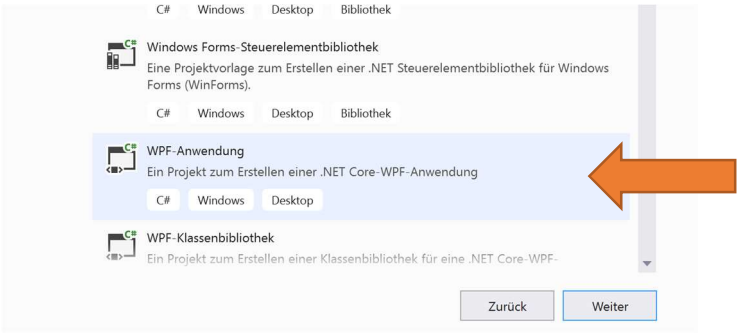
Fachhochschule Kiel
Hochschule für Angewandte Wissenschaften

SPI-Oszilloskop

Vorbereitungsaufgabe: Zeichnen von Polygonen

In der Vorbereitungsaufgabe sollen Sie sich mit weiteren Grafikfunktionen der „Windows Presentation Foundation“ (WPF) vertraut machen. Dazu zeichnen Sie verschiedene Polygone. Gehen Sie wie folgt vor:
Erstellen Sie eine WPF-Applikation.



Fügen Sie vier „Button“-Objekte hinzu und definieren Sie jeweils eine „Click-Callback“ für jeden Button, z.B. durch einen Doppelclick im Designer, um später verschiedene Formen zu zeichnen. Danach ergänzen Sie ein „Canvas“-Objekt ein. Bei der „Canvas“ (englisch für „Leinwand“) handelt es sich um die Zeichenfläche, die Sie bereits kennen. Sie verwenden zum Zeichnen einen Streckenzug („Polyline“). Da diese „Polyline“ innerhalb der „Canvas“ dargestellt werden soll, muss sie ein Unterobjekt von „Canvas“ sein. Daher wird das Objekt „Canvas“ in einen Start und ein Ende-Tag geteilt und die „Polyline“ eingefügt. Führen Sie dies direkt in XAML aus.
Aus

```
<Canvas/>
```

wird dann:

```
<Canvas>
        <Polyline></Polyline>
</Canvas>
```

Sie können wie bisher Eigenschaften der GUI-Elemente anpassen, indem sie Eigenschaften direkt in XAML oder über das Eigenschaftenfenster verändern. Ändern Sie bei der Polyline folgende Eigenschaften:

	Name	Stroke	StrokeThickness
Polyline	poly	Green	3

Wie Sie schon vermuten, steht „StrokeThickness“ für die Stärke der Linie und „Stroke“ für den Pinsel, definiert über eine Standardfarbe.
Geben Sie der Canvas den Namen „myCanvas“.

Die Punkte können Sie über eine „PointCollection“ als einzelne Objekte übergeben. Bilden sie aus den Punkten eine neue „PointCollection“ und fügen Sie die einzelnen Punkte mit „.Add(new Point(x,y))“ hinzu. Die folgende „Button-Callback“ zeichnet beispielsweise ein gleichschenkliges Dreieck, wie in Abbildung 1 zu sehen ist.

```
private void Button_Click(object sender, RoutedEventArgs e)
{
       PointCollection pts = new PointCollection();
       pts.Add(new Point(0, 0));
       pts.Add(new Point(200, 0));
       pts.Add(new Point(100, 200));
       pts.Add(new Point(0, 0));
       poly.Points = pts;
}
```

