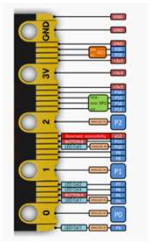
LYD-Kit Spor 1 (LKS1)

Byg med sensorer

Modul b – "Værkstedsmodul"

- Perspektivering omkring indbyggede sensorer.
- Brug radio mellem Micro:Bit's.
- Byg med eksterne kredsløb.

Projekt støttet af: midt regionmidtjylland



Til lærer som info/inspiration.
En skabelon til et værkstedsforløb i Teknologifag, eller elementer herfra kan bruges i andre forløb for eksempel i Fysik B omkring sensorer og el-teknik (se ex slide 26/27)

Kan bruges som et udvidelses modul, hvor man går mere i dybden med programmering i MicroPython og opbygning af knapper og evt. elektronik med Micro:Bitten som omdrejningspunkt.

Med hensyn til programmeringsdelen henvises til de samme opslagsværker som til Modul a:

Python Tutorial (w3schools.com) og BBC micro:bit MicroPython documentation — BBC micro:bit MicroPython 1.0.1 documentation (microbit-micropython.readthedocs.io)
(Klikbare links på slides sidst i stakken – De er desværre ikke klikbare når de er i noterne! – der er referencer til disse med links i slides undervejs.)

Dækker Spor 1 Modul B i LYD-Kit materialet. Der er selvstændige slide stakke for Modul A og Modul B

Slide stak er tænkt som inspiration til lærer – ikke som et færdigt forløb overfor eleverne, men det kan bruges som skabelon.

Version 23-Feb-2022

- Tilføjet elementer til modul b der kommer tættere på 'metallet' så der er muligheder for at koble mere til fysik og elektronik.

Version 14-Feb-2022

- Modul A,B og C som selvstændige slide stakke.

Version 04-Feb-2022

Rettet/tilføjet sådan at det kan bruges som 3 forløb af kortere varighed. Og lidt mere om grundelementer i programmering i Modul A.

- Modul a – som det basale og nødvendige for at kunne bruge LYD-Kit materialerne med en basal forståelse for hvad der sker i koden – 'Værkstedskørekort'
- Modul b – som et værkstedsmodul for dem som vil vide mere/gå dybere i Microbitten for dem som vil bygge selv med Micro:Bitten.
- Modul c – som en case, hvor man med udgangspunkt i færdig kode kan lave funktions kritik, teknologivurdering og produktdesign.

Version 23-Nov-2020

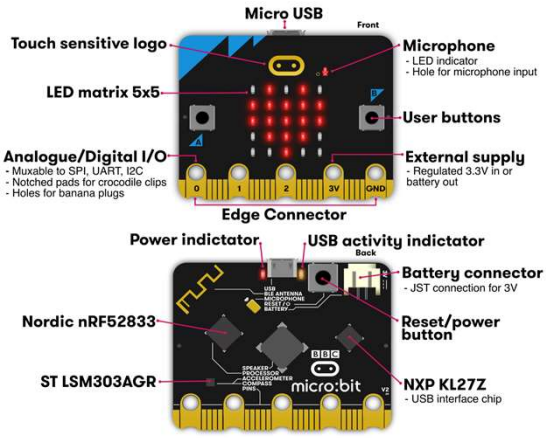
- Lagt op efter test med 1x og 2x på Struer Statsgymnasium. Der blev alle 3 moduler kørt som et samlet forløb på 10 lektioner.

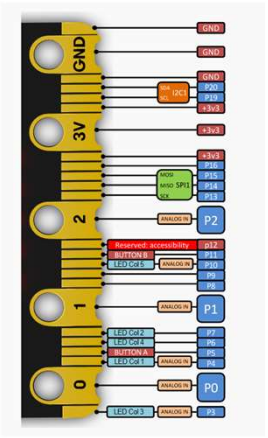
BEMÆRK at LYD-Kit er organiseret sådan at man IKKE behøver at komme dybt i programmeringen, da der er en række færdige programmer og skabeloner, som man kan bygge ovenpå – både unde at ændre i kode og ved at lave simple modifikationer. Man skal dog have en basis forståelse for Python og et såkaldt udviklingsmiljø (IDE – Integrated Development Environment) for at kunne benytte de færdige programmer. Det er det vi kalder Lukket bog anvendelse. Det er omvendt også muligt at bruge det som 'åben bog' til at gå dybere i programmeringsdelen, hvis man vil. Derved understøttes den didaktiske tilgang – Faded Guidance, og UMC (Use Modify Create).

Modul B er et "Åben bog" modul for dem der vil dybere.

Spor 1 Modul b: Micro:Bit V2 – edge connector og radio

Projekt støttet af:





Micro:Bit i tidligere version (V1) havde ikke MEMS mikrofon og højtaler på, så der skulle man selv tilføje dem via Edge Connector.
Et eksempel på at sætte nye funktioner på print (PCB) når man får nye teknologiske muligheder. (MEMS).
PIN's på V1 er tilsvarende.

I Spor 1 - Modul A var fokus på de indbyggede sensorer i Microbitten.

Modul B har fokus på hhv. Radio og Edge Connector, hvortil man kan bygge egne elektroniske 'kredsløb'.

- Med krokodillekabler (Direkte på PIN0,1,2, 3V og GND),
- Med udvidelses boards og breadboards hvor man kan opbygge tidlige modeller af elektriske kredsløb på samme måde man gør med diverse arduino boards.
- Ved at lodde direkte på PIN's hvis man synes.

PIN's på Edgeconnector kan benyttes både til input og output, og det både som digital og analog, samt en I2C bus kommunikation så mere avancerede komponenter kan kobles til – ex. displays, men også digitale sensorer der kommunikerer deres værdier over I2C bussen.

I denne gennemgang er der fokus på basal digital og analog input, samt på de 3 'store' PIN's, samt 3V og GND. der kan bruges sammen med krokodille kabler.

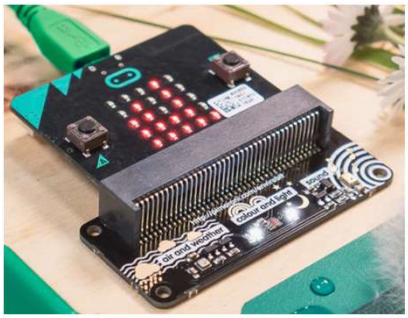
I forbindelse med LYD-Kit fokuseres på input delen, sådan at man kan koble alle mulige sensorer på og sende værdier videre over radio.
Det benyttes i de færdige software byggeklodser i Spor 2.

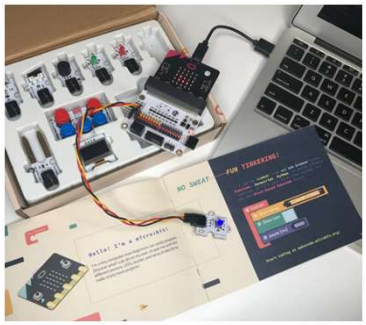
MEN man kan altså også bygge videre på output delen og kontrollere motorer, relæer osv. hvis man måtte ønske.

Man kan også købe en række udvidelsesmoduler der giver nye sensor muligheder og/eller mere færdige løsninger som eksempelvis Elekfreaks Tinker KIT og Pimoronis Enviro:Bit. (Se næste slide)

Spor 1 Modul b: Ex på udvidelses kits.

Projekt støttet af:
midt regionmidtjylland





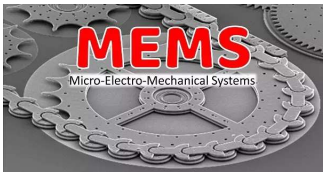
For at give et billede på de eksempler der er nævnt i noterne på forrige side.

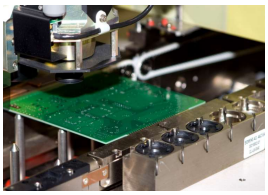
Klikbare links på billederne til producenters hjemmesider. Og på slides sidst i stakken.

Der findes flere danske leverandører man kan finde ved en søgning hvis man ønsker.

Spor 1 Modul b: MEMS.

Projekt støttet af:





Mindre – men nok så vigtigt - en helt anden fremstillings proces og kan indgå i produktion af PCB (printed circuit board) på linje med IC'er af forskellig slags

Introduktion som kan bruges til at perspektivere det med sensorer ved at tale om udviklingen fra større Elektro-mekaniske systemer til Micro-Electro-Mechanical-Systems. (MEMS)

Der kan være en dialog omkring produktionsteknologier, og deres betydning for produkter.

Fra spåntagning og støbning af mekaniske komponenter til fremstilling med teknologier der svarer til litografi.

Teknologi, der bringer (noget) mekanik ned i en dimension der sammen med elektronik kan monteres på printplader (PCB's) med samme proces teknologi som integrerede kredse.

Det betyder en effektivisering/omkostnings reduktion i montage fremfor de 'større' komponenter som tidligere blev brugt for at 'interagere med den fysiske verden'.

Fra manuel montage til montage i makiner (SMT – Surface Mount Technology – en væsentlig grund til at elektronik er så billig at producere og at det kan være så småt. Se link på nederste billede til højre.

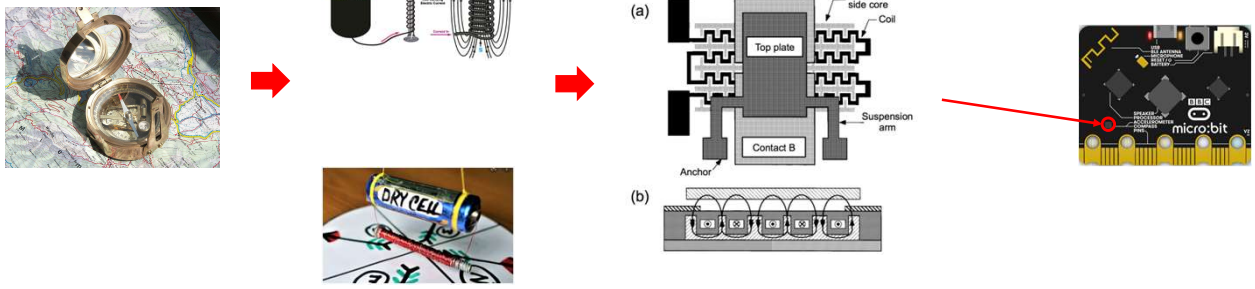
Selv højtaleren og mikrofonen fås nu som MEMS. Der er så en pris at betale mht. lyd kvalitet, og helt overordnet skal man naturligvis balancere kvalitet og omkostning i forhold til det produkt man ønsker at lave.

Micro:Bitten er ikke til at lave færdige produkter med, men et værktøj til at lære om sensorer og mikrocontrollere – og samtidigt et værktøj til hurtigt at bygge funktionsmodeller med.

Video om MEMS som klikbar link på figur med MEMS logo. – Kan evt. gives som hjemmeopgave.
The World Of Microscopic Machines – YouTube

Spor 1 Modul b: MEMS.

Projekt støttet af:



Eksempel man kan bruge til at perspektivere ud fra, hvis man synes

Kompasset brugt lang tid inden der blev fundet på at generere strøm.

