Hur kan man effektivisera arbete och allmän användning av en dator?

Folke Ishii 3C

Gymnasiearbete HT 2020

Sofie Kjellgren

Abstract

A GUI is a form of user interface that represents information with graphical elements such as icons and buttons. If used correctly, a GUI can greatly enhance the user experience and can even achieve speed comparable to that of a command line interface.

In this paper the process of creating a simple GUI is explained. It is also focused on OOP (Object Oriented Programming) in the programming language Python. In order to create a GUI, the basics of GUI design and implementation is examined. This paper also details how a GUI can be designed to speed up and simplify the use of common computer tasks with only a keyboard and a seven-inch LCD screen.

Key words: Graphical User Interface, Object Oriented Programming, user experience

Innehållsförteckning

1. Inledning

1.1 Bakgrund

Min plan inför detta arbete var att göra någonting som har med programmering att göra, då jag har ett stort intresse för datorer. Jag hade även som mål att göra någonting tillsammans med alla Raspberry Pi-saker som jag hade liggandes hemma och jag kollade igenom mina saker för att se vad jag hade. Det var då jag hittade en skärm på 7 tum och tänkte att jag kunde kombinera det men en GUI på något sätt och vis. Jag ägnade nämligen min fritid till att utveckla några GUIs, eller Graphical User Interface, och jag visste att jag kunde kombinera detta med skärmen.

Jag kom dock inte på direkt vad jag skulle göra, och det var först när jag såg en YouTube-video som handlade om Elgato Stream Deck som jag insåg vad jag skulle göra. Jag skulle göra min egen version av Stream Deck med min touch skärm. Jag insåg snabbt att det inte var en touch-skärm trots att det stod det på lådan. Jag fick planera om och bestämde mig för att använda kortkommandon, med windowsknappen som kombinationstangent. Det var när jag spelade spel och kollade på film samtidigt som jag fick idéer till arbetet. Man ska kunna styra applikationen med kortkommandon utan att det påverkar andra program. Det var så att jag behövda pausa spelet, byta över till VLC, pausa tv-serien eller gå till nästa episod, spola över introt, byta över till och återuppta spelet. Detta tog en massa tid. Min plan för arbetet var att snabbt och smitit utföra handlingar såsom att pausa en film eller kolla på vädret.

Sedan kom det till själva processen att skapa applikationen. Jag bestämde mig för att använda mig av Python, som var det lättaste språket. När det kommer till att utveckla GUIs, så finns

det generellt sätt tre olika programmeringsspråk att välja mellan. Python, Java och C++. C++ är snabbt och man kan skapa kraftfulla applikationer men det är väldigt svårt och tar tid att utveckla ett stabilt program. Python är mycket långsammare men det är mycket lättare att skapa ett program och det kommer vara mer stabilt. Java är lite mitt emellan. Jag valde Python för att det var det språk jag hade mest erfarenhet och jag trodde inte jag skulle hinna om jag hade behövt göra applikationen i C++ eller Java.

Efter att ha valt ett programmeringsspråk, behövde jag välja ett applikationsramverk för att utveckla applikationen. Av dessa finns det fyra stora att välja emellan. Tkinter, wxPython, PyGTK och PyQt. Tkinter är inbyggt i Python men eftersom den är dålig för mer komplexa applikationer valde jag inte den. wxPython, PyGTK och PyQt är rätt så lika, men efter att ha provat olika valde jag PyQt, plus att jag hade mest erfarenhet med den.

1.2 Syfte

Syftet med detta arbete var att skapa en informativ och lättanvänd informationspanel. Man ska kunna utföra handlingar på ett mycket smidigt och snabbt sätt som inte påverkar användningen av andra program.

1.3 Metod

För att utveckla en liknande GUI ska först basen skapas så att ett tomt fönster dyker upp på skärmen. Efter det så ska hanteringen av `native events` skapas så en användare ska kunna kommunicera med applikationen. Därefter behövs en startsida som visas när applikationen startas och efter det ska varje sida skapas. Först behövs ett system för att växla mellan sidorna behövs och efter det har gjorts kan varje sida utvecklas var för sig.

1.3 Frågeställningar och avgränsningar

Denna produkt ska kunna antingen vara lösningen till ett svara på följande frågor och problem:

1. Hur kan produktiviteten ökas när man sitter vid datorn?
2. Hur ska man snabbt och smidigt få reda på information gällande ens dator och omgivning när man är upptagen med annat på datorn?
3. Vad krävs för att förbättra ens upplevelse med datorn?
4. Hur gör man när ett eller flera program tar upp all yta på skärmen/skärmarna och man vill kolla något snabbt som till exempel tiden utan att kolla på mobilen och/eller stänga ner programmen?

Avgränsningar för arbetet:

1. Applikationen ska fungera på en sju tums 800 x 480 pixlars skärm
2. Den måste kunna styras enbart med Numpad0 + Numpad tangenter samt page down
3. Den får inte interferera med andra program och tvärtom

1.4 Kravspecifikation

Hårdvara

Krav

- Kabeln till tangentbordet ska åka under stativet
- Skydda PCB:en från damm

Önskemål

- Justerbar för vinkel

Mjukvara

Krav

- Programmet ska inte ta fokus när kommandon utförts. Med andra ord ska man kunna fortsätta skriva i samma textruta efter att ett kommando har utförts.
- Andra program (och eventuellt muspekaren) ska inte kunna hamna på skärmen med programmet.
- Programmet ska styras med tangentbordet och andra tangentkombinationer som kan uppstå ska förtryckas.
- Det ska vara lätt att lägga till / ta bort program.
- Programmet ska starta med operativsystemet.

