Schulwissen

Zusammenfassung aller relevanten Dinge der Schule

Table of contents

Generell

Hier werden alle für die Schule relevanten Dinge in kurzen und praktischen Zusammnenfassungen aufgegriffen.

Abitur Zusammenfassungen

Folgend sind alle für das Abitur relevanten Zusammenfassungen zu finden.

IT-HW Zusammenfassung

ADC

Der Analog-Digital-Converter wandelt die reale Welt (unendlich viele Werte und stufenlos) in digitale Werte um. Diese Welt hat Stufen und eine begrenzte Anzahl an möglichen Werten. Diese sind abhängig von der Bitanzahl des AD-Converters. Um die Anzahl der möglichen Werte zu errechnen, wird folgende Formel benutzt:

$$AnzahlWerte/Stufen = 2^{AnzahlBits} - 1$$

Dies wird auch Quantisierung genannt. Damit hat jeder AD-Wandler automatisch auch eine Auflösung. Diese beschreibt wie viele analoge Werte in einem digitalen zusammengefasst werden. Auch diese lässt sich berechnen. Die Formel hierzu ist:

$$Auflösung = \frac{höchsterWert(analog)}{AnzahlWerte}$$

Somit hat jeder Wert einen potenziellen Messfehler. Dieser berechnet sich durch die folgende Formel:

$$Quantisierungsfehler = +- Auflösung/2$$

Es kann durchaus vorkommen, dass wir einen AD-Wandler programmieren müssen und den eingelesen Wert ausgeben. Dieser wird dann jedoch immer zwischen 0 und 1 sein, da Mbed den Wert bereits normiert. Normierung bedeutet, den Wert immer zwischen 0 und 1 auszugeben, sprich den Wert quasi in Prozent anzugeben.

$$normalisiert = \frac{analogerWert}{MaxAnalogerWert}$$

Konzepte

Es gibt verschiedene Konzepte die bei der Nutzung von ADCs Anwendung finden. Die beiden wichtigsten sind die Kurvenglättung und die Fehlerkorrektur:

Kurvenglättung

Um einen Mittelwert zu erreichen werden in einem bestimmten Interval die Werte in ein Array gespeichert und aus diesem ein Mittelwert berechnet. Dies führt dazu, dass kurzzeitige Extreme in analogen Welt den digitalen Sensor nicht beeinflussen. Gleichzeitig bringt dies jedoch eine Verzögerung mit sich.

Fehlerkorrektur

Eine andere Methode um Schwankungen zu reduzieren ist es eine Schwelle einzubauen, damit ein analoger Wert der sich genau am Wendepunkt zweier digitaler Werte befindet den

Sensor nicht verwirrt bzw. das System beeinflusst. Hier wird dann eine sogenannte Hysterese eingebaut. Die hälfte wird nun auf die Schwelle aufaddiert, die andere hälfte subtrahiert. Dies führt dazu, dass es größere Änderungen im analogen Wert geben muss und kleine Schwankungen das System nicht beeinflussen.

Programmierung

Unser Mikrocontroller enthält auch einen AD-Wandler. Dieser lässt sich wie folgt programmieren:

```
// Deklaration
AnalogIn ad(PA_0);
// Einlesen
float in = ad;
// Ausgeben (normiert)
lcd.printf("AD Wert normiert: %f", in);
```

PWM

Pulsweitenmodulation steuert Geräte, wie z.b. Lampen oder Motoren, durch schnelles an- und wieder ausschalten des Stroms, welcher diese Geräte ansteuert. Dies führt bei Lampen zu geringerer Helligkeit und bei Motoren zu niedrigerer Geschwindigkeit. Damit können Lampen gedämmt, bzw. Geräte allgemein stufenlos verstellt werden, auch wenn diese in ihrer eigentlichen Konfiguration nur eine binäre Struktur haben (an oder aus). Der Tastgrad definiert die Helligkeit bzw. allgemein gesagt die Intensität des Geräts. Dieser ist wie folgt definiert:

$$Tastgrad = t_{on}/T$$

T ist die festgelegte Periodendauer, meist 1 Sekunde. t_{on} ist die Zeit, die der Strom in der Periode anliegt. Diese Zeit ist variabel, hier findet die Modulation statt.

