

## Implementación de Manejadores de Dispositivos

# Acelerómetro MPU9250 con I<sup>2</sup>C

**Alumno: Pablo Daniel Folino**

**Docente: Gonzalo Sanchez**

2021

---

### Índice

Registro de cambios.....	2
Trabajo Final mpu9250_i2c_driver.c.....	3
A) Generar un nuevo Device Tree.....	3
B) Conectar el MPU9250.....	6
C) Generar el módulo para la MPU9250.....	7

## Registro de cambios

Revisión	Cambios realizados	Fecha
1.0	Creación del documento	13/11/2021
1.1	Generación del Device Tree	23/11/2021
1.2	Documentación de Hardware	25/11/2021

## Trabajo Final mpu9250 i2c driver.c

### A) Generar un nuevo Device Tree

Para generar el nuevo Device Tree se hizo:

- 1) Abrir una terminal en la PC y [setear las variables de entorno](#)

```
$ export PATH=$PATH:$HOME/ISO_II/toolchain/arm-mse-linux-gnueabi/bin  
$ export CROSS_COMPILE=arm-linux-  
$ export ARCH=arm
```

- 2) Dirigirse al directorio:

```
/home/pablo/ISO_II/linux_kernel/linux-stable/arch/arm/boot/dts
```

Y copiar `m335x-evm.dts` a `am335x-evm-boneblack-Folino.dts`

- 3) Editar el archivo `am335x-evm-boneblack-Folino.dts`

**Nota:** el `m335x-evm.dts` ya tiene configurado los pines para la `pinmux_i2c1_pins`

Ir a la sección `&i2c1` y agregar nuestro driver, el archivo queda:

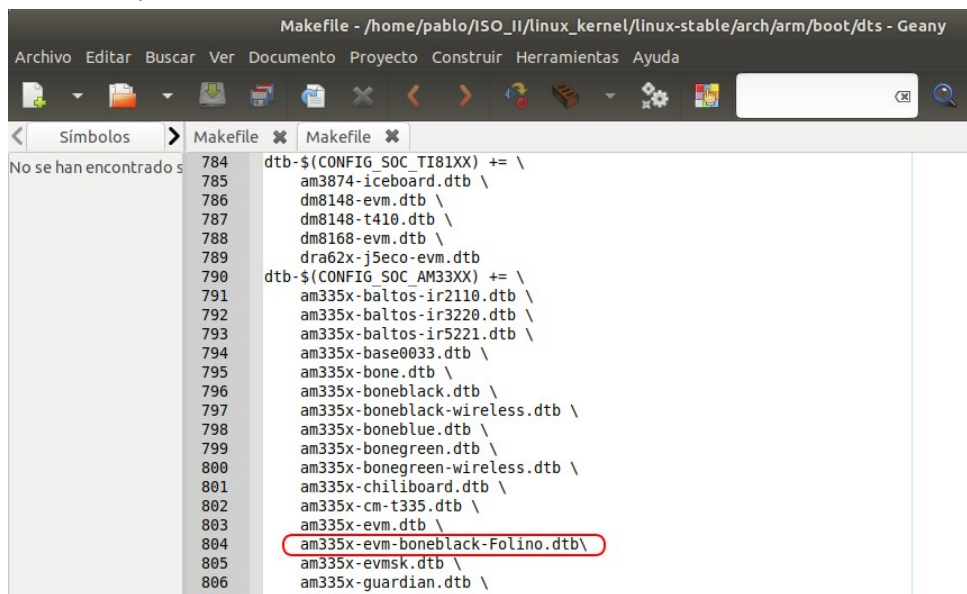
```
&i2c1 {  
    pinctrl-names = "default";  
    pinctrl-0 = <&i2c1_pins>;  
  
    status = "okay";  
    clock-frequency = <100000>;  
  
    my_mpu9250: my_mpu9250@68 {  
        compatible = "mse,my_mpu9250";  
        reg = <0x68>;  
        status = "okay";  
    };  
  
    lis331dlh: lis331dlh@18 {  
        compatible = "st,lis331dlh", "st,lis3lv02d";  
        reg = <0x18>;  
        Vdd-supply = <&lis3_reg>;  
        Vdd_IO-supply = <&lis3_reg>;  
  
        st.click-single-x;
```

