

Deploy to hardware

William Salmon

`will.salmon@codethink.co.uk`

Deploy to hardware

William Salmon

`will.salmon@codethink.co.uk`

Who am I?

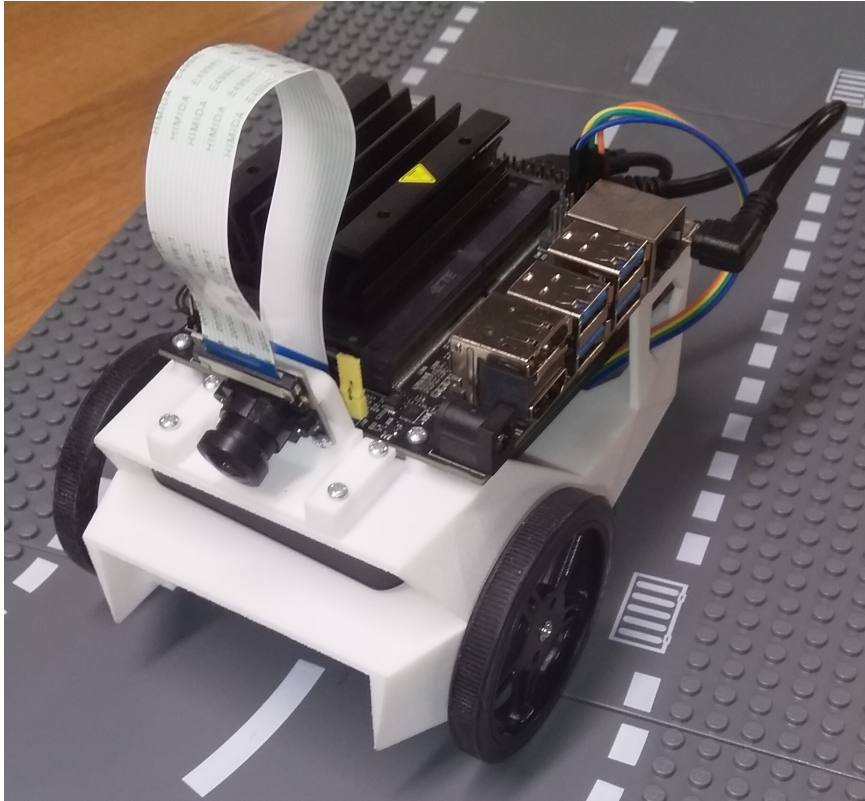
**What have I been
working on**

Celduin

The key parts for CRASH

- Promote open tools
- Promote Reproducible & trustable workflows
- Fast builds

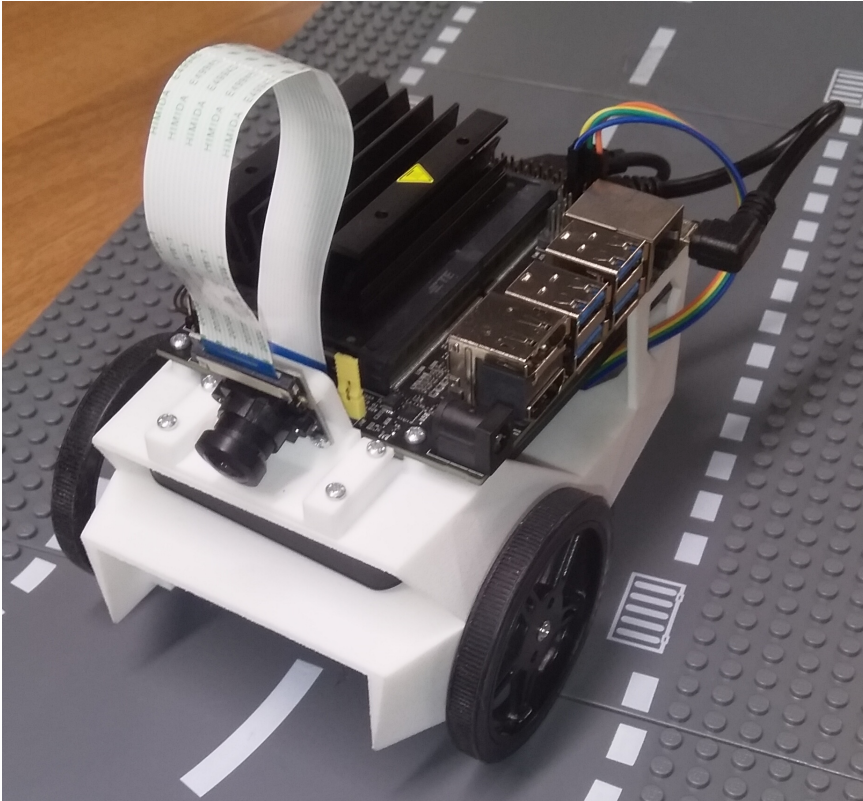
CRASH



Celduin Robotic Automation Showcase

[Hosted on gitlab.com](https://gitlab.com)

