SONY
make.believe

Dynamic Android Sensor HAL

D * A * S * H

Oskar Anderö
Senior Software Architect
E-mail: oskar.andero@sonymobile.com

SONY
make.believe

D * A * S * H

- Just another sensor HAL implementation
- But...
  - Easy to integrate
  - Configurable
  - Scalable

SONY
make.believe

D * A * S * H

- DASH is open-source!
  - Github: *https://github.com/sonyxperiadev/DASH*
- Allows openness
  - For mod hackers to enjoy
  - 3rd party vendors
  - Sensor HAL example for anyone

SONY
make.believe

What we had

- Messy code
- Lots of #ifdefs
- Many branches
- Different types of Linux drivers

SONY
make.believe

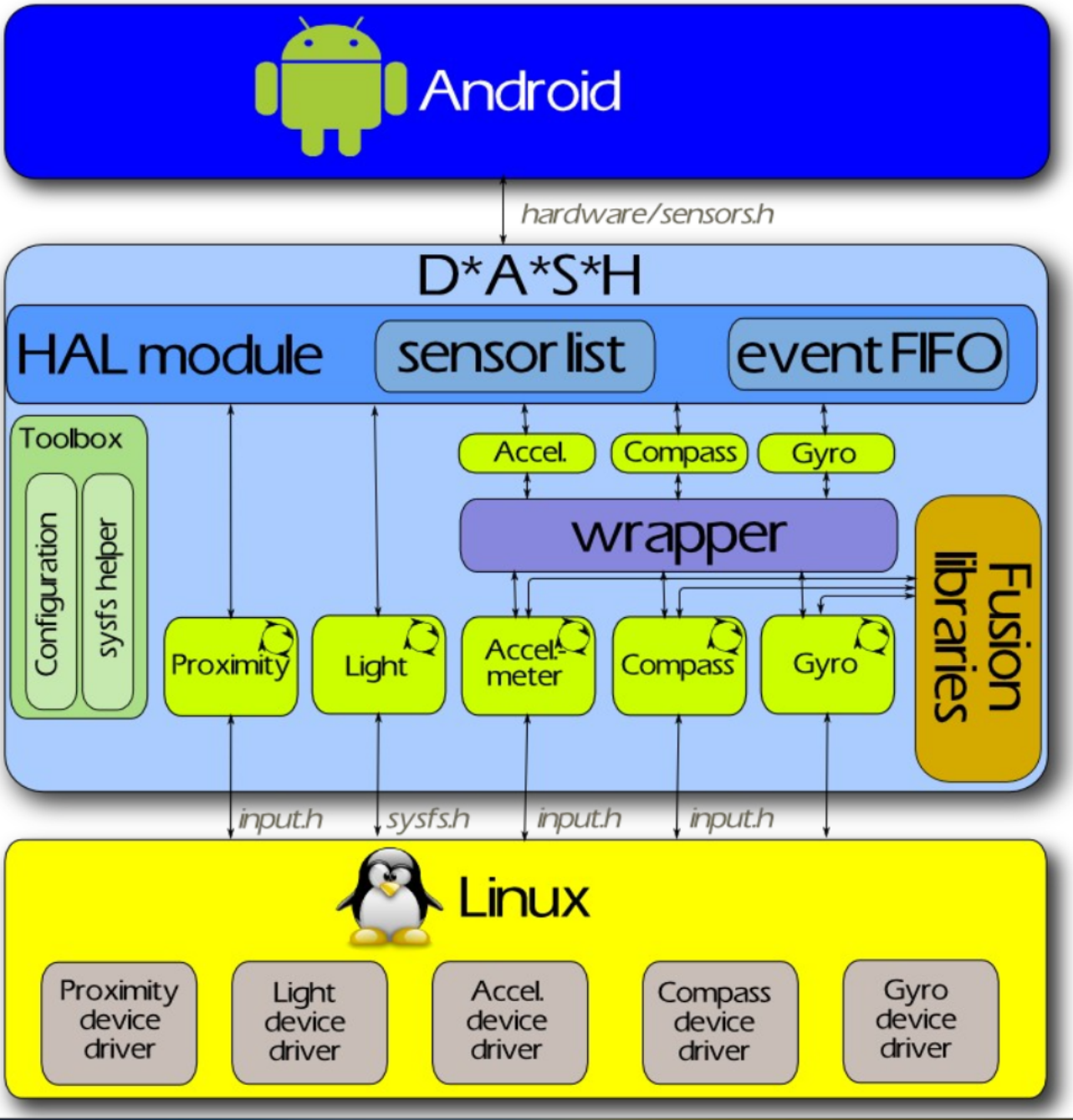
What we wanted

- What lead to the current situation?
  - Different sensors on different products
  - Different mounting on different products
  - Different 3rd party sensor libraries for different sets of sensors
- What to do....?
- Solution: abstract the abstraction!

SONY
make.believe

What we wanted

- Scalable
- Configurable
- Common code should be common
- Ease integration of sensor fusion
- Avoid multiple branches



