Systeme hoher Sicherheit und Qualität WiSe 15/16

Christoph Lüth
Martin Ring

4. Übungsblatt

Ausgabe: 01.12.15
Abgabe: 15.12.15

In zu entwickelnden autonome Auto soll ein *safety layer* die Freiheit von Kollision mit stationären Hindernissen sicherstellen. Das auf dieses Gebiet spezialisierte Ingenierbüro *Krach & Bumm* hat für eine geringe sechsstellige Summe folgende Anforderungsspezifikation entwickelt:

(SD-1) Als Distanzmessungen stehen N_{SONIC} nach vorne gerichtete Ultraschallsensoren zur Verfügung, die aber lediglich eine Entfernung und keine Richtung messen, sowie ein Laserscanner, welcher N_{SCDATA} gerichtete Entfernungen misst, und als Messdaten jeweils eine Liste aus Entfernungen und Winkel zurückgibt. Der Laserscanner hat eine Öffnungswinkel von 180°, so dass der i-te Messwert sich auf den Winkel $\alpha_i = (i-1)\frac{180}{N_{SCDATA}-1}$ bezieht (mit $i = 1, \ldots, N_{SCDATA}$).

(SD-2) Das Fahrzeug soll immer so rechtzeitig bremsen, dass zu Hindernissen eine Sicherheitsdistanz von D_{SAFE} eingehalten wird.

(SD-3) Nur bei einer momentanen Geschwindigkeit des Autos v über v_{SLOW} muss die Kollision mit Hindernissen geprüft werden („Langsamfahrt“).

(SD-4) Eine Kollision ist möglich, wenn:

(SD-4-1) mindstens 3 Ultraschallsensoren dreimal hintereinander Hindernisse messen, die näher als die Bremsdistanz sind, oder

(SD-4-2) der Laserscanner misst ein Hindernis in Fahrtrichtung, welches näher als die Bremsdistanz ist.

Die Bremsdistanz d_{brk}ergibt sich aus der momentanen Geschwindigkeit v und der Bremsbeschleunigung a_{brk} als

$$d_{\text{brk}} = \frac{v^2}{2a_{brk}}$$

4.1 *Implementierung abstrakt* — *7 Punkte*

Wir wollen diese Anforderungsspezifikation jetzt implementieren. Der *safety layer* soll im Endeffekt aus einer Funktion bestehen, die zyklisch mit einer Zykluszeit ΔT aufgerufen wird. Die Funktion liest die Eingabedaten (Messungen der Ultraschallsensoren und des Laserscanners, momentane Geschwindigkeit v und Fahrtrichtung ω des Fahrzeugs), und berechnet als Ausgabe, ob eine Notbremsung eingeleitet werden muss (*STOP*) oder nicht.

Als ersten Schritt der Implementierung modellieren Sie den *safety layer* in SysML als Zustands- und Aktivitätsdiagramm, ggf. mit OCL constraints; hierzu gehört auch eine Liste der Systemparameter (als BDD).

4.2 *Implementierung getestet* — *5 Punkte*

Die zu erstellenden Implementierung muss getestet werden. Entwickeln Sie deshalb aus der Spezifikation und der Modellierung in 4.1 Testfälle, bestehend aus Eingabedaten und Ausgabe.

4.3 *Implementierung konkret* *8 Punkte*

Geben Sie eine Implementierung des *safety layers* in der populären Programmiersprache C an. Der *safety layer* besteht dabei aus einer Funktion safety, welche wie oben beschrieben zyklisch aufgerufen werden soll. Die Sensoren kommunizieren mit der Steuerungseinheit über Register, die direkt auf bestimme Adressen im Speicher abgebildet werden. Die Funktion liest die Eingabedaten aus globalen Variablen, welche an dieser Stelle im Speicher stehen; die Variablen sind in C als **volatile** qualifiziert.

```
volatile int supersonic[NUM_SUPERSONIC];
volatile int lasersc[SCDATA_SIZE][2];
volatile int v;
volatile int w;

typedef enum { SAFETY_OK, SAFETY_BRK } safety_status;

safety_status safety()
{
  if (v< V_SLOW) ...
}
```

Der Einfachheit halber sind alle Daten ganze Zahlen in den Einheiten Millimeter und Sekunde (also Geschwindigkeit in mm/s etc.) beziehungsweise (für die Fahrtrichtung) in hunderstel Grad (wobei 0 Grad Geradeausfahrt bedeuten); in der Realität würde jeder Sensor natürlich Daten in einer anderen Einheit liefern, die erst wieder fehlerträchtig konvertiert werden müssten.

Implementierung Sie jetzt die Testfälle aus 4.2. Erfüllen Ihre Testfälle die Anforderungen von *modified condition/decision coverage*? Wenn nein, erweitern Sie diese dahingehend.

Testen Sie zum Schluss ihre Implementierung mit den Testfällen. Sie können dazu das Testrahmenwerk *CUnit*[1] nutzen, oder selbst ein entsprechendes Programm schreiben, welches die Funktion safety aufruft. Zeigen Sie, dass alle Tests erfolgreich durchlaufen werden.

Änderungen:

- *08.12.2015 15:15 Formel für α_i korrigiert wie in der Übung besprochen.*

[1] `http://cunit.sourceforge.net`