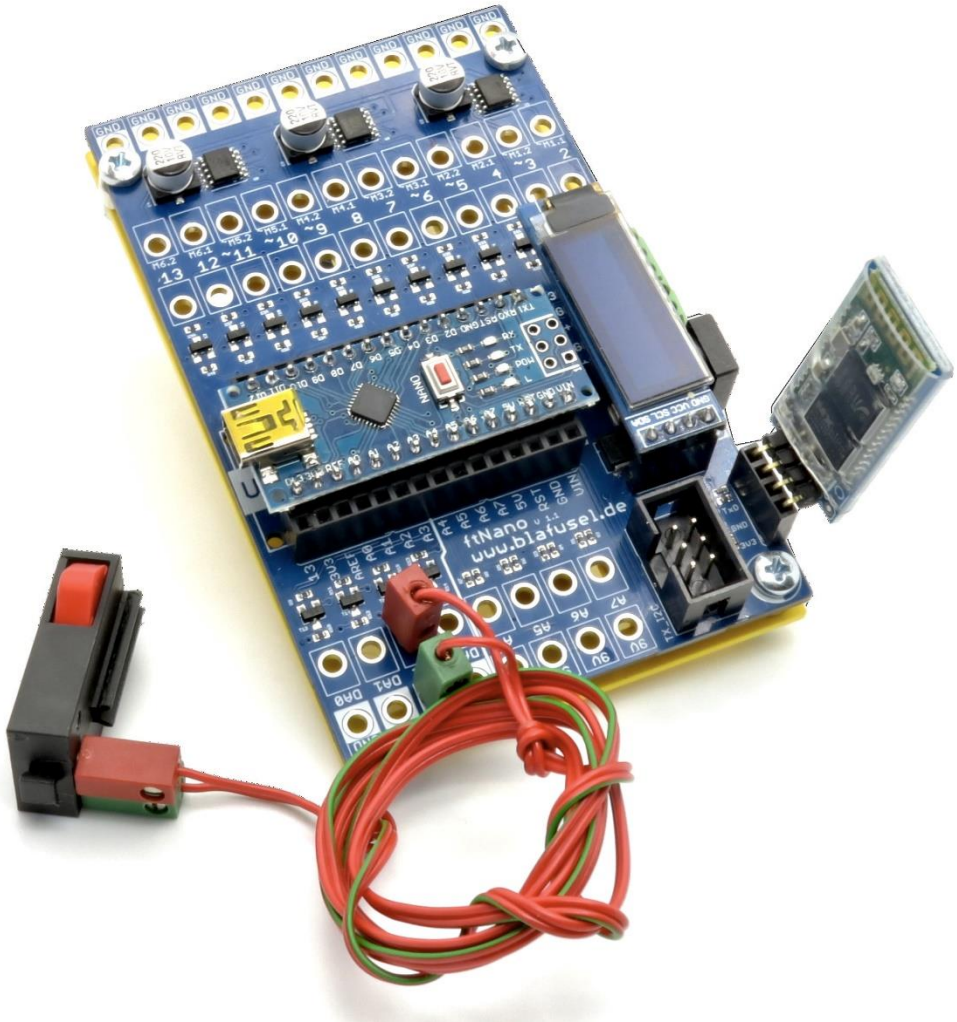
ftNano



Benutzerhandbuch für Version 1.1

Florian Schäffer
https://www.blafusel.de/
https://www.obd2-shop.eu/

1.0 | September 2024

ISBN: 9783754348796

Inhaltsverzeichnis

1 Einführung

Konstruktionsbausteine von Fischertechnik eignen sich gut, um technische Funktionsmodelle aufzubauen. Diese können über elektronische Aktoren (bspw. Lampen und Motoren) sowie Sensoren (bspw. Taster) verfügen. Mit diesen ist bereits eine einfache Steuerung des Modells möglich. Erweiterte Möglichkeiten ergeben sich, wenn Elektronik, ein Computer oder Mikrocontroller das Modell steuert.

Alternativ zu den von Fischertechnik angebotenen Controllern bietet der ftNano die Verbindung der Modelle mit dem beliebten Arduino-Entwicklungsboard. Dadurch stehen alle Funktionen zur Verfügung, die das einsteigerfreundliche Arduino-Konzept bietet, wie zum Beispiel:

- Arduino Nano
- Programmierung in C/C++ (Arduino IDE) oder Scratch (Nepo/OpenRoberta Lab) usw.
- Keine speziellen Treiber oder Libraries notwendig
- Einbinden von zahlreichen Sensoren (bspw. Helligkeit, Temperatur, Entfernung) aus dem Arduino-Universum, die sonst nicht unterstützt werden
- Anschluß von sogenannten *Shields* auf denen beliebte Module etc. fertig aufgebaut sind (z. B.: SD-Speicherkarten oder LC-Displays, I²C, SPI)
- Kommunikation über sämtliche Netzwerktechniken wie WLAN, Bluetooth usw.

Diese Anleitung setzt voraus, daß der Benutzer mit der Arduino-Programmierung in C/C++ mittels Arduino-IDE sowie dem Arduino an sich vertraut ist. Ebenfalls werden Elektronikkenntnisse vorausgesetzt. Für die Darstellung von Schaltungen wird Fritzing benutzt, womit der Benutzer ebenfalls vertraut sein sollte, wenn er damit arbeiten möchte.

Der ftNano ist so konzipiert, daß er vor allem die unterschiedlichen Spannungen, die bei Fischertechnik und Mikrocontrollern wie dem ATmega (der eigentliche Mikrocontroller bei Arduino-Boards) anpaßt und zueinander kompatibel macht. Zudem wird für Motoren und Lampen ein größerer Strom benötigt, als es Mikrocontroller liefern können.

Dabei befindet sich auf dem ftNano-Board kein Mikrocontroller. Das Board enthält alle notwendigen Komponenten, um auf die vorgesehenen Steckplätze einen handelsüblichen Arduino Nano ohne Löten einfach aufzustecken. Dadurch wird es einfach, den Arduino zu wechseln, sollte dieser beispielsweise beschädigt werden. Außerdem werden die Kosten gesenkt.

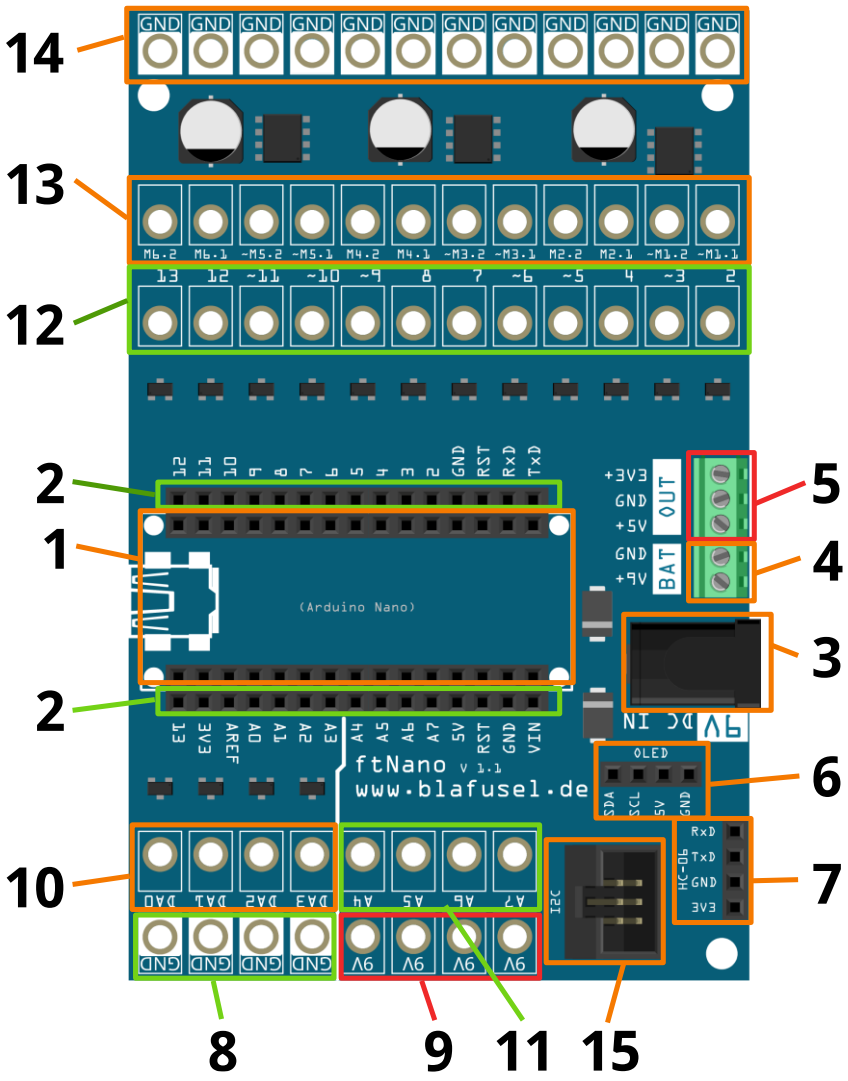
Bewußt wurde darauf geachtet, den ftNano so zu konstruieren, daß keine zusätzlichen Bibliotheken für die Programmierung erforderlich sind und sich am Konzept der Programmierung mit der Arduino-IDE und der Benutzung von I/O-Anschlüssen nichts ändert. Die Programmierung und Nutzung sollen so möglichst der üblichen Arduino-Technik entsprechen. Eine Nutzung und Programmierung mit Programmiersprachen für die es keine spezielle Bibliothek gibt (bspw. grafische Sprachen wie Scratch (https://s4a.cat) oder OpenRoberta (https://lab.open-roberta.org ist deshalb problemlos möglich. Außerdem sind alle Erweiterungen (Shields) etc. die es für den Arduino gibt uneingeschränkt nutzbar, da sämtliche Pins des Arduino Nano über zwei Buchsenleisten zugänglich sind.

Wichtige Merkmale des ftNano:

- bis zu 16 digitale I/O kompatibel zu 9 V/20 mA
- bis zu 6 digitale Leistungsausgänge 9 V mit je bis zu 200 mA inkl. Schutz gegen Rückwärtsspannungen (induktive Lasten)
- Alle digitalen I/Os mit MOSFETs ausgestattet (präziser, schneller Pegelwechsel, Interruptfähigkeit entsprechend ATmega328)
- 4 analoge Eingänge 0 V – 10 V
- Buchsenleiste für OLED-Display
- Buchsenleiste für HC-05/HC-06 Bluetooth-Modul
- Wannenstecker für I²C
- Buchsenleisten für sämtliche I/O-Pins des Arduino Nano

2 Funktionsübersicht

Der ftNano bietet folgende Anschlüsse:

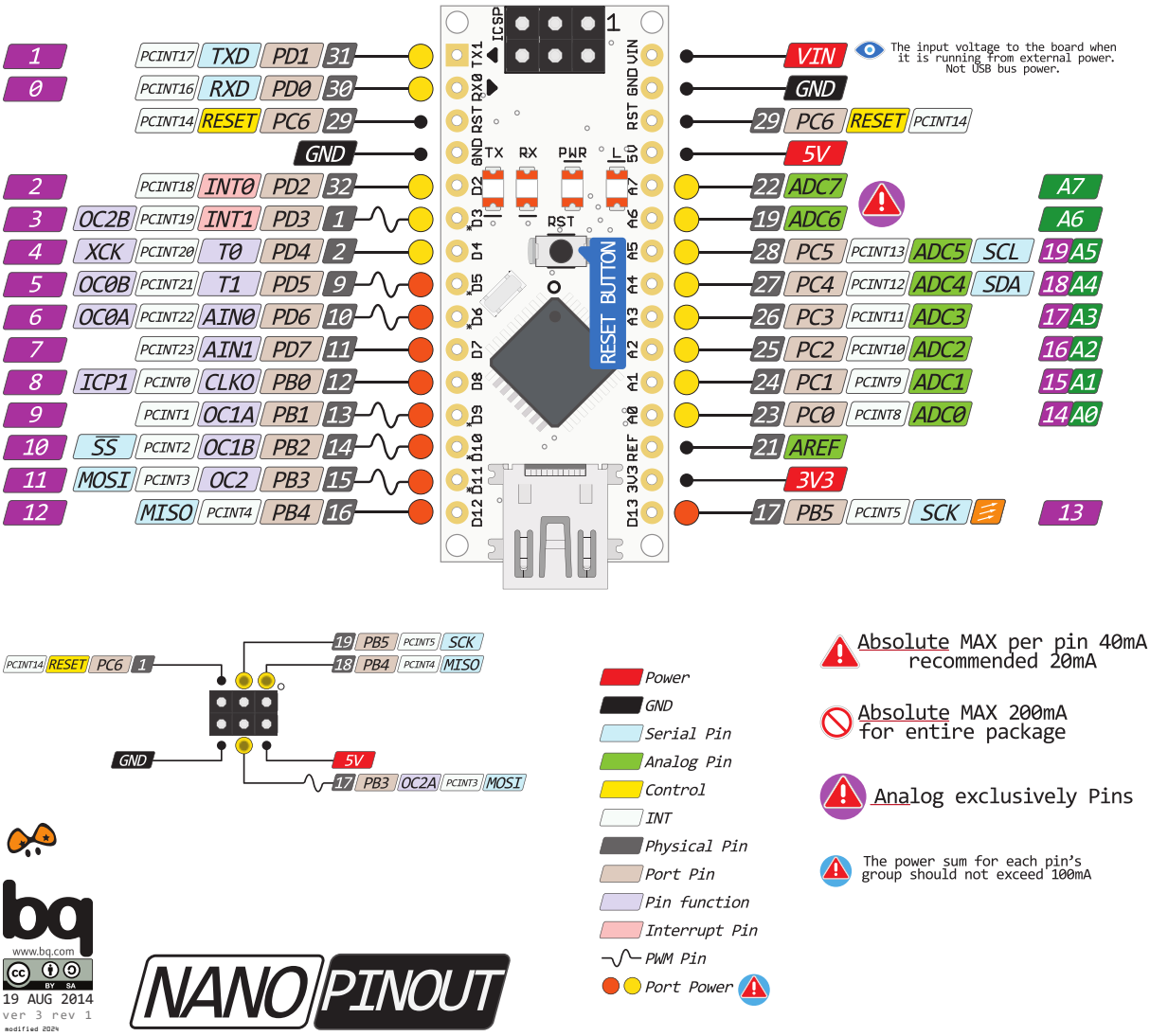


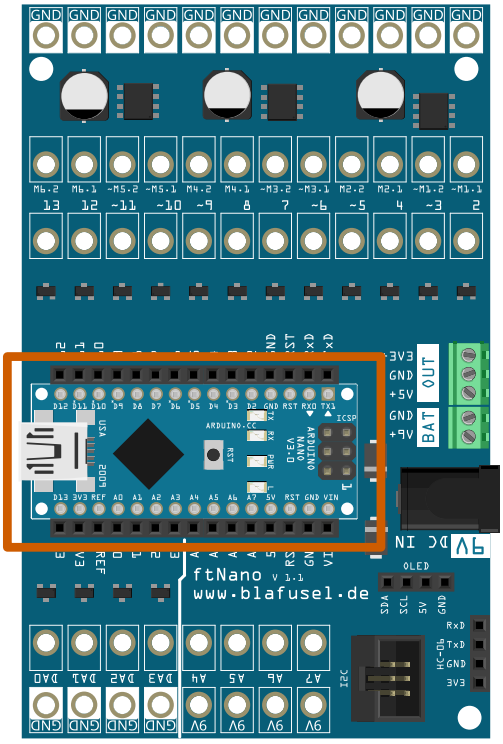
1. Steckplatz für Arduino Nano
2. Buchsenleiste für Arduino-Pins
3. DC Eingang Hohlbuchse
4. DC Eingang Schraubklemmen
5. Schraubklemmen Spannungsausgang
6. Buchsenleiste für OLED-Display
7. Buchsenleiste für Bluetooth-Modul
8. Fischertechnik-Kontakt GND/Masse
9. Fischertechnik-Kontakt 9 V (VCC)
10. Fischertechnik-Kontakt digitale I/O
11. Fischertechnik-Kontakt analoge I/O
12. Fischertechnik-Kontakt digitale I/O
13. Fischertechnik-Kontakt digitale Last-Ausgänge
14. Fischertechnik-Kontakt GND/Masse
15. Wannenstecker für I²C

2.1 Steckplatz für Arduino Nano

In diese Buchsenleisten wird der Arduino Nano eingesetzt. Die USB-Buchse weist wie im Bestückungsdruck zu sehen, nach links. https://store.arduino.cc/products/arduino-nano

Über die USB-Buchse des Arduino wird der Mikrocontroller programmiert und der Arduino mit Spannung versorgt. Diese Spannung liegt auch am ftNano an. Der Arduino kann nicht genügend Strom für die Versorgung von Fischertechnik-Bauteilen etc. liefern. Aus diesem Grund ist eine (zusätzliche) Spannungsversorgung über einen der DC-Eingänge notwendig.





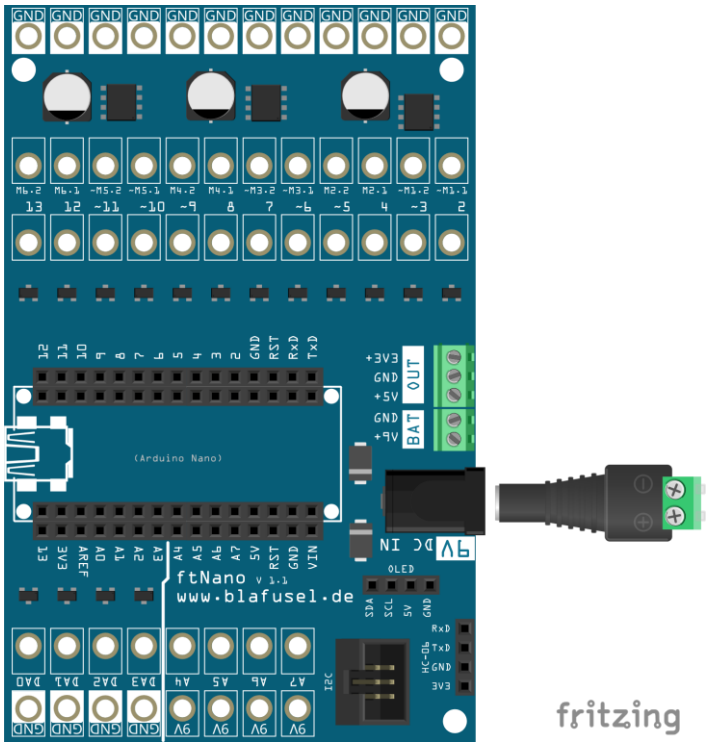
2.2 Buchsenleisten für Arduino-Pins

Über die zwei Buchsenleistenreihen besteht die Möglichkeit, mit Jumper-Kabeln direkten Kontakt zu den I/O-Pins des Arduinos herzustellen. Bei direkter Nutzung der Arduino Nano-Anschlüsse gelten die Parameter zu Strom und Spannung für den Arduino Nano: https://docs.arduino.cc/hardware/nano

2.3 DC Eingang Hohlbuchse

Über die Hohlbuchse kann der ftNano (sowie der Arduino) mit Spannung versorgt werden.

Der ftNano muß immer mit einer externen Spannungsquelle betrieben werden. Andernfalls kommt die Versorgungsspannung von der USB-Buchse über den Arduino, was zu Zerstörung von Arduino und/oder dem USB-Anschuß im PC führen kann, wenn zu viel Strom fließt und die Spannungsregler überlastet werden.



Die Eingangsspannung darf **7–12 Volt** betragen (bzw. gemäß Spezifikation Arduino Nano). **Die Spannung darf nicht größer sein**, da ansonsten der Arduino und der ftNano beschädigt werden. Für Fischertechnik werden 9 V Gleichspannung benötigt. Es ist vorab sicherzustellen, daß ein geeignetes Netzteil bzw. Spannungsquelle benutzt wird. Die Spannung sollte möglichst frei von Störsignalen und Spannungsschwankungen sein. Die Leistung hängt von der benutzten externen Komponenten ab, die vom ftNano gesteuert werden. Eine Ausgangsleistung von 2 A dürfte für die meisten Anwendungen ausreichend sein. Am besten eignen sich handelsübliche Schaltnetzteile (Kurzschlußfest und mit Überlastschutz) in Form von Steckern für die Steckdose für um die 10 €.

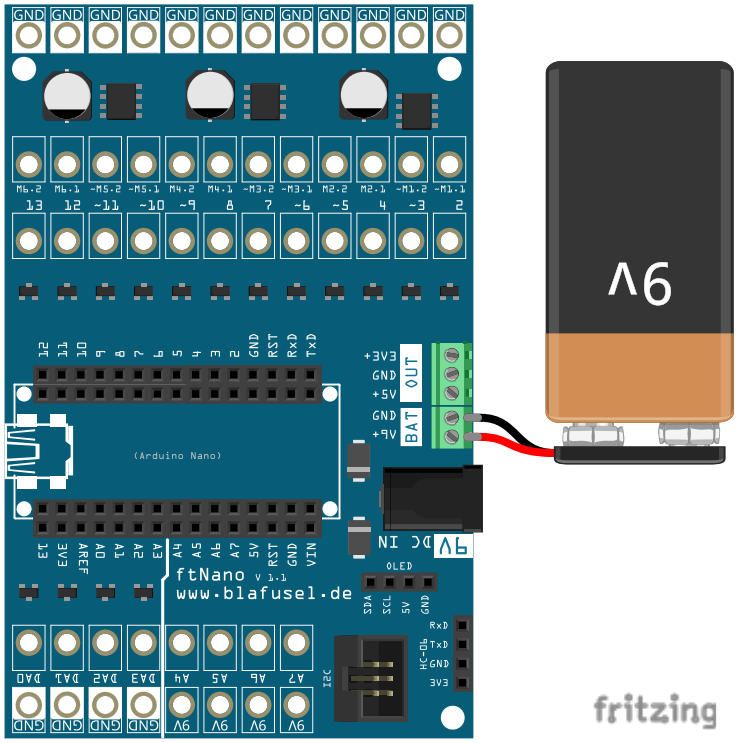
Festspannungsnetzteil

Die DC-Buchse ist passend für Stecker mit den Maßen 5,50 mm × 2,10 mm. Der innere Stift ist VCC (Pluspol) und Außen GND/Masse. Der ftNano ist gegen Verpolung geschützt. Die Hohlbuchse und die Schraubklemmen sind über Dioden (ca. 0,3 V Vorwärtsspannung) gegeneinander geschützt, so daß bspw. ein Akku und ein Netzteil gleichzeitig angeschlossen sein kann, ohne daß der Akku geladen wird.

Je nach Modell und internem Aufbau kann es sein, daß der aufgesteckte Arduino Nano (minderwertige Nachbauten) eine externe Spannung an die USB-Buchse führt. Dies kann zur Zerstörung der USB-Hardware im PC führen. Es empfiehlt sich daher, den Arduino nur dann mit USB zu verbinden, wenn keine externe Spannung am ftNano anliegt. Einen guten Schutz bietet eine galvanische Trennung von USB zwischen PC und Arduino mit extra Hardware. Ein einfacher USB-Hub kann ggf. auch schon für Schutz sorgen.

2.4 DC Eingang Schraubklemme

Schraubklemme zum alternativen Anschluß der Versorgungsspannung zum Beispiel aus einer Batterie. Es gelten die Angaben aus dem vorherigen Abschnitt (Hohlbuchse).



