K3s

Developing Turnkey Devices for Edge Use Cases

Mark Abrams - Field Engineer, Edge Specialist
Rancher Labs

DevOps INSTITUTE
ADVANCING THE HUMANS OF DEVOPS

GLOBAL SKILup SM FESTIVAL
by DevOps Institute

December 10, 2020

Turnkey

Turnkey solutions are those which result in the end user having full access to a solution simply by plugging in and turning on. With modern methods, turnkey systems can also be highly customized solutions with site specific configuration applied automatically.

Turnkey solutions at the edge are personalized appliances.

Defining Turnkey for the Edge

hint: not sneakernet

- Sneaker net was used to customize the end product
- At least some level of connectivity is required
    - Expect network latency
    - Expect disconnected networks
    - Expect zero trust security

K3s

Container Orchestration

Container Orchestration

K3s is a Kubernetes Distribution

- Kubernetes allows us to declare a system “desired state”
- Built to manage containers so you don’t have to
- K3s helps us isolate the hardware from the software
  - Can manage hardware and software components
  - Easily updated/upgraded without flashing firmware
  - Can provide reliability, availability and scalability

Turnkey Use Cases

Solutions Across the Edge

Retail and Branch

- POS systems
- Inventory
- Ordering
- Couponing and discounts
- AuthN/AuthZ
- Loyalty tracking

Industrial IoT

- Assembly line gateway
- Messaging (amqp/mqtt)
- GPU/AI/ML inference
- Redundant systems

Resource Heavy

- Large HPC
- GPU/AI/ML inference
- Threat detection and response

Looking Ahead

The world of tomorrow?

- Thermal cameras
- Self driving cars – a.k.a. mobile data center
- Civil engineering
  - Buildings
  - Greenhouses/farming
  - Infrastructure
  - Energy

Common Operations

Turnkey solutions tend to have these things in common

- Collect information from a non-technical individual
- Pull configuration from a remote location
- Install/Setup/Configure applications
- Configure / Modify the underlying system

Example Turnkey

Problem Statement

Real World Problems

A marketing giveaway

- $100.00 budget
- Highlights the k3s lightweight Kubernetes project
- Demonstrates simplicity of use

A Real World Solution

Taking it to task

- Raspberrypi 4B 4GB ($50)
- 16GB SD Card ($12)
- Case ($20 – it’s custom)
- Power supply ($8)

Real Problems

V1 - its not turnkey

First you must install an OS

Then you need to configure your network

Once that is setup, you can install K3s

RANCHER

First Things First:

1. Plug in a keyboard to one of the USB ports
2. Plug in a mouse to another USB port
3. Connect a monitor using an HDMI cable

1. Connect your system

- Use the power supply that came with your k3s Raspberry Pi and plug in to the micro-usb port.
- After the color splash screen there is a prompt to enter recovery mode. At the prompt, hold down the (shift) key.
- This opens the NOOBS installer where you can easily select the Raspbian distribution and setup your network
- Follow the next set of steps to install the Raspbian OS.

2. Install an operating system

- After the NOOBS install screen opens, press (w) to setup your WiFi network.
- Select your SSID and enter credentials as needed. The list of installable options will update.
- De-select the default Raspbian
- Select Raspbian Lite to provide the most capacity and resources available to your k3s cluster
- Press (i) to install the OS.
- Press "yes" button to confirm overwriting the existing OS
- Press "OK" when prompted that the OS is installed
- The system will reboot.

Login credentials:

user: **pi**
password: **raspberry**

Run the following command to install k3s:

```
curl -sfL https://get.k3s.io | sh -
```

Congratulations!

