# dsGPIO

## schneller und resourcenschonender Zugriff auf die GPIOs des Raspberry Pi ohne root-Berechtigung

| Datum | Version |
| --- | --- |
| 28.02.2019 | Erste Version |

# Inhaltsverzeichnis

# 1. Allgemeines

Für Linux gibt es eine Kernel-API namens GPIO UAPI mit der der Zugriff auf vorhandene GPIOs möglich ist. Eine ausführliche Dokumentation der GPIO UAPI scheint nicht zu existieren. Allerdings stellt die zugehörige Include-Datei eine gute Informationsquelle dar.
Die dsGPIO-Bibliothek stellt eine komfortable Möglichkeit des Zugriffs auf die GPIOs zur Verfügung und verwendet dafür die GPIO UAPI.
Im Gegensatz zu den versierten Bibliotheken für den GPIO Zugriff, wie z.B. pigpio, benötigen die Aufrufe der dsGPIO Funktionen keine root-Rechte zur Ausführungszeit.
Zudem ist es ohne Weiteres möglich mehrere Programme gleichzeitig laufen zu lassen, die Funktionen der dsGPIO-Library nutzen.

Allerdings ist der Funktionsumfang der dsGPIO-Bibliothek auf das Auslesen bzw. Setzen des Pin-Levels und die Abfrage eines Statuswechsels über einen Event-Handler beschränkt. Ausserdem werden derzeit nur die GPIOs der 40-poligen Pinleiste des Raspberry Pi unterstützt.
Von diesen Pins werden, unabhängig von der Version des Raspberry Pi, die physikalischen Pin-Nummern in ihre Broadcom-Nummern und umgekehrt mittels Tabelle gemapped. Das funktioniert so weit zwar gut, kann aber unter Umständen eine Fehlerquelle darstellen.

# 2. Datentypen und Strukturen

Zentrales Element der Library ist eine Tabelle zur Verwaltung der genutzten Pins und Callback-Funktionen.
Diese Tabelle vom Typ struct _bcm_pin_map hat folgenden Aufbau:

| | | |
|---|---|---|
| int | phys | Physikalische Pin-Nummer |
| uint8_t | bcm | Broadcom Nummer des Pins |
| int | fd | Linehandle des Pins |
| pthread_t* | pCallback | Ponter auf Thread-Struktur |
| struct _event_thread_arg* | pArgs | Pointer auf Thread-Argumente |

Die Struktur vom Typ struct _event_thread_arg, die die Argumente für den Callback-Thread beinhaltet, ist folgendermassen zusammengesetzt:

| | | |
|---|---|---|
| int | eventFlags | Auslösende Ereignisse des Callbacks |
| int | linefd | Linehandle zum GPIO |
| uint8_t | pin | Physikalische Pin-Nummer |
| pinCallback_t | callBack | Pointer auf die Callback-Funktion |
| void* | pUserData | Optionaler Pointer auf Anwenderdaten |

# 3. Funktionen

## 3.1. Interne Funktionen

### 3.1.1. static int mapFindBCM(uint8_t gpio)

| | | |
|---|---|---|
| Argumente | uint8_t gpio | BCM Nummer des gesuchten Pins |
| Beschreibung | Sucht in der Tabelle nach dem Eintrag, der zum Pin mit der angegebenen BCM Nummer gehört | |
| Rückgabewert | Der Index des gefundenen Eintrags oder DSGPIO_ERROR_NO_SUCH_BCM_PIN falls kein Eintrag gefunden wurde. | |

### 3.1.2. static void* eventThread(void* pArg)

| | | |
|---|---|---|
| Argumente | void* pArg | Zeiger auf die zugehörige struct _event_thread_arg Struktur. |
| Beschreibung | Diese Funktion wird als pthread gestartet, sobald eine Callback-Funktion für ein Event definiert wird. Sie prüft das Vorliegen des Events um dann die Callback-Funktion auszuführen. | |
| Rückgabewert | keiner | |

