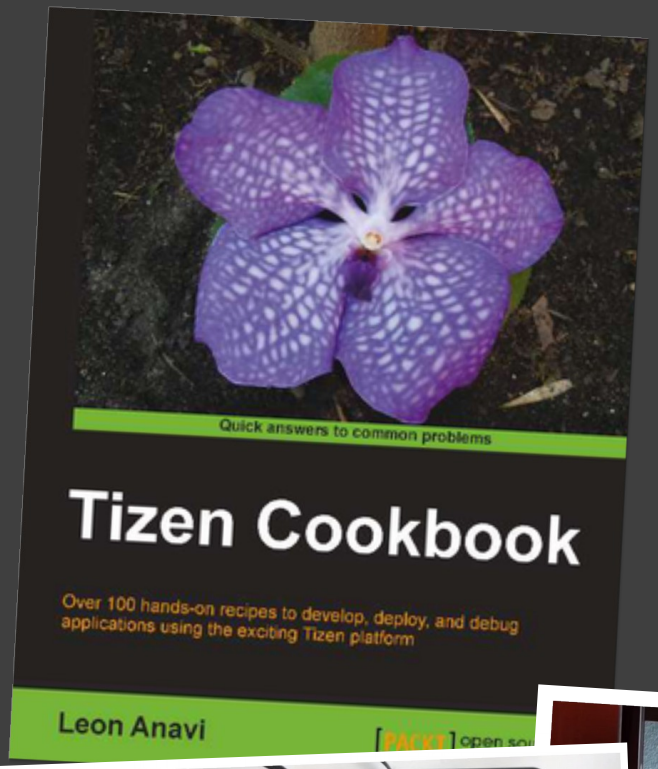
Как проектът Yocto помага за създаване на Internet of Things?

Леон Анави
@leonanavi
leon@anavi.org

За мен...



Contributions

May Jun Jul Aug Sep Oct Nov Dec Jan Feb Mar Apr

M

W

F

Summary of Pull Requests, issues opened, and commits. Learn more.

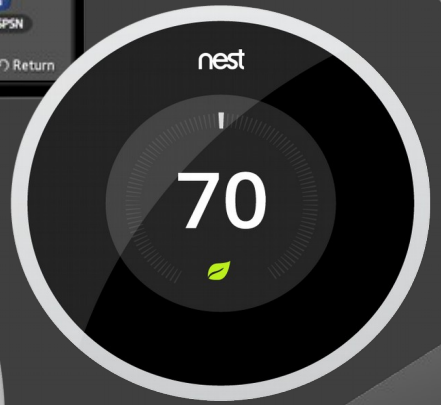
Less More

Contributions in the last year	Longest streak	Current streak
218 total	36 days	36 days
May 14, 2014 – May 14, 2015	April 9 – May 14	April 9 – May 14