Önskemål

- Använda lite av datorns resurser
- Programmet ska fungera på både Windows och Linux
- Det ska fungera på flera olika skärmstorlekar

1.5 Material

- Sju tums skärm 800x480 LCD HDMI
- ”Monitor Driver Board”
- 12V DC Adapter

2 Teoretisk bakgrund

2.1 Applikation

I skapandet av applikationer så finns det mycket att tänka på. Arbetet handlar om att skapa en GUI (Graphical User Interface / Grafiskt Användargränssnitt) och koppla det med olika kommandon som datorn utför. Vad krävs egentligen för att skapa ett grafiskt användargränssnitt?

I utvecklingen av ett program finns det mycket som behövs tänkas på. Den viktigaste att tänka på är hanteringen av trådar. Trådar är i princip när en CPU kan köra program samtidigt och detta är mycket viktigt när det gäller GUIs. När en GUI utvecklas, ska en tråd hantera alla grafiska element såsom knappar och bilder. Om datorn behöver beräkna något ska man hantera det på en separat tråd, så att applikationen inte hänger sig. För att beräkna saker, behöver den göra det parallellt med den grafiska delen av programmet. Detta är väldigt viktigt. Varje sekund så skaffar samt bearbetar denna tråd information och skickar över det till den grafiska tråden via signaler. Därefter kan den grafiska tråden visa all information.

När det gäller kortkommandon, finns det några saker som behövs tänkas på, speciellt om det ska göras på liknande sätt som denna applikation. Programmet ska kunna känna igen kortkommandot oberoende på vilket program som ligger i fokus och det ska inte ändra fokus. Detta problem löses med globala kortkommandon, som är kortkommandon som ett program kan känna igen när som helst. PyQt5 stödjer inte globala kortkommandon men ett bibliotek som gör det är *pyqtkeybind*. Med detta program kan Windows hantera händelser via Qt:s ”QAbstractNativeEventFilter” genom att installera det ett program. Sedan kan kortkommandon initieras och därefter kan datorn lyssna efter dem via hanteraren och när den upptäcker att användaren har trycket på en viss tangent-kombination så körs en funktion i programmet.

Ett problem uppstod när tangent-kombinationerna byttes från att använda windowsknappen som kombinationstangent till numpad-0. Eftersom windowsknappen finns bland `fsModifiers`, de tangenter som kombineras med en `uVirtKey` för att skapa ett kortkommando, blir det lätt att använda den som en kombinationstangent. Numpad-0 finns dock inte bland dessa knappar och det blir omöjligt att skapa ett kortkommando[1]. Med programmet AutoHotKey kan detta problem lösas genom att datorn inte skickar ut meddelandet till operativsystemet att en knappt har tryckts efter tangenten har släppts, till skillnad från det vanliga när tangenten trycks ned. Under tiden som numpad-0 är nertryckt, kan programmet känna av om en annan tangent trycks ner och kan då skicka ut en annan tangentkombination som ett vanligt program kan känna igen. Numpad-0 + Numpad-7 blir till Ctrl+F13. En tabell över alla kombinationer kan då skapas:

Tangentkombinationer som sidorna kan kontrollera och styra över vad som kommer hända:

TANGENT	KOMBINATION SOM SKICKAS
NUMPAD 7	Ctrl + F13
NUMPAD 8	Ctrl + F14
NUMPAD 9	Ctrl + F15
NUMPAD 4	Ctrl + F16
NUMPAD 5	Ctrl + F17
NUMPAD 6	Ctrl + F18
NUMPAD +	Ctrl + F19

Tangentkombinationer som varje sida ej kan kontrollera:

TANGENT	KOMBINATION SOM SKICKAS
NUMPAD , (DEL)	Ctrl + Alt + F13
NUMPAD -	Ctrl + Alt + F14
NUMPAD *	Ctrl + Alt + F15
NUMPAD /	Ctrl + Alt + F16
NUMPAD + PAGE DOWN	Ctrl + Alt + F17

[1] https://docs.microsoft.com/sv-se/windows/win32/api/winuser/nf-winuser-registerhotkey?redirectedfrom=MSDN

Programmet kan även visa datorns volym, vilket inte är möjligt genom PyQt5. För att få systemvolymen, används PyCAW. Med PyCAW kan en pekare skaffas som pekar till ”slutpunkten” för högtalarna[2]. Med `IAudioEndpointVolume::GetMasterVolumeLevel`[3] fås volymen i decibel, men volymens önskas att fås i procent. Får att få volymen i procent, behöves metoden `IAudioEndpointVolume::GetMasterVolumeLevelScalar`[4] användas vilket ger volymen i procent.

För att få tiden i sekunder samt datumet, används `datetime.now()`. Med hjälp av funktionen `strftime()` kan information såsom tiden på dygnet extraheras.[5]. `%H` ger timmen med nolla framför ensiffriga nummer baserat på 24-timmars klockan. `%M` ger minuter och `%S` ger sekunder i samma format. För datum ger `%V` ISO 8601 veckan, `%a` ger dagen i tre bokstäver, `%d` ger dagen med nolla framför ensiffriga nummer, `%b` ger månaden i tre bokstäver och `%Y` ger hela året. Det är viktigt att nollan sätts framför ensiffriga nummer så att strängen som håller datum och tid alltid har samma längd. Sedan behöver typsnittet vara icke-proportionellt så att den är alltid lika bred.

Applikationen använder sig av hemlig information för att kunna fungera. VLC kräver ett lösenord och Spotify kräver `CLIENT_ID`, `CLIENT_SECRET` samt `REDIRECT_URI`. För att detta inte ska läckas är det viktigt att denna information inte finns inuti koden. Vad man gör är att skapa en `credentials.cfg` fil som innehåller all hemlig information. När ett program behöver informationen läser den bara in det från filen. För att denna fil inte ska hamna på git så behöver man lägga till filen i `.gitignore`.

