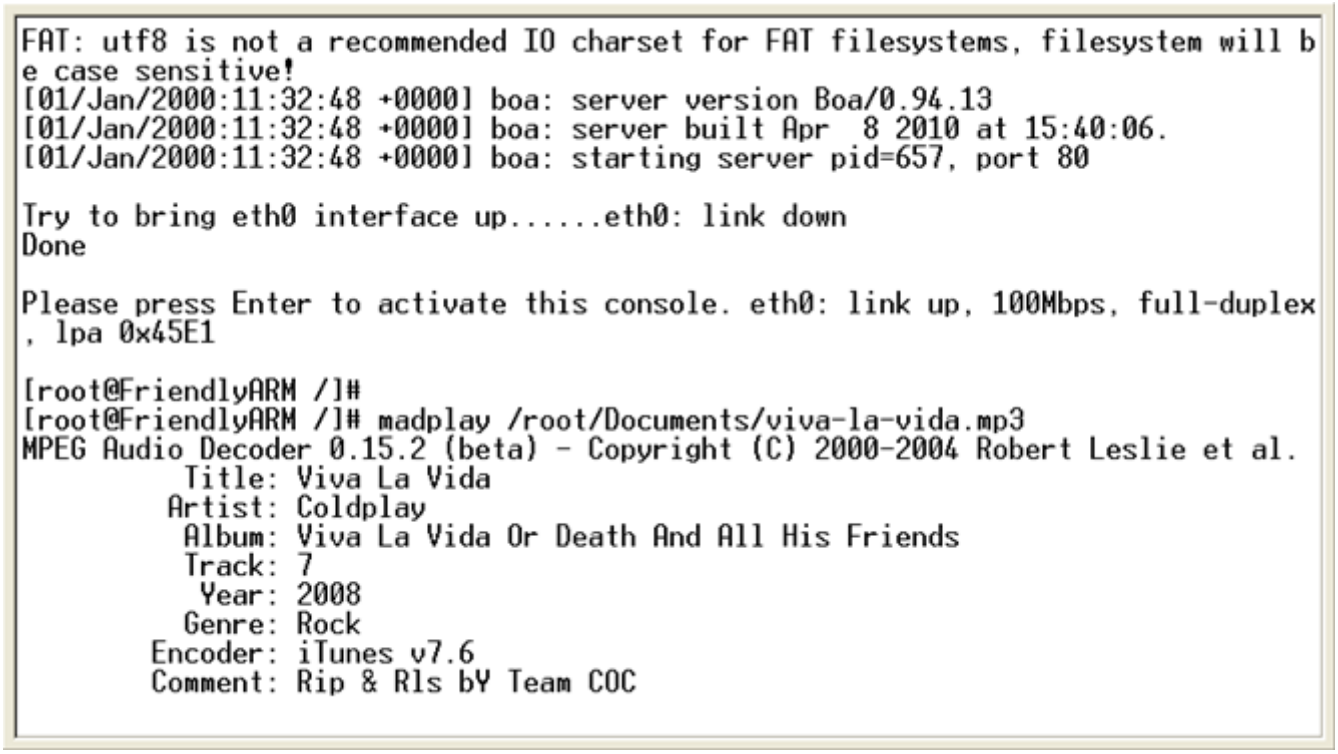
This section will give a brief introduction on how to run Linux commands and various application programs in Linux via a super terminal. Before move forward,

please connect your board with a PC and start a super terminal. The following screenshot is what you might see after you set up your super terminal and connection with your board

The madplay utility is an mp3 player migrated by FriendlyARM. It can be run in various ways and the most straightforward one is this:

#madplay your.mp3

This command will play "your.mp3" in its default way. You can get help by running "madplay -h". Below is a screenshot of how it works.



In the latest Linux kernel, we integrated a driver for ALSA audio interface. The madplay utility plays audio files via this interface too and related ALSA libraries are also integrated into the system.

4.2.2 Terminate Program

To terminate a running program you can press Ctrl + C in a terminal. For instance, if you are running madplay you can press Ctrl + C to terminate it. If a program runs in the background you need to issue the "kill" command to terminate it.

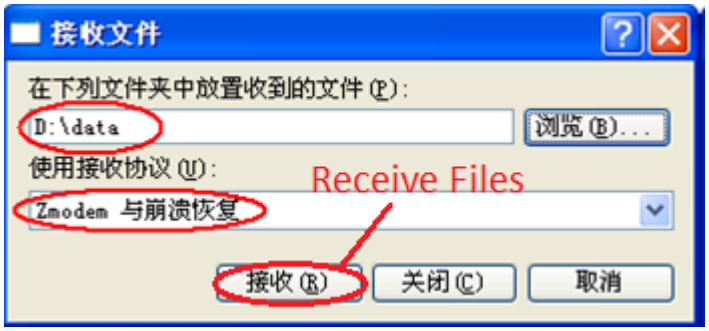
4.2.3 File Transfer To and From PC via Serial Port

Note: some users may not get file transfers done successfully via a USB to Serial connector it could result from the cable's bad quality

After login into the Tiny4412 system via a serial port, you can transfer files to and from a host PC by using command "**rz**"or "**sz**" as follows:

(1) Transfering files by using "sz"

Open a super terminal, click on the mouse's right button, then click on "Receive files" to set up the destination directory and the protocol this transfer will use, see the screenshot below:



Type "**sz /root/Documents/viva-la-vida.mp3**" in the shell to transfer the "viva-la-vida.mp3" file under the "/root/Documents" directory to the host PC. It took a while to transfer this big file. After it is done, the system will save it in the directory you

set in the previous step. Please see the screen-shot below:

(2) Transfering files by using "rz"

In your Tiny4412 system, type "rz" to receive files from a host PC.

Open a super terminal, click on the mouse's right button, select "Send file", set up the file being sent and the protocol the transfer will use. Then send the file:

Click on "Send" and the board will begin to receive your file

After the transfer is done, the current directory will get this file. You can verify it by using “md5sum” to check whether this file is the same as the original one.

4.2.4 LED Test

(1) LED Server

After the system starts up it will automatically start a LED service (/etc/rc.d/init.d/leds), it actually runs a led-player script. After the led-player script is run a pipe file led-control will be created in the /tmp directory.

Users can change an LED’s flashing by setting its parameters

```
#echo 0 0.2 > /tmp/led-control
```

After this command is executed each of the 4 LEDs will be flashing one by one with a 0.2 second in between.

```
#echo 1 0.2 >/tmp/led-control
```

After this command is executed 4 LEDs will be running the accumulator one by one

with a 0.2 second in between.

```
#/etc/rc.d/init.d/leds stop
```

After this command is executed all 4 LEDs will be turned off.

```
#/etc/rc.d/init.d/leds start
```

After this command is executed all 4 LEDs will be turned on.

(2) Manipulating a Single LED

The /bin/leds utility can be used to manipulate a single led. To launch this utility users need to stop the led-player service first:

```
#/etc/rc.d/init.d/leds stop
```

This command will stop the led-player service. To get more information for the usage of "led" you can type the following command:

```
[root@fa /]# led
Usage: leds led_no 0|1
```

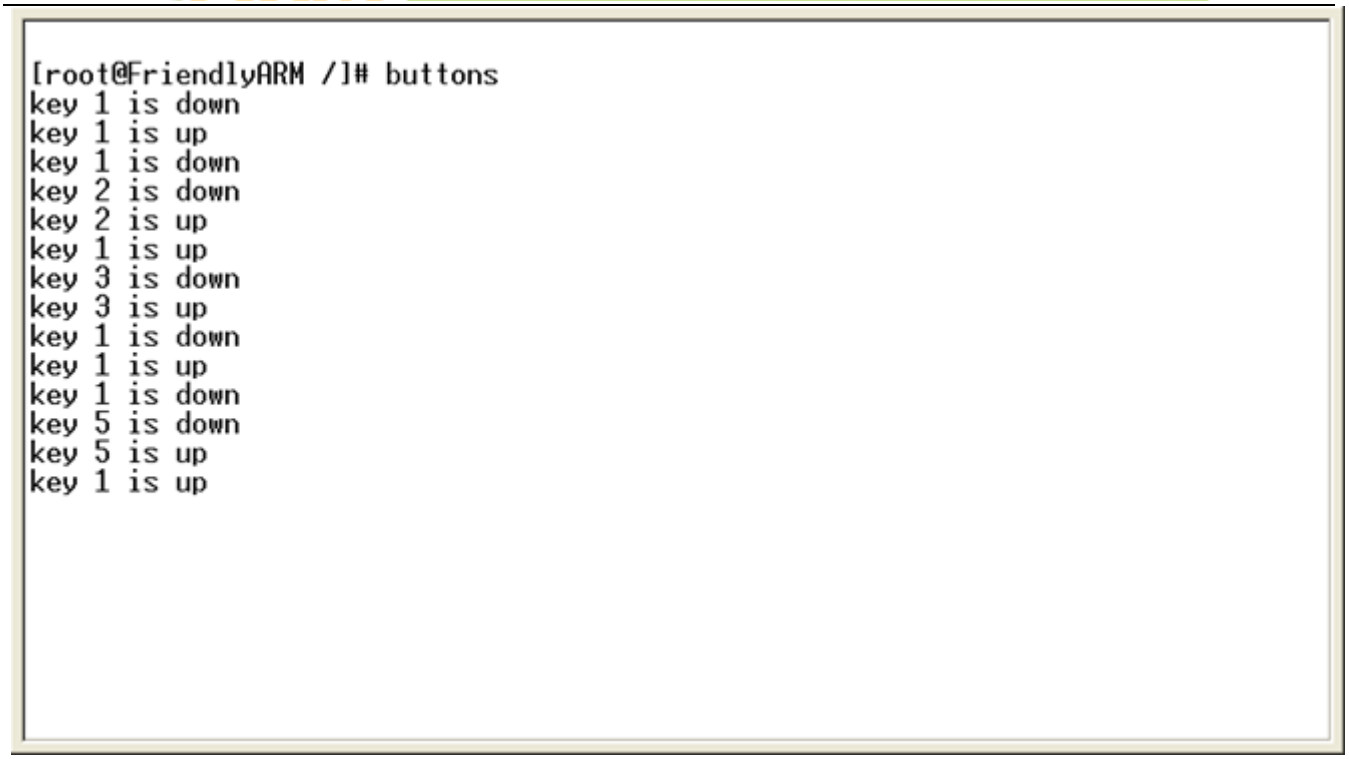
led_no: the LED you want to manipulate (0/1/2/3). "0" and "1" represents "turn off" and "turn on" respectively

```
#led 2 1
```

This will turn on LED3

4.2.5 User Button Test

Type the "**buttons**" command, press a user button and you will see the following scenario



4.2.6 Serial Port Test

Note: the armcomtest utility is a straightforward and easy to use program developed by FriendlyARM for Linux. It doesn't rely on system calls or hardware. After Linux is loaded Serial Ports1, 2, 3 and 4 correspond to **/dev/ttySAC0, 1, 2 and 3.**

To test Serial Port 2 you need a PC with a serial port. Please connect CON2 to the PC via our extension board. Type the following command:

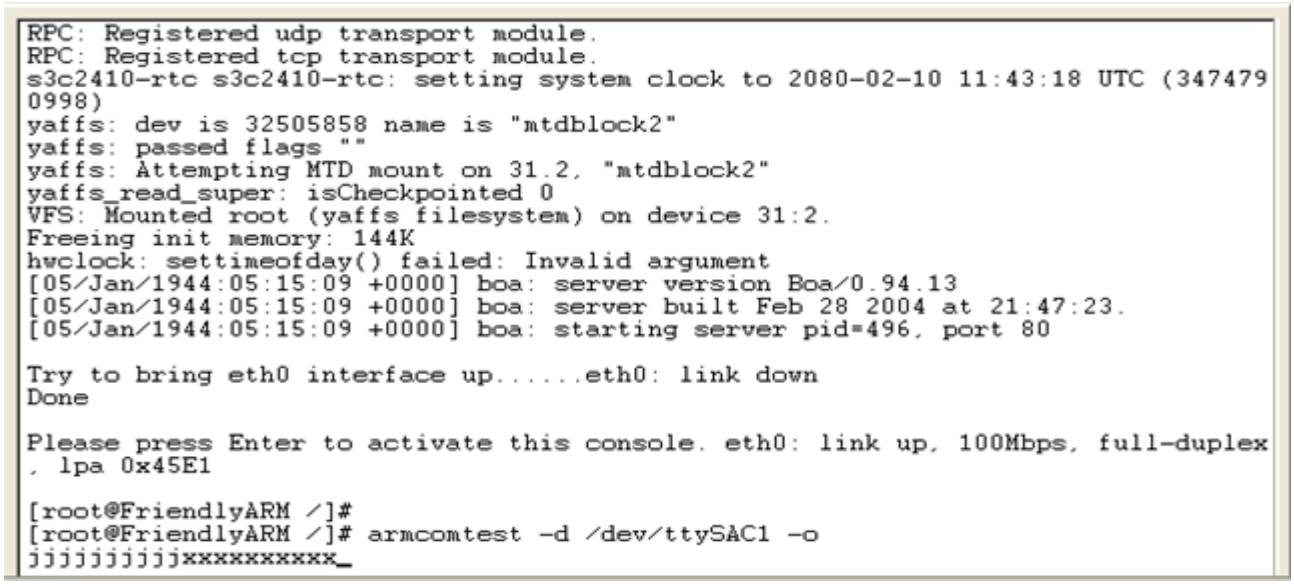
#armcomtest –d /dev/ttySAC1 -o

Now if you type characters (in Serial Port Assistant) on your board they will be output to your PC's super terminal simultaneously and vice versa

To test Serial Port 3 you need to connect CON3 via our extension board and type the command below:

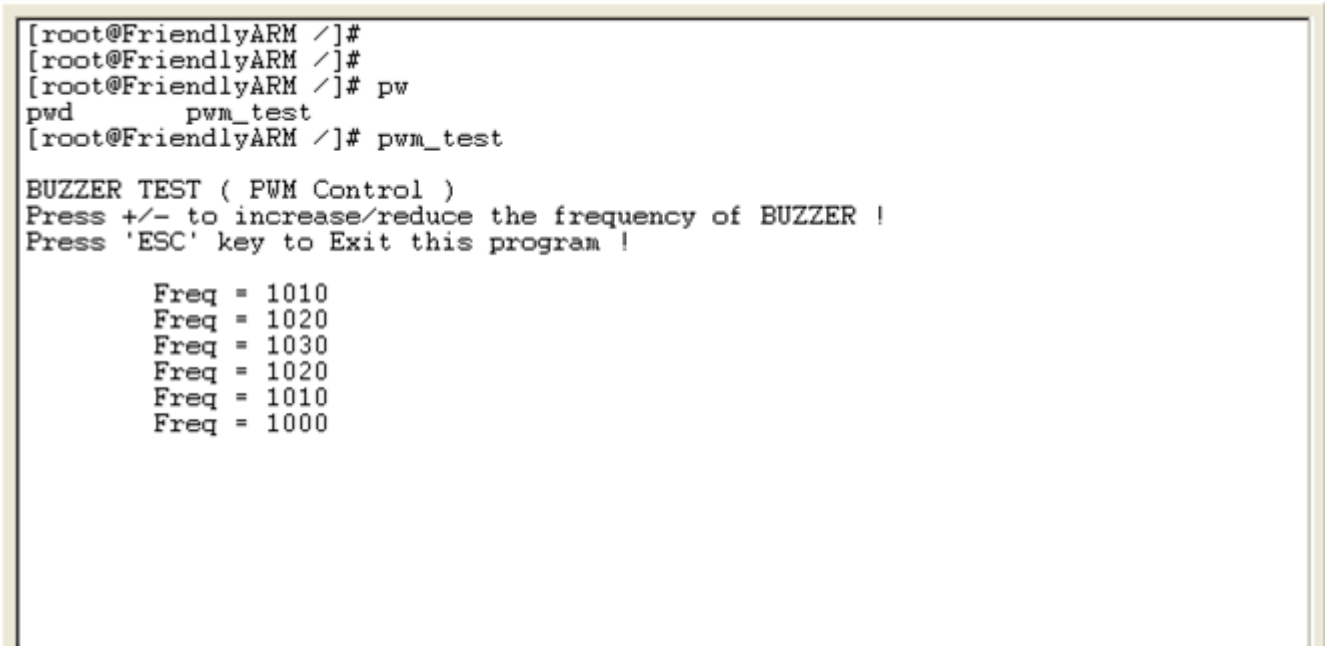
#armcomtest –d /dev/ttySAC2 -o

Here is a screenshot



4.2.7 PWM Buzzer Test

Type “pwm_test” in a terminal and you will be able to hear beeps. Press “+” or “-” you can turn up or down. Press “ESC” to exit.



4.2.8 Backligh Control

Note: the device file for the LCD backlight is /dev/backlight-1wire.

We also implement the feature of adjusting backlight for 1-wire precise touch LCDs. It supports up to 127 levels of adjustment. To turn off the backlight you can feed “0” to the device file as follows:

Type the command: echo 0 > /dev/backlight

When feed 1-127 to the device file you will observe different levels of light. 127 is the highest.In general 15 will begin to show some light. 1-15 makes the screen completely dark. Values higher than 127 will be treated as 127.

Try: echo 15 > /dev/backlight you will be able to see some light.

4.2.9 I2C-EEPROM Test

Type “i2c –w” in a terminal you will be able to write data (0x00-0xff) to 24C08.

```
[root@FriendlyARM /]#
[root@FriendlyARM /]#
[root@FriendlyARM /]# i2c -w
Open /dev/i2c/0 with 8bit mode
  Writing 0x00-0xff into 24C08

 0000|   00 01 02 03 04 05 06 07    08 09 0a 0b 0c 0d 0e 0f
 0010|   10 11 12 13 14 15 16 17    18 19 1a 1b 1c 1d 1e 1f
 0020|   20 21 22 23 24 25 26 27    28 29 2a 2b 2c 2d 2e 2f
 0030|   30 31 32 33 34 35 36 37    38 39 3a 3b 3c 3d 3e 3f
 0040|   40 41 42 43 44 45 46 47    48 49 4a 4b 4c 4d 4e 4f
 0050|   50 51 52 53 54 55 56 57    58 59 5a 5b 5c 5d 5e 5f
 0060|   60 61 62 63 64 65 66 67    68 69 6a 6b 6c 6d 6e 6f
 0070|   70 71 72 73 74 75 76 77    78 79 7a 7b 7c 7d 7e 7f
 0080|   80 81 82 83 84 85 86 87    88 89 8a 8b 8c 8d 8e 8f
 0090|   90 91 92 93 94 95 96 97    98 99 9a 9b 9c 9d 9e 9f
 00a0|   a0 a1 a2 a3 a4 a5 a6 a7    a8 a9 aa ab ac ad ae af
 00b0|   b0 b1 b2 b3 b4 b5 b6 b7    b8 b9 ba bb bc bd be bf
 00c0|   c0 c1 c2 c3 c4 c5 c6 c7    c8 c9 ca cb cc cd ce cf
 00d0|   d0 d1 d2 d3 d4 d5 d6 d7    d8 d9 da db dc dd de df
 00e0|   e0 e1 e2 e3 e4 e5 e6 e7    e8 e9 ea eb ec ed ee ef
 00f0|   f0 f1 f2 f3 f4 f5 f6 f7    f8 f9 fa fb fc fd fe ff

[root@FriendlyARM /]#
```

Type “i2c –r” in a terminal you will be able to read data from 24C08.

```
 00f0|  f0 f1 f2 f3 f4 f5 f6 f7   f8 f9 fa fb fc fd fe ff

[root@FriendlyARM /]# i2c -r
Open /dev/i2c/0 with 8bit mode
  Reading 256 bytes from 0x0

 0000|  00 01 02 03 04 05 06 07   08 09 0a 0b 0c 0d 0e 0f
 0010|  10 11 12 13 14 15 16 17   18 19 1a 1b 1c 1d 1e 1f
 0020|  20 21 22 23 24 25 26 27   28 29 2a 2b 2c 2d 2e 2f
 0030|  30 31 32 33 34 35 36 37   38 39 3a 3b 3c 3d 3e 3f
 0040|  40 41 42 43 44 45 46 47   48 49 4a 4b 4c 4d 4e 4f
 0050|  50 51 52 53 54 55 56 57   58 59 5a 5b 5c 5d 5e 5f
 0060|  60 61 62 63 64 65 66 67   68 69 6a 6b 6c 6d 6e 6f
 0070|  70 71 72 73 74 75 76 77   78 79 7a 7b 7c 7d 7e 7f
 0080|  80 81 82 83 84 85 86 87   88 89 8a 8b 8c 8d 8e 8f
 0090|  90 91 92 93 94 95 96 97   98 99 9a 9b 9c 9d 9e 9f
 00a0|  a0 a1 a2 a3 a4 a5 a6 a7   a8 a9 aa ab ac ad ae af
 00b0|  b0 b1 b2 b3 b4 b5 b6 b7   b8 b9 ba bb bc bd be bf
 00c0|  c0 c1 c2 c3 c4 c5 c6 c7   c8 c9 ca cb cc cd ce cf
 00d0|  d0 d1 d2 d3 d4 d5 d6 d7   d8 d9 da db dc dd de df
 00e0|  e0 e1 e2 e3 e4 e5 e6 e7   e8 e9 ea eb ec ed ee ef
 00f0|  f0 f1 f2 f3 f4 f5 f6 f7   f8 f9 fa fb fc fd fe ff

[root@FriendlyARM /]#
```

4.2.10 A/D Conversion Test

Type “adc-test” in a terminal, you will be able to test AD conversion. By adjusting the W1 resistor you will observe the output.

```
[root@FriendlyARM /]# adc-test
press Ctrl-C to stop
ADC Value: 0
ADC Value: 0
ADC Value: 0
ADC Value: 152
ADC Value: 295
ADC Value: 559
ADC Value: 800
ADC Value: 882
ADC Value: 890
ADC Value: 891
ADC Value: 892
```

4.2.11 WiFi

For users to utilize USB WiFi or SD WiFi cards we originally developed a command set:USB WiFi kits for the Mini2440 system. This command set supports up to one thousand various USB WiFi cards (most of which utilize similar chips). We also integrated this command set into our Tiny4412 system and also integrated the SD WiFi driver.

This command set includes a WiFi driver and three commands:

- scan-wifi – scans nearby wireless networks
- start-wifi – starts connecting to a wireless network
- stop-wifi – closes a wireless connection

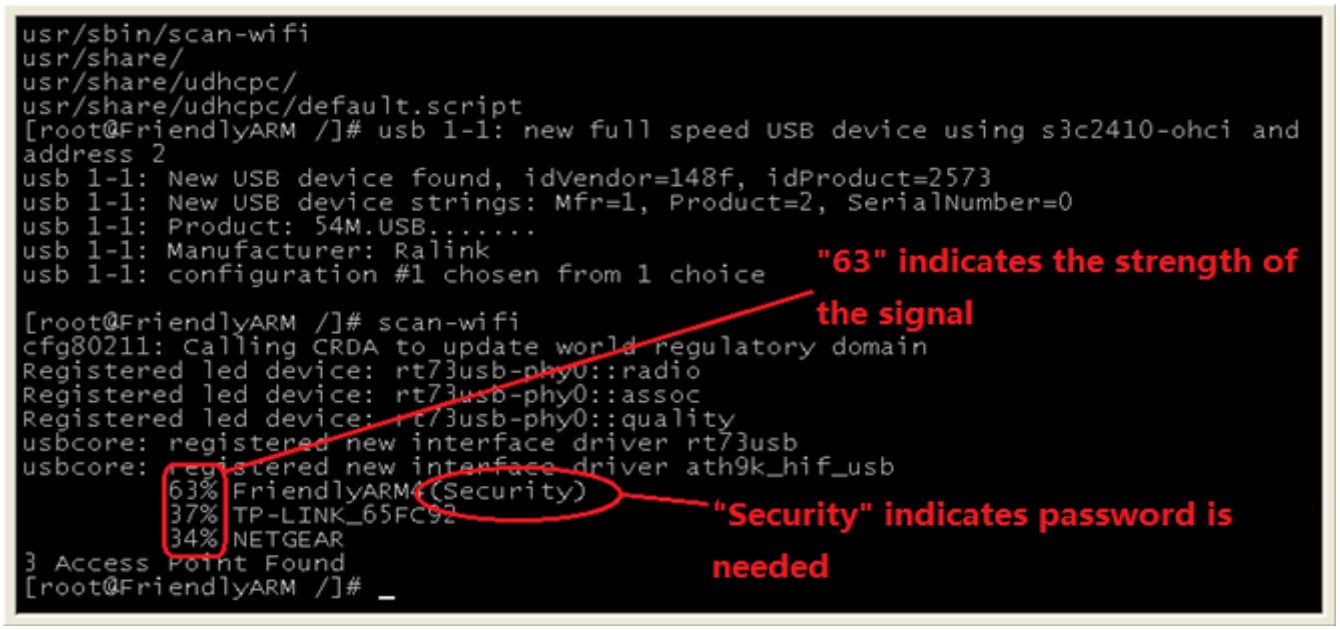
All of them are under the "/usr/sbin" directory

1. Scanning Nearby Wireless Networks

Connect your USB WiFi card to your board you will see the following screenshot

```
lib/modules/2.6.32.2-FriendlyARM/net/wireless/cfg80211.ko
lib/modules/2.6.32.2-FriendlyARM/net/mac80211/
lib/modules/2.6.32.2-FriendlyARM/net/mac80211/mac80211.ko
lib/firmware/
lib/firmware/rt73.bin
lib/firmware/ar9271.fw
usr/
usr/sbin/
usr/sbin/start-wifi
usr/sbin/wpa_supplicant
usr/sbin/stop-wifi
usr/sbin/scan-wifi
usr/share/
usr/share/udhcpc/
usr/share/udhcpc/default.script
[root@FriendlyARM /]# usb 1-1: new full speed USB device using s3c2410-ohci and
address 2
usb 1-1: New USB device found, idVendor=148f, idProduct=2573
usb 1-1: New USB device strings: Mfr=1, Product=2, SerialNumber=0
usb 1-1: Product: 54M.USB.......
usb 1-1: Manufacturer: Ralink
usb 1-1: configuration #1 chosen from 1 choice

[root@FriendlyARM /]# _
```

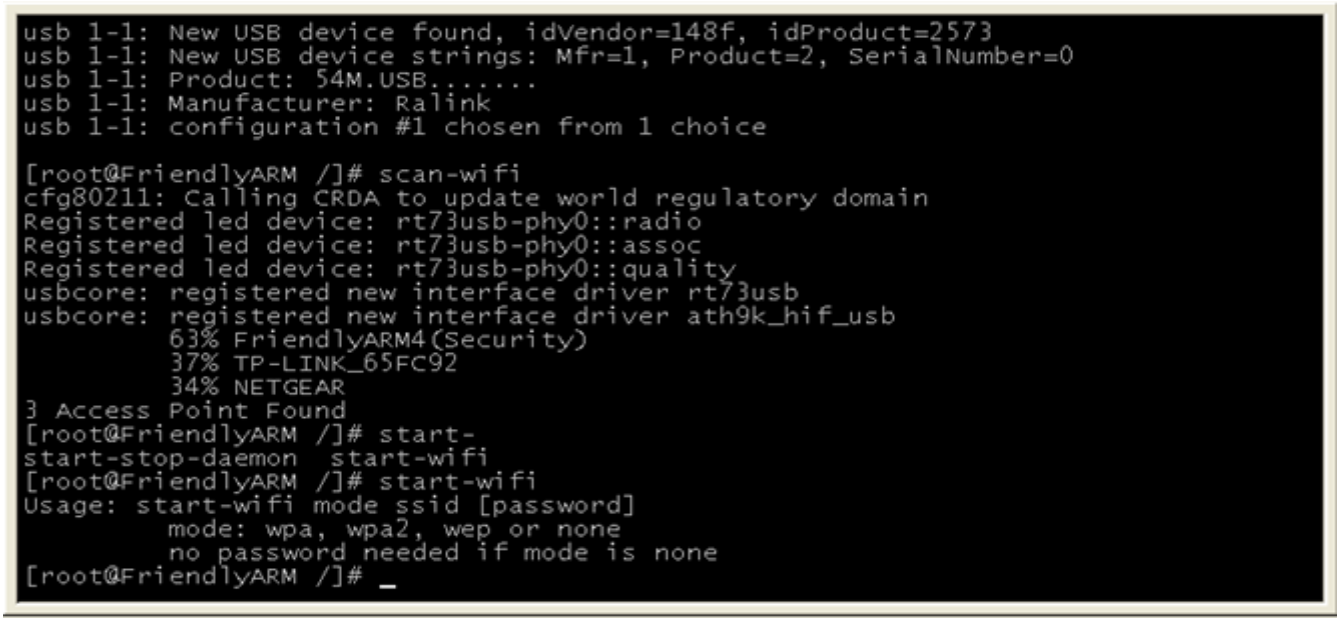
Execute the scan command to search for a network: #scan-wifi



In our example it found three networks and “63%” indicated the strength of the signals. Those that need passwords will be noted by “Security”.

