Manipulasjon og overvåkning (av planter)

Eirk V. Norheim

April 6, 2025

1 Introduksjon

Eg satt meg ned og ønskja å laga eit system som kunne gjære tre ting

- Fotografere mine plantar, hvert 15. minutt, og lagre bilda.
- Sende disse som liveoppdateringe til ein stad eg kan sjå dem, sjølv om eg er ein helt anna stad.
- Lage ein timelapse av bilda i ettertid

For ett system som skal håndtere alt ditte, valgte eg å bruke ein Raspberry Pi, kobla opp med Raspberry pi sin Camera Module 2.

2 Fotografering

Steg 1 er nok det enklaste, etter å ha installert pakken Picamera2 treng man egentlig bare å skrive eit kort python-script for å få dette til å virke. Heilt fritt for ingeniørverdige ablegøyer og smartheter var det likevel ikkje. I figur **??** kan du sjå koden, eg skal og gå gjennom og forklare kva han gjær.

Koden består av 5 delar

- Først importerar den nødvendige modular
- Deretter definerar og kjøyrer den en funksjon $drepexistinginstances()$, som skal motvirke ein bug i systemet eg oppdaga, at dersom Raspberry Pi en kræsja, ville den ikkje lukke kameraet, og det skapte trøbbel når PI-en blei starta igjen. Mesteparten av den koden har eg ikkje skreve sjølv, men fant i ein blogg.
- Deretter definerer koden to filbaner, en til mappa kor bilda som blir tatt skal bli lagra, og ein filbane til ett dokument som heter $piccounterfile$, som lagrer kor monge bilder vi har tatt. Koden sjekkar og om disse filene faktisk eksistera, og lagar dem hvis ikkje.
- Det neste koden gjer er å opne $piccounterfile$ og lese frå den kor monge bilder vi allereie har tatt, før det øker dette tallet med ein, og skriv dette tilbake i fila. Grunnen til at vi bryr oss om kor monge gangar vi har kjørt fila er for å kunne namngi bilda vi tar i kronologisk rekkefølge.
- Den siste delen av koden er den som faktisk tar bilda, og er nokså enkel

```python
from picamera2 import Picamera2
import os
import sys
import psutil

def drep_existing_instances():
    current_pid = os.getpid()
    script_name = os.path.basename(__file__)  #Henta koden for å slette eksisterende instances fra ett standardprosjekt på en RPI-blogg
#Dette på grunn av en bug som oppsto hvis RPI kræsja, som gjorde at den ikke kjørte picam.close() nederst i koden
    for process in psutil.process_iter(['pid', 'name', 'cmdline']):
        try:
            if process.info['pid'] != current_pid and process.info['cmdline']:
                # Check if the script name appears in the command line arguments
                if script_name in " ".join(process.info['cmdline']):
                    print(f"Stopping existing instance: PID {process.info['pid']}")
                    os.kill(process.info['pid'], 9)  # Force kill the process
        except (psutil.NoSuchProcess, psutil.AccessDenied, psutil.ZombieProcess):
            continue

drep_existing_instances()

#Alt etter dette er derimot skrevet for egen maskin

base_dir = "/home/eirik/Desktop/Planteprosjekt/regular pics"
pic_counter_file = "/home/eirik/Desktop/Planteprosjekt/pic_counter.txt"

os.makedirs(base_dir, exist_ok=True)

if os.path.exists(pic_counter_file):
    with open(pic_counter_file, "r") as f:
        try:
            pic_count = int(f.read().strip())
        except ValueError:
            pic_count = 0  # Hvis filen er tom eller inneholder feil verdi
else:
    pic_count = 0

picam2 = Picamera2()
filename= f"/home/eirik/Desktop/Planteprosjekt/regular pics/New/plant{pic_count}.jpg"
secondFilename = f"/home/eirik/Desktop/Planteprosjekt/regular pics/temp/plant{pic_count}.jpg"
picam2.start_and_capture_file(filename)
picam2.start_and_capture_file(secondFilename)
pic_count+=1
with open(pic_counter_file, "w") as f:
    f.write(str(pic_count))
picam2.close()
```

