Blinken

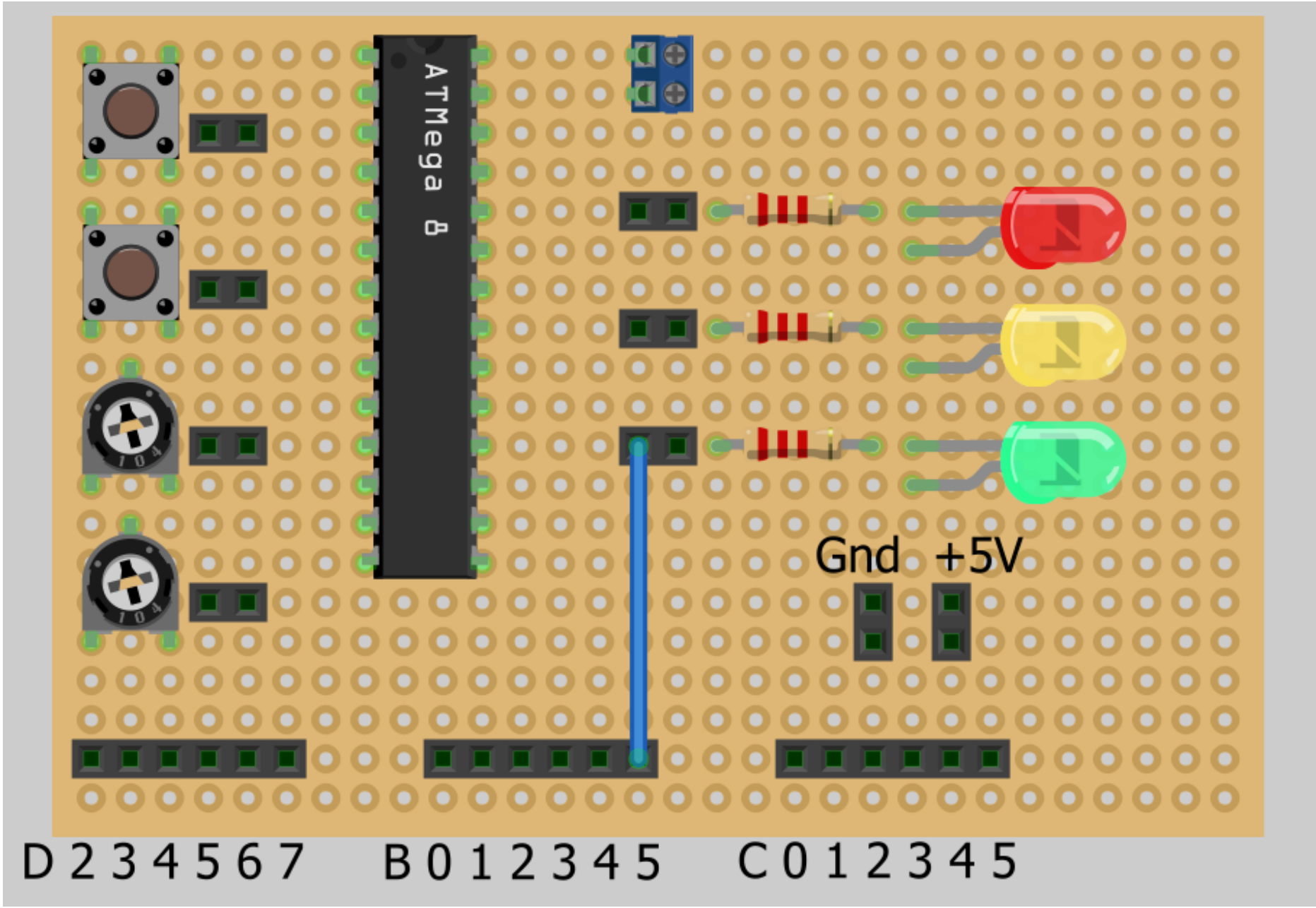
Starten Sie die Arduino IDE und geben Sie den folgenden Programmcode ein:

```
// Projekt 1 - LED Blinker

int ledPin = 13;

void setup()
{
  pinMode(ledPin, OUTPUT);
}

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

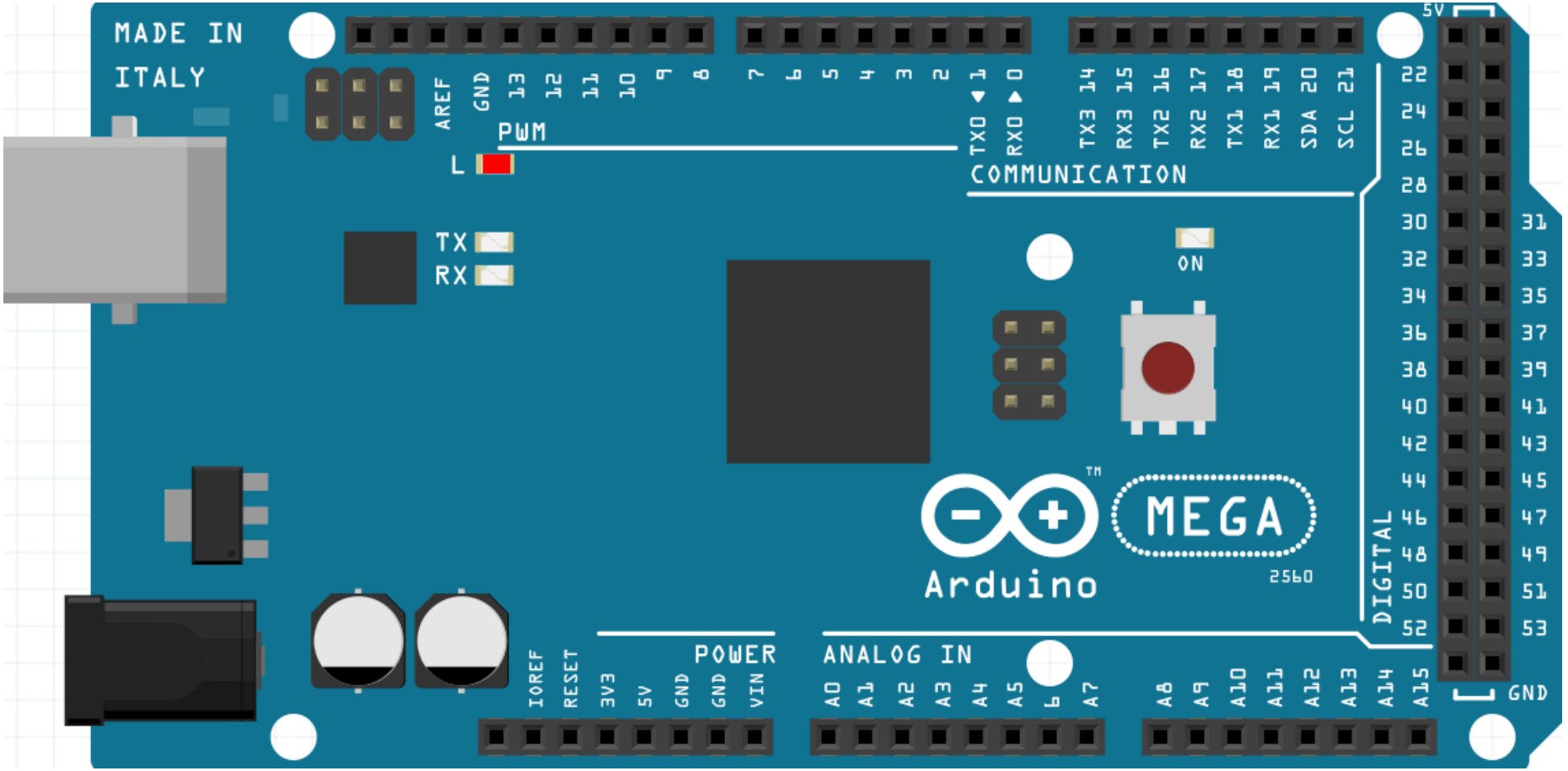
Compile **Stop** **Neu** **Öffnen** **Speichern** **Upload** **Terminal**

- Clicken Sie auf **Compile** - das Programm wird in Maschinencode übersetzt
- Clicken Sie auf **Upload** - der Maschinencode wird in den MCU übertragen

MyAvr: Verbindeden Port B5 mit einer LED
Diese sollte nach dem Hochladen des Programmes blinken.



Der Arduino hat auf dem Board eine LED, die mit Pin 13 verbunden ist. Diese sollte nach dem Hochladen des Programmes blinken.



Nun sehen wir uns den Code genauer an:

// Projekt 1 - LED Blinker

Die erste Seite enthält einen Kommentar. Mit Kommentaren kann man die Lesbarkeit
Von Programmen deutlich verbessern.

Die Zeichen **//** gelten immer für die aktuelle Zeile.

