Qualcomm Developer Network Presents

Developing for Industrial IoT with Embedded Linux OS on DragonBoard™ 410c by Timesys University

Co-sponsored by Qualcomm Technologies, Inc. and Arrow Electronics

Session 1

Introduction to DragonBoard 410c and Starting Development of Your Embedded Linux based Industrial Internet of Things (IIoT) Device

Maciej Halasz, Vice President of Technology
Timesys Corporation

©2017 Timesys Corp.

Webinar Series

- **Session 1:** Introduction to DragonBoard 410 SoC and Starting Development of Your Embedded Linux based “Industrial Internet of Things” (IIoT) Device
  - Setup for designing IIoT products
  - How to assemble and deploy initial BSP
- **Session 2**: Application Development for Embedded Linux
  - Application development environment setup
  - How to reflect product requirements in the BSP
  - Communication in the IIoT system
- **Session 3**: Building a Cutting-Edge User Interface with Qt®
  - Developing modern, rich UIs for factory terminals
- **Session 4**: Embedded Products Security
  - Designing security-rich devices

©2017 Timesys Corp.

Session 1 — Agenda

- **Development Environment**
- **Deploying Yocto Project®/OpenEmbedded Linux BSP to your DragonBoard 410c**
- **Design considerations for IIoT products**
  - Requirements
- **Yocto Project/Open Embedded introduction**
  - Yocto Project – what is it
  - How to get and setup Yocto for the 96Boards™
  - Re-building BSP image from scratch
- **Software Development Kit**

©2017 Timesys Corp.

Build Environment

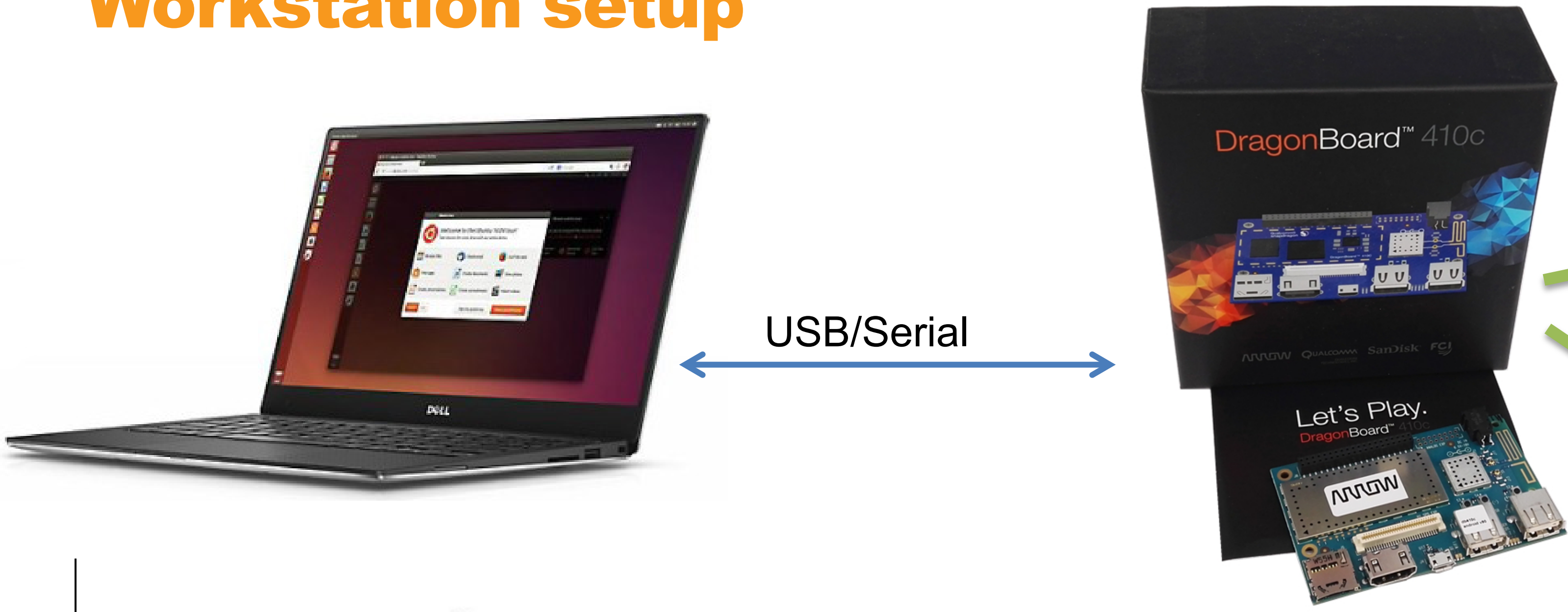
©2017 Timesys Corp.

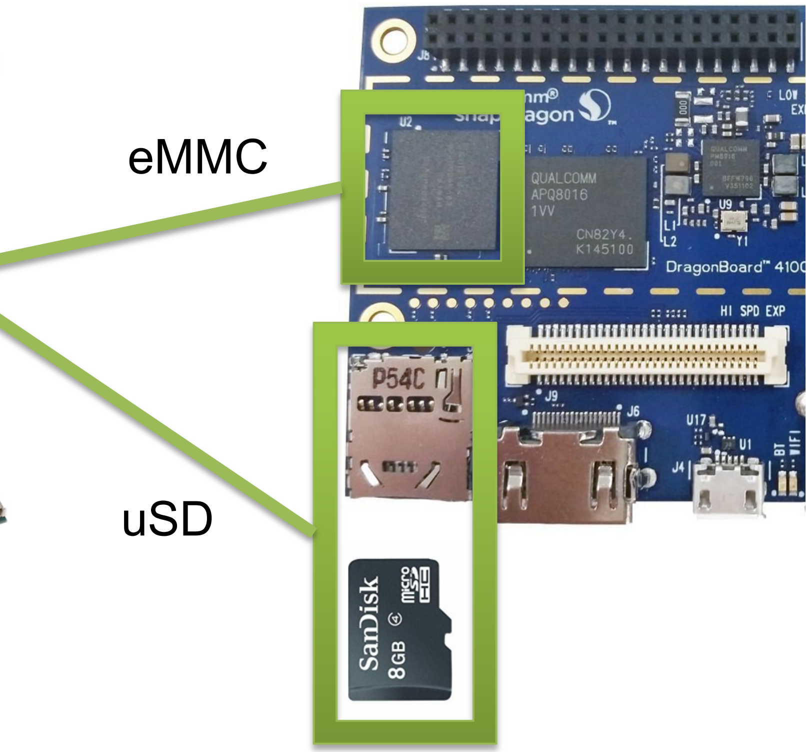
What we need to build a product

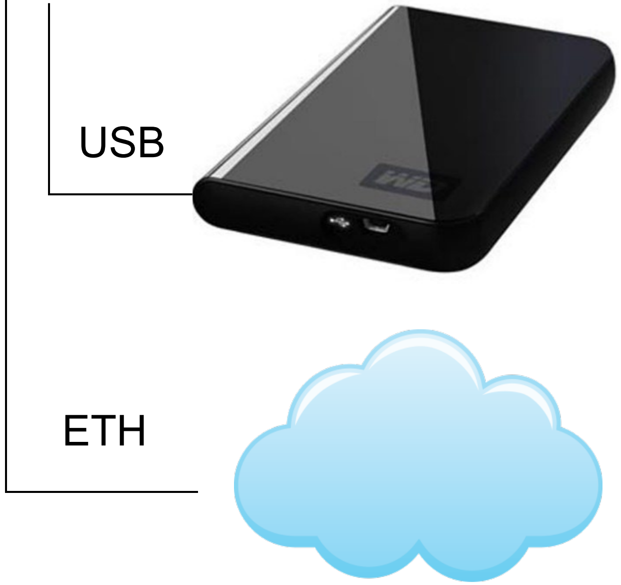
- A host machine
  - Linux is the best — any recent version is ok (Timesys® recommends Ubuntu® LTS)
  - Windows is fine, but you'll need a Windows® 10 native Linux support or a virtual machine with a Linux desktop OS installation (Timesys recommends Oracle® VirtualBox™)
- Cross-development environment / SDK
- Linux source code (BSP) for the product
  - Bootloader
  - Linux kernel
  - APIs
- Various Linux host utilities needed by Yocto/OpenEmbedded Linux BSP
- IDE for application and system level development. WYSIWYG IDE for faster UI development
- A hardware development kit

©2017 Timesys Corp.

