

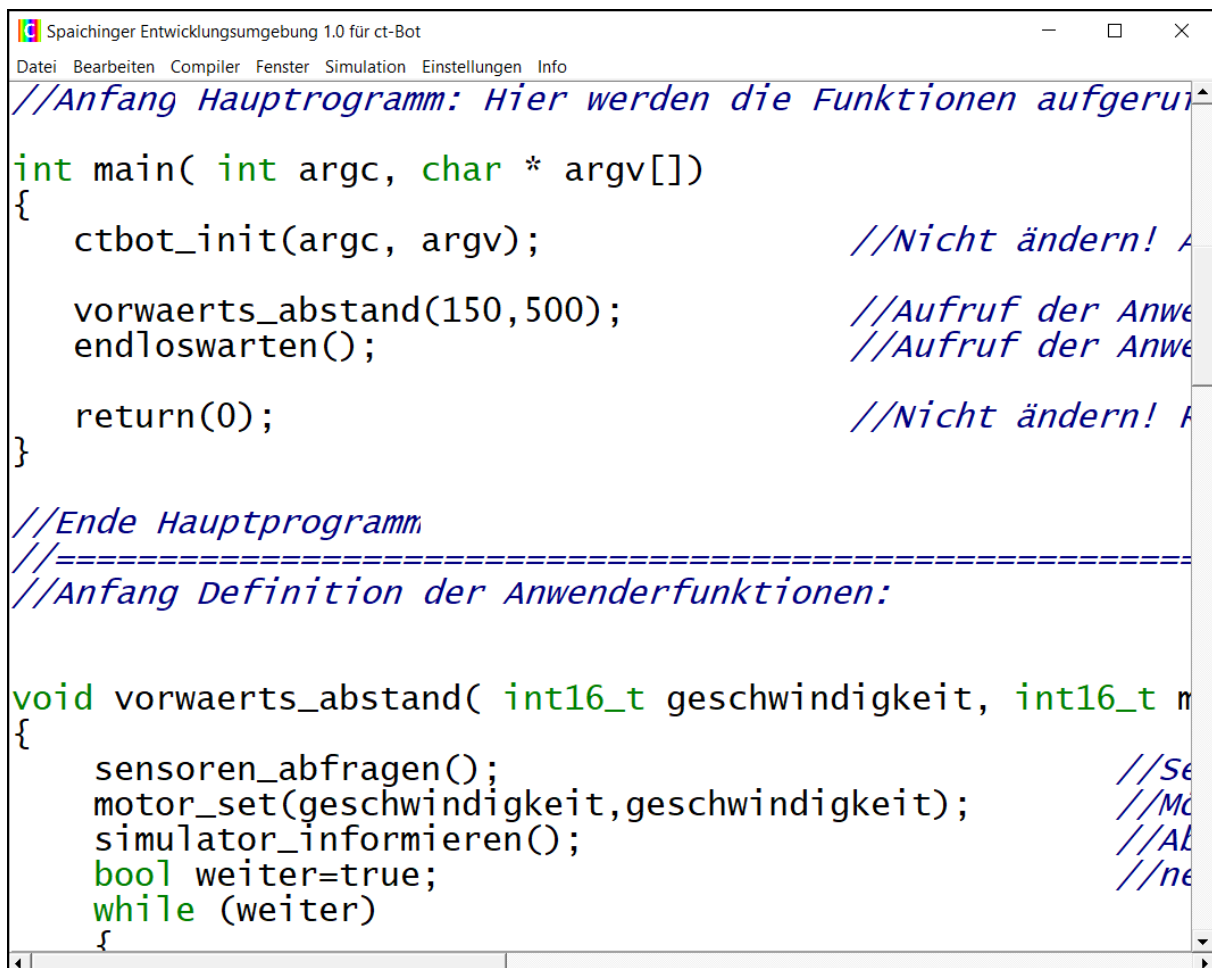
Bedienungsanleitung

Spaichinger

Entwicklungsumgebung 1.0

Zur C-Programmierung und Simulation von ct-Bots (Fahrrobotern)

