wiringCPI

KULLANIM KILAVUZU

Crocus PI wiring C/C++ kütüphanesi

v1.1

İçindekiler

1- Revizyonlar

Tarih	Version	Açıklama
02.03.2023	v1.0	Başlangıç versiyonu
17.04.2023	v1.1	Pwm fonksiyonları güncellendi. Bug giderildi.

Örneklerle anlatım için : https://www.youtube.com/@crocuspi

2- Gereksinimler

wiringCPI ve gpiod kütüphaneleri sisteminizde kurulu olmalıdır. <wiringCPI.h> kütüphanesini yazılımınıza eklemelisiniz. Derleme işlemlerinde link olarak **-lwiringCPI -lpthread -lgpiod** parametrelerini eklemelisiniz. Ortam kurulumu eğitim videoları ve örnek projeler için youtube kanalını inceleyiniz.

3- GPIO Kütüphanesi

int gpioRead (int pin) ;

Giriş pininin durumunu okur.

int gpioWrite (int pin, int value) ;

Çıkış pininin durumunu değiştirir. Value lojik 1 veya 0 olabilir.

4- Seri Port Kütüphanesi

Belirtilen seri cihaz için dosya işlemleri gerçekleştirerek seri portun kullanımını sağlar.

Kullanmak için aşağıdaki kodu programınıza eklemeniz gerekir:

```
#include <wiringSerial.h>
```

Fonksiyonlar:

int serialOpen (char *device, int baud) ;

Seri portu kullanım için açar ve iletişim hızını ayarlar. Okuma için zaman aşımı(timeout) süresi 10 saniyedir. Fonksiyon açılan cihaz için oluşturulan dosya işaretçisini döner veya hata durumunda -1 değerini döner, errno değişkenine uygun hata değeri atanır. Hatanın sebebini öğrenmek için errno global değişken değerine bakılmalı.

void serialClose (int fd) ;

Dosya işaretçisi ile tanımlanmış cihazı kapatır.

void serialPutchar (int fd, unsigned char c) ;

Dosya işaretçisi ile tanımlanan cihaza bir byte veri gönderir.

void serialPuts (int fd, char *s) ;

Dosya işaretçisi ile tanımlanan cihaza null karakteri ile biten karakter dizisini(string) gönderir.

void serialPrintf (int fd, char *message, ...) ;

“printf” fonksiyonunu seri port için simüle eder.

int serialDataAvail (int fd) ;

Okunabilecek karakter sayısını döner veya hata durumunda -1 değerini döner, errno değişkenine uygun hata değeri atanır.

int serialGetchar (int fd) ;

Seri porttan gelen bir sonraki karakterin değerini döner. Bu fonksiyon veri olmadığı durumda 10 saniyeye kadar portu bloke eder, bu durumda -1 değerini döner.

void serialFlush (int fd) ;

İşaretçi ile belirtilen cihaza gelmiş ve gönderimek üzere bekleyen tüm veriyi temizler.

Not: Dosya işaretçisi (**fd**) standard Linux dosya işaretçisidir. read(),write() vb. gibi sistem fonksiyonlarını ihtiyaç halinde kullanılabilir. Örneğin büyük blok halinde ikili veri göndermek istediğiniz serialPutchar() veya serialPuts() fonksiyonlarının yeteri kadar verimli olmadığı durumlarda write() fonksiyonu kullanılabilir.

a) İleri Düzey Seri Port Kontrolü

İleri düzey port kontrolü yapmak gerektiğinde örneğin parity kontrolü, bit sayısı gibi parametrelerde değişiklik yapmak isterseniz, aşağıdaki gibi geleneksel "termios" kütüphanesini kullanabilirsiniz.

Programınıza kütüphaneyi ekleyin:

```
#include <termios.h>
```

Aşağıdaki gibi parametreleri, ayarlayabilirsiniz:

```
  struct termios options ;

  tcgetattr (fd, &options) ;          // Mevcut ayarları oku
  options.c_cflag &= ~CSIZE ;         // veri boyutu
  options.c_cflag |= CS7 ;            // Veri çerçeve boyutu(7 bit)
  options.c_cflag |= PARENB ;         // Parity aktif – varsayılan olarak çift
  tcsetattr (fd, &options) ;          // Yeni ayarları uygula
```

Burada "**fd**" dosya işaretçisi seri portu açarken kullandığımız serialOpen() fonksiyonun dönüş değeridir.

Ayarlayabileceğiniz parametrelerin tümünü görmek için "tcgetattr" kütüphanesinin kullanım sayfalarını inceleyebilirsiniz .

```
man tcgetattr
```

5- RS485 Port Kütüphanesi

Belirtilen seri cihaza bağlı RS485 modülünü dosya işlemleri gerçekleştirerek kullanımını sağlar.

