# Yocto Project and OpenEmbedded development training

**On-site training, 3 days**

<table>
<thead>
<tr>
<th>Title</th>
<th>Yocto Project and OpenEmbedded development training</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Training objectives</strong></td>
<td>• Be able to understand the role and principle of an embedded Linux build system, and compare Yocto Project/OpenEmbedded to other tools offering similar functionality.&lt;br&gt;• Be able to configure and build basic embedded Linux system with Yocto, and install the result on an embedded platform.&lt;br&gt;• Be able to write and extend recipes, for your own packages or customizations.&lt;br&gt;• Be able to use existing layers of recipes, and create your own new layers.&lt;br&gt;• Be able to integrate support for your own embedded board into a BSP layer.&lt;br&gt;• Be able to create custom images.&lt;br&gt;• Be able to use the tools and workflows suitable to develop applications with the Yocto Project SDK.</td>
</tr>
<tr>
<td><strong>Materials</strong></td>
<td>Check that the course contents correspond to your needs: <a href="https://bootlin.com/doc/training/yocto">https://bootlin.com/doc/training/yocto</a></td>
</tr>
<tr>
<td><strong>Duration</strong></td>
<td><strong>Three</strong> days - 24 hours (8 hours per day).&lt;br&gt;40% of lectures, 60% of practical labs.</td>
</tr>
<tr>
<td><strong>Trainer</strong></td>
<td>One of the engineers listed on <a href="https://bootlin.com/training/trainers/">https://bootlin.com/training/trainers/</a></td>
</tr>
<tr>
<td><strong>Language</strong></td>
<td>Oral lectures: English, French.&lt;br&gt;Materials: English.</td>
</tr>
<tr>
<td><strong>Audience</strong></td>
<td>Companies and engineers interested in using the Yocto Project to build their embedded Linux system.</td>
</tr>
</tbody>
</table>
| **Prerequisites** | **Knowledge of embedded Linux** as covered in our embedded Linux training ([https://bootlin.com/training/embedded-linux/](https://bootlin.com/training/embedded-linux/))

**Knowledge and practice of UNIX or GNU/Linux commands**

People lacking experience on this topic should get trained by themselves, for example with our freely available on-line slides: [https://bootlin.com/blog/command-line/](https://bootlin.com/blog/command-line/)
### Required equipment

For on-site sessions at our customer location, the customer must provide:

- Video projector
- One PC computer on each desk (for one or two persons) with at least an Intel Core i5 processor, 8 GB of RAM, and Ubuntu Linux 20.04 installed in a **free partition of at least 50 GB**
- Distributions others than Ubuntu Linux 20.04 are not supported, and using Linux in a virtual machine is not supported.
- **Unfiltered and fast connection to Internet**: at least 50 Mbit/s of download bandwidth, and no filtering of web sites or protocols.
- **PC computers with valuable data must be backed up** before being used in our sessions.

### Materials

Electronic copies of presentations and labs.
Electronic copy of lab files.

### Hardware, first option

BeagleBone Black board
- An ARM AM335x processor from Texas Instruments (Cortex-A8 based), 3D acceleration, etc.
- 512 MB of RAM
- 2 GB of on-board eMMC storage (4 GB in Rev C)
- USB host and device
- HDMI output
- 2 x 46 pins headers, to access UARTs, SPI buses, I2C buses and more.

![BeagleBone Black board](image)

### Hardware, second option

STMicroelectronics STM32MP157D-DK1 Discovery board
- STM32MP157D (dual Cortex-A7) CPU from STMicroelectronics
- USB powered
- 512 MB DDR3L RAM
- Gigabit Ethernet port
- 4 USB 2.0 host ports
- 1 USB-C OTG port
- 1 Micro SD slot
- On-board ST-LINK/V2-1 debugger
- Arduino Uno v3-compatible headers
- Audio codec
- Misc: buttons, LEDs

![STMicroelectronics STM32MP157D-DK1 Discovery board](image)
### Day 1 - Morning

**Lecture - Introduction to embedded Linux build systems**
- Overview of an embedded Linux system architecture
- Methods to build a root filesystem image
- Usefulness of build systems

**Lecture - Overview of the Yocto Project and the Poky reference system**
- Organization of the project source tree
- Building a root filesystem image using the Yocto Project

**Lab - First Yocto Project build**
- Downloading the Poky reference build system
- Building a system image

### Day 1 - Afternoon

**Lecture - Using Yocto Project - basics**
- Organization of the build output
- Flashing and installing the system image

**Lab - Flashing and booting**
- Flashing and booting the image on the board

**Lecture - Using Yocto Project - advanced usage**
- Configuring the build system
- Customizing the package selection

**Lab - Using NFS and configuring the build**
- Configuring the board to boot over NFS
- Learn how to use the PREFERRED_PROVIDER mechanism
**Day 2 - Morning**

<table>
<thead>
<tr>
<th>Lecture - Writing recipes - basics</th>
<th>Lab - Adding an application to the build</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Writing a minimal recipe</td>
<td>• Writing a recipe for <em>nInvaders</em></td>
</tr>
<tr>
<td>- Adding dependencies</td>
<td>• Adding <em>nInvaders</em> to the final image</td>
</tr>
<tr>
<td>- Development workflow with <em>bitbake</em></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Lecture - Writing recipes - advanced features</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Extending and overriding recipes</td>
</tr>
<tr>
<td>- Adding steps to the build process</td>
</tr>
<tr>
<td>- Learn about classes</td>
</tr>
<tr>
<td>- Analysis of examples</td>
</tr>
<tr>
<td>- Logging</td>
</tr>
<tr>
<td>- Debugging dependencies</td>
</tr>
</tbody>
</table>

**Day 2 - Afternoon**

<table>
<thead>
<tr>
<th>Lab - Learning how to configure packages</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Extending a recipe to add configuration files</td>
</tr>
<tr>
<td>• Using <code>ROOTFS_POSTPROCESS_COMMAND</code> to modify the final <em>rootfs</em></td>
</tr>
<tr>
<td>• Studying package dependencies</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Lecture - Layers</th>
<th>Lab - Writing a layer</th>
</tr>
</thead>
<tbody>
<tr>
<td>- What layers are</td>
<td>- Learn how to write a layer</td>
</tr>
<tr>
<td>- Where to find layers</td>
<td>- Add the layer to the build</td>
</tr>
<tr>
<td>- Creating a layer</td>
<td>- Move <em>nInvaders</em> to the new layer</td>
</tr>
</tbody>
</table>

•
### Day 3 - Morning

**Lecture - Writing a BSP**
- Extending an existing BSP
- Adding a new machine
- Bootloaders
- Linux and the linux-yocto recipe
- Adding a custom image type

**Lab - Implementing the kernel changes**
- Extend the kernel recipe to add the nunchuk driver
- Configure the kernel to compile the nunchuk driver
- Play *nInvaders*

### Day 3 - Afternoon

**Lecture - Creating a custom image**
- Writing an image recipe
- Adding users/groups
- Adding custom configuration
- Writing and using package groups recipes

**Lab - Creating a custom image**
- Writing a custom image recipe
- Adding *nInvaders* to the custom image

**Lecture - Creating and using an SDK**
- Understanding the purpose of an SDK for the application developer
- Building an SDK for the custom image

**Lab - Experimenting with the SDK**
- Building an SDK
- Using the Yocto Project SDK