Erasmus+ KA-202

Strategisches Partnerschaftsprojekt für berufliche Bildung und Ausbildung

Projekttitel: "Lehren und Lernen mit Arduinos in der Berufsbildung"

Projekt-Akronym: "ARDUinVET"

Projekt Nr.: “2020-1-TR01-KA202-093762”

ARDUinVET GUIDEBOOK

"Dieses Projekt wird durch das Erasmus+ Programm der Europäischen Union finanziert. Die Europäische Kommission und die türkische Nationalagentur können jedoch nicht für die Verwendung der darin enthaltenen Informationen verantwortlich gemacht werden."

Koordinator:

Gölbaşı Mesleki ve Teknik Anadolu Lisesi (Ankara / TURKEY)

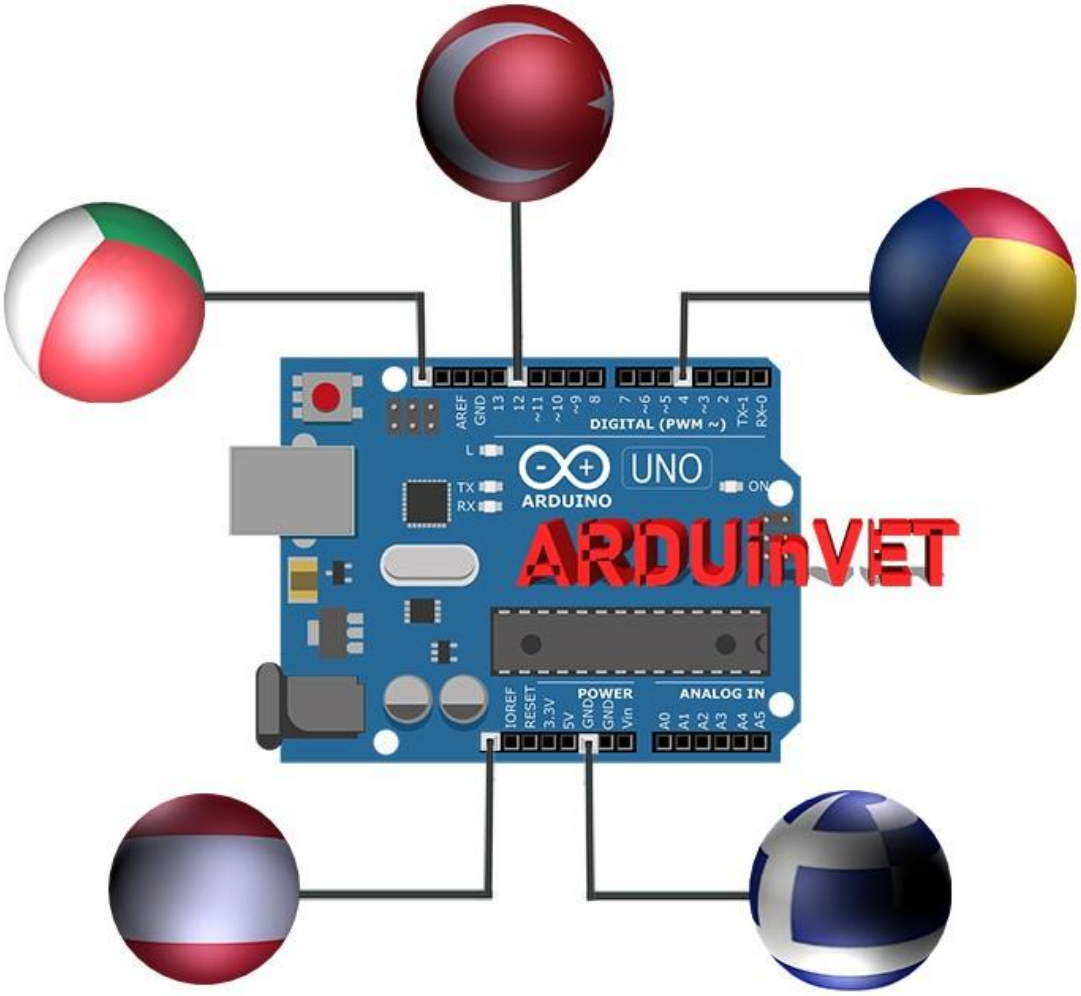
PARTNERS:

2 EK Peiraia (Piraeus / Greece)

Liceul Tehnologic Grigore Moisil Braila (Braila / Romania)

Istituto di Istruzione Superiore Einstein De Lorenzo Potenza (Potenza / Italy)

HTBLA Wolfsberg / (College for Engineering Wolfsberg) (Wolfsberg / Austria)



Zusammenfassung des Projekts

"Lehren und Lernen mit Arduinos in der Berufsausbildung"

Die Technologie entwickelt sich Tag für Tag sehr schnell. Es ist eine Tatsache, dass die technologischen Entwicklungen den Menschen dazu zwingen, sich in seinem/ihrem Beruf ständig weiterzuentwickeln. Mehr als in jedem anderen Bereich entwickeln sich die Elektronik und die Informations- und Kommunikationstechnologie am schnellsten. Diese Entwicklungen haben dazu geführt, dass immer komplexere Steuerungssysteme hergestellt werden. Arduinos werden nun zur Steuerung dieser komplexen Systeme verwendet.

Da wir Bildungseinrichtungen sind, die Technologie unterrichten, müssen wir die Entwicklungen in der Welt verfolgen. Das Projekt "Lehren und Lernen von Arduinos in der Berufsausbildung" zielt darauf ab, Arduino-Anwendungen an die Berufsausbildung anzupassen und ein effizienteres Trainingsset und einen Leitfaden für die Labors und Werkstätten von Berufs- und Technikstudenten zu entwickeln.

Das Hauptziel dieses Projekts ist die Entwicklung eines Leitfadens für gute Praktiken und eines Arduino-Ausbildungssets, die Einführung von Arduino-Ausbildungsmodellen in andere TeilnehmerInnen während ihrer Besuche im Gastland, der Vergleich verschiedener Bildungssysteme und Ausbildungsmethoden mit anderen teilnehmenden Schulen und der Austausch bewährter Praktiken. TeilnehmerInnen der Projekte: LehrerInnen aus den Bereichen Elektrotechnik, Elektronik, IKT und Automatisierung in der beruflichen Bildung. Die Teilnehmer unterrichten Arduino in berufsbildenden Schulen. Es gibt einen Koordinator und insgesamt 4 Partner im Projekt. Die Partner sind Berufsschulen aus der Türkei, Italien, Rumänien, Österreich und Griechenland. Die Gesamtzahl der Mobilitäten im Projekt beträgt 40. Während des Projekts werden insgesamt 5 TPMs durchgeführt, wobei jeder Partner mit 2 Teilnehmern an jedem TPM teilnimmt. Ein TPM wird in jedem Land stattfinden.

Mit den Projektaktivitäten wird sichergestellt, dass die besten Praktiken für den Arduino-Unterricht ausgetauscht werden und die Partner diese an ihre Bildungsumgebung anpassen. Die Projektpartner werden Arduino-Experimentierkästen und einen Leitfaden für bewährte Verfahren erstellen. Unsere Projektaktivitäten zielen darauf ab, ein erfolgreiches Lern- und Lehrumfeld für alle Lehrer und Schüler der beruflichen Bildung zu schaffen. Voraussetzung für erfolgreiches Lernen ist, dass die Lehrer ihre Rolle bei der Förderung des Lernens stärken. Die Lehrer brauchen neue Experimentierkästen und Module für Innovation, Teamarbeit, Feedback und Bewertung. Die Lehrkräfte sollten die Möglichkeit erhalten, sich beruflich weiterzuentwickeln.

Die wichtigsten Ergebnisse des Projekts sind: ein Arduino-Schulungsset für den Arduino-Unterricht in der beruflichen Bildung und ein Leitfaden für dieses Set, eine Projektwebsite und eine Projekt-DVD. Die Methodik, die bei der Durchführung des Projekts angewendet wird, ist "Make-Develop-Share". In unserem Projekt werden gute Praktiken zuerst hergestellt, dann entwickelt und schließlich geteilt. In unserem Projekt ist alles offen und transparent. Jede Aufgabe und Verantwortung wird im Projekt aufgezeichnet oder schriftlich festgehalten. Die Produkte des Projekts sind nicht nur schriftliche oder

dokumentarische Produkte, sondern auch ein praktisches Arduino-Schulungsset und Bausätze.

Langfristig wollen wir das Projekt an Lehrer, Schüler und Berufsschulen, lokale Bildungseinrichtungen, den Elektronik- und IKT-Arbeitsmarkt weitergeben. Wir glauben, dass die Ergebnisse und Produkte unserer Projekte durch die Verbreitungsaktivitäten kurz- und langfristige Auswirkungen auf die berufliche Bildung und den Arbeitsmarkt haben werden. Das Gesamtbudget für das Projekt mit 5 Partnern beträgt 59.000 Euro.

INHALT:

Co-funded by the
Erasmus+ Programme
of the European Union

PROJEKT-MODULE und KITS von ARDUinVET

Co-funded by the Erasmus+ Programme of the European Union

Erasmus+ KA-202

Strategisches Partnerschaftsprojekt für berufliche Bildung und Ausbildung

Projekttitel: "Lehren und Lernen mit Arduinos in der Berufsbildung"

Projekt-Akronym: "ARDUinVET"

Projekt Nr.: “2020-1-TR01-KA202-093762”

EINFÜHRUNG in ARDUINOS

(GRUNDLAGEN von ARDUINOS)

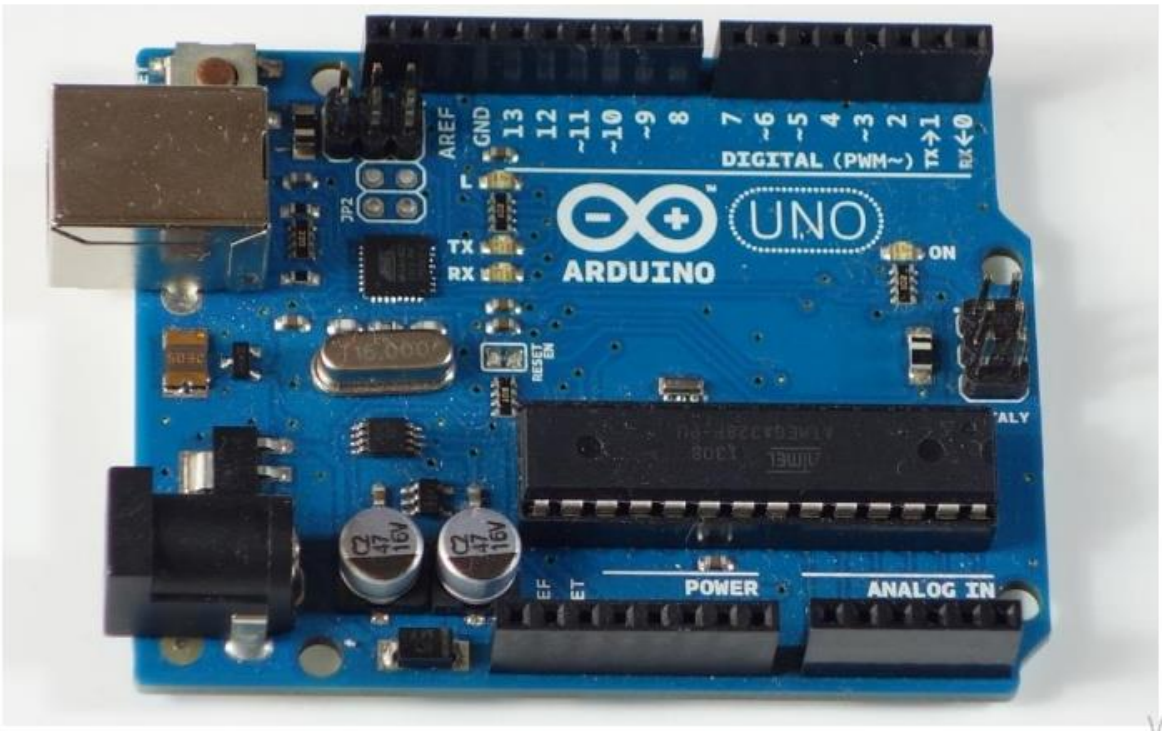
EINFÜHRUNG in den ARDUINO

Arduino ist eine Prototyp-Plattform (Open-Source), die auf einer einfach zu bedienenden Hardware und Software basiert. Sie besteht aus einer Platine, die programmiert werden kann (Mikrocontroller genannt), und einer vorgefertigten Software namens Arduino IDE (Integrated Development Environment), die zum Schreiben und Hochladen des Computercodes auf die Platine verwendet wird.

Mit anderen Worten: Der Arduino ist ein kleiner Computer, den Sie so programmieren können, dass er Informationen aus Ihrer Umgebung liest und Befehle an die Außenwelt sendet. All das ist möglich, weil man verschiedene Geräte und Komponenten an den Arduino anschließen kann, um zu tun, was man will. Man kann damit erstaunliche Projekte durchführen, es gibt keine Grenzen für das, was man damit realisieren kann – mit Fantasie ist alles möglich!

Vereinfacht ausgedrückt ist der Arduino ein winziges Computersystem, das mit Ihren Anweisungen programmiert werden kann, um mit verschiedenen Formen der Eingabe und Ausgabe zu interagieren. Das aktuelle Arduino-Board-Modell, der Uno, ist im Vergleich zur durchschnittlichen menschlichen Hand ziemlich klein.

Der Arduino ist das in der Abbildung unten dargestellte Board.

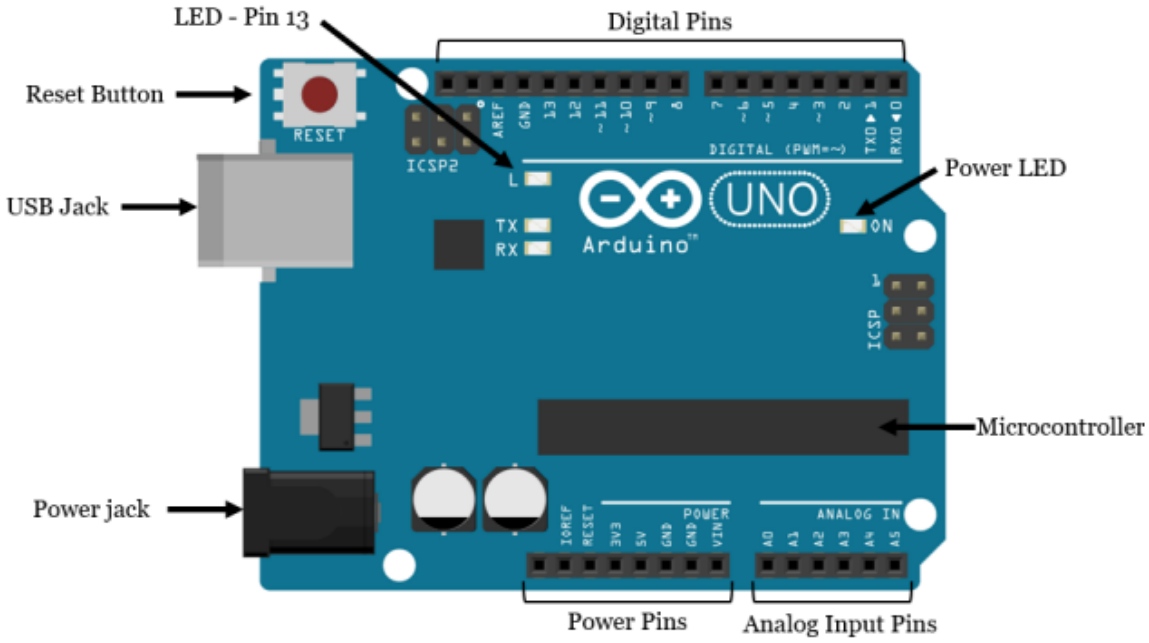


Im Grunde handelt es sich um eine kleine Entwicklungsplatine mit einem Gehirn (auch Mikrocontroller genannt), welche man an elektrische Schaltungen anschließen kann. Das macht es einfach, Eingänge zu lesen, Daten von außen zu lesen, Ausgänge zu steuern oder einen Befehl

nach außen zu senden. Das Gehirn dieser Platine (Arduino Uno) ist ein ATmega328p-Chip, auf dem du deine Programme speichern kannst, die deinem Arduino sagen, was er tun soll.

Das Arduino Uno Board erkunden

In der Abbildung unten sehen Sie ein beschriftetes Arduino-Board. Schauen wir uns an, was jedes Teil macht.



- **Mikrocontroller:** Der ATmega328p ist das Gehirn des Arduino. Alles auf dem Arduino-Board ist dazu gedacht, diesen Mikrocontroller zu unterstützen. Hier speichern Sie Ihre Programme, um dem Arduino zu sagen, was er tun soll.

- **Digitale Pins:** Arduino hat 14 digitale Pins, die von 0 bis 13 beschriftet sind und als Eingänge oder Ausgänge fungieren können. o Wenn sie als Eingänge eingestellt sind, können diese Pins Spannung lesen. Sie können nur zwei Zustände lesen: HIGH oder LOW. o Wenn sie als Ausgänge eingestellt sind, können diese Pins Spannung anlegen. Sie können nur 5 V (HIGH) oder 0 V (LOW) anlegen.

- **PWM-Stifte:** Dies sind digitale Stifte, die mit einem ~ gekennzeichnet sind (Stifte 11, 10, 9, 6, 5 und 3). PWM steht für "Pulsweitenmodulation" und ermöglicht es den digitalen Pins, unterschiedliche Spannungswerte "vorzutäuschen". Sie werden später mehr über PWM erfahren.

- **TX- und RX-Pins:** digitale Pins 0 und 1. Das T steht für "Transmit" und das R für "Receive". Der Arduino verwendet diese Pins, um mit anderen elektronischen Geräten über die serielle Schnittstelle zu kommunizieren. Der Arduino verwendet diese Pins auch für die Kommunikation mit Ihrem Computer, wenn Sie neuen Code hochladen. Vermeiden Sie die Verwendung dieser Pins für andere Aufgaben als die serielle Kommunikation, es sei denn, Ihnen gehen die Pins aus.

- **LED am digitalen Pin 13:** Dies ist nützlich für ein einfaches Debugging der Arduino-Skizzen.

- TX- und RX-LEDs: Diese Leds blinken, wenn Informationen zwischen dem Computer und dem Arduino gesendet werden.

- Analoge Pins: Die analogen Pins sind mit A0 bis A5 beschriftet und werden häufig zum Auslesen analoger Sensoren verwendet. Sie können verschiedene Spannungswerte zwischen 0 und 5 V auslesen. Außerdem können sie wie die Digitalpins auch als digitale Ausgangs-/Eingangsstifte verwendet werden.

- Stromversorgungspins: Der Arduino liefert 3,3 V oder 5 V über diese Pins. Dies ist sehr nützlich, da die meisten Komponenten 3,3 V oder 5 V für den Betrieb benötigen. Die mit "GND" bezeichneten Pins sind die Erdungspins.

- Reset-Knopf: Wenn Sie diesen Knopf drücken, wird das Programm, das gerade in Ihrem Arduino ausgeführt wird, neu gestartet. Neben den Stromversorgungspins gibt es auch einen Reset-Pin, der als Reset-Knopf fungiert. Wenn Sie eine kleine Spannung an diesen Pin anlegen, wird der Arduino zurückgesetzt.

- Power ON LED: leuchtet, wenn der Arduino mit Strom versorgt wird.

- USB-Buchse: Sie benötigen ein USB-A-Stecker-auf-USB-B-Stecker-Kabel (siehe Abbildung unten), um Programme von Ihrem Computer auf Ihr Arduino-Board zu übertragen. Über dieses Kabel wird Ihr Arduino auch mit Strom versorgt.

- Stromanschluss: Sie können den Arduino über den Stromanschluss mit Strom versorgen. Die empfohlene Eingangsspannung beträgt 7V bis 12V. Es gibt verschiedene Möglichkeiten, den Arduino mit Strom zu versorgen: z.B. wiederaufladbare Batterien, Einwegbatterien, Wandwarzen und Solarpanel.

Arduino-Merkmale

Arduino Uno; hat einen Atmel Atmega 328P Mikrocontroller und verfügt über einen USB-Anschluss, einen Stromanschluss und einen Reset-Button. Arduino hat alles, was ein Mikrocontroller haben sollte.

Microcontroller	Atmega328P
Betriebsspannung	5V
Eingangsspannung (empfohlen)	7-12V

Eingangsspannung (Grenzwert)	6-20V
Digitale Eingangs-/Ausgangspin	14
PWM-Eingangs-/Ausgangspin	6
Analoger Eingangspin	6
Gleichstrom pro Eingangs-/Ausgangspin	20mA
DC-Strom für 3,3V	50mA
Flash memory	32 KB
Sram	2KB
EEPROM	1 KB
Taktgeschwindigkeit	16 MHz
Länge	68.6 mm
Breite	53.4 mm
Gewicht	25 g
Abbildung: Merkmale des Arduino Uno	

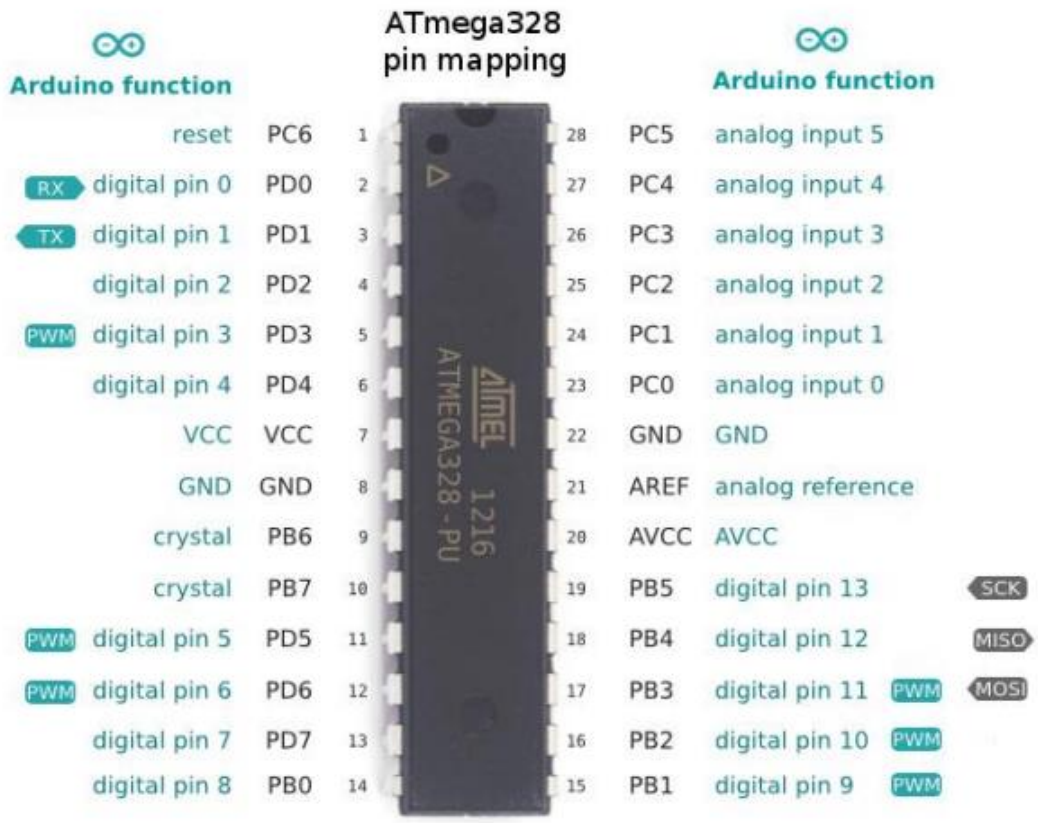


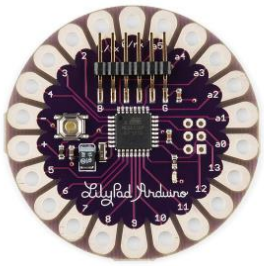
Abbildung: Atmega 328P Pins

ANDERE ARDUINO-TYPEN

MEGA ARDUINO

Das Mega Arduino ist mit dem Atmega 2560 Mikrocontroller ausgestattet. Es hat 54 digitale Eingangs-/Ausgangs-Pins, 16 analoge Eingänge, 4 serielle Hardware-Ports und einen 16-MHz-Quarzoszillator. Sie wird sowohl über USB als auch über einen Gleichstromadapter mit Strom versorgt. Im Allgemeinen wird die Karte, die die gleichen Funktionen wie der Arduno UNO hat, bei größeren Projekten bevorzugt, da sie mehr Pins hat.	

ARDUINO LILYPAD

Lilypad ist zum Nähen auf Kleidern und Stoffen gedacht. Auf diese Weise kann es in unterschiedlichsten Projekten verwendet werden. Es ist mit einem Atmega 168V Mikrocontroller ausgestattet.	

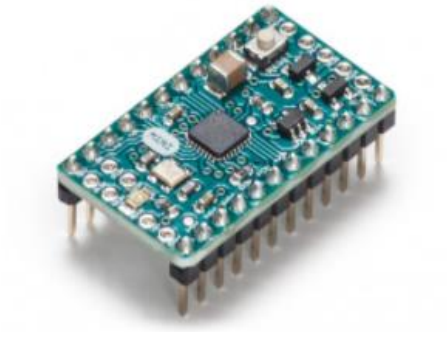
ARDUINO ETHERNET

Arduino Ethernet verfügt über einen Ethernet-Chip und einen Ethernet-Anschluss für Projekte mit Internetanschluss. Es gibt auch einen SD-Card-Slot auf der Karte, die das Atmega 328-Modell als Mikrocontroller hat.	

ARDUINO BLUETOOTH

Es gibt ein Bluetooth-Modul auf Arduino BT, ideal für die Erstellung von Anwendungen, die mit dem Bluetooth-Protokoll kommunizieren. Dieses Modul kann auch verwendet werden, um Arduino über Bluetooth zu programmieren.	

ARDUINO MİNİ

Hierbei handelt es sich um ein Arduino-Modell, das auf einem Breadboard betrieben oder in ein anderes Design integriert werden kann. Es ist mit einem Mikrocontroller des Modells Atmega 168 oder Atmega 328 ausgestattet. Es ist ideal für Anwendungen, bei denen eine geringe Größe besonders wichtig ist.	

ARDUINO NANO

Es handelt sich um ein sehr kleines Modell, das für Anwendungen auf der Leiterplatte geeignet ist. Es verfügt über einen Atmega 328- oder Atmega 168-Mikrocontroller, einen Spannungsregler, einen Seriell-USB-Wandlerchip, einen Gleichspannungseingang und einen Mini-USB-Anschluss.	

ARDUINO LEONARDO

Es ist eines der Arduino-Boards, das einen Atmega 32u4 Mikrocontroller auf dem Arduino Leonardo enthält und keinen zusätzlichen Chip für den USB-Anschluss benötigt. Mit 20 digitalen Eingängen / Ausgängen und 12 analogen Eingängen, hat der Mikrocontroller auf dem Board eine Oberflächenmontage Abdeckung. Dank seiner USB-Anschlussmöglichkeiten kann der Leonardo als Maus oder Tastatur an den Computer angeschlossen werden.	

ARDUINO ESPLORA

Esplora ist ein Arduino-Board, das im Gegensatz zu den anderen verschiedene Sensoren enthält. Dank der Sensoren auf der Karte ist es möglich, viele Anwendungen auszuführen, ohne dass weitere Zusätze und übermäßige elektronische Kenntnisse erforderlich sind. Esplora ist mit einem Schiebepotentiometer, einem Licht- und Tonsensor, einem Temperatursensor, einem Tongenerator, einem 2-Achsen-Mini-Analog-Joystick, einer 3-Farben-LED und einem Beschleunigungsmesser ausgestattet. Wie Leonarda ist auch Esplora mit einem Atmega 32U4 AVR-Mikrocontroller ausgestattet. Über den Micro-USB-Anschluss können Anwendungen entwickelt werden, die als Maus oder Tastatur fungieren können.	

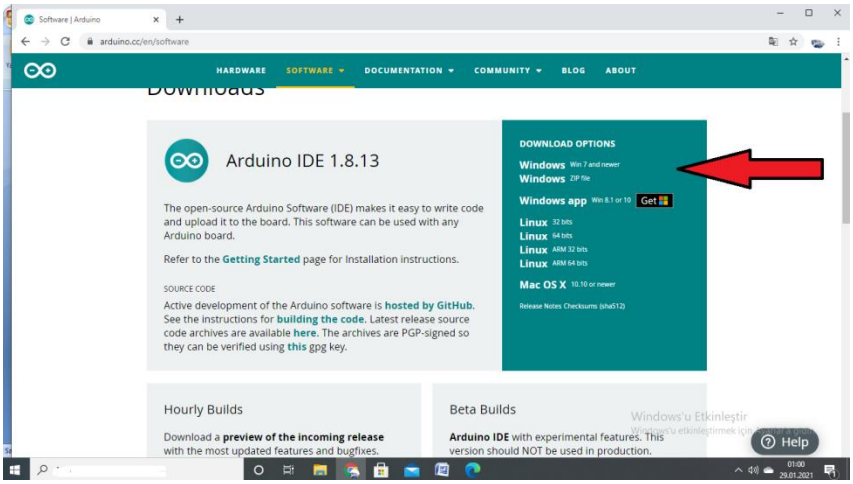
Herunterladen der Arduino IDE

In der Arduino IDE (Integrated Development Environment) entwickeln Sie Ihre Programme, die dem Arduino sagen, was er tun soll.

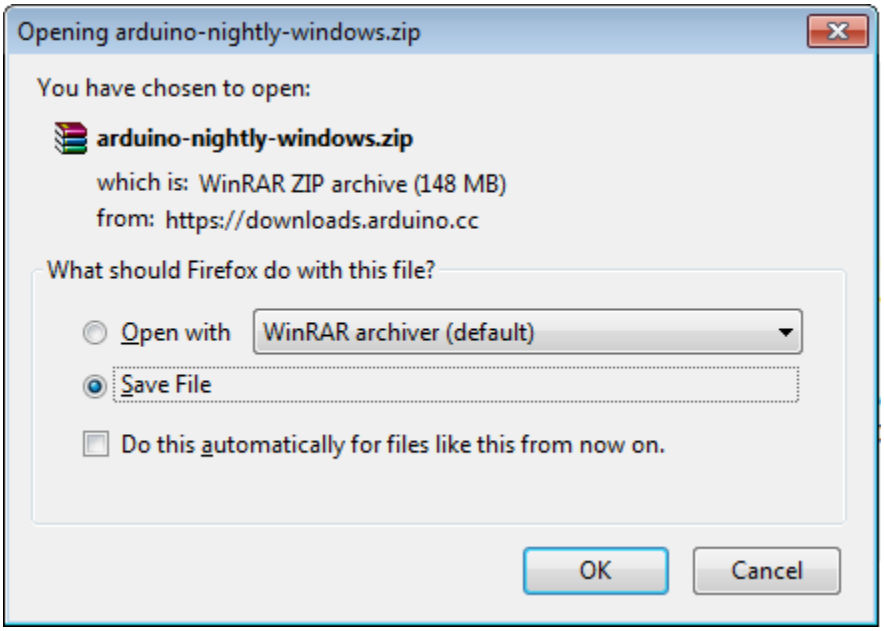
Um die Arduino IDE für Windows zu installieren, müssen wir den Anweisungen folgen.

Mit der Arduino IDE können Sie neue Programme über USB auf den Hauptchip, den ATmega328p, laden.

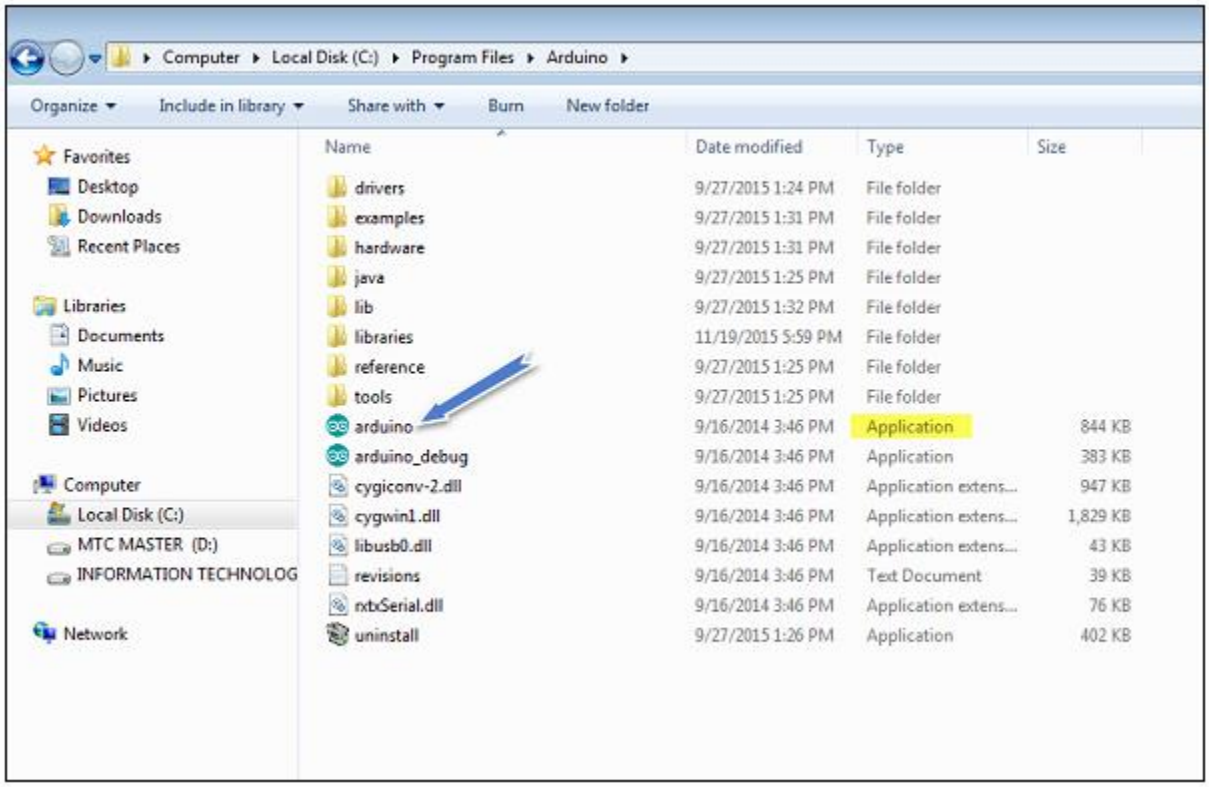
Um die Arduino IDE herunterzuladen, klicken Sie bitte auf den folgenden Link: https://www.arduino.cc/en/Main/Software.



Wählen Sie das Betriebssystem, das Sie verwenden, und laden Sie es herunter. Nachdem die Arduino IDE-Software heruntergeladen wurde, müssen wir den Ordner entpacken.



In diesem Ordner befindet sich das Anwendungssymbol mit dem “Unendlichkeits”-Symbol (application.exe). Doppelklicken Sie auf das Symbol, um die IDE zu starten. Folgen Sie dann einfach dem Installationsassistenten, um die Arduino-IDE zu installieren.

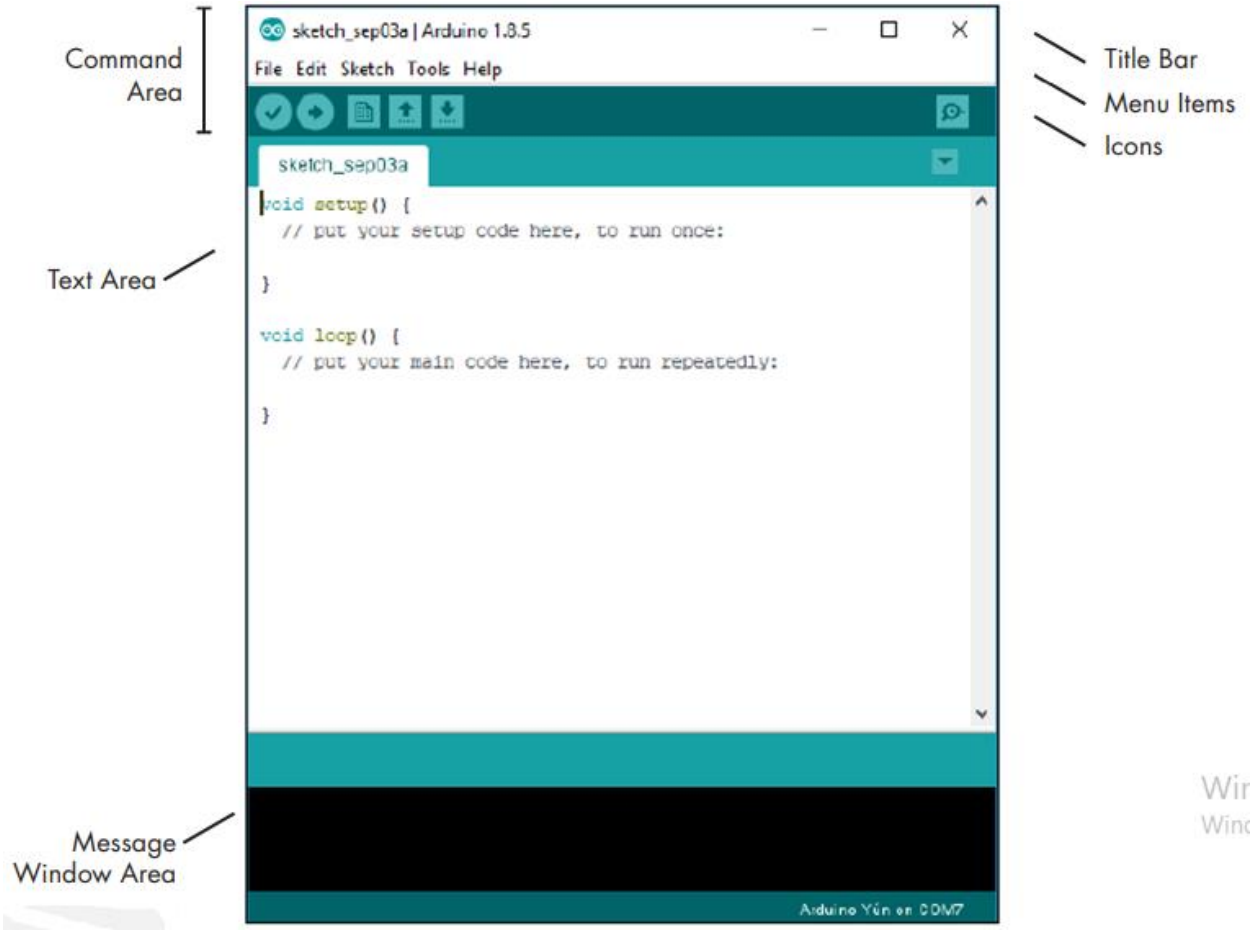


Arduino IDE Fenster zum Schreiben von Programmen

Wenn Sie die Arduino-IDE zum ersten Mal öffnen, sollten Sie etwas ähnliches sehen wie in der Abbildung unten.

Wie in der Abbildung unten gezeigt, ähnelt die Arduino IDE einer einfachen Textverarbeitung. Die IDE ist in drei Hauptbereiche unterteilt: den Befehlsbereich, den Textbereich und den

Nachrichtenfensterbereich.



Menüpunkte

Wie bei jedem Textverarbeitungsprogramm oder Texteditor können Sie auf einen der Menüpunkte klicken, um die verschiedenen Optionen anzuzeigen.

Datei: Enthält Optionen zum Speichern, Laden und Drucken von Skizzen, eine Reihe von Beispielskizzen zum Öffnen sowie das Untermenü "Einstellungen".

Bearbeiten: Enthält die üblichen Kopier-, Einfüge- und Suchfunktionen, die in jedem Textverarbeitungsprogramm enthalten sind.

Skizze: Enthält die Funktion zum Überprüfen Ihrer Skizze vor dem Hochladen auf eine Platine sowie einige Skizzenordner- und Importoptionen.

Werkzeuge: Enthält eine Vielzahl von Funktionen sowie die Befehle zur Auswahl des Arduino-Boardtyps und des USB-Anschlusses.

Hilfe: Enthält Links zu verschiedenen Themen von Interesse und die Version der IDE.

Was ist die Skizze:

Ein Arduino-Sketch ist eine Reihe von Anweisungen, die Sie erstellen, um eine bestimmte Aufgabe zu erfüllen; mit anderen Worten, ein Sketch ist ein Programm.

Der Sketch ist nichts anderes als eine Reihe von Anweisungen, die der Arduino ausführen soll. Skizzen, die mit der Arduino-IDE erstellt werden, werden als .pde-Dateien gespeichert. Um

einen Sketch zu erstellen, müssen Sie die drei wichtigsten Teile erstellen: Variablendeklaration, die Setup-Funktion und die Hauptschleifenfunktion.

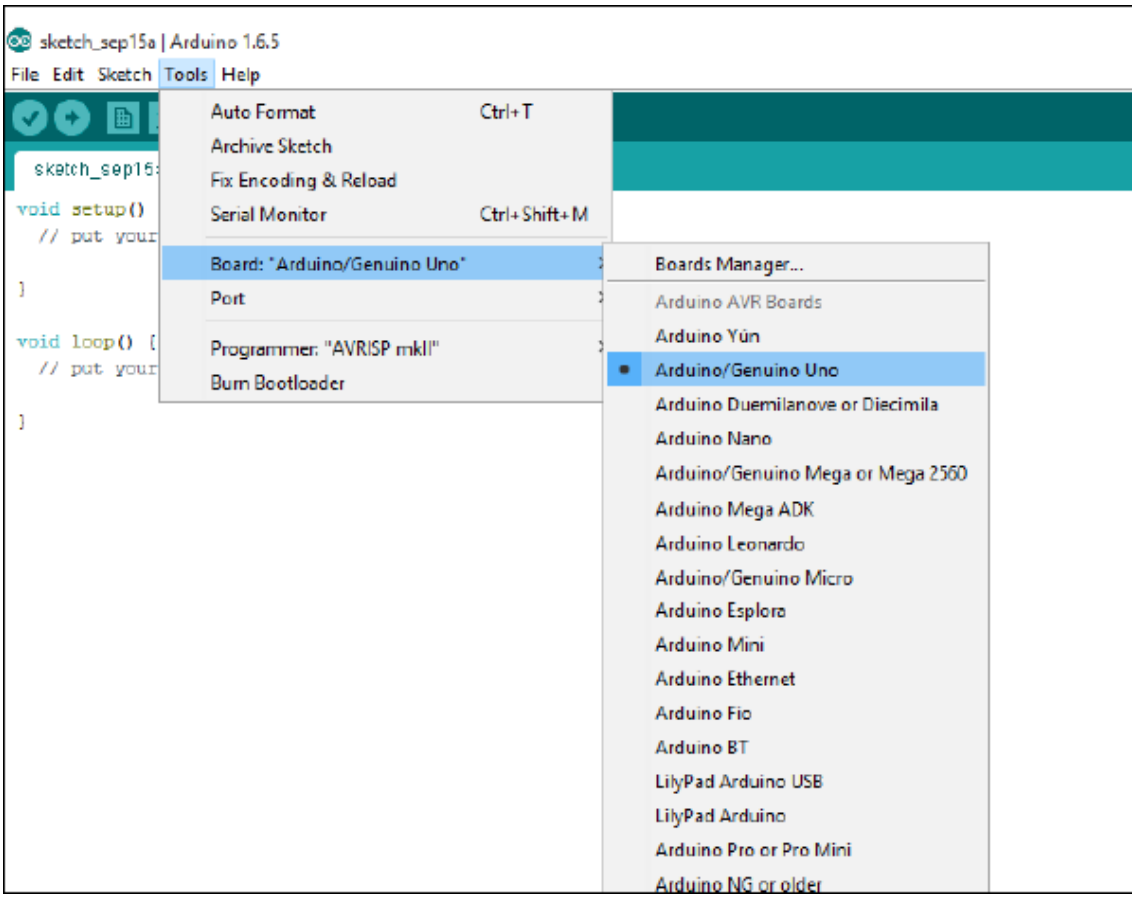
Arduino IDE Toolbar Buttons

Unterhalb der Menü-Symbolleiste befinden sich sechs Symbole. Fahren Sie mit der Maus über jedes Symbol, um seinen Namen anzuzeigen. Die Symbole lauten von links nach rechts wie folgt:

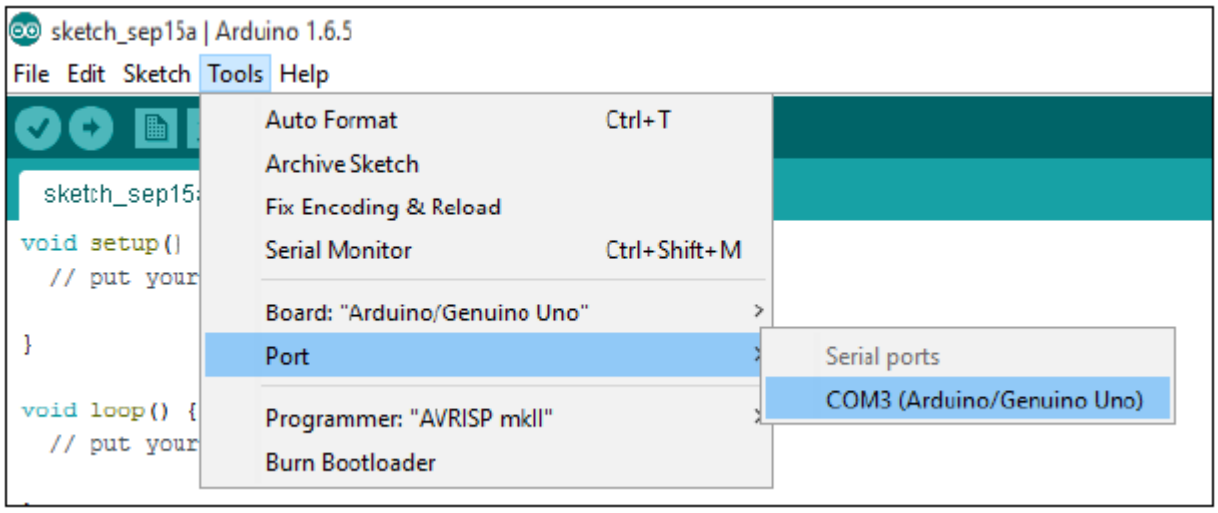
	**Überprüfen (Kompilieren):** Klicken Sie hier, um zu prüfen, ob der Arduino-Sketch gültig ist und keine Programmierfehler enthält.
	**Neu:** Klicken Sie hierauf, um eine neue leere Skizze in einem neuen Fenster zu öffnen.
	**Öffnen:** Klicken Sie hier, um eine gespeicherte Skizze zu öffnen.
	**Speichern:** Klicken Sie auf diese Schaltfläche, um die geöffnete Skizze zu speichern. Wenn die Skizze noch keinen Namen hat, werden Sie aufgefordert, einen zu erstellen.
	**Hochladen:** Klicken Sie auf diese Schaltfläche, um die Skizze zu überprüfen und anschließend auf das Arduino-Board hochzuladen.
	**Serieller Monitor:** Klicken Sie auf diese Schaltfläche, um ein neues Fenster zu öffnen, in dem Sie Daten zwischen Ihrem Arduino und der IDE senden und empfangen können.

Anschließen des Arduino

Verbinden Sie Ihren Arduino UNO über USB mit Ihrem Computer.
Nachdem Sie Ihren Arduino mit einem USB-Kabel angeschlossen haben, müssen Sie sicherstellen, dass die Arduino IDE das richtige Board ausgewählt hat.
In unserem Fall verwenden wir den Arduino Uno, also sollten wir zu **Werkzeuge Board: Arduino/Genuino Uno.**



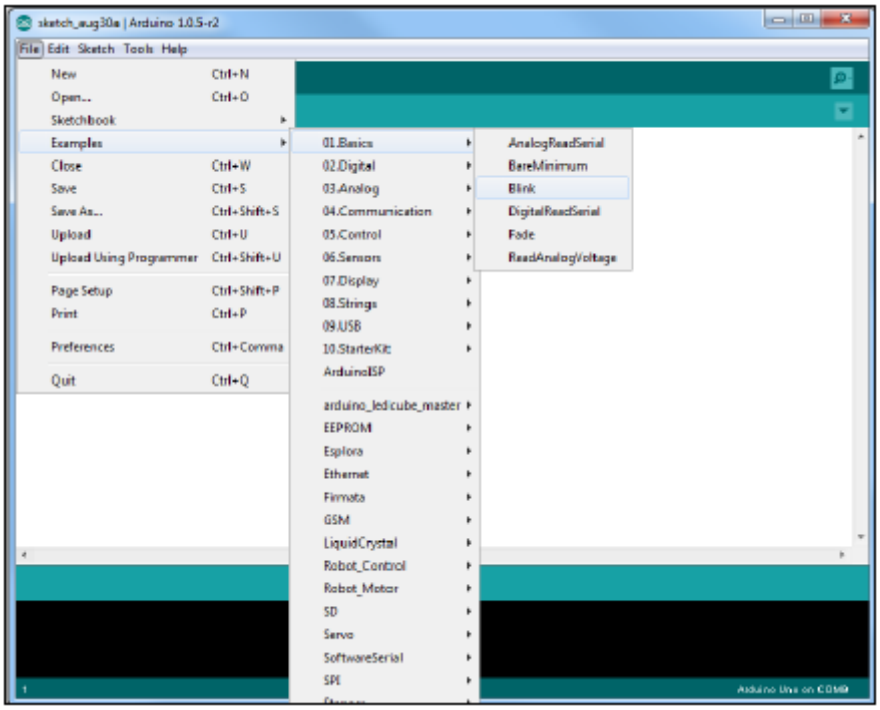
Then, you should select the serial port where your Arduino is connected to. Go to W**erkzeuge ▸ Port** und wählen Sie den richtigen Anschluss.



Hochladen einer Arduino-Skizze

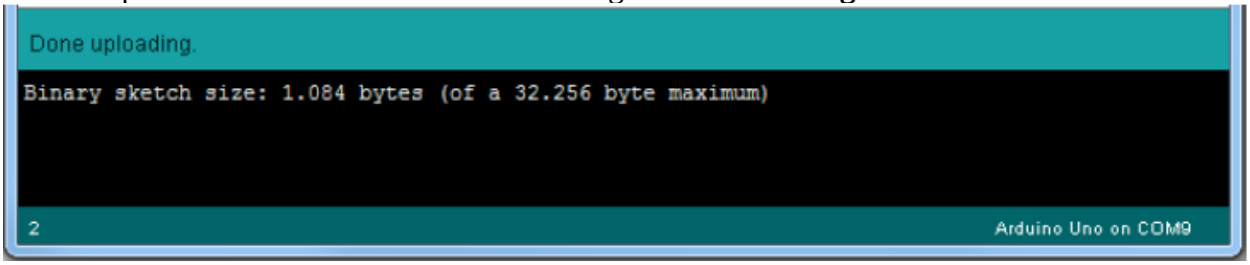
Um Ihnen zu zeigen, wie Sie Ihren Code auf Ihr Arduino-Board hochladen können, zeigen wir Ihnen ein einfaches Beispiel. Dies ist eines der einfachsten Beispiele - es besteht aus dem Blinken der On-Board-LED oder dem digitalen Pin 13 jede Sekunde.

1. Öffnen Sie Ihre Arduino IDE.
2. Gehe zu **Datei ▸ Beispiele ▸ 01.Basics ▸ Blink**



Standardmäßig ist die Arduino IDE für den Arduino UNO vorkonfiguriert. Klicken Sie auf die Schaltfläche **Hochladen** und warten Sie ein paar Sekunden.

Nach ein paar Sekunden sollten Sie die Meldung **"Hochladen abgeschlossen"** sehen.



Dieser Code blinkt einfach die integrierte LED auf Ihrem Arduino UNO (rot hervorgehoben). Sie sollten sehen, wie die kleine LED eine Sekunde lang leuchtet und dann wiederholt für eine weitere Sekunde ausgeschaltet wird.

Einen Ausgang steuern und einen Eingang lesen

Ein Arduino-Board enthält digitale Pins, analoge Pins und PWM-Pins.

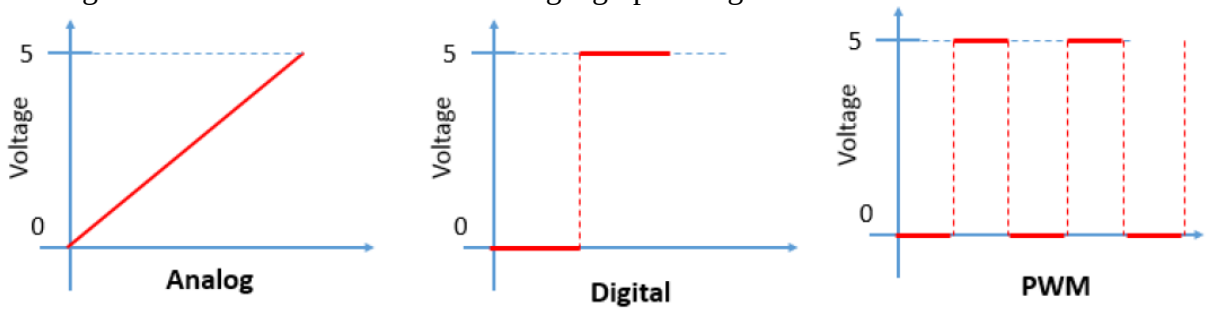
Unterschied zwischen digital, analog und PWM

Bei **digitalen Pins** gibt es nur zwei mögliche Zustände, nämlich ein oder aus. Diese können auch als High oder Low, 1 oder 0 und 5V oder 0V bezeichnet werden.

Wenn zum Beispiel eine LED eingeschaltet ist, dann ist ihr Zustand High oder 1 oder 5V. Wenn sie ausgeschaltet ist, ist ihr Zustand Low, 0 oder 0V.

Bei **analogen Pins** gibt es eine unbegrenzte Anzahl möglicher Zustände zwischen 0 und 1023. So können Sie Sensorwerte ablesen. Bei einem Lichtsensor z. B. wird bei sehr dunkler Beleuchtung 1023 angezeigt, bei sehr heller Beleuchtung 0. Bei einer Helligkeit zwischen dunkel und sehr hell wird ein Wert zwischen 0 und 1023 angezeigt.

PWM-Pins sind digitale Pins, d. h. sie geben entweder 0 oder 5 V aus. Allerdings können diese Pins "falsche" Zwischenwerte zwischen 0 und 5V ausgeben, da sie eine "Pulsweitenmodulation" (PWM) durchführen können. PWM ermöglicht die "Simulation" unterschiedlicher Leistungsniveaus durch Oszillation der Ausgangsspannung des Arduino.



Steuerung eines Ausgangs

Um einen digitalen Ausgang zu steuern, verwenden Sie die Funktion digitalWrite() und schreiben zwischen Klammern den Pin, den Sie steuern möchten, und dann HIGH oder LOW.

Um einen PWM-Pin zu steuern, verwenden Sie die Funktion analogWrite() und schreiben zwischen die Klammern den Pin, den Sie steuern möchten, und eine Zahl zwischen 0 und 255.

Lesen einer Eingabe

Um einen analogen Eingang zu lesen, verwenden Sie die Funktion analogRead() und für einen digitalen Eingang verwenden Sie digitalRead().

Hinweis: Der beste Weg, Arduino zu lernen, ist zu üben. Machen Sie also viele Projekte und fangen Sie an, etwas zu bauen.

Co-funded by the Erasmus+ Programme of the European Union

Erasmus+ KA-202

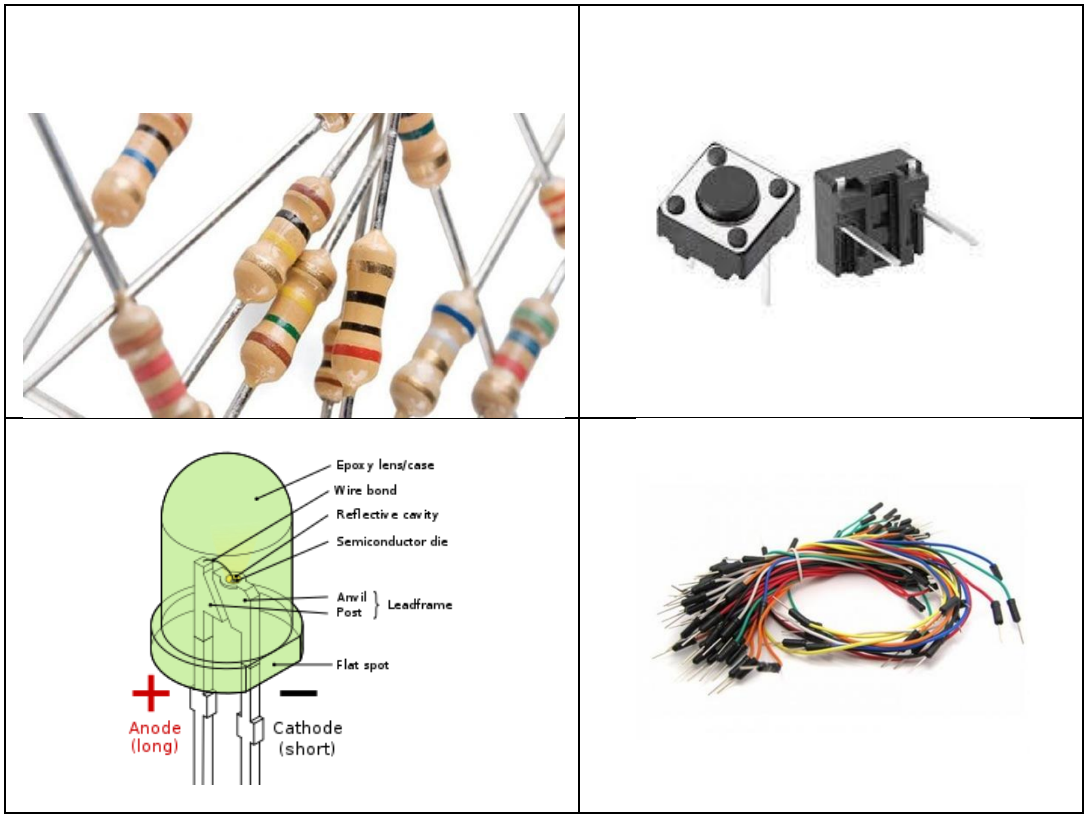
Strategisches Partnerschaftsprojekt für die berufliche Bildung und Ausbildung

Projekttitel: "Lehren und Lernen mit Arduinos in der Berufsbildung"

Projekt-Akronym: "ARDUinVET"

Projekt Nr.: "2020-1-TR01-KA202-093762"

Arduino Eingangs-/Ausgangsmodul und Kit-Modul



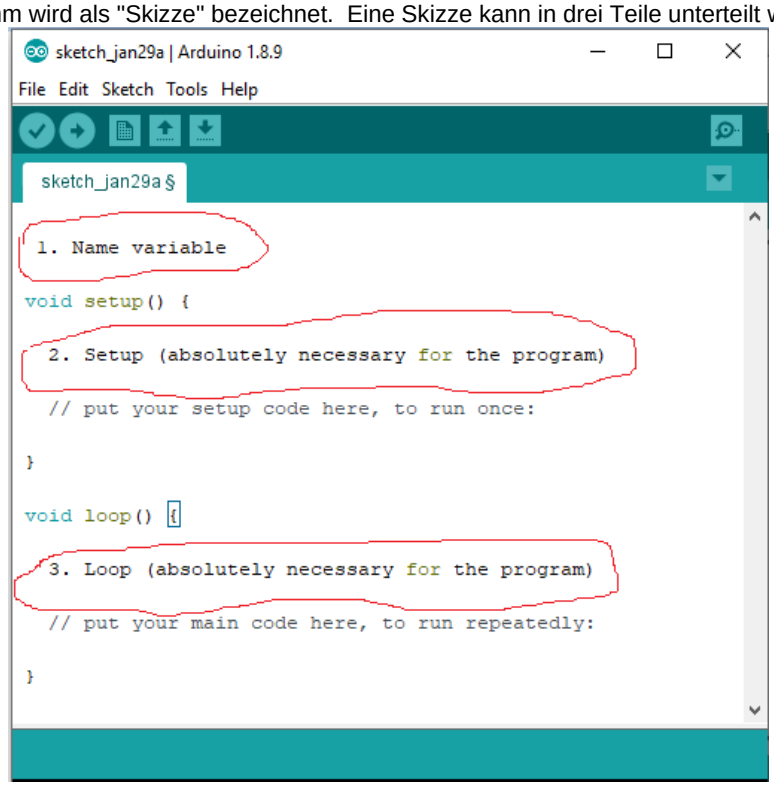
Planung unserer Projekte

Wenn Sie mit Ihren ersten Projekten beginnen, versuchen Sie eventuell, Ihre Skizze sofort zu anzufertigen, nachdem Ihnen eine neue Idee gekommen ist. Doch bevor Sie damit beginnen, sind ein paar grundlegende Vorbereitungsschritte angebracht. Schließlich ist Ihr Arduino-Board kein

Gedankenleser; es braucht präzise Anweisungen, und selbst wenn diese Anweisungen vom Arduino ausgeführt werden können, sind die Ergebnisse möglicherweise nicht so, wie Sie es erwartet haben, wenn Sie auch nur ein kleines Detail übersehen haben.
Egal, ob Sie ein Projekt erstellen, das einfach nur ein Licht blinkt, oder ein automatisiertes Modellbahnsignal, ein detaillierter Plan ist die Grundlage für den Erfolg. Wenn Sie Ihre Arduino-Projekte entwerfen, sollten Sie die folgenden grundlegenden Schritte befolgen:
1. Definieren Sie Ihr Ziel. Legen Sie fest, was Sie erreichen wollen.
2. Schreiben Sie Ihren Algorithmus. Ein Algorithmus ist eine Reihe von Anweisungen, die beschreiben, wie Ihr Projekt durchgeführt werden soll. Ihr Algorithmus listet die Schritte auf, die notwendig sind, um das Ziel Ihres Projekts zu erreichen.
3. Wählen Sie Ihre Hardware aus. Lege fest, wie sie mit dem Arduino verbunden werden soll.
4. Schreiben Sie Ihren Sketch. Erstelle dein erstes Programm, das dem Arduino sagt, was er tun soll.
5. Verdrahten Sie es. Verbinde deine Hardware, Schaltkreise und andere Dinge mit dem Arduino-Board.
6. Testen und Fehlersuche. Funktioniert es? In dieser Phase identifizieren Sie Fehler und finden deren Ursachen, sei es im Sketch, in der Hardware oder im Algorithmus.
Je mehr Zeit Sie für die Planung Ihres Projekts aufwenden, desto einfacher wird die Test- und Fehlerbehebungsphase sein.

