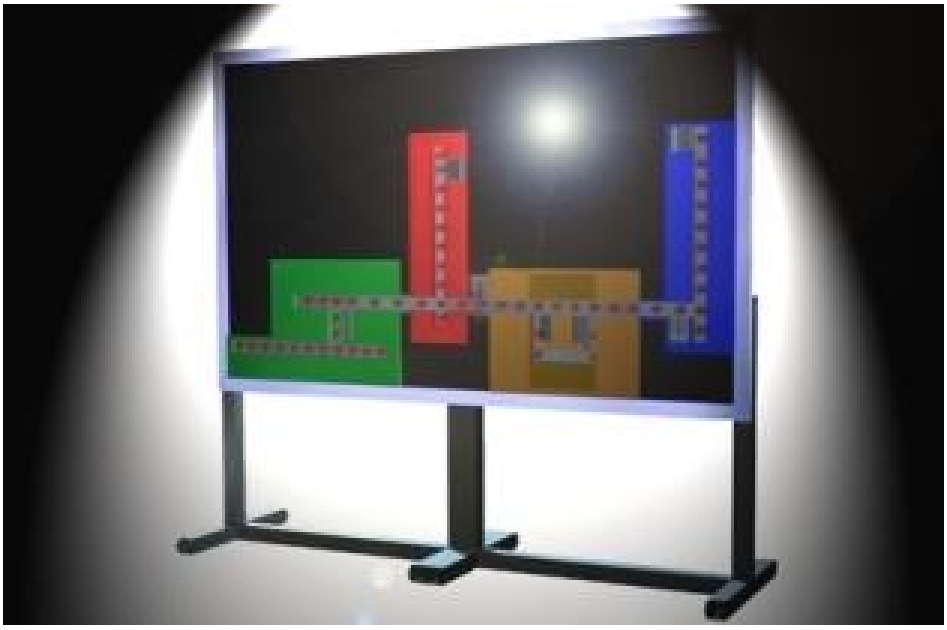
Xplore –New Automation Award 2008
Abschlussdokumentation

GoBuilding

Entwicklung eines Empfangs- und Wegeleitsystems für Besucher der BBS 1 Mainz

Fachschule Automatisierungstechnik, BBS 1 Mainz

Das Team
Marcus Astheimer
Hans-Jürgen Reck
Oliver Schneider
Robert Schönfelder



Betreuende Lehrer/in
Herr Löser, Herr Musielack, Frau Führich-Albert

Inhaltsverzeichnis

1. Einleitung

1.1. Vorwort

Diese Dokumentation entstand im Rahmen des internationalen Automatisierungs-Wettbewerbs XPLORE und der Weiterbildungsmaßnahme zum staatlich geprüften Techniker für Produktions- und Prozessautomatisierung innerhalb des Moduls Projektarbeit an der Fachschule für Automatisierungstechnik der BBS 1 in Mainz.

1.2. Die Projektidee GoBuilding

Mit dem Projekt „GoBuilding“ sollte für die BBS 1 in Mainz ein System realisiert werden, welches Besuchern, Schülern aber auch Lehrern ein einfach zu bedienendes Empfangsportal und Leitsystem bietet.
Die Vorgabe lautete ein Leitsystem für die Schule zu erarbeiten, welches aus einem Empfangsterminal und einem Modell der Schule besteht.

Den ersten Einsatz hatte unser System am Infotag der BBS 1 am 9. Februar 2008. An diesem Tag fanden auch die Projektpräsentationen der Fachschule Automatisierungstechnik statt. Unser System stand am geplanten Platz im Foyer der Schule und wurde hier von den Besuchern der Schule auf „Herz und Nieren“ getestet.

1.3. Die BBS 1 in Mainz, Zahlen und Fakten

An der BBS 1 Mainz lernen Ca. 5000 Schüler und unterrichten 180 Lehrkräfte.



Die Schule teilt sich in 4 Gebäudeteile auf, einen Werkstattflügel mit 2 Ebenen,
ein Hauptgebäude mit 6 Ebenen und daran direkt angeschlossen 2 Seitenflügel mit jeweils 5 Ebenen.

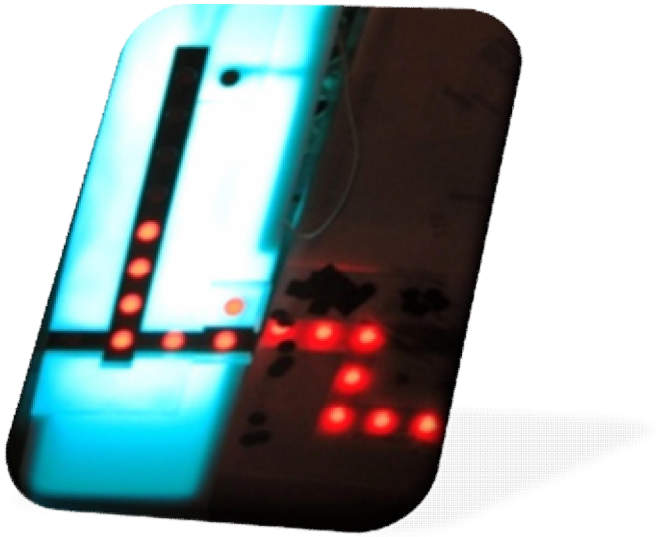

Insgesamt stehen so verteilt auf den 18 Ebenen 280 Klassen- und Fachräume zur Auswahl.

2. Umsetzung des Projekt GoBuilding

Beschreibung der Funktionsweise vom Plexiglasmodell

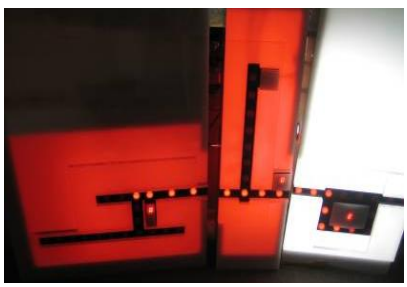
Aufgabe war es, ein Konzept für die Ansteuerung der Wegeanzeige (LED's) zu erstellen, das für den Besucher leicht verständlich ist. Zuerst war geplant, die LED's einfach nur aufleuchten zu lassen. Bei dieser Variante wäre es jedoch für die Besucher schwierig geworden zu erkennen, welches Treppenhaus er benutzen soll. Die Lösung dieses Problems ist ein Lauflicht.

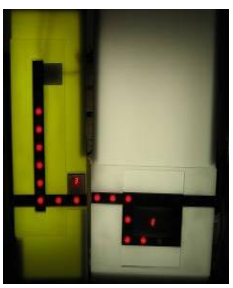
Wählt der Besucher nun einen Raum im Erdgeschoss aus, so wird ein Lauflicht vom Ausgangspunkt bis zum gewünschten Raum angezeigt. Die Stockwerksfarbe (verschieden farbige Leuchtstofflampen) wird nur in den Gebäudeteilen angesteuert, welche durchquert werden müssen.



Wird nun ein Raum auf einer anderen Ebene ausgewählt, so wird mit einem Lauflicht der Weg zum Treppenhaus angezeigt und die LED unter dem Treppenhausmodul fängt an zu blinken. Dieser Weg leuchtet jetzt dauerhaft und das Lauflicht hat jetzt das Treppenhaus als Ausgangspunkt.
Um dies auch für den W-Bau zu realisieren, muss das Treppenhaus im C-Bau benutzt werden, wenn der Zielraum im Kellergeschoss des W-Baus liegt.

Für die Anzeige der Ebene wird der Gebäudeteil in der jeweiligen Stockwerksfarbe beleuchtet um dem Besucher anzuzeigen auf welcher Ebene sich der Raum befindet.

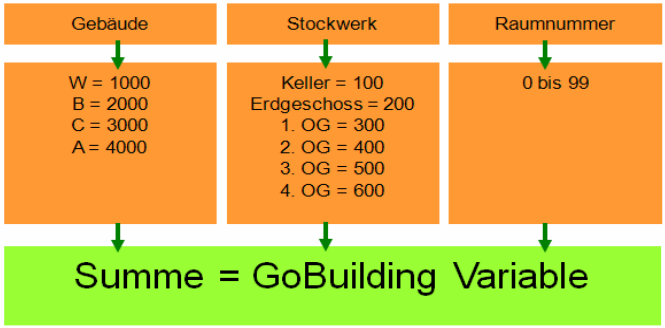




2.1. Der OPC Server

2.1.1. Die Grundüberlegung der OPC Variablen

Bereits kurz nach der Erstellung der Topologie und den Raumlisten (280 Räume) war klar, dass wir zur Lösung der Programmieraufgabe sowohl PC WorX als auch bei der Visualisierung im Projekt GoBuilding ein Variablensystem benötigen, das einfach zu handhaben ist und uns eine parallele Programmierung ermöglicht. Ergebnis dieser Überlegungen war unsere GoBuilding Variable, die es ermöglicht, alle Informationen in einem Variablenwert auszudrücken. Diese Grundüberlegung war notwendig, um ein Variablenchaos im Projekt zu vermeiden. Mit der Idee der GoBuilding Variablen war somit klar, dass anstatt vieler einzelner Variablen nur ein einziger Zahlenwert zwischen Steuerung und Visualisierung ausgetauscht werden muss.
In Verbindung mit der OPC Funktionalität der Variablen war somit der Grundstein für den Datenaustausch zwischen Steuerung und der Visualisierung gelegt.



