1 Introduction
2 Greybus for IOT
3 Greybus
4 Samples
5 Limitations / Known issues
Project ARA

What is Project ARA?

The goal of project ARA was:
- to create a modular smartphone

Features

- Interchangeable modules
- Modules can be added or removed at runtime
- There can be many types of modules:
  - Screen
  - Camera
  - Speaker
  - E-ink
  - ...

Greybus

An RPC protocol to manage and control modules.
Features

- hotplug / hot unplug
- Modules discovery
- Class and protocols to talk to modules
### Main classes

- Camera
- Audio
- HID
- I2C
- SPI
- GPIO
- SDIO
- PWM
- UART
Greybus for IOT

Why Greybus may be useful for IOT?

- Free
- Highly documented
- Will be merged to mainline kernel soon (currently in linux-next)
- Keep the intelligence in the host
- It just works!
Greybus for IOT

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
Greybus for IOT
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

<table>
<thead>
<tr>
<th>UniPro features</th>
</tr>
</thead>
<tbody>
<tr>
<td>• High speed physical interface</td>
</tr>
<tr>
<td>• High bandwidth</td>
</tr>
<tr>
<td>• Low power</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>But</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Doesn’t support hotplug / hot unplug</td>
</tr>
<tr>
<td>• Just a network</td>
</tr>
</tbody>
</table>
Greybus / UniPro topology
Greybus / IOT topology
Greybus sysfs

<table>
<thead>
<tr>
<th>sysfs layout</th>
</tr>
</thead>
<tbody>
<tr>
<td>▪ /sys/bus/greybus/devices/</td>
</tr>
<tr>
<td>▪ 1-1: module</td>
</tr>
<tr>
<td>▪ 1-1.1: interface</td>
</tr>
<tr>
<td>▪ 1-1.1.1: bundle 1</td>
</tr>
<tr>
<td>▪ 1-1.1.ctrl: control bundle</td>
</tr>
</tbody>
</table>
Greybus manifest

[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

[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
Firmware sample

```c
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;
}
```
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
## Limitations

### Security
- No security (except the one provided by transport medium)
- Not safe to use for some usages

### Other
- Need Greybus module on the host
- Only work on local network
### Requirements
- A Beagle Bone Black to run gbsim
- A computer to run Greybus and gbridge

### Sources
- `git clone https://github.com/anobli/gbridge.git -b ELCE`
- `git clone https://github.com/anobli/greybus.git -b ELCE`
- `git clone https://github.com/anobli/gbsim.git -b ELCE`
Next steps

- Update gbridge and gbsim to work with the latest version of Greybus
- Port Greybus to an OS for MCU.
Greybus was driven by Google to develop a module phone
Now, we are free to do whatever we want
Thank you