Mobile Application development 2022
Research assignment

12.08.2022

TTOW0615

Timo Laakkonen

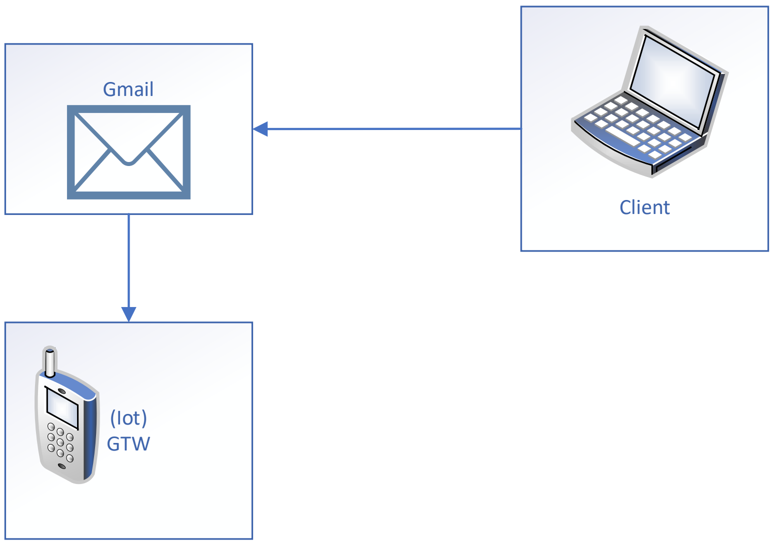
Sisällysluettelo

1 The main idea

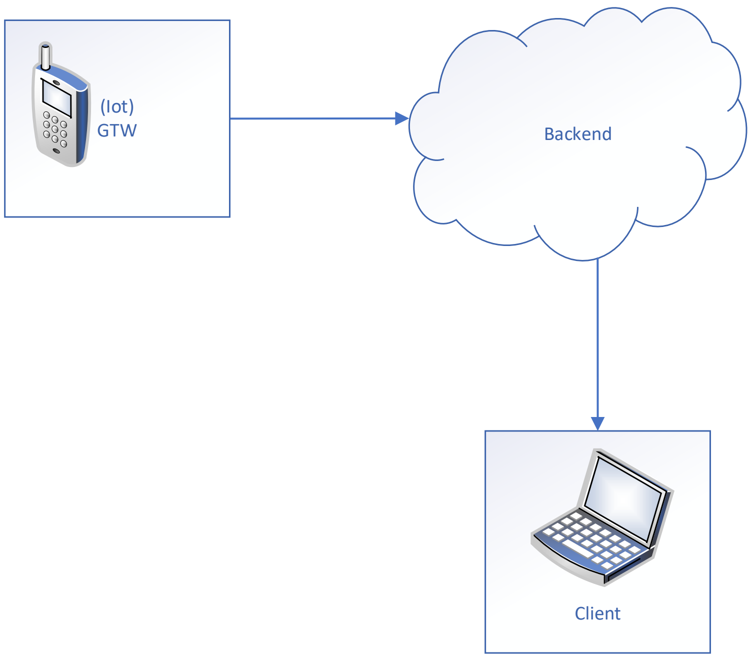
The mobile device works like iot (internet of things) or gateway, it is like service. The sensors can be connected to device e.g. humidity, pressure, temperature, lux, winspeed. The client can take contact to device by email and will get response (sensor data) to webbrowser's page which the email with query was sent. Unfortunately because the sensors were not available for this exercise, the device will be send as an example it's location, ip-address, socketid, battery level, usedmemory, totalmemory as an response for the client request.
In this exercise the working flow will be presented as developing mode.

1.1 Communication

The client sends an email to chosen mobile device's address, mobile device reads inbox continuesly.

