制御情報工学実習 II

第３回まとめレポート
（ロータリーエンコーダー、
車両型ロボット、
A/D 変換（フォトセンサなど））

２０２４年１２月６日提出

２S１６番　上﨑 悠希

１.実験の目的

ロータリーエンコーダーの理解、A/D 変換（フォトセンサなど）の利用、車両型ロボット操作およびセンサを利用した車両型ロボットの簡単な制御を目的とする。

２.使用装置・計測器

・車両型ロボット

・パーソナルコンピューター（プログラム時）

３.実験方法

エンコーダーの信号に留意したプログラムによる、A/D 変換を用いた車両型ロボットの簡単な操作によって目的を達成する。また、この実験を達成するために用いたプログラムを付録（１）に示す。

これからこのプログラムを各所に分けて説明することで目的の達成をここに証明する。

まずプログラム冒頭（3～22）に書いてある A/D 変換について記す。

この関数たちはそれぞれ役割がありすべてがあることによって A/D 変換が成り立っている。

”ADStart”で A/D 変換の開始、”ADCexec”で A/D 変換の実行、”ADRead”

で A/D 変換された値の返却をしている。また今回のプログラムでは、フォトリフレクタの信号を A/D 変換することによって、通った場所の暗さを判別し、車両を制御するものになっている。また、暗さを判別するために用いたしきい値についてまとめたものを表１に示す。

表１　フォトリフレクタのしきい値

	Ｐ７端子の電圧値[mV]	デジタル符号
黒い紙を置いたとき	280	57.344
白い紙を置いたとき	1480	303.104
中間の値（しきい値）	880	180.224

続いて（37～60）に書いてある車両制御のプログラム（エンコーダの値に留意）について記す。

この関数たちは車両を各方向に動かすメソッドが詰められている。

どちらに動くかは関数名を見れば明白だが、その方法は単純にモータを制御しているのではなく、エンコーダーの信号を利用したものである。つまるところ、エンコーダーの信号と、シフト演算等を利用して前進や後退などのプログラムを組んでいるのである。

この 2 つに分けて説明したことがこのプログラムの本質でありその他のプログラム（main 関数）はこれらを利用し実際に動かしているだけである。

3.実行結果

完全と言っていいほどにプログラム道理に車両型ロボットの制御に成功した。つまりこれは、今回の目的である、エンコーダーの理解、A/D 変換の利用及びセンサーの利用を含めた車両型ロボットの制御に成功したと言っても過言ではないだろう。

４.考察

今回の課題を完全に達成できたと仮定するのであれば、車両型ロボットのさらなる利用や、車両型ロボットにとどまらない A/D 変換の利用など、幅広い場面で今回学んだことを役に立たせることができるのではないかと考える。また、エンコーダーの信号などの理解を更に深めることによって、今まで以上に実用的なプログラムを組めるようになるのではないかと考える。

５．感想

今回の授業はとても実用的なことを沢山学ぶことができ、とても良かったように思う。学んだことを忘れずに、３年からの実習にいかし、将来的にもっと実用的なプログラムを組めるようになりたい。

