CHALMERS

Remote logging

Insamling samt lagring av data med grafritning av både sparad och realtidsdata.

Examensarbete inom Data- och Informationsteknik

VIKTOR ANDERSSON
OSCAR JOHANSSON

Institutionen för Data och Informationsteknik
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET
Göteborg, Sverige 2021

EXAMENSARBETE

Remote logging

Insamling samt lagring av data med grafritning av både sparad och realtidsdata.

VIKTOR ANDERSSON
OSCAR JOHANSSON

Institutionen för Data- och Informationsteknik
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET
Göteborg 2021

Remote logging
Insamling samt lagring av data med grafritning av både sparad och realtidsdata.
VIKTOR ANDERSSON
OSCAR JOHANSSON

©VIKTOR ANDERSSON, OSCAR JOHANSSON, 2021

Handledare: Torbjörn Tjelldén, Chalmers tekniska högskola
Andreas Petersson, Protrol
Björn Fälldin, Protrol

Examinator: Arne Linde, Chalmers tekniska högskola

Institutionen för Data- och Informationsteknik
Chalmers tekniska Högskola / Göteborgs Universitet
412 96 Göteborg
Telefon: 031-772 1000

The Author grants to Chalmers University of Technology and University of Gothenburg the non-exclusive right to publish the work electronically and in a non-commercial purpose make it accessible on the Internet.

The Author warrants that he/she is the author to the Work, and warrants that the Work does not contain text, pictures or other material that violates copyright law.

The Author shall, when transferring the rights of the Work to a third party (for example a publisher or a company), acknowledge the third party about this agreement. If the Author has signed a copyright agreement with a third party regarding the Work, the Author warrants hereby that he/she has obtained any necessary permission from this third party to let Chalmers University of Technology and University of Gothenburg store the Work electronically and make it accessible on the Internet.

Omslag:
Protrol fjärrterminal IPC4022.

Institutionen för Data- och Informationsteknik
Göteborg 2021

SAMMANFATTNING

Detta projekt har gått ut på att utveckla ett lagringssystem till en fjärrterminal från företaget Protol. Det främsta behovet är att kontinuerligt spara ner data. Tidigare fanns begränsade möjligheter för minneskapacitet. Systemet behövs eftersom frekvensavvikelser på kraftnätet har och kommer fortsätta öka i takt med att användandet av förnybara energikällor ökar. Ett stort fokus under projektet har varit att mäta begränsningar hos systemet och utreda vart dessa finns. Systemet innefattar hämtning, lagring och möjlighet till grafritning av data från terminalen. Detta hanteras av olika processortrådar för att förhindra de olika delarna av programmet från att blockera varandra. Mängden data, det vill säga antalet paket och storleken på paketen som skall hanteras, påverkar programmets prestanda. Programmet hanterar alla meddelanden vid den hastighet som fjärrterminalen kan skicka. Graferna uppdateras 1 gång per sekund, vilket är i långsammaste laget. Vid stor belastning på grafritningen märks dessutom en fördröjning på pakethantering av.

Nyckelord: Fjärrterminal, loggningstation, programmering, grafritning, Protrol

ABSTRACT

The aim of this project is to develop a remote-logging system for Protrol's remote terminal. The main goal is to be able to continuously and quickly save data as there were previously limited opportunities for memory storage. The system is needed since frequency deviation on the power grid increases with renewable power sources. A major focus during the project has been to measure the limits of the system and why the limits exist. The system includes retrieval, storage and the possibility of plotting data from the terminal. This is handled by different threads to prevent the different parts from interfering with each other. The program handles all messages at the speed that the remote terminal can send. The graph updates once per second, which is a bit on the slow side. In the event of a heavy load on the graph drawing, a delay in package handling is also noticeable.

Keywords: Remote terminal, logging station, programming, graph plotting, Protrol

FÖRORD

Efter långt sökande efter lämpligt examensarbete hittade vi till slut till Protrol. De presenterade ett projekt som fångade vårt intresse direkt. Vi tillhandahölls en laptop och en fjärrterminal och arbetade därefter så mycket som möjligt på distans. Arbetet har skett till stor del genom videomöte med skärmdelning. Avstämningar veckovis med diskussion har hela tiden lett oss i rätt riktning.

Examensarbetet har gjorts på Chalmers tekniska högskola hos institutionen för data- och informationsteknik trots att vi båda går på institutionen för elektroteknik. Båda studenternas utbildning är på kandidatnivå, 180hp, och därför har examensarbetet haft en omfattning om 15 hp.

Vi vill rikta ett stort tack till Protrol som lade ner tid och energi för att vi skulle få ett roligt och lärorikt examensarbete. Speciellt tack till Andreas Petersson och Björn Fälldin som har varit handledare. Tack till Johan Olson och Nils Åkesson som har hjälp oss att testa programmet på plats.

Tack till Robin Kaveh och David Junermark, två studenter på Chalmers, som har agerat bollplank när vi har haft många olika ideer. Tack till Torbjörn Tjelldén, vår handledare från Chalmers, som har hjälpt oss skriva rapport och fokusera på rätt delar.

TERMINOLOGI OCH FÖRKORTNINGAR

JSON	Ett sätt att strukturera upp data för att skicka information.
Paket	Den insamling information som fjärrterminalen ska skicka ut. All information i JSON-format.
Meddelande	Vad som skickas från fjärrterminal, varje utskick är ett meddelande.
utf-8	En typ av teckenkodning
Socket	Ände på en kommunikationskanal som används för att upprätthålla en konstant kommunikation. Med en fast nätverksadress och port.
Variabeltyp	Olika variabeltyper hanterar olika typer av data. Exempelvis en string hanterar text.
Wireshark	Ett program som mäter nätverkstrafik
Pandas	Ett bibliotek som är gjort för att hantera och analysera data.
Matplotlib	Ett av flera sätt att visualisera data. Det används för att plotta och genom funktionen FuncAnimation i biblioteket kontinuerligt hämtar data till plot.
Os	Möjliggör kommunikation och interaktion med operativsystem.
Sys	Tillhandahåller nödvändig information om konstanter, funktioner och metoder.
Datetime	Hanterar datum och tid. Möjlighet till översättning mellan olika format samt aktuell tid.
Json	Används för att hantera JSON-objekt, främst kodning samt avkodning.
Threading	Möjliggör trådbaserad parallellisering. Vid trådning stör de två olika processerna inte varandra så länge de inte använder samma variabler eller filer.
Csv	Används för att enkelt kunna spara data till CSV-fil med CSV-format. Kan även hanteras med andra bibliotek men CSV är det enklaste.
Pathlib	Används för att kontrollera om filväg existerar.
PyQt5	En koppling mellan python och QT som är flera olika C++-bibliotek. PyQt5 används för att skapa, visa och hantera användargränssnitt.

INNEHÅLLSFÖRTECKNING

1. INLEDNING 1
1.1 Syfte/Mål 1
1.2 Problembeskrivning/Frågeställningar 2
1.3 Avgränsningar 2
1.4 Metod 3
1.5 Rapportens struktur 3
2. TEKNISK BAKGRUND 4
2.1 Fjärrterminal 4
2.2 Modifierad fjärrterminal 4
2.3 Terminalens hemsida 4
2.4 Kommunikationspaket 5
2.7 Python 6
2.8 QT-designer 6
3. SYSTEMUTVECKLING 7
3.1 Hämtning av data 7
3.2 Lagring av data 9
3.3 Allmänna problem och optimering 10
3.3.1 Förlorade paket 10
3.3.2 Fördröjning 12
3.3.3 Sammanfattning av optimering 13
3.4 Användargränssnitt 13
3.5 Användargränssnittets funktioner 14
3.6 Grafritning 17
3.7 Sammanfattning av genomförande 20
4. RESULTAT 21
4.1 ANVÄNDARGRÄNSSNITT 25
5. SLUTSATS/DISKUSSION 26
5.1 Reflektion om utvecklingsarbetet 26
5.2 Utvecklingspotential 26
REFERENSER 28
BILAGOR 29

1. INLEDNING

Protrol är ett företag som dels arbetar med utveckling av produkter och tjänster inom elkraftsbranschen. Uppdraget i det här arbetet går ut på att överföra, hantera och lagra mätdata från en testutrustning, för prov på kraftstationer. Testutrustningen hanterar i det här fallet spänning, ström samt ett antal andra parametrar (både beräknade och generella analoga ingångar) hos ström- och spänningstransformatorer för en generator. De beräknade parametrarna kan exempelvis vara aktiv och reaktiv effekt.

I Protrols befintliga system finns det begränsade möjligheter att lagra större mängder data. Krav fanns på att mätdata ska kunna loggas kontinuerligt och sparas ner på disk. Därför behövde ett system utvecklas som sparar och hanterar all data för att kunna se händelser innan ett generatorhaveri eller liknande inträffar. Den större minneslagringen behövs eftersom frekvensavvikelser på kraftnätet har ökat med förnybara energikällor. Projektet genomförs för att underlätta användningen av förnybara energikällor. När användningen av förnybara energikällor ökar minskar de energikällor som skadar miljö och natur.

Det färdiga programmet ska användas på en ny applikation, som bygger på samma plattform som fjärrterminalen. Kontrollenheten används alltså bara under utvecklingen av detta program för att kontinuerligt kunna testa och se resultat. Programmet är avsett att senare användas med den nya applikationen. Detta för att övervaka mätdata vid generering av elproduktion från olika typer av kraftverk, t.ex vattenkraftverk, så de håller rätt frekvens vid inkoppling på elnätet. Ett annat syfte för programmet är att övervaka och mäta simulerad ödrift[1] hos lokala anläggningar.

All programmering för loggsystemet kommer ske i Python då det är det språk som används av företaget. Vilket format datan ska sparas i får studenterna välja själva utefter vad som verkar mest lämpligt ur synpunkt som diskuteras nedan. I ett första skede är textformat mest troligt.

1.1 Syfte/Mål

Huvudmålet är att utreda hur mycket data som kan mottas utan att riskera att förlora information. Det vill säga analysera hur många kanaler som kan hanteras och vid vilken överföringshastighet. Identifiera var som systemet begränsas. Analysera varför systemet innehar denna flaskhals.

[1] Ödrift är när ett geografiskt delområde har självständig elproduktion.

Programmet skall dessutom utvecklas med ett grafiskt användargränssnitt som visualiserar den uppmätta datan i grafer. Detta ska fungera för både sparad data samt realtidsdata. Det grafiska användargränssnittet kommer att programmeras i Python. Gränssnittet ska vara lätthanterligt, med ett fåtal knappar, och vid sidan om användargränssnittet ska graferna presenteras. Data ska kunna visualiseras i grafer för att få fjärrtermianlens mätresultat lättförståeligt.

1.2 Problembeskrivning/Frågeställningar

Examensarbetet går ut på att tillverka ett datalagringsprogram för att ta emot och lagra data från en fjärrterminal. Lagringsprogrammet implementeras i en dator som i detta fall är en vanlig laptop. Färjeterminalen har några olika protokoll som används för att kommunicera med programmet. Programmet ska bearbeta och läsa data från terminalen. Då extern lagring inte existerar blir arbetes största fokus att etablera tvåvägskommunikation samt bearbeta datan.

För att ta emot informationen till datorn kommer en kommunikationskanal behöva upprätthållas. Laptopen kommer även behöva svara kontrollsystemet enligt uppkopplingens kommunikationsprotokoll för att ge status om mottagen information. Detta kommer att ske genom att öppna en socket.

Överföringshastighet ska utvärderas. Det vill säga med hur hög kommunikationshastighet systemet kan ta emot och hantera information innan data går förlorad. Med hur hög hastighet kan fjärrterminalen kommunicera? Vilken storlek har paketen, hur många kanaler kan programmet hantera? Är det endast en viss variabel som begränsar systemet? Vad leder till att systemet begränsas?

1.3 Avgränsningar

Datan som kommer från fjärrterminalen antas vara korrekt. En annotering med information om tiden är synkroniserad eller inte syns i användargränssnittet, men någon kontroll av tid sker inte. Endast datainsamling från en fjärrterminal åt gången är möjlig.

Det finns inget behov hos företaget att kunna se data från fler dygn samtidigt. Enbart grafritning från en fil åt gången är möjlig. Det vill säga alla grafer som plottas är från samma datum. Projektet har även som avgränsning att kommunikationshastigheten är max 100 meddelanden per sekund, eftersom detta är en begränsning hos fjärrterminalen. Grundprogrammet med grafritning och användargränssnitt kommer vara utvecklat för PC.

1.4 Metod

Projektet påbörjades med att få kontakt med fjärrterminalen genom ett program som tillhandahållits av företaget, detta program användes endast för att testa uppkoppling mellan terminal och dator. När kontakt etablerats skrevs motsvarande kod i Python för att skapa en kommunikationskanal i vad som blir programmet. Då paket var uppdelade i flera meddelanden skrevs dessa ihop tills ett paket var komplett. De fullständiga paketen hanteras av JSON's bibliotek och innehållet sparades i fil.

För att skapa ett användargränssnitt användes programmet QT-designer. När kod för ett användargränssnitt genererats programmeras varje del var för sig för att få önskad funktion. Till detta behövdes biblioteket "PyQt5".

Plottningen utförs med hjälp av biblioteket "matplotlib" och skall befinna sig i programmets huvudtråd tillsammans med användargränssnittet. Detta då båda dessa funktioner behöver befinna sig i huvudtråden på grund av Pythons uppbyggnad. Matplotlib används för att plotta och genom funktionen FuncAnimation i biblioteket också kontinuerligt hämta data till plot.

1.5 Rapportens struktur

I kapitel 2. teknisk bakgrund ges information som krävs för att förstå resten av rapporten. I kapitel 3. systemutveckling finns en beskrivning av hur systemet är uppbyggt. Nästa kapitel, 4. resultat innehåller projektets resultat. I kapitel 5. slutsats/diskussion diskuteras resultaten och vad som kan utvecklas.

2. TEKNISK BAKGRUND

I detta kapitel presenteras begrepp, tekniker och verktyg för att ge bättre förståelse för programmets uppbyggnad och den datakommunikation som sker mellan programmet och fjärrterminalen.

