

IoT Kompendium



Versjon 0.3

1 Innhold

2	Kompetansemål.....	4
3	Oppsett av Adafruit ESP32-S3 TFT Feather.....	5
4	Laste ned CircuitPython på mikrokontroller	6
4.1	Laste ned bibliotekfiler	6
5	IDE (integrated development environment)	7
6	W3Schools.com.....	8
7	Mine første programmer på ESP32-S3	9
7.1	Blink.....	9
7.2	NeoPixel LED	9
8	AHT20 temperatur og fuktighetssensor.....	10
9	TFT skjerm.....	11
9.1	Legge til flere tekstelementer.....	12
10	Sette opp WiFi på mikrokontrolleren	13
10.1	Personlige data i settings.toml	13
10.2	Kode for WiFi tilkobling.....	13
10.3	Kode for å liste tilgjengelige nett	14
10.4	Hente klokke fra nett	14
10.5	Legg ut WiFi data og klokke på TFT skjerm	15
10.6	Oppdaterer klokken i hovedprogrammet	15
11	MQTT protokollen.....	16
12	Adafruit IO	17
12.1	Sett opp konto	17
12.2	Hente nøkler til koden	17
12.3	Sett opp MQTT mot Adafruit IO	18
12.4	Abonnere på melding fra AIO	20
12.4.1	Callback funksjoner	20
12.4.2	Lese kø fra AIO	21
12.5	Eksempel på dashboard	22
13	Azure IoT Central.....	23
13.1	Lag ny device på Azure IoT Central.....	23
13.2	Finn tilkoblingsnøkler for device.....	24
13.3	Koble ESP32-S3 til IoT Central.....	26
13.4	Initiering av IoT Central	26

13.5	Hovedprogram IoT Central	27
13.6	Presentere data på IoT Central	28
13.7	Lag eget dashboard	29
14	Azure IoT Hub	30
14.1	Lag ny device på Azure IoT Hub	30
14.2	Finn tilkoblingsnøkkel til IoT Hub Device	31
14.3	Koble ESP32-S3 til IoT Hub	32
14.4	Azure IoT Hub data visning	33
15	MQTT broker på Mosquitto server	34
16	Batterimonitor	38
17	OTA – Over The Air opplasting av kode	38
18	Vedlegg	39
18.1	Basis kode for wifi, aht20, neopixel for CP9	39
18.2	3d printing av holder for ESP32-S3, aht20 og batteri	39
18.3	Følg Adafruits guider	39
18.4	Håndtering av multiple SSID	40
18.4.1	<i>Passordfil</i>	40
18.4.2	<i>Wifi_passord</i>	41
18.4.3	<i>find_ssid_and_password()</i>	41
18.4.4	<i>Io_konto</i>	41
19	Logg på Azure for lærer-bruker	42
19.1	Presentere data på IoT Central	44
20	Legg til Elevgruppe i kvs-central	47

2 Kompetansemål

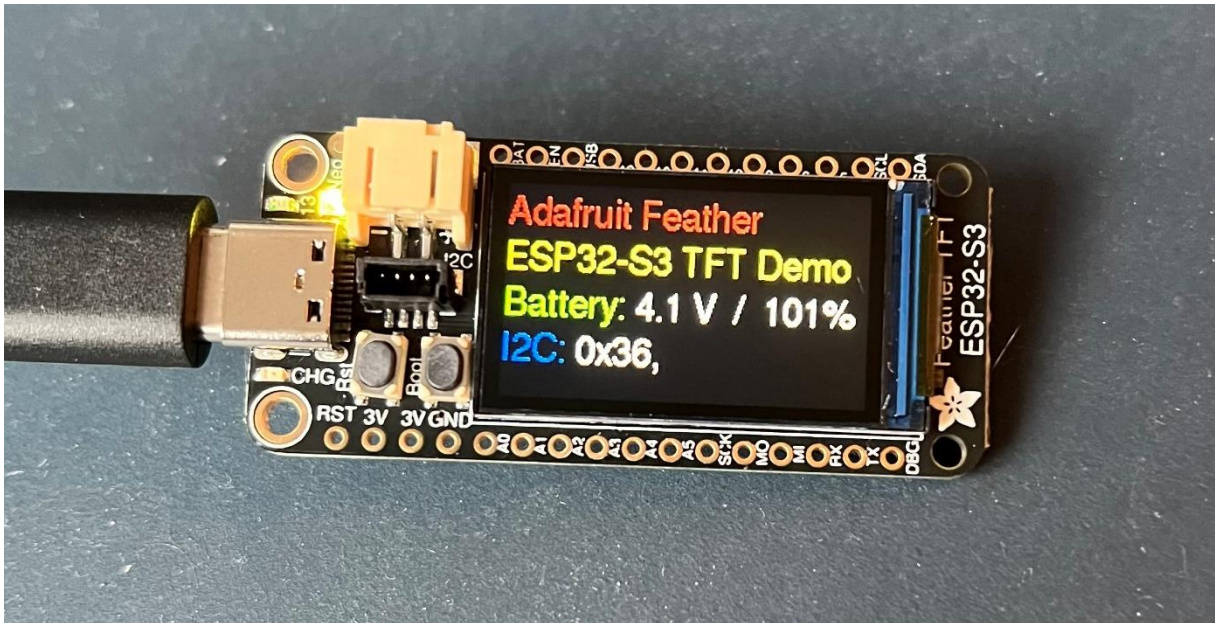
For Vg3 har UDIR utformet følgende kompetansemål som dekkes i dette emnet:



- Kjerneelementet konfigurering og programmering handler om å konfigurere elektronisk utstyr og informasjons- og kommunikasjonssystem. Vidare handler det om å programmere mikrokontrollbaserte system. Kjerneelementet handler og om å teste, tilpasse og feilrette program
- installere, programmere og setje i drift trådlause sensor- og kommunikasjonssystem med ulik rekkjevidd og lågt straumforbruk, og beskrive eigenskapane til og bruken av ulike trådlause sensor- og kommunikasjonssystem
- reparere og vedlikehalde trådlause sensor- og kommunikasjonssystem med høvesvis kort og lang rekkjevidd og lågt straumforbruk ved hjelp av eigna verktøy, programvare og dokumentasjon

3 Oppsett av Adafruit ESP32-S3 TFT Feather

På vg3 brukes **Adafruit ESP32-S3 TFT Feather** til undervisningen.



Adafruit har sin egen guide for oppsett av denne mikrokontrolleren [her](#).

- [Overview](#)
- [Pinouts](#)
- [Low Power Usage](#)
- [Power Management](#)
- [CircuitPython](#)
- [CircuitPython Essentials](#)

Les gjennom Overview og Pinouts i første omgang.

4 Laste ned CircuitPython på mikrokontroller

For at dere skal kunne programmere kortet må dere laste ned siste versjon av CircuitPython på kortet i form av en xxx.UF2 fil. Den inneholder «operativsystemet» og en god del av de vanligste bibliotek modulene.

- Følg guide her for å sette kortet i boot modus:

[CircuitPython](#)

- Slik ser kortet ut i FT232RL BOOT
- Kaldstart kortet når det har fått ny UF2 kode




For 2024 er det Circuit Python 9.0.0 som gjelder

CircuitPython 9.0.0

This is the latest **stable** release of CircuitPython that will work with the Feather ESP32-S3 TFT PSRAM.

Use **this release** if you are new to CircuitPython.

[Release Notes for 9.0.0](#)


DOWNLOAD .UF2 NOW 

4.1 Laste ned bibliotekfiler

Sammen med Circuit Python må det også lastes ned tilhørende bibliotekfiler. Det **må** være samme versjon av Circuit Python og bibliotekene. Bibliotekene laster dere ned som en bundle og den pakkes ut og lagres lokalt på skolepc. I tillegg til en \lib katalog inneholder bundle-filen også nesten 1800 eksempel filer for hvordan bibliotekene brukes.

Når dere trenger et bibliotek til koden dras de enkelt over til \lib katalog på CIRCUITPY.

Bibliotekene finner dere [her](#) eller for versjon 9 ved å trykke på knappen under.

adafruit-circuitpython-bundle-9.x-mpy-20240307.zip 

5 IDE (integrated development environment)

Det er mange editorer som kan brukes for å programmere i Circuit Python. Skolen har valgt å bruke MU som er editoren Adafruit anbefaler for nybegynnere.


Dere kan selvsagt bruke andre editorer, men da er dere mer på egenhånd.

Last ned eller oppgrader MU [her](#)

Download Mu

The simplest and easiest way to get Mu is via the official installer for Windows or Mac OSX (we no longer support 32bit Windows). We also have an experimental AppImage for Linux users running on Intel based hardware.

The current recommended version is Mu 1.2.0. We advise people to update to this version via the links for each supported operating system. All previous beta versions of Mu [can be downloaded from here](#).



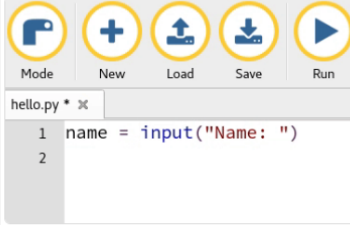
Windows Installer

Download
Instructions

Er dere nye eller har glemt hvordan MU brukes bør dere gå gjennom de tre korte tutorials nedenfor som dere finner [her](#).

Tutorials

Each tutorial is like a self-contained lesson, explaining a specific aspect of Mu so you have the skills needed to achieve your learning and coding goals. Most of all, they're written to be both educational and entertaining.



Start Here!

All the basics you need to know to get started and help with first steps in using Mu. **If you only read one thing, make it this!**

Read Level: **SUPER EASY**

Adafruit CircuitPython
Use CircuitPython on Adafruit's line of boards.

BBC micro:bit
Write MicroPython for the BBC micro:bit.

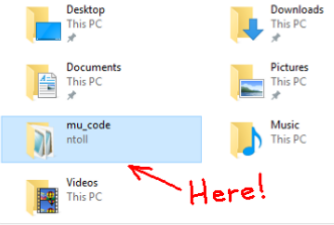
Pygame Zero
Make games with Pygame Zero.

Python 3
Create code using standard Python 3.

Using Modes

What are modes? How are they used in Mu? We show how modes help you make all sorts of fun and awesome projects.

Read Level: **SUPER EASY**



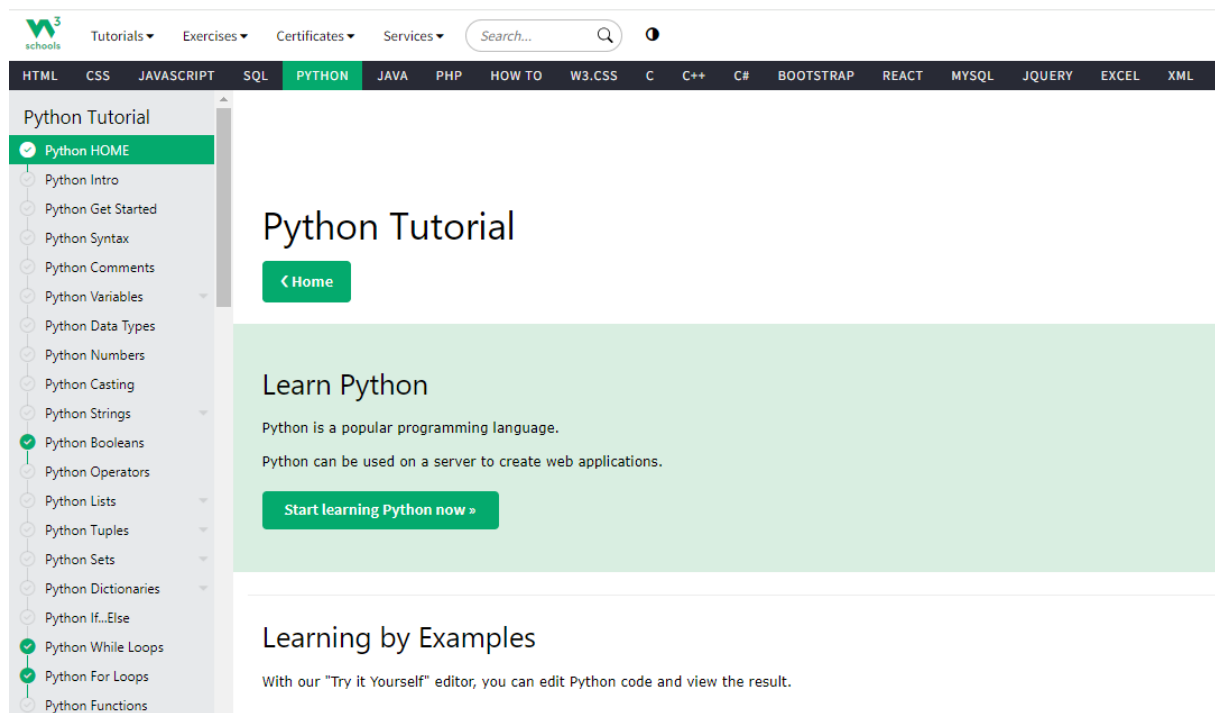
Where are my Files?

