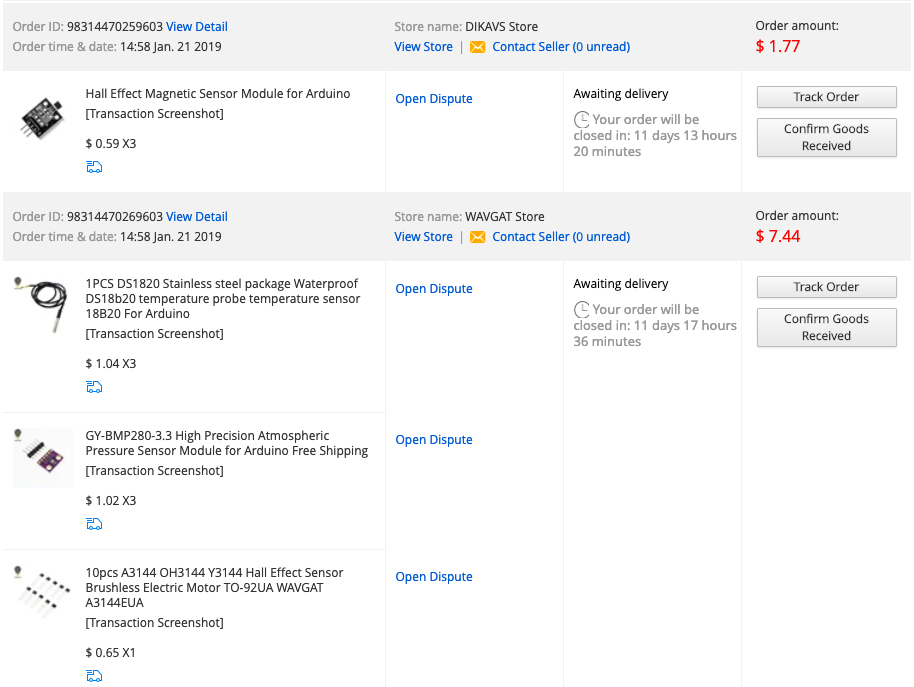
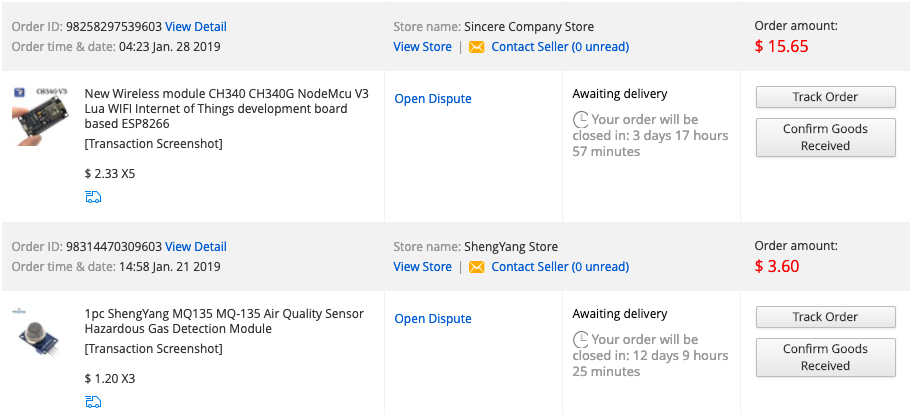
# Vedlegg

### Skjermbilde 1



### Skjermbilde 2



### Kode 3

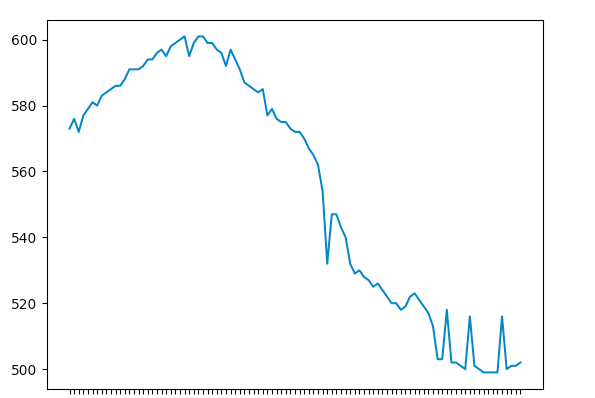
##### luftkvalitetssender.pde

/\*  
 \* Denne koden leser av PPM verdier for luftforurensning fra en MQ135 gasssensor,   
 \* viser dette på en LCD skjerm og sender informasjonen over UDP  
 \*/  
// --- Oppsett til WiFi ---   
#include <WiFi.h>  
#include "esp\_wpa2.h" //wpa2 library for å koble til bedriftsnettverk  
#include <WiFiUdp.h>  
#define EAP\_IDENTITY "\*\*\*\*" // Brukernavnet ditt med 3 bokstaver fra for og etternavn og et 2-sifret tall  
#define EAP\_PASSWORD "\*\*\*\*" //Passordet  
  
