

1. 03.button

2024/7/18 Table of Contents

03.button

目的

構成データ

ボタンスイッチ制御

デバイスドライバ

例題 btevent

課題1 btled

課題2 btlaunch

1.1. 目的

組込みアプリケーション開発 03.button

1.2. 構成データ

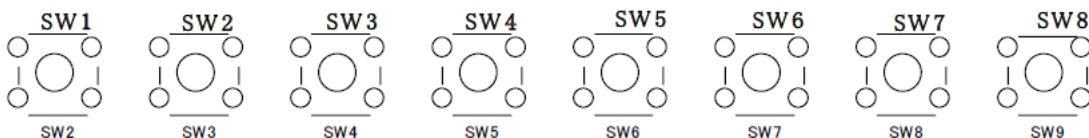
1.2.1. /media/sf_ArmadilloX1/hwpwm/work/R06_2024/Apllication_debug/text/practiceディレクトリ

▼ .../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/Application_debug/text/practice$ tr
2 ./
3 | 03.button/
4 | | btevent.c*          <----- 例題 デバイス制御用ソース
5 | | btlaunch.c*       <----- 課題2 デバイス制御用ソース
6 | | btled.c*           <----- 課題1 デバイス制御用ソース
7 | | button.cmd*        <----- 課題2 コマンドリスト
8 | | drivers/
9 | | | buttons/
10 | | | | buttons.c*     <----- ドライバソース
11 | | | | Makefile*      <----- ドライバ用Makefile
12 | | | | leds/
13 | | | | Makefile*      <----- デバイス制御用Makefile
14
```

1.3. ボタンスイッチ制御

1.3.1. デバイス仕様



1.4. デバイスドライバ

1.4.1. ソース

button.c

▼ 03.button/drivers/button.c

```

#include <linux/module.h>
#include <linux/input.h>
#include <linux/interrupt.h>
#include <linux/platform_device.h>
#include <linux/gpio.h>

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

#define CPLD_GPIO_SW1    138 /* GPIO5_I010 */
#define CPLD_GPIO_SW2    139 /* GPIO5_I011 */
#define CPLD_GPIO_SW3    140 /* GPIO5_I012 */
#define CPLD_GPIO_SW4    141 /* GPIO5_I013 */
#define CPLD_GPIO_SW5    142 /* GPIO5_I014 */
#define CPLD_GPIO_SW6    143 /* GPIO5_I015 */
#define CPLD_GPIO_SW7    144 /* GPIO5_I016 */
#define CPLD_GPIO_SW8    145 /* GPIO5_I017 */

#define BUTTON_NR        8

static unsigned short buttons_map[] = {
    BTN_0, BTN_1, BTN_2, BTN_3, BTN_4, BTN_5, BTN_6, BTN_7,
};

static unsigned int buttons_gpio[] = {
    CPLD_GPIO_SW1,
    CPLD_GPIO_SW2,
    CPLD_GPIO_SW3,
    CPLD_GPIO_SW4,
    CPLD_GPIO_SW5,
    CPLD_GPIO_SW6,
    CPLD_GPIO_SW7,
    CPLD_GPIO_SW8,
};

static int irq_sw[BUTTON_NR];

// 割込みでのボタン状態取得関数(buttons_irq)
static irqreturn_t buttons_irq(int irq, void *dev)
{
    unsigned char buf;
    struct input_dev *input = dev;
    int data_sw[BUTTON_NR] = {0};
    int i;

    // ボタンの状態を取得します。(cpld_read)
    buf = cpld_read(CPLD_READ_SW);
    for (i = 0; i < ARRAY_SIZE(buttons_map); i++) {
        // 取得した値を1ビットずつ取り出して各ボタンの状態を取得します。
        data_sw[i] = (buf >> i) & 0x01;
        // ドライバから入力されたボタンの状態を検索・設定します。(input_report_key)
        input_report_key(input, buttons_map[i], data_sw[i]);
    }

    // 設定されたボタン情報を通知します。(input_sync)
    input_sync(input);

    return IRQ_HANDLED;
}

// probe関数(buttons_probe)
static int buttons_probe(struct platform_device *pdev)
{
    struct input_dev *input;
    int error = 0;

```

```

int i;

// デバイス構造体を初期化します。(input_allocate_device)
input = input_allocate_device();
if (!input) {
    error = -ENOMEM;
    goto err_ret;
}

// デバイス構造体へ情報設定をします。
input->name = "armadillo-x1-extension-btms";
input->phys = "armadillo-x1/input1";
input->id.bustype = BUS_HOST;
input->dev.parent = &pdev->dev;

input->keycode = buttons_map;
input->keycodemax = ARRAY_SIZE(buttons_map);
input->keycodesize = sizeof(unsigned short);

// KEYイベントを登録します。(set_bit)
// ->この設定によって、デバイスがどのような機能を持つのかを設定します。
set_bit(EV_KEY, input->evbit);
for (i = 0; i < ARRAY_SIZE(buttons_map); i++)
    // KEYイベントとボタンとの関連付けをします。(set_bit)
    set_bit(buttons_map[i], input->keybit);

// デバイス構造体の情報を登録します。(dev_set_drvdata)
dev_set_drvdata(&pdev->dev, input);

// GPIOに割り当てられているIRQ番号を取得します。(gpio_to_irq)
for (i = 0; i < ARRAY_SIZE(buttons_gpio); i++) {
    irq_sw[i] = gpio_to_irq(buttons_gpio[i]);
    if (irq_sw[i] < 0)
        goto err_free_mem;
}

// 割り込みハンドラを登録します。(request_irq)
// ->IRQと割り込みハンドラをカーネルに登録します。
// ボタン毎に割り込み番号が異なるため、ボタン毎に割り込みを登録します。
for (i = 0; i < ARRAY_SIZE(buttons_gpio); i++) {
    error = request_irq(irq_sw[i], buttons_irq, IRQF_TRIGGER_RISING | IRQF_TRIGGER_FALLING, "armadill
    if (error)
        goto err_free_irq;
}

// デバイスを登録します。(input_register_device)
error = input_register_device(input);
if (error)
    goto err_free_irq;

return 0;

err_free_irq:
// 割り込みハンドラ情報を削除します。(free_irq)
for (i = 0; i < ARRAY_SIZE(buttons_gpio); i++) {
    free_irq(irq_sw[i], input);
}
err_free_mem:
// デバイス構造体を解放します。(input_free_device)
input_free_device(input);
err_ret:
return error;
}

// remove関数(buttons_remove)

```

```

static int buttons_remove(struct platform_device *pdev)
{
    int i;

    // デバイス構造体の情報を取得します。(dev_get_drvdata)
    struct input_dev *input = dev_get_drvdata(&pdev->dev);

    // デバイスを解除します。(input_unregister_device)
    input_unregister_device(input);

    // 割り込みハンドラ情報を解放します。(free_irq)
    for (i = 0; i < ARRAY_SIZE(buttons_gpio); i++) {
        free_irq(irq_sw[i], input);
    }

    // デバイス構造体を解放します。(input_free_device)
    input_free_device(input);

    return 0;
}

// プラットフォームドライバ
static struct platform_driver buttons_driver = {
    .probe = buttons_probe,
    .remove = buttons_remove,
    .driver = {
        .name = "armadillo-x1-extension-btms",
        .owner = THIS_MODULE,
    },
};

static struct platform_device *pdev;

// 初期化関数(buttons_init)
static int __init buttons_init(void)
{
    int ret;

    // プラットフォームデバイスを登録します。(platform_device_register_simple)
    // ->プラットフォーム依存のデバイス情報を登録します。
    pdev = platform_device_register_simple("armadillo-x1-extension-btms", -1, NULL, 0);
    if (IS_ERR(pdev)) {
        ret = (int)pdev;
        goto err_ret;
    }

    // プラットフォームドライバを登録します。(platform_driver_register)
    // ->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、
    // 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。
    ret = platform_driver_register(&buttons_driver);
    if (ret < 0)
        goto err_platform_device_unregister;

    return 0;

err_platform_device_unregister:
    platform_device_unregister(pdev);
err_ret:
    return ret;
}

// 終了関数(buttons_exit)
static void __exit buttons_exit(void)
{
    // プラットフォームドライバを解除します。(platform_driver_unregister)

```

```

195     platform_driver_unregister(&buttons_driver);
196     // プラットフォームデバイスを解除します。(platform_device_unregister)
197     platform_device_unregister(pdev);
198 }
199
200 // 初期化の際に、初期化関数が呼ばれるように登録します。
201 module_init(buttons_init);
202 // 終了する際に、終了関数が呼ばれるように登録します。
203 module_exit(buttons_exit);
204
205 // MODULE_LICENSEは"GPL"とします。
206 MODULE_LICENSE("GPL");

```

Makefile

▼ /media/sf_ArmadilloX1/hwpwm/work/R06_2024/Application_debug/text/practice-example/02.led/drivers/leds/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 := buttons.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/03.button/drivers/
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  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/03.button/drivers/
2 make -C /home/atmark/linux-4.9-x1-at27_dbg M=`pwd` ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- 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/03.button/drivers/but
5 Building modules, stage 2.
6 MODPOST 1 modules
7 make[2]: 警告: ファイル '/media/sf_ArmadilloX1/hwpwm/work/R06_2024/Application_debug/text/practice-example/03.butt
8 CC /media/sf_ArmadilloX1/hwpwm/work/R06_2024/Application_debug/text/practice-example/03.button/drivers/but
9 LD [M] /media/sf_ArmadilloX1/hwpwm/work/R06_2024/Application_debug/text/practice-example/03.button/drivers/but
10 make[2]: 警告: 時刻のずれを検出。不完全なビルド結果になるかもしれません。
11 make[1]: ディレクトリ '/home/atmark/linux-4.9-x1-at27_dbg' から出ます
12 atmark@atde8:/media/sf_ArmadilloX1/hwpwm/work/R06_2024/Application_debug/text/practice-example/03.button/drivers/

```

sudo make modules_install

▼ \$ sudo make modules_install

```

1 atmark@atde8:/media/sf_ArmadilloX1/hwpwm/work/R06_2024/Application_debug/text/practice-example/03.button/drivers/
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/03.button/drivers/but
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
10 atmark@atde8:/media/sf_ArmadilloX1/hwpwm/work/R06_2024/Application_debug/text/practice-example/03.button/drivers/

```

sudo make myinstall

▼ \$ sudo make myinstall

```

1 atmark@atde8:/media/sf_ArmadilloX1/hwpwm/work/R06_2024/Application_debug/text/practice-example/03.button/drivers/
2 cp -p *.c /media/sf_ArmadilloX1/hwpwm/dbg/debug_share_hwpwm/R06_2024/04_practice
3 cp -p *.ko /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

▼ root@armadillo:/lib/modules/4.9.133-at27/extra# insmod leds.ko

```

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

```

1.4.3. デバイスファイル

"/dev/input/event**"

*には連番

⚠ ボタンスイッチ、センサなど複数の入力デバイスがある場合、ソースファイル内ではデバイスファイル/dev/input/event*の*を**決め打ち**しているので注意すること

ボタンイベント

読み出したイベントデータは次のinput_event構造体の形で表示

▼ input_event構造体

```
1 #include <linux/input.h>
2
3 struct input_event {
4     struct timeval time;
5     __u16 type;
6     __u16 code;
7     __s32 value;
8 };
```

type	code	value
EV_SYN(0)	0	0
EV_KEY(1)	BTN_0(256)	1(押下), 0(解放)
	BTN_1(257)	
	BTN_2(258)	
	BTN_3(259)	
	BTN_4(260)	
	BTN_5(261)	
	BTN_6(262)	
	BTN_7(263)	

- ボタンを「押す」または「離す」イベントが発生するとEV_KEY（キー入力）イベントがtypeフィールドに格納
- ボタンの種類とイベントの内容はcodeとvalueフィールドに格納
- 一連のイベントの終わりにEV_SYN（同期）イベントが発生
- ボタンイベントの場合はEV_KEYイベントの直後に必ずEV_SYNイベントが発生

1.5. 例題 btevent

ボタンイベントの内容を表示する

- Buttonが押されたとき --> "EV_KEY: button N down" (Nは1-8)
- Buttonが離されたとき --> "EV_KEY: button N up" (Nは1-8)
- 同期イベント発生時 --> "EV_SYN:-----"

1.5.1. ソース

btevent.c

▼ 03.button/btevent.c


```

1  #include <stdio.h>
2  #include <sys/types.h>
3  #include <sys/stat.h>
4  #include <fcntl.h>
5  #include <unistd.h>
6  #include <linux/input.h>
7
8  // ボタン制御用ファイル
9  #define BTEV_FILE      "/dev/input/event3"
10
11 int main(void)
12 {
13     int fd;
14     int ret;
15     struct input_event ev;
16
17     // ボタン制御用ファイルをオープンします。
18     fd = open(BTEV_FILE, O_RDONLY);
19     // オープンに失敗したら、main関数をエラー終了します。
20     if (fd < 0){
21         perror("failed to open device\n");
22         return 1;
23     }
24
25     for(;;){
26         // ボタンの押下状態をリードします。
27         ret = read(fd, &ev, sizeof(ev));
28         // リードに失敗したら、main関数をエラー終了します。
29         if (ret < 0){
30             perror("failed to read events");
31             return 1;
32         }
33
34         switch (ev.type){
35             case EV_KEY:
36                 // ボタンを「押す」または「離す」イベントが発生したことを表示します。
37                 printf("EV_KEY: button %d %s\n",
38                     ev.code - BTN_0 + 1, ev.value ? "down" : "up");
39                 break;
40             case EV_SYN:
41                 // 同期イベントが発生したことを表示します。
42                 printf("EV_SYN: -----\n");
43                 break;
44             default:
45                 // 不明なイベントが発生したら、main関数をエラー終了します。
46                 fprintf(stderr, "unknown event\n");
47                 return 2;
48         }
49     }
50
51     // ボタン制御用ファイルをクローズします。
52     close(fd);
53
54     return 0;
55 }

```

Makefile

▼ 03.button/Makefile

```

1 CC = arm-linux-gnueabi-gcc
2 #TARGET = btevent btled btlaunch
3 TARGET = btevent
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/03.button$ make cl
2 rm -f btevent

```

make

▼ \$ make

```

1 atmark@atde8:/media/sf_ArmadilloX1/hwpwm/work/R06_2024/Application_debug/text/practice-example/03.button$ make
2 arm-linux-gnueabi-gcc -gdwarf-2 -O0 btevent.c -o btevent

```

sudo make install

▼ \$ sudo make install

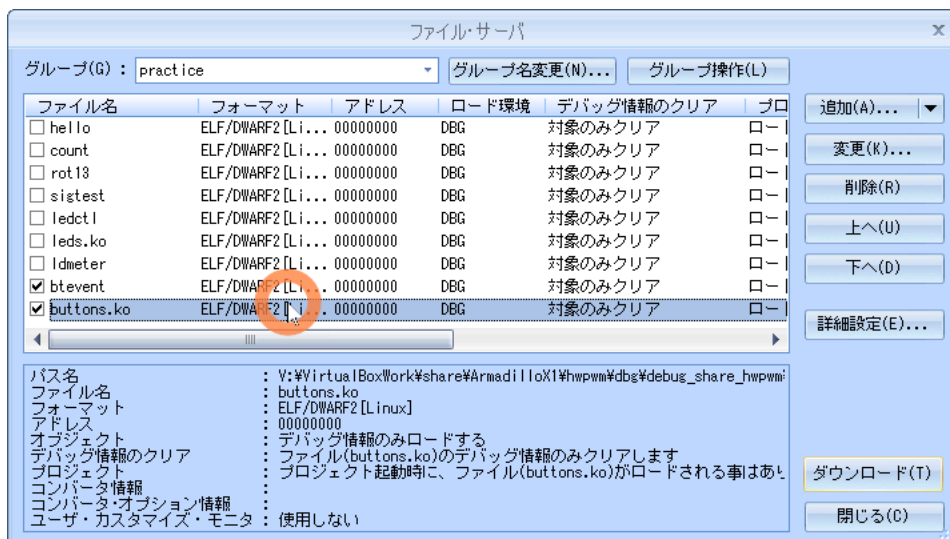
```

1 atmark@atde8:/media/sf_ArmadilloX1/hwpwm/work/R06_2024/Application_debug/text/practice-example/03.button$ sudo mc
2 cp -p btevent /work/linux/nfsroot/debug/04_practice
3 cp -p btevent /media/sf_ArmadilloX1/hwpwm/dbg/debug_share_hwpwm/R06_2024/04_practice
4 cp -p btevent.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 と # insmod buttons.ko

```

1 root@armadillo:/debug/04_practice# cd /lib/modules/4.9.133-at27/extra/
2
3 root@armadillo:/lib/modules/4.9.133-at27/extra# insmod buttons.ko
4
5 root@armadillo:/lib/modules/4.9.133-at27/extra# lsmod
6 Module          Size Used by
7 buttons         3065  0
8 leds            2103  0

```

実行結果

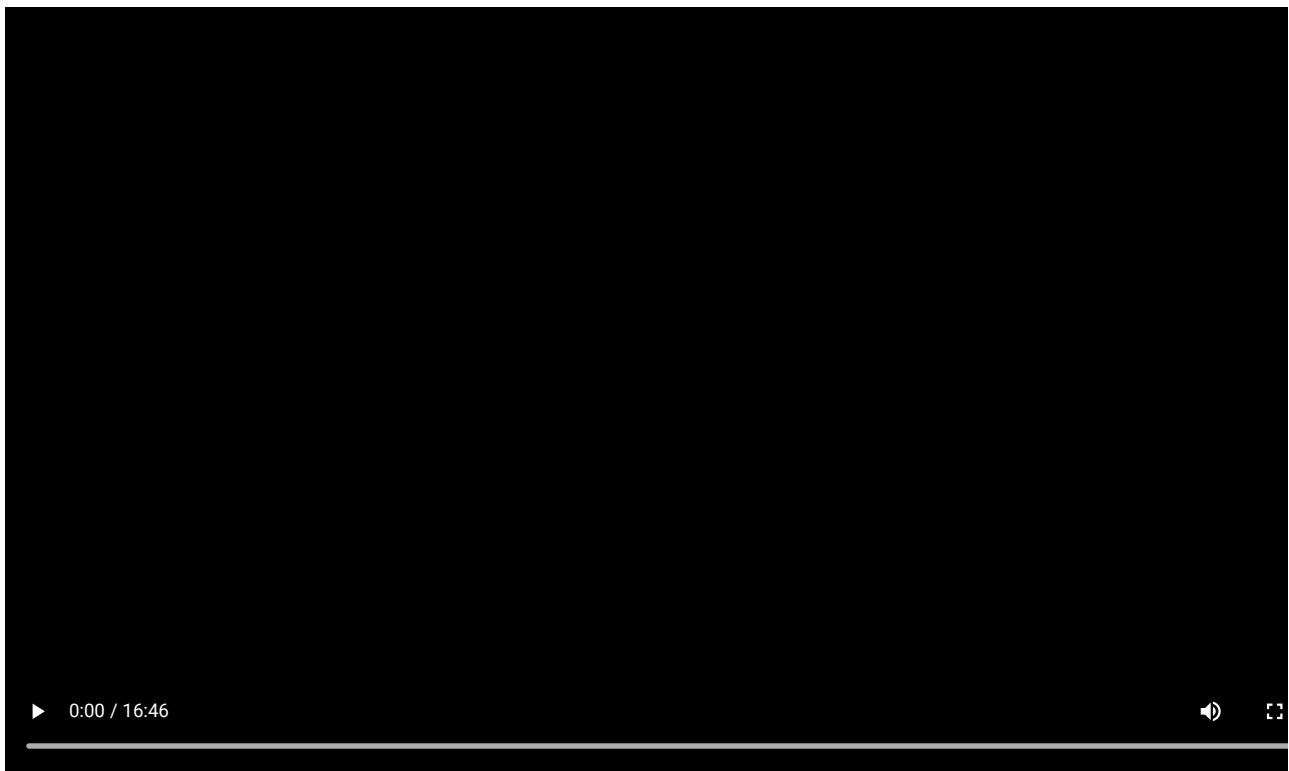
▼ root@armadillo:/debug/04_practice# ./btevent

```
1 root@armadillo:/debug/04_practice# ./btevent
2 EV_KEY: button 1 down
3 EV_SYN: -----
4 EV_KEY: button 1 up
5 EV_SYN: -----
6 EV_KEY: button 1 down
7 EV_SYN: -----
8 EV_KEY: button 1 up
9 EV_SYN: -----
10 EV_KEY: button 2 down
11 EV_SYN: -----
12 EV_KEY: button 2 up
13 EV_SYN: -----
14 EV_KEY: button 3 down
15 EV_SYN: -----
16 EV_KEY: button 3 up
17 EV_SYN: -----
18 EV_KEY: button 4 down
19 EV_SYN: -----
20 EV_KEY: button 4 up
21 EV_SYN: -----
22 EV_KEY: button 5 down
23 EV_SYN: -----
24 EV_KEY: button 5 up
25 EV_SYN: -----
26 EV_KEY: button 6 down
27 EV_SYN: -----
28 EV_KEY: button 6 up
29 EV_SYN: -----
30 EV_KEY: button 7 down
31 EV_SYN: -----
32 EV_KEY: button 7 up
33 EV_SYN: -----
34 EV_KEY: button 8 down
35 EV_SYN: -----
36 EV_KEY: button 8 up
37 EV_SYN: -----
38 EV_KEY: button 1 down
39 EV_SYN: -----
40 EV_KEY: button 1 up
41 EV_SYN: -----
42 EV_KEY: button 1 down
43 EV_SYN: -----
44 EV_KEY: button 1 up
45 EV_SYN: -----
46 EV_KEY: button 1 down
47 EV_SYN: -----
48 EV_KEY: button 1 up
49 EV_SYN: -----
50 ^C
```

実行している様子

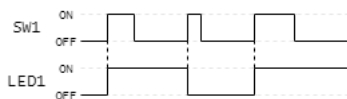
▼ button デバイスファイルによる ボタンイベントを実行している動画

<https://youtu.be/XWLqbuTBHVQ>



1.6. 課題1 btled

ボタンスイッチによる LED のオルタネート動作



SW2 ～ 8 も同様

1.6.1. ソース

btled.c

▼ 03.button/btled.c

```

1  #include <stdio.h>
2  #include <string.h>
3  #include <stdlib.h>
4  #include <sys/types.h>
5  #include <sys/stat.h>
6  #include <fcntl.h>
7  #include <unistd.h>
8  #include <linux/input.h>
9
10 #define BUFLen      1024
11 #define BUTTON_NR    8
12 // コマンドランチャーの設定ファイル
13 #define CMD_FILE      "button.cmd"
14 // ボタン制御用ファイル
15 #define BTEV_FILE     "/dev/input/event3"
16
17 char cmd[BUTTON_NR][BUFLen];
18
19 int read_config()
20 {
21
22
23
24
25
26
27     // コマンドランチャーの設定ファイルをオープンします。
28
29
30
31
32
33
34
35     // コマンドランチャーの設定ファイルからコマンドを1行読み込みます。
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64

```

```
65
66     // コマンドランチャーの設定ファイルをクローズします。
67
68     return 0;
69 }
70
71 int main(void)
72 {
73
74
75
76
77     // コマンドランチャーの設定ファイルからコマンドを読み込みます。
78
79
80
81     // ボタン制御用ファイルをオープンします。
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113     // ボタン制御用ファイルをクローズします。
114     close(fd);
115
116     return 0;
117 }
```

Makefile

▼ 03.button/Makefile

```

1 CC = arm-linux-gnueabi-gcc
2 #TARGET = btevent btled btlaunch
3 TARGET = btled
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/03.button$ make cl
2 rm -f btled

```

make

▼ \$ make

```

1 atmark@atde8:/media/sf_ArmadilloX1/hwpwm/work/R06_2024/Application_debug/text/practice-example/03.button$ make
2 arm-linux-gnueabi-gcc -gdwarf-2 -O0 btled.c -o btled

```

sudo make install

▼ \$ sudo make install

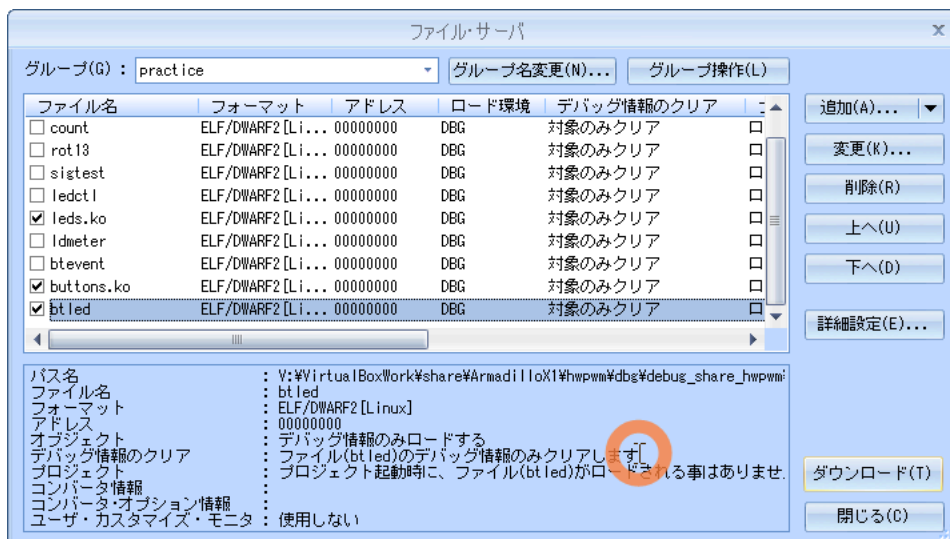
```

1 atmark@atde8:/media/sf_ArmadilloX1/hwpwm/work/R06_2024/Application_debug/text/practice-example/03.button$ sudo mc
2 [sudo] atmark のパスワード:
3 cp -p btled /work/linux/nfsroot/debug/04_practice
4 cp -p btled /media/sf_ArmadilloX1/hwpwm/dbg/debug_share_hwpwm/R06_2024/04_practice
5 cp -p btled.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 buttons.ko

```

1 root@armadillo:/debug/04_practice# cd /lib/modules/4.9.133-at27/extra/
2
3 root@armadillo:/lib/modules/4.9.133-at27/extra# insmod leds.ko
4
5 root@armadillo:/lib/modules/4.9.133-at27/extra# insmod buttons.ko
6
7 root@armadillo:/lib/modules/4.9.133-at27/extra# lsmod
8 Module                Size  Used by
9 buttons                3065  0
10 leds                   2103  0

```

実行

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

```

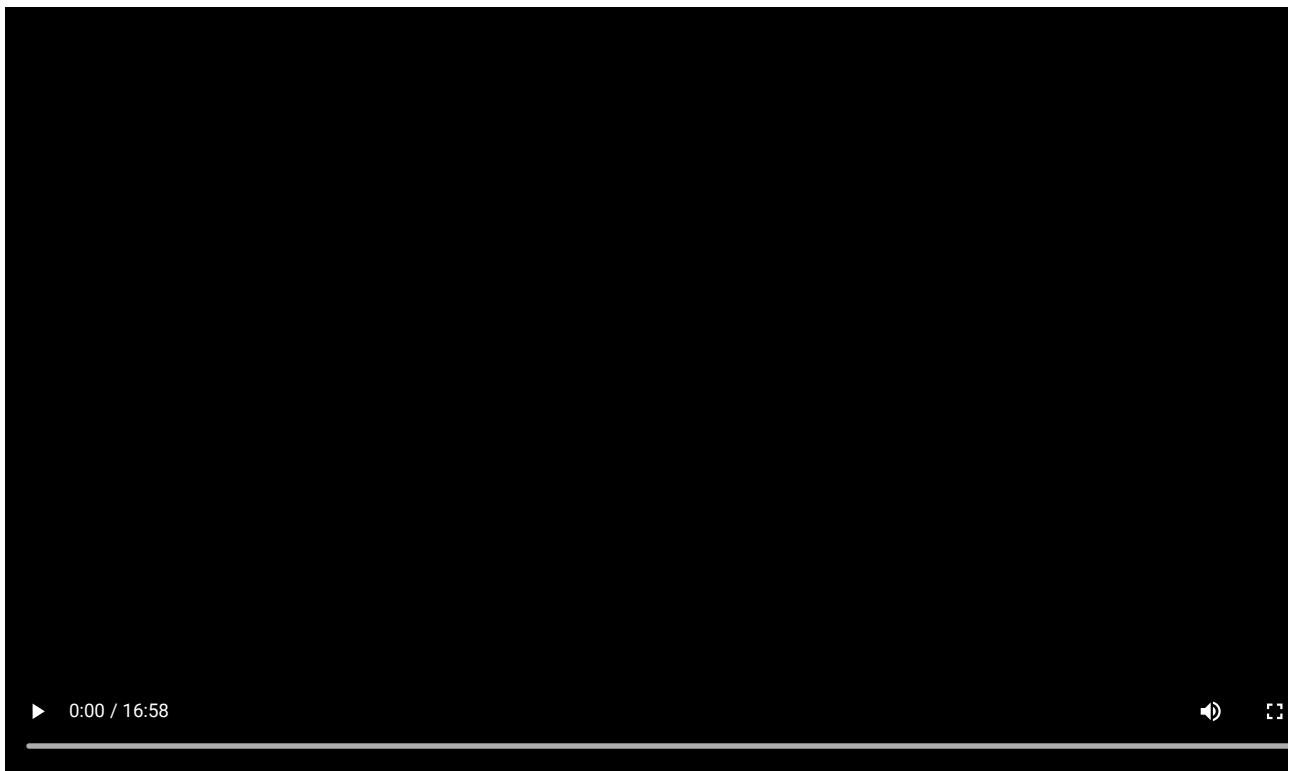
1 root@armadillo:/debug/04_practice# ./btled
2 ^C ← Ctrl + c

```

実行している様子

▼ btled による SW と LED の連動動画

<https://youtu.be/VSQp8s2DCQM>



1.7. 課題2 btlaunch

button.cmd に記載しているコマンドを 各ボタンスイッチで実行する

1.7.1. ソース

button.cmd

▼ 03.button/button.cmd

```
1 1: echo -n 1 > /sys/class/leds/led_ext/brightness
2 2: echo -n 2 > /sys/class/leds/led_ext/brightness
3 3: echo -n 4 > /sys/class/leds/led_ext/brightness
4 4: echo -n 8 > /sys/class/leds/led_ext/brightness
5 5: echo -n 16 > /sys/class/leds/led_ext/brightness
6 6: echo -n 32 > /sys/class/leds/led_ext/brightness
7 7: echo -n 64 > /sys/class/leds/led_ext/brightness
8 8: echo -n 128 > /sys/class/leds/led_ext/brightness
```

btlaunch.c

▼ 03.button/btlaunch.c

```

1  #include <stdio.h>
2  #include <string.h>
3  #include <stdlib.h>
4  #include <sys/types.h>
5  #include <sys/stat.h>
6  #include <fcntl.h>
7  #include <unistd.h>
8  #include <linux/input.h>
9
10 #define BUFLen      1024
11 #define BUTTON_NR    8
12 // コマンドランチャーの設定ファイル
13 #define CMD_FILE      "button.cmd"
14 // ボタン制御用ファイル
15 #define BTEV_FILE     "/dev/input/event3"
16
17 char cmd[BUTTON_NR][BUFLen];
18
19 int read_config()
20 {
21
22
23
24
25
26
27     // コマンドランチャーの設定ファイルをオープンします。
28
29
30
31
32
33
34
35     // コマンドランチャーの設定ファイルからコマンドを1行読み込みます。
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64

```

```
65
66     // コマンドランチャーの設定ファイルをクローズします。
67
68     return 0;
69 }
70
71 int main(void)
72 {
73
74
75
76
77     // コマンドランチャーの設定ファイルからコマンドを読み込みます。
78
79
80
81     // ボタン制御用ファイルをオープンします。
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113     // ボタン制御用ファイルをクローズします。
114     close(fd);
115
116     return 0;
117 }
```

Makefile



button.cmd をコピーする myinstall コマンドを追加

▼ 03.button/Makefile

```

1 CC = arm-linux-gnueabi-gcc
2 #TARGET = btevent btled btlaunch
3 TARGET = btlaunch
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 myinstall :
14     cp -p button.cmd /work/linux/nfsroot/debug/04_practice
15     cp -p button.cmd /media/sf_ArmadilloX1/hwpwm/dbg/debug_share_hwpwm/R06_2024/04_practice
16
17 clean:
18     rm -f $(TARGET)
19
20 .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/03.button$ make cl
2 rm -f btlaunch

```

make

▼ \$ make

```

1 atmark@atde8:/media/sf_ArmadilloX1/hwpwm/work/R06_2024/Application_debug/text/practice-example/03.button$ make
2 arm-linux-gnueabi-gcc -gdwarf-2 -O0 btlaunch.c -o btlaunch

```

sudo make install

▼ \$ sudo make install

```

1 atmark@atde8:/media/sf_ArmadilloX1/hwpwm/work/R06_2024/Application_debug/text/practice-example/03.button$ sudo m
2 [sudo] atmark のパスワード:
3 cp -p btlaunch /work/linux/nfsroot/debug/04_practice
4 cp -p btlaunch /media/sf_ArmadilloX1/hwpwm/dbg/debug_share_hwpwm/R06_2024/04_practice
5 cp -p btlaunch.c /media/sf_ArmadilloX1/hwpwm/dbg/debug_share_hwpwm/R06_2024/04_practice

```

sudo make myinstall


▼ \$ sudo make myinstall

```

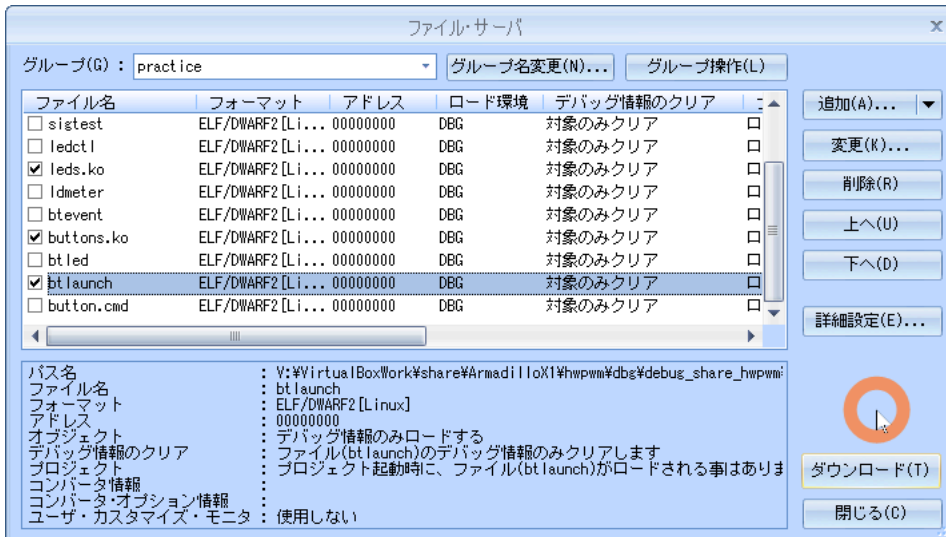
1 atmark@atde8:/media/sf_ArmadilloX1/hwpwm/work/R06_2024/Application_debug/text/practice-example/03.button$ sudo m
2 cp -p button.cmd /work/linux/nfsroot/debug/04_practice
3 cp -p button.cmd /media/sf_ArmadilloX1/hwpwm/dbg/debug_share_hwpwm/R06_2024/04_practice

```

CSIDEでロード

 CSIDE がロードできるのはバイナリファイルのみなので、button.cmd のロードは不要

▼ 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 buttons.ko

```

1 root@armadillo:/debug/04_practice# cd /lib/modules/4.9.133-at27/extra/
2
3 root@armadillo:/lib/modules/4.9.133-at27/extra# insmod leds.ko
4
5 root@armadillo:/lib/modules/4.9.133-at27/extra# insmod buttons.ko
6
7 root@armadillo:/lib/modules/4.9.133-at27/extra# lsmod
8 Module                Size Used by
9 buttons               3065  0
10 leds                  2103  0

```

実行

▼ root@armadillo:/debug/04_practice# ./btlaunch button.cmd

```

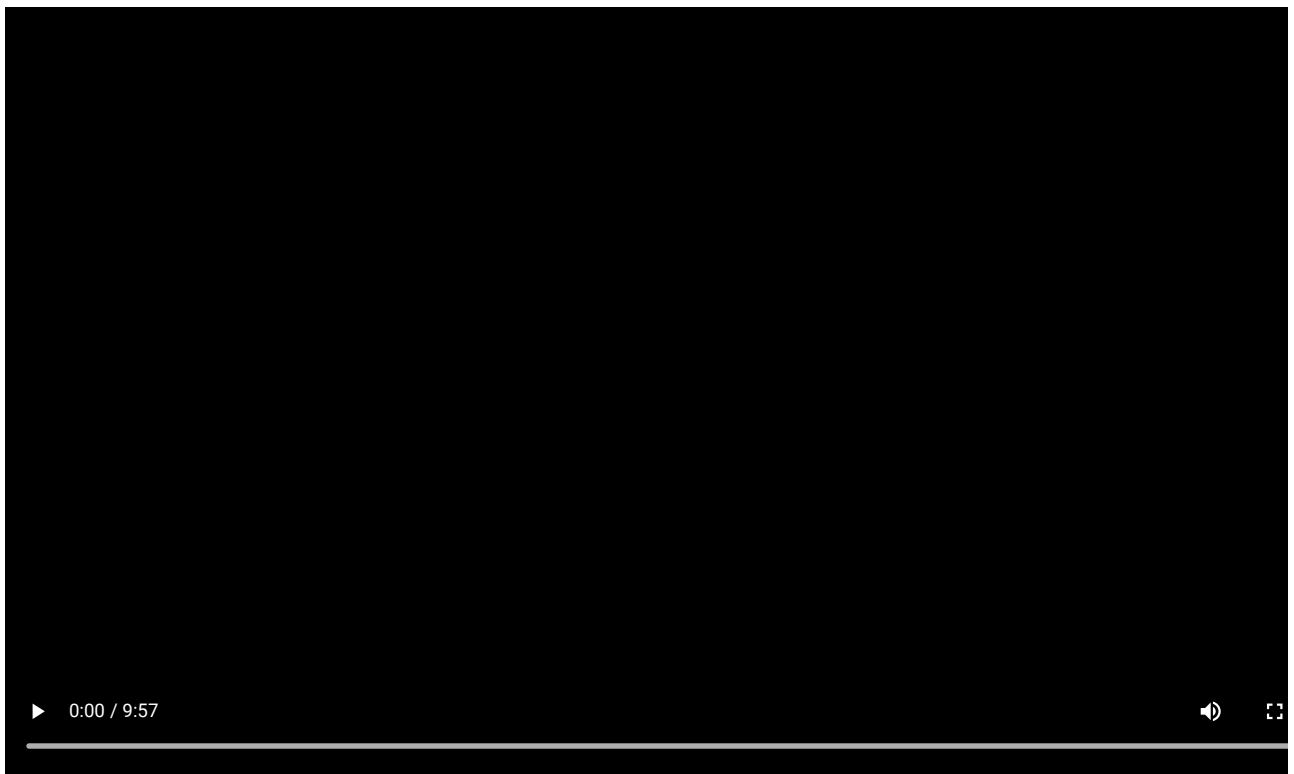
1 root@armadillo:/debug/04_practice# ./btlaunch button.cmd
2 1: echo -n 1 > /sys/class/leds/led_ext/brightness
3 2: echo -n 2 > /sys/class/leds/led_ext/brightness
4 3: echo -n 4 > /sys/class/leds/led_ext/brightness
5 4: echo -n 8 > /sys/class/leds/led_ext/brightness
6 5: echo -n 16 > /sys/class/leds/led_ext/brightness
7 6: echo -n 32 > /sys/class/leds/led_ext/brightness
8 7: echo -n 64 > /sys/class/leds/led_ext/brightness
9 8: echo -n 128 > /sys/class/leds/led_ext/brightness
10 ^C

```

実行している様子

▼ btlaunch による SW と LED の連動動画

<https://youtu.be/NWjMb6h09No>



1.7.3. ヒント

system