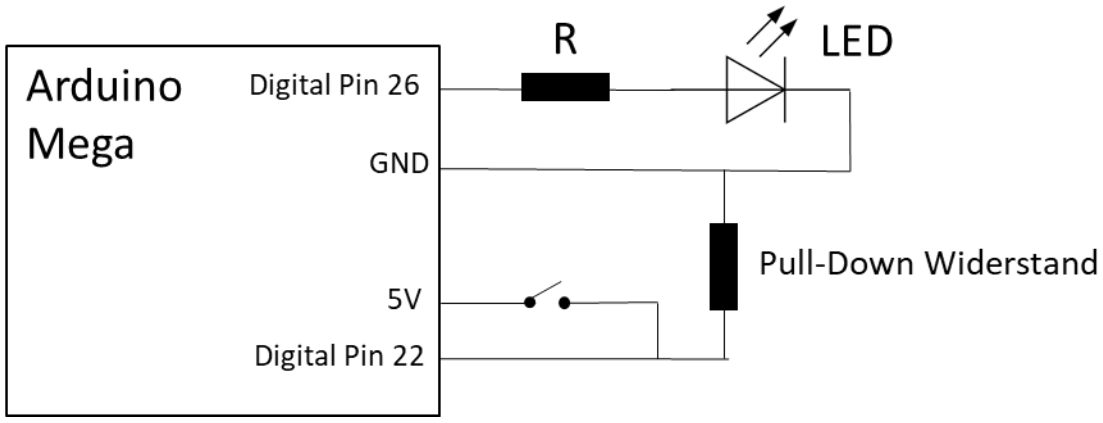
Übung 02: Digitale Ein- und Ausgabe, GPIO

Aufgabe 1: Digitale Ein- und Ausgabe mit Arduino Library

a) Die finale Verkabelung ist unten abgebildet. Unbedingt darauf achten, dass bei einem Steckbrett einige Löcher leitend miteinander verbunden sind, siehe auch Vorlesung 01.

b) Der Code konfiguriert Digital Pin 26 zunächst als Ausgang, Digital Pin 22 als Eingang. Das muss nur einmal gemacht werden, deshalb ist die Setup-Routine geeignet.
In der Endlosschleife wird in der if-Bedingung geprüft, ob die Spannung an Digital Pin 22 (Button) gerade HIGH ist. Das ist gleichbedeutend damit, dass der Taster gedrückt wird. Falls ja, wird der Zustand der LED geändert. Das kann man trickreich machen, in dem man einfach den aktuellen Zustand der LED ausliest und diesen invertiert.

c) Vermutlich wird das beobachtete Verhalten nicht korrekt sein. Es kann sein dass die LED unkontrolliert flackert, dauerhaft an oder aus ist, unabhängig davon, ob man den Schalter drückt oder nicht.
Problematisch ist die Situation wenn der Schalter offen ist, also nicht gedrückt wird. Dann wäre der Spannungspegel am Digital Pin 22 undefiniert. Deshalb muss man dafür sorgen, dass in diesem Fall, der Spannungspegel am Digital Pin 22 auf LOW gezogen werden: Ein Pull-Down Widerstand wird eingesetzt, siehe Schaltplan.



d) Vermutlich wird man beobachten, dass nicht jeder Tastendruck den Zustand der LED ändert. Der Grund ist ein nicht prellungsfreier Taster. Der Taster ist ein mechanisches Element, das hin- und herschwingt. Deshalb kann der Pegel an Digital Pin 22 beim Drücken des Tasters kurz auf HIGH gehen, dann nochmal kurz auf LOW bevor er sich dann final auf HIGH einschwingt. Das muss man soforttechnisch abfangen, z.B. durch ein Delay von 200 ms, siehe Source Code.

