

# LVC21-114: The Case for UEFI Boot on Arm-powered IoT Devices (Again)

For about the millionth time.

;-)

David Tischler



# Yet Another Take on Why Standards Matter

Warning: This is not a technical talk, its a story about attempting to bring up a Yocto-based OS on an Arm IoT device. Oh, and:

\$

€

¥

£

# Buzzword Bingo

But first, let's talk about the 5G and Edge Computing revolution (that will of course be powered by the quantum hyperscale blockchain AI running on distributed serverless multicloud K8S).

Feel free to insert any buzzwords I may have missed...

# But seriously, IoT and Edge Computing is everywhere

My \$dayjob is at an IoT Fleet Management platform, balena.io

We support ~60 Arm device types, including all of the various Pi's, Jetsons, Beagles, and more enterprise-focused boards from Variscite, CTI, Compulab, etc.

We build our own Yocto-based OS, with a container engine and VPN link

Which actually means we have to build ~60 unique OS'es, with differing bootloaders and boot partitions and alignment. (Spoiler alert: this is not scalable)

We build **two** x86 images: NUC, and generic x86-64.

# But seriously, IoT and Edge Computing is everywhere

balena

What is balena? balenaCloud More products Resources Pricing Customers About Login Sign up

## The container-based platform for deploying IoT applications

Comprehensive device deployment and management infrastructure, hosted by balena.

**balenaCloud**

Your first 10 devices are always free and full-featured.

Email

Get started

GIT SERVER BUILDER DOCKER REGISTRY BALENA API YEM MANAGER

AIO 3288C	BalenaOS 2.38.3+rev10	armv7hf	<a href="#">Production</a>	<a href="#">Development</a>
Aetina N310 TX2	BalenaOS 2.56.0+rev4	aarch64	<a href="#">Production</a>	<a href="#">Development</a>
Aetina NS10 TX2	BalenaOS 2.56.0+rev4	aarch64	<a href="#">Production</a>	<a href="#">Development</a>
Asus Tinker Board	BalenaOS 2.38.3+rev1	armv7hf	<a href="#">Production</a>	<a href="#">Development</a>
Asus Tinker Board S	BalenaOS 2.38.3+rev1	armv7hf	<a href="#">Production</a>	<a href="#">Development</a>
Aviavision JN30B Nano	BalenaOS 2.67.3+rev2	aarch64	<a href="#">Production</a>	<a href="#">Development</a>
Balena Fin (CM3)	BalenaOS 2.72.0+rev1	armv7hf	<a href="#">Production</a>	<a href="#">Development</a>
BananaPI-M1+	BalenaOS 2.12.7+rev4	armv7hf	<a href="#">Production</a>	<a href="#">Development</a>
BeagleBoard-XM	BalenaOS 2.38.0+rev1	armv7hf	<a href="#">Production</a>	<a href="#">Development</a>
BeagleBone Black	BalenaOS 2.73.1+rev2	armv7hf	<a href="#">Production</a>	<a href="#">Development</a>
BeagleBone Green	BalenaOS 2.58.3+rev1	armv7hf	<a href="#">Production</a>	<a href="#">Development</a>
BeagleBone Green Gateway	BalenaOS 2.67.3+rev1	armv7hf	<a href="#">Production</a>	<a href="#">Development</a>
BeagleBone Green Wireless	BalenaOS 2.51.1+rev2	armv7hf	<a href="#">Production</a>	<a href="#">Development</a>
CTI Astro TX2 G+	BalenaOS 2.56.0+rev4	aarch64	<a href="#">Production</a>	<a href="#">Development</a>
CTI Orbitty TX2	BalenaOS 2.51.1+rev4	aarch64	<a href="#">Production</a>	<a href="#">Development</a>

# You Are All Engineers

...and u-boot, device tree, bootloader, kernel and board enablement experts. But the other 99.999% of the world is not. So what happens when “regular” business users and IT departments want to begin participating in this Edge Computing revolution?

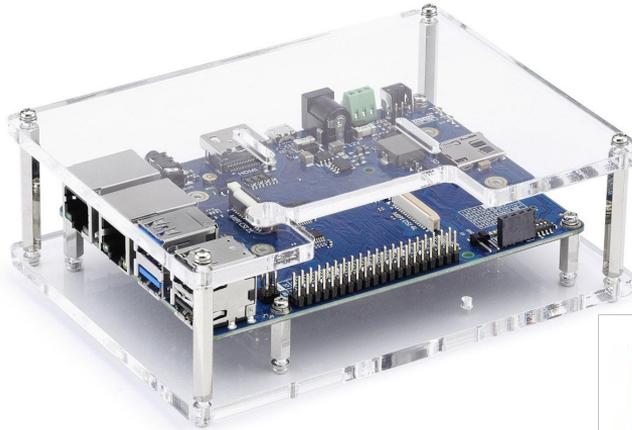
Let's find out!