Mehrzeilige Kommentare schreibt man in der Form:

/* Text innerhalb von Schräger und Asterics
wird vom Compiler ignoriert ***/**

int ledPin = 13;

Mit dem Schlüsselwort **int** wird eine Integer Variable deklariert.

Eine Variable ist ein Speicher für beliebige Werte. Vielfach ändert sich der Wert einer Variable mehrfach im Programmablauf. Eine Integer Variable kann Werte zwischen **-32768** und **+32767** annehmen und belegt im Speicher 2 Byte.

Der Variable wurde hier der Name ledPin zugewiesen. Diese Namen können beliebig vergeben werden.

Schließlich wird der Variable der Wert 13 zugewiesen. Diese Wertzu weisung könnte auch noch später erfolgen. In diesem Fall wäre auch **int ledPin;** ausreichend.

Die Variable **ledPin** wird hier dafür verwendet, um später einen Wert am Digital - Pin 13 auszugeben. Unter ledPin kann man sich in größeren Programmen mehr vorstellen als nur unter der Zahl **13**.

Wichtig : Jede Befehlzeile wird immer mit einem ; abgeschlossen. Hierdurch weiss der Compiler, daß die Befehlzeile hier zu Ende ist.

Der ATMega8 verfügt über 3 Ports (**PORTD**, **PORTB**, **PORTC**).

Die Pin's dieser Ports können als Eingang oder Ausgang benutzt werden. Um die Programmierung mit Arduino zu vereinfachen wurden die Pins von **0 .. 13** durchnummeriert.

Da die Pins auf dem MyAvr - Board etwas anders beschriftet sind, müssen wir uns folgendes Pin – Mapping einprägen.

Der PortD wird von der LCD - Anzeige genutzt, PortC enthält die Analogeingänge. Es empfielt sich deshalb für Ein- und Ausgaben **PORTB** (**Pin 8 .. 13**) zu benutzen:

Arduni Pin	**MyAvr**
8	**Port B0**
9	**Port B1**
10	**Port B2**
11	**Port B3**
12	**Port B4**
13	**Port B5**

Pin Mapping Arduino - AVR

Digitale Ein-/Ausgänge

Arduino	AVR
D0	Port D0
D1	Port D1
D2	Port D2
D3	Port D3
D4	Port D4
D5	Port D5
D6	Port D6
D7	Port D7
D8	Port B0
D9	Port B1
D10	Port B2
D11	Port B3
D12	Port B4
D13	Port B5

Analoge Eingänge

Arduino	AVR
A0	Port C0
A1	Port C1
A2	Port C2
A3	Port C3
A4	Port C4
A5	Port C5

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

