Class documentatie

Car Control classes

Douwe Osinga, Niek Aukes (met assistentie van aidan Coller)

Inhoud

[Multitasker class 2](#_Toc23866671)

[Private variabelen 2](#_Toc23866672)

[Public variabelen 2](#_Toc23866673)

[Constructors 3](#_Toc23866674)

[Tasker 3](#Tasker)

# Multitasker class

Het doel van de multitasker class is om ervoor te zorgen dat de arduino getimede functies kan uitvoeren.

Deze class heeft een singelton design, wat betekent dat er slechts één instantie per programma kan zijn.

## Multitasker class

### Private variabelen

De Multitasker class gebruikt de volgende private variabelen:

* unsigned long RegisteredTimers[20]

Dit zijn alle timers die de class registreerd van type unsigned long.

Je kunt het maximaal aantal timers aanpassen door de array lists te vergroten of verkleinen.

* void (\*RegisteredTasks[20])()

Dit zijn de bijbehorende voids die de class moet uitvoeren op een specifiek punt.

* void (\*RegisteredCallbacks[20])()

Dit zijn de callbacks die de class moet maken nadat de geregistreerde functie is uitgevoerd.

Deze variabelen zijn private, dus je kunt niet bij deze variabelen tijdens de runtime. Ze zijn hier alleen genoemd voor de begrijpelijkheid van de code.

### Public variabelen

* static MultiTasker\* Tasker

De pointer naar de instantie van de multitasker die wordt gemaakt in het programma.

* IMultiTasker\* ClassMultiTaskers[10]

De ClassMultiTaskers zijn alle multitaskers die gespecialiseerd zijn in classes callen.

### Constructors

* MultiTasker()

Maakt een nieuwe [*Tasker*](#Tasker), die alle kleinere multitaskers checkt.

### Methods

* int RegisterTask(void (\*Task)(),

unsigned long AfterMilliSeconds, void (\*Callback)() = nullptr)

Registreerd een nieuwe task die wordt gecallt nadat de gespecificeerde tijdsperiode is verstreken.

**Void (\*Task)** is de functie die moet worden gecallt, **AfterMilliSeconds** is de tijd in milliseconden waarna de functie moet worden gecallt, en **CallBack** is de functie die eventueel direct na Task moet worden gecallt.

* void Distribute()

Deze functie zorgt ervoor dat de functies op tijd worden uitgevoerd. Distribute() checkt voor elke multitasker en voor elke task in die multitasker of de gespecificeerde tijd verstreken is, en als dat zo is, activeerd hij de functie, en daarna eventueel de callback, waarna hij de timer verwijderd, zodat de functie niet meer wordt gecallt.

## IMultitasker class

* class IMultiTasker

Is puur voor templates, zodat de gebruiker een eigen definitie kan stellen aan de instantie van de multitasker.

### Method(s)

* virtual void Distribute() = 0

Wordt elke frame gecalled, er is verder geen definitie dus het doet nog niks, je kunt zelf het iets laten doen.

## Class Multitasker

* template <class T>

class ClassMultiTasker : public IMultiTasker

Gespecialiseerde multitasker voor classes, gedefinieerd als ‘T’.

### Constructors

* ClassMultiTasker()

Maakt een nieuwe tasker specifiek voor classes, die alle andere kleinere multitaskers houdt en checkt voor functies die moeten worden geaciveerd.