

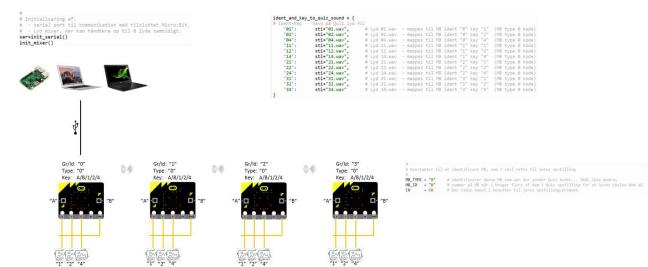


Projekt støttet af:  
**midt**  
regionmidtjylland

Kan bruges sammen med opgave som beskrevet i oplæg vedr. "Lyd i det offentlige rum", med 3 projektoplæg om Nudging, Augmenting samt leg og læring.

Indeholder nogle Byggeklodser i HW (elektronik) og SW (Software) som også kan bruges til andre projektopgaver.

Kan bruges uden at skulle skrive kode ved at benytte de udleverede kode eksempler, og ændre lidt i dem.



# LYD-Kit Spor 2 (LKS2) 2022

Slide stak er tænkt som inspiration til lærer – ikke som et færdigt forløb overfor eleverne.  
En skabelon som der kan plukkes i.

Version 25-feb-2022

- Rettet fejl mht udskrift med noter.

Version 07-Jan-2022

- Tilføjet slides med beskrivelse af optioner – Headless og Fjernkontrol, samt tilføjet et par åbenbog opgaver.

Version 15-Dec-2021

- Tilføjet slides med links til supplerende materiale, som tidligere har været placeret andre steder – Notebooks og projektoplæg.
  - Rettet småting i tekst på slides med Micro:Bit type x, samt rettet i header.
  - Tilføjet introducerende slides omkring fundament og protokol.

Version 18-nov-2021

Rettet i forhold til Version 011121:

- Slide 9 – Stavefejl
  - Slide 3 – fjernet forkert indhold i note, og tilføjet ny note i både slide 3 og 2.

# Spor 2 Modul a: Intro

Materialer/Byggeklodser - hardware

Projekt støttet af:



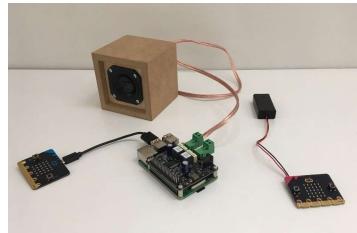
Sensor input fra Micro:Bit via radio/USB kommunikation til PC/MAC/Raspberry Pi, der kan generere lyd via forstærker

Forstærker modul



IQaudio digiAMP+

Højtalere og Micro:Bit



Raspberry Pi mini computer



```
# Initialisering af:  
# - seriel port til kommunikation med tilsluttet Micro:Bit.  
# - mixer, der kan håndtere op til 8 lyde samtidigt.  
ser=serial()  
init_mixer()  
  
# Loop forever  
while True:  
    key = get_microbit_input(ser)  
    if key:  
        print(key, " tegn modtaget på den serielle")  
        if key == "A":  
            play_sound("cheer-cloud.wav")  
        if key == "B":  
            play_sound("buu-trombone.wav")
```

Programeksempler kan køre på PC/MAC med en Micro:Bit tilsluttet!  
Så man behøver IKKE at have opstilling med PI for at komme i gang.

Omvendt kan opstilling med PI gøre det mere til et ‘produkt’, og man kan med PI lave en opstilling der fungerer uden at man skal have skærm og tastatur tilsluttet, og kan få program til at køre når PI tilsluttes strøm. (Headless).

Raspberry pi reference

[Teach, Learn, and Make with Raspberry Pi](#)

IQAudio

[Buy a IQaudio DAC+ – Raspberry Pi](#)

Som alternativ til IQAudio DigiAMP+ kan man bruge HiFiBerry AMP2

[HiFiBerry Amp2 | HiFiBerry](#)

Højtalere enhed

[PLS-P830983 - Højttalere & PA - FreQuence.dk](#)

Kabinet som ses på billedet i slide er lavet af Via Nova i Struer ud fra skitse, som findes sidst i slide stakken.

# Spor 2 Modul a: Intro

Materialer/Byggeklodser – Fundament/udviklingsmiljø

Projekt støttet af:



LYD-Kit Funktionsbibliotek og en række færdige programmer,  
der kan køre på henholdsvis PC/MAC/Raspberry PI og Micro:Bits.  
Tekst baseret Programming og et Cross platform IDE (udviklingsmiljø)



Udviklingsmiljø (Thonny) er frit og gratis tilgængeligt og kører på både PC/MAC og Raspberry PI.

Det introduceres i Spor 1, sammen med Microbit.

Se "Notat om at komme i gang med Thonny" som findes under filer i folderen "Installations vejledninger". Under generelt.

Men gentages her for god ordens skyld – Som klikbare links i slide (Python, Micropython og Thonny)

Links:

[Thonny, Python IDE for beginners](#)

[Python Tutorial \(w3schools.com\)](#)

[BBC micro:bit MicroPython documentation — BBC micro:bit MicroPython 1.0.1](#)

[documentation \(microbit-micropython.readthedocs.io\)](#)

# Spor 2 Modul a: Intro

Materialer/Byggeklodser – Basis funktioner til at afspille lyd

De grundlæggende funktioner der bliver brugt til at lave lyd på PC/MAC/PI ligger samlet i filen "LKLlib.py".

```
init_mixer()
play_sound(sound)
play_background(sound)
stop_background()
set_background_vol(volume)
set_channel_vol(ch,vol_l,vol_r)
stop_sound(ch)
```

For mere beskrivelse af funktionerne og deres parametre kan man kikke ind i LKLlib.py, men kode eksemplerne som følger med kan bruges uden at man går ind i beskrivelsen. Filen LKLlib.py skal bare ligge i samme folder som programfilerne, og de brugte lydfiler.

## OPGAVE:

Load filen "LKS2Ma-Afspil-lyde.py"

Kør program

Forhold jer til brugen af funktionerne

```
Thonny - C:\Users\Knud.Funch\SoundHub.Denmark A SYSTEM - Kit projekt - Dokumenter\Projekt 2019-21\udvikling - moduler\LKS2Ma\LKS2Ma-Afspil-lyde.py ⑧ 39:1
File Edit View Run Tools Help
LKLlib.py LKS2Ma-Miljøspil-2.py LKS2Ma-Case1.py LKS2Ma-Case2.py LKS2Ma-Afspil-lyde.py
1 # LKS2Ma-Afspil-lyde.py
2 #
3 # Version 0.3 - 13 Jan-2021, Knud Funch, SoundHub Denmark - LKLlib til undervisning - Region Midtjylland
4 # Version 1.0 - 22-okt-2021, Justeret til Spor 2, Modul a, Introduktion.
5 #
6 #
7 # Import af de biblioteker og funktioner vi har brug for i denne sammenhæng
8 from LKLlib import *
9
10 # En liste af fil navne på lyd filer med kortte lyklip med fugle lyde
11 fuglelyde = [
12     "agle.wav",           # Index 0
13     "solsort.wav",        # Index 1
14     "tjørn.wav",          # Index 2
15     "skovskade.wav"       # Index 3
16 ]
17
18 # Afspil nogle lyde
19 init_mixer()
20
21 play_background("bogb-trafik.wav") # starter lydfil fra listen med baggrundslyde og brug den med index 0
22 time.sleep(5)                      # vent 5 sekunder - lyd fortæller i baggrund
23 play_background("bogb-fodt.wav")    # hører den overtagere fra den første baggrundslyd og bliver ved ..
24 time.sleep(1)                      # hører ikke mere
25 play_sound("start-seq.wav")       # læner sig ud og spiller 'ovens'
26 time.sleep(1)                      # hører ikke mere
27 for x in range(0, len(fuglelyde)): # først hører vi starter afspilning af alle lydfiler i listen fuglelyde som spiller oven hinanden og oven baggrundslydene
28     print(x)                         # viser nummeret af fuglelyden
29     print(fuglelyde[x])
30     time.sleep(1)
31
32 time.sleep(20)                     # vent til alle fuglelyde er færdig spillet
33
34 # og lad os tage en tilfældig fuglelyd til at slutte med
35 # og lad os tage en tilfældig fuglelyd til at slutte med
36 play_sound(fuglelyde[random.randint(0, len(fuglelyde)-1)])
```

Shell

Python 3.7.9 (bundled)

Anbefaling vis program i Thonny og kør det – fremfor at vise slide.

For at eleverne kan køre program – 'løse opgaven' – skal de have downloaded alle filer til Spor 2 – modul a.

Det er de filer som ligger i folder strukturen: LKS2/Ma efter download. Se filoversigt sidst i slide stakken.

BEMÆRK – Kode kører på PC/MAC/PI.

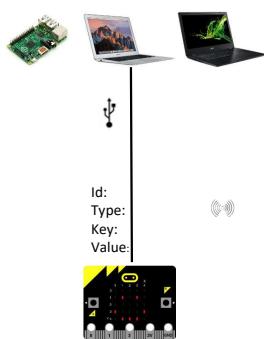
Bemærk samtidigheden af nogle af lydene. Forskel med kald `play_background()` og `play_sound()`

Husk at Thonny skal stå til at udføre program på PC/MAC/PI – Det ses ved at der står "Python 3.x.x (bundled)" i shell  
i modsætning til "Micropython..", hvis Thonny er sat til at udføre kode på Micro:bitten.

# Spor 2 Modul a: Intro

Materialer/Byggeklodser – Microbits og Protokol

Projekt støttet af:



Der er lavet en simpel protokol til at udveksle sensor events og værdier mellem Micro:Bits og programmer der kan køre på PC/MAC og Raspberry Pi.

En Micro:Bit koblet på USB kommunikerer til PC/MAC/PI over seriel forbindelse, og Micro:Bits kommunikerer inbyrdes via Radio på en bestemt kanal (CH), som sættes op som konstant i koden til Micro:Bittene. Ved at benytte forskellige kanaler (CH) kan man have flere opstillinger i gang på samme tid i et klasselokale. Hver elevgruppe skal have én bestemt kanal tildelt (mellem 0 og 83).

Der er en række færdige programmer kaldet "MBType-x.py", som man kan lade på Micro:Bits og derved få dem til at kommunikere sensor inputs. (x er et 0,1,2,3,4,5,6,A,B,C,K,T og de forskellige typer beskrives i det følgende)

Protokollen består af at der sendes et Id, en type, en key og eventuelt en værdi – afhængig af typen. Id kan bruges til at have flere sensorer af samme type i en opstilling. Type og key kan bruges til at identificere specifik sensor event. Man kan komme langt bare med key.

Til PC/MAC/PI er der lavet et funktionsbibliotek – LKlib.py - med de grundlæggende funktioner, samt en række programeksempler/skabeloner man kan tage udgangspunkt i.

Man kan dermed rimelig let og hurtigt kombinere en række forskellige sensor typer og udvide funktionalitet af program på PC/MAC/PI.

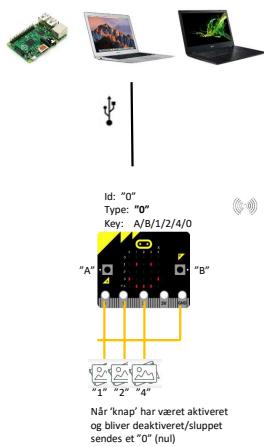
Kan med fordel lave et demo setup på skolen med type 0,1,2,3 og 6 (evt. 4, og 5) sammen med case 1 kode eksemplet,  
Til at fortælle om de forskellige typer.

Pointen er ikke at forstå i detaljen, men bare overordnet at man har nogle færdige kode 'stumper' man kan putte ned på MicroBitten og så bruge sensor inputtet fra den/dem sammen med nogle basis funktioner på PC/PI/MAC til at lave handlinger som at afspille lyd.

# Spor 2 Modul a: Intro

Materialer/Byggeklodser – Software MB type **0,1 og 2**

Projekt støttet af:



Id: "0"  
Type: "0"  
Key: A/B/1/2/4/0

Id: "0"  
Type: "1"  
Key: H/V/F/T/0

Id: "0"  
Type: "2"  
Key: L/I

Når "knap" har været aktiveret  
og bliver deaktivert/slippet  
sendes et "0" (nul)

Vip med lille vinkel til højre  
hhv. venstre giver "H" eller "V"

Vip med stor vinkel  
frem/tilbage giver "F" eller "T".

Når MB har været vippet og  
kommer tilbage til vandret  
sendes et "0" (nul)

Koden til de 3 typer findes under filer /LKS2/MicroBit-Koder-1, med navnene:

MBType-0.py  
MBType-1.py  
MBType-2.py

De kan bruges som de er og downloades i Microbits som "main.py" så virker det umiddelbart.

Ved en genstart/reset af MB vises Id og Type på det lokale display.

Er man ikke helt tilfreds med eksempelvis det lys niveau som skifter mellem L og I - eller den vinkel som genererer et H/V  
så kan man let rette det ved at ændre en konstant i koden inden man downloader til MB.

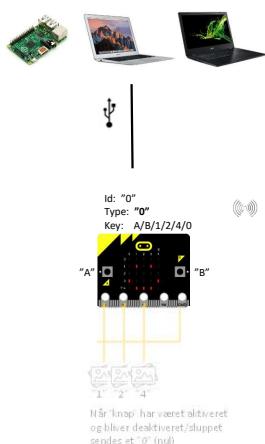
Ønsker man helt ny funktionalitet kan man lave en ny type – Bare hold fast i 'protokollen' med at der sendes Id+type+key for den trigger man ønsker sig.