2. Start Connection

“start-wifi” will connect your board to a specified netowork. After execute “start-wifi” you will see the following information:



mode – security type, such as “wpa”, “wpa2”, “wep” or “none”. “None” means that network doesn’t require a password

ssid – network name such as “FriendlyARM4”, “NETGEAR” shown above.

password – password to log in the network.

We will present two examples: one for a network that doesn’t require a password and the other for a network that does require a password

2.1 Connecting to an Open Network that Doesn’t Require a Password

Step1: Command “scan-wifi” to scan your nearby networks. Here we found “FriendlyARM-Test” which was dedicated for this testing

```
[root@FriendlyARM /]# scan-wifi
cfg80211: Calling CRDA to update world regulatory domain
Registered led device: rt73usb-phy0::radio
Registered led device: rt73usb-phy0::assoc
Registered led device: rt73usb-phy0::quality
usbcore: registered new interface driver rt73usb
usbcore: registered new interface driver ath9k_hif_usb
         63% FriendlyARM4(Security)
         37% TP-LINK_65FC92
         34% NETGEAR
3 Access Point Found
[root@FriendlyARM /]# start-
start-stop-daemon  start-wifi
[root@FriendlyARM /]# start-wifi
Usage: start-wifi mode ssid [password]
         mode: wpa, wpa2, wep or none
         no password needed if mode is none
[root@FriendlyARM /]# scan-wifi
         54% FriendlyARM4(Security)
         34% TP-LINK_65FC92
         37% test engineers(Security)
        100% FriendlyARM-Test
4 Access Point Found
[root@FriendlyARM /]# _
```

Step2: Command “start-wifi none FriendlyARM-Test” to connect to this network

```
[root@FriendlyARM /]# start-
start-stop-daemon  start-wifi
[root@FriendlyARM /]# start-wifi
Usage: start-wifi mode ssid [password]
         mode: wpa, wpa2, wep or none
         no password needed if mode is none
[root@FriendlyARM /]# scan-wifi
         54% FriendlyARM4(Security)
         34% TP-LINK_65FC92
         37% test engineers(Security)
        100% FriendlyARM-Test
4 Access Point Found
[root@FriendlyARM /]# start-wifi none FriendlyARM-Test
udhcpc (v1.13.3) started
Sending discover...
Sending discover...
Sending discover...
Sending discover...
Sending select for 192.168.3.100...
Lease of 192.168.3.100 obtained, lease time 7200
deleting routers
route: SIOCDELRT: No such process
adding dns 192.168.3.1
[root@FriendlyARM /]# _
```

Moments later you will notice that your board will be allocated an IP. Here we got "192.168.3.100". "Ping" this network to test your connection.

```
[root@FriendlyARM /]# start-wifi none FriendlyARM-Test
udhcpc (v1.13.3) started
Sending discover...
Sending discover...
Sending discover...
Sending discover...
Sending select for 192.168.3.100...
Lease of 192.168.3.100 obtained, lease time 7200
deleting routers
route: SIOCDELRT: No such process
adding dns 192.168.3.1
[root@FriendlyARM /]# ping 192.168.3.1
PING 192.168.3.1 (192.168.3.1): 56 data bytes
64 bytes from 192.168.3.1: seq=0 ttl=64 time=24.194 ms
64 bytes from 192.168.3.1: seq=1 ttl=64 time=21.053 ms
64 bytes from 192.168.3.1: seq=2 ttl=64 time=20.289 ms
64 bytes from 192.168.3.1: seq=3 ttl=64 time=20.235 ms
64 bytes from 192.168.3.1: seq=4 ttl=64 time=21.180 ms
64 bytes from 192.168.3.1: seq=5 ttl=64 time=20.631 ms
^C
--- 192.168.3.1 ping statistics ---
6 packets transmitted, 6 packets received, 0% packet loss
round-trip min/avg/max = 20.235/21.263/24.194 ms
[root@FriendlyARM /]# _
```

Or you can try this IP in your PC's web browser.

2.2 Connect to a Network That Requires a Password

This procedure is very similar to the previous one but you need to know the network's security type and its password:

Step1: Select your network's security type. (The router we used in this example was TL-WR740N.) Open its web page.

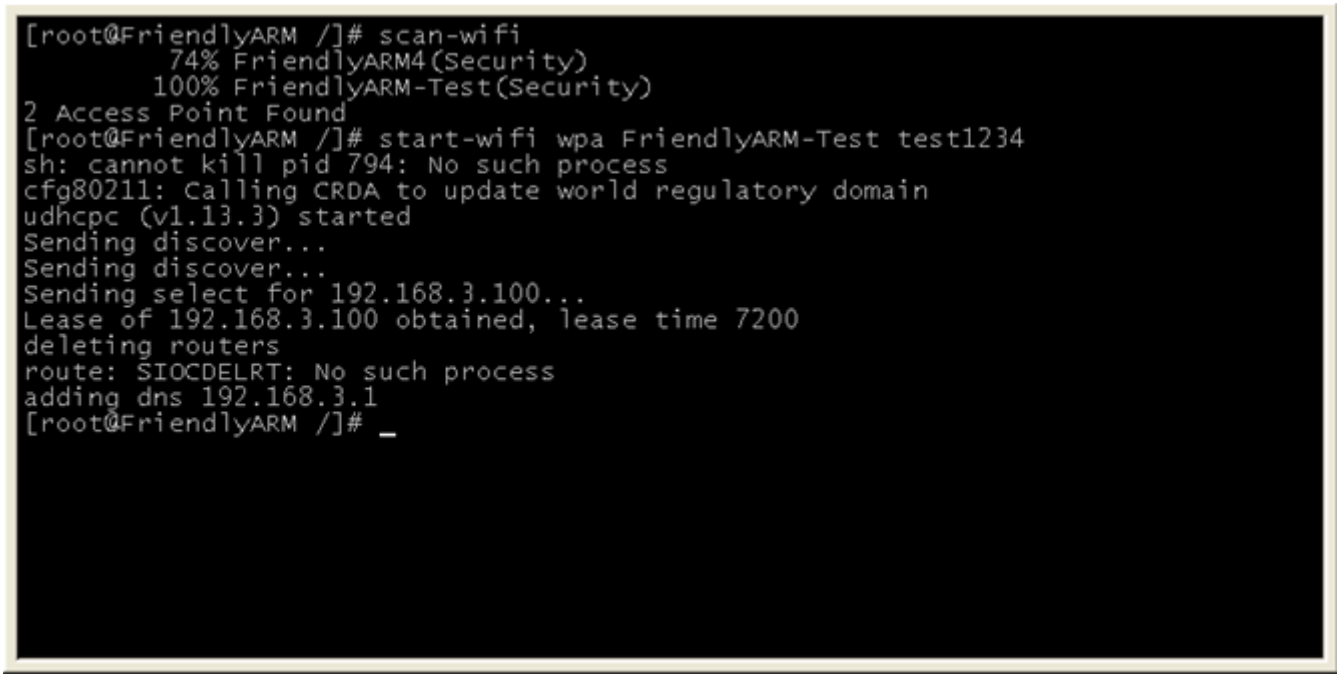
You can choose from the following three:

- WPA-PSK/WPA2-PSK
- WPA/WPA2
- WEP

In our example we selected “WPA” and the password was “test1234”. After that click on “Save” and reboot your router. Note: on how to configure your router you need to refer to your router’s manual.

Step2: Command “scan-wifi” to scan your nearby networks. Here we found “FriendlyARM-Test” which was dedicated for this testing

Step3: Command “start-wifi wpa FriendlyARM-Test test1234” to connect to this network.



Moments later you will notice that your board will be allocated an IP. Here we got "192.168.3.100". "Ping" this network to test your connection

```
[root@FriendlyARM /]# scan-wifi
         74% FriendlyARM4(Security)
        100% FriendlyARM-Test(Security)
2 Access Point Found
[root@FriendlyARM /]# start-wifi wpa FriendlyARM-Test test1234
sh: cannot kill pid 794: No such process
cfg80211: Calling CRDA to update world regulatory domain
udhcpc (v1.13.3) started
Sending discover...
Sending discover...
Sending select for 192.168.3.100...
Lease of 192.168.3.100 obtained, lease time 7200
deleting routers
route: SIOCDELRT: No such process
adding dns 192.168.3.1
[root@FriendlyARM /]# ping 192.168.3.1
PING 192.168.3.1 (192.168.3.1): 56 data bytes
64 bytes from 192.168.3.1: seq=0 ttl=64 time=31.870 ms
64 bytes from 192.168.3.1: seq=1 ttl=64 time=76.107 ms
64 bytes from 192.168.3.1: seq=4 ttl=64 time=42.121 ms
```

Or you can try this IP in your PC's web browser

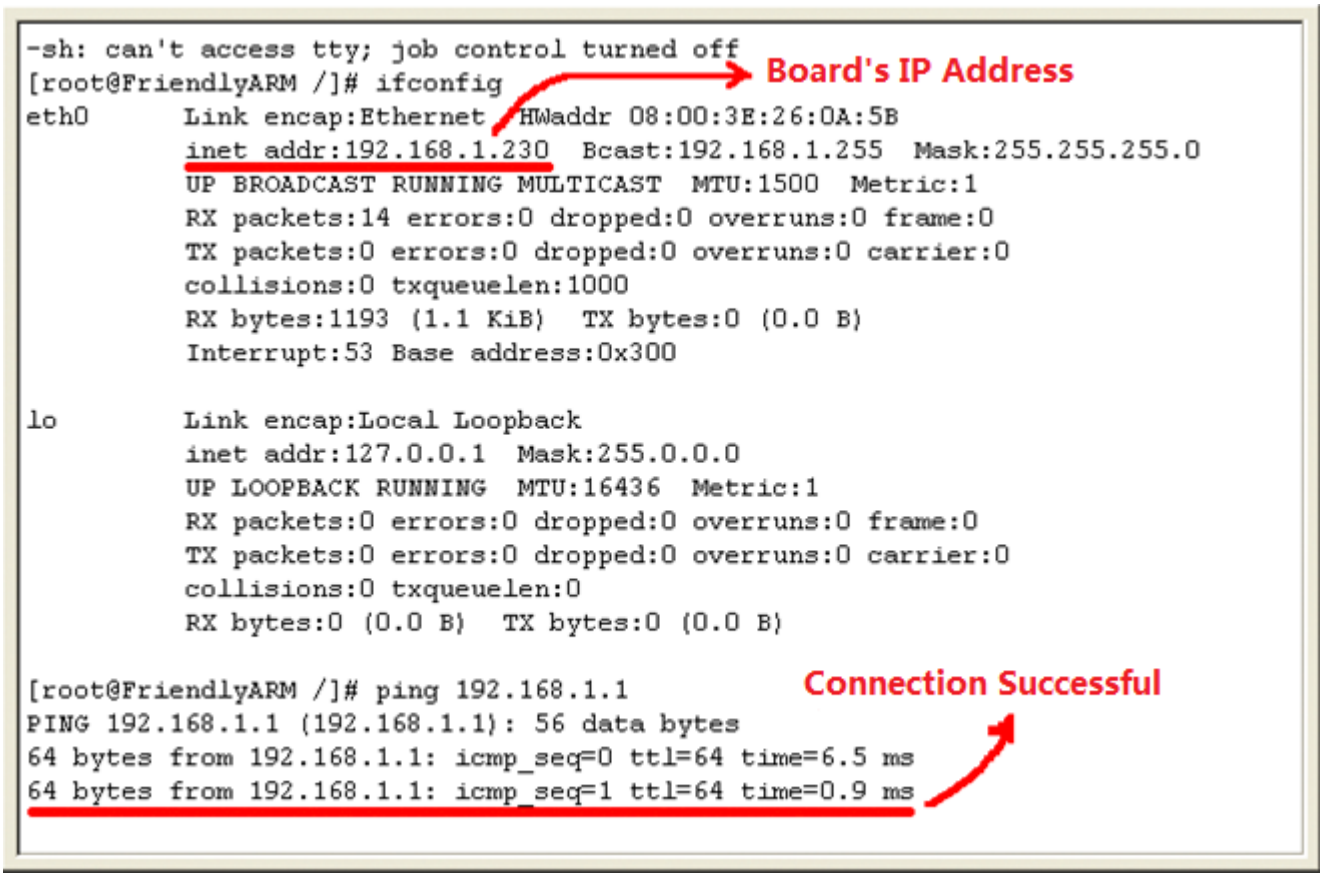
3. Close USB WiFi Connection

Command “stop-wifi” to close your USB WiFi connection

4.2.12 Telnet

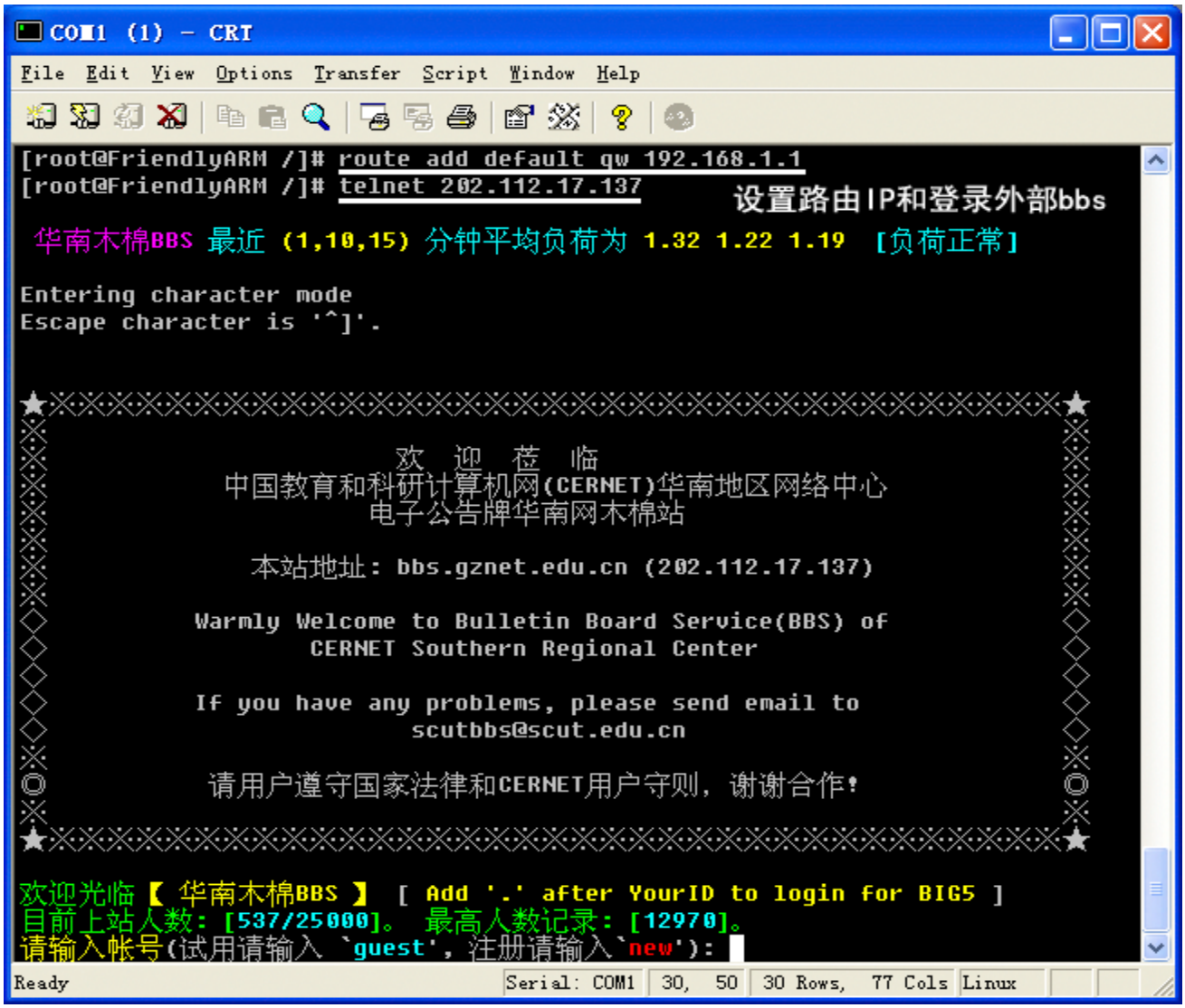
“Telnet” is a popular utility. If your board is connected to the internet you can telnet a bbs.

First make sure your board’s IP is 192.168.1.230 and your board is communicating with other machines.



Then configure your router’s IP: **route add default gw 192.168.1.1**

Now you can telnet a BBS. Here we visited “bbs.scut.edu.cn”.



4.2.13 Ethernet Configuration

Connect your board to the internet, write down your gateway IP(the one in our example was 192.168.1.1) and configure your router:

route add default gw 192.168.1.1

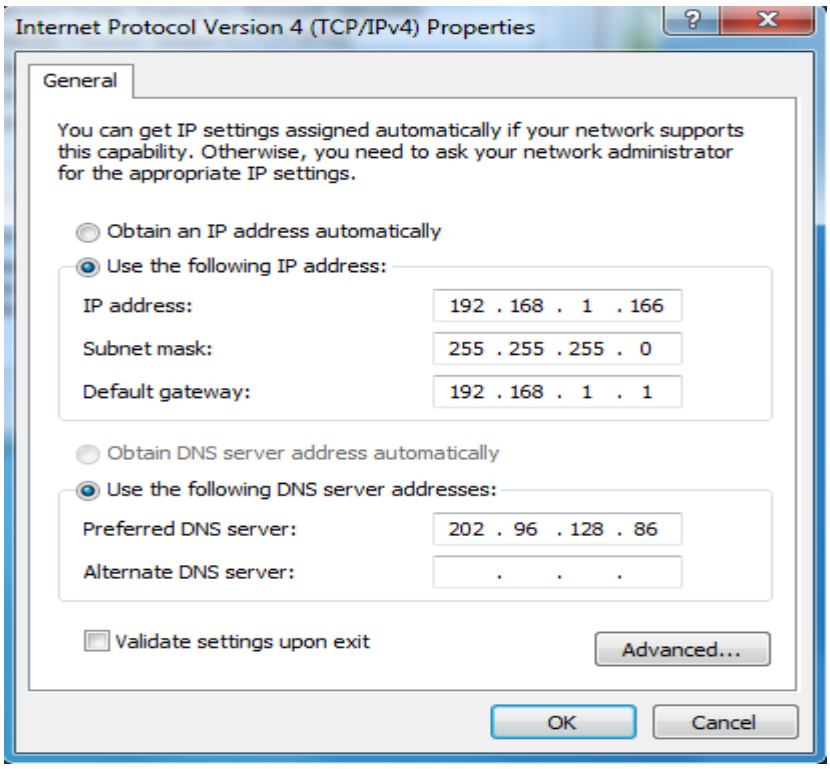
Now you can visit an IP address on the internet e.g. you can ping bbs.scut.edu.cn (IP: 202.112.17.137):

#ping 202.112.17.137

If it is a success you will see the following output

```
[root@FriendlyARM /]# route add default gw 192.168.1.1
[root@FriendlyARM /]# ping 202.112.17.137
PING 202.112.17.137 (202.112.17.137): 56 data bytes
64 bytes from 202.112.17.137: icmp_seq=0 ttl=52 time=1509.6 ms
64 bytes from 202.112.17.137: icmp_seq=1 ttl=52 time=1426.0 ms
64 bytes from 202.112.17.137: icmp_seq=2 ttl=52 time=1446.8 ms
_
```

To ping through an outside website you also need to configure your DNS. You may get it from your network manager



The one in our example was “202.96.128.86”. Therefore we set our board as follows:

#rm /etc/resolv.conf; This is to remove the existing configuration file.

#touch /etc/resolv.conf; This is to generate a resolv.conf file

#echo nameserver 202.96.128.86 >> /etc/resolv.conf; Set up the DNS configuration file resolv.conf with your DNS IP or you can edit it with vi.

```
[root@FriendlyARM /]# rm /etc/resolv.conf
[root@FriendlyARM /]# touch /etc/resolv.conf
[root@FriendlyARM /]# echo nameserver 202.96.128.86 >> /etc/resolv.conf
[root@FriendlyARM /]# cat /etc/resolv.conf
nameserver 202.96.128.86
[root@FriendlyARM /]# ping www.163.com
PING www.cache.split.netease.com (220.181.28.54): 56 data bytes
64 bytes from 220.181.28.54: icmp_seq=0 ttl=53 time=1353.8 ms
64 bytes from 220.181.28.54: icmp_seq=1 ttl=53 time=1378.0 ms
64 bytes from 220.181.28.54: icmp_seq=2 ttl=53 time=1398.1 ms
64 bytes from 220.181.28.54: icmp_seq=4 ttl=53 time=1356.0 ms
64 bytes from 220.181.28.54: icmp_seq=5 ttl=53 time=1314.9 ms

--- www.cache.split.netease.com ping statistics ---
7 packets transmitted, 5 packets received, 28% packet loss
round-trip min/avg/max = 1314.9/1360.1/1398.1 ms
[root@FriendlyARM /]# _
```

4.2.14 Configure MAC Address

The MAC address in the Tiny4412 is “soft” therefore you can change it via “ifconfig”.

First check your current MAC address via “ifconfig”:

#ifconfig ;

```
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.1.0     *               255.255.255.0   U     0      0        0 eth0
default         192.168.1.1     0.0.0.0         UG    0      0        0 eth0
[root@FriendlyARM /]# cat /etc/resolv.conf
nameserver 192.168.1.1
[root@FriendlyARM /]# ifconfig
eth0      Link encap:Ethernet  HWaddr 08:90:90:90:90:90
          inet addr:192.168.1.230  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:34 errors:0 dropped:0 overruns:0 frame:0
          TX packets:15 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:5236 (5.1 KiB)  TX bytes:977 (977.0 B)
          Interrupt:51

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)

[root@FriendlyARM /]# _
```

In our example the MAC was “08：90：90：90：90：90”, this is the default MAC address and has been hard-coded in the kernel. If you want to update it you have to recompile the kernel. In order to change the MAC dynamically you need to close your network connection and then fill your new MAC:

#ifconfig eth0 down

#ifconfig eth0 hw ether 00:11:AA:BB:CC:DD; note: **a,b,c,d,e,f…** could be lower case

Restart the network, check your MAC via “ifconfig” and verify your network via “ping”:

#ifconfig eth0 up

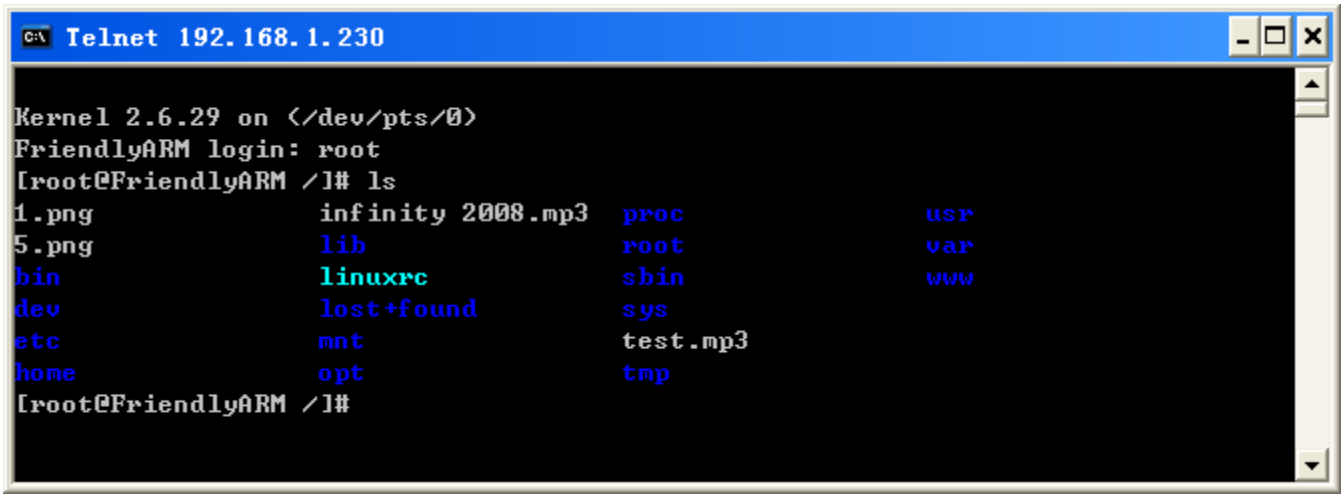
#ifconfig

#ping 192.168.1.1

```
[root@FriendlyARM /]# ifconfig eth0 hw ether 00:11:aa:bb:cc:dd
[root@FriendlyARM /]# ifconfig eth0 up
[root@FriendlyARM /]# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:11:AA:BB:CC:DD
          inet addr:192.168.1.230  Bcast:192.168.1.255  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:8 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:4791 (4.6 KiB)  TX bytes:672 (672.0 B)
          Interrupt:53 Base address:0x300

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)

[root@FriendlyARM /]# ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1): 56 data bytes
64 bytes from 192.168.1.1: icmp_seq=0 ttl=64 time=3.2 ms
```

4.2.15 Telnet Tiny4412

If the system reboots normally it will automatically start a telnet service therefore users can telnet the board too. You can try typing "**telnet 192.168.1.230**" from a command line, type "root" and you will be able to login.

