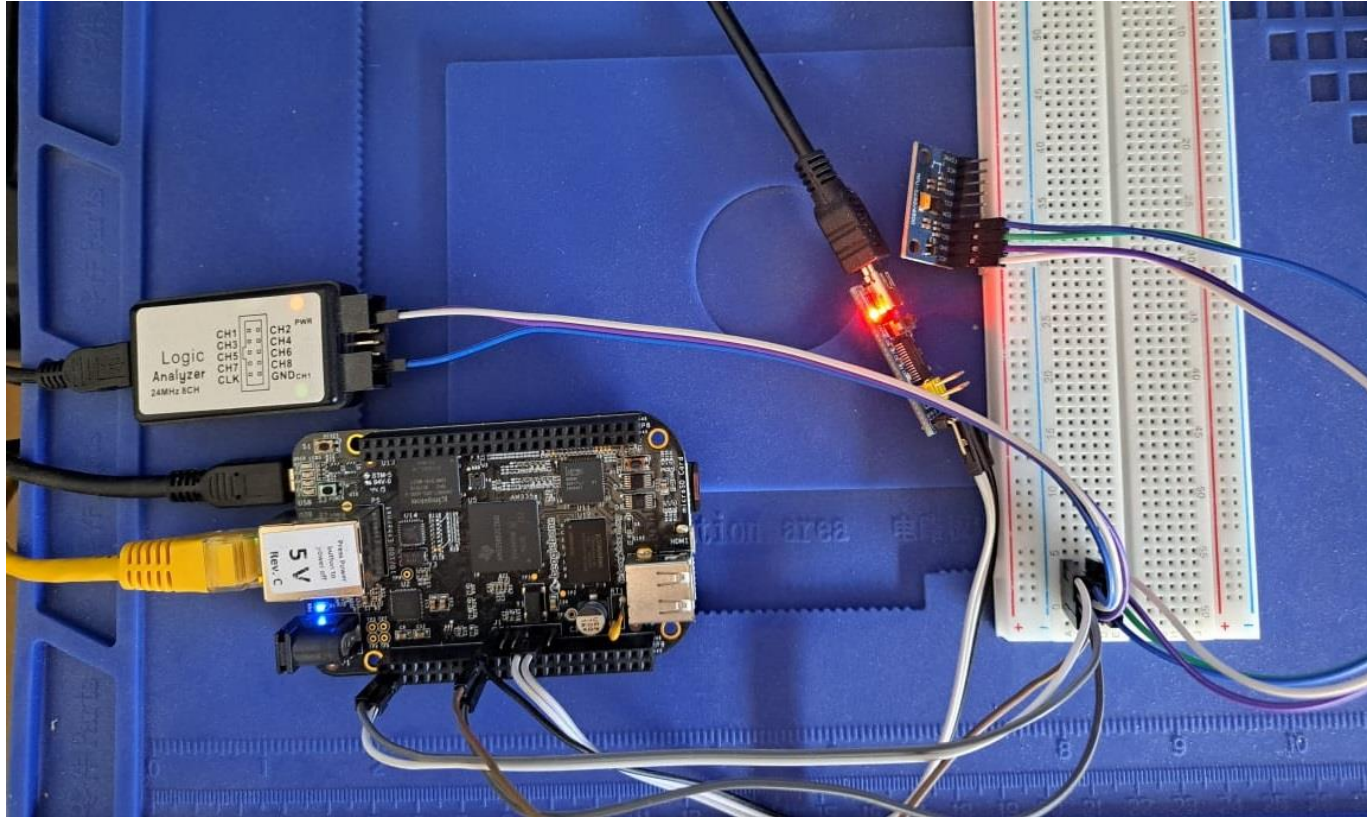


Maestría en sistemas embebidos
Implementación de Manejadores de Dispositivos

DRIVER I2C EN BEAGLEBONE BLACK MPU9250

Autor: Ing Junior Monroy Guerrero

Implementacion del hardware



Sensor MPU-9250



1. Este módulo contiene salida ADC de 16 bits de acelerador de 3 ejes y giroscopio de 3 ejes
2. Voltaje de potencia: 3 - 5V
3. Modo de comunicación: protocolo de comunicación I2C/SPI
4. Rango de giroscopio: +/-250, +/-500, +/-1000, +/-2000 dps
5. Rango de acelerador: +/-2G, +/-4G, +/-8G, +/-16G
6. Registro: 1MHz
7. SPI / I2C: 400KH

