Linux Device Driver for Gyroscope

Sahana H.G ,Shaheen
Department of Electrical and Electronics
Birla Institute of Technology and Science Pilani, Goa Campus–403726

Device drivers are software interfaces between software applications and hardware devices. As part of complex operating system, device drivers are considered extremely difficult to develop. It is a software program that controls a particular type of hardware device that is attached to a computer. The device driver developers must have an in depth understanding of given hardware and software platforms. This paper presents a device driver for gyroscope L3G4200D GY50.

***Keywords*—Device driver, Linux, code generation, embedded software, Gyroscope, Raspberry Pi.**

I. INTRODUCTION

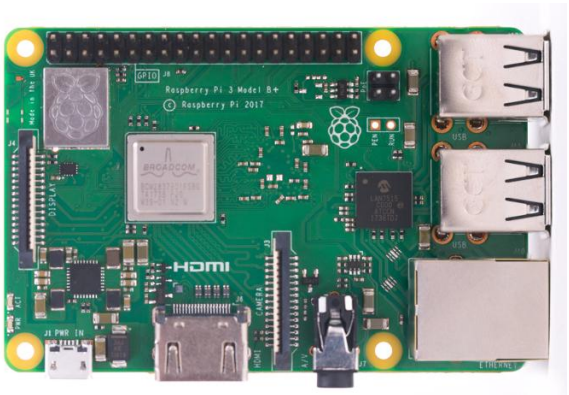
The L3G4200D is a low-power three-axis angular rate sensor able to provide unprecedented stablility of zero rate level and sensitivity over temperature and time. It includes a sensing element and an IC interface capable of providing the measured angular rate to the external world through a digital interface (I2C/SPI). The sensing element is manufactured using a dedicated micro-machining process developed by STMicroelectronics to produce inertial sensors and actuators on silicon wafers. The IC interface is manufactured using a CMOS process that allows a high level of integration to design a dedicated circuit which is trimmed to better match the sensing element characteristics. The L3G4200D has a full scale of ±250/±500/ ±2000 dps and is capable of measuring rates with a user-selectable bandwidth. The L3G4200D is available in a plastic land grid array (LGA) package and can operate within a temperature range of -40 °C to +85 °C.

The role of a driver is to provide mechanisms which allows normal user to access protected parts of its system, in particular ports, registers and memory addresses normally managed by the operating system. One of the good features of Linux is the ability to extend at runtime the set of the features offered by the kernel. Users can add or remove functionalities to the kernel while the system is running. These “programs” that can be added to the kernel at runtime are called “module” and built into individual files with .ko (Kernel object) extension. The Linux kernel takes advantages of the possibility to write kernel drivers as modules which can be uploaded on request.

This method has different advantages: The kernel can be highly modularized, in order to be compatible with the most possible hardware. A kernel module can be modified without need of recompiling the full kernel.

The Linux kernel offers support for different classes of device modules: “char” devices are devices that can be accessed as a stream of bytes (like a file) “block” devices are accessed by filesystem nodes (example: disks).“network” interfaces are able to exchange data with other hosts. A device driver contains at least two functions: A function for the module initialization (executed when the module is loaded with the command "insmod"). A function to exit the module (executed when the module is removed with the command "rmmod") These two are like normal functions in the driver, except that these are specified as the init and exit functions, respectively, by the macros module init() and module exit(), which are defined in the kernel header module.h.

