BEUTH HOCHSCHULE
FÜR TECHNIK
BERLIN
University of Applied Sciences

Dynamische Flüssigkeitssimulation auf einem RGB-Würfel

von

Kayoko Abe, Oleksandra Baga, Heiko Radde

EDV.Nr.:826058, 849852, 887027

Dokumentation des Projektes im Modul
‘Fortgeschrittene ARM Programmierung’
von Herrn Prof. Dr. Görlich

Tag der Abgabe 7. April 2019

Inhaltsverzeichnis

Abbildungsverzeichnis

Quelltextverzeichnis

Kapitel 1

Einführung

Im Rahmen des Moduls ‘Fortgeschrittene ARM Programmierung’ wird der Keil Evaluation Board MCB2388 verwenden, welcher einen NXP LPC2388 Microcontroller installiert und verschiedene Schnittstellen wie LCD, USB oder Ethernet besitzt. Der ARM7-Prozessor auf dem Board kann mit der bis zu seinesr höchsten Betriebsfrequenz von ?? MHz getaktet werden. Diese Merkmale lassen unterschiedliche Applikationen umsetzen.
Insbesondere unter Berücksichtigung von seiner hohen Taktrate sind wir zur Idee gekommen, Flüssigkeit zu simulieren und in 3 Dimension anhand eines aus 6 LED-Matrix Panels aufgebauten Würfel zu visualisieren. Zur Realisierung unseres Projekts sollte eine sehr hohe Taktrate benötigt werden, um die Bewegung der Wasserteilchen auszurechnen und diese auf den LED Panels in Echtzeit zu übertragen. Zusätzlich zu dem Entwicklungsboard sowie zu den Panels wird ein Accelerometer benötigt, um die 3-D Position des Würfels zu bestimmen und die Flüssigkeit darin entsprechend zu visualisieren.
Im folgenden Kapital wird die detaillierte Beschreibung jeder Komponente bezüglich der Theorie und der Implementierung vorgestellt.

Kapitel 2

Implementierung

2.1 Entwicklungsumgebung

Zur Entwicklung der Softwarekomponente soll erst Keil® Microcontroller Development Kit (MDK) auf einem Rechner installiert werden. Das Kit ist besonders gut geeignet für Arm®-basierte Microcontroller und beinhaltet alle notwendigen Komponente für Embedded Applikationen. Die Edition MDK-Lite ist konstenlos jedoch mit der Beschränkung von 32KBytes Codegröße verfügbar. Zusätzlich muss Legacy Package ARMv7 (NXP LPC 2388) installiert werden. Anschließend soll die Integrierte Entwicklungsumgebung Keil μversion 5 erfolgreich installiert sein.
Der Board kann durch einen USB-Anschluss vom Rechner eingeschaltet werden. Der ULINK-ME kann ebenfalls durch einen USB-Anschluss zum Laden des Programms und zum Debuggen verwendet werden.
Der Quellcode wurde auf C geschrieben und durch Doxygen dokumentiert.

2.2 Pinbelegung

Der Accelerometer und ein der 6 Panels werden durch freie GPIO-Pins mit dem Board verbunden. Abbildung 2.1 zeigt diese Pinbelegung. Es ist dabei zu beachten, die bereits vom Hersteller belegten Pins zu vermeiden, um den Board erfolgreich in Betrieb zu nehmen.
Die belegten Pins und die relevanten Ports 2 und 3 müssen wie Code **??** initialisiert werden, um diese als GPIO/Output-Pins verwenden zu können.