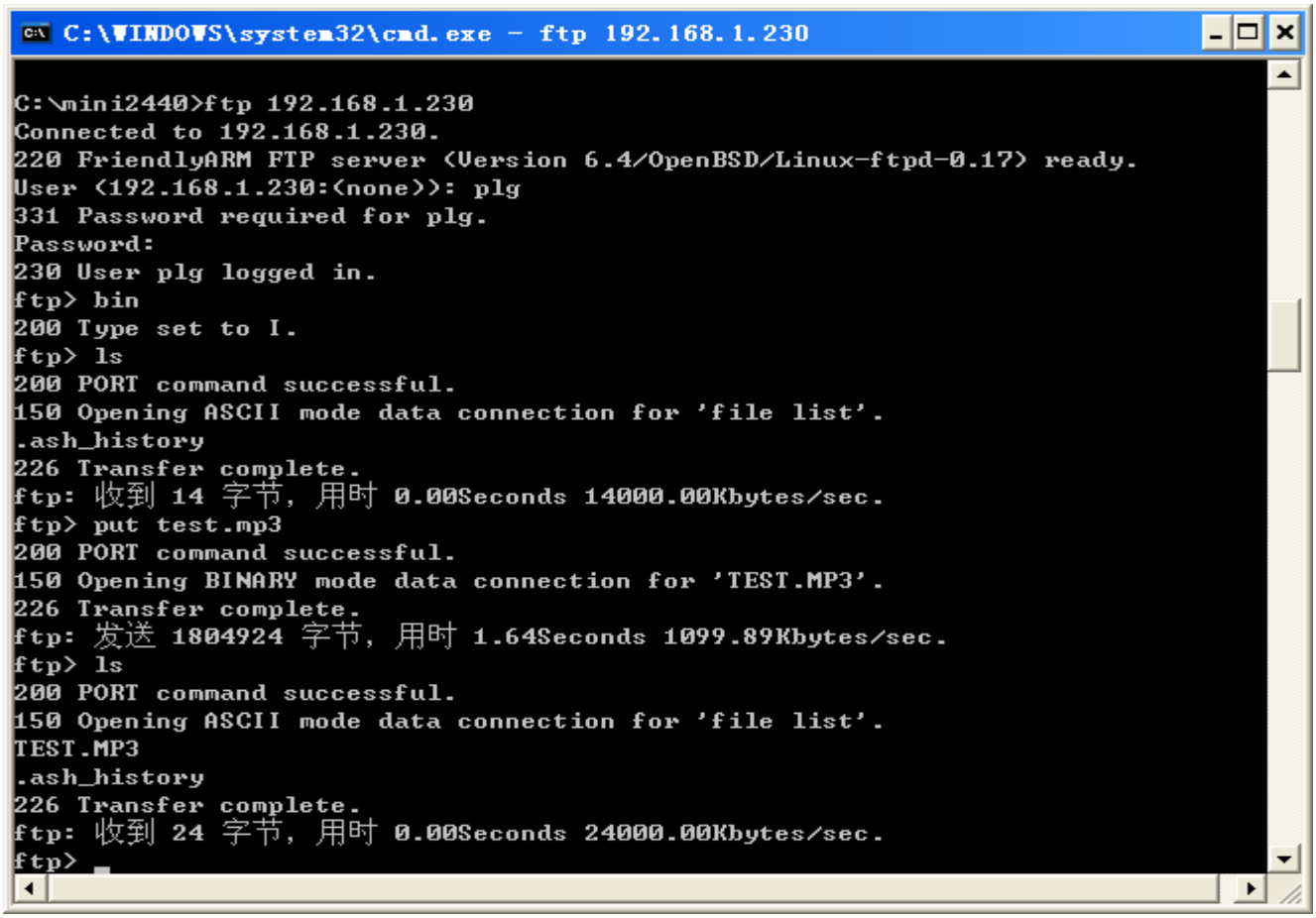


4.2.16 FTP

After the system boots normally, it will automatically start a telnet service. Users can ftp a remote host via “ftp” in the command line utility in both Linux and Windows. Users can transfer files to the board from a host PC.

Note: please make sure you have a file ready in your FTP directory. Here we had “test.mp3”.The account for login is plg and the password is plg.

After file transfer is doen you will see a test.mp3 file in your target board’s /home/plg directory.



4.2.17 LED Control via Web

Click on “Manipulating LEDs via HTML” on the test page of our web server, the following page will be loaded

You can test each of these items. The “LED Test” will manipulate the LEDs via CGI programs and it includes two display modes and three display rates

To stop the web service you need to type the following commands:

#/etc/rc.d/init.d/httpd stop

Then restart the service

#/etc/rc.d/init.d/httpd start

4.2.18 Configure System Clock

The Linux command for updating time is "**date**", to synchronize the S3C6420 time with Linux's system time you can use "**hwclock**":

(1) date -s 042916352007 #set time to 2007-04-29 16:34

(2) hwclock -w # save your setting to 4412's RTC

(3) Command "hwclock –s" to update Linux's system time with RTC. Usually this command will be included in "/etc/init.d/rcS" for auto run

Note: our system's "/etc/init.d/rcS" includes "*hwclock –s*" already.

4.2.19 Save Date to Flash Permenantly

The Tiny4412 system applies yaffs2 thus can save data dynamically and will not lose any even when the system is powered off. After the system boots, please try the following command:

#cp / shanghaitan.mp3 /home/plg

This will create a duplicate file under "/home/plg". Power off and on you will observe that the file still exists.

4.2.20 Setup Auto Run Program

Users can set up programs that will be automatically run on system startup in the boot script. It is similar to Window's Autobat. It is under the /etc/init.d/rcS directory, the contents are as follows (they may be different in differed systems)

```
#! /bin/sh
```

```
PATH=/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/bin:
runlevel=S
prevlevel=N
umask 022
export PATH runlevel prevlevel
#
# Trap CTRL-C &c only in this shell so we can interrupt subprocesses.
#
trap ":" INT QUIT TSTP
/bin/hostname FriendlyARM
[ -e /proc/1 ] || /bin/mount -n -t proc none /proc
[ -e /sys/class ] || /bin/mount -n -t sysfs none /sys
[ -e /dev/tty ] || /bin/mount -t ramfs none /dev
/bin/mount -n -t usbfs none /proc/bus/usb
echo /sbin/mdev > /proc/sys/kernel/hotplug
/sbin/mdev -s
/bin/hotplug
# mounting file system specified in /etc/fstab
mkdir -p /dev/pts
mkdir -p /dev/shm
/bin/mount -n -t devpts none /dev/pts -o mode=0622
/bin/mount -n -t tmpfs tmpfs /dev/shm
/bin/mount -n -t ramfs none /tmp
/bin/mount -n -t ramfs none /var
mkdir -p /var/empty
mkdir -p /var/log
mkdir -p /var/lock
mkdir -p /var/run
mkdir -p /var/tmp
/sbin/hwclock -s
syslogd
/etc/rc.d/init.d/netd start
echo " " > /dev/tty1
echo "Starting networking..." > /dev/tty1
sleep 1
/etc/rc.d/init.d/httpd start
echo " " > /dev/tty1
echo "Starting web server..." > /dev/tty1
sleep 1
/etc/rc.d/init.d/leds start
echo " " > /dev/tty1
echo "Starting leds service..." > /dev/tty1
```

```
echo " "
sleep 1
echo " " > /dev/tty1
/etc/rc.d/init.d/alsaconf start
echo "Loading sound card config..." > /dev/tty1
echo " "
/sbin/ifconfig lo 127.0.0.1
/etc/init.d/ifconfig-eth0
/bin/qtopia &
echo " " > /dev/tty1
echo "Starting Qtopia, please waiting..." > /dev/tty1
```

4.2.21 Take Screenshot with Snapshot

Users can take screenshots with “snapshot” and save them as png files

#snapshot pic.png

Executing this command will take a screenshot of the current LCD display and save it as “pic.png”.

4.2.22 Check RAM Info

The Tiny4412 system incorporates a 1G DDR3 RAM. Some users complained that they can only find less than 1G available this is because the multi-media driver takes quite a lot. In general users can check the RAM info by commanding “cat /proc/meminfo”, however this only shows the amount available to the system.

4.3 Setup Fedora9 Development Environment

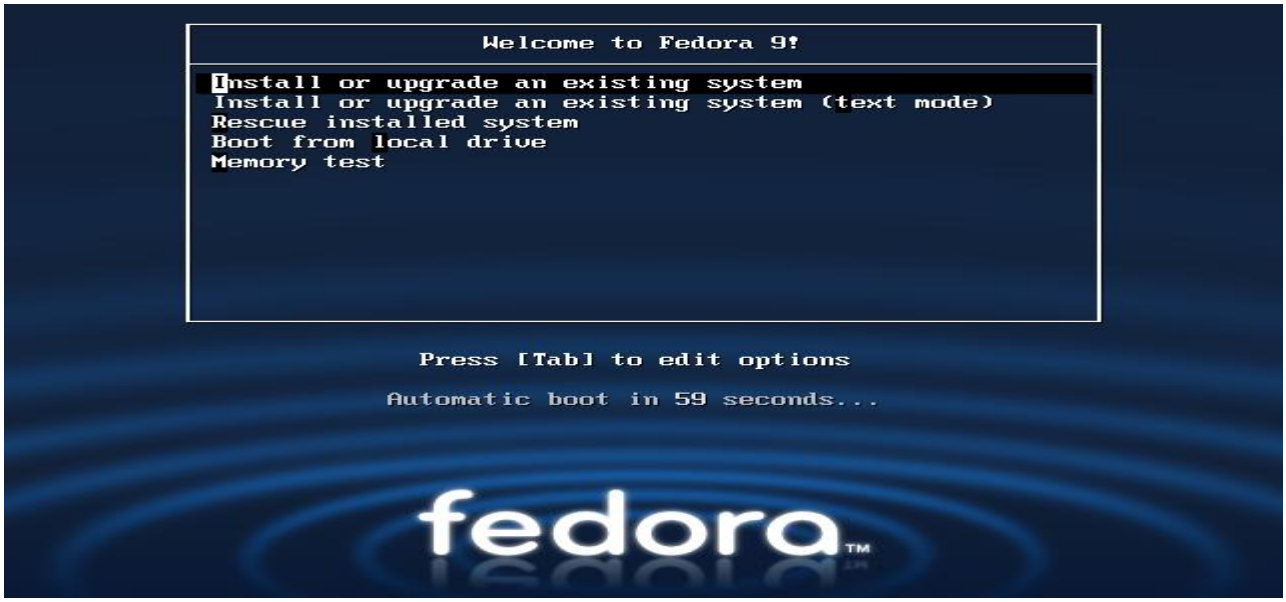
This section will guide you through the steps on how to install Fedora 9.0 on a PC and set up your Linux development environment. All our software development and

testing for the Tiny4412 were based on Fedora 9.0. We didn't test it on other platforms. We strongly suggest you use this platform as we do, which you can download from its website (ftp://download.fedora.redhat.com/pub/fedora/linux/releases/9/Fedora/i386/iso/Fedora-9-i386-DVD.iso).

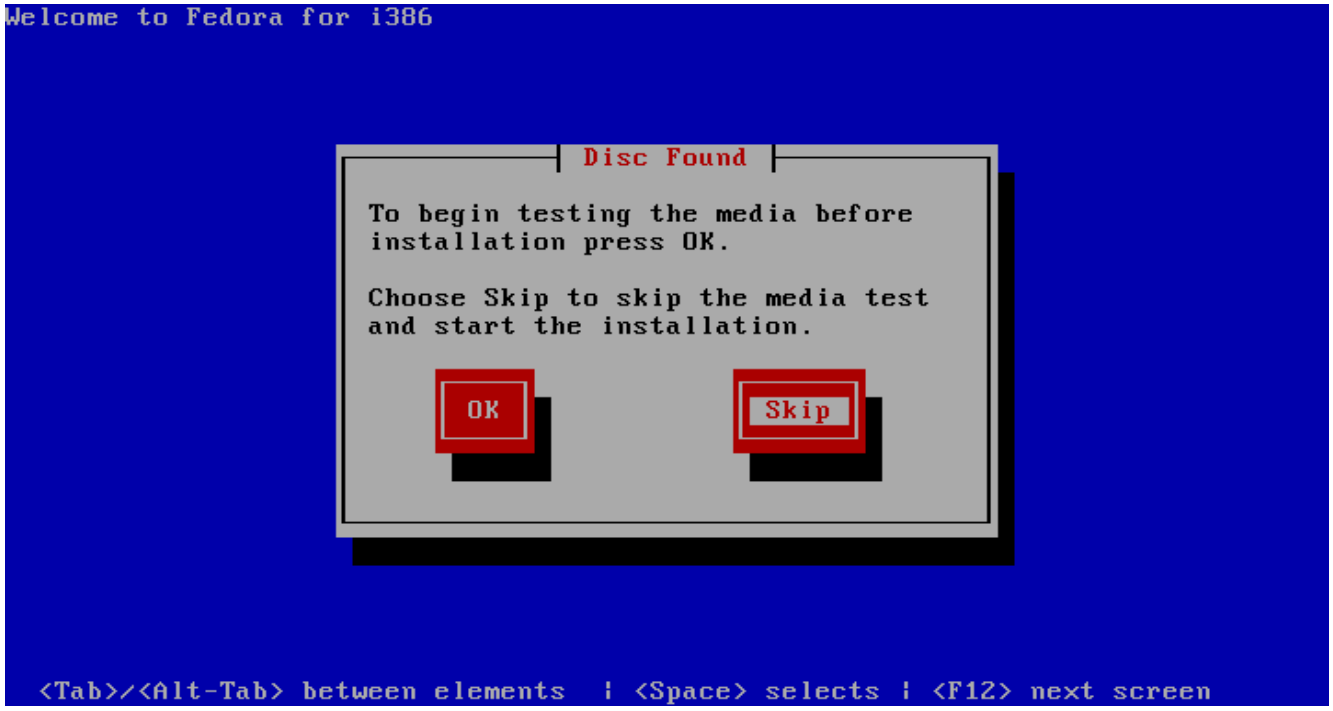
The reason why we chose Fedora 9.0 is that it is easy to be installed and set up. Fedora 10 and later versions are more complicated and therefore may not be easy for beginners and Fedora 8 and earlier versions are a little bit obsolete. Please follow the steps below to install.

4.3.1 Install Fedora 9.0

Step1： Insert the first disk in the CDROM/DVD, set the boot sequence to CDROM in the BIOS. After reboot the system it will prompt the user to the following interface, just press "enter"

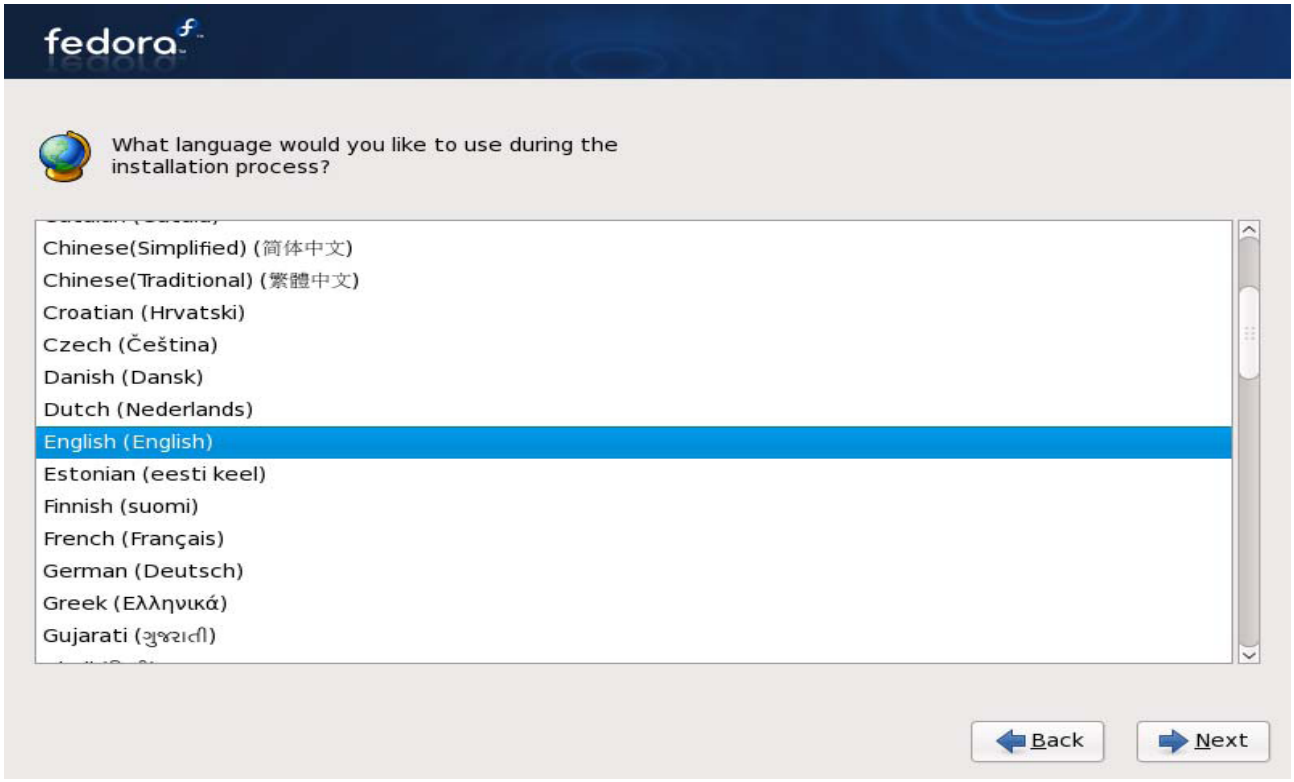


Step2：The system will check the installation disk. It can be ignored, just press “Skip” to the next step

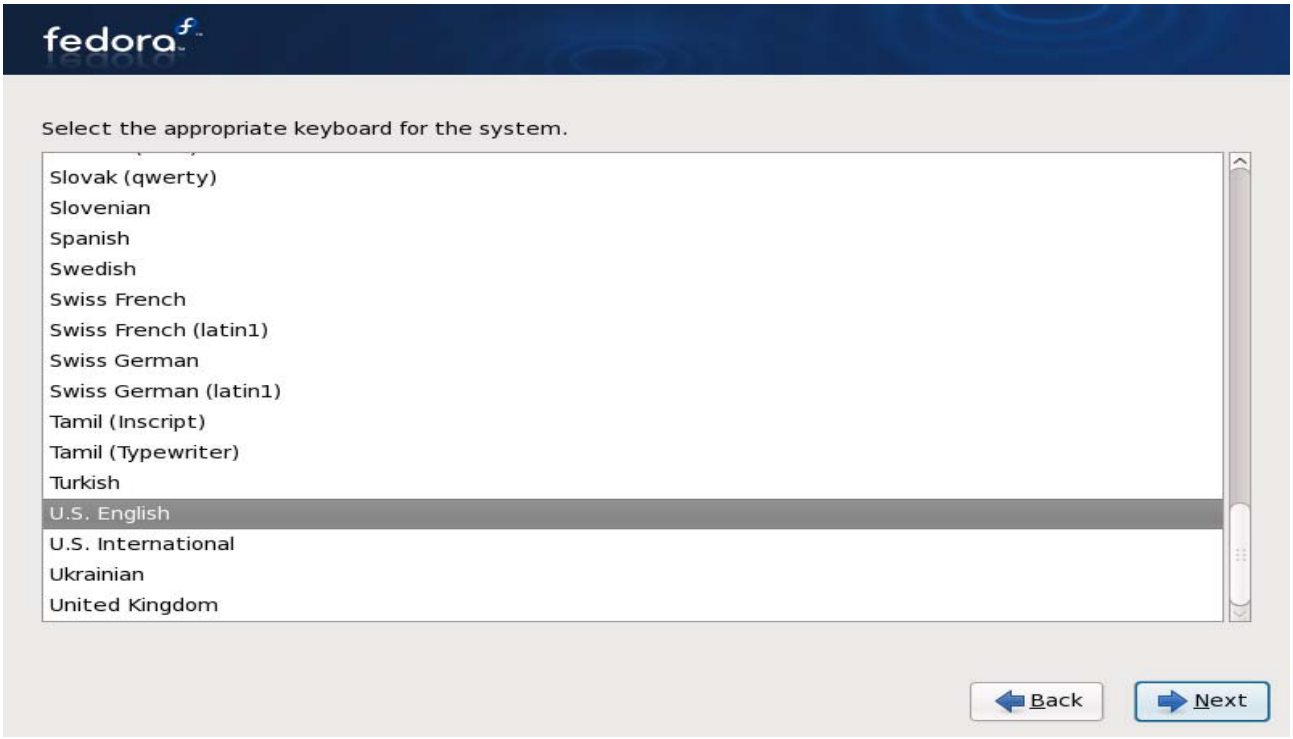


Step3：it enters the graphic interface, click on the “Next” button.

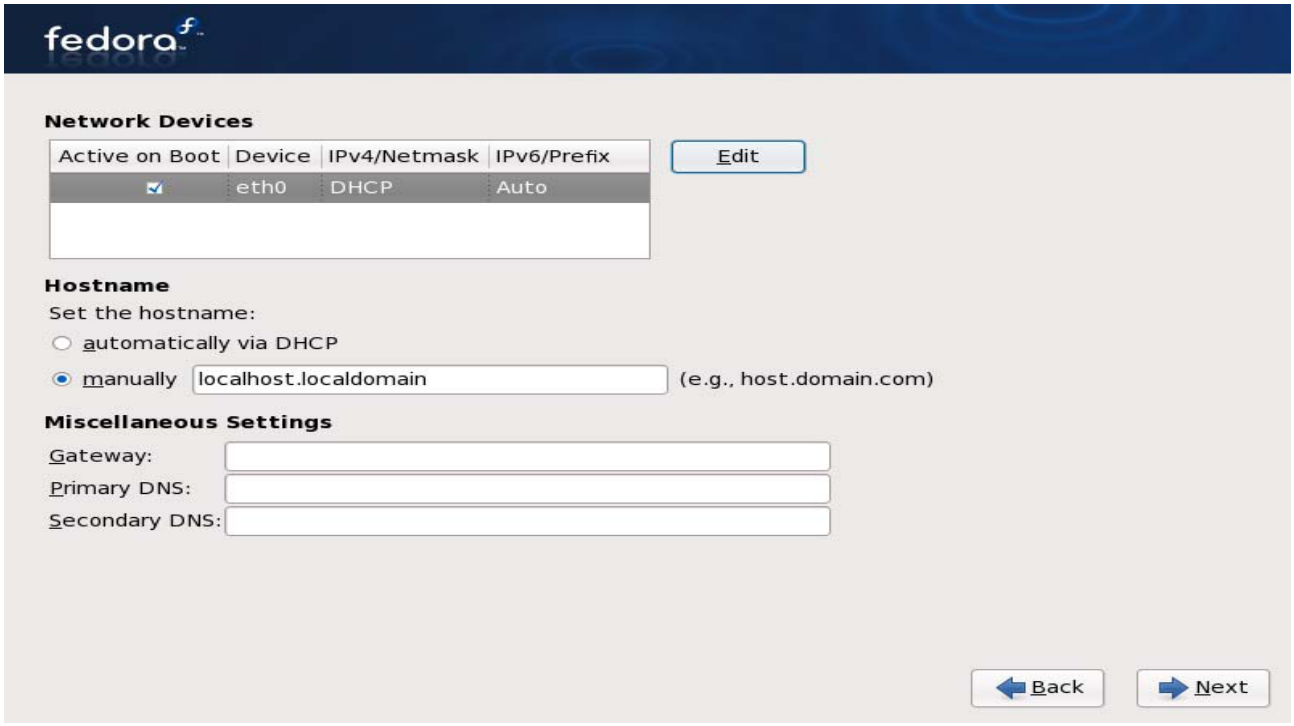
Step4：set the installation language. In this example, we chose the simplified English.



