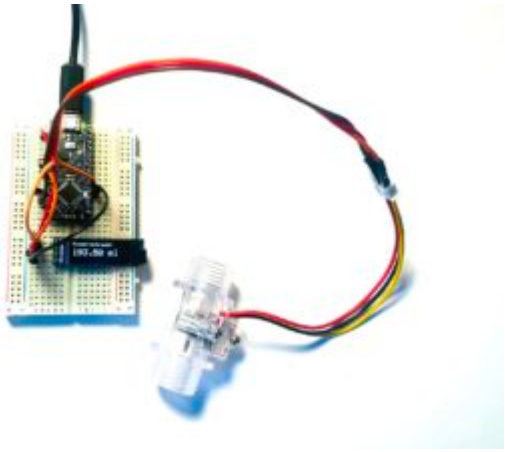
Mit einem Durchflussmesser den Wasserverbrauch messen



In diesem Projekt erfährst du, wie du mit einem Arduino, einem Durchflussmesser und einem OLED-Display einen einfachen, aber **effektiven Wasserzähler** baust. **Der Zähler misst die verbrauchte Wassermenge in Echtzeit und visualisiert sie direkt auf dem Display.**

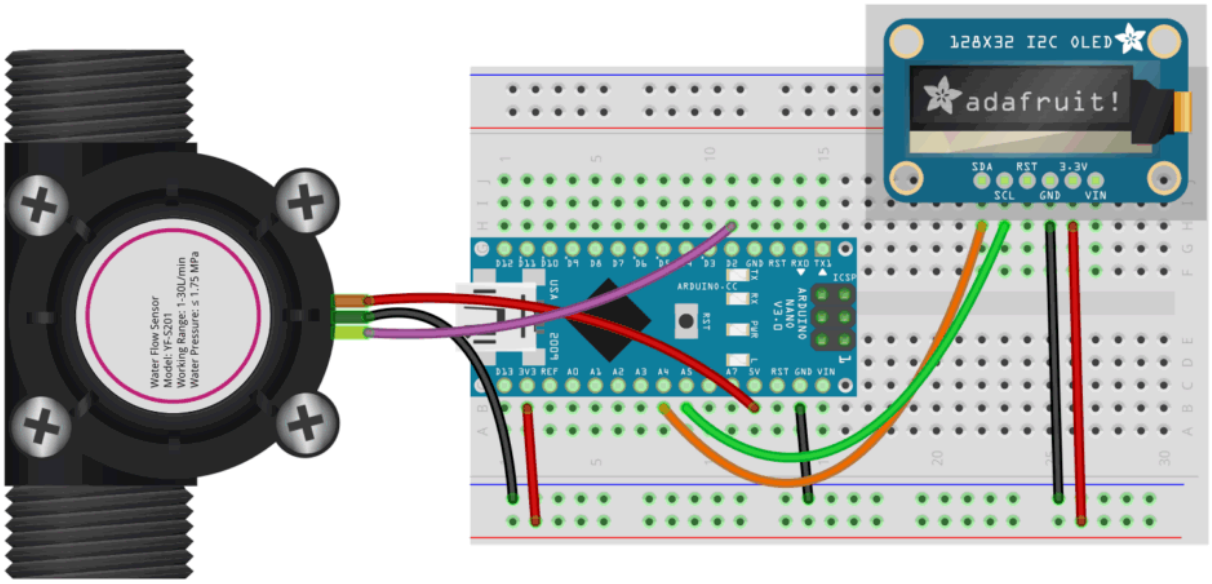
Benötigte Bauteile

Diese Bauteile brauchst du für dieses Projekt:

- **Mikrocontroller**: Zum Beispiel ein Arduino Nano oder Arduino UNO
- **Durchflussmesser**: Ein Turbinen-Durchflussmesser (z.B. YF-S201C*)
- **Display**: Ein 0,91“ OLED-Display (SSD1306-Chip)
- **Breadboard** und Jumper-Kabel
- **Messbecher**: Für die Kalibrierung, idealerweise 500 ml oder 1 Liter

Aufbau des Wasserzählers

Der Aufbau ist kompakt und nutzt die I²C-Pins des OLED-Displays und deines Arduinos, um die Verkabelung zu vereinfachen. Der Durchflussmesser benötigt (neben 5V und Erde) nur eine Verbindung. Auf der folgenden Skizze verwende ich einen Arduino Nano, du kannst aber auch einen UNO oder jeden anderen Arduino oder ESP32/8266 verwenden.



