Integration von FS20 PCS und FS20 PCE in eigene Anwendungen

Über die Einbindung in EventGhost sind mit dem FS20 PCE und dem FS20 PCS bereits viele Möglichkeiten vorhanden, um ein breites Spektrum an Anwendungen abzudecken. Darüber hinaus können die beiden Geräte auch in eigene Software-Lösungen integriert werden, damit spezielle Anwendungsfälle realisiert werden können.
Im Rahmen dieses Artikels wird eine solche Anwendung beispielhaft erläutert anhand eines Rollladenschalters, der nur aktiviert werden soll, wenn eine Terrassentür geschlossen ist.

Die Software-Umgebung

Für die hier vorgestellte Demoanwendung wurde C# als Programmiersprache gewählt, für die Erstellung der Anwendung selbst lässt sich die kostenlose Entwicklungsumgebung Visual Studio 2008 Express beziehungsweise eine der Nachfolgeversionen verwenden.

Die Sender und Empfänger FS20 PCS/PCE werden über USB als HID (Human Interface Device) an den Computer angeschlossen, deshalb sind hier keine weiteren Treiber notwendig. Eine Integration wäre so auch mit anderen Programmiersprachen auf unterschiedlichen Plattformen möglich, sofern diese HID unterstützen, denn Bibliotheken zur Kommunikation mit HID sind für viele Programmiersprachen frei verfügbar.

Kommunikation über USB

Bei der verwendeten Kommunikationsbibliothek wird die Verbindung zum Gerät über die Herstelleridentifikationsnummer (Vendor-ID oder kurz VID) und die Produktidentifikationsnummer (Product-ID oder kurz PID) hergestellt. Die VID von ELV ist 0x18EF, die PID des FS20 PCE ist 0xE014 und die des FS20 PCS ist 0xE015. Diese Informationen können, wenn diese nicht bekannt sind, auch über die Eigenschaften der Geräte im Windows-Gerätemanager oder aber über noch tiefer gehende Programme wie USB-View herausgefunden werden.

Durch die Kommunikationsroutinen der Bibliothek kann man sich für Events registrieren, so dass hier darauf reagiert werden kann, wenn man das Gerät vom PC trennt oder Daten empfangen werden. Hierbei muss beachtet werden, dass durch die Nutzung dieser Kommunikationsbibliothek keine anderen Anwendungen, wie z. B. EventGhost, die Geräte zeitgleich nutzen können.

Im Gegensatz zu Verbindungen über einen virtuellen Com-Port oder über einen direkten USB-Treiber muss man hier nicht auf eine Baudrate, Parität oder andere Merkmale achten. Bei HID-Geräten muss lediglich auf die Länge der zu übertragenden Daten und die korrekte Form der Daten Rücksicht genommen werden. Hier steht das erste Byte der übertragenen Daten für die HID-Report-ID, welche beim Senden an das Gerät 0x01 und beim Empfangen vom Gerät 0x02 ist – somit fangen die eigenen Daten erst beim zweiten Byte an.

Beim FS20 PCS bestehen die eigenen Daten immer aus der Anzahl der Bytes (Länge von Befehls-ID und Nutzdaten), der Befehls-ID und den Nutzdaten – in dieser Reihenfolge. Die Nutzdaten müssen jedoch nicht vorhanden sein, hierbei gibt es die Besonderheit, dass immer 11 Byte gesendet werden und nicht genutzte Daten mit 0x00 aufgefüllt werden. Als Beispiel ergibt sich so für das Abfragen der Firmwareversion folgender Aufbau:

0x01 0x01 0xF0 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

Die weiteren Befehle und der Aufbau der Nutzdaten können dem Kommunikationsprotokoll, welches bei der EventGhost-Installation enthalten ist, entnommen werden.

Senden von Funkbefehlen über den FS20 PCS

Es gibt 2 Befehle, mit denen Funkbefehle gesendet werden können: 0xF1 für das einmalige Senden und 0xF2 für das mehrfache Senden. Die Übertragung des Funkbefehls erfolgt aber stets in derselben Form: Zuerst kommen zwei Byte mit dem Hauscode, gefolgt von jeweils einem Byte mit der Adresse, dem Befehl und dem Erweiterungsbyte.