2.1.2. Die GoBuilding OPC Variablen im Projekt

GoBuilding,
diese OPC Variable wird von dem PC WorX Programm gelesen. Die Visualisierung schreibt einen Zahlenwert in die GoBuilding Variable, der je nach Stockwerk, Gebäudeteil und Raumnummer einen fest definierten Wert hat. Das SPS Programm entnimmt diesem Wert lesend die benötigte Rauminformation.

GoBuildingFeedback,
diese Variable ist ebenfalls in dem WorX Baustein enthalten und wird von der Visualisierung gelesen. In dem Debug und Testprogramm wird diese Variable mittels Move Baustein aus der GoBuilding Variable erzeugt. Auf diese Variable reagiert die Visualisierung mit der passenden Bildschirmanzeige.

Visualisierung_Aktivieren Variable,
diese Variable war ursprünglich nicht geplant, wurde aber benötigt, um die Visualisierung der Ergebnisanzeigen zu vereinfachen. Sie wird wie die GoBuilding Feedback Variable von dem WorX Programm erzeugt, enthält aber nur die Gebäude und Stockwerksinformationen. Die Erzeugung dieser Variable wurde ebenfalls in den Visu_Transfer Baustein integriert.
Durch diese zusätzliche Variable ist es möglich, Wegbeschreibungen stockwerks- und gebäudebezogen anzuzeigen.

2.2. Die Visualisierung

Die Visualisierung ist der Teil in dem Projekt, mit dem der Benutzer später seine Suche ausführt. Sie ist sowohl für die Suchanfrage und die Weiterleitung an die Steuerung verantwortlich, aber auch als Anzeige der gewünschten Route zu einem gesuchten Klassenraum.

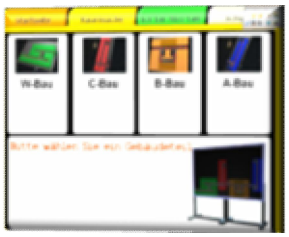
Als Eingabe dient ein Touchscreen in unserem Terminal. Daher ist ein Aspekt die „Fingerfreundlichkeit“ der Visualisierung. Dies bedeutet, dass die Buttons und Bedienelemente so angeordnet sein müssen, das man diese mit dem Finger betätigen kann.
Ein wichtiger Aspekt ist deshalb die Benutzerfreundlichkeit des Systems. Es soll jedem Besucher der Schule ermöglichen, mit einfachen Klicks auf dem Touchscreen das gewünschte Ziel zu finden, ohne eine Anleitung oder Einweisung in unserem System zu haben.

2.2.1. Die Gliederung der GoBuilding Visualisierung

Startseite

Dies ist das erste Registerblatt auf der Visualisierungsoberfläche. Es enthält eine Begrüßung und allgemeine Informationen zur BBS1.
Diese Oberfläche enthält keinerlei aktive Navigationsoptionen und ist eine rein informative Seite, um den Besucher am Terminal zu begrüßen.

Raumsuche

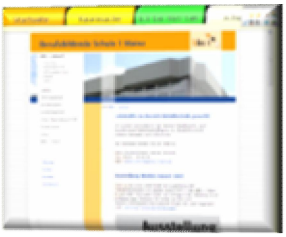


Dieses Registerblatt ist für die Schüler und Lehrer an der BBS1 gedacht. Man kennt die Raumnummer, weiß aber nicht, wo sich dieser befindet. Hier schafft die Raumsuche Abhilfe, in drei Schritten wird man zum passenden Raum geführt. Diese führt über eine Gebäude- und Stockwerksauswahl zur Anzeige der betreffenden Räume, die auf dieser Ebene liegen.

Detailsuche

Der Quickfinder ist die Funktion für alle Besucher der BBS1. Über einfache Buttons wird der Besucher direkt zu einem Raum oder einer Auswahlseite geführt. In dieser Suchmaske ist es nicht erforderlich, die Raumnummer zu kennen. Es stehen vielmehr besucherorientierte Suchfunktionen zur Verfügung wie z.B. die Abteilungsleitersuche oder eine Fachbereichssuche, die dann zum jeweiligen Ansprechpartner in der Schule führen.

Homepage der BBS1



Für weiterführende Informationen ist die Registerkarte WWW gedacht. Diese öffnet die Schulhomepage, auf der dann weitere Informationen über die Schule abgerufen werden können.

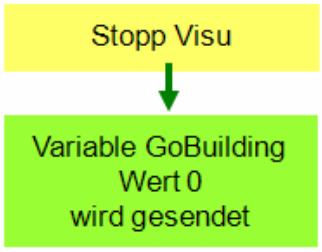
2.2.2. Grundfunktionen der GoBuilding Visualisierung

Je nach Auswahl wird der Wert der GoBuilding Variable und somit auch die GoBuildingFeedback Variable verändert. Somit wird je nach Wertigkeit und Auswahl dem Benutzer der passende Bildschirminhalt mit entsprechenden Auswahlmöglichkeiten angezeigt. Eine weitere Besonderheit ist die Startbildstruktur im Projekt. Die Anfangsbilder werden mit der GoBuildingFeedback Wertigkeit 5 angezeigt. Somit kann die Steuerung jederzeit wieder zum Startbild wechseln.
Erste Überlegung waren die Ausgangszustände der Visualisierung.
Die Wertigkeit 5 der GoBuilding Variable wurde als Ausgangszustand für Visualisierung und Steuerung festgelegt.
Die Werte zwischen 5 und 100 wurden für die Visualisierungsfunktionen reserviert.

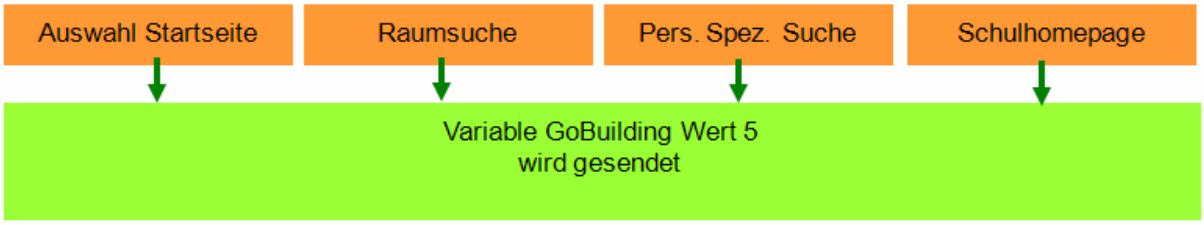
Start der Visualisierung
Beim Start der Visualisierung wird der Wert 5 in die GoBuilding Variable geschrieben. Dies ist für die Steuerung der Ausgangswert zur Aktivierung des Programms. Beleuchtungseffekte sind aktiv.

Die Steuerung wartet auf eine Raumauswahl.

Beenden der Visualisierung
Wird die Visualisierung beendet, um den Terminalrechner runterzufahren, wird der Wert 0 in die GoBuilding Variable geschrieben. Dies versetzt die Steuerung in den Programmzustand AUS. Lichteffekte und alle Programmfunktionen werden deaktiviert, der Rechner wird automatisch runtergefahren und ausgeschaltet.



