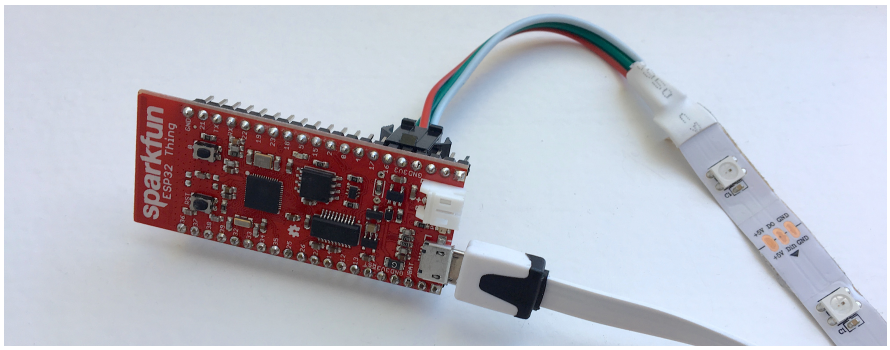
Tilslut LED-strip

Tilslut jeres LED-strip som vist på billedet her:



Forbindelserne er:

Ledningsfarve	Type	ESP32 pin	LED-strip
Hvid	Jord	GND	GND
Rød	3.3V	3V3	5VDC
Grøn	Kontrolsignal	Pin 16	DIN

Første MicroPython program

Åbn Mu-editoren. Opret en fil "ledstrip_demo.py" med dette indhold:

```
import machine
import neopixel

# 30 LED'er tilsluttet pin 16
ledstrip = neopixel.NeoPixel(machine.Pin(16), 30)

# Indstil LED'ernes farver
ledstrip[0] = (255, 0, 0)
ledstrip[9] = (0, 0, 255)

# Opdater LED'erne ved at kalde ledstrip.write()
ledstrip.write()
```

Afprøv programmet

- Tryk på Run-knappen i Mu for at køre programmet på Microcontrolleren.
- Tjek, at den første diode lyser rødt (diode 0) og den tiende lyser blåt (diode 9).

Opgaver

- Udvid programmet, så hver anden diode farves rød og hver anden farves blå for de første 10 dioder.
- *Gør koden pænere*: Opret variable til farverne (`blue = (0, 0, 255)`, osv.).

Martin Dybdal
dybber@di.ku.dk

Animationer

For at lave animationer skal vi bruge funktionen `sleep_ms` fra biblioteket `time`.

```
import machine
import neopixel
import time

red = (255, 0, 0)
blue = (0, 0, 255)

ledstrip = neopixel.NeoPixel(machine.Pin(16), 30)

# Tænd den første diode
ledstrip[0] = red
ledstrip.write()

# Vent 200 millisekunder
time.sleep_ms(200)

# Tænd den næste diode
ledstrip[1] = blue
ledstrip.write()
time.sleep_ms(200)

# Fortsæt selv ...
```

Opgaver

- Fortsæt mønsteret for de første 10 LED'er.
- En LED slukkes ved at sætte den til (0, 0, 0). Sluk den forrige LED i hvert trin, så der kun er en LED tændt ad gangen.

Gør koden pænere: Variable og Funktioner

- Lav en variabel til at styre hastigheden (delayet).
- Skriv en funktion `fill(leds, color)`, der tænder alle LED'nerne i den givne farve (tænd kun de 10 første - vi gør den smartere senere).
- Skriv en funktion `clear(leds)`, der slukker alle LED'erne (kun de 10 første).
- Skriv kode der afprøver `fill()` og `clear()` funktionerne. Kør koden og tjek at de virker som forventet.

Mini-projekt

Lav en regnbue, en istap, animation af ild, eller noget kunstnerisk i selv finder på. Brug max 10 minutter. Her er nogle flere farver:

Rød	`(255, 0, 0)`	**Slukket**	`(0, 0, 0)`
Grøn	`(0, 255, 0)`	**Gul**	`(255, 255, 0)`
Blå	`(0, 0, 255)`	**Lilla**	`(127, 0, 255)`
Hvid	`(255, 255, 255)`	**Tyrkis**	`(0, 255, 255)`

Martin Dybdal
dybber@di.ku.dk

Overfør wifi-modul

Før vi kan logge på Wifi, skal vi bruge et lille modul, der gør det lettere.