2.5 Schraubklemme Spannungsausgang

Ausgang für die Spannungen, die auf dem Arduino erzeugt werden. Die maximale Belastbarkeit des Arduino Nano ist zu beachten.

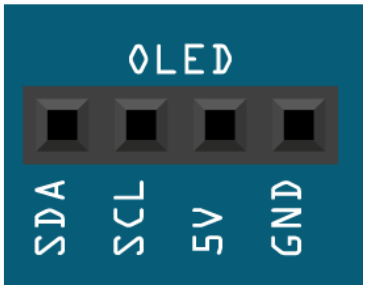
2.6 Buchsenleiste für OLED-Display

Buchsenleiste an dem die I²C-Signale (I2C, IIC) SDA (Arduino Signal A4), SCL (Arduino Signal A5), 5 V und GND anliegen. Hier kann bspw. ein OLED-Display der Größe 0.91 Inch, 128 x 32 Pixels aufgesteckt werden. Je nach Display-Shield ist es möglich, das Shield so aufzustecken, daß Display quer zur Längsseite des ftNano und oberhalb der Hohlbuchse liegt.

Die Anordnung der Pins kann je nach OLED-Shield unterschiedlich sein. Ggf. sind Jumper-Kabel zu verwenden.

Bei Verwendung des Buchsenleiste ist zu beachten, daß die angegebenen I/O-Pins belegt werden und nicht für andere Zwecke genutzt werden: A4 und A5

Pinbelegung:



Beachten Sie die Hinweise zu Pull-Up-Widerständen in 4.3

2.7 Buchsenleiste für Bluetooth-Modul

Auf die Buchsenleiste kann ein Bluetooth-Modul vom Typ HC-05, HC-06 oder ähnlich gesteckt werden, um Daten drahtlos zu übertragen. Mit einer 90° gewinkelter Pinleiste kann das Modul ggf. senkrecht aufgesteckt werden, so daß das Modul nicht übersteht und die Antenne für eine optimale Leistung nicht durch andere Baugruppen verdeckt wird.

Die Verbindung RxD vom Arduino muß an TxD am Bluetooth-Modul und TxD vom Arduino an RxD vom Modul angeschlossen werden. Der TxD-Pin der Buchse ist bereits mit einem Spannungsteiler auf den Signalpegel 3,3 V für den Eingang (RxD) vom Bluetooth-Modul versehen, so daß die Arduino-Signalspannung von 5 V reduziert wird. Der Dateneingang (RxD) vom Arduino ist kompatibel zu 3,3 D:\work\ftnano\test, so daß hier keine Anpassung des Spannungspegels erforderlich ist.

Die serielle hardwaremäßige Schnittstelle des Arduino ist mit dem USB-Wandler auf dem Arduino verbunden und wird zur Übertragung von Daten (serieller Monitor) und des Programmcodes benutzt. Bei der Programmübertragung von der Arduino IDE zum Arduino stört ein angeschlossenes Bluetooth-Modul die Kommunikation.

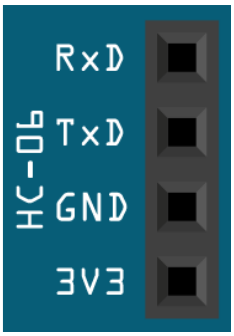
Während der Übertragung muß mindestens die Versorgungsspannung VCC zum Bluetooth-Modul getrennt werden.

Die Bluetooth-Module benötigen eine Betriebsspannung (VCC/VDD) von 3,3 V bis 5 V. Wenn die 3,3 V an der Buchsenleiste nicht ausreichen, können 5 V vom Arduino genutzt werden.

Die Anordnung der Pins kann je nach Bluetooth-Shield unterschiedlich sein. Ggf. sind Jumper-Kabel zu verwenden.

Bei Verwendung des Buchsenleiste ist zu beachten, daß die angegebenen I/O-Pins TxD (Daten-Ausgang vom Arduino Pin D1) und RxD (Daten-Eingang zum Arduino Pin D0) belegt werden und nicht für andere Zwecke genutzt werden.

Pinbelegung:



Die Signale an der Buchsenleiste können auch für andere Hardware benutzt werden.

2.8 Fischertechnik-Kontakt GND/Masse

Kontakt-Bohrungen, um Fischertechnik-Flachstecker aufzunehmen.

Verbindung mit GND.

Die Löcher für die Fischertechnik-Kontakte sind so dimensioniert, daß ein sicherer Kontakt und Halt für die Flachstecker von Fischertechnik entsteht. Sitzt der Stecker zu locker, können die Kontaktelemente am Flachstecker mit einem kleinen Schraubenzieher minimal gespreizt werden.

2.9 Fischertechnik-Kontakt 9 V (VCC)

Kontakt-Bohrungen, um Fischertechnik-Flachstecker aufzunehmen.

Verbindung mit der Versorgungsspannung VCC zur Versorgung von Komponenten.

2.10 Fischertechnik-Kontakt digitale I/O

Kontakt-Bohrungen, um Fischertechnik-Flachstecker aufzunehmen.

Die Kontakte sind den entsprechenden I/O-Pins des Arduinos gemäß der Beschriftung zugeordnet. Analoge Arduino-Eingänge können immer wie digitale Ein-/Ausgänge benutzt werden. Hier werden die vier analog/digitalen I/Os des Arduinos als digitale I/Os benutzt.

Die digitalen Ein-/Ausgänge (I/O) verfügen über einen Treiber, um die Spannung des Arduino Nano (5 V) an Fischertechnik 9 V anzupassen. Die maximale Belastbarkeit eines Ausgangs liegt bei etwa

20 mA. Als Eingänge genutzt sind die die Pins über die Treiber kompatibel zu 9 V und verfügen bereits über einen Pull-Up-Widerstand, so daß ein unbeschalteter Eingang ein High-Signal liefert.

2.11 Fischertechnik-Kontakt analoge I/O

Kontakt-Bohrungen, um Fischertechnik-Flachstecker aufzunehmen.

Die Kontakte sind den entsprechenden I/O-Pins des Arduinos gemäß der Beschriftung zugeordnet. Hier handelt es sich um vier analoge Eingänge.

Die Eingänge sind über einen Spannungsteiler mit dem Arduino verbunden (2× 47 kΩ, siehe Schaltplan). Die maximale Eingangsspannung an den analogen Eingänge des Arduinos beträgt 5 V. Über den Spannungsteiler kann eine Spannung von maximal 10 V über die analogen Fischertechnik-Kontakte eingespeist werden.

2.12 Fischertechnik-Kontakt digitale I/O

Kontakt-Bohrungen, um Fischertechnik-Flachstecker aufzunehmen.

Die Kontakte sind den entsprechenden digitalen I/O-Pins des Arduinos gemäß der Beschriftung zugeordnet. Es gibt keinerlei Einschränkungen hinsichtlich der normalen Funktionalität. Die mit einer Tilde („~“) gekennzeichneten Anschlüsse bieten die Möglichkeit, ein PWM-Signal auszugeben (Standardfunktion des Arduino)

Die digitalen Ein-/Ausgänge (I/O) verfügen über einen Treiber, um die Spannung des Arduino Nano (5 V) an Fischertechnik 9 V anzupassen. Die maximale Belastbarkeit eines Ausgangs liegt bei etwa 20 mA. Als Eingänge genutzt sind die die Pins über die Treiber kompatibel zu 9 V und verfügen bereits über einen Pull-Up-Widerstand, so daß ein unbeschalteter Eingang ein High-Signal liefert.

2.13 Fischertechnik-Kontakt digitale Last-Ausgänge

Kontakt-Bohrungen, um Fischertechnik-Flachstecker aufzunehmen.

Diese Kontakte sind über einen (Motor-) Treiber mit dem Arduino verbunden und dienen als Ausgänge zum Anschluß von höheren Lasten wie Lampen oder Motoren. Jeder Ausgang kann ca. 200 mA liefern. Die Treiber-ICs verfügen über einen Überlastschutz. Wenn die Ausgänge genutzt werden, sind die jeweiligen digitalen Ausgänge nicht für andere Zwecke nutzbar (siehe 4.10)

2.14 Fischertechnik-Kontakt GND/Masse

Kontakt-Bohrungen, um Fischertechnik-Flachstecker aufzunehmen.

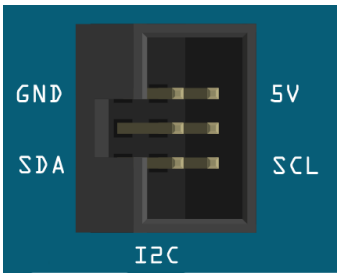
Verbindung mit GND.

2.15 Wannenstecker für I²C

Wannenstecker an dem die I²C-Signale (I2C, IIC) SDA (Arduino Signal A4), SCL (Arduino Signal A5), 5 V und GND anliegen. In den Wannenstecker passen Stecker für Flachbandkabel. Dies wird vom TXT-Controller von Fischertechnik zum Anschluß von Sensoren etc. genutzt.

Bei Verwendung des Buchsenleiste ist zu beachten, daß die angegebenen I/O-Pins belegt werden und nicht für andere Zwecke genutzt werden: A4 und A5

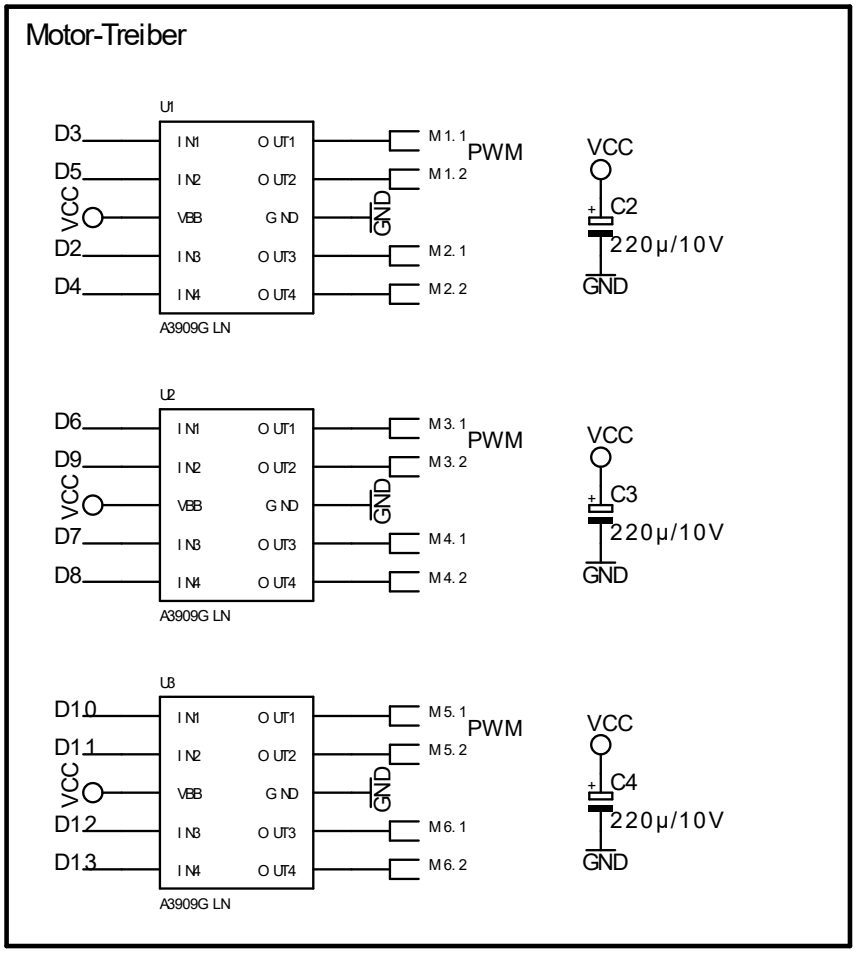
Pinbelegung:



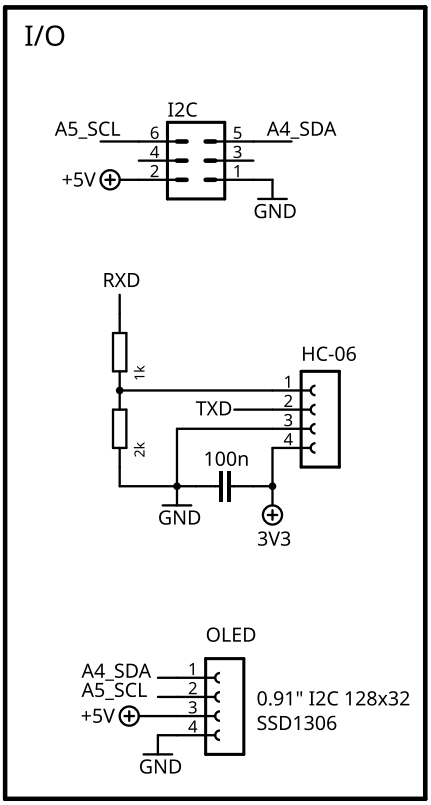
Beachten Sie die Hinweise zu Pull-Up-Widerständen in 4.3

3 Schaltplan

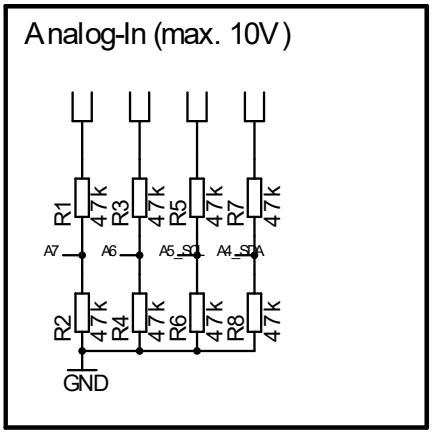
3.1 (Motor-) Treiber D2–D13



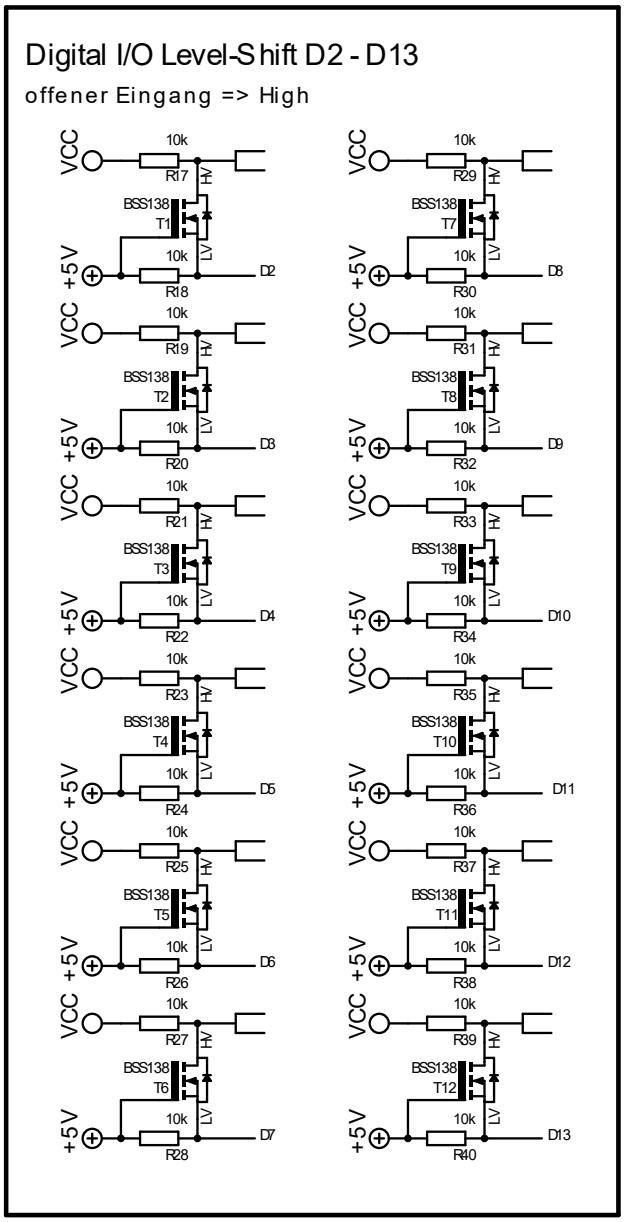
3.2 Buchsenleiste I²C, HC-06 und OLED



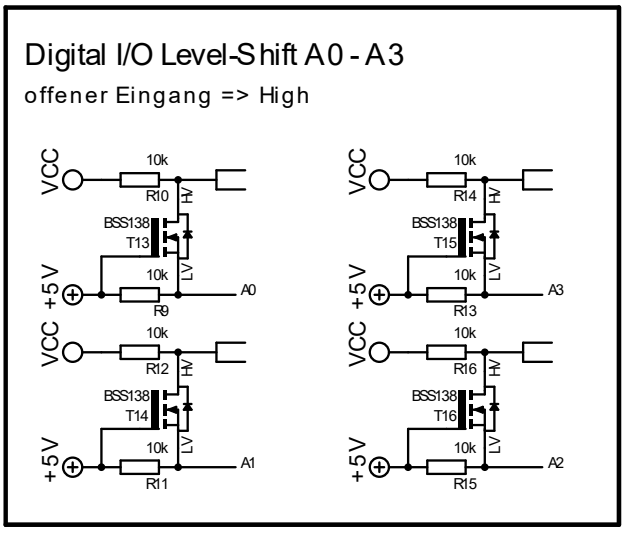
3.3 Analoge Eingänge A4–A7



3.4 Digitale Ein-/Ausgänge D2–D13



3.5 Digitale Ein-/Ausgänge A0–A3



4 Inbetriebnahme

4.1 Vorbereitung

Nachdem auf die Buchsenleiste für den Arduino Nano (s. Kapitel 2.1) ein solcher gesteckt wurde, ist der ftNano betriebsbereit und kann über einen der DC-Eingänge mit Gleichspannung versorgt werden (s. Kapitel 2.3 und 2.4). Die Power-LED auf dem Arduino Nano leuchtet daraufhin.

Die hier verwendeten Bauteile für das Programm Fritzing (https://fritzing.org/) können auf der Seite https://github.com/coderfls/Fritzing-Parts heruntergeladen werden.

4.2 Arduino programmieren

Der Arduino wird wie üblich in C/C++ programmiert. Hierfür wird in dieser Dokumentation die Arduino-IDE (https://www.arduino.cc/en/software, https://docs.arduino.cc/software/ide-v2/tutorials/getting-started-ide-v2) unter Windows benutzt, es können aber auch andere Entwicklungsumgebungen eingesetzt werden.

Zur Programmübertragung vom PC auf den Arduino muß der Arduino Nano mit einem USB-Kabel am PC angeschlossen sein. Siehe hierzu auch Abschnitt 2.3.

In der IDE muß wie üblich der passende COM-Port gewählt werden (*Werkzeuge*|*Port*), an dem der Arduino Nano zu finden ist. Außerdem muß bei *Werkzeuge*|*Board* der Eintrag *Arduino Nano* stehen bzw. über das Untermenü *Arduino AVR Boards* gewählt werden. Je nach Modell des Arduino muß bei *Werkzeuge*|*Prozessor* gewählt werden, welcher ATmega (ATmega128 oder ATmega328) auf dem Nano verbaut ist und ob ggf. der alte Bootloader (*old*) installiert ist.

4.3 I²C Pull-Up-Widerstände

Der I²C-Bus erfordert zwei Pull-Up-Widerstände von ca. 10 kΩ bis 4,7 kΩ bei 5 V an den Signalen SDA und SCL. Je nach benutzter Hardware ist zu berücksichtigen, ob diese schon verbaut sind, extern ergänzt werden müssen oder die im ATmega328 integrierten aktiviert werden. Auf vielen Shields sind die Widerstände bereits verbaut. C/C++-Libraries unter Arduino aktivieren meistens die internen Pull-Ups der I/O-Pins.

Es ist dabei zu beachten, daß nicht mehrere Pull-Ups in der Schaltung eingesetzt werden, da sich dann durch die Parallelschaltung der Gesamtwiderstand zu stark verringern kann und die Hardware nicht mehr korrekt funktioniert.

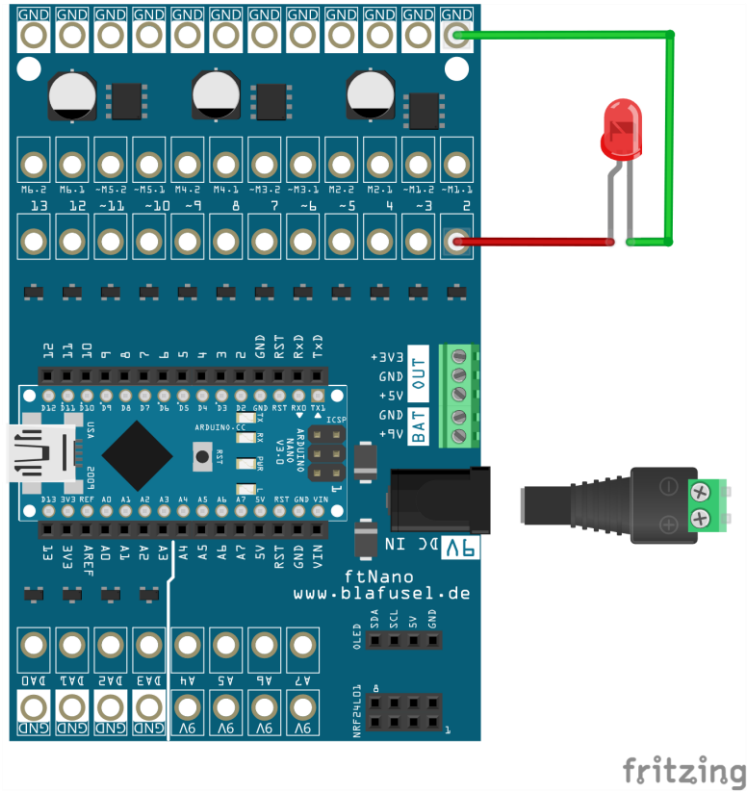
4.4 Digitale Ausgänge D2–D13

digital_out

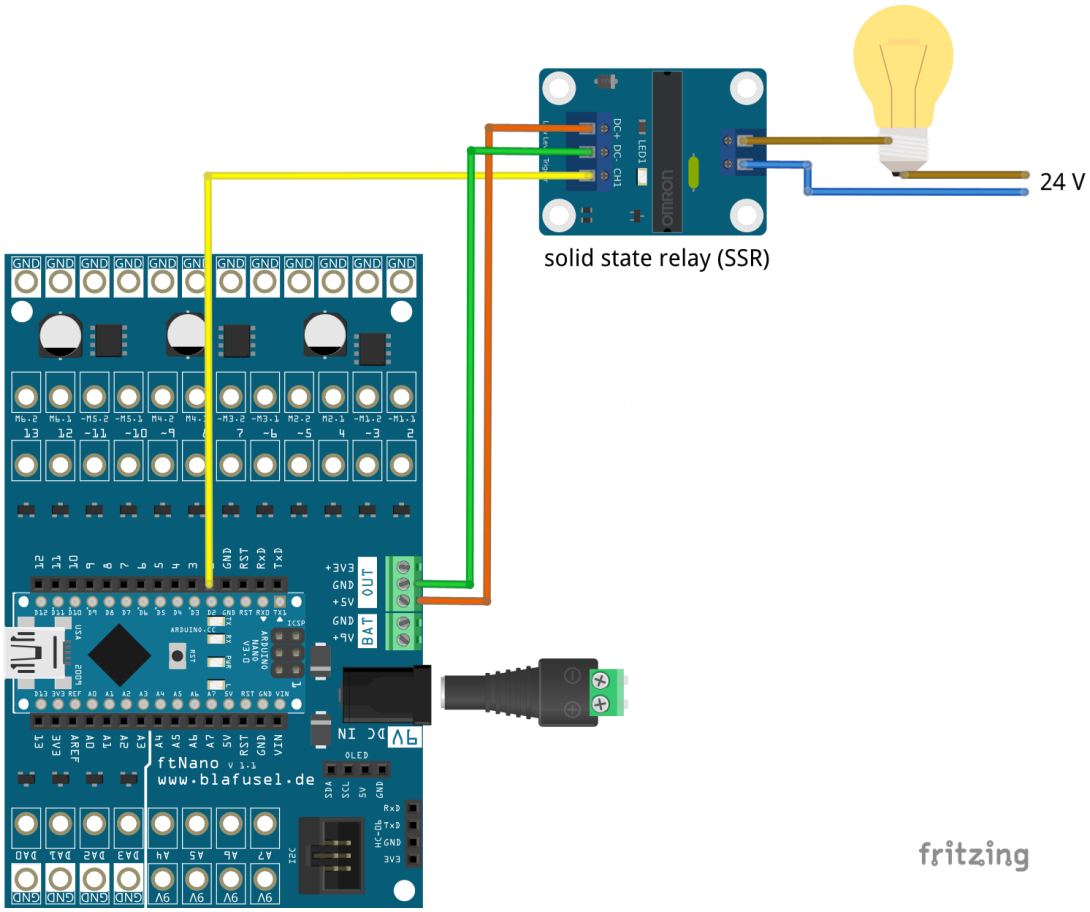
Die digitalen I/O-Anschlüsse können wie üblich einzeln als Ausgang genutzt werden.

