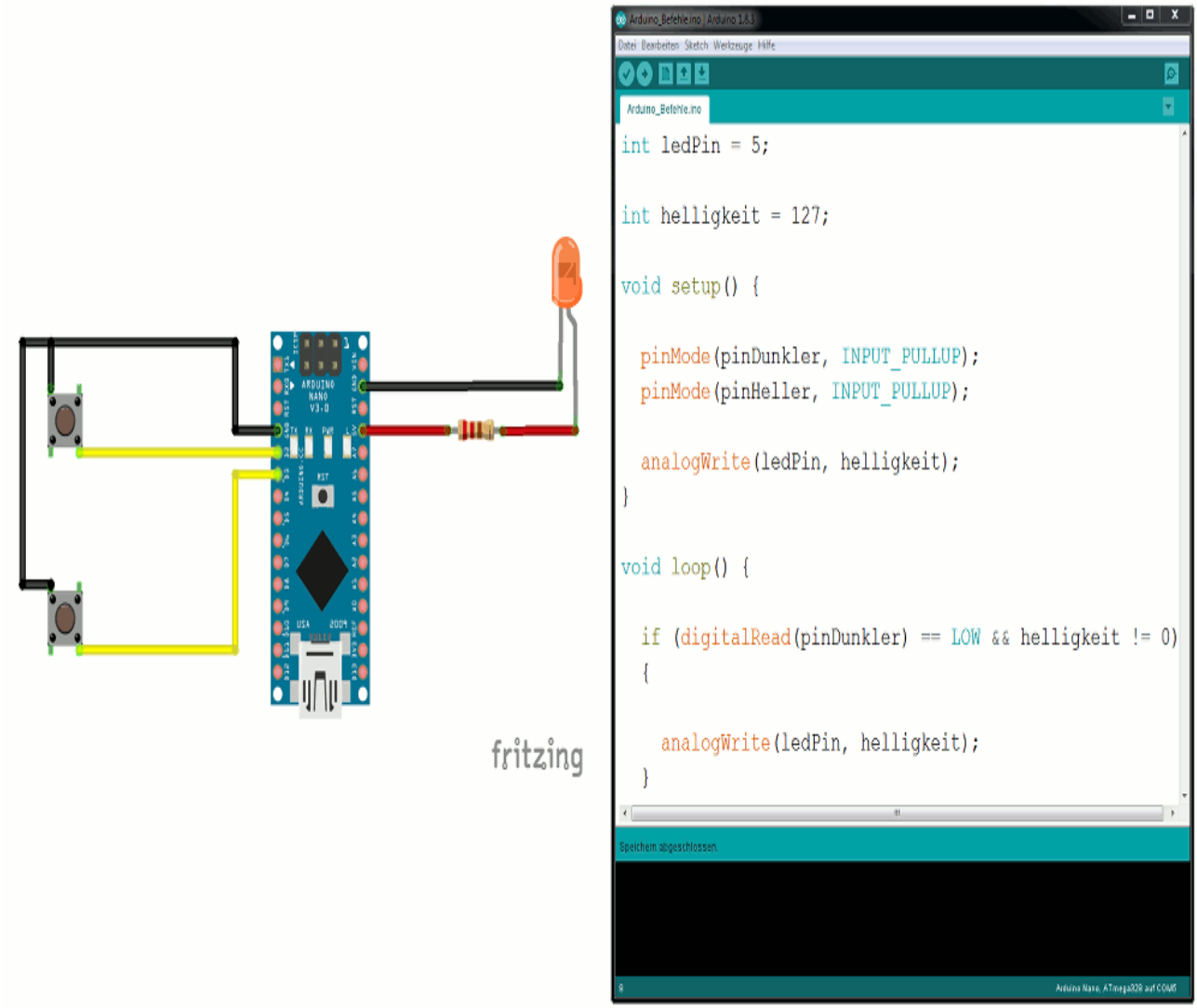
Arduino Befehle – Liste mit Erklärung auf Deutsch

13. Februar 2018Arduino, Informationen



In der Liste für Arduino Befehle findet Ihr die gängigsten Anweisungen und eine kurze Erklärung der selbigen. Diese Übersicht ist sicherlich weder vollständig noch erhebt sie einen Anspruch alle existierenden Aspekte komplett darzustellen. Allerdings findet ihr im Folgendem einen guten Überblick über die wichtigsten Arduino Befehle.

