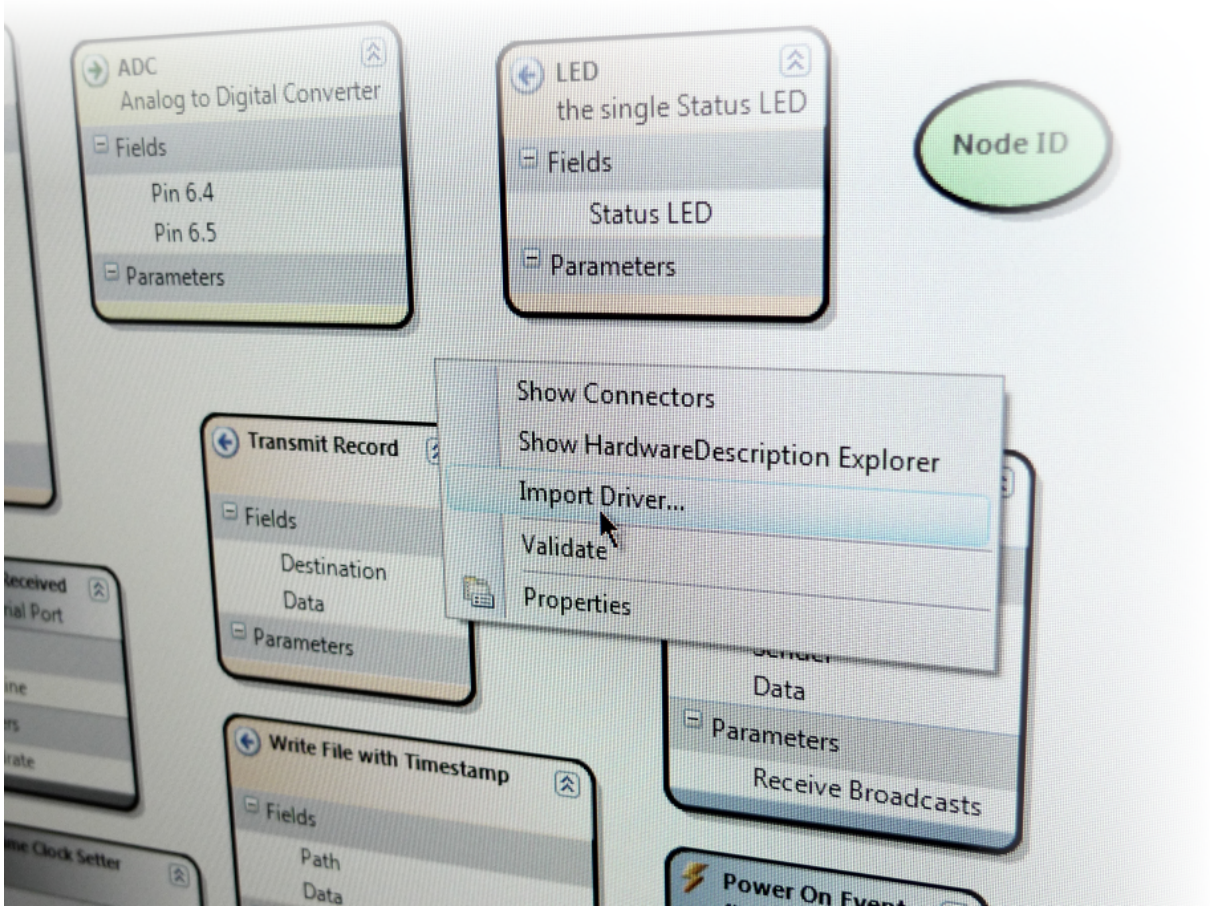
Telematics
Computer Systems

Freie Universität Berlin

Anwenderhandbuch Flow

Flow Version 1.3



Autor: Florian Hinz

5. Mai 2010

Inhaltsverzeichnis

Einleitung

Im Folgenden wird beschrieben, wie *Flow* innerhalb von Visual Studio eingesetzt wird und wie die Benutzeroberfläche effektiv zu bedienen ist. *Flow* ist eine Erweiterung für die integrierte Entwicklungsumgebung Visual Studio und soll die Arbeit bei der Modellierung von Sensorknotenanwendungen für die drahtlosen Sensorknoten der Freien Universität Berlin erleichtern.

Vorbereitung

Zum Austesten der Anwendungsbeispiele auf den nächsten Seiten, wird ein lauffähiges System zur Sensorknotenentwicklung mit installiertem Visual Studio 2008 mit Service Pack 1 und *Flow* benötigt. Wenn keine Sensorknotenhardware zur Verfügung steht oder die Programme erst einmal simuliert werden sollen, muss zusätzlich der Simulator für drahtlose Sensorknoten installiert sein, der in Kapitel 4 vorgestellt wird. Für das Testen der Funkkommunikation mit der Hardware sollten mehrere Sensorknoten vom Typ MSB-430H zur Verfügung stehen.

Noch einmal im Überblick:

- Visual Studio 2008 (Version 9.0) mit Service Pack 1
- *Flow* in Version 1.3
- Sensorknotenhardware
- Erweiterungsmodul MSB-430S
- *Simulator for Flow* in Version 2.2 als Simulationsumgebung
- Treiber für die Unterstützung zusätzlicher Module

Sollte benötigte Software fehlen, dann wenden Sie sich bitte an den Administrator.

In den folgenden Kapiteln werden wichtige Grundlagen vorgestellt und Schritt für Schritt anhand einiger Beispiele dargestellt, wie auf einfache Art und Weise mit *Flow* die ersten Anwendungen für die Sensorknoten entstehen. In Kapitel 5 und 6 werden zwei Beispiele für Anwendungen gezeigt, die autark auf den Sensorknoten laufen. Die Beschreibung der Simulationsfunktion ist in Kapitel 4 zu finden.

Die Sensorknoten

Das Institut für Informatik der Freien Universität Berlin arbeitet seit einigen Jahren an der Entwicklung drahtloser Sensorknoten. In verschiedenen Evolutionsstufen sind diese mit der Zeit kleiner und stromsparender geworden. Sie sollen bei der Beobachtung und Analyse großflächiger Untersuchungsgebiete (z.B. Naturschutzgebiete bei der Vogelbeobachtung) über lange Zeiträume hinweg helfen, ohne dabei die natürlichen Gegebenheiten durch menschlichen Einfluss zu stören. Besonders stromsparende Bauteile und eine intelligente Steuerung erlauben das Sammeln von Daten über Monate hinweg ohne Wechsel der Batterien.

Die Sensorknoten der neuesten Generation sind modular aufgebaut. Der Mikroprozessor ist zusammen mit einer Funkschnittstelle, einem Speicherkartenslot und vielen Signalein- und Ausgängen auf dem Grundmodul *(MSB-430H)* untergebracht.

Es gibt derzeit zwei Erweiterungsmodule: Für interne Tests wurde das Modul *MSB-430S* entwickelt. Auf diesem sind zusätzliche Buttons, drei LEDs und einige Sensoren untergebracht. Für den realen Einsatz wurde das Wytham-Modul entwickelt. Es soll von Forschern der Oxford University verwendet werden, um Vogelpopulationen zu beobachten unter anderem im Hinblick auf den Klimawandel. An dieses Modul kann ein externer RFID-Leser mit separater Stromversorgung (regelbar über Modul) und mehrere Temperatur- und Luftfeuchtigkeitssensoren über einen *1Wire*-Bus angeschlossen werden. Je nach Einsatzzweck können weitere, aufsteckbare Module entwickelt werden, auf denen beliebige Sensoren untergebracht werden können.

Abbildung 2.1 – Sensorknoten der FU-Berlin in verschiedenen Konfigurationen (v.l.: Grundmodul mit MSB-430S-Erweiterung, Wytham-Konfiguration, Grundmodul)

Wichtige Grundlagen

Die Benutzeroberfläche von *Flow* basiert auf Visual Studio 2008 und soll hier anhand der Abbildung 3.1, einem Screenshot eines geöffneten Dataflows erläutert werden:

1. Auf der **Zeichenfläche** in der Mitte werden die Datastructures und Dataflows grafisch dargestellt. Hier werden die Elemente erstellt, aus denen die spätere Anwendung besteht.
2. Der **Solution Explorer** ist rechten Rand zu finden und enthält alle Dateien, die für ein erfolgreiches Entstehen einer neuen Anwendung benötigt werden. Über ein *Kontextmenü*, das durch einen Rechtsklick auf den Projekttitel sichtbar wird, können über den Befehl *Add* neue Datastructures und Dataflows hinzugefügt werden.
3. Aus der **Toolbox** am linken Rand wählt man sich zum Erstellen neuer Elemente den gewünschten Elementtyp aus. Man zieht ihn per *Drag-And-Drop* aus der Toolbox auf die Zeichenfläche. Eine Ausnahme bildet das *Connection*-Werkzeug. Dieses wird mit einem Einfachklick ausgewählt und mit zwei weiteren Klicks werden die Endpunkte der Verbindung festgelegt.
4. Über das **Properties**-Fenster müssen die meisten Elemente für ihren Einsatzzweck angepasst werden. Dazu muss ein Element auf der Zeichenfläche ausgewählt werden. Die Eigenschaften lassen sich aber nur in den Datastructures verändern!
5. In der **Bedienleiste** am oberen Rand findet man schließlich die Funktionen mit denen eine fertige Anwendung auf die Sensorknoten überspielt werden oder der Simulator gestartet werden kann.

Die Komponenten einer Anwendung im Überblick:

- **Hardware Description (`.hardwareinfo`):** Enthält alle Funktionen, die von der Hardware zur Verfügung gestellt werden. Durch Treiber können die Funktionen ergänzt werden. Die Roten Linien stellen Verknüpfungen zwischen Elementen und Funktionen dar, sie spielen für die Bedienung keine Rolle.
- **Datastructure:** Enthält Strukturen (z.B. Variablen, Records, Formeln), die für die Anwendung benötigt werden und hier angepasst werden.
- **Dataflow:** Enthält den Programmablauf, der aus Funktionen und Strukturen zusammengesetzt wird.

