

Astro Pi

Noten er en introduktion til Python programmering af en Raspberry Pi med en SenseHat og et kamera tilkoblet. Der sidder to af disse microcomputere ombord på den internationale rumstation, ISS, hvor de kaldes Astro Pi, og det er muligt for elever at lave forsøg på ISS ved at programmere disse Astro Pi. Eleverne kan deltage gennem konkurrencen [mission space-lab](#) og noten skal ses som et supplement til Astro-pi.org's ressourcerum, <https://astro-pi.org/resources/>. På ressource hjemmesiden er det især

- [Mission space-lab guidelines](#)
- [Astro-pi-mission-zero-project.](#)
- [Astro Pi Mission Space Lab Phase 2 guide](#)

som er relevante.

Danske bidrage

- [Elevkode](#)
- [iftek side](#)

Formålet med noten er, at klæde lærere på til at hjælpe eleverne med at udvikle deres egne programmer til deltagelse i rumkonkurrencen. Noten guider elever og lærere gennem, opsætning af Astro Pi, simpel dataopsamling og dataanalyse. Eleverne bliver fortrolige med Python programmering gennem små opgaver, som tager udgangspunkt i kode som de skal modificere. Vi har på den måde prøvet at gøre programmeringen relativt simpel, mens vi udnytter potentialet i Astro Pi computeren. Det kræver derfor ingen forudgående erfaring med Python programmeringssproget.

De to første afsnit, **Virtuel introduktion til Astro Pi** og **Gemme data Astro Pi** kan bruges med elever uden selve Astro Pi computeren, da det hele foregår virtuelt. **Virtuel adgang til Astro Pi** kan ses som en guide til læreren i opsætning, eller til elever med særlig teknisk interesse.

Første gang med Astro Pi er programmering og arbejde direkte på Astro Pi computeren. Dette kræver en Raspberry Pi med tilhørende senseHat. Hvis man går videre fra første rundte sender ESERO en pakke med alt udstyret!!! **Undersøgelse af ukendt land eller hvad sker der i mit køleskab når døren er lukket** er en øvelse hvor eleverne bruger deres færdigheder og Astro Pi computeren til at undersøge et ukendt og ugæstfrit miljø, køleskabet. Det kan bruges som et selvstændigt projekt hvor dataopsamling og videnskabelig metode er centralt. De sidste afsnit, **Brug af kamera**, introducerer kameraet hvor der kan tages billeder af Jorden fra ISS. Det sidste afsnit, **Computer vision med Astro Pi**, er en introduktion til computer vision, hvor eleverne lærer at detekterer jord, hav og rumstation og laver en bevægelsessensor. Det er populært at bruge kameraet i elevprojekter og målet med afsnittet er at gøre det overkomeligt for elever og lærere. Arbejdet er støttet af [ESERO danmark](#).

Python koden som bliver gennemgået i dokumentet kan hentes på, <http://iftek.dk/astro-pi/>.

Rune Klarskov, Mads Peter H. Steenstrup, August 2021.





This work is licensed under a [Creative Commons](#)
[Attribution-NonCommercial-ShareAlike 4.0 International License](#).

| | |
|--|-----------|
| Virtuel introduktion til Astro Pi | 3 |
| Mål | 3 |
| At gemme data | 3 |
| Gemme data Astro Pi | 4 |
| Gamme data som cvs kolonner | 4 |
| Virtuel adgang til Astro Pi | 6 |
| VNC adgang til Astro Pi | 6 |
| Astro Pi på netværket uden skærm og tastatur | 7 |
| Terminalen | 7 |
| Find IP adresse | 8 |
| SSH setup uden skærm, lidt mere avanceret | 8 |
| Log in med SSH og Enable VNC | 8 |
| Første gang med Astro Pi | 8 |
| Tænd og sluk | 9 |
| Første dataopsamling med Astro Pi | 9 |
| Sende data til Astro Pi | 10 |
| VNC viewer | 10 |
| Terminal | 10 |
| Data ud af RPI | 10 |
| Undersøgelse af ukendt land eller hvad sker der i mit køleskab når døren er lukket. | 11 |
| Hvor er ISS | 12 |
| Brug af kamera | 13 |
| programmer | 13 |
| Open CV | 14 |
| Slet billeder | 15 |
| Kameraopløsning | 15 |
| Exif data | 16 |
| Computer vision med Astro Pi | 17 |
| Jord, luft, vand og ja rumstation | 17 |
| HSV og BGR | 18 |
| Life in space | 19 |



Virtuel introduktion til Astro Pi

Der findes en online Astro Pi emulatior, så man kan komme i gang uden selve Astro Pi computeren. Start her: projects.raspberrypi.org/en/projects/getting-started-with-the-sense-hat

Følgende afsnit giver en god indføring. Der er flere eksempler end man har tid og lyst til, men så må man udvælge.

Mål

Målet med disse øvelser er at blive fortrolig med python syntaksen til at betjene vores Astro Pi. Mere specifikke skal I kunne

- Vise tekst og tal på senseHat display.
- Vise måledata på senseHat display.
- Skrive måledata på computerskærmen.
- Gemme måledata i fil, virtuelt med trinket.

På <https://trinket.io> kan man gemme sin kode og køre den virtuelt. De fleste elever brugte bare emulatoren på Astro Pi hjemmesiden, men trinket kan bruges til at dele eksempelkode mm.

Opgave

- Prøv at udforsk hvad senseHat modulet kan, fokuser på de afsnit i røde bokse.

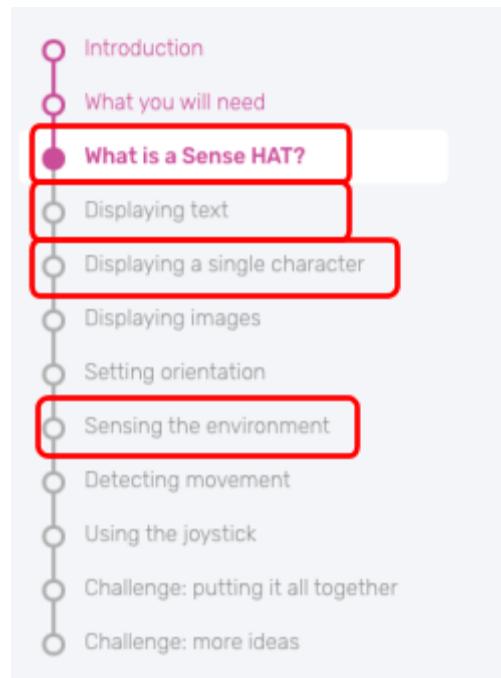
Gemme data, virtuelt

Det er selvfølgeligt vigtigt at kunne gemme sin data. Det er heldigvis ikke så svært.

Den simpleste, [AstroPiSimpleGemData](#)

```
with open ('gemData.csv', 'w') as file:
    file.write('gem denne streng')
    file.close
```

Denne kodestump danner en csv fil og genner den.
with open har



- “w” - write,
- “a” - append,
- “r” - read

Settings, hvor man overskriver med *write*, tilføjer med *append* og læser med *read*.

Opgave

- Kør programmet.
- Lav den om til append og gem det samme en masse gange (kør programmet igen og igen).
- Gem 'gem denne streng \n' i stedet for, hvad er forskellen? (n er new line).
- Som diskuteret i afsnittet dataformater kan I også gemme værdier, på samme måde som i printer. Implementer `file.write('Pi er ca %s \n'%(3.141579))`.

Gamme data som csv kolonner

Hvis vi skal importere data i regneark og lignende er standarden at gemme som kommasepareret fil, csv. Det kan jo godt give problemer for os da vi bruger komme til noget andet og hvis I får problemer med det så brug ; i stedet.

Et lidt større eksempel ([gemDataKolonner.py](#))

```
import datetime
from time import sleep
from sense_hat import SenseHat
sense = SenseHat()

start_time = datetime.datetime.now()
now_time = datetime.datetime.now()
duration = datetime.timedelta(seconds=20)
i = 0

with open ("test.csv", "w") as file:
    file.write("time , temperature , pressure \n")

while now_time < start_time + duration:
    t = sense.get_temperature()
    p = sense.get_pressure()
    now_time= datetime.datetime.now()

    with open ("test.csv", "a") as file:
        file.write("%s, %s, %s \n" % (now_time, t,p))
    sleep(1)
```



Programmet gemmer filen test.csv med overskrifterne time, temperature, pressure. Variablene bliver aflæst i et loop på 20 sekunder og skrives på hver ny linje i linje 20. \n giver ny linje.

Opgave

- Kør koden og kopier csv filen over i et regneark og lav en kurve.
- Forklar forskellen på linjerne `with open ("test.csv", "w") as file:` og `with open ("test.csv", "a") as file:` i programmet.
- Tilføj en måling af fugtigheden med `h = sense.get_humidity()` og gem den ved at ændre på næstnederste linje.

Variablen `i = 0` bruges til at tælle med. Hvis I indsætter `i = i + 1` i while løkken under variablen `now_time`, bliver i en større hver gang programmet kommer forbi den kodedel.

Opgave

- Lav en variabel som tæller op og gem den.

Måden vi laver gentagne målinger er ved at bruge en løkke eller et loop. Her bruger vi et `while` loop som tjekker om betingelsen `now_time < start_time + duration` er sand og køre den indrykkede kodedel igen og igen indtil betingelsen ikke længere er opfyldt. Her er det variablen `now_time` som bliver opdateret og loopet kører indtil tiden er gået.

Her er en simpel implementering af et while loop.

```
from time import sleep
i = 0
while i < 20:
    print(i)
    i = i + 1
    sleep(1)
```

Opgave

- Beskriv i ord hvad de forskellige linjer gør.
- Lav om i koden så `i = i + 2` og beskriv hvad der sker.

Tiden bliver styret af biblioteket `datetime`. Vi kan finde antallet af sekunder siden starten ved at beregne forskelle `interval = now_time - start_time`.

Hvis vi vil have det i sekunder kan vi bruge `interval.total_seconds()`.

Opgave

- Lav om i filen, [gemDataKolonner.py](#), så det er tiden som er gået i sekunder vi gemmer.