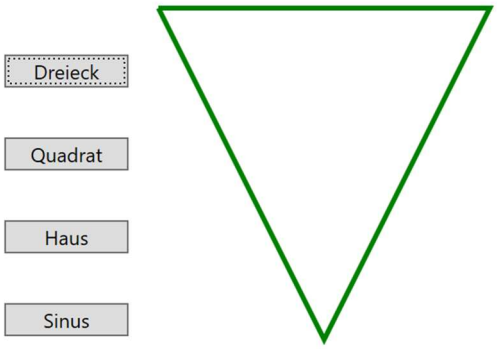
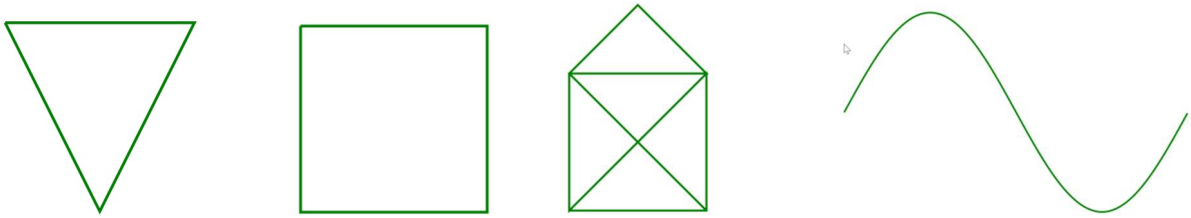


Abbildung 1: Eine gleichschenkliges Dreieck, durch WPF gezeichnet

Testen Sie zunächst diese Zeichnung. Beachten Sie, dass das Dreieck „auf dem Kopf“ steht. Der Grund ist, dass bei allen hardwarenahen Zeichenbefehlen der Koordinatenursprung (0,0) immer in der linken oberen Ecke liegt, daher liegen größer werdende x-Werte wie gewohnt weiter rechts, positivere y-Werte jedoch weiter unten. Implementieren Sie selbst ein Quadrat und ein Haus (vom Nikolaus), um den Umgang mit dem Koordinatensystem zu üben.

Versuchen Sie abschließend mit dem letzten Button ***mit einer for-Schleife*** eine Sinus-Funktion zu plotten. Dabei soll genau eine Periode der Funktion sichtbar sein. Auch wenn Sie bei der „Canvas“ keine Breite (Width) vorgegeben haben, können Sie die aktuelle Breite mit „`myCanvas.ActualWidth`“ abrufen.

Ihr Programm sollte am Ende die folgenden Formen erzeugen können:



Hinweise

1. *Die Sinusfunktion finden Sie unter „Math.Sin(…)“, die Kreiszahl Pi unter Math.PI.*
2. *Sie sollten bei der „Canvas“ auch eine Margin-Eigenschaft ergänzen, damit sie nicht über den Buttons liegt.*
3. *Der Sinus soll nicht „auf dem Kopf“ stehen.*

Fertigen Sie jeweils „Screenshots“ des Programms mit allen Formen an und bringen Sie einen Ausdruck, Handyfotos oder etwas Ähnliches zum Labortermin mit.

Einleitung zur Laboraufgabe

Innerhalb dieser Aufgabe sollen Sie ein Oszilloskop auf einem Embedded-System programmieren. Zu diesem Zweck wurde im IoT-PC am SPI-Bus ein AD(Analog/Digital) -Wandler angeschlossen. Ihre Aufgabe ist das Auslesen dieses Bausteins, die Berechnung der aufgenommenen Spannungspegel, sowie die grafische Ausgabe auf dem Display. Ferner sollen Sie noch eine Funktion zum Triggern der Anzeige des Oszilloskops und eine Anzeige der Signalkenngrößen implementieren.

