Systemtechnik

Hardwarenahe Programmierung
Protokoll 2: SD Card

Verfasser:

- **Jonas Ruhland**
- **Clemens Rumpfhuber**

Übungsleiter: DI Mag. Ferdinand Hell

SD Card am Raspberry Pi

Konfiguration des Raspberry Pis

SDIO-Einheit aktivieren

Um die sekundäre SDIO-Einheit auf die Raspberry Pi Pins zu aktivieren, muss folgendes in die Bootconfig geschrieben werden:

```
pi@raspi-it-01:~# nano /boot/config.txt
```

```
...
dtoverlay=sdio,poll_once=off,bus_width=1
...
```

Software installieren

Für das weitere Vorgehen, benötigt man den „Device Tree Compiler“ am Raspberry Pi. Um sicher zu gehen, dass die neueste Version installiert wird, laden wir uns die neuen Packetlisten herunter und installieren ausstehende Updates.

```
pi@raspi-it-01:~# sudo apt update

pi@raspi-it-01:~# sudo apt dist-upgrade -y

pi@raspi-it-01:~# sudo apt install device-tree-compiler -y
```

Device Tree konfigurieren

Es muss ein neues File für die SPI (Serial Peripheral Interface) Bus erstellt werden.

```
pi@raspi-it-01:~# nano mmc_spi.dts
```

```
/dts-v1/;
/plugin/;

/ {
   compatible = "brcm,bcm2835", "brcm,bcm2836", "brcm,bcm2708",
"brcm,bcm2709";

   fragment@0 {
      target = <&spi0>;
      frag0: __overlay__ {
         status = "okay";
         sd1 {
                reg = <0>;
                status = "okay";
                compatible = "spi,mmc_spi";
                voltage-ranges = <3000 3500>;
                spi-max-frequency = <8000000>;
         };
      };
   };
};
```

Um das File dem Bootloader vom Raspberry Pi zur Verfügung zu stellen, muss das File kompiliert werden.

```
pi@raspi-it-01:~# dtc -@ -I dts -O dtb -o mmc_spi.dtbo mmc_spi.dts
```

Das mit dem oben genannten Befehl erstellte Datei, muss in den Overlays Ordner des Bootloaders verschoben werden. Dazu werden Sudo- beziehungsweise Rootrechte benötigt.

```
pi@raspi-it-01:~# sudo cp mmc_spi.dtbo /boot/overlays/
```

Damit die kompilierte Datei vom Bootloader berücksichtigt wird, müssen noch zwei Konfigurationszeilen im Bootconfig-File hinzugefügt werden.

```
pi@raspi-it-01:~# sudo nano /boot/config.txt
──────────────────────────────────────────────────────────────────────────────


...
dtoverlay=mmc_spi
dtparam=spi=on
```

Nun kann der Raspberry Pi neugestartet werden.

```
pi@raspi-it-01:~# sudo shutdown -r now
```

Partition auf SD Karte erstellen

Zuerst kann man die vorhandenen Partitionen auflisten, um herauszufinden, welche Disk beziehungsweise Partition von uns formatiert wird.

```
root@raspi-it-01:~# lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
mmcblk0     179:0    0  7.4G  0 disk
├─mmcblk0p1 179:1    0  256M  0 part /boot
└─mmcblk0p2 179:2    0  7.1G  0 part /
mmcblk2     179:32   0 14.4G  0 disk
└─mmcblk2p1 179:33   0 14.4G  0 part
```

Die oben markierte Disk formatieren wir jetzt mit ext4.

```
root@raspi-it-01:~# mkfs.ext4 /dev/mmcblk2
mke2fs 1.44.5 (15-Dec-2018)
Found a dos partition table in /dev/mmcblk2
Proceed anyway? (y,N) y
Discarding device blocks: done
Creating filesystem with 3780608 4k blocks and 946560 inodes
Filesystem UUID: 4f893e48-7019-4870-a44f-1838fb3d5f7f
Superblock backups stored on blocks:
      32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information:
done
```

