

Embedded Systems I

Lab 3: Einführung in die Arduino Plattform

Arduino Hardware

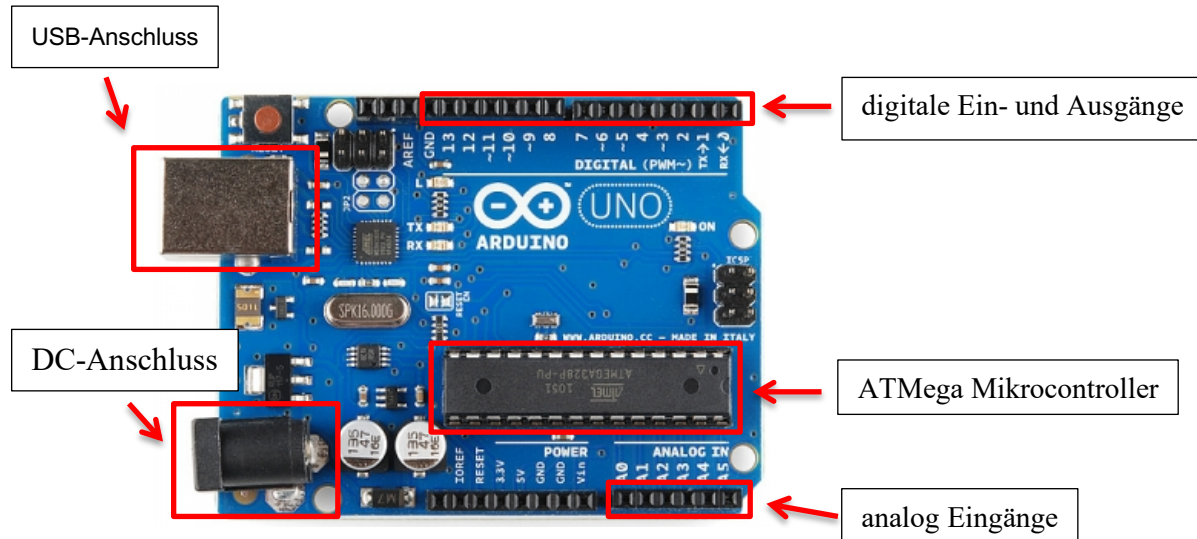


Abbildung 1 : Arduino Uno Board

Die verwendeten Arduino UNO Boards (Abbildung 1) bieten im Zusammenspiel mit der Entwicklungsumgebung eine kostengünstige OpenSource Plattform zur Realisierung erster selbstentwickelter Mikrocontrollerprojekte. Auf der Projekt Homepage: www.arduino.cc stehen verschiedene Versionen der Arduino Boards zur Verfügung, die je nach Aufgabengebiet umfangreicher oder anwendungsspezifischer ausgestattet sind. Boards der UNO Serie zeichnen sich durch ihren günstigen Anschaffungspreis, ihre Robustheit und eine vielfältige Einsatzmöglichkeit aus.

Arduino Boards basieren auf den weitverbreiteten ATMega Mikrocontrollern von ATMEL.

Anders als andere Entwicklungsumgebungen wird bei der Verwendung der Arduino-Plattform kein zusätzlicher, kostenintensiver, Programmieradapter benötigt.

Durch die Verwendung eines speziellen, auf dem Controller vorinstallierten Bootloaders, kann der Mikrocontroller des Arduino Boards direkt über den integrierten USB-Anschluss beschrieben werden.

Für Leistungsintensivere Versuchsaufbauten, die die maximale Leistungsabgabe des USB-Anschlusses überschreiten (max. 500mA), bietet das UNO Board die Möglichkeit eine zusätzliche DC-Spannungsquelle anzuschließen. Auf dem Board befinden sich Festspannungsregler für 3.3V und 5V,

die die angelegte Eingangsspannung umwandeln, um die gängigsten ICs, ohne zusätzliche Spannungsregler, direkt an dem Arduino Board betreiben zu können.