SONY
make.believe

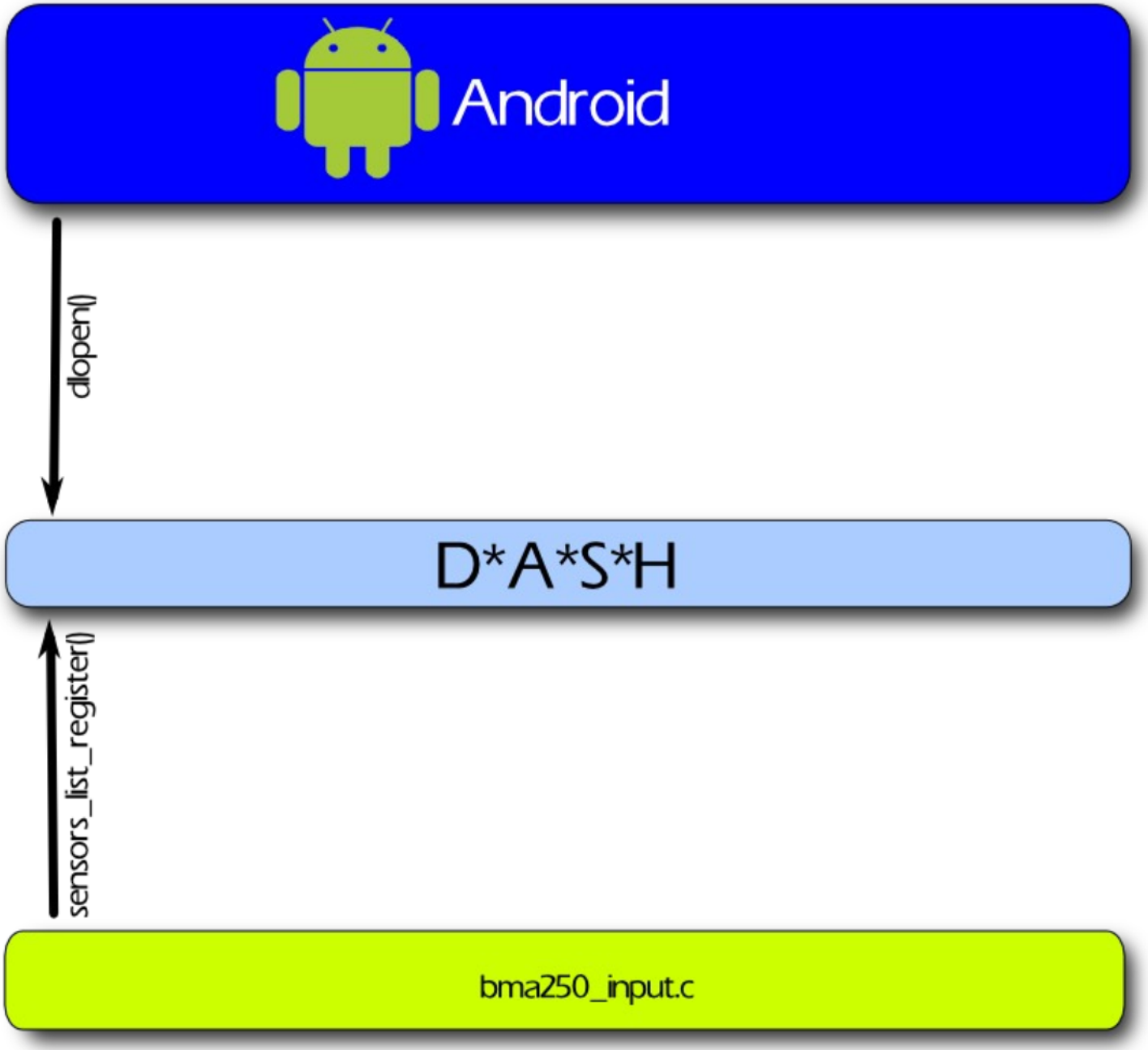
Code example

- BMA250
  - Accelerometer from Bosch
  - Linux interface is input device
  - Polls events in kernel
  - Polling frequency is set by sysfs
- Implemented in *sensors/bma250_input.c*

SONY
make.believe

bma250_input.c

```
static struct sensor_desc bma250_input = {
        .sensor = {
                name: "BMA250 accelerometer",
                vendor: "Bosch Sensortec GmbH",
                version: sizeof(sensors_event_t),
                handle: SENSOR_ACCELEROMETER_HANDLE,
                type: SENSOR_TYPE_ACCELEROMETER,
                maxRange: 156.96, /* max +/-16G */
                resolution: 20,
                power: 0.003,
                minDelay: 10000000
        },
        .api = {
                init: bma250_input_init,
                activate: bma250_input_activate,
                set_delay: bma250_input_delay,
                close: bma250_input_close
        },
};
```

