Wer blinkt so spät durch Nacht und Wind…

Es sind Leuchtdioden an den Beinchen (Pins) Nummer 10, 11, 12, 13 angeschlossen. Als ersten Schritt werden wir eine davon zum Blinken bringen: Einschalten – Warten – Ausschalten – Warten und so weiter…

```
void setup() {
  pinMode(10, OUTPUT);
}

void loop() {
  digitalWrite(10, HIGH);
  delay(100);
  digitalWrite(10, LOW);
  delay(1000);
}
```

Vorschläge zum Experimentieren

- Die Wartezeiten verändern, so dass die Leuchtdiode schneller oder langsamer oder seltener blinkt.
- Einen anderen Pin verwenden, um eine der anderen Leuchtdioden blinken zu lassen.
- Das "pinMode" und "digitalWrite" kopieren, um mehrere Leuchtdioden gleichzeitig blinken zu lassen.
- Eine Folge von digitalWrite und delay Befehlen, um ein Muster zu blinken wie z.B. SOS.
- Was passiert, wenn die Wartezeiten sehr klein sind? Welchen Einfluss hat das Verhältnis zwischen An- und Aus-Zeit?

Erklärungen

```
void setup() { … }
```

Dies definiert ein Unterprogramm namens "setup". Dieses ist speziell: Es wird einmal nach jedem Start des Mini-Computers ausgeführt. Es kann benutzt werden, um Einstellungen und Startwerte festzulegen.

```
void loop() { … }
```

Definiert das „loop“ Unterprogramm. Es wird vom Mini-Computer immer wieder neu ausgeführt bis jemand den Strom ausschaltet. Es können weitere eigene Unterprogramme definiert werden.

Die Klammern `{ … }` umschließen eine Befehlsfolge. Die meisten Befehle enden mit einem Semikolon `;` damit der Computer weiß, wo der nächste Befehl losgeht. Die Befehle werden in normaler Leserichtung abgearbeitet (links nach rechts, oben nach unten).

```
pinMode(10, OUTPUT);
```

Sagt, dass der Pin 10 als Ausgang benutzt werden soll. Dies erlaubt es, mit dem nächsten Befehl die Spannung an dem Pin festzulegen.

```
digitalWrite(10, HIGH);
digitalWrite(10, LOW);
```

HIGH verbindet den Pin 10 im Mini-Computer mit den 5V der Stromversorgung und LOW mit den 0V. Das sind die + und - Leitungen auf dem Steckbrett. Die Bezeichnung kommt daher, dass in vielen Signaldiagrammen die 5V oben und die 0V unten sind.

```
delay(1000);
```

Dieser Befehl verzögert die Ausführung unseres Programms um die gewünschte Anzahl an Milli-Sekunden. 1000ms sind eine Sekunde. Der Mini-Computer wartet hier und prüft die interne Uhr bis genügend Zeit vergangen ist. Für noch kürzere Wartezeiten gibt es folgenden Befehl:

```
delayMicroseconds(5);
```

Wartet 5 Micro-Sekunden. 1000us = 1ms. Kleinere Zeiten sind ungenau.

Ein wenig Tastsinn...

Nachdem wir nun immerhin Leuchdioden ein und ausschalten können, versuchen wir als nächstes ein Beinchen als Eingang für eine Taste zu benutzen. An Pin 5 ist eine Leuchtdiode angeschlossen, an Pin 7 eine Taste und an Pin 9 ein Lautsprecher.

```
void setup() {
  pinMode(5, OUTPUT);
  pinMode(7, INPUT_PULLUP);
}

void loop() {
  bool taste;
  taste = digitalRead(7);
  if (taste == LOW) {
    digitalWrite(5, HIGH); // einschalten
  } else {
    digitalWrite(5, LOW); // ausschalten
  }
}
```

Vorschläge zum Experimentieren

- Die Leuchtdiode soll ausgeschaltet sein, wenn die Taste gedrückt ist. Es gibt mindestens zwei Wege, dies zu erreichen.
- Nachmessen: Welche Spannung liegt am Pin an wenn die Taste gedrückt ist und welche sonst?
- Die Leuchtdiode soll eine halbe Sekunde leuchten wenn die Taste gedrückt wird und dann wieder aus gehen.
- Schalte den Lautsprecher ein wenn die Taste gedrückt ist und wieder aus, wenn die Taste nicht gedrückt ist.
- Lass den Mini-Computer SOS piepsen (**· · · – – – · · ·**) wenn die Taste gedrückt wird.