# Let's Perform an Experiment

I have an I-Pi SMARC PX30:

- Rockchip PX30
- 2gb RAM
- 2x LAN
- USB 3.0
- CANBUS



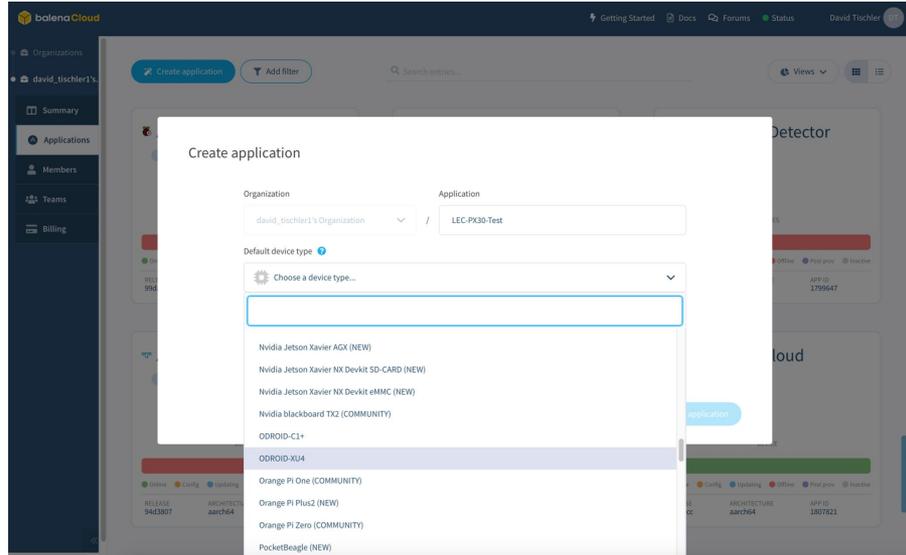
Now I want to boot balenaOS, so that I can control the board using the native cloud platform



