Low Level Sensor Programing and Security Enforcement with MRAA

Brendan Le Foll <brendan.le.foll@intel.com>

Agenda

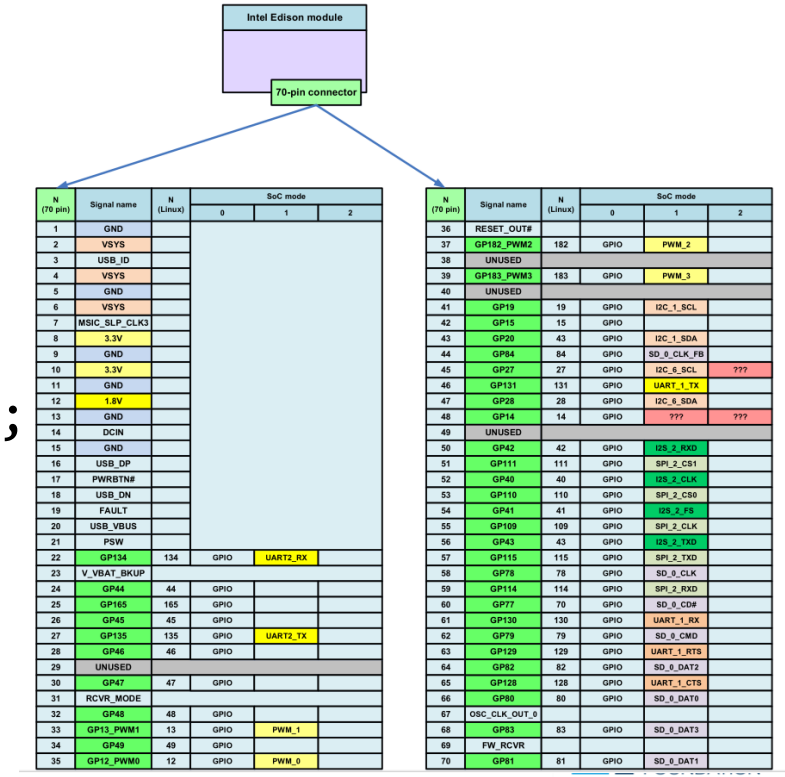
- What’s mraa/mraa.io/libmraa
- Why / What / Where
- Mraa on peripheral manager (PIO)
- Mraa with AGL’s AFB
- Future

Mraa.io - 101

- Simple userspace I/O protocol control for:
  - Gpio, i2c, SPI, ADCs, PWM, UART, 1-Wire
- Made for monkeys – easier is better
- Bindings for C, Python (2 & 3), node.js, & java
- Supports a large number of development boards, platforms, daughterboards & expansion boards

Libmraa - C API

```
void set_gpio_high(int num)
{
  mraa_gpio_context gpio;
  gpio = mraa_gpio_init(num);
  mraa_gpio_dir(gpio, MRAA_GPIO_OUT);
  mraa_gpio_write(gpio, 1);
  mraa_gpio_close(gpio);
}
```



N (70 pin)	Signal name	N (Linux)	SoC mode 0	SoC mode 1	SoC mode 2
1	GND				
2	VSYS				
3	USB_ID				
4	VSYS				
5	GND				
6	VSYS				
7	MSIC_SLP_CLK3				
8	3.3V				
9	GND				
10	3.3V				
11	GND				
12	1.8V				
13	GND				
14	DCIN				
15	GND				
16	USB_DP				
17	PWRBTN#				
18	USB_DN				
19	FAULT				
20	USB_VBUS				
21	PSW				
22	GP134	134	GPIO	UART2_RX	
23	V_VBAT_BKUP				
24	GP44	44	GPIO		
25	GP165	165	GPIO		
26	GP45	45	GPIO		
27	GP135	135	GPIO	UART2_TX	
28	GP46	46	GPIO		
29	UNUSED				
30	GP47	47	GPIO		
31	RCVR_MODE				
32	GP48	48	GPIO		
33	GP13_PWM1	13	GPIO	PWM_1	
34	GP49	49	GPIO		
35	GP12_PWM0	12	GPIO	PWM_0	

