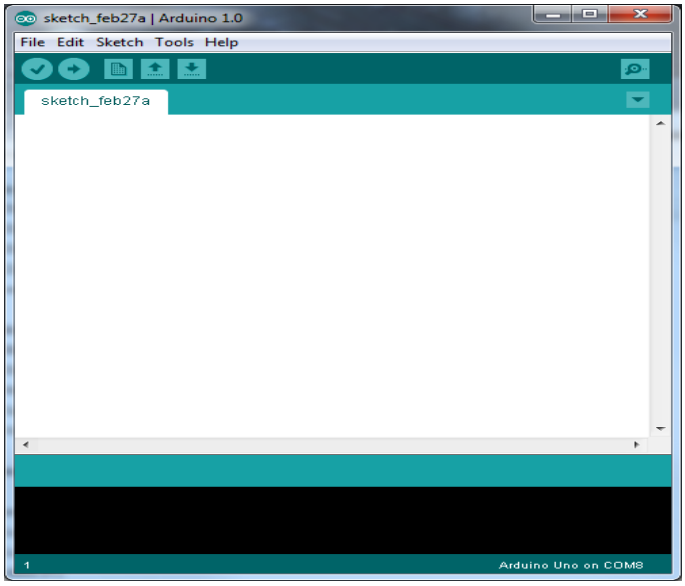
InfoSphere
World of Informatics

Merkblatt 2 – Wie wird der Arduino programmiert?

1 Übersicht

Für die Programmierung steht ein Programm zur Verfügung. Hier kann der Quelltext geschrieben, überprüft, kompiliert und anschließend auf den Arduino geladen werden. Wenn ihr das Programm startet, erscheint folgendes Fenster.

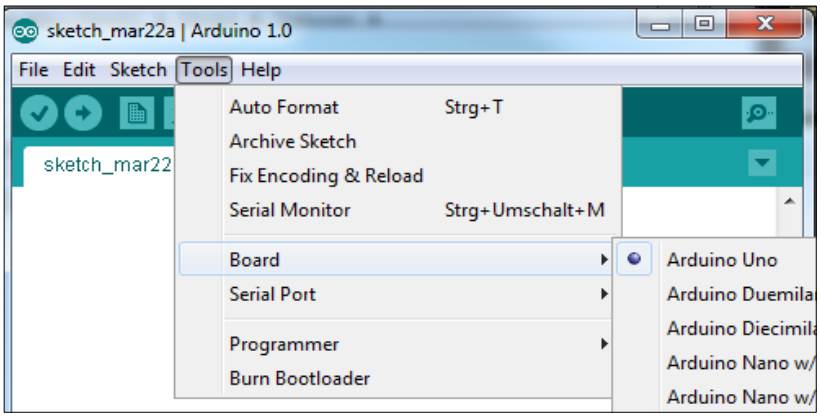


Mit dieser Schaltfläche wird das Programm auf den Arduino geladen.

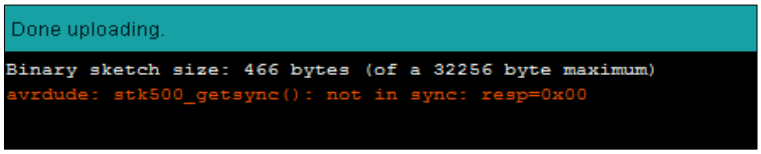
Mit dieser Schaltfläche kann der Quelltext überprüft werden. Enthält er Fehler, werden im unteren schwarzen Feld entsprechende Meldungen ausgegeben.

Im Hauptfenster wird der Quelltext geschrieben. Vor dem Laden auf das Arduino-Board müssen folgende Einstellungen überprüft werden:

- Ist das richtige Board ausgewählt? Dafür schaut ihr unter dem Menüpunkt *Tools -> Board* nach. Hier muss "Arduino Und' ausgewählt sein.



- Es muss die richtige serielle Schnittstelle ausgewählt sein. Hierfür schaut ihr unter dem Menüpunkt *Tools -> Serial Port* nach. Es stehen möglicher-weise mehrere zur Verfügung (COM1, COM2, etc.). Wählt zunächst die höchste Nummer. Sollte nach dem Hochladen das schwarze Feld wie rechts aussehen, so habt ihr die falsche Schnittstelle ausgewählt, nehmt eine andere.



