Seriële plotter met wpf

Inhoudsopgave

[Inleiding 1](#_Toc168400903)

[Opzetten project 2](#_Toc168400904)

[Framework & Template 2](#_Toc168400905)

[Refrences 2](#_Toc168400906)

[Verbinden met een poort 2](#_Toc168400907)

[UML 3](#_Toc168400908)

[Schema 3](#_Toc168400909)

[Uitleg 3](#_Toc168400910)

[Main 3](#_Toc168400911)

[Class channel 4](#_Toc168400912)

[Sercom 4](#_Toc168400913)

[De code 4](#_Toc168400914)

[MainWindow.xaml.cs 4](#_Toc168400915)

[Indieping 14](#_Toc168400916)

[MainWindow.xaml 15](#_Toc168400917)

[dataHandler.cs 16](#_Toc168400918)

# Inleiding

In dit project heb ik een Seriële plotter gemaakt met wpf. Deze is gemaakt om degenen van Arduino 2.0 te vervangen. Dees project is echter nog niet af maar voert zijn doel wel juist en goed uit. Ik heb gebruik gemaakt van Microsoft visual studio om dit project te realiseren. Deze Seriële plotter maakt gebruik van een komma om de waardes te seperaten. Van dit project is er ook een github repostorie aangemaakt onder de naam ***ArduinoSerialPlotterImproved*** door de gebruiker LucaRamboer. Oftewel onder deze link <https://github.com/LucaRamboer/ArduinoSerialPlotterImproved> hier kun je de hele code terug vinden en enige vorige versies van het project om het verloop van het project te kunnen raadplegen.

# Opzetten project

## Framework & Template

In dit project heb ik er voor gekozen om de template WPF App (.NET Framework) te gebruiken in visual studio van microsoft. Dees was gekozen zo dat de libarie voor de Seriële poorten die mij gegeven was gebruikt kon worden. Hier is dan ook de code op gebaseerd.

## Refrences

Voor dit project zal je de DDL System.Windorws.Forms moeten toe voegen bij refrences. Wil je echter geen gebruik maken van de file saving mechanic met folder selector dan is deze stap niet nodig.

Voor de poort description gebruiken we een andere refrence namelijk de DDL System.Management zonder deze zal het niet lukken om de discription van de poort in te lezen. Je kan wel echter de gewone poort naam verkrijge met de standaart IO.Ports.

## Verbinden met een poort

Als de drivers van de arduino of van enige seriele poort waar je mee zou willen verbinden niet geïnstalleerd zouden zijn, dan zal het programma de poort niet herkennen en zal je niks kunnen doen met de software. Anders kan je de drop down list gebruiken om je poort te kiezen en dan zo verbinden.

# UML

## Schema

Afbeelding met tekst, schermopname, diagram, Parallel

Automatisch gegenereerde beschrijving

## Uitleg

### Main

De eerste 2 events **stateChanged** en **sizeChanged** worden opgeroepen wanneer het formaat wijzig van de ui of het geminimaliseerd wordt. Dit wordt geregeld door de knoppen hier onder.

Deze zullen dan de objecten aan roepen en de nodige info mee geven.

**connectBtton.click** zorgt er voor dat er een verbinding geopent wordt tussen het programma en de seriele poort. Deze kan ook de verbinding sluiten als deze al open zou zijn.

**refreshButton.Click** zorgt er voor dat het programma opnieuw kijkt welke poorten er open zijn en geeft deze door aan de gebruiker.

**Metingenbox.TextChanged** veranderd de grote van de array waar de metingen in worden opgeslagen door in de objeccten sizzing aan te roepen en geeft de nieuwe waarde door.

Zowel **Ymin.TextChanged** als **Ymax.TextChanged** veranderen intern een variable van waarde door de waarde dat getyped werd in de vakken.

**scaleOverride.Checked** past een paar variable aan van waarde en roept de hoofd functie opnieuw op.

**scaleOverride.Unchecked** roept ook de hoofd functie terug op en ook een andere functie dat de uiterste van alle kanalen gaat zoeken.

**ySeg.TextChanged** past een variable aan en roept de hoofdfunctie terug op.

**SenderBox.KeyDown** kijkt na welke knop er ingedrukt word als deze de enter knop is stuurt hij de data in de senderbox door de seriele communicatie en leegt deze box.

**SaveToFile.Checked** en **SaveToFile.Unchecked** zetten een de zelfde variable respectievelijk op true en op false

**pathSelector.Click** dit event open de folder selector dialog.

### Class channel

De funcite **SetOrigin** past een variable aan binnen het object.

De functie **Sizing** wordt aangeroepen wanneer er een verandering in het aantal metingen gebeurd en de array zich daar op moet aanpassen. Deze functie past daar door ook de grote van de array aan.

De functie **Line** schuift alle entries 1 plaats op en voegt de nieuwe waarde toe aan de array.

De functie **drawingChannel** tekent de array op onze chart.

De functie **remove** verwijdert de array van de chart en leegt het object.

### Sercom

Deze zit verwerkt in onze main file. Dit is de buffer tussen ons programma en de arduino. De arduino zal ook de snelheid van het signaal bepalen.

Wanneer er dan data binnenkomt van de arduino zal het event **SerPort.DataRecieved** oproepen wat op zijn beurt dan de data proccesed en dan de juiste functies oproept.

