230418_summary

0️⃣ todo

건강한싸피생활 o
꼬맨틀 o
지뢰찾기 o
PPT구성 세모
PPT디자인 세모
Flex센서공부 o
교보재 현황 여쭤보기 x
Gsat모고2
한의원
시스템아키텍처 작성방법
폴더정리

1️⃣ Embedded Class (feat. hy)

MCP3008 개념

- 10비트 해상도의 아날로그-디지털 변환기(ADC) 칩
- SPI(Serial Peripheral Interface) 프로토콜을 사용하여 마이크로컨트롤러와 통신함
- 8개의 아날로그 입력 채널을 제공하며, 이 채널은 0V에서 5V까지의 전압을 측정할 수 있음
- 내부적으로 참조 전압을 생성하므로 외부 참조 전압이 필요없음

MCP3008 활용

- 아날로그 센서 데이터를 디지털로 변환하여 마이크로컨트롤러에서 처리
- 온도, 조도, 압력 등의 센서로 데이터를 수집하고 인터넷에 연결된 컴퓨터나 서버로 전송하는 IoT(Internet of Things) 시스템에서도 사용됨

라이브러리 설치

- RPI에 MCP3008 연결해서 C언어로 개발해야할 때 필요한 라이브러리 설치

```
sudo apt-get update
sudo apt-get install -y git-core
sudo apt-get install -y build-essential
sudo apt-get install -y git
sudo apt-get install -y cmake
sudo apt-get install -y libusb-1.0-0-dev
sudo apt-get install -y libgtk-3-dev
sudo apt-get install -y libglfw3-dev
sudo apt-get install -y libgles2-mesa-dev
sudo apt-get install -y libdrm-dev
sudo apt-get install -y libdrm2
sudo apt-get install -y libegl1-mesa-dev
sudo apt-get install -y libegl1-mesa
sudo apt-get install -y libegl1-mesa-drivers
sudo apt-get install -y libgles2-mesa
sudo apt-get install -y libudev-dev
sudo apt-get install -y libvl53l0x-dev
sudo apt-get install -y libmpsse-dev
sudo apt-get install -y libftdi-dev
sudo apt-get install -y libhidapi-dev
sudo apt-get install -y libwiringpi-dev
sudo apt-get install -y libpigpio-dev
sudo apt-get install -y libcap-dev
sudo apt-get install -y libncurses-dev
sudo apt-get install -y libncurses5-dev
sudo apt-get install -y libncursesw5-dev
sudo apt-get install -y libncurses6-dev
sudo apt-get install -y libncursesw6-dev
sudo apt-get install -y libncursesw6
sudo apt-get install -y libncurses5
sudo apt-get install -y libncurses6
sudo apt-get install -y libncursesw6
```

```
sudo apt-get install -y libncurses5-dev
sudo apt-get install -y libncursesw5-dev
sudo apt-get install -y libncurses6-dev
sudo apt-get install -y libncursesw6-dev
sudo apt-get install -y libncursesw6
sudo apt-get install -y libncurses5
sudo apt-get install -y libncurses6
sudo apt-get install -y libncursesw6
sudo apt-get install -y libncurses5-dev
sudo apt-get install -y libncursesw5-dev
sudo apt-get install -y libncurses6-dev
sudo apt-get install -y libncursesw6-dev
sudo apt-get install -y libncursesw6
sudo apt-get install -y libncurses5
sudo apt-get install -y libncurses6
sudo apt-get install -y libncursesw6
sudo apt-get install -y libncurses5-dev
sudo apt
```