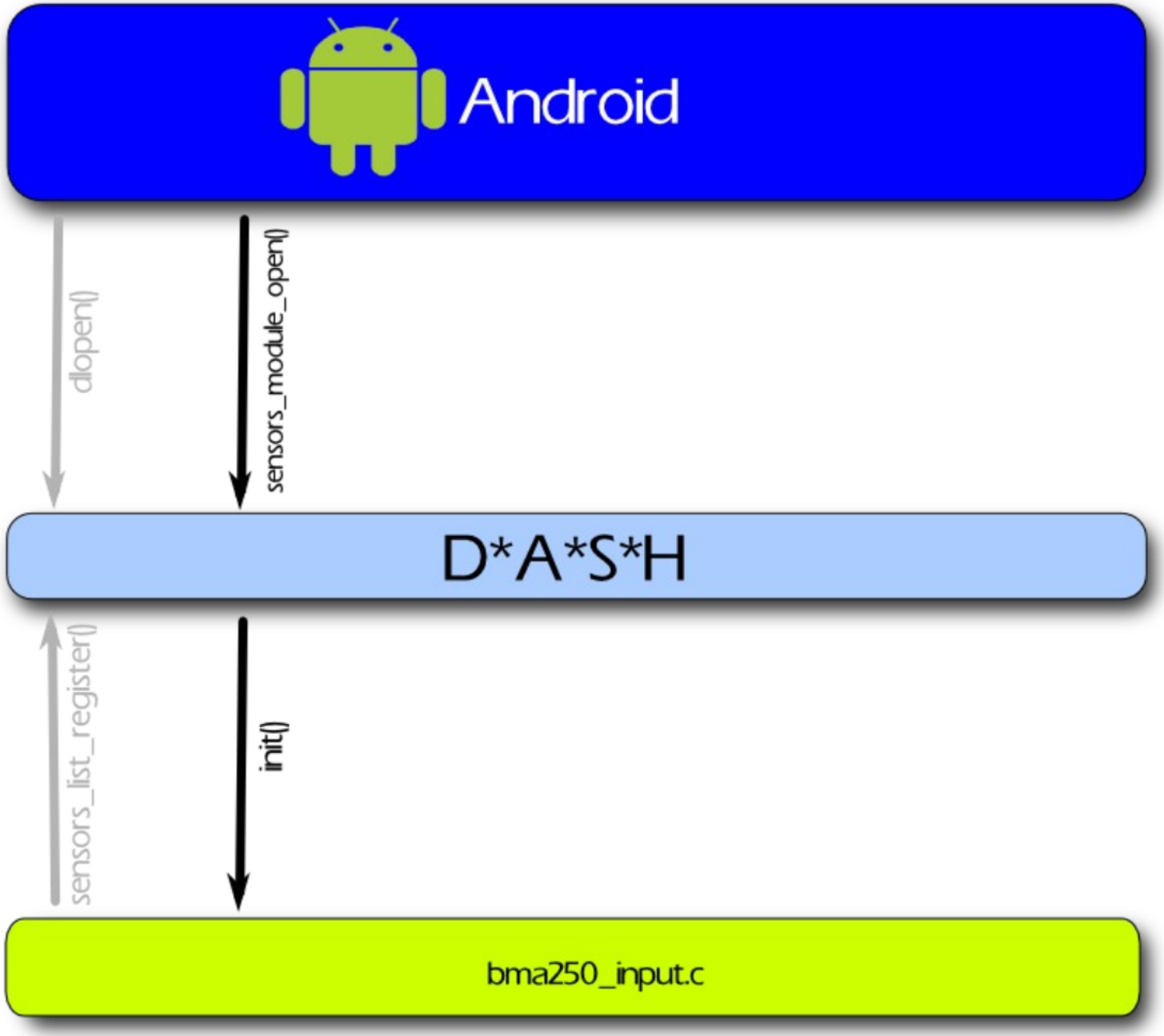
SONY
make.believe

bma250_input.c

```
list_constructor(bma250_input_init_driver);
void bma250_input_init_driver()
{
        (void)sensors_list_register(
                                &bma250_input.sensor,
                                &bma250_input.api);
}
```

...and in sensors_list.h:

```
#define list_constructor(fn) extern void fn(void) \
                        __attribute__ ((constructor));
```

SONY
make.believe

bma250_input.c

```
static int bma250_input_init(struct sensor_api_t *s)
{
        ...
        bma250_input_read_config(d);

        /* check for availablity */
        ...
        sensors_sysfs_init(&d->sysfs,
                                        BMA250_INPUT_NAME,
                                        SYSFS_TYPE_INPUT_DEV);
        sensors_select_init(&d->select_worker,
                                         bma250_input_read, s, -1);
        return 0;
}
```