Så kan man fortsat bruge **get\_microbit\_input()** på PC/MAC/PI uden at der skal ændres i LKlib.py!

# Spor 2 Modul a: Intro

Materialer/Byggeklodser – Input til PC/MAC/PI

Projekt støttet af:



Med `get_micropbit_input()` funktionen fra LKlib.py kan man i PC/MAC/PI modtage Id, Type, og Key fra alle MB's der kommunikerer på samme radiokanal (CH).

Dermed kan man få mange forskellige sensor input's ind til sit PC/MAC/PI program.

Kalder man `get_micropbit_input()` uden argumenter får man key return.  
Key kan så bruges til at identificere det specifikke sensor event og bruges til at aktivere en handling i program på PC/MAC/PI.

Det er underordnet hvilken MB (Type) man tilslutter via USB – Bare det kun er én af dem og resten er med batteri som forsyning.

Skift fokus fra kode i Micro:bit til kode på PC/MAC/PI og de grundlæggende funktioner til at få input fra Micro:Bit til program der kører på PC/MAC/PI.

Grunden til at vi vil det er at vi kan lave mere avancerede lyde på PC/MAC/PI.

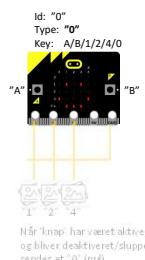
Men også at man så kan lave sammensatte systemer med mange sensor inputs til et overordnet program der kan lave sammenstillinger etc. hvis man vil gå videre.

# Spor 2 Modul a: Intro

Materialer/Byggeklodser – Input til PC/MAC/PI

```
#  
# Initialisering af:  
# - serial port til kommunikation med tilsluttet Micro:Bit.  
# - Lyd mixer, der kan håndtere op til 8 lyde samtidigt.  
serinit_serial()  
init_mixer()
```

```
while True:  
    k = get_microbit_input(ser)  
    if k:  
        print(k, " modtaget på den serielle")
```



Når lysensor detekterer lys over et bestemt niveau og det tidligere har været under sendes et 'L' (stort 1). Modsat når der detekteres et niveau under hvor det tidligere har været over sendes 'T' (lille 0).

Fokus på kode i PC/MAC/PI

# Spor 2 Modul a: Intro

Materialer/Byggeklodser – Load MBType-x kode på Microbit

Projekt støttet af:  
**midt**  
regionmidtjylland

## OPGAVE:

Tilslut Micro:Bit på USB – og hvis den ikke er gjort klar til MicroPython skal man installere firmware på den via Thonny.

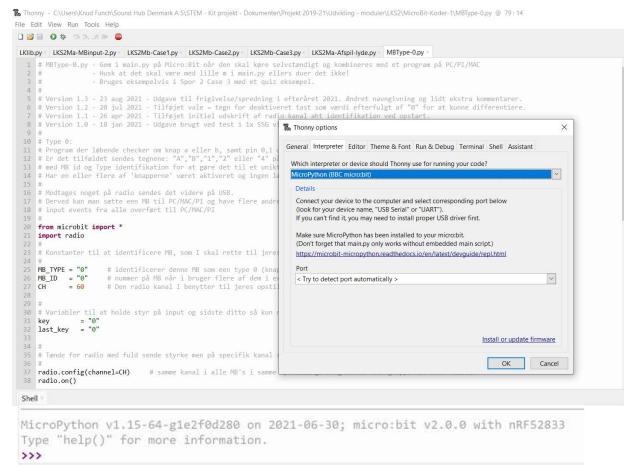
Skift til Micro:Bit/MicroPython i Thonny

Load "MBType-0.py"

Sæt Radio kanal (CH) til unikt nummer så I ikke sender til hinanden.

Gem det på jeres Micro:Bit under main.py så koden ligger lokalt.  
(HUSK at main.py skal skrives med små bogstaver. Main.py duer IKKE)

Skift tilbage til Python på MAC/PC i "Select interpreter ..."



The screenshot shows the Thonny IDE interface. The code editor contains the following Python script:

```
1 # MBType-0.py - Gem 1 main.py på Micro:Bit så den skal køre selvstændigt og kombineres med et program på PC/PI/PMC
2 # Husk at det skal være med lille 'l' i main.py ellers duer det ikke!
3 # Version 1.0 - 23 aug 2022 - Tilføjet funktion til at få identificeringen op i et ekstra kommando.
4 # Version 1.1 - 28 jul 2022 - Tilføjet valg = tegn for deaktivere test som vendt efterfulgt af "0" for at kunne differentiere.
5 # Version 1.2 - 26 apr 2023 - Tilføjede initial udskrift af radio.channel() identifikation ved opstart.
6 # Version 1.3 - 18 jan 2023 - Tilføjte brugt red test i teksten
7 # Version 1.4 - 26 apr 2023 - Tilføjede initial udskrift af radio.channel() identifikation ved opstart.
8 # Version 1.5 - 26 apr 2023 - Tilføjte brugt red test i teksten
9 # Før 0:
10 # Program der løbende checker om knap a eller b, samt pin 0,1
11 # Hvis der tildækkes sendes tegnene "A","B","1","2" eller "4"
12 # Hvis ikke tildækkes sendes tegnene "0","3","5","6","7","8","9"
13 # Hvis ikke tildækkes fra alle overfart til PC/PMC/PI
14 # Hvis en eller flere af "knapperne" været aktiveret og ingen 15:
15 # Radiotegn nogen på radio sendes det videre på USB
16 # Derved kan man sætte em PB til PC/PMC/PI og have flere andre
17 # input/knapper fra alle overfart til PC/PMC/PI
18 # Hvis ikke tildækkes fra alle overfart til PC/PMC/PI
19 # Før microbit import *
20 import radio
21 # Konstater til at identificere PB, som I skal rette til jeres
22 # kanal
23 # Konstater til at identificere radio
24 # Konstater til at holde styr på input og slukke ditto så kun
25 # radio.config(channel=CH) - "0" # Identifierer denne PB som den type ej (kanal
26 # PB_00 til PB_09) - "1" # nummer på PB ej i bruger klavye af den
27 # radio.config(channel=CH) - "2" # Den radio kanal I benytter til jeres opstilling
28 # radio.config(channel=CH) - "3" # Sender til at holde styr på input og slukke ditto så kun
29 # radio.config(channel=CH) - "4" # nummer på PB ej i bruger klavye af den
30 # radio.config(channel=CH) - "5" # Den radio kanal I benytter til jeres opstilling
31 # radio.config(channel=CH) - "6" # Sender til at holde styr på input og slukke ditto så kun
32 # radio.config(channel=CH) - "7" # nummer på PB ej i bruger klavye af den
33 # radio.config(channel=CH) - "8" # Den radio kanal I benytter til jeres opstilling
34 # radio.config(channel=CH) - "9" # Sender til at holde styr på input og slukke ditto så kun
35 # Tænde for radio med fuld sende styrke men på specifikt kanal
36 # radio.on()
37 radio.config(channel=CH) # samme kanal i alle PB's i sammenhæng
38 radio.on()
```

The terminal window shows the MicroPython startup message:

```
MicroPython v1.15-64-g1e2f0d280 on 2021-06-30; micro:bit v2.0.0 with nRF52833
Type "help()" for more information.
>>>
```

## Fokus: Kode i Micro:Bit!!

Mere om Thonny og skift mellem at køre på PC/MAC/PI og MicroBit i  
"Notat om at komme i gang med Thonny" som findes under filer i folderen "Installations  
vejledninger".

Husk at alle i lokalet skal have et entydigt kanal nummer til radio kommunikation for ikke  
at 'genere' hinanden.

Kanalnumre kan være i området: fra og med 0 til og med 83 i henhold til:

[Radio — BBC micro:bit MicroPython 1.0.1 documentation \(microbit-micropython.readthedocs.io\)](https://microbit-micropython.readthedocs.io)

Husk at man kan se om MicroBitten er klar til at udføre kode ved at der i shell skal stå  
"MicroPython.." i modsætning til "Python ..."

Samt ">>>".

Hvis fortolkeren er sat korrekt til MicroPython, men der ikke står det forventede i shell  
kan det skyldes at Micro:Bitten er  
i gang med at eksekvere kode (fordi der allerede ligger en fil med navnet main.py lokalt  
på den).

Så skal man stoppe program ved at trykke på det røde stop symbol i Thonny.

Hjælper det heller ikke skal man forsøge at resette Micro:Bit og efterfølgende en stop kommando igen.

Kan man ikke komme i kontakt med MicroBitten kan man forsøge at se om der er forbindelse ved at åbne valg af fortolker dialog boks og åbne for den der viser port ( forbundne enheder).

Endelig kan man vha samme dialog box genskrive firmware til Micro:Bit, hvis der fortsat ikke er hul igennem.

# Spor 2 Modul a: Intro

Basis funktioner til at læse input fra MicroBit -> PC/MAC/PI

Projekt støttet af:



De grundlæggende funktioner der bliver brugt til at få input fra Micro:Bits over til det program som kører på PC/MAC/PI ligger samlet i filen "Lklib.py". De er:

```
init_serial()  
get_microbit_input(ser)
```

```
24 while True:  
25     key = get_microbit_input(ser)  
26     if key == "A":  
27         print(key, " tegn modtaget på den serielle")  
28         if key == "A":  
29             play_sound("cheer-crowd.wav")  
30         if key == "B":  
31             play_sound("buu-trombone.wav")  
32     else:  
33         print("timeout - så man kan lave andet i programmet, end at vente på input fra den serielle hvis man vil")
```

For mere beskrivelse af funktionerne og deres parametre kan man kikke ind i Lklib.py, men kode eksemplerne som følger med kan bruges uden at man går ind i beskrivelsen. Filen Lklib.py skal bare ligge i samme folder som programfilerne.

## OPGAVE:

Har I husket at skifte tilbage til Python på MAC/PC i "Select interpreter ..."?

Load filen "LKS2Ma-MBinput-1.py"

Kør filen – Reset Micro:Bit som fortsat skal være koblet til USB  
Forhold jer til brugen af funktionerne

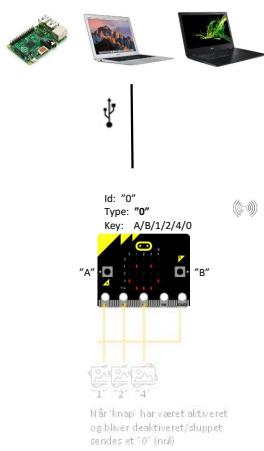
Fokus: Kode i Pc/MAC/PI – så tilbage til python fortolkeren der (>>>Python ....) fremfor (>>>Micropython ...)

Anbefaling: Vis koden i Thonny og kør den – fremfor at vise det som slide!!

# Spor 2 Modul a: Intro

Materialer/Byggeklodser – Input til PC/MAC/PI

Projekt støttet af:



Kalder man `get_micropbit_input()` med argumentet `all_info=True` får man både id, type og key samt en værdi return. Det med værdi bruges til nogle specifikke typer – som beskrives senere.

Det kan så bruges til at identificere det specifikke event fra en specifik Micro:Bit og bruges til at aktivere en handling i program på PC/MAC/PI. Man kan have flere sensorer af samme type forskellige steder til at overvåge forskellige ting

## Fokus på kode i PC/MAC/PI

Udvidet funktionalitet i funktionen `get_micropbit_input()` til der hvor det kan udnyttes.  
Man kan komme langt bare med 'key'!

I Opstillinger med flere Micro:Bits af samme type (som vi kommer ind på til case 3 – QUIZ) kan man udnytte den ekstra information.

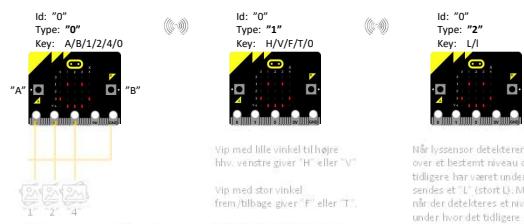
Og også når/hvis man vil bruge de MB typer som sender værdier til PC/MAC/PI.

