# VEDLEGG

## Skjermbilde 1

Order ID: 98314470259603 View Detail
Order time & date: 14:58 Jan. 21 2019

Store name: DIKAVS Store
View Store | Contact Seller (0 unread)

Order amount:
$ 1.77

Hall Effect Magnetic Sensor Module for Arduino [Transaction Screenshot]

$ 0.59 X3

Open Dispute

Awaiting delivery

Your order will be closed in: 11 days 13 hours 20 minutes

Track Order

Confirm Goods Received

Order ID: 98314470269603 View Detail
Order time & date: 14:58 Jan. 21 2019

Store name: WAVGAT Store
View Store | Contact Seller (0 unread)

Order amount:
$ 7.44

1PCS DS1820 Stainless steel package Waterproof DS18b20 temperature probe temperature sensor 18B20 For Arduino [Transaction Screenshot]

$ 1.04 X3

Open Dispute

Awaiting delivery

Your order will be closed in: 11 days 17 hours 36 minutes

Track Order

Confirm Goods Received

GY-BMP280-3.3 High Precision Atmospheric Pressure Sensor Module for Arduino Free Shipping [Transaction Screenshot]

$ 1.02 X3

Open Dispute

10pcs A3144 OH3144 Y3144 Hall Effect Sensor Brushless Electric Motor TO-92UA WAVGAT A3144EUA [Transaction Screenshot]

$ 0.65 X1

Open Dispute

## Skjermbilde 2

Order ID: 98258297539603 View Detail
Order time & date: 04:23 Jan. 28 2019

Store name: Sincere Company Store
View Store | Contact Seller (0 unread)

Order amount:
$ 15.65

New Wireless module CH340 CH340G NodeMcu V3 Lua WIFI Internet of Things development board based ESP8266 [Transaction Screenshot]

$ 2.33 X5

Open Dispute

Awaiting delivery

Your order will be closed in: 3 days 17 hours 57 minutes

Track Order

Confirm Goods Received

Order ID: 98314470309603 View Detail
Order time & date: 14:58 Jan. 21 2019

Store name: ShengYang Store
View Store | Contact Seller (0 unread)

Order amount:
$ 3.60

1pc ShengYang MQ135 MQ-135 Air Quality Sensor Hazardous Gas Detection Module [Transaction Screenshot]

$ 1.20 X3

Open Dispute

Awaiting delivery

Your order will be closed in: 12 days 9 hours 25 minutes

Track Order

Confirm Goods Received

# 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);

  }
```

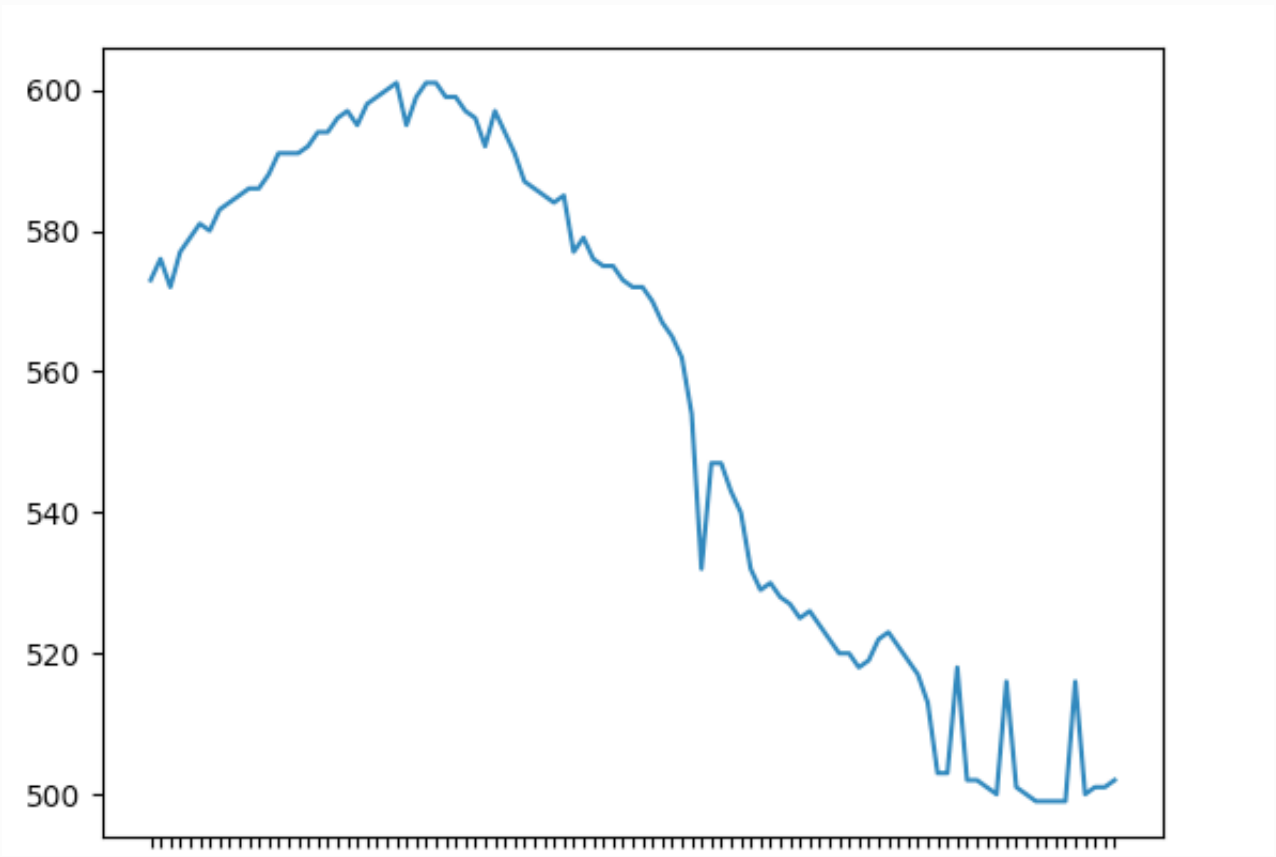
```
  /*
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0,0);
  lcd.print("ArQ=");
  lcd.print(sensor_avlesning,DEC);
  lcd.print(" PPM");
  lcd.println("       ");
  lcd.print("  ");
  delay(100);        */
  }
  //Wait for 1 second
}
//, const char * networkUsnm, const char networkPswd
void connectToWiFi(const 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