SONY
make.believe

bma250_input.c

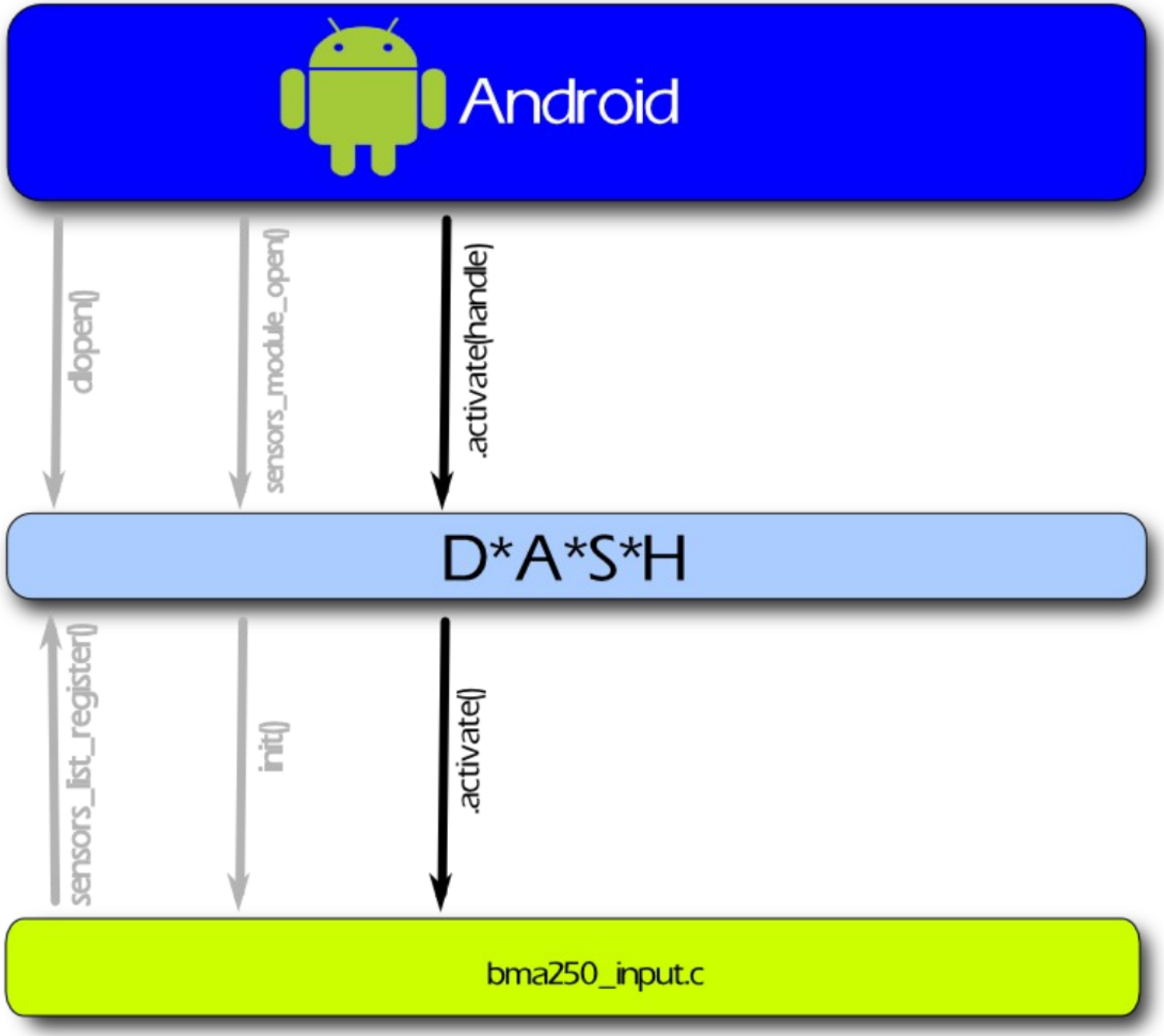
```
static void bma250_input_read_config(struct sensor_desc *d)
{
        ...
        if (!sensors_have_config_file()) {
                ALOGI("%s: No config file found: using default config.",
                     __func__);
                return;
        }
        for (i = 0; i < (sizeof(conf_values)/sizeof(conf_values[0])); i++) {
                int value;
                if (sensors_config_get_key("bma250input", conf_values[i].key,
                                           TYPE_INT,
                                           (void*)&value,
                                           sizeof(value)) < 0) {
                        ALOGE("%s: failed to read %s", __func__, conf_values[i].key);
                        return;
                }
                ...
}
```

SONY
make.believe

bma250_input.c

- Sensors.conf example:

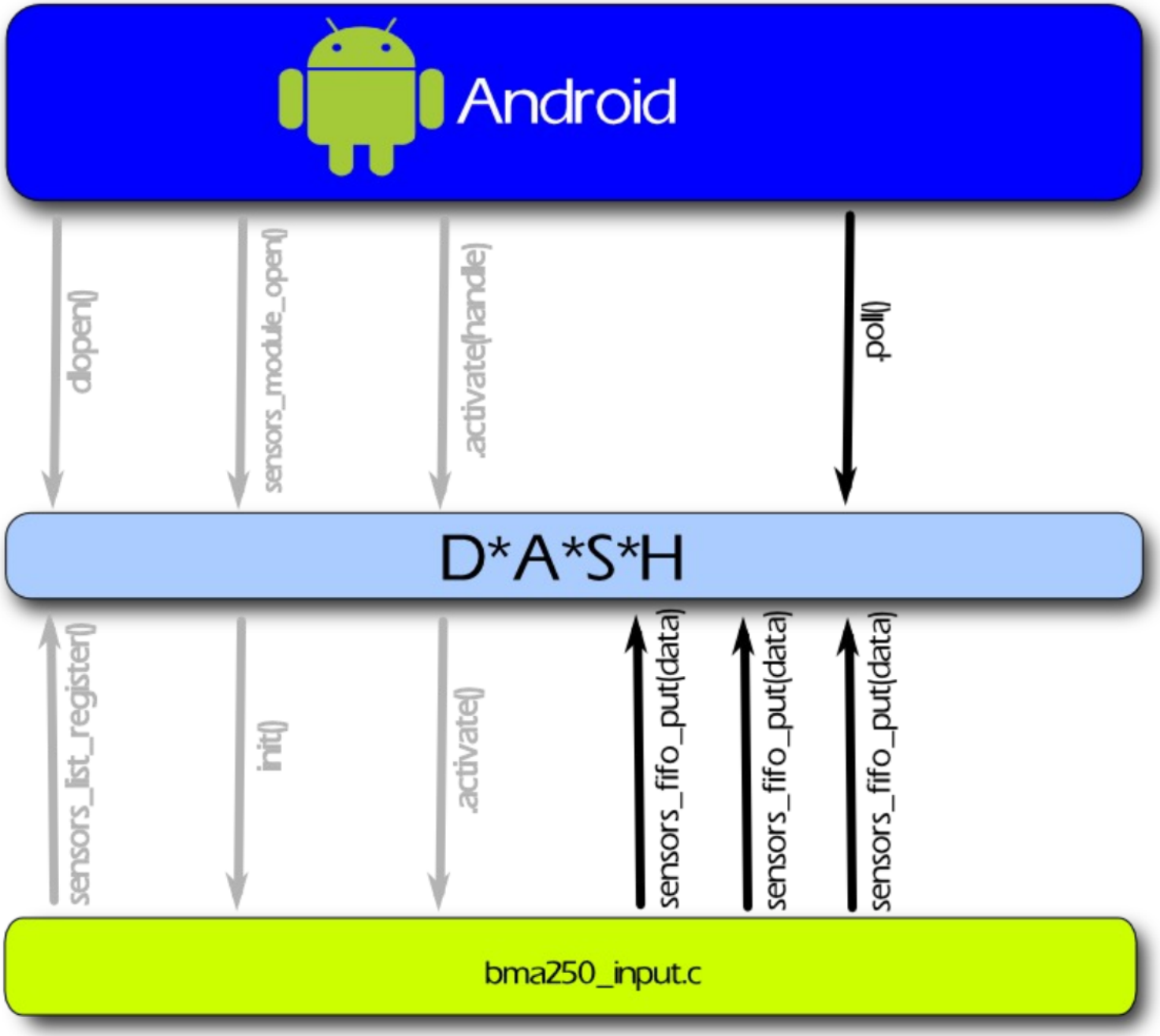
```
# Setup accelerometer axis
bma250input_axis_x = 0
bma250input_axis_y = 1
bma250input_axis_z = 2

bma250input_neg_x = 0
bma250input_neg_y = 1
bma250input_neg_z = 1
```