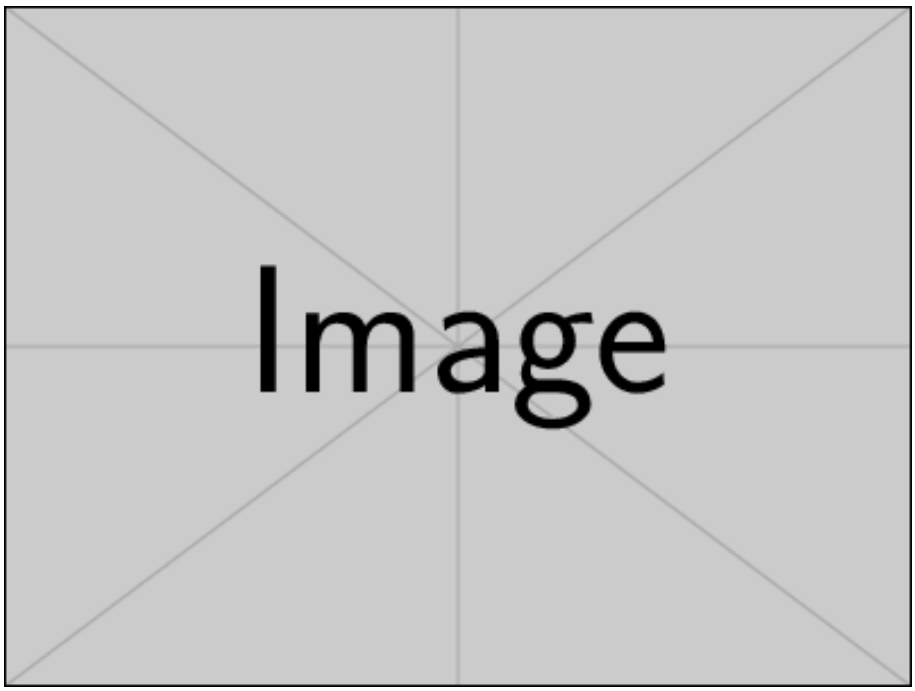

Abbildung 2.1: Pinbelegung

2.3 Flüssigkeitssimulation

2.4 Accelerometer

2.4.1 Hardware

Für die Ermittlung der Position des Würfels wurde ein digital Beschleunigungsmesser verwendet. Für ein Projekt standen zwei Beschleunigungsmesser zur Verfügung: BNO055 von Adafruit und ADXL345 von Analog Devices. Zwar ein Beschleunigungsmesser von Adafruit mächtiger ist und mehr Messungen erlaubt, das Projekt wurde am Ende unter der Verwendung des Beschleunigungsmessers ADXL345 geschrieben. Nach mehreren Versuchen, mit Adafruit zu arbeiten, wurde es festgestellt, dass Beschleunigungsmesser nicht arbeitsfähig ist. In alle Registern des Bausteins anstatt der vorprogrammierte Daten liegt nur die Zahl "0xe5", was laut der Hersteller zeigt, dass Registern irgendwie geleert wurden.

Für die Verwendung des ADXL345 sollte man in Programm nur die Adressen ändern und richtig alle Pins belegen. Die ALT ADDRESS wurde auf 1 gesetzt und die 7-Bit-I2C-Adresse für das Gerät wurde somit als "0x1D"festgelegt. In Programm wurde direkt dir entsprechende 8-Bir-I2C-Adresse "0x3A"verwendet und die notwendige Änderung der Adresse in "0x3Bäus "0x3A"wurde gerade in I2C Routine erledigt. So kann man unterscheiden, ob es Lesen oder Schreiben Operation durchführen sollen werden.

Es gibt keine internen Pull-Up- oder Pull-Down-Widerstände für nicht verwendete Pins. Daher gibt es keinen bekannten Status oder Standardstatus für den CS- oder ALT ADDRESS-Pin, wenn er potentialfrei bleibt oder nicht verbunden ist. Es ist erforderlich, dass der CS-Pin an VDD angeschlossen ist und dass der ALT ADDRESS-Pin bei Verwendung von

I2C entweder an VDD oder GND angeschlossen ist (wir haben an VDD angeschlossen).

Die gemessene Daten werden aus die Register 0x32 bis Register 0x37 gelesen. Diese sechs Bytes (Register 0x32 bis Register 0x37) sind jeweils acht Bits und enthalten die Ausgangsdaten für jede Achse. Register 0x32 und Register 0x33 enthalten die Ausgangsdaten für die X-Achse, Register 0x34 und Register 0x35 die Ausgangsdaten für die Y-Achse und Register 0x36 und Register 0x37 die Ausgangsdaten für die Z-Achse.

Die Ausgangsdaten sind zwei Komplemente, mit DATAx0 als niedrigstwertigem Byte und DATAx1 als höchstwertigem Byte, wobei x X, Y oder Z darstellt. Um die Daten richtig zu lesen, wurde es ein Mehrbyte-Lesen von 2 Register durchgeführt und die Daten aus zwei Lesevorgängen nach dem Lesen als 16-Bit Ausgangsdaten dargestellt.

```
1  uint16_t i2c16bit = 0x00;
2  ReadLenght = 2;
3  GlobalI2CAddr = addr;
4  I2CMasterBuffer[0] = regs[0];
5  I2CMasterBuffer[1] = regs[1];
6  ...
7  // merge the data from two registers
8  i2c16bit = i2c16bit | I2CReadBuffer[1]; // [REG0, REG1]: REG1 as MSB
9  i2c16bit = i2c16bit << 8;
10
11 i2c16bit = i2c16bit | I2CReadBuffer[0]; // [REG0, REG1]: REG0 as LSB
```

2.4.2 I2C und Keil Board

Für unseres Projekt haben wir I2C Block auf dem Keil Board in Master-Modus programmiert. Mit Master-Sendermodus werden Daten vom Master (Keil Board) zum Slave (Beschleunigungsmesser) übertragen. So erlaubt uns den Beschleunigungsmesser zu initialisieren.