Grundstruktur einer Skizze

Das Arduino-Programm wird als "Skizze" bezeichnet. Eine Skizze kann in drei Teile unterteilt werden.



1. Variable benennen:
Im ersten Teil werden die Elemente des Programms benannt. Dieser Teil ist nicht unbedingt notwendig.

2. Setup (für das Programm unbedingt erforderlich):

Das Setup wird nur einmal durchgeführt. Hier teilen Sie dem Programm zum Beispiel mit, welcher Pin (Steckplatz für Kabel) ein Eingang und was ein Ausgang auf den Karten sein soll. Definiert als Ausgang: Der Pin soll eine Spannung ausgeben. Zum Beispiel: An diesem Pin soll eine LED leuchten.
Definiert als Eingang: Die Karte soll eine Spannung ausgeben. Ein Beispiel: Ein Schalter wird betätigt. Die Karte erkennt dies, weil sie am Input-Pin eine Spannung erhält.

3. Schleife (unbedingt notwendig für das Programm):
Dieser Schleifenteil wird von der Platine ständig wiederholt. Es nimmt den Sketch von Anfang bis Ende auf Skizze von Anfang bis Ende und fängt wieder von vorne an und so weiter.

Weitere Syntaxregeln

Diese Regeln müssen beim Schreiben von Arduino-Programmen unbedingt beachtet werden. Andernfalls wird Ihr Programm fehlschlagen.**;**

(Semikolon):

; (Semikolon) wird verwendet, um eine Anweisung zu beenden. Wenn Sie vergessen, eine Zeile mit einem Semikolon zu beenden, führt dies zu einem Compilerfehler.

Beispiel: int a=13;

{} (geschwingene Klammern):

Geschwungene Klammern sind ein wichtiger Bestandteil der Arduino-Programmiersprache. Sie werden in mehreren verschiedenen Konstrukten verwendet, was für Anfänger manchmal verwirrend sein kann. Auf eine öffnende geschwungene Klammer "{" muss immer eine schließende geschweifte Klammer "}" folgen.

Die wichtigsten Verwendungen von geschwungene Klammern: Funktionen, Schleifen, bedingte Anweisungen

Beispiel:

```
void myfunction(datentyp argument) { statements(s) }
```

// (einzeiliger Kommentar) und /* */ (mehrzeiliger Kommentar):

Dies sind Zeilen im Programm, die dazu dienen, sich selbst oder andere über die Funktionsweise des Programms zu informieren. Sie werden vom Compiler ignoriert und nicht an den Prozessor exportiert, so dass sie keinen Platz auf dem Atmega-Chip benötigen.
Kommentare dienen nur dazu, Ihnen zu helfen, zu verstehen (oder sich daran zu erinnern), wie Ihr Programm funktioniert oder

um andere darüber zu informieren, wie Ihr Programm funktioniert. Es gibt zwei verschiedene Möglichkeiten, eine Zeile als Kommentar zu kennzeichnen:

Beispiel:

x = 5; // Dies ist ein einzeiliger Kommentar. Alles nach den Schrägstrichen ist ein Kommentar // bis zum Ende der Zeile

x = 5; /*Dies ist ein mehrzeiliger Kommentar. ..... Dies ist das Ende eines mehrzeiligen Kommentars. */

#define:

#define ermöglicht es dem Programmierer, einem konstanten Wert einen Namen zu geben, bevor das Programm kompiliert wird. Definierte Konstanten in Arduino nehmen keinen Programmspeicherplatz auf dem Chip ein. Im Allgemeinen wird das Schlüsselwort const für die Definition von Konstanten bevorzugt und sollte anstelle von #define verwendet werden.

Beispiel:

#define ledPin 3 // Der Compiler ersetzt zur Kompilierzeit jede Erwähnung von ledPin durch den Wert 3.

#include:

#include wird verwendet, um externe Bibliotheken in Ihren Sketch einzubinden. Dies gibt dem Programmierer Zugang zu einer großen Gruppe von Standard-C-Bibliotheken (Gruppen von vorgefertigten Funktionen) und auch Bibliotheken, die speziell für Arduino geschrieben wurden.

Beispiel:

#include <servo.h>

Arduino - Datentypen

Datentypen werden zur Deklaration von Variablen oder Funktionen unterschiedlichen Typs verwendet. Der Typ einer Variablen bestimmt, wie viel Platz sie im Speicher einnimmt und wie das gespeicherte Bitmuster interpretiert wird.

Ausdrücke, die zur Speicherung beliebiger Informationen im Speicher verwendet werden und deren Wert sich während des Programmablaufs ändern kann, werden als Variablen bezeichnet. Variablen können Zahlen, Zeichen oder logische Ausdrücke sein. Je nach Variablentyp sollte der passende Datentyp gewählt werden. Je nach Datentyp wird ein bestimmter Bereich zugewiesen, in dem die Variable im Speicher definiert wird.

Die folgende Tabelle enthält alle Datentypen, die Sie bei der Arduino-Programmierung verwenden werden.

Type	Enthält
boolean	kann entweder true oder false enthalten
char	-128 to 127
byte	0 to 255
unsigned char	0 to 255
int	-32,768 to 32,767
unsigned int	0 to 65,535
word	(wie unsigned int)
long (or long int)	-2,147,483,648 to 2,147,483,647
unsigned long	0 to 4,294,967,295
float	-3.4028235E+38 to 3.4028235E+38
double	(wie bei Float)

Arduino - Variablen und Konstanten

Bei der Definition der Variablen müssen der Name der Variablen, der Wert der Variablen und der entsprechende Datentyp für die Variable festgelegt werden.

Die Definition erfolgt wie im nachstehenden Beispiel dargestellt:

int LED =12;

Hier: **int**=Datentyp LED=Variablenname 12=Variablenwert

Variablen, die Arduino verwendet, haben eine Eigenschaft, die Scope genannt wird. Ein Scope ist ein Bereich des Programms und es gibt drei Stellen, an denen Variablen deklariert werden können. Diese sind:

- Innerhalb einer Funktion oder eines Blocks, was als lokale Variablen bezeichnet wird.

- In der Definition von Funktionsparametern, die als formale Parameter bezeichnet werden.

- Außerhalb aller Funktionen, was als globale Variablen bezeichnet wird.

Eine Konstante ist ein Variablenqualifizierer, der das Verhalten der Variablen verändert und sie "schreibgeschützt" macht. Das bedeutet, dass die Variable wie jede andere Variable ihres Typs verwendet werden kann, ihr Wert aber nicht geändert werden kann. Sie erhalten einen Compiler-Fehler, wenn Sie versuchen, einer const-Variablen einen Wert zuzuweisen. Konstanten, die mit dem Schlüsselwort const definiert werden, gehorchen den Regeln des Variablen-Scopings, die

auch für andere Variablen gelten. Dies und die Fallstricke bei der Verwendung von #define machen das Schlüsselwort const zu einer überlegenen Methode für die Definition von Konstanten und ist der Verwendung von #define vorzuziehen.

Beispiel:

const float pi = 3.14;

Notes:

#define **or** const: Sie können entweder const oder #define verwenden, um numerische oder String-Konstanten zu erstellen. Für Arrays müssen Sie const verwenden. Im Allgemeinen ist const gegenüber #define für die Definition von Konstanten vorzuziehen.

Arduino – Bediener

Ein Operator ist ein Symbol, das den Compiler anweist, bestimmte mathematische oder logische Funktionen auszuführen. In Arduinos können die folgenden Arten von Operatoren verwendet werden.

Arithmetische Operatoren:

Angenommen, die Variable A hat den Wert 10 und die Variable B den Wert 20, dann -

Name des Bedieners	Bediener einfach	Beispiel
Zuweisungsoperator	=	A = B
Addition	+	A + B ergibt 30
Subtraktion	-	A - B ergibt -10
Multiplikation	*	A * B ergibt 200
Division	/	B / A ergibt 2

Modulo	%	B % A ergibt 0

Vergleichsoperatoren:

Angenommen, die Variable A hat den Wert 10 und die Variable B den Wert 20, dann -

Operator Name	Operator Symol	Beispiel
gleich	==	(A == B) ist wahr
Nicht gleich	!=	(A != B) ist nicht wahr
Weniger als	<	(A < B) ist wahr
Größer als	>	(A > B) ist nicht wahr
kleiner als oder gleich	<=	(A <= B) ist wahr
größer als oder gleich	>=	(A >= B) ist nicht wahr

Boolesche Operatoren

Angenommen, die Variable A hat den Wert 10 und die Variable B den Wert 20, dann -

Operator Name	Operator einfach	Beispiel
and	&&	(A && B) ist wahr
or	||	(A || B) ist wahr
not	!	!(A && B) ist falsch

Bitweise Operatoren

Angenommen, die Variable A enthält 60 und die Variable B enthält 13, dann -

Operator Name	Operator einfach	Beispiel
und	&	(A & B) ergibt 12 das ist 0000 1100
oder	|	(A | B) ergibt 61 das ist 0011 1101
exklusiv oder	^	(A ^ B) ergibt 49 das ist 0011 0001
nicht	~	(~A) ergibt -60 das ist 1100 0011
nach links schieben	<<	A << 2 ergibt 240 das ist 1111 0000
nach rechts schieben	>>	A >> 2 ergibt 15 das ist 0000 1111

Arduino - E/A-Funktionen (Befehle)

Funktionen ermöglichen die Strukturierung von Programmen zur Ausführung einzelner Aufgaben. Der typische Fall für die Erstellung einer Funktion ist, wenn man dieselbe Aktion mehrmals in einem Programm ausführen muss. Sie werden deutlicher, wenn wir unten konkrete Programmbeispiele in Schaltkreisen und Arduino-Programmen zeigen. Um die verschiedenen Befehlsfunktionen zu erklären, haben wir sie in einzelne Befehle unterteilt

Die Pins auf dem Arduino-Board können entweder als Eingänge oder als Ausgänge konfiguriert werden. Wir werden die Funktion der Pins in diesen Modi erklären. Es ist wichtig zu beachten, dass die meisten analogen Pins des Arduino auf die gleiche Weise konfiguriert und verwendet werden können wie die digitalen Pins.

pinMode() Funktion:

Mit der Funktion pinMode() kann ein bestimmter Pin so konfiguriert werden, dass er sich entweder wie ein Eingang oder ein Ausgang verhält. Es ist möglich, die internen Pull-Up-Widerstände mit dem Modus INPUT_PULLUP zu aktivieren.
Syntax: pinMode(pin, mode)

```
Void setup () {
 pinMode (pin , mode);
                }
```

Parameter: pin: der Arduino-Pin: Nummer, für den der Modus eingestellt werden soll.
Modus: INPUT, OUTPUT, oder INPUT_PULLUP.
Beispiele:
pinMode(13, OUTPUT); // setzt den digitalen Pin 13 als Ausgang
pinMode(5, INPUT); // stellt den digitalen Pin 5 als Eingang ein

digitalWrite() Funktion:

Die Funktion digitalWrite() wird verwendet, um einen HIGH- oder LOW-Wert an einen digitalen Pin zu schreiben. Wenn der Pin mit pinMode() als OUTPUT konfiguriert wurde, wird seine Spannung auf den entsprechenden Wert gesetzt: 5V für HIGH, 0V (Masse) für LOW. Wenn der Pin als INPUT konfiguriert ist, aktiviert (HIGH) oder deaktiviert (LOW) digitalWrite() den internen Pullup am Eingangspin. Es wird empfohlen, den pinMode() auf INPUT_PULLUP zu setzen, um den internen Pull-up-Widerstand zu aktivieren.
Wenn Sie pinMode() nicht auf OUTPUT setzen und eine LED an einen Pin anschließen, wenn Sie digitalWrite(HIGH) aufrufen, kann die LED dunkel erscheinen. Ohne explizite Einstellung von pinMode() hat digitalWrite() den internen Pull-up-Widerstand aktiviert, der wie ein großer Strombegrenzungswiderstand wirkt.
Syntax:

```
digitalWrite (pin ,value);
```

pin: die Nummer des Pins, dessen Modus Sie einstellen möchten
Wert:HIGH (1), oder LOW(0).
Beispiel:

```
digitalWrite(LED, HIGH); // Einschalten der LED
digitalWrite(LED, LOW); // LED ausschalten
```

digitalRead()Funktion:

Die Funktion digitalRead() liest den Wert von einem angegebenen digitalen Pin, entweder HIGH oder LOW.

Syntax: digitalRead(pin) pin: die Nummer des Arduino-Pins, den Sie lesen möchten.

Beispiele:

```
val = digitalRead(inPin); // Lesen des Eingangspins
```

Hinweis: Die analogen Eingangspins können als digitale Pins verwendet werden, die als A0, A1, etc. bezeichnet werden.

delay() Funktion:

Die Funktion dalay() hält das Programm für die als Parameter angegebene Zeitspanne (in Millisekunden) an. (Eine Sekunde besteht aus 1000 Millisekunden.)
Syntax: delay(ms):
ms: die Anzahl der Millisekunden, die das Programm pausieren soll. Erlaubte Datentypen: unsigned long.
Beispiele:

```
delay(1000); // wartet für 1 Sekunde
delay(2000); // wartet 2 Sekunden lang
```

analogWrite() Funktion:
Die Funktion analogWrite() schreibt einen analogen Wert (PWM-Welle) an einen Pin. Er kann verwendet werden, um eine LED mit unterschiedlicher Helligkeit zu beleuchten oder einen Motor mit verschiedenen Geschwindigkeiten anzutreiben. Nach einem Aufruf von analogWrite() erzeugt der Pin bis zum nächsten Aufruf von analogWrite() (oder einem Aufruf von digitalRead() oder digitalWrite()) an demselben Pin eine gleichmäßige Rechteckwelle mit dem angegebenen Tastverhältnis.
Beispiele:
Setzt die Ausgabe an die LED proportional zum vom Potentiometer gelesenen Wert.

```
val = analogRead(analogPin); // Lesen des Eingangspins
analogWrite(ledPin, val / 4); // analogRead-Werte gehen von 0 bis 1023, analogWrite-Werte von 0 bis 255
```

analogRead( ) Funktion

Arduino ist in der Lage zu erkennen, ob an einem seiner Pins eine Spannung anliegt und meldet dies über die Funktion digitalRead(). Es gibt einen Unterschied zwischen einem Ein/Aus-Sensor (der das Vorhandensein eines Objekts erkennt) und einem analogen Sensor, dessen Wert sich ständig ändert. Um diese Art von Sensor auszulesen, benötigen wir einen anderen Typ von Pin.

Im unteren rechten Teil des Arduino-Boards sehen Sie sechs Pins mit der Bezeichnung "Analog In". Diese speziellen Pins zeigen nicht nur an, ob eine Spannung an ihnen anliegt, sondern auch deren Wert. Mit der Funktion analogRead() können wir die an einem der Pins anliegende Spannung auslesen.

Diese Funktion gibt eine Zahl zwischen 0 und 1023 zurück, die für Spannungen zwischen 0 und 5 Volt steht. Wenn zum Beispiel eine Spannung von 2,5 V an Pin Nummer 0 anliegt, gibt analogRead(0) 512 zurück.

Syntax: analogRead(pin);

pin: die Nummer des zu lesenden analogen Eingangspins von 0 bis 5.

Example:

val = analogRead(analogPin); // Lesen des Eingangspins Serial.println(val); // Debug-Wert

if Funktion:

Die if-Anweisung prüft eine Bedingung und führt die folgende Anweisung oder eine Reihe von Anweisungen aus, wenn die Bedingung "wahr" ist.

Syntax: if (Bedingung) { //Anweisung(en) }

Zustand: ein boolescher Ausdruck (d. h. er kann wahr oder falsch sein).

Beispiele: (Die Klammern können nach einer if-Anweisung weggelassen werden. In diesem Fall wird die nächste Zeile (definiert durch das Semikolon) die einzige bedingte Anweisung.

if (x > 120) digitalWrite(LEDpin, HIGH);

wenn (x > 120)

digitalSchreiben(LEDAnschluss, HIGH);

wenn (x > 120) {digitalSchreiben(LEDpin, HIGH);}

wenn (x > 120) {

digitalWrite(LEDpin1, HIGH);

digitalSchreiben(LEDPin2, HIGH);

} // alle sind korrekt

if-else Anweisung:

Die if...else-Anweisung ermöglicht eine bessere Kontrolle über den Codefluss als die grundlegende if-Anweisung, da mehrere Tests gruppiert werden können. Eine else-Klausel (falls überhaupt vorhanden) wird ausgeführt, wenn die Bedingung in der if-Anweisung zu false führt. Die else-Klausel kann einen weiteren if-Test einleiten, so dass mehrere, sich gegenseitig ausschließende Tests gleichzeitig ausgeführt werden können.

Jeder Test geht zum nächsten über, bis ein wahrer Test gefunden wird. Wenn ein wahrer Test gefunden wird, wird der zugehörige Codeblock ausgeführt, und das Programm springt dann zu der Zeile, die auf die gesamte if/else-Konstruktion folgt. Erweist sich kein Test als wahr, wird der standardmäßige else-Block ausgeführt, sofern einer vorhanden ist, und legt das Standardverhalten fest. Es ist eine unbegrenzte Anzahl solcher else if-Verzweigungen zulässig.

Syntax:

```
if (Bedingung is TRUE) {
  // führe  Sache A aus
}
else
  //wenn NICHT,  führe Sache B aus
}
```

```
if (Bedingung1) {
  // führe Sache A aus
}
else if (Bedingung2) {
  // führe Sache B aus
}
else {
  // führe Sache C
}
```

Beispiele: (Nachstehend ein Auszug aus einem Code für ein Temperatursensorsystem.)

```
if (Temperatur >= 70) {
  // Gefahr! Schalten Sie das System ab.
}
else if (Temperatur >= 60) { // 60 <= Temperatur < 70
  // Warnung! Aufmerksamkeit des Benutzers erforderlich.
}
sonst { // Temperatur < 60
  // Sicher! Gewöhnliche Aufgaben fortsetzen.
}
```

für das Kommando:

Die for-Anweisung wird verwendet, um einen Block von Anweisungen zu wiederholen, die in geschweifte Klammern eingeschlossen sind. Ein Inkrementzähler wird in der Regel zum Erhöhen und Beenden der Schleife verwendet. Die for-Anweisung ist für alle sich wiederholenden Operationen nützlich und wird oft in Kombination mit Arrays verwendet, um mit Sammlungen von Daten/Pins zu arbeiten.

Syntax:

```
for (Initialisierung; Bedingung; Inkrement) {
  // Anweisung(en);
}
```

Parameter:

Initialisierung: geschieht zuerst und genau einmal.

Bedingung: Jedes Mal, wenn die Schleife durchlaufen wird, wird die Bedingung getestet; wenn sie wahr ist, wird die Anweisung blockiert und das Inkrement wird ausgeführt, dann wird die Bedingung erneut getestet. Wenn die Bedingung falsch wird, endet die Schleife.

Inkrement: wird bei jedem Durchlauf der Schleife ausgeführt, wenn die Bedingung wahr ist.

Beispiele:

```
for (int i = 0; i <= 255; i++) {
   analogWrite(PWMpin, i); }

for (int x = 2; x < 100; x = x * 1,5) {
        println(x); }

for (int i = 0; i > -1; i = i + x) {
  analogWrite(PWMpin, i); }
```

switch...case Befehl

Wie **if**-Anweisungen steuern switch/case den Programmablauf, indem sie es dem Programmierer ermöglichen, unterschiedlichen Code anzugeben, der unter verschiedenen Bedingungen ausgeführt werden soll. Insbesondere vergleicht eine switch-Anweisung den Wert einer Variablen mit den in case-Anweisungen angegebenen Werten. Wenn eine case-Anweisung gefunden wird, deren Wert mit dem der Variablen übereinstimmt, wird der Code in dieser case-Anweisung ausgeführt. Die break – Anweisung beendet die switch-Anweisung und wird in der Regel am Ende eines jeden Falls verwendet. Ohne eine break-Anweisung führt die switch-Anweisung die folgenden Ausdrücke weiter aus ("falling-through"), bis ein break oder das Ende der switch-Anweisung erreicht ist.

Syntax:

```
switch (var) {
  case label1:
   // Anweisungen
   abbrechen;
```

```
  case label2:
    // Anweisungen
    abbrechen;
  Standard:
    // Anweisungen
    abbrechen;
}
```

Beispiel Code:

```
switch (var) {
  Fall 1:
    //etwas tun, wenn var gleich 1 ist
    abbrechen;
Fall 2:
    //  etwas tun, wenn var gleich 2 ist
    abbrechen;
  Standard:
    //wenn nichts anderes zutrifft, mache den Standard
    // Voreinstellung ist optional
    abbrechen;
}
```

var: eine Variable, deren Wert mit verschiedenen Fällen verglichen werden soll. Erlaubte Datentypen: int, char.

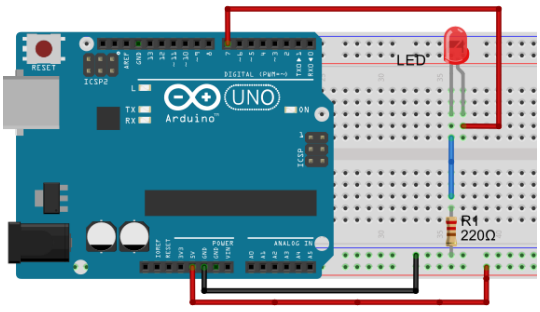
label1, label2: Konstanten. Erlaubte Datentypen: int, char.

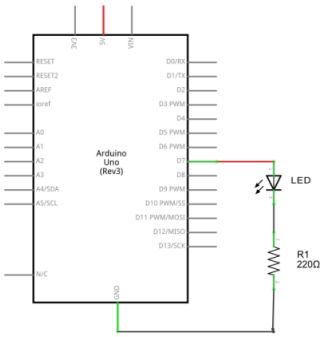
Anmerkung:

Arduino-Schaltungen werden auf zwei Arten dargestellt;

1-) Breadboard-Ansicht

2-) Schematische Darstellung





1-) Breadboard Ansicht 2-) Schematische Darstellung

Wir werden die Breadboard-Ansicht verwenden, die häufiger verwendet wird.

Genug geredet! Lasst uns produktiv werden!

Schaltungen von Arduino Button und LED Modul Kit

Die meisten Pins des Arduinos können als Eingang oder Ausgang konfiguriert werden. Das Button- und LED-Modul-Kit ist das erste Trainings-KIT von ARDUinVET, mit dem Schüler die I/O-Systeme des Arduinos erlernen können. Daher kann es als I/O Arduino Trainings-KIT bezeichnet werden. In diesem Training Kit, wie unten gezeigt, sind 6 Knöpfe als Eingänge mit dem Arduino verbunden. Und ein Summer, ein 2-poliger Stecker, sowie 6 LEDs sind als Ausgänge angeschlossen.

Die Schüler können dieses Training Kit verwenden, um die E/A-Systeme des Arduinos zu erlernen, und es erleichtert ihnen das Testen der Arduino-Schaltung. Außerdem können sie ein Breadboard verwenden, um Arduino-Schaltungen und Experimente zu testen.

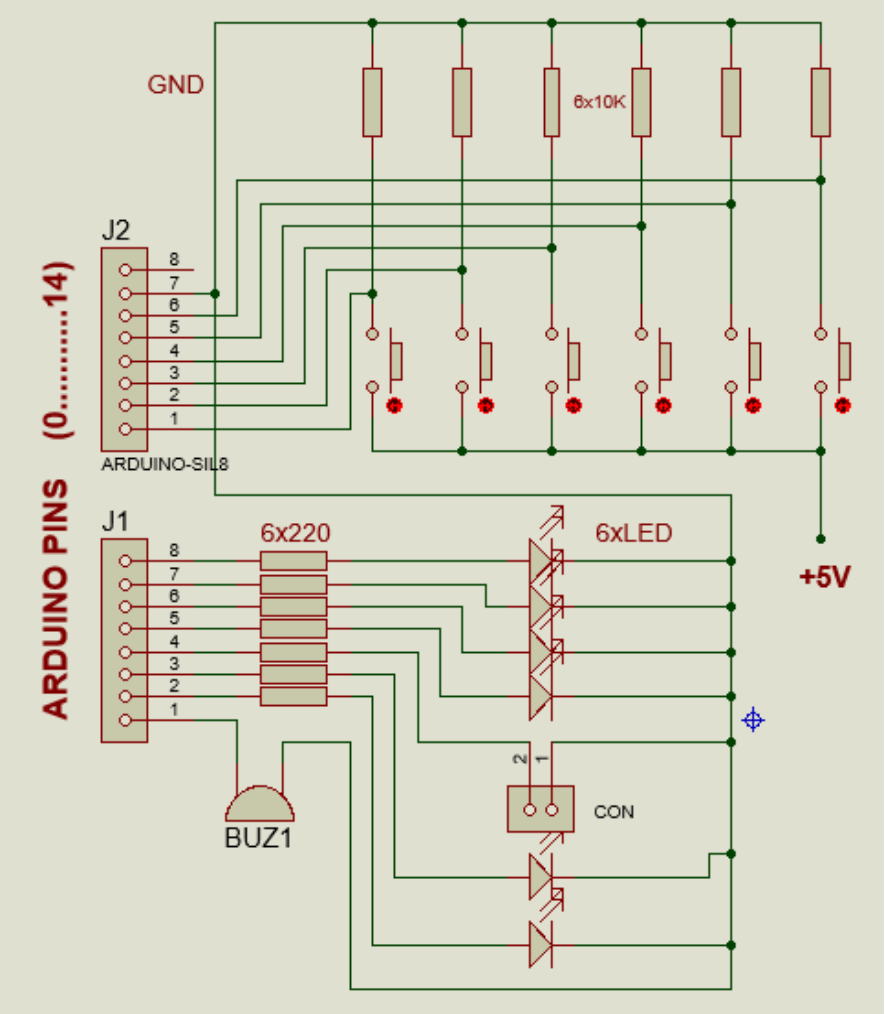


Abbildung: Offener Stromkreis von Taster und LED-Modul-Bausatz

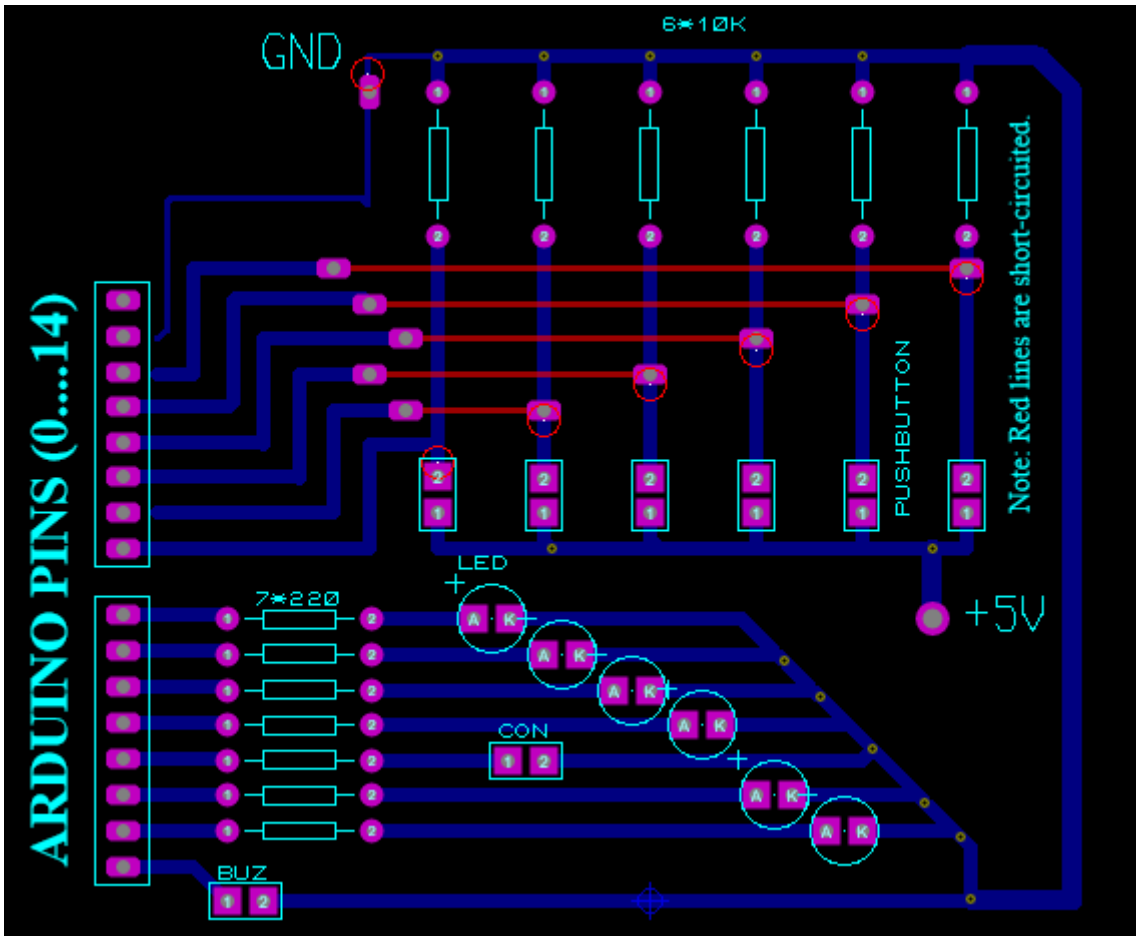


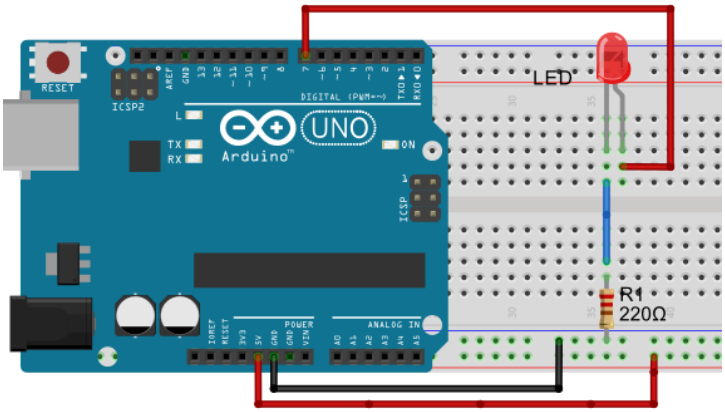
Abbildung: Leiterplattenschema des Bausatzes für Tasten- und LED-Module

Beispiele für Arduio-Schaltungen und –Programme

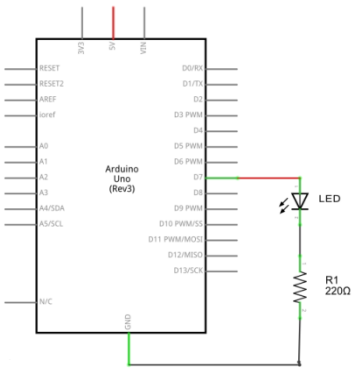
Schaltung 1:

Titel der Schaltung: LED blinkt - Programm

Schaltungsbeschreibung: Eine LED wird an den 13. Pin des Arduino angeschlossen. Die LED blinkt kontinuierlich im 1-Sekunden-Intervall.



1-) Breadboard Ansicht



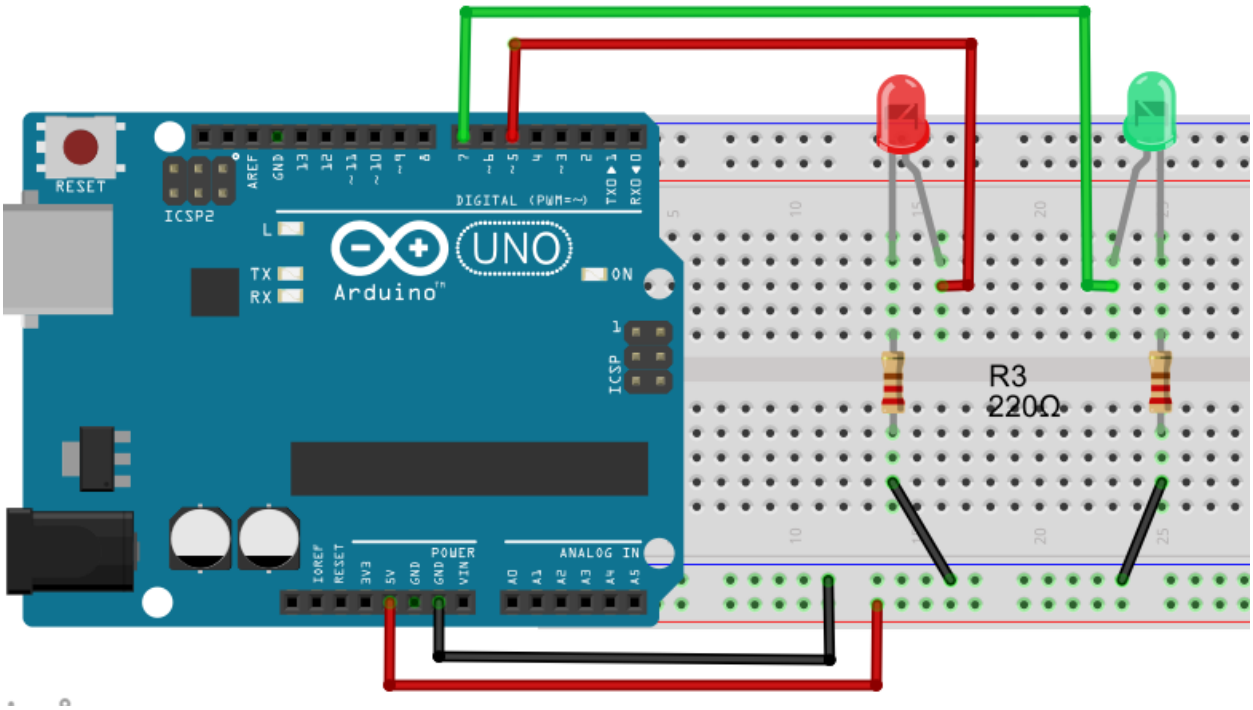
2-) Schematische Darstellung

```
/* LED-Blinkprogramm, Ein- und Ausschalten einer LED */
int led = 7; // Ganzzahlige Variable led wird deklariert

void setup() { // die Methode setup() wird nur einmal ausgeführt

pinMode(led, OUTPUT); // die led-PIN wird als Digitalausgang deklariert

}

void loop() { // die loop()-Methode wird wiederholt

digitalWrite(led, HIGH); // Einschalten der led

delay(1000); // Anhalten des Programms für 1000 Millisekunden

digitalWrite(led, LOW); // Ausschalten der led

delay(1000); // Anhalten des Programms für 1000 Millisekunden

}
```

Schaltung 2:

Titel der Schaltung: Flip-Flop , 2 LEDs blinken Programm

Schaltungsbeschreibung: 2 LEDs blinken nacheinander im Abstand von 2 Sekunden.

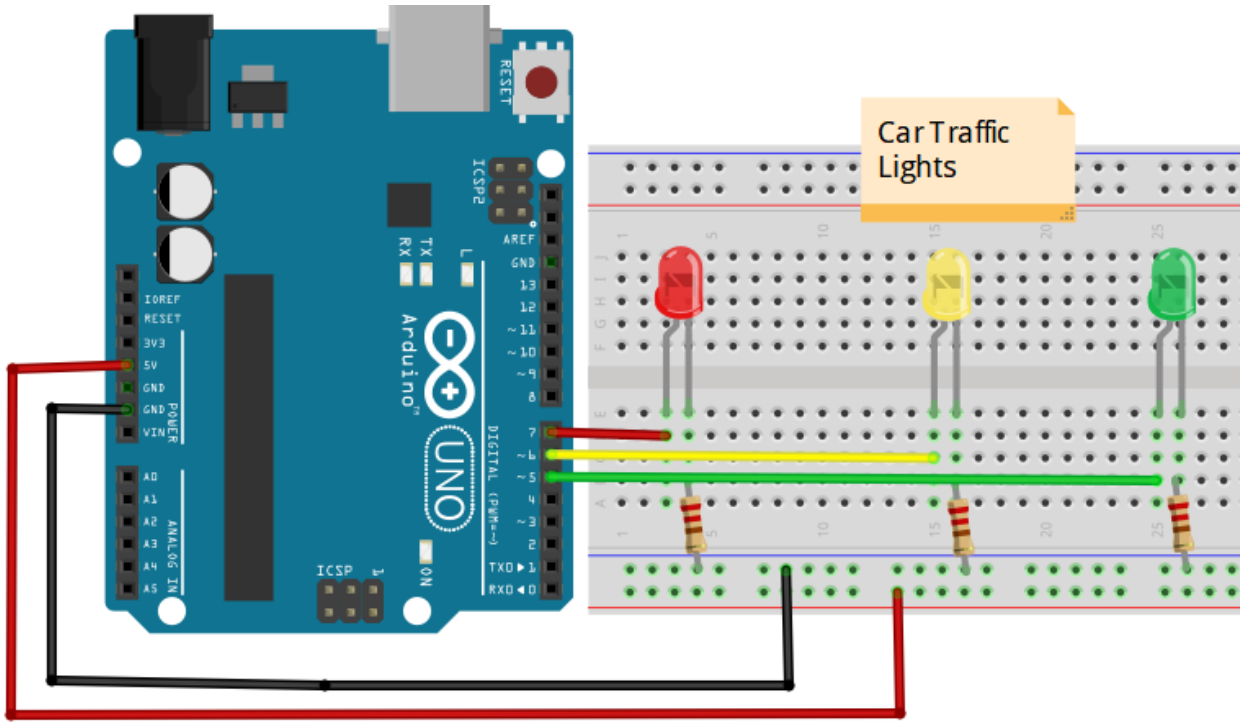


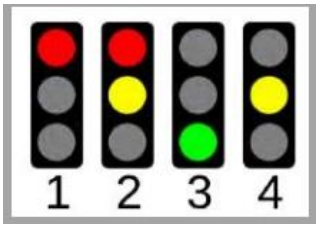

```
/*    Flip Flop     */

int greenLED=5;                    // Pin, an dem die grüne LED angeschlossen ist
int redLED=7;                      // Pin, an dem die rote LED angeschlossen ist

void setup()  {
pinMode(greenLED, OUTPUT);         // grüner LED-Pin wird als OUTPUT initialisiert
pinMode(redLED, OUTPUT);           // roter LED-Pin wird als OUTPUT initialisiert
                  }

void loop(){
digitalWrite(greenLED,  HIGH);          // Einschalten der grünen LED
digitalWrite(redLED,  LOW);             // rote LED ausschalten
pause(2000);
```

```
digitalWrite(greenLED,  LOW);          // grüne LED ausschalten
digitalWrite(redLED,  HIGH);           // rote LED einschalten
pause(2000);
                    }
```

Schaltung 3:

Titel der Schaltung:: Ampelanlagen
Beschreibung der Schaltung: In diesem Projekt werden wir eine Ampelanlage bauen. Es gibt 3 LEDs mit verschiedenen Farben (grün, gelb und rot).





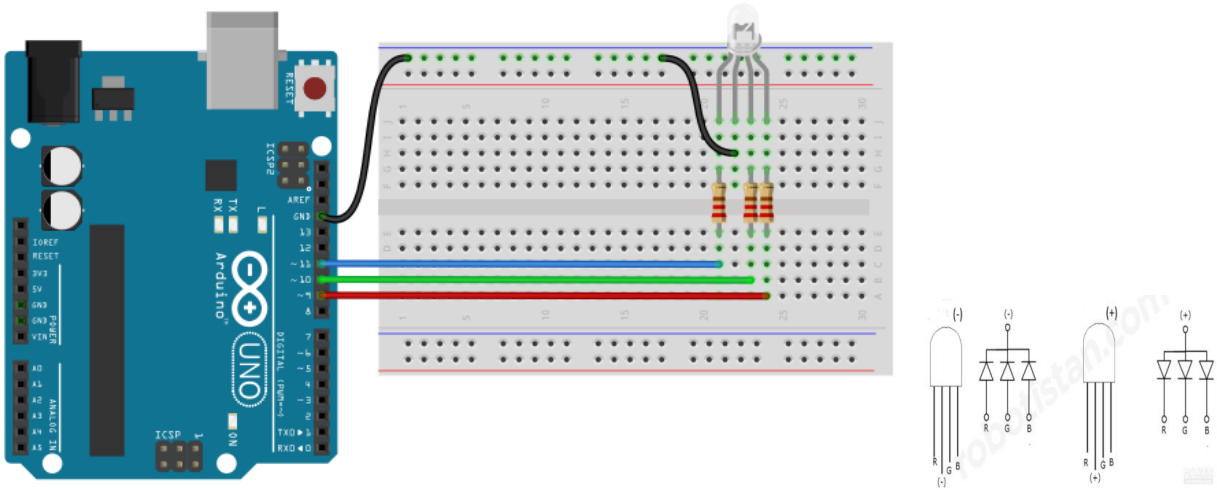

```
/* Ampelanlage */

int redLED = 7;
int yellowLED =6;
int greenLED = 5;
```

```
void setup() {                          // hier initialisieren wir unsere Pins als Ausgänge
pinMode(redLED,  OUTPUT);
pinMode(yellowLED,  OUTPUT);
pinMode(greenLED,  OUTPUT);
}

void loop() {

digitalWrite(redLED,  HIGH);            // roteLED leuchtet 9 Sekunden lang
digitalWrite(yellowLED,  LOW);
digitalWrite(greenLED,  LOW);
delay(9000);

digitalWrite(redLED,  HIGH);            // roteLED und gelbeLED leuchten 2 Sekunden lang
digitalWrite(yellowLED,  HIGH);
digitalWrite(greenLED,  LOW);
delay(2000);

digitalWrite(redLED,  LOW);             // grüneLED leuchtet 9 Sekunden lang
digitalWrite(yellowLED,  LOW);
digitalWrite(greenLED,  HIGH);
delay(9000);

digitalWrite(redLED,  LOW);             // Wieder leuchtet die gelbe LED für 2 Sekunden
digitalWrite(yellowLED,  HIGH);
digitalWrite(greenLED,  LOW);
delay(2000);

// Wieder leuchtet die gelbe LED für 2 Sekunden }
```

Schaltung 4:

Titel der Schaltung: RGB LED ANWENDUNG

Schaltungserklärung: Die RGB-LED besteht aus 3 LEDs, die gemeinsam in einem einzigen Gehäuse untergebracht sind. Es ist möglich, die Lichtintensität der drei Farben digital zu steuern. Darüber hinaus können die gewünschten Farben durch die Verwendung der PWM-Technik erhalten werden.



/* Wir lassen jede Farbe der RGB-LED in Intervallen von 1 Sekunde aufblitzen. Wenn wir weißes Licht anzeigen wollen, müssen wir alle LEDs einschalten.*/

const int BlueLed=11; // wir schließen die blaue LED an Pin-11 an

const int GreenLed=10; // wir schließen die grüne LED an Pin-10 an

const int RedLed=9; // wir schließen die rote LED an Pin-11 an

// Wir weisen die Pins, an denen die LEDs angeschlossen sind, als Ausgänge zu. **void setup() {**
pinMode(BlueLed,OUTPUT);
pinMode(GreenLed,OUTPUT);
pinMode(RedLed,OUTPUT); }

// Die Schleife beginnt hier.

void loop() {
digitalWrite(BlueLed, LOW); // RedLed ist eingeschaltet.
digitalWrite(GreenLed, LOW);
digitalWrite(RedLed, HIGH);
delay(1000);

digitalWrite(BlueLed, LOW); // Grünes Licht ist eingeschaltet .
digitalWrite(GreenLed, HIGH);
digitalWrite(RedLed, LOW);
delay(1000);

digitalWrite(BlueLed, HIGH); // BlueLed ist eingeschaltet.

```
digitalWrite(GreenLed,  LOW);
digitalWrite(RedLed, LOW);
delay(1000);
```

// Wir zeigen die weiße Farbe an, indem wir alle Leds aktivieren.

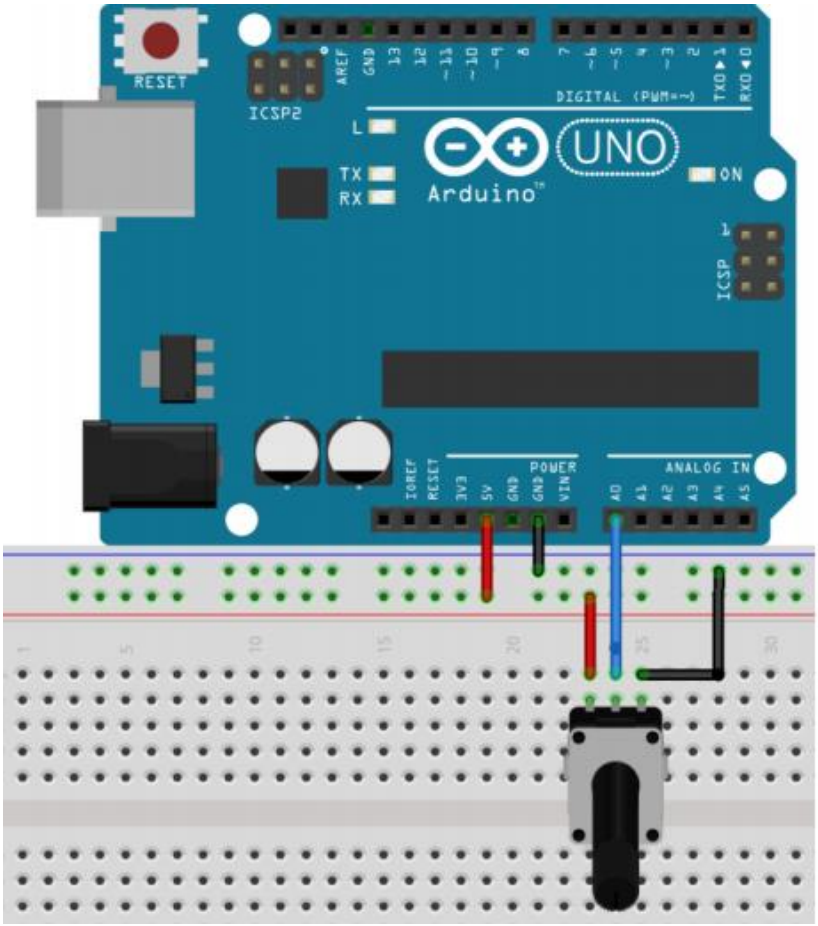
```
digitalWrite(BlueLed,  HIGH);
digitalWrite(GreenLed,  HIGH);
digitalWrite(RedLed,  HIGH);
delay(1000);
}
```

Schaltung xxxxx:

Titel der Schaltung: Lesen der analogen Spannung, Lesen des Wertes vom Potentiomete

Schaltung Erläuterung: Hier lernen wir, wie man einen analogen Eingang an Analog-Pin-0 liest. Der Eingang wird von analogRead() in eine Spannung umgewandelt und auf dem seriellen Monitor der Arduino IDE ausgegeben.

Potentiometer: Ein Potentiometer (oder Poti) ist ein einfacher elektromechanischer Wandler. Er wandelt eine Dreh- oder Linearbewegung des Bedieners in eine Widerstandsänderung um. Diese Änderung wird (oder kann) zur Steuerung von allem Möglichen verwendet werden, von der Lautstärke einer Hi-Fi-Anlage bis zur Richtung eines riesigen Containerschiffs.	



/* ReadAnalogVoltage : Liest einen Analogeingang an Pin 0, wandelt ihn in eine Spannung um und gibt das Ergebnis auf dem seriellen Monitor aus.
Eine grafische Darstellung ist mit dem seriellen Plotter möglich (Menü Extras > Serieller Plotter). Verbinden Sie den mittleren Pin eines Potentiometers mit Pin A0, und die äußeren Pins mit +5V und Masse. */

// die Setup-Routine läuft einmal, wenn Sie Reset drücken:

void setup() {

// Initialisierung der seriellen Kommunikation mit 9600 Bits pro Sekunde:

Serial.begin(9600); }

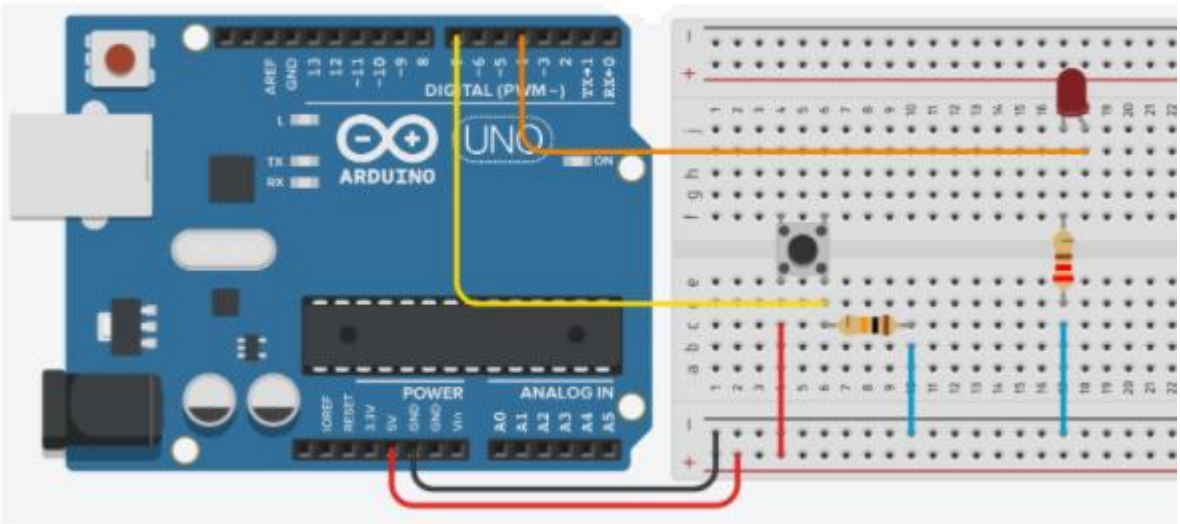
// Die Schleifenroutine läuft immer wieder und ewig:

```
void loop() {
        // Lesen Sie den Eingang am analogen Pin 0:
  int sensorValue = analogRead(A0);
       // Drucken Sie den gelesenen Wert aus:
  Serial.println(sensorValue);
  delay(1000);        // Verzögerung zwischen den Lesevorgängen für Stabilität
}
```

Schaltung xxxxx:

Titel der Schaltung: Verwendung von Tasten auf Arduinos

Schaltung Erläuterung: Wenn Sie einen an Pin 7 angeschlossenen Taster drücken, schaltet dieser eine (LED) ein und aus, die an Digitalpin 4 angeschlossen ist,




```
/* Taster Schaltet eine Leuchtdiode (LED) ein und aus, die an den digitalen Pin 4 angeschlossen ist, wenn ein an Pin 7 angeschlossener Taster gedrückt wird.       */
void setup()
{
  pinMode(4, OUTPUT);  // Pin-4 ist Ausgang
```

```
  pinMode(7, INPUT);      // Pin7 ist Eingang. 7
}
void loop()
{
   if (digitalRead(7) == HIGH)          // Wenn Pin7 "1" ist,
    digitalWrite(4, HIGH);              // Led ist eingeschaltet,
    if (digitalRead(7) == LOW)          // Wenn Pin7 "0" ist,
     digitalWrite(4, LOW);              // Led ist AUS.
}
```

ANMERKUNG: Der IF-THEN-Befehl kann auch für den Anschluss von Tastern an die Eingangspins des Arduinos verwendet werden. Diese Verbindung kann auf zwei verschiedene Arten hergestellt werden. Pull_Up-Verbindung und Pull-down-Verbindung.

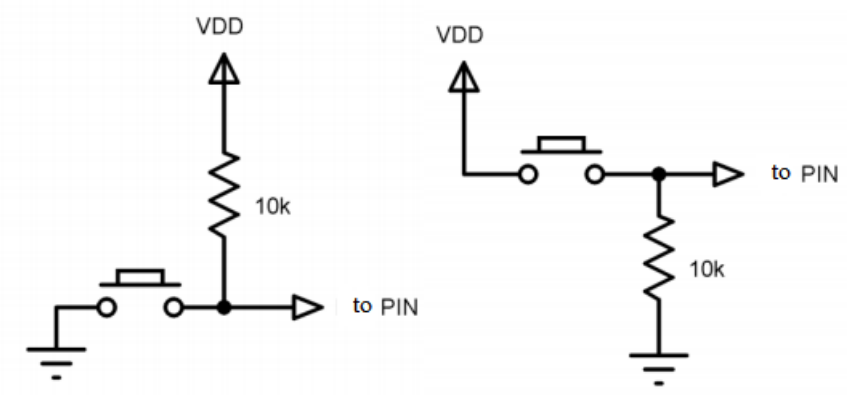
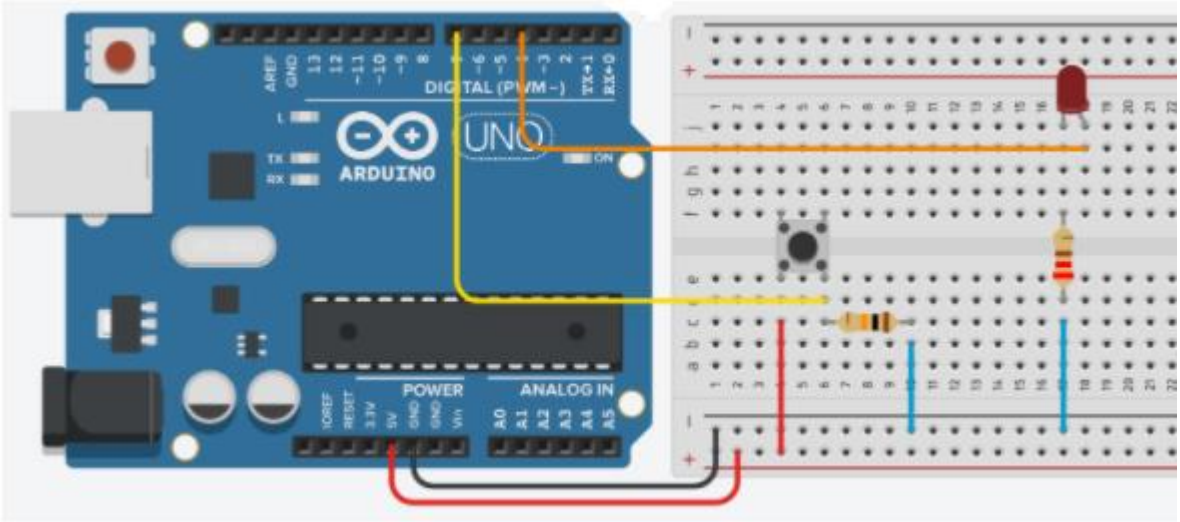


Abbildung: Schalter oder Buttons, die für den IF...THEN-Befehl verwendet werden können

Schaltung xxxxx:

Titel der Schaltung: Verwendung des ELSE-Befehls auf Arduinos

Schaltung Erläuterung: Wenn man einen Taster drückt, der an Pin 7 angeschlossen ist, schaltet der Taster eine (LED) ein und aus, die an den digitalen Pin 4 angeschlossen ist. Wir werden auch lernen, die Pins mit der Variablen "int" zu bestimmen.

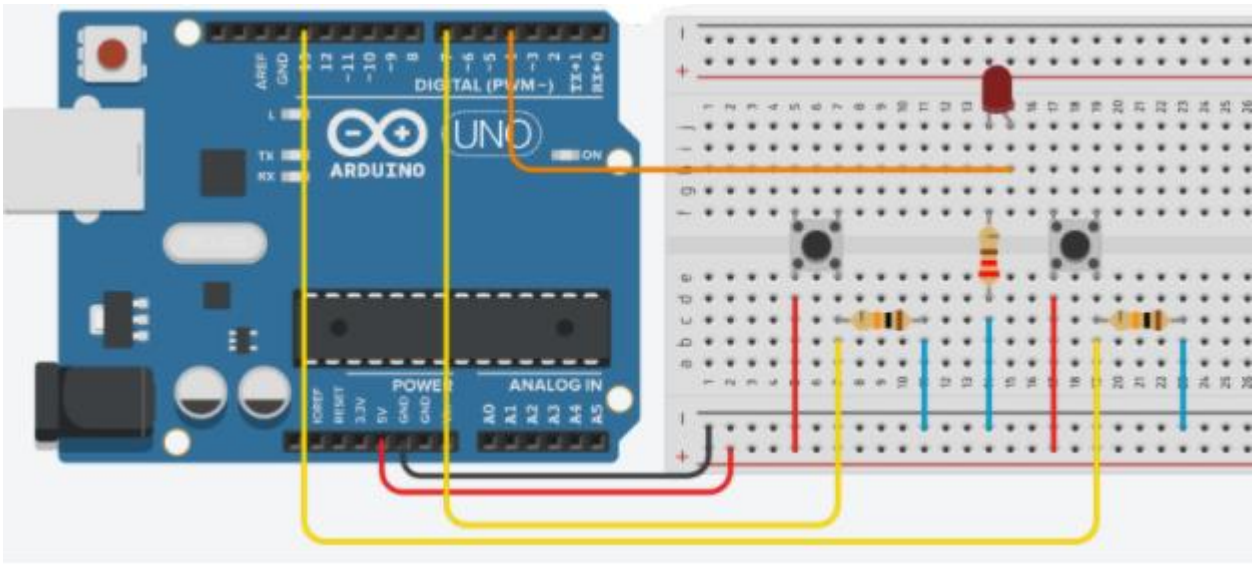



```
 int led = 4;
int buton =7;
void setup()
{
  pinMode(led, OUTPUT);
  pinMode(buton, INPUT);
}
void loop()
{
    if (digitalRead(buton) == HIGH)          // Die Schaltfläche ist eingeschaltet,
       digitalWrite(led, HIGH);              // Led ist eingeschaltet
     else                                    // Wenn nicht,
       digitalWrite(led, LOW);               // Led ist AUS.
}
```

Schaltung xxxxx:

Titel der Schaltung: 2 Knöpfe versus 1 LED

Schaltung Erläuterung: Ein Knopf schaltet die Led ein, ein anderer Knopf schaltet die Led aus.



/* 2 Tasten gegen 1 LED

Die Knöpfe sind mit 10K-Widerständen in Reihe geschaltet. */

```
int led = 4;                    // Der Pin-4 ist als "led" belegt
int button1 = 7;                // Der Pin-7 wird als "Taste1" zugewiesen
int button2 = 13;               // Der Pin-13 wird als "Taste2" zugewiesen
void setup()
{
  pinMode(led, OUTPUT);                 // led=Ausgang
  pinMode(button1, INPUT);              // Taste1=Eingabe
  pinMode(button2, INPUT);              // Taste2=Eingabe
}
void loop()
{
  if (digitalRead(button1) == HIGH)     // WENN "Taste1" eingeschaltet (aktiv) ist,
     digitalWrite(led, HIGH);           // Led ist eingeschaltet.
    if (digitalRead(button2) == HIGH)   // WENN "Taste2" eingeschaltet (aktiv) ist,
     digitalWrite(led, LOW);            // Led ist AUS.
```

}

HOW-TO Verwendung des ARDUINO SERIAL MONITOR

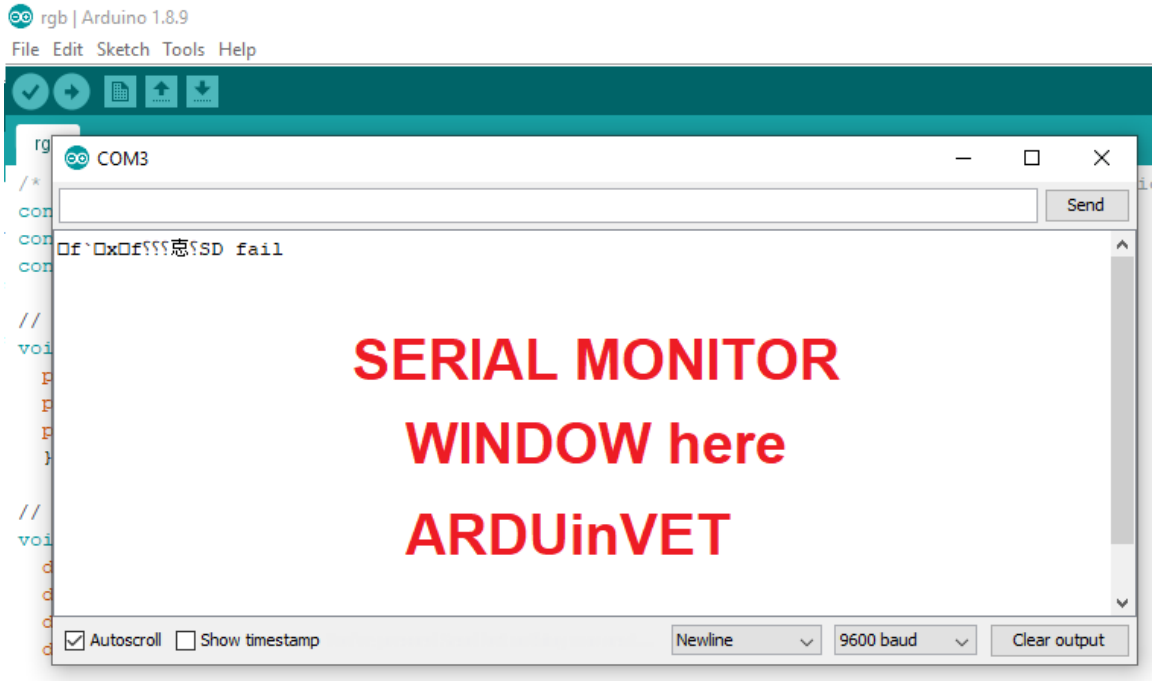
Die Arduino IDE verfügt über eine Funktion, die eine große Hilfe beim Debuggen von Skizzen oder bei der Steuerung des Arduino über die Computertastatur sein kann.