Kullanmak için aşağıdaki kodu programınıza eklemeniz gerekir:

```
#include <wiringRS485.h>
```

Fonksiyonlar:

int serialRS485Open (char *device, int baud) ;

Seri portu kullanım için açar ve iletişim hızını ayarlar. Okuma için zaman aşım süresi 10 saniyedir. Fonksiyon açılan cihaz için oluşturulan dosya işaretçisini döner veya hata durumunda -1 değerini döner, errno değişkenine uygun hata değeri atanır. Hatanın sebebini öğrenmek için errno global değişken değerine bakılmalı.

void serialRS485Close (int fd) ;

Dosya işaretçisi ile tanımlanmış cihazı kapatır.

void serialRS485Putchar (int fd, unsigned char c) ;

Dosya işaretçisi ile tanımlanan cihaza bir byte veri gönderir.

void serialRS485Puts (int fd, char *s) ;

Dosya işaretçisi ile tanımlanan cihaza null karakteri ile biten string gönderir.

void serialRS485Printf (int fd, char *message, ...) ;

“printf” fonksiyonunu seri port için simüle eder.

int serialRS485DataAvail (int fd) ;

Okunabilecek karakter sayısını döner veya hata durumunda -1 değerini döner, errno değişkenine uygun hata değeri atanır.

int serialRS485Getchar (int fd) ;

Seri porttan gelen bir sonraki karakterin değerini döner. Bu fonksiyon veri olmadığı durumda 10 saniyeye kadar portu bloke eder, bu durumda -1 değerini döner.

void serialRS485Flush (int fd) ;

İşaretçi ile belirtilen cihaza gelmiş ve gönderimek üzere bekleyen tüm veriyi temizler.

void serialRS485Direction (int direction) ;

Veri alma veya gönderme modunu ayarlar.

Not: Dosya işaretçisi (**fd**) standard Linux dosya işaretçisidir. read(),write() vb. gibi sistem fonksiyonlarını ihtiyaç halinde kullanabilirsiniz. Örneğin büyük blok halinde ikili veri göndermek istediğiniz durumda serialPutchar() veya serialPuts() fonksiyonlarının yeteri kadar verimli olmadığı durumlarda write() fonksiyonu kullanılabilir.

6- SPI Library

Kullanmak için aşağıdaki kodu programınıza eklemeniz gerekir:

```
#include <wiringSPI.h>
```

Fonksiyonlar:

int wiringCPISPISetup (int channel, int speed) ;

Belirtilen kanalı istenilen hızda haberleşme için kurulumunu gerçekleştirir. Hız 500000 ile 50000000 arasında olabilir.

Fonksiyon açılan cihaz için oluşturulan dosya işaretçisini döner veya hata durumunda -1 değerini döner, errno değişkenine uygun hata değeri atanır. Hatanın sebebini öğrenmek için errno global değişken değerine bakılmalı.

int wiringCPISPIDataRW (int channel, unsigned char *data, int len) ;

Belirtilen SPI kanalında eş zamanlı hem okuma hem de yazma işlemi gerçekleştirilir. Gelen veri tampon data üzerine yazılacaktır.

7- I2C Kütüphanesi

Kullanmak için aşağıdaki kodu programınıza eklemeniz gerekir:

```
#include <wiringI2C.h>
```

i2c cihazını standart sistem komutları kullanarak kontrol edebilirsiniz.

```
i2cdetect -y 0
```

Fonksiyonlar:

int wiringCPII2CSetup (int devId,int channel) ;

Belirtilen kanal ve cihaz id için i2c kurulumu gerçekleştirilir. i2cdetect programını kullanılarak her bir i2c cihazı için id bilgisini öğrenebilirsiniz. Kanal sıfır için “/dev/i2c-0”, kanal bir için “/dev/i2c-1” kullanılacaktır.

Fonksiyon, açılan cihaz için oluşturulan dosya işaretçisini döner veya hata durumunda -1 değerini döner, errno değişkenine uygun hata değeri atanır. Hatanın sebebini öğrenmek için errno global değişken değerine bakılmalı.

Örneğin: ID si 0x21 olan bir cihaz , kanal 0 ‘a bağlı ise kullanım:

```
wiringCPII2CSetup(0x20,0) ;
```

şeklinde olmalıdır.

Aşağıdaki fonksiyonların tamamında hata durumunda dönüş değeri negatiftir.

int wiringCPII2CRead (int fd) ;

Bazı cihazlar adres(register) bilgisi gönderilmeden okunabilmektedir. Bu tip cihazlardan veri okur.

int wiringCPII2CWrite (int fd, int data) ;

Bazı cihazlar adres(register) bilgisi gönderilmeden veri yazılabilmektedir. Bu tip cihazlara veri yazar.

int wiringCPII2CWriteReg8 (int fd, int reg, int data) ;
Belirtilen adres(register) bilgisine 8 bitlik veri yazar.

int wiringCPII2CWriteReg16 (int fd, int reg, int data) ;
Belirtilen adres(register) bilgisine 16 bitlik veri yazar.

int wiringCPII2CReadReg8 (int fd, int reg) ;
Belirtilen adresten(register) 8 bitlik veri okur.

int wiringCPII2CReadReg16 (int fd, int reg) ;
Belirtilen adresten(register) 16 bitlik veri okur.

