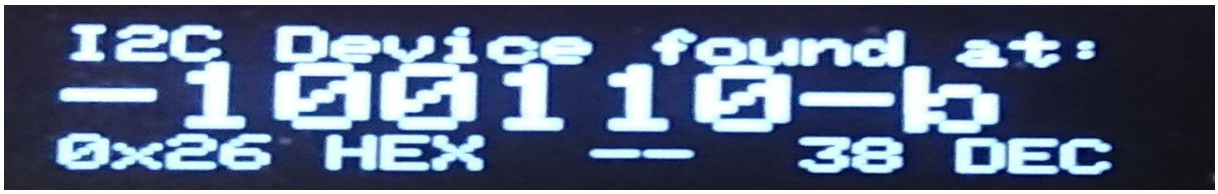


Ein einfacher und kostengünstiger Frequenzgenerator -

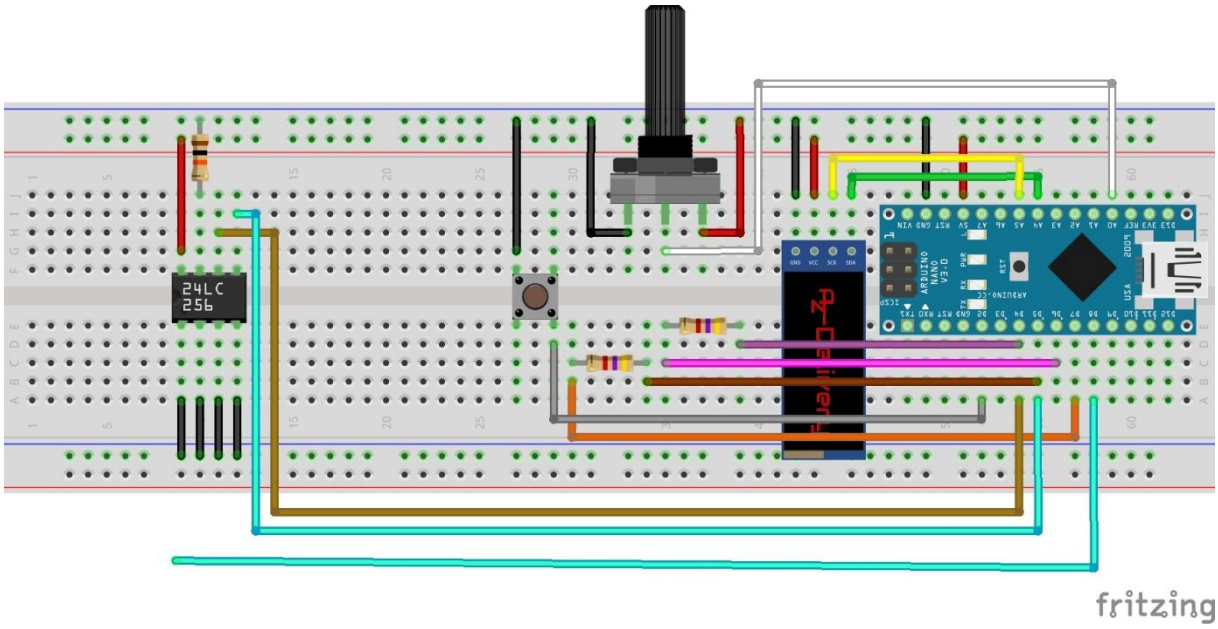
Aus der Reihe „Multitool“ Teil 2

Hallo und willkommen zum zweiten Teil unserer Reihe rund um unser "Mini Tool". Heute spendieren wir unserem Mini Tool zusätzlich zu der Verbesserung der Verwendung des I2C Scanners eine weitere Funktion: Ein Frequenzgenerator mit einem festen Tastgrad von 0,5 (symmetrischer Puls), einstellbar in einem Frequenzbereich von ca. 200 Hz bis ca. 25 Khz. Die Amplitude beträgt 5 Volt, die Effektivspannung Uev ca. 2.5 Volt. Der Puls steht am Pin 8 des Arduinos gegen Masse (GND) des Arduino Nanos für eigene Anwendungen oder Tests zur Verfügung. Das Einstellen der Frequenz in dem o.g. Bereich an Pin 8 des Arduinos erfolgt durch ein 10 K-Ohm Drehpotentiometer, das an den Analog Eingang A0 des Arduinos angeschlossen ist. Das Umschalten zwischen dem Modus "I2C Busscanner" und "Frequenzgenerator" und zurück, erfolgt durch Drücken des Tasters T1.Als weiteres "Bonbon" im heutigen Teil erfolgt eine Verbesserung der Funktion des IC2 Scanners. Beim Anschließen an einen I2C Prüfling muss nun NICHT mehr auf den korrekten Anschluss von den Datenleitungen SDA und SCL geachtet werden. Die Adressermittlung erfolgt jetzt unabhängig der Datenleitungsbeschaltung und wird, wie gewohnt, direkt im Display in den Formaten Binär, Dezimal und Hexadezimal angezeigt.

Um die neuen Funktionen implementieren zu können, benötigen wir folgende leicht erweiterteHardware:

Anzahl	Bezeichnung	Anmerkung
1	Arduino Nano	
1	0,91 Zoll OLED I2C Display 128 x 32 Pixel	
2	4,7 KOhm Widerstände	
1	10-KOhm Präzisions –Drehpotentiometer	
1	Taster	

Danach kann die Hardware gemäß folgender Fritzing-Zeichnung Schaltung wie folgt erweitert werden:



Bitte beachten Sie beim Zusammenbau die etwas geänderte Schaltung der beiden Pullup Widerstände. Diese sind nun nicht mehr direkt an Vcc (5 Volt) angeschlossen, sondern werden ihrerseits auch wieder durch Ports gesteuert. Warum dies Sinn macht, erfahren Sie im nächsten Teil dieser Reihe.