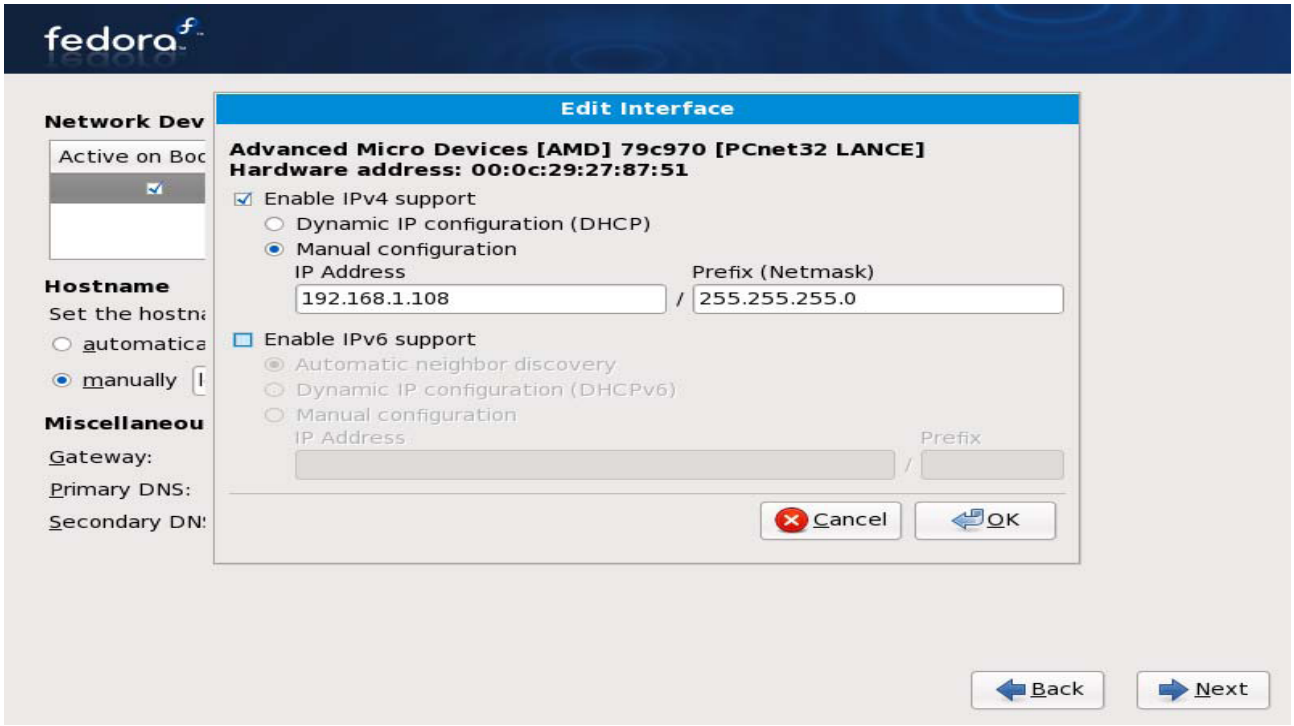
Step5：set the keyboard, in this example, we chose the U.S. key board.



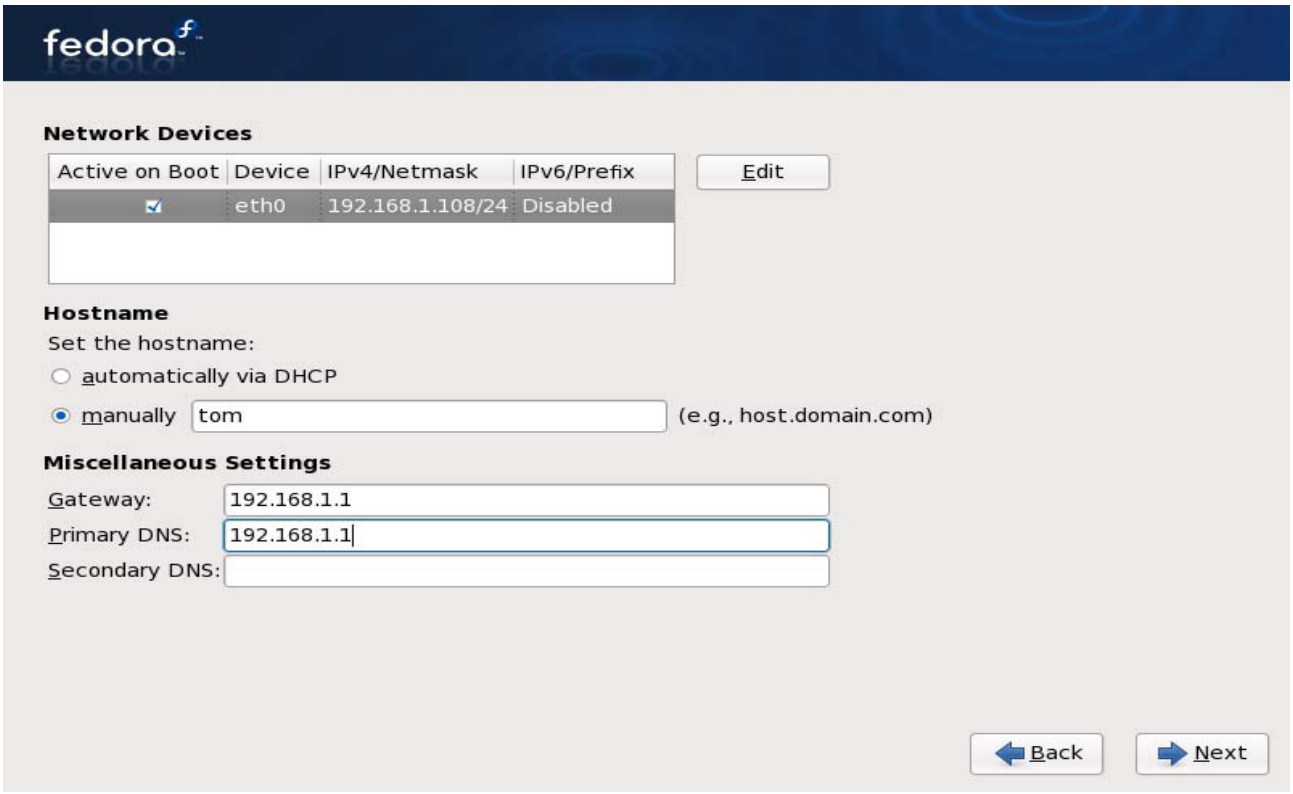
Step 6：configure the network.



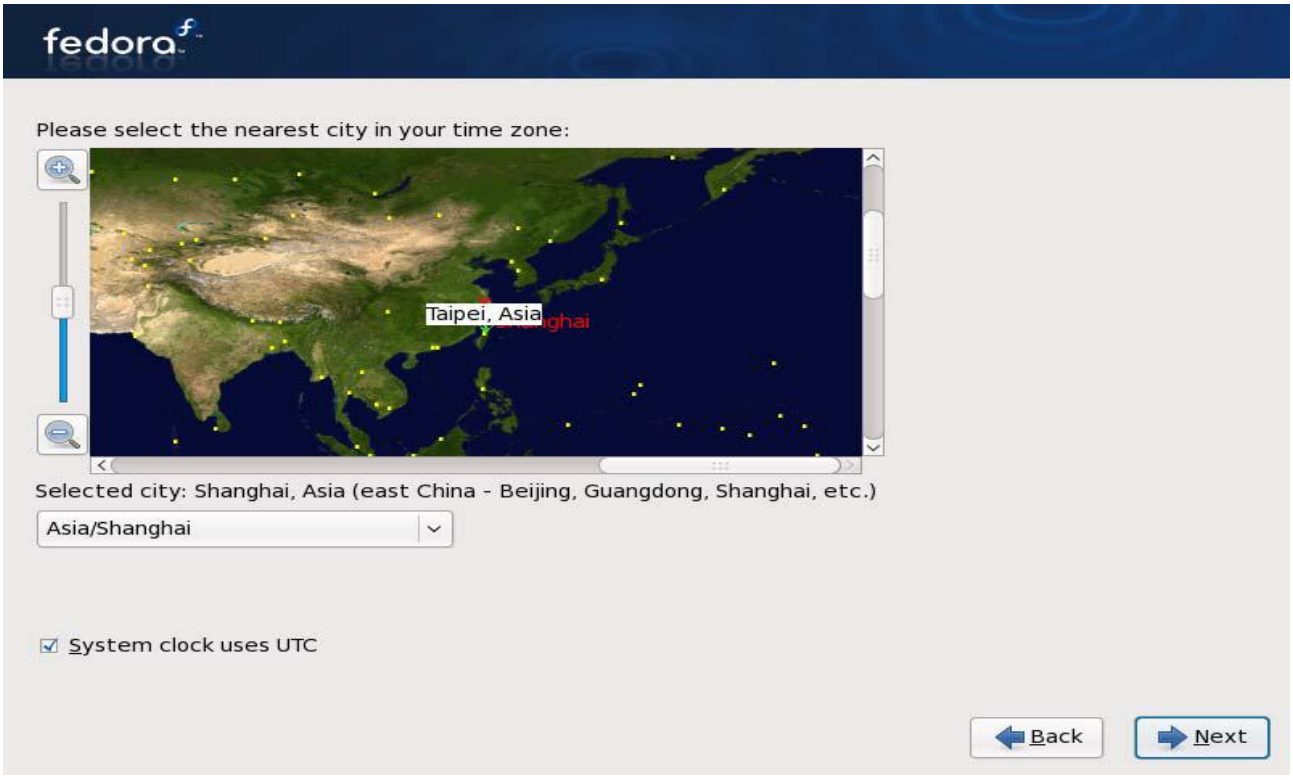
In our example, we didn't set it as "DHCP", we used a static IP instead, and typed the IP and subnet mask as follows.



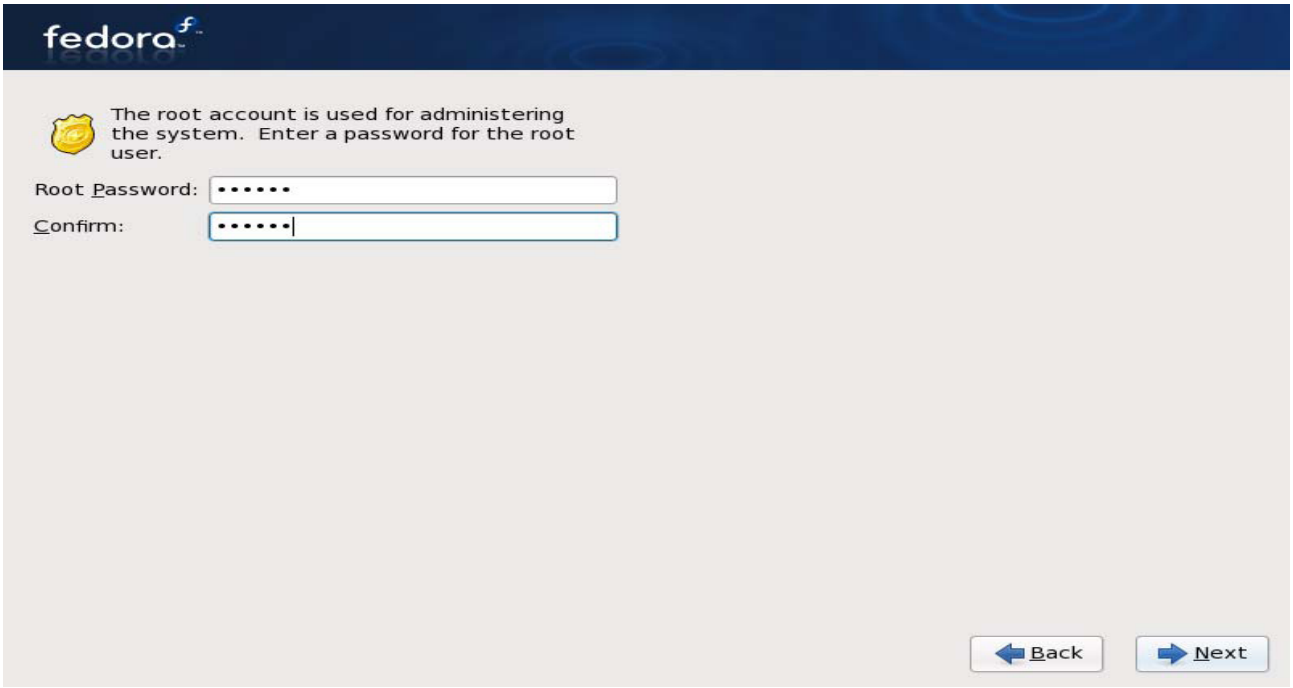
Click on the OK button and go on to set the machine name, gateway and DNS.



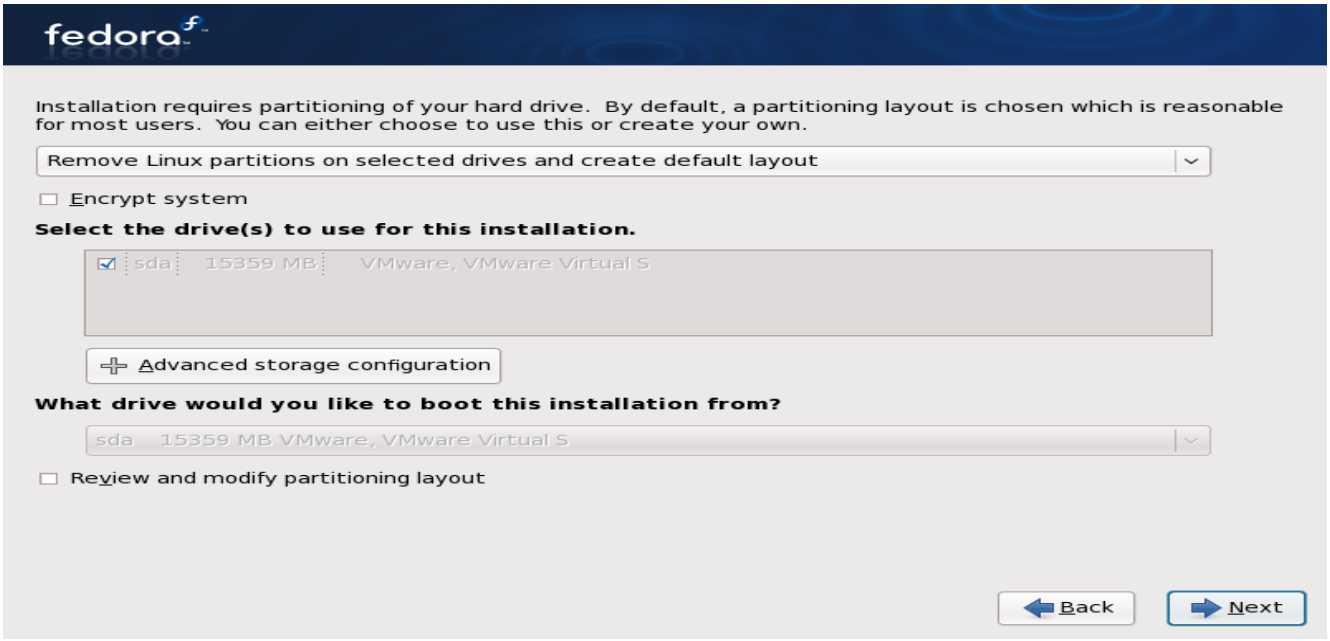
Step 7：set the time zone. We chose “Asia/Shanghai”.



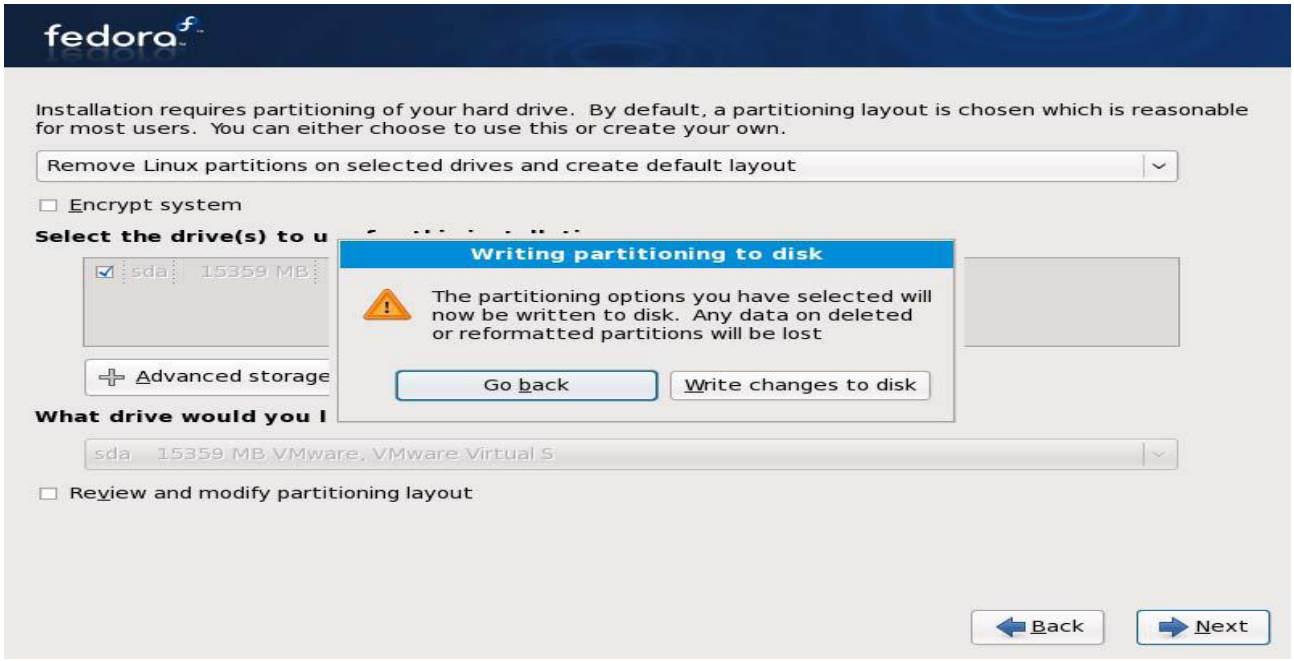
Step 8：set up the administrator's password, i.e. the root's password. "root" is the super user. It should be at least 6 characters



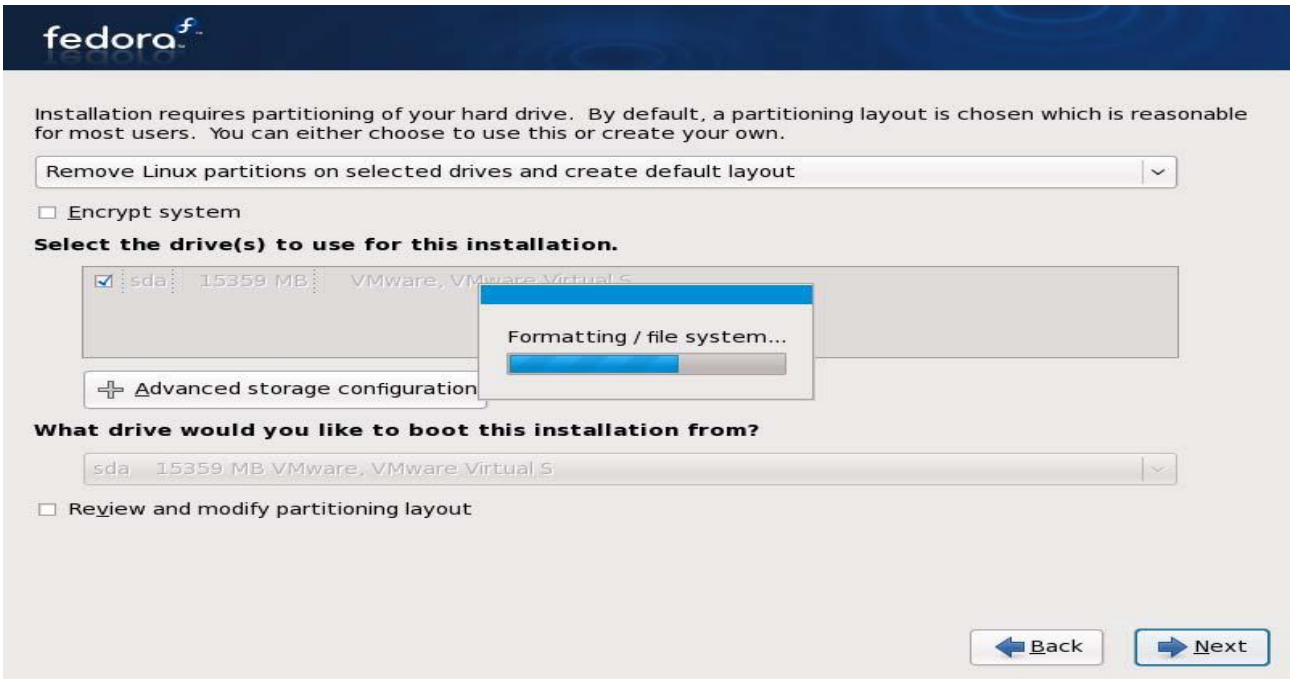
Step 9：disk partition. We followed the default option. Before do this, please back up disk data.



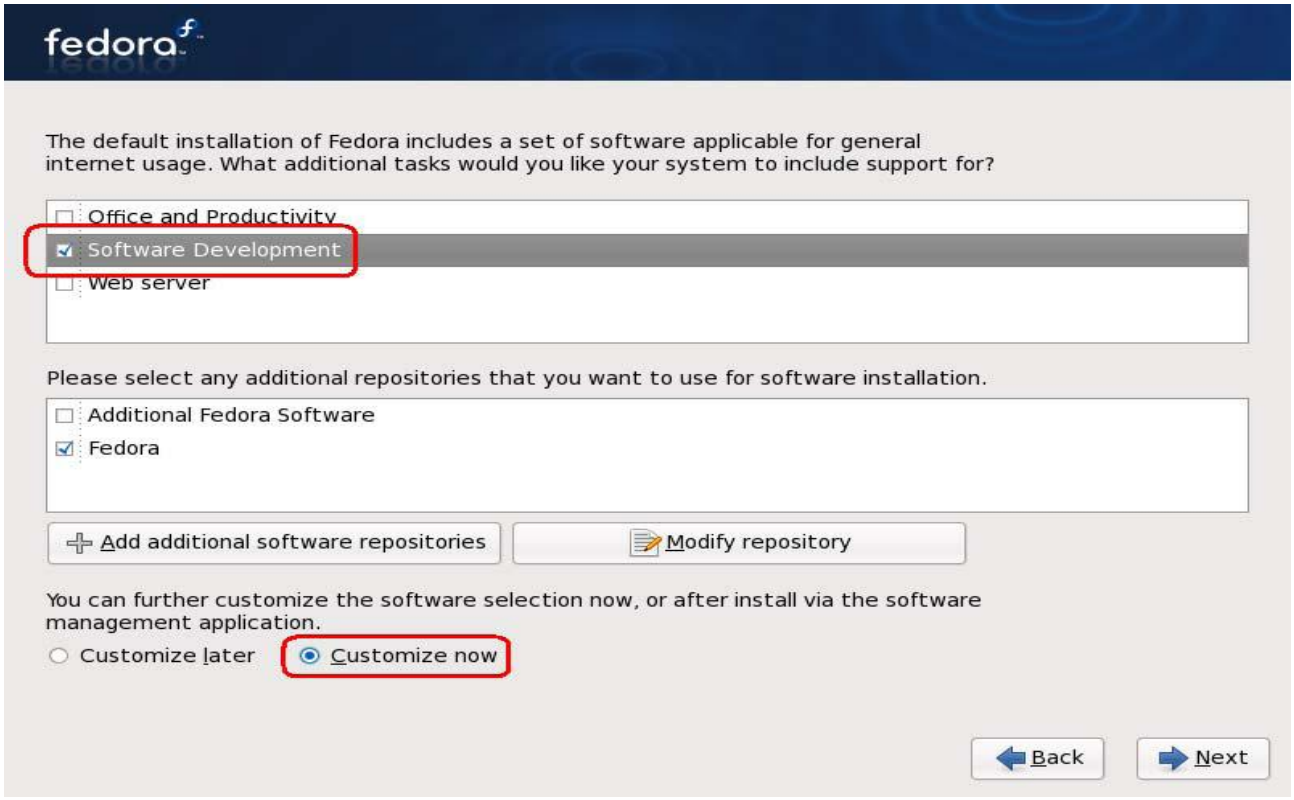
Click on “Next”, it will warn the user that all the data will be deleted. Usually we would do this installation in VMWARE, so we chose “Write changes to disk” and disk format would begin.



