作業報告書（2023 年 5 月 18 日）

J20413 北野正樹

【作業内容】

プログラムの内容の理解

どのようにセンサーとモーターを連携させるのかの動きを理解する

【作業項目】

① 光センサーのデータ取得について理解する

班員が光センサのデータ取得のヘッダーを作っていたのでそちらを流用する。

光センサーのヘッダーを以下に示す。

```
#ifndef stdio
#include<stdio.h>
#endif
#ifndef stdlib
#include<stdlib.h>
#endif
#ifndef wiringPi
#include "wiringPi.h"
#endif
#ifndef wiringPiI2C
#include "wiringPiI2C.h"
#endif
#define OS_L 5
#define OS_ML 6
#define OS_M 13
#define OS_MR 19
#define OS_R 26

int sensor(){
	int bi = 0b00000;
	if(digitalRead(OS_L)==1) bi |= (1<<0);//bit0 turns 1
	if(digitalRead(OS_ML)==1)bi |= (1<<1);//bit1 turns 1
	if(digitalRead(OS_M)==1) bi |= (1<<2);//bit2 turns 1
	if(digitalRead(OS_MR)==1)bi |= (1<<3);//bit3 turns 1
	if(digitalRead(OS_R)==1) bi |= (1<<4);//bit4 turns 1
	return bi;
}
```

このヘッダには光センサのピン番号と光センサから読み取った値をバイナリデータとして返すヘッダーだ。

作業報告書（2023 年 5 月 18 日）

J20413 北野正樹

② モータ駆動のヘッダーについて理解する

こちらも班員がモーター駆動のヘッダーを作っていただいたのでそちらを流用する。

モーター駆動のヘッダーを以下に示す。

```
#ifndef stdio
#include<stdio.h>
#endif
#ifndef stdlib
#include<stdlib.h>
#endif
#ifndef wiringPi
#include "wiringPi.h"
#endif
#ifndef wiringPiI2C
#include "wiringPiI2C.h"
#endif
// 以下、定数宣言です
// PWM ユニットの I2C アドレス
// i2cdetect で確認可能、違っていたら修正して下さい

#define PWMI2CADR 0x40

// モータードライバの各入力が接続されている PWM ユニットのチャネル番号
// 右側のモーター：パワーユニットの K1 または K2 に接続（説明書は誤り）
// ENA は PWM 駆動に使う（1 でブリッジ動作、0 はブリッジオフ）
// IN1 と IN2 は右車輪の回転方向を決める（後進：0,1、前進：1,0）（0,0 と 1,1 はブレーキ）
#define ENA_PWM 8
#define IN1_PWM 9
#define IN2_PWM 10
// 左側のモーター：パワーユニットの K3 または K4 に接続（説明書は誤り）
// ENB は PWM 駆動に使う（1 でブリッジ動作、0 はブリッジオフ）
// IN3 と IN4 は左車輪の回転方向を決める（後進：0,1、前進：1,0）（0,0 と 1,1 はブレーキ）
#define ENB_PWM 13
#define IN3_PWM 11
#define IN4_PWM 12 // PWM モジュールのレジスタ番号
#define PWM_MODE1 0
#define PWM_MODE2 1
#define PWM_SUBADR1 2
```

作業報告書（2023 年 5 月 18 日）

J20413 北野正樹

```
#define PWM_SUBADR2 3
#define PWM_SUBADR3 4
#define PWM_ALLCALL 5
// PWM 番号×4+PWM_0_??_? でレジスタ番号は求まる
#define PWM_0_ON_L 6
#define PWM_0_ON_H 7
#define PWM_0_OFF_L 8
#define PWM_0_OFF_H 9
// PWM 出力定数
#define PWMFULLON 16
#define PWMFULLOFF 0
// プリスケーラのレジスタ番号
// PWM 周波数を決めるレジスタ番号、100Hz なら 61 をセット
#define PWM_PRESCALE 254

int set_pwm_output(int fd, int pwmch, int outval){
//motor_drive( )から呼ばれる関数、PWM ユニットへの書き込みをやっています
//直接、他から呼び出す必要はないと思われますが、必要ならどうぞ

   int ef = 0;
   int regno;
   if ((pwmch < 0) || (pwmch > 15)) ef = 1;// チャネルの指定違反チェック
   if ((outval < 0) || (outval > 16)) ef = ef + 2;// 出力値の指定違反チェック
   if (ef == 0){regno = PWM_0_ON_L + pwmch * 4; // 1ch あたり 4 レジスタで 16ch 分あるので
      if (outval == 16){
            wiringPiI2CWriteReg8(fd,regno+3,0);
            wiringPiI2CWriteReg8(fd,regno+1,0x10);
      } else {
            wiringPiI2CWriteReg8(fd,regno+1,0);
            wiringPiI2CWriteReg8(fd,regno+3,outval);
      }
   }
   return ef; // エラーがなければ 0 が返る
}

int motor_drive(int fd, int rm, int lm){ // モーターを制御するための関数
// fd は I2C 初期化時のファイルディスクプリタ（デバイス番号のようなもの）
```