Da Hauscode und Adresse üblicherweise als vier- bzw. achtstellige Zahlen aus den Ziffern 1 bis 4 dargestellt werden, müssen diese in ein bzw. zwei Byte konvertiert werden. Hierbei steht eine Ziffer für zwei Bit des Bytes, somit sind immer vier Ziffern gleich acht Bit bzw. ein Byte. In der Klasse FS20Command wurde hierfür eine Konvertierungsfunktion mit dem Namen „FS20StringToByte“ (siehe Bild 1) geschrieben. Diese Funktion ermittelt für jede Stelle den Wert, wobei dieser um eins reduziert wird (0 bis 3 statt 1 bis 4) und anschließend an der entsprechenden Stelle des Bytes aufaddiert wird, was hier durch Multiplikation des Wertes mit 0, 4, 16 und 64 erreicht wurde. Die Befehle und deren Wert bei der Übertragung können dem Kommunikationsprotokoll entnommen werden. Hierbei ist nur wichtig, dass bei den Befehlen 0x00 bis 0x1F (0 bis 31 dezimal) der Wert des Erweiterungsbytes nicht relevant ist, während bei den Befehlen 0x20 bis 0x3F (32 bis 63 dezimal) der Wert des Erweiterungsbytes Auswirkungen auf das Verhalten des Aktors hat wie zum Beispiel die Einschaltdauer oder die Rampenzeit, während ansonsten die interne Timerzeit des Aktors verwendet wird.

Die möglichen Werte des Erweiterungsbytes stehen ebenfalls im Kommunikationsprotokoll, diese können aber auch berechnet werden. Hierzu wird es in 4-Bit-Werte aufgeteilt, die sogenannten High-Nibble und Low-Nibble, welche somit jeweils Werte von 0x0 bis 0xF (0 bis 15 dezimal) besitzen. Anschließend kann über die Formel „2^High-Nibble * Low-Nibble * 0,25s“ die Dauer berechnet werden.

Durch die Berechnung ergeben sich Sonderfälle, so dass immer, wenn das Low-Nibble 0x0 ist, auch das Ergebnis null Sekunden lautet, wodurch praktisch die Timerzeit ausgeschaltet wird. Dies bewirkt z. B. ein endloses bzw. sofortiges Einschalten je nach Befehl. Auch gibt es einen maximalen Wert des High-Nibble, hier sind keine Werte größer als 0xC (12 dezimal) zulässig. Ebenfalls sind viele Zahlen doppelt, z. B.: 2^0 * 2 * 0,25s = 0,5s und 2^1 * 1 * 0,25s = 0,5s.

```
public static byte FS20StringToByte (string value)
{
    byte temp = 0x00;
    int tempValue = int.Parse (value);
    temp += (byte) ((tempValue % 10) – 1);
    tempValue = (tempValue – (tempValue % 10)) / 10;
    temp += (byte) ((tempValue % 10) – 1) * 4);
    tempValue = (tempValue – (tempValue % 10)) / 10;
    temp += (byte) ((tempValue % 10) – 1) * 16);
    tempValue = (tempValue – (tempValue % 10)) / 10;
    temp += (byte) ((tempValue % 10) – 1) * 64);

    return temp;
}
```

Bild 1: Konvertierungsfunktion „FS20StringToByte“

In der Demoanwendung wird die Zeitberechnung in der FS20Time-Klasse berücksichtigt und z. B. eine geordnete Liste aller möglichen nicht doppelten Zeiten über die Methode „GetPossibleUnicTimes“ erstellt.

Empfangen von Funkbefehlen mit dem FS20 PCE

Beim Empfangen werden die Daten bereits in eine besser lesbare Darstellung umgewandelt, so dass hier anders vorgegangen werden muss als bei der Konvertierung der Daten für die Übertragung zum FS20 PCS.

Von der Kommunikationsbibliothek werden die Funkbefehle als 13 Byte langes Signal empfangen, wobei das erste Byte die HID-Report-ID und das zweite die Länge ist. Der Hauscode wird mit vier Byte Länge mit jeweils High-Nibble und Low-Nibble empfangen, bei denen jedes Nibble für eine Stelle steht und nur noch als Zeichenkette im Hexadezimalformat genutzt werden muss. Für die Adresse werden nur zwei Byte benötigt, die Berechnung stellt sich genauso wie beim Hauscode dar.

