Напишемо сервер, який отримує данні від джерела данних, зберігає та відправляє клієнту, та також по необхідності завантажує та відправляє данні клієнту:

```
'use strict';
// завантаження бібліотек
const fs = require('fs');
const http = require('http');
const WebSocket = require('ws');
//завантажуємо сторінку
const index = fs.readFileSync('index.html');
// Завантажуємо модуль для побудови графіку
const jsFile = fs.readFileSync('sm.js');
// На любий http запит віддаємо 200 та сторінку
const server = http.createServer((req, res) => {


  // на запит від клієнту
  switch (req.url) {
    // на запит модулю
    case "/sm.js" :
    // відправляємо модуль
    res.writeHead(200, {"Content-Type": "text/javascript"});
    res.write(jsFile);
    break;
    // на інші
    default :
    //передаємо сторінку
      res.writeHead(200, {"Content-Type": "text/html"});
      res.write(index);
  }

  res.end();
});

// підключаємо на порт 8000
server.listen(8000, () => {
  console.log('Listen port 8000');
});

// Запускаємо веб сокет, передаємо в нього змінну hhtp server
const ws = new WebSocket.Server({ server });
// на подію on conection(коли отримує запит на сокет)
// отримуємо сокет conection, який має додаткові методи
```

```
ws.on('connection', (connection, req) => {
  // отримуємо remote adress
  // відправляємо клієнту отримані повідомлення
    const ip = req.socket.remoteAddress;
    console.log(`Connected ${ip}`);
    connection.on('message', message => {
      console.log('Received: ' + message);
      for (const client of ws.clients) {
        if (client.readyState !== WebSocket.OPEN) continue;
        //if (client === connection) continue;
        client.send(message);
        //якщо на сервер прийшов код команди на завантаження
        if (message ==123456){
        // завантажуємо лог файли та відсилаємо клієнту
        let fileContent = fs.readFileSync("log1.txt", "utf8");
        let fileContent2 = fs.readFileSync("log2.txt", "utf8");
        client.send(fileContent)
        client.send(fileContent2)
      }
      // в іншому випадку, кожен монітор по id зберігаємо в свій лог файл
      var arrr = JSON.parse(message);
      var idd = arrr.id;
      if (idd === 1){
        fs.writeFileSync("log1.txt",message)
      }
      if (idd === 2){
        fs.writeFileSync("log2.txt",message)
      }
      }
    });


  connection.on('close', () => {
  console.log(`Disconnected ${ip}`);

  });
});
```