6A	106	USER_CTRL	R/W	-	FIFO_EN	I2C_MST_EN	I2C_IF_DIS	-	FIFO_RST	I2C_MST_RST	SIG_COND_RST
6B	107	PWR_MGMT_1	R/W	H_RESET	SLEEP	CYCLE	GYRO_STANDBY	PD_PTAT	CLKSEL[2:0]		
		-		-							
41	65	TEMP_OUT_H	R	TEMP_OUT_H[15:8]							
42	66	TEMP_OUT_L	R	TEMP_OUT_L[7:0]							
1A	26	CONFIG	R/W	-	FIFO_MODE	EXT_SYNC_SET[2:0]			DLPF_CFG[2:0]		
1B	27	GYRO_CONFIG	R/W	XGYRO_Ct_en	YGYRO_Ct_en	ZGYRO_Ct_en	GYRO_FS_SEL [1:0]		-	FCHOICE_B[1:0]	
1C	28	ACCEL_CONFIG	R/W	ax_st_en	ay_st_en	az_st_en	ACCEL_FS_SEL[1:0]		-		

Pasos para la descarga en Buildroot

1. Se descargo configuro y compilo Buildroot
2. Se copiaron los archivos de arranque en la memoria sd del al beaglebone
3. Se determino que faltaban librerías las cuales fueron copiadas de la carpeta salida del buildroot
4. Se procedió al inicio de la placa junto con sus comandos de arranque

```
setenv ipaddr 192.168.0.100
setenv serverip 192.168.0.2
setenv bootargs root=/dev/nfs rw ip=192.168.0.100 console=ttyS0,115200n8 nfsroot=192.168.0.2:/home/junior/ISO_II/nfsroot,nfsvers=3
saveenv
tftp 0x81000000 zImage
tftp 0x82000000 am335x-boneblack.dtb
bootz 0x81000000 - 0x82000000
```

COMPILACION Y MODULOS CON CROSSCOMPILADOR BUILDROOT

```
export PATH=$HOME/IMD/buildroot/output/host/bin:$PATH
```

```
arm-linux-gcc -o test_read_write_i2c test_read_write_i2c.c
```

junior@Desktop: ~/ISO_II/nfsroot/root/IMD_modulos/mpu9250

```
50$ export PATH=$HOME/IMD/buildroot/output/host/bin:$PATH
50$ arm-linux-gcc -o test_read_write_i2c test_read_write_i2c.c
50$
```

Copiar

```
scp /home/junior/IMD/buildroot/output/host/bin /home/junior/ISO_II/nfsroot/buildroot
```

```
export PATH=$PATH:$HOME/IMD/buildroot/output/host/bin
```

```
export CROSS_COMPILE=arm-linux-
```



```
export ARCH=arm
```

```
make
```

junior@Desktop: ~/ISO_II/nfsroot/root/IMD_modulos/mpu9250

```
50$ export PATH=$PATH:$HOME/IMD/buildroot/output/host/bin
50$ export CROSS_COMPILE=arm-linux-
50$ export ARCH=arm
50$ make
```

COMPILACION DE DTS EN BUILDROOT

am335x-boneblack.dts					
Name			Size	Location	
 am335x-boneblack.dts			3.3 kB	output/build/uboot-2022.04/arch/arm/dts	
 am335x-boneblack.dts			3.3 kB	output/build/linux-5.15.35/arch/arm/boot/dts	

SE MODIFICAN LOS .dts

En la raiz de Buildroot de compila make linux-rebuild 2>&1 | tee build.log asi modifica solo los dts

```
},
&am33xx_pinmux {
    i2c1_pins: pinmux_i2c1_pins {
        pinctrl-single,pins = <
            AM33XX_IOPAD(0x958, PIN_INPUT_PULLUP | MUX_MODE2) /* spi0_d1.i2c1_sda */
            AM33XX_IOPAD(0x95c, PIN_INPUT_PULLUP | MUX_MODE2) /* spi0_cs0.i2c1_scl */
        >;
    };
};

/*Habilitamos el bus I2C1*/
&i2c1 {
    pinctrl-names = "default";
    pinctrl-0 = <&i2c1_pins>;
    status = "okay";
    clock-frequency = <400000>;

    mympu9250: mympu9250@68 {
        compatible = "mse,mympu9250";
        reg = <0x68>;
    };
};
```