Hier noch einmal die Verbindungen im Detail:

Verbindung des Durchflussmessers:

- **VCC**: An den 5V-Pin des Arduinos
- **GND**: An einen GND-Pin des Arduinos
- **Signal**: An den digitalen Pin D2 des Arduinos. Dieser Pin ist für Interrupts geeignet, was eine präzise Messung der Impulse ermöglicht.

Verbindung des OLED-Displays:

- **VCC**: An den 3.3V-Pin des Arduinos
- **GND**:An einen GND-Pin des Arduinos
- **SDA (Daten)**: An den SDA-Pin des Arduino. Beim Nano und UNO ist das Pin A4.
- **SCL (Takt)**: An den SCL-Pin des Arduinos. Beim Nano und UNO ist das Pin A5.

Die benötigten Bibliotheken

Für dieses Projekt benötigst du zwei Bibliotheken für das OLED-Display. Suche hierfür im Bibliotheksverwalter nach **Adafruit SSD1306**. Bei der Installation wirst du gefragt, ob du auch die Bibliothek **Adafruit GFX Library** installieren möchtest – bestätige das bitte mit **Ja**.

Für den Durchflussmesser benötigst du in der Regel keine Bibliothek – seine Impulse verarbeitest du selbst im Sketch.

Der Sketch für den Wasserzähler

Der folgende Code liest die Impulse des Durchflussmessers und zeigt die gemessene Wassermenge auf dem OLED-Display an. Im Folgenden gehe ich von einer Wassermenge von 2,25ml pro Impuls aus. Ich empfehle dir jedoch, deinen Sensor selbst zu kalibrieren.

```
//Wasserverbrauch messen mit einem Durchflussmesser
//polluxlabs.net

#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

// Deklariere die Display-Parameter für ein 0,91'' Display
#define SCREEN_WIDTH 128 // OLED-Breite in Pixel
```

```
#define SCREEN_HEIGHT 32 // OLED-Höhe in Pixel

// Deklariere ein OLED-Display Objekt
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

// Pin für den Durchflussmesser
const int sensorPin = 2;

// Variable für die Impulszählung. `volatile` ist wichtig für Interrupts.
volatile long impulse_count = 0;

// Die kalibrierte Konstante für deinen Sensor (2.25 ml/impuls)
const float ML_PER_IMPULSE = 2.25;

void setup() {
  // Startet die serielle Kommunikation zur Fehlersuche
  Serial.begin(9600);

  // Initialisiert das OLED-Display
  if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println(F("SSD1306-Zuweisung fehlgeschlagen"));
     for (;;); // Endlosschleife, wenn das Display nicht gefunden wird
  }
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(SSD1306_WHITE);
  // Initialisiert den Sensor-Pin als Eingang
  pinMode(sensorPin, INPUT);
   // Richtet einen Interrupt ein, der die Funktion count_impulses bei jedem Impuls aufruft
      attachInterrupt(digitalPinToInterrupt(sensorPin), count_impulses, RISING);
}

void loop() {
  // Eine Kopie des Zählers erstellen, um race conditions zu vermeiden
```

```
  long current_impulses = impulse_count;

  // Berechnung des Wasserverbrauchs
  float water_volume_ml = current_impulses * ML_PER_IMPULSE;

  // Löscht den Display-Inhalt
  display.clearDisplay();

  // Zeigt den Titel an
  display.setTextSize(1);
  display.setCursor(0, 0);
  display.println("Wasserverbrauch:");

  // Zeigt den Wert an
  display.setTextSize(2);
  display.setCursor(0, 16);
  display.print(water_volume_ml);
  display.print(" ml");

  // Aktualisiert das Display
  display.display();

  // Kleine Pause, um das Display nicht zu überfordern
  delay(100);
}

// Interrupt-Service-Routine (ISR)
// Diese Funktion wird automatisch bei jedem Impuls aufgerufen
void count_impulses() {
  impulse_count++;
}
```

Kalibrierung des Sensors

Der Wert in der Zeile **`const float ML_PER_IMPULSE = 2.25;`** ist entscheidend für die Genauigkeit deiner Messung. Je nachdem, welches Modell du verwendest oder auch, wenn dein Sensor nicht präzise misst, kann dieser Wert abweichen. Um die richtige