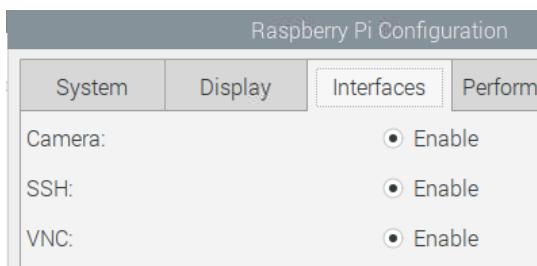


Adgang til din Astro Pi

Det simplest er at sætte tastatur, mus og monitor med HDMI direkte i Astro Pi, hvis man skal klare opsætningen. Når eleverne skal eksperimentere er det smartest med en virtuel adgang, hvor eleverne kan styre Astro Pi direkte fra deres egne computere. Der er en mere udførlig gennemgang i appendiks: opsætning af fjernadgang til Astro Pi.

VNC adgang til Astro Pi

Hvis I har tilsluttet monitor, mus og tastatur er det relativ simpelt at få adgang. I kan bruger Virtual Network Computing (VNC) til at kontrollere jeres Astro Pi fra jeres egen computer. På jeres Astro Pi skal I aktivere den under Preferences->Raspberry Pi Configuration -> Interfaces:



På jeres computer skal I have programmet, realVNC, realvnc.com



På jeres Astro Pi er det et VNC ikon, i højre hjørne, klik på det. Det giver den ip adresse hvor I forbinde til Astro Pi, her 192.168.1.5

Connectivity

192.168.1.5
Connecting users can enter this address in [VNC Viewer](#)

På jeres egen computer i VNC programmet skriver I IP adressen (forklares nedenfor) og navn og kode, så er I inde.



Standard **brugernavn**: pi og **adgangskode**: raspberry . Det kan være en riktig god at skifte den på egen Astro Pi og lave nogle standardiserede for sin klasse.



NB! Det virker generelt bedst hvis man er på samme netværk, computer og Astro Pi. Hvis skolen har sikkerhedsprotokoller kan det være svært at forbinde. Det simpleste er at købe en wifi-router til undervisningen, sørge før at den kan være nok på af gangen.

Fast IP-adresse

For at logge på jeres Astro Pi skal I kende IP-adressen. IP-adresserne er som udgangspunkt dynamiske og bliver tildelt når de logger på routeren. De holder sig ofte et stykke tid men kan godt skifte fra uge til uge.

Løsning 1

Den mest håndholdte løsning er at logge ind som admin på routeren og se listen af Astro Pi computere med tilhørende IP-adresser. Det virker, er simpelt og lidt besværligt.

Løsning 2

Det er muligt at få Astro Pi computeren til at ønske en IP-adresse. Hvis den er fri bliver den tildelt. Det kræver lidt opsætning,

- Find router adresse (her 192.168.1.1): `ip r | grep default` hvilket for mig giver `default via 192.168.1.1 dev wlan0 proto dhcp src 192.168.1.4 metric 303.`
- Updater filen `dhpcd.conf` ved at skrive `sudo nano /etc/dhpcd.conf` i terminalen.
- Generelt skal der indtastes

```
interface wlan0

static ip_address=STATIC_IP/24

static routers=ROUTER_IP

static domain_name_servers=DNS_IP
```

For min PI nr. 7

```
interface wlan0

static ip_address=192.168.1.107/24

static routers=192.168.1.1

static domain_name_servers=192.168.1.1
```

- Gem filen med `ctrl o`.

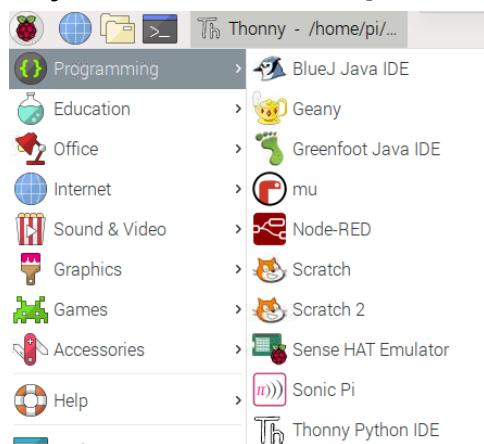


- Reboot og tjek om IP-adressen er den rigtige, 192.168.1.107.

Dette gøres som klart lettest med skærm og tastatur tilsluttet.

Første gang med Astro Pi

Med den virtuelle adgang eller med skærm og tastatur er det nu I skal arbejde med jeres PI. Der er flere editorer, vi bruger **Thonny** som findes under Programming.



Pas på med at have for mange programmer åbne på samme tid, det er en microcomputer med begrænset regnekraft.

Opave

- Åben Thonny og skriv `print('Astro Pi')`
- Gem programmet som `printAstroPi.py`, og lav en mappe som I kan arbejde i.



, undgå som altid æøå, mellemrum og mærkelige tegn.



- Find `printAstroPi.py`, og kør det,

I har nu jeres egen mappe og skal IKKE ændre i andre mapper, vel?

I princippet har I nu en Linux computer som kan langt det meste som I har brug for. Tag et kig i menuen og se hvor mange gratis open source programmer der er og hvor meget en computer til 500 kr kan!

Tænd og sluk

Vores Astro Pi kan godt lide at blive slukket rigtigt, hvilket man gør som man forventer, ingen hints. Hvis den er slukket tænder man den ved at tænde for strømmen, bum computere er nemme.

Første dataopsamling med Astro Pi

Vi kan selvfølgeligt køre programmer på vores Astro Pi ellers var den jo ubrugelig.

Opgave

- Copy-paste programkoden fra (aDataKolonner.py) ind i Thonny editoren og kør programmet. (copy paste er ikke helt stabilt, men prøv et par gange hvis det ikke virker umiddelbart og husk at paste er ctrl v, og ikke cmd v i linux, brug evt. musen og højreklik).
- Det kan være raret at se at programmet kører og stopper så skriv `print('starter')` og `print('slutter')` fornuftige steder i programmet.
- Åben `test.csv` filen inde fra Thonny og tjek at I har fået det data I vil.

I har nu samlet data op med jeres Astro Pi og er faktisk ikke så langt fra at kunne lave eksperimenter på ISS.

Dataopsamling ISS way

I opgaven ovenfor bliver data gemt i samme mappe som `aDataKolonner.py` filen ligger. Vi kører simpelthen programmet fra den samme mappe. Det er ikke sådan programmet bliver kørt på ISS. Her vil jeres program ligge i en mappe og programmet vil så bliver kørt centralt. Vi har stadigt brug for at vores data kommer til at ligge i den samme mappe som jeres kodelfil. Til det formål bruger vi biblioteket

```
from pathlib import Path
```

og finder stien til mappen, på engelsk Path ved,

```
dir_path = Path(__file__).parent.resolve()
```

Hvis vi nu printer `dir_path` får vi den absolute adresse på den mappe filen ligger i.

Hvis I laver en fil med det her i kan I finde ud af hvilken mappe I er i,

```
from pathlib import Path
dir_path = Path(__file__).parent.resolve()
print(dir_path)
```

Vidste I det?



Opgave

Vi vil lave om i filen gemDataKolonner.py så du er sikker på at stien er den rigtige.

- Tilføj `from pathlib import Path` og
`dir_path = Path(__file__).parent.resolve()` til filen
- Lav om i koden hvor der står `with open (filename as w)` så der står,
`with open (str(dir_path)+"/test_path.csv" as w) as file.`
- Gør det samme der hvor data bliver skrevet.
- Som altid kør filen, nogen fejl? Gemmer den data?
- Prøv at skift folder og kør programmet, virker det stadigt?

Her er en kode som virker, `gemDataKolonnerPath.py`

SenseHat cheat sheet

Vigtigt at finde et cheat sheet, heldigvis er det her, [senseHat_cheat_sheet](#).

Opgave

- Brug cheat sheet til at finde andre sensorer og prøv jer frem.
- Få det målte vist på senseHat displayet.

Sende data til Astro Pi

Den simpleste måde er nok standard copy-paste, hvor Astro Pi bruger `ctrl` og ikke `cmd` tasten.
Hvis det ikke virker kan man bruge VNC viewer programmet.

VNC viewer

I midten toppen er der en lille menu hvor man kan sende filer til Astro Pi computeren. Den er lidt hemmelig, men man kan godt finde den.



Terminal

Ligesom vi kan få adgang til vores Astro Pi gennem ssh, kan vi også sende filer med scp.

Det kan gøres med scp sådan i en terminal

```
scp fil brugernavn@ipAdresse:
```

eks.

```
scp sort.jpg pi@192.168.1.99:
```

Derefter kodeord

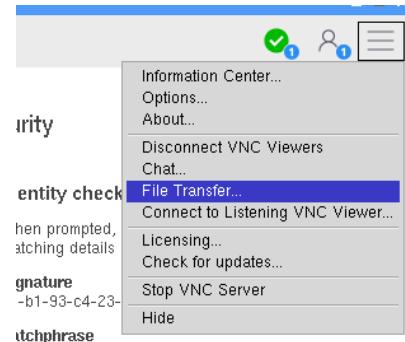
Data ud af RPI

Med VNC kan man overføre filer og mapper, **godt for backup, backup er vigtigt!!!**

- Med Raspbian, styresystemet, klik på VNC

symbolet 

- Vælg File Transfer fra drop down menuen i højre hjørne.
- Find filen og overfør den.



Det kan også gøres med terminalen på jeres lokale maskine og scp:

Eks. hvis jeg vil flytte filen test.csv filen fra Astro Pi til min egen computer

```
scp pi@192.168.1.99:/home/pi/mp/test.csv /Users/mpsteenstrup
```

altså

```
scp pi@ipAdresse:/stiTilFil stiTilMappe
```

Her har jeg lavet mappen mp og kørt programmet (gemDataKolonner.py) fra denne mappe.

Opgave

- Overfør datafilen til computren og åben den, eks. i LoggerPro.
- Undersøg hvor meget filen fylder og overvej om det kan give problemer i et 3 timer langt forsøg med maksimal 3Gb til rådighed.

Undersøgelse af ukendt land eller hvad sker der i mit køleskab når døren er lukket?

Selvom målet er at lave forsøg i vægtløs tilstand, ombord på den internationale rumstation, kan vi også bruge vores Astro Pi til at lave forsøg her på Jorden. Det er en central del når eleverne udvikler programmer, at de først udfører test hervede, før koden bliver sendt op. En afprøvning ad måleudstyret er essentiel for et vellykket eksperiment, især når det ikke bare lige kan gøres om.

