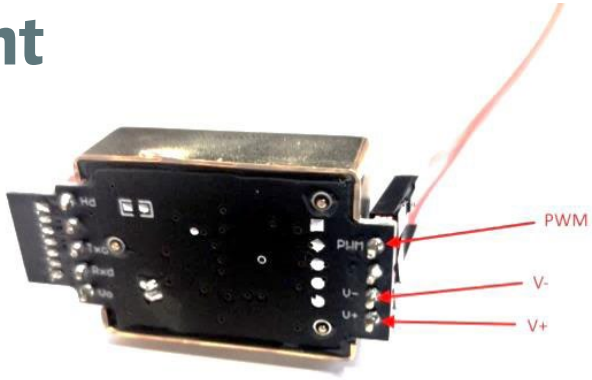
Downloadmaterial zum Beitrag „Mit einer CO_2-Ampel die Luftqualität in Klassenzimmern bestimmen“ – MINT *Zirkel* 3-2022

Fertigungsaufgabe: Bau einer einfachen CO_2-Ampel im Unterricht

1. Besorge dir folgende Bauteile:

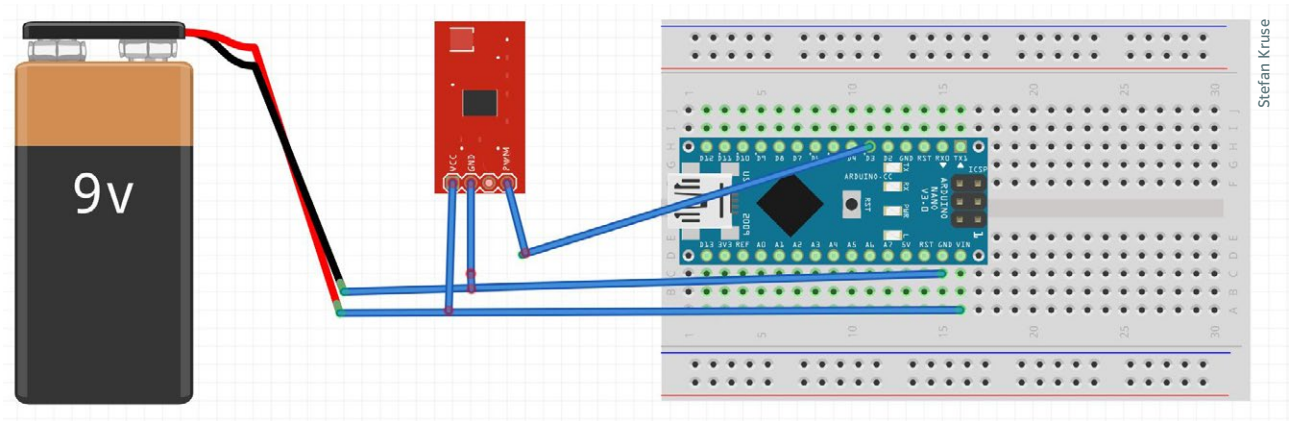
- 1 Arduino UNO-Mikrocontroller
- 1 MH-Z19B – CO_2-Sensor
- 5 Steckkabel für die Verbindungen
- je 1 LED in den Farben Grün, Gelb, Orange und Rot
- 1 Vorwiderstand 330 Ohm
- 1 Breadboard mit 400 Pins
- 1 9-V-Batterie plus Anschluss



Stefan Kruse

So sieht die Anschlussbelegung des CO_2-Sensors aus

2. Verdrahte die Messschaltung wie abgebildet. Die Batterie wird mit den Eingängen V_{in} (Pluspol) und GND (Minuspol) verbunden. Der Anschluss V– des CO_2-Sensors wird ebenfalls mit dem GND-Eingang verbunden. Da der Arduino eine konstante Versorgungsspannung für viele Sensoren von 5 V zur Verfügung stellt, wird der Anschluss V+ des Sensors direkt mit dem Ausgangspin +5 V verbunden. Nun fehlt noch das Datensignal des Sensors. Hierzu wird der OUT-Pin des Sensors mit dem Pin 3 des Arduino verbunden. Achte beim Anschließen darauf, dass keine Wackelkontakte entstehen und alle Verbindungen festgesteckt sind.