Kubernetes is running on your Rancher Pi.
Try some of these commands:

```
sudo kubectl get node
sudo kubectl create deployment nginx
--image=nginx
```

Visit https://github.com/rancher/k3s to learn more

A Real Solution

V2 - Plug and play

Plug the device in

Visit the configuration page

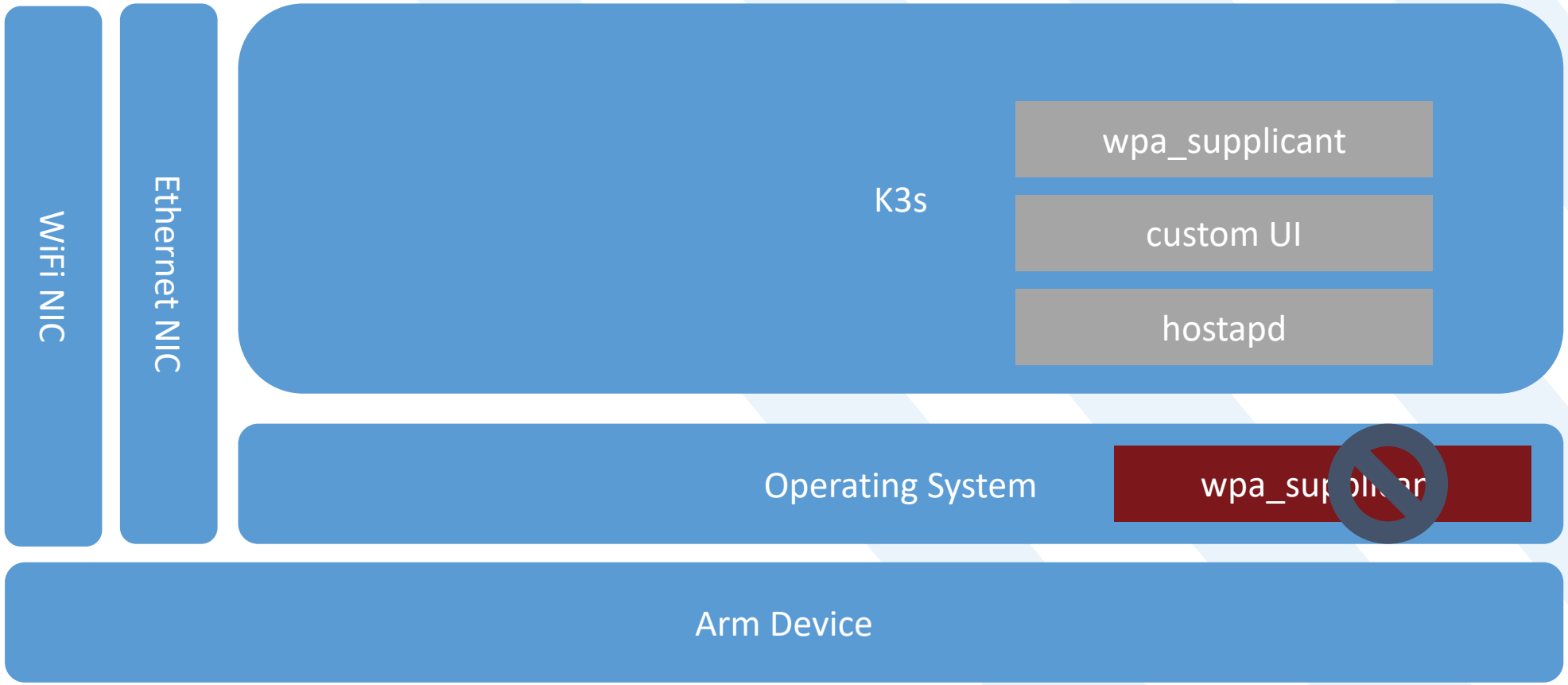
Enter credentials for your selected network