Mu stores all your code in a special folder so it's always in the right place. This tutorial explains where and how to find it.

Read Level: **SUPER EASY**

6 W3Schools.com

Dersom dere lurer på noe ang python kan dere bruke W3Schools eller andre hjelpemidler på nett. ChatGPT gir ofte god, men ofte også avansert hjelp. For å øke forståelsen er det lurt å programmere selv.



The screenshot shows the W3Schools website interface. At the top, there is a navigation bar with links for Tutorials, Exercises, Certificates, and Services, along with a search bar. Below this is a dark navigation bar with various programming topics: HTML, CSS, JAVASCRIPT, SQL, PYTHON (highlighted), JAVA, PHP, HOW TO, W3.CSS, C, C++, C#, BOOTSTRAP, REACT, MYSQL, JQUERY, EXCEL, and XML. On the left side, a sidebar menu titled "Python Tutorial" lists various topics, with "Python HOME" selected and highlighted in green. The main content area features a large heading "Python Tutorial" with a green button labeled "< Home". Below this is a green banner with the text "Learn Python" and a sub-heading "Python is a popular programming language. Python can be used on a server to create web applications." A green button labeled "Start learning Python now >" is positioned below the text. Further down, there is a section titled "Learning by Examples" with the text "With our 'Try it Yourself' editor, you can edit Python code and view the result."

7 Mine første programmer på ESP32-S3

På mikrokontrolleren er det nå et «Hello World» program `code.py` som kjører.

Åpne MU i CircuitPython mode og Load `code.py` fra CIRCIUTPY (drive for ESP32-S3).

Endre «World» til ditt eget navn og Save. Koden blir lagret på kortet og programmet kjører. Du kan se output i MU eller på skjermen.

Dere skal først følge guide og kjøre programmene Blink og NeoPixel. Det vil da få en grunnleggende forståelse for hvordan dere lager et program, overfører det til mikrokontrolleren og kjører programmet. Dere vil også få trening i å overføre en bibliotekfil for å kunne kjøre NeoPixel programmet.

- Gå til [CircuitPython Essentials](#)

7.1 Blink

Følg guiden og kjør programmet.

Prøv å endre frekvensen for blinkingen.

7.2 NeoPixel LED

Følg guiden og kjør programmet.

Prøv å endre på fargene på LED og kjør regnbue programmet.

[CircuitPython Essentials](#) inneholder også eksempler på hvordan dere kan bruke ulike inn- og utganger på kortet.

8 AHT20 temperatur og fuktighetssensor

Skolen har kjøpt inn en kombinert temperatur og fuktighetssensor av typen AHT20.

Dere finner dokumentasjon og eksempler på bruk [her](#).



Koble AHT20 til STS3-S3 med medfølgende JST kabel.

Last ned følgende bibliotek:

- adafruit_ahtx0.mpy
- adafruit_bus_device

Sett opp kortet i koden:

```
import board
import adafruit_ahtx0
aht20 = adafruit_ahtx0.AHTx0(board.STEMMA_I2C())
```

I program loopen kan man skrive ut sensor data på denne måten:

```
83 # Hovedprogram
84 while True:
85     temp = float(aht20.temperature)
86     humid = float(aht20.relative_humidity)
87     print(f"AHT20 - Temperature: {temp:.1f} C")
88     print(f"AHT20 - Humidity: {humid:.1f} %")
89     time.sleep(10)
```

9 TFT skjerm

Last ned følgende bibliotek til \lib på kortet ditt:

`adafruit_display_text`

For å sette opp TFT displayet må man først laget et objekt

```
skjerm = board.DISPLAY
```

Man lager så en gruppe som skal inneholde alle elementene du ønsker å vise på skjermen. Det kan være både tekst og bitmaps.

```
gruppe = displayio.Group()
```

Så lager man teksten en ønsker å vise på skjermen og legger den til i gruppen

```
eier_tekst = bitmap_label.Label(
    terminalio.FONT,
    text="Mikal's ESP32-S3",
    x=0, y=15, color=GUL, scale=2)
gruppe.append(eier_tekst)
```

Her har jeg brukt den innebygde fonten terminalio.FONT. Teksten er oppgitt mellom «». x og y er posisjoner på skjermen, color er farge i hex og scale er størrelse fra 1-?

Til slutt settes root_group til gruppen du har laget

```
skjerm.root_group = gruppe
```

```
20 # TFT skjerm
21 import board
22 import displayio
23 import terminalio
24 from adafruit_display_text import bitmap_label
25 WHITE = 0xFFFFFF
26 RED = 0xFF0000
27 GREEN = 0x00FF00
28 BLUE = 0x0000FF
29 GUL = 0xFFFF00
30 LILLA = 0xFF00FF
31
32 print("Sett opp tft")
33 skjerm = board.DISPLAY
34 gruppe = displayio.Group()
35 eier_tekst = bitmap_label.Label(terminalio.FONT, text="Mikal's ESP32-S3", x=0, y=15, color=GUL, scale=2)
36 gruppe.append(eier_tekst)
37 skjerm.root_group = gruppe
```

9.1 Legge til flere tekstelementer

Man kan legge til eller fjerne flere tekstelementer til skjerm gruppen lengre ned i koden som vist nedenfor. Etter at du har laget AHT20 objektet kan du hente data og vise de på skjermen.

```
79 temp = float(aht20.temperature)
80 humid = float(aht20.relative_humidity)
81 temp_text = bitmap_label.Label(terminalio.FONT, text=f"Temperatur: {temp:.1f} C", x=0, y=50, color=GREEN, scale=2)
82 humid_text = bitmap_label.Label(terminalio.FONT, text=f"Fuktighet: {humid:.1f} %", x=0, y=80, color=BLUE, scale=2)
83 gruppe.append(temp_text)
84 gruppe.append(humid_text)
85 skjerm.root_group = gruppe
```

Linje 79 og 80: har jeg hentet henholdsvis temperatur og fuktighet.

Linje 81 og 82: bygget opp tekstelement

Linje 83 og 84: lagt tekstelementer inn i gruppe

I hovedprogrammet trenger du kun å oppdatere «text» delen av temp_text som vist nedenfor.

```
87 # Hovedprogram
88 while True:
89     temp = float(aht20.temperature)
90     humid = float(aht20.relative_humidity)
91     print(f"AHT20 - Temperature: {temp:.1f} C")
92     print(f"AHT20 - Humidity: {humid:.1f} %")
93
94     # oppdater TFT
95     temp_text.text = f"Temperatur: {temp:.1f} C"
96     humid_text.text = f"Fuktighet: {humid:.1f} %"
97
98     time.sleep(10)
```

Oppdater koden din og endre temperatur på sensor ved å holde på den. Verifiser at temperaturen endrer seg på skjermen.

10 Sette opp WiFi på mikrokontrolleren

10.1 Personlige data i settings.toml

Det er anbefalt å lage en fil som inneholder private data adskilt fra koden din. Det er fordi da kan du dele koden din eller laste den opp på GitHub uten at dine data kommer på avveie.

I tidligere versjoner kalte Adafruit eksempel filen for secrets.py og i CP9 er den kalt **settings.toml**. Denne filen ligger CIRCUITPY driven din.

Her legger du inn SSID og tilhørende passord. Dersom du endrer navn på konstanter, må du også gjøre det i koden din. CP9 bruker «os» biblioteket til å hente disse fra settings filen.

```
CIRCUITPY_WIFI_SSID = "IoT-Eksamen"
```

```
CIRCUITPY_WIFI_PASSWORD = "k0bletil"
```

10.2 Kode for WiFi tilkobling

Her er kode for tilkobling til WiFi som foreslått i CP9

```
1 import time
2 import wifi
3 import microcontroller
4 import os
5
6 # koble til wifi
7 try:
8     ssid = os.getenv("CIRCUITPY_WIFI_SSID")
9     passord = os.getenv("CIRCUITPY_WIFI_PASSWORD")
10    wifi.radio.connect(ssid, passord)
11    print(f"Tilkoblet: {ssid}")
12 except Exception as e:
13    print(f"Wifi tilkobling feilet. Error:{e} - kort restarter om 30s")
14    time.sleep(30)
15    microcontroller.reset()
16 print("Min IP adresse er:", (wifi.radio.ipv4_address))
```

Nå er kortet på nett og har fått ip adresse

10.3 Kode for å liste tilgjengelige nett

Dersom du ønsker å liste opp alle tilgjengelige nett, vise signalstyrke og kanal kan du scanne nettverket på denne måten:

```

7 #Liste opp alle tilgjengelige nett
8 print("Available WiFi networks:")
9 for network in wifi.radio.start_scanning_networks(stop_channel=13):
10     print(f"SSID:{str(network.ssid, 'utf-8')} RSSI:{network.rssi} Channel:{network.channel}")
11 wifi.radio.stop_scanning_networks()

```

10.4 Hente klokke fra nett

Vi kan hente klokken fra internett og sette den på kortet. Da trenger du å importere følgende bibliotek.

adafruit_ntp.mpy må du kopiere fra bibliotek på skolepc.

```

import rtc
import socketpool
import adafruit_ntp

68 # hent klokke fra nettverk
69 pool = socketpool.SocketPool(wifi.radio)
70 ntp = adafruit_ntp.NTP(pool, tz_offset=2)
71 rtc.RTC().datetime = ntp.datetime
72 tid = rtc.RTC().datetime
73 print(f"Tid: {tid[3]:02d}:{tid[4]:02d} {tid[2]:02d}/{tid[1]:02d}/{tid[0]:02d}")

```

Du har nå satt et rtc (real time clock) objekt på kortet ditt. Linje 59 viser hvordan du henter klokken inn i en variabel «tid» i et datetime format som er vist nedenfor. Linje 60 er eksempel på utskrift av tidspunkt.

Ln70: Klokken som hentes er GMT. For lokaltid må Tz_offset=2 brukes ved sommertid og Tz_offset=1 ved vintertid.

Ln73: «:02d» formateringen gjør at alle tall skrives ut med minst 2 siffer.

```

cal = ntp.datetime
year  = cal[0]
mon   = cal[1]
day   = cal[2]
hour  = cal[3]
minute = cal[4]

```

Henting av klokke fra internett feiler innimellom. La kortet starte på nytt så ordner det seg som oftest.

10.5 Legg ut WiFi data og klokke på TFT skjerm

Du kan legge til flere linjer i displayet lenger ned i koden når du er koblet opp på nett. Her har jeg lagt til klokken, SSID og IP Adr.

Dersom vi vil fjerne et element på skjermen bruker man `remove()` funksjonen

```

63 tidspunkt = f"Tid: {tid[3]:{tid[4]} {tid[2]}/{tid[1]}/{tid[0]}"
64 ssid_text = bitmap_label.Label(terminalio.FONT, text=f"SSID: {ssid}", x=0, y=130, color=BLUE, scale=1)
65 ipadr_text = bitmap_label.Label(terminalio.FONT, text=f"IP: {wifi.radio.ipv4_address}", x=120, y=130, color=BLUE, scale=1)
66 time_text = bitmap_label.Label(terminalio.FONT, text=tidspunkt, x=0, y=110, color=LILLA, scale=1)
67 gruppe.append(ssid_text)
68 gruppe.append(ipadr_text)
69 gruppe.append(time_text)
70 gruppe.remove(oppstart_tekst)
71 skjerm.root_group = gruppe

```

10.6 Oppdaterer klokken i hovedprogrammet

```

91 # Hovedprogram
92 while True:
93     temp = float(aht20.temperature)
94     humid = float(aht20.relative_humidity)
95     print(f"AHT20 - Temperature: {temp:.1f} C")
96     print(f"AHT20 - Humidity: {humid:.1f} %")
97
98     # oppdater TFT
99     temp_text.text = f"Temperatur: {temp:.1f} C"
100    humid_text.text = f"Fuktighet: {humid:.1f} %"
101    tid = rtc.RTC().datetime
102    time_text.text = f"Tid: {tid[3]:02d}:{tid[4]:02d} {tid[2]:02d}/{tid[1]:02d}/{tid[0]:02d}"
103    time.sleep(10)

```

Ln101: Legg merke til at «tid» henter tiden fra det lokal rtc objektet og ikke fra nett.

