Übung 02: Digitale Ein- und Ausgabe, GPIO

Hardware:

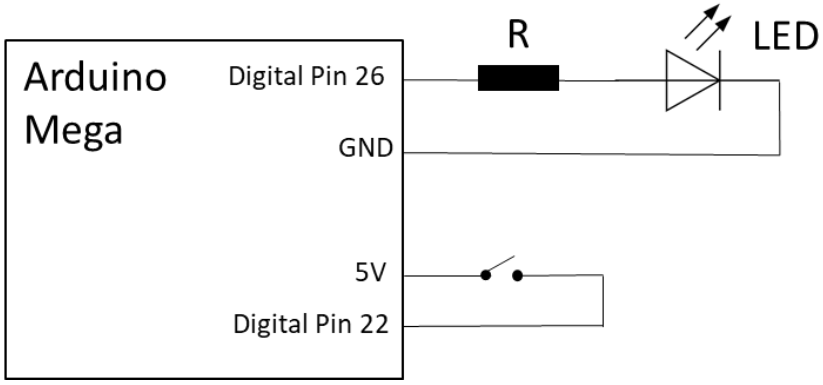
- Basis: Arduino, Steckbrett, Kabel
- Blaue LED
- Widerstände: 100 Ohm, 1 kOhm
- Taster
- Ultraschallsensor HC-SR04

Informationen:

- Datenblatt, Kap.13: I/O Ports
- Arduino Language Reference: https://www.arduino.cc/en/Reference/HomePage
- Arduino Pin Mapping: https://www.arduino.cc/en/Hacking/PinMapping2560

Aufgabe 1: Digitale Ein- und Ausgabe mit Arduino Library

Toggeln: Bei geschlossenem Schalter soll der Zustand der LED geändert werden. Ist die LED gerade ausgeschaltet, wird sie angeschaltet. Ist die LED gerade angeschaltet, wird sie ausgeschaltet.



a) Bauen Sie die Schaltung auf dem Steckbrett nach! Verwenden Sie *Digital Pin 26*, *Digital Pin 22*, *GND* und 5*V*, siehe Skizze.

b) Erklären Sie den rechten Code. Testen Sie den Code auf dem Mikrocontroller. Funktioniert das „Togglen“ bereits korrekt?

c) **Pull-Up / Pull-Down:** Fügen Sie der Schaltung einen 1 kOhm Widerstand hinzu, so dass bei geschlossenem Schalter am Digital Pin 22 `HIGH` (LOW) anliegt. Bei offenem Schalter soll `LOW` Anliegen. Aktualisieren Sie auch den Schaltplan.

```
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));
   }
}
```

d) **Entprellung:** Schaltet jeder Tastendruck Ihre LED auch wirklich um? Erklären!
Bauen Sie eine Verzögerung von ca. 200 ms in Ihr Programm ein, um korrektes Verhalten zu erzielen. https://www.mikrocontroller.net/articles/Entprellung

Aufgabe 2: Digitale Ein- und Ausgabe mit AVR Libc

a) Mit welchem ***Pin*** und ***Port*** des Mikrocontrollers ist die Buchse Digital Pin 22 und Digital Pin 26 des Mega Boards verbunden? Hinweis: Pin Mapping

b) Schreiben Sie das Programm aus Aufgabe 1 um. Wichtige Einschränkung: Sie dürfen die folgenden Kommandos **nicht verwenden**: `pinMode(.)`, `digitalRead(.)` und `digitalWrite(.)`. Ersetzen Sie diese! *Hinweise*:
- Sie müssen direkt Register programmieren! Relevante Register sind `DDRX`, `PORTX` und `PINX`, wobei das „X“ jeweils durch den korrekten Buchstaben zu ersetzen ist.
- Versuchen Sie nur die benötigten Bits zu modifizieren.

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

Um eine Entfernungsmessung anzustoßen, sendet der Mikrocontroller ein kurzes elektrisches Signal an den Sensor, wodurch eine Schallwelle vom Ultraschall-Entfernungsmesser ausgelöst wird. Sobald die Schallwelle gegen eine Wand oder ein sonstiges Hindernis stößt, wird sie reflektiert und kommt irgendwann auch wieder zum Ultraschallsensor zurück. Sobald der Sensor diese zurückgekehrte Schallwelle erkennt, sendet der Sensor ein spezielles Signal zurück an den Mikrocontroller.

Der Mikrocontroller misst die Zeit zwischen dem Aussenden und der Rückkehr der Schallwelle und rechnet diese Zeit dann in eine Entfernung um.

