Technische Informatik
Prof. Dr. M. Bogdan

UNIVERSITÄT LEIPZIG
Institut für Informatik

Praktikum Eingebettete Systeme

WS 2015/16

Übungsblatt 3

Abnahme: 01.12.2015 von 15:15 Uhr bis 17:15 Uhr

Hinweise: Die Hardware steht Ihnen im Hardwarelabor P535 zur Verfügung.
Auf den Laborrechnern ist für die Lösung der Praktikumsaufgaben die Arduino IDE installiert.

Tipp: Die Aufgaben bauen gelegentlich aufeinander auf. Nutzen Sie daher Ihren bereits abgenommenen Arbeitsstand, um den Arbeitsaufwand zu reduzieren. Sie sollten aus Eigeninteresse einen wiederverwendbaren und lesbaren Quellcode schreiben.

Aufgabe 1: Fragen im Vorfeld

1. Was ist ein Scheduler, wann ist der Einsatz eines Schedulers empfehlenswert ?
2. Was unterscheidet Präemptives-Multitasking vom Kooperativen-Multitasking ?
3. Was Unterscheidet die Von-Neumann-Architektur von einer Harvard-Architektur ?
4. Was bedeutet in C folgende Deklaration: “double (*f)();“
5. Ist die Deklaration “double (*f)();“ identisch mit “double *f();“ ?
   Wenn nein, welche Unterschiede resultieren aus den beiden Deklarationen ?
6. Was wird mit der folgenden, komplexen Deklaration erreicht: “void (*f2)(double (*f)());“ ?
7. Welche Auswirkung hätte die folgende Methode allgemein auf eine Klasse:
   “virtual void MethodeXY(void) = 0;“

Aufgabe 2: Scheduler über Funktionszeiger

1. Implementieren Sie auf dem Arduino Uno einen Scheduler, der kooperatives Multitasking mit den folgenden Vorgaben ermöglicht:

```
bool addTask(void (* func)(void), unsigned long timer, bool reshot = true);

void runTasks(void);
```

Die Funktion "addTask()" soll einen Funktionszeiger übernehmen. Dieser Funktionszeiger soll die Adresse der Funktion beinhalten, den der Scheduler ausführen soll. Als weitere Übergabeparameter soll "timer" die Zeit bis zum Funktionsaufruf beinhalten und der Parameter "reshot" die wiederholte Ausführung steuern.[1] Die Funktion runTasks() soll in der Main-Loop aufgerufen werden und stets alle Tasks ausführen, die zeitlich an der Reihe sind. Die Ausführungs-Prioritäten sollen sich aus der Reihenfolge ergeben, in der die Tasks hinzugefügt wurden.

Aufgabe 3: Beispielanwendung des Scheduler über Funktionszeiger

Implementieren Sie nach den folgenden Vorgaben ein kleines Beispielprogramm. Dieses soll den in Aufgabe 2 entwickelten Scheduler nutzen.

1. Task 1 soll lediglich die LED 1 in einer Periode von einer Sekunde blinken lassen.
2. Task 2 soll beim Drücken des Tasters S1 die LED 2 solange Leuchten lassen, wie der Taster gedrückt wird.

[1] Ist reshot true, soll der Task periodisch ausgeführt werden

Aufgabe 4: Scheduler über Taskobjekte

1. Implementieren Sie auf dem Arduino Uno einen objektorientierten Scheduler, der kooperatives Multitasking nach den folgenden Vorgaben ermöglicht:

Im Sinne der objektorientierten Programmierung wäre es wünschenswert, statt Funktionszeiger, diesmal Tasks als Objekte an den Scheduler zu übergeben. Passen Sie hierfür die Methode "addTask()" entsprechend an. Sie können auf diesem Weg, die im Tasks benötigten Daten als Klassenvariablen kapseln, um globalen Variablen zu vermeiden. Um einen Task aus dem Scheduler zu entfernen, fügen Sie die Methode removeTask() hinzu.

```
bool addTask(Task * task, unsigned long timer, bool reshot = true);
void removeTask(Task* task);
void runTasks(void);

class Task
{
        public:
        virtual void update(void) = 0;
};
```

Die Klasse Task ist eine rein abstrakte Klasse. Alle speziellen Tasks sollen ebenfalls als Klassen definiert werden und von dieser Klasse erben. Der Scheduler soll, nachdem die Ausführungszeit für einen Task gekommen ist, diesen Task über die Methode void update(void) der entsprechenden Taskklasse starten.

Aufgabe 5: Beispielanwendung des Scheduler über Taskobjekte

Realisieren Sie die Anwendung aus dem Aufgabenblatt 2, Aufgabe 4 mittels eines Scheduler über Taskobjekte.

1. Implementieren Sie den Sende- und Empfangsvorgang jeweils über einen Task.
2. Ein zweiter Task soll die aufgesteckte Zusatz-PWM-LED periodisch auf oder abdimmen. Handelt es sich um die Sendeeinheit, so soll die LED stets von 0 auf 100 Prozent herauf[2] geregelt werden. Bei der Empfangseinheit soll diese stets von 100 auf 0 Prozent abgedimmt werden.

[2]Nach dem die LED eine Helligkeit von 100% erreicht hat, soll sich diese **stufenlos** zurück auf 0% setzen.