Wearable Sensor App Manual

Bourdage, R., van Dijk, M. & Gawhns, D.

Leiden Institute of Advance Computer Science

June 2021

Contents

1 Introduction

The "liacs.sensorapplication" software was created for the Samsung Gear Fit Pro 2, which is a digital watch issued in 2016 containing an accelerometer, gyroscope, barometer and GPS sensors.This Samsung wearable runs on Tizen OS version 2.3.1:13 and is based on Linux OS.The wearable sensor was selected for a research project as it's operating system allowed for the extraction of raw data files. In addition, the wearable had to be customizable to incorporate the following additional research project requirements: patient id, the recording of sensor files, saving of the measurement settings and privacy settings. Samsung offered Tizen Studio software environment free of charge to design applications and services for the watch. Many programming examples and API documentation were also available online.(see: https://developer.samsung.com/tizen/Galaxy-Watch /certificate/creating-certificate.html)

2 The Software Package

The software package contains a Tizen OS widget application named "liacs.sensorapplication" visible in the list of applications with title "Sensors" and a digital brain as icon. The application carries a snipper entry widget control to enter the person id, a number between 000 and 999, and two buttons; the RESTART button to start and restart measurements and the CLEAN button to be pressed 3x to remove all measurements from the watch.The activity data is stored in sensor files and have a file name which contains the person id, watch id, date, time and sensor type. The package is configured by a configuration file which is pushed on the watch in an application independent accessible file system folder.

The configuration is stored for every measurement. The configuration file consists of: a) the recording in terms of frequency; b) recorded sensors; and c) the centre and size of the privacy circle. For reproducibility and data management, a naming convention was developed for all files, including the con files, listed in the following order: a) Patient ID; b) Watch ID; c) Date (YYYY, MM , DD); d) Time (HR, Minute, Seconds). For an example:

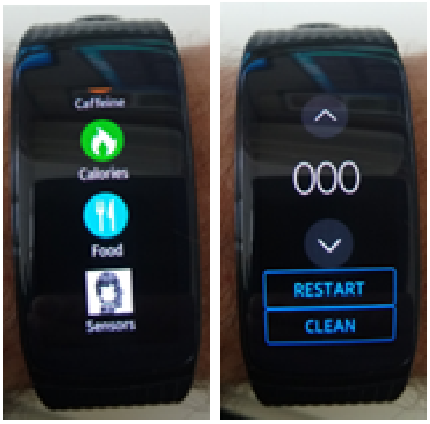
001 2021 05 13 10 44 00 d9a8 aag.dat

002 2021 05 12 11 09 09 d9d2 bar.dat

003 2021 05 13 10 43 01 d9d7 con.dat

004 2021 05 12 11 06 49 d9d6 gps.dat

For power consumption and memory size, the gravity accelerometer and gyroscope sensors with high frequency readouts are separately stored from the low frequency readouts of the barometer, battery charge percentage and GPS values. The sensor values are stored in a commonly used file format, the comma separated value file format.



The software package also contains a Tizen OS widget service named "liacs.sensorservice" which runs in the background, not visible in the list of applications and cannot be deleted by the user of the watch. The sensor service processes the messages RESTART and CLEAN send by the sensor application. After the start of the watch, the sensor service is activated and waiting for these messages. After one valid message received, the service starts the recording of the sensor values and stores it into comma separated values files. The measurement is based on the contents of the uploaded configuration file. Wifi on the watch needs to be on when uploading the con file and to download the collected data off of the watch. Otherwise, wifi can be off during data collection. In addition, the watch's GPS needs to be turned on to collect GPS data.

A figure in Appendix A shows a state diagram with transitions and format "event - actions", as a formal description of the states and relevant state events between the watch application manager, the sensor application and service.

The wearable was manually tested. The accelerometer and the gyroscope were validated by perpetually rotating the watch at different speeds. The barometer was validated by comparing height differences in a GPS track with the air pressure measured. Zero measurements were obtained by calculating accelerometer means and standard deviations and comparing the results between watches. The variance between watches was of 0.028 on average. In order to account for this variance, a threshold was created by using the following formula: threshold = mean(G) + 4 * std(G), where G = the gravity acceleration vector norm [sqrt(ax2 + ay2 + az**2)]. For reproducibility, it is advised to calibrate the sensor values and convert them to SI basic units before data processing.

For a Python notebook, see Appendix B for calculating means and standard deviations and Appendix C for plots. Below, find sample of calculation outcomes.