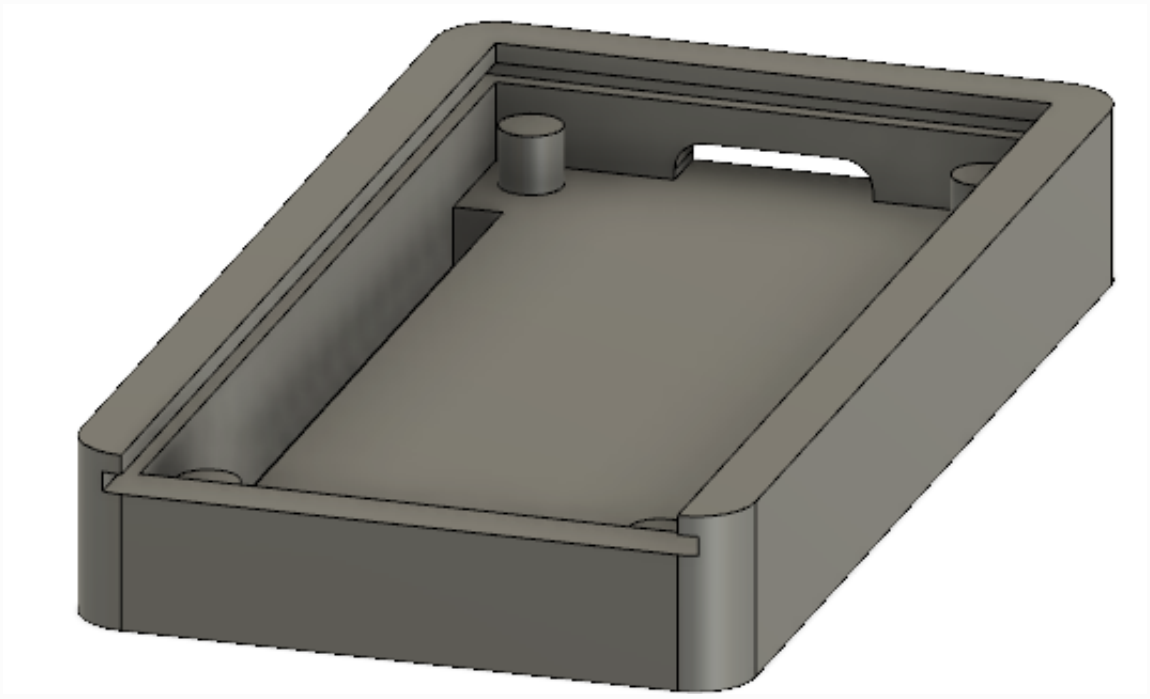
```
from configparser import ConfigParser

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
```

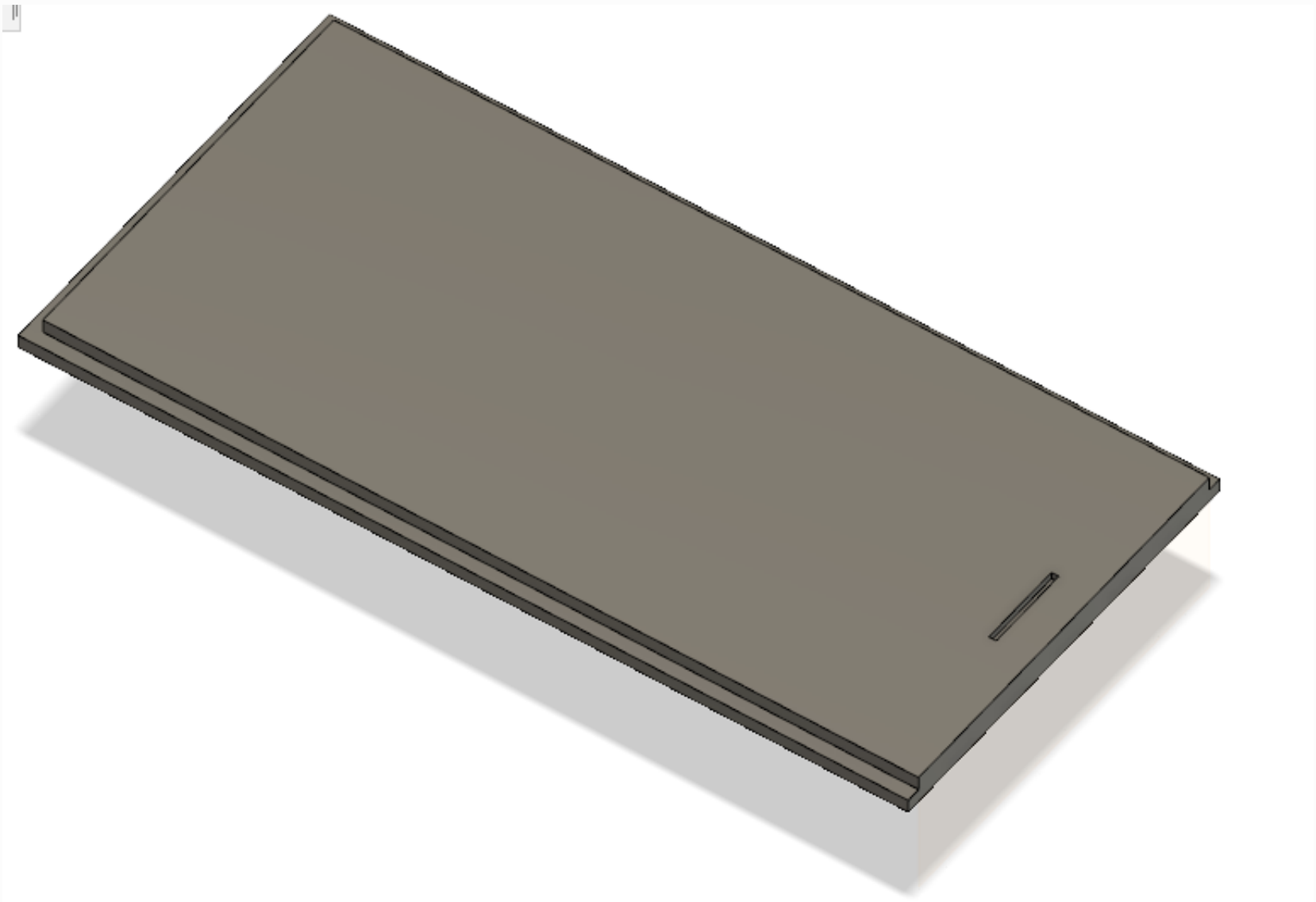
postgresinsert.py

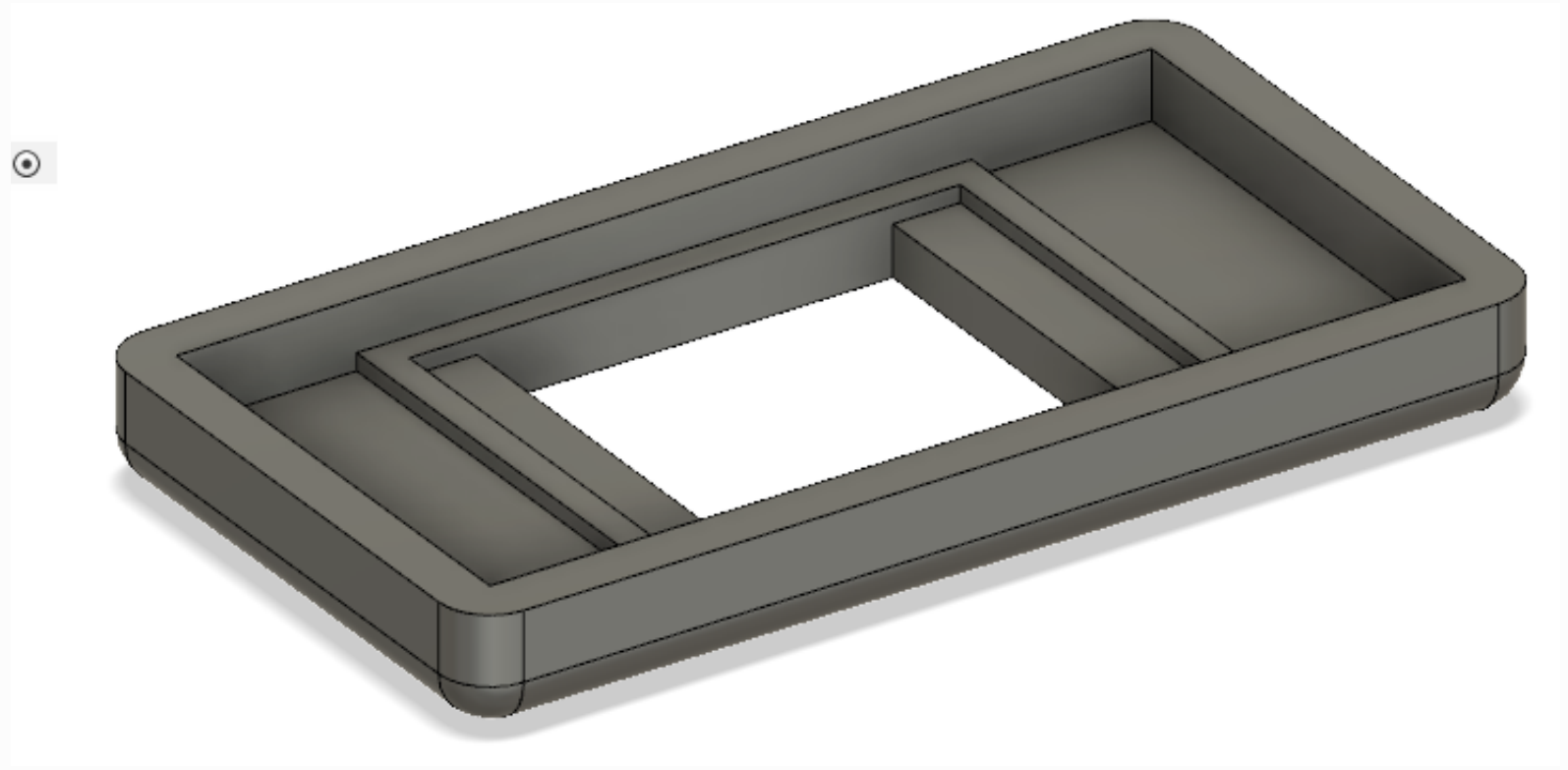
```
import psycopg2
from psycopg2 import sql
from config import config
import socket
from datetime import datetime as date

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])
```