2.2 Spotify och Väder

En API är en specifikation för hur program kan överföra information och kommunicera med varandra. Spotify och Väder använder sig av API över HTTPS för att få information. Spotify skickar en begäran till Spotify servrar och servrarna skickar tillbaka information om vilken låt som spelas. Det är samma princip för väder, då en begäran skickas till SMHI med koordinater och tillbaka skickas vädret på den platsen.

Både Spotify- och väderapplikationen använder sig av API:s för att skaffa information från tjänster. Spotify läser in från `credentials.cfg` och ansluter sig till Spotifys nätverk. När programmet frågar efter vad användaren spelar på Spotify. Spotify skickar tillbaka en JSON-fil med information och den information som önskas fås tas fram. Väderapplikationen fungerar ungefär likadant då den skickar en begäran till `opendata-download-metfcst.smhi.se` som skickar tillbaka en JSON-fil beroende få longitud och latitud som

[2] https://docs.microsoft.com/en-us/windows/win32/coreaudio/audio-endpoint-devices
[3] https://docs.microsoft.com/en-us/previous-versions/ms678749(v=vs.85)
[4] https://docs.microsoft.com/en-us/windows/win32/api/endpointvolume/nf-endpointvolume-iaudioendpointvolume-getmastervolumelevelscalar
[5] https://man7.org/linux/man-pages/man3/strftime.3.html

specificerades i URL:n. En arbetar-tråd bearbetar informationen och skickar informationen vidare till huvudapplikationen.

2.3 VLC

Applikationen behöver skicka kommandon till en VLC instans, till skillnad från Spotify och SMHI kan detta inte göras via en API eftersom VLC är ett program på datorn och inte en server över internet. Istället så används nätverksprotokollet Telnet för att kommunicera med VLC. För att starta Telnet behöver först VLC öppnas och sedan initierar programmet en Telnet kommunikation via port `4212` på `localhost`. Sedan skickas ett lösenord för att verifiera och därefter kan programmet och VLC kommunicera med varandra. Applikationen skickar en begäran om information till VLC som skickar tillbaka information och/eller utfärdar ett kommando såsom att pausa videon.

2.4 Applikationer, Mappar och hemsidor

Alla dessa tre applikationer bygger på samma princip, information läses från en JSON-fil som består av en lista med varje applikation. Varje listelement ska, för detta ändamål, innehålla namnet som visas på skärmen, vad som ska ske när användaren väljer detta element samt en ikon. I ”Browsers” fall så består informationen av en URL, ”Mappar” och ”Appar” innehåller en sökväg. ”Appar” är ett specialfall då det är bökigt att öppna ett Steam-spel från en sökväg och man ska använda `steam://<AppID>` då `AppID` är spelets applikations-ID.[6]

Alla element läggs i en rutnätslayout som användaren kan skrolla igenom.

2.5 Systeminformation

För att visa systeminformation, används modulerna psutil (python system and process utilities) och GPUtil. Med psutil är det lätt att få hur mycket RAM som används och hur mycket av CPU:n som används. Kommandot `cpu_percent()` ger ett flyttal som visar hur mycket av CPU:n som används i procent. Kommandot `virtual_memory().percent` ger hur mycket minne som datorn använder i procent, också som ett flyttal.

När det kommer till hur mycket av nätverket som används, så blir det lite mer komplicerat. Kommandot `net_io_counter(pernic=True)` ger hur mycket data som har skickats/mottagits. För att beräkna hur mycket som har skickats varje sekund, får man köra kommandot varje sekund och beräkna skillnaden mellan den nuvarande och den som kördes för en sekund sedan.[7]

GPUtil användes för att skaffa information om grafik-kortet och det görs genom att förs skaffa GPU:n via kommandot `getGPUs()` och spara det i en variabel, som till exempel `_gpu`. För att få hur mycket av GPU:n som används kör man kommandot `_gpu[0].load` vilket ger

[6] https://developer.valvesoftware.com/wiki/Steam_Application_IDs
[7] https://psutil.readthedocs.io/en/latest/index.html?highlight=net_io_counter#psutil.net_io_counters

en användningen i procent som ett flyttal. För att få temperaturen, används kommandot `_gpu[0].temperature`.

2.6 Gränssnitt

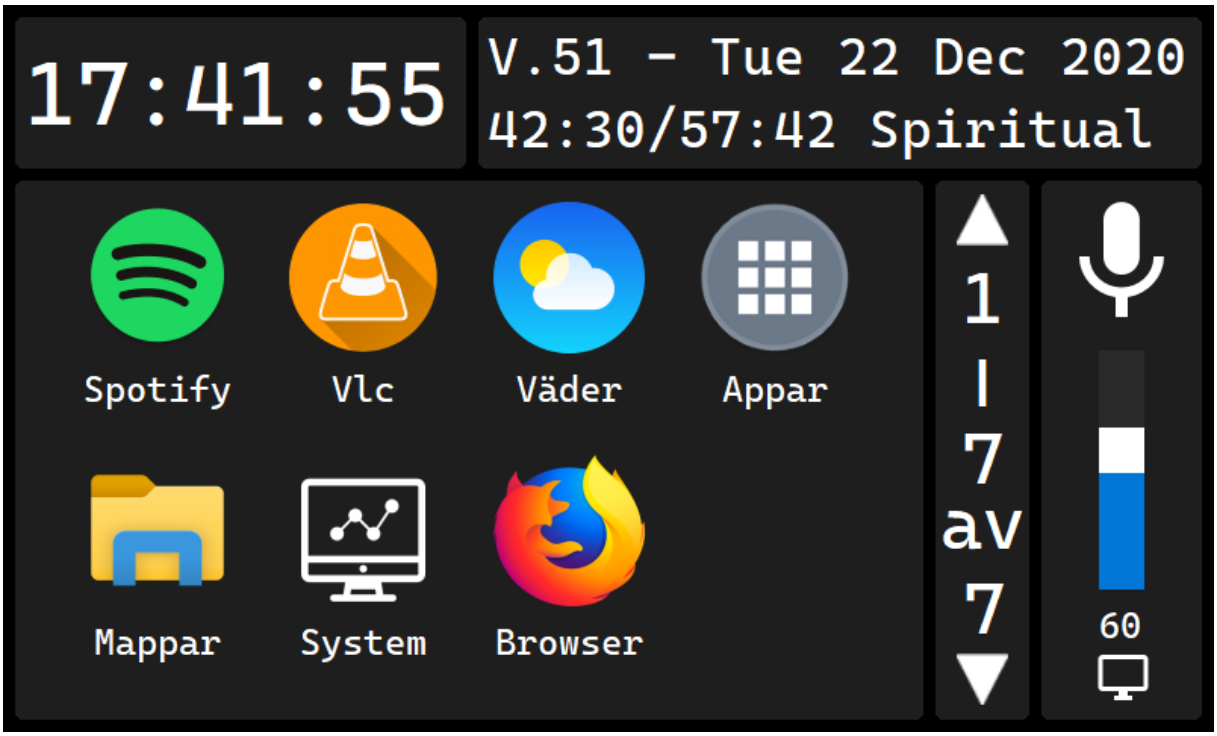
Gränssnittet visas genom att använda ett ”stylesheet”, som berättar hur alla saker ska se ut. Bakgrunden är en bild som laddas in. Resten av gränssnittet består mestadels utav textrutor och några få ikoner.