2 Die Grundlagen

2.1 Ein Arduino Sketch

Ein Programm, welches für den Arduino geschrieben wird, wird auch Sketch genannt und besteht in unserem Fall aus zwei Methoden:

```
void setup () {
}

void loop () {
}
```

Das Grundgerüst eines Arduino-Sketch

Die **Setup-Methode** wird zu Anfang des Programms *genau einmal ausgeführt*. Hier können Variablen mit Startwerten versehen werden. Ein weiterer wichtiger Nutzen liegt in der Definition von Ein- und Ausgängen, da digitale Pins sowohl Ein- als auch Ausgänge sein können. Dies **muss** hier spezifiziert werden. Dafür dient eine Methode namens `"pinMode()"`. Diese erwartet zwei Argumente: Die Nummer des Pins und die Angabe, ob es sich um einen Ausgang (OUTPUT) oder Eingang (INPUT) handeln soll. Folgender Quelltext beispielsweise definiert den Pin mit der Nummer 4 als Ausgang:

```
void setup () {
  pinMode(4,OUTPUT);
}
```

Die Funktion `pinMode` zur Definition von Ein- und Ausgängen

Ganz wichtig: Achtet auf Groß- und Kleinschreibung.

Die **Loop-Methode** wird wiederholt ausgeführt. Hier gehört also alles hin, was endlos ausgeführt werden soll. Beispielsweise fängt eine Ampelschaltung immer wieder von vorne an. Man muss also nur einen" Durchlauf" programmieren und die Loop-Methode macht den Rest.

Möchte man im Programm **Variablen** verwenden, so müssen für diese erst einmal definiert werden, um was für eine Art von Variable es sich handelt. Wir verwenden nur die Zahlenvariable vom Typ Integer. Diese wird mit dem Schlüsselwort „int“ definiert. Diese Definition muss über der Setup-Methode stehen. Will man eine Integer-Variable namens Zahl definieren und ihr den Startwert 5 zuweisen, geht dies so:

```
int Zahl;

void setup(){
  Zahl = 5;
}

void loop(){
}
```

Die Definition einer Integer-Variable erfolgt mit dem Schlüsselwort `int`

2.2 Die Switch-Case-Anweisung

Switch-Case dient zum Verzweigen ähnlich wie die If-Anweisung. Mit einem if wird geprüft, ob **eine** Bedingung zutrifft, also zum Beispiel ob der Wert der Variablen *Zahl* gleich 5 ist. Falls ja wird der entsprechende Quelltext ausgeführt.

```
if (Zahl == 5)
```

Mit einer Switch-Case-Anweisung kann man eine solche Variable nicht nur auf einen Wert überprüfen, sondern auf **beliebig viele**. Das Prinzip ist einfach: Wenn die Variable *Zahl* 5 ist, dann *tue dies;* wenn der Wert 6 ist, dann *tue das*; wenn der Wert 7 ist, dann *tue jenes*, usw.

```
switch(Zahl){
  case 5: {tue dies}
  case 6: {tue das}
  case 7: {tue jenes}
}
```

Die Verzweigung mittels switch-case

Am Ende jedes Programmblocks, der ausgeführt wird, muss allerdings das Schlüsselwort **break** stehen. Damit wird signalisiert, dass die Anweisungen für diesen bestimmten Wert beendet sind.

```
int Zahl;
int Zahl2;

void setup(){
  Zahl = 5;
}

void loop(){
  switch(Zahl) {
    case 4: {Zahl2=10; break;}
    case 5: {Zahl=4; break;}
  }
}
```

Welchen Wert erhält die Variable Zahl2?

_____

_____

Die Variable Zahl2 wird nach dem zweiten Durchlauf auf den Wert 10 gesetzt.

3 Die Lampensteuerung

Im Prinzip sind die Lampen ganz einfache Elemente. Man kann sie ein- und ausschalten. Hierfür gibt es eine Funktion, die dies bewerkstelligt: `"digitalWrite()"`.