Напишемо клієнт, який отримує данні від сервера, будує графіки в залежності від монітору, який надіслав данні, також відправляє команду на завантаження данних від сервера по необхідності.

```
 <!DOCTYPE html>
<html>
  <head>
    <title>ECG</title>
  </head>
```

```
<body>
<h1 align="center">Cardiomonitors</h1>
<p><input type="button" value="load data" onclick="onpress()"></p>
<div id="pat1" style="height: 300px; width: 100%;"></div>
<div id="pat2" style="height: 300px; width: 100%;"></div>
<script src="https://canvasjs.com/assets/script/jquery-1.11.1.min.js"></script>
<script src="https://canvasjs.com/assets/script/jquery.canvasjs.min.js"></script>

<!-- Створюємо об'єкт, який виводить повідомлення-->
<div id="chat"></div>
<!-- Створюємо об'єкт, в який вписуємо текст-->
<input id="msg" type="text">
<style>
input { border: 1px solid green; }
</style>



<script>
  const CHAR_RETURN = 13;
  //url веб сокет серверу
  const socket = new WebSocket('ws://localhost:8000/');
  console.log(socket)
  //беремо ссилки на HTML елементи
  const chat = document.getElementById('chat');
  const msg = document.getElementById('msg');
  //автоматично натискає на поле мемедж
  msg.focus();

  // функція writeline
  const writeLine = text => {
    //створюємо новий елемент
    const line = document.createElement('div');
    // встявляємо в блок вмісткість
    line.innerHTML = `<p>${text}</p>`;
    // додаємо новий елемент
    chat.appendChild(line);
  };
  //події
  socket.onopen = () => {
    writeLine('connected');
  };

  socket.onclose = () => {
    writeLine('closed');
```

```
  };

// змінні лічильників
  var datacounter1 = 0
  var datacounter2 = 0
  var dataPoints1 = []
  var dataPoints2 = []
  socket.onmessage = event => {
    //writeLine(event.data);

      var dt = event.data
      var arrr = JSON.parse(dt);
      var idd = arrr.id;
      var dtt = arrr.DATA;
    // кожний клієнт відсилає данні з унікальним Id по цьому id відбувається визнач
ення
    // до якого монітору належить сигнал
    // якщо перший монітор:
          if(idd === 1) {


      //-------------Графіки----------------//
    var chart = new CanvasJS.Chart("pat1",
    {
      title:{
      text: "monitor1"
      },
      data: [
      {
        type: "line",
        dataPoints: dataPoints1
      }
      ]
    });
    //створюємо масив точок
    for(var i = 0;i<827;i++){
      dataPoints1.push({x:datacounter1, y: dtt[i]})
      //додаємо відлік часу перерахований в секунди по частоті дискретизації
      datacounter1 += (1/100)
    }
    //якщо відмальовано більше 100 секунд
    if (datacounter1 > 100){
      //обнуляємо вектори
      datacounter1 = 0
      dataPoints1 = []
    }
    //будуємо графік
```

```
    chart.render();
      }

      //для 2 монітору
      else if (idd === 2) {
    var chart = new CanvasJS.Chart("pat2",
    {
      title:{
      text: "monitor2"
      },
      data: [
      {
        type: "line",
        dataPoints: dataPoints2
      }
      ]
    });
    for(var i = 0;i<827;i++){
      dataPoints2.push({x:datacounter2, y: dtt[i]})
      datacounter2 += (1/100)
    }
    if (datacounter2 > 100){
      datacounter2 = 0
      dataPoints2 = []
    }
    chart.render();
  }


  };

  //по натисканні на кнопку load
  // на сервер відправляється код команди
function onpress() {
  socket.send(123456);
}

  // обробник натискання колавші ентер(компонент для ручного вводу точок)
  msg.addEventListener('keydown', event => {
      // якщо ентер
    if (event.keyCode === CHAR_RETURN) {
      // берему поточне речення
        const s = msg.value;
        //обнуляємо щоб перевід строки не попав в строку
      msg.value = '';
      //виводимо на екран
      writeLine(s);
```

```
      //відправляємо в сокет
      socket.send(s);
    }
  });

</script>
<script type="text/javascript" src="https://canvasjs.com/assets/script/canvasjs.min
.js"></script>
</body>
</html>
```

Напишемо два клієнта на пайтоні, які симулюють роботу двох кардіо моніторів, та відправляють на сервер данні з власним ідентифікатором

Клієнт 1:

```
import asyncio
import websockets
import json

Fs = 1000
#сигнал
sig = [-0.04736953792203903,-0.04811678798757516, -0.047932619796921414, -
0.04778090832472873, -0.04764157992641114, -0.04752170786394537, -
0.047437059532571095, -0.04738578525056236, -0.04735619719514137, -
0.047345183115303516, -0.04735323021429148, -0.04736953792203903]
d = 1
async def monitor1():
    #під'єднання
    uri = "ws://localhost:8000"
    async with websockets.connect(uri) as websocket:
        while True:
            #Кожну секунду надсилаємо данні
            blog = {'DATA': sig, 'id': 1}

            to_json = json.dumps(blog)

            await websocket.send(to_json)
            await asyncio.sleep(d)
            #відправляємо у веб сокет

        greeting = await websocket.recv()
        print(f"< {greeting}")

asyncio.get_event_loop().run_until_complete(monitor1())
```