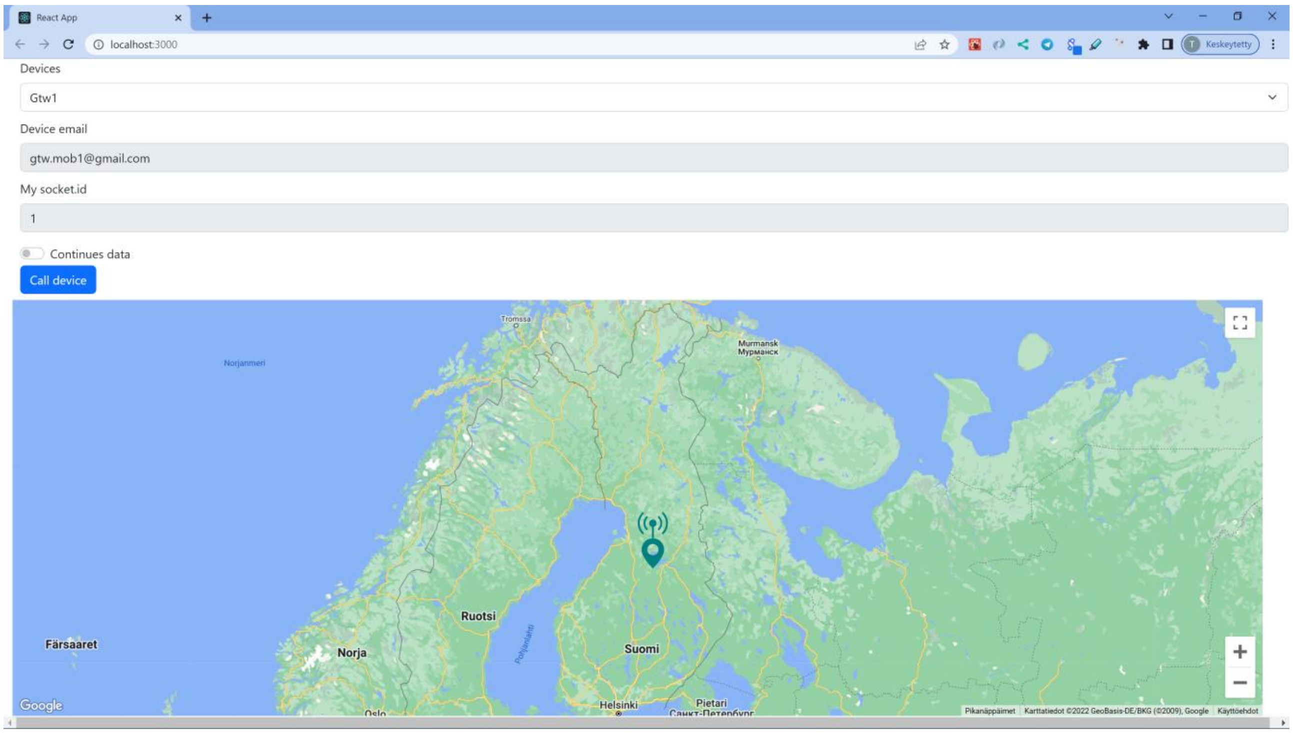


The mobile device responses and sends data for the client's request.



2 Using the demo

The client application opens to the following view



Devices

Gtw1

Device email

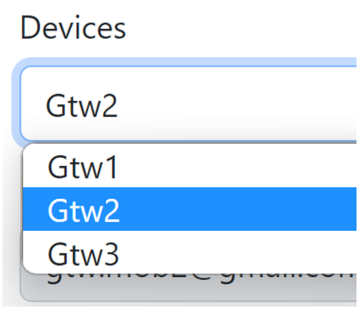
gtw.mob1@gmail.com

My socket.id

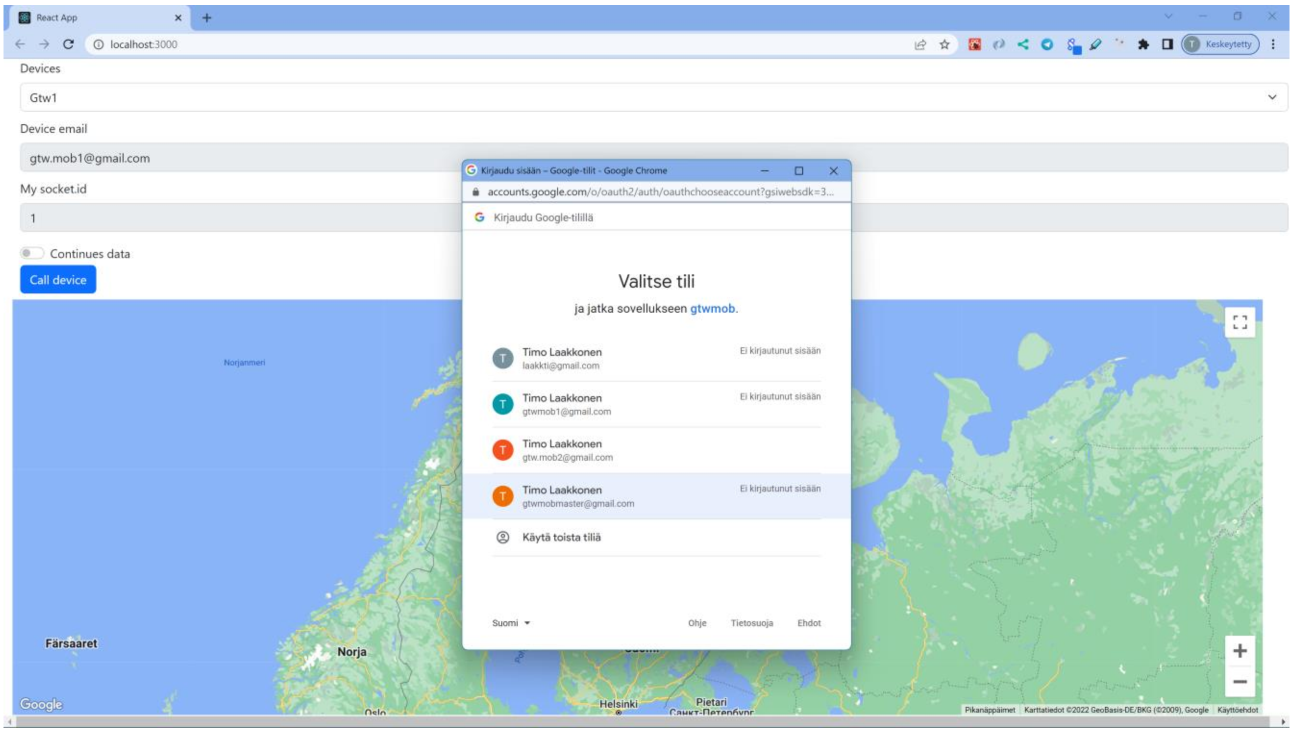
1

Continues data

The device whose data you want to view can be selected from the dropdown menu



Then pressing/tabbing “Call device”-button the data/info from the chosen device will be queried. But before the response/results you must login to google account, this service is under test mode. This demonstration has only one account having permission to use google api. In the future there coud be registration for this service so users can be added to list of permission.