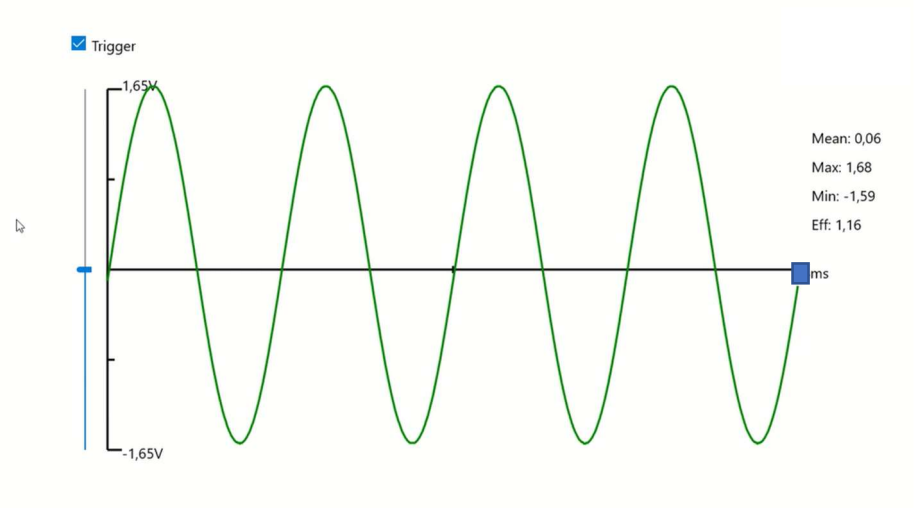
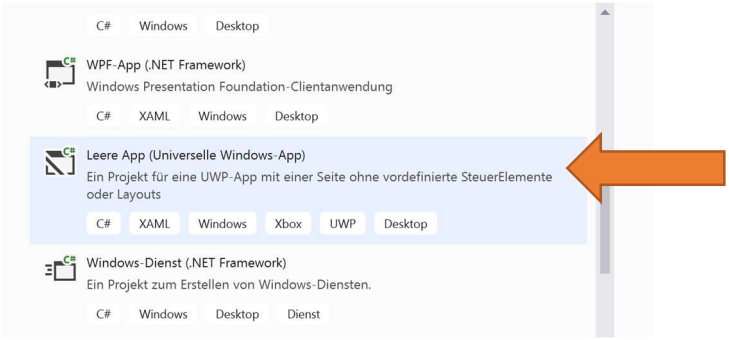


Abbildung 2: Fertiges Oszilloskop-Programm auf dem IoT-PC

Hinweis: Dem hier verwendeten Versuchsaufbau fehlen zu einem echten Oszilloskop noch eine Eingangsfrequenzbeschränkung (Anti-Aliasingtiefpassfilter) und ein variabler Eingangsverstärker. Dennoch wird in dieser Aufgabenstellung der Begriff „Oszilloskop“ zur Vereinfachung verwendet.

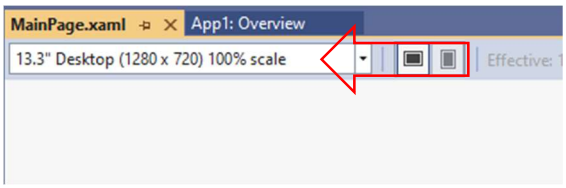
Aufgabe 1: Einstellungen des Projektes und der grafischen Oberfläche

Das Projekt wird auf dem IoT-PC durchgeführt. Erstellen Sie daher zunächst ein Projekt für den ARM Prozessor. An dieser Stelle die wichtigsten Schritte.



- Erstellen Sie ein neues Projekt
- Wählen Sie als Projekttyp eine Visual C# Leere App(Windows Universal) aus.
- Vergeben Sie als Projektnamen Ihren Nachnamen_V4 (besser ohne Sonderzeichen wie Ü).
- Lassen Sie diese Einstellungen für die Zielplattform unverändert und bestätigen Sie nur mit „Ok".
- Wählen Sie im Projektmappen-Explorer die Dateien „MainPage.xaml" an und Sie sehen im „Designer" eine leere „Page".

Im heutigen Versuch wollen wir die Fläche unseres Displays voll ausnutzen. Eine sinnvolle Einstellung für das vorliegende Zielsystem ist 13" Desktop (1280x720) mit einer Skalierung von 100%.

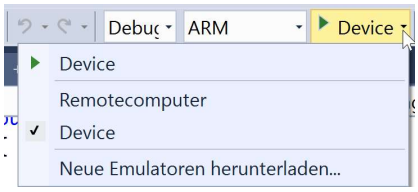


Wie Sie Abbildung 2 entnehmen können, soll der Hintergrund der Applikation hell sein. Bei UWP-Applikationen kann dies sinnvoll erreicht werden, indem das Thema des Hintergrundes angepasst wird. Fügen Sie daher in der Datei App.xaml im Objekt „Application" den Wert „Light" für die Eigenschaft „RequestedTheme" ein. Damit fordert die Applikation bei Ihrem Wirtsystem einen nach Möglichkeit hellen Hintergrund an, ohne dabei die Grundeinstellungen mit einer konkreten Farbe zu überschreiben.