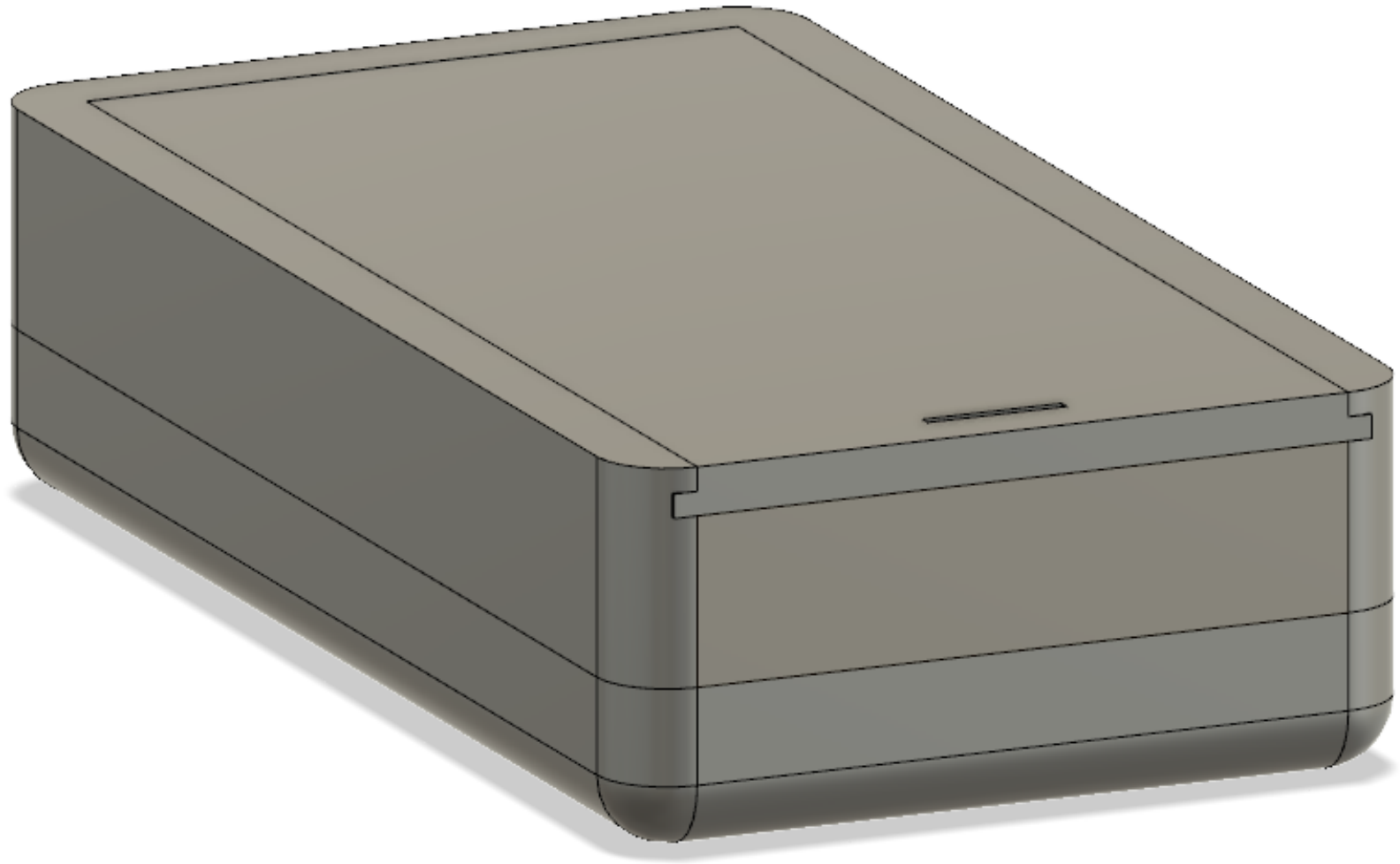
Illustrasjon 6-8

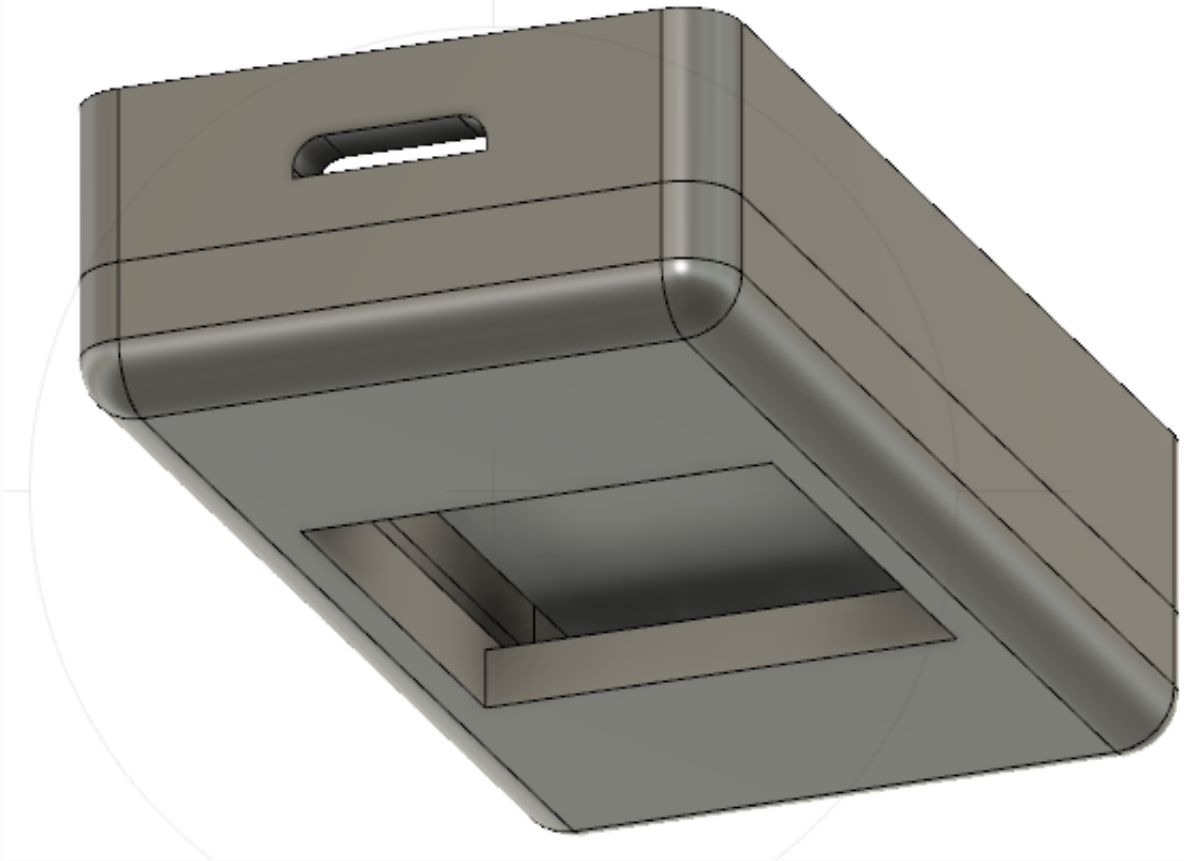