Elektromagneter kan også bruges til maget. (Og meget andet – Blandt andet induktion)

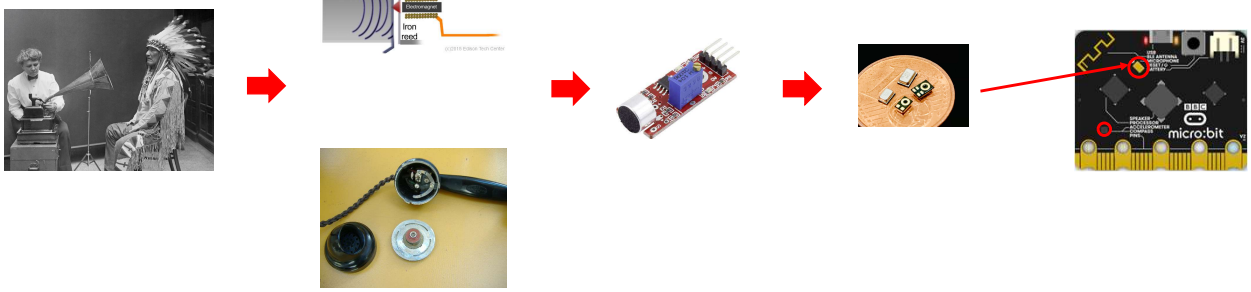
Nu lavet som MEMS der kan sættes på print.

Den der sidder på Micro:Bitten kan både måle et magnetfelt (og bruges til kompas) samt til at måle acceleration. Så man kan sige at de 2 naturkræfter: ElektroMagnetismen og Tyngdekræften; kan måles med den lille MEMS på Microbitten og laves om til elektriske signaler der igen laves om til digitale værdier som man kan behandle med kode i programmer. Så send lige en tanke til Ørsted og Newton; ☺ samt alle de ingeniører der har gjort MEMS'en mulig. ☺ ☺

Sådanne sensorer er der mange af i alle mulige produkter i dag. Fra smartphones til biler og i diverse produktionsudstyr.

Spor 1 Modul b: MEMS.

Projekt støttet af:



Eksempel man kan bruge til at perspektivere ud fra, hvis man synes

Mikrofonen

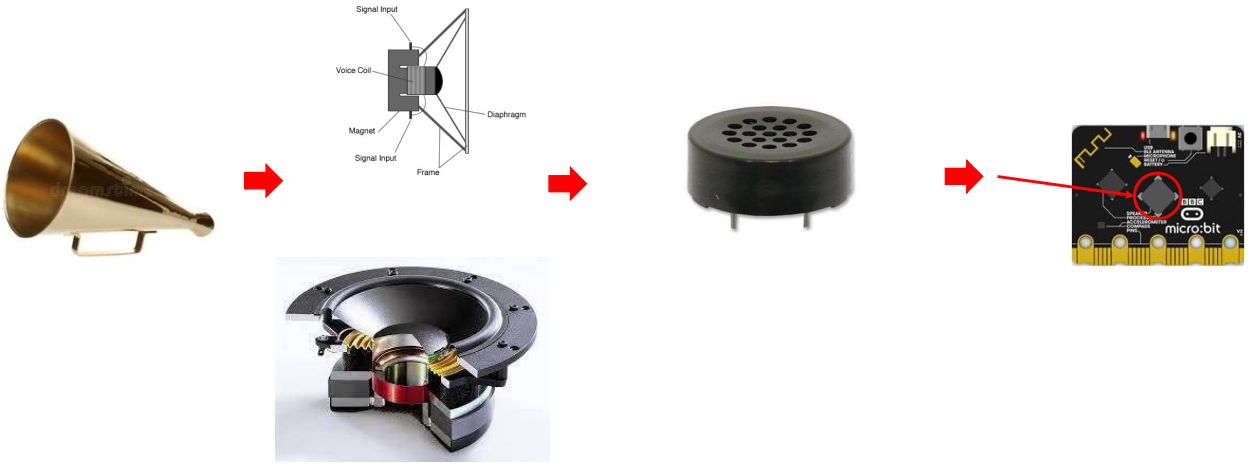
Inden elektronikken så lavede Edison en tragt der med en nål kunne ridse i en lakcylinder og derved 'optage' lyd, som kunne afspilles igen gennem den samme tragt når lakvalsen blev roteret med samme nål i rillen!

Elektromagneten (induktion) også grundlag for at lave lyd om til elektriske signaler – Brugt til telefonen lige inden skiftet til år 1900!

Nu fås mikrofoner også som MEMS enheder der kan sættes på print plader (PCB's) med samme proces teknologi som andre IC (integrerede kredse).
Og med ganske fornuftig kvalitet.

Spor 1 Modul b: MEMS.

Projekt støttet af:



Eksempel man kan bruge til at perspektivere ud fra, hvis man synes

Højtaleren

Også inden den elektroniske forstærker – Hænderne op foran munden og råb – Så kan man høre det langt væk!

Igen i forbindelse med elektromagnetismen så også højtaleren (i princippet en omvendt mikrofon) der kan lave elektriske signaler om til lyd.
Signalerne skal dog forstærkes for at det kan flytte nok luft til at man kan høre det.

Og selvom man nu kan få højtalere som MEMS, der kan monteres direkte på printkort (PCB) så er der grænser for hvor højt (og godt de kan spille).

Når man taler om earbuds – dvs når højtaleren kommer tæt på trommehinden er det en anden sag.

Spor 1 Modul b: Micro:bit og trådløs kommunikation.

Projekt støttet af: midt regionmidtjylland

Micro:bitten har også indbygget en antenne og en 'radio' sender så man kan kommunikere trådløst mellem Micro:bit's.

Der er mange slags trådløs kommunikation – Så her kan man tage en snak om det, men her er der tale om kommunikation mellem Micro:Bits med egen protokol

HW er forberedt på at kunne kommunikere via BlueTooth.

Til LYD-Kit og MicroPython benyttes en mindre krævende protokol, med en række grundlæggende funktioner, som er tilgængelige via radio biblioteksmodulet.
Se: BBC micro:bit MicroPython documentation — BBC micro:bit MicroPython 1.0.1 documentation (microbit-micropython.readthedocs.io)
(Bemærk både afsnittet i Tutorials og i API under label Radio)
Der er klikbar link til tutorial på billedet til venstre og til API til højre – men igen bedre at vise det direkte i dokumentationen.

De næste slides viser hvordan man kan introducere det til eleverne – Men igen så er der flere måder at gribe det an på.

Radio introduceres fordi den bruges i de færdige kodeeksempler der er lavet til Spor 2 og 4 i LYD-Kit.

Man kan bruge de færdige kode eksempler uden at kende til koden, men for dem som vil dybere er eksemplerne i de følgende slides tænkt som en introduktion.

Igen så husk opslagsværket – søg med radio

Spor 1 Modul b: Micro:bit og trådløs kommunikation.

Projekt støttet af: midt regionmidtjylland

```
# LKS1Mb-modtager-1.py
# Checker om der er kommet besked på default kanal hvert halve sekund. (500 msek).
# Der udskrives None i shell, hvis der ikke er modtaget noget.
# Er der modtaget noget udskrives det i shell.
from microbit import *
import radio

radio.on()

while True:
    besked = radio.receive()
    print(besked)
    sleep(500)
```

Der er en 'postkasse' hvorfra man kan hente modtagne beskeder.
Man får ikke besked på at der er kommet et brev, så man må ud og se om der er kommet noget.
Det bliver gjort hvert halve sekund.

Tømmer man postkassen oftere end postbuddet har været der med post modtager man ingenting. (None)
Tømmer man den for sjældent kan det ende med at postbuddet ikke kan aflevere posten fordi postkassen er overfyldt, og modtageren får ikke alle beskeder.

Man kan bruge de færdige kode eksempler i Spor 2 uden at kende til koden.

Modul B i Spor 1 er for dem som vil dybere.

Eksemplet i disse slides er tænkt som en basal introduktion, hvor eleverne kan få en basis forståelse for timing, og adressering, så de kan lave kommunikation indenfor en gruppe af Micro:bits uden at de forstyrrer andre i samme lokale.

For at holde det enkelt benytter eksemplerne her og de færdige programkoder i LYD-Kit Spor 2 bare kanal parameteren til at adskille grupper af Micro:Bits, selvom der er flere lag i adresseringen på radio der kunne bruges (se address og group som andre parametre man kan bruge i kaldet til radio.config() i API beskrivelsen).

Lad alle elever i klassen med hver deres Micro:bit koblet til deres PC/MAC loade eller indtaste koden i slide.
Forklar den kort.

Lav så eksempler på sender, som vist på de næste slides.
Spørg om forventning til det eleverne modtager og kør sender delen fra lærer PC/MAC på storskærm.

BEMÆRK: Linjen ”from microbit import *” er ikke nødvendig i radio eksemplerne, da der ikke bruges funktioner/metoder fra Micro:bit biblioteket/modulet, men er taget med for at holde konsistens med eksemplerne fra Modul A, og for at gøre det enkelt når der skiftes over til at lave blandede funktioner.

Spor 1 Modul b: Micro:bit og trådløs kommunikation.

Projekt støttet af:
midt regionmidtjylland

```python
# LKS1Mb-Sender-1.py
# Besked fra input sendt een gang for hver runde.
# på default kanal med default sendestyrke
from microbit import *
import radio

radio.on()

while True:
    besked = input("Indtast det som skal sendes: ")  # Venter på input fra shell
    radio.send(besked)
    sleep(500)          # Betyder ikke noget da program venter på input hver gang det kommer rundt
```

```python
# LKS1Mb-Sender-2.py
# Besked fra input sendt kontinuerligt (løbende) med
# på default kanal med default sendestyrke med 500msec interval
from microbit import *
import radio

radio.on()
besked = input("Indtast det som skal sendes: ")   # Venter på input fra shell

while True:
    radio.send(besked)
    sleep(500)              # Er vigtig nu da program IKKE venter på input hver gang det kommer rundt
                            # Bemærk forskel i modtager ift. sender-1.
```

```python
# LKS1Mb-Sender-3.py
# Besked sendt kontinuerligt på default kanal med default sendestyrke med 500msec interval.
# MEN med et løbenummer mellem 1 og 9, så man kan se om alle beskeder kommer igennem.
# Forsøg med forskellige ventetider og se resultatet på modtager siden.
from microbit import *
import radio

radio.on()
besked_nr = 1
besked = "Hej med jer "          # Her er besked bare skrevet direkte til variablen. Venter ikke på input

while True:
    radio.send(besked+str(besked_nr))
    print(besked+str(besked_nr))  # Når man sammensætter skal begge være strenge - derfor str() på integer
    besked_nr = besked_nr+1
    if besked_nr>9:
        besked_nr = 1
    sleep(500)                    # Betyder noget for hvor mange beskeder der bliver sendt over tid.
                                  # Kør med 1000 hhv 100 msec her og se hvad der sker på modtager siden.
```

Fra Lærers PC/MAC køres koderne fra :

LKSMb-Sender-1.py
LKSMb-Sender-2.py
LKSMb-Sender-3.py - Prøv med forskellige tider i sleep – 1000 hhv. 100msec.

Man kan have dem preloaded i faner i Thonny, så man ikke skal indtaste mens eleverne kikker på.

Forklar kort om koden. Spørg hvad eleverne forventer at se i modtager.
Kør program og se om det passer med forventning.

HUSK – at få fokus i shell når der skal indtastes én besked – ellers kommer det indtastede ind i koden hvad kan være uheldigt!

(Kan undgås ved bare at have besked liggende i variabel som i Sender-3, men så får man ikke vist standard input i Python og vist at program 'hænger' mens det venter)

Det er IKKE tænkt at denne slide skal vises! Bedre med koden på storskærm og med koden kørende.

Men man kan bruges den som opsamling/repetition.

BEMÆRK: Linjen ”from microbit import *” er ikke nødvendig i radio eksemplerne, da der ikke bruges funktioner/metoder fra Micro:bit biblioteket/modulet, men er taget med for at holde konsistens med eksemplerne fra Modul A, og for at gøre det enkelt når der skiftes over til at lave blandede funktioner.