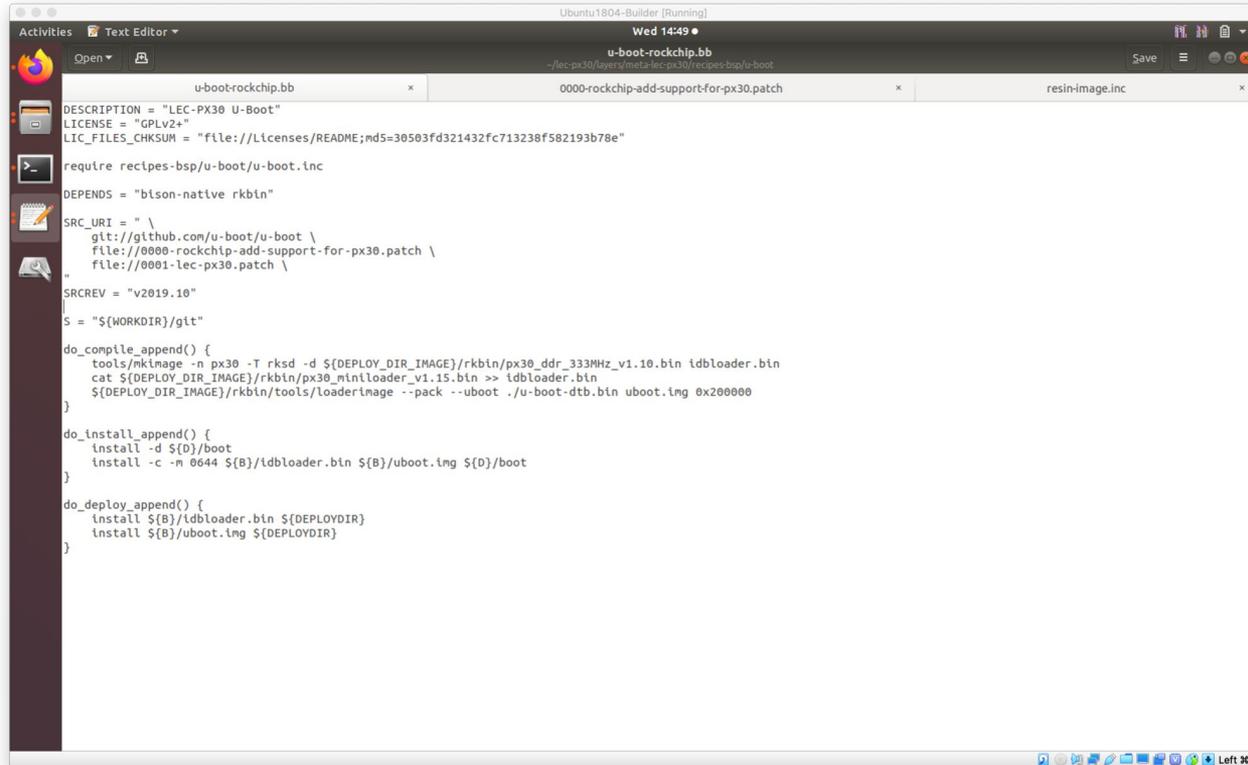
# I Am Not An Engineer

I just want to run this OS. This shouldn't be complicated. Just flash something to a USB stick and boot, right?

Wrong.



# I Am Not An Engineer



The image shows a screenshot of a Linux desktop environment with a terminal window open. The terminal window title is "Ubuntu1904-Builder [Running]" and the current time is "Wed 14:49". The terminal displays the contents of a BitBake recipe file named "u-boot-rockchip.bb". The recipe includes a description, license, source URIs, and several task definitions for compilation, installation, and deployment.

```
DESCRIPTION = "LEC-PX30 U-Boot"
LICENSE = "GPLv2+"
LIC_FILES_CHKSUM = "file://Licenses/README;md5=30503fd321432fc713238f582193b78e"

require recipes-bsp/u-boot/u-boot.inc

DEPENDS = "bison-native rkbin"

SRC_URI = " \
    git://github.com/u-boot/u-boot \
    file://0000-rockchip-add-support-for-px30.patch \
    file://0001-lec-px30.patch \
"

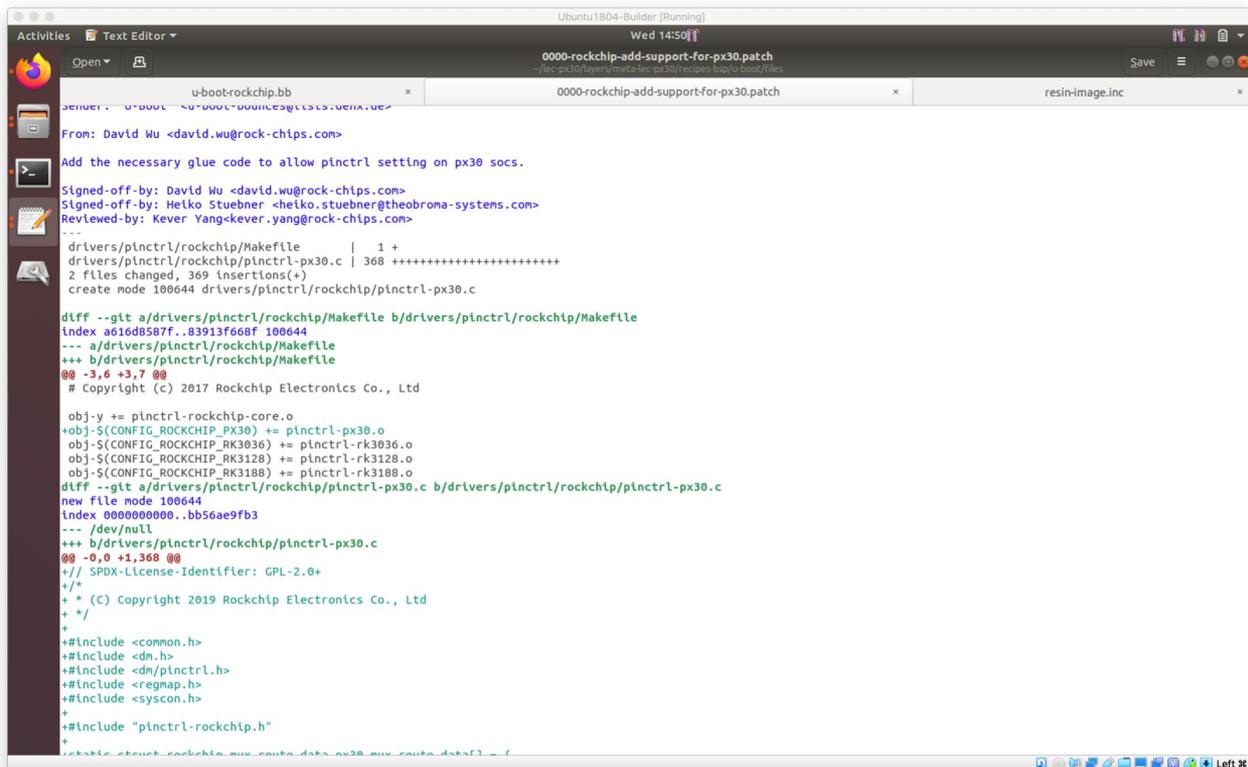
SRCREV = "v2019.10"
S = "${WORKDIR}/git"

do_compile_append() {
    tools/mkImage -n px30 -T rkfd -d ${DEPLOY_DIR_IMAGE}/rkbin/px30_ddr_333MHz_v1.10.bin idbloader.bin
    cat ${DEPLOY_DIR_IMAGE}/rkbin/px30_miniloader_v1.15.bin >> idbloader.bin
    ${DEPLOY_DIR_IMAGE}/rkbin/tools/loaderImage -pack --uboot ./u-boot-dtb.bin uboot.img 0x200000
}

do_install_append() {
    install -d ${D}/boot
    install -c -m 0644 ${B}/idbloader.bin ${B}/uboot.img ${D}/boot
}

do_deploy_append() {
    install ${B}/idbloader.bin ${DEPLOYDIR}
    install ${B}/uboot.img ${DEPLOYDIR}
}
```