# Spor 2 Modul a: Intro

Materialer/Byggeklodser – Input til PC/MAC/PI - mere info

Projekt støttet af:



```
#  
# Initialisering af:  
# - serial port til kommunikation med tilsluttet Micro:Bit.  
# - Lyd mixer, der kan håndtere op til 8 lyde samtidigt.  
serinit_serial()  
init_mixer()  
  
modtaget_tuple = get_microbit_input(ser,all_info=True)  
if modtaget_tuple:  
    if TEST_PRINT_ON: print(modtaget_tuple, " tuple modtaget på seriell port (MB ident, MB type, MB key, MB value)")  
    i,t,k,v = modtaget_tuple  
  
  
  
  
  


Id: "0"  
Type: "0"  
Key: A/B/1/2/4/0



Id: "0"  
Type: "1"  
Key: H/V/F/T/0



Id: "0"  
Type: "2"  
Key: L/I



Når knap har været slået ret og bliver deaktivert/slippet sendes et "0" (nul)



Når lysensor detekterer lys over et bestemt niveau og det tidligere har været under sendes et "L" (stort L). Modsat når der detekteres et niveau under hvor det tidligere har været over sendes "T" (lille t).



Vip med lille vinkel til højre hhv. venstre giver "H" eller "V".  
Vip med stor vinkel frem/tilbage giver "F" eller "T".  
Når MB har været vippet og kommer tilbage til vandret sendes et "0" (nul)


```

# Spor 2 Modul a: Intro

Basis funktioner til at læse input fra MicroBit -> PC/MAC/PI

Projekt støttet af:



## OPGAVE:

Load filen:

"LKS2Ma-MBinput-2.py"

Kør programmet på PC/MAC/PI

```
23 while True:  
24     modtaget_tuple = get_microbit_input(ser,all_info=True)  
25     if modtaget_tuple:  
26         i,t,k,v = modtaget_tuple  
27         print(modtaget_tuple," tuple modtaget på seriell port (MB ident, MB type, MB key, MB value)")  
28         if k == "A":  
29             play_sound("cheer-crowd.wav")  
30         if k == "B":  
31             play_sound("buu-trombone.wav")  
32         if k == TABT_FORBINDELSE: ser=init_serial()
```

Forhold jer til funktionerne og forskellen til LKS2Ma-MBinput-1.py

- At man får både key, og ident, type og value (0 når der ikke er noget).
  - Hvad kan det bruges til?

Fokus på kode på PC/MAC/PI

Anbefaling: Vis koden i Thonny og kør den – fremfor at vise det som slide!!

Ud over det med at få alle parametre fra get\_microbit\_input() kaldet så er der også den forskel at der her er lavet en retablering af den serielle forbindelse til Microbitten, hvis den serielle forbindelse afbrydes undervejs. En detalje som kan bruges i åben bog snak om kommunikations sikkerhed og nedbrud, samt hvordan funktionen er lavet i Lklib.py.

MEN det er kun hvis man vil dybere i den retning.

Det er underordnet for at bruge programeksemplerne i 'lukket' bog mode.

# Spor 2 Modul a: Intro

Materialer/Byggeklodser – Software MB type 3,4,5 og 6

Projekt støttet af:



Når magnetfelt værdi har været over en bestemt værdi og tidligere var under sendes "M" (stort M). Modsat sendes "m" når det er under og tidligere var over.

Id: "0"  
Type: "4"  
Key: B/b



Når digital værdi på pin1 er høj og tidligere har været lavt sendes et "B" (stort B). Modsat når der er lavt niveau efter det har været højt så sendes et "b" (lille b)  
Typisk til en bevægelses sensor. (PIR)  
Bemærk "B" sendes også fra type 0 - Kan det udnyttes?

A Scratch logo featuring a yellow and black chevron pattern. Above the logo, the text "Id: \"0\"", "Type: \"5\"", and "Key: F/f" is displayed in a white sans-serif font.

år analog værdi på pin 1 er over et bestemt niveau og det tidligere har været under sendes et "F" (stort F). Modsat der detekteres et niveau under hvor det tidligere har været over sendes "f" (lille f). (typisk til en fugtighedsmåler)

The image shows a Microbit board with its pins visible. Pin 0 is connected to ground, pin 6 is connected to a digital output pin, and the S/s pin is connected to a digital input pin.

B version 2 med mikrofon.  
Når der detekteres støj over et  
bestemt niveau og det tidligere  
været under sendes et "S"  
(lille s). Modsat når der  
detekteres et niveau under  
det tidligere har været  
under sendes "s" (stort s).

Koden til de 3 typer findes under filer /LKS2/MicroBit-Koder-1, med navnene:

MBType-3.py  
MBType-4.py  
MBType-5.py  
MBType-6.py

## FOKUS – Kode i Microbit – flere varianter/typer af sensor input

# Spor 2 Modul a: Intro

Eksempel med MBType-4 kode i Micro:Bit

Thonny - C:\Users\muud\Downloads\Hub-Denmark A SYSTEM - Kit projekt - Dokumenter\Projekt 2019-21\Udvikling - moduler\LKS2Ma-MBinput-2.py @ 34:1

LKS2Ma-MBinput-2.py - MBtype-4.py

```

15 # - seriel port til kommunikation med tilsluttet Micro:bit.
16 # - lyd mixer, der kan håndtere op til 8 lyde samtidigt.
17 ser = init_serial()
18 init_mixer()
19
20
21 # Loop forever
22
23 while True:
24     modtaget_tuple = get_microbit_input(ser, all_info=True)
25     if modtaget_tuple:
26         i, t, k, v = modtaget_tuple
27         print(modtaget_tuple, " tuple modtaget på seriel port (MB ident, MB type, MB key, MB value)")

Sæt
('0', '4', 'B', 0) tuple modtaget på seriel port (MB ident, MB type, MB key, MB value)
('0', '4', 'B', 0) tuple modtaget på seriel port (MB ident, MB type, MB key, MB value)
('0', '4', 'B', 0) tuple modtaget på seriel port (MB ident, MB type, MB key, MB value)
('0', '4', 'B', 0) tuple modtaget på seriel port (MB ident, MB type, MB key, MB value)
('0', '4', 'B', 0) tuple modtaget på seriel port (MB ident, MB type, MB key, MB value)
('0', '4', 'B', 0) tuple modtaget på seriel port (MB ident, MB type, MB key, MB value)
('0', '4', 'B', 0) tuple modtaget på seriel port (MB ident, MB type, MB key, MB value)
('0', '4', 'B', 0) tuple modtaget på seriel port (MB ident, MB type, MB key, MB value)

```

MBType-4.py kode loaded lokalt i i Micro:bitten (main.py).  
PIR sensor koblet til Microbitten.

LKS2Ma-MBinput-2.py i PC/MAC/PI.  
Key = "B" modtages når der er bevægelse foran PIR sensor

Blot med som inspiration.

Hvis man vil er det bedre at demonstrere det real life med kode kørende på Pc og med Micro:Bit og PIR sensor tilsluttet.

Man kan også lave det til en opgave. Lad eleverne loade en type-4 kode, og koble en PIR sensor på MicroBitten og kør med kode der aflæser værdier på PC/MAC/PI.

Ved bevægelse kan se hvad der indlæses fra **get\_microbit\_input()** kaldet.

Da type-4 sender key ="B" ved bevægelse, og type-0 også sender "B" (når der trykkes på knap B) så får man med samme kode

på PC stadig afspillet en lyd (buu\_sound.wav), da der ikke testes for type i LKS2Ma-Mbinput-2.py.

D.v.s. at man kan 'simulere' PIR sensor med type-0 og tryk på knap B.

Man kan derudover lave opgaver med konstruktion af div elektriske kredsløb og kombinere med de eksisterende typer af Micro:Bit koder, og/eller etablering af nye typer.

# Spor 2 Modul a: Intro

Materialer/Byggeklodser – Oversigt Microbit-koder-1

Projekt støttet af:



Type (navn)	Key (ASCII)	Beskrivelse
MBType-0.py	A/B/1/2/4/0	Når 'knap' har været aktiveret og bliver deaktiveret/slippet sendes et "0" (nul som tegn)
MBType-1.py	H/V/F/T/0	Når MB bliver vippet. Der sendes et nul når tilbage i udgangspunkt
MBType-2.py	L/I	Når der er lys over et bestemt niveau sendes L. Ved skift til lidt lys sendes lille I.
MBType-3.py	M/m	Når magnetfelt er over et bestemt niveau sendes M. Ved skift til lidt lys sendes lille m
MBType-4.py	B/b	Når digital input på pin 1 bliver højt sendes stort B. Når det går lavt sendes lille b Mulig konflikt med type 0!! Men det kan også udnyttes (simulering af PIR sensor (bevægelse) med en type 0)
MBType-5.py	F/f	Når analog værdi på pin 1 kommer over en bestemt værdi sendes F. Når den kommer under sendes lille f
MBType-6.py	S/s	Når indbygget mikrofon detekterer støj over et bestemt niveau sendes stort S. når det kommer under sendes lille s

Oversigt over de Micro:Bit type koder som er gennemgået indtil videre, og som er tilstrækkelige til projektopgaven og de programeksempler der følger med til spor2. Efterfølgende vises flere færdige typer, men det er for dem som vil gå dybere.

# Spor 2 Modul a: Intro

Materialer/Byggeklodser – MBtype-T – Tinker Kit

Projekt støttet af:



Id: "0"  
Type: "A"  
Key: K/k/C/c/B/b/F/P  
Value: None/1-5/0-1023



"K" samt Value 1-5 (integer) når key (1-5) aktiveres  
"K" når key deaktiveres  
"C"/"c" når crash sensor aktiveres/deaktiveres.  
"B"/"b" når PIR (bevægelses) sensor aktiveres/deaktiveres  
"F" samt Value 1-5 (fugt niveau) når værdi ændrer sig.  
"P" samt Value (0-1023) når ny værdi læses fra potentiometer.



Koden til "Type T" findes under filer /LKS2/MicroBit-Koder-2, med navn:

MBType-T.py

De kan bruges som de er og downloades i Microbits som "main.py" så virker det umiddelbart.  
Ved en genstart/reset af MB vises id og Type på det lokale display.

Er man ikke helt tilfreds med eksempelvis de niveau skift som er i fugtmåling så kan man let rette det ved at ændre en konstant i koden inden man downloader til MB.

Under /LKS2/Microbit-Koder-2 findes der desuden en række andre typer der kan bruges til at sende sensor værdier til PI/PC/MAC. (Se næste slide for en oversigt).

Som et eksempel på en kombination af mulige sensorer der kan aflæses og sendes.

Her er det alle input sensorer fra det udvidelses KIT til Micro:Bit som hedder Tinker-KIT.

Her kommer den fjerde parameter i spil.

- Eksempelvis til at overføre potentiometer værdier til PC/MAC/PI

# Spor 2 Modul a: Intro

Materialer/Byggeklodser – Oversigt Microbit-koder-2

Projekt støttet af:



Type (navn)	Key (ASCII)	Value	Beskrivelse
MBType-A.py	P	0-1023	Når analog værdi på pin 1 ændrer sig tilstrækkeligt sendes P plus værdi. (eg potentiometer)
MBType-B.py	X (Y,Z,V)	+/- 2000	Når acceleration i X-retning ændres tilstrækkeligt sendes et X efterfulgt af værdi (accelerationen). Hvis det sættes op så bruges acceleration i andre aksler, alternativt kombineret (V for vektor) Med V kan værdi også blive større end 2000!
MBType-C.py	I	0-255	Når lysintensitet ændres tilstrækkeligt sendes et I for intensitet efterfulgt af værdi
MBType-D.py	M	80 tusind Op til 3 mill.	Når magnetfelt styrken ændres tilstrækkeligt sendes et M efterfulgt af værdi (Mulig konflikt med type 3, hvis man har begge i en opstilling – Så skal man skelne dem på type)
MBType-K.py	A/B/1/2/4/0	0 når tast aktivieres 65,66,49,50,52 når hhv. A,B,1,2,4 tast deaktivieres	Alternativ til Type 0 – Kompatibel med type nul, men er hurtigere (kun med 10 msec sleep i mødsætning til 50msec i type 0, og radio buffer tømmes inden sleep). Desuden angives den tast der er deaktivert i value når "0" sendes som key.
MBType-T.py	K/k C/c B/b F P	1,2,3,4,5 0 0 1,2,3,4,5 0-1023	Tinker Kit sensor inputs når Keyboard er på pin0, potentiometer på pin1,fugtmåler på pin2, crash sensor på pin8 og PIR sensor på pin12. Se mere i kommentarerne i koden.

Der er yderligere kode stumper som kan sende værdier til PC/MAC/PI som kan ses i denne oversigt.

Her kommer den sidste retur parameter i spil.

Her kommer værdier fra sensorerne med over til PC/MAC/PI.

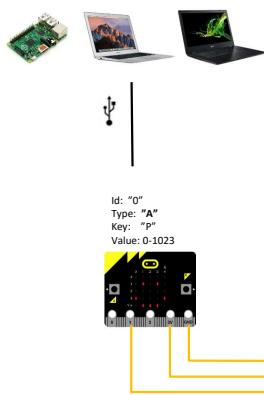
De bruges ikke i kode eksemplerne case 1,2,3 i Spor 2, men ligger til dem som vil mere.

I spor 4 er der et enkelt eksempel der benytter dem.

Man kan her tænke på at lave opgaver der logger sensor værdier etc.

# Spor 2 Modul a: Intro

Eksempel med MBType-A kode i Micro:Bit



```

Thonny - C:\Users\Knud Funch\Sound Hub Denmark A SYSTEM - Kit projekt - Dokumenter\Projekt 2019-21\Udvikling - moduler\LKS2Ma\MBinput-2.py | 34:1
File Edit View Run Tools Help
LKS2Ma-MBinput-2.py
15 # - seriel port til kommunikation med tilsluttet Micro:Bit.
16 # - Lyd mixer, der kan håndtere op til 8 lyde samtidigt.
17 serinit_serial()
18 init_mixer()
19
20
21 # Loop forever
22
23 while True:
24     modtaget_tuple = get_microbit_input(ser,all_info=True)
25     if modtaget_tuple:
26         i,t,k,v = modtaget_tuple
27         print(modtaget_tuple," tuple modtaget på seriel port (MB ident, MB type, MB key, MB value)")

Shell:
tuple modtaget på seriel port (MB ident, MB type, MB key, MB value)
('0', 'A', 'P', 497) tuple modtaget på seriel port (MB ident, MB type, MB key, MB value)
('0', 'A', 'P', 484) tuple modtaget på seriel port (MB ident, MB type, MB key, MB value)
('0', 'A', 'P', 473) tuple modtaget på seriel port (MB ident, MB type, MB key, MB value)
('0', 'A', 'P', 462) tuple modtaget på seriel port (MB ident, MB type, MB key, MB value)
('0', 'A', 'P', 463) tuple modtaget på seriel port (MB ident, MB type, MB key, MB value)
('0', 'A', 'P', 459) tuple modtaget på seriel port (MB ident, MB type, MB key, MB value)
('0', 'A', 'P', 454) tuple modtaget på seriel port (MB ident, MB type, MB key, MB value)

MBType-A.py kode loaded lokal i i Micro:bitten (main.py).
Potentiometer koblet til Microbitten.

LKS2Ma-MBinput-2.py i PC/MAC/PI.
Key = "B" modtages når der er bevægelse foran PIR sensor

```

Blot med som inspiration.

