



# GoDroid

## Application Notes

## Sensor Configuration and Use

v1.1



# Copyright Statement

Copyright in this document is owned by GoWarrior Community. Any person is hereby authorised to view, copy, print and distribute this document subject to the following conditions:

- The document may be used for informational purposes only
- The document may be used for non-commercial purposes only
- Any copy of this document or portion thereof must include this copyright notice

This document is provided "as is" without any warranty of any kind, either express or implied, statutory or otherwise; without limiting the foregoing, the warranties of satisfactory quality, fitness for a particular purpose or non-infringement are expressly excluded and under no circumstances will GoWarrior Community be liable for direct or indirect loss or damage of any kind, including loss of profit, revenue, goodwill or anticipated savings.

This document is intended only to assist the reader in the use of the product. GoWarrior Community makes no representations or warranties with respect to the accuracy or completeness of the contents of this document, which is used at your own risk and should not be relied upon. The information could include technical inaccuracies or typographical errors. No license, whether express, implied, arising by estoppel or otherwise, to any intellectual property right is granted by this document.

The product described in this document is subject to continuous development and improvements. GoWarrior Community also reserves the right to make changes to the specifications and product description at any time without notice.

Third-party brands and names mentioned in this publication are for identification purpose only and may be the property of their respective owners.

Android™ is a registered trademark of Google Inc. Linux® is a registered trademark of Linus Torvalds. Microsoft® and Windows® are registered trademarks of Microsoft Corporation. Supply of this product does not convey a license nor imply a right under any patent, or any other industrial or intellectual property right of Google Inc., Linus Torvalds, and Microsoft Corporation to use this implementation in any finished end-user or ready-to-use final product. It is hereby notified that a license for such use is required from Google Inc., Linus Torvalds and Microsoft Corporation.

For the latest version of this document refer to:

[www.gowarriorosh.com](http://www.gowarriorosh.com)

**Copyright © 2016 GoWarrior Community All Rights Reserved.**

# Table of Contents

<b>Preface .....</b>	<b>1</b>
Overview .....	1
Audience .....	1
Applicable Products.....	1
Reference Documents.....	1
Conventions.....	2
How to Contact Us.....	3
<b>1 Introduction .....</b>	<b>4</b>
<b>2 Driver Porting .....</b>	<b>6</b>
2.1 Modifying Driver .....	6
2.2 Adding MPU6050 Info Structure.....	7
2.3 Testing Sensor Driver .....	8
<b>3 Sensor HAL .....</b>	<b>10</b>
3.1 Adding New Sensor Support in GoDroid .....	10
3.2 Sensor Testing.....	15
<b>4 Open MPU6050 Accel Sensor in GoDroid.....</b>	<b>17</b>
4.1 Hardware Connection .....	17
<b>Revision History .....</b>	<b>19</b>
Document Change History.....	19
Software Changes.....	19

## List of Tables

Table 1. Typographical Conventions.....	2
Table 2. Symbol Conventions .....	2
Table 3. Document Change History .....	19
Table 4. Software Change History.....	19

## List of Figures

Figure 1. Architecture of Android Subsystem.....	4
Figure 2. Sensor Box Installation Page .....	15
Figure 3. Sensor Box Select Page .....	16
Figure 4. Accelerometer Sensor Page.....	16
Figure 5. Connecting MPU6050 to TIGER Board .....	18

# Preface

## Overview

This document mainly describes the adding/porting sensors in GoDroid system. This manual is organized into the following chapters:

- **Chapter 1: Introduction**

This chapter provides information on Android sensor architecture.

- **Chapter 2: Driver Porting**

This chapter gives compact description about how to port sensor driver in GoDroid.

- **Chapter 3: Sensor HAL**

This chapter provides general overview on how to add hardware abstraction layer for new sensor.

- **Chapter 4: Open MPU6050 Accel Sensor in GoDroid**

This chapter introduces how to open MPU6050 Accel Sensor in GoDroid.

## Audience

This manual is primarily written to provide complete guidance for those who wants to exploit GoWarrior TIGER Board, such as makers, tinkers, innovators, students, etc.

## Applicable Products

This manual is applicable for the GoWarrior TIGER Board.