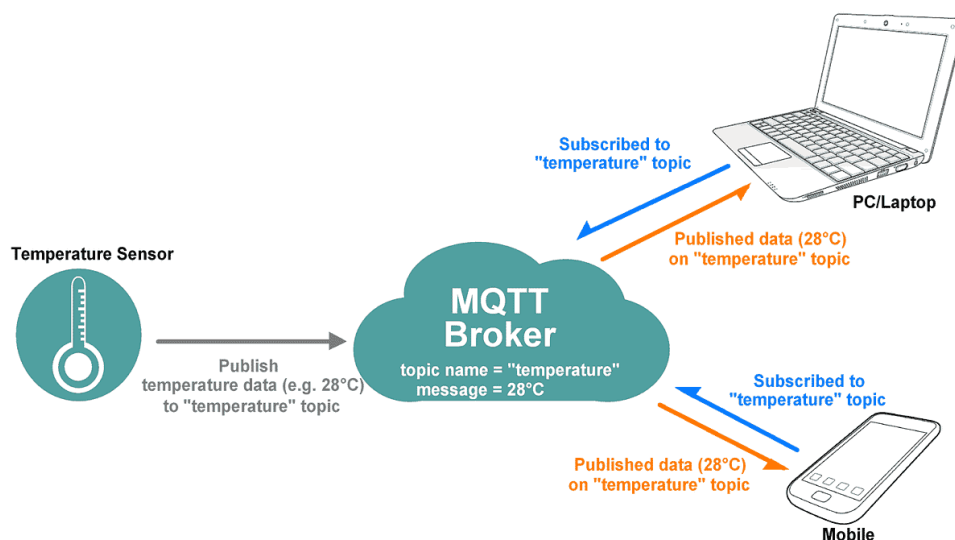
11 MQTT protokollen

MQTT: The Standard for IoT Messaging

MQTT is an OASIS standard messaging protocol for the Internet of Things (IoT). It is designed as an extremely lightweight publish/subscribe messaging transport that is ideal for connecting remote devices with a small code footprint and minimal network bandwidth. MQTT today is used in a wide variety of industries, such as automotive, manufacturing, telecommunications, oil and gas, etc.

Les mer om MQTT her:

Getting started with MQTT



MQTT Broker er en applikasjon som administrerer data. Det er mange som tilbyr denne tjenesten som en sky tjeneste. De fleste aktørene har en gratis versjon som tillater begrenset trafikk. Dere er kjent med Adafruit IO fra bruk på skolen tidligere.

I dette kurset skal vi også se på hvordan vi kan sette opp en MQTT Broker på vår egen server.

Noder som er tilkoblet kan sende (publish) data til broker. For at broker skal kunne sortere data blir de lagret med et topic. Dette topic kan andre enheter abonnere (subscribe) på. Da vil broker sende data til alle abonnenter tilhørende et topic når det blir oppdatert hos broker.

Se neste kapittel for eksempel på hvordan dette gjøres i koden.

12 Adafruit IO

12.1 Sett opp konto

For å sende data til Adafruit IO må man først ha en gratis konto hos Adafruit.

Dersom du ikke har konto, kan du følge guide [her](#)

For å lage feeds og dashboards følger du guide [her](#)

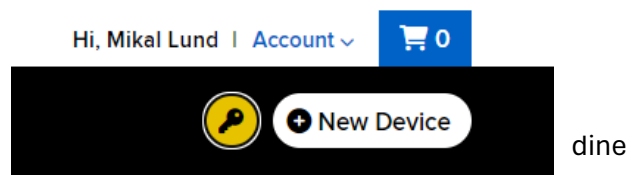
I kodeeksemplene som følger bruker jeg følgende feeds:

Default	
Feed Name	Key
<input type="checkbox"/> Fuktighet	humid-s3
<input type="checkbox"/> Temperatur	temp-s3

12.2 Hente nøkler til koden


Når du har laget konto henter du frem IO nøkkelen.

Trykk på den gule nøkkelen og du får opp egne koder.



YOUR ADAFRUIT IO KEY ×

Your Adafruit IO Key should be kept in a safe place and treated with the same care as your Adafruit username and password. People who have access to your Adafruit IO Key can view all of your data, create new feeds for your account, and manipulate your active feeds.



If you need to regenerate a new Adafruit IO Key, all of your existing programs and scripts will need to be manually changed to the new key.

Username

Active Key REGENERATE KEY

[Hide Code Samples](#)

Arduino

```
#define IO_USERNAME "KrokeideVGS"
#define IO_KEY      "aio_CJKj34aXBayGT0vJgS0wZPgzs1h0"
```

Linux Shell

```
export IO_USERNAME="KrokeideVGS"
export IO_KEY="aio_CJKj34aXBayGT0vJgS0wZPgzs1h0"
```

Scripting

```
ADAFRUIT_IO_USERNAME = "KrokeideVGS"
ADAFRUIT_IO_KEY = "aio_CJKj34aXBayGT0vJgS0wZPgzs1h0"
```

Kopier de to linjene under Scripting og lim de inn i filen settings.toml på kortet ditt.

12.3 Sett opp MQTT mot Adafruit IO

Last ned følgende bibliotek:

adafruit_minimqtt

adafruit_io

adafruit_connection_manager.mpy

```
import ssl
import adafruit_minimqtt.adafruit_minimqtt as MQTT
from adafruit_io.adafruit_io import IO_MQTT
```

```
85 # Sett opp Adafruit MQTT
86 def tilkoblet_AIO(client):
87     print("Koblet til Adafruit IO!")
88
89 broker_adr    = "io.adafruit.com"
90 aio_username  = os.getenv("ADAFRUIT_IO_USERNAME")
91 aio_key       = os.getenv("ADAFRUIT_IO_KEY")
92 temp_feed     = "temp-s3"
93 humid_feed    = "humid-s3"
94
95 # Initialize a new MQTT Client object
96 mqtt_client = MQTT.MQTT(
97     broker    = broker_adr,
98     username= aio_username,
99     password= aio_key,
100    socket_pool=pool,
101    ssl_context=ssl.create_default_context(),
102 )
```

Linje86: denne funksjonen blir automatisk kjørt når AIO er tilkoblet. Det er her du eventuelt kan abonnere på meldinger

Linje 89: nettadressen til AIO

Linje 90-91: henter data fra settings.toml

Linje 92-93: lager variable lik «KEY» for dine feeds på AIO

Linje 96: lager mqtt_client objekt.

```

104 # Initialize Adafruit IO MQTT "helper"
105 io = IO_MQTT(mqtt_client)
106 # Fortell AIO hvilken metode den skal kalle når oppkoblet
107 io.on_connect = tilkoblet_AIO

```

Linje 104-107: CP9 har innført et `io_helper` bibliotek for å gjøre det enklere.

```

121 while True:
122     try:
123         # Hvis vi har mistet kontakten med AIO...
124         if not io.is_connected:
125             print("Kobler til Adafruit IO...")
126             io.connect()
127
128             temp = float(aht20.temperature)
129             humid = float(aht20.relative_humidity)
130             print(f"AHT20 - Temperature: {temp:.1f} C")
131             print(f"AHT20 - Humidity: {humid:.1f} %")
132
133             # oppdater AIO
134             io.publish(temp_feed, f"{temp:.1f}")
135             io.publish(humid_feed, f"{humid:.1f}")
136
137             # oppdater TFT
138             temp_text.text = f"Temperatur: {temp:.1f} C"
139             humid_text.text = f"Fuktighet: {humid:.1f} %"
140             tid = rtc.RTC().datetime
141             time_text.text=f"Tid: {tid[3]:02d}:{tid[4]:02d} {tid[2]:02d}/{tid[1]:02d}/{tid[0]:02d}"
142             time.sleep(10)
143
144     except Exception as e:
145         print(f"Klarer ikke sende data til AIO. Error:{e} - kort restarter om 30s")
146         time.sleep(30)
147         microcontroller.reset()

```

Linje 124-126: kobler opp mot AIO

Linje 134-135: Sender data til AIO. Data må være av type «text»

Linje 122: Har lagt inn en try-except her for å resette kortet om en ikke klarer å koble på AIO

Utskrift i terminalvindu og AIO:

```

Tilkoblet: ST117-gjest
Min IP adresse er: 192.168.85.58
Tid: 13:18 22/3/2024
Kobler til Adafruit IO...
Koblet til Adafruit IO!
AHT20 - Temperature: 22.5 C
AHT20 - Humidity: 32.4 %

```

Default		
Feed Name	Key	Last value
<input type="checkbox"/> Fuktighet	humid-s3	32.9
<input type="checkbox"/> Temperatur	temp-s3	22.2

12.4 Abonnere på melding fra AIO

Som eksempel på abonnering kan vi lage til styring av neopixel på kortet fra AIO

Lag en ny feed for denne og kall den for eksempel «neopixel-s3»

Kopier neopixel.my bibliotek til kortet om du ikke har gjort det tidligere

Sett opp NeoPixel i koden

```
32 # Initialise NeoPixel
33 pixel = neopixel.NeoPixel(board.NEOPIXEL, 1, brightness=0.3)
```

Legg til neo_feed variable

```
95 temp_feed = "temp-s3"
96 humid_feed = "humid-s3"
97 neo_feed = "neopixel-s3"
```

Legg til abonnering på neo_feed

```
87 # Sett opp Adafruit MQTT
88 def tilkoblet_AIO(client):
89     print("Koblet til Adafruit IO!")
90     client.subscribe(neo_feed)
```

12.4.1 Callback funksjoner

En callback funksjon er en måte å fortelle en annen applikasjon hvilken funksjon hos deg som skal kalles ved en hendelse. I linje 111 nedenfor forteller vi mqtt at når tilkobling er gjort så kalles funksjonen som her er kalt «tilkoblet_AIO». Denne funksjonen må man selv definere.

Det lages også en callback funksjon som kalles når det kommer inn meldinger fra broker inn i køen. Den er i dette tilfellet kalt «handtere_melding». Dere kan fritt navngi callback funksjonene.

Legg til on_message metode for å fortelle hvor meldinger fra io skal håndteres

```
110 # Fortell AIO hvilken metode den skal kalle når oppkoblet
111 io.on_connect = tilkoblet_AIO
112 io.on_message = handtere_melding
```

Lag funksjon for å dekode melding fra AIO

```
92 # funksjon som håndterer meldinger fra AIO
93 def handtere_melding(klient, topic, melding):
94     print(f"Feed {topic} received new value: {melding}")
95     if topic == neo_feed:
96         pixel.fill(int(melding[1:], 16))
```

12.4.2 Lese kø fra AIO

I hovedprogrammet må du lese kø fra AIO i loopen i hovedprogrammet som vist i linje 129. Dersom det er en melding i køen vil den automatisk bli sendt til callback funksjonen som er spesifisert i linje 112 over.

```
120 # Hovedprogram
121 while True:
122     try:
123         # Hvis vi har mistet kontakten med AIO...
124         if not io.is_connected:
125             print("Kobler til Adafruit IO...")
126             io.connect()
127
128         # sjekk om det er meldinger fra AIO i køen
129         io.loop()
```

Svakhet i programmet:

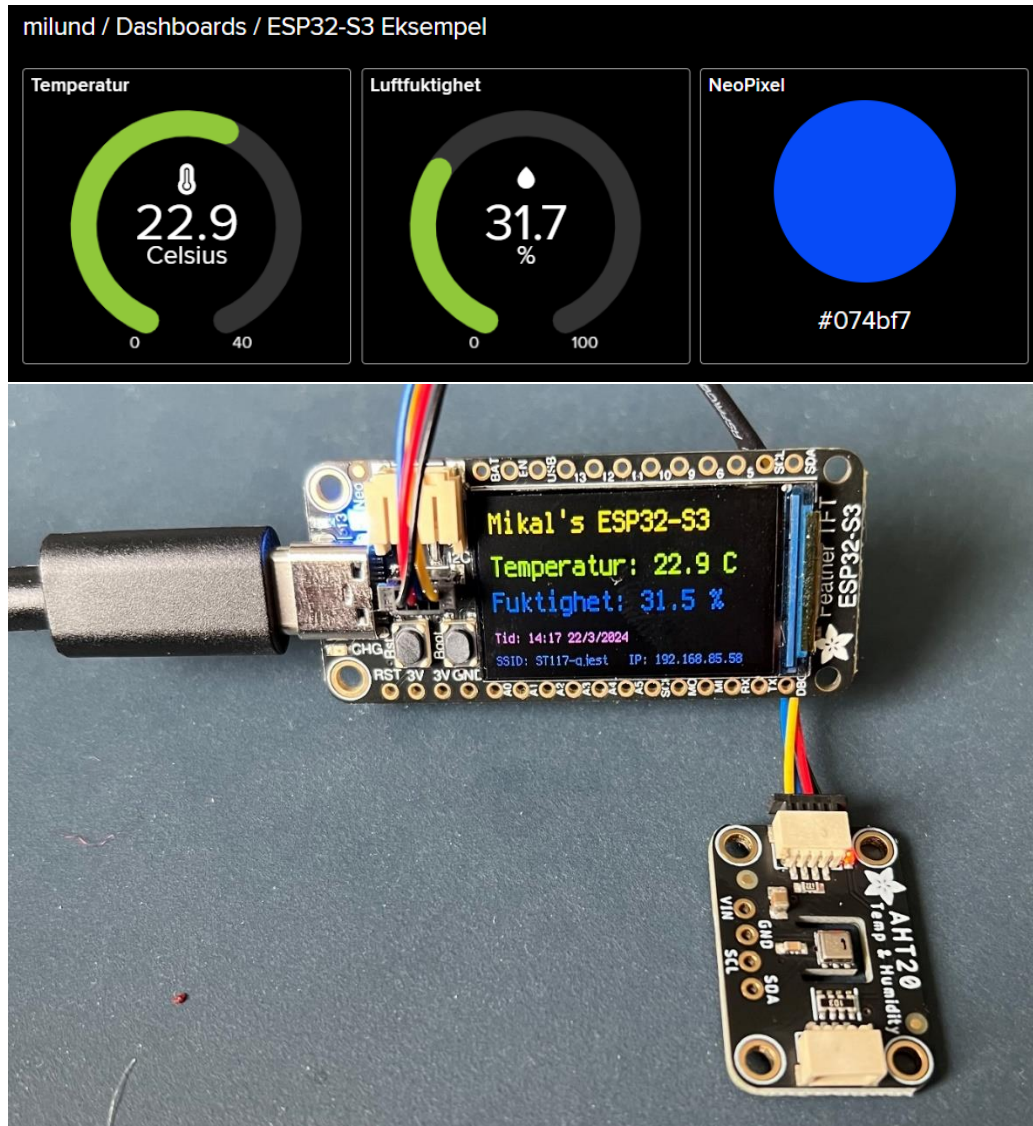
Nå er det en sleep i koden på 10s mellom hver iterasjon. Det betyr at meldingskøen fra AIO kun blir lest hvert 10s.