Workstation setup







ubuntu

96Boards

Embedded Linux from a Trusted Source
LinuxLink by timesys

- Utilities
- Yocto Project-Core
- Metalayer from Qualcomm Technologies, Inc.
- Timesys metalayer
- 3rd party metalayers
- Development Tools
- Support

©2017 Timesys Corp.

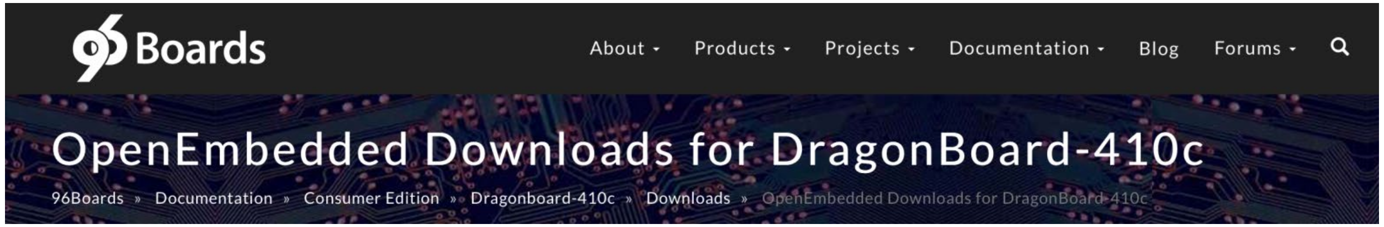
Initial Board Setup

©2017 Timesys Corp.

Where to find Yocto/OpenEmbedded Linux BSP binaries for deployment

- **Files representing reference software for the DragonBoard 410c can be downloaded from the 96Boards website:**

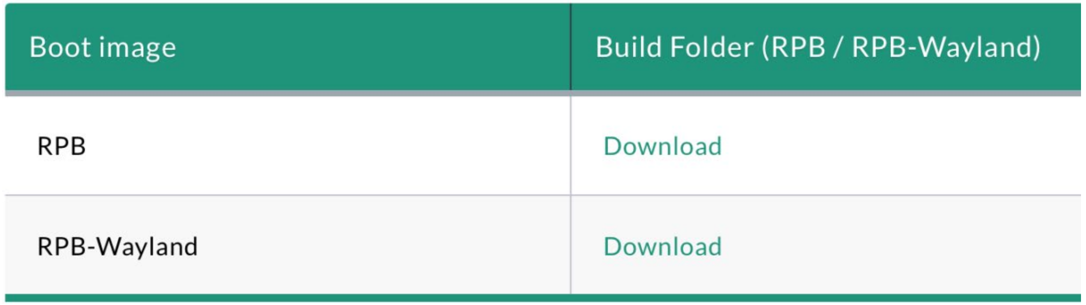
  http://builds.96boards.org/releases/dragonboard410c/linaro/openembedded/latest/rpb/



OpenEmbedded Downloads for DragonBoard-410c

96Boards » Documentation » Consumer Edition » Dragonboard-410c » Downloads » OpenEmbedded Downloads for DragonBoard-410c

- **Linux images**
  - Debian®
  - OpenEmbedded (Yocto)
- **Yocto images**
  - Split into deployable components
    - Root filesystem
    - Bootloader
    - Linux kernel



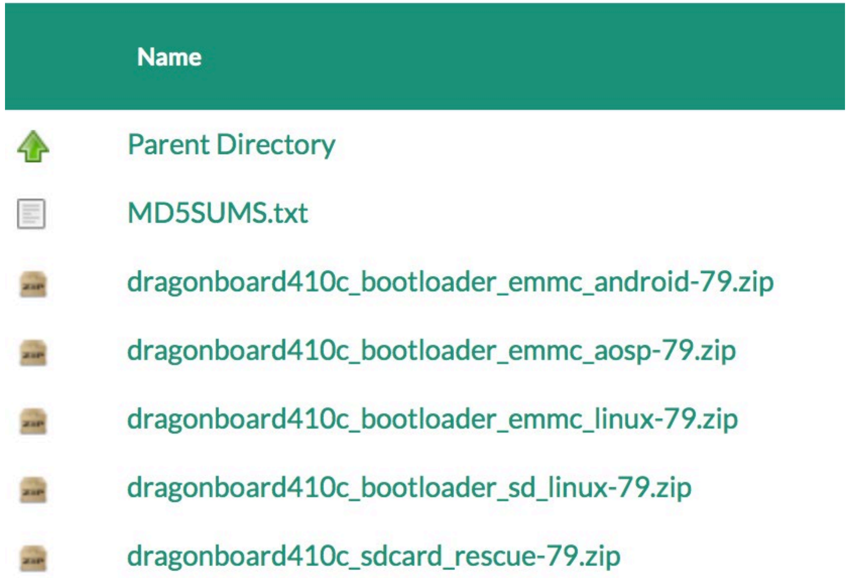
Boot image	Build Folder (RPB / RPB-Wayland)
RPB	Download
RPB-Wayland	Download

©2017 Timesys Corp.

Deploying Linux images to the DragonBoard 410c

- **DragonBoard 410c supports two methods of software deployment**
  1. SD card (complete software image/rescue image)
  2. USB connector (individual software images)
- **Second method is intended for customizations and frequent updates**
- **Rescue images are provided at**

  http://builds.96boards.org/releases/dragonboard410c/linaro/rescue/latest/



Name
Parent Directory
MD5SUMS.txt
dragonboard410c_bootloader_emmc_android-79.zip
dragonboard410c_bootloader_emmc_aosp-79.zip
dragonboard410c_bootloader_emmc_linux-79.zip
dragonboard410c_bootloader_sd_linux-79.zip
dragonboard410c_sdcard_rescue-79.zip

- **Board rescue process:**
  - Step 1: Download and unzip SD card Install Image
  - Step 2: Find SD card device name
  - Step 3: Install Rescue Image onto SD card
  - Step 4: Boot DragonBoard 410c from an SD card
  - Step 6: Flash Yocto/OpenEmbedded Linux BSP images into EMMC™

©2017 Timesys Corp.

