Hochschule Niederrhein
Fachbereich Elektrotechnik und Informatik
Labor für Echtzeitsysteme

Praktikum Echtzeitsysteme
Termine 1-3
B-I-5

Autor: Tobias Hahnen,
Matrikelnummer 1218710
Mike Wandels,
Matrikelnummer 1165207

Gruppe: C

Datum: 3. Januar 2020

Version: 1.0.0

Inhaltsverzeichnis

1 Beschreibung der Aufgabenstellung

Im Zuge des Praktikums Echtzeitsysteme soll eine Steuerung für eine Carrerabahn erstellt und erweitert werden, sodass ein Fahrzeug eine Bahn abfahren und ausmessen kann und danach gegen einen Gegner ein Rennen fahren kann. Diese Steuerung sollte dabei unabhängig von Auto, Carrerabahn und der verwendeten Spur sein.
Dazu fährt das Auto vor Rennbeginn die Strecke ab und misst anhand der Zeit, die das Auto durch die Lichtschranken braucht, die Länge der einzelnen Elemente und fügt diese einer Liste hinzu. Es gibt zwei Bahnen mit unterschiedlichen Elementen und einer jeweils anderen Anordnung, daher muss die Steuerung auch das beste aus der Anordnung herausholen.
Im anschliessenden Rennen soll sich das Fahrzeug gegen den Gegner behaupten, indem es auf die unterschiedlichen Stati reagiert. Das heisst, dass die Auslenkung in den Kurven beachtet werden muss sowie die Tatsache, ob sich das Fahrzeug auf der Innen- oder Aussenbahn befindet und bei vorkommenden Gefahrenstellen, ob sich bereits der Gegner darin befindet oder nicht und dementsprechend wartet oder durchfährt. Anhand der in den Stati übergebenen Informationen sollte dann auch eine dynamische Geschwindigkeitsanpassung implementiert werden, sodass auf unterschiedlichen Streckenabschnitten unterschiedliche Geschwindigkeiten gefahren werden können, um das meiste aus diesen Abschnitten herauszuholen, ohne dass das Fahrzeug aus der Bahn fliegt oder eine Kollision verursacht.

2 Bedienung der Applikation

Bedient wird die Applikation über die Kommandozeile, indem sie wie folgt aufgerufen wird:

```
$ ./race {Geschwindigkeit} {Runden}
```

Dabei gibt die *Geschwindigkeit* die zum Start des Rennens an, in der Erkundungsphase fährt das Fahrzeug in einer anderen.
Die *Runden* geben die Länge des Rennens an, die Erkundungsphase ist davon unabhängig.

Während des Rennens gibt es einige Ausgaben auf dem Bildschirm, die allerdings nur eine Information angeben, man kann nicht mit dem Programm interagieren, nachdem es gestartet wurde. Wenn man es allerdings per Keyboard Interrupt (Strg-C) abbricht, stoppt auch das Fahrzeug auf der Strecke.

3 Generierung und Installation

Zur Generierung wird das Build-Management-Tool *Make* und die dazugehörige Makefile benötigt. Ausserdem wird der Compiler *GCC/G++*, die Echtzeitbibliothek *RT* und ein Unixartiges Betriebssystem vorausgesetzt.
Die Erstellung des Programms erfolgt über die folgenden Aufrufe des Tools *Make*:

```
$ make clean
$ make all
```

Danach kann das Programm wie in der Sektion *Bedienung der Applikation* beschrieben aufgerufen werden.

4 Skizzierung der Lösung

Die Lösung zu der gegebenen Aufgabenstellung baut auf dem vorgegebenen Code-Gerüst auf.

4.1 Verbale Beschreibung der Lösung

Das Programm durchläuft im Wesentlichen zwei Schritte.

Der 1. Schritt ist die Erkundung und Vermessung der Strecke, dafür wird die Art des momentanen Elements durch das erhaltene Statuswort des Carrerabahn-Treibers ermittelt und die Länge mithilfe der Durchfahrtszeit und der angegebenen Geschwindigkeit errechnet. Diese Informationen werden dann in einer Liste für den zweiten Schritt abgelegt. Der Schritt ist beendet, sobald das Fahrzeug das Start/ Ziel Element erneut erreicht, danach wartet es auf den Start des Rennens, die durch eine simple Benutzereingabe erfolgt.