Wassermenge pro Impuls zu ermitteln, gehe wie folgt vor:

1. **Vorbereitung**: Halte einen Messbecher mit bekanntem Volumen (z. B. 500 ml) bereit. Lade den obenstehenden Sketch auf deinen Arduino.
2. **Nullstellung**: Starte den Sketch neu, indem du den Reset-Knopf auf deinem Arduino drückst. Das Display sollte **0.00 ml** anzeigen.
3. **Messung**: Lasse genau 500 ml Wasser konstant schnell durch den Sensor in den Messbecher laufen.
4. **Wert ablesen**: Warte, bis die Messung auf dem Display nicht mehr steigt. Notiere den angezeigten Wert (z.B. 650 ml).
5. **Neuer Menge**: Berechne die korrekte Menge Wasser pro Impuls: **2,25ml x (500ml / 650ml) = 1,73ml/Impuls**
6. **Code anpassen**: Ändere die Zeile im Sketch zu **const float ML_PER_IMPULSE = 1.73;**.
7. **Testen**: Lade den Sketch erneut hoch und wiederhole den Test mit 500 ml Wasser. Der angezeigte Wert sollte nun sehr nah an 500 ml liegen.

Um das Ergebnis genauer zu machen, kannst du auch mehrere Messungen durchführen und einen Mittelwert finden. Es kann durchaus sein, dass z.B. die Fließgeschwindigkeit des Wassers Auswirkungen auf die Zahl der gemessenen Impulse hat.

So funktioniert der Sketch

Lass uns schnell einen Blick auf die wichtigsten Teile des Sketchs werfen:

1. Die Setup-Funktion

Der Befehl **attachInterrupt()** ist hier das Herzstück. Anstatt im Loop ständig zu prüfen, ob ein Impuls vom Sensor kommt,

richtest du hier einen sogenannten **Hardware-Interrupt** ein. Das bedeutet, dass der Arduino-Chip sofort seine aktuelle Aufgabe unterbricht und direkt zu der kleinen Funktion **count_impulses()** springt, sobald er ein Signal am Pin D2 empfängt. Das macht die Messung extrem reaktionsschnell und präzise.

2. Der Loop: Die Hauptlogik

Dieser Teil wird immer wieder wiederholt.

- **Impulse auslesen**: **long current_impulses = impulse_count;** kopiert den Wert des Volatile-Zählers. Das ist eine Schutzmaßnahme. Da der Zähler **impulse_count** jederzeit von der Interrupt-Funktion geändert werden könnte, sichern wir den Wert, bevor wir damit rechnen, um Fehler zu vermeiden.
- **Berechnung**: **float water_volume_ml = current_impulses * ML_PER_IMPULSE;** ist der Kern des Projekts. Hier wird die Anzahl der Impulse mit dem Kalibrierungsfaktor multipliziert, um das tatsächliche Wasservolumen in Millilitern zu erhalten.
- **Display-Update**: Die folgenden Befehle, wie **display.clearDisplay()**, **display.setCursor()** und **display.print()**, kümmern sich darum, die berechneten Daten auf dem OLED-Display darzustellen. Der Befehl **display.display()** sendet am Ende die gesammelten Daten an das Display, um die Anzeige zu aktualisieren.
- **Pause**: **delay(100);** sorgt für eine kurze Pause. Das ist wichtig, um zu verhindern, dass das Display zu schnell aktualisiert wird und um die CPU-Ressourcen des Arduinos zu schonen.

3. Die Funktion count_impulses(): Der Interrupt

Dies ist die kleine, aber sehr wichtige Funktion, die durch den Interrupt-Befehl ausgelöst wird. Sie wird nicht vom **Loop**

aufgerufen, sondern springt automatisch in Aktion, sobald der Sensor einen Impuls sendet. Ihre einzige Aufgabe ist es, den Zähler **impulse_count** um eins zu erhöhen.

Wie geht es weiter?

Du kannst du nun den Wasserverbrauch messen und auf einem Display anzeigen. Eine interessante Weiterentwicklung könnte z.B. ein LED-Ring sein, der anzeigt, wie sehr du dich beim Händewaschen einem festgelegten Maximalverbrauch näherst. Oder du misst in deinem Garten den Wasserverbrauch und speicherst die Daten z.B. bei Adafruit IO in der Cloud.