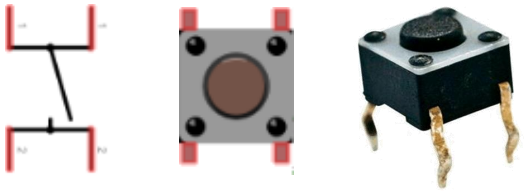
4.3. Aufgaben zu Schaltern

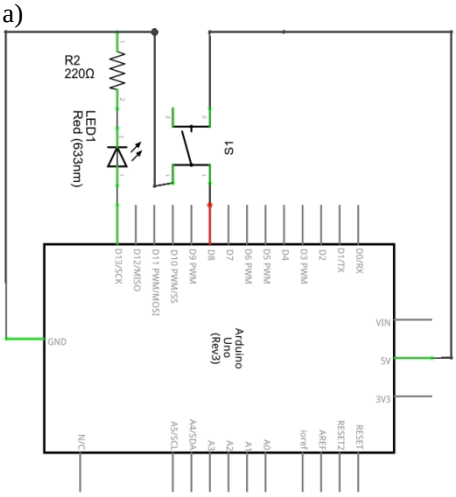
Aufgabe 1
Realisiere die Schaltung 4.3.1. mit dem Sketch 4.3.4, speichere ihn unter **Taster1.ino** und dokumentiere die Funktion mit einem Video.

Aufgabe 2
Vergleiche anhand der Anschlüsse in 4.3.1. die Lage des Tasters auf dem **Schaltplan (links)** und in der **Aufbauskizze (mitte)**. Wie unterscheidet sich die Lage des Tasters in den beiden Darstellungen? Betrachte nun das **Foto (rechts)** des Tasters. Er hat vier Beinchen, von denen je zwei leitend zu einem Paar verbunden sind. Markiere das sichtbare Paar auf dem Foto.

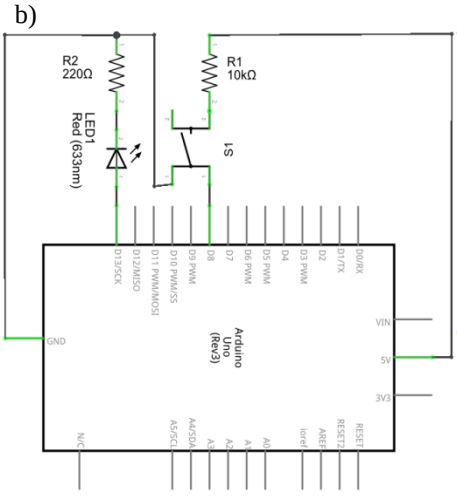


Aufgabe 3
Beschreibe jeweils in Worten, in welche (technische!) Richtung der Strom bei geöffnetem und bei geschlossenen Schaltern in den folgenden Anordnungen fließt.
Welche Schaltung zerstört das Board durch Kurzschluss, welche gibt unabhängig von der Schalterstellung das gleiche Signal und welche funktioniert einwandfrei, aber genau umgekehrt wie die Originalschaltung?

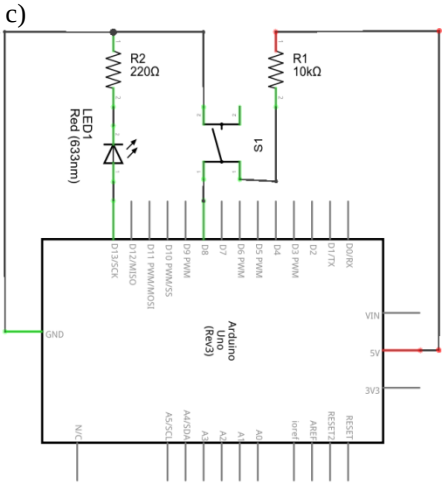
a)



b)



c)



Aufgabe 4
Vervollständige die Zustandstabelle für Aufgabe 1 c)

Schaltung	4.3.1. (Original)		Aufgabe 1c)	
Taster	geschlossen	offen	geschlossen	offen
Pin 8	HIGH	LOW		

Aufgabe 5
Erstelle (z.B. mit PAPDesign) einen Ablaufplan für den Sketch 4.3.4.

