 [slojong](#) / [gnublinx](#)

👁 Watch1

★ Star0

🍴 Fork0

🔗 Code

🔔 Issues0

🔀 Pull requests0

📁 Projects0

🛡 Security

📊 Insights

Join GitHub today

Dismiss

GitHub is home to over 40 million developers working together to host and review code, manage projects, and build software together.

Sign up

Cross-compile the linux kernel 2.6.33 for the Gnublin board. <http://gnublin.org>

📌 22 commits

🌿 1 branch

📦 0 releases


👤 1 contributor

Branch: master ▾

New pull request

Find file

Clone or download ▾

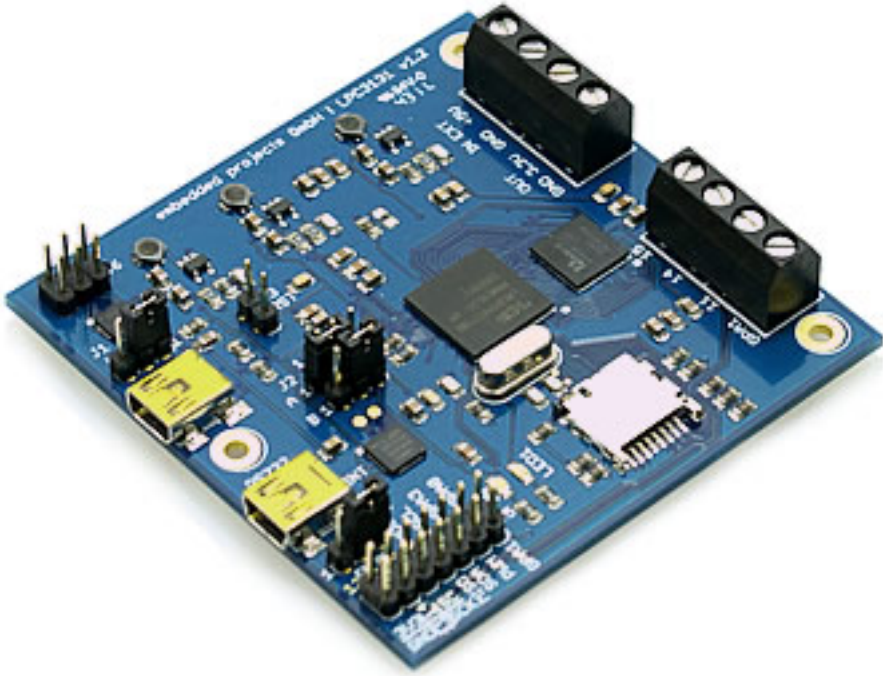
 [slojong](#) Link to image corrected

Latest commit 20333c7 on 3 Feb 2013

📁 gnublin @ 81fd024	Added the submodules and a script preparing the linux source for comp...	7 years ago
📁 linux-2.6.33-lpc313x @ 33debcf	Added the submodules and a script preparing the linux source for comp...	7 years ago
📁 sd-card	Extra script for the mount operation.	7 years ago
📄 .gitmodules	Added the submodules and a script preparing the linux source for comp...	7 years ago
📄 0006_lpc313x_ADC_PWM_2.patch	Added another patch for the ADC/PWM driver and a clean script.	7 years ago
📄 Gnublin.jpg	Wrote a little how-to for linux beginners.	7 years ago
📄 README.md	Link to image corrected	7 years ago
📄 build.sh	Commented out the mount script.	7 years ago
📄 clean.sh	Scripts improved.	7 years ago
📄 configure.sh	Mounting the SD memory card to sd-card and setting ARCH and CROSS_COM...	7 years ago
📄 mount.sh	Scripts improved.	7 years ago
📄 patch.sh	<a href="#">Added another patch for the ADC/PWM driver and a clean script.</a>	7 years ago
📄 setenv.sh	Mounting the SD memory card to sd-card and setting ARCH and CROSS_COM...	7 years ago

📖 README.md

# gnublinx



Gnublin is a low-cost embedded GNU/Linux board based on a low-power ARM9 microcontroller unit.

Some key features of the board:

- CPU Speed: 180 MHz
- Selectable boot-up: SD/MMC, UART, DFU

The board makes the following interfaces available to the outside world:

- 1-Wire
- A/D converter
- I2C
- PWM
- SPI
- USB 2.0 OTG (Host, Device)
- UART

The UART is not directly accessible on a pin header. On-board there's a USB UART bridge and at boot time the UART is configured as a console. Though after soldering two wires to the two soldering pads nearby the bridge you can use the UART directly.

## Operating system

The Gnublin board comes with a SD memory card and a pre-installed linux. The board is ready to be used, plug a USB cable to the port labeled with RS232 and start a serial communication program such as minicom and picocom. The hardware flow control must be turned off and the bitrate set to 115200 with the parameter setting 8N1 (8 data bits, no parity bit and one stop bit).

## Compiling the kernel & modules

This is the most interesting part of this documentation. If you never cross-compiled something or even never compiled a kernel and modules then gnublinx is made for you. It provides some scripts to patch the original 2.6.33 LPC linux and to compile it afterwards.

The first step is to install the cross-compiler toolchain. This documentation doesn't go into depth but you can follow this [guide](#). Unfortunately the gnublin wiki is in German but all the commands to be executed are listed there. The next step then is to get the source. Either use git or get it on the [download page](#). The patch is done to introduce some modifications required to get the LPC Linux run on the Gnublin board. The build script actually compiles the kernel and all configured modules.

```
git clone --recursive git://github.com/slojong/gnublinx.git
./patch.sh
sudo ./build.sh /media/gnublin
```

The build script asks for the mount point of your SD memory card with your Gnublin linux system. If you don't specify it, the kernel and its modules will be installed to `/tmp/gnublin`. And the script must be called as root. The reason for this is that all the files on the SD memory card belong root. The script will fail with permissions denied error messages if you run it as a non-privileged user. Until now there's no fakeroot environment used which would allow you with no root privileges to install the files as you were root.


Essentially the build script enters the linux source directory and calls 'make kernel' and 'make modules'. After the compilation it makes a backup of your old kernel and modules and then copies the new kernel with its modules to the memory card if you passed the mount point.

If nothing went wrong you should have cross-compiled and installed the kernel and the modules. Congratulations!

Now if you are experimenting with configuring and cross-compiling you can reset the previous done work by calling `./clean.sh`.

Hopefully after experimenting and reading the build script this brief how-to helps you to better understand the steps to be done to get run a new kernel or modules. If something is unclear, leave an issue [here](#).

© 2019 GitHub, Inc. [Terms](#) [Privacy](#) [Security](#) [Status](#) [Help](#)



[Contact GitHub](#) [Pricing](#) [API](#) [Training](#) [Blog](#) [About](#)