Demo

Let’s visualize the end product

Solution Architecture

Architecture



Code and Process

Build Process

The exact steps are scripted and under source control in the github repository

http://github.com/mak3r/turnkey

1. Temporarily configure internet access. I used ethernet with dhcp
2. Install k3s Here are some recommended options:
   - `curl -sfL https://get.k3s.io | INSTALL_K3S_EXEC="--tls-san raspberrypi --write-kubeconfig /home/pi/.kube/config --no-deploy servicelb --resolv-conf /var/lib/rancher/turnkey/resolv.conf" sh -`
   - The servicelb must be disabled
   - A host network must be available to kubernetes
3. Make sure there is a temporary network for k3s to startup
   - One way to do this is to add this to `/etc/rc.local`

```
ip link set dev eth0 up
ip addr add 192.168.1.1/24 brd 192.168.255.255 dev eth0
route add default gw 192.168.1.1
```

Build Process (Continued)

A few more steps to build the device image

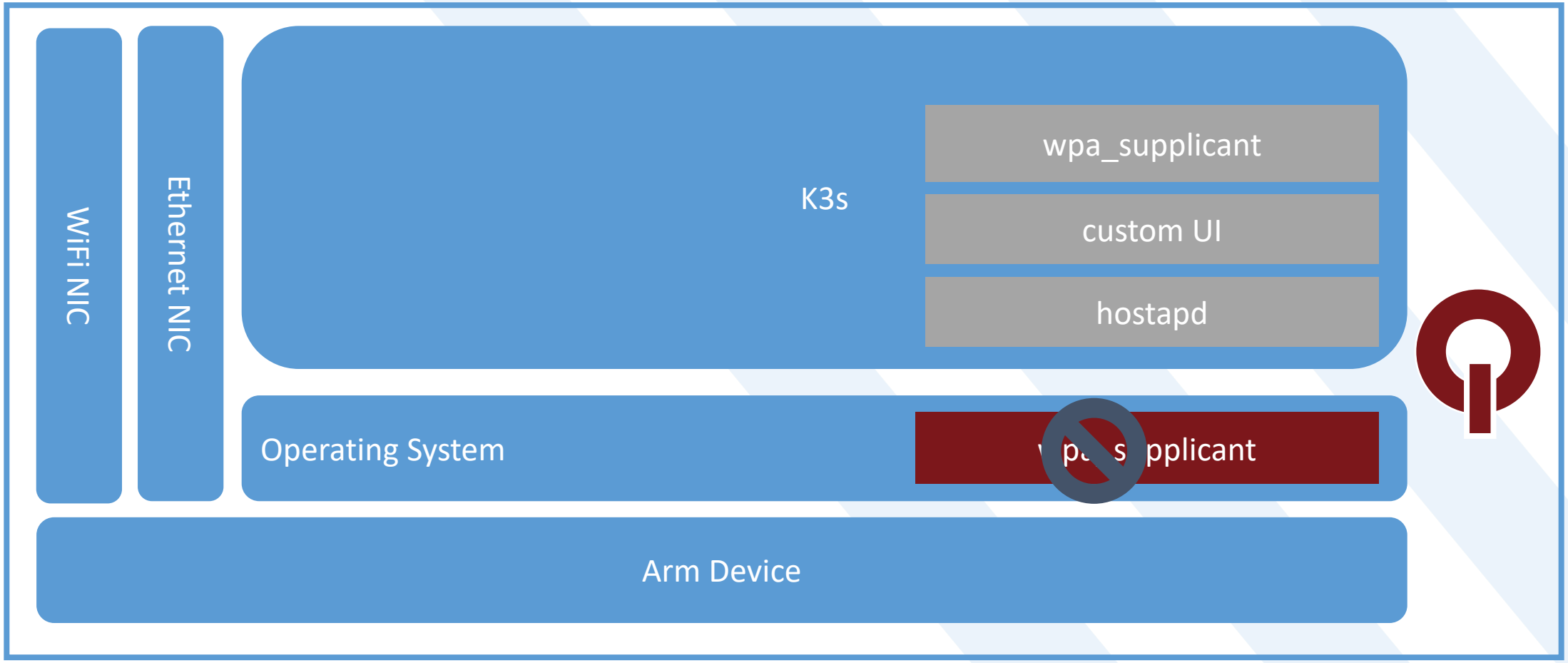
http://github.com/mak3r/turnkey

4. Add 2 files to `/var/lib/rancher/k3s/server/manifests/`
   - k8s/turnkey-ns.yaml
   - k8s/interactive-setup.yaml
5. Configure a resolv.conf for kubernetes
   - `/var/lib/rancher/turnkey/resolv.conf`