# I Am Not An Engineer

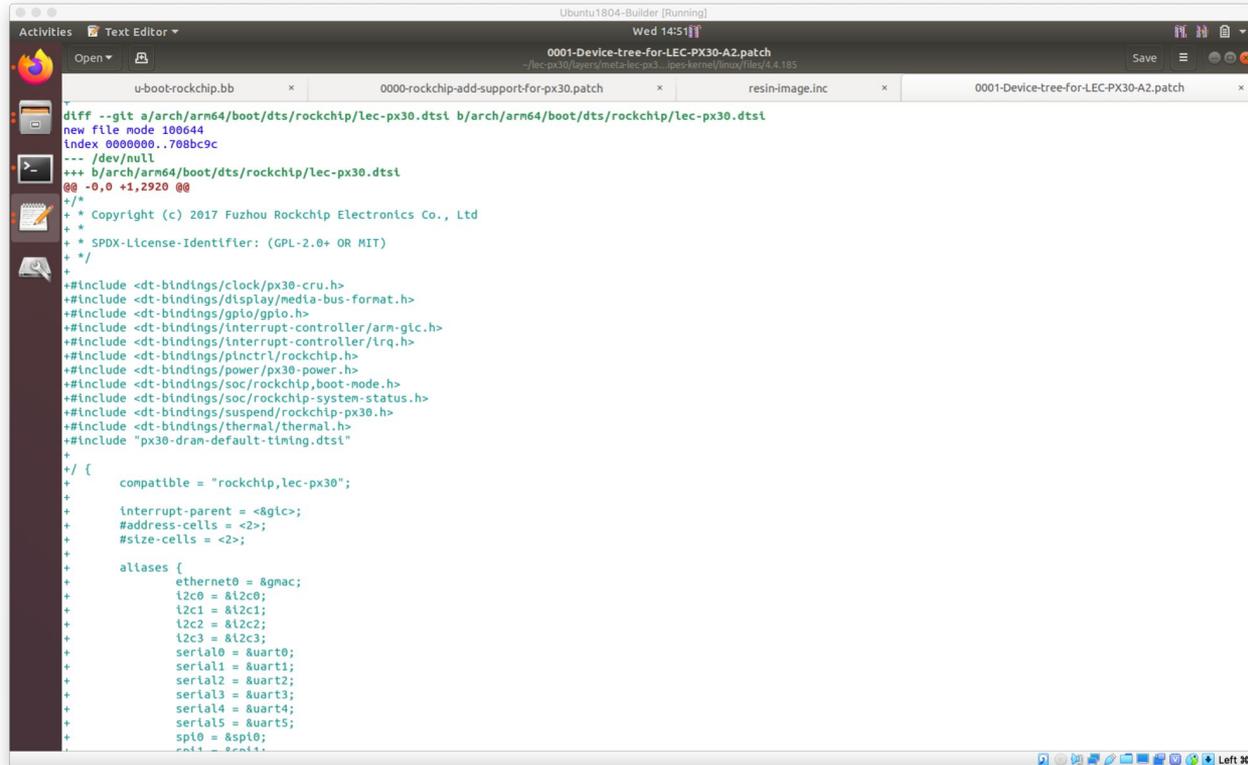


```
u-boot-rockchip.bb x 0000-rockchip-add-support-for-px30.patch resin-image.inc
From: David Wu <david.wu@rock-chips.com>
Add the necessary glue code to allow pinctrl setting on px30 socs.
Signed-off-by: David Wu <david.wu@rock-chips.com>
Signed-off-by: Heiko Stuebner <heiko.stuebner@theobroma-systems.com>
Reviewed-by: Kever Yang <kever.yang@rock-chips.com>
---
 drivers/pinctrl/rockchip/Makefile | 1 +
 drivers/pinctrl/rockchip/pinctrl-px30.c | 368 +++++++++++++++++++++
 2 files changed, 369 insertions(+)
 create mode 100644 drivers/pinctrl/rockchip/pinctrl-px30.c

diff --git a/drivers/pinctrl/rockchip/Makefile b/drivers/pinctrl/rockchip/Makefile
index a616d8587f..83913f668f 100644
--- a/drivers/pinctrl/rockchip/Makefile
+++ b/drivers/pinctrl/rockchip/Makefile
@@ -3,6 +3,7 @@
