Industrial-grade Open Source Base Layer Development

Yoshitake Kobayashi, Toshiba Corp.
Urs Gleim, Siemens AG

Embedded Linux Conference Europe, Prague, October 24, 2017
What is CIP?
What is CIP?

- One of the most conservative open source project in the Linux Foundation
- One of the most important projects for our civilization
What is CIP?

- One of the most conservative open source project in the Linux Foundation

- CIP aims to
  - Provide an open source base layer for CIP related embedded systems
  - Work closely with the upstream community

- CIP does not aim to
  - Create a new Linux distribution
Our Civilization is run by Linux
There are issues to be solved…


ELCE17, Prague, Czech Republic
A Railway System:

25-50 years products life-cycle

with very reluctant nature for product update and upgrade of hardware and base software platform
Railway Example

3 – 5 years development time

2 – 4 years customer specific extensions

1 year initial safety certifications / authorization

3 – 6 months safety certifications / authorization for follow-up releases (depending on amount of changes)

25 – 50 years lifetime

Image: http://www.deutschebahn.com/contentblob/10862328/20160301+Stw+M%C3%BClheim+Innenansicht+1+(1)/data.jpg, ELCE17, Prague, Czech Republic
Power Plant Control Example

- 3 – 5 years development time
- 0.5 – 4 years customer specific extensions
- 6 – 8 years supply time
- 15+ years hardware maintenance after latest shipment
- 20 – 60 years product lifetime

Image: http://zdnet1.cbsistatic.com/hub/i/r/2016/02/29/10853777-39b2-40c0-9d8c-dbba5feb65be/resize/770xauto/490141cef9bddd0db66b492698b53a56/powerplant.jpg
Industrial IoT: Edge and Fog Computing

Functionality is moving from the cloud to the “Edge”

- Increasing number of networked industrial-grade devices
- Security management requires harmonized software landscape

Application examples on IIoT infrastructure

- Plant analytics
- SCADA functionality
- Plant (device) mgmt.
- Local / real-time analytics
- IoT Gateways

Data collection
- Pre-processing
- Sensor / actor connectivity

IoT: Internet of Things  IIoT: Industrial IoT  SCADA: Supervisory Control And Data Acquisition
ELCE17, Prague, Czech Republic
Requirements for the Civil infrastructure systems

- **Industrial Grade**
  - Reliability
  - Functional Safety
  - Security
  - Real-time capabilities

- **Sustainability**
  - Product life-cycles of 10 – 60 years

- **Conservative Upgrade/Update Strategy**
  - Firmware updates only if industrial-gradeness is jeopardized
  - Minimize the risk of regressions
  - Keeping regression test and certification efforts low

This has to be achieved with ...

**Maintenance costs**
- Low maintenance costs for commonly uses software components
- Low commissioning and update costs

**Development costs**
- Don’t re-invent the wheel

**Development time**
- Shorter development times for more complex systems

ELCE17, Prague, Czech Republic
The Problems we face …

• The systems that support our modern civilization need to survive for a VERY LONG TIME. Until now the corresponding industrial grade super long term maintenance has been done individually by each company.

• These systems not only have to survive for a long time, they must be “INDUSTRIAL GRADE” (robust, secure and reliable). And at the same time the industry will also need to catch up with the latest technology trends.
The genesis of a collaborative project
Linux Foundation Projects

Driving joint efforts and backing them with people and budget.

The majority focusses in IT, enterprise, cloud technologies.

ELCE17, Prague, Czech Republic
The Solutions we need ...

- **We need a Collaborative framework** to maintain the same open source based system for many, many, many years to keep it secure, robust and reliable.

- AND most importantly, we need to do this collaboratively in the **upstream communities**, not locally.
CIP is our solution...

Establishing an **Open Source Base Layer** of industrial-grade software to enable the use and implementation of software building blocks for Civil Infrastructure Systems

[https://www.cip-project.org/](https://www.cip-project.org/)

since April 2016
The backbone of CIP are the member companies

Open source projects (Upstream work)

Budget

Developers, maintainers

Optional: funding of selected projects

Contribution & usage / integration

CIP Super Long Term Support Project

CIP source code repositories

18
What is CIP, again?
What is “Open Source Base Layer (OSBL)”?

- OSBL is a set of industrial grade core open source software components, tools and methods

- Open source based reference implementation
- Start from a minimal set for controllers in industrial grade systems

Non-CIP packages
Linux distribution (e.g. Debian) may extend/include CIP packages.