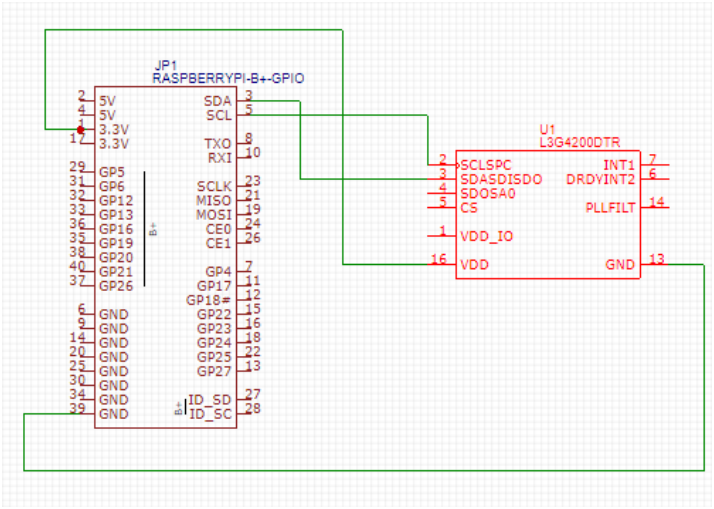
The Raspberry Pi is a low cost, credit-card sized computer that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. It is a capable little device that enables people of all ages to explore computing, and to learn how to program in languages like Scratch and Python. It’s capable of doing everything you’d expect a desktop computer to do, from browsing the internet and playing high-definition video, to making spreadsheets, word-processing, and playing games.



II. METHODOLOGY

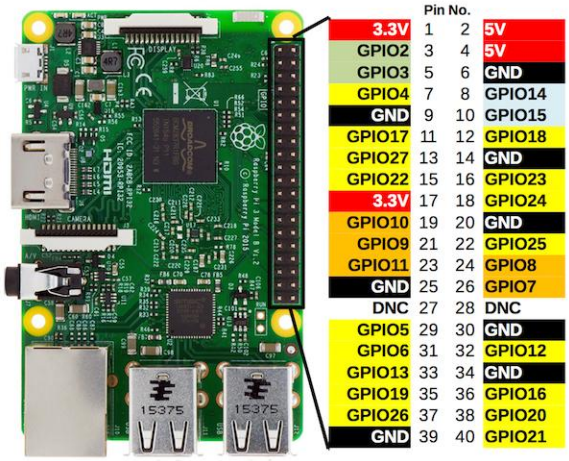
The GY50 gyroscope is connected to the RaspberryPi through I2C bus. There are 3 internal registers in the GY50 which holds the gyroscope data for the 3 Cartesian axises.
There are 5 control registers, which are used to configure the power mode, enabling / disabling High pass filter, configure interrupts, etc.

Hardware Schematic :



Following are the pins used for this project :

SCL : 5
SDA : 3
GND : 6
VCC : 1



These are connected to the corresponding pins of the gyroscope.

All the development of the driver was done on the Raspberry Pi itself. Access to a terminal in the Raspberry Pi was done using SSH, Putty Terminal and XMing Server for GUI and the following extra packages were installed using the following command:
sudo apt-get install linux-headers-$(uname -r).

The driver is initialized with adxl_init() function. It creates a driver handle at /dev/gyro, acquires handle for the i2c bus, configures the gyroscope. The following code acquires the i2c bus:

```
my_adap = i2c_get_adapter(1); // 1 means i2cbus
if(!(my_clienti2c_new_dummy(my_adap, 0x69))) {
printk(KERN_INFO "Couldn't acquire i2c slave");
unregister_chrdev(MAJOR_NO, DEVICE_NAME);
device_destroy(cl, first);
class_destroy(cl);
return -1;
        }
```

The I2C bus is handled using I2C_SMBUS drivers. This is a subset of I2C protocol developed by Intel and is widely supported by most of the peripheral chips. The access to the gyroscope is verified by reading the DEVID register of the gyroscope. Then the gyroscope is configured to start the measurement.

Firstly, CTRL_REG1 is configured to enable X,Y,Z and turn off power down by writing 0b00001111 into the CTRL_REG1 register. Next, CTRL_REG2 can be configured to enable High Pass filter operation. CTRL_REG3 is configured to generate data ready interrupt on INT2. CTRL_REG4 enables us to choose the full-scale range +/-(250dps or 500dps or 2000dps). We have configured to be +/-2000dps by setting the appropriate bits.Then, the output X,Y,Z data registers are read out using the smbus_read function and the MSB and LSB registers are concatenated to obtain the angular acceleration values.