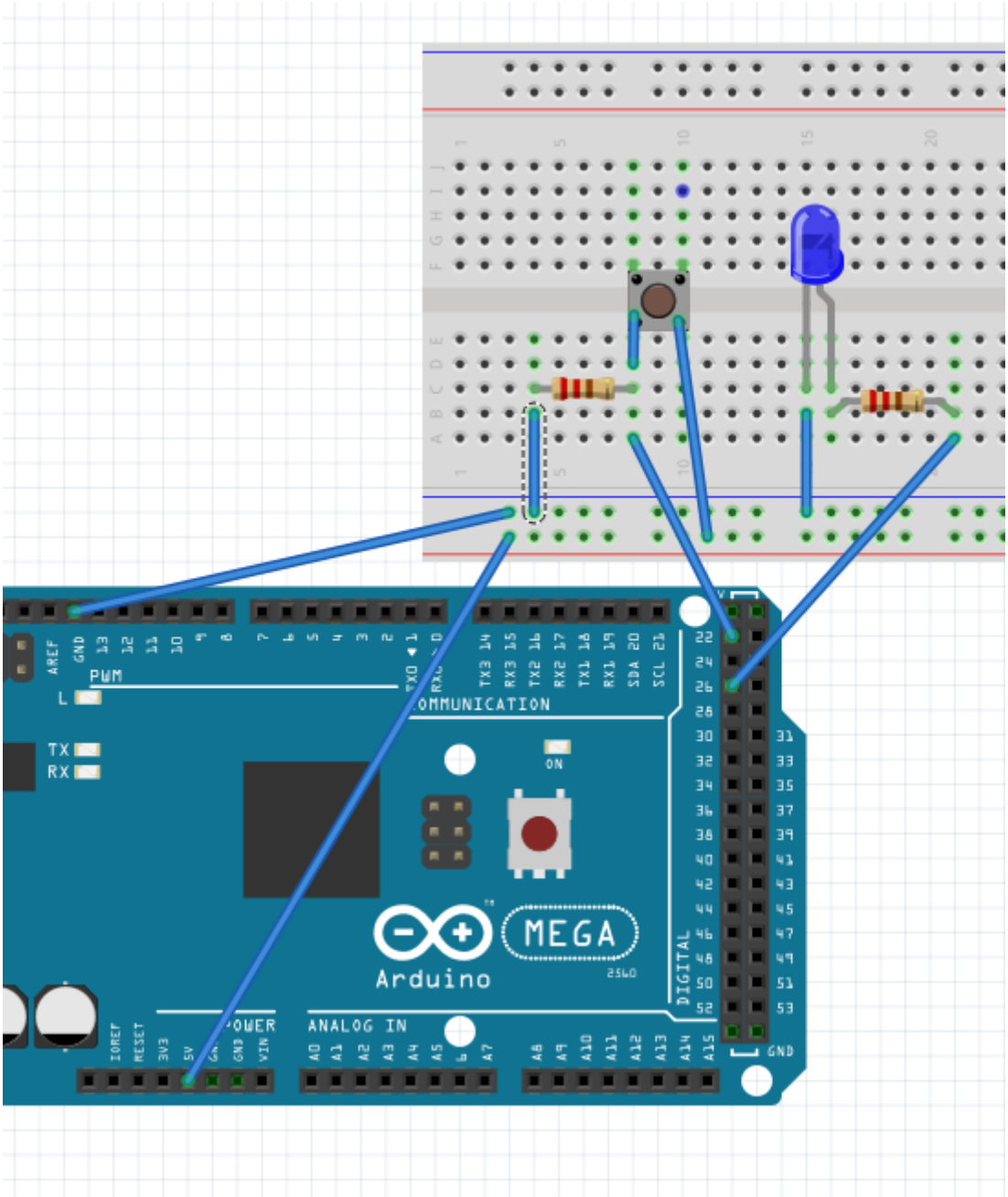
```
#include <Arduino.h>

int led = 26;                 // Digital pin 26,
int button = 22;           // digital pin 22

void setup() {
      pinMode(led, OUTPUT);
      pinMode(button, INPUT);
}
```

```
void loop() {
	if (digitalRead(button) == HIGH) {
		digitalWrite(led, !digitalRead(led));
		delay(200);
	}
}
```

Finale Verkabelung:



Aufgabe 2: Digitale Ein- und Ausgabe mit AVR Libc

a) https://www.arduino.cc/en/Hacking/PinMapping2560
Am besten verwendet man die Tabelle und schaut an, was in der 2. Spalte steht.
Digital Pin 22 / Button: Port A, Pin #0 (PA0)
Digital Pin 26 / LED: Port A, Pin # (PA4)
Die 1. Spalte der Tabelle gibt an, welche Pinnummer des IC-Bausteines der jeweilige Ausgang hat.
Am besten man schaut sich mal Seite 2 des Datenblattes an.

b) Siehe Source Code, Hinweise:
- Achten Sie darauf für Eingabe das PIN, für Ausgabe das PORT Register zu verwenden.
- Machen Sie sich mit den Bitoperationen vertraut.
- Das 2. Statement im setup(.) könnte man weglassen, da normalerweise die moisten Register auf 0x00 vorinitialisiert sind.
- Verknüpft man ein Bit mit XOR 1, so „toggelt“ das.
- Das „Toggeln“ kann man natürlich auch analog zu Aufgabe 1 umsetzen. Also erst den aktuellen Zustand der LED über das Pin Register auslesen:

```
if (PINA & (1 << PA4) {   // LED gerade an
      ➔ ausschalten
else {   // LED gerade aus
      ➔ anschalten
}
```

- Es gibt vordefinierte Makros für alle Registernamen und Bits der Register, siehe Datenblatt Kapitel 33, Seite 399.

```
#include <Arduino.h>

int led = 26;             // Digital pin 26, Port A, Pin 4
int button = 22;          // digital pin 22, Port A, Pin 0

void setup() {
      DDRA |=   (1 << DDA4);   // LED output
      DDRA &= ~ (1 << DDA0);   // button input
}

void loop() {
      if (PINA & (1 << PA0)) {   // button pin / digital pin 22 is high
            PORTA ^= (1 << PA4);  // XOR 1: trick to toggle LED!
            delay(200);
      }
      delay(200);
}
```

Aufgabe 3: Entfernungsmesser [optional, Vertiefung von Aufgabe 2]

a) Siehe: https://www.arduino.cc/reference/de/language/functions/advanced-io/pulsein/
Misst die Zeit bis am Eingang LOW anliegt.
Über den Trigger-Pin wird eine Messung angestoßen. Über den Echo-Pin wird der Mikrocontroller informiert, dass die reflektierte Schallwelle angekommen ist.

b) Siehe Verkabelung auf Angabe.

c) Trigger Pin: Digital Pin 7 -> PH 3 (Port H, Pin #4)
Echo Pin: Digital Pin 6 -> PH 4 (Port H, Pin #3)
sdfdsf

```
#include <Arduino.h>

long entfernung=0;

void setup() {
      Serial.begin (9600);
      DDRH |= (1 << DDH4);    // trigger pin
      DDRH &= ~(1 << DDH3);   // echo pin
}

void loop() {
      PORTH &= ~(1 << PH4);
      delay(5);
      PORTH |= (1 << PH4);
      delay(10);
      PORTH &= ~(1 << PH4);
      long elapsedTime = pulseIn(echo, HIGH);
      long distance = (elapsedTime/2) * 0.03432;
      if (distance >= 500 || distance <= 0)  {
            Serial.println("Kein Messwert");
      } else       {
            Serial.print(distance);
            Serial.println(" cm");
      }
      delay(1000);
   }
```