4) En el directorio:

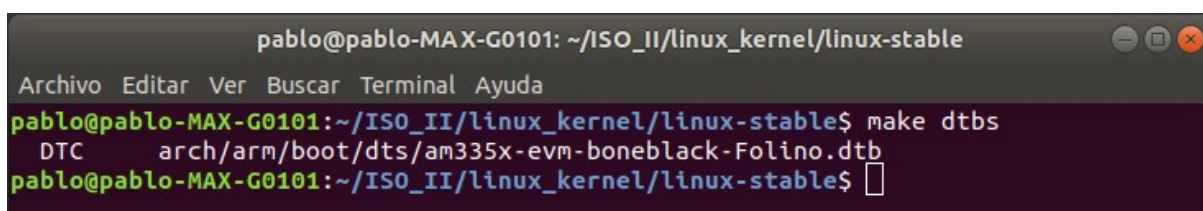
**/home/pablo/ISO\_II/linux\_kernel/linux-stable/arch/arm/boot/dts**

Editar el archivo Makefile, ir a la sección **tb-\$(CONFIG\_SOC\_AM33XX)** y agregar nuestro dts (**am335x-evm-boneblack-Folino.dts**)

El archivo queda:



5) Parados en el directorio **/home/pablo/ISO\_II/linux\_kernel/linux-stable/** se ejecutó un **make dtbs** para compilar solamente los Device Tree.



6) Copiar el **am335x-evm-boneblack-Folino.dtb** al directorio (servidor NFS)

**/home/pablo/var/lib/tftpboot.**

7) Resetear la **SBC**, y ejecutar la siguiente secuencia de inicio, en el **GTKTerm**:

```
=> setenv ipaddr 192.168.0.100
=> setenv serverip 192.168.0.50
=> tftp 0x81000000 zImage
=> tftp 0x82000000 am335x-evm-boneblack-Folino.dtb
=> setenv bootargs console=ttyS0,115200n8
=> setenv bootargs root=/dev/nfs rw ip=192.168.0.100 console=ttyS0,115200n8
nfsroot=192.168.0.50:/home/pablo/ISO_II/nfsroot,nfsvers=3
=> bootz 0x81000000 - 0x82000000
```

```

GtkTerm - /dev/ttyUSB0 115200-8-N-1
File Edit Log Configuration Controlsignals View Help
=> setenv ipaddr 192.168.0.100
=> setenv serverip 192.168.0.50
=> tftp 0x81000000 zImage
link up on port 0, speed 100, full duplex
Using ethernet@4a100000 device
TFTP from server 192.168.0.50; our IP address is 192.168.0.100
Filename 'zImage'.
Load address: 0x81000000
Loading: #####
#####
#####
#####
#####
6.7 MiB/s
done
Bytes transferred = 4554936 (4580b8 hex)
=> tftp 0x82000000 am335x-evm-boneblack-Folino.dtb
link up on port 0, speed 100, full duplex
Using ethernet@4a100000 device
TFTP from server 192.168.0.50; our IP address is 192.168.0.100
Filename 'am335x-evm-boneblack-Folino.dtb'.
Load address: 0x82000000
Loading: #####
5.7 MiB/s
done
Bytes transferred = 65867 (1014b hex)
=> setenv bootargs console=ttyS0,115200n8
=> setenv bootargs root=/dev/nfs rw ip=192.168.0.100 console=ttyS0,115200n8 nfsroot=192.1
68.0.50:/home/pablo/ISO_II/nfsroot,nfsvers=3
=> bootz 0x81000000 - 0x82000000 █

/dev/ttyUSB0 115200-8-N-1 DTR RTS CTS CD DSR RI

