Skriptum Mikrocontroller

Teil 1 GPIOs Hardware

DI Wolfgang Zukrigl, HTL Litec
DI(FH) Andreas Pötscher, HTL Litec

1.9.2025

Inhaltsverzeichnis

Datum	Beschreibung der durchgeführten Änderung	Autor
08.09.2018	V2.0 . . . vollständige Neu-Erstellung des Skriptums, Verwenden der Bibliothek LitecAVRTools	Wolfgang Zukrigl, HTL Linz–Paul-Hahn-Straße (LiTec)
16.02.2019	V2.1 . . . direktes Programmieren der Special Function Register ausführlicher erklärt	Wolfgang Zukrigl, HTL Linz–Paul-Hahn-Straße (LiTec)
16.02.2019	V2.2 . . . Aufteilung des Gesamtskriptums in drei Teile	Wolfgang Zukrigl, HTL Linz–Paul-Hahn-Straße (LiTec)

Datum	Beschreibung der durchgeführten Änderung	Autor
26.08.2024	V2.3 . . . übernahme im Markdown und allgemeine Überarbeitung	Andreas Pötscher, HTL Linz–Paul-Hahn-Straße (LiTec)

Allgemeines über GPIOs

Pins und Ports

Ein Pin ist Anschluss eines ICs, der mit der Platine verlötet wird.

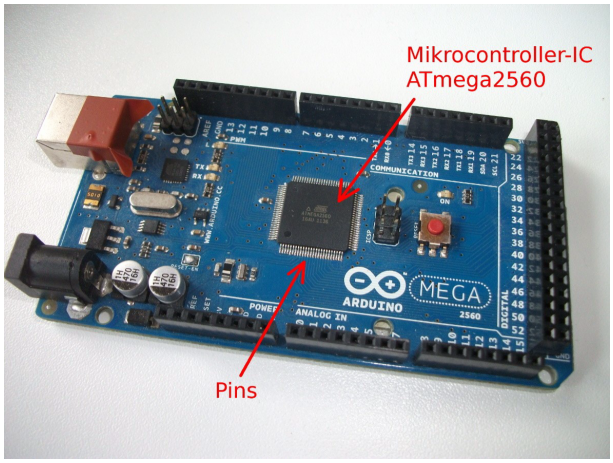

Abbildung 1: Pin eines Microcontrollers

Die meisten Pins der AVR-Mikrocontroller sind so genannte GPIO-Pins: General Purpose Input Output Pins – Allgemein verwendbare Ein-/Ausgangs-Pins.Sie können entweder als digitale Eingänge oder aus digitale Ausgänge verwendet werden.

Als Eingang verwendeter Pin: Eine externe Schaltung (z.B: ein Thermostat) legt über eine Verbindungsleitung eine Spannung von entweder 0 (Low-Pegel) oder 5V (High-Pegel) an einen Mikrocontroller-Pin. Diese Spannung wird gegen die gemeinsame GND-Verbindungsleitung gemessen. Des Programm, das im Mikrocontroller läuft, kann diesen Spannungspegel in Form einer Ein-Bit-Information (0 oder 1) einlesen. Im Falle des Thermostats bedeuten z.B: 0V bzw. eine eingelesene logische 0, dass die Raumtemperatur unter 20°C liegt, und ein High-Pegel bzw. eine vom Programm eingelesene logische 1 bedeuten über 20°C.

Als Ausgang verwendeter Pin: Das Programm, das im Mikrocontroller läuft, erzeugt nun durch Schreiben einer 1 bzw. einer 0 auf einen speziellen Speicherplatz einen High- oder Low-Spannungspegel (5V / 0V) zwischen einem seiner Pins und GND. Eine über eine Verbindungsleitung an den Pin angeschlossene externe Schaltung (z.B. eine Schaltung, die ein Ofen ein bzw. aus-schaltet) liest diesen Spannungspegel ein, und agiert dementsprechend. Im Falle der Ofen-Ansteuerung könnte ein am Mikrocontroller-Pin ausgegebener Low-Spannungspegel bedeuten, dass die Schaltung den Ofen ausschaltet, und bei einem High-Spannungspegel heizt der Ofen.

> **Wichtig!**
>
> *Üblicherweise sind die meisten Pins eines Mikrocontrollers GPIO-Pins (General Purpose Input/Output). Sie können entweder als Eingänge oder als Ausgänge verwendet werden. Das im Mikrocontroller laufende Programm muss jeden verwendeten GPIO-Pin zu einem Ein- oder zu einem Ausgang programmieren. Das passiert in der Initialisierungsphase des Programms.*

Port: Mikrocontroller-Pins sind in Achter-Gruppen zusammengefasst. Das heißt 8 Pins gemeinsam bilden einen Port. Die Ports heißen Port A, Port B, Port C, usw.

Jeder der acht Pins eines Ports hat eine Nummer zwischen 0 und 7, um den Pin eindeutig identifizieren zu können. Der Namen eines Pins setzt sich bei den AVR-Mikrocontrollern zusammen aus dem Buchstaben „P“, dem Kenn-Buchstaben des Ports („A“, „B“, „C“, usw.) und der Nummer des Pins zwischen 0 und 7.

Zum Beispiel hat der Pin mit der Nummer 0 vom Port B die Bezeichnung PB0. Die Pins PB0, PB1, PB2 bis PB7 gemeinsam sind der Port B.

> **Beispiel**
>
> Der Pin PD2 ist der Pin mit der Nummer 2 von Port D. Da die Pins von 0 bis 7 durchnummeriert sind, könnte man auch sagen, dass PD0 der erste, PD1 der zweite und PD2 eigentlich der dritte Pin von Port D ist. Um Verwirrung zu vermeiden spricht man bei PD2 aber trotzdem vom Pin Nummer 2 (oder zweiten Pin) von Port D. PD0 ist somit der „nullte“ Pin von Port D.