Weil der Pegelwandler (Level Shifter) die Versorgungsspannung (9 V) über einen Widerstand mit 10 kΩ bereitstellt, kann eine LED i. d. R. direkt und ohne Vorwiderstand angeschlossen werden.

> Beachten Sie die Ausführungen im Abschnitt 4.10 zur doppelten Belegung.



Größere Lasten oder abweichende Spannungen zu 9 V können vom Arduino über ein Relais oder Solid-State-Relais angesteuert werden.



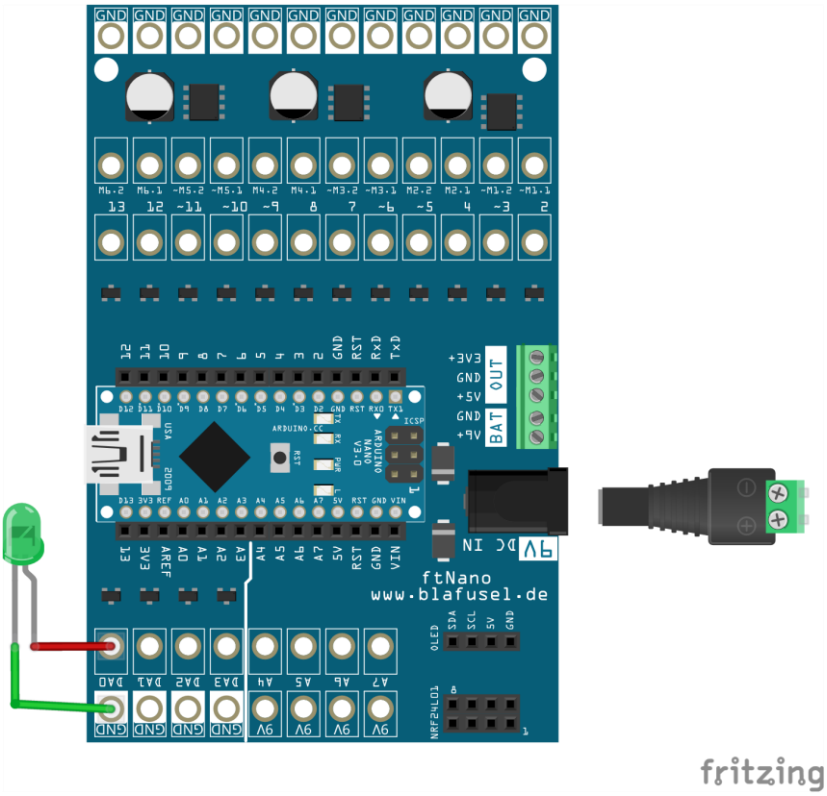

```
void setup() {
  pinMode(2, OUTPUT);
}

void loop() {
  digitalWrite(2, HIGH);
  delay(2000);
  digitalWrite(2, LOW);
  delay(200);
}
```

4.5 Digitale Ausgänge DA0–DA3

da_out

Die I/O-Pins DA0–DA4 können wie üblich einzeln als digitale Ausgang genutzt werden.

Weil der Pegelwandler die Versorgungsspannung (9 V) über einen Widerstand mit 10 kΩ bereitstellt, kann eine LED i. d. R. direkt und ohne Vorwiderstand angeschlossen werden.

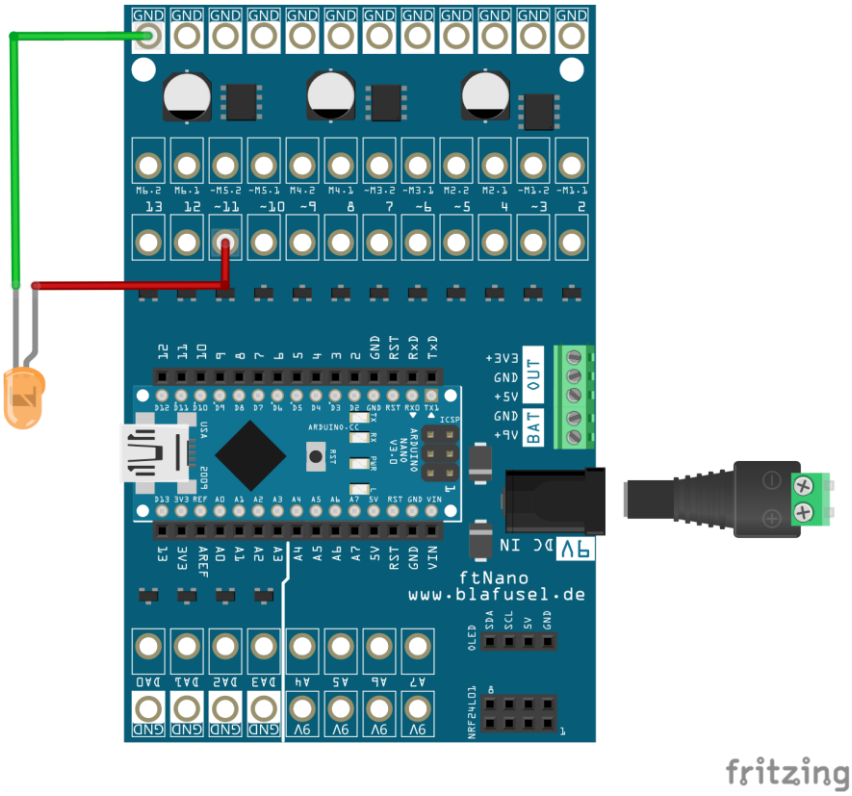



```
void setup() {
  pinMode(A0, OUTPUT);
}

void loop() {
  digitalWrite(A0, HIGH);
  delay(2000);
  digitalWrite(A0, LOW);
  delay(200);
}
```

4.6 PWM an digitalen Ausgängen

pwm

Der Arduino bietet einige Ausgänge, die ein Pulsweitenmoduliertes Signal (PWM) ausgeben können, ohne daß hierfür Ressourcen im Programm notwendig sind. Diese sind mit einer Tilde („~“) gekennzeichnet. Seitens Arduino wird dies als analoges Ausgangssignal bezeichnet, was faktisch falsch ist https://docs.arduino.cc/learn/microcontrollers/analog-output. Die I/O-Pins können über die Pegelwandler (Level Shifter) als Ausgang für 0 V bzw. 9 V Signalpegel genutzt werden.



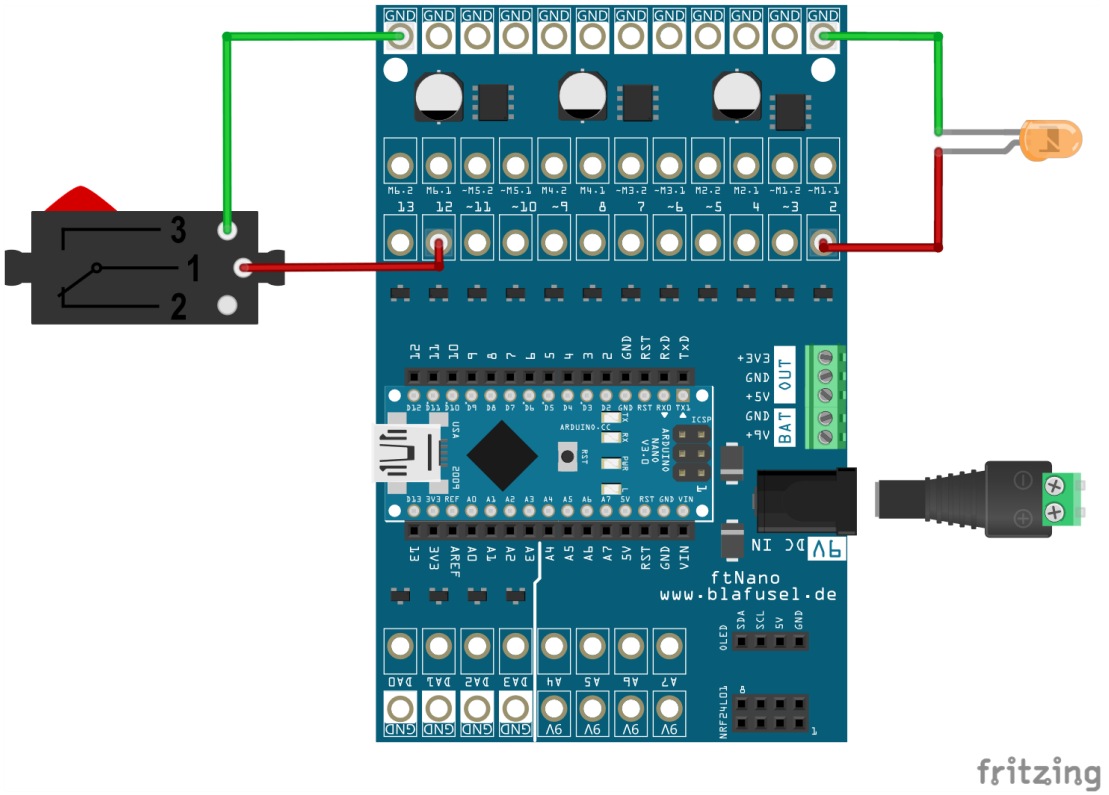

```
uint8_t i;

void setup() {}   // useless stupid stuff

void loop() {
  for (i = 0; i <= 255; i++)  {
    analogWrite(11, i);   // PWM
    delay(20);
  }
}
```

4.7 Digitale Eingänge D2–D13

digital_in

Die I/O-Pins können über die Pegelwandler (Level Shifter) als Eingang für 0 V bzw. 9 V Signalpegel genutzt werden.