Image Deployment — SD card (1)

Helper for the flashing process:

- **Step 1:** From the URL provided on the previous slide download the rescue image for the DragonBoard 410c. The image to be downloaded:

  dragonboard410c_sdcard_rescue-79.zip

- Uncompress the archive with the command:

```
$ unzip dragonboard410c_sdcard_rescue-79.zip
```

- **Step 2:** Find microSD card device name on your host. Run the following command in a terminal window:

```
$ lsblk
```

→ **Note**: Run this command before and after inserting the microSD card into host PC microSD card reader

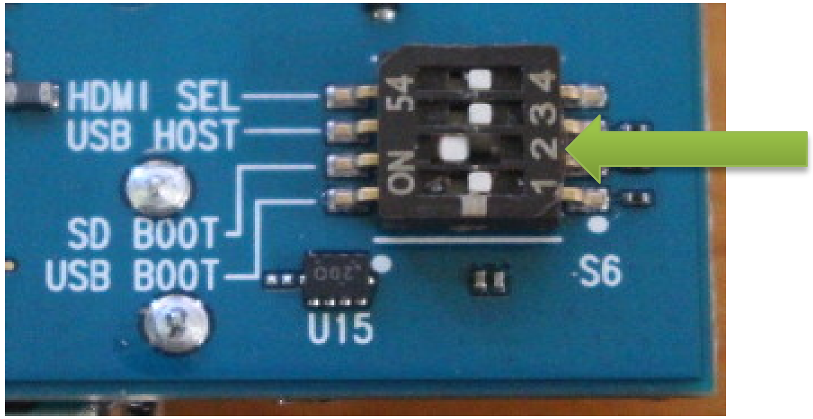
- **Step 3:** Install rescue image onto microSD card. Use the following command to flash image to microSD card

```
$ sudo dd if=db410c_sd_rescue.img of=/dev/XXX bs=4M oflag=sync status=noxfer
```

©2017 Timesys Corp.

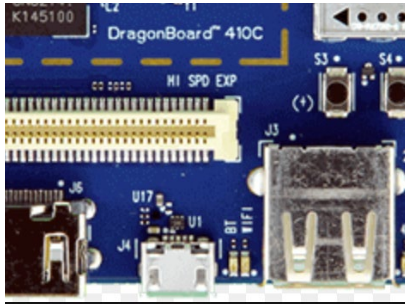
Image Deployment — SD card (2)

- **Helper for the flashing process:**
  - **Step 4:** Booting DragonBoard 410c with SD card
    - Make sure DragonBoard 410c is unplugged from power
    - Set S6 switch on DragonBoard 410c to **0-1-0-0**, “SD Boot switch” should be set to “ON”.
    - Insert the microSD card into the DragonBoard 410c
    - Connect DragonBoard 410c microUSB port to PC’s USB Host with a cable
    - Plug power adaptor into DragonBoard 410c, wait 5s for board to boot up. **Note:** No onboard LED will light up in this process!
    - We have set the DragonBoard in the “**fastboot mode**”



©2017 Timesys Corp.

Image Deployment — SD card (3)



- **Helper for the flashing process:**
  - **Step 5:** Flash Yocto/OpenEmbedded Linux BSP images
  - In the fastboot mode, DragonBoard can accept images sent from the Host PC via USB OTG port.
    - On the Host PC, check for connected fastboot devices with the command:

```
$ sudo fastboot devices
```

- Create partitions on the EMMC and Flash the bootloader. Use flashall script provided

```
$ sudo ./flashall
```

- Boot and filesystem images can be found in: http://builds.96boards.org/releases/dragonboard410c/linaro/openembedded/latest/rpb/
- Flash the runtime files with the command

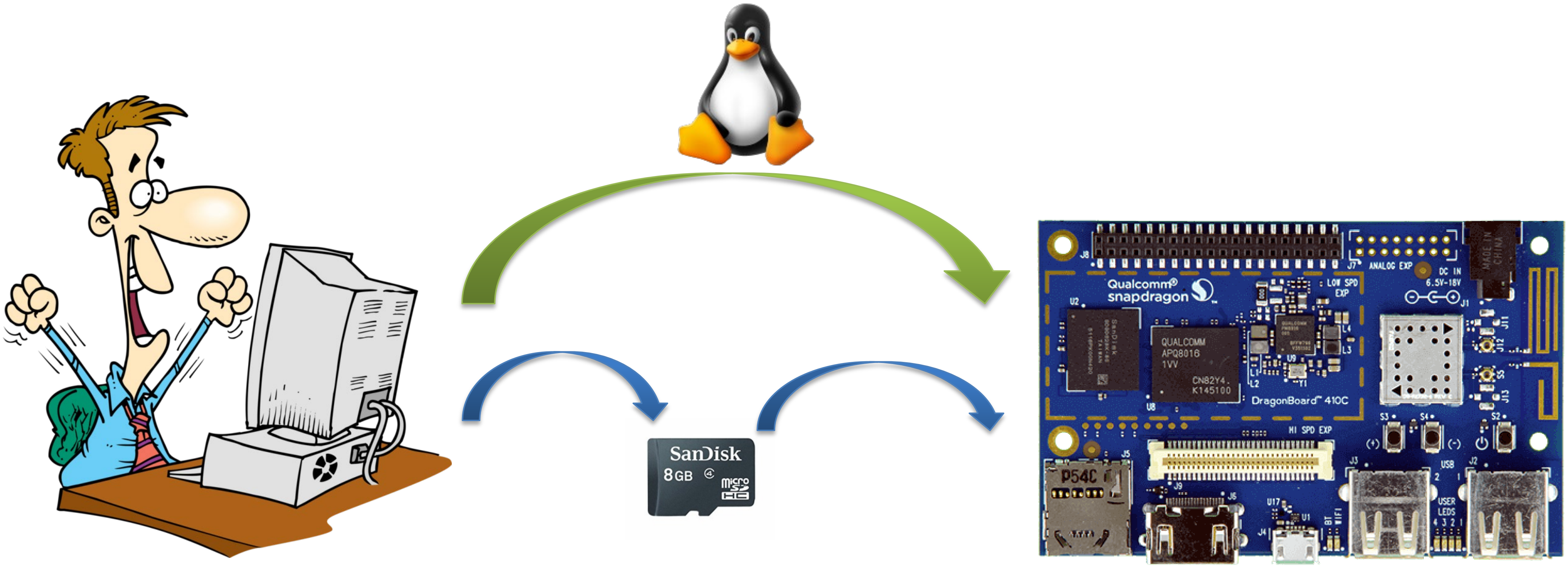
```
$ sudo fastboot flash boot boot--4.9-r0-dragonboard-410c-20170803175648-18-18.img
$ sudo fastboot flash rootfs rpb-console-image-dragonboard-410c-20170803175648-18.rootfs.ext4
```

- Remove the SD Card
- Set S6 switch on DragonBoard 410c back to 0-0-0-0, all switches set to “OFF”
- Power cycle the DragonBoard 410c to reboot.

©2017 Timesys Corp.

LAB 1 — Yocto/OpenEmbedded Linux BSP Deployment

- Deploy Yocto Linux BSP images to the DragonBoard 410c using board recovery method
- Boot the board to verify proper image flashing into eMMC



©2017 Timesys Corp.