3 Resultatredovisning

3.1 Applikation och användning

Koden finn tillgänglig på Github.

När det kommer till användning av datorns resurser använder programmet 0%-0.3% av CPU:n och runt 49 MB RAM. När en sida ska göra utföra något speciellt krävande arbete såsom att skaffa en bild och visa den, ökar användningen av resurser i en sekund för att sedan återgå antingen efter en sekund eller när en bild inte ska visas längre.

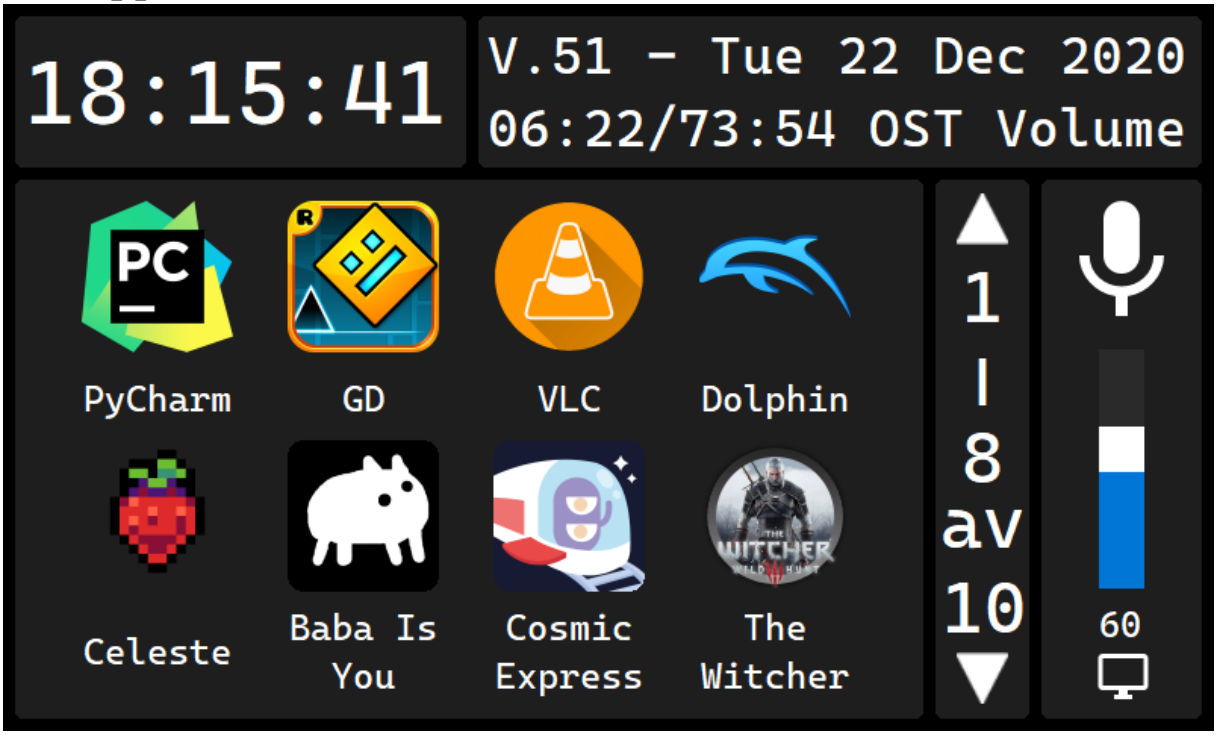


Figur 1, startsidan på applikationen.

Startsidan består av fem delar som har sig egen ruta. Uppifrån ner och vänster till höger så är den första delen tiden på dygnet som en 24-timmars klocka. Nästa ruta innehåller information om datumet, med vecka, veckodygn, datum, månad samt år. Under i samma ruta visas information om Spotify, nämligen hur långt in i låten samt längden på låten och vad låten heter. I nästa ruta är rutan som alla sidor ligger i. Nästa ruta består av en rullningslist som visar olika beroende på vad som visas på förgående ruta. Startskärmen visar hur många olika sidor det finns medan ”Appar” visar hur många appar som finns samt vilka av dessa som är på skärmen. Sista rutan visar volymen samt en mikrofon. Mikrofonen visar om mikrofonen är på eller av. Ljudet visar volymen på datorn.