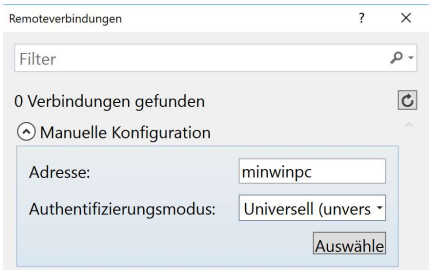
```
<Application
    x:Class="Uebung4.App"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:Uebung4"
    RequestedTheme="Light">
</Application>
```

Fügen Sie einen „Slider", eine „Checkbox", eine „Canvas" und vier Textblöcke in ihre Applikation ein.

Wählen Sie in der Nähe des Startknopfes anstelle von „x86" „ARM" aus und in dem kleinen nach unten zeigenden Dreieck anstelle von „Device" „Remotecomputer".



Es öffnet sich ein Dialog, der sie bittet, das Zielsystem anzugeben. Hier tragen Sie unter „Manuelle Konfiguration“ „minwinpc“ ein. Alternativ können Sie auch die IP-Adresse angeben, die Ihnen auf dem Bildschirm des IoT-PCs angezeigt wird.



Bestätigen Sie mit „Auswählen“ und starten Sie das Programm.

Das Einfügen des Koordinatensystems wird durch eine gemeinschaftliche Programmieraufgabe geschehen. Folgen Sie dazu den Anleitungen des Dozenten.

Aufgabe 2: Ansteuerung der Hardware

Wie in den vorrausgegangenen Aufgaben, sollen Sie die Hardware durch eine eigene Klasse ansteuern. Erstellen Sie daher eine neue Klasse „Oszilloskop“ und implementieren Sie die folgenden Methoden in dieser Klasse.

Der IoT-PC ist mit einem Analog/Digital-Wandler des Typs ADC104S051 von Texas Instruments ausgestattet worden. Die Kommunikation über die SPI-Schnittstelle statt. Abbildung 3 gibt das Timing-Diagramm aus dem Datenblatt wieder.

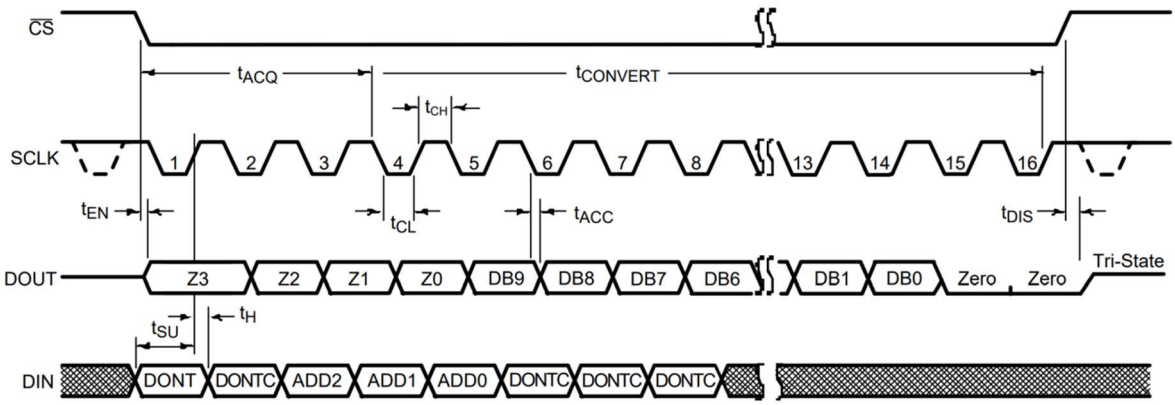


Abbildung 3: ADC10S051 “Serial Timing Diagram”

Beachten Sie die beiden oberen Kurven. Die „Chip-Select“ Leitung ist „Low-Active“, das bedeutet, dass sie während der Kommunikation mit dem AD-Wandler „Low“ sein muss. Zu

Beginn ist das „Clock"-Signal „High" (CPOL=1) und die Kommunikation beginnt dann mit der zweiten Flanke (CPHA=1). Dies stellt den „SPI-Mode 3" dar. Da die Hauptplatine den SPI-Bus mit einer maximalen Frequenz von 4,8 MHz betreiben kann, wählen wir diesen Wert.