Hvis man vil er det bedre at demonstrere det real-life med kode kørende på PC/MAC/PI og med Micro:Bit og Potentiometer tilsluttet.

Ved bevægelse af potentiometeret kan man se hvad der indlæses fra `get_microbit_input()` kaldet på PC/MAC/PI

Man kan også lave det til en opgave.

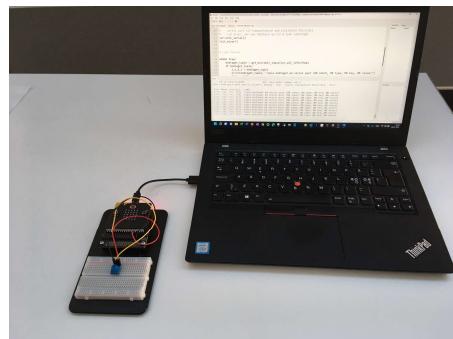
Man kan derudover lave opgaver med konstruktion af div elektriske kredsløb og kombinere med de eksisterende typer af Micro:Bit koder, og/eller etablering af nye typer.

Projekt støttet af:



# Åben bog del til modul a

Hvis man ønsker at dykke dybere i koden og at lave nye typer.



# Spor 2 Modul a: Intro

Åben bog – MBType-x koder

Projekt støttet af:



Overordnet struktur for alle MBType-x koder:



Sæt radiokanal op til den gruppe man vil lave (holde kommunikationen indenfor)  
Udskriv information om MicroBit (Kanal,Identifier,Type) så man kan skelne de forskellige MB's i en gruppe fra hinanden.

Læs sensor input (key) – (Forskellig for de forskellige Typer)

Hvis der er sket ændringer i sensor input (key) så:

- Laves en event med typen, identifier og key, som sendes:
  - på den serielle kanal (USB kabel til PC/PI/MAC) med print kommando'en
  - på radio kanal til de andre MB's i gruppen.
- Key udskrives på Microbittens lokale display for at vise hvad der er registreret.

Hvis noget er modtaget på radio, så:

- sendes det på den serielle (USB) til PC/MAC/PI (med print kommando)

Hvis man vil åbne op for hvordan de forskellige MB Type x koder fungerer kan man tage udgangspunkt i Type 0 og forklare at de alle bruger samme overordnede struktur.

Det kan springes over, hvis man kører 'lukket' bog med fokus på at lave et færdigt produkt med de eksisterende programeksempler.

Udskrift af information på Micro:Bittens lokale display er værdifuldt når man har flere Micro:Bits i en opstilling, og de fleste af dem er forsynet med batteri.

Når der kun er én MicroBit i opstilling som er tilsluttet PI/PC/MAC vil det kun være den som videregiver sensor input (events – ident,type,key,value) – men den gør det for alle MicroBit's i gruppen!

Det gør ikke noget at alle forsøger at sende på den serielle kanal (USB), da det kun er den ene som er koblet op.

Tilsvarende gør det ikke noget at den som er koblet på USB sender lokale sensor event på radio til de andre.

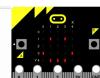
# Spor 2 Modul a: Intro

Åben bog – MBType-0 koden som eksempel

```

1 # MBType-0.py - Gem i main.py på Micro:Bit når den skal køre selvstændigt og kombineres med et program på PC/PI/MAC
2 #
3 # - Husk at det skal være med lille n i main.py ellers dør det ikke!
4 #
5 # Version 1.3 - 23 aug 2021 - Ældgave til frigivelse/spredning i efteråret 2021. Andret navngivning og lidt ekstra kommentarer.
6 # Version 1.2 - 20 jul 2021 - Tilføjet vale = tagn for deaktiveret tast som verdifølgt af "0" for at kunne differentiere.
7 # Version 1.1 - 26 apr 2021 - Tilføjet initial udskrift af radio kanal, idt identifikation ved opstart.
8 # Version 1.0 - 18 jan 2021 - Ældgave brugt ved test i 1x SSG vinter 2020/2021. LYD-KIT Region Midtjylland
9 #
10 # Type 0:
11 # Hoved program der løbende checker om knap a eller b, samt pin 0,1 og 2 på MB er aktiveret
12 # Er det tilfældet sendes tegnene: "A", "B", "1", "2" eller "0" på USB og Radio, sammen
13 # med MB id og type identifikation for at gøre det til et unikt event med flere MB i opstilling.
14 # Har en eller flere af 'knapperne' været aktiveret og ingen længere er aktive så sendes et "0"
15 #
16 # Modtages noget på radio sendes det videre på USB.
17 # Derved kan man sætte en MB til PC/MAC/PI og have flere andre der sender på radio og få
18 # input events fra alle overført til PC/MAC/PI
19 #
20 from microbit import *
21 import radio
22 #
23 # Konstanter til at identificere MB, som I skal rette til jeres opstilling
24 #
25 MB_TYPE = "0"      # identificerer denne MB som en type 0 (knapper)... SKAL ikke ændres
26 MB_ID = "0"        # nummer på MB når i bruger flere af dem i ex en Quiz opstilling for at kunne skelne dem ad
27 CH     = 60         # Den radio kanal I benytter til jeres opstilling/produkt (0-83)
28 #
29 #
30 # Variabler til at holde styr på input og sidste ditto så kun ny/andrede knap tryk sendes.
31 key     = "0"
32 last_key = "0"
33 #
34 #
35 # Tænde for radio med fuld sende styrke men på specifik kanal som benyttes i opstillingen
36 #
37 radio.config(channel=CH)    # samme kanal i alle MB's i samme opstilling. Brug nummer som gruppe har fået tildelt
38 radio.on()

```



Så kan man gå ind i koden til MicroBitten.  
Bedre at vise det i Thonny end som slide!!  
Eller uddel som lektie – enten som fil eller udskrift.

## Spor 2 Modul a:

Åben bog opgave – Lav en Type N der læser og sender lyd niveau

**Opgave 1:** Lav en MBType-E.py til Micro:Bit (version 2), som sender type = "E", key="N" (noise) efterfulgt af værdi v lig med aflæst lyd niveau fra den interne mikrofon på microbit (0-255), som passer ind i den protokol som er i Lyd-Kit"

HINT: Tag udgangspunkt i MBType-C med hensyn til at sende værdi, og i funktionen `microphone.sound_level()` for at læse lydniveau fra mikrofon. [Microphone V2 — BBC micro:bit MicroPython 1.0.1 documentation \(micropython.readthedocs.io\)](https://microcontroller.readthedocs.io/en/latest/micropython/microbit/Microphone_V2.html)

Type (navn)	Key (ASCII)	Value	Beskrivelse
MBType-E.py	N	0-255	Når lyd niveau ændres tilstrækkeligt sendes et N for Noise/niveau efterfulgt af værdi.

**Opgave 2:** Brug 4 MBType-E.py til Micro:Bit, med id 1-4 og placer dem rundt om i klasseværelset. Lav program på PC/MAC/PI, der aflæser lyd niveau de 4 steder, og se hvor der er mest 'larm'.

HINT: Tag udgangspunkt i "LKS2Ma-MBinput-2.py", der direkte vil udskrive hvad der modtages (print tuple). Udvid program til at monitorere over tid og gemme værdier. Lav en grafisk repræsentation eller gem i fil.

### Opgaver

Ny kode til en ny MB type.

Anvend den i en større sammenhæng indenfor den struktur som er fastlagt i LYD-Kit.

For at inspirere til at tænke videre, og for at forstå værdien af at bygge ovenpå noget andet.

## Spor 2 Modul a:

Åben bog opgave – Lav en Type 7 der etablerer 5 knapper på analoginput

**Opgave 1:** Lav en MBType-7.py til Micro:Bit, som sender type = "7", key="K" (Keyboard) efterfulgt af værdi v lig med 1,2,3,4 eller 5 afhængig af aflæst niveau på analog input PIN 1. Lav et spændingsdelings kredsløb med modstande i serie og knapper imellem som giver 5 forskellige spændingsniveauer, som kobles til Microbit. Skal fortsat passe ind i den protokol som er i Lyd-Kit"

HINT: Tag udgangspunkt i MBType-A med hensyn til at læse analog værdi, og i stedet for at sende den analoge værdi til PC/MAC/PI så skal den lokalt laves om til værdien 1,2,3,4 og 5 afhængig af spænding der matcher hvilken knap der er aktiveret.

Type (navn)	Key (ASCII)	Value	Beskrivelse
MBType-E.py	N	0-255	Når lyd niveau ændres tilstrækkeligt sendes et N for Noise/niveau efterfulgt af værdi.

**Opgave 2:** Brug 2 MBType-7.py til Micro:Bit, med id 1 og 2. Og med 2 gange det kredsløb der er lavet til knapper. Lav program på PC/MAC/PI, der laver 10 forskellige lyde for alle 10 knapper.

HINT: Tag udgangspunkt i "LKS2Ma-MBinput-2.py", og test både på id og værdi der modtages.

### Opgaver

Ny kode til en ny MB type.

Anvend den i en større sammenhæng indenfor den struktur som er fastlagt i LYD-Kit.

For at inspirere til at tænke videre, og for at forstå værdien af at bygge ovenpå noget andet.

Projekt støttet af:



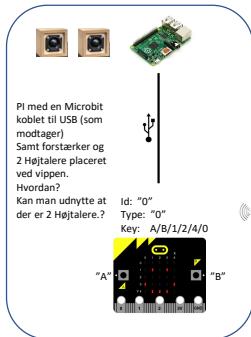
# Modul b

Cases – Programeksempler der kan bruges til at komme i gang til projekttopgaverne i Teknologiprojektoplæg vedrørende lyd i det offentlige rum.

# Spor 2 Modul b: Cases

Case 1 – Augmenting

```
# Initialisering af:  
# - serial port til kommunikation med tilsluttet Micro:Bit.  
# - Lyd mixer, der kan håndtere op til 8 lyde samtidigt.  
serinit_serial()  
init_mixer()
```



```
64 while True:  
65     k = get_microbit_input(ser)  
66     if k:  
67         print(k, " modtaget på den serielle")  
68         # Afspil en tilfældig lyd fra listen af lyde når der kommer en af triggerne nedenfor.  
69         # Dvs tilfældig lyd fra en trigger fra en MB med enten type 0,1,2,3,4 eller 6  
70         # Hvad skal der til for at få trigger fra type 5 til også at virke?  
71         if (k in ["1", "2", "4", "B", "M", "L", "S", "H", "V"]):  
72             play_background(lyde[random.randint(0,len(lyde)-1)], forever=False)  
73  
74     if k == "A": stop_background()
```



Ved at bruge accelerometer funktion på én Micro:Bit og placere den på én vippe kan man lave en lyd når vippen  
benyttes. Load MBType-1.py kode i en Micro:Bit med batteri placeret på vippen, og brug en anden Micro:Bit som  
modtager koblet til Pi/PC/MAC via USB – Det behøver ikke være samme type som den der er på vippen.

Type 1 sensor placeret på en vippe

Flere sensorer og flere legeredskaber ... ... Hvad vil det betyde for delen med højtaleren??

Ved at putte en MBType-1 på en vippe (og evt. justere følsomhed) kan man lave en lyd når vippen bevæger sig den ene hhv. den anden vej. (H/V)

I case 1 koden er der handling på flere sensor inputs og med tilfældige lyde fra en liste  
- men bruger man kun én type sensor er det jo kun den der kommer igennem så det er ikke et problem.

- man kan naturligvis bare modificere i koden, hvis man vil.

Der afspilles en tilfældig lyd fra en liste med lyde.

Man kan bare skifte lydene ud, og modificere længden af listen.

Hvis der kun er et element i listen vil det altid være den ene lyd som spilles.

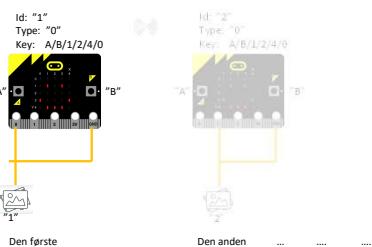
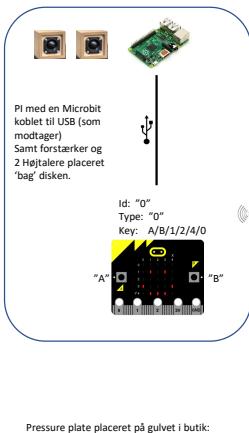
- MEN så kan man naturligvis også lave det mere enkelt og lade være med at bruge random.