2.1 Fjärrterminal

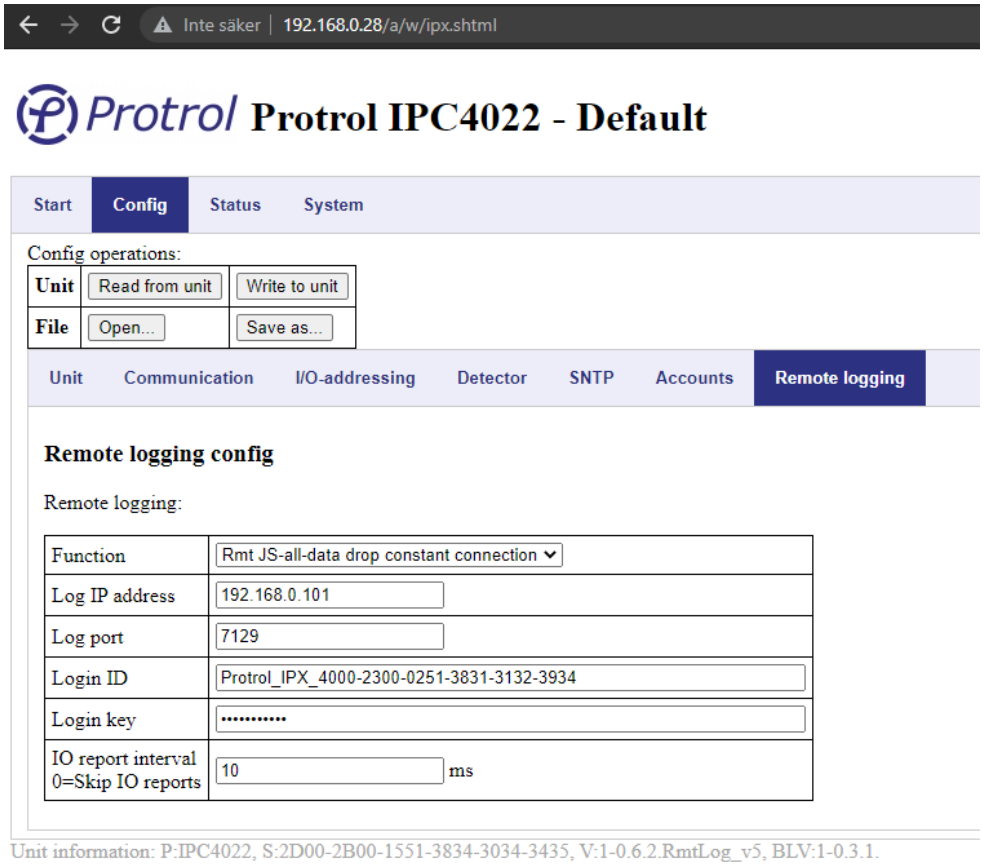
IPC4022 är en fjärrterminal [1]. Enhetens huvudsakliga uppgift är att detektera jordfel, fasavbrott och överström. Enheten kan dessutom utföra manövrer på kopplingsapparater och ta emot felsignaler. Denna IPC har tre stycken analoga, samt åtta binära ingångar. Den är vidare utrustad med tre binära utgångar. Företaget har olika varianter på enheten med olika antal analoga ingångar och binära ut/-ingångar,. Vid behov kan ytterligare kontakter anslutas med hjälp av expansionskort. Det finns en RJ-45 port på konsolen. Denna används vid kommunikation med PC. Fjärrterminalen används inom elproduktion, kopplingsstationer inom eldistribution och driftövervakning av övriga anläggningar.

2.2 Modifierad fjärrterminal

I detta projekt används en terminal med modifierad mjukvara som har tillhandahållits av Protrol. Den har 80 binära in/-utgångar och 64 analoga ingångar, alltså totalt 224 kanaler. Ett fåtal av kanalerna har simulerade värden och vissa andra kan manipuleras genom att koppla in en 12-volts spänningskälla till en binäringång. Den manipulerade ingången kommer då visas som logisk etta i systemet. De flesta av kanalerna har alltså inget värde. Anledningen till att 224 kanaler har lagts in i systemet är för att lättare kunna mäta prestanda. Fjärrterminalen skickar ut meddelanden innehållande värden på alla kanaler genom kommunikationsporten. Meddelanden skickas med, från hemsidans inställningar, justerbart intervall.

2.3 Terminalens hemsida

Inställningar för terminalen ändras via ett användargränssnitt som nås genom en webbläsare, där skrivs enhetens IP-adress in i adressfältet. Användaren behöver sedan logga in med hjälp av ett användarnamn samt lösenord för att få åtkomst till inställningarna. För att kunna få tillgång till hemsidan måste fjärrterminalen vara ansluten till datorn, detta sker via ethernetkabel. Här finns bland annat möjlighet att ändra med vilket intervall data skickas, enhetens tid och olika anslutningsinställningar.



Figur 1: Fjärrterminalens hemsida med kommunikationsinställningar.

2.4 Kommunikationspaket

Alla paket följer JSON standard och innehåller information från fjärrterminalen.

- Hello-paket skickas vid uppkoppling. Här presenteras den anslutna enheten med ID, information om eventuellt lösenord och ifall tiden är synkroniserad eller inte.

- Info-paket skickas efter Hello-paket innehållande vilka kanaler som finns tillgängliga och information om antal decimaler och enhet på vardera kanal.

- Data-paket skickas löpande med intervall som har bestämts från hemsidan. Innehållet består av mätvärden på alla kanaler.

2.7 Python

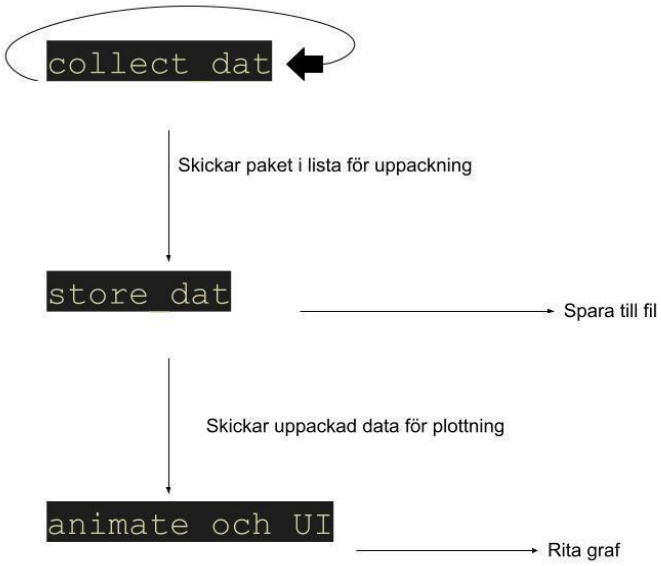
Python är ett objektorienterat programspråk. Språket är lättförståeligt och tydligt. En skillnad mellan python och andra programmeringsspråk är att Pythons trådhantering är trådbaserad parallellisering.

2.8 QT-designer

En integrerad utvecklingsmiljö för utveckling av applikationer med grafiskt användargränssnitt. Drag- och släppbaserat program som skapar programfönster och genererar en pythonkod med fönstrets utseende. Programfönstrets funktion programmeras av utvecklaren.

3. SYSTEMUTVECKLING

Fjärrterminalen hämtar data från elnätet genom mätning som skickas till PC genom en kommunikationskanal. Mätningarna loggas i programmet som består av fyra delar. Första delen hämtar in data från porten genom att etablera en koppling i Python mellan fjärrterminal och datorn. Här kontrolleras att paketen följer standard för JSON. Den andra delen packar upp JSON-paketen och sparar ner informationen från dessa i fil. För att interagera med programmet behövs den tredje delen, användargränssnitt. Här påverkas grafer med olika val. Och för att visualisera datan behövs den sista delen, grafritningen. Hur de olika delarna kommunicerar med varandra illustreras i bilden nedan.



Figur 2: Visar hur det olika trådarna arbetar tillsammans.

I vardera delkapitel förklaras funktionerna för att vidga förståelsen och visa hur funktioner fungerar och är uppbyggda.

3.1 Hämtning av data

Kommunikationen mellan fjärrterminalen och datorn upprättas på en port. När kommunikationen är uppsatt skickas meddelanden med intervaller som bestäms från hemsidan. Datorn lyssnar på porten och tar emot alla meddelanden som kommer in. Biblioteket för denna kommunikation är “socket” som skapar en koppling mellan PC och IPC. För att upprätthålla kommunikation mellan terminalen och PC måste den valda IP-adressen ställas in på datorns nätverkskort. Dessutom måste denna IP adress och korrekt port ställas in i etableringen av socket i programkoden. Biblioteket används för att kontinuerligt hämta data från port.

Först ska data från Hello-paketet hanteras. Är ett lösenord valt på terminalens hemsida så måste programmet svara med det valda lösenordet för att starta kommunikationen. Skulle inte ett lösenord vara valt så börjar fjärrterminalen skicka paket. Info-paketets information ska sparas ner i programmet för att senare sparas i fil. Därefter skickas Data-paket som sparas undan till rätt kanal. Denna del av programmet skall befinna sig i en egen tråd för minska processor-kön, detta blir kodens första tråd. För att skapa trådar används biblioteket "threading" som möjliggör att olika delar av programmet kan köras parallellt.

Inkommen data från port lagras teckenkodad i en variabel av typen byte enligt standard för pythons sockethantering. I byte skrivs hela meddelandet på en rad som börjar med "b'". Meddelanden innehåller inte radbyten eller indenteringar utan istället "\n" och "\t" i textform, samt att vissa parenteser inte följer med. Detta tyder på att meddelande blivit teckenkodat. Standarden för JSON är förlorad hos paketen. Meddelanden avkodas med utf-8 och läggs i en variabel av typen string för att senare i programmet kontrollera om meddelanden är hela paket eller ifall flera meddelanden behöver läggas ihop för att bilda ett paket. Efter ändringarna och konkateneringen följer paketen JSON-standard, förutsatt att det bara är ett paket.

Det noterades tidigt i projektet att paket kom in som två meddelanden. Detta undersöktes för att redogöra ifall paketen mottogs uppdelade. Från Wireshark kan det läsas ut att meddelanden som är större än 590 bytes har blivit uppdelade av fjärrterminalen. Paketen som skickas kontinuerligt är på mer än en kilobyte. Därför kommer paketen in som två meddelanden. Tidigare har fjärrterminalen inte behövt hantera så stora paket och det har därför inte varit ett problem. Eftersom detta är testutrustning för att utveckla systemet räcker det att veta att uppdelningen inte sker i programmet. Se figur 2 för nätverkstrafik på port. När det bekräftats att meddelanden är delade och att det inte kunde ändras i nuläget behövde meddelanden sättas ihop. Detta åtgärdades genom att lägga ihop delar av meddelanden tills det blev helt.

No.	Time	Source	Destination	Protocol	Length	Info
8	1.086513	192.168.0.101	192.168.0.28	TCP	54	7129 → 52048 [ACK] Seq=1 Ack=1902 Win=64978 Len=0
9	2.007554	192.168.0.28	192.168.0.101	TCP	590	52048 → 7129 [PSH, ACK] Seq=1902 Ack=1 Win=2144 Len=536
10	2.049179	192.168.0.101	192.168.0.28	TCP	54	7129 → 52048 [ACK] Seq=1 Ack=2438 Win=65392 Len=0
11	2.061886	192.168.0.28	192.168.0.101	TCP	470	52048 → 7129 [PSH, ACK] Seq=2438 Ack=1 Win=2144 Len=416
12	2.103146	192.168.0.101	192.168.0.28	TCP	54	7129 → 52048 [ACK] Seq=1 Ack=2854 Win=64976 Len=0
13	3.002483	192.168.0.28	192.168.0.101	TCP	590	52048 → 7129 [PSH, ACK] Seq=2854 Ack=1 Win=2144 Len=536
14	3.042681	192.168.0.101	192.168.0.28	TCP	54	7129 → 52048 [ACK] Seq=1 Ack=3390 Win=65392 Len=0
15	3.043613	192.168.0.28	192.168.0.101	TCP	469	52048 → 7129 [PSH, ACK] Seq=3390 Ack=1 Win=2144 Len=415
16	3.084361	192.168.0.101	192.168.0.28	TCP	54	7129 → 52048 [ACK] Seq=1 Ack=3805 Win=64977 Len=0
17	3.999962	192.168.0.28	192.168.0.101	TCP	590	52048 → 7129 [PSH, ACK] Seq=3805 Ack=1 Win=2144 Len=536
18	4.041016	192.168.0.101	192.168.0.28	TCP	54	7129 → 52048 [ACK] Seq=1 Ack=4341 Win=65392 Len=0

Figur 2: Nätverksloggning mellan PC och fjärrterminal genom wireshark.

Förutom att paketen måste vara hela för att följa standard för JSON behöver de även vara isolerade. Vid högre belastning kan flera paket hämtas i samma meddelande då flera meddelanden väntar på att tas emot. Tidigare blev en ansenlig mängd paket förlorade då de inte var isolerade. För att garantera att meddelanden är åtskilda används en if-sats. Denna påträffar de fall där två eller fler paket befinner sig i samma meddelande. Då läggs tecknet "^" till, vilket aldrig förekommer i meddelandet, mellan varje paket. Därefter används funktionen "split" med argumentet "^" som fungerar genom att dela upp AllData varje gång tecknet förekommer. De uppdelade paketet läggs i en lista som förlänger allDataList vilket är listan i väntan på uppackning. Då allt i AllData har hanterats rensas variabeln.

```
if AllData.find('}\n{') != -1:
      AllData=AllData.replace('}\n{', '}\n^{')
      allDataList.extend(AllData.split("^"))
      AllData=""
```

3.2 Lagring av data

Den andra tråden skall hantera datan som ska lagras i fil. Informationen från paketen packas upp med biblioteket "json". Detta fungerar så länge meddelandet följer standard för JSON. De uppackade objekten ska lagras i en CSV-fil, för att inte ta upp för mycket minne, men ändå vara lätthanterliga. Denna fil ska sparas med dagens datum som filnamn och finns det inte redan en fil med detta namn ska en CSV-fil skapas som heter dagens datum. För att spara filen som dagens datum användes biblioteket datetime som hanterar datum och tid. För att lagra datan i CSV-fil nyttjades Pythons bibliotek "csv" för att enkelt kunna hantera CSV-filer. Kan även hanteras med hjälp av andra bibliotek men CSV är det mest lätthanterliga. För att kontrollera ifall filen existerar, används "pathlib". Finns inte filen behöver den skapas och en header behöver skrivas.

I dagens CSV-fil ska först vetskapen om vilka in- och utgångar som finns tillgängliga sparas ned. Denna information tas ifrån Info-paketet och sparas som en header för att inte interferera med datan när detta ska läsas av senare. Efter detta så skall all data från Data-paketet sparas ned i CSV-filen samt vilken tidsmarkör varje paket har, för lättare utläsning senare.