## 3.2. API-Funktionen

### 3.2.1. int pinLock(uint8_t pin, int mode)

| | | |
|---|---|---|
| Argumente | uint8_t pin | BCM Nummer des gewünschten Pins |
| | int mode | Festlegen des Pin-Modus als Ausgang (DSGPIO_PIN_MODE_OUTPUT) oder Eingang (DSGPIO_PIN_MODE_INPUT). |
| Beschreibung | Fordert über die GPIO UAPI einen Linehandle für den Pin an und initialisiert den zugehörigen Tabellen-Eintrag mit den entsprechenden Werten. | |
| Rückgabewert | Bei fehlerfreier Ausführung wird DSGPIO_ERROR_NO_ERROR zurückgegeben.<br>Andernfalls ist der Rückgabewert einer dieser Fehlercodes:<br>DSGPIO_ERROR_NO_SUCH_BCM_PIN: der angegebene Pin wurde nicht in der Tabelle gefunden.<br>DSGPIO_ERROR_GPIO_MODE: der angegebene Pin-Modus ist ungültig.<br>DSGPIO_ERROR_REQUEST_LINE_HANDLE: beim Anfordern des Linehandle trat ein Fehler auf.<br>DSGPIO_ERROR_OPEN_DEVICE: die zugehörige Gerätedatei konnte nicht geöffnet werden.<br>DSGPIO_ERROR_OUT_OF_MEMORY: es gab ein Problem mit dem verfügbaren Speicher.<br>DSGPIO_ERROR_HANDLE_IN_USE: der angegebene Pin ist bereits anderweitig in Verwendung. | |

### 3.2.2. int pinRelease(uint8_t pin)

| | | |
|---|---|---|
| Argumente | uint8_t pin | BCM Nummer des gewünschten Pins |
| Beschreibung | Setzt den zum übergebenen Pin zugehörigen Tabellen-Eintrag zurück und gibt den Linehandle auf den Pin wieder frei. | |
| Rückgabewert | Bei fehlerfreier Ausführung wird DSGPIO_ERROR_NO_ERROR zurückgegeben.<br>Andernfalls ist der Rückgabewert einer dieser Fehlercodes:<br>DSGPIO_ERROR_NO_SUCH_BCM_PIN: der angegebene Pin wurde nicht in der Tabelle gefunden.<br>DSGPIO_ERROR_PIN_NOT_LOCKED: der angegebene Pin ist nicht in Benutzung.<br>DSGPIO_ERROR_PIN_RELEASE: beim freigeben des Lineandles trat ein Fehler auf. | |

### 3.2.3. int pinState(uint8_t pin, uint8_t action, int state)

| | | |
|---|---|---|
| Argumente | uint8_t pin | BCM Nummer des gewünschten Pins |
| | uint8_t action | Aktion, die durchgeführt werden soll: DSGPIO_ACTION_GET_STATE zum Lesen oder DSGPIO_ACTION_SET_STATE zum Setzen des Status eines Pins |
| | int state | Nur beim Setzen des Status relevant: DSGPIO_PIN_STATE_HIGH oder DSGPIO_PIN_STATE_LOW |
| Beschreibung | Sucht den Tabellen-Eintrag für den gewünschten Pin.<br>Abhängig vom Parameter action wird dann der aktuelle Status des Pins gelesen (DSGPIO_ACTION_GET_STATE) oder der Pin auf den in state übergebenen Status gesetzt (DSGPIO_ACTION_SET_STATE).<br>Achtung! Der Rückgabewert ist abhängig von der ausgeführten Aktion. | |
| Rückgabewert | Bei fehlerfreier Ausführung wird beim Setzen des Status DSGPIO_ERROR_NO_ERROR, beim Lesen des Status DSGPIO_PIN_STATE_HIGH oder DSGPIO_PIN_STATE_LOW zurückgegeben.<br>Andernfalls ist der Rückgabewert einer dieser Fehlercodes:<br>DSGPIO_ERROR_NO_SUCH_BCM_PIN: der angegebene Pin wurde nicht in der Tabelle gefunden.<br>DSGPIO_ERROR_PIN_NOT_LOCKED: der angegebene Pin ist nicht in Benutzung.<br>DSGPIO_ERROR_GPIO_ACTION: das Argument für den Parameter action ist nicht gültig.<br>DSGPIO_ERROR_GPIO_STATE: der Wert des neuen Status beim Setzen ist nicht gültig.<br>DSGPIO_ERROR_SET_LINE_VALUES: beim Setzen des Pin-Status trat ein Fehler auf.<br>DSGPIO_ERROR_GET_LINE_VALUES: beim Lesen des Pin-Status trat ein Fehler auf. | |