Die Eingänge sind Low-Aktiv, weil der Pegelwandler einen Pull-Up-Widerstand besitzt. Das bedeutet, daß das Eingangssignal im unbeschalteten Zustand High (1) führt und gegen GND (Low/0) geschaltet werden muß.




```
void setup() {
  pinMode (2, OUTPUT);
  pinMode (12, INPUT);
}

void loop() {
  if (!digitalRead(12))              // Input is low active
    digitalWrite (2, HIGH);          // key pressed (closed)
  else
    digitalWrite (2, LOW);
}
```

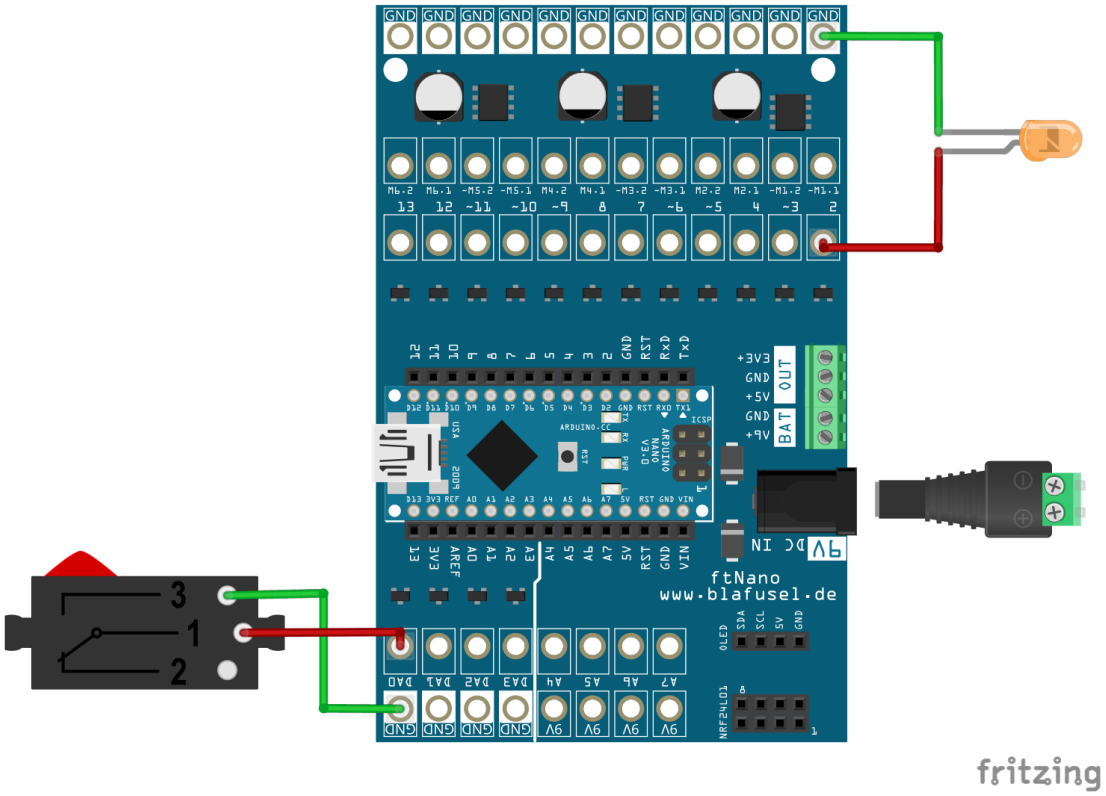
> Auf dem Arduino ist die interne LED „L“ an Pin 13 angeschlossen. Je nach Board kann es sein, daß dies die Nutzung des Pins als Eingang behindert, weil intern immer Low-Pegel am Eingang erkannt wird. Es ist empfehlenswert, Pin 13 nur als Ausgang zu nutzen.

4.8 Digitale Eingänge DA0–DA3

da_in

Die I/O-Pins können über die Pegelwandler (Level Shifter) als Eingang für 0 V bzw. 9 V Signalpegel genutzt werden.

Die Eingänge sind Low-Aktiv, weil der Pegelwandler einen Pull-Up-Widerstand besitzt. Das bedeutet, daß das Eingangssignal im unbeschalteten Zustand High (1) führt und gegen GND (Low/0) geschaltet werden muß.

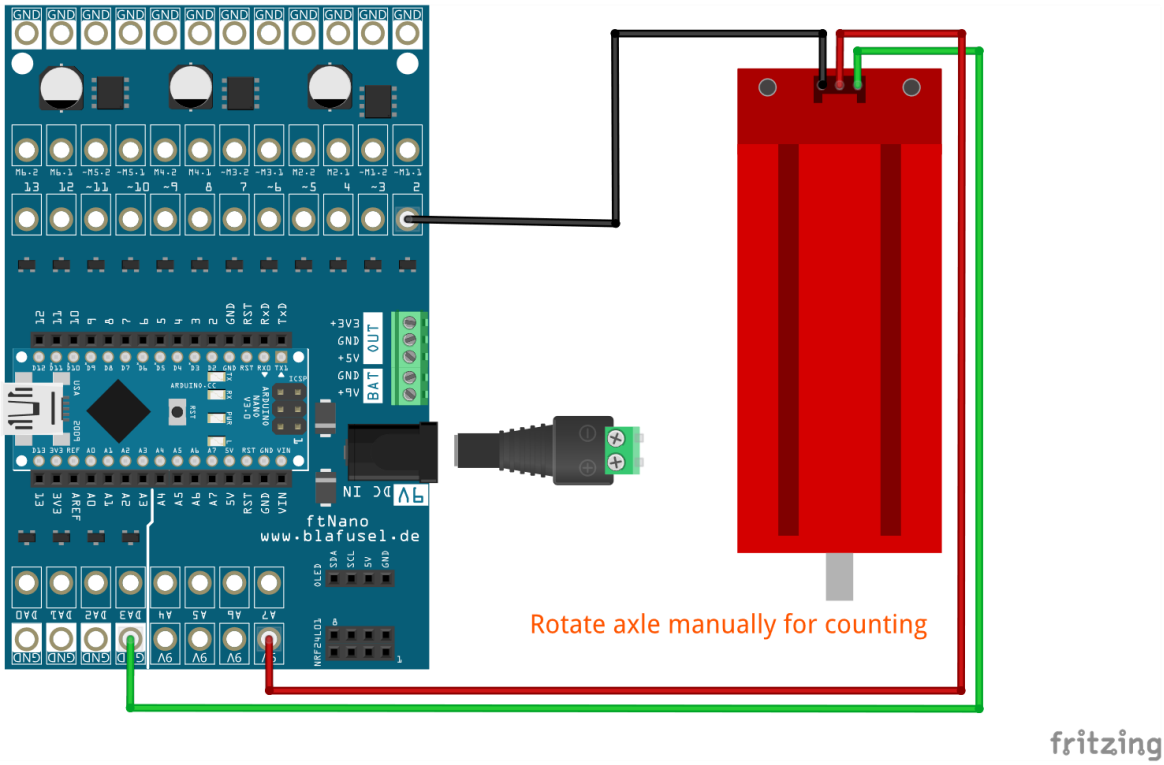



```
void setup() {
  pinMode (2, OUTPUT);
  pinMode (A0, INPUT);
}

void loop() {
  if (digitalRead(A0))          // Input is low active
    digitalWrite (2, HIGH);     // key pressed (opened in this example!)
  else
    digitalWrite (2, LOW);
}
```

4.9 Interrupt-Eingänge (Zähler) D2 und D3

irq

Der Arduino Nano kann an den Eingängen 2 und 3 Interrupts erkennen, die die normale Programmausführung unterbrechen (https://www.arduino.cc/reference/en/language/functions/external-interrupts/attachinterrupt/). Diese Eingänge werden genutzt, um jederzeit auf Ereignisse zu reagieren oder unabhängig vom eigentlichen Programm Impulse zu zählen (bspw. von Tastern, Drehencodern oder Motoren)




```
volatile uint16_t cnt;    // must be volatile

void setup() {
  pinMode (2, INPUT);
  attachInterrupt(digitalPinToInterrupt(2), count, CHANGE);
  Serial.begin(9600);     // for serial output open "serial monitor"
}

void loop() {
  delay (1000);
  Serial.println(cnt);    // output actual counter value
  cnt = 0;
}

// count interrupts
void count() {
  cnt++;    // increment
}
```

4.10 Digitale Last-Ausgänge M1.1–M6.2

motor

Über (Motor-) Treiber stehen leistungsfähige Ausgänge als Alternative zu den digitalen Ausgängen D2–D13 bereit. Je zwei digitale Ausgänge D2–D13 bilden eine Gruppe von zwei Lastausgängen und können entweder über die Level Shifter für 9 V benutzt werden (siehe 4.3) oder werden für den Motortreiber belegt.

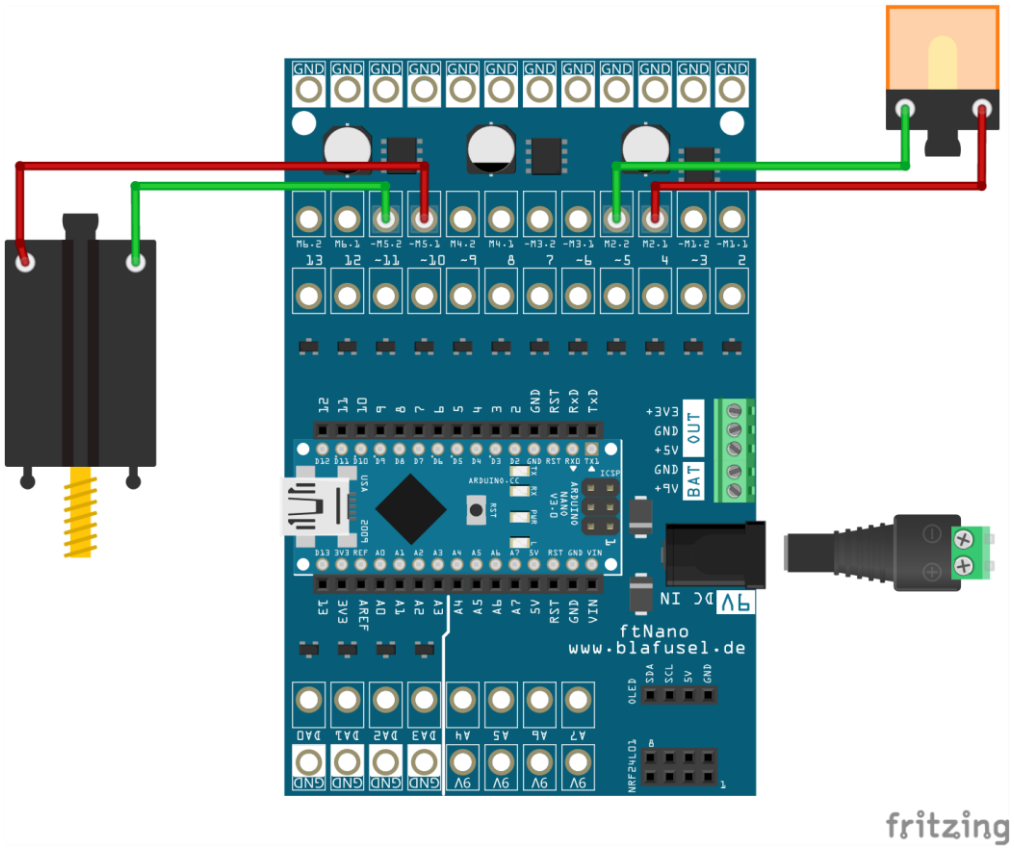
Zuordnung der Ausgänge (vgl. mit Schaltplan):

Digital D…	Last M…	PWM?
3	1.1	✓
5	1.2	✓
2	2.1	—
4	2.2	—
6	3.1	✓
9	3.2	✓
7	4.1	—
8	4.2	—
10	5.1	✓
11	5.2	✓
12	6.1	—
13	6.2	—

Die beiden Ausgänge müssen unterschiedliche Signalpegel führen, um die Last anzusteuern. Damit kann dann auch die Drehrichtung eines Elektromotors beeinflußt werden. Sind beide Ausgänge Low, befindet sich der Treiberbaustein (Dual Full Bridge Motor Driver A3909) im Sleep-Modus (Ausgänge High Impedance). Sind beide Steuersignalpegel High, wird die Last nicht angesteuert (Low, Halt). Einzelne Ausgänge können mit einem vom Arduino automatisch erzeugten PWM-Signal gesteuert werden (mit Tilde gekennzeichnet).