För att ange när en ny uppkoppling har etablerats skrivs en kommentar i filen. Detta för att veta vilken fjärrterminal data har lagrats från och när det ändrats. Eftersom en fil behöver börja med en header behöver kommentaren skrivas efter kolumnnamnen. Detta problem löses genom att sätta en flagga till sann när en ny uppkoppling förekommer. Ifall det behöver skrivas en header skrivs kommentaren efter denna. För att kommentaren inte ska vara i vägen när CSV-filen söks efter data användes en parameter ur pandas read_csv. Denna parameter fungerar genom att "comment=#" skrivs för att ange att allt som börjar med "#" är en kommentar. Om inte argumentet anges så antas att det inte finnas kommentarer i filen.

Om användaren vill se data i realtid eller tidigare data från samma dag så läses dessa värden från en lista som byggs på under dagen. Detta för att inte interferera med lagringen av data, vilket görs ifall samma fil öppnas på två olika ställen i koden samtidigt. För att plotta sparad data används däremot filläsning. Då det är olika filer för varje dag kan till exempel gårdagens fil öppnas för att plottas och läsas samtidigt som data sparas ner till dagens fil utan att de interfererar med varandra. För att hämta datan används “Pandas”. Biblioteket nyttjas för att kunna hämta kolumner från csv-fil och lägga i variabler på ett enkelt sätt.

Dessa funktioner, hämtning och lagring, har ändrats under projektets gång. Hur de var tidigare och varför de ändrades behandlas i nästa delkapitel, 3.2.

3.3 Allmänna problem och optimering

Till en början kunde programmet inte hantera meddelanden särskilt fort. Flera paket förlorades, särskilt vid kortare meddelandeintervall. Detta felsöktes och delades upp till två problem, förlorade paket och fördröjning. Dessa problem och hur de undviks förklaras i detta delkapitel.

3.3.1 Förlorade paket

Paket räknas som förlorade om de inte följer JSON-standard vid avläsning. Detta händer i de fall ett paket inte är helt. Eftersom inget paket är helt när det kommer in så sätts paket ihop av programmet. Blir det något fel där kommer paketet inte kunna hanteras. Vid kortare meddelandeintervall, vilket leder till högre belastning, ökar antalet fel vid ihopsättningen. Paketen sitter ihop eftersom Python hämtar data från port i variabeltypen byte och när data avkodas igen har programmet förlorat vetskapen om var paketen börjar och slutar.

Inkommande “Hello-paket” ser ut så här:

```
b'{\n\t"pktType":"Hello",\n\t"uid":"Protrol_IPX_4000-2300-0251-3831-3132-
3934",\n\t"timestamp":1483275791756,\n\t"timeSynchronized":false,\n\t"key":
"Not set
yet",\n\t"unitType":"IPC4022",\n\t"unitSwType":"IPC4/IPX1SW",\n\t"unitSwver
":"1-0.6.2.RmtLog_v5"\n}\n'
```

Allt står på en rad och har förlorat JSON-standarden. Detta avkodas med utf-8 och ser därefter ut som nästkommande stycke.

```
{
    "pktType":"Hello",
    "uid":"Protrol_IPX_4000-2300-0251-3831-3132-3934",
    "timestamp":1483275991015,
    "timeSynchronized":false,
    "key":"Not set yet",
    "unitType":"IPC4022",
    "unitSwType":"IPC4/IPX1SW",
    "unitSwver":"1-0.6.2.RmtLog_v5"
}
```


Vilket innebär att efter avkodningen är JSON-standard tillbaka.

Däremot så tappas “b’” vid avkodningen, då avkodad text läggs i en string, och därmed vetskapen om var paket börjar och slutar. Då varje meddelande inte är ett helt paket konkateneras flera meddelanden tills ett paket blir komplett. Därför har det vid högre kommunikationshastighet märkts tydligare då meddelanden kommer med högre intensitet och ibland läggs på hög innan de hanteras. Det leder till att högen inte kan hanteras då flera JSON-paket “sitter ihop” eftersom alla meddelande som låg på kö har konkaternerats och inte längre följer standarden som ett paket.

En tidigare anledning till detta problem var när data skulle skrivas in i CSV-fil i ena tråden och samtidigt hämtas ut från samma fil i en annan tråd så blockerar trådarna varandra. Detta blir en begränsning när information inte kan skrivas in i filen utan väntar på att filen ska bli tillgänglig. Data fortsätter anländer till porten medan funktionen väntar och när inkommen data sedan ska hanteras har det kommit fler än ett paket. För att programmet skulle fungera vid försök att packa upp flera ihopsittande paket räknades meddelandet som förlorat och togs bort.

Detta åtgärdades först delvis genom att dela upp hämtning av data och hantering samt lagring av data till två skilda funktioner på olika trådar. Datan som kommer in läggs kontinuerligt i en efterföljande ledig plats i en lista från tråden för hämtning. Listan hämtas av tråden för hantering och lagring och minskar för varje hanterat objekt. En annan aspekt som har optimerat programmet är att när data från innevarande dag ska plottas hämtas denna data inte från fil. Vilket leder till att programmet inte blockeras då inte samma fil öppnas på två ställen samtidigt. I stället är nuvarande dags data globala variabler som rensas för varje dygn.

Detta har senare uppdaterats ytterligare genom att dela upp ev. ihopsatta paket med en split-funktion och lägga alla hämtade paket i listan som hanteras av tråden för datahantering. Detta förklaras mer ingående under delkapitel 3.1 Hämtning av data. Efter detta förloras inga paket. Dock kan systemet vid hård belastning, alltså många paket som sitter ihop flera gånger, hamna upp till 0.5 sekunder efter med hantering av data. Denna fördröjning åtgärdas relativt fort. Detta eftersom det är mindre tidskrävande att packa upp paket än högsta möjliga kommunikationshastighet hos fjärrterminalen, inom rimliga gränser på grafritandet.

Om ett meddelande som inte följer standarden försöker avkodas stannar programmet på grund av ett felmeddelande. För att undvika detta hanteras alla meddelanden av en try-funktion, denna funktion testar att köra vad som ligger i “try:”. Om det ger felmeddelande körs i stället det som ligger i “except:” där meddelandet som inte kunde avkodas sparas till egen fil.

3.3.2 Fördröjning

När programmet inte längre förlorar paket har det noterats problem med grafritningsfunktionen. För att inte hindra kö-systemet är grafritningen begränsad till en gång per sekund. Vid vissa fall, till exempel när alla grafer ritas i båda subplottarna eller när många mätvärden ritas ut på flera kanaler, kommer kön för hantering växa under korta perioder.

#	Funktion	Beroende				
			10 ms			
1	Collect		medd. kommer in	Hantering av medd.	Börjar från början och väntar på nästa paket, kan även hantera flera paket samtidigt.	
2	Store	1			5 ms	Börjar om och väntar på att hantera fler paket
3	Animate	2	Anropas en gång per sekund och kontrollerar om det finns något som ska plottas			100 ms

Figur 3: Tidsschema som visar vad som tar tid i koden.

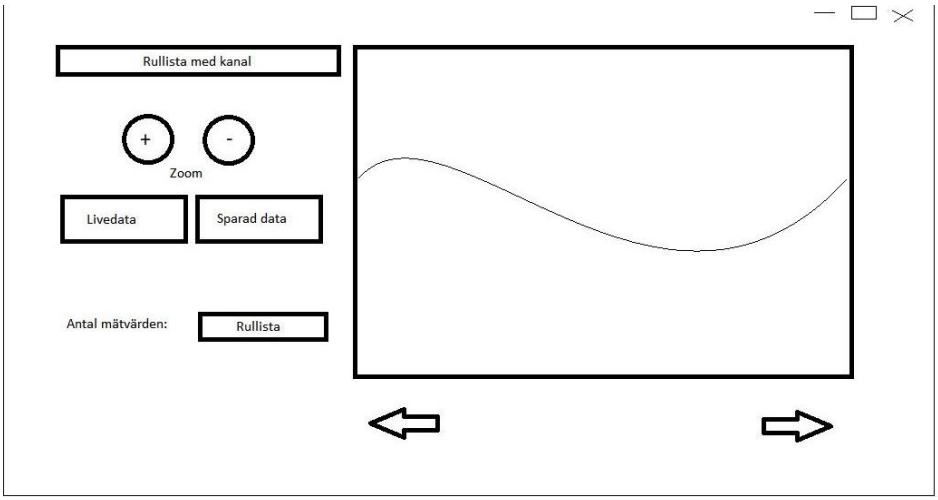
Då sparning av paket är det som tar kortast tid borde inte kön byggas på men eftersom grafritningen behöver data från paketlagringsfunktionen blockerar trådarna varandra. Detta problem uppstår enbart vid hög belastning. Vilket också medför att tider från tidsschemat inte nödvändigtvis stämmer överens då mätningarna är genomförda vid normal belastning.

3.3.3 Sammanfattning av optimering

Den främsta anledningen till att paketen nu kan hanteras är att kösystemet delar på paket som sitter ihop. De delade paketen hålls sedan åtskilda i en lista i väntan på att hanteras. Efter denna ändring kan programmet hantera alla inkommande paket istället för att radera de paket som ankom medans programmet var upptaget med nedsparning av tidigare paket. Detta tillsammans med att trådar har införts gjorde att målet att kunna ta emot 100 meddelanden per sekund uppnås.

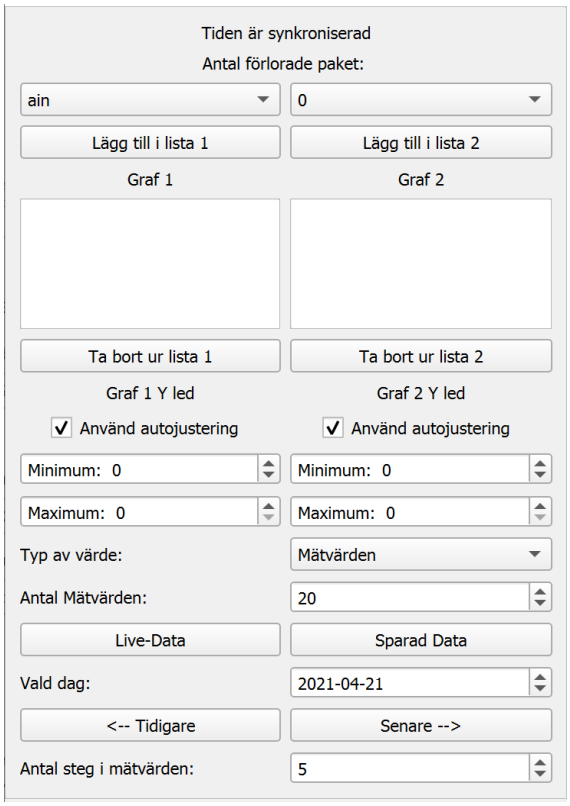
3.4 Användargränssnitt

Den större delen utav användargränssnittet är konstruerat i QT designer [2]. Detta valdes då det är ett lättanvänt program för att konstruera enklare gränssnitt. Först skapades en skiss med en ide om hur programmet skulle se ut, alltså vilka knappar, rullistor och liknande som behövdes för alla funktioner. Denna skiss illustreras i figuren nedan.



Figur 4: Första skissen för hur användargränssnittet skulle se ut.

Skissen förverkligades visuellt i QT som sedan genererade den Python kod som krävdes för att få det önskade gränssnittet. Efter detta behövde funktioner för samtliga knappar skapas för önskat resultat. Slutligen ändrades stiltypen för att generera ett, till utseendet, modernare program, jämfört med det fyrkantiga QT själv skapar. När allt detta var gjort behövde alla funktioner som har med knappar, rullistor och så vidare testas och slutresultatet blev följande:



Figur 5: Visar det kompletta användargränssnittet

3.5 Användargränssnittets funktioner

För att användargränssnittets knappar ska fungera behöver en funktion anropas vid interaktion med fönstret. Ett exempel på hur funktioner anropas är:

```
self.buttonLiveData.clicked.connect(self.livedata)
```

Detta innebär att när knappen “buttonLiveData” klickas på anropas funktionen “livedata”

Funktionen “addGraph1” anropas när knappen med texten “Lägg till i graf 1” trycks ned. Då tas kanalens namn och nummer och läggs till i listan för graf 1. Varje gång funktionen blir anropad läggs även alla kanaler med nummer till i listan “plot1_terms” för att användas vid grafritning.

```
def addGraph1(self):
    global plot1_terms
    channel_1 = self.comboBox.currentText()
    number_1 = self.comboBox_2.currentText()
    self.listWidget.addItem(channel_1 + number_1)
    plot1_terms =  [str(self.listWidget.item(i).text()) for i in
range(self.listWidget.count())]
```

För att kunna ta bort objekt i listan anropas funktionen “remove1”. Då tas det markerade objektet i listan bort och “plot1_terms” uppdateras.

```
def remove1(self):
    global plot1_terms
    self.listWidget.takeItem(self.listWidget.currentRow())
    plot1_terms =  [str(self.listWidget.item(i).text()) for i in
range(self.listWidget.count())]
```

Båda ovanstående funktioner för att lägga till i graf två är identisk fast med “2” i variabelnamnen i stället.

När antalet mätvärden ändras anropas funktionen “valuechange”. Då kontrolleras inställt intervall som meddelanden skickas med och om det är inställt på antal mätvärden eller antal sekunder. Vid respektive inställning beräknas hur många mätvärden som ska visas och vad som ska stå i användargränssnittet.

```
def valuechange(self):
    global Nof_object, freq_now
    if stored:
        freq_now=freq[x_line]
    else:
        freq_now=program_freq
    if self.comboBox_3.currentText() == "Sekunder":
        Nof_object = int((self.spinBox1.value())*(freq_now))
        self.label1.setText("Antal Sekunder:")
        self.label_4.setText("Antal steg i sekunder:")
    else:
        Nof_object = int(self.spinBox1.value())
        self.label1.setText("Antal Mätvärden:")
        self.label_4.setText("Antal steg i punkter:")
```

När live-visning är valt ändras variabeln “filename”, som väljer vilken fil som data ska hämtas från, till dagens datum. När dagens datum ska plottas hämtas denna data dock inte ur filen. Flaggan och variabeln “filename” för informationen vidare genom programmet som i stället hämtar data ur globala variabler. Intervallet som meddelanden skickas med hämtas från det senaste meddelandet och en flagga “stored” sätts till “False”. Denna flagga används sedan för att veta om livedata eller sparad data används, då detta skiljer sig åt.

```
def livedata(self):
    global filename, stored, freq_now
    filename=str(date.today())
    freq_now=program_freq
    stored=False
```

Denna “storeddata” fungerar på omvänt sätt jämfört föregående, när sparad data är valt så får variabeln “filename” värdet av det valda datumet. Meddelandehastigheten hämtas från det första meddelande som mottogs under aktuell dag och tidigare nämnda flagga sätts till sann.

```
def storeddata(self):
    global filename, stored, freq_now
    chosenDate=self.dateEdit.date()
    filename=chosenDate.toString(QtCore.Qt.ISODate)
    freq_now=freq[x_line]
    stored=True
```

Funktionen “scroll_left” fungerar bara vid visning av sparad data. Funktionen förflyttar grafen till vänster med inställt värde. Beroende på om operatören har valt sekunder eller mätvärden är det olika beräkningar för att programmet ska flyttas rätt antal mätvärden. Begränsning för att inte flytta för långt åt vänster (under 0) finns med för att graffönstret ska fungera vid ogiltig indata.

```
def scroll_left(self):
    global x_line
    if self.comboBox_3.currentText() == "Sekunder":
        if int(x_line-(self.spinBox.value()*(freq_now)))<=0:
            x_line=0
        else:
            x_line=x_line-int(self.spinBox.value()*(freq_now))
    else:
        if x_line-self.spinBox.value()<=0:
            x_line=0
        else:
          x_line=x_line-self.spinBox.value()
```

En funktion “scroll_right” finns också som fungerar på samma sätt som tidigare funktion, fast flyttar grafen framåt. Även denna med begränsning för gå för långt, alltså att välja värden som inte finns.