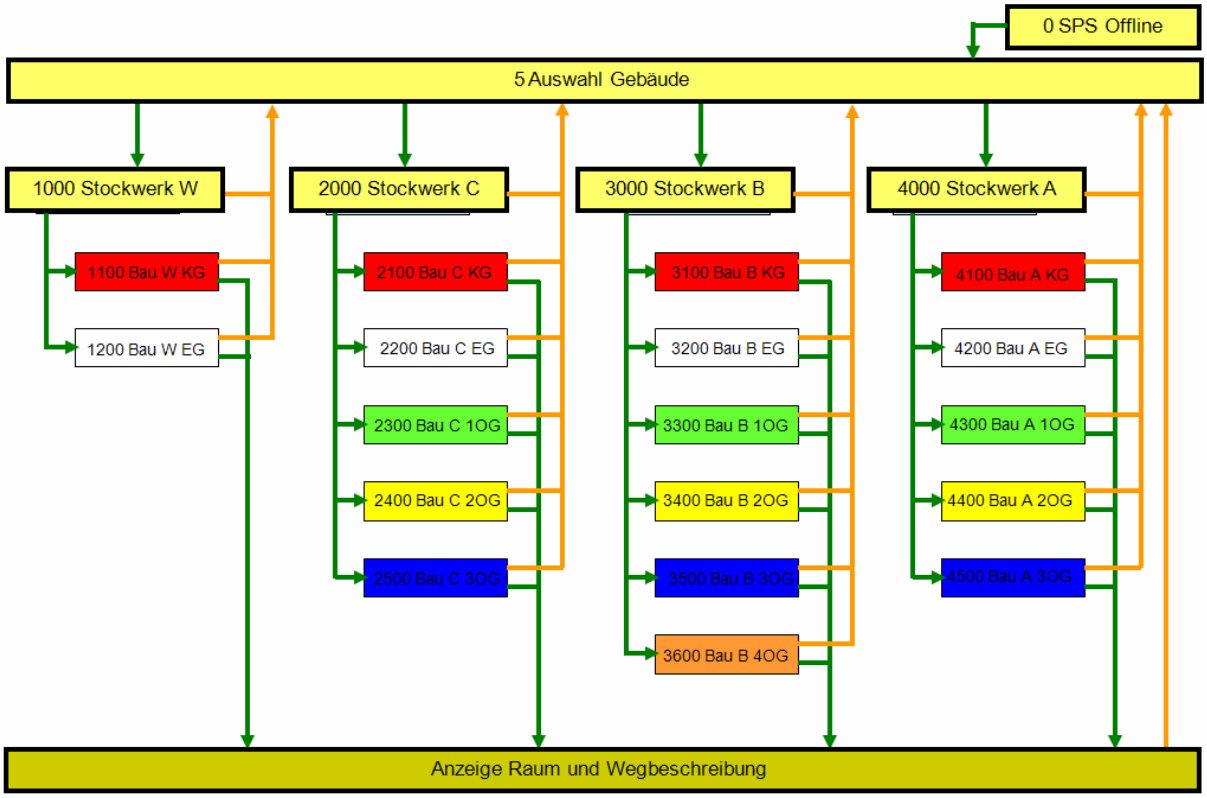
Auswahl der Hauptseiten
Beim Wechsel der Hauptvisualisierungsansichten wird ebenfalls der Wert 5 in die GoBuilding Variable geschrieben. Dies sorgt dafür, dass die aktuelle Navigation auf dem Plexiglasmodell beendet wird, wenn der Benutzer eine neue Ansicht auswählt.



Die Steuerung wartet jetzt wieder auf eine Raumauswahl. Beleuchtungseffekte sind aktiv.

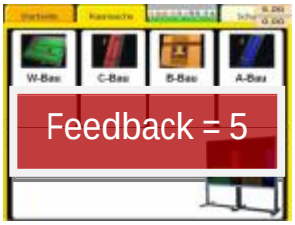
2.2.3. Die Raumsuche

Die Raumsuche befindet sich im Visu+ Projekt unter dem Screen DeRaumsuche. In diesem Screen befinden sich insgesamt 291 Objekte verteilt auf 29 Layer. Die Objekte haben 573 Verknüpfungen zur GoBuilding-, GoBuildingFeedback- bzw. Visualisierung_Aktivieren Variable.

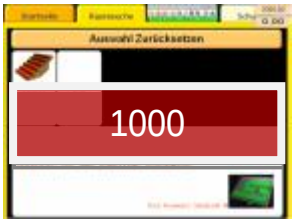


Die Zahlenwerte in diesem Schema geben an, bei welchem GoBuildingFeedback Wert die einzelnen Objekte (Schaltflächen bzw. Bilder) angezeigt werden.

Ist die Steuerung online, hat die GoBuildingFeedback Variable den Wert 5 und es wird die Hauptauswahl angezeigt. Diese umfasst 5 Objekte, je einen Button für den W-Bau, C-Bau, B-Bau und den A-Bau und ein Bild mit einem kurzen Erklärungstext. Diese Objekte sind nur mit dem Wert 5 sichtbar.

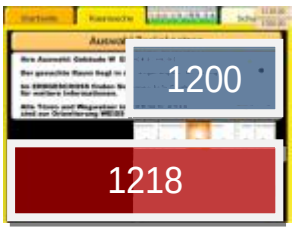


Erfolgt nun die Auswahl W-Bau, wird an die Steuerung der Wert 1000 gesendet. Auf der Feedback Variable wird dieser Wert 1000 wieder zurückgegeben. Jetzt werden 3 Objekte sichtbar, 2 Buttons für die Auswahl Keller oder Erdgeschoss und ein Bild mit Beschreibung. Diese 3 Objekte sind nur bei dem Wert 1000 sichtbar.
Zusätzlich wird noch ein Objekt „Auswahl Zurücksetzten“ aktiv. Dieses ist immer sichtbar, wenn die Feedback Variable einen größeren Wert als 5 hat und sendet bei Betätigung die 5 auf der GoBuilding Variable und löst somit einen Sprung auf die Hauptauswahl aus.



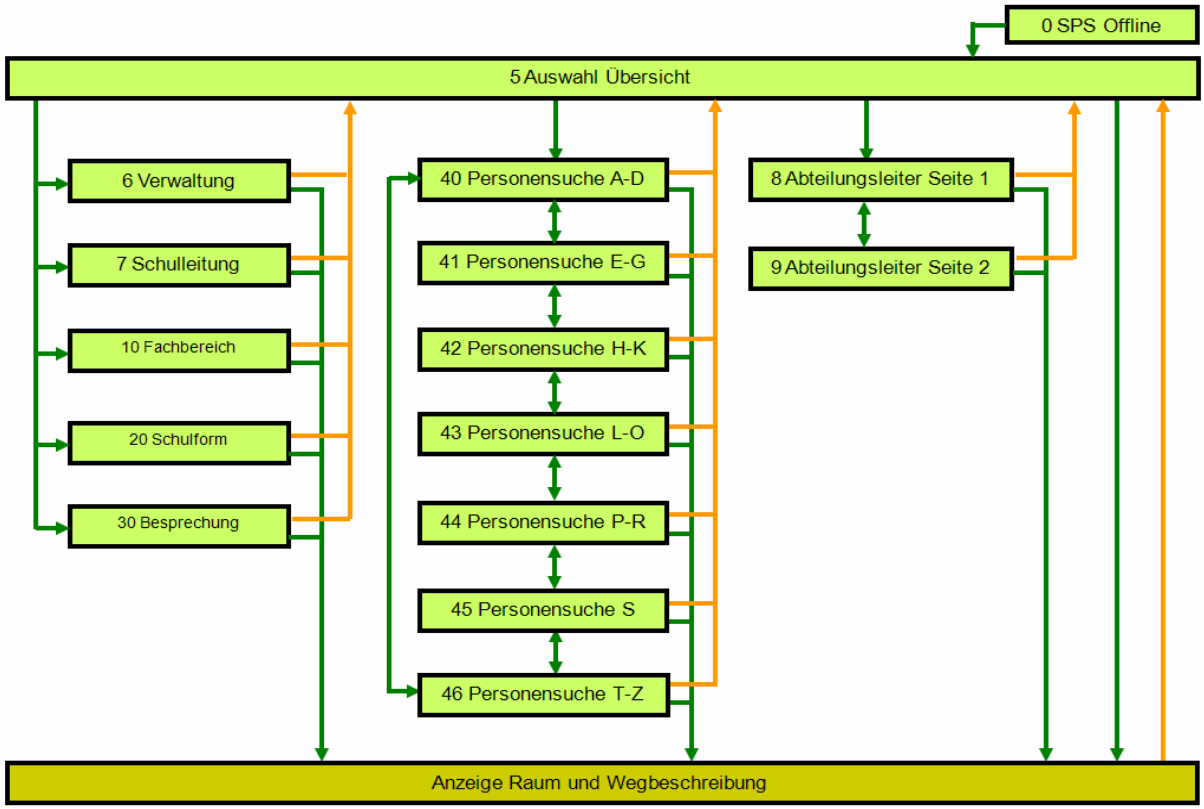
Wird nun als nächste Auswahl das Erdgeschoss ausgewählt, wird die 1200 in die GoBuilding Variable geschrieben.
Dieser Wert steht für W-Bau (1000) und Erdgeschoss (200).
Bei dem Wert 1200 werden nun alle Räume angezeigt, die im W-Bau Erdgeschoss zur Auswahl stehen.

Wurde dann der Raum W 18 ausgewählt, wird die 1218 in die GoBuilding Variable geschrieben. Auf der Feedback Variable aktiviert jetzt die 1218 die Karte zum Raum W18.
Hier kommt jetzt noch die Visualisierung_Aktivieren Variable zum Einsatz. Diese wird von dem Verarbeitungsbaustein der Steuerung auf 1200 gesetzt. Mit dieser Variable werden je nach Gebäude und Stockwerk die Wegbeschreibungen und weitere Informationen angezeigt.

