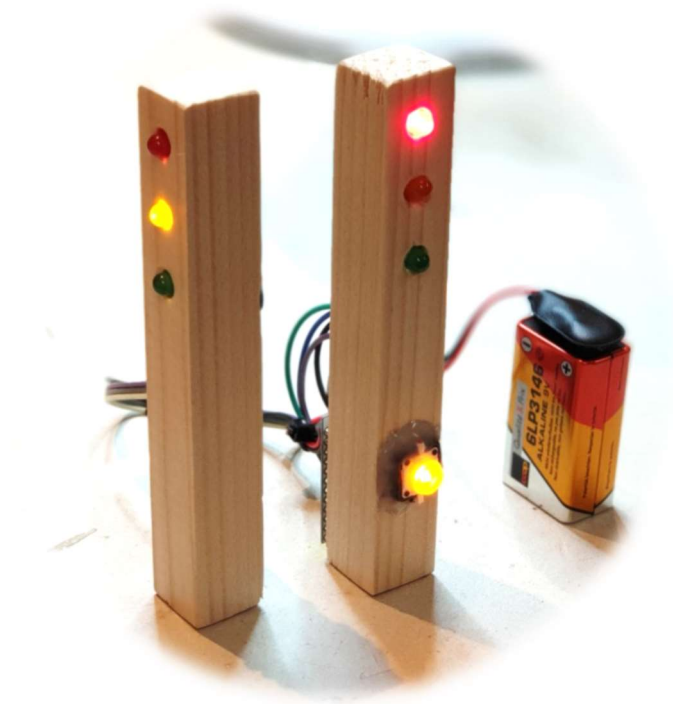


Ampel- Steuerung

mit Arduino



Manuel Haag

April 2020

1 Einleitung

In diesem Projekt bauen wir eine Ampelanlage bestehend aus:

- einer Ampel für Autos
- einer Ampel für Fussgänger mit Knopf
- einer Steuerung mit Arduino

Die Anlage läuft selbständig mit einer 9V Batterie.

1.1 Funktionsweise der Ampel

Beim Start (oder nach RESET) blinken beide Ampeln orange.

Sobald der Knopf bei der Fussgängerampel gedrückt wird, schalten zuerst beide Ampeln auf Rot, danach folgt die Grünphase (10 Sekunden) für Fussgänger.

Danach stellt die Fussgängerampel auf Rot und die Autos bekommen grün (zeitlich unbeschränkt).

Bei erneutem Drücken des Knopfes erfolgt wieder eine Grünphase für die Fussgänger.

Die Grünphase für Autos dauert jedoch mindestens 10 Sekunden (falls der Knopf zu früh gedrückt wird).

2 Bauteile

2.1 Stromversorgung

Die Ampel wird mit einer 9V Batterie betrieben und über einen Stecker mit Kabel am Arduino angeschlossen.

ACHTUNG: die Batterie darf nicht verkehrt angeschlossen werden, sonst könnte die Steuerung kaputt gehen!

Beim Anschliessen des Steckers an die Batterie immer darauf achten, dass die Polung stimmt!

Die beiden abisolierten Kabelenden (rot und schwarz) dürfen sich nicht berühren, sonst gibt es einen Kurzschluss und das Kabel wird heiss oder beginnt zu schmelzen!

Rot = Pluspol

Schwarz = Minuspol



Aufgaben:

1. Schliesse die Batterie an ein Messgerät (Voltmeter) an und überprüfe, wieviel Spannung (Volt) die Batterie noch hat. Wenn die Batterie neu ist, sollte sie etwas mehr als 9V anzeigen.
2. Was passiert, wenn du Plus und Minus vertauschst?

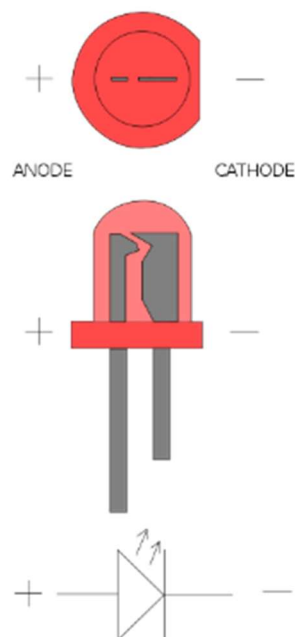
Tipp: Spannungen bis etwa 50 Volt sind ungefährlich. Über 50 Volt ist gefährlich und darf nicht berührt werden! Die Steckdose hat 230V und ist somit lebensgefährlich!

2.2 LED

Leuchtdioden (Englisch: Light Emitting Diode – *LED*) sind kleine Lämpchen. Es gibt sie in verschiedenen Grössen und Farben. Wir verwenden Gelb, Rot und Grün.



Die LEDs funktionieren nur, wenn Plus und Minus richtig angeschlossen sind. Den Minuspol erkennt man am kürzeren Bein und an der flachen Seite der LED:



Damit die LEDs nicht kaputt gehen, braucht es vor jeder LED einen Schutzwiderstand.

2.3 Widerstand

Widerstände «bremsen» den Stromfluss. Es gibt schwache und starke Widerstände. Der Wert des Widerstands ist mit Farbringen gekennzeichnet und heisst «Ohm». Beim Widerstand spielt es keine Rolle, wie man ihn anschliesst (Plus/Minus).