Die Initialisierung der SPI-Kommunikation erfordert den Zugriff auf verschiedene statische und nicht statische Methoden verschiedener Klassen, die an dieser Stelle nicht detailliert erläutert werden sollen. Daher wird die Methode zum Initialisieren eines SPI-Kommunikationsobjektes an dieser Stelle vorgegeben (s.u.). Sie benötigen zwei weitere using-Direktiven:

```
using Windows.Devices.Enumeration;
using Windows.Devices.Spi;
```

Der Code für die Methode InitSPI() lautet:

```
private async Task InitSPI()
{
       String spiDeviceSelector = SpiDevice.GetDeviceSelector();
       IReadOnlyList<DeviceInformation> devices = await
       DeviceInformation.FindAllAsync(spiDeviceSelector);
       var Settings = new SpiConnectionSettings(0);
       Settings.ClockFrequency = 4800000;
       Settings.Mode = SpiMode.Mode3;
       ADC = await SpiDevice.FromIdAsync(devices[0].Id, Settings);
}
```

Die Methode erzeugt und initialisiert ein Objekt vom Typ „SpiDevice" mit dem Namen „ADC". Beachten Sie, dass hier auch die Verbindungsgeschwindigkeit auf 4,8 MHz und der SPI-Mode auf 3 festgelegt wird.

In dieser Übung verwenden Sie GPIO, um die Chip-Select-Leitung des AD-Wandlers zur Kommunikation zu wählen.
Die Zuordnung können Sie der untenstehenden Tabelle entnehmen.

GPIO Nummer	*Anschluss*
28	Chip Select LED-Streifen
35	Chip Select ADC
12	Chip Select DAC
25	Reset LED-Streifen
69	Reset DAC

Setzen Sie alle relevanten GPIO-Leitungen auf den Wert „High“ und senken Sie nur Pin 35 für die eigentliche Datenkommunikation auf „Low“, um den AD-Wandler auszuwählen.

Der folgende Code setzt die Richtung der drei GPIO als Ausgang und alle drei auf „HIGH“, da die Chip-Select-Leitungen „low active“ sind.

```
private void initGPIO()
{
  GpioController gpio = GpioController.GetDefault();
  PinChipSelectLED = gpio.OpenPin(28);
  PinChipSelectADC = gpio.OpenPin(35);
  PinChipSelectDAC = gpio.OpenPin(12);

  PinChipSelectLED.SetDriveMode(GpioPinDriveMode.Output);
  PinChipSelectADC.SetDriveMode(GpioPinDriveMode.Output);
  PinChipSelectDAC.SetDriveMode(GpioPinDriveMode.Output);

  PinChipSelectLED.Write(GpioPinValue.High);
  PinChipSelectADC.Write(GpioPinValue.High);
  PinChipSelectDAC.Write(GpioPinValue.High);
}
```

Sehen Sie sich nochmals das Timing Diagramm an. Die Kommunikation mit dem AD-Wandler funktioniert bidirektional, sodass Daten gleichzeitig gesendet und empfangen werden. Praktisch bedeutet das, dass Sie ein Array mit Steuerbytes an den AD-Wandler senden und ein gleichgroßes Array mit gewandelten Werten sofort zurückerhalten. Dabei wird ***jeder Wert*** durch ***zwei Bytes*** repräsentiert. Sie müssen daher ***doppelt so viele Bytes*** schicken und empfangen, wie sie ***Werte*** von AD-Wandler holen möchten. Das gesendete Array besteht dabei abwechselt aus einem Statusbyte und einem, dass nur aus Nullen besteht. Die beiden empfangenen enthalten die 10 Bits des Wertes als „Nutzlast“.