React App
localhost:3000

Devices
Gtw1

Device email
gtw.mob1@gmail.com

My socket.id
1

Continues data

Call device

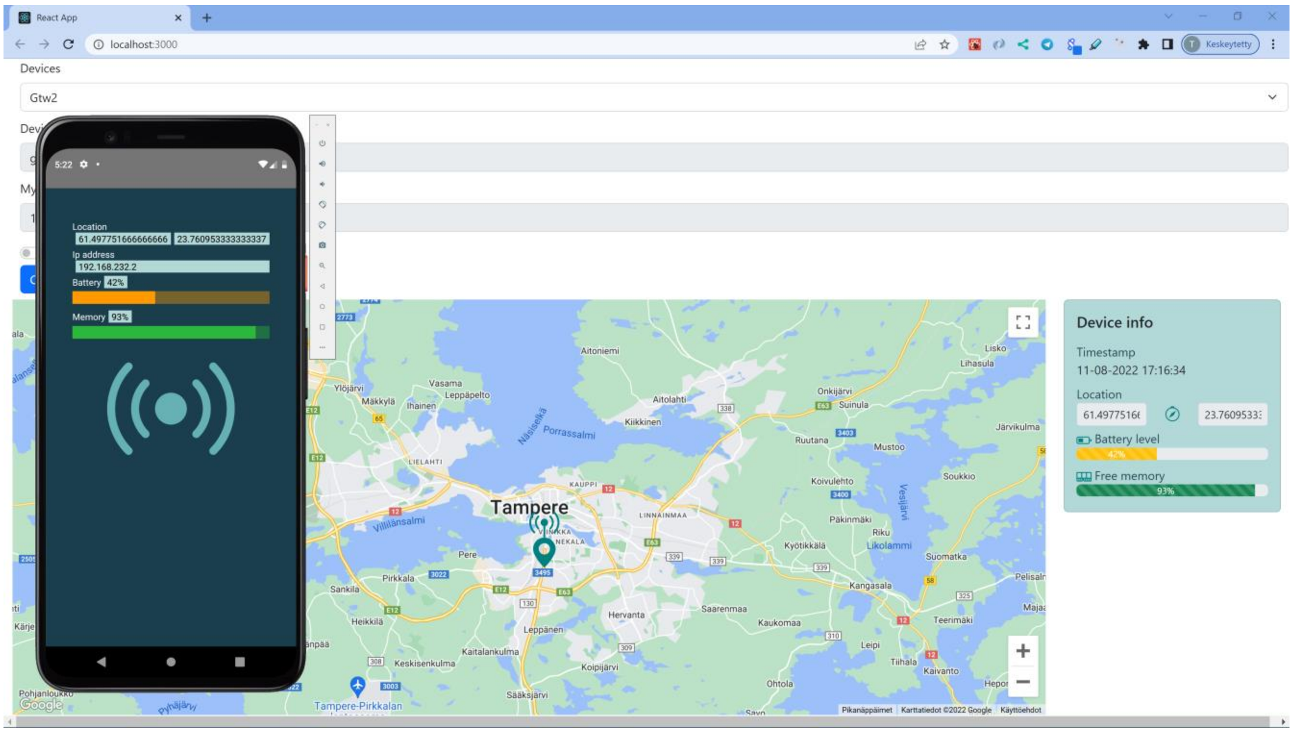
Kirjaudu sisään – Google-tilit - Google Chrome
accounts.google.com/o/oauth2/auth/oauthchooseaccount?gsiwebsdk=3...
Kirjaudu Google-tilillä

Valitse tili
ja jatka sovellukseen gtwmob.

Timo Laakkonen laakkti@gmail.com Ei kirjautunut sisään
Timo Laakkonen gtwmob1@gmail.com Ei kirjautunut sisään
Timo Laakkonen gtw.mob2@gmail.com
Timo Laakkonen gtwmobmaster@gmail.com Ei kirjautunut sisään
Käytä toista tiliä

Suomi Ohje Tietosuoja Ehdot

The mobile device responses with data, which is shown on the client’s screen. For the purpose of demonstration, the information is displayed on the screen of mobile devices as well (in practice, there is no screen at all).