Spor 1 Modul b: Kanaler til at skelne elevgrupper.

Projekt støttet af:
midt regionmidtjylland

```
radio.config(channel=7)  # Kanal der sendes på - kan være mellem 0 og 83
```

Man kan i koden sætte den radio kanal man kommunikerer på med linjen ovenfor. Der sættes kanal til nummer 7, som er det samme som den kanal der default kommunikeres på (hvis man udelader linjen i koden).

Man kan sætte kanal til en værdi mellem 0 og 83 (se dokumentation for MicroPython på Micro:bit)

Hvis man ikke er på samme kanal, som senderen, modtager man ikke beskederne.

Når I skal arbejde med sender og modtager i grupper i samme lokale og på samme tid SKAL i bruge forskellige kanaler i de forskellige grupper.

Bed elever om at tilføje linien fra slide i i modtager (LKSMb-modtager-1.py) som alle gerne skal have i deres egen udgave af Thonny på deres egen PC/MAC, med en tilkoblet Micro:Bit.

Vis at med argumentet 7 ”(channel=7)” så modtager de fortsat alle beskederne der sendes (med koden i Thonny på Lærer PC – som i Slide 10)

Del eleverne i grupper og giv hver gruppe en kanal som de skal bruge i det efterfølgende.

Lav en test hvor du som lærer sender besked til bestemte grupper og se at andre ikke får beskeden. Brug LKSMb-sender-4.py til det.

Det kan samtidigt bruges til at forklare om dictionaries i Python, som bruges i kode eksempler i Spor 2 og 4.

HUSK at antallet af indgange og kanaler skal matche antal elevgrupper og de uddelte kanaler.

Spor 1 Modul b: Kanaler til at skelne elevgrupper.

Projekt støttet af:
midt regionmidtjylland

```
# LKS1Mb-Sender-4.py
# Besked sendt på forskellige kanaler til forskellige elevgrupper
# Elevergrupper skal gerne kun modtage besked med det
# kanal nummer de har fået tildelt.

from microbit import *
import radio

radio.on()
# Dictionary med elevgruppe som nøgle (til opslag)
# og kanalnummer som det man får retur
elev_gr_til_k_nr = { # elevgruppe:  Kanalnummer:
                                1:  10,
                                2:  20,
                                3:  30,
                                4:  40,
                                5:  50,
                                6:  60,
                                7:  70,
                                8:  80
    }

# Default max 29 tegn - 3 til nummer. Altså 26 i første del af besked.
#          12345678901234567890123456789
besked = "Hej jer med kanalnummer: "

while True:
    for elev_gr in elev_gr_til_k_nr:     # For alle nøgler i distionary
        k_nr = elev_gr_til_k_nr[elev_gr] # slås kanal nummeret op og bruges
        radio.config(channel=k_nr)       # til at sætte radiokanal op
        radio.send(besked+str(k_nr))     # til at sende besked på
        print(besked+str(k_nr))          # skriver hvad der sendes i shell
        sleep(500)                       # holder en pause.
```

Vis og Kør programmet LKSMb-sender-4.py fra lærer PC/MAC i Thonny på storskærm fremfor som slide.

Forklar koden – især nu Dictionary delen – som nyt begreb.

Henvis til dokumentation hvo de kan læse mere (w3schools) – klikbar link på slide, men også i bunden af slide stak.
(Vis på skærm ved at skifte til browser er den anbefalede metode)

Effektiv måde at lave opslag på ud fra en nøgle (her elevgruppe) til en værdi (her et kanalnummer)

Som gammeldags telefonbog, adresseliste etc.

Hvis grupperne har fået modificeret deres modtager SW korrekt vil de se at de kun modtager den besked der er tiltænkt dem.

Spor 1 Modul b: Option: Kryptering - Sender.

Projekt støttet af:
midt regionmidtjylland

```python
# LKS1Mb-Krypt-Sender.py
# Ved hjælp af et dictionary kan man lave en simpel kryptering ved
# at udskifte tegn med andre tegn lagt ud i dictionary.
# Læser input fra shell og sender krypteret besked på valgt kanal - En gang!
# OBS - importerer IKKE microbit modul, da det giver memory fejl på V1 af MicroBitten. OK på V2!
import radio

radio.on()
radio.config(channel=10,length=83) # Brug kanalnummer der er udleveret/aftalt
                                   # Length sat op til 83 for at kunne håndtere en tekst på op til 80 tegn plus afslutning
cipher = {
# Tegn (nøgle - key) mappet til encodingsværdi (value) - Man kan lave koden om ved at udskifte value
# SKAL være:
#  - Ens i både sender og modtager for at kunne dekryptere.
#  - Entydige (kun indgå een gang) både som key og som value (da den inverse bruger value som key)
#    til at lave det modsatte opslag.
    'a':'c','b':'d','c':'e','d':'f','e':'g','f':'h','g':'i','h':'j','i':'k','j':'l','k':'m','l':'n','m':'o',
    'n':'p','o':'q','p':'r','q':'s','r':'t','s':'u','t':'v','u':'w','v':'x','w':'y','x':'z','y':'a','z':'b',
    '0':'2','1':'3','2':'4','3':'5','4':'6','5':'7','6':'8','7':'9','8':'0','9':'1',
    ' ':' '
}

inp_tekst = input("Input tekst til encodning(a til z små og ikke øæå) Max 80 tegn: ")
print("Input tekst: ",inp_tekst)

# Encodning (Krytptering) af input tekst/streng ved at udskifte alle tegn een for een vha opslag i cipher
enc_tekst = ""
for i in range(0,len(inp_tekst)):
    if inp_tekst[i] in cipher.keys(): enc_tekst=enc_tekst+cipher[inp_tekst[i]]    # udelader tegn der ikke er med i dictionary
radio.send(enc_tekst)
print("Sendt tekst: ",enc_tekst)
```

Option/Kan udelades. Simpel kryptering - sender

Efter at have introduceret radio og kanaler kan man lade eleverne arbejde i grupper og bruge sender kode/selv skrive sender kode på én MB og sende beskeder til resten af gruppen på den tildelte kanal.
Som lærer kan man så 'lytte med' på de enkelte gruppers kanaler og spørge ind til om de sender noget og så kunne fortælle dem at man kan se hvad de sender.
Det kan bruges til en snak om kommunikations sikkerhed og kryptering.

Det kan også bruges til niveau deling til den/de grupper som er hurtige til at fange tingene versus dem som ikke er.

Man kan så lave en kobling tilbage til dictionaries og vise at man kan bruge det til andre ting, som for eksempel en simpel tegnudskiftnings kryptering.

Man kan udlevere de 2 programmer:
LKSMb-Sender-Krypt.py
LKSMb-Modtager-Krypt.py

Og stille en opgave hvor eleverne:
- Skal ændre kryptering

- Sammenbygge de 2 så man:
    - Ved tryk på knap A på Micro:bitten går ind og venter på indtastet besked som efterfølgende sendes
    - Ellers står og venter på beskeder, som decodes og udskrives når de kommer.

BEMÆRK

At danske tegn ‘æøå’ ikke fungerer ifm krypteringen på Micro:bit. Kun de internationale tegn.

At der kun kan sendes strenge på max 29 tegn med mindre man gør buffer længere med ”lenght” parameteren i kald til radio.config()

At man vil få memory problemer med import af alle funktioner fra microbit biblioteket/modulet på Micro:Bit version 1. Ingen problemer på Version 2!

- Derfor er linjen ”from microbit import *” udeladt i disse eksempler for at sikre de kan køre på både V1 og V2 af Micro:Bitten.
- Kan håndteres ved kun at importere de nødvendige funktioner på V1 – se løsningseksemplet LKSMb-Krypt-komb.py, der kan køre på både V1 og V2.

Spor 1 Modul b: Option: Kryptering - Modtager.

Projekt støttet af:

```python
# LKS1Mb-Krypt-Modtager.py
# Står og lytter efter beskeder på bestemt kanal og decoding modtaget besked og udskriver den i shell
# Decryptering ved hjælp af det inverse (omvendte) disctionary fra value til key i forhold til oprindeligt.
# OBS - importerer IKKE microbit modul, da det giver memory fejl på V1 af MicroBitten. OK på V2!

import radio

radio.on()
radio.config(channel=10,length=83) # Brug kanalnummer der er udleveret/aftalt
                                   # Length sat op til 83 for at kunne håndtere en tekst på op til 80 tegn plus afslutning
cipher = {
# Tegn (nøgle - key) mappet til encodingsværdi (value) - Man kan lave koden om ved at udskifte value
# SKAL være:
#  - Ens i både sender og modtager for at kunne dekryptere.
#  - Entydige (kun indgå een gang) både som key og som value (da den inverse bruger value som key)
#    til at lave det modsatte opslag.
    'a':'c','b':'d','c':'e','d':'f','e':'g','f':'h','g':'i','h':'j','i':'k','j':'l','k':'m','l':'n','m':'o',
    'n':'p','o':'q','p':'r','q':'s','r':'t','s':'u','t':'v','u':'w','v':'x','w':'y','x':'z','y':'a','z':'b',
    '0':'2','1':'3','2':'4','3':'5','4':'6','5':'7','6':'8','7':'9','8':'0','9':'1',
    ' ':' '
}

# Det omvendte distionary til dekryptering (value bliver til key og key til value for alle elemeter (items)
invers_cipher = dict((v, k) for k, v in cipher.items())

while True:
    enc_tekst = radio.receive()
    if enc_tekst != None:       # Der modtages noget forskelligt fra None. altså en besked på kanalen!
        # Decodning ved udskiftning af tegn i det omvendte distionary
        dec_tekst = ""
        for i in range(0,len(enc_tekst)):   # For alle tegn i teksten
            dec_tekst=dec_tekst+invers_cipher[enc_tekst[i]]
        print("Modtaget: ",enc_tekst)
        print("Decodet: ", dec_tekst)
```

Det samme som i sender!!

Option/Kan udelades. Simpel kryptering Modtager

Ud over kommentarerne i forrige slide (om sender siden) kan det bemærkes at det er en simpel monoalfabetisk substitution der foretages og med den rækkefølge af værdier som er angivet i eksemplet svarer det til en cæsaralgoritme hvor nøglen bare er et antal pladser i alfabetet som tegnene flyttes.

Det vil man programteknisk kunne lave enklere ved simple matematiske funktioner på tegnværdi (plads i alfabetet) – men her er historien ligeså meget det med disctionaries rent programteknisk. Og med hensyn til kryptering kan man blande værdierne i en anden rækkefølge så det er mere end en simpel cæsar substitution.

MEN igen – Ambitionen her er ikke at gå dybt og detaljeret ind i kryptering.

Spor 1 Modul b: Option: Et muligt svar på opgave.

