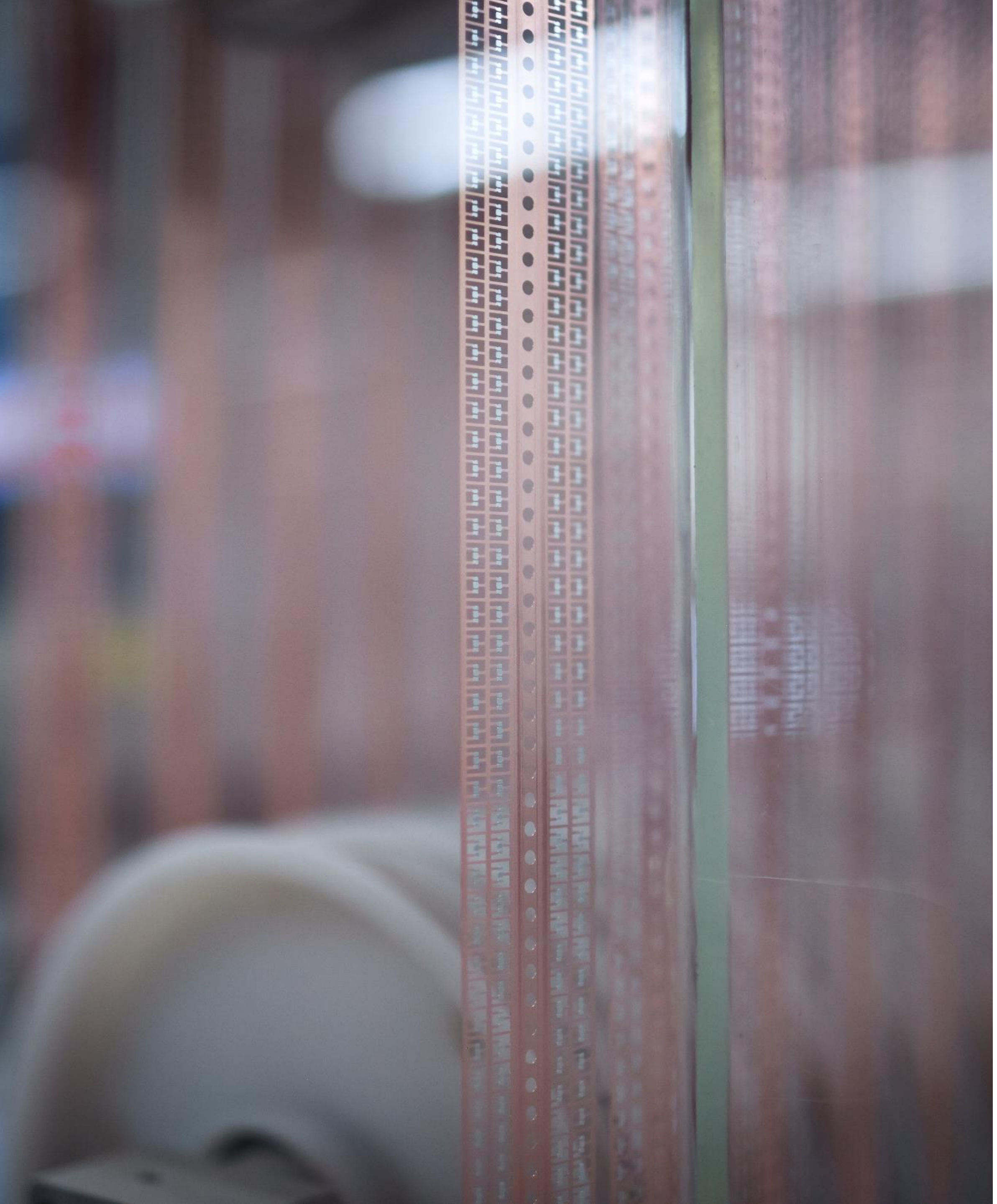
HOW TO INTEGRATE NFC CONTROLLERS IN LINUX

WEBINAR SERIES:
NFC SOFTWARE INTEGRATION

JORDI JOFRE
NFC READERS
NFC EVERYWHERE
28/09/2017

PUBLIC

SECURE CONNECTIONS
FOR A SMARTER WORLD



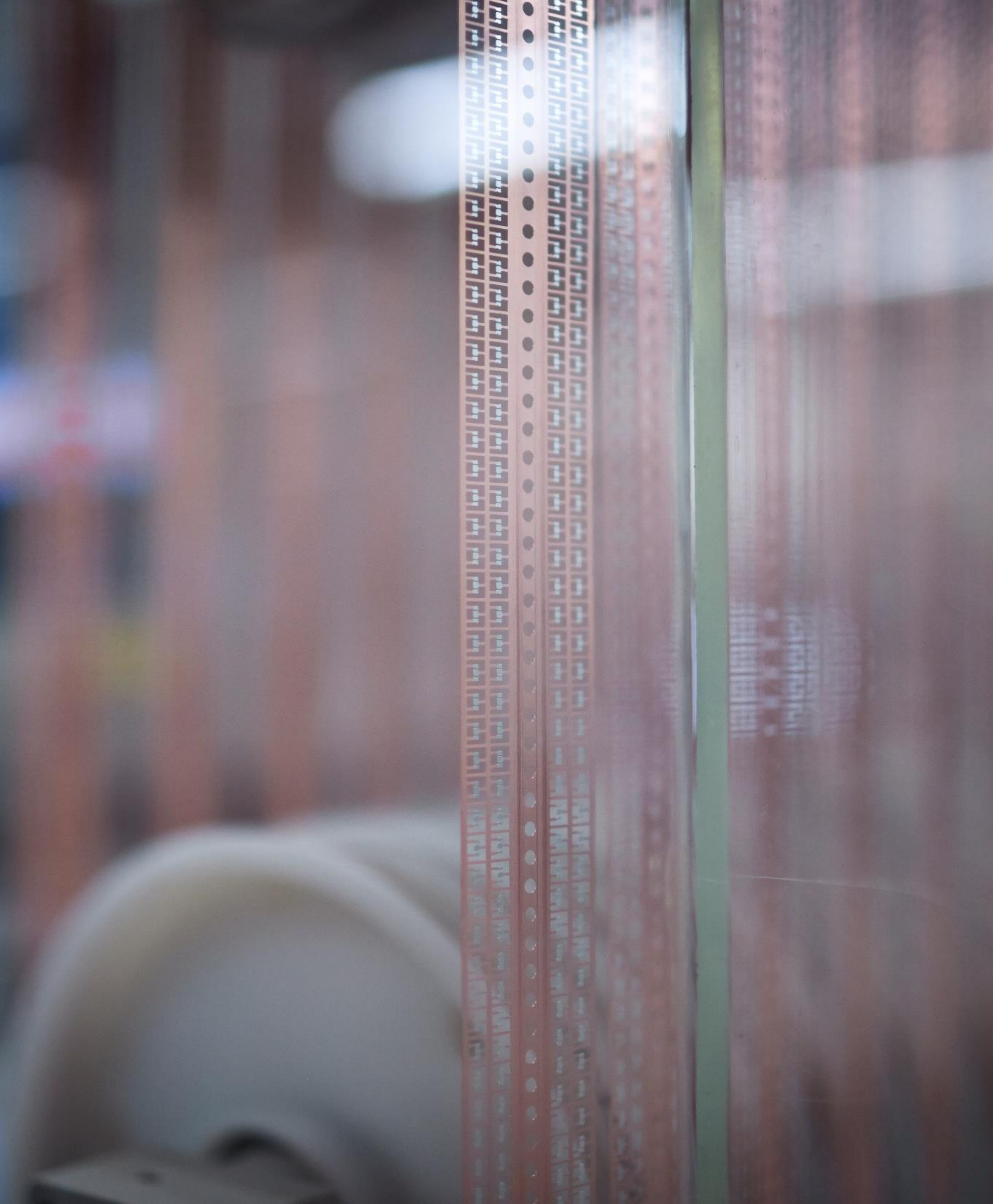
Agenda

NFC software integration webinar series

Session I, 14th September
How to integrate NFC frontends in Linux.

Session II, 28th September
How to integrate NFC controllers in Linux.

Session III, 11th October
How to port the NFC Reader Library to K64F.



Agenda

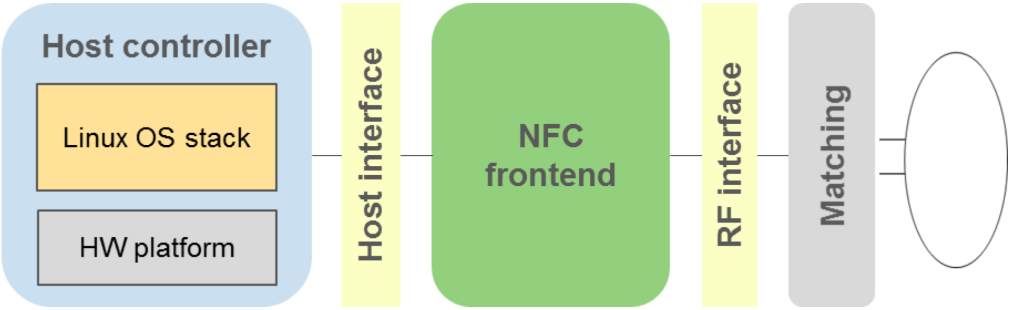
NFC software integration webinar series

Session III, 11th October
How to integrate NFC controllers in Linux.