	Byte 0								Byte 1							
	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Gesendet	DONTC	DONTC	ADD2	ADD1	ADD0	DONTC	DONTC	DONTC	DONTC	DONTC	DONTC	DONTC	DONTC	DONTC	DONTC	DONTC
Empfangen	Z3	Z2	Z1	Z0	DB9	DB8	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Zero	Zero

Sie können zwei bis 2*n Werte senden und erhalten 1 bis n Werte zurück. Es ist sinnvoll die DONTC (Don’t care)-Bits mit Nullen zu füllen.

Das Datenblatt gibt Ihnen folgende Informationen über das Statusregister (Abbildung 4).

Table 2. Control Register Bits

Bit 7 (MSB)	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DONTC	**DONTC**	**ADD2**	**ADD1**	**ADD0**	**DONTC**	**DONTC**	**DONTC**

Table 3. Control Register Bit Descriptions

Bit #:	Symbol:	Description
7 - 6, 2 - 0	DONTC	Don't care. The value of these bits do not affect device operation.
5	ADD2	These three bits determine which input channel will be sampled and converted in the next track/hold cycle. The mapping between codes and channels is shown in Table 4.
4	ADD1	
3	ADD0	

Table 4. Input Channel Selection

ADD2	ADD1	ADD0	Input Channel
x	0	0	IN1 (Default)
x	0	1	IN2
x	1	0	IN3
x	1	1	IN4

Abbildung 4: Beschreibung des Kontrollregisters

Wie lautet ein einfaches Kontrollregister-Byte für die Abfrage des ersten Kanals des ADC?

Byte für Kanal 1:

Dieses Byte muss als „jedes erste von zwei“ Bytes in das gesendete Array geschrieben werden. Die Anzahl der geschriebenen entspricht gleichzeitig den empfangenen Werten. Sie erhalten für jedes gesendete Byte eines als Antwort. Aus dieser Antwort müssen Sie die passenden Bits herauslösen, um den aktuellen Spannungswert am AD-Wandler zu bestimmen. Abbildung 5 zeigt einen Ausschnitt aus dem Timing Diagramm aus Abbildung 3:

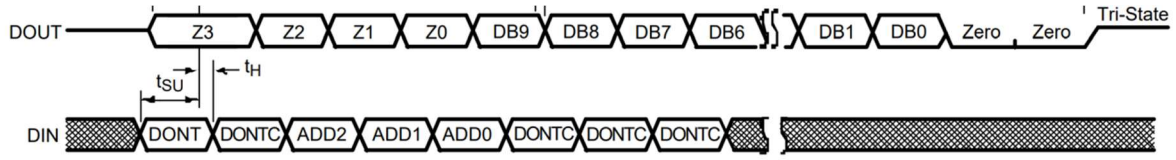


Abbildung 5: Auszug aus dem Timing Diagramm

Die untere Zeitachse zeigt die Bits, die Sie senden (Eingang des AD-Wandlers „DIN“), die obere zeigt den Ausgang des AD-Wandlers. Die Werte Z3 bis Z0 spielen für unsere Betrachtung keine Rolle. DB9—DB0 sind Ihre Nutzdaten, gefolgt von zwei Bit, die immer den Wert Null haben. Ihre erste Aufgabe ist, ein Spannungsmessgerät über den AD-Wandler zu realisieren. Erstellen Sie dazu in der Klasse Oszilloskop einen ***Timer***, der alle 500 ms den Wert vom AD-Wandler holt, und in die Ausgabe schreibt. Verwenden Sie dazu den Befehl:

```
System.Diagnostics.Debug.WriteLine(/*Hier Wert einfügen*/);
```

Dieser Befehl schreibt den Wert in die Standardausgabe des Programms, in Ihrem Fall die Aufgabe des Visual-Studios, die Sie im unteren Teil des Fensters auf dem PC (nicht dem IoT-PC!) finden. Diese Vorgehensweise ist bei der Implementierung eines Programms zu empfehlen, da sie hier ohne grafische Oberfläche Werte schnell ausgeben können. Für das gleichzeitige Senden und Empfangen der Daten nutzen Sie die Methode

```
public void TransferFullDuplex(byte[] writeBuffer, byte[] readBuffer);
```

der Klasse „SPIDevice“. Das Objekt wurde bereits durch InitSPI() erzeugt.