Kan du forbedre denne løsningen uten at du sender data oftere til AIO?

12.5 Eksempel på dashboard

Her er et eksempel på hvordan du kan bygge opp et dashboard i AIO for å presentere data.

Kan du lage ditt eget?



13 Azure IoT Central

13.1 Lag ny device på Azure IoT Central

Elev logger inn med sin egen bruker her:

<https://kvs-central.azureiotcentral.com>

Dersom linken ikke fungerer må lærer legge deg til som APP Operator i Azure kvs-central.

Eksempel Sjetes bruker:

Device name	Device ID	Device status	Device template	Organization
ESP32-S3 Elev	22zo3rva6p5	Provisioned	ESP32-S3 Elev	kvs-central
esp32s3-ml	esp32s3-ml	Provisioned	ESP32-S3	kvs-central
mlu-mobil	mlu-mobil	Provisioned	IoT Plug and Play mobile	kvs-central
my-phone-as-device-Ptz_...	my-phone-as-device-Ptz_El...	Provisioned	IoT Plug and Play mobile	kvs-central

Velg +NEW og legg til nytt device

Create a new device ×

To create a new device, select a device template, a name, and a unique ID. [Learn more](#)

Device name * ⓘ

Device ID * ⓘ

Organization * ⓘ

Device template *

ESP32-S3 Elev
▼

Simulate this device?
 A simulated device generates telemetry that enables you to test the behavior of your application before you connect a real device.

No

Azure IoT Edge device?
 Azure IoT Edge moves cloud analytics and custom business logic from the cloud to your devices.

No

Create
Cancel


Velg et «device name» som inkludere ditt egen navn eller initialer.

Velg «Device template»: ESP32-S3 Elev. La andre innstillinger være default.

Trykk «Create»

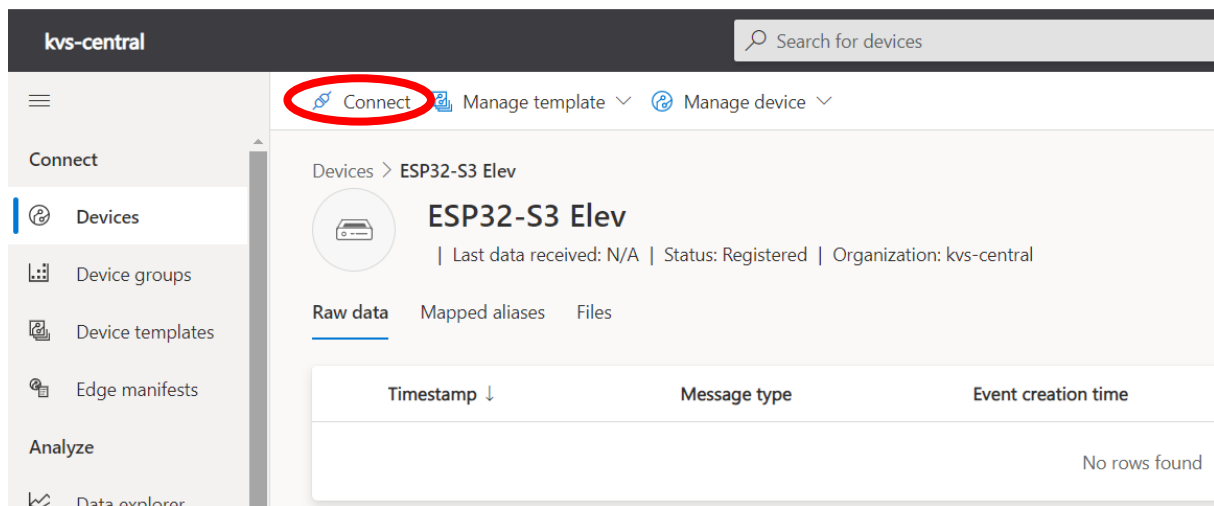
13.2 Finn tilkoblingsnøkler for device

Velg ditt device i listen

 **All devices**
Device explorer helps you see all your devices. Detailed information like device raw data helps you troubleshoot. [Learn more](#)

Device name	Device ID	Device status	Device template	Organization
ESP32-Sjete	1fngtcy9ajn	Provisioned	ESP32-S3 Elev	kvs-central
ESP32-S3 Elev	22zo3rva6p5	Provisioned	ESP32-S3 Elev	kvs-central

Trykk «Connect» for å få frem nøkler til koden din




The screenshot shows the IoT Hub interface for the organization 'kvs-central'. A search bar at the top right contains the text 'Search for devices'. The left sidebar has a 'Connect' section with a sub-menu 'Devices' highlighted. In the main content area, the 'Connect' button is circled in red. Below it, the device 'ESP32-S3 Elev' is shown with its status 'Registered' and organization 'kvs-central'. The 'Raw data' tab is selected, showing a table with columns 'Timestamp ↓', 'Message type', and 'Event creation time'. The table is currently empty, displaying 'No rows found'.

Device connection groups ×

ID scope ⓘ


 


Device ID ⓘ


Choose the connection type for this device. You can change this later if you need to.

Authentication type

 **Key** QR code

Shared Access Signatures (SAS) use security tokens and keys to connect to IoT Central. Use the SAS keys from the default enrollment group shown below to register your device. [Learn more](#) 

Primary key ⓘ

Kopier «ID scope», «Device ID» og «Primary key» til settings.toml på mikrokontrolleren.

```
# Azure IoT Central private nøkler
IOT_CENTRAL_IDSCOPE = "0ne00C0A9E5"
IOT_CENTRAL_DEVICEID = "1fngtcy9ajj"
IOT_CENTRAL_PRIMARYKEY = "PLSZuYNsOlOp2KVrmMCTmTa7IJNipXgKHfE/bcpZJrE="
```

13.3 Koble ESP32-S3 til IoT Central

Nå skal vi koble kortet opp mot Azure IoT Central.

Hent kode fra Vedlegg [her](#).

Bibliotek du trenger for å kjøre kode mot Azure IoT Central. Noen har du allerede om du har kjørt Adafruit IO.

Adafruit_azureiot

Adafruit_logging.mpy

Adafruit_binascii.mpy

13.4 Initiering av IoT Central

```
17 # IoT Central
18 import json
19 from adafruit_azureiot import IoTCentralDevice
20 from adafruit_azureiot.iot_mqtt import IoTResponse

96 # sett opp Azure
97 esp = None
98 idscope = os.getenv("IOT_CENTRAL_IDSCOPE")
99 devid = os.getenv("IOT_CENTRAL_DEVICEID")
100 primkey = os.getenv("IOT_CENTRAL_PRIMARYKEY")
101
102 az_device = IoTCentralDevice(pool, esp, idscope, devid, primkey)
103
104 print("Connecting to Azure IoT Central...")
105 az_device.connect()
106 print("Connected to Azure IoT Central!")
```

Linje 98-100: henter nøklene fra settings.toml

13.5 Hovedprogram IoT Central

Man bør sette opp et intervall for hvor ofte en sender data til Azure. Gratis versjonen som vi bruker, har en begrensning på 8000 meldinger pr dag totalt.

I koden har jeg satt den til 100s, mens jeg tester. Kanskje den kan stå på 500s etter hvert.

Hvert 10 sekund oppdateres temperatur og fuktighet på skjermen.

Hvert 100 sekund sendes data til Azure og klokken viser når data er sendt.

```

111 # Hovedprogram
112 AZURE_OPPDATER = 100 # hvor ofte oppdatere Azure
113 TFT_OPPDATER = 10 # hvor ofte oppdatere TFT display
114 azure_clock = 100 # loopteller initiert til 100 for å oppdatere Azure ved oppstart
115
116 while True:
117     try:
118         if azure_clock % TFT_OPPDATER == 0:
119             # oppdater display
120             temp = float(aht20.temperature)
121             humid = float(aht20.relative_humidity)
122             print(f"AHT20 - Temperature: {temp:.1f} C")
123             print(f"AHT20 - Humidity: {humid:.1f} %")
124             temp_text.text = f"Temperatur: {temp:.1f} C"
125             humid_text.text = f"Fuktighet: {humid:.1f} %"
126
127         if azure_clock >= AZURE_OPPDATER:
128             # send melding til Azure
129             message = {"Temperature": aht20.temperature, "Humidity": aht20.relative_humidity}
130             az_device.send_telemetry(json.dumps(message))
131             tid = rtc.RTC().datetime
132             time_text.text= f"Data Sendt: {tid[3]:02d}:{tid[4]:02d} {tid[2]:02d}/{tid[1]:02d}/{tid[0]:02d}"
133             azure_clock = 1
134         else:
135             azure_clock += 1
136     except Exception as e:
137         print(f"Klarer ikke sende data til AZURE. Error:{e} - kort restarter om 30s")
138         time.sleep(30)
139         microcontroller.reset()
140
141     time.sleep(1)
142     print(f"{azure_clock=}")

```

13.6 Presentere data på IoT Central

Følgende må være på plass:

- Tilkoblingsnøkler er lagt inn i sw koden på device.
- Koden kjører på device og terminal vindu sier «Connected to Azure IoT Central!»

Gå tilbake til browser vindu for IoT Central og refresh siden.

Kortet ditt er nå tilkoblet og data fra device er mottas

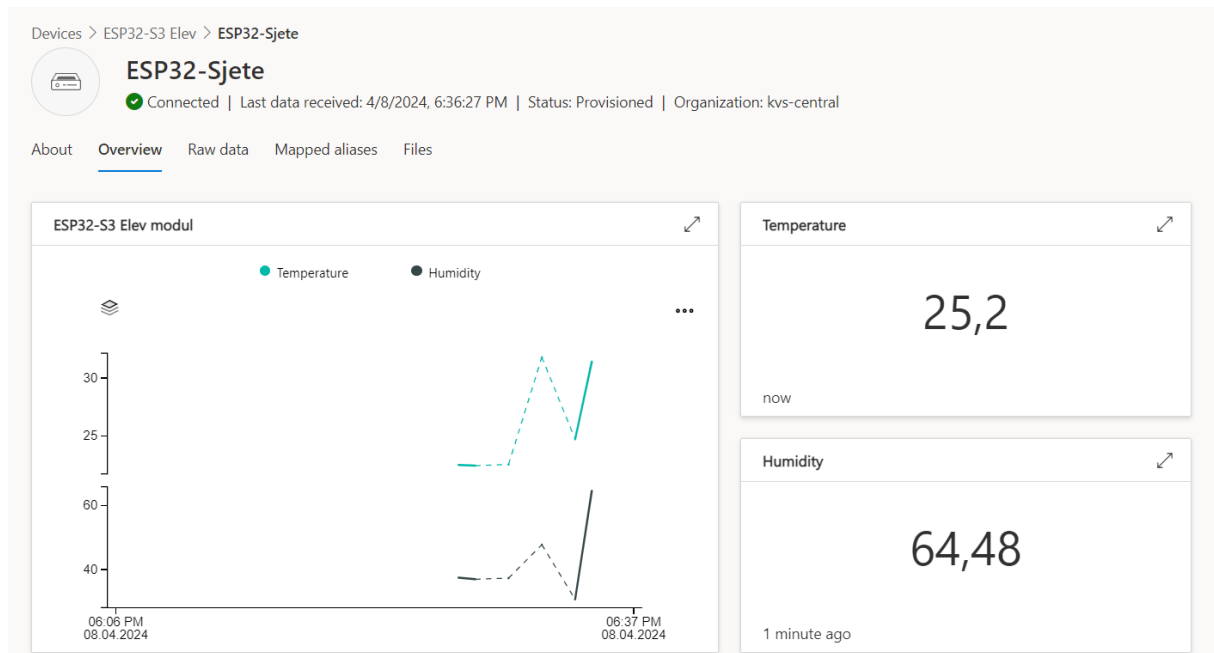
Devices > ESP32-S3 Elev > ESP32-Sjete

ESP32-Sjete
 Connected | Last data received: 4/8/2024, 6:34:45 PM | Status: Provisioned | Organization: kvs-central

