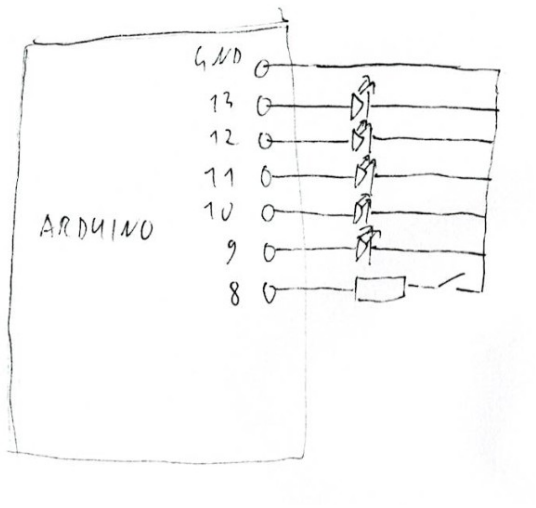
Einführung Arduino

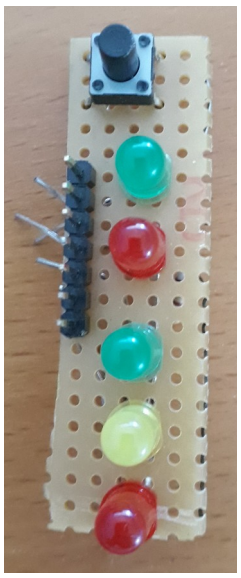
Nr	Titel	Was lerne ich dort?	Welche Befehle sind neu?	bearbeitet
0	Vorwort	Benötigte Hardware Hilfen	-	
1	Blink	Struktur von Arduino-Programmen Konfigurieren von Pins für Eingabe oder Ausgabe Ein- und Ausschalten von LEDs Wartezeit einbauen	void loop und void setup pinMode digitalWrite delay	
2	Wechselblinker	Umgang mit mehreren LEDs		
3	Lampen und Taster	Auf einen Tastendruck reagieren Programmteile nur	DigitalRead if else ==	
4	Serielle Konsole (millis)	Datenaustausch zwischen Arduino und PC	Serial.begin Serial.print Serial.println millis	
5	Warten auf Knopfdruck	Auf ein Ereignis warten	while	
6	Umschalten bei Knopfdruck	Der Arduino „merkt“ sich ein Ergebnis	-	
7	Blinken bis Knopfdruck	Umgang mit Varibalen	Variablen: int =	
8	Zeitversatz	Umgang mit Tabellen (Arrays) Zählschleife	Arrays: int messwerte[] for	
8b	Zeitversatz, kurz geschrieben	Kurze und lange Schreibweisen	-	
9	Analoge Ein- und Ausgänge	Umgang mit analogen Signalen	AnalogWrite analogRead	

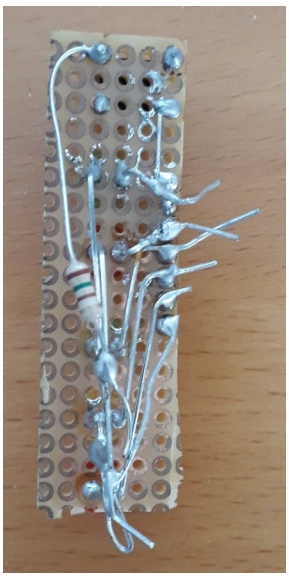
0. Vorwort

Hilfestellungen
Alle Programme sind hier abgedruckt. Zusätzlich finden sie sich auch auf der Website www.wuenscher.net/arduino.
In jedem Kapitel findet sich ein QR-Code mit dem Link zu den Programmen.