SONY
make.believe

bma250_input.c

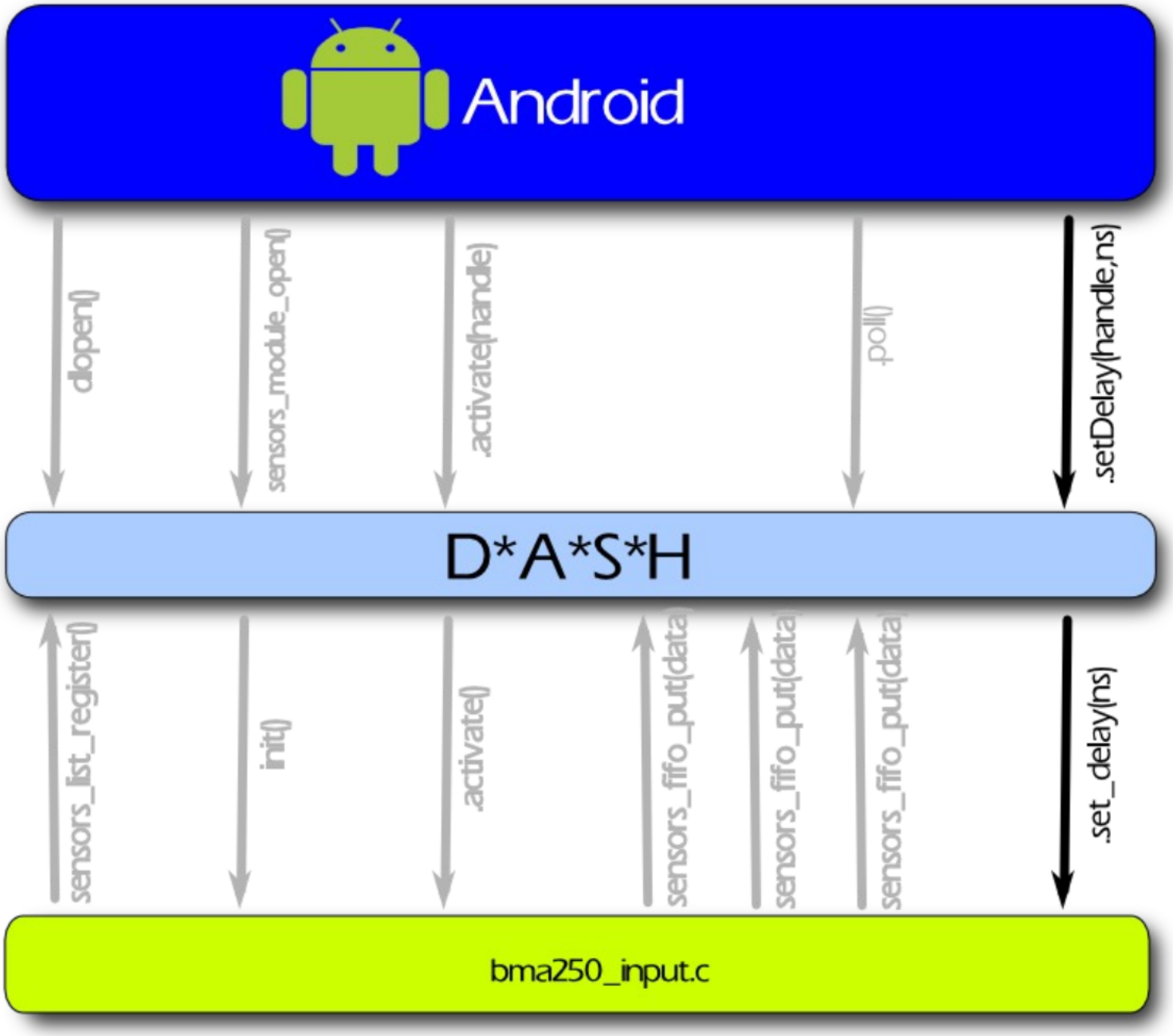
```
static int bma250_input_activate(struct sensor_api_t *s, int enable)
{
        if (enable && (fd < 0)) {
                fd = open_input_dev_by_name(BMA250_INPUT_NAME,
                        O_RDONLY | O_NONBLOCK);
                ...
                d->select_worker.set_fd(&d->select_worker, fd);
                d->select_worker.resume(&d->select_worker);
        } else if (!enable && (fd > 0)) {
                d->select_worker.set_fd(&d->select_worker, -1);
                d->select_worker.suspend(&d->select_worker);
        }
        return 0;
}
```



SONY
make.believe

bma250_input.c

```
static void *bma250_input_read(void *arg)
{
        ...
        int fd = d->select_worker.get_fd(&d->select_worker);
        while (read(fd, &event, sizeof(event)) > 0) {
                switch (event.type) {
                case EV_ABS:
                        switch (event.code) {
                        case ABS_X:
                              d->current_data[0] = ev2grav(event.value);
                              ...
                case EV_SYN:
                        data.acceleration.x = (d->neg_x ? -d->current_data[d->axis_x] :
                                               d->current_data[d->axis_x]);
                        ...
                        data.timestamp = get_current_nano_time();

                        sensors_fifo_put(&data);
                        goto exit;
   ...
}
```