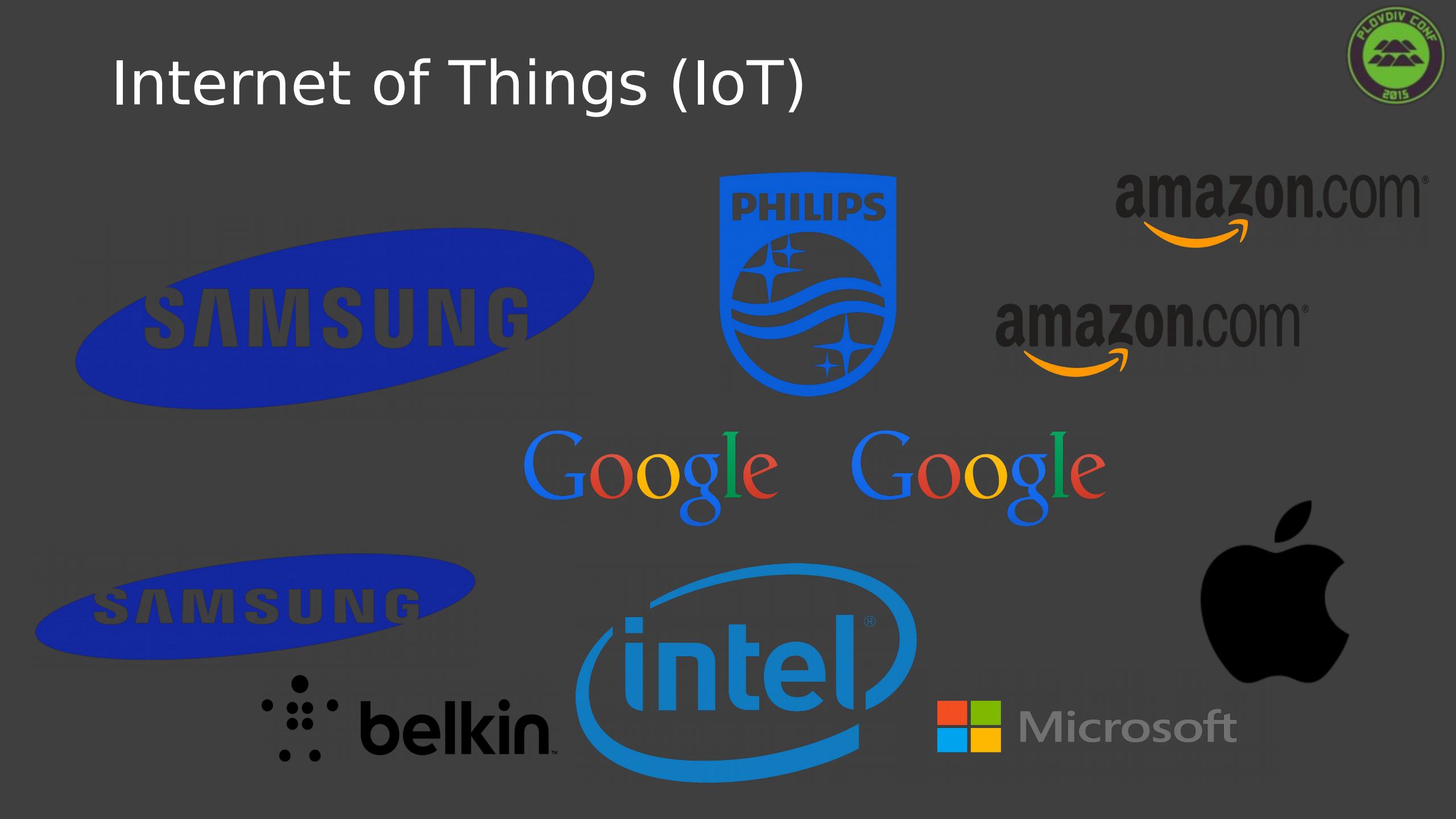
Internet of Things (IoT)

Направете устройството по-интелигентно и променете начина, по който го използват потребителите чрез Интернет!

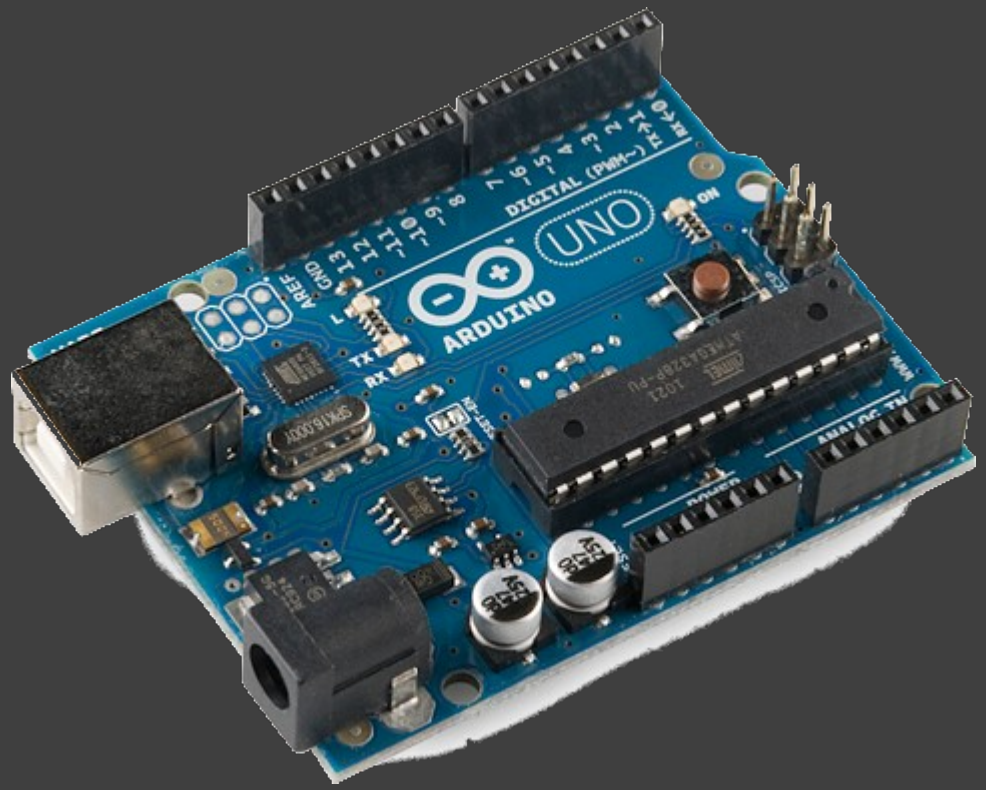
Internet of Things (IoT)