Systems integration



Testing and Deployment

- Try to test what we are going to deploy as early as possible
- Trade off between feed back time and mirroring prod
 - Docker does a good job of this but is not suitable for our case
 - For embedded systems we can only get so close due to the importance of hardware
 - Sensors
 - Projects like those talk about earlier are very important for CI
 - But allowing devs to test on desk realistically is very valuable.

Systems integration

Tools:

- Package
 - Apt
 - Dnf
- Layered
 - Docker
- Full "disk image"/partition
 - Buildroot
 - Yocto
 - Buildstream

CI / CD with Hardware

What tools let us:

- Continuously integrate with hardware
- Continuously deploy with hardware

In a way that is Testable and reproducible

"A/B" Systems could reduce the burden on Deploy and Test

Deployment

Tool options

- Mender
- Lava
- OStree
- Aktualizr

OSTree

- OSTree uses a content addressable store so it only needs to download what has changed
- OSTree has commits, a little like docker layers
- It dose not do on board integration like apt/dnf, it just deploys the changed files!

Examples

- Flatpack
- Fedora Atomic/silverblue
- Aktualizr -- Uptane

OSTree on embedded

- Almost everything can be inside the OSTree deployment
- Diff based updates -> very fast updates
 - The fast update lends its self to the developer workflow
- The same docker like workflow of using the same OSTree commit / image for test and deploy
 - OSTree brings this to places were you care about things that docker ignores, kernel, drivers, etc
- When we do "incremental" updates, we have already integrated
 - OSTree increments do not need locally integrating

Building our OSTree commits

Integration tool can create the commits

But a dev can checkout a commit/deployment, tweak commit and push to a local test

This is much better than just tweaking with a files system as the commit knows all the files they changed

Even if the dev used a integration tool, they dont need to reflash the hole thing.

Dev Commit

Our team used OSTree in its developer workflow early on and found it very useful.

With A/B updates if you upload a broken system, then your build can fall back to the last state.

Multi pipe line

1. Support:

- Docker image creation
- Infra repo

2. Components:

- ML model training data
- ML model store
- App code

3. Integration:

- Integration configuration, a bst/yocto project
 - App tests in intergration tool

Integration pipeline

Celduin > CRASH > Jetbot System - Buildstream > Pipelines > #113703206

passed Pipeline #113703206 triggered 1 day ago by William Salmon

Delete

Merge branch 'willsalmon/systemdconfs' into 'master'

Systemd conf overLay

See merge request [!20](#)

5 jobs for **master** in 28 minutes and 11 seconds (queued for 1 second)

latest

2d866187 ...

No related merge requests found.

Pipeline Jobs **5** Tests **0**



Integration pipeline for master

Key steps:

- Build
- Test
- Push to OSTree
- Push disk image

Building this set up

<https://gitlab.com/celduin/crash>

Most of the CI uses gitlab but could use github+CI or similar

- Docker image registry
- Gitlab runner with docker-machine driver creates
 - GPU runners
 - Arm runners
 - Multi size and privileges x86

What does gitlab not have?

- Note: different trade offs for different git servers/CI setups
- Controlled artifacts
 - All our code/elements are opensource but our sources and imports from hardware vendors are not
- Runner bastion
- Ostree server
 - Git pages
- Build cache

Our solution for a OSTree server

Our POC

- Docker running nginx
- Docker with rsync
- This will only scale so well! But as almost everything is immutable it caches VERY well
 - almost all of the OSTree content is a **Content address able store**
- Scaling up is clearly possible as flatpack has demonstrated.

Notes / Summary

I am pleased with our software and its stack

I am pleased with how we build our software

I am pleased with how we integrate our software

I am please with our OTA infra as a POC but would not like to make any promises about it.

Looking forward

- Cache
- REAPI -- Remote Execution API
- OSTree tooling and server
 - Currently there are some implementation but they are very specific
 - The generic tooling is powerful but requires a lot of boiler plate

Any Questions