Für die in den Übungen dieses Moduls verwendeten Versuche wird keine zusätzliche Spannungsquelle benötigt.

Die Ein- und Ausgabemöglichkeiten (I/O) des UNO Boards gliedern sich in zwei Hauptgruppen. Die analog Eingänge befinden sich an der unteren Seite der Platine und können zum Einlesen analoger Spannungen genutzt werden. Die digitalen Ein- und Ausgabeanlüsse (oder auch Ein- und Ausgabepins) befinden sich auf der oberen Seite der Platine und werden für die Ein- und Ausgabe digitaler Signale verwendet. Das bedeutet sie können nur zwei verschiedene Spannungspegel annehmen (TRUE= 5 Volt und FALSE = 0 Volt). Sie können zum Beispiel zum Auswerten eines Schalters oder Tasters genutzt werden.

Die digitalen Anschlüsse des UNO Boards unterteilen sich nochmals in zwei Untergruppen. Die normalen Anschlüsse die ausschließlich für digitale ein und Ausgabe genutzt werden können und die auf der Platine mit einem ~ markierten Anschlüsse. Diese Pins können zur Ausgabe eines PWM-Signals (Pulsweitenmodulation) genutzt werden.

Bei der Pulsweitenmodulierung wird das Verhältnis zwischen Ein- und Ausschaltzeit eines digitalen Signals verändert um einem Verbraucher im arithmetischen Mittel mehr oder weniger elektrische Leistung zur Verfügung zu stellen. Die Pulsweitenmodulation arbeitet mit einer festen Frequenz. Wird diese Frequenz zu gering gewählt kommt es zu wahrnehmbarem Stocken oder Flackern des angeschlossenen Verbrauchers. Zum Beispiel würde eine mit 50% Leistung bei 1Hz angesteuerte LED für den Betrachter deutlich flackern, da sie eine halbe Sekunde mit 100% Leistung und eine halbe Sekunde 0% Leistung versorgt werden. Im arithmetischen Mittel würde die LED jedoch 50% Leistung zur Verfügung gestellt bekommen (Abbildung 2).

Die Arduino Boards besitzen 16kHz PWM Ausgänge, wodurch LEDs ohne, für das menschliche Auge sichtbares Flackern, gedimmt und Motoren (mit entsprechender Leistungsverstärkung) sanft angefahren werden können.

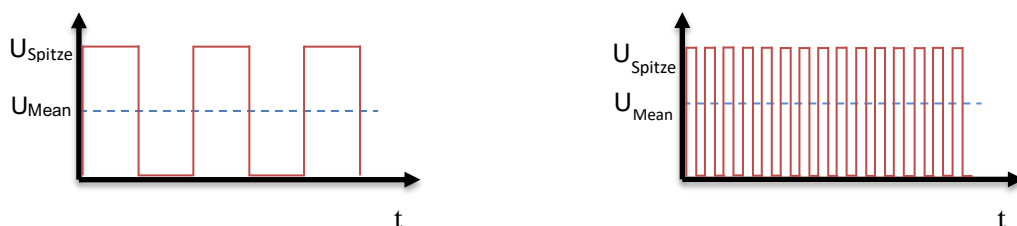


Abbildung 2: pulsweitenmodelliertes Signal

Arduino Software



Die Programmierung der Arduino Hardware erfolgt über die auf www.arduino.cc verfügbare, kostenlose, Entwicklungsumgebung.

Die Entwicklungsumgebung (Abbildung 3) ist in Java programmiert und basiert auf verschiedenen OpenSource Projekten. Die Entwicklungsumgebung ist speziell für den schnellen und unkomplizierten Einstieg in die Mikrocontrollerprogrammierung entwickelt worden und bietet daher eine überschaubare Oberfläche mit allen anfangs benötigten Werkzeugen.