Projekt støttet af:
midt regionmidtjylland

```
# LKS1Mb-Krypt-Kombo.py
# Ved hjælp af et dictionary kan man lave en simpel kryptering ved
# at udskifte tegn med andre tegn lagt ud i dictionary.
# Læser input fra shell og sender krypteret besked på valgt kanal - En gang
# OBS - importerer KUN button_a metoden/funktione fra microbit modul, da en fuld import af alle metoder/funktioner
# giver memory fejl på V1 af MicroBitten. Der vil ikke være problemer på V2 af microbitten da den har mere memory.
from microbit import button_a   # Henter kun det der bruges!
import radio

radio.on()
radio.config(channel=10,length=83) # Brug kanalnummer der er udleveret/aftalt
                                   # Length sat op til 83 for at kunne håndtere en tekst på op til 80 tegn plus afslutning
cipher = {
# Tegn (nøgle - key) mappet til encodingsværdi (value) - Man kan lave koden om ved at udskifte value
# SKAL være:
#  - Ens i både sender og modtager for at kunne dekryptere.
#  - Entydige (kun indgå een gang) både som key og som value (da den inverse bruger value som key)
#  - til at lave det modsatte opslag
   'a':'c','b':'d','c':'e','d':'f','e':'g','f':'h','g':'i','h':'j','i':'k','j':'l','k':'m','l':'n','m':'o',
   'n':'p','o':'q','p':'r','q':'s','r':'t','s':'u','t':'v','u':'w','v':'x','w':'y','x':'z','y':'a','z':'b',
   '0':'2','1':'3','2':'4','3':'5','4':'6','5':'7','6':'8','7':'9','8':'0','9':'1',
   ' ':' '
}
# Det omvendte dictionary til dekryptering (value bliver til key og key til value for alle elemeter (items)
invers_cipher = dict((v, k) for k, v in cipher.items())

def hent_og_send_input():
    inp_tekst = input("Input tekst til encodning(a til z små og ikke æøå) Max 80 tegn: ")
    print(inp_tekst)
    # Encodning (kryptering) af input tekst/streng ved at udskifte alle tegn een for een vha opslag i cipher
    enc_tekst = ""
    for i in range(0,len(inp_tekst)):
        if inp_tekst[i] in cipher.keys(): enc_tekst=enc_tekst+cipher[inp_tekst[i]]    # udelader tegn der ikke er med i dictionary
    radio.send(enc_tekst)
    print("Sendt tekst: ",enc_tekst)

while True:
    enc_tekst = radio.receive()
    if enc_tekst != None:          # Der modtages noget forskelligt fra None. altså en besked på kanalen!
        # Decodning ved udskiftning af tegn i det omvendte dictionary
        dec_tekst = ""
        for i in range(0,len(enc_tekst)):   # For alle tegn i teksten
            dec_tekst=dec_tekst+invers_cipher[enc_tekst[i]]
        print("Modtaget: ",enc_tekst)
        print("Decodet: ", dec_tekst)

    if button_a.is_pressed(): hent_og_send_input()
```

Option/Kan udelades.

En mulig løsning på opgave om at kombinere sender og modtager.
Findes som kode:
LKS1Mb-Krypt-Kombo.py

Spor 1 Modul b: Indbygget sensor til pitch over radio.

Projekt støttet af:

```
# LKS1Mb-AccToPitch-Sender.py
# Accelerometer data fra x-akse sendt på radio som tekst streng

from microbit import *
import radio

display.show("ATP-S")            # Udskriver på MB display for at identificere funktion
                                 # som 'Acc -> Pitch sender' når MB kører med batteri (svært at se ellers).
radio.on()
radio.config(channel=22)         # hvis kanal angives skal det være samme kanal i både sender og modtager
                                 # Hvis kanal ikke angives er default kanal nr 7 i begge
while True:
    acc_x,acc_y,acc_z = accelerometer.get_values()
    f_num = acc_x+1500           # adderer tilstrækkelig stor værdi til at de fleste er indenfor OK pitch område.
    print(f_num,"  ",type(f_num)) # test udskrift for at vise hvad vi har fået fra funktionskaldet til accelerometeret
    radio.send(str(f_num))       # radio.send() funktionen sender tekst strenge så tal konverteres til tekst-streng.
    sleep(50)                    # Bør bruge samme værdi i både sender og modtager. Samme takt.
```

Opgave:

- Indsæt kommentarer i sidste del af modtager. Hvad bruges nof_nones til?
- Få sender og modtager til at køre på 2 MB's i gruppe. Gerne med batteri.
- Brug tildelt kanal i gruppen!

Ekstra opgave:

- Lav sender der bruger magnetometer eller lyssensor til at generere og sende pitch i stedet for accelerometeret.

```
# LKS1Mb-AnaPitch-Modtager.py
# Forventer at modtage en tal værdi fra radio som kan bruges til at lave en lyd.
# Vil IKKE virke hvis der modtages andet end et tal. Så vil program gå ned med en fejl.
# Sikrer selv at værdi er mellem 50 og 3907 før den kalder music.pitch() med det tal der modtages
# De 3907 kommer fra V1 af Micro:Bitten, da den ikke kan håndtere pitch højere end 3907.
# Forklar hvad nof_nones bliver brugt til. Hvad er effekten?
from microbit import *
import music
import radio

display.show("APM")              # Udskriver på MB display for at identificere funktion
                                 # som 'Analog Pitch Modtager' når MB kører med batteri (svært at se ellers).
radio.on()
radio.config(channel=22)         # Brug kanalnummer der er udleveret/aftalt (0-83)

nof_nones = 0

while True:
    f_str = radio.receive()
    print(f_str,"  ",type(f_str))    # Test udskrift af hvad der modtages i shell incl typen af det
    if f_str:                        # Det samme som "f_str != None:" men IKKE som "f_str == True:"
        f_num = int(f_str)           # Men altså 'noget er modtaget' fra funktionen. Tekst-streng konverteres til tal.
        nof_nones = 0
        if (f_num>50) and (f_num<3907):
            music.pitch(f_num)
        else:
            music.stop()
    else: nof_nones = nof_nones+1
    if nof_nones > 20 : music.stop()
    sleep(50)
```

Udlever koderne i
LKS1Mb-AnaPitch-Modtager.py
LKS1Mb-AccToPitch-Sender.py

Stil opgaverne i slide – evt. som hjemmeopgaver.

Kombination af læring fra Modul a og radio delen af Mb, så det tjener som en form for repetition.

Dernæst at forstå det med at hvis radio forbindelsen afbrydes eller sender slukkes så skal/kan modtager finde ud af det, når der hele tiden sendes
Her er det godt, da der ellers ville være lyd kontinuerligt, som evt. panik pga. at lyden blev ved måske viste i modul a ☺

Det benyttes i corona badge case i modul c, hvorfor det er godt at forstå her for dem som vil kikke ind i koden til coronabadge (åben bog)

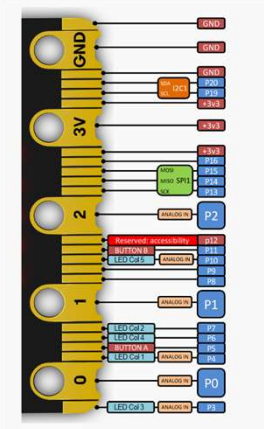
Forklar at print kommandoerne er med for at kunne følge med/for at debugge og her for at understøtte forklaringen om at der er forskel på tekst strenge og tal, hvorfor man skal konvertere.
Print sætningerne kan fjernes og giver som sådan ingen mening når MicroBitten kører

for sig selv med batteri, og ikke er koblet til PC/MAC med USB kabel.
Omvendt så fungerer det fortsat selvom den ikke er tilsluttet via kabel til Pc/MAC (og kan skrive til Shell).

Her er der et naturligt skifte, som evt. kan passe med lektions slutning.
Som det næste bliver der fokus på de eksterne PIN's

Spor 1 Modul b: Edge connector.

Projekt støttet af:
midt regionmidtjylland



Micro:Bit har en række PIN som kan bruges til at bygge egne elektroniske kredsløb. Der er både Digitale og Analoge inputs og outputs.
I første omgang bruges PIN 0,1 og 2, samt krokodillekabler og med fokus på input og SW siden
Man kan gå videre og dybere med flere inputs og outputs ved at bruge breakoutboards, breadboards og diverse komponenter og knytte an til elektronik/fysik

I denne gennemgang er der fokus på basal digital og analog input.

Bruger de 3 'store' PIN's, (0,1,2) samt 3V og GND, der kan bruges sammen med krokodille kabler.

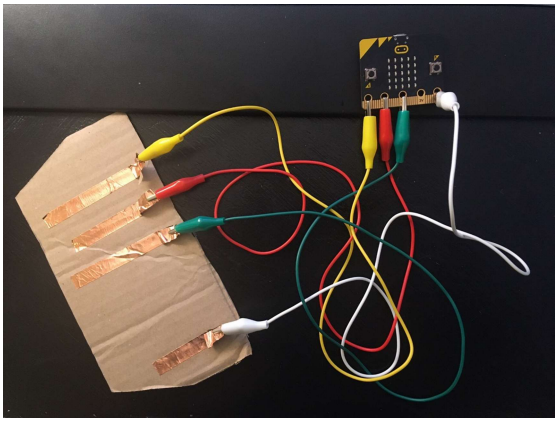
Der kan let bygges yderligere på efterfølgende afhængig af niveau og årgang.

Spor 1 Modul b: PIN's.

Projekt støttet af:
midt regionmidtjylland

Med kobber tape og 4 krokodilleledninger.

- Sæt 4 ledninger på pin 0,1,2 samt GND på én MB
- Sæt 4 strimler kobbertape på bord eller et stykke pap og forbind til MB som vist



Start simpelt med kobbertape og krokodillekabler. Tænkt som introduktion.

Der kan let bygges yderligere på efterfølgende afhængig af niveau og årgang.

Ved kortslutning mellem GND og PIN 0, 1, 2 vil funktionen/metoden is_touched() returnere TRUE.

Henvis til dokumentation for Micropython på Micro:Bit – tutorial delen for introduktion. Klikbar link på billedet i slide.

BEMÆRK at der i MB V2 er 2 måder at bruge is_touched() på – resistive (default) og capasitive (hvor man ikke behøver at kortslutte til GND).
Se API dokumentationen. Default er resistive.

I LYD-Kit sammenhæng er det valgt at bruge resistive, da det betyder man kan bruge både V1 og V2 Micro:Bits til alle kode eksempler.

BEMÆRK at der også er en funktion der hedder read_digital()
Se API delen af dokumentationen.

Den vil også kunne bruges men så skal man ikke koble mellem GND og PIN, men mellem

3V og PIN – og i virkeligheden til et eller andet kredsløb som genererer et signal på 3V som indikation for noget ex. en PIR sensor.

read_digital() kan bruges på flere PIN's end 0,1 og 2. Se supplerende materialer.

Spor 1 Modul b: PIN's. Opgave.