Det finns en funktion som kontrollerar om tiden är synkroniserad vid valt mätvärde och skriver ut det i användargränssnittet. Den fungerar både på nutida data och sparad data.

```
def Timesynced(self):
    if stored:
        if time_sync[x_line]:
            self.label_6.setText('Tiden är synkroniserad')
        else:
            self.label_6.setText('Tiden är inte synkroniserad')
    else:
        for rows in time_sync:
            if rows==True:
                self.label_6.setText('Tiden är synkroniserad')
            else:
                self.label_6.setText('Tiden är inte synkroniserad')
```

3.6 Grafritning

För att utveckla grafritningen har biblioteket “matplotlib” använts då biblioteket är väldokumenterat [3]. Kod som använts bryts under detta delkapitel ner och förklaras.

Nedanstående kod körs bara en gång och tar bort knappar från graf-fönstret och startar två subplots. Knapparna tas bort då flera av dem förlorar sin funktion när grafen kontinuerligt uppdateras, t.ex. “Back” och “Zoom”. Andra implementeras i koden, t.ex. “Customize”.

```
plt.style.use('fast')
toolbar = plt.get_current_fig_manager().toolbar
unwanted_buttons = ['Home', 'Back', 'Forward', 'Pan', 'Zoom', 'Customize',
'Subplots']
for x in toolbar.actions():
    if x.text() in unwanted_buttons:
        toolbar.removeAction(x)

fig = plt.figure(1)
ax1 = fig.add_subplot(2,1,1)
ax2 = fig.add_subplot(2,1,2)
plt.tight_layout()
```

Nedanstående stycke kontrollerar ifall dagens datum är valt, och om så är fallet används variabler i stället för fil. Variablerna hanteras av lagringstråden och är globala vilket är anledningen till att det är möjligt att hämta datan i denna tråd.

```
if filename!=str(date.today()):
    read_file = Path(dirname(__file__)+"/"+filename+".csv")
    if read_file.exists():
        if file_found>0:
            print("Filen hittades!")
            file_found=0
        data = pd.read_csv(filename+'.csv', comment='#', low_memory=False)
        time_values = data['Time']
        time_sync = data['timesync']
        freq = data['freq']
    else:
        if file_found==0:
            print("Filen hittades inte")
            file_found+=1
else:
    time_values=timeList
    time_sync=timeSyncList
```

I nästkommande stycke visas hur grafritandet genomförs. Det finns en if-sats för varje graf, “plot1_1-5” och “plot2_1-5”. Hur många grafer som ska ritas avgörs av längden på listorna “plot1/2_terms”, som kommer från användargränssnittet. Det går att rita två signaler med olika storheter, exempelvis spänning och frekvens, i samma graf men bör av användaren undvikas för tydlighetens skull. För realtids-grafritning sparas inkommande data i en lista som skickas till grafritnings-tråden, där informationen tas hand om och ritas.

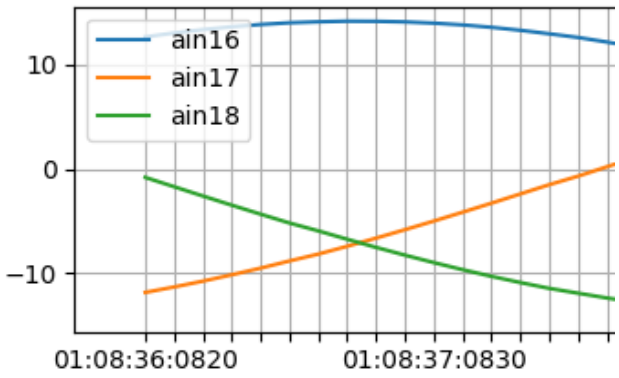
```
nOfGraphs1=len(plot1_terms)
if  nOfGraphs1>=1:
      if plot1Auto:
            ax1.autoscale()
        else:
                  ax1.set_ylim([plot1ymin,plot1ymax])
        if filename!=str(date.today()):
                  plot1_1 = data[plot1_terms[0]]
        else:
                  if plot1_terms[0][:3]=="ain":
                  for ix in range(0,len(dataAinList)):

      plot1_1.append(dataAinList[ix][int(plot1_terms[0][3:])])
                  if plot1_terms[0][:3]=="bin":
                  for ix in range(0,len(dataBinList)):

      plot1_1.append(dataBinList[ix][int(plot1_terms[0][3:])])
                  if plot1_terms[0][:4]=="bout":
                  for ix in range(0,len(dataBoutList)):

      plot1_1.append(dataBinList[ix][int(plot1_terms[0][4:])])
        if stored:
                  ax1.plot(time_values[x_line:Nof_object+x_line],
plot1_1[x_line:Nof_object+x_line], label=plot1_terms[0])
        else:
                  time_values=time_values[Nof_object:]
                  plot1_1=plot1_1[Nof_object:]
                  ax1.plot(time_values[-Nof_object:], plot1_1[-
Nof_object:], label=plot1_terms[0])
        ax1.legend(loc='upper left')
        ax1.grid()
```

Till en början lästes aktuell grafritning ur CSV-filen, men detta blev ett problem då inte samma fil kan öppnas samtidigt på två olika ställen i koden. Om sparad data är valt ritas grafen med start på vald dags första objekt och om live är valt ritas nuvarande objekt. Sparad data från nuvarande datum hämtas inte ur fil. “Legend” är med på varje subplot för att uppdateras men “grid” är bara med på första grafen i vardera fönster då den uppdateras med grafens fönster.

Nästa stycke behåller var femte tidsmarkering i grafen för att underlätta tidsavläsning. Detta utförs genom att använda modulusberäkning vilket innebär att vid de fall där antal mätpunkter inte är jämnt delbart med 5 visas färre tidsmarkeringar.

```
n = Nof_object/5
[l.set_visible(False) for (i,l) in enumerate(ax1.xaxis.get_ticklabels()) if
i % n != 0]
[l.set_visible(False) for (i,l) in enumerate(ax2.xaxis.get_ticklabels()) if
i % n != 0]
```

Två subplottar används då ett krav från beställaren var att kunna plotta data med två olika enheter, exempelvis spänning och frekvens. I varje subplot tillåts fem grafer för data ska presenteras tydligt i plotfönstret och minska både delay och risk för förlorade paket. Längst upp till vänster på vardera plot visas vilka kanaler som plottas i vilken färg. Det ska vara valbart i användargränssnittet vilka kanaler som ska plottas.



Figur 6: Visar etiketterna till grafen.

Hela programmet har senare delats för att testa eventuella skillnader med att använda en “oscilloskop-vy” till grafritningen. Alltså att x-axeln återanvänds och det äldsta värdet ersätts med det nya för att inte behöva rita om hela grafen. Det här har tyvärr inte givit önskat resultat ännu men har absolut potential att uppnå en mindre krävande grafritning.

3.7 Sammanfattning av genomförande

Hämtning sker genom en port. Datan som kommer in behöver anpassas för att kunna hanteras och lagras i fil. Här har det stötts på två svårigheter som till större del har lösts. Ena svårigheten har varit att paket inte har kunnat hanterats vilket har lösts genom flera olika modifikationer. Andra svårigheten har varit att det förekommer fördröjning vid hög belastning. Fördröjningen har minskats genom att begränsa uppdatering av graf. Ett användargränssnitt har konstruerats för att kunna interagera med programmet. För att visualisera data utförs grafritning med hjälp av matplotlib.

4. RESULTAT

Programmet lagrar data som planerat, med minimalt antal, under 0.001‰, förlorade paket vid hastigheter upp till 100 meddelanden per sekund. Vilket är begränsningen hos fjärrterminalen då den inte kan kommunicera med högre hastighet än så. Den låga andel förlorade paket har uppnåtts genom att använda en lista mellan de två trådar som hanterar inhämtning och lagring av data.

Från början var antal förlorade paket i korrelation med vilket intervall fjärrterminalen skickar paket med. Vid längre intervall förlorades nästintill inga paket alls, medan om den sände 100 meddelanden per sekund så förlorades fler. Kvoten lyckades först halveras på grund av två ändringar. Den första är de två trådar som har lagts till och bidrar till att programmet hinner hämta data från porten när ny data finns tillgänglig. Samt att vid grafritning för aktuell dag hämtas data från flera globala listor som uppdateras kontinuerligt i stället för att hämta från fil. Mer ingående förklaring finns i delkapitel 3.3.1 Förlorade paket. När både tråden för hantering av data och tråden för grafritning läste och skrev till samma fil skapades ett ofrivilligt kö-system. Det här då bara en av funktionerna kan öppna samma fil samtidigt. Detta ledde till att systemet inte kunde spara värden samtidigt som plottningen hämtade ut värden till grafen.

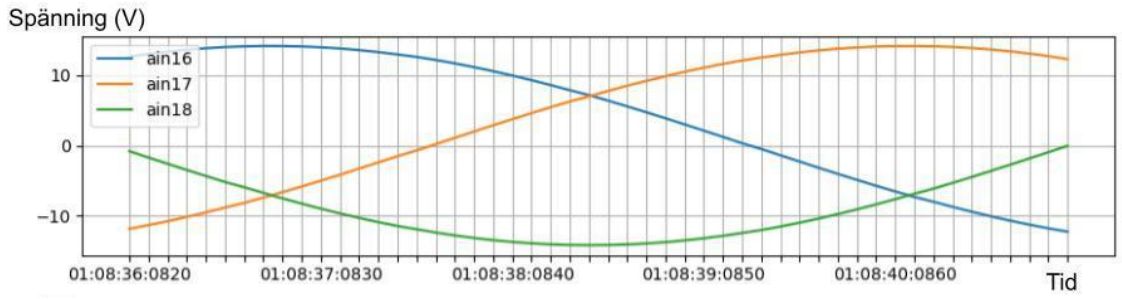
Efter dessa ändringar förlorades fortfarande för många paket vilket ledde till en slutgiltig ändring. Här skapas ovan nämnda lista för att hantera paket som sitter ihop och därmed annars hade gått förlorade. Den fungerar genom att enskilda, kompletta paket läggs till i listan enligt tidsordning från inhämtningstråden. Tråden för hantering bockar sedan av listan genom att ta ut ett paket och hantera det för att därefter ta nästa. För vidare information se delkapitel 3.3.1 Förlorade paket. Detta fungerar då tiden för att packa upp ett paket är liten, mindre än 10 ms. 10 ms är det kortaste intervallet för meddelanden att skickas från fjärrterminalen. Detta betyder att kön aldrig blir problematiskt stor vid rimligt antal grafer och mätvärden. En viss fördröjning kan märkas av vid stor belastning. En fördjupning av det slutgiltiga resultatet hittas i delkapitel 3.1 Hämtning av data och delkapitel 3.2 Lagring av data.

Grafritingen fungerar men programmet skulle få ett större användningsområde ifall uppdateringen sker mer frekvent. Den är nu begränsad till att uppdateras en gång per sekund för att kunna garantera att uppackningen av paket inte får för stor fördröjning. Hur grafritning genomförs hittas i delkapitel 3.6 Grafritning. Då det har upptäckts att grafritningen tar mycket prestanda och den högsta prioriteten på projektet var att lagra datan korrekt. En anledning till den krävande uppdatering är att matplotlib inte är optimerat för snabb grafritning. Detta valdes ändå på grund av att det är ett välkänt och väldokumenterat bibliotek som till en början verkade kunna leverera önskad prestanda [3]. Det är dock bara under denna testning och utveckling av programmet som 224 kanaler används. Senare ska programmet användas med en applikation som använder ca 20 kanaler vilket innebär att uppdateringen sker oftare då bara drygt 9% av datan skickas och hanteras. Detta leder till att mer processorkraft kan användas till grafritningen, och därmed kan grafritningen uppnå mer frekvent uppdatering.

Det finns en begränsning i programmet då det inte går att ställa in trådprioritet i Python [4]. Detta leder till att det finns det risk att grafen prioriteras över sparning av data, vilket skulle leda till att om mycket plottas så kan sparningen halka efter. Det kan då skapas en ofrivillig kö med paket utanför programmet som tas omhand när det finns processortid över. Programmet kan hantera den ofrivilliga kön men det tar processorkraft som är till större användning i andra funktioner. Vidare utläggning finns i 4.2.2 Fördröjning. Programmet har testats på två olika datorer med olika prestanda. Det har då endast märkts av en liten skillnad trots stor skillnad i prestanda på de olika datorerna.

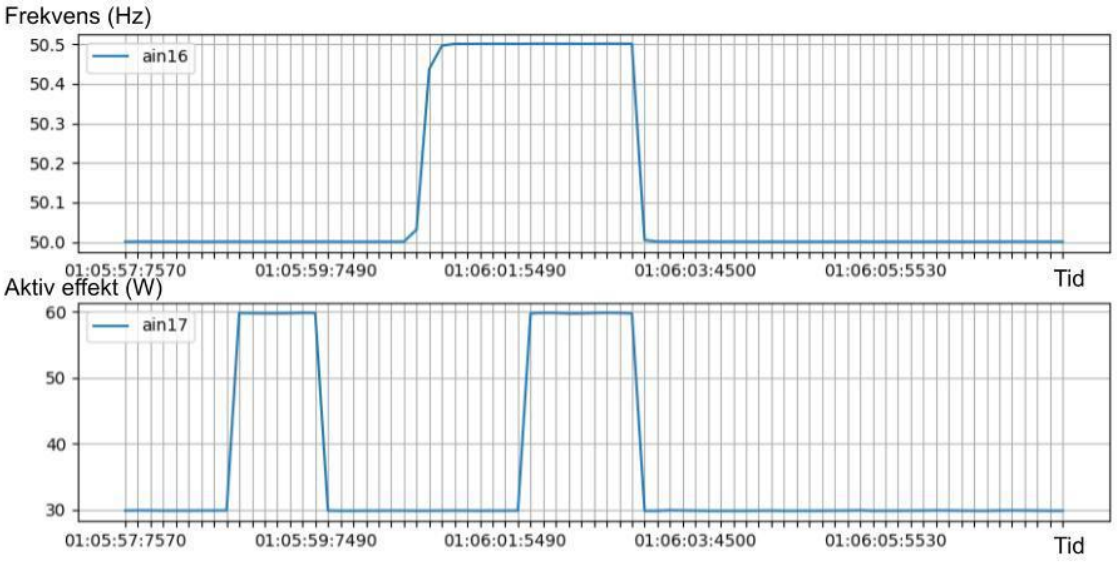
Programmet har förutom att testats på den modifierade fjärrterminalen även testats i simuleringar av dess tänkta användningsområde.