- Recap about last session
- PN7150 NFC controller family
  - PN7150 NFC controller concept
  - NFC software libraries available for Linux
  - Integration in Linux using the Linux libnfc-nci SW stack
- PN7462 NFC controller family
  - PN7462 NFC controller concept
  - PN7462 NFC integration in Linux
- Wrap up and Q&A

Recap about last session

NFC frontends software integration in Linux

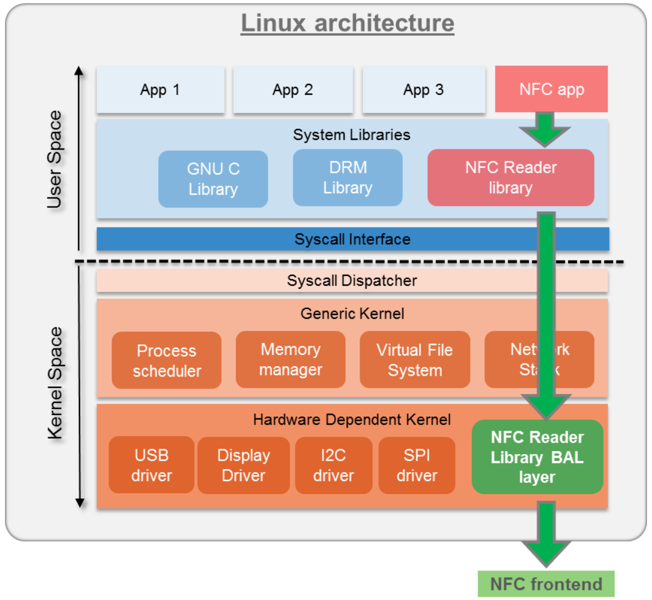


- NFC Frontends expose a ‘register interface’ towards the host controller through the host interface

- The NFC Reader Library can be installed on a generic GNU/Linux platform.

- **High latency:** Access the host interface (I^{2}C, SPI, UART drivers) in Linux Kernel space is slow.
- **High CPU load:** There is a lot of code involved just to write one register in Linux context switching.

Solutions



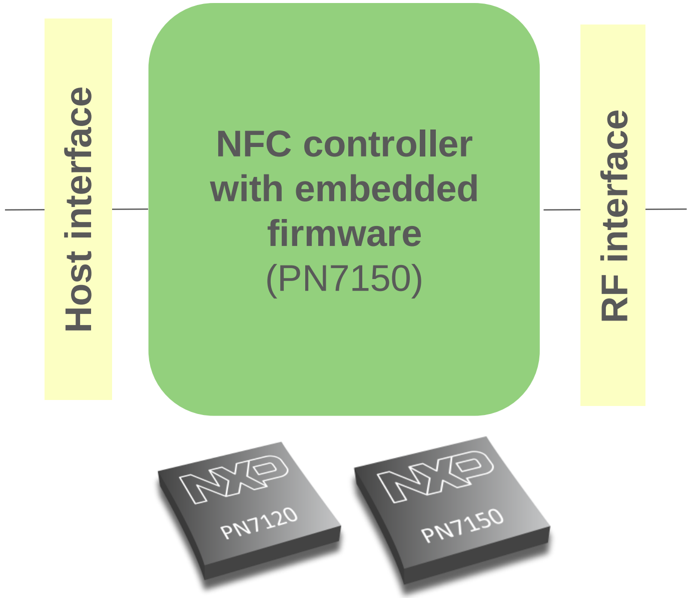
- Increase CPU/SPI clock as much as the MCU can process.
- Reduce SPI / host interface interactions as much as possible: Linux driver is optimized for few long transactions rather than lots of short ones
- **Move NFC Reader Library BAL module to Kernel space.**

PN7150 NFC controller family concept

PN7150 NFC controller family host and contactless interfaces

Host interface

- PN7150 is connected to the host device through an I^{2}C physical interface.
- PN7150 logical interface (API) is based on the NCI NFC Forum standard.
- It supports additional , NXP proprietary extensions specific to the PN7150 chip.
- In few NCI commands, host SW can configure the PN7150 to notify for card or peer detection and start data exchange.



Host interface

NFC controller with embedded firmware
(PN7150)

RF interface

RF interface

- Full NFC Forum compliancy with small form factor antenna.
- Ultra-low power consumption in polling loop mode allows without impacting the user experience.
- Embedded NFC firmware providing all NFC protocols as pre-integrated feature
- Load modulation schemes: Active & Passive

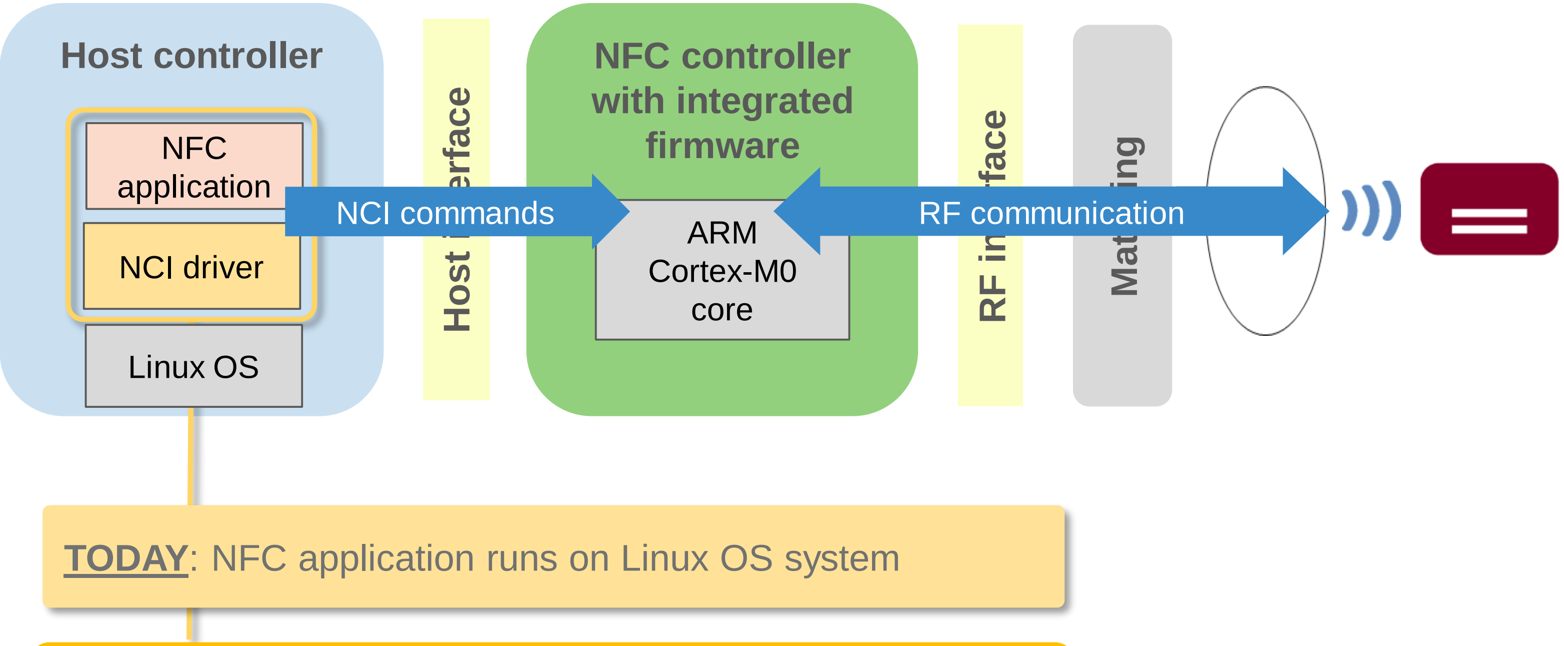
PN7150 NFC controller family are integrated solutions combining an NFC frontend together with an MCU

PN7150 NFC controller family are managed by NCI commands

Host controller

- SW implementation is required for both entities, the NFC FW for the controller side, and the NCI driver from the device host.
- NCI defines a standard communication channel between the NFC controller and the host device.
- It provides manufacturers with a standard interface they can use for whatever kind of NFC-enabled device they build.