// --- LCD skjerm konfigurasjon ---  
//#include <LiquidCrystal\_I2C.h>  
//LiquidCrystal\_I2C lcd(0x27,16,2);  
// --- Luftkvalitssensor ---  
int sensor\_avlesning;  
int gammelVerdi = 0;  
  
  
// --- WiFi nettverk brukernavn og passord ---  
const char \* ssid = "Akademiet";  
const char \* networkUsnm = "\*\*\*\*"; //Samme brukernavn  
const char \* networkPswd = "\*\*\*\*"; //Samme passord  
  
// --- IP addresse til å sende UDP data til ---  
const char \* udpAddress = "10.25.9.178";  
const int udpPort = 16969;  
  
// --- Teste forbindelse ---   
boolean connected = false;  
  
// --- Type til UDP library ---  
WiFiUDP udp;  
  
int counter = 0;  
  
// --- Definsere pins til sensorer  
//int lcdPin = 39;  
int sensorPin = 36;  
  
void setup(){  
 // Initilize hardware serial:  
 Serial.begin(9600);  
   
 //Connect to the WiFi network  
 //, networkUsnm, networkPswd  
 connectToWiFi(ssid);  
 pinMode(sensorPin,INPUT);  
 //pinMode(lcdPin, OUTPUT);  
 pinMode(LED\_BUILTIN,OUTPUT);  
}  
  
void loop(){  
 //only send data when connected   
 sensor\_avlesning = digitalRead(sensorPin);  
 Serial.print(sensor\_avlesning);  
 if(connected){  
 //Send a packet  
 /\* udp.beginPacket(udpAddress,udpPort);  
 udp.printf("Verdi:",avlesning); \*/  
 while (digitalRead(sensorPin) != gammelVerdi){  
 udp.beginPacket(udpAddress,udpPort);  
 udp.printf("%d",sensor\_avlesning);  
 udp.endPacket();  
 Serial.println(sensor\_avlesning);  
 gammelVerdi = digitalRead(sensorPin);  
   
 }  
 /\*  
 lcd.init();  
 lcd.backlight();  
 lcd.setCursor(0,0);  
 lcd.print("ArQ=");  
 lcd.print(sensor\_avlesning,DEC);  
 lcd.print(" PPM");  
 lcd.println(" ");   
 lcd.print(" ");  
 delay(100); \*/   
 }  
 //Wait for 1 second  
}  
//, const char \* networkUsnm, const char networkPswd  
void connectToWiFi(const char \* ssid){  
 Serial.println("Connecting to WiFi network: " + String(ssid));  
  
 // delete old config  
 WiFi.disconnect(true);  
 //register event handler  
 WiFi.onEvent(WiFiEvent);  
   
 //Initiate connection  
 WiFi.mode(WIFI\_STA); //init wifi mode  
 esp\_wifi\_sta\_wpa2\_ent\_set\_identity((uint8\_t \*)EAP\_IDENTITY, strlen(EAP\_IDENTITY)); //provide identity  
 esp\_wifi\_sta\_wpa2\_ent\_set\_username((uint8\_t \*)EAP\_IDENTITY, strlen(EAP\_IDENTITY)); //provide username  
 esp\_wifi\_sta\_wpa2\_ent\_set\_password((uint8\_t \*)EAP\_PASSWORD, strlen(EAP\_PASSWORD)); //provide password  
 esp\_wpa2\_config\_t config = WPA2\_CONFIG\_INIT\_DEFAULT(); //set config settings to default  
 esp\_wifi\_sta\_wpa2\_ent\_enable(&config); //set config settings to enable function  
 WiFi.begin(ssid);  
 while (WiFi.status() != WL\_CONNECTED) {  
 delay(500);  
 Serial.print(".");  
 counter++;  
 if(counter>=60){ //after 30 seconds timeout - reset board  
 ESP.restart();  
 }  
 }  
 Serial.println("");  
 Serial.println("WiFi connected");  
 Serial.println("IP address set: ");   
 Serial.println(WiFi.localIP()); //print LAN IP  
 }  
  