- Download filen "`wifi.py`" fra kursussiden på Absalon og gem den i mappen "mu_code" på din computer
- Tryk på "Files" i Mu og træk filen "`wifi.py`" fra højre side til venstre side for at overføre den til microcontrolleren

Forbind til WiFi

Opret en ny fil med følgende indhold og gem som "`wifi_test.py`":

```
import wifi
wifi.connect("FemTech.dk2", "0F76FMYHB0Q")
```

Når du kører filen, burde den skrive følgende (og et par andre linjer):

```
Connecting to WiFi network...
Successfully connected to "FemTech.dk2".
```

Hent vejrdata fra OpenWeatherMap

Vi har forberedt et MicroPython projekt, der henter vejrdata fra OpenWeatherMap.org:

- Hent "`weather.py`" fra kursussiden på Absalon
- Åbn den i Mu-editoren og prøv at køre koden

Nu burde der i konsol-vinduet blive vist noget i stil med:

```
{'timezone': 7200, 'cod': 200, 'dt': 1561470780, 'base': 'stations',
  'weather': [{'id': 804, 'icon': '04d', 'main': 'Clouds',
               'description': 'overcast clouds'}],
  'sys': {'message': 0.0076, 'country': 'DK',  'sunrise': 1561429582,
          'sunset': 1561492681, 'id': 1575, 'type': 1},
  'name': 'Copenhagen', 'clouds': {'all': 100}, 'coord': {'lon': 12.57, 'lat': 55.69},
  'visibility': 10000, 'wind': {'speed': 5.1, 'deg': 150}, 'id': 2618425,
  'main': {'pressure': 1023, 'humidity': 64, 'temp_min': 23.89,
           'temp_max': 28.33, 'temp': 26.27}}
Temperatur i Koebenhavn: 26.27
```

Aflæs enkeltværdier

Ovenstående udskrift kan læses som: "'timezone' har værdien 7200, 'cod' har værdien 200, 'dt' har værdien 1561470780" og så videre. Længere nede kan man se, at 'wind' har værdien `{'speed': 5.1, 'deg': 150}`, det betyder, at 'wind' indeholder to navngivne værdier ('speed' og 'deg')

I Python gør man som følger for at aflæse de enkelte værdier:

```
# Aflæs sigtbarhed (i meter)
visibility = weather_data["visibility"]
# Aflæs temperatur til variabel (grader celcius)
temperature = weather_data["main"]["temp"]
# Aflæs vindhastighed (i meter pr. sekund)
windspeed = weather_data["wind"]["speed"]
```


Martin Dybdal
dybber@di.ku.dk

Vælg mellem farver

Tilføj LED-strip initialisering øverst i "`weather.py`" (efter **`import`**'s):

```
ledstrip = neopixel.NeoPixel(machine.Pin(16), 30)

green = (0, 255, 0)
blue = (0, 0, 255)
```

Kopier dine `fill()` og `clear()` funktioner fra tidligere ind i "`weather.py`".

Tilføj derefter følgende linjer kode nederst i "`weather.py`":

```
# Funktionen tempToColor konverterer en temperatur i grader celcius
# til en (r,g,b)-farve værdi
def tempToColor(temp):
    # Grøn hvis 15 grader eller højere
    # Blå hvis lavere end 15 grader
    if temp >= 15:
        return green
    else:
        return blue

color = tempToColor(temperature)
fill(ledstrip, color)
```

Opgaver

- Ændr "`tempToColor`" så LED'erne farves røde, når temperaturen er over 35 °C
- Ændr "`tempToColor`" så LED'erne farves gule, når temperaturen er 25-35°C
- Ændr "`tempToColor`" så LED'erne farves hvide, når temperaturen er under 0°C