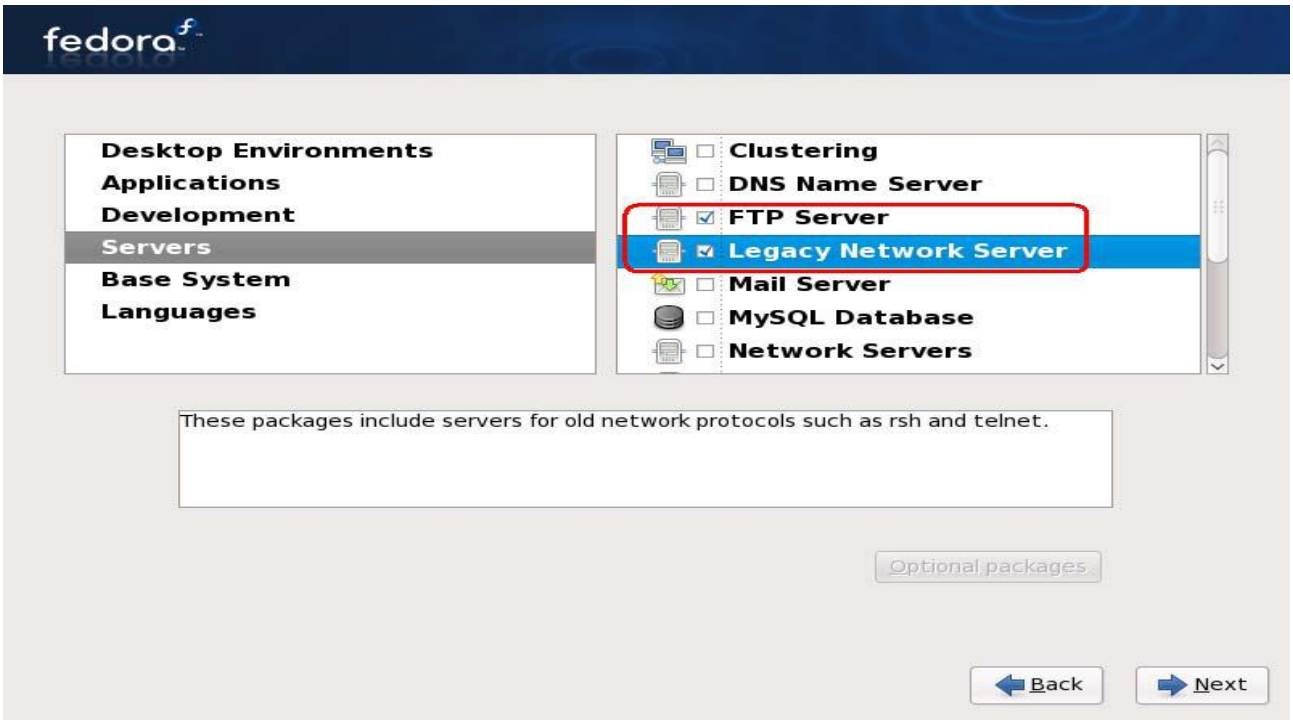
Here is the format process:



Step 11：select the installation type, in this example, we chose "customize"

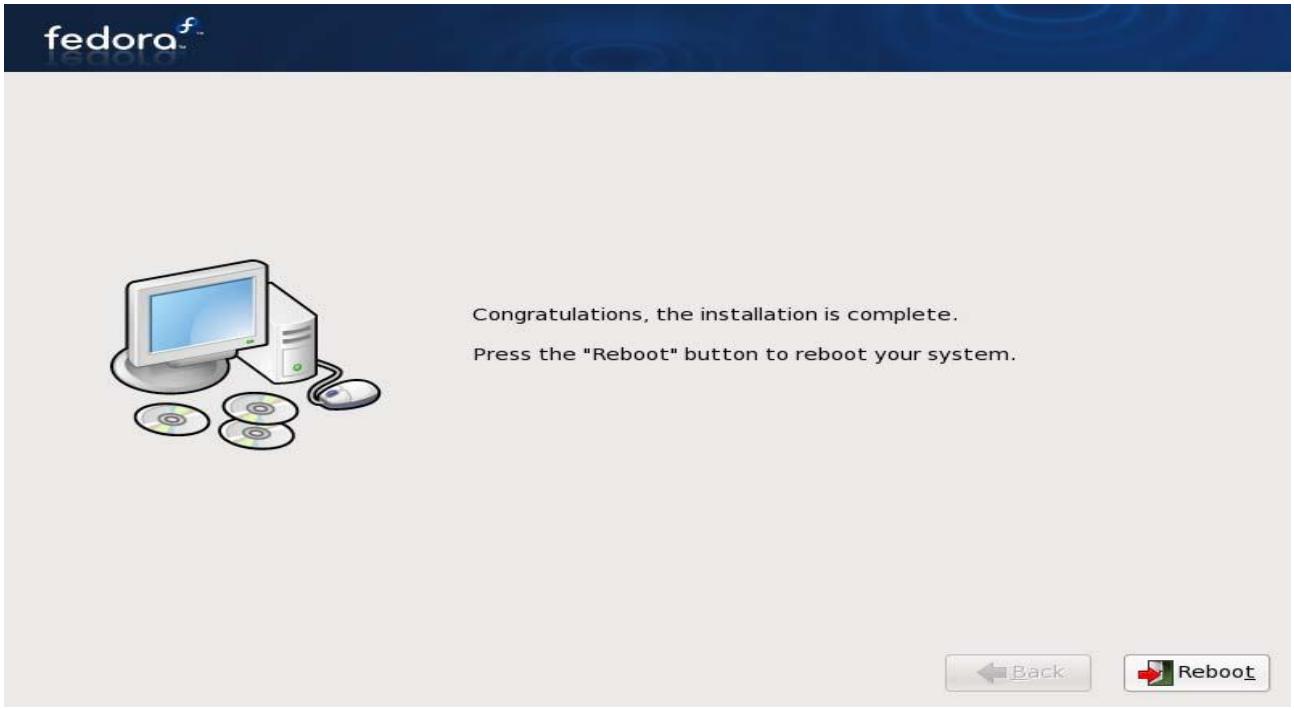


Step 12：configure the "server" item as follows:



Step 13：begin installation

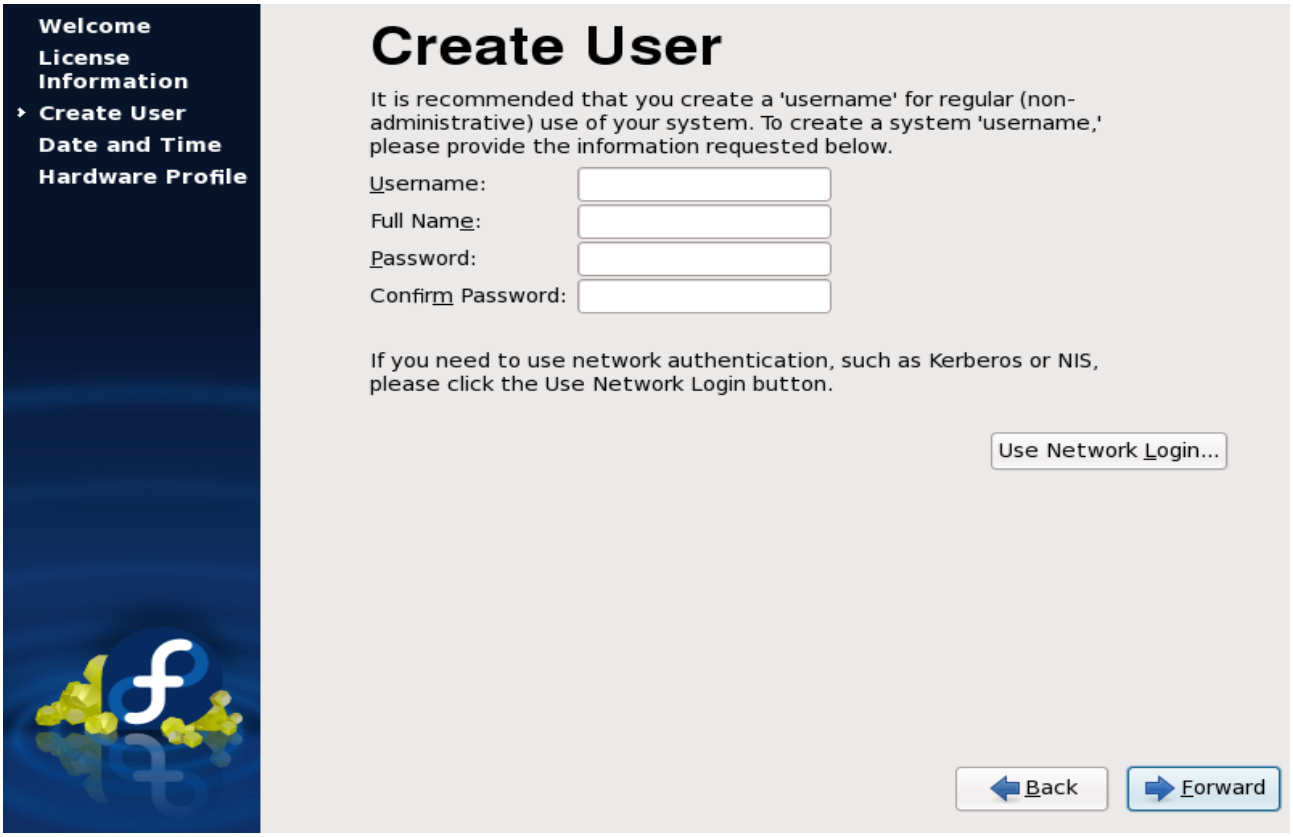
Step14: installation complete.



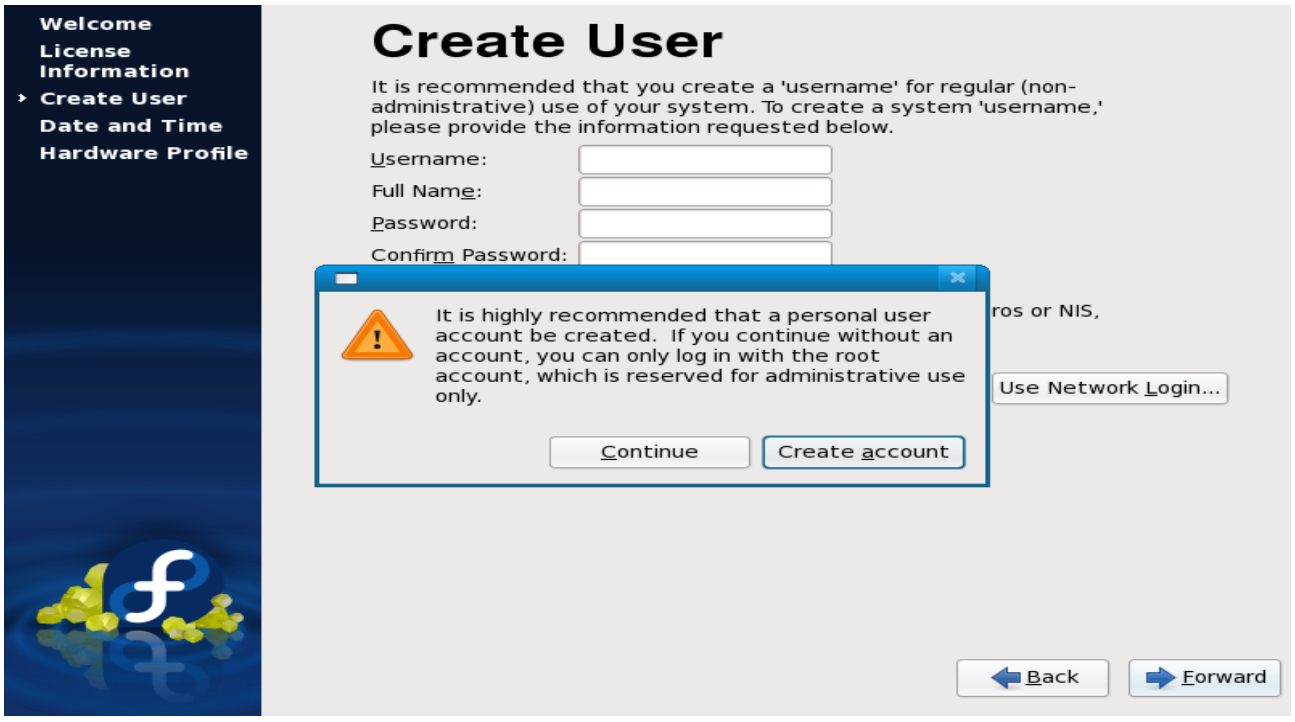
Step15: after installation completed, click on the reboot button on the page shown in step 14

Step16: skip this license page and go “forward”

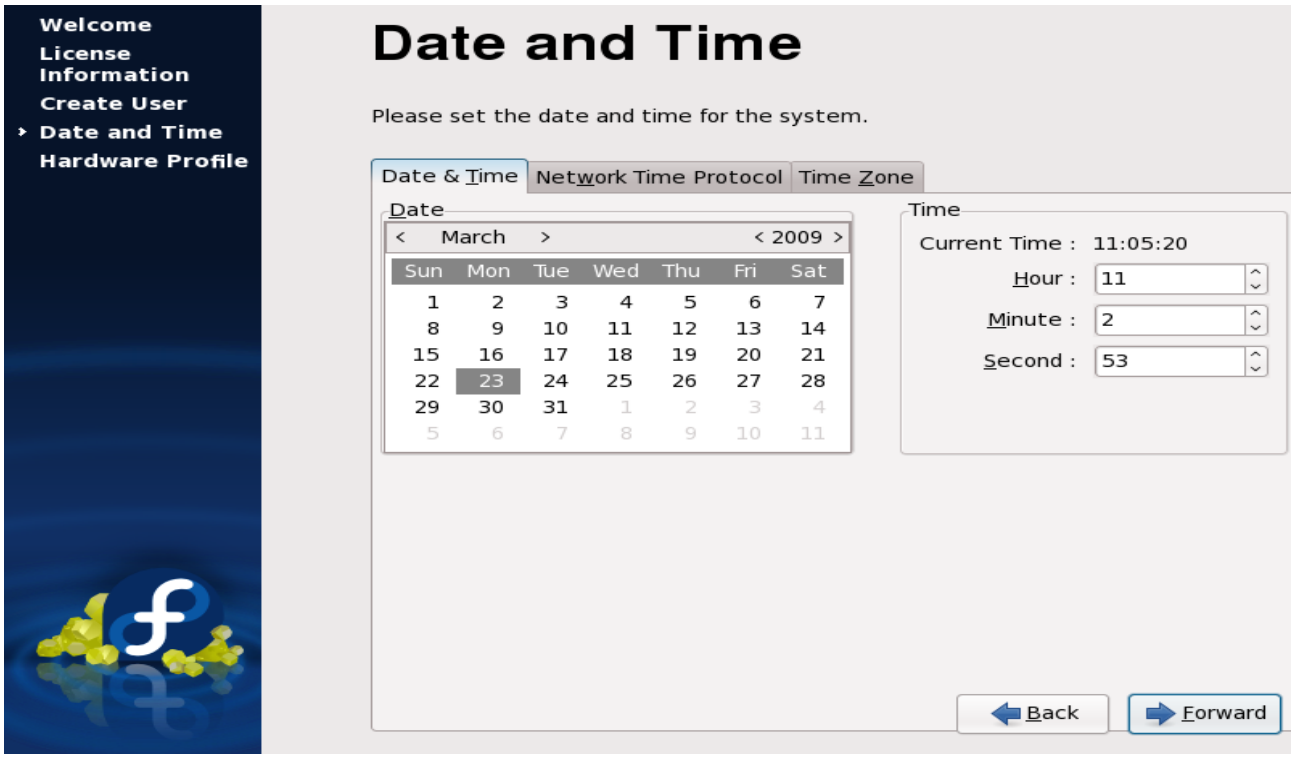
Step17: create new users. We ignored user creation and went to the next step.



Press “continue” to go on.

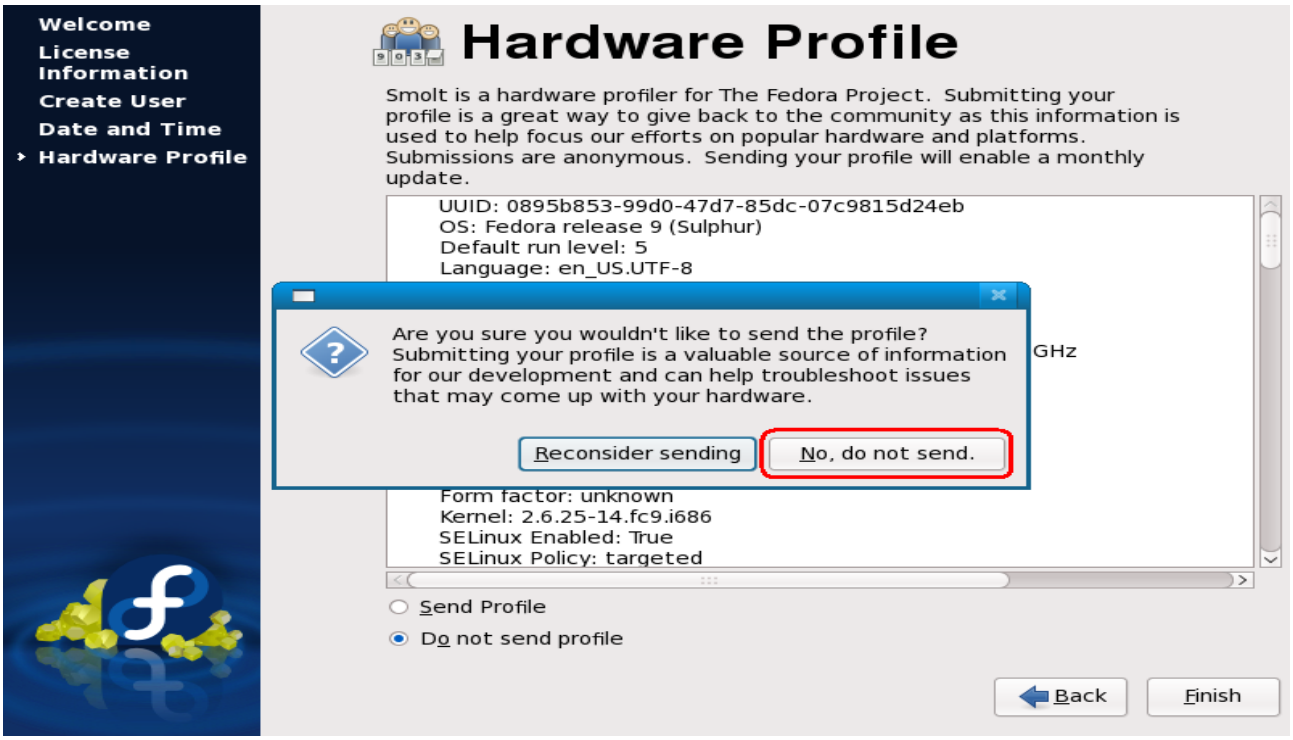


Step18: setup date and time. We ignored this and went to the next step.



Step19: confirm hardware information. We just clicked on “Finish”.

On the popup window shown below, just click on the red marked button.

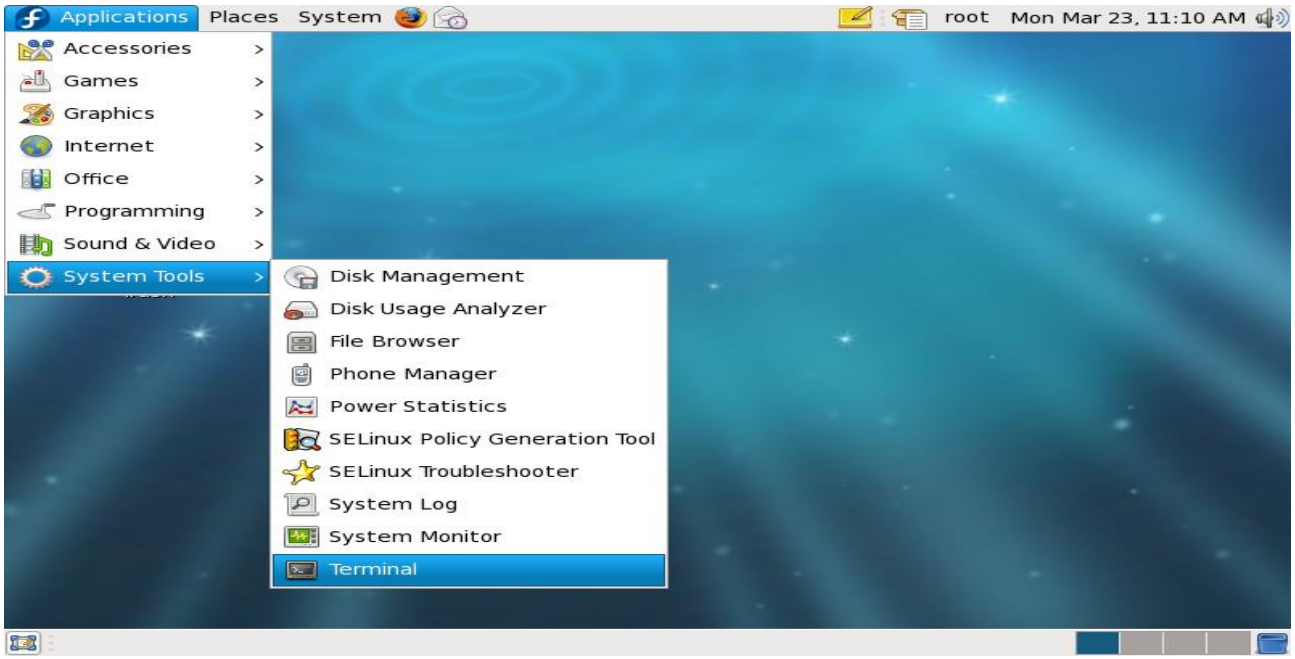


Step 20: on the login page, login as “root”

Input the password we just created for “root”

When login as “root”, the following popup window will show up, just click on “Continue”

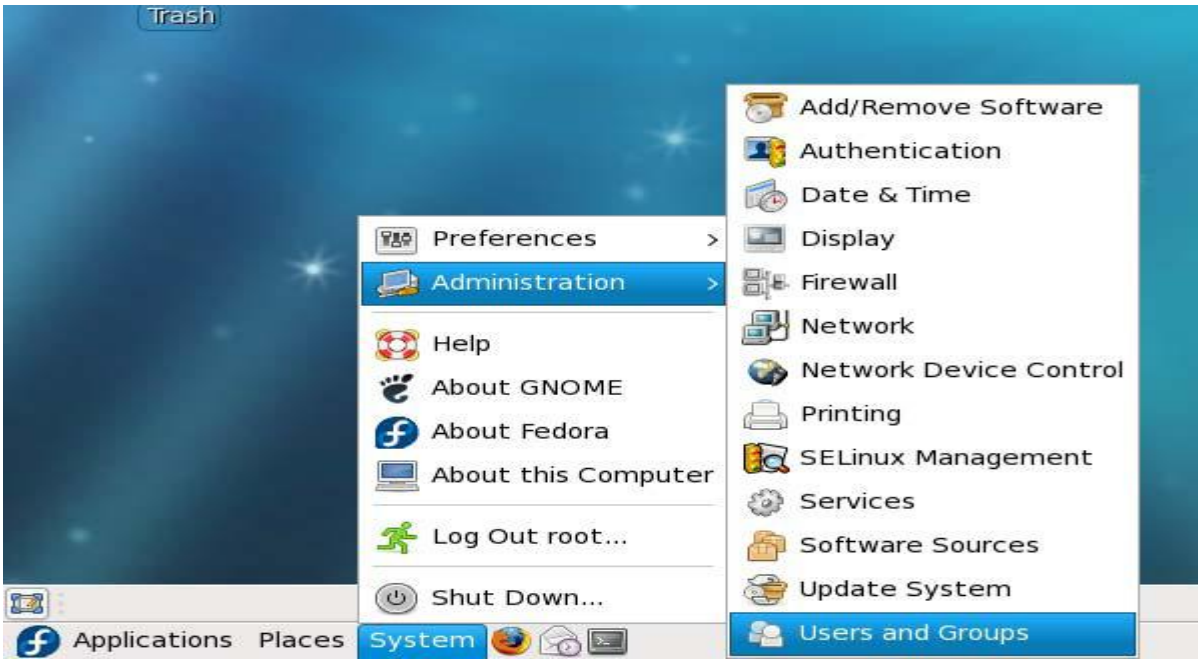
Below is the interface the user will see after a successful login.