> **Wichtig!**
>
> *Ports sind Achter-Gruppen von GPIO-Pins. Der Name eines GPIO-Pins setzt sich zusammen aus „P“, dem Kennbuchstaben des Ports, und seiner Nummer (0 bis 7).*

Das folgende Bild zeigt einen Auszug der Pinbelegung (pinout) des ATmega2560. Beispielhaft sind die acht Pins des Port L (Gehäuse-Nummer 35 bis 42) hervorgehoben.

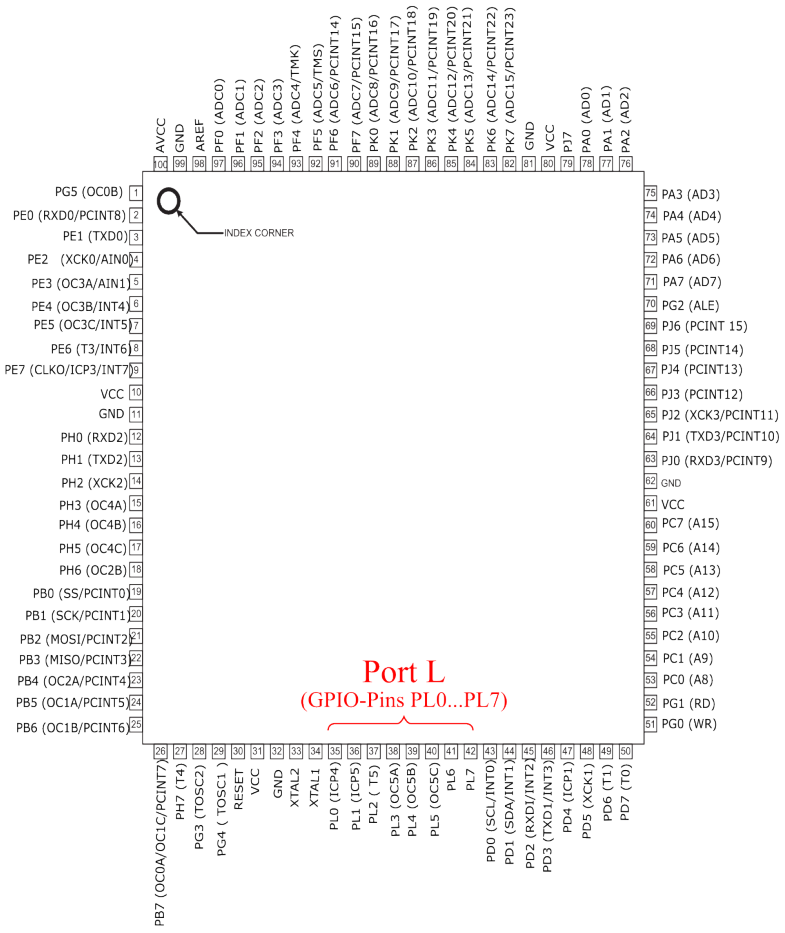

Abbildung 2: Pinbelegung des Atmega 2560

Alternative Funktionen von GPIO-Pins

Die meisten GPIO-Pins haben eine oder mehrere sogenannte alternative Funktion. Das bedeutet, dass der Pin entweder als GPIO-Pin (Ein- oder Ausgang) verwendet werden kann, oder es wird eine seiner alternativen Funktionen aktiviert, und der Pin kann nicht mehr als Ein- oder Ausgang verwendet werden.

Betrachten Sie die Pins PE0 und PE1 in der folgenden Abbildung. Die Pins haben die Gehäuse-Nummern 2 und 3. Pin PE0 hat zwei alternative Funktionen, nämlich „RXD0“ und „PCINT8“. Pin PE1 hat die alternative Funktion TXD0.

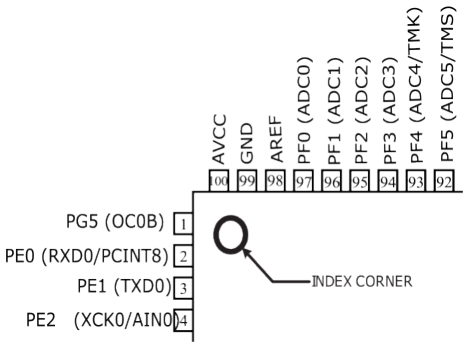

Abbildung 3: Pins PE0 und PE1 des Atmega 2560

Beim Arduino-Mega2560 Board (das als Herz den ATmega2560 Mikrocontroller verwendet), werden diese beiden Pins als serielle Schnittstelle verwendet. Die GPIO-Pins PE0 und PE1 können daher nicht als Ein- oder Ausgänge verwendet werden, sondern es ist ihre alternative Funktion RXD0 bzw. TXD0 aktiv.

Nach einem Reset sind (fast) alle GPIO-Pins als GPIO-Pins initialisiert, das heißt ihre alternativen Funktionen sind nicht aktiv. Eine Ausnahme sind die Pins PE0 bzw. PE1, die nach einem Reset zwar GPIO-Pins sind, bei denen aber vom Bootloader-Programm die alternative Funktion RXD0 bzw. TXD0 aktiviert wird. Der Bootloader wird nach einem Reset als erstes vor dem Benutzer-Programm ausgeführt.

> **Wichtig!**
>
> *GPIO-Pins haben eine oder mehrere alternative Funktionen. Ist eine alternative Funktion aktiviert, so kann der Pin nicht mehr als normaler Ein- oder Ausgang verwendet werden.*

Wenn eine alternative Funktion aktiviert wird, kann es sein, dass nur ein Teil der GPIO-Funktionalität deaktiviert wird. Zum Beispiel kann beim Timer/Counter1-Ausgang OC1A immer noch mittels GPIO-Funktionalität kontrolliert werden, ob der Pin PB5/OC1A ein Eingang oder Ausgang sein soll. Allerdings kann im Falle eines Ausgangs nicht mehr kontrolliert werden, ob der ein High- oder Low-Pegel ausgegeben wird, da diese Funktion der Timer/Counter1 übernimmt.