Einbinden des EXT4-Dateisystems

Zuerst muss das Zielverzeichnis erstellt werden und die benötigten Zugriffsrechte für den Benutzer „pi“ geben.

```
root@raspi-it-01:~# mkdir /media/sd0/

root@raspi-it-01:~# chmod 775 /media/sd0/

root@raspi-it-01:~# chown pi:pi /media/sd0/
```

Anschließend kann die Disk auf in das Verzeichnis gemountet werden.

```
root@raspi-it-01:~# mount -t ext4 /dev/mmcblk2 /media/sd0/ -o user=pi
```

Überprüfen der Konfiguration

Man kann sich nach der erfolgreichen Konfiguration, den Mountpoint anzeigen lassen.

```
root@raspi-it-01:~# lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
mmcblk0     179:0    0  7.4G  0 disk
├─mmcblk0p1 179:1    0  256M  0 part /boot
└─mmcblk0p2 179:2    0  7.1G  0 part /
mmcblk2     179:32   0 14.4G  0 disk /media/sd0
```

Praktische Anwendung

Sensordaten auslesen und auf SD Karte schreiben

```
#include <stdint-gcc.h>
#include <stdio.h>
#include <wiringPi.h>
#include <time.h>

#define DHTMAXTIMINGS 85
#define DHTPIN 11

FILE *logFile;

void setup() {
    // setup wiring pi
    wiringPiSetupPhys();
}

void openLogFile() {
    // open file in a (append) mode
    // => append text to file; when not exists, create file
    logFile = fopen ("/media/sd0/rumpfhuber.log", "a");
}

void closeLogFile() {
    fclose(logFile);
}

void readDHT(double *temperature, double *humidity, int *worked) {
    int dht11_dat[5] = {0, 0, 0, 0, 0};
    int dhtCounter = 0;

    while (*humidity == 0 && *temperature == 0 && dhtCounter < 5) {
        // predefine values
        uint8_t laststate = HIGH, counter, j = 0, i;
        dht11_dat[0] = dht11_dat[1] = dht11_dat[2] = dht11_dat[3] = dht11_dat[4] = 0;

        // set pin mode
        pinMode(DHTPIN, OUTPUT);
        digitalWrite(DHTPIN, LOW);
        delay(18);
        digitalWrite(DHTPIN, HIGH);
        delayMicroseconds(40);
        pinMode(DHTPIN, INPUT);

        // digital read
        for (i = 0; i < DHTMAXTIMINGS; i++) {
            counter = 0;
            while (digitalRead(DHTPIN) == laststate) {
                counter++;
                delayMicroseconds(1);
                if (counter == 255) { break; }
            }
            laststate = digitalRead(DHTPIN);

            if (counter == 255) break;

            if ((i >= 4) && (i % 2 == 0)) {
                dht11_dat[j / 8] <<= 1;
                if (counter > 50)
                    dht11_dat[j / 8] |= 1;
                j++;
            }
        }
```

```
        // calculate values
        if ((j >= 40) && (dht11_dat[4] == ((dht11_dat[0] + dht11_dat[1] +
                    dht11_dat[2] + dht11_dat[3]) & 0xFF))) {
            *temperature = dht11_dat[0] + (dht11_dat[1] / 10);
            *humidity = dht11_dat[2] + (dht11_dat[3] / 10);
            *worked = HIGH;
        } else
            *worked = LOW;

        dhtCounter++;
    }
}

void writeSensorDataToFile(unsigned int amount, unsigned int delayTime) {
    char formatedText[60], timeText[24];
    int worked = 0, i = 0;
    double temperature = 0, humidity = 0;

    while (i++ < amount) {
        // read sensor
        readDHT(&temperature, &humidity, &worked);

        // format time string
        time_t current_time = time(NULL);
        struct tm tm = *localtime(&current_time);
        strftime(timeText, sizeof timeText, "%Y-%d-%m %H:%M:%S:", &tm);

        // check if dht11 is working
        if (worked)
            sprintf(formatedText, "%s Temperature: %.2f Humidity: %.2f\n", timeText,
temperature, humidity);
        else
            sprintf(formatedText, "%s Sensor not working...\n", timeText);

        // write string into file
        fputs(formatedText, logFile);

        // delay
        delay(delayTime);
    }
}

int main() {
    setup();
    openLogFile();

    // check if file stream is successfully open
    if (logFile != NULL)
        writeSensorDataToFile(12, 10000);

    closeLogFile();
}
```