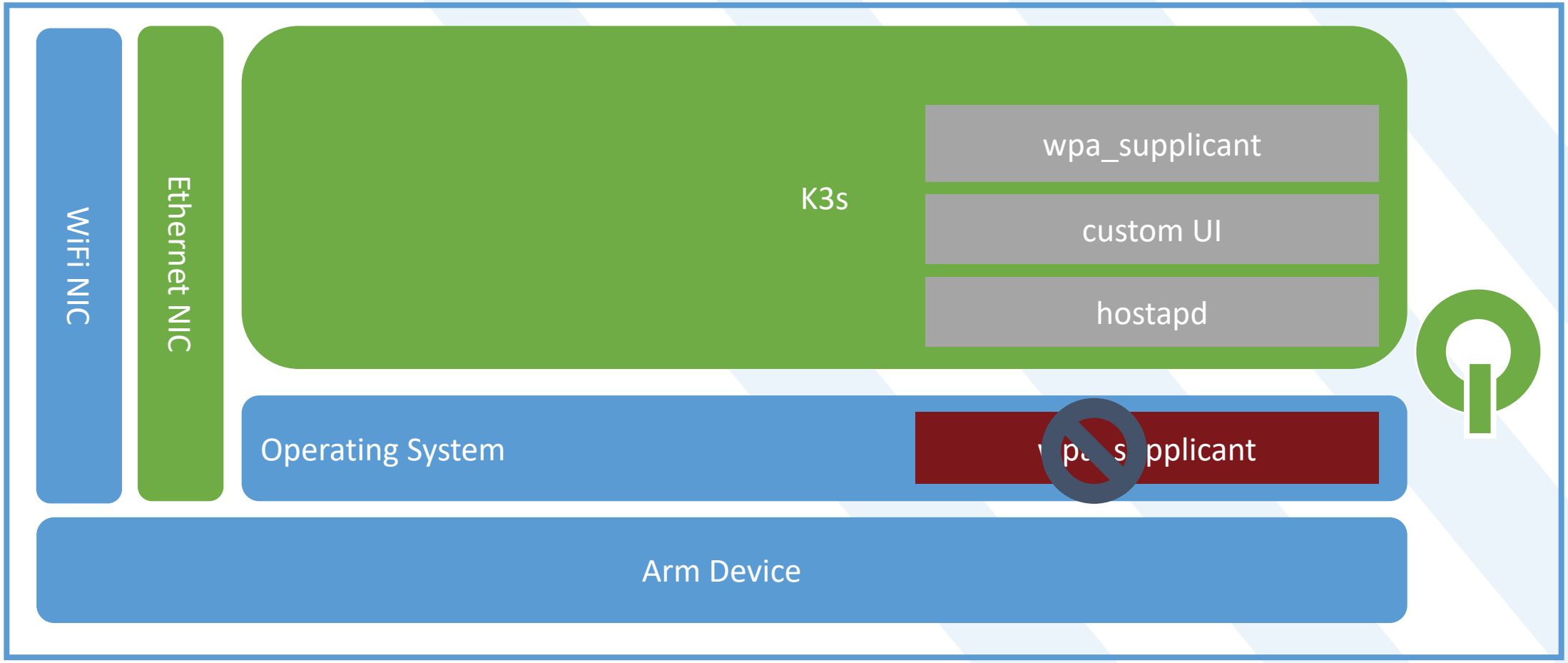
```
domain lan
nameserver 192.168.1.1
```

