



Übungsblatt 9

Abgabetermin 11.07.2006

Aufgabe 1: Virtual Time CSMA

- Ist VT-CSMA soft- oder hard-real-time fähig und warum?
- Warum wird eine virtuelle Zeit eingeführt und wie läuft sie im Bezug zur realen Zeit?
- Definieren Sie Deadline, Ankunftszeit, Sendezeit, Übertragungsdauer und stellen Sie deren Zusammenhang auf einer Zeitachse grafisch dar.
- Ist die reale/virtuale Zeit vor, nach oder genau bei der Sendezeit, wenn der Sendeversuch startet?
- Ist die reale/virtuale Zeit vor, nach oder genau bei der Sendezeit, wenn das Sendeversuchsende erreicht ist?
- Wie und wann wird die virtuelle Zeit mit der realen Zeit synchronisiert?
- Wie sieht das Sendeprotokoll für die in der Tabelle 1 dargestellten Nachrichten aus? Füllen Sie die fehlenden Zellen aus und stellen Sie das Sendeprotokoll grafisch dar. Interpretieren Sie das Ergebnis.

Sender	Nachrichten	T_M	RC	D_M	L_M
1	M_1	10	0	15	
2	M_2	10	10	30	
3	M_3	20	20	55	
4	M_4	15	20	60	

Tabelle 1: Nachrichtenspezifikation

Aufgabe 2: CAN praktisch

Die praktischen Aufgaben werden im Anschluss an die Besprechung der theoretischen Aufgaben von den Studenten am Rechner vorgestellt und durch Sebastian oder Michael abgenommen.

- a) Ziel der abschliessenden praktischen Aufgabe ist die Implementierung eines Regelalgorithmus (P-Regler) für einen Motor.
- Der Motor ist über einen Microcontroller (Motortreiber) mit dem CAN-Bus verbunden.
 - Der Motor dreht pro Minute maximal 4800 mal.
 - Der Odometriesensor hat eine Auflösung von 500 Ticks je Umdrehung.
 - Die Anzahl der zurückgelegten Ticks werden 20 mal die Sekunde über den CAN-Bus publiziert.
 - Die auf dem CAN-Bus laufenden sowie zu sendenden Nachrichten mit den Reglerwerten sind in der Tabelle 2 spezifiziert.
 - Das Programm, welches den Motor auf dem Linux-PC simuliert, ist zum Testen eures Reglers unter `~mschulze/SimulaM` zu finden. SimulaM verwendet bereits automatisch das Device `/dev/pcan24` sowie eine CAN-Übertragungsgeschwindigkeit von 250 kbps.

Sender	CAN-ID	Länge	Inhalt	Wertebereich	Periode
Motortreiber	0x01	2	Odometrieticks	0-65535	20 s ⁻¹
Regler	0x02	2	PWM{H L}	0-400	20 s ⁻¹

Tabelle 2: Nachrichtenspezifikation für die Motorregelung

Hinweis

Im Labor besteht ein CAN-Knotennetz zwischen den Rechnern (eoslab-0{1,3,4,6}) auf der rechten Seite. Zum Testen des eigenen Monitorprogrammes sowie zum Testen des eigenen Sendeprogramms stehen die Programme `receivetest` und `transmittest` unter `/usr/local/bin/` zur Verfügung.

Programmierungsumgebung:

- a) include-Dateien für die Verwendung der CAN-Schnittstelle sind `pcan.h` sowie `libpcan.h`.
- b) Die zuzubindende Bibliothek ist `libpcan.so`.
- c) Das zu verwendende Device ist `/dev/pcan24`.

- d) Mit Hilfe von **cat /proc/pcan** kann der Status des CAN-Knotens aus der Shell heraus abgefragt werden
- e) Um den PCAN-Dongle neu zu initialisieren, ist es nicht notwendig den Rechner neu zu starten. Mit dem Befehl **sudo /etc/init.d/pcan restart** wird die Reinitialisierung durchgeführt.

Punkteverteilung

Aufgabe	Punkte
CAN theoretisch	2 / 2 / 5 / 2 / 2 / 1 / 6
CAN praktisch	x