Für den Befehl wird ein Byte übertragen, in dem dieser Befehl als Zeichenkette im Hexadezimalformat mit den Werten 0 bis 31 (0x00 bis 0x1F hexadezimal) enthalten ist. Dazu kommt das folgende High-Nibble, das angibt, ob eine Timerzeit mitgesendet wurde und somit der Befehl mit 32 bis 63 (0x20 bis 0x3F hexadezimal) beim FS20 PCS vergleichbar wäre. Das folgende Low-Nibble und die nächsten zwei Byte ergeben als Zeichenkette im Hexadezimalformat den Zeitfaktor, welcher, mit 0,25 Sekunden multipliziert, die Timerzeit ergibt. Als letztes Byte wird noch die Firmwareversion des FS20 PCE übergeben, wobei das High-Nibble die Stelle vor und das Low-Nibble die Stelle hinter dem Punkt ist.

Aufbau der Oberfläche

Als Oberflächentechnologie wurde WPF (Windows Presentation Foundation) verwendet, wir beschreiben hier die einzelnen Bedienoberflächenelemente (UI-Elemente) in XAML (ähnlich aufgebaut wie XML). Die Bearbeitung der Oberfläche kann in der Entwurfsansicht durch „Zusammenklicken“ der UI-Elemente oder wahlweise direkt im XAML erfolgen. Wie unter anderem bei „Windows Forms“ gibt es eine Quelltextdatei im Hintergrund, die die Oberfläche mit der Programmlogik verknüpft. WPF bietet bei der Bearbeitung gegenüber Windows Forms den Vorteil,

```
<TextBlock Grid.Row="2" Grid.Column="1" Margin="5" HorizontalAlignment="Center">Hauscode</TextBlock>
<TextBox Grid.Row="2" Grid.Column="21" HorizontalAlignment="Center" VerticalAlignment="Top"
         Margin="0,5,0,0" MaxLength="8" Width="60" Name="txbHauscode"
         Text="{Binding Hauscode)" TextChanged="TxbHauscode_TextChanged" />
```

Bild 2: Ein Beispiel für das Data-Binding unter WPF

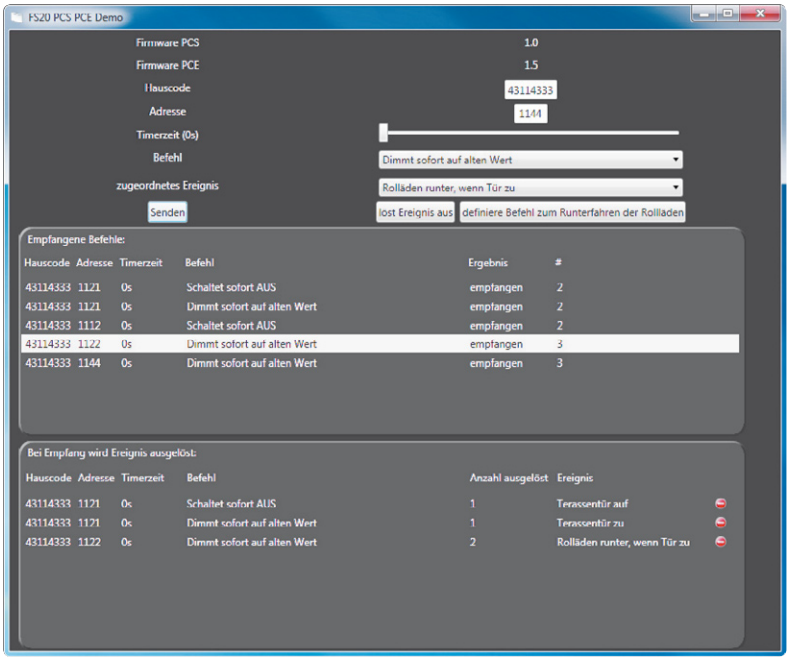

Bild 3: Die Oberfläche der Demoanwendung

dass beispielsweise an eine Textbox direkt eine Variable gebunden wird (siehe Bild 2), dies wird Data-Binding genannt. Für den hierarchischen und geordneten Aufbau wurden Grid-Panels (Tabellen) gewählt, in welche die anderen Elemente der Bedienoberfläche platziert wurden.

Mit der Demoanwendung (siehe Bild 3) lassen sich sowohl FS20-Befehle mittels der Oberfläche erzeugen als auch mittels des FS20 PCE und PCS versenden bzw. empfangen. Ebenfalls können empfangene Befehle einem Ereignis zugeordnet werden, so dass beispielsweise der FS20-Befehl „Schaltet sofort auf AUS“ mit dem Ereignis „Terrassentür zu“ verknüpft wird und dies entsprechend in der Software verarbeitet wird.

