【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。

1. 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  #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）  // 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  // 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  // 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;  } |

【作業時間】

・作業時間：90分

・報告書作成時間：30分