		file	watch id	G mean	G std	G threshold
1	0	SensorFiles\000 8468 2021 05 14 09 21 42 aag.dat	8468	9.86597550164938	0.02727677575889011	9.97508260468494
2	1	SensorFiles\000 9669 2021 05 13 10 41 58 aag.dat	9669	9.86377448290946	0.030101077533829172	9.984178793044777
3	2	SensorFiles\000 aa1d 2021 05 14 09 22 36 aag.dat	aa1d	9.835042203588255	0.02841113151599695	9.948686729652243
4	3	SensorFiles\000 aa85 2021 05 12 11 08 02 aag.dat	aa85	9.689395988420705	0.02601773725365709	9.793466937435333
5	4	SensorFiles\000 ab75 2021 05 14 09 21 11 aag.dat	ab75	9.804296024317141	0.02537352530564862	9.905790125539735
6	5	SensorFiles\000 ab79 2021 05 12 11 07 35 aag.dat	ab79	9.940976285533468	0.026582402128834517	10.047305894048806
7	6	SensorFiles\000 d835 2021 05 14 09 23 11 aag.dat	d835	9.713414578861642	0.02445733455845776	9.811243917095473
8	7	SensorFiles\000 d8eb 2021 05 12 11 08 58 aag.dat	d8eb	9.830541106707518	0.026136408316424557	9.935086739973215
9	8	SensorFiles\000 d8eb 2021 05 13 10 42 08 aag.dat	d8eb	9.984861696657145	0.032183044748011785	10.113593875649192
10	9	SensorFiles\000 d8eb 2021 05 14 09 21 42 aag.dat	d8eb	9.850056899730458	0.025893550034521855	9.953631099868545
11	10	SensorFiles\000 d96d 2021 05 14 09 22 21 aag.dat	d96d	9.691030880764878	0.02618579691145297	9.79577406841069
12	11	SensorFiles\000 d9a8 2021 05 12 11 09 02 aag.dat	d9a8	9.788083917614303	0.05738249377855914	10.017613892728539
13	12	SensorFiles\000 d9a8 2021 05 13 10 44 00 aag.dat	d9a8	9.822500498419007	0.026799568478801276	9.929698772334213
14	13	SensorFiles\000 d9d2 2021 05 12 11 09 09 aag.dat	d9d2	9.858874368926472	0.02788675194953208	9.9704213767246
15	14	SensorFiles\000 d9d7 2021 05 12 11 06 49 aag.dat	d9d7	9.759701779355858	0.028337831406508886	9.873053104981894
16	15	SensorFiles\000 d9d7 2021 05 13 10 43 01 aag.dat	d9d7	9.861420691529656	0.03063625876171213	9.983965726576503
17	16	SensorFiles\000 da42 2021 05 13 10 42 22 aag.dat	da42	9.72526086504724	0.026432347643453074	9.830990255621051
18	17	SensorFiles\000 daa0 2021 05 12 11 07 18 aag.dat	daa0	9.703385223500074	0.024832253617908893	9.80271423797171

For every sensor readout frequency, a separate sensor file is created. This reduces power consumption and memory usage. The application contains a clock which is used for all recordings so the data can be aligned with time. The sensor values are stored in a commonly used file format, the comma separated value file format.

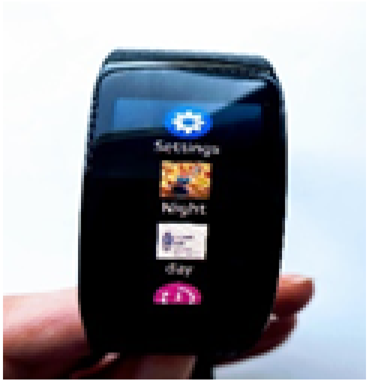
The wearable chosen meets the requirements set out in 2018.

3 How to use the Watch

1. Charge watch on charging dock

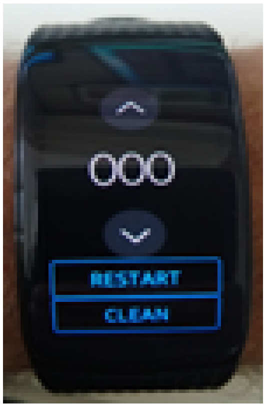
2. Switch on the watch after charging by pressing and holding small side button

3. Access the Settings by pressing small side button once



4. Also found on the main screen is the Sensors app

5. Press on the Sensor app, input an identifier and press restart to begin measurement. To delete any data collected, press Clean button three times



6. To turn off watch, press and hold small side button

4 Installation of Tizen Studio Environment

1. Install Tizen Studio 4.1 with IDE installer. Once instillation is completed, the package manager will be prompted
2. Install the Tizen extensions for native development for version 2.3.1 (tab: Main SDK). The watch has version 2.3.1:13 of the Tizen OS, therefore choose to install the 2.3.1 SDK together with the certificate SDK.
3. Install the Tizen extension for Certificates (tab: Extension SDK)*
4. Make a Samsung developer account, this will be required to create the necessary certificates: https://developer.samsung.com/sa/signIn.do
5. Run package manager (Tizen Studio - tools), go through the wizard steps.

Note: Installation guide of the Tizen Studio development environment can be found in https://developer.samsung.com/tizen/Galaxy-Watch/certificate/creating-certificate.html