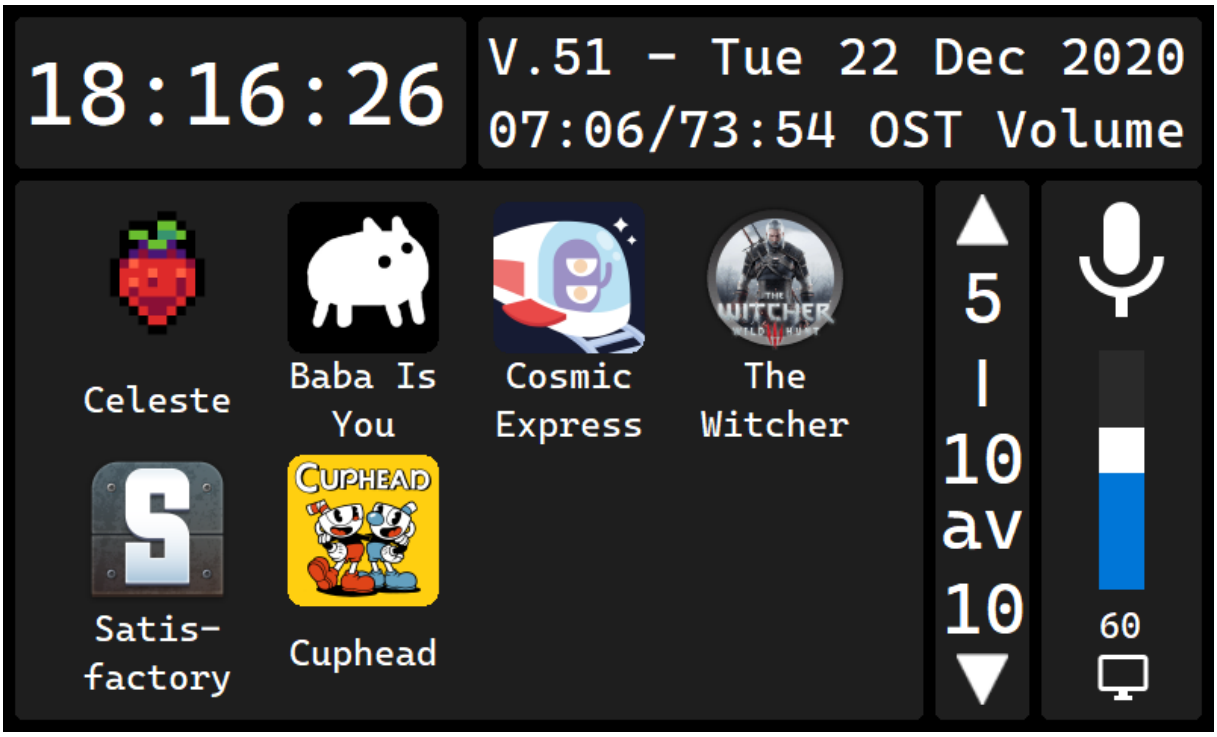
Som ett test sattes skärmen upp när en användare satt och spelade ett datorspel samtidigt som dem kollade på en film på en annan skärm. När användaren (jag) behövde pausa filmen behövde jag pausa spelet, byta fokus till filmspelaren, klicka på en knapp för att pausa filmen, sedan gå tillbaka till spelet och sist återuppta spelet. Detta uppmättes till att ta cirka 3 sekunder. Sedan skulle användaren genomföra samma sak, fast med skärmen och applikationen installerad. Det enda som användaren behövde göra var att klicka på Numpad 0 och sedan Numpad 5. Det tog runt en halv sekund. Applikationen sparade 2.5 sekunder.

3.2 Applikation-startare



Figur 2, Appar, första sidan

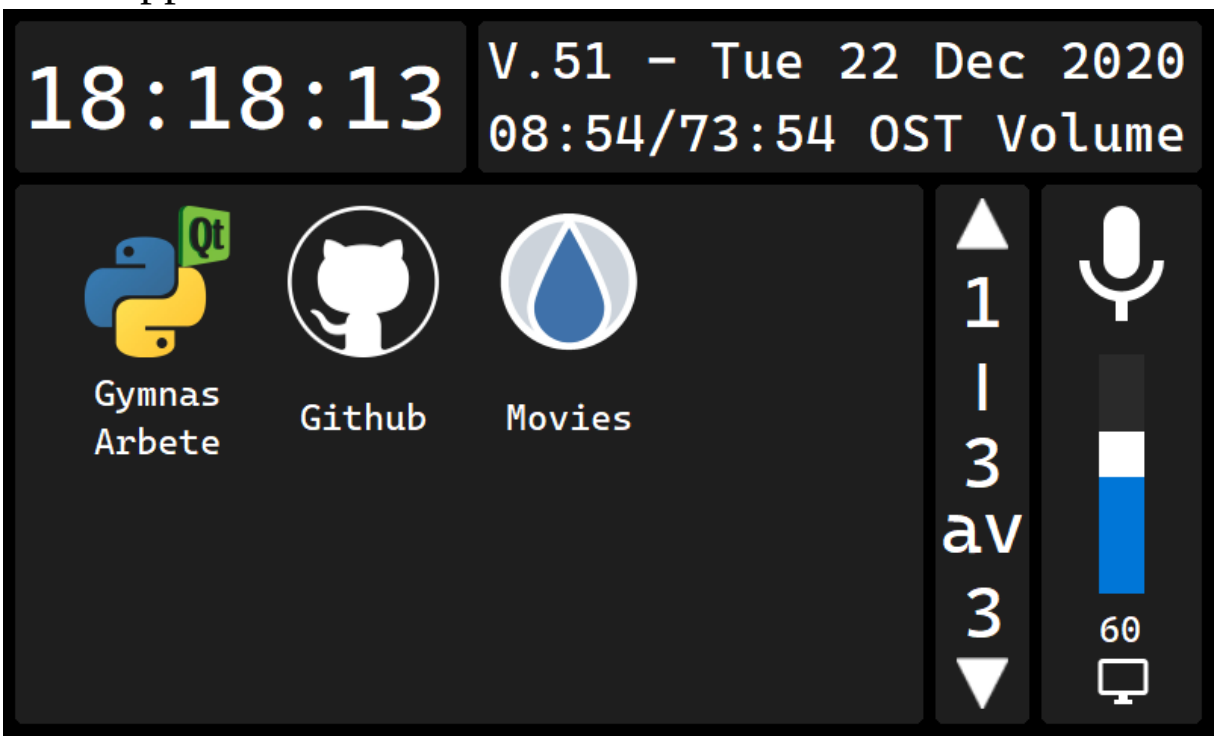
Applikation-startaren består av en rutnätslayout. Här ser vi att applikationerna nummer ett till nummer fem visas på skärmen. Med Numpad0 + Numpad7 kan man skrolla ner:



Figur 3, Appar, andra sidan

Den nedre raden har flyttats upp och en ny rad visas. På skärmen visas applikation nummer fem till tio, som ses på rullningslisten. Om man väljer ett program så startas programmet.

3.3 Mappar



Figur 4, Mappar

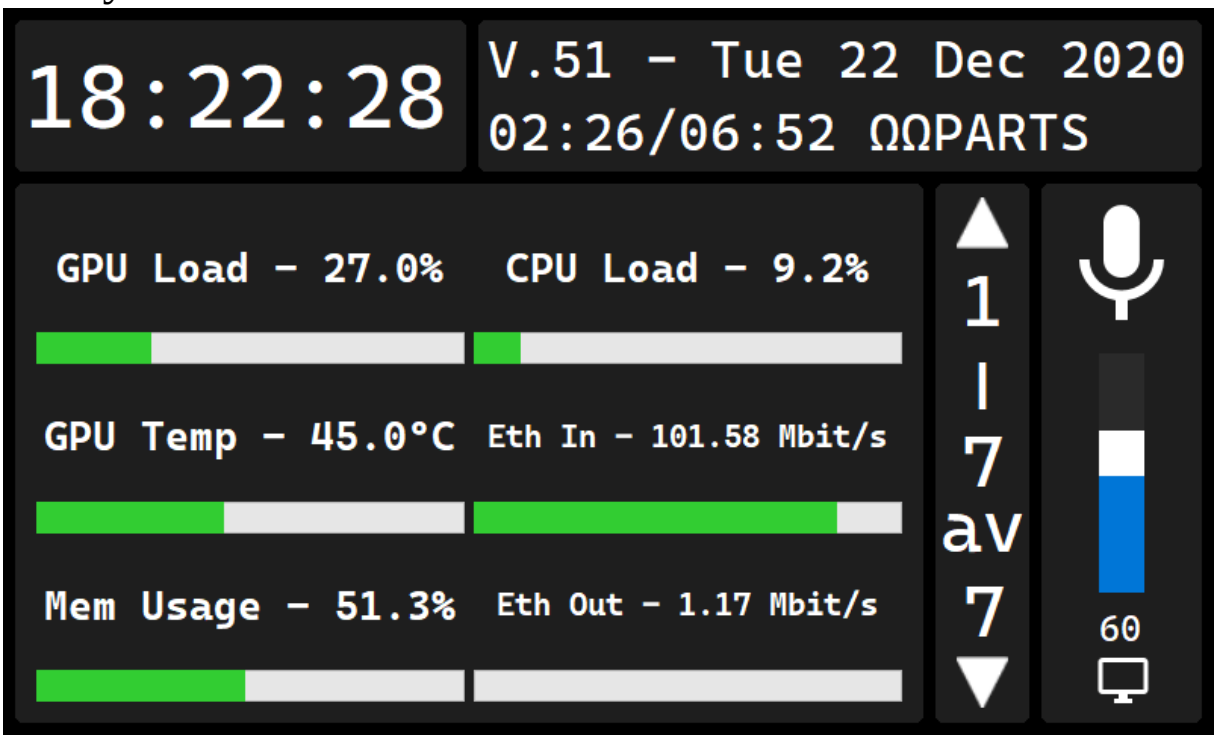
Mappar visar olika mappar på datorn. Om användaren väljer en mapp öppnas utforskaren i den mapp.

3.4 Spotify

Figur 5, Spotify

Spotify sidan visar mer information om låten. Till vänster finner vi låtens bild. Den övre textrutan visar artistens/bandets namn och den nedre rutan visar låtens namn. Lägg märke till hur textrutan där nere är samma som den uppe till höger. Textrutan uppe till höger kan dock inte visa hela låtens namn.

