# Build and deploy Embedded Linux for BeagleBone Black

## Components of Embedded Linux

### **Toolchain**

In order to develop any application you need a toolchain and it is the first element of embedded linux.

Toolchain should support all the languages that you would be using for your development and it is better to choose the one which has robust implementation of the Portable Operating System Interface (POSIX) and other system interfaces.

Your toolchain should remain constant throughout your project life cycle so that you avoid subtle bugs.

We can obtain the toolchain by downloaing and installing the .tar file but here we will choose to build the toolchain from source code with the help of a tool called crosstool-NG.

Technical requirements:

* A Linux-based host system with autoconf, automake, bison, bzip2, cmake, flex, g++, gawk, gcc, gettext, git, gperf, help2man, libncurses5-dev, libstdc++6, libtool, libtool-bin, make, patch, python3-dev, rsync, texinfo, unzip, wget, and xz-utils or their equivalents installed.

$ sudo apt-get install autoconf automake bison bzip2 cmake flex g++ gawk gcc gettext git gperf help2man libncurses5-dev libstdc++6 libtool libtool-bin make patch python3-dev rsync texinfo unzip wget xz-utils

A toolchain is a set of tools that compiles source code into executables that can run on your target device and includes a compiler, a linker, and runtime libraries.

Initially, you need one to build the other three elements of an embedded Linux system: the bootloader, the kernel, and the root filesystem.

It has to be able to compile code written in assembly,C, and C++ since these are the languages used in the base open source packages.

Projects included in the GNU toolchain are: :

* [GNU make](https://en.wikipedia.org/wiki/Make_(software)): an automation tool for compilation and build
* [GNU Compiler Collection](https://en.wikipedia.org/wiki/GNU_Compiler_Collection) (GCC): a suite of compilers for several programming languages
* [GNU C Library](https://en.wikipedia.org/wiki/GNU_C_Library) (glibc): core C library including headers, libraries, and dynamic loader
* [GNU Binutils](https://en.wikipedia.org/wiki/GNU_Binutils): a suite of tools including linker, assembler and other tools
* [GNU Bison](https://en.wikipedia.org/wiki/GNU_Bison): a [parser generator](https://en.wikipedia.org/wiki/Parser_generator), often used with the [Flex lexical analyser](https://en.wikipedia.org/wiki/Flex_lexical_analyser)
* [GNU m4](https://en.wikipedia.org/wiki/GNU_m4): an [m4](https://en.wikipedia.org/wiki/M4_(computer_language)) [macro processor](https://en.wikipedia.org/wiki/General-purpose_macro_processor)
* [GNU Debugger](https://en.wikipedia.org/wiki/GNU_Debugger) (GDB): a code debugging tool
* [GNU Autotools](https://en.wikipedia.org/wiki/GNU_Autotools) (GNU Build System): [Autoconf](https://en.wikipedia.org/wiki/Autoconf), [Automake](https://en.wikipedia.org/wiki/Automake) and [Libtool](https://en.wikipedia.org/wiki/Libtool)

Types of toolchains:

* **Native**: This toolchain runs on the same type of system (sometimes the same actual system) as the programs it generates. This is the usual case for desktops and servers, and it is becoming popular on certain classes of embedded devices. The Raspberry Pi running Debian for ARM, for example, has self-hosted native compilers.
* **Cross**: This toolchain runs on a different type of system than the target, allowing the development to be done on a fast desktop PC and then loaded onto the embedded target for testing.

Almost all embedded Linux development is done using a cross-development toolchain, partly because most embedded devices are not well suited to program development since they lack computing power, memory, and storage, but also because it keeps the host and target environments separate.

### **Bootloader**

### Kernel

### Root Filesystem