실행순서

```
$ cd G547/Project
$ sudo rmmod main.ko
$ sudo make clean
$ ls
Makefile  README.md  main.c  mpu6050.h  userspace.c
```

```
$ cd G547/Project
$ ls
Makefile  README.md  main.c  mpu6050.h  userspace.c

$ sudo make all
$ ls
Makefile        main.c    main.mod.c  modules.order  userspace.c
Module.symvers  main.ko   main.mod.o  mpu6050.h
README.md       main.mod  main.o      userspace

$ sudo insmod main.ko
$ sudo cat /dev/MPU6050
▒▒▒▒('▒8`▒▒ ▒v`▒▒@▒▒▒accel_readings:x:-12668,y:-6288,z:4616,Gyro_readings:x:-1,y:-18,z:-33,
▒▒▒▒('▒8`▒▒ ▒v`▒▒@▒▒▒accel_readings:x:-12628,y:-6412,z:4804,Gyro_readings:x:-1,y:-18,z:-33,
▒▒▒▒('▒8`▒▒ ▒v`▒▒@▒▒▒accel_readings:x:-12668,y:-6328,z:4580,Gyro_readings:x:-6,y:-18,z:-32,
▒▒▒▒('▒8`▒▒ ▒v`▒▒@▒▒▒accel_readings:x:-12664,y:-6368,z:4772,Gyro_readings:x:-3,y:-18,z:-33,
▒▒▒▒('▒8`▒▒ ▒v`▒▒@▒▒▒accel_readings:x:-12708,y:-6256,z:4608,Gyro_readings:x:-2,y:-17,z:-30,
d ▒▒▒$▒▒@0▒▒@▒▒▒accel_readings:x:-12668,y:-6396,z:4784,Gyro_readings:x:-1,y:-18,z:-28,
d ▒▒▒$▒▒@0▒▒@▒▒▒accel_readings:x:-12724,y:-6256,z:4852,Gyro_readings:x:-1,y:-19,z:-28,
d ▒▒▒$▒▒@0▒▒@▒▒▒accel_readings:x:-12668,y:-6612,z:4704,Gyro_readings:x:-2,y:-17,z:-28,
▒▒▒▒('▒8`▒▒ ▒v`▒▒@▒▒▒accel_readings:x:-12756,y:-6340,z:4628,Gyro_readings:x:-3,y:-18,z:-28,
▒▒▒▒('▒8`▒▒ ▒v`▒▒@▒▒▒accel_readings:x:-12692,y:-6368,z:4916,Gyro_readings:x:-3,y:-17,z:-30,
d ▒▒▒$▒▒@0▒▒@▒▒▒accel_readings:x:-12672,y:-6348,z:5000,Gyro_readings:x:-1,y:-17,z:-35,
d ▒▒▒$▒▒@0▒▒@▒▒▒accel_readings:x:-12692,y:-6252,z:4680,Gyro_readings:x:-3,y:-16,z:-32,
d ▒▒▒$▒▒@0▒▒@▒▒▒accel_readings:x:-12664,y:-6268,z:4660,Gyro_readings:x:-4,y:-18,z:-32,
d ▒▒▒$▒▒@0▒▒@▒▒▒accel_readings:x:-12628,y:-6320,z:4756,Gyro_readings:x:-3,y:-16,z:-33,
d ▒▒▒$▒▒@0▒▒@▒▒▒accel_readings:x:-12632,y:-6352,z:4832,Gyro_readings:x:-3,y:-16,z:-28,
d ▒▒▒$▒▒@0▒▒@▒▒▒accel_readings:x:-12684,y:-6344,z:4800,Gyro_readings:x:-4,y:-18,z:-27,
d ▒▒▒$▒▒@0▒▒@▒▒▒accel_readings:x:-12712,y:-6368,z:4764,Gyro_readings:x:2,y:-17,z:-29,
▒d ▒▒▒$▒▒@0▒▒@▒▒▒accel_readings:x:-12636,y:-6332,z:4868,Gyro_readings:x:1,y:-20,z:-27,
...
```