Arduino Anweisungen lassen sich in folgenden Kategorien unterteilen: Funktionen, Datentypen und Operatoren. Dazu kommen noch Schleifen, Verzweigungen und Klassen (wobei letztere nur bei der Verwendung von Bibliotheken eine ernsthaste Rolle spielen.

Funktionen

Als Funktionen werden diejenigen Arduinobefehle bezeichnet, die „Dinge tun“. Zusätzlich zu den Standartfunktionen können eigene Funktionen erzeugt werden. Dies zu beschreiben, würde allerdings über diesen Beitrag hinausgehen.

setup()

In der Setup()-Funktion werden alle Anweisungen nur einmal beim Start des Arduinos ausgeführt. Setup() muss in jedem Sketch vorhanden sein.

```
setup() {}
```

```
1  setup() {}
```

loop()

In der Loop()-Funktion werden alle Anweisungen wiederholt ausgeführt. Die Loop()-Funktion muss in jedem Sketch vorhanden sein.

```
loop() {}
```

```
1  loop() {}
```

pinMode(pin, modus)

Mit dem Arduino Befehl pinMode() kann festgelegt werden in welchen Modus ein digitaler Pin betrieben werden soll. Mit dem Argument pin wird die Pinnummer übergeben. Das Argument modus kann folgenden Zustände haben:

1. “INPUT” : Der Pin kann ein digitales Signales messen (Explizit ohne internen Pullup Widerstand).
2. “OUTPUT” : An dem Pin kann eine Spannung angelegt werden.
3. “INPUT_PULLUP” : Wie 1., allerdings wird bei diesem Argument der internet Pullup Widerstand des Eingangs aktiviert.

```
pinMode( 1, INPUT); // pin 1 ist ein Eingang pinMode(12, OUTPUT); // pin 12 ist ein Ausgang
pinMode(4, INPUT_PULLUP); //pin 4 ist ein Eingang mit internem Pullup
```

```
   pinMode( 1, INPUT); // pin 1 ist ein Eingang
1  pinMode(12, OUTPUT); // pin 12 ist ein
2  Ausgang
3  pinMode(4, INPUT_PULLUP); //pin 4 ist
   ein Eingang mit internem Pullup
```

digitalWrite(pin, zustand)

DigitalWrite() die Spannung an dem Pin pin zu manipulieren. Diese kann zwei Zustände haben. Wenn zustand den Wert “HIGH” hat wird die Pinspannung auf (i.d.R.) auf 5V gesetzt. Mit dem Wert “LOW” wird die Spannung auf (i.d.R.) 0V gesetzt.

```
digitalWrite(2, HIGH); digitalWrite(2, LOW);
```

```
1  digitalWrite(2, HIGH);
2  digitalWrite(2, LOW);
```

digitalRead(pin)

Die Funktion digitalRead() hat die Aufgabe zu bestimmen, ob die Spannung an dem dem Pin pin den Wert “HIGH” oder “LOW”. Ist die Spannung “LOW” gibt sie den Wert “0” zurück, ist sie “HIGH” wird eine “1” zurück gegeben.

int Variable = digitalRead(3);

```
1  int Variable = digitalRead(3);
```

analogRead(pin)

AnalogRead() gibt einen Wert zwischen “0” und “1023” in Abhängigkeit der Spannung, die an einem Analogpin pin anliegt zurück. Wenn aus diesem Wert eine Spannung berechnet werden soll, geschieht das folgendermaßen: Spannung = Rückgabewert/1023 * Vcc.

int Variable = analogRead(a0); U = (Variable/1023) * 5; // In U ist die aktuelle Spannung an Pin a0 abgelegt.

```
   int Variable = analogRead(a0);
1  U = (Variable/1023) * 5; // In
2  U ist die aktuelle Spannung an
   Pin a0 abgelegt.
```

analogWrite(pin , pwmWert)

Der Arduino Befehl analogWrite() ermöglicht es an einem PWM-Pin pin eine modulierte Spannung auszugeben. An das Argument pwmPin können Werte “0” bis “255” übergeben werden.

