Mobile Application development 2022

**Research assignment**

12.08.2022

TTOW0615

Timo Laakkonen

# 

Sisällysluettelo

[1 Routing 1](#_Toc65210399)

# Idea

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

In this exercise the working flow/system will be prentee as developing mode.

# Use in action/function long story short



Client send email to address, which mobile device reads continuesly.



The mobile device send data as an response for the client.

Tekniikka toiseen kappaleeseen viittaus ehkä tänne tai …

When

# Websocket

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

# For the technique

Action/launched will take place when client send email to address of the device. Device reads the oldest certais subject having email.

Mobile device is based on React Native platform framework **(tsekkaa ohjeista ja esimerkeistä…**

# Sequence

Mobile

Reads email whicj´h have subject ”” the content is as a JSON-format, the contnt will be parsed to

# Gmail

To use gmail api it is needed to be signedin and there must do some setting in

<https://console.cloud.google.com/apis/dashboard?project=gtwmob>

enable apis & services and create credientals API keys and OAuth 2.0 Client IDs

http web is uses for reading messages and token is needed for header every sigle requests and (reading and sendin posts)

This is another story the focus is in Mobile Applivation Development.

Because of the main topic is mobile this is not going forward more…

# Mobile

NO NO NO sillä yhteys muodostetaan serveriin jonka Ip saatiin in email message. 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 communicating via websocket to another device/client.

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

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

Next device will discover it’s location and memory values as well battery level.

Device will create websocket connection to ip-address given in email message.

Before sending data via websocet data object is converted as json-string.

Every iot-device has it’s own email address, which inbox is readed continuesly

Packages dependensies

"@react-native-community/geolocation": "^2.1.0",

    "@react-native-google-signin/google-signin": "^7.2.2",

    "axios": "^0.27.2",

    "react": "17.0.2",

    "react-native": "0.68.2",

    "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 convert email to encode/decode 64-bit xxxx

**react-native-device-info**

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

**react-native-elements**

Some ui issues/elements

**react-native-svg & react-native-svg-transformer**

To show svg-type of images.

# Backend

Webserver

Express is for productive mode, it has frontend as well.

Backend is needed for host client webpage 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.

"cors": "^2.8.5",

    "express": "^4.18.1",

    "ip": "^1.1.8",

    "websocket": "^1.0.34"

**cors**

Enable the mechanism that allows restricted [resources](https://en.wikipedia.org/wiki/Web_resource) on a [web page](https://en.wikipedia.org/wiki/Web_page) to be requested from another [domain](https://en.wikipedia.org/wiki/Domain_name) outside the domain from which the first resource was served

**express**

web-server

**ip**

for getting ip-addres of host backend server

**websocket**

for websocket server the host of…

Tässä kerrotaan kun server havaitsee että uusi client on ottanuty yhteyden niin genroi sen

socket-objektiin id:een, jonka perusteella socket tunnistetaan jatkossa, socket tallennetaan arrayhin, jossa on kaikki voimassa olevat socketit.

sender: 'mob'

# Client

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 rto another device/client. The email sended has subject “query” content is stringyfied as JSON format so its easy to parse to object. The content of the message is in json.format having values for socket id and server ip.

{"ip":"[192.168.43.249:8080](http://192.168.43.249:8080/)","socketId":3}

Kuva, joka sisältää kohteen teksti

Kuvaus luotu automaattisesti

"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 location with marker on the map

**react-bootstrap & react-bootstrap-icons**

for creating responsive an nice looking interface

# Code level

## Mobile

React Native and some packets

### Files

**App.js**

main program

Functions:

getScope

getMessages

markAsRead

readMessage

getConfig

**Geolocation.js**

getting location of the device

**Deviceinfo.js**

Getting information of the device, in this app ipAddress, batteryLEvel, usedMemory, totalMemory.

const message = {

      sender: 'mob',

      location,

      id: socketId,

      ipAddress: ipAddress,

      batteryLevel: batteryLevel,

      usedMemory: usedMemory,

      totalMemory: totalMemory,

      time: Date.now(),

    };

## Backend

Node/Express/websocket

### Files

Index.js

## Client

In this demostration the client is wep app, it could be mobile app as well

React/Bootstrap

### Files

**useImportScript.js**

**EmailForm.js**

**SendMessage.js**

**MapView.js**

**App.js**

handleSendMessage

getDateTime

# Some words for the code made

Base64 on email

# Mobile device

The first contact to the mobile device is via email. Logging in to google api is necessary.  
The token is needed for header part of every query to every function of google api. For getting token some setting is needed is described in paragraph 5.

