```
***BOMRING KODE***

#include <Arduino_APDS9960.h>
#include <WiFi.h>
#include <PubSubClient.h>
#include <Wire.h>

//Wifi og wifipassord
const char* ssid = "NTNU-IOT";
const char* password = "";

//Broker adresse
const char* mqtt_server = "10.25.18.138";

WiFiClient espClient;
PubSubClient client(espClient);
long lastMsg = 0;
char msg[50];

int pushButton = 25;

void setup() {
  Serial.begin(115200);
  APDS.begin();
  pinMode(pushButton, INPUT);
  Serial.println("start");
  //MQTT setup
  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
}

//Kilde: Jiteshsaini 2022
void setup_wifi() {
  delay(10);
  //Kobler til wifi:
  Serial.println();
  Serial.print("Kobler til: ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi kobling opprettet");
  Serial.println("IP addresse: ");
  Serial.println(WiFi.localIP());
}

```

```
//Kilde: Jiteshsaini 2022
void callback(char* topic, byte* message, unsigned int length) {  //Funksjon som kalles på når en melc
  Serial.print("Melding ankommet topic: ");
  Serial.print(topic);
  Serial.print(". Melding: ");
  String messageTemp;

  for (int i = 0; i < length; i++) {
    Serial.print((char)message[i]);
    messageTemp += (char)message[i];
  }
  Serial.println();

  if (String(topic) == "esp32/output") {
    Serial.print("Endrer output til: ");
    if (messageTemp == "on") {
      Serial.println("på");
    } else if (messageTemp == "off") {
      Serial.println("av");
    }
  }
}

//Kilde: Jiteshsaini 2022
void reconnect() {  //Denne funksjonen kobler ESPen til MQTT
  client.subscribe("esp32/output");
  //Looper til en kobling er opprettet
  while (!client.connected()) {
    Serial.print("Forsøker å opprette kobling til mqtt...");
    //Prøver å koble til
    if (client.connect("bom1", "njaal", "3Inshallah4")) { //Erstatt bom1 med et unikt navn for hver bc
      Serial.println("connected");
      //Topic som det subscribes til
      client.subscribe("esp32/output");
    } else {
      Serial.print("mislykket kobling, rc=");
      Serial.print(client.state());
      Serial.println(" Prøver igjen om 5 sekund");
      delay(5000);
    }
  }
}

//Kilde: Øian 2024
bool button(int trueTime, bool pulldown) {
 //trueTime er hvor lenge man vil at knappen skal returnere "true"
  static bool val, buttonVar, lastButtonState = false;
  static uint32_t timer;
  if (digitalRead(pushButton) == pulldown) {
    buttonVar = true;  //Setter buttonVar til true mens knappen er klikket ned
    timer = millis();
  }
  if ((digitalRead(pushButton) != pulldown) && (buttonVar == true))  //Vil kjøre når knappen slippes c
  {
```

```
    val = true;
    if (millis() - timer > trueTime) {
      val = false;
      buttonVar = false;
    }
  }
  return val;
}

short proxRead() {  //Leser av nærhetsensoren og returnerer det som en short int
  static short proximity;
  if (APDS.proximityAvailable()) proximity = APDS.readProximity();
  return proximity;
}

short carCount(short proxy) {  //Teller hvor mange biler som har kjørt forbi bommen, tar inn proximity
  static short cCounter = -1;  //Vil autmatisk telle +1 når koden kjøres første gang, verdi på -1 gjør
  static bool countState = false;
  if (proxy < 200) countState = true;  //Veien ligger på runt 220, registerer når noe har kommet til b
  if (proxy > 200 && countState) {     //Når bilen har kjørt helt gjennom bommen, vil bilen bli telt
    cCounter++;
    countState = false;
  }
  return cCounter;
}

short carCount60s() {  //Teller hvor mange biler som har passert de siste 60 sekundene
  static unsigned long carArr[100] = {};
  static short prevCount = carCount(proxRead());
  static short cc60;
  static bool flag = false;
  if (prevCount != carCount(proxRead()) && flag) {  //Om det har passert en ny bil telles det opp en v
    cc60++;                                         //og tidspunktet lagres
    carArr[cc60 - 1] = millis();
    prevCount = carCount(proxRead());
  } else if (prevCount != carCount(proxRead())) {  //Uten flagget vil funksjonen telle en bil mer enn
    flag = true;
    prevCount = carCount(proxRead());
  }

  if (carArr[0] + 60000 <= millis() && carArr[0] != 0) {  //Etter at det har gått 60 sekunder i første
    for (int i = 0; i < cc60; i++) {                      //og alle tidspunkter går ned en indeks
      carArr[i] = carArr[i + 1];
    }
    cc60--;
  }
  return cc60;
}

int* colorRead() {  //Leser av fargesensoren og returnerer det som ett array
  static int rgb[3];
  while (!APDS.colorAvailable()) {
    delay(5);
  }
  APDS.readColor(rgb[0], rgb[1], rgb[2]);
```

```
  return rgb;
}

int* calibrateCol() {  //Tar 10 målinger over 1.2 sekunder og finner gjennomsnittet
  static uint32_t colCalTime = millis();
  static short count;
  static int base[3], prevBase[3];

  if (button(1250, true) && millis() - colCalTime >= 100) {  //Hvert 100 millisekund tar den en måling
    int* read;
    read = colorRead();
    for (short i; i <= 2; i++) {
      base[i] = base[i] + read[i];
    }
    count++;
    Serial.printf("count: %d\n", count);
    colCalTime = millis();
  }
  if (count == 10) {  //Etter 10 målinger vil gjennomsnittet bli lagret
    for (short i; i <= 2; i++) {
      base[i] = (base[i] - prevBase[i]) / 10;
      prevBase[i] = base[i];
    }
    count = 0;
  }
  return base;
}

String IDcheck() {  //Retunerer en kommaseparert fargekode med carCount60s på slutten
  String ID;
  int* baseColor;
  baseColor = calibrateCol();
  int* curColor;
  curColor = colorRead();
  static int colorCheck[3];
  for (short i; i <= 2; i++) {  //Får verdiene fra bomringen til å stemme overens med verdiene på lade
    if (i == 2) {
      curColor[i]++;
    }
    ID += String(colorCheck[i] = map(colorCheck[i] = curColor[i] - baseColor[i], -10, 255, 0, 24));  /
    ID += ",";                                                                                        /
  }
  ID += String(carCount60s());
  return ID;
}

void printOnce() {  //Printer kun når det er ny informasjon, og om den lagra informasjonen har hendt d
  static String prevInput = IDcheck();
  static String dataArr[50] = { String(0) };
  static uint32_t dataTime[50] = { millis() };
  static int j;
  static bool io = false;