Read

```
/* **** */
static ssize_t dev_read(struct file *filep, char *buffer, size_t len, loff_t *offset){

    int CountError = 0;

    int ret;

    /* El offset esta configurado cuando se hace la funcion de lseek */
    pr_info("\n### i2c Inicia direccion MPU9250 0x%02X ###", (char)(*offset));

    /*Seteo el Address a partir del cual quiero escribir*/
    ADDRESS[0] = (char)(*offset);

    ret = i2c_master_send(modClient, ADDRESS, 1);

    if(ret < 0){

        printk("\n### KERN_INFO i2C No se pudo configurar el registro para leer los datos desde el MPU9250 ###");

    }

    pr_info("\n### i2c Cantidad de bytes configurados para leer = %d ###", len);

    ret = i2c_master_recv(modClient, message, len);

    if(ret != len){

        printk("\n### KERN_INFO i2C No se pudieron recibir %0d bytes desde el MPU9250 ###", len);

    }

    /* La idea es mandarle datos al usuario */
    CountError = copy_to_user(buffer, message, ret);

    if (CountError==0){          /* si no hay error, envia el dato */

        printk("\n### KERN_INFO Char Sent %d characters to the user ###", ret);

        return (ret=0);          /* Limpia la posicion de inicio y retorna a la posicion de inicio*/

    }

    else {

        printk("\n### KERN_INFO Char Failed to send %d characters to the user ###", CountError);

    }

}
```

Read



```
58.914471]
58.914471] ### 6lseek en ejecucion mympu9250 ###
** LECTURA I2C DE I2C_REG_TEMP 0x41 **
58.921015]
58.921015] ### i2c Inicia direccion MPU9250 0x41 ###
58.927678]
58.927678] ### i2c Cantidad de bytes configurados para leer = 2 ###
** SE LEERAN 2 bytes**[ 58.938335]
58.938335] ### 6Char Sent 2 characters to the user ###

** REGISTRO=0x41 --> Valor: 0x0C **
** REGISTRO=0x42 --> V[ 58.946442]
58.946442] ### 6lseek en ejecucion mympu9250 ###
lor: 0xB0 ***
** TEMPERATURA GRADOS :30.67 ***

** ESCRI[ 58.955615]
```


Write

```

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

    int ret;
    char buf[256];
    pr_info("\n ### i2c Start Reg Address MPU9250: 0x%02X ###", (char)(*offset));
    pr_info("\n ### i2c Cantidad de bytes a escribir: %0d ###", len);

    /*Cargo el address que viene en el offset*/
    buf[0] = (char)(*offset);

    /*Agarro los datos provenientes del usuario*/
    ret = copy_from_user(message, buffer, len);

    size_of_message = strlen(message);

    pr_info("\n ### i2c Bytes recibidos desde el usuario: %d ###", size_of_message);
    /*Copio el mensaje proveniente del usuario en el mensaje a enviar al usuario*/

    /*Recordar que el address que hay que poner al principio hay que ponerlo y por eso va el +1*/
    strcpy(buf+1, message);

    /*Escribo por el I2C con el address al principio*/
    ret = i2c_master_send(modClient, buf, size_of_message);
    pr_info("\n### i2c Se escribieron %d bytes a través del I2C ###", ret);

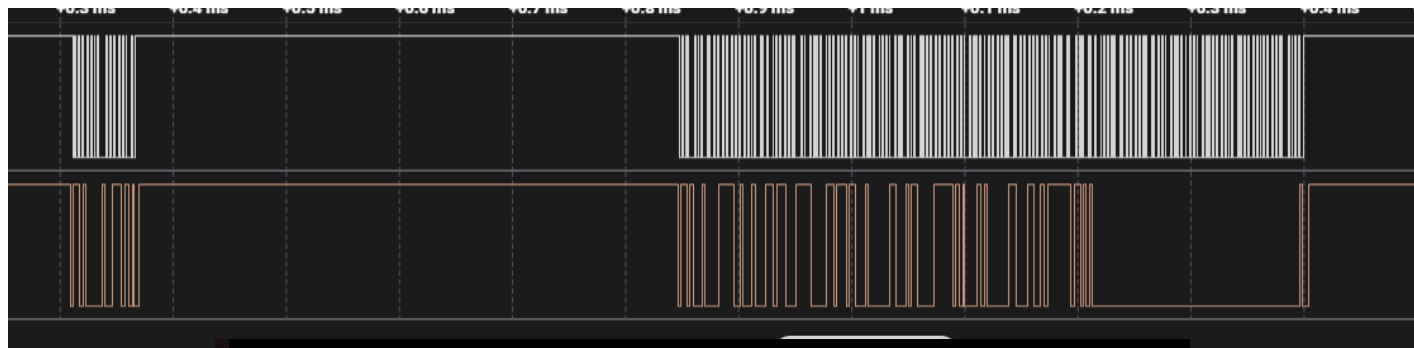
    return len;
}

static int dev_release(struct inode *inodep, struct file *file){
    int i=0;
    int rv;
    char buf[1] = {0x6B};
    char mpu9250_output_buffer[21];
    printk(KERN_INFO "EBBChar: Device successfully closed\n");

    /*Lectura de 22 registros ,dispositivo sigue respondiendo*/
    buf[0] = 0x3B;
    pr_info("READ 21 REGISTER FROM ADDRES 0x3B\n");
    rv = i2c_master_send(modClient, buf, 1);
    rv = i2c_master_recv(modClient, mpu9250_output_buffer, 21);
    pr_info("Datos Recibido: %0d\n", rv);
    for (i = 0; i < 21; i++)
    {

