

1. 05.motor

2024/7/18 Table of Contents

05.motor

目的

構成データ

モータo制御

デバイスドライバ

例題 asevent

課題1 asled

1.1. 目的

組込みアプリケーション開発 05.motor

1.2. 構成データ

1.2.1. /media/sf_ArmadilloX1/hwpwm/work/R06_2024/Apllication_debug/text/practiceディレクトリ

▼ .../share/ArmadilloX1/hwpwm/work/R06_2024/Application_debug/text/practice/ の構成

```
1 user@1204PC-Z490M:/mnt/v/VirtualBoxWork/share/ArmadilloX1/hwpwm/work/R06_2024/Apllication_debug/text/practice$ tr
2 ./
3 | 05.motor/
4 | | drivers/
5 | | | buttons/
6 | | | leds/
7 | | | motor/
8 | | | | Makefile* <----- ドライバ用Makefile
9 | | | | motor_hwpwm.c* <----- ドライバソース
10 | | | Makefile* <----- デバイス制御用Makefile
11 | | | mtctl.c* <----- 例題 デバイス制御用ソース
12 | | | mtfan2.c* <----- 課題2 デバイス制御用ソース
13 | | | mtfan.c* <----- 課題1 デバイス制御用ソース
```

1.3. モーター制御

1.3.1. デバイス仕様

⚠ DCモーターは、ハードウェアPMWによる制御（ソフトウェアPWMよりも滑らか）

モーター: RE 140RA 2270（マブチモーター）

- モーターの駆動電圧: 約 2.5V (max 1.5A)
- モーターをフル回転 (MOT_PWM duty100%duty100%) の状態で 起動すると電源、及び周辺回路に悪影響を与える可能性あり
- 起動時は多大な突入電流 2.5A 程度流れる
- モータを起動する時は、PWM 制御によってモータの回転速度を 徐々に上げていく
- 起動時の突入電流を 1A 以下 (700mA 程度) に抑えるには、500mS 程度の時間をかけ、徐々に duty が 100% になるように制御する
- 通常動作時に PWM 制御によって duty を 100% 以下にすると、モータの消費電流が増加
- duty が 70% 程度になると、消費電流は duty100% の時の倍近くにまで増加
- 長時間モータを連続駆動すると、モータ用電源IC U7 BA00DD0WHFP と ドライバー IC U6 :TB65 52 FNG が発熱して高温状態になる
- 連続駆動する時は、duty100% で使用することを推奨します。

1.4. デバイスドライバ

1.4.1. ソース

motor_hwpwm.c

▼ 05.motor/drivers/motor_hwpwm.c

```

#include <linux/init.h>
#include <linux/module.h>
#include <linux/platform_device.h>
#include <linux/fs.h>
#include <linux/pwm.h>

#include <asm/armadilloX1-ext-cpld.h>

// モータ動作モードのマクロ
#define MOT_PWM          0x01
#define MOT_CW           0x02
#define MOT_CCW          0x04
#define MOT_STBY         0x08
#define MODE_BREAK       (MOT_CW|MOT_CCW|MOT_STBY)
#define MODE_CCW         (MOT_CCW|MOT_PWM|MOT_STBY)
#define MODE_CCW_BR      (MOT_CCW|MOT_STBY)
#define MODE_CW          (MOT_CW|MOT_PWM|MOT_STBY)
#define MODE_CW_BR       (MOT_CW|MOT_STBY)
#define MODE_STOP        (MOT_STBY)
#define MODE_STBY        (0)

#define MAX_ROTATION     100
#define MIN_ROTATION     -100

#define PWM_PERIOD       500000 // PWMの1周期の時間(単位: ns)

// 設定された値を保持する変数
static int set_duty = 0;

// PWMデバイス
static struct pwm_device *pwm;

// 属性ファイルから現在設定されている値を読み出す関数(motor_rotation_show)
static ssize_t motor_rotation_show(struct device *dev, struct device_attribute *attr, char *buf)
{
    return sprintf(buf, "%d\n", set_duty);
}

// 属性ファイルへの書き込み関数
static ssize_t motor_rotation_store(struct device *dev, struct device_attribute *attr, const char *buf, size_t count)
{
    int ret;
    long val;
    unsigned int cycle;
    struct pwm_state state;

    // バッファからデータを取り出し、数値に変換する。
    ret = kstrtol(buf, 10, &val);
    if (ret)
        return ret;

    // 0以外が指定された場合の処理
    if (val != 0) {
        // 現在のPWMデバイスの設定を取得する。
        pwm_get_state(pwm, &state);

        // PWMを設定変更前に停止状態にする。
        state.enabled = 0;

        // PWMデバイスに設定を適用する。
        ret = pwm_apply_state(pwm, &state);
        if (ret)
            return ret;
    }
}

```

```

// 正転の場合の処理(正の値を設定したとき)
if (val > 0) {
    if (val > MAX_ROTATION) {
        val = MAX_ROTATION;
    }

    // 指定されたデューティ比にするためのパルス信号のHigh時間を計算する。
    val = ((val / 10) * 10);
    cycle = (PWM_PERIOD * val) / 100;

    // 現在のPWMデバイスの設定を取得する。
    pwm_get_state(pwm, &state);

    // PWMの1周期あたりのHigh時間を設定する。
    state.duty_cycle = cycle;

    // PWMを動作状態にする。
    state.enabled = 1;

    // PWMデバイスに設定を適用する。
    ret = pwm_apply_state(pwm, &state);
    if (ret)
        return ret;

    // モータ動作モードをCWにセットする。
    cpld_write(CPLD_WRITE_MOT, MODE_CW);

// 反転の場合の処理(負の値を設定したとき)
} else {
    if (val < MIN_ROTATION) {
        val = MIN_ROTATION;
    }

    // 指定されたデューティ比にするためのパルス信号のHigh時間を計算する。
    val = ((val / 10) * 10);
    cycle = (PWM_PERIOD * -val) / 100;

    // 現在のPWMデバイスの設定を取得する。
    pwm_get_state(pwm, &state);

    // PWMの1周期あたりのHigh時間を設定する。
    state.duty_cycle = cycle;

    // PWMを動作状態にする。
    state.enabled = 1;

    // PWMデバイスに設定を適用する。
    ret = pwm_apply_state(pwm, &state);
    if (ret)
        return ret;

    // モータ動作モードをCCWにセットする。
    cpld_write(CPLD_WRITE_MOT, MODE_CCW);
}

// 0が指定された場合の処理
} else {
    // モータ動作モードをストップにセットする。
    cpld_write(CPLD_WRITE_MOT, MODE_STOP);

    // 現在のPWMデバイスの設定を取得する。
    pwm_get_state(pwm, &state);

    // PWMを停止状態にする。
    state.enabled = 0;

```

```

        // PWMデバイスに設定を適用する。
        ret = pwm_apply_state(pwm, &state);
        if (ret)
            return ret;
    }

    // 設定された値を保持する。
    set_duty = val;

    return count;
}

// motor_rotationの最大値を返す関数(motor_max_rotation_show)
static ssize_t motor_max_rotation_show(struct device *dev, struct device_attribute *attr, char *buf)
{
    return sprintf(buf, "%d\n", MAX_ROTATION);
}

// motor_rotationの最小値を返す関数(motor_min_rotation_show)
static ssize_t motor_min_rotation_show(struct device *dev, struct device_attribute *attr, char *buf)
{
    return sprintf(buf, "%d\n", MIN_ROTATION);
}

// デバイス属性構造体
static DEVICE_ATTR_RW(motor_rotation);
static DEVICE_ATTR_RO(motor_max_rotation);
static DEVICE_ATTR_RO(motor_min_rotation);

// モータ デバイス属性グループ構造体
static struct attribute *motor_class_attrs[] = {
    &dev_attr_motor_rotation.attr,
    &dev_attr_motor_max_rotation.attr,
    &dev_attr_motor_min_rotation.attr,
    NULL,
};

ATTRIBUTE_GROUPS(motor_class);

// モータクラスデバイス構造体
static struct class motor_class = {
    .owner          = THIS_MODULE,
    .name           = "motor",
    .dev_groups     = motor_class_groups,
};

// probe関数(motor_probe)
static int motor_probe(struct platform_device *pdev)
{
    struct device *dev;
    struct pwm_state state;
    int ret;

    // 属性ファイルを作成
    // /sys/class/motor/配下にファイルを作成し、/sys/class/motor/motor0/motor_rotationに
    // 値を書き込むとデバイス書き込み関数(motor_rotation_store)が呼び出されるように
    // 作成します。
    dev = device_create(&motor_class, NULL, 0, NULL, "motor0");
    if (IS_ERR(dev)){
        dev_err(&pdev->dev, KERN_ERR "failed to create device.\n");
        return PTR_ERR(dev);
    }

    // 使用するPWMデバイスの情報を取得する。
    pwm = pwm_request(1, "Hardware-PWM Motor");
    if (IS_ERR(pwm)) {

```

```

        dev_err(&pdev->dev, KERN_ERR "failed to request PWM device.\n");
        device_destroy(&motor_class, 0);
        return PTR_ERR(pwm);
    }

    // モータ動作モードをストップにセットする。
    cpld_write(CPLD_WRITE_MOT, MODE_STOP);

    // 現在のPWMデバイスの設定を取得する。
    pwm_get_state(pwm, &state);

    // PWMの1周期の時間を設定する。
    state.period = PWM_PERIOD;

    // PWMを停止状態にする。
    state.enabled = 0;

    // PWMデバイスに設定を適用する。
    ret = pwm_apply_state(pwm, &state);
    if (ret) {
        dev_err(&pdev->dev, KERN_ERR "failed to apply state PWM.\n");
        pwm_free(pwm);
        device_destroy(&motor_class, 0);
        return ret;
    }

    return 0;
}

// remove関数(motor_remove)
static int motor_remove(struct platform_device *pdev)
{
    struct pwm_state state;

    // モータ動作モードをスタンバイにセットする。
    cpld_write(CPLD_WRITE_MOT, MODE_STBY);

    // 現在のPWMデバイスの設定を取得する。
    pwm_get_state(pwm, &state);

    // PWMを停止状態にする。
    state.enabled = 0;

    // PWMデバイスに設定を適用する。
    pwm_apply_state(pwm, &state);

    // PWMデバイスを解放する。
    pwm_free(pwm);

    // モータクラスデバイス構造体を解除します。(device_destroy)
    device_destroy(&motor_class, 0);
    return 0;
}

// プラットフォームドライバ構造体の宣言
// ドライバにprobe関数とremove関数、デバイスドライバ構造体を設定します。
static struct platform_driver motor_driver = {
    .probe      = motor_probe,
    .remove     = motor_remove,
    .driver     = {
        .name   = "armadillo-x1-extension-motor",
        .owner  = THIS_MODULE,
    },
};

```

```

260 static struct platform_device *pdev;
261
262 // 初期化関数(motor_init)
263 static int __init motor_init(void)
264 {
265     int ret = 0;
266
267     // モータクラスデバイスを登録します。(class_register)
268     // /sys/class/motor/配下にファイルを作成し、/sys/class/motor/motor0/motor_rotationに
269     // 値を書き込むとデバイス書き込み関数(motor_rotation_store)が呼び出されるように
270     // 登録します。
271     ret = class_register(&motor_class);
272     if (ret)
273         goto err1;
274
275     // プラットフォームドライバを登録します。(platform_driver_register)
276     // 登録したプラットフォームデバイスのリソース情報を取得し、
277     // プラットフォームドライバとして、probe関数とremove関数を登録します。
278     ret = platform_driver_register(&motor_driver);
279     if (ret)
280         goto err2;
281
282     // プラットフォームデバイスを登録します(platform_device_register_simple)
283     // プラットフォーム依存のデバイス情報を登録します。
284     pdev = platform_device_register_simple("armadillo-x1-extension-motor", -1, NULL, 0);
285     if (!pdev)
286         goto err3;
287
288     return 0;
289
290 err3: platform_driver_unregister(&motor_driver);
291 err2: class_unregister(&motor_class);
292 err1: return ret;
293 }
294
295 // 終了関数(motor_exit)
296 static void __exit motor_exit(void)
297 {
298     // プラットフォームデバイスを解除します。(platform_device_unregister)
299     platform_device_unregister(pdev);
300
301     // プラットフォームドライバを解除します。(platform_driver_unregister)
302     platform_driver_unregister(&motor_driver);
303
304     // モータクラスデバイスを解除します。(class_unregister)
305     class_unregister(&motor_class);
306 }
307
308 // 初期化の際に、初期化関数が呼ばれるように登録します。
309 module_init(motor_init);
310
311 // 終了する際に、終了関数が呼ばれるように登録します。
312 module_exit(motor_exit);
313
314 // モジュールについての説明
315 MODULE_DESCRIPTION("motor driver");
316
317 // MODULE_LICENSEは"GPL"とします。
318 MODULE_LICENSE("GPL");

```


Makefile

▼ 05.motor/drivers/motor/Makefile

```
1  KERNELDIR = /home/atmark/linux-4.9-x1-at27_dbg
2  ARCH = arm
3  PREFIX = arm-linux-gnueabi-
4  MOD_PATH = /work/linux/nfsroot
5
6  EXTRA_CFLAGS += -gdwarf-2 -O0
7
8  obj-m := motor_hwpwm.o
9
10 modules:
11     $(MAKE) -C $(KERNELDIR) M=`pwd` ARCH=$(ARCH) CROSS_COMPILE=$(PREFIX) modules
12
13 modules_install:
14     $(MAKE) -C $(KERNELDIR) M=`pwd` ARCH=$(ARCH) INSTALL_MOD_PATH=$(MOD_PATH) modules_install
15
16 myinstall:
17     cp -p *.ko /media/sf_ArmadilloX1/hwpwm/dbg/debug_share_hwpwm/R06_2024/04_practice
18     cp -p *.c /media/sf_ArmadilloX1/hwpwm/dbg/debug_share_hwpwm/R06_2024/04_practice
19
20 clean:
21     $(MAKE) -C $(KERNELDIR) M=`pwd` clean
```


1.4.2. 動作確認

make clean

▼ \$ make clean

```
1  atmark@atde8:/media/sf_ArmadilloX1/hwpwm/work/R06_2024/Application_debug/text/practice-example/05.motor/drivers/n
2  make -C /home/atmark/linux-4.9-x1-at27_dbg M=`pwd` clean
3  make[1]: ディレクトリ '/home/atmark/linux-4.9-x1-at27_dbg' に入ります
4  CLEAN /media/sf_ArmadilloX1/hwpwm/work/R06_2024/Application_debug/text/practice-example/05.motor/drivers/motc
5  CLEAN /media/sf_ArmadilloX1/hwpwm/work/R06_2024/Application_debug/text/practice-example/05.motor/drivers/motc
6  make[1]: ディレクトリ '/home/atmark/linux-4.9-x1-at27_dbg' から出ます
```

make modules

 「make[2]: 警告: ファイル 'media/sf_ArmadilloX1/hwpwm/work/R06_2024/Application_debug/text/practice-example/02.led/drivers/leds/leds.o' の修正時刻 20 は未来の時刻です」と表示された場合は chrony を ATDE8 と ArmadilloX1 にインストールすると解決する

▼ \$ make modules

```
1  atmark@atde8:/media/sf_ArmadilloX1/hwpwm/work/R06_2024/Application_debug/text/practice-example/05.motor/drivers/n
2  make -C /home/atmark/linux-4.9-x1-at27_dbg M=`pwd` ARCH=arm CROSS_COMPILE=arm-linux-gnueabi- modules
3  make[1]: ディレクトリ '/home/atmark/linux-4.9-x1-at27_dbg' に入ります
4  CC [M] /media/sf_ArmadilloX1/hwpwm/work/R06_2024/Application_debug/text/practice-example/05.motor/drivers/motc
5  Building modules, stage 2.
6  MODPOST 1 modules
7  CC /media/sf_ArmadilloX1/hwpwm/work/R06_2024/Application_debug/text/practice-example/05.motor/drivers/motc
8  LD [M] /media/sf_ArmadilloX1/hwpwm/work/R06_2024/Application_debug/text/practice-example/05.motor/drivers/motc
9  make[1]: ディレクトリ '/home/atmark/linux-4.9-x1-at27_dbg' から出ます
```

sudo make modules_install

▼ \$ sudo make modules_install

```

1 | atmark@atde8:/media/sf_ArmadilloX1/hwpwm/work/R06_2024/Application_debug/text/practice-example/05.motor/drivers/n
2 | [sudo] atmark のパスワード:
3 | make -C /home/atmark/linux-4.9-x1-at27_dbg M='pwd' ARCH=arm INSTALL_MOD_PATH=/work/linux/nfsroot modules_install
4 | make[1]: ディレクトリ '/home/atmark/linux-4.9-x1-at27_dbg' に入ります
5 |   INSTALL /media/sf_ArmadilloX1/hwpwm/work/R06_2024/Application_debug/text/practice-example/05.motor/drivers/motc
6 |   DEPMOD  4.9.133-at27
7 | depmod: WARNING: could not open modules.order at /work/linux/nfsroot/lib/modules/4.9.133-at27: No such file or dir
8 | depmod: WARNING: could not open modules.builtin at /work/linux/nfsroot/lib/modules/4.9.133-at27: No such file or c
9 | make[1]: ディレクトリ '/home/atmark/linux-4.9-x1-at27_dbg' から出ます

```

sudo make myinstall

▼ \$ sudo make myinstall

```

1 | atmark@atde8:/media/sf_ArmadilloX1/hwpwm/work/R06_2024/Application_debug/text/practice-example/05.motor/drivers/n
2 | cp -p *.ko /media/sf_ArmadilloX1/hwpwm/dbg/debug_share_hwpwm/R06_2024/04_practice
3 | cp -p *.c /media/sf_ArmadilloX1/hwpwm/dbg/debug_share_hwpwm/R06_2024/04_practice

```

cd

▼ root@armadillo:/# cd /lib/modules/4.9.133-at27/extra/

```

1 | root@armadillo:~# cd /lib/modules/4.9.133-at27/extra/

```

insmod

▼ #insmod leds.ko, #insmod motor.ko, #insmod buttons.ko

```

1 | root@armadillo:/lib/modules/4.9.133-at27/extra# insmod motor_hwpwm.ko
2 | root@armadillo:/lib/modules/4.9.133-at27/extra# insmod leds.ko
3 | root@armadillo:/lib/modules/4.9.133-at27/extra# insmod buttons.ko
4 | root@armadillo:/lib/modules/4.9.133-at27/extra# lsmod
5 | Module                Size  Used by
6 | buttons                3065  0
7 | leds                   2103  0
8 | motor_hwpwm            4415  0

```

1.4.3. デバイスファイル

"/sys/class/motor/motor0/motor_rotation"

デバイスファイルに以下の値を書込むと回転


mode	speed
CW	1 ~ 100
CCW	-1 ~ -100
STOP	0

▼ デバイスファイルによる回転制御

```

1 | root@armadillo:/lib/modules/4.9.133-at27/extra# echo 100 > /sys/class/motor/motor0/motor_rotation
2 | root@armadillo:/lib/modules/4.9.133-at27/extra# echo 0 > /sys/class/motor/motor0/motor_rotation
3 | root@armadillo:/lib/modules/4.9.133-at27/extra#

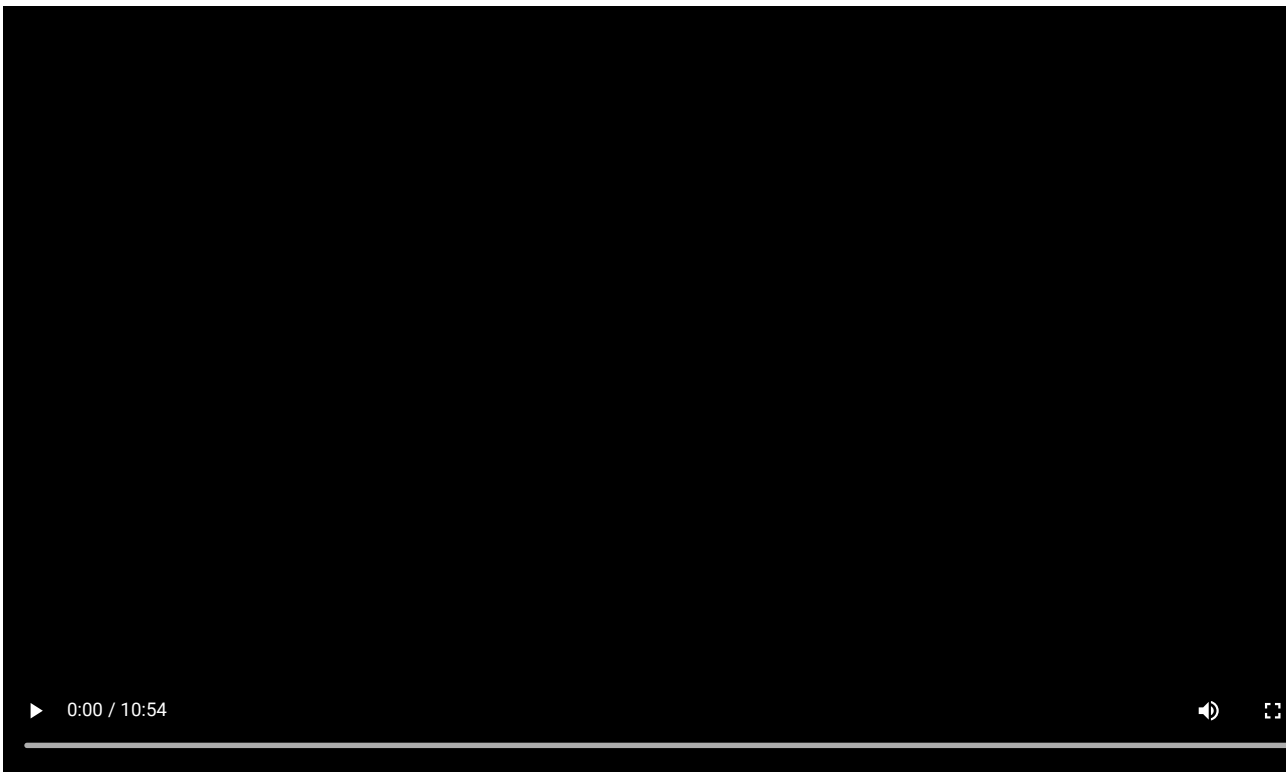
```

 DCモーターが回転しない場合、motor_hwpwm.c のソースを確認すること

1.4.4. 実行している様子

▼ motorデバイスファイルによるモータ制御を実行している動画

https://youtu.be/nsv_obnWsRA



1.5. 例題 mtctl

モーター制御コマンド

- 引数が指定されない時は モーター の回転状態を表示
- 引数を1つ指定すると モーター の回転状態を変更

1.5.1. ソース

mtctl.c

▼ 05.motor/mtctl.c

```

1  #include <stdio.h>
2  #include <fcntl.h>
3  #include <unistd.h>
4  #include <stdlib.h>
5
6  // モータ制御用ファイル
7  #define MOTOR_FILE      "/sys/class/motor/motor0/motor_rotation"
8  #define DATALEN        16
9
10 int main(int argc, char *argv[])
11 {
12     int fd;
13     int rot;
14     char *p;
15     int ret, n;
16     char data[DATALEN];
17
18     // モータ制御用ファイルをオープンします。
19     fd = open(MOTOR_FILE, O_RDWR);
20     // オープンに失敗したら、main関数をエラー終了します。
21     if (fd < 0){
22         perror("open");
23         return 1;
24     }
25
26     switch(argc){
27     case 1:
28         // モータの回転状態を読み出します。
29         ret = read(fd, data, DATALEN);
30         // 読み出しに失敗したら、main関数をエラー終了します。
31         if (ret < 0){
32             perror("read");
33             return 1;
34         }
35         // モータの回転状態を表示します。
36         data[ret] = '\0';
37         printf("rotation: %s", data);
38         break;
39     case 2:
40         // 文字列を数値に変換します。
41         rot = strtol(argv[1], &p, 0);
42         // 変換に失敗したら、main関数をエラー終了します。
43         if (*p != '\0'){
44             fprintf(stderr, "invalid number\n");
45             return 2;
46         }
47
48         // 数値が範囲内でなければ、main関数をエラー終了します。
49         if (rot < -100 || rot > 100){
50             fprintf(stderr, "out of range\n");
51             return 2;
52         }
53
54         // 数値を文字列に変換します。
55         n = sprintf(data, "%d", rot);
56         // モータの状態変化を書き込みます。
57         ret = write(fd, data, n);
58         // 書き込みに失敗したら、main関数をエラー終了します。
59         if (ret < 0){
60             perror("write");
61             return 1;
62         }
63         break;
64     default:

```

```

65         // 引数の数が指定どおりでない場合、main関数をエラー終了します。
66         fprintf(stderr, "Usage: %s [rotation]\n", argv[0]);
67         return 2;
68     }
69
70     // モータ制御用ファイルをクローズします。
71     close(fd);
72     return 0;
73 }

```

Makefile

▼ 05.motor/Makefile

```

1  CC = arm-linux-gnueabi-hf-gcc
2  #TARGET = mtctl mtfan mtfan2
3  TARGET = mtctl
4  CFLAGS = -gdwarf-2 -O0
5
6  all: $(TARGET)
7
8  install :
9      cp -p $(TARGET) /work/linux/nfsroot/debug/04_practice
10     cp -p $(TARGET) /media/sf_ArmadilloX1/hwpwm/dbg/debug_share_hwpwm/R06_2024/04_practice
11     cp -p $(TARGET).c /media/sf_ArmadilloX1/hwpwm/dbg/debug_share_hwpwm/R06_2024/04_practice
12
13  clean:
14      rm -f $(TARGET)
15
16  .PHONY: clean

```

1.5.2. 動作確認

make clean

▼ \$ make clean

```

1  atmark@atde8:/media/sf_ArmadilloX1/hwpwm/work/R06_2024/Application_debug/text/practice-example/05.motor$ make clean
2  rm -f mtctl

```

make

▼ \$ make

```

1  atmark@atde8:/media/sf_ArmadilloX1/hwpwm/work/R06_2024/Application_debug/text/practice-example/05.motor$ make
2  arm-linux-gnueabi-hf-gcc -gdwarf-2 -O0 mtctl.c -o mtctl

```

sudo make install

▼ \$ sudo make install

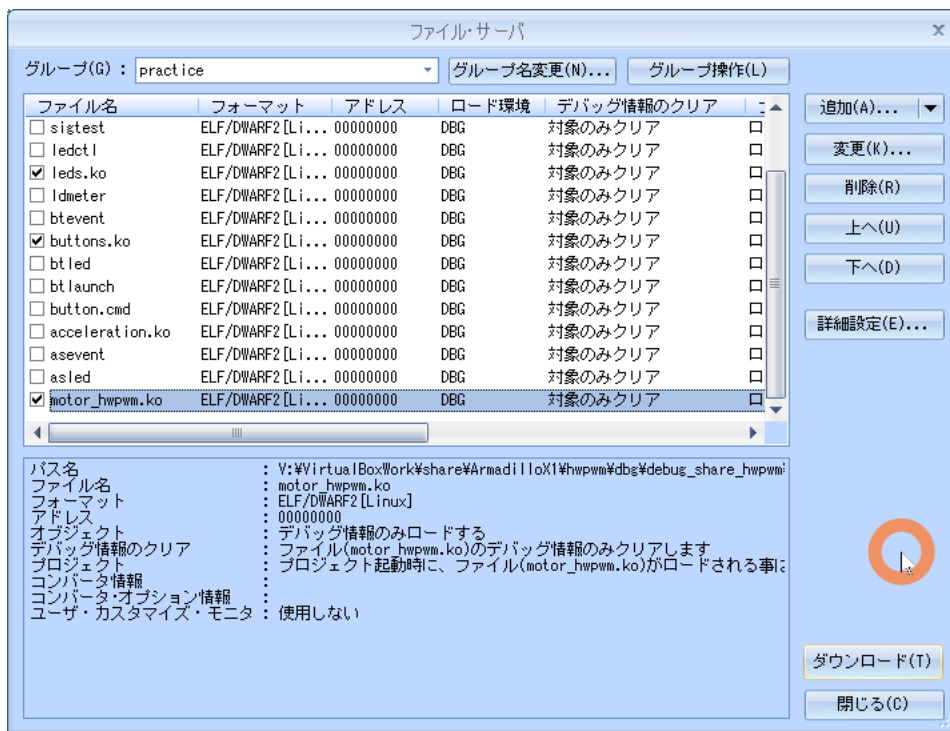
```

1  atmark@atde8:/media/sf_ArmadilloX1/hwpwm/work/R06_2024/Application_debug/text/practice-example/05.motor$ sudo make install
2  [sudo] atmark のパスワード:
3  cp -p mtctl /work/linux/nfsroot/debug/04_practice
4  cp -p mtctl /media/sf_ArmadilloX1/hwpwm/dbg/debug_share_hwpwm/R06_2024/04_practice
5  cp -p mtctl.c /media/sf_ArmadilloX1/hwpwm/dbg/debug_share_hwpwm/R06_2024/04_practice

```

CSIDEでロード

▼ メニュー「ファイル」 - 「ロード」



insmod (既にinsmod済みなら割愛)

⚠ leds.ko も insmod しておくこと

▼ #insmod leds.ko, #insmod motor ko, #insmod buttons.ko

```

1 root@armadillo:/lib/modules/4.9.133-at27/extra# insmod motor_hwpwm.ko
2 root@armadillo:/lib/modules/4.9.133-at27/extra# insmod leds.ko
3 root@armadillo:/lib/modules/4.9.133-at27/extra# insmod buttons.ko
4 root@armadillo:/lib/modules/4.9.133-at27/extra# lsmod
5 Module                Size  Used by
6 buttons                3065  0
7 leds                   2103  0
8 motor_hwpwm            4415  0

```

実行結果

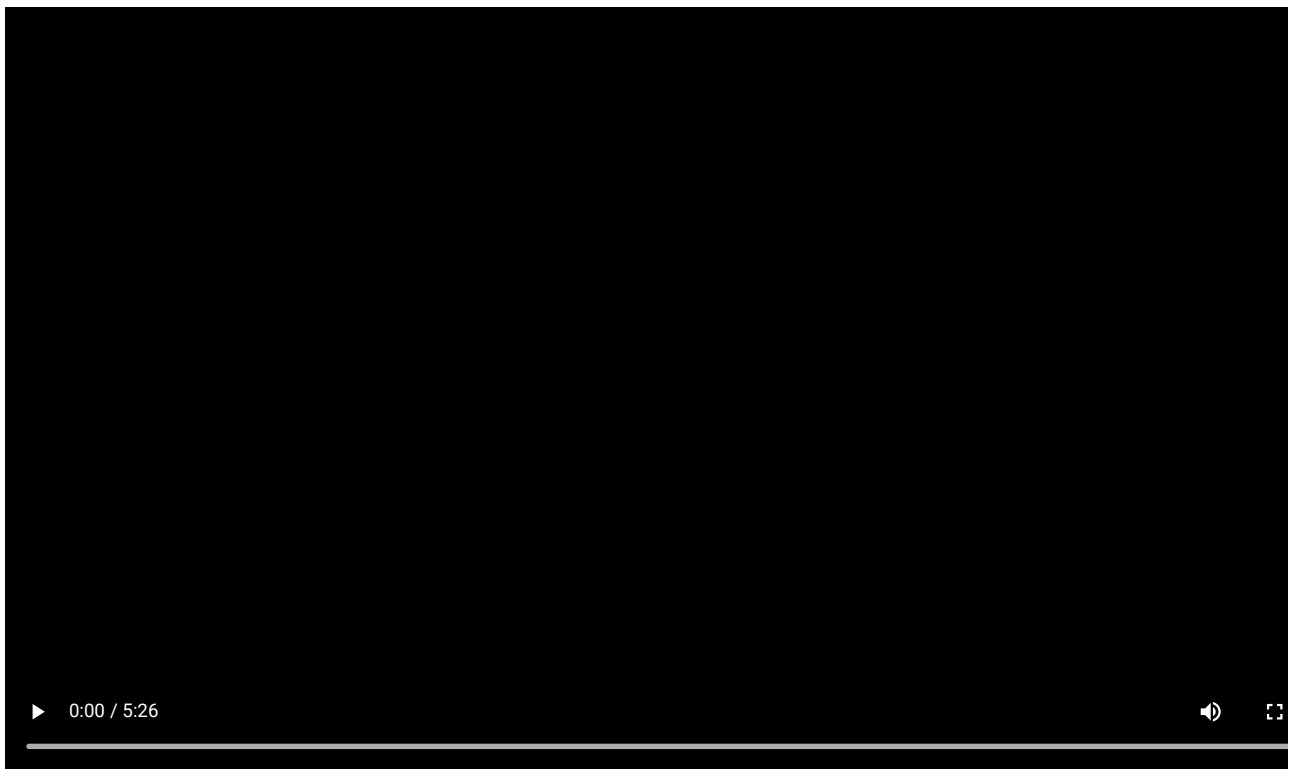
▼ root@armadillo:/debug/04_practice# ./mtctl

```
1 root@armadillo:/debug/04_practice# ./mtctl
2 rotation: 0
3 root@armadillo:/debug/04_practice# ./mtctl -100
4 root@armadillo:/debug/04_practice# ./mtctl -90
5 root@armadillo:/debug/04_practice# ./mtctl -80
6 root@armadillo:/debug/04_practice# ./mtctl -70
7 root@armadillo:/debug/04_practice# ./mtctl -60
8 root@armadillo:/debug/04_practice# ./mtctl -50
9 root@armadillo:/debug/04_practice# ./mtctl -40
10 root@armadillo:/debug/04_practice# ./mtctl -30
11 root@armadillo:/debug/04_practice# ./mtctl -20
12 root@armadillo:/debug/04_practice# ./mtctl -10
13 root@armadillo:/debug/04_practice# ./mtctl 0
14 root@armadillo:/debug/04_practice# ./mtctl 10
15 root@armadillo:/debug/04_practice# ./mtctl 20
16 root@armadillo:/debug/04_practice# ./mtctl 30
17 root@armadillo:/debug/04_practice# ./mtctl 40
18 root@armadillo:/debug/04_practice# ./mtctl 50
19 root@armadillo:/debug/04_practice# ./mtctl 60
20 root@armadillo:/debug/04_practice# ./mtctl 70
21 root@armadillo:/debug/04_practice# ./mtctl 80
22 root@armadillo:/debug/04_practice# ./mtctl 90
23 root@armadillo:/debug/04_practice# ./mtctl 100
24 root@armadillo:/debug/04_practice# ./mtctl
25 rotation: 100
26 root@armadillo:/debug/04_practice# ./mtctl 0
```

実行している様子

▼ mtctl を実行している動画

https://youtu.be/Z1M_sQEWNLy



1.6. 課題1 mtfan

- SW3 モーターの強さを100にして回転
- SW2 モーターの強さを50にして回転
- SW1 モーター停止

bs	mode	speed
SW1	stop	0
SW2	弱	50
SW3	強	100

1.6.1. ソース

mtfan.c

▼ 05.motor/mtfan.c


```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <fcntl.h>
4  #include <unistd.h>
5  #include <linux/input.h>
6
7  // ボタン制御用ファイル
8  #define BUTTON_FILE      "/dev/input/event3"
9  // モータ制御用ファイル
10 #define MOTOR_FILE       "/sys/class/motor/motor0/motor_rotation"
11 #define DATALEN         16
12
13 int fd_mt;
14
15 void change_motor(int rotation)
16 {
17
18
19
20
21
22
23
24
25
26
27
28
29 }
30
31 int main(void)
32 {
33
34
35
36
37     // モータ制御用ファイルをオープンします。
38
39
40
41
42
43
44
45     // ボタン制御用ファイルをオープンします。
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64

```

```

65
66
67 // モータの回転数を指定し、モータ状態変更関数を呼び出します。
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91 // モータ制御用ファイルをクローズします。
92
93 // ボタン制御用ファイルをクローズします。
94
95 return 0;
96 }

```

Makefile

▼ 05.motor/Makefile

```

1 CC = arm-linux-gnueabi-gcc
2 #TARGET = mtctl mtfan mtfan2
3 TARGET = mtfan
4 CFLAGS = -gdwarf-2 -O0
5
6 all: $(TARGET)
7
8 install :
9     cp -p $(TARGET) /work/linux/nfsroot/debug/04_practice
10    cp -p $(TARGET) /media/sf_ArmadilloX1/hwpwm/dbg/debug_share_hwpwm/R06_2024/04_practice
11    cp -p $(TARGET).c /media/sf_ArmadilloX1/hwpwm/dbg/debug_share_hwpwm/R06_2024/04_practice
12
13 clean:
14     rm -f $(TARGET)
15
16 .PHONY: clean

```

1.6.2. 動作確認

make clean

▼ \$ make clean

```

1 atmark@atde8:/media/sf_ArmadilloX1/hwpwm/work/R06_2024/Application_debug/text/practice-example/05.motor$ make clean
2 rm -f mtfan

```

make

▼ \$ make

```
1 | atmark@atde8:/media/sf_ArmadilloX1/hwpwm/work/R06_2024/Application_debug/text/practice-example/05.motor$ make
2 | arm-linux-gnueabi-gcc -gdwarf-2 -O0 mtfan.c -o mtfan
```

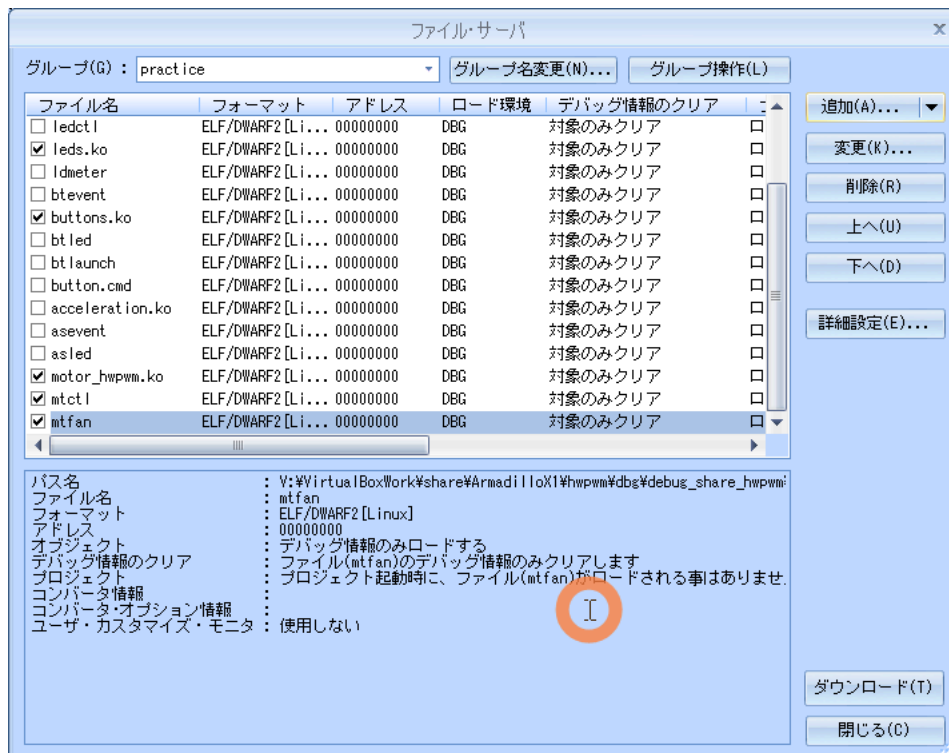
sudo make install

▼ \$ sudo make install

```
1 | atmark@atde8:/media/sf_ArmadilloX1/hwpwm/work/R06_2024/Application_debug/text/practice-example/05.motor$ sudo mak
2 | [sudo] atmark のパスワード:
3 | cp -p mtfan /work/linux/nfsroot/debug/04_practice
4 | cp -p mtfan /media/sf_ArmadilloX1/hwpwm/dbg/debug_share_hwpwm/R06_2024/04_practice
5 | cp -p mtfan.c /media/sf_ArmadilloX1/hwpwm/dbg/debug_share_hwpwm/R06_2024/04_practice
```

CSIDEでロード

▼ メニュー「ファイル」 - 「ロード」



insmod (既にinsmod 済みなら割愛)

▼ # insmod leds.ko と # insmod motor_hwpwm.ko

```
1 | root@armadillo:/lib/modules/4.9.133-at27/extra# insmod motor_hwpwm.ko
2 | root@armadillo:/lib/modules/4.9.133-at27/extra# insmod leds.ko
3 | root@armadillo:/lib/modules/4.9.133-at27/extra# insmod buttons.ko
4 | root@armadillo:/lib/modules/4.9.133-at27/extra# lsmod
5 | Module                Size  Used by
6 | buttons                3065  0
7 | leds                   2103  0
8 | motor_hwpwm            4415  0
```

実行

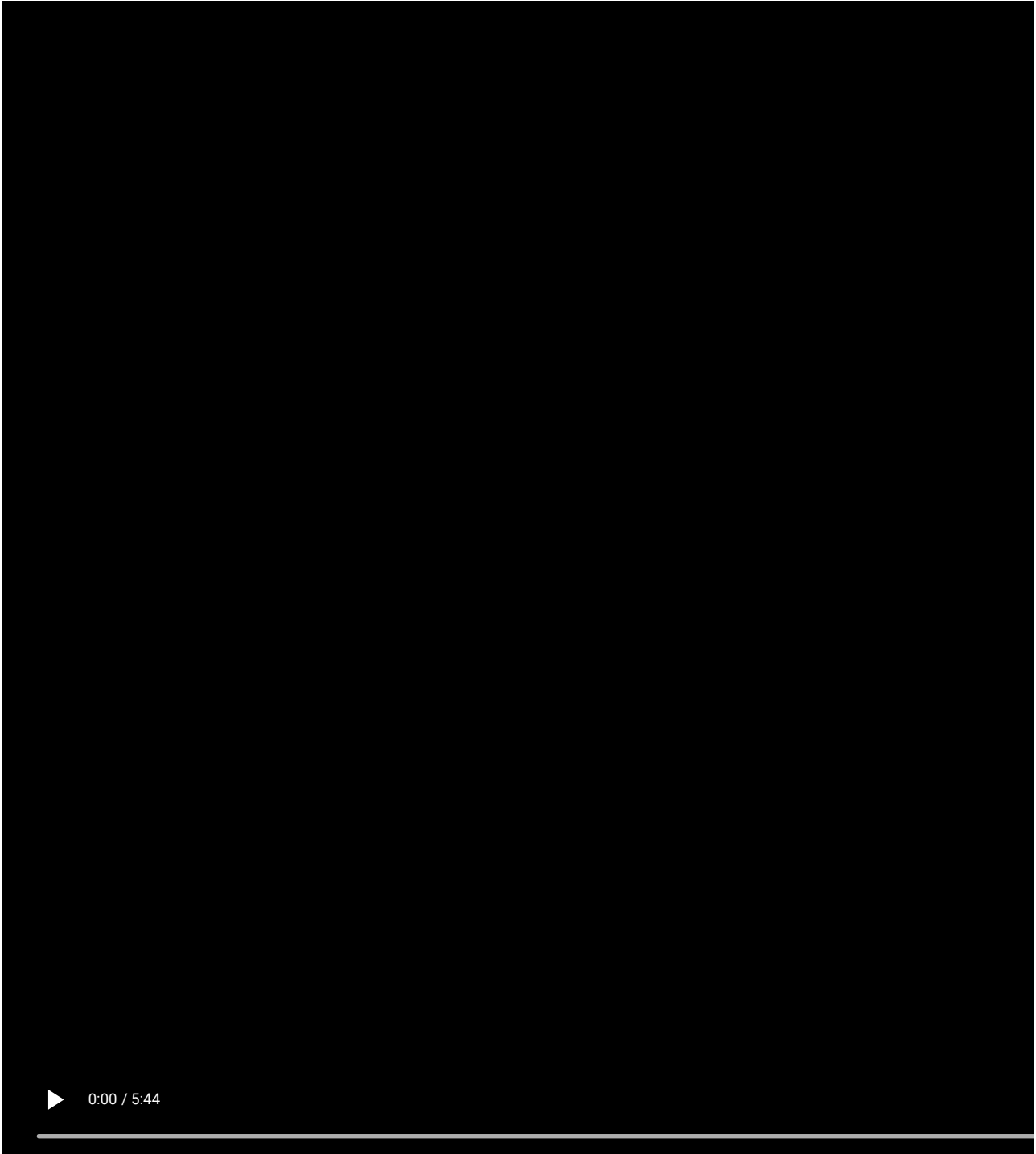
▼ root@armadillo:/debug/04_practice# ./mtfan

```
1 | root@armadillo:/debug/04_practice# ./mtfan
```

実行している様子



<https://youtu.be/yuZ88HaSEAw>



1.7. 課題2 mtfan2

LEDのインジケータを追加

bs	mode	speed	LED
SW1	stop	0	ALL OFF
SW2	弱	50	LED1～4 ON
SW3	強	100	ALL ON

1.7.1. ソース

mtfan2.c

▼ 05.motor/mtfan2.c

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <fcntl.h>
4  #include <unistd.h>
5  #include <linux/input.h>
6
7  // ボタン制御用ファイル
8  #define BUTTON_FILE      "/dev/input/event3"
9  // LED制御用ファイル
10 #define LED_FILE "/sys/class/leds/led_ext/brightness"
11 // モーター制御用ファイル
12 #define MOTOR_FILE      "/sys/class/motor/motor0/motor_rotation"
13 #define DATALEN        16
14
15 int fd_mt, fd_led;
16
17 void change_motor(int rotation)
18 {
19
20
21
22
23
24
25
26
27
28
29
30
31 }
32
33 void change_led(int num)
34 {
35
36
37
38
39
40
41
42
43
44
45
46
47 }
48
49 int main(void)
50 {
51
52
53
54
55     // モーター制御用ファイルをオープンします。
56
57
58
59
60
61
62
63     // LED制御用ファイルをオープンします。
64

```

```
65
66
67
68
69
70
71 // ボタン制御用ファイルをオープンします。
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93 // モータの回転数を指定し、モータ状態変更関数を呼び出します。
94
95 // LEDの点灯数を指定し、LED状態変更関数を呼び出します。
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123 // モータ制御用ファイルをクローズします。
124
125 // LED制御用ファイルをクローズします。
126
127 // ボタン制御用ファイルをクローズします。
128
129 return 0;
```

```
130 | }
```

Makefile

▼ 05.motor/Makefile

```
1 CC = arm-linux-gnueabi-gcc
2 #TARGET = mtctl mtfan mtfan2
3 TARGET = mtfan2
4 CFLAGS = -gdwarf-2 -O0
5
6 all: $(TARGET)
7
8 install :
9     cp -p $(TARGET) /work/linux/nfsroot/debug/04_practice
10    cp -p $(TARGET) /media/sf_ArmadilloX1/hwpwm/dbg/debug_share_hwpwm/R06_2024/04_practice
11    cp -p $(TARGET).c /media/sf_ArmadilloX1/hwpwm/dbg/debug_share_hwpwm/R06_2024/04_practice
12
13 clean:
14     rm -f $(TARGET)
15
16 .PHONY: clean
```

1.7.2. 動作確認

make clean

▼ \$ make clean

```
1 atmark@atde8:/media/sf_ArmadilloX1/hwpwm/work/R06_2024/Application_debug/text/practice-example/05.motor$ make cle
2 rm -f mtfan2
```

make

▼ \$ make

```
1 atmark@atde8:/media/sf_ArmadilloX1/hwpwm/work/R06_2024/Application_debug/text/practice-example/05.motor$ make
2 arm-linux-gnueabi-gcc -gdwarf-2 -O0 mtfan2.c -o mtfan2
```

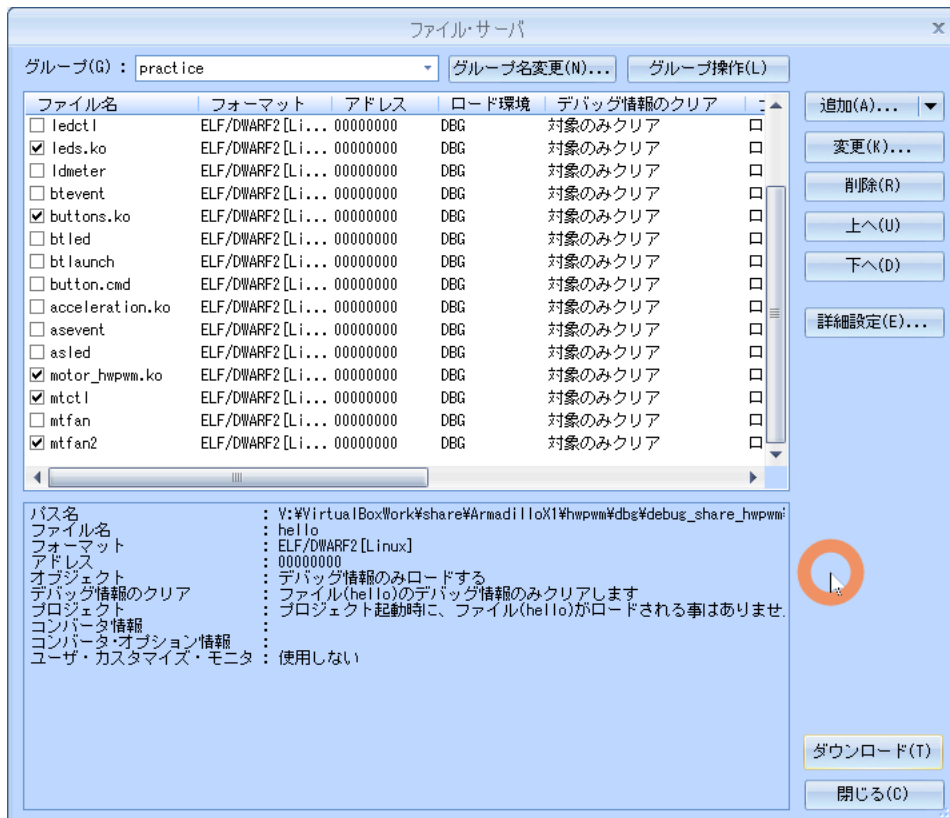
sudo make install

▼ \$ sudo make install

```
1 atmark@atde8:/media/sf_ArmadilloX1/hwpwm/work/R06_2024/Application_debug/text/practice-example/05.motor$ sudo mak
2 [sudo] atmark のパスワード:
3 cp -p mtfan2 /work/linux/nfsroot/debug/04_practice
4 cp -p mtfan2 /media/sf_ArmadilloX1/hwpwm/dbg/debug_share_hwpwm/R06_2024/04_practice
5 cp -p mtfan2.c /media/sf_ArmadilloX1/hwpwm/dbg/debug_share_hwpwm/R06_2024/04_practice
```

CSIDEでロード

▼ メニュー「ファイル」 - 「ロード」



insmod (既にinsmod 済みなら割愛)

▼ # insmod leds.ko と # insmod motor_hwpwm.ko

```

1 | root@armadillo:/lib/modules/4.9.133-at27/extra# insmod motor_hwpwm.ko
2 | root@armadillo:/lib/modules/4.9.133-at27/extra# insmod leds.ko
3 | root@armadillo:/lib/modules/4.9.133-at27/extra# insmod buttons.ko
4 | root@armadillo:/lib/modules/4.9.133-at27/extra# lsmod
5 | Module                Size  Used by
6 | buttons                3065  0
7 | leds                   2103  0
8 | motor_hwpwm            4415  0

```

実行

▼ root@armadillo:/debug/04_practice# ./mtfan

```

1 | root@armadillo:/debug/04_practice# ./mtfan

```

実行している様子

▼

<https://youtu.be/8gDiPCHvqsw>