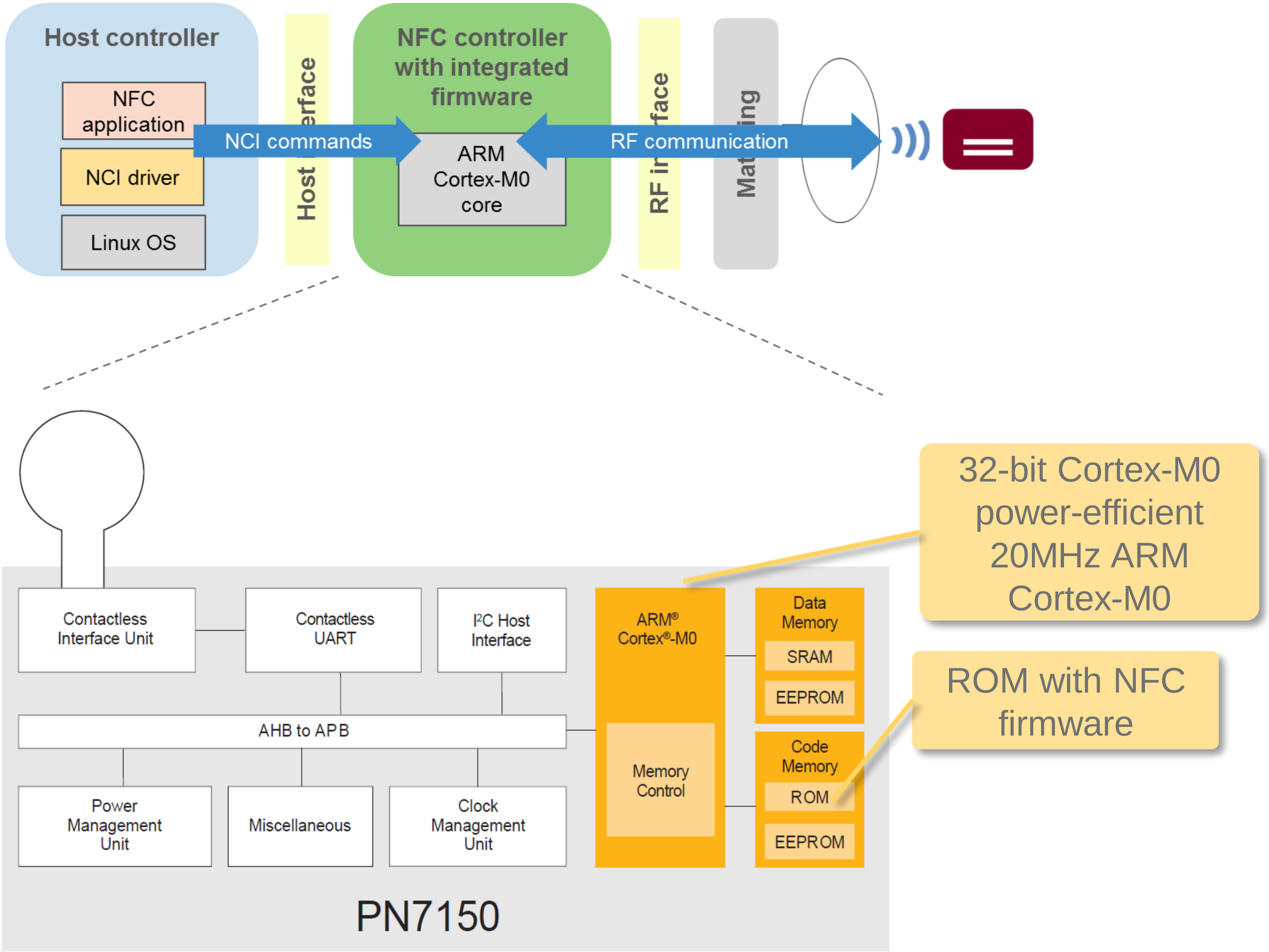
PN7150 NFC controller embeds an ARM Cortex-M0 microcontroller core loaded with integrated firmware supporting the NCI 1.0 host communication.



TODAY: NFC application runs on Linux OS system

There is NCI driver support for Linux, Android, Windows IoT, RTOS and bare metal systems.

PN7150 NFC controller family architecture



32-bit Cortex-M0 power-efficient 20MHz ARM Cortex-M0

ROM with NFC firmware

NFC controller with integrated FW

- Implements NCI 1.0 compliant protocol.
- Runs the NFC stack and RF protocols.

Key benefits

- Critical timing constraints on host are handled by the embedded FW.
- Host interactions are reduced since some functionalities are handled autonomously by the embedded FW.

PN7150 NFC software integration in Linux

NFC software libraries available for Linux

NFC software libraries available for Linux

M.U.S.C.L.E.
MOVEMENT FOR THE USE OF SMART CARDS IN A LINUX ENVIRONMENT

MUSCLE PC/SC Lite

- MUSCLE is a project to develop a set of compliant drivers, API's, and a resource manager for various smart cards and readers for the GNU environment.
- Middleware to access a smart card using the SCard API.
- Open source implementation of PC/SC (Microsoft OS).

→ Library focused on smartcard integration into computing environment rather than an NFC SW stack

nfcpy

- Implements NFC Forum specifications for data exchange with NFC devices and tags.
- Python module for NFC.
- USB and UART devices support.
- EUPL licensed project and available at GitHub.

← Not reusable for PN7150, NCI support is not included

Not all NFC functionalities implemented (e.g. Card emulation limited to Type 3 tags)

NFC software libraries available for Linux (II)

Libnfc

(From NFC Tools)

- Academic LGPL licensed project.
- Written in C.
- USB and UART devices support.
- Open source and community supported.
- Support for various operating systems (Linux, Mac OS, Windows).
- No ROM memory access on Linux.

Not reusable for PN7150 since NCI support is not included

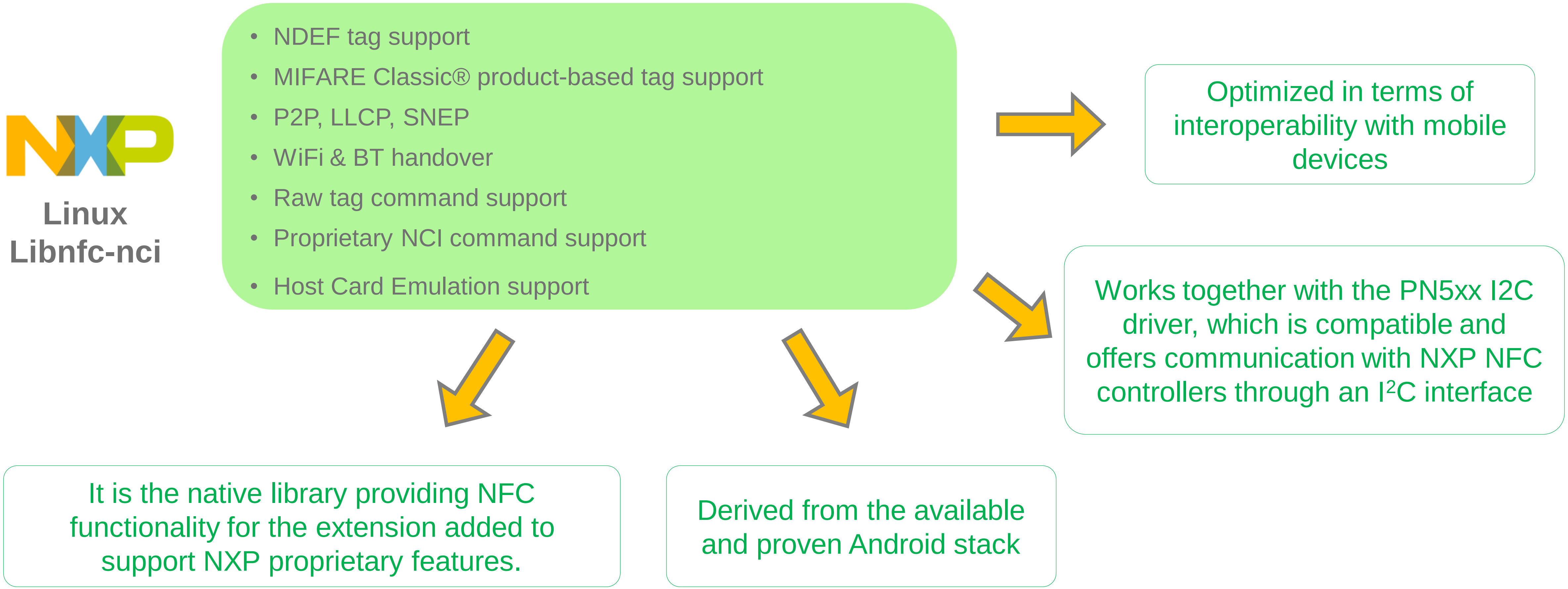
Linux NFC

- Maintained by Intel open source community.
- Aims to be HW independent (NXP, TI, Inside Secure, etc). Supports HCI, NCI and USB hardware.
- GPLv2 licensed.
- Split between kernel and user spaces.

Linux NFC

NXP originally contributed to this SW stack. Support has stopped for their NCI-based NFC controllers.