Projekt støttet af:

Brug udleveret kode:
- LKS1Mb-DigPitch-Modtager.py
- LKS1Mb-3Key-Sender.py
- Put sender koden i den MB som har tilsluttet de 4 krokodilleledninger og kobberstrimler.
- Brug tildelt kanal i gruppen!
- Udvid sender til at sende 8 forskellige Keys (incl. nul) ud fra de mulige kombinationer der er af PIN tilstande.

```
# LKS1Mb-3Key-Sender.py
# Sender 3 værdier for hhv pin 0, 1 og 2 på radio.
# PIN 0,1,2 is_touched -> sender "1", "2", "4".
# Når der ikke er nogen PINs der er aktive sendes et "0"
# PIN kan være aktive samtidigt så der er flere kombinationsmuligheder.
# Opgave: Udvid til at sende 7 forskellige keys "1" - "7" ud fra de mulige kombinationer.

from microbit import *
import radio

display.show("3KS")             # Udskriver en tekst på display for at identificere MB når der kommer strøm på
sleep(1000)                     # Venter 1 sekund inden vi går videre for at besked kan ses færdig.
                                # Bliver overskrevet, da key løbende udskrives i display.
radio.on()
radio.config(channel=33)        # Brug kanalnummer der er udleveret/aftalt (0-83)

while True:
    p0 = pin0.is_touched()
    p1 = pin1.is_touched()
    p2 = pin2.is_touched()
    if p0:
        key = "1"
    elif p1:
        key = "2"
    elif p2:
        key = "4"
    else:
        key = "0"

    radio.send(key)             # radio.send() funktionen sender key der er et tegn
    display.show(key)           # viser key på MBs display
    sleep(50)                   # Bør bruge samme værdi i både sender og modtager. Samme takt.
```

```
# LKS1Mb-DigPitch-Modtager.py
# Forventer at modtage en key for et input der er aktivt og nul når ingen er aktive.
# Reagerer på key's med værdi 1,2,3,4,5,6,7 og spiller en tone/frekvens for hver af dem.
# Modtages 0 stoppes tone.
#
from microbit import *
import music
import radio

display.show("DPM")             # Udskriver på MB display for at identificere funktion
                                # som "Digital Pitch Modtager" når MB kører med batteri (svært at se ellers).
radio.on()
radio.config(channel=33)        # Brug kanalnummer der er udleveret/aftalt (0-83)

nof_nones = 0

# Dictionary med key (1-7) som nøgle (til opslag) - Key som tegn derfor i ""
# og tone som frekvens som det man får retur      - Frekvens som tal derfor ikke i ""
key_til_pitch = { # key:  Frekvens:
                  "1":  440,        # A4 - Kammertonen
                  "2":  493,        # B4
                  "3":  523,        # C5
                  "4":  587,        # D5
                  "5":  659,        # E5
                  "6":  698,        # F5
                  "7":  783         # G5
    }

while True:
    key_str = radio.receive()
    if key_str:                     # noget modtaget
        nof_nones = 0
        print(key_str)              # som testudskrift i shell når koblet til PC/MAC
        if key_str == "0" : music.stop()
        else: music.pitch(key_til_pitch[key_str])
    else: nof_nones = nof_nones+1
    if nof_nones > 20 :             # sender er slukket eller udenfor rækkevidde. Stop evt. lyd!
        music.stop()
        nof_nones = 0
    sleep(50)
```

Udlever koden til sender og modtager.
LKS1Mb-DigPitch-Modtager.py
LKS1Mb-3Key-Sender.py

Gennemgå/forklar overordnet.
Lad dem få det op at køre i grupperne med 2 Micro:Bits så de kan lave 3 toner i modtager med 3 inputs fra sender.
Hvis modtager koden kører på en V1 MicroBit, så husk at der skal tilsluttes en højtaler til PIN0 og GND.

Giv eleverne opgave med at udvide sender sådan at den kan sende 7 forskellige key's til modtager ud fra de mulige kombinationer af de 3 inputs på PIN0, 1 og 2.
Lad dem selv tænke over det inden der bliver givet hints (næste slide) – Det kan være der kommer andre løsninger!

Programteknisk mål her er at lære at bruge boelske kombinationer og til basal kombinatorik (tæl til 7 med 3 'digitale' (On/Off) (High/Low) (True/False) inputs)!

MEN der er andre løsninger så vent med at skubbe i retningen af logiske operatorer – brug det ifm hint og opsamling (hvor også alternativ kan vises hvis ikke det kommer fra eleverne.)

Spor 1 Modul b: PIN's. Opgave. Uddybende hints

Projekt støttet af:

Pin 0,1,2.is_touched() funktionerne returnerer enten True eller False afhængig af om der er en kortslutning eller ej. Dvs. der er 8 forskellige kombinationer af de 3

- Opgaven er nu at lave en ny version af senderen, som kan sende 8 forskellige key's til modtageren.
- Hint – Logiske operatorer and og not Python Operators (w3schools.com)
- Men der er også andre veje.

```
# LKS1Mb-3key-Sender.py
# Sender 3 værdier for hhv pin 0, 1 og 2 på radio.
# PIN 0,1,2 is_touched -> sender "1", "2", "4".
# Når der ikke er nogen PINs der er aktive sendes et "0"
# PIN kan være aktive samtidigt så der er flere kombinationsmuligheder.
# Opgave: Udvid til at sende 7 forskellige keys "1" - "7" ud fra de mulige kombinationer.

from microbit import *
import radio

display.show("3KS")             # Udskriver en tekst på display for at identificere MB når der kommer strøm på
sleep(1000)                     # Venter 1 sekund inden vi går videre for at besked kan ses færdig.
                                # Bliver overskrevet, da key løbende udskrives i display.
radio.on()
radio.config(channel=33)        # Brug kanalnummer der er udleveret/aftalt (0-83)

while True:
    p0 = pin0.is_touched()
    p1 = pin1.is_touched()
    p2 = pin2.is_touched()
    if p0:
        key = "1"
    elif p1:
        key = "2"
    elif p2:
        key = "4"
    else:
        key = "0"

    radio.send(key)             # radio.send() funktionen sender key der er et tegn
    display.show(key)           # viser key på MBs display
    sleep(50)                   # Bør bruge samme værdi i både sender og modtager. Samme takt.
```

p0	p1	p2	Key
False	False	False	”0”
True	False	False	”1”
False	True	False	”2”
True	True	False	”3”
False	False	True	”4”
True	False	True	”5”
False	True	True	”6”
True	True	True	”7”

Når I har løst opgaven og vist det frem så kan I enten:

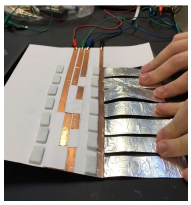
- Lav 2 sendere og udvid til 15 forskellige toner i modtager. – Eller:
- Byg eller skitser en fysisk/mekanisk 'knap' eller tastatur der kan bruge det I lige har været igennem med kobbertape. Hvad skal der ellers bruges af materiale?

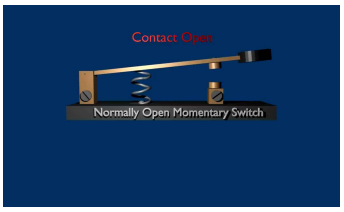
Uddybende hints til dem der ikke kommer i gang.

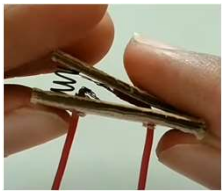
Ekstra opgaver til dem som hurtigt klarer det.

Spor 1 Modul b: PIN's. Opgave. Også i mekanik.

Projekt støttet af: midt regionmidtjylland







DIY On Off Switch - YouTube

Make awesome Switches from waste cardboard - YouTube

DIY push button switch - YouTube

For at inspirere til også at tænke på 'mekanik' – Den fysiske kontakt.

For at der i grupperne vil kunne arbejdes med andre ting end software.

For at give et første billede på at man typisk skal have flere domæner i spil i produktudvikling.

Spor 1 Modul b: Et muligt svar på opgave.

Projekt støttet af:

midt regionmidtjylland

```
# LKS1Mb-7Key-Sender.py
# Et muligt svar på opgaven stillet med udgangspunkt i LKS1Mb-3Key-Sender.py
# Sender key = "1" til "7" for mulighe kombinationer af PIN 0,1 og 2.
# Sender key = "0" løbende hvis der ikke er nogen af de 3 PINs der er aktive.
#
from microbit import *
import radio

display.show("7KS")             # Udskriver en tekst på display for at identificere MB når der kommer strøm på
sleep(1000)                     # Venter 1 sekund inden vi går videre for at besked kan ses færdig.
                                # Bliver overskrevet, da key løbende udskrives i display.
radio.on()
radio.config(channel=33)        # Brug kanalnummer der er udleveret/aftalt (0-83)

while True:
    p0 = pin0.is_touched()              # Læser et øjebliksbillede af pin 0,1 og 2 ind i variabler
    p1 = pin1.is_touched()              #       p0      p1      p2          Key
    p2 = pin2.is_touched()              #-------------------------------------------------------
    if p0 and p1 and p2:                #       True    True    True        "7"
        key = "7"                       #
    elif not p0 and p1 and p2:          #       False   True    True        "6"
        key = "6"                       #
    elif p0 and not p1 and p2:          #       True    False   True        "5"
        key = "5"                       #
    elif not p0 and not p1 and p2:      #       False   False   True        "4"
        key = "4"                       #
    elif p0 and p1 and not p2:          #       True    True    False       "3"
        key = "3"                       #
    elif not p0 and p1 and not p2:      #       False   True    False       "2"
        key = "2"                       #
    elif p0 and not p1 and not p2:      #       True    False   False       "1"
        key = "1"                       #
    else:                               #       False   False   Fasle       "0"  sidste mulighed derfor bare else
        key = "0"
    radio.send(key)             # radio.send() funktionen sender key som er et tegn
    display.show(key)           # viser key på MBs display
    sleep(50)                   # Bør bruge samme værdi i både sender og modtager. Samme takt.
```

Et svar. Der er andre!

Det er svært at undgå nogle af de primære (1,2,4) på vejen til eller fra de andre.

Kan forbedres ved at man checker key 2 gange med ex 20 ms mellemrum og ser om værdien er den samme.

Et muligt svar man kan bruge til at opsummere med efter opgaven.

Skal naturligvis IKKE udleveres inden opgaven, men kan deles efterfølgende til støtte for dem som evt. ikke fik det til at fungere.

Brug også til at tage en snak om fordele/ulemper ved denne måde at få 7 forskellige key's.

Fordele:
- 7 værdier med kun 3 digitale inputs
- Billigere i elektronik (kun 3 inputs)

Ulemper
- Usikker aftastning – Kan ikke altid ramme 7 uden også at få andre på vejen
- Mere besværligt at lave knapper i mekanik aht koblingen til de 3 inputs.
- Dyrere i mekanik

Igen er det også vigtigt at sige at der er flere løsninger. Der er som flere veje til løsninger!

Derfor er det også godt at vise alternativ løsning på næste slide.

Spor 1 Modul b: Et andet muligt svar på opgave.

Projekt støttet af:
midt regionmidtjylland

```
# Alternativ-7Key-Sender.py
#
from microbit import *
import radio

display.show("7KS")              # Udskriver en tekst på display for at identificere MB når der kommer strøm på
sleep(1000)                      # Venter 1 sekund inden vi går videre for at besked kan ses færdig.
                                 # Bliver overskrevet, da key løbende udskrives i display.
radio.on()
radio.config(channel=33)         # Brug kanalnummer der er udleveret/aftalt (0-83)

while True:
    key_val = 0                  # Ingen pin -> værdien 0
    p0 = pin0.is_touched()
    p1 = pin1.is_touched()
    p2 = pin2.is_touched()
    if p0:                       # Hvis Pin 1 -> +1 dvs 1
        key_val = key_val + 1
    if p1:                       # Hvis PIN 2 -> +2 dvs enten 2 eller 3
        key_val = key_val + 2
    if p2:                       # Hvis PIN 3 -> +4 dvs enten 4, 5, 6 eller 7
        key_val = key_val + 4

    key_str = str(key_val)       # laver key_val om til tegn
    radio.send(key_str)          # radio.send() funktionen sender key som er et tegn
    display.show(key_str)        # viser key på MBs display
    sleep(50)                    # Bør bruge samme værdi i både sender og modtager. Samme takt.
```

Et alternativt svar. Der er som regel ikke kun én løsning!

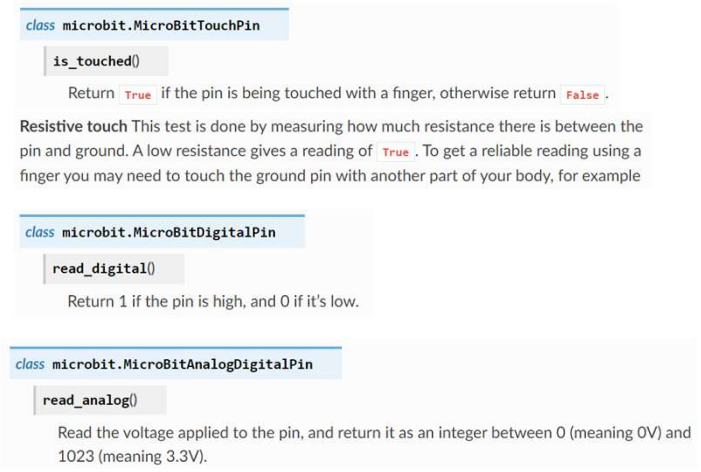
Et alternativt svar svar man kan bruge til at opsummere med efter opgaven.

