# Erstellung einer virtuellen Maschine:

Um Yocto benutzen zu können mussten wir zur aller erst eine virtuelle Linux Maschine erstellen Yocto nur auf einem Linux basierten Betriebssystem funktioniert. Die folgenden Betriebssysteme wurden von den Entwicklern getestet und als stabil bezeichnet:

* Ubuntu 12.04 (LTS)
* Ubuntu 13.10
* Ubuntu 14.04 (LTS)
* Fedora release 19 (Schrödinger's Cat)
* Fedora release 20 (Heisenbug)
* CentOS release 6.4
* CentOS release 6.5
* Debian GNU/Linux 7.x (Wheezy)
* OpenSUSE 12.2

Da wir mit Ubuntu bis jetzt am meisten Erfahrung hatten, haben wir uns recht schnell dafür entschieden Ubuntu 14.04 (LTS) zu benutzen.

# Yocto Installation:

Sobald wir mit der Erstellung einer virtuellen Maschine fertig waren mussten wir uns um die Installation vom Yocto Projekt kümmern.

Um Yocto (bzw. Poky) zu installieren mussten wir zuerst die folgenden Pakete mittels der Kommandozeile „apt-get install“ installieren:

* gawk
* wget
* git-core
* diffstat
* unzip
* texinfo
* gcc-multilib
* build-essential
* chrpath
* socat
* cpio
* python
* python3
* python3-pip
* python3-pexpect
* xz-utils
* debianutils
* iputils-ping
* libsdl1.2-dev
* xterm

Danach haben wir im Homeverzeichnis einen „Yocto“ Ordner angelegt indem wir mittels der Kommandozeile „git clone git://git.yoctoproject.org/poky --branch daisy“ den source Code vom Yocto Projekt heruntergeladen haben.

# Erstellung eines einfachen Images:

Mit Hilfe der Kommandozeile „source poky/oe-init-build-env“ wird es schließlich möglich Poky zu benutzen. Poky benutzt „BitBake“ um aus „Rezpeten“, Images zu machen. Bitbake ist im Grunde genommen ein Make Werkzeug das andere Make Werkzeuge (wie die, die wir ganz am Anfang installiert haben) benutzt um das Image zu bauen.

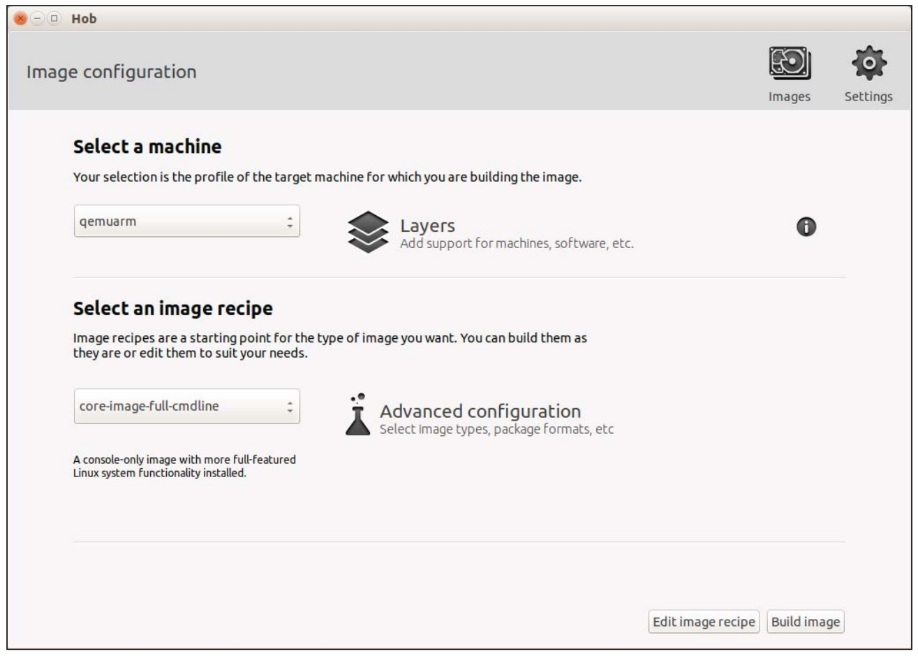
Wir haben uns entschieden zuerst ein einfaches Image zu bauen mit Hilfe vom core-image-minimal Rezept. Dieses baut ein einfaches und leeres Root Filesystem, das eigentlich zum Testen und entwickeln von Kernels und Bootloaders gedacht ist. Um dies zu bewerkstelligen haben wir die Kommandozeile „bitbake core-image-minimal“ benutzt.

