WFO: Arduino Informatik-Schulprojekt

Erzeugt von Doxygen 1.8.11

Inhaltsverzeichnis

Kapitel 1

Klassen-Verzeichnis

1.1 Auflistung der Klassen

Hier folgt die Aufzählung aller Klassen, Strukturen, Varianten und Schnittstellen mit einer Kurzbeschreibung:

Kapitel 2

Klassen-Dokumentation

2.1 AudioSensor Klassenreferenz

Diese Klasse ermöglicht die Verwendung des Audiosensor.

```
#include <AudioSensor.h>
```

Öffentliche Methoden

- AudioSensor (unsigned dig_pin, unsigned ana_pin=0)
- bool laut ()
- void einschalten ()
- void ausschalten ()
- int missLautstaerke ()

2.1.1 Ausführliche Beschreibung

Diese Klasse ermöglicht die Verwendung des Audiosensor.

2.1.2 Beschreibung der Konstruktoren und Destruktoren

2.1.2.1 AudioSensor::AudioSensor (unsigned *dig_pin,* unsigned *ana_pin* = `0`) [inline],[explicit]

Konstruktor

Erstellt ein Objekt, welches zur Abfrage des Audiosensor dient. Beispiel:

```
AudioSensor sensor;
```

2.1.3 Dokumentation der Elementfunktionen

2.1.3.1 void AudioSensor::ausschalten ( ) [inline]

Schaltet den Sensor aus. Beispiel:

```
sensor.ausschalten();
```

2.1.3.2 void AudioSensor::einschalten ( ) [inline]

Schaltet den Sensor ein. Beispiel:

```
sensor.einschalten();
```

2.1.3.3 bool AudioSensor::laut ( ) [inline]

Prüft, ob die aktuelle Lautstärke den Grenzwert überschritten wurde. Der Grenzwert wurde mit der "↩ Einstellschraube" justiert. Beispiel:

```
if( sensor.laut() ){
 // mach irgendwas
}
```

2.1.3.4 int AudioSensor::missLautstaerke ( ) [inline]

Der Grenzwert wurde mit der "Einstellschraube" justiert. Beispiel:

```
int lautstaerke = sensor.missLautstaerke();
```

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- audiosensor/AudioSensor.h

2.2 Display Klassenreferenz

Diese Klasse ermöglicht die Verwendung und gezielte Ansteuerung der Anzeige.

`#include <Display.h>`

Abgeleitet von LiquidCrystal_I2C.

Öffentliche Methoden

- Display ()
- void initialisiere ()
- void geheErsteZeile ()
- void geheZweiteZeile ()
- void gehePosition (unsigned zeile, unsigned position)
- void loescheText ()
- void schreibeText (const char text[])
- void schreibeText (int num)
- void schreibeText (float num, int dec=1)

2.2.1 Ausführliche Beschreibung

Diese Klasse ermöglicht die Verwendung und gezielte Ansteuerung der Anzeige.

2.2.2 Beschreibung der Konstruktoren und Destruktoren

2.2.2.1 Display::Display ( ) [inline],[explicit]

Konstruktor

Erstellt ein Objekt, welches die Ansteuerung der Anzeige ermöglicht. Beispiel:

```
Display anzeige;
```

2.2.3 Dokumentation der Elementfunktionen

2.2.3.1 void Display::geheErsteZeile ( ) [inline]

Setzt den Cursor der Anzeige auf den Beginn der ersten Zeile. Beispiel:

```
anzeige.geheErsteZeile();
```

2.2.3.2 void Display::gehePosition (unsigned *zeile,* unsigned *position*) [inline]

Setzt den Cursor der Anzeige an die jeweilige Position.

Parameter

zeile	Zeilennummer (0 oder 1)
position	Positionsnummer (0 ... 15)

Beispiel:

```
anzeige.gehePosition(1, 10);
```

2.2.3.3 void Display::geheZweiteZeile ( ) [inline]

Setzt den Cursor der Anzeige auf den Beginn der zweiten Zeile. Beispiel:

```
anzeige.geheZweiteZeile();
```

2.2.3.4 void Display::initialisiere ( ) [inline]

Initialisiert die Anzeige. Beispiel:

```
anzeige.initialisiere();
```

2.2.3.5 void Display::loescheText ( ) [inline]

Löscht den gesamten Anzeigetext.

Beispiel:

```
anzeige.loescheText();
```

2.2.3.6 void Display::schreibeText (const char *text[ ]*) [inline]

Gibt den jeweiligen Text auf dem Display von der aktuellen Position aus beginned aus.