N (70 pin)	Signal name	N (Linux)	SoC mode 0	SoC mode 1	SoC mode 2
36	RESET_OUT#				
37	GP182_PWM2	182	GPIO	PWM_2	
38	UNUSED				
39	GP183_PWM3	183	GPIO	PWM_3	
40	UNUSED				
41	GP19	19	GPIO	I2C_1_SCL	
42	GP15	15	GPIO		
43	GP20	43	GPIO	I2C_1_SDA	
44	GP84	84	GPIO	SD_0_CLK_FB	
45	GP27	27	GPIO	I2C_6_SCL	???
46	GP131	131	GPIO	UART_1_TX	
47	GP28	28	GPIO	I2C_6_SDA	
48	GP14	14	GPIO	???	???
49	UNUSED				
50	GP42	42	GPIO	I2S_2_RXD	
51	GP111	111	GPIO	SPI_2_CS1	
52	GP40	40	GPIO	I2S_2_CLK	
53	GP110	110	GPIO	SPI_2_CS0	
54	GP41	41	GPIO	I2S_2_FS	
55	GP109	109	GPIO	SPI_2_CLK	
56	GP43	43	GPIO	I2S_2_TXD	
57	GP115	115	GPIO	SPI_2_TXD	
58	GP78	78	GPIO	SD_0_CLK	
59	GP114	114	GPIO	SPI_2_RXD	
60	GP77	70	GPIO	SD_0_CD#	
61	GP130	130	GPIO	UART_1_RX	