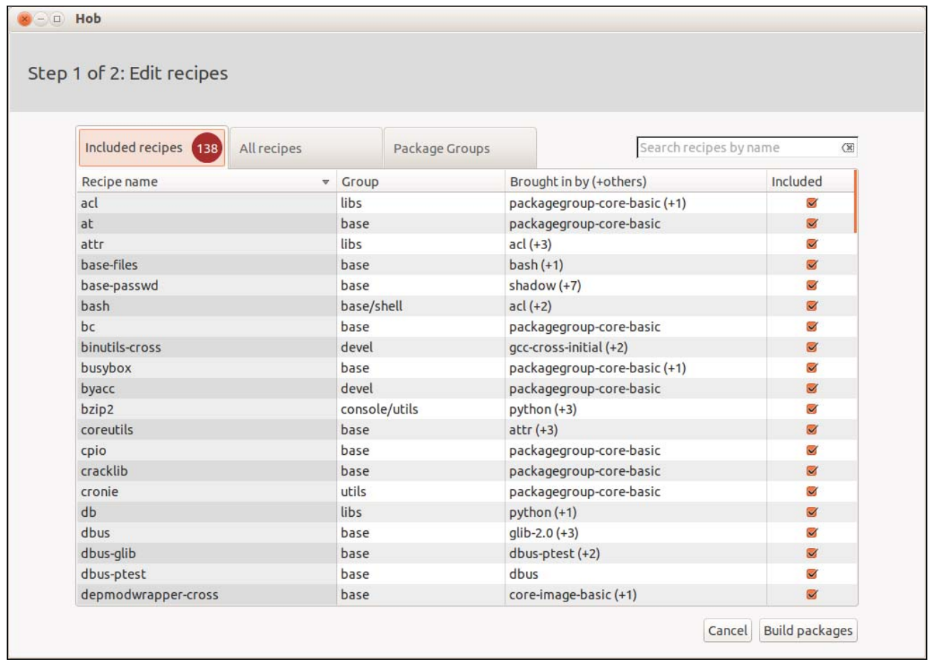
Da wir den BitBake allerdings auf einer virtuellen Maschine und nicht auf einem nativen Linux Betriebssystem haben laufen lassen hat er um die 12 Stunden gedauert (auf einem nativen Betriebssystem soll der BitBake zwischen 3 und 4 Stunden dauern).