Der 2. Schritt, das eigentliche Rennen, besteht im Wesentlichen aus zwei Teilen, der eine behandelt unser Fahrzeug und der andere das des Gegners. Beide laufen in einem separaten Thread, sodass sie beide auf dieselben Ressourcen zugreifen können.
Der Gegnerthread dient zur Überwachung, an welcher Stelle der Rennbahn sich der Gegner befindet um die Kollisionsstelle fachgerecht zu handhaben. Dafür wird die Position des Gegners mit einem Zeiger auf das entsprechende Bahnelement gespeichert, sodass es vom Thread, der unser Fahrzeug handhabt, abgerufen werden kann.
Der Thread, der unser Fahrzeug steuert, handhabt die gesamte Steuerung, bestehend aus der dynamischen Geschwindigkeitsanpassung, der richtigen Reaktion an der Gefahrenstelle, basierend auf der Position des Gegners, sowie der Erkennung von Auslenkung in Kurven und der damit verbundenen Reaktion auf Möglichkeiten, aus der Bahn zu fliegen.
Dazu wird das aktuelle, vom Carrerabahn-Treiber eingelesene, Statuswort ausgewertet, welches zuerst auf die Auslenkung getestet wird, was bedeutet, dass sich das Fahrzeug in einer Kurve befindet. Sollte die Auslenkung zu hoch sein, wird dementsprechend reagiert. Danach wird das aktuelle Element mit dem des Gegners verglichen, um eine Kollision an der Gefahrenstelle zu vermeiden. Zuletzt wird die Geschwindigkeit anhand des derzeitigen Elements dynamisch ermittelt und angepasst.

Wenn das Rennen beendet wurde, nach der übergebenen Anzahl von Runden, bleibt das Fahrzeug am Start/ Ziel Element stehen und das Programm endet.

