lea.schoenberger [☺] tu-dortmund.de
nils.hoelscher [☺] tu-dortmund.de
nick.pietrass [☺] tu-dortmund.de
jan.pomplun [☺] tu-dortmund.de

Übung zur Vorlesung
Eingebettete Systeme
Wintersemester 18/19

Aufgabenblatt 4 (Praxis)

(10 Punkte)

Hinweis: Abgabe des Theorieteils (einzeln oder in Zweiergruppen) bis zum 12.11.2018 um 10:00 durch Einwurf in den Briefkasten (Erdgeschoss OH16, gegenüber von Raum E16). Eine Abgabe per E-Mail ist *nicht* möglich. Besprechung: 14.-16.11.2018.

1 Vorbereitung (3 Punkte)

Hinweis: Diese Aufgabe muss abgegeben werden!

Lesen Sie noch vor Beginn der praktischen Übung die Kapitel 1 und 4 der OSEK-Specifications und beantworten Sie die folgenden Fragen.

- Wie wird ein Task im OSEK-Betriebssystem terminiert?
- OSEK unterscheidet zwischen zwei Tasktypen. Nennen Sie diese und erläutern Sie den Unterschied.
- In welchem Zustand befindet sich ein vom Scheduler aktivierter Task? Was zeichnet diesen zudem aus?

2 EV3OSEK Einrichten (2 Punkte)

Wählen Sie im CI-Lab die virtuelle Maschine ***es*** aus und melden Sie sich an. Auf dem Netzwerklaufwerk ***es*** finden Sie die Datei `04.zip`. Kopieren Sie sie in Ihr Home-Verzeichnes und entpacken Sie sie. Wechseln Sie in den Ordner `newlib` im Basisordner `ev3osek`, öffnen Sie ein Terminal und kompilieren Sie mit dem Befehl `make` die Standardbibliothek für EV3OSEK.

3 ECRobot API (5 Punkte)

Wechseln Sie in das Verzeichnis `../example/CollisionDetect`, in dem sich die Datei `collision.c` befindet. Öffnen Sie diese mit einem Texteditor (z.B. via `vim collision.c` oder `nano collision.c`).

Ergänzen Sie die durch Kommentare markierten Stellen, sodass der Roboter eine 180-Grad-Wende vor einem Hindernis macht. Wichtig ist es dabei, zu beachten, dass der Task `CheckDistance` alle 30ms ausgeführt wird. Speichern Sie Ihre Änderungen und kompilieren Sie das Programm mit dem Befehl `make`. Kopieren Sie anschließend die Dateien `boot.scr` und `*.bin` auf die MircoSD-Karte. Wenn die MicroSD-Karte wieder im Roboter platziert wurde, sollte das Programm ausgeführt werden.

Benutzen Sie zur Lösung dieser Aufgabe bitte die Kapitel 1 und 4 der OSEK-Specifications, die Sie bereits gelesen haben, sowie die folgende API:

```
int main(){
    …
    checkedDistance = CheckDistance();
    …
    if(checkedDist  > THRESHOLD){
        turn180();
    }
    …
```

```
void turn180(){
    // breaking
    ecrobot_set_motor_mode_speed(PORT, 1, 0);
    S32 motorDegr = ecrobot_get_motor_rev(PORT);
    S32 pos = (degr + 180) % 360;
    float dist = RADIUS * pi;
    int refs = (int) 360 * (dist / LENGTH_WHEELS);
}
```

Servo motor	Description
`S32 ecrobot_get_motor_rev(U8 port_id)`	Gets Servo Motor revolution value in degree. Wrapper of nxt_motor_get_count. Parameters: `port_id: EV3_PORT_1, EV3_PORT_2, EV3_PORT_3, EV3_PORT_4` Returns: Servo Motors revolution in degree

`void ecrobot_set_motor_speed(U8 port_id, S8 speed)`	Sets Servo Motor PWM value. Wrapper of nxt_motor_set_speed, but brake mode is fixed as brake.
	
	Parameters:
	`port_id: EV3_PORT_1, EV3_PORT_2, EV3_PORT_3, EV3_PORT_4`
	`speed: -100 to +100`

`void ecrobot_set_motor_mode_speed(U8 port_id, S32 mode, S8 speed)`	Sets Servo Motor brake mode and PWM value. Wrapper of nxt_motor_set_speed.
	
	Parameters:
	`port_id: EV3_PORT_1, EV3_PORT_2, EV3_PORT_3, EV3_PORT_4`
	`mode: 0(float), 1(brake)`
	`speed: -100 to +100`

`void ecrobot_set_motor_rev(U8 port_id, S32 rev)`	Sets Servo Motor revolution value in degree. Wrapper of nxt_motor_set_count.
	
	Parameters:
	`port_id: EV3_PORT_1, EV3_PORT_2, EV3_PORT_3, EV3_PORT_4`
	`rev: Servo Motors revolution in degree`

Abbildung 1: Motors API.

Ultrasonic sensor	Description
`void ecrobot_init_sonar_sensor(U8 port_id)`	Init a NXT sensor port for Ultrasonic Sensor. Parameters: `port_id: EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D`
`S32 ecrobot_get_sonar_sensor(U8 port_id)`	Get Ultrasonic Sensor measurement data in cm. Parameters: `port_id: EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D` Returns: Distance in cm (0 to 255), -1 (failure)

`void ecrobot_get_sonar_sensor_single_shot(U8 port_id, U8 data_buffer[8])`	Set the mode of the Lego ultrasonic sensor at the specified port to ULTRASONIC_MODE_SINGLE_SHOT. After that get the range of the Lego ultrasonic sensor connected at the specified port and store it in the buffer. The sensor meassures distances from 0 to 255 in cm. If nothing is located in front of the sensor, the value will be 255. All 8 entries of the array will be values returned by the sensor. If less than 8 objects are detected, some entries will be set to 255.
	
	Parameters:
	`port_id: EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D`
	`data_buffer: Buffer to store the result in`

`void ecrobot_term_sonar_sensor(U8 port_id)`	Terminate I2C used for for Ultrasonic. Parameters: `EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D`

Abbildung 2: Ultrasonic Sensor API.

Allgemeine Hinweise: Alle Übungstermine und weitere Informationen zur Veröffentlichung und Abgabe der Übungszettel sowie zum Erreichen der Studienleistung finden Sie unter

`https://ls12-www.cs.tu-dortmund.de/daes/de/lehre/lehrveranstaltungen/wintersemester-2018/es-1819.html`.