【作業内容】

プログラムの概要の理解

前へ進むプログラムの作成

【作業項目】

1. 本日作成したプログラムを下記に示す

|  |
| --- |
| #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\_1 5  #define GPIO\_2 6  #define GPIO\_3 13  #define GPIO\_4 19  #define GPIO\_5 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, lm); // 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\_1) == LOW) {  motor\_drive(fd, 0, 0);  return 0;  }  if (digitalRead(GPIO\_3) == LOW) {  motor\_drive(fd, 10, 10);  } else {  motor\_drive(fd, 0, 0);  }  }  return 0;  } |

以上のプログラムで前へ前進させることができる。Motor\_drive関数に出力を入れると左右のモータを制御できる。

【作業時間】

・作業時間：90分

・報告書作成時間：10分