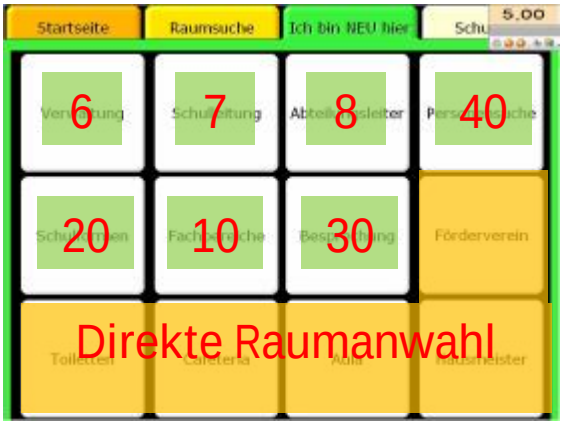


2.2.4. Die Detailsuche

Das Funktionsprinzip der Detailsuche ist das gleiche wie auch bei der Raumsuche.
Der einzige Unterschied liegt darin, dass hier noch Werte zwischen 5 und 46 verwendet werden, um die Auswahlseiten zu aktivieren.



Die Zahlenwerte in diesem Schema geben an, bei welchem GoBuildingFeedback Wert die einzelnen Objekte (Schaltflächen bzw. Bilder) angezeigt werden.
Auf der Hauptseite befinden sich 5 Buttons, mit denen die Toiletten, Cafeteria, Aula, Hausmeister und der Förderverein verknüpft sind. Bei Auswahl dieser werden direkt das Zielbild und die Wegbeschreibung aktiviert, die zu diesen Räumen führt. Das Funktionsprinzip ist hier das gleiche wie auch bei der Raumsuche. Die GoBuildingFeedback Variable aktiviert das Zielbild und die Visualisierung_Aktivieren Variable die Wegebeschreibung.



2.3. Die Programmierung mit PC Worx

2.3.1. Die Programmbausteine

Die Herausforderung bei der Programmierung bestand darin, trotz des großen Programmieraufwands eine Struktur reinzubringen. Über die Variable GoBuilding werden ca. 280 verschiedene Räume in 4 Gebäudeteilen und bis zu 6 Stockwerken an die Steuerung gesendet.

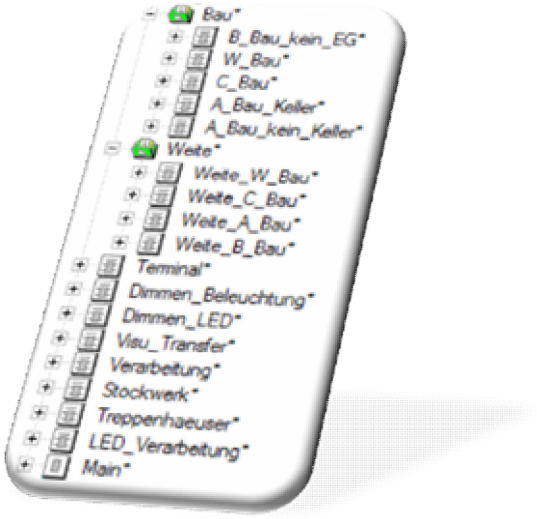
Das Programm wurde in mehrere Funktionsbausteine unterteilt, welche dann entweder zu einem weiteren Baustein zusammengefasst oder direkt im Main-Programm aufgerufen werden. Hierdurch bleibt das Projekt übersichtlich.

Die Funktionsbausteine lassen sich wie folgt einordnen:

- *Verarbeitung*
  Hier wird die GoBuilding Variable in ihre Bestandteile Bau, Stockwerk und Raum zerlegt.
  Programmiersprache: FBS

- *Visu_Transfer*
  Dieser Baustein dient dazu, die Visualisierung zu steuern. Aus der GoBuilding Variable, welche von der Visualisierung geschickt wird, werden die Variablen GoBuildingFeedback und Visualisierung_Aktivieren erzeugt.
  Programmiersprache: FBS

- *Weite*
  In diesen Bausteinen wird den Räumen eine sogenannte *Weite* zugeordnet. Diese wird für die Ablaufsprache benötigt.
  Programmiersprache: FBS



- *Stockwerk*
  Hier werden die Leuchtstofflampen mit der Farbe des anzuzeigenden Stockwerks angesteuert.
  Programmiersprache: FBS

- *Treppenhäuser*
  In den Treppenhäusern sitzt eine LED, welche durch Blinken anzeigt, welches Treppenhaus benutzt werden soll.
  Programmiersprache: FBS

- *Bau*
  Diese Bausteine dienen dazu, die LED's (Wegeanzeige) in Abhängigkeit von der *Weite* anzusteuern.
  Programmiersprache: AS

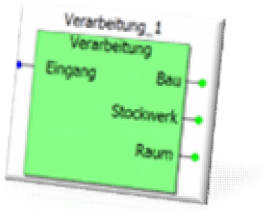
- *Terminal*
  Hier wird das Signal des Bewegungssensors verarbeitet und das Relais, welches den Monitor im Terminal an- und ausschaltet, angesteuert. Hier wird die Anlage auch in den sogenannten "Standby"-Modus gesetzt. Dies geschieht durch Überschreiben der GoBuilding Variable.
  Programmiersprache: AS

- *Dimmen_LED*
  Damit man die „Start-LED" (Ausgangspunkt im B-Gebäude) von den restlichen LED's unterscheiden kann, pulsiert diese.
  Programmiersprache: AS

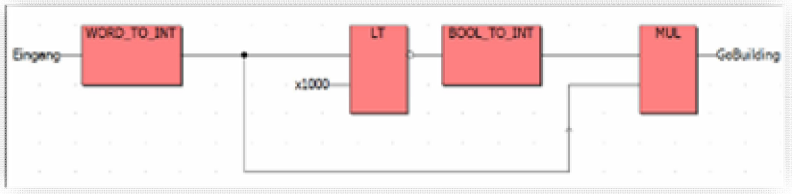
- *Dimmen_Beleuchtung*
  Um auf das Leitsystem im Foyer aufmerksam zu machen, wurden im Terminal als auch im Plexiglasmodell Beleuchtungen installiert. Diese werden über ein Dimmer-Modul pulsierend angesteuert.
  Programmiersprache: AS

Die Funktionsbausteine Verarbeitung, *Weite, Bau* und *Treppenhäuser* sind in dem Baustein *LED-Verarbeitung* zusammengefasst.

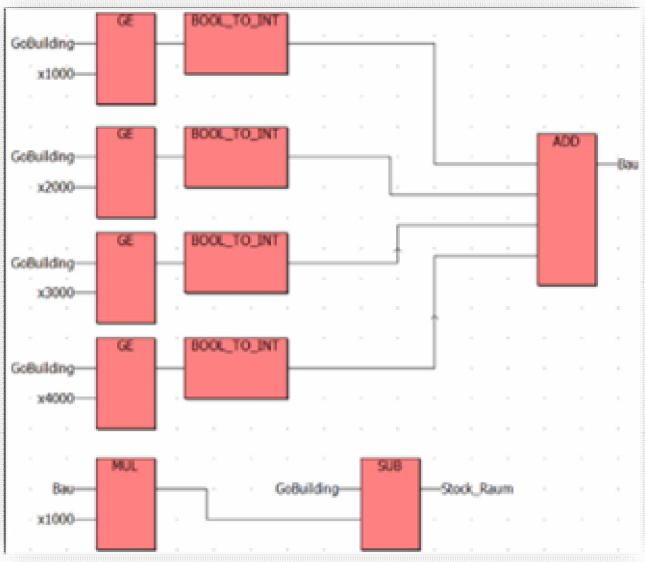
2.3.1.1. Der „*Verarbeitung*“-Baustein



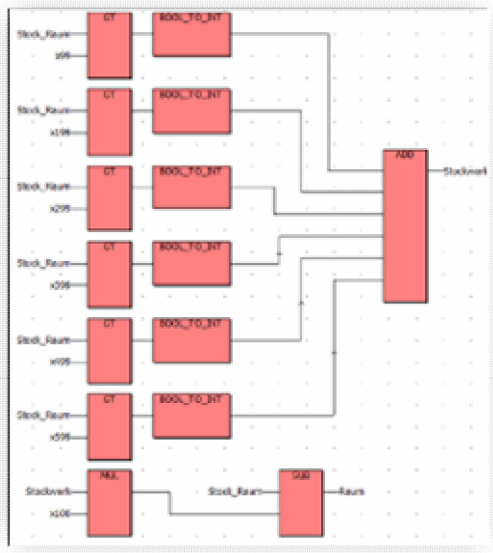
Dieser Baustein hat einen Eingang, dem man die globale OPC-Variable GoBuilding im Datentyp Word zuweist. Als Ausgang werden die Variablen Bau, Stockwerk und Raum im Datentyp Integer ausgegeben. Dies wird benötigt, um die weitere Verarbeitung zu erleichtern.
Als erstes wird der Eingangswert vom Datentyp Word nach Integer konvertiert. Mit Hilfe des Vergleichs auf kleiner 1000 wird die Variable erst weitergegeben, wenn der Wert höher ist, sonst wird eine 0 gesendet.



