Produktvideo
mit erweiterten
Demoanwendungen
Alle Infos: Webcode #2001

Software kostenfrei als Download

Komfortable PC-Software

Direkte Live-Ansteuerung vom PC

125 einzeln ansteuerbare RGB-LEDs

Demoprojekte für eigene Projekte verfügbar

VIDEO ONLINE

360° ONLINE

5x5x5-RGB-Cube

in eigene Anwendungen integrieren

Mit der mitgelieferten Konfigurationssoftware können auf dem 5x5x5-RGB-Cube statische Sequenzen aufgespielt werden, die man anschließend ohne einen PC abspielen kann. Ebenso gibt es die Möglichkeit, den Cube zur Laufzeit mit dynamisch erzeugten Bildern zu betreiben, was bereits für die Live-Vorschau der Konfigurationssoftware genutzt wird.
Natürlich reizt den Programmierer solch ein von außen programmierbares AVR-System zu eigenen Lösungen und Anwendungen fern der „vorgestanzten" Software. Im Rahmen dieses Artikels wird mittels einer Demosoftware die Nutzung des Preview-Modus zur dynamischen Anzeige dargestellt und mit Quellcodebeispielen erläutert.

Cube für Individualisten

Die mit dem Cube mitgelieferte Konfigurationssoftware bietet bereits eine große Anzahl von Möglichkeiten, zudem ist sie ohne Programmierkenntnisse bedienbar – sie erfordert nur die eigene Kreativität.

Auf der Grundlage der hier vorgestellten und zum Download bereitgestellten Demosoftware kann man mit dem Cube sehr individuelle Anwendungen erschließen, etwa zur Visualisierung von Musik oder zur Nutzung unter anderen Betriebssystemen als MS Windows. Da es sich bei dem Cube um ein sehr ansehnliches und auch aus größerer Entfernung gut sichtbares Lichteffektgerät handelt, das zudem über eine Programmierschnittstelle verfügt, bieten sich vielfältige individuelle Einsätze, etwa im Diskobetrieb, aber auch z. B. in der Ausbildung geradezu an: ein für Programmierer ideal geeignetes Objekt für die Einbindung in eigene Anwendungen!

Die Entwicklungsumgebung

Für die Demoanwendung wurde C# als Programmiersprache gewählt. Für die Erstellung der Anwendung lässt sich die kostenlose Entwicklungsumgebung Visual Studio 2010 Express [1] beziehungsweise eine der Nachfolgeversionen verwenden.

Der Cube wird von der Konfigurationssoftware über den Silabs USBXpress-Treiber und die zugehörige SiUSBXp-Kommunikationsbibliothek (SiUSBXp.dll) angesprochen. Für die Demosoftware [2] wird der Cube über den CP2102_ID-Changer, welcher der Demosoftware beiliegt, auf das Ansprechen via „Virtuellem Com Port" (VCP) umprogrammiert.

Ebenso kann der Cube für die Nutzung der Konfigurationssoftware mit dem CP2102_ID-Changer wieder auf USBXpress programmiert werden. Damit der Cube richtig erkannt wird, muss der Treiber installiert werden, dieser liegt der Demosoftware [2] bei oder kann auf der Silabs-Homepage [3] für das jeweilige Betriebssystem bezogen werden.

Die Nutzung des Cubes über den VCP bietet den Vorteil, dass dieser bereits von vielen Programmiersprachen ohne zusätzliche Kommunikationsbibliotheken unterstützt wird und somit universeller einsetzbar ist. Durch diese Umstellung hat man beispielsweise die Möglichkeit, den Cube nach der Umprogrammierung an einem Linux-System oder Mac OS mit den Treibern von Silabs [3] zu verwenden, sofern die gewählte Programmiersprache von den Systemen unterstützt wird.