Hinweis 1: Erstellen Sie eine Instanz von „Oszilloskop“ in der „MainPage“ und initialisieren und starten Sie den Timer im Konstruktor von „Oszilloskop“.

*Hinweis 2: Sie können Schiebe-Operatoren (<< und >>) **oder** Masken und Faktoren verwenden, um die Bits für das zu sendende Byte „passend“ zu schieben.*

Hinweis 3: Berechnen Sie den Wert aus den empfangen Werten den AD-Wandler-Wert als „int“ und geben Sie zunächst diesen aus.

Rechnen Sie den Wert des AD-Wandlers zu einer Spannung um. Damit Sie sowohl positive, als auch negative Spannungen messen können, ist der Spannungseingang neben dem ADC noch mit einer Pegelwandlung versehen (Abbildung 6).

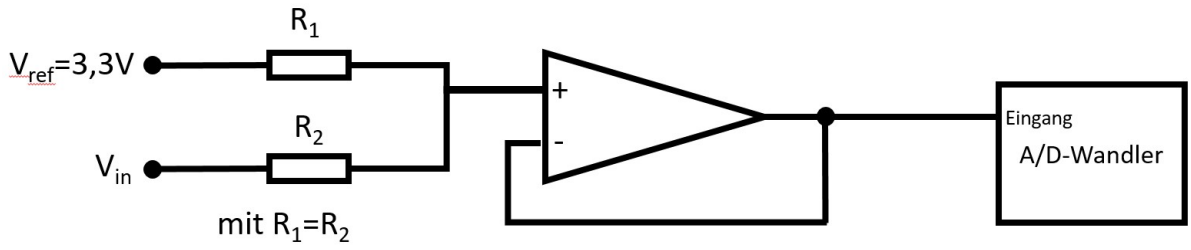


Abbildung 6: Eingangsschaltung des A/D-Wandlers

Sie erhalten daher bei 0V einen „mittleren“ Byte-Wert. Ermitteln Sie die (Integer)-Werte für 0V, +1,65V (halbe Referenzspannung) und -1,65V (negative halbe Referenzspannung).

- 1,65 V	
0 V	
+1,65 V	

Ermitteln Sie eine Formel zur Umrechnung der Integer-Werte in Spannungen.

Rechnen Sie die ermittelten „Integer-Werte“ mit Ihrer Formel in Spannungen um. Benutzen Sie zur Ausgabe wieder die „Debug“-Ausgabe und testen Sie ihr Programm im Wertebereich von 1,8V bis -1,8V.

Aufgabe 3: Grafische Darstellung

Sie sind in der Lage, Werte vom AD-Wandler zu empfangen und korrekt in Spannungen umzurechnen. Beschriften Sie die Spannungsachsen korrekt.
Zur korrekten Beschriftung fehlt noch die Zeitachse. Sie können viele Werte nacheinander aufnehmen, indem sie ein größeres Array abschicken und empfangen. Da der „Canvas“-Bereich 960-Pixel breit ist, nehmen Sie exakt so viele Werte auf.

Wie groß muss das zu sendende/Empfangende Array sein?

Berechnen Sie, welcher Zeit das auf der y-Achse entspricht. Folgende Randdaten müssen Sie beachten:

1. Der SPI-Bus hat eine Frequenz von 4,8 MHz→ 4.800.000 Takte/s
2. Jeder Wert wird durch ***zwei*** Bytes übertragen. Die Hauptplatine unterbricht das Clock-Signal nach jedem Byte für zwei Takte. Daher dauert die Übertragung

eines Bytes 10 Takte und folglich die Übertragung eines Wertes (zwei Bytes) 20 Takte →20Takte/Wert

3. Insgesamt erhalten sie 960 Werte.

Wie vielen Millisekunden (ms) entsprechen diese 960 Werte?