Developing Industrial IoT Devices

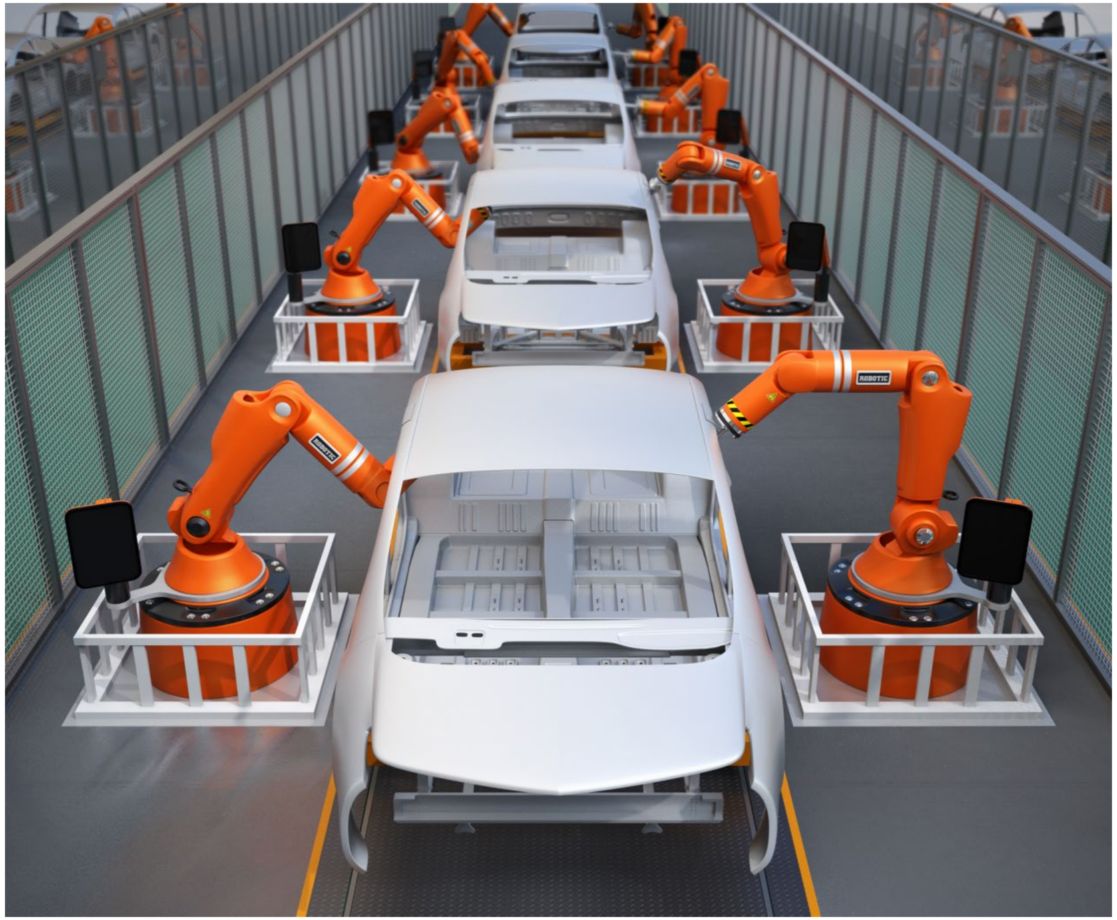
©2017 Timesys Corp.

Industrial IoT has many requirements

©2017 Timesys Corp.

Why IoT is important in Industrial setting

- **Industrial plants**
  - Operate at capacity
  - Operate at efficiency
- **Challenges**
  - Predict undesired process conditions
  - Minimize equipment failures
- **Industrial IoT provides**
  - Optimal business performance
  - Improved process reliability
    - Capturing and analyzing data
    - Identify warnings and potential issues
    - Preemptive service of equipment



©2017 Timesys Corp.

Example of an IIoT system

- **Data collection end-point**
  - Sensors
  - Local connectivity
  - Actuators
  - Displays
  - Autonomous decision logic
- **Data concentrator**
  - Sensors
  - Local connectivity
  - Resources for local data processing
  - Cloud connection infrastructure
- **Analytics**
  - Integrate multiple production processes
  - Can be company wide

©2017 Timesys Corp.

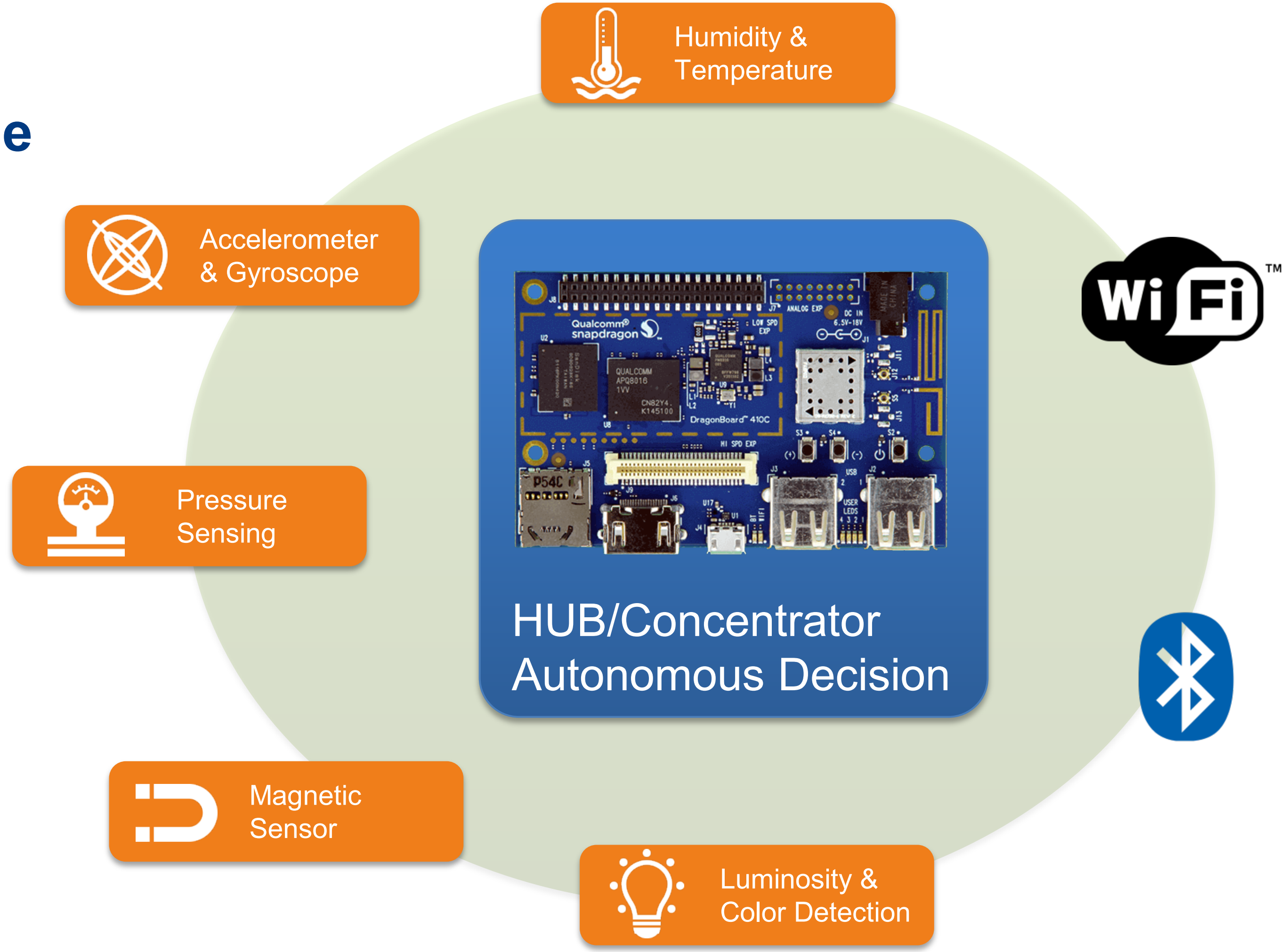
Why do companies care about IIoT?