Pin-mapping der Arduino-Boards

Bei den Arduino-Boards verbinden Leiterbahnen die GPIO-Pins des Mikrocontrollers mit Stiftleisten. Die einzelnen Anschlüsse der Stiftleisten sind mit Nummern beschriftet, die nur bei Verwendung der Arduino core-library zum Programmieren verwendet werden können.

Das folgende Bild zeigt, mit welchen GPIO-Pins die einzelnen Anschlüsse der Stiftleisten verbunden sind („pin-mapping“).

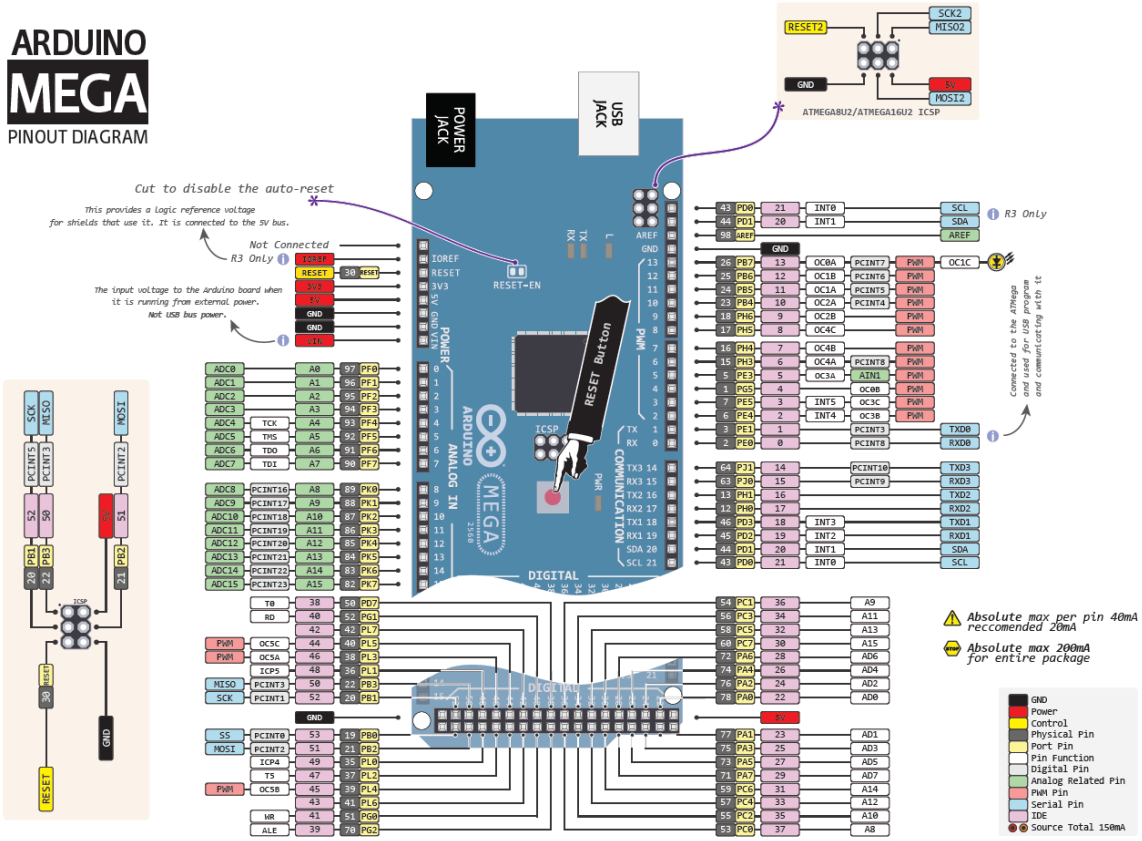

Abbildung 4: Arduino Mega Pinout Diagramm

GPIO-Pin als Ausgang

Wird vom Programm aus ein GPIO-Pin als Ausgang programmiert, so gibt der Mikrocontroller entweder einen High- oder einen Low-Pegel aus (5V gegen GND bzw. 0V gegen GND). Vom Programm aus kann der gewünschte Pegel vorgegeben werden, und zwar durch Schreiben einer 1 bzw. einer 0 in ein Bit in einem Special-Function-Register (oder durch Verwenden einer Bibliotheks-Funktion).

Das folgende Bild zeigt, die Verhältnisse, wenn der Pin PC6 vom Programm, das in der CPU läuft, zu einem Ausgang programmiert wird, der einen High-Pegel ausgibt.