Freeware für Microsoft Windows



```
Spaichinger Entwicklungsumgebung 1.0 für ct-Bot
Datei Bearbeiten Compiler Fenster Simulation Einstellungen Info
//Anfang Hauptprogramm: Hier werden die Funktionen aufgerufen
int main( int argc, char * argv[])
{
    ctbot_init(argc, argv);           //Nicht ändern! A
    vorwaerts_abstand(150,500);      //Aufruf der Anwe
    endloswarten();                  //Aufruf der Anwe
    return(0);                       //Nicht ändern! A
}

//Ende Hauptprogramm
//=====
//Anfang Definition der Anwenderfunktionen:

void vorwaerts_abstand( int16_t geschwindigkeit, int16_t m
{
    sensoren_abfragen();             //Se
    motor_set(geschwindigkeit,geschwindigkeit); //Mo
    simulator_informieren();         //Ab
    bool weiter=true;                //ne
    while (weiter)
    {
```

Dr. Markus Ziegler

www.spaichinger-schallpegelmesser.de

Dezember 2017

Inhaltsverzeichnis

1	Überblick	3
2	Installation	3
3	Copyright	4
4	Simulationsumgebung	5
5	C-Programmierung des ct-Bots	6
5.1	Wichtige Besonderheiten bei der Bot-Programmierung	7
6	Öffnen, Speichern und Weitergeben von Dateien	9
7	Problembehandlung	9
8	Softwarefehler	9

1 Überblick

Diese kostenlose und übersichtliche Entwicklungsumgebung ist ideal geeignet zum spielerischen Erlernen der Programmiersprache C im Kontext der Programmierung eines Fahrroboters "ct-Bot". Hierzu wird kein echter Roboter benötigt, da das Verhalten des Roboters beim Durchfahren von verschiedenen Labyrinthen simuliert wird. Der Roboter muss hierbei nicht nur durch das Labyrinth hindurch finden, sondern auch darauf achten, dass er in kein "Loch" (schwarzes Feld) hineinfällt. Zur Wahrnehmung seiner Umwelt stehen dem Roboter verschiedene Sensoren zur Verfügung:

- zwei Encoder zur Bestimmung der Radumdrehungen der zwei Antriebsräder
- zwei Abstandssensoren
- zwei Liniensensoren zur Erkennung von Linien auf dem Boden
- zwei Abgrundsensoren zur Erkennung von "Löchern" im Boden
- einen "Maussensor" zur Messung der Geschwindigkeit
- zwei Lichtsensoren

Zudem besitzt er an den zwei Antriebsrädern jeweils einen Motor und 8 LEDs (Aktoren).

Die simulierten Sensoren verhalten sich wie reale Sensoren.

Die Spaichinger Entwicklungsumgebung läuft unter dem Betriebssystem Microsoft Windows (Windows 7, Windows 8, Windows 8.1 und Windows 10) auf jedem Notebook oder PC. Die Downloaddatei "ctBotSpaichingen.zip" umfasst alle benötigten Komponenten: Entwicklungsumgebung, Compiler und Simulationssoftware.

2 Installation

Die Downloaddatei "ctBotSpaichingen.zip" brauchen Sie nur in einem beliebigen Verzeichnis zu entpacken. Mit einem Klick auf "ctBotstart.exe" startet dann die Entwicklungsumgebung „Spaichinger Entwicklungsumgebung für ct-Bot“. Die Software braucht also nicht installiert zu werden (portable Software) und kann sogar von einem USB-Stick aus betrieben werden. Beim Start erzeugt die Software den Ordner „ctBot_Dateien“ im Verzeichnis „Dokumente“. Dort werden die erzeugten Dateien abgespeichert.

Sonst nimmt die Software keinerlei Änderungen an Ihrem Betriebssystem vor. Es erfolgt auch kein Eintrag in die Windows-Registry.