```

8) Cuando aparece **buildroot login:** escribir **root**, y el sistema estándar listo para poder trabajar en la SBC.

```

GtkTerm - /dev/ttyUSB0 115200-8-N-1
File Edit Log Configuration Controlsignals View Help

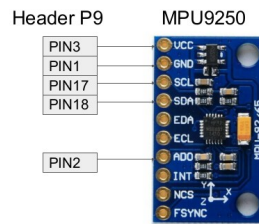
Welcome to Buildroot
buildroot login: [ 37.605084] vbat: disabling
[ 37.607928] lis3_reg: disabling
[ 37.611086] v1_8d: disabling
[ 37.613979] v3_3d: disabling
[ 37.616924] wlan-en-regulator: disabling

Welcome to Buildroot
buildroot login: root
# ls
char_driver          ioctl_usuario        timer
class_driver         misc_driver          wait_queue
hello_platform_driver simple_platform_driver
hello_world          sleep

```

## B) Conectar el MPU9250

Conectar al **Header P9** de la Beagle Bone Black el módulo:



**PIN\_OUT** de la BeagleBoneBlack

P9				P8			
GND	1	2	GND	GND	1	2	GND
3.3V	3	4	3.3V	GPIO1_6	3	4	GPIO1_7
5V Raw	5	6	5V Raw	GPIO1_2	5	6	GPIO1_3
5V	7	8	5V	GPIO2_2	7	8	GPIO2_3
	9	10		GPIO2_5	9	10	GPIO2_4
Serial4_RX/GPIO0_30	11	12	GPIO1_28	eQEP2bB/GPIO1_13	11	12	GPIO1_12/eQEP2bA
Serial4_TX/GPIO0_31	13	14	GPIO1_18/PWM1A	PWM2B/GPIO0_23	13	14	GPIO0_26
GPIO1_16	15	16	GPIO1_19/PWM1B	GPIO1_15	15	16	GPIO1_14
I2C1_SCL/SPI0_CS0/GPIO0_17	17	18	GPIO0_4/SPI0_MOSI/I2C1_SDA	GPIO0_27	17	18	GPIO2_1
I2C2_SCL/GPIO0_13	19	20	GPIO0_12/I2C2_SDA	PWM2A/GPIO0_22	19	20	GPIO1_31
Serial2_TX/SPI0_MISO/GPIO0_3	21	22	GPIO0_2/Serial2_RX/SPI0_SCLK	GPIO1_30	21	22	GPIO1_5
GPIO1_17	23	24	GPIO0_15/Serial1_TX	GPIO1_4	23	24	GPIO1_1
GPIO3_21	25	26	GPIO0_14/Serial1_RX	GPIO1_0	25	26	GPIO1_29
eQEP0B/GPIO3_19	27	28	GPIO3_17/SPI1_CS0	GPIO2_22	27	28	GPIO2_24
SPI1_MISO/GPIO3_15	29	30	GPIO3_16/SPI1_MOSI	GPIO2_23	29	30	GPIO2_25
SPI1_SCLK/GPIO3_14	31	32	VDD_ADC	GPIO0_10	31	32	GPIO0_11
AIN4	33	34	GND_ADC	eQEP1B/GPIO0_9	33	34	GPIO2_17
AIN6	35	36	AIN5	eQEP1A/GPIO0_8	35	36	GPIO2_16
AIN2	37	38	AIN3	Serial5_TX/GPIO2_14	37	38	GPIO2_15/Serial5_RX
AIN0	39	40	AIN1	GPIO2_12	39	40	GPIO2_13
GPIO0_20	41	42	GPIO0_7/SPI1_CS1/eQEP0A	eQEP2A/GPIO2_10	41	42	GPIO2_11/eQEP2B
GND	43	44	GND	GPIO2_8	43	44	GPIO2_9
GND	45	46	GND	GPIO2_6	45	46	GPIO2_27

