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

Folke Ishii 3C

Gymnasiearbete HT 2020

Sofie Kjellgren

# Abstract

Ja

Innehållsförteckning

[Abstract 1](#_Toc59289461)

[1. Inledning 2](#_Toc59289462)

[1.1 Bakgrund 2](#_Toc59289463)

[1.2 Syfte 3](#_Toc59289464)

[1.3 Frågeställningar och avgränsningar 3](#_Toc59289465)

[1.4 Kravspecifikation 3](#_Toc59289466)

[1.5 Metod och material 4](#_Toc59289467)

[2 Teoretisk bakgrund 4](#_Toc59289468)

[2.1 Applikation 4](#_Toc59289469)

[2.2 Spotify och Väder 5](#_Toc59289470)

[2.3 VLC 6](#_Toc59289471)

[2.4 Applikationer, Mappar och hemsidor 6](#_Toc59289472)

[2.5 Systeminformation 6](#_Toc59289473)

[2.6 Gränssnitt 6](#_Toc59289474)

[3 Resultatredovisning 6](#_Toc59289475)

[3.1 Applikation och användning 6](#_Toc59289476)

[3.2 Applikation-startare 6](#_Toc59289477)

[3.3 Mappar 6](#_Toc59289478)

[3.4 Spotify 6](#_Toc59289479)

[3.5 Systeminformation 6](#_Toc59289480)

[3.6 VLC 6](#_Toc59289481)

[3.7 Väderprognos 6](#_Toc59289482)

[3.8 Hemsida-startare 6](#_Toc59289483)

[4 Diskussion och slutsatser 6](#_Toc59289484)

[5 Källförteckning 6](#_Toc59289485)

[6 Bilagor 7](#_Toc59289486)

# 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 tid 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 Frågeställningar och avgränsningar

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

1. Hur kan man öka produktiviteten 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 mitt arbete:

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 Metod och material

# 2 Teoretisk bakgrund

## 2.1 Applikation

I skapandet av applikationer så finns det mycket att tänka på. Det jag gjort i mitt arbete är att skapa en GUI (Graphical User Interface / Grafiskt Användargränssnitt) och kopplat det med olika kommandon som datorn utför. Vad krävs egentligen för att skapa ett grafiskt användargränssnitt?

När man ska utveckla program finns det mycket man ska tänka på. Den viktigaste att tänka på är hur man ska hantera trådar. Trådar är i princip när en CPU kan köra program samtidigt, och detta är mycket viktigt för GUI. När man gör en GUI, 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 extremt 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 saker man ska tänka på, speciellt om man ska göra det på det sättet jag implementerade de på. 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 vi hantera Windows händelser via Qt:s ”QAbstractNativeEventFilter” genom att installera det på vårt program. Sedan kan vi initiera kortkommandon och lyssna efter dem via vår hanterare och när den upptäcker att användaren har trycket på en viss tangent-kombination så kör det kod som gör vad vi vill i programmet.

Ett problem som uppstod var 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 man kan då inte skapa ett kortkommando[[1]](#footnote-2). Med programmet AutoHotKey kan detta problem lösas genom att datorn inte skickar ut meddelandet till operativsystemet att en knappt har tryckts efter man har släppt tangenten, och inte vanligtvis när man trycker ner den. 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 vårt program kan känna igen. Numpad-0 + Numpad-7 blir till Ctrl+F13.

Programmet kan även visa datorns volym, vilket inte är möjligt genom PyQt5. För att få systemvolymen, användsPyCAW. Med PyCAW kan en pekare skaffas som pekar till ”slutpunkten” för högtalarna[[2]](#footnote-3). Med IAudioEndpointVolume::GetMasterVolumeLevel*[[3]](#footnote-4)* fås volymen i decibel, men volymens önskas att fås i procent. Får att få volymen i procent, behöver man använda metoden IAudioEndpointVolume::GetMasterVolumeLevelScalar*[[4]](#footnote-5)* vilket ger oss volymen i procent.

För att få tiden i sekunder samt datumet, används datetime.now(). För att få ut vad vi vill, används strftime(), ett sätt att hantera datum och tid[[5]](#footnote-6). %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 så 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

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 specificerades i [URL:n](file:///C:\Users\folke\OneDrive\Documents\Programmering\Python\Github\Gymnasiearbete\docs\n). En arbetar-tråd bearbetar informationen och skickar informationen vidare till huvudapplikationen.

## 2.3 VLC

Applikationen behöver skicka kommandon till en VLC instans, och till skillnad från Spotify och SMHI kan detta inte göras via en API eftersom VLC är ett program och inte en hemsida. 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.

## 2.5 Systeminformation

## 2.6 Gränssnitt

# 3 Resultatredovisning

## 3.1 Applikation och användning

Applikationen uppfyllde alla mjukvarokrav och ett enda önskemål. Jag hade som önskemål att den skulle använda lite av datorns resurser och den använder 0%-0.3% av CPU:n och runt 0.2% av allt RAM.

## 3.2 Applikation-startare

## 3.3 Mappar

## 3.4 Spotify

## 3.5 Systeminformation

## 3.6 VLC

## 3.7 Väderprognos

## 3.8 Hemsida-startare

# 4 Diskussion och slutsatser

# 5 Källförteckning

* <https://docs.python.org/3/>
* <https://docs.python.org/3/library/telnetlib.html>
* <https://doc.qt.io/qtforpython/>
* <https://github.com/codito/pyqtkeybind>
* <https://psutil.readthedocs.io/en/latest/>
* <https://spotipy.readthedocs.io/en/2.16.0/>
* <https://requests.readthedocs.io/en/master/>
* <https://github.com/AndreMiras/pycaw>

# 6 Bilagor

1. <https://docs.microsoft.com/sv-se/windows/win32/api/winuser/nf-winuser-registerhotkey?redirectedfrom=MSDN> [↑](#footnote-ref-2)
2. <https://docs.microsoft.com/en-us/windows/win32/coreaudio/audio-endpoint-devices> [↑](#footnote-ref-3)
3. <https://docs.microsoft.com/en-us/previous-versions/ms678749(v=vs.85)> [↑](#footnote-ref-4)
4. <https://docs.microsoft.com/en-us/windows/win32/api/endpointvolume/nf-endpointvolume-iaudioendpointvolume-getmastervolumelevelscalar> [↑](#footnote-ref-5)
5. <https://man7.org/linux/man-pages/man3/strftime.3.html> [↑](#footnote-ref-6)