Nedenstående side indeholder et forsøg hvor temperaturen og trykket bliver målt inde i et køleskab. Eksperimentet vil kunne køre i et modul, 90 minutter, efter at eleverne er blevet fortrolig med at måle og gemme data på Astro Pi computeren.

Vi opbygger det som en klassisk fysikrapport og der er fokus på analyse af fejlkilder, kalibrering af måleudstyr og simpel databehandling. Projektet er relativt styret, men det er muligt at udvide med elevernes egne forslag til målinger af, magnetfelter, luftfugtighed, længere tid osv.

Formål

Formålet er at undersøge tryk og temperatur i et køleskab lige efter det er lukket.

Teori

Vi har observeret at et køleskab binder lige efter det er lukket. Ud fra idealgasligningen ved vi at der er en sammenhæng mellem tryk og temperatur, hvor trykket er proportionalt med temperaturen. Et fald i temperatur vil altså medføre et trykfald. Den varme luft som bliver lukket ind i køleskabet bliver afkölet og trykket falder. Efter et stykke tid vil trykket udligne sig, da køleskabet ikke er helt tæt.

Apparatur

Raspberry Pi, senseHat, power bank.

Programmet

Hvis vi er heldige så kan wifi signale godt trænge ind i køleskabet og vi har fuld kontrol over vores Astro Pi gennem hele eksperimentet. Programmet vi skal bruge ligner til forveksling programmet, `gemDataKolonne.py`, med variablene time, temperature, pressure.

For at holde det adskilt laver vi en ny kodelinje.

`koeleskab.py`

```
import datetime
from time import sleep
from sense_hat import SenseHat
sense = SenseHat()

start_time = datetime.datetime.now()
```



```

now_time = datetime.datetime.now()
duration = datetime.timedelta(seconds=180)

with open ("test.csv", "w") as file:
    file.write("time , temperature , pressure \n")

while now_time < start_time + duration:
    t = sense.get_temperature()
    p = sense.get_pressure()
    now_time= datetime.datetime.now()

    with open ("koeleskab.csv", "a") as file:
        file.write("%s, %s, %s \n" % (now_time, t,p))
        sleep(0.1)

```

Udførelse

Placer jeres Astro Pi i et køleskab og start programmet. Åben køleskabet og kig efter noget lækkert. Luk det igen og vent til programmet har kørt færdigt, spis eventuelt det lækre I fandt.

Databehandling

Databehandlingen foregår som altid med at lave grafer her en over temperaturen og en over trykket som funktion af tiden, (t, temp) og (t, tryk). Det kan være en ide at lave tidsmålingern om til antal sekunder efter start i stedet for now_time, se ovenfor hvordan I gør.

Opgave

- Udfør forsøget og lav databehandlingen.
- Overvej hvorvidt jeres resultater giver mening.

Hvis jere målinger ligner mine så giver jeres trykmåling fine resultater mens temperaturmålingerne er helt ved siden af. Lad os arbejde lidt med temperurmåleren, selvom det ikke bliver helt godt.

Kalibrering af måleudstyr

Vores Raspberry Pi computer med senseHat er ret kompakt, hvilket ofte er fedt. Når vi skal måle temperatur er det bare ikke så smart, at have sensoren siddende lige oven på en CPU som de fleste ved bliver varm. Det er snarere reglen end undtagelsens at måleudstyr skal kalibreres så lad os prøve det. Kalibreringen foregår ved at måle den faktiske temperatur og den målte temperatur på vores Astro Pi og lave en graf med de to målinger en ud af hver akse.

Opgave

- Kør programmet i hhv. køleskabet og i klasselokalet og mål temperaturen med et normalt velfungerende termometer.
- I har nu to datapunkter og kan jo lave en hypotetisk sammenhæng, lineær er den simpleste og simpelt er godt når man ikke ved bedre, vi ved ikke bedre.



- Udtænk en måde at få flere datapunkter så I kan teste den lineære sammenhæng (lad vær med at putte jeres Astro Pi i ovnen, fryseren, tørretumbleren, vand eller noget andet frollet).
- Det er ikke sikkert at der er en flot funktionel sammenhæng, lineær, eksponentiel, potens, eller lignende. Som I ved fra matematik kan man bruge polynomier til at komme tæt på. Gør det og vurder inden for hvilket område I er rimeligt sikre på jeres kalibrering.

Når I har kalibreringsfunktionen er det rimeligt let at implementerer den i koden. Her skal I være opmærksomme på at Python bruger `**` for potensopløsning, eks er x i anden er $x**2$. Hvis kalibreringskurven eks. er givet ved $y = 1.2x - 14$, vil den rigtige temperatur kunne findes ved.

```
t = 1.2*t-14.
```

- Implementer jeres kalibrering af temperaturen.

I har nu en fint kalibreret temperatormåler, men kan I overhovedet måle ændring når køleskabet åbnes og lukkes? Jeg kunne ikke da jeg prøvede, men jeg brugte heller ikke et af de vigtigste redskaber i værktøjskassen, **gennemsnit**.

Simpel kode til at tage gennemsnittet af temperaturen over ti målinger

```
i = 0
t = 0
while i<10:
    t = t + sense.get_temperature()
    i = i + 1
t = t/10
```

Opgave

- Implementer gennemsnit og se om I kan få temperaturmålingerne til at blive mere konsistente.
- Overvej hvor lang tid det giver mening at tage gennemsnit over, I har data fra trykket til at hjælpe jer.

Konklusion

Hvad kan I faktisk konkludere ud fra jeres undersøgelser, om hvad der sker i jeres køleskab?

Perspektivering

Det er aldrig rart at ens udstyr ikke virker 100% som man gerne vil, men trods alt bedre at finde ud af her på Jorden end på ISS eller Månen eller Mars. Når I skal designe eksperimenter er det vigtigt at undersøge.

- Kan jeres Astro Pi faktisk måle det I gerne vil, eks. temperatur.
- Kan den måle det med en præcision som giver mig noget håb om at svare på spørgsmålet.



Det sidste kan være svært at svare på og I må ikke være for kritiske for så risikerer I at intet kan lade sig gøre. Det er også muligt at lave sit eksperiment om så man bruger andre variable, eks. ved vi at tryk og temperatur er korrelerede, så måske kan en troværdig trykmåling bruges i stedet. Det kan også være at det er ændringer i stedet for absolutte værdier I er interesserende. Hvor meget ændrer det magnetiske felt sig rundt om Jorden er ofte nemmere at måle end hvor stort den absolute magnetisme er om bord på ISS. Det kræver med andre ord fantasi og forståelse for måleinstrumenter at lave forsøg, hvem havde troet det ;-).

Hvor er ISS

ISS bruger ca. 90 minutter til at komme en hel gang rundt om Jorden, ca. et fysikmodul!!!! Som <http://www.iss tracker.com/> viser bevæger ISS sig ikke over samme område hver omgang. Det er ret let at finde den nuværende placering af ISS med koden (getLocation.py)

```
from orbit import ISS
location = ISS.coordinates()
print(location)
```

Python er et smukt programmeringssprog med rigtigt mange brugbare biblioteker og `orbit` er et af dem.

Resultatet 2022-01-02:9.47 er
IERS2010 latitude +51.6927 N longitude 105.3626 E elevation 418943.1 m

at google maps kan forstå det NB I får selvfølgeligt andre resultater da det er en anden dag.
Koordinaterne kan kopieres direkte ind i google maps, 51.6927, 122.3626.

[google maps koordinater](#)

Interessant at se her om det er nat det sted på jorden:

<https://www.timeanddate.com/worldclock/sunearth.html>

Vi kan lave det om til breddegrader og længdegrader så det kan bruge til beregninger.

```
from orbit import ISS
location = ISS.coordinates() # Equivalent to
ISS.at(timescale.now()).subpoint()
print(location)

print(f"breddegrader: {location.latitude.degrees}")
print(f"længdegrader: {location.longitude.degrees}")
```



```
if location.latitude.degrees>0:  
    print("nordlige halvkugle")  
else:  
    print("sydlige halvkugle")
```

Opgave

- Find ISS position og tjek med isstracker og google maps.
- Skriv et program som kun tager data når ISS er på den sydlige halvkugle.
- Skriv et program som hvor I er sikre på at tage data over Afrika, men ikke vil tage data noget andet sted.

Brug af kamera

Der er to kameraer på ISS, ét som peger ned mod Jorden og ét som peger ind i rumstationen.

Billeder fra rumstationen på Flickr, [Billeder normalt kamera](#), [Billeder IR filter](#).

God beskrivelse på [Astro Pi-pictures](#).

Camera documentation, [Camera dokumentation](#).

Stil skarpt

Med det nye HD kamera kommer der også objektiver. For at kunne stille skarpt bruger vi preview.

```
import time  
import picamera  
camera = picamera.PiCamera()  
camera.start_preview()  
time.sleep(100)  
camera.stop_preview()  
camera.close()
```

Programmet viser preview i 100 sekunder. Når I har indstillet fokus kan I stoppe med **ctrl-c**.



Tag billeder med programmering.

Programmerne bruger biblioteket `PiCamera` og virker derfor kun på en Raspberry Pi. Vi kommer gennem programmerne gennem opgaverne nedenunder.

- `firstPictures.py` simpelt program til at gemme fire billeder
- `image.py`, bruger `openCV` til at behandle billedet og gemme det.
- `video.py`, bruger `openCV` til at behandle og gemme video.
- `saveDateinformation.py`, importer og gem dato og tidspunkt.

Opgave - Første billeder

- Kør programmet, `firstPicture.py` og åben folderen og se dine billeder.
- Kør programmet igen og tjek om den overskriver de første billeder.
- Læs kommentarerne i programfilen.
- Ret `('img%d.jpg'%i)` til `('img%04d.jpg'%i)`. Hvad er forskellen?
- Find ud af hvordan man laver om på opløsningen ved at kigge i [dokumentationen](#).

Opgave - Fotoautomat

Vores helt egen fotoautomat, tager udgangspunkt i programmet (`takePicture.py`). Husk at test hver gang I laver noget om, så fanger I lettere fejl.

- Skriv en besked i konsolen om at billedet tages, `print('NU')`.
- Ret i koden så pausen på 2 sekunder er efter der bliver skrevet `print('NU')`.
- Nu er det bare at skrive pauser og tekst, så har I lavet en nedtælling og en rigtig fotoautomat.