Gentag uendeligt

Næste skridt er at sørge for, at vores vejr-melding bliver ved med at opdatere sig. I MicroPython er der ikke en draw-funktion som i Processing, men vi kan skrive tilsvarende selv. Hvis der er kode, vi vil have kørt igen og igen kan man skrive "**`while`** `True:`" og alle linjer der er rykket ind med 4 mellemrum bagefter, vil så blive gentaget uendeligt mange gange (eller ind til programmet stoppes).

```
while True:
    kommandoer der skal gentages
```

Nu er opgaven at få jeres kode til minde om følgende pseudokode*:

```
Gentag uendeligt:
    1. Log på Wifi
    2. Hent vejr data
    3. Sluk alle dioder
    4. Vis vejrdata på lysdioder
    5. Log af Wifi
    6. Vent 2 minutter
```

(Man logger af Wifi ved at skrive `wifi.disconnect()`)

*"Pseudo" referer til at det er kode der *minder om det rigtige, men ikke er fuldstændig lig det*. I kender allerede pseudo-præfikset fra ord som pseudonym eller pseudointellektuel.

Martin Dybdal
dybber@di.ku.dk

For-løkken

Der er noget utilfredsstillende ved, at vi er blevet nødt til at skrive så mange linjer, der minder om hinanden:

```
leds[0] = color
leds[1] = color
leds[2] = color
leds[3] = color
...
```

Og hvordan skulle vi programmere, hvis vi ville have LED-strippen til at lyse med 17 LED'er, når temperaturen er 17 grader celcius?
I stedet for at gentage samme kommando igen og igen kan vi bruge *løkker* (eng. loops), de gør det nemt at gentage de samme kommandoer mange gange. Vi har allerede set et eksempel på en løkke: **`while`**-løkken. Nu skal vi lære om for-løkken.
Opret en ny fil `loops.py`:

```
import machine
import neopixel
import time

ledstrip = neopixel.NeoPixel(machine.Pin(16), 30)
purple = (127, 0, 255)

for i in range(30):
    ledstrip[i] = purple
    ledstrip.write()
    time.sleep_ms(100)
```

Opgaver

1. Prøv at ændre `range(30)` til `range(17)` eller `range(3)`
2. Prøv at tilføje linjen **`print`**`(i)` inde i løkken for at se, hvordan variablen `i` ændrer sig
3. Skriv en løkke der slukker alle LED'er (øjeblikkeligt - intet sleep)
4. Omskriv funktionen `fill(leds, color)`, så den slukker alle LED'er (ikke bare de 10 første)
5. Skriv en ny funktion `fillN(leds, color, n)`, der tænder de `n` første LED'er
6. Skriv en løkke, der tænder hver anden LED (dem med lige index)*
7. Skriv en løkke, der tænder LED'erne én ad gangen, men starter fra den sidste (29) og slutter med den første (0) - søg gerne hjælp online.

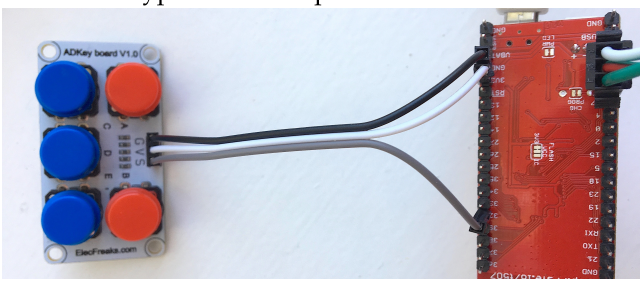
*Tip: En løsning kan være kun at tælle til 15, men lave en lille beregning på indexet. En anden løsning bruger modulus (se ark #4 fra uge 1).

Martin Dybdal
dybber@di.ku.dk

Martin Dybdal
dybber@di.ku.dk

Tilslut keypad

Tilslut et Keypad som vist på billedet:



Type	ESP32	Keypad
Jord	GND	G
3.3V	3V3	V
Signal	Pin 32	S

Hent og overfør TinkerLib

- Hent filen "tinkerlib.py" fra Absalon
- Overfør den til microcontrolleren på samme måde som med `wifi.py`

Programmering af keypad

Her er et eksempel, på hvordan keypaddet bruges:

```
import machine
import tinkerlib

def keypadButtonDown(key):
    print(key)

pin32 = machine.Pin(32, mode=machine.Pin.IN)
keypad = tinkerlib.ADKeypad(pin32, button_down=keypadButtonDown)
```

Prøv at køre programmet og tryk på knapperne.