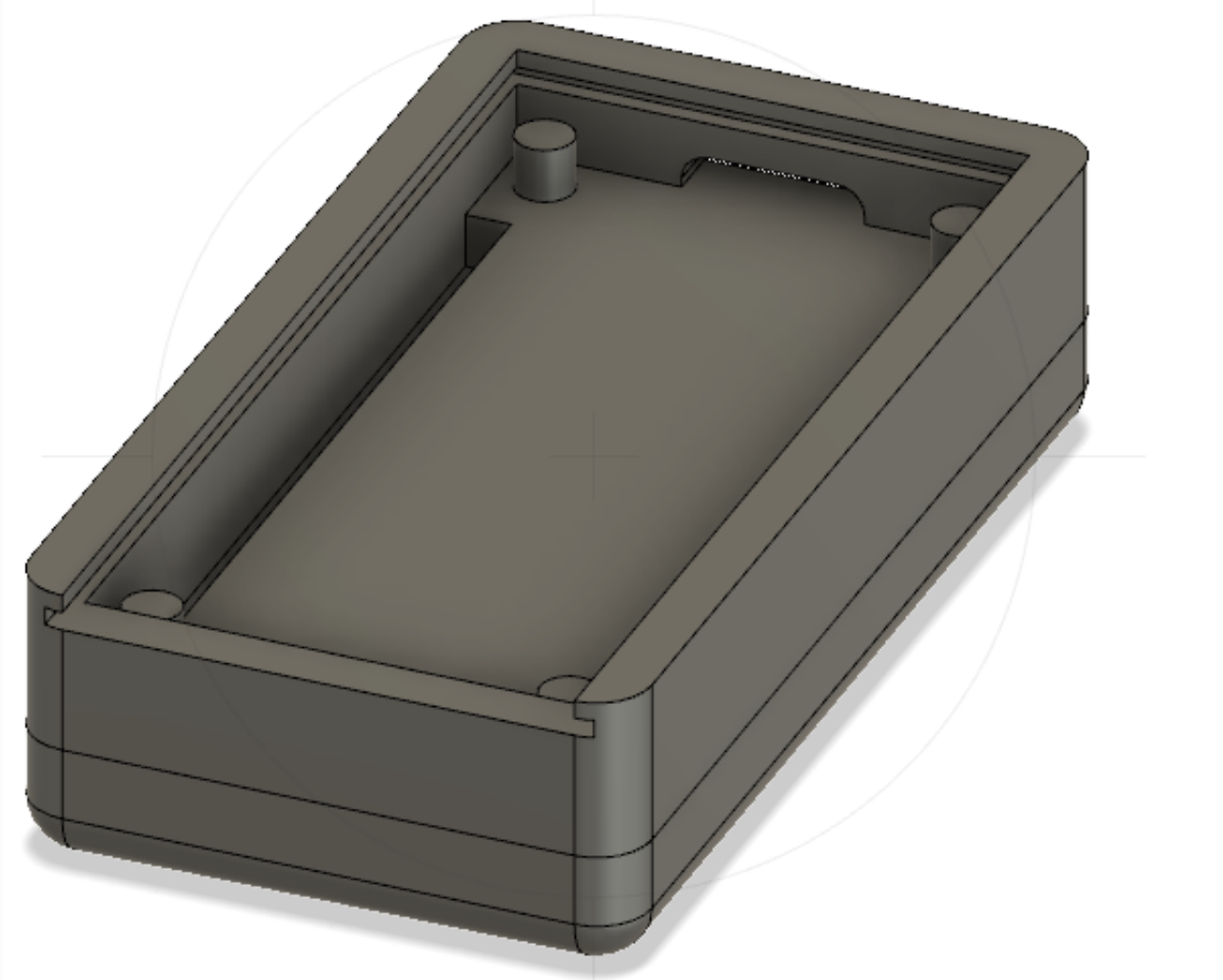




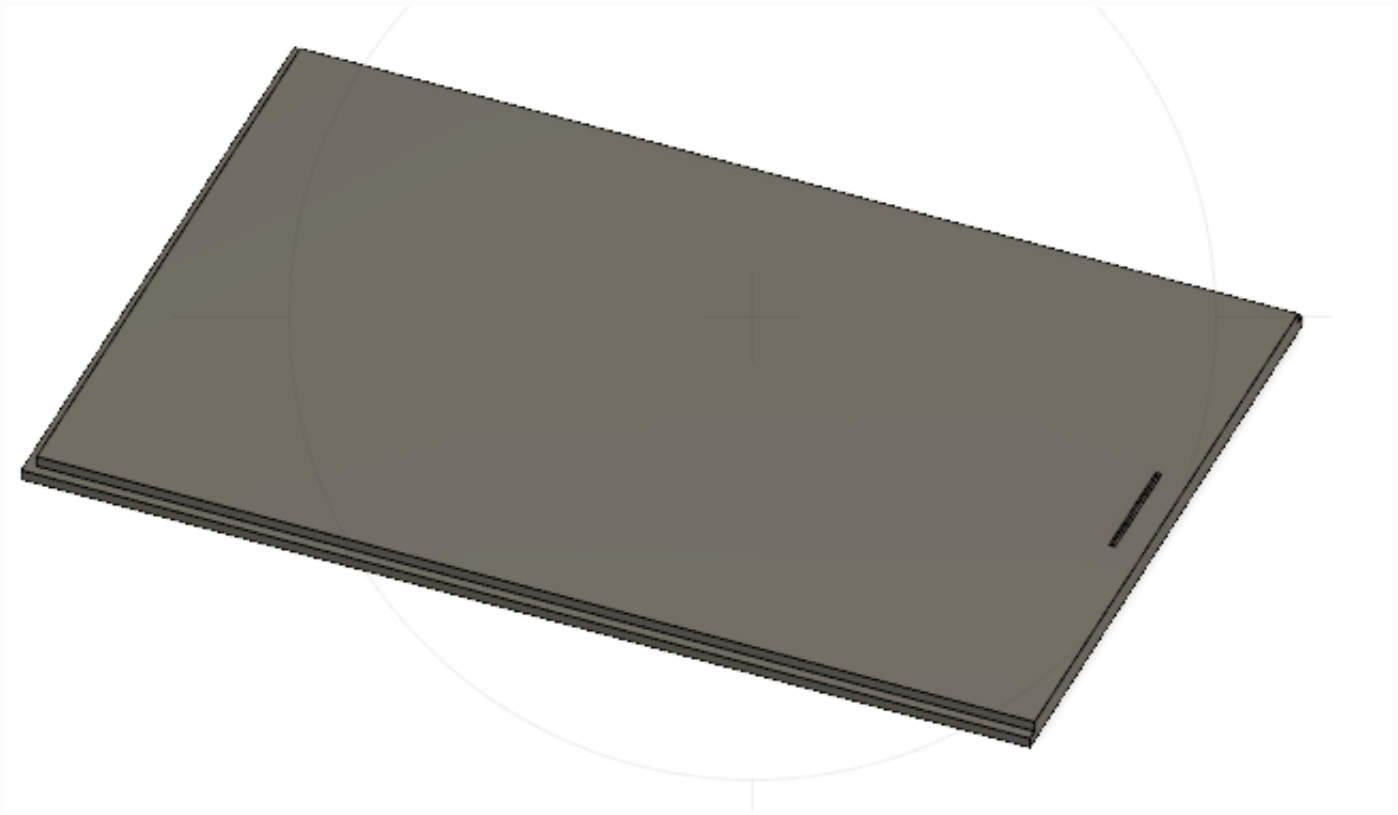
Illustrasjon 9-11

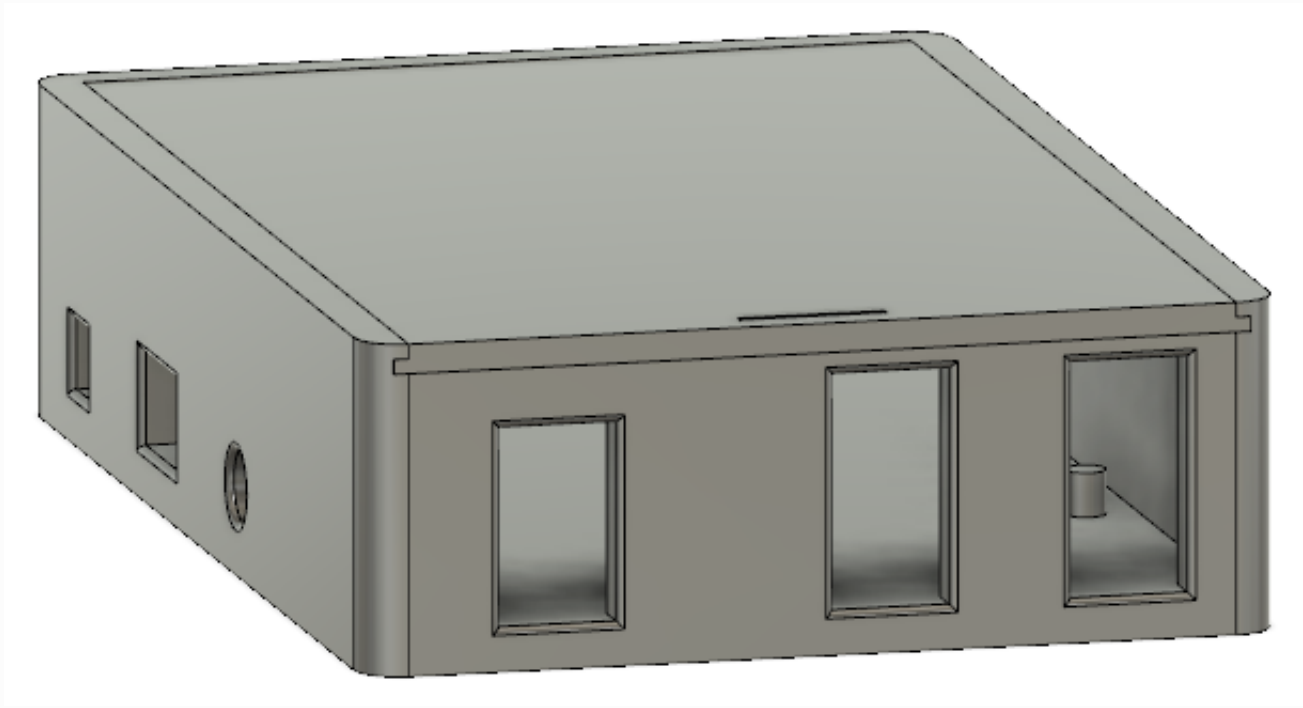