Es vereinfacht die Nutzung, wenn für die Lastausgänge im Programm mittels #define ein Makro erstellt wird, bei dem sinnvoll benannten Bezeichnern die digitalen Ports zugeordnet werden. Bspw.:

```
#define M1_1      3
#define M1_2      5
```



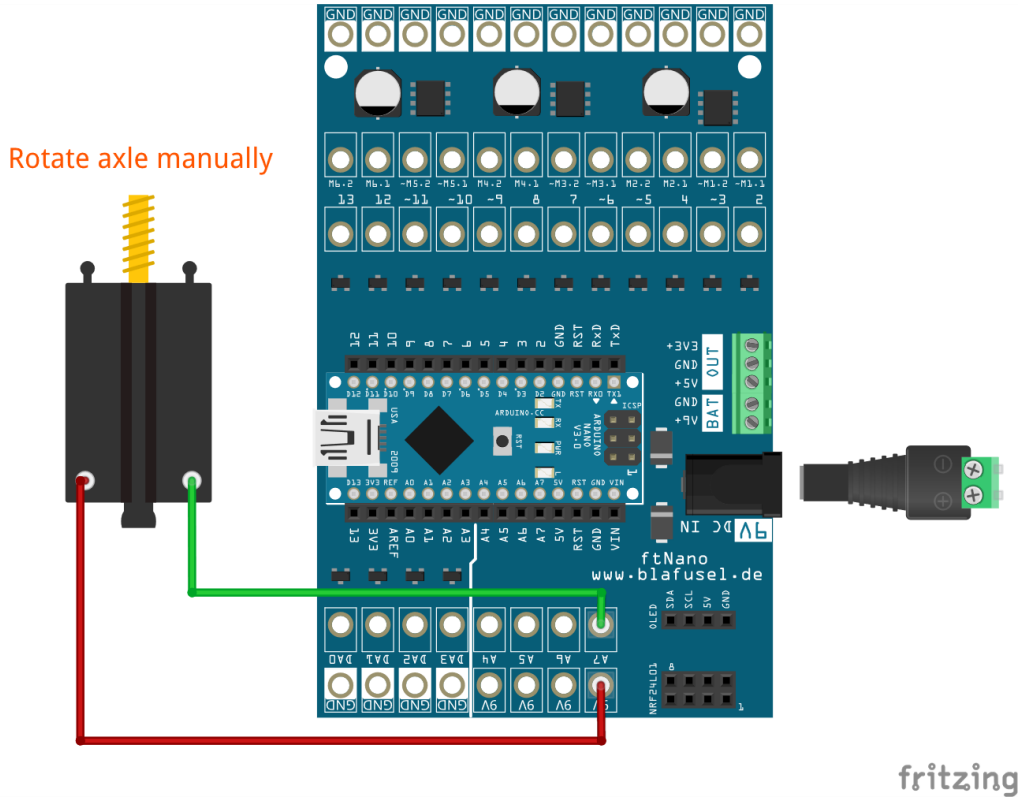
```
#define M2_1  2
#define M2_2  4
#define M5_1  10
#define M5_2  11

void setup() {
  pinMode (M2_1, OUTPUT);
  pinMode (M2_2, OUTPUT);
  pinMode (M5_1, OUTPUT);
  pinMode (M5_2, OUTPUT);
}

void loop() {
  // rotate one direction
  digitalWrite (M5_1, LOW);
  analogWrite (M5_2, 40);   // PWM - must maybe higher
  delay(2000);

  // stop
  digitalWrite (M2_1, LOW);   // light bulb
  digitalWrite (M2_2, HIGH);  // light bulb ON
  digitalWrite (M5_1, LOW);
  digitalWrite (M5_2, LOW);
  delay(500);
  digitalWrite (M2_1, LOW);   // light bulb
  digitalWrite (M2_2, LOW);  // light bulb OFF

 // rotate other direction
  digitalWrite (M5_2, LOW);
  analogWrite (M5_1, 255);   // PWM - must maybe higher
  delay(2000);

  // stop
  digitalWrite (M2_1, LOW);   // light bulb
  digitalWrite (M2_2, HIGH);  // light bulb ON
  digitalWrite (M5_1, LOW);
  digitalWrite (M5_2, LOW);
  delay(500);
  digitalWrite (M2_1, LOW);   // light bulb
  digitalWrite (M2_2, LOW);  // light bulb OFF
}
```

4.11 Analoge Eingänge A4–A7

analog_in

An vier analogen Eingängen befindet sich ein Spannungsteiler (s. Schaltplan). Über diesen können analoge Spannungen im Bereich 0 V – 10 V gemessen werden.

> Diese Ausgänge können nicht als digitale Ein-/Ausgänge genutzt werden, da kein Treiber vorhanden ist. Die Pins A4 und A5 werden auch für I²C benötigt.




```
void setup() {
  Serial.begin(9600);
}

void loop() {
  // rotate motor axle to work as an generator
  Serial.println(analogRead(A7));  //read analog value
  delay (50);
}
```

4.12 Taster an 5 V Signaleingang

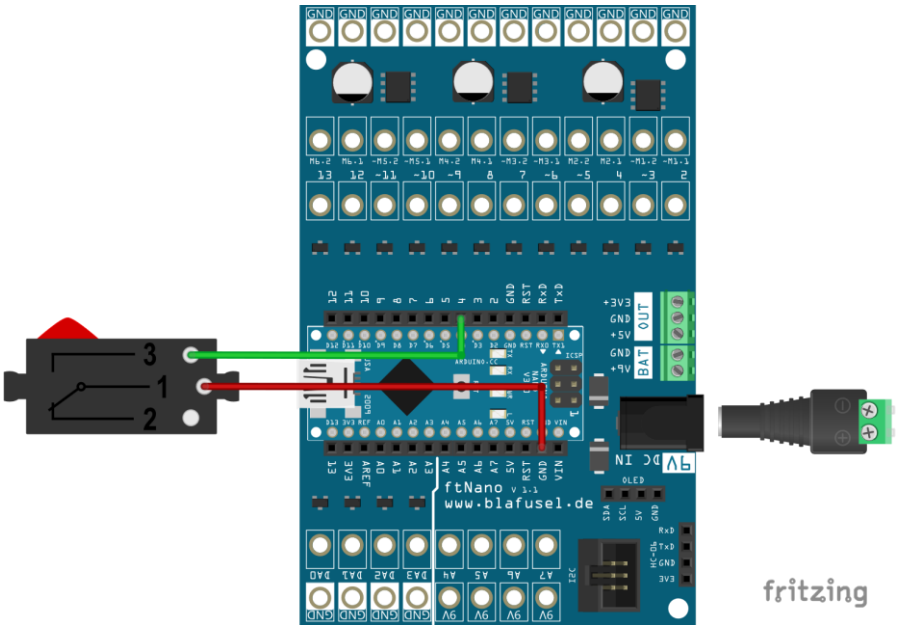
Beliebige Taster und Schalter können wie üblich direkt an den Arduino angeschlossen werden, wenn die Taster mit Signalpegeln von 5 V betrieben werden.

> Um definierte Signalpegel LOW bzw. HIGH zu erhalten, muß der Anschluß des Schalters, der mit dem I/O-Pin des ATmega verbunden ist, mit einem Pull-Up bzw, Pull-Down-Widerstand verbunden sein.

button

Das Beispiel fragt den Zustand eines Tasters (Umschalters) an Pin D4 des Arduino ab. Es werden nur die Anschlüsse 1 und 3 des Umschalters benutzt. Im Ruhezustand ist der Taster (Pin 3) offen und wird bei Betätigung mit Pin 1 verbunden, so daß der I/O-Pin D4 auf Masse gezogen wird. Der Taster ist damit Low-Aktiv (sendet ein Low-Signal bei Betätigung).

Damit die Position im Ruhezustand ebenfalls einen Signalpegel liefert und D4 nicht nur *in der Luft hängt*, ist ein Pull-Up-Widerstand erforderlich. Hierfür wird der interne Widerstand des I/O-Pins per Befehl aktiviert, so daß kein externes Bauteil benötigt wird.

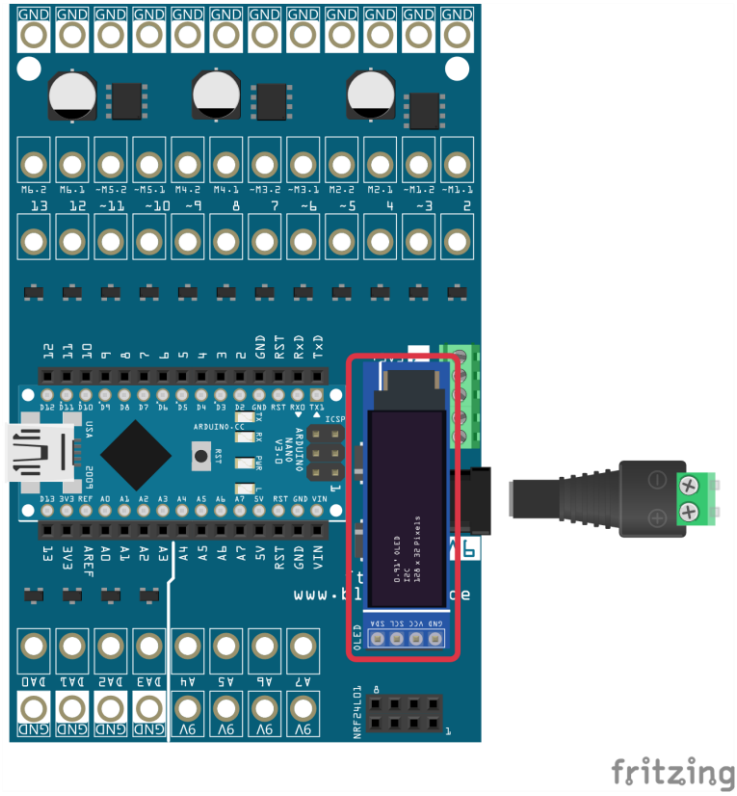



```
#define tasterIO 4
uint8_t lastStatus = 0;
void setup() {
  Serial.begin(9600);
  pinMode(tasterIO, INPUT_PULLUP);
  lastStatus = digitalRead(tasterIO);
  Serial.write ("\nUse the button\n\n");
}
void loop() {
  uint8_t keyStatus = digitalRead(tasterIO);
  if (keyStatus != lastStatus) {
    if (digitalRead(tasterIO) == HIGH)
      Serial.write ("NOT pressed\n");
    else
      Serial.write ("pressed\n");
    lastStatus = keyStatus;
  }
  delay (100);
}
```

5 OLED-Display

oled

Auf die Buchsenleiste OLED kann ein Display mit I²C-Bus aufgesetzt oder angeschlossen werden. Die Signale an der Buchsenleiste können auch für andere Hardware benutzt werden. Beachten Sie die Hinweise in Kapitel 2.6.



Zur Ansteuerung von Grafik-Displays gibt es eine Vielzahl an Bibliotheken. Diese müssen ggf. über die Arduino-IDE und *Werkzeuge|Bibliotheken verwalten* nachträglich installiert werden. Die Bibliotheken liefern meistens auch Beispielprogramme mit, die über *Datei|Beispiele|…* geöffnet werden können. Im Beispielordner befindet sich ein Programm, welches Text und eine Zahl auf dem Display ausgibt.