作業報告書（2023 年 5 月 18 日）

J20413 北野正樹

```
// lm は左モーター、rm は右モーターの駆動数値で、-16～+16 の範囲で指定
// 負の場合は後方向に回転、正の場合は前方向に回転
// 絶対値が大きいほど、パワーが大きくなる
// PWM ユニット自体は 12 ビット精度だが、上位 4 ビット分を制御
// あまり細かく制御しても、ロボカーの動きとしては大差ないと考えられるため
// 必要と思うなら、自分でマニュアルを見てプログラムを書いて下さい
   set_pwm_output(fd, ENA_PWM, 0); // Right motor disable
   set_pwm_output(fd, ENB_PWM, 0); // Left motor disable
   // Right motor PWM control
   if (rm < 0){
      set_pwm_output(fd, IN1_PWM, 0); // OUT1->GND
      set_pwm_output(fd, IN2_PWM, 16); // OUT2->+Vs
      rm = abs(rm);
   } else {
      set_pwm_output(fd, IN1_PWM, 16); // OUT1->+Vs
      set_pwm_output(fd, IN2_PWM, 0); // OUT2->GND
   }
   // Left motor PWM control
   if (lm < 0){
      set_pwm_output(fd, IN3_PWM, 0); // OUT3->GND
      set_pwm_output(fd, IN4_PWM, 16); // OUT4->+Vs
      lm = abs(lm);
   } else {set_pwm_output(fd, IN3_PWM, 16); // OUT3->+Vs
      set_pwm_output(fd, IN4_PWM, 0); // OUT4->GND
   }
   if (lm > 16) lm = 16;
   if (rm > 16) rm = 16;
   set_pwm_output(fd, ENA_PWM, rm); // Right motor PWM start
   set_pwm_output(fd, ENB_PWM, lm); // Left motor PWM start
   return 0;// 戻り値は常に 0
}//この他に、プログラムの最初の方で以下の PWM ユニットの初期化が必要。

int motor_r(int fd, int rm){ // モーターを制御するための関数
// fd は I2C 初期化時のファイルディスクプリタ（デバイス番号のようなもの）
// lm は左モーター、rm は右モーターの駆動数値で、-16～+16 の範囲で指定
// 負の場合は後方向に回転、正の場合は前方向に回転
// 絶対値が大きいほど、パワーが大きくなる
```

