

Medizinisch-Technische Informatik an der Hochschule Reutlingen:

„... das Bachelor Studium mit der Eintrittskarte in die Erfolgswelt der Medizintechnik“

<http://www.inf.reutlingen-university.de/studium/bachelor/medizinisch-technische-informatik/>

Vorlesungsbegleitblatt – Eingebettete Systeme und Robotik: SS2020

Workshop US-Sensor SR04

Vorwort

Im Rahmen des diesjährigen Projektes kann Ultraschall Abstandsmesssensor vom Typ SR04 zum Einsatz kommen. Der hier beschriebene Workshop zeigt, wie sich eine API für den Sensor entwickeln lässt, die später in anderen Projekten wiederverwendet werden kann.

Hintergrund

Das Messen von Abständen ist eine häufige Aufgabe bei der Implementierung von embedded Systemen, z.B. bei der Implementierung einer Einparkhilfe, eines Füllstandssensors oder beim Aufbau eines Abstandswarngerätes.

Die Messung einer Entfernung per Ultraschall ist ein sehr einfaches, preiswertes und robustes Verfahren. Entsprechend häufig sind in der sogenannten „maker-Szene“ Projekte zu finden, bei denen entsprechende Sensoren zum Einsatz kommen.

Als ein Standardbauteil, welches vorwiegend in China produziert wird, hat sich der Arduino Ultraschall Abstandssensor SR04 etabliert, den wir auch in diesem Projekt zum Einsatz bringen werden.

Technische Grundlagen

Das Messverfahren beruht darauf, dass ein Ultraschallsender einen Burst¹ aussendet und anschließend gewartet wird, bis das Echo des Bursts, welches von einem Gegenstand in der Nähe zurückgeworfen wird, wiederum beim Sender eintrifft. Die Laufzeit des Bursts

¹ Ein Burst bezeichnet eine Folge von mehreren Schwingungen, zu denen der Ultraschallsender angeregt wird. Die Anregung des Senders erfolgt vorzugsweise mit der Resonanzfrequenz (Eigenfrequenz) des Senders, da auf diese Art ein Signal mit einer besonders hohen Amplitude generiert werden kann.

ermöglicht die Umrechnung in einen Abstand, da die Schallausbreitungsgeschwindigkeit in Luft als annähernd konstant betrachtet werden darf.

Hierbei muss beachtet werden, dass vom Echo immer nur „die erste Welle“ ausgewertet werden kann, also die Reflexion des Gegenstandes, der den geringsten Abstand zum Sender hat. Alle anderen Reflexionen gehen in einem Gemisch von Echos unter, so dass nicht mehr ohne weiteres festgestellt werden kann, welches Echo von welchem Gegenstand verursacht wird.

Bevor ein neuer Burst ausgesendet wird, müssen alle im Raum vorhandenen Echos soweit abgeklungen sein, dass diese unterhalb der Detektionsschwelle des Empfängers liegen, so dass diese den Empfang des Echos des nächsten Bursts nicht mehr stören können.

Die technische Umsetzung dieses Verfahren beim Sensor SR04 ist in zahlreichen Büchern und Artikeln beschrieben. Der nachfolgende Text stammt von der Seite:

<http://www.mikrocontroller-elektronik.de/ultraschallsensor-hc-sr04/>

(Zugriff: 2020-06-23)



Technische Daten des Ultraschallsensors HC-SR04

Ultraschallsensor HC-SR04	
Betriebsspannung	5V (+/- 10%)
Strombedarf	ca. 2mA pro Messung
Signal Level	TTL-Pegel
max. messbare Entfernung	ca. 3m
min. messbare Entfernung	ca. 2 cm
Maximalen Messungen pro Sekunde	50
Ultraschallkapseln	zwei (Sender und Empfänger)
Genauigkeit	ca. 3mm
Pinbelegung	Pin 1: VCC Pin 2: Trigger Pin 3: Echo Pin 4: GND

Ein englisches Manual findet sich unter folgender Adresse:

https://docs.google.com/document/d/1Y-yZnNhMYy7rwhAgyL_pfa39RsB-x2qR4vP8saG73rE/edit

Das Datenblatt findet man unter folgender Adresse;

<https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf>

Wie erfolgt der Messvorgang beim Ultraschallsensor HC-SR04?

Der Ultraschallsensor HC-SR04 benötigt zunächst eine Versorgungsspannung von 5V, dafür stehen die Anschlüsse +5V (Pin 1) und GND (Pin 4) bereit. Die eigentliche Messung wird über den Anschluss Trigger (Pin 2) gestartet. Der Messvorgang wird durch eine fallende Flanke am Trigger-Eingang ausgelöst. Das vorhergehende High-Signal muss dabei eine Mindestzeit von 10 μ s anliegen.