# De code

## MainWindow.xaml.cs

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**IO**;**

**using** System**.**IO**.**Ports**;**

**using** System**.**Linq**;**

**using** System**.**Reflection**;**

**using** System**.**Runtime**.**Remoting**.**Channels**;**

**using** System**.**Text**;**

**using** System**.**Threading**.**Tasks**;**

**using** System**.**Threading**;**

**using** System**.**Windows**;**

**using** System**.**Windows**.**Controls**;**

**using** System**.**Windows**.**Data**;**

**using** System**.**Windows**.**Documents**;**

**using** System**.**Windows**.**Input**;**

**using** System**.**Windows**.**Media**;**

**using** System**.**Windows**.**Media**.**Imaging**;**

**using** System**.**Windows**.**Navigation**;**

**using** System**.**Windows**.**Shapes**;**

**using** System**.**Text**.**RegularExpressions**;**

**using** System**.**Linq**.**Expressions**;**

**using** System**.**Windows**.**Media**.**Animation**;**

**using** static System**.**Net**.**Mime**.**MediaTypeNames**;**

**using** System**.**Management**;**

**namespace** WPFCharting

**{**

/// <summary>

/// Interaction logic for MainWindow.xaml

/// </summary>

**public** **partial** class MainWindow **:** Window

**{**

int i**;**

**private** bool scaleChanging**,** savetofile**,** connected**,** interval0**;**

**public** static Line xAxisLine**,** yAxisLine**,** line**;**

**public** static double xAxisStart **=** 150**,** yAxisStart **=** 250**,** yAxisStop **=** 50**;**

**public** static double xinterval **{** **get;** **set;** **}** **=** 50**;**

**public** static double yinterval **{** **get;** **set;** **}** **=** 50**;**

**public** double yPointInterval**;**

**public** int ysegments **=** 10**,** ysegmentsOld **=** 10**;**

**public** static double ystart **=** 0**,** ystop **=** 50**;**

**public** double yboxMin **=** 0**,** yboxMax **=** 50**;**

double yPoint**,** xPoint**,** yValue**,** MaxIndex**,** MinIndex**;**

**public** static double yscale**;**

**private** double tempD**;**

**private** int count**;**

**private** int colorCount**;**

**private** int textOutLines**,** metingen **=** 50**;**

**private** string ReceivedData**;**

**public** string data**;**

**private** string tempS**;**

**private** SerialPort SerPort**;**

**readonly** Regex regexN **=** **new** Regex**(**"^-{0,1}[0-9]{0,}$"**);**

**readonly** Regex regex **=** **new** Regex**(**"[^0-9]+"**);**

**readonly** Regex regexIn **=** **new** Regex**(**"[0-9.]+"**);**

TextBlock yTextBlock0**,** textBlock**;**

TextBox box**;**

static **public** Point origin**;**

string pathFile **=** System**.**IO**.**Path**.**Combine**(**Environment**.**GetFolderPath**(**Environment**.**SpecialFolder**.**MyDocuments**),** "PlotOutput.txt"**);**

**public** static string**[]** split **=** **new** string**[]** **{** **};**

string**[]** output **=** **new** string**[**1**];**

**private** List**<**channel**>** channels **=** **new** List**<**channel**>();**

**private** SolidColorBrush**[]** colors **=** **new** SolidColorBrush**[]** **{**

Brushes**.**Blue**,**

Brushes**.**Brown**,**

Brushes**.**Red**,**

Brushes**.**Green**,**

Brushes**.**Magenta**,**

Brushes**.**Yellow**,**

Brushes**.**DarkGreen

**};**

System**.**Windows**.**Forms**.**FolderBrowserDialog openFileDlg **=** **new** System**.**Windows**.**Forms**.**FolderBrowserDialog**();** //defines folder dialog

System**.**Windows**.**Threading**.**DispatcherTimer dispatcherTimer **=** **new** System**.**Windows**.**Threading**.**DispatcherTimer**();** //defines interval timer

