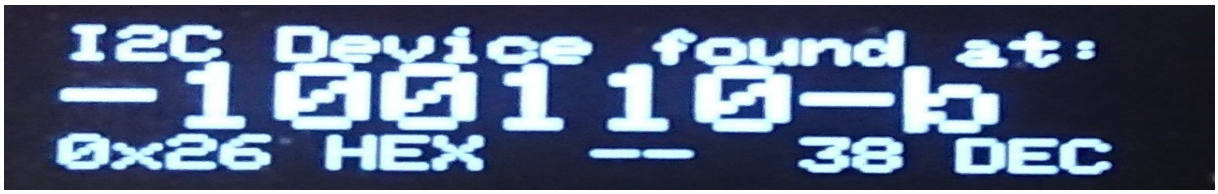


Ein praktischer und kostengünstiger I2C Scanner - Reihe „Multitool“ Teil 1

Hallo und willkommen zu einer neuen Reihe rund um unseren Arduino! In dieser Reihe werden wir uns über insgesamt 3 Teile ein kleines kostengünstiges Elektronik Multitool bauen, das es ermöglicht IC2 Adressen aus einem angeschlossenen I2C Bauteil herauszulesen, (I2C Scanner), eine Rechteckspannung von 5 Volt in variabler Frequenz auszugeben und Spannung im Bereich von 0-5 Volt messen kann. In unserem ersten Teil werden wir einen I2C Scanner bauen. Doch warum schon wieder einen I2C - Scanner? Auslöser war, dass ich für ein anderes Projekt dringend eine Bestimmung der IC2 Adresse eines Bausteines brauchte, jedoch nicht so einfach ein Datenblatt für diesen Baustein im Internet fand. Also musste ein selbstgebauter I2C Scanner her. Auf nun folgenden Suche nach einem kostengünstigen Open Source IC2 Scanner waren die selbstaufgelegten Vorgaben für diesen Scanner die folgenden:

1) ***Einfachheit***: Kein Anschluss an einen PC benötigt. Für den Betrieb oder das Ablesen der I2C Adressen ist kein PC nötig.

2) ***Minimalistisch***: So kleine Bauteile wie möglich. (Unterbringung in einem kleinen Gehäuse)

3) ***Kosteneffizienz:*** Keine unnötig teure Hardware.

4) ***Hardwareeffizienz***: keine "brachliegenden" Ressourcen der Hardware wie WLAN, Bluetooth etc.

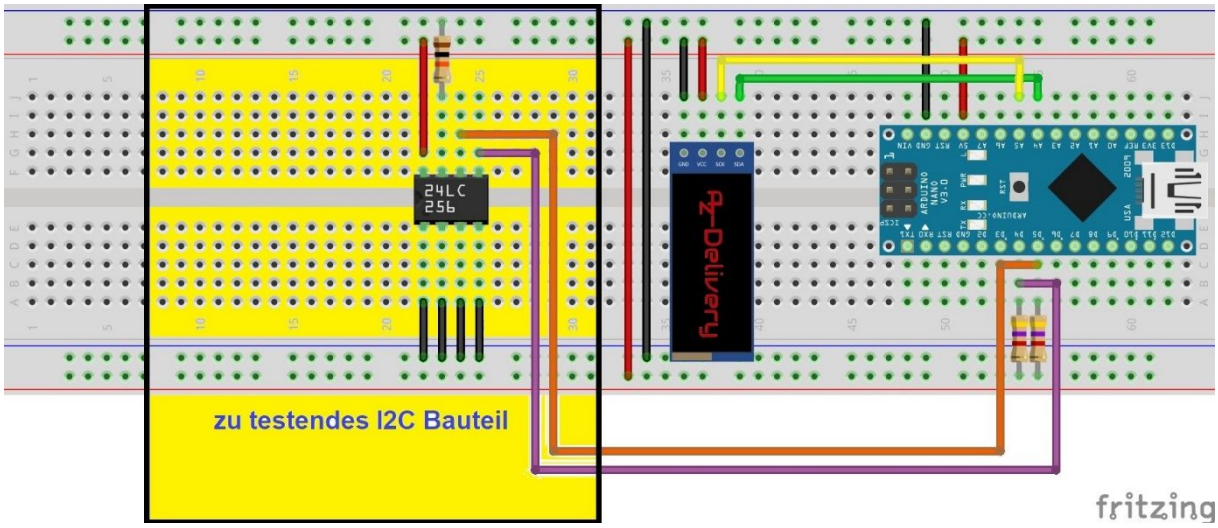
5) ***Sparsam:*** Betrieb mit einem Akkupack möglich.