Der Ultraschallsensor HC-SR04 sendet daraufhin nach ca. 250 μ s ein 40 kHz Burst-Signal für die Dauer von 200 μ s zur eigentlichen Sensorkapsel (Transducer). Danach geht der Ausgang Echo (Pin 3) sofort auf H-Pegel und der Ultraschallsensor wartet auf den Empfang des akustischen Echos. Sobald das Echo registriert wird, fällt der Ausgang auf Low-Pegel. Nach 20 ms kann die nächste Messung erfolgen.

Um die genaue Entfernung zu ermitteln, muss ein Mikrocontroller also lediglich für 10 μ s ein High-Signal an den Trigger-Eingang legen und danach messen wie lange das High-Signal (Wartezeit auf Echo) am Echo-Signal anliegt. Wenn das High Signal länger als 200 ms angelegt war, dann wurde kein Hindernis vom Sensor erkannt (außer Reichweite).

Berechnung der Entfernung anhand der Schallgeschwindigkeit

Auch ein für den Mensch nicht hörbarer Ultraschallton ist ein akustischer Schall. Er verhält sich demnach entsprechend den physikalischen Gesetzen. Der Schall hat eine bestimmte Geschwindigkeit, er legt in der Sekunde 330 Meter zurück. Wenn man also weiß wie lange es dauert bis der Ultraschallsensor sein eigenes Echo empfängt, dann kann man leicht ausrechnen wieviel Meter bzw. auch Zentimeter der Schall zurückgelegt hat. Diese errechnete Entfernung muss man dann allerdings noch durch 2 teilen, denn wir wollen ja nur die einmalige Wegstrecke berechnen und nicht den gesamten Hin- und Rückweg.

Genau genommen ist die Schallgeschwindigkeit nicht ganz genau 330 Meter pro Sekunde groß, denn die Ausbreitungsgeschwindigkeit ist auch abhängig von der Temperatur. Die genauere Berechnung der Ausbreitungsgeschwindigkeit würde so aussehen:

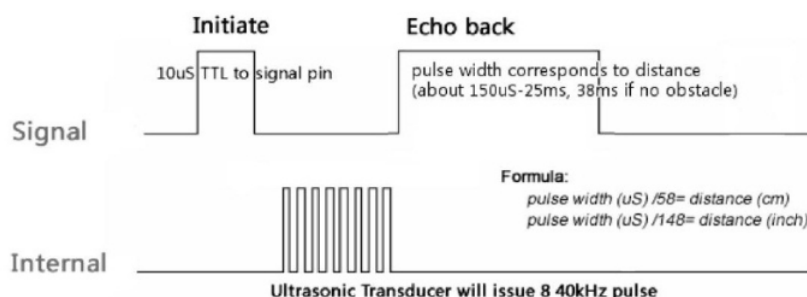
Ausbreitungsgeschwindigkeit (in Luft) = $331,5 + (0,6 * \text{Temp}^\circ)$

Bei einer Raumtemperatur von 20° ergibt sich somit: $331,5 + (0,6 * 20) = 343,5 \text{ m/s}$
Pro gemessener Mikrosekunde wäre der Schall also 0,03434 cm unterwegs gewesen.

Vorüberlegungen

Abschätzungen und Grobdimensionierung

Der zu erfassende zeitliche Puls hat folgende Gestalt:



<https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf>

Der maximal zu messende Abstand beträgt 3 m (siehe oben). Dies entspricht einer Schalllaufzeit von 52ms (Hin- und Rückweg). Um sicherzustellen, dass alle Echos abgeklungen sind, bevor ein neuer Burst ausgesendet, sollte eine Messperiode von ca. 60ms nicht unterschritten werden. D.h. mit dem Sensor ist eine maximale Abtastrate von ca. 16 Hz zu erreichen.

Die Laufzeit des Schalls kann über den Capture Mode eines Timers gemessen werden. Der Timer zählt hierbei die Taktimpulse, die zwischen dem Senden des Bursts und dem Eintreffen des Echos am Takteingang des Zählers eintreffen. Um die bestmögliche zeitliche Auflösung zu implementieren, wird eine möglichst hohe Zählerfrequenz so gewählt, dass innerhalb der Messperiode der Zähler gerade noch nicht überläuft (bei einem Überlauf könnte die Zeitdifferenz zwischen den Zählerständen des Starts und des Endes der Messung nicht mehr eindeutig bestimmt werden).