62	GP79	79	GPIO	SD_0_CMD	
63	GP129	129	GPIO	UART_1_RTS	
64	GP82	82	GPIO	SD_0_DAT2	
65	GP128	128	GPIO	UART_1_CTS	
66	GP80	80	GPIO	SD_0_DAT0	
67	OSC_CLK_OUT_0				
68	GP83	83	GPIO	SD_0_DAT3	
69	FW_RCVR				
70	GP81	81	GPIO	SD_0_DAT1	

Libmraa - C++

```
mraa::Gpio* gpio0 = new mraa::Gpio(8);
gpio0->dir(mraa::DIR_OUT);
gpio0->write(1);
```

Libmraa – Object API pt.1 (Python)

```
from mraa import *
x = Gpio(8)
x.dir(DIR_OUT)
x.write(1)
x = "memory is not my problem!"
```

Libmaa – Object API pt.2 (Python)

```
from mraa import *
def hello():
  print("You do not talk about the
GIL")

x = Gpio(8)
x.dir(DIR_IN)
x.isr(EDGE_BOTH, hello)
```

Why?

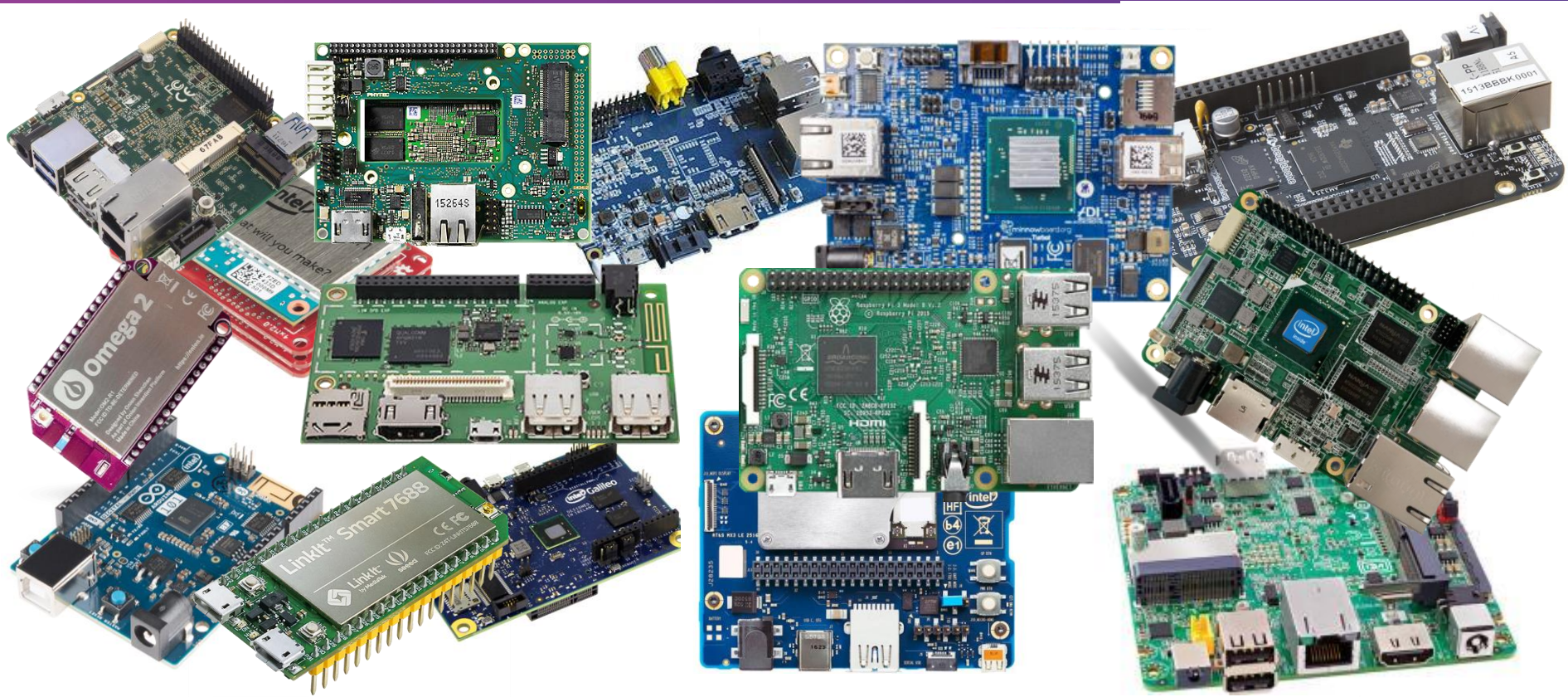
- Unified API
- Demo of http://mraa.io/demo