# Spor 2 Modul b: Cases

Case 2 – Nudging

```
# 
# Initialisering af:
# - serial port til kommunikation med tilsluttet Micro:Bit.
# - Lyd mixer, der kan håndtere op til 8 lyde samtidigt.
serinit_serial()
init_mixer()
```

```
87 while True:
88     k = get_microbit_input(ser)
89     if k:
90         print(k, " modtaget på den serielle")
91         if k == "1":
92             stop_sound(ch_left) # Hvis der allerede er en igang.
93             ch_left = play_sound(lydeV[random.randint(0,len(lydeV)-1)],vol_l=0.05,vol_r=0)
94         elif k == "2":
95             stop_sound(ch_right) # Hvis der allerede er en igang.
96             ch_right = play_sound(lydeH[random.randint(0,len(lydeH)-1)],vol_l=0,vol_r=0.05)
97         elif k == "B":
98             play_background(lydeB[random.randint(0,len(lydeB)-1)])
99         elif k == "A":
100            stop_background()
```



Ved at bruge pin 0,1,2 på op til 3 Micro:bits kan I lave op til 3 'pressure plates' uden at I nødvendigvis behøver at give den et unikt id.  
Dvs. at I bare kan putte type 0 kodden i op til 3 Micro:bits uden at ændre i den.  
Altså den som ligger i: "MBType 0.py"

Ved at bruge id nummeret kan man så lave endnu flere hvis det skulle være ...  
Tænk over andre sensor typer der kunne bruges i et butiks setup.

## Skift lydfilerne i tabellerne ud med nogle andre lyde

- Ex. lyden af en gulerod der skrælles, et æble der bides/spises, og en prop der trækkes af en flaske.

Placer nogle plader der kortsluttes ved 2 forskellige MB's på 2 forskellige steder i en butik

- Ex. foran grøntsagsafdelingen og foran vinafdelingen
- Højttaler fra højre kanal ved grøntsagerne og den fra venstre kanal ved vinen.

Så har man en opstilling, hvor man kan teste om man kan nudge folk til at købe flere grøntsager h.h.v. mere vin.

Der kan også være forskellig baggrunds musik Ex. Italiensk og Fransk samtidigt med de specifikke lyde hvis man ønsker.

Det kan måske også påvirke købsmønsteret. (Fransk versus Italiensk vin)

Det relaterer sig direkte til projektoplæg "Lyd i det offentlige rum" placeret i folderen "Ex på forløb" under filer i Spor 2.

Konkret nævnes et emperisk forsøg der har vist merslag med fransk baggrundsmusik i den TED talk som er med i projektoplæg i linken

[Harnessing the Power of Sound | Steve Keller | TEDxNashville – YouTube](#)

**BEMÆK:**

- Det er OK at bruge 2 Micro:Bit's med samme id både som modtager og sender – (Der er jo IKKE nogen brug af 'knapperne på modtager').
- Hvis man kun bruger Knap 1 på den ene og knap 2 på den anden behøver man heller ikke at skelne på id for at have dem 2 forskellige steder i butikken.

MEN det vil omvendt være fin stil at give dem unikke numre så man kan se hvilken MB der kommer events fra i programmet på PI.

Giver god information ifm debugging/test af en opstilling.

# Spor 2 Modul b: Cases

Case 3 – Quiz eksempel

QUIZ med 12 fugle billeder og lyde.

2 overordnede tilstande/modes:

**Øve/Practice mode**

Hvor man hører lyden til det billede man trykker på en enkelt gang.

**Quiz mode**

Hvor man hører en tilfældig fuglelyd indtil man trykker på det billede som hører til lyden, hvorefter der kommer en ny tilfældig fuglelyd.

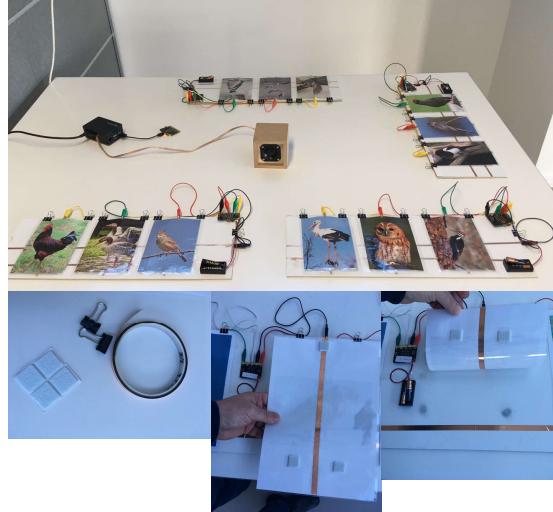
Ved korrekt svar kommer der en 'cheer' lyd.

Ved forkert svar kommer der en 'buh' lyd.

- Knapperne er lavet af lamineret A4 papir med fuglebilleder, med en strimmel kobberfolie og nogle filtpuder på bagsiden.
- 3 billeder pr. Micro:Bit hvor billederne er forbundet til pin0, 1 og 2 via krokodilleledninger.
- GND koblet til kobberfolie på plade under billederne.  
(Det som laver kontakt når man trykker på billederne)
- Micro:Bittene kommunikerer via radio, og én af dem kommunikerer med PC/MAC/PI via USB.

Man kan lave denne Quiz uden at skulle kode. Man skal som det eneste sætte Radio kanal og id i Microbitterne som loades med "MB Type 0" kode.

Man kan lave en ny Quiz uden ændringer i koden, ved bare at skifte lydfilerne og de tilhørende billeder.



Bemærk at ny Quiz kan laves uden ændring i kode – Bare ved at udskifte lydfilerne.

Så kan man holde fokus på at opgaven er at finde en interessant Quiz og evt. have en 'kunde' som har et bestemt tema der skal følges.

Ex. Historie læreren i en anden klasse, som ønsker en Quiz med taler og kendte personer etc. etc ..

Fokus på kunden, brugssituation etc.

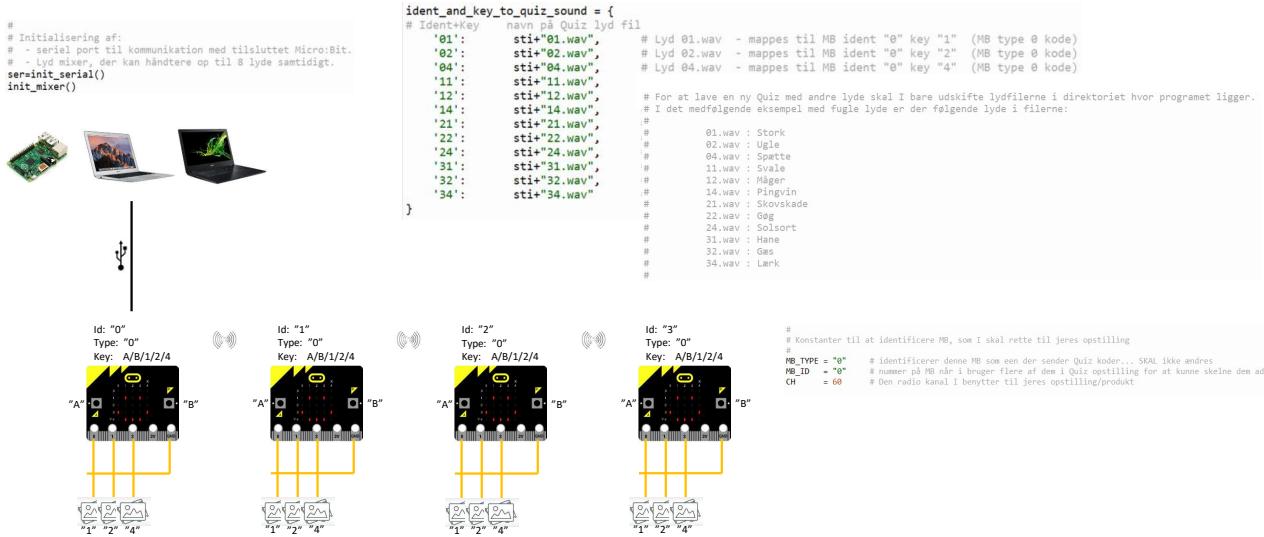
Den tekniske del af opgaven bliver at finde de interessante lydklip – Evt. at få dem klippet til så man umiddelbart kan lave associationen til billede.

Og at tænke over det 'fysiske' design af Quiz knapperne og den kontekst hvori de skal indgå. Ex. plakater på væg i en biograflobby eller ....

Og eleverne kan hurtigt komme til et produkt, som de så kan teste af ex. med elever fra en anden klasse og få noget feedback.

# Spor 2 Modul b: Cases

Case 3 – Quiz eksempel overordnet kobling



Man kan lave en ny udgave af Quiz alene ved at udskifte lyd filerne og lave nye billeder på knapperne!

Man kan gøre det helt uden at lave om i koden ved at bruge de generiske fil navne:  
01.wav,02.wav,04.wav .etc. som mappes til id+key fra Micro:Bit (knapperne 01,02,03 etc.)

Man skal som det eneste sikre sig at man har givet Micro:Bitterne i opstillingen et unikt radio CH nummer, samt id 0,1,2 og 3!

BEMÆRK at man kan tage én microbit mere (OK med samme id som en af de andre) og lade den være modtager på PC/PI og så have alle 4 der sender med batteri.

Hvis man vil have flere kombinationer af lyde og billeder (flere knapper) kan man gøre det alene ved at udvide i tabel, og tilføje Micro:Bits med id 4,5 osv.

# Spor 2 Modul b: Cases

Case 3 – Quiz eksempel – åben bog state/event

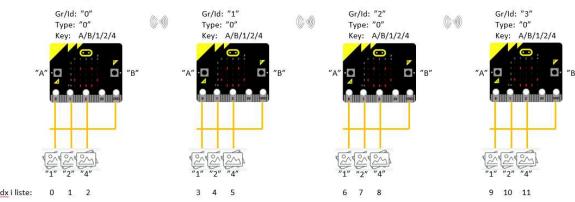


Trigger/input Tilstand/ Mode	"A"	"B"	Valid idx/pos ud fra id og key (i+k)	Alt andet
PRACTICE/ Øve	Ingenting	Find et tilfældigt idx (i+k) Start afspilning af tilhørende lyd  Skift mode til QUIZ	Afspil lyd der svarer til i+k fra input	Ignoreres/ Ingenting
QUIZ	Stop Quiz lyd  Skift mode til PRACTICE	Ingenting	Hvis idx/pos fra input svarer til den som spiller så: Cheer (korrekt svar) Find nyt tilfældigt idx Start afspilning af tilhørende lyd. Ellers Buh (forkert svar) fortsæt med samme quiz lyd (nyt gæt)	Ignoreres/ Ingenting

Hvis man vil kikke ind i koden for Quiz eksemplet bør man starte med at tale om den overordnede logik med udgangspunkt i dette State/event diagram, og nedbrydningen deraf.

# Spor 2 Modul b: Cases

Case 3 – Quiz eksempel – åben bog state/event



Valid idx/pos  
ud fra id og key (i+k)

```

63 ident_and_key_to_quiz_sound = {
64     # Ident+key    navn på Quiz lyd fil
65     '01':      sti+"01.wav",   # Lyd 01.wav - mappes til MB ident "0" key "1"  (MB type 0 kode)
66     '02':      sti+"02.wav",   # Lyd 02.wav - mappes til MB ident "0" key "2"  (MB type 0 kode)
67     '04':      sti+"04.wav",   # Lyd 04.wav - mappes til MB ident "0" key "4"  (MB type 0 kode)
68     '11':      sti+"11.wav",   # Lyd 11.wav - mappes til MB ident "1" key "1"  (MB type 0 kode)
69     '12':      sti+"12.wav",   # Lyd 12.wav - mappes til MB ident "1" key "2"  (MB type 0 kode)
70     '14':      sti+"14.wav",   # Lyd 14.wav - mappes til MB ident "1" key "4"  (MB type 0 kode)
71     '21':      sti+"21.wav",   # Lyd 21.wav - mappes til MB ident "2" key "1"  (MB type 0 kode)
72     '22':      sti+"22.wav",   # Lyd 22.wav - mappes til MB ident "2" key "2"  (MB type 0 kode)
73     '24':      sti+"24.wav",   # Lyd 24.wav - mappes til MB ident "2" key "4"  (MB type 0 kode)
74     '31':      sti+"31.wav",   # Lyd 31.wav - mappes til MB ident "3" key "1"  (MB type 0 kode)
75     '32':      sti+"32.wav",   # Lyd 32.wav - mappes til MB ident "3" key "2"  (MB type 0 kode)
76     '34':      sti+"34.wav"   # Lyd 34.wav - mappes til MB ident "3" key "4"  (MB type 0 kode)
77 }

```

# Spor 2 Modul b: Cases

Case 3 – Quiz eksempel – åben bog state/event

Projekt støttet af:



Trigger/Input	"A"	"B"
Tilstand/ Mode		
PRACTICE/ Øve	Ingenting	Find et tilfældigt idx Start afspilning af tilhørende lyd Skift mode til QUIZ
QUIZ	Stop Quiz lyd  Skift mode til PRACTICE	Ingenting

```

139 modtaget_tuple = get_microbit_input(ser,all_info=True)
140 if modtaget_tuple:
141   if TEST_PRINT_ON: print(modtaget_tuple, " tuple modtaget på serial port (MB ident, MB type, MB key, MB value)")
142   i,t,k,v = modtaget_tuple
143   ident_and_key_input = i+k
144

145
146 if k == "A" and mode != PRACTICE:
147   mode = PRACTICE
148   stop_sound(quiz_channel)
149   if TEST_PRINT_ON: print(mode)
150 if k == "B" and mode != QUIZ:
151   mode = QUIZ
152   stop_sound(quiz_channel)
153   ident_and_key_playing, quiz_channel = start_random_quiz_lyd()

# skift til øve mode efter at have været i quiz mode
# stop afspilning af Quiz lyd
# skift til quiz mode efter at have været i øve mode
# stop afspilning af practise lyd