3 Copyright

Der Entwickler und Programmierer

Dr. Markus Ziegler

78549 Spaichingen

www.spaichinger-schallpegelmesser.de

E-Mail: ziegler@spaichinger-schallpegelmesser.de

stellt die Spaichinger Entwicklungsumgebung (ctBotstart.exe) kostenlos zur Verfügung (Freeware).

Alle anderen in der Downloaddatei "ctBotSpaichingen.zip" enthalten Dateien können ebenfalls kostenlos verwendet werden:

- Die gelinkten Dateien für den ct-Bot (-> Header) wurden von der Zeitschrift c't entwickelt (Open Source: GNU-Lizenz) und von Markus Ziegler an die Bedürfnisse der Schule angepasst. Die ursprünglichen Dateien findet man unter <https://www.heise.de/ct/projekte/machmit/ctbot/browser>
- Der Simulator ct-Sim wurde von der Zeitschrift c't entwickelt (Open Source: GNU-Lizenz) und von Markus Ziegler an die Bedürfnisse der Schule angepasst. Die ursprünglichen Dateien findet man unter <https://www.heise.de/ct/projekte/machmit/ctbot/browser>
- Der verwendete C-Compiler "gcc" in der Fassung von MinGW ist ebenfalls "Open Source" und steht unter der GNU-Lizenz. Download: <http://mingw.org/>
- Das verwendete Java 1.6 darf ebenfalls kostenlos genutzt werden.

Ich bedanke mich herzlich bei der Fachzeitschrift c't und allen anderen Programmierern, die bei der Entwicklung des ct-Bots, der Simulationsumgebung ct-Sim und des GNU C-Compilers gcc eine tolle Arbeit geleistet haben und ihr Werk frei zur Verfügung stellen.

4 Simulationsumgebung

Nach dem erfolgreichen Kompilieren kann die Simulationsumgebung (ct-Sim Version Spaichingen) durch Betätigen von „Simulation“ oder der Funktionstaste F7 aus der Spaichinger Entwicklungsumgebung heraus gestartet werden.

1. Gewünschtes Labyrinth auswählen

2. Testbot (= eigener Bot) in das Labyrinth setzen

3. Testbot starten

4. ggfs. Pause betätigen

Neustart: Falls ein Neustart des Bots durchgeführt werden soll: Nacheinander wieder 2. und 3. durchführen