Inhalt auf der Log-Datei

```
pi@raspi-it-01:/media/sd0 $ cat rumpfhuber.log
2021-14-10 10:08:30: Sensor not working...
2021-14-10 10:08:40: Temperature: 24.00 Humidity: 22.00
2021-14-10 10:08:50: Temperature: 24.00 Humidity: 22.00
2021-14-10 10:09:00: Temperature: 24.00 Humidity: 22.00
2021-14-10 10:09:10: Temperature: 24.00 Humidity: 22.00
2021-14-10 10:09:20: Temperature: 24.00 Humidity: 22.00
2021-14-10 10:09:30: Temperature: 24.00 Humidity: 22.00
2021-14-10 10:09:40: Temperature: 24.00 Humidity: 22.00
2021-14-10 10:09:50: Temperature: 24.00 Humidity: 22.00
```

SD Karte am Arduino

Beim Arduino muss nur das richtige Platinen-Modul mit SD Kartenleser aufgesteckt werden. Dann kann direkt über eine eingebaute Library darauf geschrieben werden.

```
#include <SPI.h>
#include <SD.h>
#include <Wire.h>
#include <DS3231.h>
#include <LightDependentResistor.h>

#define SDCARD_PIN 10
#define SENSOR_PIN A1
#define BUTTON_PIN 2

#define OTHER_RESISTOR 3300 //ohms
#define USED_PHOTOCELL LightDependentResistor::GL5528

LightDependentResistor photocell(SENSOR_PIN, OTHER_RESISTOR, USED_PHOTOCELL);

DS3231 clock;
RTCDateTime dt;

File myFile;
char formatedString[50];
float sensorValue;

bool doing = 1;

unsigned long delayTime = 0;
unsigned long lastTime = 1500;

void sdCardSetup() {
  // Open serial communications and wait for port to open:
  Serial.begin(115200);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for native USB port only
  }

  Serial.println("Initializing SD card...");
  if (!SD.begin(SDCARD_PIN)) {
    Serial.println("Initialization failed!");
    while (1);
  }
  Serial.println("Initialization done.");

  // open the file. note that only one file can be open at a time,
  // so you have to close this one before opening another.
  myFile = SD.open("syt.csv", FILE_WRITE);
  Serial.println("Open File succeeded.");
}

void setup() {
  // setup sd card
  sdCardSetup();

  // set csv header
  myFile.println("Timestamp;Lux");

  // begin clock
  clock.begin();
}

void loop() {
  if (digitalRead(BUTTON_PIN)) {
      myFile.close();
      Serial.println("Eject SD Card...");
      doing = 0;
  }
```

```
  if(doing == 1) {
    if ((millis() - lastTime) > delayTime) {
      sensorValue = photocell.getCurrentLux();
      dt = clock.getDateTime();

      sprintf(formatedString, "%04d-%02d-%02d %02d-%02d-%02d;%i.%i",
        dt.year, dt.month, dt.day, dt.hour, dt.minute, dt.second, (int)sensorValue,
              (int)((sensorValue - (int)sensorValue) * 10)
      );

      Serial.println(formatedString);
      myFile.println(formatedString);

      lastTime = millis();
    }
  }
}


/*
Inhalt der CSV Datei:

Timestamp;Lux
2021-11-04 11-07-33;73.2
2021-11-04 11-07-35;64.7
2021-11-04 11-07-36;71.5
2021-11-04 11-07-38;3.5
2021-11-04 11-07-39;3.8
2021-11-04 11-07-41;58.9
2021-11-04 11-07-42;34.2
2021-11-04 11-07-44;71.5

*/
```