Zum späteren Testen: Bei welcher Eingangsfrequenz passen genau drei Perioden in diese Zeitspanne?

Nehmen Sie bei jedem Timer-Tick 960 Werte auf und stellen Sie diese grafisch dar. Dazu ist es nötig, dass Ihr Oszilloskop-Objekt das Main-Page-Objekt informiert, wenn neue Daten angekommen sind. Die geschieht durch ein Event. Dazu benötigen Sie zunächst einen Delegatentyp, der definiert, wie die Daten übertragen werden sollen. Sie können ihn direkt im Namespace definieren. Da er jedoch zum Oszilloskop gehört, definieren Sie ihn bitte in der Datei „Oszilloskop.cs“.

```
public delegate void neueDatenDelegatenTyp(double[] buffer);
```

Ergänzen Sie das Oszilloskop um ein Event:

```
public event neueDatenDelegatenTyp neueDaten;
```

Abonnieren Sie das Event aus der „MainPage“ heraus und zeichnen Sie die grafische Darstellung in das Koordinatensystem der „Canvas“. Lösen Sie das Event in der Timer-Callback aus, nachdem Sie neue Werte empfangen haben. Ändern Sie das Zeitintervall des Timers auf 100ms. Testen Sie mit einer Sinusfunktion mit Hilfe des Funktionsgenerators. Fügen Sie eine korrekte Achsenbeschriftung ein. Nutzen Sie den halben Wert der Referenzspannung als Obergrenze und die negative halbe Referenzspannung als Minimalwert und skalieren Sie die Kurve entsprechend. Ermitteln Sie die maximale, die minimale und die durchschnittliche Spannung und deren Effektivwert und stellen Sie sie auf dem Display dar. Die Berechnung können Sie innerhalb der Callback in der GUI durchführen.

Zusatzaufgabe 1: Trigger

Das dargestellte Sinussignal „springt“ bei der Darstellung bei jeder Abtastung, da wir immer zu einem beliebigen Zeitpunkt mit der Abtastung beginnen. Oszilloskope haben daher eine „Trigger“-Funktion, bei der Sie eine charakteristische Stelle im Signal zu erkennen versuchen und die Darstellung immer an dieser Stelle beginnen. Ist das Signal kontinuierlich (ohne „Sprünge“), so ist das im einfachsten Fall der Durchgang durch einen bestimmten Spannungswert in eine bestimmte Richtung. Der Spannungswert, der durchschritten wird, wird dabei als „Trigger Level“ bezeichnet und kann entweder von unten nach oben (steigende Flanke) oder von oben nach unten (fallende Flanke) durchschritten werden.

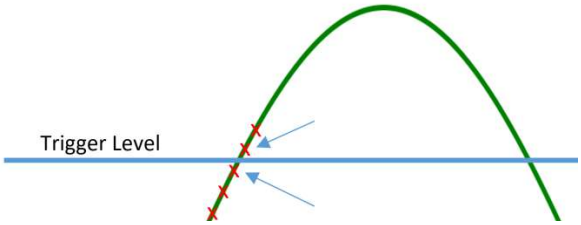


Abbildung 7: Trigger Level

Da wir im Vorfeld einer Abtastung nicht wissen können, wann dieser Durchstoß erfolgen wird, ist es nötig mehr Werte aufzunehmen. Erweitern Sie daher die Signalaufnahme auf die doppelte Anzahl von Werten und bestimmen Sie in einer zusätzlichen Schleife die Position des „Triggers“ innerhalb des Arrays (Abbildung 7). Verwenden Sie den Slider, um das Triggerlevel einzustellen (Maximum, Minimum und Step beachten!) und geben Sie den Wert an das Oszilloskop-Objekt weiter. Dazu müssen sie die Informationen in das Oszilloskop übertragen. Sie benötigen dafür weitere Felder im Oszilloskop-Objekt. Sie können diese „public“ machen und von der „MainPage“ aus ändern.

Eine bessere Lösung erreichen Sie durch die Verwendung von Eigenschaften!