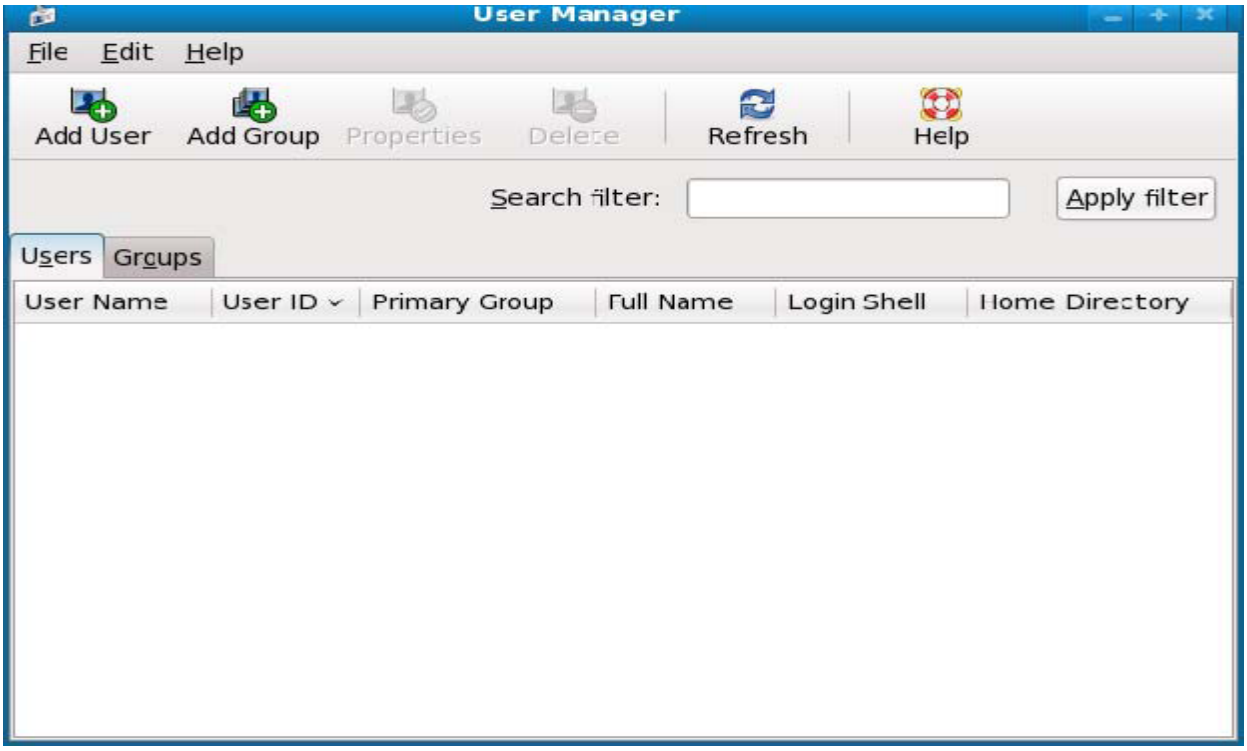
4.3.2 Add User Account

To create a new user (not root) account, here are the steps:

Step 1: go to “Users and Groups”

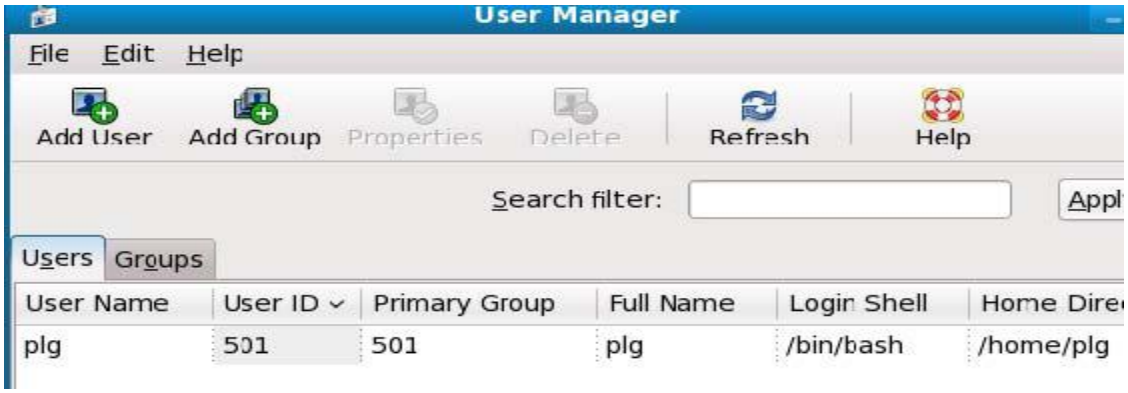


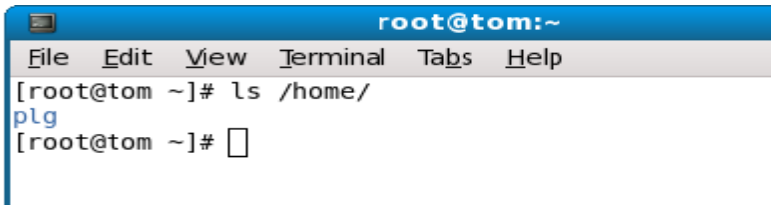
Step 2: open the “Users Manager” window



Step 3: click on the “Add User” button, type the user name and password

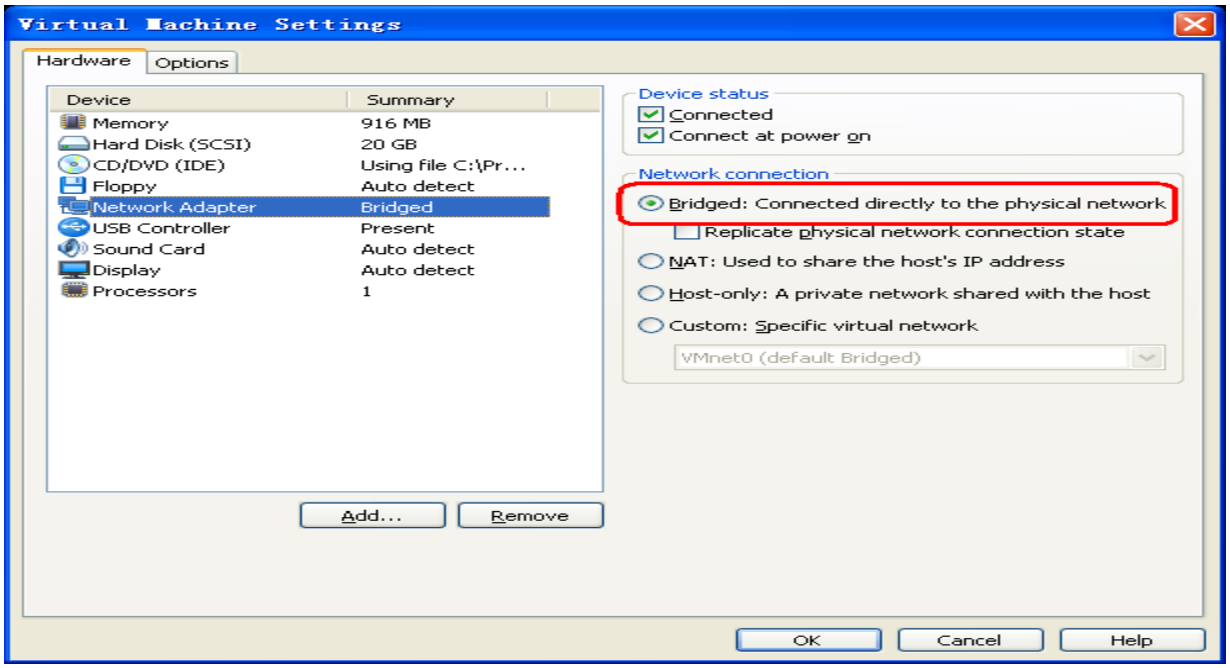
Click on “OK”, you will see that a new “plg” user has been created, and a “plg” directory has been created in the “/home” directory too.





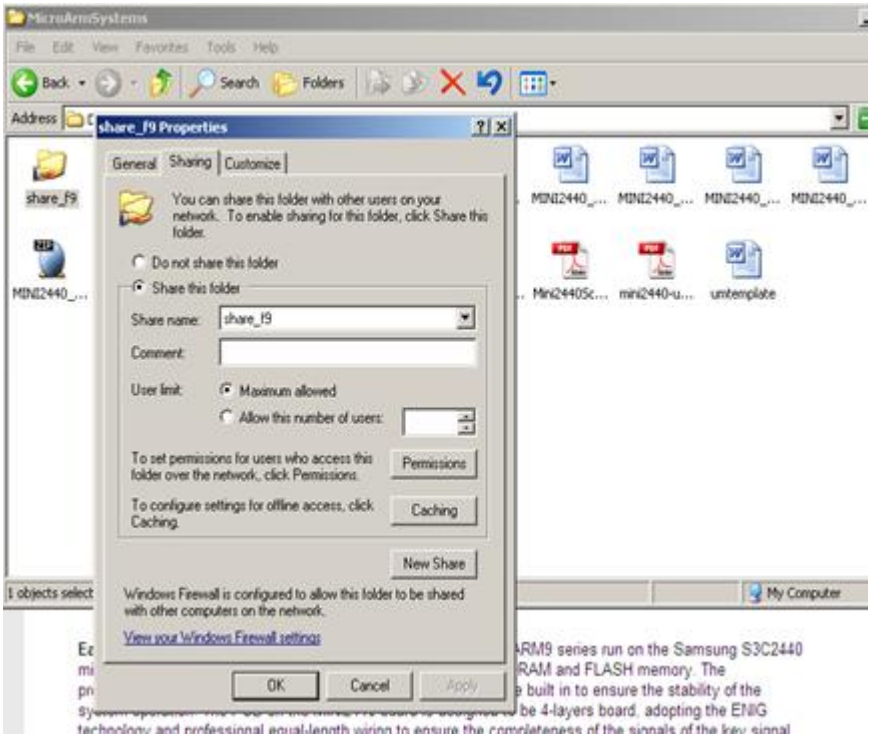
4.3.3 Access Windows Files

You can easily access shared files in Windows from either a virtual machine or a real Fedora9 system as long as they can communicate. To connect to a Windows from a virtual machine, the easiest way is to set “Guest” to “Bridge” in the network configuration.

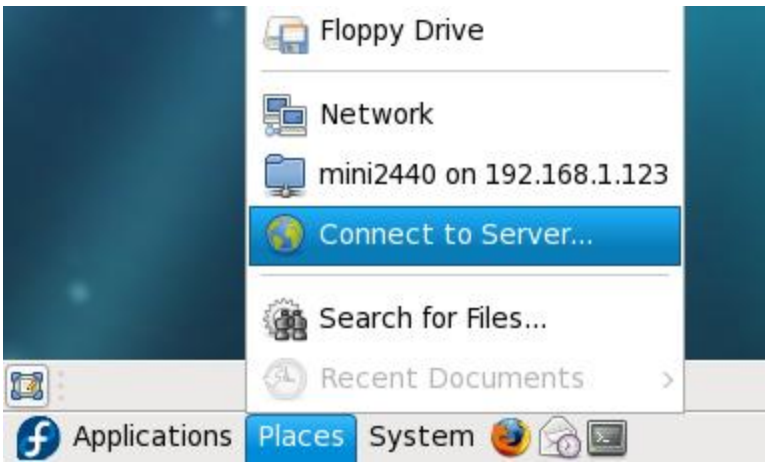


To access shared files in Windows, please following the steps below:

Step 1: set a shared directory in Windows. Here we set a “share_f9”



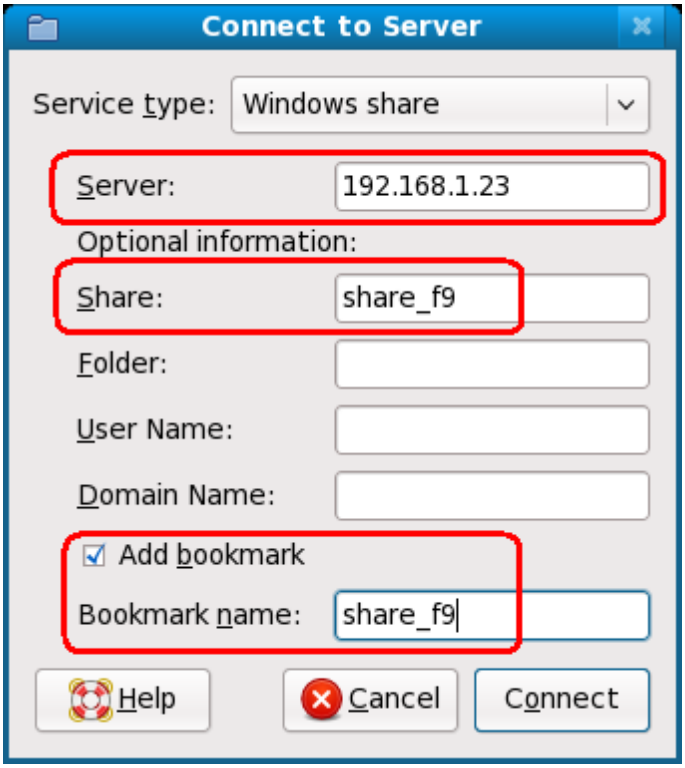
Step 2: set Fedora9



Open the window below:

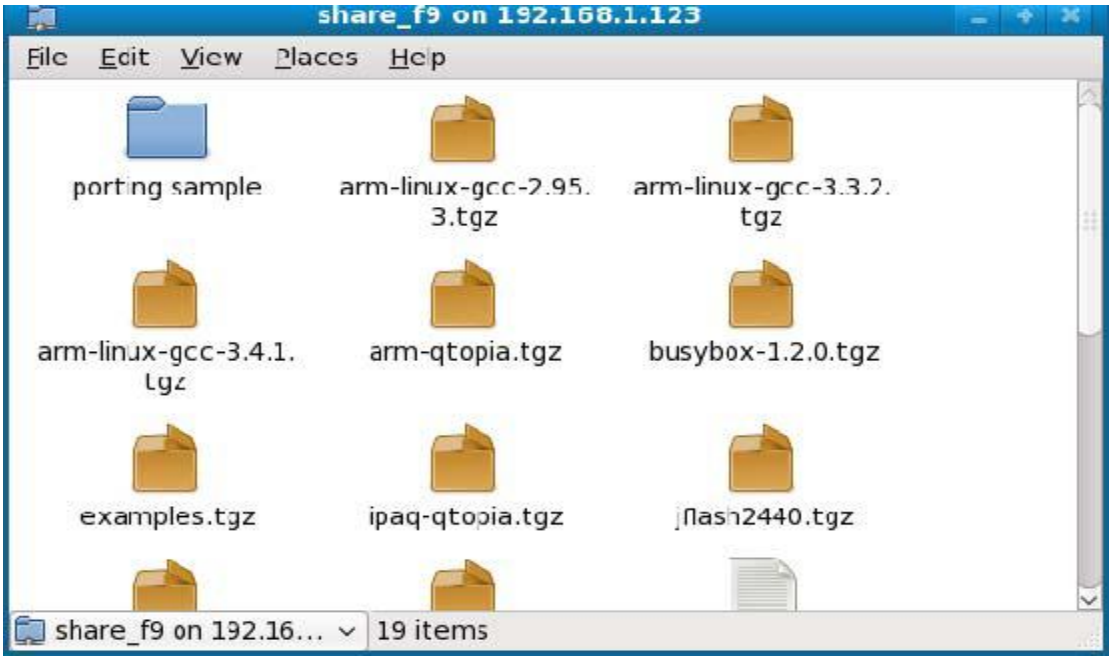
Select “Windows share” in the “service type” field

Input the shared file’s name and its windows machine IP

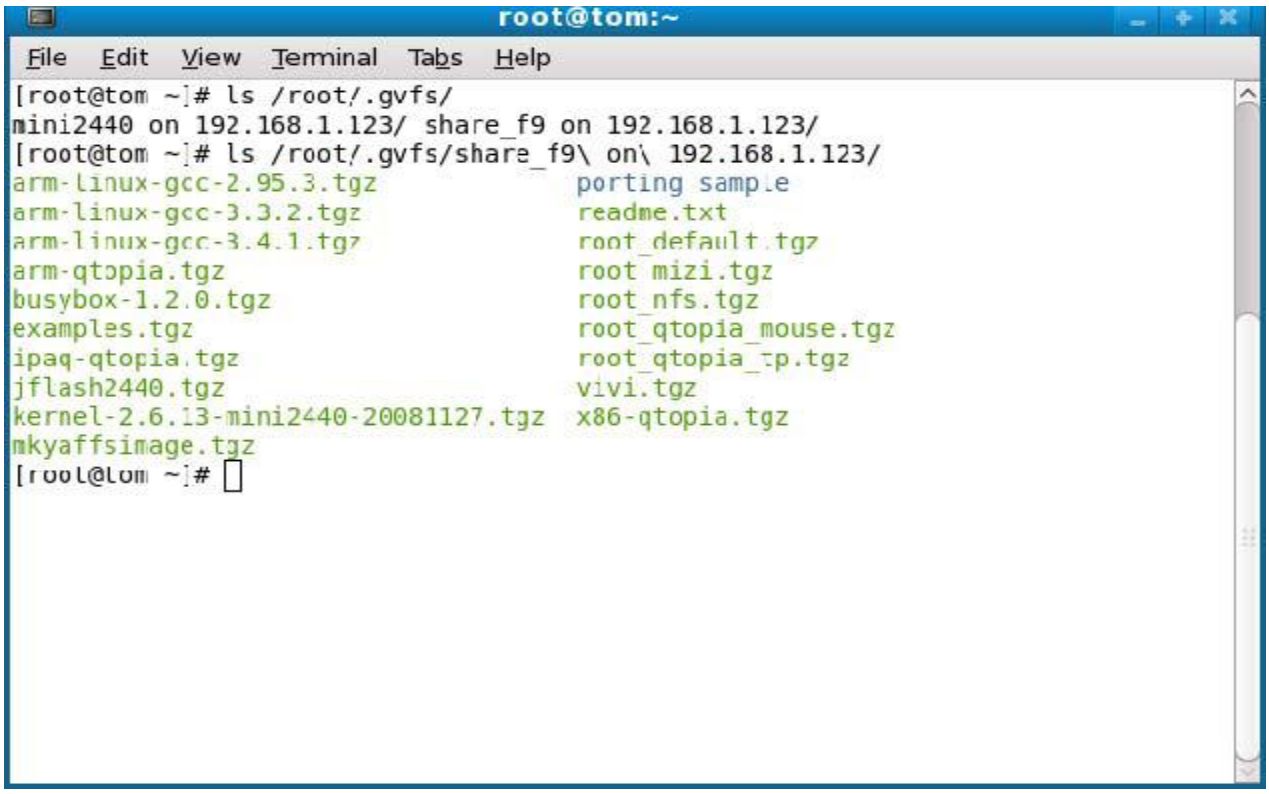


Click on “connect”, the following window will show up:

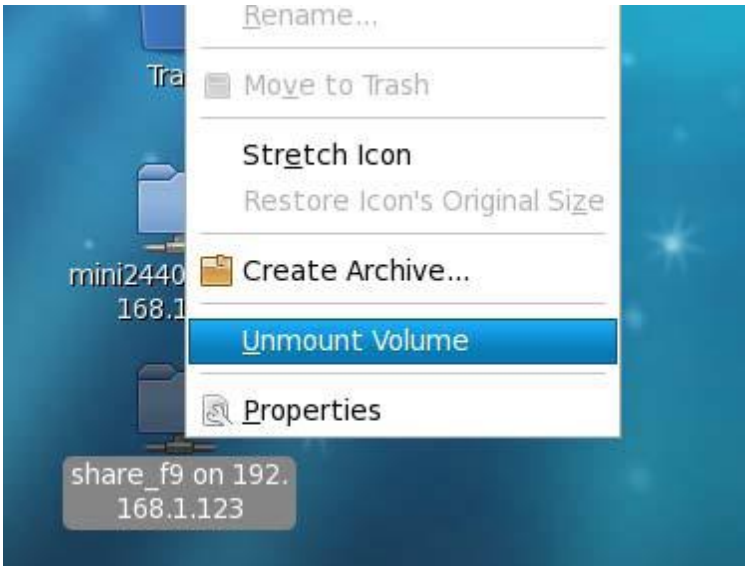
Go ahead and “connect” again, you will see the shared files you just set in your windows system.



If you want to access this directory from the command line utility, you can do it by hitting the TAB key.



To disconnect the shared directory, right click on the shared directory and following the operations in the screenshot below:



4.3.4 Set Up Cross Compile Environment

To compile kernels, Qtopia/Qt4, bootloader and other programs in Linux you need a cross compile environment. We used arm-linux-gcc-4.5.1 and its by defauly supports armv6 command sets. The following steps will introduce how to build a compile environment.

Step 1: copy the compressed file "arm-linux-gcc-4.5.1-v6-vfp-20101103.tgz" in the shipped CD into a system's directory, e.g "tmp\", enter this directory and execute the following commands:

#cd \tmp

#tar xvzf arm-linux-gcc-4.5.1-v6-vfp-20101103.tgz –C /

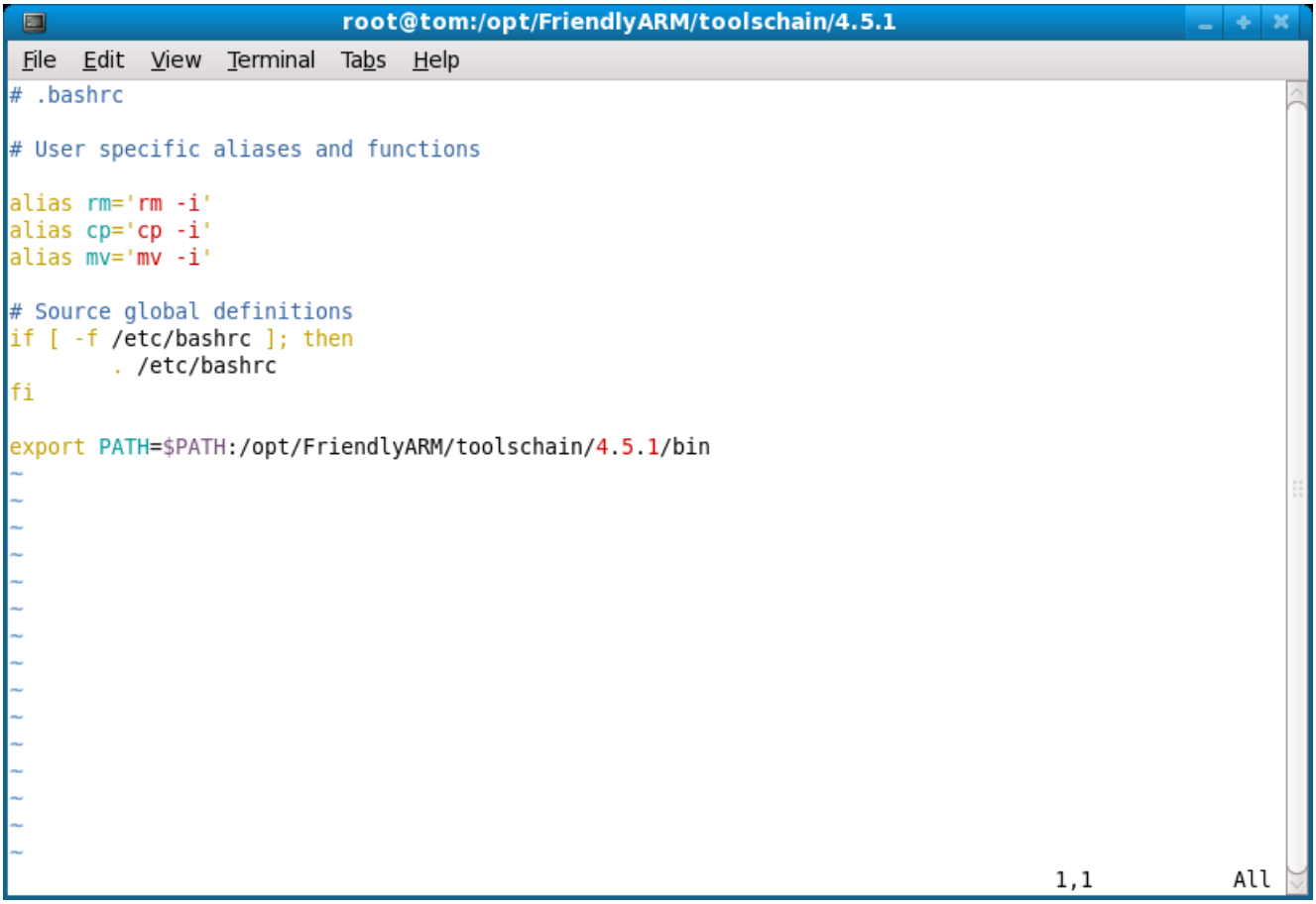
Note: there is a space after "C" and "C" is a capital letter.

These commands will install "arm-linux-gcc" in the "**/opt/FriendlyARM/toolschain/4.5.1**"

Step 2: run the command below to add the compiler's path to system variables:

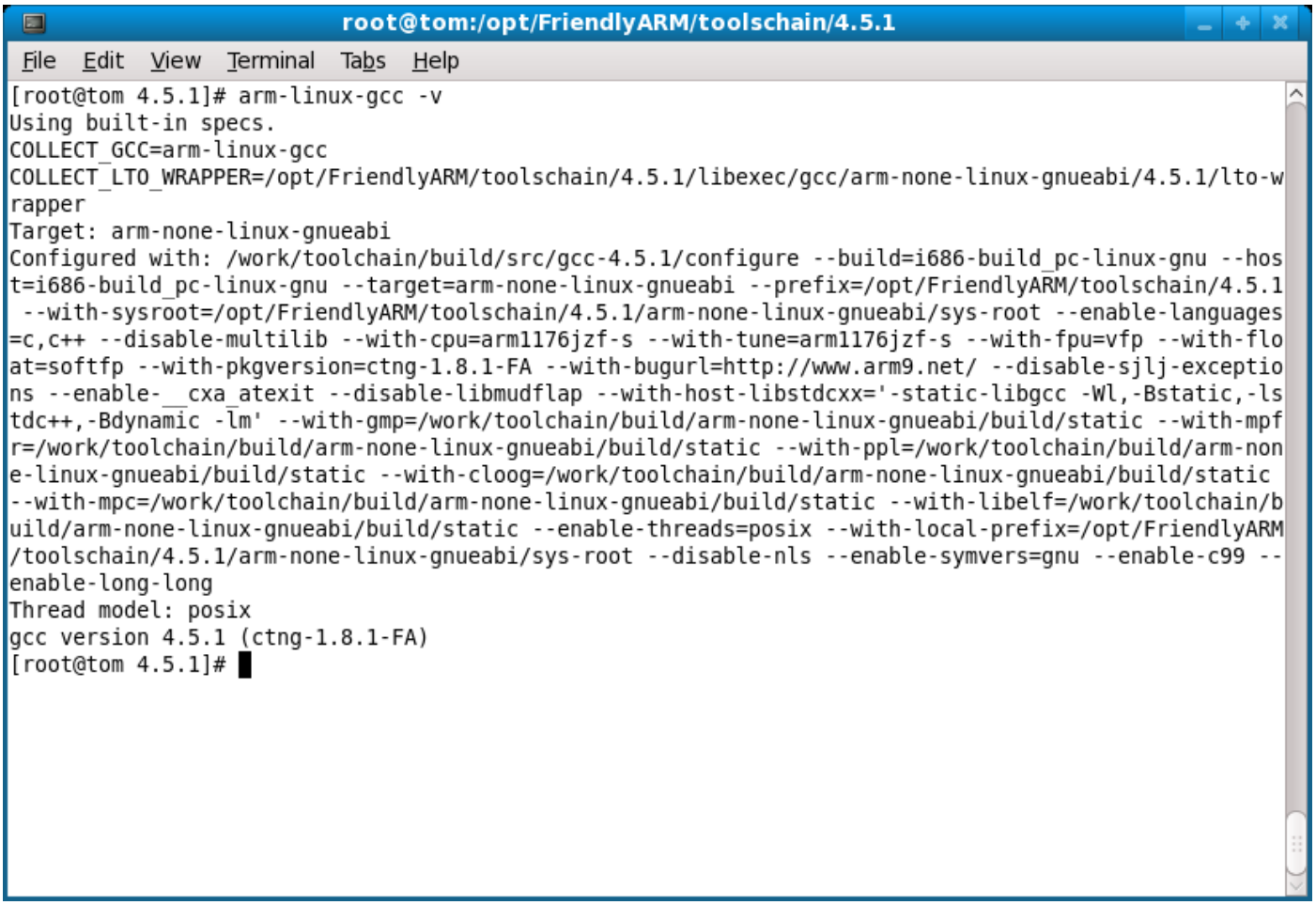
#gedit /root/.bashrc

This is to edit the "/root/.bashrc" file. Update the last line with "**export PATH=$PATH:/opt/FriendlyARM/toolschain/4.5.1/bin**" in the opened file, save and exit the file.



Logout and login the system again (no need to reboot the system, just go to "start"-> "logout"), the above settings will take into effect. Type "arm-linux-gcc –v", if the

messages depicted in the screen shot below appear, it indicates the compile environment has been set up successfully.



4.4 Uncompress Source Code and Install Application Utilities

This section will introduce how to uncompress all the source code that users may need and install some application utilities including:

- Linux kernel source code
- Qtopia-2.2.0 source code (for x86 and arm)
- arm-qt-extended-4.4.3 source code (i.e. Qtopia4, for x86 and arm)
- QtE-4.7.0 (for ARM)

- Busybox-1.17 source code
- Sample programs code (developed by FriendlyArm)
- Target file system directory
- File system image maker (for YAFFS2)
- Linux logo maker: logo_maker

Note: all source code and utilities should be uncompressed and compiled with arm-linux-gcc-4.4.1

4.4.1 Uncompress Source Code

Firstly, create a working directory: /opt/FriendlyARM/tiny4412/linux

After execute command "mkdir –p /opt/FriendlyARM/tiny4412/linux", all the source code in the following steps will be uncompressed in this work directory

(1) Get Linux source code ready

In Fedora9, create a temporary director "/tmp/linux" by running the following command

#mkdir /tmp/linux

Copy all the files in the linux directory in the shipped CD to "/tmp/linux"

(2) Uncompress the Linux kernel source code

In the work directory /opt/FriendlyARM/tiny4412/linux, run the commands below:

#cd /opt/FriendlyARM/tiny4412/linux

#tar xvzf /tmp/linux/linux-3.5-20131010.tar.gz

A linux-3.5 directory is created and it will include a complete copy of linux kernel source code.