  if (j < 10) {
    j = 0;
  }
```

```
  if (prevInput != IDcheck()) {
    String data = IDcheck();
    for (int i; i <= 10; i++) {  //Sjekker om nåværende datapunktet er anderledes fra de 10 siste
      if (data == dataArr[i]) {
        io = true;
        break;
      } else io = false;
    }
    if (!io) {                                //Om ny data er annerledes, send data
      int length = data.length();             //Kilde: Geeks for geeks 2023
      char* sendArr = new char[length + 1];  // -----""-----
      strcpy(sendArr, data.c_str());          // -----""-----
      Serial.println(data);
      client.publish("esp32/output", sendArr);
      dataArr[j] = data;
      j++;
    }
    prevInput = IDcheck();
  }
}

void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();
  printOnce();
}

***LADESTASJON KODE***

#include <Arduino_APDS9960.h>
#include <WiFi.h>
#include <PubSubClient.h>
#include <Wire.h>


//Wifi og wifipassord
const char* ssid = "NTNU-IOT";
const char* password = "";

//Broker adresse
const char* mqtt_server = "10.25.18.138";

WiFiClient espClient;
PubSubClient client(espClient);
long lastMsg = 0;
char msg[50];
int kwattsCharged;
String green = "1.1.1";

int pushButton = 25;

void setup() {
  Serial.begin(115200);
```

```
  APDS.begin();
  pinMode(pushButton, INPUT);
  Serial.println("start");
  //MQTT settup
  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
}

//Kilde: Jiteshsaini 2022
void setup_wifi() {
  delay(10);
  //Kobler til wifi:
  Serial.println();
  Serial.print("Kobler til: ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi kobling opprettet");
  Serial.println("IP addresse: ");
  Serial.println(WiFi.localIP());
}

//Kilde: Jiteshsaini 2022
void callback(char* topic, byte* message, unsigned int length) {  //Funksjon som kalles på når en melc
  Serial.print("Melding ankommet topic: ");
  Serial.print(topic);
  Serial.print(". Melding: ");
  char charMessage;
  for (int i = 0; i < length; i++) {
    charMessage += (char)message[i];
  }
  int intValue = charMessage - '0';
  Serial.println(intValue);
  kwattsCharged = intValue;
}

//Kilde: Jiteshsaini 2022
void reconnect() {  //Denne funksjonen kobler ESPen til MQTT
  client.subscribe("car2Charge");
  //Looper til en kobling er opprettet
  while (!client.connected()) {
    Serial.print("Forsøker å opprette kobling til mqtt...");
    //Forsøker å koble til
    if (client.connect("ESP32Charge", "njaal", "3Inshallah4")) {
      Serial.println("connected");
      //Topic som det subscribes til
      client.subscribe("car2Charge");
```

```
    } else {
      Serial.print("mislykket kobling, rc=");
      Serial.print(client.state());
      Serial.println(" Prøver igjen om 5 sekund");
      delay(5000);
    }
  }
}

//Kilde: Øian 2024
bool button(int trueTime, bool pulldown) {
  //trueTime er hvor lenge man vil at knappen skal returnere "true"
  static bool val, buttonVar, lastButtonState = false;
  static uint32_t timer;
  if (digitalRead(pushButton) == pulldown) {
    buttonVar = true;  //Setter buttonVar til true mens knappen er klikket ned
    timer = millis();
  }
  if ((digitalRead(pushButton) != pulldown) && (buttonVar == true))  //Vil kjøre når knappen slippes o
  {
    val = true;
    if (millis() - timer > trueTime) {
      val = false;
      buttonVar = false;
    }
  }
  return val;
}

int* colorRead() {  //Leser av fargesensoren og returnerer det som ett array
  static int rgb[3];
  while (!APDS.colorAvailable()) {
    delay(5);
  }
  APDS.readColor(rgb[0], rgb[1], rgb[2]);
  return rgb;
}

int* calibrateCol() {  //Tar 10 målinger over 1.2 sekunder og finner gjennomsnittet
  static uint32_t colCalTime = millis();
  static short count;
  static int base[3], prevBase[3];
  if (button(1250, true) && millis() - colCalTime >= 100) {  //Hvert 100 millisekund tar den en måling
    Serial.print("Counts: ");
    Serial.println(count);
    int* read;
    read = colorRead();
    for (short i; i <= 2; i++) {
      base[i] = base[i] + read[i];
    }
    count++;
    colCalTime = millis();
  }
  if (count == 10) {  //Etter 10 målinger vil gjennomsnittet bli lagret
    for (short i; i <= 2; i++) {
```

```
      base[i] = (base[i] - prevBase[i]) / 10;
      prevBase[i] = base[i];
    }
    count = 0;
  }
  return base;
}

String IDcheck() {  //Retunerer en kommaseparert fargekode med lademengden på slutten
  String ID;
  int* baseColor;
  baseColor = calibrateCol();
  int* curColor;
  curColor = colorRead();
  static int colorCheck[3];
  for (short i; i <= 2; i++) {
    ID += String(colorCheck[i] = map(colorCheck[i] = curColor[i] - baseColor[i], -10, 255, 0, 24));  /
    ID += ",";                                                                                        /
  }
  ID += String(kwattsCharged);
  return ID;
}

void printOnce() {  //Printer kun når det er ny informasjon, og om den lagra informasjonen har hendt d
  static String prevInput = IDcheck();
  static String dataArr[50] = { String(0) };
  static uint32_t dataTime[50] = { millis() };
  static int j;
  static bool io = false;
  if (j <= 10) {
    j = 0;
  }
  if (prevInput != IDcheck()) {
    String data = IDcheck();
    for (int i; i < 10; i++) {  //Sjekker om nåværende datapunktet er annerledes fra de 10 siste
      if (data == dataArr[i]) {
        io = true;
        break;
      } else {
        io = false;
        Serial.println("ji");
      }
    }
    if (!io) {                              //Om ny data er anderledes, send data
      int length = data.length();           //Kilde: Geeks for geeks 2023
      char* sendArr = new char[length + 1];  // -----""-----
      strcpy(sendArr, data.c_str());        // -----""-----
      Serial.println(data);
      client.publish("esp32/output", sendArr);
      dataArr[j] = data;
      j++;
    }
    prevInput = IDcheck();
  }
}
```

```
void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();
  printOnce();
}

***FAKTURERINGS PROGRAM***

#Importerer bibliotek
#KILDE: SQLite. (23/5/2024) What is SQLite? (Versjon 3.46.0)[Programvare] https://sqlite.org/ Hentet (
#Eclipse. (29/4/2024) Paho.mqtt.python (Versjon 2.1.0 ) [Programvare] https://github.com/eclipse/paho.