Die wichtigsten Elementtypen in den **Datastructures**:

- **Variable:** Ein „Behälter“ zum Speichern eines einzelnen Wertes. Das kann ein Text, eine Zahl oder eine boolescher[1] Wert sein.
- **Record:** Ein Datensatz, in dem verschiedene Datentypen zusammengefasst werden, wie z.B. Ein Messwert mit der dazugehörigen Zeitangabe *(Timestamp)*.
- **Formula:** In diesem Element kann eine Nummer, eine Zeichenkette oder eine einfache mathematische Formel untergebracht werden. Der Datentyp wird je nach Eingabe au-

[1] Ein boolescher Wert *(Boolean)* kann nur die Werte Wahr oder Falsch, bzw. True oder False annehmen.

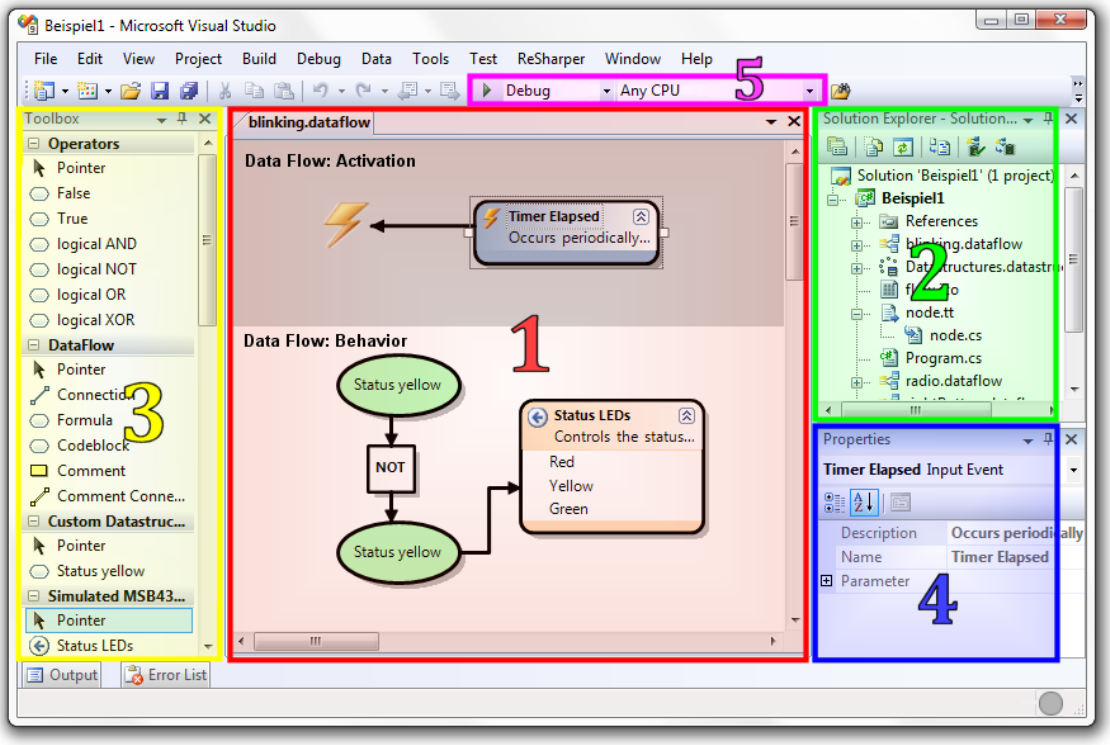
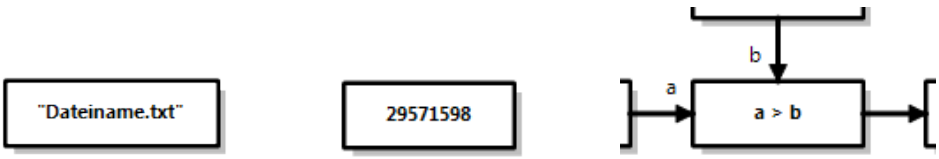


Abbildung 3.1 – Die Benutzeroberfläche von *Flow* ist in mehrere Bereiche unterteilt.

tomatisch erkannt *(siehe Abbildung 3.2)*. Bei einer Formel müssen die zu verrechnenden Eingabewerte einfach durch Verbindungen von anderen Elementen festgelegt werden. Die Verbindungen werden mit Buchstaben versehen und diese können dann innerhalb der Formel verwendet werden. Dieses Element wird aber auch verwendet, um feste Zahlen oder Zeichenketten anzulegen wie z.B. Dateinamen, Verzeichnispfade oder Sendeadressen. Zeichenketten müssen dabei immer von Anführungszeichen umgeben sein.



(a) **(b)** **(c)**

Abbildung 3.2 – In 3.2a wurde das Ergebnis der Formula durch die Anführungszeichen als Zeichenkette erkannt; in 3.2b wurde eine Zahl erkannt; in 3.2c wurde nach Eingabe der Formel mit den numerischen Eingaben a und b der Datentyp *Boolean* festgestellt.

Die wichtigsten Elementkategorien für die **Dataflows**:

- **Input:** Eingelesener Wert, **dargestellt durch einen grünen Pfeil nach rechts**, wie z.B. eine geschlossene Verbindung bei den Pins, die Uhrzeit der Echtzeituhr, die Werte des Analog-Digital-Wandlers, Temperatur oder Luftfeuchtigkeit.
- **Output:** Die Möglichkeit, Informationen nach außen zu bringen, **dargestellt durch einen nach links gerichteten blauen Pfeil**: LEDs, Pins oder der Summer (boolesche Werte), Daten auf die Speicherkarte schreiben, versenden oder über die serielle Schnittstelle ausgeben.
- **Event:** Ereignis, **symbolisiert durch einen orangenen Blitz**, das einen Programmablauf *(Dataflow)* auslösen kann: Ein Knopfdruck, geschlossene Pin-Verbindung, Zeitgeber oder empfangene Daten.

Die Elemente können anhand dieser Kategorien in der Toolbox unterschieden werden *(siehe Abbildung 3.3)*.

(a) **(b)** **(c)**

Abbildung 3.3 – Ein Input 3.3a wird durch einen grünen Pfeil nach rechts dargestellt; Ein Output 3.3b mit einem blauen Pfeil nach links; Ereignisse 3.3c symbolisiert ein orangefarbener Blitz.

Funktionsweise der *Flow* -Grundidee: Datenflüsse

Flow basiert auf der Idee, einen Datenfluss *(Dataflow)* zwischen Elementen grafisch mit gerichteten Verbindungen *(Pfeilen)* darzustellen. Mit jedem Fluss wird ein Wert vom Ursprung entlang des Pfeils zum Ziel kopiert.

⇒ Wichtiger Hinweis zur Bedienung: Soll der Wert eines **ganzen** Elementes kopiert werden, dann klicken Sie mit dem Connection-Werkzeug auf den **Titel** des Elements. Soll hingegen nur ein im Element gespeicherter **Wert** kopiert werden, dann klicken Sie auf den **Namen des Wertes**.

Auf Abbildung 3.4 ist ein Beispiel zu sehen, das im Folgenden erläutert wird:

Das obere Drittel ist grau hinterlegt und stellt den *Activation*-Bereich dar *(Aktivierung)*. Hier werden die Ereignisse festgelegt, durch die der darunter modellierte Dataflow aktiviert werden soll. Im Beispiel ist das Ereignis die Ansteuerung von Pin 1.3. Im *Behavior*-Bereich *(Verhalten)* geschieht dann folgendes: Die grüne Ellipse stellt eine boolesche Variable dar und der Pfeil zum `NOT`-Feld sorgt dafür, dass der Wert der Variable hergenommen und im `NOT`-Feld negiert wird. Dieser negierte Wert wird vom nächsten Pfeil aufgenommen und wieder in die (gleiche) Variable kopiert.

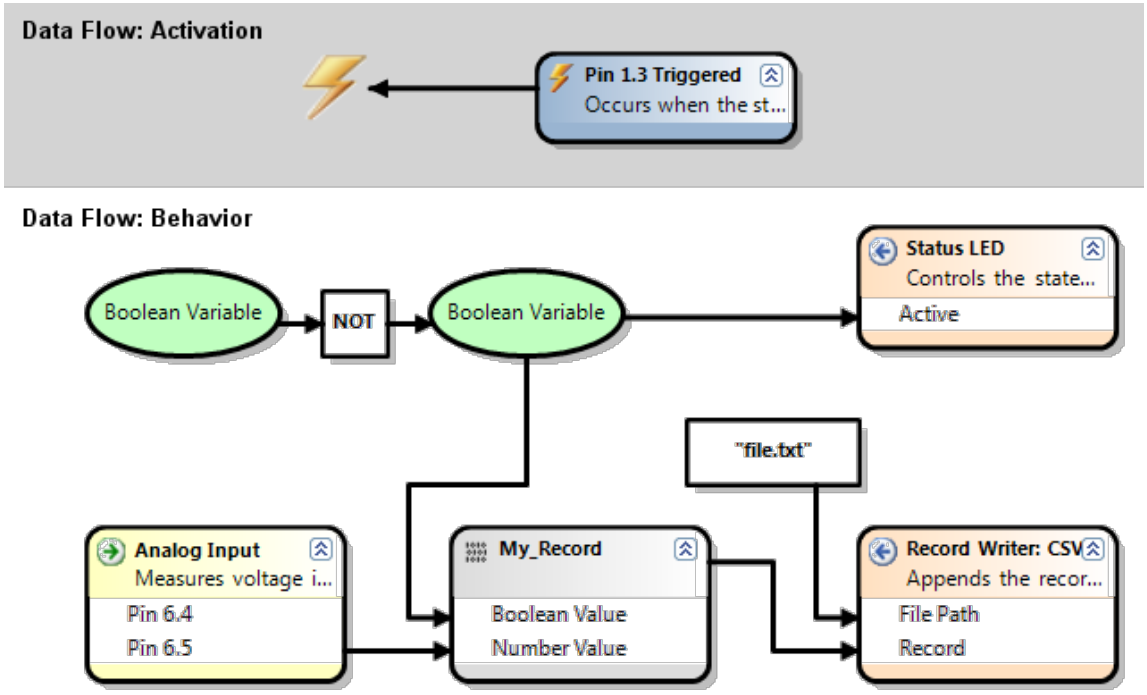


Abbildung 3.4 – Datenflüsse zwischen Elementen werden grafisch durch Pfeile repräsentiert, wobei der Datenfluss entlang der Pfeilrichtung verläuft.