Aufgabe 6
a) Speichere den folgenden Sketch unter **Taster2.ino**, übertrage ihn und dokumentiere die Funktion mit einem Video.
b) Lösche die drittletzte Zeile `delay(200);`, kompiliere und übertrage. Erkläre das veränderte Verhalten der LED. Beachte, dass die Programmschritte in Millionstelsekunden durchlaufen werden. Wie lange dauert demgegenüber ungefähr ein Schaltvorgang?
c) Betrachte die fünftletzte Zeile `ZustandLED = !ZustandLED.` Was bedeutet ein vorangestelltes Ausrufezeichen in der Programmiersprache C? Warum kann man diesen Befehl nicht auf beliebige Variablen anwenden?

```
int LEDPin = 9;                    // LED auf 9
int TasterPin = 2;                 // Taster auf 2
int ZustandTaster;                 // Variable für den Tasterzustand
int ZustandLED;                    // Variable für den LED-Zustand
void setup()
{
  pinMode(LEDPin,OUTPUT);                 // LED als Ausgang
  pinMode(TasterPin,INPUT);               // Taster als Eingang
}
void loop()
{
  ZustandTaster = digitalRead(TasterPin);// Taster einlesen
```

```
  if(ZustandTaster == HIGH)              // Wenn Taster an, dann
  {
    ZustandLED = !ZustandLED;            // ZustandLED wechseln
    digitalWrite(LEDPin,ZustandLED);     // LED umschalten
    delay(200);                          // auf Taste warten
  }
}
```

Aufgabe 7

a) Kopiere die Sketche **Lauflicht1.ino** und **Taster1.ino** hintereinander in einen neuen Sketch. Benutze die kopierten Inhalte, um den folgenden Sketch zu erstellen. Speichere ihn unter **Taster3.ino**, übertrage ihn und dokumentiere die Funktion mit einem Video.

b) In Aufgabe 6 haben wir festgestellt, dass die Abfrage des Schalters nur funktioniert, wenn anschließend eine Pause `delay(200)` von ca. 200 ms eingelegt wird, um auf das Zurückschnappen der Mechanik zu warten. Wo sind die Pausen in diesem Sketch versteckt?

c) Die Funktion `void Lauflicht ()` wird ohne Eingabe (leere Klammer) deklariert, obwohl der Taster mittels `ZustandTaster = digitalRead(TasterPin);` abgefragt wird, was ja wohl eine **Eingabe** darstellt. Vergleiche mit der Funktion $f(x) = x^2$ und erkläre, warum die Abfrage des Tasters zwar eine Eingabe ist, aber **nicht für diese Funktion**.

d) Die Funktion `void Lauflicht ()` ohne Ausgabe (Typ `void`) deklariert, obwohl die Variable `ZustandLauflicht` geändert wird, was ja wohl eine **Ausgabe** darstellt. Vergleiche wieder mit der Funktion $f(x) = x^2$ und erkläre den Unterschied zwischen der **Funktion** `void Lauflicht()` und der **Variablen** `ZustandLauflicht`.

e) Welcher funktionelle Unterschied besteht zwischen der `For`-Schleife und der `while`-Schleife?