Makefile (raw)

```
OUTPUT = userspace

obj-m := main.o

all:
  make -C /lib/modules/$(shell uname -r)/build M=$(shell pwd) modules
  gcc -o $(OUTPUT) $(OUTPUT).c

clean:
  make -C /lib/modules/$(shell uname -r)/build M=$(shell pwd) clean
  rm userspace
```

main.c (raw)

```
#include <linux/module.h>
#include <linux/init.h>
#include <linux/slab.h>
#include <linux/i2c.h>
#include <linux/fs.h>
#include <linux/kernel.h>
#include <linux/version.h>
#include <linux/cdev.h>
#include <linux/uaccess.h>


#include "mpu6050.h"

#define DRIVER_NAME "MPU6050"
#define DRIVER_CLASS "MPU6050Class"

static struct i2c_adapter * MPU6050_i2c_adapter = NULL;
static struct i2c_client * MPU6050_i2c_client = NULL;

static struct gyroData gyro;
static struct accelData accel;

#define I2C_BUS_AVAILABLE       1              /* I2C Bus SDA at GPIO2 SCL at GPIO3 available on the raspberry */
#define SLAVE_DEVICE_NAME       "MPU6050"       /* Device and Driver Name */
#define MPU6050_SLAVE_ADDRESS    0x68           /* I2C address of MPU6050 */

static const struct i2c_device_id MPU6050_id[]={
{ SLAVE_DEVICE_NAME, 0},
{ }
};

static struct i2c_driver MPU6050_driver = {
 .driver = {
   .name = SLAVE_DEVICE_NAME,
   .owner = THIS_MODULE
  }
};

static struct i2c_board_info MPU6050_i2c_board_info = {
 I2C_BOARD_INFO(SLAVE_DEVICE_NAME, MPU6050_SLAVE_ADDRESS)
};                                  //Platform device


static dev_t dev_no;        //variable for device number
static struct cdev c_dev;     //variable for the character device structure
static struct class *dev_class;  //varible for the device class



static int open_driver(struct inode *i, struct file *f)   //open function of file
{
 printk("Device : Open() \n");
 return 0;
}

static int close_driver(struct inode *i, struct file *f)  //close function of file
{
        printk("Device : Close()\n");
        return 0;
}

static uint32_t read_accelerometer(void)                                //reading accelerometer values from MPU_6050 sensor
{

    uint8_t lsb,msb,lsb1,msb1,lsb2,msb2;
    uint16_t x_accel,y_accel,z_accel;

    lsb = i2c_smbus_read_byte_data(MPU6050_i2c_client, 0x3C); //reading ACCEL_XOUT_LSB value of MPU6050 from reg60 located at 0x3C
    msb = i2c_smbus_read_byte_data(MPU6050_i2c_client, 0x3B); //reading ACCEL_XOUT_LSB value of MPU6050 from reg59 located at 0x3B
    x_accel = ((uint16_t)msb<<8) | ((uint16_t)lsb);           // concatinating the 8-bit values to form 16-bit value
    accel.x= x_accel;

    lsb1 = i2c_smbus_read_byte_data(MPU6050_i2c_client, 0x3E); //reading ACCEL_YOUT_LSB value of MPU6050 from reg62 located at 0x3E
    msb1 = i2c_smbus_read_byte_data(MPU6050_i2c_client, 0x3D); //reading ACCEL_YOUT_MSB value of MPU6050 from reg61 located at 0x3D
    y_accel = ((uint16_t)msb1<<8) | ((uint16_t)lsb1);          // concatinating the 8-bit values to form 16-bit value
    accel.y=  y_accel;

    lsb2 = i2c_smbus_read_byte_data(MPU6050_i2c_client, 0x40);  //reading ACCEL_ZOUT_LSB value of MPU6050 from reg64 located at 0x40
    msb2 = i2c_smbus_read_byte_data(MPU6050_i2c_client, 0x3F);  //reading ACCEL_ZOUT_MSB value of MPU6050 from reg63 located at 0x3F
    z_accel = ((uint16_t)msb2<<8) | ((uint16_t)lsb2);          // concatinating the 8-bit values to form 16-bit value
    accel.z=  z_accel;
    return 0;

}
static uint32_t read_gyroscope(void)                          //reading gyroscope values from MPU_6050 sensor