Soll die Messperiode ca. 100 ms betragen, so ergäbe sich bei einem 16-Bit Zähler eine Taktfrequenz von $2^{16} / 0,1s = 655360 \text{ Hz}$

Da in unserer Anwendung der Zählertakt aus dem SMCLK erzeugt werden soll und dieser auf 1 MHz festgelegt wird, erfolgt die Festlegung des Zählertaktes auf 500 kHz. Dies wird erreicht, wenn der Zähler einen Vorteiler =2 erhält. Die zeitliche Auflösung eines Zählerimpulses läge somit bei $2\mu s$ und dies entspricht einer Entfernung von $2\mu s / 2 \cdot 343 \text{ m/s} = 0,34 \text{ mm}$ (Hin- und Rückweg!). Bei der gewählten Frequenz ist die Genauigkeit der Zeitmessung also weit höher als die Genauigkeit des Sensors selbst (ca. 3mm – siehe oben) und damit eine wichtige Anforderung erfüllt.

Wird eine 10 Hz Abtastrate erwünscht (100ms Periodendauer) so kann für die Erzeugung der Grundfrequenz der Up Mode eines Zählers genutzt (Rückstellung des Zählers beim Überschreiten des in CCR0 hinterlegten Wertes). CCR0 wird somit auf den Wert $100ms / 2\mu s = 50000$ gesetzt.

Zusammengefasst ergeben sich aus den Vorüberlegungen also folgende Anforderungen:

$$f_{\text{Cnt}} = 500\text{kHz}$$

$$f_{\text{Clk SMCLK}} = 1 \text{ MHz}$$

$$\text{CCR0} = 50000 \rightarrow 10 \text{ Hz Wiederholrate der Messung}$$

$$\text{Vorteiler des Zählers: } /2$$

Umrechnungsfaktor Zählerstand \rightarrow Entfernung:

$$\text{distance} = \text{Laufzeit } s \div 2 \cdot 343 \text{ m/s}$$

$$\text{distance} = \text{Cnt} \cdot \frac{1}{500000 \frac{1}{s}} \div 2 \cdot 343 \frac{\text{m}}{\text{s}} = \text{Cnt} \cdot 0,343$$

Wird der Umrechnungsfaktor 343 gewählt, so erhält man den Abstand in mm.