CIP Reference Filesystem image with SDK (CIP Core packages)

CIP SLTS Kernel

CIP Reference Hardware

User space
Kernel
Hardware

ELCE17, Prague, Czech Republic
Development plan

CIP will increase the development effort to create an industrial grade common base-layer.

Phase 1:
- Define supported kernel subsystems, arch.
- Initial SLTS component selection
- Select SLTS versions
- Set-up maintenance infrastructure (build, test)

Phase 2:
- Patch collection, stabilization, backport of patches for CIP kernel packages
- Support more subsystems
- Additional core packages

Phase 3:
- Domain specific enhancements, e.g. communication protocols, industrial IoT middleware
- Optionally: more subsystems
- Optionally: more core packages

Core Packages

Kernel (SLTS)

add. pkgs

Core Packages

Kernel (SLTS)

add. pkgs

Core Packages

Kernel (SLTS)
## Vision: Technical topics and related projects

*Topics will be added or removed to reflect CIP technical interests*

### Application support
- App Framework
- HMI Framework
- FW update
- App deployment

### Middleware / Tools
- Coherent Security Mechanisms
- Configuration
- Device management
- Industrial Zeroconf
- LXC
- Cgroups

### Security
- Anomaly detection
- Live patching
- LSM
- SELinux

### Functional Safety
- Monitoring/error detection
- SIL2LinuxMP
- SIL3 support

### Userland Isolation
- LXC
- Cgroups

### Kernel Isolation
- SafeG
- Jailhouse

### Communication
- RTOS
- PREEMPT-RT

### Real-time support
- GPGPU/FPGA real-time
- Xenomai

### Monitoring / Tracing
- Ftrace
- ktap
- RAS

### Heterogeneous Computing
- SoC FPGA

### Linux Kernel
- OPC UA
- TSN
- Echonet
- OM2M
- Industrial special-purpose protocols
- Avnu

### Middleware / Tools
- Infrastructure and Services
- Build and production
  - Yocto Project
  - Debian build system
- Testing
  - LTP
  - kselftest
  - KernelCI
  - Fuego
- Support
  - SLTS
  - Backwards compatibility
- Development process
  - SIL2 support
  - SIL3 support
- Legal topics
  - SPDX
  - FOSSology
- License Clearing
  - Export Control

### Hardware / SoC (x86 or ARM based)

### Legend
- To be specified / implemented by CIP
- Integration / cooperation

---

ELCE17, Prague, Czech Republic

22
CIP activities and status
Announcements

• CIP testing project released B@D v1.0
• CIP Core project launched
• CIP decided to take Debian as a primary reference distribution
Scope of activities

User space
- App container infrastructure (mid-term)
- App Framework (optionally, mid-term)
- Domain Specific communication (e.g. OPC UA)
- Shared config. & logging
- Multimedia

Middleware/Libraries
- CIP Core Packages
- Safe & Secure Update
- Monitoring
- Security
- Real-time support
- Real-time / safe virtualization

Linux Kernel
- Super Long Term Supported Kernel (STLS)

Tools
- Build environment (e.g. bitbake, dpkg)
- Test automation
- Tracing & reporting tools
- Configuration management
- Device management (update, download)
- Application life-cycle management

Concepts
- Functional safety architecture/strategy, including compliance w/ standards (e.g., NERC CIP, IEC61508)
- Long-term support Strategy: security patch management
- Standardization collaborative effort with others
- License clearing
- Export Control Classification

Kernel space

On-device software stack

Product development and maintenance

ELCE17, Prague, Czech Republic
CIP Activities

1 Kernel maintenance
   • The first action taken by the CIP project is to select and maintain Linux kernels for very long time (+15 years). To achieve goal a group of experts has been assigned.

2 Testing
   • PREEMPT_RT patches are added to the CIP kernel

3 Testing
   • Civil infrastructure industry has high stability, reliability and security standards in order to ensure safety critical systems. The CIP Testing project has been formed to address this reality. So far the efforts are focused on testing the CIP kernel. In the future they will be extended to the complete CIP platform.