 # Copyright (c) 2017 Rockchip Electronics Co., Ltd

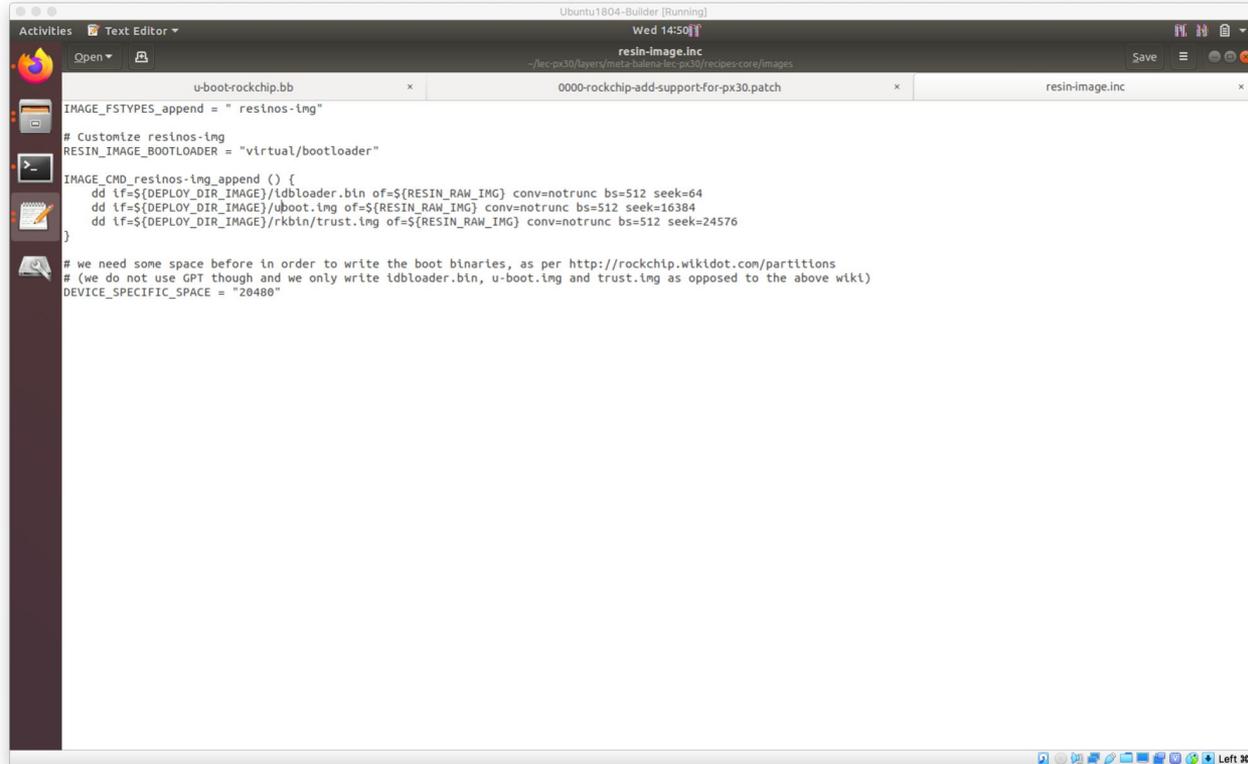
obj-y += pinctrl-rockchip-core.o
obj-$(CONFIG_ROCKCHIP_PX30) += pinctrl-px30.o
obj-$(CONFIG_ROCKCHIP_RK3036) += pinctrl-rk3036.o
obj-$(CONFIG_ROCKCHIP_RK3128) += pinctrl-rk3128.o
obj-$(CONFIG_ROCKCHIP_RK3188) += pinctrl-rk3188.o
diff --git a/drivers/pinctrl/rockchip/pinctrl-px30.c b/drivers/pinctrl/rockchip/pinctrl-px30.c
new file mode 100644
index 0000000000..bb56ae9fb3
--- /dev/null
+++ b/drivers/pinctrl/rockchip/pinctrl-px30.c
@@ -0,0 +1,368 @@
+// SPDX-License-Identifier: GPL-2.0+
+/*
+ * (c) Copyright 2019 Rockchip Electronics Co., Ltd
+ */
+
+#include <common.h>
+#include <dm.h>
+#include <dm/pinctrl.h>
+#include <regmap.h>
+#include <syscon.h>
+
+#include "pinctrl-rockchip.h"
+
+static struct rockchip_mux_gpio_data px30_mux_gpio_data[] = {
```