Der serielle Monitor ist ein separates Pop-up-Fenster, das als separates Terminal fungiert, das serielle Daten empfängt und sendet. Siehe das Symbol ganz rechts im Bild hier.

Serielle Daten werden über ein einziges Kabel gesendet (in unserem Fall jedoch normalerweise über USB) und bestehen aus einer Reihe von 1en und 0en, die über das Kabel gesendet werden. Die Daten können in beide Richtungen gesendet werden (in unserem Fall über zwei Drähte).

Sie werden den Serial Monitor verwenden, um Arduino Software Sketches zu debuggen oder um Daten zu sehen, die von einem funktionierenden Sketch gesendet werden. Sie müssen einen Arduino über USB an Ihren Computer angeschlossen haben, um den Serial Monitor aktivieren zu können.

Öffnen Sie den Serial Monitor, indem Sie in der IDE auf das Feld Serial Monitor klicken. Es sollte so aussehen wie auf dem Screenshot unten. Stellen Sie sicher, dass die Baudrate (Geschwindigkeit) auf 9600 eingestellt ist. Sie befindet sich in der unteren rechten Ecke. (Wichtig ist, dass sie in unserem Programm und hier gleich eingestellt ist. Da der Standardwert hier 9600 ist, stellen wir unsere



Hauptbefehle zur Verwendung von Serial Monitor

Serial.begin(); Legt die Datenrate in Bits pro Sekunde (Baud) für die serielle Datenübertragung fest. Stellen Sie sicher, dass Sie für die Kommunikation mit Serial Monitor eine der Baudraten verwenden, die im Menü in der unteren rechten Ecke des Bildschirms aufgeführt sind. Sie können jedoch auch andere Raten angeben, z. B. um über die Pins 0 und 1 mit einer Komponente zu kommunizieren, die eine bestimmte Baudrate benötigt.

Syntax: Serial.begin(speed);

Beispiel: Serial.begin(9600);

Serial.print(); Druckt Daten an der seriellen Schnittstelle als menschenlesbaren ASCII-Text aus. Dieser Befehl kann viele Formen annehmen. Zahlen werden unter Verwendung eines ASCII-Zeichens für jede Ziffer gedruckt. Fließkommazahlen werden ebenfalls als ASCII-Ziffern gedruckt, wobei zwei Dezimalstellen voreingestellt sind. Bytes werden als einzelnes Zeichen gesendet. Zeichen und Zeichenketten werden so gesendet, wie sie sind. Zum Beispiel:

Serial.print(78;) gives "78"

Serial.print('N'); gives "N"

Serial.print("Hello ArduinVet."); gives "Hello ArduinVet."

Serial.println(); Druckt Daten an der seriellen Schnittstelle als menschenlesbaren ASCII-Text, gefolgt von einem Wagenrücklaufzeichen (ASCII 13 oder '\r') und einem Zeilenumbruchzeichen (ASCII 10 oder '\n'). Dieser Befehl hat die gleichen Formen wie Serial.print().

Serial.println(val);

Serial.println(val, format); Serial.printle(13, BIN); ergibt "1101", Binärwert 15 auf dem Monitor.

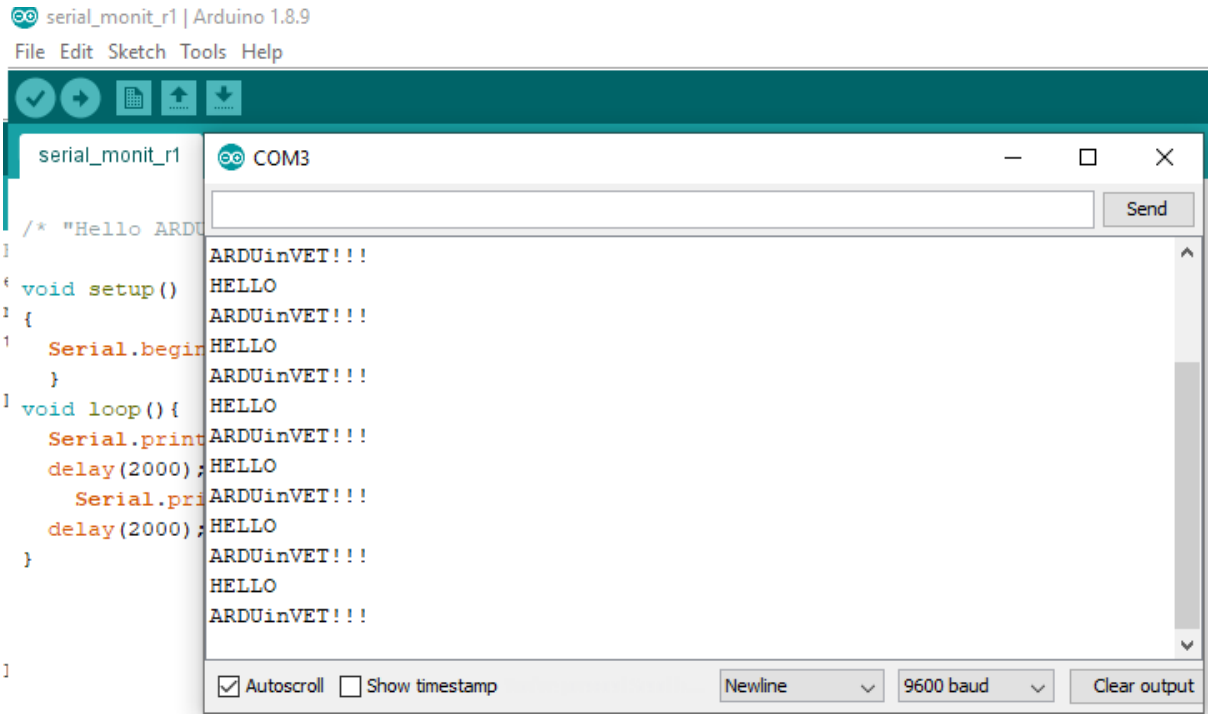
HINWEIS: Der einzige Unterschied zwischen Serial.print und Serial.println besteht darin, dass Serial.println bedeutet, dass die nächste Nachricht, die über die serielle Schnittstelle gesendet wird, in der nächsten Zeile beginnt. Es gibt noch eine dritte Neuerung, die Sie vielleicht bemerkt haben. Es steht etwas in Anführungszeichen ("). Dies wird als String bezeichnet.

Schaltung xxxxx:

Titel der Schaltung: "Hallo ARDUinVET" Anwendung im seriellen Monitor"

Schaltung Erläuterung: Hallo ARDUinVET wird auf dem seriellen Monitor zu sehen sein.

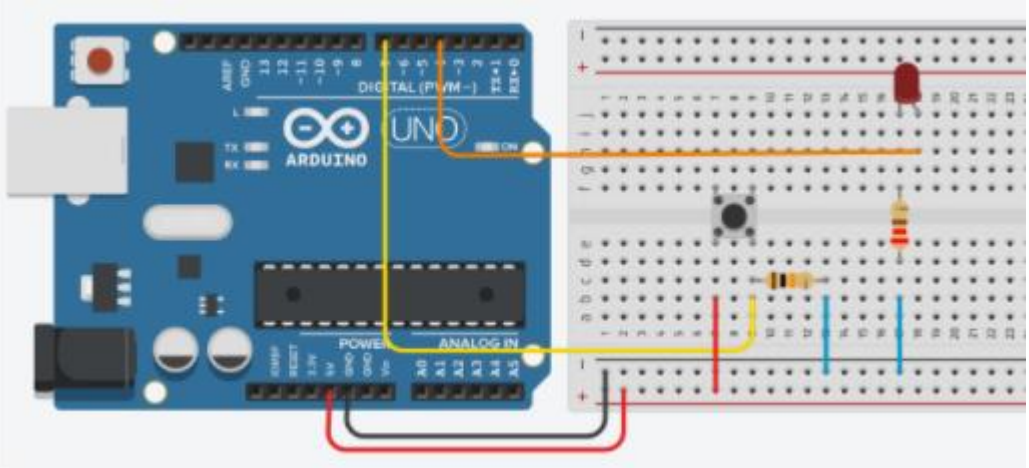
```
/* "Hallo ARDUinVET" Anwendung im seriellen Monitor */void setup()
{
  Serial.begin(9600);     // Geschwindigkeit der seriellen Kommunikation
}
void loop()
{
  Serial.println("  HELLO ");
  delay(2000);
    Serial.println("ARDUinVET!!!");
  delay(2000);
}
```



Schaltung xxxxx:

Titel der Schaltung: " Analyse des Schaltflächenstatus im seriellen Monitor"

Schaltung Erläuterung: Wenn die Taste gedrückt wird, erscheint die Meldung "Led is ligthing, LED=ON" auf dem seriellen Monitor und die Led leuchtet.
Wenn die Taste nicht gedrückt wird, erscheint die Meldung "Taste ist nicht gedrückt" auf dem seriellen Monitor und die Led leuchtet.



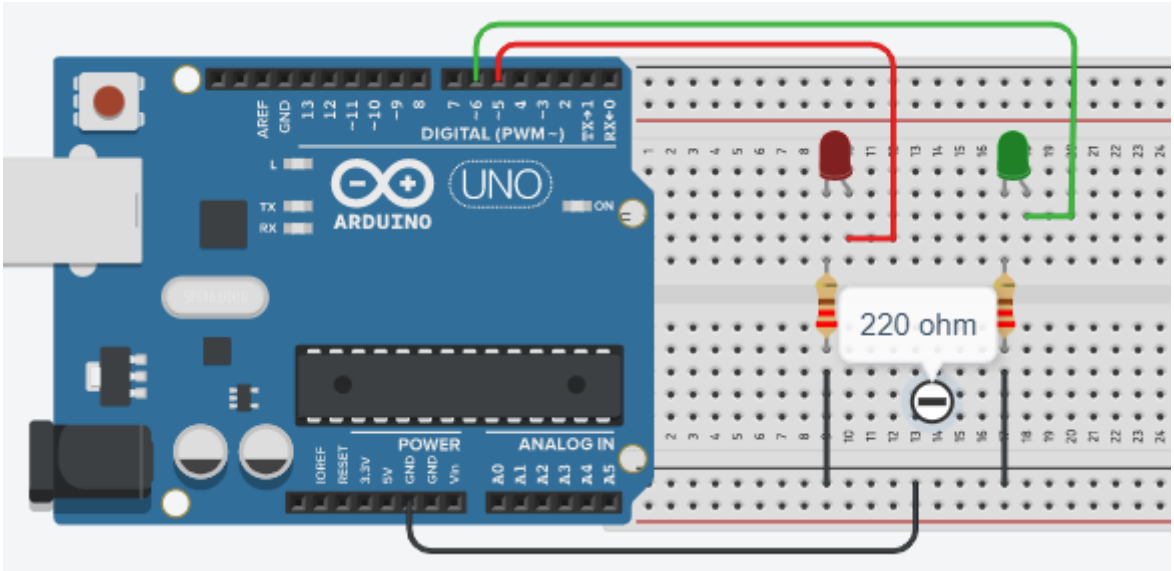

```
/* Analyse des Tastenstatus auf dem seriellen Monitor */
void setup()                    {
  pinMode(4, OUTPUT);
  pinMode(7, INPUT);
    Serial.begin(9600);         }
void loop()                      {
  if (digitalRead(7) == HIGH)         // Lesen Sie das digitale Signal an Pin-7
    {
    digitalWrite(4, HIGH);
    Serial.println("Led is ligthing, LED=ON");
    delay(250);
    }
   else                          // Wenn die vorherige Bedingung nicht erfüllt ist.
    {
    digitalWrite(4, LOW);
    Serial.println("Button is not pressed");
    delay(1000);
    }
}                                 //Die Ansicht des seriellen Monitors ist unten zu sehen.
```

```
Button is not pressed
Button is not pressed
Led is ligthing, LED=ON
Led is ligthing, LED=ON
Led is ligthing, LED=ON
Led is ligthing, LED=ON
Led is ligthing, LED=ON
Button is not pressed
```

Schaltung xxxxx:

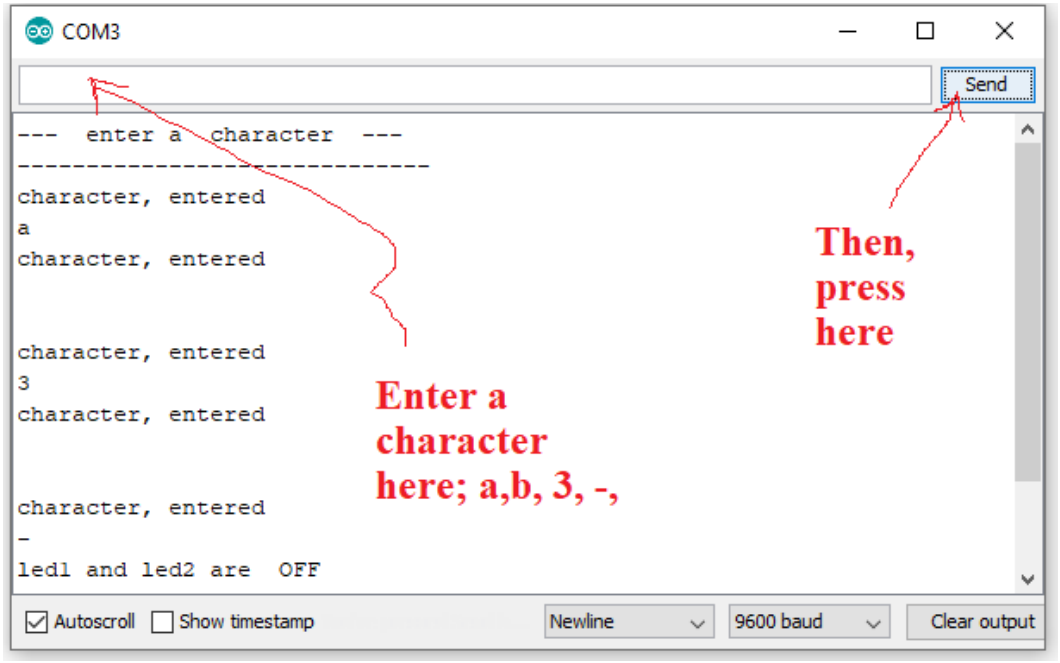
Titel der Schaltung: " Senden von Daten vom seriellen Monitor".

Schaltung Erläuterung: Wenn das Zeichen "A" auf der Tastatur gedrückt wird, leuchtet led1 auf. Wenn das Zeichen "3" auf der Tastatur gedrückt wird, leuchtet led2 auf. Wenn das "-"-Zeichen auf der Tastatur gedrückt wird, sind led1 und led2 AUS.



/* Senden von Daten vom seriellen Monitor */

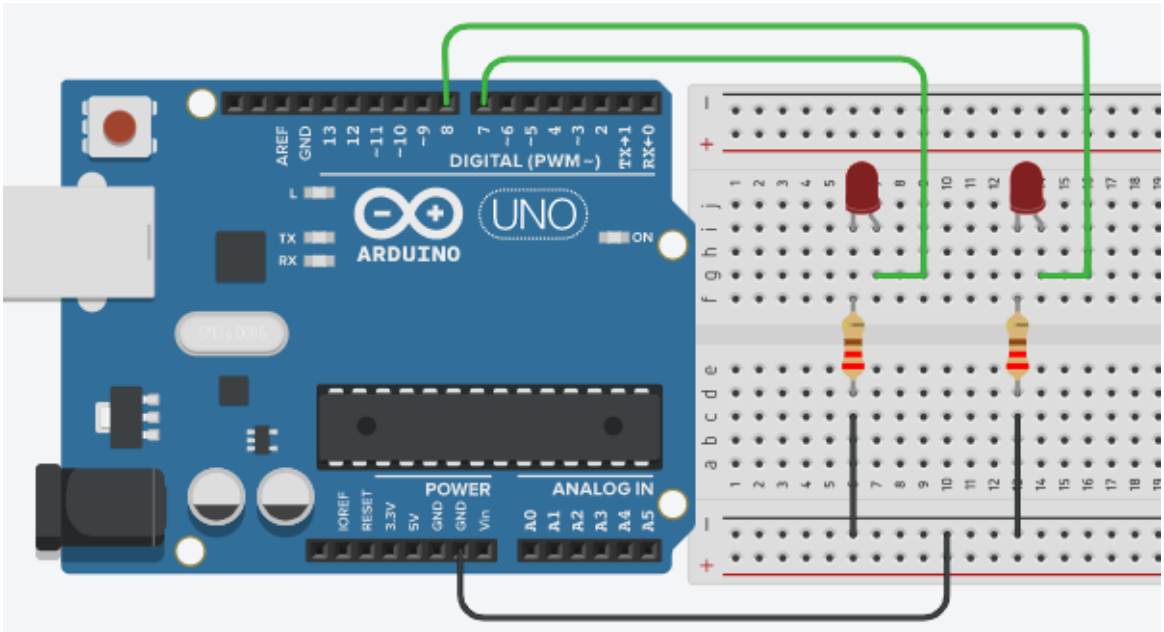
```
char character;
#define  led1 5
#define  led2 6

 void setup()   {
pinMode(led1, OUTPUT);
pinMode(led2, OUTPUT);
Serial.begin(9600);
Serial.println ("---  enter a  character  --- ");
Serial.println ("-----------------------------");
}

void loop()
{                                                                    “
if (Serial.available()>0)
{
character=Serial.read();
Serial.println ("character, entered ");
Serial.println (character);

 if (character=='A') digitalWrite (led1, 1);

if (character=='a') digitalWrite (led1, 1);

if (character=='3') digitalWrite (led2, 1);

if (character=='-')
{
digitalWrite(led1,0);
digitalWrite(led2,0);
Serial.println ("led1 and led2 are  OFF ");
}
}
}
```



Schaltung xxxxx:

Titel der Schaltung: "Den FOR-Befehl lernen"

Erläuterung des Kreislaufs:



/* "Lernen des FOR-Befehls" */

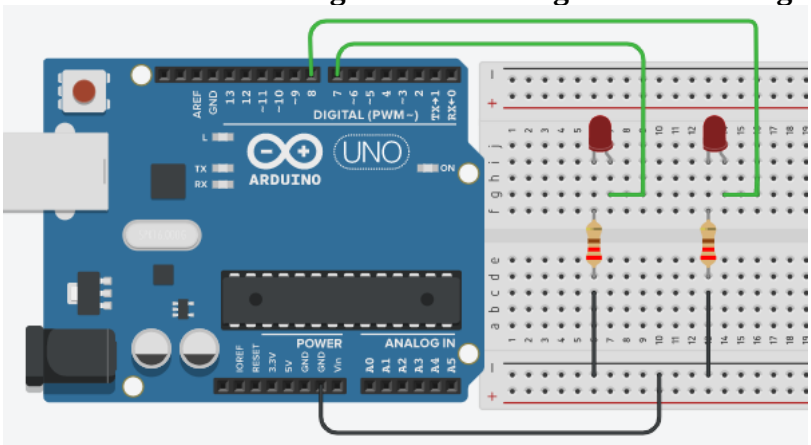
```
int   led1 = 7;
int   led2 = 8;

void setup() {
Serial.begin(9600);
pinMode(7,OUTPUT);
pinMode(8,OUTPUT);
}

void loop()
{
for(int i=1; i<6; i++)
{
Serial.println(i);
digitalWrite(led1, 1);
digitalWrite(led2, 1);
delay(1000);

 digitalWrite(led1, 0);
digitalWrite(led2, 0);
delay(1000);
}
}
```

Schaltung xxxxx:

Titel der Schaltung: " FOR-Befehl versus serieller Monitor"

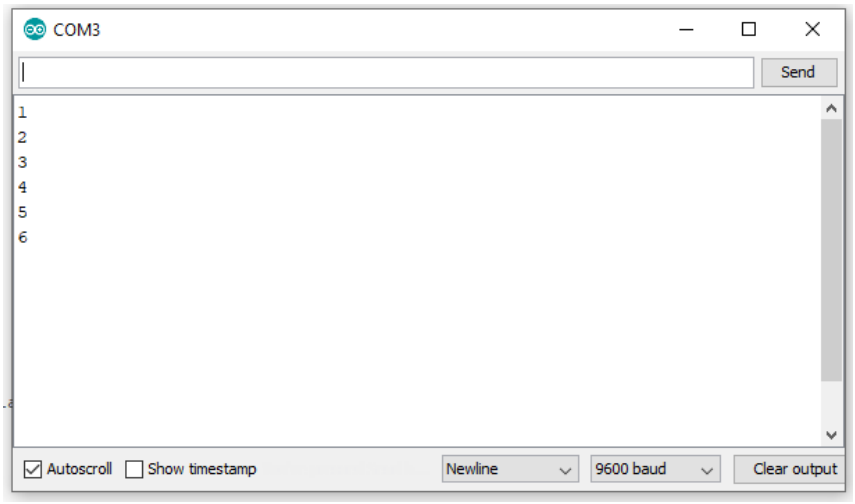
Schaltung Erläuterung: Bei dieser Anwendung blinken die LEDs 6 Mal. Die Zahlen 1, 2, 3, 4, 5, 6 erscheinen auf dem seriellen Monitor. Dann wird es in die while-Schleife eingegeben, indem die for-Schleife verlassen wird. Das Programm wird an dieser Stelle angehalten. Um neu zu starten, muss die Reset-Taste gedrückt werden.

Hier verwenden wir die gleiche Schaltung wie im vorherigen Beispiel.



```
/* "FOR Command versus Serial Monitor" */

int   led1 = 7;
int   led2 = 8;

void setup()                  {
Serial.begin(9600);
pinMode(7,OUTPUT);
pinMode(8,OUTPUT);      }

void loop() {

  for(int i=1; i<=6; i++)              // der Wert von i = 1, 2,3, 4,5, 6

  {

 Serial.println(i);                    // Schreiben Sie die Werte von i auf den seriellen Monitor.
digitalWrite(led1, 1);
digitalWrite(led2, 1);
delay(1000);
digitalWrite(led1, 0);
digitalWrite(led2, 0);
delay(1000);
}
while(1);                 //  die for-Schleife nicht in eine Endlosschleife gerät.

}
```

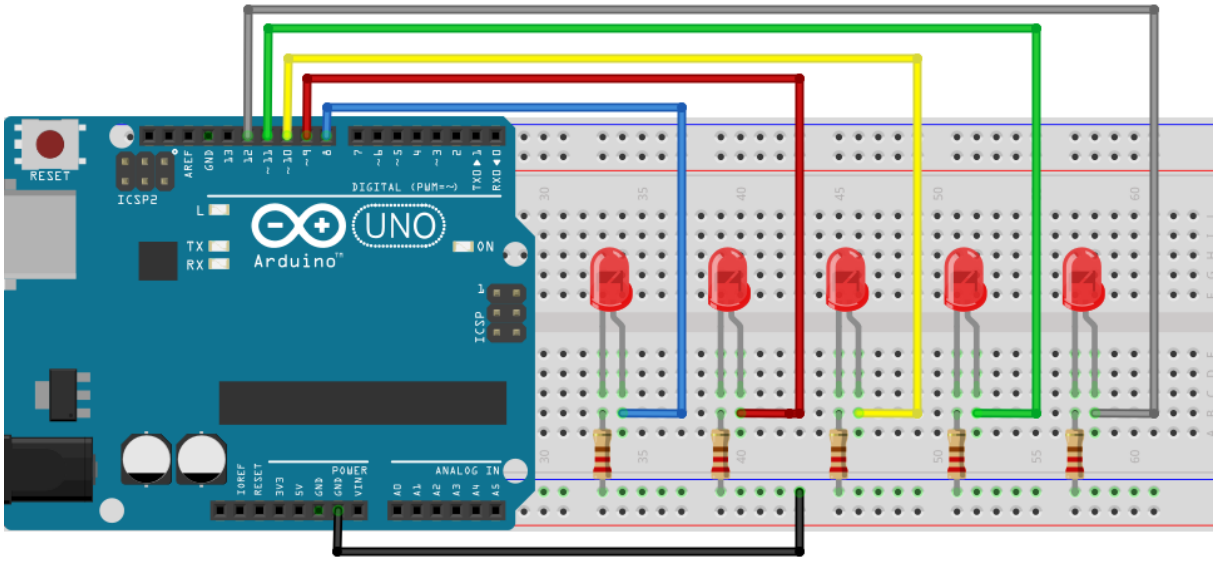
HINWEIS: Zum Neustart muss die Reset-Taste gedrückt werden.



Schaltung xxxxx:

Titel der Schaltung: " Knight Rider mit 5 Leds unter Verwendung des FOR-Befehls "

Erklärung der Schaltung:



/* Knight Rider mit 5 Leds unter Verwendung des FOR-Befehls */

```
void setup() {
pinMode(8, OUTPUT);
pinMode(9, OUTPUT);
pinMode(10, OUTPUT);
pinMode(11, OUTPUT);
pinMode(12, OUTPUT);    }

void loop() {

 for (int b=8; b<=12; b++)                    // Der Aufwärtszähler beginnt hier

  {

 digitalWrite(b, HIGH);
delay(150);
digitalWrite (b, LOW);
delay(150);

  }

  for (int b=12; b>=8; b--)                    // Der Abwärtszähler beginnt hier

  {
```

```
 digitalWrite (b, HIGH);
delay(150);
digitalWrite (b, LOW);
delay(150);

  }
}
```

Schaltung xxxxx:

Titel der Schaltung: " Erzeugung zufälliger Farben mit RGB-LED, Verwendung des Befehls switch/case"

Schaltungserklärung: Hier werden die Befehle Random und Switch/Case in der Anwendung verwendet. In dieser Schaltung werden die Farben Rot, Grün und Blau zufällig erzeugt.

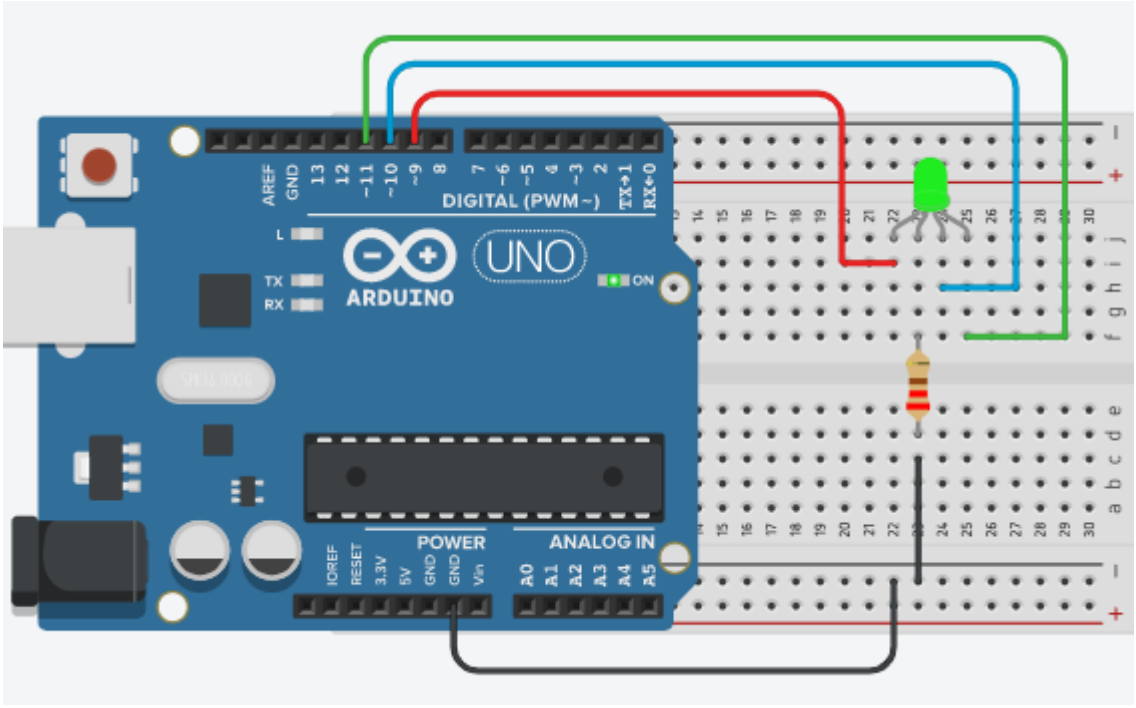
HINWEIS:

random() : Die Funktion random erzeugt Zufallszahlen.

Syntax: random(max), random(min, max)

min: Untergrenze des Zufallswertes (optional).

max: Obergrenze des Zufallswertes.




```
/*     Erzeugung zufälliger Farben mit RGB-LED, Verwendung des Befehls switch/case */

#define R  9
#define G 10
#define B 11
int colour;
int dly=3000;

void setup() {
pinMode(R, OUTPUT);
pinMode(G, OUTPUT);
pinMode(B, OUTPUT);
Serial.begin(9600);
}

void loop()
{
colour=random(4);
Serial.println(colour);

switch(colour)  {

 case 0:
digitalWrite (R, 1);              //RedLED=ON
digitalWrite (G, 0);
digitalWrite (B, 0);
delay(dly);
break;
```

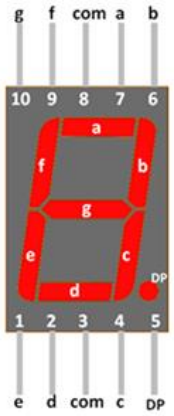
```
 case 1:
digitalWrite (R, 0);          //GreenLED=ON
digitalWrite (G, 1);
digitalWrite (B, 0);
delay(dly);
break;

 case 2:                      //BlueLED=ON
digitalWrite (R, 0);
digitalWrite (G, 0);
digitalWrite (B, 1);
delay(dly);
break;

 case 3:                      //Keine Farbe

digitalWrite (R, 0);
digitalWrite (G, 0);
digitalWrite (B, 0);
delay(dly);
break;
}
}
```

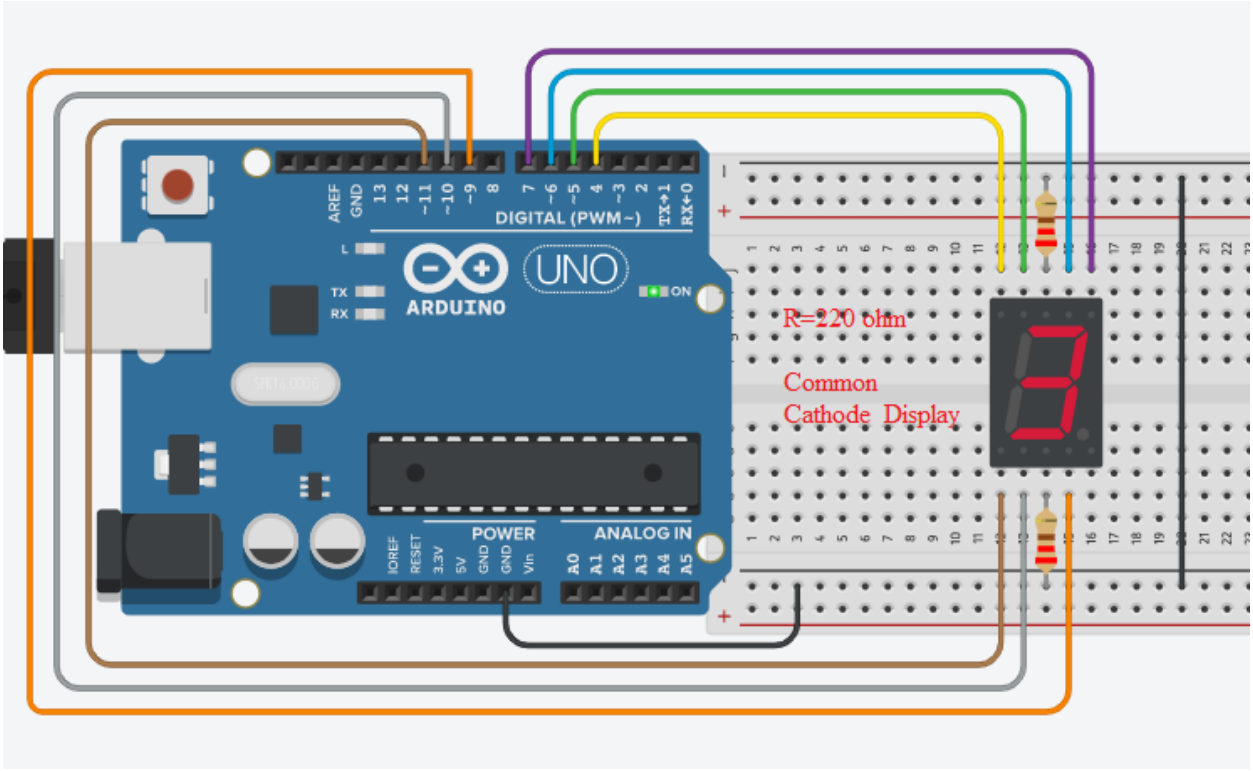
Schaltung xxxxx:

Titel der Schaltung: " 0-9 UpCounter mit 7-Segment Common-Anode-Anzeige "

Schalutungserklärung: Um eine Zahl auf dem Display zu sehen, leuchtet die entsprechende LED der 7 LEDs auf, die durch die Buchstaben a, b, c, d, e, f, g dargestellt werden.

HINWEIS: Eine Sieben-Segment-Anzeige ist ein elektronisches Anzeigegerät zur Darstellung von Dezimalzahlen.






```
/*     " 0-9 UpCounter mit 7-Segment Common-Cathoode Anzeige"     */

int       a=6,   b=7,   c=9,    d=10,     e=11,   f=5,    g=4;
int  number;

void setup() {
pinMode(a,  OUTPUT);
pinMode(b,  OUTPUT);
pinMode(c,  OUTPUT);
pinMode(d,  OUTPUT);
pinMode(e,  OUTPUT);
pinMode(f,  OUTPUT);
pinMode(g,  OUTPUT);
}

void loop()  {
for(number=0;  number< =9;  number++)    {
delay(1000);
switch(number)  {

 case
0:
digitalWrite (a, HIGH);
digitalWrite (b,
HIGH);
digitalWrite (c, HIGH);
digitalWrite (d, HIGH);
digitalWrite (e, HIGH);
digitalWrite (f, HIGH);
digitalWrite (g, LOW);
break;

case 1:
digitalWrite (a, LOW);
```

```
digitalWrite (b, HIGH);
digitalWrite (c,
HIGH);
digitalWrite (d, LOW);
digitalWrite (e,
LOW);
digitalWrite (f, LOW);
digitalWrite (g, LOW);
break;

case 2:
digitalWrite (a, HIGH);

digitalWrite (b, HIGH);
digitalWrite (c,
LOW);
digitalWrite (d, HIGH);
digitalWrite (e,
HIGH);
digitalWrite (f, LOW);
digitalWrite (g, HIGH);
break;

case 3:
digitalWrite (a, HIGH);
digitalWrite (b,
HIGH);
digitalWrite (c, HIGH);
digitalWrite (d, HIGH);

digitalWrite (e, LOW);
digitalWrite (f,
LOW);
digitalWrite (g, HIGH);
break;

case 4:
digitalWrite (a,LOW );
digitalWrite (b, HIGH);
digitalWrite (c,
HIGH);
digitalWrite (d, LOW);
digitalWrite (e,
LOW);
digitalWrite (f, HIGH);
digitalWrite (g,
HIGH);
break;

case 5:
digitalWrite (a, HIGH);
digitalWrite (b, LOW);
digitalWrite (c, HIGH);
digitalWrite (d, HIGH);
digitalWrite (e, LOW);
digitalWrite (f, HIGH);
digitalWrite (g, HIGH);
break;

case
6:
digitalWrite (a, HIGH);
digitalWrite (b, LOW);
digitalWrite (c, HIGH);
digitalWrite (d, HIGH);
digitalWrite (e, HIGH);
digitalWrite (f, HIGH);
digitalWrite (g,
HIGH);
break;

case 7:
digitalWrite (a, HIGH);

digitalWrite (b, HIGH);
digitalWrite (c, HIGH);
digitalWrite (d, LOW);
digitalWrite (e, LOW);
digitalWrite (f, LOW);
digitalWrite (g, LOW);
break;

case 8:
digitalWrite (a, HIGH);

digitalWrite (b, HIGH);
digitalWrite (c,
HIGH);
digitalWrite (d, HIGH);
digitalWrite (e, HIGH);
digitalWrite (f, HIGH);
digitalWrite (g, HIGH);
break;

case 9:
digitalWrite (a, HIGH);
digitalWrite (b,
HIGH);
```

```
digitalWrite (c, HIGH);
digitalWrite (d, HIGH);

digitalWrite (e, LOW);
digitalWrite (f, HIGH);
digitalWrite (g, HIGH);
break;
}
}
}
```

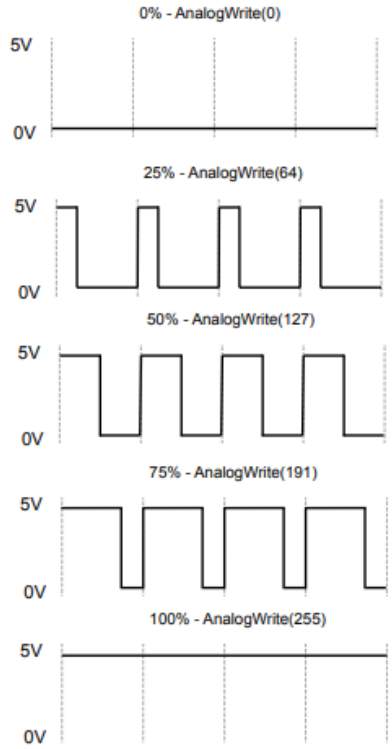
Schaltung xxxxx:

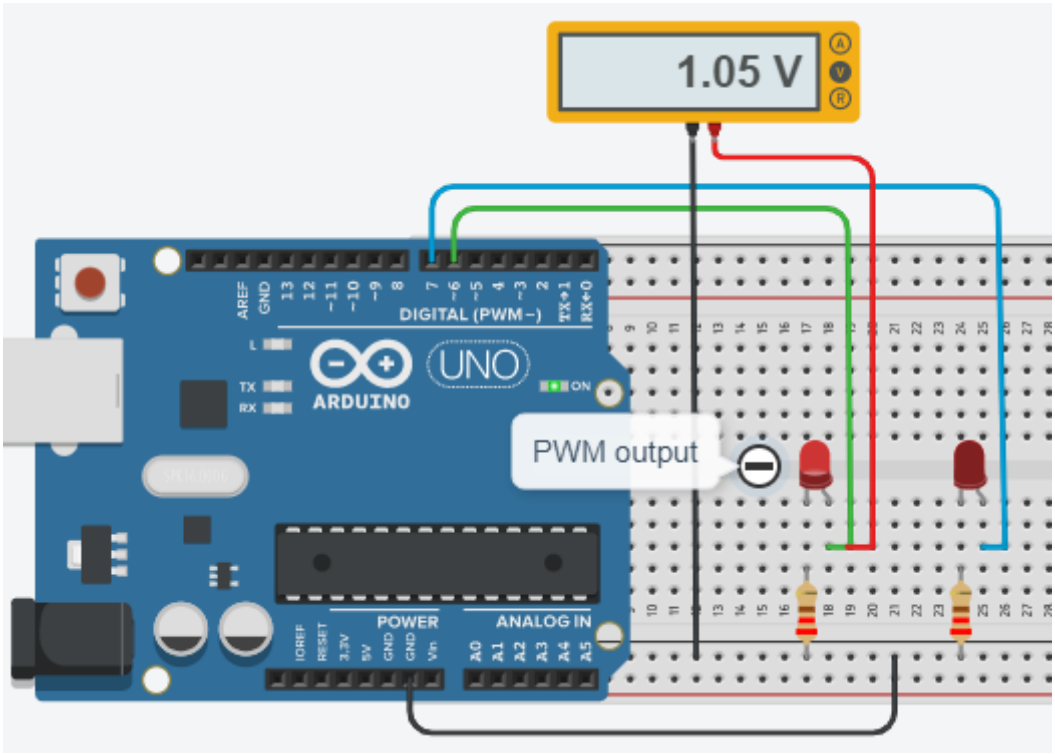
Schaltungstitel: " Anwendung der PWN-Methode"

Schaltungserklärung: In der Praxis werden Pin-6 als PWM-Ausgang und Pin-7 als Digitalausgang verwendet. Daher soll der Unterschied zwischen ihnen beobachtet werden. Anwendungen wie die Einstellung der LED-Helligkeit und die Steuerung der Motordrehzahl können mit der PWM-Methode durchgeführt werden.

HINWEIS: Der Arduino unterstützt PWM (an bestimmten Pins, die mit einer Tilde (~) auf Ihrem Arduino-Board gekennzeichnet sind - Pins 3, 4, 5, 9, 10 und 11) mit 500 Hz. (500 Mal pro Sekunde.) Sie können einen Wert zwischen 0 und 255 eingeben. 0 bedeutet, dass es nie 5V sind. 255 bedeutet, dass er immer 5 V beträgt. Zu diesem Zweck rufen Sie analogWrite() mit dem Wert auf. Das Verhältnis der "EIN"-Zeit zur Gesamtzeit wird als "Tastverhältnis" bezeichnet. Ein PWM-Ausgang, der die Hälfte der Zeit eingeschaltet ist, hat ein Tastverhältnis von 50 %.

Man kann sich PWM so vorstellen, dass er für x/255 eingeschaltet ist, wobei x der Wert ist, den Sie mit analogWrite() senden. Nachfolgend ein Beispiel, das zeigt, wie die Impulse aussehen:





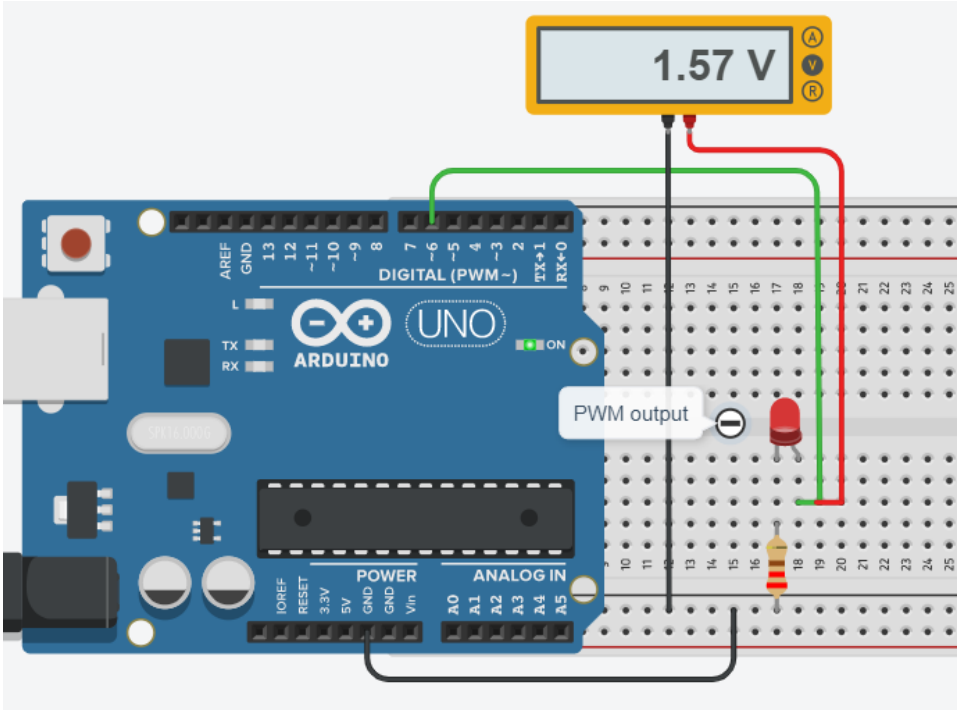
/* Erlernen der PWN-Technik mit Hilfe von analogWrite() */

```
#define    led1  6
#define    led2  7

void setup()   {
pinMode(led1, OUTPUT);
pinMode(led2, OUTPUT);       }

void loop()                {

 analogWrite(led1, 60);        //  Der Wert 60 wird an den Stift Led1 gesendet, d. h. 1,05 V.

 analogWrite(led2,60);         //  Der Wert 60 wird an den Stift Led2 gesendet, d. h. 1,05 V.

delay (100);                   // nur die led1 leuchtet.
}
```

Schaltung xxxxx:

Schaltungstitel: " So ändern Sie die Helligkeit einer LED von Minimum auf Maximum."

Schaltungserklärung: Mit Hilfe der PWM-Technik wird die Helligkeit einer LED vom Minimum zum Maximum verändert. Hier werden analogWrite () und die for-Befehle zusammen verwendet.



/* So ändern Sie die Helligkeit einer LED von Minimum auf Maximum */

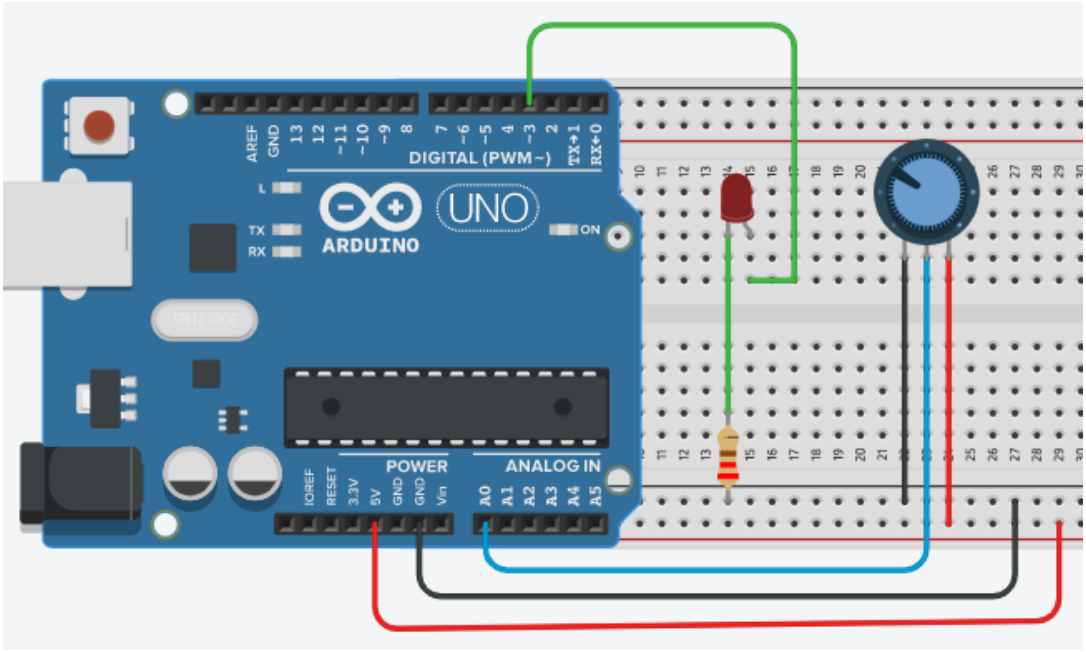
```
#define led1    6
int a;

void setup()          {
Serial.begin(9600);
pinMode(led1, OUTPUT);    }

void loop() {

 for (a=0; a<=255; a++) {

 Serial.println(a);

analogWrite(led1, a);

 delay (100);
}                }
```

HINWEIS: Auf dem Voltmeter werden Werte im Bereich von 0 bis 5 Volt angezeigt. Zählende Zahlen von 0 bis 255 werden auf dem seriellen Monitor angezeigt.

Schaltung xxxxx:

Schaltungstitel: " Potentiometer blendet LED ein."

Schaltungserklärung: Mit Hilfe eines Potentiometers (10K) wird die Helligkeit einer LED vom Minimum zum Maximum verändert. Hier wird die Funktion von map() verwendet.



Hinweis:

map() Function:

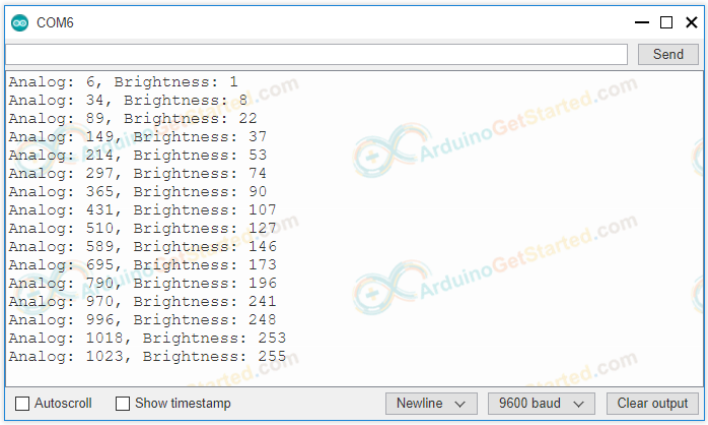
Bildet eine Zahl von einem Bereich auf einen anderen um. Das heißt, ein Wert von "fromLow" wird auf "toLow" abgebildet, ein Wert von "fromHigh" auf "toHigh", Werte dazwischen auf Werte dazwischen, usw.
Die Werte werden nicht auf den Bereich beschränkt, da Werte außerhalb des Bereichs manchmal beabsichtigt und nützlich sind. Die Funktion constrain() kann entweder vor oder nach dieser Funktion verwendet werden, wenn Grenzen für die Bereiche gewünscht sind.
Die Funktion map() verwendet ganzzahlige Berechnungen und erzeugt daher keine Brüche, auch wenn die Berechnungen dies nahelegen würden. Restbruchteile werden abgeschnitten und nicht gerundet oder gemittelt.

Syntax: map(value, fromLow, fromHigh, toLow, toHigh);

Beispiel: val = map(val, 0, 1023, 0, 255);

```
/*     Potentiometer blendet LED ein     */

int LED_PIN = 3;

void setup() {
Serial.begin(9600);
pinMode(LED_PIN, OUTPUT);      }

void loop() {
```

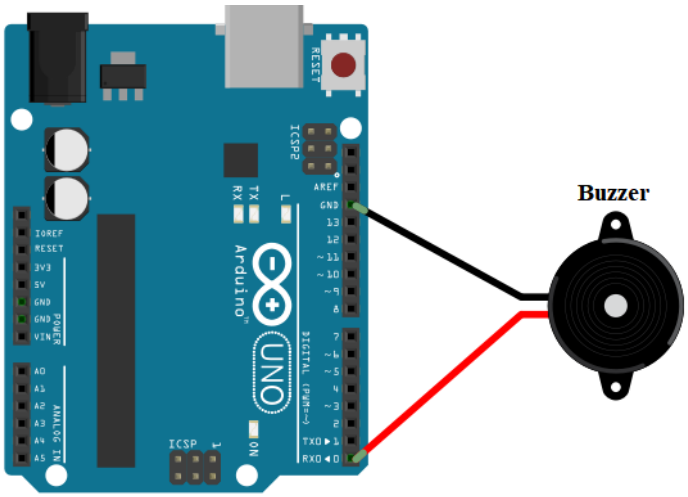
```
 // liest den Eingang am Analogpin A0 (Wert zwischen 0 und 1023)
int analogValue = analogRead(A0);
 // skaliert es auf Helligkeit (Wert zwischen 0 und 255)
int brightness = map(analogValue, 0, 1023, 0, 255);
 // stellt die Helligkeit der LED ein, die an Pin 3 angeschlossen ist
analogWrite(LED_PIN, brightness);
 // den Wert ausdrucken
 Serial.print("Analog: ");
 Serial.print(analogValue);
 Serial.print(" Brightness: ");
 Serial.println(brightness);
 delay(100);
// Serial Monitor Bildschirm i unten
}
```



Schaltung xxxxx:

Schaltungstitel: " To use a buzzer"

Schaltungserklärung: Ein Buzzer ist ein Audiosignalgerät, das mechanisch, elektromechanisch oder piezoelektrisch (kurz Piezo) sein kann. Typische Verwendungszwecke von Summern in der Industrie sind Alarmgeräte, die ein summendes oder piependes Geräusch erzeugen, während sie brummen müssen.

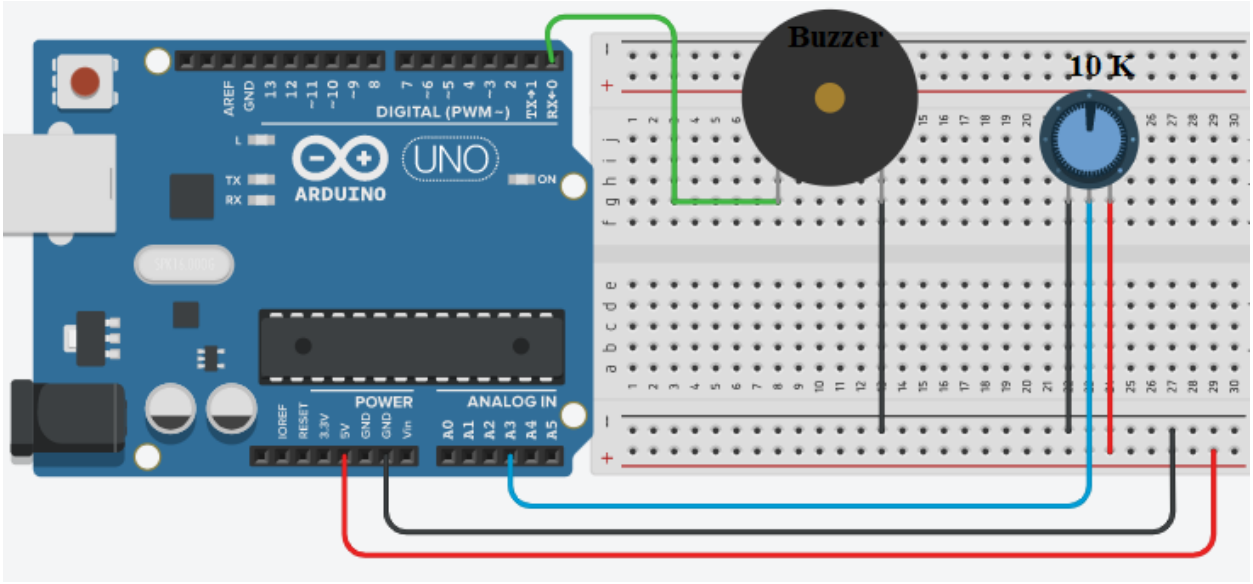



```
/* Verwendung eines Summers in Arduino-Schaltungen     */

#define buzzer_pin  0

 void setup()   {

 pinMode(buzzer_pin, OUTPUT);          }

void loop()        {

 digitalWrite(buzzer_pin, HIGH);

 delay(500);

 digitalWrite(buzzer_pin, LOW);

 delay(500);      }
```

Schaltung xxxxx:

Schaltungstitel: " Steuerung eines Summers mit einem Potentiometer "

Circuit Explanation: " When the the value of potentiometer is higher than 500, the buzzer sounds."




```
/* Steuerung eines Summers mit einem Potentiometer  */

const int POT_PIN = A3;              // Arduino-Pin verbunden mit Pot-Pin

const int BUZZER_PIN  = 0;          // Arduino-Pin verbunden mit dem Pin des Summers
const int ANALOG_THRESHOLD  = 500;
int analogValue;

void setup() {

  pinMode(BUZZER_PIN, OUTPUT);      // Arduino-Pin auf Ausgabemodus setzen
}

void loop() {

 analogValue = analogRead(POT_PIN);            // Lesen Sie den Eingang am analogen Pin

if(analogValue > ANALOG_THRESHOLD)
digitalWrite(BUZZER_PIN, HIGH);               // Buzzer einschalten

else
digitalWrite(BUZZER_PIN, LOW);                // Buzzer ausschalten

}
```

Erasmus+ KA-202

Strategisches Partnerschaftsprojekt für berufliche Bildung und Ausbildung

Projekttitel: "Lehren und Lernen mit Arduinos in der Berufsbildung"

Projekt-Akronym: "ARDUinVET"

Projekt Nr.: “2020-1-TR01-KA202-093762”

LCD-Modul und Schulungs-KIT

In diesem Modul wollen wir erklären, wie man Statusmeldungen oder Sensormesswerte des Arduino auf LCD-Displays anzeigt. Sie sind sehr weit verbreitet und eine schnelle Möglichkeit, eine lesbare Schnittstelle zu Ihrem Projekt hinzuzufügen.

Ein LCD ist die Abkürzung für Liquid Crystal Display. Es handelt sich um eine Anzeigeeinheit, die Flüssigkristalle verwendet, um ein sichtbares Bild zu erzeugen. Wenn Strom an diese spezielle Art von Kristall angelegt wird, wird er undurchsichtig und blockiert die Hintergrundbeleuchtung, die sich hinter dem Bildschirm befindet. Infolgedessen wird dieser bestimmte Bereich im Vergleich zu anderen dunkel. Und so werden die Zeichen auf dem Bildschirm angezeigt.

Anschluss eines 16×2-Zeichen-LCD-Moduls an den Arduino

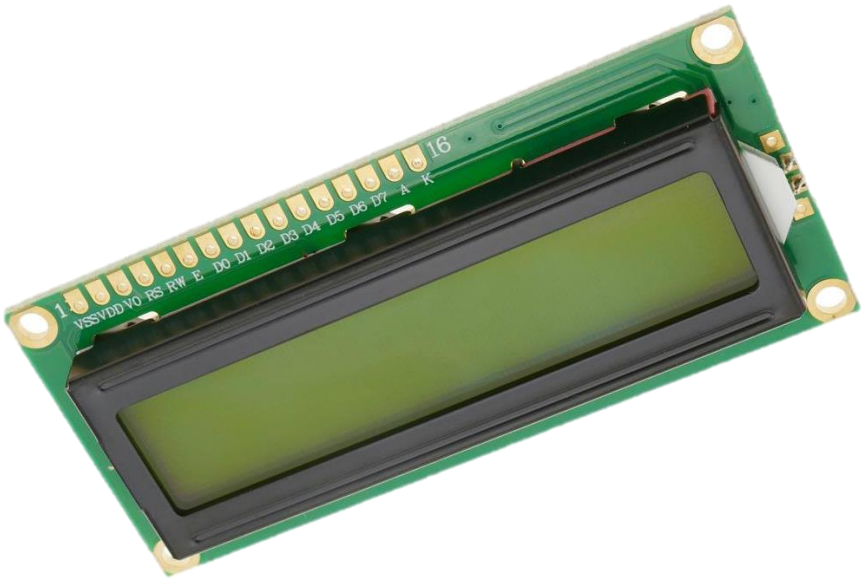
Es gibt verschiedene Arten von LCD-Displays, die Sie an Ihren Arduino anschließen können. Die gängigste basiert auf einem LCD-Controller-Chip mit paralleler Schnittstelle von Hitachi, dem HD44780.

Diese LCDs sind ideal für die Anzeige von Text/Zeichen, daher der Name "Character LCD". Das Display hat eine LED-Hintergrundbeleuchtung und kann 32 ASCII-Zeichen in zwei Reihen mit je 16 Zeichen anzeigen.

Für jedes Zeichen gibt es ein kleines Rechteck auf dem Display, jedes dieser Rechtecke ist ein Raster von 5×8 Pixeln.

Obwohl sie nur Text anzeigen, gibt es sie in vielen Größen und Farben: zum Beispiel 16×1, 16×4, 20×4, mit weißem Text auf blauem Hintergrund, mit schwarzem Text auf grünem Hintergrund und viele mehr.

Die Arduino-Gemeinschaft hat bereits eine Bibliothek für HD44780-LCDs entwickelt (LiquidCrystal Library**)**; Wir werden sie also in Kürze miteinander verknüpfen.