Ziel

Loch

Startposition

Bot

Zeit: 00:00:06.520

Zeitlupe: 0

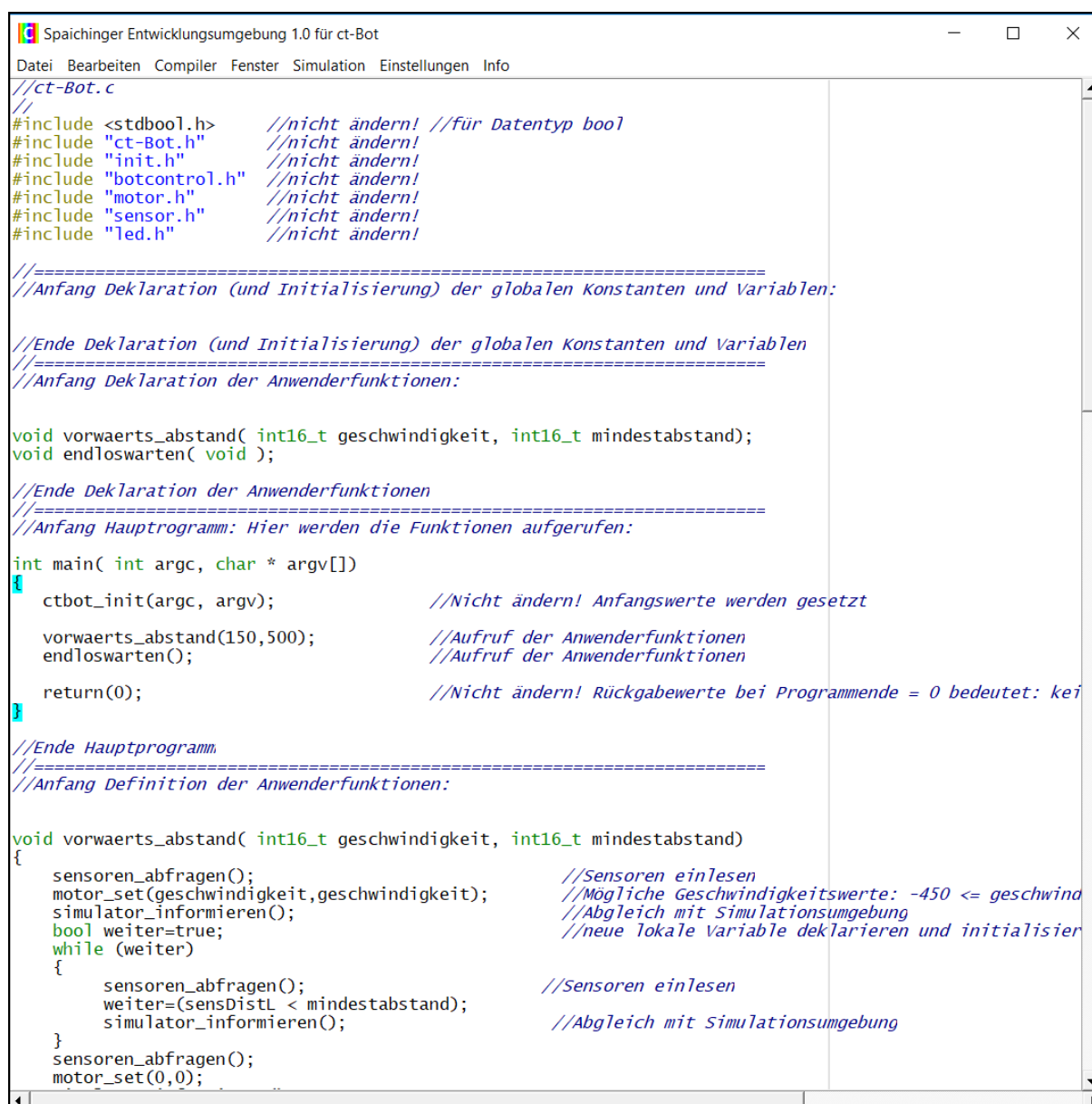
Info: Bot Sim-Bot setzt seine ID selbst auf:128
Info: Setze die Id von Bot Sim-Bot auf 128

Zusätzlicher Bot: Falls zwei Bots sich gleichzeitig im Labyrinth bewegen sollen, kann unter „Bots“ ein weiterer Bot geladen werden.

Ggfs. Zeitlupe einstellen, damit die Bewegung des Bots und die Sensorwerte genauer beobachtet werden können.