Die weitere Verarbeitung der Variable besteht darin, alle Informationen rauszuholen. Beginnen wir damit, den Gebäudeteil zu filtern. Dies geschieht mit Hilfe eines Größer-Vergleiches.
Die Ergebnisse werden dann addiert, nachdem sie vom Datentyp Bool zu Integer konvertiert wurden.
Nun haben wir die erste Information. Um an die Weiteren zu kommen, wird von der Eingangsvariable die Stockwerksinformation subtrahiert.

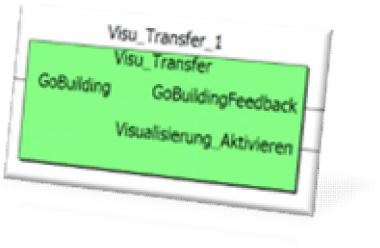


Jetzt erhalten wir die Variable Stock_Raum. *Der Name resultiert auf den restlichen Informationen, welche noch in ihr stecken.*

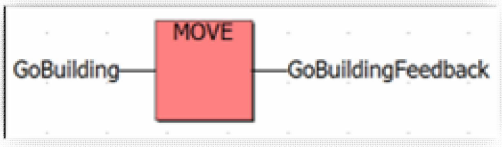


Das Vorgehen wiederholt sich jetzt nochmals, um an die Information des Stockwerks zu kommen. Mit Hilfe des Größer-Vergleichs bekommen wir dann nach der Addition die Ausgangsvariable Stockwerk. Jetzt wird auch hier von der Variable Stock_Raum die Stockwerksinformation subtrahiert und man erhält letztlich die Raum Variable.
Jetzt haben wir alle Informationen aus der GoBuilding Variable herausgeholt und können die Integer Werte nun für die weitere Programmierung verwenden.

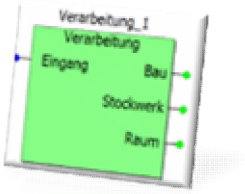
2.3.1.2. Der *„Visu_Transfer“*-Baustein



Da die Visualisierung auf die Variablen GoBuildingFeedback und Visualisierung_Aktivieren Aktionen ausführen soll, müssen sie aus der „GoBuilding“ Variable erzeugt werden.
Dies sind dann auch schon alle benötigten Ein- und Ausgangsvariablen im Datentyp Word.



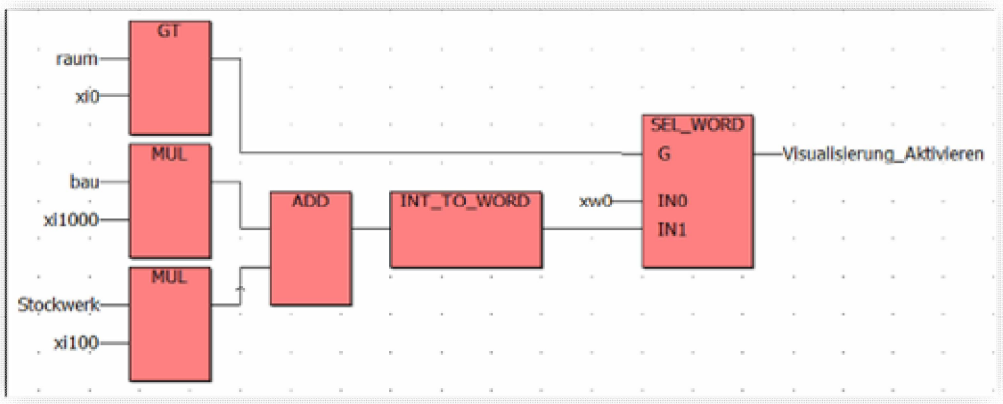
Über die GoBuildingFeedback Variable erkennt die Visualisierung, ob die Steuerung im „Run“-Modus ist. Für diese Erkennung wird mittels eines „Move“-Bausteins der Wert der GoBuilding in die GoBuildingFeedback Variable geschrieben. Diese OPC-Variable sendet den Wert dann wieder zurück an die Visualisierung.



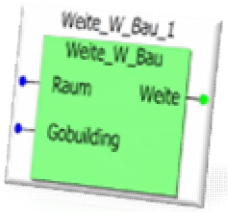
Für die Bearbeitung der zweiten Ausgangsvariable werden zuerst mittels des Bausteins „Verarbeitung“ die Variablen für den Bau, das Stockwerk und den Raum erzeugt. Diese werden für die weitere Verarbeitung benötigt.

Durch die Visualisierung_Aktivieren Variable soll die Visualisierung eine Wegbeschreibung anzeigen. Da wir aus zeitlichen und personellen Gründen nicht für jeden der 280 Räum eine eigene Wegbeschreibung erstellen konnten, wurde diese auf den Bau und das Stockwerk beschränkt.

Hierzu werden die Variablen „Stockwerk“ und „Bau“ mit ihrer ursprünglichen Wertigkeit (aus der GoBuilding Variable) multipliziert und anschließend addiert. Da die Ausgangsvariable den Datentyp Word hat, muss der erzeugte Integer Wert noch konvertiert werden.
Die Wegbeschreibung soll natürlich nur angezeigt werden, wenn der Benutzer einen Raum ausgewählt hat. Hierfür wird ein Größer-null-Vergleich mit der Raum-Variablen durchgeführt. Die Funktion des letzten Bausteines ist eine Selektion zweier Eingangswerte des Datentyp Word („IN0“ und „IN1“). Wird jetzt am Eingang „G“ ein „low“ Signal gesendet, so wird der Wert, welcher bei „IN0“ anliegt, auf den Ausgang geschrieben, bei einem „high“ Signal der Eingang „IN1“.



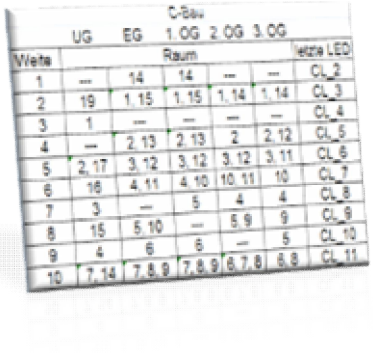
2.3.1.3. Der „*Weite*“-Baustein



Für jeden Bau existiert ein Weite-Funktionsbaustein, welcher jedem Raum eine Weite zuweist.

Wozu wird die Weite überhaupt benötigt?

Sie ist ein Index dafür, wie viele LED's im ausgewählten Bau als Weganzeige angehen sollen. Die Variable wird in den Funktionsbausteinen „*Bau*“ verwendet und dient dort als Weiterschaltbedingung (Transition).

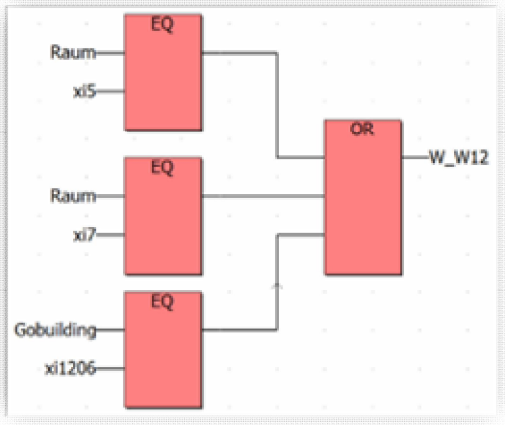

C-Bau

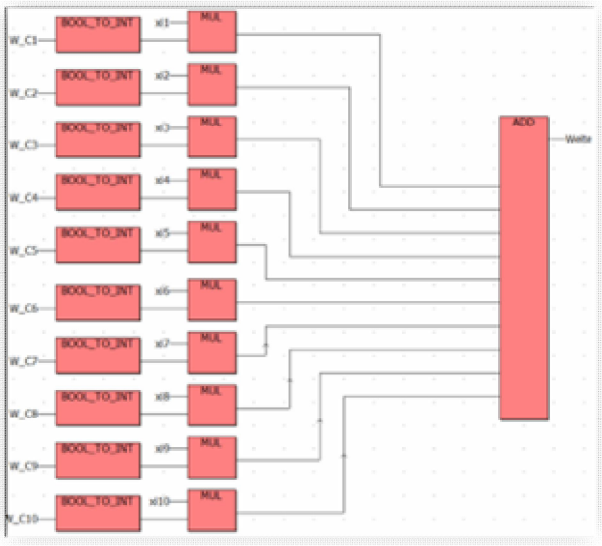
Weite	UG	EG	1.OG	2.OG	3.OG	letzte LED
	Raum					
1	---	14	14	---	---	CL_2
2	19	1, 15	1, 15	1, 14	1, 14	CL_3
3	1	---	---	---	---	CL_4
4	---	2, 13	2, 13	2	2, 12	CL_5
5	2, 17	3, 12	3, 12	3, 12	3, 11	CL_6
6	16	4, 11	4, 10	10, 11	10	CL_7
7	3	---	5	4	4	CL_8
8	15	5, 10	---	5, 9	9	CL_9
9	4	6	6	---	5	CL_10
10	7, 14	7, 8, 9	7, 8, 9	6, 7, 8	6, 8	CL_11

Hier ist eine kurze Übersicht über die „*Weite*“ im C-Bau, welche in der ersten Spalte angegeben ist.
In den nächsten fünf Spalten sind die Räume zu den entsprechenden „*Weite*“ eingetragen.
Als Orientierung ist in der letzten Spalte dann die LED angegeben, welche als letzes angesteuert wird.