Um mit dem Programmieren zu beginnen muss das Arduino-Board mit einem USB-Kabel mit dem PC verbunden und die Arduino Software gestartet werden. Bei erstmaliger Verwendung des Arduino-Boards an dem Computer ist es notwendig einen Treiber für den USB-Serial Converter des Arduino-Boards zu installieren, der den USB(Universal Serial Bus) als einen COM Anschluss des Computers virtualisiert.

Nach starten der Software wird das Arduino-Board automatisch erkannt und der Board-Typ, sowie der

verwendete serielle Anschluss werden am unteren Rand der Programmierumgebung angezeigt.

Das Programm, oder auch Sketch, wird in C-Code in der Entwicklungsumgebung programmiert. Nach erfolgreichen Kompilieren über den -Knopf am oberen Rand des Programmierfensters kann das Programm über Betätigung des  - Knopfes direkt auf den Mikrocontroller geladen werden. Um dem Nutzer dieses Vorgehen zu ermöglichen, ist auf dem mit dem Arduino Board gelieferten Mikrocontroller ein spezieller Bootloader installiert. Der Bootloader empfängt den Arduino Sketch über die serielle Verbindung und schreibt die empfangenen Daten in den Speicher des Mikrocontrollers. Am Ende des Schreibvorganges wird der Mikrocontroller automatisch neugestartet und führt den gespeicherten Programmablauf aus.

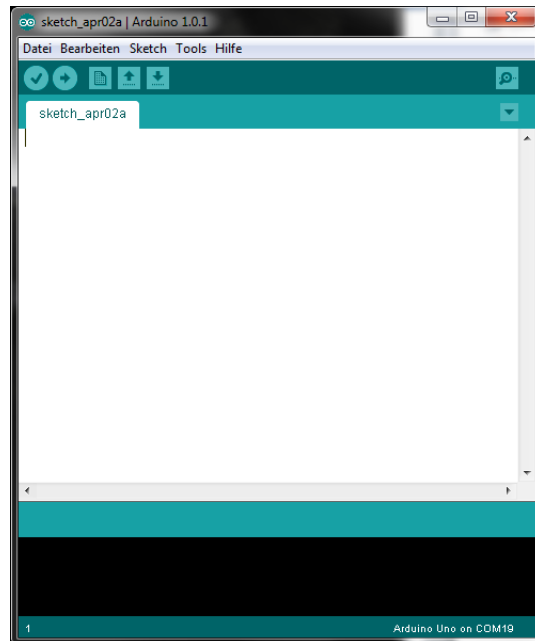


Abbildung 3: Entwicklungstool

Grundlegende Funktionen und Programmierung

Jedes Arduino Programm besteht aus einem Sketch-Ordner und mehrer Unterverzeichnisse. Um ein neues Programm zu erstellen, klicken Sie auf den Reiter „Datei“ gefolgt von „Neu“. Es wird ein leerer Sketch erstellt und geöffnet. Klicken Sie nun als erstes erneut auf „Datei“ → „Speichern unter“. An dieser Stelle werden Sie nach einem Namen und dem gewünschten Speicherort des Programms gefragt. Es ist empfehlenswert die Programme mit einem funktionsbeschreibenden Namen und einer Versionsnummer zu versehen.

Wie in Abbildung 4 zu erkennen verfügt die Arduino Entwicklungsumgebung über eine Highlighting Funktion, die die Lesbarkeit des Programmcodes erheblich verbessert. Bekannte Datentypen, Funktionen und Schleifen werden Orange dargestellt, während eingefügte Programmpassagen wie z.B. Additionen schwarz dargestellt werden.

Kommentare die durch „//“ oder „/* */“ gekennzeichnet werden, werden grau hinterlegt um zu signalisieren das der Compiler Sie nicht berücksichtigen wird.