Tabelle 1

VCP-Kommunikationsparameter

Baudrate	128.000
Datenbits	8
Parität	none/keine
Stoppbits	1

Kommunikation über USB

Für die Kommunikation mit dem Cube wird in C# die SerialPort-Klasse verwendet, welcher der Name und die Kommunikationsparameter bei der Initialisierung übergeben werden. Die Parameter für die Kommunikation über VCP sind der Tabelle 1 zu entnehmen. In der Demoanwendung geschieht dies in der Menu-Klasse in der ConnectToComPort-Methode. Anschließend wird die Verbindung über die Open-Methode der SerialPort-Klasse geöffnet und kann über die Read- und Write-Methoden genutzt werden.

Die Datenübertragung erfolgt in einem festen Kommunikationsrahmen (Tabelle 2). Bei der Übertragung gibt es zwei Steuerzeichen, die in den weiteren Daten nicht vorkommen dürfen, dies sind zum einen das Startzeichen (0x02) und zum anderen das Escape-Zeichen (0x10). Wenn eines dieser Steuerzeichen in der Länge, den Nutzdaten oder der Checksumme enthalten ist, wird diesem das Escape-Zeichen vorangestellt und das höchstwertige Bit (0x80) gesetzt. Hieraus ergeben sich folgende Umsetzungen: 0x02 wird zu 0x1082 und 0x10 wird zu 0x1090. In der Demoanwendung findet die Kommunikation in der ComPortDevice-Klasse statt. Die einzelnen Befehle und die zugehörigen Antworten können den Tabellen 3 und 4 entnommen werden. Bei allen 16-Bit-Zahlen (2 Byte) wird das jeweils höchstwertige Byte zuerst übertragen (Big-Endian).

Tabelle 2

Kommunikationsrahmen

Inhalt	Byteanzahl	Beschreibung
STX (0x02)	1	konstantes Startzeichen
Nutzdatenlänge	2	Anzahl der Nutzdaten (Befehl und Parameter) im Rahmen
Befehl	1	Befehl der Übertragung siehe Tabelle 3 und 4
Parameter	0 bis 530	Parameter des Befehls siehe Tabelle 3 und 4
Checksumme	2	CRC16 von allen vorherigen Bytes, siehe Abschnitt Checksummen-Berechnung

Checksummen-Berechnung

Zur Sicherung der Übertragung wird aus den zu übertragenden Daten eine Checksumme (CRC) mit einer Länge von 16 Bit (2 Byte) berechnet. Die Parameter für die Berechnung sind der Initialewert (0xFFFF) und das CRC-Polynom (0x8005). Details zur Berechnung sind in [4] zu finden.

In der Demosoftware wird für die Berechnung die „CRC16“-Klasse verwendet. Nach der Initialisierung wird die Checksumme durch Übergabe der Daten an die Shift-Methoden berechnet. Hierbei ist zu beachten, dass zuerst die Steuerzeichen aus der Länge und den Nutzdaten ersetzt werden müssen, bevor diese an die Shift-Methoden übergeben werden. Anschließend werden die Steuerzeichen aus dem errechneten CRC entfernt und zusammen mit den übrigen Daten an das Gerät übertragen.