5 C-Programmierung des ct-Bots

Beim Start der Software „Spaichinger Entwicklungsumgebung für ct-Bot“ (ctBotstart.exe) wird automatisch ein einfaches C-Programm für den ct-Bot geladen. Dieses Programm kann sofort durch Betätigen des Buttons „Compiler“ oder der Funktionstaste F6 kompiliert werden. Anschließend kann durch Betätigen des Buttons „Simulation“ oder der Funktionstaste F7 die Simulation gestartet werden (siehe Kapitel [Simulationsumgebung](#)).



```
Spaichinger Entwicklungsumgebung 1.0 für ct-Bot
Datei Bearbeiten Compiler Fenster Simulation Einstellungen Info
//ct-Bot.c
//
#include <stdbool.h> //nicht ändern! //für Datentyp bool
#include "ct-Bot.h" //nicht ändern!
#include "init.h" //nicht ändern!
#include "botcontrol.h" //nicht ändern!
#include "motor.h" //nicht ändern!
#include "sensor.h" //nicht ändern!
#include "led.h" //nicht ändern!

//=====
//Anfang Deklaration (und Initialisierung) der globalen Konstanten und Variablen:

//Ende Deklaration (und Initialisierung) der globalen Konstanten und Variablen
//=====
//Anfang Deklaration der Anwenderfunktionen:

void vorwaerts_abstand( int16_t geschwindigkeit, int16_t mindestabstand);
void endloswarten( void );

//Ende Deklaration der Anwenderfunktionen
//=====
//Anfang Hauptprogramm: Hier werden die Funktionen aufgerufen:

int main( int argc, char * argv[])
{
    ctbot_init(argc, argv); //Nicht ändern! Anfangswerte werden gesetzt

    vorwaerts_abstand(150,500); //Aufruf der Anwenderfunktionen
    endloswarten(); //Aufruf der Anwenderfunktionen

    return(0); //Nicht ändern! Rückgabewerte bei Programmende = 0 bedeutet: kei
}

//Ende Hauptprogramm
//=====
//Anfang Definition der Anwenderfunktionen:

void vorwaerts_abstand( int16_t geschwindigkeit, int16_t mindestabstand)
{
    sensoren_abfragen(); //Sensoren einlesen
    motor_set(geschwindigkeit,geschwindigkeit); //Mögliche Geschwindigkeitswerte: -450 <= geschwind
    simulator_informieren(); //Abgleich mit Simulationsumgebung
    bool weiter=true; //neue lokale Variable deklarieren und initialisier
    while (weiter)
    {
        sensoren_abfragen(); //Sensoren einlesen
        weiter=(sensDistL < mindestabstand); //Abgleich mit Simulationsumgebung
        simulator_informieren();
    }
    sensoren_abfragen();
    motor_set(0,0);
}
```

Dieses Beispielprogramm veranlasst den Bot geradeaus zu fahren, bis ein Hindernis auftaucht und dann anzuhalten und zu warten.

Das Beispielprogramm habe ich sehr ausführlich dokumentiert, damit es als Ausgangspunkt für eigene Programme dienen kann. Dieses Beispielprogramm kann durch Betätigen von „Datei → neu ct-Bot.c“ jederzeit wieder hergestellt werden.

Im unteren Teil des Programms finden Sie alle globalen Variablen aufgelistet, in denen die Sensorwerte abgespeichert sind:

```
Spaichinger Entwicklungsumgebung 1.0 für ct-Bot
Datei Bearbeiten Compiler Fenster Simulation Einstellungen Info
    sensoren_abfragen();
    simulator_informieren();
}
}

// Ende Definition der Anwenderfunktionen
//=====

//Informationen:
//Überblick über die vorhandenen Sensorwerte
//Diese sind alle als globale Variablen bereits in sensor.c deklariert.
//Alle Sensorwerte werden durch den Aufruf sensoren_abfragen() aktualisiert.
//Vom Anwender dürfen die Sensordaten nur gelesen,
//aber nicht geändert werden!
//
// int16_t sensEncPulsL = 0; /*!< Encoder linkes Rad Summe der Pulse*/
// int16_t sensEncPulsR = 0; /*!< Encoder rechtes Rad Summe der Pulse*/
//
// int32_t sensEncl = 0; /*!< Encoder linkes Rad Summe der Pulse*/
// int32_t sensEncr = 0; /*!< Encoder rechtes Rad Summe der Pulse*/
//
// int16_t sensLDRL = 0; /*!< Lichtsensor links */
// int16_t sensLDRR = 0; /*!< Lichtsensor rechts */
//
// int16_t sensDistL = 0; /*!< linker Abstandssensor (IR-Sensor) */
// int16_t sensDistR = 0; /*!< rechter Abstandssensor (IR-Sensor) */
//
// int16_t sensBorderL = 0; /*!< Abgrundsensoren links */
// int16_t sensBorderR = 0; /*!< Abgrundsensoren rechts */
//
// int16_t sensLineL = 0; /*!< Linienensensor links */
// int16_t sensLineR = 0; /*!< Linienensensor rechts */
//
//
// int8_t sensMouseDX; /*!< Maussensor Delta X, (Mausgeschwindigkeit) positive Werte zeigen q
// int8_t sensMouseDY; /*!< Maussensor Delta Y, (Mausgeschwindigkeit) positive Werte zeigen i
//
//
//Überblick über die möglichen Aktoren:
//
// Ansteuern der Antriebsmotoren:
// motor_set(int16_t geschwindigkeit_L, int16_t geschwindigkeit_R); //Werte zwischen -450 und 450 mögli
//
// Ein- und Ausschalten von einzelnen LEDs:
// LED_on(uint8_t LED)
// LED_off(uint8_t LED)
// hierbei muss einen der folgenden Konstanten für LED eingesetzt werden:
// LED_ROT, LED_ORANGE, LED_GELB, LED_GRUEN, LED_TUERKIS, LED_WEISS, LED_ALL
// Ausgabe einer 8-Bit-Variablen als LED-Kette:
// LED_set(uint8_t LED)
```