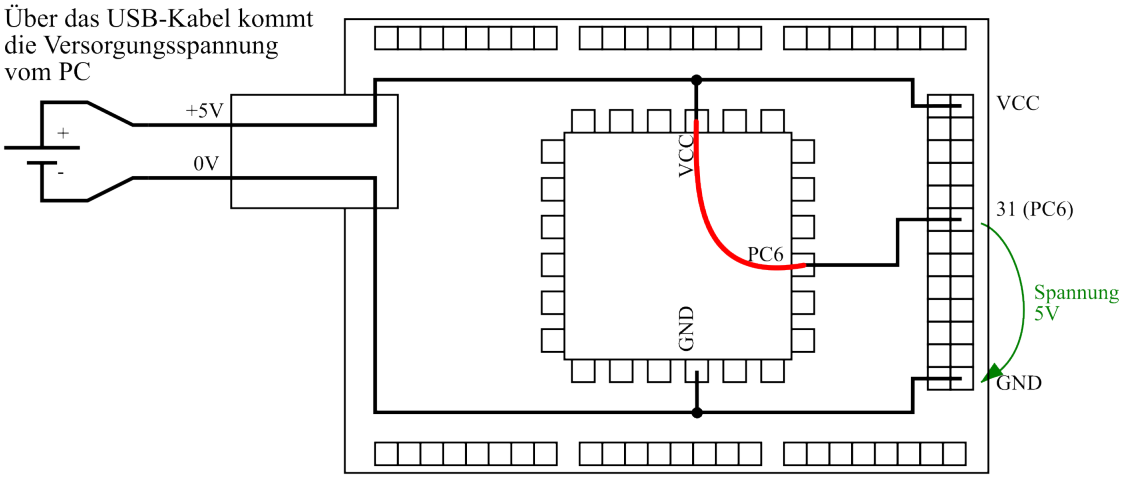

Abbildung 5: High Pegel

Im Bild ist das Arduino-Board mit seinen Stiftleisten und dem ATmega-Mikrocontroller symbolisch dargestellt. Über zwei Drähte im USB-Kabel erhält das Board seine Spannungsversorgung von 5V (Die im Bild mittels Batteriesymbol dargestellte Quelle ist tatsächlich ein Schaltnetzteil im PC). Auf dem Board ist der 5V-Anschluss vom USB-Stecker einerseits mit den VCC-Pins des Mikrocontrollers, andererseits mit einigen VCC-Stiftleisten-Anschlüssen mittels Leiterbahn verbunden. Der 0V-Anschluss vom USB-Stecker ist mit den GND-Pins des Mikrocontrollers und außerdem mit GND-Stiftleisten-Anschlüssen verbunden.

Da der Pin PC6 als Ausgang programmiert ist, der einen High-Pegel ausgibt, schaltet sich im Inneren des Mikrocontrollers ein Schalt-Transistor ein der im Mikrocontroller eine Verbindung zwischen dem VCC-Pin und dem PC6-Pin herstellt. Diese Verbindung durch den Schalt-Transistor ist im Bild rot dargestellt.

Durch diese Verbindung ist der Stiftleistenanschluss 31, der mit dem GPIO-Pin PC6 per Leiterbahn verbunden ist, auch mit dem Plus-Pol der Versorgungsquelle verbunden. Da alle GND-Stiftleistenanschlüsse direkt mit dem Minus-Pol der Versorgungsquelle verbunden sind, ist die Spannung zwischen Stiftleistenanschluss 31 und dem und GND-Stiftleistenanschluss gleich der Versorgungsspannung von 5V, also dem gewünschten High-Pegel.

Im folgenden Bild ist PC6 immer noch als Ausgang programmiert, aber das Programm gibt einen Low-Pegel aus.

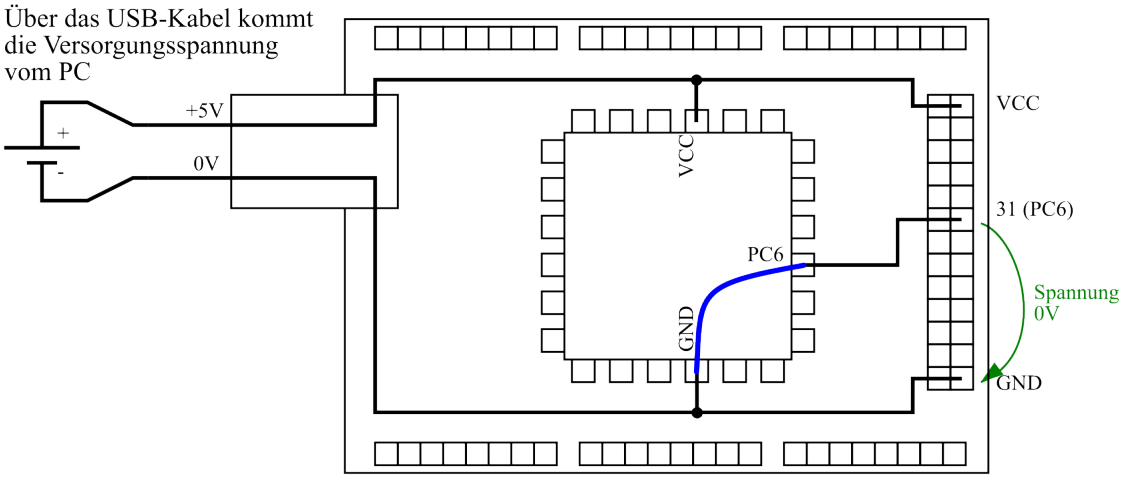

Abbildung 6: Low Pegel

Sobald das im Mikrocontroller laufende Programm einen Low-Pegel am Ausgangs-Pin PC6 ausgibt, schaltet sich im Inneren des Mikrocontrollers ein Schalt-Transistor ein, der eine Verbindung zwischen dem Pin PC6 und dem GND-Pin herstellt. Diese Verbindung ist im Bild blau eingezeichnet.

Der Stiftleistenanschluss 31, der über eine Leiterbahn mit dem GPIO-Pin PC6 verbunden ist, ist über diese interne Verbindung nun mit GND verbunden. Der Stiftleisten-Anschluss 31 und der GND-Stiftleistenanschluss sind daher direkt miteinander verbunden, weswegen die Spannung zwischen diesen beiden Anschlüssen gleich 0V ist. Es ergibt sich also der gewünschte Low-Pegel.

