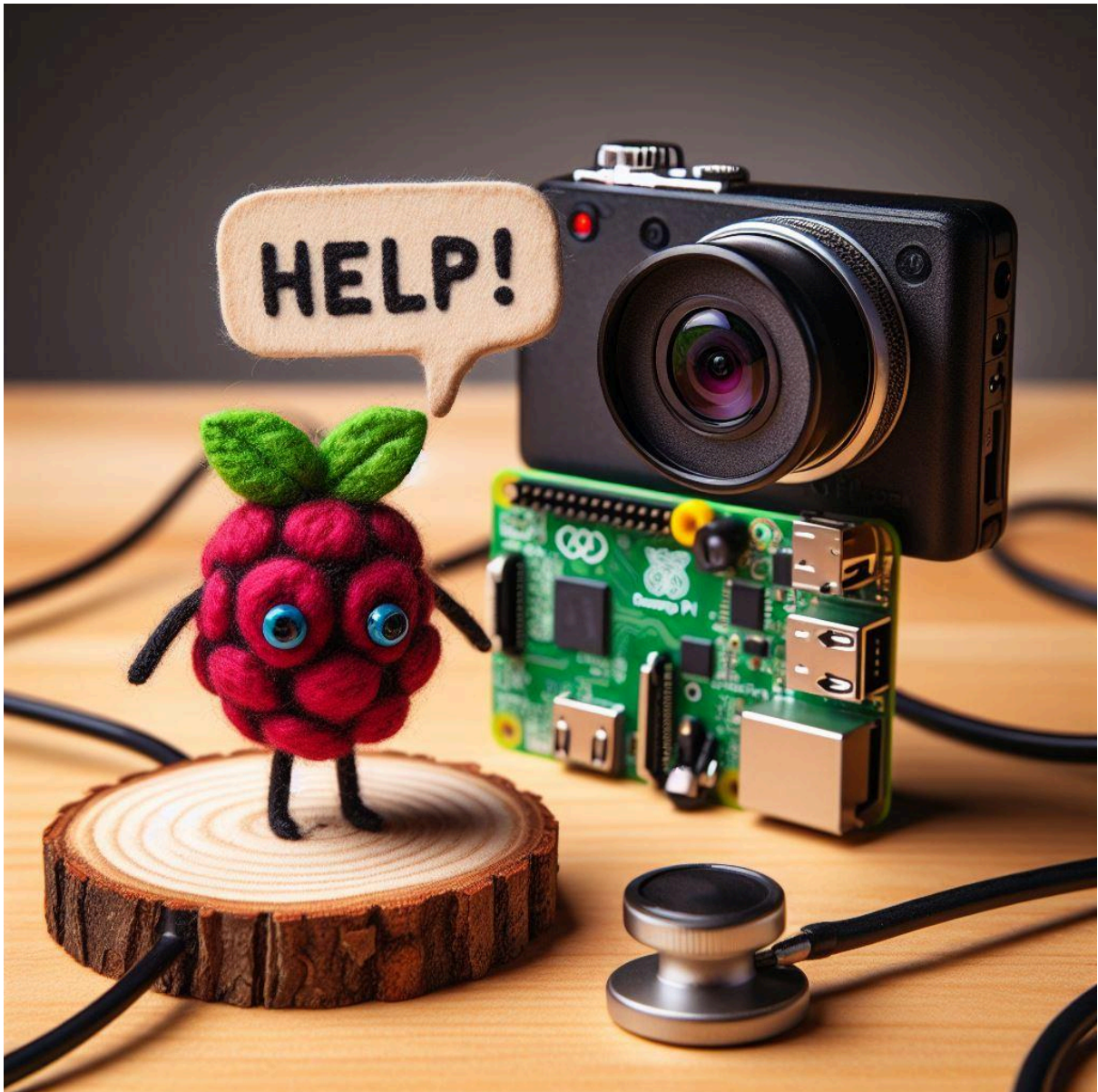


RPI Emergency Assistant

Teknisk Dokumentation



Johan Romeo - JINV23 Campus Mölndal

Innehållsförteckning - Teknisk Dokumentation

Innehållsförteckning - Teknisk Dokumentation.....	1
Produktöversikt.....	2
Funktion och användning.....	2
Funktionsbeskrivning.....	2
Användning av produkten.....	2
Teknik och verktyg.....	3
Hårdvarukomponenter.....	3
Raspberry Pi 5.....	3
Mikrofon.....	3
Kameramodul.....	3
Utvecklingsverktyg.....	5
Pyton.....	5
PocketSphinx - LiveSpeech.....	6
Picamera2.....	7
Azure IoT SDK.....	7
Datahantering och lagring.....	8
Azure Blob Storage.....	8
Arkitektur och design.....	12
Klass- och modulstruktur.....	12
Referenser.....	13
Officiella Dokumentation.....	13