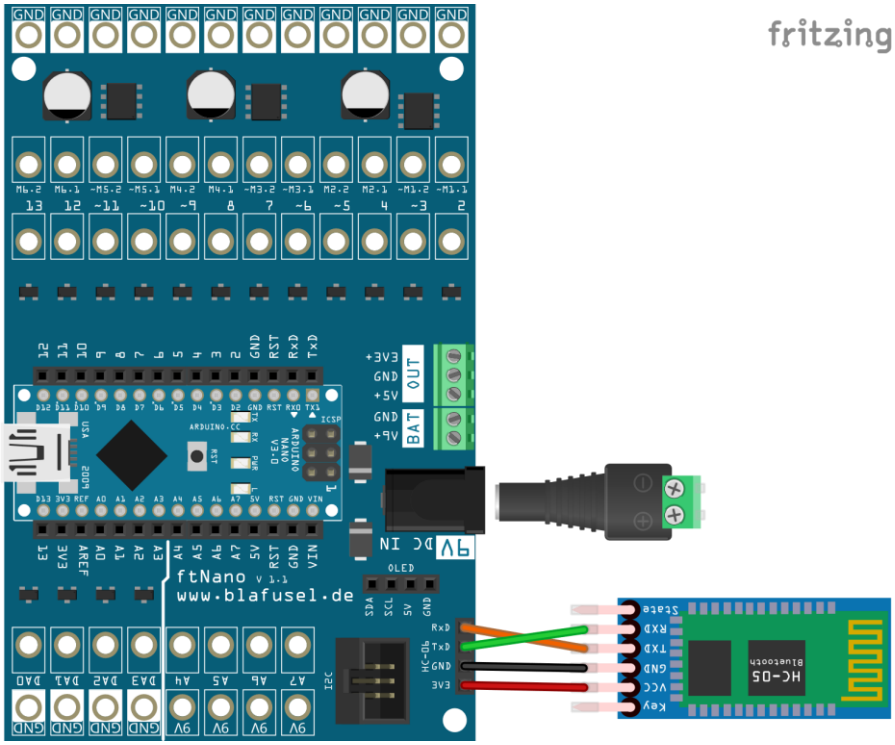
Beachten Sie die Hinweise zu Pull-Up-Widerständen in 4.3

---

6 Bluetooth-Modul

bluetooth

Mit einem Bluetooth-Modul können Daten zwischen Arduino und Smartphone oder PC ausgetauscht werden. Die Buchsenleiste stellt die notwendigen Pins bereit und ist über die hardwareseitige serielle Schnittstelle des ATmega programmierbar.



Im Beispielordner befindet sich ein einfaches Programm zum Senden von Daten an ein Smartphone.

Unter Android müssen Sie das Bluetooth-Modul koppeln und eine Terminal-App nutzen:

- Installieren Sie das Programm auf dem Arduino. Beachten Sie, daß Sie dazu die Verbindung zum BT-Modul trennen müssen (s. Kapitel 2.7).
- Nachdem das BT-Modul mit Spannung versorgt ist (und alle vier Verbindungen hergestellt sind), öffnen Sie in Android Bluetooth-Einstellungen.

- Aktivieren Sie Bluetooth. Es werden Geräte in der Nähe gesucht. Nach ein paar Sekunden wird Ihnen das BT-Modul als verfügbar angezeigt. Tippen Sie auf den Eintrag, um das Pairing mit dem Gerät zu starten.

- Geben Sie den Pairing-Code ein. Dieser lautet i. d. R „1234“ oder „0000“ und tippen Sie dann auf *OK*. Das neue BT-Gerät wird aufgelistet.

- Starten Sie die Terminal-App (bspw. *Serial Bluetooth Terminal* aus dem Google Play Store: https://t1p.de/v6x73).
- Tippen Sie auf das Menü-Symbol oben links und öffnen Sie den Bereich *Devices*. Tippen Sie auf das von Ihnen benutzte BT-Modul. Es wird versucht eine Verbindung zum Modul aufzubauen. Sollte dies nicht automatisch funktionieren, gehen Sie zur Terminalansicht

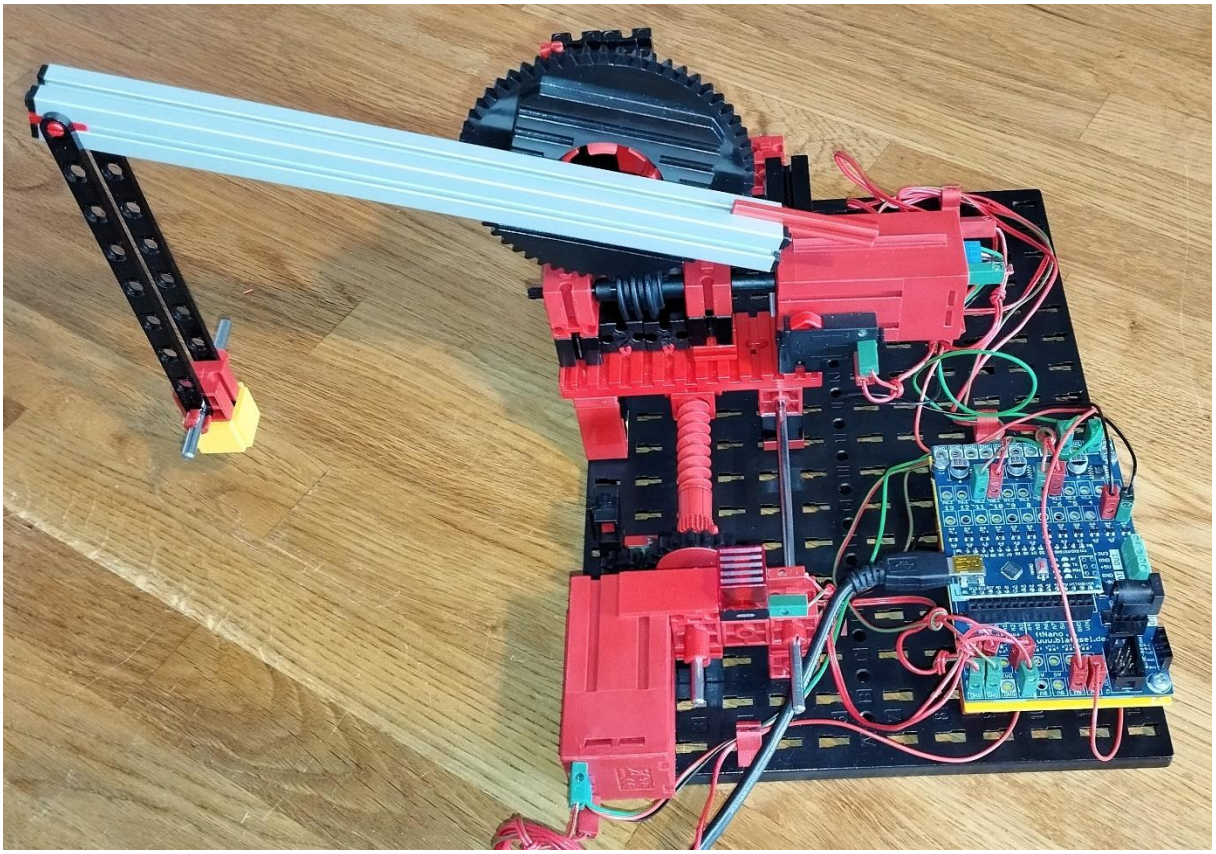
zurück und tippen auf den Button für *Connect/Disconnect*. Es werden die eingehenden Daten angezeigt.

- Sollten Sie Sonderzeichen am Zeilenende sehen, öffnen Sie das Menü, wählen *Settings*, wechseln auf den Tab *Receive* und stellen *Newline* auf *CR+LF*.
- Wenn Sie in der Arduino-IDE den seriellen Monitor öffnen (Baudrate 9600 einstellen), sehen Sie hier die Daten, die vom Arduino an das BT-Modul geschickt werden.

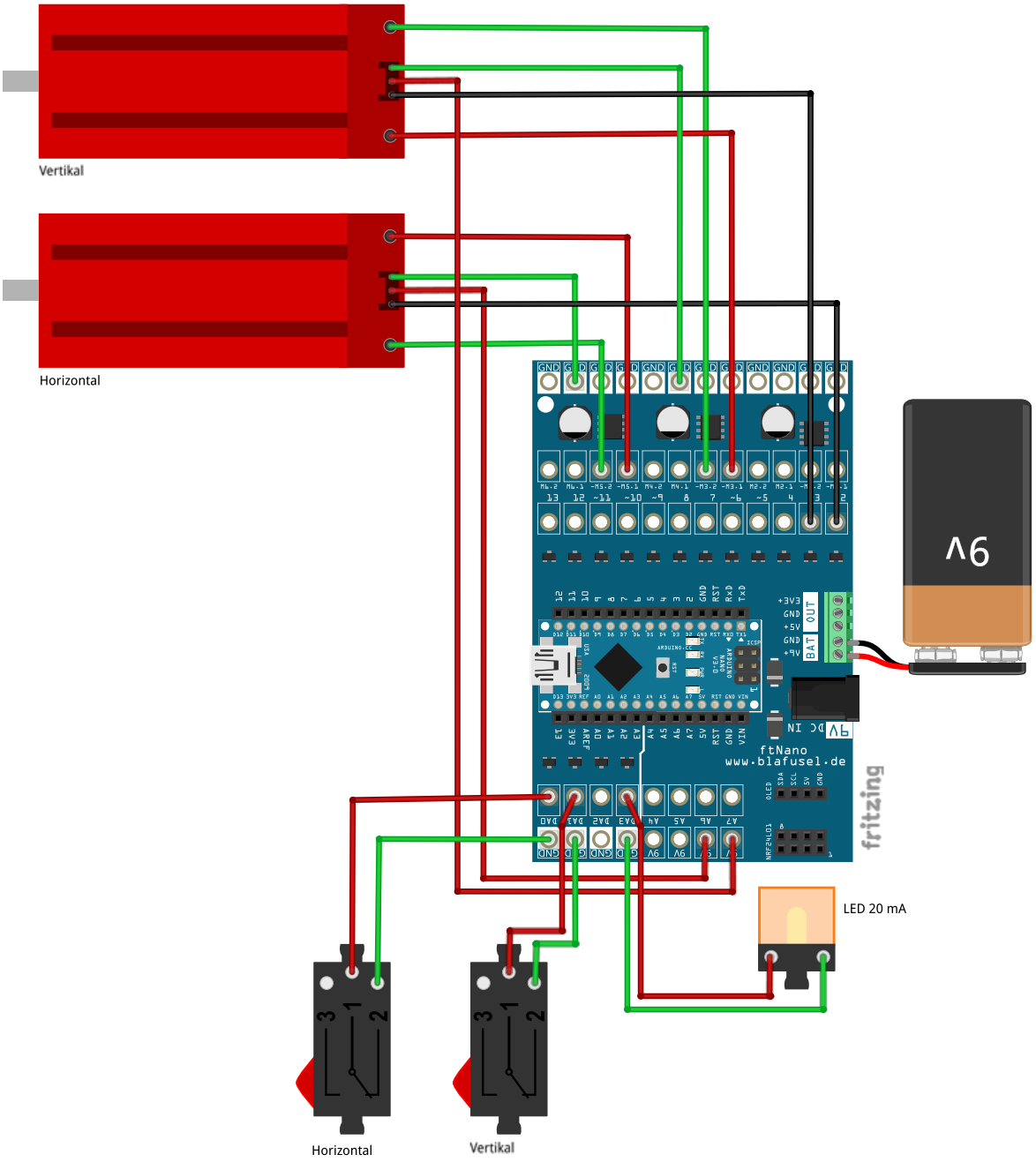
7 Modellbeispiel

kran

Abgebildet ist ein einfacher Kran mit zwei Achsen (horizontale Bewegung und vertikale Bewegung des Auslegers). Ein Schalter befindet sich in der Home-Position einer jeden Achse. Nachdem der Kran und der Ausleger soweit verfahren wurden, daß der Taster berührt wird, können die weiteren Position durch Zählen der Schritte der Encoder-Motoren bestimmt werden. Da die digitalen I/Os über einen Pull-Up-Widerstand verfügen, ist keine weitere Beschaltung oder die Benutzung der internen Widerstände des ATmega328 notwendig. Während der Motorbewegungen blinkt eine LED (bspw. Art.-Nr. 162134, keine Glühbirne benutzen).



Das Beispielprogramm bewegt den Kran ein wenig hin und her und beginnt dann von vorne. Im Ordner befinden sich hochauflösende Fotos.


Vertikal
Horizontal
9V
LED 20 mA
ftNano
www.blafusel.de
fritzing
Horizontal
Vertikal