Erklärungen

```
pinMode(7, INPUT_PULLUP);
```

Stellt den Pin 7 als Eingang ein. `INPUT_PULLUP` ist ein spezieller Modus für Tasten und Schalter. Damit wird ein interner Widerstand zwischen 5V und dem Pin aktiviert, so dass man den nicht selber anschließen muss.

```
bool taste;
```

Dies definiert eine Variable namens „taste“ mit dem Inhaltstyp „bool“. Variablen sind Container in denen man sich eine Information für später merken kann. Es gibt viele Typen für verschiedene Verwendungszwecke. Der Typ `bool` kennt nur die beiden Werte `true` und `false` (ja/nein) bzw. `HIGH` und `LOW`. Der Name ist eine Erinnerung an George Boole.

```
taste = digitalRead(7);
```

Das `digitalRead` prüft die Spannung an Pin 7 und mit der Zuweisung „**=**“ wird das Ergebnis in die Variable „taste“ kopiert. Bei einer Spannung größer 3V ist das Ergebnis `true` bzw `HIGH`, ansonsten `false` bzw. `LOW`.

```
if (BEDINGUNG) { A… }
if (BEDINGUNG) { A… } else { B… }
```

Dies ist eine Verzweigung im Programmfluss. Falls die Bedingung zutrifft werden die Befehle in A ausgeführt, ansonsten die in B falls vorhanden. Im Beispielprogramm wird der Vergleich „**==**“ benutzt. Nicht mit der Zuweisung „**=**“ verwechseln.

```
tone(9, 440);
```

Dieser Befehl stellt einen Hardware-Zähler am Pin 9 so ein, dass auf dem Pin ein rechteckiges Spannungssignal mit der Frequenz 440Hz erzeugt wird. Wenn die Spannung HIGH (5V) ist, fließt ein Strom durch den Widerstand und den Lautsprecher zu der 0V Leitung. Der Strom erzeugt ein Magnetfeld und dieses bewegt die Membran des Lautsprechers durch Anziehung/Abstoßung zu dem Permanentmagneten.

```
noTone(9);
```

Und dann war plötzlich wieder Stille…

Mensch - Maschine Kommunikation

Oft ist es nützlich, mehr aus dem Inneren des Mini-Computers zu erfahren als mit einer morsenden Leuchtdiode kommunizierbar ist. Wir wollen Texte vom Mini-Computer an unseren PC übertragen und dort live sehen. Damit es interessanter wird, haben wir am Pin A0 und A1 jeweils einen Drehwiderstand als einfache Sensoren angeschlossen.

```
void setup() {
  pinMode(A0, INPUT);
  Serial.begin(57600);
  Serial.println(“Hallo Welt!“);
}

void loop() {
  int wert;
  wert = analogRead(A0);
  Serial.println(wert);
  delay(10);
}
```

In der Arduino Umgebung kann man im Menü „Werkzeuge“ den Punkt „Serieller Monitor“ anklicken. Dies liefert ein Fenster in dem die Textausgaben zu sehen sind. Mit „Serieller Plotter“ bekommt man eine grafische Darstellung der übertragenen Messwerte.

Vorschläge zum Experimentieren

- Lies auch den Messwert an Pin A1. Erweitere die Textausgabe so, dass beide Werte mit einem Leerzeichen “ “ getrennt ausgegeben werden.
- Lass eine Leuchtdiode blinken. Der Drehwiderstand an A0 kann die Periodendauer (An+Auszeit) festlegen und der Andere an A1 den Anteil der An-Zeit. Was passiert bei kurzen Perioden?

```
int periode = analogRead(A0);
int anzeit = periode*analogRead(A1)/1024;
```

- Noch mehr Krach: Die Frequenz per Drehwiderstand einstellen.