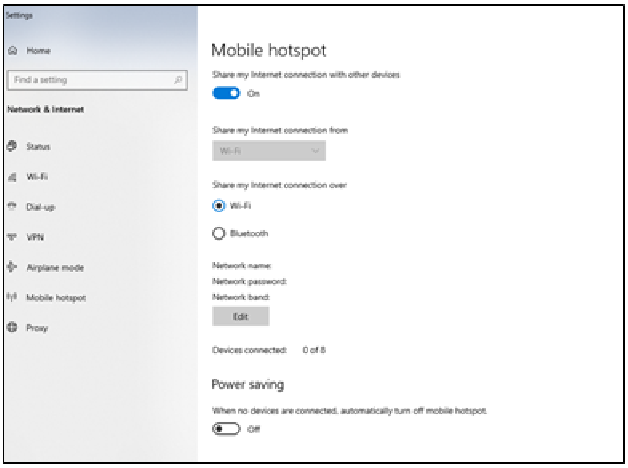
4.1 *More on Certificates

1. Connect watch via device manager (see How to Connect Watch to Tizen)
2. Open Certificate Manager
3. Hover over the little plus to add a new certificate
4. Click Samsung Certificate
5. You will be asked to make an author certificate and a distributor certificate
6. If you want to add a watch: click on the plus sign as if adding a new certificate, click samsung certificate, select use existing certificate profile, click no when asked if you want to create a new author certificate, create new distributor certificate, you will be asked to chose a password and either add a DUID manually or connect to the new device. Alternatively, you can use a .txt file with a new DUID per line to create certificates.

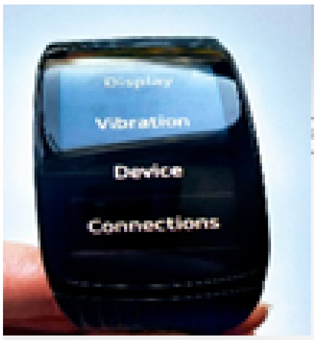
Tip: consider setting the date to one day in the future while setting up the watch to avoid certification date issues.

5 How to Connect Watch to Tizen

1. To connect the watch to Tizen Studio, go to the computer's "Settings" window, navigate to the "Mobile Hotspot" and turn it on. It will then display a Network Name and Network Password



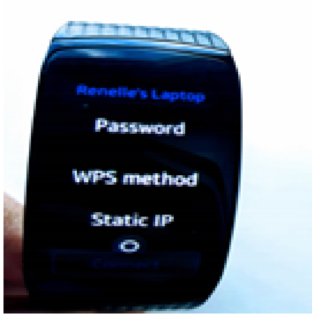
2. After the hotspot is turned on, go into the watches' settings, and select the "Connections"



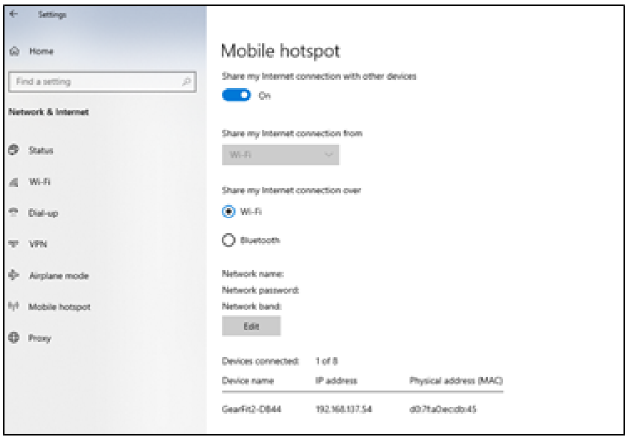
3. Go to “Wi-Fi” and turn it on

4. Press on “Wi-Fi Networks” and select the laptop’s mobile hotspot

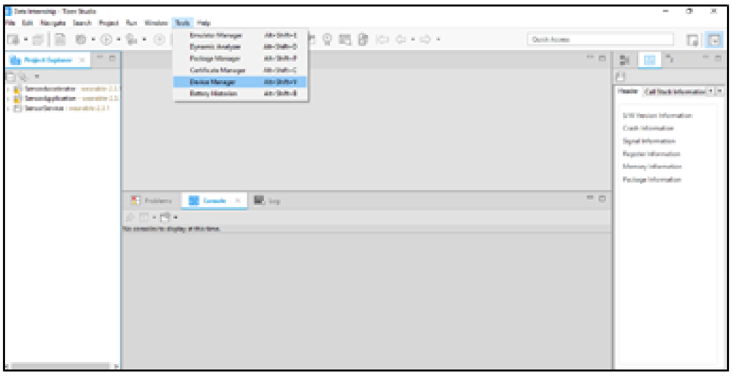
5. Press on “Password” to enter the laptop’s password



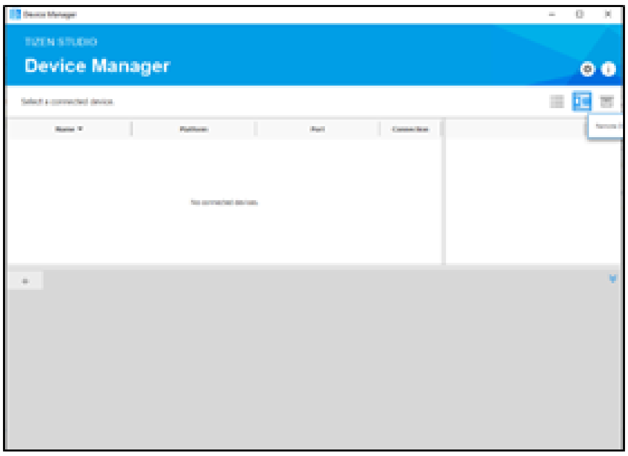
6. Make sure that the watch has connected to the laptop