Funktionen `keypadButtonDown` bliver kaldt hver gang, der bliver trykket på en af knapperne, lidt ligesom `keyPressed` i Processing. Argumentet key er et tal mellem 0 og 4, afhængigt af hvilken knap der blev trykket på.

Simpel nedtælling

Indtast følgende i slutningen af samme fil og kopier funktionerne "`clear`" og "`fillN`", som I skrev i filen "loops.py". Husk også at importere `neopixel` og `time` modulerne.

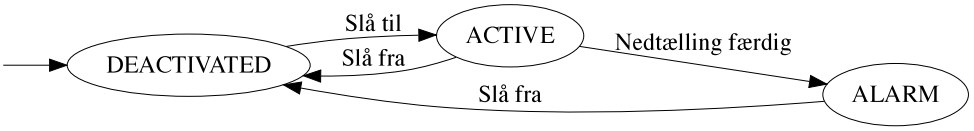
```
timeleft = 30000
blue = (0, 0, 255)
ledstrip = neopixel.NeoPixel(machine.Pin(16), 30)

while True:
    clear(ledstrip)
    fillN(ledstrip, blue, timeleft/1000)
    timeleft = timeleft - 10
    time.sleep_ms(10)
```

I skal nu ændre ovenstående kode til at bruge keypaddet til at styre "timeleft". Én af knapperne skal lægge 1000 til timeleft, en anden skal trække 1000 fra.

Martin Dybdal
dybber@di.ku.dk

Æggeur

Vi skal nu have kodet vores lille nedtællingsur om til at blive et funktionelt æggeur (eller pair-programming ur).
Før vi går i gang med at kode en løsning, skal vi planlægge, hvordan det skal virke. I denne omgang har vi på forhånd bestemt hvordan, og uret skal derfor virke efter følgende tilstandsdiagram:



Opret en global variabel `alarm_state` og sæt den til "DEACTIVATED".

Visning (View)

Vi kan opdele ændringerne i det, der har med visning at gøre, og det der har med styring at gøre. Her først de ændringer I skal lave ifht. visning:

1. Når tilstanden er DEACTIVATED vises minutter tilbage i rød farve
2. Når tilstand er ACTIVE vises minutter tilbage i blå farve
3. Når tilstanden er ALARM tændes alle 30 LED'er i grøn farve

Styring (Control)

Nu skal I ændre, hvordan styringen skal fungere jvf. tilstandsdiagrammet.
Først skal i ændre i **`while`**-løkken:

4. `timeleft` skal kun tælle ned, når tilstanden er ACTIVE.
5. Hvis tiden er gået (timeleft <= 0) og tilstanden er ACTIVE, så skal der skiftes tilstand til ALARM.

Derefter skal I ændre i `keypadButtonDown`:

6. Hvis der bliver trykket på "sæt alarm" knappen og tilstanden er DEACTIVATED, skiftes tilstand til ACTIVE.
7. Hvis der bliver trykket på "sæt alarm" knappen og tilstanden er ACTIVE eller ALARM, skiftes tilstand til DEACTIVATED.
   Samtidig sættes timeleft tilbage til 7 minutter (420000ms).

Tæl i minutter frem for timer

Lad os ændre uret til at tælle minutter frem for sekunder:

8. Sæt `timeleft` til 7 minutter (420000ms) når programmet starter
9. Hvis der trykkes på "forøg", lægges der et minut (60000ms) til `timeleft`
10. Hvis der trykkes på "reducer", trækkes der et minut (60000ms) fra `timeleft`
11. Ved visning af resterende tid divideres `timeleft` med yderligere 60.

Martin Dybdal
dybber@di.ku.dk