About Overview **Raw data** Mapped aliases Files

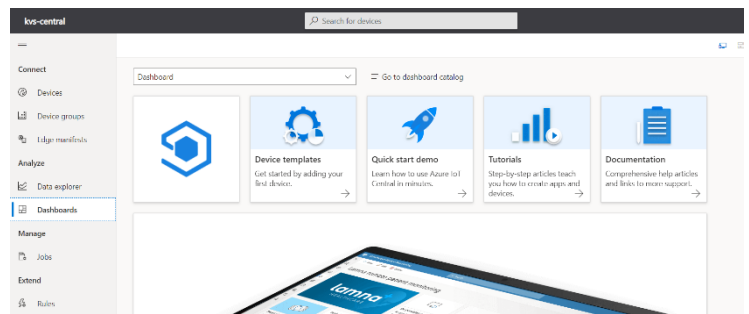
Timestamp ↓	Message type	Event creation time	Humidity	Temperature
> 4/8/2024, 6:34:45 PM	Telemetry		64.4847	31.4077
> 4/8/2024, 6:33:03 PM	Telemetry		30.7259	24.6931
> 4/8/2024, 6:31:22 PM	Telemetry		47.7311	31.7972
> 4/8/2024, 6:29:40 PM	Telemetry		37.3145	22.5046

Velg Overview for å se default dashboard for ditt kort



13.7 Lag eget dashboard

Velg «Dashboard» og «Go to dashboard catalog» og lag ditt eget



Velg +NEW for å lage et personlig dashboard

Velg gjenkjennbart navn

Velg Edit dashboard for å bygge innhold

Her kan du bygge opp ditt eget dashboard.

Velg «Device Group» ESP32-S3 Elev og velg ditt eget Device i listen.

Legg til telemetry data ved å trykke «+Capability».

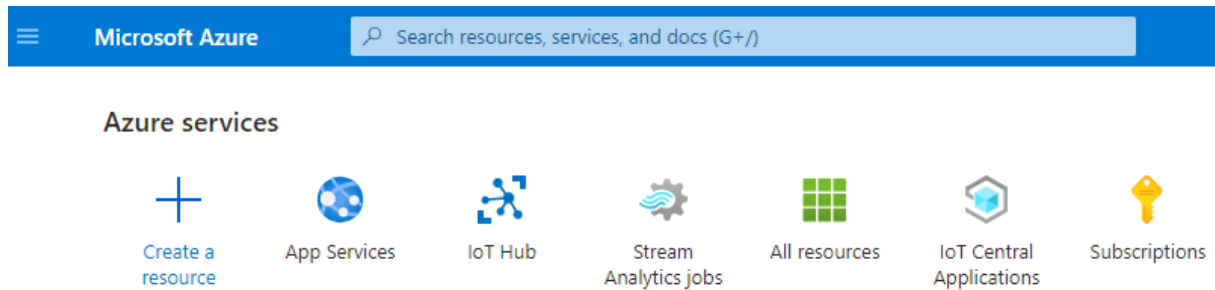
Velg Update og Save når du er ferdig

Ekspesimenter med ulike innstillinger slik at du får det som du vil.

14 Azure IoT Hub

14.1 Lag ny device på Azure IoT Hub

Logg på din Azure konto. Velg IoT Hub og gå inn på kvshub.



Velg Devices i menyen til venstre og velg «+ Add Device»

The left screenshot shows the 'kvshub | Devices' page. The 'Add Device' button is circled in red. Below it is a table of existing devices:

Device ID	Type	Status
testml	IoT Device	Enabled
test3	IoT Device	Enabled
test2	IoT Device	Enabled
test-oj	IoT Device	Enabled
testid	IoT Device	Enabled

The right screenshot shows the 'Create a device' form. The 'Device ID' field is filled with 'ESP32-S3-elev'. The 'Authentication type' is set to 'Symmetric key'. The 'Auto-generate keys' checkbox is checked. The 'Connect this device to an IoT hub' checkbox is also checked.

Skriv inn en Device ID og velg «Save»

14.2 Finn tilkoblingsnøkkel til IoT Hub Device

Velg din Device i listen, her ESP32-S3-elev

Nå får du opp nøkler du trenger i koden din for å koble til IoT Hub

The screenshot shows the Microsoft Azure IoT Hub console for a device named 'ESP32-S3-elev'. The interface includes a search bar at the top, navigation links, and a list of device properties. The 'Device ID' is 'ESP32-S3-elev'. The 'Primary key' and 'Secondary key' are both masked with dots. The 'Primary connection string' and 'Secondary connection string' are also masked. The 'Enable connection to IoT Hub' option is set to 'Enable'. The 'Parent device' is 'No parent device'. The 'Distributed Tracing (preview)' option is 'Not configured'.

Kopier «Device ID» og «Primary Connection String» til settings.toml filen din

```
13  
14 IOT_HUB_DEVICEID = "ESP32-S3-elev"  
15 IOT_HUB_PRIMARY_CONNECTION_STRING = "HostName=kvshub.azure  
16 |
```

14.3 Koble ESP32-S3 til IoT Hub

Nå skal vi koble kortet opp mot Azure IoT Hub.

Hent kode fra Vedlegg [her](#).

Bibliotek du trenger for å kjøre kode mot Azure IoT Hub. Noen har du allerede om du har kjørt Adafruit IO eller IoT Central.

Adafruit_azureiot

Adafruit_logging.mpy

Adafruit_binascii.mpy

```

18 # IoT Hub
19 import json
20 from adafruit_azureiot import IoTHubDevice

94
95 # sett opp Azure IoT Hub
96 esp = None
97 primkey = os.getenv("IOT_HUB_PRIMARY_CONNECTION_STRING")
98
99 az_device = IoTHubDevice(pool, esp, primkey)
100
101 print("Connecting to Azure IoT Hub...")
102 az_device.connect()
103 print("Connected to Azure IoT Hub!")
104

```

I loopen i hovedprogrammet legger du inn sending av melding til IoT Hub med et passende intervall. Det er kun navnet på funksjonen som sender til IoT Hub som er annerledes enn IoT Central

```

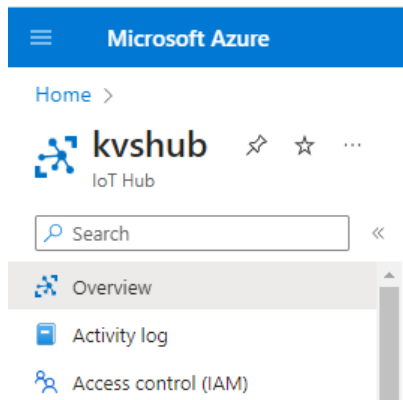
if azure_clock >= AZURE_OPPDATER:
    # send melding til Azure
    message = {"Temperature": aht20.temperature, "Humidity": aht20.relative_humidity}
    az_device.send_device_to_cloud_message(json.dumps(message))

```

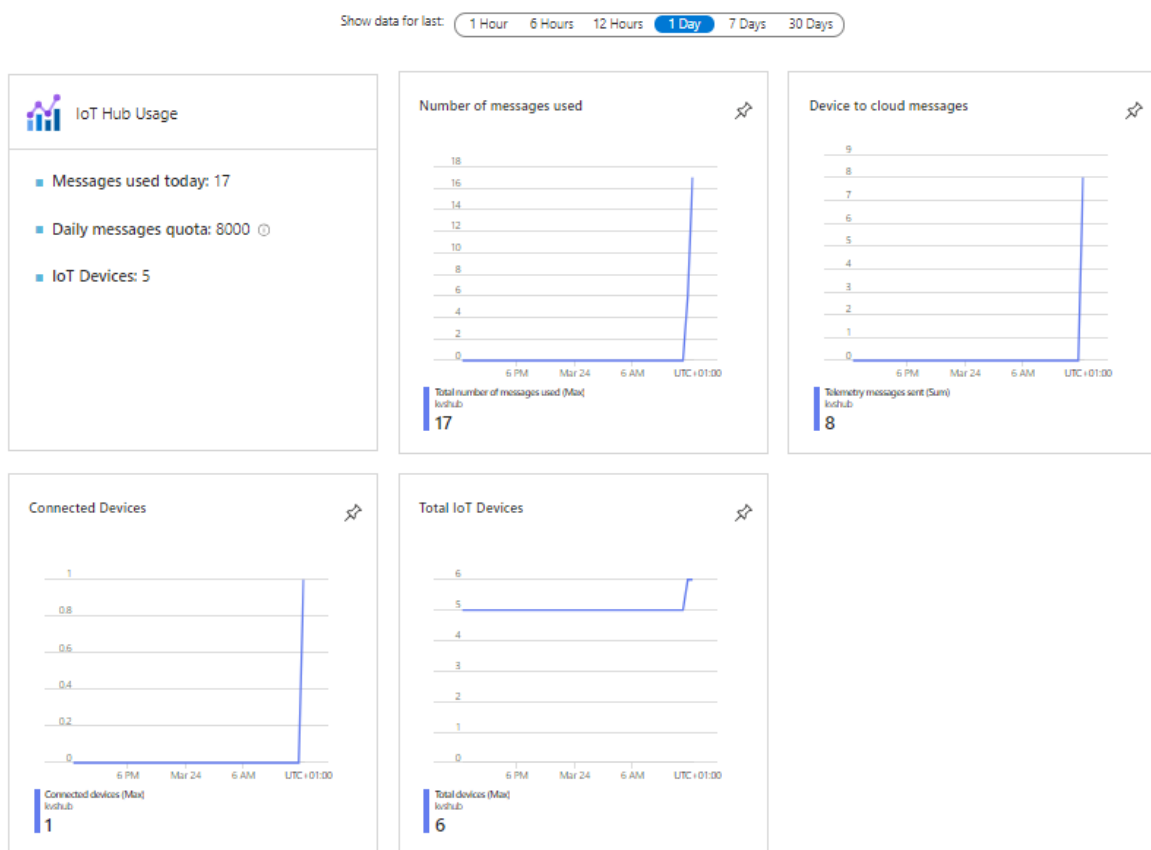
Resten av koden finner du under IoT Central [kapittel](#).

14.4 Azure IoT Hub data visning

Velg Overview for å se status på IoT Hub



Du kan ikke se selva data, men en oversikt



For å vise data må det settes opp andre tjenester

15 MQTT broker på Mosquitto server

Nå skal det settes opp egen server som broker. Dette er beskrevet i IT-kompendiet.

Her kommer hva som må på plass i koden på ESP32-S3.

Ta utgangspunkt i koden som kjørte på Adafruit IO.

Følgende mqtt bibliotek importeres – IO bibliotekene kan fjernes fra koden:

```

12 # mqtt
13 import ssl
14 import adafruit_minimqtt.adafruit_minimqtt as MQTT
15 import json

```

I settings.toml legger du inn broker adressen din

```

20 # Mosquitto oppsett - Bruk data fra din egen server
21 MOSQ_BROKER_IP_ADDR = "172.23.0.112"
22 MOSQ_USERNAME = "iotadmin"
23 MOSQ_PASSWORD = "iotadmin"

```

MQTT klienten settes opp slik:

```

105 mosq_broker_adr = os.getenv("MOSQ_BROKER_IP_ADDR")
106 mosq_username   = os.getenv("MOSQ_USERNAME")
107 mosq_password   = os.getenv("MOSQ_PASSWORD")
108 mosq_topic      = "test/sensor"
109
110 # Initialize a new MQTT Client object
111 mqtt_client = MQTT.MQTT(
112     broker = mosq_broker_adr,
113     username= mosq_username,
114     password= mosq_password,
115     socket_pool=pool,
116     port=1883,
117     is_ssl=False,|
118 )

```

Pr nå blir ikke username og passord brukt da broker er satt opp med åpen tilgang. Men dersom den blir satt opp med credentials kan det gjøres på denne måten.