//wifi event handler  
void WiFiEvent(WiFiEvent\_t event){  
 switch(event) {  
 case SYSTEM\_EVENT\_STA\_GOT\_IP:  
 //When connected set   
 Serial.print("WiFi connected! IP address: ");  
 Serial.println(WiFi.localIP());   
 //initializes the UDP state  
 //This initializes the transfer buffer  
 udp.begin(WiFi.localIP(),udpPort);  
 connected = true;  
 break;  
 case SYSTEM\_EVENT\_STA\_DISCONNECTED:  
 Serial.println("WiFi lost connection");  
 connected = false;  
 break;  
 }  
}

##### luftkvalitetslogger.py

import socket  
import sqlite3  
import time  
from datetime import datetime  
import matplotlib.pyplot as plt  
import os  
  
filnummer = 0  
if os.path.exists('luftkvalitetslogg.db') or os.path.exists('luftkvalitetslogg'+str(filnummer)+'.db'):  
 filnummer += 1  
 conn = sqlite3.connect('luftkvalitetslogg'+str(filnummer)+'.db')  
 time.sleep(1)  
 c = conn.cursor()  
 if os.stat('luftkvalitetslogg' + str(filnummer) + '.db').st\_size == 0:  
 c.execute('CREATE TABLE grupperom(ID INTEGER PRIMARY KEY, time TEXT, airquality INTEGER, irsensor INTEGER)')  
else:  
 conn = sqlite3.connect('luftkvalitetslogg.db')  
 c = conn.cursor()  
  
UDP\_IP = "10.25.10.63"  
UDP\_PORT = 16969  
  
sock = socket.socket(socket.AF\_INET, # Internet  
 socket.SOCK\_DGRAM) # UDP  
sock.bind((UDP\_IP, UDP\_PORT))  
c.execute('SELECT MAX(ID) as Nyeste FROM grupperom')  
  
id\_test = c.fetchone()[0]  
if type(id\_test) != int:  
 id = 0  
else:  
 id = id\_test  
  
log = []  
counter = 0  
  
while counter < 100:  
 data, addr = sock.recvfrom(1024) # buffer size is 1024 bytes  
 data = int(data)  
 tid = str(datetime.now())  
 id += 1  
 info = [(id, tid, data, 1)]  
 c.executemany('''INSERT INTO grupperom VALUES(?,?,?,?)''', info)  
 c.execute('SELECT \* FROM grupperom ORDER BY id DESC LIMIT 1')  
 conn.commit()  
 print(c.fetchone())  
 time.sleep(0.1)  
 counter += 1  
  
for row in c.execute('SELECT \* FROM grupperom ORDER BY id DESC LIMIT 100'):  
 log += c.fetchall()  
x\_verdi = [x[1] for x in log]  
y\_verdi = [x[2] for x in log]  
plt.plot(x\_verdi,y\_verdi)  
plt.show()