Produktöversikt

Funktion och användning

Funktionsbeskrivning

Produkten är avsedd för att vara en IoT-enhet som kontinuerligt lyssnar på omgivningen med hjälp av en usb-mikrofon. När ordet “help” fångas upp, aktiveras kameran och en tio sekunder lång videoinspelning med ljud sätts igång. I slutet av videoinspelningen tas även en bild. Videoinspelningen och bilden skickas till Microsoft Azures Blob Storage för lagring och hantering.

Användning av produkten

Produktens är avsedd för följande användare:

- De äldre människorna som inte kan resa sig själva när de till exempel ramlat och slagit sig, då syftet med produkten är att de ska ropa “help” för att få hjälp.
- De människor som har tillgång till Microsoft Azures Blob Storage för att snabbt se videoinspelning samt bild för att snabbt kunna agera utefter situationen i fråga om så krävs.

Teknik och verktyg

Hårdvarukomponenter

Raspberry Pi 5

Raspberry Pi 5 har använts i detta projekt då den är liten och kompakt och har dessutom stöd för de tekniska komponenter som krävdes, såsom extern kameramodul för videoinspelningar samt usb-mikrofon för röstaktivering. Förutom detta integreras den enkelt med Microsoft Azure IoT Hub och Blob Storage, tack vare Azures SDK för Python.

Operativsystemet som har använts är Raspberry Pi OS som baserar sig på Debian (Linux), med nuvarande version, Bookworm. [\[1\]](#)

Mikrofon

Den externa mikrofonen som använts för detta projekt är en budget “plug-and-play” usb-mikrofon [\[2\]](#), vars enda syfte är att aktivt lyssna efter en specifik nyckelfras från användaren, i detta fall ‘help’.

För att på ett lyckat sätt använda sig av en mikrofon i detta program måste man ange id på ljudkortet samt id på mikrofonen. Detta görs genom att köra kommandot: “arecord -l” i terminalen på din Raspberry Pi 5 enhet.

Kommandot kommer att lista systemets ljudkort samt ljudingångar och visas så här:

```
(rpi-env) pi@raspberrypi:~/rpi-emergency-assistant $ arecord -l
**** List of CAPTURE Hardware Devices ****
card 2: Device [USB PnP Sound Device], device 0: USB Audio [USB Audio]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
(rpi-env) pi@raspberrypi:~/rpi-emergency-assistant $
```

Kameramodul

Kameramodulen som använts i detta projekt är från tredjepartsleverantör och är inte en av Raspberry Pi's officiella kameramoduler.

Teknisk specifikation - Freenove 5MP Camera for Raspberry Pi:

- Still resolution: 8 Megapixels
- Sensor resolution: 3280 x 2464 pixels
- Sensor size: 1/4 inch
- Pixel size: $1.12 \times 1.12 \mu\text{m}$
- Lens: fixed-focus
- Field of view: 120°
- Aperture: F/2.2
- Video modes: 1080p 30fps, 720p 60fps

I paketet inkluderas:

- 1 x Camera
- 1 x 15cm Ribbon cable for Raspberry Pi 4B / 3B+ / 3B / 3A+ / 2B / 1B+ / 1A+ and NVIDIA Jetson Nano
- 1 x 15cm Ribbon cable for Raspberry Pi 5 / Zero W / Zero
- 1 x Adjustable holder (Needs simply assembled)
- 1 x Screwdriver [\[3\]](#)

För att ansluta kameramodulen till Raspberry Pi 5 krävs ett kameraband. Det ska fästas i både kameran och i Raspberry Pi 5 enhetens CAM/DPI ingång.