```

```
{
    uint8_t lsb,msb,lsb1,msb1,lsb2,msb2;
    uint16_t gyro_x,gyro_y,gyro_z;

    lsb = i2c_smbus_read_byte_data(MPU6050_i2c_client, 0x44);  //reading GYRO_XOUT_LSB value of MPU6050 from reg68 located at 0x44
    msb = i2c_smbus_read_byte_data(MPU6050_i2c_client, 0x43);  //reading GYRO_XOUT_MSB value of MPU6050 from reg67 located at 0x43
    gyro_x = ((uint16_t)msb<<8) | ((uint16_t)lsb);             // concatinating the 8-bit values to form 16-bit value
    gyro.x=  gyro_x;

    lsb1 = i2c_smbus_read_byte_data(MPU6050_i2c_client, 0x46); //reading GYRO_YOUT_LSB value of MPU6050 from reg70 located at 0x46
    msb1 = i2c_smbus_read_byte_data(MPU6050_i2c_client, 0x45); //reading GYRO_YOUT_MSB value of MPU6050 from reg69 located at 0x45
    gyro_y = ((uint16_t)msb1<<8) | ((uint16_t)lsb1);           // concatinating the 8-bit values to form 16-bit value
    gyro.y=  gyro_y;

    lsb2 = i2c_smbus_read_byte_data(MPU6050_i2c_client, 0x48); //reading GYRO_YOUT_LSB value of MPU6050 from reg72 located at 0x48
    msb2 = i2c_smbus_read_byte_data(MPU6050_i2c_client, 0x47); //reading GYRO_YOUT_MSB value of MPU6050 from reg71 located at 0x47
    gyro_z= ((uint16_t)msb2<<8) | ((uint16_t)lsb2);            // concatinating the 8-bit values to form 16-bit value
    gyro.z=  gyro_z;
    return 0;
}

static ssize_t read_driver(struct file *File, char __user *user_buffer, size_t count, loff_t *offs)  //read function of file
{
 int to_copy, not_copied, delta;
 char out_string[100];                // declaring no of characters to get printed
 int16_t accel_x,accel_y,accel_z;
 int16_t x_gyro,y_gyro,z_gyro;
 int32_t k,k1;

 to_copy = min(sizeof(out_string), count);

 k=read_accelerometer();             // Reading accelerometer values
 accel_x = accel.x;
 accel_y = accel.y;
 accel_z = accel.z;

 k1=read_gyroscope();              // Reading gyroscope values
 x_gyro = gyro.x;
 y_gyro = gyro.y;
 z_gyro = gyro.z;

 snprintf(out_string, sizeof(out_string), "accel_readings:x:%d,y:%d,z:%d,Gyro_readings:x:%d,y:%d,z:%d,\n",accel_x,accel_y,accel_z,x_gy
 not_copied = copy_to_user(user_buffer, out_string, to_copy);

 delta = to_copy - not_copied;

 return delta;
}

///////////////// IOCTL FUNCTION ////////////////////////////////////
long ioctl(struct file *file, unsigned int ioctl_num, unsigned long ioctl_param)      // number and param for ioctl
{

 switch(ioctl_num)
 {
  case IOCTL_GYRO:
  read_gyroscope();
  copy_to_user((struct gyroData *)ioctl_param,&gyro,sizeof(struct gyroData));   //passing gyro data to user space
  break;

  case IOCTL_ACCEL:
  read_accelerometer();
  copy_to_user((struct accelData *)ioctl_param,&accel,sizeof(struct accelData)); //passing accel data to user space
  break;

 }
return 0;
}



static struct file_operations fops  = {
 .owner = THIS_MODULE,
 .open = open_driver,
 .release = close_driver,
 .unlocked_ioctl = ioctl_dev,
 .read = read_driver,
};

static int __init mydriver_init(void)
{
 int ret =-1;
 u8 check;
 printk("Driver for MPU6050 sensor registered\n");
 ////////////////////reserve <major,minor>/////////////////////////
 if ( alloc_chrdev_region(&dev_no, 0, 1, DRIVER_NAME) < 0) {
               printk("Failed to assign Device Number!\n");
```

```
        }
        printk("Driver with device number %d for MPU6050 sensor registered\n", dev_no);

   ////////////////dynamically create device node in /dev directory     ///////////////////////