Zusätzlich finden Sie hier einen Überblick über die Ansteuerung der möglichen Aktoren (Motoren und LEDs).

5.1 Wichtige Besonderheiten bei der Bot-Programmierung

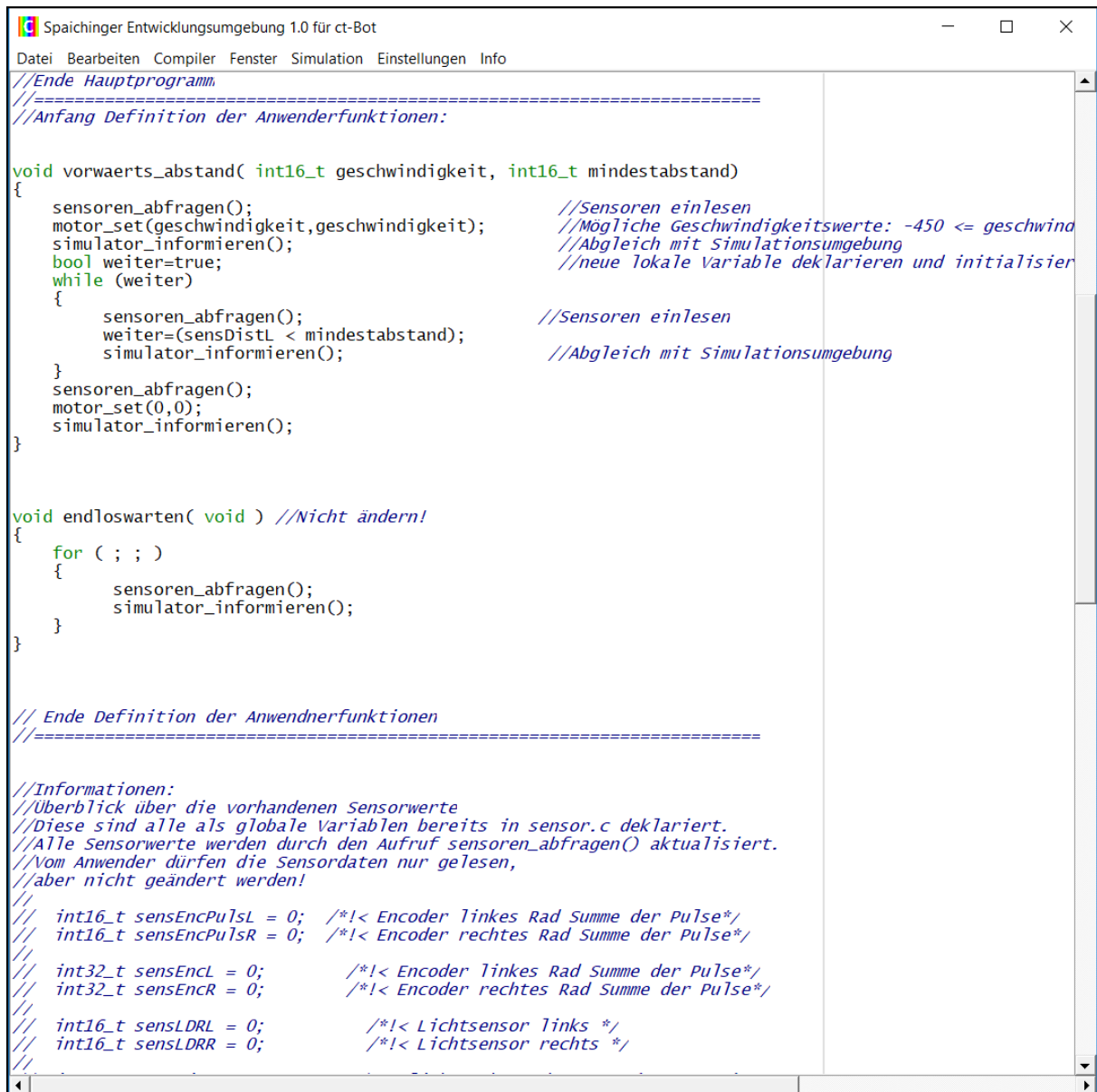
Mit den obigen Informationen und den üblichen Ansi-C-Befehlen lässt sich der Bot problemlos programmieren, wenn folgende Besonderheiten beachtet werden:

- Jedes Mal **bevor** auf ein Aktor oder ein Sensorwert zugegriffen wird, **muss** die Funktion **sensoren_abfragen()** aufgerufen werden.
- Nach dem Aufruf eines Aktors oder eines Sensorwertes **muss** die Funktion **simulator_informieren()** aufgerufen werden.
- Der Kopf des Programms: Die Zeilen zwischen „//ct-Bot.c“ und „//Anfang Deklaration (und Initialisierung) der globalen Konstanten und Variablen:“

dürfen nicht geändert werden. D.h., es dürfen keine Headerdateien entfernt oder hinzugefügt werden.

Mit den beiden Funktionen **sensoren_abfragen()** und **simulator_informieren()** werden die Sensorwerte aktualisiert und ein Abgleich mit der Simulationsumgebung geschaffen. Ohne diesen Abgleich endet die Simulation mit der Fehlermeldung „Ein oder mehrere Bots waren viel zu langsam“.

Die richtige Verwendung der Funktionen **sensoren_abfragen()** und **simulator_informieren()** ist im Beispielprogramm gut ersichtlich:



```
//Ende Hauptprogramm
//=====
//Anfang Definition der Anwenderfunktionen:

void vorwaerts_abstand( int16_t geschwindigkeit, int16_t mindestabstand)
{
    sensoren_abfragen(); //Sensoren einlesen
    motor_set(geschwindigkeit,geschwindigkeit); //Mögliche Geschwindigkeitswerte: -450 <= geschwind
    simulator_informieren(); //Abgleich mit Simulationsumgebung
    bool weiter=true; //neue lokale Variable deklarieren und initialisieren
    while (weiter)
    {
        sensoren_abfragen(); //Sensoren einlesen
        weiter=(sensDistL < mindestabstand); //Abgleich mit Simulationsumgebung
        simulator_informieren();
    }
    sensoren_abfragen();
    motor_set(0,0);
    simulator_informieren();
}

void endloswarten( void ) //Nicht ändern!
{
    for ( ; ; )
    {
        sensoren_abfragen();
        simulator_informieren();
    }
}

// Ende Definition der Anwenderfunktionen
//=====

//Informationen:
//Überblick über die vorhandenen Sensorwerte
//Diese sind alle als globale Variablen bereits in sensor.c deklariert.
//Alle Sensorwerte werden durch den Aufruf sensoren_abfragen() aktualisiert.
//Vom Anwender dürfen die Sensordaten nur gelesen,
//aber nicht geändert werden!
//
// int16_t sensEncPulsL = 0; /*!< Encoder linkes Rad Summe der Pulse*/
// int16_t sensEncPulsR = 0; /*!< Encoder rechtes Rad Summe der Pulse*/
//
// int32_t sensEncl = 0; /*!< Encoder linkes Rad Summe der Pulse*/
// int32_t sensEncr = 0; /*!< Encoder rechtes Rad Summe der Pulse*/
//
// int16_t sensLDRL = 0; /*!< Lichtsensor links */
// int16_t sensLDRR = 0; /*!< Lichtsensor rechts */
//
//
```