## Beaglebone Black Pinout Diagram

P9				P8			
Function	Physical Pins		Function	Function	Physical Pins		Function
DGND	1	2	DGND	DGND	1	2	DGND
VDD 3.3 V	3	4	VDD 3.3 V	MMC1_DAT6	3	4	MMC1_DAT7
VDD 5V	5	6	VDD 5V	MMC1_DAT2	5	6	MMC1_DAT3
SYS 5V	7	8	SYS 5V	GPIO_66	7	8	GPIO_67
PWR_BUT	9	10	SYS_RESET	GPIO_69	9	10	GPIO_68
UART4_RXD	11	12	GPIO_60	GPIO_45	11	12	GPIO_44
UART4_TXD	13	14	EHRPWM1A	EHRPWM2B	13	14	GPIO_26
GPIO_48	15	16	EHRPWM1B	GPIO_47	15	16	GPIO_46
SPIO_CS0	17	18	SPIO_D1	GPIO_27	17	18	GPIO_65
I2C2_SCL	19	20	I2C_SDA	EHRPWM2A	19	20	MMC1_CMD
SPIO_DO	21	22	SPIO_SCLK	MMC1_CLK	21	22	MMC1_DAT5
GPIO_49	23	24	UART1_TXD	MMC1_DAT4	23	24	MMC1_DAT1
GPIO_117	25	26	UART1_RXD	MMC1_DAT0	25	26	GPIO_61
GPIO_115	27	28	SPI1_CS0	LCD_VSYNC	27	28	LCD_PCLK
SPI1_DO	29	30	GPIO_112	LCD_HSYNC	29	30	LCD_AC_BIAS
SPI1_SCLK	31	32	VDD_ADC	LCD_DATA14	31	32	LCD_DATA15
AIN4	33	34	GND_ADC	LCD_DATA13	33	34	LCD_DATA11
AIN6	35	36	AIN5	LCD_DATA12	35	36	LCD_DATA10
AIN2	37	38	AIN3	LCD_DATA8	37	38	LCD_DATA9
AIN0	39	40	AIN1	LCD_DATA6	39	40	LCD_DATA7
GPIO_20	41	42	ECAPWMO	LCD_DATA4	41	42	LCD_DATA5
DGND	43	44	DGND	LCD_DATA2	43	44	LCD_DATA3
DGND	45	46	DGND	LCD_DATA0	45	46	LCD_DATA1

**LEGEND**

- Power, Ground, Reset
- Digital Pins
- PWM Output
- 1.8 Volt Analog Inputs
- Shared I2C Bus
- Reconfigurable Digital



## C) Generar el módulo para la MPU9250

Para generar el nuevo Device Tree se hizo:

- 1) Abrir una terminal en la PC y setear las variables de entorno
- 2) Crear( o ir) el siguiente directorio;

```
$ mkdir -p -v /home/pablo/ISO_II/nfsroot/root/mpu9250_i2c_driver  
$ cd /home/pablo/ISO_II/nfsroot/root/mpu9250_i2c_driver
```

