embitel
delivered with passion

Workflow Guide : Leveraging the power of YOCTO

Linux OS Porting for Embedded Applications

This document is created for the community of developers working on embedded Linux applications. The purpose of the document is to guide you through the steps of building an image for Linux distribution, using the Yocto Framework.

P.S – In order to best utilize the information in this document, the following know-how is a must:

- Familiarity with basic Linux utilities and text editors
- Experience with Python is helpful, but not a necessity.
- Understanding of the Yocto Project and Open Embedded Project.
- Basic understanding of the build systems

Understanding System Requirements and the OpenEmbedded (OE) Build System:

Before we discuss about the workflow, it is imperative to develop a basic understanding regarding the system requirements and OE Build System. Here are the necessary details.

a. System Requirements:

1. **A Host System**: Ideally a system with a minimum free disk space of 50 Gbytes and that runs on any Linux distribution (i.e. Ubuntu, Fedora, CentOS, openSUSE, or Debian,). Most often a native Linux machine function is used as the development host.

***2*. Packages for Builds**: Do ensure that your host development system has the following packages (with respect to Linux Distribution – Ubuntu, Fedora, CentOS & more)

- Essentials- like gawk, wget, git-core, diffstat, unzip, texinfo, build-essential, socat, cpio, python, xz-utils etc
- Graphical and Eclipse Plug-In Extras–like libsdl1.2-dev,xterm etc
- Documentation –like docbook-utils ,fop ,dblatex ,xmlto

3. Any release of the Yocto Project

b. OpenEmbedded Build System and BitBake Tool

OpenEmbedded (OE) is the build system for the Yocto Project. The central component of this build system is **BitBake.**

BitBake performs tasks like parsing the Metadata, Creating task lists from the Metadata and more.

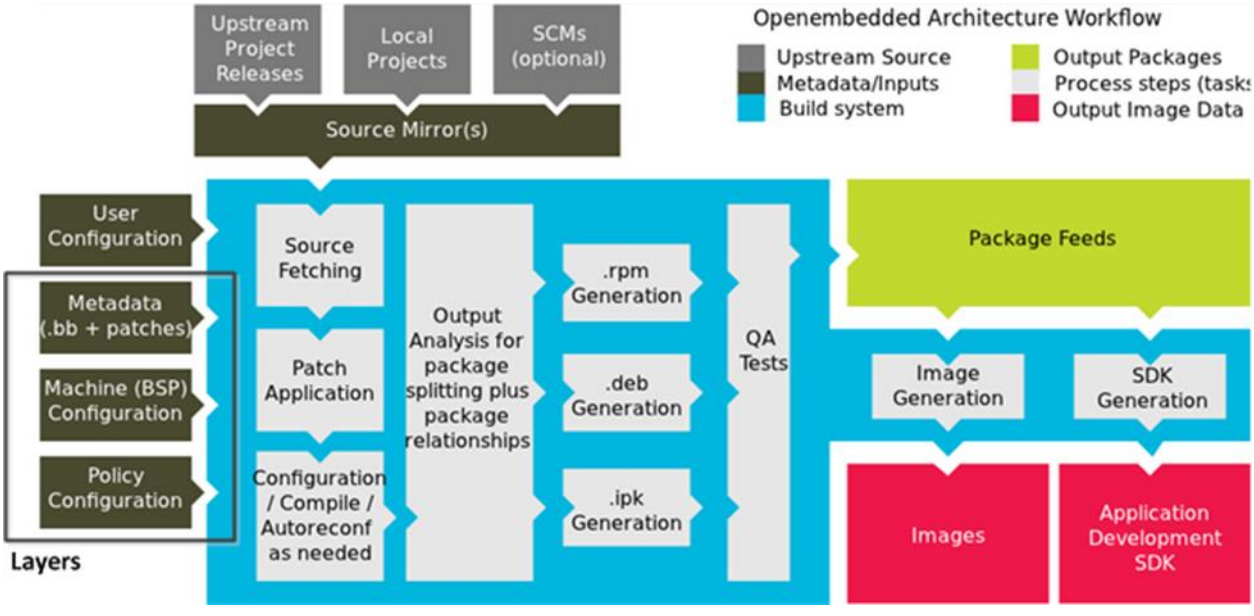


Figure 1: YoctoIDE

The BitBake tool consists of the following functional blocks:

a) User Configuration: This includes the metadata for managing the YOCTO build process. As a developer, you can define the build environment by specifying target architecture, location to store the downloaded source, and other build properties using User Configuration file.

b) Metadata, Machine & Policy Configuration Layers: These layers consist of data critical for the management of the build process.