Note: 20131010 is the date when FriendlyARM released the new version, the file name in the shipped CD may be different.

(3) Uncompress and Install the target file system

In the work directory /opt/FriendlyARM/tiny4412/linux, run the commands below:

#cd /opt/FriendlyARM/tiny4412/linux

#tar xvzf /tmp/linux/rootfs_qtopia_qt4-20131010.tgz

A rootfs_qtopia-qt4 directory is created it will include a complete copy of linux kernel source code.

Note: 20131010 is the date when FriendlyARM released the new version, the file name in the shipped CD may be different.

(4) Uncompress and install Qtopia source code

In the work directory /opt/FriendlyARM/tiny4412/linux, run the commands below:

#cd /opt/FriendlyARM/tiny4412/linux

#tar xvzf /tmp/linux/x86-qtopia-20100420.tar.gz

#tar xvzf /tmp/linux/arm-qtopia-20101105.tar.gz

An x86-qtopia directory and an arm-qtopia directory will be created, and their source code will be uncompressed into these two directories.

Note: in this release, supports for mouse and tp are all included in one package. And the source code for the embedded browser konquor is included too.

(5) Uncompress and install qt-extended-4.4.3 source code

In the work directory /opt/FriendlyARM/tiny4412/linux, run the commands below:

#cd /opt/FriendlyARM/tiny4412/linux

#tar xvzf /tmp/linux/x86-qt-extended-4.4.3-20101003.tgz

#tar xvzf /tmp/linux/arm-qt-extended-4.4.3-20101105.tgz

An x86-qt-extended-4.4.3 and an arm-qt-extended-4.4.3 are created their source code will be uncompressed into these two directories.

(6) Uncompress and install QtE-4.7.0 source code

In the work directory /opt/FriendlyARM/tiny4412/linux, run the commands below:

#cd /opt/FriendlyARM/tiny4412/linux

#tar xvzf /tmp/linux/x86-qte-4.6.1-20100516.tar.gz

#tar xvzf /tmp/linux/arm-qte-4.7.0-20101105.tar.gz

An x86-qte-4.6.1 and an arm-qte-4.7.0 are created their source code will be uncompressed into these two directories.

(7) Uncompress and install busybox source code

The Busybox is a compact Linux tool kit. Here we used busybox-1.17.2. Users can download its latest version from http://www.busybox.net

In the work directory /opt/FriendlyARM/tiny4412/linux, run the commands below:

#cd /opt/FriendlyARM/tiny4412/linux

#tar xvzf /tmp/linux/busybox-1.17.2-20101120.tgz

A busybox-1.17.2 directory is created its source code will be extracted into this

directory.

Note: for the sake of users, we have made a default configuration file: fa.config.

(8) Uncompress and install Linux sample programs

In the work directory /opt/FriendlyARM/tiny4412/linux, run the commands below:

#cd /opt/FriendlyARM/tiny4412/linux

#tar xvzf /tmp/linux/examples-tiny4412-20131010.tgz

An examples directory is all the source code will be extracted into this directory.

Note: all these sample programs are developed by FriendlyARM.

4.4.2 Create Target File System

We offered the following two packages:

- rootfs_qtopia_qt4-20131010.tgz

Execute the following commands:

#cd /opt/FriendlyARM/tiny4412/linux

#tar xvzf /tmp/linux/ rootfs_qtopia_qt4-20131010.tgz

A rootfs_qtopia_qt4 will be created.

This package includes qtopia-2.2.0, Qtopia4 and QtE-4.7.0, busybox and some command line utilities. It has the following excellent features:

- auto detection of touch screen and launching the calibration utility if necessary. If no touch screen is connected system will enable the mouse.

- auto detection of command or high speed SD cards (up to maximum memory of 32G)

and flash drives

- auto detection of USB mouse or touch screen

- support co-existence of a USB mouse and a touch screen (since Linux-2.6.36)

4.4.3 Install Logo Maker

LogoMaker is developed by FriendlyARM for making linux logos. There are many resources describing how to convert image files such as bmp, jpg, png and so on to linux logos using command line tools. We created this graphic version which is based on Fedora9.Execute the command below:

#tar xvzf /tmp/linux/logomaker.tgz –C /

Note: “C” is capitalized and means “change”.

After executing the above commands, LogoMaker will be installed in the /usr/sbin directory. It only has one file. After installing it, type “logomake” in a command line window, you will see the following screenshot

4.5 Configure and Compile Kernel

Type the following command to compile:

#cp /opt/FriendlyARM/tiny4412/android/linux-3.5

#cp tiny4412_linux_defconfig .config : there is a space after "defconfig" and a "."

#make ; begins to compile

After the compilation is done, an image file **zImage** will be generated under "**arch/arm/boot**".

4.6 Make File System Image

Please make sure you have installed "mktools" tools and have an image directory ready before continue.

Enter "/opt/FriendlyARM/tiny4412/linux" and execute the following command:

#make_ext4fs –s –l 314572800 –a root –L linux rootfs_qtopia_qt4.img rootfs_qtopia_qt4

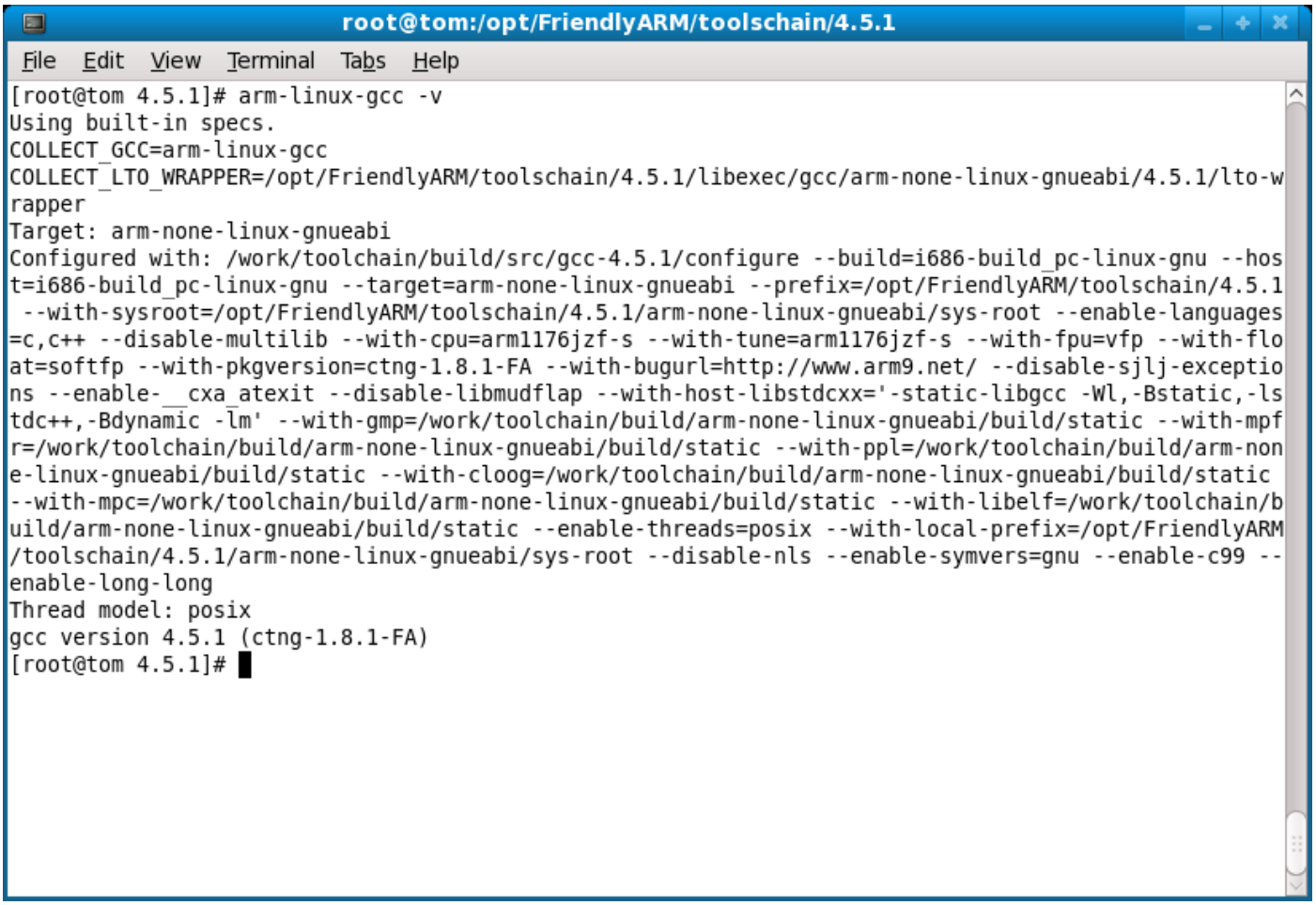
This will compress the whole "rootfs_qtopia_qt4" into a yaffs2 rootfs_qtopia_qt4.img file. It is the same as the one in "/images/Linux/" in the shipped CD. Download it to your board's NAND Flash.

4.7 Sample Linux Programs

This section lists some sample Linux programs for users' reference.

You can find those programs under "**/opt/FriendlyARM/tiny4412/examples**".All

the following programs are compiled with arm-linux-gcc-4.5.1-v6-vfp. We don't guarantee they can be compiled and run with other corss compilers.



4.7.1 "Hello, World!"

The source code of "Hello,World" is under "**/opt/FriendlyARM/tiny4412/linux/examples/hello**". Its contents are as follows:

```
#include <stdio.h>

int main(void) {

printf("hello, FriendlyARM!\n");

}
```

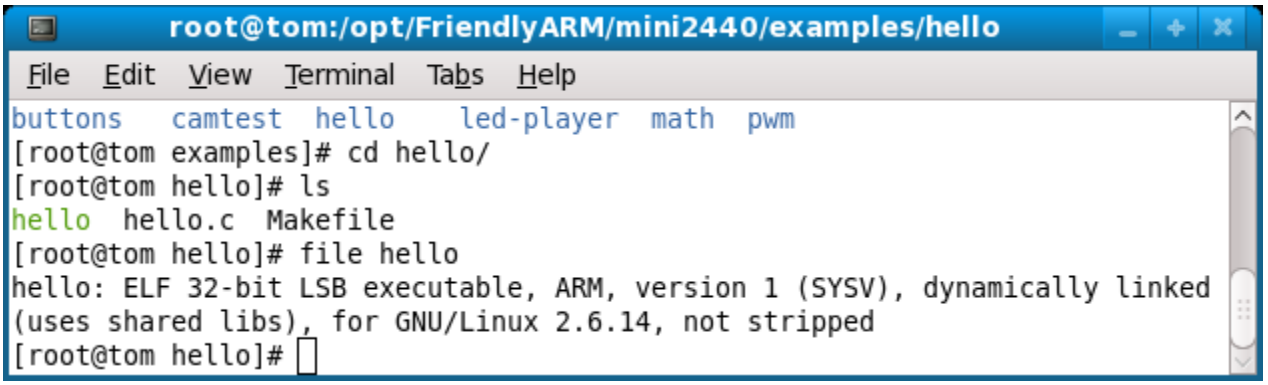
Step1: Compile Hello,World

Enter the directory where the source code is located and execute “make”:

#cd /opt/FriendlyARM/tiny4412/linux/examples/hello

#make

A “hello” executable will be generated and you can check whether it is for ARM by commanding “file”:



Step2: Download “Hello,World” to Board

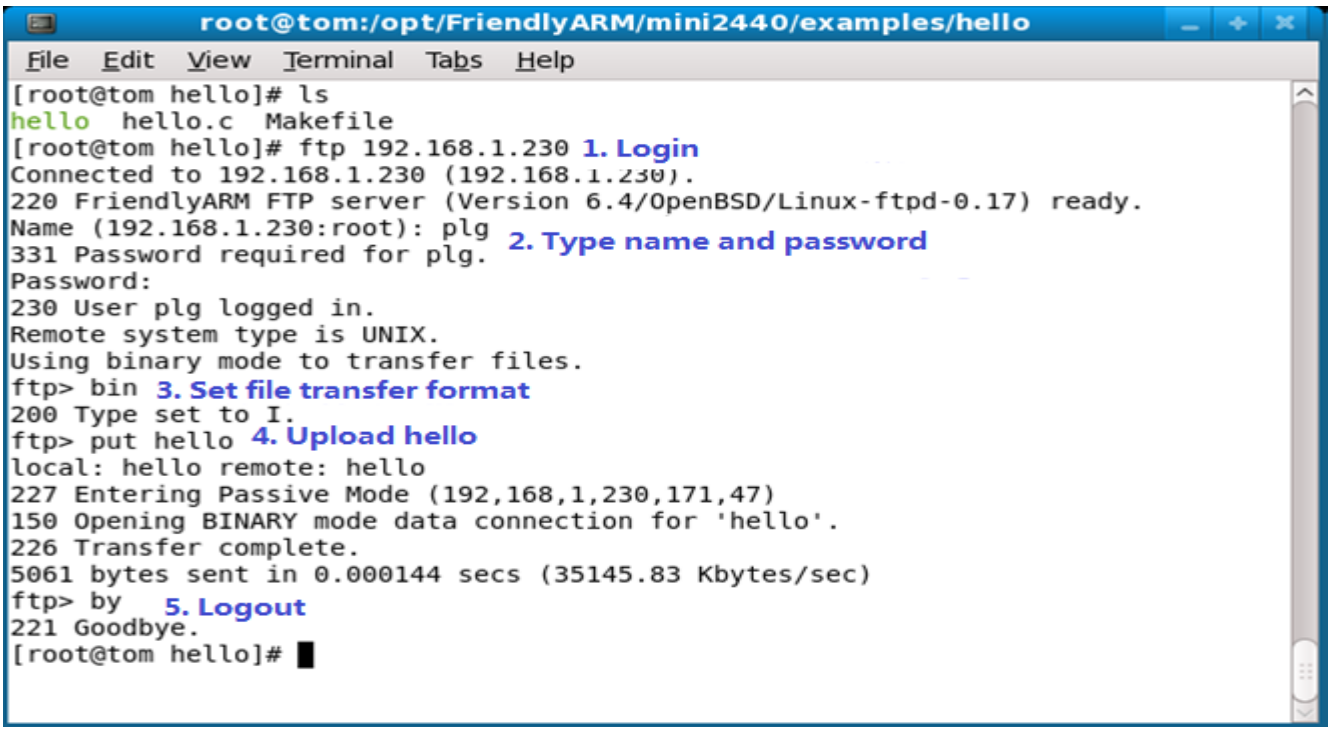
You can download your executable to the board in any of the following ways:

- FTP file transfer (recommended)
- Copy to a media (such as flash drives)
- File transfer via serial port

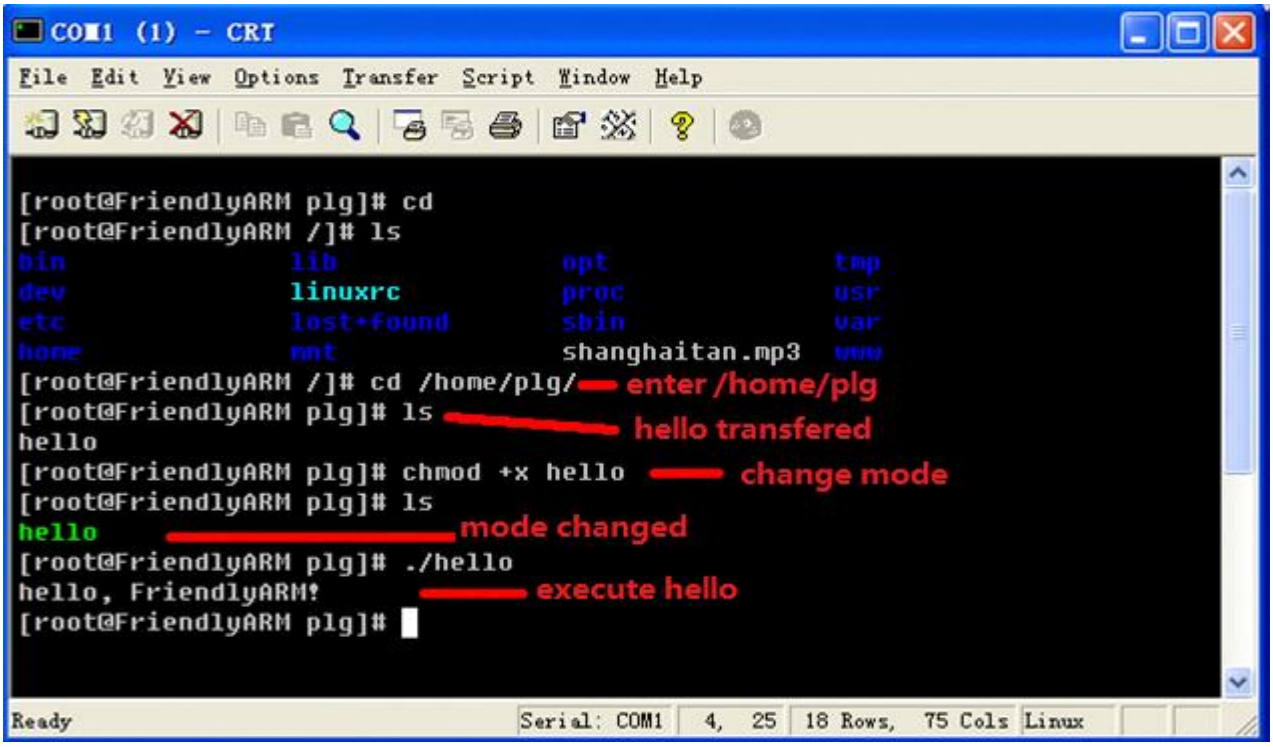
(1) FTP File Transfer

Note: login your board via FTP, transfer your executable to it and change its file property to executable.

First, execute your commands in PC



Go to your board and execute the following commands:



(2) Copy to Flash Drive

Note: copy your executable to a flash drive, mount it to your board and copy the file to "/bin"

1. Copy to Flash Drive

Connect your flash drive to your PC and execute the following commands

#mount /dev/sda1 /mnt ; mount your drive

#cp hello /mnt ; copy your file to the drive

#umount /mnt ; unmount your drive

2. Copy to Board

Insert your drive to your board's USB host, it will be automatically mounted under "/udisk". Please execute the following command

#cd /udisk

#./hello ； execute "hello"

Note: if you take out your drive directly you need to go back to the root directory and execute "umount /udisk" for the next mount

```
usb 1-1: Product: DataTraveler 2.0
usb 1-1: Manufacturer: Kingston
usb 1-1: SerialNumber: 001AA0A0BF1AC8C1155A0318
usb 1-1: configuration #1 chosen from 1 choice
scsi1 : SCSI emulation for USB Mass Storage devices
scsi 1:0:0:0: Direct-Access     Kingston DataTraveler 2.0 1.00 PQ: 0 ANSI: 2
sd 1:0:0:0: [sda] 7823296 512-byte hardware sectors: (4.00 GB/3.72 GiB)
sd 1:0:0:0: [sda] Write Protect is off
sd 1:0:0:0: [sda] Assuming drive cache: write through
sd 1:0:0:0: [sda] 7823296 512-byte hardware sectors: (4.00 GB/3.72 GiB)
sd 1:0:0:0: [sda] Write Protect is off
sd 1:0:0:0: [sda] Assuming drive cache: write through
 sda: sda1
sd 1:0:0:0: [sda] Attached SCSI removable disk
FAT: utf8 is not a recommended IO charset for FAT filesystems, filesystem will b
e case sensitive!

[root@FriendlyARM /]# cd /udisk/
[root@FriendlyARM /udisk]# ls
hello   images  linux   mp3     photo   video
[root@FriendlyARM /udisk]# ./hello
hello, FriendlyARM!
[root@FriendlyARM /udisk]# _
```

(1) File Transfer via Serial Port

Download your file to the board via serial port and change its property to executable

#chmod +x hello

Note: some users do this via a USB to Serial connector. This may not be successful due to the connector's quality issues therefore we recommend file transfer via FTP

4.7.2 LED Test Program

The source code of "Hello,World" is under "**/opt/FriendlyARM/tiny4412/linux/examples/hello**". Its contents are as follows:

Program Description:	
Source Code Location	/opt/FriendlyARM/tiny4412/linux/linux-xxx/drivers/char
Driver	tiny4412_leds.c
Device Type	misc
Device Name	/dev/leds
Test Program Source Code Location	/opt/FriendlyARM/tiny4412/linux/examples/leds
Test Program Name	led.c
Executable Name	led
Test Program's Location in Board	
Note: the LED driver has been compiled into the kernel by default and you cannot load it via insmod	
Program:	

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/ioctl.h>
int main(int argc, char **argv)
{
int on;
int led_no;
int fd;
/* Check parameters */
if (argc != 3 || sscanf(argv[1], "%d", &led_no) != 1 || sscanf(argv[2],"%d", &on) != 1 ||
on < 0 || on > 1 || led_no < 0 || led_no > 3) {
```

```
fprintf(stderr, "Usage: leds led_no 0|1\n");
exit(1);
}
/*Open /dev/leds file*/
fd = open("/dev/leds0", 0);
if (fd < 0) {
fd = open("/dev/leds", 0);
}
if (fd < 0) {
perror("open device leds");
exit(1);
}
/*Manipulate led via ioctl and input parameters */
ioctl(fd, on, led_no);
/*Close device*/
close(fd);
return 0;
}
```

You can compile the program, download it and run

4.7.3 User Button Test Program

Program Description:	
Source Code Location	/opt/FriendlyARM/tiny4412/linux/linux-xxx/drivers/char
Driver	tiny4412_buttons.c
Device Type	misc
Device Name	/dev/buttons
Test Program Source Code Location	/opt/FriendlyARM/tiny4412/linux/examples/buttons
Test Program Name	Button_test.c
Executable Name	buttons
Test Program's Location in Board	
Note: the button driver has been compiled into the kernel by default and you cannot load it via insmod	
Program:	

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/ioctl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
```

```
#include <sys/select.h>
#include <sys/time.h>
#include <errno.h>
int main(void)
{
int buttons_fd;
char buttons[6] = {'0', '0', '0', '0', '0', '0'};
buttons_fd = open("/dev/buttons", 0);
if (buttons_fd < 0) {
perror("open device buttons");
exit(1);
}
for (;;) {
char current_buttons[6];
int count_of_changed_key;
int i;
if (read(buttons_fd, current_buttons, sizeof current_buttons) != sizeof current_buttons) {
perror("read buttons:");
exit(1);
}
for (i = 0, count_of_changed_key = 0; i < sizeof buttons / sizeof buttons[0]; i++) {
if (buttons[i] != current_buttons[i]) {
buttons[i] = current_buttons[i];
printf("%skey %d is %s", count_of_changed_key? ", ": "", i+1, buttons[i] == '0' ? "up" : "down");
count_of_changed_key++;
}
}
if (count_of_changed_key) {
printf("\n");
}
}
close(buttons_fd);
return 0;
}
```

You can compile the program, download it and run

4.7.4 PWM Buzzer Program

Program Description:

Source Code Location	/opt/FriendlyARM/tiny4412/linux/linux-xxx/drivers/char
Driver	tiny4412_pwm.c
Device Type	misc
Device Name	/dev/pwm
Test Program Source Code Location	/opt/FriendlyARM/tiny4412/linux/examples/pwm
Test Program Name	pwm_test.c
Executable Name	Pwm_test
Test Program's Location in Board	
Note: the pwm driver has been compiled into the kernel by default and you cannot load it via insmod	

Program:

```
#include <stdio.h>
#include <termios.h>
#include <unistd.h>
#include <stdlib.h>
#define PWM_IOCTL_SET_FREQ 1
#define PWM_IOCTL_STOP 2
#define ESC_KEY 0x1b
static int getch(void)
{
struct termios oldt,newt;
int ch;
if (!isatty(STDIN_FILENO)) {
fprintf(stderr, "this problem should be run at a terminal\n");
exit(1);
}
// save terminal setting
if(tcgetattr(STDIN_FILENO, &oldt) < 0) {
perror("save the terminal setting");
exit(1);
}
// set terminal as need
newt = oldt;
newt.c_lflag &= ~( ICANON | ECHO );
if(tcsetattr(STDIN_FILENO,TCSANOW, &newt) < 0) {
perror("set terminal");
exit(1);
}
ch = getchar();
// restore termial setting
if(tcsetattr(STDIN_FILENO,TCSANOW,&oldt) < 0) {
perror("restore the termial setting");
```

```
exit(1);
}
return ch;
}
static int fd = -1;
static void close_buzzer(void);
static void open_buzzer(void)
{
fd = open("/dev/pwm", 0);
if (fd < 0) {
perror("open pwm_buzzer device");
exit(1);
}
// any function exit call will stop the buzzer
atexit(close_buzzer);
}
static void close_buzzer(void)
{
if (fd >= 0) {
ioctl(fd, PWM_IOCTL_STOP);
close(fd);
fd = -1;
}
}
static void set_buzzer_freq(int freq)
{
// this IOCTL command is the key to set frequency
int ret = ioctl(fd, PWM_IOCTL_SET_FREQ, freq);
if(ret < 0) {
perror("set the frequency of the buzzer");
exit(1);
}
}
static void stop_buzzer(void)
{
int ret = ioctl(fd, PWM_IOCTL_STOP);
if(ret < 0) {
perror("stop the buzzer");
exit(1);
}
}
```

```
int main(int argc, char **argv)
{
int freq = 1000 ;
open_buzzer();
printf( "\nBUZZER TEST ( PWM Control )\n" );
printf( "Press +/- to increase/reduce the frequency of the BUZZER\n" ) ;
printf( "Press 'ESC' key to Exit this program\n\n" );
while( 1 )
{
int key;
set_buzzer_freq(freq);
printf( "\tFreq = %d\n", freq );
key = getch();
switch(key) {
case '+':
if( freq < 20000 )
freq += 10;
break;
case '-':
if( freq > 11 )
freq -= 10 ;
break;
case ESC_KEY:
case EOF:
stop_buzzer();
exit(0);
default:
break;
}
}
}
```