- 3) Se crea un archivo **mpu9250\_i2c\_driver.c** con el siguiente contenido:

```
#include <linux/module.h>  
#include <linux/miscdevice.h>  
#include <linux/i2c.h>  
#include <linux/fs.h>  
#include <linux/of.h>  
//#include <linux/uaccess.h>  
  
static char message[256] = {0};          //< Memory for the string that is  
passed from userspace  
  
/* Private device structure */  
struct mse_dev {  
    struct i2c_client *client;  
    struct miscdevice mse_miscdevice;  
    char name[9]; /* msdrvXX */  
  
};  
  
/*  
 * Definicion de los ID correspondientes al Device Tree. Estos deben ser  
 * informados al kernel mediante la macro MODULE_DEVICE_TABLE  
 *  
 * NOTA: Esta seccion requiere que CONFIG_OF=y en el kernel  
 */  
  
/* "mse,my_mpu9250" es el nombre que machea con el Device Tree*/  
static const struct of_device_id mse_dt_ids[] = {  
    { .compatible = "mse,my_mpu9250", },  
    { /* sentinel */ }  
};  
  
MODULE_DEVICE_TABLE(of, mse_dt_ids);
```

```
/* User is reading data from /dev/msedrvXX */
static ssize_t mse_read(struct file *file, char __user *userbuf, size_t count,
loff_t *ppos) {

    /* Instancio un puntero a la estructura */
    struct mse_dev *mse;

    int error_count = 0;
    int Ret;

    /* Se usa para obtener la estructura completa */
    mse = container_of(file->private_data, struct mse_dev, mse_miscdevice);

    /* Aqui ira las llamadas a i2c_transfer() que correspondan pasando
    * como dispositivo mse->client*/
    //pr_info("mse_read() fue invocada.");

    Ret = i2c_master_recv(mse->client, message, count);
    // copy_to_user has the format ( * to, *from, size) and returns 0 on success
    error_count = copy_to_user(userbuf, message, count);

    if (error_count==0){                // Todo OK
        return 0;
    }
    else {
        pr_info("MPU9250: Error al enviar %d caracteres al usuario\n",
        error_count);
        return -1;
    }
}

static ssize_t mse_write(struct file *file, const char __user *buffer, size_t len,
loff_t *offset) {

    struct mse_dev *mse;
    int error_count = 0;

    mse = container_of(file->private_data, struct mse_dev, mse_miscdevice);

    /* Aqui ira las llamadas a i2c_transfer() que correspondan pasando
    * como dispositivo mse->client*/

    error_count = copy_from_user(message ,buffer, len);
    error_count = i2c_master_send(mse->client,message,len);

    //pr_info("mse_write() fue invocada.");
    return len;
}
```



```
static long mse_ioctl(struct file *file, unsigned int cmd, unsigned long arg) {
    struct mse_dev *mse;

    mse = container_of(file->private_data, struct mse_dev, mse_miscdevice);

    /* Aqui ira las llamadas a i2c_transfer() que correspondan pasando
     * como dispositivo mse->client
     */

    pr_info("my_dev_ioctl() fue invocada. cmd = %d, arg = %ld\n", cmd, arg);
    return 0;
}

/* declaracion de una estructura del tipo file_operations */
static const struct file_operations mse_fops = {
    .owner = THIS_MODULE,
    .read = mse_read,
    .write = mse_write,
    .unlocked_ioctl = mse_ioctl,
};

/*-----*/
static int mse_probe(struct i2c_client *client, const struct i2c_device_id *id) {
    struct mse_dev *mse;
    static int counter = 0;
    int ret_val;

    pr_info("MPU9250: Inicializando el driver...\n");

    /* Allocate new private structure */
    mse = devm_kzalloc(&client->dev, sizeof(struct mse_dev), GFP_KERNEL);

    /* Store pointer to the device-structure in bus device context */
    i2c_set_clientdata(client, mse);

    /* Store pointer to I2C client device in the private structure */
    mse->client = client;

    /* Initialize the misc device, mse is incremented after each probe call */
    sprintf(mse->name, "mse%02d", counter++);

    mse->mse_miscdevice.name = mse->name;
    mse->mse_miscdevice.minor = MISC_DYNAMIC_MINOR;
    mse->mse_miscdevice.fops = &mse_fops;

    /* Register misc device */
    ret_val = misc_register(&mse->mse_miscdevice);
    if (ret_val != 0) {
        pr_err("No se pudo registrar el dispositivo %s\n", mse-
            >mse_miscdevice.name);
        return ret_val;
    }

    pr_info("Dispositivo %s: minor asignado: %i\n", mse->mse_miscdevice.name,
        mse->mse_miscdevice.minor);

    return 0;
}
```

```
static int mse_remove(struct i2c_client * client) {
    struct mse_dev * mse;

    pr_info("MPU9250: Removiendo el driver...\n");
    /* Get device structure from bus device context */
    mse = i2c_get_clientdata(client);

    /* Deregister misc device */
    misc_deregister(&mse->mse_miscdevice);

    return 0;
}

/*-----*/

static struct i2c_driver mse_driver = {

    .probe= mse_probe,
    .remove= mse_remove,
    .driver = {
        .name = "mse_driver",
        .owner = THIS_MODULE,
        .of_match_table = of_match_ptr(mse_dt_ids),
    },
};

/*-----*/

/*****
 * Esta seccion define cuales funciones seran las ejecutadas al cargar o
 * remover el modulo respectivamente. Es hecho implicitamente,
 * termina declarando init() exit()
 *****/
module_i2c_driver(mse_driver);

/*****
 * Seccion sobre Informacion del modulo
 *****/
MODULE_AUTHOR("Pablo Daniel Folino <pfolinos@gmail.com>");
MODULE_LICENSE("GPL");
MODULE_DESCRIPTION("Este modulo es un driver para MSE-IMD-2021");
MODULE_INFO(mse_imd, "Basado en el Drive de Gonzalo Sanchez");
MODULE_VERSION("1.1");
```