Tabelle 3

Befehls-Übersicht

Befehl	Parameter	Parameterlänge	Beschreibung
v (0x76)		0	liest die Firmwareversion aus
N (0x4E)		0	Normal-Modus starten
P (0x50)		0	Preview-Modus starten
D (0x44)		0	Demo-Modus starten
m (0x6D)		0	Modus auslesen
C (0x43)	X-, Y-, Z-Koordinate je 1 Byte (0x00 bis 0x04)	3	schaltet eine RGB-LED an der Koordinate aus
S (0x53)	X-, Y-, Z-Koordinate je 1 Byte (0x00 bis 0x04), R-, G-, B-Farbwert je 1 Byte (0x00 bis 0xFF)	6	schaltet eine RGB-LED an der Koordinate mit der angegebenen Farbe ein
G (0x47)	zu jeder LED R-, G-, B-Farbwert zu je 1 Byte (0x00 bis 0xFF)	375	schaltet alle RGB-LEDs mit den angegebenen Farben ein, Reihenfolge siehe Abschnitt „Vollständige Bilder übertragen“
K (0x4B)	1 Byte Anzahl der genutzten Sequenzen (0x00 bis 0x14), je Sequenz 8 Byte Konfiguration	1 bis 161	überträgt die Konfiguration in das EEPROM des Cubes, Aufbau der Konfiguration siehe Abschnitt „Statische Konfiguration übertragen“
k (0x6B)		0	liest die Konfiguration aus dem EEPROM des Cubes aus
F (0x46)	2 Byte Nummer des Flash-Speichers, 528 Byte Flash-Speicher-Inhalt	530	schreibt Daten von Mustern in den Flash-Speicher, Aufbau der Daten siehe Abschnitt „Statische Konfiguration übertragen“
f (0x66)	2 Byte Nummer des Flash-Speichers	2	liest den Inhalt des Flash-Speichers aus
X (0x58)		0	Werksreset starten
! (0x21)		0	Bootloader starten

Einzelne LEDs ein- oder ausschalten

Einzelne LEDs können mit den Befehlen „S“ und „C“ angesteuert werden. Hierbei hat jede LED eine X-, Y- und Z-Koordinate, welche jeweils einen Wert von 0 bis 4 haben kann. Wenn man die Seite mit den Tasten vor sich hat, geht die X-Achse von links nach rechts, die Y-Achse von unten nach oben und die Z-Achse von vorn nach hinten. Beim Befehl „C“ werden nur die Koordinaten als Parameter übergeben, um die LED auszuschalten. Beim Befehl „S“ wird neben den Koordinaten noch der Rot-, Grün- und Blau-Farbwert, welcher als je ein Byte (0x00 bis 0xFF) definiert ist, als Parameter übergeben, um die LED mit der entsprechenden Farbe anzuschalten. Wenn nur einzelne LEDs an- bzw. ausgeschaltet werden sollen, können diese Funktionen sehr gut verwendet werden. Wenn viele LEDs an- oder ausgeschaltet werden sollen, empfiehlt sich die Übertragung eines gesamten Bildes.

Vollständige Bilder übertragen

Ein komplettes Bild kann mit dem

Befehl „G“ übertragen werden, hierbei ist die richtige Reihenfolge der Daten wichtig, damit jede LED die gewünschte Farbe hat. Der Index im Parameter einer LED wird über folgende Formel berechnet:
index = (x * 75) + (y * 15) + (z * 3)

An dem Index und den 2 folgenden Bytes werden die Rot-, Grün- und Blau-Farbwerte eingefügt. Ein Beispiel für die Verteilung der Daten kann der Tabelle 5 entnommen werden.

Statische Konfiguration übertragen

Die Konfiguration teilt sich in zwei Bereiche, zum einen den EEPROM und zum anderen den Flash-Speicher. Im EEPROM stehen die Anzahl der Sequenzen und die Konfigurationsdaten jeder Sequenz, welche der Tabelle 6 entnommen werden können.

Der EEPROM kann mit dem „K“-Befehl beschrieben und mit dem „k“-Befehl ausgelesen werden. Der Flash-Speicher des Cubes hat 4096 Flashpages, die jeweils 528 Byte groß sind. Die Bilder einer Sequenz werden mit derselben Byte-Reihenfolge wie bei der Übertragung von vollständigen Bildern gespeichert, hierbei werden die Daten aller Bilder aneinander gereiht und zu jeweils 528 Byte in eine Flashpage gespeichert, Tabelle 7 zeigt ein Beispiel. Nicht vollständig ausgefüllte Flashpages werden aber nicht von der folgenden Sequenz weiter genutzt, diese fängt immer bei der nächsten ungenutzten Flashpage an. Die Übertragung der Daten erfolgt jeweils mit dem Index der Flashpage und den Befehlen „F“ zum Schreiben und „f“ zum Lesen. Sollten die Konfiguration im EEPROM und die Daten im Flash nicht zueinander passen oder unvollständig übertragen worden sein, fällt dies meist beim Abspielen auf, weil etwa die Farben nicht stimmen oder vorher verwendete Bilder verschoben zu sehen sind.