Клієнт 2:

```
async def monitor2():
    #під'єднання
    uri = "ws://localhost:8000"
    async with websockets.connect(uri) as websocket:
        while True:
            #Кожну секунду надсилаємо данні
```

```
            blog2 = {'DATA': sig, 'id': 2}

            to_json2 = json.dumps(blog2)

            await asyncio.sleep(d)
            #відправляємо у веб сокет
            await websocket.send(to_json2)
            await asyncio.sleep(d)
        greeting = await websocket.recv()
        print(f"< {greeting}")

asyncio.get_event_loop().run_until_complete(monitor1())
```

Тестування роботи програми:

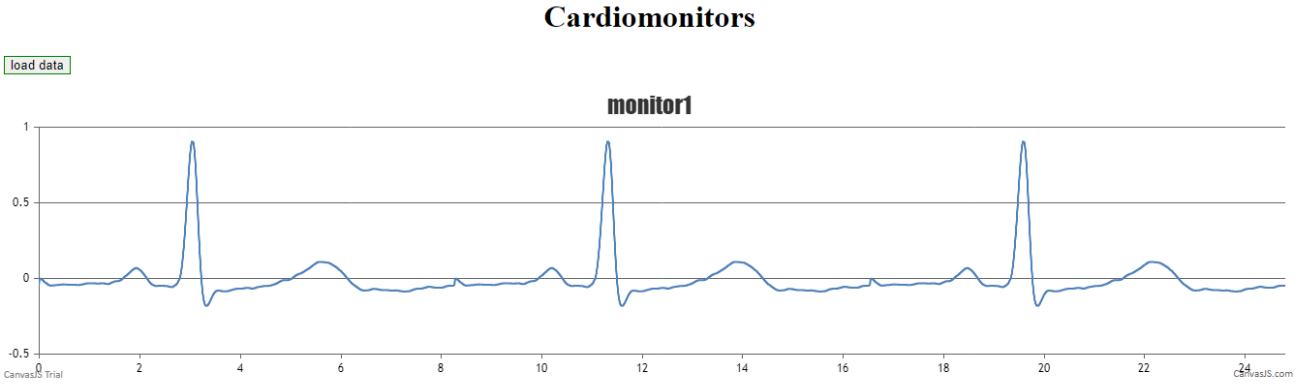
1) Запускаємо сервер:

```
PS C:\project> node server.js
Listen port 8000
```

2) Під'єднуємося з веб клієнту за посиланням: ws://localhost:8000/

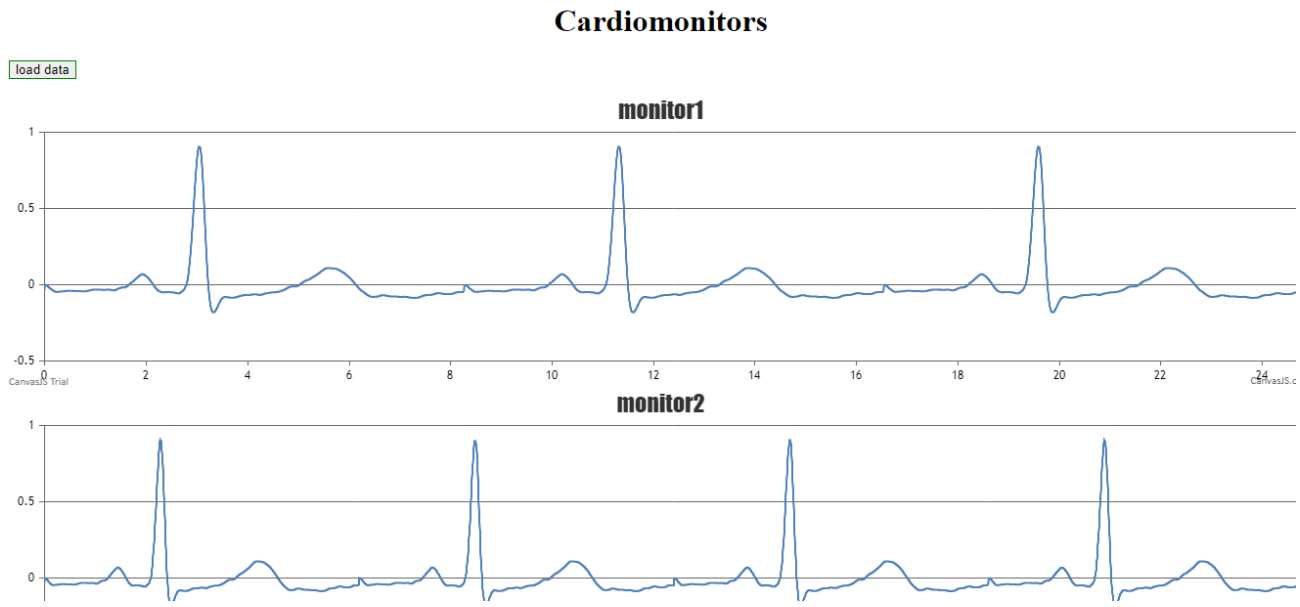
Отримуємо пусте вікно з кнопкою 'Завантажити'