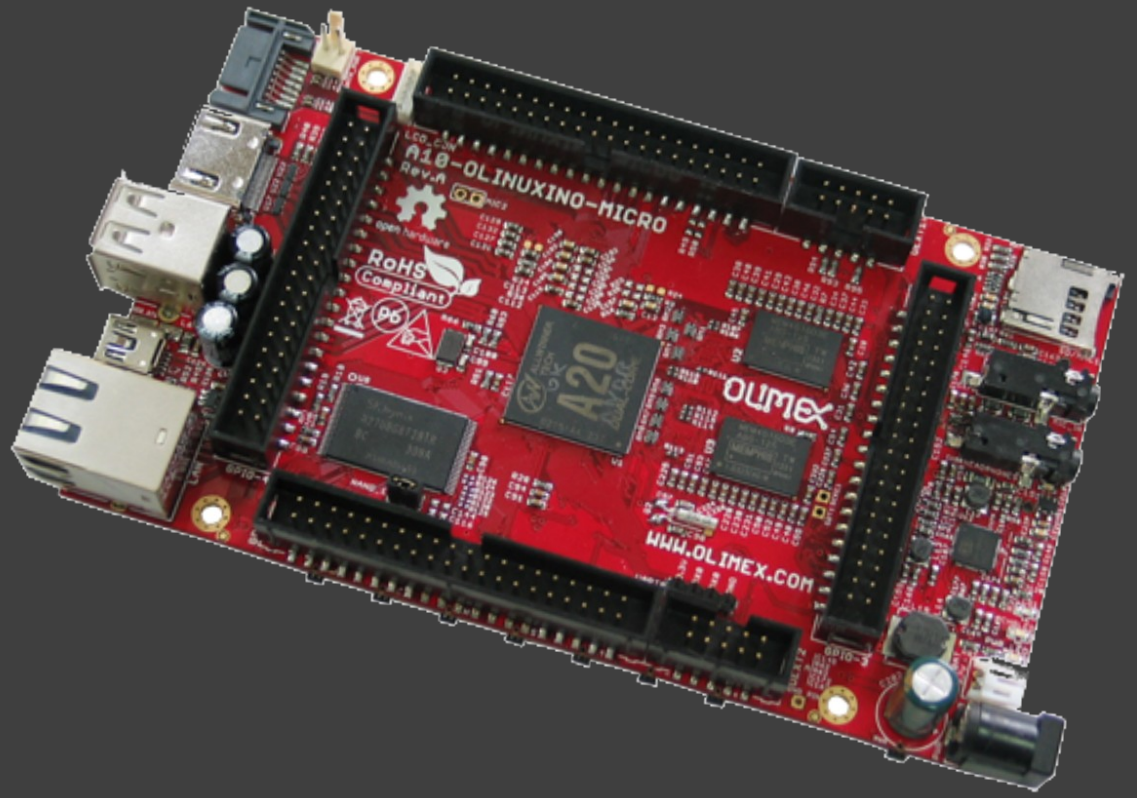
Internet of Things (IoT)