Verwendete Hardware
Alle Programme funktionieren mit der gleichen Hardware, die an den Arduino angeschlossen sein muss. Diese kann selbst gelötet werden.
Benötigte Bauteile:
LEDs, 5mm, geeignet für den Betrieb an 5V, 2x rot, 2x grün, 1x gelb
Lochrasterplatine, mindestens 5cm x 2cm, einseitig mit Lötaugen versehen reicht
Taster für Printmontage
Widerstand 150 Ohm oder 1 kOhm oder dazwischen







1. Blink

Ziel: Die Leuchtdiode an Pin 12 (gelbes Licht) soll abwechselnd an- und ausgeschaltet werden. Dazu sind folgende Befehle wichtig:

pinMode(12,OUTPUT)	versetzt den Pin 12 in den Ausgabemodus
digitalWrite(12,HIGH) bzw. digitalWrite(12,LOW)	setzt Pin 12 auf hohes Potential (+) bzw. niedriges Potenzial (-)
delay(1000)	wartet 1000 Millisekunden (1 Sekunde)

1. Tippe das folgende Programm ab.

```
void setup()
{
   pinMode(12,OUTPUT);
}

void loop()
{
   digitalWrite(12,HIGH);
   delay(1000);
   digitalWrite(12,LOW);
   delay(1000);
}
```

2. Verändere das Programm so, dass die LED schneller bzw. langsamer blinkt.

3. Verändere das Programm so, dass die grüne LED blinkt.

4. Verändere das Programm so, dass die LED immer kurz aufblitzt und dann länger dunkel ist.

5. Bestimme die Flimmer-Verschmelz-Frequenz. Das ist die Frequenz, bei der das Auge dem Flimmern der LED nicht mehr folgen kann.

2. Wechselblinker

Ziel: Die Leuchtdioden an Pin 11 (grünes Licht) und Pin 13 (rotes Licht) sollen abwechselnd blinken. Dazu sind folgende Befehle wichtig:

pinMode(12,OUTPUT)	versetzt den Pin 12 in den Ausgabemodus
digitalWrite(12,HIGH) bzw. digitalWrite(12,LOW)	setzt Pin 12 auf hohes Potential (+) bzw. niedriges Potenzial (-)
delay(1000)	warte 1000 Millisekunden (1 Sekunde)

1. Tippe das folgende Programm ab.

```
void setup()
{
   pinMode(11,OUTPUT);
   pinMode(13,OUTPUT);
}

void loop()
{
   digitalWrite(11,HIGH);
   digitalWrite(13,LOW);
   delay(1000);
   digitalWrite(11,LOW);
   digitalWrite(13,HIGH);
   delay(1000);
}
```

2. Erweitere das Programm zu einem Lauflicht, in dem die LEDs rot, gelb und grün immer nacheinander aufleuchten.

3. Verändere das Programm so, dass rote, gelbe und grüne LED eine Ampel nachbilden.

4. Verändere das Programm so, dass die fünf LEDs Auto- und Fußgängerampel mit überlappenden Rotphasen darstellen.

3. Taster

Ziel: Solange der Taster gedrückt ist, soll die grüne LED leuchten, ansonsten die rote LED. Dazu sind folgende neue Befehle wichtig:

pinMode(8,INPUT)	versetzt den Pin 8 in den Eingabemodus
digitalWrite(8,HIGH) bei einem Pin der auf INPUT geschaltet ist	sorgt dafür, dass das Ergebnis zuverlässig HIGH ist, wenn der Taster nicht gedrückt ist

if (Bedingung)
{
}
else
{
}
	Wenn die Bedingung erfüllt ist, werden die Befehle ausgeführt, die in den ersten geschweiften Klammern stehen, ansonsten werden die Befehle ausgeführt, die in den zweiten geschweiften Klammern stehen. Der Teil ab else darf auch weggelassen werden.

digitalRead(8)	liest den Zustand von Pin 8 aus
==	führt einen Vergleich durch

1. Tippe das folgende Programm ab.

```
void setup()
{
   pinMode(8,INPUT);
   digitalWrite(8,HIGH);
   pinMode(11,OUTPUT);
   pinMode(13,OUTPUT);
}

void loop()
{
   if(digitalRead(8)==LOW)
   {
      digitalWrite(11,HIGH);
      digitalWrite(13,LOW);
   }
   else
   {
      digitalWrite(11,LOW);
      digitalWrite(13,HIGH);
   }
   delay(10);
}
```

2. Verändere das Programm so, dass es bei gedrückter Taste zwischen grün und rot hin und her wechselt, bei losgelassener Taste jedoch garnicht leuchtet.