### 3.2.4. int pinHandler(uint8_t pin, uint8_t action, int event, pinCallback_t cb, void* pData)

| | | |
|---|---|---|
| Argumente | uint8_t pin | BCM Nummer des gewünschten Pins |
| | uint8_t action | Aktion, die durchgeführt werden soll: DSGPIO_ACTION_SET_HANDLER |

| | | |
|---|---|---|
| | | zum Installieren oder DSGPIO_ACTION_CLEAR_HANDLER zum Löschen des Eventhandlers. |
| | int event | Auslösendes Event, GPIOEVENT_EVENT_RISING_EDGE für steigende, GPIOEVENT_EVENT_FALLING_EDGE für fallende oder eine Kombination daraus für beide Flanken. |
| | pinCallback_t cb | Zeiger auf die Funktion, die als Eventhandler installiert werden soll. |
| | void* pData | Optionaler Zeiger auf Benutzer-Daten. |
| Beschreibung | Installiert oder entfernt einen Eventhandler für einen Pin. Abhängig vom Parameter action wird dann die angegebene Funktion als Eventhandler installiert (DSGPIO_ACTION_SET_HANDLER) oder der Eventhandler gelöscht (DSGPIO_ACTION_CLEAR_HANDLER).<br>Achtung! Der Pin darf nicht bereits in Nutzung sein, wenn ein Eventhandler genutzt werden soll.<br>Beim Löschen des Eventhandlers wird der Pin freigegeben und muss für weitere Verwendung mit pinLock() erneut reserviert werden. | |
| Rückgabewert | Bei fehlerfreier Ausführung wird DSGPIO_ERROR_NO_ERROR zurückgegeben.<br>Andernfalls ist der Rückgabewert einer dieser Fehlercodes:<br>DSGPIO_ERROR_NO_SUCH_BCM_PIN: der angegebene Pin wurde nicht in der Tabelle gefunden.<br>DSGPIO_ERROR_GPIO_ACTION: der in action angegebene Parameter ist ungültig.<br>DSGPIO_ERROR_REQUEST_LINE_HANDLE: beim Anfordern des Linehandles ist ein Fehler aufgetreten.<br>DSGPIO_ERROR_OUT_OF_MEMORY: es gab ein Problem mit dem verfügbaren Speicher.<br>DSGPIO_ERROR_OPEN_DEVICE: beim Zugriff auf die Gerätedatei trat ein Fehler auf. | |

## 3.3. Beispiel für eine Callback-Funktion

### 3.3.1. void callBackFunc(uint8_t pin, struct gpioevent_data* event, void* pData)

| | | |
|---|---|---|
| Argumente | uint8_t pin | BCM Nummer des gewünschten Pins |
| | struct gpioevent_data* event | Zeiger auf Event-Informationen |
| | void* pData | Zeiger auf Benutzerdaten aus dem Programm (optional) oder NULL |
| Beschreibung | Diese Funktion wird per pinHandler() registriert und bei jedem Eintreten des definierten Events ausgeführt.<br>Die Parameter pin und event werden dabei mit Daten aus der GPIO UAPI gefüllt.<br>pData enthält den Zeiger auf die Benutzerdaten, der beim registrieren mit pinHandler() angegeben wurde, so dass Ämderungen an Daten des übergeordneten Programms möglich sind. | |
| Rückgabewert | Es wird kein Wert zurückgegeben. | |

# 4. Installation

Zur Installation wird auf dem Zielsystem der C-Compiler gcc benötigt. Der ist i.d.R. bereits installiert.
Um die statische und/oder dynamische Bibliothek libdsGPIO.a bzw. libdsGPIO.so zu erzeugen müssen zunächst die aktuellen Sourcen aus meinem Repository geholt werden.