import sqlite3
import paho.mqtt.client as mqtt

# Konfigurerer innloggingsinformasjon og broker adresse for MQTT
broker_address = "10.25.18.138"
port = 1883
username = "njaal"
password = "3Inshallah4"

# Global variabel for å lagre motatt beskjed fra MQTT
global_payload = None

# Callback for når klienten kobler seg til broker
def on_connect(client, userdata, flags, rc):
    if rc == 0:
        print(f"Tilkobling var vellykket {rc}")
        # Abonnerer på topic "esp32/output"
        client.subscribe("esp32/output")
    else:
        print(f"Mislykket tilkobling {rc}")

# Callback for når en melding er motatt fra brokeren
def on_message(client, userdata, msg):
    global global_payload
    # Dekoder meldingen og bruker split funksjonen for å gjøre meldingen om til en liste
    payload = msg.payload.decode('utf-8')
    print(f"Motatt melding: '{payload}' på topic: '{msg.topic}'")
    global_payload = payload.split(',')
    global_payload = payload.split(',')
    print(global_payload)
    # Caller på check_values funksjonen avhengig av den nye meldingen som har blitt motatt
    check_values(global_payload)

# Callback for når beskjed publiseres
def on_publish(client, userdata, mid):
    print(f'Følgende melding ble publisert:{mid}')

# Lager en instans av mqtt.Client
client = mqtt.Client("Python_faktura")

# Setter passord og brukernavn
```

```
client.username_pw_set(username, password)

# Tildeler de ulike callback-funksjonene
client.on_connect = on_connect
client.on_message = on_message
client.on_publish = on_publish

# Kobler seg til broker
client.connect(broker_address, port, 60)

# Dette er hovedfunksjonen til programmet, calles kun når en melding har blitt motatt
def check_values(payload):
    # Kobler seg på databasen
    conn = sqlite3.connect('smartcity.db')
    cursor = conn.cursor()

    # Query som henter alle radene i tabellen:
    cursor.execute('SELECT * FROM bilskilt2')
    match_found = False
    #Legger til payload i trafikk.txt fil slik at denne informasjonen kan anvendes av andre program
    fil_plassering = '/var/www/html/trafikk.txt'
    with open(fil_plassering, 'w') as file:
        file.write(f"{payload[3]},{payload[3]}\n")
    # Itererer gjennom alle radene i den første tabellen for å prøve å finne en match
    for row in cursor:
        id, num1, num2, num3, classification = row
        if num1 == payload[0] and num2 == payload[1] and num3 == payload[2]:
            print(f'Bilen er av type: {classification}')
            match_found = True
            # Hvis det finnes en match i tabellen henter programmet alle radene i den neste tabellen
            if match_found == True:
                cursor.execute('SELECT * FROM bompenger')
                # Itererer gjennom radene i den neste tabellen
                for row in cursor:
                    id, pris, prisklasse = row
                    # Dersom det finnes en match i denne tabellen brukes denne
                    # informasjonen til å  regne ut et fakturabeløp
                    if prisklasse == classification:
                        fakturerings_belop = pris * float(payload[3]) + 20
                        print(f'Totalt faktureringsbeløp på: {fakturerings_belop}kr')
                        # Faktureringsbeløpet sendes til topicen pytophp
                        try:
                            msg = str(fakturerings_belop)
                            pubMsg = client.publish(
                            topic = 'pytophp',
                            payload = msg.encode('utf-8'),
                            qos = 0,
                            )
                            if pubMsg.is_published():
                                print("Velykket fakturering:")
                        finally:
                            print(f'Faktureringsbeløp sendt til eier av {classification}')
            break

    if not match_found:
```

```
        print('Ingen match i database')

    conn.close()

# Setter opp tabell dersom den ikke allerede eksisterer
if __name__ == "__main__":
    conn = sqlite3.connect('smartcity.db')
    cursor = conn.cursor()
    cursor.execute('''
        CREATE TABLE IF NOT EXISTS bilskilt (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            num1 TEXT,
            num2 TEXT,
            num3 TEXT,
            classification TEXT
        )
    ''')

    # Verdier som kan legges til i tabellen
    # cursor.execute('INSERT INTO bilskilt (num1, num2, num3, classification) VALUES (?, ?, ?, ?)', ('
    # cursor.execute('INSERT INTO bilskilt (num1, num2, num3, classification) VALUES (?, ?, ?, ?)', ('
    # cursor.execute('INSERT INTO bilskilt (num1, num2, num3, classification) VALUES (?, ?, ?, ?)', ('
    # cursor.execute('INSERT INTO bilskilt (num1, num2, num3, classification) VALUES (?, ?, ?, ?)', ('
    conn.commit()
    conn.close()

# Starter MQTT loopen for å kunne prosessere callback-funksjonene
client.loop_forever()
```

MOTAK AV FAKTURA PÅ WEBSERVEREN

```
<?php
//KILDE: Namoshek. (24/4/2024) php-mqtt/client (Versjon 2.1.0) [Programvare] https://github.com/php-mq
//KILDE: Guowei, Li. (5.8.2021) How to use MQTT in PHP https://www.emqx.com/en/blog/how-to-use-mqtt-in
// Må inkludere vendor ettersom det er der php-bibliotekene lagres
require 'vendor/autoload.php';

// Importerer alle bibliotek
use PhpMqtt\Client\MqttClient;
use PhpMqtt\Client\ConnectionSettings;

// Oppretter en rekke variabler for å kunne koble seg på MQTT
$server   = '10.25.18.138';
$port     = 1883;
$clientId = 'PHP_Client';
$username = 'njaal';
$password = '3Inshallah4';
$topic    = 'pytophp';

//$jsonFile = 'data.json';
//if (file_exists($jsonFile)) {
//    $data = json_decode(file_get_contents($jsonFile), true);
//} else {
//    $data = [];
//}
```

```php
//$dijkstra = file_get_contents('data.txt');
//echo $dijkstra;

try {
    // Lager en ny instans av MQTT client
    $mqtt = new MqttClient($server, $port, $clientId);
    $connectionSettings = (new ConnectionSettings)
        ->setUsername($username)
        ->setPassword($password);
    // Kobler seg til MQTT brokeren
    $mqtt->connect($connectionSettings, true);
    echo "Oprettet tilkobling til Broker .\n";

    // Abonnerer til topic 'pytophp'
    $mqtt->subscribe($topic, function ($topic, $message) {
        printf("Melding er blitt motatt på topic [%s]: %s\n", $topic, $message);