> **Wichtig!**
>
> *Programmiert das Mikrocontroller-Programm einen GPIO-Pin zu einem Ausgang und gibt einen High-Pegel aus, so verbindet ein Schalt-Transistor im Inneren des Mikrocontrollers den GPIO-Pin mit VCC. Bei einem Low-Pegel verbindet ein anderer Schalt-Transistor den GPIO-Pin mit GND.*

Die beiden Schalt-Transistoren im Mikrocontroller, die einen GPIO-Pin mit entweder VCC oder GND verbinden, haben im eingeschalteten Zustand einen nicht zu vernachlässigenden ohmschen Widerstand. Einerseits sorgt dieser ohmsche Widerstand für Kurzschlusssicherheit: Sollte ein Ausgangs-GPIO-Pin versehentlich gegen GND oder gegen VCC kurzgeschlossen werden, so werden die Schalt-Transistoren nicht thermisch zerstört. Andererseits sollte Strom, der in oder aus dem GPIO-Pin fließt, 10mA bis 20mA nicht übersteigen.

LED an einem GPIO-Pin anschließen

Zuerst sollte vorweggenommen werden, dass eine LED immer in Serie mit einem Vorwiderstand an den Mikrocontroller angeschlossen werden soll (ca. 330 Ω, um den LED-Strom auf ca. 10mA zu begrenzen).

Es gibt zwei Möglichkeiten, eine LED durch einen Mikrocontroller anzusteuern. In beiden Fällen muss der zugehörige GPIO-Pin zu einem Ausgang programmiert werden.

Das folgende Bild zeigt die erste Variante. Die Serienschaltung aus Vorwiderstand und LED ist zwischen GPIO-Pin PC6 und GND geschaltet.

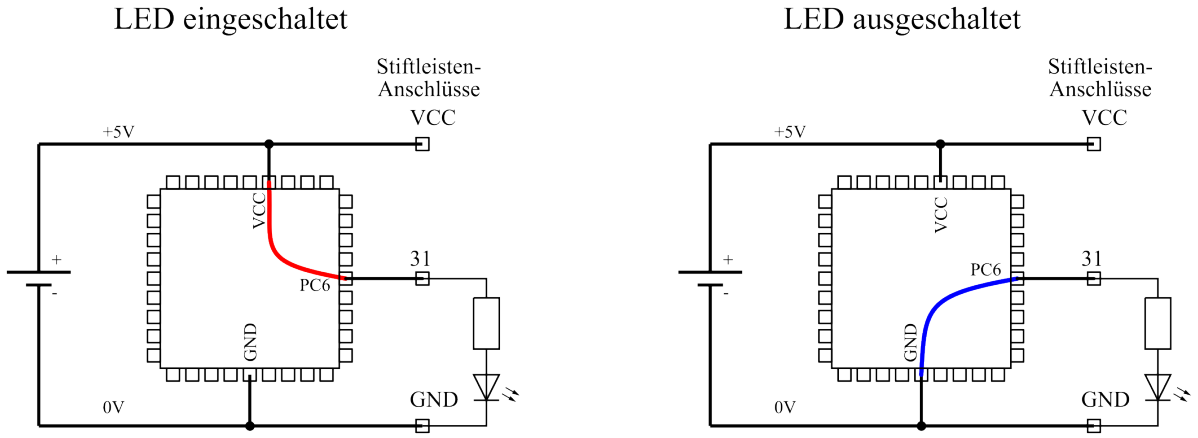

Abbildung 7: Erste Variante zum Anschließen von Leds

Wenn das Mikrocontroller-Programm einen High-Pegel am GPIO-Pin PC6 ausgibt, so ist die LED eingeschaltet: Es fließt Strom vom Pluspol der Quelle über die VCC-Verbindungsleitung zum VCC-Pin des Mikrocontrollers, weiter über den Schalt-Transistor (rote Verbindung) zum Pin PC6, weiter zum Stiftleistenanschluss 31, über Vorwiderstand und LED und die GND-Leitung zurück zum Minus-Pol der Quelle.

Wenn das Mikrocontroller-Programm einen Low-Pegel am GPIO-Pin PC6 ausgibt, leuchtet die LED nicht. Die beiden Anschlüsse der Widerstands-LED-Serienschaltung sind über den anderen Schalt-Transistor (blaue Verbindung) direkt miteinander verbunden. Es kann kein Strom vom Plus- zum Minus-Pol der Quelle über die Serienschaltung fließen.

> **Wichtig!**
>
> *Wenn die Serienschaltung aus Vorwiderstand und LED zwischen einen GPIO-Pin (Ausgang) und GND geschaltet wird, so leuchtet die LED bei einem High-Pegel und ist ausgeschaltet bei einem Low-Pegel.*

Bei der zweiten Variante ist die Serienschaltung aus Vorwiderstand und LED zwischen VCC und dem GPIO-Pin angeschlossen.

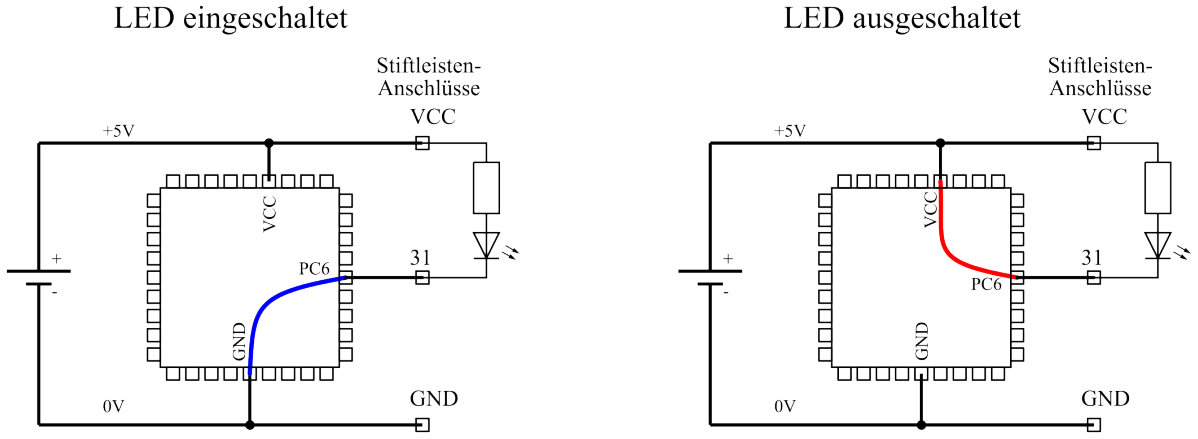

Abbildung 8: Zweite Variante zum Anschließen von Leds

Um die LED zum Leuchten zu bringen, muss das Mikrocontroller-Programm nun einen Low-Pegel ausgeben. Dann kann Strom vom Plus-Pol der Quelle über Widerstand und LED zum Minus-Pol der Quelle fließen, wie man im linken Teilbild sehen kann.