I detta fall krävdes ytterligare konfiguration i systemets "config.txt" för att manuellt aktivera kameran. Konfigurationen kan se annorlunda ut beroende på leverantör, och i detta fall ser det ut så här:

```
Run the command to open and edit the configuration file.
sudo nano /boot/config.txt
Add dtoverlay=imx219 to the final line of the /boot/config.txt. Press Ctrl+O, Enter to save the change and
Ctrl+Z to exit.
dtoverlay=imx219

pi@raspberrypi: ~
File Edit Tabs Help
GNU nano 7.2 /boot/config.txt *
disable_overscan=1

# Run as fast as firmware / board allows
arm_boost=1

[cm4]
# Enable host mode on the 2711 built-in XHCI USB controller.
# This line should be removed if the legacy DWC2 controller is required
# (e.g. for USB device mode) or if USB support is not required.
otg_mode=1

[all]

#dtoverlay=ov5647
dtoverlay=imx219

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute  ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify  ^_ Go To Line

Reboot your raspberry pi.
sudo reboot
```

Länk till Freenove guide till hur man installerar kameramodulen finns här: [\[4\]](#)

Utvecklingsverktyg

Python

Python som utvecklingsspråk valdes av en rad olika anledningar:

- Förvärva ny kunskap i ett programmeringsspråk aldrig tidigare använt.
- Många officiella bibliotek samt tredjepartsbibliotek som tillhandahåller kod för att på ett smidigt sätt komma igång med utförande av projekt.
- Hög kompatibilitet med Raspberry Pi 5, då Python3 är förinstallerat på Raspberry Pi OS. Då detta projekt använder tredjepartsbibliotek är det nödvändigt att skapa en virtuell utvecklingsmiljö i Raspberry Pi 5. Detta förhindrar tredjepartsbibliotek från att "störa" den globala Python-installationen.

Exempel för att skapa en virtuell utvecklingsmiljö som heter "rpi-env" i kronologisk ordning finnes nedan:

1) `mkdir rpi-env`

Förklaring: Skapar en ny mapp med namn rpi-env i din nuvarande path

2) `cd rpi-env`

Förklaring: Går in i mappen rpi-env

3) `python -m venv - --system-site-packages rpi-env`

Förklaring: Skapar en virtuell miljö, vid namn rpi-env, med de globala Python3 paketen inkluderade

4) `source rpi-env/bin/activate`

Förklaring: Aktiverar den virtuella miljön i ditt system. [\[5\]](#)

PocketSphinx - LiveSpeech

PocketSphinx laddas ned med tillhörande tränad standardmodell samt ordförråd och är open-source. Modellen baseras på US-Engelska och kan därför fungera tveksamt när man som icke-amerikansk ska aktivera modellen genom ett valt ord.

I kombination med PocketShpinx, och för att den ska fungera, krävs ytterligare två bibliotek:

- ALSA Development Kit - Advanced Linux Sound Architecture tillhandahåller ett API, till PortAudio, som ger tillgång till Raspberry Pi's ljudkort. [\[6\]](#).
Detta installeras via terminalfönster på följande sätt:
`sudo apt-get install libasound-dev` [\[7\]](#)
- PortAudio - Används av PocketSphinx för avlyssning av mikrofon i realtid.
Extrahering av `pa_stable_v190700_20210406.tgz`-fil krävs och i detta projekt har den extraherats inuti den virtuella miljön.
[\[8\]](#)
- Kodexempel nedan visar hur LiveSpeech har konfigurerats för att lyssna efter "keyphrase" ("help") samt graden av känslighet den ska ha. "kws_threshold=1e-20" tyder på en hög grad av känslighet:
- Kodexemplet är taget från den officiella hemsidan för att sedan modifieras:
[\[9\]](#) och [\[10\]](#)

```
def listen_for_keyphrase(self) -> bool:
    """The microphone is listening for the chosen keyphrase. If caught, it prints out the keyphrase to the console.

    Raises:
        Exception: If any errors occurs during speech recognition.

    Returns:
        bool: Returns True if the keyphrase is caught.
    """

    try:
        # kws_threshold used for keyphrase detection accuracy
        speech = LiveSpeech(keyphrase=self.keyphrase, kws_threshold=1e-20)
        for phrase in speech:
            print(f'{phrase} keyphrase detected!')
            # If keyphrase is detected, return True to let script in '__main__' start
            return True
    except Exception as e:
        print(f'Error occured in listen_for_keyphrase: {e}')
```

Picamera2

Picamera2 är Raspberry Pi's egna Python-bibliotek och ämnar sig för de Raspberry Pi modeller som använder Raspberry Pi OS Bullseye eller senare version samt kameramoduler med tillhörande kameraband, men har också ett begränsat stöd för usb-kameror.