SONY
make.believe

bma250_input.c

```
static int bma250_input_set_delay(struct sensor_api_t *s, int64_t ns)
{
        ...
        d->delay = ns;
        d->select_worker.set_delay(&d->select_worker, ns);

        /* rate */
        ret = d->sysfs.write_int(&d->sysfs, "bma250_rate", ms);
        ...

        /* range */
        ret = d->sysfs.write_int(&d->sysfs, "bma250_range", 2);
        ...

        /* resolution */
        ret = d->sysfs.write_int(&d->sysfs, "bma250_resolution",
                (ms > 50) ? 0 : 1);
        ...

        return ret;
}
```


Case study: integrate on Xperia S

Case study: integrate on Xperia S

SONY
make.believe

Case study: integrate on Galaxy Nexus

SONY
make.believe

Case study: integrate on Galaxy Nexus

- Step 1: What kernel drivers do we have?
- Step 2: Write user-space drivers
- Step 3: Build it
- Step 4: Push it to target

SONY
make.believe

What kernel drivers do we have?

- Step 1: What kernel drivers do we have?
  - Use some application
  - adb shell dmesg
  - adb shell getevent -i
  - We are lucky: look in AOSP code!
- Result:
  - Sharp GP2A proximity and light sensor
  - Bosch BMP180 pressure sensor
  - Invensense MPU3050 Gyroscope
  - BMA250 Accelerometer
  - YAS530 Compass

SONY
make.believe

Writing user-space drivers have

- Step 2: Writing user-space driver
- Create a BMP180 user-space driver
  - Already included in DASH!
  - ..but the driver is not the same.
- Fix 1: input name is different
- Fix 2: sysfs attributes are different
  - Add enable attribute
  - Rename poll rate attribute

SONY
make.believe

Writing user-space drivers have

- Create a Sharp GP2A proximity user-space driver
  - Again - there is already one!
- Fix 1: input name is different
- Fix 2: add handler for sysfs attribute “enable”

SONY
make.believe

Writing user-space drivers have

- Create a Sharp GP2A light user-space driver
- Very similar to Sharp proximity
  - Uses enable sysfs and input device
- Fix 1: Copy-paste proximity user-space driver
- Fix 2: Use ABS_MISC instead of ABS_DISTANCE

SONY
make.believe

Writing user-space drivers have

- Create MPU3050 user-space driver
  - There is a user-space driver.
  - Drivers and lib are incompatible with Sony solution!
- Fix 1: Port Galaxy invensens lib to DASH
  - Update the way DASH interfaces with the new lib
- Fix 2: Add compass and accelerometer
  - SOMC version uses AKM and BMA250 through wrapper layer
- Fix 3: Make it polling

SONY
make.believe

Building DASH

- Step 3: Build it
  - Make sure to enable sensors in makefile

```
SOMC_CFG_SENSORS_PRESSURE_BMP180 := yes
SOMC_CFG_SENSORS_PROXIMITY_SHARP_GP2 := yes
SOMC_CFG_SENSORS_GYRO_MPU3050 := yes
SOMC_CFG_SENSORS_LIGHT_SHARP_GP2 := yes
```

  - mm

SONY
make.believe

Install DASH

- Step 4: Push it to target
  - adb root
  - adb remount
  - adb push $OUT/system/lib/hw/sensors.default.so \
    /system/lib/hw/sensors.tuna.so
  - adb shell stop
  - adb shell start

SONY
make.believe

All done!

SONY
make.believe

Conclusion

- DASH is a...
  - Generic
  - Configurable
  - Scalable
  - Open-source!

...framework for sensors

SONY
make.believe

Questions?

https://github.com/sonyxperiadev/DASH

SONY

make.believe