Der Variablenwert wird anschließend über den nächsten Pfeil an den Parameter des Output-Elements *Status LED* weitergereicht. Das sorgt dafür, dass die LED entsprechend des Wertes der Variablen ein- oder ausgeschaltet wird.

Zusammen mit dem Ereignis bedeutet das: Jedes mal, wenn der Pin 1.3 angesteuert wird, wird die LED abwechselnd ein- oder ausgeschaltet!

Im Behavior-Bereich wird gleichzeitig der aktuelle Wert des Analog-Digital-Wandlers *(Input)* zusammen mit dem negierten Variablenwert in einen Record übertragen. Der Record wird dann zum Parameter *Record* des *Output* Record-Writers weitergegeben. Dieser schreibt den Record in eine Datei auf den Flash-Speicher des Sensorknotens. Der Dateiname ist mit einer *Formula* angegeben und wird über einen Pfeil an den *File Path*-Parameter übergeben.

Die Simulationsfunktion

Mit der Simulationsfunktion haben Sie die Möglichkeit, die Funktion der modellierten Anwendung vor dem Übertragen auf die Sensorknoten am eigenen Computer zu testen und eventuelle Probleme noch vor einem Feldtest zu erkennen. Der Simulator steht Ihnen nach Installation der Simulationserweiterung zur Verfügung und wird als eigener Projekttyp behandelt. Wollen Sie eine neue Anwendung modellieren, dann wählen Sie im *New Project*-Dialog den Projekttypen *MSB430H Flow Simulated Platform* ⇒ *Flow Node Simulator* aus.

Die Modellierung neuer Datastructures und Dataflows unterscheidet sich nicht von der Vorgehensweise mit den Projekttypen für die reale Hardware. Der Unterschied besteht nur darin, dass der fertige Code zwar kompiliert aber nicht an die Sensorknoten übertragen werden muss. Stattdessen wird eine Simulation gestartet. Dazu wählen Sie die Konfiguration *Debug* und klicken auf den grünen Pfeil daneben *(siehe Abbildung 4.1)*.

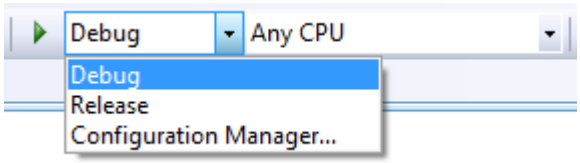


Abbildung 4.1 – Konfigurationsauswahl: Zum Start des Simulators wählen Sie die Konfiguration *Debug* aus.

Als erstes öffnet sich ein zentrales Übersichtsfenster *(siehe Abbildung 4.2)*. In der oberen Liste werden Ihnen alle gefundenen Anwendungen angezeigt. Ist die gesuchte Anwendung nicht zu finden, können Sie über den Button *Add* weitere „`.exe\-` oder „`.dll\-`Dateien nach Anwendungen durchsuchen lassen.

Wenn Sie sich für eine Anwendung entschieden haben, die Sie testen möchten, starten Sie beliebig viele Sensorknotenobjekte der ausgewählten Anwendung über den Button *Create Instance* oder mit einem Doppelklick auf den Anwendungstitel. Die Hardwareausstattung der Sensorknoten muss nicht extra eingestellt werden, die Simulation passt sich den genutzten Funktionen der Anwendung an und zeigt nur die verwendeten Module an.

Im unteren Bereich des Fensters werden alle erstellten Sensorknotenobjekte sowie deren Status *(ready/started/stopped)* angezeigt. Mit den beiden Buttons unten können alle Knoten zugleich gestartet oder gestoppt werden. Der Zustand einzelner Knoten kann per Doppelklick verändert werden.

Neue Sensorknotenobjekte *(siehe Abbildung 4.3)* sind zunächst inaktiv, damit Sie vor dem Start Einstellungen vornehmen können. In der oberen Leiste können Sie das einzelne Objekt steuern: Die *NodeID* lässt sich ändern und die Simulationsfunktion kann mit dem Button „Power On“ / „Power Off“ gestartet oder gestoppt werden. Außerdem können Sie über *Change View* zu einer optisch ansprechenderen Ansicht wechseln, in der Sie jedoch nicht alle Einstellungen manipulieren können *(siehe Abbildung 4.4)*.

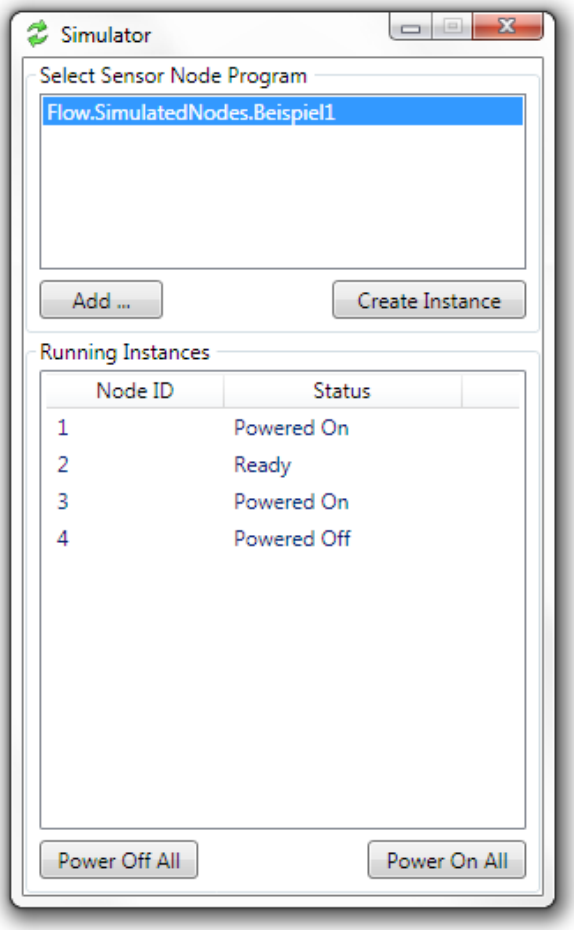


Abbildung 4.2 – Das Übersichtsfenster des Simulators. Oben kann aus den verfügbaren Anwendungen gewählt werden, unten ist der Status aller *Sensorknotenobjekte* ablesbar.

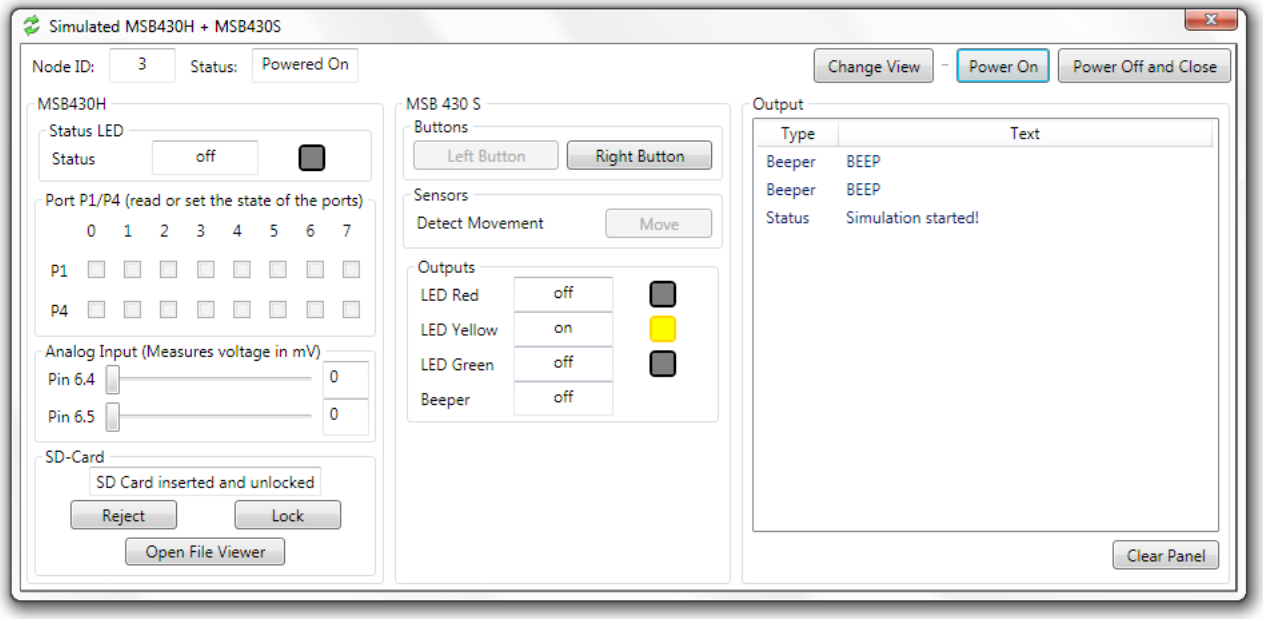


Abbildung 4.3 – Das Simulationsfenster eines Sensorknotenobjektes in der detaillierten Ansicht. Es passt sich den verwendeten Modulen automatisch an.