Die Bausteine haben als Ausgang nur die ermittelte „*Weite*“ in Integer und als Eingang entweder die Variablen Raum und GoBuilding oder nur GoBuilding im Datentyp Integer.

Kommen wir nun zur Erläuterung, wie den Räumen die Weite zuordnet wird. Dabei wurden zwei Varianten verwendet. Einmal über die direkte Raumnummer, dies ist aber nur möglich, wenn der Raum auf allen Ebenen (Stockwerken) an der gleichen Position sitzt. Ist dies nicht der Fall, so geschieht die Zuweisung absolut über die Integer-Variable GoBuilding.
Für jeden Raum wird nun ein Vergleich durchgeführt. Hierbei wird die Variable mit der Raumnummer verglichen. Alle Räume mit derselben Weite werden dann auf einen Oder-Baustein gelegt. Als Ergebnis wird dann die zugehörige Variable „high“ gesetzt.

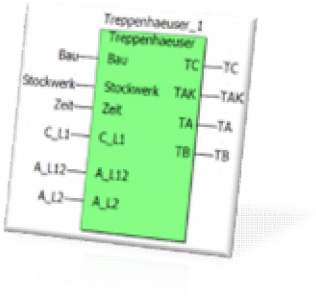




Da diese Variablen den Datentyp Bool haben, also nur zwei Zustände annehmen können (low oder high), werden diese am Ende des Funktionsbausteins weiterverarbeitet.
Hierzu werden sie in den Datentyp Integer konvertiert und mit dem zugehörigen nummerischen Wert multipliziert. Die Ergebnisse aller Multiplikationen werden dann auf einen Additions-Baustein gelegt und als Resultat haben wir die *„Weite“* Ausgangsvariable im Datentyp Integer.

2.3.1.4. Der *„Treppenhäuser“*-Baustein

Durch LED´s unter den Treppenhausmodulen wird dem Besucher angezeigt welches Treppenhaus er benutzen soll.

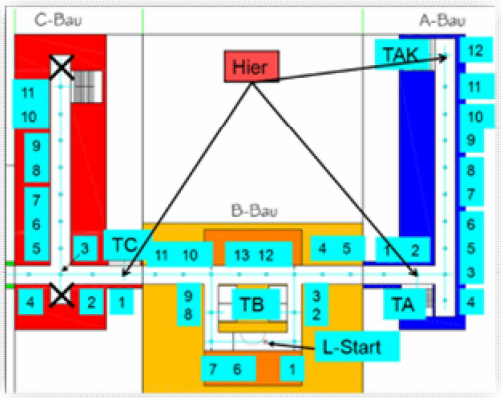


Durch die 7-Segment-Anzeige, welche im Treppenhausmodul integriert ist, wird das Stockwerk angezeigt. Die Ansteuerung der Anzeige erfolgt durch einen Mikrocontroller. Durch die blinkenden LED’s wird er dann auf das zu benutzende Treppenhaus hingewiesen.

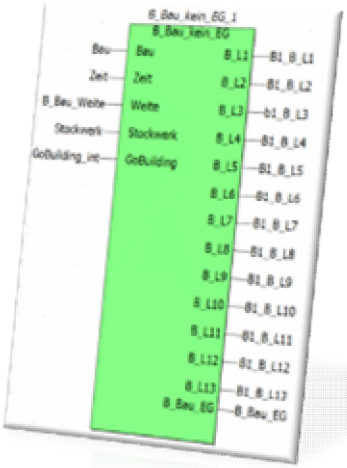
Dieser Baustein steuert alle Treppenhaus LED’s an. Als Eingänge hat er die Integer-Variablen Bau und Stockwerk, die Time-Variable Zeit, über welche man die Blinkgeschwindigkeit einstellt und 3 Eingangsvariablen im Datentyp Bool.

Diese drei Variablen sind die LED’s, welche direkt vor den Treppenhäusern sitzen (siehe rechts). Das hier nur drei benötigt werden, liegt an der Ansteuerung der Weganzeige.

Die vier Ausgangsvariablen sind dann die LED’s, welche direkt unter dem Treppenhausmodul sitzen *(„TC“, „TB“, „TA“, „TAK“).*

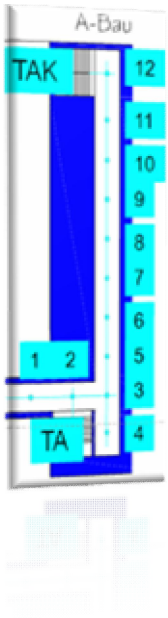


2.3.1.5. Der „*Bau*“-Baustein



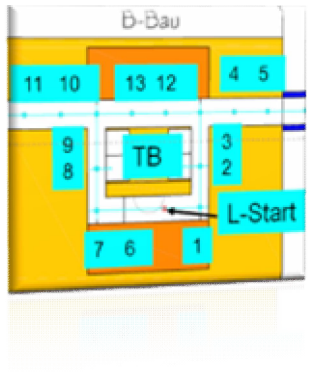
Dieser Baustein hat die Funktion, die LED's (Weganzeige) anzusteuern. Jeder Gebäudeteil hat mindestens einen Baustein. Der A- und B-Bau besitzen jeweils zwei Funktionsbausteine. Bei diesen zwei Gebäudeteilen war dies notwendig.

Der Grund dafür liegt darin:



A-Bau

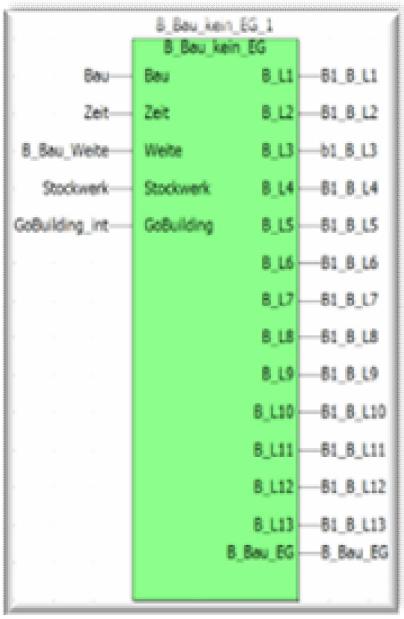
Der A-Bau besitzt zwei Treppenhäuser, ein vorderes (TK) und ein hinteres (TAK). Das Problem das aufkam, war, dass man durch das Vordere (TK) nicht in den Keller gehen kann. Dafür muss man das Hintere (TAK) benutzen. So sind wir auch auf die Namen gekommen: T=Treppenhaus, A=A-Bau, K=Keller.
Für die Programmierung hatte dies die Auswirkung, dass der Keller im A-Bau einen eigenen Funktionsbaustein bekommt. Dieser zeigt zuerst zweimal den Weg zum hinteren Treppenhaus (TAK) und dann den Weg zum ausgewählten Raum. Hierfür waren dann auch noch zusätzliche Variablen für die Leuchtstofflampen Ansteuerung nötig. Diese werden im Baustein „Stockwerk“ verwendet, um die richtige Stockwerksfarbe anzuzeigen.



B-Bau

Hier trat das Problem auf, dass das Treppenhaus (TB) mitten im Gebäude sitzt. Dadurch kann die Anzeige zum Treppenhaus nicht stehen bleiben, im Gegensatz zu den anderen Gebäudeteilen, da die LED's für die weitere Anzeige der Räume benötigt werden und die Zuordnung mit der Stockwerksfarbe wäre dann auch falsch.
Also wird auch hier ein weiterer Funktionsbaustein für das Erdgeschoss benötigt.
Die Funktionsweise des Erdgeschossbausteins ist ähnlich wie beim A-Bau-Kellergeschoss. Es wird zuerst zweimal der Weg zum Treppenhaus gezeigt, mit der Beleuchtung des Erdgeschosses und anschließend schaltet die Stockwerksanzeige zum ausgewählten Raum um. Für diese Umschaltung wird auch eine weitere Variable benötigt.