## Reference Documents

- [http://processors.wiki.ti.com/index.php/Android\\_Sensor\\_PortingGuide](http://processors.wiki.ti.com/index.php/Android_Sensor_PortingGuide)
- <http://developer.Android.com>
- <http://developer.android.com/tools/sdk/ndk/index.html>
- <https://source.android.com/source/building-running.html>

## Conventions

### Typographical Conventions

Item	Format
codes, keyboard input commands, file names, equations, and math	Courier New, Size 10.5
Variables, code variables, and code comments	<i>Courier New, Size, Italic</i>
Menu item, buttons, tool names	Ebrima, Size 10.5, Bold e.g. Select USB Debugging
Screens, windows, dialog boxes, and tabs	<b>Ebrima, Size 10.5, Bold</b> <b>Enclosed in double quotation marks</b> e.g. Open the “Debug Configuration” dialog box

Table 1. Typographical Conventions

### Symbol Conventions




Item	Description
 <i>Caution</i>	Indicates a potential hazard or unsafe practice that, if not avoided, could result in data loss, device performance degradation, or other unpredictable results.
 <i>Note</i>	Indicates additional and supplemental information for the main contents.
 <i>Tip</i>	Indicates a suggestion that may help you solve a problem or save your time.

Table 2. Symbol Conventions

## How to Contact Us

Submit all your comments and error reports to the author at:

[info@gowarriorosh.com](mailto:info@gowarriorosh.com)

Tel: +886-2-8752-2000

Fax: +886-2-8751-1001

For questions regarding GoWarrior, contact our support team at the email listed below:

[support@gowarriorosh.com](mailto:support@gowarriorosh.com)

# 1 Introduction

This chapter provides a step-by-step guide on Android subsystem overview. Herein we used MPU6050 accelerometer sensor as an example.

Figure 1 shows Android subsystem overview. The *build/core* directory includes most files of the build system, where *main.mk* is the entry of the whole build system.

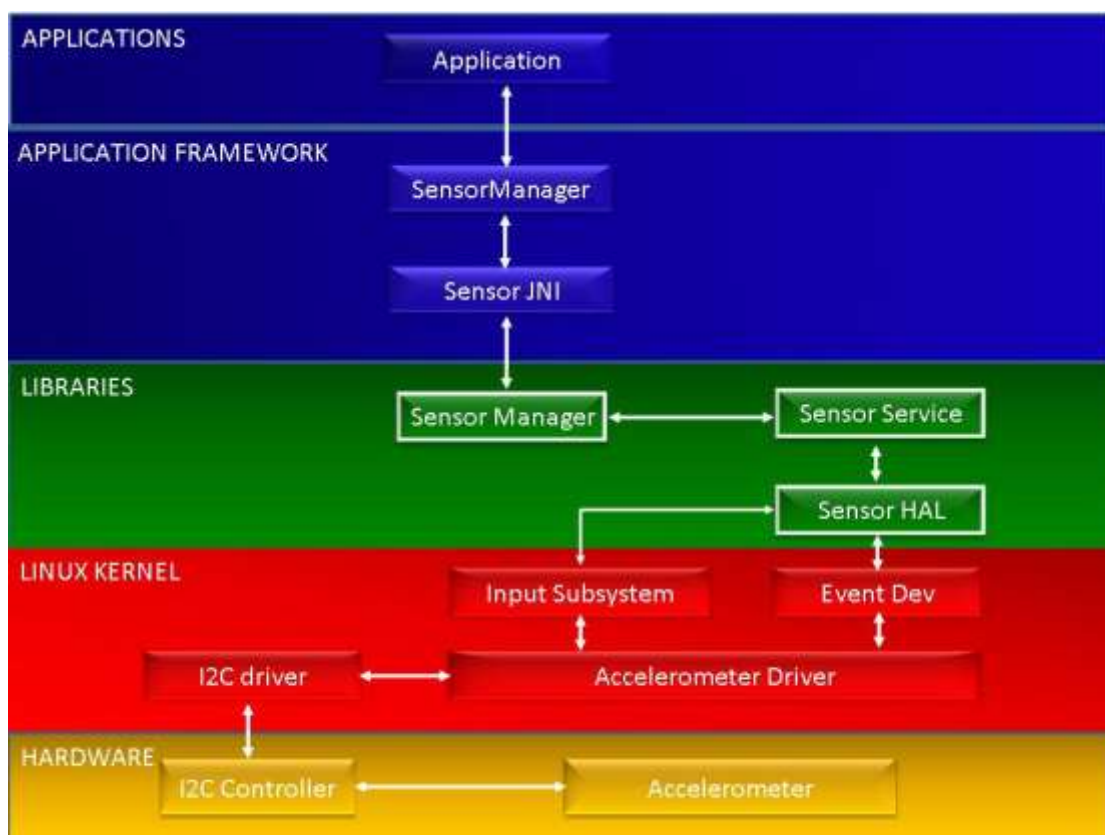


Figure 1. Architecture of Android Subsystem

- **Application Framework:** Sensor Applications uses the Sensor application framework to get the sensor data. It communicates with C++ layer through sensor Java native interface (JNI).
- **Sensor Libraries:** Sensor middle layer mainly consists of Sensor Manager, Sensor service and Sensor hardware abstraction layer.