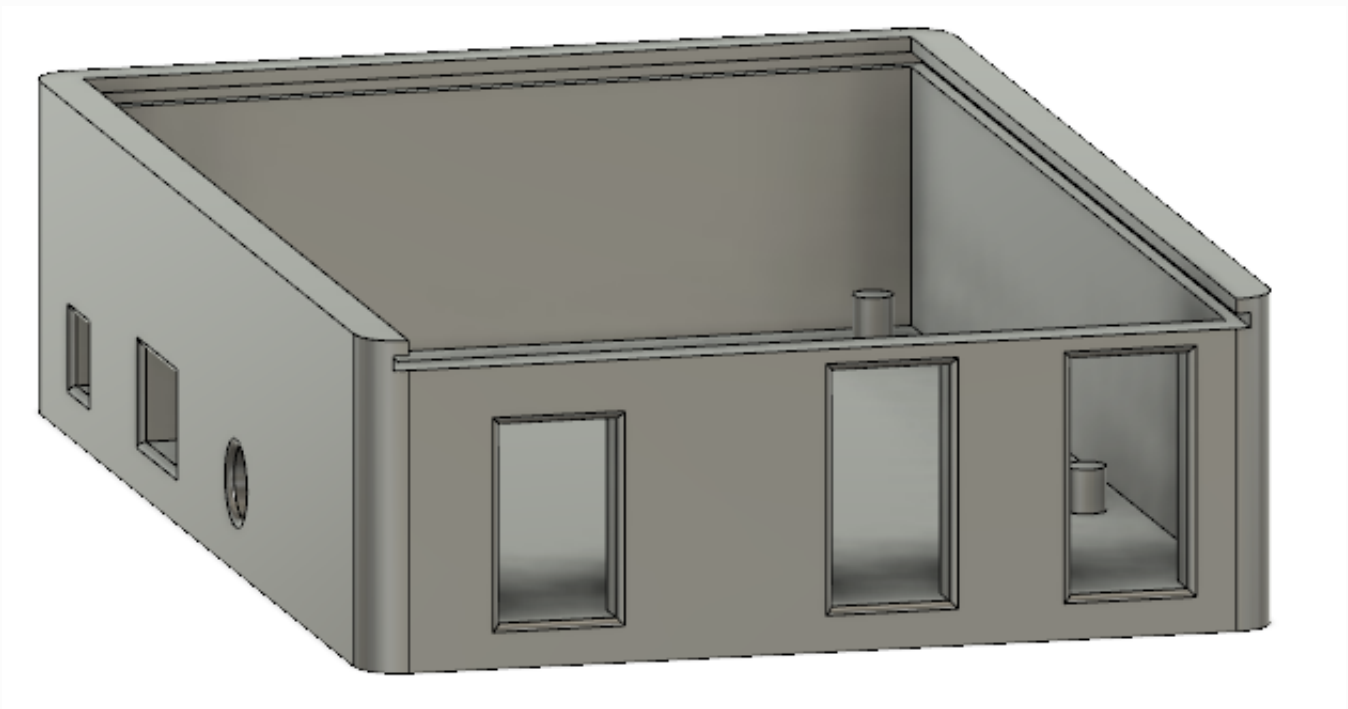




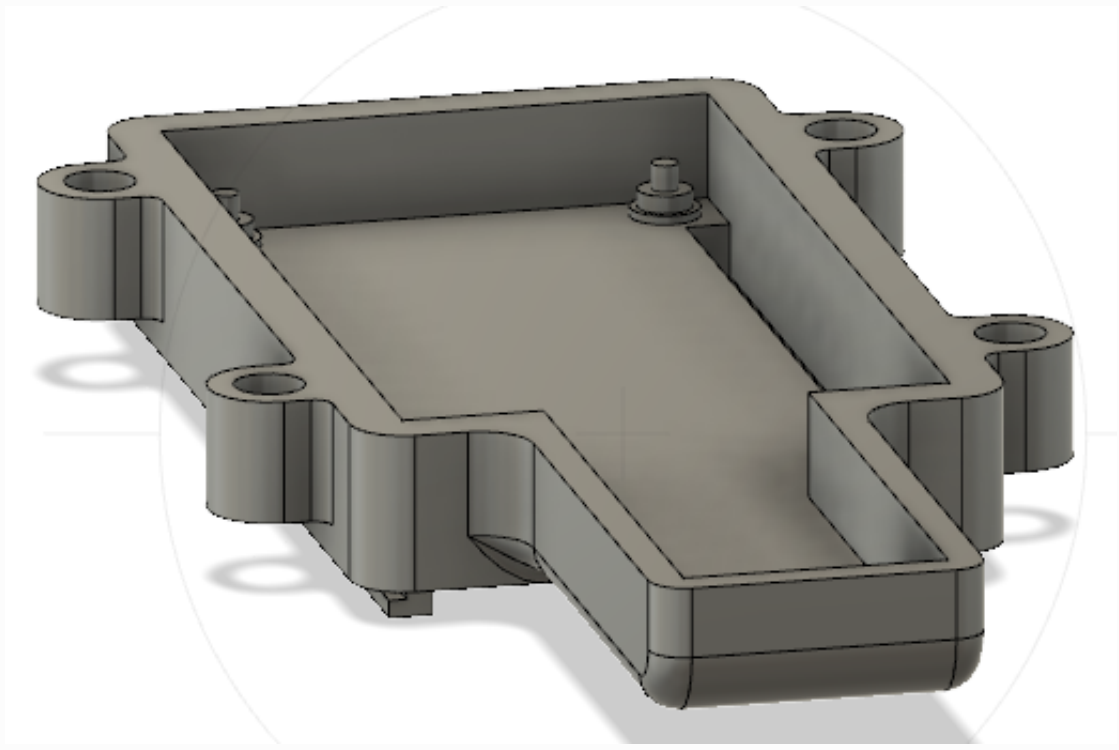
Illustrasjon 12-14