        // Programmet skjekker at data.json eksisterer
        $jsonFile = 'data.json';
        if (file_exists($jsonFile)) {
            // Dersom filen eksisterer er $data variabelen lik inneholdet i filen:
            $data = json_decode(file_get_contents($jsonFile), true);
        } else {
            // Hvis ikke er det en tom liste
            $data = [];
        }
        // Her filtreres ut uønskede meldinger,
        // grunnen til at meldinger som starter med '{'
        // filtreres ut er at det konstant ble publisert en
        // melding som startet med dette tegnet, som vi ikke
        // ønsket å legge til i tabellen til brukeren på nettsiden
        if(str_starts_with($message, "{")) {
            echo "   Filtrert ut";
        } else {
            // Hvis ikke meldingen starter med '{', legges meldingen og et tidsstempel til i .json fil
            // Her er ikke biltypen avhengig av noe annet, den ble for enkelhetsskyld bestemt til å væ
            // en diesel bil
            $nydata = ['tidspunkt' => date("h:i:sa"), 'belop' => $message, 'Biltype' => 'Diesel'];
            $data[] = $nydata;
            file_put_contents($jsonFile, json_encode($data, JSON_PRETTY_PRINT));
        }}, 0);
        //$payload = $dijkstra;
        //$cleanedPayload = trim($payload);
        //$mqtt->publish('phptozumo', json_encode($cleanedPayload), 0, true);

    // Vi ønsker at programmet skal konstant være i MQTT-loopen
    // slik at det alltid kan motta meldinger mens det kjører
    $mqtt->loop(true);
    //$mqtt->disconnect();
// til slutt en catch i tilfelle en feil skal oppstå
} catch (\Exception $e) {
    echo sprintf("An error occurred: %s\n", $e->getMessage());
}
?>
```

```
***BRUKER 1***

// KILDE: Szot, Kamil. (16/4/2013). Creating a table with PHP foreach function
// https://medium.com/personal-work-revision/how-to-parse-json-into-html-table-with-php-for-dummies-1d

<?php
// henter data fra data.json
$jsonFile = 'data.json';
$data = [];
if (file_exists($jsonFile)) {
    // Lagrer innholdet i data.json i variabelen $data
    $data = json_decode(file_get_contents($jsonFile), true);
}
?>

<!DOCTYPE html>
<html>
<head>
    <title>Mitt dashbord:</title>
</head>
<body>
    // Bilde legges inn
    <h3>Min side:</h3>
    <img src="/BIELSYS.png" alt="Image" width="300">
</body>
<br>
</body>
</html>

<head>
    // Charset bestemmer hvordan karakterer skal kodes i HTML-dokumentet
    <meta charset="UTF-8">
    <title>Mitt Dashboard</title>
    <style>
        // Konfigurerer hvordan tabellen skal se ut
        table {
            width: 100%;
            border-collapse: collapse;
        }
        table, th, td {
            border: 1px solid black;
        }
        th, td {
            padding: 10px;
            text-align: left;
        }
    </style>
</head>
<body>
    // Overskrift
    <h1>Mitt dashbord</h1>
    <table>
        <thead>
            // Genererer den øverste radene i tabellen
```

```
            <tr>
                <th>Tidspunkt</th>
                <th>Beløp</th>
                <th>Biltype</th>
            </tr>
        </thead>
        <tbody>
        <script>
                // Denne funksjonen gjør at siden oppdateres hvert 50. sekund
                function autoRefresh() {
                    window.location = window.location.href;
                }
                setInterval('autoRefresh()', 50000);
            </script>
            // Denne koden bruker $data-variabelen til å legge til rader i tabellen
            <?php if (!empty($data)): ?>
                // For each funksjonen itererer gjennom verdiene i $data og legger de til
                <?php foreach ($data as $row): ?>
                    <tr> // Bruker 'tidspunkt', 'belop' og 'Biltype' for å plassere verdiene i riktig
                        <td><?php echo htmlspecialchars($row['tidspunkt']); ?></td>
                        <td><?php echo htmlspecialchars($row['belop']); ?> kr</td>
                        <td><?php echo htmlspecialchars($row['Biltype']); ?></td>
                    </tr>
                <?php endforeach; ?>
            <?php else: ?>
                <tr>
                    <td colspan="3">No data available.</td>
                </tr>
            <?php endif; ?>
        </tbody>
    </table>
</body>
</html>
***FORSIDE***
<!doctype html>
<!-- Her er det brukt en template for å lage layouten til forsiden:
    KILDE: Quakit. (2024) CSS Grid Templates https://www.quackit.com/html/templates/css_grid_templates
<title>CSS Grid Template 1</title>
<style>
body {
  display: grid;
  grid-template-areas:
    "header header header"
    "nav article ads"
    "nav footer footer";
  grid-template-rows: 80px 1fr 70px;
  grid-template-columns: 20% 1fr 15%;
  grid-row-gap: 10px;
  grid-column-gap: 10px;
  height: 100vh;
  margin: 0;
  }
header, footer, article, nav, div {
  padding: 1.2em;
  background: Grey;
```

```
  }
#pageHeader {
  grid-area: header;
  }
#pageFooter {
  grid-area: footer;
  }
#mainArticle {
    grid-area: article;
  }
#mainNav {
  grid-area: nav;
  }
#siteAds {
  grid-area: ads;
  }
/* Stack the layout on small devices/viewports. */
@media all and (max-width: 575px) {
  body {
    grid-template-areas:
      "header"
      "article"
      "ads"
      "nav"
      "footer";
    grid-template-rows: 80px 1fr 70px 1fr 70px;
    grid-template-columns: 1fr;
 }
}
</style>
<body>
<!-- BRUKERLISTE -->
  <header id="pageHeader">
    <h1> SMART CITY <h1>
  </header>
  <article id="mainArticle">Brukere
    <head>
        <title>Sender til brukersiden</title>
      </head>
     <body>
       <h1> Bruker 1 </h1>
       <!-- Input boks der inputtet/passordet "Bruker 1" leder til brukersiden -->
       <form action="bruker1.php" method="POST">
         <input type="Bruker 1"/>
       </form>
     </body>
     <body>
       <h1> Bruker 2 </h1>
       <!-- Input boks for bruker 2, sender brukeren til en test side -->
       <form action="ny_test.php" method="POST">
         <input type="Bruker 1"/>
       </form>
     </body>
     <body>
       <h1> Bruker 3 </h1>
```

```
      <!-- Input boks for bruker 3, sender brukeren til en annen test side -->
      <form action="brukere.php" method="POST">
        <input type="Bruker 1"/>
      </form>
     </body>