Figure 1: Kode for bildetaking, skrevet i Python, kan og sees på Github

3 Liveoppdatering

For å få tilgang på desse bileta på farta, har eg valgt å bruka rclone biblioteket i rPi og lasta opp til min egen google drive, det involverar å laga ein token frå google drive, og relativt enkel secureshell kode for å laste opp til denne driven. Men realistisk sett var denne delen stort sett berre å følgje ett par guidar på internettet [1] [2], SSH koden kan du sjå i figur 2

[1] https://pimylifeup.com/raspberry-pi-rclone/
[2] https://rclone.org/drive/

```
#!/bin/bash
|
LOCAL_FOLDER="/home/eirik/Desktop/Planteprosjekt/regular pics"
REMOTE_FOLDER="Planteprosjekt:/Plantebilder"

LATEST_FILE=$(ls -t "$LOCAL_FOLDER"/*.jpg | head -n 1)

if [ -n "$LATEST_FILE" ]; then


    rclone copy "$LATEST_FILE" "$REMOTE_FOLDER"
    echo "Uploaded: $LATEST_FILE to Google Drive!"
else
    echo "No PNG files found in $LOCAL_FOLDER!"
fi
```

Figure 2: Kode for å laste opp bilda til google drive, skrevet i secureshell

4 Ein halvintelligent detalj

En viktig del av heile dette systemet er at det må fungere heile tida, dessverre er RPI noe uforitsigbar, den kan plutselig reboote, eller miste tilkoblinga til internett. I tillegg er det ikkje særlig effektivt å kjøyre kode heile tida. For å løyse dette brukar vi RPI si innebygde fil Crontab. Her skriver vi tre linjer.
$*/15 * * * */filbane/Bildetager.py$ Programmet kjører og tar bilder hvert 15. minutt

$*/5 * * * */filbane/reconnect\_wifi.sh$ Denne linjen sørger for å opprettholde wifitilkoblingen døgnet rundt

$*/15 * * * */filbane/upload\_pics.sh$ Denne laster opp det nyeste bildet hvert 15. minutt.

Crontab står for cron table og brukes spesielt på UNIX og Linux OS, til å kjøyre gitte kommandoar ved gitte intervall eller tidspunkt.

5 Timelapse

Siste del av systemet er å kunne laga ein timelapse av det heile når plantane mine er ferdig vokst. Dette gjærast også i Python og brukar cv2, os og glob biblioteka til å slå saman alle bileta i ei mappe til ein video med ein valfri framerate. Du kan sjå koden i figuren under

Koden består av 8 hovuddelar

- importerar dei nødvendige biblioteka
- definerar filbaner for mappa som skal gjøres om til en timelapse
- sorterer bilda etter alder, ettersom systemet med økende tall ikkje blir riktig alfabetisk
- sjekker om det faktisk er bilder i mappa
- bruker det første bildet i mappa til å hente dimensjonane til bilda
- brukar cv2 sin funksjonalitet til å sette opp videoskriveren til å skrive .mp4 fil

```python
import cv2
import os
import glob

# filbaner for hvor jeg vil hente og legge filene
image_folder = "/home/eirik/Desktop/Planteprosjekt/regular_pics/sessionx"
output_video = "/home/eirik/Desktop/Planteprosjekt/timelapse.mp4"

#sorter etter alder
images = sorted(glob.glob(os.path.join(image_folder, "*.jpg")), key=os.path.getmtime)

#Sjekker etter filer i mappen
if not images:
    print("No images found in the folder!")
    exit()

#Bruker det første bildet til å hente dimensjonene til videoen
frame = cv2.imread(images[0])
height, width, layers = frame.shape

fourcc = cv2.VideoWriter_fourcc(*'mp4v')  # Kode for .mp4
video = cv2.VideoWriter(output_video, fourcc, 10, (width, height))#setter opp videoskriveren

# Prosesserer hvert enkelt bilde og legger det til i videoen
for img in images:
    frame = cv2.imread(img)
    video.write(frame)

video.release()#legger videoen i den definerte mappen
print("Suksess")#Sjekk om hele koden har kjørt
```

Figure 3: Kode for å laga ein timelapse av bileta, skrevet i python. Du kan og sjå heile fila i Github

- prosseserar hvert enkelt bilde og legger til i videoen
- når den har lagt til alle bilda legger den videoen i den tidligere definerte mappa og timelapsen er ferdig