Billeder som data

Billederne består af RGB værdien for hver ‘pixel’ med værdier mellem 0 og 255. Vi kan bruge billederne som data og lave databehandling på dem. Det kan være interessant i forhold til billedgenkeldelse, bevægelsessensorer eller som starten på machine learning.

I programmet (`image.py`) bliver data fra kameraet indlæst som en liste, Array. I programmet udvælges 3. kolonne som er den blå farve.

```
from picamera.array import PiRGBArray
from picamera import PiCamera
from time import sleep
import matplotlib.pyplot as plt
import numpy as np

with PiCamera() as camera:
    rawCapture = PiRGBArray(camera)
```



```

# giver kameraet tid til at indstille sig
sleep(1)
camera.capture(rawCapture, format='rgb')
image = rawCapture.array
image = image[:, :, 2]

print(np.mean(image))
plt.imshow(image, cmap='gray')
plt.show()

```

Opgave

- Peg kameraet mod noget farvet og tag et billede.
- Undersøg hvordan billedet ser ud hvis man vælger den blå farve.
- Tag differencen mellem to farver.

Slet billeder

Det er desværre ikke muligt at gemme billeder indefra rumstationen. Hvis programmet skal have lov til at komme i orbit, skal det derfor slette ALLE billederne indefra rumstationen. Det er stadigt muligt at tage billeder og bruge informationen til at undersøge livet om bord, men de skal altså slettes før programmet er kørt færdigt. Denne koden sletter alle filer med endelsen `.jpg`.

`getFilesWithEnding.py`

PAS PÅ, den sletter faktisk filerne!!!

Opgaver - slet filerne

- Opret en ny mappe, `sletTest`
- Tag et par billeder i mappen eks. Med `raspistill -o test.jpg` i konsolen.
- Ret i koden så det kun er den mappes indhold som slettes.
- Udkommenter de nederste 3 linjer, `#`, og tjek om det er de billeder som skal slettes.
- Kør programmet hvor I sletter billederne, uhuuu.

Kameraopløsning

NB øvelsen er til det gamle kamera, vi forventer meget højere oplosning med det nye!
Vi bruger billederne på [Flickr](#) fra tidligere missioner.



På billedet [billeder/Bangkok14.39052N99.30089N.jpg] kan man rimeligt tydeligt se en ø og koordinaterne er skrevet på billedet. Vi vil prøve at finde størrelsesforholdet af vores billed. Det er ikke muligt at bruge koordinaterne direkte, men hvis man søger lidt på google maps ligner det at det godt kan være øen Mali Kyun på Thailands kyst [google maps](#).

Opgave

Vi vil nu måle afstanden i google maps og den tilsvarende afstand i pixels på billedet. Vi skal gøre det vandret og vi ved at billedet har en oplosning på (1920x1080) pixels.

- Brug værktøjerne i google maps til finde afstanden fra spidsen af Mali Kyun til Great Western Torres islands (en lille ø).
- Mål længden i pixels på billedet.
- Beregn længden per pixel.
- Beregn længden af den synlige del af Jorden fra rumstationen.
- Giv et bud på usikkerheden.
- Vurder om det er muligt at se Emma Mærsk på 397m længde.

min løsning:

($d_{Kyun_Torres} = 180.63\text{ km}$, Antal pixels = 851px, Oplosning = $0.21\text{ km/px} = 210\text{ m/px}$, Bredden af billedet er $d=1634\text{ px} \Rightarrow 347\text{ km}$, Hvis jeg har målt 10px forkert bliver intervallet [210m;215m] per px.)

Vi har altså ca. en oplosning på 210m per pixel, med en oplosning på oplosning på (1920x1080).

Opgave

De kamera som sidder på ISS er PiCamera v1 med en maksimal oplosning på (2592x1944) ved still billeder. Ved video er det (1920x1080).

- Beregn oplosningen per pixel med den høje oplosning.

Opgave

ISS bevæger sig med en fart på ca $v = 7.6\text{ km/s}$

- Hvor lang tid går der for at et objekt har bevæget sig ud af billedet?
- Hvordan vil I bruge kameraet til at estimere farten af ISS? Kan man bruge det sammen med en højdemåling?

Exif data

Exif data er metadata som er gemt i billederne. [Astro Pi Mission Space Lab Phase 2 guide](#) har et godt eksempel og via [oversigten over programmer til rådighed](#) en [grundig beskrivelse](#) af hvordan man gemmer data i exif format.

Hvis man skal læse exif data kan det gøres med programmet
(exifRead.py)



For at kunne køre (exifRead.py) skal du installere [pillow](#), hvis din python installation ikke har den allerede. Brug følgende kommando med den såkaldte pip installer:

```
pip3 install Pillow  
og hvis du er på en pi:  
Sudo pip3 install Pillow
```

```
import PIL.ExifTags  
import PIL.Image  
img = PIL.Image.open('gps1.jpg')  
exif_data = img._getexif()  
  
exif = {  
    PIL.ExifTags.TAGS[k]: v  
    for k, v in img._getexif().items()  
    if k in PIL.ExifTags.TAGS  
}  
  
print(exif)  
print('Brightness', exif['BrightnessValue'])
```

Opgave

- Indlæs et billed fra ISS og undersøg hvilke information I kan hve ud af det.

Computer-vision med Astro Pi

Computer vision går ud på at lade computeren foretage valg baseret på de billeder den modtager. Der er mange forskellige måder at udvælge og behandle billeder og vil vil kun berøre en meget lille del af det. Vi starter med at arbejde med det kamera som vender mod jorden. Her er det muligt at tage billeder på tilfældige tidspunkter i løbet at de tre timer man har til rådighed, men det er også muligt at udvælge hvornår der skal filmes. Billederne kan groft sagt udvælges ud fra motiv eller sted, koordinater. Selv om det er muligt, at vide hvor ISS er når billedet bliver taget, kan vi ikke på forhånd vide hvor den vil bevæge sig hen over i de tre timer vi har til rådighed. Nogle muligheder, hvis man vil udvælger efter motiv er, kystlinje, bjergkæder, skyformationer, byer eller måske lyserøde solnedgangsskyer, er skyer lyserøde oppefra? Det afhænger alt sammen af om man kan tage billeder på det helt rigtige tidspunkt.

Jord, luft, vand og ja rumstation

Programmet, (landSeaDetection.py), er inspireret af 2019 deltageres program, hvor de målte fraktaldimensioner af skyformationer. Programmet inddeler billedet i fire dele, land, skyer, hav



og rumstation. Billedet viser ISS på vej ind over Thailands kyst.

Hovedideen er at udvælge efter farve og vi bruger OpenCV biblioteket og HSV farkekoden som er forklaret her, Link til [OpenCv HSV forklaring](#).

Her kan man lege med dem allesammen,

<http://colorizer.org/>

Opgave

- Find et billede eller to og test programmet.

Programmet er en lille smule uoverskueligt men er delt ind i funktioner til at finde de forskellige dele.

Koden nedenfor bliver brugt til at finde hav.

```
def find_sea(img):  
    hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)  
    mask = cv2.inRange(hsv, (90,50,50), (125,255,255))  
    output = cv2.bitwise_and(img, img, mask=mask)  
    return output
```

I linje 2 bliver billedet oversat fra BGR til HSV. Fordelen ved HSV er at der nu kun er en variabel til at bestemme farven.

- H, hue, giver farven, [0;180]
- S, saturation, hvor mættet farven er, eller hvor meget gråt der er i, [0;255]
- V, value, hvor lys farven er, [0;255]

```
mask = cv2.inRange(hsv, (90,50,50), (125,255,255))
```

Giver et nyt billede med hvid (255) der hvor pixelværdierne er inden for intervallet, H:[90;125], S:[50;255], V[50;255], og sort, (0) ellers.

Opgave

- Gå ind på <http://colorizer.org/> og undersøg hvilke farver der ligger i intervallerne.

```
output = cv2.bitwise_and(img, img, mask=mask)
```

Tager det oprindelige billede `img` og viser de steder hvor `mask` er hvid.

Opgave

Det kan være illustrativt at se `mask`

- Ret i linje 16 så det bliver sort/hvid billede af vand som I ser (I skal bruge `mask` variablen).

Opgave



Programmet beregner procentdelen af billedet som er hhv. hav, land, skyer og rumstationen. Hvis vi vil sige noget om det vi flyver over, er det måske ikke så relevant med rumstationen.

- Lav om i `calculate_percentage` funktionen så I fjerner den del som hører til rumstationen.

HSV og BGR

Det kan være lidt forvirrende at arbejde med HSV filtre, hvis man er vant til RGB. Ved RGB farver er alle farver bygget op af rød, grøn, blå (RGB) og blandinger af dem. I det menneskelige øje er der receptorer som reagerer på hhv. rødt, grønt og blåt lys, de hedder tappe. Det er også de farver som en computerskærm kan udsende og hvis vi eks. ser gult lys er den røde og grønne lysbølger som rammer øjet og vi fortolker det som gult lys.

I programmerne, (HSVTrackbar.py) og (RGBTrackbar.py), kan I prøve med skydere. Den virker også på jeres bærbare, hvis I har installeret openCV.

Opgave

- Find noget med klare farver, måske den sværste opgave.
- Eksperimenter med skyderne, så I dedekterer objekterne hver for sig.
- Eksperimenter med lysforholdene, kan I stadigt dedekterer objekterne?

Opgave

Her er en lidt større opgave hvor I skal bruge jeres viden om hvordan man tager billeder og gemmer dem, sammen med detektionen.

- Sammensæt et program som gemmer et billede når et farvet objekt kommer indenfor linsen.

Life in space

Der er også et kamera inde i rumstationen. Her er den store udfordring at vi ikke må hente billeder ned fra rumstationen. Vi bliver derfor nødt til at udvikle metoder til at lave undersøgelser uden at kunne se dem!!!

Her er forslag til tre undersøgelser.

Farver på ISS

Når man ser billeder indefra ISS virker det meget sort/hvidt. En af de hindringer for fremtidige rumrejser er også det psykiske miljø. Det kunne være interessant at undersøge farverne som kameraet kan se. Det kan også være interessant at se om astronauternes tøj er farvet, her skal man så også detekttere hvornår astronauterne er i billedet.

Lysforhold på ISS



Ligesom vi er vant til at se farver er vi også vant til at lyset skifter i løbet af dagen og i overgangen fra dag til nat og tilbage igen. Variationen kan undersøges på Jorden og sammenholdes med forholdene på rumstationen. Det kan komplementeres med variation i tryk, temperatur og luftfugtighed, som vi også observerer på Jorden.