6. Stop k3s `sudo systemctl stop k3s`
7. Image the sd card for use in other devices

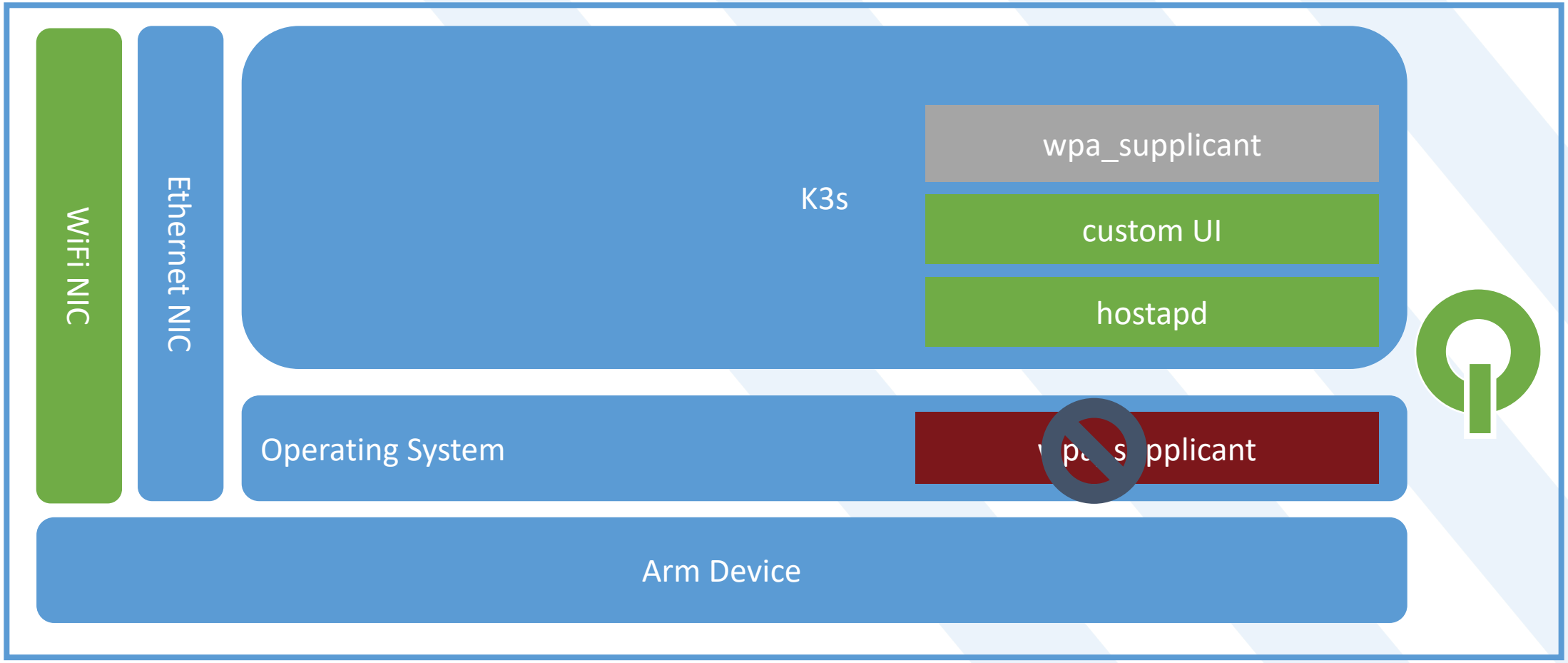
Here is our device



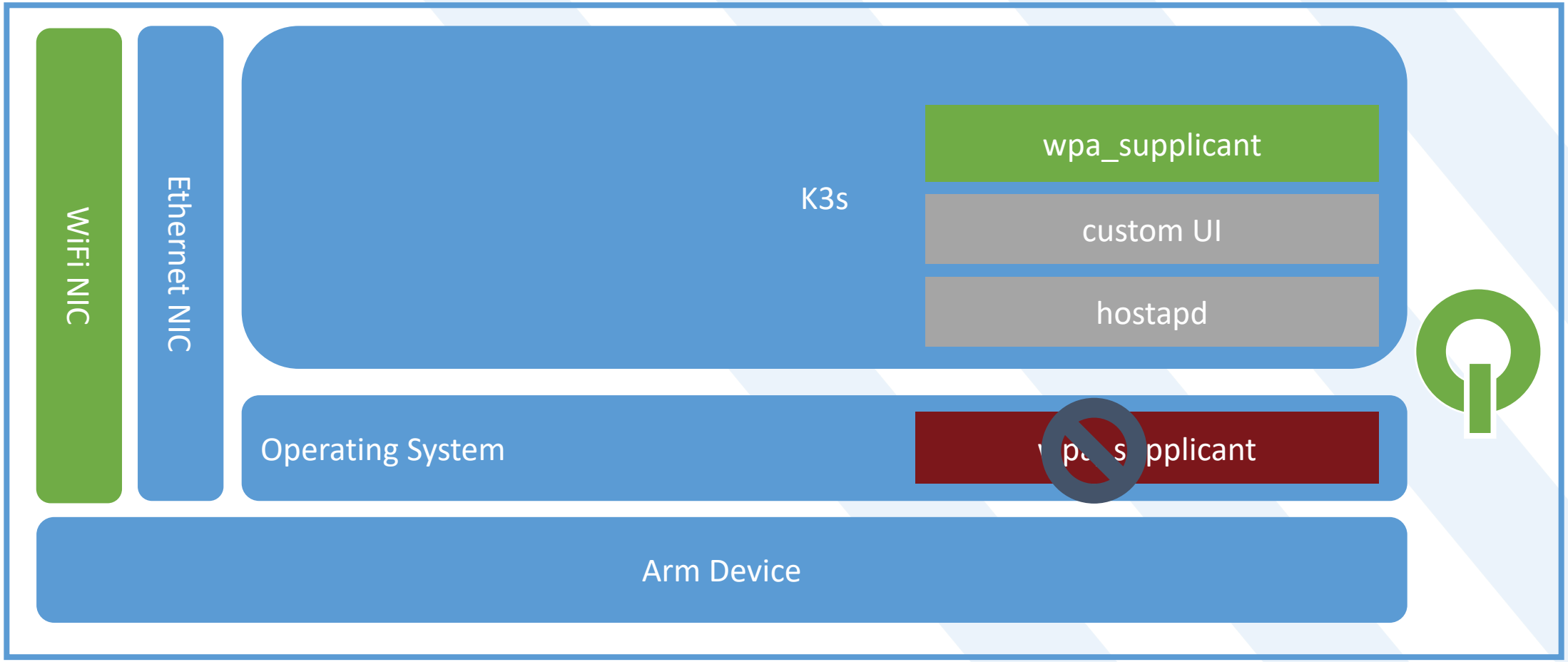
Step 1 - User turns it on



Step 2 - Setup and AP to capture configuration data



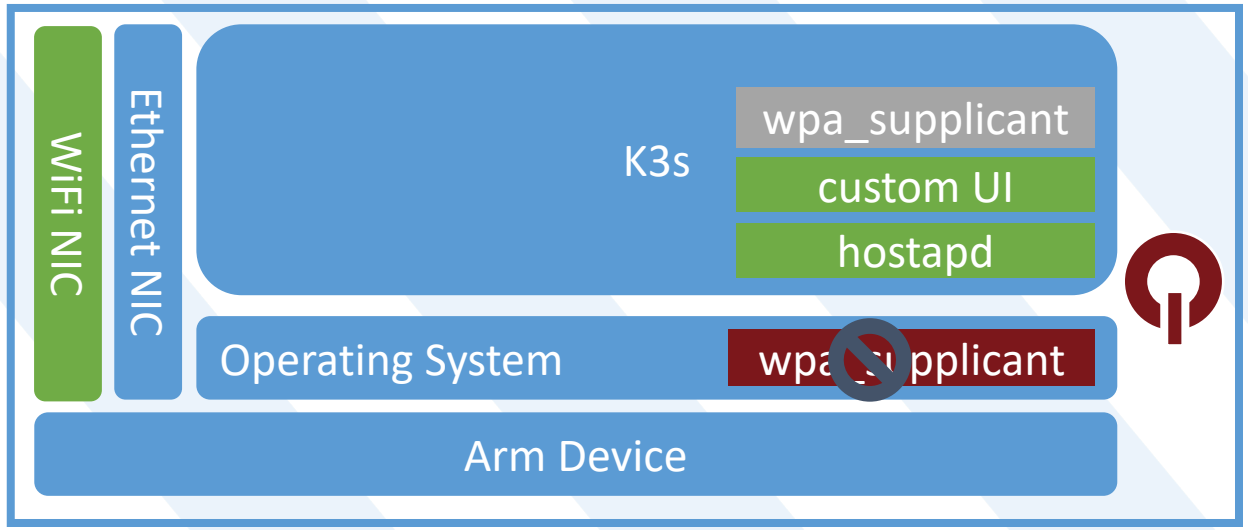
Step 3 - Shutdown the AP and use the configuration data to connect to an SSID