React App
localhost:3000

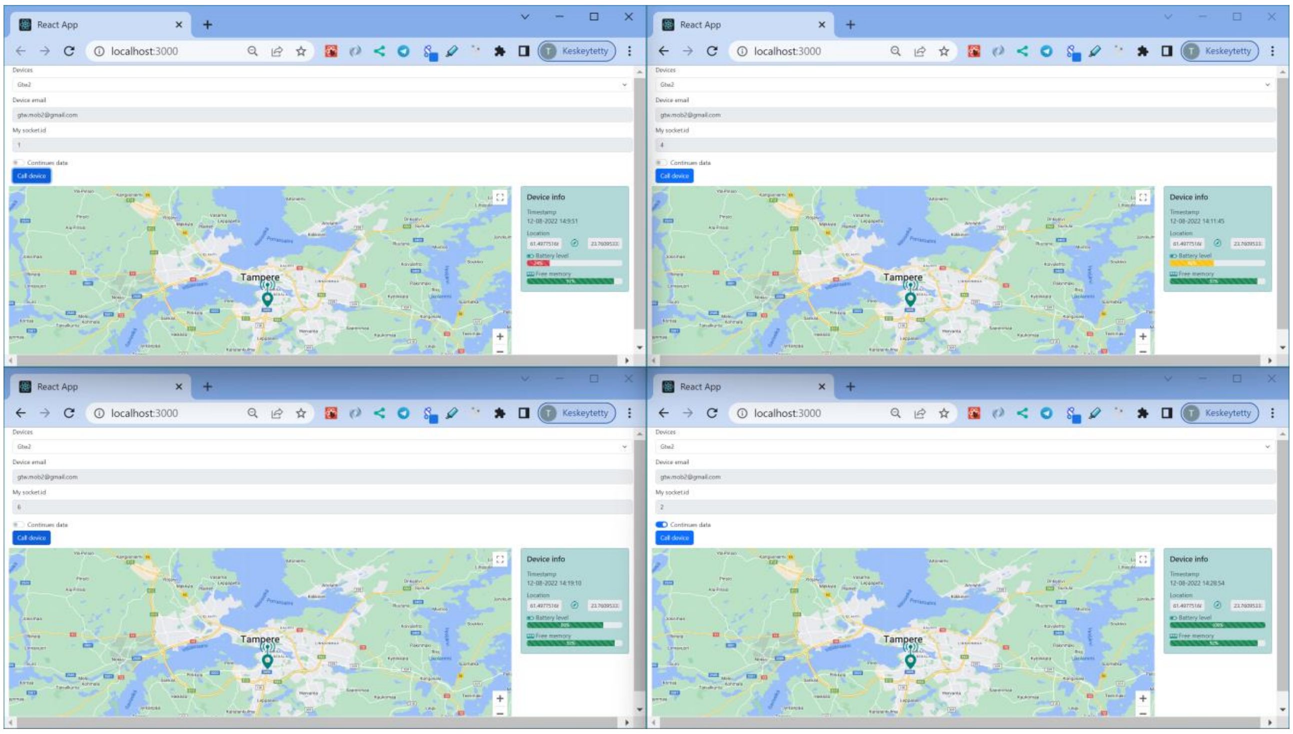
Devices
Gtw2

5:22

Location
61.497751666666666 23.760953333333337
Ip address
192.168.232.2
Battery 42%
Memory 93%

Device info
Timestamp
11-08-2022 17:16:34
Location
61.4977516(23.7609533:
Battery level
42%
Free memory
93%

In the next picture you can see four clients has taken a contact to mobile device by email and has got the response via websocket server. As can be seen every client has own sockedId, which makes it possible to get response to the right address. The device is located in Tampere, nothing but time and battery level has changed during contacts of the clients. So the mobile app serves fine.



Devices: Gtw2
Device email: gtw.mob2@gmail.com
My socket id: 1
Continues data
Device info
Timestamp: 12-08-2022 14:9:51
Location: 61.4977516 23.7609533
Battery level
Free memory

Devices: Gtw2
Device email: gtw.mob2@gmail.com
My socket id: 4
Continues data
Device info
Timestamp: 12-08-2022 14:11:45
Location: 61.4977516 23.7609533
Battery level
Free memory

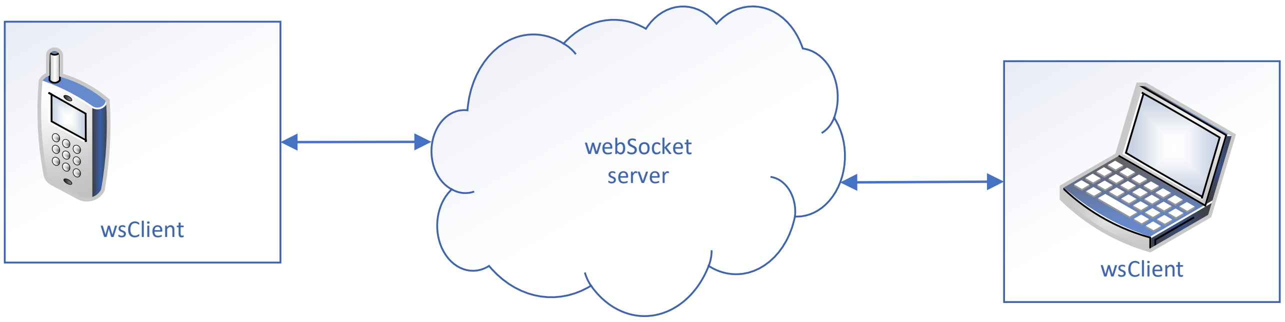
Devices: Gtw2
Device email: gtw.mob2@gmail.com
My socket id: 6
Continues data
Device info
Timestamp: 12-08-2022 14:19:10
Location: 61.4977516 23.7609533
Battery level
Free memory

Devices: Gtw2
Device email: gtw.mob2@gmail.com
My socket id: 2
Continues data
Device info
Timestamp: 12-08-2022 14:28:54
Location: 61.4977516 23.7609533
Battery level
Free memory

3 About the technique

3.1 Websocket (connection between devices)

Websocket protocol connection is straightforward and faster (almost realtime) and easier to use than e.g. http(s).



When the client has launched it creates connection with websocket server, which responses with id for the client. Websocket server saves client sockets in an array and adds unique id property to every socket object, which will be send to the client’s websocket.

3.2 Gmail api

To use gmail api for reading and sending the message it is needed to be signedin and before signin there must do some setting in
https://console.cloud.google.com/apis/dashboard?project=gtwmob
e.g. enabling apis & services and creating credientals API keys and OAuth 2.0 Client IDs.

In mobile device web protocol http is used for reading messages and token is needed for header every single requests (reading and sending messages). For client app javascript & library approach is used.

3.3 Mobile device

Is coded using React Native and some libraries.

The only fixed connection/interface outside of the device is a gmail account, so the first contact to the mobile device is via email. Every mobile device has unique gmail-address.

The temporary connection is a websocket-connection to websocket server for the address which is mentioned in the message read.

Axios is used for communicating with gmail web/http application interface.

Device reads the oldest unread inbox message having subject label “query” and marks it as read.