Die LED leuchtet nicht, wenn das Mikrocontroller-Programm einen High-Pegel ausgibt. Im rechten Teilbild sieht man, dass in diesem Fall kein Pfad für einen Stromfluss vom Plus-Pol zum Minus-Pol der Quelle existiert, der auch über die Serienschaltung führt.

> **Wichtig!**
>
> *Wenn die Serienschaltung aus Vorwiderstand und LED zwischen VCC und einen GPIO-Pin (Ausgang) geschaltet wird, so leuchtet die LED bei einem Low-Pegel und ist ausgeschaltet bei einem High-Pegel.*

GPIO-Pin als Eingang

Wenn das Programm im Mikrocontroller einen GPIO-Pin zu einem Eingang programmiert, so muss eine externe Schaltung den Pin entweder mit VCC verbinden und damit eine Spannung von 5V zwischen dem Pin und GND erzeugen (High-Pegel). Oder die externe Schaltung verbindet den GPIO-Pin mit GND, wodurch die sie eine Spannung von 0V zwischen Pin und GND hervorruft (Low-Pegel).

Das Programm im Mikrocontroller kann einlesen, welchen der beiden Spannungspegel die externe Schaltung am Pin hervorruft.

Im folgenden Bild sind zwei Beispiele dargestellt. Im linken Teilbild verbindet ein Draht den zu einem Eingang programmierten GPIO-Pin mit VCC.

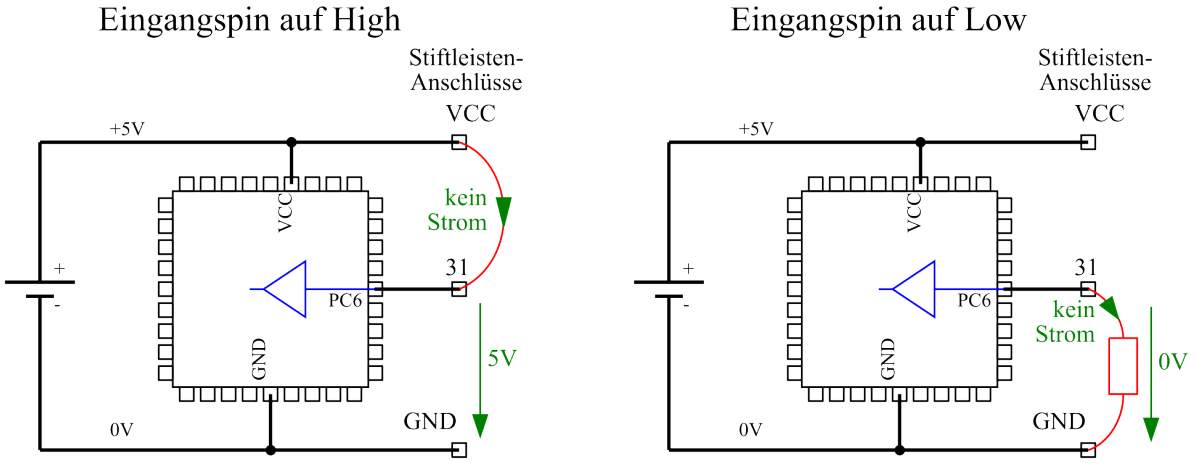

Abbildung 9: GPIO Pin als Eingang

Das im Inneren des Mikrocontrollers dargestellte Dreieck stellt einen Pufferverstärker dar (input-buffer). Sie können sich vorstellen, dass dieser Puffer-Verstärker den Spannungspegel am GPIO-Pin misst, und einen Low-Pegel in eine logische 0 für das laufende Programm, bzw. einen High-Pegel in eine logische 1 umwandelt.

Im linken Teilbild ist die Spannung zwischen GPIO-Pin und GND gleich 5V, da der Pin über den Draht und die VCC-Leitung mit dem Plus-Pol der Versorgungsquelle verbunden ist, der

GND-Anschluss hingegen mit dem Minus-Pol. Das Programm liest daher eine logische 1, wenn es den Spannungszustand des Pins abfragt.

Der Eingang des Puffer-Verstärkers hat einen unendlich hohen Eingangswiderstand, das heißt, es fließt kein Strom in oder aus dem Puffer-Verstärker.

Im rechten Teilbild ist der GPIO-Pin mit GND verbunden. Allerdings wird für die Verbindung an Stelle eines Drahts ein Widerstand verwendet. Da beim Eingang des Pufferverstärkers kein Strom fließt, fließt aber auch über den Widerstand kein Strom. Daher macht es in diesem Fall keinen Unterschied, ob für die Verbindung zwischen GPIO-Pin und GND ein Draht oder ein Widerstand verwendet wird: Laut ohmschem Gesetz fällt an einem Widerstand, über den ein Strom von null Ampere fließt, eine Spannung von null Volt ab.

Im rechten Teilbild liest das Programm wegen der Spannung von 0V zwischen GPIO-Eingang und GND eine logische 0 ein, wenn es den Spannungszustand des GPIO-Pins abfragt.