For at understrege pointen med at der ikke kun er én løsning, men at der er forskellige veje med hver deres fordele/ulemper.

Skal naturligvis IKKE udleveres inden opgaven, men kan deles efterfølgende til støtte for dem som evt. ikke fik det til at fungere.

Spor 1 Modul b: Option: Byg kredsløb. (input)

Projekt støttet af: midt regionmidtjylland



class microbit.MicroBitTouchPin

is_touched()

Return True if the pin is being touched with a finger, otherwise return False.

Resistive touch This test is done by measuring how much resistance there is between the pin and ground. A low resistance gives a reading of True. To get a reliable reading using a finger you may need to touch the ground pin with another part of your body, for example

class microbit.MicroBitDigitalPin

read_digital()

Return 1 if the pin is high, and 0 if it's low.

class microbit.MicroBitAnalogDigitalPin

read_analog()

Read the voltage applied to the pin, and return it as an integer between 0 (meaning 0V) and 1023 (meaning 3.3V).

Brug PIN 0, 1 og 2 til aflæsning af om der er kontakt til jord – Det der er brugt i eksemplerne ovenfor.

Brug PIN 0,1,2,8,13,14,15,16 til aflæsning af om der er høj spænding (2.1V – 3.3V) eller lav (0V-0.9V) på PIN.

Brug PIN 0, 1 og 2 til aflæsning af analoge værdier fra et kredsløb. Der sidder 10-bits A/D konvertere som kan omsætte spænding mellem 0 og 3.3V til et heltal mellem 0 og 1023.

Edge Connector and Pinout (microbit.org)

is_touched()
Returnerer True/False baseret på lille (kortslutning) eller stor modstand fra PIN til GND og er brugt i eksemplerne tidligere i denne slide stak.

read_digital()
Returnerer 1 hvis der er 3.3V på PIN og 0 hvis der er 0V på PIN. Dvs. PIN kan kobles til et kredsløb der:

- Leverer max 3,3V på PIN og har fælles GND med Micro:Bitten. Det kan eksempelvis være en PIR sensor.
- Kan den drives af 3.3V/90mAmp kan den forsynes fra MB's 3V PIN.
- Skal kredsløb have en større spænding/strøm end 3.3V skal den forsynes af ekstern forsyning og må ikke kobles sammen med 3V PIN på MB.
- Eksemplerne med 'knapper' på PIN0,1 og 2 kan laves med read_digital() og kobling fra PIN til 3V i stedet for til GND, og så er der mulighed for 8 forskellige inputs på samme MB. Se supplerende materiale.

read_analog()
Returnerer en værdi mellem 0 og 1023 (begge incl.) ud fra en spænding på PIN mellem 0V og 3.3V. Dvs. PIN kan kobles til kredsløb der:

- Har fælles GND med MB og leverer et signal på PIN mellem 0 og 3.3V.
- Kan det drives af 3,3V/90mAmp kan den forsynes fra MB's 3V PIN.

- Skal det have større spænding/strøm skal det forsynes eksternt.
- Det kan eksempelvis være et Potentiometer,
- eller en temperaturføler (pt100 set i fysik undervisningsmaterialer el og sensor).
- eller en stain-gauge (også brugt i fysik undervisningsmaterialer)
- eller et spændingsdeler kredsløb med 5 knapper der så vil give 5 forskellige niveauer af spænding på PIN – så kan man også få 'knapper' den vej.
- eller en fugtighedsmåler – samme princip som temperaturføleren – En variabel modstand
- Man kan jo bare prøve at lade 3V løbe gennem ens hud til PIN og se om man kan se forskel på værdi afhængig af hvor hårdt man klemmer og hvor fugtig ens hud er.

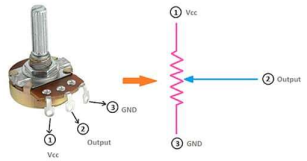
Spor 1 Modul b: Option: Byg kredsløb. Opgave 1

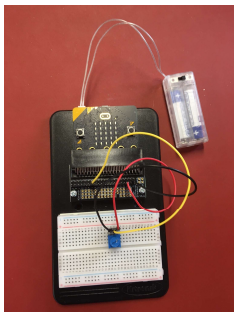
Projekt støttet af: midt regionmidtjylland

```
# LKS1Mb-AccToPitch-Sender.py
# Accelerometer data fra x-akse sendt på radio som tekst streng

from microbit import *
import radio

display.show("ATP-S")           # Udskriver på MB display for at identificere funktion
                                # som 'Acc -> Pitch sender' når MB kører med batteri (svært at se ellers).
radio.on()
radio.config(channel=22)        # hvis kanal angives skal det være samme kanal i både sender og modtager
                                # Hvis kanal ikke angives er default kanal nr 7 i begge
while True:
    acc_x,acc_y,acc_z = accelerometer.get_values()
    f_num = acc_x+1500          # adderer tilstrækkelig stor værdi til at de fleste er indenfor OK pitch område.
    print(f_num," ",type(f_num)) # test udskrift for at vise hvad vi har fået fra funktionskaldet til accelerometeret
    radio.send(str(f_num))      # radio.send() funktionen sender tekst strenge så tal konverteres til tekst-streng.
    sleep(50)                   # Bør bruge samme værdi i både sender og modtager. Samme takt.
```

Opgave:

- Load LKS1Mb-AnaPitch-Modtager.py på én MB
- Lav et kredsløb med et potentiometer på PIN1 af en anden MB.
- Modificer koden i LKS1Mb-AccToPitch-sender.py så den sender værdier fra potentiometer.





```
# LKS1Mb-AnaPitch-Modtager.py
# Forventer at modtage en tal værdi fra radio som kan bruges til at lave en lyd.
# Vil IKKE virke hvis der modtages andet end et tal. Så vil program gå ned med en fejl.
# Sikrer selv at værdi er mellem 50 og 3907 før den kalder music.pitch() med det tal der modtages
# De 3907 kommer fra V1 af Micro:Bitten, da den ikke kan håndtere pitch højere end 3907.
# Forklar hvad nof_nones bliver brugt til. Hvad er effekten?
from microbit import *
import music
import radio

display.show("APM")             # Udskriver på MB display for at identificere funktion
                                # som 'Analog Pitch Modtager' når MB kører med batteri (svært at se ellers).
radio.on()
radio.config(channel=22)        # Brug kanalnummer der er udleveret/aftalt (0-83)

nof_nones = 0

while True:
    f_str = radio.receive()
    print(f_str," ",type(f_str))  # Test udskrift af hvad der modtages i shell incl typen af det
    if f_str:                     # Det samme som "f_str != None:" men IKKE som "f_str == True:"
        f_num = int(f_str)        # Men altså 'noget er modtaget' fra funktionen. Tekst-streng konverteres til tal.
        nof_nones = 0
        if (f_num>50) and (f_num<3907):
            music.pitch(f_num)
        else:
            music.stop()
    else: nof_nones = nof_nones+1
    if nof_nones > 20 : music.stop()
    sleep(50)
```

Genbrug koderne udleveret tidligere
(Brug til test om eleverne får gemt programmer og sikrer backups ☺)

LKS1Mb-AnaPitch-Modtager.py
LKS1Mb-AccToPitch-Sender.py

Udlever breakoutboard, bredboard samt potentiometer. Fortæl om det.

Stil opgaven fra slide.

HINT: Brug kaldet pin1.read_analog() og læg en tilpas stor værdi til (for at komme over de 50) input er mellem 0 og 1023!

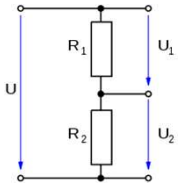
HINT: Det midterste ben på potentiometer er typisk signal benet. De andre 2 skal til hhv. GND og forsyning 3V.

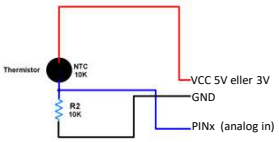
At der er så meget genbrug understøtter 'mønstergenkendelse' og repetition.
Det kunne naturligvis også stilles mere åbent.

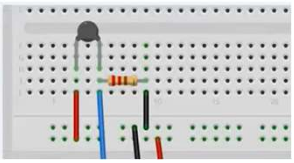
Her er det tænkt som introduktion – inden man stiller en mere åben opgave

Spor 1 Modul b: Option: Byg kredsløb. Opgave 2

Projekt støttet af:







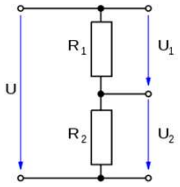
I fysik undervisning bruges voltmeter til at måle på en temperaturføler (variabel modstand) placeret i spændingsdeler og evt. i en broopstilling.

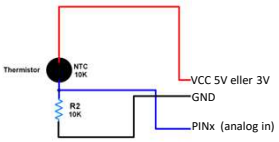
Med reference til fysik lærebogen Orbit B STX, s.253 til 259

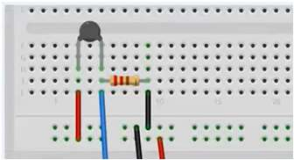
Så her en god mulighed for at knytte an til fysik undervisningen.

Spor 1 Modul b: Option: Byg kredsløb. Opgave 2

Projekt støttet af:

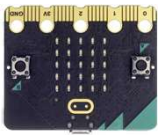






I stedet for voltmeteret kan man bruge en Micro:Bit og aflæse signalet via A/D converteren på en PIN

Og bruge det til at sende værdier og lave temperatur om til en tone …



Brug en Microkontroller med en A/D konverter til at måle værdien og få det ind i det digitale domæne så man kan bruge det til digital proces kontrol og monitorering.

Microbitten har en 10-bits A/D (analog til digital) konverter indbygget så 0-3V værdier konverteres til værdier mellem 0 og 1023.

I professionelle temperaturmålere er der i dag indbygget Microkontroller og A/D konvertere med højere opløsning (flere bits) til mere nøjagtige målinger/udvidede temperaturområder etc.

Spor 1 Modul b: Option: Opgave 2 - Perspektivering

Projekt støttet af: midt regionmidtjylland

Horizontal Processing Tank | VARO HPT - Cook, mix and cool in one batch - YouTube

Første selv-kalibrerende temperaturføler - iTHERM TrustSens | Endress+Hauser

Og sådan er det i øvrigt med de fleste af de ting I bygger på HTX – det er 'kun' tidlige mock-ups/modeller/prototyper. Pas på med prototype begrebet, da prototyper kan have mange formål undervejs i en udviklingsproces. Kan bruges i flere faser i IPU modellen. I en produktions virksomhed kan det være et næsten færdigt produkt til test af processer/kvalitet etc., hvor det ifm SW projekter ofte bruges om en meget tidlig model hvor funktionalitet testes af for at se om produktet er værd at fortsætte med.

Ude i den virkelige verden bruges temperaturføler til mange ting – herunder i procesanlæg i fødevare industrien.

Her kan man tydeligt få en fornemmelse for at der er mere i det end en forståelse for fysikken (el-læren) eller om mikrokontrollere og software.
Der er krav ud fra miljø (tanke og temperatur og bakterier) som gør at følere skal pakkes ind i noget passende mekanik, at der er mange af dem som skal kobles sammen i et kontrol og styringsanlæg etc.