Im Master-Empfängermodus werden Daten von einem Slave-Sender empfangen. Nach der Initialisierung des Beschleunigungsmessers wird es immer weiter in diesem Modus gearbeitet, da wir nur ständig die ermittelte Position ablesen wollen.

Für unseres Projekt haben wir I2C1 Block gewählt. Mithilfe von Schaltbild haben wir festgestellt, welche Pins stehen uns zu Verfügung und sind mit anderen Peripheriefunktionen, die während unseres Projekt notwendig sein könnten, nicht belegt.

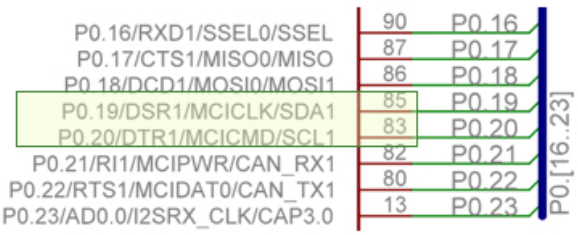

Abbildung 2.2: 2C1 Pin Belegung

Das PCONP-Register ermöglicht das Deaktivieren ausgewählter Peripheriefunktionen, um Energie zu sparen. Wenn ein Peripheriesteuerbit 1 ist, ist dieses Peripheriegerät aktiviert. Wenn ein Peripherie-Bit 0 ist, wird die Uhr des Peripheriegeräts deaktiviert, um Energie zu sparen. Dem I2C1 entspricht das Bit 19 in PCONP-Register. Um I2C1 zu aktivieren, muss man den Wert 2^{19} ins Register schreiben, was in Hexadezimal Format $0x00080000$ entspricht. Mit der Verwendung von logischen Operator OR wird die I2C1 aktiviert und die andere Peripheriefunktionen unverändert geblieben.

```
1 PCONP |= 0x00080000;
```

```
1 VICVectCntl19 = 0x0000001;   // select a priority slot for interrupt
2 VICVectAddr19 = (unsigned)i2c_irq; //pass the address of the IRQ
3 VICIntEnable |= 0x00080000; // enable interrupt
4 PINSEL1 |= 0x000003C0; //Switch GPIO to I2C pins
```

Bevor der Master-Modus aufgerufen werden kann, muss das I2CONSET-Register initialisiert werden

```
1 bla
```

Schieberegister I2DAT enthält ein Byte der zu übertragenden seriellen Daten oder ein gerade empfangenes Byte. Daten in I2DAT werden immer von rechts nach links verschoben. Das erste zu sendende Bit ist das MSB (Bit 7), und nachdem ein Byte empfangen wurde, befindet sich das erste Bit der empfangenen Daten im MSB von I2DAT.

Steuerregister I2CONSET und I2CONCLR enthalten Bits, die zur Steuerung der folgenden I2C-Blockfunktionen verwendet werden: Start und Neustart einer seriellen Übertragung, Beenden einer seriellen Übertragung, Bitrate, Adresserkennung und Bestätigung. Der Inhalt des I2C-Steuerregisters kann als I2CONSET gelesen werden. Beim Schreiben auf I2CONSET werden Bits im I2C-Steuerregister gesetzt, die denen im geschriebenen Wert entsprechen. Umgekehrt löscht das Schreiben in I2CONCLR Bits im I2C-Steuerregister, die denen im geschriebenen Wert entsprechen.

2.4.3 Software

```
1 unsigned int ADXLI2CAdresss = 0x3A;
```

2.4.4 Probleme und Lösungen

2.5 Würfel mit LED-Panels

Die Komponente Würfel empfängt Daten der Panelinformationen von der Komponente Flüssigkeitssimulation und beleuchtet die Panels entsprechend dieser Daten. Die Anzahl der Teilchen pro Pixel dient als Intensität der blauen Farbe.

2.5.1 Hardware

Der RGB-Würfel wird aus sechs 32×32 RGB Matrix LED-Panels aufgebaut. Ein Panel benötigt den Strom von $2A$. Die Panels werden anhand einer Stromversorgung geschaltet.