**public** MainWindow**()**

**{**

InitializeComponent**();**

FetchAvailablePorts**();**

//regelt alle events die moeten geregeld worden

**this.**StateChanged **+=** **(**sender**,** e**)** **=>**

**{**

run**();**

**};**

**this.**SizeChanged **+=** **(**sender**,** e**)** **=>**

**{**

run**();**

**};**

connectButton**.**Click **+=** **(**sender**,** e**)** **=>**

**{**

**if** **(**connected**)** disconnect**();** **else** connect**();**

**};**

refreshButton**.**Click **+=** **(**sender**,** e**)** **=>**

**{**

FetchAvailablePorts**();**

**};**

metingenBox**.**TextChanged **+=** **(**sender**,** e**)** **=>**

**{**

**try**

**{**

metingen **=** Int32**.**Parse**(**metingenBox**.**Text**);**

**foreach** **(**var channel **in** channels**)**

**{**

channel**.**Sizing**(**metingen**);**

**}**

run**();**

**}**

**catch** **{** **}**

**};**

YMax**.**TextChanged **+=** **(**sender**,** e**)** **=>**

**{**

**try**

**{**

yboxMax **=** Double**.**Parse**(**YMax**.**Text**);**

**if** **(**scaleOverride**.**IsChecked **==** **true)**

**{**

ystop **=** yboxMax**;**

run**();**

**}**

**}**

**catch** **{** **}**

**};**

YMin**.**TextChanged **+=** **(**sender**,** e**)** **=>**

**{**

**try**

**{**

yboxMin **=** Double**.**Parse**(**YMin**.**Text**);**

**if** **(**scaleOverride**.**IsChecked **==** **true)**

**{**

ystart **=** yboxMin**;**

run**();**

**}**

**}**

**catch** **{** **}**

**};**

scaleOverride**.**Checked **+=** **(**sender**,** e**)** **=>**

**{**

ystop **=** yboxMax**;**

ystart **=** yboxMin**;**

run**();**

**};**

scaleOverride**.**Unchecked **+=** **(**sender**,** e**)** **=>** **{**

findLimits**();**

run**();**

**};**

ySeg**.**TextChanged **+=** **(**sender**,** e**)** **=>**

**{**

**try** **{**

ysegments **=** Int32**.**Parse**(**ySeg**.**Text**);**

run**();**

**}** **catch** **{** **}**

**};**

senderBox**.**KeyDown **+=** **(**sender**,** e**)** **=>**

**{**

**if** **(**e**.**Key **==** Key**.**Enter**)** **{**

sendData**();**

senderBox**.**Clear**();**

**}**

**};**

SaveToFile**.**Checked **+=** **(**sender**,** e**)** **=>** **{**

savetofile **=** **true;**

**};**

SaveToFile**.**Unchecked **+=** **(**sender**,** e**)** **=>** **{**

savetofile **=** **false;**

**};**

pathSelector**.**Click **+=** **(**sender**,** e**)** **=>**

**{**

selectPath**();**

**};**

//puts the right settings on the iterval timer

dispatcherTimer**.**Tick **+=** **new** EventHandler**(**dispatcherTimer\_Tick**);**

dispatcherTimer**.**Interval **=** **new** TimeSpan**(**0**,** 0**,** 1**);**

dispatcherTimer**.**Start**();**

**}**

static ManagementObject**[]** FindPorts**(){**

**try**

**{**

ManagementObjectSearcher searcher **=** **new** ManagementObjectSearcher**(**"root\\CIMV2"**,** "SELECT \* FROM WIN32\_PnPEntity"**);**

List**<**ManagementObject**>** objects **=** **new** List**<**ManagementObject**>();**

**foreach** **(**ManagementObject obj **in** searcher**.**Get**())**

**{**

objects**.**Add**(**obj**);**

**}**

**return** objects**.**ToArray**();**

**}**

**catch** **(**Exception ex**)**

**{**

MessageBox**.**Show**(**ex**.**Message**,** "Error...!"**);**

**return** **new** ManagementObject**[]** **{** **};**

**}**

**}**

static string**[]** FindAllPorts**()**

**{**

List**<**string**>** ports **=** **new** List**<**string**>();**

**foreach** **(**ManagementObject obj **in** FindPorts**())**

**{**

**try**

**{**

**if** **(**obj**[**"Caption"**]** **!=** **null)**

**{**

**if** **(**obj**[**"Caption"**].**ToString**().**Contains**(**"(COM"**))**

**{**

string ComName **=** ParseCOMName**(**obj**);**

**if** **(**ComName **!=** **null)**

ports**.**Add**(**ComName **+** "\n" **+** obj**[**"Description"**].**ToString**());**

**}**

**}**

**}**

**catch** **(**Exception ex**)**

**{**

MessageBox**.**Show**(**ex**.**Message**,** "Error...!"**);**

**}**

**}**

**return** ports**.**ToArray**();**

**}**

static string ParseCOMName**(**ManagementObject obj**)**

**{**

string name **=** obj**[**"Name"**].**ToString**();**

int startIndex **=** name**.**LastIndexOf**(**"("**);**

int endIndex **=** name**.**LastIndexOf**(**")"**);**

**if(**startIndex **!=** **-**1 **&&** endIndex **!=** 1**)**

**{**

name **=** name**.**Substring**(**startIndex **+** 1**,** endIndex **-** startIndex **-** 1**);**

**return** name**;**

**}**

**return** **null;**

**}**

void FetchAvailablePorts**()**

**{**

Portsbox**.**Items**.**Clear**();**

**foreach** **(**var port **in** FindAllPorts**())**

**{**

Portsbox**.**Items**.**Add**(**port**);**

**}**

**}**

void connect**()**

**{**

//hardcoding some parameters, check MSDN for more

**try** //in het geval dat het verbinden mislukt voor welke reden dan ook zal dit er voor zorgen dat de code niet crashed

**{**

SerPort **=** **new** SerialPort**();** //instantiate our serial port SerPort

SerPort**.**BaudRate **=** 9600**;**

SerPort**.**PortName **=** Portsbox**.**Text**.**Substring**(**0**,** Portsbox**.**Text**.**IndexOf**(**"\n"**));**

SerPort**.**Parity **=** Parity**.**None**;**

SerPort**.**DataBits **=** 8**;**

SerPort**.**StopBits **=** StopBits**.**One**;**

SerPort**.**ReadBufferSize **=** 200000000**;**

SerPort**.**DataReceived **+=** SerPort\_DataReceived**;**

SerPort**.**Open**();**

connected **=** **true;**

connectButton**.**Content **=** "disconnect"**;**

refreshButton**.**IsEnabled **=** **false;**

Portsbox**.**IsEnabled **=** **false;**

**}**

**catch** **(**Exception ex**)**

**{**

MessageBox**.**Show**(**ex**.**Message**,** "Error...!"**);**

**if(**ex**.**Message**.**EndsWith**(**"does not exist."**))** FetchAvailablePorts**();** // als de gebruiker probeert te verbinden met een poort dat niet bestaad zal de poort lijst hetnieuwen