Stefan Kruse

So wird die Messschaltung verdrahtet

3. Installiere die Software zur Programmierung eines Arduino auf deinem Rechner. Das Programm findest du kostenfrei auf **www.arduino.cc** im Bereich „Software“. Installiere die für dein Betriebssystem passende Softwareversion und besorge dir außerdem ein USB-Verbindungskabel, welches zu deiner Arduino-Version passt.

4. Öffne das Programm mit der Software für den Arduino. Übertrage nun den folgenden Programmcode in ein neues Arduino-Script. Beachte: Alles, was hinter dem schrägen Doppelstrich (//) steht, wird vom Programm ignoriert und dient nur als Hinweis.

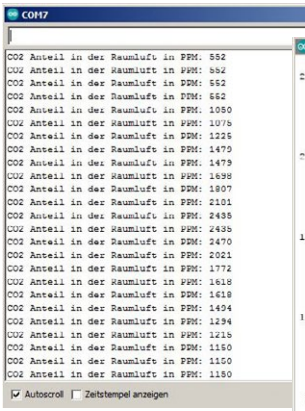
```
// CO2 Ampel mit dem Sensor MH-Z19B

int SensorPin = 3; // Der PWM-Ausgang des Sensors wird an Pin3 des Arduino angeschlossen.
int Messbereich = 5000; // Der Messbereich wird von 0-5000 ppm eingestellt.
unsigned long ZeitMikrosekunden; // Variable für die Dauer des PWM-Signalpegels in
Mikrosekunden.
unsigned long ZeitMillisekunden; // Variable für die Dauer des PWM-Signalpegels in
Millisekunden.
int PPM = 0; // Variable für den CO2-Messwert in ppm.
float Prozent=0; // Variable für die prozentuale Länge des PWM-Signals.
void setup()
{
```

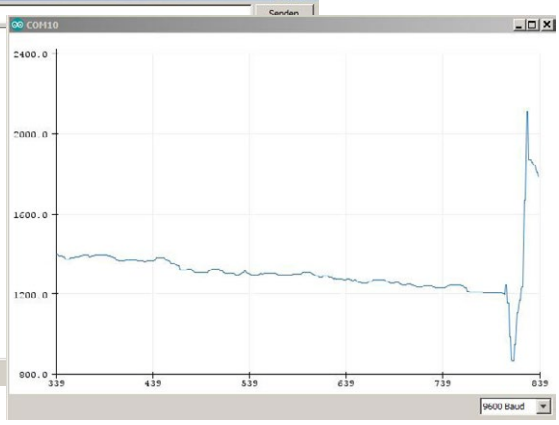
```
pinMode(SensorPin, INPUT); // Pin 3 für die Sensorwerte wird als Eingang definiert.
Serial.begin(9600); // Aufbau der seriellen Verbindung für die Anzeige der Werte am seriellen
Monitor.
}
void loop()
{
ZeitMikrosekunden = pulseIn(SensorPin, HIGH, 2000000); // Zeitmessung, ab der das Signal auf HIGH
wechselt.
ZeitMillisekunden = ZeitMikrosekunden/1000; // Umwandeln der Zeiteinheit von Mikrosekunden in
Millisekunden.
float Prozent = ZeitMillisekunden / 1004.0; // gemessene Signaldauer/die max. mögliche Signaldauer
= Prozentwert von 0 - 5000 PPM.
PPM = Messbereich * Prozent; // Berechnung des PPM-Werts aus der prozentualen Signaldauer und
dem maximalen Messbereich.
Serial.print(„CO2 Anteil in der Raumluft in PPM: „); // Ausgabe des CO2-Werts über den seriellen
Monitor des Programms.
Serial.println(PPM);
delay(2000); // Veränderbare Wartezeit von 2 Sekunden bis zur Erfassung des nächsten Messwertes.
}
```