Dazu das Repository mit git clonen

```
git clone https://github.com/dreamshader/dsGPIO
```

anschliessend in das build-Verzeichnis wechseln und sudo make install aufrufen:

```
cd build
sudo make install
```

Um die Bibliothek wieder vom System zu löschen im build-Verzeichnis sudo make uninstall aufrufen:

```
sudo make uninstall
```

# 5. Konstanten und Definitionen

```
#define DSGPIO_ERROR_NO_ERROR               0
#define DSGPIO_ERROR_NO_SUCH_BCM_PIN       -1
#define DSGPIO_ERROR_REQUEST_LINE_HANDLE   -2
#define DSGPIO_ERROR_HANDLE_IN_USE         -3
#define DSGPIO_ERROR_GPIO_MODE             -4
#define DSGPIO_ERROR_PIN_NOT_LOCKED        -5
#define DSGPIO_ERROR_PIN_RELEASE           -6
#define DSGPIO_ERROR_OPEN_DEVICE           -7
#define DSGPIO_ERROR_OUT_OF_MEMORY         -8
#define DSGPIO_ERROR_GPIO_STATE            -9
#define DSGPIO_ERROR_GPIO_ACTION          -10
#define DSGPIO_ERROR_SET_LINE_VALUES      -11
#define DSGPIO_ERROR_GET_LINE_VALUES      -12

#define DSGPIO_GPIODEV                     "gpiochip0"
#define DSGPIO_CONSUMER_LABEL              "dsGPIO"

#define DSGPIO_PIN_MODE_OUTPUT              1
#define DSGPIO_PIN_MODE_INPUT               2

#define DSGPIO_PIN_STATE_HIGH               1
#define DSGPIO_PIN_STATE_LOW                0
#define DSGPIO_PIN_STATE_NO_STATE          -1

#define DSGPIO_ACTION_SET_MODE             0b00000001
#define DSGPIO_ACTION_GET_MODE             0b00000010
#define DSGPIO_ACTION_SET_STATE            0b00000100
#define DSGPIO_ACTION_GET_STATE            0b00001000
#define DSGPIO_ACTION_SET_HANDLER          0b00010000
#define DSGPIO_ACTION_CLEAR_HANDLER        0b00100000


typedef void (*pinCallback_t) (uint8_t pin, struct
gpioevent_data* event, void* pData);
```

# 6. Das Beispiel-Programm

```
#define MYPIN 18

int main( int argc, char* argv[] )
{
   int exitCode = 0;
   uint8_t pin = MYPIN;

   if( (exitCode = pinLock( pin, DSGPIO_PIN_MODE_OUTPUT )) >= 0 )
   {

      if( exitCode = pinState( pin, DSGPIO_ACTION_SET_STATE,
                               DSGPIO_PIN_STATE_HIGH ) >= 0 )
      {

         exitCode = pinState( pin, DSGPIO_ACTION_GET_STATE, 0 );
         printf("set state: %d\n", exitCode );
      }
      else
      {
         fprintf(stderr, "set state failed!\n");
      }

      pinRelease( pin );

      exitCode =  pinHandler( pin, DSGPIO_ACTION_SET_HANDLER,
                              GPIOEVENT_REQUEST_RISING_EDGE |
                              GPIOEVENT_REQUEST_FALLING_EDGE,
                              &callBackFunc, NULL );
      sleep(8);

      exitCode = pinHandler(pin, DSGPIO_ACTION_CLEAR_HANDLER, 0,
                            NULL, NULL);

      printf("handler cleared[%d] ...\n", exitCode);
      sleep(3);

      pinRelease( pin );

   }
   else
   {
      fprintf(stderr, "lock failed!\n");
   }

   printf("ends with exitcode %d\n", exitCode);
   return( exitCode );
}
```

# 7. Externe Referenzen

Eine gute Informationsquelle zur GPIO UAPI ist die zugehörige Include-Datei: https://github.com/torvalds/linux/blob/master/include/uapi/linux/gpio.h.

Alle zugehörigen Source-Dateien inkl. Makefile stehen in meinem offenen Online-Repository zum kostenlosen Download bereit: https://github.com/dreamshader/dsGPIO.