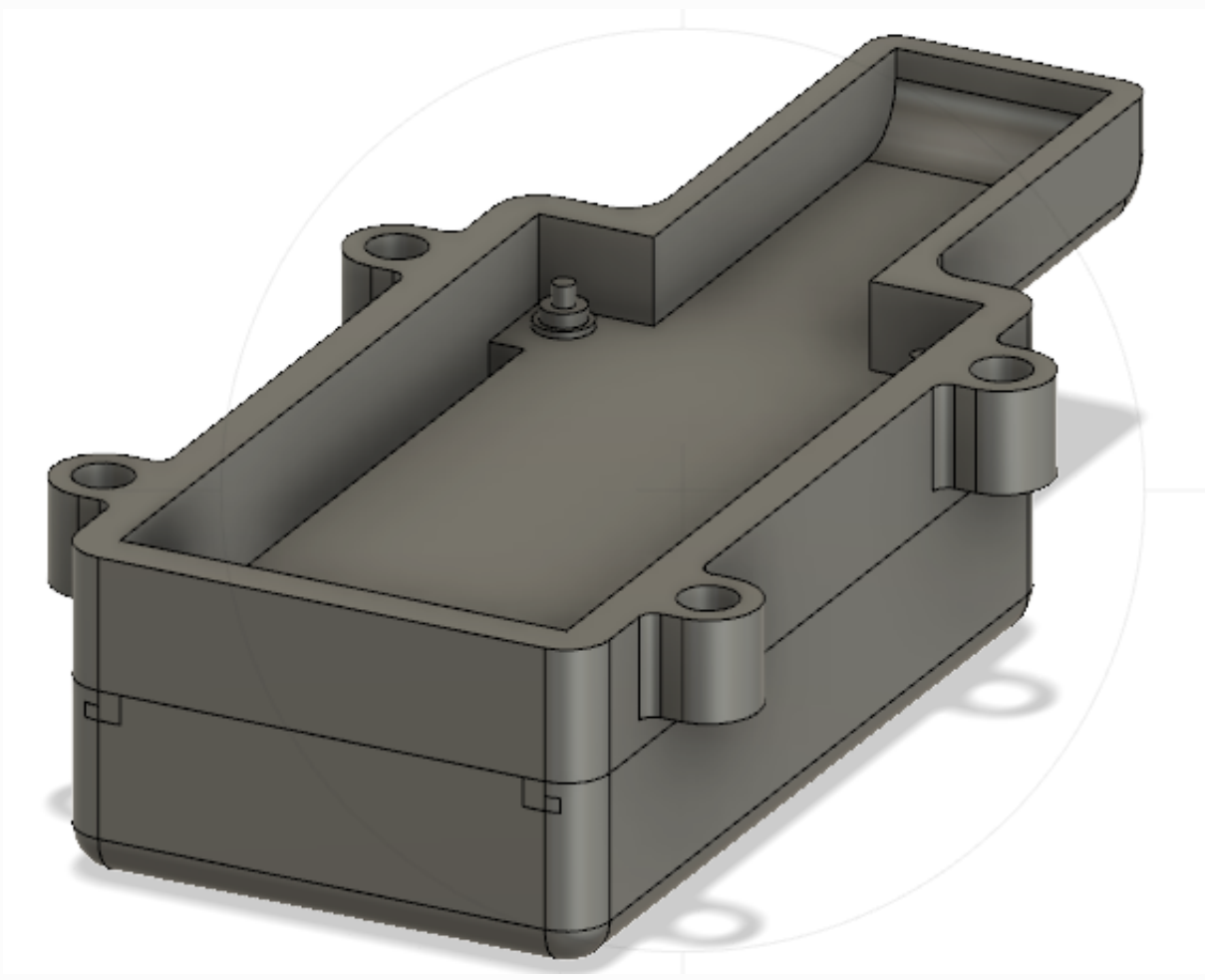


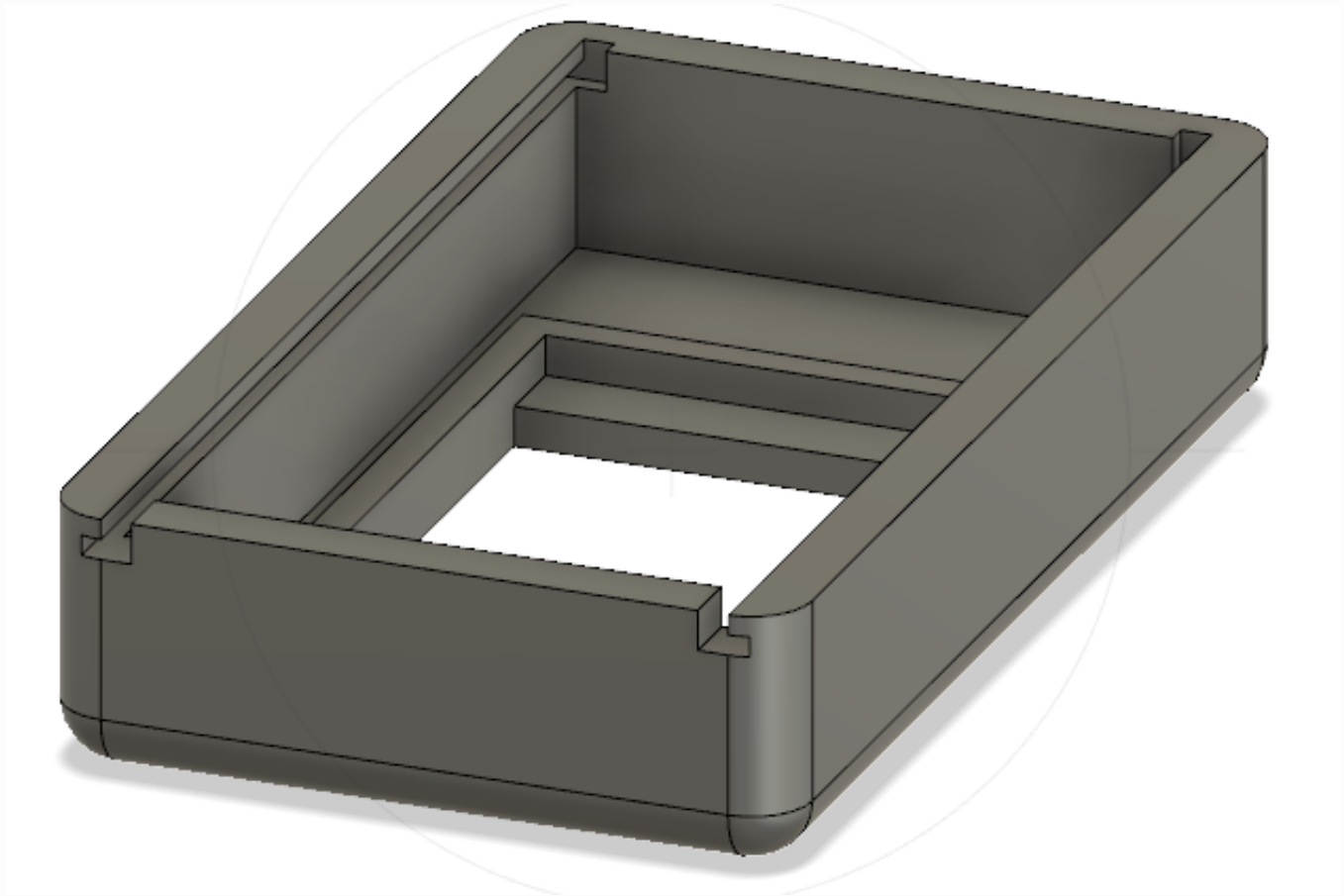




Illustrasjon 15-17