Ein Arduino Programm muss immer die beiden Funktionen **setup()** und **loop()** enthalten. **Wichtig: Die Befehle stehen innerhalb der { } Klammern innerhalb einer Funktion**

Die Funktion **setup()** wird bei jedem Programmstart **nur 1x** ausgeführt und enthält meist Deklarationen, die sich im Programm nicht mehr ändern.

Alle Befehle in der Funktion **loop()** werden ständig in einer Schleife wiederholt.

Die Pins der Ports D, B, C können sowohl als **Ausgang** als auch als **Eingang** benutzt werden.

Mit **pinMode(13, OUTPUT)** wird der Pin 13 als Ausgang konfiguriert.

Damit wir uns die Pins besser merken können, haben wir zuvor der Variable **ledPin** den Wert 13 zugewiesen.

Deshalb können wir statt **pinMode(13, OUTPUT)** auch **pinMode(ledPin, OUTPUT) benutzen.**

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

Die Funktion **loop()** ist die Main – Routine eines Arduino Programmes. Die Befehle innerhalb der **{ }** Klammern werden abgearbeitet und ständig in einer Schleife wiederholt.

Mit **digitalWrite(13, HIGH)** wird der zuvor als Ausgang konfigurierte Pin 13 auf High – Level geschaltet.

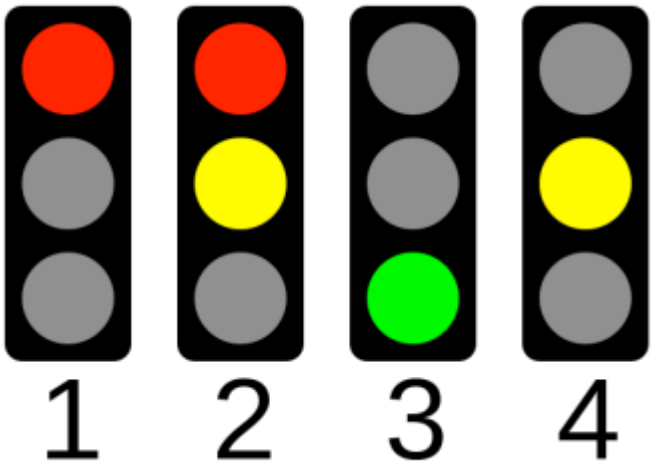
Mit **digitalWrite(13, LOW)** wird der Ausgang auf Low – Level geschaltet.

Statt **13** können wir hier auch wieder unsere Variable **ledPin** einsetzen.

Mit dem Befehl **delay(1000)** wird das Programm für 1000 ms unterbrochen. Statt 1000 können beliebige Zahlen in Milliskunden eingesetzt werden.

Das Programm schaltet die LED an Pin 13 ein, wartet 1 Sekunde, schaltet die LED wieder aus und wartet erneut 1 Sekunde. Danach wiederholt sich der Ablauf ständig und die LED blinkt im Rhythmus von 1 Sekunde.

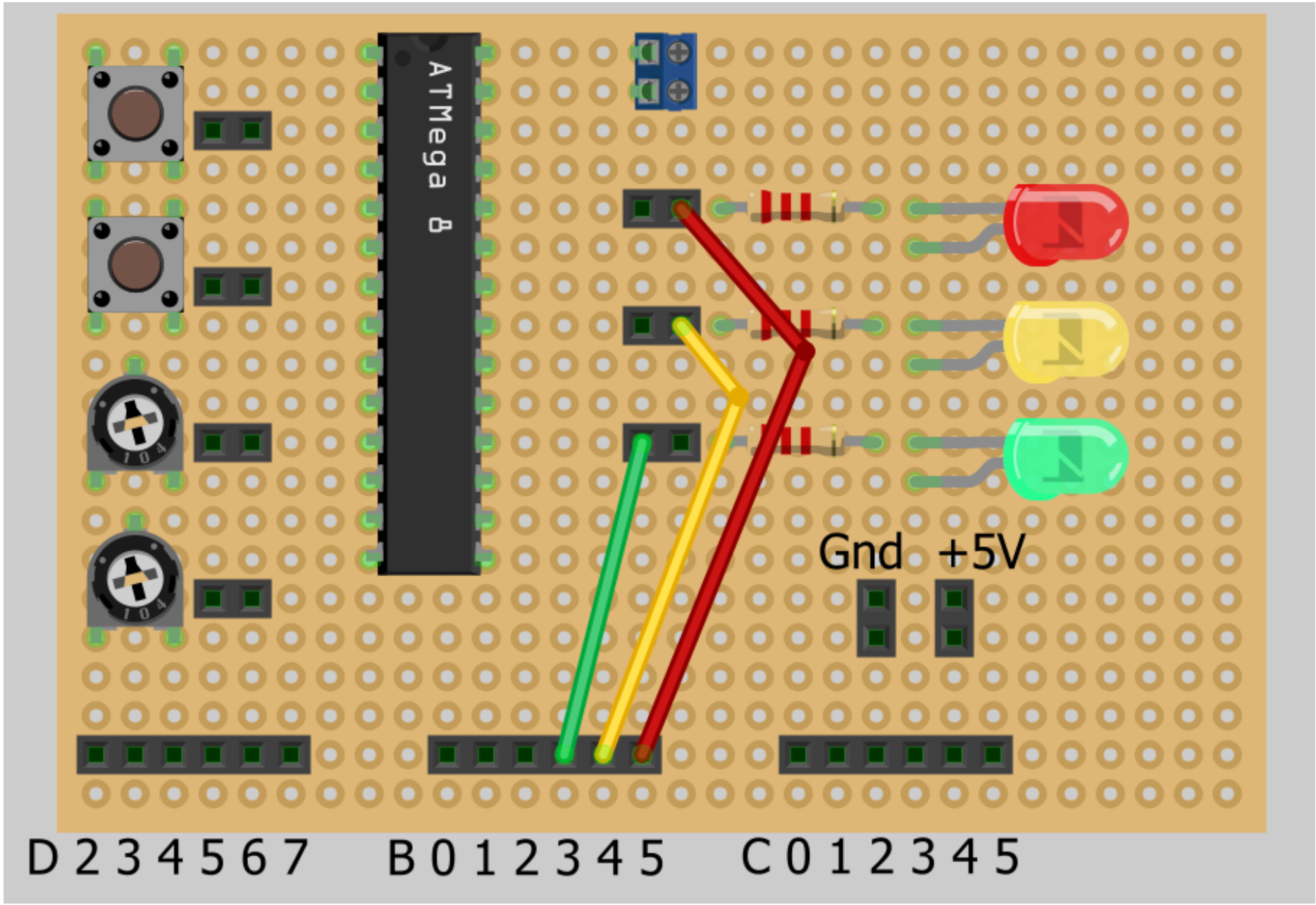
Mit den erlernten Befehlen können wir nun eine Ampel programmieren.



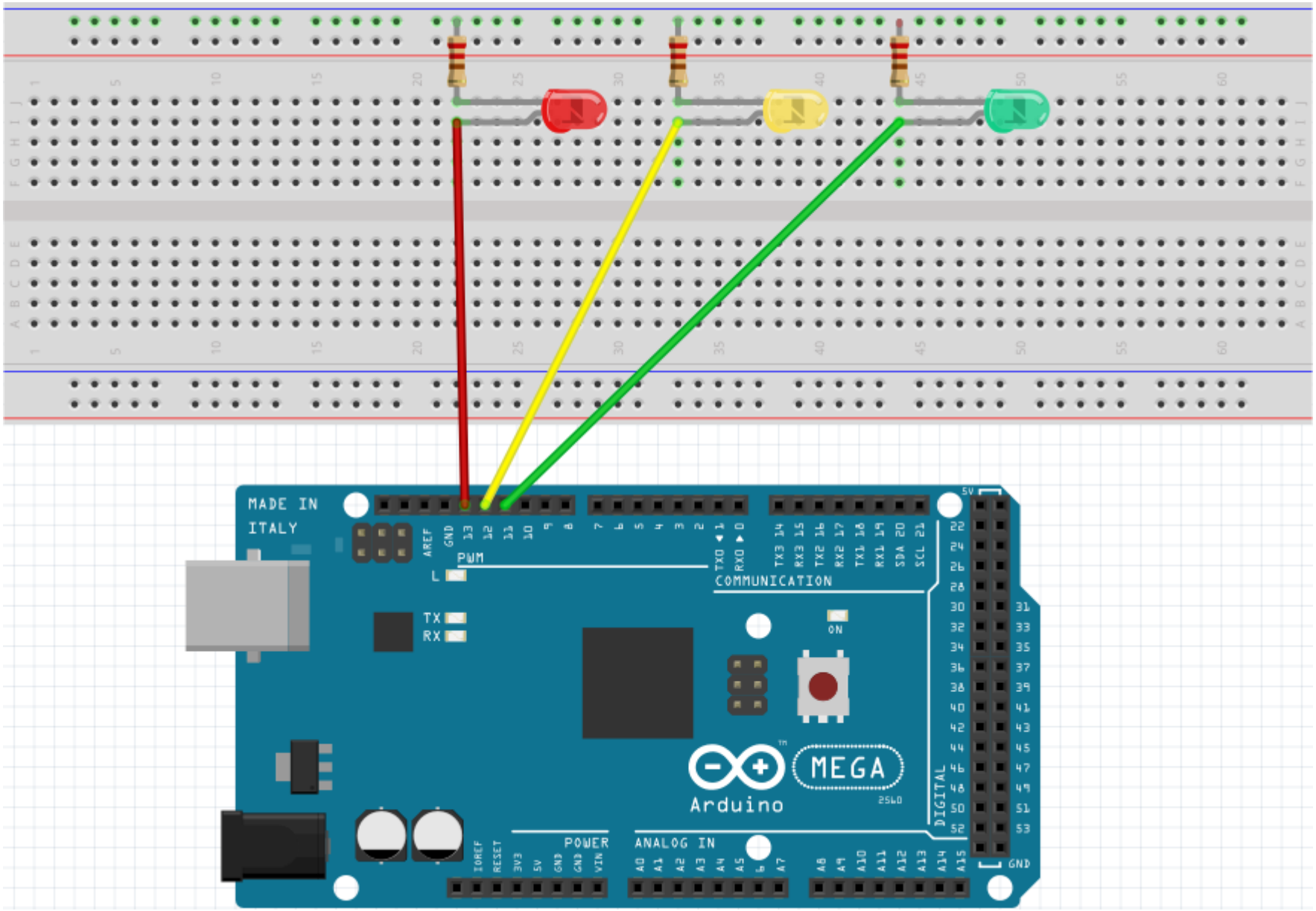
- Achten Sie auf die **{ }** Klammern
- Achten Sie auf das **;** am Ende jeder Befehlzeile
- Für die Verzögerung von 5 Sekunden wird die Integer Variable **int ledDelay = 5000** verwendet.