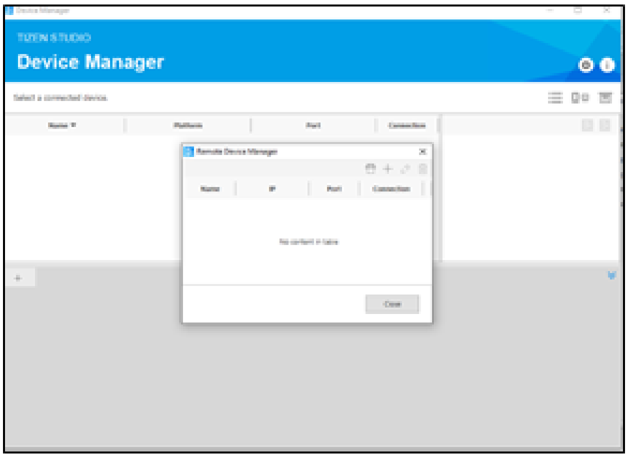
7. Open Tizen, and select "Device Manager" from the Tools tab



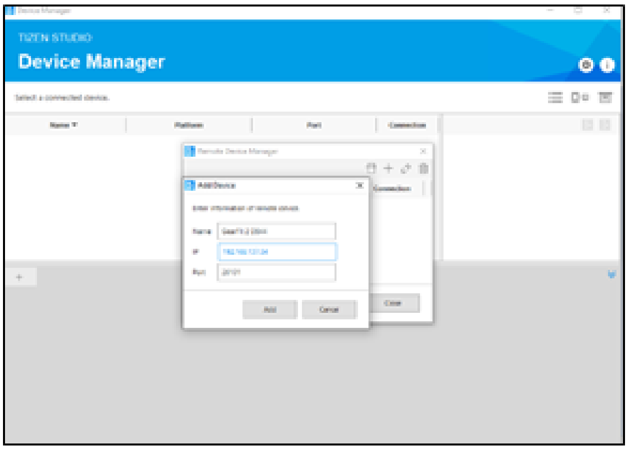
8. When the Device Manager opens, select the "Remote Device Manager"



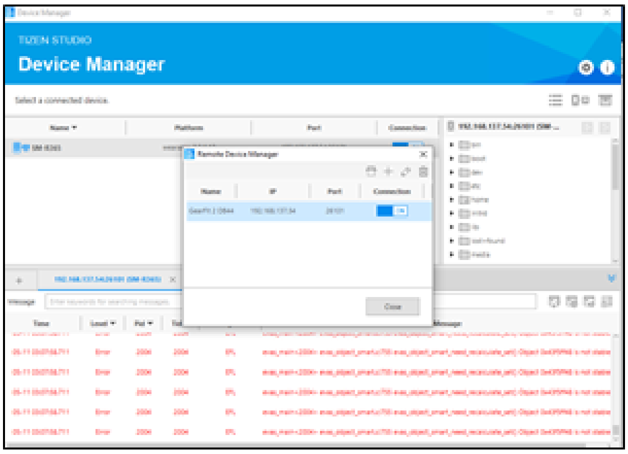
9. In the Remote Device Manager, select the plus sign to connect the watch



10. Put in Name, and IP address listed in the laptop's mobile hotspot window



11. Press “Add”, and then turn on the connection, the device information should then appear in the Device Manager



6 Installation of LIACS Sensor App and Sensor Service

1. Make a GitHub account
2. Ask for an invite to this repository. The current owner is the LiacsProjects organisation run by MarinusVanDijk (m.k.van.dijk@liacs.leidenuniv.nl)
3. Download from the main branch the software project sensor application and sensor service. Click on "code" and download zip file
4. Open the project files one by one with the Tizen Studio 4.1
5. Open the device manager and make connection to the watch
6. Select the sensor application project, right-click on mouse, choose run-as, choose native device. The software is now downloaded to the watch
7. Select the sensor service project, right-click on mouse, choose run-as, choose native device
8. The sensor application is found at the bottom of all applications
9. Put watch in debug mode, switch all app off with settings only switch on wifi
10. Push - with the device manager - a configuration file with name "configuration.dat" on folder "/opt/var/tmp/". Notes about the configuration file: Write timer can be set to a higher frequency than the data is collected to miss fewer signals The privacy circle has a max range of 5000 mt, anything higher sets the privacy circle to 100 mt
11. Do a zero measurement (for calibration offline) for 15 minutes, upload the sensor + con files

NOTE: You can also use the sdb (Smart Development Bridge) tool which come with Tizen Studio instead of the Device Manager. (See How to use SDB).