Så et godt eksempel på kobling fra fysik til teknologi i HTX. (og stadigvæk noget grundlæggende til fælles med det der er kikket på omkring sensorer i almindelighed)

Spor 1 Modul b: Option: Byg med andre aktuatorer.

Projekt støttet af:
midt regionmidtjylland

I denne slide stak er fokus på sensor siden og med højtaler/lyd som aktuator.
Der kan naturligvis også bygges med andre aktuatorer for at lave systemer der kan andet.

- Motorer
- Lys

Nu er eleverne muligvis trætte af gentagelser omkring at lave sensor input om til lyd, så her kan man vise at man kan bruge andre ting som aktuator (output) end en højtaler. Brug en af de foregående sendere og lav en ny modtager med en motor eller lys på i stedet.

Spor 1 Modul b: Option: Andre aktuatorer. (output)

Projekt støttet af: midt regionmidtjylland

write_digital(*value*)

Set the pin to high if `value` is 1, or to low, if it is 0.

write_analog(*value*)

Output a PWM signal on the pin, with the duty cycle proportional to the provided `value`. The `value` may be either an integer or a floating point number between 0 (0% duty cycle) and 1023 (100% duty).

set_analog_period(*period*)

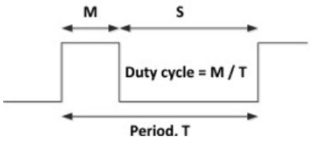
Set the period of the PWM signal being output to `period` in milliseconds. The minimum valid value is 1ms.

Brug PIN 0,1,2,8,13,14,15,16 til styring af kredsløb der reagerer på høj spænding (2.1V – 3.3V) eller lav (0V-0.9V) på PIN.

Brug PIN 0,1,2 til styring af kredsløb der reagerer på variabel spænding – Pulse Width Modulated (PWM).

Læs mere i dokumentationen.

PWM kan bruges til at lave lyd
PWM kan også bruges til at lave lysstyring/dæmpning af en diode.
PWM kan også bruges til at styre en Servo motor.



Kommandoer i MicroPython for Micro:Bit, som kan bruges til styring af kredsløb med output's.

Man kan bruge 3V forsyningen fra Micro:Bitten, men for at lave noget der er mere robust anbefales at man forsyner fra anden kilde, da Micro:Bitten ikke kan levere så meget strøm.

V2 kan levere lidt mere end V1. (V1 kan levere op til 90mAmp og V2 kan levere op til 190mAmp)

Microbitten ikke har en D/A (digital til analog) converter som kan lave præcis og glat regulering af udgangsspænding.

Den kan, som så mange andre mikrokontrollere, bare skifte mellem 0 og fuld spænding på PIN (3.3V).

Ved at tænde og slukke for den med en bestemt frekvens, og ved at have den tændt i en bestemt andel af tiden indenfor perioden kan den emulere spændingsniveuer mellem 0 og 3.3V. Det er hvad man kalder Pulse Width Modulation (PWM)

Se slide i supplerende materialer for yderligere baggrund. (lyd og lys eksempler med

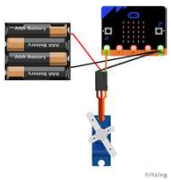
PWM)

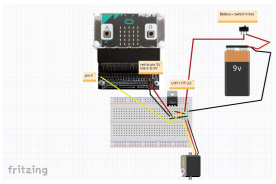
PS:

Med hensyn til styring af lys kan man ud over at benytte PWM også benytte biblioteket neopixels – se dokumentationen
Med hensyn til motor styring kan man få kontroller boards til mange forskellige slags motorer som styres via I2C – se dokumentationen.

Spor 1 Modul b: Option - Servomotor

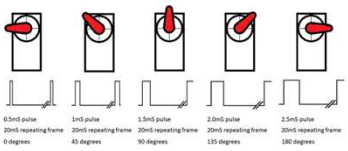
Projekt støttet af:





Positional Servo (med indbygget potentiometer)

Kontinuerlig/360 graders Servo (uden indbygget potentiometer)



Der findes små servo-motorer, der kan drives af de 3V som Micro:Bitten kan levere (90mAmp V1 og 190mAmp V2).
MEN generelt anbefales at have ekstern forsyning til Motorerne!

SÆT ALDRIG spændinger på mere end 3.3V på Micro:Bittens PIN's – Særligt ikke 3V!

Micro:Bitten kan forsynes ved at sætte forsyning på 3V PIN, så man ikke behøver selvstændig forsyning til Microbitten. Det sker typisk når man benytter div. færdige kontroller boards.

Man kan koble motor til vha. krokodille kabler, men her vil det være mere hensigtsmæssigt at bruge breadboards eller at bruge et købemodul med andre forbindelses muligheder for motor og forsyning. De har typisk 3 bens forbindelser, som passer med de ledninger/stik som sidder på servo-motorerne.

BEMÆRK:

Der findes overordnet 2 slags servo motorer:
- Positionsbestemt (med potentiometer indbygget)
- Kontinuerlig eller 360 graders (uden potentiometer indbygget)

De første reagerer på PWM med et udslag til en bestemt position. De har en begrænset vandring ex. 180 grader, men mange har mindre ex. 120 grader. (+/- 60 grader fra center position)

DERFOR kan der være issues mht. pulsebredde og udslag som man skal være opmærksom på i forbindelse med den bestemte motor man vælger og benytter.

De andre reagerer på de samme PWM signaler, men kører kontinuerligt.
Center positionen = grader vil betyde ingen bevægelse. 0 grader er fuld fart i den ene retning. 180 (max udslag) er fuld fart i den modsatte retning.

Pulsbredde mellem stilstand og fuld fat vil give bevægelser med mindre hastighed.

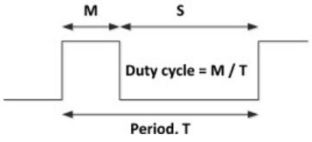
BEMÆRK – At der ifm færdige boards findes 2 typer:
- Dem som benytter PWM på PIN0,1 og 2
- Dem som benytter I2C kommunikation mellem Micro:Bit og controller board, hvor man typisk kan tilkoble flere forskellige slags motorer (også stepper og DC motorer)

Den som er vist på slide er af den simple type, som koden der vises i det efterfølgende kan bruges med (også med direkte opkobling med krokodille kabler og/eller breadboard.

Klikbare links i slide, men de findes også på sidste slide i stakken.

Spor 1 Modul b: Option - Servomotor.

Projekt støttet af: midt regionmidtjylland



- 1 sek. = 1000msek, så med en periode på 20msek kommer der en puls 50 gange i sekundet. Dvs. en frekvens på 50Hz (1000/20 = 50).
- Med en periode på 20msek udgør en puls på 500microsek en Duty cycle på 500/20000 = 0,025 = 2,5% af perioden.
- Når write_analog() har et område på 0-1023 svarer 100% til værdi på 1023 og 50% til 1023/2 = 512 afrundet.

Puls (microsek)	Duty cycle I % af perioden	I analog værdi (0-1023)
500	2.5	26
1000	5	51
1500	7.5	77
2000	10	102
2500	12.5	128

```
# Genererer pulser på PIN0 fra 1.5 microsek (analog værdi på 77)
# til max på 2500 microsek (analog værdi på 128)
# Til brug for at finde max udslag på servo motor.
# For at teste i modsat retning brug range(77,26,-1)
# Analog værdi på 26 svarer til puls på 500 microsek.
#
from microbit import *

pin0.set_analog_period(20)      # 50 Hz 50 gange på et sekund
for pulse in range(77,128,1):   # pulse som % af fuld duty med 1023.
    pin0.write_analog(pulse)
    print("'Duty' værdi",pulse) # Noter værdi når motor når max udslag
    sleep(500)
pin0.write_analog(77)           # for at sætte motor i center position
```

PWM -> Servo styring

BEMÆRK: Man skal checke op på servo motorens specifikation for at finde dens operations område.

Hvis ikke man kan finde specifikationer bør man teste det af. (Se hvordan i Slide)
Ex. hvor stor en vinkel den kan håndtere, og/eller hvilken puls bredde der giver max udslag.

Typisk styres servo motorer med pulser i 50Hz – D.v.s. at perioden typisk skal være 20msek – men der kan findes andre.

Pulse range og max vinkel udslag hænger sammen.

- Ex1. Puls range på 500 til 2500 mikrosekunder og vinkel på 0-180 (+/- 90), så vil minimum pulse (500) give udslag til 0 grader og max pulse (2400) til 180 grader.
- Ex2. Puls range på 1000 til 2000 mikrosekunder og vinkel på 0-180 (+/- 90), så vil minimum pulse (1000) give udslag på 0 grader og max pulse (2000) til 180 grader.
- Ex3. Puls range på 1000 til 2000 mikrosekunder og vinkel på 0-120 (+/-60), så vil minimum pulse (1000) give udslag på 0 grader og max pulse (2000) til 120 grader.
- Pulse på 1.5 mikrosekunder vil typisk altid give neutral position (den hvorfra man kan gå +/- antal grader)

BEMÆRK: Når perioden er sat og der er skrevet en analog værdi til en PIN sikrer den underliggende biblioteksfunktion for at der bliver ved med at blive sendt pulser ud på PIN.

Derfor behøver man i MicroPython koden kun at lave én udskrivning af værdi når der sker en ændring.

MEN der er altså fortløbende pulser på PIN!

Spor 1 Modul b: Option - Servomotor

Projekt støttet af:

midt regionmidtjylland

```
# LKS1Mb-DigServo-Modtager.py
# Modtager en key fra LKS1Mb-3Key-Sender.py og sætter servo motor på PIN0 i en
# position der matcher key. Hvis key = "0" eller sender slukkes returneres til
# center position (med 360 servo - stoppes motor)
#
from microbit import pin0,display,sleep
import radio

display.show("DSM")       # Udskriver på MB display for at identificere funktion
                          # som 'Digital Servo Modtager' når MB kører med batteri (svært at se ellers).
radio.on()
radio.config(channel=44) # Brug kanalnummer der er udleveret/aftalt (0-83)

nof_nones = 0

center_pulse_val = 77   # svarer til 1500 microsek ved periode på 20 msek.
min_pulse_val    = 30   # svarer til 500 microsek  - ret hvis anden spec.!
max_pulse_val    = 122  # svarer til 2500 microsek - ret hvis anden spec.!
min_vinkel       = 0
max_vinkel       = 180  # ret hvis anden spec.!

# Dictionary med key som nøgle (til opslag) - Key som tegn derfor i ""
# og servo vinkel som det man får retur     - Vinkel som tal derfor ikke i ""
key_til_vinkel = { # key:  Vinkel:
                   "0":   90,   # Bemærk center position ved 120 graders motor er 60 og IKKE 90!
                   "1":  180,
                   "2":   70,
                   "4":   40
    }
```

Opgave 1:

- Load LKS1Mb-Servo-Modtager.py på en MicroBit hvor der er koblet en servo motor til på PIN0.
- Load LKS1Mb-3key-Sender på anden MB.
- HUSK samme kanal nummer i radio_config.
- Se at I kan sætte motor til forskellige positioner med 'knap' tryk fra sender.

```
# Funktion der tager værdier i en range og laver om til en anden range
# Svarer til arduino c's map() funktion! MEN i python er map() noget andet - se dokumentation
# Derfor kalder vi den her for convert_range
#
def convert_range(x, in_min, in_max, out_min, out_max):
    return int((x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min)

vinkel_0    = int((max_vinkel-min_vinkel)/2)  # Svarer til center position og stopped med 360 graders servo.
ny_vinkel_0 = vinkel_0
pin0.set_analog_period(20)

while True:
    key_str = radio.receive()
    if key_str:                         # noget modtaget
        nof_nones = 0
        print(key_str)                  # som testudskrift i shell når koblet til PC/MAC
        if key_str in key_til_vinkel: ny_vinkel_0 = key_til_vinkel[key_str]
    else: nof_nones = nof_nones+1
    if nof_nones > 20 :                 # sender er slukket eller udenfor rækkevidde. Sæt motor i center!
        ny_vinkel_0 = int((max_vinkel-min_vinkel)/2)
        nof_nones = 0
    if (vinkel_0 != ny_vinkel_0):    # Behøver kun at sætte værdi ved ændring. PWM kører i baggrund.
        pulse_val = convert_range(ny_vinkel_0,min_vinkel,max_vinkel,min_pulse_val,max_pulse_val)
        print("pulse_val",pulse_val) # test udskrift
        pin0.write_analog(pulse_val)  # sender PWM til servo
        vinkel_0 = ny_vinkel_0
    sleep(50)
```

Opgave 2:

- Få flere inputs i sender/modtager til flere positioner.
- Lav en AnaServo udgave der kan reagere på sendere der benytter sig af mere analoge sensorer – Ex. potentiometer.
- Se på LKS1Mb-AnaPitch-Sender for inspiration

Udlever koden
LKS1Mb-Servo-Modtager

Få det til at fungere med servo motor. Sæt værdier så det passer med motoren.