Bevor es nun an das Hochladen des Codes geht, stellen Sie bitte sicher, dass bereits die Bibliothek „Soft Wire“ von Steve Marple in Ihrer IDE installiert ist. Die Bibliothek kann andernfalls von hier heruntergeladen werden. Weitere Informationen zu dieser Bibliothek finden sie auch auf dieser Seite.

Nun kann folgender Code auf den Arduino Nano geladen 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 SoftI2CA 4
#define SoftI2CB 5
#define POTAConfig 6
```

```
#define POTBConfig 7


#define ModeSwitch 2     // Pin Modus Taster
#define SQAREWaveOutputPin 8
#define POTENTIOMETER A0


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

SSD1306AsciiWire oled;
SoftWire sw(SoftI2CA, SoftI2CB);
SoftWire sw_rev(SoftI2CB, SoftI2CA);

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

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

byte SelectedMode = 0;
int oldvaluea = 0;


ISR(TIMER1_COMPA_vect){ //Timer1 interrupt. Schaltet Pin 8 um.
 // Weitere Infos auf: https://www.mikrocontroller.net/articles/AVR-Tutorial:_Timer
 digitalWrite(SQAREWaveOutputPin,!(digitalRead(SQAREWaveOutputPin)));// Aufruffrequenz
maximal: 2.604 Khz
 //Frequenz an Pin 8(Hz) = (Arduino clock speed 16,000,000Hz) / (prescaler * (compare match
register + 1)) / 2
}

void setup()
{
 pinMode(ModeSwitch,INPUT_PULLUP);
 pinMode(SQAREWaveOutputPin, OUTPUT);
 pinMode(SoftI2CA, OUTPUT);
 pinMode(SoftI2CB, OUTPUT);
 pinMode(POTAConfig, OUTPUT);
 pinMode(POTBConfig, OUTPUT);
 digitalWrite (POTAConfig, HIGH);
 digitalWrite (POTBConfig, HIGH);
 Serial.begin(9600);
 cli();                         //stoppe alle Interrupts
 TCCR1A = 0;                    // set entire TCCR1A register to 0 TCCR - Timer/Counter Control
Register
 TCCR1B = 0;                    // Setze Timer/Counter Control Register TCCR1B auf 0
 TCCR1B |= (1 << WGM12);            // Schalte Clear Timer on Compare (CTC) Modus ein
 // TCCR1B |= (1 << CS12) | (1 << CS10);  // Setze CS10 und CS12 Bit auf 1 für den 1024 Prescaler.
Maximalfrequenz: 7.812 Khz
```