        if ((dev_class = class_create(THIS_MODULE, DRIVER_CLASS)) == NULL)
    {
                printk("Failed to create Device Class!\n");
        unregister_chrdev(dev_no, DRIVER_NAME);  //unregistering the character device with major and minor number
                return (-1);
        }

 ////////////// creating device node ////////////////////////
        if (device_create(dev_class, NULL, dev_no, NULL, DRIVER_NAME) == NULL)
    {
                printk("Failed to create device file!\n");
        class_destroy(dev_class);                   //destroying the device class
        unregister_chrdev(dev_no, DRIVER_NAME);      //unregistering the character device with major and minor number
                return (-1);
        }

////////////// Link file_operations and Cdev to device node//////////////////
        cdev_init(&c_dev, &fops);

        /* register device to kernel */
        if (cdev_add(&c_dev, dev_no, 1) == -1)
    {
                printk("Failed to register device to kernel!\n");
        device_destroy(dev_class, dev_no);              //destroy device node
        class_destroy(dev_class);                       //destroying the device class
        unregister_chrdev(dev_no, DRIVER_NAME);         //unregistering the character device with major and minor number
                return (-1);
        }

        MPU6050_i2c_adapter = i2c_get_adapter(I2C_BUS_AVAILABLE);

        if(MPU6050_i2c_adapter != NULL) {
                MPU6050_i2c_client = i2c_new_client_device(MPU6050_i2c_adapter, &MPU6050_i2c_board_info);
                if(MPU6050_i2c_client != NULL) {
                        if(i2c_add_driver(&MPU6050_driver) != -1) {
                                ret = 0;
                        }
                        else
                                printk("Can't add driver...\n");
                }
                i2c_put_adapter(MPU6050_i2c_adapter);
        }

        printk("MPU6050 Driver Init\n");

  check = i2c_smbus_read_byte_data(MPU6050_i2c_client, 0x75);     // Reading value from WHO AM I stored at reg117 located at 0x75 whic
    printk("Checking whether communication is established or not: 0x%x\n",check);

     i2c_smbus_write_byte_data(MPU6050_i2c_client, 0x6B, 0x01);  //Configuring power management-1 located at 0x6B
     i2c_smbus_write_byte_data(MPU6050_i2c_client, 0x1B, 0x18);  //Configuring Gyroscope at reg27 located at 0x1B with full range of +
     i2c_smbus_write_byte_data(MPU6050_i2c_client, 0x1C, 0x00);  //Configuring accelerometer at reg28 located at 0x1C with full range

        return ret;

}

static void __exit mydriver_exit(void) {
        i2c_unregister_device(MPU6050_i2c_client);   // unregistering the i2c client device
        i2c_del_driver(&MPU6050_driver);          // unregister I2C driver
        cdev_del(&c_dev);                      //deleting the link between cdev and file operations
        device_destroy(dev_class, dev_no);     //destroy device node
        class_destroy(dev_class);             //destroying the device class
        unregister_chrdev_region(dev_no, 1);  //unregistering the character device with major and minor number
    printk("Bye:Driver for MPU6050 sensor unregistered!\n");
}

module_init(mydriver_init);
module_exit(mydriver_exit);
MODULE_AUTHOR("SAINADH");
MODULE_LICENSE("GPL");
MODULE_DESCRIPTION("MPU6050 Sensor Driver");
```