Picamera2 bygger på open-source biblioteket "libcamera", som har stöd för komplexa kamerasystem i Linux.

Följande två importeringar har använts i samband med Picamera2:

H264Encoder: Ansvarar för komprimering av video och ger stöd för att spela in i upp till 1080p30 fps(frames per second).

FfmpegOutput: Möjliggör skapandet av video i **.mp4** format. Detta används i programmet, då VLC just nu har problem med uppspelning av **.h264** format. Ger dessutom stöd åt ljud i samband med videoinspelningarna.

Koden i projektet baseras på exempel och information hittad i den officiella dokumentationen som finns här: [\[11\]](#)

Azure IoT SDK

Azure IoT SDK möjliggör användandet av kod för att etablera kommunikation mellan IoT-enheter (Raspberry Pi 5) och Microsoft Azure IoT Hub samt Blob Storage.

Detta möjliggör automatisk etablering av kontakt, skapande av SAS och uppladdning av video och bild tagna av IoT-enheten (Raspberry Pi 5) till tillhörande Blob Storage.

[\[12\]](#)

En SAS-url (Shared Access Signature) kan göra det möjligt för tredje part att komma åt de inspelningar och bilder som ligger lagrade i en Blob Storage container på ett säkert sätt.

[\[13\]](#)

Projektet har använt sig av exempelkod från den officiella dokumentationen men gjorts om för att passa in i programmets struktur. I dagsläget inte någon bra lösning för att tillhandahålla tredjepart med bilder och video, utan kan endast laddas ner direkt från Blob Storage, vilket inte är optimalt.

Datahantering och lagring

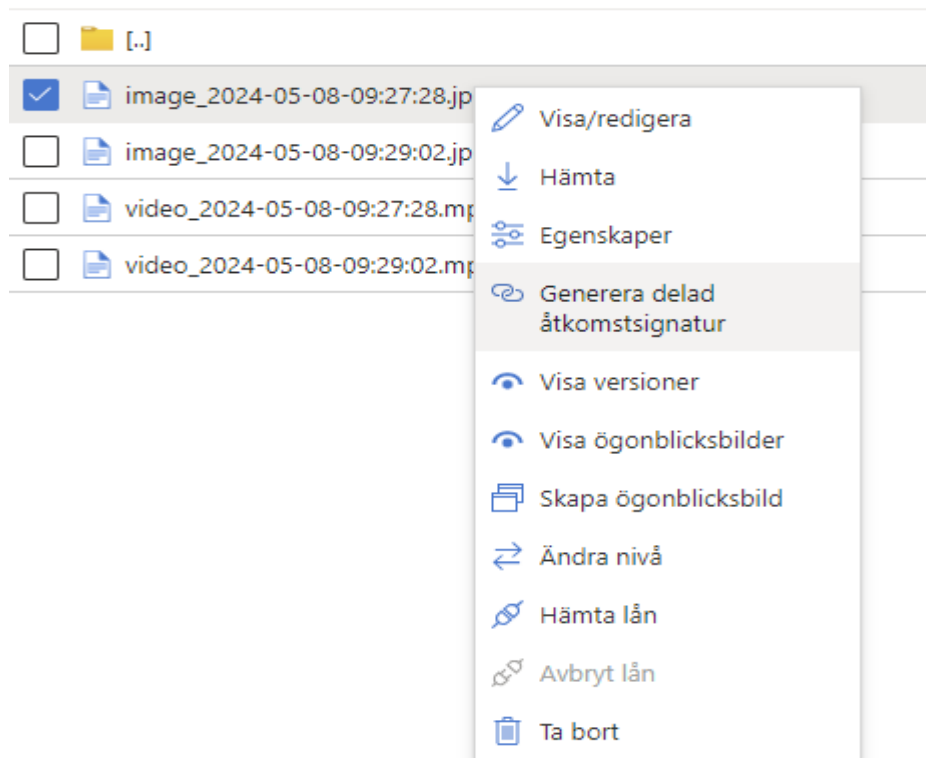
Azure Blob Storage

Nedan visas ett exempel på hur det ser ut efter lyckad uppladdning från Raspberry Pi 5 IoT enhet till en Blob Storage container:

The screenshot shows the Microsoft Azure portal interface for a Blob Storage container named 'rp1container'. The top navigation bar includes 'Microsoft Azure' and 'Uppgradera'. A search bar is present on the right. The left sidebar shows the container name and a search bar. The main content area displays the container's details, including the authentication method (Access Key) and the location (rp1container / rpi-emergency-assistant). A table lists the uploaded files, showing their names and the time they were last modified.