Im unteren Bereich des Programmierfensters wird die aktuelle Größe des Programms im Speicher des Mikrocontrollers, sowie die maximal mögliche Größe des Programms angezeigt. Die maximal verfügbare Größe ist von dem auf dem Board verwendeten Mikrocontroller abhängig. Zum Beispiel verfügt ein Arduino MEGA Board, mit seinem ATmega 2560, über einen 8fach größeren internen Speicher (256kByte) als ein Arduino UNO Board mit dem verwendeten ATmega 328 (32kByte).

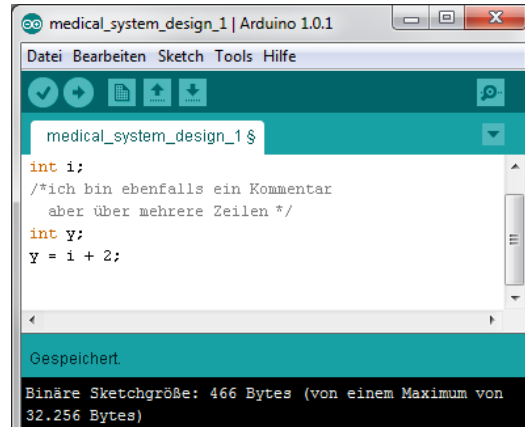


Abbildung 4: Highlighting

Einfache Arduino Programme teilen sich in zwei grundlegende Sektionen auf. Der „setup()“-Funktion (rot markiert in Abbildung 5) und der „loop()“ Funktion (blau markiert in Abbildung 5).

Die „setup()“-Funktion wird einmalig ausgeführt, sobald der Mikrocontroller an die Versorgungsspannung angeschlossen wird. Innerhalb dieser Funktion werden Grundlegende Einstellungen vorgenommen, die Sie im Verlauf des Moduls kennen lernen werden. In Abbildung 5 wird die Funktion „pinMode()“ verwendet um festzulegen, dass es sich bei dem Anschluss 2 um einen Eingang handeln soll und daher der Interne Pulluptransistor des Mikrocontrollers deaktiviert sein muss. Pin3 wird simultan dazu als Ausgang deklariert.

Die „loop()“-Funktion wird nach ausführen der „setup()“-Funktion gestartet und endlos wiederholt.

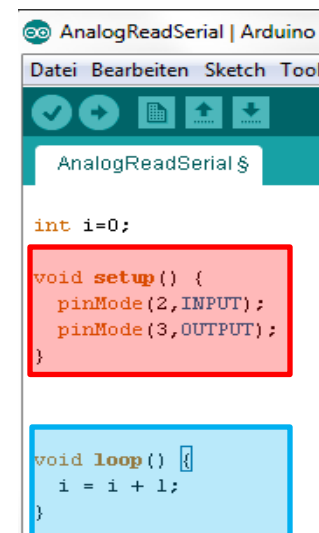





Abbildung 5: Sketch- Grundstruktur

Im Fall des Programms in Abbildung 5 wird die globale Zählervariable „i“ bei jedem Schleifendurchlauf um eins erhöht.

Die Arduino Entwicklungsumgebung verfügt über eine Debughilfe, die auftretende Fehler während der Kompilierung im unteren Teil des Programmfensters anzeigt. Auf der linken Seite der Abbildung 6 ist zu erkennen, dass der Compiler nach Drücken des -Knopfes, zum korrekten Kompilieren des Programmes, ein „;“ vor der „}“ erwartet. Nach der Korrektur des Fehlers und erneuten Betätigung des -Knopfes wird das Programm korrekt kompiliert und kann durch Drücken des -Knopfes auf das Arduino-Board übertragen werden.