Widerstände können zum Beispiel dazu dienen, LEDs vor zu hohen Strömen zu schützen. Für unsere Ampel braucht es vor jeder LED einen kleinen Widerstand von 220 Ohm. Dieser hat die Farbcodierung ROT-ROT-BRAUN:

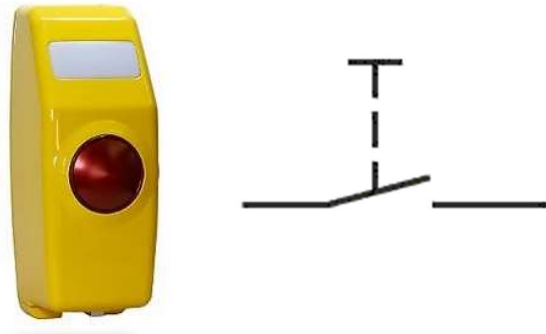


Aufgaben:

1. Löte einen 560 Ohm Widerstand (Farbcode GRÜN-BLAU-BRAUN) an ein Bein einer LED.
2. Schliesse die LED mit dem Vorwiderstand an die 9V Batterie an, so dass sie leuchtet.
3. Was passiert, wenn du Plus und Minus vertauschst?

2.4 Taster Knopf

Ein Taster ist ein kleiner Schalter, welcher den Stromkreis schliesst, solange er gedrückt wird. Damit kann man der Ampelsteuerung mitteilen, dass jemand über den Fussgängerstreifen gehen will.



Wir verwenden diesen Taster mit eingebautem LED-Lämpchen, welches aufleuchtet, wenn der Knopf gedrückt wurde.



Aufgaben:

1. Finde mithilfe eines «Durchgangspieters» heraus, welche Anschlussbeine zum Taster gehören und welche zur LED.
2. Mache eine kleine Zeichnung mit den Anschlüssen für später.

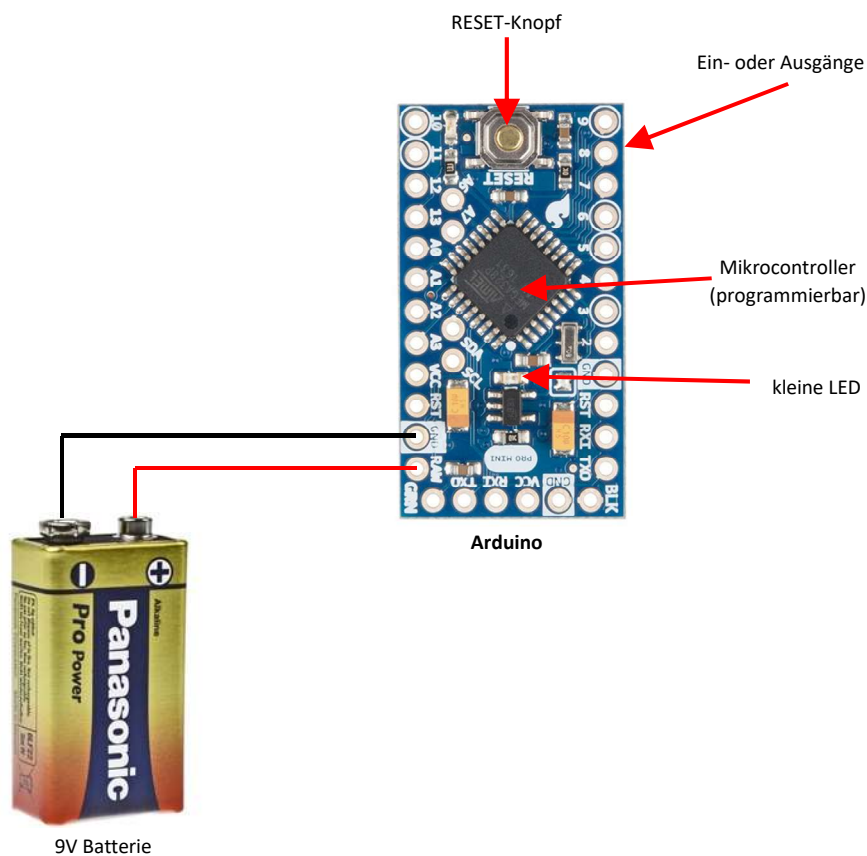
2.5 Arduino

Der Arduino ist ein kleiner Computer. Er besteht aus einem Mikrocontroller, den man programmieren kann. Wir werden ihn so programmieren, dass er die LEDs der Ampel aufleuchten lässt und die Ampel genau so funktioniert, wie wir es möchten.

Er hat mehrere Anschlüsse für Ein- und Ausgänge. An den Ausgängen werden die LEDs angeschlossen, an einem Eingang der Taster.

Der Arduino benötigt eine Stromversorgung (9V Batterie) und kann mit dem RESET-Knopf jederzeit neu gestartet werden.

Zusätzlich hat es auf dem Arduino eine kleine grüne LED, welche nützlich ist, wenn man schnell etwas Testen will.