NFC software libraries available for Linux (III)



Linux Libnfc-nci

- NDEF tag support
- MIFARE Classic® product-based tag support
- P2P, LLCP, SNEP
- WiFi & BT handover
- Raw tag command support
- Proprietary NCI command support
- Host Card Emulation support

Optimized in terms of interoperability with mobile devices

Works together with the PN5xx I2C driver, which is compatible and offers communication with NXP NFC controllers through an I^2C interface

It is the native library providing NFC functionality for the extension added to support NXP proprietary features.

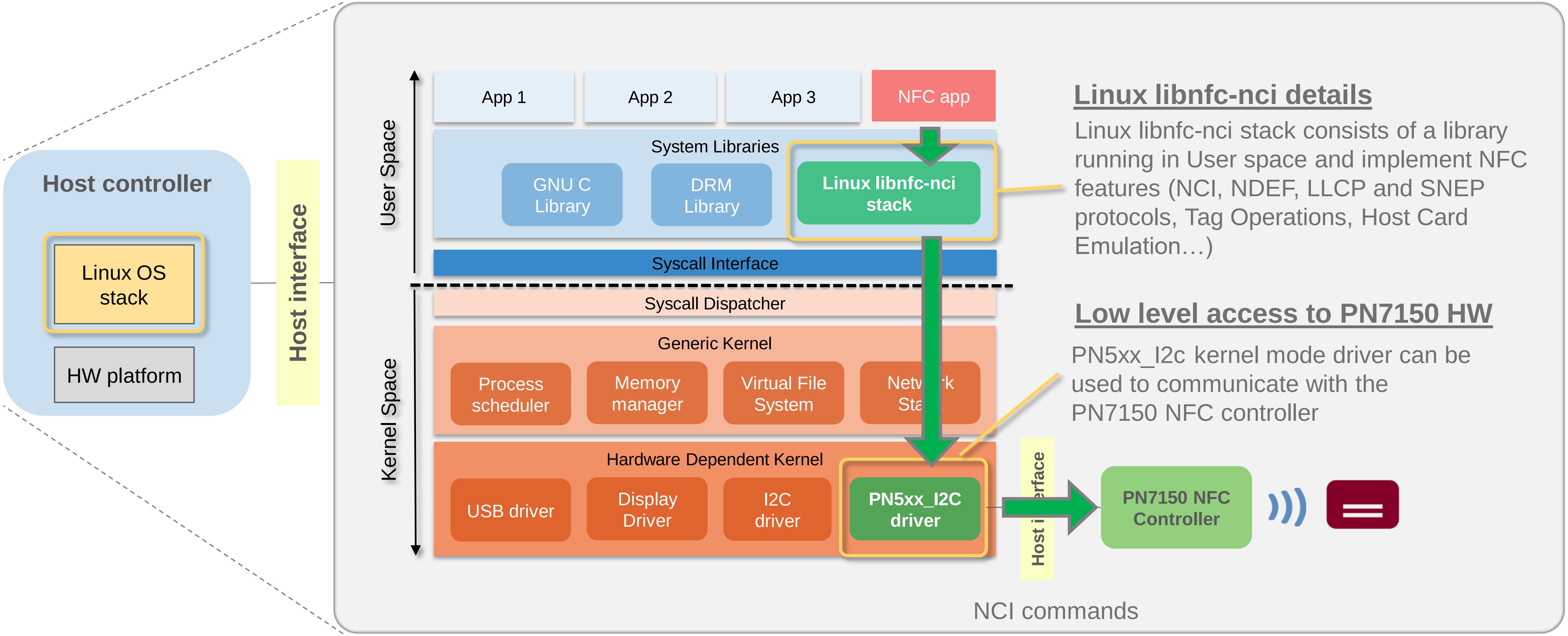
Derived from the available and proven Android stack

Linux Libnfc-nci is the best choice for integrating PN7150 NFC controllers in Linux

PN7150 NFC software intregration in Linux

Linux libnfc-nci stack

Linux libnfc-nci stack mapping in Linux architecture



Linux libnfc-nci details

Linux libnfc-nci stack consists of a library running in User space and implement NFC features (NCI, NDEF, LLCP and SNEP protocols, Tag Operations, Host Card Emulation…)

Low level access to PN7150 HW

PN5xx_I2c kernel mode driver can be used to communicate with the PN7150 NFC controller

PN7150 FW autonomously handles part of the NFC activity so it requires much less interaction on the I^{2}C interface than NFC frontends, reducing context switching.

Linux libnfc-nci stack architecture in detail

Libnfc-nci library

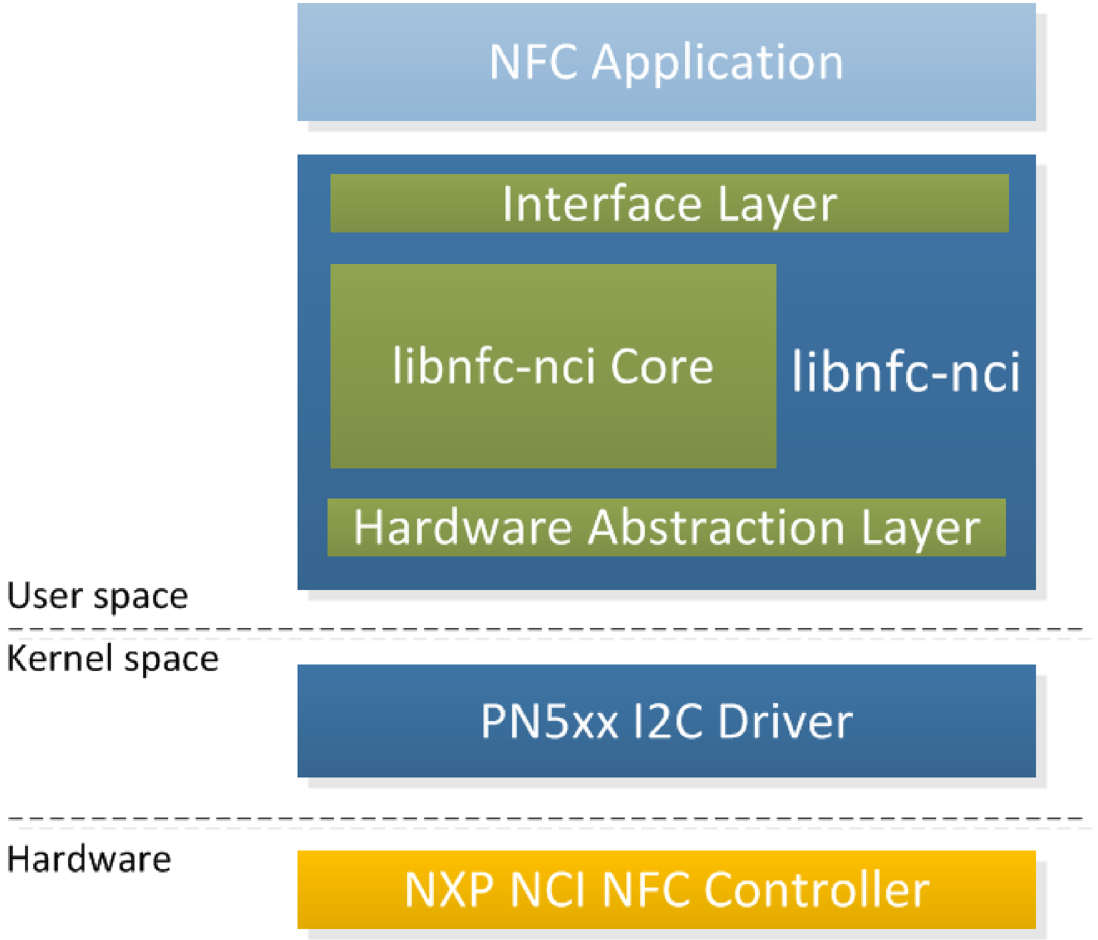
Interface Layer: exposes the library API to the user application.

Libnfc-nci Core: implements the NFC functionality (NCI, NDEF, LLCP and SNEP protocols, tag operations, Host Card Emulation…)

Hardware Abstraction Layer: provides connection to the kernel driver as well as basic functionalities such as self-test or firmware update.