# I Am Not An Engineer



```
diff --git a/arch/arm64/boot/dts/rockchip/lec-px30.dtsi b/arch/arm64/boot/dts/rockchip/lec-px30.dtsi
new file mode 100644
index 0000000..709bc9c
--- /dev/null
+++ b/arch/arm64/boot/dts/rockchip/lec-px30.dtsi
@@ -0,0 +1,2920 @@
+
+/*
+ * Copyright (c) 2017 Fuzhou Rockchip Electronics Co., Ltd
+ *
+ * SPDX-License-Identifier: (GPL-2.0+ OR MIT)
+ */
+
+#include <dt-bindings/clock/px30-cru.h>
+#include <dt-bindings/display/media-bus-format.h>
+#include <dt-bindings/gpio/gpio.h>
+#include <dt-bindings/interrupt-controller/arm-gic.h>
+#include <dt-bindings/interrupt-controller/irq.h>
+#include <dt-bindings/pinctrl/rockchip.h>
+#include <dt-bindings/power/px30-power.h>
+#include <dt-bindings/soc/rockchip/boot-mode.h>
+#include <dt-bindings/soc/rockchip-system-status.h>
+#include <dt-bindings/suspend/rockchip-px30.h>
+#include <dt-bindings/thermal/thermal.h>
+#include "px30-dran-default-timing.dtsi"
+
+/* {
+
+    compatible = "rockchip,lec-px30";
+
+    interrupt-parent = <gic>;
+    #address-cells = <2>;
+    #size-cells = <2>;
+
+    aliases {
+        ethernet0 = &mac;
+        i2c0 = &i2c0;
+        i2c1 = &i2c1;
+        i2c2 = &i2c2;
+        i2c3 = &i2c3;
+        serial0 = &uart0;
+        serial1 = &uart1;
+        serial2 = &uart2;
+        serial3 = &uart3;
+        serial4 = &uart4;
+        serial5 = &uart5;
+        spi0 = &spi0;
+        spi1 = &spi1;
```

# I Am Not An Engineer



The screenshot shows a text editor window titled "resin-image.inc" with the following content:

```
u-boot-rockchip.bb          0000-rockchip-add-support-for-px30.patch          resin-image.inc
IMAGE_FSTYPES_append = " resnos-tmg"
# Customize resnos-tmg
RESIN_IMAGE_BOOTLOADER = "virtual/bootloader"

IMAGE_CMD_resnos-tmg_append () {
    dd if=${DEPLOY_DIR_IMAGE}/idbloader.bin of=${RESIN_RAW_IMG} conv=notrunc bs=512 seek=64
    dd if=${DEPLOY_DIR_IMAGE}/u-boot.tmg of=${RESIN_RAW_IMG} conv=notrunc bs=512 seek=16384
    dd if=${DEPLOY_DIR_IMAGE}/rkbin/trust.tmg of=${RESIN_RAW_IMG} conv=notrunc bs=512 seek=24576
}

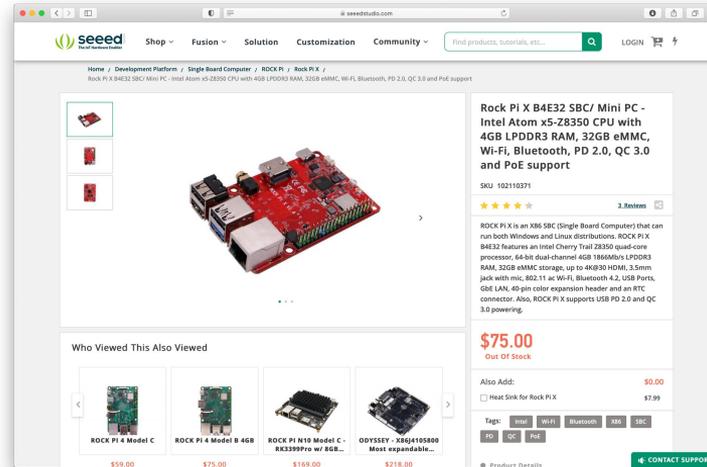
# we need some space before in order to write the boot binaries, as per http://rockchip.wikidot.com/partitions
# (we do not use GPT though and we only write idbloader.bin, u-boot.tmg and trust.tmg as opposed to the above wiki)
DEVICE_SPECIFIC_SPACE = "20480"
```