3 Programmierung des Arduino

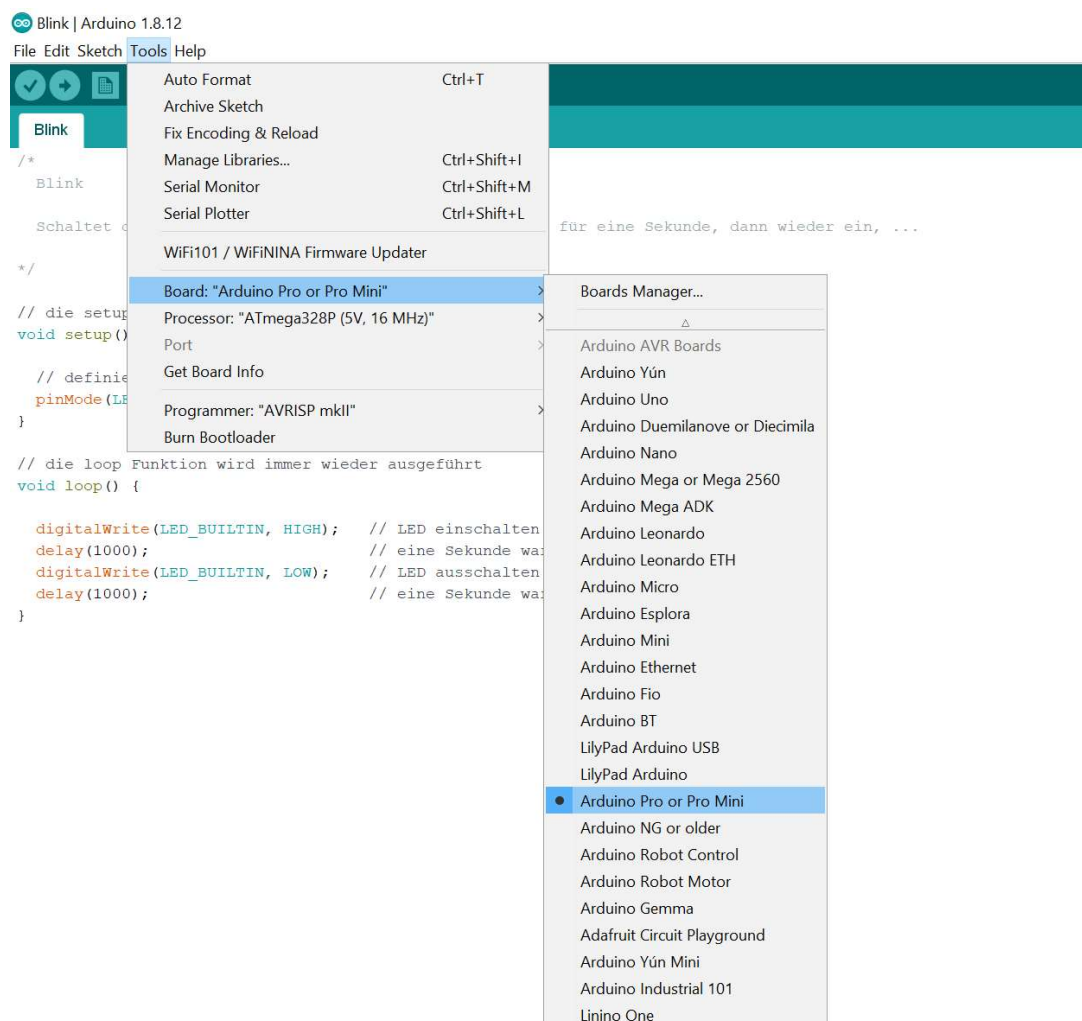
3.1 Programmierung über FTDI-Kabel

Der Arduino wird über einen normalen PC oder Laptop mit einem speziellen FTDI Kabel über USB programmiert. Das Programm bleibt auf dem Arduino gespeichert, auch wenn man die Batterie abhängt. Wenn er einmal programmiert ist, läuft der Arduino selbständig (ohne PC). Jedesmal, wenn man am Programm etwas ändern will, muss man den Arduino neu programmieren.

Für die Programmierung benötigt man eine Entwicklungsumgebung auf dem PC namens Arduino IDE. Diese kann hier downgeloadet werden:

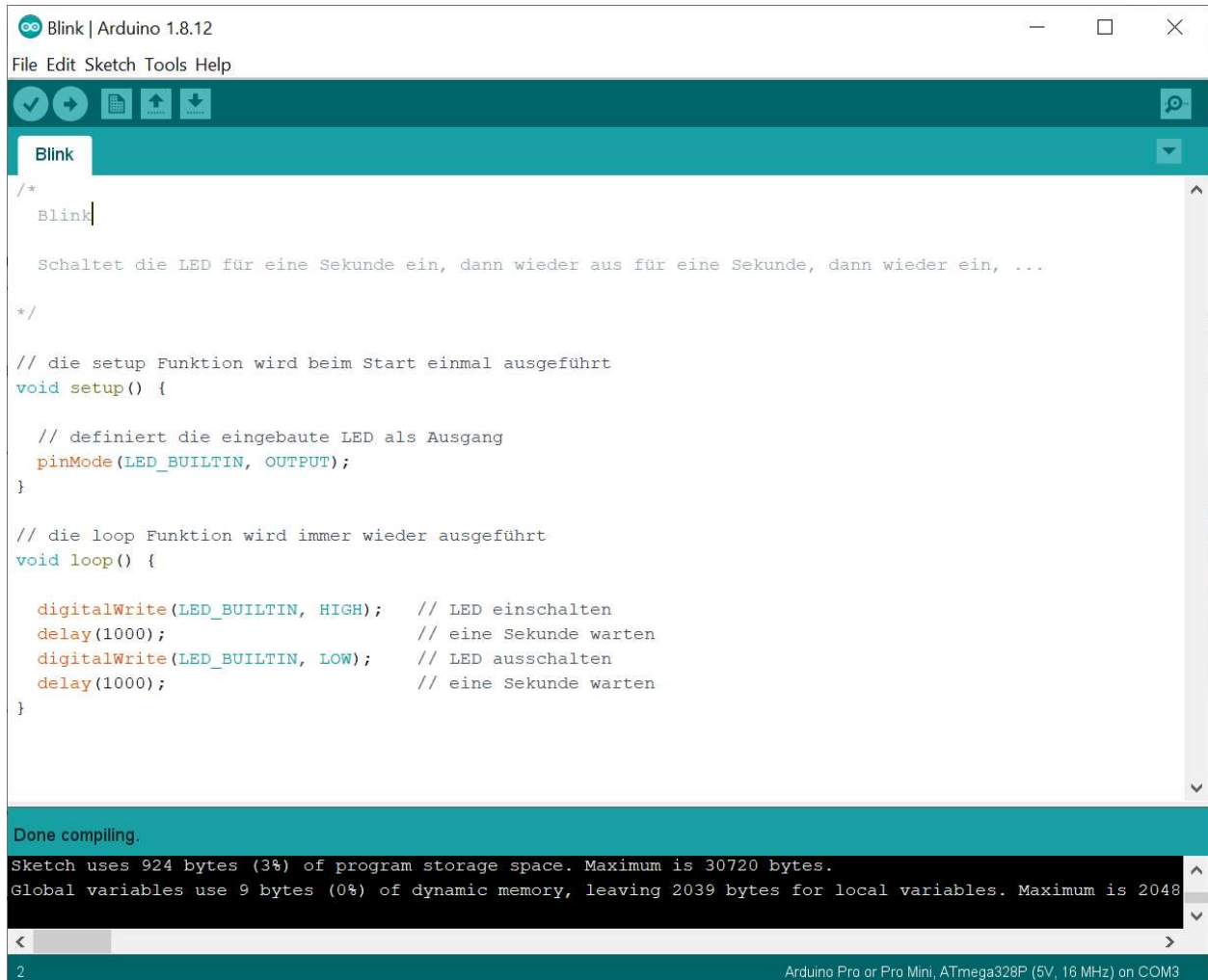
<https://www.arduino.cc/en/Main/Software>

Unter Tools – Board muss «Arduino Pro or Pro Mini» ausgewählt werden



3.2 Erstes Testprogramm «Blink»

Zu Beginn schreiben wir ein erstes kleines Programm, welches die kleine grüne LED auf dem Arduino blinken lässt.

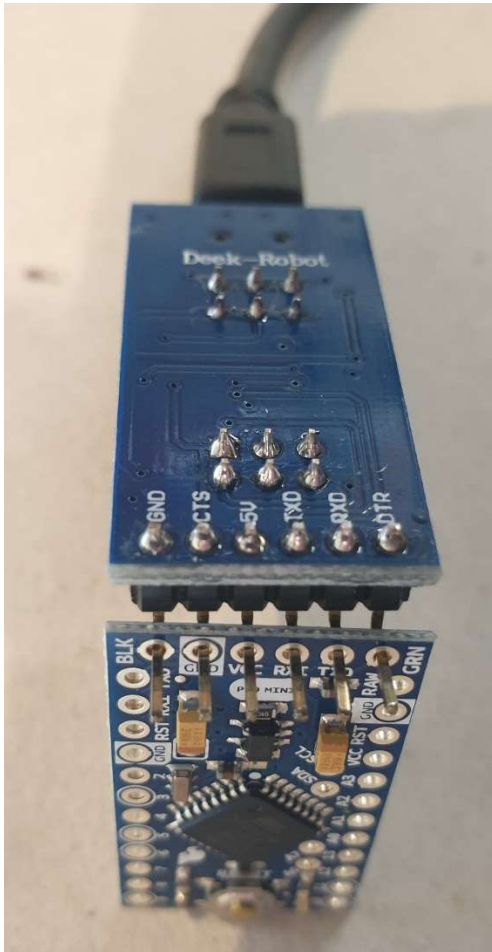
The image is a screenshot of the Arduino IDE interface. At the top, the window title is "Blink | Arduino 1.8.12". Below the title bar is a menu bar with "File", "Edit", "Sketch", "Tools", and "Help". Underneath the menu bar is a toolbar with icons for checking, running, saving, uploading, and downloading. A tab labeled "Blink" is active. The main text area contains the following code:

```
/*  
  Blink  
  
  Schaltet die LED für eine Sekunde ein, dann wieder aus für eine Sekunde, dann wieder ein, ...  
*/  
  
// die setup Funktion wird beim Start einmal ausgeführt  
void setup() {  
  
  // definiert die eingebaute LED als Ausgang  
  pinMode(LED_BUILTIN, OUTPUT);  
}  
  
// die loop Funktion wird immer wieder ausgeführt  
void loop() {  
  
  digitalWrite(LED_BUILTIN, HIGH); // LED einschalten  
  delay(1000); // eine Sekunde warten  
  digitalWrite(LED_BUILTIN, LOW); // LED ausschalten  
  delay(1000); // eine Sekunde warten  
}
```