Микроконтролер (MCU)



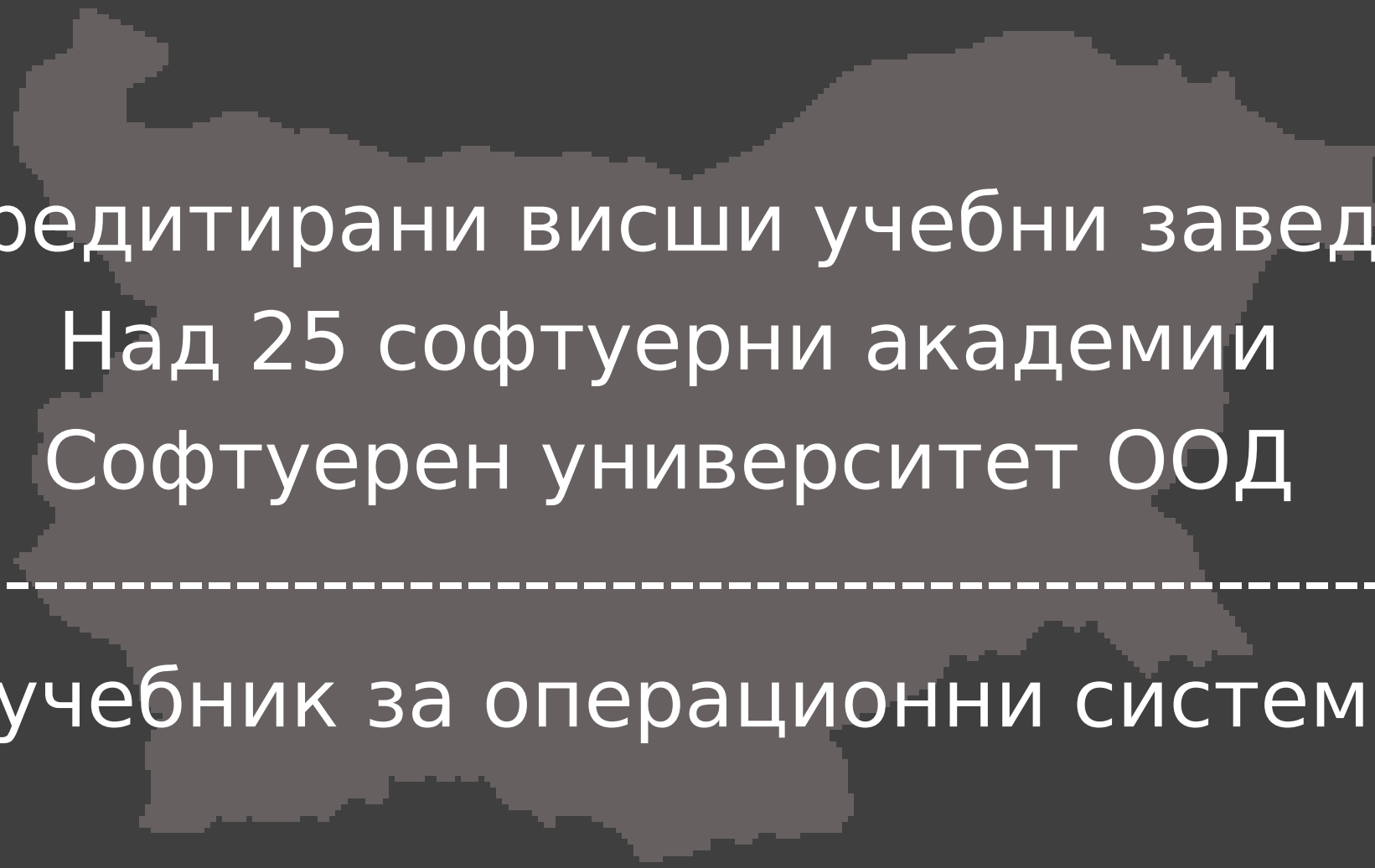
Single Board Computer (SBC)



SBC с ARM и Intel процесори

Операционна система (ОС) е...