Schließe den Arduino an deinen Computer an. Wähle das richtige Modell aus (z. B. Arduino/Genuino UNO) und stelle den richtigen Port (z. B. COM 10) ein. Übertrage nun das Programm in den Mikrocontroller. Bei einer Fehlermeldung wird dir im Programm angezeigt, wo dieser liegt und ob du dich eventuell vertippt hast oder der falsche Port gewählt wurde.

5. Wähle nun die Option „serieller Monitor“ in der Arduino-Software im Menü „Werkzeug“ aus. Nun werden dir die gemessenen Werte angezeigt. Alternativ kannst du über die Funktion „serieller Plotter“ auch ein Zeitdiagramm anzeigen lassen.



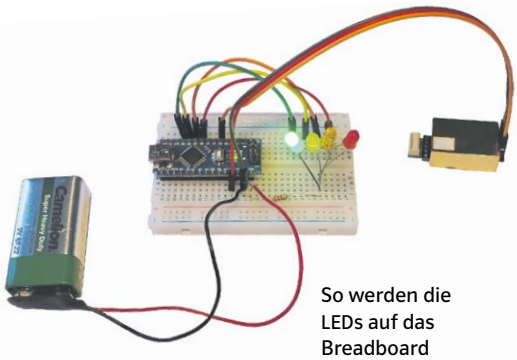
So werden dir die gemessenen Werte angezeigt



So wird dir ein Zeitdiagramm angezeigt

Stefan Kruse

6. Erweitere dein Projekt um eine optische Anzeige in Form von vier farbigen LEDs. Stecke dazu die LEDs so auf das Breadboard, dass die Kathoden (negativ, kürzerer Anschluss) in einer Anschlussreihe des Breadboards stecken. In diese Reihe wird auch der 330-Ohm-Widerstand gesteckt und auf der anderen Breadboard-Seite mit dem GND (Minuspol) des Arduino verbunden. Die Anoden (positiv, längerer Anschluss) werden so geschaltet, dass sie jeweils in versetzte Reihe gesteckt werden. Die Anschlussreihen, in denen die Anoden stecken, werden folgendermaßen mit den Anschlusspins am Arduino verbunden: rote LED mit Pin 6, orange LED mit Pin 7, gelbe LED mit Pin 8 und grüne LED mit Pin 9.