Machbarkeitsbetrachtung

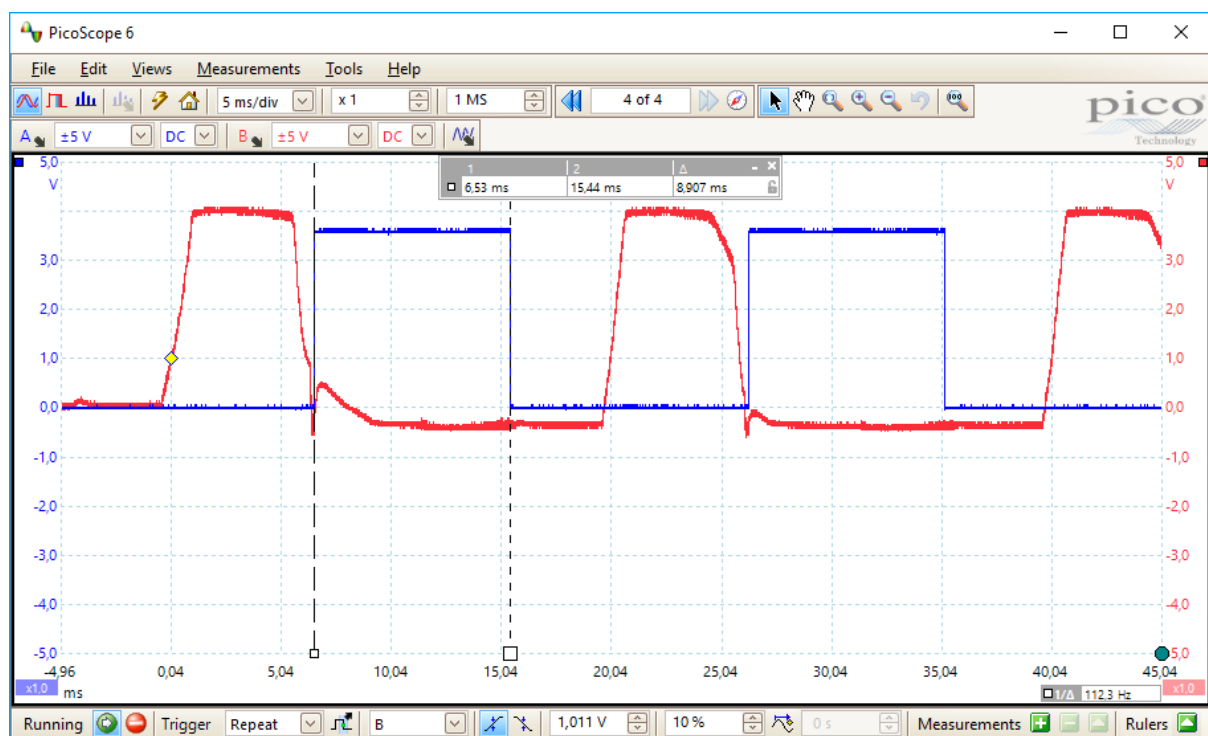
Versorgungsspannung

Laut Datenblatt muss der Sensor mit 5V versorgt werden. Da beim MSP430 die Eingangsspannung an den Portpins nicht größer als V_{CC} sein darf (3,3V), stellt sich die Frage, ob der Sensor auch außerhalb der Spezifikation, bei einer Versorgungsspannung von 3,3V funktioniert?

Hierzu wird folgendes Experiment durchgeführt:

Der Sensor wird an eine 3,3V Versorgungsspannung angeschlossen (z.B. vom Launchpad) und anschließend wird der Trigger-Pin kurz mit dem Finger berührt. Die auf unseren Fingern abgreifbare Störspannung (jeder Mensch ist eine Antenne für die uns umgebenden elektromagnetischen Felder und trägt die Antennenspannung somit auf dem Körper) genügt, um am Sensor eine Messung zu triggern. D.h. es müsste dann auch ein Echo-Signal am Echo-Ausgang des Sensors detektierbar sein. Dies lässt sich mit einem Osci prüfen.

Das nachfolgende Bild zeigt das Verhalten des Sensors bei 3,6V (das ist z.B. die beim Raspberry PI vorhandene Versorgungsspannung). Es ist zu erkennen, dass die Messung tatsächlich auch noch bei 3,6V funktioniert.



Rot: Störsignal des Fingers, mit dem die Messung getriggert wird

Blau: Laufzeit eines Echos (hier: vom Messtisch bis zur Decke)

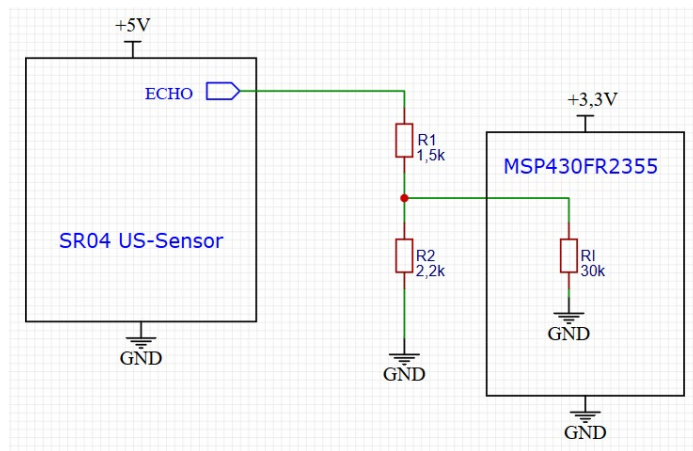
Bei 3,3 V hingegen zeigt der Sensor leider keine Reaktion.

Fazit: Wir müssen bei unserem Launchpad den Sensor tatsächlich mit 5V versorgen und darauf achten, dass die an den MSP430 zurückgegebenen Signale (hier: das ECHO-Signal) den Maximalwert von 3,3V nicht überschreiten.

Anpassung der Datensignale

Während der Sensor mit den vom Launchpad am DIO-Port ausgegebenen Triggersignal an von 3,3V an seinem TRIGGER Eingang klarkommt (dies wird auch bei 5V V_{cc} noch als H-Signal akzeptiert), muss das vom Sensor kommende ECHO Signal von 5V auf einen Wert $< 3,3V$ reduziert werden, um den Digitaleingang des MSP430 nicht zu zerstören. Dies lässt sich über einen Spannungsteiler bewerkstelligen.

Hierbei ergibt sich z.B. folgende Konstellation:



R_1 und R_2 werden so gewählt, dass vom 5V Signal des Sensors nur noch V beim MSP430 ankommen. Der Innenwiderstand des MSP430 Portpin von ca. 30k kann bei der Berechnung vernachlässigt werden, solange R_2 und R_1 vergleichsweise niederohmig sind (d.h. der Strom durch diese beiden Widerstände ist ca. 10 mal so hoch wie der Strom, der über den Innenwiderstand fließt).