7 How to Use Smart Development Bridge (SDB)

The Tizen Studio contains a command prompt tool called “sdb.exe”, the Smart Development Bridge tool. This tool can install packages on the target, as well as pull and push files from and to the target. https://docs.tizen.org /application/tizen-studio/common-tools/smart-development-bridge/

1. First, after installation of the Tizen Studio environment, copy the sdb.exe from c:/TizenStudio/tools/sdb.exe to c:/Windows/System32 folder
2. Open a git bash command terminal or windows command prompt

Try the following:

1. Get the version number of the smart development bridge

```
$ sdb version
```

2. Get the list of all devices connected

```
$ sdb devices
```

3. Connect to a device

```
$ sdb connect 192.168.137.95:26101
```

4. Get the ip address

```
$ sdb get-serialno
```

```
Richard@HPWhiteFullFunction MINGW64 /c/LiacsProjects/FilesUpload
$ sdb get-serialno
192.168.137.95:26101
```

5. Install a target package on a device

```
$ sdb install "package name.tpk"
```

```
Richard@HPWhiteFullFunction MINGW64 /c/LiacsProjects/FilesUpload
$ sdb install liacs.sensorapplication-1.0.0-arm.tpk
WARNING: Your data are to be sent over an unencrypted connection and could be read by others.
pushed liacs.sensorapplication-1.0.0-arm.tpk    100%        48KB          0KB/s
1 file(s) pushed. 0 file(s) skipped.
liacs.sensorapplication-1.0.0-arm.tpk   159KB/s (49453 bytes in 0.302s)
path is /opt/usr/apps/tmp/liacs.sensorapplication-1.0.0-arm.tpk
__return_cb req_id[281240002] pkg_type[tpk] pkgid[liacs.sensorapplication] key[start] val[update]
__return_cb req_id[281240002] pkg_type[tpk] pkgid[liacs.sensorapplication] key[install_percent] val[30]
__return_cb req_id[281240002] pkg_type[tpk] pkgid[liacs.sensorapplication] key[install_percent] val[60]
__return_cb req_id[281240002] pkg_type[tpk] pkgid[liacs.sensorapplication] key[install_percent] val[100]
__return_cb req_id[281240002] pkg_type[tpk] pkgid[liacs.sensorapplication] key[end] val[ok]
spend time for pkgcmd is [5543]ms
```

6. Execute a Linux command on the target

```
$ sdb shell ls -l opt/var/tmp
```

```
Richard@HPWhiteFullFunction MINGW64 /c/LiacsProjects/FilesUpload
$ sdb shell ls -l opt/var/tmp
total 16
-rwxr-xrwx 1 developer developer  733 May  5 23:14 configuration.dat
-rwxr-xrwx 1 developer developer 1598 Apr 10 05:37 testupload.txt
```

7. Pull the configuration.dat file from the target

```
$ sdb pull opt/var/tmp/configuration.dat
```

```
Richard@HPWhiteFullFunction MINGW64 /c/LiacsProjects/FilesUpload
$ sdb pull opt/var/tmp
pulled              configuration.dat   100%        733 B          0KB/s
pulled                 testupload.txt   100%       1598 B          0KB/s
2 file(s) pulled. 0 file(s) skipped.
opt/var/tmp                         13KB/s (2331 bytes in 0.162s)
```

8. Push the configuration.dat file from the target

```
$ sdb push opt/var/tmp/configuration.dat
```

```
Richard@HPWhiteFullFunction MINGW64 /c/LiacsProjects/FilesUpload
$ sdb push c:/LiacsProjects/FilesUpload/configuration.dat opt/var/tmp
WARNING: Your data are to be sent over an unencrypted connection and could be read by others.
pushed              configuration.dat   100%        733 B          0KB/s
1 file(s) pushed. 0 file(s) skipped.
c:/LiacsProjects/FilesUpload/configuration.dat   2KB/s (733 bytes in 0.334s)
```

9. Pull all sensor files from the target

```
$ sdb pull opt/usr/apps/liacs.sensorservice/data
```

```
Richard@HPWhiteFullFunction MINGW64 /c/LiacsProjects/FilesUpload
$ sdb pull opt/usr/apps/liacs.sensorservice/data
pulled 000 D8F8 2021 05 06 09 11 40 aag.dat     100%       10MB        159KB/s
pulled 000 D8F8 2021 05 06 09 11 40 gps.dat     100%        0 B          0KB/s
pulled 100 D8F8 2021 05 05 20 25 05 gps.dat     100%       74 B          0KB/s
pulled 000 D8F8 2021 05 06 09 11 40 bar.dat     100%      648KB        712KB/s
pulled 100 D8F8 2021 05 05 20 25 05 bar.dat     100%     1955KB        351KB/s
pulled 100 D8F8 2021 05 05 20 25 05 con.dat     100%      673 B          0KB/s
pulled 100 D8F8 2021 05 05 20 25 05 aag.dat     100%       34MB       1640KB/s
pulled 000 D8F8 2021 05 06 09 11 40 con.dat     100%      673 B          0KB/s
8 file(s) pulled. 0 file(s) skipped.
opt/usr/apps/liacs.sensorservice/data   733KB/s (49818262 bytes in 66.294s)
```

10. Remove all sensor files from the target

```
$ sdb shell rm opt/usr/apps/liacs.sensorservice/data/*.dat
```

11. Disconnect the device

```
$ sdb disconnect
```

8 Possible Errors

1. Tip Windows Installations: In case you cannot run the Device Manager and get a message that msvcr120dll is missing, you will have to load an extension for your OS:

   `https://answers.microsoft.com/en-us/windows/forum/windows_10-performance`
   `/msvcr120dll-is-missing-or-error/aafe820f-4dbb-4043-aba2-e4ac2dcf69c1`

2. In case you cannot run the Device Manager and get a message that msvcp120dll is missing, you will have to load an extension for your OS:

   `https://answers.microsoft.com/en-us/windows/forum/windows_7-windows_`
   `install/missing-msvcp120dll-file/f0a14d55-73f0-4a21-879e-1cbacf05e906`

3. While uploading the package of the sensor service on another watch, the launch process ends up in an error 75

   `"signature invalid device unique id"`

Additionally, there are forum questions asked and answered on:

https://wiki.tizen.org/SDK

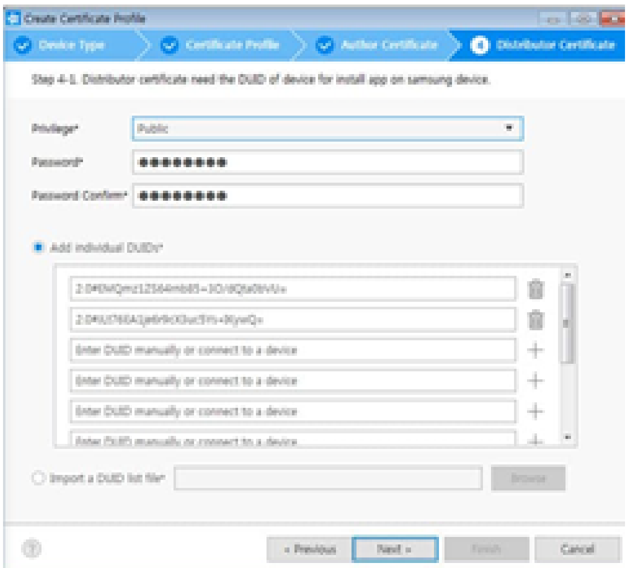
or more information can be found at:

https://docs.tizen.org/application/tizen-studio/common-tools/certificate-registration/

and at:

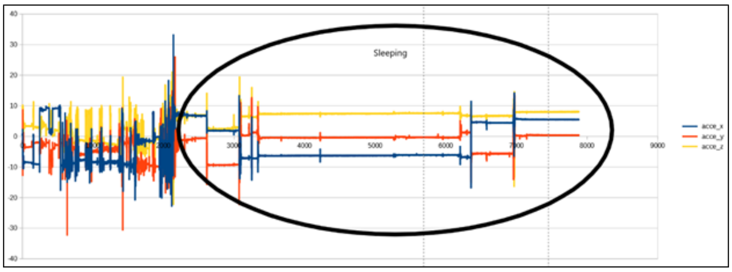
https://developer.samsung.com/galaxy-watch-develop /getting-certificates/create.html

Another hint: In one of the steps in the Certificate Manager you need to supply is a list of DUID.

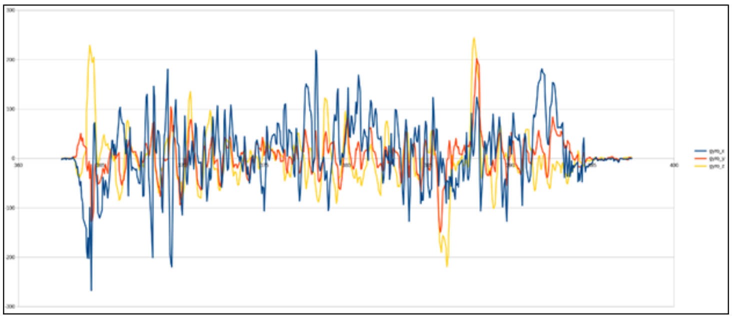


9 Example Data

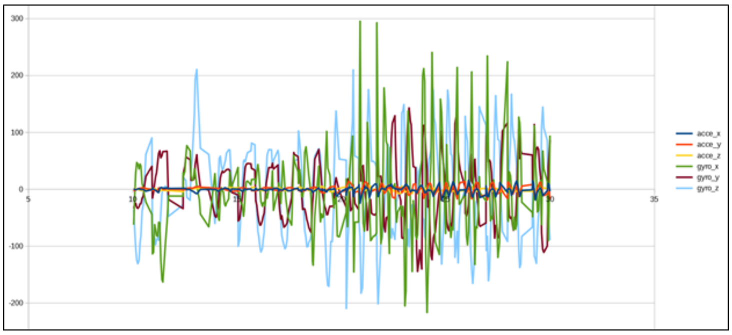
1. Activity timeline of gravity accelerometer values with sample frequency of 10 Hz. The first part shows sitting then standing, activity indoors, then sleeping; the watch is on the non-dominant wrist



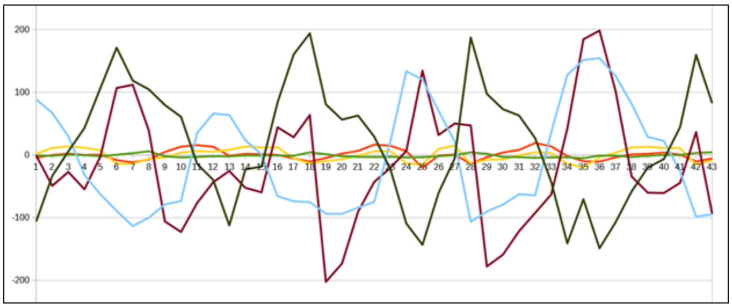
Gyroscope, 10 Hz:



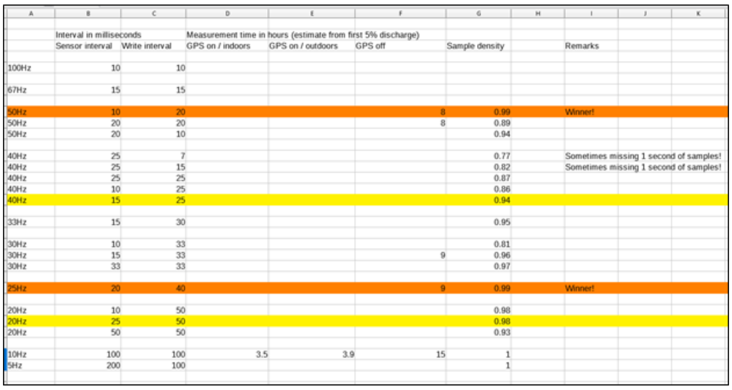
2. A close-up moment from walking to running:



Close-up on sample scale:



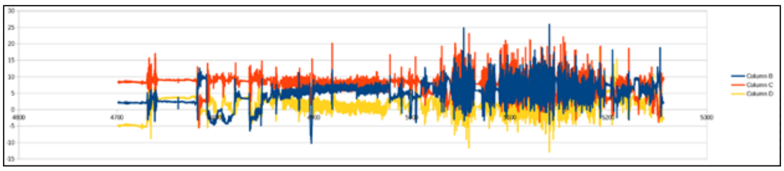
3. Interval Combination Experiment:

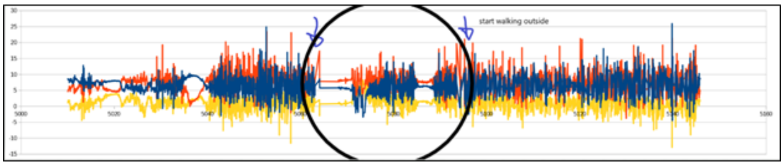


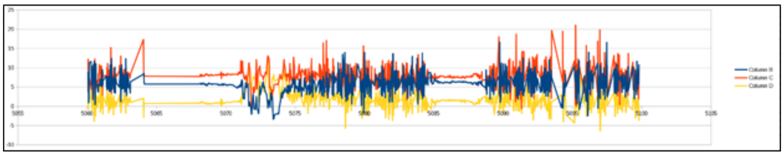
A	B	C	D	E	F	G	H	I	J	K
	Interval in milliseconds		Measurement time in hours (estimate from first 5% discharge)							
	Sensor interval	Write interval	GPS on / indoors	GPS on / outdoors	GPS off	Sample density		Remarks		
100Hz	10	10								
67Hz	15	15								
50Hz	10	20			8	0.99		Winner!		
50Hz	20	20			8	0.89				
50Hz	20	10				0.94				
40Hz	25	7				0.77		Sometimes missing 1 second of samples!		
40Hz	25	15				0.82		Sometimes missing 1 second of samples!		
40Hz	25	25				0.87				
40Hz	10	25				0.86				
40Hz	15	25				0.94				
33Hz	15	30				0.95				
30Hz	10	33				0.81				
30Hz	15	33			9	0.96				
30Hz	33	33				0.97				
25Hz	20	40			9	0.99		Winner!		
20Hz	10	50				0.98				
20Hz	25	50				0.98				
20Hz	50	50				0.93				
10Hz	100	100	3.5	3.9	15	1				
5Hz	200	100				1				

The best combination is 50Hz and 25Hz and lower. For 50Hz take a 10 ms for the sensor intervals and 20 ms for the writer timer interval. The sample density is a measure calculated by the number of records in the sensor files divided by the time in seconds. This result is divided by the sample frequency expected by the interval settings. Sample density = number of records / seconds / expected frequency For example: For 50Hz the samples expected is 50 per second. If the number of records in the sensor file is 4800, the sample density is $4800/5000 = 0.96$. For the best result the sample density must be 1.0