4 CIP Core
   • This project focus to create reference minimal file system images that allow testing the CIP Core packages: a set of industrial-grade components that require super long-term maintenance.
CIP SLTS Kernel development (1/5)

CIP SLTS (linux-4.4.y-cip), Maintenance period 10 years and more (10-20 years)

- Official CIP SLTS kernel tree based on linux-stable.git
- Maintainer: Ben Hutchings
- Linux 4.4.92-cip11 released on 18th October 2017
CIP SLTS Kernel development (2/5)

• Kernel maintenance policy
  • [https://wiki.linuxfoundation.org/civilinfrastructureplatform/cipkernelmaintenance](https://wiki.linuxfoundation.org/civilinfrastructureplatform/cipkernelmaintenance)
  • Follow the stable kernel development rule as the basis
  • Feature backports are acceptable
    • All features has to be in upstream kernel before backport to CIP kernel
  • CIP has “Upstream first” policy
  • Validation will be done by CIP test infrastructure and/or members

• Current backported features on 4.4.y-CIP
  • Kernel Self Protection Project related features
    • Address Space Layout Randomization for user space process (ASLR)
    • GCC’s undefined behaviour Sanitizer (UBSAN)
    • Faster page poisoning
  • Board support patches for Renesas RZ/G and Siemens IoT2000 series
4.4-stable review patch. If anyone has any objections, please let me know.
------------------
From: Christoph Hellwig <hch@lst.de>
commit f507b54dccfd8000c517d740bc45f20c74532d18 upstream.

The job structure is allocated as part of the request, so we should not
free it in the error path of bsg_prepare_job.

Signed-off-by: Christoph Hellwig <hch@lst.de>
Reviewed-by: Ming Lei <ming.lei@redhat.com>
Signed-off-by: Jens Axboe <axboe@kernel.dk>
Signed-off-by: Greg Kroah-Hartman <gregkh@linuxfoundation.org>

---
block/bsg-lib.c | 1 -
1 file changed, 1 deletion(-)

--- a/block/bsg-lib.c
+++ b/block/bsg-lib.c
@@ -147,7 +147,6 @@ static int bsg_create_job(struct device
failjob_rls_rqst_payload:
    kfree(job->request_payload.sg_list);
failjob_rls_job:
-    kfree(job);
+    return -ENOMEM;
}
CIP SLTS Kernel development (4/5)

Next CIP SLTS kernel (tbd)

Mainline

4.4

Stable (linux-stable)

CIP SLTS (linux-4.4.y-cip)

Stable (linux-stable-x.y)

NEXT CIP SLTS (TBD)

Backported patches

Maintained by Ben Hutchings

Approx. 2-3 years

Stop backporting. Focus to security fix only

Take over from maintainer

Feature backports

CIP will pick up next version from stable tree

Backported patches

ELCE17, Prague, Czech Republic
Out-of-tree drivers

• In general, all out-of-tree drivers are unsupported by CIP

• Users can use CIP kernel with out-of-tree drivers
  • If a bug is found in such a modified kernel, users will first demonstrate that it exists in the CIP kernel source release in order for the CIP maintainers to act on it.
2 CIP SLTS real-time support (1/2)

CIP SLTS+PREEMPT_RT (will be separately maintained by CIP members)
• CIP kernel tree based on linux-stable-rt and patches from CIP SLTS
• Validation will be done by CIP
• CIP RT is currently under development at the following URL
  • https://github.com/igaw/linux-cip-rt
• CIP has become a Gold Member of the Real Time Linux Project

• What’s next
  • Work together with the RTL Project
  • A CIP member is working to become the maintainer of 4.4.y-stable-rt, the base version of the CIP Kernel.

• More information
  • https://wiki.linuxfoundation.org/realtime/rtl/start
CIP testing (1/3)

Milestones of CIP testing and current status

1. Board at desk - single dev
   • A setup that allows a developer to test the CIP kernel on the CIP selected hardware platform connected locally to her development machine using kernelCI tools.

2. CIP kernel testing
   • Test the CIP kernel on a regular basis and share the results with other CIP community members.

3. Define kernel testing as a service within CIP
   • Define the testing environment within CIP assuming that, in some cases, some members may share the tests, test results or laboratories while others may not.

4. From kernel testing to system testing
   • Once the testing environment has been ready and works for the kernel, explore how to extend it to the entire CIP platform.

https://wiki.linuxfoundation.org/civilinfrastructureplatform/ciptesting
CIP testing (2/3)

• CIP Testing project
  (https://wiki.linuxfoundation.org/civilinfrastructureplatform/ciptesting)

• B@D designed to:
  • Test Linux kernels and base systems.
  • Locally: no need of a centrally managed service.
  • On hardware connected to your dev machine.