  </article>
<!-- NAVIGASJON HØYRE SIDE-->
  <nav id="mainNav">
  <arcticle id="Navigasjon">
    <head>
        <title>Ruteplanlegging</title>
  </head>
  <body>
      <h1> Planlegg din rute her </h1>
      <!-- Submit knapp som leder til ruteplanleggingssiden -->
      <form action="ruteplanlegging.php" method="POST">
        <input type="submit"/>
      </form>
  </body>
  </article>
  </nav>
<!-- VENSTRE SIDE -->
  <div id="siteAds">BIELSYS 2024
 </div>
 <!-- FOOTER -->
 <footer id="pageFooter">Footer <head>
       <!-- submit knapp som leder til en annen testside -->
       <title>Redirecting using Form Tags</title>
     </head>
    <body>
     <form action="test.php" method="POST">
       <input type="submit"/>
     </form>
    </body>
  </footer>
</body>

***RUTEPLANLEGGING***

<!DOCTYPE html>
// KILDE: Alaeddin, Amama. (27/2/2017) $_POST vs. $_SERVER['REQUEST_METHOD'] == 'POST'
// https://stackoverflow.com/questions/409351/post-vs-serverrequest-method-post Hentet (31/5/2024)
// KILDE: W3schools. (2024) PHP Superglobal -$GET https://www.w3schools.com/php/php_superglobals_get.a
<html>
<head>
    // Overskrift
    <title>Planlegg din rute her.</title>
</head>
<head>
    <h1> Planlegg din rute her </h1>
        // Legger inn bilde av kartet
        <img src="/kartet.png" alt="Image" width="600">
</head>
<body>
```

```
<?php
// Bruker her en superglobal variabel $_SERVER, 'Serverwide' global variabel
// Venter til brukerens input har blitt "submitted"
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // henter brukerens input og lagrer det i to variabler
    $startposisjon = $_POST['startposisjon'];
    $sluttposisjon = $_POST['sluttposisjon'];

    // Tømmer data.txt
    $file = './data.txt';
    // Legger til de nye inputtene fra brukeren
    file_put_contents($file, "$startposisjon,$sluttposisjon\n");
    // echo for å gi beskjed til brukeren av nettsiden at informasjonen har blitt sendt til bilen
    echo "<p>Din rute har blitt sendt til bilen.</p>";
}
?>
// Form genererer felt som brukeren kan fylle inn
// Inputtene til brukeren blir ikke sendt før submit knappen trykkes
<form method="post" action="">
  <label for="startposisjon">Din startposisjon:</label>
  // Gir inputtene ID og navn for å kunne lagre inputtene som variabler i PHP
  <input type="text" id="startposisjon" name="startposisjon"><br><br>
  <label for="sluttposisjon">Din sluttposisjon:</label>
  <input type="text" id="sluttposisjon" name="sluttposisjon"><br><br>
  <input type="submit" value="Submit">
</form>
</body>
</html>

***ALGORITME***

#FERDIG VERSJON:

#IMPORTS
import os
import time
import random
# KILDE: Redhia, Azka. (21/9/2023) Dijkstra's Algorithm in Python: Finding the shortes path
# https://medium.com/@azkardm/dijkstras-algorithm-in-python-finding-the-shortest-path-bcb3bcd4a4ea Her

# KILDE: Python Software Foundation. (2023). Python: The os module (Versjon 3.10) [Programvare].
# https://www.python.org/ Hentet (2/6/2024)
# KILDE: Python Software Foundation. (2023). Python: The random module (Versjon 3.10) [Programvare].
# https://www.python.org/ Hentet (2/6/2024)

import paho.mqtt.client as mqtt

# Inloggingsdetaljer og konfigurasjon
broker_address = "10.25.18.138"
port = 1883
username = "njaal"
password = "3Inshallah4"

# Callback for tilkobling
```

```
def on_connect(client, userdata, flags, rc):
    if rc == 0:
        print(f"Vellykket tilkobling {rc}")
        #client.subscribe("esp32/output")
    else:
        print(f"Mislykket tilkobling {rc}")

# Callback for når beskjed publiseres
def on_publish(client, userdata, mid):
    print("Rute sendt til bil")

# Callback for når en melding blir mottatt fra brokeren
def on_message(client, userdata, msg):
    global global_payload
    # Dekoder meldingen og printer den
    payload = msg.payload.decode('utf-8')
    print(f"Mottok melding: '{payload}' på topic: '{msg.topic}'")

# Lager en instans av mqtt.Client
client = mqtt.Client("Dijkstras")

# Setter passord og brukernavn
client.username_pw_set(username, password)
#Bruker randints for å simulere trafikk:
Val1 = random.randint(1,10)
Val2 = random.randint(3,7)
Val3 = random.randint(1,4)
#Henter den faktiske trafikkdataen fra bomstasjonen:
with open('/var/www/html/trafikk.txt', 'r') as file:
    # Les fil og split
    innhold = file.read()
    splitcontents = innhold.split(',')
#print(f'Trafikk1 {innhold[0]}')
#print(f'Trafikk2 {innhold[2]}')

# Listen med trafikkdata
bom_verdi =  [Val1, 1, Val2, 5, Val3, int(innhold[0]), Val3, 9, int(innhold[2]), Val2, 3, Val1, 3, 7]
# Nodene i kartet
punkter = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H']
ekstremalpunkter = ['G', 'H']

#Bruker OS funksjon for å hente start og sluttpunkt fra nettsiden
with open('/var/www/html/data.txt', 'r') as file:
    # Les fil og split
    contents = file.read()
    splitcontents = contents.split(',')
    print(f'startpunkt {contents[0]}')
    print(f'sluttpunkt {contents[2]}')

start = contents[0]
stopp = contents[2]

#Kilde brukt for å forstå og lage dijkstras algoritme, tilpasset vårt system:
#kilde: https://medium.com/@azkardm/dijkstras-algorithm-in-python-finding-the-shortest-path-bcb3bcd4a4
```

```
#Lager en liste som inneholder lister med alle veiene mellom de forskjellige nodene, inkludert tetthet
linjer = [
    ['A', 'B', bom_verdi[0]],
    ['A', 'C', bom_verdi[1]],
    ['B', 'A', bom_verdi[2]],
    ['B', 'D', bom_verdi[3]],
    ['B', 'G', 0],
    ['C', 'A', bom_verdi[4]],
    ['C', 'D', bom_verdi[5]],
    ['C', 'E', bom_verdi[6]],
    ['D', 'B', bom_verdi[7]],
    ['D', 'C', bom_verdi[8]],
    ['D', 'F', bom_verdi[9]],
    ['E', 'C', bom_verdi[10]],
    ['E', 'F', bom_verdi[11]],
    ['E', 'H', 0],
    ['F', 'D', bom_verdi[12]],
    ['F', 'E', bom_verdi[13]],
    ['G', 'B', 0],
    ['H', 'E', 0]
]

#Oppretter tabellen som i dijkstras algoritme vil oppdataeres med informasjon om den veien som gir lav
#Inkludert er den totale verdien på vei til hver node. Denne starter vi med å sette til uendelig stor,
tabell = [
    ['A', '', float('inf')],
    ['B', '', float('inf')],
    ['C', '', float('inf')],
    ['D', '', float('inf')],
    ['E', '', float('inf')],
    ['F', '', float('inf')],
    ['G', '', float('inf')],
    ['H', '', float('inf')]
]

#Definerer dijkstra-funksjonen med listen over punkter, veier, start og stopp.