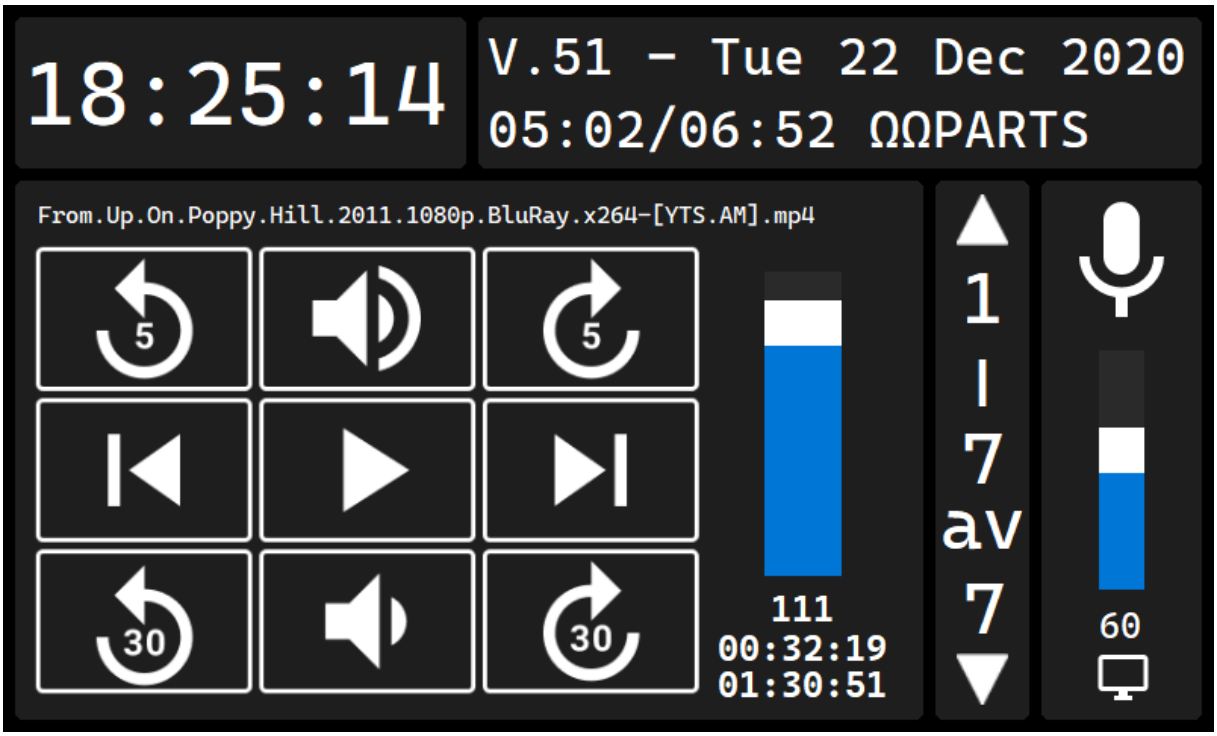
3.5 Systeminformation



Figur 6, Systeminformation

Systeminformation använder sig av sex olika förloppsindikatorer för att visa hur mycket av datorns resurser som används.

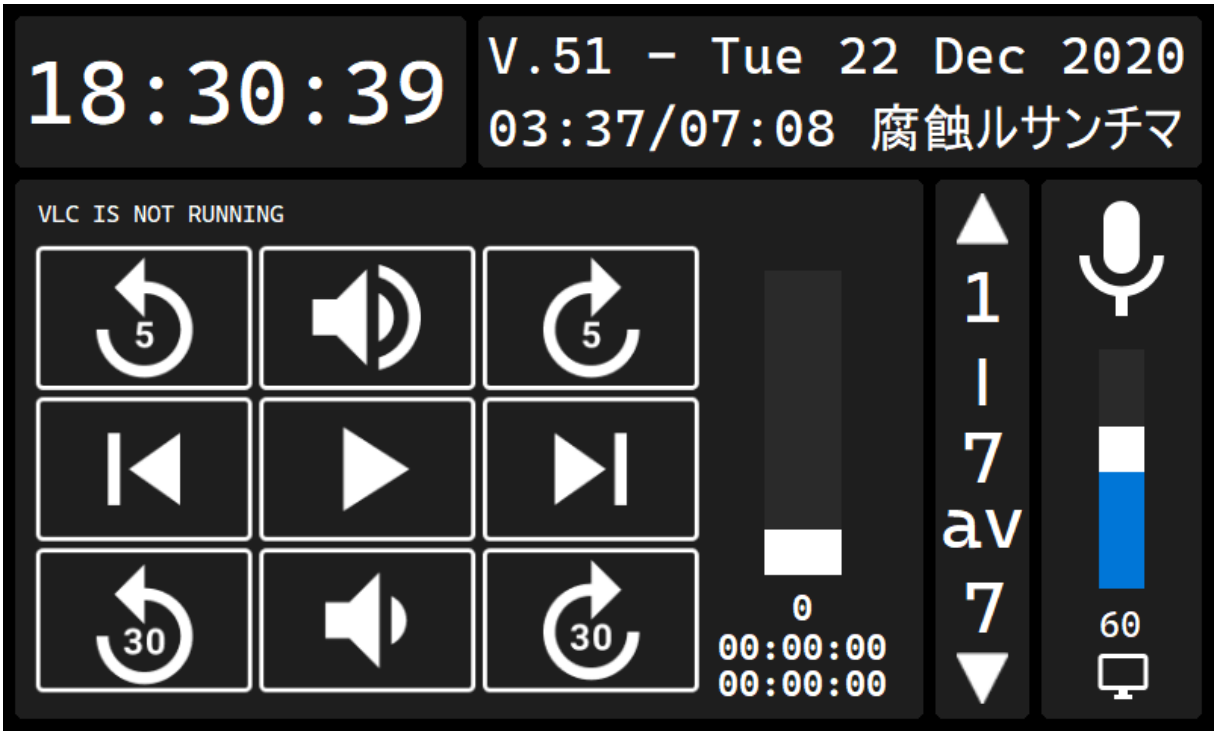
3.6 VLC



Figur 7, VLC när film visas

VLC sidan visar ett tre gånger tre rutnät som korresponderar till knapparna Numpad1 – Numpad9, som också ligger i ett tre gånger tre rutnät. Knappen längst upp till vänster – Numpad7 spolar tillbaka filmen fem sekunder. Nästa knapp ökar volymen. Därefter så spolas tiden fram fem sekunder, sedan spelas förgående film i spellista upp, sedan pausas/spelas filmen. Därefter så visas nästa film, sedan spolas tiden 30 sekunder tillbaka, sedan sänks volymen och sist så spolas tiden fram 30 sekunder.