3. Erweitere Dein Programm zu einer Bedarfsampel: Solange der Taster nicht gedrückt wird, ist die Autoampel grün und die Fußgängerampel rot. Wird der Taster gedrückt, wird die Autoampel rot, dann die Fußgängerampel auf grün und eine Weile wieder auf rot. Hatten die Fußgänger genug Zeit die Fahrbahn zu verlassen, schaltet die Autoampel wieder auf grün.

4. Serielle Konsole

Ziel: Der Arduino sendet sekündlich eine Nachricht an den Computer, ob der an Pin 8 angeschlossene Taster gedrückt ist.

Serial.begin(9600)	stellt die Verbindung zum Computer mit 9600 Baud (Bit pro s) her
Serial.print()	überträgt Daten an den Computer
Serial.println()	überträgt Daten an den Computer und sorgt für einen Zeilenumbruch
millis()	gibt die Millisekunden seit dem Einschalten des Arduinos aus

1. Tippe das folgende Programm ab. Öffne nach dem Hochladen den Seriellen Monitor mit dem Button  oben rechts in der Arduino-Umgebung.

```
void setup()
{
   pinMode(8,INPUT);
   digitalWrite(8,HIGH);
   Serial.begin(9600);
   Serial.println("Die Uebertragung kann beginnen.");
}

void loop()
{
   if(digitalRead(8)==LOW)
   {
      Serial.println("Taster gedrueckt");
   }
   else
   {
      Serial.println("Taster nicht gedrueckt");
   }
   delay(1000);
}
```

2. Verändere Dein Programm so, dass es zusätzlich und in der gleichen Zeile die seit dem Start vergangenen Sekunden an den Computer überträgt.

5. Warten bis Knopfdruck

Ziel: Der Arduino wartet (z.B. bei rot zeigender Fußgängerampel und grün zeigender Autoampel), bis der Knopf gedrückt ist. Dann läuft der Ampelzyklus einmal durch.
Dazu sind folgende neue Befehle wichtig:

while (Bedingung)
{
  Befehle
}
	Solange die Bedingung in der runden Klammer erfüllt ist, werden die in der geschweiften Klammer stehenden Befehle immer wieder ausgeführt.

1. Tippe das folgende Programm ab.

```
void setup()
{
  pinMode(8,INPUT);
  digitalWrite(8,HIGH);
  pinMode(9,OUTPUT);
  ...
}

void loop()
{
  … Befehle, die die rote Fußgängerampel und die grüne Autoampel einschalten
  while(digitalRead(8)==HIGH)
  {
    delay(10);
  }
  … Befehle (mit delays), die einen Ampelzyklus durchlaufen lassen
}
```

2. Verändere das Programm so, dass die Fußgänger so lange grün behalten, wie die Taste gedrückt ist. Tipp: Dazu brauchst Du an einer anderen Stelle nochmal eine Zeile mit while.

6. An- und Ausschalten

Ziel: Durch kurzes Antippen des Tasters wird die rote LED an- und ausgeschaltet.

1. Tippe das folgende Programm ab.

```
void setup()
{
   pinMode(8,INPUT);
   digitalWrite(8,HIGH);
   pinMode(13,OUTPUT);
   digitalWrite(13,LOW);
}

void loop()
{
   while(digitalRead(8)==HIGH)
   {
      delay(10);
   }
   digitalWrite(13,HIGH);
   while(digitalRead(8)==LOW)
   {
      delay(10);
   }
   while(digitalRead(8)==HIGH)
   {
      delay(10);
   }
   digitalWrite(13,LOW);
   while(digitalRead(8)==LOW)
   {
      delay(10);
   }
}
```

2. Wird der Taster gedrückt oder losgelassen, kann es passieren, dass er für ein paar Millisekunden prellt, also mechanisch federt und dabei mehrfach seinen Zustand verändert. Passe Dein Programm so an, dass das Prellen nicht zu unerwünschtem Schalten führt, ohne dass dadurch der Knopf schlechter bedienbar wird.

7. Blinken bis Knopfdruck

Ziel: Eine gelb blinkende Ampel soll durch einen Knopfdruck ein- und ausgeschaltet werden. Dazu muss der Arduino zwei Aufgaben erledigen: Den Knopf überwachen und die Ampel blinken lassen.

int zaehler;	Legt eine Variable an, in der man ganze Zahlen speichern kann.
zaehler=zaehler-1;	Reduziert den Inhalt der Variablen um Eins.