- **Metadata:** This layer consists of user-supplied recipe files, patches, and append files.

- **Machine Configuration (BSP):** This layer consists of information specific to your target architecture for which the image is being built. The information specific to the machine configuration is provided by the BSP layer of the Yocto layered architecture model.

- **Distro Layer/Policy Configuration:** This layer consists of data that specifies the policy configurations for the specific distribution. This layer includes class files, configuration files and recipes. These recipes would include custom image recipes, distribution-specific configuration, initialization scripts.

c) Source Files: These include sources such as Upstream releases, local projects, and Source Control Manager (SCMs), from where the build system downloads source files to build an image.

d) Build System: This block specifies the processes during which the BitBake fetches source, applies patches, executes compilation, analyzes output for package generation, generates and tests these packages, generates images and cross-development tools.

e) Package Feeds: This module consists of directories with various types of output packages in RPM, DEB or IPK format. Package feeds are used while building an image or SDK, produced by the build system. They are also used for extending or updating existing images on devices at runtime by copying and sharing them on web server.

f) Images: This is an output module that manages the Linux Images created by the Build System.

g) Application Development SDK: The module consists of various cross-development tools, which are built either along with the image or separately with BitBake.

Workflow for using Yocto project as a build system for Embedded Linux:

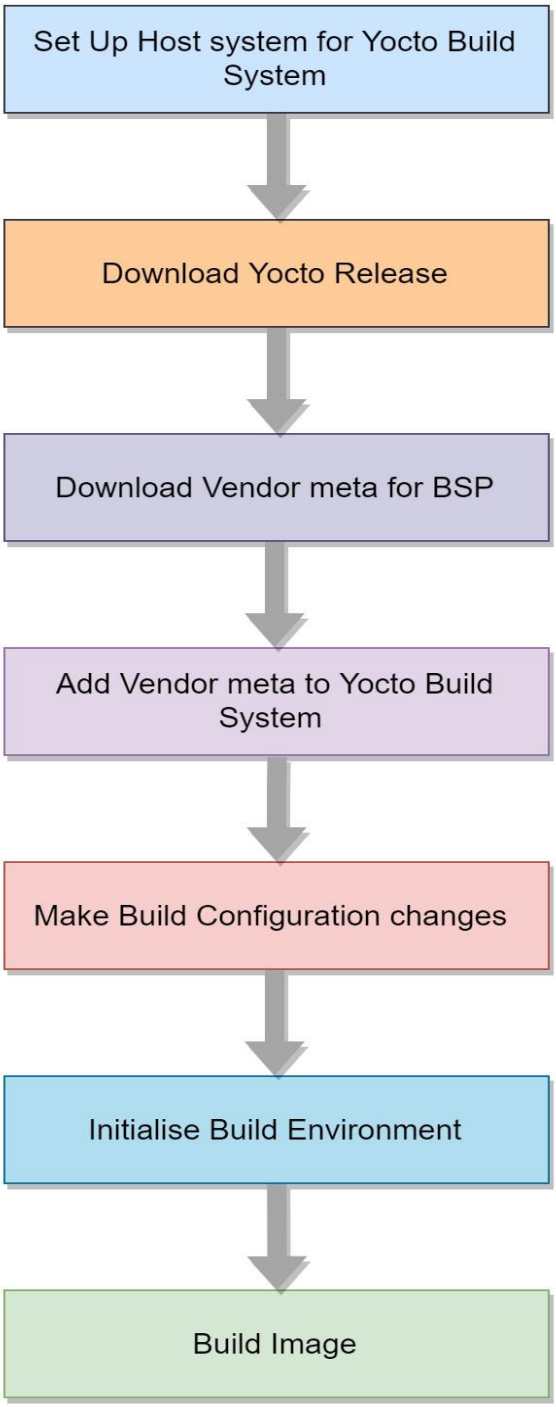


Figure 2: Workflow: Building Linux Image using Yocto

1. **Setup the Host System for YOCTO Build System:** The host system should comply with the minimum system requirements, as mentioned before.
   Additionally, you should test your host build system for the following:

   a. Required Packages
   b. Build system is meeting the Minimal Version Requirements of Git, tar, and Python