Unten sehen Sie die LCD-Pinbelegung:

- VSS: ist ein Masse-Pin und sollte mit der Masse des Arduino verbunden werden;

-

- VDD: ist mit 5 V verbunden;

- VD: für die Kontrasteinstellung (verbunden mit dem mittleren Pin des Potentiometers);

- RS: steuert, in welchem Speicherbereich des LCD die gesendeten Daten gespeichert werden;

- R/W: Pin zur Auswahl des Lese-/Schreibmodus;

- E: wenn aktiviert, ermöglicht es dem LCD-Modul, spezielle Befehle auszuführen;

- D0 bis D7: Datenübertragung;

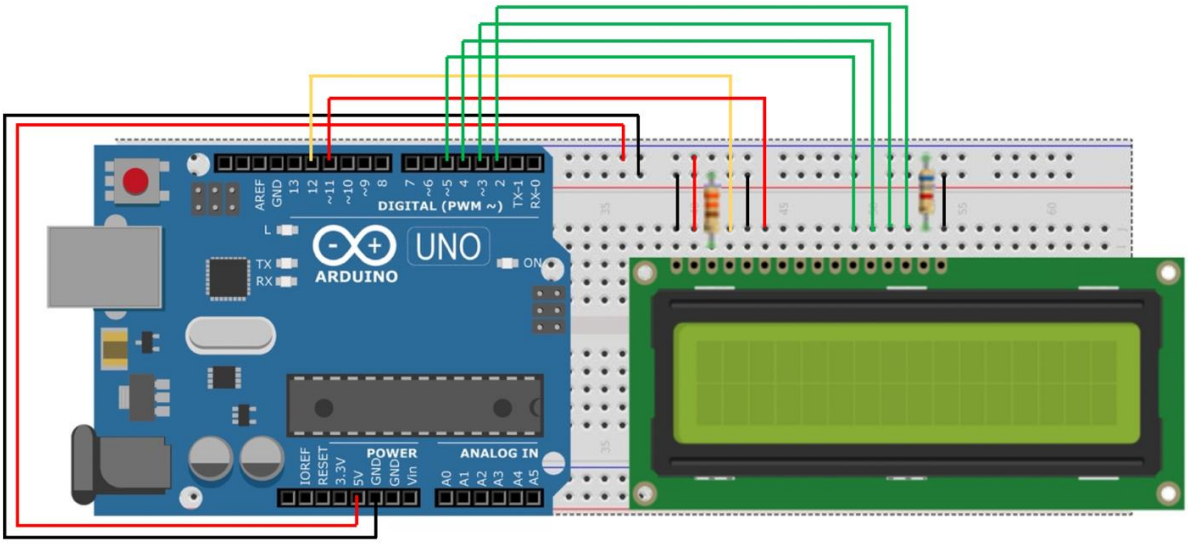
- A und K: Anode und Kathode für die Hintergrundbeleuchtung des LCD-Moduls.

Schaltung 1:

Titel der Schaltung: "Drucken einer Nachricht auf dem LCD"

Erläuterung der Schaltung: Es ist möglich, ein LCD-Display an den Mikrocontroller anzuschließen und die gewünschten Nachrichten auf den Bildschirm zu drucken.

Hinweis: Sie müssen die Bibliothek LiquidCrystal.h einbinden, um die unten beschriebenen LCD-Funktionen nutzen zu können.



/* Drucken einer Nachricht */

#include <LiquidCrystal.h> // den Bibliothekscode einbinden

//Konstanten für die Anzahl der Zeilen und Spalten des LCDs

const int numRows = 2;

const int numCols = 16;

// Initialisierung der Bibliothek mit den Nummern der Schnittstellenpins

```
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup()

{

lcd.begin(numCols, numRows);

lcd.print("hello, world!"); // Eine Nachricht auf dem LCD ausgeben.

}
```

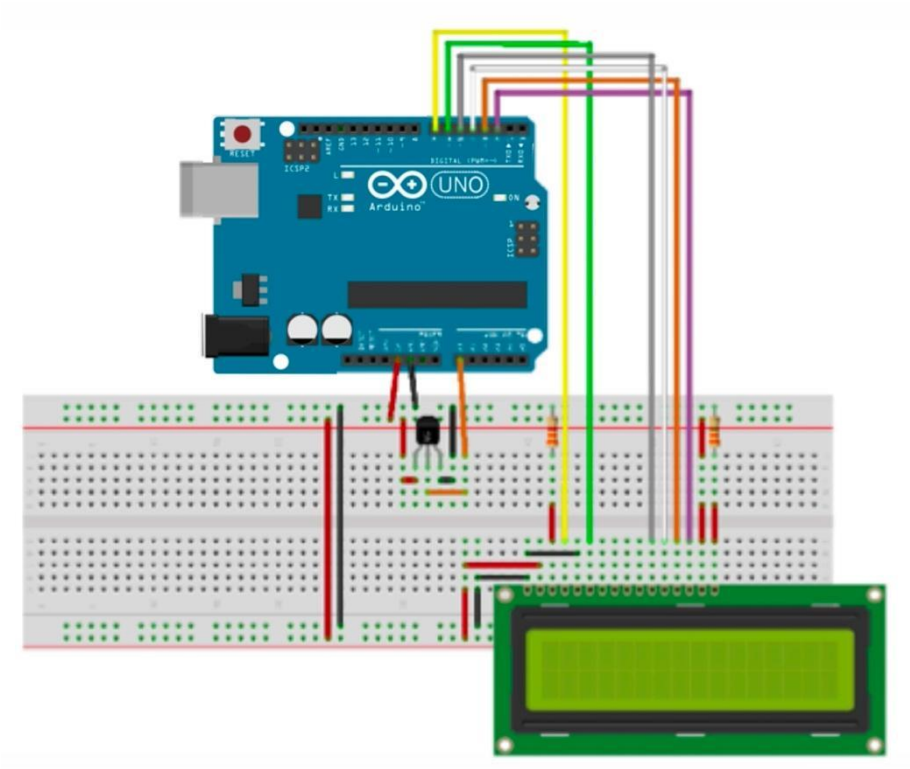
Schaltung 2:

Schaltkreistitel: "Digitales Thermometer"

Erläuterung der Schaltung: Erstellen eines digitalen Thermometers mit Arduino. Die Temperatur wird mit dem LM35-Temperatursensor gemessen und muss auf einem LCD-Display angezeigt und jede Sekunde aktualisiert werden.

Mit dem Vo-Pin des LCD-Displays wird der Kontrast der auf dem Display angezeigten Zeichen eingestellt. Wie bereits erwähnt, muss er an einen 3,3 kΩ-Widerstand angeschlossen werden, der mit GND verbunden ist. Bei einigen Displays kann es jedoch erforderlich sein, den Vo-Pin direkt mit GND zu verbinden.

Hinweis: Der LM35-Sensor verfügt über drei Anschlüsse: einen für die Stromversorgung, einen für die Masse und einen für den Ausgang der Spannung, die proportional zur erfassten Temperatur ist, die 10 mV pro Grad Celsius entspricht und in Grad Celsius kalibriert ist.



/* Digitales Thermometer */

```
#include <LiquidCrystal.h> //Bibliothek zur Ansteuerung der LCD-Anzeige

#define pin_temp A0 //Temperatursensor Vout Fußanschlusspin

float temp = 0; //Variable, in der die ermittelte Temperatur gespeichert wird

LiquidCrystal lcd(7, 6, 5, 4, 3, 2); //Initialisierung der Bibliothek mit LCD-Anzeigepins


void setup()

{

lcd.begin(16, 2); //Setzen der Anzahl der Spalten und Zeilen in der LCD-Anzeige lcd.setCursor(0, 0);
//Bewegen des Cursors über die erste Zeile (Zeile 0) und die erste Spalte lcd.print ("Temperatur:");
Drucken der Meldung 'Temperatur:' in der ersten Zeile

//*Einstellung der ADC-Vref auf 1,1V

(für größere Genauigkeit bei der Temperaturberechnung)

WICHTIG: Wenn Sie Arduino Mega verwenden, ersetzen Sie INTERNAL durch INTERNAL1V1 */
analogReference(INTERNAL);

}
```

```
void Schleife()

{

/*Berechnung der Temperatur ==============================================*/

temp = 0;


for (int i = 0; i < 5; i++) { //Die nächste Anweisung wird 5 Mal ausgeführt

temp += (analogRead(pin_temp) / 9.31); // Berechnet die Temperatur und die Summe in der Variablen
'temp'


}

temp /= 5; //Berechnet den mathematischen Durchschnitt der Temperaturwerte

/*=====================================================================*/

/*Ich sehe die Temperatur auf dem LCD-Display
==========================================*/

lcd.setCursor(0, 1); //lcd.print(temp); Bewege den Cursor über die erste Spalte und die zweite Zeile
lcd.print(temp);

// Form auf LCD-Anzeige Temperatur


lcd.print(" C"); // Ein Leerzeichen und die Schriftart 'C' auf der Anzeige formen

/*=======================================*/


delay(1000); //Verzögern um eine Sekunde (kann geändert werden)
```

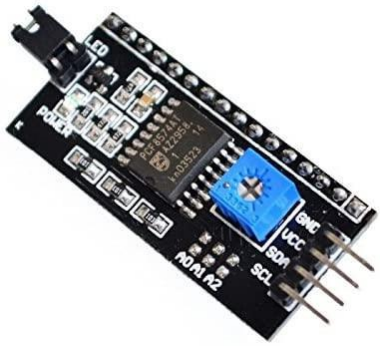
}

I2C-LCD mit Arduino verbinden

Wenn man ein LCD-Display mit dem Arduino verbinden will, muss man eine Menge Pins am Arduino verbrauchen. Selbst im 4-Bit-Modus benötigt der Arduino immer noch insgesamt sieben Anschlüsse, also die Hälfte der verfügbaren digitalen I/O-Pins.

Die Lösung ist die Verwendung eines LCD-Displays, das mit dem I2C-Protokoll arbeitet. Es verbraucht nur zwei E/A-Pins, die nicht einmal Teil eines digitalen E/A-Pin-Satzes sein müssen und auch mit anderen I2C-Geräten geteilt werden können.

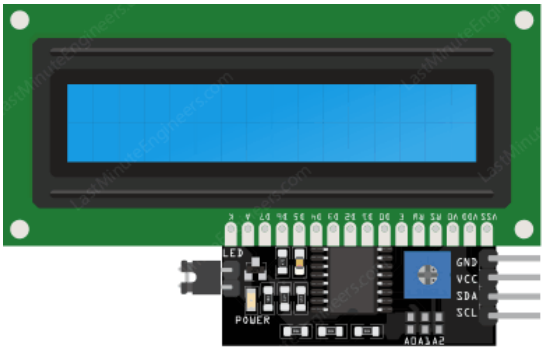
Die am weitesten verbreitete I2C-LCD-Anzeige besteht aus einer HD44780-basierten Zeichen-LCD-Anzeige und einem I2C-LCD-Adapter.



Der wichtigste Teil des Adapters ist der 8-Bit I/O Expander Chip - PCF8574. Dieser Chip wandelt die I2C-Daten von einem Arduino in die vom LCD-Display benötigten parallelen Daten um. Die Platine wird auch mit einem kleinen Potentiometer geliefert, mit dem der Kontrast des Displays fein eingestellt werden kann. Wenn Sie mehr als ein Gerät auf demselben I2C-Bus verwenden, müssen Sie möglicherweise eine

andere I2C-Adresse für das Board einstellen, damit es nicht mit einem anderen I2C-Gerät in Konflikt gerät. Aus diesem Grund hat die Platine drei Lötbrücken (A0, A1 und A2).

Ein I2C-LCD hat nur 4 Pins, die es mit dem Arduino verbinden.



Die Pinbelegung ist wie folgt:

- GND: ist ein Masse-Pin und sollte mit der Masse des Arduino verbunden werden;

- VCC: versorgt das Modul und das LCD mit Strom. Schließen Sie es an den 5V-Ausgang des Arduino oder eine separate Stromversorgung an;

- SDA: ist ein serieller Datenpin. Diese Leitung wird sowohl zum Senden als auch zum Empfangen verwendet. Schließen Sie ihn an den SDA-Pin des Arduino an;

- SCL: ist ein Pin für den seriellen Takt. Dies ist ein Taktsignal, das vom Bus-Master-Gerät geliefert wird. Schließen Sie es an den SCL-Pin des Arduino an.

Auf den Arduino-Boards mit dem R3-Layout befinden sich SDA (Datenleitung) und SCL (Taktleitung) an den Stiftleisten in der Nähe des AREF-Pins. Sie werden auch als A5 (SCL) und A4 (SDA) bezeichnet. Schauen Sie in der folgenden Tabelle nach, um die richtigen Pins je nach Arduino-Modell zu identifizieren.

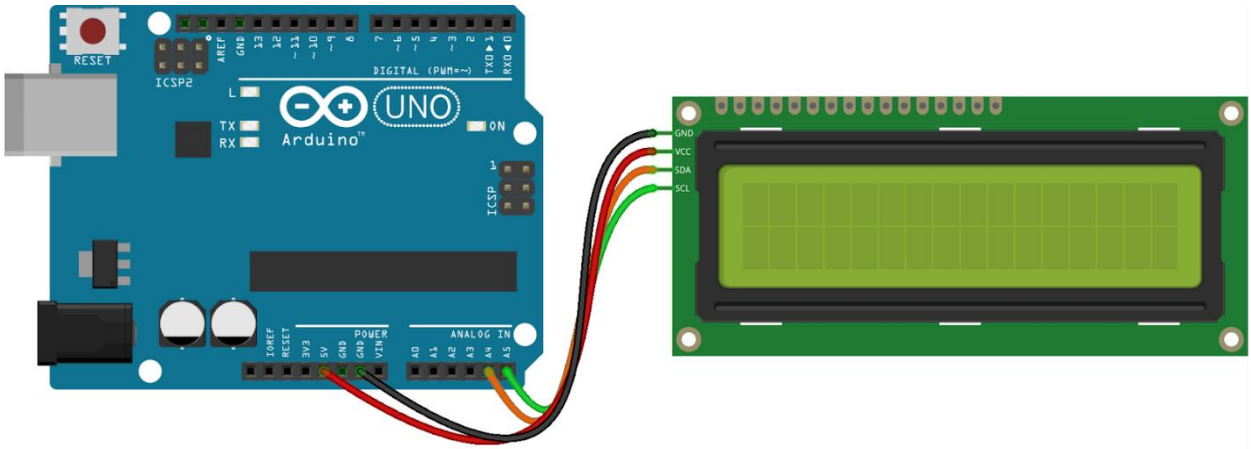
	SCL	SDA
Arduino Uno	A5	A4
Arduino Nano	A5	A4
Arduino Mega	21	20
Leonardo/Micro	3	2

Kreislauf 3:

Titel der Schaltung: "Drucken einer Nachricht auf dem I2C-LCD"

Erläuterung der Schaltung: Es ist möglich, ein LCD-Display mit nur 4 Pins an den Mikrocontroller anzuschließen und die gewünschten Nachrichten auf den Bildschirm zu drucken.

Hinweis: Sie müssen eine Bibliothek namens LiquidCrystal_I2C installieren. Diese Bibliothek ist eine verbesserte Version der LiquidCrystal-Bibliothek, die mit der Arduino IDE mitgeliefert wird.




```
/* Drucken einer Nachricht auf einem I2C-Display */

#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x3F,16,2); // Setzt die LCD-Adresse auf 0x3F für ein 16 Zeichen und 2 Zeilen Display

void setup() {

  lcd.init();

  lcd.clear();

  lcd.backlight(); // Sicherstellen, dass die Hintergrundbeleuchtung eingeschaltet ist

  // Drucken Sie eine Meldung auf beiden Zeilen des LCD-Displays.

  lcd.setCursor(2,0); //Setzen des Cursors auf Zeichen 2 in Zeile 0

  lcd.print("Hallo Welt!");

  lcd.setCursor(2,1); //Marke auf Zeichen 2 in Zeile 1 setzen

  lcd.print("mit I2C-Protokoll!");

}
```

```
void loop() {

}
```

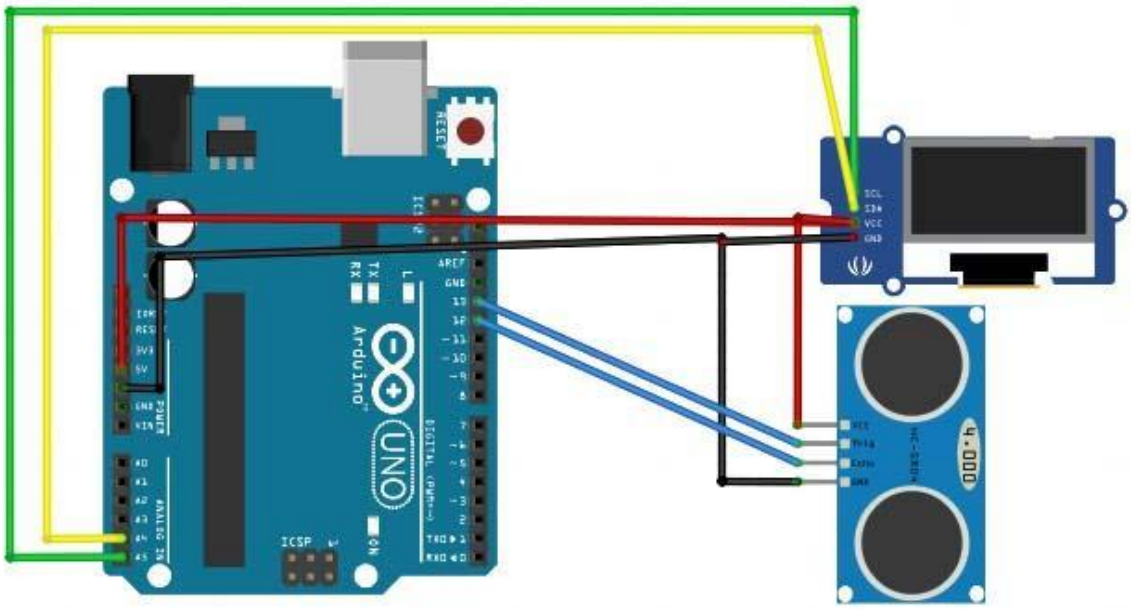
OLED-Grafik-Display-Modul mit Arduino verbinden

Eine weitere Möglichkeit, das I2C-Protokoll zu nutzen, ist die Wahl eines OLED-Displays (Organic Light-Emitting Diode). Sie sind superleicht und dünn und erzeugen ein helleres und schärferes Bild.

Ein OLED-Display funktioniert ohne Hintergrundbeleuchtung. Deshalb hat das Display einen so hohen Kontrast, einen extrem weiten Betrachtungswinkel und kann tiefe Schwarzwerte anzeigen. Durch das Fehlen der Hintergrundbeleuchtung wird der für den Betrieb des OLED-Displays erforderliche Strom erheblich reduziert.

Wie in der Abbildung zu sehen ist, handelt es sich bei der Pinbelegung um die klassische vierpolige I2C-Schnittstelle, die bereits oben beschrieben wurde.

Die Arduino-Gemeinschaft hat bereits einige Bibliotheken entwickelt, um mit diesen OLED-Displays umzugehen, wie z. B. die SSD1306-Bibliothek von Adafruit. Um die Bibliothek zu installieren, navigieren Sie zu Sketch > Include Library > Manage Libraries... Warten Sie, bis der Library Manager den Bibliotheksindex heruntergeladen und die Liste der installierten Bibliotheken aktualisiert hat.



Kreislauf 4:

Titel der Schaltung: "Abstandssensor"

Schaltung Erläuterung: Die Entfernung wird mit dem HC-SR04 Ultraschallsensor gemessen und auf einem OLED-Display angezeigt.

Hinweis: Es gibt nur vier Pins, um die Sie sich beim HC-SR04 kümmern müssen: VCC (Power), Trig

(Trigger), Echo (Receive) und GND (Ground).

/* Abstandssensor */

#include <SPI.h> // Diese Bibliothek ermöglicht die Kommunikation mit SPI-Geräten, wobei der Arduino als Master-Gerät fungiert.

#include <Wire.h> // Diese Bibliothek ermöglicht die Kommunikation mit I2C / TWI-Geräten.

#include <Adafruit_GFX.h> // die Bibliotheken für das OLED-Display

#include <Adafruit_SSD1306.h>

#define CommonSenseMetricSystem

#define trigPin 13 // definiere die Pins des Sensors

#define echoPin 12

void setup() {

Serial.begin (9600);

pinMode(trigPin, OUTPUT);

pinMode(echoPin, INPUT);

display.begin(SSD1306_SWITCHCAPVCC, 0x3C); //initialisieren mit der I2C-Adresse 0x3C (128x64)

display.clearDisplay();

}

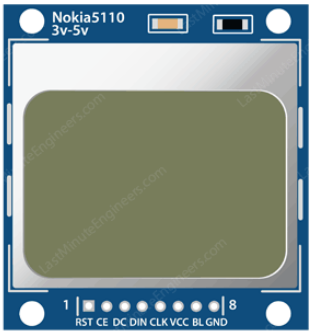
void loop() {

```
long Dauer, Abstand;


digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);


Dauer = pulseIn(echoPin, HIGH);
Abstand = (Dauer/2) / 29,1;
display.setCursor(22,20); //gekühlte Anzeige
display.setTextSize(3); //Größe des Textes
display.setTextColor(WHITE); //wenn man schwarz schreibt, löscht es alles
display.println(distance); //Drucke unsere Variable
display.setCursor(85,20);
display.setTextSize(3);


 display.println("cm");


display.display();


delay(500);
display.clearDisplay();
```

```
Serial.println(distance);//debug



}
```

Verbindung des Nokia 5110 Grafik-LCD-Displays mit Arduino

Sie können Arduino mit kleinen LCDs verbinden, ähnlich denen, die Nokia in seinen 3310- und 5110-Handys verwendet. Diese Displays sind klein (nur etwa 1,5”), preiswert, einfach zu bedienen, ziemlich stromsparend (nur 6 bis 7 mA) und können sowohl Text als auch Bitmaps anzeigen.



Es handelt sich um Grafikdisplays mit 84×48 Pixeln. Sie werden über eine serielle Busschnittstelle, ähnlich wie SPI, an Mikrocontroller angeschlossen. Das LCD verfügt auch über eine Hintergrundbeleuchtung in verschiedenen Farben wie Rot, Grün, Blau und Weiß. Bei der Hintergrundbeleuchtung handelt es sich um vier LEDs, die an den Rändern des Displays angebracht sind.

Es hat 8 Pins, die es mit dem Arduino verbinden, die Pinbelegung ist wie folgt:

- RST: Setzt das Display zurück. Es handelt sich um einen aktiven Low-Pin, d.h. Sie können das Display zurücksetzen, indem Sie ihn auf Low ziehen. Sie können diesen Pin auch mit dem Arduino-Reset verbinden, so dass er den Bildschirm automatisch zurücksetzt;

- CE(Chip Enable): wird verwendet, um eines der vielen angeschlossenen Geräte auszuwählen, die sich denselben SPI-Bus teilen;

- D/C(Data/Command): Dieser Pin teilt dem Display mit, ob die empfangenen Daten ein Befehl oder anzeigbare Daten sind;

- DIN: ist ein serieller Datenpin für die SPI-Schnittstelle;

- CLK: ist ein serieller Clock-Pin für die SPI-Schnittstelle;

- VCC: versorgt das LCD mit Strom, den wir an den 3,3-Volt-Pin des Arduino anschließen;

- BL(Backlight): steuert die Hintergrundbeleuchtung des Displays. Um die Helligkeit zu steuern, können Sie ein Potentiometer hinzufügen oder diesen Pin mit einem beliebigen PWM-fähigen Arduino-Pin verbinden;

- GND: ist ein Masse-Pin und sollte mit der Masse des Arduino verbunden werden.

Sie können die Datenübertragungspins an jeden beliebigen digitalen I/O-Pin anschließen. Das LCD hat 3-V-Kommunikationspegel, daher können wir diese Pins nicht direkt mit dem Arduino verbinden. Eine Möglichkeit besteht darin, jedem Datenübertragungs-Pin Widerstände zuzuordnen. Fügen Sie einfach 10kΩ-Widerstände zwischen den CLK-, DIN-, D/C- und RST-Pins und einen 1kΩ-Widerstand zwischen CE ein. Der Pin für die Hintergrundbeleuchtung (BL) ist über einen 330Ω-Strombegrenzungswiderstand an 3,3 V angeschlossen. Sie können ein Potentiometer hinzufügen oder diesen Pin mit einem beliebigen PWM-fähigen Arduino-Pin verbinden, wenn Sie die Helligkeit steuern möchten.

Die Arduino-Gemeinschaft hat bereits einige Bibliotheken entwickelt, um mit diesen NOKIA-Displays umzugehen, wie z. B. die PCD8544 Nokia 5110 LCD-Bibliothek von Adafruit. Um die Bibliothek zu

installieren, navigieren Sie zu Sketch > Include Library > Manage Libraries... Warten Sie, bis der Library Manager den Bibliotheksindex heruntergeladen und die Liste der installierten Bibliotheken aktualisiert hat.

Schaltung 5:

Schaltkreistitel: "Textdrehung"

Erläuterung des Schaltkreises: Sie können den Inhalt der Anzeige drehen, indem Sie die Funktion setRotation() aufrufen. Damit können Sie Ihr Display im Hochformat betrachten oder es auf den Kopf stellen.

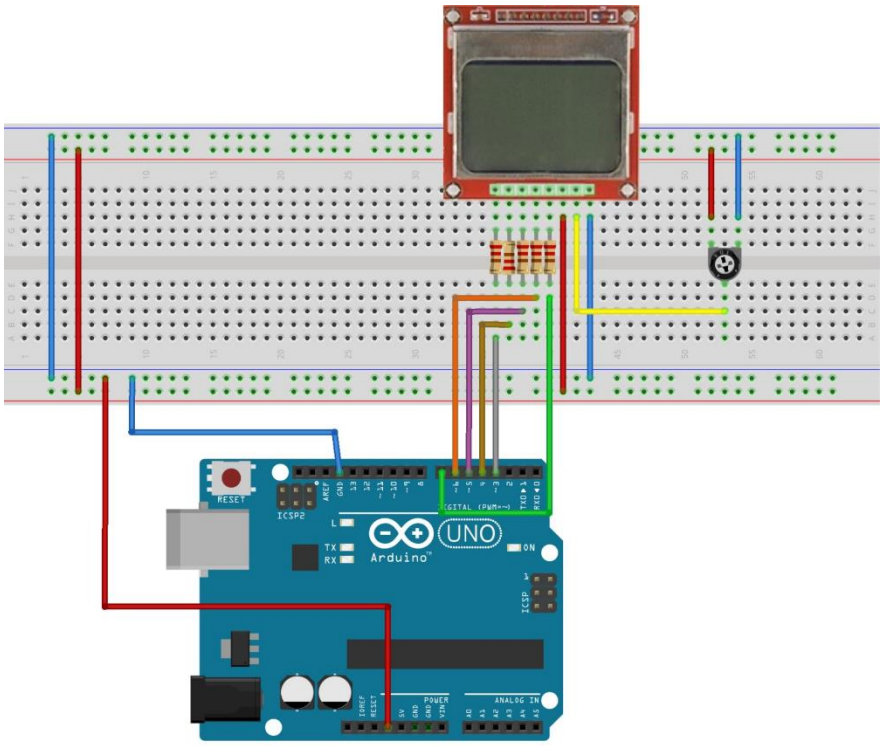
Hinweis: Die Funktion akzeptiert nur einen Parameter, der 4 kardinalen Drehungen entspricht. Dieser Wert kann eine beliebige nicht-negative ganze Zahl sein, beginnend mit 0. Jedes Mal, wenn Sie den Wert erhöhen, wird der Inhalt des Displays um 90 Grad gegen den Uhrzeigersinn gedreht. Zum Beispiel:

- 0 - Behält den Bildschirm in der Standard-Querformat-Ausrichtung bei.

- 1 - Dreht den Bildschirm um 90° nach rechts.

- 2 - Dreht den Bildschirm auf den Kopf.

- 3 - Dreht den Bildschirm um 90° nach links.



/* Text Rotation */

#include <SPI.h> // diese Bibliothek ermöglicht die Kommunikation mit SPI-Geräten, wobei der Arduino das Master-Gerät ist.

#include <Adafruit_GFX.h> // die Bibliotheken für das OLED-Display

#include <Adafruit_PCD8544.h>

// LCD-Objekt für Software deklarieren SPIAdafruit_PCD8544(CLK,DIN,D/C,CE,RST);

Adafruit_PCD8544 display = Adafruit_PCD8544(7, 6, 5, 4, 3);

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


        //Initialisierung des Displays
        display.begin();


        display.setContrast(57); // Sie können den Kontrast ändern, um die Anzeige anzupassen


        display.clearDisplay(); // Löscht den Puffer.


// Textdrehung
while(1)
{
        display.clearDisplay();
        display.setRotation(rotatetext);
        display.setTextSize(1);
        display.setTextColor(BLACK);
        display.setCursor(0,0);
        display.println("Textdrehung");
```

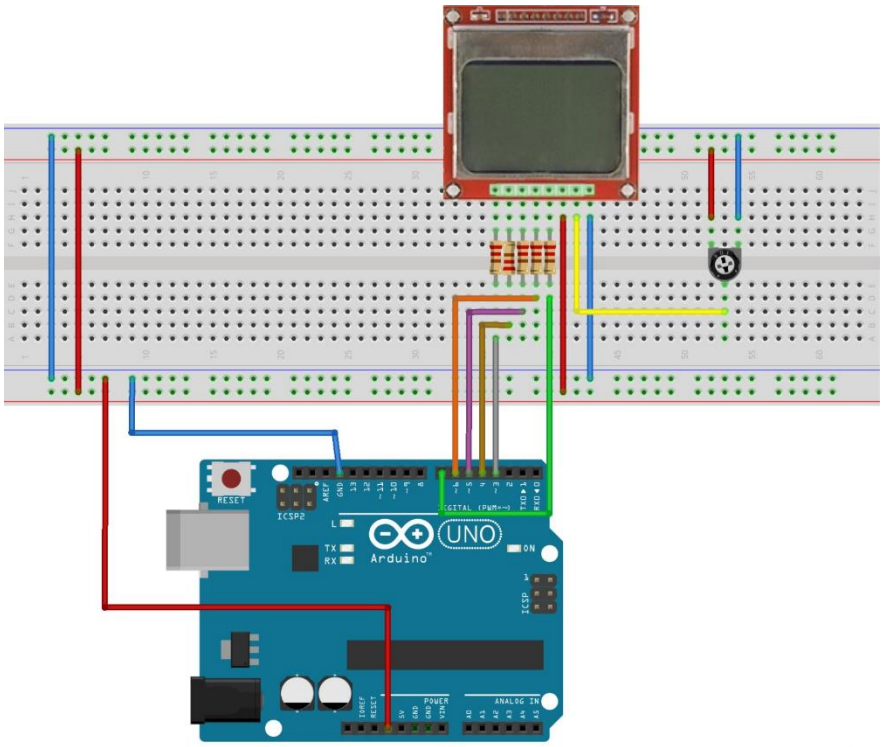
```
        display.display();

        delay(1000);

        display.clearDisplay();

        rotatetext++;

}

}


void loop() {}
```

Kreislauf 6:

Titel des Schaltkreises: "Marilyn Bmp Image"

Erläuterung der Schaltung: wie man Bitmap-Bilder auf dem LCD-Display des Nokia 5110 zeichnet. In diesem Beispiel ist ein Porträt von Marilyn Monroe zu sehen.

Hinweis: Die Bildschirmauflösung des Nokia 5110 LCD-Displays beträgt 84×48 Pixel, so dass Bilder, die größer sind als diese, nicht korrekt angezeigt werden. Um ein Bitmap-Bild auf dem Nokia 5110 LCD-Display anzuzeigen, müssen wir die Funktion drawBitmap() aufrufen. Sie benötigt sechs Parameter: X-Koordinate der linken oberen Ecke, Y-Koordinate der linken oberen Ecke, Byte-Array der monochromen Bitmap, Breite der Bitmap in Pixel, Höhe der Bitmap in Pixel und Farbe. In unserem Beispiel ist das Bitmap-Bild 84×48 groß. Die X- und Y-Koordinaten werden also auf 0 gesetzt, während Breite und Höhe auf 84 und 48 gesetzt werden.



```
/* Marilyn Bmp Image */


#include <SPI.h>

#include <Adafruit_GFX.h>

#include <Adafruit_PCD8544.h>


Adafruit_PCD8544 display = Adafruit_PCD8544(7, 6, 5, 4, 3);


// 'Marilyn Monroe 84x48', 84x48px
```

```
const unsigned char MarilynMonroe [] PROGMEM = {
        0x00, 0x00, 0x7f, 0x00, 0x02, 0xfe, 0xf8, 0x00, 0x00, 0x00, 0x00, 0xbe, 0x00,
        0x00, 0x1f, 0xe0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x20, 0x00, 0x00, 0x00, 0x3f, 0x80, 0x00,
0x00,
        0x00, 0x00, 0x00, 0xf0, 0x00, 0x00, 0x1f, 0xe1, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0xc0,
        0x00, 0x00, 0x0f, 0xf1, 0xc0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0e, 0xd8, 0xe0,
        0x00, 0x00, 0x00, 0x00, 0x1f, 0x80, 0x00, 0x07, 0xe0, 0x70, 0x00, 0x00, 0x00, 0x00, 0x03,
        0x3f, 0xe0, 0x00, 0x07, 0xf0, 0x78, 0x00, 0x00, 0x00, 0x01, 0xe0, 0x70, 0x00, 0x0f, 0xee,
        0x7c, 0x00, 0x00, 0x00, 0x03, 0xc0, 0x00, 0x00, 0x0f, 0xf7, 0x1c, 0x00, 0x00, 0x00, 0x00, 0x00,
        0x07, 0x80, 0x00, 0x0f, 0xc7, 0xf3, 0x1e, 0x00, 0x00, 0x00, 0x07, 0xc0, 0x00, 0x0f, 0xf3,
        0xdf, 0x7f, 0x80, 0x00, 0x00, 0x00, 0x07, 0xfe, 0x00, 0x08, 0x7d, 0xef, 0xff, 0xc0, 0x00, 0x00,
        0x00, 0x7f, 0xff, 0x80, 0x30, 0x0f, 0xfc, 0xe0, 0xc0, 0x00, 0x00, 0x01, 0x9e, 0x73, 0xc0, 0xe0,
        0x07, 0xf8, 0xc1, 0xc0, 0x00, 0x00, 0x03, 0xfc, 0x00, 0x01, 0xc0, 0x0f, 0xfd, 0xe1, 0x80, 0x00,
        0x00, 0x03, 0xf8, 0x00, 0x01, 0x9c, 0x0f, 0xff, 0xc1, 0xc0, 0x00, 0x00, 0x02, 0xc0, 0x00, 0x01,
        0x9f, 0xbf, 0xfe, 0x01, 0x40, 0x00, 0x00, 0x02, 0x60, 0x00, 0x03, 0x07, 0xef, 0xff, 0x01, 0x40,
        0x00, 0x00, 0x00, 0x60, 0x00, 0x07, 0x01, 0xf7, 0xff, 0x80, 0xc0, 0x00, 0x00, 0x00, 0x50, 0x01,
        0xdf, 0x00, 0x7f, 0xff, 0x1c, 0x80, 0x00, 0x00, 0x00, 0x40, 0x01, 0xff, 0x00, 0x1f, 0xff, 0x1e,
        0xe0, 0x00, 0x00, 0x02, 0x08, 0x00, 0x3f, 0x80, 0x07, 0xef, 0x03, 0xe0, 0x00, 0x00, 0x06, 0x08,
        0x00, 0x03, 0xc0, 0x07, 0xdf, 0x07, 0xc0, 0x00, 0x06, 0x08, 0x0f, 0x81, 0x80, 0x1f, 0xdf,
        0x1f, 0x80, 0x00, 0x00, 0x03, 0x08, 0x1f, 0x98, 0x00, 0x3f, 0xfe, 0x19, 0x80, 0x00, 0x00, 0x18,
        0x08, 0x3f, 0xfe, 0x00, 0x7f, 0xfe, 0x3f, 0x00, 0x00, 0x08, 0x08, 0x30, 0x3f, 0x00, 0xff,
        0xff, 0x3f, 0x00, 0x00, 0x01, 0xe0, 0x76, 0x0f, 0x89, 0xff, 0xff, 0x9f, 0x00, 0x00, 0x00, 0x00,
        0x03, 0xe0, 0x7f, 0xc3, 0x81, 0xff, 0xfe, 0x9f, 0x80, 0x00, 0x00, 0x03, 0xf0, 0x7f, 0xf3, 0xc3,
```

```
        0xff, 0xfe, 0x1f, 0x00, 0x00, 0x00, 0x03, 0xf0, 0x7f, 0xfd, 0xc3, 0xff, 0xfe, 0x5e, 0x00, 0x00, 0x00,
        0x00, 0x03, 0xf0, 0x7f, 0xff, 0xc3, 0xff, 0xf3, 0x1e, 0x00, 0x00, 0x03, 0xf0, 0x71, 0xff,
        0x87, 0xff, 0xe3, 0xff, 0x00, 0x00, 0x07, 0xf0, 0x7c, 0x3f, 0x87, 0xff, 0xe3, 0xfe, 0x00,
        0x00, 0x00, 0x0f, 0xf0, 0x3c, 0xff, 0x05, 0xff, 0xf3, 0xfc, 0x00, 0x00, 0x0f, 0xf0, 0x0f,
        0xfe, 0x09, 0xff, 0xf7, 0xfc, 0x00, 0x00, 0x00, 0x08, 0xf8, 0x01, 0xfc, 0x19, 0xff, 0xff, 0xf8,
        0x00, 0x00, 0x00, 0x0c, 0x78, 0x00, 0x00, 0x13, 0xff, 0xff, 0xf8, 0x00, 0x00, 0x00, 0x00, 0x0e, 0x78,
        0x00, 0x00, 0x23, 0xff, 0xff, 0xf0, 0x00, 0x00, 0x00, 0x00, 0x0e, 0xf8, 0x00, 0x00, 0x47, 0xff, 0xff,
        0xf0, 0x00, 0x00, 0x0c, 0xfa, 0x00, 0x01, 0x8f, 0xff, 0xff, 0xe0, 0x00, 0x00, 0x00, 0x00, 0x08,
        0x7b, 0x00, 0x03, 0x3f, 0xff, 0xff, 0xe0, 0x00, 0x00, 0x00, 0x0c, 0x3b, 0xf8, 0x0f, 0xff, 0xff,
        0xff, 0xe0, 0x00, 0x00, 0x0f, 0xbb, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff0, 0x00, 0x00, 0x00,
        0x07, 0xfb, 0xff, 0xff, 0xff, 0xff, 0xff0, 0x00, 0x00, 0x00, 0x71, 0xff, 0xff, 0xff, 0xff,
        0xff, 0xff, 0xe0, 0x00, 0x00, 0x00, 0x00, 0x41, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xe0, 0x00, 0x00, 0x00
};


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

        display.begin();

        display.setContrast(57);

        display.clearDisplay();
```

```
        // Bitmap anzeigen
        display.drawBitmap(0, 0, MarilynMonroe, 84, 48, BLACK);
        display.display();


        // Anzeige invertieren
        //display.invertDisplay(1);
}

void loop() {}
```

Co-funded by the Erasmus+ Programme of the European Union

Erasmus+ KA-202

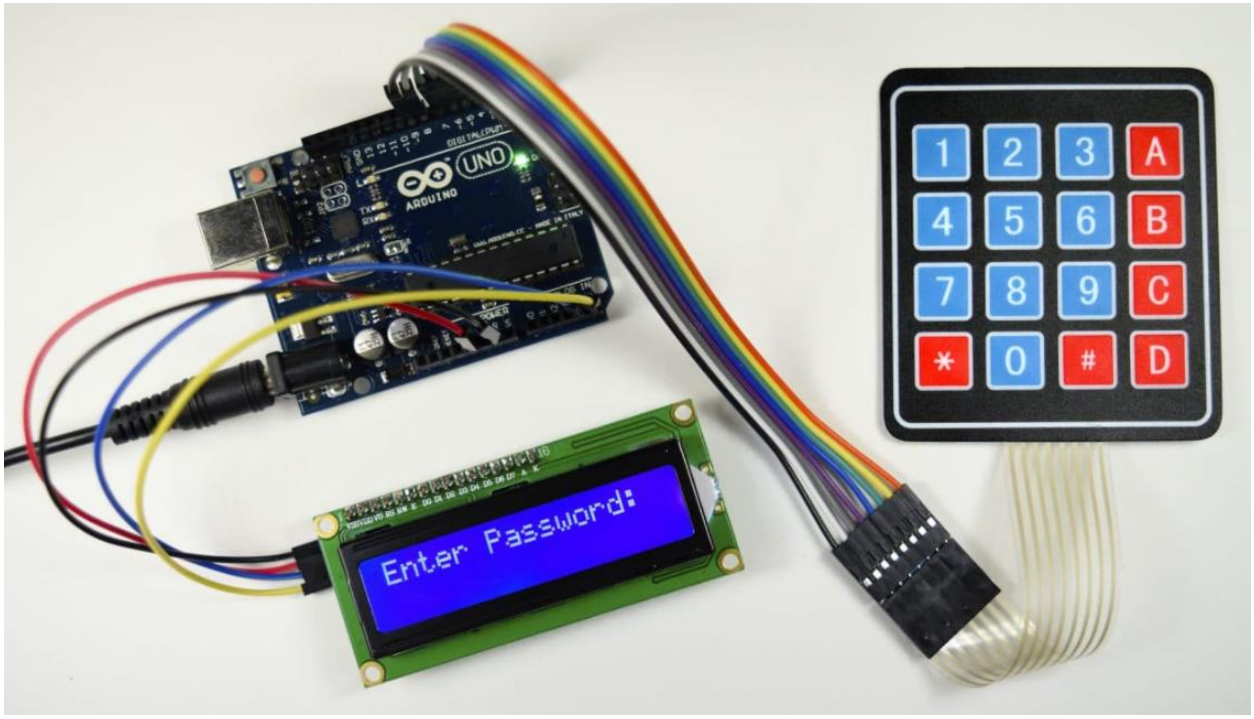
Strategisches Partnerschaftsprojekt für berufliche Bildung und Ausbildung

Projekttitel: "Lehren und Lernen mit Arduinos in der Berufsbildung"

Projekt-Akronym: "ARDUinVET"

Projekt Nr.: “2020-1-TR01-KA202-093762”

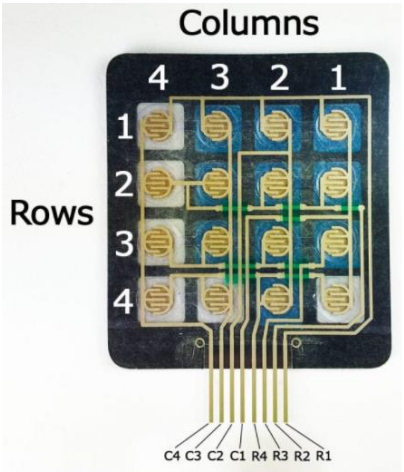
Tastaturmodul und Schulungskit



Was ist ein Keypad?

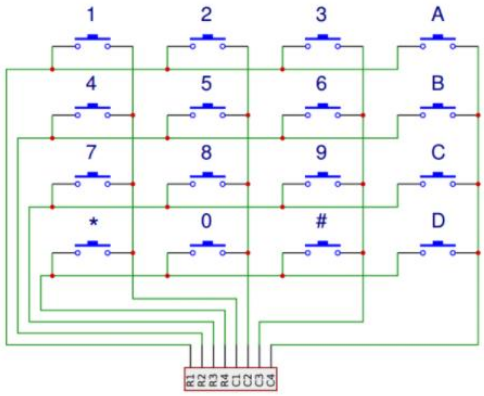
Ein Keypad ist ein System von Tasten, die in einer Matrix angeordnet sind und als Schaltvorrichtung dienen, um eine Verbindung zwischen einer Zeile und einer Spalte herzustellen.

Wir werden eine Folientastatur mit einer 4X4-Matrix verwenden, weil sie dünn ist und eine Klebeunterstützung hat, so dass sie auf die meisten flachen Oberflächen geklebt werden kann. Unter jeder Taste befindet sich ein Membranschalter. Jeder Schalter in einer Reihe ist mit den anderen Schaltern in der Reihe durch eine Leiterbahn unterhalb des Pads verbunden. Jeder Schalter in einer Spalte ist auf die gleiche Weise verbunden - eine Seite des Schalters ist mit allen anderen Schaltern in dieser Spalte durch eine Leiterbahn verbunden. Jede Reihe und Spalte ist mit einem einzigen Stift verbunden, so dass insgesamt 8 Stifte auf einer 4X4-Tastatur vorhanden sind.	

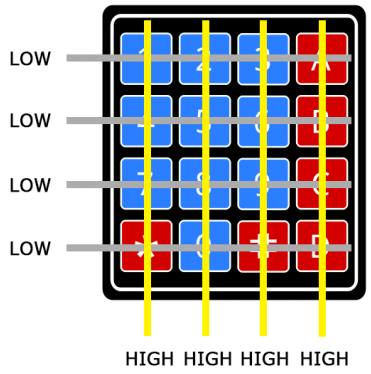
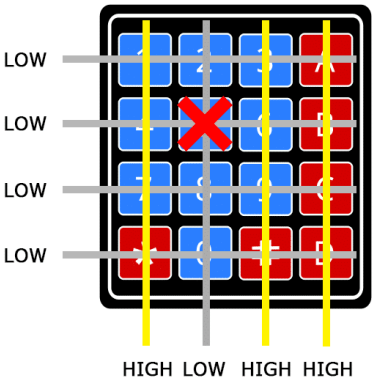
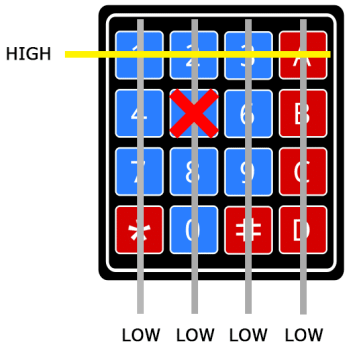


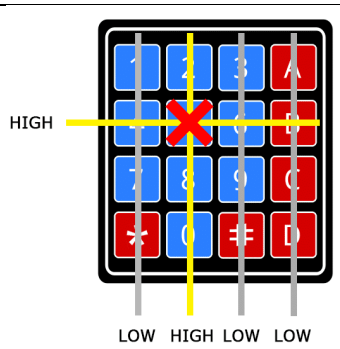
Durch Drücken einer Taste wird der Schalter zwischen einer Spalten- und einer Zeilenleitung geschlossen, so dass Strom zwischen einem Spaltenstift und einem Zeilenstift fließen kann.

Der Schaltplan für ein 4X4-Tastenfeld zeigt, wie die Zeilen und Spalten verbunden sind:



Der Arduino erkennt, welche Taste gedrückt wird, indem er den Zeilen- und Spaltenpin erkennt, der mit der Taste verbunden ist.This happens in four steps:

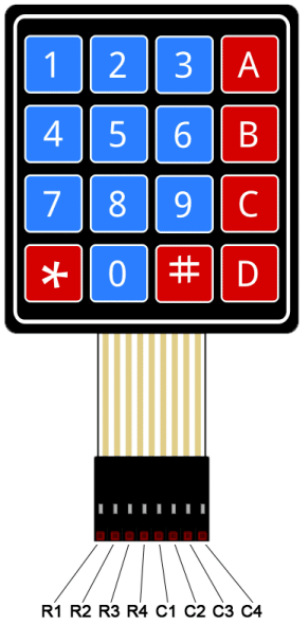
1. Erstens, wenn keine Tasten gedrückt werden, werden alle Spaltenstifte auf HIGH und alle Zeilenstifte auf LOW gehalten	
2. Wenn eine Taste gedrückt wird, wird der Spaltenstift auf LOW gezogen, da der Strom von der HIGH-Spalte zum LOW-Zeilenstift fließt:	
3. Der Arduino weiß jetzt, in welcher Spalte sich die Taste befindet, also muss er jetzt nur noch die Reihe finden, in der sich die Taste befindet. Dazu schaltet er jeden der Reihenpins auf HIGH und liest gleichzeitig alle Spaltenpins aus, um festzustellen, welcher Spaltenpin auf HIGH zurückkehrt	

4. Wenn der Spaltenpin wieder auf HIGH geht, hat der Arduino den Zeilenpin gefunden, der mit der Taste verbunden ist:	

Aus dem obigen Diagramm können Sie ersehen, dass die Kombination aus Zeile 2 und Spalte 2 nur bedeuten kann, dass die Taste mit der Nummer 5 gedrückt wurde.

DAS TASTENFELD AN DEN ARDUINO ANSCHLIESSEN

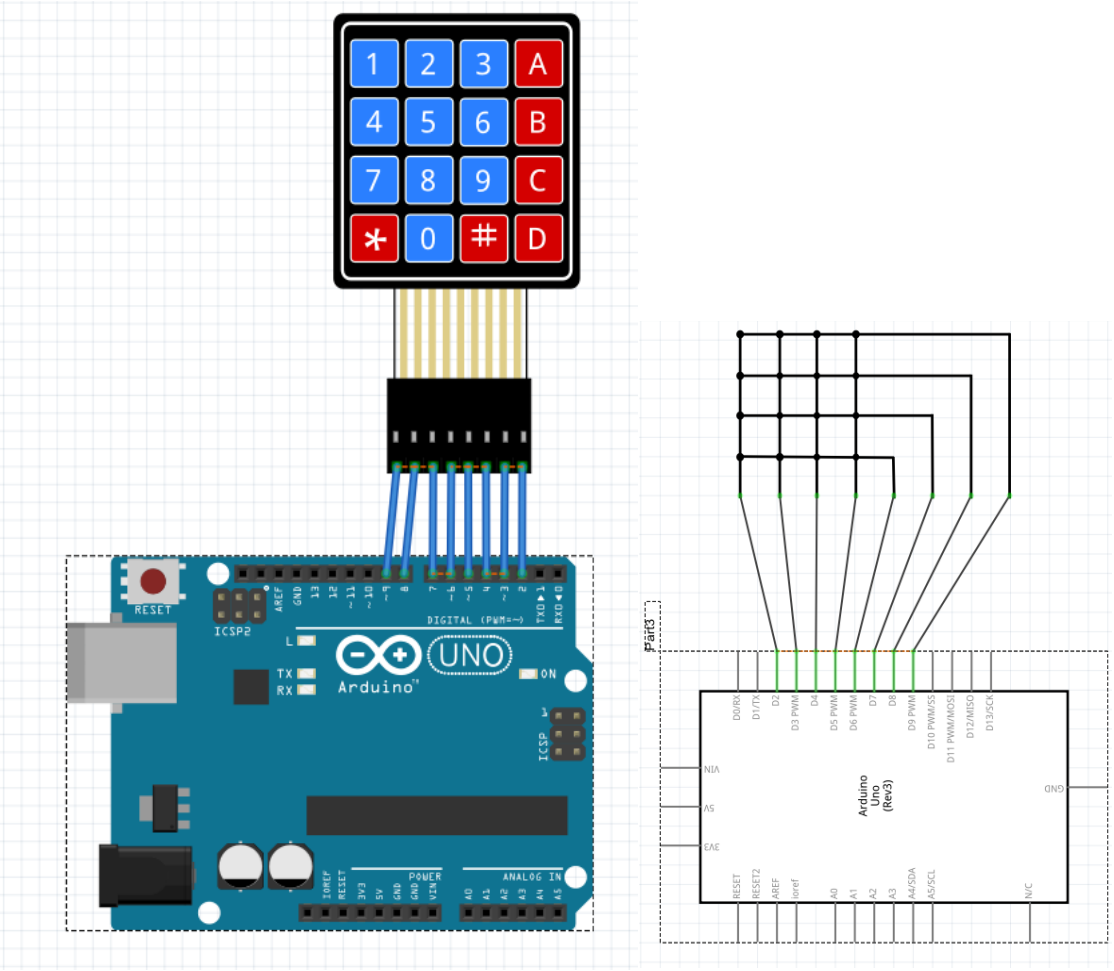
Die Pinbelegung der meisten Folientastaturen sieht wie folgt aus:



Schaltung 1

Titel der Schaltung: Serieller Monitor zur Anzeige der gedrückten Taste

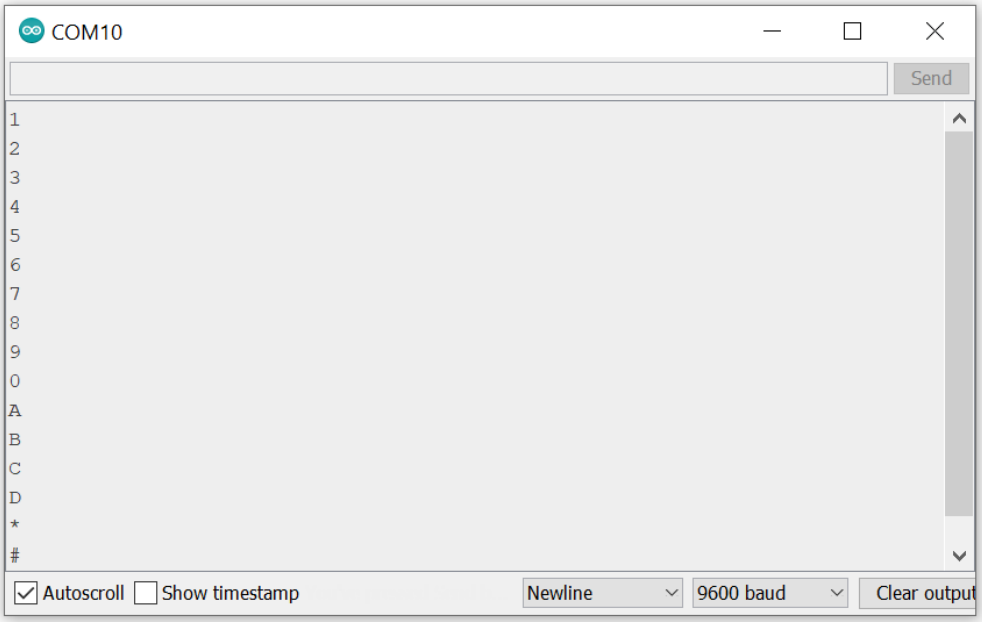
Beschreibung der Schaltung: Das Programm zeigt uns, wie wir jeden Tastendruck auf dem seriellen Monitor ausgeben können.



1-) Ansicht auf der Leiterplatte 2-) Schematische Darstellung

```
/* "Serieller Monitor zeigt die gedrückte Taste an" */
#include <Keypad.h>
const byte ROWS = 4;
const byte COLS = 4;
char hexaKeys[ROWS][COLS] = {
  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
  {'7', '8', '9', 'C'},
  {'*', '0', '#', 'D'}
};
byte rowPins[ROWS] = {9, 8, 7, 6};
byte colPins[COLS] = {5, 4, 3, 2};
```

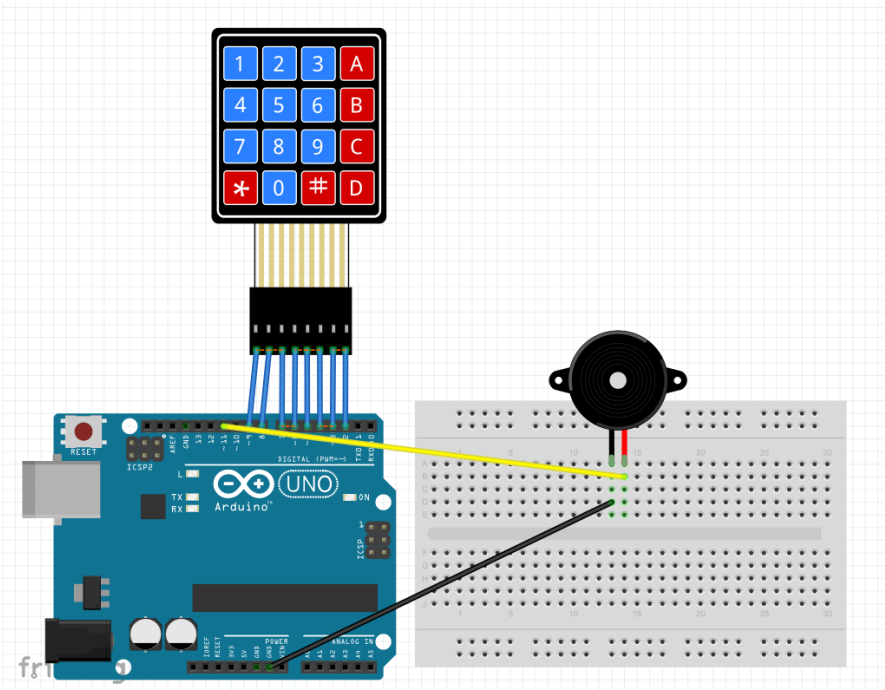
```
Keypad customKeypad = Keypad(makeKeymap(hexaKeys), rowPins, colPins, ROWS,
COLS);
void setup(){
  Serial.begin(9600);
}
void loop(){
  char customKey = customKeypad.getKey();
  if (customKey){
    Serial.println(customKey);
  }
}
```



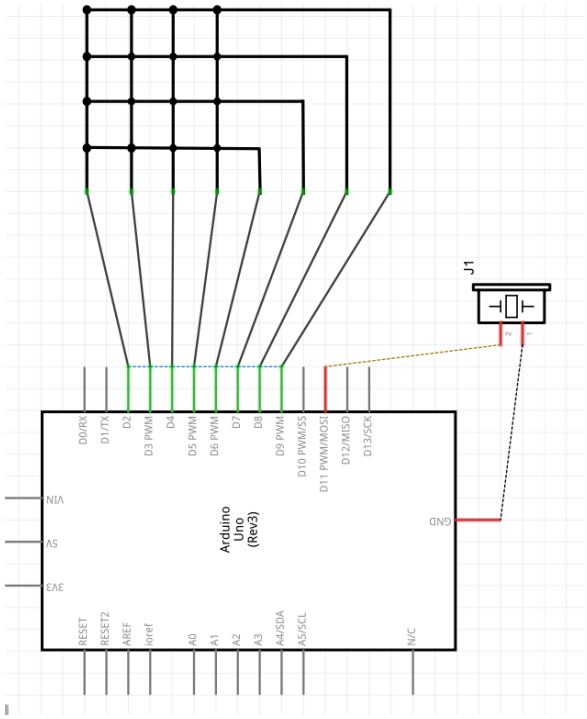
Schaltung 2

Titel der Schaltung: Arduino-Tastatur Beep

Beschreibung der Schaltung: Wenn eine Taste auf dem Tastenfeld gedrückt wird, ertönt der Piezo-Summer



1-) Ansicht auf der Leiterplatte



2-) Schematische Darstellung

/* "Arduino Keypad Beep" */

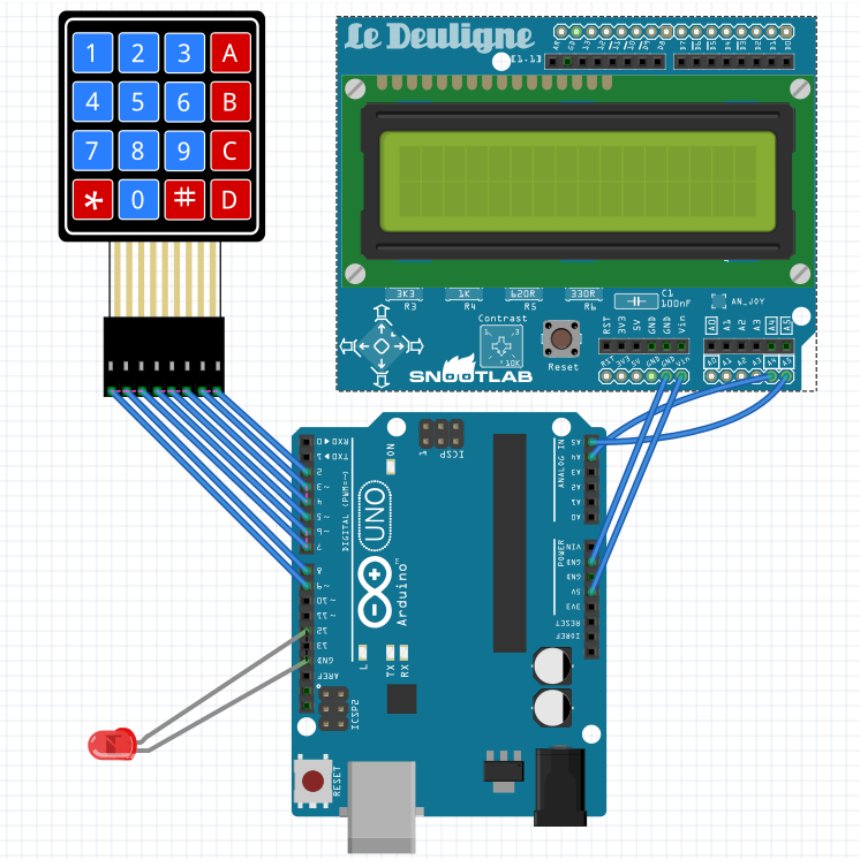
#include <Keypad.h>