.... много, много, много сложно нещо, изградено от много други сложни неща!



51 акредитирани висши учебни заведения

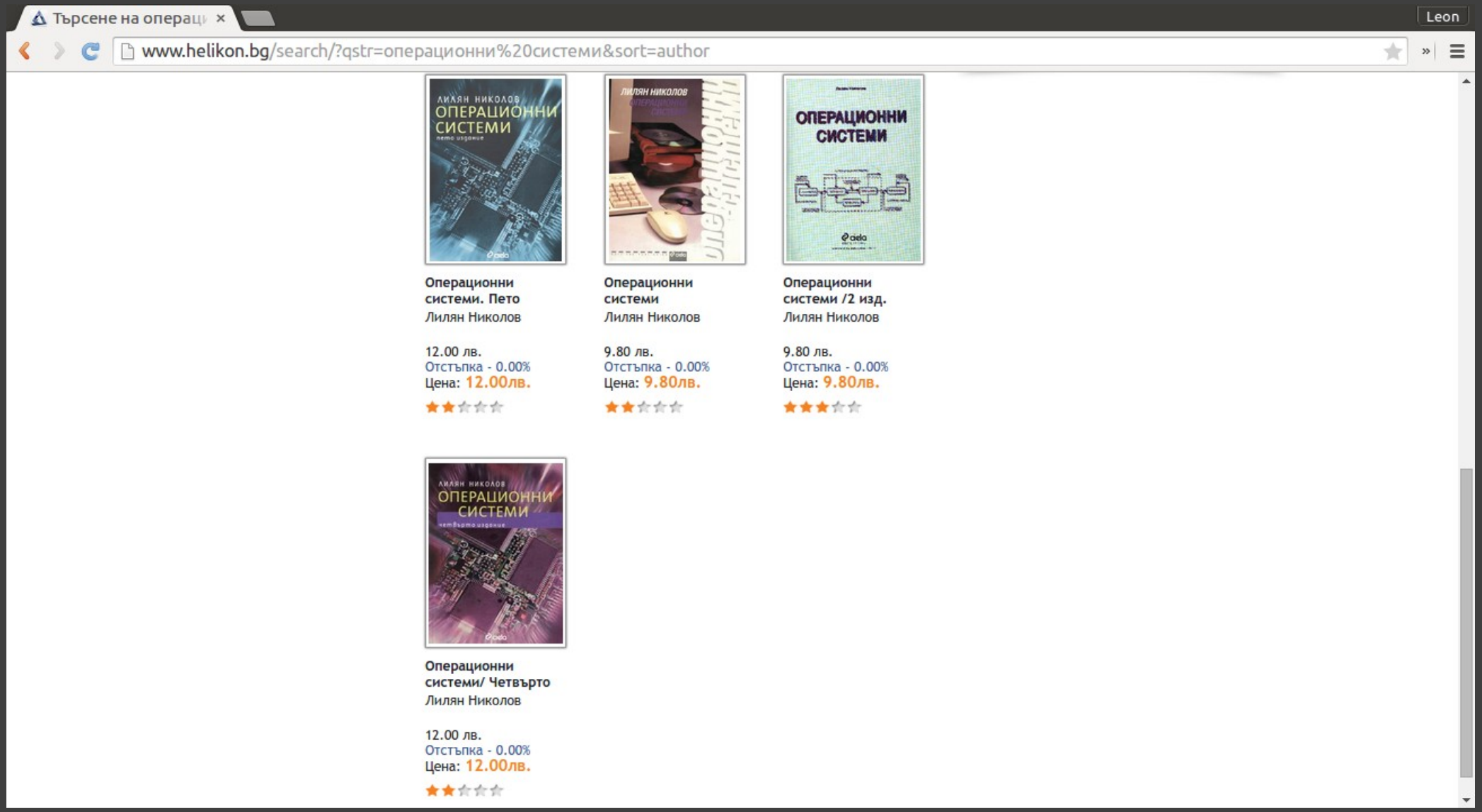
Над 25 софтуерни академии

Софтуерен университет ООД

---

1 учебник за операционни системи

Търсене в книжарница



Архитектура на GNU/Linux дистрибуция

Приложения

Програмни среди (frameworks)

Основни компоненти

Linux ядро и драйвери

Компоненти на GNU/Linux дистрибуция

X11

Wayland & Weston

GNU C Library

OpenSSL

Qt

EFL

GCC

SQLite

GNOME

KDE

Crosswalk

PulseAudio

Как да си направим и да поддържаме собствена GNU/Linux дистрибуция?