Bevægelse på ISS

Hvor hurtigt bevæger astronouterne sig og hvor tit er de i det modul hvor Astro Pi sidder? Kan man for øvrigt måle tryk-, temperatur- og accelerations-udsving når en astronaut passerer? Det kan også være man kan se på orienteringen af astronouter og deres bevægelse. Til det skal vi kunne detektere bevægelse.

Bevægelsessensor

Vi vil udvikle en algoritme som kan vise om der er bevægelse ved at se på ændringen mellem to billeder. Programmet (motionDetection.py) gør netop det. Vi bruger igen openCV til at behandle video og indlæser det i **frame**.

For at gøre billedet simplere laver vi det først om til gråtoner

```
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
```

For at minimere støj køre vi også billedet gennem et gaussfilter, hvor hver pixel bliver en vægtet sum af de omkringliggende. Vægtningen er som en normalfordeling eller en gaussfordeling.

```
frame = cv2.GaussianBlur(gray, (21, 21), 0)
```

(21, 21) angiver (sigmaX, sigmaY) altså hvor bred gaussfordelingen skal være, hvilket i runde træk svarer til at hver pixel er gennemsnit af den 21 til venstre, højre, op og ned. 0 angiver hvad programmet gør i kanterne hvor der ikke er 21 pixels.

Opgave

- Prøv at lav gaussfordelingen om, husk at der er to steder i programkoden.
- Hvorfor vil man gerne have billeder som er ufokuserede og gråtonede når man skal detektere bevægelse?

motionDetection.py

```
import numpy as np
import cv2

threshold=100

img = cv2.VideoCapture(0)
ret, frame = img.read()
height, width, nchannels = frame.shape
```



```

gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
frame = cv2.GaussianBlur(gray, (21, 21), 0)

while(True):
    frame0 = frame
    ret, frame = img.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY) # We apply a black &
white filter
    frame = cv2.GaussianBlur(gray, (21, 21), 0) # Then we blur the
picture
    if not ret:
        break

    # how different is it?
    if np.sum( np.absolute(frame-frame0) )/np.size(frame) > threshold:
        print('change')
    else:
        print( 'no change' )

    # show it
    cv2.imshow('Type "q" to close',frame)

    # exit if so-commanded
    key = cv2.waitKey(1) & 0xFF
    if key == ord("q"):
        break

# When everything done, release the imgture
img.release()
print('Video exit' )

```

Programmet tager to billeder og sammenligner dem med

```
if np.sum( np.absolute(frame-frame0) )/np.size(frame) > threshold:
```

Linjen sammenligner `np.sum(np.absolute(frame-frame0))/np.size(frame)` som er summen af den absolutte forskel på de to billeder divideret med størrelsen af billedet, med en tærskelværdi.

Opgave

- Prøv programmet af og ret i tærskelværdierne indtil I synes at den detekterer ordentligt.
- Prøv at skift `cv2.imshow` linjen ud med `cv2.imshow('Type "q" to close',np.absolute(frame-frame0))`. Hvad ser i nu?



- Undersøg hvordan billedet ændrer sig når I forøger sigmaX og sigmaY, I kan gå højt op hvis I vil.

Som I kan se i konsollen skriver den det samme mange gange. Det kan gøres smartere!!

Opgave

I skal lave et program som tæller hvor mange gange I flytter en hånd ind foran kameraet. Det skal I gøre ved at

- Definer en ny variabel til at tælle, eks. `count=0` og en til at holde styr på om der er sket en ændring eks. `change=True`.
- Opdater count hvis `change=True`.
- Prøv det af og bliv nok skuffede, den tæller sikkert for meget.

Det at den tæller for meget kan I klare, enten ved at lave et tidsinterval hvor `count` ikke opdateres eller kun opdatere den hvis den ændrer sig fra `False` til `True`.

Opgave

- Find en måde at forbedre det.

Billedbehandling er et kæmpe område og bliver brugt på alt fra cancerdiagnostik til detektion af sorte huller. Det er også et område som kan være ret teknisk, både programmeringsmæssigt og matematisk. Opgaven i Astro Pi konkurrencen er at udtænke interessante eksperimenter med simple løsninger. Her er det vigtigt at tage et skridt af gangen og sætte delmål i sin programmering eks. tage billede -> tag billede et betemt sted -> vurder om det er land -> gem billedet. Så kan man senere se om det indeholder byer og supertankere og måske kan man svare på sit grundlæggende spørgsmål, *om Månenens placering har betydning for fragttransporten til og fra Rotterdam*.

Vi håber at noten har givet mod på at få sit eget program i kredsløb.

Apendiks: opsætning af fjernadgang til Astro Pi

Det er rigtig smart at kunne styre sin Astro Pi fra egen computer, men det kræver noget opsætning. Her er en guide.

Der er en ret grundig beskrivelse her,

<https://desertbot.io/blog/headless-raspberry-pi-4-remote-desktop-vnc-setup>

men hvis I følger det nedenfor kommer I også i mål.



Astro Pi på wifi

Hvis I har installeret styresystemet Raspbian, preinstalleret på Astro Pi, har I en desktop hvor I kan logge på nettet oppe i højre hjørne.

Hvis I hellere vil bruge terminalen kan I ændre i wpa_supplicant.conf filen ved at skrive:

```
sudo nano /etc/wpa_supplicant/wpa_supplicant.conf
```

I terminalen.

I filen kan I se hvilke netværk jeres Astro Pi skal logge på.

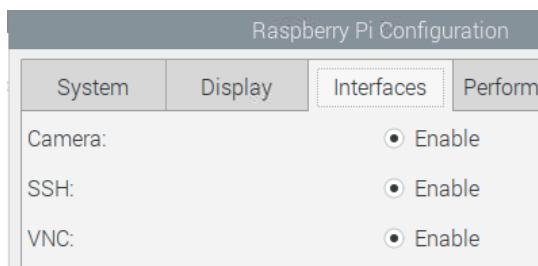
```
country=DK
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
network={
    ssid="network name"
    psk="password"
    key_mgmt=WPA-PSK
    priority = 10 }
```

Hvor I selv skal skrive netværksnavn og password, de er begge case sensitive. priority=10 giver netop dette netværk højeste prioritet, default er 0.

I kan også bruge denne metode til at slette netværk som I ikke længere vil benytte.

VNC adgang til Astro Pi

I kan bruger Virtual Network Computing (VNC) til at kontrollere Astro Pi fra jeres computer. På jeres Astro Pi skal I aktivere den under Preferences->Raspberry Pi Configuration -> Interfaces:



På jeres computer skal I have programmet, realVNC, realvnc.com

I programmet skriver I IP adressen (forklaries nedenfor) og navn og kode, så er I inde.



Standard **brugernavn**: pi og **adgangskode**: raspberry . Det kan være en rigtig god at skifte den på egen Astro Pi og lave nogle standardiserede for sin klasse.



NB! Det virker generelt bedst hvis man er på samme netværk, computer og Astro Pi.

Astro Pi på netværket uden skærm og tastatur

Hvis det ikke er muligt at tilslutte skærm og tastatur kan man stadigt godt få adgang til sin Astro Pi, hvis VNC er slæet til. Hvis I har den kørende på skolen, men eleverne låner den hjem kan det give problemer med at logge på hjemmenetværket men det de skal gøre er:

- Indsæt SD kortet i egen computer.
- (mac) Åben Terminale og find boot mappen, `cd /Volumes/boot`.
- Lav en fil med navnet **wpa_supplicant.conf** (mac: `nano wpa_supplicant.conf` og `ctrl x` for at gemme)
- Skriv i filen.

```
country=DK
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
network={
    ssid="network name"
    psk="password"
    key_mgmt=WPA-PSK
    priority = 10
}
```

Når Astro Pi starter logger den selv på det netværk I har givet.

Terminalen

Terminalen er et uundværligt værktøj når man programmere. På mac og linux og Astro Pi er den preinstallert, men på Windows maskiner skal I installere den, [Link til microsoft webstore](#).
I terminalen kan man skrive kommandoer til computeren



Hvis man vil vide hvilke filer der er i en mappe kan man skrive `ls`. Der er masser af gode grunde til at bruge terminalen men her skal vi især bruge den til at finde IP adressen på vores Astro Pi.

Find IP adresse

Hvis Astro Pi er på samme netværk som jeres computer kan I skrive `arp -a` i terminalen og finde IP adressen. Det er også muligt at bruge mobiltelefonen til at scanne og der er ret mange forskellige programmer.



Hvis I har tilsluttet jeres AstroPi til en skærm kan I finde ip adressen under VNC symbolet



Nu er I klar til at taste det ind i VNC og I får en virtuel adgang til jeres Astro Pi.

SSH setup uden skærm, lidt mere avanceret

SSH giver mulighed for at få adgang til Astro Pi med en terminal. Det er slået fra fra starten, men kan slås til med:

- (mac) Åben Terminale og find boot mappen, cd /Volumes/boot
- Lav en fil med navn ssh, > ssh (ja du skal skrive > og så ssh)
- Tjek om filen er der med kommandoen, ls

Når Astro Pi starter vil den slå ssh til.