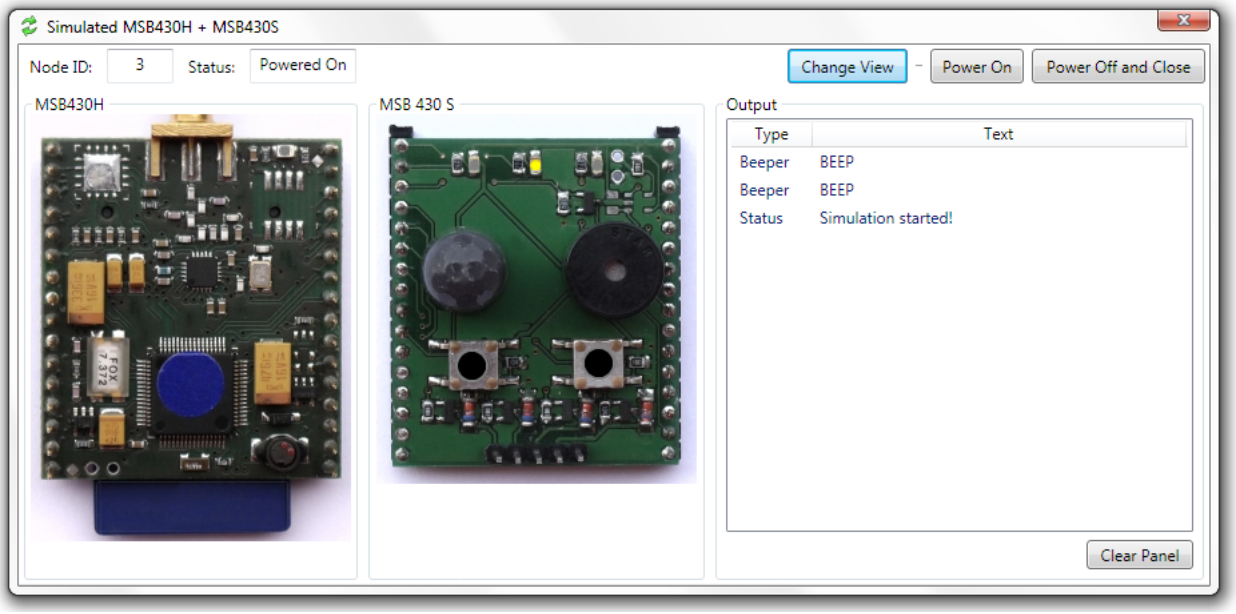


Abbildung 4.4 – Bildliche Darstellung der Simulationsfunktion. Die Buttons können weiter betätigt werden. Die Ansicht ist jederzeit wechselbar.

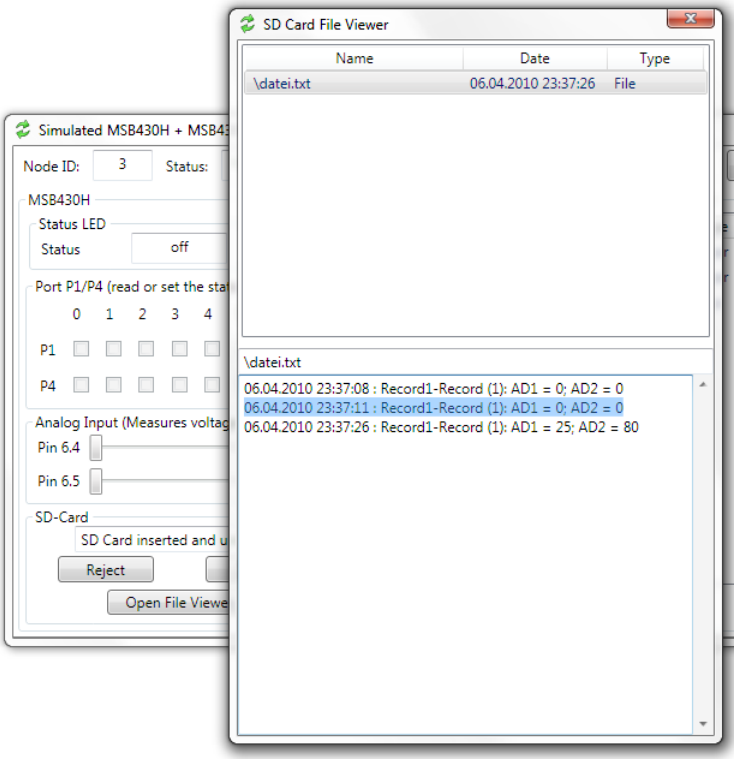


Abbildung 4.5 – Dateianzeige mit Inhalt der SD-Karte. Ein Doppelklick auf eine Datei zeigt den Inhalt an.

Die verwendeten Module werden nebeneinander dargestellt. Alle in der Anwendung benötigten Werte können Sie modifizieren und an die späteren Bedingungen im Feldversuch anpassen. Am rechten Rand befindet sich das Outputfenster für Textausgaben. Hier werden die Ausgaben der seriellen Schnittstelle oder andere Statusmeldungen angezeigt. Die aktuellste Ausgabe erscheint immer am oberen Ende. Die Anzeige können Sie über den Button *Clear Panel* zurücksetzen.

Den Flash-Speicher können Sie über die Buttons des Basismoduls MSB-430H einsehen. Bei einem Klick auf *Open File Viewer* öffnet sich die Dateianzeige im Fenster „SD Card File Viewer“ *(siehe Abbildung 4.5)*. In der grafischen Ansicht öffnen Sie sie durch einen Klick auf die blaue Speicherkarte beim MSB430H. Die angelegten Dateien sind im oben Bereich mit Pfadangabe abgebildet. Zum Anzeigen des Dateiinhalts klicken Sie doppelt auf den Dateinamen. Den Inhalt finden Sie dann im unteren Bereich. Die Texte können Sie dort markieren und in die Zwischenablage kopieren. Über das Kreuz oben rechts schließen Sie die Dateianzeige wieder. Mit dem Button „Reject“/“Insert“ kann die Speicherkarte entfernt oder eingesetzt werden und mit „Lock“ kann der Schreibschutz aktiviert werden.

Das Schließen eines Simulationsfensters bewirkt das Beenden der Simulation für das entsprechende Objekt. *Achtung*: Die angelegten Dateien sind nicht persistent, sie werden mit Beenden der Simulation gelöscht. Wenn Sie das Übersichtsfenster schließen, werden sämtliche Sensorknotenobjekte ebenfalls angehalten und geschlossen.

Die erste Anwendung

In diesem Kapitel wird der Anwender Schritt für Schritt in den Arbeitsablauf mit der *Flow* -Entwicklungsumgebung eingeführt. Für die erste Anwendung wird nur mit den Tastern und den LEDs zur Signalisierung gearbeitet. Dazu wird der Treiber für das Erweiterungsmodul MSB-430S benötigt.

Die Aufgabe: Zum Signalisieren, dass der Sensorknoten aktiv ist, soll die gelbe LED ständig blinken, d.h. jede Sekunde umschalten. Wird der rechte Taster gedrückt während die gelbe LED leuchtet, soll der Summer *(Beeper)* kurz aktiviert werden.

Schritt 1: Sobald die MSB-430H-Plattform und der Simulator auf dem Entwicklungsrechner installiert sind, stehen in Visual Studio zwei verschiedene Projekttypen bereit. Sie werden über den *New Project*-Dialog erzeugt *(File → New → Project)*. Wenn direkt für die Sensorknotenhardware entwickelt wird, lautet der Projekttyp „MSB-430H Flow Hardware Platform“. Für den Simulator wird der Projekttyp „MSB-430H Flow Simulated Platform“ gewählt.

⇒ Legen Sie nun ein neues Projekt des passenden Typs an. Unter *Name* geben Sie den Namen „Beispiel1“ für diese zu modellierende Anwendung ein. Das neue Projekt wird innerhalb einer sogenannten *Solution/Lösung* angelegt. Dieser können im Verlauf noch weitere Projekte hinzugefügt werden. Im Eingabefeld *Solution Name* können Sie der Solution einen weiteren, übergeordneten Titel geben *(siehe Abbildung 5.1)*. Den Titel bitte später nicht mehr ändern, sonst kommt es zu Problemen!

Schritt 2: Ein neu initialisiertes Projekt enthält bereits alle Funktionen des Basismoduls MSB-430H. Diese werden in der *Hardware Description*[1] als Modell grafisch dargestellt, zu finden im *Solution Explorer (siehe Abbildung 5.2)*.

⇒ Mit einem Klick auf *Ok* initialisieren Sie ein neues Projekt und lassen die Entwicklungsumgebung vorkonfigurieren.

Schritt 3: Für diese Beispielanwendung muss das Erweiterungsmodul MSB-430S verwendet werden, damit die Taster und die LEDs zur Verfügung stehen. Um dessen Funktionen in *Flow* verfügbar zu machen, muss der entsprechende Treiber geladen werden.

⇒ Zum Import von Treibern öffnen Sie zuerst die *Hardware Description (Die Datei mit der Endung `.hardwareinfo`)*. Dann öffnen Sie das Kontextmenü durch einen Rechtsklick auf den Hintergrund der Zeichenfläche und wählen *Import Driver (siehe Abbildung 5.3)*. Nach kurzer Zeit öffnet sich ein Auswahldialog. Wählen Sie den passenden Treiber für Ihren Projekttypen (Hardware/Simulation) aus und klicken Sie auf *Ok*. Speichern und schließen Sie die Hardware Description anschließend.