Load LKS1Mb-3Key-Sender på en anden og se man kan kontrollere motor på afstand.

Se Slide 16 mht. inspiration til opgave 2.

Projekt støttet af:

midt regionmidtjylland

Supplerende Materiale

PIN I/O oversigt (anbefalet brug)

Pulse Width Modulation (PWM) -> lyd og lys.

Samling af links til materiale, som der henvises til undervejs.

Spor 1 Modul b: Edge connector.

Projekt støttet af:

P0	DIG I/O	Analog I/O	
P1	DIG I/O	Analog I/O	
P2	DIG I/O	Analog I/O	
P3			Anbefales ikke brugt til I/O da der er forskel på V1 og V2 selvom display slået fra
P4			Anbefales ikke brugt til I/O da der er forskel på V1 og V2 selvom display slået fra
P5			Anbefales ikke brugt til I/O da den bruges til Knap A
P6			Anbefales ikke brugt til I/O da der er forskel på V1 og V2 selvom display slået fra
P7			Anbefales ikke brugt til I/O da der er forskel på V1 og V2 selvom display slået fra
P8	DIG I/O		
P9			Anbefales ikke brugt til I/O da der er forskel på V1 og V2 selvom display slået fra
P10			Anbefales ikke brugt til I/O da der er forskel på V1 og V2 selvom display slået fra
P11			Anbefales ikke brugt til DIG I/O da den bruges til Knap B
P12	DIG I/O		Står til reserveret så lad være med at bruge den til Dig I/O - selvom den virker på både V1 og V2
P13	DIG I/O		
P14	DIG I/O		
P15	DIG I/O		
P16	DIG I/O		
P19			Reserveret til I2C kommunikation
P20			Reserveret til I2C kommunikation
PIN 3,4,6,7,9,10 opfører sig uhensigtmæssigt på MB V2 ifrohold til V1			
Digital høj på én PIN medfører høj på de øvrige! (ved read_digital() og display_off())			
Analog ind på én PIN har afsmittende effekt på andre pins. (ved read_analog() og display_off())			

Edge Connector and Pinout (microbit.org)

Input/Output Pins — BBC micro:bit MicroPython 1.0.1 documentation (microbit-micropython.readthedocs.io)

Efter test af både digital og analog input på hhv V1 og V2 af microbitten er det konstateret at man skal være varsom med at benytte PIN3,4,6,7,9,10 til andet end display (5*5 LED på forsiden) på Micro:bit V2, selvom der i dokumentationen står at man kan bruge disse PIN's til andet når bare man har slået displayet fra.

Er der input på én af de nævnte PIN's er der afsmittende effekt på de andre. Det gør sig kun gældende på V2 – Ikke på V1.

Derfor anbefales det kun at bruge PIN's: 0,1,2,8,13,14,15,16 til digital I/O og PIN's: 0,1,2 til analog I/O for ikke at få problemer.

MEN man kan for eksempel uden problemer lave 8 entydige digitale inputs (ex. knapper) med én Micro:Bit!

Så i LYD-Kit Spor 2 kan man for eksempel lave en åben bog opgave med at lave en MBType8.py til at håndtere 8 entydige inputs. (som alternativ til type 0)

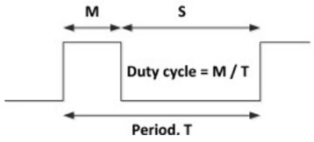
BEMÆRK:

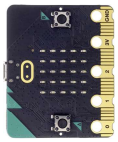
- MBtype0.py (Færdig kode til LYD-Kit Spor 2) bruger "is_touched()" hvor der testes på kortslutning (lav modstand) mellem PIN og GND – som kun virker på PIN0,1 og 2.
- Bruger man digital_read() så måles på om der er en spænding tæt på de 3V og det

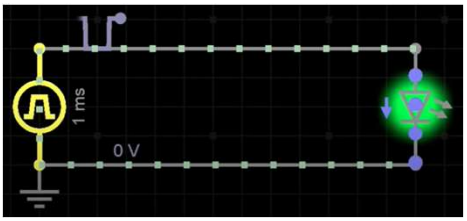
kan så bruges med diverse kredsløb der kan levere de 3V som input til PIN.

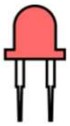
Spor 1 Modul b: PWM.

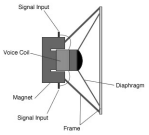
Projekt støttet af:

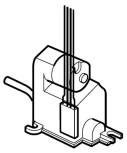












Pulse Width Modulation (PWM) - Electronics Basics 23 - YouTube

Man kan bruge mikrokontrollere til PWM styring af forskellige aktuatorer, herunder LED's som vist i slide 27 i LKS1Ma (Modul A slide stakken)
Der er det ikke beskrevet at det er PWM, men der bliver benyttet default periode og duty cycle ændres fra 0 til 100% ved at der skrives en værdi mellem 0 og 1023 med write_analog().

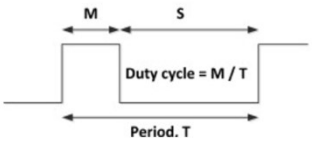
De 0-1023 kommer fra at man i microbit biblioteket har villet have konsistens mellem read-analog() og write_analog() – og hvor sidstnævnte stammer fra at der er implementeret en 10-bits A/D konverter til at omsætte analog input til digitale værdier. Det kunne naturligvis have været anderledes. Og det er anderledes i andre microkontrollere og software biblioteker.

I LYD-Kit eksemplerne har vi brugt music biblioteket/modulet til micropython til microbitten. Men man kan altså også lave lyd med PWM – Se næste slide.

Da PWM også kan bruges til at styre andet end lyden (som eksempelvis lys-dioden og servo-motoren vil det være en udmærket kobling mellem disse og grundlæggende elektronik).

Spor 1 Modul b: PWM -> lyd.

Projekt støttet af:
midt regionmidtjylland



- Med Duty cycle på 50% er der spænding på halvdelen af Perioden.
- 1 sek. = 1000msek, så med en periode på 20msek kommer der en puls 50 gange i sekundet. Dvs. en frekvens på 50Hz (1000/20 = 50).
- 1000 msek er 100000 microsek.

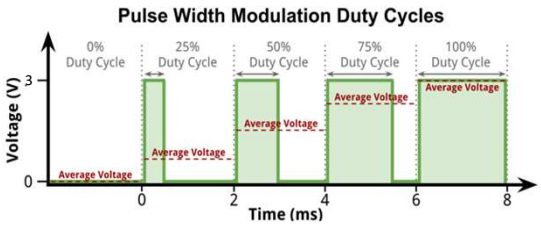
Periode (msek)	Periode (microsek)	Frekvens (Hz)
20	20000	50
10	10000	100
1	1000	1000
0.5	500	2000

```
# Generer lyd (pitch) uden brug af musik bibliotek med PWM
# Virker både på V1 og V2 med højtaler koblet til PIN0 og GND
# Bemærk at man ikke kan komme under 256 microsek iht dokumentation
# Med V2 kan man bruge pin_speaker istedet for pin0
from microbit import *

for period in range(10000,300,-100): # Fra lav hørbar frekvens til høj
    pin0.set_analog_period_microseconds(period)
    pin0.write_analog(512) # int(1023/2) -> 50% duty cycle
    sleep(50)
pin0.write_analog(0)       # for at stoppe lyden igen
```

PWM -> Lyd

Spor 1 Modul b: PWM -> Lys.

Projekt støttet af:



PWM -> Lys

Illustrationer fra beskrivelse til en NodeMCU og PWM med Arduino, men igen så kan det tilsvarende laves med Micro:Bit – slide 27 i LKS1Ma (Modul A slide stak).
Default periode og variabel duty fra 0-100% ved at bruge 0-1023 i write_analog() til PIN med dioden tilsluttet.

Kilder til Illustrationer som klikbare links på slide, men også listet sidst i stak.

Spor 1 Modul b: Reference links

Projekt støttet af:

Python Tutorial (w3schools.com)
BBC micro:bit MicroPython documentation — BBC micro:bit MicroPython 1.0.1 documentation (microbit-micropython.readthedocs.io)

Hardware (microbit.org)
Edge Connector and Pinout (microbit.org)

Kitronik Inventor's Kit for the BBC micro:bit – Kitronik Ltd
Micro:bit Tinker Kit (elecfreaks.com)
enviro:bit – Pimoroni

Surface-mount technology – Wikipedia
The World Of Microscopic Machines – YouTube

Radio — BBC micro:bit MicroPython 1.0.1 documentation (microbit-micropython.readthedocs.io) - Tutorial delen
Radio — BBC micro:bit MicroPython 1.0.1 documentation (microbit-micropython.readthedocs.io) - API delen

Radio — BBC micro:bit MicroPython 1.0.1 documentation (microbit-micropython.readthedocs.io) - Radio config

Python Dictionaries (w3schools.com)

Spor 1 Modul b: Reference links

Projekt støttet af:

Input/Output Pins — BBC micro:bit MicroPython 1.0.1 documentation (microbit-micropython.readthedocs.io) – Tutorial delen
Input/Output Pins — BBC micro:bit MicroPython 1.0.1 documentation (microbit-micropython.readthedocs.io) – API delen

Python Operators (w3schools.com)

Make awesome Switches from waste cardboard – YouTube
DIY On Off Switch – YouTube
DIY push button switch – YouTube

Horizontal Processing Tank | VARO HPT - Cook, mix and cool in one batch – YouTube
Første selv-kalibrerende temperaturføler - iTHERM TrustSens | Endress+Hauser

Using a servo with the micro:bit : Help & Support (microbit.org)
Servo:Lite Board til :MOVE mini (podconsultsbutik.dk)
180 grader servo til micro:bit (podconsultsbutik.dk)
Servo Motors High Efficiency and Power - Different types of servo motors and how they work (jameco.com)
What Are Servos – A Brief Guide – Kitronik Ltd

Pulse Width Modulation (PWM) - Electronics Basics 23 – YouTube
NodeMCU PWM with Arduino IDE | NodeMCU (electronicwings.com)
Pulse Width Modulation with analogWrite | Robotic Controls (robotic-controls.com)