2. Download the required version of the YOCTO release: Set up the latest Yocto Project files on your host development system by cloning a local copy of the Poky Git repository.

```
$ git clone http://git.yoctoproject.org/git/poky

$ cd poky

$ git checkout -b fido origin/fido (Any poky release branch
 may be checked out like Jethro,Dizzy,Daisy,Dora)
```

3. Download the vendor provided meta for the BSP: Depending on the processor platform (ARM, MIPS, PowerPC, and x86), you can download the meta data for BSP provided by the specific vendor.

```
$ git clone git://git.yoctoproject.org/meta-ti meta-ti

$ cd meta-ti

$ git checkout -b fido origin/fido
```

4. Add the Vendor meta to the Yocto build system: Add the Vendor meta to your build host environment. For example, meta-ti is the metadata for the ti specific target boards .

```
Vim conf/bblayers.conf

BBLAYERS ?= " \
##OEROOT##/meta \
##OEROOT##/meta-yocto \
##OEROOT##/meta-ti \                    // ti layer added
"
```

5. Make the Build configuration changes: Check the local configuration file and make the build configuration changes by editing the local.conf files. This should be done before calling the BitBake command to initialize the build.

```
$ vim conf/local.conf

MACHINE ?= "beaglebone"     // if building for beagle bone
```

6. Initialize the Build Environment: To define the Open Embedded build environment, the specific setup script on the build host is executed.

```
$ source oe-init-build-env
```

The script creates a Build Directory, which is located in the Source Directory. After that the current working directory is set to the Build Directory. Once the build is completed, the Build Directory will have all the files created during the build.

7. Start Building the image: Now the YOCTO IDE/ framework has received all the commands necessary for it to build the Linux image. Next through a series of actions Yocto (IDE)will build the image as per the information/ specifications in the metadata.

$bitbake -k core-image-minimal

8. Writing the Linux image: Depending on the information provided in the TARGET_DEVICE command, you can write the Linux image on any of the target device like SATA drive, SD card or even USB key with the help of mkefidisk.sh script included in the poky repository.

Why Use Yocto framework for Linux projects?

1. **Easy customization:** Yocto has a very robust and powerful customization architecture which offers numerous customization options such as footprint sizes, enabling/ disabling of components like graphics subsystems, visualization middleware, and services.

2. **Vendor Support:** Yocto Project enjoys the support of most of the semiconductor and OS vendors and major electronic manufacturing firms. Thus with Yocto, you can leverage a solid support ecosystem and achieve your project goals.

3. **Reusability:** The application developed using Yocto can be reused and build for different boards by changing the vendor meta. This allows for reuse of code and resources across similar product lines.

4. Simplified Build operations for Embedded Linux applications: Single common framework of Yocto has helped get rid of the dependencies on discrete build systems wherein each of the SoC vendors created their own build framework compatible only with their microprocessor platforms.

5. Seamless addition of UI components: Support for enhancing user experience for devices with display. This is facilitated by system components like Qt, Clutter, such as X11, GTK+, and SDL .

6. Emulator support: It supports hardware and device emulation with the QEMU Emulator. Thus, an image built through Yocto Project can boot inside a QEMU emulator, with the development environment acting as a test platform for embedded software.

7. Added Convenience: With its customizable recipes, tools and templates for building systems and porting OS, Yocto enables the developers to focus on other core development tasks.

8. Systematically Managed & Updated: Every 6 months, a new release of Yocto including kernel (LTSI), toolchain, and package versions is released.

9. Reduced Time To Market for Crucial Embedded Linux applications: With Yocto, a developer can build the entire Linux system from scratch in few hours, depending on the project components.

10. Readily Available Development Tools: As a Yocto user, you get access to a wide array of development tools like Application development Toolkit (ADT), ECLIPSE IDE Plug-in, Graphical UI for Embedded devices (Matchbox), tools for QA – among others.