- **Input Subsystem:** This is a generic Linux framework for all input devices like keyboard, mouse, and touchscreen. It defines a standard set of events. It interfaces to user space through `/sys/class/input` interface.
- **Evdev:** Evdev provides a generic way for input device events to be accessible under `/dev/input/eventX`.
- **MPU6050 Accelerometer Driver:** This driver communicates with MPU6050 accelerometer chip via I2C bus.

## 2 Driver Porting

This chapter summarizes sensor driver porting in GoDroid. Take MPU6050 accelerometer sensor as example.

### 2.1 Modifying Driver

MPU6050 driver needs to have some structure as follow:

```
static const struct i2c_device_id mpu6050_ids[] = {  
    { "mpu6050", 0 },  
    { }  
};  
  
MODULE_DEVICE_TABLE(i2c, mpu6050_ids);  
  
static struct i2c_driver mpu6050_i2c_driver = {  
    .driver      = {  
        .name     = "mpu6050",  
        .owner    = THIS_MODULE,  
        .pm       = &mpu6050_pm,  
    },  
    .probe       = mpu6050_probe,  
    .remove      = mpu6050_remove,  
    .id_table    = mpu6050_ids,  
};  
  
static int __init mpu6050_init(void)  
{  
    int ret;
```

```
ret=i2c_add_driver(&mpu6050_i2c_driver);  
  
return ret;  
  
}  
  
static void __exit mpu6050_exit(void)  
{  
  
    i2c_del_driver(&mpu6050_i2c_driver);  
  
}  
  
module_init(mpu6050_init);  
module_exit(mpu6050_exit);
```

When finding I2C device with name = "mpu6050", the driver will be initialized by running mpu6050\_probe function.

## 2.2 Adding MPU6050 Info Structure

Under

linux\kernel\alidrivers\modules\aliarch\arm\mach-ali3921\ali-s3921.c, we can find codes as follow:

```
static struct i2c_board_info mpu6050_device[] __initdata = {  
  
    {  
  
        I2C_BOARD_INFO("mpu6050", 0x68),  
  
        .platform_data = &mpu6050_pdata,  
  
    },  
  
};  
  
.....  
  
static void __init ali_s3921_init_machine  
(  
  
    void  
  
)
```

```
{  
  
.....  
  
    i2c_register_board_info(1,                                mpu6050_device,  
    ARRAY_SIZE(mpu6050_device));  
  
.....  
  
}
```

Function `i2c_register_board_info` means register a new I2C device in system, in `mpu6050_device` structure, the property "mpu6050" will match driver compatible string shows in [2.1 Modifying](#) which will cause the driver initialization, 0x68 is mpu6050 chip address. Please change this address corresponding to your chip address. And the first parameter in `i2c_register_board_info` function means which I2C bus to select; in this case it means select I2C-1.

## 2.3 Testing Sensor Driver

When device node is ready and driver is registered, probe function will run. And in MPU6050 accelerometer sensor, it will register some devices such as:

```
/sys/devices/platform/ali_i2c_bus.1/i2c-1/1-0068/sensors/MPU6050-accel
```

You can enable MPU6050 accelerometer sensor by command below:

```
echo                                1                                >  
/sys/devices/platform/ali_i2c_bus.1/i2c-1/1-0068/sensors/MPU6050-accel/enable
```

And if MPU6050 has connected to I2C-1 correctly, then by using `getevent` command you will get event data as below:

```
~# getevent  
  
add device 1: /dev/input/event3  
  
    name:      "PixArt USB Optical Mouse"  
  
add device 2: /dev/input/event2  
  
    name:      "ali_ir"
```

```
add device 3: /dev/input/event1
    name:      "gyroscope"
add device 4: /dev/input/event0
    name:      "MPU6050-accel"

/dev/input/event0: 0003 0001 0000000d
/dev/input/event0: 0003 0002 fffffd2e
/dev/input/event0: 0000 0000 00000000
/dev/input/event0: 0003 0000 fffffdbf
/dev/input/event0: 0003 0001 0000000f
/dev/input/event0: 0003 0002 fffffd2d
```

## 3 Sensor HAL

Sensor hardware abstraction layer (HAL) is located in <android source>/hardware/ali/ libsensors directory.