Es gilt (unter Vernachlässigung von R_i):

$$V_{in} = \frac{R_1}{R_1 + R_2} \cdot 5V = \frac{2,2k}{1,5k + 2,2k} \cdot 5V = 2,97V$$

Dieser Pegel wird vom MSP430 noch sicher als H-Pegel erkannt.

Design

Einstellung der Timer Grundfrequenz

Der Timer soll mit 500kHz getaktet werden. Dieser Takt wird aus dem SMClk abgeleitet und der Vorteiler des Zählers des Timers muss auf den Wert 2 eingestellt werden.

SMCLK kann über den REFO und den FLL erzeugt werden, um eine ausreichende Genauigkeit für die Messung zu erhalten. (Praktischer Hinweis: ... also mal nach einem Sample Programm suchen, welches einen SMCLK mit 1 MHz aus dem REFO erzeugt). Um die Messung noch genauer zu machen, könnte der REFO auch über den externen Uhrenquarz getaktet werden.

Auswahl des Timers

Festlegung des Capture Eingangs

Bei der Auswahl des Timers ist darauf zu achten, dass das Capture Compare Input (CCI) Pin und das PWM out Pin, über welches das periodisch wiederkehrende TRIGGER Signal ausgegeben wird, über die Pfostenstecker des Launchpads zugänglich ist.

Gewählt wird Timer2_B3, CCI1A (also P5.0 / Pin 26), da dieses Pin nicht von weiteren Funktionen, die man später vielleicht gebrauchen könnte (serielle Schnittstelle, usw.), überlagert ist.

Timer2_B3 muss also für CCR1 in den Capture Mode versetzt werden und über SMCLK getaktet werden.

Festlegung des Trigger Ausgangs

Damit die US-Messung ausgelöst wird, muss in periodischem Abstand das Trigger Signal erzeugt werden (z.B. alle 100ms: somit ist sichergestellt, dass wirklich alle Echos abgeklungen sind, bevor eine neue Messung gestartet wird).

Das Trigger Signal wird als PWM-Output ebenfalls von Timer2 erzeugt. Hierzu wird Timer2 im Up-Mode betrieben und mit 500 kHz getaktet. Soll die Messung alle 100 ms erfolgen, so muss CCR0 mit dem Wert 50000-1 belegt werden.

Das PWM-Signal soll über CCR2 erzeugt werden (CCR1 ist ja schon belegt) und eine Dauer von ca. 20µs haben. Dies entspricht bei einem 500kHz Takt (2µs Periodendauer) einem Zählerstand von 10 Zählerimpulse.

Der Output Mode wird so gewählt, dass der Ausgang am Anfang der Periode auf L steht und 10 Zählerimpulse vor dem Erreichen des in CCR0 hinterlegten Wertes, also beim Erreichen des in CCR2 zu hinterlegenden Wertes, auf H geht. Beim anschließenden Rückstellen des Zählers beim Erreichen von CCR0 geht der Ausgang wieder auf L. Dies entspricht dem SET/RESET mode des PWM-Ausganges.

14.2.5.1.1 Output Example – Timer in Up Mode

The OUTn signal is changed when the timer counts up to the TBxCLn value, and rolls from TBxCL0 to zero, depending on the output mode. Figure 14-12 shows an example using TBxCL0 and TBxCL1.

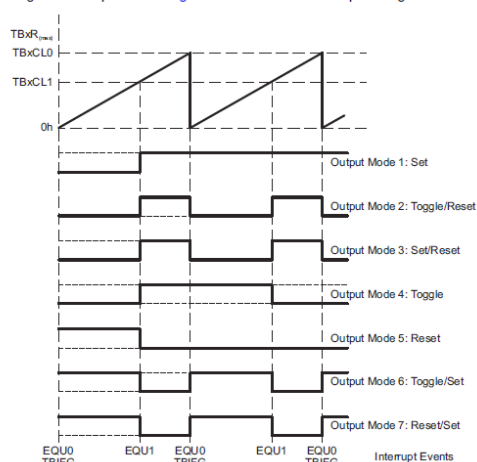
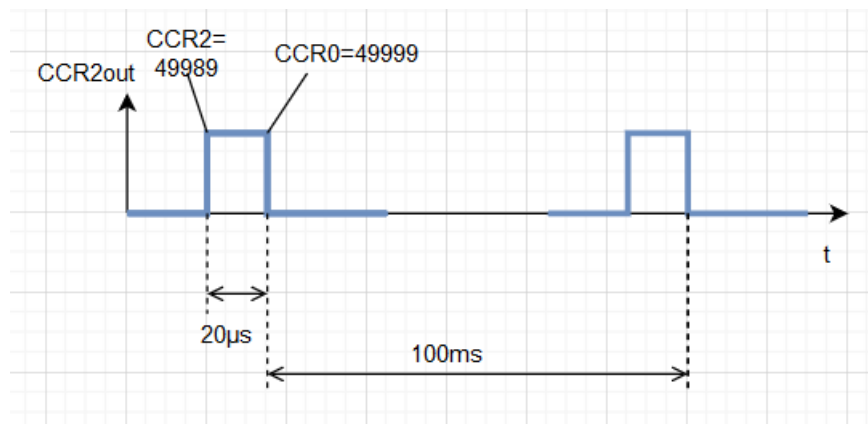