3) запускаємо перший монітор



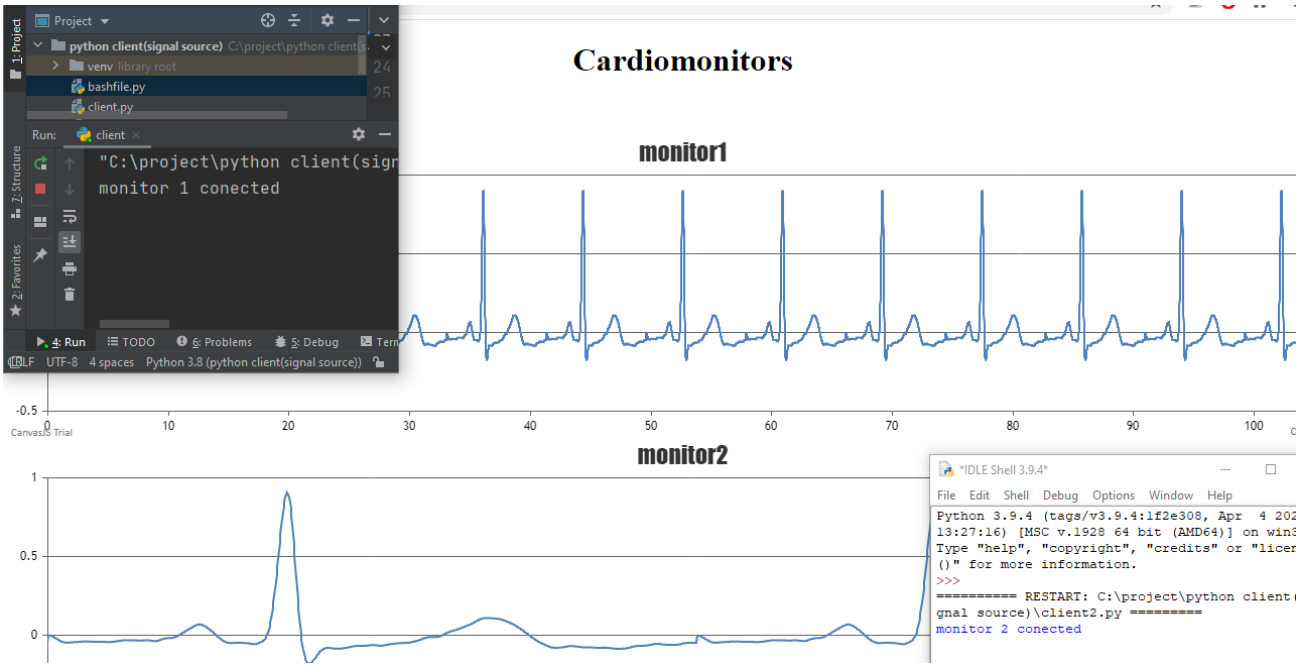
З'являється вікно першого монітору, та відображається кардіосигнал