```
// Ampelschaltung
int ledDelay = 5000;                    // Umschaltverzögerung
int ledrot = 13;                        //  Port B5
int ledgelb = 12;                       //  Port B4
int ledgruen  = 11;                     //  Port B3

void setup()
{
  pinMode(ledrot,     OUTPUT);
  pinMode(ledgelb,   OUTPUT);
  pinMode(ledgruen, OUTPUT);
}

void loop()
{
  digitalWrite(ledrot, HIGH);           // schalte Rot ein
  delay(ledDelay);                      // warte 5 Sekunden

  digitalWrite(ledgelb, HIGH);          // schalte Gelb ein
  delay(2000);                          // warte 2 Sekunden

  digitalWrite(ledgruen, HIGH);         // schalte Gruen ein
  digitalWrite(ledrot, LOW);            // Schalte Rot aus
  digitalWrite(ledgelb, LOW);           // Schalte Gelb aus
  delay(ledDelay);                      // warte 5 Sekunden

  digitalWrite(ledgelb, HIGH);          //Schalte gelb ein
  digitalWrite(ledgruen, LOW);          // Schalte Gruen aus
  delay(2000);                          // Warte 2 Sekunden

  digitalWrite(ledgelb, LOW);           // Schalte Gelb aus
                                        // wiederhole Schleife endlos
 }
```

Ampelschaltung



Ampelschaltung

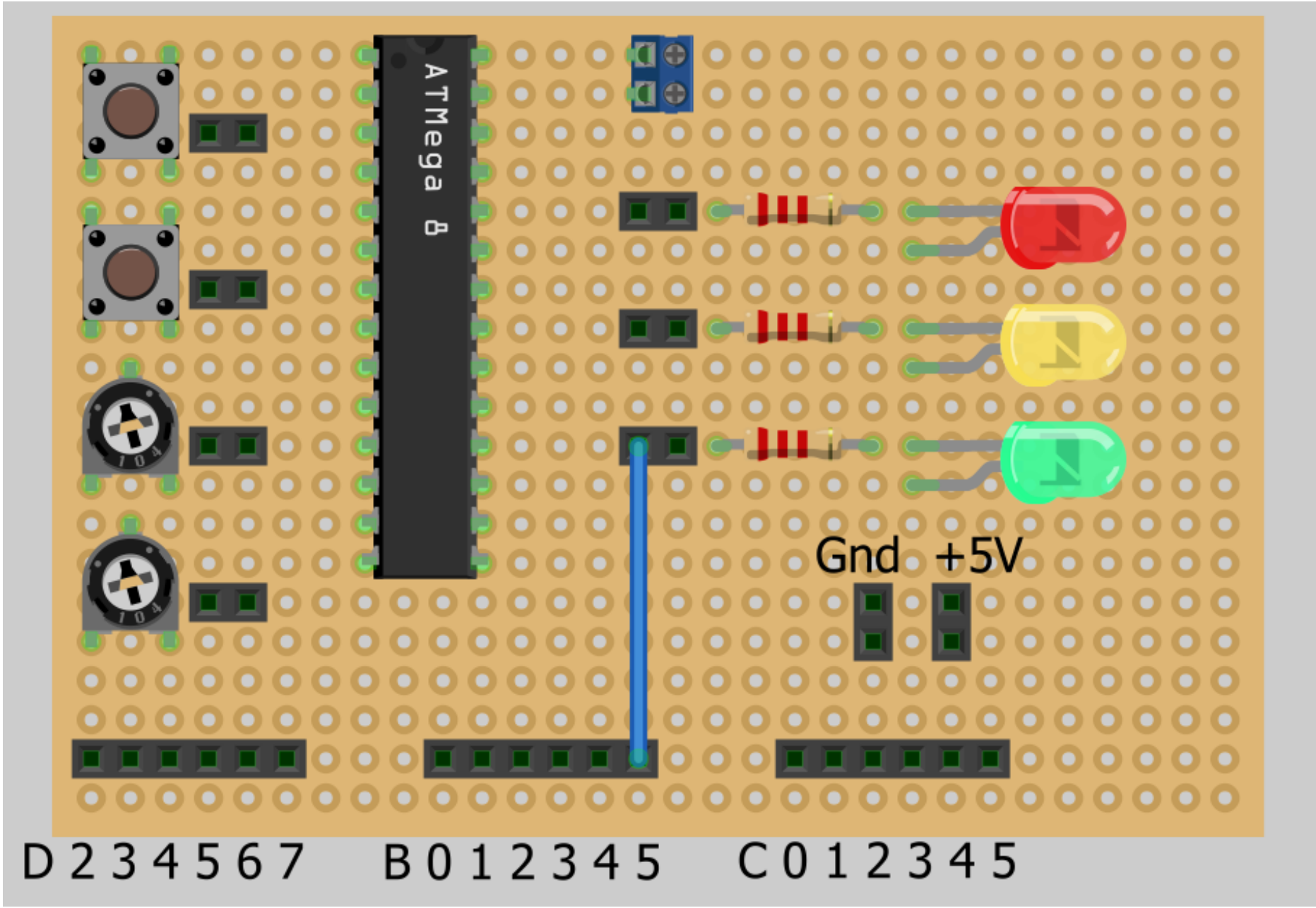