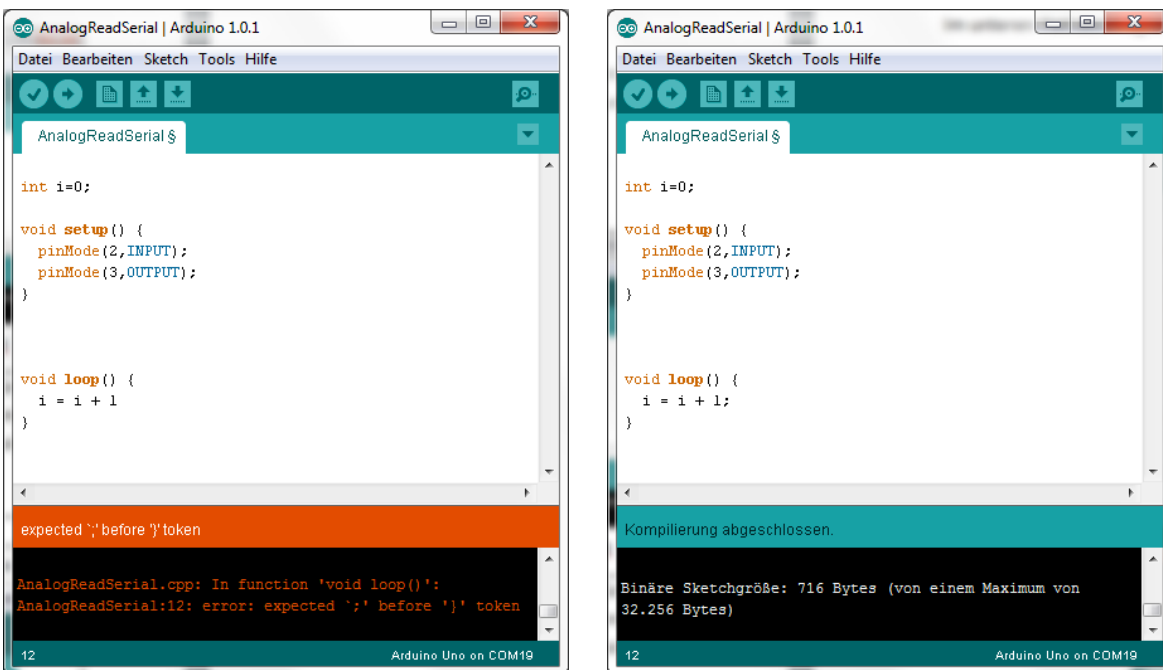


Abbildung 6: Debughilfe

Hinweise zur Versuchsdurchführung

Trennen Sie vor jeder Schaltungsveränderung das USB-Kabel von Ihrem PC um Beschädigungen des USB-Ports und des Mikrocontrollers zu vermeiden.

Vor der Inbetriebnahme einer neuen oder veränderten Schaltung lassen Sie diese durch die betreuenden Mitarbeiter kontrollieren.

Bauen Sie alle Schaltungen auf dem Ihnen zu Verfügung gestellten Steckboard auf und verbinden Sie die Schaltung über Kabelbrücken mit dem Mikrocontroller. Stecken Sie niemals Bauteile direkt in die Anschlüsse des Arduino UNO Boards.

Aufgabe 1:

Schalten Sie die integrierte LED an Pin 13 ein und aus.

Erläuterung der benötigten Hardware

Das Arduino UNO Board verfügt über eine integrierte gelbe LED. Diese ist über einen Vorwiderstand an Pin 13 angeschlossen und kann direkt angesteuert werden.

Die LED befindet sich wie in Abbildung 7 zu erkennen direkt neben Anschlusspin 13.