4) Запускаємо другий монітор



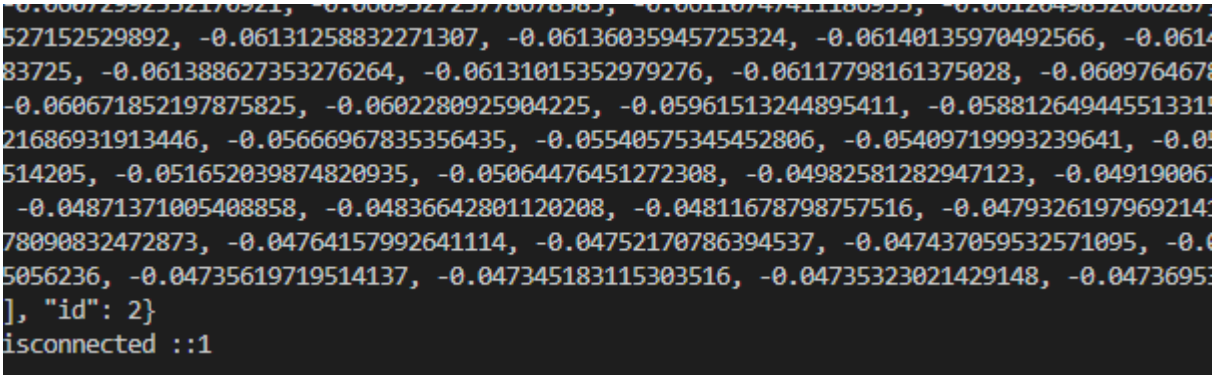
З'являється вікно другого сигналу, і відображається сигнал