**}**

**}**

void disconnect**()** **{**

**try**

**{**

SerPort**.**Close**();** //sluit de verbinding met de poort

**}**

**catch** **(**Exception ex**)**

**{**

MessageBox**.**Show**(**ex**.**Message**,** "Error...!"**);**

**}**

connected **=** **false;**

connectButton**.**Content **=** "connect"**;**

refreshButton**.**IsEnabled **=** **true;**

Portsbox**.**IsEnabled **=** **true;**

**}**

**private** void NumberValidationTextBox**(object** sender**,** TextCompositionEventArgs e**)** //Zorgt er voor dat enkel nummers in de textbox gezet kunnen worden

**{**

e**.**Handled **=** regex**.**IsMatch**(**e**.**Text**);**

**}**

**private** void NumberValidationTextBoxNegatif**(object** sender**,** TextCompositionEventArgs e**)** //Zorgt er voor dat enkel nummers in de textbox gezet kunnen worden maar ook negatieve

**{**

box **=** **(**TextBox**)**sender**;**

e**.**Handled **=** **!**regexN**.**IsMatch**(**box**.**Text**.**Insert**(**box**.**CaretIndex**,** e**.**Text**));**

**}**

**private** void SerPort\_DataReceived**(object** sender**,** SerialDataReceivedEventArgs e**)**

**{**

**try**

**{**

ReceivedData **=** SerPort**.**ReadLine**();** //read the line from the serial port

**}** **catch** **{** **};**

Dispatcher**.**Invoke**(new** Action**(**ProcessingData**));** //at each received line from serial port we "trigger" a new processingdata delegate

**}**

**private** void run**()**

**{**

xinterval **=** **(this.**ActualWidth **-** xAxisStart **-** 70**)** **/** **(**metingen **-** 1**);** //bepaalt de juiste hoeveelheid pixels per meting

**if** **(**xinterval **<** 0**)** xinterval **=** 0**;**

**foreach** **(**var channel **in** channels**)** // roept voor elk bestaand kannaal de volgende functies op.

**{**

channel**.**setOrigin**(this.**ActualHeight**);**

**}**

chartCanvas**.**Children**.**Clear**();**

yAxisStart **=** 250**;**

**if(this.**ActualHeight **<** 550**)**

**{**

**if** **(this.**ActualHeight **<** 300**)**

yAxisStart **=** 0**;**

**else**

**{**

yAxisStart **=** 250 **-** **(**550 **-** **this.**ActualHeight**);**

**}**

**}**

xAxisLine **=** **new** Line**()**

**{**

X1 **=** xAxisStart**,**

Y1 **=** **this.**ActualHeight **-** yAxisStart**,**

X2 **=** **this.**ActualWidth **-** 70**,**

Y2 **=** **this.**ActualHeight **-** yAxisStart**,**

Stroke **=** Brushes**.**LightGray**,**

StrokeThickness **=** 1**,**

**};**

yAxisLine **=** **new** Line**()**

**{**

X1 **=** xAxisStart**,**

Y1 **=** yAxisStop**,**

X2 **=** xAxisStart**,**

Y2 **=** **this.**ActualHeight **-** yAxisStart**,**

Stroke **=** Brushes**.**LightGray**,**

StrokeThickness **=** 1**,**

**};**

**if** **(**xAxisLine**.**X2 **<** xAxisLine**.**X1**)** xAxisLine**.**X2 **=** xAxisLine**.**X1**;**

**if** **(**yAxisLine**.**Y2 **<** yAxisLine**.**Y1**)** yAxisLine**.**Y2 **=** yAxisLine**.**Y1**;**

chartCanvas**.**Children**.**Add**(**xAxisLine**);**

chartCanvas**.**Children**.**Add**(**yAxisLine**);**

origin **=** **new** Point**(**xAxisLine**.**X1**,** yAxisLine**.**Y2**);**

xrun**();**

yrun**();**

**foreach** **(**var ch **in** channels**)** ch**.**drawingchannel**();**

**}**

**private** void xrun**()**

**{**

// x axis lines

xPoint **=** origin**.**X **+** xinterval**;**

**for(**i **=** 0**;** i **<** metingen **-** 1**;** i**++)**

**{**

line **=** **new** Line**()**

**{**

X1 **=** xPoint**,**

Y1 **=** yAxisLine**.**Y1**,**

X2 **=** xPoint**,**

Y2 **=** yAxisLine**.**Y2**,**

Stroke **=** Brushes**.**LightGray**,**

StrokeThickness **=** 1**,**

Opacity **=** 1**,**

**};**

**if** **(**line**.**Y2 **<** line**.**Y1**)** **{** line **=** **null;** **break;** **}**

chartCanvas**.**Children**.**Add**(**line**);**

xPoint **+=** xinterval**;**

**}**

**}**

// y axis lines

**private** void yrun**()** **{**

yPointInterval **=** **(**yAxisLine**.**Y2 **-** yAxisLine**.**Y1 **-** 1**)** **/** ysegments**;**