Tabelle 4

Antwort-Übersicht

Befehl	Parameter	Parameter-länge	Beschreibung
v (0x76)	Version je Stelle 2 Byte	6	ausgelesene Firmwareversion
N (0x4E)		0	Normal-Modus wurde gestartet
P (0x50)		0	Preview-Modus wurde gestartet
D (0x44)		0	Demo-Modus wurde gestartet
m (0x6D)	0 = Normal, 1 = Preview, 2 = Demo	1	aktueller Modus des Cubes
C (0x43)		0	RGB-LED wurde ausgeschaltet
S (0x53)		0	RGB-LED wurde eingeschaltet
G (0x47)		0	RGB-LEDs wurden eingeschaltet
K (0x4B)		0	Konfiguration wurde im EEPROM gespeichert
k (0x6B)	1 Byte Anzahl der genutzten Sequenzen (0x00 bis 0x14), je Sequenz 8 Byte Konfiguration	1 bis 161	Konfiguration aus dem EEPROM, Aufbau der Konfiguration siehe Abschnitt „Statische Konfiguration übertragen“
F (0x46)	2 Byte Nummer des Flash-Speichers	2	Daten wurden in den Flash-Speicher geschrieben
f (0x66)	2 Byte Nummer des Flash-Speichers, 528 Byte Flash-Speicher-Inhalt	530	Daten aus dem Flash-Speicher, Aufbau der Daten siehe Abschnitt „Statische Konfiguration übertragen“
X (0x58)		0	Werksreset wurde gestartet
! (0x21)		0	Bootloader wurde gestartet

Tabelle 5

Beispiel der Datenverteilung eines Bildes

Index	Farbe	X-Koordinate	Y-Koordinate	Z-Koordinate
0	Rot	0	0	0
1	Grün	0	0	0
2	Blau	0	0	0
3	Rot	0	0	1
4	Grün	0	0	1
5	Blau	0	0	1
372	Rot	4	4	4
373	Grün	4	4	4
374	Blau	4	4	4

Aufbau der Demoanwendung

Menu-Klasse

Der Einstiegspunkt der Konsolenanwendung (Bild 1) ist die Main-Methode der Menu-Klasse. In dieser wird die ConnectToComPort-Methode aufgerufen, damit der Nutzer die Nummer des VCP eingeben und die Verbindung zum Gerät aufgebaut werden kann.

Anschließend wird die ShowMenu-Methode aufgerufen, in der der Nutzer die Menüoptionen angezeigt bekommt und seine Auswahl treffen kann, durch welche die aufzurufende Methode bestimmt wird. Die SwitchMode-Methode und die TurnOnAllTheLeds-Methode sind direkt in der Menu-Klasse implementiert, während die anderen Methoden die speziellen Klassen nutzen, um den Cube anzusprechen. Die SwitchMode-Methode fordert zur Eingabe des gewünschten Modus auf und überträgt diesen als Befehl an den Cube. Bei der TurnOnAllTheLeds-Methode muss der Nutzer die

Tabelle 6

Sequenzdaten im EEPROM

Inhalt	Byteanzahl	Beschreibung
Startadresse	2	Index der ersten Flashpage, in dem die Daten der Sequenz stehen
Bildanzahl	2	Anzahl der Bilder, die zu der Sequenz gehören und ab der Startadresse im Flash gespeichert sind
Intervall	2	Intervall, das, mit 10 Millisekunden multipliziert, der Abspielgeschwindigkeit einer Sequenz entspricht
Wiederholungen	1	Anzahl der Wiederholungen, bevor die Sequenz endet
Aktion am Ende	1	Aktion, die ausgeführt wird, wenn die Sequenz beendet ist, mögliche Werte sind alle LEDs aus (Wert 0x00), letztes Bild stehen lassen (Wert 0x01), nächste Sequenz starten (Wert 0x02) und zufällige Sequenz starten (Wert 0x03)