Sensor Library – “UPM"

- http://upm.mraa.io
  - Sensor browser/integration
  - Documentation / quirks
  - Links to datasheets etc…
- Mostly C++ (some C), java, node.js, python

Yellow Mongoose - Tony Hisgett from Birmingham, UK (CC BY 2.0)

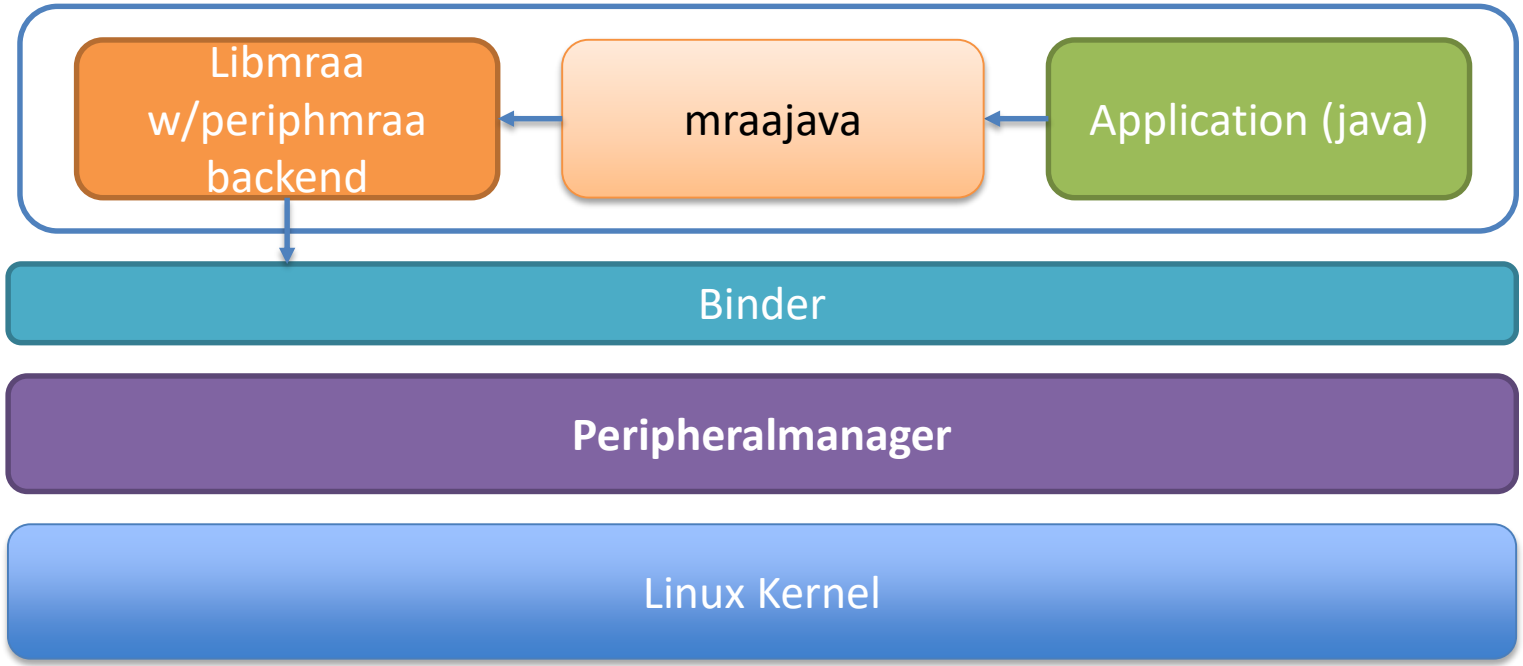
(some) supported boards



Adding a board

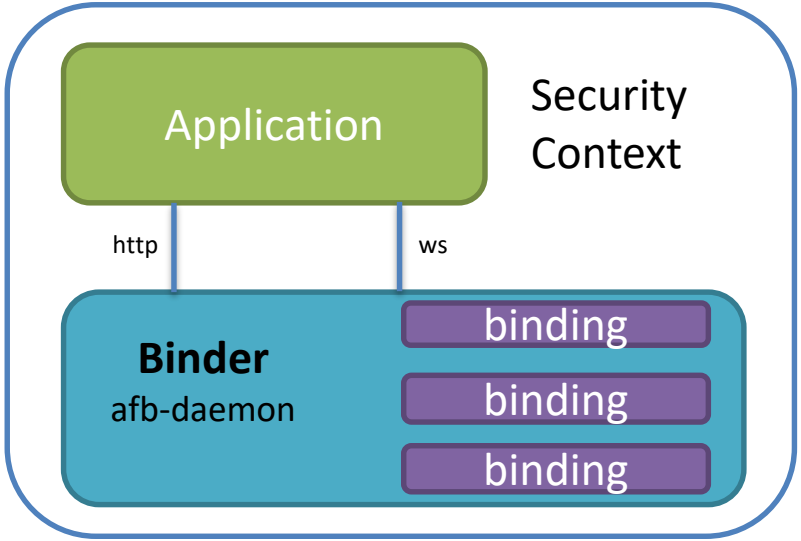
- There are 3 ways to use your board with mraa
  - **Raw mode**, without a platform configuration you can use mraa but no pinmuxing or platform configuration will be done or known. However you can still access your platform IO and this is very useful for testing as well as writing a new platform configuration
  - **C platform configuration**, there are many examples of how to add a mraa_board_t and load it, it's surprisingly easy and many companies & individuals have managed to do it on their own unassisted by ourselves. Depending on platform complexity a mraa_board_t must be complemented by an advanced function array which can override mraa's default IO functions. This is very useful in complex platforms such as Edison or where there are hardware oddities to overcome. Mraa can either be configured with just one platform or with all platforms from that architecture.
  - **Json file**, mraa configuration files for 'simple' platforms will be able to provide everything via a json file that is loaded via mraa_load_platform("myconfig.json") using the same configuration format as imraa

Android Things (a.k.a Brillo/Weave)