> **Wichtig!**
>
> *Wird ein GPIO-Pin als Eingang programmiert, muss eine externe Schaltung eine Spannung von 5V (High-Pegel) oder 0V (Low-Pegel) zwischen Pin und GND erzeugen. Meist geschieht das, indem die externe Schaltung den Eingangspin des Mikrocontrollers mit VCC oder mit GND verbindet. Der von der externen Schaltung angelegte Spannungspegel kann vom Programm eingelesen werden.*

Der interne pullup-Widerstand

Eine externe Schaltung muss einen als Eingang programmierten GPIO-Pin entweder mit GND verbinden oder mit VCC, damit die Spannung zwischen Pin und GND entweder 5V oder 0V beträgt. Es gibt aber auch noch einen dritten Fall: Die externe Schaltung verbindet den Pin nirgendwo hin, der Pin ist also offen (potenzialfrei).

Ein offener GPIO-Eingangs-Pin wird durch elektrische Felder (verursacht von benachbarten Leiterbahnen) beeinflusst, und die Spannung zwischen dem Pin und GND ist völlig undefiniert. Dementsprechend ist nicht vorhersagbar, ob das Programm eine logische 0, oder eine logische 1 einliest, wenn es den Spannungspegel des Eingangs-Pin einliest.

Um diesen Fall zu verhindern, besteht bei den AVR-Mikrocontrollern für jedem zu einem Eingang programmierten GPIO-Pin die Möglichkeit, einen so genannten internen pullup-Widerstand zu aktivieren.

Im folgenden Bild sind zwei Fälle dargestellt, bei denen ein GPIO-Pin zu einem Eingang mit akti- viertem pullup-Widerstand programmiert wurde.

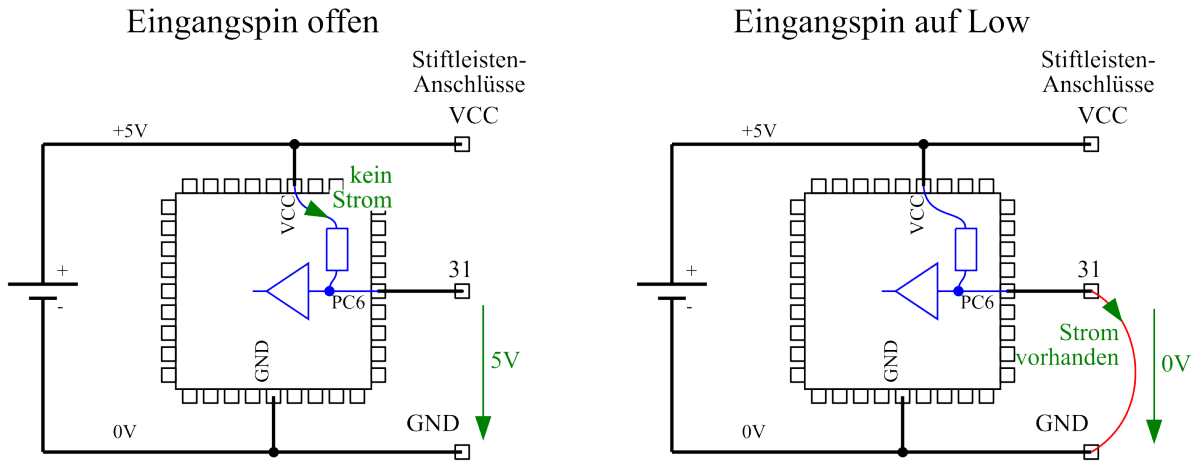

Abbildung 10: GPIO Eingangs-Pin mit aktiviertem pullup-Widerstand

Im linken Teilbild gibt es keine externe Schaltung, die den GPIO-Eingangs-Pin mit GND oder VCC verbindet. Der Pin wäre potenzialfrei, wenn der interne pullup-Widerstand nicht aktiviert worden wäre. Über diesen internen pullup-Widerstand fließt kein Strom, da in den Eingang des Puffer-Verstärkers kein Strom fließen kann. Der relativ hochohmige interne pullup-Widerstand (ca. 20 $k\Omega$) verbindet den GPIO-Pin mit VCC, und somit entsteht ein gültiger High-Pegel.

Im rechten Teilbild verbindet ein Draht (oder ein geschlossener Taster) den GPIO-Eingangs-Pin mit GND. In diesem Fall fließt Strom vom Plus-Pol der Quelle über die VCC-Leitung zum VCC-Pin und weiter über den internen pullup-Widerstand. In den Eingang des Puffer-Verstärkers fließt nach wie vor kein Strom, aber der Strom fließt weiter über den externen Draht zur GND-Leitung und zum Minus-Pol der Quelle. Am Draht fällt keine Spannung ab, so dass die Spannung zwischen GPIO-Pin und GND gleich null Volt ist. Es entsteht somit ein Low-Pegel.

> **Wichtig!**
>
> *Ein verwendeter Eingang darf nicht potenzialfrei bleiben. Um das zu verhindern, wird bei nicht beschalteten Eingängen der interne pullup-Widerstand aktiviert. Er verbindet hochohmig den GPIO-Eingangs-Pin mit VCC.*

Anschließen eines Tasters an den Mikrocontroller

Es gibt drei Möglichkeiten einen Taster an einen AVR-Mikrocontroller anzuschließen.

Externer pull-down Widerstand

Bei der ersten Variante wird der Taster zwischen VCC und den GPIO-Eingangs-Pin geschaltet, und ein zusätzlich verbindet ein Widerstand (typischerweise 10 $k\Omega$) den Pin mit GND. Der interne pullup-Widerstand des GPIO-Eingangs-Pins ist nicht aktiviert.