### 3.1 Adding New Sensor Support in GoDroid

Take MPU6050 Accelerometer as example.

Accelerometer sensor class is implemented in `AccelSensor.h` and `AccelSensor.cpp` file.

```
struct input_event;

class MPUAcclSensor : public SensorBase {
    int mEnabled;
    int64_t mDelay;
    InputEventCircularReader mInputReader;
    sensors_event_t mPendingEvent;
    bool mHasPendingEvent;
    char input_sysfs_path[PATH_MAX];
    int input_sysfs_path_len;

    int setInitialState();

public:
    MPUAcclSensor();
    virtual ~MPUAcclSensor();
    virtual int readEvents(sensors_event_t* data, int count);
```

```
virtual bool hasPendingEvents() const;

virtual int setDelay(int32_t handle, int64_t ns);

virtual int setEnable(int32_t handle, int enabled);

virtual int64_t getDelay(int32_t handle);

virtual int getEnable(int32_t handle);

}; include $(call all-makefiles-under,$(LOCAL_PATH))
```

Add new sensor supported in `sensors.cpp` file.

1. Add new sensor to `sSensorList` Structure.

```
static const struct sensor_t sSensorList[] = {

.

.

{ "MPU6050 Accelerometer", "MPU6050", 1,

SENSORS_ACCELERATION_HANDLE,

SENSOR_TYPE_ACCELEROMETER, ACCEL_MPU6050_RANGE,

ACCEL_MPU6050_RESOLUTION, ACCEL_MPU6050_POWER, 10000, 0, 0, { } },

};
```

2. Create object of new sensor.

```
sensors_poll_context_t::sensors_poll_context_t()

{

mSensors[acc] = new MPUAcclSensor();

...

.

mPollFds[acc].fd = mSensors[acc]->getFd();

mPollFds[acc].events = POLLIN;

mPollFds[acc].revents = 0;
```

```
int wakeFds[2];

int result = pipe(wakeFds);

LOGE_IF(result<0, "error creating wake pipe (%s)",
strerror(errno));

fcntl(wakeFds[0], F_SETFL, O_NONBLOCK);

fcntl(wakeFds[1], F_SETFL, O_NONBLOCK);

mWritePipeFd = wakeFds[1];


mPollFds[wake].fd = wakeFds[0];

mPollFds[wake].events = POLLIN;

mPollFds[wake].revents = 0;
```

3. Update `sensors_poll_context_t` structure and add new sensor case to `handleToDriver` function.

```
struct sensors_poll_context_t {

    struct sensors_poll_device_t device; // must be first

    .

    .

private:

    enum {

        accel          = 0,

        // gyro          = 1,

        numSensorDrivers,

        numFds,

    };

int sensors_poll_context_t::handleToDriver(int handle) {

    switch (handle) {

        case ID_A:
```



```
        return acc;

case ID_M:
```

#### 4. Android HAL modification

This type of calibration can be achieved in Android HAL by modifying `readEvents ()` function in

`/hardware/ali/libensors/MPUAcclSensor.cpp` file

For MPU6050 driver send input event as flowing:

```
static void mpu6050_accel_work_fn(struct work_struct *work)
{
    struct mpu6050_sensor *sensor;
    u32 shift;
    ktime_t timestamp;
    .
    .
    .
    input_report_abs(sensor->accel_dev, ABS_X,
        (sensor->axis.x >> shift));
    input_report_abs(sensor->accel_dev, ABS_Y,
        (sensor->axis.y >> shift));
    input_report_abs(sensor->accel_dev, ABS_Z,
        (sensor->axis.z >> shift));
    .
    .
}

static inline void input_report_abs(struct input_dev *dev, unsigned int
code, int value)
```

```
{  
  
    input_event(dev, EV_ABS, code, value);  
  
}
```