Längst upp så visas titeln på filmen. Till höger om rutnätet visas volymen på VLC, under det så visas hur långt in i filmen man är och sedan hur lång filmen är. Om sidan öppnas och VLC inte kommer igång visas:



Figur 8, VLC är inte igång

3.7 Väderprognos

Figur 9, Väderprognos

Vädret visas i ett fyra gånger sex rutnät som visar vädret de närmaste 24 timmar. Ikonen representerar vädret i helhet den timmen. Bredvid ikonen visas tiden på dygnet. Under tiden visas temperaturen och sedan nederbörd.

3.8 Ställ och skärm

På grund av vissa omständigheter kunde stället inte skrivas ut.

4 Diskussion och slutsatser

Användning

Programmet lyckas verkligen med att effektivisera och förenkla användning, speciellt när man är inuti ett spel och inte kan byta fokus. Efter att ha använt applikationen ett tag är jag väldigt nöjd med applikationen. En av de bättre aspekterna är hur man kan styra hela programmet med enbart en hand. Detta är en stor fördel om man använder en kontroll. Innan när man styrde programmet med två händer behövde man lägga ner kontrollen för att utföra ett kommando men nu behövs inte det vilket sparar en massa tid. Att många av sakerna kan utföras med ett enda kommando gör det lättare att komma ihåg samt att utföra det.

Det finns några saker som kan förbättras. Ett problem gällande väder sidan är att det kan bli svårt att se texten då den är rätt så liten. En förbättring man skulle kunna göra är att dela upp dygnet på två eller tre delar och låta användaren skrolla igenom dygnet. Sedan så har Spotify-sidan ett litet problem, och det är att den inte uppdateras när en ny låt spelas. Den nuvarande lösningen är att man måste gå tillbaka till startskärmen och sedan in i sidan igen. System-sidan skulle kunna förbättras med användandet av runda förloppsindikatorer istället för raka.

Programmeringen

När koden började att skrivas, hade jag inte särskilt mycket erfarenhet gällande utveckling av GUIs. Efter att ha utvecklat några andra program vid sidan om efter arbetet, och kollat igenom koden märktes några saker som skulle kunna förbättras. Det största problemet är kod inte återanvänds med OOP. Mappar, Browser och Appar fungerar likadant och man skulle lätt kunna skapa en bas klass för dessa tre och det enda som varje sida behöver hantera är vad som händer när användaren väljer ett element. Jag blev inte heller nöjd med strukturen. Hur det fungerar nu är att det inte är en python-modul[8] vilket leder till att man importerar program som om det vore inom samma mapp.

```
.
└── Gymnasiearbete/
    ├── app/
    │   ├── assets/
    │   │   └── ...
    │   ├── widgets/
    │   │   └── ...
    │   ├── app.pyw
    │   ├── infothread.py
    │   └── ...
    └── docs/
        └── ...
```

Så här ser strukturen ut för projektet och för att `app.pyw` ska kunna använda sig av klassen `InfoThread` från `infothread.py` måste man skriva `from infothread import InfoThread` eftersom filerna är inom samma mapp. Denna metod fungerar men det blir väldigt jobbigt när man ska använda sig av saker som ligger flera mappar bort antingen upp eller ner. För att lösa detta borde man ha lagt en `setup.py` i `Gymnasiearbete` och kört `pip install -e .` för att göra det till en modifierbar modul. Då kan man importera filer via

```
from Gymnasiearbete.app.infothread import InfoThread
```

vilket är mycket bättre speciellt om man ska distribuera sitt program.

Eftersom hårdvaran inte kunde framställas, uppnåddes inga krav eller önskemål på hårdvaru-sidan. När det kommer till mjukvaran uppnåddas alla krav och ett önskemål. Det önskemål som uppfylldes var att applikation skulle använda lite av datorns resurser. Programmet använder mycket lite av datorns processor men relativt mycket med minne. 0.25 % av allt minne är inte mycket men om ett annat språk skulle användas som C++ skulle denna siffra vara mycket lägre.

Sammanfattning

För att förenkla och effektivisera allmän användning av en dator, kan en skärm läggas till för att kolla på saker som normalt sätt måste göras genom att uppehålla det man håller på med. Detta leder till att många av dagliga funktioner går att göra smidigt och snabbt och i många

[8] https://docs.python.org/3/tutorial/modules.html

fall bättre. Att utveckla ett program med en föränderlig metod leder till att det går lätt och snabbt att lägga till- och ändra på sidor om det behövs.

Det går definitivt att effektivisera och förenkla allmän användning enbart med hjälp av en liten skärm. Att enkelt kunna navigera runt sidor med en hand som visar information underlättar en när man bland annat spelar spel och kollar på film samtidigt.

5 Källförteckning

Andre Miras, (2020), pycaw (version 20181226). Hämtad 29 september 2020. https://github.com/AndreMiras/pycaw

Arun Mahapatra, (2020), pyqtkeybind (version 0.0.6). Hämtad 8 oktober 2020. https://github.com/codito/pyqtkeybind

Giampaolo Rodola, (2020), psutil (version 5.7.2). Hämtad 7 oktober 2020. https://psutil.readthedocs.io/en/latest/

Kenneth Reitz, (2020), Requests (version 2.24.0). Hämtad 29 september 2020. https://requests.readthedocs.io/en/master/

Paul Lamere, (2020), Spotipy (version 2.16.0). Hämtad 30 september 2020. https://spotipy.readthedocs.io/en/2.16.0/

Python Software Foundation. (2021). *Python 3.9.1 documentation*. Hämtad 18 september 2020, https://docs.python.org/3/

Python Software Foundation. (2021). *Telnet client*. Hämtad 7 oktober 2020, https://docs.python.org/3/library/telnetlib.html

The Qt Company. (2020). *Qt for Python Modules*. Hämtad 28 augusti 2020, https://doc.qt.io/qtforpython/api.html

6 Bilagor

Kod: https://github.com/Secozzi/Gymnasiearbete