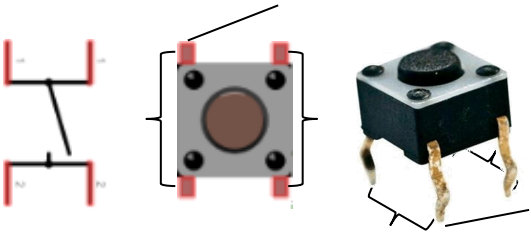
f) Zeichne (z.B. mit PAPDesign) einen Ablaufplan für **Taster3.ino**.

```
int TasterPin = 2;                  // Taster auf 2
int ZustandTaster;                  // Variable für den Tasterzustand
int ZustandLauflicht;               // Variable für den Lauflicht-Zustand
int LEDPin[] = {3,4,5,6,7,8,9,10}; // LED-Array mit Pin-Adressen
int Pause = 200;                    // Pause in ms
void setup()
{
  pinMode(TasterPin,INPUT);          // Taster als Eingang
  for(int i = 0; i < 8; i = i + 1)  // For-Schleife mit Zählvariable i
  {
    pinMode(LEDPin[i],OUTPUT);       // Pin i als Ausgang festlegen
  }
}
void Lauflicht()                     // Funktion Lauflicht mit Tasterabfrage
{
  for(int i = 0; i < 8; i = i + 1)         // For-Schleife mit Zählvariable i
  {
    digitalWrite(LEDPin[i],HIGH);          // Pin i anschalten
    delay(Pause);                          // Pause
    digitalWrite(LEDPin[i],LOW);           // Pin i ausschalten
    ZustandTaster = digitalRead(TasterPin);// Taster einlesen
    if(ZustandTaster == HIGH)              // Wenn Taster an, dann
    {
      ZustandLauflicht = !ZustandLauflicht;// Zustand Lauflicht wechseln
    }
  }
}
void loop()
{
  ZustandTaster = digitalRead(TasterPin);  // Taster einlesen
  if(ZustandTaster == HIGH)                // Wenn Taster an, dann
  {
    ZustandLauflicht = !ZustandLauflicht;  // Zustand Lauflicht wechseln
  }
  while(ZustandLauflicht == HIGH)          // Solange ZustandLauflicht an
  {
    Lauflicht();                           // Lauflicht einschalten
  }
}
```

4.3. Lösungen zu den Aufgaben zu Schaltern

Aufgabe 1
http://www.poenitz-net.de/Informatik/4.Mikrocontroller/4.3.Taster1.mp4

Aufgabe 2
Der Taster ist in der Aufbauskizze um 90° gedreht:



Aufgabe 3

a) Schalter geschlossen: Der Strom läuft ohne Widerstand von 5 V über Pin D 8 zu GND ⇒ **Kurzschluss** ⇒ Board zerstört!
Schalter offen: D8 mit GND verbunden ⇒ LOW; normale Steuerung der LED über D 13.

b) Schalter geschlossen: D8 über 10 kΩ-Vorwiderstand mit 5V und gleichzeitig ohne Widerstand mit GND verbunden ⇒ LOW
Schalter offen: D8 mit GND verbunden ⇒ **ebenfalls** LOW; normale Steuerung der LED über D 13; siehe a).

c) Schalter geschlossen: D8 über 10 kΩ-Vorwiderstand mit 5V und gleichzeitig ohne Widerstand mit GND verbunden ⇒ LOW; siehe b)
Schalter offen: D8 über 10 kΩ-Vorwiderstand mit 5V verbunden. Da bei der Kompensationsspannung 5 V kein Strom fließt, fällt am Vorwiderstand keine Spannung ab ⇒ Messwert 5 V ⇒ HIGH. ⇒ funktioniert genau umgekehrt wie die Originalschaltung!

Aufgabe 4
siehe rechts

Aufgabe 5
siehe rechts

Schaltung	4.3.1. (Original)		Aufgabe 1c)	
Taster	geschlossen	offen	geschlossen	offen
Pin 8	HIGH	LOW	LOW	HIGH

Aufgabe 6

a) http://www.poenitz-net.de/Informatik/4.Mikrocontroller/4.3.Taster2.mp4

b) Der Schalter wird auch bei sehr kurzer Betätigung einige Hundertstelsekunden geschlossen bleiben. In dieser Zeit wird das komplette Programm einige zehntausend Mal durchlaufen und der Inhalt der Variable `ZustandLED` jedesmal geändert bzw. die LED an- und ausgeschaltet. Der Endzustand ist dann völlig zufällig und unberechenbar.

c) Das vorangestellte Ausrufezeichen bedeutet die Umkehrung des Variableninhaltes und kann eigentlich nur bei logischen bzw. binären Variablen vom Typ `boolean` funktionieren, welche nur die beiden Werte TRUE (1 bzw. an) und FALSE (0 bzw. aus) kennen. Die Schlüsselworte HIGH bzw. LOW führen beim Arduino-Compiler offensichtlich dazu, dass die `integer`-Variable `ZustandTaster` als `boolean` gedeutet wird.