### Skjermbilde 4



### Kode 5

##### rom.ini

[postgresql]  
host=localhost  
database=rom  
user=fersch  
password=Akademiet

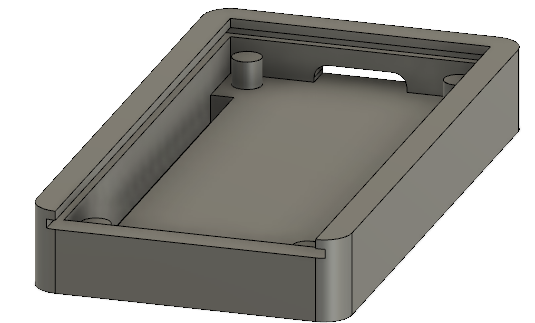
##### config.py

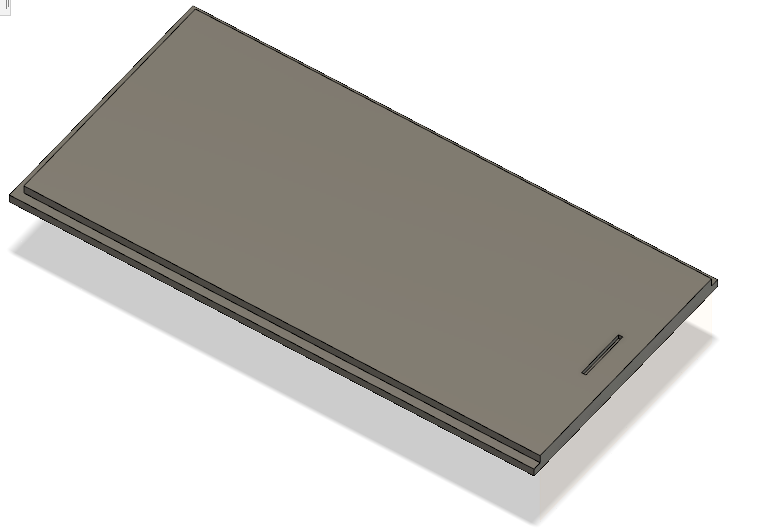
from configparser import ConfigParser  
  
def config(filename='rom.ini', section='postgresql'):  
 # create a parser  
 parser = ConfigParser()  
 # read config file  
 parser.read(filename)  
  
 # get section, default to postgresql  
 db = {}  
 if parser.has\_section(section):  
 params = parser.items(section)  
 for param in params:  
 db[param[0]] = param[1]  
 else:  
 raise Exception('Section {0} not found in the {1} file'.format(section, filename))  
  
 return db

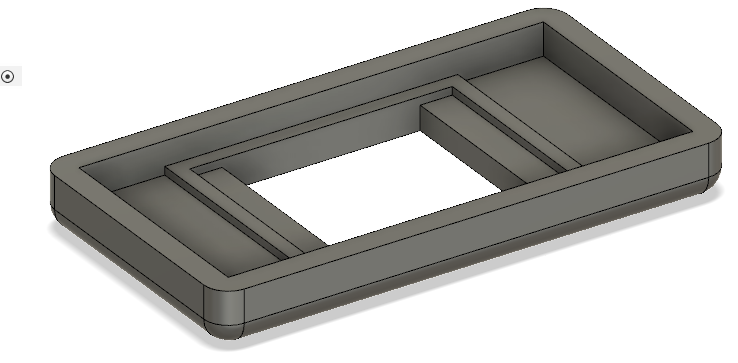
##### postgresinsert.py

import psycopg2  
from psycopg2 import sql  
from config import config  
import socket  
from datetime import datetime as date  
  
UDP\_IP = "10.25.10.163"  
UDP\_PORT = 16969  
sock = socket.socket(socket.AF\_INET, # Internet  
 socket.SOCK\_DGRAM) # UDP  
sock.bind((UDP\_IP, UDP\_PORT))  
  
data, addr = sock.recvfrom(1024) # buffer size is 1024 bytes  
data = str(data)  
data = data[2:len(data)-1]  
data = data.split(",")  
rom\_id = data[0]  
rom\_id = str(rom\_id)  
romnavn = "rom"+rom\_id  
print(romnavn)  
  
def insert(romnummer, co2ppm,tempc,humidity,irsensor):  
 """ Connect to the PostgreSQL database server """  
 conn = None  
 try:  
 # read connection parameters  
 params = config()  
 # connect to the PostgreSQL server  
 print('Connecting to the PostgreSQL database...')  
 conn = psycopg2.connect(\*\*params)  
  
 # create a cursor  
 cur = conn.cursor()  
  
 # execute a statement  
 print('PostgreSQL database version:')  
 cur.execute('SELECT version()')  
 # display the PostgreSQL database server version  
 db\_version = cur.fetchone()  
 print(db\_version)  
  
 # close the communication with the PostgreSQL  
 except (Exception, psycopg2.DatabaseError) as error:  
 print(error)  
 finally:  
 if conn is not None:  
 cur.execute(sql.SQL('''SELECT MAX(ID) FROM {}''').format(sql.Identifier(romnummer)))  
 id\_test = cur.fetchone()[0]  
 if type(id\_test) != int:  
 id = 1  
 else:  
 id = id\_test+1  
 print('Database connection is opened.')  
  
 datetime = str(date.now())  
 info = [(id,datetime,co2ppm,tempc,humidity,irsensor)]  
 cur.executemany(sql.SQL('''INSERT INTO {} VALUES(%s, %s, %s, %s, %s, %s)''').format(sql.Identifier(romnummer)), info)  
 cur.execute(sql.SQL('''SELECT \* FROM rom1 ORDER BY id DESC LIMIT 1'''))  
 print(type(cur.fetchone()))  
 conn.commit()  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 insert(romnavn, data[1],data[2],data[3],data[4])