def dijkstra(punkter, linjer, start, stopp):

    #Oppretter lister og variabler brukt i algoritmen. Disse forklares senere i koden.
    ikke_utforsket = punkter
    utforsket = []
    kart = []
    dist1 = 0
    remlist = []
    kart1 = []

    #Her fjerner vi alle ekstremalpunkter som ikke er satt til stopp punkt fra listen over noder som s
    #Dette gjøres fordi vi vet vi ikke ønsker å utforske disse nodene, dersom de hadde blitt utforsket
    #stoppe opp her.
    for e in range(len(ekstremalpunkter)):
        if ekstremalpunkter[e] != stopp:
            remlist.append(ekstremalpunkter[e])
    ikke_utforsket = [item for item in ikke_utforsket if item not in remlist]

    #Setter første node som utforskes til startpunktet
```

```
    for punkt in tabell:
        if punkt[0] == start:
            punkt[2] = 0

    #Denne delen kjøres igjennom helt til vi har nådd stopp noden via den veien med lavest verdier.
    while len(utforsket) <= len(tabell): #Koden kjører til alle noder er utforsket
        for vei in linjer: #Ser på alle veiene ut fra noden som utforskes
            if vei[0] == start: #Utforsker først startnoden, dette punktet vil endres underveis når kc
                if vei[1] in ikke_utforsket: #Utforsker kun veier som går til ikke-utforskede noder.
                    for punkt in tabell: #Ser gjennom listene i tabellen.
                        if punkt[0] == vei[1] and vei[2] + dist1 < punkt[2]: #Finner listen i tabeller
                                                                             #Sjekker at denne veien t
                            punkt[1] = vei[0] #I det tifellet hvor alt i if-setningen stemmer settes r
                            for p in tabell: #Denne for-løkken går gjennom tabellen  og finner den all
                                if p[0] == vei[0]:
                                    dist = p[2] #dist-variabelen settes til den totale veien til noder
                            punkt[2] = vei[2] + dist #veien til noden koblet til noden som utforskes v
                            dist1 = punkt[2] #dist1-variabelen settes til den nye totale verdien til v

        utforsket.append(start) #noden som akkurat ble utforsket settes inn i en liste med alle utfors

        #Fjerner den utforskede noden fra listen over ikke-utforskede noder:
        for punk in range(len(ikke_utforsket) - 1):
            if ikke_utforsket[punk] == start:
                del ikke_utforsket[punk]

        Setter nytt startpunkt basert på den noden med en vei til den akkurat utforskede noden med lav
        lengde = float('inf')
        for punkt in tabell:
            if punkt[0] in ikke_utforsket:
                if lengde > punkt[2]:
                    lengde = punkt[2]
                    start = punkt[0]

    #Lager en liste som inneholder nodene som gir den lavest mulige verdien på veien fra start til stc
    i = True
    while stopp != '':
        for punkt in tabell:
            if punkt[0] == stopp:
                if i == True:
                    total_lengde = punkt[2]
                    i = False
                kart.append(punkt[0])
                stopp = punkt[1]

    #inverterer listen slik at den inneholder nodene fra start til stopp.
    kart1 = kart[::-1]
    return kart1, total_lengde #returnerer liste med noder i rekkefølge og den totale verdien til denr

def f(): #Denne funksjonen tar listen fra dijkstra og omgjør det til en liste med instruksjoner til bi

    #Definerer to lister som brukes senere i funksjonen
    intKart = []
    dirKart = []
```

```
#Omgjør nodene i listen fra dijkstra til tall. Grunnen til at disse er i par er fordi kjørebanen e
for i in dijkstra(punkter, linjer, start, stopp)[0]:
    if(i == 'A' or i == 'F'):
        intKart.append(2)
    if(i == 'B' or i == 'E'):
        intKart.append(3)
    if(i == 'C' or i == 'D'):
        intKart.append(1)
    if(i == 'G' or i == 'H'):
        intKart.append(4)

#Denne koden er hva man vil kalle hard-coded. Dette ble gjort av tidsmessige årsaker.
#Her genereres listen med instruksjoner.
for i in range(len(intKart)-2):
    if(intKart[i] == 4):
        if(intKart[i+2] == 1):
            dirKart.append(2)
        else:
            dirKart.append(3)
    elif(intKart[i] == 3):
        if(intKart[i+1] == 1):
            if(intKart[i+2] == 1):
                dirKart.append(3)
            else:
                dirKart.append(2)
        else:
            dirKart.append(1)
    elif(intKart[i] == 2):
        if(intKart[i+1] == 1):
            if(intKart[i+2] == 1):
                dirKart.append(1)
            else:
                dirKart.append(2)
        else:
            if(intKart[i+2] == 4):
                dirKart.append(1)
            else:
                dirKart.append(3)
    else:
        if(intKart[i+1] == 1):
            if(intKart[i+2] == 2):
                dirKart.append(3)
            else:
                dirKart.append(1)
        elif(intKart[i+1] == 2):
            dirKart.append(3)
        else:
            if(intKart[i+2] == 2):
                dirKart.append(1)
            else:
                dirKart.append(2)

#Dersom vi stopper i punkt G, som er ladestasjonen, gis instrukser som trengs for lading av bilen.
if(stopp == 'G'):
    dirKart.append(1)
```

```
            dirKart.append(5)

        #Dersom vi stopper i punkt H, som er parkeringsplassen, gis instruks som trengs for parkering
        if(stopp == 'H'):
            dirKart.append(6)

        return dirKart #returnerer listen med instruksjoner

#MQTT FUNKSJONER HER

# Alle callback for ulike situasjoner:
client.on_connect = on_connect
client.on_message = on_message
client.on_publish = on_publish
# Kobler seg til mqtt brokeren
client.connect(broker_address, port, 60)


#Er txt filen oppdatert?