Systemet har testats på ett simulerat vattenkraftverk. Detta för att utföra en mer specifik testning med fokus på hur systemet fungerar i en situation som programmet är skapat för. Dessa tester överensstämmer mer med verkligheten än testerna på den modifierade fjärrterminalen. Till en början mättes 3 faser med hjälp av programmet som presenteras nedan.



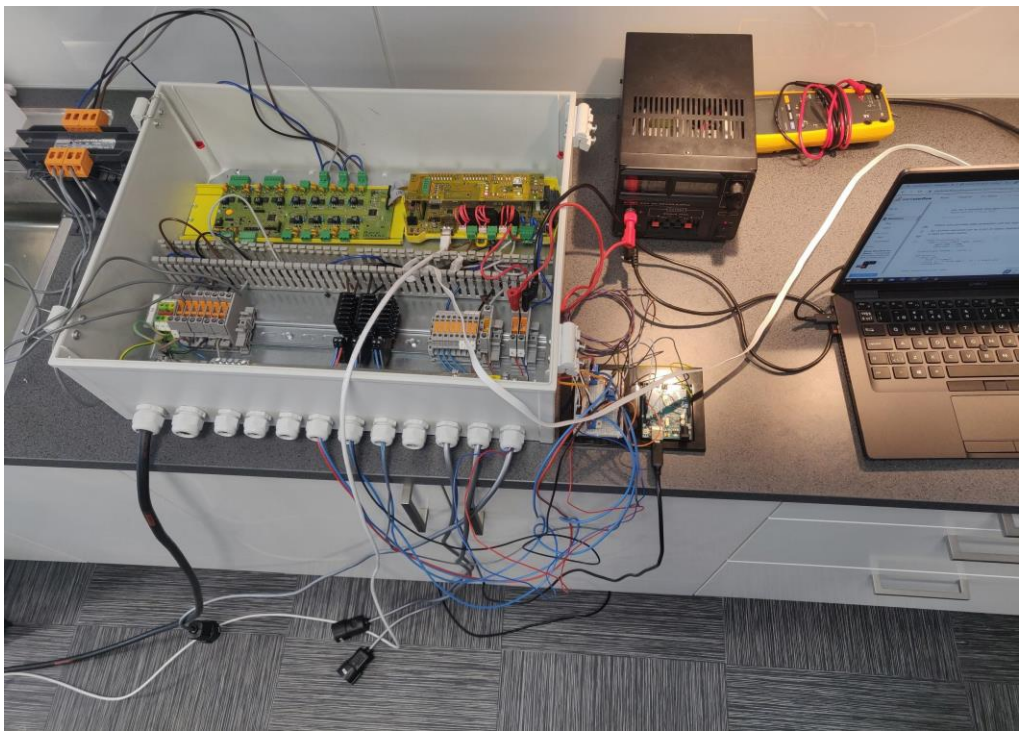
Figur 7: Simulering av tre faser som pendlar mellan -15V och +15V fasförskjutna 120°, Periodtid på 20 sek. Samplat med 5 Hz. 50 värden på spänning, mätning under 10 sek.

Hur systemet fungerar på en ändring av frekvens och aktiv effekt mättes även där dessa storheter loggades, visas i figur 8.



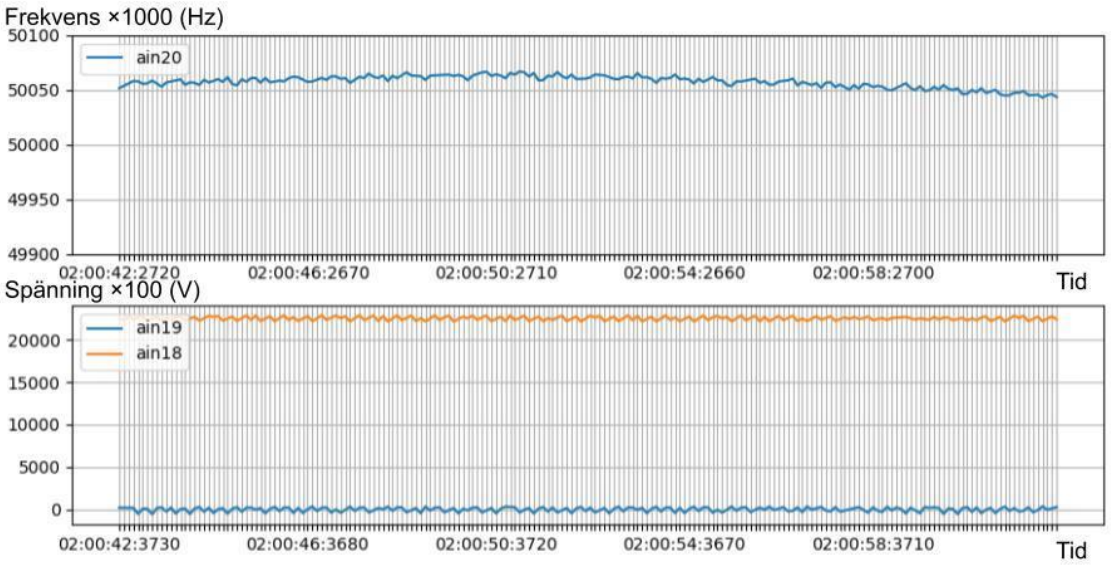
Figur 8: Simulering av stegökning på frekvens och aktiv effekt. Frekvensökning från 50 Hz till 50.5 Hz under 4 sek i grafen på kanal på ain16, övre grafen. I nedre grafen ökning av aktiv effekt i två steg vardera under 1 sek, ain17. 75 mätvärden, samplat med 7.5 Hz, alltså under 10 sek.

Systemet testades även vid anslutning till elnätet.



Figur 9: Testutrustning vid inkoppling på elnätet.

Här utfördes mätningar. Vid den här typen av datalagring fungerar realtidsfunktionen vid rimligt antal mätvärden. Operatören kan följa vad som händer genom att iaktta graferna. En grafritning av elnätet kan ses nedan i figur 10.



Figur 10: Inkoppling och loggning av elnätet. Frekvensen är multiplicerad med 1000 och spänningen är multiplicerad med 100 för att se större skillnad. Frekvens kring 50.05 Hz, övre grafen ain20. Fasspänning kring 230 V och fasdifferans kring 0 V i nedre grafen, ain18 fasspänning och ain19 fasdifferens. 200 mätvärden samplat med 10Hz, alltså mätning under 20 sek.

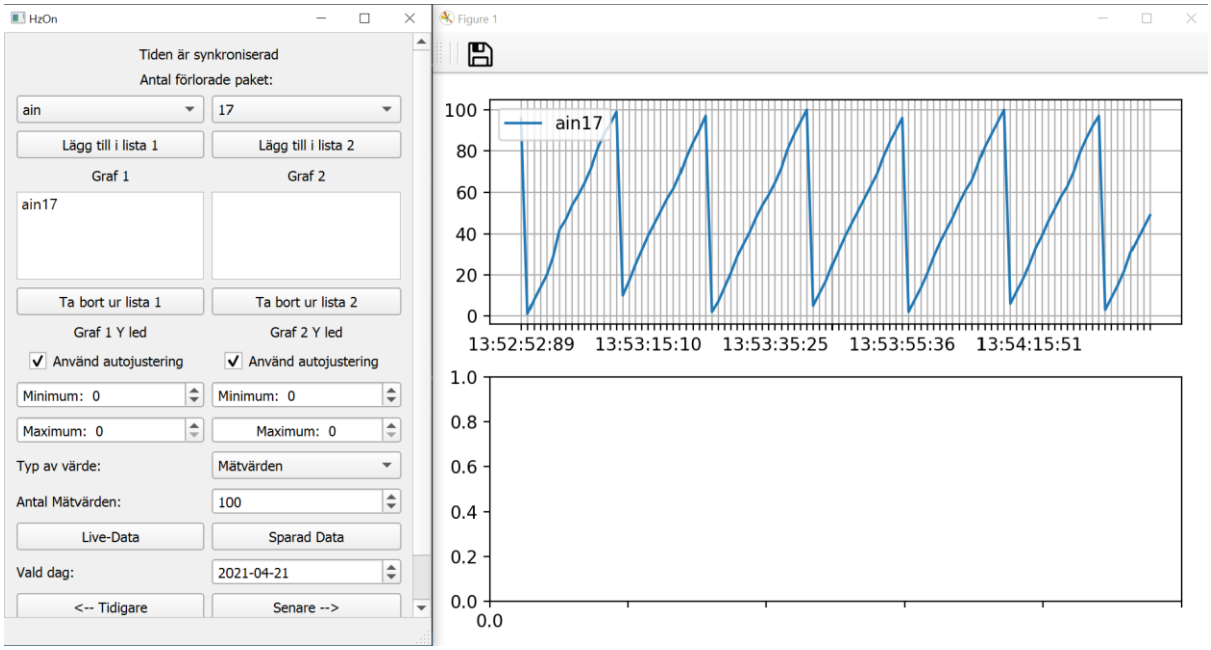
Det finns även en mindre variant, vilket är en förgrening av programmet. Detta mindre program kan enbart hämta och logga data, alltså är användargränssnittet och grafritningen borttagna. Programmet förgrenades för att kunna logga data med en Raspberry Pi ute vid kraftverk. Då dessa minidatorer har lägre prestanda blev det stora programmet för tungt och blev långsamt på en Pi. Detta är tänkt att användas konstant vid generatorer av förnybar energi och logga data kontinuerligt under längre tid. Skulle operatören senare vilja visualisera datan så kan filen skickas och öppnas på en annan PC med det större grundprogrammet. Detta delprogram har använts med godtyckliga resultat under projektets gång.

4.1 ANVÄNDARGRÄNSSNITT

Kanal väljs med hjälp av det två övre rullistorna. Vald kanal sparas sedan in i en listorna med "Lägg till i lista x" knapparna. I nedanstående bild har en av listorna ain17 som kanal och den andra listan är tom. De valda kanalerna i varje lista genereras sedan i en graf i ett annat fönster. Båda listorna kan ha upp till 5 separata kanaler, alltså kan totalt 10 grafer genereras samtidigt. Operatören kan sedan enkelt ta bort kanaler ur listorna genom att markera en kanal i listan och sedan trycka på knappen "Ta bort ur lista x" för att ta bort den valda kanalen.

Användaren väljer om autojustering ska användas för graf genom att i så fall bocka i rutan för vardera subplot. Väljer användaren att inte använda detta skall ett min- och maxvärde för y-axeln ställas in i rullistan. Då är y-axeln låst med de angivna värden som tillåts vara mellan -50 000 och +75 000. Grafens omfång kan dessutom ändras i x-led, antingen i antal mätvärden eller i antal sekunder.

"Live-Data" och "Sparad Data" knapparna bestämmer om graferna ska presentera datan som läses av i realtid eller data som har lagrats tidigare. Väljs "Sparad Data" väljer operatören vilken dag som ska visas med hjälp av datumlistan. Med "Tidigare" och "Senare" knapparna kan grafen förflyttas med valt antal steg i x-led. Det kompletta användargränsnittet med grafritningsfönstret presenteras här.



Figur 7: Visar användargränssnitt och graffönster.

5. SLUTSATS/DISKUSSION

Projektet har bestått av att utveckla ett loggningsprogram med möjlighet till grafritning för att lagra och visualisera data. Huvudmålet för projektet har varit att undersöka begränsningar för datahantering. Fjärrterminalen skickar data med som kortast 10 ms intervall. Denna datahastighet kan hanteras av programmet och lagras korrekt. Alltså har målet uppfyllts.

Grafritning av sparad och aktuell data fungerar, dock långsammare än önskat. För att begränsa prestandaanvändning, tillåts grafen uppdateras som mest en gång per sekund. Detta hade kunnat undvikas genom att rita grafer på ett annat sätt. Alltså borde grafritning ha utvärderats mer i början av projektet för att nå ett bättre resultat. Istället valdes grafritning från ett bibliotek som inte fungerar så effektivt som önskat i detta fall.

Projektgruppen är nöjd med resultaten och anser att arbetet har genomförts på ett lärorikt och målmedvetet sätt. Det har varit ett väl fungerande samarbete mellan mellan projektets medlemmar och företaget

För att blicka tillbaka på projektets frågeställning så hade det kunnat hanteras på ett annat sätt. Pakethantering borde ha felsökts grundligt tidigare för att inte behöva lägga ner den tid på ändringar som har gjorts. Ett stort problem var att följa standarden för JSON för att paketen ska kunna packas upp. Vilket innebar att meddelanden som kom in tillsammans behövde delas upp. Detta hade givit mer tid till att utföra tester och mätningar för att identifiera programmets begränsningar.

Användaren bör, för att uppnå högre prestanda, även göra ett undantag i Windows Defender för detta programs mapp. Defender vill annars genomsöka varje enskilt paket som kommer in och inte enbart det första när anslutning upprättas. Detta leder till att Defender tar en signifikant mängd processorkraft vid högre kommunikationshastighet. Som i sin tur leder till att hämtning och sparning av data får lägre prioritet och kan fördröjas.

5.1 Reflektion om utvecklingsarbetet

Projektet följde inte planeringsrapporten då det har lagts ner för stort fokus på detaljer i programmet jämfört med rapporten. Detta innebär att tänkta tidsramen har passerats.

5.2 Utvecklingspotential

Företaget har redan nu tankar på att ändra meddelanden till ett mer komprimerat, exempelvis i binärt, format för att erhålla en mindre krävande kommunikation. När meddelanden är mindre kommer det gå snabbare för programmet att läsa in och det skulle fungera smidigare.