[1] Zu finden in der `.hardwareinfo`-Datei

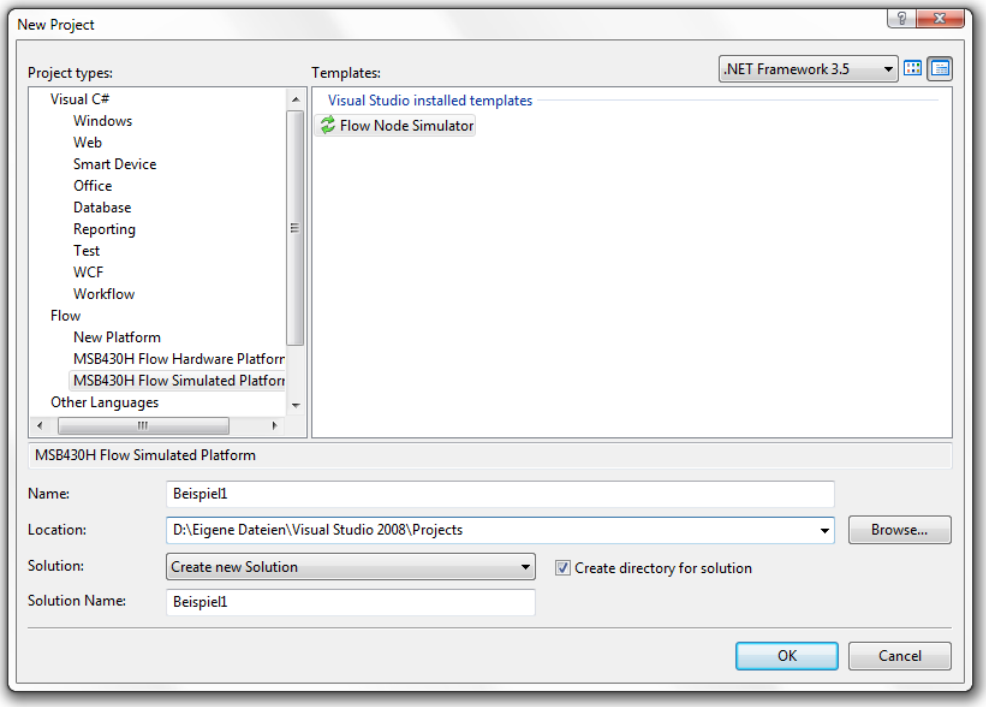


Abbildung 5.1 – Erstellen eines neuen *Flow* -Projektes, hier mit dem Namen „Beispiel1“.

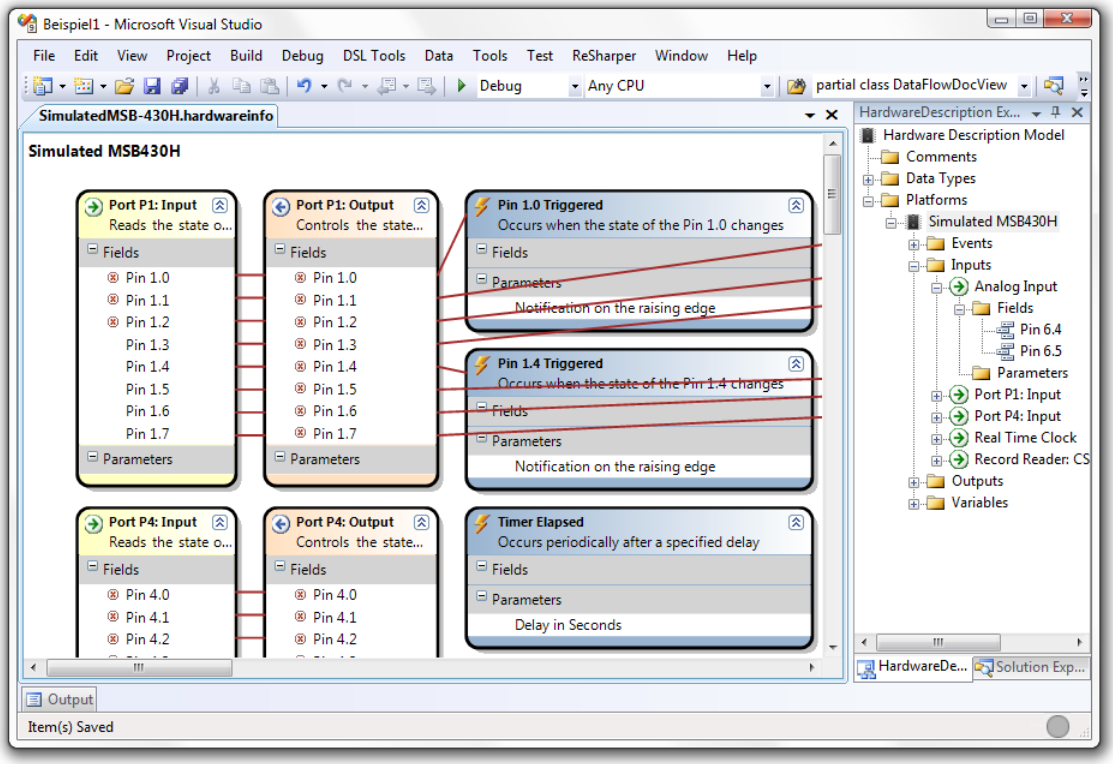


Abbildung 5.2 – Ein neues Projekt mit geöffneter Hardware Description.

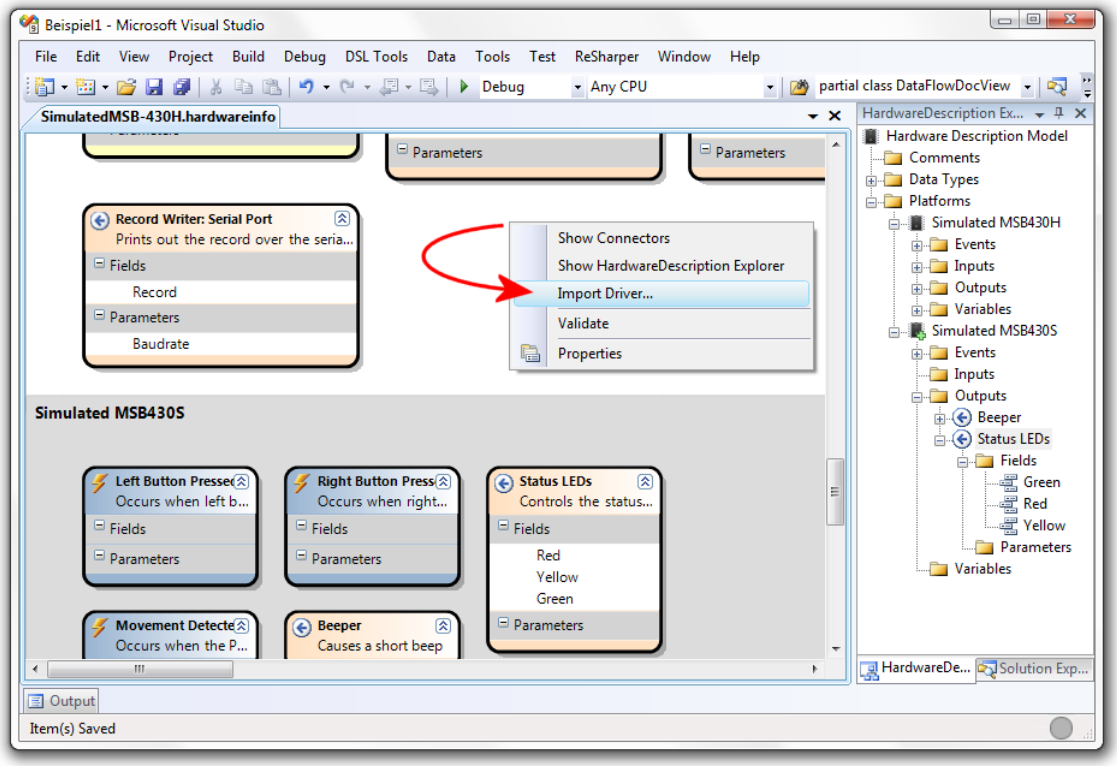


Abbildung 5.3 – Hardware Description mit geladenem Treiber für das Modul MSB-430S und ausgeblendeten Verbindungen. Rechts ist der Hardwaredescription Explorer eingeblendet, der ebenfalls über das Kontextmenü geöffnet werden kann.

Die Hardware Description wird dadurch um Funktionen des Treibers erweitert. Dabei werden auch neue Verbindungen *(rote Linien)* angelegt, die das Modell unübersichtlich machen könnten. Diese Verbindungen können über das Kontextmenü mit dem Befehl *Show Connectors* ein- und ausgeblendet werden.

Schritt 4: Damit ein Wechsel des LED-Zustands stattfinden kann, muss der aktuelle Wert für die LED in einer Variablen gespeichert werden. Variablen müssen vor der Verwendung in einem Datastructure-Modell angelegt werden *(siehe Abbildung 5.4)*.

⇒ Erstellen Sie zuerst im Projekt ein neues *Datastructure*-Modell: Öffnen Sie das Kontextmenü des Projektelements im *Solution Explorer* und klicken Sie auf *Add* → *Datastructures*. Geben Sie einen Namen an und klicken Sie auf *Ok*. In diesem Modell können alle Datenstrukturen erstellt werden, die für das Projekt benötigt werden.

! Sollten Sie noch die Hardware Description geöffnet haben, wird Ihnen ein Fehler beim Öffnen des erstellten Datastructure-Modells angezeigt[2]. Schließen Sie in dem Fall die Hardware Description und öffnen Sie das neu erstellte Datastructure-Modell anschließend von Hand im Solution Explorer.

[2] Aus technischen Gründen können Datastructure-Modelle, Dataflows und die Hardware Description nicht gleichzeitig angezeigt werden. Inhaltliche Abhängigkeiten könnten bei Änderungen zu Inkonsistenzen führen.