```
#include <ezBuzzer.h>
const int BUZZER_PIN = 11;
const int ROW_NUM    = 4; // four rows
const int COLUMN_NUM = 4; // four columns
char keys[ROW_NUM][COLUMN_NUM] = {
  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
  {'7', '8', '9', 'C'},
  {'*', '0', '#', 'D'}
};
byte pin_rows[ROW_NUM] = {9, 8, 7, 6};      // connect to the row pinouts of the keypad
byte pin_column[COLUMN_NUM] = {5, 4, 3, 2}; // connect to the column pinouts of the
keypad
Keypad keypad = Keypad(makeKeymap(keys), pin_rows, pin_column, ROW_NUM,
COLUMN_NUM );
ezBuzzer buzzer(BUZZER_PIN); // create ezBuzzer object that attach to a pin;
void setup() {
  Serial.begin(9600);
}
void loop() {
  buzzer.loop(); // MUST call the buzzer.loop() function in loop()
  char key = keypad.getKey();
  if (key) {
    Serial.print(key); // prints key to serial monitor
    buzzer.beep(100);  // generates a 100ms beep
  }
}
```

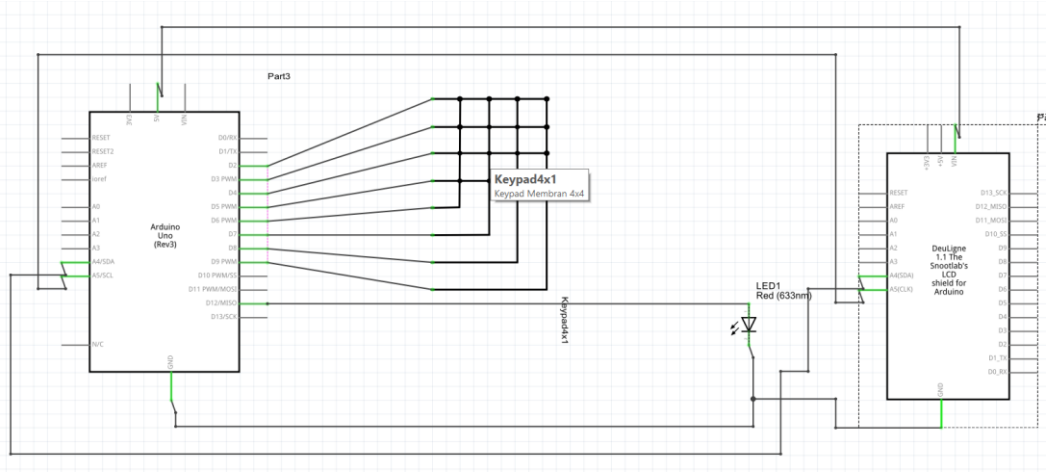
Schaltung 3

Titel der Schaltung: Arduino Entriegelungscode

Beschreibung der Schaltung: Programm, das ein Tastenfeld auf dem Arduino einrichtet. Eine LED leuchtet auf, wenn Sie den richtigen Code eingeben



1-) Ansicht auf der Leiterplatte



2-) Schematische Darstellung

/* Arduino Entriegelungscode */

#include <Keypad.h> //Libraries you can download them via Arduino IDE

#include <Wire.h>

```
#include <LCD.h>
#include <LiquidCrystal_I2C.h>
#define Solenoid 12          //Actually the Gate of the transistor that controls the solenoid
                             //in my case I use a simple LED
#include <liquidcrystal_i2c.h></liquidcrystal_i2c.h></lcd.h></wire.h></keypad.h>
#define I2C_ADDR 0x27 // LCD i2c Adress and pins
#define BACKLIGHT_PIN 3
#define En_pin 2
#define Rw_pin 1
#define Rs_pin 0
#define D4_pin 4
#define D5_pin 5
#define D6_pin 6
#define D7_pin 7
LiquidCrystal_I2C
lcd(I2C_ADDR,En_pin,Rw_pin,Rs_pin,D4_pin,D5_pin,D6_pin,D7_pin);
const byte numRows= 4; //number of rows on the keypad
const byte numCols= 4; //number of columns on the keypad
int code = 1234;  //here is the code
int tot,i1,i2,i3,i4;
char c1,c2,c3,c4;
//keymap defines the key pressed according to the row and columns just as appears on the
keypad char keymap[numRows][numCols]=
{
{'1', '2', '3', 'A'},
{'4', '5', '6', 'B'},
{'7', '8', '9', 'C'},
{'*', '0', '#', 'D'}
};
//Code that shows the keypad connections to the arduino terminals
byte rowPins[numRows] = {9,8,7,6}; //Rows 0 to 3
byte colPins[numCols]= {5,4,3,2}; //Columns 0 to 3
//initializes an instance of the Keypad class
```

```
Keypad myKeypad= Keypad(makeKeymap(keymap), rowPins, colPins, numRows,
numCols);
void setup()
        {
         lcd.begin (16,2);
         lcd.setBacklightPin(BACKLIGHT_PIN,POSITIVE);
         lcd.setBacklight(HIGH);
         lcd.home ();
         lcd.print("ROMANIA DoorLock");
         lcd.setCursor(9, 1);
         lcd.print("Standby");
          pinMode(Solenoid,OUTPUT);
         delay(2000);
        }
void loop()
{
    char keypressed = myKeypad.getKey();  //The getKey fucntion keeps the program
runing, as long you didn't press "*" the whole thing bellow wouldn't be triggered
      if (keypressed == '*')          // and you can use the rest of you're code simply
         {
          lcd.clear();
          lcd.setCursor(0, 0);
          lcd.print("Enter Code");              //when the "*" key is pressed you can enter
the passcode
              keypressed = myKeypad.waitForKey();  // here all programs are stopped until
you enter the four digits then it gets compared to the code above
              if (keypressed != NO_KEY)
               {
                c1 = keypressed;
                lcd.setCursor(0, 1);
                lcd.print("*");
                }
              keypressed = myKeypad.waitForKey();
```

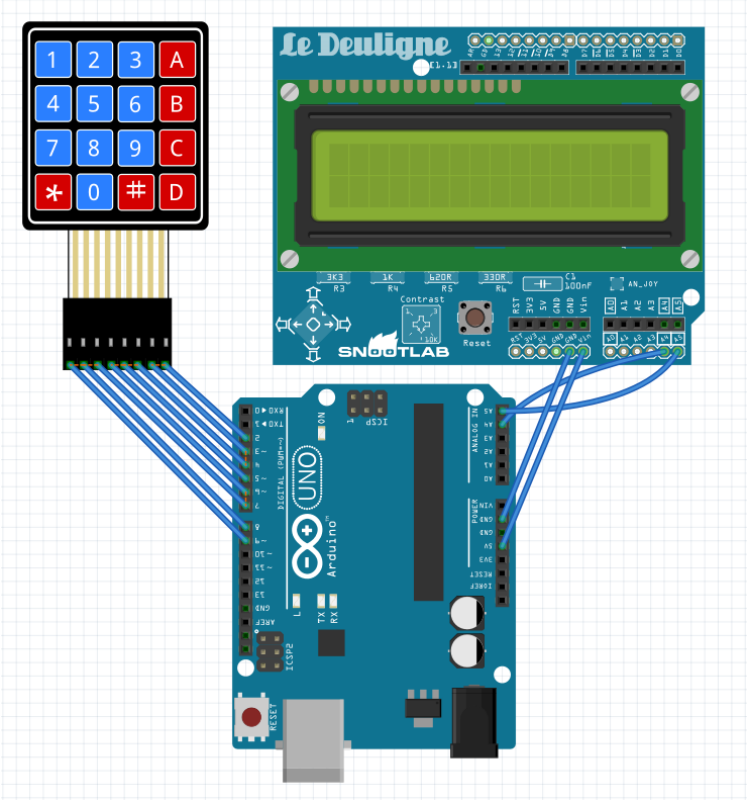
```
if (keypressed != NO_KEY)
  {
   c2 = keypressed;
   lcd.setCursor(1, 1);
   lcd.print("*");
   }
  keypressed = myKeypad.waitForKey();
if (keypressed != NO_KEY)
  {
   c3 = keypressed;
   lcd.setCursor(2, 1);
   lcd.print("*");
   }
  keypressed = myKeypad.waitForKey();
if (keypressed != NO_KEY)
  {
   c4 = keypressed;
   lcd.setCursor(3, 1);
   lcd.print("*");
   }
  i1=(c1-48)*1000;       //the keys pressed are stored into chars I convert them to
int then i did some multiplication to get the code as an int of xxxx
  i2=(c2-48)*100;
  i3=(c3-48)*10;
  i4=c4-48;
  tot=i1+i2+i3+i4;
 if (tot == code) //if the code is correct you trigger whatever you want here it just
print a message on the screen
{
 lcd.clear();
 lcd.setCursor(0, 0);
 lcd.print("Welcome");
 digitalWrite(Solenoid,HIGH);
```

```
 delay(3000);
 digitalWrite(Solenoid,LOW);
        lcd.setCursor(7, 1);
        lcd.print("ROMANIA DoorLock");

        delay(3000);
        lcd.clear();
        lcd.print("ROMANIA DoorLock");
        lcd.setCursor(9, 1);
        lcd.print("Standby");

       }
       else //if the code is wrong you get another thing
       {
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("WRONG CODE");
        delay(3000);
        lcd.clear();
        lcd.print("ROMANIA DoorLock");
        lcd.setCursor(9, 1);
        lcd.print("Standby");
        }
           }
}
```

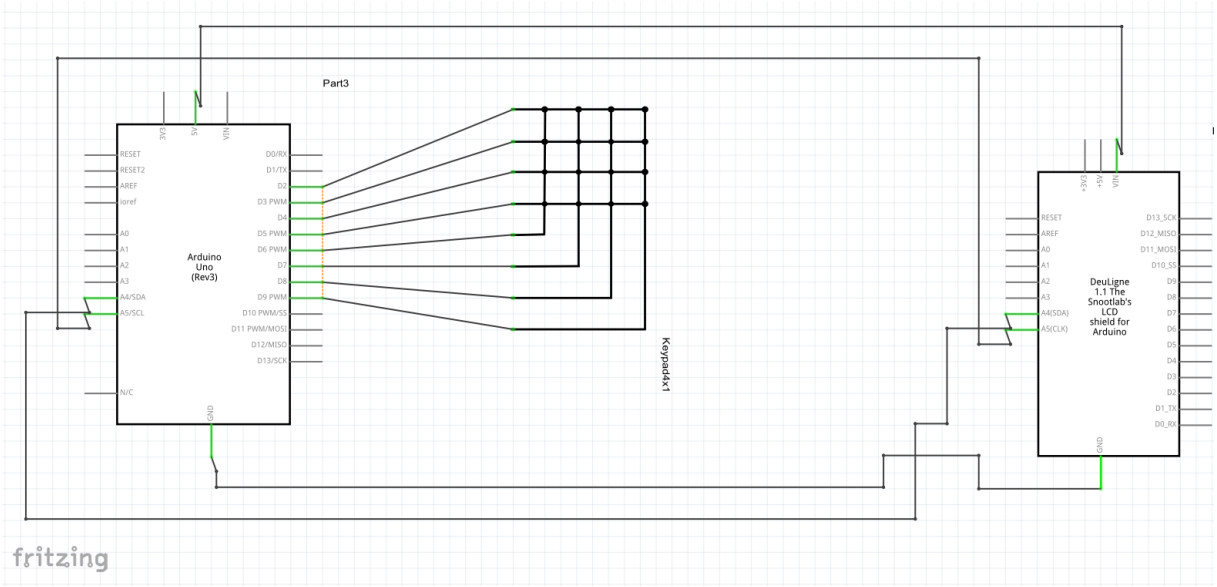
Schaltung 4

Titel der Schaltung: Arduino-Rechner

Beschreibung der Schaltung: Das Programm führt grundlegende mathematische Berechnungen durch



1-) Ansicht auf der Leiterplatte



2-) Schematische Darstellung

/* Arduino-Rechner */

#include <Keypad.h>

```
#include <EEPROM.h>
#include <LCD.h>
#include <LiquidCrystal_I2C.h>
#define I2C_ADDR 0x27          //I2C adress
#define BACKLIGHT_PIN 3        // Declaring LCD Pins
#define En_pin 2
#define Rw_pin 1
#define Rs_pin 0
#define D4_pin 4
#define D5_pin 5
#define D6_pin 6
#define D7_pin 7
const byte ROWS = 4; // Four rows
const byte COLS = 4; // Four columns
// Define the Keymap
char keys[ROWS][COLS] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};
byte rowPins[ROWS] = {9,8,7,6}; //Rows 0 to 3
byte colPins[COLS]= {5,4,3,2}; //Columns 0 to 3
LiquidCrystal_I2C
lcd(I2C_ADDR,En_pin,Rw_pin,Rs_pin,D4_pin,D5_pin,D6_pin,D7_pin);
Keypad kpd = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS ); //  Create
the Keypad
long Num1,Num2,Number;
 char key,action;
 boolean result = false;
 void setup()
        {
         lcd.begin (16,2);
```

```
        lcd.setBacklightPin(BACKLIGHT_PIN,POSITIVE);
          lcd.setBacklight(HIGH); //Lighting backlight
       lcd.print("Calculator Ready"); //Display a intro message
 lcd.setCursor(0, 1);   // set the cursor to column 0, line 1
 lcd.print("A=+ B=- C=* D=/"); //Display a intro message
  delay(6000); //Wait for display to show info
   lcd.clear(); //Then clean it
                //        for(i=0 ; i<sizeof(code);i++){       //When you upload the code the first
time keep it commented
//          EEPROM.get(i, code[i]);           //Upload the code and change it to store it in the
EEPROM
//          }       //Then uncomment this for loop and reupload the code (It's done only once)
       }
void loop() {
 key = kpd.getKey(); //storing pressed key value in a char
if (key!=NO_KEY)
DetectButtons();
if (result==true)
CalculateResult();
DisplayResult();
}
void DetectButtons()
{
    lcd.clear(); //Then clean it
   if (key=='*') //If cancel Button is pressed
   {Serial.println ("Button Cancel"); Number=Num1=Num2=0; result=false;}
       if (key == '1') //If Button 1 is pressed
   {Serial.println ("Button 1");
   if (Number==0)
   Number=1;
   else
   Number = (Number*10) + 1; //Pressed twice
   }
```

```
    if (key == '4') //If Button 4 is pressed
{Serial.println ("Button 4");
if (Number==0)
Number=4;
else
Number = (Number*10) + 4; //Pressed twice
}
    if (key == '7') //If Button 7 is pressed
{Serial.println ("Button 7");
if (Number==0)
Number=7;
else
Number = (Number*10) + 7; //Pressed twice
}
  if (key == '0')
{Serial.println ("Button 0"); //Button 0 is Pressed
if (Number==0)
Number=0;
else
Number = (Number*10) + 0; //Pressed twice
}
    if (key == '2') //Button 2 is Pressed
{Serial.println ("Button 2");
 if (Number==0)
Number=2;
else
Number = (Number*10) + 2; //Pressed twice
}
    if (key == '5')
{Serial.println ("Button 5");
 if (Number==0)
Number=5;
else
```

```
Number = (Number*10) + 5; //Pressed twice
}
    if (key == '8')
{Serial.println ("Button 8");
 if (Number==0)
Number=8;
else
Number = (Number*10) + 8; //Pressed twice
}
  if (key == '#')
{Serial.println ("Button Equal");
Num2=Number;
result = true;
}
    if (key == '3')
{Serial.println ("Button 3");
 if (Number==0)
Number=3;
else
Number = (Number*10) + 3; //Pressed twice
}
    if (key == '6')
{Serial.println ("Button 6");
if (Number==0)
Number=6;
else
Number = (Number*10) + 6; //Pressed twice
}
    if (key == '9')
{Serial.println ("Button 9");
if (Number==0)
Number=9;
else
```

```
  Number = (Number*10) + 9; //Pressed twice
  }
    if (key == 'A' || key == 'B' || key == 'C' || key == 'D') //Detecting Buttons on Column 4
 {
  Num1 = Number;
  Number =0;
  if (key == 'A')
  {Serial.println ("Addition"); action = '+';}
   if (key == 'B')
  {Serial.println ("Subtraction"); action = '-'; }
   if (key == 'C')
  {Serial.println ("Multiplication"); action = '*';}
   if (key == 'D')
  {Serial.println ("Division"); action = '/';}
  delay(100);
 }
}
void CalculateResult()
{
 if (action=='+')
   Number = Num1+Num2;
 if (action=='-')
   Number = Num1-Num2;
 if (action=='*')
   Number = Num1*Num2;
 if (action=='/')
   Number = Num1/Num2;
}
void DisplayResult()
{
 lcd.setCursor(0, 0);   // set the cursor to column 0, line 1
 lcd.print(Num1); lcd.print(action); lcd.print(Num2);
  if (result==true)
```

```
 {lcd.print(" ="); lcd.print(Number);} //Display the result
   lcd.setCursor(0, 1);   // set the cursor to column 0, line 1
 lcd.print(Number); //Display the result
}
```

Co-funded by the Erasmus+ Programme of the European Union

Erasmus+ KA-202

Strategisches Partnerschaftsprojekt für die berufliche Bildung und Ausbildung

Projekttitel: "Lehren und Lernen mit Arduinos in der Berufsbildung"

Projekt-Akronym: "ARDUinVET"

Projekt Nr.: “2020-1-TR01-KA202-093762”

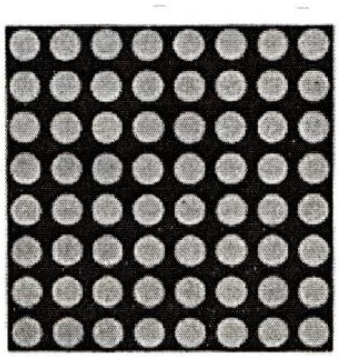
Dot-Matrix-Modul und Schulungskit

Was ist ein Dot-Matrix-Display?

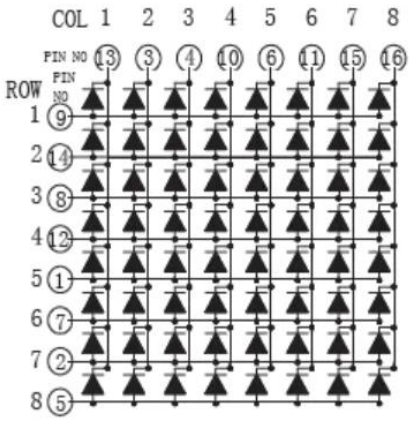
Eine Dot-Matrix-Anzeige ist eine Gruppe von LEDs, die in einem bestimmten System angeordnet sind. Eine Standard-LED-Matrix besteht aus 5x7 oder 8x8 LEDs, die in Reihen und Spalten angeordnet sind, wie in der Abbildung unten dargestellt. Eine 5*7 DOT-Matrix hat 5 Zeilen und

7 Spalten von LEDs und eine 8*8 DOT-Matrix hat 8 Zeilen und 8 Spalten von LEDs, die miteinander verbunden sind.

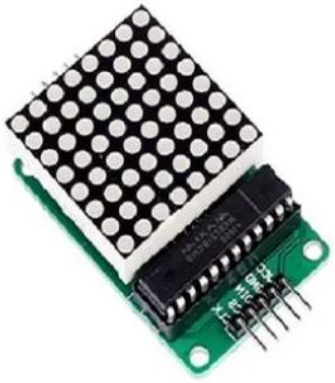
5*7 DOT-Matrix-Anzeige

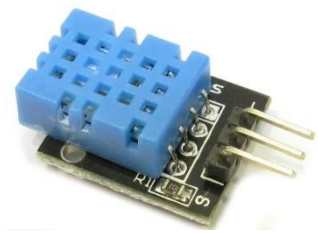


8*8 DOT-Matrix-Anzeige

Wenn wir uns ein Stück der 8x8-Punktmatrix ansehen, enthält es 16 Pins, von denen 8 für Zeilen und 8 für Spalten verwendet werden. Das bedeutet, dass in Zeilen und Spalten insgesamt 64 LEDs vorhanden sind, angefangen von Pin 1 bis Pin 8. Pin Nummer 1 ist R5 (Reihe-5) und Pin Nummer 8 ist R3 (Reihe-3) auf der Unterseite. Auf der Oberseite befindet sich von Pin 9 (Reihe-1) bis Pin 16 (Spalte-1). Aber ein Neuling immer verwirrt und beginnt bei Null, weil wir das Bild / Diagramm kennen. oft bekommen wir aus irgendeiner Quelle, auch wir haben zu sortieren, die eine +VE und -VE. vielleicht ein Experte kann von gemeinsamen Kathode / Anode Typ zu verstehen.	

Die LED-Punktmatrixanzeige MAX7219 ist eine der beliebtesten Anzeigen auf dem Markt und wird von	

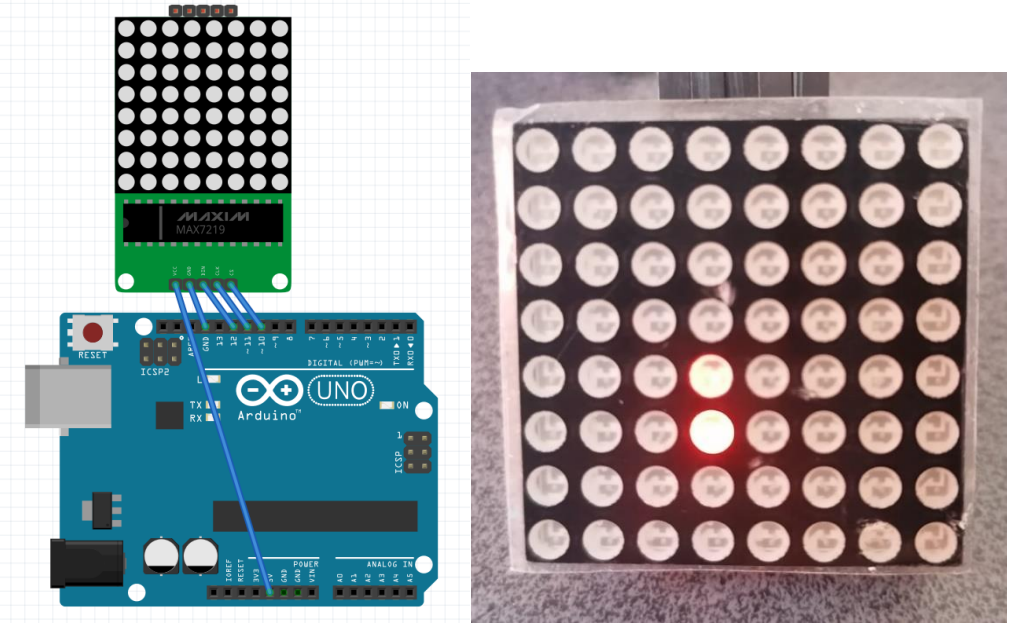
Studenten, Elektronikbastlern und vor allem in industriellen Anwendungen eingesetzt. Es gibt zwei Arten von Modulen, die allgemein erhältlich sind. Diese sind das generische Modul. Jedes Modul besteht aus zwei Einheiten. Das eine ist die 8X8-LED-Punktmatrix und das andere ist der MAX7219 IC. MAX7219 ist ein Anzeigentreiber-IC mit gemeinsamer Kathode und seriellen Ein- und Ausgängen. Er hat eine einstellbare Stromstärke, die mit nur einem externen Widerstand eingestellt werden kann. Darüber hinaus verfügt er über eine vieradrige serielle Schnittstelle, die leicht an alle Mikroprozessoren angeschlossen werden kann. Er kann 64 einzelne LEDs, die an seinen Ausgangspins angeschlossen sind, über nur 4 Drähte mit Hilfe von Arduino ansteuern. Außerdem kann er Punktmatrixanzeigen, 7-Segment-Anzeigen und Balkendiagramme ansteuern. Darüber hinaus verfügt der MAX7219 über einen eingebauten BCD-Decoder, der die Verwendung mit numerischen 7-Segment-Anzeigen erleichtert. Außerdem verfügt er über ein 8×8 statisches RAM, das zum Speichern von Zahlen verwendet werden kann. Er ist einer der beliebtesten Anzeigetreiber-ICs.	

	Der DHT11 ist ein digitaler Temperatur- und Feuchtigkeitssensor. Er verwendet einen kapazitiven Feuchtigkeitssensor und einen Thermistor, um die Umgebungsluft zu messen und gibt ein digitales Signal am Datenpin aus (keine analogen Eingangspins erforderlich).

Schaltung 1:

Titel der Schaltung: Alle LEDs nacheinander anzeigen

Beschreibung der Schaltung: Die Dot-Matrix zeigt alle LEDs nacheinander an



1-) Breadboard view

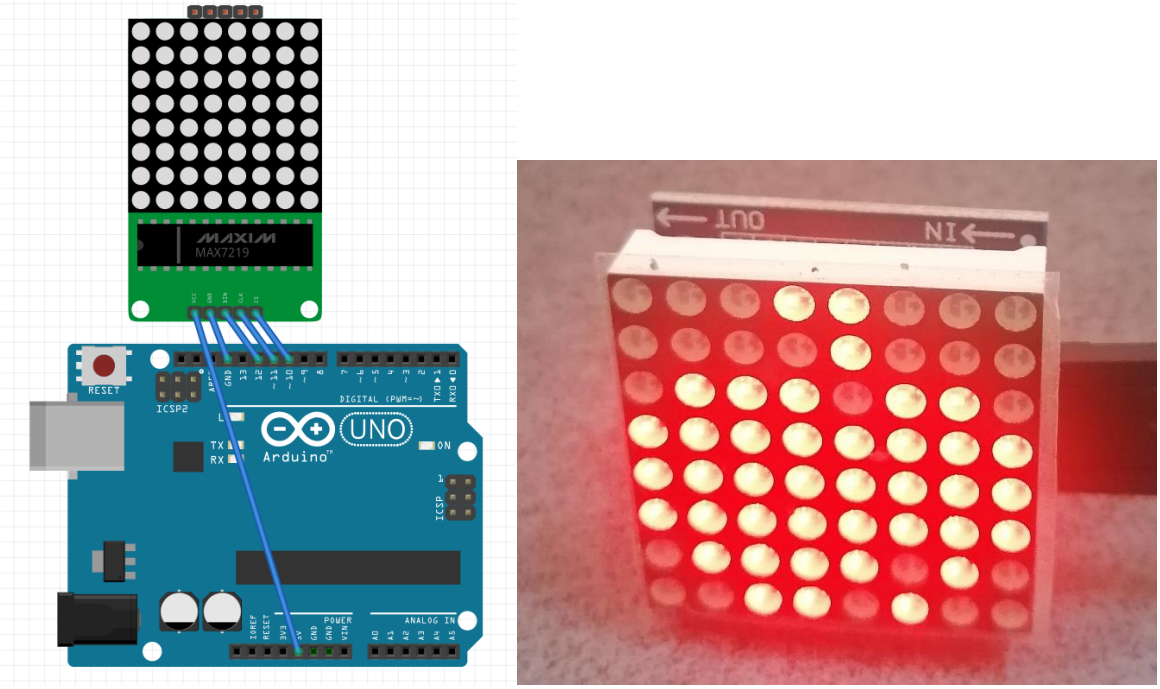
```
/* Alle LEDs nacheinander anzeigen */
#include <LedControl.h>
int DIN = 12;
int CS = 10;
int CLK = 11;
LedControl lc=LedControl(DIN, CLK, CS,0);
void setup() {
  lc.shutdown(0,false);
  lc.setIntensity(0,0);
  lc.clearDisplay(0);}
void loop() {
  for(int j=0;j<8;j++){
```

```
    for(int i=0;i<8;i++){
      lc.setLed(0,j,i,true);
      delay(100);
      lc.setLed(0,j,i,false); } }}
```

Schaltung 2:

Titel der Schaltung: Anzeige des Apple-Symbols

Beschreibung der Schaltung: Die Dot-Matrix-Schnittstelle zeigt das Apple-Symbol an



1-) Breadboard view

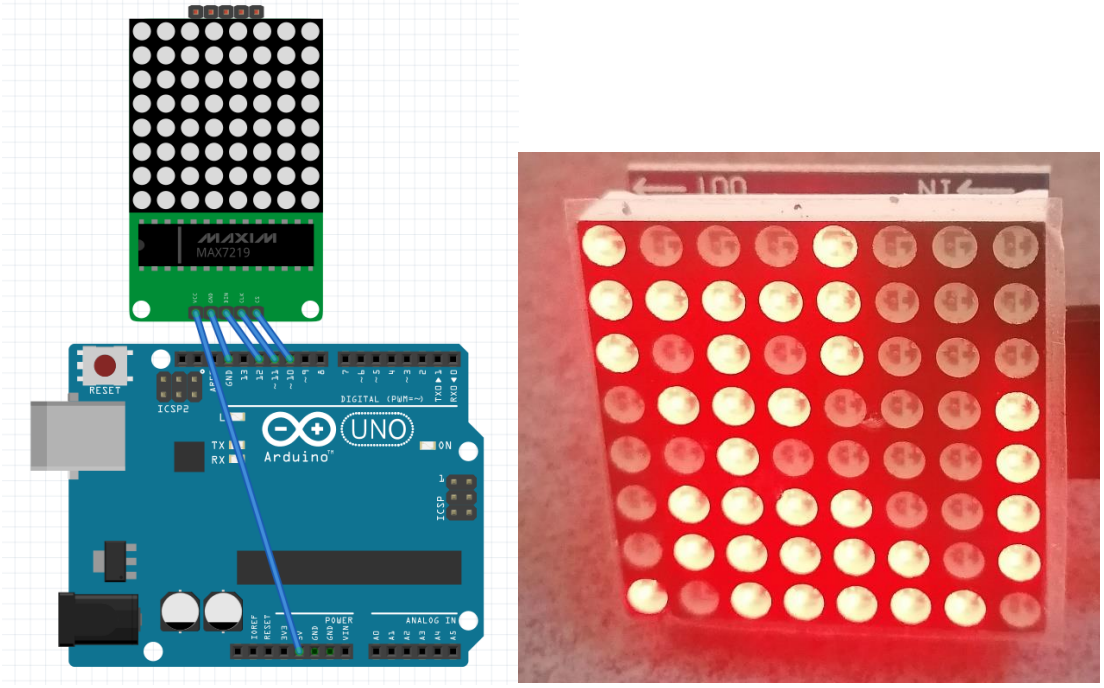
```
/* Anzeige des Apple-Symbols*/
#include <LedControl.h>
int DIN = 12;
int CS = 10;
int CLK = 11;
LedControl lc=LedControl(DIN, CLK, CS,0);
byte Apple
[8]={B00011000,B00001000,B01110110,B11111111,B11111111,B11111111,B01111010,B001
10100};
void setup() {
  lc.shutdown(0,false);
```

```
lc.setIntensity(0,0);
lc.clearDisplay(0);  }
void loop(){
for(int i=0;i<8;i++) lc.setRow(0,i,Apple[i]);   }
```

Schaltung 3:

Titel der Schaltung: Display a cat icon

Beschreibung der Schaltung: Die Dot-Matrix-Schnittstelle zeigt ein Katzensymbol an



1-) Breadboard view

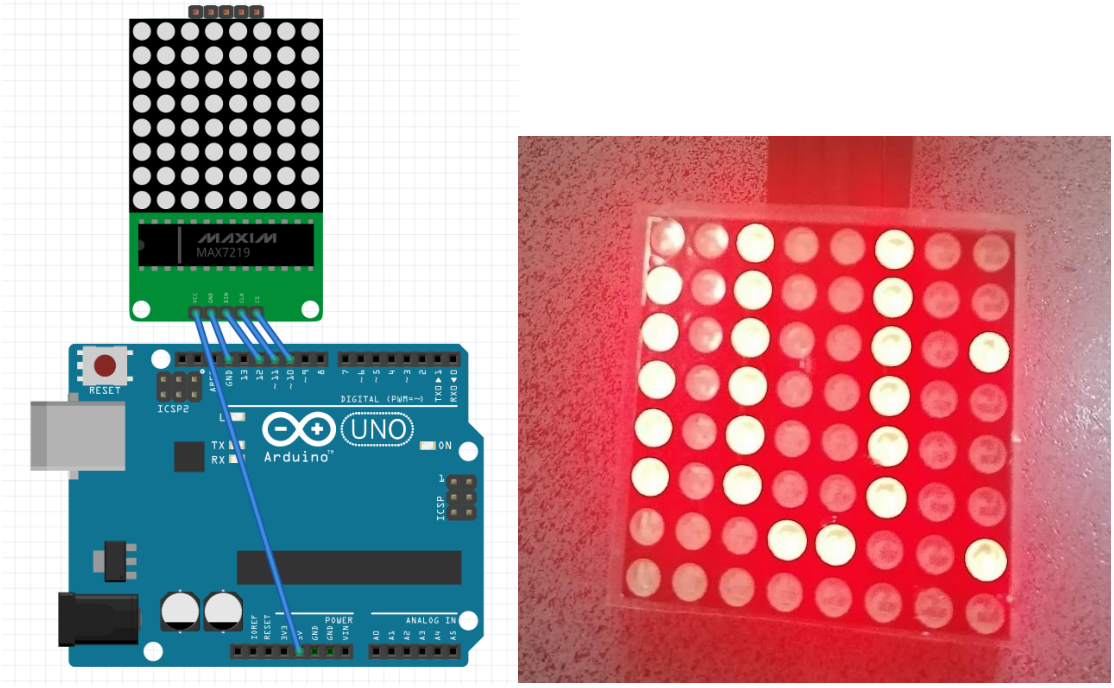
```
/* Anzeige eines Katzensymbols*/
#include <LedControl.h>
int DIN = 12;
int CS = 10;
int CLK = 11;
LedControl lc=LedControl(DIN, CLK, CS,0);
int Cat[8]
={B10001000,B11111000,B10101000,B01110001,B00100001,B01111001,B01111101,B10111
110 };
void setup() {
  lc.shutdown(0,false);
```

```
 lc.setIntensity(0,0);
 lc.clearDisplay(0);  }
 void loop(){
   for(int i=0;i<8;i++) lc.setRow(0,i,Cat[i]); }
```

Schaltung 4:

Titel der Schaltung: Fließtext ARDUINVET anzeigen

Beschreibung der Schaltung: Die Dot-Matrix-Schnittstelle zeigt den Fließtext ARDUINVET



1-) Ansicht auf der Leiterplatte

```
/*  Display flowing text ARDUINVET*/
//D10=CS
//D11=CLK
//D12=DIN
#include <MaxMatrix.h> //include matrix library
#include <avr/pgmspace.h>
#include <stdlib.h>
PROGMEM const unsigned char CH[] = {
3, 8, B00000000, B00000000, B00000000, B00000000, B00000000, // space
1, 8, B01011111, B00000000, B00000000, B00000000, B00000000, // !
3, 8, B00000011, B00000000, B00000011, B00000000, B00000000, // "
```

```
5, 8, B00010100, B00111110, B00010100, B00111110, B00010100, // #
4, 8, B00100100, B01101010, B00101011, B00010010, B00000000, // $
5, 8, B01100011, B00010011, B00001000, B01100100, B01100011, // %
5, 8, B00110110, B01001001, B01010110, B00100000, B01010000, // &
1, 8, B00000011, B00000000, B00000000, B00000000, B00000000, // '
3, 8, B00011100, B00100010, B01000001, B00000000, B00000000, // (
3, 8, B01000001, B00100010, B00011100, B00000000, B00000000, // )
5, 8, B00101000, B00011000, B00001110, B00011000, B00101000, // *
5, 8, B00001000, B00001000, B00111110, B00001000, B00001000, // +
2, 8, B10110000, B01110000, B00000000, B00000000, B00000000, // ,
4, 8, B00001000, B00001000, B00001000, B00001000, B00000000, // -
2, 8, B01100000, B01100000, B00000000, B00000000, B00000000, // .
4, 8, B01100000, B00011000, B00000110, B00000001, B00000000, // /
4, 8, B00111110, B01000001, B01000001, B00111110, B00000000, // 0
3, 8, B01000010, B01111111, B01000000, B00000000, B00000000, // 1
4, 8, B01100010, B01010001, B01001001, B01000110, B00000000, // 2
4, 8, B00100010, B01000001, B01001001, B00110110, B00000000, // 3
4, 8, B00011000, B00010100, B00010010, B01111111, B00000000, // 4
4, 8, B00100111, B01000101, B01000101, B00111001, B00000000, // 5
4, 8, B00111110, B01001001, B01001001, B00110000, B00000000, // 6
4, 8, B01100001, B00010001, B00001001, B00000111, B00000000, // 7
4, 8, B00110110, B01001001, B01001001, B00110110, B00000000, // 8
4, 8, B00000110, B01001001, B01001001, B00111110, B00000000, // 9
2, 8, B01010000, B00000000, B00000000, B00000000, B00000000, // :
2, 8, B10000000, B01010000, B00000000, B00000000, B00000000, // ;
3, 8, B00010000, B00101000, B01000100, B00000000, B00000000, // <
3, 8, B00010100, B00010100, B00010100, B00000000, B00000000, // =
3, 8, B01000100, B00101000, B00010000, B00000000, B00000000, // >
4, 8, B00000010, B01011001, B00001001, B00000110, B00000000, // ?
5, 8, B00111110, B01001001, B01010101, B01011101, B00001110, // @
4, 8, B01111110, B00010001, B00010001, B01111110, B00000000, // A
4, 8, B01111111, B01001001, B01001001, B00110110, B00000000, // B
4, 8, B00111110, B01000001, B01000001, B00100010, B00000000, // C
```

4, 8, B01111111, B01000001, B01000001, B00111110, B00000000, // D

4, 8, B01111111, B01001001, B01001001, B01000001, B00000000, // E

4, 8, B01111111, B00001001, B00001001, B00000001, B00000000, // F

4, 8, B00111110, B01000001, B01001001, B01111010, B00000000, // G

4, 8, B01111111, B00001000, B00001000, B01111111, B00000000, // H

3, 8, B01000001, B01111111, B01000001, B00000000, B00000000, // I

4, 8, B00110000, B01000000, B01000001, B00111111, B00000000, // J

4, 8, B01111111, B00001000, B00010100, B01100011, B00000000, // K

4, 8, B01111111, B01000000, B01000000, B01000000, B00000000, // L

5, 8, B01111111, B00000010, B00001100, B00000010, B01111111, // M

5, 8, B01111111, B00000100, B00001000, B00010000, B01111111, // N

4, 8, B00111110, B01000001, B01000001, B00111110, B00000000, // O

4, 8, B01111111, B00001001, B00001001, B00000110, B00000000, // P

4, 8, B00111110, B01000001, B01000001, B10111110, B00000000, // Q

4, 8, B01111111, B00001001, B00001001, B01110110, B00000000, // R

4, 8, B01000110, B01001001, B01001001, B00110010, B00000000, // S

5, 8, B00000001, B00000001, B01111111, B00000001, B00000001, // T

4, 8, B00111111, B01000000, B01000000, B00111111, B00000000, // U

5, 8, B00001111, B00110000, B01000000, B00110000, B00001111, // V

5, 8, B00111111, B01000000, B00111000, B01000000, B00111111, // W

5, 8, B01100011, B00010100, B00001000, B00010100, B01100011, // X

5, 8, B00000111, B00001000, B01110000, B00001000, B00000111, // Y

4, 8, B01100001, B01010001, B01001001, B01000111, B00000000, // Z

2, 8, B01111111, B01000001, B00000000, B00000000, B00000000, // [

4, 8, B00000001, B00000110, B00011000, B01100000, B00000000, // \ backsplash

2, 8, B01000001, B01111111, B00000000, B00000000, B00000000, //]

3, 8, B00000010, B00000001, B00000010, B00000000, B00000000, // hat

4, 8, B01000000, B01000000, B01000000, B01000000, B00000000, // _

2, 8, B00000001, B00000010, B00000000, B00000000, B00000000, // `

4, 8, B00100000, B01010100, B01010100, B01111000, B00000000, // a

4, 8, B01111111, B01000100, B01000100, B00111000, B00000000, // b

4, 8, B00111000, B01000100, B01000100, B00101000, B00000000, // c

4, 8, B00111000, B01000100, B01000100, B01111111, B00000000, // d

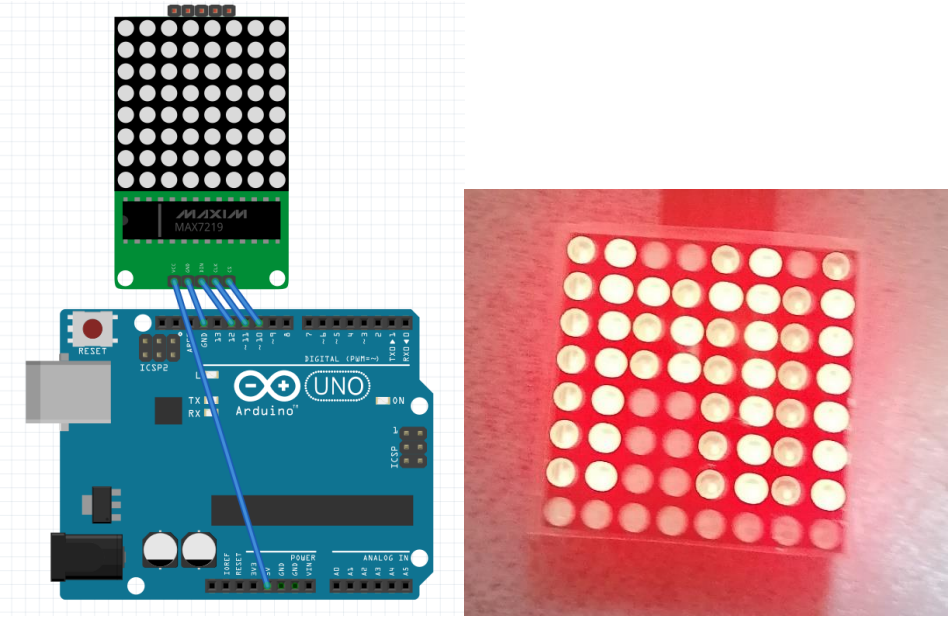
```
4, 8, B00111000, B01010100, B01010100, B00011000, B00000000, // e
3, 8, B00000100, B01111110, B00000101, B00000000, B00000000, // f
4, 8, B10011000, B10100100, B10100100, B01111000, B00000000, // g
4, 8, B01111111, B00000100, B00000100, B01111000, B00000000, // h
3, 8, B01000100, B01111101, B01000000, B00000000, B00000000, // i
4, 8, B01000000, B10000000, B10000100, B01111101, B00000000, // j
4, 8, B01111111, B00010000, B00101000, B01000100, B00000000, // k
3, 8, B01000001, B01111111, B01000000, B00000000, B00000000, // l
5, 8, B01111100, B00000100, B01111100, B00000100, B01111000, // m
4, 8, B01111100, B00000100, B00000100, B01111000, B00000000, // n
4, 8, B00111000, B01000100, B01000100, B00111000, B00000000, // o
4, 8, B11111100, B00100100, B00100100, B00011000, B00000000, // p
4, 8, B00011000, B00100100, B00100100, B11111100, B00000000, // q
4, 8, B01111100, B00001000, B00000100, B00000100, B00000000, // r
4, 8, B01001000, B01010100, B01010100, B00100100, B00000000, // s
3, 8, B00000100, B00111111, B01000100, B00000000, B00000000, // t
4, 8, B00111100, B01000000, B01000000, B01111100, B00000000, // u
5, 8, B00011100, B00100000, B01000000, B00100000, B00011100, // v
5, 8, B00111100, B01000000, B00111100, B01000000, B00111100, // w
5, 8, B01000100, B00101000, B00010000, B00101000, B01000100, // x
4, 8, B10011100, B10100000, B10100000, B01111100, B00000000, // y
3, 8, B01100100, B01010100, B01001100, B00000000, B00000000, // z
3, 8, B00001000, B00110110, B01000001, B00000000, B00000000, // {
1, 8, B01111111, B00000000, B00000000, B00000000, B00000000, // |
3, 8, B01000001, B00110110, B00001000, B00000000, B00000000, // }
4, 8, B00001000, B00000100, B00001000, B00000100, B00000000, // ~
};
int data = 12;    // DIN pin of MAX7219 module
int load = 10;    // CS pin of MAX7219 module
int clock = 11;  // CLK pin of MAX7219 module
int maxInUse = 1;    //change this variable to set how many MAX7219's you'll use
MaxMatrix m(data, load, clock, maxInUse); // define module
byte buffer[10];
```

```
void setup(){
  m.init(); // module initialize
  m.setIntensity(2); // dot matix intensity 0-15
  Serial.begin(9600); // serial communication initialize
  Serial.println("ARDUinVET");  }
void loop(){
     printStringWithShift("ARDUinVET    ", 100);
   m.shiftLeft(false, true);  }
void printCharWithShift(char c, int shift_speed){
  if (c < 32) return;
  c -= 32;
  memcpy_P(buffer, CH + 7*c, 7);
  m.writeSprite(32, 0, buffer);
  m.setColumn(32 + buffer[0], 0);
  for (int i=0; i<buffer[0]+1; i++)
  {    delay(shift_speed);
    m.shiftLeft(false, false);  }
}
void printStringWithShift(char* s, int shift_speed){
  while (*s != 0){
    printCharWithShift(*s, shift_speed);
    s++;  }
}
void printString(char* s)
{  int col = 0;
  while (*s != 0)
  {    if (*s < 32) continue;
    char c = *s - 32;
    memcpy_P(buffer, CH + 7*c, 7);
    m.writeSprite(col, 0, buffer);
    m.setColumn(col + buffer[0], 0);
    col += buffer[0] + 1;
    s++;  }        }
```

Schaltung 5:

Titel der Schaltung: Anzeige aller Buchstaben des Alphabets

Beschreibung der Schaltung: Die Dot Matrix Schnittstelle zeigt jeden Buchstaben des Alphabets nach 1 Sekunde an



1-) Ansicht auf der Leiterplatte

```
/*  Display every letter of alphabet*/
#include <MaxMatrix.h>//download from https://code.google.com/archive/p/arudino-maxmatrix-library/downloads
int DIN = 12;   // DIN pin of MAX7219 module
int CLK = 11;   // CLK pin of MAX7219 module
int CS = 10;    // CS pin of MAX7219 module
int maxInUse = 1;
MaxMatrix m(DIN, CS, CLK, maxInUse);
char A[] = {8, 8,
B00111000,B01000100,B01000100,B01000100,B01111100,B01000100,B01000100,B01000010
0};
```

```
char B[] = {8, 8,
B01111000,B01000100,B01000100,B01111000,B01000100,B01000100,B01111000,B00000000};
char c[] = {8, 8,
B00000000,B00111100,B01000000,B01000000,B01000000,B01000000,B00111100,B00000000};
char d[] = {8, 8,
B00000000,B01111000,B01000100,B01000100,B01000100,B01000100,B01111000,B00000000};
char e[] = {8, 8,
B00000000,B01111100,B01000000,B01000000,B01111100,B01000000,B01000000,B01111100};
char f[] = {8, 8,
B00000000,B01111100,B01000000,B01000000,B01111100,B01000000,B01000000,B01000000};
char g[] = {8, 8,
B00000000,B00111100,B01000000,B01000000,B01001110,B01000010,B00111110};
char h[] = {8, 8,
B00000000,B01000100,B01000100,B01111100,B01000100,B01000100,B01000100,B00000000};
char i[] = {8, 8,
B00000000,B01111100,B00010000,B00010000,B00010000,B00010000,B01111100,B00000000};
char j[] = {8, 8,
B00000000,B00000100,B00000100,B00000100,B00000100,B01000100,B00111000,B00000000};
char k[] = {8, 8,
B00000000,B00100100,B00101000,B00110000,B00101000,B00100100,B00100010,B00000000};
char l[] = {8, 8,
B00000000,B01000000,B01000000,B01000000,B01000000,B01000000,B01111100,B00000000};
```

```
char n[] = {8, 8,
B00000000,B01000010,B01100010,B01010010,B01001010,B01000110,B01000010,B00000000};
char o[] = {8, 8,
B00111100,B01000010,B01000010,B01000010,B01000010,B01000010,B01000010,B00111100};
char p[] = {8, 8,
B00000000,B00111000,B00100100,B00100100,B00111000,B00100000,B00100000,B00000000};
char q[] = {8, 8,
B00111100,B01000010,B01000010,B01000010,B01001010,B01000110,B00111110,B00000001};
char r[] = {8, 8,
B01111000,B01000100,B01000100,B01111000,B01100000,B01010000,B01001000,B01000100};
char s[] = {8, 8,
B00111100,B01000000,B01000000,B00111000,B00000100,B00000100,B01111000,B00000000};
char t[] = {8, 8,
B00000000,B01111100,B00010000,B00010000,B00010000,B00010000,B00010000,B00000000};
char u[] = {8, 8,
B00000000,B01000010,B01000010,B01000010,B01000010,B01000010,B00111100,B00000000};
char v[] = {8, 8,
B00000000,B00100100,B00100100,B00100100,B00100100,B00100100,B00011000,B00000000};
char w[] = {8, 8,
B00000000,B01010100,B01010100,B01010100,B01010100,B01010100,B00111000,B00000000};
char x[] = {8, 8,
B00000000,B01000100,B00101000,B00010000,B00101000,B01000100,B00000000,B00000000};
```

```
char y[] = {8, 8,
B10000001,B01000010,B00100100,B00011000,B00011000,B00011000,B00011000,B00011000};
char z[] = {8, 8,
B00000000,B01111100,B00001000,B00010000,B00100000,B01111100,B00000000,B00000000};void setup() {
  m.init(); // MAX7219 initialization
  m.setIntensity(5); // initial led matrix intensity, 0-15    }
void loop() {
  // Seting the LEDs On or Off at x,y or row,column position
  m.setDot(6,2,true);
  delay(1000);
  m.setDot(6,3,true);
  delay(1000);
  m.clear(); // Clears the display
  for (int i=0; i<8; i++){
    m.setDot(i,i,true);
    delay(300);}
  m.clear();
  // Displaying the character at x,y (upper left corner of the character)
  m.writeSprite(0, 0, A);
  delay(1000);
  m.writeSprite(0, 0, B);
  delay(1000);
  m.writeSprite(0, 0, c);
  delay(1000);
  m.writeSprite(0, 0, d);
  delay(1000);
  m.writeSprite(0, 0, e);
  delay(1000);
  m.writeSprite(0, 0, f);
  delay(1000);
m.writeSprite(0, 0, g);
```

```
 delay(1000);
m.writeSprite(0, 0, h);
 delay(1000);
m.writeSprite(0, 0, i);
 delay(1000);
m.writeSprite(0, 0, j);
 delay(1000);
m.writeSprite(0, 0, k);
 delay(1000);
 m.writeSprite(0, 0, l);
 delay(1000);
m.writeSprite(0, 0, n);
 delay(1000);
m.writeSprite(0, 0, o);
 delay(1000);
  m.writeSprite(0, 0, p);
 delay(1000);
  m.writeSprite(0, 0, q);
 delay(1000);
  m.writeSprite(0, 0, r);
 delay(1000);
  m.writeSprite(0, 0, s);
 delay(1000);
  m.writeSprite(0, 0, t);
 delay(1000);
  m.writeSprite(0, 0, u);
 delay(1000);
  m.writeSprite(0, 0, v);
 delay(1000);
  m.writeSprite(0, 0, w);
 delay(1000);
  m.writeSprite(0, 0, x);
 delay(1000);
```

m.writeSprite(0, 0, y);

delay(1000);

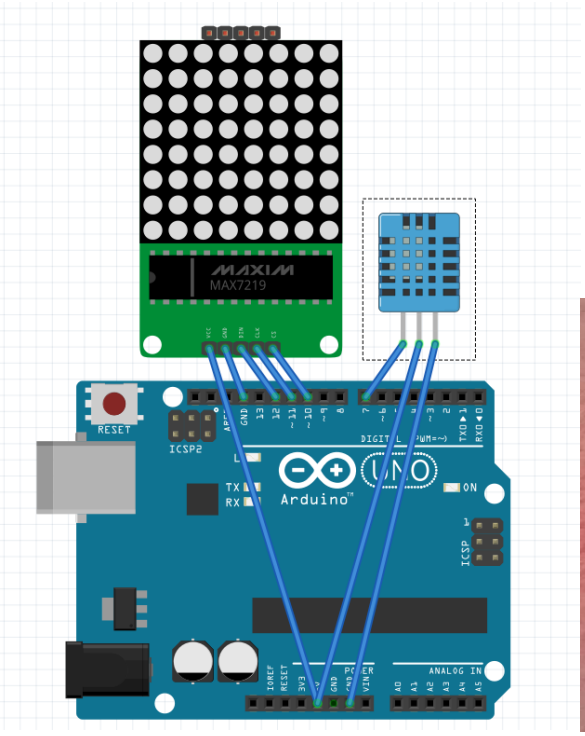
m.writeSprite(0, 0, z);

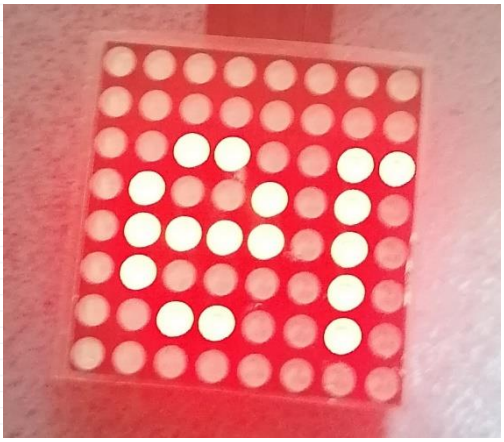
delay(1000);}

Schaltung 6:

Titel der Schaltung: Schnittstelle Dot Matrix mit Arduino

Beschreibung der Schaltung: Die Dot-Matrix-Schnittstelle zeigt die mit dem Temperatursensor DHT11 gemessene Temperatur an





1-) Ansicht auf der Leiterplatte

/* Interface Dot Matrix with Arduino */

//D7= temp

//D10=CS

//D11=CLK

//D12=DIN

#include <MaxMatrix.h> //include matrix library

#include <avr/pgmspace.h>

#include <stdlib.h>

#include "DHT.h" //include the temp sensor library

```
#define DHTPIN 7     // what pin we're connected to
#define DHTTYPE DHT11   // DHT 11 temp&humid sensor
DHT dht(DHTPIN, DHTTYPE);
PROGMEM const unsigned char CH[] = {
3, 8, B00000000, B00000000, B00000000, B00000000, B00000000, // space
1, 8, B01011111, B00000000, B00000000, B00000000, B00000000, // !
3, 8, B00000011, B00000000, B00000011, B00000000, B00000000, // "
5, 8, B00010100, B00111110, B00010100, B00111110, B00010100, // #
4, 8, B00100100, B01101010, B00101011, B00010010, B00000000, // $
5, 8, B01100011, B00010011, B00001000, B01100100, B01100011, // %
5, 8, B00110110, B01001001, B01010110, B00100000, B01010000, // &
1, 8, B00000011, B00000000, B00000000, B00000000, B00000000, // '
3, 8, B00011100, B00100010, B01000001, B00000000, B00000000, // (
3, 8, B01000001, B00100010, B00011100, B00000000, B00000000, // )
5, 8, B00101000, B00011000, B00001110, B00011000, B00101000, // *
5, 8, B00001000, B00001000, B00111110, B00001000, B00001000, // +
2, 8, B10110000, B01110000, B00000000, B00000000, B00000000, // ,
4, 8, B00001000, B00001000, B00001000, B00001000, B00000000, // -
2, 8, B01100000, B01100000, B00000000, B00000000, B00000000, // .
4, 8, B01100000, B00011000, B00000110, B00000001, B00000000, // /
4, 8, B00111110, B01000001, B01000001, B00111110, B00000000, // 0
3, 8, B01000010, B01111111, B01000000, B00000000, B00000000, // 1
4, 8, B01100010, B01010001, B01001001, B01000110, B00000000, // 2
4, 8, B00100010, B01000001, B01001001, B00110110, B00000000, // 3
4, 8, B00011000, B00010100, B00010010, B01111111, B00000000, // 4
4, 8, B00100111, B01000101, B01000101, B00111001, B00000000, // 5
4, 8, B00111110, B01001001, B01001001, B00110000, B00000000, // 6
4, 8, B01100001, B00010001, B00001001, B00000111, B00000000, // 7
4, 8, B00110110, B01001001, B01001001, B00110110, B00000000, // 8
4, 8, B00000110, B01001001, B01001001, B00111110, B00000000, // 9
2, 8, B01010000, B00000000, B00000000, B00000000, B00000000, // :
2, 8, B10000000, B01010000, B00000000, B00000000, B00000000, // ;
3, 8, B00010000, B00101000, B01000100, B00000000, B00000000, // <
```

3, 8, B00010100, B00010100, B00010100, B00000000, B00000000, // =

3, 8, B01000100, B00101000, B00010000, B00000000, B00000000, // >

4, 8, B00000010, B01011001, B00001001, B00000110, B00000000, // ?

5, 8, B00111110, B01001001, B01010101, B01011101, B00001110, // @

4, 8, B01111110, B00010001, B00010001, B01111110, B00000000, // A

4, 8, B01111111, B01001001, B01001001, B00110110, B00000000, // B

4, 8, B00111110, B01000001, B01000001, B00100010, B00000000, // C

4, 8, B01111111, B01000001, B01000001, B00111110, B00000000, // D

4, 8, B01111111, B01001001, B01001001, B01000001, B00000000, // E

4, 8, B01111111, B00001001, B00001001, B00000001, B00000000, // F

4, 8, B00111110, B01000001, B01001001, B01111010, B00000000, // G

4, 8, B01111111, B00001000, B00001000, B01111111, B00000000, // H

3, 8, B01000001, B01111111, B01000001, B00000000, B00000000, // I

4, 8, B00110000, B01000000, B01000001, B00111111, B00000000, // J

4, 8, B01111111, B00001000, B00010100, B01100011, B00000000, // K

4, 8, B01111111, B01000000, B01000000, B01000000, B00000000, // L

5, 8, B01111111, B00000010, B00001100, B00000010, B01111111, // M

5, 8, B01111111, B00000100, B00001000, B00010000, B01111111, // N

4, 8, B00111110, B01000001, B01000001, B00111110, B00000000, // O

4, 8, B01111111, B00001001, B00001001, B00000110, B00000000, // P

4, 8, B00111110, B01000001, B01000001, B10111110, B00000000, // Q

4, 8, B01111111, B00001001, B00001001, B01110110, B00000000, // R

4, 8, B01000110, B01001001, B01001001, B00110010, B00000000, // S

5, 8, B00000001, B00000001, B01111111, B00000001, B00000001, // T

4, 8, B00111111, B01000000, B01000000, B00111111, B00000000, // U

5, 8, B00001111, B00110000, B01000000, B00110000, B00001111, // V

5, 8, B00111111, B01000000, B00111000, B01000000, B00111111, // W

5, 8, B01100011, B00010100, B00001000, B00010100, B01100011, // X

5, 8, B00000111, B00001000, B01110000, B00001000, B00000111, // Y

4, 8, B01100001, B01010001, B01001001, B01000111, B00000000, // Z

2, 8, B01111111, B01000001, B00000000, B00000000, B00000000, // [

4, 8, B00000001, B00000110, B00011000, B01100000, B00000000, // \ backsplash

2, 8, B01000001, B01111111, B00000000, B00000000, B00000000, //]

```
3, 8, B00000010, B00000001, B00000010, B00000000, B00000000, // hat
4, 8, B01000000, B01000000, B01000000, B01000000, B00000000, // _
2, 8, B00000001, B00000010, B00000000, B00000000, B00000000, // `
4, 8, B00100000, B01010100, B01010100, B01111000, B00000000, // a
4, 8, B01111111, B01000100, B01000100, B00111000, B00000000, // b
4, 8, B00111000, B01000100, B01000100, B00101000, B00000000, // c
4, 8, B00111000, B01000100, B01000100, B01111111, B00000000, // d
4, 8, B00111000, B01010100, B01010100, B00011000, B00000000, // e
3, 8, B00000100, B01111110, B00000101, B00000000, B00000000, // f
4, 8, B10011000, B10100100, B10100100, B01111000, B00000000, // g
4, 8, B01111111, B00000100, B00000100, B01111000, B00000000, // h
3, 8, B01000100, B01111101, B01000000, B00000000, B00000000, // i
4, 8, B01000000, B10000000, B10000100, B01111101, B00000000, // j
4, 8, B01111111, B00010000, B00101000, B01000100, B00000000, // k
3, 8, B01000001, B01111111, B01000000, B00000000, B00000000, // l
5, 8, B01111100, B00000100, B01111100, B00000100, B01111000, // m
4, 8, B01111100, B00000100, B00000100, B01111000, B00000000, // n
4, 8, B00111000, B01000100, B01000100, B00111000, B00000000, // o
4, 8, B11111100, B00100100, B00100100, B00011000, B00000000, // p
4, 8, B00011000, B00100100, B00100100, B11111100, B00000000, // q
4, 8, B01111100, B00001000, B00000100, B00000100, B00000000, // r
4, 8, B01001000, B01010100, B01010100, B00100100, B00000000, // s
3, 8, B00000100, B00111111, B01000100, B00000000, B00000000, // t
4, 8, B00111100, B01000000, B01000000, B01111100, B00000000, // u
5, 8, B00011100, B00100000, B01000000, B00100000, B00011100, // v
5, 8, B00111100, B01000000, B00111100, B01000000, B00111100, // w
5, 8, B01000100, B00101000, B00010000, B00101000, B01000100, // x
4, 8, B10011100, B10100000, B10100000, B01111100, B00000000, // y
3, 8, B01100100, B01010100, B01001100, B00000000, B00000000, // z
3, 8, B00001000, B00110110, B01000001, B00000000, B00000000, // {
1, 8, B01111111, B00000000, B00000000, B00000000, B00000000, // |
3, 8, B01000001, B00110110, B00001000, B00000000, B00000000, // }
4, 8, B00001000, B00000100, B00001000, B00000100, B00000000, // ~
```