Die aktuelle Firmware-Version des FS20 PCE/PCS wird in den ersten beiden Zeilen der Anwendung ausgegeben. In den darauf folgenden Zeilen folgen die Elemente zur Erstellung des FS20-Befehls, so lassen sich Hauscode, Adresse, Timerzeit und der eigentliche Befehl mit entsprechenden Elementen der Bedienoberfläche einstellen. Die Übertragung des zusammengestellten Befehls mit dem FS20 PCS erfolgt sinngemäß mit dem darunter platzierten Button „Senden“.

Wenn der FS20 PCE FS20-Befehle vom FS20 PCS oder anderen FS20-Geräten empfängt, werden diese in der unter dem „Senden“-Button befindlichen Liste dargestellt. Diese gruppiert gleiche Befehle, um die Anzahl der empfangenen Befehle möglichst übersichtlich zu halten, und inkrementiert bei Empfang zweier gleicher Befehle den dazugehörigen Zähler auf der rechten Seite.

Die eigentliche Intention der Anwendung kommt erst in der Verknüpfung der empfangenen FS20-Befehle mit Ereignissen zum Tragen. Diese wird mit dem Button „löst Ereignis aus“ erstellt und nutzt sowohl die zuvor erläuterten Elemente zur Definition eines FS20-Befehls als auch das Kombinationsfeld zur Auswahl eines Ereignisses über den Button. Unter der Bedingung, dass der verknüpfte FS20-Befehl vom FS20 PCE empfangen wurde, wird das jeweilige Ereignis ausgelöst und entsprechend verarbeitet. Nicht mehr erwünschte Verknüpfungen können mit dem Entfernen-Button eines jeden Eintrags auf der rechten Seite aus der Liste gelöscht werden.

Ein Ereignis, welches aus dem zuvor genannten Kombinationsfeld ausgewählt werden kann, ist hier das Herunterfahren der Rollläden, wenn die Terrassentür geschlossen ist. Um den hierfür notwendigen FS20-Befehl zu definieren, wird der Button „definiere Befehl zum Runterfahren der Rollläden“ genutzt, welcher sich rechts vom Button zum Verknüpfen von Ereignissen mit FS20-Befehlen befindet. Dieser speichert den aktuellen FS20-Befehl, basierend auf der Bedienoberfläche, zwischen, der beim Auslösen des Ereignisses letztlich gesendet wird.

Bedingtes Versenden von FS20-Befehlen

Bevor auf Ereignisse reagiert werden kann, müssen diese, wie zuvor erläutert, mit FS20-Befehlen verknüpft werden.

Die auswählbaren Ereignisse sind:

- Kein Ereignis
- Terrassentür auf
- Terrassentür zu
- Rollläden runter, wenn Tür zu

Der Empfang eines FS20-Befehls, welcher mit dem Ereignis „kein Ereignis“ verknüpft ist, wird von der Software logischerweise nicht gesondert verarbeitet. Die Anwendung arbeitet mit einer Booleschen Variablen, die den Status der Terrassentür repräsentiert. Diese wird beim Eingang eines FS20-Befehls, der mit einem Ereignis um die Terrassentür verknüpft ist, entsprechend editiert. Bei der Verarbeitung des verbleibenden Ereignisses wird sie letztlich genutzt, so dass erst nach einer erfolgreichen Prüfung, ob die Terrassentür geschlossen ist, der zuvor vordefinierte FS20-Befehl zum Schließen der Rollläden gesendet wird. Wenn diese Definition zuvor nicht stattgefunden hat, wird eine entsprechende Fehlermeldung ausgegeben und kein Befehl gesendet.

Fazit

Anhand dieses Beispiels kann man nachvollziehen, wie das FS20-System für eigene Anforderungen gezielt einsetzbar ist. Auf dieselbe Weise können auch unterschiedlichste andere Anwendungsfälle realisiert werden, so dass nur mit FS20-Komponenten eine intelligente, an die eigenen Bedürfnisse angepasste Haussteuerung realisierbar ist. ELV

Weitere Infos:

Demoanwendung FS20 PCS und PCE Download:
www.elv.de: Webcode: #1242

Entwicklungsumgebung Download:
www.microsoft.com/visualstudio/en-us/products/2008-editions/express

C# HID Klassenbibliothek:
www.codeproject.com/KB/cs/USB_HID.aspx