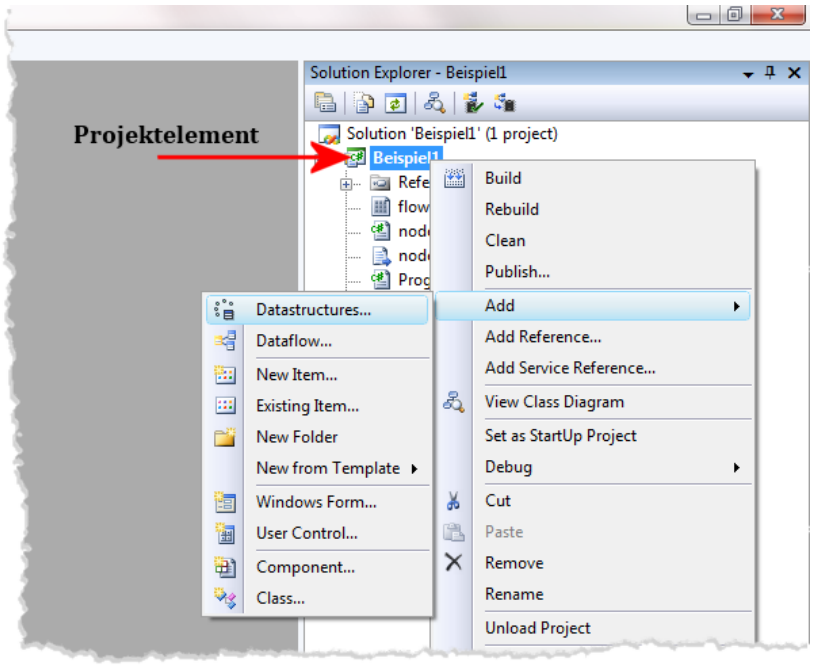


Abbildung 5.4 – Hinzufügen einer neuen Datenstruktur über das Kontextmenü des Projektelements.

Schritt 5: Der Elementtyp *Variable* befindet sich in der Toolbox auf der linken Seite. Es muss eine Variable angelegt und anschließend konfiguriert werden *(ganz wichtig!) (siehe Abbildung 5.5)*.

⇒ Im Datastructure-Modell ziehen Sie eine Variable aus der *Toolbox* per *Drag-and-Drop* auf die Zeichenfläche. Wenn Sie ein Element auf der Zeichenfläche in der Mitte auswählen, können Sie dessen Eigenschaften im Properties-Fenster konfigurieren. Geben Sie der Variablen den Namen „Status yellow“ und wählen Sie den Datentyp *Boolean* aus. Speichern Sie das Datastructure-Modell bitte nach jeder Änderung!

Schritt 6: Die Hardware und die Datenstrukturen sind nun vorbereitet und es kann mit der eigentlichen Modellierung des Datenflusses begonnen werden, welcher das spätere Verhalten der Hardware/des Simulators festlegt. Für jeden Teilaspekt des späteren Verhaltens mit eigener Aktivierungsbedingung wird ein eigener Datenfluss *(Dataflow)* modelliert. Für dieses Beispiel werden insgesamt zwei Dataflows benötigt: Der erste soll dafür sorgen, dass die gelbe LED ständig blinkt.

⇒ Rufen Sie das Kontextmenü des Projekts auf, genau wie beim Erstellen der Datenstrukturen, wählen dann aber *Add* → *Dataflow* aus. Geben Sie dem Dataflow einen passenden Namen wie zum Beispiel „blinking“ und erstellen Sie ihn mit einem Klick auf *Ok*.

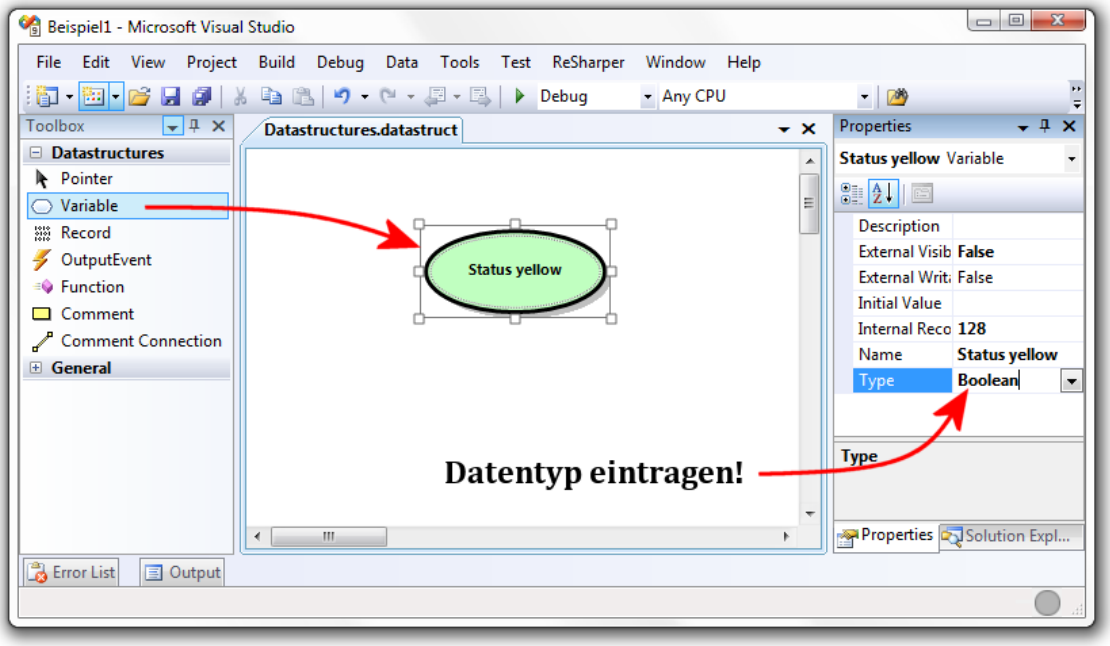


Abbildung 5.5 – Per *Drag-and-Drop* oder Doppelklick wird eine Variable von der *Toolbox* auf die Zeichenfläche gezogen und dann über das *Properties*-Window konfiguriert.

Schritt 7: Der Datenfluss wird jetzt aus Elementen und Pfeilverbindungen modelliert.

⇒ Ziehen Sie die nötigen Elemente wie auf Abbildung 5.6 zu sehen aus der *Toolbox* auf die Zeichenfläche und verbinden Sie sie. Sie benötigen

- das Event „Timer elapsed“ (im Activation-Bereich)
- zwei Mal die eben angelegte Variable „Status yellow“
- ein „logical NOT“
- den Output „Status LEDs“ des Moduls MSB430S

Zum Verbinden wählen Sie das *Connection*-Werkzeug in der Toolbox aus und klicken dann nacheinander auf die beiden zu verbindenden (Unter-)Elemente in *Reihenfolge des Datenflusses*, z.B. von der Variablen zur LED, wenn die Variable die LED steuern soll. Sollte etwas grundsätzlich unklar sein, lesen Sie noch einmal das Kapitel 3 ab Seite 4.

Erklärung der Abbildung 5.6: Das *Periodic Timer Activated*-Event im Activation-Bereich[3] gibt immer beim Auslösen nach der eingestellten Zeit einen „Impuls“ an den Master-Trigger weiter, dargestellt durch die Verbindung zum Blitz-Symbol, woraufhin der modellierte Datenfluss im Behavior-Bereich darunter ausgeführt wird.

Es können auch mehrere Events im Activation-Bereich angelegt und mit dem Master-Trigger

[3] Der Activation-Bereich befindet sich im oberen Teil des Zeichenblatts und ist grau hinterlegt. Der abgebildete Blitz stellt den Master-Trigger dar, der durch den Impuls eines in diesem Bereich abgelegten Events ausgelöst wird.

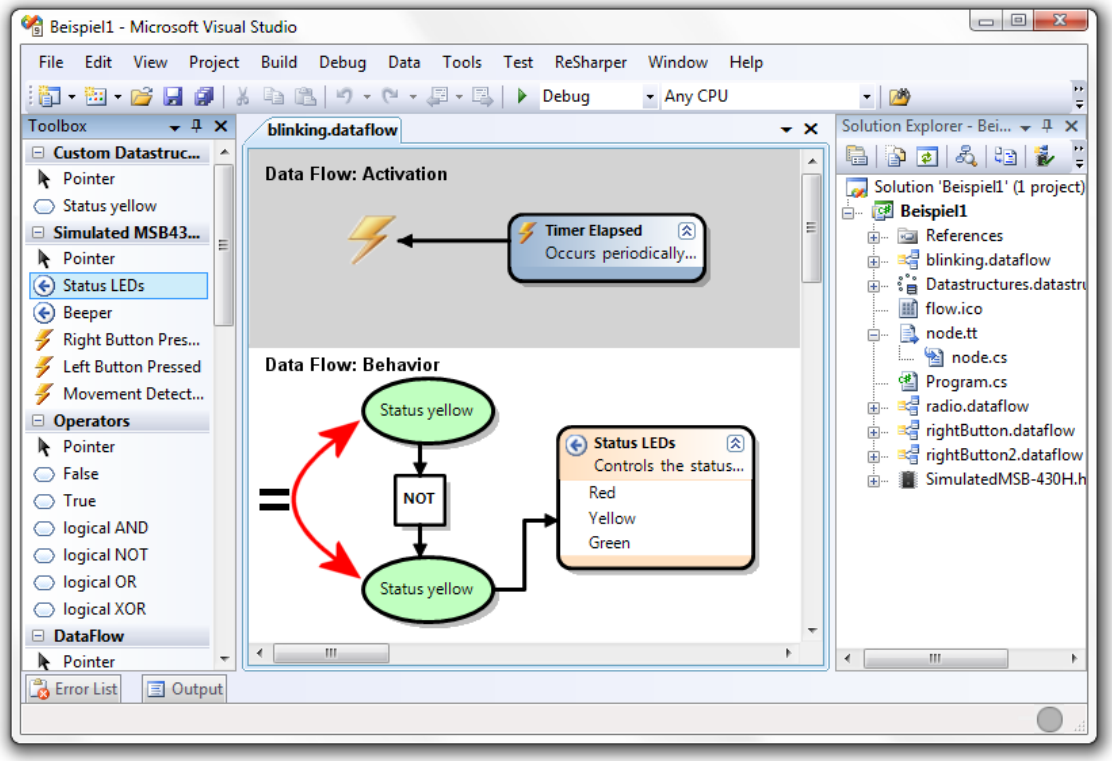


Abbildung 5.6 – Erzeugte Elemente, verbunden mit dem *Connection*-Tool.

verbunden werden. Das wirkt dann wie eine logische *Oder*-Verknüpfung. Das hier verwendete Event soll regelmäßig auslösen *(Timer Elapsed)*. Über die Eigenschaft *Delay in Seconds* im Properties-Fenster kann die Zeitspanne eingestellt werden, in diesem Fall die `1`. Es können nur ganze Sekunden angegeben werden.

