

制御機器製作Ⅱ訓練課題(実技) 「自走ロボット」

製作報告書

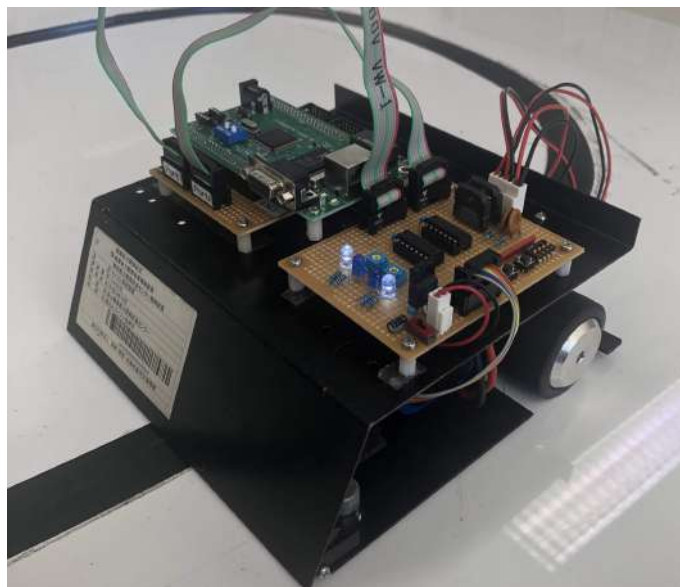
組込みソフトウェア科
令和元年7月生

島先 哲也

ライントレース「自走ロボ」製作

目次

- ❑ 全体仕様(課題内容)
- ❑ ハードウェア設計、製作
 - ❑ 使用機器、使用部品
 - ❑ 回路図
 - ❑ 基板
- ❑ ソフトウェア設計、制作
 - ❑ ソフトウェア仕様
 - ❑ 要求分析
 - ❑ 設計
 - ❑ 関数、変数表
 - ❑ ソースコード



仕様

- 入力 - フォト・マイクロセンサ ×2
- 出力 - DCモーター ×2
- 電源 - 充電式ニカド電池7.2v(モーター用)
- 乾電池9v
- 動作 - 図 1 のコースを、どちらの向きでも3周以上安定して走る。
- プログラムはRTOSカーネルの機能を使用し、
マルチタスクで作成する

図 1、走行コース

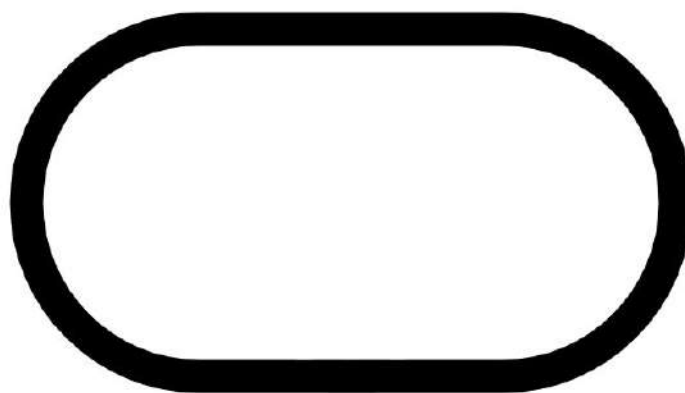
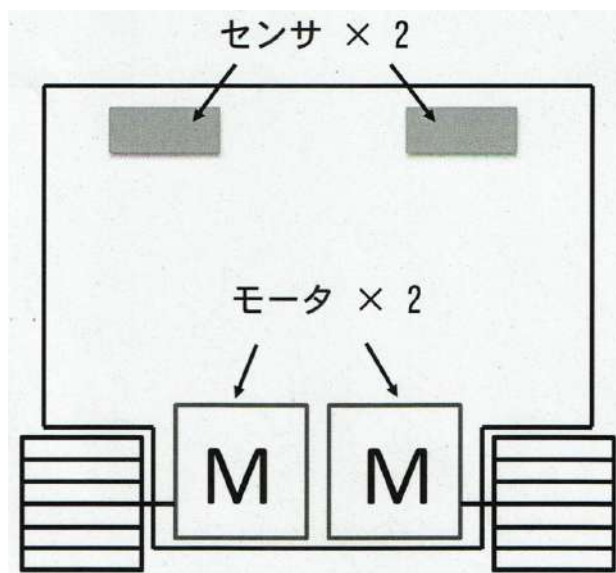


図 2、システム概要図



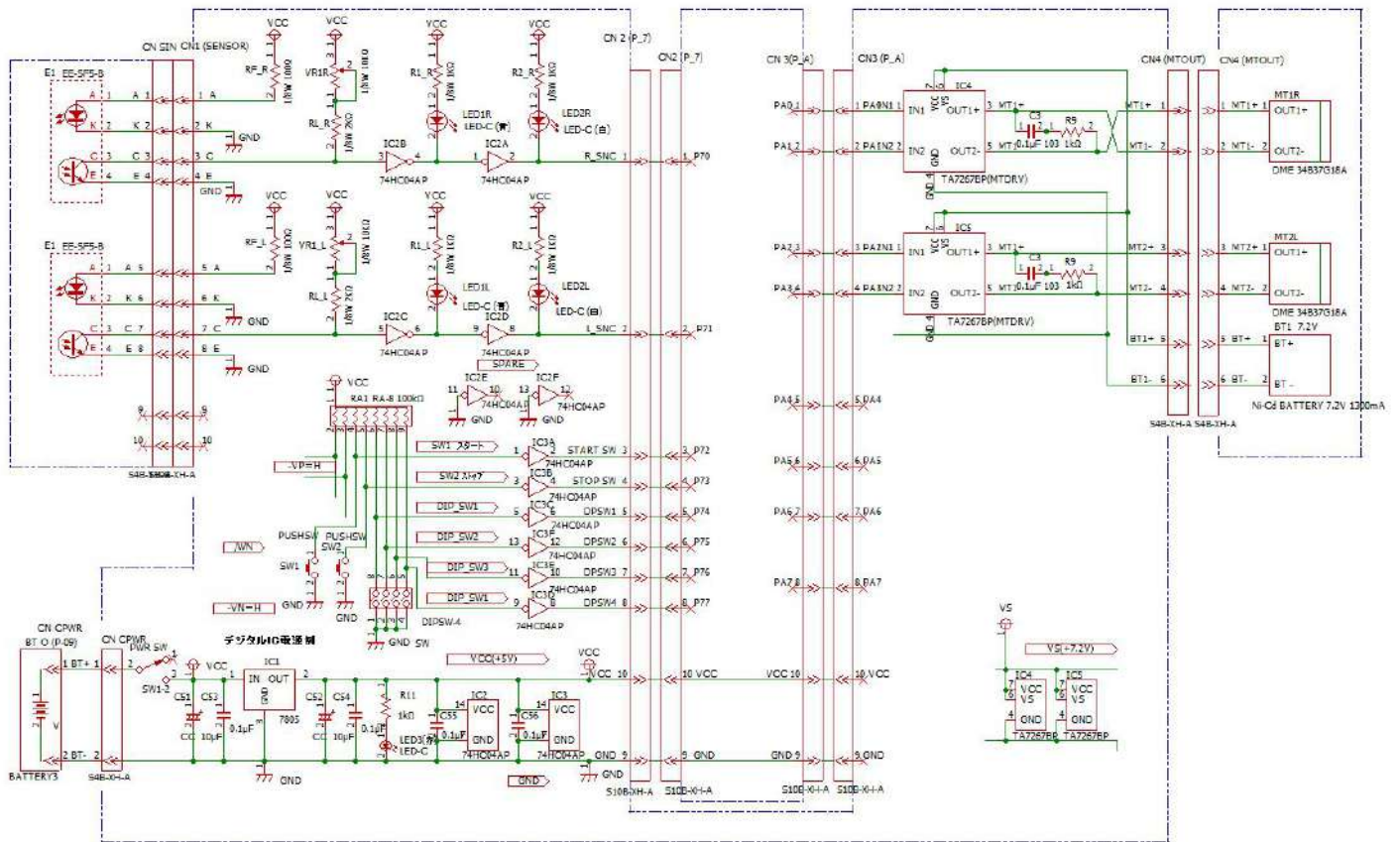
使用機器

- ☐ 自走ロボット筐体
- ☐ センサ(OMRON EE-SF5×2)
- ☐ モータ(JAPAN SARVO DME34B37G18A×2)
- ☐ 充電式ニカド電池(TAMIYA Ni-Cd BATTERY 7.2v 1300mAh)
- ☐ 乾電池(アルカリ 006P型 9V)
- ☐ 3端子レギュレータ(TA7805)
- ☐ モータドライバIC(東芝 TA7267BP×2)
- ☐ マイコンボード(H8/3069F)
- ☐ OS(Toppers JSP1.4)
- ☐ PC(AT互換機)

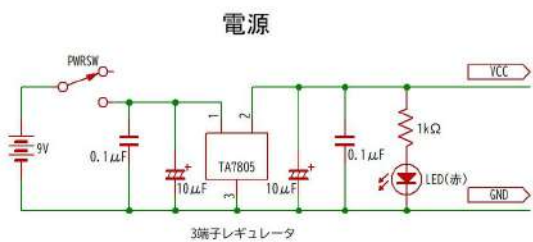
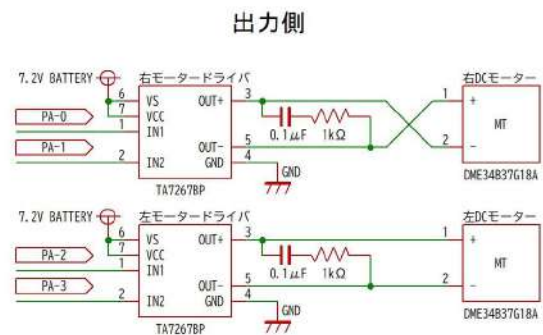
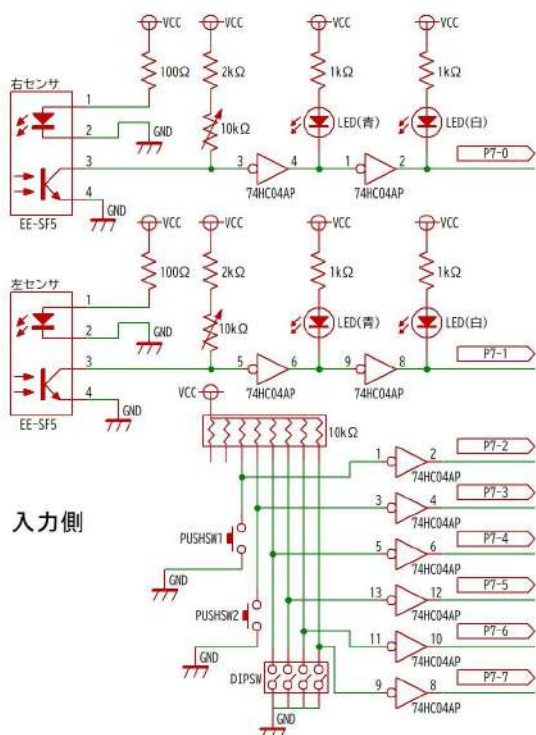
使用部品

- ☐ ユニバーサル基板×1
- ☐ LED 赤×1
- ☐ LED 青×2
- ☐ LED 白×2
- ☐ 抵抗(1k Ω)×7(LED用、モータドライバ安定化用)
- ☐ 抵抗(2k Ω)×2(センサ受光部用)
- ☐ 抵抗(100 Ω)×2(センサ発光部用)
- ☐ 可変抵抗器(10k Ω)×2(センサ受光感度調整用)
- ☐ 集合抵抗(10k Ω)×1(スイッチ用)
- ☐ デジタルIC74HC04×2
- ☐ 電解コンデンサ(10 μ F)×2
- ☐ 積層セラミックコンデンサ(0.1 μ F)×4
- ☐ セラミックコンデンサ(0.1 μ F)×2
- ☐ 3端子スライドスイッチ×1
- ☐ プッシュスイッチ×2
- ☐ 4極ディップスイッチ×1
- ☐ 10pinコネクタ×3(センサ用、portA,port7用)
- ☐ 6pinコネクタ×1(モータ、バッテリー用)
- ☐ 2pinコネクタ×1(電源用)
- ☐ DC電源スナップボタンケーブル

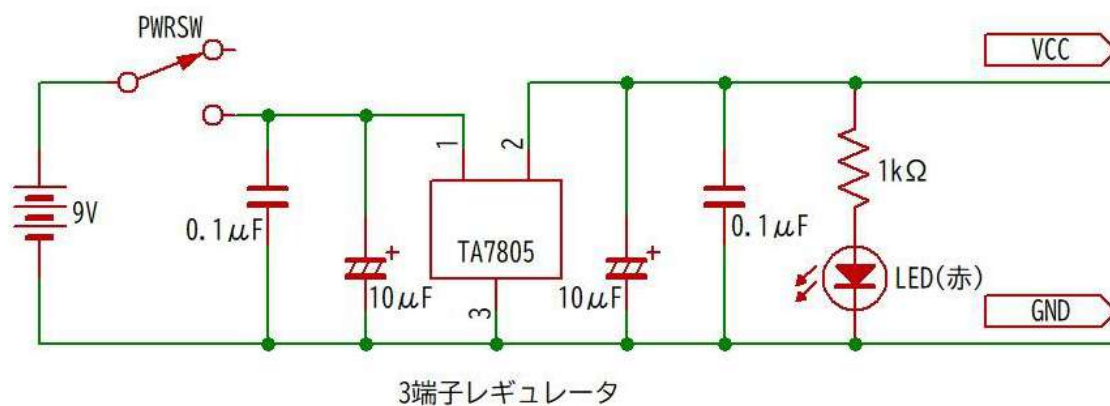
回路図：完全版



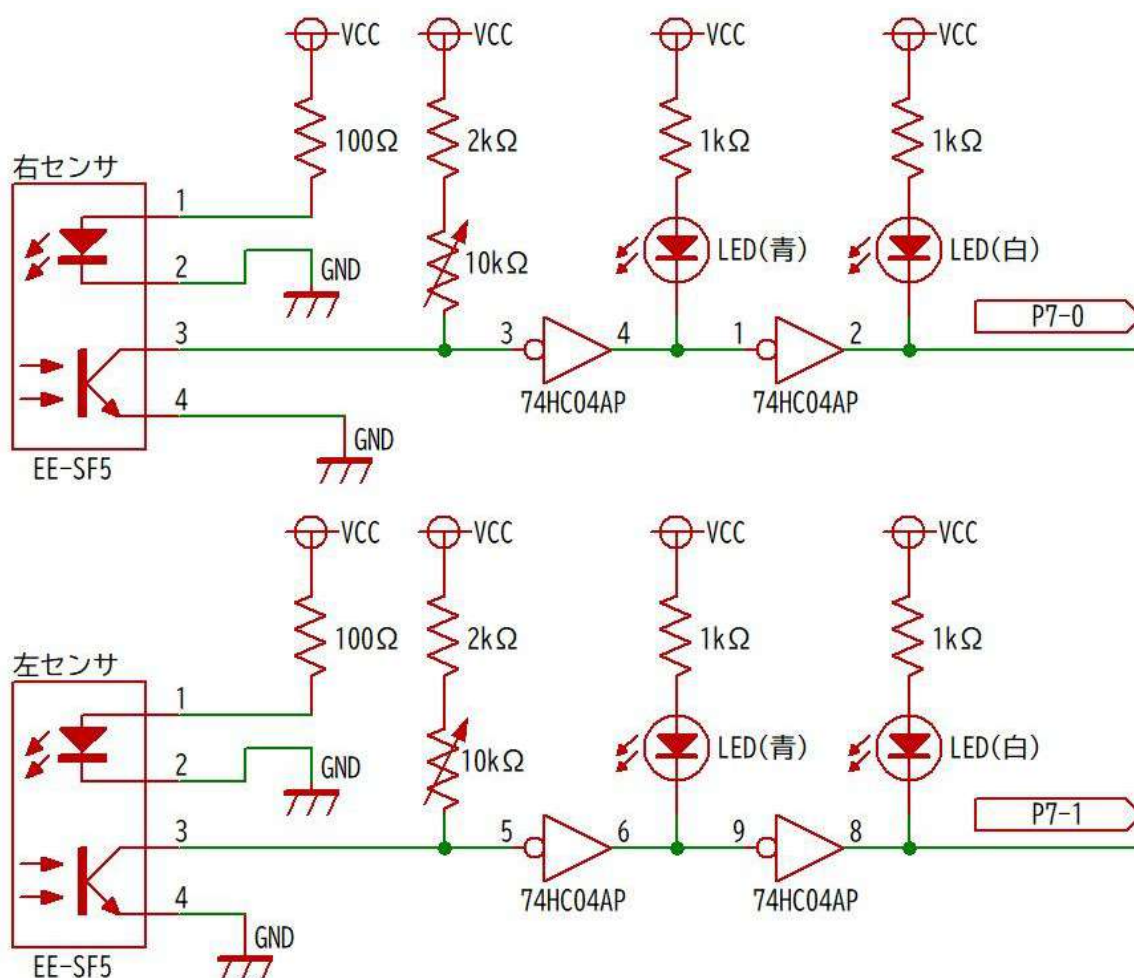
回路図：略図



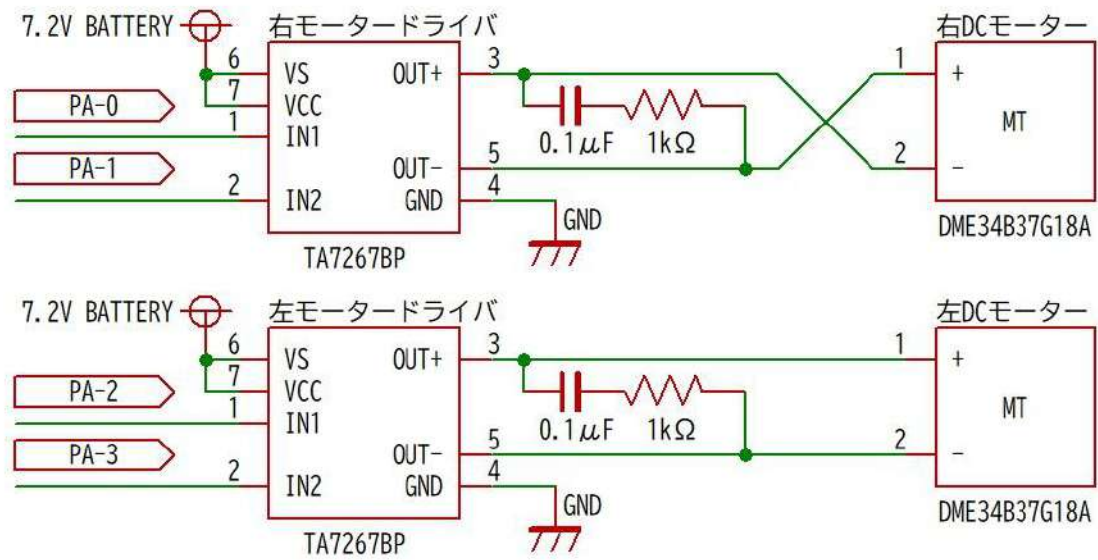
回路図：詳細（電源）



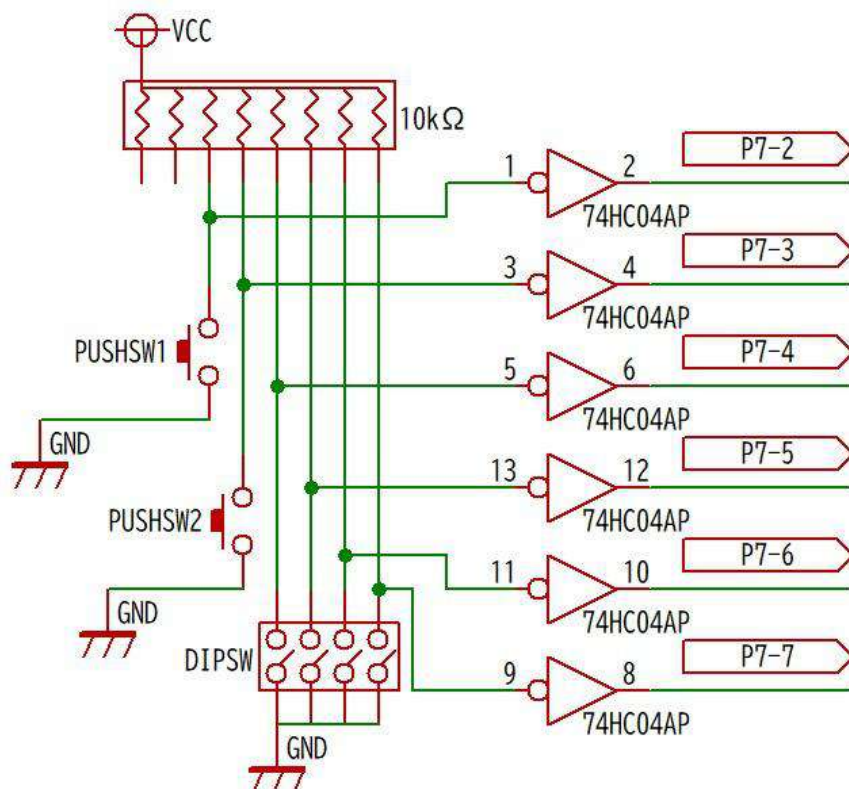
回路図：詳細（入力：センサ）



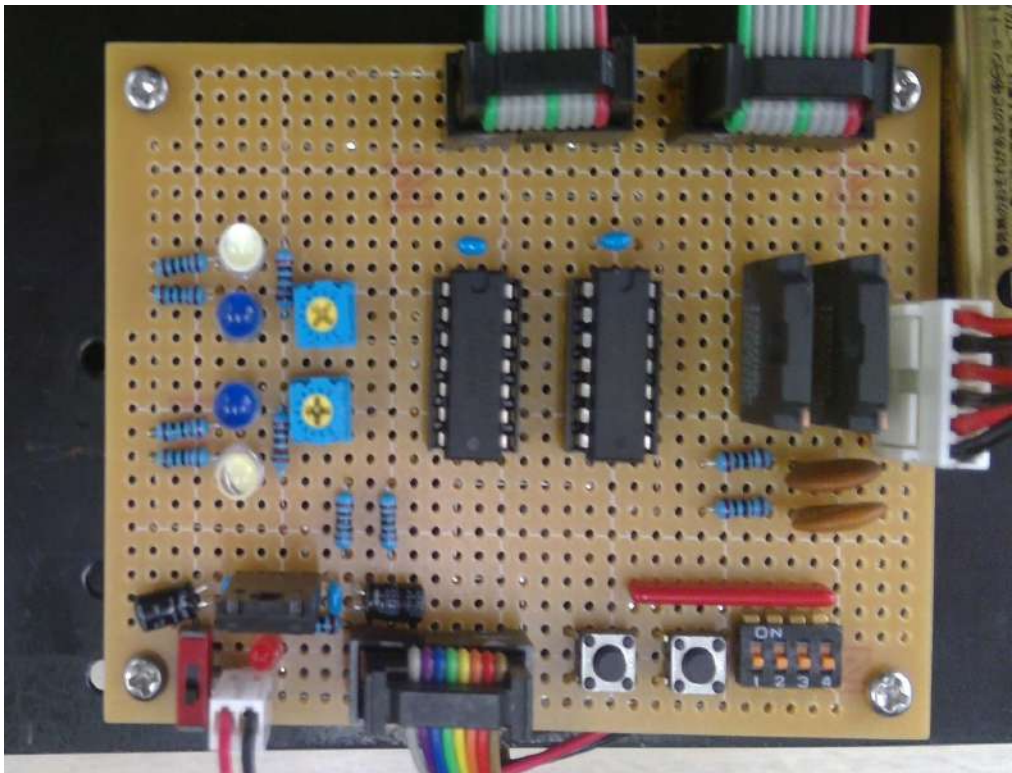
回路図: 詳細(出力)



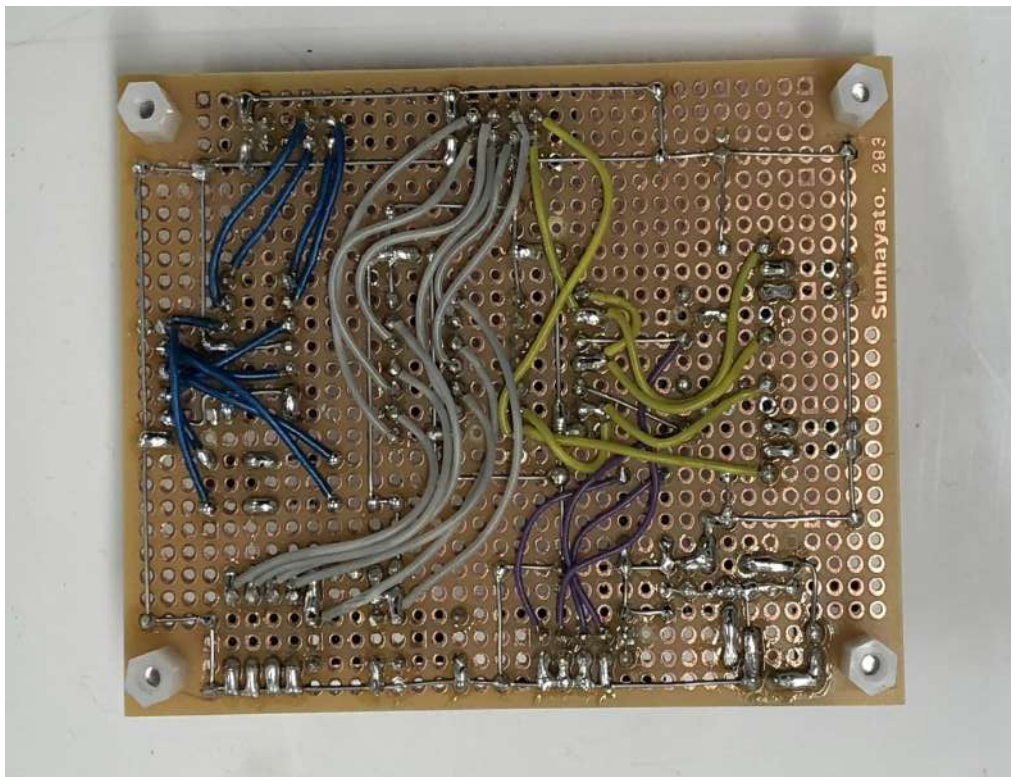
回路図: 詳細(入力: スイッチ)



基板：表



基板：裏



ソフトウェア仕様

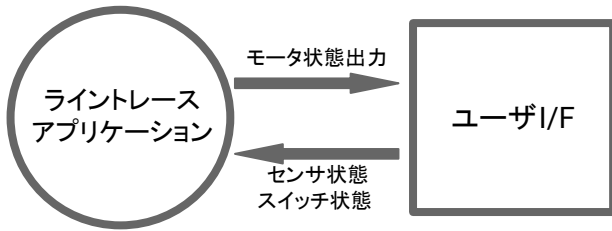
・ 入力	フォト・マイクロセンサ ×2 (右: P7 ₀ 、左: P7 ₁) プッシュSW ×2 (SW1: P7 ₂ 、SW2: P7 ₃) 4極DIPスイッチ (P7 ₄₅₆₇)
・ 出力	DCモーター ×2 (右: PA ₀₁ 、左: PA ₂₃)
・ 動作	1. 電源ONで停止状態で起動 2. プッシュスイッチ1で走行開始する 3. プッシュスイッチ2で走行停止する 4. 走行中左右センサに白検出で直進 5. 走行中左センサ白検出、右センサ黒検出で右折 6. 走行中左センサ黒検出、右センサ白検出で左折 7. 走行中左右センサに黒検出で後退 8. ディップスイッチを切り替えることで、 複数のパターンでの走行が可能。

要求分析: イベントリスト

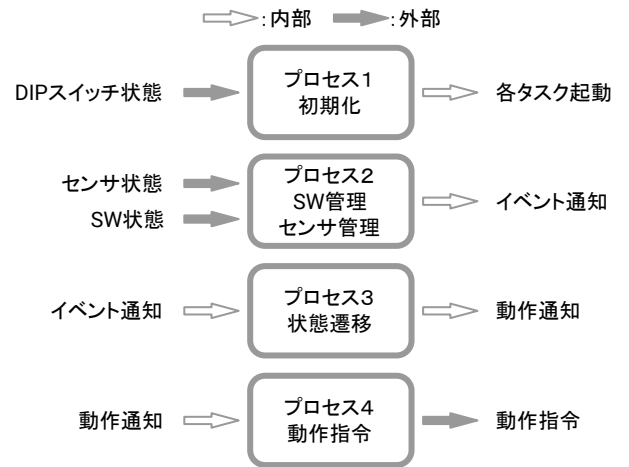
イベント	スティミュラス	アクション	レスポンス
動作開始	電源: ON	初期化	走行停止
走行要求	PUSHSW1: ON	停止→走行開始 走行→走行維持	走行開始 or 走行継続
停止要求	PUSHSW2: ON	走行→走行停止 停止→停止維持	走行停止 or 停止維持
センサ受信	両センサ: 白検出	両モータ: 前回転	直進走行
センサ受信	右センサ: 黒検出 左センサ: 白検出	右モータ: 後回転 左モータ: 前回転	右折走行
センサ受信	右センサ: 白検出 左センサ: 黒検出	右モータ: 前回転 左モータ: 後回転	左折走行
センサ受信	両センサ: 黒検出	両モータ: 後回転	後退走行
動作停止	電源: OFF	—	走行停止
走行モード 変更	DIPスイッチ 切り替え	初期化の際にactする MODE_TASKを変更	—

要求分析

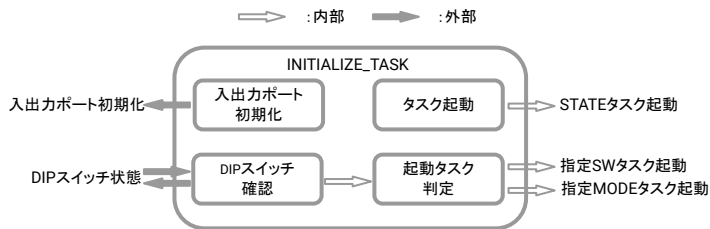
コンテキスト・ダイアグラム



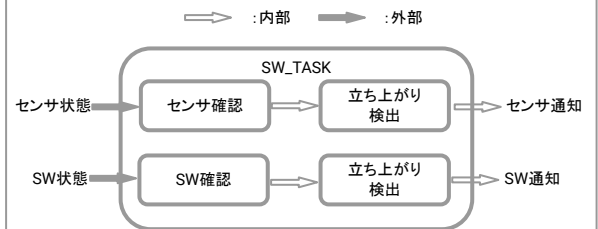
データ・フロー・ダイアグラム



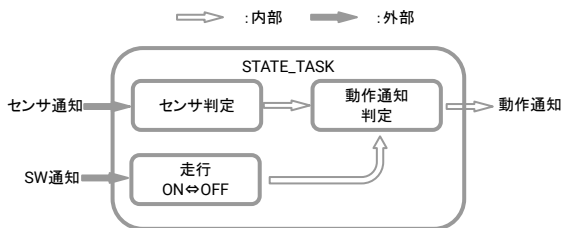
DFDの詳細化: プロセス1



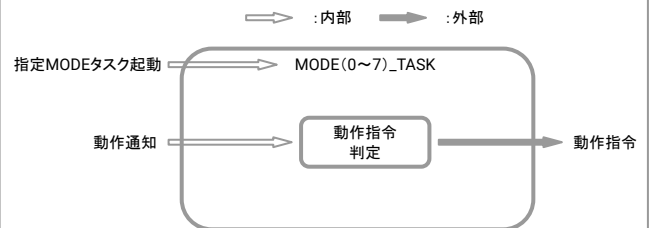
DFDの詳細化: プロセス2



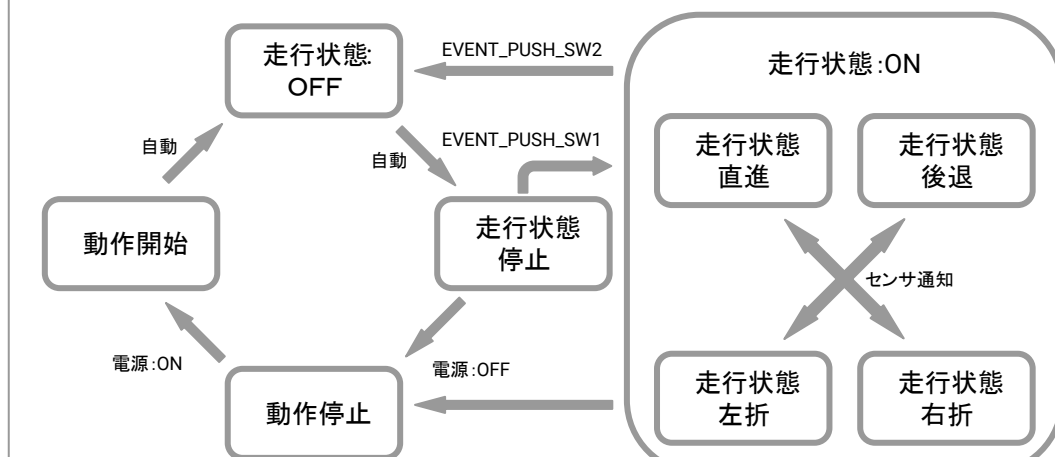
DFDの詳細化: プロセス3



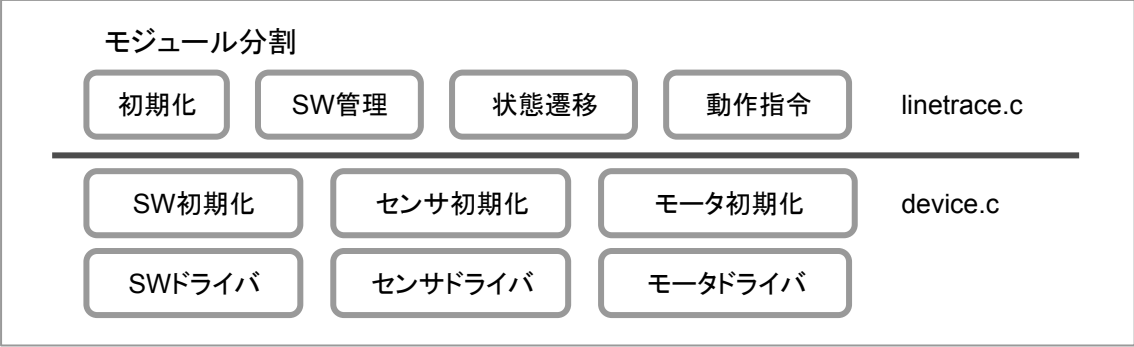
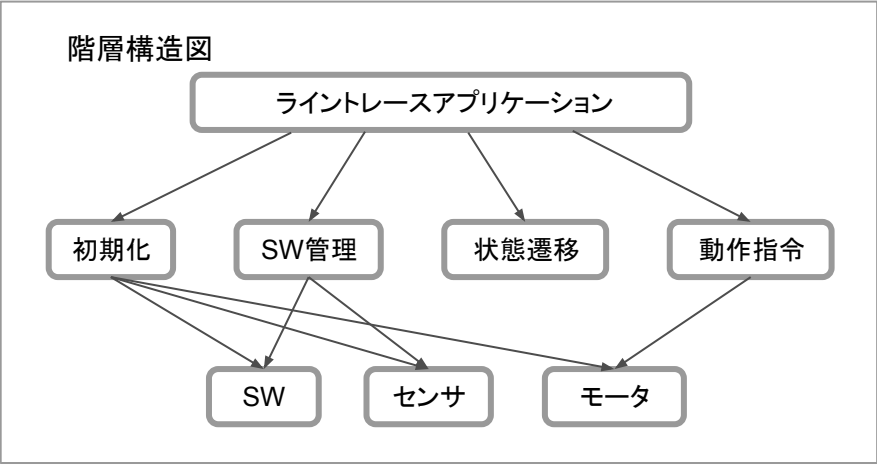
DFDの詳細化: プロセス4



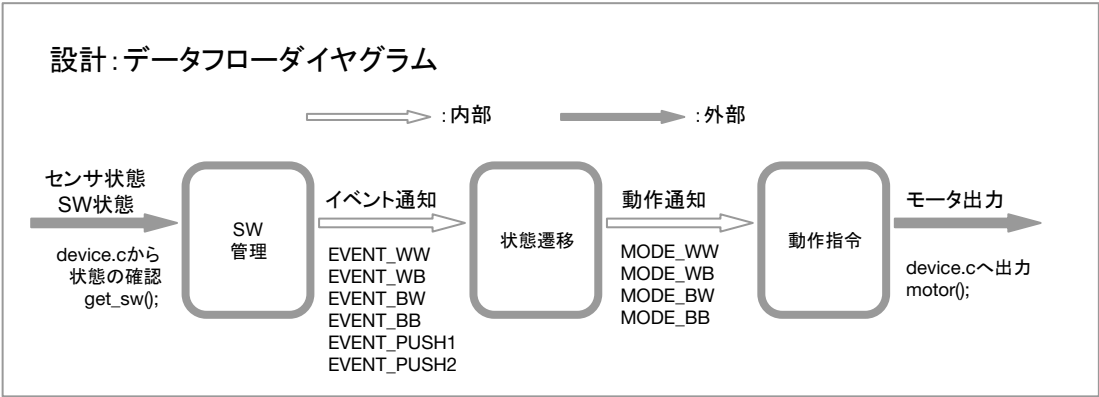
状態のモデル化



設計



モジュール仕様設計				
名称	目的	引数	戻り値	優先度
initialize	入出力の初期化 タスクの起動	—	—	5
sw	センサ確認 各通知	—	—	7
state	状態遷移 各通知	—	—	9
mode	ドライバ書き込み	—	—	11
initialize_sw	センサ入力設定	—	—	—
initialize_motor	モータドライバ出力設定	—	—	—
get_sw	センサ状態読み出し	—	センサ値	—
motor	モータドライバ書き込み	走行値	—	—



関数表

名前	目的	動作
initialize	初期設定	ポート入出力、必要タスクの起動
sw	センサ、SW検出	センサ判定、SW入力判定
slow_sw	センサ、SW検出 (遅い動作用)	センサ判定、SW入力判定、 一定周期での停止動作指令
state	状態遷移	状態判定、状態遷移、動作通知
mode0	動作指令	動作判定、動作通知
mode1	動作指令	動作判定、動作通知
mode2	動作指令	動作判定、動作通知
mode3	動作指令	動作判定、動作通知
mode4	動作指令	動作判定、動作通知
mode5	動作指令	動作判定、動作通知
mode6	動作指令	動作判定、動作通知
mode7	動作指令	動作判定、動作通知

変数表

使用した関数	データ型	名称	目的	動作
initialize	ER	ercd	リターンパラメータ	正常終了orエラー通知
	UB	dip_state	DIPスイッチ状態判定	DIP状態を入れ、=>>4を行う
	UB	sw	入力値代入	入力値を代入
sw	ER	ercd	リターンパラメータ	正常終了orエラー通知
	UB	sw	入力値代入	入力値を代入
	static UB	state_ww	立ち上がり検出	0x01の時のみイベント通知
		state_wb		
		state_bw		0x03の時、イベント通知
		state_bb		
		state_sw1		
		state_sw2		
slow_sw	ER	ercd	リターンパラメータ	正常終了orエラー通知
	UB	sw	入力値代入	入力値を代入
state	ER	ercd	リターンパラメータ	正常終了orエラー通知
	FLGPTN	flgptn	イベントフラグ	状態通知の受け取り
	static UB	state	状態遷移	プッシュスイッチによる 走行⇄停止の切り替え
mode	ER	ercd	リターンパラメータ	正常終了orエラー通知
	FLGPTN	flgptn	イベントフラグ	動作通知の受け取り

.cfgファイル

```
#define _MACRO_ONLY
#include "../linetrace.h"

INCLUDE("../linetrace.h");

CRE_TSK(INITIALIZE_TASK, {TA_HLNG|TA_ACT,0,initialize,INITIALIZE_PRIORITY,STACK_SIZE,NULL});
CRE_TSK(SW_TASK, { TA_HLNG ,0,sw,SW_PRIORITY,STACK_SIZE,NULL});
CRE_TSK(SLOW_SW_TASK, { TA_HLNG ,0,slow_sw,SW_PRIORITY,STACK_SIZE,NULL});
CRE_TSK(STATE_TASK, { TA_HLNG ,0,state,STATE_PRIORITY,STACK_SIZE,NULL});
CRE_TSK(MODE0_TASK, { TA_HLNG ,0,mode0,MODE_PRIORITY,STACK_SIZE,NULL});
CRE_TSK(MODE1_TASK, { TA_HLNG ,0,mode1,MODE_PRIORITY,STACK_SIZE,NULL});
CRE_TSK(MODE2_TASK, { TA_HLNG ,0,mode2,MODE_PRIORITY,STACK_SIZE,NULL});
CRE_TSK(MODE3_TASK, { TA_HLNG ,0,mode3,MODE_PRIORITY,STACK_SIZE,NULL});
CRE_TSK(MODE4_TASK, { TA_HLNG ,0,mode4,MODE_PRIORITY,STACK_SIZE,NULL});
CRE_TSK(MODE5_TASK, { TA_HLNG ,0,mode5,MODE_PRIORITY,STACK_SIZE,NULL});
CRE_TSK(MODE6_TASK, { TA_HLNG ,0,mode6,MODE_PRIORITY,STACK_SIZE,NULL});
CRE_TSK(MODE7_TASK, { TA_HLNG ,0,mode7,MODE_PRIORITY,STACK_SIZE,NULL});

CRE_FLG(SW_FLG, {TA_TFIFO | TA_WSGL | TA_CLR,0});
CRE_FLG(MODE_FLG, {TA_TFIFO | TA_WSGL | TA_CLR,0});

#ifdef CPUEXC1
DEF_EXC(CPUEXC1, {TA_HLNG,cpuexc_handler});
#endif

#include "../jss/systask/timer.cfg"
#include "../jss/systask/serial.cfg"
#include "../jss/systask/logtask.cfg"
```

.hファイル

```
#define INITIALIZE_PRIORITY      5
#define SW_PRIORITY              7
#define STATE_PRIORITY           9
#define MODE_PRIORITY            11

#ifndef _MACRO_ONLY
extern void initialize(VP_INT exinf);
extern void sw(VP_INT exinf);
extern void slow_sw(VP_INT exinf);
extern void state(VP_INT exinf);
extern void mode0(VP_INT exinf);
extern void mode1(VP_INT exinf);
extern void mode2(VP_INT exinf);
extern void mode3(VP_INT exinf);
extern void mode4(VP_INT exinf);
extern void mode5(VP_INT exinf);
extern void mode6(VP_INT exinf);
extern void mode7(VP_INT exinf);

#endif
```


device.cファイル

```
#include "s_services.h"
#include "h8_3069f.h"

void initialize_sw(void);
void initialize_port(void);
UB get_sw(void);
void motor(volatile UB data);

void initialize_sw(void){
}

void initialize_port(void){
    sil_wrb_mem((VP)H8P4DDR, 0xff);           //Port4 : Output
    sil_wrb_mem((VP)H8PADDR, 0xff);          //PortA : Output

    sil_wrb_mem((VP)H8P4DR, 0x00);
    sil_wrb_mem((VP)H8PADR, 0x00);
}

UB get_sw(void){
    return(sil_reb_mem((VP)H8P7DR));
}

void motor(volatile UB data){
    if(data & 0x08)    sil_wrb_mem((VP)H8PADR, sil_reb_mem((VP)H8PADR)|0x08);
    else              sil_wrb_mem((VP)H8PADR, sil_reb_mem((VP)H8PADR)&0xf7);
    if(data & 0x04)    sil_wrb_mem((VP)H8PADR, sil_reb_mem((VP)H8PADR)|0x04);
    else              sil_wrb_mem((VP)H8PADR, sil_reb_mem((VP)H8PADR)&0xfb);
    if(data & 0x02)    sil_wrb_mem((VP)H8PADR, sil_reb_mem((VP)H8PADR)|0x02);
    else              sil_wrb_mem((VP)H8PADR, sil_reb_mem((VP)H8PADR)&0xfd);
    if(data & 0x01)    sil_wrb_mem((VP)H8PADR, sil_reb_mem((VP)H8PADR)|0x01);
    else              sil_wrb_mem((VP)H8PADR, sil_reb_mem((VP)H8PADR)&0xfe);
}
```

.cファイル

```
#include <t_services.h>
#include "kernel_id.h"
#include "linetrace.h"
#include "sys_config.h"

/*sw*/
#define SW_SENSOR      0x03
#define SW_PUSH1       0x04
#define SW_PUSH2       0x08
#define SW_DIP1        0x10
#define SW_DIP2        0x20
#define SW_DIP3        0x40
#define SW_DIP4        0x80
#define SW_DIP123      0x70

/*sw state*/
#define WW             0x00
#define WB             0x01
#define BW             0x02
#define BB             0x03

/*sw flg*/
#define EVENT_WW       0x01
#define EVENT_WB       0x02
#define EVENT_BW       0x04
#define EVENT_BB       0x08
#define EVENT_PUSH1    0x10
#define EVENT_PUSH2    0x20

/*state*/
#define POWER_OFF      0x00
#define POWER_ON       0x01

/*mode flg*/
#define MODE_RUN        0x01
#define MODE_RIGHT      0x02
#define MODE_LEFT       0x04
#define MODE_BLACK      0x08
#define MODE_STOP       0x10

/*mode*/
#define LS_RS          0x00    //左S・右S
#define LS_RB          0x01    //左S・右B
#define LS_RD          0x02    //左S・右D
#define LS_RN          0x03    //左S・右N
#define LB_RS          0x04    //左B・右S
#define LB_RB          0x05    //左B・右B
#define LB_RD          0x06    //左B・右D
#define LB_RN          0x07    //左B・右N
#define LD_RS          0x08    //左D・右S
#define LD_RB          0x09    //左D・右B
#define LD_RD          0x0A    //左D・右D
#define LD_RN          0x0B    //左D・右N
#define LN_RS          0x0C    //左N・右S
#define LN_RB          0x0D    //左N・右B
#define LN_RD          0x0E    //左N・右D
#define LN_RN          0x0F    //左N・右N
```

.cファイル

```
void initialize(VP_INT exinf)    //初期設定
{
    ER        ercd;
    UB        dip_state;
    UB        sw;

    initialize_sw();
    initialize_port();

    sw = get_sw();
    dip_state = (sw & SW_DIP123);
    dip_state >>=4;

    switch(dip_state){
    case 0x00:    ercd = act_tsk(MODE0_TASK);    break;//内輪逆転、外輪正転
    case 0x01:    ercd = act_tsk(MODE1_TASK);    break;//内輪逆転、外輪停止（三角）
    case 0x02:    ercd = act_tsk(MODE2_TASK);    break;//内輪逆転、外輪フリー（三角）
    // case 0x03:    ercd = act_tsk(MODE3_TASK);    break;//
    // case 0x04:    ercd = act_tsk(MODE4_TASK);    break;//
    // case 0x05:    ercd = act_tsk(MODE5_TASK);    break;//
    // case 0x06:    ercd = act_tsk(MODE6_TASK);    break;//
    case 0x07:    ercd = act_tsk(MODE7_TASK);    break;//
    default:      ercd = act_tsk(MODE0_TASK);    break;//内輪逆転、外輪正転
    }

    if(sw & SW_DIP4)    ercd = act_tsk(SLOW_SW_TASK);//スロー走行モード
    else                ercd = act_tsk(SW_TASK);
    ercd = act_tsk(STATE_TASK);

    ext_tsk();
}

void sw(VP_INT exinf){
    ER    ercd;
    UB    sw;
    static UB state_ww = 0;
    static UB state_wb = 0;
    static UB state_bw = 0;
    static UB state_bb = 0;
    static UB state_sw1 = 0;
    static UB state_sw2 = 0;

    for(;;){
        sw = get_sw();
        state_ww <=<= 1;
        state_wb <=<= 1;
        state_bw <=<= 1;
        state_bb <=<= 1;
        state_sw1 <=<= 1;
        state_sw2 <=<= 1;

        if((sw & SW_SENSOR)==WW) state_ww |= 0x01;
        else state_ww &= 0xfe;
        if((sw & SW_SENSOR)==WB) state_wb |= 0x01;
        else state_wb &= 0xfe;
        if((sw & SW_SENSOR)==BW) state_bw |= 0x01;
        else state_bw &= 0xfe;
        if((sw & SW_SENSOR)==BB) state_bb |= 0x01;
        else state_bb &= 0xfe;
        if(sw & SW_PUSH1) state_sw1 |= 0x01;
        else state_sw1 &= 0xfe;
        if(sw & SW_PUSH2) state_sw2 |= 0x01;
        else state_sw2 &= 0xfe;

        if(state_ww == 0x01)    ercd = set_flg(SW_FLG,EVENT_WW);
        else if(state_wb == 0x01) ercd = set_flg(SW_FLG,EVENT_WB);
        else if(state_bw == 0x01) ercd = set_flg(SW_FLG,EVENT_BW);
        else if(state_bb == 0x01) ercd = set_flg(SW_FLG,EVENT_BB);
        if(state_sw1 == 0x03)    ercd = set_flg(SW_FLG,EVENT_PUSH1);
        if(state_sw2 == 0x03)    ercd = set_flg(SW_FLG,EVENT_PUSH2);

        ercd = dly_tsk(10);
    }
}
```

.cファイル

```
void slow_sw(VP_INT exinf){
    ER   ercd;
    UB   sw;

    for(;;){
        sw = get_sw();

        if((sw & SW_SENSOR)==WW)   ercd = set_flg(SW_FLG,EVENT_WW);
        if((sw & SW_SENSOR)==WB)   ercd = set_flg(SW_FLG,EVENT_WB);
        if((sw & SW_SENSOR)==BW)   ercd = set_flg(SW_FLG,EVENT_BW);
        if((sw & SW_SENSOR)==BB)   ercd = set_flg(SW_FLG,EVENT_BB);
        if(sw & SW_PUSH1)          ercd = set_flg(SW_FLG,EVENT_PUSH1);
        if(sw & SW_PUSH2)          ercd = set_flg(SW_FLG,EVENT_PUSH2);

        ercd = dly_tsk(4);
        ercd = set_flg(MODE_FLG,MODE_STOP);
        ercd = dly_tsk(6);
    }
}

void state(VP_INT exinf){
    ER   ercd;
    FLGPTN flgptn;
    static UB state = POWER_OFF;

    for(;;){
        ercd = wai_flg(SW_FLG,EVENT_WW | EVENT_WB | EVENT_BW | EVENT_BB | EVENT_PUSH1 |
EVENT_PUSH2,TWF_ORW,&flgptn);

        if((state == POWER_ON)&&(flgptn == EVENT_WW))
            ercd = set_flg(MODE_FLG,MODE_RUN);
        else if((state == POWER_ON)&&(flgptn == EVENT_WB))
            ercd = set_flg(MODE_FLG,MODE_RIGHT);
        else if((state == POWER_ON)&&(flgptn == EVENT_BW))
            ercd = set_flg(MODE_FLG,MODE_LEFT);
        else if((state == POWER_ON)&&(flgptn == EVENT_BB))
            ercd = set_flg(MODE_FLG,MODE_BLACK);
        else if((state == POWER_OFF)&&(flgptn == EVENT_PUSH1)){
            state = POWER_ON;
            ercd = set_flg(MODE_FLG,MODE_RUN);
        }
        else if((state == POWER_ON)&&(flgptn == EVENT_PUSH2)){
            state = POWER_OFF;
            ercd = set_flg(MODE_FLG,MODE_STOP);
        }
    }
}

void mode0(VP_INT exinf){///内輪逆転、外輪正転(最速)

    ER   ercd;
    FLGPTN flgptn;

    for(;;){
        ercd = wai_flg(MODE_FLG,MODE_RUN | MODE_RIGHT | MODE_LEFT | MODE_BLACK |
MODE_STOP,TWF_ORW,&flgptn);

        switch(flgptn){
            case MODE_RUN:      motor(LD_RD);
                                break;
            case MODE_RIGHT:    motor(LD_RB);
                                break;
            case MODE_LEFT:     motor(LB_RD);
                                break;
            case MODE_BLACK:    motor(LB_RB);
                                break;
            case MODE_STOP:     motor(LS_RS);
                                break;
            default:            break;
        }
    }
}
```

工夫点、改良点

ハード

- ・プッシュスイッチの増設
- ・DIPスイッチの増設
- ・白青LEDなど、視覚的に分かりやすい造り
- ・モータードライバとモータの間にコンデンサを設け、安定化を図った
- ・右モータードライバとモータ間の配線を逆に繋げ、左右のモータに流す信号を同じにした
- ・デジタルICにコンデンサを付けた

ソフト

- ・初期化の際にDIPスイッチを判定することによって、actするMODE_TASKを切り替え、複数のパターンで走行が可能
- ・プッシュスイッチで走行⇄停止を操作でき、筐体の操作が容易
- ・DIPスイッチの操作で遅い走行が可能