Tabelle 7

Beispiel Nutzung des Flash-Speichers

Flashpage-Index	Inhalt
0	Sequenz 1 Bild 1 (375 Byte), Sequenz 1 Bild 2 (153 Byte)
1	Sequenz 1 Bild 2 (222 Byte)
2	Sequenz 2 Bild 1 (375 Byte)
3	Sequenz 3 Bild 1 (375 Byte), Sequenz 3 Bild 2 (153 Byte)
4	Sequenz 3 Bild 2 (222 Byte), Sequenz 3 Bild 3 (306 Byte)
5	Sequenz 3 Bild 3 (69 Byte), Sequenz 3 Bild 4 (375 Byte)

gewünschte Farbe als Hexadezimalzahl in der GetRgbColor-Methode eingeben und anschließend wird über den „G“-Befehl ein komplettes einfarbiges Bild übertragen. In der Menu-Klasse gibt es auch mehrere Methoden, die den Nutzer zur Eingabe eines Wertes aufrufen und diesen auf Gültigkeit überprüfen. Diese sind die GetValidMode-Methode zur Eingabe eines Modus, die GetValidUInt-Methode zur Eingabe einer vorzeichenlosen Ganzzahl, die GetValidInt-Methode zur Eingabe einer vorzeichenbehafteten Ganzzahl, die GetTotalBingos-Methode und die GetMaxLeds-Methode für die Parameter der LedBingo-Klasse. Mit Hilfe dieser Methoden werden die Parameter vom Nutzer abgefragt und in den Methoden, die die anderen Klassen aufrufen, genutzt.

ComPortDevice-Klasse

Die Kommunikation mit dem Cube ist in der ComPortDevice-Klasse gekapselt, welche die SendCommandToDevice-Methode zur Verfügung stellt. Diese Methode baut den zuvor beschriebenen Datenrahmen zusammen und liefert das Ergebnis der ReadAnswerAfterCommand-Methode mit der Antwort vom Gerät zurück. Die EnqSpecialBytes-Methode und die DeenqSpecialBytes-Methode sorgen bei der Kommunikation dafür, dass keine Steuerzeichen in den Daten sind bzw. machen dies in der Antwort rückgängig.

CRC16-Klasse

In der CRC16-Klasse wird die Prüfsumme für die Datenübertragung berechnet. Hier stehen neben der CRC16_init-Methode zum Initialisieren die Shift-Methoden für die Verarbeitung der Prüfsumme zur Verfügung.

LedCoordinate-Klasse

Die LedCoordinate-Klasse dient hauptsächlich zur Speicherung der Informationen einer einzelnen LED. Neben der Farbe, die mit den Einzelwerten für Rot in dem R-Attribut, Grün in dem G-Attribut und Blau in dem B-Attribut gespeichert ist, enthält die Klasse die Koordinaten der LED, welche in dem X-, Y- und Z-Attribut gespeichert sind. Bei den Koordinaten wird auch sichergestellt, dass diese in dem gültigen Bereich von 0 bis 4 sind.