digitalWrite(3, 128); // an den Pin 3 wird eine Spannung von 127/255 * Vcc ausgegeben.

```
1  digitalWrite(3, 128); // an den Pin 3 wird eine Spannung von 127/255 * Vcc
   ausgegeben.
```

delay(zeit)

Die Delay()-Anweisung lässt den Arduino warten. Das Argument zeit übergibt, die zu wartene Zeit in Millisekunden.

delay(1500); //Der Arduino wartet für 1,5 Sekunden

```
1  delay(1500); //Der Arduino wartet für 1,5 Sekunden
```

millis()

Der Arduino Befehl Millis() gibt die seit Start des Arduinos vergangene Zeit in Millisekunden zurück. Wenn der Arduino irgendetwas alle 100 Millisekunden tun soll, ist es besser die Funktion millis() als delay(zeit) zu benutzen.

long variable = millis();

```
1  long variable = millis();
```

Erklärung der Datentypen in Arduino Befehlen

Mit der folgenden Arduino Befehls Erklärung zum Thema Datentypen, möchte ich einen kurzen Überblick über die gängisten Varianten geben.

Byte (byte)

Der Datentyp Byte speichert eine Abfolge von acht Bit, die als Zahl zwischen 0 und 255 zugewiesen wird. Er wird zum Beispiel benötigt um Daten im Eeprom zu speichern.

byte variable = 128;

```
1 byte variable = 128;
```

Boolean (bool)

Boolean speichert ein “true” oder “false”. Diese können alternative auch als 1 oder 0 angegeben werden.

bool variable = true; bool variable2 = 1; // in beiden Fällen wird der Wert "true" übergeben

```
bool variable = true;
1 bool variable2 = 1; //
2 in beiden Fällen
  wird der Wert "true"
  übergeben
```

Integer (int)

Ein Integer ist eine 16 bit lange binäre Zahl. Im dezimalen Zahlensystem kann eine Zahl zwischen -32768 und 32768 übergeben werden, da ein Bit als Vorzeichen dient. Alternativ kann der Integer als “unsigned” deklariert werden. In diesem Fall stehen die vollen 16 Bit zur Verfügung (0 – 65536).

int zahl = -500; unsigned int zahl1 = 50000;

```
1 int zahl = -500;
2 unsigned int
  zahl1 = 50000;
```

long (long)

Der Long Datentyp ist eine Art erweiterter Integer, aber er hat eine Länge von 32 Bit. Er kann signed Werte zwischen ca -2.1 und 2,1 Milliarden aufnehmen. Sollte er als unsigned deklariert sein umfasst er Werte zwischen 0 und ca 4,2 Milliearden.

long zahl = -1000000; unsigned long zahl1 = 4000000000;

```
1 long zahl = -1000000;
2 unsigned long zahl1 =
  4000000000;
```

float (float)

Der Arduino Datentyp float stellt Kommerzahlen dar. Er ist ebenfalls 32 Bit lang, hat allerdings nur eine Genauigkeit von 7 bis acht Stellen. Die Restlichen Stellen dienen zur Darstellung der Zehnerpotenz. Laut Arduino Reference kann er Zahlen zwischen -3.4028235*10^38 und -3.4028235*10^38 abspeichern.

float zahl = 0.557;

```
1  float zahl = 0.557;
```

double (double)

Double hat bei den Arduinos Uno und Nano keinen unterschied zu dem Datentyp float.

double zahl = 0.557;

```
1  double zahl = 0.557;
```

Character (char)

Ein Character ist ein Buchstabe, der in Ascii kodiert ist. Eine Googlesuche nach dem Stichwort “Ascii Tabelle” fördert diese schnell zutage.

char buchstabe = 'a';

```
1  char buchstabe = 'a';
```

String (char[])

Einen Datentüt String gibt es in der Arduinowelt nicht. Strings werden als ein Array aus Character angesehen, die null terminiert sind. Die Arduino Befehle zur deklaration:

char wort[] = "hallo"; char wort1[] = {'h' , 'a' , 'l' , 'l' , 'o' , '\0'};

```
1  char wort[] = "hallo";
2  char wort1[] = {'h' ,
   'a' , 'l' , 'l' , 'o' , '\0'};
```