So werden die LEDs auf das Breadboard gesteckt

Stefan Kruse

7. Speichere das geöffnete Programm unter einem anderen Namen ab. Ergänze die zusätzlichen Zeilen in dem bestehenden Programmcode des Scripts.

```
// CO2-Ampel mit dem Sensor MH-Z19B und einer 4-farbigen LED-Anzeige

int ROT=6; // rote LED an Pin 6
int ORANGE=7; // orange LED an Pin 7
int GELB=8; // gelbe LED an Pin 8
int GRUN=9; // grüne LED an Pin 9
int SensorPin = 3; // Der PWM-Ausgang des Sensors wird an Pin 3 des Arduino angeschlossen.
```

```
int Messbereich = 5000; // Der Messbereich wird von 0 bis 5000 ppm eingestellt.
unsigned long ZeitMikrosekunden; // Variable für die Dauer des PWM-Signalpegels in
Mikrosekunden.
unsigned long ZeitMillisekunden; // Variable für die Dauer des PWM-Signalpegels in
Millisekunden.
int PPM = 0; // Variable für den CO2-Messwert in ppm.
float Prozent=0; // Variable für die prozentuale Länge des PWM-Signals.
void setup()
{
pinMode(SensorPin, INPUT); // Pin 3 für die Sensorwerte wird als Eingang definiert.
Serial.begin(9600); // Aufbau der seriellen Verbindung für die Anzeige der Werte am seriellen
Monitor.
pinMode(ROT, OUTPUT); // Pin 6 ist ein Ausgang.
pinMode(ORANGE, OUTPUT); // Pin 7 ist ein Ausgang.
pinMode(GELB,OUTPUT); // Pin 8 ist ein Ausgang.
pinMode(GRUN,OUTPUT); // Pin 9 ist ein Ausgang.
}
void loop()
{
ZeitMikrosekunden = pulseIn(SensorPin, HIGH, 2000000); // Zeitmessung, ab der das Signal auf HIGH
wechselt.
ZeitMillisekunden = ZeitMikrosekunden/1000; // Umwandeln der Zeiteinheit von Mikrosekunden in
Millisekunden.
float Prozent = ZeitMillisekunden / 1004.0; // gemessene Signaldauer/die max. mögliche Signaldauer
= Prozentwert von 0 bis 5000 PPM.
PPM = Messbereich * Prozent; // Berechnung des PPM-Werts aus der prozentualen Signaldauer und
dem maximalen Messbereich.
Serial.print("CO2 Anteil in der Raumluft in PPM: "); // Ausgabe des CO2-Werts über den seriellen
Monitor des Programms.
Serial.println(PPM);
if (PPM <700) // Wenn der Messwert kleiner als 700 ist, …
{
digitalWrite(GRUN, HIGH); // … ist die grüne LED an.
digitalWrite(GELB, LOW); // … ist die gelbe LED aus.
digitalWrite(ORANGE, LOW); // … ist die orange LED aus.
digitalWrite(ROT, LOW); // … ist die rote LED aus.
}
if (PPM >=7000 && PPM <=1000) // Wenn der Messwert größer-gleich 700 und kleiner-gleich 1000 ist, …
{
digitalWrite(GRUN, LOW); // … ist die grüne LED aus.
digitalWrite(GELB, HIGH); // … ist die gelbe LED an.
digitalWrite(ORANGE, LOW); // … ist die orange LED aus.
digitalWrite(ROT, LOW); // … ist die rote LED aus.
}
if (PPM >1000 && PPM <=1400) // Wenn der Messwert größer als 1000 und kleiner-gleich 1400 ist, …
{
digitalWrite(GRUN, LOW); // … ist die grüne LED aus.
digitalWrite(GELB, LOW); // … ist die gelbe LED aus.
digitalWrite(ORANGE, HIGH); // … ist die orange LED an.
digitalWrite(ROT, LOW); // … ist die rote LED aus.
}
if (PPM >1400) //Wenn der Messwert größer als 1400 ist, …
{
digitalWrite(GRUN, LOW); // … ist die grüne LED aus.
digitalWrite(GELB, LOW); // … ist die gelbe LED aus.
digitalWrite(ORANGE, LOW); // … ist die orange LED aus.
digitalWrite(ROT, HIGH); // … ist die rote LED an.
}
delay(3000); // Veränderbare Wartezeit von 3 Sekunden bis zur Erfassung des nächsten Messwertes.
}
```

Nun sollte deine Anlage zuverlässig anzeigen, ob die Luft im Klassenzimmer gut ist oder ob gelüftet werden muss. Vielleicht kannst du noch ein ansprechendes Gehäuse für die Anlage bauen?

Prof. Dr. Stefan Kruse hat die Professur für Technikdidaktik und Ingenieurpädagogik an der Pädagogischen Hochschule in Weingarten inne. Seine Forschungsschwerpunkte liegen in den Bereichen Neue Medien, Digitalisierung und Energie- und Fahrzeugtechnik. Neben seiner Tätigkeit an der Hochschule ist er Geschäftsführer der Deutschen Gesellschaft für Technische Bildung und Herausgeber der Genius- und DLR-Unterrichtsmaterialien.