Im Behavior-Bereich wird zuerst die Variable „Status yellow“ eingelesen, dann über einen logischen Operator negiert und wieder in die Variable zurückgeschrieben. Schließlich wird der neue Wert der Variablen an die gelbe LED des *MSB430S LEDs*-Output-Ports weitergegeben. Diese Operation wird jede Sekunde durchgeführt, wodurch die gelbe LED im Ein-Sekunden-Rhythmus blinkt.

Schritt 8: Der zweite Dataflow soll nun den Beeper kurz einschalten, sobald der rechte Taster betätigt wird und *gleichzeitig* die gelbe LED eingeschaltet ist, also die Variable „Status yellow“ auf TRUE gesetzt ist. Der Beeper summt übrigens nur einmal kurz bei Zuweisung eines TRUE-Wertes und wird anschließend automatisch wieder ausgeschaltet.

⇒ Legen Sie den Dataflow an und nennen Sie ihn beispielsweise „rightButton“ *(siehe Abbildung 5.7)*. Im Activation-Bereich gibt es ein Event („Right Button Pressed“) und eine zusätzliche Bedingung (Variable „status yellow“ gesetzt auf TRUE), die Sie über eine logische `AND`-Verknüpfung mit dem Master-Trigger verbinden müssen. Im Behavior-Bereich des Dataflows müssen Sie dann nur noch den Beeper aktivieren.

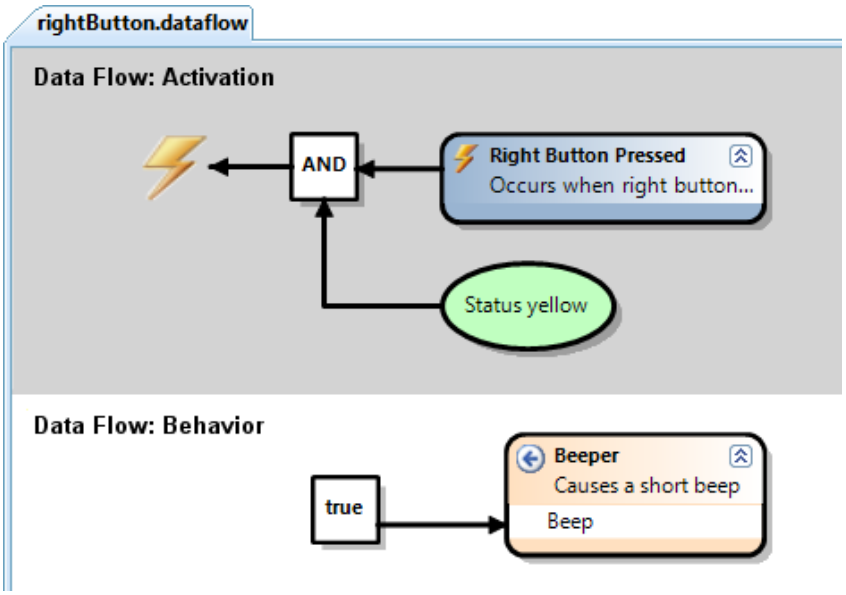


Abbildung 5.7 – Zwei logisch verknüpfte Bedingungen im Activation-Bereich und ein einfaches Aktivieren des Beepers im Behavior-Bereich.

Schritt 9: Die Anwendung ist nun vollständig modelliert. Um ungültige Verbindungen in den Modellen auszuschließen, muss als erstes geprüft werden, ob die Modellierung fehlerfrei ist.

⇒ Prüfen Sie ihre Modellierung, indem Sie im *Solution Explorer* auf den zweiten Button von rechts klicken *(siehe Abbildung 5.8)*. Die *Flow* -Buttons im Solution Explorer sind nur sichtbar, wenn ein *Flow* -Projekt ausgewählt ist. Gibt es Probleme, werden Ihnen diese in der *Error List* von Visual Studio angezeigt.

Schritt 10: Nach fehlerfreier Validierung muss der Quellcode für die Sensorknoten generiert werden. Nach jeder Veränderung der Modelle muss der Quellcode neu generiert werden. Ansonsten würde später nicht der aktuelle Code auf die Sensorknoten programmiert werden.

⇒ Den Codegenerator starten Sie auch im Solution Explorer mit dem Button ganz rechts.

Schritt 11 ***bei Verwendung des Simulators****:* Zum Testen der Anwendung mit dem Simulator wird in der Konfigurationsauswahl in der Symbolleiste von Visual Studio die Konfiguration *Debug* verwendet.

⇒ Klicken Sie auf den grünen Pfeil neben der Konfiguration *Debug* zum Start des Simulators *(siehe Abbildung 5.9)*. Wie Sie mit dem Simulator umgehen, können Sie auf Seite 8 nachlesen.

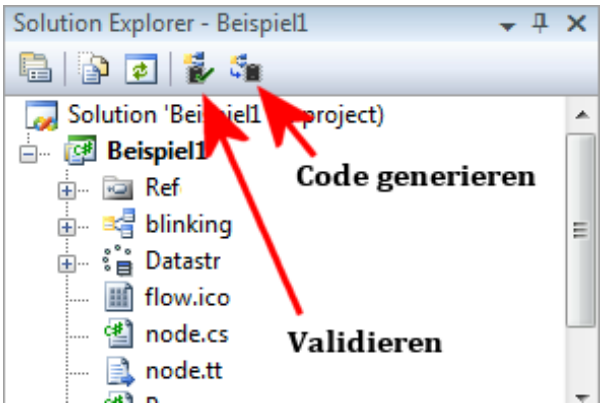


Abbildung 5.8 – Der linke Button im *Solution Explorer* überprüft die Gültigkeit der Modellierung und der rechte Button generiert den Quellcode für die Sensorknoten.

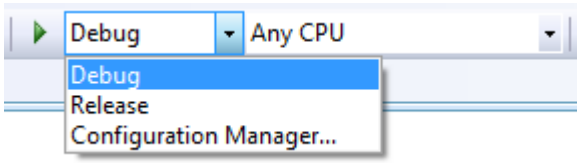


Abbildung 5.9 – Konfigurationsauswahl: Zum Start des Simulators wählen Sie die Konfiguration *Debug* aus.

Schritt 11 ***bei Verwendung der Sensorknotenhardware:*** Wenn die letzten beiden Schritte erfolgreich abgeschlossen wurden, kann das gesamte Projekt kompiliert und auf die Sensorknoten übertragen *(geflasht)* werden. Dazu stehen zwei Konfigurationen bereit, die entweder den Code nur kompilieren *(„Build“)* oder ihn im Anschluss gleich über ein USB-JTAG-Interface auf einen angeschlossenen Sensorknoten flashen *(„Build and Flash“)*.

⇒ Starten Sie den Kompilationsvorgang, indem Sie eine Konfiguration über die Symbolleiste von Visual Studio auswählen und mit dem grünen Pfeil daneben starten *(siehe Abbildung 5.10)*. Während des Kompilierens[4] wird Ihnen die Ausgabe des Compilers im Visual Studio *Output*-Fenster angezeigt. Sollten Fehler auftreten, so sind diese dort und in der *Error List* zu sehen.

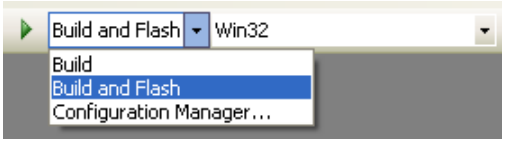


Abbildung 5.10 – *Build*: Nur kompilieren oder *Build and Flash*: Kompilieren und anschließendes Flashen auf einen angeschlossenen Sensorknoten.

[4]Der Kompiliervorgang kann entweder über das Menü oder über die Tastenkombination `Shift+Strg+B` gestartet werden.

Ein komplexeres Beispiel

Die Anwendung aus dem ersten Beispiel wird nun erweitert um die Funkkommunikation des Basismoduls MSB-430H.

Die Aufgabe: Nach jedem Betätigen des rechten Tasters soll per Funk als Broadcast[1] eine Nachricht an die anderen Sensorknoten verschickt werden. Die Nachricht soll den Zustand der gelben LED beinhalten. Empfängt ein anderer Sensorknoten diese Nachricht und die gelbe LED war beim beim Sender aktiv, soll er kurz summen. Folglich summt nicht nur der Knoten, an dem der rechte Button bei aktiver gelber LED betätigt wurde (wie in Beispiel 1 modelliert), sondern auch alle anderen Knoten, die in Empfangsreichweite sind.