Turnkey Example

Setting up the AP uses K8s Job Type

```yaml
apiVersion: batch/v1
kind: Job
metadata:
  name: hostapd
  namespace: turnkey
spec:
  template:
    metadata:
      labels:
        turnkey/workloadselector: job-turnkey-hostapd
    spec:
      containers:
      - args:
        - up
        image: mak3r/hostapd:v0.0.10
```

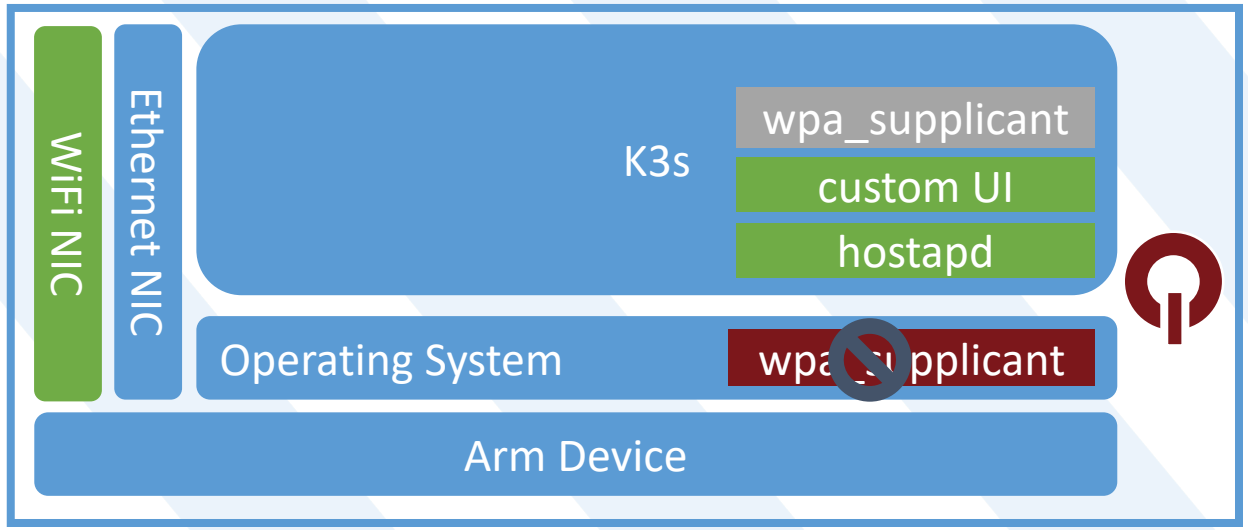


The UI Job has 2 containers

```yaml
kind: Job
metadata:
  name: ui
  namespace: turnkey
spec:
  template:
    metadata:
      labels:
        turnkey/workloadselector: job-turnkey-ui
    spec:
      initContainers:
        - args:
          - scan
          image: mak3r/wifi:v0.0.10
```