Namn	Har ändrats
<input type="checkbox"/> [.]	
<input type="checkbox"/> image_2024-05-08-09:27:28.jpg	2024-05-08 09:27:42
<input type="checkbox"/> image_2024-05-08-09:29:02.jpg	2024-05-08 09:29:38
<input type="checkbox"/> video_2024-05-08-09:27:28.mp4	2024-05-08 09:27:37
<input type="checkbox"/> video_2024-05-08-09:29:02.mp4	2024-05-08 09:29:34


Önskar man ge ut en SAS-url, kan detta göras genom att högerklicka på önskad uppladdning och välja **Generera delad åtkomstsignatur**:



Följande ruta öppnas och tillhandahåller vilken **behörighet** som ska ges, **start- och slutdatum/tid** för SAS-url, för ökad säkerhet vid delning av data till tredjepart:

rpi-emergency-assistant/image_2024-05-08-09:27:28.j

Blob

 Spara  Ta bort  Hämta  Uppdatera  Ta bort

Översikt Versioner Ögonblicksbilder Redigera Generera delad åtkomstsignatur

En signatur för delad åtkomst (SAS) är en URI som ger begränsad åtkomst till en Azure Storage blob, i lagringskontonnyckel. [Lär dig mer om att skapa ett SAS-konto](#)

Signeringsmetod

☒ Kontonnyckel ☐ Användardelegeringsnyckel

Signeringsnyckel ⓘ

Nyckel 1 ▼

Princip för lagrad åtkomst

Inget ▼

Behörigheter * ⓘ

Läs ▼

Start- och slutdatum/tid ⓘ

Starta

05/08/2024  09:35:52

(UTC+01:00) Amsterdam, Berlin, Bern, Rom, Stockholm, Wien

Förfallodatum

05/08/2024  17:35:52

(UTC+01:00) Amsterdam, Berlin, Bern, Rom, Stockholm, Wien

Tillåtna IP-adresser ⓘ

till exempel 168.1.5.65 eller 168.1.5.65-168....

Tillåtna protokoll ⓘ

☒ Endast HTTPS ☐ HTTPS och HTTP

Skapa SAS-token och webbadress

Efter att ha valt **“Skapa SAS-token och webbadress”** dyker en ruta upp med en unik webbadress för den valda bilden:

Skapa SAS-token och webbadress

SAS-token för blob ⓘ
sp=r&st=2024-05-08T07:35:52Z&se=2024-05-08T15:35:52Z&spr=https&sv=2022-11-02&sr=b&sig=IhtpN9N8MR9xOrRthCqQa7UHdELH3dH0WyOZVbU2jo8%3D ⓘ

Webbadress för blob-SAS
https://rp1storage.blob.core.windows.net/rp1container/rpi-emergency-assistant/image_2024-05-08-09%3A27%3A28.jpg?sp=r&st=2024-05-08T07:35:52Z&se=2024-05-08T15:35:52Z&spr=https&sv=2022-11-02&sr... ⓘ

Arkitektur och design

Klass- och modulstruktur

Klasser och moduler har delats upp i programmet för ökad läsbarhet och modularitet. Detta bidrar till att programmet kan vidareutvecklas vid ett senare tillfälle med den struktur som redan finns.

Nedan följer en kort beskrivning om varje klass och modul och vad de gör:

- **azure_module.py** - Här finns kod som möjliggör kommunikation mellan Raspberry Pi 5 IoT-enhet, Azure IoT Hub och Blob Storage.
- **rpi_camera.py** - Klass är avsedd för hantering av Picamera2 biblioteket genom uppdelade metoder som hanterar video konfiguration, videoinspelning och tagning av bilder.
- **rpi_microphone.py** - Klass är avsedd för hantering av PocketShpnix LiveSpeech biblioteket genom en metod som möjliggör för mikrofon att lyssna efter en specifik “keyphrase” (“help”). Metoden returnerar en True boolean när “help” har registrerats och används för att starta flödet av programmet.
- **output_filename_module.py** - Ansvarar över att ge unika namn till bilder och videoinspelningar. Namnen ges baserat på filens prefix, alltså “video” eller “image” och baserat på detta, ges ett datum då filen skapades och med rätt format, t.ex “image_2024-05-08-09:27:28.jpg”. Val av namngivelse är till för tredjepart och hur denna lätt skall förstå när en specifik nödsituation har uppstått.
- **rpi_system_module.py** - Tillhandahåller metoder för hantering av filer som skapats och metod för att starta om python-skriptet om fel uppstår.
- **main.py** - Här finns variabler som håller strängvärden, såsom vilken mikrofon systemet använder sig av, vilken keyphrase som användaren skall säga, prefix för video och image samt skapande av objekt av typen Cam, Mic och BlobService.
Här finns även metoden som anropas för att starta skriptet och utföra alla nödvändiga steg från det att mikrofonen börjar lyssna till att filer har laddats upp till en Blob Storage container.