- **Increase asset utilization**
  - Reduce unplanned downtime
  - Predict failures
  - Pro-active response
- **Increase operating efficiency**
  - Reduction in energy usage
  - Increased engineering effectiveness by monitoring
  - Integrated decision support mechanism
- **Increase in Safety**
  - Minimize risks by ensuring stable operations
  - No production interruptions due to safety check
- **Reduce maintenance cost**
  - Optimize maintenance based on real asset conditions
  - Pre-emptive addressing of issues

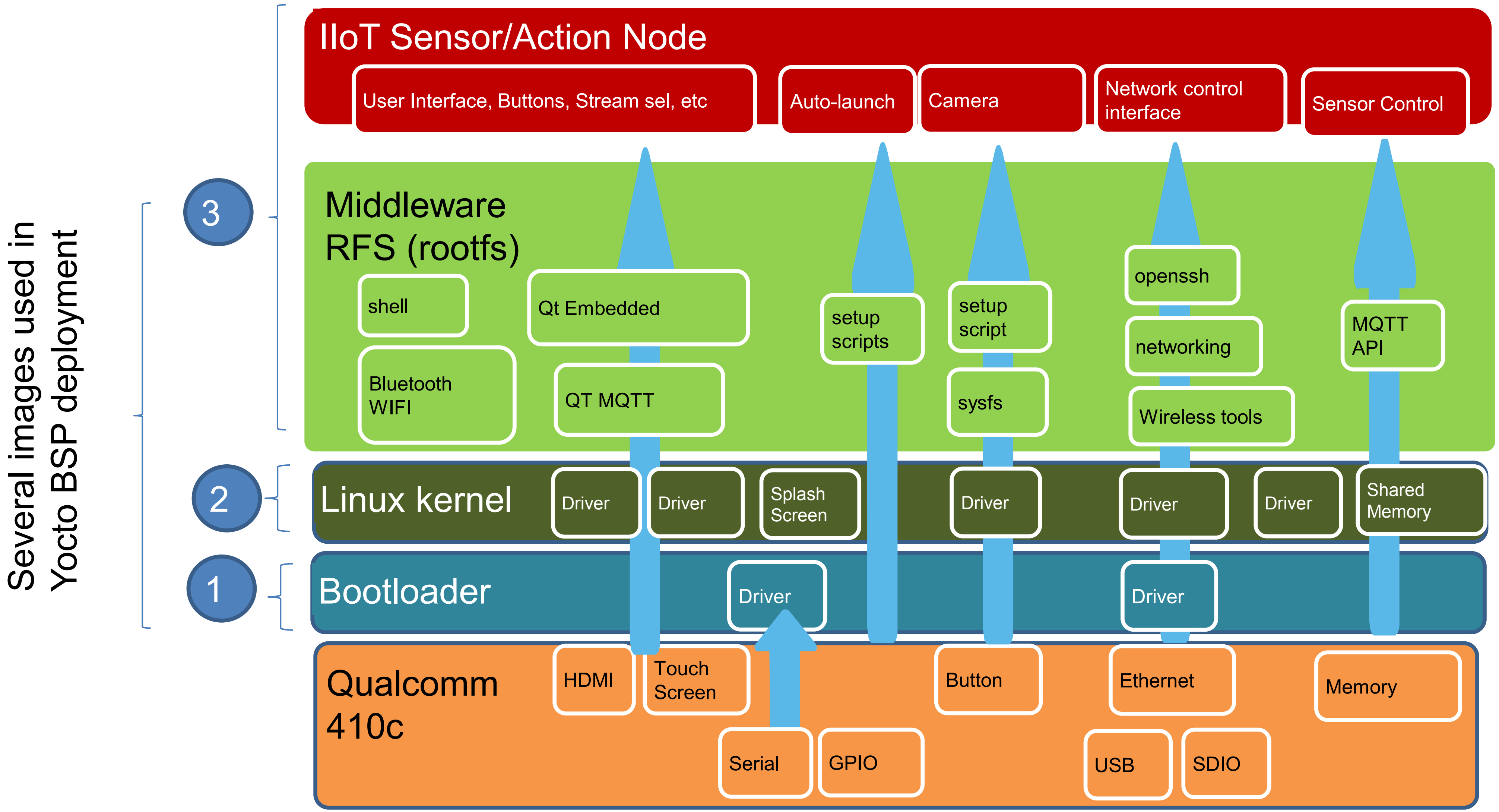
©2017 Timesys Corp.

IIoT Point requirements

- Requirements are Industry Process specific
- Topography of IIoT can vary
- Requirements typically include
  - Connectivity:
    - Bluetooth®
    - Ethernet ®
    - WiFi™
    - BUS
  - Sensors:
    - Discrete
    - Continuous
  - Sensor examples
    - Temperature
    - Pressure
    - Movement
    - ON/OFF


Humidity & Temperature
Accelerometer & Gyroscope
Pressure Sensing
Magnetic Sensor
Luminosity & Color Detection
HUB/Concentrator Autonomous Decision
Wi Fi™

Software for IIoT device — baseline

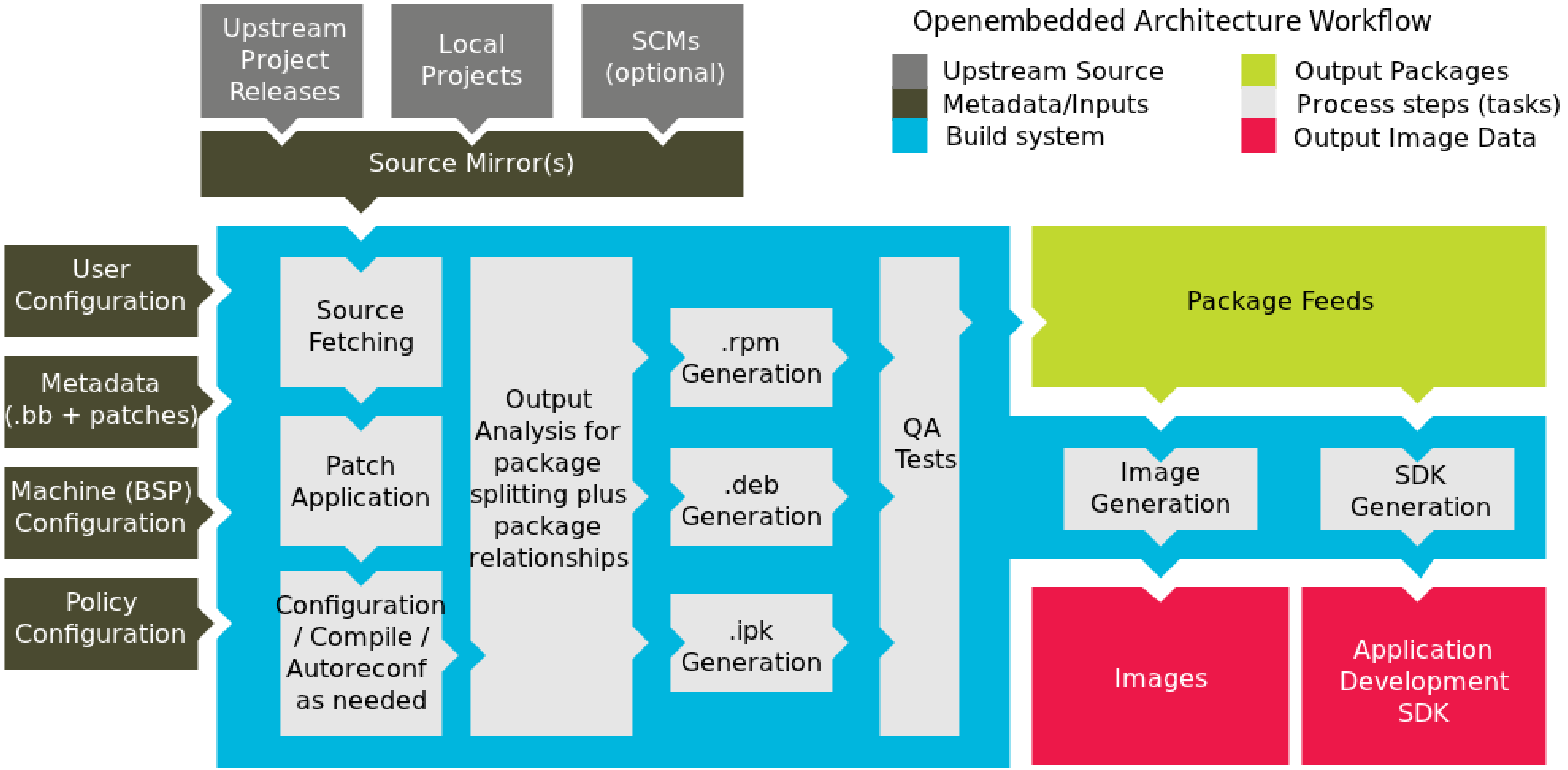


©2017 Timesys Corp.