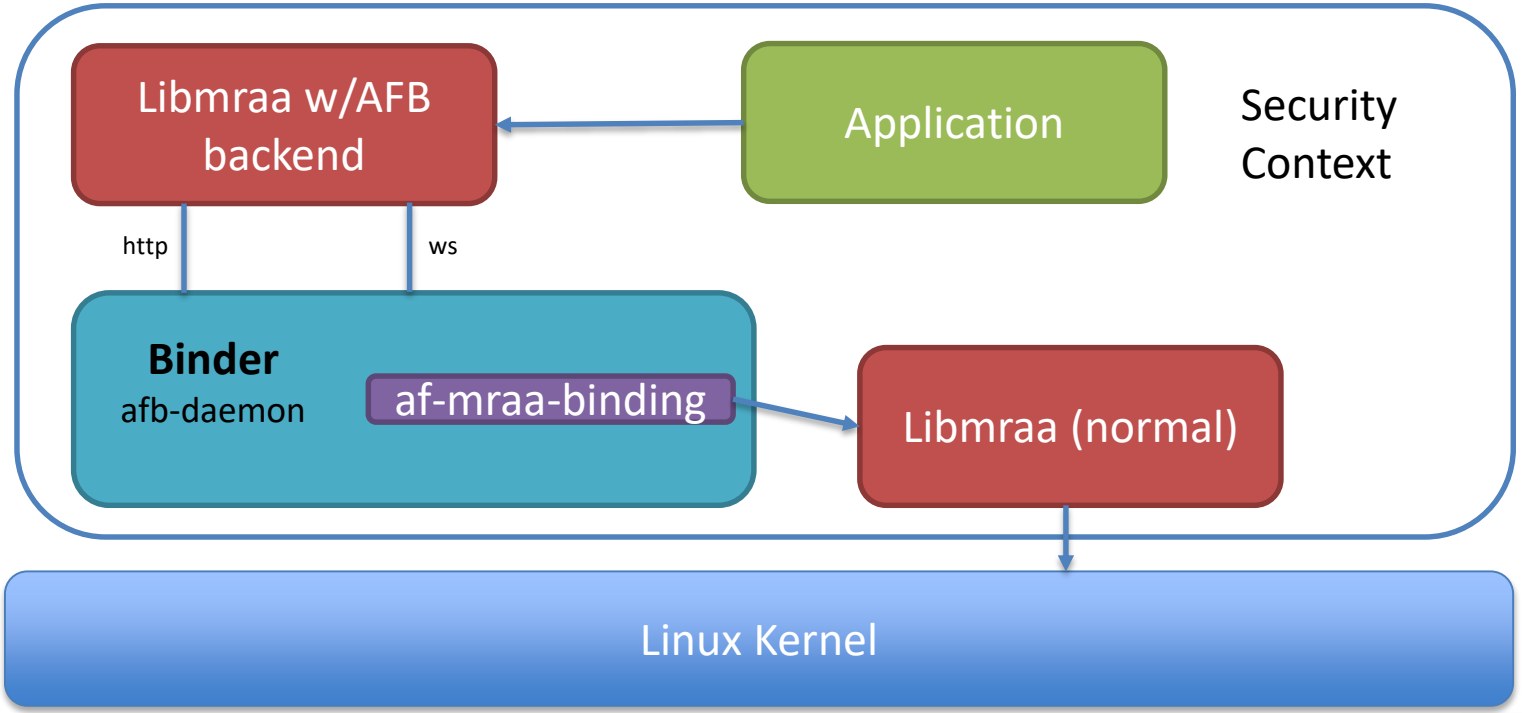
libmraa + AFB

AFB – Application Framework Binder

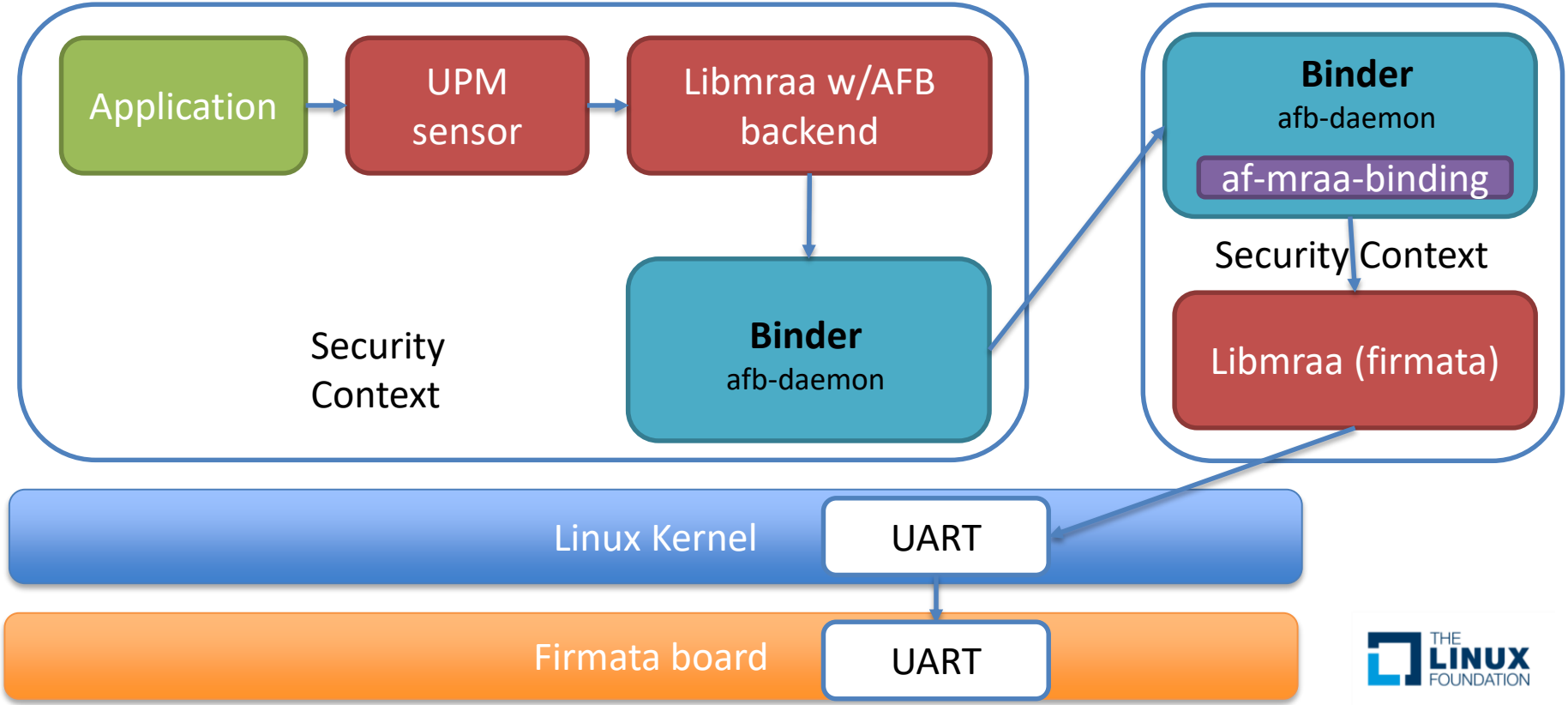


The ***binder*** provides the way to connect applications to the services that it needs.
It provides a fast way to securely offer APIs to applications written in any language and running almost anywhere.
The ***binder*** is developed for AGL.
The ***binder*** is the usual name.
The binary is named **afb-daemon**.
The name **afb-daemon** stands for ***Application Framework Binder Daemon***.
The word *daemon*, here, denote the fact that the ***binder*** makes witchcraft to connect applications to their expected services.
(note: that usually the term of daemon denotes background process but not here).
Each ***binder*** **afb-daemon** is in charge to bind one instance of an application or service to the rest of the system, applications and services.
Within AGL, the connection between services and/or applications is tuned by the AGL framework and the AGL system.

Libmraa + AFB architecture



Libmraa + AFB (complex)



Integrating AFB as a backend

- Mraa has support for platforms to override default behaviour, we do this for odd platforms (Quark X1000), old non standard kernels (Rpi) or because we're not talking to the linux kernel directly at all (firmata, d2xx, peripheralmanager)
- The AGL 'platform' in mraa is compiled in with the –DBUILDARCH="AFB", this will trigger all future calls to mraa calls to go through to the AFB bus to the af-mraa-binding.