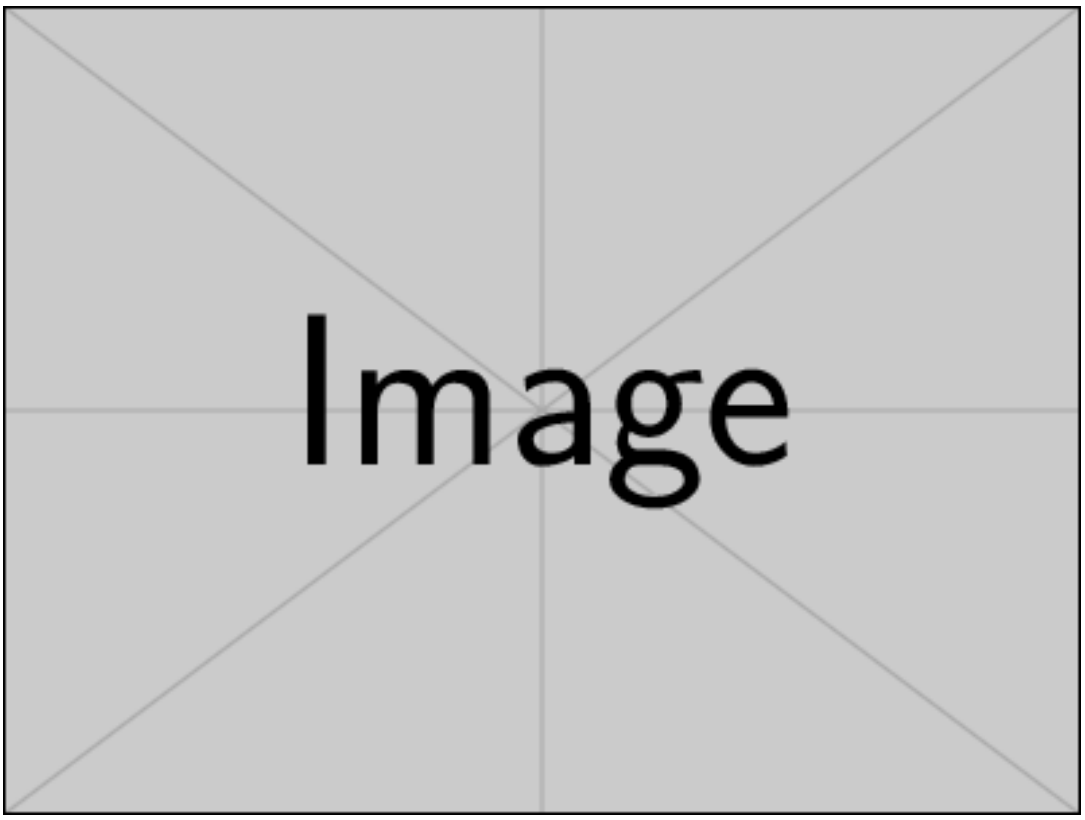

Abbildung 2.3: Aufbau von sechs 32×32 RGB Matrix LED-Panels

Ein Panel wird über 16 Kabel direkt mit dem Board verbunden und die restlichen fünf Panels werden nebeneinander anhand Flachbandkabel verbunden (sogenannte Daisy-Chain). Abbildung 2.3 stellt den gesamten Aufbau der Komponente Würfel dar.

2.5.2 Mechanismus und Konfiguration einer Anzeige

Eine Anzeige hat 32 LEDs in einer Reihe beziehungsweise $32 \times 32 = 1024$ LEDs. Diese werden in 16 Teile abgeschnitten. Erster Teil ist für die 1.- und 16. Zeile, zweiter Teil ist für die 2.- und 17. Zeile bis hin zum sechzehnten Teil für die 16.- und 32. Zeile.
Die Information eines Pixels wird per Takt mittels des Schiebregisters horizontal verschoben. Werden mehrere Panels verkettet, wird die Information zum weiteren Panel verschoben. Das heißt, die verketteten Panels können als ein Panel mit den breiteren Spalten betrachtet werden.

<Schritte>

1. Setzen alle Pins auf `Low` (anfang)
2. `R1,G1,B1,R2,G2,B2` Pins auf `High/Low` setzen
3. `CLK` Pin auf `High` setzen
4. `CLK` Pin auf `Low` setzen
5. Schritte 2-4 für alle Spalten wiederholen

6. Mit Adresspins `ABCD` eine gezielte Zeile auswählen
7. `OE` Pin auf `High` setzen
8. `LAT` Pin auf `High` setzen (erleuchtet LED)
9. `LAT` Pin auf `Low` setzen
10. `OE` Pin auf `Low` setzen

Jeder Zyklus leuchtet LEDs und schaltet diese gleich aus. Die Unterfunktion `refresh` muss daher kontinuierlich aufgerufen werden.

2.5.3 Software

Kapitel 3

Ergebnis