Koble til server med

```
121     print("Kobler til Mosquitto IO...")
122     mqtt_client.connect()
```

Dette kan gjøres inne i en try-except om du ønsker feilmelding om pålogging mislykkes

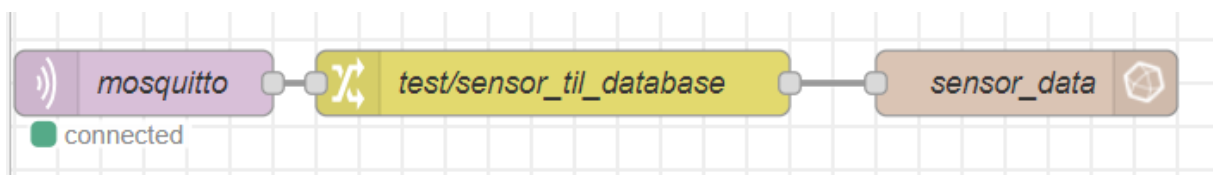
```
120 try:
121     print("Kobler til Mosquitto IO...")
122     mqtt_client.connect()
123 except Exception as e:
124     print("MQTT connection failed:", e)
125     # print feilen til skjermen og fortsett
126 else:
127     print("Koblet til Mosquitto IO...")
```

Når du er tilkoblet Mosquitto server oppdateres data ved at temperatur og fuktighets verdiene slås sammen og sendes som json

I hovedprogrammet hentes inn temperatur og fuktighet og sendes til Mosquitto

```
151     # oppdater Mosquitto server
152     sensor_data = json.dumps({"temperature":temp, "humidity":humid})
153     mqtt_client.publish(mosq_topic, sensor_data)
154     print("Sendt temperatur og fuktighet to Mosquitto")
```

Åpne Node-RED i Portainer og verifiser at EWSP32-S3 er tilkoblet

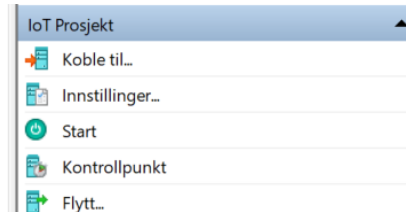


Følg oppsett i IOT-stack Kompendium for å sende data til database og presentere disse.

15.1 Oppstart av ESP32-S3 og MING-stack

1. Både ESP32-S3 og PC/Server må tilkobles samme nett; IoT-Eksamen_EL3. Pass på at PC ikke er tilkoblet kablet nett samtidig
2. Start opp ESP32-S3 kortet. Pass på at den kommer opp på nett.

3. Start opp virtuell server:
4. velg Koble til når server har startet opp



5. Logg på iotprosjekt:
bruker=iotadmin
passord=iotadmin
6. Legg merke til IP adressen til serveren din –
Her 172.23.0.112

```

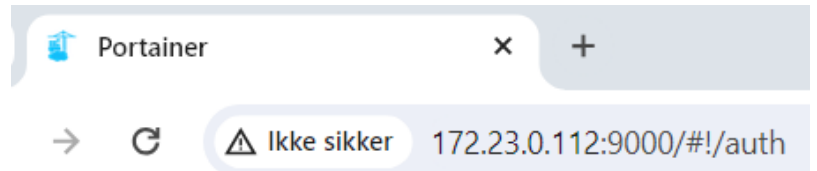
Password:
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 6.5.0-27-generic x86_64)

* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:       https://ubuntu.com/pro

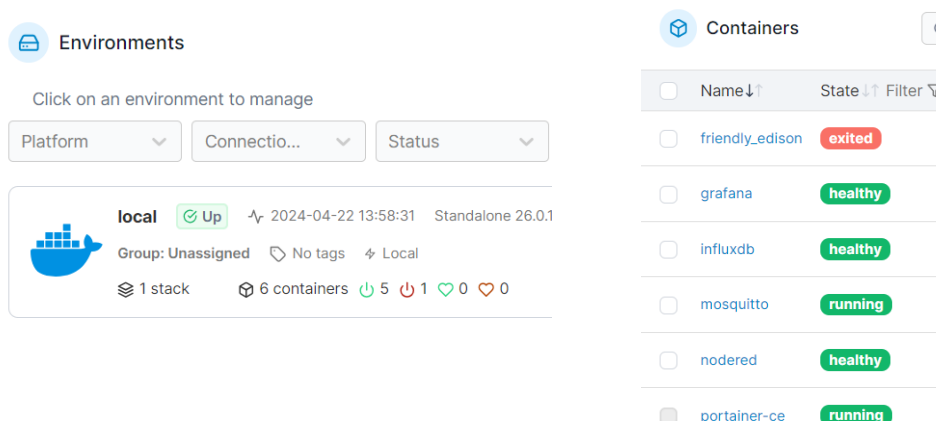
System information as of Mon Apr 22 11:53:28 AM UTC 2024

System load:                0.14892578125
Usage of /:                  74.1% of 9.75GB
Memory usage:               11%
Swap usage:                 0%
Processes:                  123
Users logged in:            0
IPv4 address for br-431b6d81b9b7: 172.18.0.1
IPv4 address for docker0:   172.17.0.1
IPv4 address for eth0:      172.23.0.112
    
```

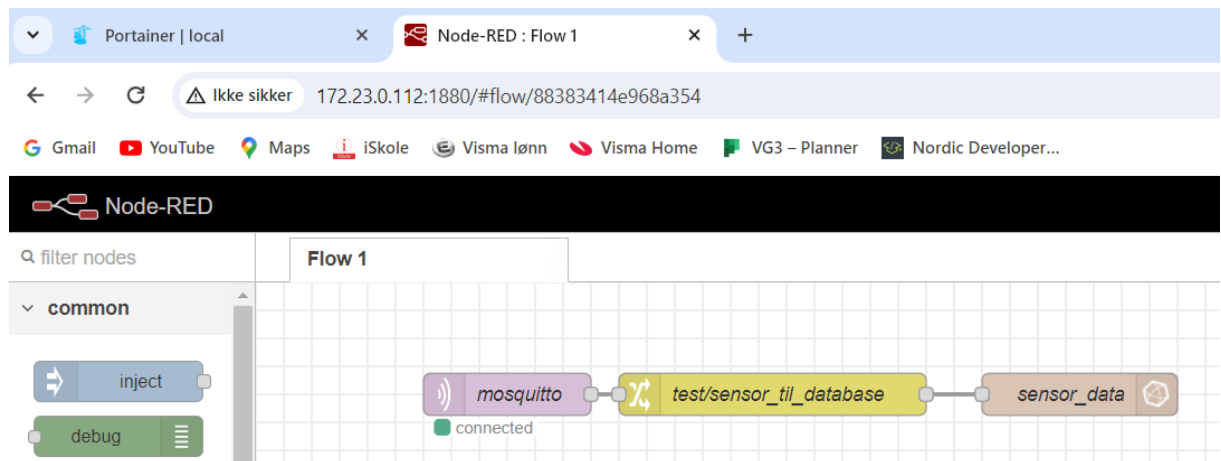
7. Start opp Protainer på port 9000.



8. Trykk på hvalen for å se at hele MING stack er oppe og kjører

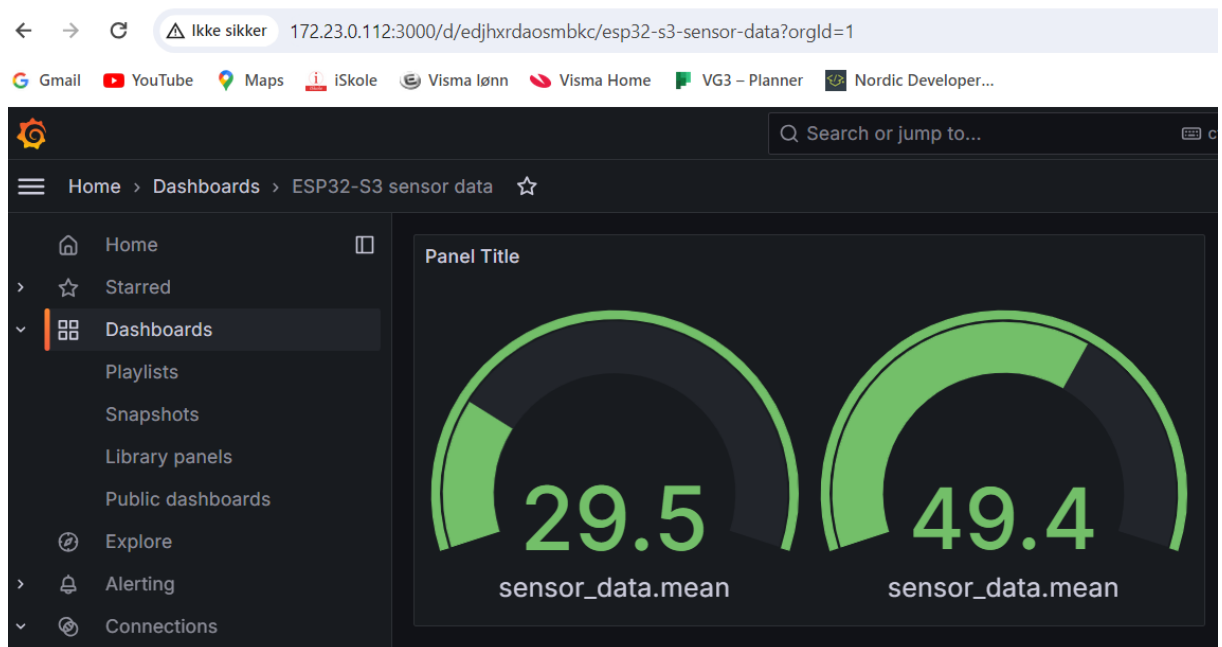


9. Start Node-RED på port 1880 – se at status på Mosquitto er Connected



The screenshot shows a web browser window with two tabs: 'Portainer | local' and 'Node-RED : Flow 1'. The address bar shows the URL '172.23.0.112:1880/#flow/88383414e968a354'. The Node-RED interface is visible, showing a flow named 'Flow 1' with three nodes: 'mosquitto' (status: connected), 'test/sensor_til_database', and 'sensor_data'.

10. Koble til Grafana på port 3000. Velg dashboard fra listen



The screenshot shows a web browser window with the URL '172.23.0.112:3000/d/edjhrdaosmbkc/esp32-s3-sensor-data?orgId=1'. The Grafana interface is visible, showing a dashboard titled 'ESP32-S3 sensor data'. The dashboard contains two gauge charts, both labeled 'sensor_data.mean', with values 29.5 and 49.4. The left sidebar shows the navigation menu with 'Dashboards' selected.

16 Batterimonitor

ESP32-S3 har en egen batterimonitor for måling av strøm på et tilkoblet LIPO batteri.

Les om denne [her](#) og prøv å få det til

17 OTA – Over The Air opplasting av kode

Kan vi få dette til på ESP32-S3?

18 Vedlegg

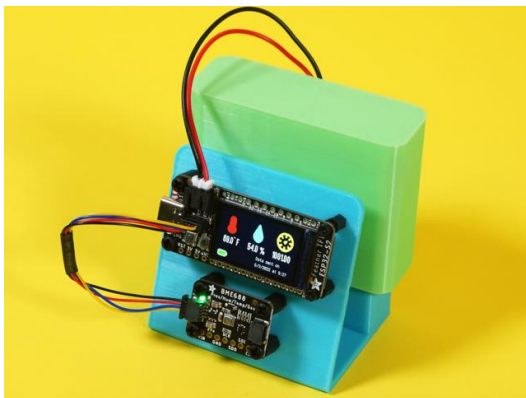
18.1 Basis kode for wifi, aht20, neopixel for CP9



wifi aht20 neopixel
cp9 tekstfil.txt

18.2 3d printing av holder for ESP32-S3, aht20 og batteri

Dersom du har lyst å prøve deg på å lage en holder for kortet ditt kan du følge linken [her](#)



18.3 Følg Adafruits guider

Adafruit har laget gode guider for kortene sine.

Følg linken under for å hente mer informasjon og for eksempler på hvordan du bruker funksjoner på kortet.

[Adafruit ESP32-S3 TFT Feather](#)

18.4 Håndtering av multiple SSID

Her er eksempel på hvordan man kan lage kode som har en tabell med wifi credentials slik at du kan flytte mellom ulike WiFi nett uten å endre koden.

Koden bruker en passordfil istedenfor settings.toml.

18.4.1 Passordfil

Det er anbefalt å lage en fil som inneholder private data adskilt fra koden din. Det er fordi da kan du dele koden din eller laste den opp på GitHub uten at dine data kommer på avveie.