Ein System das alle die o.g. Kriterien für einen IC2 Scanner erfüllt, konnte ich nicht jedoch nicht finden. Also hieß es: Selbst machen und so ist dieser Blog entstanden. Kommen wir zunächst zu einigen Überlegungen:
Das erste Ziel, ohne PC auszukommen, kann nur mit einem eigenen Display erreicht werden, auf dem die Daten direkt angezeigt werden können. Durch die Vorgabe eines minimalistischen Designs, auf dem nur so wenig wie möglich Platz zur Verfügung steht, blieb nach Sondierung der Displays auf der AZ-Delivery Seite nur das O-Led 128x64 Pixel in der engeren Auswahl. Dieses Display benötigt aber seinerseits schon einen I2C Bus um zu funktionieren. Was nun? Prozessoren mit 2 parallelen I2C Bussen "on Board" wie sie der ESP32 haben, fielen aber aufgrund der überzogenen Hardware (Bluetooth, WLAN, großer Speicher) für dieses Projekt durch. Ebenso aus gleichem Grunde der ESP8266 (Kosteneffizienz und Hardwareeffizienz).Blieb also nur der Arduino Nano für das Projekt übrig, den wir mit einem kleinen "Trick" auch nutzen können: Wir emulieren per Software einen weiteren I2C Port, der für das Scannen benutzt wird (Im folgenden Software I2C genannt).

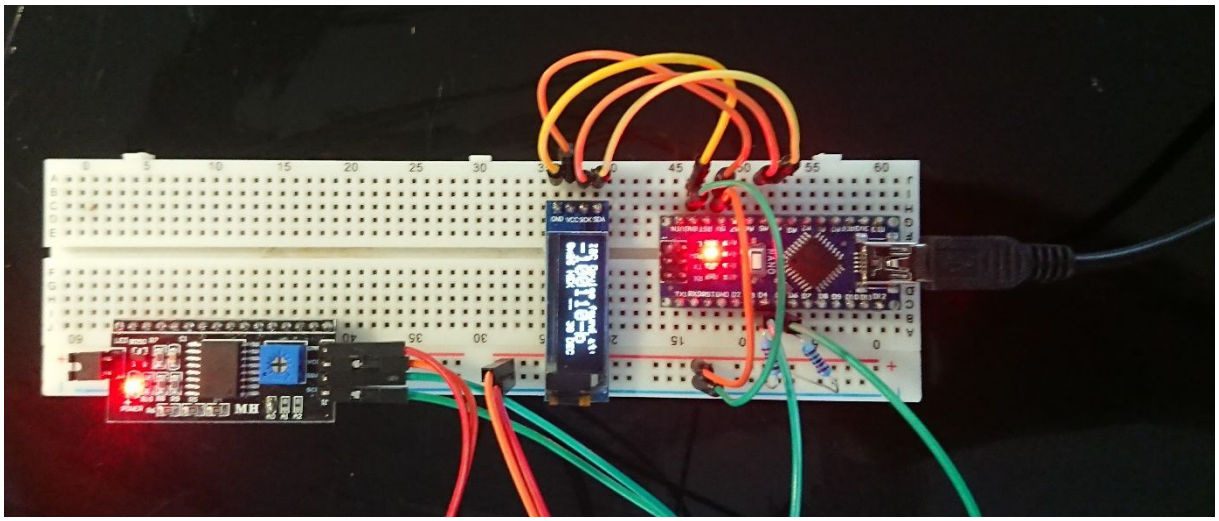
Der Hardware I2C Bus wird für die Ansteuerung des O-Led Displays genutzt. Glücklicherweise gibt es bereits eine fertige Bibliothek um einen I2C Bus in Software zu emulieren. Diese heißt „Soft Wire“ und ist von Steve Marple.
Die Bibliothek kann von hier heruntergeladen werden:
https://github.com/stevemarple/SoftWire
Weitere Informationen zu dieser Bibliothek finden sich auch auf der Seite *https://www.arduinolibraries.info/libraries/soft-wire*
Bevor also mit unserem heutigen Projekt begonnen werden kann, muss erst diese Bibliothek in bekannter Weise installiert werden. Für den I2C Scanner aus dem heutigen Blog benötigen Sie darüber hinaus noch folgende Teile aus dem Shop:

Anzahl	Bezeichnung	Anmerkung
1	Arduino Nano	
1	0,91 Zoll OLED I2C Display 128 x 32 Pixel	
2	4,7 KOhm Widerstände	

Danach kann die Hardware gemäß folgender Fritzing-Zeichnung aufgebaut werden:



Beachten Sie die 4,5 K-Ohm Pullup-Widerstände für die Software I2C Schnittstelle. Diese werden für die Funktion benötigt. Aufgebaut auf einem Breadboard und in Betrieb sieht das ganze nun so aus:



Nach dem Aufbau kann folgender Code hochgeladen werden:

```
// Tobias Kuch 2020 GPL 3.0  tobias.kuch@googlemail.com
// https://github.com/kuchto

/*
Dieser kurze Sketch durchsucht den I2C-Bus nach Geräten. Wenn ein Gerät gefunden wird, wird es auf einem 128x64 OLed Display
mit seiner Adresse in den Formaten Binär, Hexadezimal und Dezimal angezeigt.
*/

#include <Wire.h>
#include <SoftWire.h>
#include <AsyncDelay.h>
#include "SSD1306Ascii.h"
#include "SSD1306AsciiWire.h"

#define oLed_I2C_ADDRESS 0x3C
#define SoftSDA 4
#define SoftSCL 5

const uint8_t firstAddr = 1;
const uint8_t lastAddr = 0x7F;

SSD1306AsciiWire oled;
SoftWire sw(SoftSDA, SoftSCL);

//------------------------------------------------------------------------------

uint8_t I2C_Device_found_at_Address= 0;
uint16_t Round = 0;
bool AlreadyScan = false;

void setup()
{
  Wire.begin();
```

```
  Wire.setClock(400000L);
  sw.setTimeout_ms(200); // Set how long we are willing to wait for a device to respond
  oled.begin(&Adafruit128x32, oLed_I2C_ADDRESS);
  oled.setFont(Adafruit5x7);
  oled.clear();
  oled.set1X();
  oled.clear();
  oled.setCursor(0, 0);
  oled.print("Scanning I2C Bus...");
  oled.setCursor(0, 1);
  oled.print("Initital Scan.");
}
//------------------------------------------------------------------------------

bool scanI2C()
{
bool detected=false;
for (uint8_t addr = firstAddr; addr <= lastAddr; addr++)
  {
  delayMicroseconds(50);
  uint8_t startResult = sw.llStart((addr << 1) + 1); // Signal a read
  sw.stop();
  if ((startResult==0) & (I2C_Device_found_at_Address != addr)) // New I2C Device found
    {
    Round = 0;
    detected=true;
    I2C_Device_found_at_Address = addr;
    AlreadyScan = false;
    oled.set1X();
    oled.setCursor(0,0);
    // oled.clear();
    oled.print("I2C Device found at:  ");
    oled.setCursor(0,1);
    oled.set2X();
    oled.setCursor(0,1);
    oled.print("-");
    oled.print(addr,BIN);
    oled.print("-b    ");
    oled.set1X();
    oled.setCursor(0,3);
    oled.print("0x");
    if(addr<16) oled.print("0");
    oled.print(addr,HEX);
    oled.print(" HEX  --  ");
    if(addr<10) oled.print("0");
    oled.print(addr,DEC);
    oled.print(" DEC ");
    delay(100);
    break;
    }  // Device Found END
  }  // Scan Round END

if (!detected)
```

```
  {
  I2C_Device_found_at_Address = 0;
  Round++;
  }
return detected;
} // Function END

void loop()
{
 if ((!scanI2C()) && (Round > 2))
  {
   if (!AlreadyScan)
    {
     AlreadyScan = true;
     oled.clear();
     oled.setCursor(0, 0);
     oled.print("Scanning I2C Bus...");
    }
   Round = 0;
   I2C_Device_found_at_Address = 0;
  }
 delay(200);
}
```

Sobald nun ein I2C adressierbarer und funktionsfähiger Baustein an den Port 5 (SCL) und Port 4 (SDA) angeschlossen wird, wird dessen Adresse umgehend im O-Led Display in den verschiedenen Zahlensystemen Binär, Dezimal und Hexadezimal angezeigt. Ich wünsche Ihnen viel Spaß beim Bauen und beim Verwenden des I2CScanners. Alle Projekte und Informationen meiner Blogs finden Sie auch unter https://github.com/kuchto.