WEP Warwick Evans Publishing

International Journal of Computer Science and Information Technology

ISSN: 3005-9682 (Print), ISSN: 3005-7140 (Online) | Volume 3, Number 2, Year 2024
DOI: https://doi.org/10.62051/ijcsit.v3n2.23
Journal homepage: https://wepub.org/index.php/IJCSIT/index

Data Reading Based on the IIO Subsystem Framework

Wei Pang

College of Electronic Information, Southwest Minzu University, Cheng Du, China

ABSTRACT

This document outlines how to read data from the ICM-20608 sensor using the i.MX 6ULL application processor within the IIO framework on a Linux system. The i.MX 6ULL is an application processor from NXP Semiconductors based on the ARM Cortex-A7 architecture, designed specifically for low-power and high-performance embedded applications. The ICM-20608 is a high-performance six-axis sensor that integrates a three-axis gyroscope and a three-axis accelerometer, widely used in motion detection and attitude estimation.

KEYWORDS

Iio Subsystem; I.MX6ULL; ICM20608; Data Reading; Embedded

1. INTRODUCTION

With the explosion of smartphones, IoT devices, industrial IoT, and wearable devices, the demand for sensors continues to grow rapidly. For example, sensors such as accelerometers, light sensors, gyroscopes, barometers, magnetometers found in smartphones or smart wristbands are essentially ADCs [1-2]. If we refer to the manuals of these sensors, we find that they all have an ADC internally. Sensors provide I2C or SPI interfaces for external communication, allowing SoCs to obtain ADC values from the sensors through these interfaces to obtain desired measurement results [3-4]. To manage the increasing number of ADC-type sensors, the Linux kernel introduced the IIO subsystem, which has been developed by Jonathan Cameron and the Linux-IIO community since 2009 [5].

This article focuses on the i.MX 6ULL chip from NXP Semiconductors, which utilizes the ARM Cortex-A7 core, employing the IIO subsystem framework to retrieve data from the ICM20608 chip, including its three-axis accelerometer, three-axis gyroscope, and temperature readings.

2. IIO SUBSYSTEM FRAMEWORK

2.1. Overview of the IIO Subsystem Framework

Long ago, support for the above hardware was scattered across various parts of the Linux source code. The advent of IIO provided a unified framework for accessing and controlling various types of sensors, and standardized interfaces for user-space applications to access sensors: sysfs/devfs. It also bridged the gap between the Hwmon and Input subsystems [6].

In the IIO subsystem, device drivers are responsible for direct interaction with hardware. They read data from or write data to hardware, and convert this data into appropriate formats for user-space programs to use. The IIO subsystem provides a set of kernel APIs for drivers to utilize. These APIs allow drivers to register devices, configure device parameters, handle interrupts, and transfer data between devices. The IIO subsystem offers a standardized set of interfaces for user space, typically

located in the /sys/bus/iio filesystem. Through this unified interface, developers can simplify device driver development and application design without needing to design separate interfaces for each device. A simplified structural diagram is shown in the figure.

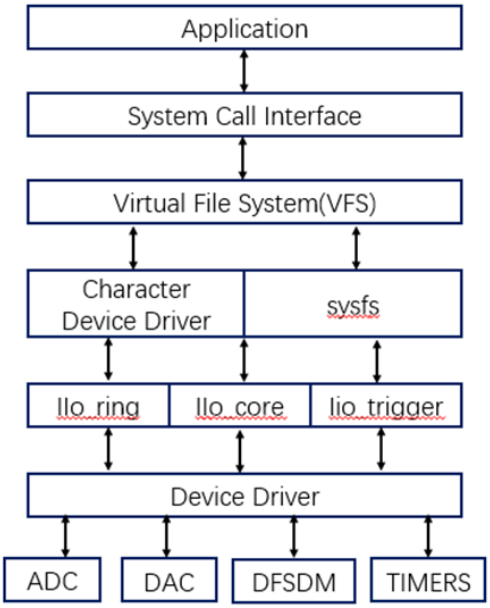

Figure 1. Structure Diagram of the IIO Subsystem

2.2. Introduction to the Iiolib

libiio is an open-source library initiated by Analog Devices for accessing IIO (Industrial I/O) devices. It encapsulates access to /sys/bus/iio/devices (for configuring IIO) and /dev/iio/deviceX (for reading and writing IIO), and provides convenient IIO command-line tools for testing (such as iio_info and iio_readdev), as well as an iiod server. The iiod server includes local and remote backends to support access from both local clients and remote clients. IIO supports various standard Linux device access interfaces including char device, sysfs, configfs, and debugfs.