Linux Libnfc-nci repository permanently updated and maintained by NXP

```
$ git clone https://github.com/NXPNFCLinux/linux_libnfc-nci.git
```



NFC Application

Interface Layer

libnfc-nci Core

libnfc-nci

Hardware Abstraction Layer

User space

Kernel space

PN5xx I2C Driver

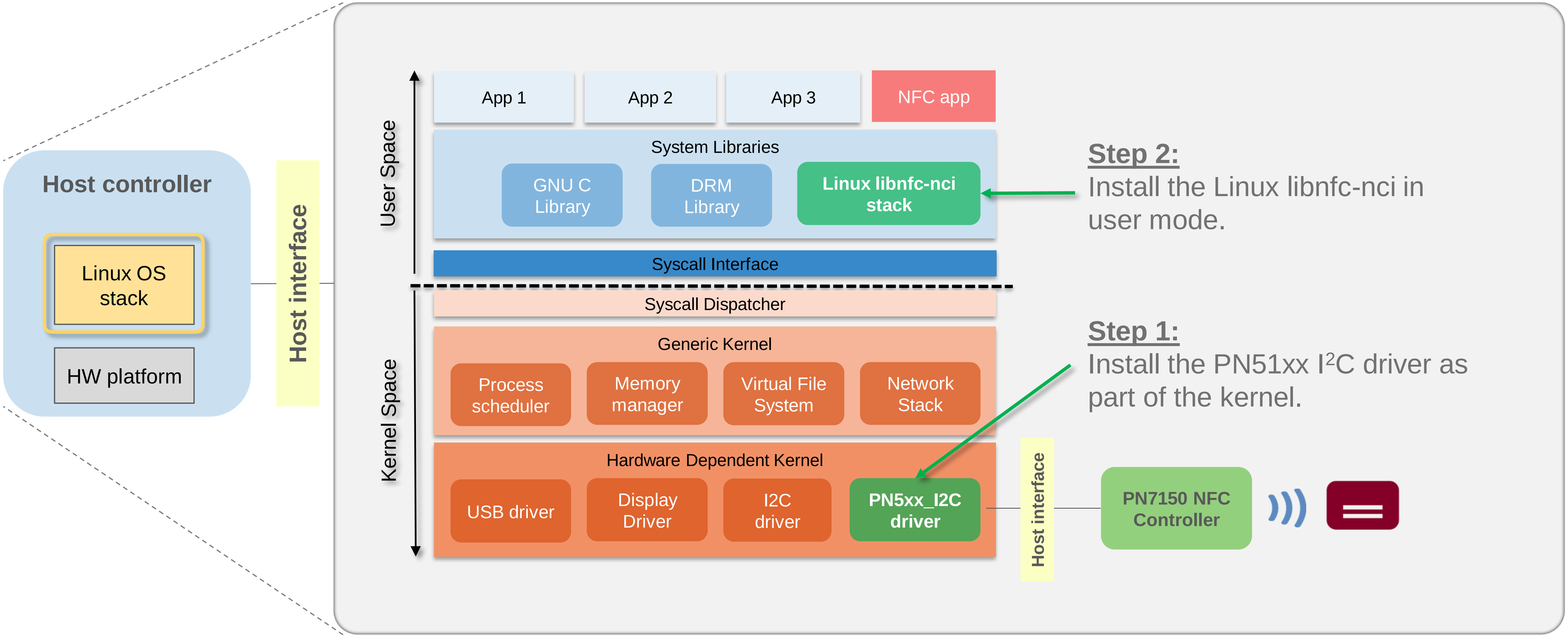
Hardware

NXP NCI NFC Controller

*For alternatives to PN5xx_I2C kernel driver, check AN11769

Integrating the Linux-libnfc-nci stack into a Linux system

Integrating the Linux libnfc-nci stack into a Linux system



Step 2:
Install the Linux libnfc-nci in user mode.

Step 1:
Install the PN51xx I^{2}C driver as part of the kernel.

Integration steps are described in the AN11769 – PN71xx Linux Software Stack Integration Guidelines in detail.

Step 1: Installing PN5xx I2C driver

- Steps to install the PN5xx I2C driver:

1. Download the driver source code

```
$ git clone https://github.com/NXPNFCLinux/linux_libnfc-nci.git
```

2. Include the driver to the kernel

```
Obj-y       +=nxp-pn5xx/
Source “drivers/misc/nxp-pn5xx/Kconfig”
```

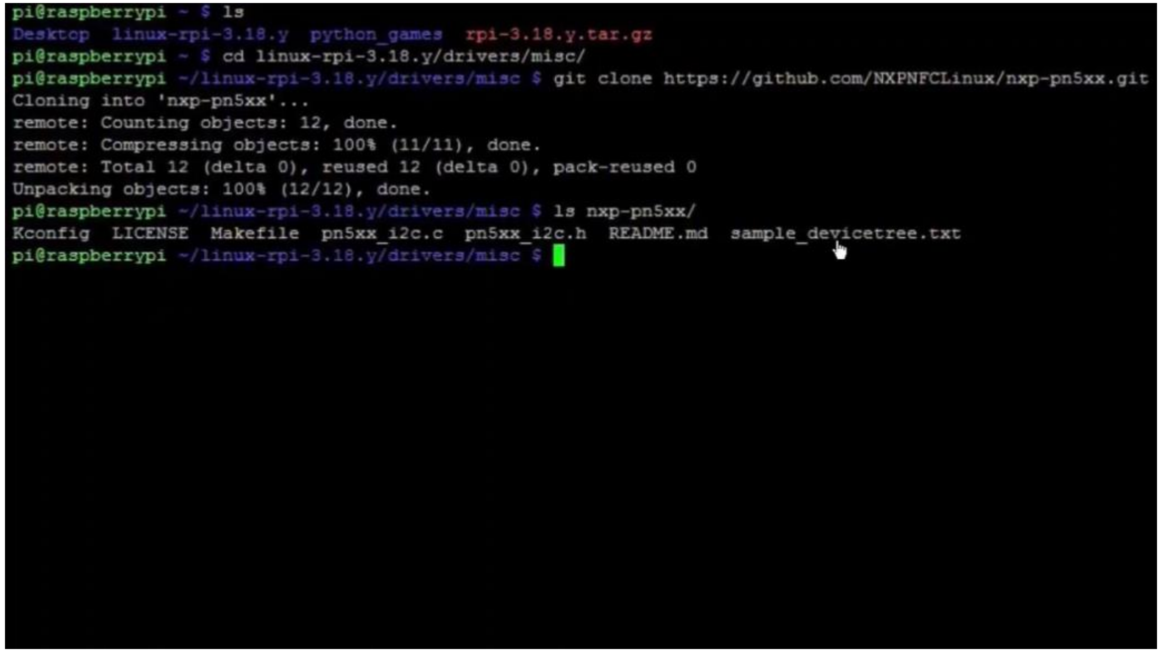
3. Create the device node

```
&i2c{
   status = "okay";
   pn547: pn547@28 {
   compatible = "nxp,pn547";
   reg = <0x28>;
   clock-frequency = <400000>;
   interrupt-gpios = <&gpio2 17 0>;
   enable-gpios = <&gpio4 21 0>;
   };
};
```

4. Build the driver

5. Change access to device node

```
ACTION==“add”, KERNEL==“pn544”, MODE=“066”
```



```
pi@raspberrypi ~ $ ls
Desktop  linux-rpi-3.18.y  python_games  rpi-3.18.y.tar.gz
pi@raspberrypi ~ $ cd linux-rpi-3.18.y/drivers/misc/
pi@raspberrypi ~/linux-rpi-3.18.y/drivers/misc $ git clone https://github.com/NXPNFCLinux/nxp-pn5xx.git
Cloning into 'nxp-pn5xx'...
remote: Counting objects: 12, done.
remote: Compressing objects: 100% (11/11), done.
remote: Total 12 (delta 0), reused 12 (delta 0), pack-reused 0
Unpacking objects: 100% (12/12), done.
pi@raspberrypi ~/linux-rpi-3.18.y/drivers/misc $ ls nxp-pn5xx/
Kconfig  LICENSE  Makefile  pn5xx_i2c.c  pn5xx_i2c.h  README.md  sample_devicetree.txt
pi@raspberrypi ~/linux-rpi-3.18.y/drivers/misc $
```

Live demo with RaspberryPi: http://youtu.be/TCgCRi-tKxM

Step 2: Installing the Linux libnfc-nci

- Steps to install the Linux libnfc-nci:

1. Download the driver source code

```
$ git clone https://github.com/NXPNFCLinux/linux_libnfc-nci.git
```

2. Build the library

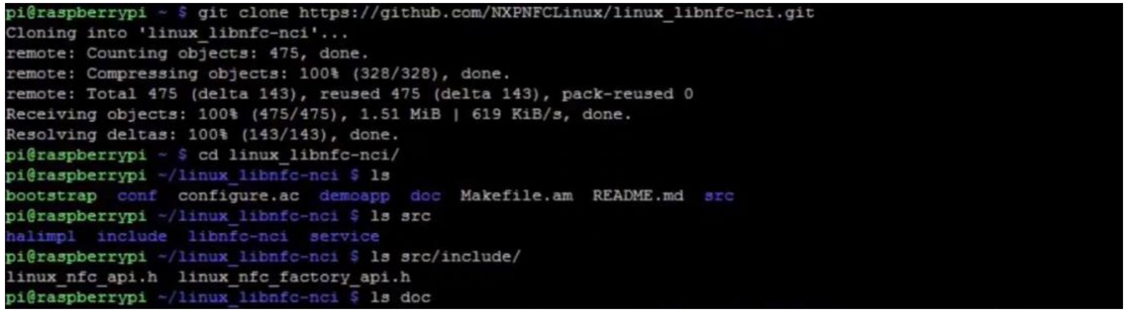
```
$ ./bootstrap
$ ./configure <OPTIONS>
$ make
```