Ett övergripande flöde av programmet demonstreras nedan:

- Skript körs och talar om programmet att lyssna efter keyphrase “help”.
- Användare ropar “help” och en videoinspelning på 10 sekunder med ljud sätts igång.
- Unik namn på video har tilldelats och skickas nu upp till Blob Storage
- En bild över situationen tas.
- Unik namn på image har tilldelats och skickas nu upp till Blob Storage
- Video- och bildfiler flyttas från root map till respektive mappar för senare användning.

Referenser

[2] -

https://www.amazon.com/SunFounder-Microphone-Raspberry-Recognition-Software/dp/B01KLRBHGM/ref=sr_1_2?dib=eyJ2IjoiMSJ9.s1LC_Mfh6rAvRyOYgUIC9jONcBm5fbq0eLKou_Iz5c6wjOdC7klHg8x4BVRG8CWK5YJureJGbp2IdOT_tRLQMjdNjrl8rTElVlOazpIGW4bglu7uSKPtK-71ElsRGQa1KPJIQWRcJWGR2XQgjlW8a5_XfDgCwXv-1Y5G7WJdz-K709X3WUJlrsC9KIm05ODNpz7-OZEWiTzDxzpTO2hMCG8wp5zkzAIM-xX-vWgxQbE.54Is2CJFYWoIW3mEg3SE17uPLdVRC3M13_qPacEHE4&dib_tag=se&keywords=raspberry%2Bpi%2Bmicrophone&qid=1709283550&sr=8-2&th=1

[3] -

https://www.amazon.se/Freenove-Camera-Raspberry-Adjustable-Viewing/dp/B08Q34FKFY/ref=sr_1_20?crid=2S89REB28AAUK&dib=eyJ2IjoiMSJ9.LLgDGsTJMKZ5AZeN9FM-qlhsZRWA0lCRKOnwIwkZQ3Nx4MzB7TtH0LIf_gMtlJzeLl3c1gpXwRMTBrovDcvMpKinJuU5TVIY0uupNblmu93Prgnwpy4YC2irHpEFwCJmoR7B73gXGC4QOdfvgYNNbEY7hpsJCQNexGjIAQ1AqX-0UpnukpStZlZLHBE0cyj1MeD0qZwKP6Cyns7iUhQhajrBerk8ggZDryiuQh3GypZVQFtzT4MmZBiKLsxgeZfi2ykSFGZTHf29FOXRbj24sHaBjJA1eVoO5Sx6amIFS_c.1-e4UqZSL1SIYF7Q7E7HSCaO92GbSPabhRh_vhlT2f4&dib_tag=se&keywords=kamera%2Braspberry%2Bpi%2B5&qid=1710429409&srefix=kamera%2Braspberry%2Bpi%2B5%2Caps%2C151&sr=8-20&th=1

[6] - <https://wiki.debian.org/ALSA>

Officiella Dokumentationer

[1] - <https://www.raspberrypi.com/documentation/computers/os.html>

[4] -

https://github.com/Freenove/Freenove_Camera_Module_for_Raspberry_Pi/blob/master/Tutorial.pdf

[5] -

<https://www.raspberrypi.com/documentation/computers/os.html#using-pip-with-virtual-environments>

[7] - https://portaudio.com/docs/v19-doxydocs/compile_linux.html

[8] - <https://files.portaudio.com/download.html>

[9] - <https://pocketsphinx.readthedocs.io/en/stable/#apidoc>

[10] - https://pocketsphinx.readthedocs.io/en/latest/config_params.html

[11] - <https://datasheets.raspberrypi.com/camera/picamera2-manual.pdf>

[12] - <https://learn.microsoft.com/en-us/azure/iot-hub/file-upload-python>

[13] - <https://learn.microsoft.com/en-us/azure/storage/common/storage-sas-overview>