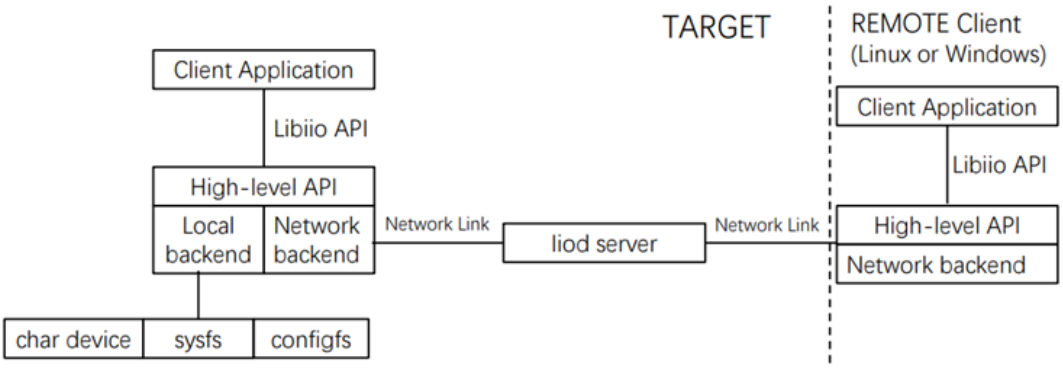

Figure 2. Iibiio-overview

Apart from user-space applications accessing IIO devices, there are also other device drivers within the kernel that utilize the IIO subsystem's APIs to write device drivers compatible with their own frameworks. For instance, once a certain SoC ADC is supported within the IIO subsystem, various hardware devices can be connected to this ADC channel. A typical example is a touch chip, where developers need to write a touch driver within the input subsystem framework. In the touch driver's IRQ handler, developers call IIO in-kernel APIs to read the X and Y values of the touchscreen.

3. CORE CONTROLLER

The i.MX 6ULL is an application processor developed by NXP Semiconductors based on the ARM Cortex-A7 architecture, with a maximum clock frequency of up to 528 MHz. It is known for its low power consumption and high efficiency, suitable for a wide range of embedded applications including industrial control, IoT devices, smart home appliances, and medical devices.

It supports external DDR3/DDR3L and LPDDR2 memory, with a maximum capacity of up to 1 GB. Multimedia processing capabilities include a 24-bit LCD controller with a maximum resolution of WXGA (1366x768) and basic graphics acceleration, but it does not support advanced multimedia processing such as video decoding acceleration.

Storage interfaces include NAND, NOR, eMMC, and SD card interfaces, providing flexible storage solutions. Network connectivity features an integrated 10/100 Ethernet controller, suitable for industrial-grade Ethernet applications.Additionally, it supports various peripheral interfaces such as USB 2.0, UART, I2C, SPI, and CAN buses, facilitating connectivity with external devices.

4. ICM20608

The ICM-20608 is a 6-axis MEMS sensor produced by InvenSense, consisting of 3-axis accelerometer and 3-axis gyroscope functionalities. Both the gyroscope and accelerometer feature 16-bit ADCs, and when using the SPI interface, communication speeds can reach up to 8 MHz. The 3-axis orientation of the ICM-20608 is depicted in Figure 3.

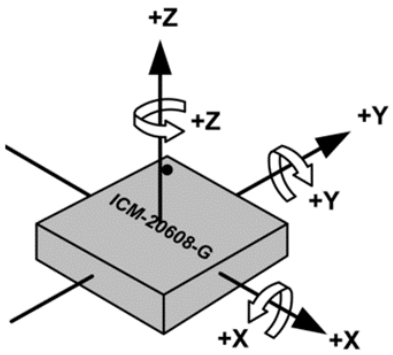

Figure 3. ICM-20608 Axis Orientation and Polarity Detection

The key features of the ICM-20608 are as follows:

(1) Gyroscope supports X, Y, and Z-axis outputs, with internal integrated 16-bit ADCs. Measurement ranges can be set to ±250, ±500, ±1000, and ±2000°/s.

(2) Accelerometer supports X, Y, and Z-axis outputs, with internal integrated 16-bit ADC. Measurement ranges can be set to ±2g, ±4g, ±8g, and ±16g.

(3) User-programmable interrupts.

(4) Internal 512-byte FIFO.

(5) Built-in digital temperature sensor.