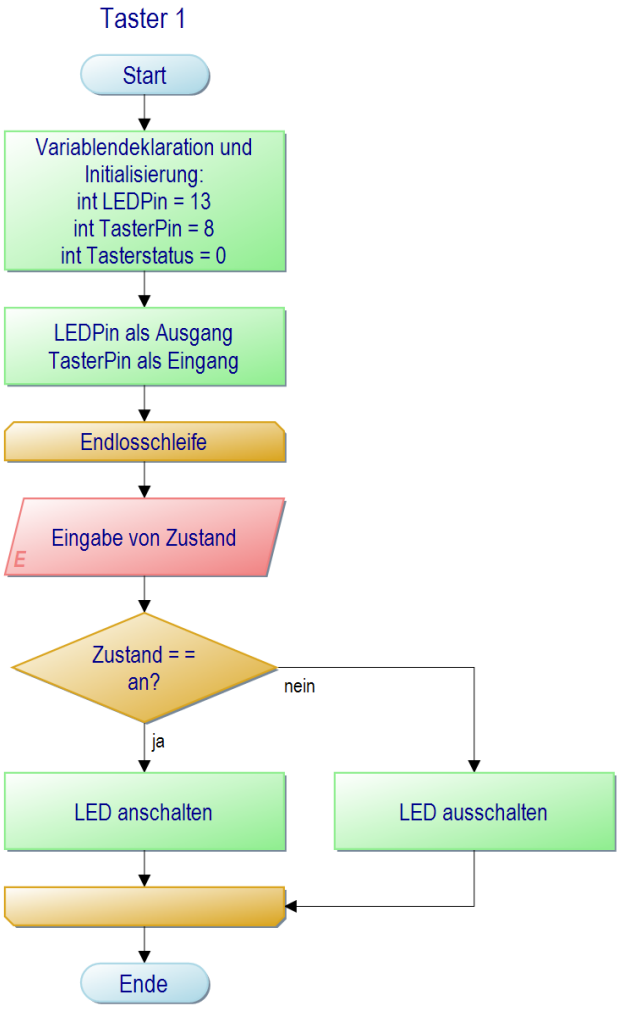
Aufgabe 7

a) http://www.poenitz-net.de/Informatik/4.Mikrocontroller/4.3.Taster3.mp4

b) Die Pausen in der For-Schleife erfüllen einen doppelten Zweck. Sie lassen die LED so lange an, dass sie für das menschliche Auge sichtbar werden und ermöglichen das Zurückschnappen der Schaltermechanik vor dem nächsten Einlesen.

c) Der Ablauf bzw. die Ausgabe der Funktion `void Lauflicht ()` ist unabhängig vom Wert der Variable `ZustandTaster`. Sie erscheint daher nicht in den Argumentklammern.

d) Die Variable `ZustandLauflicht` legt fest, **ob** die Funktion `ZustandLauflicht()` beim nächsten



Durchlauf ausgeführt wird. Sie hat keinen Einfluss darauf, **wie** die Funktion ausgeführt wird und hat daher nichts mit ihrem Ergebnis zu tun.

e) Während die For-Schleife eine vorher festgelegte Anzahl von Wiederholungen durchläuft, kann die while-Schleife von beliebigen Ereignissen innerhalb oder außerhalb der Schleife gesteuert werden.

f) siehe unten. Die Funktionen (Unterprogramme) `void setup()` und `void Lauflicht()` sind getrennt gezeichnet