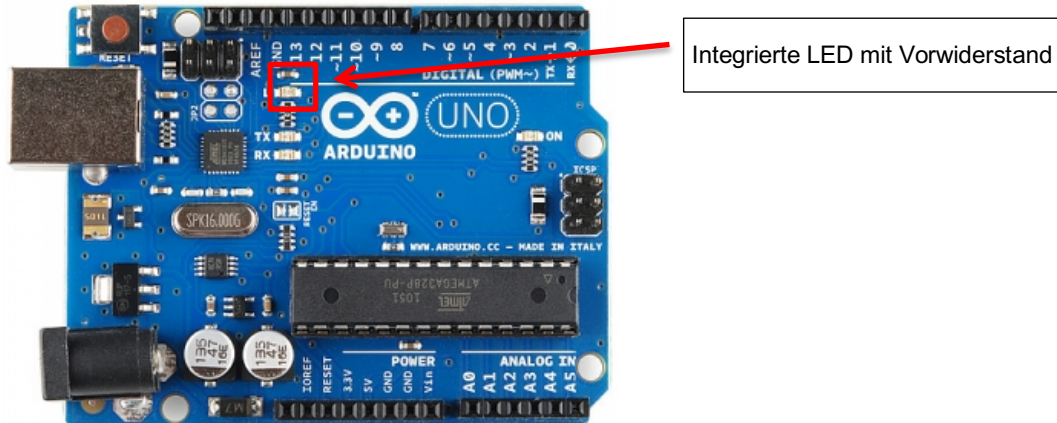


Abbildung 7 : integrierte LED mit Vorwiderstand

Erläuterung der benötigten Funktionen

pinMode(pin,mode)

Um die LED an dem Ausgangspin einschalten zu können, muss der Anschlusspin als Ausgang deklariert werden, damit die internen Funktionen des Mikrocontrollers die Ausgabe nicht behindern. Wird der Pin nicht als Ausgang deklariert, bleibt er in dem zuletzt verwendeten Modus

und es kann unter bestimmten Voraussetzungen zu Fehlfunktionen kommen (anlaufen von Aktoren, anzeige falscher Werte, etc.)

Um den Ausgang klar zu definieren wird die Funktion „pinMode()“ in der „setup()“ Funktion des Arduino Programms aufgerufen. Dabei wird der Funktion als erster Parameter einer der Anschlusspins (*pin*) von 0-13 und als zweiter Parameter der gewünschte Ausgangsmodus (*mode*), „OUTPUT“ oder „INPUT“, übergeben.

Beispielaufruf: `pinMode(3,OUTPUT);` //der Pin 3 wird als Ausgang definiert

digitalWrite(pin,value)

Aller digitalen Anschlüsse des Arduino Boards können entweder 0 Volt (LOW) oder 5 Volt (HIGH) annehmen. Die Spannungspegel der einzelnen Pins können über den Befehl „digitalWrite()“ beeinflusst werden. Die Funktion erwartet als ersten Parameter (*pin*) einen Anschlusspin, 0-13, und als zweiten Parameter (*value*) den gewünschten Pegel als HIGH oder LOW.

Beispielaufruf: `digitalWrite(3,HIGH);` //der Pin 3 nimmt einen Wert HIGH (5Volt) an

Aufgabe 2:

Schließen Sie eine externe LED an den Pin 8 des Arduino Boards an

Erläuterung der benötigten Hardware

Bei der Leuchtdioden, oder auch kurz LED (light-emitting-diode), handelt es sich um ein unipolares Halbleiterbauteil, das bei Anlegen einer Versorgungsspannung Licht emittiert. In der bedrahteten Bauform ist die Anode (+) an dem längeren Drahtanschluss und die Kathode (-) an dem kürzeren Drahtanschluss herausgeführt. Wird eine Gleichspannung an die Anode (+) und Kathode (-) der Diode angelegt, emittiert die Diode Licht.

Die digitalen Ausgänge des Arduino UNO Boards können bis zu 40mA Versorgungsstrom liefern. Bei den im Versuch verwendeten LEDs handelt es sich ausschließlich um

low-current LEDs die besonders stromsparend designt wurden. Um den Strom durch die LED zu begrenzen und die LED nicht zu zerstören, verschalten Sie als Vorwiderstand einen 330Ω Widerstand in Reihe zu der LED (Abbildung 8).