4.2 Datenflussdiagramm

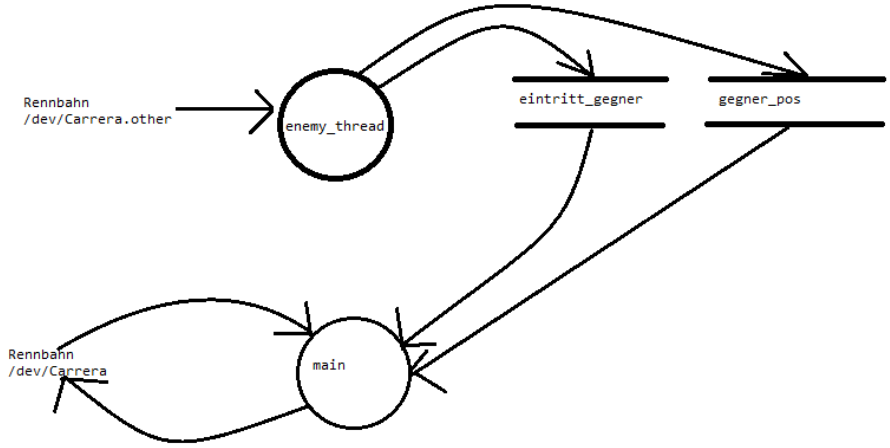

Abbildung 1: Treiber zu Gegnerthread und Mainthread

4.3 Strukturgramme: Mainthread

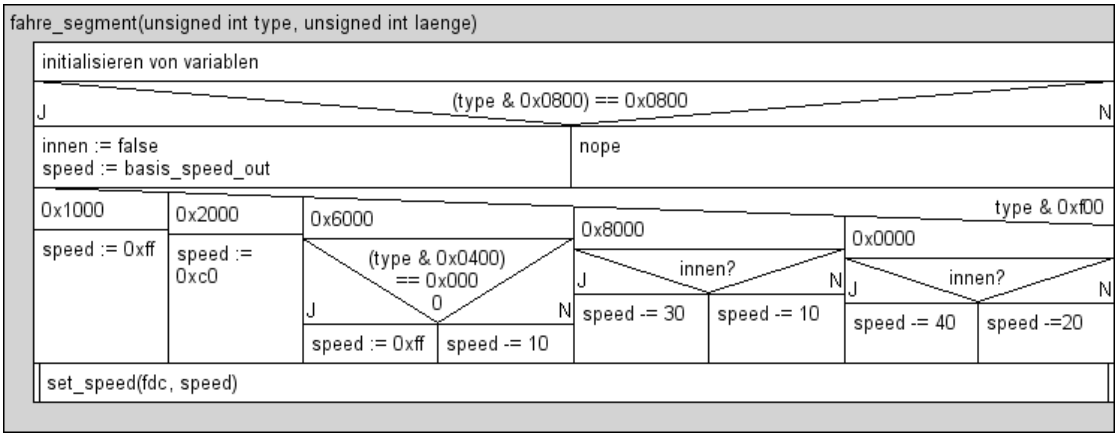

Abbildung 2: Funktion: *void fahre_segment(unsigned int type, unsigned int laenge)*