Poky

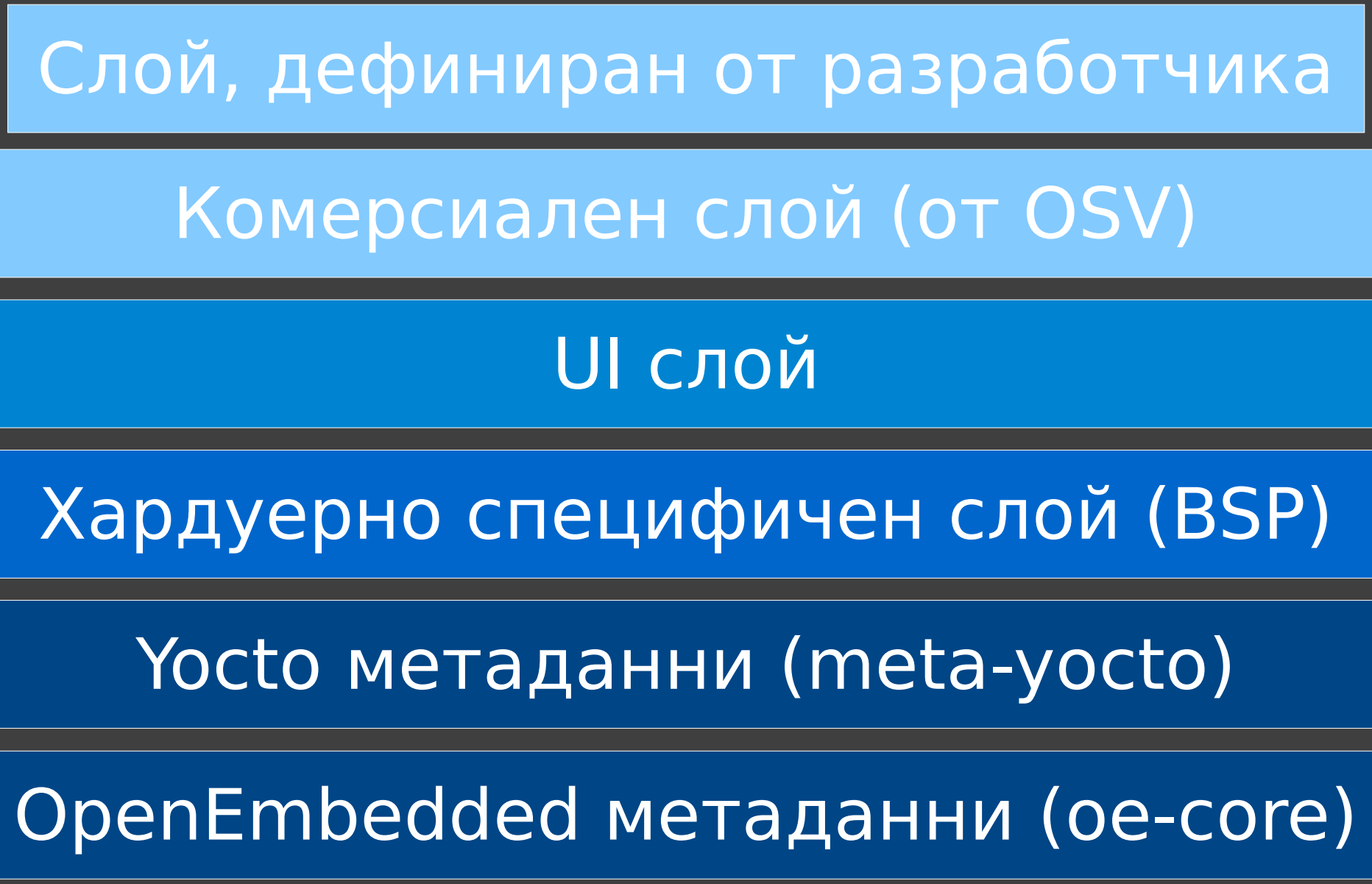
Poky = bitbake + метаданни

- Poky – система за изграждане на дистрибуция, използвана от Yocto Project
- Bitbake – инструмент за планиране и изпълнение на задачи
- метаданни – дефиниция на задачи

Метаданни

- **Конфигурации (.conf)** – глобално декларирани променливи
- **Класове (.bbclass)** – капсулирана логика за компилиране, пакетиране или друга задача, която може да се наследява
- **Рецепти (.bb)** – алгоритъм със стъпки за изпълнение при създаване на конкретен софтуер

Слоеве


Слой, дефиниран от разработчика
Комерсиален слой (от OSV)
UI слой
Хардуерно специфичен слой (BSP)
Yocto метаданни (meta-yocto)
OpenEmbedded метаданни (oe-core)

Рецепти