# End Result?

It took me 6 months of reading, research, learning, and “side project” on/off work to boot this board and successfully run my OS of choice. This is not feasible.

So out of curiosity, I wonder how hard it is to boot an x86 board? Are all IoT projects this painful?

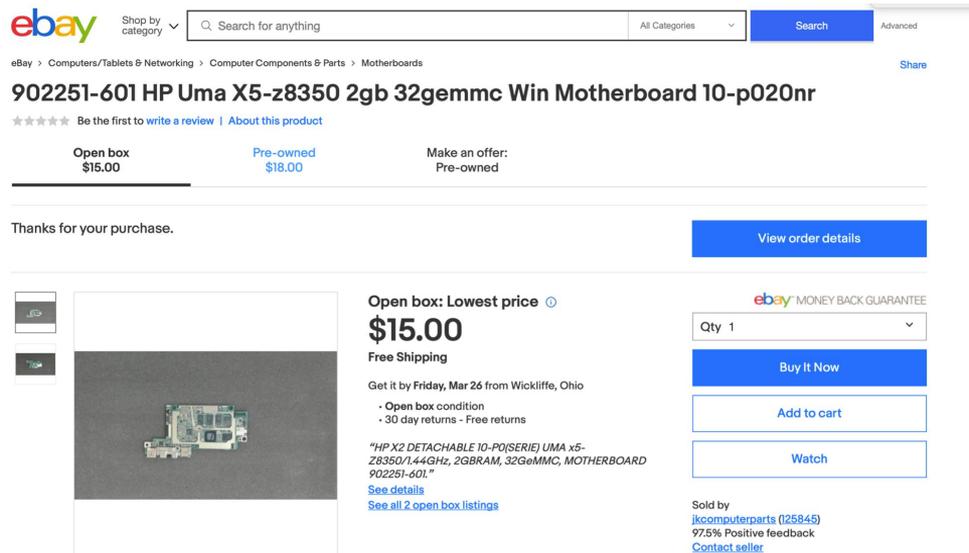
I have seen small x86 devices before, like this “Rock Pi X”:



# x86 Comparo

Hmmm, the “Rock Pi X” is Out of Stock, so let’s see what’s on ebay:

This is interesting, the same SoC, but originally in a 2-in-1 tablet laptop thing:



The screenshot shows an eBay product listing for an HP motherboard. The listing includes the eBay logo, a search bar, and navigation links. The product title is "902251-601 HP Uma X5-z8350 2gb 32gmmc Win Motherboard 10-p020nr". The price is listed as \$15.00 for an open box item. The listing also features a "Buy It Now" button, an "Add to cart" button, and a "Watch" button. The seller's name is "jkcomputerparts" with a 97.5% positive feedback rating. The listing includes a photo of the motherboard and a detailed description of the product specifications.

Shop by category  All Categories  Advanced

eBay > Computers/Tablets & Networking > Computer Components & Parts > Motherboards Share

### 902251-601 HP Uma X5-z8350 2gb 32gmmc Win Motherboard 10-p020nr

★★★★★ Be the first to [write a review](#) | [About this product](#)

Open box **\$15.00** Pre-owned **\$18.00** Make an offer: Pre-owned

Thanks for your purchase.