６．付録

（１）サンプルプログラム

下にサンプルプログラムを示す。また、非常に見づらいため、プログラムを閲覧できるリンクを次に示す。

プログラムリンク：https://github.com/soooda-0215/-/tree/main

```
#include <h8/reg3067.h>
#include <mes2.h>
int ADdata = 0;
// A/D Conv. Start
void ADStart(void){
ADCSR = ADCSR | 0x20;
}

// Read aft. A/D Conv.
int ADRead(void){
    if((ADCSR & 0x80) == 0x80){
        ADdata = (ADDRAH << 2) | (ADDRAL & 0x03);
        ADCSR = (ADCSR & (~0x80));
    }
    return ADdata;
}
```

```
// A/D Conv. Execution
void ADCexec(){
if((ADCSR & 0x20) == 0x00) //  問５
        ADStart();
}
void init() {
    //  モータ用端子(P4)の設定
    // P40 と P41 はモータ 1 の出力用、P42 と P43 はモータ 2 の出力用、P44〜P47 はエンコーダ（入力用）
    P4DDR = 0x0F; //  下位 4 ビット(P40-P43)を出力、上位 4 ビット(P44-P47)を入力に設定

    // LED 用端子(P6)の設定
    //  全て出力用（P60, P61, P62 のみ使用）
    P6DDR = 0x07; //  下位 3 ビット(P60-P62)を出力に設定

    //  スイッチ(PA)の設定
    // PA4 が白、PA5 が黒、PA6 が青、PA7 が赤に対応（入力用）、PA0〜3 は 7 セグメント LED（出力用）に対応
    PADDR = 0x0F; //  上位 4 ビット(PA4-PA7)を入力、下位 4 ビット(PA0-PA3)を出力に設定
}

//  車体を前進させる
void motor_fwd() {
    P4DR = 0x06 | (P4DR & 0xF0); // P42 と P41 を 1 にセット（0000 0110）
```

```
}

// 車体を停止させる
void motor_stop() {
    P4DR = P4DR & 0xF0; // 下位 4 ビットを全て 0 にして停止
}

// 車体を後退させる
void motor_back() {
    P4DR = 0x09 | (P4DR & 0xF0); // P43 と P40 を 1 にセット（0000 1001）
}

// 車体を右旋回させる
void motor_right() {
    P4DR = 0x02 | (P4DR & 0xF0); // P41 のみ 1 にセット（0000 0010）
}

// 車体を左旋回させる
void motor_left() {
    P4DR = 0x04 | (P4DR & 0xF0); // P42 のみ 1 にセット（0000 0100）
}

int main(void) {
    int flag = 0; // 0: 停止、1: 動作中
int adcdata = 0;
```

```
    init(); // 初期化


ADStart();
    while (1) {
        // ボタンの状態を確認
        if ((PADR & 0x40)==0x00) { // 青ボタン（PA6）が押されたとき
            flag = 1; // 動作開始
        }
        if ((PADR & 0x80)==0x00) { // 赤ボタン（PA7）が押されたとき
            flag = 0; // 動作停止
        }


ADCexec();
  adcdata = ADRead();//a/d 変換した値を保存


        // flag の状態に応じて動作を制御
        if (flag == 1) {
if(adcdata>180){
            motor_fwd(); // 前進
}
if(adcdata<180){//find black
            motor_stop(); // 停止
}
        }
```

```
if (flag == 0){
motor_stop(); //停止
}
      }

      return 0;
}
```

（２）ロータリーエンコーダーについて

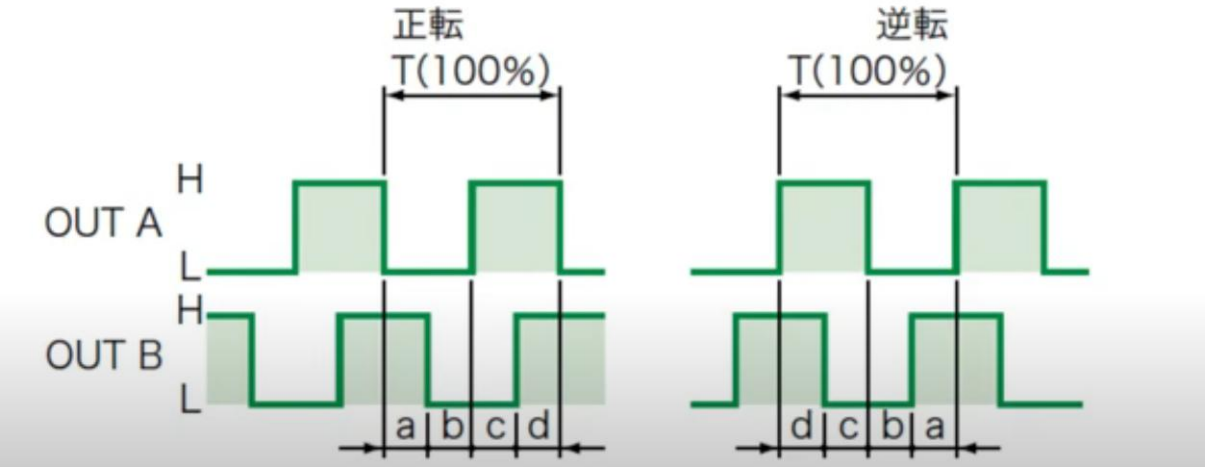

図１　ロータリーエンコーダーの出力形態。

ロータリーエンコーダーには２つの出力形態があり、その詳細を上の図１に示す。

７．参考文献

授業資料（担当：長峯）

YouTube 動画（https://youtu.be/MECn6SkGS2o）