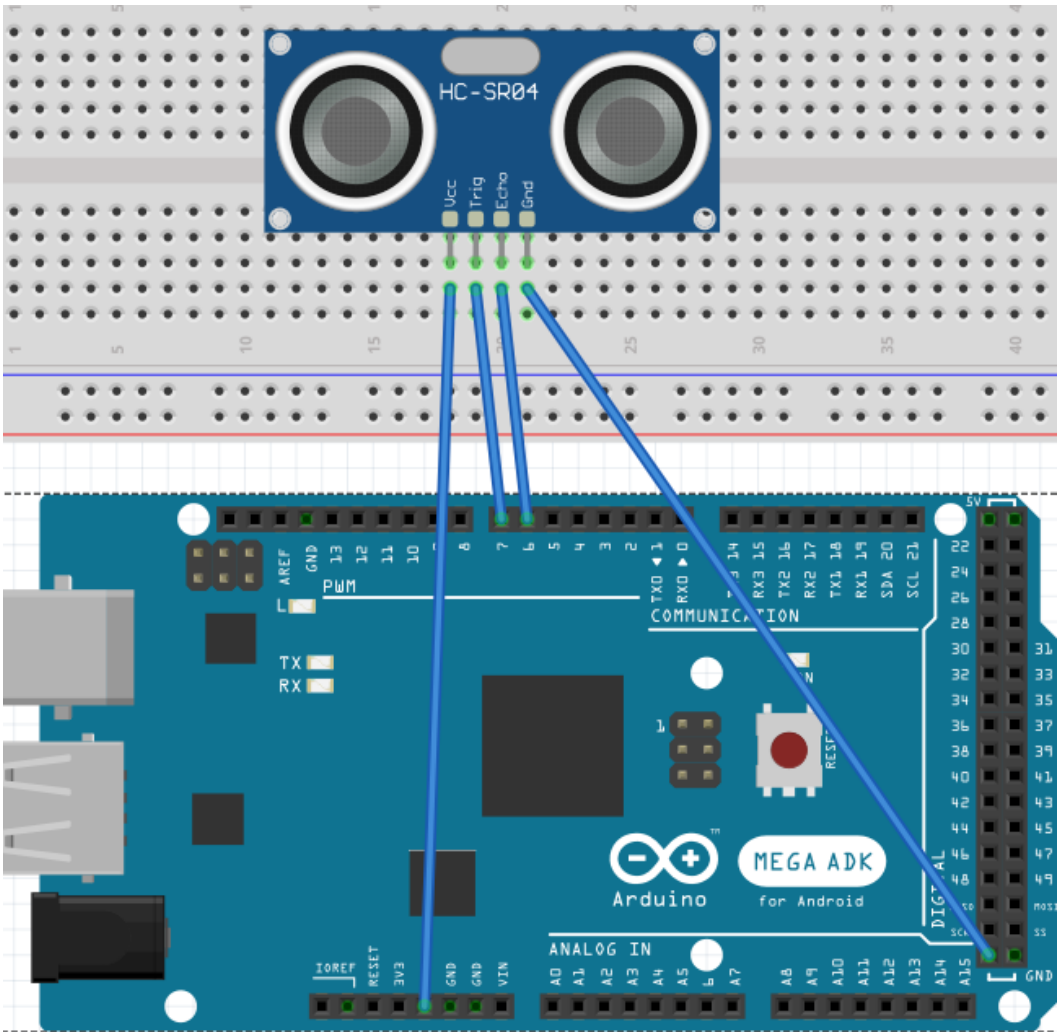
Anschlüsse:

Mikro-controller	Entfernungs-
5V	VCC
7	Trig
6	Echo
GND	Gnd

```
#include <Arduino.h>

int trigger=7;
int echo=6;

long entfernung=0;

void setup() {
  Serial.begin (9600);
  pinMode(trigger, OUTPUT);
  pinMode(echo, INPUT);
}

void loop() {
  digitalWrite(trigger, LOW);
  delay(5);
  digitalWrite(trigger, HIGH);
  delay(10);
  digitalWrite(trigger, LOW);
  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);
}
```

a) Versuchen Sie den Code zu verstehen! Was macht die Arduino-Funktion `pulseIn(.)`?
b) Bauen Sie die Schaltung auf, Verkabelung, siehe nächste Seite. Testen Sie mir der seriellen Konsole.
c) Wie Aufgabe 2: Modifzieren Sie den Code. **Kein** `pinMode(.)`, `digitalRead(.)` und `digitalWrite(.)`! `pulseIn(.)` dürfen Sie verwenden.

(umblättern)


HC-SR04
Vcc
Trig
Echo
Gnd
PWM
COMMUNICATION
TX
RX
Arduino
MEGA ADK
for Android
ANALOG IN
IOREF
RESET
3V3
GND
VIN