- Набор от инструкции за изпълнение на задачи, чрез които се създава софтуер
- Може да се наследяват класове (.bbclasses) и да включват файлове с общи дефиниции (.inc)
- Съдържат променливи, ключови думи, коментари и функции, написани на shell script или Python

Разширение на рецепти

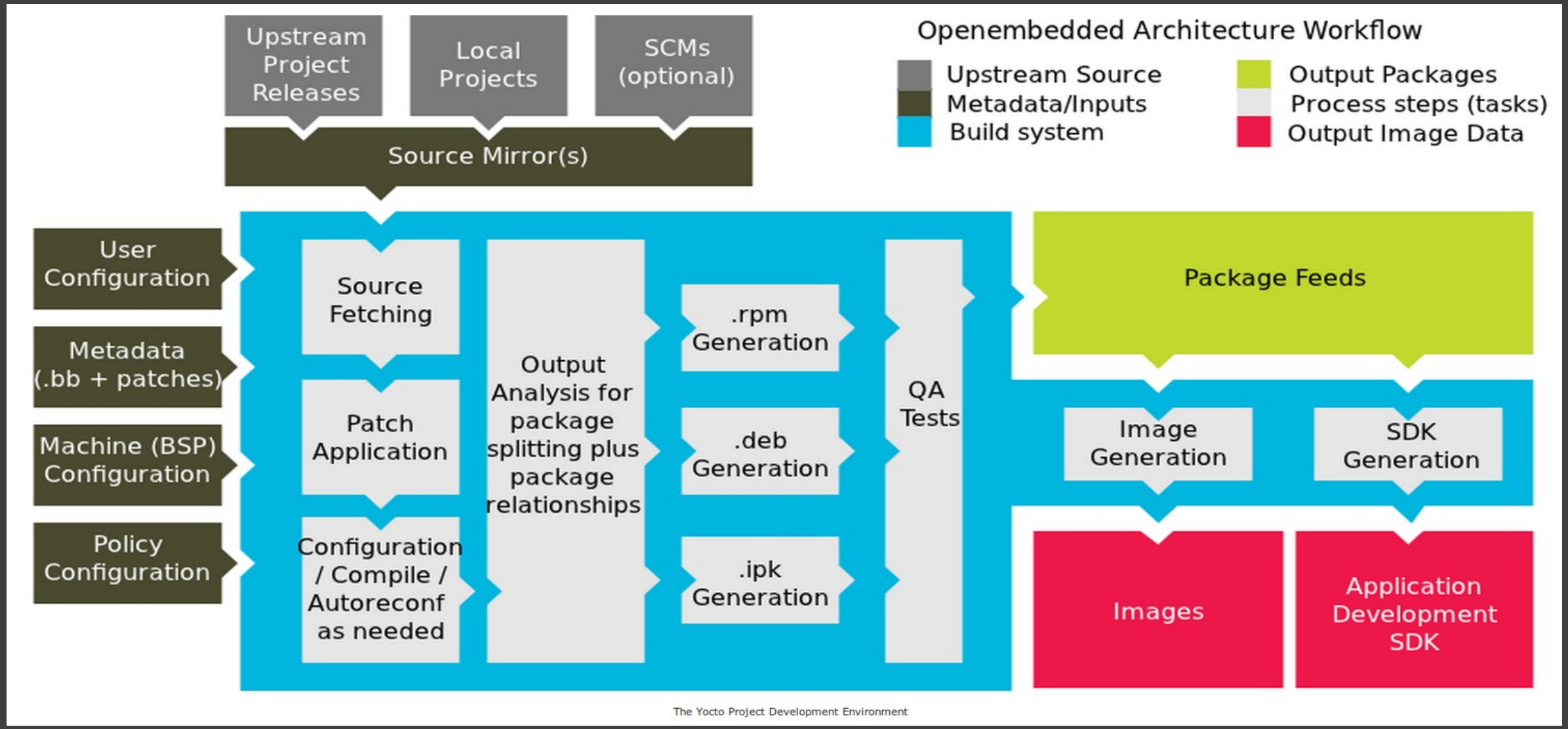
- Добавяне на допълнителни инструкции чрез .bbappend
- Създаване на patch (кръпка)

```
diff --git a/configure.ac b/configure.ac
index 05e883d..24f761f 100644
--- a/configure.ac
+++ b/configure.ac
@@ -156,7 +156,7 @@ fi
  AC_ARG_ENABLE(libinput-backend, [  --disable-libinput-backend],,
-            enable_libinput_backend=yes)
+            enable_libinput_backend=no)
```