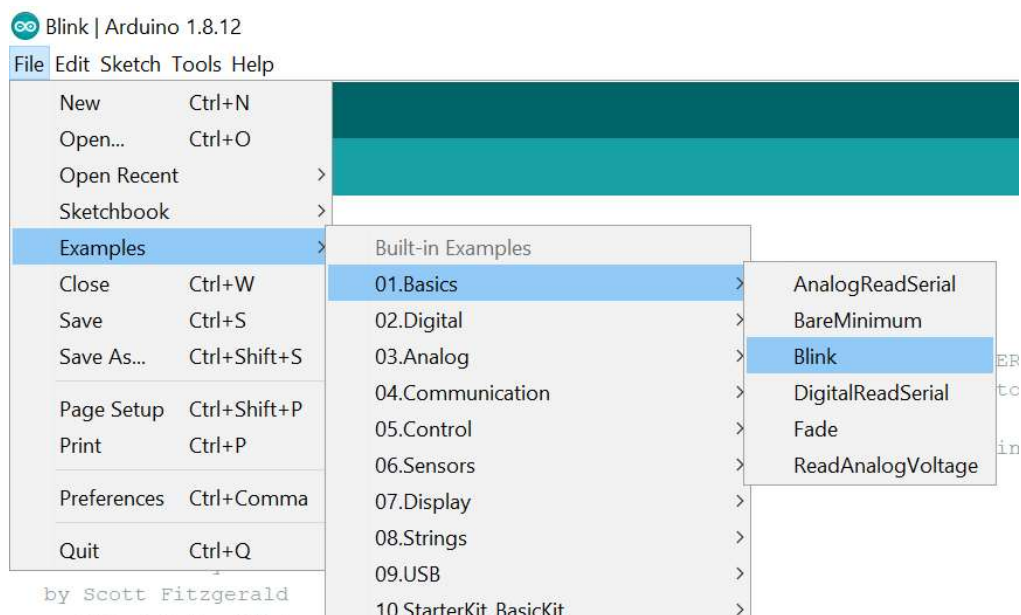
At the bottom of the IDE, there is a status bar. The top part of the status bar says "Done compiling." in a teal background. Below that, on a black background, it says "Sketch uses 924 bytes (3%) of program storage space. Maximum is 30720 bytes." and "Global variables use 9 bytes (0%) of dynamic memory, leaving 2039 bytes for local variables. Maximum is 2048". The bottom of the status bar shows a line number "2" on the left and "Arduino Pro or Pro Mini, ATmega328P (5V, 16 MHz) on COM3" on the right.

Aufgaben:

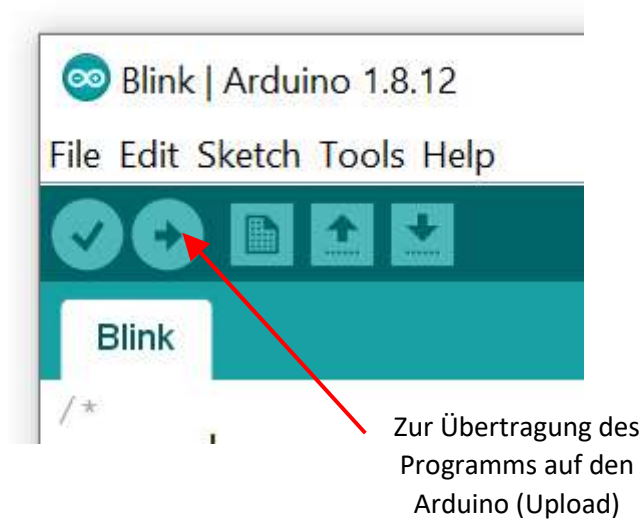
1. Verbinde den Arduino mit dem FTDI-Kabel über USB mit dem PC genau wie abgebildet. Stecke vorher die Batterie ab, falls es eine hat.



2. Öffne das Blink Demo Programm unter File → Examples → 01.Basics → Blink



- Übertrage das Programm «Blink» auf den Arduino indem du auf «Upload» klickst.



- Das Programm «Blink» startet automatisch nach erfolgreichem Upload. Beobachte, ob die grüne LED im Sekundentakt blinkt. Du kannst den Arduino auch vom FTDI-Kabel abhängen und stattdessen mit Batterie betreiben.
- Verändere die Werte von `delay(1000)` auf Zeile 34 und 36. Gib zum Beispiel `delay(100)` oder `delay(3000)` ein. Übertrage das veränderte Programm erneut auf den Arduino. Was passiert?

Tipp: 1 Sekunde = 1000 Millisekunden (Tausendstelsekunden)

3.3 Programm für die Ampelsteuerung

Nun ist es Zeit, das richtige Ampelprogramm auf den Arduino zu laden.

