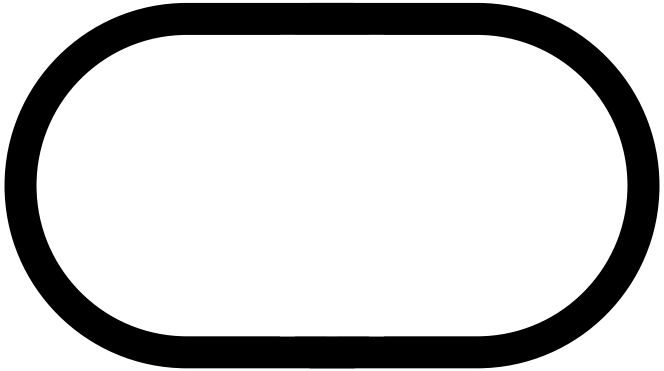
ライントレース自走ロボ製作

目次

- 仕様
- 作業工程
- ハードウェア製作
- ソフトウェア製作
- ソースコード
- 反省点、改善点

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　　下図のコースを、どちらの向きでも3周以上安定して走る。



作業工程

初期予定スケジュール

スケジュール表		12 月																
項目	内容	2	3	9	10	11	12	13	16	17	18	19	20	23	24	25	26	
プロジェクトの開始	- 作業打ち合わせ																	走行会・発表会
	- 製作仕様書作成																	
ハードウェアの設計、製作	- 回路図（たたき台）作成																	
	- テスト回路作成																	
	- 必要資材選定																	
	- 回路図（清書）作成																	
	- 基板製作（はんだ付け）																	
	- 単体動作確認テスト																	
ソフトウェア設計、製作	- 要求分析、設計作成																	
	- プログラム作成（代表）																	
	- 総合動作テスト																	
報告書作成	- スライド作成													資料制作				
	- 報告書作成（各自）													報告書制作				

実際のスケジュール

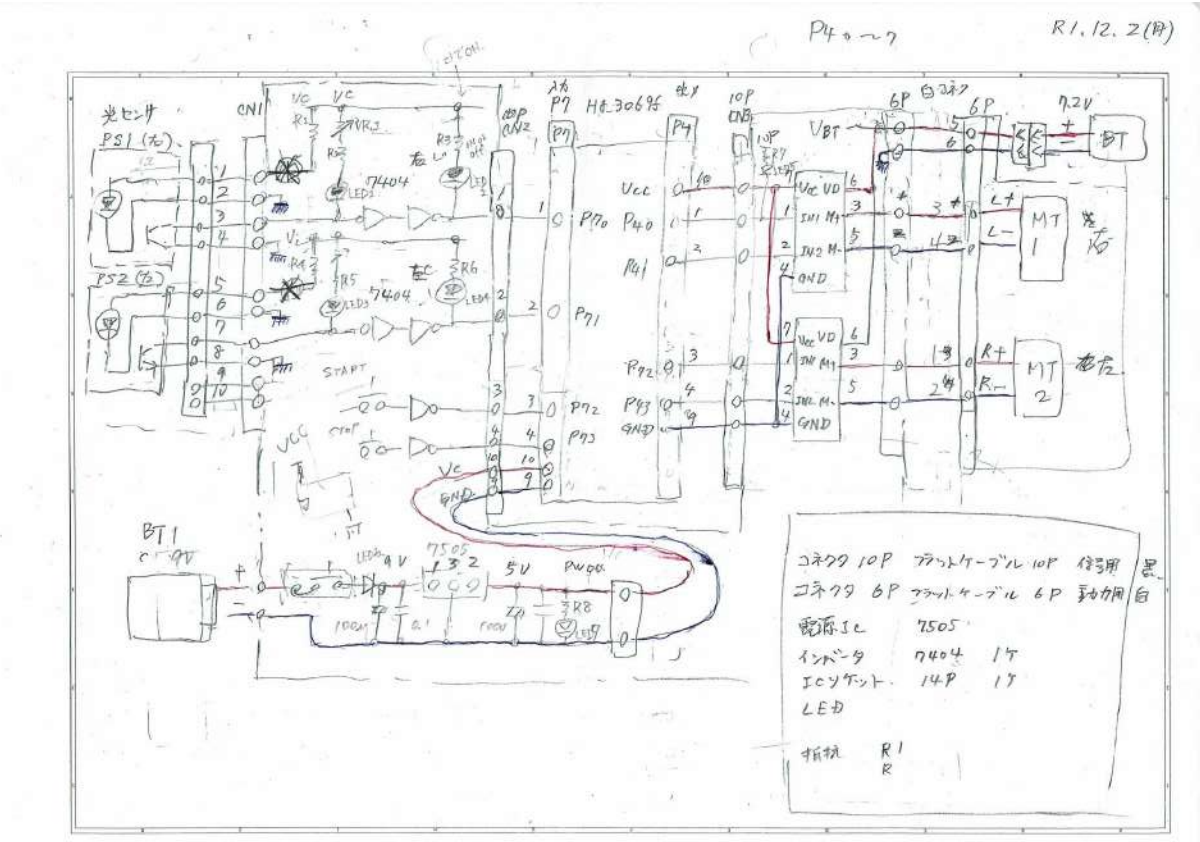
スケジュール表		12 月																
項目	内容	2	3	9	10	11	12	13	16	17	18	19	20	23	24	25	26	
プロジェクトの開始	- 作業打ち合わせ																	走行会・発表会
	- 製作仕様書作成																	
ハードウェアの設計、製作	- 回路図（たたき台）作成	手書き																
	- テスト回路作成		ブレッドボードで動作などの確認															
	- 必要資材選定																	
	- 回路図（清書）作成			EXCEL, BSCHDV														
	- 基板製作（はんだ付け）						はんだ付け											
	- 単体動作確認テスト																	
ソフトウェア設計、製作	- 要求分析、設計作成										DFD等の作成							
	- プログラム作成（代表）																	
	- 総合動作テスト																	
報告書作成	- スライド作成										googleスライドでの資料制作							
	- 報告書作成（各自）																	

ハードウェア製作

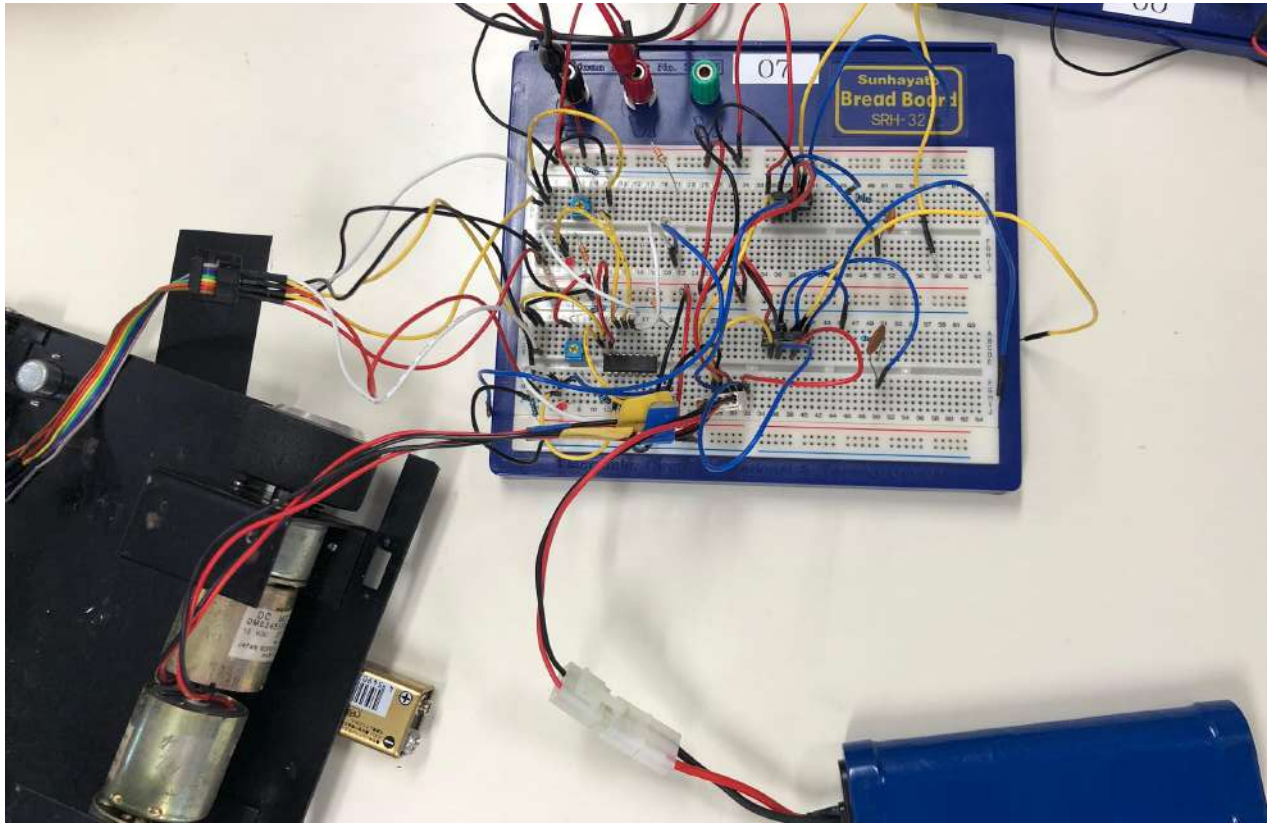
ハードウェア製作：目次

- 試案回路図（手書き）
- テスト回路
- 使用機器
- 使用部品
- 回路図
- 基板製作（はんだ付け）
- 基板
- 動作テスト

試案回路図(手書き)


R1.12.2
光センサ
PS1
PS2
CN1
CN2
P7
P4
CN3
6P
7.2V
BT
7404
START
STOP
BT1
9V
7505
5V
MT 1
MT 2
コネクタ 10P フラットケーブル 10P 信号用 黒
コネクタ 6P フラットケーブル 6P 動力用 白
電源IC 7505
インバータ 7404 1ケ
ICソケット 14P 1ケ
LED
抵抗 R1
R

テスト回路



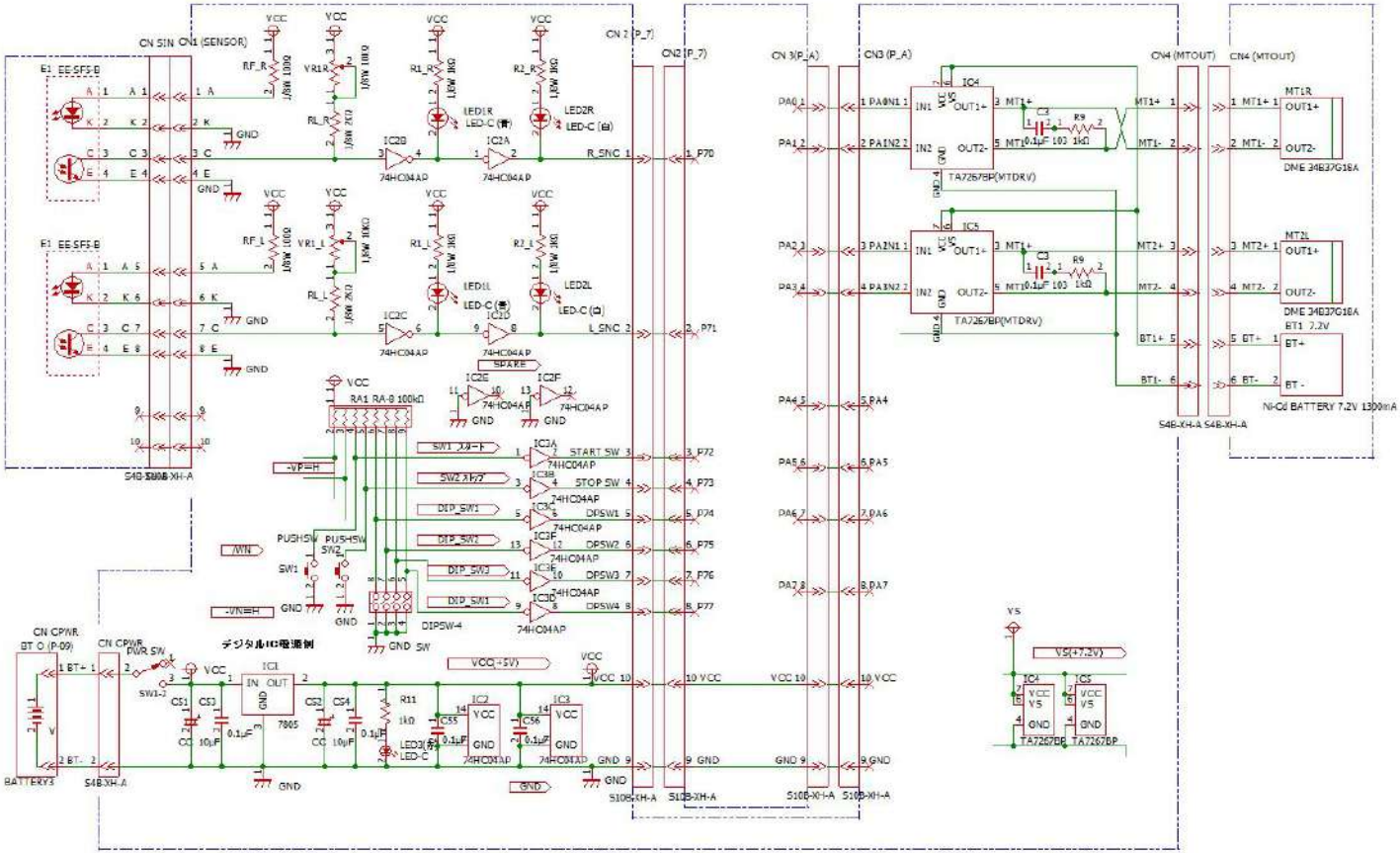
使用機器

・自走ロボット筐体
・センサ（OMRON EE-SF5×2）
・モータ（JAPAN SARVO　DME34B37G18A×2)
・充電式ニッカド電池（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）

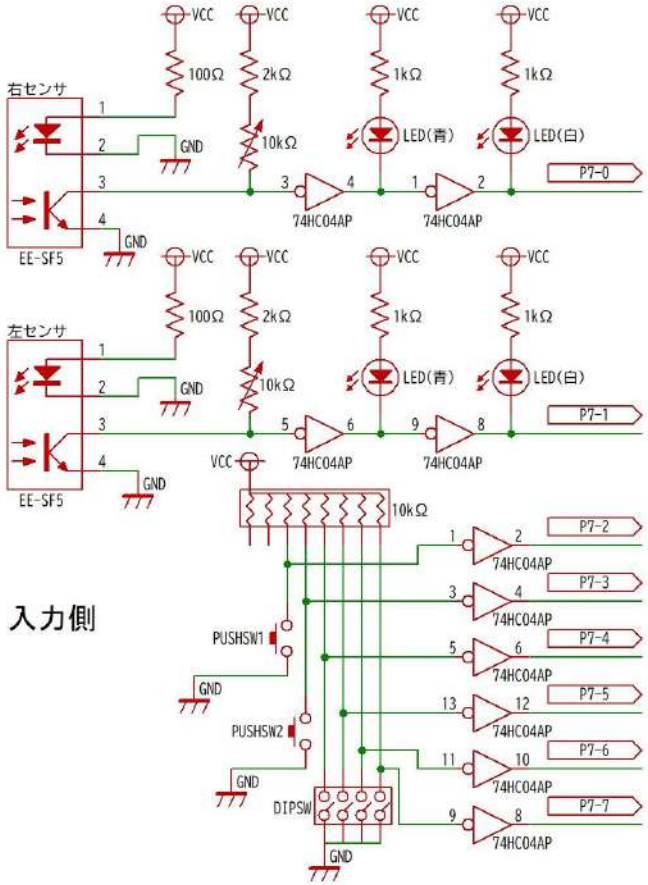
使用部品

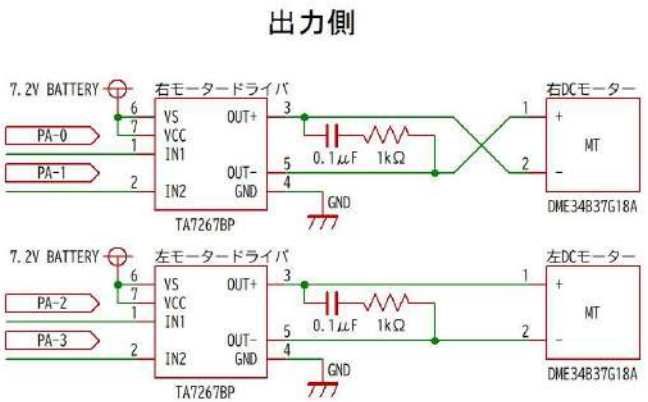
・ユニバーサル基盤×1
・デジタルIC74HC04×2
・LED 赤×1
・LED 青×2
・LED 白×2
・抵抗(1kΩ)×7（LED用、モータードライバ安定化用）
・抵抗(2kΩ)×2（センサ受光部用）
・抵抗(100Ω)×2（センサ発光部用）
・可変抵抗器（10kΩ）×2（センサ受光感度調整用）
・集合抵抗(10kΩ)（スイッチ用）
・3端子スライドスイッチ×1
・プッシュスイッチ×2
・4極ディップスイッチ×1
・電解コンデンサ(10μF)×2
・積層セラミックコンデンサ(0.1μF)×4
・セラミックコンデンサ(0.1μF)×2
・10pinコネクタ×3（センサ用、portA,port7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

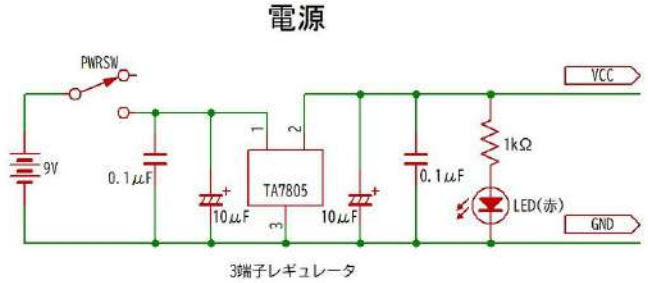
回路図:清書


CN 5IN CN1 (SENSOR)
CN 7 (P_7)
CN2 (P_7)
CN 3(P_A)
CN3 (P_A)
CN4 (MTOUT)
CN4 (MTOUT)
CN CPWR
BT O (P-09)
PWR SW
BATTERY3
S4B-XH-A
S10B-XH-A
74HC04AP
TA7267BP(MTDRV)
DME 34B37G18A
NI-Cd BATTERY 7.2V 1300mA
デジタルIC電源例
DIPSW-4
PUSHSW
SPARE
VCC
GND
VS(+7.2V)

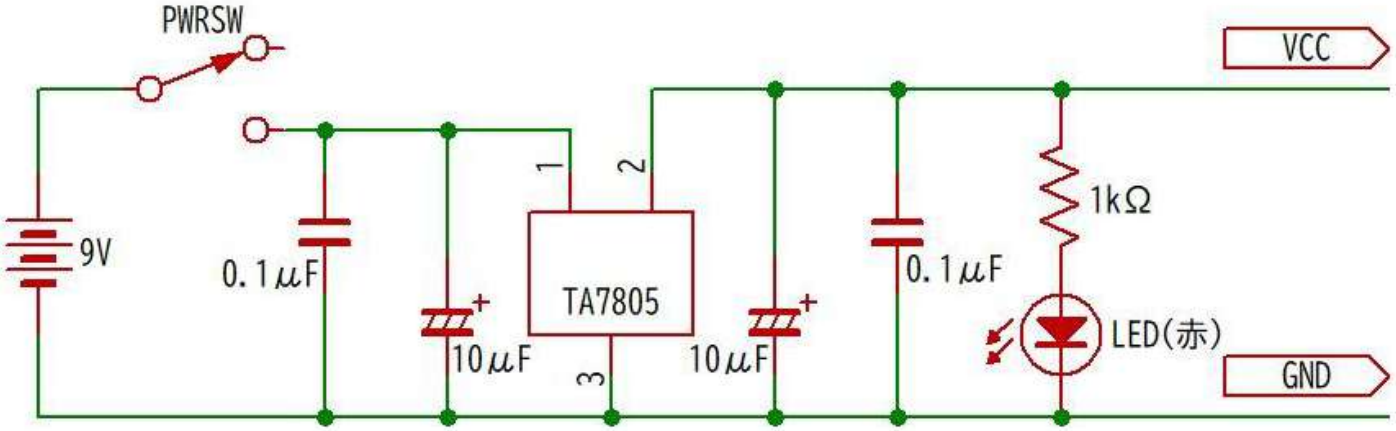
回路図:全体


右センサ
左センサ
EE-SF5
VCC
GND
100Ω
2kΩ
10kΩ
1kΩ
LED(青)
LED(白)
74HC04AP
P7-0
P7-1
P7-2
P7-3
P7-4
P7-5
P7-6
P7-7
入力側
PUSHSW1
PUSHSW2
DIPSW


出力側
7.2V BATTERY
右モータードライバ
左モータードライバ
TA7267BP
VS
VCC
IN1
IN2
OUT+
OUT-
GND
PA-0
PA-1
PA-2
PA-3
0.1μF
1kΩ
右DCモーター
左DCモーター
MT
DME34B37G18A

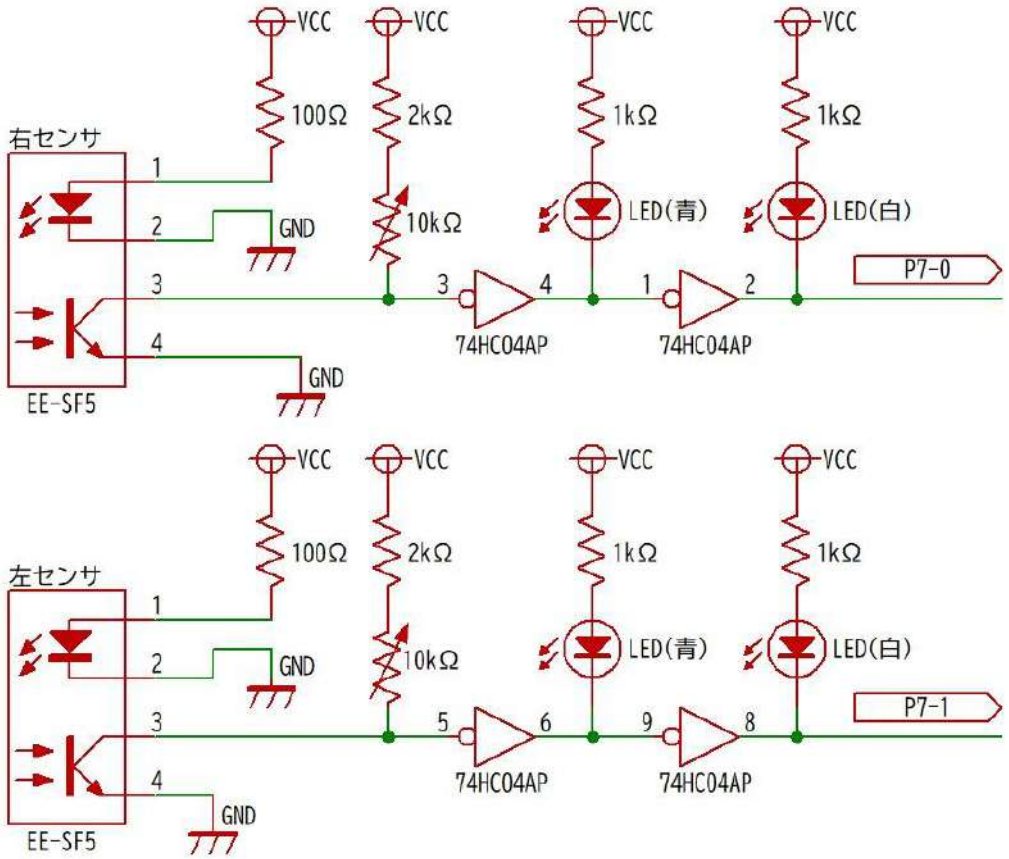

電源
PWRSW
9V
0.1μF
TA7805
10μF
1kΩ
LED(赤)
VCC
GND
3端子レギュレータ

回路図:電源

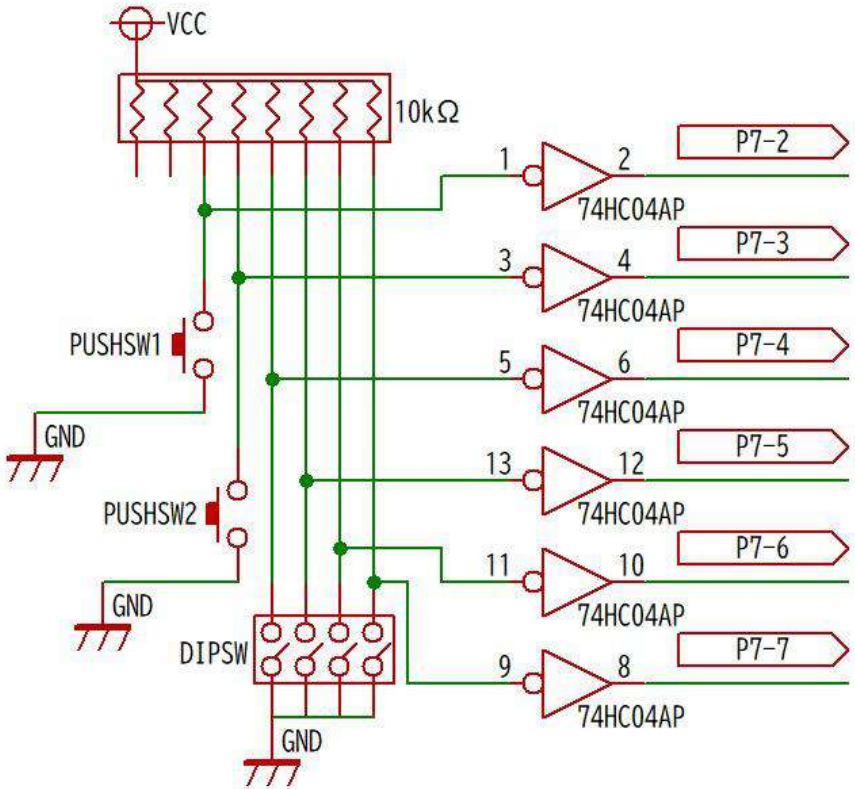


3端子レギュレータ

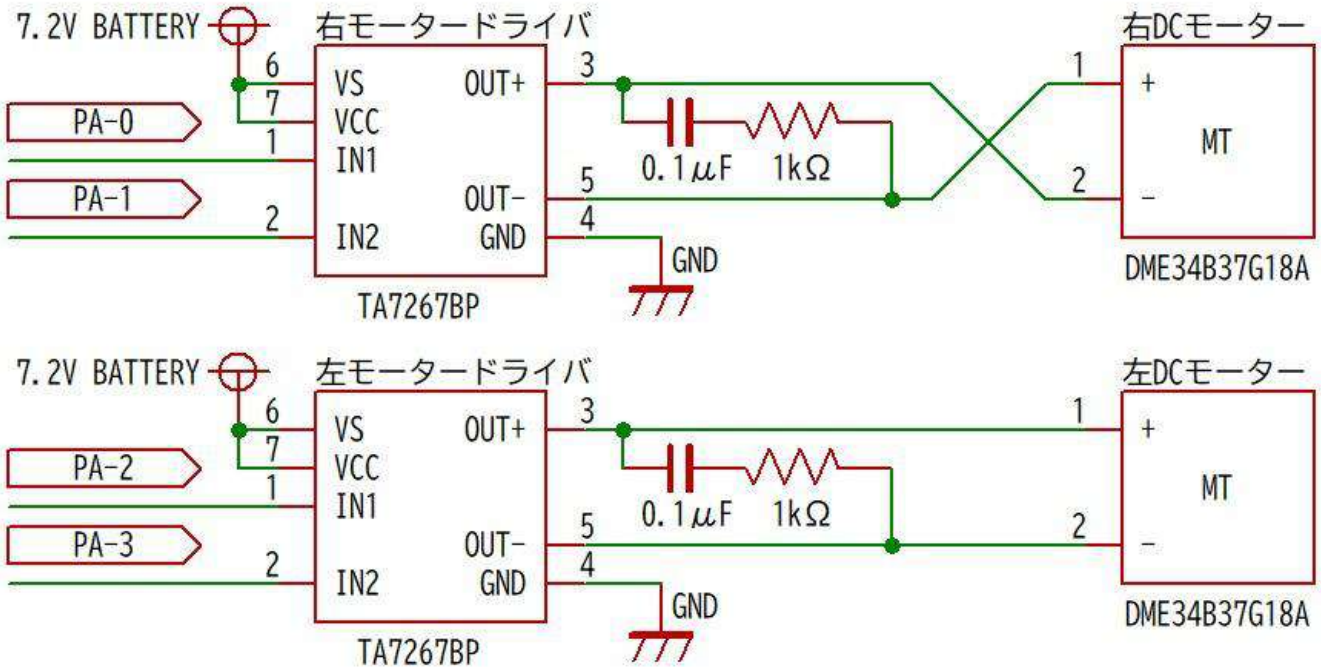
回路図:センサ


VCC
VCC
VCC
VCC
100Ω
2kΩ
1kΩ
1kΩ
右センサ
1
2
3
4
GND
10kΩ
LED(青)
LED(白)
P7-0
3
4
1
2
74HC04AP
74HC04AP
GND
EE-SF5
VCC
VCC
VCC
VCC
100Ω
2kΩ
1kΩ
1kΩ
左センサ
1
2
3
4
GND
10kΩ
LED(青)
LED(白)
P7-1
5
6
9
8
74HC04AP
74HC04AP
GND
EE-SF5

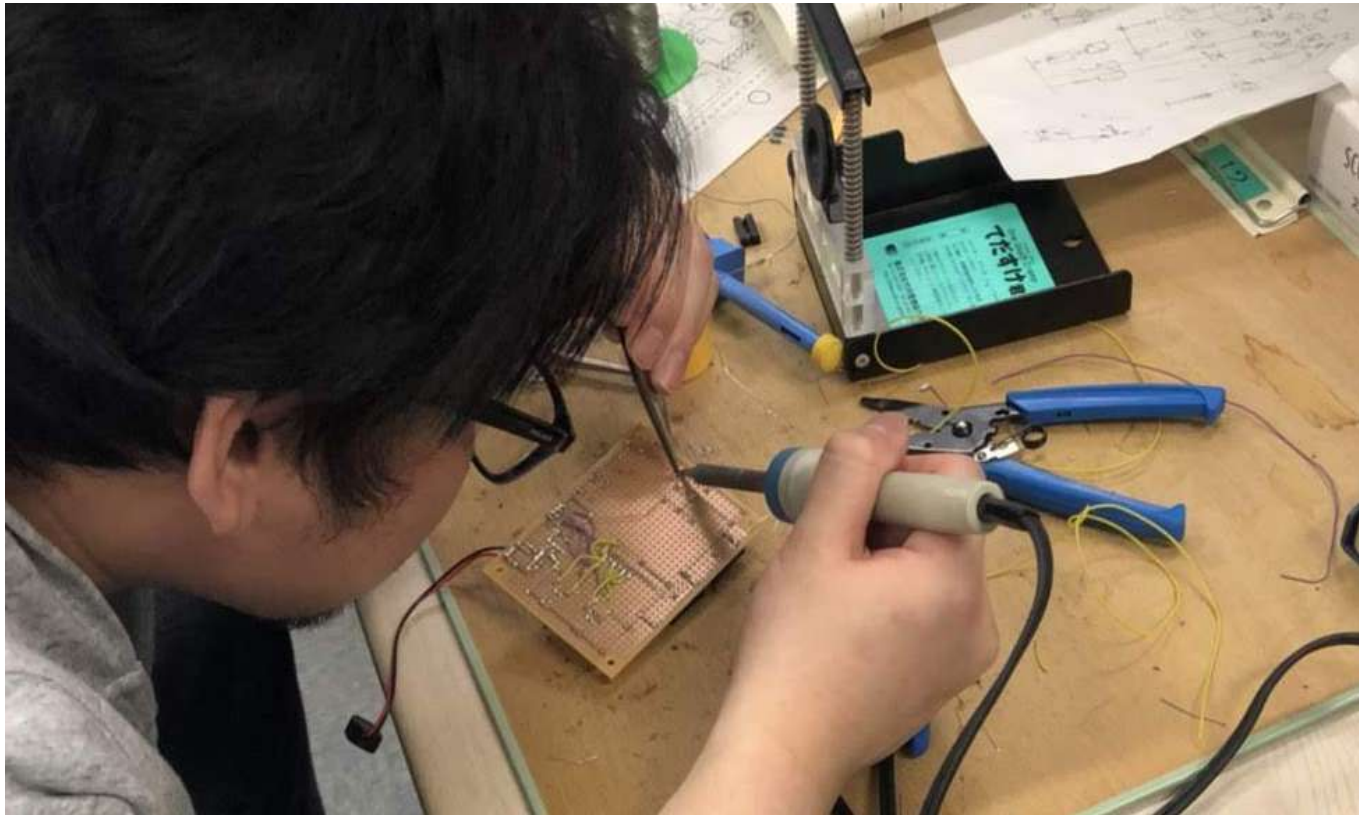
回路図:スイッチ


VCC
10kΩ
PUSHSW1
PUSHSW2
DIPSW
GND
GND
GND
1
2
3
4
5
6
13
12
11
10
9
8
74HC04AP
74HC04AP
74HC04AP
74HC04AP
74HC04AP
74HC04AP
P7-2
P7-3
P7-4
P7-5
P7-6
P7-7

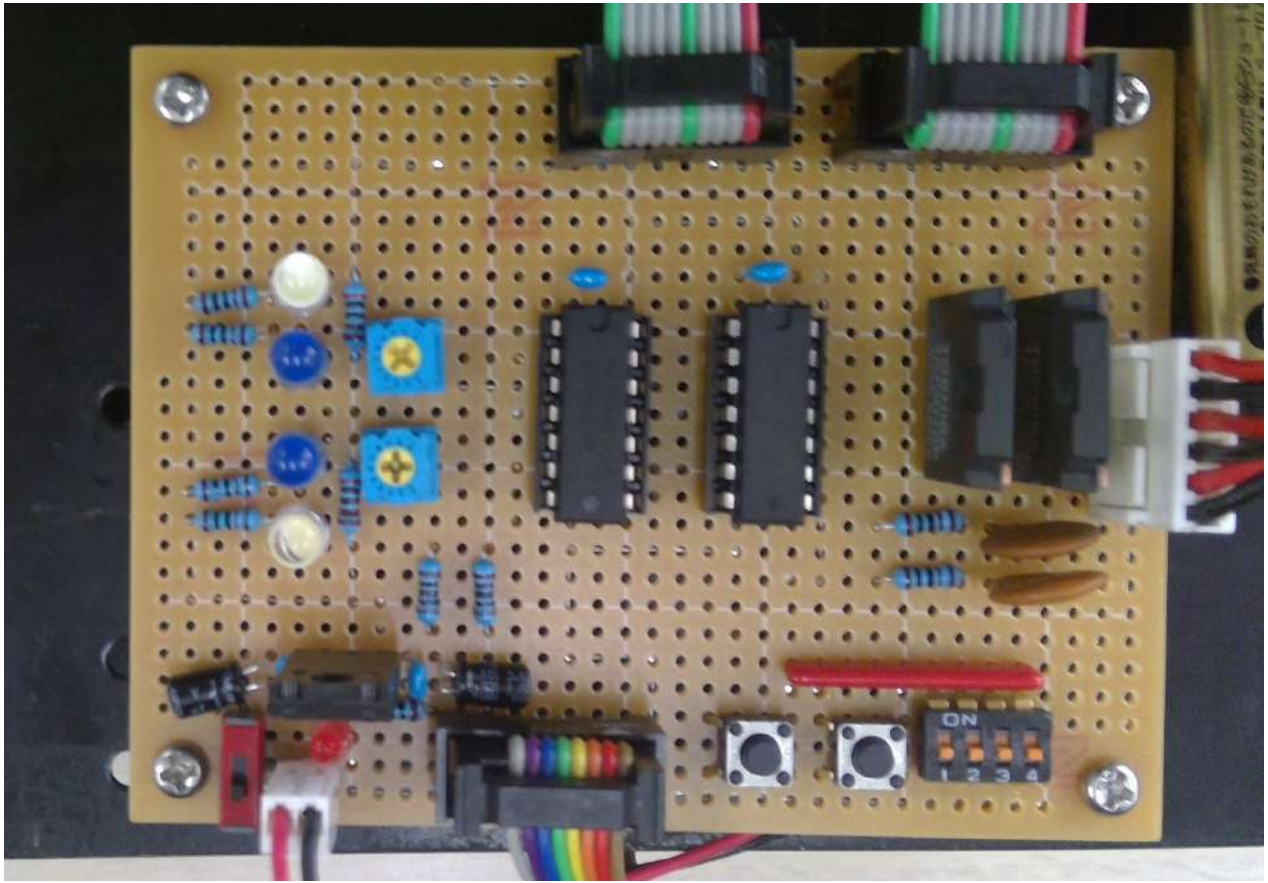
回路図:モーター


7.2V BATTERY
右モータードライバ
右DCモーター
PA-0
PA-1
VS
VCC
IN1
IN2
OUT+
OUT-
GND
0.1μF
1kΩ
MT
DME34B37G18A
TA7267BP
7.2V BATTERY
左モータードライバ
左DCモーター
PA-2
PA-3
VS
VCC
IN1
IN2
OUT+
OUT-
GND
0.1μF
1kΩ
MT
DME34B37G18A
TA7267BP

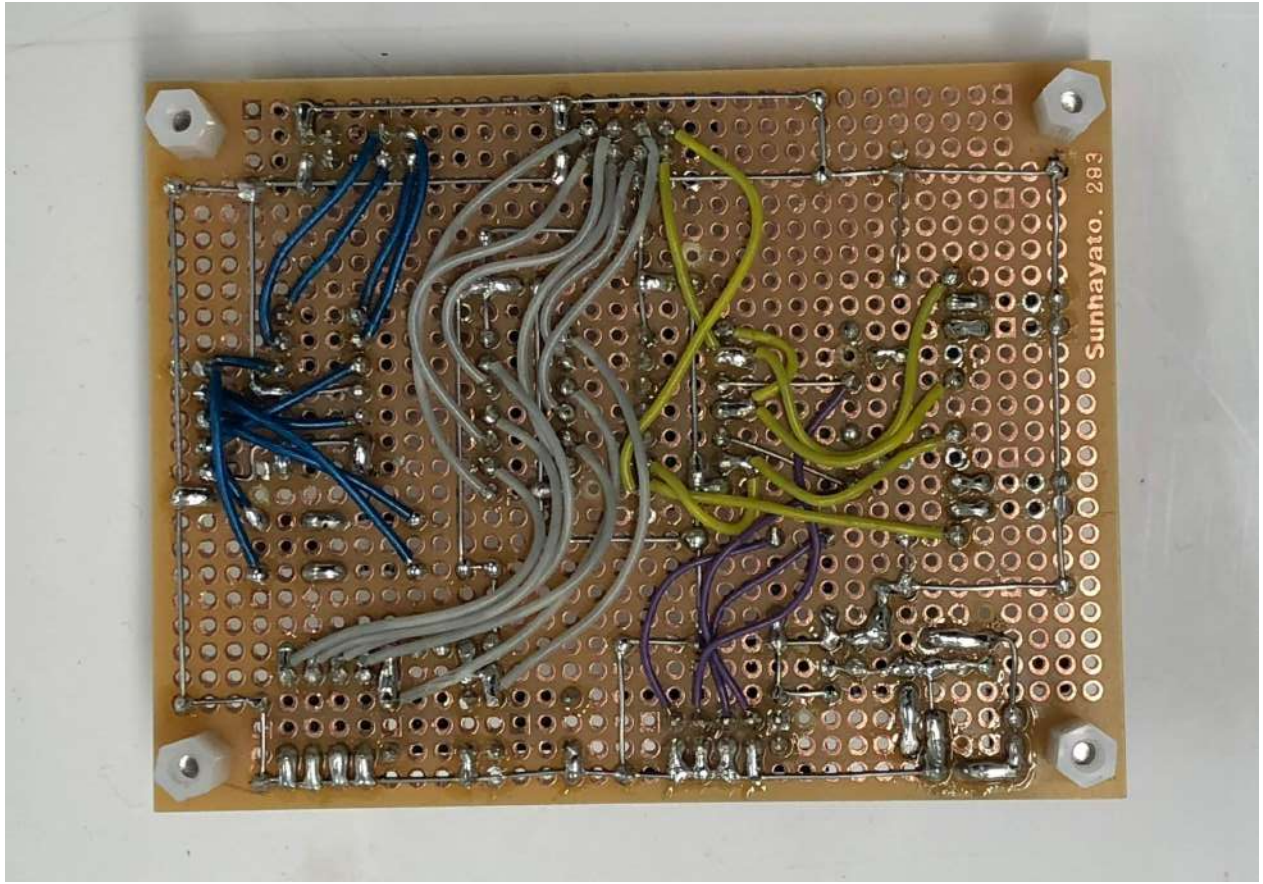
はんだ付け



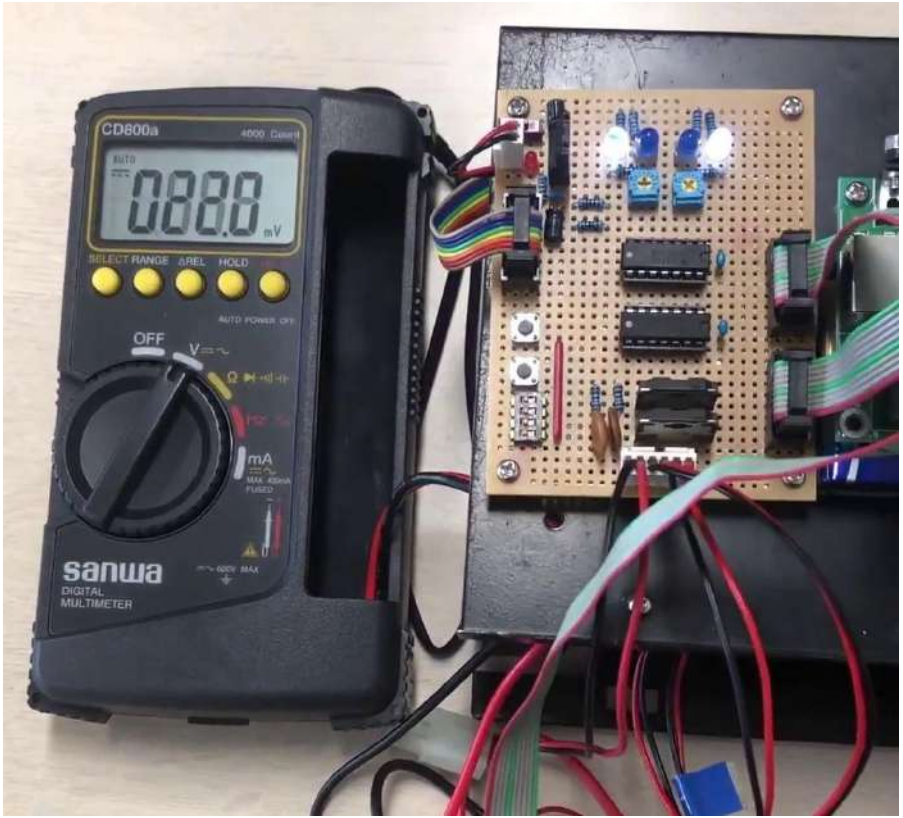
実際の基板（表）



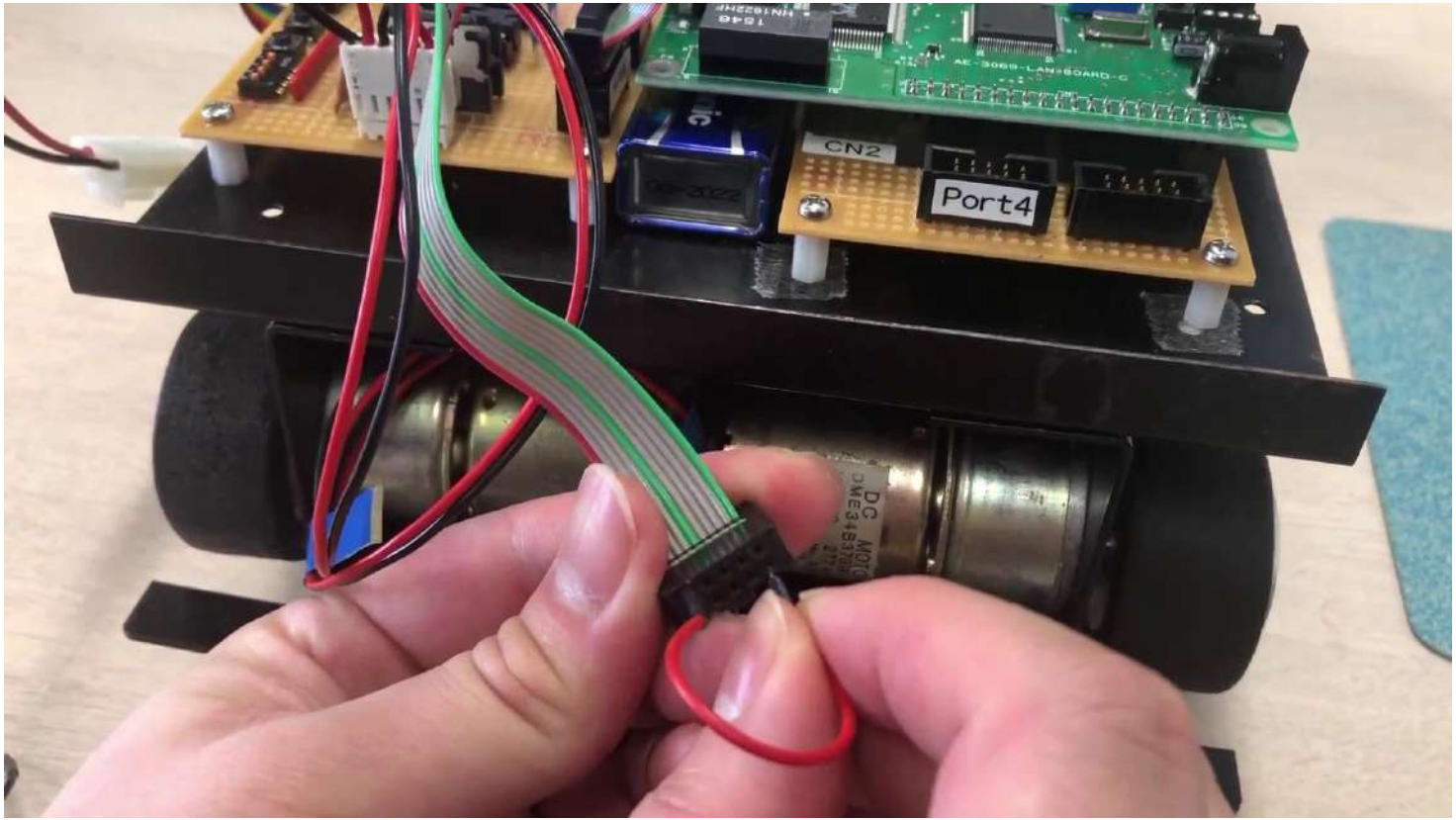
実際の基板(裏)



導通チェック：センサ入力テスト



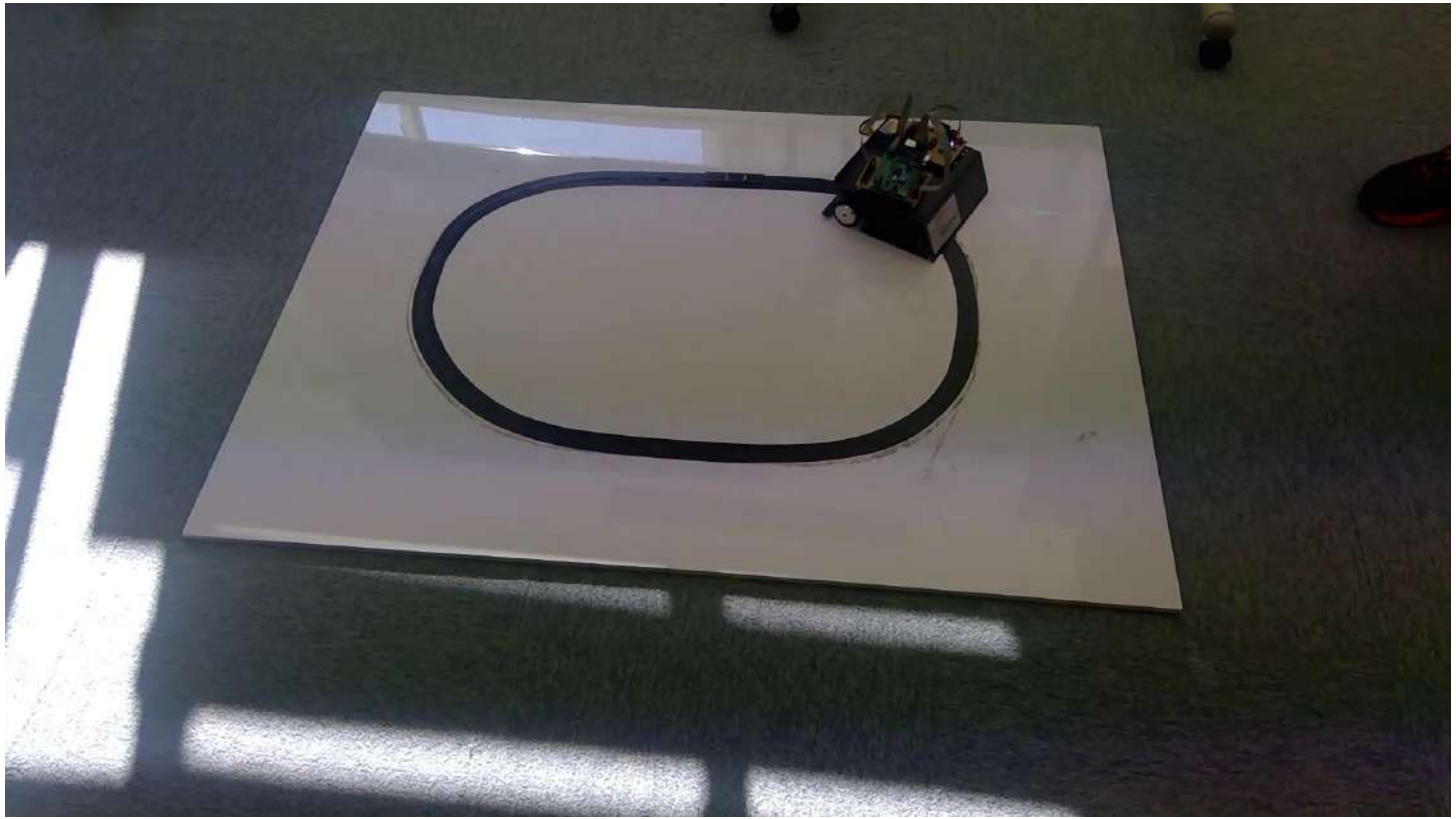
導通チェック：モータ出力テスト



テストプログラム

```
void test(VP_INT exinf){
        UB                      sw;
        ER                      ercd;
        UB                      state=0;
        initialize_port();
        for(;;){
                sw = get_sw();
                if(((sw & 0x03) == 0x00)&&(state == 1))       sil_wrb_mem((VP)H8P4DR,0x0A);
                else if(((sw & 0x03) == 0x01)&&(state == 1)) sil_wrb_mem((VP)H8P4DR,0x09);
                else if(((sw & 0x03) == 0x02)&&(state == 1)) sil_wrb_mem((VP)H8P4DR,0x06);
                else if(((sw & 0x03) == 0x03)&&(state == 1)) sil_wrb_mem((VP)H8P4DR,0x00);
                if(sw & 0x04)       state = 1;
                if(sw & 0x08){
                        state = 0;
                        sil_wrb_mem((VP)H8P4DR,0x00);
                }
        }
}
```

作動テスト



ソフトウェア設計

ソフトウェア設計

- 仕様
- 要求分析
  - イベントリスト
  - コンテキスト・ダイヤグラム
  - データ・フロー・ダイヤグラム（DFD)
  - DFDの詳細化
  - 状態のモデル化
- 設計
  - 階層構造図
  - モジュール分割
  - モジュール仕様設計
  - データ・フロー・ダイヤグラム

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（$P7_{4567}$）

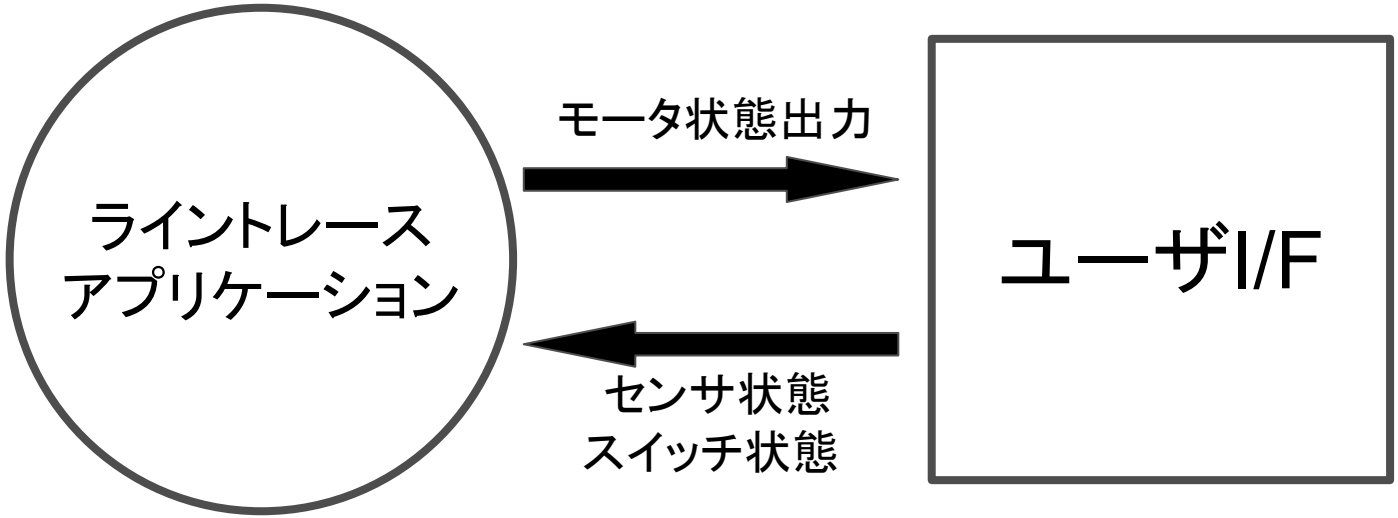
- 出力　　DCモーター　×2（右：PA_{01}、左：PA_{23}）
  （PA 0000|1010で左右ホイル前転）

- 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。

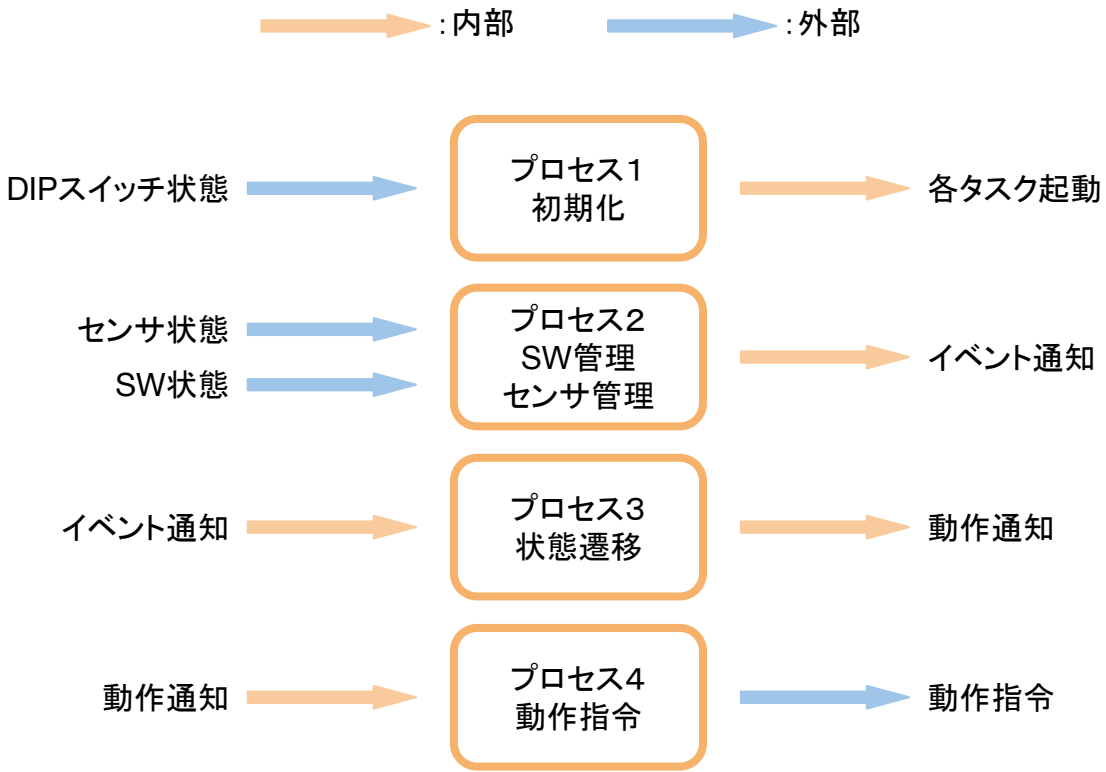
イベントリスト

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	停止状態なら走行開始 走行状態なら走行維持	走行開始 or走行継続
停止要求	PUSHSW2：ON	走行状態なら走行停止 停止状態なら停止維持	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	—

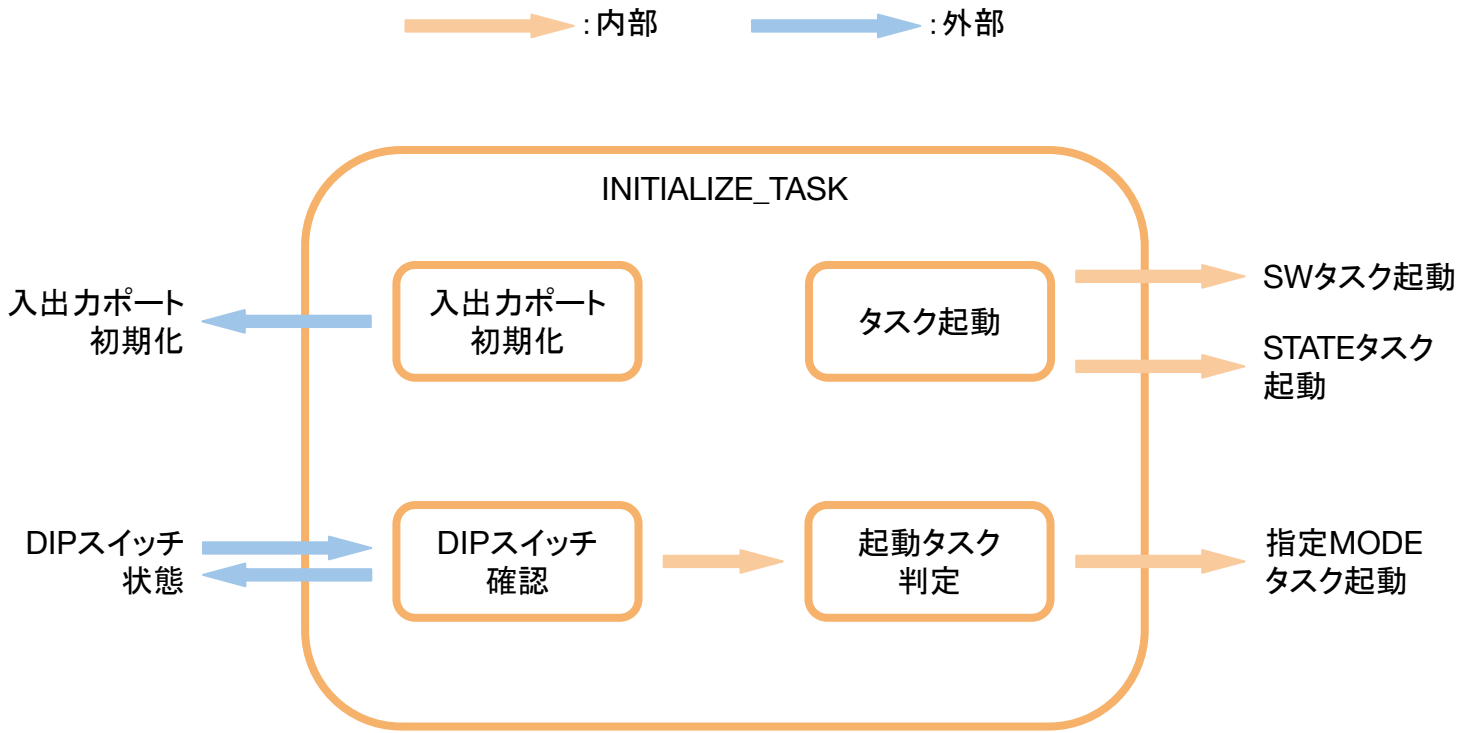
コンテキスト・ダイヤグラム



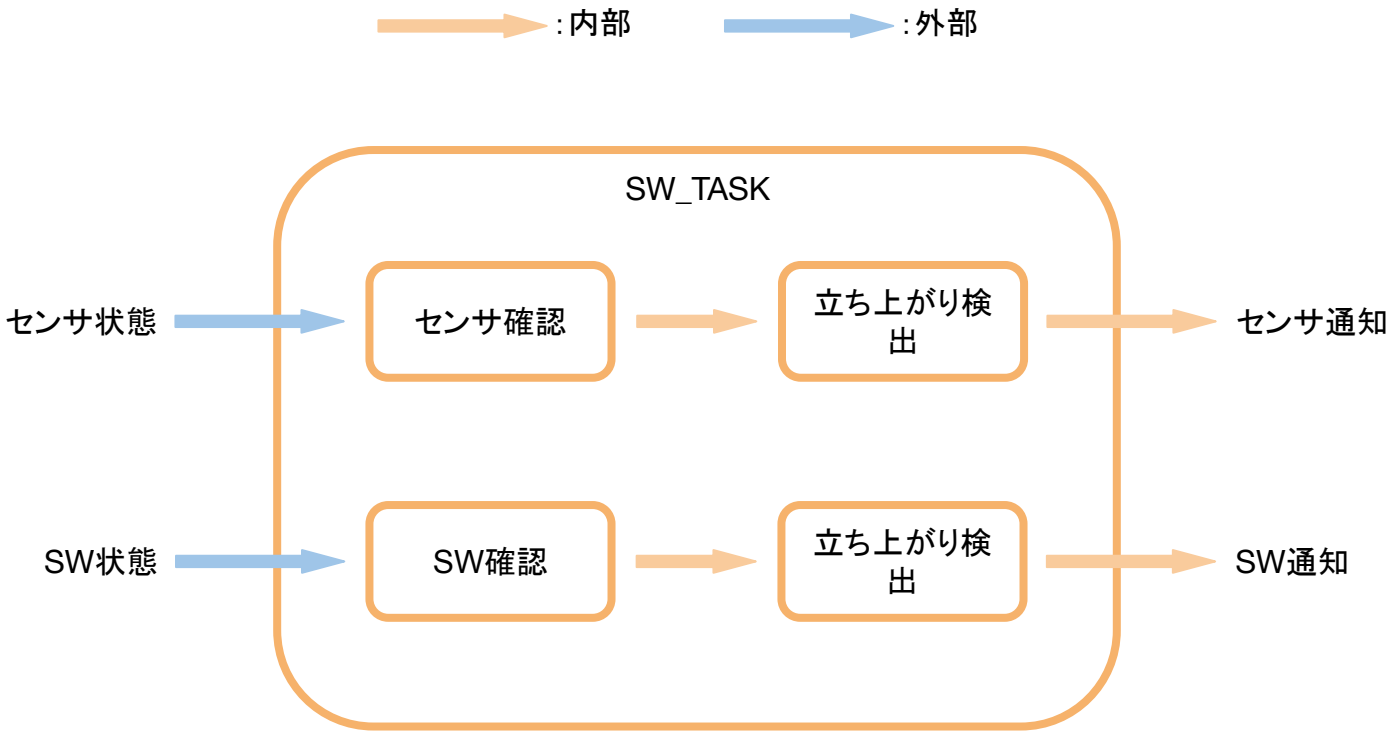
データ・フロー・ダイヤグラム



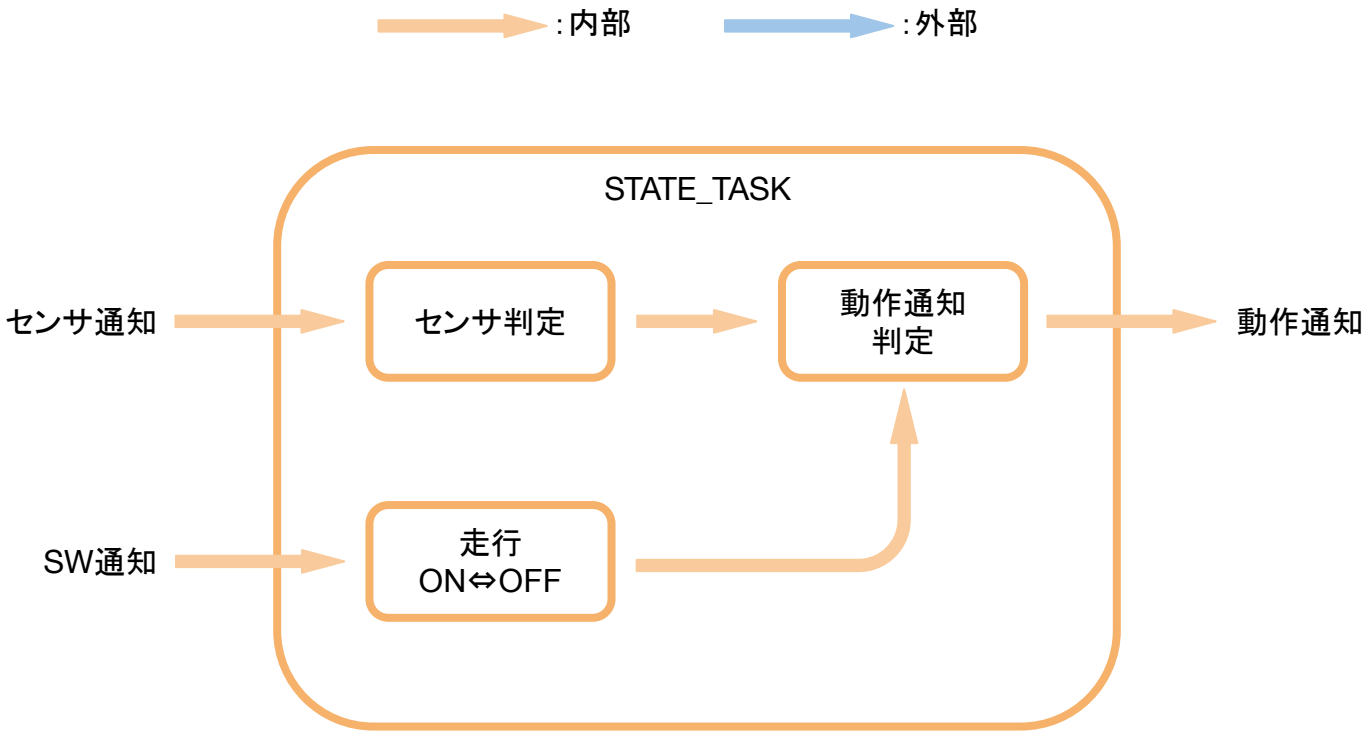
DFDの詳細化：プロセス1



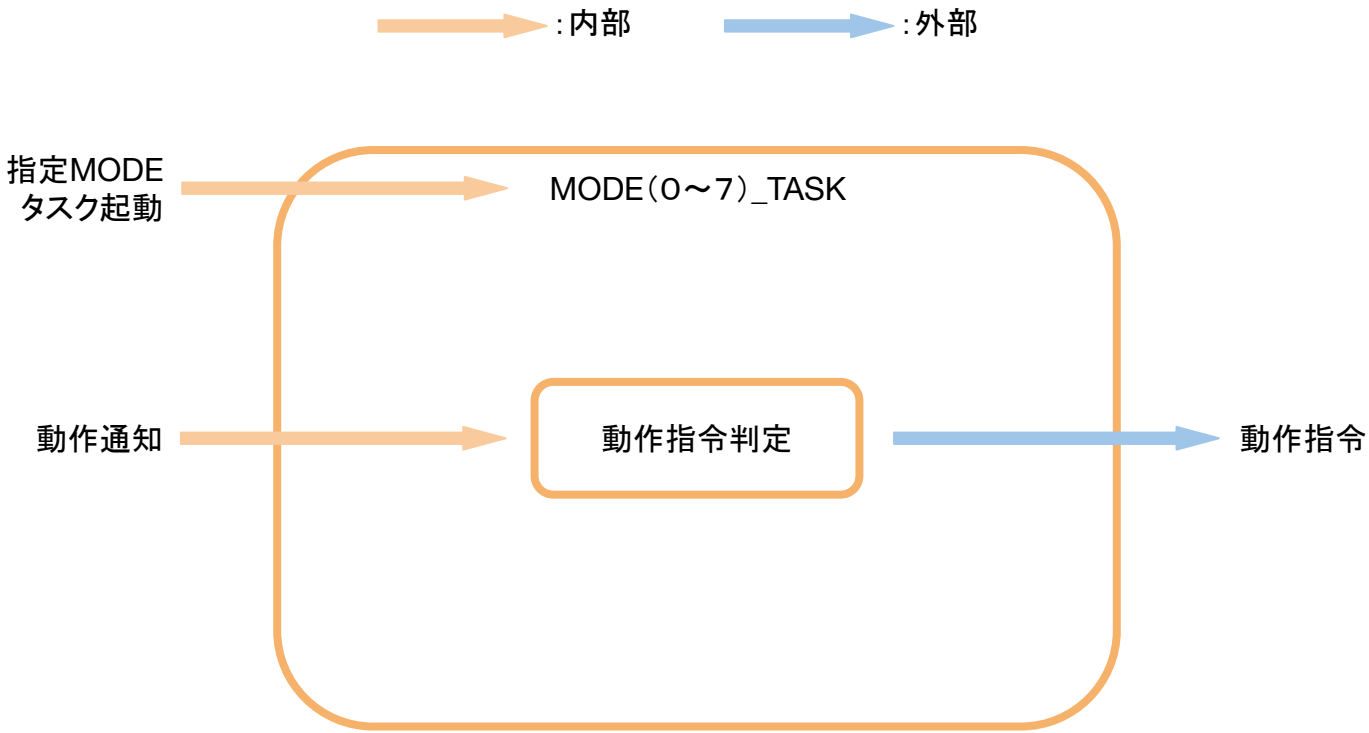
DFDの詳細化：プロセス2



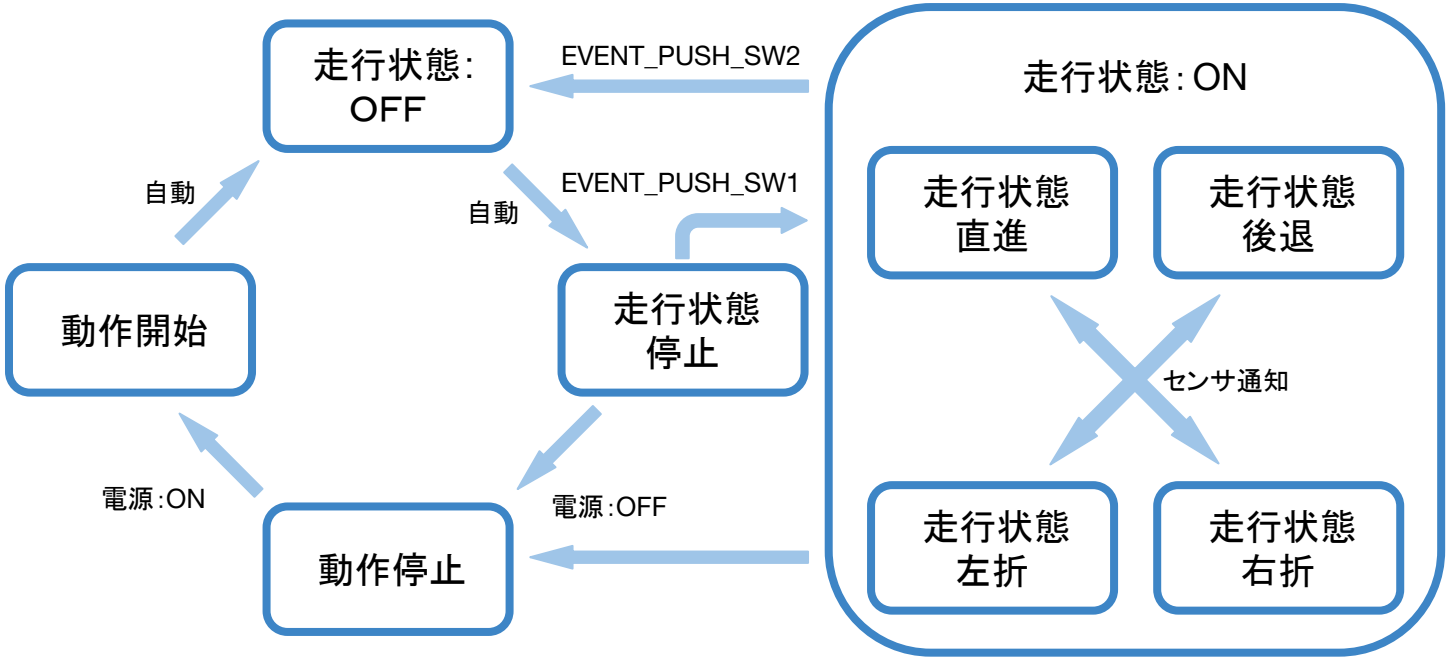
DFDの詳細化：プロセス3



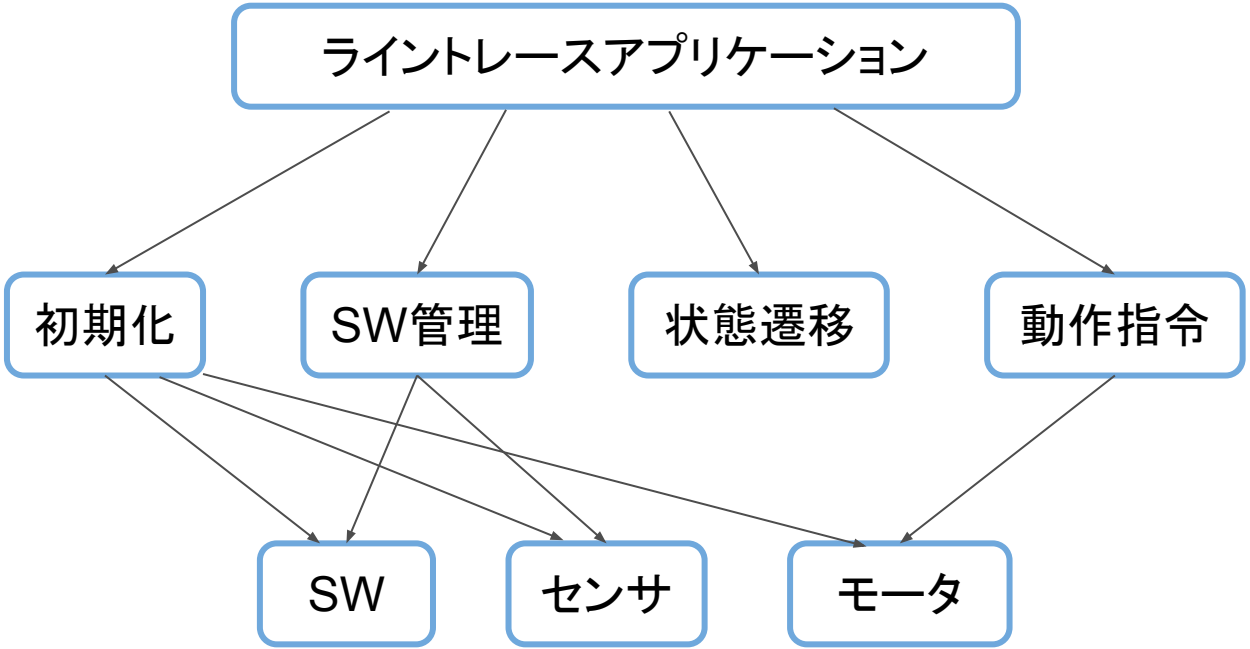
DFDの詳細化：プロセス4



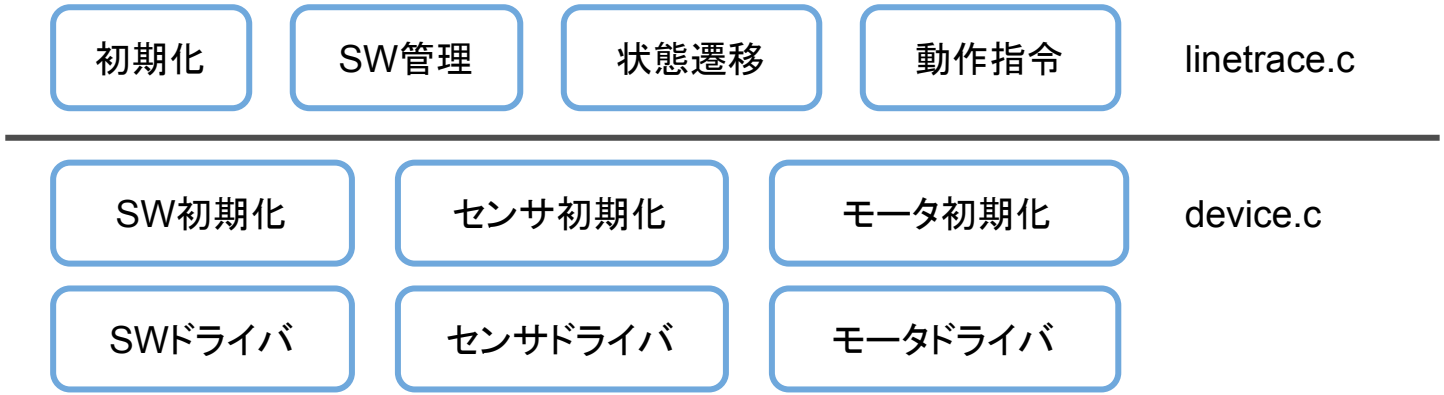
状態のモデル化


走行状態:
OFF
EVENT_PUSH_SW2
走行状態:ON
走行状態
直進
走行状態
後退
自動
EVENT_PUSH_SW1
自動
動作開始
走行状態
停止
センサ通知
電源:ON
電源:OFF
動作停止
走行状態
左折
走行状態
右折

設計：階層構造図



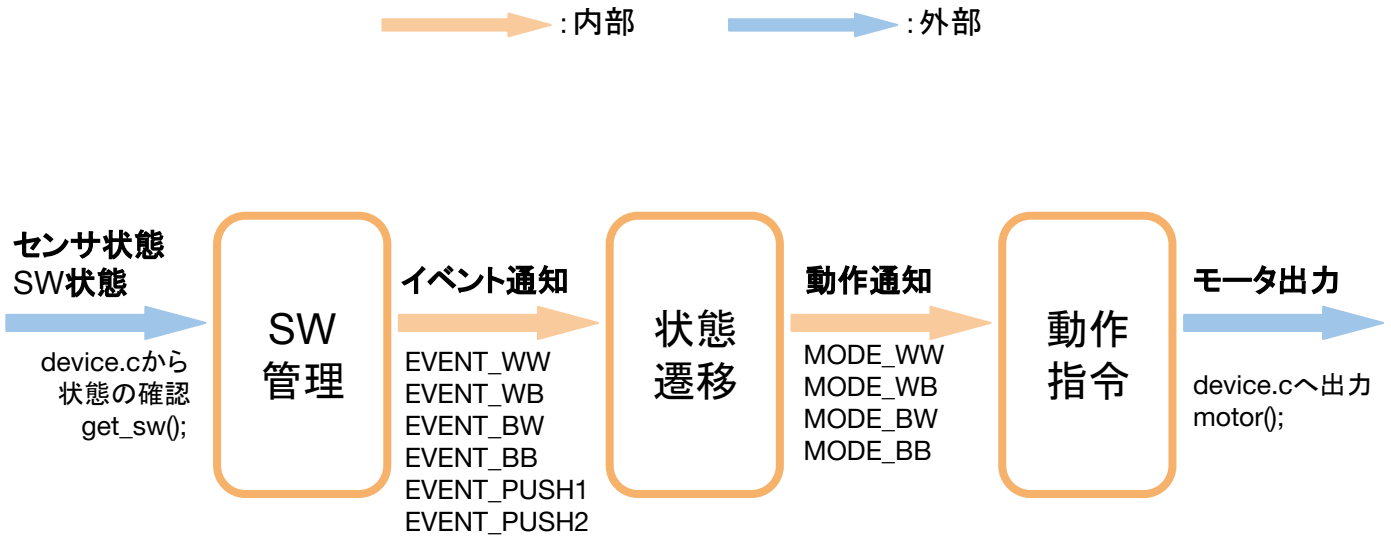
設計：モジュール分割


初期化
SW管理
状態遷移
動作指令
linetrace.c
SW初期化
センサ初期化
モータ初期化
device.c
SWドライバ
センサドライバ
モータドライバ

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	—	—

設計：データ・フロー・ダイヤグラム



ソースコード

linetrace.cfg

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

linetrace.h

```
#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

```c
#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);
}
```

linetrace.c

```
#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
```

linetrace.c

```
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();
}
```

linetrace.c

```
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);
    }
}
```

```
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);
    }
}
```

linetrace.c

```
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);
        }
    }
}
```

linetrace.c

```
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);
            default:          break;
        }
    }
}
```

反省点、改善点

反省点、改善点

・記録を日付つけたメモとしてメンバー各自がリーダーに渡し、
　リーダー日誌にまとめ、回覧をすればよかった。

・センサの特性を理解するのに時間がかかった。
　指でも検知すると思い、理解が遅れてしまった。

・はんだ付けで時間がかかってしまい、
　一人しか作業にあたれない時間が長くなり、
　班での活動時間が減ってしまった。

工夫点、改良点

工夫点、改良点

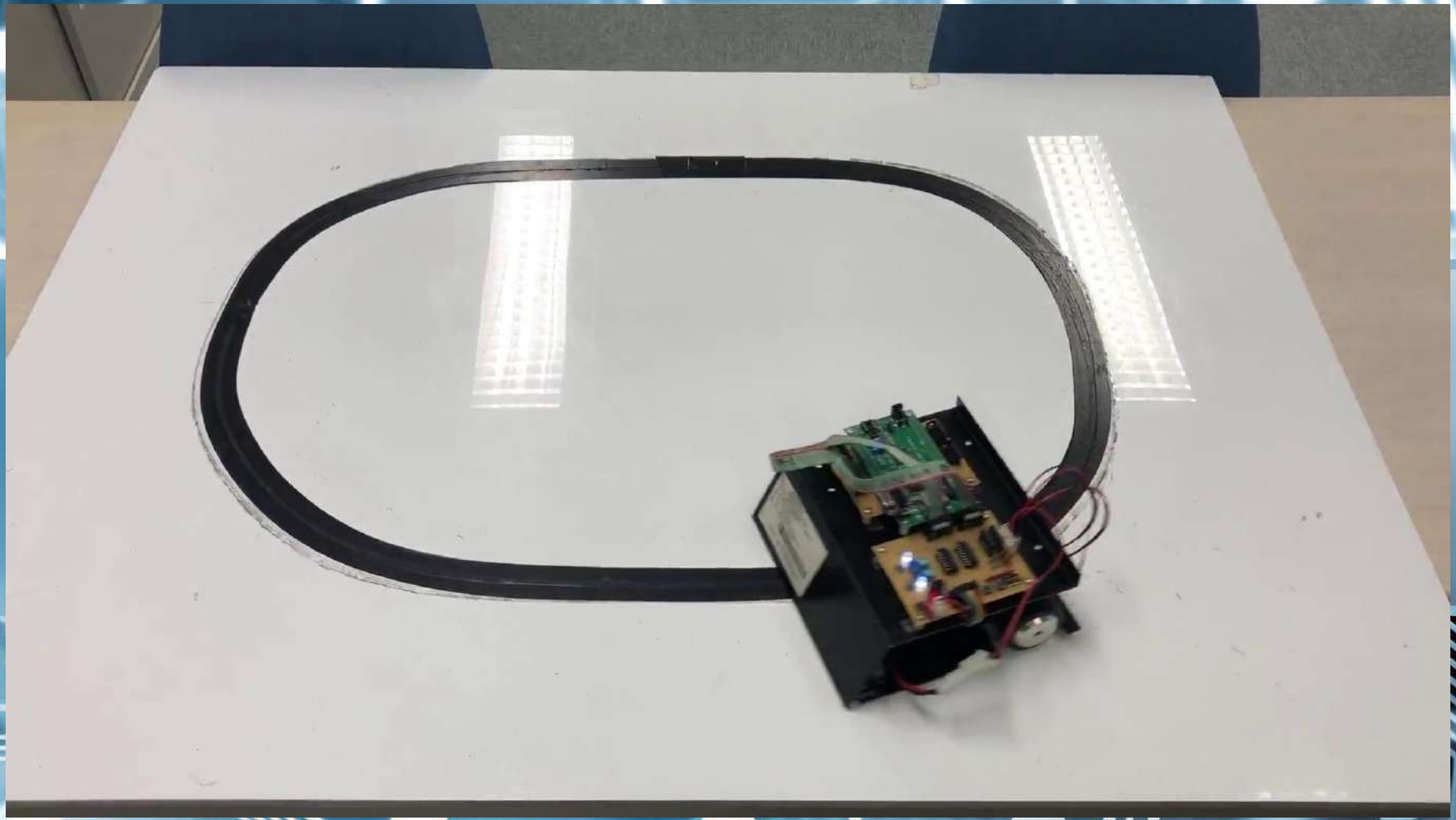
ハード

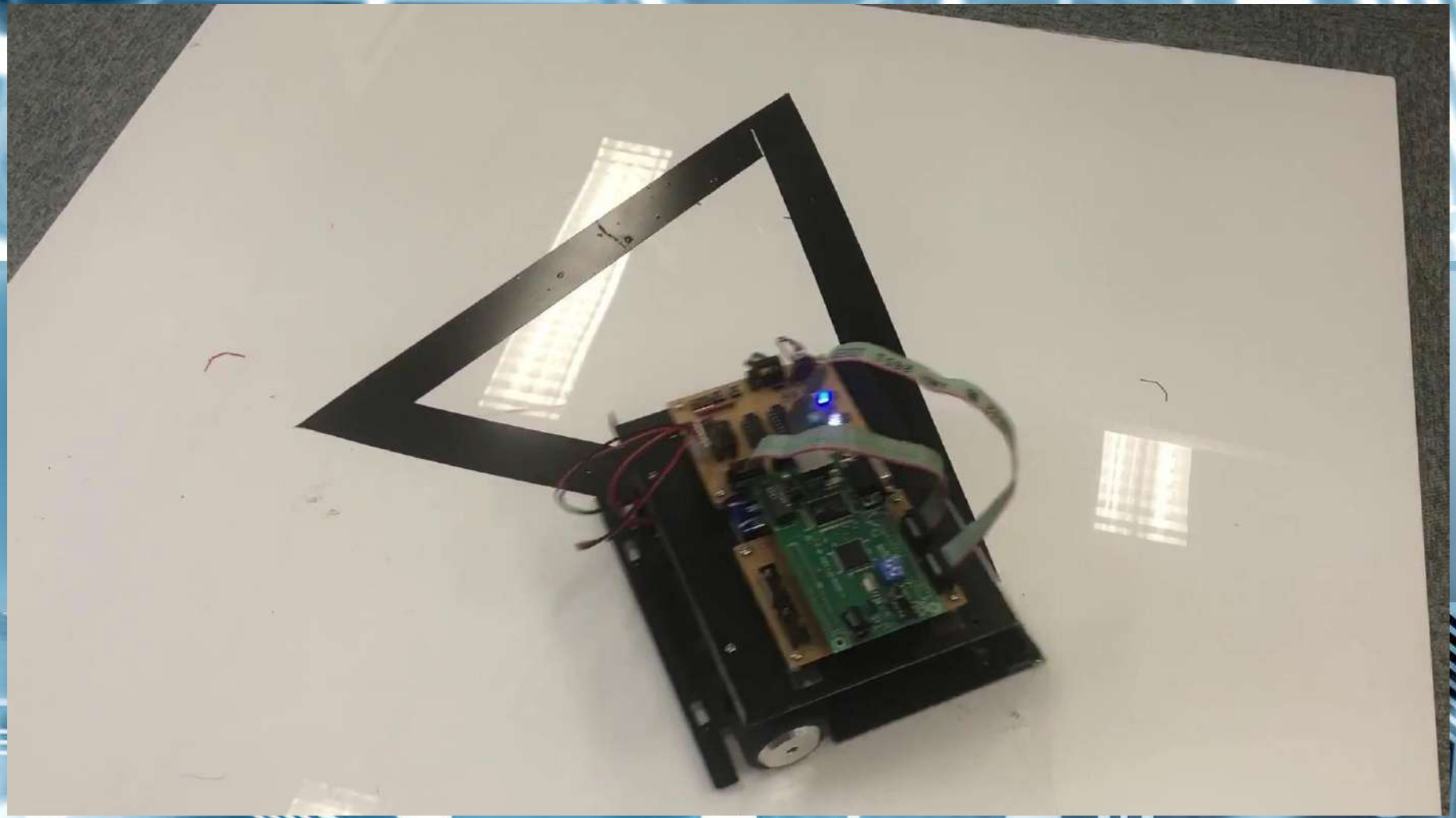
・プッシュスイッチの増設
・DIPスイッチの増設
・白青LEDなど、視覚的に分かりやすい造り
・モータードライバとモータの間にコンデンサを設け、安定化を図った
・右モータドライバとモータ間の配線を逆に繋げ、左右のモータに流す信号を同じにした
・安定化のために、デジタルICの近くにコンデンサを付けた
・電池省エネのために、動作表示LEDを330Ωの抵抗を1kΩに変えた（約10mA→約3mA）

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

～完成～





完