• Latest status
  • CIP testing environment (B@D v1.0) just released
    (https://goo.gl/4RFrJ1)
    • Based on kernelci.org
    • Linux and Windows 10 as Host OS supported.
    • Shipped as a VM and Vagrant based environment.
    • Results and logs sharing capabilities.

• Check the source code involved
  • https://gitlab.com/cip-project/cip-testing/board-at-desk-single-dev/tree/master
Next Steps

• Collaboration with other testing effort
  • CIP had a meeting with AGL members for testing collaboration

• During the coming months the team will focus on:
  • Defining how tests should look like.
  • Defining how results should be shared.
  • Increasing the test coverage of the CIP Kernel
Debian as CIP primary reference distribution

• What does the primary distribution means?
  • CIP will select CIP Core package from Debian packages
  • CIP would like to work with Debian community

• CIP members also interested in Yocto Project as a build tool
  • CIP might create meta-cip layer
    • Users can get SLTS benefit from CIP Core packages
    • Other OE-layers could be extend CIP Core (Will not SLTS by CIP)
### CIP Core Packages (1/5)

An example of minimal package set for CIP base layer

**Candidates for initial component set**

<table>
<thead>
<tr>
<th>CIP Kernel</th>
<th>CIP Core Packages</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• Kernel</td>
</tr>
<tr>
<td></td>
<td>• Linux kernel + backported patches</td>
</tr>
<tr>
<td></td>
<td>• PREEMPT_RT patch</td>
</tr>
<tr>
<td></td>
<td>• Bootloader</td>
</tr>
<tr>
<td></td>
<td>• U-boot</td>
</tr>
<tr>
<td></td>
<td>• Shells / Utilities</td>
</tr>
<tr>
<td></td>
<td>• Busybox</td>
</tr>
<tr>
<td></td>
<td>• Base libraries</td>
</tr>
<tr>
<td></td>
<td>• Glibc</td>
</tr>
<tr>
<td></td>
<td>• Tool Chain</td>
</tr>
<tr>
<td></td>
<td>• Binutils</td>
</tr>
<tr>
<td></td>
<td>• GCC</td>
</tr>
<tr>
<td></td>
<td>• Security</td>
</tr>
<tr>
<td></td>
<td>• OpenSSL</td>
</tr>
</tbody>
</table>

**Dev packages**

<table>
<thead>
<tr>
<th>Flex</th>
<th>Git</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bison</td>
<td>Glib</td>
</tr>
<tr>
<td>autoconf</td>
<td>Gmp</td>
</tr>
<tr>
<td>automake</td>
<td>Gzip</td>
</tr>
<tr>
<td>bc</td>
<td>gettext</td>
</tr>
<tr>
<td>bison</td>
<td>Kbd</td>
</tr>
<tr>
<td>Bzip2</td>
<td>Libibverbs</td>
</tr>
<tr>
<td>Curl</td>
<td>Libtool</td>
</tr>
<tr>
<td>Db</td>
<td>Libxml2</td>
</tr>
<tr>
<td>Dbus</td>
<td>Mpclib</td>
</tr>
<tr>
<td>Expat</td>
<td>Mpfr4</td>
</tr>
<tr>
<td>Flex</td>
<td>Ncurses</td>
</tr>
<tr>
<td>gawk</td>
<td>Make</td>
</tr>
<tr>
<td>Gdb</td>
<td>M4</td>
</tr>
</tbody>
</table>