```

# Spor 2 Modul b: Cases

Case 3 – Quiz eksempel – åben bog state/event

Projekt støttet af:



Valid idx/pos  
ud fra gr/id og key

Afspil lyd der svarer til idx/pos fra input

Hvis idx/pos fra input svarer til den som  
spiller så:  
Cheer (korrekt svar)  
Find nyt tilfældigt idx.  
Start afspining af tilhørende lyd.  
Ellers  
Buh (forkert svar)  
fortsæt med samme quiz lyd (nyt gæt)

```
155 if ident_and_key_input in ident_and_key_to_quiz_sound:  
156     if mode == PRACTICE:  
157         stop_sound(quiz_channel)  
158         quiz_channel = play_sound(ident_and_key_to_quiz_sound[ident_and_key_input])  
159     if mode == QUIZ:  
160         if ident_and_key_input == ident_and_key_playing:      # Korrekt svar  
161             stop_sound(quiz_channel)                         # stop playing quiz lyden  
162             play_sound(cheer_or_buh_sounds["RIGTIG"])        # cheer for rigtigt svar  
163             time.sleep(6)                                    # vent til cheer er over inden ny quiz lyd startes  
164             ident_and_key_playing, quiz_channel = start_random_quiz_lyd()  
165     else:  
166         play_sound(cheer_or_buh_sounds["FORKERT"])       # buuh lyd ved forkert svar  
167
```

Projekt støttet af:



# Option

Headless

Start program fra boot, så det kan køre uden skærm og tastatur på en  
Raspberry PI

# Spor 2 Option

Headless opstilling – Program der starter fra boot på Raspberry Pi.

```
pi@raspberrypi:~$ cd ..  
pi@raspberrypi:~/home$ cd ..  
pi@raspberrypi:/$ dir  
bin dev home lost+found mnt proc run srv tmp var  
boot etc lib media opt root sbin sys usr  
pi@raspberrypi:/$ cd etc  
pi@raspberrypi:/etc$ sudo nano rc.local
```



```
## rc.local  
# This script is executed at the end of each multiuser runlevel.  
# Make sure that the script will "exit 0" on success or any other  
# value on error.  
#  
# In order to enable or disable this script just change the execution  
# bits.  
#  
# By default this script does nothing.  
# Print the IP address  
_IP=$(hostname -I) || true  
if [ "$_IP" ]; then  
    printf "My IP address is %s\n" "$_IP"  
fi  
  
# Her indsættes sti til program/script man vil have till at starte fra boot  
# (I headless opstilling).  
# SKAL ende med et & tegn for at den process/tråd man starter skal gå i baggrunden  
# så boot sekvens kan fortsætte og derned få en prompt etc. efterfølgende.  
# Hvis ikke forbliver proces i forgrund og man kan ikke retablere setup med prompt/GUI etc...  
# Med mindre man på anden vis kan stoppe den proces man starter)  
#  
# Linien SKAL starte med python3 for at få den rigtige version af python fortolkeren startet!!  
python3 /home/pi/LKS2Mb-Case3/LKS2Mb-Case3.py &  
exit 0
```



- Fra kommando linjen i Raspberry OS finder man direktoriet etc.
- Med nano tekstu redigerer man filen "rc.local", og tilføjer den sti og programkode man ønsker skal startes fra boot. Brug sudo foran for at gøre det med administrator rettigheder.
- HUSK at linjen skal afsluttes med "&" tegnet!!

Tilføj program som skal køre fra boot til filen "rc.local" i direktoriet "/etc".

Det program som skal startes skal have fuld sti udpegnings til de lydfiler det bruger, og bør ikke have testudskrifter i shell da det kan 'drille' ifm terminal kommandoer.

Her bruges LKS2Mb-Case3.py som eksempel og på SD image med programmer tilføjet er det forberedt.

Hvis man glemmer "&" tegnet i slutningen så vil processen med programmet ikke få sin egen tråd og vil blokere for at man kan komme tilbage og få lavet noget om.

Kommandoerne som bruges for at navigere i direktoriestrukturen er

cd ..	Change direktorie efterfulgt af mellemrum og 2 gange '' for at gå eet niveau op.
cd <doktorie navn>	Change direktorie efterfulgt af et direktorie navn for at gå ind i et direktorie.
cd /<doktorie navn>	Change direktorie til det som er i navn fra root. Ex.: "cd /etc"
	bringer dig direkte til etc direktoriet uanset hvor du står.
dir	for at se hvad der ligger i direktoriet man står i
sudo	som prefix inden ex. åbning af fil med editor for at få lov til at ændre – gør efterfølgende med direktorie rettigheder
nano <fil navn>	Simpel texteditor.

For flere kommandoer se:

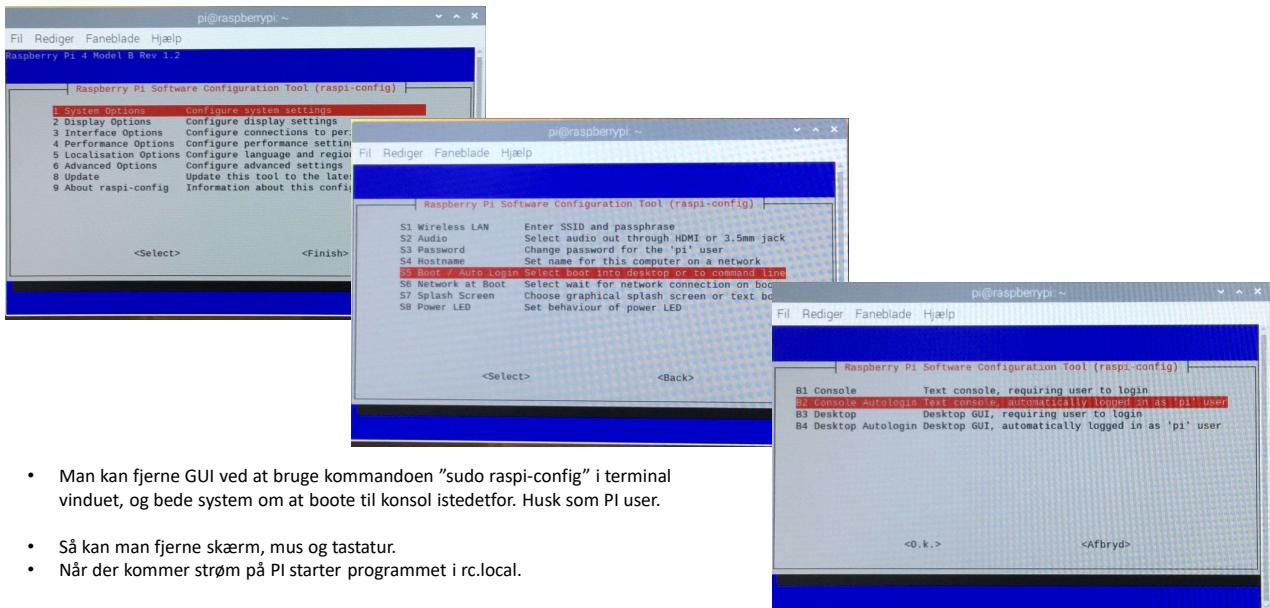
[42 of the Most Useful Raspberry Pi Commands \(circuitbasics.com\)](#)

Og med flere detaljer:

[Raspberry Pi Documentation - Using Linux](#)

# Spor 2 Option

Headless opstilling – Program der starter fra boot på Raspberry Pi.



Fjern GUI så PI starter op i terminal mode.

Når man har tilføjet program til start fra boot ved at tilføje det til rc.local kan man reboote og program vil starte uagtet der også starter en fuld GUI, og man kan så 'bare' fjerne tastatur og skærm.

BEMÆRK at den process man har startet tager kontrollen over lydkort og seriell input, så man kan ikke samtidigt starte et nyt python program som bruger lyd og seriell port!

MEN for at gøre det færdigt kan man fjerne GUI delen og boote i ren konsol/terminal mode.

Det kan man sikre vha raspi-config.

HUSK at brug PI som bruger.

Man kan så altid komme den anden vej og sætte GUI på igen vha. "sudo raspi-config" i konsol vinduet (når der er skærm og tastatur sat på PI).

Og man kan fjerne program fra boot sekvens ved at ud kommentere det i rc.local.

# Spor 2 Option

Headless opstilling – Program der starter fra boot på Raspberry Pi.

- Lydkort som default output device for lyd

- Gå ind i terminal vindue
- Skift til boot direktorie – "cd /boot"
- Åbn config.txt og tilføj lydkort, som vist.

```

pi@raspberrypi:~$ nano /boot/config.txt
# Uncomment some or all of these to enable the optional hardware interfaces
#dtoverlay=lgc_armon
#dtoverlay=lgc_oh
#dtoverlay=lgc_oh

# Uncomment this to enable infrared communication.
#dtoverlay= gpio-ir,gpio_pin=17
#dtoverlay= gpio-ir-tx,gpio_pin=18

# Additional overlays and parameters are documented /boot/overlays/README

# Enable audio (loads snd_bcm2835)
#Lægge til denne linje før den valgt af lydkort nedenfor igennem vil give mulighed for at skifte lyd mellem headphones og Headphone (Jack stik) og Høre via GUI. (samtid Ignaudio hvis den er med)
# MEN det gør IKKE ifm Headless - Der skal dem være udkommenteret!
# dtoverlay=audio=0

# For at sikre lydkort virker - også sør venligst
# Brug hifiberry-dacplus med et dacplus modul (2 kanaler forstærker)
# Brug hifiberry-dac med et Recreate Modul (4 kanals forstærker)
# Brug i2audio-dacplus med et i2audio Modul (2 kanals forstærker valgt som det bedste til LydKit)

#dtoverlay=hifiberry-dacplus
#dtoverlay=hifiberry-dac
dtoverlay=i2audio-dacplus,auto_mute_amp

[d14]
# Enable DRM VC4 V3D driver on top of the dispmanx display stack
dtoverlay=vc4-fkms-v3d
max_framebuffers=2

```

For at køre headless og sikre at lyd kommer ud gennem det tilsluttede lydkort på raspberry PI, skal man sikre at lydkortet er default lyd kilde (At PI ikke bruger jack stik som output device).

Har man lavet opstilling med SD image fra lyd-kit projektet er det på plads. Men for god ordens skyld repeteres det lige her.

- Gå ind i terminal vindue
- Skift til boot direktorie - cd /boot
- Åbn config.txt og tilføj lydkort, som vist i slide. – sudo nano config.txt

Projekt støttet af:



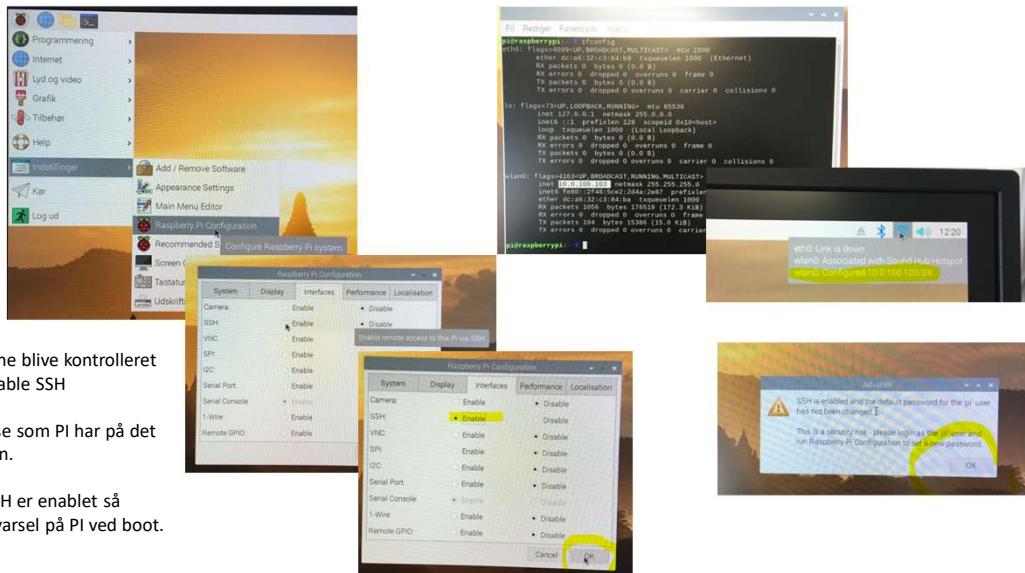
# Option

Fjernkontrol.

Start/stop programmer uden skræm og tastatur tilkoblet Raspberry PI,  
over netværk vha. terminalvindue på PC/MAC

# Spor 2 Option

Headless opstilling – Fjernkontrol.



- Sæt PI op til at kunne blive kontrolleret over net, ved at enable SSH
- Aflæs den IP adresse som PI har på det lokale net. Husk den.
- Bemærk at efter SSH er enablet så kommer der en advarsel på PI ved boot.

## Fjernkontrol – clips fra PI

Sæt SSH op via menu.