Ytterligare en tråd på funktionen som har hand om grafritning skulle leda till att programmet fungerar smidigare. Detta då trådar delar upp tiden fördelaktigare men kräver lite mer processorkraft. Grafritningen kräver mycket om grafen ska uppdateras frekvent. Dock går inte denna funktion att tråda som vanligt då Python's uppbyggnad blockerar den för att köras parallellt. Detta för att en vanlig tråd inte hade varit trådsäker [5]. Detta problem ska kunna lösas med "multiprocessing" [6]. Vilket är ett annat bibliotek i python som fungerar på liknande sätt som threading-biblioteket. Om prestandan på grafritning i programmet anses vara för krävande är detta en teoretisk möjlig lösning.

Då grafritningen bara uppdateras max 1 gång per sekund och är relativt krävande finns det utvecklingspotential genom möjligheten att utföra grafritningen med något annat bibliotek.

REFERENSER

[1] IPC402x Manual, Mölndal, Sverige. [Online]. Tillgänglig: https://www.protrol.se/uploaded_files/IPC402x_UM_2102_sv1.pdf?v20210224164847, Hämtad: 2021-02-11

[2] QT Designer Manual, 2021. [Online]. Tillgänglig: https://doc.qt.io/qt-5/qtdesigner-manual.html, Hämtad: 2021-03-09

[3] Matplotlib User's Guide, 2021. [Online]. Tillgänglig: https://matplotlib.org/stable/users/index.html, Hämtad: 2021-03-15

[4] Thread-based parallelism Python 3.9.5 documentation, 2021. [Online]. Tillgänglig: https://docs.python.org/3/library/threading.html, Hämtad: 2021-04-28

[5] Matplotlib 3.4.2 documentation, 2021. [Online]. Tillgänglig: https://matplotlib.org/stable/faq/howto_faq.html#work-with-threads, Hämtad: 2021-04-16

[6] Process-based parallelism Python 3.9.5, 2021. [Online]. Tillgänglig: https://docs.python.org/3/library/multiprocessing.html, Hämtad: 2021-04-16