userspace.c (raw)

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/ioctl.h>
#include<time.h>
#include<fcntl.h>
```

```
#include<signal.h>
#include<unistd.h>


#include "mpu6050.h"

int file_desc;


/* Functions for the ioctl calls */
int ioctl_accelerometer(int file_desc, struct accelData *msg)    //Function that reads the Values from Accelerometer
{
 int ret_val;

 ret_val = ioctl(file_desc, IOCTL_ACCEL,msg);    //ioctl function call with request code  IOCTL_ACCEL
 return ret_val;
}

int ioctl_gyroscope(int file_desc, struct gyroData *msg)    //Function that reads the Values from Gyroscope
{
 int ret_val;

 ret_val = ioctl(file_desc, IOCTL_GYRO,msg);     //ioctl function call with request code  IOCTL_GYRO
 return 0;
}


/* Main - Call the ioctl functions */
int main(void)
{
 int ret_val,i;
 struct gyroData gyro_data;
 struct accelData accel_data;
 float xaccel,yaccel,zaccel,xgyro,ygyro,zgyro;

 file_desc = open(DEVICE_FILE_NAME,0); // opening the device node and returning the value to file_desc

 if(file_desc<0)
 {
  printf(" Failed to open device %s\n",DEVICE_FILE_NAME);  //Display if permission is denied to open /dev/MPU6050
  exit(-1);
 }

 while(1)
 {
  ioctl_accelerometer(file_desc,&accel_data);   //Calling the accelerometer ioctl function call
  xaccel= ((float)(accel_data.x))/16384;       // Calibrating the obtained value as per register map datasheet
  yaccel= ((float)(accel_data.y))/16384;       // Calibrating the obtained value as per register map datasheet
  zaccel= ((float)(accel_data.z))/16384;       // Calibrating the obtained value as per register map datasheet
  printf("Raw accelerometer readings: x:%d , y:%d, z:%d\n",accel_data.x,accel_data.y,accel_data.z);
  printf("After calibrating accelerometer readings: x:%f , y:%f, z:%f\n",xaccel,yaccel,zaccel);
  ioctl_gyroscope(file_desc,&gyro_data);      //Calling the gyroscope ioctl function call
  xgyro= ((float)(gyro_data.x))/16.4;        // Calibrating the obtained value as per register map datasheet
  ygyro= ((float)(gyro_data.y))/16.4;        // Calibrating the obtained value as per register map datasheet
  zgyro= ((float)(gyro_data.z))/16.4;        // Calibrating the obtained value as per register map datasheet
  printf("Raw gyroscope readings:x:%d , y:%d, z:%d\n",gyro_data.x,gyro_data.y,gyro_data.z);
  printf("After calibrating gyroscope readings:x:%f , y:%f, z:%f\n",xgyro,ygyro,zgyro);
  sleep(5);
 }
}
```

mpu6050.h

```
#ifndef CHAR_CONFIG_H
#define CHAR_CONFIG_H

#include <linux/ioctl.h>

#define MAGIC_NUM 225


struct gyroData
{
 int16_t x;
 int16_t y;
 int16_t z;
};

struct accelData
{
 int16_t x;
 int16_t y;
 int16_t z;
```

```
};

//IOCTL interface prototypes

#define IOCTL_GYRO  _IOWR(MAGIC_NUM, 0, struct gyroData*)
#define IOCTL_ACCEL  _IOWR(MAGIC_NUM, 1, struct accelData*)



//Device file interface
#define DEVICE_FILE_NAME "/dev/mpu6050"

#endif
```

2️⃣ trouble

문제상황

- 커널에 디바이스 드라이버를 올려
- 디바이스 파일이 생성됨 (/dev/MPU6050)
- 그 데이터를 userspace 에서 사용하고 싶어 (main.c)
- 근데 이상한 값이 가져와짐 !
- 이론 : 첫 값을 변수로 저장해버리고 그거를 출력하는걸지도.

3️⃣ PPT 디자인

CONTENTS