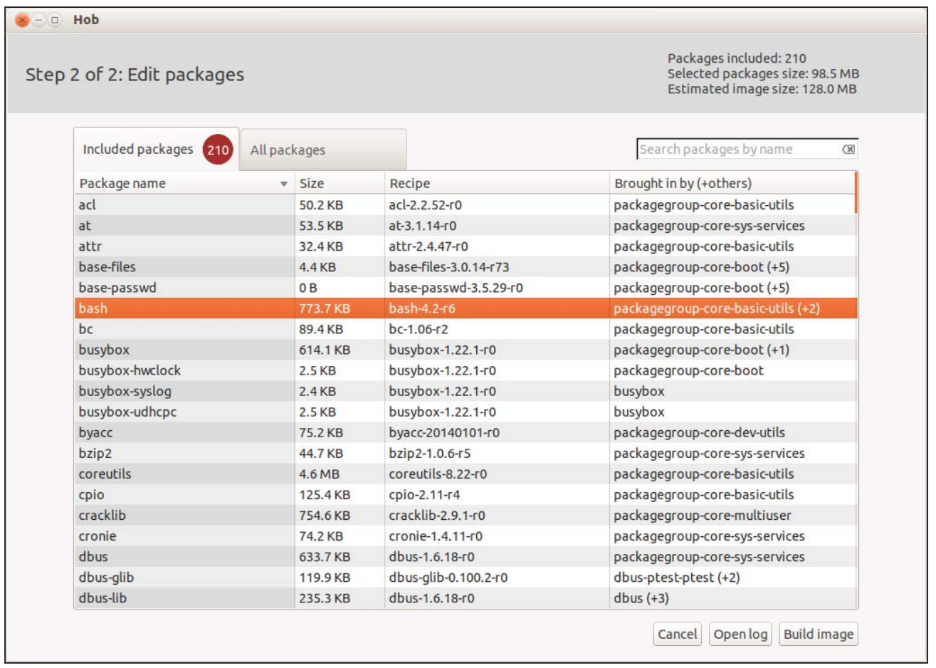
Als das Image gebaut war konnten wir es mit Hilfe von der Kommandozeile „runqemu qemux86“ emulieren.

# Erstellung eines Root Filesystems mit Hob:

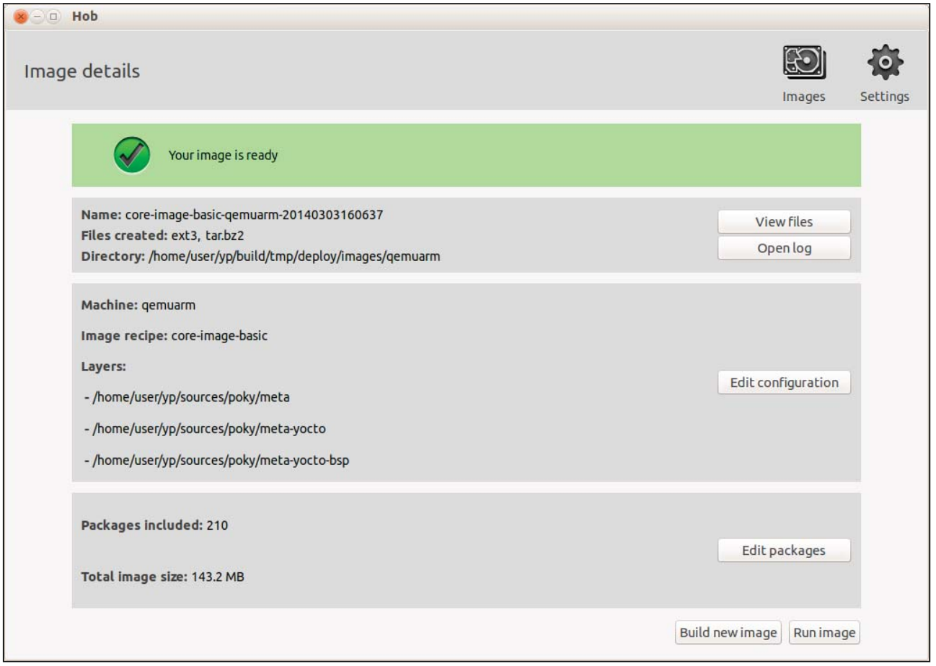
Poky bietet ein graphisches User Interface namens „Hob“ an das den BitBake Prozess übersichtlicher für den Benutzer macht:

Mit Hilfe von Hob kann man dann ganz einfach eine ziel Maschine und ein Rezept aussuchen.

Danach kann man sogar wählen welche Pakete das Image braucht und welche nicht.



Nach dem auswählen der Pakete kann der Build des Images beginnen.



Wenn der Build dann zu Ende ist kann das Image emuliert werden: