面白プログラムの発表

1126100
前川大輝

発表の流れの説明

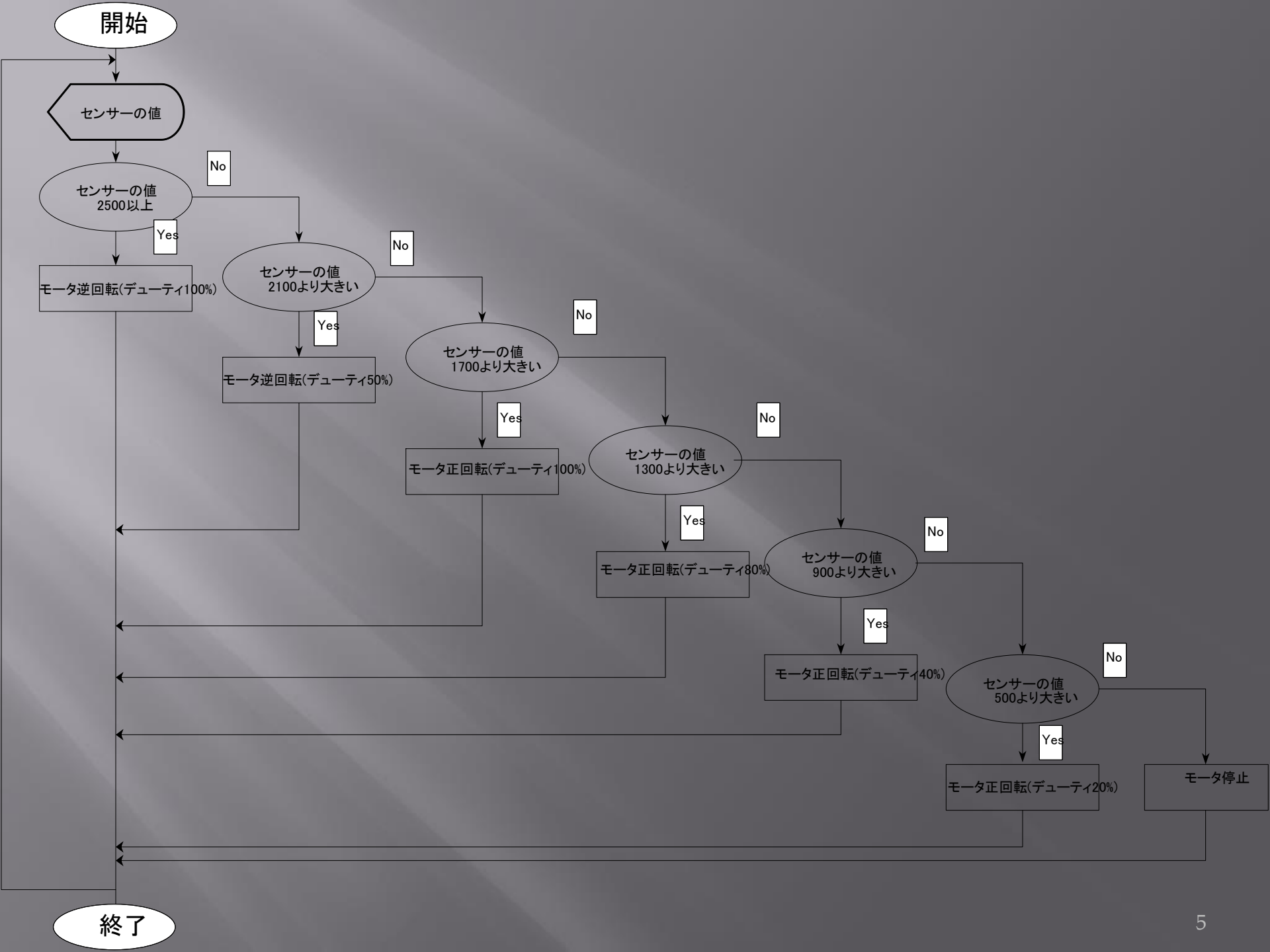
1．今回の学習目的
2．プログラム概要説明
3．実機によるデモンストレーション
4．自作ヘッダーファイルの解説
5．main文の解説
6．使用する回路の説明
7．質疑応答

１．今回の学習目的

A/D変換器を用いてPSDセンサーの制御を可能にすること

2.プログラム概要説明

センサーの返した値に応じてモータの回転数を変化させるプログラム


開始
センサーの値
センサーの値
2500以上
No
Yes
モータ逆回転(デューティ100%)
センサーの値
2100より大きい
No
Yes
モータ逆回転(デューティ50%)
センサーの値
1700より大きい
No
Yes
モータ正回転(デューティ100%)
センサーの値
1300より大きい
No
Yes
モータ正回転(デューティ80%)
センサーの値
900より大きい
No
Yes
モータ正回転(デューティ40%)
センサーの値
500より大きい
No
Yes
モータ正回転(デューティ20%)
モータ停止
終了

３．実機によるデモンストレーション

それではさっそくデモンストレーションします。

プログラムを解説します

まず、必要なヘッダーファイルのインクルードと
コンフィギュレーションの設定

これについて
説明します

```
#include "maekawa.h"

_FOSC(CSW_FSCM_OFF & XT_PLL8);
_FWDT(WDT_OFF);
_FBORPOR(PBOR_ON & BORV_20 & PWRT_64 & MCLR_EN);
_FGS(CODE_PROT_OFF);
```

maekawa.hの中身(1)

```
#include "p30F3014.h"
#include "outcompare.h"
#include "timer.h"
#include "adc12.h"
```

maekawa.hの中身(2)

```
//LCD用に定義
#define LCD_RS LATBbits.LATB5
#define LCD_E LATBbits.LATB4
#define LCD_D4 LATBbits.LATB9
#define LCD_D5 LATBbits.LATB10
#define LCD_D6 LATBbits.LATB11
#define LCD_D7 LATBbits.LATB12
```

maekawa.hの中身(3)

```
//プロトタイプ関数宣言
void lcd_out8(unsigned char);
void lcd_out4(int,unsigned char);
void machi_usec(int);
void machi_msec(int);
void lcd_data(unsigned char);
void lcd_clear(void);
void lcd_format(void);
void lcd_puts(unsigned char *);
void lcd_convert(long);
void motor(float,short);
void motor_format(void);
unsigned int get_SensorValue(void);
void device_format(void);
```

maekawa.c
から参照している

maekawa.hの中身(4)

```
#define CLOCK  80

/*PWMのピンを定義*/
#define PWM_1A_AND_2A LATDbits.LATD1
#define PWM_1B LATFbits.LATF0
#define PWM_2B LATFbits.LATF1
#define PWM_3A LATDbits.LATD2
#define PWM_4A LATFbits.LATF6

/*モータの正回転、逆回転を定義*/
enum MotionType{normal,reverse};
/*モータの種類を定義*/
enum MotorType{motor1,motor2};
```

maekawa.cについて解説します

maekawa.cの中身(1)

```
#include "stdio.h"
#include "maekawa.h"
```

さらにこの下には13個の関数が定義されています

maekawa.cの中身(2)　motor関数

Duty比設定(+100~-100)
motor1かmotor2を選択

```
void moto (float Duty,short MotorType){
        if(Duty < 0){
                Duty = -Duty;
                Duty = Duty/100;
                /*逆回転にセット*/
                switch(MotorType){
                        case motor1:
                                PWM_1B = 0;
                                PWM_2B = 1;
                        break;
                        case motor2:
                                PWM_3A = 0;

                                PWM_4A = 1;
                        break;
                }
        }else if(Duty >= 0){
                Duty = Duty/100;
                /*正回転にセット*/
                switch(MotorType){
                        case motor1:
                                PWM_1B = 1;
                                PWM_2B = 0;
                        break;
                        case motor2:
                                PWM_3A = 1;
                                PWM_4A = 0;
                        break;
                }
        }
        switch(MotorType){
                case motor1:
                        SetDCOC2PWM(3125*Duty);
                break;
                case motor2:
                        SetDCOC1PWM(3125*Duty);
                break;
        }
}
```

それぞれデューティ比をセット

maekawa.cの中身(3)　motor_format関数

```
void motor_format(void){
        //初期化
        TRISD = 0x0000;
        TRISF = 0x0000;
        OpenOC2(OC_IDLE_CON & OC_TIMER2_SRC &
OC_PWM_FAULT_PIN_DISABLE,0,0);
        OpenOC1(OC_IDLE_CON & OC_TIMER2_SRC &
OC_PWM_FAULT_PIN_DISABLE,0,0);


        //10msec(4/80Mhz×64×3125=10msec)
        OpenTimer2(T2_ON & T2_GATE_OFF & T2_PS_1_64 &
T2_SOURCE_INT,3125-1);

}
```

motor1用にOPCをオープン

motor2用にOPCをオープン

maekawa.cの中身(4)　get_SensorValue関数(1)

ADコンバーター初期設定変数宣言

```
unsigned int ADconfig1 = ADC_MODULE_ON & ADC_IDLE_CONTINUE & ADC_FORMAT_INTG &
                         ADC_CLK_AUTO & ADC_AUTO_SAMPLING_OFF & ADC_SAMP_OFF;
unsigned int ADconfig2 = ADC_VREF_AVDD_AVSS & ADC_SCAN_OFF &ADC_SAMPLES_PER_INT_1
                         & ADC_ALT_BUF_OFF & ADC_ALT_INPUT_OFF;
unsigned int ADconfig3 = ADC_SAMPLE_TIME_8 & ADC_CONV_CLK_SYSTEM &
                         ADC_CONV_CLK_32Tcy;
unsigned int ADconfigPort = ENABLE_AN1_ANA;
unsigned int ADconfigScan = 0x000;
          //自動取得しないため関係ない
unsigned int Channel1 = ADC_CH0_POS_SAMPLEA_AN1 & ADC_CH0_NEG_SAMPLEA_NVREF;

unsigned int ADvalue[5],i,ave=0;
```

これらの初期設定について説明していきます。

```
unsigned int ADconfig1 = ADC_MODULE_ON & ADC_IDLE_CONTINUE &
ADC_FORMAT_INTG &     ADC_CLK_AUTO & ADC_AUTO_SAMPLING_OFF &
ADC_SAMP_OFF;
```

- ADC_MODULE_ON　　ADCモジュールのon/off
- ADC_IDLE_CONTINUE　　アイドルモード時に動作継続
- ADC_FORMAT_INTG　　変換結果のデータ形式が符号なし整数
- ADC_CLK_AUTO　　変換開始トリガを手動に設定
- ADC_AUTO_SAMPLING_OFF　　自動サンプリングOFF
- ADC_SAMP_OFF　　サンプル開始制御OFF

```
unsigned int ADconfig2 = ADC_VREF_AVDD_AVSS & ADC_SCAN_OFF
&ADC_SAMPLES_PER_INT_1 & ADC_ALT_BUF_OFF & ADC_ALT_INPUT_OFF;
```

- ADC_VREF_AVDD_AVSS　　VrefH = AVDD(28),VrefL = AVSS
- ADC_SCAN_OFF　　チャンネル０は使わない
- ADC_SAMPLES_PER_INT_1　　1回ごとに割り込み発生
- ADC_ALT_BUF_OFF　　バッファーモードOFF

- ADC_ALT_INPUT_OFF　　交互変換指定OFF

今回は1つしか
ため込まないため

unsigned int ADconfig3 =ADC_SAMPLE_TIME_8&ADC_CONV_CLK_SYSTEM
& ADC_CONV_CLK_32Tcy;

- ADC_SAMPLE_TIME_8　　自動サンプル時間指定
- ADC_CONV_CLK_SYSTEM　　変換用クロックをシステムクロックに設定
- ADC_CONV_CLK_32Tcy　　変換クロック選択

今回自動サンプリング機能を使わないため
これらの初期設定は意味を持たない

unsigned int ADconfigPort =ENABLE_AN1_ANA

ADPCFGレジスタ
のPCFG1ビットが０となります

ADPCFG：16個のピンをデジタルで使うかアナログ入力にするかの設定を行います。この設定は単純でアナログ入力にするピンに対応するビットを０にするだけです

```
unsigned int ADconfigScan = 0x000;
```

0000 0000 0000
ADCSSLレジスタに設定する値

ADCSSLレジスタ：チャネル０の自動スキャンに含める入力を指定します。含める入力に相当するビットを1にします

今回はチャネル０を使いませんので
該当ビットは存在しません

unsigned int Channel1 = ADC_CH0_POS_SAMPLEA_AN1 &
ADC_CH0_NEG_SAMPLEA_NVREF

ADCHSレジスタの設定

・ADC_CH0_POS_SAMPLEA_AN1　　チャンネル1に接続します

・ADC_CH0_NEG_SAMPLEA_NVREF　　A/Dはチャンネルごとに測定する範囲を設定できます。上記設定は各々のチャンネルポートとGND間を測定する設定です。

以上で初期設定変数の解説を終了します

main.cの中身(4)　Get_SensorValue関数

```
unsigned int get_SensorValue(void){
        //AD initialize
        OpenADC12(ADconfig1,ADconfig2,ADconfig3,ADconfigPort,ADconfigScan);

        for(i=0;i<3;i++){
                SetChanADC12(Channel1);
                ADCON1bits.SAMP = 1;

                while(ADCON1bits.DONE == 0);

                ADvalue[i] = ReadADC12(0);
                ave += ADvalue[i];
        }
        ave /= 3;

        if(ave <= 200){                         //ノイズ対策
                ave = 0;
        }

        return ave;
}
```

先ほど説明した初期設定変数は省略

先ほどの設定で
ADCオープン

ADC
使用開始

ADCのチャンネル１を
選択

DONEレジスタを判定
し0(変換中)である限り時間待ち

0番地レジスタからデータを取得
(今回はチャンネル１の値)

maekawa.cの中身(5)　device_format関数

```
void device_format(void){
        TRISB = 0x1CF;

        lcd_format();
        motor_format();

}
```

maekawa.cに定義されている
関数を使用するための
総合的な入出力設定

main文の解説をします

```
int main(void){
    device_format();

                while(1){

                ADvalue = get_SensorValue();

                lcd_convert((long)ADvalue);
                machi_msec(100);
                lcd_clear();

                if(ADvalue >= 2500){
                        motor(-100,motor1);
                }else if(ADvalue > 2100){
                        motor(-50,motor1);
                }else if(ADvalue > 1700){
                        motor(100,motor1);
                }else if(ADvalue > 1300){
                        motor(80,motor1);
                }else if(ADvalue > 900){
                        motor(40,motor1);
                }else if(ADvalue > 500){
                        motor(20,motor1);
                }else{
                        motor(0,motor1);
                }
        }
}
```

初期化関数実行

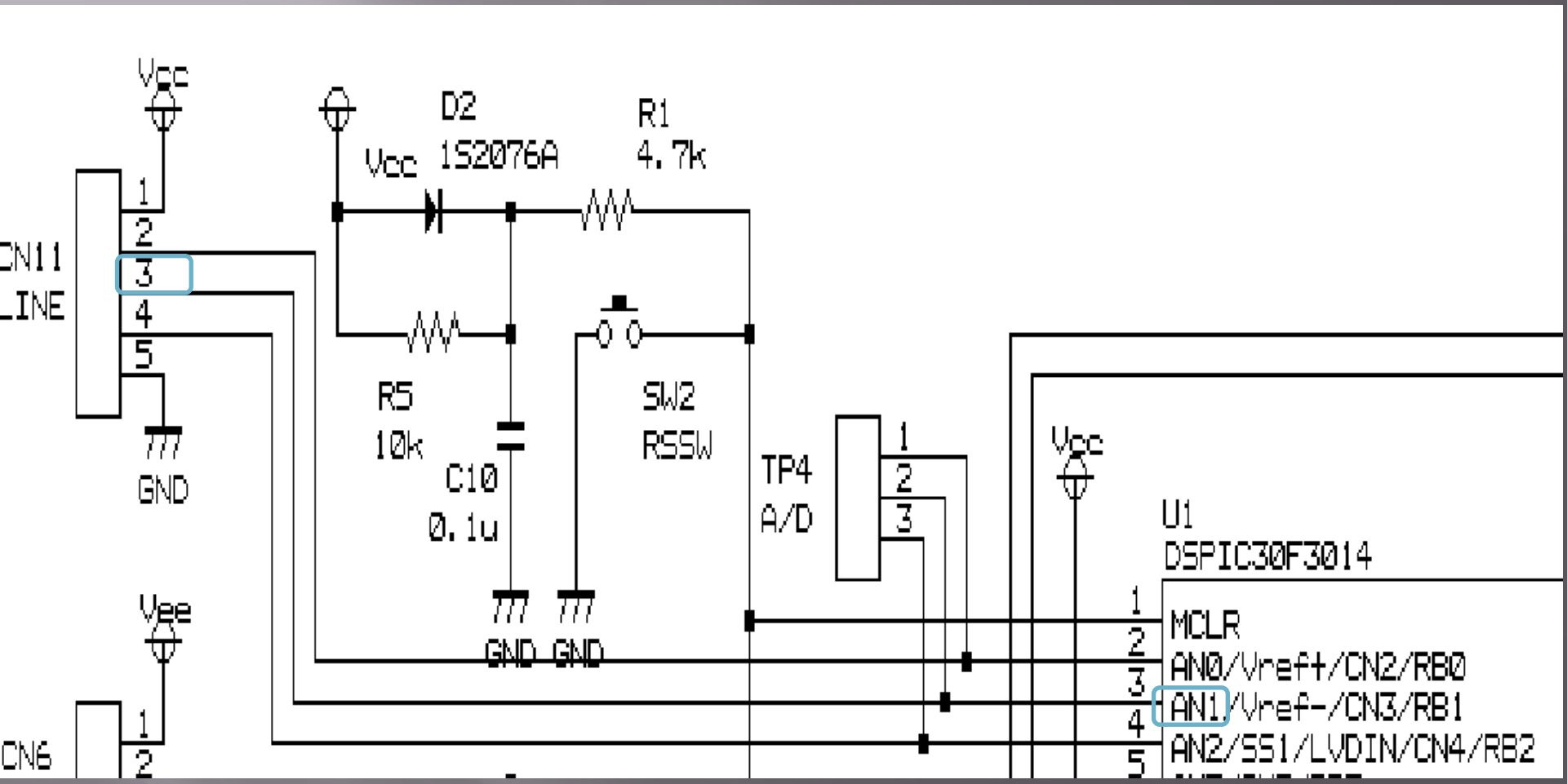
センサーの値を取得

LCDにADの値を表示

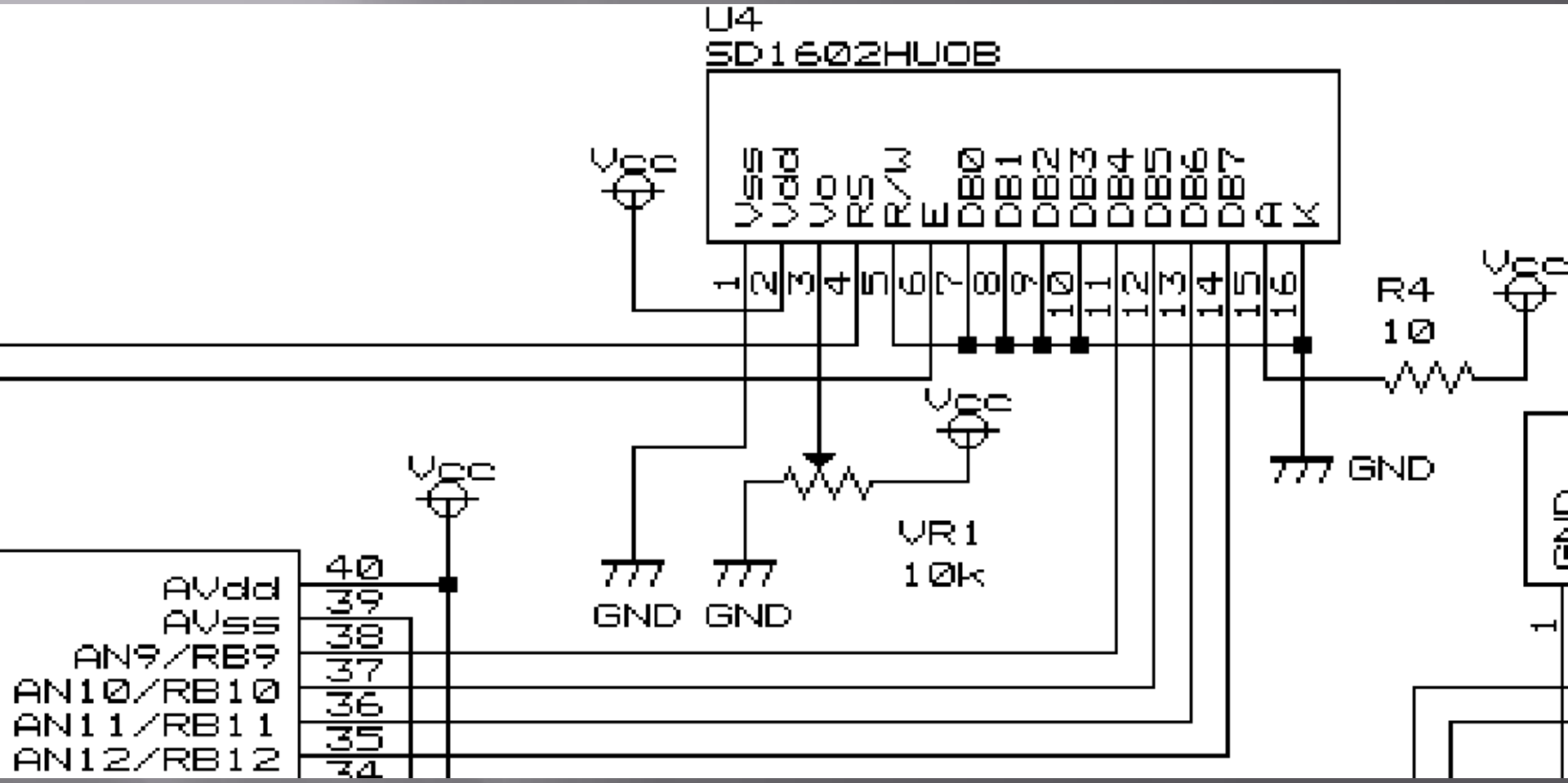
センサーの値に応じて
motor1の回転数を
変化させる

今回使用した回路を解説します

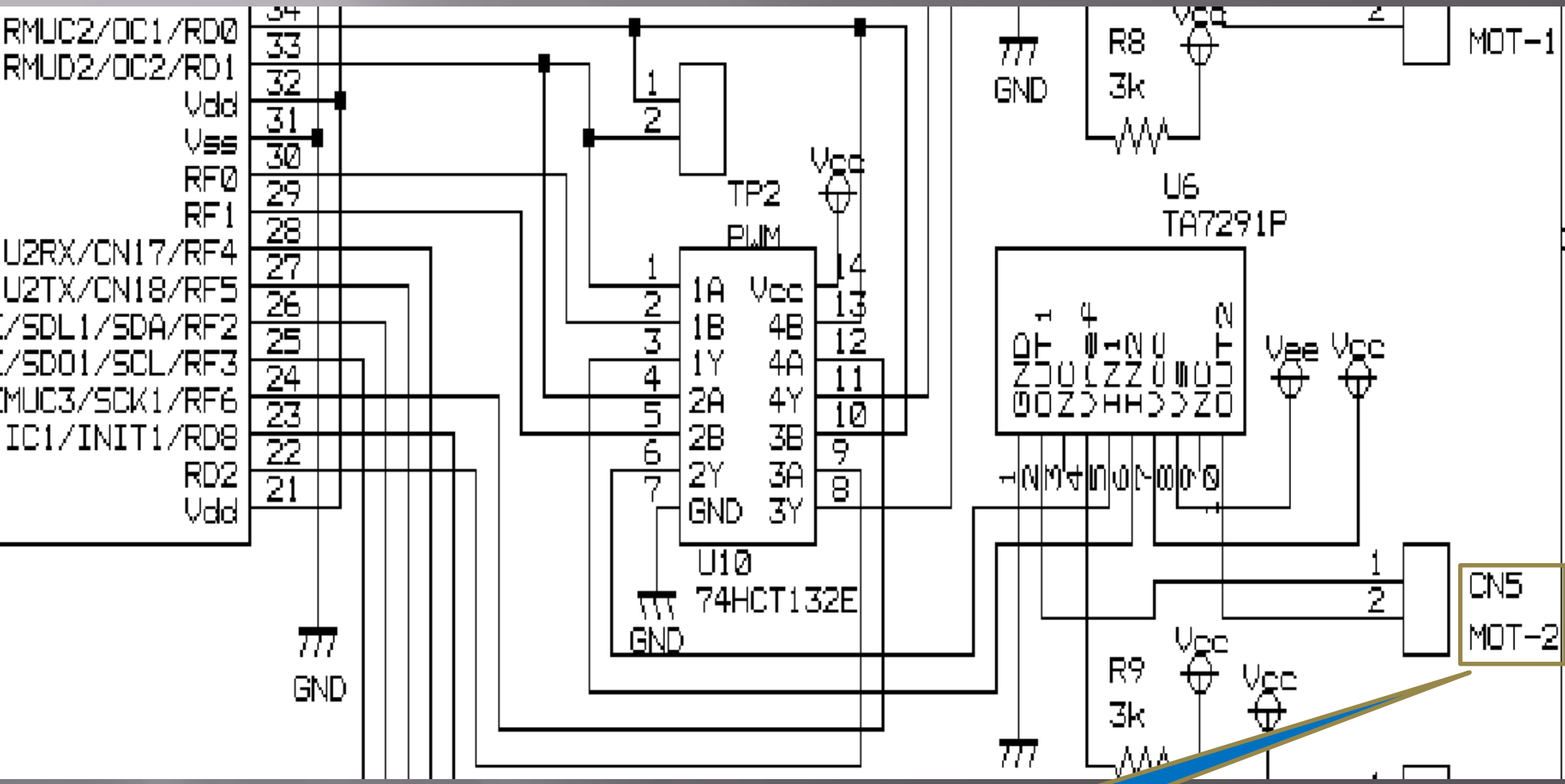
センサーの回路図


Vcc
D2
1S2076A
R1
4.7k
Vcc
CN11
LINE
1
2
3
4
5
R5
10k
C10
0.1u
SW2
RSSW
GND
GND
GND
TP4
A/D
1
2
3
Vcc
U1
DSPIC30F3014
1
2
3
4
5
MCLR
AN0/Vref+/CN2/RB0
AN1/Vref-/CN3/RB1
AN2/SS1/LVDIN/CN4/RB2
Vee
CN6
1
2

LCDの回路図


U4
SD1602HUOB
Vcc
R4
10
VR1
10k
GND
AVdd
AVss
AN9/RB9
AN10/RB10
AN11/RB11
AN12/RB12
40
39
38
37
36
35

motor1の回路図



CN 4
MOT-1

これですべての解説を終了します。

7. 質疑応答

質問はなんでもいいです。

以上にて発表を終了させていただきます。

ご清聴ありがとうございました。