Yocto Project

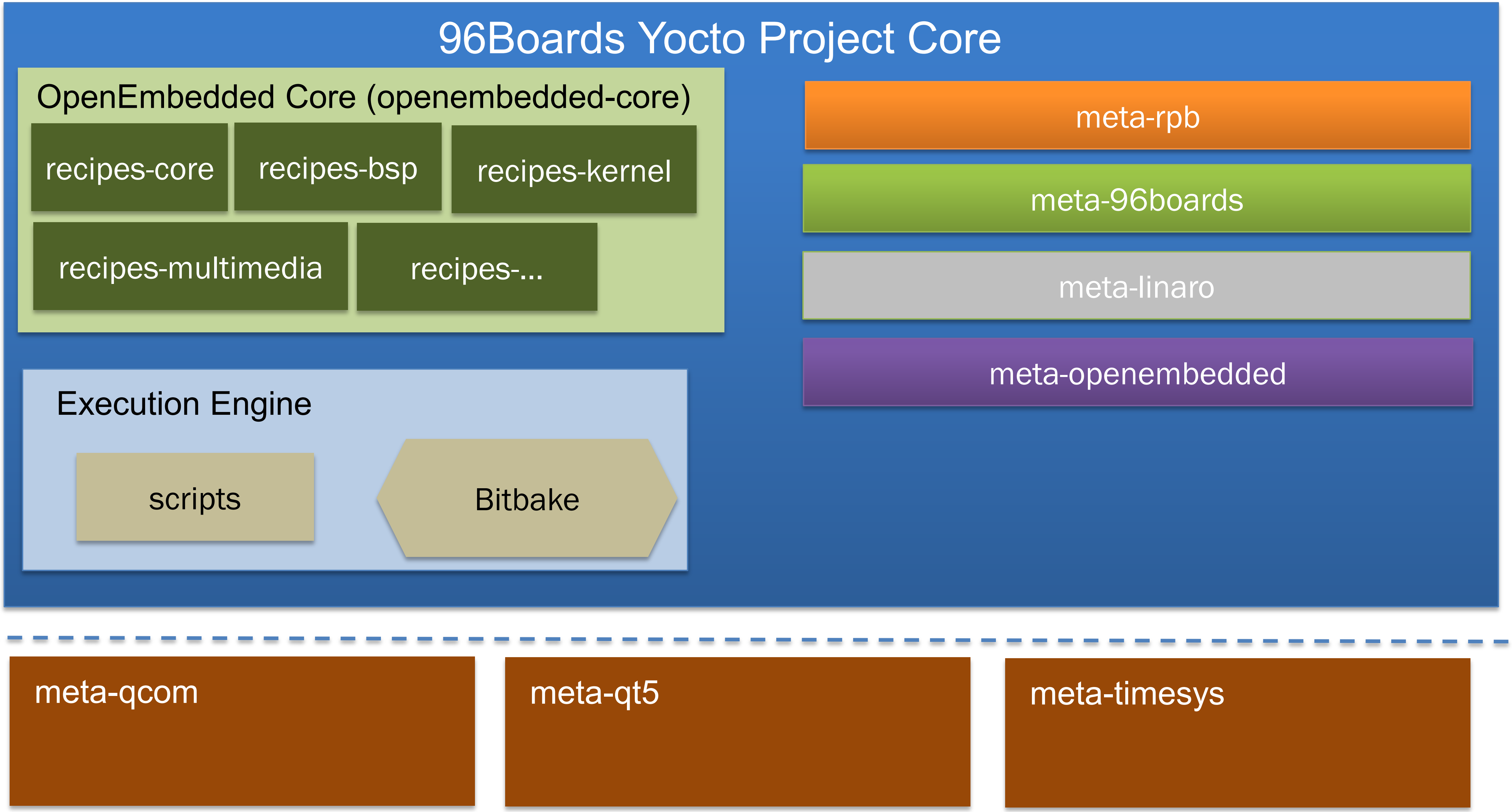
©2017 Timesys Corp.

Yocto/Open Embedded build process



©2017 Timesys Corp.

96Boards Yocto structure



96Boards Yocto Project Core

OpenEmbedded Core (openembedded-core)

recipes-core

recipes-bsp

recipes-kernel

recipes-multimedia

recipes-...

Execution Engine

scripts

Bitbake

meta-rpb

meta-96boards

meta-linaro

meta-openembedded

meta-qcom

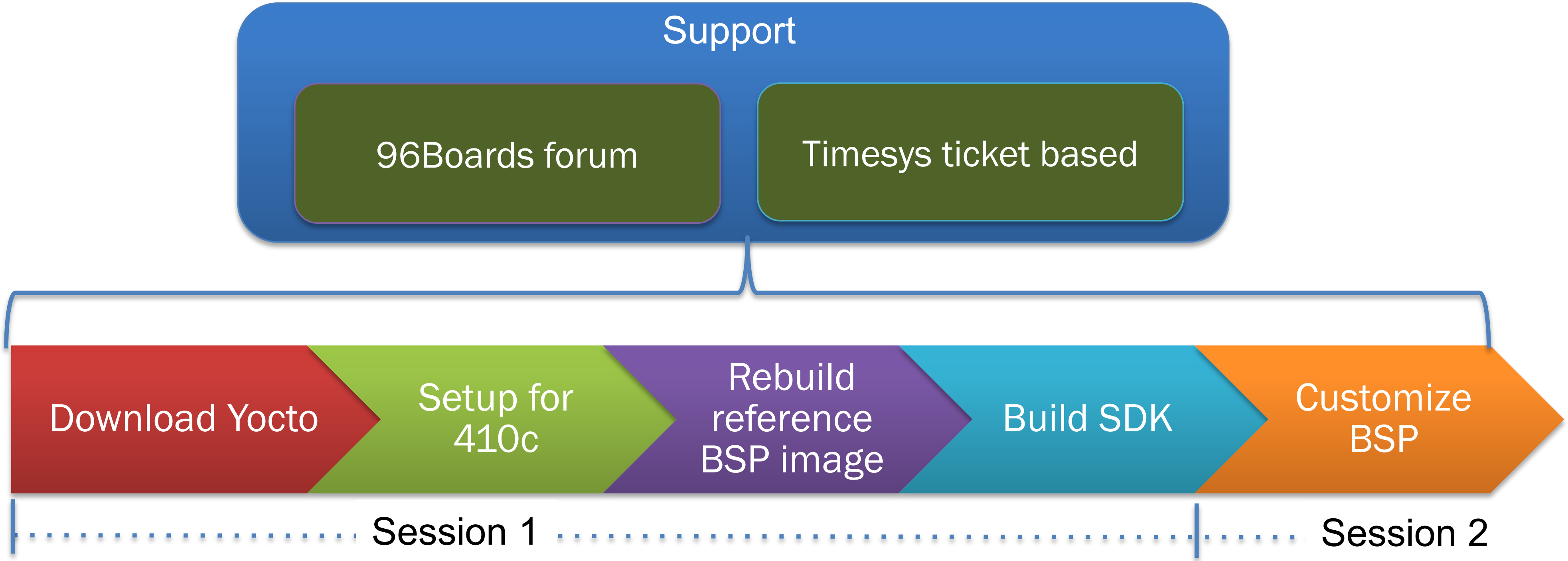
meta-qt5

meta-timesys

©2017 Timesys Corp.