4) Crear el archivo Makefile, con el siguiente contenido:

```
ifneq ($(KERNELRELEASE),)
obj-m := mpu9250_i2c_driver.o
else
    KDIR := $(HOME)/ISO_II/linux_kernel/linux-stable
all:
    $(MAKE) -C $(KDIR) M=$$PWD
endif
```

5) Compilar con **make**.

6) Desde la terminal ir al directorio **mpu9250\_i2c\_driver** e instalar el módulo con **insmod mpu9250\_i2c\_driver.ko**

```

GtkTerm - /dev/ttyUSB0 115200-8-N-1
File Edit Log Configuration Controlsignals View Help
mpu9250_i2c_driver.ko
# insmod mpu9250_i2c_driver.ko
[ 87.253087] mpu9250_i2c driver: loading out-of-tree module taints kernel.
[ 87.260954] MPU9250: Inicializando el driver...
[ 87.265893] Dispositivo mse00: minor asignado: 60
[ 87.284283] omap_gpio 44e07000.gpio: Could not set line 6 debounce to 200000 microseconds (-22)
[ 87.293152] sdhci-omap 48060000.mmc: Got CD GPIO
[ 87.298115] sdhci-omap 48060000.mmc: supply vqmmc not found, using dummy regulator

```

7) También se debe observar que el driver se registró a si mismo dentro del platform bus:

**\$ ls -l /sys/bus/i2c/drivers/**

```

GtkTerm - /dev/ttyUSB0 115200-8-N-1
File Edit Log Configuration Controlsignals View Help
# ls -l /sys/bus/i2c/drivers/
total 0
drwxr-xr-x  2 root    root          0 Jan  1 00:00 dummy
drwxr-xr-x  2 root    root          0 Jan  1 00:00 lp872x
drwxr-xr-x  2 root    root          0 Jan  1 00:00 lp873x
drwxr-xr-x  2 root    root          0 Jan  1 00:00 lp87565
drwxr-xr-x  2 root    root          0 Jan  1 00:00 menelaus
drwxr-xr-x  2 root    root          0 Jan  1 00:01 mse_driver
drwxr-xr-x  2 root    root          0 Jan  1 00:00 palmas
drwxr-xr-x  2 root    root          0 Jan  1 00:00 pcf857x
drwxr-xr-x  2 root    root          0 Jan  1 00:00 tps65023
drwxr-xr-x  2 root    root          0 Jan  1 00:00 tps65217
drwxr-xr-x  2 root    root          0 Jan  1 00:00 tps65218
drwxr-xr-x  2 root    root          0 Jan  1 00:00 tps65910
drwxr-xr-x  2 root    root          0 Jan  1 00:00 twl
drwxr-xr-x  2 root    root          0 Jan  1 00:00 twl6040
#
/dev/ttyUSB0 115200-8-N-1
DTR RTS CTS CD DSR RI