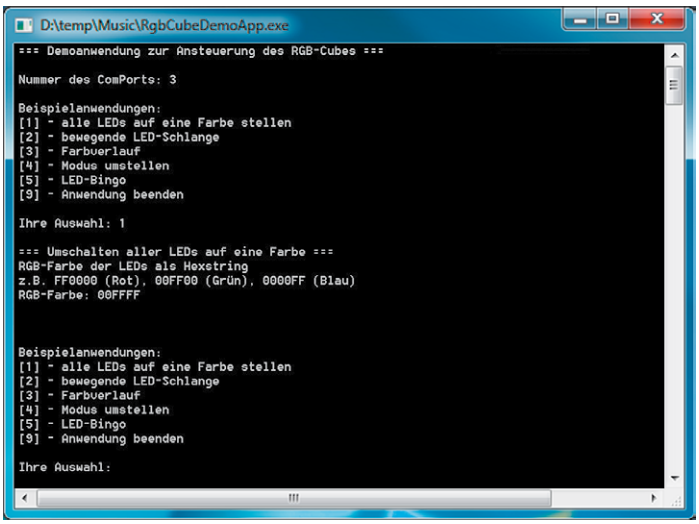

Bild 1: Die Demoanwendung zur Ansteuerung des Cubes in der Konsole

Für die Nutzung in der LedSnake-Klasse besitzt die Klasse die MovePoint-Methode, die eine neue Instanz der LedCoordinate-Klasse zurückgibt, die auf einer zufälligen Achse um eine Position verschoben ist, zu der aktuellen Instanz der LedCoordinate-Klasse.

LedColorGradient-Klasse

Durch die LedColorGradient-Klasse erfolgt die Darstellung eines Farbverlaufs auf dem Cube. Über die Attribute StartX, StartY und StartZ lässt sich der Punkt verschieben, von dem der Farbverlauf ausgeht. Der Farbverlauf nutzt den HSV-Farbraum, welcher über das SaturationPercentage-Attribut die Farbsättigung und über das BrightnessvaluePercentage den Helligkeitswert erhält. Mit dem HueIntervall-Attribut wird bestimmt, in welchem Intervall der Farbton, welcher zwischen 0 und 359 liegt, wechselt. Die Anzeige wird über die Start-Methode gestartet und über die Stop-Methode beendet, diese starten bzw. stoppen jeweils die zyklische Ansteuerung des Cubes, welche über einen Timer mit 40 Millisekunden Intervall und der ViewTimer_Elapsed-Methode realisiert wurde. In dieser wird zuerst über die GetNextColor-Methode die nächste Farbe berechnet und der Liste der zuletzt genutzten Farben am Anfang hinzugefügt, anschließend wird die letzte Farbe aus der Liste entfernt, so dass diese Liste immer 14 Einträge hat, was die maximale Anzahl genutzter Farben zu einem Zeitpunkt auf dem Cube ist. Nachdem die nächsten Farben feststehen, wird für jede LED die neue Farbe ermittelt, hierzu wird die absolute Entfernung der LED zum Startpunkt berechnet und anhand dessen die Farbe aus der Liste der zuletzt genutzten Farben verwendet. Abschließend werden die Daten über den „G“-Befehl zum Cube übertragen. Die ConvertHsvToRgb-Methode wird hier wie auch in weiteren Klassen zu Hilfe genommen, um aus der Farbe im HSV-Farbraum die entsprechenden Rot-, Grün- und Blau-Farbwerte zu erhalten.

LedSnake-Klasse

Mit der LedSnake-Klasse wird eine sich bewegende Schlange von LEDs auf dem Cube dargestellt. Die Anzeige erfolgt wie bei der LedColorGradient-Klasse, nur dass der Timer ein Intervall von 100 Millisekunden hat. In der ViewTimer_Elapsed-Methode wird die ShowLedSnakeMain-Methode aufgerufen. In dieser wird die MovePoint-Methode der LedCoordinate-Klasse aufgerufen, um die nächste LED zu erhalten, und in einer Warteschlange (Queue) gespeichert. Anschließend wird die LED über die ChangeLedAtCoordinate-Methode in der nächsten Farbe angeschaltet, und wenn mehr als 4 LEDs an sind, wird die erste aus der Queue genommen und ausgeschaltet. In der ChangeLedAtCoordinate-Methode wird für das Einschalten einer LED der „S“-Befehl und für das Ausschalten einer LED der „C“-Befehl an den Cube gesendet. Die verwendeten GetNextColor- und ConvertHsvToRgb-Methoden sind identisch mit denen in der LedColorGradient-Klasse.