def file_updated(file_path, initial_mod_time):
    current_mod_time = os.path.getmtime(file_path)
    return current_mod_time != initial_mod_time


#Sjekk txt filen om noe nytt er lagt til:
file_path = '/var/www/html/data.txt'
initial_mod_time = os.path.getmtime(file_path)

#All kode som skal loope på kjøre konstant inne i while løkka
while True:
    #mqtt loop
    #client.loop_forever()
    # Skjekker om filen har blitt oppdatert
    if file_updated(file_path, initial_mod_time):
        #client.loop_forever()
        initial_mod_time = os.path.getmtime(file_path)
        #print(f'Instruksjoner: {f()}')
        #publish til topic web2zumo
        print("Filen har blitt oppdatert")
        # Åpner filen og leser innholdet
        with open('/var/www/html/data.txt', 'r') as file:
            # Les fil og split
            contents = file.read()
            # henter innholdet i form av en liste, splitter den, første indeks blir start, andre blir
            splitcontents = contents.split(',')
            print(f'Nytt startpunkt {contents[0]}')
            print(f'Nytt sluttpunkt {contents[2]}')


        start = contents[0]
        stopp = contents[2]
        try:
            # Omformaterer listen som f() returnerer:
            msg_str = str(f())
            msg = ""
```

```
            for i in str(f()):
                if i == "[" or i == " " or i == "]" or i == ",":
                    msg = msg
                else:
                    msg += str(i)
                print(msg)
            # Publiserer meldingen på web2Zumo topicen
            pubMsg = client.publish(
            topic = 'web2Zumo',
            payload = msg.encode('utf-8'),
            qos = 0,
            )
            #pubMsg.wait_for_publish
            #if pubMsg.is_published():
            #    print("Instruksjoner er sendt")
        finally:
             print(f'LEVERT')
             # Tømmer msg variabelen
             msg = None



client.loop_forever()
```

```
***ESP32 MASTER***

#include <Wire.h>
#include <WiFi.h>
#include <PubSubClient.h>

// wifi og wifipassord
const char* ssid = "NTNU-IOT";
const char* password = "";

//Broker adresse
const char* mqtt_server = "10.25.18.138";

WiFiClient espClient;
PubSubClient client(espClient);
long lastMsg = 0;
char msg[50];
int value = 0;
char courseGlobal[]={};
byte courseLength;
int send, lastSent;

//Kilde: Jiteshsaini 2022
void setup_wifi() { //Setter opp wifi tilkobling
  delay(10);
  Serial.println();
  Serial.print("Kobler til: ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);
```

```
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi kobling opprettet");
  Serial.println("IP addresse: ");
  Serial.println(WiFi.localIP());
}

//Kilde: Jiteshsaini 2022
void callback(char* topic, byte* message, unsigned int length) { //Funksjon som kalles på når en meldi
  Serial.print("Melding ankommet topic: ");
  Serial.print(topic);
  Serial.print(". Melding: ");
  char courseArray[length+1]={};

  for (int i = 0; i < length; i++) {;
    courseArray[i] = (char)message[i];
    courseLength++;
  }
  courseArray[length] = '\0';
  for (int i = 0; i < length; i++) {
    int intValue = courseArray[i] - '0';  // Konverterer elemetene i courseArray til integers
    Serial.print(intValue);
    courseGlobal[i]=courseArray[i];
  }
  Serial.println();
}

//Kilde: Jiteshsaini 2022
void reconnect() { //Denne funksjonen kobler ESPen til MQTT
  client.subscribe("web2Zumo");
  // Looper til en kobling er opprettet
  while (!client.connected()) {
    Serial.print("Forsøker å opprette kobling til mqtt...");
    //Prøver å koble seg til
    if (client.connect("ESP32client", "njaal", "3Inshallah4")) {
      Serial.println("connected");
      // Topic som det subscribes til
      client.subscribe("web2Zumo");
    } else { //Om tilkobling mislykkes prøves det igjen etter 5 sekunder.
      Serial.print("mislykket kobling, rc=");
      Serial.print(client.state());
      Serial.println(" Prøver igjen om 5 sekund");
      delay(5000);
    }
  }
}



void setup()
{
```

```
  Wire.begin(); //Starter I2C kommunikasjon som master
  Serial.begin(115200);
  Serial.println("start");
  // mqtt settup
  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
}

byte x = 0;

void loop() {
  Wire.beginTransmission(1); // transmit to device #1
  for (int i=0; i < courseLength; i++){
    Wire.write(courseGlobal[i]);
    }
  Wire.endTransmission();    // stop transmitting
  courseLength = 0;
  Wire.requestFrom(1, 1);
  while(Wire.available() > 0) {
    int c = Wire.read();
    if ((c > 0) && c != lastSent) { //Sender kun data videre om verdien ikke er 0 eller samme data der
      Serial.println(c);
      send=c;
    }
  }
  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  if (send>0) { //Konverterer meldingen til et char array og publisher på car2Charge topicen.
    lastSent = send;
    char sendString[8];
    itoa(send, sendString, 10);
    Serial.print("Verdi som blir sendt:  ");
    Serial.println(sendString);

    client.publish("car2Charge", sendString);
    send = 0;
  }
}

***BILKODE***

//Antatt forbruk 0.2kWh per km, 400 km rekkevidde. Maks batteri er 80 kWh.+

#include <Wire.h>
#include <Zumo32U4.h>
#include <EEPROM.h>
```

```
/*
*****************************************************************************************************
*************************************************|----------|*************************************
*************************************************|LES README|*************************************
*************************************************|----------|*************************************
*****************************************************************************************************
*/
Zumo32U4Motors motors;
Zumo32U4ButtonC buttonC;
Zumo32U4LineSensors lineSensors;
Zumo32U4OLED display;
Zumo32U4Encoders encoder;

bool pidFlag = true;  //For å kunne tvinge linjefølging av
byte power, input, courseArrlength;
float disGlobal;
unsigned long distance;
int courseArray[30] = {};
bool distSend = false;
static int drip[5];  //Trengs for å kunne lese av spesifik sensor

int rightSpeed = 200;
int leftSpeed = 200;
int previousError;
float output;
double integral;
double derivative;
unsigned int lineSensorValues[5];

float distMeasure() { //Måler avstand kjørt, 1m kjørt tilsvarer 10km i simuleringen.
  int currRotLeft = encoder.getCountsAndResetLeft();
  int currRotRight = encoder.getCountsAndResetRight(); //Teller motorrotasjoner og resetter tellingen
  float leftDist = ((abs(currRotLeft)) * 3.1415 * 0.039) / 910;
  float rightDist = ((abs(currRotRight)) * 3.1415 * 0.039) / 910; //Konverterer motorrotasjoner til me
  float distPart = (10 * (leftDist + rightDist) / 2); //Tar gjennomsnittet av høyre og venstredistanse
  return distPart;
}



int batteryDrain() {    //Genererer batterinivået, som er mellom 0 og 80
  int battery = 80 - (disGlobal / 5);
  if (battery < 0) {
    battery = 0;
  }
  return battery;
}



void showBattery() {   //Viser batterinivå og avstand kjørt på displayet
  display.gotoXY(0, 0);
  display.print("Power:  ");
  display.gotoXY(0, 1);
  display.print(power);
```

```
  display.gotoXY(0, 3);
  display.println("Distance drove; ");
  display.gotoXY(0, 4);
  display.print(disGlobal);
  display.print("km     ");
}



void Receive(int howMany) {   //Tolker meldinger fra ESP og konverterer fra char array til int
  static bool startRouteFlag = true;
  while (0 < Wire.available())  //Looper gjennom data mottatt
  {
    byte receivedByte = Wire.read();
    courseArray[courseArrlength] = receivedByte - '0';  //Konverterer fra string til integer
    courseArrlength++;
    if (startRouteFlag) {
      input = courseArray[0];
      startRouteFlag = false;
    }
  }
}

void Charge() {     //Viser i displayet at bilen lader
  motors.setSpeeds(0, 0);
  display.clear();
  display.println("CHARGING");
}



void sendCharge() {   //Sender antall kWh ladet til ESP, sendes en gang når bilen lader.
  if (distSend == true) {
    int kWhCharged = 0.2 * disGlobal;
    Wire.write(kWhCharged);
    disGlobal = 0;  //Nullstiller avstand kjørt.
    distSend = false; //Sørger for at avstanden blir kun sendt en gang hver gang den lader.