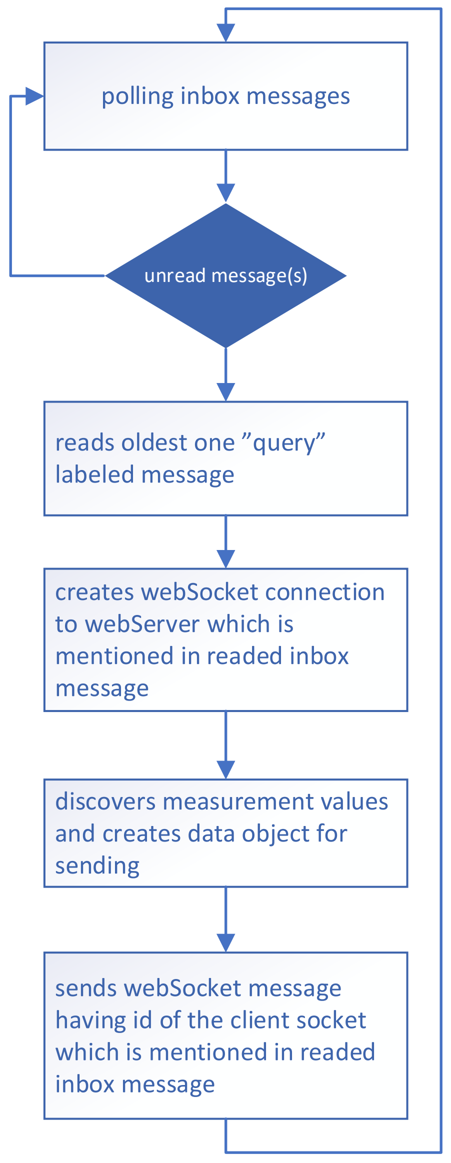
The content of message is in json -format so it is easily parsed as an object. The data includes client socket id and socket server ip address.

Signin to google api is necessary and the token is needed for header part of every message/query to google api. For getting token, some settings are needed (in paragraph 3.2 Gmail api).

3.3.1 Main loop/sequence

The application will be running/active all the time. UseEffect hook is used instead of timer. The functionality is based on dependency variable value change which causes re-rendering. Async/await has been used.

Flowchart of workflow


polling inbox messages
unread message(s)
reads oldest one ”query” labeled message
creates webSocket connection to webServer which is mentioned in readed inbox message
discovers measurement values and creates data object for sending
sends webSocket message having id of the client socket which is mentioned in readed inbox message

step 1

device reads continuesly gmail inbox until unread message having subject title “query” is found, if there is several messages the oldest one is chosen for reading and handling.
if there is none, just continue “polling and waiting”…

step 2

when an unread message is discovered it will be read and marked as read. Content of the message is base64 encoded so it have to be decode.
The decoded content is in JSON format so it’s very easy to convert to object so the data can be handled.

step 3
next the message's id-value is used to set state variable to webserver address and useEffect hook will be launched to create webSocket connection to webSocket server, the ip-address was given in email message.

step 4
next the device will discover it's geolocation, ip-address, memory values and battery level. The message to send via webSockert server to client is in following form

```
const message = {
      sender: 'mob',
      location,
      id: socketId,
      ipAddress: ipAddress,
      batteryLevel: batteryLevel,
      usedMemory: usedMemory,
      totalMemory: totalMemory,
      time: Date.now(),
    };
```

The parameter sender is for client so it will know who sent the message, when server creates socket the message is sent to client the parameter is "server" and when mobile device sends message parameter has value "mob" so client knows how to handle messages.

step5

device sends websocket message/data to the server.
Before sending data via websocket data object is converted as json-string.

3.3.2 Package dependencies

```
  "dependencies": {
    "@react-native-community/geolocation": "^2.1.0",
    "@react-native-google-signin/google-signin": "^7.2.2",
    "axios": "^0.27.2",
    "react-native-base64": "^0.2.1",
    "react-native-device-info": "^10.0.2",
    "react-native-elements": "^3.4.2",
    "react-native-svg": "^12.4.3",
    "react-native-svg-transformer": "^1.0.0"
  }
```

@react-native-community/geolocation
Getting location of the device

@react-native-google-signin/google-signin
Login into google services apis.

Axios
For communicating with gmail, getting messages, reading messages and mark as read

react-native-base64
for email message to base64 decode

react-native-device-info
for getting information of device such batterylevel, ip-address, memory…

react-native-elements
some user interface issues/elements

react-native-svg & react-native-svg-transformer
To show svg-type of images.

3.3.3 Files

App.js
Includes main code

Geolocation.js
getting geolocation of the device

Deviceinfo.js
Getting information of the device, in this app the interests are ipAddress, batteryLevel, usedMemory, totalMemory

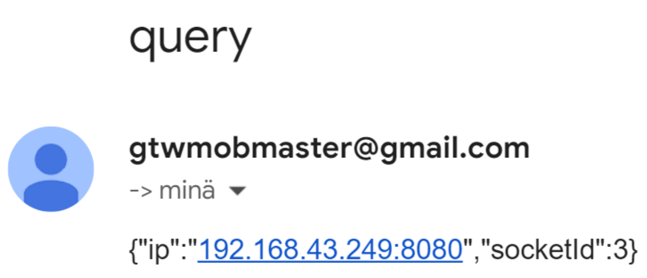
3.4 Client

In this demostration the client is wep app, it could be mobile app as well but for presentation web app is more reasonable to arrange.

The client is based on React, which is frontend javascript library. User interface has been powered by react bootstrap.

At the first thing when app is launched it takes websocket contact to backend and get an id which will be used to recognize device when comminucating via websocket to another device/client. The email which will be sended has subject “query”, The content of the message is in JSON format having values for server ip and socket id.