```

**\$ ls -l /sys/bus/i2c/devices**

```

GtkTerm - /dev/ttyUSB0 115200-8-N-1
File Edit Log Configuration Controlsignals View Help
# ls -l /sys/bus/i2c/devices
total 0
lrwxrwxrwx  1 root    root          0 Jan  1 00:00 0-002d -> ../../../../devices/platform/ocp/44c00000.interconnect/44c00000.interconnect:segment@200000/44e0b000.target-module/44e0b000.i2c/i2c-0/0-002d
lrwxrwxrwx  1 root    root          0 Jan  1 00:00 1-0018 -> ../../../../devices/platform/ocp/48000000.interconnect/48000000.interconnect:segment@0/4802a000.target-module/4802a000.i2c/i2c-1/1-0018
lrwxrwxrwx  1 root    root          0 Jan  1 00:00 1-001b -> ../../../../devices/platform/ocp/48000000.interconnect/48000000.interconnect:segment@0/4802a000.target-module/4802a000.i2c/i2c-1/1-001b
lrwxrwxrwx  1 root    root          0 Jan  1 00:00 1-0039 -> ../../../../devices/platform/ocp/48000000.interconnect/48000000.interconnect:segment@0/4802a000.target-module/4802a000.i2c/i2c-1/1-0039
lrwxrwxrwx  1 root    root          0 Jan  1 00:00 1-0048 -> ../../../../devices/platform/ocp/48000000.interconnect/48000000.interconnect:segment@0/4802a000.target-module/4802a000.i2c/i2c-1/1-0048
lrwxrwxrwx  1 root    root          0 Jan  1 00:00 1-0068 -> ../../../../devices/platform/ocp/48000000.interconnect/48000000.interconnect:segment@0/4802a000.target-module/4802a000.i2c/i2c-1/1-0068
lrwxrwxrwx  1 root    root          0 Jan  1 00:00 i2c-0 -> ../../../../devices/platform/ocp/44c00000.interconnect/44c00000.interconnect:segment@200000/44e0b000.target-module/44e0b000.i2c/i2c-0
lrwxrwxrwx  1 root    root          0 Jan  1 00:00 i2c-1 -> ../../../../devices/platform/ocp/48000000.interconnect/48000000.interconnect:segment@0/4802a000.target-module/4802a000.i2c/i2c-1
#
/dev/ttyUSB0 115200-8-N-1
DTR RTS CTS CD DSR RI

```

```
$ ls /sys/bus/i2c/drivers/mse_driver
```

```
GtkTerm - /dev/ttyUSB0 115200-8-N-1
File Edit Log Configuration Controlsignals View Help
# ls /sys/bus/i2c/drivers
dummy      lp87565    palmas     tps65217   twl
lp872x     menelaus   pcf857x    tps65218   twl6040
lp873x     mse_driver tps65023    tps65910
# ls /sys/bus/i2c/drivers/mse_driver
1-0068 bind module uevent unbind
#
```

8) De la misma forma debe poder visualizarse el módulo dentro de `/sys` y el dispositivo registrado dentro de la clase correspondiente al framework utilizado:

```
$ ls -l /sys/module/mpu9250_i2c_driver/drivers
```

```
GtkTerm - /dev/ttyUSB0 115200-8-N-1
File Edit Log Configuration Controlsignals View Help
# ls -l /sys/module/mpu9250_i2c_driver/drivers
total 0
lrwxrwxrwx    1 root    root          0 Jan  1 00:14 i2c:mse_driver -> ../../../../bus/i2c/drivers/mse_driver
#
```

```
$ ls /sys/class/misc
```

```
GtkTerm - /dev/ttyUSB0 115200-8-N-1
File Edit Log Configuration Controlsignals View Help
# ls /sys/class/misc
cpu_dma_latency mse00 vga_arbiter
loop-control ubi_ctrl
# ls /sys/class/misc/mse00
dev power subsystem uevent
#
```

9) Al haberse creado la entrada en el sysfs, udev se encarga de crear el file device en `devtmpfs`

```
$ ls -l /dev
```

```
GtkTerm - /dev/ttyUSB0 115200-8-N-1
File Edit Log Configuration Controlsignals View Help
# ls -l /dev | grep mse
crw-rw----    1 root    root      10,  60 Jan  1 00:01 mse00
#
```

10) En una terminal en la PC ir al directorio:

```
cd /home/pablo/ISO_II/nfsroot/root/mpu9250_i2c_driver
```