3. Install the library

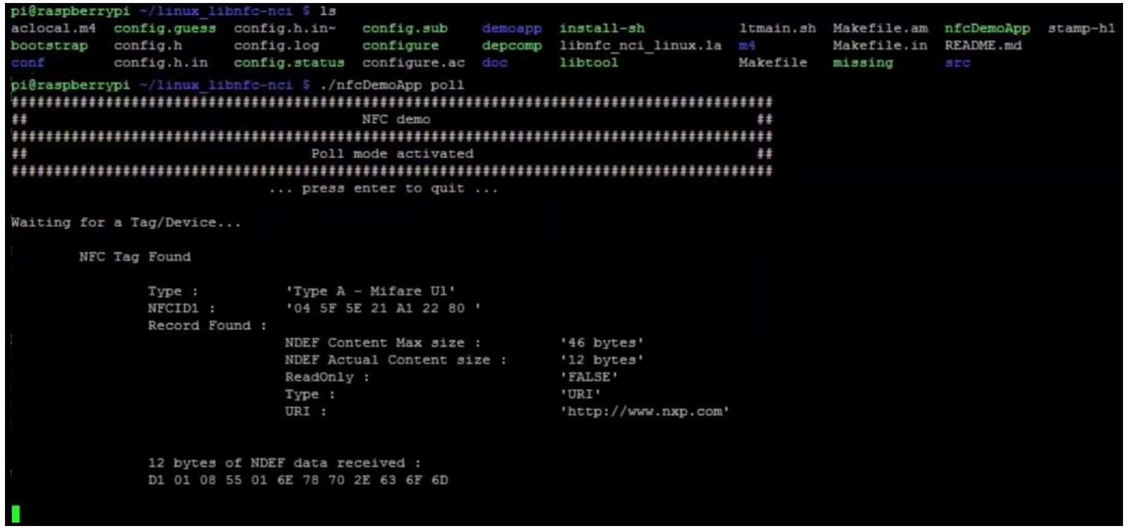
```
$ make install
```

4. Using the application

```
$ ./nfcDemoApp <OPTIONS>
```



```
pi@raspberrypi ~ $ git clone https://github.com/NXPNFCLinux/linux_libnfc-nci.git
Cloning into 'linux_libnfc-nci'...
remote: Counting objects: 475, done.
remote: Compressing objects: 100% (328/328), done.
remote: Total 475 (delta 143), reused 475 (delta 143), pack-reused 0
Receiving objects: 100% (475/475), 1.51 MiB | 619 KiB/s, done.
Resolving deltas: 100% (143/143), done.
pi@raspberrypi ~ $ cd linux_libnfc-nci/
pi@raspberrypi ~/linux_libnfc-nci $ ls
bootstrap  conf  configure.ac  demoapp  doc  Makefile.am  README.md  src
pi@raspberrypi ~/linux_libnfc-nci $ ls src
halimpl  include  libnfc-nci  service
pi@raspberrypi ~/linux_libnfc-nci $ ls src/include/
linux_nfc_api.h  linux_nfc_factory_api.h
pi@raspberrypi ~/linux_libnfc-nci $ ls doc
```



```
pi@raspberrypi ~/linux_libnfc-nci $ ls
aclocal.m4  config.guess  config.h.in~   config.sub     demoapp  install-sh            ltmain.sh  Makefile.am  nfcDemoApp  stamp-h1
bootstrap   config.h      config.log     configure      depcomp  libnfc_nci_linux.la   m4         Makefile.in  README.md
conf        config.h.in   config.status  configure.ac   doc      libtool               Makefile   missing      src
pi@raspberrypi ~/linux_libnfc-nci $ ./nfcDemoApp poll
#########################################################################################
##                                       NFC demo                                      ##
#########################################################################################
##                                 Poll mode activated                                 ##
#########################################################################################
                              ... press enter to quit ...

Waiting for a Tag/Device...

        NFC Tag Found

                Type :          'Type A - Mifare Ul'
                NFCID1 :        '04 5F 5E 21 A1 22 80 '
                Record Found :
                                NDEF Content Max size :         '46 bytes'
                                NDEF Actual Content size :      '12 bytes'
                                ReadOnly :                      'FALSE'
                                Type :                          'URI'
                                URI :                           'http://www.nxp.com'


                12 bytes of NDEF data received :
                D1 01 08 55 01 6E 78 70 2E 63 6F 6D

```

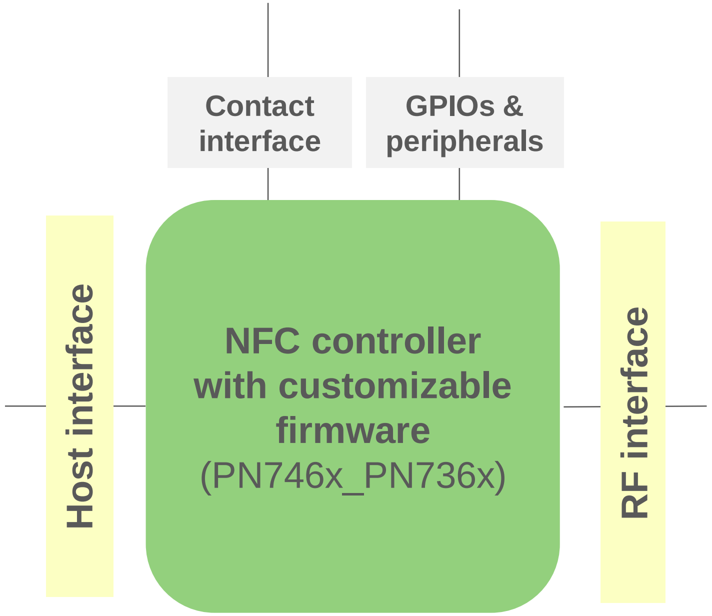
Live demo with RaspberryPi: http://youtu.be/TCgCRi-tKxM

PN7462 NFC controllers family concept

PN7462 NFC controller family host and contactless interfaces

Host interface

- PN7462 can be connected to the host device through I2C, SPI, USB or HSUART
- The customizable FW gives developers the widest range of options for functionality and enables a high degree of design flexibility.
- PN7462 host interface can be used to interact with a high level API-like offered by the FW logic programmed into PN7462 internal MCU.



RF interface

- High RF output power frontend IC for transfer speed up to 848 Kbps
- Compliance with EMV and NFC Forum standards.
- Low power card detection mode.
- Dynamic Power Control (DPC) support.
- Load modulation schemes: Active & Passive.

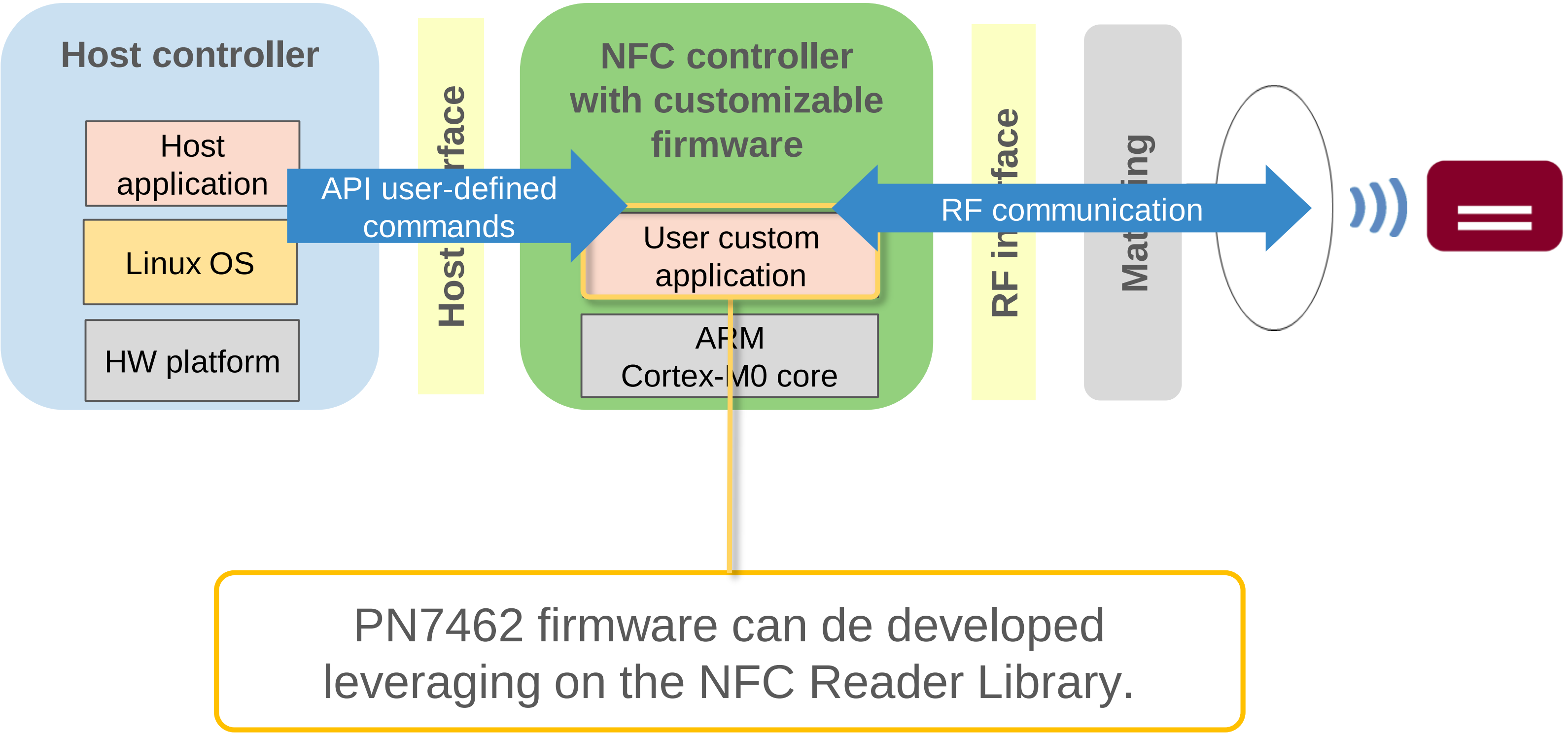
PN7462 NFC controller family are integrated solutions combining an NFC frontend together with an MCU with customizable firmware