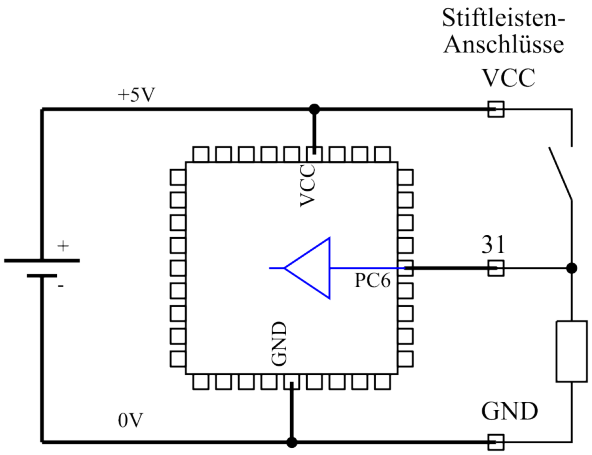

Abbildung 11: Taster an AVR Mikrocontroller mit pull-down Widerstand

Bei offenem Taster entsteht ein Low-Pegel, da kein Strom über den Widerstand fließen kann, und der Widerstand den Pin mit GND verbindet.

Bei geschlossenem Taster fließt Strom vom Plus-Pol über den geschlossenen Taster und den Widerstand zum Minus-Pol der Quelle. Da bei gedrücktem Taster der Pin mit VCC verbunden ist, entsteht ein High-Pegel. Wenn das Mikrocontroller-Programm den Zustand des Pins einliest, liest es daher eine logische 1.

Da bei dieser Variante ein externer Widerstand den Pin mit GND verbindet, spricht man von einem externen pulldown-Widerstand.

> **Wichtig!**
>
> *Variante 1 für Taster: Taster zwischen VCC und GPIO-Eingangs-Pin, externer pull-down-Widerstand zwischen GPIO-Eingangs-Pin und GND. Ein gedrückter Taster erzeugt einen High-Pegel, ein offener Taster einen Low-Pegel.*

Externer pull-up Widerstand

Bei der zweiten Variante sind die Position von Taster und Widerstand vertauscht. Der Widerstand heißt nun externer pullup-Widerstand, da er den Pin „nach oben“ zu VCC verbindet.

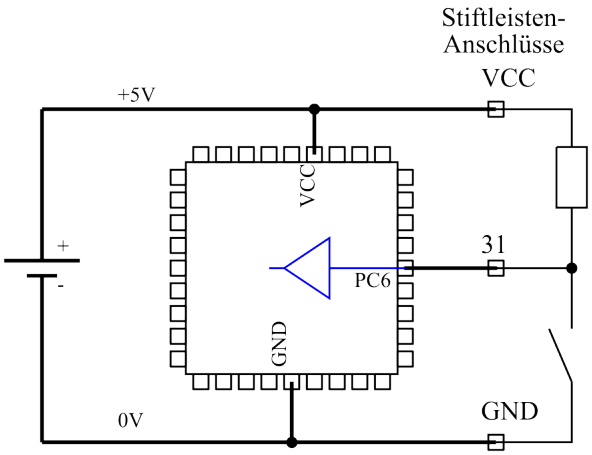

Abbildung 12: Taster an AVR Mikrocontroller mit pull-up Widerstand

Bei gedrücktem Taster verbindet der Taster den GPIO-Eingangs-Pin mit GND, es entsteht ein Low- Pegel. Wenn das Mikrocontroller-Programm zu diesem Zeitpunkt den Zustand des Eingangs-Pins einliest, liest es eine logische 0. Gleichzeitig fließt Strom vom Plus-Pol der Quelle, den Widerstand und den geschlossenen Taster zum Minus-Pol.

Bei offenem Taster verbindet der Widerstand, über den nun kein Strom fließt den GPIO-Eingangs-Pin mit VCC, wodurch ein High-Pegel entsteht.

> **Wichtig!**
>
> *Variante 2 für Taster: Externer pullup-Widerstand zwischen VCC und GPIO-Eingangs-Pin, Taster zwischen GPIO-Eingangs-Pin und GND. Ein gedrückter Taster erzeugt einen Low-Pegel, ein offener Taster einen High-Pegel.*

Interner pull-up Widerstand

Die dritte Variante gleicht der zweiten Variante, nur wird der externe pullup-Widerstand weggelassen, und stattdessen der interne pullup-Widerstand des GPIO-Eingangs-Pin aktiviert.

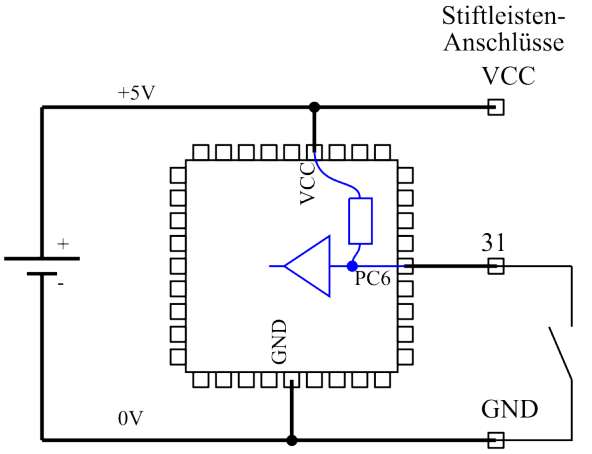

Abbildung 13: Taster an AVR Mikrocontroller mit internem pull-up Widerstand

Ein gedrückter Taster erzeugt einen Low-Pegel, wobei ein kleiner Strom über den internen pullup-Widerstand und den Taster fließt. Bei offenem Taster verhindert der interne pullup-Widerstand, dass der nicht angeschlossene GPIO-Eingangs-Pin potenzialfrei ist.

> **Wichtig!**
>
> *Variante 3 für Taster: Taster zwischen GPIO-Eingangs-Pin und GND. Der interne pullup-Widerstand des GPIO-Eingangs-Pins muss aktiviert werden. Ein gedrückter Taster erzeugt einen Low-Pegel, ein offener Taster einen High-Pegel.*