BILAGOR

```
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
import os
import sys
from datetime import date
import socket
import json
import datetime
import threading
import csv
from pathlib import Path
from PyQt5 import QtCore, QtGui, QtWidgets
from os.path import dirname

Nof_object=20
plot1_terms=[]
plot2_terms=[]
file_search=''
filename=str(date.today())
stored=False
lost_pack=0
x_line=0
time_values=[]
file_found=0
time_sync=[False]
freq=[1]
program_freq=0
freq_now=0
plot1ymin=0
plot1ymax=0
plot1Auto=True
plot2ymin=0
plot2ymax=0
plot2Auto=True
dataBinList=[]
dataBoutList=[]
dataAinList=[]
timeList=[]
timeSyncList=[]
allDataList=[]

class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(474, 642)
        MainWindow.setStyleSheet("selection-background-color: rgb(106, 106, 106);\n""")
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setAutoFillBackground(False)
        self.centralwidget.setStyleSheet("")
        self.centralwidget.setObjectName("centralwidget")
        self.gridLayout = QtWidgets.QGridLayout(self.centralwidget)
        self.gridLayout.setContentsMargins(0, 0, 0, 0)
        self.gridLayout.setSpacing(5)
        self.gridLayout.setObjectName("gridLayout")
        self.scrollArea = QtWidgets.QScrollArea(self.centralwidget)
        self.scrollArea.setWidgetResizable(True)
        self.scrollArea.setObjectName("scrollArea")
        self.scrollAreaWidgetContents_2 = QtWidgets.QWidget()
        self.scrollAreaWidgetContents_2.setGeometry(QtCore.QRect(0, 0, 472, 620))
        self.scrollAreaWidgetContents_2.setObjectName("scrollAreaWidgetContents_2")
```

```
self.gridLayout_2 = QtWidgets.QGridLayout(self.scrollAreaWidgetContents_2)
self.gridLayout_2.setObjectName("gridLayout_2")
self.spinBox_5 = QtWidgets.QSpinBox(self.scrollAreaWidgetContents_2)
self.spinBox_5.setObjectName("spinBox_5")
self.spinBox_5.setPrefix("Maximum:  ")
self.spinBox_5.setMaximum(75000)
self.spinBox_5.setAlignment(QtCore.Qt.AlignCenter)
self.gridLayout_2.addWidget(self.spinBox_5, 10, 1, 1, 1)
self.label_8 = QtWidgets.QLabel(self.scrollAreaWidgetContents_2)
self.label_8.setEnabled(True)
self.label_8.setAlignment(QtCore.Qt.AlignCenter)
self.label_8.setObjectName("label_8")
self.gridLayout_2.addWidget(self.label_8, 7, 0, 1, 1)
self.label_9 = QtWidgets.QLabel(self.scrollAreaWidgetContents_2)
self.label_9.setAlignment(QtCore.Qt.AlignCenter)
self.label_9.setObjectName("label_9")
self.gridLayout_2.addWidget(self.label_9, 7, 1, 1, 1)
self.spinBox_2 = QtWidgets.QSpinBox(self.scrollAreaWidgetContents_2)
self.spinBox_2.setAutoFillBackground(False)
self.spinBox_2.setObjectName("spinBox_2")
self.spinBox_2.setPrefix("Minimum:  ")
self.spinBox_2.setMinimum(-50000)
self.spinBox_2.setMaximum(750000)
self.gridLayout_2.addWidget(self.spinBox_2, 9, 0, 1, 1)
self.spinBox_3 = QtWidgets.QSpinBox(self.scrollAreaWidgetContents_2)
self.spinBox_3.setObjectName("spinBox_3")
self.spinBox_3.setPrefix("Minimum:  ")
self.spinBox_3.setMinimum(-50000)
self.spinBox_3.setMaximum(750000)
self.gridLayout_2.addWidget(self.spinBox_3, 9, 1, 1, 1)
self.label_5 = QtWidgets.QLabel(self.scrollAreaWidgetContents_2)
self.label_5.setObjectName("label_5")
self.gridLayout_2.addWidget(self.label_5, 11, 0, 1, 1)
self.comboBox_3 = QtWidgets.QComboBox(self.scrollAreaWidgetContents_2)
self.comboBox_3.setObjectName("comboBox_3")
self.comboBox_3.addItem("")
self.comboBox_3.addItem("")
self.gridLayout_2.addWidget(self.comboBox_3, 11, 1, 1, 1)
self.pushButton_3 = QtWidgets.QPushButton(self.scrollAreaWidgetContents_2)
self.pushButton_3.setEnabled(True)
self.pushButton_3.setContextMenuPolicy(QtCore.Qt.NoContextMenu)
self.pushButton_3.setStyleSheet("")
self.pushButton_3.setCheckable(False)
self.pushButton_3.setAutoDefault(False)
self.pushButton_3.setDefault(False)
self.pushButton_3.setFlat(False)
self.pushButton_3.setObjectName("pushButton_3")
self.gridLayout_2.addWidget(self.pushButton_3, 16, 0, 1, 1)
self.buttonLiveData = QtWidgets.QPushButton(self.scrollAreaWidgetContents_2)
self.buttonLiveData.setCheckable(True)
self.buttonLiveData.setObjectName("buttonLiveData")
self.gridLayout_2.addWidget(self.buttonLiveData, 13, 0, 1, 1)
self.buttonSavedData = QtWidgets.QPushButton(self.scrollAreaWidgetContents_2)
self.buttonSavedData.setCheckable(True)
self.buttonSavedData.setObjectName("buttonSavedData")
self.gridLayout_2.addWidget(self.buttonSavedData, 13, 1, 1, 1)
self.label_3 = QtWidgets.QLabel(self.scrollAreaWidgetContents_2)
self.label_3.setObjectName("label_3")
self.gridLayout_2.addWidget(self.label_3, 14, 0, 1, 1)
self.label1 = QtWidgets.QLabel(self.scrollAreaWidgetContents_2)
self.label1.setObjectName("label1")
self.gridLayout_2.addWidget(self.label1, 12, 0, 1, 1)
self.dateEdit = QtWidgets.QDateEdit(self.scrollAreaWidgetContents_2)
self.dateEdit.setDate(QtCore.QDate.currentDate())
```

```
self.dateEdit.setObjectName("dateEdit")
self.gridLayout_2.addWidget(self.dateEdit, 14, 1, 1, 1)
self.spinBox1 = QtWidgets.QSpinBox(self.scrollAreaWidgetContents_2)
self.spinBox1.setMinimum(1)
self.spinBox1.setMaximum(500)
self.spinBox1.setProperty("value", 20)
self.spinBox1.setObjectName("spinBox1")
self.gridLayout_2.addWidget(self.spinBox1, 12, 1, 1, 1)
self.pushButton_4 = QtWidgets.QPushButton(self.scrollAreaWidgetContents_2)
self.pushButton_4.setObjectName("pushButton_4")
self.gridLayout_2.addWidget(self.pushButton_4, 16, 1, 1, 1)
self.label_4 = QtWidgets.QLabel(self.scrollAreaWidgetContents_2)
self.label_4.setObjectName("label_4")
self.gridLayout_2.addWidget(self.label_4, 17, 0, 1, 1)
self.label_7 = QtWidgets.QLabel(self.scrollAreaWidgetContents_2)
self.label_7.setAlignment(QtCore.Qt.AlignCenter)
self.label_7.setObjectName("label_7")
self.gridLayout_2.addWidget(self.label_7, 1, 0, 1, 2)
self.buttonChooseGraph = QtWidgets.QPushButton(self.scrollAreaWidgetContents_2)
self.buttonChooseGraph.setObjectName("buttonChooseGraph")
self.gridLayout_2.addWidget(self.buttonChooseGraph, 3, 0, 1, 1)
self.listWidget = QtWidgets.QListWidget(self.scrollAreaWidgetContents_2)
self.listWidget.setMaximumSize(QtCore.QSize(16777215, 200))
self.listWidget.setObjectName("listWidget")
self.gridLayout_2.addWidget(self.listWidget, 5, 0, 1, 1)
self.listWidget_2 = QtWidgets.QListWidget(self.scrollAreaWidgetContents_2)
self.listWidget_2.setMaximumSize(QtCore.QSize(16777215, 200))
self.listWidget_2.setObjectName("listWidget_2")
self.gridLayout_2.addWidget(self.listWidget_2, 5, 1, 1, 1)
self.pushButton = QtWidgets.QPushButton(self.scrollAreaWidgetContents_2)
self.pushButton.setObjectName("pushButton")
self.gridLayout_2.addWidget(self.pushButton, 6, 0, 1, 1)
self.pushButton_2 = QtWidgets.QPushButton(self.scrollAreaWidgetContents_2)
self.pushButton_2.setObjectName("pushButton_2")
self.gridLayout_2.addWidget(self.pushButton_2, 6, 1, 1, 1)
self.label_6 = QtWidgets.QLabel(self.scrollAreaWidgetContents_2)
self.label_6.setAlignment(QtCore.Qt.AlignCenter)
self.label_6.setObjectName("label_6")
self.gridLayout_2.addWidget(self.label_6, 0, 0, 1, 2)
self.comboBox = QtWidgets.QComboBox(self.scrollAreaWidgetContents_2)
self.comboBox.setStyleSheet("")
self.comboBox.setObjectName("comboBox")
self.comboBox.addItem("")
self.comboBox.addItem("")
self.comboBox.addItem("")
self.gridLayout_2.addWidget(self.comboBox, 2, 0, 1, 1)
self.comboBox_2 = QtWidgets.QComboBox(self.scrollAreaWidgetContents_2)
self.comboBox_2.setObjectName("comboBox_2")
self.comboBox_2.addItem("")
self.comboBox_2.addItem("")
self.comboBox_2.addItem("")
self.comboBox_2.addItem("")
self.comboBox_2.addItem("")
self.comboBox_2.addItem("")
self.comboBox_2.addItem("")
self.comboBox_2.addItem("")
self.comboBox_2.addItem("")
self.comboBox_2.addItem("")
self.comboBox_2.addItem("")
self.comboBox_2.addItem("")
self.comboBox_2.addItem("")
self.comboBox_2.addItem("")
self.comboBox_2.addItem("")
self.comboBox_2.addItem("")
```

```
        self.comboBox_2.addItem("")
        self.comboBox_2.addItem("")
        self.comboBox_2.addItem("")
        self.comboBox_2.addItem("")
        self.comboBox_2.addItem("")
        self.gridLayout_2.addWidget(self.comboBox_2, 2, 1, 1, 1)
        self.buttonChooseGraph_2 = QtWidgets.QPushButton(self.scrollAreaWidgetContents_2)
        self.buttonChooseGraph_2.setObjectName("buttonChooseGraph_2")
        self.gridLayout_2.addWidget(self.buttonChooseGraph_2, 3, 1, 1, 1)
        self.label = QtWidgets.QLabel(self.scrollAreaWidgetContents_2)
        self.label.setMaximumSize(QtCore.QSize(16777215, 20))
        self.label.setAlignment(QtCore.Qt.AlignCenter)
        self.label.setObjectName("label")
        self.gridLayout_2.addWidget(self.label, 4, 0, 1, 1)
        self.label_2 = QtWidgets.QLabel(self.scrollAreaWidgetContents_2)
        self.label_2.setAlignment(QtCore.Qt.AlignCenter)
        self.label_2.setObjectName("label_2")
        self.gridLayout_2.addWidget(self.label_2, 4, 1, 1, 1)
        self.spinBox_4 = QtWidgets.QSpinBox(self.scrollAreaWidgetContents_2)
        self.spinBox_4.setObjectName("spinBox_4")
        self.spinBox_4.setPrefix("Maximum:  ")
        self.spinBox_4.setMaximum(750000)
        self.gridLayout_2.addWidget(self.spinBox_4, 10, 0, 1, 1)
        self.spinBox = QtWidgets.QSpinBox(self.scrollAreaWidgetContents_2)
        self.spinBox.setMaximum(2000)
        self.spinBox.setProperty("value", 5)
        self.spinBox.setObjectName("spinBox")
        self.gridLayout_2.addWidget(self.spinBox, 17, 1, 1, 1)
        self.checkBox = QtWidgets.QCheckBox(self.scrollAreaWidgetContents_2)
        self.checkBox.setObjectName("checkBox")
        self.checkBox.setChecked(True)
        self.gridLayout_2.addWidget(self.checkBox, 8, 0, 1, 1, QtCore.Qt.AlignHCenter)
        self.checkBox_2 = QtWidgets.QCheckBox(self.scrollAreaWidgetContents_2)
        self.checkBox_2.setObjectName("checkBox_2")
        self.checkBox_2.setChecked(True)
        self.gridLayout_2.addWidget(self.checkBox_2, 8, 1, 1, 1, QtCore.Qt.AlignHCenter)
        self.scrollArea.setWidget(self.scrollAreaWidgetContents_2)
        self.gridLayout.addWidget(self.scrollArea, 0, 0, 1, 1)
        MainWindow.setCentralWidget(self.centralwidget)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)
        self.actionNew = QtWidgets.QAction(MainWindow)
        self.actionNew.setObjectName("actionNew")
        self.actionSave = QtWidgets.QAction(MainWindow)
        self.actionSave.setObjectName("actionSave")
        self.actionCopy = QtWidgets.QAction(MainWindow)
        self.actionCopy.setObjectName("actionCopy")
        self.actionPaste = QtWidgets.QAction(MainWindow)
        self.actionPaste.setObjectName("actionPaste")

        self.buttonChooseGraph.clicked.connect(self.addGraph1)          #Funktioner för att
lägga till i listor
        self.buttonChooseGraph_2.clicked.connect(self.addGraph2)

        self.pushButton.clicked.connect(self.remove1)                    #tar bort ur
listwidget1
        self.pushButton_2.clicked.connect(self.remove2)
        self.spinBox1.valueChanged.connect(self.valuechange)
        self.buttonLiveData.clicked.connect(self.livedata)
        self.buttonSavedData.clicked.connect(self.storeddata)
        self.pushButton_3.clicked.connect(self.scroll_left)
        self.pushButton_4.clicked.connect(self.scroll_right)
        self.buttonLiveData.clicked.connect(self.Timesynced)
```

```
        self.buttonSavedData.clicked.connect(self.Timesynced)
        self.buttonChooseGraph.clicked.connect(self.Timesynced)
        self.buttonChooseGraph_2.clicked.connect(self.Timesynced)
        self.pushButton_3.clicked.connect(self.Timesynced)
        self.pushButton_4.clicked.connect(self.Timesynced)
        self.comboBox_3.currentTextChanged.connect(self.valuechange)
        self.checkBox.stateChanged.connect(self.plot1_y)
        self.checkBox_2.stateChanged.connect(self.plot2_y)




        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "HzOn"))
        self.label_8.setText(_translate("MainWindow", "Graf 1 Y led"))
        self.label_9.setText(_translate("MainWindow", "Graf 2 Y led"))
        self.label_5.setText(_translate("MainWindow", "Typ av värde:"))
        self.comboBox_3.setItemText(0, _translate("MainWindow", "Mätvärden"))
        self.comboBox_3.setItemText(1, _translate("MainWindow", "Sekunder"))
        self.pushButton_3.setText(_translate("MainWindow", "<-- Tidigare"))
        self.buttonLiveData.setText(_translate("MainWindow", "Live-Data"))
        self.buttonSavedData.setText(_translate("MainWindow", "Sparad Data"))
        self.label_3.setText(_translate("MainWindow", "Vald dag:"))
        self.label1.setText(_translate("MainWindow", "Antal Mätvärden:"))
        self.pushButton_4.setText(_translate("MainWindow", "Senare -->"))
        self.label_4.setText(_translate("MainWindow", "Antal steg i mätvärden:"))
        self.label_7.setText(_translate("MainWindow", "Antal förlorade paket: "))
        self.buttonChooseGraph.setText(_translate("MainWindow", "Lägg till i lista 1"))
        self.pushButton.setText(_translate("MainWindow", "Ta bort ur lista 1"))
        self.pushButton_2.setText(_translate("MainWindow", "Ta bort ur lista 2"))
        self.label_6.setText(_translate("MainWindow", "Tiden är synkroniserad"))
        self.comboBox.setItemText(0, _translate("MainWindow", "ain"))
        self.comboBox.setItemText(1, _translate("MainWindow", "bin"))
        self.comboBox.setItemText(2, _translate("MainWindow", "but"))
        self.comboBox_2.setItemText(0, _translate("MainWindow", "0"))
        self.comboBox_2.setItemText(1, _translate("MainWindow", "1"))
        self.comboBox_2.setItemText(2, _translate("MainWindow", "2"))
        self.comboBox_2.setItemText(3, _translate("MainWindow", "3"))
        self.comboBox_2.setItemText(4, _translate("MainWindow", "4"))
        self.comboBox_2.setItemText(5, _translate("MainWindow", "5"))
        self.comboBox_2.setItemText(6, _translate("MainWindow", "6"))
        self.comboBox_2.setItemText(7, _translate("MainWindow", "7"))
        self.comboBox_2.setItemText(8, _translate("MainWindow", "8"))
        self.comboBox_2.setItemText(9, _translate("MainWindow", "9"))
        self.comboBox_2.setItemText(10, _translate("MainWindow", "10"))
        self.comboBox_2.setItemText(11, _translate("MainWindow", "11"))
        self.comboBox_2.setItemText(12, _translate("MainWindow", "12"))
        self.comboBox_2.setItemText(13, _translate("MainWindow", "13"))
        self.comboBox_2.setItemText(14, _translate("MainWindow", "14"))
        self.comboBox_2.setItemText(15, _translate("MainWindow", "15"))
        self.comboBox_2.setItemText(16, _translate("MainWindow", "16"))
        self.comboBox_2.setItemText(17, _translate("MainWindow", "17"))
        self.comboBox_2.setItemText(18, _translate("MainWindow", "18"))
        self.comboBox_2.setItemText(19, _translate("MainWindow", "19"))
        self.comboBox_2.setItemText(20, _translate("MainWindow", "20"))
        self.buttonChooseGraph_2.setText(_translate("MainWindow", "Lägg till i lista 2"))
        self.label.setText(_translate("MainWindow", "Graf 1"))
        self.label_2.setText(_translate("MainWindow", "Graf 2"))
        self.checkBox.setText(_translate("MainWindow", "Använd autojustering"))
        self.checkBox_2.setText(_translate("MainWindow", "Använd autojustering"))
```

```
        self.actionNew.setText(_translate("MainWindow", "New"))
        self.actionSave.setText(_translate("MainWindow", "Save"))
        self.actionSave.setStatusTip(_translate("MainWindow", "Saves a file"))
        self.actionSave.setShortcut(_translate("MainWindow", "Ctrl+S"))
        self.actionCopy.setText(_translate("MainWindow", "Copy"))
        self.actionPaste.setText(_translate("MainWindow", "Paste"))

    def addGraph1(self):
        global plot1_terms
        x = self.comboBox.currentText()
        y = self.comboBox_2.currentText()

        self.listWidget.addItem(x + y)

        plot1_terms =  [str(self.listWidget.item(i).text()) for i in
range(self.listWidget.count())]


    def addGraph2(self):
        global plot2_terms
        z = self.comboBox.currentText()
        w = self.comboBox_2.currentText()

        self.listWidget_2.addItem(z + w)

        plot2_terms =  [str(self.listWidget_2.item(i).text()) for i in
range(self.listWidget_2.count())]


    def remove1(self):
        global plot1_terms
        self.listWidget.takeItem(self.listWidget.currentRow())
        plot1_terms =  [str(self.listWidget.item(i).text()) for i in
range(self.listWidget.count())]

    def remove2(self):
        global plot2_terms
        self.listWidget_2.takeItem(self.listWidget_2.currentRow())
        plot2_terms =  [str(self.listWidget_2.item(i).text()) for i in
range(self.listWidget_2.count())]

    def valuechange(self):
        global Nof_object, freq_now
        if stored:
            freq_now=freq[x_line]
        else:
            freq_now=program_freq
        if self.comboBox_3.currentText() == "Sekunder":
            Nof_object = int((self.spinBox1.value())*(freq_now))
            self.label1.setText("Antal Sekunder:")
            self.label_4.setText("Antal steg i sekunder:")
        else:
            Nof_object = int(self.spinBox1.value())
            self.label1.setText("Antal Mätvärden:")
            self.label_4.setText("Antal steg i punkter:")

    def livedata(self):
        global filename, stored, freq_now
        filename=str(date.today())
        freq_now=program_freq
        stored=False

    def storeddata(self):
        global filename, stored, freq_now
```

```
    chosenDate=self.dateEdit.date()
    filename=chosenDate.toString(QtCore.Qt.ISODate)
    freq_now=freq[x_line]
    stored=True

def scroll_left(self):
    global x_line
    if self.comboBox_3.currentText() == "Sekunder":
        if int(x_line-(self.spinBox.value()*(freq_now)))<=0:
            x_line=0
        else:
            x_line=x_line-int(self.spinBox.value()*(freq_now))
    else:
        if x_line-self.spinBox.value()<=0:
            x_line=0
        else:
            x_line=x_line-self.spinBox.value()

def scroll_right(self):
    global x_line
    if self.comboBox_3.currentText() == "Sekunder":
        if len(time_values)-1 <= (x_line+(self.spinBox.value()*(freq_now))+Nof_object):
            x_line= len(time_values)-Nof_object-1
        else:
            x_line=x_line+int(self.spinBox.value()*(freq_now))
    else:
        if len(time_values)-1 <= x_line+self.spinBox.value()+Nof_object:
            x_line= len(time_values)-Nof_object-1
        else:
            x_line=x_line+self.spinBox.value()

def Timesynced(self):
    if stored:
        if time_sync[x_line]:
            self.label_6.setText('Tiden är synkroniserad')
        else:
            self.label_6.setText('Tiden är inte synkroniserad')
    else:
        for rows in time_sync:
            if rows==True:
                self.label_6.setText('Tiden är synkroniserad')
            else:
                self.label_6.setText('Tiden är inte synkroniserad')


def plot1_y(self):
    global plot1ymin, plot1ymax, plot1Auto
    plot1ymin=self.spinBox_2.value()
    plot1ymax=self.spinBox_4.value()
    if self.checkBox.isChecked():
        plot1Auto=True
    else:
        plot1Auto=False

def plot2_y(self):
    global plot2ymin, plot2ymax, plot2Auto
    plot2ymin=self.spinBox_3.value()
    plot2ymax=self.spinBox_5.value()
    if self.checkBox_2.isChecked():
        plot2Auto=True
    else:
        plot2Auto=False
```

```
plt.style.use('fast')                                                                                    #
toolbar = plt.get_current_fig_manager().toolbar                                                          #
Bestämmer hur plotfönstret ser ut
unwanted_buttons = ['Home', 'Back', 'Forward', 'Pan', 'Zoom', 'Customize', 'Subplots']                   #
for x in toolbar.actions():                                                                              #
