Arduino Cheatsheet Deutsch Cheat Sheet

by 4rtech via cheatography.com/32532/cs/10027/

Struktur

void setup()	wird einmal ausgeführt
void loop()	wird ewig ausgeführt

Kontrolle

if (x<5) { *Code* }	*Code* wird ausgeführt wenn WAHR
else { *Code* }	*Code* wird ausgeführt wenn **if** FALSCH
for (int i = 0; i < 255; i++) { *Code* }	**i** ist 0, *Code* wird ausgeführt, **i** wird um 1 erhöht. Solange wie **i** kleiner 255.
while (x < 6) { *Code* }	*Code* wird solange ausgeführt wie x kleiner 6. **x** muss im *Code* geändert werden.

Weitere Syntax

// Kommentar...	einzeiliger Kommentar
/ Kommentar... */*	mehrzeiliger Kommentar
#define PIN 13	definiere Konstante
#include <library.h>	Library einfügen

Operatoren

=	definieren
+, -	Addition, Subtraktion
*, /	Multiplikation, Division
%	Modulo / Rest
==	ist gleich?
!=	ist nicht gleich?
<	ist kleiner
>	größer
<=	kleiner oder gleich
>=	größer oder gleich

Gemischte Zuweisung

x ++	Vergrößerung von x um 1
x --	Verkleinerung von x um 1
x += y	Vergrößerung von x um y
x -= y	Verkleinerung von x um y
x *= y	Multiplikation von x mit y
x /= y	Division von x durch y

Konstanten

true, **false**

HIGH, **LOW**

INPUT, **OUTPUT**

Datentypen

void	
boolean	0, 1, true oder false
char	z.B. 'a' oder -128 bis 127

Datentypen (cont)

unsigned char	0 bis 255
int	-32.768 bis 32.767
unsigned int	0 bis 65.535
long	-2.147.483.648 bis 2.147.483.647
float	$-3{,}4028235*10^{38}$ bis $3{,}4028235*10^{38}$

Arrays

int meineInts[6];	erzeugt *int* Array mit 6 Stellen
int meinePins[] = 2,4,6,8,10;	erzeugt und füllt Array mit 5 Stellen
int meineWerte[6] = 2,-4,9,3;	erzeugt Array mit 6 Stellen und füllt 5 davon

Strings

char S1[15];	erzeugt String mit 15 Stellen
char S2[10] = 'A','r','d','u','i','n','o';	erzeugt String mit 10 Stelle und füllt die ersten 7
char S3[] = "Arduino";	erzeugt String und füllt ihn mit "Arduino"
char S4[15] = "Arduino";	erzeugt String mit 15 STelle und füllt die ersten 7 mit "Arduino"

Umwandlung

char() int() long()

byte() word() float()

Bedingungen

static	erzeugt Variable die zwischen Funktionsaufrufen nicht gelöscht wird
volatile	erzeugt Variable die von Interrupts verändert werden kann
const	erzeugt unveränderbare Variable

Digital I/O

pinMode(*Pin*, *mode*)	ändert *Pin* zu INPUT, OUTPUT oder INPUT_PULLUP
digitalWrite(*Pin*, *Wert*)	schaltet Output ein (1/HIGH) oder aus (0/LOW)
int **digitalRead**(*Pin*)	liest ob *Pin* ein oder ausgeschaltet ist (0/1)

Analog I/O

int **analogRead**(*analogPin*)	ließt Spannung auf *analogPin* als Wert zwischen 0 und 1023
analogReference(mode)	legt Referenzspannung für *HIGH* fest
DEFAULT, INTERNAL, EXTERNAL	Spannungsreferenz-modi
analogWrite(*Wert*)	erzeugt PWM-Welle mit einem Wert zwischen 0 und 255

Interrupts

digitalPinToInterrupt(*Pin***)**	wandelt digitalen *Input-Pin* in Interrupt-Pin um
LOW, **CHANGE**, **RISING**, **FALLING**	Interrupt-Typen
attachInterrupt(*Interrupt, Function, Typ***)**	verbindet *Interrupt-Pin* mit *Funktion*
detachInterrupt(*interrupt***)**	hebt Verbindung zwischen *Funktion* und *Interrupt-Pin* auf
interrupts()	aktiviert Interrupts
noInterrupts()	deaktiviert Interrupts

Fortgeschrittene I/O

tone(*Pin, Frequenz***)**	*Frequenz* in Hz wird auf *Pin* ausgegeben
tone(*Pin, Frequenz, Dauer***)**	auf *Pin* wird eine *Frequenz* für *Dauer* Millisekunden ausgegeben
noTone(*Pin***)**	Tonausgabe auf *Pin* wird beendet
pulseIn(*Pin, Wert***)**	misst die Dauer die *Pin* *Wert annimmt (HIGH/LOW)

Zeit

unsigned int **millis()**	Millisekunden seit Programmstart. ~50 Tage Overflow

Zeit (cont)

unsigned int **micros()**	Microsekunden seit Programmstart. ~70 min bis Overflow
delay(*ms***)**	Pause für *ms* Millisekunden
delayMicroseconds(*us***)**	Pause für *us* Microsekunden

Mathematik

min(x, y), **max**(x, y), **abs**(x)	Minimum, Maximum, absoluter Wert
pow(*Basis, Exponent***)**	Basis^{Exponent}
sqrt(x)	Wurzel(x)
sin(x), **cos**(y), **tan**(z)	Sinus, Cosinus, Tangens
map(*Wert, lim1U,* lim1O*,* lim2U*,* lim2O*)**	skaliert *Wert* zwischen *lim1U* und *lim1O* zu Wert zwischen *lim2U* und *lim2O*
constrain(*Wert, limU, limO***)**	beschränkt *Wert* zwischen *unterem* und *oberem Limit*

Zufallszahlen

randomSeed(*Wert***)**	Ausgangswert für Zufallsgenerator
random(-*max***)**	Zufallszahl zwischen 0 und *max*
random(-*min, max***)**	Zufallszahl zwischen *min* und *max*

Serial

Serial.**begin**(9600);	startet **Serial**-verbindung mit Baudrate
Serial.**println(***inhalt***)**;	sendet Zeile mit *Inhalt*
Serial.**print(***text***)**;	sendet *Text*
Serial.**write(***daten***)**;	sendet *Daten* als Binärcode
Serial.**flush()**;	wartet bis *Daten* gesendet sind
Serial.**end()**;	beendet Serial-Verindung

Servo

#include <Servo.h>	fügt Servo-Library hinzu
attach(*pin***)**	fügt *Servopin* hinzu
write(*Winkel***)**	stellt Servomotor auf *Winkel*(0-180°)
read()	liest Winkel von Servomotor