Die Eingangsvariablen:

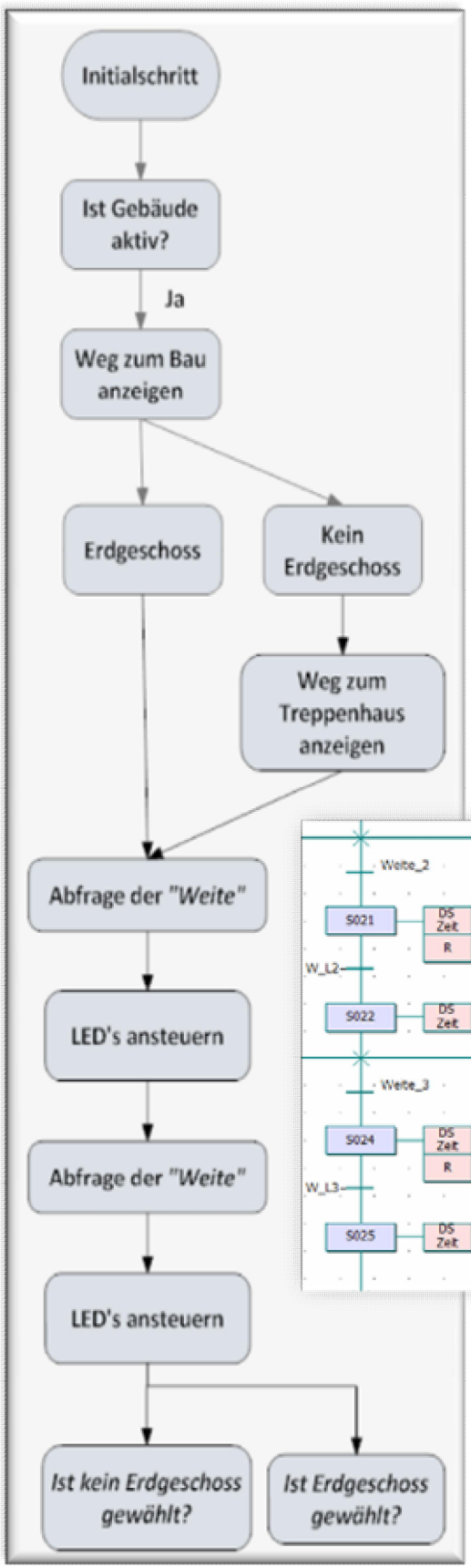


- Die Bau Variable aus dem Verarbeitungs-Baustein Datentyp Integer. Sie dient zur Aktivierung des Bausteins.
- Die Variable Zeit im Datentyp Time. Durch sie wird die Geschwindigkeit eingestellt, wie schnell die LED's nacheinander aufleuchten.
- Die Weite des dazugehörigen Bausteins im Datentyp Integer.
- Die Stockwerk Variable aus dem Verarbeitungsbaustein im Datentyp Integer. Sie dient unter anderem zur Aktivierung des Bausteins.
- Als letzte dann die GoBuilding Variable im Datentyp Integer. Diese wird zum zurücksetzten der Schrittkette benötigt. Wird hier der Wert „5" von der Visualisierung gesendet, so springt die Ablaufsteuerung wieder in ihren Initialschritt.

Die Ausgangsvariablen:

- Die Hauptgruppe von Ausgangsvariablen sind die LED's, welche im Datentyp Bool angesteuert werden.
- Des Weiteren sind bei den zwei Sonderbausteinen noch weitere Variablen zur Ansteuerung der Stockwerksfarbe notwendig gewesen (siehe oben).

Kommen wir nun zur eigentlichen Programmierung des Funktionsbausteins. Dieser wurde in Ablaufsprache geschrieben, um das Lauflicht zu generieren. Die grundsätzliche Funktion ist bei allen programmierten Schrittketten-Funktionsbausteinen identisch.



Jede Schrittkette beginnt mit einem „Initialschritt“, womit die Anlage in eine Grundstellung gebracht wird. Hier werden alle in der Kette verwendeten LED’s zurückgesetzt.

Nun folgt eine Transition, wodurch die Kette aktiviert wird. Diese Abfrage erfolgt über einen Vergleich von Bau- und Stockwerksinformationen.

Wird nun die Transition „wahr“, folgt die nächste Aktion, die den Weg vom Startpunkt zum ausgewählten Gebäudeteil anzeigt.
Jetzt folgt die erste alternative Verzweigung. Hierzu wird die Stockwerksinformation benötigt.

Erdgeschoss
Dies bedeutet, dass das Lauflicht vom Startpunkt aus zum gewünschten Raum angezeigt werden soll.

Kein Erdgeschoss
Dieser Weg hat zur Folge, dass der Weg vom Startpunkt zum Treppenhaus dauerhaft angezeigt wird. Das Lauflicht startet dann vom Treppenhaus zum gewählten Raum.

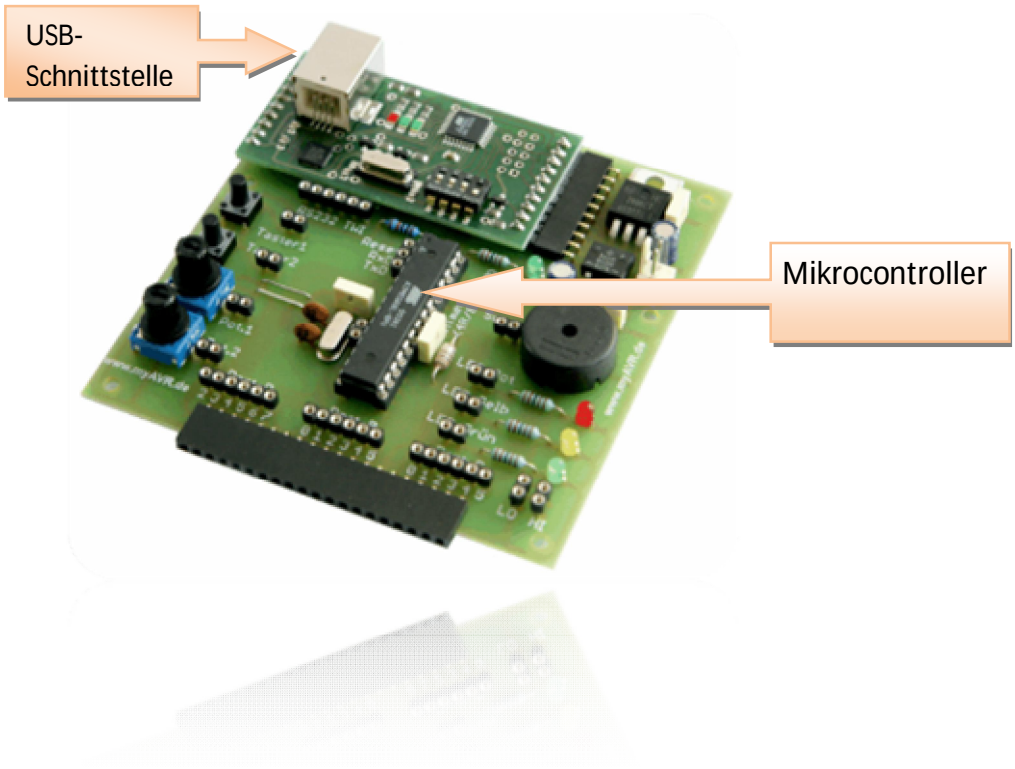
Als nächstes wird die Weite-Information abgefragt. Anhand von dieser Information werden jetzt die LED’s nacheinander angesteuert. In Abhängigkeit von der *„Weite“* wird die Kette weiter durchlaufen oder durch eine Verzweigung erfolgt ein Sprung. Es stehen immer zwei Sprünge zur Wahl, je nachdem ob sich der gewählte Raum im Erdgeschoss befindet oder nicht. Befindet sich der Raum im Erdgeschoss, so geht der Sprung wieder zum Anfang (Initialschritt). Hierdurch startet das Lauflicht wieder am Ausgangspunkt.
Befindet sich der Raum aber nicht im Erdgeschoss, sondern auf einer anderen Ebene des Gebäudes, so erfolgt der Sprung in den Alternativ-Zweig „Kein Erdgeschoss“.

2.4. Der Mikrocontroller

Wir standen vor dem Problem, eine 7-Segment-Anzeige anzusteuern ohne weitere Ausgänge der SPS zu belegen. Da in jedem Modul des Plexiglasmodells eine Platine zur Ansteuerung der Leuchtstofflampen geplant war und hier bereits die 24V Signale (Stockwerksansteuerung) der SPS zur Verfügung standen, überlegten wir uns, diese Signale mit einem Mikrocontroller zu verarbeiten.
Also haben wir auf den Platinen zusätzlich die Signale der Steuerung über einen Spannungsteiler auf 5V gebracht, da dies die Arbeitsspannung des Mikrocontrollers ist.