Schritt 1: Das Projekt aus Aufgabe 1 kann weiterverwendet werden. In der Datastructure wird ein neuer Record mit nur einem Feld benötigt. Dieser kann dann im Dataflow mit einem booleschen Wert gefüllt und über die Funkverbindung verschickt werden.

⇒ Sie beginnen mit dem geöffneten Projekt mit der Lösung von Aufgabe 1 und legen im bereits vorhandenen Datastructure-Modell einen neuen Record per Drag-and-Drop aus der *Toolbar* an. Dann fügen Sie dem neuen Record über sein Kontextmenü ein Feld hinzu *(siehe Abbildung 6.1)*. Im *Properties*-Fenster des Records müssen Sie den Namen *(„Radio Data Record“)* und bei dem Feld den Datentyp *(„Boolean“)* einstellen.

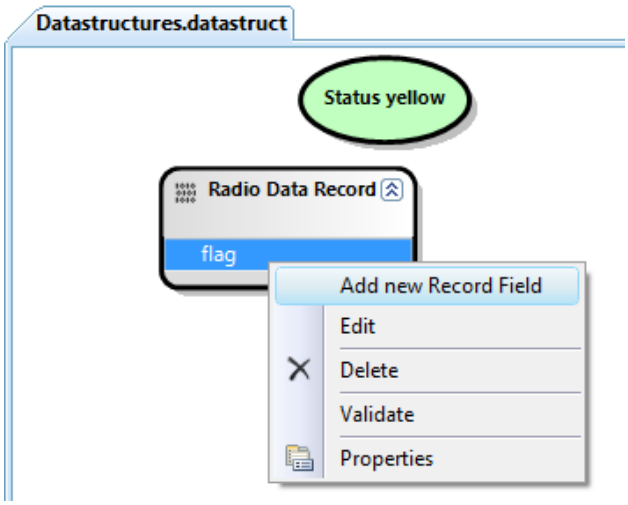


Abbildung 6.1 – Ein Record-Element wurde dem bestehenden Datastructure-Modell hinzugefügt über das Kontextmenü ein Feld mit dem Namen „flag“ erstellt.

[1] Ein Broadcast bezeichnet das Versenden von Daten ohne bestimmten Empfänger, jeder in Empfangsreichweite kann das Paket entgegennehmen. Die Nachricht wird nicht weitergeleitet (geflutet). Als Empfängeradresse für einen Broadcast wird die 0 erwartet.

Schritt 2: Ein neuer Dataflow wird angelegt. Der Dataflow aus dem letzten Beispiel kann nicht weiterverwendet werden, da dieser abhängig vom Status der Variablen *Status yellow* aktiviert wird. Das Record-Paket soll aber bei jedem Tastendruck versendet werden!

⇒ Legen Sie einen neuen Dataflow mit dem Namen „rightButton2“ an. Für die Aktivierung reicht ein einfacher Tastendruck-Event aus. Im Behavior-Bereich verpacken Sie die Variable *Status yellow* in den Record *Radio Data Record* verschicken ihn mittels des *Transmit Record*-Output-Ports an die Broadcastadresse 0. zum Angeben der Adresse verwenden Sie ein Formel-Element, in das Sie einfach nur die Zahl (0) eintragen *(siehe Abbildung 6.2)*.

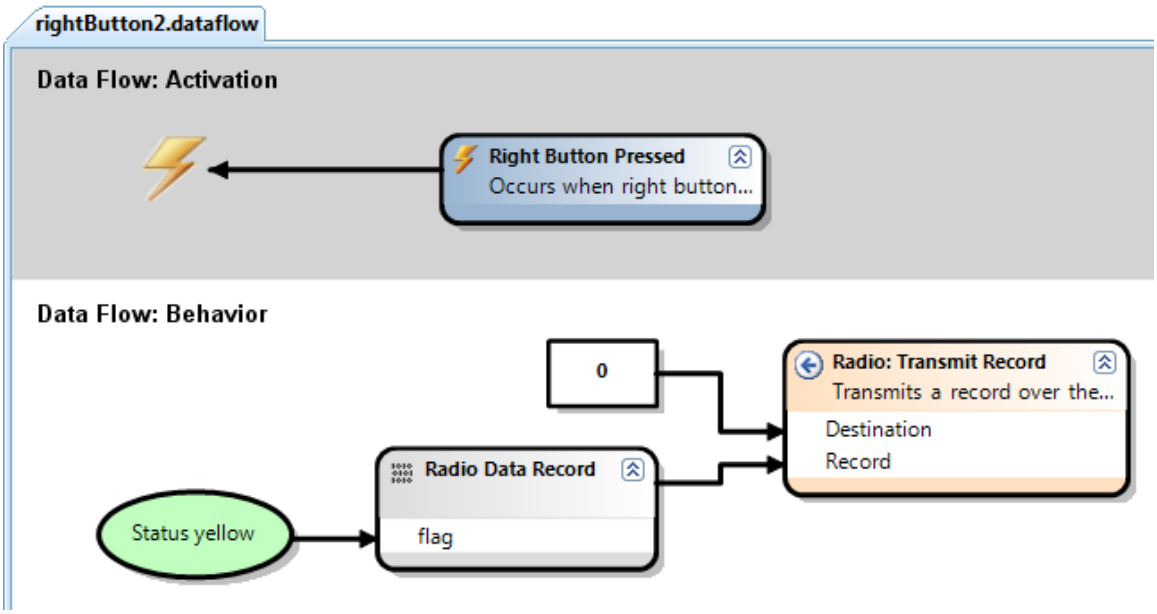


Abbildung 6.2 – Die Variable *Status yellow* wird in den Record *Radio Data Record* verpackt und per Broadcast versendet.

Schritt 3: Die anderen Sensorknoten in Reichweite sollen nun auf das Funkpaket reagieren können, dazu muss ein eigener Dataflow angelegt werden.

⇒ Legen Sie einen weiteren Dataflow mit dem Namen „radio“ an.

⇒ Ob ein Paket über Funk empfangen wird müssen Sie im Activation-Bereich überprüfen, indem Sie das „Radio: Record received“-Event mit dem Master-Trigger verbinden. Soll der Dataflow ausschließlich beim Empfang eines bestimmten Record-Typen aktiviert werden, dann fügen Sie den Record-Typen zum Activation-Bereich hinzu und verbinden Sie ihn mit dem „Record“-Parameter des „Radio: Record received“-Events und dann mit dem Master-Trigger.

⇒ Im Behavior-Bereich nehmen Sie die Daten als einen „Radio Data Record“ entgegen und geben den booleschen Wert darin an den Summer weiter. Ist der Wert TRUE (wie beim Sender des Pakets), wird der Summer kurzzeitig aktiv *(siehe Abbildung 6.3)*.

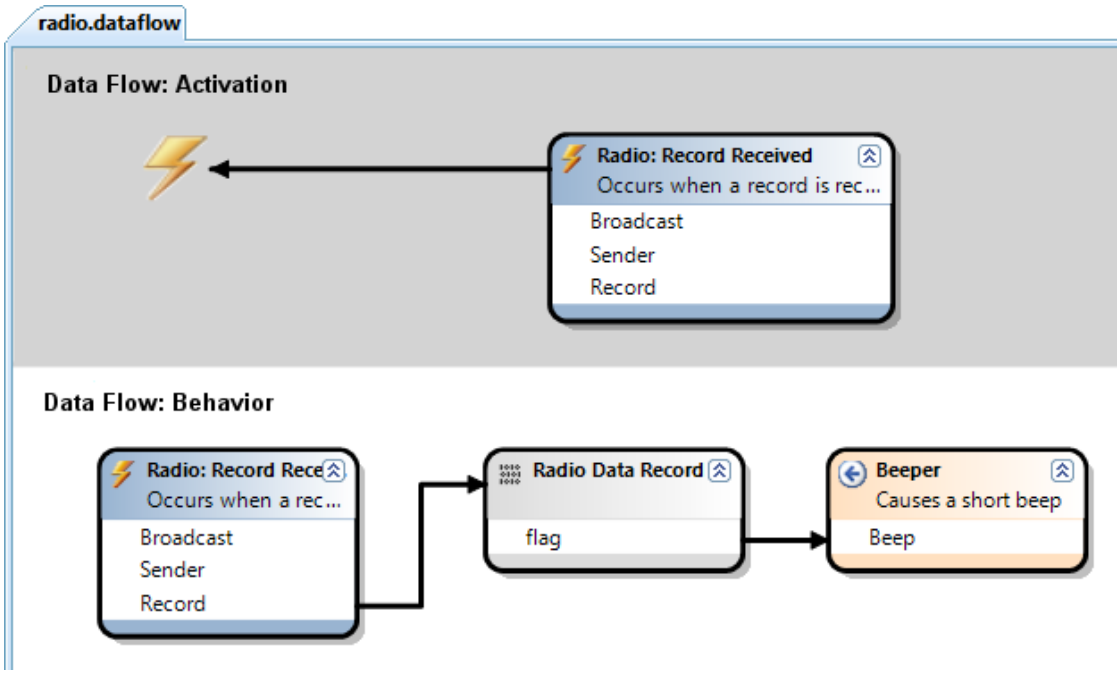


Abbildung 6.3 – Wird ein „Radio Data Record“ empfangen, wird der enthaltene Wert entpackt und an den Summer zur kurzen Aktivierung weitergereicht.

Schritt 4: Diese neue Modellierung muss nun überprüft werden und der Code muss neu generiert werden. Anschließend kann dann das Projekt kompiliert und die Anwendung auf mehrere Knoten verteilt werden. Da alle Sensorknoten die selbe Anwendung verwenden und mit Broadcasts gearbeitet wird, reagieren alle Knoten auf einen Tastendruck eines beliebigen anderen Knotens.

⇒ Gehen Sie für Sie zum Überprüfen, Code generieren und kompilieren des Projekts vor, wie auf Seite 18 beschrieben.

Abbildungsverzeichnis