LedBingo-Klasse

Über die LedBingo-Klasse werden alle LEDs in einer zufälligen Reihenfolge über den „S“-Befehl eingeschaltet. Nach jedem Einschalten wird geprüft, ob die

Tabelle 8

LED-Zustand in der LedBingo-Klasse

Wert	Bedeutung
0	LED ist aus
1	LED ist ein und ist nicht im Bingo
2	LED ist ein und ist im Bingo

gewünschte Anzahl an Bingos – also 5 in einer Reihe liegende LEDs – vorliegt. Bei der Initialisierung wird die Reihenfolge, in der die LEDs angeschaltet werden, durch 1000 Mischungen von jeweils 2 LEDs festgelegt. Über die totalBingos- und maxLeds-Parameter bei der Initialisierung werden zusätzliche Bedingungen für das Stoppen der Anzeige festgelegt. Die Anzeige erfolgt wie bei der LedSnake-Klasse, und beim Stoppen wird die ShowResult-Methode aufgerufen, welche alle LEDs in Abhängigkeit des entsprechenden Wertes in der ledsOn-Variablen einfärbt.

In der ViewTimer_Elapsed-Methode wird immer jeweils die nächste LED über den „S“-Befehl eingeschaltet und die zusätzlichen Stoppbedingungen geprüft. Die Anzahl der neu hinzugekommenen Bingos durch das Einschalten einer LED wird in den CheckForBingo-, CheckForDiagonalBingo- und CheckForCrossoverDiagonalBingo-Methoden überprüft, ebenfalls wird dort der Zustand gesetzt, siehe Tabelle 8.

Spektrumanalysator Demoanwendung zur Visualisierung von Musik

Die folgende Demoanwendung zur Nutzung des Cubes als Spektrumanalysator wurde getrennt von den vorigen Anwendungen in ein separates Projekt ausgegliedert. Der Grund hierfür liegt in der Nutzung der freien Audio-Bibliothek „NAudio“[5], die nur für das .Net-Framework 4.0 oder höher vorliegt. Im Gegensatz dazu können die vorherigen Demoanwendungen bereits ab dem .Net-Framework 3.5 genutzt werden.

Die Oberfläche (Bild 2) des Spektrumanalysators wurde mit der Oberflächentechnologie WPF (Windows Presentation Foundation) erstellt. Eine Beschreibung, wie eine Oberfläche in WPF erstellt wird, kann [6] entnommen werden.

Aufbau der Oberfläche

In der Oberfläche wird der VCP ausgewählt, an dem der Cube aktiv ist, anschließend kann über die entsprechende Schaltfläche die Verbindung aufgebaut werden. Das Spektrum wird gegen den Uhrzeigersinn auf den Außenflächen des Cubes dargestellt, der Startpunkt kann ebenfalls in der Oberfläche eingestellt werden. Über 2 Schaltflächen können MP3-Dateien zur Trackliste hinzugefügt und entfernt werden, welche am unteren Ende der Anwendung dargestellt wird. In der Mitte der Oberfläche befinden sich die Schaltflächen zur Steuerung des Abspielens, welche das Abspielen starten, pausieren, stoppen oder zum nächsten bzw. vorherigen Lied wechseln. Daneben erfolgt die Anzeige der aktuellen Position und der Liedlänge. Darunter wird der Name des aktuellen Liedes angezeigt.