Die Funktion erwartet zwei Argumente: Erstens den Pin, an dem die zu steuernde Lampe angeschlossen ist und zweitens ob sie eingeschaltet (HIGH) oder ausgeschaltet (LOW) werden soll. Folgender Quelltext würde also eine Lampe an Pin 3 einschalten:

```
digitalWrite(3,HIGH);
```

Die Funktion `digitalWrite` zum Steuern von digitalen Ausgängen

Eine weitere Funktion, die ihr für die Lampensteuerung braucht, ist die Wartefunktion `"delay()"`. Damit kann man die Programmausführung für eine bestimmte Zeit, die in Klammern angegeben wird, anhalten. Diese Zeit wird in Millisekunden angegeben. Eine Sekunde dauert 1000 Millisekunden.

```
delay(1000);
```

Die Funktion `delay` zum Pausieren des Programms

Auf dem Arduino-Board befindet sich bereits eine fest verbaute Lampe. Auch die lässt sich über einen Pin ansteuern, nämlich über den Pin mit der Nummer 13.

Abbildung 1: Eingebaute LED

(a) Schreibt ein Arduino-Sketch, der die Lampe auf dem Board zum Leuchten bringt. Ladet das Programm anschließend auf den Arduino, um es zu testen.
(b) Erweitert nun euren Sketch, so dass die Lampe blinkt. Ladet auch das Programm anschließend auf den Arduino, um es zu testen.

4 Der Taster

Der Taster ist ein Element, welches an einem Eingang angeschlossen wird. In der Programmierung wird abgefragt, ob der Taster gedrückt ist oder nicht. Für einen an einem digitalen Eingang angeschlossenen Taster nutzt man die Funktion `"digitalRead()"`.

Die Funktion erwartet ein Argument, nämlich die Nummer des Pins an dem der Taster angeschlossen ist. Die Funktion liefert die möglichen Rückgabewerte "HIGH" (Taster ist gedrückt) und "LOW" (Taster ist nicht gedrückt).

Folgendes Beispiel speichert in der Variable "Eingangswert", ob ein Taster, der an Pin 10 angeschlossen ist, gedrückt ist der nicht.

```
int Eingangswert;

void setup(){
}

void loop(){
  Eingangswert = digitalRead(10);
}
```

Die Funktion `digitalRead` zum Auslesen eines digitalen Eingangs

Um auf einen Taster reagieren zu können, muss man an geeigneter Stelle abfragen, ob der Taster gedrückt wurde.
Schreibt ein Sketch, welches einen Taster an Pin 10 abfragt und wenn dieser gedrückt wurde eine Lampe an Pin 5 einschaltet.

_____

_____

_____

_____

_____

_____

5 Der Drucksensor

Der Drucksensor ist ebenso ein Element, welches an einem Eingang angeschlossen wird. Im Gegensatz zum Taster handelt es sich um einen analogen Eingang. Um einen solchen Eingang abzufragen nutzt man die Funktion "`analogRead()`".

Die Funktion erwartet als Argument die Nummer des Pins an dem der Sensor angeschlossen ist. Auch diese Funktion liefert einen Rückgabewert, nämlich das, was der Sensor misst. Im Fall des Drucksensors also die Stärke des ausgeübten Drucks. Diese Stärke wird in einem Zahlenwert ausgedrückt, der zwischen 0 (kein Druck) und 1023 (maximaler Druck) liegt. Dieser Zahlenwert wird von der Funktion "`analogRead`" zurückgegeben.

Folgendes Beispiel speichert den Druck des Sensors, welcher an Pin 6 angeschlossen ist, in der Variable "Eingangswert".

```
int Eingangswert;

void setup(){
}

void loop(){
  Eingangswert = analogRead(6);
}
```

Die Funktion `analogRead` zum Auslesen eines analogen Eingangs

Wenn eine Ampel über einen Drucksensor gesteuert wird, möchte man, dass sie ab einem bestimmten Druck reagiert. Dieser Druck sollte nicht zu klein, sein, denn sonst würde die Ampel bereits auf einen Vogel oder ein herunterfallendes Blatt reagieren.

Schreibt ein Sketch welches einen Drucksensor am Pin 1 abfragt, ob ein bestimmter Druck (als Zahlenwert: 611) überschritten ist, und falls ja eine Lampe an Pin 5 einschaltet.

_____

_____

_____

_____

Quellenverzeichnis:

Abb. 1 - *Quelle: InfoSphere (Marc Weintz)*

Alle Screenshots - *Quelle: Screenshot der Arduino-Software (http://www.arduino.cc)*

- *Quelle: pixabay.com, Autor: PublicDomainPictures (CC0)*