Communicating with gmail api is web/http using axios.

## Main loop

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

Flowshart workflow



At the step 1 in main loop code 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/handling.

if there is none, just looping and waiting…  
When a 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.

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.  
Another parameter of the message is socketId of the client which server had given.  
To take contact to client It’s socketId in the server is needed.

Next it will be checked that webSocket connection has been created and settled down.

Next the device will discover it’s location, ip-address, memory values and battery level.

The message to send via webSockert server to Clien is 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’s know how seent message, when server creates socket the message is sent to client the parameter is “server” so client knows how to handle messages.

It can seen as strange because the loopin inside useEffect hook will work when the value of dependency state variable will change it need to be changed every round.

Async await technique is used for every step on the sequence, reading requests from email, collecting the data, sending response

## Steps workflow

# Limitations

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

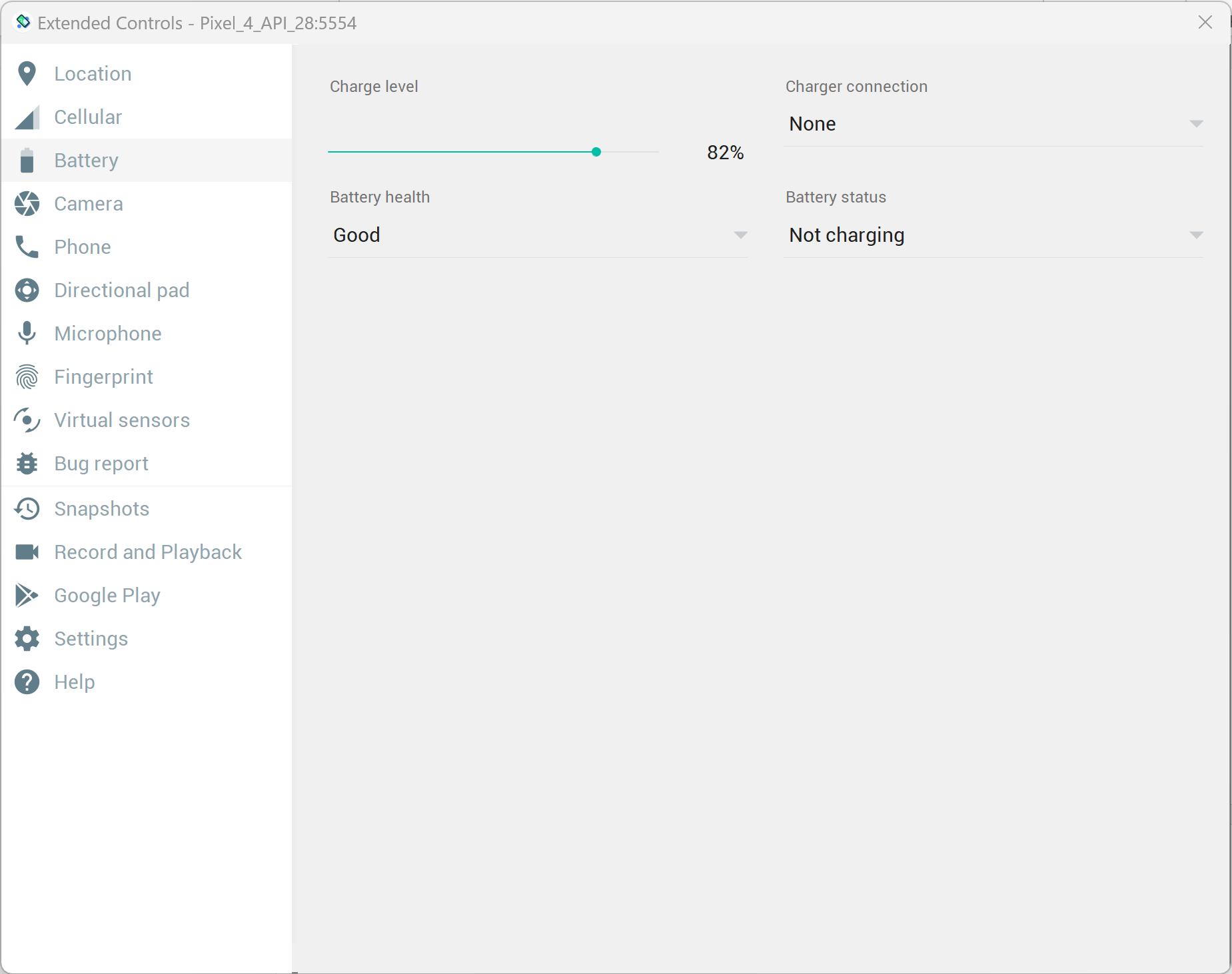
# Demostration/Simulation

Location

Kuva, joka sisältää kohteen kartta

Kuvaus luotu automaattisesti

Battery level

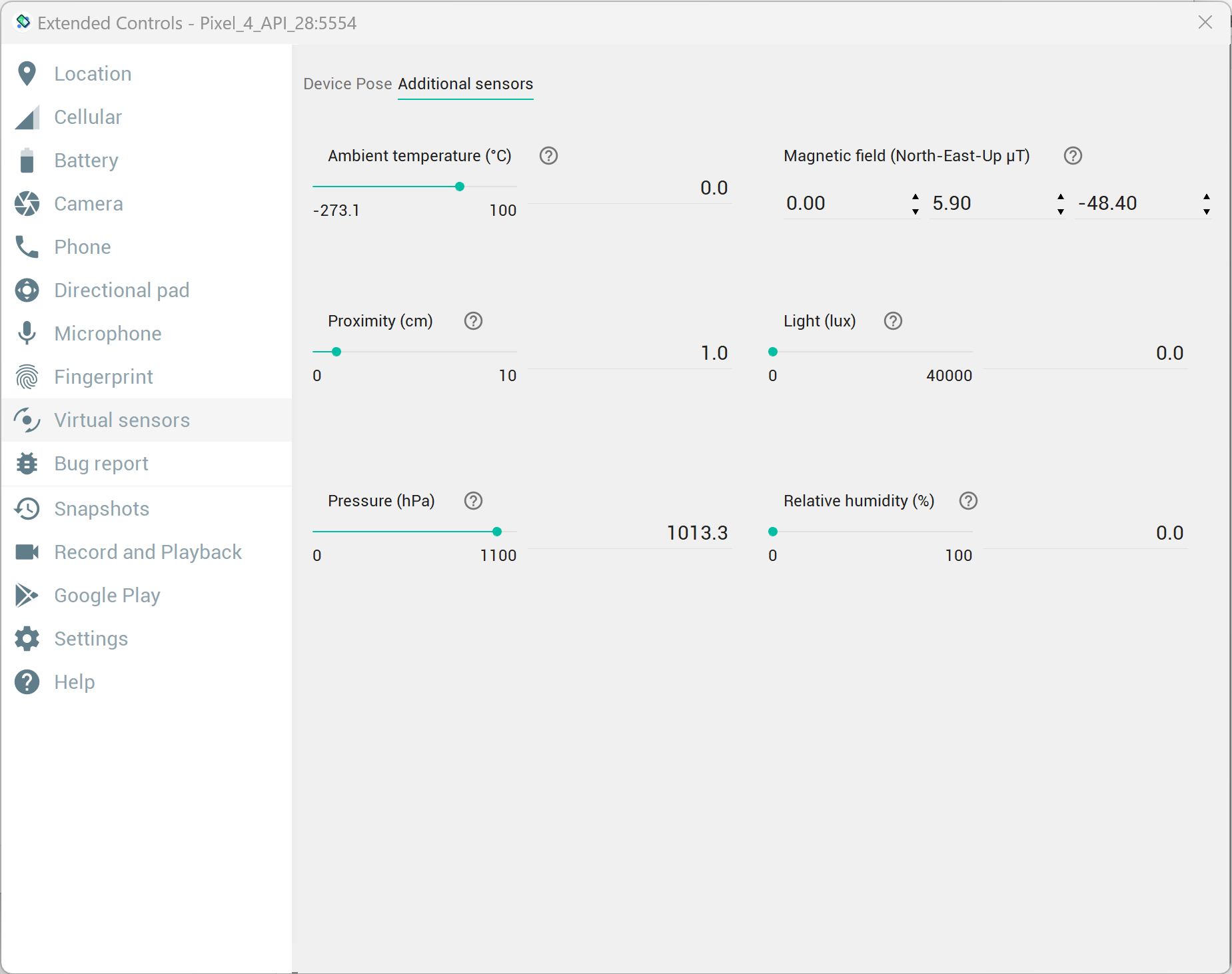


# To be noticed to pay attention ??? Huomioitavaa

Pitäis huomioids jos jokin chain ei toimi kuinka sovellus käytäytt tai toipuu kun jos esim serveri on alhaalla ja nousee ylös.

# For developing/future

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



I believe that React client app would be quite easily converted to Mob React Native, acrtually I have tested some function PWA or native or ionic.

Webview could also be worth of testing.

# Last/Final words

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/frsameworks. Not need for any compromise, everything in the plan was made successfully so React Native rules as cross-platform development.