作業報告書（2023 年 5 月 18 日）

J20413 北野正樹

```
// PWM ユニット自体は 12 ビット精度だが、上位 4 ビット分を制御
// あまり細かく制御しても、ロボカーの動きとしては大差ないと考えられるため
// 必要と思うなら、自分でマニュアルを見てプログラムを書いて下さい
   set_pwm_output(fd, ENA_PWM, 0); // Right motor disable
   // Right motor PWM control
   if (rm < 0){
      set_pwm_output(fd, IN1_PWM, 0); // OUT1->GND
      set_pwm_output(fd, IN2_PWM, 16); // OUT2->+Vs
      rm = abs(rm);
   } else {
      set_pwm_output(fd, IN1_PWM, 16); // OUT1->+Vs
      set_pwm_output(fd, IN2_PWM, 0); // OUT2->GND
   }   if (rm > 16) rm = 16;
   set_pwm_output(fd, ENA_PWM, rm); // Right motor PWM start
   return 0;// 戻り値は常に 0
}//この他に、プログラムの最初の方で以下の PWM ユニットの初期化が必要。

int motor_l(int fd, int lm){ // モーターを制御するための関数
// fd は I2C 初期化時のファイルディスクプリタ（デバイス番号のようなもの）
// lm は左モーター、rm は右モーターの駆動数値で、-16～+16 の範囲で指定
// 負の場合は後方向に回転、正の場合は前方向に回転
// 絶対値が大きいほど、パワーが大きくなる
// PWM ユニット自体は 12 ビット精度だが、上位 4 ビット分を制御
// あまり細かく制御しても、ロボカーの動きとしては大差ないと考えられるため
// 必要と思うなら、自分でマニュアルを見てプログラムを書いて下さい
   set_pwm_output(fd, ENB_PWM, 0); // Left motor disable
   // Left motor PWM control
   if (lm < 0){
      set_pwm_output(fd, IN3_PWM, 0); // OUT3->GND
      set_pwm_output(fd, IN4_PWM, 16); // OUT4->+Vs
      lm = abs(lm);
   } else {set_pwm_output(fd, IN3_PWM, 16); // OUT3->+Vs
      set_pwm_output(fd, IN4_PWM, 0); // OUT4->GND
   }
   if (lm > 16) lm = 16;
   set_pwm_output(fd, ENB_PWM, lm); // Left motor PWM start
   return 0;// 戻り値は常に 0
```

```
}//この他に、プログラムの最初の方で以下の PWM ユニットの初期化が必要。

int motor_init(){
   int fd;
   wiringPiSetupGpio(); /* BCM_GPIO ピン番号で指定*/
   fd = wiringPiI2CSetup(PWMI2CADR); //  この fd がファイルディスクプリタ
   if (fd < 0){
      printf("I2C の初期化に失敗しました。終了します。¥n");
      exit(EXIT_FAILURE);
   }
   wiringPiI2CWriteReg8(fd,PWM_PRESCALE,61);      //PWM 周期 10ms に設定
   wiringPiI2CWriteReg8(fd,PWM_MODE1,0x10);      //SLEEPmode
   wiringPiI2CWriteReg8(fd,PWM_MODE1,0);     //NORMALmode
   delay(1); // wait for stabilizing internal oscillator
   wiringPiI2CWriteReg8(fd,PWM_MODE1,0x80);      //Restart all PWM ch
   return fd;
}
```

これは光センサーからデータを読み取り、モーターにしかるべき動きをするプログラムだ。
これらのプログラムを理解することができた。
私はこれらをヘッダーに分割するのは面倒なので、一つのファイルにしてみた。
自分で改良したプログラムを以下に示す。

```
#include <stdio.h>
#include <stdlib.h>
#include <wiringPi.h>
#include <wiringPiI2C.h>

// PWM ユニットの I2C アドレス
#define PWMI2CADR 0x40

// PWM 制御に使う。（1 でブリッジ動作、0 はブリッジオフ）
#define ENA_PWM 8

// IN1 と IN2 は右車輪の回転方向を決める（後進：0, 1,  前進：1, 0）（0, 0 と 1, 1 はブレーキ）
#define IN1_PWM 9
#define IN2_PWM 10
```