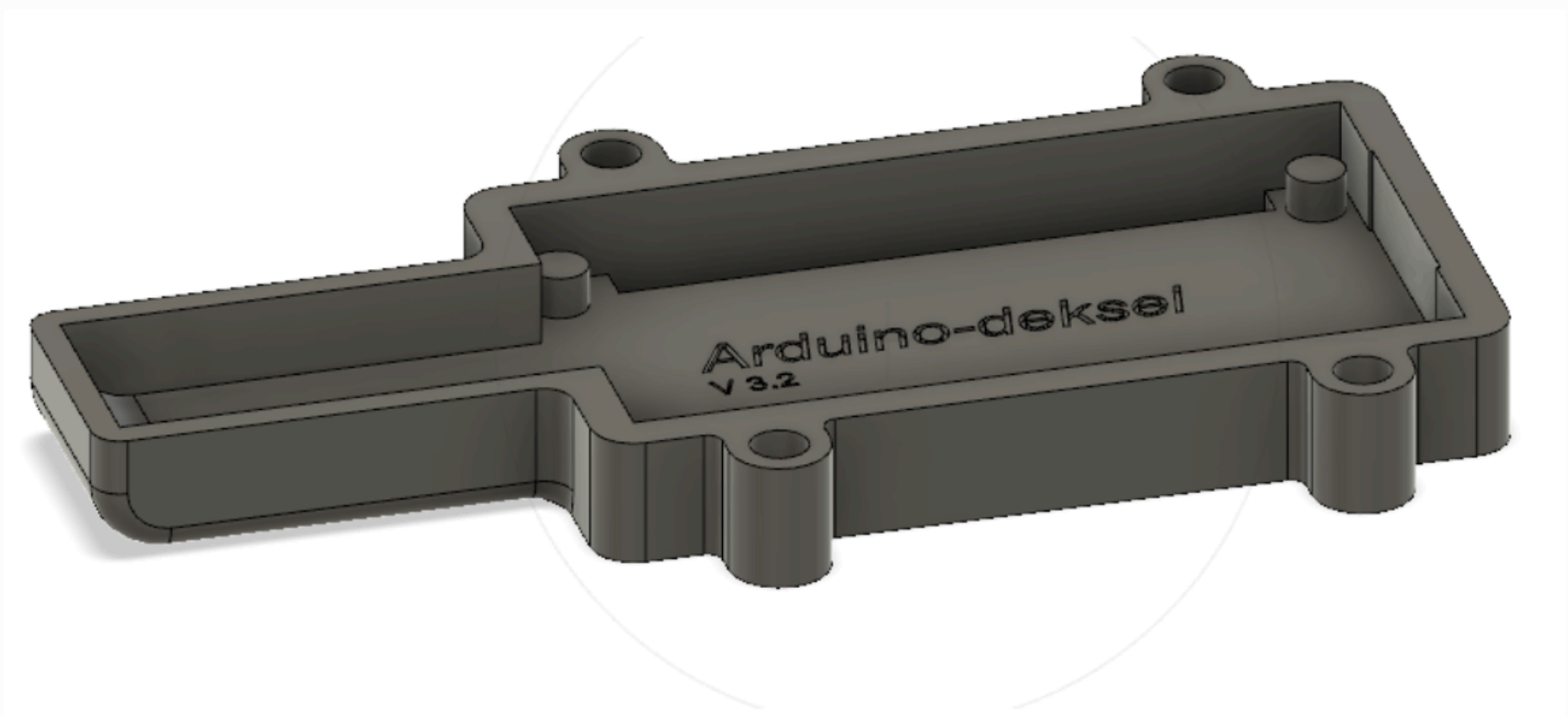


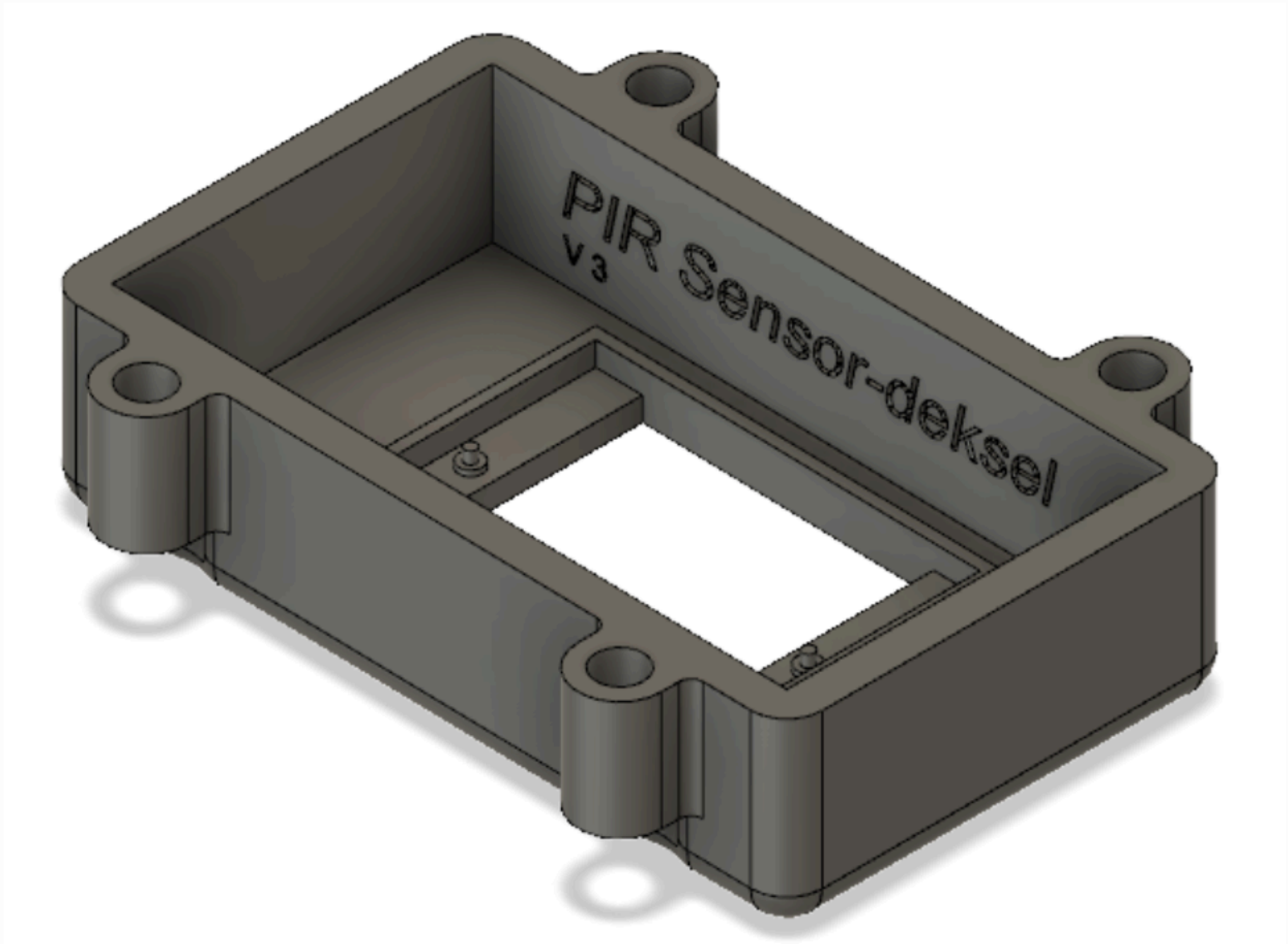




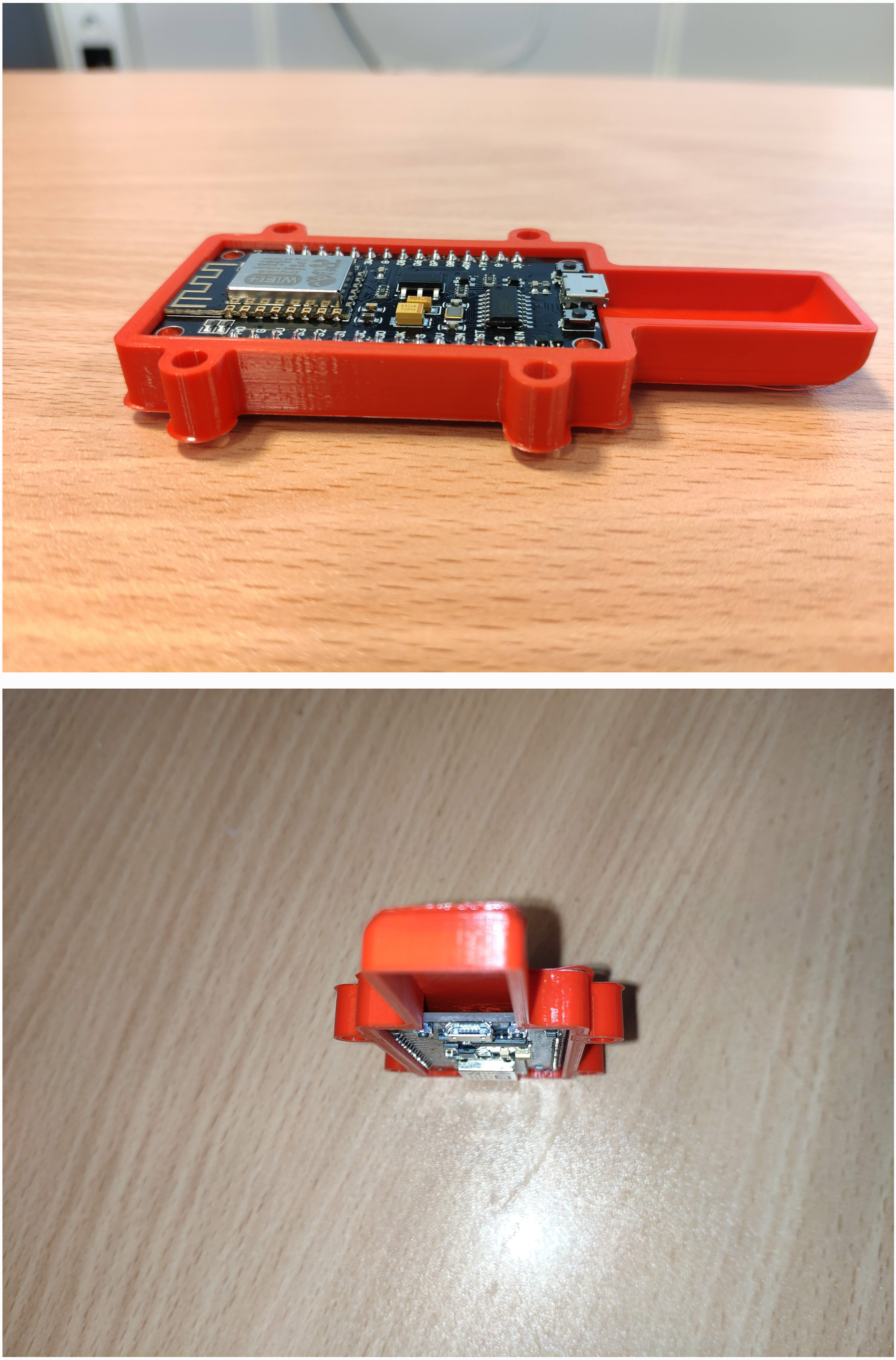
Bilde 18-19

Illustrasjoner 20-21






Bilde 22-24



## 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
```