{"ip":"192.168.43.249:8080","socketId":3}


query

gtwmobmaster@gmail.com
-> minä

{"ip":"192.168.43.249:8080","socketId":3}

3.4.1 Package dependencies

```
  "dependencies": {
    "buffer": "^6.0.3",
    "google-map-react": "^2.2.0",
    "react-bootstrap": "^2.4.0",
    "react-bootstrap-icons": "^1.8.4",
  }
```

Buffer
One piece of function when email is encoded base&4-format which is needed when sending email

google-map-react
for show the location with marker on the map

react-bootstrap & react-bootstrap-icons
for creating responsive an nice looking interface

3.4.2 Files

App.js
main application code

EmailForm.js
userinterface powered by bootstrap ui-components

SendMessage.js
workflow for sending message via gmail api using javascript libraries.

useImportScript.js
has dependency to SendMessage.js, it has custom hook for loading javascript libraries/methods.

MapView.js
for showing a map and a geolocation on the map with marker, has zoom and center properties.

devices.json
includes iot-devices gmail addresses, will be updated when the device is added or removed.

```
{
    "devices": [{
            "name": "Gtw1",
            "email": "gtw.mob1@gmail.com"
        },
        {
            "name": "Gtw2",
            "email": "gtw.mob2@gmail.com"
        },
        {
            "name": "Gtw3",
            "email": "gtw.mob3@gmail.com"
        }
    ]
}
```

3.5 Backend

Node/Express/websocket

Backend is needed for host client webpage (in production mode) and also websocket server. Every data is sended for backend and backend delivers the message/data to the target device/interface.

When websocket client is launched it takes contact to backend server the socket id is generated for the socket object and the sockets are saved for an array. Using id in communication the target socket will be found from an array by id.

3.5.1 Package dependencies

```
  "dependencies": {
    "cors": "^2.8.5",
    "express": "^4.18.1",
    "ip": "^1.1.8",
    "socket.io": "^4.5.1"
  }
```

Cors
enables the mechanism that allows restricted resources on a web page to be requested from another domain outside the domain from which the first resource was served

Express
web-server for productive mode, it has frontend as well.

Ip
for getting ip-address of backend server, client will ask that and send it to the mobile device.

socket.io
for the socket server

3.5.2 Files

Index.js
Includes all the working code

4 Instructions for using the demo

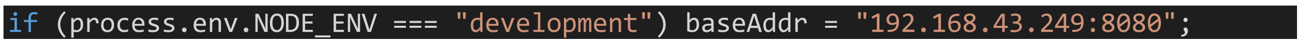
Every device has its own email address to read.
The mobile app is set to read messages from gtw.mob2@gmail.com, the another devices and email addresses exists but there is not mobile devices in function because multiple emulators at the same time running is not possible or I don’t know yet how to manage that.
For demo purpose there is couple of changing measure values

4.1 Limitations

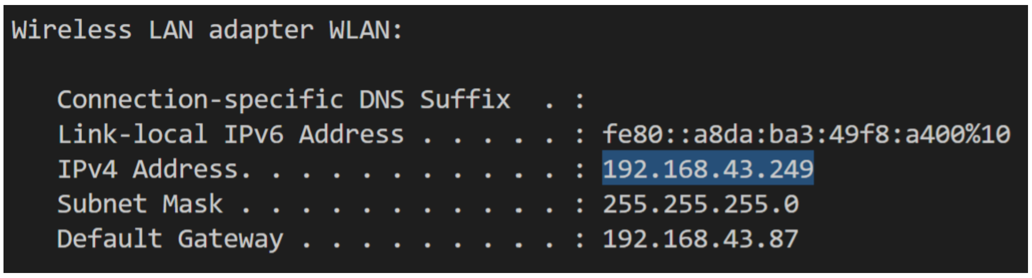
Sending email via google api has a delay for multiple email for the same email address. The delay is about 2 minutes

4.2 Developing mode in your developing environment

Change hardcoded ip for baseAddr varible in client app (App.js) to ip you have.



```
if (process.env.NODE_ENV === "development") baseAddr = "192.168.43.249:8080";
```



```
Wireless LAN adapter WLAN:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::a8da:ba3:49f8:a400%10
   IPv4 Address. . . . . . . . . . . : 192.168.43.249
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.43.87
```