5) Запустимо обидва монітори одразу

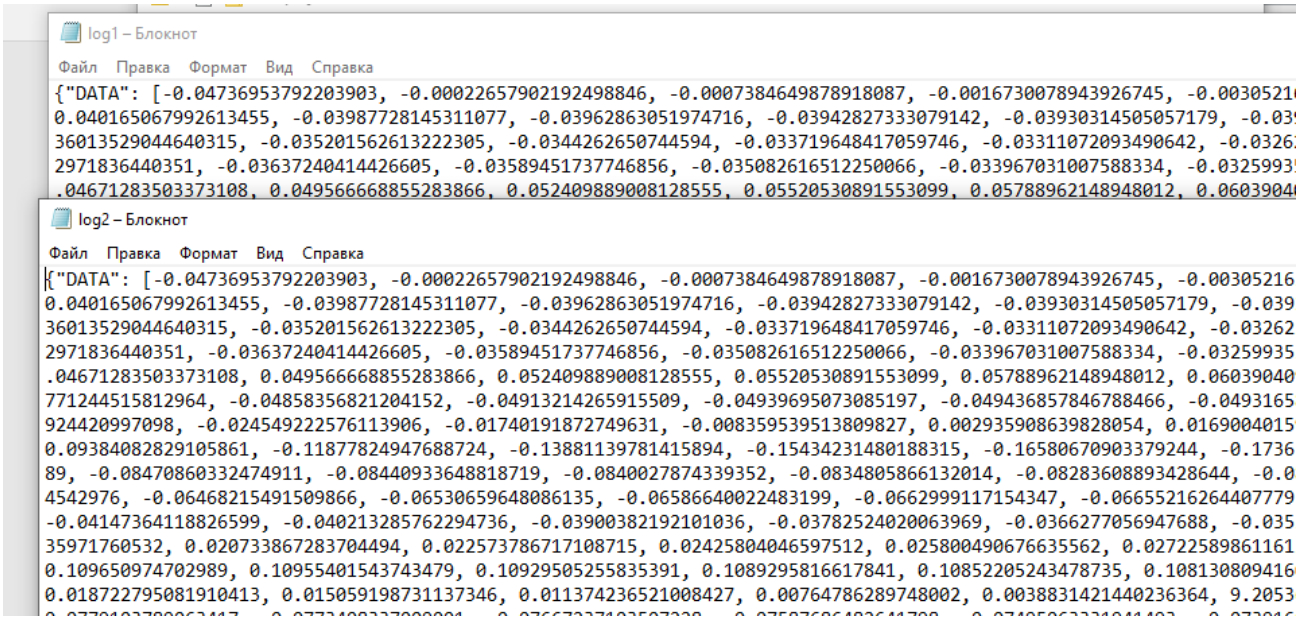


Бачимо данні надсилаються коректно

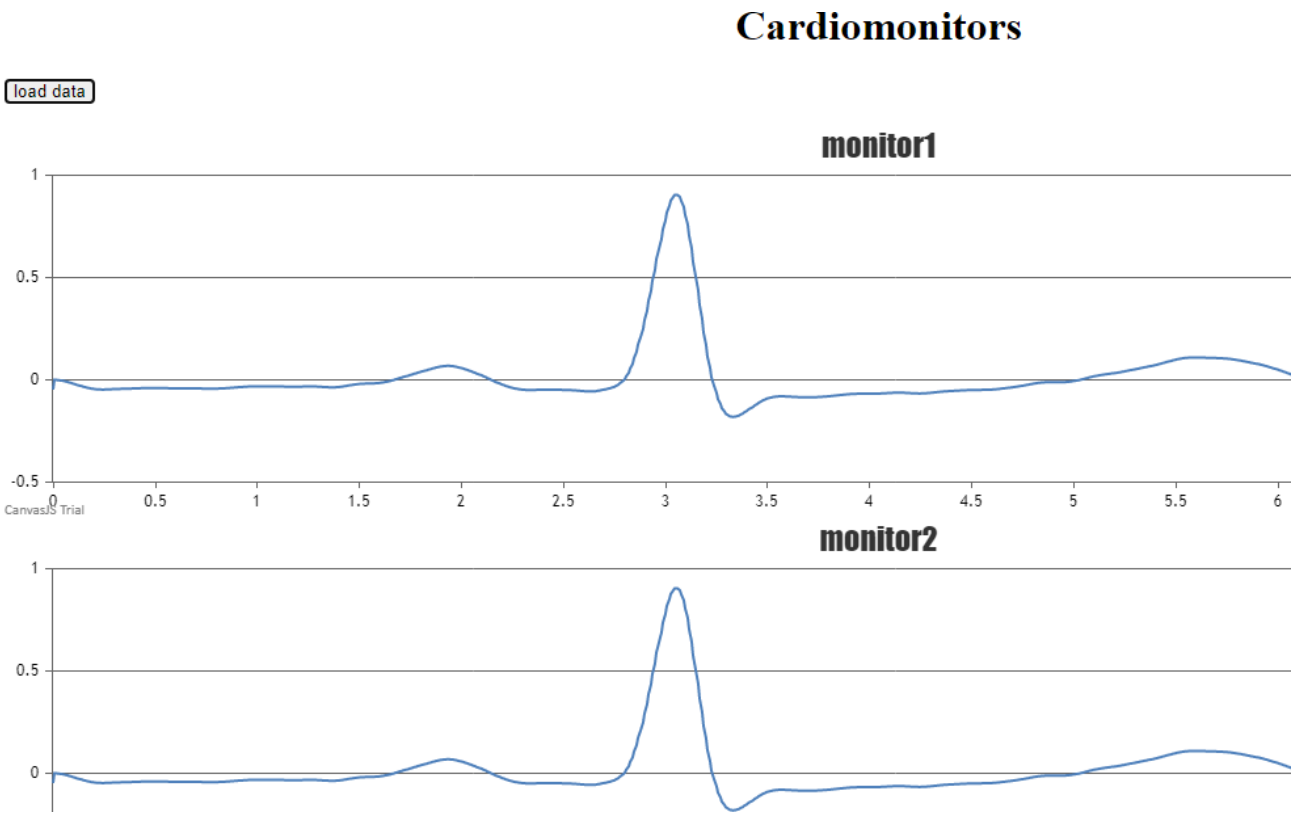
На сервері бачимо лог:



6) Отримали файли логу



7) оновимо сторінку і завантажимо лог файли



При натисканні на кнопку load data повернули два збережених сигнали