Programmierung

Um auf unserem Mikrocontroller Pulsweitenmodulation zu programmieren, sind folgende Befehle wichtig:

```
// Deklaration
PwmOut licht(PC_6);
// Initialisierung
licht.period_us(100); // Legt die Periode auf 100 mikrosekunden fest
```

```
// Festlegen des Tastgrads (hier 0,5 fest)
licht = 0.5
```

I2C

I2C ist ein serieller Kommunikationsbus zum Steuern mehrerer Geräte. Der Mikrocontroller selbst ist hier nur durch 2 Leitungen mit den anderen Geräten verbunden. Diese sind die SDA (Serial Data) und SCL (Serial Clock). Serial Data ist eine bidirektionale halb duplex Kommunikationsleistung, was bedeutet, dass zwar in beide Richtungen kommuniziert werden kann, jedoch immer nur in eine gleichzeitig. Serial Clock gibt den Takt an, damit die Übertragung Takt synchron erfolgt. Bei der steigenden Flanke auf dieser Leitung folgt auf der SDA-Leitung das nächste Bit. Dieses bei der fallenden Flanke der Clock übernommen. Die Übertragung zwischen zwei Geräten läuft folgendermaßen ab:

- Vor der Übertragung sind beide Leitungen auf „1“ (Strom liegt an)
- Der Master zieht die SDA-Leitung auf 0, während die Clock auf 1 ist → Startbedingung
- Die folgenden 8 Bit sind die 7 Bit Adresse des Slaves und das Write/Read Bit, welches angibt in welche Richtung Daten übertragen werden sollen
- Der Slave mit der angegebenen Adresse bestätigt den Empfang, durch eine 0 auf der SDA-Leitung beim nächsten Takt
- Nach der Übertragung wird bei Clock = 1, die SDA-Leitung ebenfalls auf 1 gezogen → Endbedingung

Programmierung

Um dies in C bzw. C++ zu programmieren werden folgende Befehle benutzt:

```
// Definition Geräteadresse
#define RTC 0xD0
// Deklaration I2C
I2C i2c(PB_11, PB_10); // SDA, SCL
char daten[2] // Datenarray festlegen
i2c.write(daten, 2, 0) // Schreibt das Datenarray mit der Länge 2 vom
Master zum Slave. Das letzte Bit gibt an ob die Kommunikation
weitergeht oder aufhört.
```

UART

UART (Universal Asynchronous Receiver Transmitter) ist eine 2-Draht Schnittstelle. Die Verbindungen gehen jeweils von TX (Transmit-Ausgang) zu RX (Receive-Eingang). Beide dieser Verbindungen sind voll duplex und können somit gleichzeitig senden und empfangen. Jeder der Empfänger besteht aus einem Schieberegister und einem Taktgenerator. Der Ruhepegel ist " 1" und die Übertragung beginnt somit mit einer "0". Das erste Bit startet den Taktgenerator. Eine UART Verbindung braucht keine Clock bzw. Taktleitung, da beide Empfänger über eigene Taktgeneratoren verfügen. Die Kommunikation wird deshalb auch asynchron genannt, jedoch muss bei beiden Taktgeneratoren die gleiche Frequenz, auch Baudrate genannt, eingestellt sein. Schlussendlich wird die Kommunikation mit einer "1" als Schlussbit beendet. Die Anzahl der übertragenen Bits kann eingestellt werden und beträgt bei unserem Mikrocontroller (Mbed allgemein) zwischen 5 und 8. Ebenfalls kann die Anzahl der Stoppbits definiert werden, bei Mbed zwischen 1 und 2.

Übertragungsfehler

Übertragungsfehler können durch ein Paritybit erkannt werden. Dies bedeutet, dass zwischen Start- und Stopbits jeweils ein Bit (meist "1") eingefügt wird, was bedeutet, dass der Empfänger eine Mindestanzahl an Bits auf "1" erwartet. Erhält er weniger schlieβt dieser auf einen Übertragungsfehler.