SOS Generator

```
// Project - SOS Generator
// LED verbunden mit LED Pin 13 =  Port B5
int ledPin = 13;

// Läuft nur 1 x beim Systemstart
void setup()
{
 // Setze Digitalpin als Ausgang
     pinMode(ledPin, OUTPUT);
}

// Loop, wiederhole immer wieder
void loop()
{
  // 3 dits
  for (int x=0; x<3; x++) {
  digitalWrite(ledPin, HIGH);              // Schalte LED ein
  delay(150);                              // warte 150ms
  digitalWrite(ledPin, LOW);               // Schalte LED aus
  delay(100);                              // Warte 100ms
 }

  // 100ms Abstand zwischen den Zeichen
  delay(100);

  // 3 dahs
  for (int x=0; x<3; x++) {
  digitalWrite(ledPin, HIGH);              // Schalte LED ein
  delay(400);                              // Warte 400ms
  digitalWrite(ledPin, LOW);               // Schalte LED aus
  delay(100);                              // Warte 100ms
  }
```

```
  // 100ms Abstand zwischen den Zeichen
  delay(100);

   // 3 dits noch einmal
  for (int x=0; x<3; x++) {
  digitalWrite(ledPin, HIGH);              // Schalte LED ein
  delay(150);                              // Warte 150ms
  digitalWrite(ledPin, LOW);               // Schalte LED aus
  delay(100);                              // Warte 100ms
  }