**if** **(**yPointInterval **<** 1**)** yPointInterval **=** 1**;**

yinterval **=** **(**ystop **-** ystart**)** **/** ysegments**;**

**if** **(**yinterval **==** 0**)** **{**

ysegmentsOld **=** ysegments**;**

ysegments **=** 0**;**

ySeg**.**Text **=** "0"**;**

interval0 **=** **true;**

**}** **else** **if** **(**interval0**)** **{**

ysegments **=** ysegmentsOld**;**

ySeg**.**Text **=** $"{ysegments}"**;**

interval0 **=** **false;**

**}**

yscale **=** **(**yAxisLine**.**Y2 **-** yAxisLine**.**Y1**)** **/** **(**ystop **-** ystart**);**

yValue **=** ystart **+** yinterval**;**

yPoint **=** origin**.**Y **-** yPointInterval**;**

yTextBlock0 **=** **new** TextBlock**()** **{** Text **=** $"{ystart}" **};**

chartCanvas**.**Children**.**Add**(**yTextBlock0**);**

Canvas**.**SetLeft**(**yTextBlock0**,** origin**.**X **-** 30**);**

Canvas**.**SetTop**(**yTextBlock0**,** origin**.**Y **-** 10**);**

**while** **(**yPoint **>** yAxisLine**.**Y1**)**

**{**

line **=** **new** Line**()**

**{**

X1 **=** xAxisStart**,**

Y1 **=** yPoint**,**

X2 **=** **this.**ActualWidth **-** 70**,**

Y2 **=** yPoint**,**

Stroke **=** Brushes**.**LightGray**,**

StrokeThickness **=** 1**,**

Opacity **=** 1**,**

**};**

**if** **(**line**.**X2 **<** line**.**X1**)** **{** line **=** **null;** **break;** **}**

chartCanvas**.**Children**.**Add**(**line**);**

tempS **=** **((**ystop **-** ystart**)** **/** ysegments**).**ToString**();**

**if** **(**tempS**.**IndexOf**(**','**)** **<** 0**)** textBlock **=** **new** TextBlock**(){** Text **=** $"{yValue}" **};**

**else** textBlock **=** **new** TextBlock**()** **{** Text **=** $"{Math.Round(yValue, tempS.Substring(tempS.IndexOf(',')).Length)}" **};**

chartCanvas**.**Children**.**Add**(**textBlock**);**

Canvas**.**SetLeft**(**textBlock**,** line**.**X1 **-** 30**);**

Canvas**.**SetTop**(**textBlock**,** yPoint **-** 10**);**

yPoint **-=** yPointInterval**;**

yValue **+=** yinterval**;**

**}**

**}**

**private** void ProcessingData**()** //processing received data and plotting it on a chart

**{**

**try**

**{**

split **=** ReceivedData**.**Trim**().**Split**(**','**);** //split the data separated by tabs; split[3] -> split[col1], split[col2], split[col3]; split[i]

count **=** split**.**Length**;**

data **=** "\n"**;**

**if** **(**regexIn**.**IsMatch**(**split**[**0**]))**

**{**

**for(**i **=** 0**;** i **<** count**;** i**++)** **{** split**[**i**]** **=** split**[**i**].**Replace**(**'.'**,** ','**);** **}**

**while** **(**count **!=** channels**.**Count**)**

**{**

**if** **(**count **<** channels**.**Count**)**//kijkt na of er kanalen moeten verwijderd of aangemaakt moeten worden

**{**

channels**[**channels**.**Count **-** 1**].remove();**

channels**.**RemoveAt**(**channels**.**Count **-** 1**);**

**if** **(**scaleOverride**.**IsChecked **==** **true)** findLimits**();**

**}**

**else** channels**.**Add**(new** channel**(**channels**.**Count **+** 1**,** colors**[**colorCount**++],** **this.**ActualHeight**,** chartCanvas**,** LimitBlock**));**

**if** **(**colorCount **==** colors**.**Length**)** colorCount **=** 0**;**

**}**

**foreach** **(**var ch **in** channels**)**

**{**

ch**.**Line**();**

ch**.**drawingchannel**();**

**}**

**for** **(**i **=** 0**;** i **<** count**;** i**++)**

**{**

data **+=** $"{split[i]}, "**;**

**if** **(**scaleOverride**.**IsChecked **==** **false)**

**{**

tempD **=** Double**.**Parse**(**split**[**i**]);**

**if** **(**tempD **>=** ystop**)**

**{**

MaxIndex **=** metingen**;**

scaleChanging **=** **true;**

**if** **(**tempD **!=** ystop**)**

**{**

ystop **=** tempD**;**

**}**

**}**

**else** **if** **(**tempD **<=** ystart**)**

**{**

MinIndex **=** metingen**;**

scaleChanging **=** **true;**

**if** **(**tempD **!=** ystart**)**

**{**

ystart **=** tempD**;**

**}**

**}**

**}**

**}**

**if** **(**scaleOverride**.**IsChecked **==** **false)**//kijkt of smart scaling aan staat

**{**

MaxIndex**--;**

MinIndex**--;**

//als de index van de limiet onder nul zit oftewel de limiet is gepaseerd zal er opnieuw gezocht worden naar de limiet

**if** **(**MaxIndex **<** 0**)**

**{**

findLimits**(**findmin**:** **false);**

**}**

**if** **(**MinIndex **<** 0**)**

**{**

findLimits**(**findmax**:** **false);**

**}**

**}**

//zorgt er voor dat de data de de gebruiker te zien krijgt terug mooi een punt krijgt als kommagetal en een komma als seperator en dan de laatste komma weg haald

tempS **=** "\n"**;**

**for** **(**i **=** 0**;** i **<** count**;** i**++)** **{** tempS **+=** $"{split[i].Replace(',', '.')}, "**;** **}**

tempS **=** tempS**.**Remove**(**tempS**.**Length **-** 2**);**

TextOut**.**Text **+=** tempS**;**

**if** **(**savetofile**)** //schrijft de nieuwe data naar de file als dit aan staat

**{**

**if** **(**File**.**Exists**(**pathFile**))** **{**

output**[**0**]** **=** tempS**;**

File**.**AppendAllLines**(**pathFile**,** output**);**

**}** **else** **{**

File**.**WriteAllText**(**pathFile**,** tempS**);**

**}**

**}**

textOutLines**++;**

**while** **(**textOutLines **>=** metingen**)**//kijkt of de hoeveelheid metingen in de textbox niet meer is dan de hoeveelheid metingen