Parameter

text	Ausgabetext

Beispiel:

```
anzeige.schreibeText("Testtext");
```

2.2.3.7 void Display::schreibeText (int *num*) [inline]

Gibt die jeweilige Zahl auf dem Display von der aktuellen Position aus beginned aus.

Parameter

num	Auszugebende Ganzzahl

Beispiel:

```
anzeige.schreibeText(1500);
```

2.2.3.8 void Display::schreibeText (float *num*, int *dec* = 1) [inline]

Gibt die jeweilige Zahl auf dem Display von der aktuellen Position aus beginned aus.

Parameter

num	Auszugebende Kommazahl
dec	Anzahl der Dezimalnachkommastellen Beispiel: `anzeige.schreibeText(15.5);`

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- display/Display.h

2.3 FuellstandsSensor Klassenreferenz

Diese Klasse ermöglicht die Verwendung des Füllstandssensors. Wichtig hierbei ist, dass Pin "-" mit Masse (GND), Pin "+" mit Versorgungsspannung (+5V) und Pin "S" mit Analogeingang (A0-A5) verbunden sind.

`#include <FuellstandsSensor.h>`

Öffentliche Methoden

- FuellstandsSensor (unsigned pin)
- int missFuellstandRaw ()
- float missFuellstand ()

2.3.1 Ausführliche Beschreibung

Diese Klasse ermöglicht die Verwendung des Füllstandssensors. Wichtig hierbei ist, dass Pin "-" mit Masse (GND), Pin "+" mit Versorgungsspannung (+5V) und Pin "S" mit Analogeingang (A0-A5) verbunden sind.

2.3.2 Beschreibung der Konstruktoren und Destruktoren

2.3.2.1 FuellstandsSensor::FuellstandsSensor (unsigned *pin*) [inline], [explicit]

Konstruktor

Erstellt ein Objekt, welches die Verwendung des Füllstandssensors ermöglicht.. Hierbei muss der verwendete Pin angegeben werden, wobei nur die analogen Pins A0 bis A5 verwendet werden können.

Parameter

pin	Verwendete Pin-Nummer Beispiel: `FuellstandsSensor sensor( 5 );`

2.3.3 Dokumentation der Elementfunktionen

2.3.3.1 float FuellstandsSensor::missFuellstand ( ) [inline]

Ermittelt den aktuellen Füllstand in Prozent. Beispiel:

```
int temperatur = sensor.missFuellstand();
```

2.3.3.2 int FuellstandsSensor::missFuellstandRaw ( ) [inline]

Ermittelt den aktuellen Füllstand als analogen Wert. Hinweis: Analoge Messwerte 1 160 25 250 50 260 75 270 100 300 Beispiel:

```
int temperatur = sensor.missFuellstandRaw();
```

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- fuellstandssensor/FuellstandsSensor.h

2.4 HelligkeitsSensor Klassenreferenz

Diese Klasse ermöglicht die Verwendung des Helligkeitssensors. Wichtig hierbei ist, dass ein Pin A "+" mit Versorgungsspannung (+5V) und der andere Pin B mit einem Analogeingang (A0-A5) verbunden sind. Zusätzlich muss Pin B mit einem 10K Ohm Widerstand mit Masse (-) verbunden sein.

`#include <HelligkeitsSensor.h>`

Öffentliche Methoden

- HelligkeitsSensor (unsigned pin)
- int missHelligkeitRaw ()
- float missHelligkeit ()

2.4.1 Ausführliche Beschreibung

Diese Klasse ermöglicht die Verwendung des Helligkeitssensors. Wichtig hierbei ist, dass ein Pin A "+" mit Versorgungsspannung (+5V) und der andere Pin B mit einem Analogeingang (A0-A5) verbunden sind. Zusätzlich muss Pin B mit einem 10K Ohm Widerstand mit Masse (-) verbunden sein.

2.4.2 Beschreibung der Konstruktoren und Destruktoren

2.4.2.1 HelligkeitsSensor::HelligkeitsSensor (unsigned *pin*) [inline], [explicit]

Konstruktor

Erstellt ein Objekt, welches die Verwendung des Helligkeitssensor ermöglicht. Hierbei muss der verwendete Pin angegeben werden, wobei nur die analogen Pins A0 bis A5 verwendet werden können.