6 Öffnen, Speichern und Weitergeben von Dateien

Ihr C-Programm können Sie wie üblich über „Datei → speichern“ oder die Funktionstaste F5 speichern. Den Speicherort können Sie unter „Datei → speichern unter“ auswählen. Mit „Datei → öffnen“ können Sie ihr ct-Bot-Programm wieder öffnen. Ihren fertig programmierten ct-Bot.exe finden Sie unter „Dokumente“ im Ordner „ctBot_Dateien\ctBotexe“. Dies ist z.B. wichtig, wenn Sie zwei unterschiedliche Bots in der Simulationsumgebung gleichzeitig fahren lassen wollen. In diesem Fall kopieren Sie den 2. Bot unter einem anderen Namen ebenfalls in den Ordner „ctBot_Dateien\ctBotexe“. In der Simulationsumgebung können Sie diesen dann zusätzlich laden (siehe Kapitel [Simulationsumgebung](#)).

Achtung! die Datei „**libpthread-2.dll**“, die sich ebenfalls im Ordner „ctBot_Dateien\ctBotexe“ befindet, darf **nicht gelöscht** werden, da die Bots sonst nicht mehr funktionsfähig sind!

7 Problembehandlung

- Falls nach dem ersten Start der Compiler auch nach einiger Wartezeit immer noch die Meldung „Compiler ist aktiv“ anzeigt, hat der Compiler nicht den richtigen Pfad gefunden. Abhilfe: Schließen Sie Software „Spaichinger Entwicklungsumgebung für ct-Bot“ (ctBotStart.exe) und starten sie diese dann neu. Anschließend müsste dann der Compiler richtig funktionieren.
- Falls in der Simulationsumgebung die Fehlermeldung „Ein oder mehrere Bots waren viel zu langsam“ erscheint, wurden die beiden Funktionen **sensoren_abfragen()** und **simulator_informieren()** in Ihrem C-Programm nicht richtig verwendet (siehe Kapitel [Wichtige Besonderheiten bei der Bot-Programmierung](#))
- Falls ihr Bot in der Simulationsumgebung nicht startet, könnte es daran liegen, dass die Datei „**libpthread-2.dll**“ im Ordner „Dokumente\ctBot_Dateien\ctBotexe“ versehentlich gelöscht wurde. Diesen Fehler können Sie beheben, indem Sie die Software „Spaichinger Entwicklungsumgebung für ct-Bot“ schließen und anschließend den gesamten Ordner „Dokumente\ctBot_Dateien“ löschen. Beim Neustart der Software „Spaichinger Entwicklungsumgebung für ct-Bot“ wird dann dieser Ordner mit allen Unterordnern und der Datei „libpthread-2.dll“ wieder neu erzeugt.

Bei weiteren Fragen oder Problemen wenden Sie sich bitte per E-Mail an Ziegler@spaichinger-schallpegelmesser.de.

8 Softwarefehler

Bei Fragen oder Problemen wenden Sie sich bitte per E-Mail an Ziegler@spaichinger-schallpegelmesser.de.

Falls Sie Fehler in der Software entdecken, bin ich für einen Hinweis per E-Mail immer dankbar.