**{**

TextOut**.**Text **=** TextOut**.**Text**.**Remove**(**0**,** TextOut**.**Text**.**IndexOf**(**"\n"**)** **+** 1**);**

textOutLines**--;**

**}**

**if** **((**scaleOverride**.**IsChecked **==** **false)** **&&** scaleChanging**)**//past de chart en polylines aan op de nieuwe waardes

**{**

scaleChanging **=** **false;**

run**();**

**}**

**}** **else**

**{**

TextOut**.**Text **+=** "\n" **+** ReceivedData**.**Remove**(**ReceivedData**.**Length **-** 1**);**

**}**

**if** **(**autoscroll**.**IsChecked **==** **true)** textScrol**.**ScrollToBottom**();**

**}**

**catch** **{** **}**

**}**

**private** void findLimits**(**bool findmax **=** **true,** bool findmin **=** **true)** **{** //zoekt de limieten waar er om gevraagt wordt van de huidige signale en geeft commando om de scale aan te passen

**if** **(**channels**.**Count **!=** 0**)**

**{**

**if** **(**findmax**)**

**{**

ystop **=** channels**[**0**].**Serie**[**0**];**

**foreach** **(**var ch **in** channels**)**

**{**

**for** **(**i **=** 0**;** i **<** metingen**;** i**++)**

**{**

**if** **(**ch**.**Serie**[**i**]** **>** ystop**)** **{** ystop **=** ch**.**Serie**[**i**];** MaxIndex **=** i**;** **}**

**}**

**}**

**}**

**if** **(**findmin**)**

**{**

ystart **=** channels**[**0**].**Serie**[**0**];**

**foreach** **(**var ch **in** channels**)**

**{**

**for** **(**i **=** 0**;** i **<** metingen**;** i**++)**

**{**

**if** **(**ch**.**Serie**[**i**]** **<** ystart**)** **{** ystart **=** ch**.**Serie**[**i**];** MinIndex **=** i**;** **}**

**}**

**}**

**}**

scaleChanging **=** **true;**

**}** **else**

**{**

ystop **=** yboxMax**;**

ystart **=** yboxMin**;**

**}**

**}**

**private** void sendData**()** **{** //stuurt data uit op de Seriële comunicate als er een verbinding is

**if** **(**connected**)**

**{**

**try**

**{**

SerPort**.**WriteLine**(**senderBox**.**Text**);**

**}**

**catch** **(**Exception ex**)**

**{**

**if** **(!**SerPort**.**IsOpen**)** losconection**();**

**else** MessageBox**.**Show**(**ex**.**Message**,** "Error...!"**);**

**}**

**}**

**else** **{**

MessageBox**.**Show**(**"Not connected to a port"**,** "Connection"**);**

**}**

**}**

void selectPath**()**

**{**

//opent een folder selector dialog en gebruikt de output om een folder te selecteren voor de file saving mechanic

openFileDlg **=** **new** System**.**Windows**.**Forms**.**FolderBrowserDialog**();**

var result **=** openFileDlg**.**ShowDialog**();**

**if** **(**result**.**ToString**()** **!=** string**.**Empty**)**

**{**

pathFile **=** System**.**IO**.**Path**.**Combine**(**openFileDlg**.**SelectedPath**,** "PlotOutput.txt"**);**

**}**

**}**

**private** void dispatcherTimer\_Tick**(object** sender**,** EventArgs e**)**

**{**

**if(**connected **&&** **!**SerPort**.**IsOpen**)**//checks every tick if the connection is lost

**{**

losconection**();**

**}**

**}**

//notifys the user and sets the right vars to the right values for the los of connection

void losconection**()**

**{**

connected **=** **false;**

connectButton**.**Content **=** "connect"**;**

refreshButton**.**IsEnabled **=** **true;**

Portsbox**.**IsEnabled **=** **true;**

FetchAvailablePorts**();**

MessageBox**.**Show**(**"Connection to port lost!"**,** "Connection"**);**

**}**

**}**

**}**

### Indieping

void selectPath**()**

**{**

openFileDlg **=** **new** System**.**Windows**.**Forms**.**FolderBrowserDialog**();**

var result **=** openFileDlg**.**ShowDialog**();**

**if** **(**result**.**ToString**()** **!=** string**.**Empty**)**

**{**

pathFile **=** System**.**IO**.**Path**.**Combine**(**openFileDlg**.**SelectedPath**,** "PlotOutput.txt"**);**

**}**

**}**

In dit stukje code maken we gebruik van de extra refrence. Het dialoog venster is niet het beste naar mijn mening natuurlijk, maar door onze keuze in .NET framework zijn we wat gelimiteerd en hebben we echter niet veel keuze hier in. Je zou deze optie ook kunnen weglaten en dan een standaard locatie gebruiken voor de output file.

In de code maken we gebruik van een list om al onze kanalen gebaseerd op de class channel in op te slagen. Door deze list zullen we dan de mogelijkheid hebben om zo veel als het geheugen toe staat aan kanalen te hebben en deze dan allemaal aan te roepen zonder deze handmatig te moeten definiëren.