Backend will be started with command **npm start**

Client will be started with command **npm start**

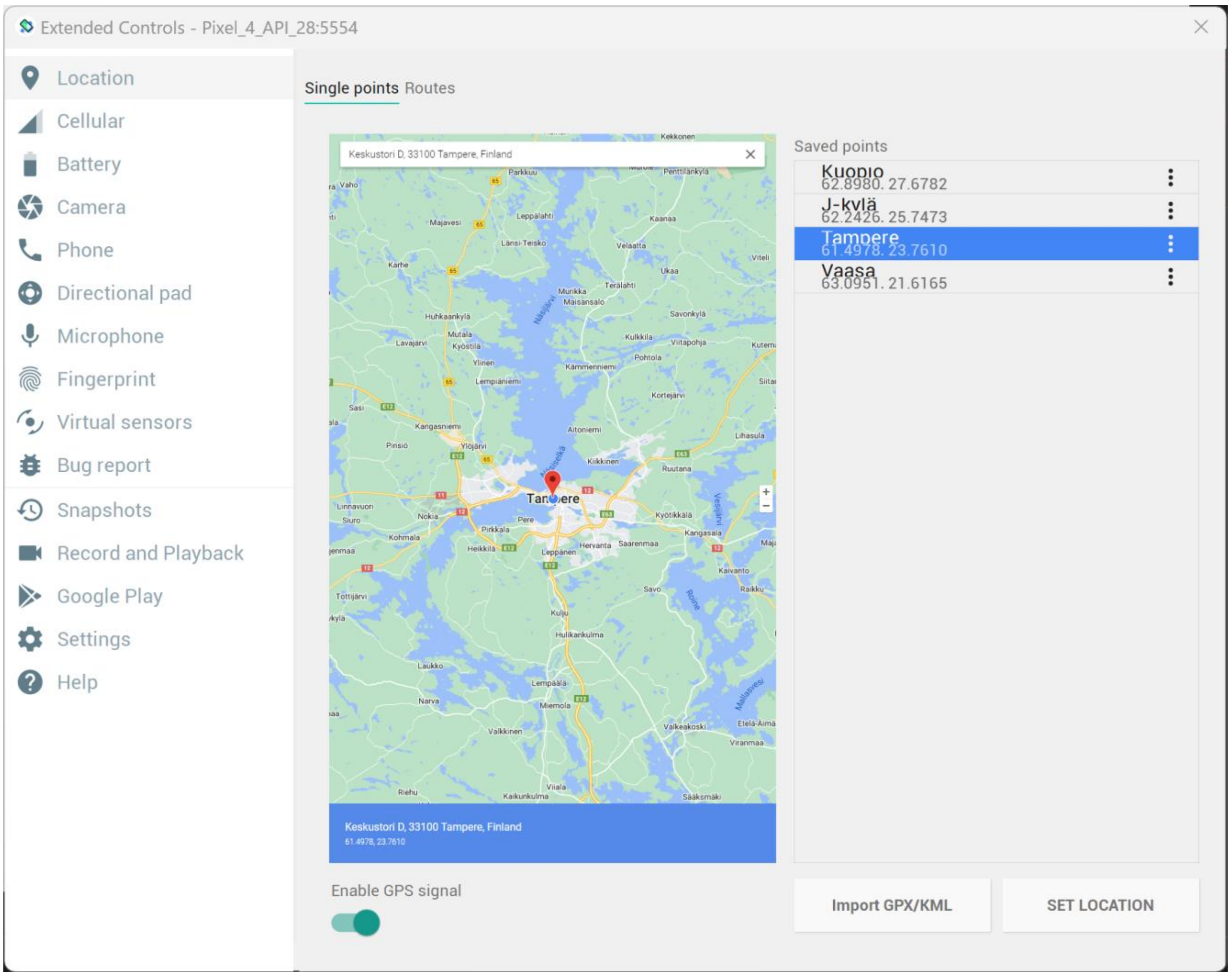
MobileApp will be started with command **npx react-native run-android**

In client app choose mobile device gtw2 from the dropdown menu, the only one mobile device (emulator) is in action and is set for reading the email address.

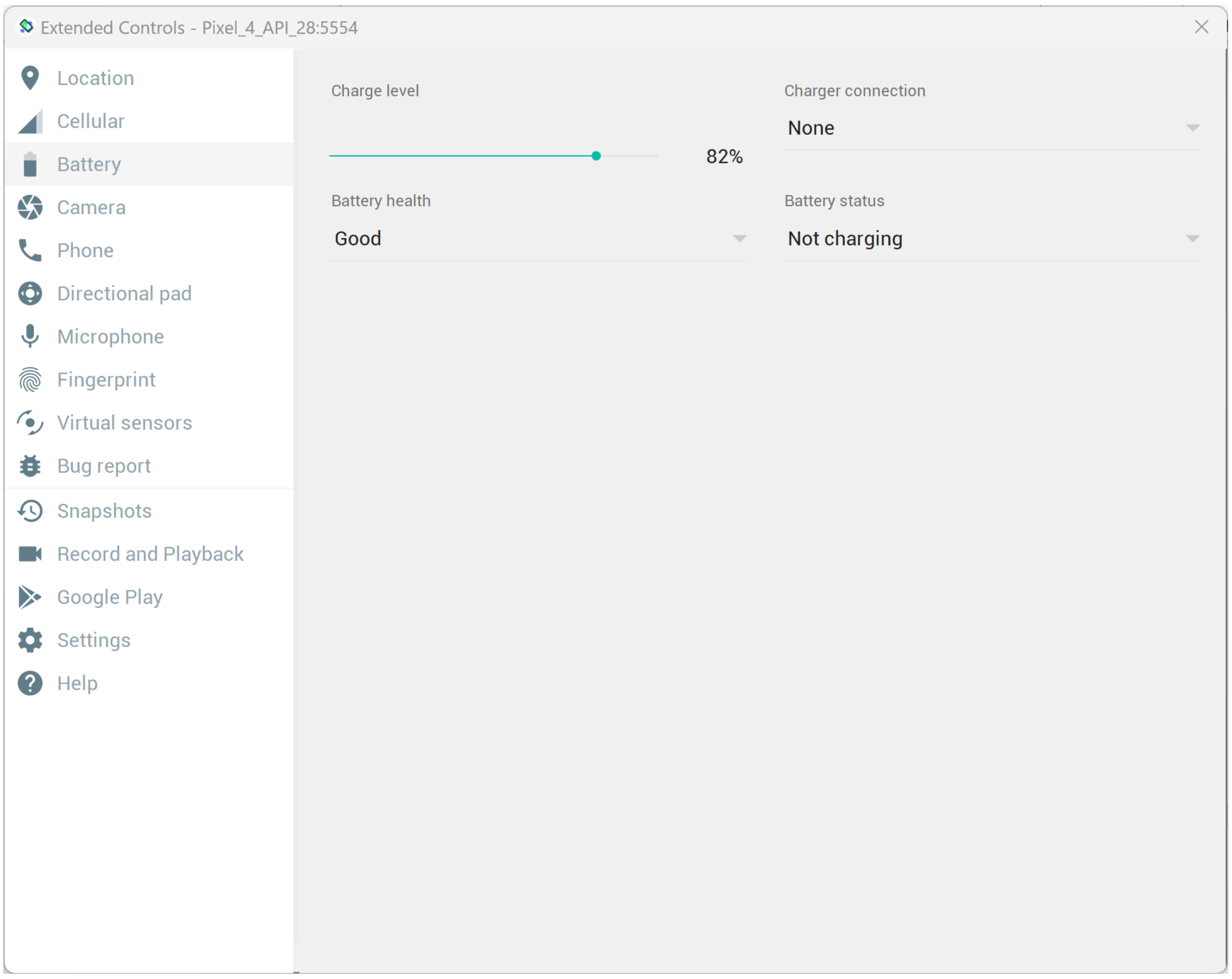
5 Emulator

Extended controls

5.1 Location



5.2 Battery level



Charge level: 82%
Charger connection: None
Battery health: Good
Battery status: Not charging