Log in med SSH og Enable VNC

Log ind på Astro Pi med ssh sådan:

- Skriv ssh pi@ipAdresse i terminalen
- Default password er, raspberry

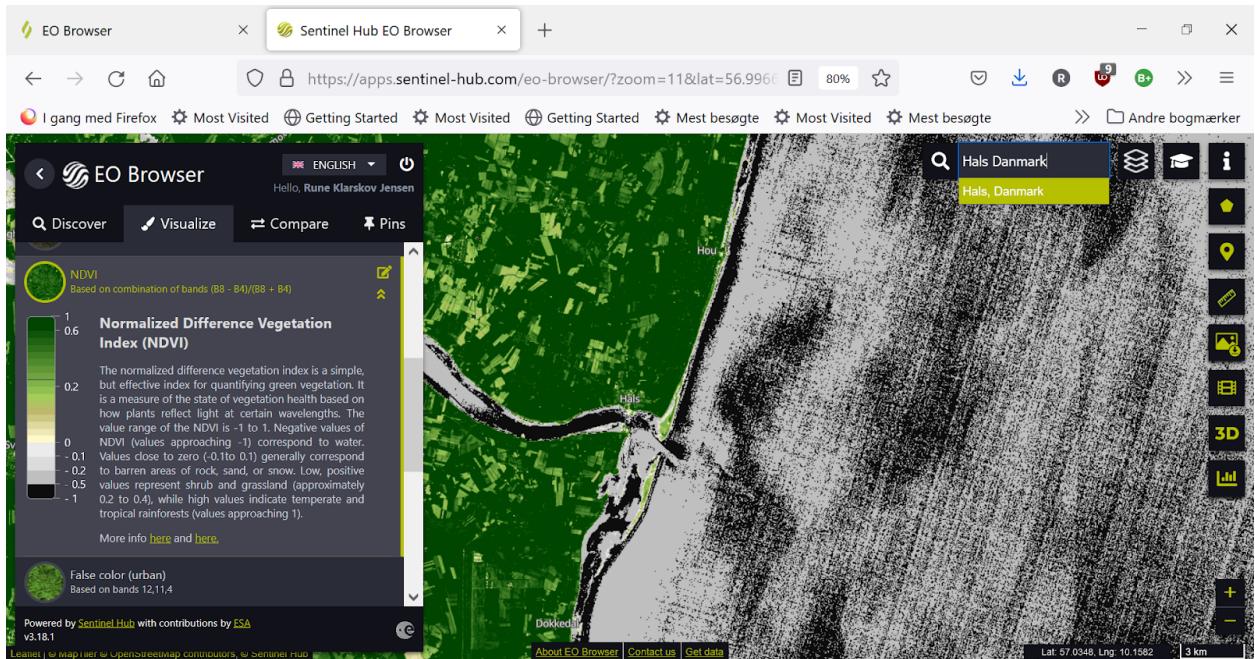
Nu er du inde på Astro Pi og kan køre programmer osv.

For at slå VNC til skal du:

- Skriv, sudo raspi-config
- Vælg **Interfacing Options**
- Vælg **VNC**
- Enable VNC, vælg **Yes (Y)**
- For the confirmation, vælg **Ok**

Det er som sagt lettere at tilslutte det til en skærm. Det kan være et TV, en projektor eller en monitor.





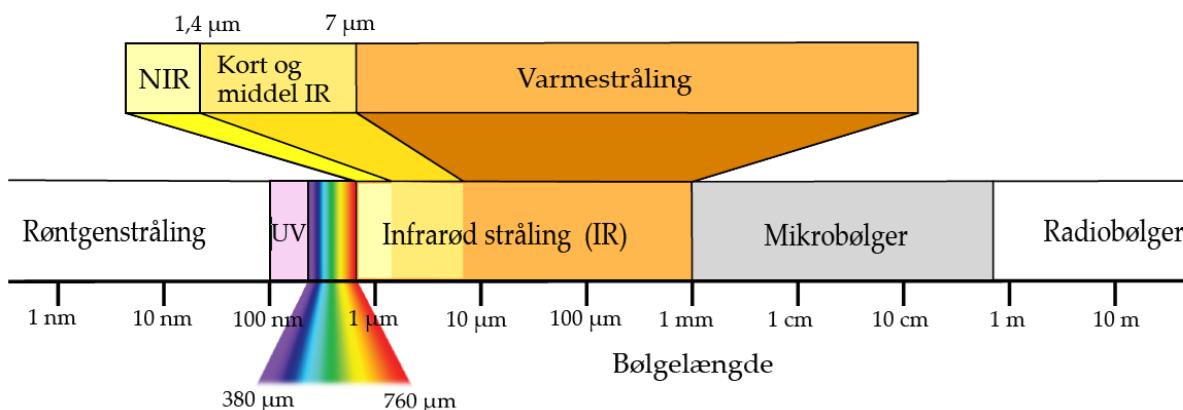
Filplacering: github.com/rune2291/FIP22

Følgende er udkast til lektionsplaner om:

- Billeder af jordens naturforhold med EO browseren uden programmering
- Pythonprogrammer med infrarødt og synligt lys
- Skyer, landjord eller hav fra pixel i billeder.

Lidt om bølgelængder:

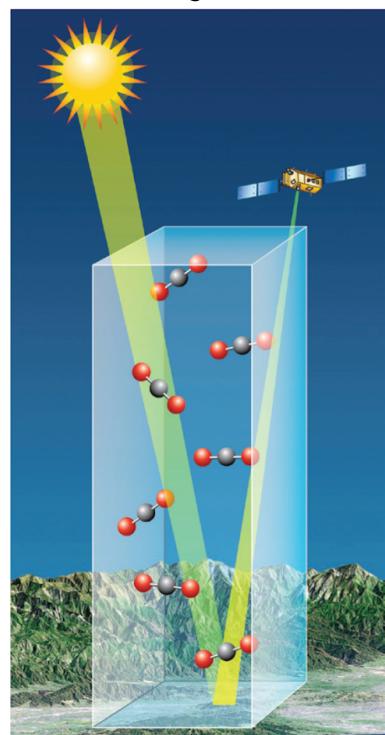
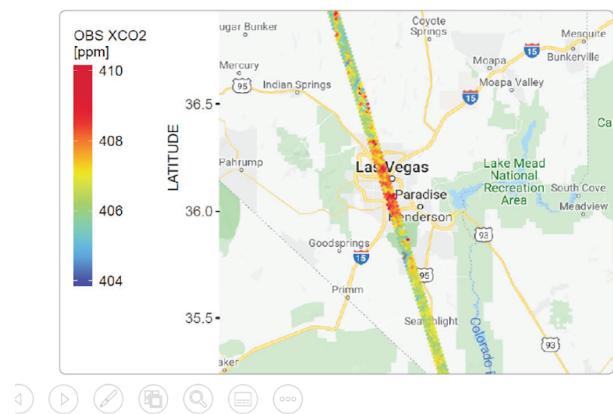
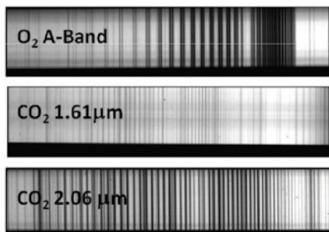
Alt lys kan beskrives som bølger med bølgelængder, her er en oversigt over nogle af de mest kendte i figuren herunder. Det synlige lys er i området fra 380 nm til 760 nm (Der er en lille fejl i billedet prøv at finde den). Iførste lige nu er opmærksomheden rettet mod NIR Nær Infra Rødt som har bølgelængder i området fra 760 nm til 1400 nm.



Lidt om Absorption:

Fra Astro PI kameraet kan der tages billeder i NIR. Men et andet sted på rumstationen kan der tages billeder med et andet kamera i bølgelængder, som kan bruges til at undersøge CO₂ indholdet i atmosfæren. I Billedet herunder øverst til venstre illustreres bølgelængder, der kan afsløre CO₂, ved at CO₂ absorbere energi på bestemte bølgelængder. Til højre illustreres hvordan solens stråler reflekteres fra Jordens overflade og opfanges i rummet af apparatet på rum stationen. I det reflekterede sollys kan man undersøge bølgelængderne og finde udtryk for CO indholdet i atmosfæren, ved hjælp af absorption.

Nederst til venstre ses en måling af CO₂ indholdet i atmosfæren over Las Vegas.

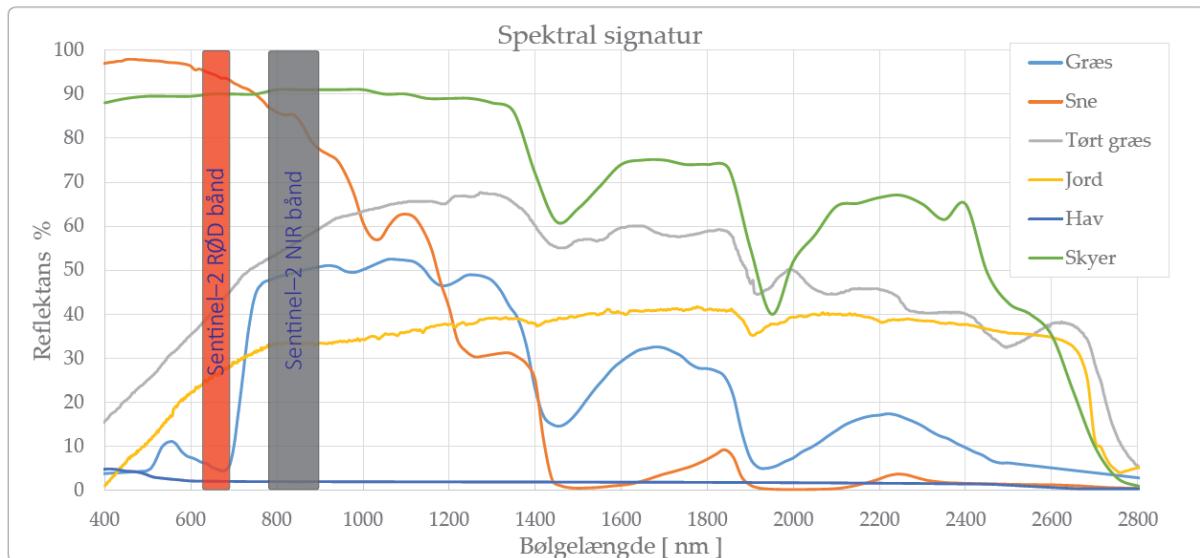


Med Astro PI kameraet kan man ikke optage billeder med bølgelængder der kan måle CO₂ indhold i atmosfæren.

Lidt om Reflektans:

Men det er muligt at tage billeder med NIR som kan bruges til at undersøge en hel masse andet. Der iblandt planternes indhold af klorofyl. For at forstå det skal man se nærmere

på bølgelængderne af lyset der reflekteres fra planter. Se figuren nedenfor:



Ser man på den blå linje i billedet der har overskriften 'Spektral signatur' kan man konkludere at græs lige præcist begynder at reflektere lys, der har en bølgelængder i de Nær InfraRøde bølgelængder. Der er næsten ingen, eller meget lidt reflekteret i den røde farve i det synlige lys om kring 600 til 680 nm. Det kan bruges til at undersøge græs og lignende måde tørt græs, se den grå linje.

Billedbehandling med 'eyecandy' i GUI - Læringsmål:

At bruge og undersøge et IT systems interaktionsdesign

Vurdere dets brugbarhed til at analysere satellitbilleder i falske farver inklusiv infrarød

Se 'skjulte' detaljer i landskabet - vegetationstyper - ændringer over tid som følge af eksempelvis klimaændringer.