4. Accelerometer 4700 - 5300 seconds:



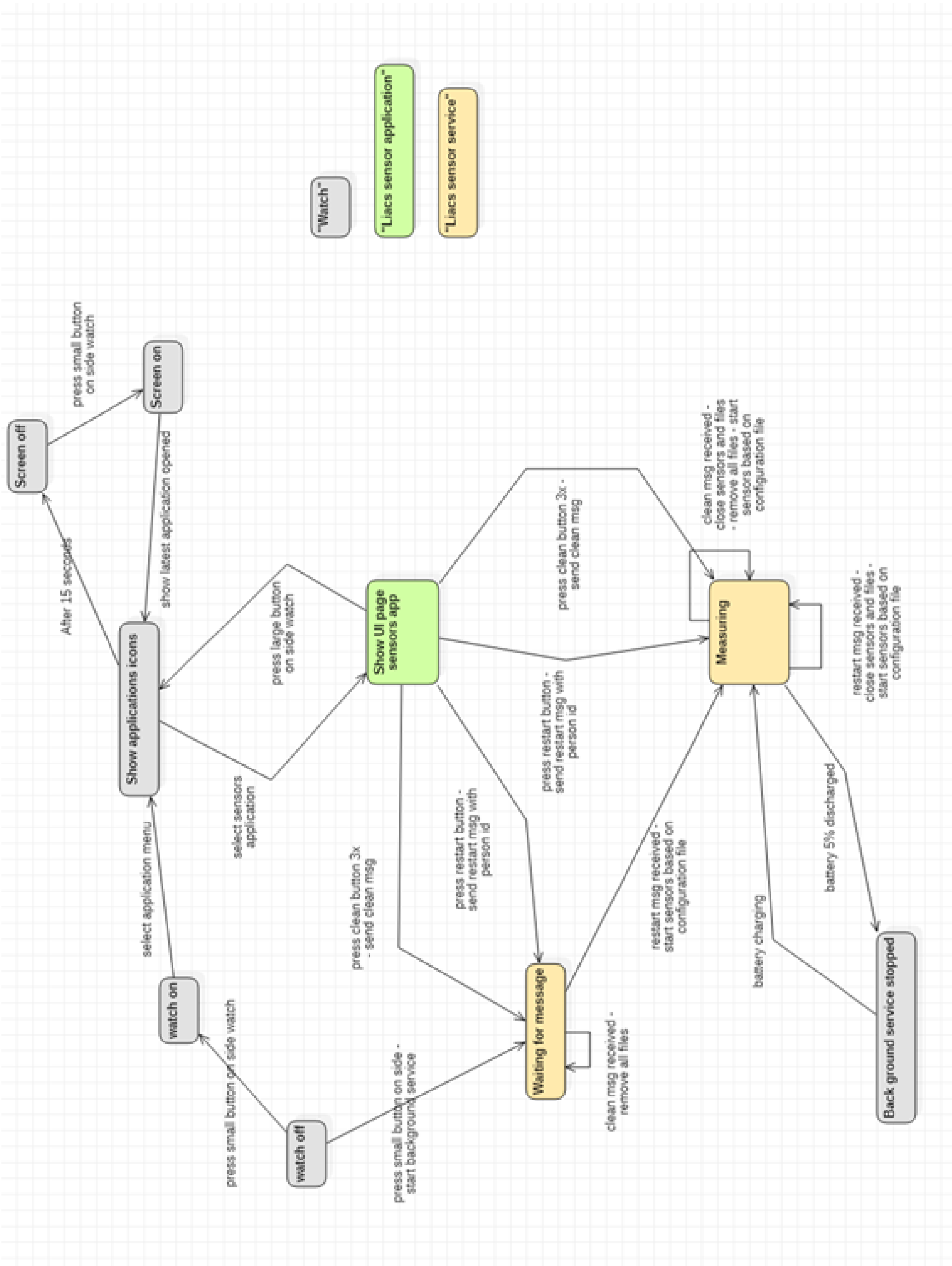




Here there is walking through a door, resting, walking again, resting again, and then stepped out of the building as GPS signal is picked up.

10 Appendix A

State Diagram, with transitions and format “event - actions”



11 Appendix B

Python Notebook, zero measurement G mean / standard deviation

```python
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt

import glob

#
# Determine the mean and std of a base line file
#
def determine_zero_measurement(file, zero, with_no_peaks):

    # Load file
    data = pd.read_csv(file, skiprows=1)
    data.columns = [s.strip() for s in data.columns]

    # Skip start and end distortions in base line measurements
    t0 = 13000
    t1 = 73000

    # G equation
    G = (np.sqrt(data['acce_x'][t0:t1]**2+data['acce_y'][t0:t1]**2+data['acce_z'][t0:t1]**2))

    # Remove peaks
    if (with_no_peaks):
        G_clean = G[ G < np.mean(G) + 4.0 * np.std(G) ]
        G_clean = G_clean[ G_clean < np.mean(G_clean) + 4.0 * np.std(G_clean) ]
        G_clean = G_clean[ G_clean < np.mean(G_clean) + 4.0 * np.std(G_clean) ]
        G_clean = G_clean[ G_clean < np.mean(G_clean) + 4.0 * np.std(G_clean) ]

    # peaks included
    else:
        G_clean = G

    zero['G mean'].append(np.mean(G_clean))
    zero['G std'].append(np.std(G_clean))
    zero['G threshold'].append(np.mean(G_clean) + 4.0 * np.std(G_clean))

    return

#
# Dictionary definition
#
zero = { "file": [], "watch id": [], "G mean": [], "G std": [], "G threshold": [] }

#
# Step through all files
#
for file in glob.glob("SensorFiles/*g.dat"):

    zero['file'].append(file)
    zero['watch id'].append(file.split()[1])
    determine_zero_measurement(file, zero, with_no_peaks = True)

#
# Convert to pandas data frame and generate csv file
#
pd_zero = pd.DataFrame(zero)
pd_zero.to_csv("Zero.csv")
```

12 Appendix C

Python Notebook, mean and standard deviation plots

```
# Plot G mean and G std
#
plt.rcParams["figure.figsize"] = (20, 12)

plt.figure()
pd_zero['G mean'].plot(x='time', y='G mean', kind='kde')
plt.legend()

plt.figure()
pd_zero['G std'].plot(x='time', y='G std', kind='kde')
plt.legend()
```