6 To be improved

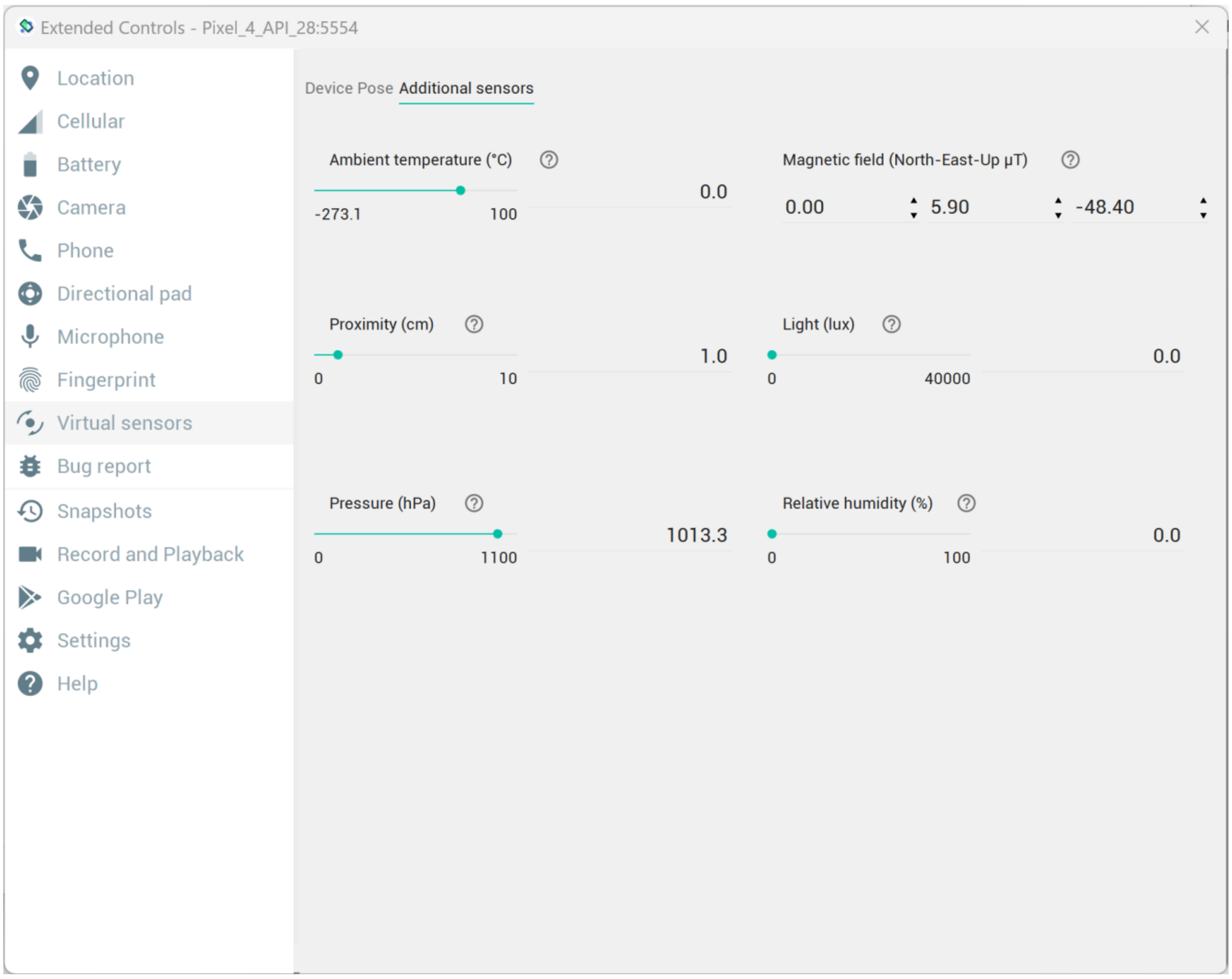
This is just a demo of technologies used and so it is having a lack of handling every possible type of error situation, some error situation may break the chain e.g. network is going down.

7 Further development/investigating

Continues data via socket.io is pretty easy to arrange, just hold the same message not mark it as read util to get permission. But the original idea was to serve several clients.

Want to try change client app (React based) to PWA, try ionic and webview for getting mobile experiment, but I think that afterall it would be most valuable do that with React Native (already tested some functions), but I am curious possible the rapid transform.

Simulate another measurements values (from virtual sensors) as you can seen in (at) the following picture.



Ambient temperature (°C) -273.1 100 0.0

Magnetic field (North-East-Up μT) 0.00 5.90 -48.40

Proximity (cm) 0 10 1.0

Light (lux) 0 40000 0.0

Pressure (hPa) 0 1100 1013.3

Relative humidity (%) 0 100 0.0

https://react-native-sensors.github.io/

8 Summary / reflection

After some troubles the demonstration works like a charm
There is not need use for native Android (android studio, java,kotlin) and ios (xcode, objective c, swift) developing platform/frameworks. Not was needed for any compromise done, everything in the plan was made successfully so React Native is my favourite cross-platform development technology.