Parameter

pin	Verwendete Pin-Nummer Beispiel: `HelligkeitsSensor sensor( A3 );`

2.4.3 Dokumentation der Elementfunktionen

2.4.3.1 float HelligkeitsSensor::missHelligkeit ( ) [inline]

Ermittelt die aktuelle Helligkeit in dunkel - entspricht 0- und hell -entspricht 100- an. Beispiel:

```
int helligkeit = sensor.missHelligkeit();
```

2.4.3.2 int HelligkeitsSensor::missHelligkeitRaw ( ) [inline]

Ermittelt die aktuelle Helligkeit als analogen Wert. Hinweis: Analoge Messwerte 0 bis 1023 Beispiel:

```
int helligkeit = sensor.missHelligkeitRaw();
```

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- helligkeitssensor/HelligkeitsSensor.h

2.5 Kabel Klassenreferenz

Diese Klasse ermöglicht die Verwendung von Kabeln. Außerdem ist es notwendig, dass eine Seite des Kabels mit Masse(GND) verbunden ist.

`#include <Kabel.h>`

Öffentliche Methoden

- Kabel (unsigned pin)
- bool schaltkreisGeschlossen ()
- bool schaltkreisGeoeffnet ()

2.5.1 Ausführliche Beschreibung

Diese Klasse ermöglicht die Verwendung von Kabeln. Außerdem ist es notwendig, dass eine Seite des Kabels mit Masse(GND) verbunden ist.

2.5.2 Beschreibung der Konstruktoren und Destruktoren

2.5.2.1 Kabel::Kabel (unsigned *pin*) [inline], [explicit]

Konstruktor

Erstellt ein Objekt, welches die Ansteuerung einer einzelnen KABEL ermöglicht. Hierbei muss der verwendete Pin angegeben werden.

Parameter

pin	Verwendete Pin-Nummer Beispiel: `Kabel bruecke( 5 );`

2.5.3 Dokumentation der Elementfunktionen

2.5.3.1 bool Kabel::schaltkreisGeoeffnet ( ) [inline]

Gibt den Zustand des Kabels zurück, ob Schaltkreis geöffnet ist. Beispiel:

```
if( bruecke.schaltkreisGeoeffnet() ){
//
}
```

2.5.3.2 bool Kabel::schaltkreisGeschlossen ( ) [inline]

Gibt den Zustand des Kabels zurück, ob der Schaltkreis geschlossen ist. Beispiel:

```
if( bruecke.schaltkreisGeschlossen() ){
//
}
```

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- kabel/Kabel.h

2.6 Lautsprecher Klassenreferenz

Diese Klasse ermöglicht die Verwendung des Lautsprechers.

`#include <Lautsprecher.h>`

Öffentliche Methoden

- Lautsprecher (unsigned pin)
- void spieleTon (int frequenz, int dauer=1000)

2.6.1 Ausführliche Beschreibung

Diese Klasse ermöglicht die Verwendung des Lautsprechers.

2.6.2 Beschreibung der Konstruktoren und Destruktoren

2.6.2.1 Lautsprecher::Lautsprecher (unsigned *pin*) [inline],[explicit]

Konstruktor

Erstellt ein Objekt, welches zur Ansteuerung des Lautsprechers dient. Beispiel:

```
Lautsprecher lautsprecher;
```

2.6.3 Dokumentation der Elementfunktionen

2.6.3.1 void Lautsprecher::spieleTon (int *frequenz*, int *dauer* = 1000) [inline]

Spielt einen Ton über den Lautsprecher. Beispiel:

Parameter

frequenz	Frequenzhöhe des Tons
dauer	Länge des Tons in Millisekunden (optional) `lautsprecher.spieleTon( NOTE_D8, 1000);`

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- lautsprecher/Lautsprecher.h

2.7 LED Klassenreferenz

Diese Klasse ermöglicht die Verwendung von LEDs. Wichtig hierbei ist, dass ein 220 Ohm Widerstand in Reihe zu schalten ist. Außerdem ist es notwendig, dass das zweite Bein der LED mit Masse(GND) verbunden ist.

```
#include <LED.h>
```

Öffentliche Methoden

- LED (unsigned pin)
- void einschalten ()
- void ausschalten ()

2.7.1 Ausführliche Beschreibung

Diese Klasse ermöglicht die Verwendung von LEDs. Wichtig hierbei ist, dass ein 220 Ohm Widerstand in Reihe zu schalten ist. Außerdem ist es notwendig, dass das zweite Bein der LED mit Masse(GND) verbunden ist.

2.7.2 Beschreibung der Konstruktoren und Destruktoren

2.7.2.1 LED::LED (unsigned *pin*) [inline], [explicit]

Konstruktor

Erstellt ein Objekt, welches die Ansteuerung einer einzelnen LED ermöglicht. Hierbei muss der verwendete Pin angegeben werden.