Figure 14-12. Output Example – Timer in Up Mode

Der in CCR2 einzutragende Wert ist also: $50000 - 1 - 10 = 49989$. Der entsprechende Out Mode für TB2.2 ist Mode 3: SET/RESET. Dies ist in der nachfolgenden Abbildung nochmals dargestellt:



Software Entwicklungsplan

Wichtig: Immer nach geeigneten Sample Programmen suchen, bei denen die Grundeinstellungen schon getroffen wurden.

Iteration 1 → Version 0.1

Programm, welches über den REFO einen 1 MHz SMCLK erzeugt

Iteration 2 → Version 0.2

Programm, welches über Timer2_B3 einen Triggerpuls mit der Länge 10µs erstellt und diesen alle 100 ms ausgibt.

Iteration 3 → Version 0.3

Programm, welches über Timer2_B3 den Echopuls des Ultraschallsensors erfasst und in eine Entfernung umrechnet.

Iteration 4 → Version 0.4

Programm, welches die errechneten Daten auf den Backchannel UART des Launchpads ausgibt.

Iteration xy → Version 1.0

Programm, welches die gemittelten Messwerte über den TI GUI-Composer ausgibt.

Die unterschiedlichen Iterationen sollen in das Gitlab Code Repository eingcheckedt werden.

Umsetzung

Die Iterationen sind im Kurs Repository unter folgendem Projektnamen hinterlegt:

esr-projekt_sr04_ultraschall_drv

Das Projekt kann unter folgender Adresse gecclont werden:

https://gitlab.reutlingen-university.de/metib4.4-esr/esr-projekt_sr04_ultraschall_drv.git

Die Angabe der Versionsnummern im Repository stimmt nicht komplett mit dem hier gezeigten Plan überein (...sowas kann vorkommen, ist jedoch nicht sonderlich schlimm, solange noch nicht die finale Version 1.0 veröffentlicht wurde).

Unbedingt beachten:

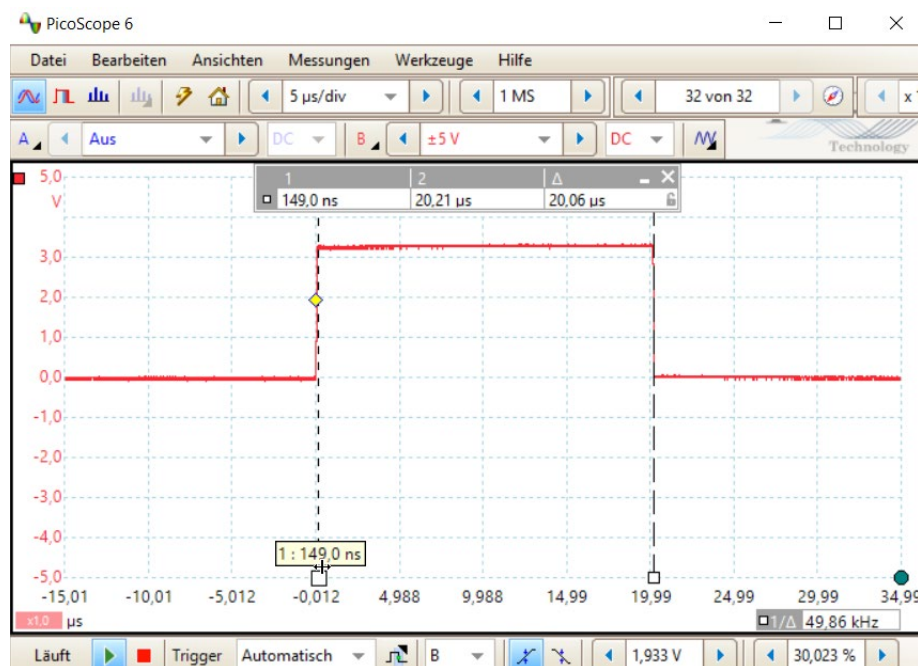
Das ECHO-Signal muss über einen Spannungsteiler auf das Launchpad übertragen werden. Dies erfolgt z.B. unter Verwendung eines Breadboards.