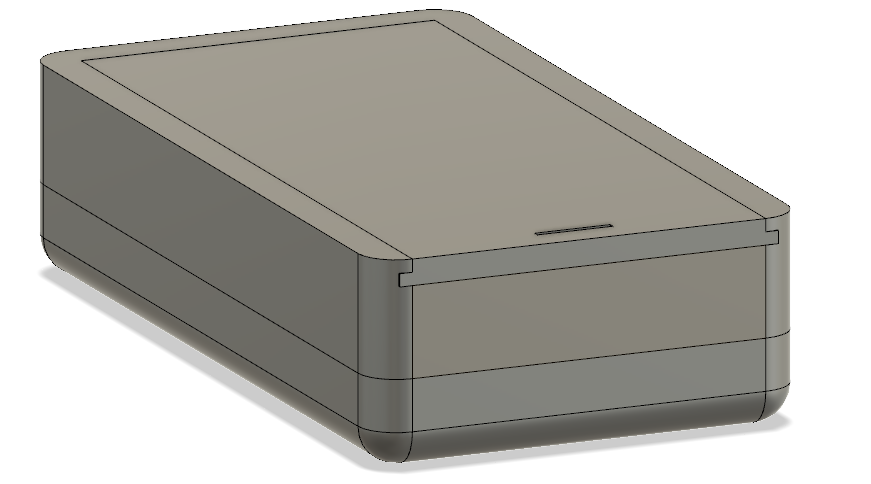
### Illustrasjon 6-8

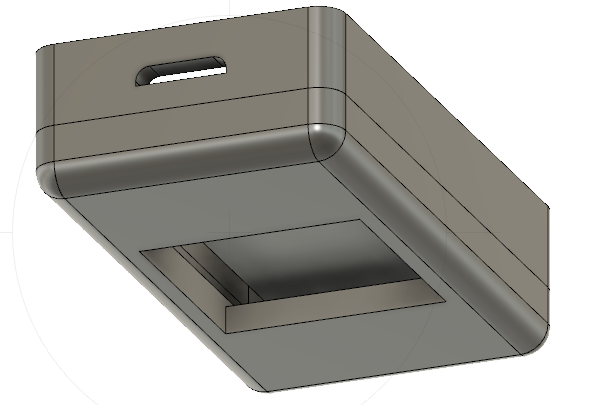


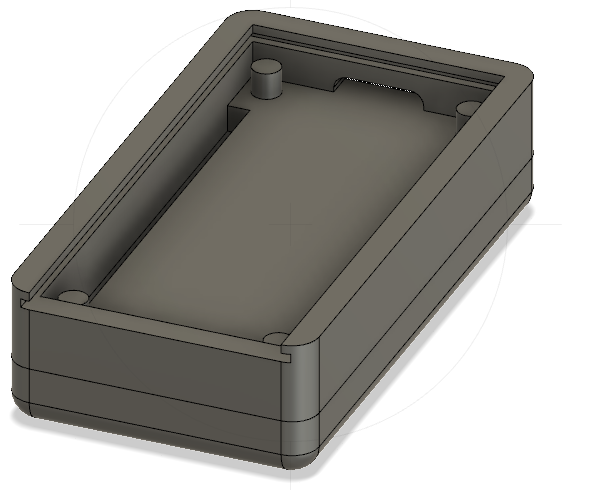




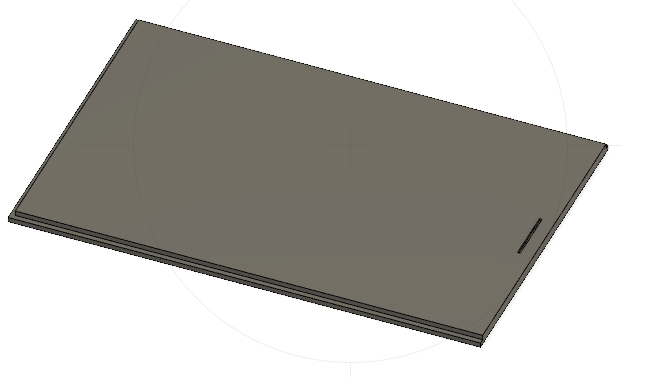
### Illustrasjon 9-11

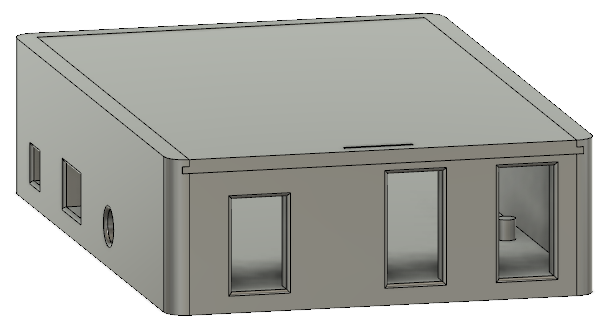


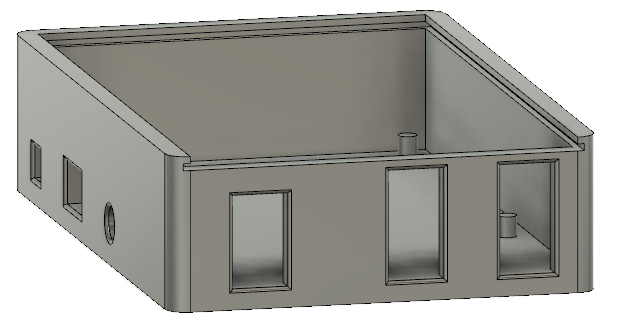




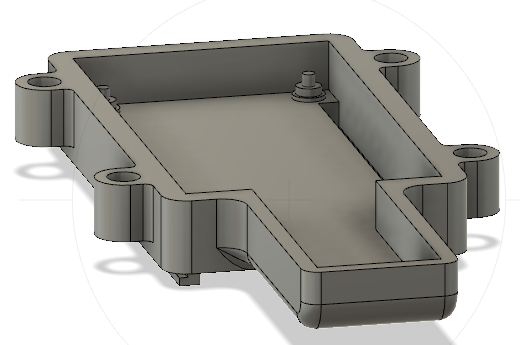
### Illustrasjon 12-14

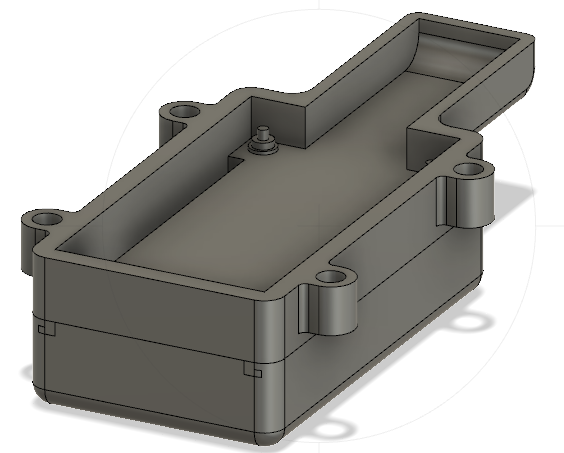


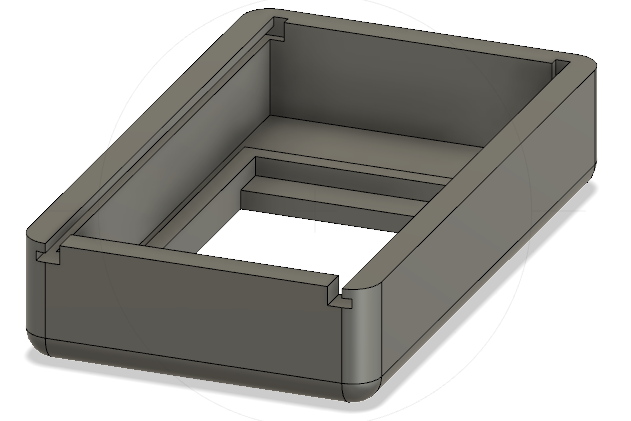




### Illustrasjon 15-17





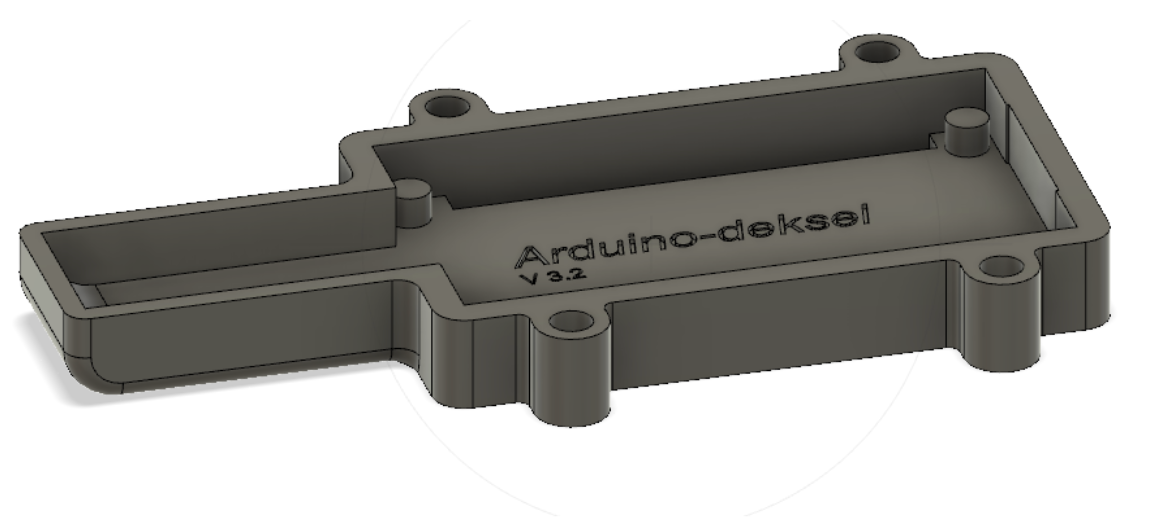