```
};
int data = 12;    // DIN pin of MAX7219 module
int load = 10;    // CS pin of MAX7219 module
int clock = 11;  // CLK pin of MAX7219 module
int maxInUse = 1;    //change this variable to set how many MAX7219's you'll use
MaxMatrix m(data, load, clock, maxInUse); // define module
byte buffer[10];
void setup(){
  m.init(); // module initialize
  m.setIntensity(2); // dot matix intensity 0-15
  Serial.begin(9600); // serial communication initialize
  Serial.println("DHTxx test!");
  dht.begin();
}
void loop(){
  int t = dht.readTemperature();
  char temp[4];
  itoa(t,temp,10); //convert int to char!!!!
  Serial.println(temp);
   printStringWithShift("BRAILA ROMANIA", 100);
  printStringWithShift("  temp: ", 100);
  printStringWithShift(temp, 100);
  printStringWithShift(" C   ", 100);
  m.shiftLeft(false, true);
  }
void printCharWithShift(char c, int shift_speed){
  if (c < 32) return;
  c -= 32;
  memcpy_P(buffer, CH + 7*c, 7);
  m.writeSprite(32, 0, buffer);
  m.setColumn(32 + buffer[0], 0);
  for (int i=0; i<buffer[0]+1; i++)
  {
```

```
    delay(shift_speed);
    m.shiftLeft(false, false);
  }
}
void printStringWithShift(char* s, int shift_speed){
  while (*s != 0){
    printCharWithShift(*s, shift_speed);
    s++;
  }
}
void printString(char* s)
{
  int col = 0;
  while (*s != 0)
  {
    if (*s < 32) continue;
    char c = *s - 32;
    memcpy_P(buffer, CH + 7*c, 7);
    m.writeSprite(col, 0, buffer);
    m.setColumn(col + buffer[0], 0);
    col += buffer[0] + 1;
    s++;
  }
}
```

Erasmus+ KA-202

Strategisches Partnerschaftsprojekt Berufsbildung

Projekttitel: "Lehren und Lernen von Arduinos in der Berufsbildung"

Projektakronym: " ARDUinVET "

Projektnummer: "2020-1-TR01-KA202-093762"

Arduino Motor Modul und Training KIT

(DC, Step, Servo)

Arduino Motor Modul und Training KIT

Das Motormodul-Trainingskit ist ein didaktisches Werkzeug, das im Rahmen des ARDUinVET-Projekts entwickelt wurde, mit dem Ziel, auf einfache Weise die Funktionsweise verschiedener Arten von Motoren (Servo, Step und DC) zu erklären, die von Arduino gesteuert werden.

Mit der Programmierplattform von Arduino beabsichtigen wir, die Programmierung der Arduinos auf einfache Weise anzugehen, um die verschiedenen Motoren mit der verwendeten Hardware zum Laufen zu bringen.

Ziel ist es, den Studierenden zu zeigen, wie man eine Leiterplatte baut und programmiert (Bild 1), sowie deren Anwendbarkeit, damit die Motoren arbeiten und verschiedene Aufgaben ausführen, in diesem speziellen Fall.

Schaltplan Motorabschirmung

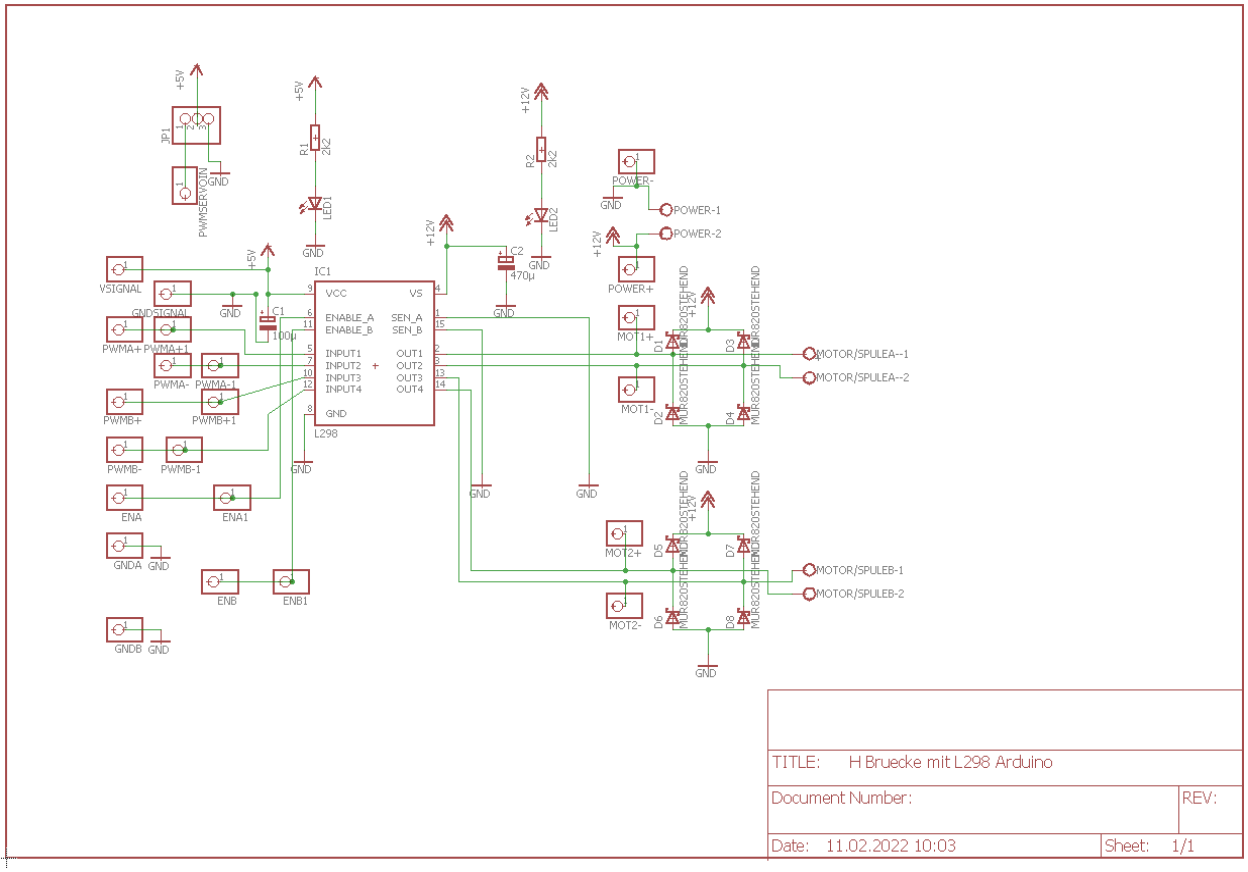


Bild: 1

Layout Motorabschirmung

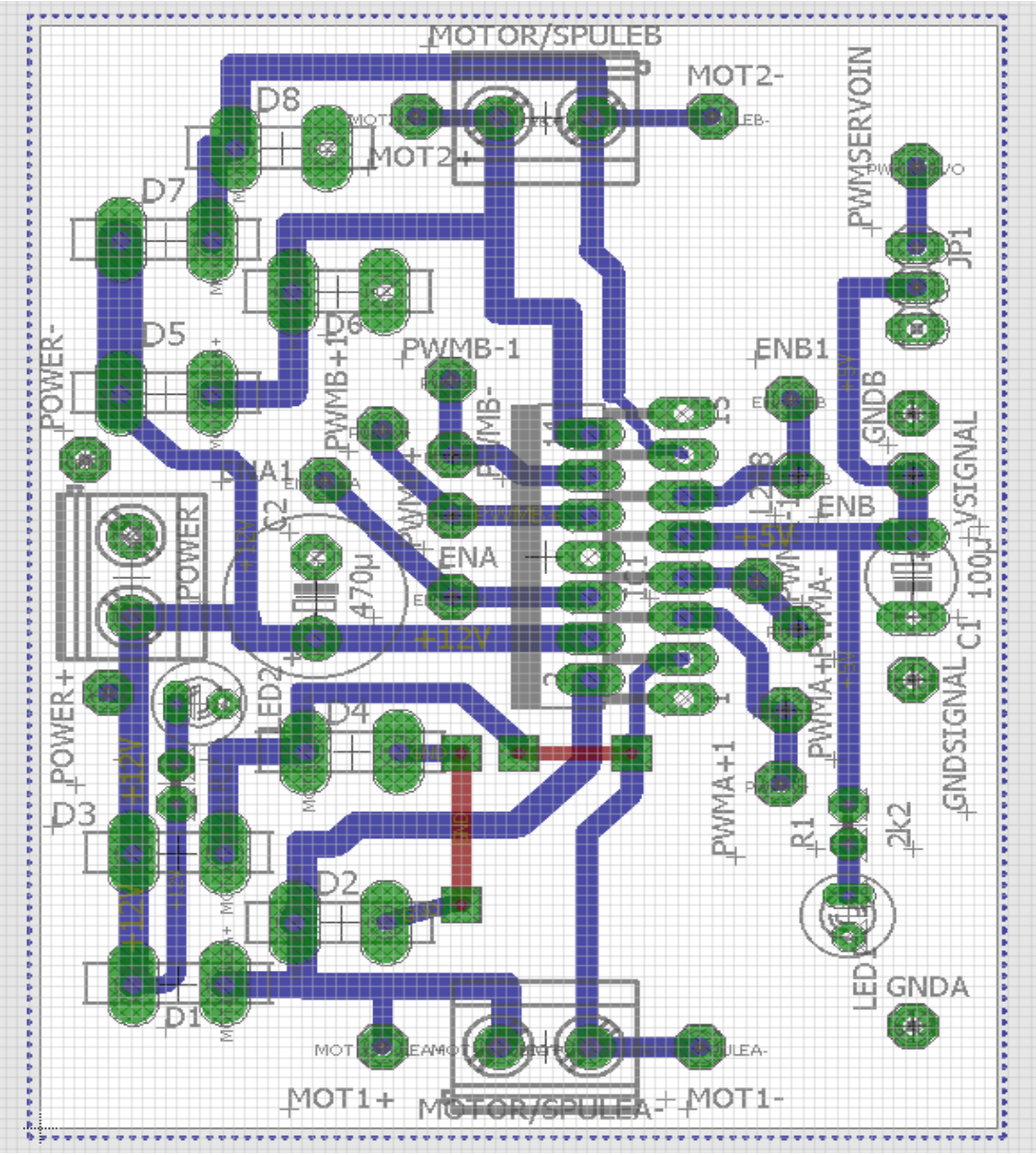


Bild: 2

Motoren

Gleichstrommotoren

Picture SEQ Figura *

Gleichstrommotoren (Gleichstrommotoren, Bild 3) sind elektronische Geräte, die durch die von Magneten erzeugten Anziehungs- und Abstoßungskräfte arbeiten.

Beispiel:

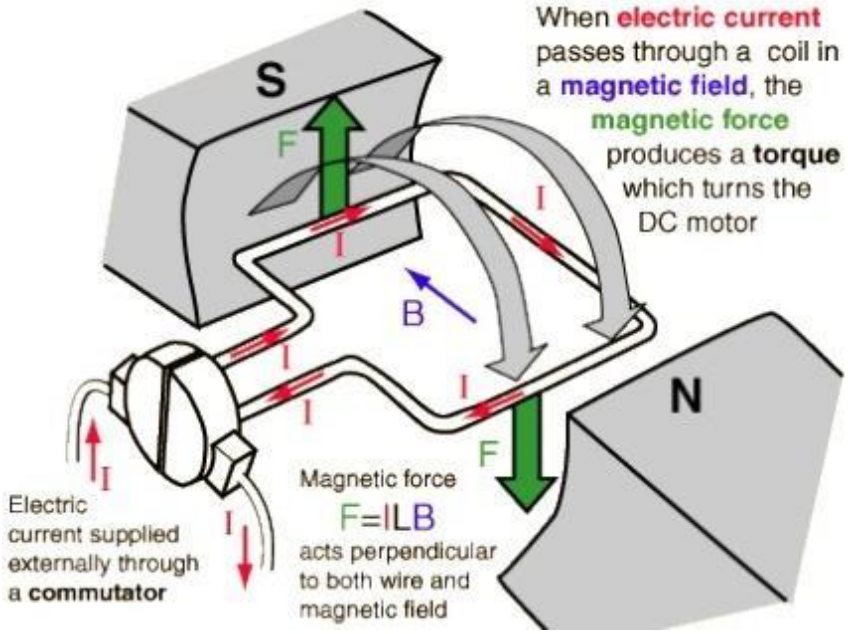
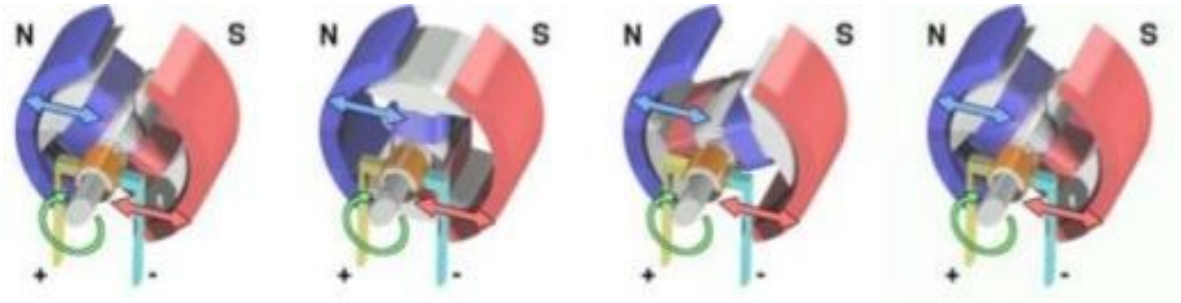


Bild 3: Gleichstrommotoren funktionieren



1 **2** **3** **4**

1. Wenn die Spule gespeist wird, wird ein Magnetfeld um den Rotor herum erzeugt, das eine Abstoßung zwischen den Magneten verursacht, wodurch sich der Motor im Uhrzeigersinn dreht;
2. Der Rotor dreht sich weiter;

3. Wenn der Rotor horizontal ausgerichtet wird, kehrt sich das Magnetfeld um und setzt die Bewegung im Uhrzeigersinn fort.
4. Der Prozess wiederholt sich kontinuierlich, während der Motor gespeist wird.

Schrittmotoren

Der Schrittmotor ist ein Elektromotor, der sich entsprechend einem Impuls bewegt, der von einem Controller empfangen wird. Die Anzahl der Schritte, die der Motor gibt, ist genau die gleiche wie die Anzahl der empfangenen Impulse und die Motordrehzahl ist die gleiche wie die Impulsfrequenz. Daher ist es ein einfaches Gerät (bürstenlos, schalterlos und ohne Encoder). Es handelt sich um Motoren, die in verschiedenen Elektronik- und Automatisierungsbereichen eingesetzt werden, z. B. Robotik, kartesische Achsbewegung.

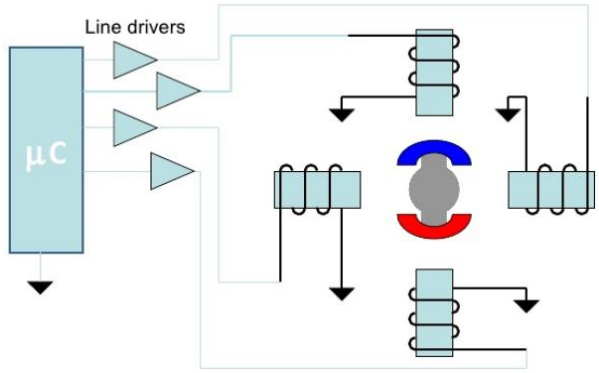


Bild 4: Einfaches Diagramm eines Schrittmotors

Bild 5: Schrittmotor

Beispiele für Schrittmotoren

Unipolarer Motor

Ein unipolarer Motor hat zwei Spulen pro Phase, eine für jede elektrische Stromrichtung. In dieser Konfiguration kann ein Magnetpol invertiert werden, ohne die elektrische Stromrichtung zu ändern, die Schaltung des Schalters kann sehr einfach für jede Spule durchgeführt werden.

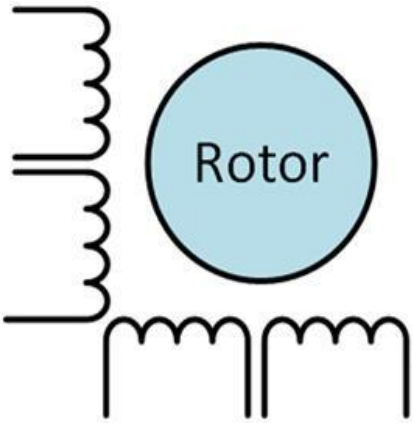


Bild 6: Motor unipolar

Cycle	C1	C2	C3	C4
A	1	0	0	0
B	0	1	0	0
C	0	0	1	0
D	0	0	0	1

Tabelle 1: Statustabelle.

Motor Bipolar

Die Bipolarmotoren haben eine einzigartige Phasenspule. Der elektrische Strom in einer Spule muss invertiert werden, um einen magnetischen Pol umzukehren. Der Stromkreis ist also etwas komplizierter und erfordert die Notwendigkeit einer H-Brücke. Es gibt zwei Verbindungen pro Phase und keine ist gemeinsam.

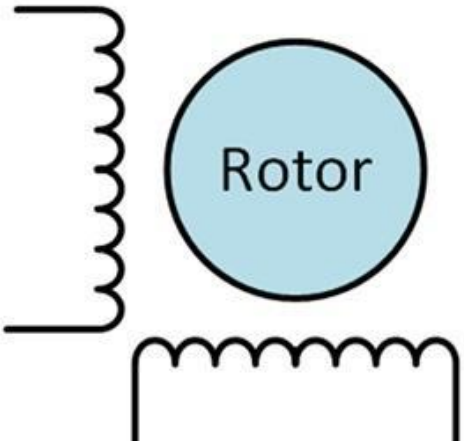


Bild 7: Motor bipolar

Cycle	C1	C2	C3	C4
A	1	0	0	0
B	0	1	0	0
C	0	0	1	0
D	0	0	0	1

Tabelle 2: Statustabelle

Funktion des Schrittmotors

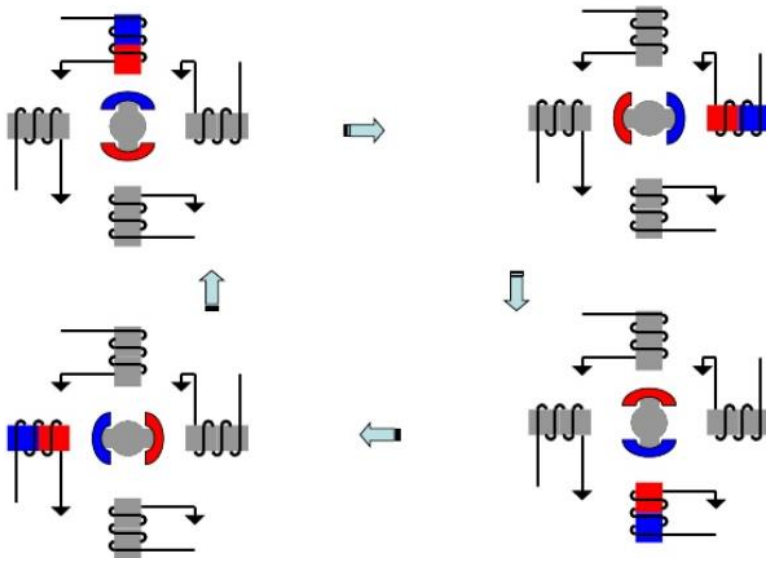


Bild 8: Schrittmotorschema

Servo Motors

Ein Servomotor ist eine elektromechanische Vorrichtung, die durch ein elektrisches Signal die Achse in eine Winkelposition bringt. Normalerweise ist dieser Motortyp kompakt und ermöglicht eine präzise Position der eigenen Achse. Derzeit werden die Servomotoren in der Robotik und Modellierung eingesetzt.

Bild1:Servomotor

Operation:

Ein Servomotor kann in vier Teile unterteilt werden:

- **Steuerschaltung** – Verantwortlich für den Empfang der PWM-Signale und Energie. Steuert die Position des Potentiometers und steuert den Motor entsprechend dem empfangenen PWM-Signal.
- **Potentiometer** – Wird mit der Servoausgangsachse verbunden und positioniert sie.
- **Motor** – Bewegt das Getriebe und die Hauptachse des Servos.
- **Antriebsgetriebe** – Reduziert die Motordrehung und verringert die Festigkeit, so dass ein überlegenes Drehmoment in der Hauptachse angewendet wird. Sie bewegen das Potentiometer zusammen mit der Achse.

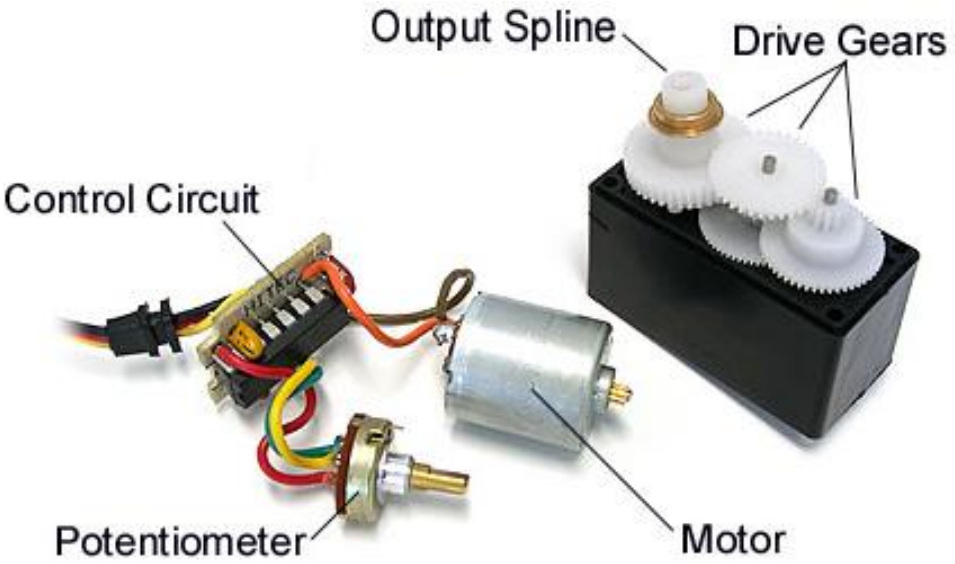


Bild 10: Servomotorteile

Servomotor-Positionen

Die Steuerung eines Servomotors erfolgt durch ein Eingangssignal, das TTL-Spannungen anzeigt, die seine Position angeben. Dieses Signalformat folgt der PWM-Modulation (Pulsweitenmodulation), wie in Bild 11 dargestellt. Die Kodierung in PWM erfolgt durch Hochpegelimpulse in Bezug auf die Gesamtschwingungsdauer, die im Fall der Servomotoren 20ms beträgt.

In diesem speziellen Fall, wenn in 20 Millisekunden 1 Millisekunde auf hohem Niveau ist, sollte sich der Servomotor in der minimalen Position befinden, wie in Bild 11 gezeigt.

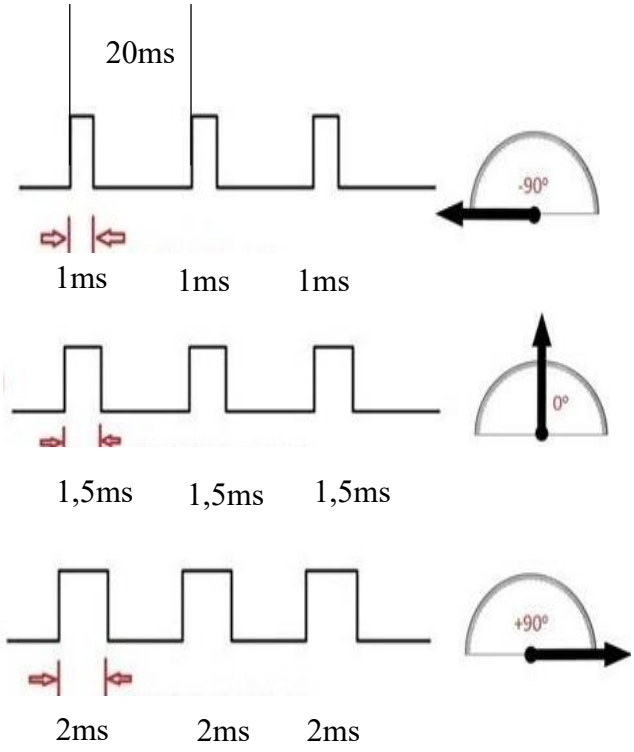


Bild 11: PWM-Servomotor

PWM (Pulsweitenmodulation)

Die PWM kann in verschiedenen Bereichen der Elektronik implementiert werden. Eine seiner Anwendungen ist in der Stromversorgung, DC-Motoren-Drehzahlregelung, Lichtsteuerung, Servomotorsteuerung und mehreren anderen Anwendungen. Durch PWM können wir Geschwindigkeit und Leistung steuern.

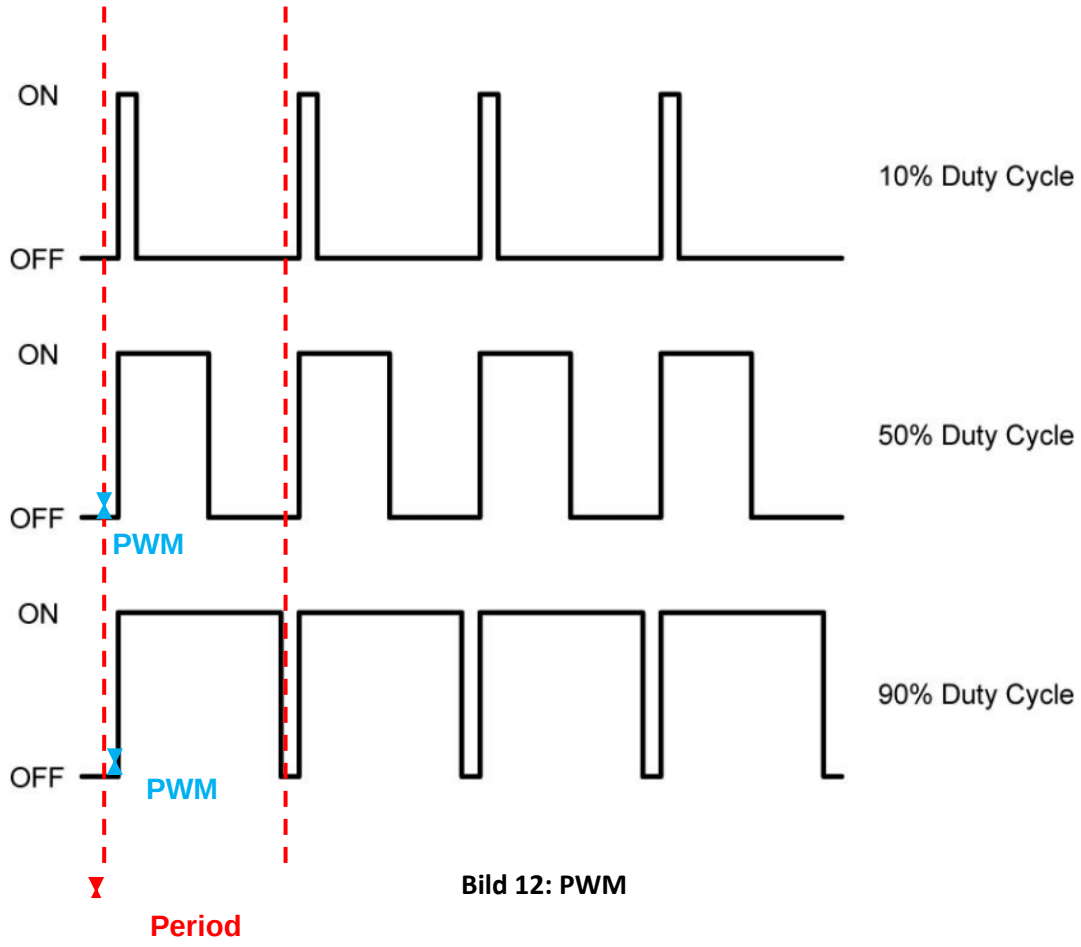


Bild 12: PWM

PWM-Betrieb

Wenn wir eine Rechteckwelle betrachten, müssen wir, um den korrekten Betrieb der PWM zu erhalten, die Pulsbreite der Welle variieren. Um zu berechnen, benötigen wir die Periode und die Pulsbreite und ihr Ergebnis wird Tastverhältnis genannt, da es durch die Gleichung definiert ist:

$$Duty\ Cycle = 100\frac{Pulse\ Width}{Period}$$

Einschaltverhältnis: Prozentwert;

Pulsbreite: Zeitreihenfolge, in der sich das Signal in hohem Pegel befindet;

Periode: Dauer eines Wellenzyklus.

HINWEIS: Die folgenden Experimente werden mit dem Arduino Motor Kit durchgeführt, das die Schüler selbst durchführen.

Schaltungstitel_1 (mit Gleichstrommotoren**):** "Ein Gleichstrommotor einfach"

Schaltungserklärung: Dieses Programm demonstriert Ein- und Ausschaltabläufe eines Gleichstrommotors

Programm:

Ein DC-Motor einfach

```
/*
    Verbindung:
Arduino |  MotorShielt
| PIN Batterie
------------------------
+5V | VSIGNAL
GND | GNDSIGNAL
4 | Ena
5 | PWMA+
6 | PWMA-
*/
//-------------------------------------------------------------
// Variablen
//-------------------------------------------------------------
int ena = 4;    Arduino Pin 4 mit ENA verbunden
int right = 5;  Arduino-Pin 5 mit PWMA+ verbunden
int links = 6;   Arduino Pin 5 mit PWMA verbunden-

//-------------------------------------------------------------
// Setup-Funktion
//-------------------------------------------------------------
void setup()
{
  pinMode(ena, OUTPUT);
  pinMode(rechts, OUTPUT);
  pinMode(links, OUTPUT);
  digitalWrite(ena, HIGH);  ENA aktivieren
  Verzögerung(900);
}
//-------------------------------------------------------------
loop-Funktion
//-------------------------------------------------------------
void loop()
{
```

```
 analogWrite(rechts, 255);  Mit maximaler Geschwindigkeit rechts abbiegen
 Verzögerung(1000);
 analogWrite(right, 0);     Ausschalten
 Verzögerung(1000);
 analogWrite(left, 255);   Mit maximaler Geschwindigkeit links abbiegen
 Verzögerung(1000);
 analogWrite(left, 0);      Ausschalten
 Verzögerung(1000);
 analogWrite(rechts, 127);  Mit halber Geschwindigkeit rechts abbiegen
 Verzögerung(1000);
 analogWrite(right, 0);     Ausschalten
 Verzögerung(1000);
 analogWrite(left, 127);   Mit halber Geschwindigkeit links abbiegen
 Verzögerung(1000);
 analogWrite(left, 0);      Ausschalten
 Verzögerung(1000);
}
```

Schaltung title_2: "One DC-Motor advance"

Schaltungserklärung: Dieses Programm steuert einen Gleichstrommotor. Die Steuerbefehle werden über den seriellen Monitor ausgegeben.

Die Steuerbefehle:

Schreiben Sie in den seriellen Monitor	*Aktion*
aufhören	Der Motor stoppt
nach rechts gehen	Motor rechts abbiegen
nach links	Motor links abbiegen
+	Geschwindigkeit erhöhen
-	Geschwindigkeit verringern

Programm:

Ein DC-Motor-Vorstoß

```
/*
    Verbindung:
Arduino |  MotorShielt
| PIN Batterie
```

```
-------------------------
+5V | VSIGNAL
GND | GNDSIGNAL
4 | Ena
5 | PWMA+
6 | PWMA-
*/
//-------------------------------------------------------------
// Variablen
//-------------------------------------------------------------
int en_a = 4;              ArduPin4-Verbindung  zum Aktivieren des Pin
int right_a = 5;           ArduPin5 Anschluss an PWMA+ Pin
int left_a = 6;            ArduPin6 Verbindung zu PWMA- Pin
String message = "";       Speichern Sie die Nachricht der Serienschnittstelle
int val = 80;              Defalt-Wert für die Geschwindigkeit
int del = 10;              Wert ein- und abnehmen durch Befehl "+" und "-"
int minimumVal = 70;       Mindestgeschwindigkeit / -wert
int maximumVal = 250;      Maximale Geschwindigkeit / Wert
bool flag_go_right = false; //wird true, wenn der Befehl "go right" ausgegeben wird
bool flag_go_left = false;  wird wahr, wenn der Befehl "Gehe nach links" ausgegeben wird

//-------------------------------------------------------------
// Setup-Funktion
//-------------------------------------------------------------
void setup()
{
 Serial.begin(9600);
 pinMode(en_a, OUTPUT);
 pinMode(right_a, OUTPUT);
 pinMode(left_a, OUTPUT);
 digitalWrite(en_a, LOW);
 analogWrite(right_a, 0);
 analogWrite(left_a, 0);
 Verzögerung(900);
}
//-------------------------------------------------------------
loop-Funktion
//-------------------------------------------------------------
void loop()
{
 if(Serial.available()>0)
 {
   message = Serial.readString();
Serial.println(" --> Eingehende Nachricht: " + Nachricht ); Sie sehen die eingehende Nachricht

   if(message.equals("stop\n"))
   {
    digitalWrite(en_a, LOW);
            analogWrite(right_a, 0);
    analogWrite(left_a, 0);
```

```
    flag_go_right = false;
    flag_go_left = false;
Serial.println(" <-- Ausgehende Nachricht: stop \n");
  }
else if(message.equals("go right\n")) //check message if "go right"
  {
digitalWrite(en_a, HOCH);        setzen Sie Pin des Moduls auf  HIGH
            analogWrite(right_a, val);       Schreiben Sie den Wert in den PWM-Pin von rechts
    analogWrite(left_a, 0);          Schreiben Sie Null auf den PWM-Pin von links
    flag_go_left = falsch;           Setzen Sie die Flags
    flag_go_right = wahr;
    Serial.println(" <-- Ausgehende Nachricht: nach rechts gehen\n");  Feedback
  }
  else if(message.equals("go left\n"))
  {
    digitalWrite(en_a, HOCH);
            analogWrite(right_a, 0);
    analogWrite(left_a, val);
    flag_go_right = false;
    flag_go_left = wahr;
    Serial.println(" <-- Ausgehende Nachricht: nach links\n");
  }
  else if (message.equals("+\n")) //check message if "+"
  {
    if(val >= maximumVal) // Maximalwert erreichen
    {
      Serial.println(" <-- Ausgehende Nachricht: Maximalwert! \n");
    }
    else if (val < maximumVal)
    {
      val = val + del;    Steigern Sie den Wert
      if(flag_go_right) // check witch direction should be increase
      {
        analogWrite(right_a, val);
      }
      else if(flag_go_left)
      {
        analogWrite(left_a, val);
      }
Serial.print(" <-- Ausgehende Nachricht: value = ");  Serial.println(val);  Serial.println(""); Feedback
    }
  }
  else if(message.equals("-\n"))
  {
    if(val <= minimumVal) // erreichen Minimum Speed
    {
      Serial.println(" <-- Ausgehende Nachricht: Mindestwert! \n");
    }
    else if (val > minimumVal)
    {
```

```
      val = val - del;
      if(flag_go_right)
      {
        analogWrite(right_a, val);
      }
      else if(flag_go_left)
      {
        analogWrite(left_a, val);
      }
Serial.print(" <-- Ausgehende Nachricht: value = ");  Serial.println(val);  Serial.println("");
    }
   }
   else / /Nachricht generieren, wenn die Nachricht unbekannt ist
   {
Serial.println("<<-- Eingehende Nachricht " + Nachricht + " IST UNBEKANNT \n\n");
   }
  }
}
```

Schaltungstitel_3: "Zwei DC-Motoren vorrücken"

Schaltungserklärung: Dieses Programm steuert zwei Gleichstrommotoren. Die Steuerbefehle werden über den seriellen Monitor ausgegeben.

Der Steuerbefehl:

Schreiben Sie in den seriellen Monitor	*Aktion*
aufhören	Der Motor stoppt
Stoppen Sie eine	Stopp Motor A
Haltestelle B	Stopp Motor b
Gehen Sie nach rechts a	Motor rechts abbiegen
Gehen Sie nach links a	Motor links abbiegen
Gehe nach rechts b	Motor b rechts abbiegen
nach links gehen b	Motor b links abbiegen
+	Geschwindigkeit erhöhen
-	Geschwindigkeit verringern

Programm:

zwei DC-Motoren vorrücken

```
/*
 Verbindung:
Arduino |  MotorShielt
| PIN Batterie
-------------------------
+5V | VSIGNAL
GND | Gnd
4 | Ena
5 | PWMA+
6 | PWMA-
8 | ENB
9 | PWMB+
10 | PWMB-
*/

//-------------------------------------------------------------
variablen
//-------------------------------------------------------------
String message = "";

int en_a = 4;
int right_a = 5;
int left_a = 6;
bool flag_go_right_a = falsch;
bool flag_go_left_a = falsch;

int en_b = 8;
 int right_b = 9;
int left_b = 10;
bool flag_go_right_b = falsch;
bool flag_go_left_b = falsch;

int val = 255;
int del = 10;
int minimumVal = 70;
int maximumVal = 250;

//-------------------------------------------------------------
Setup-Funktion
//-------------------------------------------------------------
void setup()
{
  Serial.begin(9600);

  pinMode(en_a, OUTPUT);
  pinMode(right_a, OUTPUT);
```

```
 pinMode(left_a, OUTPUT);

 pinMode(en_b, OUTPUT);
 pinMode(right_b, OUTPUT);
 pinMode(left_b, OUTPUT);

 digitalWrite(en_a, LOW);
 analogWrite(right_a, 0);
 analogWrite(left_a, 0);

 digitalWrite(en_b, LOW);
 analogWrite(right_b, 0);
 analogWrite(left_b, 0);

 Verzögerung(900);
}

//-------------------------------------------------------------
loop-Funktion
//-------------------------------------------------------------
void loop()
{
 if(Serial.available()>0)
 {
  message = Serial.readString();
Serial.println(" --> Eingehende Nachricht: " + Nachricht);

  if(message.equals("stop\n")) // beide Motoren stoppen
  {
   digitalWrite(en_a, LOW);
          analogWrite(right_a, 0);
   analogWrite(left_a, 0);
   flag_go_right_a = falsch;
   flag_go_left_a = falsch;

   digitalWrite(en_b, LOW);
   analogWrite(right_b, 0);
   analogWrite(left_b, 0);
   flag_go_right_b = falsch;
   flag_go_left_b = falsch;

   Serial.println(" <-- Ausgehende Nachricht: stop all\n");
  }
else if(message.equals("stop a\n")) // motor a stop
  {
   digitalWrite(en_a, LOW);
   analogWrite(right_a, 0);
   analogWrite(left_a, 0);
   flag_go_right_a = falsch;
   flag_go_left_a = falsch;
```

```
    Serial.println(" <-- Ausgehende Nachricht: stop a\n");
   }
   else if(message.equals("stop b\n")) // Motor B Stop
   {
    digitalWrite(en_b, LOW);
    analogWrite(right_b, 0);
    analogWrite(left_b, 0);
    flag_go_right_b = falsch;
    flag_go_left_b = falsch;
Serial.println(" <-- Ausgehende Nachricht: stop b\n");
    }
   else if(message.equals("go right a\n"))
   {
    digitalWrite(en_a, HOCH);
            analogWrite(right_a, val);
    analogWrite(left_a, 0);
    flag_go_left_a = falsch;
    flag_go_right_a = wahr;
Serial.println(" <-- Ausgehende Nachricht: go right a\n");
   }
   else if(message.equals("go right b\n"))
   {
    digitalWrite(en_b, HOCH);
    analogWrite(left_b, 0);
    analogWrite(right_b, val);
    flag_go_left_b = falsch;
    flag_go_right_b = wahr;
Serial.println(" <-- Ausgehende Nachricht: go right b\n");
   }
   else if(message.equals("go left a\n"))
   {
    digitalWrite(en_a, HOCH);
            analogWrite(right_a, 0);
    analogWrite(left_a, val);
    flag_go_right_a = falsch;
    flag_go_left_a = wahr;
    Serial.println(" <-- Ausgehende Nachricht: gehen Sie nach links a\n");
   }
   else if(message.equals("go left b\n"))
   {
    digitalWrite(en_b, HOCH);
    analogWrite(right_b, 0);
    analogWrite(left_b, val);
    flag_go_right_b = falsch;
    flag_go_left_b = wahr;
    Serial.println(" <-- Ausgehende Nachricht: links gehen b \n");
   }
   else if(message.equals("+\n"))
   {
    if(val >= maximumVal) // Maximalwert erreichen
```

```
    {
      Serial.println(" <-- Ausgehende Nachricht: Maximalwert! \n");
    }
    else if (val < maximumVal)
    {
      Gutschein = Gutschein + Del;
      if(flag_go_right_a) analogWrite(right_a, val);
      if(flag_go_right_b) analogWrite(right_b, val);
      if(flag_go_left_a) analogWrite(left_a, val);
      if(flag_go_left_b) analogWrite(left_b, val);
Serial.print(" <-- Ausgehende Nachricht: value = ");  Serial.println(val);  Serial.println("");
    }
   }
   else if(message.equals("-\n"))
   {
    if(val <= minimumVal) // erreichen Mindestwert
    {
      Serial.println(" <-- Ausgehende Nachricht: Mindestwert! \n");
    }
    else if (val > minimumVal)
    {
      val = val - del;
      if(flag_go_right_a) analogWrite(right_a, val);
      if(flag_go_right_b) analogWrite(right_b, val);
      if(flag_go_left_a) analogWrite(left_a, val);
      if(flag_go_left_b) analogWrite(left_b, val);

Serial.print(" <-- Ausgehende Nachricht: value = ");  Serial.println(val);  Serial.println("");
    }
   }
   oder
   {
Serial.println(" <<-- Eingehende Nachricht " + Nachricht + " IST UNBEKANNT \n\n");
   }
 }
}
```

Schaltungstitel_4: "Einfaches Schrittmotorprogramm"

Schaltungserklärung: "Ein Schrittmotor dreht sich im Uhrzeigersinn und gegen den Uhrzeigersinn

Programm:

Einfaches Schrittmotorprogramm

```
/*
   Verbindung:
Arduino |  MotorShielt
```

```
| PIN Batterie
------------------------
+5V | VSIGNAL
GND | GNDSIGNAL
4 | Ena
5 | PWMA+
6 | PWMA-
8 | ENB
9 | PWMB+
10 | PWMB-
*/

//---------------------------------------------------------------
//  Bibliotheken
//---------------------------------------------------------------
#include <Stepper.h>
Libraty für den Stepper
//---------------------------------------------------------------
// Konstanten
//---------------------------------------------------------------
#define SCHRITTE 200
Schrittwinkel des verwendeten Schrittmotors = 1,8 Grad
360 / 1,8° = 200

//---------------------------------------------------------------
// Klasse
//---------------------------------------------------------------
Schrittmotor (STEPS, 5,6,9,10);
  Erstellen einer neuen Instanz der Stepper-Klasse
  Parameter:
  // SCHRITTE -> die Anzahl der Schritte in einer Umdrehung des Motors
  // 5,6,9,10 -> gebrauchte Pins

//---------------------------------------------------------------
variablen
//---------------------------------------------------------------
int spe = 50;

//---------------------------------------------------------------
Setup-Funktion
//---------------------------------------------------------------
void setup()
{
  Serial.begin(9600);
  pinMode(4, INPUT);
  pinMode(8, INPUT);
digitalWrite(4 ,HIGH);    ENA Pin des MotorShielt aktivieren
digitalWrite(8 ,HIGH);    ENB Pin des MotorShielt aktivieren

  Motor.setSpeed(spe);  Objekt Motor erhält die Geschwindigkeit in "Umdrehung pro Minute"
```

```
}

void loop()
{
  Motor.step(100);    Dreht den Motor um eine bestimmte Anzahl von Schritten,
mit einer Geschwindigkeit, die durch den letzten Aufruf von setSpeed() bestimmt wird;
                in diesem Fall 100 Schritte im Uhrzeigersinn;
  Verzögerung(200);

  Motor.step(-50);    50 Schritte gegen den Uhrzeigersinn;
  Verzögerung(200);
}
```

Schaltungstitel_5: "Advanced Step Motor program"

Schaltungserklärung: Dieses Programm steuert einen Schrittmotor. Die Steuerbefehle werden über den seriellen Monitor ausgegeben.

Beispiel: Senden Sie 100 über den seriellen Monitor an den Arduino Der Schrittmotor macht 100 Schritte. Wenn Sie - 100 senden Der Schrittmotor macht 100 Schritte in die andere Richtung.

Programm:

Erweitertes Schrittmotorenprogramm

```
/*
   Verbindung:
Arduino |  MotorShielt
| PIN Batterie
-------------------------
+5V | VSIGNAL
GND | GNDSIGNAL
4 | Ena
5 | PWMA+
6 | PWMA-
8 | ENB
9 | PWMB+
10 | PWMB-
```

```
*/

//------------------------------------------------------------
//  Bibliotheken
//------------------------------------------------------------
#include <Stepper.h>
//------------------------------------------------------------
// Konstanten
//------------------------------------------------------------
#define SCHRITTE 200
//------------------------------------------------------------
// Klasse
//------------------------------------------------------------
Schrittmotor (STEPS, 5,6,9,10);
//------------------------------------------------------------
variablen
//------------------------------------------------------------

int spe = 100;
String message = "";

//------------------------------------------------------------
Setup-Funktion
//------------------------------------------------------------

void setup()
{
 Serial.begin(9600);
 pinMode(4, INPUT);
 pinMode(8, INPUT);
 digitalWrite(4 ,HIGH);
 digitalWrite(8 ,HIGH);
 Motor.setSpeed(spe);
}

void loop()
{
 if(Serial.available()>0)
 {
   message = Serial.readString();
Serial.println("\n\n --> Eingehende Nachricht: " + Nachricht );

   if(message.toInt())
   {
Serial.print(" <-- Ausgehende Nachricht: Schritte -> " + Nachricht);
    Motor.step(message.toInt());
   }
   oder
   {
Serial.println(" <<-- Eingehende Nachricht " + Nachricht + " !!! --> ist keine Zahl <-- !!! \n\n");
```

```
  }
 }

}
```

Schaltungstitel_6 (mit Servomotor): " Einfaches Servomotorprogramm "

Schaltungserklärung: In diesem Programm wird ein Servomotor durch ein PWM-Signal gesteuert . Die for-Schleife wird verwendet

```
// Simple servo Motor programm
/*
   Verbindung:
Arduino |  MotorShielt
| PIN Batterie
------------------------
+5V | VSIGNAL
GND | GNDSIGNAL
3 | PWMSERVOIN
*/
//---------------------------------------------------------------
variablen
//---------------------------------------------------------------
int del = 100;  Zeit für Verspätung

//---------------------------------------------------------------
Setup-Funktion
//---------------------------------------------------------------
void setup()
{
 Serial.begin(9600);
 pinMode(3, OUTPUT);
}

void loop()
{
for(int i = 0;  I<255;  i++)
   {
    analogWrite(3,i);
    Serial.println(i);
    Verzögerung(del);
  }

  Verzögerung(1000);
}
```

Co-funded by the Erasmus+ Programme of the European Union

Erasmus+ KA-202

Strategisches Partnerschaftsprojekt Berufsbildung

Projekttitel: "Lehren und Lernen von Arduinos in der Berufsbildung"

Projektakronym: " ARDUinVET "

Projektnummer: "2020-1-TR01-KA202-093762"

Sensormodul und Trainingskit

Sensoren

Ein Sensor ist ein Gerät, Modul oder Subsystem der Elektronik, dessen Zweck es ist, eine Schnittstelle zwischen einer Schaltung und ihrer Umgebung zu schaffen. Es ist das System, das Informationen aus der Umgebung, der systischen Welt, empfängt und diese Informationen als Wert an die Schaltung sendet. Tatsächlich erkennt es Ereignisse oder Veränderungen, die in der Umgebung passieren. Um diese Ereignisse oder Veränderungen zu erkennen, muss der Sensor bestimmte Werte in einer physikalischen Skala messen. Es misst eine Eigenschaft wie Druck, Position, Temperatur oder Beschleunigung und reagiert mit Rückmeldung. Dann wandelt es diese Werte in ein elektrisches – normalerweise – Signal um.

Daher wird ein Sensor immer mit anderer Elektronik verwendet. Sensoren finden sich in vielen Alltagsgegenständen, wie berührungs- oder bewegungsempfindlichen Geräten, Geräten, die mit Licht oder Ton betrieben werden usw. D der Einsatz von Sensoren hat sich über die traditionellen Bereiche der Temperatur-, Druck- oder Durchflussmessung, z.B. GPS-Sensoren, hinaus erweitert. Analoge Sensoren sind immer noch weit verbreitet. Zu den Anwendungen gehören Fertigung und Maschinenbau, Flugzeuge und Luft- und Raumfahrt, Autos, Medizin, Robotik und viele andere Aspekte unseres täglichen Lebens.

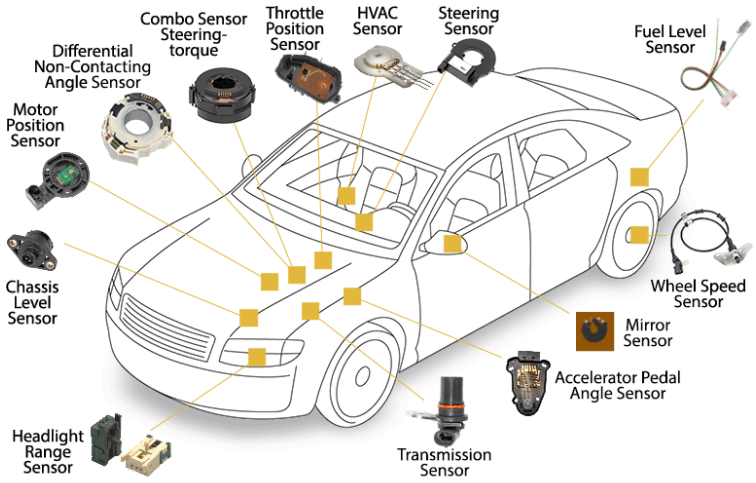


Abbildung 1. Sensoren für Kraftfahrzeuge

Sensoreigenschaften

Ein guter Sensor gehorcht folgenden Regeln:

- es muss empfindlich gegenüber der gemessenen Eigenschaft sein
- es muss unempfindlich gegenüber anderen Eigenschaften sein, die bei seiner Anwendung auftreten können
- sie darf die gemessene Eigenschaft nicht beeinflussen.

Die wichtigsten grundlegenden Eigenschaften der Sensoren sind:

- Linearität. Der Sensor verfügt über eine Eigenschaft oder ein Feature, deren Wert sich ändert. Wenn die physikalische Größe, die misst, ebenfalls geändert wird. Es ist wünschenswert, dass seine Variationen in der physikalischen Größenmessung spürbare Veränderungen in der Eigenschaft des Sensors verursachen. Diese Eigenschaft wird als Linearität bezeichnet und ist von Hauptbedeutung.
- Genauigkeit. Die Nähe des Ausgabewerts zum Eingabewert.
- Fehler: Die Differenz zwischen dem Messwert und dem Istwert.
- Toleranz: Der maximale Fehler, den der Sensor erzeugen kann.
- Full - Scale Input (FSI): Gibt an, in welchen Frames der gemessenen physikalischen Größe der Sensor verwendet werden kann
- Full - Scale Output (FSO): Legt die Werte fest, die Spannung oder Strom am Ausgang des Sensors empfangen können
- Empfindlichkeit: Es drückt aus, wie hoch das Ausgangssignal den Sensor für jede Einheit der gemessenen physikalischen Größe ausgibt.
- Auflösung: Drückt die kleinste Änderung der physischen Größe aus, die der Sensor erkennen kann, und ändert entsprechend seinen Ausgabewert.
- Hysterese: Ein Hysteresefehler bewirkt, dass der Ausgabewert in Abhängigkeit von vorherigen Eingabewerten variiert. Wenn die Ausgabe eines Sensors unterschiedlich

ist, je nachdem, ob ein bestimmter Eingangswert durch Erhöhen oder Verringern des Eingangs erreicht wurde, hat der Sensor einen Hysteresefehler.

- Verzögerung: Dies ist die Verzögerung der Änderung des Ausgabewerts nach einer Eingangsänderung.
- Totzone: Die maximale Änderung des Eingabewerts, die sich nicht auf den Ausgabewert auswirkt.

Sensor-Kategorien

Es gibt mehrere Möglichkeiten, Sensoren zu kategorisieren, von denen einige unten aufgeführt sind.

Die erste kategorisiert die Sensoren nach der Form des Ausgangswertes und teilt die analogen und digitalen Sensoren.

Bei der zweiten Kategorisierung geht es darum, was ein Sensor messen kann, mit einer deutlicheren Unterscheidung zwischen natürlichen und chemischen Sensoren. Natürliche Sensoren steuern physikalische Größen wie Ort, Masse, Strom, Zeit und ihre relative Größe, während chemische Sensoren das Vorhandensein verschiedener Gase in einer bestimmten Atmosphäre steuern.

Ein anderer Weg bezieht sich auf Materialien, deren physikalische Eigenschaften der Sensor funktioniert, mit Hauptkategorien Sensoren mit leitfähigen, Halbleiter-, dielektrischen, magnetischen und supraleitenden Materialien.

Schließlich bezieht sich eine weitere Klassifizierungsmethode auf das Einsatzgebietdes Sensors mit Hauptkategorien wie industrielle, medizinische, militärische, Umweltsensoren sowie Sensoren für Transport- und Automatisierungsanwendungen.

Abbildung 2: Verschiedene Sensoren

Grundlegende Sensoren

Hier sind einige grundlegende Sensoren für gängige Anwendungen:

Fotowiderstand LDR: Der Widerstand ist ein variabler Widerstand, dessen Wert sich je nach Licht, das auf ihn fällt, ändert.

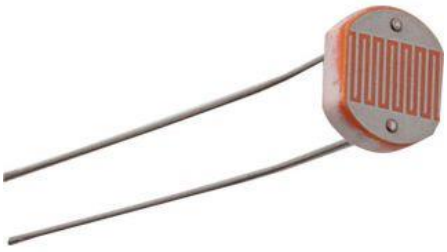


Abbildung 3: Fotowiderstand

Digitaler Leuchtkraft- / Lux- / Lichtsensor: Der TSL2561 Leuchtkraftsensor ist ein fortschrittlicher digitaler Lichtsensor, ideal für den Einsatz in einer Vielzahl von Lichtsituationen. Dieser Sensor ist präziser, ermöglicht exakte Lux-Berechnungen und kann für verschiedene Verstärkungs-/Timing-Bereiche konfiguriert werden, um Lichtbereiche von bis zu 0,1 - 40.000+ Lux im laufenden Betrieb zu erkennen. Es enthält sowohl Infrarot- als auch Vollspektrumdioden! Das bedeutet, dass Sie Infrarot-, Vollspektrum- oder vom Menschen sichtbares Licht separat messen können.

Abbildung 3: Digitaler Helligkeits-/Lux-/Lichtsensor

Temperatursensor (mit Analogausgang): Der Temperaturbereich, den er misst, beträgt üblicherweise -40°C bis +100°C mit einer Genauigkeit von ± 1°C.

Abbildung 4: Temperatursensor

Feuchtigkeits- und Temperatursensor: hochgenauer, digitaler Temperatur- und Feuchtigkeitssensor. Es verfügt über einen Messbereich von 0-100% RH mit einer Temperaturgenauigkeit von +/- 0,3 °C @ 25 °C.

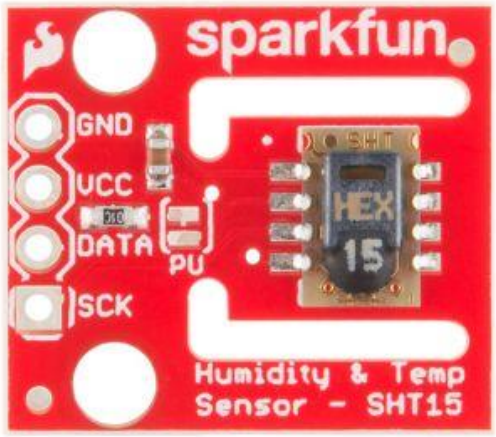


Abbildung 5: Feuchtigkeits- und Temperatursensor

Infrarotsensor: Wird zur Entfernungsberechnung verwendet. Die Entfernung, die Sie berechnen können, ist von 2cm. bis zu 400cm. mit einer Genauigkeit von einem Zentimeter.

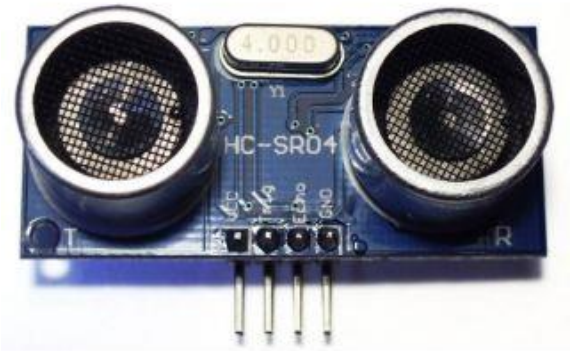


Abbildung 6: Ultraschallsensor

Bewegungssensor: Es hat die Fähigkeit, die Bewegung eines Menschen oder eines Haustieres innerhalb eines Raumes innerhalb von sechs Metern zu erkennen. Der Sensor verfügt über zwei Trimmerwiderstände, bei denen die Empfindlichkeit und die Aktivierungszeit ab dem Moment, in dem er eine Bewegung erkennt, eingestellt werden können.

Abbildung 7: Bewegungsmelder

Vibrationssensor: Wenn der Sensor Vibrationen erkennt, wird eine Spannung erzeugt, die einen Widerstand von 1 Mohm verwendet

Abbildung 8: Vibrationssensor

Dreiachsen-Beschleunigungsmesser und Gyro-Breakout-Sensor: Durch die Kombination eines 3-Achsen-Gyroskops und eines 3-Achsen-Beschleunigungsmessers auf demselben Siliziumchip zusammen mit einem integrierten Digital Motion Processor (DMP), der komplexe 9-Achsen-Motion-Fusion-Algorithmen verarbeiten kann, kann der Sensor alle Achsenausrichtungswerte zurückgeben.

Abbildung 9: Dreiachsen-Beschleunigungsmesser und Gyro-Breakout-Sensor

Gassensor: Empfindlich für LPG, Erdgas, Kohlengas. Die Ausgangsspannung steigt mit zunehmender Konzentration der gemessenen Gase.

Abbildung 11: Gassensor

Schalldetektor: Dieser Sensor bietet einen Audioausgang, aber auch eine binäre Anzeige des Vorhandenseins von Schall und eine analoge Darstellung seiner Amplitude.

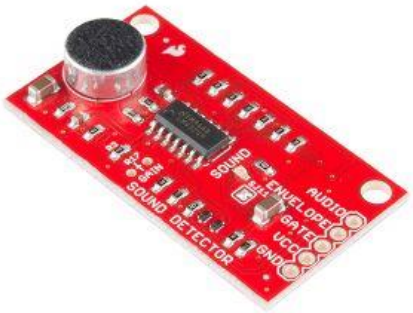


Abbildung 12: Schalldetektor

PROJEKT 1- NAME: ULTRASONIC SENSOR IM VERKEHRSZEICHEN

Das Ziel des Projekts: In diesem Projekt werden die Studierenden sehen, wie wir einen Ultraschallsensor in einer Ampel mit einem externen Pull-up-Widerstand einsetzen können.

Durch dieses Projekt können die Schüler mit einem Ultraschallsensor experimentieren, einem Gerät, das die Entfernung zu einem Objekt mithilfe von Schallwellen misst.

Methodik

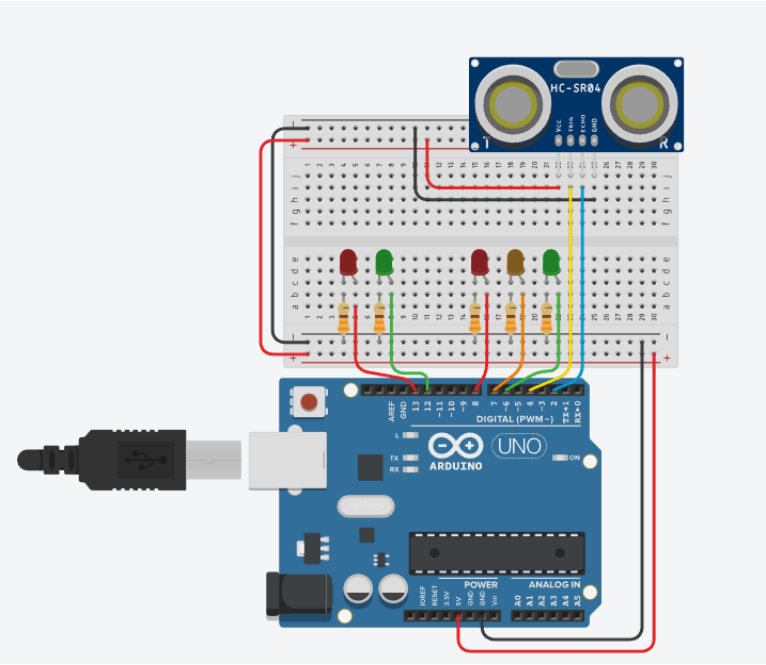
Zunächst beschreiben wir den gewünschten Workflow des Projekts. Dann bitten wir die Schüler, die benötigten Komponenten auszuwählen und Tinkercad zu verwenden, um die Schaltung zu entwerfen und zu zeichnen. Sie werden das Tinkercad Code Tool verwenden, um den Code zu schreiben.

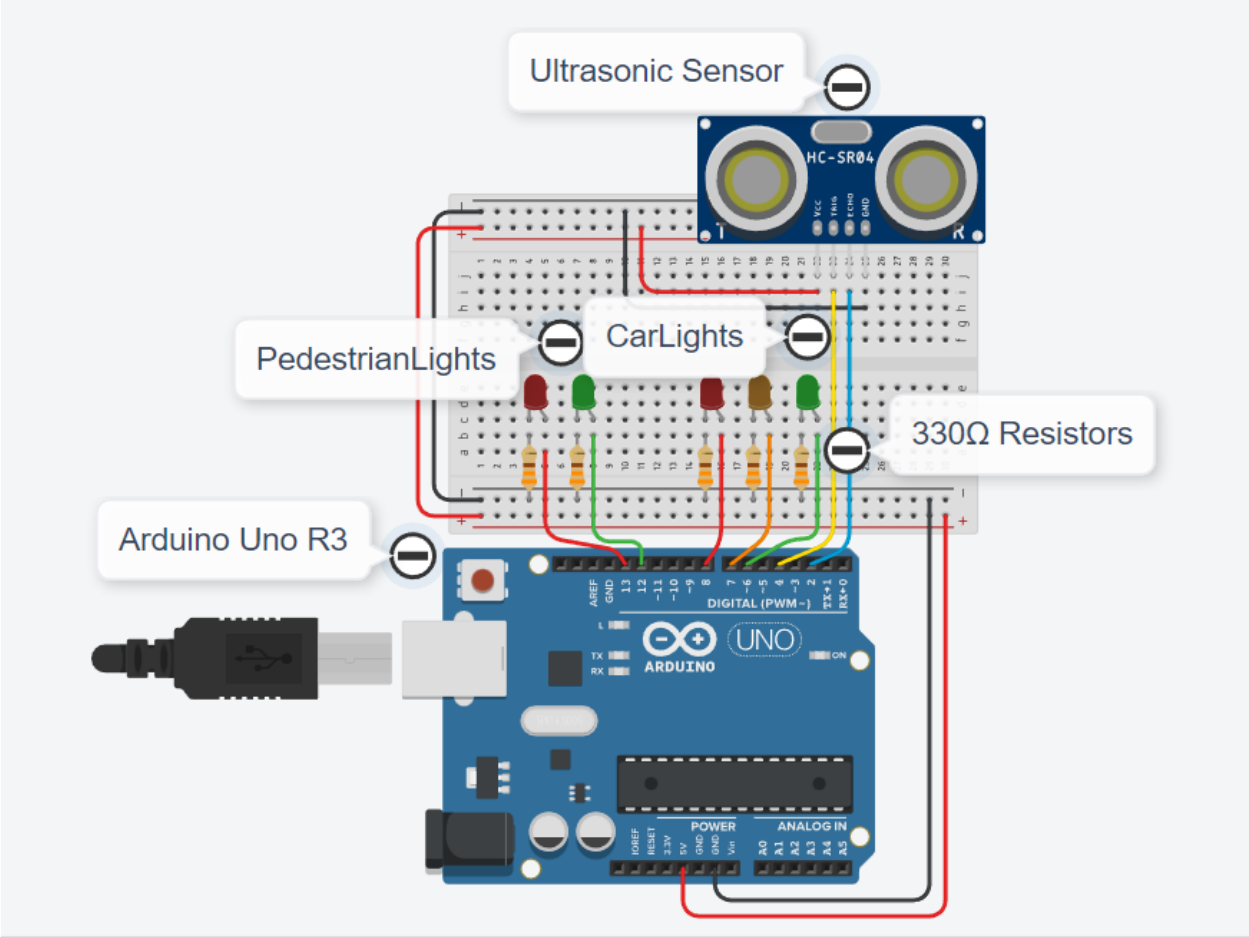
Dann erstellen sie die Schaltung, schreiben den Code auf das Arduino-Softwaretool und laden ihn auf das Arduino-Board hoch.

Die Simulation auf Tinkercad überprüft, ob die Schaltung korrekt aufgebaut wurde.

Dieses Projekt richtet sich an Grundschüler, die beginnen, von den Ergebnissen unseres Erasmus+ KA-202 Strategic Partnerships in VET Project namens "ArduinVET" zu profitieren.

Projektschaltung.





So funktioniert's:

Der Ultraschallsensor sendet, indem er eine Schallwelle mit einer Ultraschallfrequenz sendet und darauf wartet, dass sie vom Objekt zurückprallt. Die Zeitverzögerung zwischen der Übertragung des Schalls und dem Empfang des Schalls wird dann verwendet, um die Entfernung zu berechnen.

Die Schüler können den Sensor aktivieren, indem sie ihre Hand vor den Sensor legen, wodurch die Zeitverzögerung zwischen Tonübertragung und Tonempfang abnimmt. Die Entfernung wird auf dem seriellen Bildschirm angezeigt. Die Schüler können die Aktivierung der Ampeln sehen, indem sie die Ampeln beobachten.

In der Schaltung verwenden wir zusammen mit dem Ultraschallsensor einen externen Pull-up-Widerstand.

- Der Pull-up-Widerstand hält Pin 3 dauerhaft im HIGH (+5V) Zustand.
- Wenn der Ultraschallsensor aktiviert ist, ist Pin 3 kurzzeitig geerdet (LOW State).

Timing-Tabelle

Phasen	Fahrzeuge	Fußgänger	Mal	Beschreibung
0				Der Sensor wurde nicht aktiviert
1			3Sek.	Sobald der Sensor aktiviert ist, werden die Phasen 1 bis 4 aktiviert und dann kehrt das Gerät in seinen vorherigen Zustand zurück (Phase 0).
2			3Sek.	
3			4 Sek.	
4			3Sek.	
0				Bis der Sensor wieder aktiviert wird

Komponentenliste:

Unsere Komponenten sind:

- Arduino Vorstand
- Ultraschallsensor HC-SR04
- Steckbrett und Sprungdrähte
- Kabel-USB
- Arduino UNO R3
- 5XLED
- 5X330 ohm

Arduino Code des Projekts:

Projektname: ULTRASOVIC SENSOR IN AMPELN
Ampel mit Fußgängerlicht und Ultraschallsensor