Abspielen von Musik und Anzeige des Spektrums

Das Abspielen erfolgt in der AudioPlayback-Klasse,

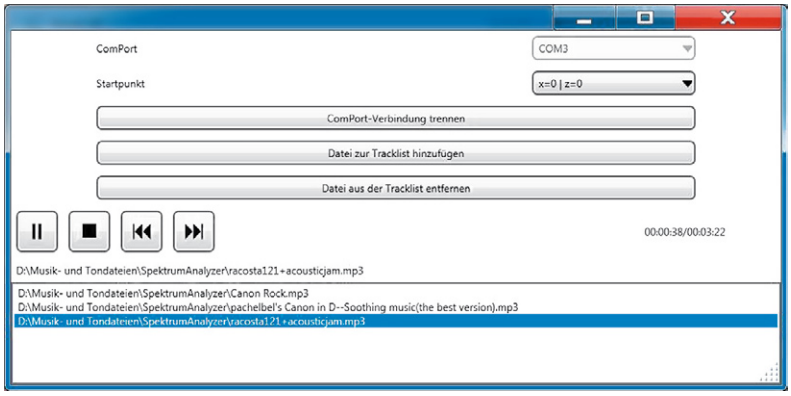

Bild 2: Die Oberfläche des Spektrumanalysators

welche die Funktionen der Audio-Bibliothek nutzt, in der das eigentliche Abspielen stattfindet. In dieser wird auch die SampleAggregator-Klasse verwendet, welche das Spektrum über entsprechende Methoden der Audio-Bibliothek berechnet. Über das FftCalculated-Event werden Daten des Spektrums an die SpectrumAnalyzer-Klasse weitergegeben, ebenso wird das PlaybackStopped-Event durchgereicht, um das Abspielen in der Oberfläche zu steuern. In der SpectrumAnalyzer-Klasse werden die Funktionen des Abspielens gekapselt und die Daten für die spätere Weitergabe an die TriggerRgbCubeWithSpectrum-Klasse aufbereitet, da der Cube deutlich weniger Daten anzeigen kann, als zur Verfügung stehen. Diese Aufbereitung umfasst die Zusammenfassung mehrerer Daten und die Normalisierung des Spektrums auf ein einheitliches Niveau, da ansonsten in einigen Bereichen nur die unteren LEDs genutzt würden. In der TriggerRgbCubeWithSpectrum-Klasse wird der Cube als 25 Säulen zu 5 LEDs interpretiert und mit den Daten vom Spektrum gefüllt. Jede dieser Säulen ist ein Objekt der LedPillar-Klasse, die in Abhängigkeit der Höhe die LEDs in der GetLedCoordinates-Methode entsprechend einfärbt.

In der Trigger-Methode der TriggerRgbCubeWithSpectrum-Klasse erfolgt die Ansteuerung des Cubes, nachdem zuerst die äußeren LEDs befüllt wurden und anschließend die mittleren LEDs mit dem Durchschnittswert gefüllt werden. Abschließend werden die Daten über den „G“-Befehl zum Cube übertragen.

In der MainWindow-Klasse, welche die PlaybackStopped-Events enthält, wird in Abhängigkeit des currentStopAction-Attributes eine weiterführende Aktion ausgeführt, wie zum Beispiel das nächste oder vorherige Lied starten. Dieser Umweg musste gewählt werden, da erst ein neues Lied abgespielt werden kann, wenn das vorherige gestoppt ist.

Fazit

Mit den Informationen dieses Artikels können vielfältige eigene Anwendungen mit dem Cube entwickelt werden. Ebenfalls können die vorgestellten Demoanwendungen die Nutzung des Cubes erweitern und ihn zu einem Blickfang machen.

Wir veranstalten einen LED-Cube-Programmierwettbewerb, bei dem es attraktive Preise zu gewinnen gibt. Näheres dazu finden Sie auf Seite 3 unter „ELV intern“.

Weitere Infos:

[1] Entwicklungsumgebung Download: www.microsoft.com/visualstudio/eng/downloads#d-2010-express
[2] Demosoftware-Download zum 5x5x5-RGB-Cube: Webcode #1248
[3] Silabs VCP Treiber: www.silabs.com/products/mcu/Pages/USBtoUARTBridgeVCPDrivers.aspx
[4] Wikipedia Artikel zur Checksumme: de.wikipedia.org/wiki/Zyklische_Redundanzprüfung
[5] NAudio Klassenbibliothek: http://naudio.codeplex.com
[6] Artikel zur FS20-PCS- und FS20-PCE-Demoanwendung: ELVjournal 6/2012, S. 48