## MainWindow.xaml

<Window x:Class=**"WPFCharting.MainWindow"**

xmlns=**"http://schemas.microsoft.com/winfx/2006/xaml/presentation"**

xmlns:x=**"http://schemas.microsoft.com/winfx/2006/xaml"**

xmlns:d=**"http://schemas.microsoft.com/expression/blend/2008"**

xmlns:mc=**"http://schemas.openxmlformats.org/markup-compatibility/2006"**

xmlns:local=**"clr-namespace:WPFCharting"**

mc:Ignorable=**"d"**

Title=**"MainWindow"** Height=**"550"** Width=**"800"**>

<Grid Name=**"GridC"**>

<Grid.ColumnDefinitions>

<ColumnDefinition Width=**"35"**/>

<ColumnDefinition Width=**"42"**/>

<ColumnDefinition Width=**"40"**/>

<ColumnDefinition Width=**"\*"**/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height=**"25"**/>

<RowDefinition Height=**"30"**/>

<RowDefinition Height=**"30"**/>

<RowDefinition Height=**"30"**/>

<RowDefinition Height=**"26"**/>

<RowDefinition Height=**"25"**/>

<RowDefinition Height=**"26"**/>

<RowDefinition Height=**"26"**/>

<RowDefinition Height=**"26"**/>

<RowDefinition Height=**"20"**/>

<RowDefinition Height=**"26"**/>

<RowDefinition Height=**"23"**/>

<RowDefinition Height=**"\*"**/>

</Grid.RowDefinitions>

<Canvas x:Name=**"chartCanvas"** Background=**"White"**/>

<Label Content=**"Serial port:"** Grid.Row=**"0"** Grid.Column=**"0"** Grid.ColumnSpan=**"2"**/>

<ComboBox Name=**"Portsbox"** Width=**"100"** Height=**"20"** Grid.Row=**"1"** Grid.ColumnSpan=**"3"**/>

<Button Name=**"connectButton"** Content=**"connect"** Width=**"100"** Height=**"20"** Grid.Row=**"2"** Grid.ColumnSpan=**"3"**/>

<Button Name=**"refreshButton"** Content=**"refresh"** Width=**"100"** Height=**"20"** Grid.Row=**"3"** Grid.ColumnSpan=**"3"**/>

<Label Content=**"Metingen:"** Grid.Row=**"4"** Grid.ColumnSpan=**"3"**/>

<TextBox Name=**"metingenBox"** Height=**"20"** Grid.Row=**"5"** Text=**"50"** Width=**"100"** MaxLength=**"4"** PreviewTextInput=**"NumberValidationTextBox"** Grid.ColumnSpan=**"3"** HorizontalAlignment=**"Left"** Margin=**"5,0,0,0"**/>

<Label Content=**"y-axis scale:"** Grid.Row=**"6"** Grid.ColumnSpan=**"3"** Height=**"26"**/>

<Label Content=**"Max:"** Grid.Row=**"7"** Grid.Column=**"0"**/>

<TextBox Name=**"YMax"** Height=**"20"** Grid.Row=**"7"** Grid.Column=**"1"** Grid.ColumnSpan=**"2"** Text=**"50"** Width=**"50"** PreviewTextInput=**"NumberValidationTextBoxNegatif"**/>

<Label Content=**"Min:"** Grid.Row=**"8"** Grid.Column=**"0"**/>

<TextBox Name=**"YMin"** Height=**"20"** Grid.Row=**"8"** Grid.Column=**"1"** Grid.ColumnSpan=**"2"** Text=**"0"** Width=**"50"** PreviewTextInput=**"NumberValidationTextBoxNegatif"**/>

<CheckBox Name=**"scaleOverride"** Content=**"scale override"** Grid.Row=**"9"** Grid.ColumnSpan=**"3"** IsChecked=**"False"**/>

<Label Content=**"segments:"** Grid.Row=**"10"** Grid.Column=**"0"** Grid.ColumnSpan=**"2"**/>

<TextBox Name=**"ySeg"** Height=**"20"** Grid.Row=**"10"** Grid.Column=**"2"** Text=**"10"** Width=**"30"** MaxLength=**"3"** PreviewTextInput=**"NumberValidationTextBox"**/>

<Label Name=**"Limit"** Grid.Row=**"11"** Grid.RowSpan=**"2"** Grid.Column=**"0"** Grid.ColumnSpan=**"3"** MaxWidth=**"117"** HorizontalAlignment=**"Left"**>

<TextBlock Name=**"LimitBlock"** TextWrapping=**"WrapWithOverflow"** Text=**"Channels of limit: "**/>

</Label>

<CheckBox Name=**"autoscroll"** Content=**"auto scroll"** Grid.Row=**"13"** Grid.ColumnSpan=**"4"** VerticalAlignment=**"Bottom"** Margin=**"0, 0, 0, 170"** IsChecked=**"True"**/>

<CheckBox Name=**"SaveToFile"** Content=**"save to file"** Grid.Row=**"13"** Grid.ColumnSpan=**"4"** VerticalAlignment=**"Bottom"** Margin=**"0, 0, 0, 155"** IsChecked=**"false"**/>

<Button Name=**"pathSelector"** Content=**"file location"** Grid.Row=**"13"** Grid.ColumnSpan=**"3"** VerticalAlignment=**"Bottom"** Margin=**"1, 0, 20, 130"**/>

<ScrollViewer Name=**"textScrol"** Grid.ColumnSpan=**"4"** VerticalAlignment=**"Bottom"** Grid.Row=**"12"** MaxHeight=**"180"** Height=**"180"** Margin=**"100, 0, 0, 20"**>