```
Deklaration von Konstanten
  const byte greenCar = 6;
  const byte orangeCar = 7;
  const byte redCar = 8;
  const byte greenPedestrian = 12;
  const byte redPedestrian = 13;
   Parameterdeklaration
  boolesche SchaltflächeOn=false;

 Parameterdeklaration Ultraschallsensor HC-SR 04 * ******
definiert PIN-Nummern
  const int trigPin = 4;  D4
  const int echoPin = 2;  D3


void setup() {
Konstanten initialisieren Ultraschallsensor HC-SR 04 * ******
  pinMode(trigPin, OUTPUT); Setzt den trigPin als Ausgang
  pinMode(echoPin, INPUT); Legt den echoPin als Eingang fest
  Serial.begin(9600); Startet die serielle Kommunikation
   Parameterinitialisierung
pinMode(greenCar,OUTPUT);
  pinMode(orangeCar,OUTPUT);
  pinMode(redCar, OUTPUT);

  pinMode(greenPedestrian,OUTPUT);
  pinMode(redPedestrian,OUTPUT);

  Initialisierung der Preise für Fußgängersignale
  digitalWrite(greenPedestrian,LOW);
  digitalWrite(redPedestrian,HIGH);

  Initialisierung von Preisen für Fahrzeugsignalisierung
  digitalWrite(greenCar, HIGH);
  digitalWrite(orangeCar,LOW);
  digitalWrite(redCar,LOW);
  pinMode (2, INPUT_PULLUP);
}

void loop() {
 * 3. Code Ultraschallsensor HC-SR04 * ******
 Löscht den trigPin
  Verzögerung(500);
  digitalWrite(trigPin, LOW);
  VerzögerungMikrosekunden(2);
```

```
Setzt den trigPin für 10 Mikrosekunden auf den Zustand HIGH
  digitalWrite(trigPin, HIGH);
  VerzögerungMikrosekunden(10);
  digitalWrite(trigPin, LOW);

Liest den echoPin, gibt die Schallwellenlaufzeit in Mikrosekunden zurück
  const lange Dauer = pulseIn(echoPin, HIGH);
  Serial.print("Dauer: ");
  Serial.println(duration);
Berechnung der Entfernung
  const Langstrecke= Dauer/58,2;
Zeigt die Entfernung auf dem seriellen Monitor aus
  Serial.print("Entfernung: ");
  Serial.println(distance);

  Verzögerung (2000);
  if (distance<20){
     buttonOn=true;
  }
  if(buttonOn == true)
    {
     buttonOn = false;

     Verzögerung(3000);
     digitalWrite(greenCar, LOW);
     digitalWrite(orangeCar,HIGH);

     Verzögerung(3000);
     digitalWrite(orangeCar,LOW);
     digitalWrite(redCar, HIGH);

     digitalWrite(greenPedestrian,HIGH);
     digitalWrite(redPedestrian,LOW);

     Verzögerung(4000);
     digitalWrite(greenPedestrian,LOW);
     digitalWrite(redPedestrian,HIGH);

     Verspätung(3000);
     digitalWrite(greenCar, HIGH);
     digitalWrite(redCar,LOW);

     Verzögerung(5000);
    }else{
     digitalWrite(greenPedestrian,LOW);
     digitalWrite(redPedestrian,HIGH);
```

```
    digitalWrite(greenCar, HIGH);
    digitalWrite(orangeCar,LOW);
    digitalWrite(redCar,LOW);
    }
}
```

PROJEKT 2 - BEZEICHNUNG: ALARMANLAGE

Das Ziel des Projekts: In diesem Projekt sehen die Studierenden, wie ein Alarmsystem funktioniert, das Flüssiggas, Bewegungen in einem geschlossenen Raum sowie ungerechtfertigten Temperaturanstieg erkennen kann. Die Alarmanlage wird über eine Taste und einen Fotowiderstandssensor gesteuert . Durch Drücken der Taste können die Schüler die Alarmanlage aktivieren. Wenn der Summer durch Drücken der Taste ertönt, können sie ihn stoppen. Wenn sie einen aktivierten Sensor per Knopfdruck verbinden, können sie das Alarmsystem deaktivieren. Der Fotowiderstand aktiviert das Alarmsystem, falls das Alarmsystem nicht aktiviert ist und die Beleuchtung schwach ist.

Durch dieses Projekt konnten die Schüler experimentieren mit:

- ✓ Ein aktives Summersensormodul mit einer integrierten Oszillatorschaltung, die eine konstante Schallfrequenz erzeugt. Es schaltet sich mit einem Arduino-Pin ein und aus
- ✓ Ein Ultraschallsensor, ein Gerät, das die Entfernung zu einem Objekt mithilfe von Schallwellen misst
- ✓ einen Gassensor und
- ✓ zwei Kontrollleuchten und eine Taste.
- ✓ ein Fotowiderstandssensor
- ✓ einen Temperatursensor

Methodik

Zunächst beschreiben wir den gewünschten Workflow des zu entwickelnden Projekts in Schritten. Die Schüler werden dann gebeten, die notwendigen Komponenten auszuwählen, um die Schaltung zu entwerfen und zu zeichnen. Sie verwenden die Anwendung "Arduino", um den Code zu schreiben.

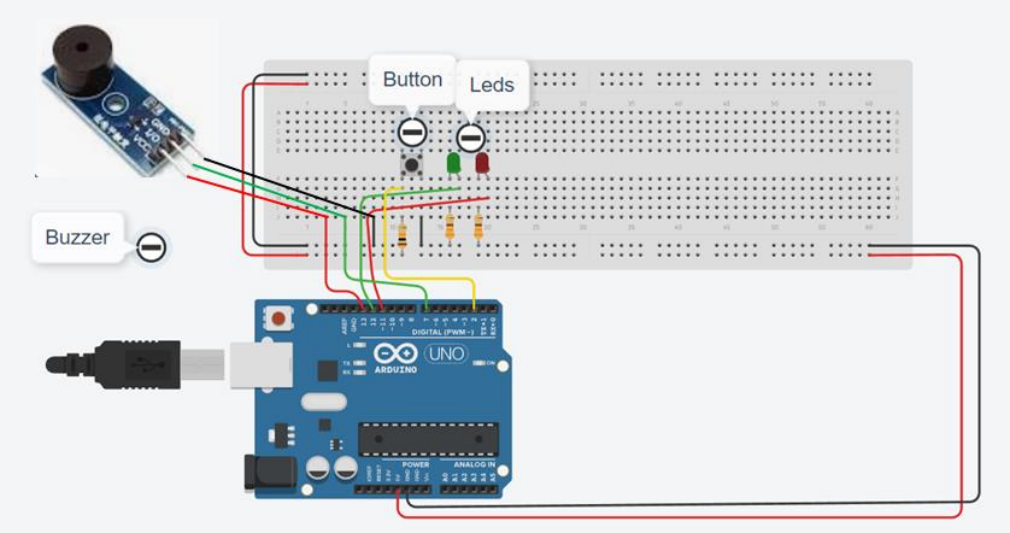
Dann erstellen sie die Schaltung, schreiben den Code auf das Arduino-Softwaretool und laden ihn auf das Arduino-Board hoch.

Die Schüler werden überprüfen, ob die Schaltung korrekt konstruiert wurde.

Dieses Projekt richtet sich an Grundschüler, die beginnen, von den Ergebnissen unseres Erasmus+ KA-202 Strategic Partnerships in VET Project namens "ArduinVET" zu profitieren.

Schritt 1 (Summer mit unterschiedlichem Tonton im Alarmsystem)

Projektschaltung.



So funktioniert's:

Eine aktive Summersensoreinheit verfügt über einen eingebauten Schwingkreis, so dass die Frequenz des Schalls konstant ist. Es ist in der Lage, den Klang selbst zu erzeugen.
Innerhalb dieser Arbeit hört der Klangton durch Code auf, den gleichen Klang zu haben. Sobald der Knopf gedrückt wird, beginnt er mit einem Arduino-Pin zu piepen, die rote LED, die wartet, schaltet sich aus und die grüne LED schaltet sich ein

In der Schaltung verwenden wir zusammen mit dem Ultraschallsensor einen externen Pull-up-Widerstand.

- Der Pull-up-Widerstand hält Pin 3 dauerhaft im HIGH (+5V) Zustand.
- Wennder Knopf gedrückt wird, ist Pin 3 kurzzeitig geerdet (LOW State).

Komponentenliste:

Unsere Komponenten sind:

- Summer
- Steckbrett und Sprungdrähte
- Kabel-USB
- Arduino UNO R3
- 2XLED
- 2X330 ohm
- Button
- 1X10Kohm

Arduino-Code von Schritt 1:

Projektname: Alarmanlage

Schritt 1 (Summer mit unterschiedlichem Klang)

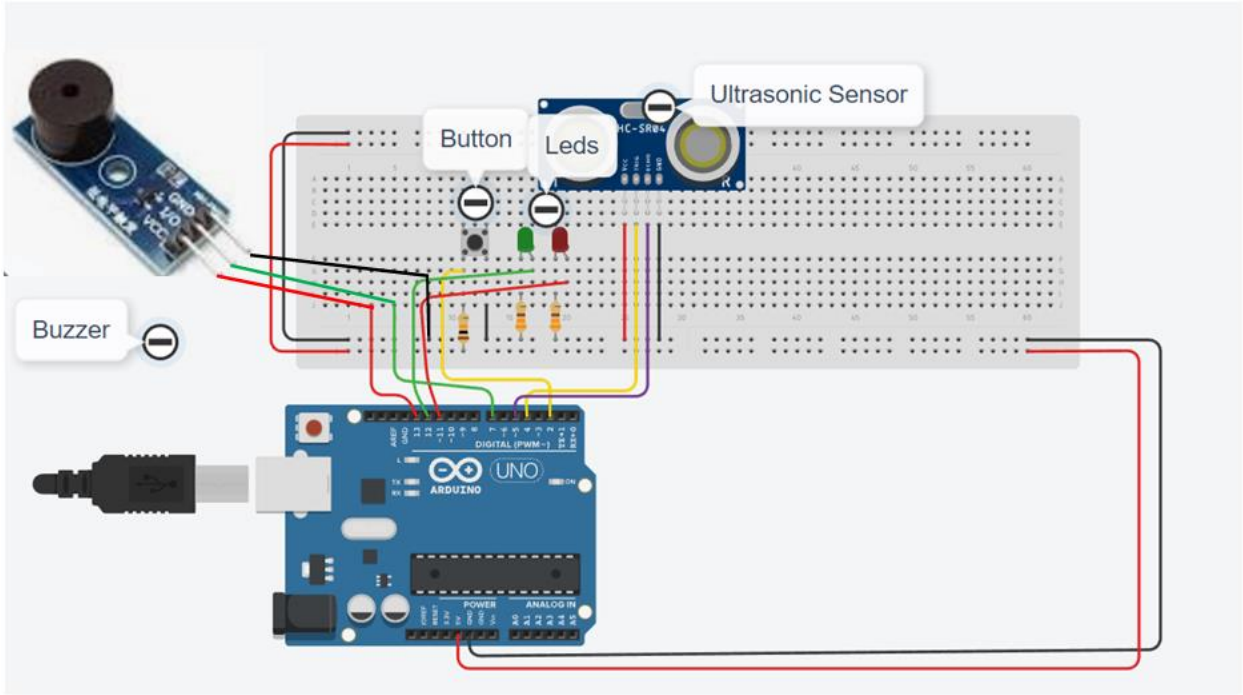
```
Deklarationsdefinition von Variablen - Konstanten
int buzz= 7; // I/O-Pin vom Summer Modul verbindet sich hier
int buzzVcc= 13; //Vcc-Pin vom Summer Modul verbindet sich hier
const int wpm = 20; Morsegeschwindigkeit in WPM
const int dotL = 1200/wpm; Berechnete Punktlänge
const int dashL = 3*dotL; Bindestrich = 3 x Punkt
const int sPause = dotL; Symbolpause = 1 Punkt
const int lPause = dashL; Buchstabenpause = 3 Punkte
const int wPause = 7*dotL; Wortpause = 7 Punkte

int red_led=12;
int green_led=11;

boolesche SchaltflächeOn=false;
void setup()
{
 Initialisierung von Variablen - Konstanten
```

```
  pinMode(buzz,OUTPUT);
  pinMode(buzzVcc,OUTPUT);

  pinMode(red_led,OUTPUT);
  pinMode(green_led,OUTPUT);

  pinMode (2, INPUT_PULLUP);
}

void loop(){
  if (digitalRead(2)== LOW){
     buttonOn=true;
     }

    Summer
    if(buttonOn==true){

     digitalWrite(red_led, HOCH);
     digitalWrite(green_led, HOCH);

     digitalWrite(buzz, LOW); Ton EIN
     Verzögerung(dashL); Tonlänge
     digitalWrite(buzz, HIGH); Ton AUS
     delay(sPause); Symbolpause
     digitalWrite(buzzVcc, HIGH);
     digitalWrite(buzz, LOW); Ton EIN
     Verzögerung (dotL); Tonlänge
     digitalWrite(buzz, HIGH); Ton AUS
     delay(sPause); Symbolpause

     digitalWrite(buzz, LOW); Ton EIN
     Verzögerung(dashL); Tonlänge
     digitalWrite(buzz, HIGH); Ton AUS
     delay(sPause); Symbolpause

     digitalWrite(buzz, LOW); Ton EIN
     Verzögerung (dotL); Tonlänge
     digitalWrite(buzz, HIGH); Ton AUS
     delay(sPause); Symbolpause

     delay(lPause-sPause); Subtrahiert bereits genommene Pause

     digitalWrite(buzz, LOW); Ton EIN
     Verzögerung(dashL); Tonlänge
     digitalWrite(buzz, HIGH); Ton AUS
     delay(sPause); Symbolpause
```

```
    digitalWrite(buzz, LOW); Ton EIN
    Verzögerung(dashL); Tonlänge
    digitalWrite(buzz, HIGH); Ton AUS
    delay(sPause); Symbolpause

    digitalWrite(buzz, LOW); Ton EIN
    Verzögerung (dotL); Tonlänge
    digitalWrite(buzz, HIGH); Ton AUS
    delay(sPause); Symbolpause

    digitalWrite(buzz, LOW); Ton EIN
    Verzögerung(dashL); Tonlänge
    digitalWrite(buzz, HIGH); Ton AUS
    delay(sPause); Symbolpause
    delay(wPause-sPause); Subtrahiert bereits genommene Pause
    Verzögerung(5000);
    Variable kehrt in ihren ursprünglichen Status zurück
    buttonOn=false;
    digitalWrite(buzz, LOW); Ton EIN
    digitalWrite(buzzVcc, LOW);
    digitalWrite(red_led, LOW);
    digitalWrite(green_led, LOW);
   }
}
```

Schritt 2 (Ultraschallsensor HC-SR04 im Alarmsystem hinzugefügt)

Schließen Sie den Ultraschallsensor HC-SR04 an



So funktioniert's:

Der Ultraschallsensor sendet eine Schallwelle mit einer Ultraschallfrequenz und wartet darauf, vom Objekt abzuprallen. Die Zeitverzögerung zwischen der Übertragung des Schalls und dem Empfang des Schalls wird dann verwendet, um die Entfernung zu berechnen. Die Schüler können das Alarmsystem durch Drücken der Taste aktivieren, dann schaltet sich die grüne LED ein, und wenn sie ihre Hand vor den Sensor legen, verringert sich die Zeitverzögerung zwischen Tonübertragung und Tonempfang und die grüne LED schaltet sich aus, schaltet die rote LED ein und der Audiosummer beginnt zu klingen. Die Entfernung wird auf dem seriellen Bildschirm angezeigt. Die Schüler können die Alarmanlage auch deaktivieren, indem sie die Taste erneut drücken.

Arduino-Code von Schritt 2:

Code von Schritt 2 - Ultraschallsensor HC-SR04 hinzugefügt
int red_led=12;
int green_led=11;
int sensorThres=400;

Summer
int buzz= 13; I/O-Pin vom Summer wird hier angeschlossen
int buzzVcc= 7; Vcc-Pin vom Summer verbindet hier
const int wpm = 20; Morsegeschwindigkeit in WPM
const int dotL = 1200/wpm; Berechnete Punktlänge
const int dashL = 3*dotL; Bindestrich = 3 x Punkt
const int sPause = dotL; Symbolpause = 1 Punkt

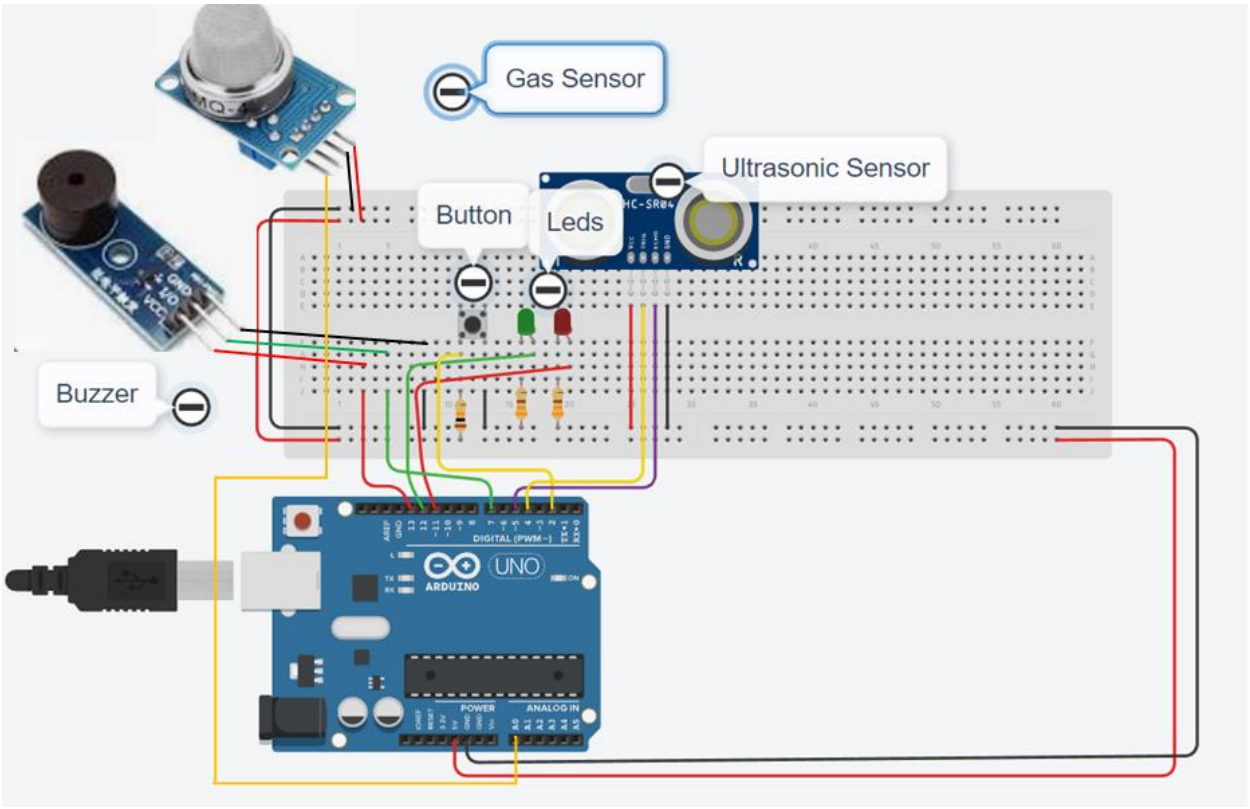
```
const int lPause = dashL; Buchstabenpause = 3 Punkte
const int wPause = 7*dotL; Wortpause = 7 Punkte

Ultraschallsensor
#define trigPin 4
#define echoPin 5

Knopf
boolesche SchaltflächeOn=false;

void setup()
{
pinMode(red_led,OUTPUT);
pinMode(buzz,OUTPUT);
pinMode(green_led,OUTPUT);

pinMode(buzz,OUTPUT); Summer-Pin als Ausgang festlegen
pinMode(buzzVcc,OUTPUT); VCC-Summer

Ultraschallsensor
pinMode(trigPin, OUTPUT);
pinMode(echoPin, INPUT);

pinMode (2, INPUT_PULLUP);

Serial.begin(9600);
}



void loop()
{
Knopf
 if (digitalRead(2)== LOW){
    buttonOn=true;
    digitalWrite(green_led, HOCH);
    }

Ultrasoc-Sensorcode
lange Dauer, Entfernung;
digitalWrite(trigPin, LOW);
VerzögerungMikrosekunden(2);
digitalWrite(trigPin, HIGH);
VerzögerungMikrosekunden(10);
digitalWrite(trigPin, LOW);
Dauer = pulseIn(echoPin, HIGH);
```

```
Entfernung = (Dauer/2) / 29,1;

digitalWrite(buzzVcc, LOW);

if (Entfernung < 20)
 {
 while(buttonOn==true){

  if (digitalRead(2)== LOW){
    buttonOn=false;
    digitalWrite(red_led, LOW);
    digitalWrite(green_led, LOW);
    digitalWrite( buzz, LOW);
 }else{
 digitalWrite(red_led, HOCH);
 digitalWrite(green_led, LOW);
 digitalWrite( buzz, HIGH);



 Summer
 digitalWrite(buzzVcc, HIGH);
 digitalWrite(buzz, LOW); Ton EIN
 Verzögerung(dashL); Tonlänge
 digitalWrite(buzz, HIGH); Ton AUS
 delay(sPause); Symbolpause

 digitalWrite(buzz, LOW); Ton EIN
 Verzögerung (dotL); Tonlänge
 digitalWrite(buzz, HIGH); Ton AUS
 delay(sPause); Symbolpause

 digitalWrite(buzz, LOW); Ton EIN
 Verzögerung(dashL); Tonlänge
 digitalWrite(buzz, HIGH); Ton AUS
 delay(sPause); Symbolpause

 digitalWrite(buzz, LOW); Ton EIN
 Verzögerung (dotL); Tonlänge
 digitalWrite(buzz, HIGH); Ton AUS
 delay(sPause); Symbolpause

 delay(lPause-sPause); Subtrahiert bereits genommene Pause

 digitalWrite(buzz, LOW); Ton EIN
 Verzögerung(dashL); Tonlänge
```

```
 digitalWrite(buzz, HIGH); Ton AUS
 delay(sPause); Symbolpause
 digitalWrite(buzz, LOW); Ton EIN
 Verzögerung(dashL); Tonlänge
 digitalWrite(buzz, HIGH); Ton AUS
 delay(sPause); Symbolpause

 digitalWrite(buzz, LOW); Ton EIN
 Verzögerung (dotL); Tonlänge
 digitalWrite(buzz, HIGH); Ton AUS
 delay(sPause); Symbolpause

 digitalWrite(buzz, LOW); Ton EIN
 Verzögerung(dashL); Tonlänge
 digitalWrite(buzz, HIGH); Ton AUS
 delay(sPause); Symbolpause
 delay(wPause-sPause); Subtrahiert bereits genommene Pause

  } //end else
  } //end while
} //end wenn

Verzögerung(100);
}
```

Schritt 3 (Gassensor im Alarmsystem hinzugefügt)

Schließen Sie den Gassensor an



So funktioniert's:

Wenn der Gasdetektor Gas erkennt und das Alarmsystem bereits aktiviert ist, erlischt das grüne Licht, das rote Licht leuchtet ein und der Summer ertönt.

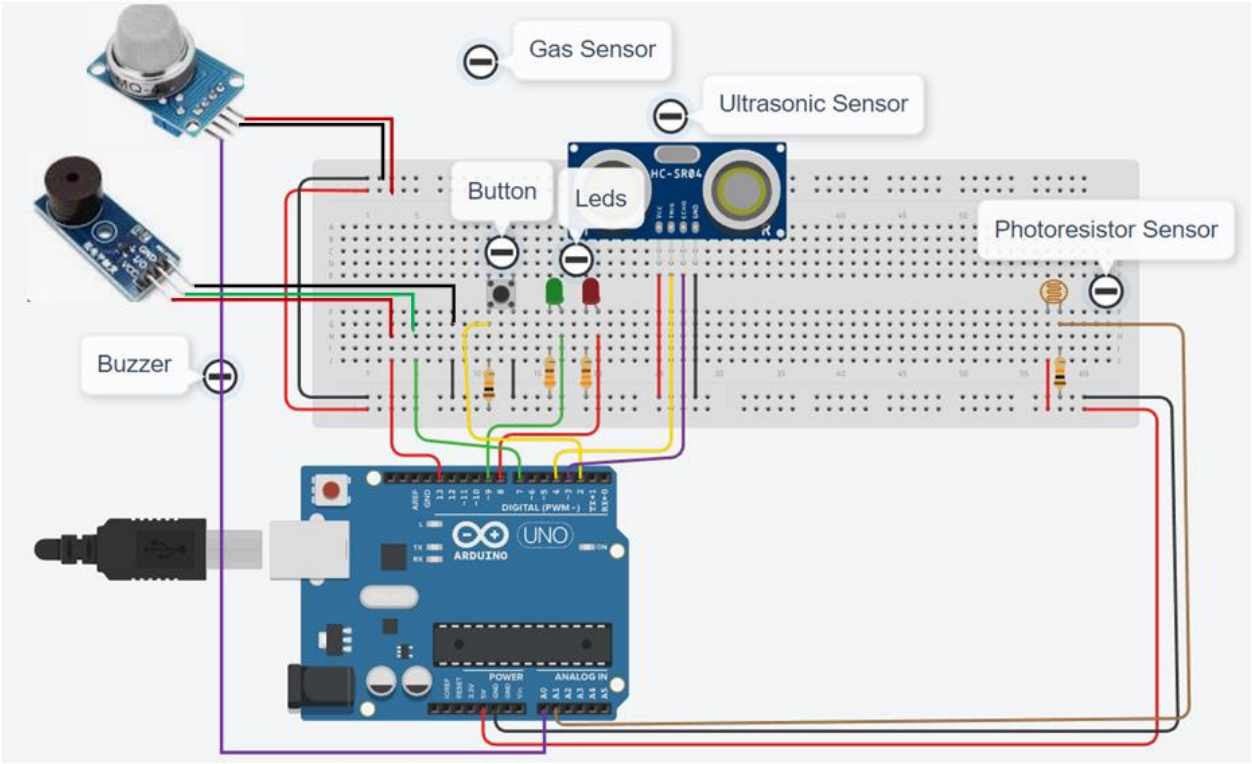
Arduino-Code von Schritt 3:

```
ARBEIT : ALARMANLAGE
GASSENSOR IM SYSTEM HINZUGEFÜGT
int red_led=11;
int green_led=12;
int gas_value;
int sensorThres=400;
```

```
Summer
int buzz= 13; I/O-Pin vom Summer wird hier angeschlossen
int buzzVcc= 7; Vcc-Pin vom Summer verbindet hier
const int wpm = 20; Morsegeschwindigkeit in WPM
const int dotL = 1200/wpm; Berechnete Punktlänge
const int dashL = 3*dotL; Bindestrich = 3 x Punkt
const int sPause = dotL; Symbolpause = 1 Punkt
const int lPause = dashL; Buchstabenpause = 3 Punkte
const int wPause = 7*dotL; Wortpause = 7 Punkte

Ultraschallsensor
#define trigPin 4
#define echoPin 5

Knopf
boolesche SchaltflächeOn=false;

void setup()
{
pinMode(red_led,OUTPUT);
pinMode(green_led,OUTPUT);
pinMode(A0;INPUT); A0 Gass

pinMode(buzz,OUTPUT); Summer-Pin als Ausgang festlegen
pinMode(buzzVcc,OUTPUT); VCC-Summer

Ultraschallsensor
pinMode(trigPin, OUTPUT);
pinMode(echoPin, INPUT);

pinMode (2, INPUT_PULLUP);

Serial.begin(9600);
}

void loop()
{
Knopf
 if (digitalRead(2)== LOW){
    buttonOn=true;
    digitalWrite(green_led, HOCH);
    Serial.println("Das Alarmsystem ist aktiviert!");
    }

   Ultrasoc-Sensorcode
```

```
lange Dauer, Entfernung;
digitalWrite(trigPin, LOW);
VerzögerungMikrosekunden(2);
digitalWrite(trigPin, HIGH);
VerzögerungMikrosekunden(10);
digitalWrite(trigPin, LOW);
Dauer = pulseIn(echoPin, HIGH);
Entfernung = (Dauer/2) / 29,1;


digitalWrite(buzzVcc, LOW);

gas_value=analogRead(A0);

if (gas_value > sensorThres oder Abstand < 20)
 {
  while(buttonOn==true){
  if (digitalRead(2)== LOW){
    buttonOn=false;
    digitalWrite(red_led, LOW);
    digitalWrite(green_led, LOW);
    digitalWrite( buzz, LOW);
    Serial.println("Das Alarmsystem ist deaktiviert!");
 }else{
 digitalWrite(red_led, HOCH);
 digitalWrite(green_led, LOW);
 digitalWrite( buzz, HIGH);
 Serial.println("DANGER!!!!");
 Serial.println(gas_value);

 Summer
 digitalWrite(buzzVcc, HIGH);
 digitalWrite(buzz, LOW); Ton EIN
 Verzögerung(dashL); Tonlänge
 digitalWrite(buzz, HIGH); Ton AUS
 delay(sPause); Symbolpause

 digitalWrite(buzz, LOW); Ton EIN
 Verzögerung (dotL); Tonlänge
 digitalWrite(buzz, HIGH); Ton AUS
 delay(sPause); Symbolpause

 digitalWrite(buzz, LOW); Ton EIN
 Verzögerung(dashL); Tonlänge
 digitalWrite(buzz, HIGH); Ton AUS
 delay(sPause); Symbolpause
```

```
digitalWrite(buzz, LOW); Ton EIN
Verzögerung (dotL); Tonlänge
digitalWrite(buzz, HIGH); Ton AUS
delay(sPause); Symbolpause

delay(lPause-sPause); Subtrahiert bereits genommene Pause

digitalWrite(buzz, LOW); Ton EIN
Verzögerung(dashL); Tonlänge
digitalWrite(buzz, HIGH); Ton AUS
delay(sPause); Symbolpause
digitalWrite(buzz, LOW); Ton EIN
Verzögerung(dashL); Tonlänge
digitalWrite(buzz, HIGH); Ton AUS
delay(sPause); Symbolpause

digitalWrite(buzz, LOW); Ton EIN
Verzögerung (dotL); Tonlänge
digitalWrite(buzz, HIGH); Ton AUS
delay(sPause); Symbolpause

digitalWrite(buzz, LOW); Ton EIN
Verzögerung(dashL); Tonlänge
digitalWrite(buzz, HIGH); Ton AUS
delay(sPause); Symbolpause
delay(wPause-sPause); Subtrahiert bereits genommene Pause
      } //end else
  }//end while
 } //end wenn

Verzögerung(100);
}
```



Schritt 4 (Fotowiderstandssensor im Alarmsystem hinzugefügt)

Schließen Sie den Fotowiderstandssensor an

So funktioniert's:

Die Alarmanlage wird aktiviert, wenn die Helligkeit abnimmt.

Arduino-Code von Schritt 4:

```
ARBEIT : ALARMANLAGE
FOTOWIDERSTANDSSENSOR IM SYSTEM HINZUGEFÜGT
int red_led=8;
int green_led=9;
int gas_value;
```

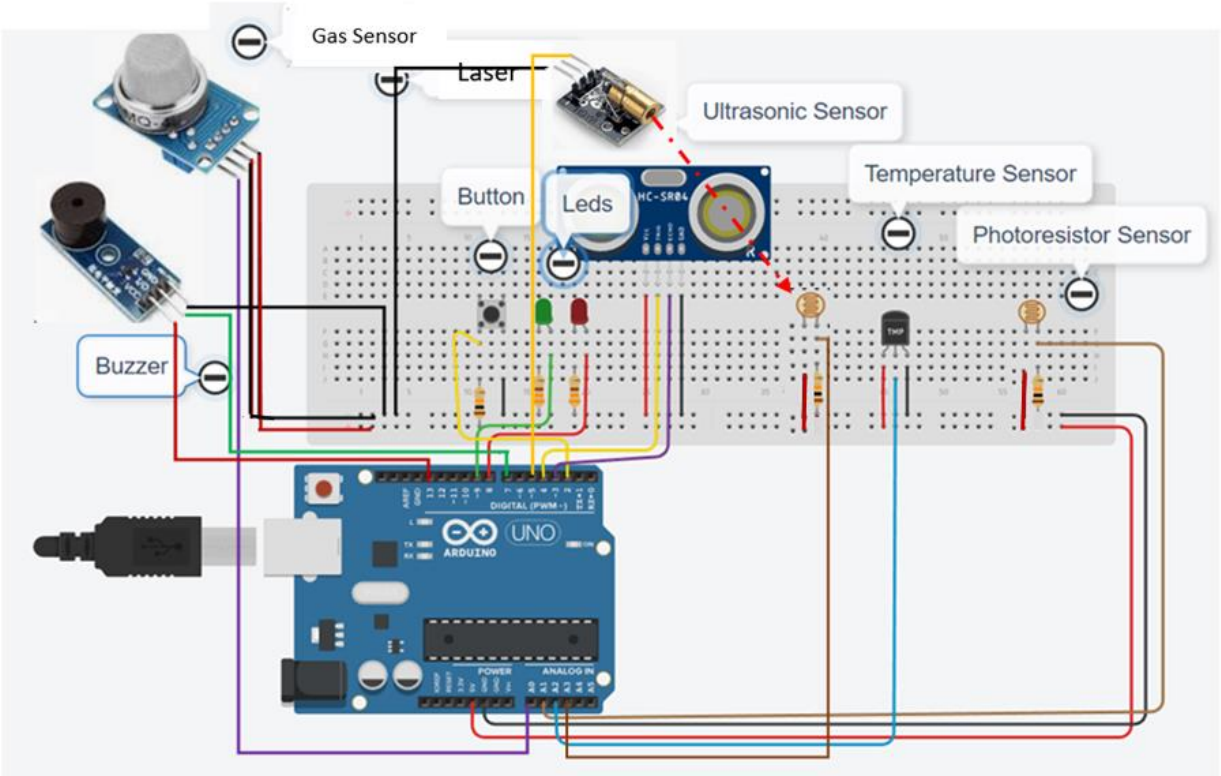
```
int sensorThres=400;

------------------Summer-----------------------
int buzz= 13; I/O-Pin vom Summer wird hier angeschlossen
int buzzVcc= 7; Vcc-Pin vom Summer verbindet hier
const int wpm = 20; Morsegeschwindigkeit in WPM
const int dotL = 1200/wpm; Berechnete Punktlänge
const int dashL = 3*dotL; Bindestrich = 3 x Punkt
const int sPause = dotL; Symbolpause = 1 Punkt
const int lPause = dashL; Buchstabenpause = 3 Punkte
const int wPause = 7*dotL; Wortpause = 7 Punkte

------------Ultraschallsensor--------------------
#define trigPin 4
#define echoPin 5

------------------Knopf-------------------------
boolesche SchaltflächeOn=false;

void setup()
{
pinMode(red_led,OUTPUT);
pinMode(green_led,OUTPUT);
digitalWrite(red_led, LOW);
digitalWrite(green_led, LOW);
pinMode(A0;INPUT); A0 Gass

pinMode(buzz,OUTPUT); Summer-Pin als Ausgang festlegen
pinMode(buzzVcc,OUTPUT); VCC-Summer

Ultraschallsensor
pinMode(trigPin, OUTPUT);
pinMode(echoPin, INPUT);

pinMode (2, INPUT_PULLUP);

Serial.begin(9600);
}
void loop()
{
 ------------Fotowiderstand---------------
 int valuePhotor = analogRead(A1);
  Serial.println("Analoger WertPhotor : ");
  Serial.println(valuePhotor);
  Verzögerung(250);
```

```
------------------Knopf------------------
 if (digitalRead(2)== LOW or valuePhotor<110){
     buttonOn=true;
     digitalWrite(red_led, LOW);
     digitalWrite(green_led, HOCH);
     }

-------------Ultrasoc-Sensor--------------
lange Dauer, Entfernung;
digitalWrite(trigPin, LOW);
VerzögerungMikrosekunden(2);
digitalWrite(trigPin, HIGH);
VerzögerungMikrosekunden(10);
digitalWrite(trigPin, LOW);
Dauer = pulseIn(echoPin, HIGH);
Entfernung = (Dauer/2) / 29,1;
Serial.println("Distance");
Serial.println(distance);

digitalWrite(buzzVcc, LOW);

gas_value=analogRead(A0);

if (gas_value > sensorThres or distance < 20){
  while(buttonOn==true){
    if (digitalRead(2)== LOW){
      buttonOn=false;
      digitalWrite(red_led, LOW);
      digitalWrite(green_led, LOW);
      digitalWrite( buzz, LOW);
      Serial.println("NO LEAKAGE");
      Serial.println(gas_value);
   }else{
   digitalWrite(red_led, HOCH);
   digitalWrite(green_led, LOW);
   digitalWrite( buzz, HIGH);
   Serial.println("DANGER!!!!");
   Serial.print("Gaswert: ");
   Serial.println(gas_value);
   Serial.print("Entfernung: ");
   Serial.println(distance);

   Summer
   digitalWrite(buzzVcc, HIGH);
   digitalWrite(buzz, LOW); Ton EIN
   Verzögerung(dashL); Tonlänge
```

```
    digitalWrite(buzz, HIGH); Ton AUS
    delay(sPause); Symbolpause

    digitalWrite(buzz, LOW); Ton EIN
    Verzögerung (dotL); Tonlänge
    digitalWrite(buzz, HIGH); Ton AUS
    delay(sPause); Symbolpause

    digitalWrite(buzz, LOW); Ton EIN
    Verzögerung(dashL); Tonlänge
    digitalWrite(buzz, HIGH); Ton AUS
    delay(sPause); Symbolpause

    digitalWrite(buzz, LOW); Ton EIN
    Verzögerung (dotL); Tonlänge
    digitalWrite(buzz, HIGH); Ton AUS
    delay(sPause); Symbolpause

    delay(lPause-sPause); Subtrahiert bereits genommene Pause

    digitalWrite(buzz, LOW); Ton EIN
    Verzögerung(dashL); Tonlänge
    digitalWrite(buzz, HIGH); Ton AUS
    delay(sPause); Symbolpause
    digitalWrite(buzz, LOW); Ton EIN
    Verzögerung(dashL); Tonlänge
    digitalWrite(buzz, HIGH); Ton AUS
    delay(sPause); Symbolpause

    digitalWrite(buzz, LOW); Ton EIN
    Verzögerung (dotL); Tonlänge
    digitalWrite(buzz, HIGH); Ton AUS
    delay(sPause); Symbolpause

    digitalWrite(buzz, LOW); Ton EIN
    Verzögerung(dashL); Tonlänge
    digitalWrite(buzz, HIGH); Ton AUS
    delay(sPause); Symbolpause
    delay(wPause-sPause); Subtrahiert bereits genommene Pause
    } //end else
  } //end while
 } //end wenn
 Verzögerung(100);
}
```

Schritt 5 (Temperatursensor und Lasersendermodul im Alarmsystem hinzugefügt)

Schließen Sie den Temperatursensor an



In diesem Schritt benötigen wir einen Lasersender, einen Fotowiderstand für den Empfänger und einen 1K-Ohm-Widerstand, um ihn zu schützen.

So funktioniert's:

Der Lasersender weist zusammen mit dem Fotowiderstandssensor, der als Empfänger fungiert, den Arduino an, das System einzuschalten, wenn der Empfang aus irgendeinem Grund unterbrochen wird. Außerdem wird das Alarmsystem aktiviert, wenn der Wert des Temperatursensors einen bestimmten Wert überschreitet.

Arduino-Code von Schritt 5:

```
ARBEIT : ALARMANLAGE
TEMPERATURSENSOR IM SYSTEM HINZUGEFÜGT
int red_led=8;
int green_led=9;
int gas_value;
int sensorThres=400;

----------Summer----------
int buzz= 13; I/O-Pin vom Summer wird hier angeschlossen
int buzzVcc= 7; Vcc-Pin vom Summer verbindet hier
const int wpm = 20; Morsegeschwindigkeit in WPM
const int dotL = 1200/wpm; Berechnete Punktlänge
const int dashL = 3*dotL; Bindestrich = 3 x Punkt
const int sPause = dotL; Symbolpause = 1 Punkt
const int lPause = dashL; Buchstabenpause = 3 Punkte
const int wPause = 7*dotL; Wortpause = 7 Punkte

----------Ultraschallsensor----------
#define trigPin 3
#define echoPin 4

----------Knopf----------
boolesche SchaltflächeOn=false;

----------Temperatursensor----------
float tempf;
int tempPin = A2;

----------Laser----------
```

```
int Laser = 5; Erstellen einer Variablen namens Laser, die dem digitalen Pin 5 zugewiesen ist
int valueLaserPh=0;// Erstellen einer Variablen mit dem Namen valueLaserPh und Setzen des Werts auf Null
float valueLaserPhF=0;// Erstellen einer Variablen mit dem Namen valueLaserPhF und Setzen des Werts auf Null

Raumtemperatur in Fa
const float baselineTemp = 100.0;

void setup()
{
  pinMode(red_led,OUTPUT);
  pinMode(buzz,OUTPUT);
  pinMode(green_led,OUTPUT);
  pinMode(A0;INPUT); A0 Gas

  pinMode(buzz,OUTPUT); Summer-Pin als Ausgang festlegen
  pinMode(buzzVcc,OUTPUT); VCC-Summer

  Ultraschallsensor
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);

----------Laser-------------------
  pinMode (Laser, OUTPUT); Festlegen des digitalen Pins 5 für den Ausgang
  digitalWrite(Laser,LOW); Stellen Sie nur sicher, dass der Laser beim Starten oder Zurücksetzen ausgeschaltet ist


  pinMode (2, INPUT_PULLUP);

  Temperatursensor
  pinMode(tempPin,INPUT);

  Serial.begin(9600);
}


void loop()
{
  digitalWrite(red_led, LOW);
```

```
 digitalWrite(Laser,HIGH);
 ----------Laser-------------------------------
 valueLaserPh=analogRead(A4); Lesen der Spannung auf A4 und Speichern des empfangenen
Wertes in "Spannung"
 valueLaserPhF = valueLaserPh * (5,0 / 1023,0); Umwandlung des in "Spannung"
gespeicherten Wertes in lesbare Informationen

 ----------Temperatursensor--------------------------
 tempf = analogRead(tempPin);
 tempf=(tempf*5)/10;
 Wandeln Sie das analoge Volt in sein Temperaturäquivalent um
 Serial.print("TEMPERATURE = ");
 Serial.print(tempf); Temperaturwert anzeigen
 Serial.print("*F");
 Serial.println();
 Verzögerung(1000); Sensorablesung jede Sekunde aktualisieren

 ----------Fotowiderstand-----------------
 int valuePhotor = analogRead(A1);
 Serial.print("Photoresistor value : ");
 Serial.println(valuePhotor);
 Verzögerung(250);

----------Knopf----------------------
 if (digitalRead(2)== LOW oder valuePhotor<20){
    buttonOn=true;
    digitalWrite(red_led, LOW);
    digitalWrite(green_led, HOCH);
    digitalWrite(Laser,HIGH); Einschalten des Lasers
    Serial.println("Das Alarmsystem ist aktiviert!");
    }

----------Ultrasoc-Sensor ----------
lange Dauer, Entfernung;
digitalWrite(trigPin, LOW);
VerzögerungMikrosekunden(2);
digitalWrite(trigPin, HIGH);
VerzögerungMikrosekunden(10);
digitalWrite(trigPin, LOW);
Dauer = pulseIn(echoPin, HIGH);
```

```
Entfernung = (Dauer/2) / 29,1;
Serial.print("Abstand : ");
Serial.println(distance);
digitalWrite(buzzVcc, LOW);
gas_value=analogRead(A0);
Serial.print("Gaswert : ");
Serial.println(gas_value);
Serial.print("Laserwert : ");
Serial.println(valueLaserPhF);

if (gas_value > sensorThres oder Abstand < 10 oder tempf>baselineTemp oder
valueLaserPhF<1)
 {
  while(buttonOn==true){
   if (digitalRead(2)== LOW){
     buttonOn=false;
     digitalWrite(red_led, LOW);
     digitalWrite(green_led, LOW);
     digitalWrite( buzz, LOW);
     Serial.println("Das Alarmsystem ist deaktiviert!");
     Entfernung=100;
     tempf=0,0;
  }else{
      digitalWrite(red_led, HOCH);
      digitalWrite(green_led, LOW);
      digitalWrite( buzz, HIGH);
      Serial.println("DANGER!!!!");
      Serial.print("Gaswert: ");
      Serial.println(gas_value);
      Serial.print("Entfernung: ");
      Serial.println(distance);
      Serial.print("Temperatur: ");
      Serial.println(tempf);
      Serial.print("Photoresistor value : ");
      Serial.println(valuePhotor);
      Serial.print("Laserwert : ");
      Serial.println(valueLaserPhF);

      ----------Summer----------
      digitalWrite(buzzVcc, HIGH);
```

digitalWrite(buzz, LOW); Ton EIN
Verzögerung(dashL); Tonlänge
digitalWrite(buzz, HIGH); Ton AUS
delay(sPause); Symbolpause

digitalWrite(buzz, LOW); Ton EIN
Verzögerung (dotL); Tonlänge
digitalWrite(buzz, HIGH); Ton AUS
delay(sPause); Symbolpause

digitalWrite(buzz, LOW); Ton EIN
Verzögerung(dashL); Tonlänge
digitalWrite(buzz, HIGH); Ton AUS
delay(sPause); Symbolpause

digitalWrite(buzz, LOW); Ton EIN
Verzögerung (dotL); Tonlänge
digitalWrite(buzz, HIGH); Ton AUS
delay(sPause); Symbolpause

delay(lPause-sPause); Subtrahiert bereits genommene Pause

digitalWrite(buzz, LOW); Ton EIN
Verzögerung(dashL); Tonlänge
digitalWrite(buzz, HIGH); Ton AUS
delay(sPause); Symbolpause
digitalWrite(buzz, LOW); Ton EIN
Verzögerung(dashL); Tonlänge
digitalWrite(buzz, HIGH); Ton AUS
delay(sPause); Symbolpause

digitalWrite(buzz, LOW); Ton EIN
Verzögerung (dotL); Tonlänge
digitalWrite(buzz, HIGH); Ton AUS
delay(sPause); Symbolpause

digitalWrite(buzz, LOW); Ton EIN
Verzögerung(dashL); Tonlänge
digitalWrite(buzz, HIGH); Ton AUS
delay(sPause); Symbolpause

```
        delay(wPause-sPause); Subtrahiert bereits genommene Pause
      }//end else
    }//end while
  } //end wenn
Verzögerung(100);
}
```

PROJEKT 3- NAME: WebServer

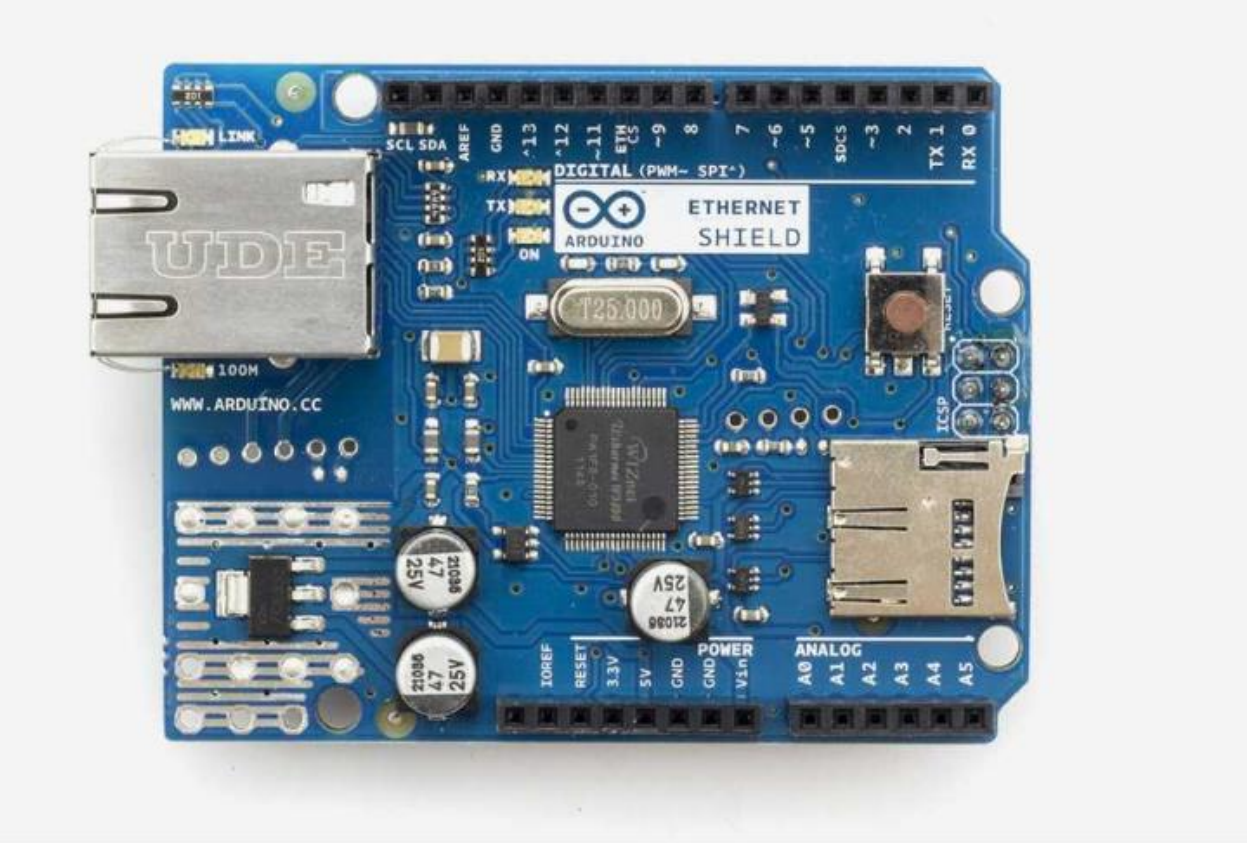
Das Ziel des Projekts: In diesem Projekt werden die Schüler sehen, wie wir einen Arduino über ein Ethernet-Shield mit einem lokalen Netzwerk verbinden. Sie experimentieren mit Sensoren und können ihre empfangenen Werte über das lokale Netzwerk auf ihrem Computerbildschirm sehen.

Durch dieses Projekt konnten die Schüler experimentieren mit:

- ✓ einen Gassensor und
- ✓ ein Fotowiderstandssensor
- ✓ einen Temperatursensor

Diese Ethernet-Abschirmung ermöglicht es einem Arduino-Board, sich mit dem Internet zu verbinden.

Komponentenliste:



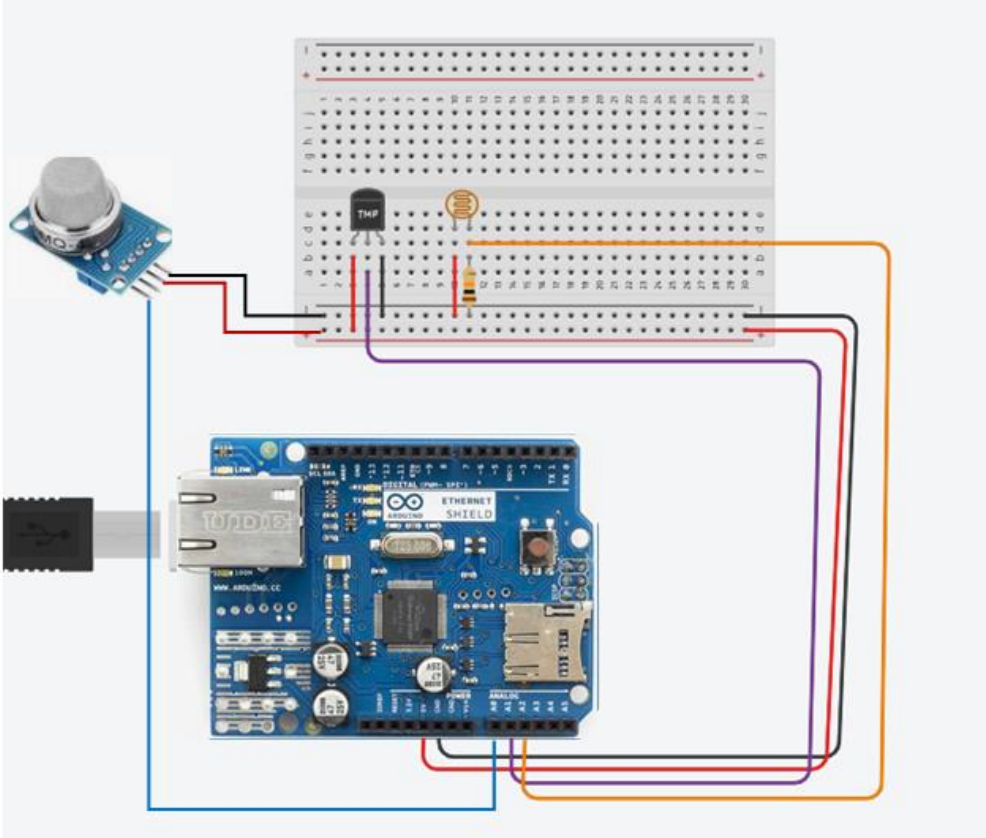
- Ethernet-Abschirmung
- Ethernet-Kabel
- Steckbrett und Sprungdrähte
- Kabel-USB
- Arduino UNO R3
- ein Fotowiderstandssensor
- ein Gassensor
- einen Temperatursensor
- ein Widerstand 10KOhm

So funktioniert's:

Das Arduino Ethernet Shield 2 ermöglicht es einem Arduino Board, sich mit dem Internet zu verbinden. Es basiert auf dem (Wiznet W5500 Ethernet-Chip). Das Wiznet W5500 bietet einen Netzwerk-Stack (IP), der sowohl TCP als auch UDP unterstützt. Es unterstützt bis zu acht gleichzeitige Socket-Verbindungen.

In diesem Beispiel suchen die Schüler die IP des Ethernet-Shields in einem lokalen Netzwerkbrowser und können die Werte der Sensoren, die sie mit dem Ethernet-Shield verbunden haben, auf einer benutzerfreundlichen Seite sehen. Die Seite wird automatisch alle 5 Sekunden aktualisiert.

Projektschaltung.



Methodik

Zunächst beschreiben wir den gewünschten Workflow des zu entwickelnden Projekts in Schritten. Die Schüler werden dann gebeten, die notwendigen Komponenten auszuwählen, um die Schaltung zu entwerfen und zu zeichnen. Sie verwenden die Anwendung "Arduino", um den Code zu schreiben.

Dann erstellen sie die Schaltung, schreiben den Code auf das Arduino-Softwaretool und laden ihn auf das Arduino-Board hoch.

Die Schüler werden überprüfen, ob die Schaltung korrekt konstruiert wurde.

Dieses Projekt richtet sich an Grundschüler, die beginnen, von den Ergebnissen unseres Erasmus+ KA-202 Strategic Partnerships in VET Projekts namens "ArduInVet" zu profitieren.

Arduino Code des Projekts:

```
/*
 Webserver

 Ein einfacher Webserver, der den Wert der analogen Eingangspins anzeigt.
 mit einem Arduino Wiznet Ethernet-Shield.