2.4.1. Mikrocontroller und „myAVR Board 2 USB“

Als Schnittstelle zum Programmieren wählten wir das „myAVR Board 2 USB“, welches sehr komfortabel über eine USB-Schnittstelle an den PC angeschlossen wird.



2.4.2. myAVR Workpad

Die Software myAVR Workpad ist eine Entwicklungsumgebung zum Kompilieren, Brennen und Testen des Controllers

Wir haben uns für die Programmiersprache „C“ entschieden.

Am Anfang des Programms definiert man mit dem Befehl „#define“ die Prozessorgeschwindigkeit. Dann fügt man den Befehlssatz (Bibliothek) des Controllers ein mit Hilfe des Befehls „#include“. Bei uns handelt es sich hier um die Bibliothek „avr\io.h“. Hier werden z. B. den Ein- und Ausgängen die absoluten Adressen zugewiesen.

```
16 //---------------------------
17 #define      F_CPU    3686400
18 #include <avr\io.h>
19 //---------------------------
```

Im nächsten Programmteil werden die Ports (Ein-/Ausgänge) initialisiert.
Zuerst werden die Eingänge mit dem Befehl „cbi (DDR*1,*2)“ festgelegt. Wobei das *1 für die Portnummer und *2 für die Pinnummer steht.
Die Ausgänge werden mit dem Befehl „sbi (DDR*1,*2)“ definiert.

```
20 // Initialisierungen (Unterprogramm)
21 //-------------------------------------------------------
22 void init()
23 {
24     // Ports initialisieren
25     cbi(DDRC,0);     // PORTC0 auf Eingang ohne PullUp    UG
26     cbi(DDRC,1);     // PORTC1 auf Eingang ohne PullUp    EG
27     cbi(DDRC,2);     // PORTC2 auf Eingang ohne PullUp     1
28     cbi(DDRC,3);     // PORTC3 auf Eingang ohne PullUp     2
29     cbi(DDRC,4);     // PORTC4 auf Eingang ohne PullUp     3
30     cbi(DDRC,5);     // PORTC5 auf Eingang ohne PullUp     4
31
32     sbi(DDRB,0);     // PORTD0 auf Ausgang a
33     sbi(DDRB,1);     // PORTC1 auf Ausgang b
34     sbi(DDRB,2);     // PORTC2 auf Ausgang c
35     sbi(DDRB,3);     // PORTC3 auf Ausgang d
36     sbi(DDRB,4);     // PORTC4 auf Ausgang e
37     sbi(DDRB,5);     // PORTC5 auf Ausgang f
38     sbi(DDRD,0);     // PORTD0 auf Ausgang g
39 }
```

Kommen wir jetzt zur Main-Funktion:

```
////////////////////////////////////////////////////////////////////////////
// Main-Funktion
////////////////////////////////////////////////////////////////////////////
main()
{
    init();                         // Aufrufen des Unterprogramms:
                                    // Initialisierungen

    while (true)                    // Mainloop-Begin
    {
        int Stockwerk = 0;          // UG=1,EG=2,1=3,2=4,3=5,4=6
        if (PINC&0b00000001)        // ist UG aktiv
            Stockwerk = 1;
        else if (PINC&0b00000010)   // ist EG aktiv
            Stockwerk = 2;
        else if (PINC&0b00000100)   // ist  1 aktiv
            Stockwerk = 3;
        else if (PINC&0b00001000)   // ist  2 aktiv
            Stockwerk = 4;
        else if (PINC&0b00010000)   // ist  3 aktiv
            Stockwerk = 5;
        else if (PINC&0b00100000)   // ist  4 aktiv
            Stockwerk = 6;
        else                        // nichts aktiv = 0
            Stockwerk = 0;

        switch (Stockwerk)
        {
            case 1 :    PORTB=0b0111110; PORTD=0x00;         // U ansteuern
            break;
            case 2 :    PORTB=0b0111001; PORTD=0b00000001;   // E ansteuern
            break;
            case 3 :    PORTB=0b0000110; PORTD=0x00;         // 1 ansteuern
            break;
            case 4 :    PORTB=0b0011011; PORTD=0b00000001;   // 2 ansteuern
            break;
            case 5 :    PORTB=0b0001111; PORTD=0b00000001;   // 3 ansteuern
            break;
            case 6 :    PORTB=0b0100110; PORTD=0b00000001;   // 4 ansteuern
            break;
            default :   PORTB=0x00; PORTD=0x00;              // kein Segment
            break;                                           // ansteuern
        }

    }
    // Mainloop-Ende
```

Hier wird zuerst das Unterprogramm zur Initialisierung aufgerufen und dann beginnt die Schleife mit dem Befehl „while (true)“.
Die Variable Stockwerk wird als Datentyp Integer deklariert und = 0 gesetzt.
Mit Hilfe des Befehls „if/else“ wird jetzt überprüft welcher Eingang „1“ ist und die Variable mit dem dazugehörigen Wert beschrieben.
In der Klammer nach dem „if“ Befehl steht ein Vergleich. Wenn dieser wahr ist, wird die nächste Zeile ausgeführt. Ist der Vergleich unwahr, springt das Programm zum nächsten „else“ Befehl. Hierbei wird abgefragt, welcher Eingang (Pin C 1-6) aktiv ist. Die Angabe des Pins erfolgt hierbei binär.

Kommen wir zu letzt zur Ansteuerung der Ausgänge:

```
switch (Stockwerk)
{
    case 1 :    PORTB=0b0111110; PORTD=0x00;         // U ansteuern
    break;
    case 2 :    PORTB=0b0111001; PORTD=0b00000001;   // E ansteuern
    break;
    case 3 :    PORTB=0b0000110; PORTD=0x00;         // 1 ansteuern
    break;
    case 4 :    PORTB=0b0011011; PORTD=0b00000001;   // 2 ansteuern
    break;
    case 5 :    PORTB=0b0001111; PORTD=0b00000001;   // 3 ansteuern
    break;
    case 6 :    PORTB=0b0100110; PORTD=0b00000001;   // 4 ansteuern
    break;
    default :   PORTB=0x00; PORTD=0x00;              // kein Segment
    break;                                           // ansteuern
}
```

In diesem Programmteil wird die Variable Stockwerk mit den Konstanten 1 bis 6 verglichen. Wird eine Übereinstimmung festgestellt, so werden die Ausgangsport‘s B und D auf die entsprechenden Werte gesetzt, welcher als Binär oder Hex angegeben ist. Trifft keine Auswahl zu, so wird die Zeile „default“ ausgeführt.

Nach der Programmierung wurde, das Programm mit Hilfe des „myAVR Board’s“ auf die vier Controller gebrannt.

3. Fazit

Es war für das gesamte Projektteam eine große Herausforderung dieses Projekt zu realisieren. Mit so viel positiver Resonanz von der Schulleitung, der Lehrerschaft und den Besuchern hatten wir nicht gerechnet und somit ist der große Einsatz des letzten dreiviertel Jahres belohnt worden.

Natürlich konnten wir in der kurzen Zeit nur einen sehr soliden Grundstein setzten, welcher noch ausbaufähig ist. Es ist vorstellbar, die Stunden- und Vertretungspläne oder einen Informationsbereich über die angebotenen Schulformen in die Visualisierung zu integrieren.
Des Weiteren ist auch eine Anbindung an die Haustechnik denkbar, um relevante Daten abzurufen (Solaranlage, Heizung, Stromverbrauch, usw.).

Durch weitere kleinere LCD-Monitore, welche an „Infopoints“ über die Schule verteilt werden, könnte man ein durchgängiges System erschaffen.

Die Idee des Wegeleitsystems ist natürlich nicht nur für Schulen interessant. So könnte es überall dort zum Einsatz kommen, wo viele Besucher ein fremdes Gebäude betreten (z.B. Krankenhäuser, Altenheime, Messen, usw.).
Das System ist Modular aufgebaut und so funktioniert das Terminal auch ohne das Plexiglasmodell. Also ist es vorstellbar auch nur das Terminal als Wegeleitsystem einzusetzen, oder ein Modell des Gebäudes nach Kundenwünschen anzufertigen.
Hierdurch besitzt das System eine sehr große Flexibilität.

Während der Projektarbeit wurden wir von Herrn Löser, Herrn Musielack und Frau Führich-Albert (Englischlehrerin) in allen relevanten Fragen und Sachverhalten stetig unterstützt. Durch die Zusammenarbeit im Team und mit den Sponsoren konnten wir das Projekt kontinuierlich entwickeln. Die Teilnahme am Wettbewerb Xplore hat wesentlich zur Umsetzung unserer Ideen beigetragen.

Es macht uns sehr stolz, hier ein System entwickelt und aufgebaut zu haben, welches auch in der Praxis zum Einsatz kommt.