We need to readEvents as follow:

```
int MPUAcclSensor::readEvents(sensors_event_t* data, int count)  
{  
    if (count < 1)  
        return -EINVAL;  
  
    if (mHasPendingEvent) {  
        mHasPendingEvent = false;  
        mPendingEvent.timestamp = getTimestamp();  
        *data = mPendingEvent;  
        return mEnabled ? 1 : 0;  
    }  
  
    .  
    .  
  
    while (count && mInputReader.readEvent(&event)) {  
        int type = event->type;  
        if (type == EV_ABS) {  
            float value = event->value;  
            if (event->code == EVENT_TYPE_ACCEL_X) {  
                mPendingEvent.acceleration.x =  
MPU_UNIT_CONVERSION(value);  
            } else if (event->code == EVENT_TYPE_ACCEL_Y) {  
                mPendingEvent.acceleration.y =  
MPU_UNIT_CONVERSION(value);  
            }  
        }  
    }  
}
```

```

        } else if (event->code == EVENT_TYPE_ACCEL_Z) {

            mPendingEvent.acceleration.z =
MPU_UNIT_CONVERSION(value);

        }

    }

//sensors.h
...

#define EVENT_TYPE_ACCEL_X      ABS_X
#define EVENT_TYPE_ACCEL_Y      ABS_Y
#define EVENT_TYPE_ACCEL_Z      ABS_Z

```

## 3.2 Sensor Testing

When sensor driver and HAL have been added, the best way to test is use some applications. As example can use Sensor BOX:



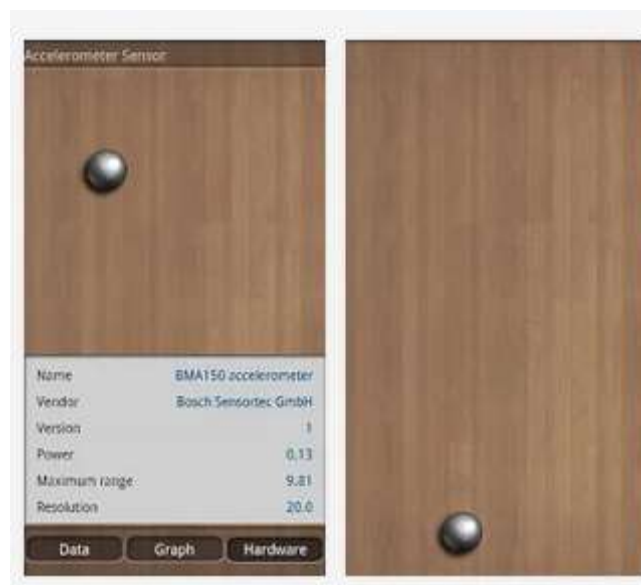
**Figure 2. Sensor Box Installation Page**

Once you will run this application, it will identify which sensor is supported.



**Figure 3. Sensor Box Select Page**

Click accelerometer sensor icon, to test MPU6050 sensor.



**Figure 4. Accelerometer Sensor Page**

## 4 Open MPU6050 Accel Sensor in GoDroid

To open MPU6050 Accel Sensor in GoDroid, it just requires changing following item in `/device/gowarrior/tigerboard/BoardConfig.mk` to true.

```
BOARD_HAVE_MPU6050_SENSOR:=false
```

After changing this item, rebuild the system and burn the image, and then restart the system. You will see the `mpu6050.ko` under `/system/modules`, and it will be `insmod` at init time, see `init.tigerboard.rc` as follow:

```
on fs
    .....

    insmod /system/modules/mpu6050.ko

    .....
```

### 4.1 Hardware Connection

As previously been mentioned in [2.2 Adding MPU6050 Info Structure.](#), MPU6050 chip uses I2C bus to interface with the. TIGER Board. MPU6050 has been configured to I2C-1 in GoDroid kernel code, so it is necessary to connect MPU6050's pins (VCC, GND, SCL, SDA) with corresponding pins of the TIGER Board expansion connector (J3), which are PIN1 ( 3.3v), PIN9 (Ground), PIN5 (I2C1-SCL), PIN3 (I2C1-SDA) respectively, as shown in Figure 5 below:



**Figure 5. Connecting MPU6050 to TIGER Board**

After Software configuration and hardware connection, next we can test MPU6050 with the help of SensorBox.

# Revision History

## Document Change History

Revision	Changes	Date
v1.1	Updated document version to 1.1.	February 29, 2016
v1.0	Initial Release	September 07, 2015

Table 3. Document Change History

## Software Changes

Revision	Changes	Date
v1.1	Install and start GoDroid with a MicroSD card.	February 29, 2016
v1.0	Initial Release	September 07, 2015

Table 4. Software Change History



[www.gowarriorsh.com](http://www.gowarriorsh.com)

Headquarters

Tel: +886-2-8752-2000

Fax: +886-2-8751-1001