Iteration 2 → Version 0.2

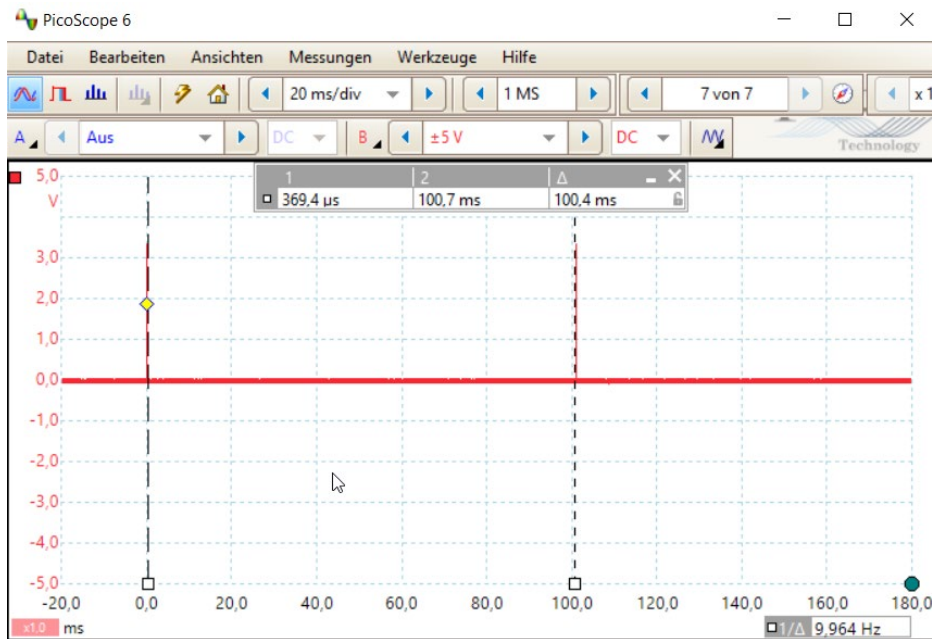
Ziel

Es soll alle 100 ms ein Triggerpuls mit einer Länge von ca. 20 μs ausgegeben werden.

Das nachfolgende Schirmbild des Osci zeigt, dass diese Anforderung erfüllt werden konnte.



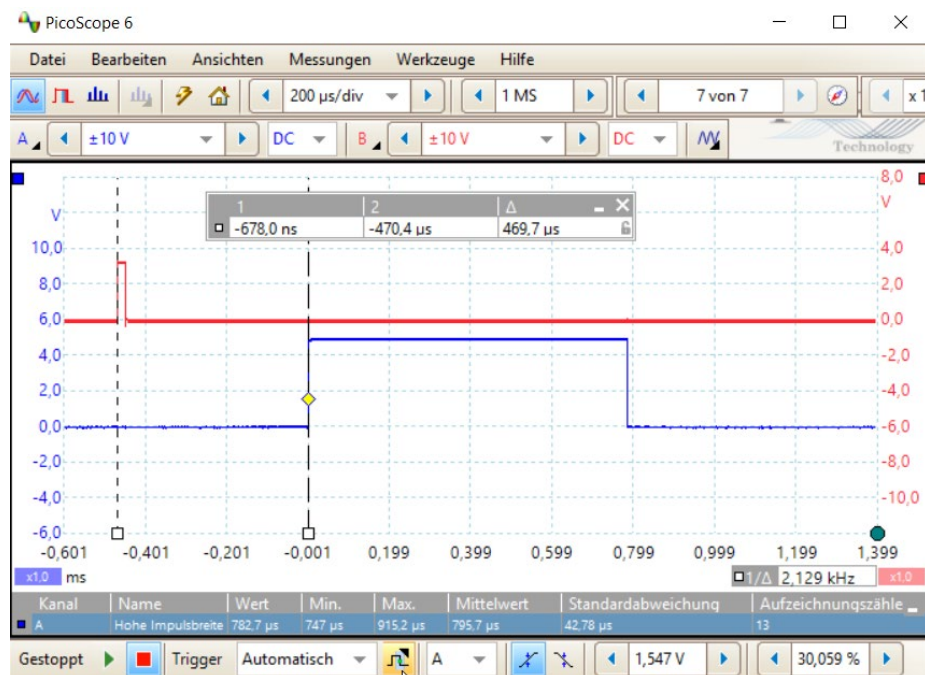
Der TRIGGER Puls hat die gewünschte Dauer von 20 μs



Die gewünschte Periode von 10 Hz wird annähernd exakt eingehalten. Die geringfügige Abweichung könnte von der Ungenauigkeit des REFO herrühren.