作業報告書（2023 年 5 月 18 日）

J20413 北野正樹

```
// 左側のモーター：パワーユニットの K3 または K4 に接続
// ENB は PWM 駆動に使う（1 でブリッジ動作、0 はブリッジオフ）
#define ENB_PWM 13

// IN3 と IN4 は左車輪の回転方向を決める（後進：0, 1、前進：1, 0）（0, 0 と 1, 1 はブレーキ）
#define IN3_PWM 11
#define IN4_PWM 12

// PWM モジュールのレジスタ番号
#define PWM_MODE1 0
#define PWM_MODE2 1
#define PWM_SUBADR1 2
#define PWM_SUBADR2 3
#define PWM_SUBADR3 4
#define PWM_ALLCALL 5

// PWM 番号＊4＋PWM_0_??_?でレジスタ番号は求まる
#define PWM_0_ON_L 6
#define PWM_0_ON_H 7
#define PWM_0_OFF_L 8
#define PWM_0_OFF_H 9

// PWM 出力定数
#define PWMFULLON 16
#define PWMFULLOFF 0

// プリスケーラのレジスタ番号
// PWM 周波数を決めるレジスタ番号、100Hz なら 61 をセット
#define PWM_PRESCALE 254

// 光センサーのピン番号
#define GPIO_L 5
#define GPIO_ML 6
#define GPIO_M 13
#define GPIO_MR 19
#define GPIO_R 26
```

```
// motor_drive()から呼ばれる関数、PWM ユニットへの書き込みをする。
int set_pwm_output(int fd, int pwmch, int outval) {
   int ef = 0;
   int regno;

   if ((pwmch < 0) || (pwmch > 15)) ef = 1; // チャネルの指定違反チェック
   if ((outval < 0) || (outval > 16)) ef = ef + 2; // 出力値の指定違反チェック
   if (ef == 0) {
      regno = PWM_0_ON_L + pwmch * 4; // 1ch あたり 4 レジスタで 16ch 分あるので
      if (outval == 16) {
         wiringPiI2CWriteReg8(fd, regno + 3, 0);
         wiringPiI2CWriteReg8(fd, regno + 1, 0x10);
      } else {
         wiringPiI2CWriteReg8(fd, regno + 1, 0);
         wiringPiI2CWriteReg8(fd, regno + 3, outval);
      }
   }
   return ef; // エラーがなければ 0 が返る
}



// モーターを制御するための関数。
// fd は I2C 初期化時のファイルディスクリプタ（デバイス番号のようなもの）
// lm は左モーター、rm は右モーターの駆動数値で、-16~+16 の範囲で指定
// 負の場合は後ろ方向に回転、生の場合は前方向に回転
// 全体値が大きいほど、パワーが大きくなる
// PWM ユニット自体は 12 ビット制度だが、上位 4 ビット分を制御
int motor_drive(int fd, int lm, int rm) {
   set_pwm_output(fd, ENA_PWM, 0); // 右のモーター無効化
   set_pwm_output(fd, ENB_PWM, 0); // 左のモーター有効化
   // 右モーターの制御
   if (rm < 0) {
      set_pwm_output(fd, IN1_PWM, 0); // OUT -> GND
      set_pwm_output(fd, IN2_PWM, rm); // OUT2 -> +Vs
      rm = abs(rm);
   } else {
      set_pwm_output(fd, IN1_PWM, rm); // OUT1 -> +Vs
```

```
    set_pwm_output(fd, IN2_PWM, 0);   // OUT2 -> GND
  }

  // 左モーターの制御
  if (lm < 0) {
    set_pwm_output(fd, IN3_PWM, 0); // OUT3 -> GND
    set_pwm_output(fd, IN4_PWM, lm); // OUT -> +Vs
    lm = abs(lm);
  } else {
    set_pwm_output(fd, IN3_PWM, lm); // OUT3 -> +Vs
    set_pwm_output(fd, IN4_PWM, 0); // OUT4 -> GND
  }
  if (lm > 16) lm = 16;
  if (rm > 16) rm = 16;
  set_pwm_output(fd, ENA_PWM, rm); // 右モータースタート
  set_pwm_output(fd, ENB_PWM, lm); // 左モータースタート
  return 0;
}

int main() {
  int fd;
  wiringPiSetupGpio();
  fd = wiringPiI2CSetup(PWMI2CADR);
  if (fd < 0) {
    printf("I2C の初期化に失敗しました。終了します。¥n");
    exit(EXIT_FAILURE);
  }
  wiringPiI2CWriteReg8(fd, PWM_PRESCALE, 61);
  wiringPiI2CWriteReg8(fd, PWM_MODE1, 0x10);
  wiringPiI2CWriteReg8(fd, PWM_MODE1, 0);
  delay(1);
  wiringPiI2CWriteReg8(fd, PWM_MODE1, 0x80);

  while(1){
    if(digitalRead(GPIO_L) == LOW && digitalRead(GPIO_R) == LOW) break;
  }
```

```
    int ms, ls, rs;
    while (1) {
        ms = 0;
        ls = 0;
        rs = 0;
        if (digitalRead(GPIO_L) == HIGH) {
            printf("right¥n");
            rs = 5;
        }
        else if (digitalRead(GPIO_ML) == HIGH) {
            printf("middle right¥n");
            rs = 3; ms=2;
        }
        else if (digitalRead(GPIO_M) == HIGH) {
            printf("middle¥n");
            ms = 6;
        }
        else if (digitalRead(GPIO_MR) == HIGH) {
            printf("middle left¥n");
            ls = 3; ms=2;
        }
        else if (digitalRead(GPIO_R) == HIGH) {
            printf("left¥n");
            ls = 5;
        }
        else {
            printf("not_read¥n");
            ls=6;
        }
        motor_drive(fd, ms+ls, ms+rs);
        delay(50);
    }
    return 0;
}
```

作業報告書（2023 年 5 月 18 日）

J20413 北野正樹

【作業時間】

・作業時間：90 分

・報告書作成時間：30 分