Building and Customizing Linux BSP

- **Yocto based desktop tools**



- **BSP Customizations**
  - Reflecting IIoT specific device requirements

©2017 Timesys Corp.

1

Initial Yocto Project Setup

©2017 Timesys Corp.

Host Environment

- Depending on the Host System used, make sure that you have installed all required packages to run Yocto
- Ubuntu/Debian

```
$ sudo apt-get install gawk wget git-core diffstat unzip texinfo \
build-essential chrpath libsdl1.2-dev xterm curl
```

- CentOS™

```
$ sudo yum install gawk make wget tar bzip2 gzip python unzip perl \
patch diffutils diffstat git cpp gcc gcc-c++ glibc-devel texinfo chrpath \
socat SDL-devel xterm
```

- For other host OS

  http://www.yoctoproject.org/docs/current/yocto-project-qs/yocto-project-qs.html

©2017 Timesys Corp.

Repo tool and GIT™ setup

- **Provides a unified command to download software from multiple sources**
- **Used by Android™ and leveraged by 96Boards**
- **Install the tool from Google®**

```
$ mkdir ~/bin
$ curl http://commondatastorage.googleapis.com/git-repo-downloads/repo > ~/bin/repo
$ chmod a+x ~/bin/repo
```

- **GIT setup**

```
$ git config --global user.name "Your Name"
$ git config --global user.email "Your Email"
$ git config --list
```

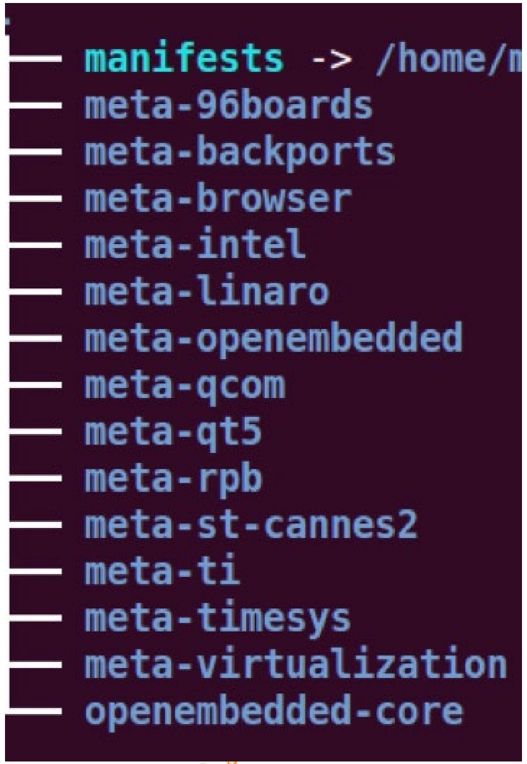
©2017 Timesys Corp.

Getting Yocto

- **Yocto for the 96Boards can be assembled from two sources:**
  1. github.org
  2. yoctoproject.org
- **Getting source code from GitHub™:**

```
$ mkdir 96boards-rpb
$ cd 96boards-rpb
$ repo init -u https://github.com/96boards/oe-rpb-manifest.git -b morty
$ repo sync
```

- **Core Yocto and additional metalayers are installed**
  - meta-96boards
    - – Holds definitions for 96Boards
  - meta-qcom
    - – Provides definition for boards including DragonBoards
  - meta-timesys



```
├── manifests -> /home/m
├── meta-96boards
├── meta-backports
├── meta-browser
├── meta-intel
├── meta-linaro
├── meta-openembedded
├── meta-qcom
├── meta-qt5
├── meta-rpb
├── meta-st-cannes2
├── meta-ti
├── meta-timesys
├── meta-virtualization
├── openembedded-core
```

©2017 Timesys Corp.

meta-timesys

- **Provides additional features from Timesys which can be applied to any Yocto Project**
  - Helps share information about YOUR current Yocto configuration with Timesys engineers and support teams
    - Facilitates investigative work on YOUR specific questions
  - Provides proactive update service
    - Find out what are the new updates available from open source that are relevant to the product
  - Provides security feed that is relevant to your board and your BSP/Yocto configuration
    - You decide when an update should be applied
    - You get an on-demand list of vulnerabilities that affect your product
  - Available from GitHub

    https://github.com/timesysgit/meta-timesys

©2017 Timesys Corp.

2

Configuring Yocto for DragonBoard 410c

©2017 Timesys Corp.

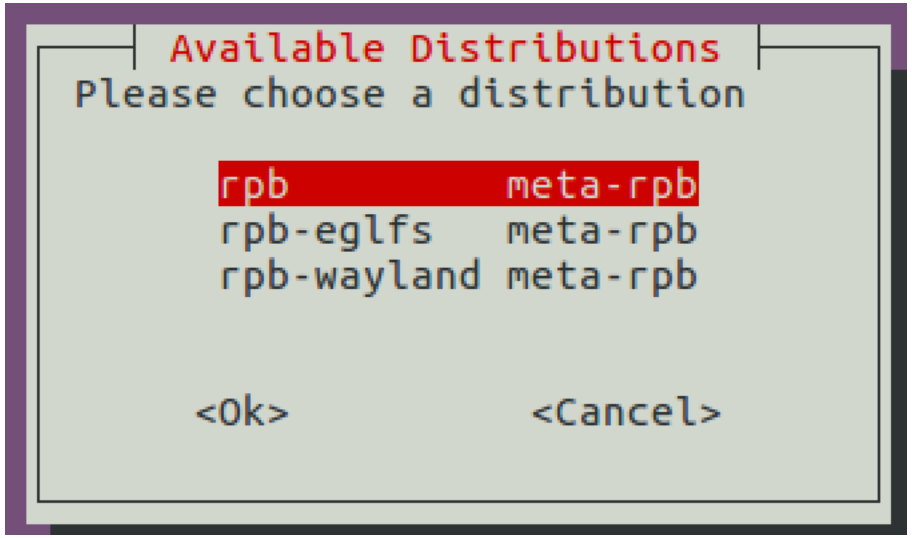
LAB 2: Yocto configuration

Running setup-environment script starts Yocto configuration wizard

- Command to configure and setup Yocto:

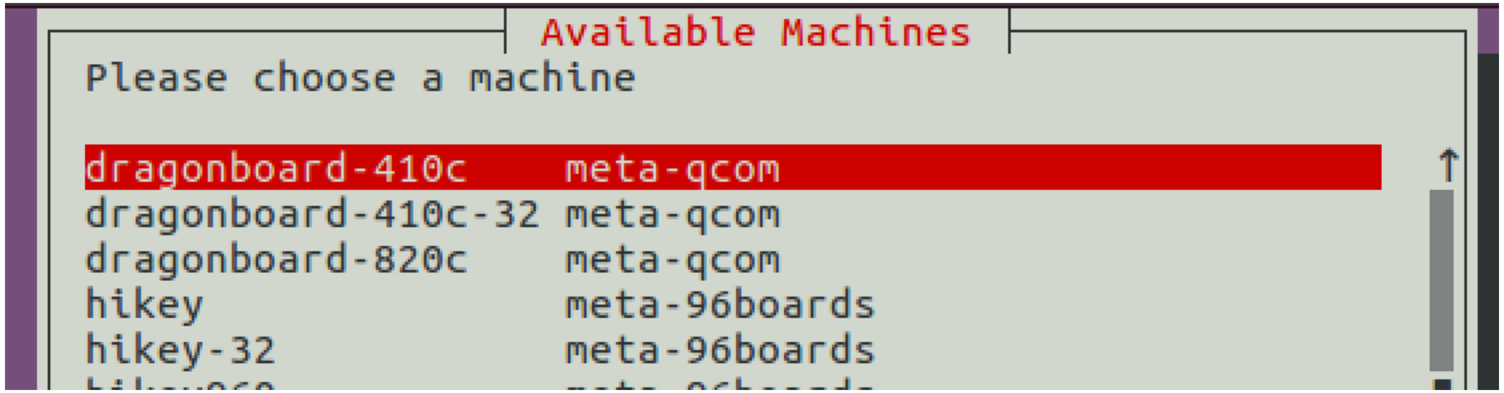
```
$ source setup-environment
```

- In the setup wizard you can choose:
  - MACHINE ID i.e. dragonboard410c
  - DISTRO i.e. rpb
- Choices come from 96Boards specific metalayers
  - meta-96boards
  - meta-qcom



```
Available Distributions
Please choose a distribution

rpb          meta-rpb
rpb-eglfs    meta-rpb
rpb-wayland  meta-rpb
```



```
Available Machines
Please choose a machine

dragonboard-410c     meta-qcom
dragonboard-410c-32  meta-qcom
dragonboard-820c     meta-qcom
hikey                meta-96boards
hikey-32             meta-96boards
hikey960             meta-96boards
```

©2017 Timesys Corp.

3

Running Yocto Build for the DragonBoard 410c

©2017 Timesys Corp.

Exercise 3: Rebuilding console images

- **96Boards Yocto takes advantage of "caches"**
  - The goal is to accelerate build process
  - Can be copied to other machines
  - Contain build output
  - The following variables are typically placed in your conf/local.conf – in Linaro™ Yocto, it is placed in site.conf
    - DL_DIR = /home/tsu/LAB-410c/96boards-yocto/downloads
    - SSTATE_DIR = /home/tsu/LAB-410c/96boards-yocto/sstate-cache
- **BSP image definition – special type of recipe**

Image Name	Description
rpb-console-image	Console image
rpb-desktop-image	Image that's based on X11, leveraging hardware acceleration
rpb-wayland-image	Image with lightweight windowing system
rpb-qt5-image	Image with QT5 toolkit

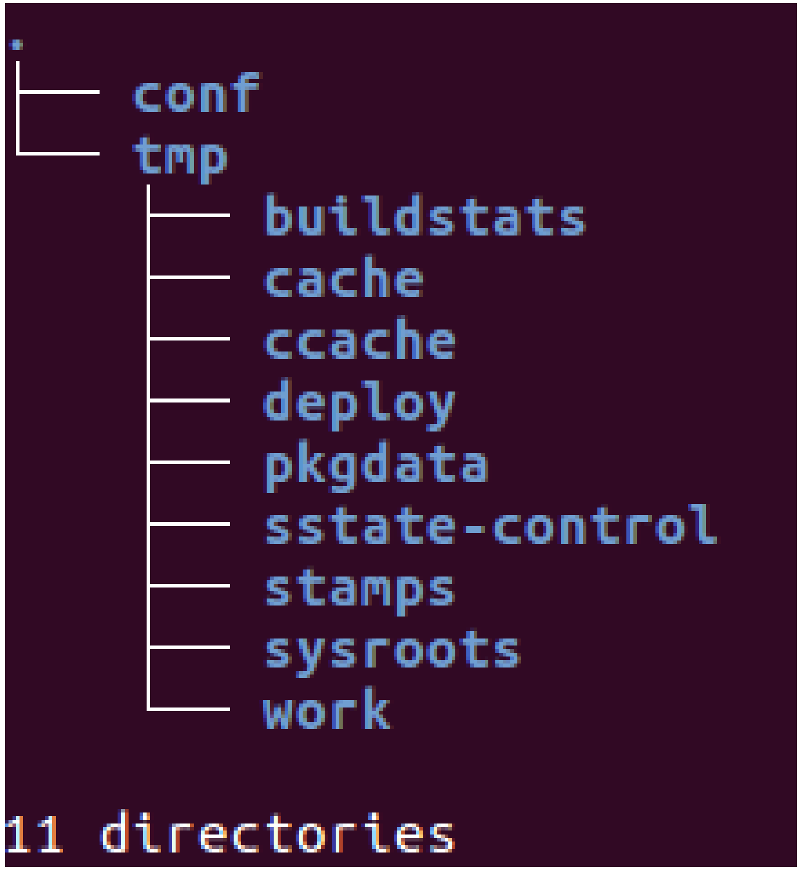
- **Building a BSP image**

```
$ bitbake rpb-console-image
```

©2017 Timesys Corp.

Yocto output

- Output from the build is stored in several directories



```
.
├── conf
└── tmp
    ├── buildstats
    ├── cache
    ├── ccache
    ├── deploy
    ├── pkgdata
    ├── sstate-control
    ├── stamps
    ├── sysroots
    └── work

11 directories
```

- **tmp/deploy**
  - Images for deployment
- **tmp/work**
  - Source code, the patches, last build

We can now re-test images built using deployment process discussed earlier

©2017 Timesys Corp.

4

Building Yocto SDK for Application Development with the DragonBoard 410c

©2017 Timesys Corp.

LAB 4: Building a Yocto SDK

- **The Yocto Project is not intended for application development**
- **A separate SDK can be generated from a Yocto Project build**
  - Once created, SDK is completely independent of the Yocto Project build system
  - Come in a form of a shell script which facilitates SDK setup on a development host
- **Yocto allows you to generate two types of SDKs:**

1. Generic i.e.

```
$ bitbake meta-toolchain
```

2. Image based i.e.

```
$ bitbake –c populate_sdk rpb-console-image
```

©2017 Timesys Corp.

Questions?

developer.qualcomm.com 96boards.org arrow.com timesys.com

©2017 Timesys Corp.

Thank you

Follow us on: f 🐦 in t

For more information, visit us at:

www.qualcomm.com & www.qualcomm.com/blog

All data and information contained in or disclosed by this document is confidential and proprietary information of Qualcomm Technologies, Inc. and/or its affiliated companies and all rights therein are expressly reserved. By accepting this material the recipient agrees that this material and the information contained therein is to be held in confidence and in trust and will not be used, copied, reproduced in whole or in part, nor its contents revealed in any manner to others without the express written permission of Qualcomm Technologies, Inc. Nothing in these materials is an offer to sell any of the components or devices referenced herein.

©2017 Qualcomm Technologies, Inc. and/or its affiliated companies. All Rights Reserved.

Qualcomm and DragonBoard are trademarks of Qualcomm Incorporated, registered in the United States and other countries. Other products and brand names may be trademarks or registered trademarks of their respective owners.

References in this presentation to “Qualcomm” may mean Qualcomm Incorporated, Qualcomm Technologies, Inc., and/or other subsidiaries or business units within the Qualcomm corporate structure, as applicable. Qualcomm Incorporated includes Qualcomm’s licensing business, QTL, and the vast majority of its patent portfolio. Qualcomm Technologies, Inc., a wholly-owned subsidiary of Qualcomm Incorporated, operates, along with its subsidiaries, substantially all of Qualcomm’s engineering, research and development functions, and substantially all of its product and services businesses, including its semiconductor business, QCT.