Iteration 3 → Version 0.3

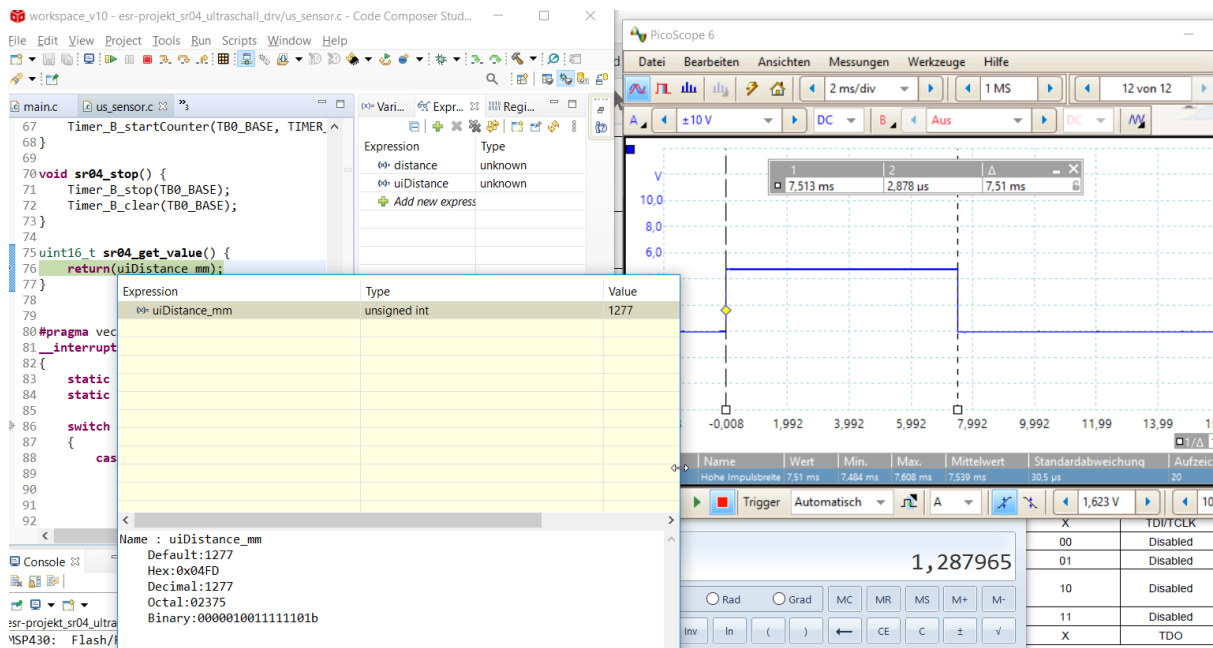
Es soll nun die Dauer des Messpulses per Capture Interrupt gemessen werden. Die Messung beginnt mit der steigenden Flanke und endet mit der fallenden Flanke. Die Ausgabe der Detektion der Laufzeit (Input des Trigger Signals) kann über Pin P1.0 auf die rote LED ausgegeben werden. Bei 10 Hz sollte das Signal sichtbar sein, und zwar umso besser, je größer der Abstand ist (desto länger ist die LED im On-Zustand).



Rot: TRIGGER Puls (ca. 3,3V); Blau: ECHO Signal (ca. 5V)

Man erkennt, dass zwischen der Ausgabe des Trigger Pulses und dem Beginn des Echo Signals (also der Ausgabe des Bursts) ein Abstand von $470\mu\text{s}$ besteht, entgegen der oben gezeigten Beschreibung, wonach dieser Abstand nur ca. $250\mu\text{s}$ betragen sollte.

Das Ergebnis der Messung (im Programm errechneter Zahlenwert) und der über die Hardware ermittelte Wert (Osci + Taschenrechner) kommen zum fast gleichen Ergebnis, womit auch wieder validiert wäre, dass das Programm seinen Zweck erfüllt. Die Abweichung lässt sich dadurch erklären, dass der Messwert stark schwankt und die im Bild dargestellten Messungen nicht zum exakt gleichen Zeitpunkt ausgeführt wurden.



Iteration 4

Ausgabe des Ergebnisses über die serielle Schnittstelle

... das ist Ihr Job :-)

Integration und Test

Hier nicht belegt.

Serienreifemachung

Hier nicht belegt.

Hier könnte dazu gehören, dass die Funktionen zur Ansteuerung des SR04 in ein eigenes Modul ausgelagert und mit einem dokumentierten API versehen werden.