**Keep these packages for Reproducible build**

<table>
<thead>
<tr>
<th>Flex</th>
<th>Git</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bison</td>
<td>Glib</td>
</tr>
<tr>
<td>autoconf</td>
<td>Gmp</td>
</tr>
<tr>
<td>automake</td>
<td>Gzip</td>
</tr>
<tr>
<td>bc</td>
<td>gettext</td>
</tr>
<tr>
<td>bison</td>
<td>Kbd</td>
</tr>
<tr>
<td>Bzip2</td>
<td>Libibverbs</td>
</tr>
<tr>
<td>Curl</td>
<td>Libtool</td>
</tr>
<tr>
<td>Db</td>
<td>Libxml2</td>
</tr>
<tr>
<td>Dbus</td>
<td>Mpclib</td>
</tr>
<tr>
<td>Expat</td>
<td>Mpfr4</td>
</tr>
<tr>
<td>Flex</td>
<td>Ncurses</td>
</tr>
<tr>
<td>gawk</td>
<td>Make</td>
</tr>
<tr>
<td>Gdb</td>
<td>M4</td>
</tr>
<tr>
<td>pax-utils</td>
<td></td>
</tr>
<tr>
<td>Pciutils</td>
<td></td>
</tr>
<tr>
<td>Perl</td>
<td></td>
</tr>
<tr>
<td>pkg-config</td>
<td></td>
</tr>
<tr>
<td>Popt</td>
<td></td>
</tr>
<tr>
<td>Procs</td>
<td></td>
</tr>
<tr>
<td>Quilt</td>
<td></td>
</tr>
<tr>
<td>Readline</td>
<td></td>
</tr>
<tr>
<td>sysfsutils</td>
<td></td>
</tr>
<tr>
<td>Tar</td>
<td></td>
</tr>
<tr>
<td>Unifdef</td>
<td></td>
</tr>
<tr>
<td>Zlib</td>
<td></td>
</tr>
</tbody>
</table>

**NOTE:** The maintenance effort varies considerably for different packages.
Current status of the Base layer development

1. Define an initial component set
   1.5 Talk to open source communities
2. Define component version
3. Contribute to upstream project
4. Start maintenance for SLTS
CIP Core

- **CIP Core is now become CIP official project**
  - CIP Core aims to provide a way to create and test installable images

- **Goal**
  - **Input**: Debian sources/binaries and cip kernel
  - **Build mechanism**: Bitbake and/or Debian build system
  - **Output**: Minimum deployable base system image for testing

- **Current status**
  - Minimal rootfs can be build for the following hardware
    - Renesas RZ/G1M (iwg20m)
    - BeagleBone Black
    - Cyclone-V
    - QEMUx86

Source code: [https://gitlab.com/cip-project/cip-core](https://gitlab.com/cip-project/cip-core)
Creating Debian-based image (Currently supported)

Deby: [https://github.com/meta-debian/meta-debian](https://github.com/meta-debian/meta-debian)
Creating Debian-based image (Other options)

ISAR: https://github.com/ilbers/isar

ELBE: https://elbe-rfs.org/

CIP Core Packages (5/5)
### Potential build tools for CIP Core (Comparison Elbe, Isar and Deby)

<table>
<thead>
<tr>
<th></th>
<th>Elbe</th>
<th>Isar</th>
<th>Deby</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Base system</strong></td>
<td><strong>Debian binary packages (no rebuilding)</strong></td>
<td></td>
<td>Binary packages <strong>cross-built</strong> from Debian source packages</td>
</tr>
<tr>
<td><strong>Build system</strong></td>
<td><strong>Custom</strong></td>
<td><strong>Bitbake</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Host tools</strong></td>
<td><strong>Debian</strong>: debootstrap, qemu, elbe-pbuilder</td>
<td><strong>Debian</strong>: multistrap, dpkg-buildpackage, qemu</td>
<td><strong>Poky</strong></td>
</tr>
<tr>
<td><strong>Metadata</strong></td>
<td>✓ ELBE-XML for project description</td>
<td>✓ Recipes for building product packages</td>
<td>✓ Common function to unpack Debian source packages</td>
</tr>
<tr>
<td></td>
<td></td>
<td>✓ Recipes for image generation</td>
<td>✓ <strong>Full recipes</strong> for cross-building every Debian source package</td>
</tr>
<tr>
<td><strong>Compilation</strong></td>
<td></td>
<td></td>
<td>✓ Affinity with Poky recipes</td>
</tr>
<tr>
<td><strong>Benefits</strong></td>
<td>✓ <strong>Re-use Debian binaries and QA</strong></td>
<td></td>
<td>✓ <strong>Fully customizability</strong></td>
</tr>
<tr>
<td></td>
<td>✓ Fast (re-use, parallel builds)</td>
<td></td>
<td>✓ No need to keep binary pkgs</td>
</tr>
<tr>
<td></td>
<td>✓ <strong>Lower development costs</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Common features</strong></td>
<td>✓ Based on Debian packages (stability, long-term maintenance)</td>
<td>✓ Generate images by installing binary packages</td>
<td>✓ Manage multiple products as a custom setting (layers or configuration files)</td>
</tr>
</tbody>
</table>