Der Quellcode ist hier zu finden: <https://github.com/helijunky/Ampel>

Übertrage das Programm wie zuvor mit dem FTDI-Kabel.

```
/*
Ampelsteuerung mit Arduino

Manuel Haag
März 2020

Funktionsweise
-----
Beim Start (oder nach RESET) blinken beide Ampeln orange.
Sobald der Knopf bei der Fussgängerampel gedrückt wird, schalten zuerst
beide Ampeln auf Rot, danach folgt die Grünphase für Fussgänger
(10 Sekunden).
Danach stellt die Fussgängerampel auf Rot und die Autos bekommen grün
(zeitlich unbeschränkt).
Bei erneutem Drücken des Knopfes erfolgt wieder eine Grünphase für die
Fussgänger. Die Grünphase für Autos dauert jedoch mindestens 10 Sekunden
(falls der Knopf zu früh gedrückt wird).

*/

// Definition der Namen für Ein- und Ausgänge am Arduino
#define LED_AUTO_ROT 9           // Ausgang 9 für Auto Rot
#define LED_AUTO_GELB 8          // Ausgang 8 für Auto Gelb
#define LED_AUTO_GRUEN 7         // Ausgang 7 für Auto Grün

#define LED_FUSSG_ROT 6           // Ausgang 6 für Fussgänger Rot
#define LED_FUSSG_GELB 5          // Ausgang 5 für Fussgänger Gelb
#define LED_FUSSG_GRUEN 4         // Ausgang 4 für Fussgänger Grün

#define LED_KNOPF 3               // Ausgang 3 für LED in Knopf
#define KNOPF 2                   // Eingang 2 für Knopf

// Variablen
bool blinken = true;             // true: Blinken eingeschaltet
                                  // false: Blinken ausgeschaltet

volatile bool gedrueckt;          // true: Knopf wurde gedrückt
                                  // false: Knopf nicht gedrückt

// die setup Funktion wird beim Start einmal ausgeführt
void setup() {

    // alle LEDs werden als Ausgang definiert
    pinMode(LED_BUILTIN, OUTPUT);
    pinMode(LED_AUTO_ROT, OUTPUT);
    pinMode(LED_AUTO_GELB, OUTPUT);
    pinMode(LED_AUTO_GRUEN, OUTPUT);
    pinMode(LED_FUSSG_ROT, OUTPUT);
    pinMode(LED_FUSSG_GELB, OUTPUT);
    pinMode(LED_FUSSG_GRUEN, OUTPUT);
    pinMode(LED_KNOPF, OUTPUT);

    // der Knopf wird als Eingang definiert
```

```

pinMode(KNOPF, INPUT_PULLUP);

// sobald der Knopf gedrückt wird, wird die Funktion "druecken" aufgerufen
// (Interrupt-Funktion)
attachInterrupt(digitalPinToInterrupt(KNOPF), druecken, FALLING);

// zum Testen werden alle LEDs kurz eingeschaltet
digitalWrite(LED_AUTO_ROT, HIGH);
digitalWrite(LED_AUTO_GELB, HIGH);
digitalWrite(LED_AUTO_GRUEN, HIGH);
digitalWrite(LED_FUSSG_ROT, HIGH);
digitalWrite(LED_FUSSG_GELB, HIGH);
digitalWrite(LED_FUSSG_GRUEN, HIGH);
digitalWrite(LED_KNOPF, HIGH);
digitalWrite(LED_BUILTIN, HIGH);
delay(2000);

// und wieder ausgeschaltet
digitalWrite(LED_AUTO_ROT, LOW);
digitalWrite(LED_AUTO_GELB, LOW);
digitalWrite(LED_AUTO_GRUEN, LOW);
digitalWrite(LED_FUSSG_ROT, LOW);
digitalWrite(LED_FUSSG_GELB, LOW);
digitalWrite(LED_FUSSG_GRUEN, LOW);
digitalWrite(LED_KNOPF, LOW);
digitalWrite(LED_BUILTIN, LOW);
delay(1000);

gedrueckt = false; // erster Knopfdruck löschen (Interrupt wird bei jedem Start
                  // einmal ausgeführt)
}

// Interrupt-Funktion wird aufgerufen, sobald der Knopf gedrückt wird
void druecken() {
  if (!gedrueckt) { // falls noch nicht gedrückt wurde...
    gedrueckt = true; // sich merken, dass Knopf gedrückt wurde
    digitalWrite(LED_KNOPF, HIGH); // LED im Knopf einschalten
    digitalWrite(LED_BUILTIN, HIGH); // kleine LED auf dem Arduino einschalten
  }
}

// die loop Funktion wird immer wieder ausgeführt
void loop() {

  // Blinken der orangen LEDs
  if (blinken) { // wenn Blinken aktiv ist
    digitalWrite(LED_AUTO_GELB, HIGH); // gelbe LED für Auto einschalten
    digitalWrite(LED_FUSSG_GELB, HIGH); // gelbe LED für Fussgänger einschalten
    digitalWrite(LED_BUILTIN, HIGH); // kleine LED auf dem Arduino einschalten
    delay(700); // 700 Millisekunden warten

    digitalWrite(LED_AUTO_GELB, LOW); // gelbe LED für Auto ausschalten
    digitalWrite(LED_FUSSG_GELB, LOW); // gelbe LED für Fussgänger ausschalten
    digitalWrite(LED_BUILTIN, LOW); // kleine LED auf dem Arduino ausschalten
    delay(700); // 700 Millisekunden warten
  }

  if (gedrueckt) { // wenn gedrückt wurde...
    if (blinken) { // wenn noch Blinken aktiv ist...
      blinken = false; // Blinken ausschalten

      // Fussgänger Ampel auf Orange stellen
      digitalWrite(LED_FUSSG_ROT, LOW);
      digitalWrite(LED_FUSSG_GELB, HIGH);
      digitalWrite(LED_FUSSG_GRUEN, LOW);
    }
  }
}