// Warte 5 Sekunden und beginne wiederhole
  delay(5000);
}
```

SOS Generator
Arduino LED Pin 13 (auf Board oder extern)



Das Programm enthält wieder viele bekannte Befehle. Man könnte das Programm mit zahlreichen Ein- / Aus - schaltbefehlen für 3 x Kurz - 3 x Lang - 3 x Kurz aufbauen.

Die Anzahl der Zeilen lässt sich reduzieren, wenn man **For .. Next Schleifen** verwendet:

```
for (int x=0; x<3; x++)
{
… ein paar Befehle die sich wiederholen
}
```

Zunächst wird eine Integer Variable deklariert und der Wert 0 zugewiesen: **int x=0**

Dann folgt eine **Bedingung**. Solange diese Bedingung **WAHR** ist, werden die Befehle innerhalb der **{ }** - Klammern wiederholt: **x < 3** Die Schleife wird solange wiederholt wie die Variable **x kleiner (<) 3** ist.

Durch das dritte Statement wird die Variable x bei jedem Durchlauf um 1 erhöht: **x++**
Statt **x++** könnte man auch schreiben **x= x +1**.

Praktisch werden die Befehle innerhalb von { } 3x wiederholt:

1. Durchlauf	**x == 0**	**0 ist kleiner 3** **am Ende wird x auf 1 erhöht**
2. Durchlauf	**x == 1**	**1 ist kleiner 3** **am Ende wird x auf 2 erhöht**
3. Durchlauf	**x == 2**	**2 ist kleiner 3** **am Ende wird x auf 3 erhöht**
4. Durchlauf	**x == 3**	**3 ist nicht kleiner 3** **Schleife wird abgebrochen**

Die folgen Bedingungen können in **For - Schleifen** verwendet werden.

== **(gleich)**
!= **(nicht gleich)**
< **(kleiner als)**
> **(größer als)**
<= **(kleiner ooder gleich)**
>= **(größer oder gleich)**

Achtung: x = 1 ist eine Wertzuweisung, x == 0 ist eine Bedingung (gleich)

Nun wäre es praktisch, wenn unser SOS – Programm auch noch Töne ausgeben könnte. Hierzu verbinden wird **Pin12 = PORT B4** mit dem **Summer**.

tone(12, 600);	Gibt Dauerton von 600 Hz an Pin 12 aus
NoTone(12);	Schaltet Ton an Pin 12 wieder aus
tone(12, 650, 100);	Gibt Ton von 650 Hz für 100 ms an Pin 12 aus

SOS Generator mit LED und Ton

```
// Project - SOS Generator
// LED verbunden mit LED Pin 13 =  Port B5
// Summer verbunden mit  Pin 12 =  Port B4

int ledPin = 13;

// Läuft nur 1 x beim Systemstart
void setup()
{
   // Setze Digitalpin als Ausgang
   pinMode(ledPin, OUTPUT);
}


// Loop, wiederhole immer wieder
void loop()
{
  // 3 dits
  for (int x=0; x<3; x++) {
  digitalWrite(ledPin, HIGH);        // Schalte   LED ein
  tone(12,600);                      // Ton ein
  delay(150);                        // warte 150ms
  digitalWrite(ledPin, LOW);         // Schalte LED aus
  noTone(12);                        // Ton aus
  delay(100);                        // Warte 100ms
  }
  // 100ms Abstand zwischen den Zeichen
  delay(100);
  // 3 dahs
  for (int x=0; x<3; x++) {
  digitalWrite(ledPin, HIGH);        // Schalte LED ein
  tone(12,600);                      // Ton ein
  delay(400);                        // Warte 400ms
  digitalWrite(ledPin, LOW);         // Schalte LED aus
  noTone(12);                        // Ton aus
  delay(100);                        // Warte 100ms
  }

  // 100ms Abstand zwischen den Zeichen
  delay(100);

   // 3 dits noch einmal
  for (int x=0; x<3; x++) {
  digitalWrite(ledPin, HIGH);        // Schalte LED ein
  tone(12,600);                      // Ton ein
  delay(150);                        // Warte 150ms
  digitalWrite(ledPin, LOW);         // Schalte LED aus
  noTone(12);                        // Ton aus
  delay(100);                        // Waits for 100ms
  }

  // Warte 5 Sekunden und beginne wiederhole
  delay(5000);
}
```