 Stromkreis:
 * Ethernet-Abschirmung an den Pins 10, 11, 12, 13
 * Analoge Eingänge an den Pins A0 bis A5 (optional)

 */

#include <SPI.h>
#include <Ethernet.h>
unsigned long refreshCounter = 0;
Geben Sie unten eine MAC-Adresse und IP-Adresse für Ihren Controller ein.
Die IP-Adresse hängt von Ihrem lokalen Netzwerk ab:
byte mac[] = {
  0xA8, 0x61, 0x0A, 0xAE, 0x63 0xA8
};
IPAddress ip(169, 254, 122, 108);

Initialisieren der Ethernet-Serverbibliothek
mit der IP-Adresse und dem Port, den Sie verwenden möchten
(Port 80 ist Standard für HTTP):
EthernetServer(80);

EthernetClient-Client;
void setup() {
  Sie können Ethernet.init(pin) verwenden, um den CS-Pin zu konfigurieren
  Ethernet.init(10);  Die meisten Arduino-Shields
  Ethernet.init(5);   MKR ETH Schild
  Ethernet.init(0);   Teensy 2.0
```

```
 Ethernet.init(20);  Teensy++ 2,0
 Ethernet.init(15);  ESP8266 mit Adafruit Featherwing Ethernet
 Ethernet.init(33);  ESP32 mit Adafruit Featherwing Ethernet

 Öffnen Sie die serielle Kommunikation und warten Sie, bis der Port geöffnet ist:
 Serial.begin(9600);
 während (! Seriell) {
              ; Warten Sie, bis der serielle Port eine Verbindung hergestellt hat. Nur für den
nativen USB-Anschluss erforderlich
 }
 Serial.println("ARDUinVET in Ethernet WebServer ");

 Starten Sie die Ethernet-Verbindung und den Server:
 Ethernet.begin(mac, ip);

 Überprüfen Sie, ob Ethernet-Hardware vorhanden ist
 if (Ethernet.hardwareStatus() == EthernetNoHardware) {
   Serial.println("Ethernet-Shield wurde nicht gefunden.  Leider kann nicht ohne Hardware
ausgeführt werden. :(");
              while (true) {
                        Verzögerung(1); nichts tun, ohne Ethernet-Hardware keinen Sinn
machen
              }
 }
 if (Ethernet.linkStatus() == LinkOFF) {
              Serial.println("Ethernet-Kabel ist nicht angeschlossen.");
 }
 Starten Sie den Server
 server.begin();
 Serial.print("server is at ");
 Serial.println(Ethernet.localIP());
}

void loop() {
    Lauschen auf eingehende Clients
   client = server.available();
   if (client) {
       Serial.println("ARDUinVET in client");
      Eine HTTP-Anforderung endet mit einer Leerzeile
      boolean currentLineIsBlank = true;
      while (client.connected()) {
          if (client.available()) {
             char c = client.read();
            Serial.write(c);
            Wenn Sie am Ende der Zeile angekommen sind (einen Zeilenumbruch erhalten
haben
```

```
            Zeichen) und die Zeile leer ist, die HTTP-Anforderung beendet wurde,
            damit Sie eine Antwort senden können
              if (c == '\n' && currentLineIsBlank) {
                  Senden eines standardmäßigen HTTP-Antwortheaders
                  client.println("HTTP/1.1 200 OK");
                  client.println("ARDinVET - HTTP");
                  client.println("Content-Type: text/html");
client.println("Verbindung: schließen");  Die Verbindung wird nach //Abschluss der Antwort
geschlossen
                  client.println("Aktualisieren: 5");  Seite automatisch alle 5 Sekunden
aktualisieren
                 client.println();
                 Webseite (Client);
                 brechen;
                 }
                 if (c == '\n') {
                       Sie beginnen eine neue Zeile
                       currentLineIsBlank = true;
                       } else if (c != '\r') {
                             Sie haben einen Charakter in der aktuellen Zeile
                             currentLineIsBlank = false;
                          }
                 }
            }
            Geben Sie dem Webbrowser Zeit, um die Daten zu empfangen
            Verzögerung(1);
            Schließen Sie die Verbindung:
            client.stop();
            Serial.println("Client getrennt");
       }
}

void webpage(EthernetClient client) { /* function webpage */
wbepage an Client senden
client.println("ARDUinVET - HTTP");
client.println("Content-Type: text/html");
client.println(""); Vergessen Sie diesen nicht
client.println("<! DOCTYPE HTML>");
client.println("<html>");
client.println("<head>");
client.println("<title>ARDUinVET</title>");
client.println("<script src='https://smtpjs.com/v3/smtp.js'></script>");
client.println("<script>");
client.println("function sendEmail() {");
/*client.println("Email.send({");
client.println(" Host: 'smtp.gmail.com',");
```

```
client.println(" Benutzername : '**********@gmail.com',");
client.println(" Passwort : '********************',");
client.println(" To : '**********@gmail.com',");
client.println(" Von : '**********@gmail.com',");
client.println(" Betreff : 'Daten von Arduino',");
client.println(" Body : 'Sensors value....',");
client.println(" }).then(");
client.println(" message => alert('mail sent successful')");
client.println(" );"); */
client.println("}");
client.println("</script>");
client.println("</head>");
client.println("<body bgcolor = '#cccc00'>");
client.println("<hr/><hr>");
client.println("<h1 style='color : #0000cc;' ><Mitte> Alarmanlage </Mitte></h1>");
client.println("<hr/><hr>");
client.println("<br><br>");
client.println("<br><br>");
client.println("<center>");
client.println("<br>Sensors Output<br>");
client.println("Gas.....:");
client.println(" <input value=" + String(analogRead(A0)) + " readonly></input>");
client.println("<br>");
client.println(" Photoresistor..:");
client.println(" <input value=" + String(analogRead(A1)) + " readonly></input>");
client.println("<br>");
client.println(" Temperatur....:");
client.println(" <input value=" + String(analogRead(A2)) + " readonly></input>");
client.println("<br>");
client.println(" </center>");
client.println("<br><br>");
client.println("<center>");
client.println("<a style='color : #0000cc;'
href='https://www.arduinvet.com/'>www.arduinvet.com</a>");
client.println("</center>");
client.println("<br><br>");
client.println("</body></html>");
client.println();
Verzögerung(1);
}
```

Ein Bildschirm der Projektseite

Co-funded by the Erasmus+ Programme of the European Union

Erasmus+ KA-202

Strategic Partnerships Project for Vocational Education and Training

Project Title: "Teaching and Learning Arduinos in Vocational Training"

Project Acronym: " ARDUinVET "

Project No: "2020-1-TR01-KA202-093762"

Free Arduino Projects

(These projects were made by students of partner schools in the ARDUinVET partnership.)

NO:	Project NAME	Country
1	SOCIAL DISTANCING HAT	TURKIYE
2	TEMPERATURE METER (GREECE)	GREECE
3	EMERGENCY BRAKE BLINKER - ADAPTIVE BRAKE LIGHT	ROMANIA
4	LINEFOLLOWER	AUSTRIA
5	AUTOMATED GREENHHOUSE (ITALY)	ITALY

1_PROJECT NAME: SOCIAL DISTANCING HAT (TURKIYE)

The Aim of the Project: In these days, when the Covid-19 pandemic is effective, following the "SOCIAL DISTANCE" rule is one of the basic conditions of not getting the disease.

In order to draw attention to the "social distaning" issue, we decide to design an Arduino project on the subject.

The rule is reminded with the “Social Distancing Hat”, which we will prepare. It will give an audible buzzer sound and at the same time a light warning.

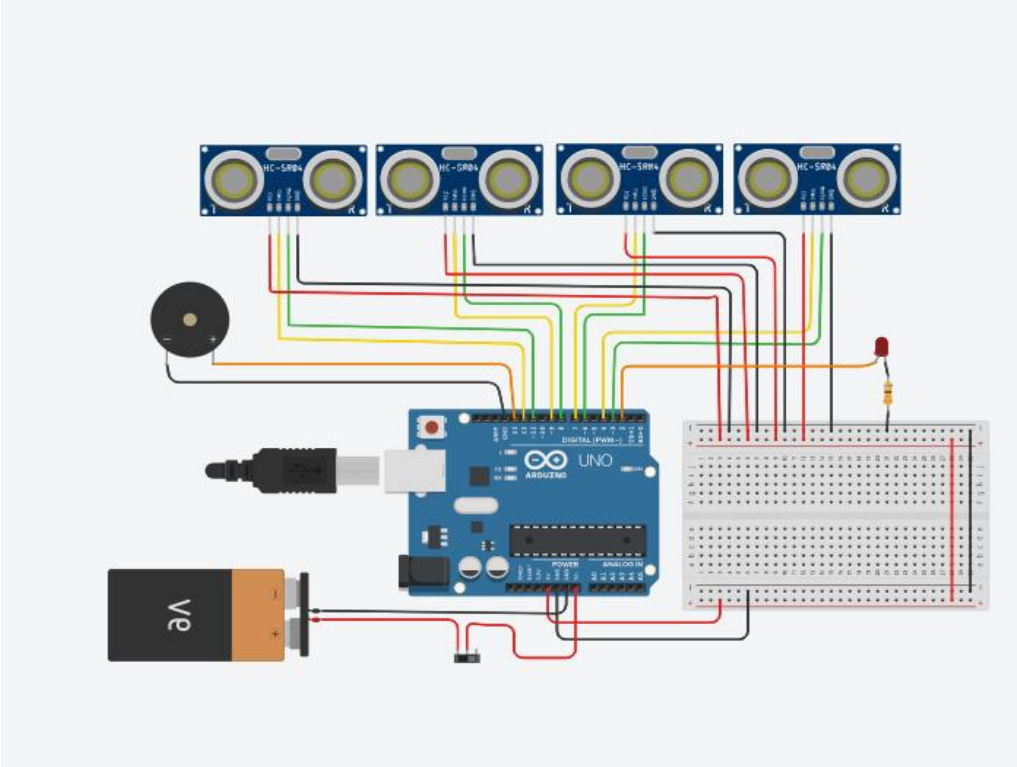
Our Project Method:

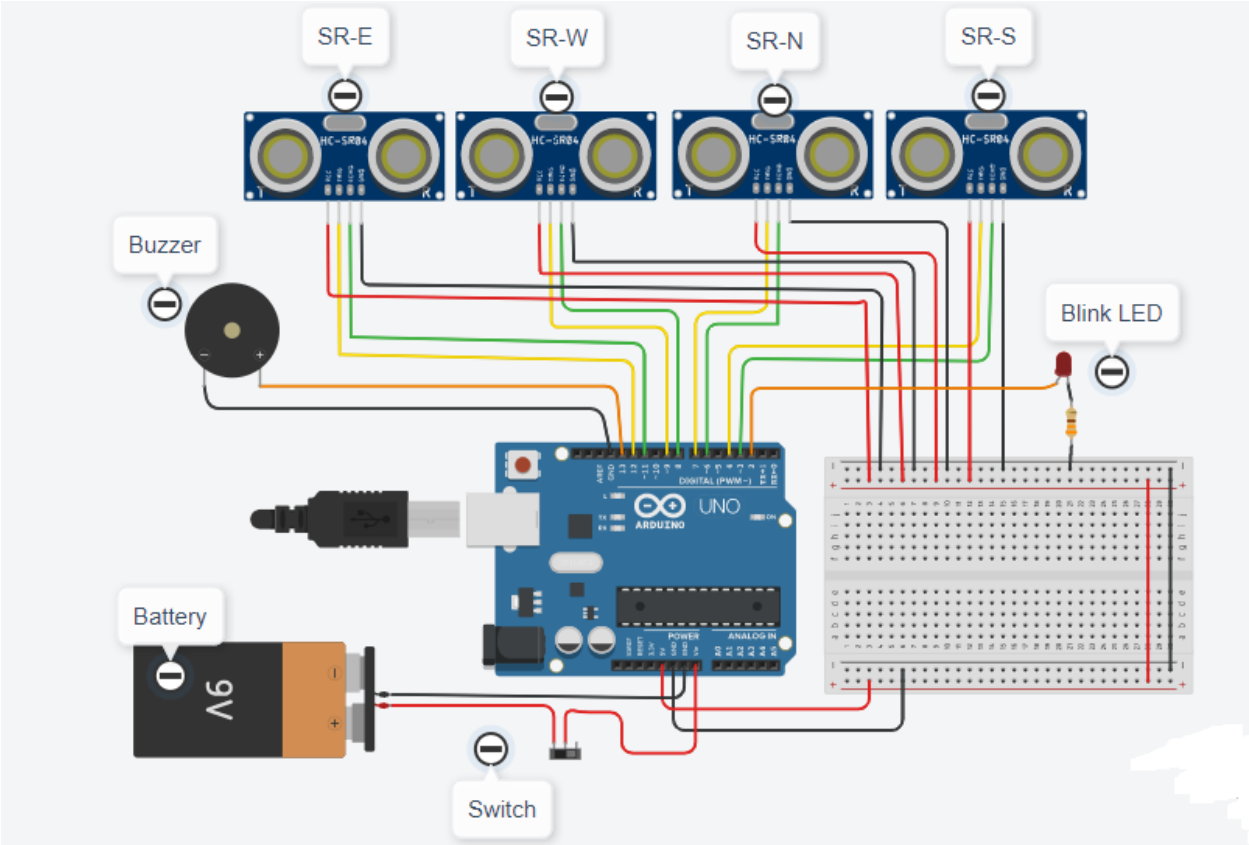
First, the algorithm of the project is prepared. According to this prepared algorithm, the logical design is made and the Arduino program is written. Arduino is used in the design of the circuit. While making these designs, our students benefit from all the results and products of the Erasmus+ KA-202 VET project, named as “ArduinVET”.

Our finalized design circuit is set up after providing the necessary equipment and programming the Arduino. Our final design circuit is visually placed on the cap.

This project is exhibited at fairs and reminds the visitors of the "Social Distancing" rule and its importance.

Project Circuit.





How It Works:

4 ultrasonic distance sensors are placed in 4 directions of the hat. When the sensors detect someone, approaching or close to 100 cm or below, the buzzer will sound and the led will flash. The circuit will be powered by a 9V battery.

List of Materials:

A Hat, an Arduino Uno R3, 4 Ultrasonic Distance Sensors, a Buzzer, a 330 Ω Resistor, a Blink-Red LED, a 9V Battery, a Slide Switch.

Arduino Program of Project:

//Project Name: SOCIAL DISTANCING HAT:

```
int trigPin=12;
int echoPin=11;
int trigPin2=9;
int echoPin2=8;
int trigPin3=7;
int echoPin3=6;
int trigPin4=4;
int echoPin4=3;
void setup()
{
pinMode(trigPin, OUTPUT);
pinMode(echoPin, INPUT);
 pinMode(trigPin2, OUTPUT);
pinMode(echoPin2, INPUT);
pinMode(trigPin3, OUTPUT);
pinMode(echoPin3, INPUT);
 pinMode(trigPin4, OUTPUT);
pinMode(echoPin4, INPUT);
pinMode(13,OUTPUT);
pinMode(2,OUTPUT);
 Serial.begin(9600);
}
void loop() {
 digitalWrite(trigPin, LOW);
delay(3);
```

```
digitalWrite(trigPin, HIGH);
delay(3);
digitalWrite(trigPin, LOW);
int time1 = pulseIn(echoPin, HIGH);
int distance1 = (time1/2) / 28.97;         //Social distance is equal to and less than 100cm.
//-----------------------------------------------------
 digitalWrite(trigPin2, LOW);
delay(3);
digitalWrite(trigPin2, HIGH);
delay(3);
digitalWrite(trigPin2, LOW);

int time2 = pulseIn(echoPin2, HIGH);
int distance2 = (time2/2)  / 28.97;    //Social distance is equal to and less than 100cm.
//-----------------------------------------------------
 digitalWrite(trigPin3, LOW);
delay(3);
digitalWrite(trigPin3, HIGH);
delay(3);
digitalWrite(trigPin3, LOW);
int time3 = pulseIn(echoPin3, HIGH);
int distance3 = (time3/2)  / 28.97;       //Social distance is equal to and less than 100cm.
//-----------------------------------------------------
digitalWrite(trigPin4, LOW);
delay(3);
digitalWrite(trigPin4, HIGH);
```

```
delay(3);
digitalWrite(trigPin4, LOW);
int time4 = pulseIn(echoPin4, HIGH);
int distance4 = (time4/2)  / 28.97;         //Social distance is equal to and less than 100cm.
//---------------------------------------------------------
if(distance1<75)
 {
  digitalWrite(13,HIGH);
  digitalWrite(2,HIGH);
  }
 else if(distance2<75)
 {
  digitalWrite(13,HIGH);
   digitalWrite(2,HIGH);
  }
 else if(distance3<75)
 {
  digitalWrite(13,HIGH);
   digitalWrite(2,HIGH);
  }
 else if(distance4<75)
 {
  digitalWrite(13,HIGH);
   digitalWrite(2,HIGH);
  }
  else
```

```
 {
 digitalWrite(13,LOW);
 digitalWrite(2,LOW);
 }
  Serial.println(distance1);      // To see which ultrosonc sensor is activated on the Serial monitor
  Serial.println(distance2);
  Serial.println(distance3);
  Serial.println(distance4);
  delay(500);
}
```

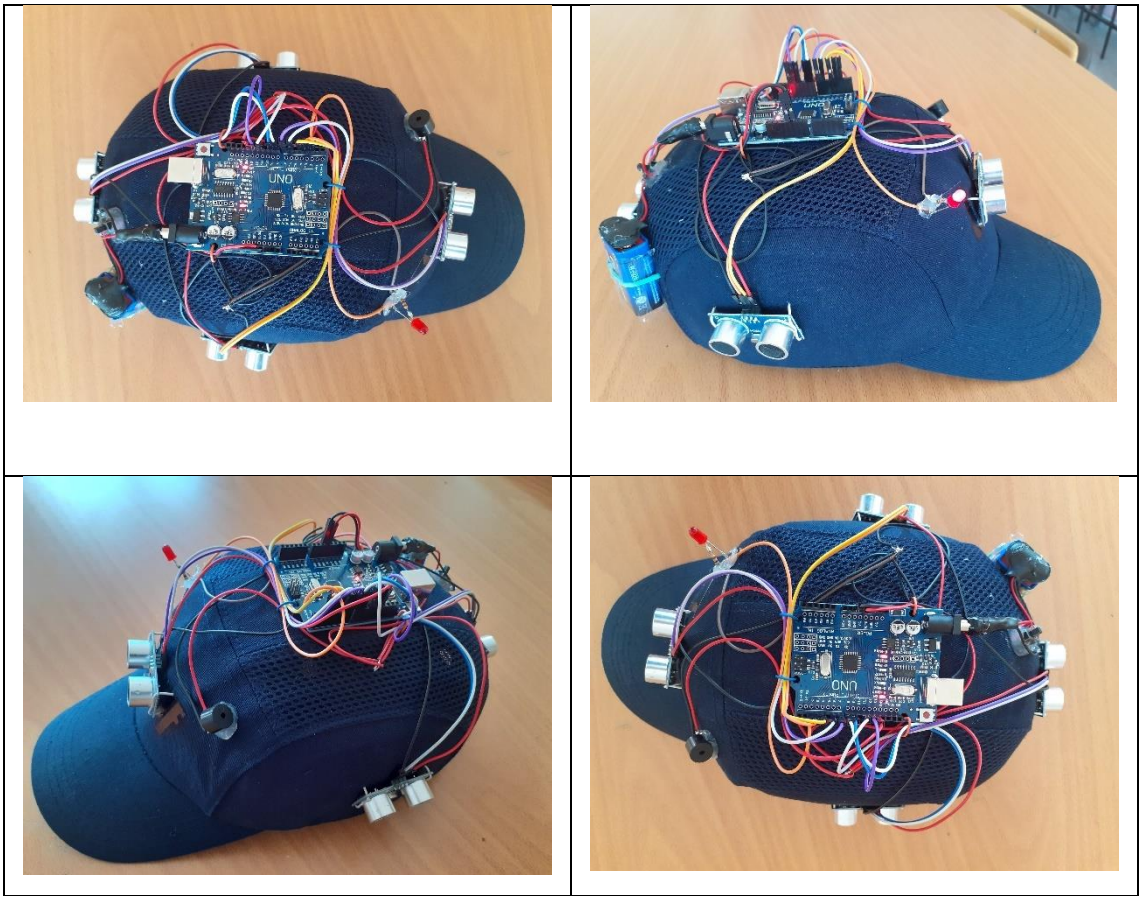


Photo: Final design circuit on the cap

2_PROJECT NAME: TEMPERATURE METER (GREECE)

The Aim of the Project: This project is an extended version of the simple basic Love Meter project someone can find in the original Arduino handbook. In this extended version, students will be able to deal with a sensor (temperature sensor), experiment with the potentiometer, light an RGB LED and get to know the LCD display (if not available, otherwise the display will be just the Serial Monitor of the Arduino software tool).

It's a project which allows the students to experiment with many components of different types and get familiar with the Input values and Display methods.

Methodology

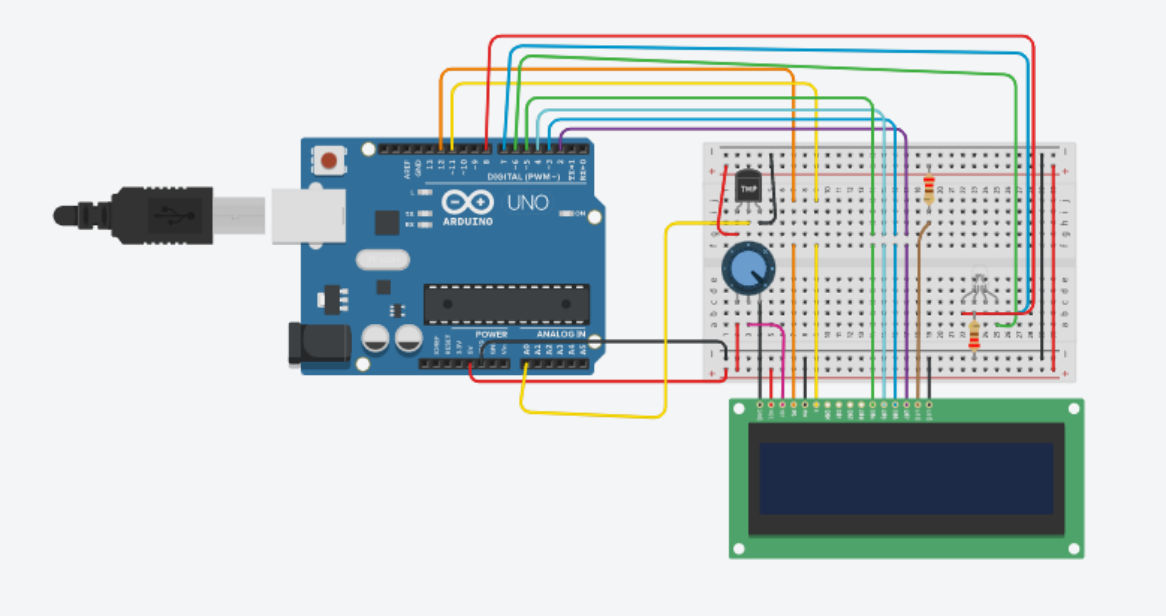
Initially, we describe the desired workflow of the project. Then we ask the students to select the needed components and use Tinkercad to design and draw the circuit. They will use the Tinkercad Code Tool to write the code.

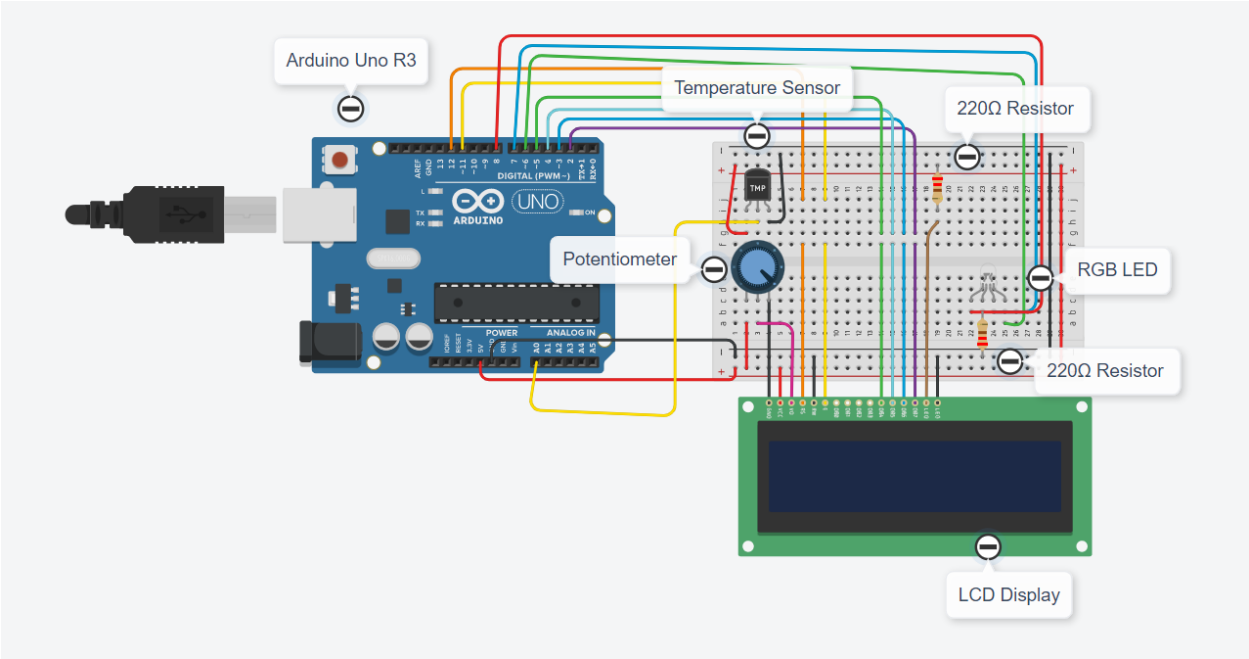
Then they will create the circuit, write the code on the Arduino software tool and uploaded it on the Arduino board.

Simulation on Tinkercad will verify that the circuit was built correctly.

This project is for basic level students, who are starting to benefit from the results and outcomes from our Erasmus+ KA-202 Strategic Partnerships in VET project, called "ArduinVET".

Project Circuit.





How It Works:

The temperature sensor inputs the values of the room temperature. Off course, students can change the temperature using their fingers. The temperature will be displayed in the LCD Display (if available) and the Serial Monitor. The RGB LED will have different colors (off, green, blue, red) according to the temperature value. The potentiometer will be used as a switch for the LCD Display. The circuit will be powered from the USB Cable (if needed, you can use a battery box instead).

Components List:

Our components are: an Arduino Uno R3, a Temperature Sensor, two (2) 220Ω Resistors, an RGB LED and (optional) a Potentiometer and a LCD Display (16*2).

Arduino Code of Project:

//Project Name: TEMPERATURE METER

// include the library code:

```
#include <LiquidCrystal.h>
int t=0;
int sensor = A0;
float temp;
float tempc;
float tempf;
// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
//************RGB**********
const int sensorPin = A0;
// room temperature in Celsius
const float baselineTemp = 20.0;
//**************************
void setup() {
  // set up the LCD's number of columns and rows:
pinMode(sensor,INPUT);
lcd.setCursor (0,0);
lcd.begin(16, 2);
  // Print a message to the LCD.
lcd.print ("    ARDUINO     ");
lcd.setCursor (0,1);
lcd.print ("TEMPERATURE METER");
```

```
delay (3000);
Serial.begin(9600);
for(int pinNumber = 6; pinNumber<8; pinNumber++){
    pinMode(pinNumber,OUTPUT); // pin6:Blue pin7:Green  pin8:Red
    digitalWrite(pinNumber, LOW);
   }
}
void loop() {
delay(2000);
t=t+2;
temp=analogRead(sensor);
//tempc=(temp*5)/10;
tempc=map(((temp-20)*3.04), 0, 1023, -40, 125);
tempf=(tempc*1.8)+32;
Serial.println("_______");
Serial.println("Temperature Logger");
Serial.print("Time in Seconds= ");
Serial.println(t);
Serial.print("Temp in deg Celcius = ");
Serial.println(tempc);
Serial.print("Temp in deg Fahrenheit = ");
Serial.println(tempf);
lcd.setCursor(0,0);
lcd.print("Temp in C = ");
lcd.println(tempc);
lcd.setCursor(0,1);
```

```
lcd.print("Temp in F = ");
lcd.println(tempf);

//*************code RGB***************
  // read the value on AnalogIn pin 0
  // and store it in a variable
  int sensorVal = analogRead(sensorPin);
  // send the 10-bit sensor value out the serial port
  Serial.println("sensor Value: ");
  Serial.println(sensorVal);

  // convert the ADC reading to voltage
  float voltage = (sensorVal/1024.0) * 5.0;
  // Send the voltage level out the Serial port
  Serial.println(", Volts: ");
  Serial.println(voltage);
   if(tempc < 20){
   digitalWrite(6, LOW);
   digitalWrite(7, LOW);
   digitalWrite(8, LOW);
   }
   else if(tempc >= 20 && tempc < 80){
         digitalWrite(6, HIGH);
   digitalWrite(7, LOW);
   digitalWrite(8, LOW);
  }
```

```
 else if(tempc >= 80 && tempc < 140){
        digitalWrite(6, LOW);
  digitalWrite(7, HIGH);
  digitalWrite(8, LOW);
}
 else if(tempc >= 140){
        digitalWrite(6, LOW);
  digitalWrite(7, LOW);
  digitalWrite(8, HIGH);
}
delay(100);
}
```

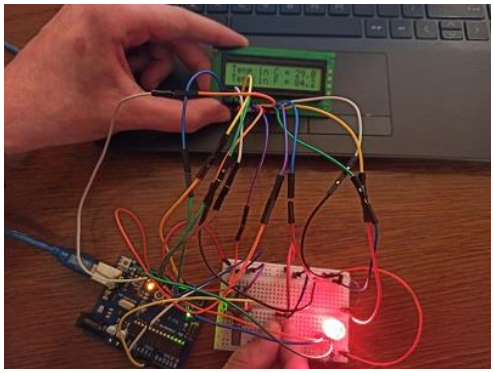
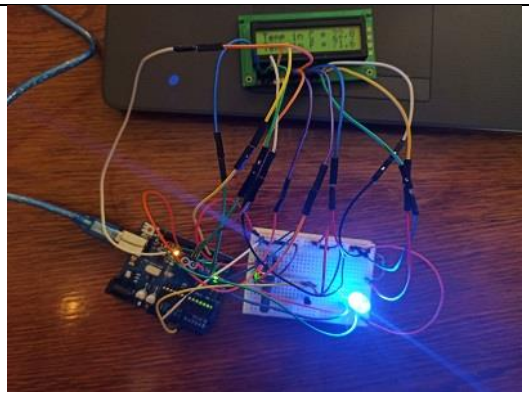
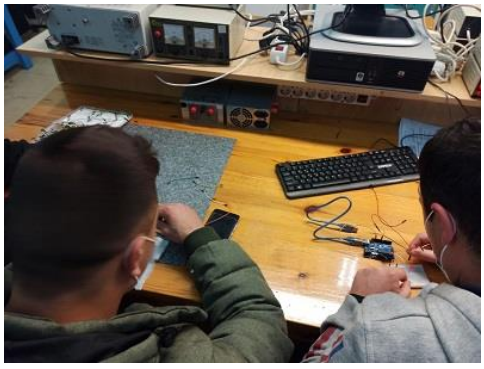





Photo: Project training kit and Students working on the project.

3_PROJECT NAME: EMERGENCY BRAKE BLINKER - ADAPTIVE BRAKE LIGHT

(ROMANIA)

The Aim of the Project: In the event of heavy braking, the adaptive brake light can warn oncoming traffic and thus help prevent rear-end collisions. They are activated if, during braking, and depending on the braking pressure and speed, if a critical braking situation is identified.

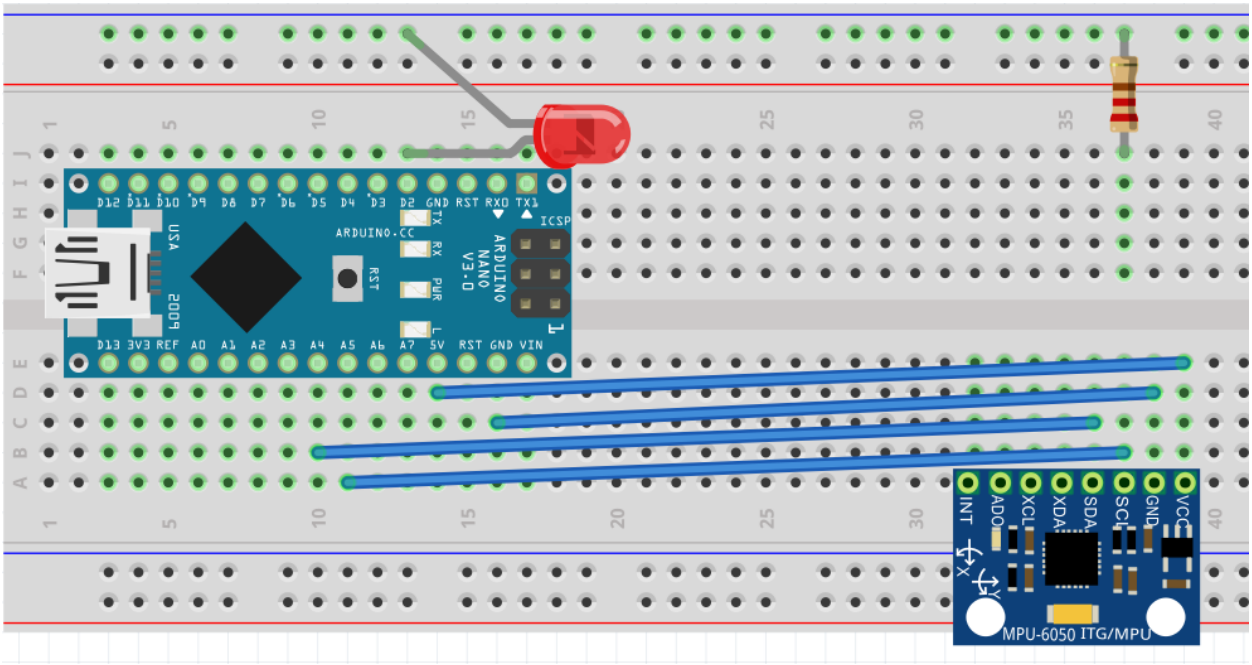
Our Project Method:

First, the algorithm of the project is prepared. According to this prepared algorithm, the logical design is made and the Arduino program is written. Arduino is used in the design of the circuit. While making these designs, our students benefit from all the results and products of the Erasmus+ KA-202 VET project, named as "ArduinVET".

Our finalized design circuit is set up after providing the necessary equipment and programming the Arduino. Our final design circuit is placed in a car.

This project is exhibited at fairs and reminds the visitors how important is traffic safety.

Project Circuit.



How It Works:

Emergency brake lights are **activated to alert vehicles behind about heavy braking**. The function means that the brake light flashes instead of - as in normal braking - shining with a constant glow. Emergency brake lights are activated at speeds above 50 km/h in the event of heavy braking.

It is based on an electronic accelerometer (MPU6050) assisted by a microcontroller and detects sudden braking based on the force of inertia and quickly flashes the 3rd brake stop, making the car much easier to notice than the rear.

MPU6050 has a 3-axis gyroscope, 3- axis Accelerometer and a Digital motion processor integrated on a single chip. It works on the power supply of 3V-5V. MPU6050 **uses the I2C protocol for communication and transfer of data**. This module has a built-in 16-bit ADC which provides great accuracy.

The Kalman filter is an efficient recursive filter that evaluates the state of a dynamic system based on a series of noise-sensitive measurements. Due to its intrinsic characteristics, it is an excellent filter for noise and disturbances that act on zero-average Gaussian systems.

List of Materials:

An Arduino Nano, an MPU6050, an Resistor 180 Ohm, an red LED.

Arduino Program of Project:

//Project Name: EMERGENCY BRAKE BLINKER - ADAPTIVE BRAKE LIGHT

```
#include<Wire.h>;
//https://www.codetd.com/en/article/11749279 this is an articol who explain very well how to
comunicate more fast with MPU6050
//in automotive conditions is many noise from the road conditions, vibrations etc. and I'll use a
very good software filter (Kalman)
const int MPU_addr = 0x68;
float kalman_old = 0;
float cov_old = 1;
float val1, val2, val3, val4, val5;
int med1_sort, med2_sort, med3_sort, med4_sort;
float avg;
int med;
```

```
int m;
int med_sort[5];
int c_avg = 1;
int c_med = 1;
int d = 0;
float old_x = 0;
float real_angle = 0;
float prev_angle = 0;
unsigned long previousMillis = 0;
const long interval = 50;
void setup() {
  Serial.begin(9600); // for USB monitor
  Serial.println ("Hit a key to start");     // signal initalization done
  while (Serial.available() == 1);
  //Connect to G-Sensor
  Wire.begin();
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x6B); // PWR_MGMT_1 register
  Wire.write(0x00); // set to zero (wakes up the MPU-6050)
  Wire.endTransmission(true);
  delay(50);
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x1C);
  Wire.write(0x00);
  Wire.endTransmission(true);
  pinMode(2, OUTPUT); //output signal for FIRST and SECOND stage.
digitalWrite (2, LOW);
//pinMode(7, OUTPUT);//output signal for SECOND stage. Enable this and next line if you want more expressive functionality.
//digitalWrite (7, LOW); If enable this option, you need enable digital output pin 7 in all sketch!!
//this signal might be use with one PNP transistor or P-Gate MOSFET for complete the circuit
//in real circuit I use a 5v blue led. In fritzing circuit use a regular 1.8 red led with aditional resistor - 180 ohm}
```

```
void loop() {
  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis >= interval) {
    previousMillis = currentMillis;
    Wire.beginTransmission(MPU_addr);
    Wire.write(0x3D);
// 0x3B (ACCEL_XOUT_H) & 0x3C (ACCEL_XOUT_L)
// 0x3D (ACCEL_YOUT_H) & 0x3E (ACCEL_YOUT_L)
// 0x3F (ACCEL_ZOUT_H) & 0x40 (ACCEL_ZOUT_L)
    Wire.endTransmission(false);
    Wire.requestFrom(MPU_addr, 2, true);
    int16_t YAxisFull =  (Wire.read() << 8 | Wire.read());
    float YAxisFinal = (float) YAxisFull / 16384.0;
    int YAxis_Filtered = general_filter(c_avg, c_med, YAxisFull);
    int YAxis_kalman = kalman_filter(YAxisFull);
    int YAxis_movavg = mov_avg(c_avg, YAxisFull);
    int YAxis_median = median(c_med, YAxisFull);
    Serial.print("YAxis_Filtered");Serial.print(YAxis_Filtered/16384.0);
    Serial.print("\t");
   // Serial.print("YAxis_kalman");Serial.print(YAxis_kalman);
  //  Serial.print("\t");
  //  Serial.print("YAxis_movavg");Serial.print(YAxis_movavg);
   // Serial.print("\t");
  //  Serial.print("YAxis_median");Serial.print(YAxis_median);
  //  Serial.print("\t");
    Serial.print("YAxisFull");Serial.println(YAxisFull/16384.0);
   // Serial.print("\t");
   // Serial.print("YAxis_Final");Serial.println(YAxis_Final);
//FIRST stage
if (YAxis_Filtered/16384.0 > 0.4 and YAxis_Filtered/16384.0 <= 0.7){
  for (int i = 1; i <= 4; i++) {
      digitalWrite (2, HIGH);
     // digitalWrite (7, HIGH);
```

```
      delay (75);
      digitalWrite (2, LOW);
     // digitalWrite (7, LOW);
      delay (50);}}
else {
  //SECOND stage - very hard brake
if (YAxis_Filtered/16384.0 > 0.7){
   for (int i = 1; i <= 7; i++) {
      digitalWrite (2, HIGH);
     // digitalWrite (7, HIGH);
      delay (75);
      digitalWrite (2, LOW);
      //digitalWrite (7, LOW);
      delay (50);}}}
    c_avg = c_avg + 1;
    c_med = c_med + 1;
    if (c_avg > 5 && c_med > 5)
    {      c_avg = 6;
      c_med = 6;    }  }}
float kalman_filter (float input)
{  float kalman_new = kalman_old;
  float cov_new = cov_old + 0.50;
  float kalman_gain = cov_new / (cov_new + 0.9);
  float kalman_calculated = kalman_new + (kalman_gain * (input - kalman_new));
  cov_new = (1 - kalman_gain) * cov_old;
  cov_old = cov_new;
  kalman_old = kalman_calculated;
  return kalman_calculated;}
float mov_avg (int counter, float input)
{  avg = 0;
  m = 0;
  if (counter == 1) {
    val1 = input;
```

```
  avg = val1;  }
else if (counter == 2) {
  val2 = input;
  avg = val2;  }
else if (counter == 3) {
  val3 = input;
  avg = val3;  }
else if (counter == 4) {
  val4 = input;
  avg = val4;  }
else if (counter == 5) {
  val5 = input;
  avg = val5;  }
else if (counter > 5 ) {
  counter = 6;
  if (val1 == 0) {
    m = m + 1;    }
  if (val2 == 0) {
    m = m + 1;    }
  if (val3 == 0) {
    m = m + 1;    }
  if (val4 == 0) {
    m = m + 1;    }
  if (val5 == 0) {
    m = m + 1;    }
  if (input == 0) {
    m = m + 1;    }
  d = 6 - m;
  if (d == 0)
  {      avg = input;
    counter = 1;    }
  else
  {      avg = (val1 + val2 + val3 + val4 + val5 + input) / d;    }
```

```
    val1 = val2;
    val2 = val3;
    val3 = val4;
    val4 = val5;
    val5 = input;  }
  return avg;}
float median (int counter, int input)
{  if (counter == 1) {
    med1_sort = input;
    med_sort[0] = med1_sort;  }
  else if (counter == 2) {
    med2_sort = input;
    med_sort[1] = med2_sort;  }
  else if (counter == 3) {
    med3_sort = input;
    med_sort[2] = med3_sort;  }
  else if (counter == 4) {
    med4_sort = input;
    med_sort[3] = med4_sort;  }
  else if (counter >= 5) {
    counter = 6;
    med_sort[4] = input;
    sort(med_sort, 5);
    med = med_sort[2];
    med1_sort = med2_sort;
    med2_sort = med3_sort;
    med3_sort = med4_sort;
    med4_sort = input;
    med_sort[0] = med1_sort;
    med_sort[1] = med2_sort;
    med_sort[2] = med3_sort;
    med_sort[3] = med4_sort;  }
  return med;}
```

```
void sort(int a[], int size) {
  for (int i = 0; i < (size - 1); i++) {
    for (int o = 0; o < (size - (i + 1)); o++) {
      if (a[o] > a[o + 1]) {
        int t = a[o];
        a[o] = a[o + 1];
        a[o + 1] = t;      }    }  }  }
//function with all filters applied
float general_filter (int counter_avg, int counter_med, float input) {
float input_movavg = mov_avg(counter_avg, input);    //Moving Avg application
float input_med = median(counter_med, input_movavg); //Median application
float input_filtered = kalman_filter(input_med);     //Kalman application
  return input_filtered;}
```

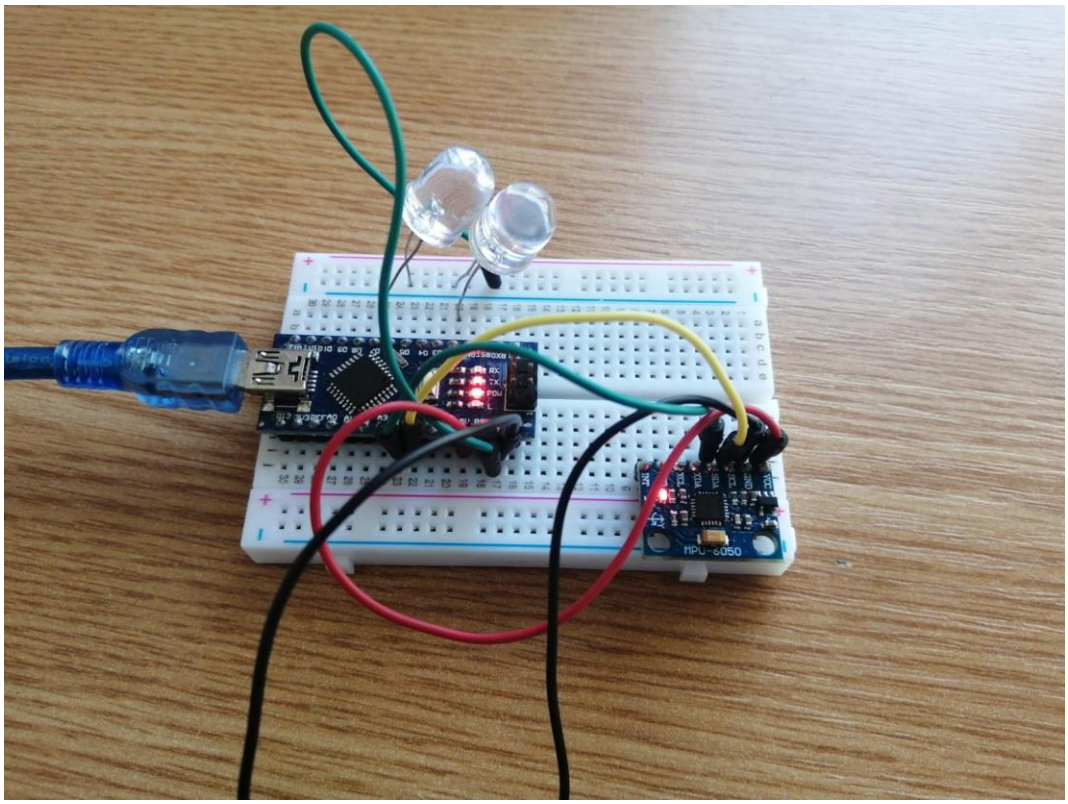


Photo: Final design circuit

4_PROJECT NAME: LINEFOLLOWER (AUSTRIA)

What a Linefollower is:

The line follower robot is a mobile machine that can detect and follow the line drawn on the floor. Generally, the path is predefined and can be either visible like a black line on a white surface with a high contrasted color. Definitely, this kind of robot should sense the line with its infrared ray (IR) sensors that installed under the robot. After that, the data is transmitted to the processor. Hence, the processor is going to decide the proper commends and then it sends them to the driver and thus the path will be followed by the line follower robot.

The important point of building a line follower robot is a good control that is sufficient to follow the path as fast as possible.

Working of Line Follower Robot

The concept of the line follower robot is related to light. Here, we use the behaviour of light on the black and white surface. The white colour reflects all the light that falls on it, whereas the black colour absorbs the light.

In this line follower robot, we use IR transmitters and receivers (photodiodes). They are used to send and receive the lights. When IR rays fall on a white surface, it is reflected towards IR receiver, generating some voltage changes.

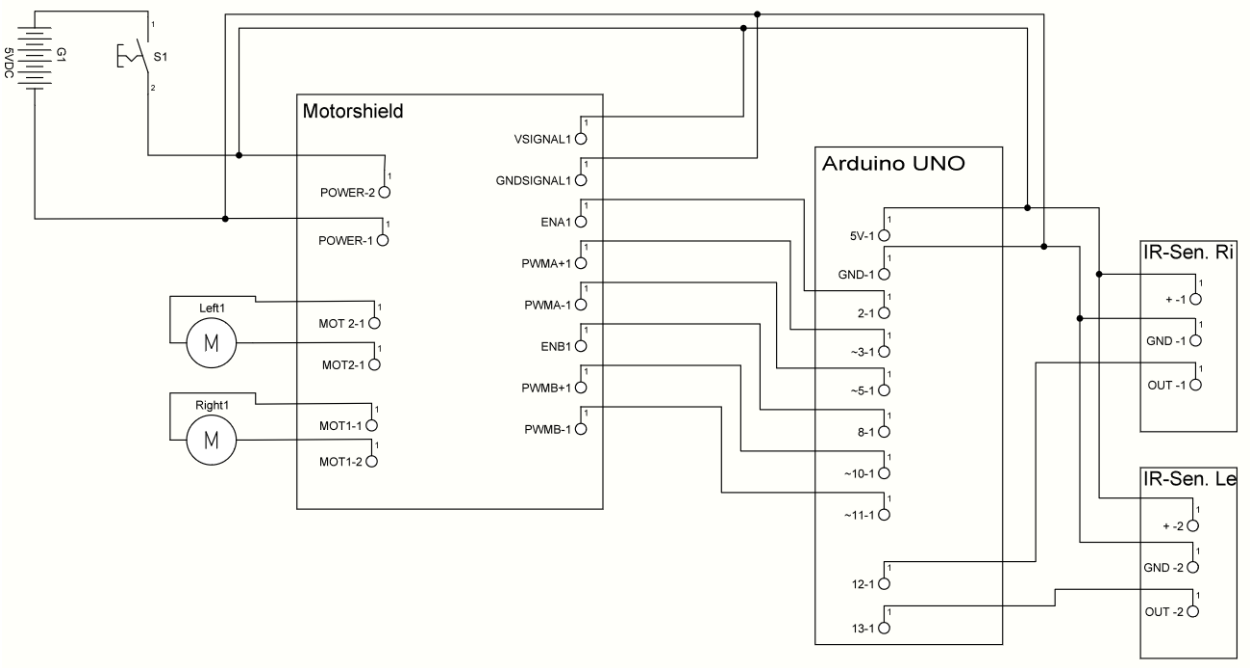
When IR rays fall on a black surface, it is absorbed by the black surface, and no rays are reflected; thus, the IR receiver doesn't receive any rays.

In this project, when the IR sensor senses a white surface, an Arduino gets 1 (HIGH) as input, and when it senses a black line, an Arduino gets 0 (LOW) as input. Based on these inputs, an Arduino Uno provides the proper output to control the bot.

Partlist:

parts	type	pieces
Arduino	Uno	1
Motorshield	by HTL Wolfsberg	1
DC-Motor	COM-Motor01, 3-9Volt DC, joy-it	2
PowerBank	Flip12 PhonePowerBank, 3350mAh	1
IR-Sensors	SEN-KY032IR, joy-it	2
Mainswitch		1
Jumperwire	RB-CB3-025, joy-it	1
USB Cable	Digitus,AK-300102-010-S, Typ A-B	s

Schematic circuit diagram:



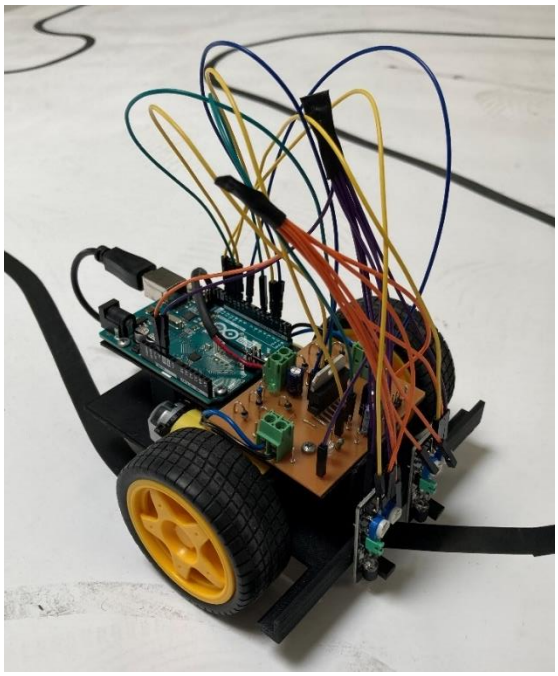


Photo: Final design circuit

Program:

// Linefollower

```
const byte sr = 12;

const byte sl = 13;

const byte enr = 2;

const byte enl = 8;

const byte mrf = 3;

const byte mrb = 5;

const byte mlf = 10;

const byte mlb = 11;


void setup()

{

  Serial.begin(9600);


  pinMode(sr,INPUT);

  pinMode(sl,INPUT);


  pinMode(enr,OUTPUT);

  pinMode(enl,OUTPUT);

  digitalWrite(enr,HIGH);

  digitalWrite(enl,HIGH);


  pinMode(mrf,OUTPUT);
```

```
  pinMode(mrb,OUTPUT);
  pinMode(mlf,OUTPUT);
  pinMode(mlb,OUTPUT);

  analogWrite(mrf,0);
  analogWrite(mrb,0);
  analogWrite(mlf,0);
  analogWrite(mlb,0);
}
  bool senR;
  bool senL;

void loop()
{
  senR = digitalRead(sr);
  senL = digitalRead(sl);

  if(!senR && !senL) //staight
  {
    analogWrite(mrf,220);
    analogWrite(mrb,0);
    analogWrite(mlf,200);
    analogWrite(mlb,0);
  }
```

```
//right sensor on line
//drive to right
else if(senR && !senL)
{
  analogWrite(mrf,0);
  analogWrite(mrb,100);
  analogWrite(mlf,255);//255
  analogWrite(mlb,0);
}

//left sensor on line
//drive to left
else if(!senR && senL)
{
  analogWrite(mrf,255);//255
  analogWrite(mrb,0);
  analogWrite(mlf,0);
  analogWrite(mlb,100);
}

// both sensors on black
// stop
else if(senR && senL)
```

```
 {
   analogWrite(mrf,0);
   analogWrite(mrb,0);
   analogWrite(mlf,0);
   analogWrite(mlb,0);
   delay(3000);
 }
}
```

5_PROJECT NAME: AUTOMATED GREEN HOUSE (ITALY)

The Aim of the Project: Why an automated greenhouse project at school?

Students at school, followed various paths on active citizenship and on the UN Agenda 2030 for Sustainable Development, about the energy sustainability, the importance of avoiding water wastage, and the need to develop innovative and environmentally friendly cultivation methods, considering the profound climate changes taking place. The young people's ecological awareness is expressed in the search for technical solutions and innovative ideas, starting from the theoretical and laboratory knowledge acquired at school inherent in the application opportunities of Arduino.

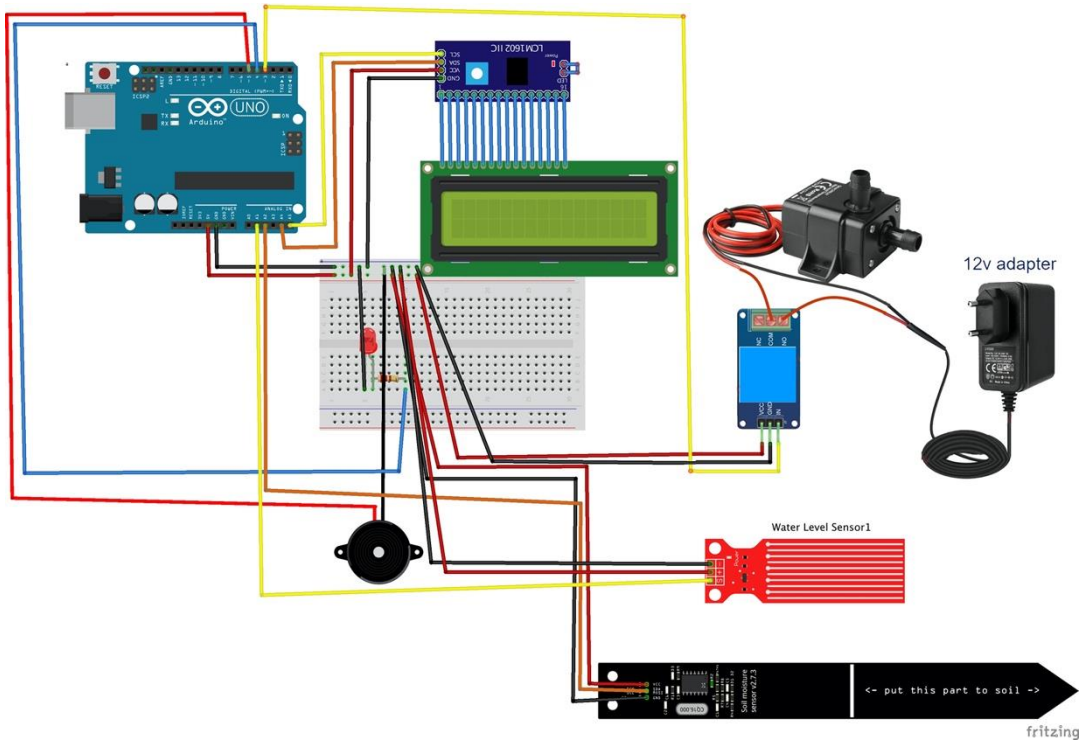
Some students supported by the teachers, designed a system for sensing soil and air temperature, soil moisture, to manage the irrigation and ventilation of a prototype greenhouse suitable for the cultivation of local fruits and vegetables.

The experimentation yielded the hoped-for results and will contribute to the actual implementation in the area in the vicinity of the school intended to host the future school vegetable garden.

Our Project Method

First, a list of all the parameters that we want to use to operate the greenhouse has been prepared. Starting from this a first version of the project algorithm was prepared. According to this algorithm, the necessary sensors were purchased and the logic project was realized. Arduino is used in circuit design. Finally, the final script was implemented. After checking the correct setting of the threshold values of the various sensors, they were positioned inside the greenhouse.

Project Circuit



How It Works:

An LCD display provides the values of temperature and humidity of the air, soil humidity and quantity of water present in the irrigation container. when the values are below a certain threshold, a motorized pump controlled by a relay starts irrigation of the soil. when the water level is below a certain threshold, the display shows a refill warning accompanied by an audible alarm.

List of Materials:

A greenhouse, an Arduino Uno R3, a temperature/humidity DHT11 sensor, a buzzer, a soil humidity sensor, a water level sensor, a red led, a 12V power adapter , a 12V pump, a 220 ohm resistor, a relay, an LCD display.

Arduino Program of Project:

//Project Name: AUTOMATED GREENHOUSE:

```
#include <LiquidCrystal_I2C.h>

#include <Wire.h>

#include <DHT.h>

#define DHTPIN 11

#define DHTTYPE DHT11 // DHT 11

DHT dht(DHTPIN, DHTTYPE);

LiquidCrystal_I2C lcd(0x27,20,4);  // set the LCD address to 0x27 for a 16 chars and 2 line display

int t, h, Lumen, lumin, water, igro, umdtr, wlevel;


const int pin_water = A1;

const int pin_igro = A2;

const int pin_pump = 3;

const int pin_led = 4;

const int pin_buzzer = 5;


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


   dht.begin();

   lcd.init();

   lcd.backlight();

   lcd.setCursor(5,0);

   lcd.print("School");

   lcd.setCursor(3,1);

   lcd.print("Greenhouse");

   delay (5000);

   lcd.clear();

   lcd.setCursor(2,0);
```

```
    lcd.print("IIS Einstein");
    lcd.setCursor(3,1);
    lcd.print("De Lorenzo");
    delay (5000);
    lcd.clear();

    pinMode(pin_pump,OUTPUT);
    pinMode(pin_led, OUTPUT);
    pinMode(pin_buzzer, OUTPUT);
    digitalWrite(pin_pump, HIGH);
}

void loop() {
    // water level
    water = analogRead (pin_water);
    wlevel = map (water,0, 1023, 0, 100);
    if(wlevel < 35){
        lcd.clear();
        tone(pin_buzzer, 800);
        digitalWrite(pin_led, HIGH);
        delay(300);
        noTone(pin_buzzer);
        digitalWrite(pin_led, LOW);
        delay(300);
        lcd.clear();
        //lcd.begin(16, 2);
        lcd.setCursor(2,0);
        lcd.print("Refill Water");
```

```
    delay (1000);
    lcd.clear();
} else {
    noTone(pin_buzzer);
    digitalWrite(pin_led, LOW);
}

// Igrometer
igro = analogRead(pin_igro);
umdtr = map (igro, 0, 1023, 0, 100);

if(umdtr < 45){
    digitalWrite(pin_pump, LOW);
    delay(10000);
    digitalWrite(pin_pump, HIGH);
}

// Lettura umidità e temperatura del sensore DHT11
h = dht.readHumidity();
t = dht.readTemperature();

// posiziona il cursore in colonna 0 e linea 1
// (nota: la linea 1 e la seconda linea, poichè si conta incominciando da 0):
lcd.setCursor(0, 0);
lcd.print("T:");
lcd.print(t);
lcd.print( (char) 223 );
lcd.print("C");
```

```
  lcd.setCursor(10, 0);
  lcd.print("H:");
  lcd.print(h);
  lcd.print("%");

  lcd.setCursor(0, 1);
  lcd.print("SH:");
  lcd.print(umdtr);
  lcd.print("%");

  lcd.setCursor(10, 1);
  lcd.print("WL:");
  lcd.print(wlevel);
  lcd.print("%");
}
```

Photo: Final design

Annexex

Arduino's mission is to enable anyone to enhance their lives through accessible electronics and digital technologies. There was once a barrier between the electronics, design, and programming world and the rest of the world. Arduino has broken down that barrier. For more information about the Arduinos, you can visit the website, below…

www.arduino.cc/

For more information about our ARDinVET project, you can visit the website, below…

www.arduinvet.com

“This project is Funded by the Erasmus+ Program of the European Union. However, European Commission and Turkish National Agency cannot be held responsible for any use which may be made of the information contained therein”