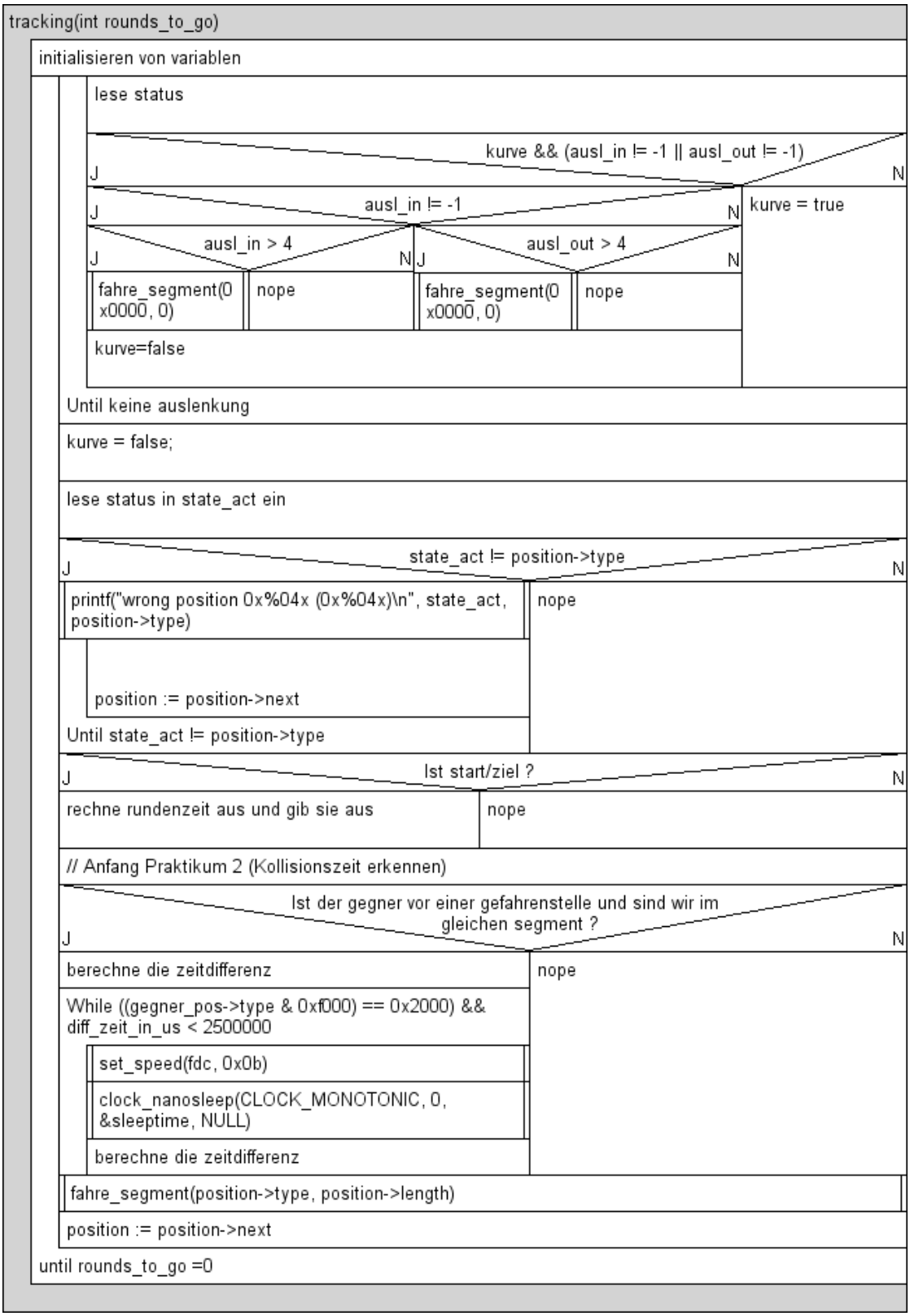

Abbildung 3: Funktion: *void tracking(int rounds_to_go)*

5 Vorbereitungen und Nachbereitungen der einzelnen Teilaufgaben

5.1 Termin No. 1

5.1.1 Längenberechnung der Kreuzungsbahn

Element	Länge
Start/Ziel → Gefahrenstelle 1	108 cm
Gefahrenstelle 1 → Kurve 1	105 cm
Kurve 1 → Kurve 2	91 cm
Kurve 2 → Gefahrenstelle 2	213 cm
Gefahrenstelle 2 → Kurve 3	88 cm
Kurve 3 → Kurve 4	112 cm
Kurve 4 → Start/Ziel	88 cm

Auto: Mercedes-Benz V8
Gesamtlänge: 8,05 m

5.1.2 Längenberechnung der Brückenbahn

Element	Länge
Start/Ziel → Kurve 1	97 cm
Kurve 1 → Kurve 2	93 cm
Kurve 2 → Brückenanfang	79 cm
Brückenanfang → Brückenende	131 cm
Brückenende → Kurve 3	28 cm
Kurve 3 → Kurve 4	91 cm
Kurve 4 → Start/Ziel	82 cm

Auto: Weiss 69
Gesamtlänge: 6,01 m

5.1.3 Unterschied zum Gegnerthread

Der Unterschied zum Gegnerthread liegt im 11. Bit, darin wird unterschieden, ob sich ein Fahrzeug aussen oder innen befindet, ergo müssen sich der Gegner und unser Fahrzeug darin unterscheiden.

5.2 Termin No. 2

5.2.1 Vorbereitung: Gegnerthread in erweiterter Form

```
void* enemy_thread(void) {
        int fde = open("/dev/Carrera.other", O_RDONLY);
        if (fde < 0) {
                perror("/dev/Carrera.other");
                return -1;
        }

        gegner_pos = root;
        ssize_t ret;
        __u16 alt, aktuell = 0;

        if ((ret = read(fde, &alt, sizeof(alt))) < 0) {
                perror("Enemy␣thread:␣read␣->␣alt");
                return NULL;
        }
```

```
        for (;;) {
                do {
                        if ((ret = read(fde, &aktuell, sizeof(aktuell))) < 0) {
                                perror("Enemy thread: read -> aktuell");
                                return NULL;
                        }

                        struct timespec deadline;
                        deadline.tv_sec = 0;
                        deadline.tv_nsec = 10000000;

                        clock_nanosleep(CLOCK_MONOTONIC, 0, &deadline, NULL);
                } while (alt == aktuell || (aktuell & 0xf000) == 0x0000);

                alt = aktuell;
                clock_gettime(CLOCK_MONOTONIC, &eintritt_gegner);

                while ((gegner_pos->type | 0x0800) != (aktuell | 0x0800)) {
                        gegner_pos = gegner_pos->next;
                }
        }
}
```

5.2.2 Kollisionsvermeidung

Testfall	t_0 - 5s	t_0	t_0	**Erwartetes Ergebnis**
1		G überf. S1	E überf. S1	KV; nach G S2 überf. hat, fährt E
2		E überf. S1		Keine KV
3	G überf. S1	E überf. S1		Keine KV
4		G überf. S4	E überf. S1	Keine KV
5		G überf. S1, bleibt stehen	E überf. S1	KV; E überf. S2 max. bei t_0 + dt +2s

ABNAHME:

5.3 Termin No. 3

5.3.1 Vorbereitung: Berücksichtigung Streckenlänge

Die Streckenlänge muss bei der dynamischen Geschwndigkeitsanpassung mitbeachtet werden, da damit die Geschwindigkeit anders angepasst werden kann. Wenn ein Element sehr lang ist, lohnt es sich auf die Höchstgeschwindigkeit zu beschleunigen, während es bei einem kurzen Element weniger lohnenswert ist, da im nächsten ggf. wieder gebremst werden muss und damit die Möglichkeit besteht aus der Bahn zu fliegen.