I tidligere versjoner kalte Adafruit eksempel filen for secrets.py og i CP9 er den kalt settings.toml. Denne filen ligger tom på CIRCUITPY driven din.

```
# Eksempel på passordfil.py
# Passord nedenfor er ikke riktige
# Kan lagres på root eller lib katalog på HW

io_konto = {
    'timezone' : 'Europe/Oslo',
    'mqtt_broker_adr' : 'io.adafruit.com',
    'kvs_broker_adr' : '95.141.91.228',
    'aio_username' : 'minAIO-bruker',
    'aio_key' : 'aio_CJKPgzs1h0',
    'iot_central_idscope' : '0ne00',
    'iot_central_deviceid' : 'min-esp32s3',
    'iot_central_primarykey' : '+XxE9dqRmveUVDthc=',
    'iot_hub_deviceid' : 'mintest',
    'iot_hub_primarykey' : 'HostName=kvshub=',
}

wifi_passord = {
    'IoT-Eksamen' : 'k0bletil',
    'IoT-Eksamen-EL3A' : 'k0bletil',
    'Krokeide_VGS' : 'Krokeide2014',
}

def find_ssid_and_password(ssid_list):
    ssid_pw = []
    for x in wifi_passord:
        if x in ssid_list:
            ssid_pw.append(x)
            ssid_pw.append(wifi_passord[x])
            break
    return ssid_pw
```


18.4.2 Wifi_passord

Denne tabellen inneholder en tabell av type Dictionaries i Python. Det vil si at tabellen inneholder en liste av KEY – VALUE par.

Både KEY og VALUE er her en tekst og de er adskilt med «:»

Par skiller med «,»

Eks

```
SSID = IoT-Eksamen      'IoT-Eksamen'      : 'k0bletil',
Passord = k0bletil
```

Legg til SSID og passords for de wifi sonene du vil bruke kortet

18.4.3 find_ssid_and_password()

Funksjonen find_ssid_and_password() får inn en liste over tilgjengelige SSID og går gjennom wifi_passord tabellen for å finne tilhørende passord og returnerer dette.

Dette gjør at du slipper å endre i koden når du bytter wifi nett.

For å bruke denne må du lagre alle tilgjengelige SSIS og det gjør du når du skanner for tilgjengelige nett i programmet ditt.

Så kaller du på denne måten:

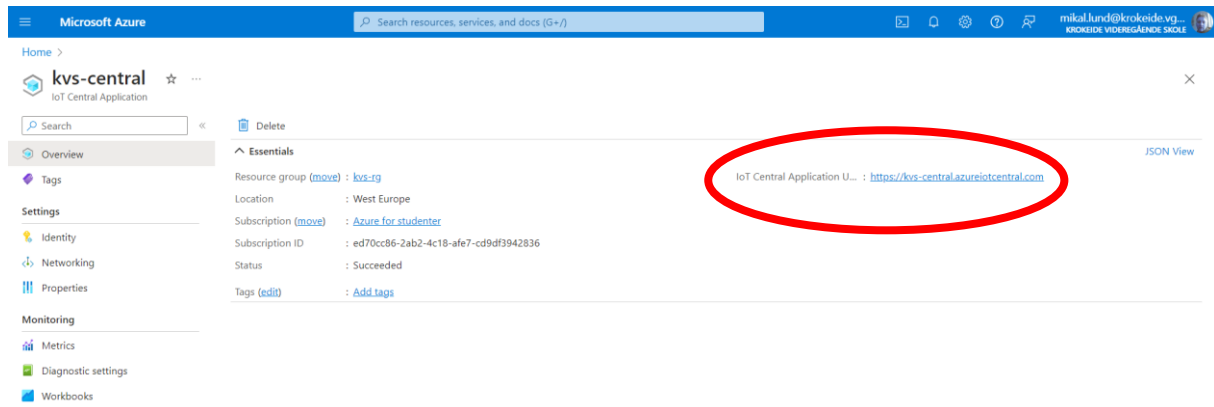
```
# finn ssid og passord fra liste i passordfil
wifi_cred = pwd.find_ssid_and_password(avail_ssid)
if wifi_cred:
    try:
        print(f"Kobler til nettverk {wifi_cred[0]}")
        wifi.radio.connect(wifi_cred[0], wifi_cred[1])
        print(f"Koblet til {wifi_cred[0]}")
    except Exception as e:
        print("Failed to connect to WiFi. Error:", e, "\nBoard will hard reset in 30 seconds.")
        time.sleep(30)
        microcontroller.reset()
```

18.4.4 lo_konto

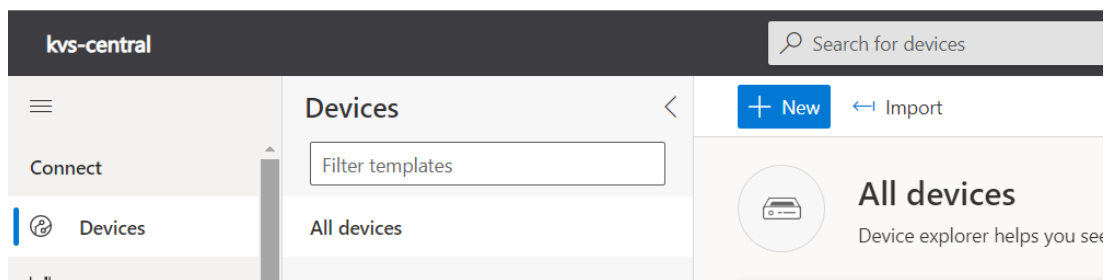
Denne tabellen inneholder dine private kontodetaljer til Adafruit IO og Azure som du ikke vil dele med andre. Her kan du også legge inn adressen til din egen mqtt broker.

19 Logg på Azure for lærer-bruker

Logg på din Azure konto. Velg IoT Central og gå inn på kvs-central. Åpne konfigurasjonsiden ved å gå til liken merket av under.



Velg +NEW



Skriv inn et unikt navn og create

Create a new device

To create a new device, select a device template, a name, and a unique ID. [Learn more](#)

Device name * ⓘ

Device ID * ⓘ

Organization * ⓘ

Device template *

The screenshot shows the Azure IoT Central interface for a device named 'ESP32-S3 Elev'. The 'Connect' button in the top navigation bar is highlighted with a red circle. The device details show it is registered and belongs to the organization 'kvs-central'. Below the details is a table with columns for 'Timestamp', 'Message type', and 'Event creation time', which currently contains no data.

Velg «Connect» for å få opp nøkler til kortet ditt

Device connection groups ×

ID scope ⓘ

📄

Device ID ⓘ

📄

Choose the connection type for this device. You can change this later if you need to.

Authentication type

Key QR code

Shared Access Signatures (SAS) use security tokens and keys to connect to IoT Central. Use the SAS keys from the default enrollment group shown below to register your device. [Learn more](#)

Primary key ⓘ

📄

Kopier disse til din settings.toml på denne måten

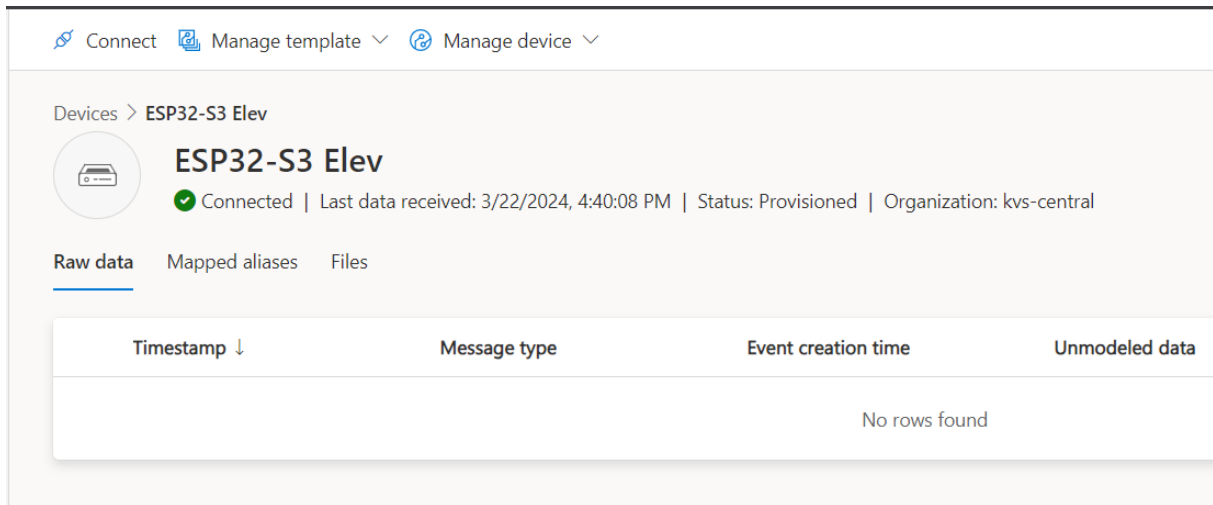
```

9 # Azure IoT Central private nøkler
10 IOT_CENTRAL_IDSCOPE = "0ne00C0A9E5"
11 IOT_CENTRAL_DEVICEID = "22zo3rva6p5"
12 IOT_CENTRAL_PRIMARYKEY = "Objy9cQlyzbTCIPYgrD+MUC+8X4G0DdU1+ttZ5MAPLY="

```

19.1 Presentere data på IoT Central

Vi ser av bildet at kortet er koblet til Azure, men ingen data vises



For å sette opp visning av data velger vi Manage template og Auto-Create Template

Data som vi sender opp kalles for telemetry. Legg til begge to og vis hvilken verditype som forventes

Data preview

Review your device's data and make any desired changes in the window below. When finished, click **Create template** Her er

Once created, you can edit or add to your template anytime.

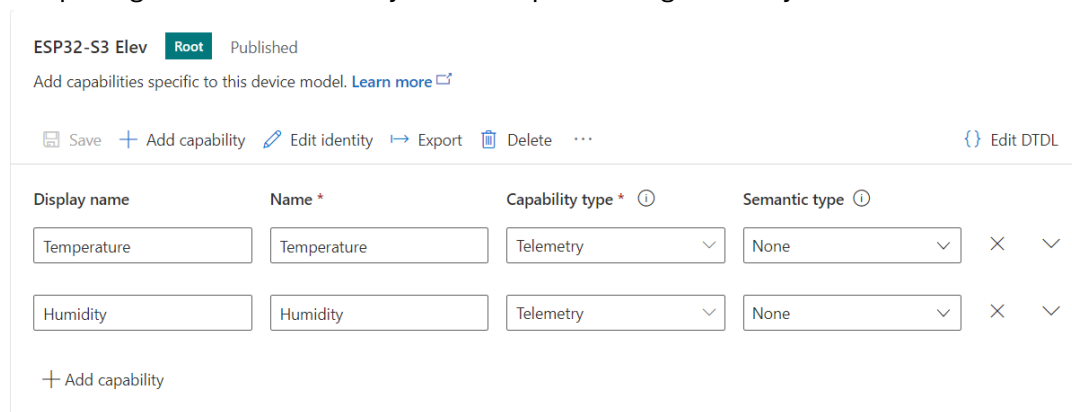
Contents

```

1 {
2   "telemetries": {},
3   "properties": {}
4 }
```

Sett opp Contents slik: «telemetries»: { «Temperature» : 20, «Humidity» : 50 }

Template generert for telemetry data: Temperature og Humidity



Trykk på hake lengst til høyre på temperatur linjen.

Her kan du fylle inn mer informasjon til visningen din.

Velg Save når du er ferdig

ESP32-S3 Elev Root Published

Add capabilities specific to this device model. [Learn more](#)

Save + Add capability Edit identity Export Delete ... Edit DTDL

Display name	Name *	Capability type * ⓘ	Semantic type ⓘ
Temperature	Temperature	Telemetry	None

Schema * Define Color

Double

Min value ⓘ Max value ⓘ Decimal places ⓘ

0 40 1

Unit Display unit Comment Description

Degree celsius C

Nå blir status til Draft – så velg Publish og Publish igjen

This device template is published. Editing published capabilities may cause breaking changes in dashboards, jobs, rules, or data exports. [Learn more](#)

Version Manage test device Publish Rename Delete

Template created and published successfully. You can create views or modify capabilities anytime below.