Open box: **Lowest price**   
**\$15.00**  
Free Shipping

Get it by **Friday, Mar 26** from Wickliffe, Ohio

- Open box condition
- 30 day returns - Free returns

*"HP X2 DETACHABLE 10-P0(SERIE) UMA x5-Z8350/1.44GHz, 2GBRAM, 32GeMMC, MOTHERBOARD 902251-601."*

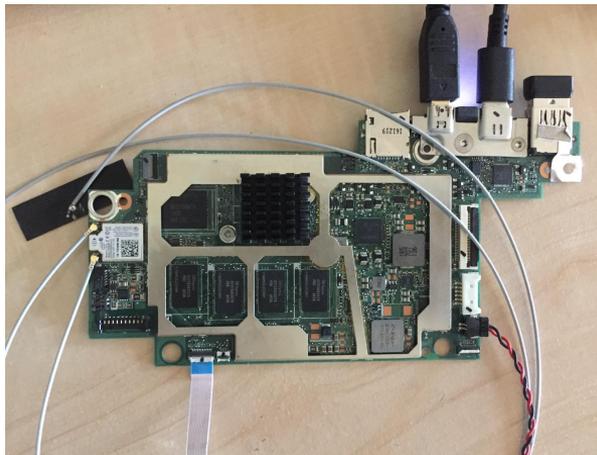
[See details](#)  
[See all 2 open box listings](#)

**ebay** MONEY BACK GUARANTEE

Qty 1

Sold by [jkcomputerparts \(25845\)](#)  
97.5% Positive feedback  
[Contact seller](#)

# x86 Comparo



# Final Results

It showed up in the mail, I plugged it in, entered UEFI, disabled Secure Boot, selected my USB stick to boot from, and it “Just Worked”.

Connected to balenaCloud, in approximately 12 minutes start to finish.

The image displays two screenshots from the balenaCloud web interface. The left screenshot shows the 'Create application' dialog box. The 'Organization' is set to 'david\_tischler's Organization' and the 'Application' is 'Atom350'. The 'Default device type' is 'Generic x86\_64 (NEW)'. The 'Application type' is 'View docs' and the 'Microservices' are 'recommended'. The 'Advanced' toggle is off. The right screenshot shows the 'Atom350' application page. The status is 'Updating' with a progress bar at 6%. The device type is 'Generic x86\_64'. The page includes a summary, device variables, device configuration, actions, and a terminal window. The terminal window shows logs for the 'iprtuss' service, including messages like 'Service exited' and 'Killed service'.

# Prototype to Production for a Typical Customer

So after that exercise in frustration on the Arm device, an IT organization can then finally begin to build out their product, scale, and launch.

6 months behind schedule.

Or at the 4-month mark, they give up, and buy Intel NUC's.

# Time, Money

I promised we would talk about money.

Now, I performed this exercise part time, for 6 months, and am not a true subject matter expert.

So what would a company do? Hire an embedded engineer? That will add 100k (USD) to the budget for their project. Let their project timeline slip? Who is going to deliver that bad news to the executive team?

No, they will pivot to using x86 devices that “just boot”

# The Future

Let's look into our crystal ball, and predict what happens in 5 to 10 years:

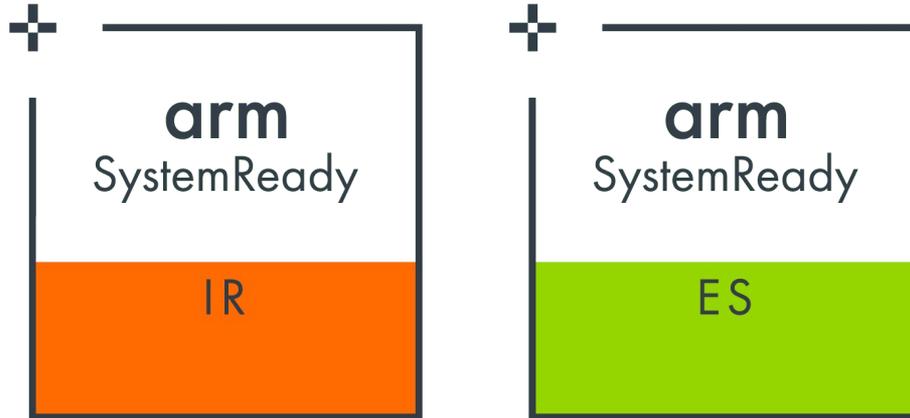


Arm loses the Edge Computing market, and x86 wins.

Because “regular businesses” want to participate in Edge computing, and they can't.

# The Ask

Board manufacturers, with the help of the SoC vendors if/as needed, produce developer boards that we know and love with UEFI, so we can just run any OS we want and not worry about learning all the things I had to learn.



# Thank you

Accelerating deployment in the Arm Ecosystem