    if x.text() in unwanted_buttons:                                                                     #
        toolbar.removeAction(x)                                                                          #


fig = plt.figure(1)
ax1 = fig.add_subplot(2,1,1)
ax2 = fig.add_subplot(2,1,2)
plt.tight_layout()

# animate funktionen hämtar data från csv-fil och sparar till variabel för att plotta.
def animate(i):
    global time_values, time_sync, freq
    global file_found
    plot1_1=[]
    plot1_2=[]
    plot1_3=[]
    plot1_4=[]
    plot1_5=[]
    plot2_1=[]
    plot2_2=[]
    plot2_3=[]
    plot2_4=[]
    plot2_5=[]
    if filename!=str(date.today()):
        read_file = Path(dirname(__file__)+"/"+filename+".csv")
        if read_file.exists():
            if file_found>0:
                print("Filen hittades!")
                file_found=0


            data = pd.read_csv(filename+'.csv', comment='#', low_memory=False)
            time_values = data['Time']
            time_sync = data['timesync']
            freq = data['freq']
        else:
            if file_found==0:
                print("Filen hittades inte")
                file_found+=1
    else:
        time_values=timeList
        time_sync=timeSyncList

    ax1.cla()
    nOfGraphs1=len(plot1_terms)
    if nOfGraphs1>=1:
        if plot1Auto:
            ax1.autoscale()
        else:
            ax1.set_ylim([plot1ymin,plot1ymax])
        if filename!=str(date.today()):
            plot1_1 = data[plot1_terms[0]]
        else:
            if plot1_terms[0][:3]=="ain":
                for ix in range(0,len(dataAinList)):
                    plot1_1.append(dataAinList[ix][int(plot1_terms[0][3:])])
            if plot1_terms[0][:3]=="bin":
```

```
                for ix in range(0,len(dataBinList)):
                    plot1_1.append(dataBinList[ix][int(plot1_terms[0][3:])])
            if plot1_terms[0][:4]=="bout":
                for ix in range(0,len(dataBoutList)):
                    plot1_1.append(dataBinList[ix][int(plot1_terms[0][4:])])
        if stored:
            ax1.plot(time_values[x_line:Nof_object+x_line], plot1_1[x_line:Nof_object+x_line],
label=plot1_terms[0])
        else:
            time_values=time_values[Nof_object:]
            plot1_1=plot1_1[Nof_object:]
            ax1.plot(time_values[-Nof_object:], plot1_1[-Nof_object:], label=plot1_terms[0])
        ax1.legend(loc='upper left')
        ax1.grid()
    if nOfGraphs1>=2:
        if filename!=str(date.today()):
            plot1_2 = data[plot1_terms[0]]
        else:
            if plot1_terms[1][:3]=="ain":
                for ix in range(0,len(dataAinList)):
                    plot1_2.append(dataAinList[ix][int(plot1_terms[1][3:])])
            if plot1_terms[1][:3]=="bin":
                for ix in range(0,len(dataBinList)):
                    plot1_2.append(dataBinList[ix][int(plot1_terms[1][3:])])
            if plot1_terms[1][:4]=="bout":
                for ix in range(0,len(dataBoutList)):
                    plot1_2.append(dataBinList[ix][int(plot1_terms[1][4:])])
            if stored:
                ax1.plot(time_values[x_line:Nof_object+x_line],
plot1_2[x_line:Nof_object+x_line], label=plot1_terms[1])
            else:
                ax1.plot(time_values[-Nof_object:], plot1_2[-Nof_object:],
label=plot1_terms[1])
            ax1.legend(loc='upper left')
    if nOfGraphs1>=3:
        if filename!=str(date.today()):
            plot1_3 = data[plot1_terms[0]]
        else:
            if plot1_terms[2][:3]=="ain":
                for ix in range(0,len(dataAinList)):
                    plot1_3.append(dataAinList[ix][int(plot1_terms[2][3:])])
            if plot1_terms[2][:3]=="bin":
                for ix in range(0,len(dataBinList)):
                    plot1_3.append(dataBinList[ix][int(plot1_terms[2][3:])])
            if plot1_terms[2][:4]=="bout":
                for ix in range(0,len(dataBoutList)):
                    plot1_3.append(dataBinList[ix][int(plot1_terms[2][4:])])
        if stored:
            ax1.plot(time_values[x_line:Nof_object+x_line], plot1_3[x_line:Nof_object+x_line],
label=plot1_terms[2])
        else:
            ax1.plot(time_values[-Nof_object:], plot1_3[-Nof_object:], label=plot1_terms[2])
        ax1.legend(loc='upper left')
    if nOfGraphs1>=4:
        if filename!=str(date.today()):
            plot1_4 = data[plot1_terms[0]]
        else:
            if plot1_terms[3][:3]=="ain":
                for ix in range(0,len(dataAinList)):
                    plot1_4.append(dataAinList[ix][int(plot1_terms[3][3:])])
            if plot1_terms[3][:3]=="bin":
                for ix in range(0,len(dataBinList)):
                    plot1_4.append(dataBinList[ix][int(plot1_terms[3][3:])])
            if plot1_terms[3][:4]=="bout":
```

```
                for ix in range(0,len(dataBoutList)):
                    plot1_4.append(dataBinList[ix][int(plot1_terms[3][4:])])
        if stored:
            ax1.plot(time_values[x_line:Nof_object+x_line], plot1_4[x_line:Nof_object+x_line],
label=plot1_terms[3])
        else:
            ax1.plot(time_values[-Nof_object:], plot1_4[-Nof_object:], label=plot1_terms[3])
        ax1.legend(loc='upper left')
    if nOfGraphs1>=5:
        if filename!=str(date.today()):
            plot1_2 = data[plot1_terms[0]]
        else:
            if plot1_terms[4][:3]=="ain":
                for ix in range(0,len(dataAinList)):
                    plot1_5.append(dataAinList[ix][int(plot1_terms[4][3:])])
            if plot1_terms[4][:3]=="bin":
                for ix in range(0,len(dataBinList)):
                    plot1_5.append(dataBinList[ix][int(plot1_terms[4][3:])])
            if plot1_terms[4][:4]=="bout":
                for ix in range(0,len(dataBoutList)):
                    plot1_5.append(dataBinList[ix][int(plot1_terms[4][4:])])
        if stored:
            ax1.plot(time_values[x_line:Nof_object+x_line], plot1_5[x_line:Nof_object+x_line],
label=plot1_terms[4])
        else:
            ax1.plot(time_values[-Nof_object:], plot1_5[-Nof_object:], label=plot1_terms[4])
        ax1.legend(loc='upper left')



    ax2.cla()
    nOfGraphs2=len(plot2_terms)
    if nOfGraphs2>=1:
        if plot2Auto:
            ax2.autoscale()
        else:
            ax2.set_ylim([plot2ymin,plot2ymax])
        if filename!=str(date.today()):
            plot2_1 = data[plot2_terms[0]]
        else:
            if plot2_terms[0][:3]=="ain":
                for ix in range(0,len(dataAinList)):
                    plot2_1.append(dataAinList[ix][int(plot2_terms[0][3:])])
            if plot2_terms[0][:3]=="bin":
                for ix in range(0,len(dataBinList)):
                    plot2_1.append(dataBinList[ix][int(plot2_terms[0][3:])])
            if plot2_terms[0][:4]=="bout":
                for ix in range(0,len(dataBoutList)):
                    plot2_1.append(dataBinList[ix][int(plot2_terms[0][4:])])
        if stored:
            ax2.plot(time_values[x_line:Nof_object+x_line], plot2_1[x_line:Nof_object+x_line],
label=plot2_terms[0])
        else:
            ax2.plot(time_values[-Nof_object:], plot2_1[-Nof_object:], label=plot2_terms[0])
        ax2.legend(loc='upper left')
        ax2.grid()
    if nOfGraphs2>=2:
        if plot2Auto:
            ax2.autoscale()
        else:
            ax2.set_ylim([plot2ymin,plot2ymax])
        if filename!=str(date.today()):
            plot2_2 = data[plot2_terms[0]]
        else:
```

```
            if plot2_terms[1][:3]=="ain":
                for ix in range(0,len(dataAinList)):
                    plot2_2.append(dataAinList[ix][int(plot2_terms[1][3:])])
            if plot2_terms[1][:3]=="bin":
                for ix in range(0,len(dataBinList)):
                    plot2_2.append(dataBinList[ix][int(plot2_terms[1][3:])])
            if plot2_terms[1][:4]=="bout":
                for ix in range(0,len(dataBoutList)):
                    plot2_2.append(dataBinList[ix][int(plot2_terms[1][4:])])
        if stored:
            ax2.plot(time_values[x_line:Nof_object+x_line], plot2_2[x_line:Nof_object+x_line],
label=plot2_terms[1])
        else:
            ax2.plot(time_values[-Nof_object:], plot2_2[-Nof_object:], label=plot2_terms[1])
        ax2.legend(loc='upper left')
    if nOfGraphs2>=3:
        if plot2Auto:
            ax2.autoscale()
        else:
            ax2.set_ylim([plot2ymin,plot2ymax])
        if filename!=str(date.today()):
            plot2_3 = data[plot2_terms[0]]
        else:
            if plot2_terms[2][:3]=="ain":
                for ix in range(0,len(dataAinList)):
                    plot2_3.append(dataAinList[ix][int(plot2_terms[2][3:])])
            if plot2_terms[2][:3]=="bin":
                for ix in range(0,len(dataBinList)):
                    plot2_3.append(dataBinList[ix][int(plot2_terms[2][3:])])
            if plot2_terms[2][:4]=="bout":
                for ix in range(0,len(dataBoutList)):
                    plot2_3.append(dataBinList[ix][int(plot2_terms[2][4:])])
        if stored:
            ax2.plot(time_values[x_line:Nof_object+x_line], plot2_3[x_line:Nof_object+x_line],
label=plot2_terms[2])
        else:
            ax2.plot(time_values[-Nof_object:], plot2_3[-Nof_object:], label=plot2_terms[2])
        ax2.legend(loc='upper left')
    if nOfGraphs2>=4:
        if plot2Auto:
            ax2.autoscale()
        else:
            ax2.set_ylim([plot2ymin,plot2ymax])
        if filename!=str(date.today()):
            plot2_4 = data[plot2_terms[0]]
        else:
            if plot2_terms[3][:3]=="ain":
                for ix in range(0,len(dataAinList)):
                    plot2_4.append(dataAinList[ix][int(plot2_terms[3][3:])])
            if plot2_terms[3][:3]=="bin":
                for ix in range(0,len(dataBinList)):
                    plot2_4.append(dataBinList[ix][int(plot2_terms[3][3:])])
            if plot2_terms[3][:4]=="bout":
                for ix in range(0,len(dataBoutList)):
                    plot2_4.append(dataBinList[ix][int(plot2_terms[3][4:])])
        if stored:
            ax2.plot(time_values[x_line:Nof_object+x_line], plot2_4[x_line:Nof_object+x_line],
label=plot2_terms[3])
        else:
            ax2.plot(time_values[-Nof_object:], plot2_4[-Nof_object:], label=plot2_terms[3])
        ax2.legend(loc='upper left')
    if nOfGraphs2>=5:
        if plot2Auto:
            ax2.autoscale()
```

```
        else:
            ax2.set_ylim([plot2ymin,plot2ymax])
        if filename!=str(date.today()):
            plot2_5 = data[plot2_terms[0]]
        else:
            if plot2_terms[4][:3]=="ain":
                for ix in range(0,len(dataAinList)):
                    plot2_5.append(dataAinList[ix][int(plot2_terms[4][3:])])
            if plot2_terms[4][:3]=="bin":
                for ix in range(0,len(dataBinList)):
                    plot2_5.append(dataBinList[ix][int(plot2_terms[4][3:])])
            if plot2_terms[4][:4]=="bout":
                for ix in range(0,len(dataBoutList)):
                    plot2_5.append(dataBinList[ix][int(plot2_terms[4][4:])])
        if stored:
            ax2.plot(time_values[x_line:Nof_object+x_line], plot2_5[x_line:Nof_object+x_line],
label=plot2_terms[4])
        else:
            ax2.plot(time_values[-Nof_object:], plot2_5[-Nof_object:], label=plot2_terms[4])
        ax2.legend(loc='upper left')

    n = Nof_object/5  # Keeps every 5th label
    [l.set_visible(False) for (i,l) in enumerate(ax1.xaxis.get_ticklabels()) if i % n != 0]
    [l.set_visible(False) for (i,l) in enumerate(ax2.xaxis.get_ticklabels()) if i % n != 0]



def collect_dat():
    global allDataList
    currentDate = date.today()

    HOST = '192.168.0.101'  # localhost
    PORT = 7129             # Port

    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as sock:
        sock.bind((HOST, PORT))
        sock.listen()
        conn, addr = sock.accept()
        with conn:
            print('Connected by', addr)
            while True:
                in_data = conn.recv(2048).decode('utf-8')       #Hämtar data från port
                AllData=in_data                                 #

                while AllData[-2] != '}' :

                        break

                    in_data = conn.recv(2048).decode('utf-8')   #Lägger ihop data om det inte
tolkas som ett helt paket
                    AllData += in_data



                # Vid två paket eller fler så läggs paketen i "allDataList" i väntan på
uppackning
                if AllData.find('}\n{') != -1:
                    AllData=AllData.replace('}\n{', '}\n^{')
                    allDataList.extend(AllData.split("^"))
                    AllData=""

                allDataList.append(AllData)

                if not in_data:
```

```
                    break
def store_dat():

    store_date= date.today()
    counter1=0
    while True:
        if len(allDataList)!=0:
            currentDate = date.today()
            AllData=allDataList[0]
            del allDataList[0]
            #hanterande av hello-paket
            if (AllData.find('Hello') != -1):
                New_connection=True
                HelloPackage = json.loads(AllData)
                timeSync = HelloPackage["timeSynchronized"]

                #kontrollerar nyckel
                if (HelloPackage["key"] == "Not set yet"):
                    print ("Connection found!")
                else:
                    print("Enter key: ")                #Inte testat!
                    keyword = input()
                    if (HelloPackage["key"] == keyword):
                        print ("Connection found!")

                IPC_ID = HelloPackage["uid"]




            #hanterande av info-paket
            if (AllData.find('AllInfo') != -1):
                global infoInterval, program_freq
                try:
                    infoPackage = json.loads(AllData)

                    #sparar undan variabel-listor
                    #timestamp = infoPackage["timestamp"]
                    timeSync = infoPackage["timeSynchronized"]
                    infoBin = infoPackage["data"]["bin"]
                    infoBout = infoPackage["data"]["bout"]
                    infoAin = infoPackage["data"]["ain"]
                    infoInterval = infoPackage["reportInterval"]
                    program_freq = int(1/(int(infoInterval)/1000))

                except:
                    print("restart")
                    #os.startfile(__file__)
                    os.execv(sys.executable, ['python'] + sys.argv)


            #hanterande av data-paket
            if (AllData.find('AllData') != -1):

                my_file = Path(dirname(__file__)+"/"+str(currentDate)+".csv")
                #Skriver Header till csv fil om det inte filen finns
                if not my_file.exists():
                    out = ['Time']
                    for i in range(0,len(infoBin)):
                        out.append(infoBin[i]["name"])
                    for i in range(0,len(infoBout)):
                        out.append(infoBout[i]["name"])
                    for i in range(0,len(infoAin)):
```

```
                        out.append(infoAin[i]["name"])
                    out.append('timesync')
                    out.append('freq')
                    with open(str(currentDate) + ".csv",'a', newline='') as csv_file:
                        writer = csv.DictWriter(csv_file, fieldnames=out)
                        writer.writeheader()


                #skriver kommentarrad vid uppkoppling
                if New_connection:
                    with open(str(currentDate) + ".csv",'a', newline='\n') as csv_file:
                        writer = csv.writer(csv_file)

                        if (timeSync==True):
                            comment_csv='#New connection: ' + IPC_ID + '    Time is
synchronized.'
                        else:
                            comment_csv='#New connection: ' + IPC_ID + '    Warning: Time is
not synchronized!'
                        writer.writerow([comment_csv])
                        New_connection=False

                #sparar undan variabel-listor
                try:
                    allDataPackage = json.loads(AllData[:AllData.rfind('\n{\n')])
                    dataTimestamp = allDataPackage["timestamp"]
                    timeSync = allDataPackage["timeSynchronized"]
                    dataBin = allDataPackage["data"]["bin"]
                    dataBout = allDataPackage["data"]["bout"]
                    dataAin = allDataPackage["data"]["ain"]
                    dataTime=datetime.datetime.fromtimestamp(dataTimestamp/1000)            #
                    tidstr=str(dataTime)                                                    #
tar bort datum från timestamp och gör om till "vanlig" tid
                    tid=tidstr.replace(tidstr[:tidstr.find(':')-2], '').replace('.', ':')   #
                    tid=tid[:-4]

                    if store_date==date.today():
                        counter1+=1
                        if counter1==20:
                            timeList.append(tid)
                            dataBinList.append(dataBin)
                            dataBoutList.append(dataBout)
                            dataAinList.append(dataAin)
                            timeSyncList.append(timeSync)
                            counter1=0
                    else:
                        timeList.clear()
                        dataBinList.clear()
                        dataBoutList.clear()
                        dataAinList.clear()
                        timeSyncList.clear()
                        store_date==date.today()

                    #skriver data till csv-fil

                    with open(dirname(__file__)+"/" + str(currentDate) + ".csv",'a+',
newline='\n') as csv_file:
                        writer = csv.writer(csv_file)
                        out = [tid]
                        for i in range(0,len(dataBin)):
                            out.append(dataBin[i])
                        for i in range(0,len(dataBout)):
                            out.append(dataBout[i])
                        for i in range(0,len(dataAin)):
```

```
                            out.append(dataAin[i])
                        out.append(timeSync)
                        out.append(program_freq)
                        writer.writerow(out)

                except ValueError as e:
                    print("Fel vid utläsning")
                    with open(dirname(__file__)+"/"  + str(currentDate) + ".csv",'a+',
newline='\n') as csv_file:
                        writer=csv.writer(csv_file)
                        writer.writerow(['#Forlorat paket, se fil forlorade paket.'])
                    file = open("C:/Users/oscar/Documents/Exjobb/Protrol/förloradePaket_" +
str(currentDate) +".txt",'a')
                    file.write(AllData+"\n")
                    file.close()



def main():

    ani = FuncAnimation(plt.gcf(), animate, 5000, interval=200) #hämtar data för plottning,
interval är antalet millisekunder mellan varje frame i plottning

    plt.show()                              #visar graf

if __name__ == '__main__':
    print(dirname(__file__)+"/")
    t = threading.Thread(target=collect_dat)    #startar en tråd med Program som hämtar och
sparar data
    t.start()
    t1 = threading.Thread(target=store_dat)                      #
    t1.start()
    app = QtWidgets.QApplication(sys.argv)  #
    app.setStyle('Fusion')                  #
    MainWindow = QtWidgets.QMainWindow()    #Användargränssnitt
    ui = Ui_MainWindow()                    #
    ui.setupUi(MainWindow)                  #
    MainWindow.show()                       #
    main()                                  #starta plotfunktion

    sys.exit(app.exec_())                   #
```