- Включване на patch към рецепта

```
SRC_URI += "file://my.patch"
```

Как работи всичко това?



Да запретнем ръкави... трябват ни:

- Мощен компютър (препоръчителен минимален хардуер: Intel i7 CPU, 8GB RAM, поне 50GB свободно пространство и добро охлаждане)
- GNU/Linux дистрибуция, на която да пуснем Yocto
- SBC (при липса може да се ползва QEMU емулатор)

Как да ползваме Poky за ARM? (1/3)

- Сваляме Poky чрез Git:

```
git clone -b dizzy git://git.yoctoproject.org/poky.git
```

- Добавяме слоеве за ARM:

```
cd poky
git clone -b dizzy git://git.yoctoproject.org/meta-fsl-arm
git clone -b dizzy git://github.com/Freescale/meta-fsl-arm-extra.git
```

- Инициализираме средата в дир build:

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

Как да ползваме Poky за ARM? (2/3)

- Настройки в conf/local.conf

```
MACHINE ??= "cubox-i"
PARALLEL_MAKE ?= "-j 8"
BB_NUMBER_THREADS ?= "8"
```

- Слоеве в conf/bblayers.conf:

```
BBLAYERS ?= " \
  /home/leon/poky/meta \
  /home/leon/poky/meta-fsl-arm \
  /home/leon/poky/meta-fsl-arm-extra \
```

Как да ползваме Poky за ARM? (3/3)

- Стартираме bitbake:

```
bitbake core-image-minimal
```

- Чакаме, чакаме, пием чай, чакаме ...
- След успешно завършване готовите образи се намират в директория **tmp/deploy/images/cubox-i/**

Полезни bitbake опции

- Изчистване на рецепта

```
bitbake foo
```

- Повторно компилиране на рецепта

```
bitbake -c clean foo
```

- Показване на задачи към рецепта

```
bitbake -f -c compile foo
```

Подготовка на microSD карта

- Демонтирайте файловата система

```
sudo umount /dev/sdX
```

- Запишете файл върху картата с dd

```
sudo dd if=tmp/deploy/images/cubox-i/core-image-minimal-cubox-i.sdcard
of=/dev/sdX
sync
```

- Алтернативно, по-бързо решение за запис

```
sudo bmaptool copy –nobmap tmp/deploy/images/cubox-i/core-image-
minimal-cubox-i.sdcard /dev/sdX
```

Благодаря Ви, въпроси?