Find IP adr. enten via terminal vindue eller på netsymbol i GUI og noter det ned – Skal bruges fra PC/MAC til fjernkontrol

Bemærk at når SSH er sat til – så vil der komme en advarsel på GUI ved efterfælgende boot til GUI – Den skal man bare ignorere.

[How to Enable SSH on Raspberry Pi \[Definitive Guide\] \(phoenixnap.com\)](https://www.phoenixnap.com/knowledgebase/article/1172/how-to-enable-ssh-on-raspberry-pi)

SSH historie:

[Secure Shell - Wikipedia, den frie encyklopædi](https://en.wikipedia.org/wiki/Secure_Shell)

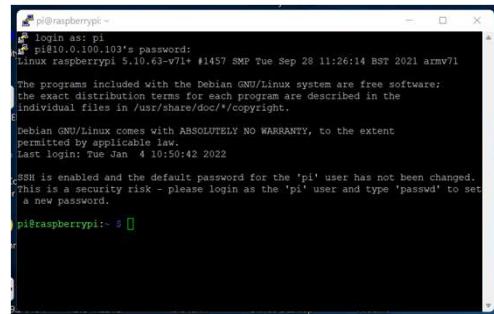
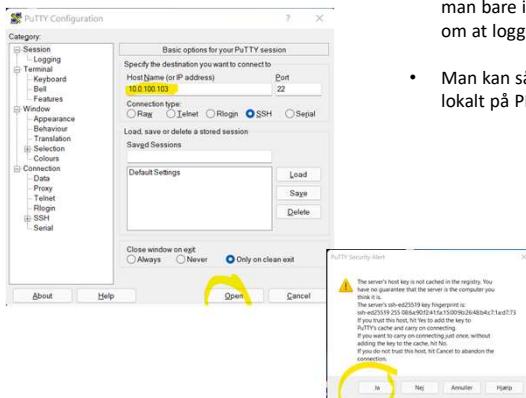
# Spor 2 Option

Headless opstilling – Fjernkontrol.

Projekt støttet af:



- På PC/MAC kan man så køre et terminal program, som over net har adgang til text terminal/konsol på PI. På Pc kan man bruge programmet Putty.
- Når man åbner forbindelsen ved at indtaste IP adressen får man en advarsel. Den skal man bare ignorere. Derefter kommer man til et terminal vindue, hvor man bliver bedt om at logge ind. Brug PI og raspberry som login og Password. (default)
- Man kan så fra terminal vinduet lave de samme ting som man kan i terminal/konsol lokalt på PI. Ex. editere i rc.local.



## Fjernkontrol på PC – via Putty

Putty kan downloades fra:

[Download PuTTY - a free SSH and telnet client for Windows](#)

### BEMÆRK

- Første gang man logger ind kommer der en Security Alert. Den skal man bare ignorere.
- Man bliver bedt om login og password. Det er “PI” og “raspberry” med mindre det er lavet om. (default burger og password)

# Spor 2 Option

Headless opstilling – Fjernkontrol, Hvis man vil stoppe en process.

Projekt støttet af:



The image shows two terminal windows. The left window displays a list of processes using the command 'ps -A'. A red arrow points to the process 'python3' with PID 467. The right window shows the command 'sudo kill 467' being entered and executed, stopping the process.

```
391 ? 00:00:00 dbus-daemon
392 ? 00:00:00 rngd
411 ? 00:00:00 wpa_supplicant
415 ? 00:00:00 thd
441 ? 00:00:00 dhcpcd
453 ? 00:00:00 cups-browsed
464 ? 00:00:00 epmd
467 ? 00:00:13 python3 ←
482 ttym1 00:00:00 login
497 ? 00:00:00 wpa_supplicant
523 ? 00:00:00 systemd
524 ? 00:00:00 (sd-pam)
535 ttym1 00:00:00 bash
550 ? 00:00:00 kworker/u9:1-hci0
552 ? 00:00:00 hciattach
553 ? 00:00:00 kworker/u9:2-hci0
567 ? 00:00:00 bluetoothd
625 ? 00:00:00 kworker/3:3-events_power_efficient
627 ttym1 00:00:00 ps
pi@raspberrypi:~ $ ps -A

root 467 5.1 3.3 78516 29784 ?
root 482 0.0 0.3 5620 2772 ttym1
root 497 0.0 0.3 10968 3292 ?
pi 523 0.0 0.8 14668 7308 ?
pi 524 0.0 0.1 16708 1744 ?
pi 535 0.0 0.4 8492 3620 ttym1
root 550 0.0 0.0 0 0 ?
root 552 0.0 0.0 2140 128 ?
root 567 0.0 0.3 9536 3340 ?
root 625 0.0 0.0 0 0 ?
root 629 0.0 0.0 0 0 ?
pi 630 0.0 0.2 9788 2488 ttym1
pi@raspberrypi:~ $ sudo kill 467
pi@raspberrypi:~ $
```

Med fjernkontrol kan man det samme som man kan fra terminalvinduet lokalt på PI.  
Dvs. man kan editere i filer med  
"sudo nano <fil navn>"  
Skifte direktorier osv.

Man kan eksempelvis rette i rc.local, hvis man vil have en anden fil til at starte fra boot – eller ingen.  
Man kan Reboote system osv.

Man kan også slå kørende processer ned med kommandoerne:  
"ps -A" Giver oversigt over kørende processer med PID Den 'vi' starter i rc.local vil have navnet "python3"  
"sudo kill <PID>" Ved hjælp af "kill" kommando og PID nummer kan man så slå en proces ned (stoppe den).

Det kan man både gøre fra terminal vindue på PI og via remote terminal vindue på PC/MAC.

Projekt støttet af:



# FAQ

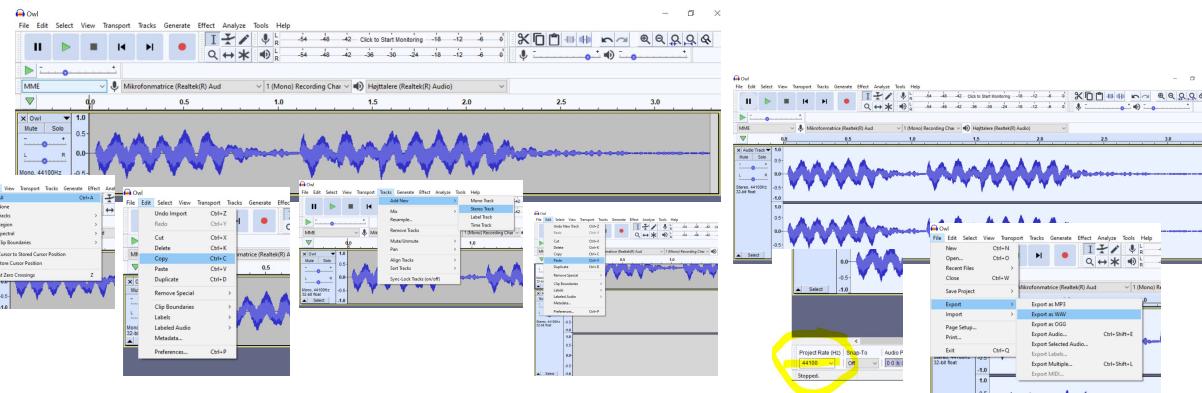
## Spor 2: FAQ – Kan man bruge alle lydfiler?

NEJ – Der kan være problemer med formaterne. Brug wav stereo filer!

BEMÆRK at ikke alle wav filer virker, og der kan også være forskel på hvilke der virker på hhv. PC/MAC og PI!

- Ex. Mono fil virker på PC men ikke på PI, Der kan være problemer med forskellige sampling rates etc. Stereo i 44100Hz virker på begge.

PRØV – hvis det ikke virker så brug ex. Audacity til at konvertere så det kommer til at fungere



Audacity kan findes på:

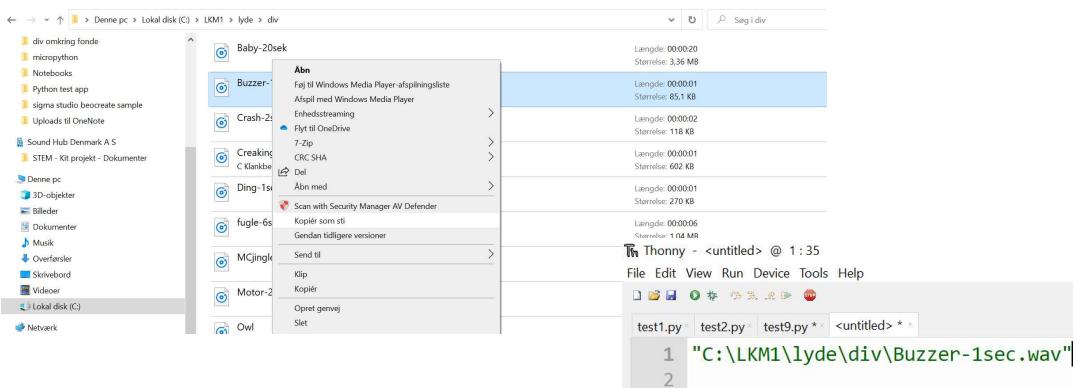
[Download | Audacity ® \(audacityteam.org\)](http://audacityteam.org)

## Spor 2: FAQ – Sti på PC (windows 10)

Hvordan finder jeg den rigtige sti på PC

Find placering/fil i file explorer. Marker filen. Tryk og hold shift samtidigt med højreklik. Så kommer kontekst menu med mulighed for at kopiere sti. Brug ctrl-v i Thonny til at få fuld sti incl. fil navn kopieret ind.

**BEMÆRK at sti så er med backslash** – Det bør ændres til forwardslash eller dobbeltslash for ikke at få andre problemer!!  
Se selvstændig slide om det.



Så længe man arbejder med lydfiler og program filer i samme folder, og man kører program fra Thonny så behøver man ikke at kende den fulde sti til hverken program eller lydfiler!

Så det er det letteste.

Hvis man vil organisere sine lydfiler i andre direktorier skal man angive fuld sti til lydfiler.  
Hvis man vil have en headless opstilling til at køre skal man også angive den fulde sti.

Ved angivelse af fuld stil kan der være issues med specialtegn som kan drille, og der er forskel på PC/MAC og PI.

## Spor 2: FAQ – Sti og filnavne

Man kan også arbejde med relativ file path i sine programmer, og dermed gøre dem mere ‘robuste’ ift. placering.

Man kan så eksempelvis have en folder med programmet og under det have en underfolder med lydfiler.  
Hvor programdirektoriets sti er placeret (på forskellige maskiner) kan man så finde program teknisk.  
Det dykker vi ikke mere ned i her, men man kan prøve nedenstående kode (på både PC/MAC og PI)

```
1 import os
2 filnavn_med_fuld_sti = os.path.abspath(__file__)
3 print(filnavn_med_fuld_sti)
4 # giver direktoriennavnet på hvor program filen ligger,
5 # hvor den udkommenterede ovenfor giver direktorie og filnavn.
6 hvor_program_ligger = os.path.abspath('.')
7 print(hvor_program_ligger)
8 # vil virke cross platform da der ikke er system specifik separator med i denne!!
9 fil_i_underfolder = os.path.join(hvor_program_ligger,'lyde','lyd1.wav')
10 print(fil_i_underfolder)
```

Hvis man vil vide mere om os.path

[os.path — Common pathname manipulations — Python 3.10.1 documentation](#)

PS – Se også ”os.sep”, som giver path separator for det specifikke operativsystem. ”/” for raspberry og ”\\” for windows. Det betyder at man kan lave noget som kan virke på tværs af

PC/MAC/PI ved at ex lave:

```
p      = os.path.abspath('.')
lokalfilnavn = p+os.sep+'lyd1.wav'
```

## Spor 2: FAQ – Sti på MAC (OSX)

Hvordan finder jeg den rigtige sti på MAC

Find fil og marker den – Brug context menu (to fingre og tryk ??). Brug kopi som sti. Kopier til Thonny.

**CHECK om sti så er med backslash** – Det bør ændres til forwardslash eller dobbeltslash for ikke at få andre problemer!!  
Se selvstændig slide om det.

### Copy File Paths From The Context Menu

The context menu on your Mac is a really powerful tool as [it lets you do much more with your files than just letting you rename or delete your files](#).

One of the useful and hidden options in your context menu lets you directly copy the path of a file to your clipboard.



Since it's hidden by default, it won't appear when you right-click on a file on your Mac. However, unhiding the option is pretty easy and all it takes is pressing and holding down the Option key. This will make the option visible in your context menu.

To use the option, right-click on a file in the Finder, hold down the Option key on your keyboard, and you'll see the **Copy <file-name.ext> as Pathname** option. Click on it to copy your file's path.

It will have copied the path of your selected file as plain text to your clipboard.

Så længe man arbejder med lydfiler og program filer i samme direktorier, og man kører program fra Thonny så behøver man ikke at kende den fulde sti til hverken program eller lydfiler!

Så det er det letteste.

Hvis man vil organisere sine lydfiler i andre direktorier skal man angive fuld sti til lydfiler.  
Hvis man vil have en headless opstilling til at køre skal man også angive den fulde sti.

## Spor 2: FAQ – forward "/" eller backwardslash "\"

Er der forskel på at bruge forward- eller backward slash i sti navne når lydfiler skal udpeges?