Parameter

pin	Verwendete Pin-Nummer Beispiel: `LED grueneLED( 5 );`

2.7.3 Dokumentation der Elementfunktionen

2.7.3.1 void LED::ausschalten ( ) [inline]

Schaltet die LED aus. Beispiel:

```
grueneLED.ausschalten();
```

2.7.3.2 void LED::einschalten ( ) [inline]

Schaltet die LED ein. Beispiel:

```
grueneLED.einschalten();
```

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- led/LED.h

2.8 Taster Klassenreferenz

Diese Klasse ermöglicht die Verwendung von Tastern. Wichtig hierbei ist, dass das Schalten gegen Masse (GND) erfolgt.

```
#include <Taster.h>
```

Öffentliche Methoden

- Taster (unsigned pin)
- bool wurdeGedrueckt ()
- void loescheMarkierung ()
- void setzeAktion (void(∗funktion)(void))

2.8.1 Ausführliche Beschreibung

Diese Klasse ermöglicht die Verwendung von Tastern. Wichtig hierbei ist, dass das Schalten gegen Masse (GND) erfolgt.

2.8.2 Beschreibung der Konstruktoren und Destruktoren

2.8.2.1 Taster::Taster (unsigned *pin*) [inline],[explicit]

Konstruktor

Erstellt ein Objekt, welches die Verwendung eines Tasters ermöglicht. Hierbei muss der verwendete Pin angegeben werden.

Parameter

pin	Verwendete Pin-Nummer Beispiel: `Taster meinTaster( 5 );`

2.8.3 Dokumentation der Elementfunktionen

2.8.3.1 void Taster::loescheMarkierung ( ) [inline]

Entfernt die Markierung, dass der Taster gedrückt wurde.

```
meinTaster.loescheMarkierung()
```

2.8.3.2 void Taster::setzeAktion (void(*)(void) *funktion*) [inline]

Setz eine Funktion, die bei Druck des Tasters ausgeführt wird. Beispiel:

```
void meineAktion(){
 // Mache irgendwas
}
meinTaster.setzeAktion( meineAktion );
```

2.8.3.3 bool Taster::wurdeGedrueckt ( ) [inline]

Überprüft, ob der Taster gedrückt wurde.

Rückgabe

wahr falls gedrückt, sonst falsch

```
if( meinTaster.wurdeGedrueckt() ){
// Mache irgendwas
}
```

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- taster/Taster.h

2.9 TemperaturSensor Klassenreferenz

Diese Klasse ermöglicht die Verwendung des Temperatursensors. Wichtig hierbei ist, dass Pin-1 mit Masse (GND), Pin-2 mit Analogeingang (A0-A5) und Pin-3 mit Versorgungsspannung (+5V) verbunden sind. (Plane Schnittfläche des Halbkreis zeigt nach unten: Pin-1 rechts, Pin-2 mittig, Pin-3 links !)

```
#include <TemperaturSensor.h>
```

Öffentliche Methoden

- TemperaturSensor (unsigned pin)
- float missTemperatur ()

2.9.1 Ausführliche Beschreibung

Diese Klasse ermöglicht die Verwendung des Temperatursensors. Wichtig hierbei ist, dass Pin-1 mit Masse (GND), Pin-2 mit Analogeingang (A0-A5) und Pin-3 mit Versorgungsspannung (+5V) verbunden sind. (Plane Schnittfläche des Halbkreis zeigt nach unten: Pin-1 rechts, Pin-2 mittig, Pin-3 links !)

2.9.2 Beschreibung der Konstruktoren und Destruktoren

2.9.2.1 TemperaturSensor::TemperaturSensor (unsigned *pin*) [inline],[explicit]

Konstruktor

Erstellt ein Objekt, welches die Verwendung des Temperatursensors ermöglicht. Hierbei muss der verwendete Pin angegeben werden, wobei nur die analogen Pins A0 bis A5 verwendet werden können.

Parameter

pin	Verwendete Pin-Nummer Beispiel: `y TemperaturSensor sensor( 5 );`

2.9.3 Dokumentation der Elementfunktionen

2.9.3.1 float TemperaturSensor::missTemperatur ( ) [inline]

Ermittelt die aktuelle Temperatur in °C. Hierbei muss der Datentyp 'float' einer Nicht-Ganzzahl (!) verwendet werden. Beispiel:

```
float temperatur = sensor.missTemperatur();
```

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- temperatursensor/TemperaturSensor.h

Index