(6) Shock resistance up to 10,000g.

(7) Supports fast I2C with speeds up to 400 kHz.

(8) Supports SPI with speeds up to 8 MHz.

The hardware schematic is shown in Figure 4. Here, the ECSPI3_SCLK of the ICM20608 is connected to the UART_RXD pin of the i.MX6ULL, ECSPI3_MOSI is connected to the

UART2_CTS pin, ECSPI3_SSO is connected to the UART2_TXD pin, and ECSPI3_MISO is connected to the UART2_RTS pin.

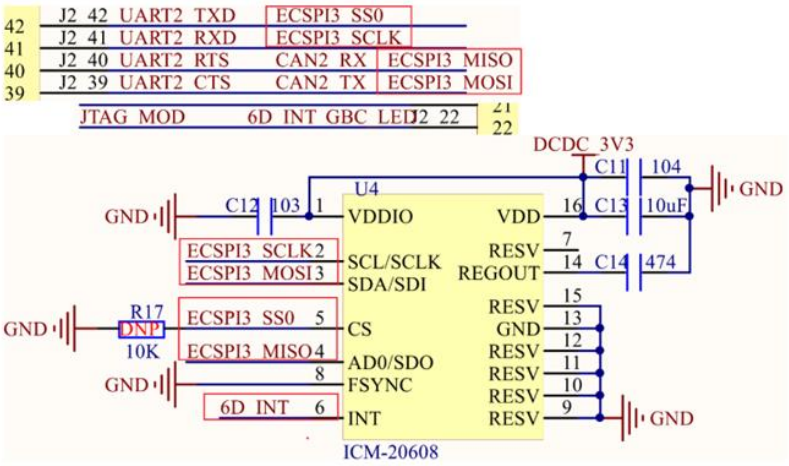

Figure 4. Pin Connection Diagram

5. SENSOR DATA READING TEST

First, load the corresponding IIO driver for the sensor. This can typically be done by executing the modprobe command with the .ko file corresponding to the sensor. Once the device driver is loaded, a device node will be created under the /sys/bus/iio/devices/ directory. We can directly obtain the sensor data by reading files under the iio node. For example, executing cat /sys/bus/iio/devices/iio/in_accel_x_raw will read the acceleration data in the X-axis direction. Device parameters can also be set by writing to configuration files. For example, echo 1000 > /sys/bus/iio/devices/iio/in_accel_scale will configure the sensor's resolution. Alternatively, user-space programs can be written to poll and read sensor data directly. The results are shown in Figure 5.

```
Raw valule:
gx = 11, gy = 11, gz = 0
ax = 30, ay = -40, az = 2070
temp = 2969
Actual Value:act gx = 0.67°/S, act gy = 0.67°/S, act gz = 0.00°/S
act ax = 0.01g, act ay = -0.02g, act az = 1.01g
act temp = 34.09°C
```

Figure 5. Output Result Diagram

6. CONCLUSION

In embedded systems, adopting the IIO subsystem allows developers to utilize its unified interface, data buffering, efficient transmission, various triggering mechanisms, and multi-device synchronization advantages to simplify driver development and data acquisition processes, thereby improving system performance and development efficiency. Additionally, the wide device support, ease of integration and expansion, and rich user-space tools of the IIO subsystem make it an ideal choice for handling industrial input and output devices in embedded systems.

REFERENCES

[1] Madieu J. Linux Device Drivers Development: Develop Customized Drivers for Embedded Linux [M]. Packt Publishing Ltd, 2017.

[2] Trivedi N, Patel H, Chauhan D. Fundamental structure of Linux kernel based device driver and implementation on Linux host machine [J]. International Journal of applied Information Systems, 2016, 10(4): 2249-0868.

[3] Madieu J. Linux Device Driver Development: Everything you need to start with device driver development for Linux kernel and embedded Linux [M]. Packt Publishing Ltd, 2022.

[4] Gawali A R. Modeling and Synthesis of Linux DMA Device Drivers using HOL4 [D]. Virginia Tech, 2024.

[5] Song C H, Kyung J H, Eun S. A Plug&Play Scheme of GPIO Sensors/Actuators in Linux Platforms[C]//2024 International Conference on Information Networking (ICOIN). IEEE, 2024: 705-707.

[6] Li Z, Wang J, Sun M, et al. Securing the device drivers of your embedded systems: framework and prototype[C]//Proceedings of the 14th International Conference on Availability, Reliability and Security. 2019: 1-10.