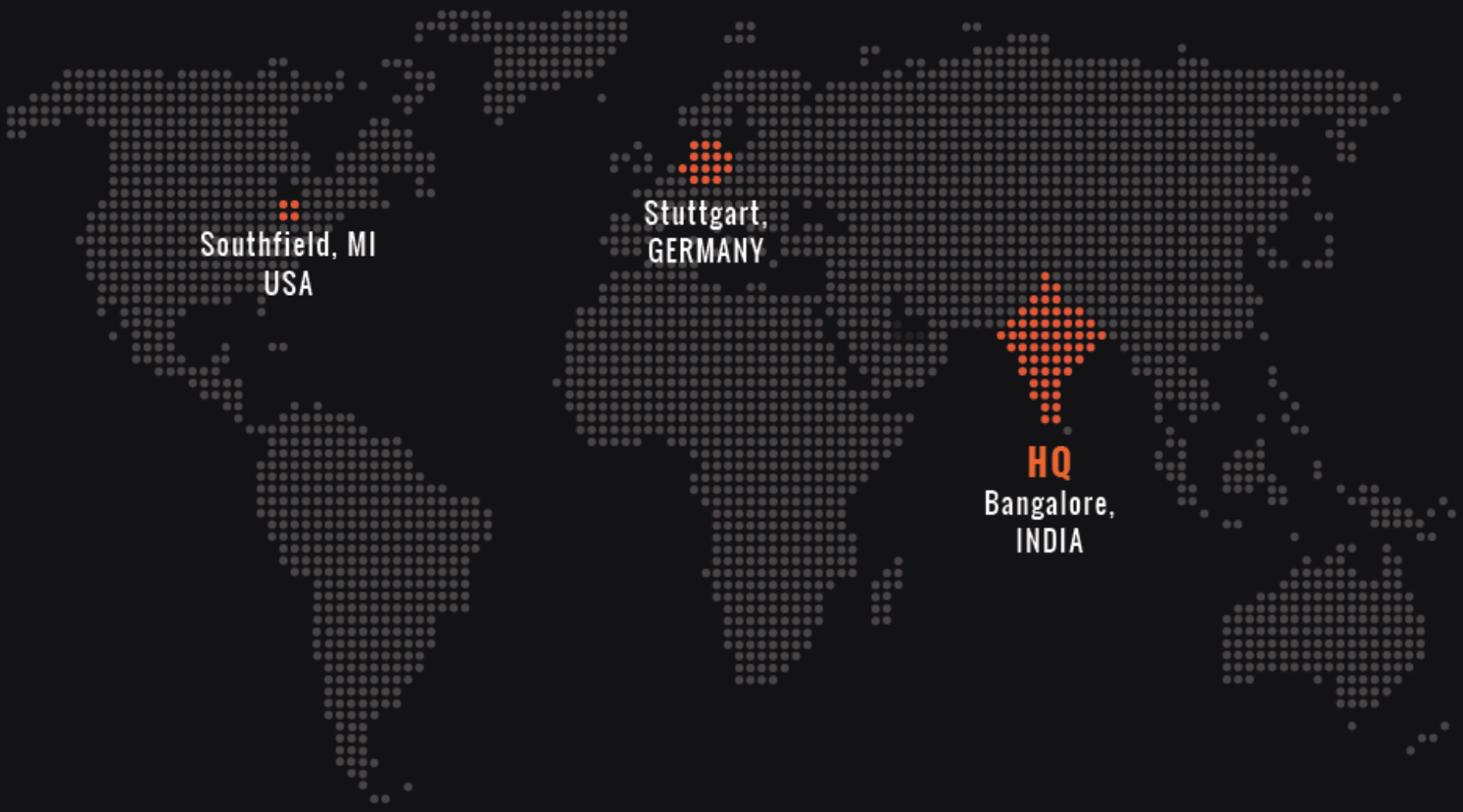
If you wish to know more about how you can use the YOCTO Project for creating Linux distributions for your embedded application development projects, please get in touch with our Product Engineering Team

about us

"Delivered with Passion" - an experience which our customers and partners have shared with us for more than 10 years. We are a product engineering services and solutions company with highly efficient project management and delivery processes.

Our Success Mantra - Domain Focus + Delivery Excellence + People Power + Partnership Ecosystem

development centres



USA

Embitel Technologies Inc.

24209 Northwestern Hwy
Suite 200A

Southfield, MI
48075

phone: +1 248 385 2017

www.embitel.com

Germany

embitel GmbH

Rommelstraße 11

70376 Stuttgart

phone: +49 (0)711 60 17 47-0
fax: +49 (0)711 60 17 47-141
mobile: +49 (0)170 16 88 0 28

www.embitel.de

India

Embitel Technologies
(India) Pvt. Ltd.

208 - 216, 2nd Floor,
Delta Block, Sigma Soft
Tech Park

Varthur-Hobli
Bangalore 560066

phone: +91 80 4169 4211
fax: +91 80 4169 4201

www.embitel.com

embitel
delivered with passion