Erklärungen

```
Serial.begin(57600);
```

Richtet die Verbindung mit der angegebenen Geschwindigkeit ein. Typische Werte sind: 9600, 14400, 19200, 28800, 38400, 57600, 115200. Die Älteren unter uns, werden sich vielleicht noch an 57k6 Modems am Analogtelefon erinnern.

```
Serial.print(“Hallo Welt!“);
Serial.println(“Hallo Welt!“);
```

Sendet den Text über das USB-Kabel zum PC. Die Variante „`println`“ sendet anschließend das Steuerzeichen für eine neue Zeile (print line).

```
int wert;
```

Definiert die Variable namens „wert“ vom Typ „int“. Das steht für Ganze Zahlen (ohne Dezimalpunkt/Komma) im Bereich von -32,768 bis 32,767. Andere ähnliche Typen sind „byte“ von 0 bis 255, „char“ von -128 bis 127 und „long“ im Bereich ±2 Milliarden.

```
wert = analogRead(A0);
```

Liest die Spannung an Pin A0 und wandelt sie in eine Ganze Zahl zwischen 0 und 1023 um. Der Wert 0 entspricht 0V am Pin und 1023 entspricht 5V. Das Messergebnis wird in die Variable „wert“ kopiert.

```
Serial.print(wert);
Serial.println(wert);
```

Der print Befehl kann auch Zahlen in Text umwandeln.

```
wert = map(wert, 0, 1023, 440, 880);
```

Die `map` Funktion (auf Deutsch „abbilden“) ist nützlich, um Werte aus einem Zahlenbereich in einen anderen umzurechnen. Einfach probieren!

```
wert = constrain(wert, 10, 150);
```

Diese Funktion schränkt den Wert auf den gewünschten Zahlenbereich ein. Im Beispiel werden Zahlen kleiner 10 durch 10 ersetzt und Zahlen größer 150 durch 150 ersetzt. Nur die Zahlen dazwischen bleiben wie sie sind.

Fast wie die Fledermaus…

Wir haben einen Ultraschall Entfernungssensor. Dieser funktioniert ähnlich wie bei der Fledermaus: Mit dem „Trigger“ Ausgang (Pin 3) wird ein Ultraschall-Impuls losgeschickt und am „Echo“ Eingang (Pin 4) messen wir die Zeit bis zum ersten Echo. Diese Zeit hängt vom Abstand ab.

```
void setup() {
  pinMode(3, OUTPUT); // Trigger
  pinMode(4, INPUT);  // Echo
  Serial.begin(57600);
}

long zeit;

void loop() {
  // den Ultraschall-Impuls los schicken
  digitalWrite(3, HIGH);
  delayMicroseconds(10);
  digitalWrite(3, LOW);

  // die Zeit bis zum Echo messen (in Micro-Sekunden)
  zeit = pulseIn(4, HIGH, 4000);

  Serial.println(zeit);
  delay(10); // 10ms warten
}
```

Vorschläge zum Experimentieren

- Schaue Dir die Messwerte im Seriellen Plotter oder der Konsole an. Was ist der kleinste und größte Wert? Wie hängen Entfernung und Messwert voneinander ab?
-

Erklärungen

```
zeit = pulseIn(4, HIGH, 4000);
```

Die Funktion `pulseIn` wartet bis die Spannung am Pin (hier 4) größer 3V wird (HIGH) und misst dann die Zeit in Micro-Sekunden bis die Spannung wieder kleiner 3V ist (LOW). Die 4000 gibt an, dass maximal 4ms gewartet werden soll. Das Messergebnis wird in die Variable „zeit“ kopiert.

```
Serial.begin(57600);
Serial.println(zeit);
```

Die erste Zeile richtet die Datenübertragung zum PC ein. Dabei wird die Übertragungsgeschwindigkeit 57600 Bit pro Sekunde benutzt. Die zweite Zeile schickt den aktuellen Wert aus der Variable „zeit“ als Text an den PC. Dort kann man diese Werte in der Seriellen Konsole und dem Seriellen Plotter sehen (Menüpunkt Werkzeuge).

```
tone(6, 440);
noTone(6);
```

Der erste Befehl erzeugt einen Ton an dem Pin 6 mit der angegebenen Frequenz, hier 440Hz.