GPIO auf dem BeagleBoneBlack in C++

Viktor Winkelmann & Oliver Niebsch

Basics: Benutzen eines GPIO Pins

1. Gewünschten Pin exportieren
2. Pin Modus einstellen
   - Richtung ("in" oder "out")
   - Bei Input: Pullup oder Pulldown
3. Value-Datei lesen oder schreiben
   - Eventuell danach wieder schließen

Variante 1a: FileSystem

```
#include <fstream>

void writeToPin(int gpioPin, int value) {
      fstream fileStream;
      string gpioPath;

      // Pin exportieren
      fileStream.open("/sys/class/gpio/export", fstream::out);
      fileStream << gpioPin; fileStream.close();
      gpioPath = "/sys/class/gpio/gpio" + to_string(gpioPin);

      // Richtung auf Output setzen
      fileStream.open((gpioPath + "/direction").c_str(), fstream::out);
      fileStream << "out"; fileStream.close();

      // Wert setzen
      fileStream.open((gpioPath + "/value").c_str(), fstream::out);
      fileStream << value; fileStream.close();
}
```

Variante 1b: FileSystem

```
#include <string>
#include <unistd.h>
#include <fcntl.h>

void writeToPin(int gpioPin, int value) {
      int fileDescriptor, length;
      char buf[42];
      string gpioPath;

      // Pin exportieren
      fileDescriptor = open("/sys/class/gpio/export", O_WRONLY);
      length = snprintf(buf, sizeof(buf), "%d", gpioPin);
      write(fileDescriptor, buf, length);
      close(fileDescriptor);
      gpioPath = "/sys/class/gpio/gpio" + to_string(gpioPin);

      // Richtung auf Output setzen
      fileDescriptor = open(gpioPath + "/direction", O_WRONLY);
      length = snprintf(buf, sizeof(buf), "out");
      write(fileDescriptor, buf, length);
      close(fileDescriptor);

      // Wert setzen
      fileDescriptor = open(gpioPath + "/value", O_WRONLY);
      length = snprintf(buf, sizeof(buf), "%d", value);
      write(fileDescriptor, buf, length); close(fileDescriptor);
}
```

Variante 2: Memory Zugriff

```
#include <sys/mman.h>
#include <unistd.h>
#include <fcntl.h>

// hier Konstanten für Speicheradressen etc.
const unsigned int GPIO1_START_ADDR = 0x4804C000;
const unsigned int GPIO1_END_ADDR = 0x4804CFFF;
const unsigned int GPIO_OUT_CONF = 0x14d;
const unsigned int GPIO_CLEARDATAOUT = 0x64;
const unsigned int GPIO_SETDATAOUT = 0x65;
const unsigned int PIN_P9_12 = 1 << 28;
// ... weitere Konstanten

void writeToPinP9_12(int value) {
      volatile unsigned int *gpio_addr = NULL;
      int fileDescriptor = open("/dev/mem", O_RDWR);

      // Speicherereich für GPIO1 Controller mappen
      gpio_addr = (volatile unsigned int *) mmap(0, GPIO1_END_ADDR - GPIO1_START_ADDR, PROT_READ |
      PROT_WRITE, MAP_SHARED, fileDescriptor, GPIO1_START_ADDR);

      // Richtung auf Output setzen
      *(gpio_addr + GPIO_OUT_CONF) &= ~PIN_P9_12;

      // Wert setzen
      if (value > 0)
            *(gpio_addr + GPIO_SETDATAOUT) = PIN_P9_12;
      else
            *(gpio_addr + GPIO_CLEARDATAOUT) = PIN_P9_12;
}
```

Variante 3: FileSystem mit mmap

```
#include <sys/mman.h>
#include <unistd.h>
#include <fcntl.h>
#include <stdio.h>
#include <string>
#include <sys/mman.h>

void writeToPin(int gpioPin, int value) {
      volatile char *pinValue;
      int fileDescriptor, length;
      string gpioPath;

      // Pin exportieren
      gleiches Verfahren wie bei Variante 1

      gpioPath = "/sys/class/gpio/gpio" + to_string(gpioPin);

      // Richtung auf Output setzen
      gleiches Verfahren wie bei Variante 1

      // Wert setzen
      fileDescriptor = open((gpioPath + "/value").c_str(), O_RDWR);
      pinValue = (volatile char *) mmap(0, 4096, PROT_READ | PROT_WRITE, MAP_SHARED,
      fileDescriptor, 0);
      *pinValue = (value < 1) ? '0' : '1';
      close(fileDescriptor);
}
```