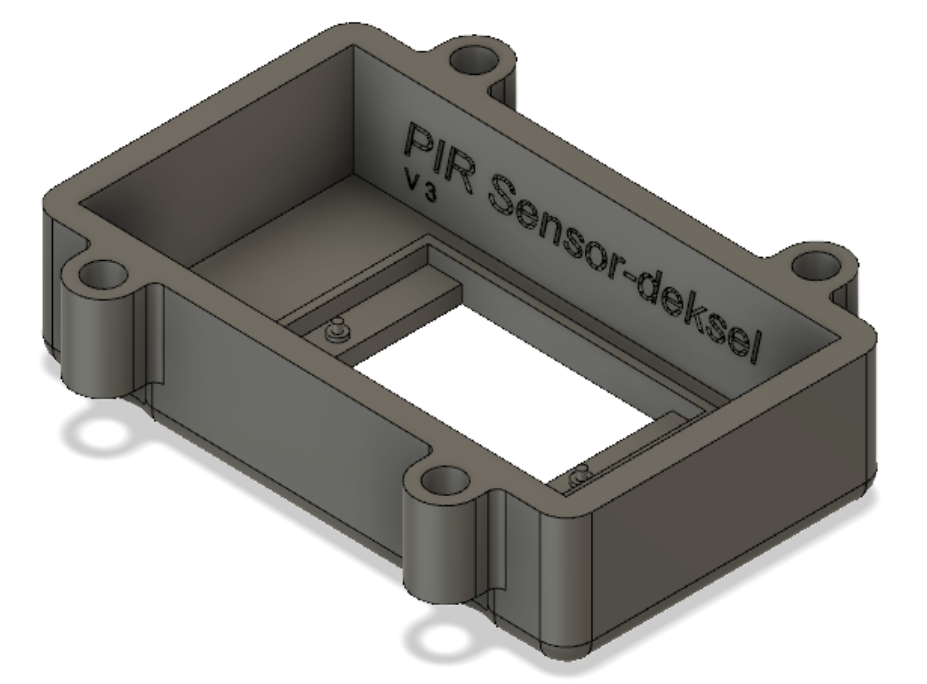
### Bilde 18-19





### Illustrasjoner 20-21

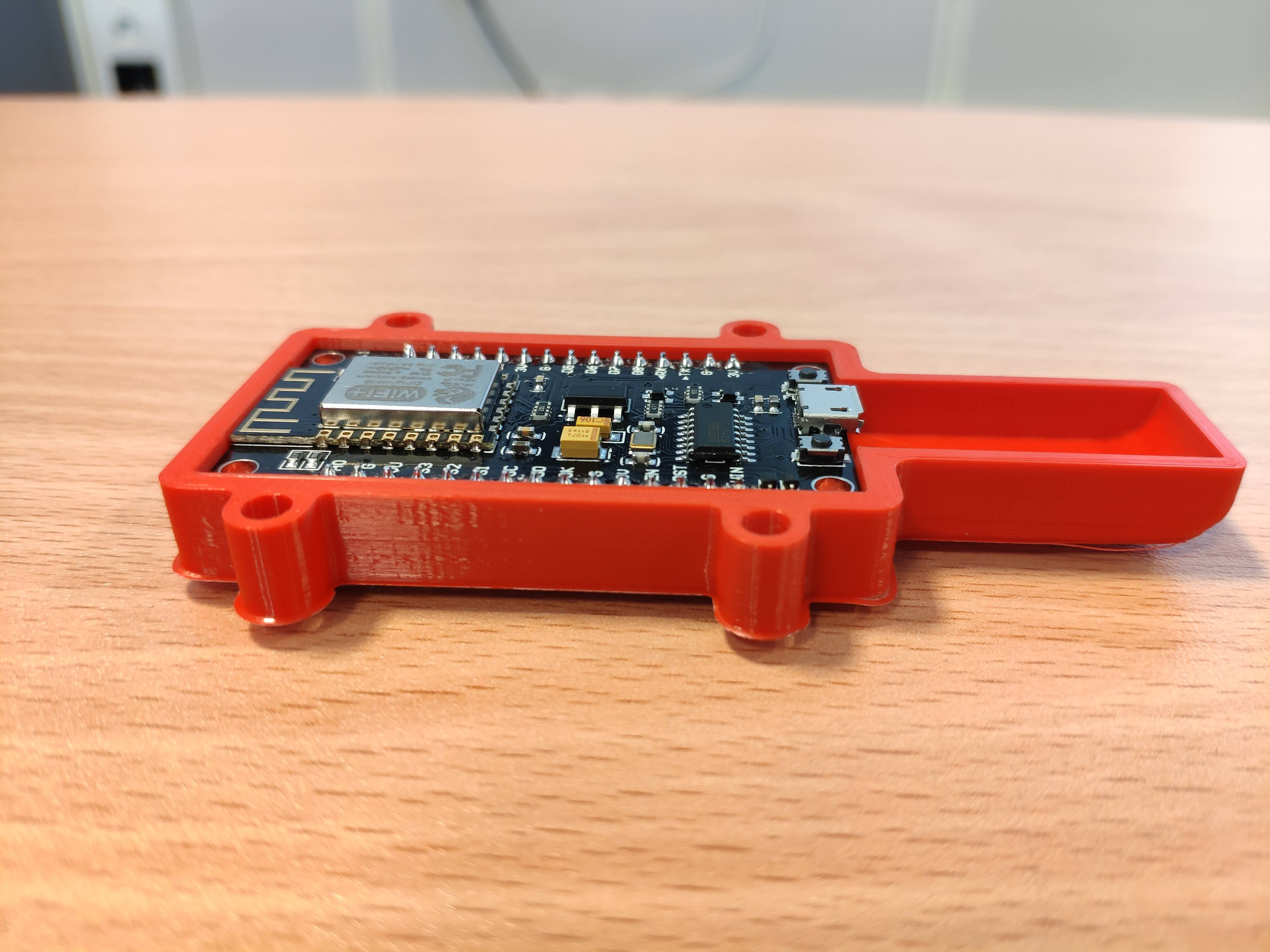




### Bilde 22-24



"ovenfra med støtte"



"fra siden"



"i profil"

### Kode 25

touple = ("217","700", "0"); #får en touple med verdiene (romnummer, co2ppm, pir-sensor)  
romnummer = touple[0]  
co2ppm = int(touple[1])  
ir = touple[2]  
  
if ir == "1":  
 status = "Opptatt"  
elif ir == "0":  
 status = "Ledig"  
  
  
if 0 <= co2ppm and co2ppm < 300: #disse verdiene må vi endre på  
 luftkval = "Dårlig"  
elif 300 <= co2ppm and co2ppm < 600: #samme her  
 luftkval = "Middels"  
elif 600 <= co2ppm and co2ppm <= 900: #samme her ( ta disse kommentarne vekk når de er fikset)  
 luftkval = "Bra"  
  
parser = "html5lib" #bruker html5lib som parser  
  
from bs4 import BeautifulSoup  
with open("grupperom.html") as fp:  
 soup = BeautifulSoup(fp,parser) #åpner filen  
  
idluft = (romnummer+"luft") #lager en id til luftkvalitet  
idstatus = (romnummer+"status") #lager en id til statusen  
  
#oppdatere luftkvaliteten til rommet:  
tag = soup.find(id=idluft) #finner taggen som stemmer med luft-id-en  
tag.string.replace\_with(luftkval) #erstatter strengen til taggen med ny verdi  
  
#oppdatere statusen til rommet:  
tag = soup.find(id=idstatus) #finner taggen som passer med status-id-en  
tag.string.replace\_with(status) #erstatter strengen til taggen med ny verdi  
  
with open("grupperom.html","w") as outf:  
 outf.write(str(soup)) #overskriver det gamle dokumentet med det nye lagd i python