- The AGL 'platform' overrides all functionality in mraa so mraa_i2c_init no longer does anything with /dev/i2c-n
- Synchronizes **`afb_wsj1_call_s`** calls

af-mraa-binding

- Af-mraa-binding initializes a ‘normal’ libmraa build against a real –DBUILDARCH. All the build architectures are supported
- Therefore you need two builds of libmraa, one to link against af-mraa-binding and another to link to your application
- Uses wrapjson from afb-utilities

Future

- Currently the AGL platform in mraa has no knowledge of what is actively supported so querying the platform with mraa-i2c or mraa-gpio does not work, the underlying platform API has to be called for this
- The binding is simplistic, it only exposes two verbs, “dev-init” to initialize a mraa context, and “command” to run a mraa command against an initialized context
- Mraa calls are mostly all blocking so calls to afb_wsj1_call_s require a lock to grab the response, currently some calls assume that the action is successful rather than really checking the response from the hardware
- Only some of mraa’s functionality is supported 100% (i2c)

More info

- http://github.com/intel-iot-devkit/mraa
- http://github.com/intel-iot-devkit/af-mraa-binding
- upm.mraa.io
- IRC: Freenode #mraa
- ML: mraa@lists.01.org

Embedded Linux Conference

Europe

THE LINUX FOUNDATION