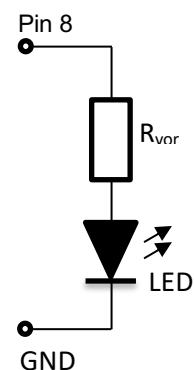


Abbildung 8: LED mit Vorwiderstand

Aufgabe 3:

Verschalten Sie einen Taster mit dem Pin 7 des Arduino Board und schalten Sie mit dem Taster die LED an Pin 8 an und aus.

Erläuterung der benötigten Hardware

Für diese Aufgabe verwenden Sie den Versuchsaufbau der Aufgabe 2 und verschalten zusätzlichen einen Kurzhubtaster mit dem Pin 7 des Arduino Boards. Der in dem Versuch verwendete Taster ist intern wie in Abbildung 9 zu erkennen verschaltet.

Wird der Taster ohne zusätzliche Beschaltung betrieben, liegen an dem Anschlusspin des Arduino Boards bei gedrücktem Taster 5 Volt (TRUE) und bei nicht gedrücktem Taster kein definierter Wert an.

Durch die Verwendung nicht abgeschirmter Kabel und dem Betrieb elektrischer Verbraucher (z.B. Schaltnetzteile) in der näheren Umgebung des Arduino Boards, kann es durch diesen nicht definierten Wert zu Fehlmessungen am Eingang des Mikrocontrollers kommen. Deshalb wird wie in Abbildung 10 zu sehen ein 10k Ω Widerstand als Pulldownwiderstand an dem Eingang des Arduino Boards verschaltet, um auch bei nicht betätigtem Taster ein klar definiertes Signal zu erhalten (0 Volt).

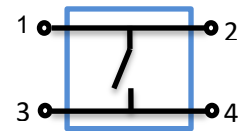


Abbildung 9:
Kurzhubtaster

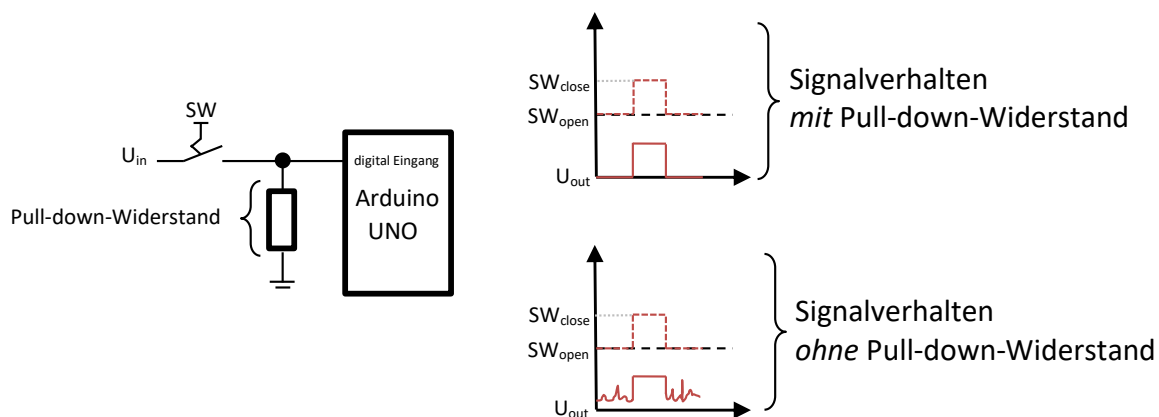


Abbildung 10: Eingangssignalverhalten

Erläuterung der benötigten Funktionen

digitalRead(pin)

Um den aktuellen Spannungspegel an dem Eingangspin einzulesen wird die Funktion „digitalRead()“ verwendet. Die Funktion erwartet als Übergabeparameter den Eingangspin (*pin*) an dem das Signal eingelesen werden soll und gibt als Rückgabewert den aktuellen Wert an diesem Anschlusspin zurück (HIGH/LOW).

Beispielaufruf: `i = digitalRead(3);` //Variable *i* speichert den aktuellen Wert an Pin 3 (HIGH/LOW)