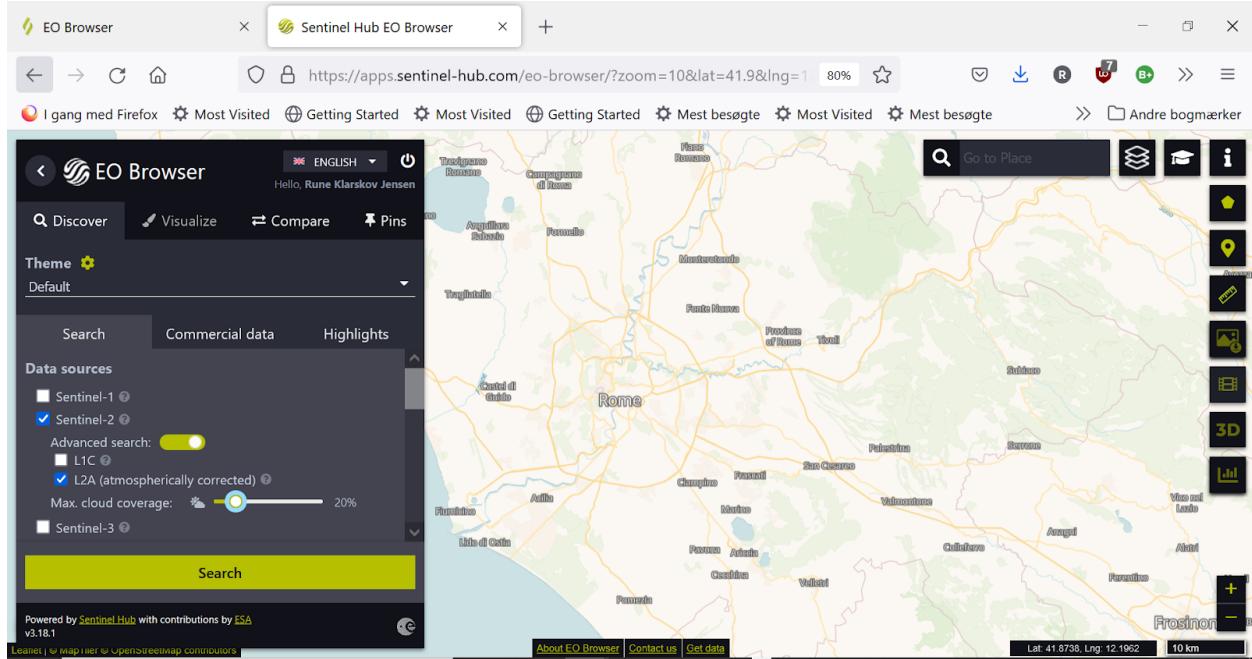
Gør sådan her trin for trin:

1. Åbn EO Browseren på denne adresse:

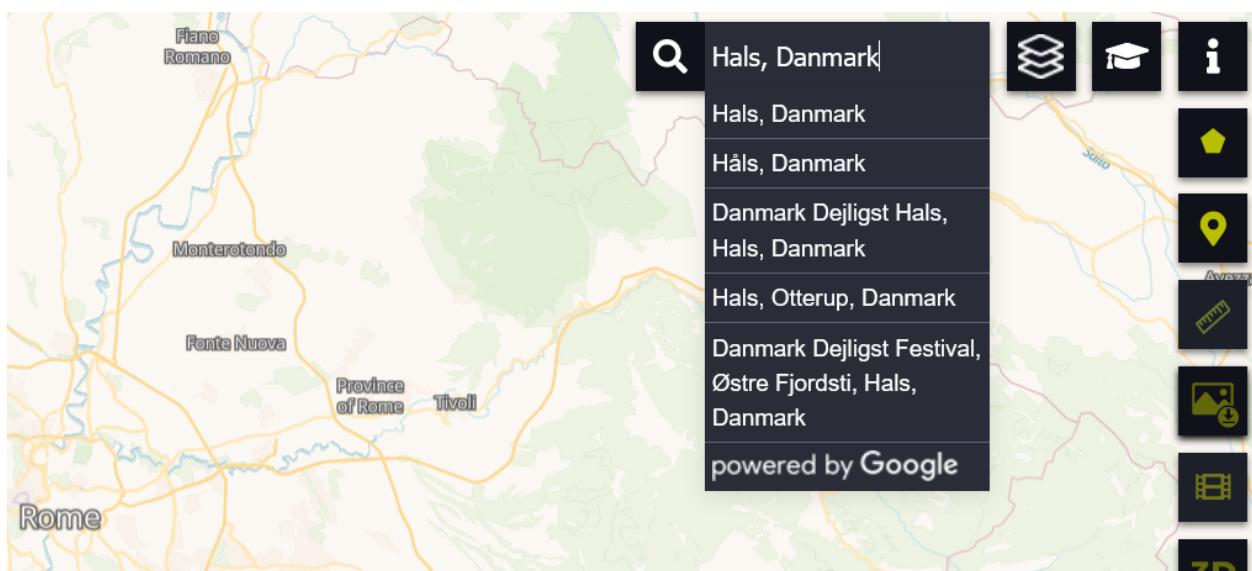
<https://www.sentinel-hub.com/explore/eobrowser>



- Tryk på Start Exploring knappen. Hvis du ønsker det kan du lave dig et login, og dermed også gemme gode billede
- Sæt flueben i Sentinel 2 og klik på Advanced Search og sæt flueben L2A og reguler skydækket til fx 20% .



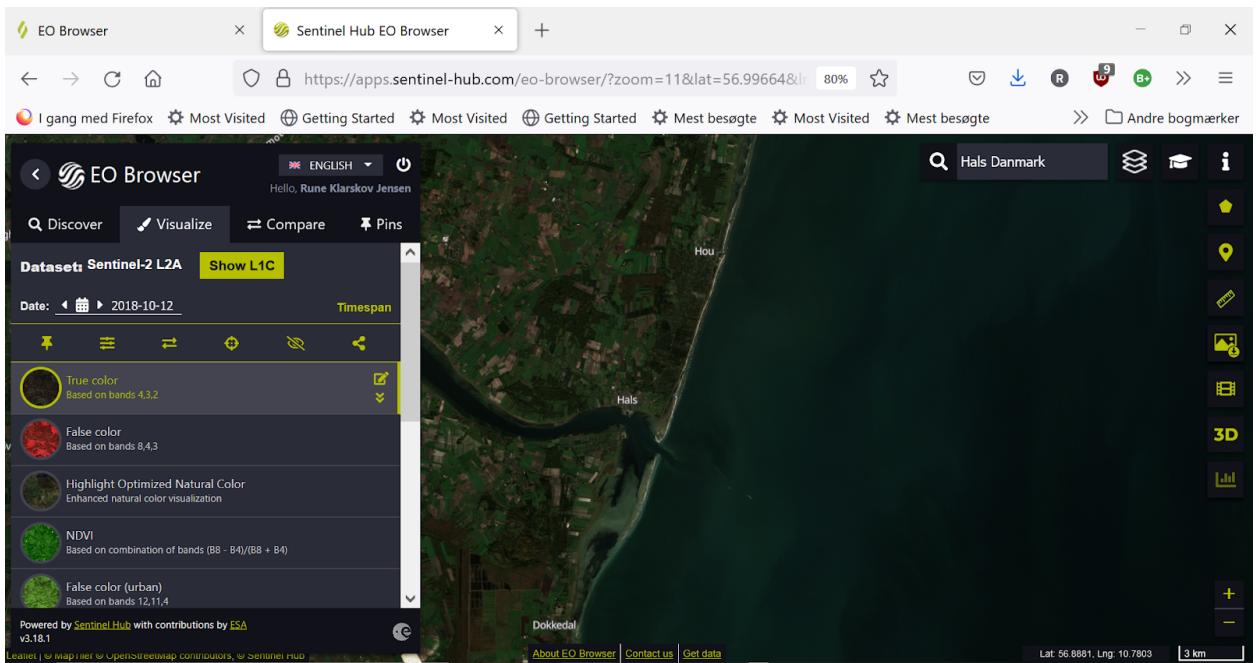
- Skriv navnet på det sted, der skal arbejdes med fx Hals, Denmark, og zoom ind til det ønskede nærområde ved at brug zoom (+)-tasten i nederste højre hjørne.



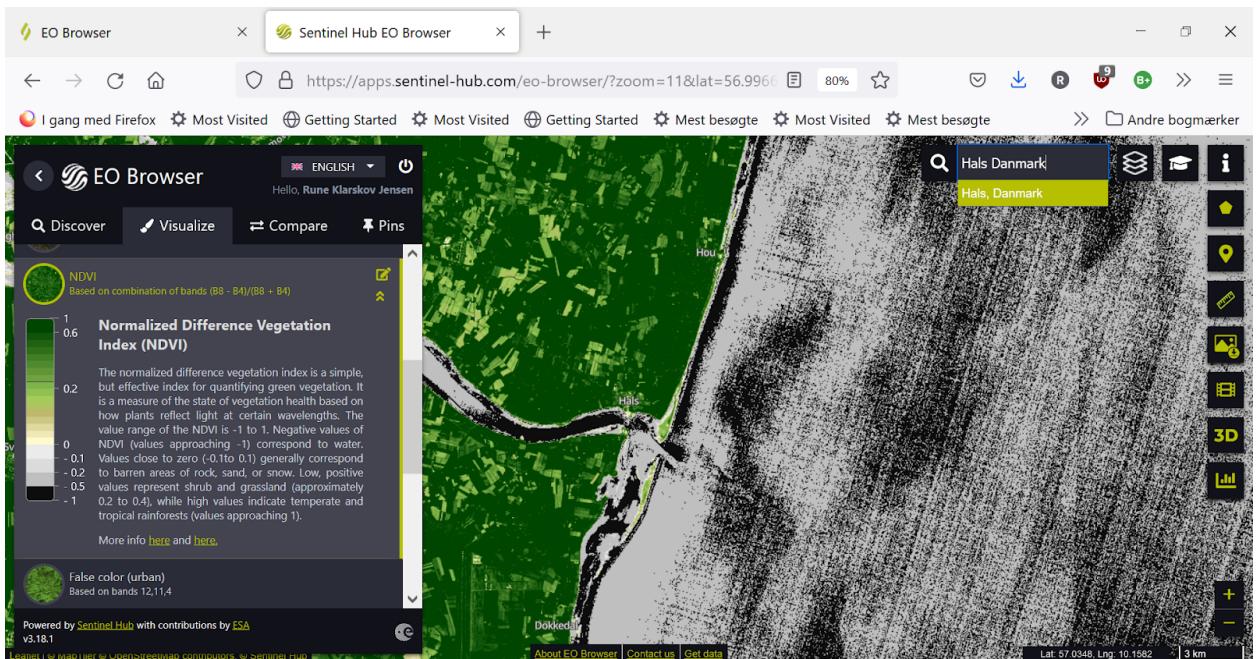
5. Rul ned i 'search vinduet 'Sæt Time Range – her sætter vi fra 2018-07-27 til 2018-10-12 tryk Search.