```
 TCCR1B |= (1 << CS12);              // Setze CS12 Bit auf 1 für den 256 Prescaler.
 TCNT1  = 0;                       // Initialisiere Zähler/Zeitgeber Register Wert auf 0
 OCR1A = 130;//  Aufruffrequenz Timer 1  241 Hz * 2
 TIMSK1 |= (1 << OCIE1A); // Erlaube Timer compare interrupt TIMSK - Timer/Counter Interrupt
Mask Register
 sei();//allow interrupts
 Wire.begin();
 Wire.setClock(400000L);
 sw.setTimeout_ms(200); // Set how long we are willing to wait for a device to respond
 sw_rev.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(50);
   break;
   } // Device Found END
  } // Scan Round END

if (!detected)
  {
  I2C_Device_found_at_Address = 0;
  Round++;
  }
return detected;
} // Function END

bool scanI2C_rev()
{
bool detected=false;
for (uint8_t addr = firstAddr; addr <= lastAddr; addr++)
  {
  delayMicroseconds(50);
  uint8_t startResult = sw_rev.llStart((addr << 1) + 1); // Signal a read
  sw_rev.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(50);
    break;
    } // Device Found END
  } // Scan Round END
```

```
if (!detected)
  {
  I2C_Device_found_at_Address = 0;
  Round++;
  }
return detected;
} // Function END

void CheckMode()
{
bool PinStatus = digitalRead(ModeSwitch);
if ((PinStatus == LOW) && !(Alreadypressed))
  {
  SelectedMode++;
  if  (SelectedMode > 1) { SelectedMode = 0; }
  Serial.print("Mode changed to: ");
  Serial.println(SelectedMode);
  delay(50);
  Alreadypressed = true;
  Modeswitched = true;
  } else if ((PinStatus == HIGH) && (Alreadypressed))
  {
  Alreadypressed = false;
  }
}

void loop()
{
CheckMode();
if (SelectedMode == 0)
  {
  if (Modeswitched)
    {
    oled.clear();
    }
  Modeswitched = false;
  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;
    }
  if ((!scanI2C_rev()) && (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(50);
  }

if (SelectedMode == 1)
  {
  if (Modeswitched)
   {
   oled.clear();
   oled.setCursor(0, 0);
   oled.print("Square Wave Generator");
   }
  Modeswitched = false;
  int Gb = analogRead(POTENTIOMETER);
  byte valuea = Gb; // map(Gb,0,1023,0,300);
  if (valuea != oldvaluea)
   {
   oldvaluea = valuea ;
   cli();//stop interrupts
   OCR1A = valuea;
   if ( TCNT1 > OCR1A )
    {
    TCNT1 = OCR1A -1;//initialize counter value to 0
    }
   sei();//allow interrupts
   }
  }
// END MainLoop
}
```

Sobald nun ein I2C adressierbarer und funktionsfähiger Baustein beliebig mit seinen Bus Ports an den Port 5 und Port 4 angeschlossen wird, und das Mini Tool im Modus „I2C Scanner“ betrieben wird, wird dessen Adresse umgehend im O-Led Display angezeigt. Der Frequenzgenerator kann für beliebige eigene Anwendungen genutzt werden, Ich wünsche Ihnen viel Spaß beim Bauen und beim Verwenden des kleinen Tools. Alle Projekte und Informationen meiner Blogs finden Sie, wie immer, auch unter https://github.com/kuchto.