8- PWM Kütüphanesi

Donanımsal PWM modülünü kullanmak için kullanılır. Fonksiyonlar başarılı ise 0 değeri döner. Hata durumunda farklı bir değer döner. Hatanın sebebini öğrenmek için errno global değişken değerine bakılmalı.

void pwmSetup (char *pwmchip, int id, int period, int duty) ;
PWM modül ayarlarını yapar. pwmchip cihaz adresini, id kanal numarasını, period nanosaniye biriminde periyodu, duty ise yüzde olarak duty-cycle gösterir.Frekans hesabını aşağıdaki formül ile gerçekleştirebilirsiniz.

$$f = \frac{1}{period * 10^{-9}}$$

1Mhz frekans, %50 duty için:
```
pwmSetup("pwmchip0",0,1000,50);
```

int pwmActivate (int pwm_id);
Belirtilen PWM kanalı aktif eder.

int pwmDeactivate (int pwm_id);
Belirtilen PWM kanalı pasif eder.

int pwmStart(int pwm_id);
Belirilen kanal için PWM sinyal çıkışını başlatır.

int pwmStop (int pwm_id);
Belirilen kanal için PWM sinyal çıkışını durdurur.

int pwmSetPeriod (int pwm_id,int period);
Belirilen kanal için PWM periyodunu ayarlar.

int pwmSetChipname(char *pwmchip);
Fonksiyonların arka planda çalışırken kullandığı pwmchip cihaz adını değiştirir.

int pwmPolarity(int pwm_id, int pol);

Belirilen kanal için polarityi ayarlar. PWM sinyali pol değişkeni ‘0’ için normal ‘1’ için sinyal invert edilir.

int pwmSetDuty (int pwm_id, int percentage);

PWM duty cycle ayarlar. percentage yüzde olarak duty değeridir. 0 ile 100 arasında bir değer verilmelidir.

9- Soft PWM Kütüphanesi

Herhangi bir GPIO pinini kullanarak PWM sinyali oluşturur. Soft olarak bu işlem gerçekleştirilir. Bu sebeple işlemci üzerinde bir yük oluşturacaktır.

Kullanmak için aşağıdaki kodu programınıza eklemeniz gerekir:

```
#include <softPwm.h>
```

int softPwmCreate (int pin, int initialValue, int pwmRange) ;

Belirtilen pinde PWM sinyali oluşturur. pwmRange * 100 uS periyot süresidir. 10 Hz için range = 1000 olmalıdır. Range değeri azaldıkça frekans artar ,hassasiyet düşer ve işlemci kullanımı artar. Yüksek frekanslarda pwm için donanım kullanınız. initialValue ilk sinyal üretiminde lojik 1 in başlama zamanını belirler.

Fonksiyon başarılı ise 0 değeri döner. Hata durumunda farklı bir değer döner. Hatanın sebebini öğrenmek için errno global değişken değerine bakılmalı.

void softPwmDuty (int pin, int percentage);

Belirtilen GPIO pin için duty-cycle (lojik 1 olma oranı) ayarlar. percentage yüzde değişkeni 0 ile 100 arasında girilebilir.

void softPwmWrite (int pin, int value) ;

PWM değerini ayarlar. (range – value) lojik bir olma süresini belirler. Value değeri 0 – max range kadar olabilir.

Not:

- Her pin için yaklaşık %2 oranında işlemci tüketilir.(10 Hz için)
- Her pin için yaklaşık %16 oranında işlemci tüketilir (100 Hz için)
- PWM çıkışını korumak için yazılımınız sürekli çalışmalıdır.

10- Zaman Fonksiyonları

void delay (unsigned int howLong)

Belirlenen milisaniye cinsinden değer kadar programı durdurur. Linux’un çoklu işlemlerinden dolayı farklılık gösterebilir. Maksimum değer unsigned 32-bit integer tipinde değer olabilir buda yaklaşık 49 güne karşılık gelir.

void delayMicroseconds (unsigned int howLong)

Belirlenen mikrosaniye cinsinden değer kadar programı durdurur. Linux'un çoklu işlemlerinden dolayı farklılık gösterebilir. Maksimum değer unsigned 32-bit integer tipinde değer olabilir buda yaklaşık 71 dakikaya karşılık gelir.

100 mikrosaniyenin altındaki değerlerde donanımsal döngü kullanılır, üstünde ise nanosleep() fonksiyonu kullanılır. Eğer thread kullanırsanız çok kısa gecikmelerin sistemin genel performansı üzerindeki etkilerini göz önünde bulundurmanız gerekebilir.

11- Lisans

wiringCPI kütüphanesi, libgpiod ve 2012 yılında Gordon Henderson (https://projects.drogon.net) tarafından yazılan desteği sona eren wiringPi kütüphanesinin Crocus PI tek kart bilgisayarlar için geliştirilmiş halidir. LGPL (GNU Lesser General Public License) lisansı altındadır.