  }
}

//Kilde: .....
void lineFollowPID() {  // Bilens linjefølgingskode. Vil følge en teip som representerer veien.
  static short prevPos;
  if (pidFlag) {
    int posisjon = lineSensors.readLine(lineSensorValues);
    int error = (posisjon - 2000) / 5;
    integral += error;
    derivative = error - previousError;
    previousError = error;
    output = error + 0.0001 * integral + 4 * derivative;
    motors.setSpeeds(leftSpeed + output, rightSpeed - output);
  }
}

void drivingMain() {      //Konverterer rutedata til bilens kjøremønster.
  static byte turnCount = 0;
```

```
switch (input) {
  case 1:  //Høyresving
    showBattery();
    static bool rightFlag = false;
    static uint32_t rightTime = millis();
    if (lineSensors.readOneSens(drip) >= 700) {  //Om bilen har kommet til et kryss vil den svinge t
      rightTime = millis();
      motors.setSpeeds(150, -100);
      rightFlag = true;
    }
    if (millis() - rightTime >= 350 && rightFlag) {  //Om bilen har fullført svingen hopper bilen ti
      input = 4;
      rightFlag = false;
      break;
    } else if (!rightFlag) lineFollowPID();  //Kjører linjefølging om ingen sving
    break;

  case 2:  //Rett frem
    static bool straightFlag = false;
    static byte straightCounter = 0;
    lineFollowPID();
    showBattery();
    if (lineSensors.readOneSens(drip) >= 700) straightFlag = true;    //Merker at den har kommet på
    else if (lineSensors.readOneSens(drip) <= 150 && straightFlag) {  //Teller + 1 etter bilen har p
      straightCounter++;
      straightFlag = false;
    }
    if (straightCounter >= 2) {  //Om den har pasert to linjer går den til case 4
      straightCounter = 0;
      input = 4;
      break;
    }
    break;
  case 3:  //Venstresving
    static bool leftFlag = false;
    static bool leftFlag2 = true;
    static byte leftCounter = 0;
    static uint32_t leftTime = millis();
    showBattery();
    lineFollowPID();
    if (lineSensors.readOneSens(drip) >= 700) {  //Merker at den rører en linje og setter av et flag
      leftFlag = true;
    }
    else if (lineSensors.readOneSens(drip) < 100 && leftFlag) {  //Når bilen har gått av linjen flip
      leftCounter++;
      leftFlag = false;
    }

    if (lineSensors.readOneSens(drip) >= 700 && leftCounter == 1) {  //Når bilen kommer til en linje
      motors.setSpeeds(-100, 150);
      leftTime = millis();
      leftFlag2 = false;
      pidFlag = false;  //Skrur av linjefølging
      leftCounter++;
    }
```

```
    if (leftFlag2 == false && millis() - leftTime >= 700) {  //Avslutter svingen og skrur på linjefø
      leftFlag2 = true;
      pidFlag = true;
    }
    if (leftCounter >= 4) {  //Resetter counter og fullfører svingen etter bilen er ute av kryset
      leftCounter = 0;
      input = 4;
      break;
    }
    break;
  case 4:  //Iterer til neste case i rutedataen
    static bool switcher = true;
    static uint32_t switcherTime = millis();
    lineFollowPID();
    showBattery();
    if (switcher) {
      switcherTime = millis();
      switcher = false;
    }
    if (millis() - switcherTime >= 300) {
      turnCount++;
      switcher = true;
      input = courseArray[turnCount];
      break;
    }
    break;
  case 5:  //Lading
    static uint32_t chargeEndTime = millis();
    static bool chargeEndFlag, chargeSendFlag = true;
    if (lineSensors.readOneSens(drip) >= 700) { //Når bilen merker teip på venstre sensor stopper de
      Charge();
      if (chargeSendFlag == true) { //Lar sendChargefunksjonen sende strøm ladet for én gang.
        distSend = true;
        chargeSendFlag = false;
      }
    } else { //Om bilen ikke merker teip på venstre side kjøres vanlig linjefølging.
      lineFollowPID();
      showBattery();
    }
    if ((turnCount + 1) != courseArrlength) { //Om bilen blir sendt ny rute, vil den først kjøre ut
      if (chargeEndFlag) {
        chargeEndTime = millis();
        chargeEndFlag = false;
      } else if (millis() - chargeEndTime >= 5000) {
        chargeEndFlag = true;
        chargeSendFlag = true;
        input = 4;
        break;
      }
    }
    break;
  case 6: //Parkere
    if (lineSensors.readOneSens(drip) >= 700) motors.setSpeeds(0,0); //Om bilen merker teip på venst
    else lineFollowPID();
    if((turnCount + 1 != courseArrlength)) input = 4;
```

```
      break;
    default:
      showBattery();
      motors.setSpeeds(0, 0);
      if (turnCount != courseArrlength) {
        input = courseArray[turnCount];
      }
      break;
  }
}


//Kilde: .....
void pidSetup() {   //Kalibrerer bilen til å følge teipen.
  lineSensors.initFiveSensors();
  bool startFlag = true;
  buttonC.waitForPress();
  motors.setSpeeds(100, -100);
  uint32_t startTime = millis();
  while (startFlag) {
    lineSensors.calibrate();
    if (millis() - startTime >= 4000) startFlag = false;
  }
  motors.setSpeeds(0, 0);
}

//Main

void setup() {
  Wire.begin(1); //Oppretter I2C kommunikasjon med ESP32.
  Wire.onRequest(sendCharge);
  Wire.onReceive(Receive);
  display.setLayout21x8();
  //EEPROM.write(1, 0);  //Kan kommentere inn denne linjen om vi vil starte bilen på full lading.
  disGlobal = EEPROM.read(1);
  pidSetup(); //Bilen vil ikke gå ut av setup før den er kalibrert.
}

void loop() {
  disGlobal += distMeasure();
  EEPROM.write(1, disGlobal);
  power = batteryDrain();
  drivingMain();
}
```