<TextBlock Name=**"TextOut"** TextWrapping=**"Wrap"**/>

</ScrollViewer>

<TextBox Name=**"senderBox"** Grid.Row=**"13"** Grid.ColumnSpan=**"4"** VerticalAlignment=**"Bottom"** Height=**"20"** Margin=**"100,0,0,0"**/>

</Grid>

</Window>

## dataHandler.cs

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**Linq**;**

**using** System**.**Runtime**.**Remoting**.**Messaging**;**

**using** System**.**Text**;**

**using** System**.**Threading**.**Tasks**;**

**using** System**.**Windows**.**Controls**;**

**using** System**.**Windows**.**Media**;**

**using** System**.**Windows**.**Shapes**;**

**namespace** WPFCharting

**{**

**public** class channel

**{**

**private** **readonly** int index**;**

**private** System**.**Windows**.**Point point **=** **new** System**.**Windows**.**Point**(**MainWindow**.**xAxisStart**,** MainWindow**.**yAxisStart**);**

**private** **readonly** SolidColorBrush color **=** Brushes**.**Blue**;**

**private** Polyline chartPolyline**;**

**private** int offset**;**

**private** double deltaValue**;**

int i**,** Size**;**

**private** int metingen **=** 50**;**

**private** double height**;**

**private** **readonly** Canvas chart**;**

**private** TextBlock Limit**;**

**private** bool offLowerLimit**,** offHigherLimit**,** limitNotified**;**

**private** double**[]** temp **=** **null;**

**public** double**[]** Serie **=** **new** double**[**50**];**

**public** channel**(**int index**,** SolidColorBrush color**,** double height**,** Canvas chart**,** TextBlock Limit**)**

**{**

**this.**index **=** index**;**

**this.**color **=** color**;**

**this.**height **=** height **-** MainWindow**.**yAxisStart**;**

**this.**chart **=** chart**;**

**this.**Limit **=** Limit**;**

**}**

**public** void setOrigin **(**double height**)**

**{**

**this.**height **=** height **-** MainWindow**.**yAxisStart**;**

**}**

**public** void Sizing**(**int verzetting**)**

**{**

Size **=** metingen**;**

metingen **=** verzetting**;**

temp **=** **new** double**[**metingen**];**//maakt een rijdeljke array aan om data tijdelijk juist te zetten met de juiste limieten

**if** **(**Size **<** metingen**)**//ziet of de array groter of kleiner moet worden om dan de juiste waardes op de juiste plek te zetten

**{**

**for** **(**i **=** 0**;** i **<** Size**;** i**++)**

**{**

temp**[**i**]** **=** Serie**[**i**];**

**}**

**}**

**else** **if** **(**Size **>** metingen**)**

**{**

offset **=** Size **-** metingen **-** 1**;**

**for** **(**i **=** 0**;** i **<** metingen **-** 1**;** i**++)**

**{**

temp**[**i**]** **=** Serie**[**i **+** offset**];**

**};**

**}**

Serie **=** temp**;** // geeft de hoofd array de juiste waardes en limieten

temp **=** **null;**// leegt de tijdelijke array

**}**

**public** void Line**()**

**{**

//schuift alle waardes door met 1 plaats

**for** **(**i **=** 0**;** i **<** metingen **-** 1**;** i**++)**

**{**

Serie**[**i**]** **=** Serie**[**i **+** 1**];**

**}**

Double**.**TryParse**(**MainWindow**.**split**[**index **-** 1**],** **out** Serie**[**metingen **-** 1**]);**

**}**

**public** void drawingchannel **()** **{**

chart**.**Children**.**Remove**(**chartPolyline**);**//haalt de vorige polyline weg van onze chart

chartPolyline **=** **new** Polyline**()** //maakt een nieuwe polyline aan

**{**

Stroke **=** color**,**

StrokeThickness **=** 2**,**

**};**

chart**.**Children**.**Add**(**chartPolyline**);**//voegt de nieuwe polyline toe aan onze chart

//logia voor op elke x-waarde de juiste y waarde te plakken en deze dan vervolgens toe te voegen aan de polyline

offLowerLimit **=** **false;**

offHigherLimit **=** **false;**

**for** **(**i **=** 0**;** i **<** metingen**;** i**++)**

**{**

point**.**X **=** MainWindow**.**xAxisStart **+** i **\*** MainWindow**.**xinterval**;**

deltaValue **=** Serie**[**i**]** **-** MainWindow**.**ystart**;**

**if** **(**deltaValue **<** 0**)** **{** point**.**Y **=** MainWindow**.**yAxisLine**.**Y2 **+** 5**;** offLowerLimit **=** **true;** **}**

**else** point**.**Y **=** height **-** **(**deltaValue **\*** MainWindow**.**yscale**);**

chartPolyline**.**Points**.**Add**(**point**);**

**}**

**if** **((**offLowerLimit **||** offHigherLimit**))**

**{**

**if** **(!**limitNotified**)** **{**

Limit**.**Text **+=** $"{index} "**;**

limitNotified **=** **true;**

**}**

**}else** **if(**limitNotified**)**

**{**

Limit**.**Text **=** Limit**.**Text**.**Replace**(**$" {index} "**,** " "**);**

limitNotified **=** **false;**

**}**

**}**

**public** void **remove()**

**{**

chart**.**Children**.**Remove**(**chartPolyline**);**//verwijdert de polyline van onze chart

**}**

**}**

**}**