The screenshot shows the EO Browser interface. On the left, there's a sidebar with a dark theme. At the top of the sidebar, it says 'Discover' and has buttons for 'Visualize', 'Compare', and 'Pins'. Below that is a 'Theme' section with a gear icon and a message: 'Login to use custom configuration instances.' Under 'Default', there are three tabs: 'Search' (which is selected), 'Commercial data', and 'Highlights'. Under 'Search', there's a dropdown menu with 'Planet-NICFI' and 'Other' options. Below that is a 'Time range [UTC]' section with two date pickers set to '2018-07-27' and '2018-10-12'. There's also a checkbox for 'filter by months'. At the bottom of the sidebar is a large yellow 'Search' button. At the very bottom of the sidebar, it says 'Free sign up for all features'. To the right of the sidebar is a map of the Rome area, showing various neighborhoods like Trevignano Romano, Anguillara Sabazia, Formello, Castel di Guido, Fiumicino, and Lido di Ostia. The map uses a color-coded legend for different land types.

6. Der kommer nu en del billeder frem og i dette tilfælde vælges billedet fra 12.oktober 2018. Her er der skyfrit i området og vi får et godt billede frem, hvor der kan studeres mange detaljer:



7. EO Browseren giver allerede her flere muligheder for videre tolkning bl.a. True color, som viser det naturlige billede, false color-billedet, som forstærker vegetationen og tydeliggør søer, hav og vådområder og NDVI som er et vegetationsindeks. Kig på de forskellige muligheder senere.
8. Vælg NDVIog tryk på de to nedadpegede pile lige til højre og se det her:



Se på indeks og observer at skalaen stemmer overens med dine forventninger.

NDVI med Python i Thonny - Læringsmål:

1. Gennem at bruge, afprøve og ændre lidt på python kode komme til at forstå koden og forstå algoritmer i billedbehandling af billeder fra rumstationen ISS.
2. Nærmere specifikt:
 - Lære hvad der menes med Normalised Difference Vegetation Index (NDVI)
 - Konvertere billeder taget med et modificeret kamera sådan at det kan bruges til at måle NDVI.

Gør sådan her trin for trin:

1. Her forklares om NDVI:
<https://projects.raspberrypi.org/en/projects/astropi-ndvi/1>



2. Hvis du ikke allerede har installeret Thonny.org så gör det ved at gå ind på Thonny.org.
3. Åben Thonny → tools → manage packages
4. Søg efter 'numpy' og installer pakken og ligeså med 'opencv-python' pakken.
5. gå herhen med din browser:
<https://projects.raspberrypi.org/en/projects/astropi-ndvi/2>
6. Rul ned til 'To begin with, you are simply going to load an image and display it on your screen.' og følg vejledningen
7. Når du bliver bedt om at hente billedet ned så læg det i samme sted om din python fil og ændr koden 'image = cv2.imread('/home/pi/park.png')' til image = cv2.imread('park.png')
8. Virker det for dig ? Fortsæt da til
<https://projects.raspberrypi.org/en/projects/astropi-ndvi/3> ved at trykke på det nederste grønne link i vejledningen og fortsæt til og med 'Colour mapping'
9. Identificer og vis din sidemakker de vigtige dele af koden i øvelserne 'Display, Contrast, NDVI og Mapping' Især 'def calc_ndvi(image):'

Opdeling i segmenter med inrange fra opencv -læringsmål:

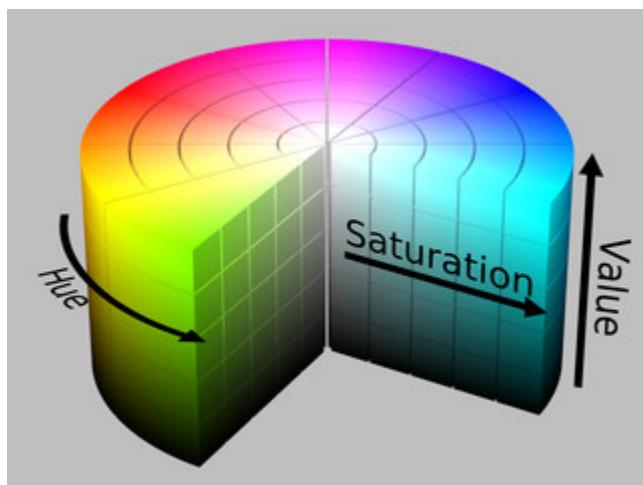
1. At sortere i- og reducere i data ud fra identifikation af informationer i data.
2. At afprøve, ændre på programmerne med henblik på at sætte dem sammen med andre programmer i en projektudvikling.

Gør det her trin for trin:

1. Se *lige* teksten her:

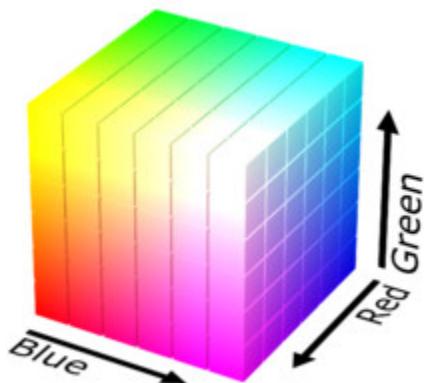


2. [HSV](#) (hue, saturation, value) colorspace er en model der repræsentere farver på en måde der ligner RGB farve modellen. Måden 'hue' kanal modellere farve på er brugbar til at udskille og segmentere objekter som for eksempel hav og skyer baseret på deres farve. Variationen i 'saturation' altså 'mætheden' går fra grå (umættet) til mættet (ingen hvide komponenter). 'Value' kanalen beskriver lysheden i farvens intensitet. Se HSV cylinderen.



By SharkDderivative work: SharkD [CC BY-SA 3.0 or GFDL], via Wikimedia Commons

3. RGB burger tre kanaler til farverne og er derfor sværere at sgementere ud fra basering på farveinformationen



By SharkD [GFDL or CC BY-SA 4.0], from Wikimedia Commons

4. Åben <https://github.com/rune2291/FIP22> med din browser Hent filen inrangeWithTrackbar.py og billedfilen 'coast.jpg' og læg begge samme sted.
5. Kør inrangeWithTrackbar.py og gør de billeder de laver til fuldskærm.
6. Ændr på tærskelværdier og noter dig hvad der sker.
7. Se på originalbilledet og sammenlign prøv nu at flytte 'skyderne' så du udskiller havet.
8. Se ind i koden og find først det sted hvor billedet konverteres til HSV.
9. Find så det sted hvor inrange bruger tærskelværdierne fra skyderne.
10. Hent nu filen landSeaDetection.py kør den og noter dig segmenteringen i landjord, skyer og hav.
11. Åben koden og find tærskelværdierne i 'inrange'
12. Forestil dig hvordan det her kan bruges til at lave et projekt der tager analysere billeder til projekter.

vil du vide mere så deltag på næste astro pi kursus

Og læs undervisningsmaterialet på :

iftek.dk/astro-pi

Filerne i første afsnit skal Birgit Justesen krediteres og takkes for - sidste afsnit skal Mads Peter Hornbak Steenstrup og elever på Rysensteen Gymnasium have tak for at bidrage med og til !

Dokumentet her er tilgængeligt så du kan kopiere og paste og redigere videre under CC BY SA betingelserne her:

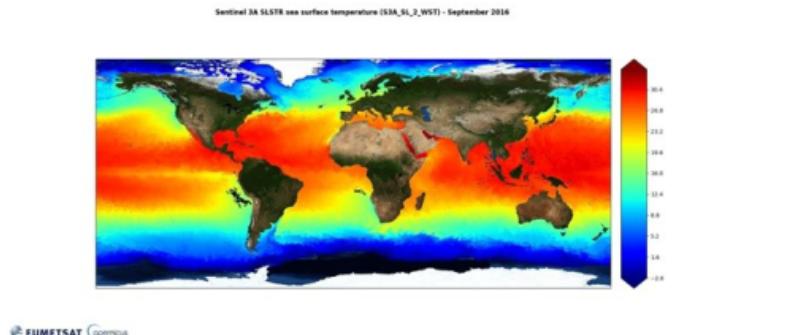
https://docs.google.com/document/d/1GSjHiCvaB_oXSosAcNmTUvfg7y0jcD7DR7BI1R2RpGA/edit?usp=sharing

Videre arbejde:



Fysikfagets Rumfysikopgaver:

Opgave 12 Sentinel-3A



Sentinel-3 er et system bestående af to satellitter Sentinel-3A og Sentinel-3B, som i fællesskab overvåger en stor del af Jorden. Med en spektrograf kan de analysere stråling i det synlige og infrarøde område opdelt i 13 spektrale bånd. Sentinel-3 bruger den infrarøde stråling fra havet til at bestemme havets overfladetemperatur.

Sentinel-3A mäter varmestrålningen från en havoverflade med temperaturen 23,4 °C.

- a) Ved hvilken bølgelængde har varmestrålningen sin største intensitet?

Sentinel-3A bevæger sig i en cirkelbane med radius $7,19 \cdot 10^6$ m omkring Jorden. Det tager 22,0 s at gennemføre en observation af et udsnit af havoverfladen.

- b) Hvor langt bevæger Sentinel-3A sig, mens den gennemfører observationen?

Sentinel-3A registrerer, hvor meget stråling den modtager i spektralbåndet S8 med bølgelængder fra 10,40 µm til 11,30 µm. Detektoren til observation af infrarød stråling har en åbning med diameter 110 mm. Grafen viser den spektrale intensitet, som Sentinel-3A observerer fra havoverfladen.



