Greybus for IoT

Greybus

Greybus

Designed for a modular smartphone

- Application layer for UniPro bus
- hotplug / hot unplug
- Modules discovery
- Class and protocols to talk to modules

Greybus

Main classes

- Camera
- Audio
- HID
- I2C
- SPI
- GPIO
- SDIO
- PWM
- UART

Greybus / UniPro topology

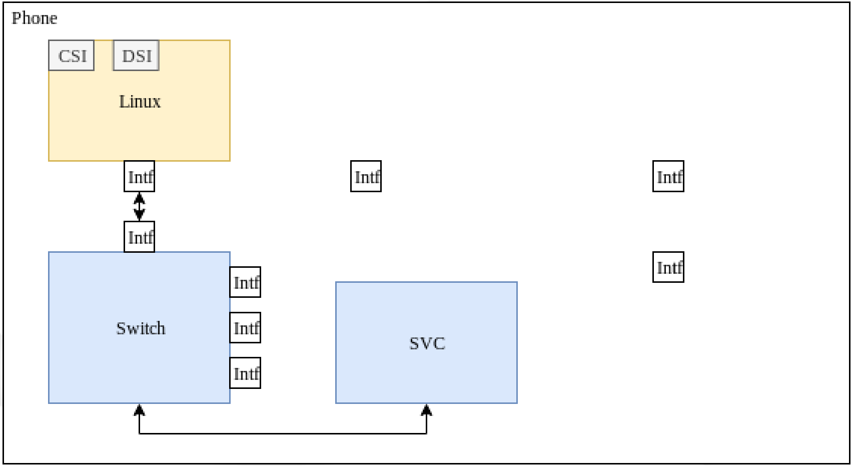


Figure 1:

Greybus / UniPro topology

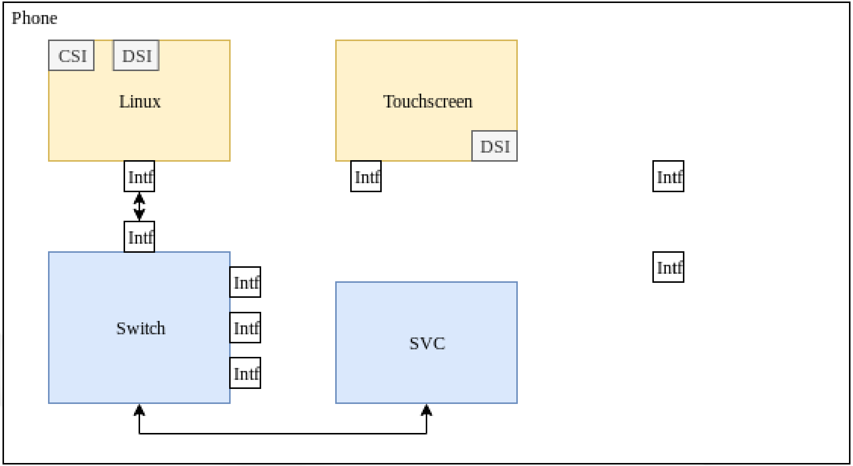


Figure 2:

Greybus / UniPro topology

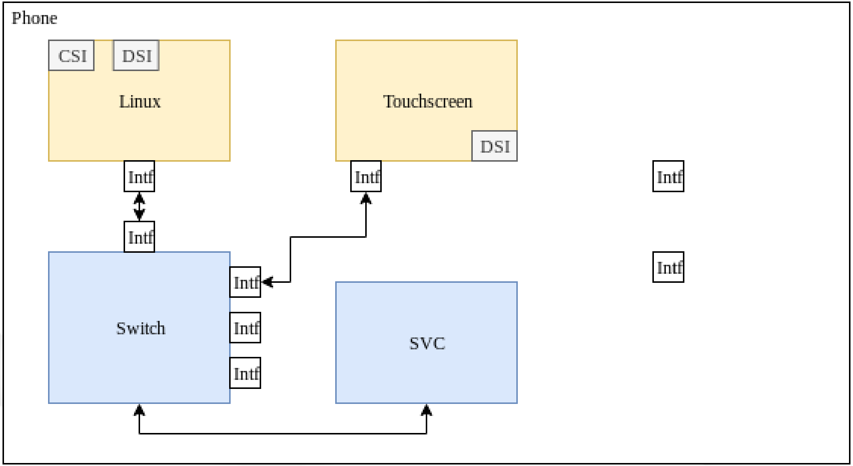


Figure 3:

Greybus / UniPro topology

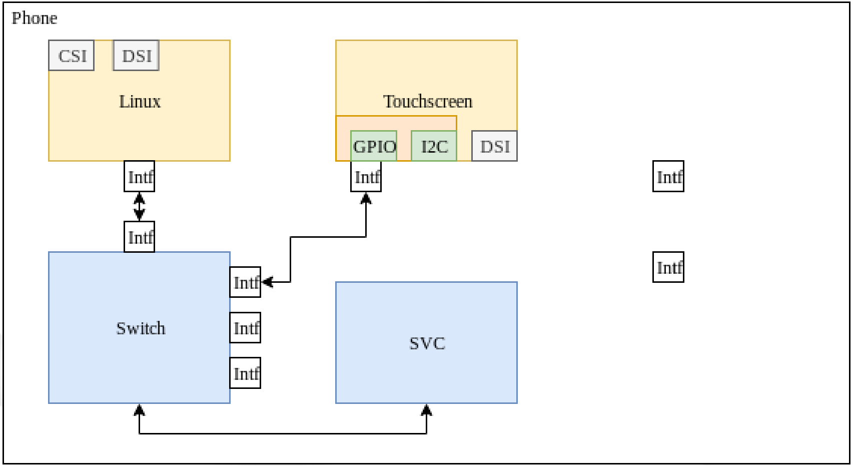


Figure 4:

Greybus / UniPro topology

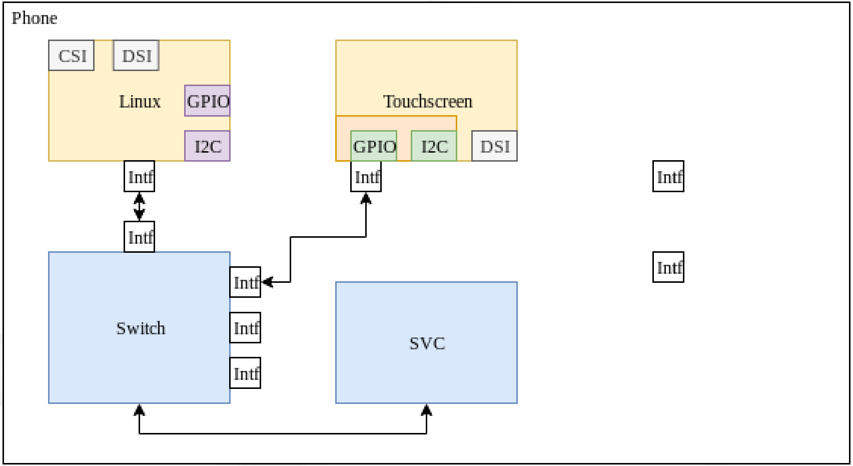


Figure 5:

Greybus / UniPro topology

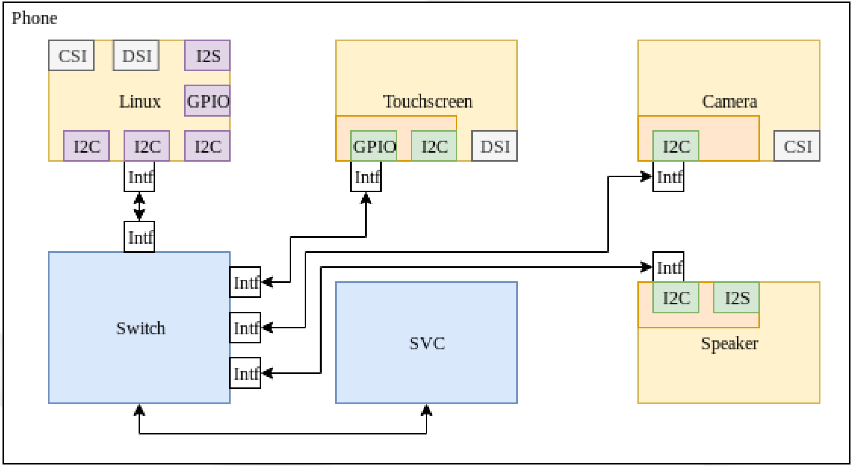


Figure 6:

Greybus

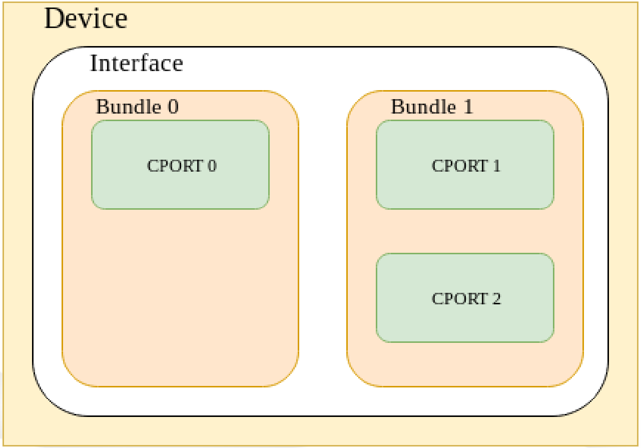


Figure 7:

Samples

Greybus manifest (1/2)

```
[manifest-header]
version-major = 0
version-minor = 1

[interface-descriptor]
vendor-string-id = 1
product-string-id = 2

[string-descriptor 1]
string = BayLibre

[string-descriptor 2]
string = Simple GPIO Interface
```

Greybus manifest (2/2)

```
[cport-descriptor 1]
bundle = 1
protocol = 0x02

[bundle-descriptor 1]
class = 2
```

Greybus GPIO sample

- /sys/class/gpio
  - export
  - gpiochip506
  - unexport
- $cat /sys/class/gpio/gpiochip506/label

  greybus_gpio
- $cat /sys/class/gpio/gpiochip506/ngpio

  6
- $ echo 506 > /sys/class/gpio/export
- $ echo out > /sys/class/gpio/gpio506/direction
- $ echo 1 > /sys/class/gpio/gpio506/value

Greybus for IOT

Why Greybus may be useful for IOT ?

- Free
- Highly documented
- Already supported by the kernel (since 4.9)
- Keep the intelligence in the host
- It just works!

Greybus for the gateway

- Discover the modules
- Discover modules features
- Load and enable drivers
- Take control of modules, using regular Linux API

Greybus for the modules

- Only control the hardware
- Handle Greybus requests
- Let the gateway do everything

Greybus / IOT topology

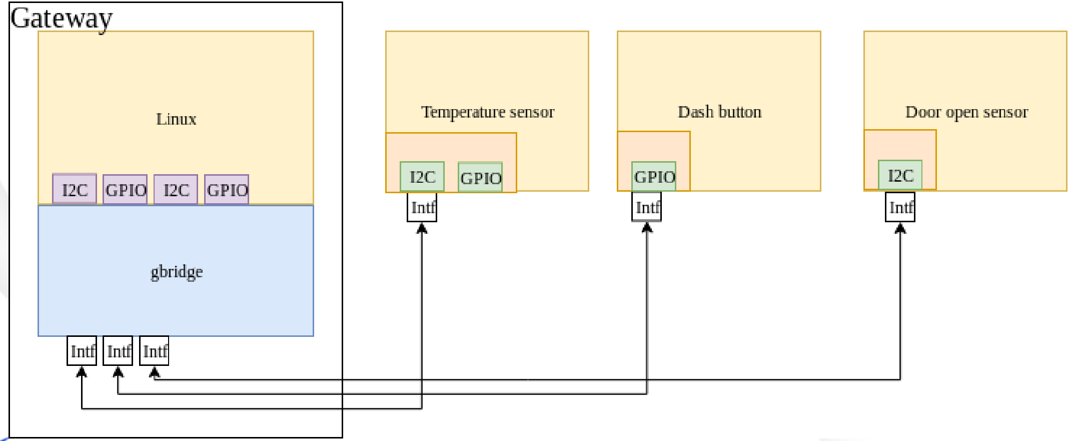
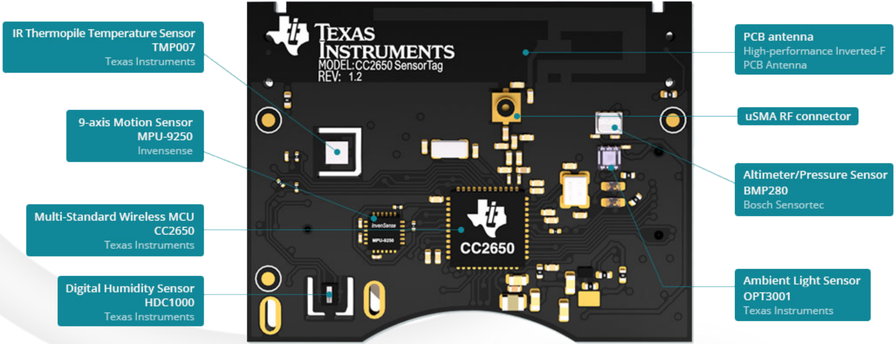