Device templates > ESP32-S3 Elev > Model > ESP32-S3 Elev

ESP32-S3 Elev Application updated: 15 minutes ago Interfaces published: 15 minutes ago

Model

- ESP32-S3 Elev
- Raw data
- Views
- Overview
- About

ESP32-S3 Elev Root Draft

Add capabilities specific to this device model. [Learn more](#)

Save + Add capability Edit identity Export Delete ... Edit DTDL

Display name	Name *	Capability type * ⓘ	Semantic type ⓘ
Temperature	Temperature	Telemetry	None
Humidity	Humidity	Telemetry	None

+ Add capability

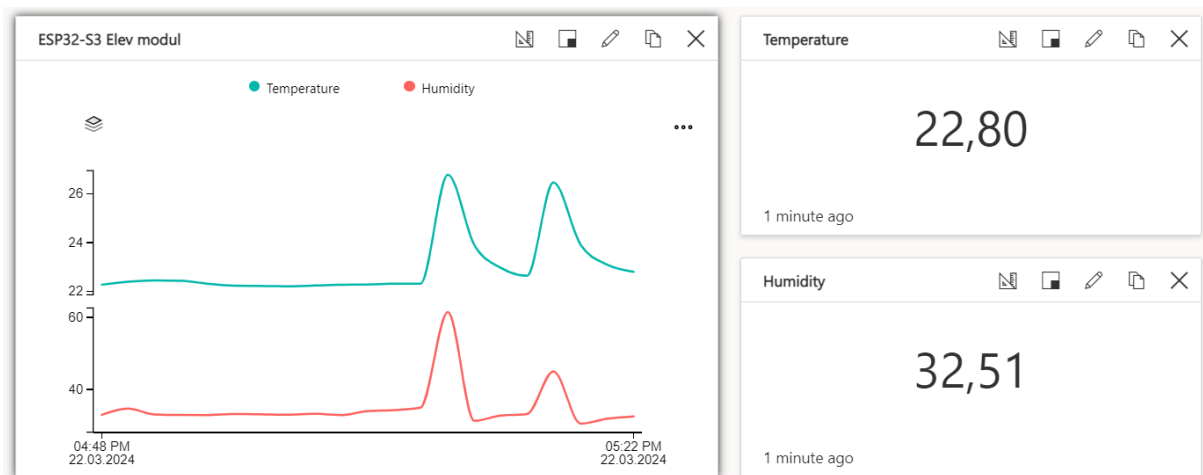
Velg Raw data for å se alle meldingen som er kommet inn

Her er eksempel på Raw data

Timestamp ↓	Message type	Event creation time	Humidity	Temperature
> 3/22/2024, 5:30:57 PM	Telemetry		33.0608	22.6446
> 3/22/2024, 5:29:15 PM	Telemetry		32.4265	22.623
> 3/22/2024, 5:27:33 PM	Telemetry		32.6798	22.6612
> 3/22/2024, 5:25:51 PM	Telemetry		32.6881	22.6671
> 3/22/2024, 5:24:10 PM	Telemetry		32.6014	22.6648
> 3/22/2024, 5:22:28 PM	Telemetry		32.5103	22.8046

Velg Views – Overview for å se data grafisk

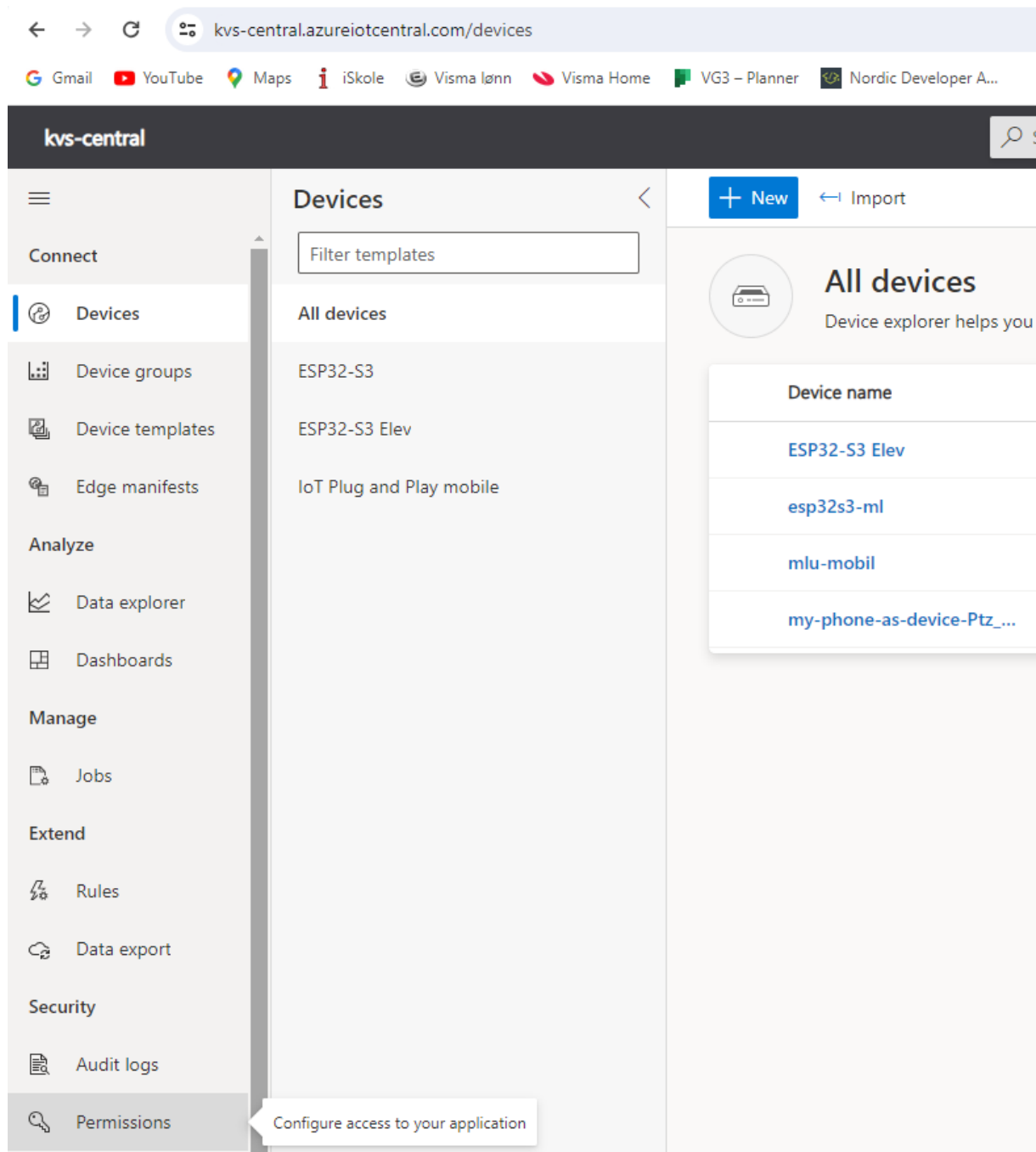
Det er mange muligheter for å konfigurere visning av data og her er et eksempel.



20 Legg til Elevgruppe i kvs-central

Lærere har rolle som «App Administrator» og/eller «App Builder» og kan da gi elever tilgang som «App Operator». Da kan elevene lage sitt eget device og presentere data for denne.

For administrering av brukere gå inn på kvs-central.

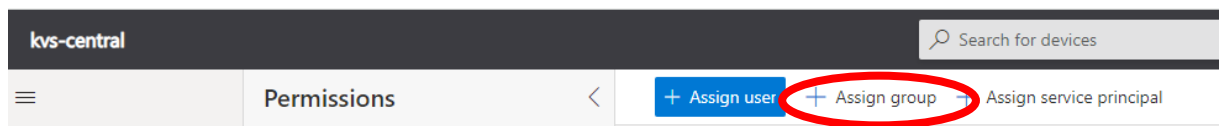
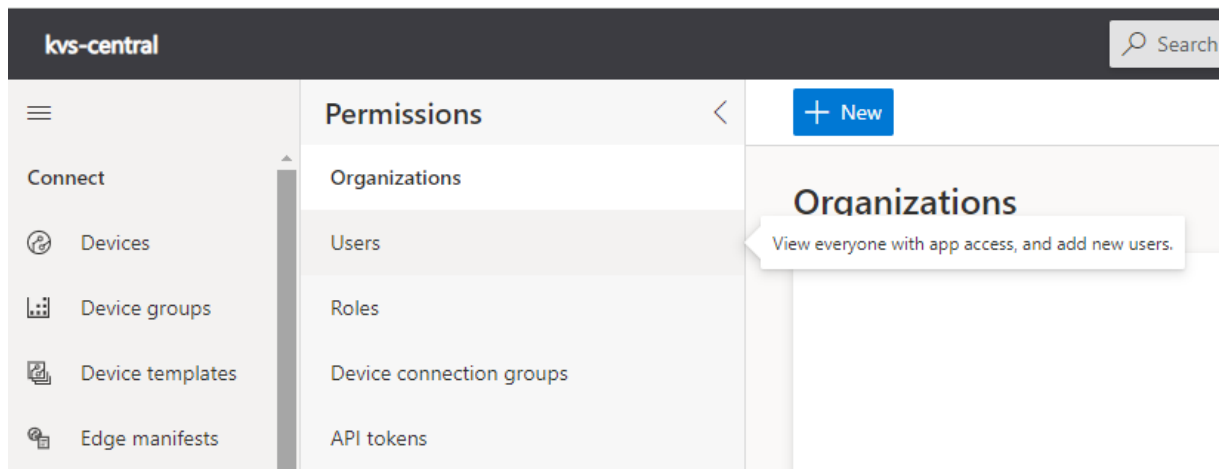


The screenshot shows the 'kvs-central' interface in a web browser. The address bar displays 'kvs-central.azureiotcentral.com/devices'. The browser's address bar and tabs are visible at the top, including links to Gmail, YouTube, Maps, iSkole, Visma lønn, Visma Home, VG3 - Planner, and Nordic Developer A... The main interface has a dark header with the 'kvs-central' logo and a search icon. A left sidebar contains navigation options: Connect, Devices (selected), Device groups, Device templates, Edge manifests, Analyze (Data explorer, Dashboards), Manage (Jobs), Extend (Rules, Data export), and Security (Audit logs, Permissions). The main content area is titled 'Devices' and includes a 'Filter templates' input field. Below this, a list of devices is shown under the heading 'All devices': ESP32-S3, ESP32-S3 Elev, and IoT Plug and Play mobile. On the right, there are '+ New' and 'Import' buttons, and a section titled 'All devices' with a sub-heading 'Device explorer helps you'. Below this is a table with the following data:

Device name
ESP32-S3 Elev
esp32s3-ml
mlu-mobil
my-phone-as-device-Ptz_...

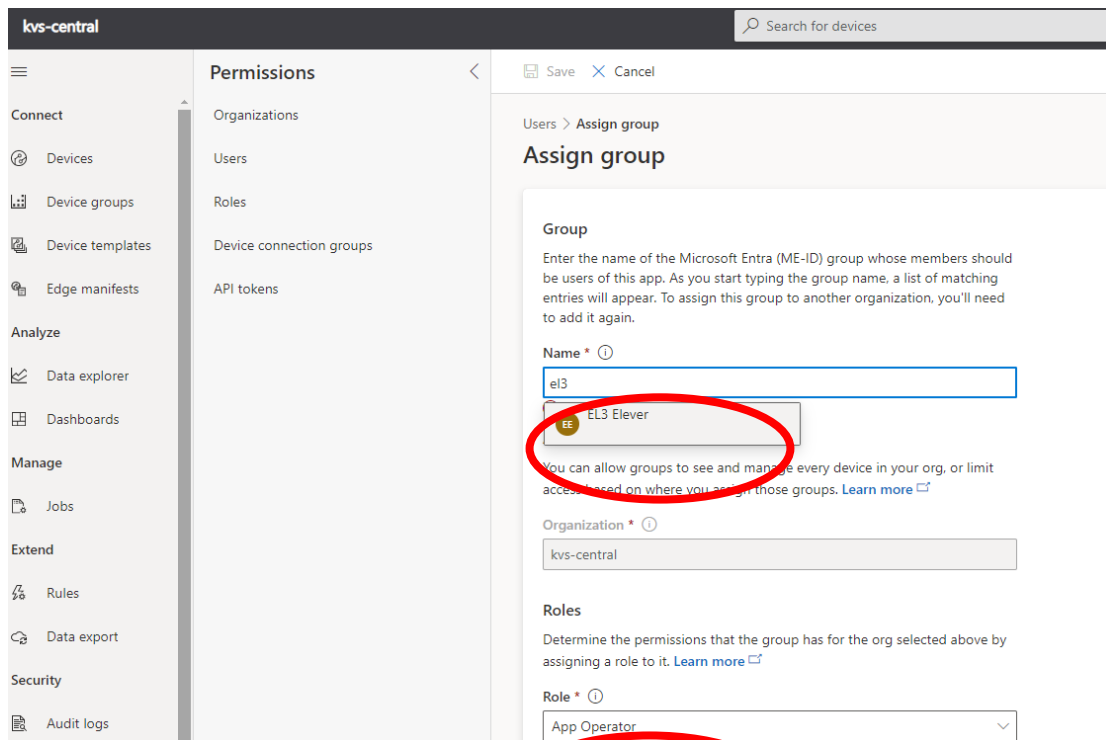
A tooltip at the bottom of the sidebar says 'Configure access to your application'.

Velg Permissions og Users



Velg «Assign group» og velg gruppen: «EL3 Elever»

Gi elevgruppen rollen: APP Operator



Trykk SAVE