```

```

// Auto Ampel auf Orange stellen
digitalWrite(LED_AUTO_ROT, LOW);
digitalWrite(LED_AUTO_GELB, HIGH);
digitalWrite(LED_AUTO_GRUEN, LOW);
delay(3000); // 3 Sekunden warten

// Auto Ampel auf Rot stellen
digitalWrite(LED_AUTO_ROT, HIGH);
digitalWrite(LED_AUTO_GELB, LOW);
digitalWrite(LED_AUTO_GRUEN, LOW);

// Fussgänger Ampel auf Rot stellen
digitalWrite(LED_FUSSG_ROT, HIGH);
digitalWrite(LED_FUSSG_GELB, LOW);
digitalWrite(LED_FUSSG_GRUEN, LOW);
delay(2000); // 2 Sekunden warten

// Fussgänger Ampel auf Grün stellen
digitalWrite(LED_FUSSG_ROT, LOW);
digitalWrite(LED_FUSSG_GELB, LOW);
digitalWrite(LED_FUSSG_GRUEN, HIGH);
digitalWrite(LED_KNOPF, LOW); // LED im Knopf ausschalten
digitalWrite(LED_BUILTIN, LOW); // kleine LED ausschalten
delay(10000); // 10 Sekunden warten

// Fussgänger Ampel auf Orange stellen
digitalWrite(LED_FUSSG_ROT, LOW);
digitalWrite(LED_FUSSG_GELB, HIGH);
digitalWrite(LED_FUSSG_GRUEN, LOW);
delay(3000); // 3 Sekunden warten

// Fussgänger Ampel auf Rot stellen
digitalWrite(LED_FUSSG_ROT, HIGH);
digitalWrite(LED_FUSSG_GELB, LOW);
digitalWrite(LED_FUSSG_GRUEN, LOW);
gedrueckt = false; // ab jetzt kann man wieder drücken
delay(2000); // 2 Sekunden warten

// Auto Ampel auf Rot+Orange stellen
digitalWrite(LED_AUTO_ROT, HIGH);
digitalWrite(LED_AUTO_GELB, HIGH);
digitalWrite(LED_AUTO_GRUEN, LOW);
delay(2000); // 2 Sekunden warten

// Auto Ampel auf Grün stellen
digitalWrite(LED_AUTO_ROT, LOW);
digitalWrite(LED_AUTO_GELB, LOW);
digitalWrite(LED_AUTO_GRUEN, HIGH);
delay(10000); // 10 Sekunden warten
}
}

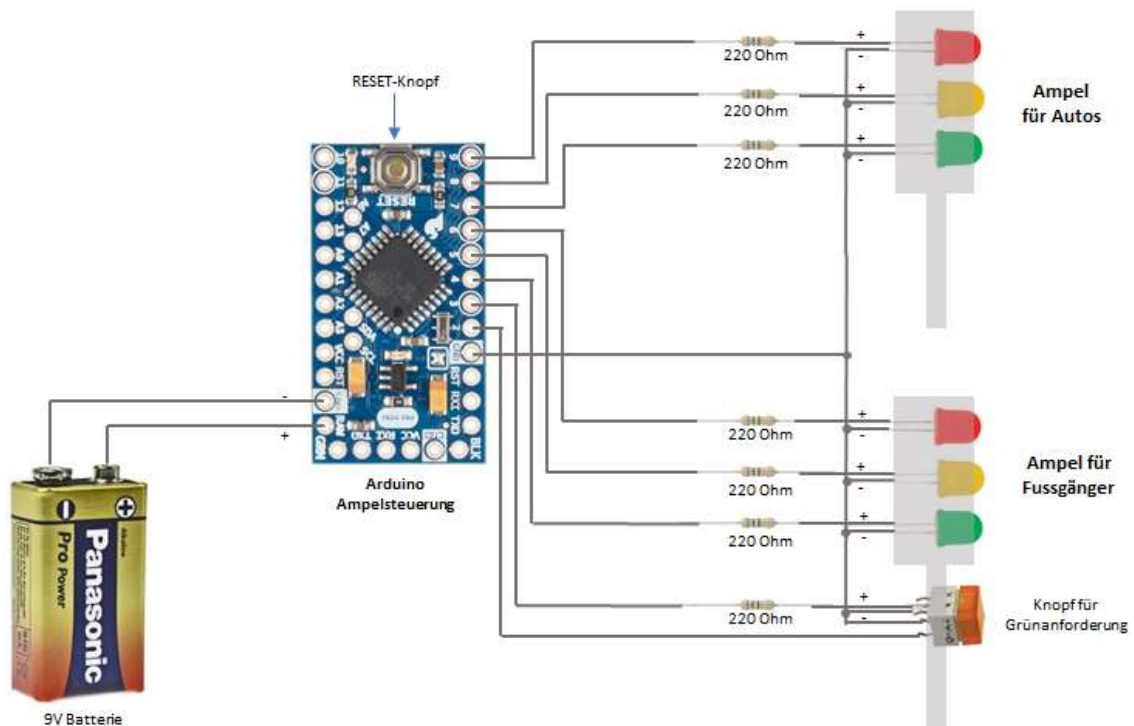
```