You can compile the program, download it and run

4.7.5 I2C-EEPROM Program

Program Description:	
Source Code Location	/opt/FriendlyARM/tiny4412/linux/linux-xxx/drivers/i2c/busses
Driver	I2c-s3c2410c
Device Type	Char

Device Name	/dev/i2c/0
Test Program Source Code Location	/opt/FriendlyARM/tiny4412/linux/examples/i2c
Test Program Name	Eeprog.c 24cxx.c
Executable Name	I2c
Test Program's Location in Board	
Note: the i2c driver has been compiled into the kernel by default and you cannot load it via insmod	
Program:	

Note: the following program depends on "24cxx.c" in the same directory.

```
#include <stdio.h>
#include <fcntl.h>
#include <getopt.h>
#include <unistd.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>
#include <sys/types.h>
#include <sys/stat.h>
#include "24cXX.h"
#define usage_if(a) do { do_usage_if( a , __LINE__); } while(0);
void do_usage_if(int b, int line)
{
const static char *eeprog_usage =
"I2C-24C08(256 bytes) Read/Write Program, ONLY FOR TEST!\n"
"FriendlyARM Computer Tech. 2009\n";
if(!b)
return;
fprintf(stderr, "%s\n[line %d]\n", eeprog_usage, line);
exit(1);
}
#define die_if(a, msg) do { do_die_if( a , msg, __LINE__); } while(0);
void do_die_if(int b, char* msg, int line)
{
if(!b)
return;
fprintf(stderr, "Error at line %d: %s\n", line, msg);
fprintf(stderr, " sysmsg: %s\n", strerror(errno));
exit(1);
}
static int read_from_eeprom(struct eeprom *e, int addr, int size)
```

```
{
int ch, i;
for(i = 0; i < size; ++i, ++addr)
{
die_if((ch = eeprom_read_byte(e, addr)) < 0, "read error");
if( (i % 16) == 0 )
printf("\n %.4x| ", addr);
else if( (i % 8) == 0 )
printf(" ");
printf("%.2x ", ch);
fflush(stdout);
}
fprintf(stderr, "\n\n");
return 0;
}
static int write_to_eeprom(struct eeprom *e, int addr)
{
int i;
for(i=0, addr=0; i<256; i++, addr++)
{
if( (i % 16) == 0 )
printf("\n %.4x| ", addr);
else if( (i % 8) == 0 )
printf(" ");
printf("%.2x ", i);
fflush(stdout);
die_if(eeprom_write_byte(e, addr, i), "write error");
}
fprintf(stderr, "\n\n");
return 0;
}
int main(int argc, char** argv)
{
struct eeprom e;
int op;
op = 0;
usage_if(argc != 2 || argv[1][0] != '-' || argv[1][2] != '\0');
op = argv[1][1];
fprintf(stderr, "Open /dev/i2c/0 with 8bit mode\n");
die_if(eeprom_open("/dev/i2c/0", 0x50, EEPROM_TYPE_8BIT_ADDR, &e) < 0,
"unable to open eeprom device file "
```

```
"(check that the file exists and that it's readable)");
switch(op)
{
case 'r':
fprintf(stderr, " Reading 256 bytes from 0x0\n");
read_from_eeprom(&e, 0, 256);
break;
case 'w':
fprintf(stderr, " Writing 0x00-0xff into 24C08 \n");
write_to_eeprom(&e, 0);
break;
default:
usage_if(1);
exit(1);
}
eeprom_close(&e);
return 0;
}
```

You can compile the program, download it and run

4.7.6 Pipe Program – Manipulating LED via Web

Program Description:	
Source Code Location	
Driver	
Device Type	
Device Name	
Test Program Source Code Location	/opt/FriendlyARM/tiny4412/linux/examples/led-player
Test Program Name	led-player.c
Executable Name	led-player
Test Program's Location in Board	
Note: to utilize math libraries you need to include its header file "pthread**.h**" and add an compile option **libpthread**	
Program:	

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/ioctl.h>
#include <sys/types.h>
```

```
#include <sys/stat.h>
#include <fcntl.h>
#include <sys/select.h>
#include <sys/time.h>
#include <string.h>
static int led_fd;
static int type = 1;
static void push_leds(void)
{
static unsigned step;
unsigned led_bitmap;
int i;
switch(type) {
case 0:
if (step >= 6) {
step = 0;
}
if (step < 3) {
led_bitmap = 1 << step;
} else {
led_bitmap = 1 << (6 - step);
}
break;
case 1:
if (step > 255) {
step = 0;
}
led_bitmap = step;
break;
default:
led_bitmap = 0;
}
step++;
for (i = 0; i < 4; i++) {
ioctl(led_fd, led_bitmap & 1, i);
led_bitmap >>= 1;
}
}
int main(void)
{
int led_control_pipe;
```

```
int null_writer_fd; // for read endpoint not blocking when control process exit
double period = 0.5;
led_fd = open("/dev/leds0", 0);
if (led_fd < 0) {
led_fd = open("/dev/leds", 0);
}
if (led_fd < 0) {
perror("open device leds");
exit(1);
}
unlink("/tmp/led-control");
mkfifo("/tmp/led-control", 0666);
led_control_pipe = open("/tmp/led-control", O_RDONLY | O_NONBLOCK);
if (led_control_pipe < 0) {
perror("open control pipe for read");
exit(1);
}
null_writer_fd = open("/tmp/led-control", O_WRONLY | O_NONBLOCK);
if (null_writer_fd < 0) {
perror("open control pipe for write");
exit(1);
}
for (;;) {
fd_set rds;
struct timeval step;
int ret;
FD_ZERO(&rds);
FD_SET(led_control_pipe, &rds);
step.tv_sec = period;
step.tv_usec = (period - step.tv_sec) * 1000000L;
ret = select(led_control_pipe + 1, &rds, NULL, NULL, &step);
if (ret < 0) {
perror("select");
exit(1);
}
if (ret == 0) {
push_leds();
} else if (FD_ISSET(led_control_pipe, &rds)) {
static char buffer[200];
for (;;) {
char c;
```

```
int len = strlen(buffer);
if (len >= sizeof buffer - 1) {
memset(buffer, 0, sizeof buffer);
break;
}
if (read(led_control_pipe, &c, 1) != 1) {
break;
}
if (c == '\r') {
continue;
}
if (c == '\n') {
int tmp_type;
double tmp_period;
if (sscanf(buffer,"%d%lf", &tmp_type, &tmp_period) == 2) {
type = tmp_type;
period = tmp_period;
}
fprintf(stderr, "type is %d, period is %lf\n", type, period);
memset(buffer, 0, sizeof buffer);
break;
}
buffer[len] = c;
}
}
}
close(led_fd);
return 0;
}
```

"make" will generate a led-player executable which is run as a server under "/sbin".

The leds.cgi gateway source code is under "/www/leds.cgi" on the board. It is a shell script and can be invoked by leds.html as an action. Here is the shell file

leds.cgi:

```
#!/bin/sh
type=0
period=1
```

```
case $QUERY_STRING in
*ping*)
type=0
;;
*counter*)
type=1
;;
*stop*)
type=2
;;
esac
case $QUERY_STRING in
*slow*)
period=0.25
;;
*normal*)
period=0.125
;;
*fast*)
period=0.0625
;;
esac
/bin/echo $type $period > /tmp/led-control
echo "Content-type: text/html; charset=gb2312"
echo
/bin/cat led-result.template
exit 0
```

4.8 Compile Qtopia-2.2.0

To make it easy for users we compile all the steps into one build script. Executing this script will compile thewhole qtopia platform and its utilities. You can start them by commanding "**run**". The compiling scripts for x86 and arm are a little bit different.

4.8.1 Compile and Run Qtopia-2.2.0 for X86

All our programs have been verified on Fedora9. We didn't try them on other

platforms. We strongly recommend our users to use Fedora9 and download it from ftp://download.fedora.redhat.com/pub/fedora/linux/releases/9/Fedora/i386/iso/Fedora-9-i386-DVD.iso.

Enter the working directory and run the following command

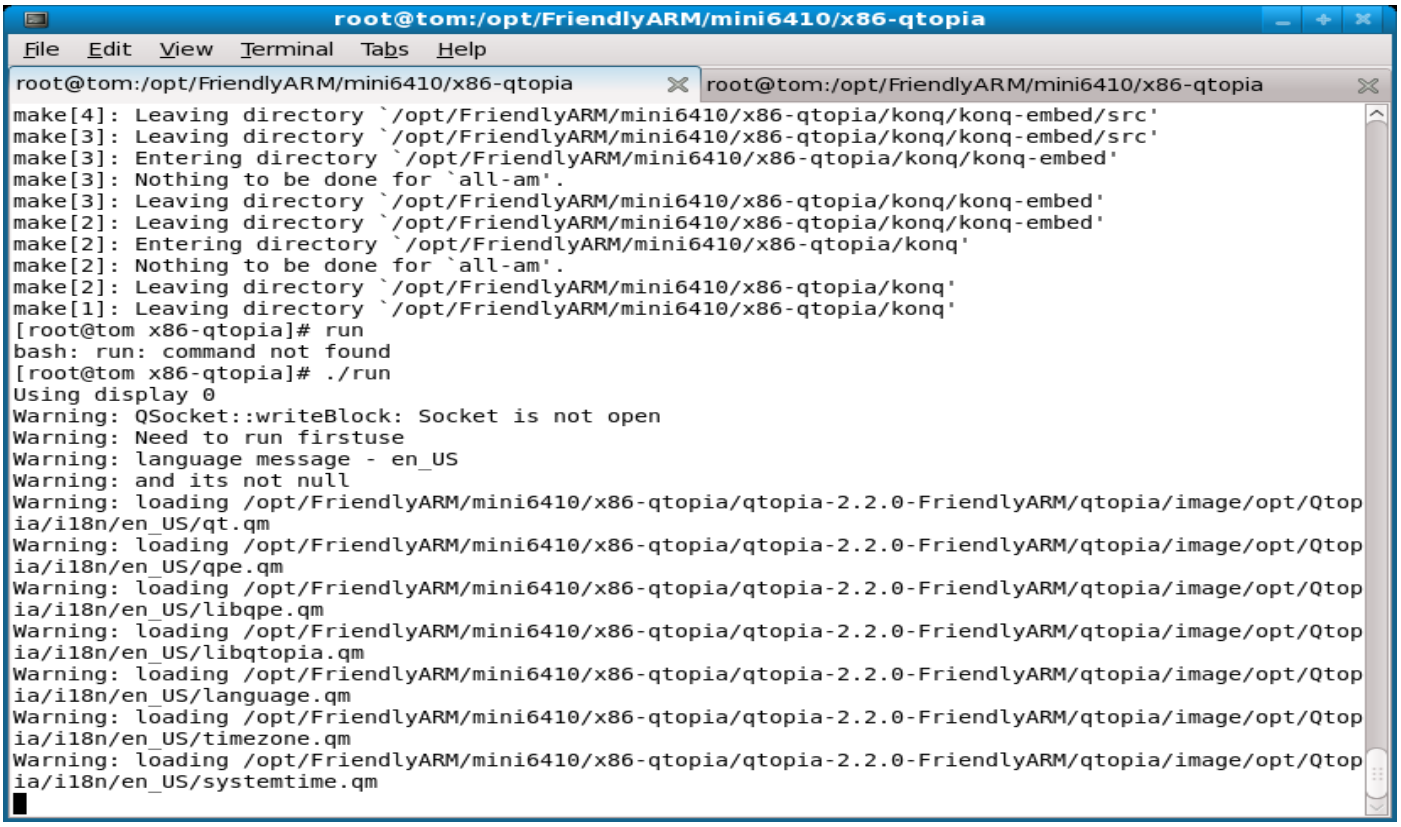
#**cd /opt/FriendlyARM/tiny4412/linux/x86-qtopia**

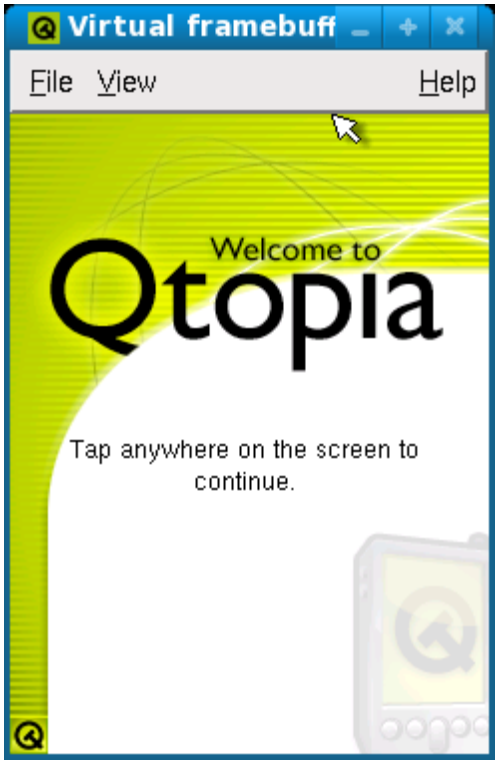
#**./build-all** (this process takes about 30 minutes)

Note: **./build-all** will automatically compile the complete Qtopia and its embedded web browser. You can execute "**./build**" first and then "./**build-konq**" to compile them separately. To run your qtopia you can type the command below:

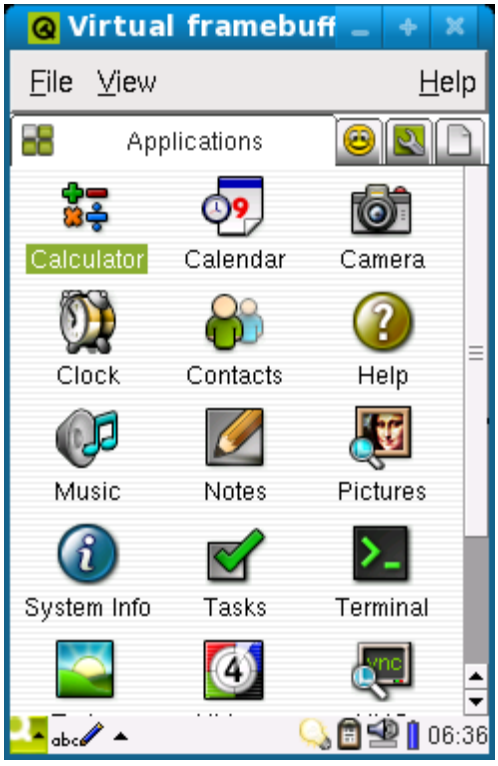
#./**run**

You will see the following screen





Follow the default options to continue and you will see the following screen



4.8.2 Compile and Run Qtopia-2.2.0 for ARM86

Please make sure your compiler is arm-linux-gcc-4.5.1 and platform is Fedora 9.

Enter the working directory and type the command below

#**cd /opt/FriendlyARM/tiny4412/linux/arm-qtopia**

#**./build-all** (this process takes about 30 minutes)

#./mktarget (this makes a file system image and will generate

"**target-qtopia-konq.tgz**")

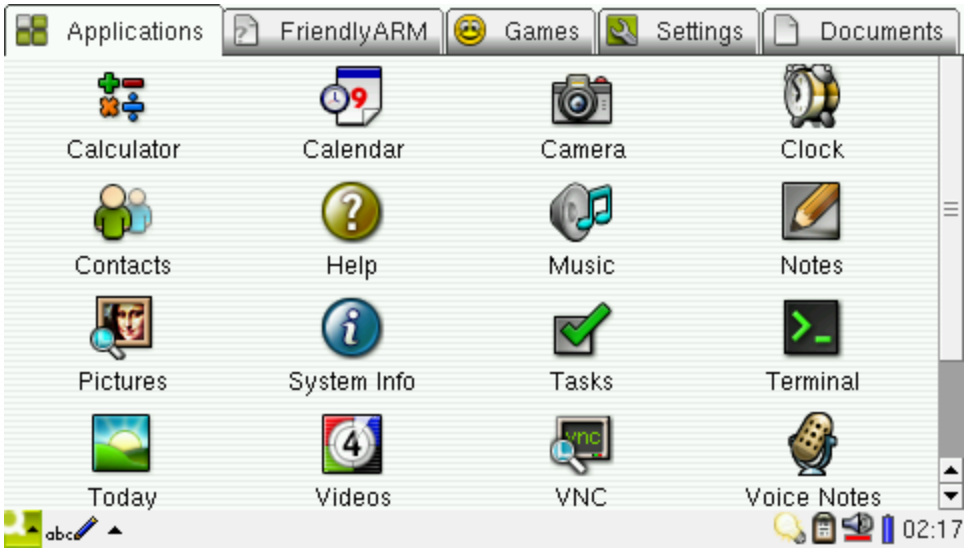
Note: "**./build-all**" will automatically compile a complete Qtopia system and the web browser and generate Jpeg, GIF, PNG image files. You can execute "**./build**" first and then "./**build-konq**" to compile them separately.

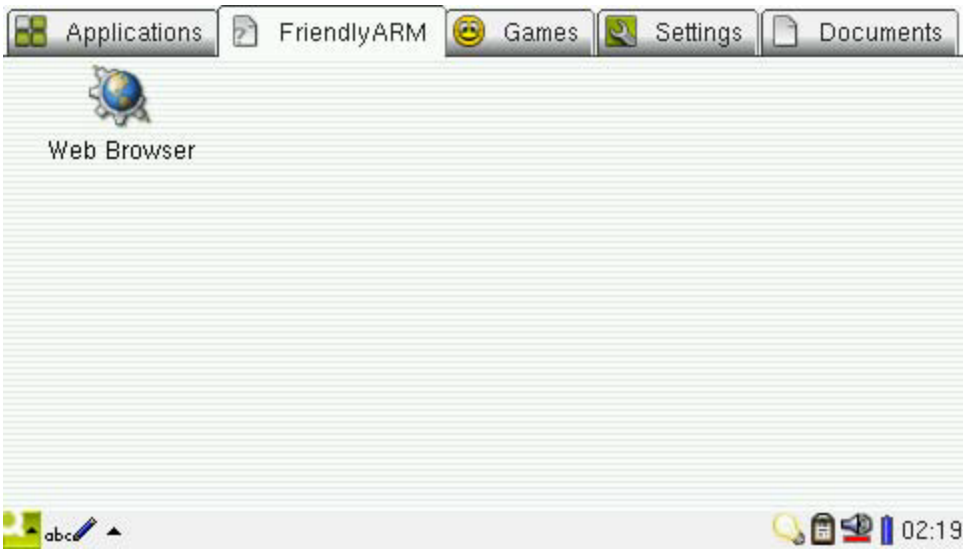
To remove your old Qtopia system you just need to delete all the files under "/opt". Then you can uncompress your target-qtopia-konq.tgz to the board's root directory via a flash drive. In our example we had it under /home/plg. Please run the command below:

#tar xvzf /home/plg/target-qtopia-konq.tgz –C /

"C" means "Change" and "/" after "C" means it will be uncompressed to the root directory. Afteryouare done, reboot your board and you will see that all your GUI components are in English now and there is a browser under the "FriendlyARM" tag. This is your own Qtopia.

Note: your new system may load parameters from "/etc/pointercal", you can delete that file too and will be directed to the calibration screen after reboot.





The above procedure is a simplied one. We hide all technical details in the build-all script you can look into it for more details

4.9 Compile QtE-4.7.0

4.9.1 Compile and Run QtE-4.7.0 for ARM

Note: please use our arm-linux-gcc-4.5.1 and Fedora9 to compile. We offered a build-all script for users to easily compile QtE-4.7.0. Please enter the source code

directory and type the following command:

#cd /opt/FriendlyARM/tiny4412/linux/arm-qte-4.7.0

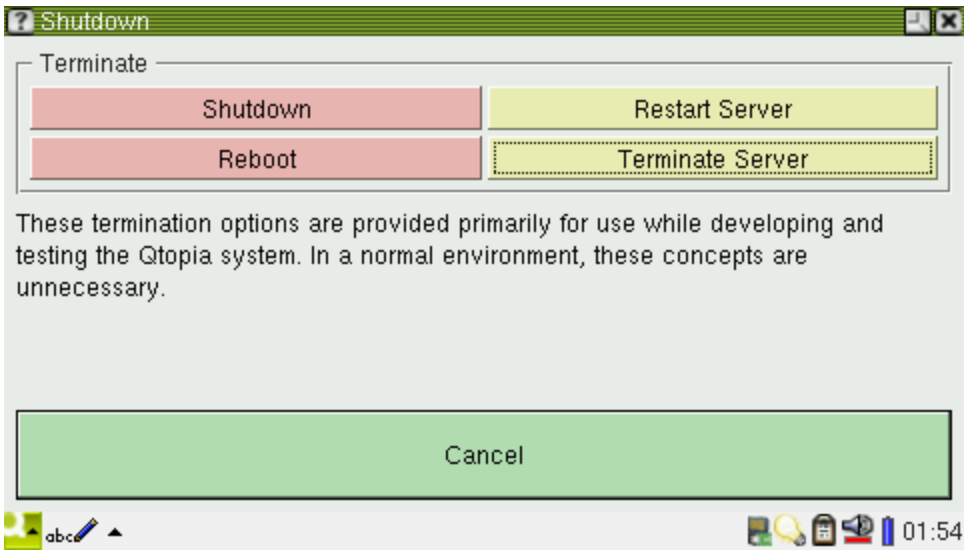
#./build-all

The build process takes a while. And after it is done, please run the mktarget script and a **target-qte-4.7.0.tgz** will be generated. Please follow the command below:

#tar xvzf target-qte-4.7.0.tgz –C /

A Trolltech directory will be generated under "/usr/local/", which includes all needed libraries and executables. Since our shipped Linux already includes QtE-4.7.0, to test your build you can delete the one on your board by "rm" the whole "**/usr/local/Trolltech**" directory.

Before running QtE-4.6.3, please stop the current running Qtopia-2.2.0. Go to "Settings" -> "Shutdown" and you will see the following screen. Click on "Terminate Server" to shut down Qtopia-2.2.0.



Or you can shut it down: either by commenting out the qtopia option in the init script

"/etc/init.d/rcS" and rebooting the system or commanding "kill all" to terminate related process (there are many options: you can even delete the whole "/opt", shut down qtopia-2.2.0 and run "qt4"

4.10 Compile Qtopia4(Qt-Extended-4.4.3)

4.10.1 Compile and Run Qt-Extended-4.4.3 for X86

Note: please use our arm-linux-gcc-4.5.1 and Fedora9 to compile. We offered a build-all script for users to easily compile Qt-Extened-4.4.3. Please enter the source code directory and type the following command:

#cd /opt/FriendlyARM/tiny4412/linux/x86-qt-extended-4.4.3

#./build

The build process takes a while. To run your compiled system please type the command below:

#./run

Now you will see the following screen:

4.10.2 Compile and Run Qt-Extended-4.4.3 for ARM

Note: please use our arm-linux-gcc-4.5.1 and Fedora9 to compile. We offered a build script for users to easily compile Qt-Extened-4.4.3. Please enter the source code directory and type the following command:

#**cd /opt/FriendlyARM/tiny4412/linux/arm-qt-extended-4.4.3**

#**./build**

The build process takes a while. To run your compiled system please type the command below:

#./**run**

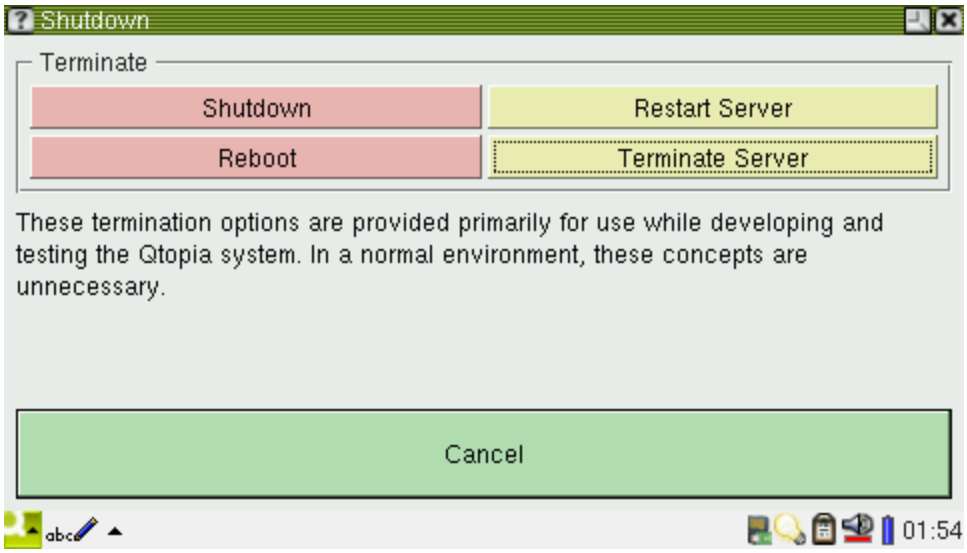
And after it is done, please run the mktarget script and a **target-qtopia4.tgz** will be

generated. Please follow the command below:

#tar xvzf target-qtopia4.tgz –C /

A Qtopia4.4.3 directory will be generated under "/opt", which includes all needed libraries and executables. Since our shipped Linux already includes QtE-4.7.0, to test your build you can delete the one on your board by "rm" the whole "**/opt/Qtopia4.4.3**" directory.

Before running Qtopia4, please stop the current running Qtopia-2.2.0. Go to "Settings" -> "Shutdown" and you will see the following screen. Click on "Terminate Server" to shut down Qtopia-2.2.0.



Or you can shut it down: either by commenting out the qtopia option in the init script "/etc/init.d/rcS" and rebooting the system or commanding "kill all" to terminate related process (there are many options: you can even delete the whole "/opt", shut down qtopia-2.2.0 and run "qtopia4 &"


2:16
Qt Extended
3/7/00
Options
Menu
Quit

5 Linux Application Development

We have another document which has very detailed information about how to do Linux development applications.