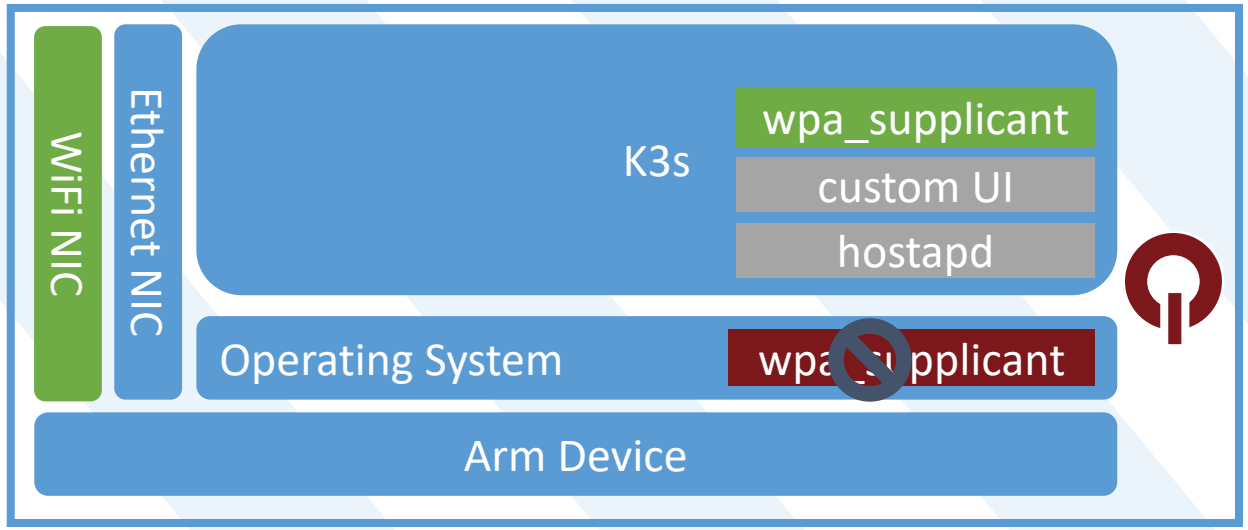
...

```yaml
      containers:
      - image: mak3r/turnkey-ui:v0.0.24
```



Submitting the configuration results in a K8s Deployment used to manage the WiFi NIC

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: wifi
  namespace: turnkey
spec:
  progressDeadlineSeconds: 600
  replicas: 1
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      turnkey/workloadselector: deployment-turnkey-wifi
  strategy:
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 0
    type: RollingUpdate
  template:
    metadata:
      labels:
        turnkey/workloadselector: deployment-turnkey-wifi
    spec:
      containers:
      - args:
        - up
        image: mak3r/wifi:v0.0.10
```



Resources for a Kubernetes Controller?

A more generic solution may provide a framework to

- Collect information from a non-technical individual
- Pull configuration from a remote location
- Install/Setup/Configure applications
- Configure / Modify the underlying system

Open Source

Want to contribute to the Turnkey project?

Do you have a need for a turnkey solution?

I’d love to hear about your problem domain, challenges and requirements.

twtr: AbramsMark

github: mak3r

Enter to Win

https://info.rancher.com/skilup-pi

Thank You!

https://github.com/mak3r/turnkey

https://github.com/mak3r/steer-case

https://github.com/rancher/system-upgrade-controller

https://github.com/rancher/k3s

https://github.com/rancher/fleet

https://github.com/rancher/rancher

---> https://info.rancher.com/skilup-pi <---