## Gaps and Common Goals between Debian and CIP

<table>
<thead>
<tr>
<th>Debian</th>
<th>CIP requires</th>
<th>Chance to collaborate with Debian</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Support</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Term: 3+2 years by Debian-LTS</td>
<td>- Term: 10+ years</td>
<td>- <strong>Longer</strong> term maintenance for limited number of packages</td>
</tr>
<tr>
<td>- Num of pkgs: 67776</td>
<td>- Num of pkgs: 10+ (minimum)</td>
<td></td>
</tr>
<tr>
<td><strong>Build</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Should support native build</td>
<td>- Need to have both native and cross build</td>
<td>- Contributing to <strong>Debian-cross</strong></td>
</tr>
<tr>
<td>- Working on cross build packaging (Debian-cross)</td>
<td>- Binary / Source code should be managed and reproducible</td>
<td></td>
</tr>
<tr>
<td>- Reproducible build</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>OSS license compliance</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- DEP-5 adoption is ongoing</td>
<td>- Generate reports automatically</td>
<td>- Exchange and share the <strong>license review results</strong></td>
</tr>
<tr>
<td><strong>Testing</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Packages has to be tested</td>
<td>- Easy to redistribute</td>
<td>- Contributing <strong>test cases</strong> to upstream</td>
</tr>
<tr>
<td>- autopkgtest</td>
<td>- All packages should be tested in timely manner</td>
<td></td>
</tr>
</tbody>
</table>
What’s currently under discussion in CIP

• Functional safety
• Security standards for industry
  • E.g. IEC62443-4
• Y2038
Summary and conclusion
Summary

• The CIP Open Source Base Layer of industrial-grade software materializes

• CIP today focusses on

  • **Kernel maintenance**: maintaining Linux kernels for very long time (+15 years) including real-time support
  
  • **Testing**: providing a test infrastructure and evolve tests
  
  • **CIP Core packages**: a set of industrial-grade components that require super long-term maintenance including the required build tool chains
Conclusion

• Our Civilization needs an Open Source Base Layer of industrial-grade software
• CIP provides this, based on Linux
• Sustainability is ensured by
  • The backing of big industrial and semiconductor companies
  • Close cooperation with and build on mature Open Source projects (Debian, PREEMPT_RT, kernelci, ...)
  • Providing elaborated tool chains
  • Ensuring in-depth tests
• CIP gets traction in the member companies
CIP meeting and presentations @ ELCE

• Tuesday, October 24\textsuperscript{th}, 13:00-14:00 (Immediately after this talk)
  CIP developers meeting/gathering at ELCE (Liben Room, Mezzanine Level)

• Tuesday, October 24, 16:55 - 17:35 (Congress Hall II)
  Maintaining a Linux Kernel for 13 Years? You Must be Kidding Me. We Need at Least 30?,
  Agustin Benito Bethencourt & Ben Hutchings (Codethink Ltd)

• Wednesday, October 25, 09:50 - 10:10 (Congress Hall)
  Keynote: Challenges in Industrializing OSS and How Siemens Tackles Them,
  Jan Kiszka (Siemens AG)
Need more information?

- Please come to CIP booth!
  - Location: Mezzanine Level

- Demo contents
  - **CIP RT kernel** (Renesas RZ/G)
  - **Industrial IoT** (Siemens IoT2020)
  - **IoT sensors** (Plat’Home OpenBlocks)
  - **B@D v1.0** (at 3:00pm-4:30pm)
Contact Information and Resources

To get the latest information, please contact:

• CIP Mailing list: cip-dev@lists.cip-project.org

Other resources

• CIP Web site: https://www.cip-project.org
• CIP Wiki: https://wiki.linuxfoundation.org/civilinfrastructureplatform/
• CIP source code
  • CIP GitLab: http://www.gitlab.com/cip-project
  • CIP kernel: git://git.kernel.org/pub/scm/linux/kernel/git/bwh/linux-cip.git
CIP whitepaper

• Year One Update + Whitepaper Release
  • [https://www.cip-project.org/blog/2017/05/31/cip-year-one-update-whitepaper-release](https://www.cip-project.org/blog/2017/05/31/cip-year-one-update-whitepaper-release)

• Everyone can download the whitepaper
  • [https://wiki.linuxfoundation.org/__media/civilinfrastructureplatform/whitepaper_short.pdf](https://wiki.linuxfoundation.org/__media/civilinfrastructureplatform/whitepaper_short.pdf)
Thank you!
Questions?