11) Crear en ese directorio el archivo `mpu9250_test.c`;

```
/
*****
***** Archivo de prueba del Módulo MPU9250 *****
*****
/

#include <stdio.h>
#include <sys/ioctl.h>
#include <fcntl.h>
#include <unistd.h>

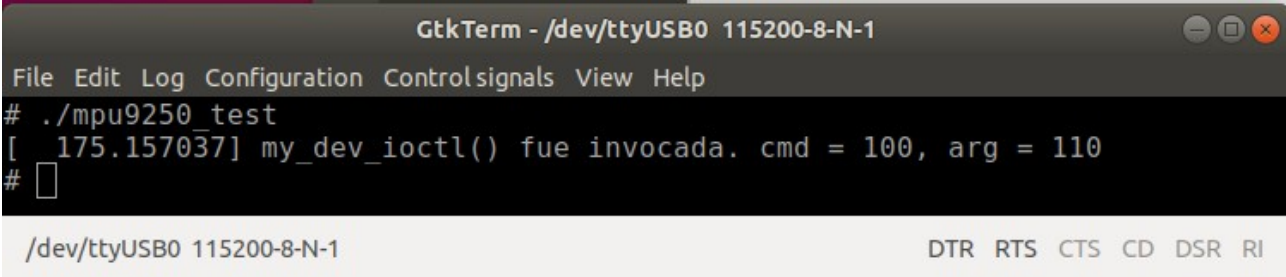
int main(void)
{
    int my_dev;
    my_dev = open("/dev/mse00", 0);
    if (my_dev < 0) {
        perror("Fail to open device file: /dev/mse00.");
    }
    else {
        ioctl(my_dev, 100, 110); /* cmd = 100, arg = 110. */
        close(my_dev);
    };
    return 0;
}
```

12) Compilarlo:

```
arm-linux-gcc -o mpu9250_test mpu9250_test.c
```

13) Ejecutar el archivo, desde la consola en la **SBC**:

```
# ./mpu9250_test
```



```
GtkTerm - /dev/ttyUSB0 115200-8-N-1
File Edit Log Configuration Controlsignals View Help
# ./mpu9250 test
[ 175.157037] my_dev_ioctl() fue invocada. cmd = 100, arg = 110
# 
/dev/ttyUSB0 115200-8-N-1 DTR RTS CTS CD DSR RI
```

14) Remover el módulo

```
$ rmmod mpu9250_i2c_driver
```

```

GtkTerm - /dev/ttyUSB0 115200-8-N-1
File Edit Log Configuration Controlsignals View Help
# rmmod mpu9250_i2c_driver
[ 342.630398] MPU9250: Removiendo el driver...
#

```

/dev/ttyUSB0 115200-8-N-1 DTR RTS CTS CD DSR RI

15) En Internet se busca un header en donde especifiquen los registros del MPU9250.h

16) Se adapta y reescribe un programa para testear el driver en Lenguaje C.

```

GtkTerm - /dev/ttyUSB0 115200-8-N-1
File Edit Log Configuration Controlsignals View Help
Gir6scopo:      (-0.041546, -0.034089, 0.004261) [rad/s]
Aceler6metro:   (-1.048714, -0.172391, 7.245224) [m/s2]
Temperatura:    22.803097 [C]

```

/dev/ttyUSB0 115200-8-N-1 DTR RTS CTS CD DSR RI

Programa principal:

```

test.c
--/ISO_11/nfsroot/root/mpu9250_i2c_driver
Guardar

// Returns the die temperature, C
float mpu9250GetTemperature_C( void )
{
    return control._t;
}

int main(){
    int ret = 0;
    printf("Starting device test code example...\n");
    fd = open("/dev/mse00", O_RDWR); // Open the device with read/write access
    if (fd < 0){
        perror("Failed to open the device...");
        return errno;
    }
    mpu9250Init( MPU9250_ADDRESS_0 );
    while(1){
        mpu9250Read();
        system("clear");
        printf( "Gir6scopo:      (%f, %f, %f) [rad/s]\r\n",
            mpu9250GetGyroX_rads(),
            mpu9250GetGyroY_rads(),
            mpu9250GetGyroZ_rads()
        );
        printf( "Aceler6metro:   (%f, %f, %f) [m/s2]\r\n",
            mpu9250GetAccelX_mss(),
            mpu9250GetAccelY_mss(),
            mpu9250GetAccelZ_mss()
        );
        printf( "Temperatura:    %f [C]\r\n\r\n",
            mpu9250GetTemperature_C()
        );
        usleep(2000000);
    }
    printf("End of the program\n");
    return 0;
}

```

C Anchura del tabulador: 8 Ln 344, Col 27 INS