JA- Backslash udpeger specialtegn i kombination med andre tegn så den skal man være forsigtig med  
– Kør slashtest.py i Thonny fra til illustration.

```

# Kode sti til at illustrere forskel med backslash "\" og forwardslash "/"
# Det ser umiddelbart neget tilforladeligt ud. MEN bemerk udskriften for t6
# - ikke som forventet backslash \s men et speciel tegn - som backspace BS!!!!
# - Det er Fordi \o tolles som et speciel tegn - som backspace BS!!!!
#
# Og bemerk også udskriften for t13-t18
# - Der kommer kun en backslash \ alle udskrifter selvom der står to efter hinanden i konstanternel
#
# Bemerk som det sidste at det kan ikke lade sig gøre at slutte med backslash
# - så får man en \e1 - prøv det!
#
# Konklusion - Brug forward slash eller dobbelt backslash i sti navne!!
#
# Hvorfor kommer man til at bruge backslash?
# - Fordi det er hvordan man får nogen til at koperer sti navn i fil/folder system!
# Hvis man virkelig det ønsker kan man nemt skifte backslash
# - Fordi det afhænger af det tegn/bogstav som kommer efter!
# - se hvilket i https://docs.python.org/2.8/ref/strings.html

t1 = "tekst streng med backslash \\ efterfulgt af space"
t2 = "tekst streng med backslash \\\\ efterfulgt af \ som i LYD"
t3 = "tekst streng med backslash \\\\ efterfulgt af \ som i lyde"
t4 = "tekst streng med backslash \\\\ efterfulgt af \ som i \d"
t5 = "tekst streng med backslash \\\\ efterfulgt af h som i hei\ol"
t6 = "tekst streng med forwardslashes / efterfulgt af space"
t7 = "tekst streng med forwardslashes // efterfulgt af \ som i LYD"
t8 = "tekst streng med forwardslashes // efterfulgt af \ som i lyde"
t9 = "tekst streng med forwardslashes //\\ efterfulgt af \ som i \d"
t10 = "tekst streng med forwardslashes //\\ efterfulgt af h som i hei\ol"
t11 = "tekst streng med forwardslashes //\\ efterfulgt af s som i baggrund"
t12 = "tekst streng med forwardslashes //\\ efterfulgt af s som i baggrund"
t13 = "tekst streng med dobbelt backslash \\\\" efterfulgt af space"
t14 = "tekst streng med dobbelt backslash \\\\" efterfulgt af \ som i LYD"
t15 = "tekst streng med dobbelt backslash \\\\" efterfulgt af \ som i lyde"
t16 = "tekst streng med dobbelt backslash \\\\" efterfulgt af \ som i \d"
t17 = "tekst streng med dobbelt backslash \\\\" efterfulgt af h som i hei\ol"
t18 = "tekst streng med dobbelt backslash \\\\" efterfulgt af h som i hei\ol"

n1 = [t1,t2,t3,t4,t5,t6,t7,t8,t9,t10,t11,t12,t13,t14,t15,t16,t17,t18]

for k in range(len(n1)):
    print("t"+str(k+1)+": "+n1[k])

```

Escape Sequence	Meaning
\n\nline	Ignored
\\\	Backslash (\)
\'	Single quote ('')
\"	Double quote (")
\a	ASCII Bell (BEL)
\b	ASCII Backspace (BS)
\f	ASCII Formfeed (FF)
\n	ASCII Linefeed (LF)
\r	ASCII Carriage Return (CR)
\t	ASCII Horizontal Tab (TAB)
\v	ASCII Vertical Tab (VT)
\ooo	ASCII character with octal value ooo
\xhh..	ASCII character with hex value hh..

```

>>> %%Run slashtest.py
t 1 : tekst streng med backslash \\ efterfulgt af space
t 2 : tekst streng med backslash \\\\ efterfulgt af \ som i LYD
t 3 : tekst streng med backslash \\\\ efterfulgt af \ som i lyde
t 4 : tekst streng med backslash \\\\ efterfulgt af \ som i \d
t 5 : tekst streng med backslash \\\\ efterfulgt af h som i hei\ol
t 6 : tekst streng med forwardslashes /\\ efterfulgt af \ som i baggrund
t 7 : tekst streng med forwardslashes //\\ efterfulgt af \ som i LYD
t 8 : tekst streng med forwardslashes //\\ efterfulgt af \ som i lyde
t 9 : tekst streng med forwardslashes //\\\\ efterfulgt af \ som i \d
t 10 : tekst streng med forwardslashes //\\\\ efterfulgt af h som i hei\ol
t 11 : tekst streng med forwardslashes //\\\\ efterfulgt af h som i hei\ol
t 12 : tekst streng med forwardslashes //\\\\ efterfulgt af h som i baggrund
t 13 : tekst streng med dobbelt backslash \\\\"\\ efterfulgt af \ som i LYD
t 14 : tekst streng med dobbelt backslash \\\\"\\ efterfulgt af \ som i lyde
t 15 : tekst streng med dobbelt backslash \\\\"\\ efterfulgt af \ som i \d
t 16 : tekst streng med dobbelt backslash \\\\"\\ efterfulgt af h som i hei\ol
t 17 : tekst streng med dobbelt backslash \\\\"\\ efterfulgt af h som i hei\ol
t 18 : tekst streng med dobbelt backslash \\\\"\\ efterfulgt af h som i baggrund

```

Projekt støttet af:



# LINKS

## Spor 2

Links til supplerende materiale

Projekt støttet af:



Fra slide 3 – gentagelse af links fra spor 1 til fundament omkring udviklingsmiljø:

[Thonny, Python IDE for beginners](#)

[Python Tutorial \(w3schools.com\)](#)

[BBC micro:bit MicroPython documentation — BBC micro:bit MicroPython 1.0.1 documentation \(micropython.readthedocs.io\)](#)

Fra Slide 2 – HW referencer

[Buy a IQaudio DAC+ – Raspberry Pi](#) – Fungerer direkte med Raspberry OS. Og SD image til LYD-Kit.  
[HiFiBerry Amp2 | HiFiBerry](#) - Kræver en ændring i config.txt- (se slide 31)

Begge kan købes i forskellige danske online butikker. Ex.

[HiFiBerry AMP2 • RaspberryPi.dk](#)  
[IQaudio DigiAMP+ Forstærker • RaspberryPi.dk](#)

[PLS-P830983 - Højtalere & PA - FreQuence.dk](#)

Her er nogle links til supplerende materiale som kan bruges i spor 2, inklusiv dem som er brugt i teknologi projektoplæg omkring lyd.

(De er i teknologi projekt oplæg dokumentet, men gentages her så de er lettere at genbruge til andet materiale)

Gode at give som lektier – og så stille lidt spørgsmål til dem – for at sikre eleverne faktisk ser dem!!

## Spor 2

Links til supplerende materiale

Projekt støttet af:



Fra Teknologi projekt oplæg – Lyd i det offentlige rum – Introduktion:

[Julian Treasure: The 4 ways sound affects us | TED Talk](#)

(Bemærk at Nokia lyden ikke er så kendt i dag, som den var i 2009, men fortsat en god video om lyd!)

[Julian Treasure: Why architects need to use their ears | TED Talk](#)

Fra Teknologi projekt oplæg nummer 1 - Lyd og liv i byrummet:

[Everyday Listening - Sound Art, Sound Installations, Sonic Inspiration](#)

[305\\_Christine\\_Kerrigan.pdf \(urban-sound-symposium.org\)](#)

Fra Teknologi projekt oplæg nummer 2 - Lyd som element til at påvirke adfærd:

[Harnessing the Power of Sound | Steve Keller | TEDxNashville – YouTube](#)

[Nudge techniques: 6 simple strategies to influence behaviour \(hatrabbits.com\)](#)

[Adfærdsøkonom nudger fulde folk på Distortion - Københavns Universitets Videoportal \(ku.dk\)](#)

Fra Teknologi projekt oplæg nummer 3 - Lyd og læring:

[Movie QUIZ \( movie sound clips\) – YouTube](#)

[Sounds of the World - An Audio Quiz - True Travel](#)

[The Super Mario Effect - Tricking Your Brain into Learning More | Mark Rober | TEDxPenn - YouTube](#)

Her er nogle links til supplerende materiale som kan bruges i spor 2, inklusiv dem som er brugt i teknologi projektoplæg omkring lyd.

(De er i teknologi projekt oplæg dokumentet, men gentages her så de er lettere at genbruge til andet materiale)

Gode at give som lektier – og så stille lidt spørgsmål til dem – for at sikre eleverne faktisk ser dem!!

Projekt støttet af:



## Spor 2

Links til supplerende materiale

Vedrørende Raspberry OS og terminal/kommando linje (slide 35):

[Raspberry Pi Documentation - Using Linux](#)  
[42 of the Most Useful Raspberry Pi Commands \(circuitbasics.com\)](#)

Slide 43 (Bearbejdning af lydfiler. Klip/tilpas længder, lav til mono/strereo, lav om i samplerate etc.)  
[Download | Audacity® \(audacityteam.org\)](#)

Vedrørende fjernadgang (slide 39 og 40)  
[How to Enable SSH on Raspberry Pi \[Definitive Guide\] \(phoenixnap.com\)](#)  
[Raspberry Pi Documentation - Remote Access](#) (avanceret)  
[Secure Shell – Wikipedia](#)

[Download PuTTY - a free SSH and telnet client for Windows](#)

Vedr. File path (slide 45)  
[os.path — Common pathname manipulations — Python 3.10.1 documentation](#)

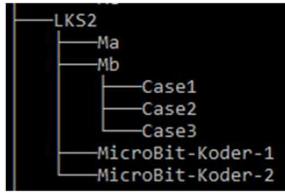
Projekt støttet af:



# Andet

# Spor 2

Supplerende materiale – Oversigt over filer og direktoiestruktur til Spor 2



bagg-flood.wav	a.wav	b-a.wav	01.wav	MBType-0.py	MBType-A.py
bagg-trafik.wav	b.wav	b-b.wav	02.wav	MBType-1.py	MBType-B.py
buu-trombone.wav	c.wav	c-a.wav	04.wav	MBType-2.py	MBType-C.py
cheer-crowd.wav	d.wav	d-b.wav	11.wav	MBType-3.py	MBType-D.py
hare.wav	e.wav	e-c.wav	12.wav	MBType-4.py	MBType-K.py
LKlib.py	f.wav	f-d.wav	14.wav	MBType-5.py	MBType-M.py
LKS2Ma-Afspil-lyde.py	g.wav	g-LKlib.py	21.wav	MBType-6.py	MBType-T.py
LKS2Ma-MBinput-1.py	h.wav	LKS2Mb-Case2.py	22.wav		
LKS2Ma-MBinput-2.py	i.wav	v-a.wav	24.wav		
skovskade.wav	j.wav	v-b.wav	31.wav		
solsort.wav	k.wav	v-c.wav	32.wav		
start-seq.wav	l.wav		34.wav		
ugle.wav	m.wav		forkert.wav		
	n.wav		fugle pix lille.pptx		
	o.wav		LKlib.py		
	p.wav		LKS2Mb-Case3.py		
	q.wav		rigtig.wav		
	r.wav		start-seq.wav		

Hvordan man håndterer uddeling af filer er op til den enkelte skole.

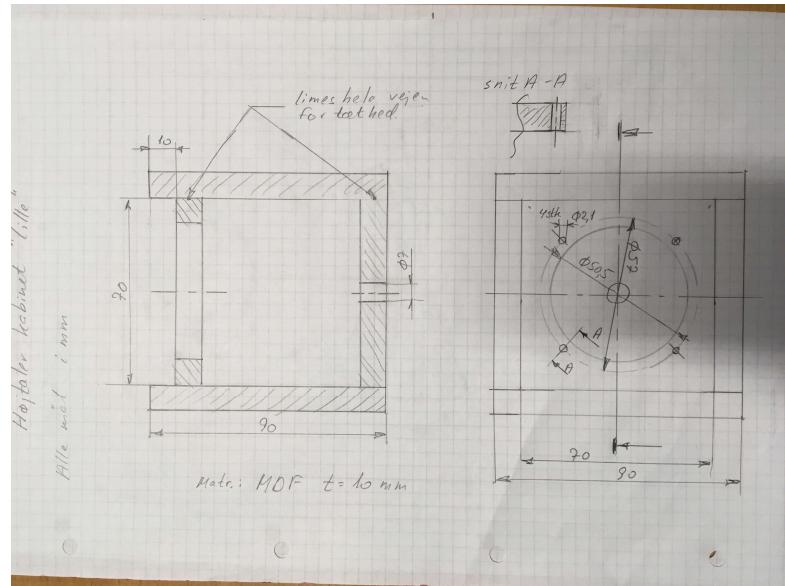
I udviklingsprojektet brugte vi en teams kanal med den enkelte klasse/gruppe, hvor vi lagde filer ud til klassen/gruppen efter behov.

Og havde en ‘master’ kanal for lærerne hvor det hele lå samlet (read only)

I klassens teams kanal kan man så have en dynamisk tilgang hvor evt. gruppe debatter også kan tages,  
og bruges til udveksling af kode eksempler som der arbejdes med i de enkelte grupper.

## Spor 2

Supplerende materiale – Forslag/mål på højtalerkabinet til anbefalet enhed PLS-P830983



Mål på kabinet som blev lavet til den anbefalede Højtalerenhed.

Kabinet lavet til udviklingsprojektet blev lavet af ViaNova i Stuer.

De forbedrede konstruktionen i forhold til tegningen.