1. Tippe das folgende Programm ab.

```
void setup()
{
   pinMode(8,INPUT);
   digitalWrite(8,HIGH);
   pinMode(12,OUTPUT);
}
void loop()
{
   int zaehler;
   zaehler=100;
   digitalWrite(12,HIGH);
   while(zaehler>0)
   {
      zaehler=zaehler-1;
      delay(10);
      if(digitalRead(8)==LOW)
      {
         zaehler=-1;
      }
   }
   digitalWrite(12,LOW);
   if(zaehler>-1)
   {
      zaehler=100;
      while(zaehler>0)
      {
         zaehler=zaehler-1;
         delay(10);
         if(digitalRead(8)==LOW)
         {
            zaehler=-1;
         }
      }
   }
   if(zaehler==-1)
   {
      delay(200);
      while(digitalRead(8)==HIGH)
      {
         delay(10);
      }
      delay(200);
   }
}
```

8. Zeitversatz

Ziel: Die rote LED soll anzeigen, was vor einer Sekunde am Taster gemacht wurde. Dazu wird alle 10 Millisekunden der Zustand des Tasters gespeichert. Man braucht also 100 Speicherplätze. Dies passiert über ein sogenanntes Array.
Dazu sind folgende neue Befehle wichtig:

int messwerte[100]	legt eine Tabelle mit 100 Einträgen an.
for(int i=0;i<100;i++)	Zählschleife. Geht den dahinter folgenden Block durch, solange der Wert von i kleiner 100 ist, beginnt dabei mit Null und erhöht i nach jedem Durchgang

1. Tippe das folgende Programm ab.

```
int messwerte[100];

void setup()
{
   pinMode(8,INPUT);
   digitalWrite(8,HIGH);
   pinMode(13,OUTPUT);
   for(int i=0;i<100;i++)
   {
      messwerte[i]=0;
   }
}

void loop()
{
   for(int i=0;i<100;i++)
   {
      if(messwerte[i]==1)
      {
         digitalWrite(13,HIGH);
      }
      else
      {
         digitalWrite(13,LOW);
      }
      if(digitalRead(8)==LOW)
      {
         messwerte[i]=1;
      }
      else
      {
         messwerte[i]=0;
      }
      delay(10);
   }
}
```

8b. Zeitversatz

Ziel: Das Programm aus Aufgabe 8 soll deutlich kürzer geschrieben werden.

1. Tippe das folgende Programm ab.

```
int messwerte[100];

void setup()
{
   pinMode(8,INPUT);
   digitalWrite(8,HIGH);
   pinMode(13,OUTPUT);
   for(int i=0;i<100;i++)
      messwerte[i]=0;
}

void loop()
{
   for(int i=0;i<100;i++)
   {
      digitalWrite(13,messwerte[i]);
      messwerte[i]=1-digitalRead(8);
      delay(10);
   }
}
```

2. Analysiere die Unterschiede zum vorherigen Programm und versuche sie zu verstehen.

9. Analoge Ein- und Ausgänge

Ziel: Die Spannung am analogen Pin A3 soll gemessen werden und die Helligkeit der LED an Pin 11 entsprechend eingestellt werden.
Dazu sind folgende neue Befehle wichtig:

analogWrite(11,128)	Stellt am Pin 11 die Helligkeit 128 ein. Maximale Helligkeit erreicht man über 255, minimale Helligkeit über 0. Auf diese Weise lassen sich alle Pins ansteuern, die auf der Platine neben ihrer Nummer eine Schlangenlinie haben
analogRead(3)	Misst die Spannung am Analogeingang 3 und gibt einen zur Spannung proportionalen Wert zwischen 0 und 1023 zurück.

1. Tippe das folgende Programm ab.

```
void setup()
{
   pinMode(11,OUTPUT);
}

void loop()
{
   analogWrite(11,analogRead(3)/4);
   delay(10);
}
```

2. Verändere das Programm so, dass sich der Helligkeitseindruck möglichst gleichmäßig verändert.

3. Erweitere Dein Programm so, dass es mit einer Zeitverzögerung von einer Sekunde reagiert. Nutze dazu die Verfahren aus den Aufgabenblättern 8 und 8b.