Figure 8:

CC26xx SensorTag



IR Thermopile Temperature Sensor
TMP007
Texas Instruments

9-axis Motion Sensor
MPU-9250
Invensense

Multi-Standard Wireless MCU
CC2650
Texas Instruments

Digital Humidity Sensor
HDC1000
Texas Instruments

PCB antenna
High-performance Inverted-F
PCB Antenna

uSMA RF connector

Altimeter/Pressure Sensor
BMP280
Bosch Sensortec

Ambient Light Sensor
OPT3001
Texas Instruments

Figure 9:

Greybus for IOT

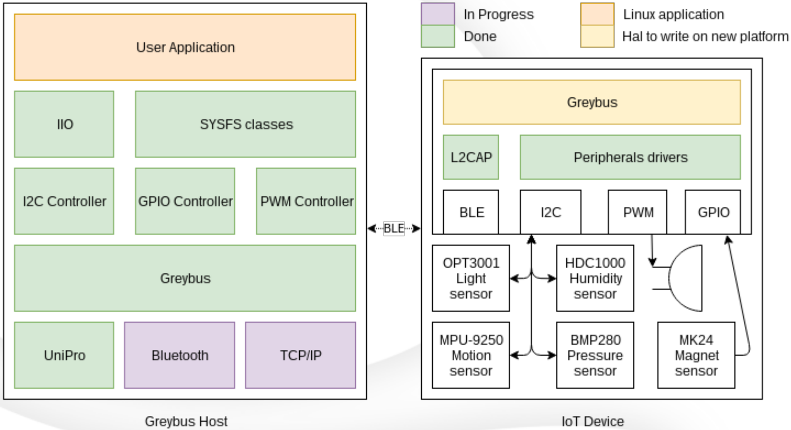


Figure 10:

Limitations / Know issues

Limitations

Performances

- Quite variable
- Some protocols only execute one RPC at time
- A high round trip latency will break down performances

Power Management

- Incomplete
- Remote wake up is missing
- Protocol overhead

Security

- No security (except the one provided by transport medium)

TODO

- Upstream `gb_netlink`
- Write a `Greybus` stack for major RTOS
- Add support of new medium to `gbridge`
  - BLE
  - 6LoWPAN
  - LR WPAN
  - ZigBee
- Encrypt traffic between modules and `gbridge`
- Build and test automatically using CI
- Write a good documentation

Contribute

Kernel

- greybus-dev@lists.linaro.org

Greybus for IoT

- abailon@baylibre.com
- https://github.com/anobli/gbridge.git

Thank you

Backup

Greybus: An application layer of UniPro

What is UniPro

UniPro is an interface to interconnect integrated circuits in mobile phone. It implements layer 1 to 4 of the OSI model.

UniPro applications layer

- UFS: Universal Flash Storage
- CSI-3: Camera Serial Interface
- DSI-2: Display Serial Interface
- Greybus

Greybus: An application layer of UniPro

UniPro features

- High speed physical interface
- High bandwidth
- Low power

But

- Doesn't support hotplug / hot unplug
- Just a network

Greybus sysfs

sysfs layout

- `/sys/bus/greybus/devices/`
- `1-1: module`
- `1-1.1: interface`
- `1-1.1.1: bundle 1`
- `1-1.1.ctrl: control bundle`

Firmware sample

```
uint8_t gb_gpio_direction_out(struct gb_operation *operation)
{
    struct gb_gpio_direction_out_request *request =
        gb_operation_get_request_payload(operation);

    gpio_direction_out(request->which, request->value);
    return GB_OP_SUCCESS;
}

uint8_t gb_gpio_set_value(struct gb_operation *operation)
{
    struct gb_gpio_set_value_request *request =
        gb_operation_get_request_payload(operation);

    gpio_set_value(request->which, request->value);
    return GB_OP_SUCCESS;
}
```