PN7462 NFC controller family are managed by an API-like interface

Host controller

- PN7462 can be run stand-alone user application or be connected to a high-end platform running an operative system like Linux.
- PN7462 custom FW can be designed to provide a higher level protocol or API instead of exposing a register interface over host interface (e.g. NFC frontends).
- This solution offers an (almost) latency independent interface between host SW stack and the RF frontend.

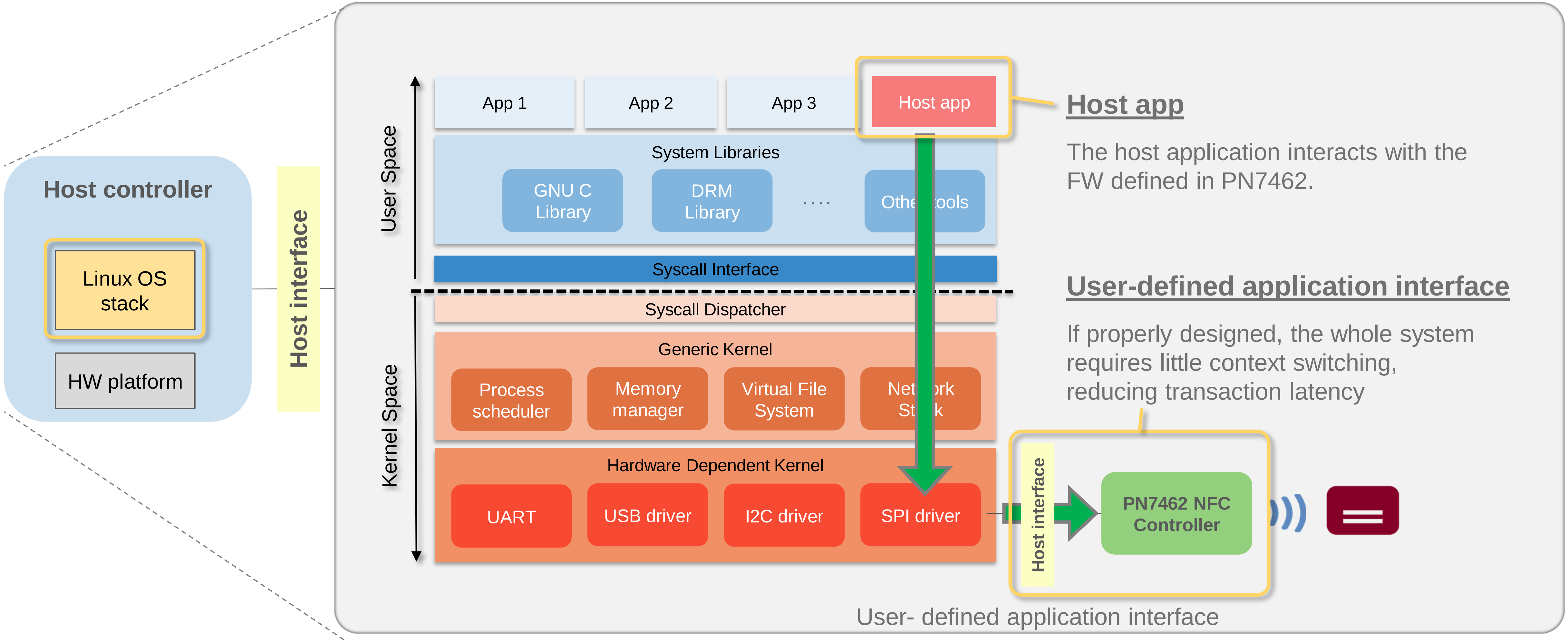
PN7462 embeds an ARM Cortex-M0 with up to 160kB of flash memory and can be configured to run fully-custom applications



PN7462 firmware can de developed leveraging on the NFC Reader Library.

PN7462 NFC controller family
NFC software integration in Linux

PN7462 architecture for Linux integration



Host controller

Linux OS stack

HW platform

Host interface

User Space

Kernel Space

App 1

App 2

App 3

Host app

System Libraries

GNU C Library

DRM Library

....

Other tools

Syscall Interface

Syscall Dispatcher

Generic Kernel

Process scheduler

Memory manager

Virtual File System

Network Stack

Hardware Dependent Kernel

UART

USB driver

I2C driver

SPI driver

Host interface

PN7462 NFC Controller

Host app

The host application interacts with the FW defined in PN7462.

User-defined application interface

If properly designed, the whole system requires little context switching, reducing transaction latency

User- defined application interface

e.g. For a payment application, PN7462 can implement the EMVCo L1 while the host app can implement the EMVCo L2 payment applications

NFC Reader Library – Simplified API

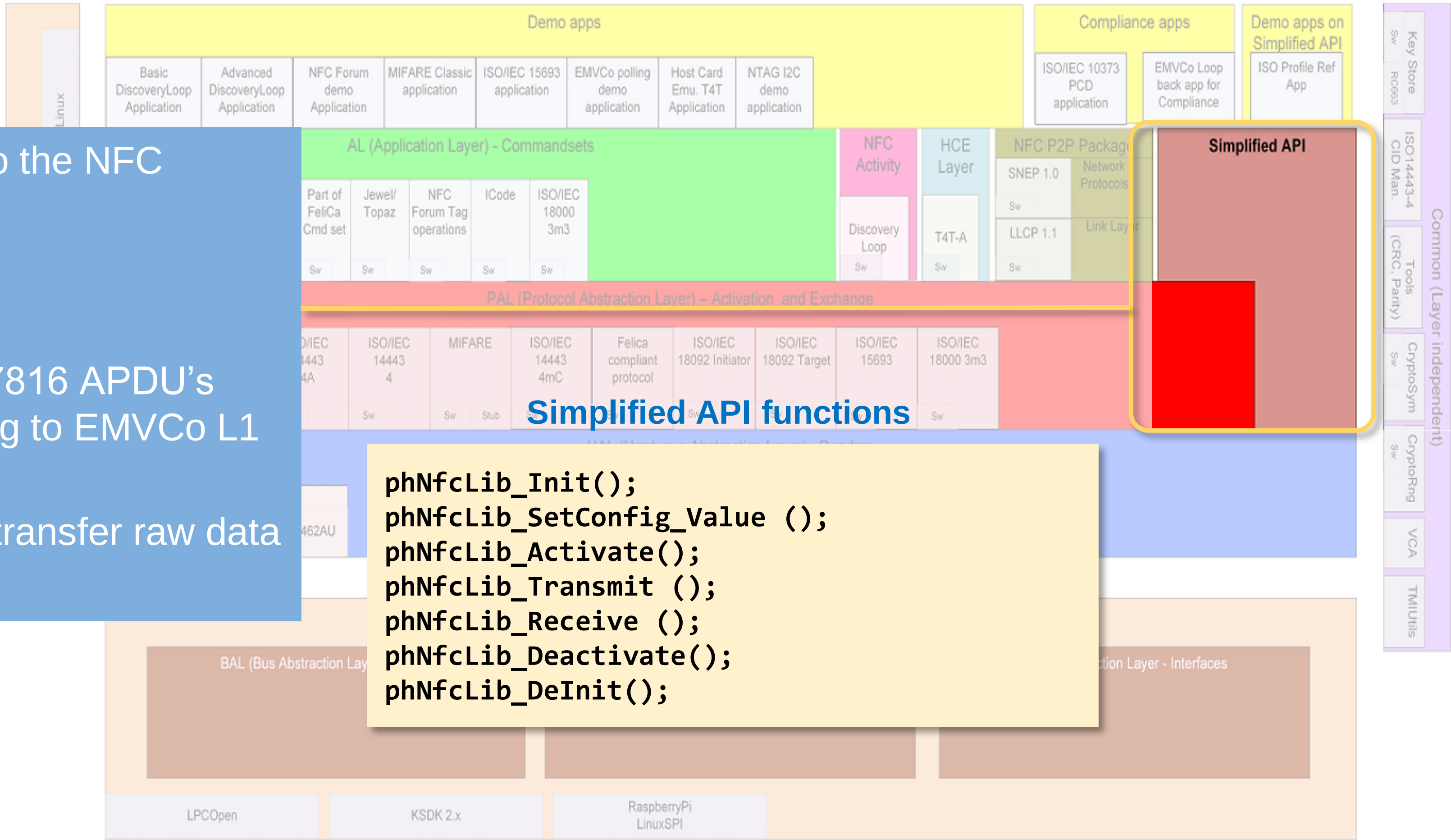
An example for developing your application interface-like development

Simplified API

Provides a high level abstraction access to the NFC Reader Library.