```

Write



```
*** ESCRI[ 58.955615]
[ 58.955615] ### i2c Start Reg Address MPU9250: 0x6A ###
TURA I2C A I2C_USER_CTRL 0x6A ***
[ 58.967128]
[ 58.967128] ### i2c Cantidad de bytes a escribir: 1 ###
[ 58.979542]
[ 58.979542] ### i2c Bytes recibidos desde el usuario:2 ###
*** ESCRITOS 1 BYTES AL REGISTRO: 0x6A ***[ 58.989959]
[ 58.989959] ### i2c Se escribieron 2 bytes a través del I2C ###
```

```
*** ESCRITURA I2C A I2C_PWR_MGMT_1 0x6B ***
[ 58.997106]
[ 58.997106] ### 6lseek en ejecucion mympu9250 ###
[ 59.008855]
[ 59.008855] ### i2c Start Reg Address MPU9250: 0x6B ###
[ 59.019204]
[ 59.019204] ### i2c Cantidad de bytes a escribir: 1 ###
[ 59.026025]
[ 59.026025] ### i2c Bytes recibidos desde el usuario:2 ###
*** ESCRITOS 1 BYTES AL REGISTRO: 0x6B ***[ 59.033492]
[ 59.033492] ### i2c Se escribieron 2 bytes a través del I2C ###
```

lseek

```
static loff_t device_lseek(struct file *file, loff_t offset, int orig) {
    loff_t new_pos = 0;

    printk("\n### "KERN_INFO "lseek en ejecucion mympu9250 ###");
    /*No importa la configuracion del offset siempre se configura con el valor del parametro */

    /*Esto es porque no tenemos ningun buffer y la idea es accederlo como un mapa de registros */
    switch(orig) {
        case 0 : /*seek set*/
            new_pos = offset;
            break;
        case 1 : /*seek cur*/
            new_pos = offset;
            break;
        case 2 : /*seek end*/
            new_pos = offset;
            break;
    }
    file->f_pos = new_pos;

    return new_pos;
}
```

Cliente i2c

```
/******LECTURA I2C*****/
printf("\n\n\n*** LECTURA I2C DE I2C_REG_TEMP 0x41 ***");

lseek(fd,I2C_REG_TEMP,SEEK_SET);

cant_lectura = 2;

printf("\n*** SE LEERAN %d bytes***",cant_lectura);

ret = read(fd,buffer,cant_lectura);

if(ret < 0){
    printf("\n*** FALLO LEER EL MENSAJE DESDE EL DISPOSITIVO ***");
    close(fd);
    return -1;
}

printf("\n*** REGISTRO=0x%02X --> Valor: 0x%02X ***",I2C_REG_TEMP,buffer[0]);

printf("\n*** REGISTRO=0x%02X --> Valor: 0x%02X ***",I2C_REG_TEMP+1,buffer[1]);

temp = (float)(buffer[0]<<8);

temp = temp + (float)buffer[1];

temp = (((float)temp - OFFSET_TEMP)/ESCALA_TEMP) + OFFSET_TEMP;

printf("\n*** TEMPERATURA CUENTAS CRUDAS:%2.2f ***",buffer[0]<<8);

printf("\n*** TEMPERATURA GRADOS :%2.2f ***",temp);
```

FIN