4 Bauanleitung

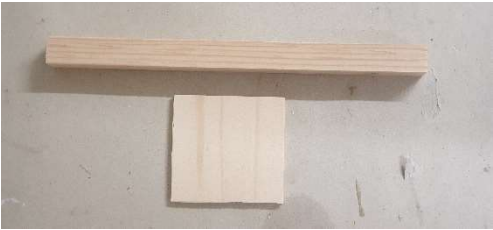


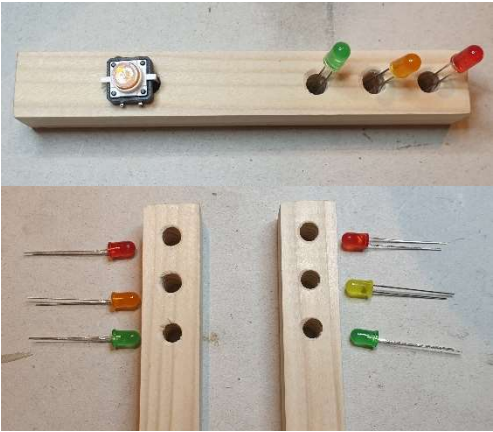
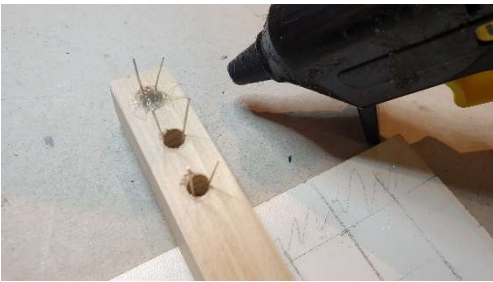
4.1 Schema

Auf dem folgenden Schema siehst du, wie die verschiedenen Teile miteinander über Kabel verbunden werden müssen.

Achtung: Plus (+) und Minus (-) immer beachten!



4.2 Arbeitsschritte

	<p>Zersäge die Holzleiste in zwei gleich lange Stücke und schleife alle Kanten. Schleife auch die Kanten des Grundbretts.</p>
	<p>Zeichne auf den Holzleisten an, wo du Löcher für die LEDs bohren willst. Miss den Durchmesser einer LED und suche den passenden Bohrer. Bohre die Löcher (insgesamt 6 Löcher).</p>
	<p>Zeichne auf einer Leiste weitere Löcher für den Drucktaster auf und bohre diese ebenfalls.</p>
	<p>Die Leiste für die Fussgängerampel sollte nun etwa so aussehen.</p> <p>Die Leiste für die Autoampel hat nur Löcher für die LEDs.</p>
	<p>Lege die LEDs in die Löcher. Unterlege die Leiste mit der Grundplatte, damit die LEDs vorne ein bisschen rausragen. Zuoberst kommt Rot, dann Gelb, dann Grün. Das kürzere Bein (-) muss bei allen LEDs auf der rechten Seite liegen. Klebe die LEDs von hinten mit Heissleim ein. Wiederhole das für die zweite Ampel.</p>

	<p>Knicke die kürzeren Beine (rechts) nach unten und verlöte alle drei Beine mit einem Draht wie auf dem Bild.</p> <p>Wiederhole das für die zweite Ampel.</p>
	<p>Kürze die Kabel etwa so wie auf dem Bild und ziehe sie durch die richtigen Löcher. Schau genau, welche Kabel einen Stecker haben.</p> <p>Die drei schwarzen Kabel werden am Ende zusammengelötet und bilden den Minuspol.</p> <p>Löte nun am Taster die Kabel an wie auf dem Bild.</p> <p>Die eingebaute LED braucht einen Vorwiderstand. Löte diesen direkt am Taster an, dort wo der Rote Punkt NICHT ist.</p> <p>Klebe den Taster vorsichtig mit Heissleim fest.</p>
	<p>Löte ein weiteres langes schwarzes Kabel (für die Autoampel) an den Minuspol und überziehe alle mit einem dicken Schrumpfschlauch.</p> <p>Nun werden die schwarzen Kabel an den Minuspol der LEDs gelötet.</p> <p>Anschliessend den Schrumpfschlauch überziehen.</p>
	<p>Löte einen 220 Ohm Vorwiderstand an das längere Bein von jeder der sechs LEDs beider Ampeln.</p> <p>Bereite drei kurze und drei lange Kabel mit einem Stecker vor zum Anschluss an die LEDs. Stülpe je einen Schrumpfschlauch über die Kabel. Das andere Ende des Widerstands verlötest du mit je einem Kabel. Die kurzen Kabel kommen an die Fussgängerampel, die langen an die Autoampel.</p> <p>Anschliessend die Schrumpfschläuche so weit wie möglich bis über die Widerstände ziehen.</p>

	<p>Überziehe das Batteriekabel mit zwei Schrumpfschläuchen.</p> <p>Löte nun beim Arduino das Batteriekabel an. Plus (rot) kommt an RAW, Minus (schwarz) and GND.</p> <p>Stecke die 9-polige Steckerleiste von oben in die Löcher 9, 8, 7, 6, 5, 4, 3, 2, GND und verlöte sie von unten.</p> <p>Schliesse nun alle Stecker in der richtigen Reihenfolge am Arduino an.</p> <p>Beachte genau das Schema auf Seite 15!</p> <p>Schliesse die Batterie an und teste, ob alles richtig funktioniert. Am Anfang müssen alle LEDs kurz aufleuchten, auch der Taster. Falls nicht alle LEDs leuchten, musst du nochmals alles kontrollieren.</p>
	<p>Wenn alles richtig funktioniert, schrumpfe alle Schrumpfschläuche mit dem Heissluftföhn fest.</p> <p>Befestige die Kabel mit Heissleim an der Ampel.</p> <p>Klebe auch den Arduino an der Fussgängerampel fest wie auf dem Bild.</p> <p>Mit einem zusätzlichen Kabelbinder kannst du das Batteriekabel mit den anderen Kabeln zusammenbinden.</p>
	<p>Leime beide Ampeln auf die Grundplatte.</p> <p>Fertig!</p>