Two different available profiles:

- **EMVCo channel**: used to transfer ISO 7816 APDU’s over the ISO 14443-4 protocol, according to EMVCo L1 spec.
- **ISO channel**: general channel, used to transfer raw data on block/frame level.



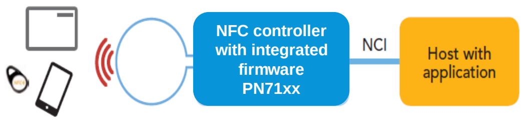
Simplified API functions

```
phNfcLib_Init();
phNfcLib_SetConfig_Value ();
phNfcLib_Activate();
phNfcLib_Transmit ();
phNfcLib_Receive ();
phNfcLib_Deactivate();
phNfcLib_DeInit();
```

Wrap up & Q&A

PN71xx vs PN7462

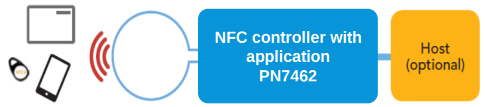
NFC controllers with integrated FW



Key benefits

- Full NFC Forum-compliant contactless interface
- Microcontroller core with integrated firmware.
- Host interface: I^2C interface over NCI protocol.

NFC controllers with customizable FW

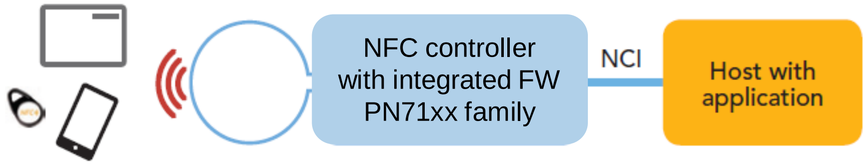


Key benefits

- Contactless interface with full NFC functionality.
- Microcontroller core with fully customizable firmware.
- One configurable host interface: I^2C, SPI, USB, HSUART.
- Contact card reader (PN7462).
- Two master interfaces: I^2C and SPI and up to 21 GPIOs

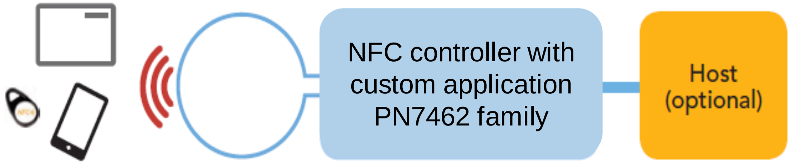
PN7150 and PN7462 considerations for Linux integration

NFC controllers with integrated FW

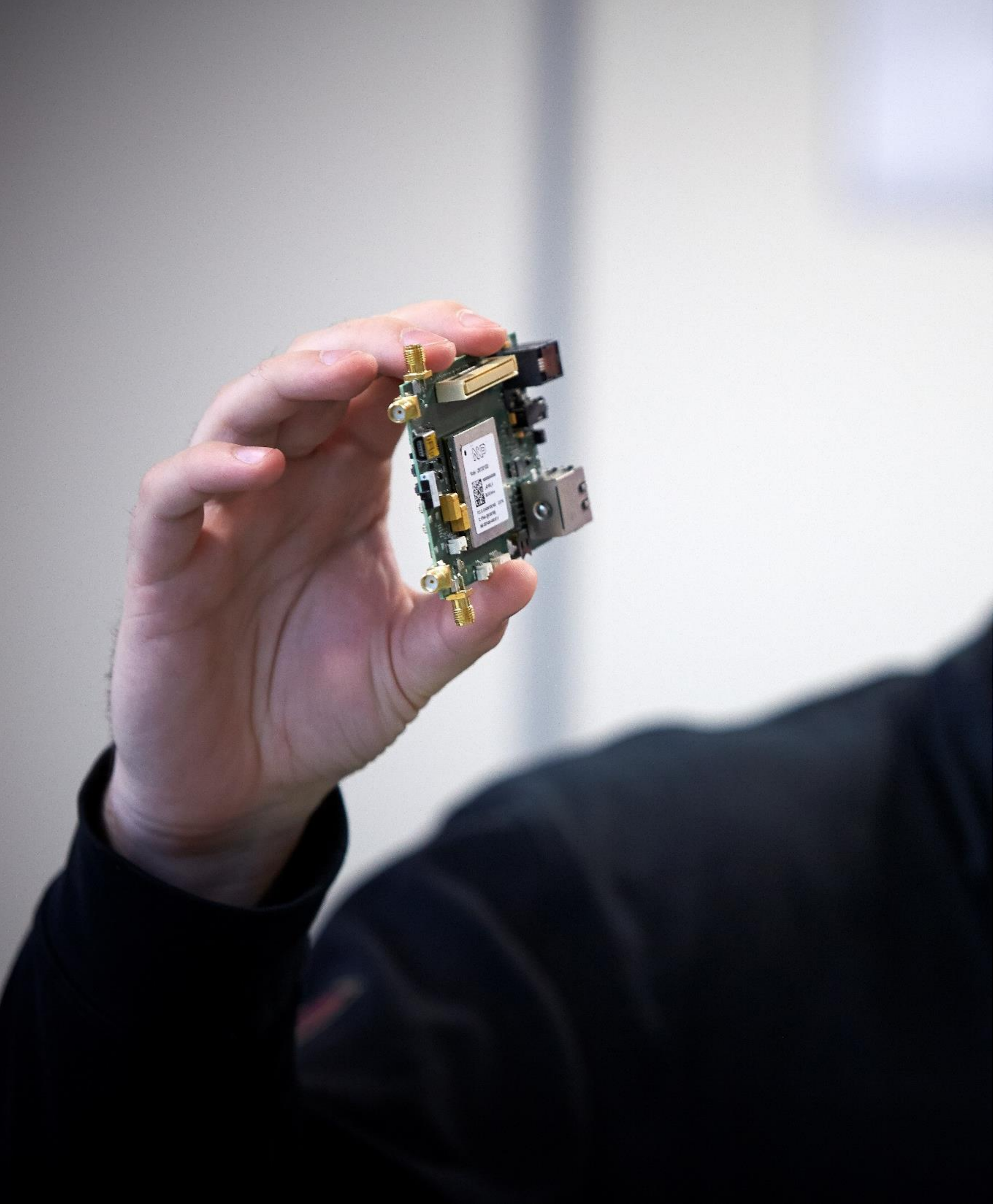


- PN7150 logical interface (API) is based on the NCI NFC Forum standard over an I^{2}C physical interface
- There are few libraries out there, but the ***Linux libnfc-nci*** is NXP NFC stack for Linux systems and the most complete.
- Host interactions are reduced since some functionalities are handled autonomously by the FW.

NFC controllers with customizable FW



- PN7462 internal flash memory can be used to program a fully custom FW that exposes a high level protocol or API towards the host system.
- PN7462 firmware can de developed leveraging on the **NFC Reader Library**.
- It offers an (almost) latency independent interface between host software stack and the RF frontend.



Reference links & info

- PN71xx family
  www.nxp.com/products/: PN7150B0HN

- PN746x_PN736x family
  www.nxp.com/products/:PN7462AUHN

- NXPNFCLinux GitHub repository
  https://github.com/NXPNFCLinux/

- NFC Reader Library
  www.nxp.com/pages/:NFC-READER-LIBRARY

Software development in Android and iOS
Embedded software for MCUs
JCOP, Java Card operating Systems
Hardware design and development
Digital, analog, sensor acquisition, power management
Wireless communications WiFi, ZigBee, Bluetooth, BLE
Contactless antenna RF design, evaluation and testing

MIFARE® product-based applications
End-to-end systems, readers and card-related designs
EMVco applications
Readers, cards, design for test compliancy (including PCI)
Secure Element management
GlobalPlatform compliant backend solutions
Secure services provisioning OTA, TSM services

We help companies leverage the mobile and **contactless revolution**

MobileKnowledge
Roc Boronat 117, P3M3
08018 Barcelona
(Spain)

Get in touch with us
www.themobileknowledge.com
mk@themobileknowledge.com