The file read operation is handle by the following code:

```
static ssize_t my_read(struct file *f, char __user *buf, size_t len, loff_t *off) {
printk(KERN_INFO "Driver read()\n");
axis_data[0] = adxl_read(my_client, ((REG_OUT_X_H << 8) | REG_OUT_X_L));
axis_data[1] = adxl_read(my_client, ((REG_OUT_Y_H << 8) | REG_OUT_Y_L));
axis_data[2] = adxl_read(my_client, ((REG_OUT_Z_H << 8) | REG_OUT_Z_L));
if (*off == 0){
        if (copy_to_user(buf, &axis_data,3) != 0){
        printk(KERN_INFO "Driver read: Inside if\n");
        return -EFAULT;}
        else {
        return 3;
        }
        }
        else return 0; }
```

When a user space application does a read operation on the driver handle, above piece of code is called. It copies the data
from the axis_data array in the kernel space to userspace using copy_to_user() and returns the number of bytes copied.

The kernel module is the compile using the command *'make all'*

It is then loaded using the following command: *sudo insmod gyro.ko*
Finally, the driver can be tested using the demo python script.
sudo python script.py

Python script :

```
# Script to display gyroscope data.
from time import sleep
def convert(x):
  if x>=128:
  x= x - 256
   return x
fd = open('/dev/gyro','r')
while(1):
   data = fd.read(3);
   foo = [ord(i) for i in data]
   foo = [convert(i) for i in foo]
   print (foo)
   sleep(0.5)
```

III. RESULT

Linux device driver for GY50 gyroscope interrupt support has been developed. Also a userspace application to demo the feature has been developed using python.

Hardware implementation :

The *dmesg* output:

```
pi@raspberrypi: ~/Desktop/proj
m), (N/A)
[  +0.000013] cfg80211:   (57240000 KHz - 63720000 KHz @ 2160000
Bm), (N/A)
[  +0.218525] systemd-journald[133]: Received request to flush r
rom PID 1
[  +1.673899] random: nonblocking pool is initialized
[  +0.227212] uart-pl011 3f201000.uart: no DMA platform data
[  +1.356573] Adding 102396k swap on /var/swap.  Priority:-1 ext
5280k SSFS
[ +28.780250] Bluetooth: Core ver 2.21
[  +0.000089] NET: Registered protocol family 31
[  +0.000012] Bluetooth: HCI device and connection manager initi
[  +0.000024] Bluetooth: HCI socket layer initialized
[  +0.000017] Bluetooth: L2CAP socket layer initialized
[  +0.000036] Bluetooth: SCO socket layer initialized
[  +0.020809] Bluetooth: BNEP (Ethernet Emulation) ver 1.3
[  +0.000019] Bluetooth: BNEP filters: protocol multicast
[  +0.000023] Bluetooth: BNEP socket layer initialized
[Apr27 17:20] Welcome to LG34200D GY50 Gyroscope!
[  +0.006550] Gyroscope detected, value = 211
[  +0.006803] X axis : 147
[  +0.000011] Y axis : 36
[  +0.000008] Z axis : 99
```

Output of the python script :

```
pi@raspberrypi: ~/Desktop/proj
pi@raspberrypi:~/Desktop/proj $ sudo insmod gyro.ko
pi@raspberrypi:~/Desktop/proj $ sudo python script.py
[-64, 119, -55]
[-99, 9, -26]
[22, -47, -62]
[-92, -69, -19]
[-30, -8, -39]
[60, -90, -81]
[120, -64, -16]
[-30, -78, 35]
[-52, 108, -94]
[-114, -39, -92]
[66, -22, 33]
[4, 23, 32]
[11, -99, 12]
[25, -109, -107]
[-38, -101, -14]
[57, 74, 91]
[-13, -112, -4]
[-57, 9, -111]
[76, 12, -34]
[-62, 102, 22]
[23, -95, -49]
[76, -2, -54]
```

IV. CONCLUSION

In this paper, the implementation of linux device driver for a gyroscope is explained. It has been tested and demonstrated on Raspberry Pi connected to an L3G4200D GY50 gyroscope running linux kernel versionL4.4.70.

V. REFERENCES

[1] www.kernel.org
[2] www.derekmolly.ie