**ITRC Data Documentation**

|  |  |
| --- | --- |
| Title | ITRC Data Documentation v 1.0 |
| Document Number |  |
| Author (Organization) | Rischan, Advanced Network Lab, CNU |
| Project Team |  |
| Creation Date | 2015-02-16 |
| Last Modified | 2015-02-17 |
| Version | 1.0 |
| Status | First Version |
| Link | https://github.com/rischanlab/Rfunf |
| Path of Data | C://ITRC\_DATA (Rischan PC) |

**Revision History:**

|  |  |  |  |
| --- | --- | --- | --- |
| Modified By | Date | Version | Comments |
| Rischan | 2015-02-17 | 1.0 | First Version |
|  |  |  |  |
|  |  |  |  |

Contents

[Research Overview 4](#_Toc411971243)

[Application Data Collector 4](#_Toc411971244)

[Data Description 5](#_Toc411971245)

[On Request Data 8](#_Toc411971246)

[Simple Location Probe 8](#_Toc411971247)

[Nearby Wi-Fi Probe 9](#_Toc411971248)

[Nearby Bluetooth Probe 10](#_Toc411971249)

[Battery Probe 10](#_Toc411971250)

[Historical Data 11](#_Toc411971251)

[Call Log Probe 11](#_Toc411971252)

[Sms Log Probe 11](#_Toc411971253)

[Installed Application probe 11](#_Toc411971254)

[Hardware Info Probe 11](#_Toc411971255)

[Browser Bookmark Probe 11](#_Toc411971256)

[Browser Search Probe 11](#_Toc411971257)

[Contact Probe 11](#_Toc411971258)

[Continuous Data 12](#_Toc411971259)

[Light Sensor Probe 12](#_Toc411971260)

[Proximity Sensor Probe 12](#_Toc411971261)

[Temperature Sensor Probe 12](#_Toc411971262)

[Magnetic Field Sensor Probe 12](#_Toc411971263)

[Pressure Sensor Probe 12](#_Toc411971264)

[Screen Probe 12](#_Toc411971265)

[Running Application Probe 12](#_Toc411971266)

[Activity Probe 12](#_Toc411971267)

[Data Extraction 14](#_Toc411971268)

[Data Visualization 14](#_Toc411971269)

[Limitation 14](#_Toc411971270)

# Research Overview

Nowadays, smartphone capability has increased significantly. Smartphone has equipped with high processor, bigger memory, bigger storage and etc. With this equipment, smartphone has capability to running complex application. Many sensor also has embedded to the smartphone. With this sensor and log capability of smartphone, we can develop many useful system or application in different domain such as healthcare (elderly monitoring system, human fall detection) , transportation(monitoring road and traffic condition), personal and social behavior, environmental monitoring(pollution, weather), and etc. To develop such system, we have to collect the user personal data and then analyze it. There are two ways to collect personal data from the users based on user involvement, they are:

1. Participatory sensing
2. Opportunistic sensing

Participatory sensing means the application still need user's intervention to complete their task. The examples for such application need user to taking text input for each time period, taking picture and etc. On the other hand, opportunistic sensing means application does not need user's intervention to complete their task, users not involved in making decisions instead smart phone itself make decisions according to the sensed and stored data.

Our research focus on opportunistic sensing which is we develop application which does not need user’s intervention. To analyze the result, because of the data that we collected does not have any label so we prefer to use unsupervised learning.

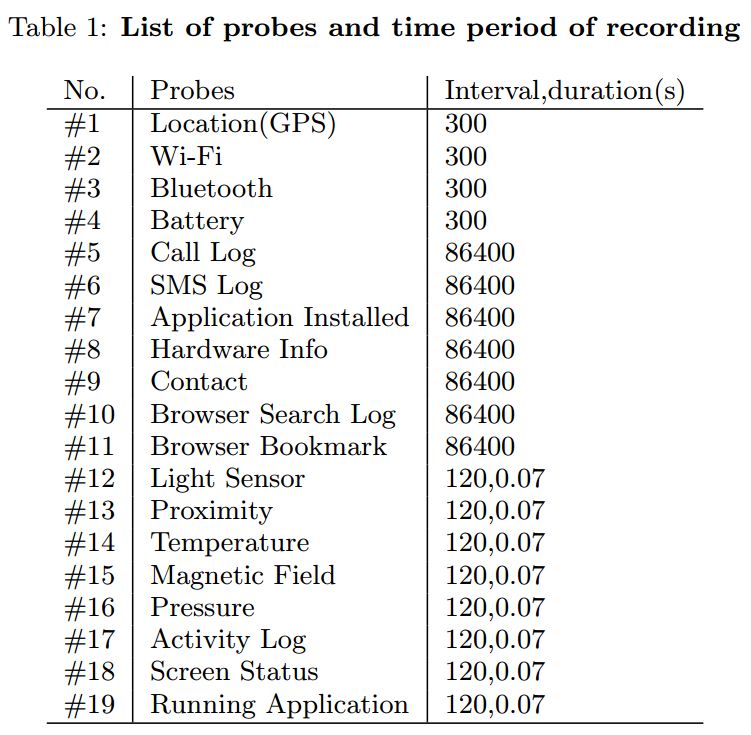
In this research we develop two systems are:

1. Application data collector
2. Data extraction and visualization

Application data collector is application that we used for collecting user’s personal data. This application is android application. After we have all of data from the user, we have to extract and visualize, and also analyze it. To extract, visualize, and analyze the data, we use R programing language.

# Application Data Collector

To develop application data collector, we do not develop from scratch, we use Funf library. The Funf Open Sensing Framework is an Android-based extensible framework, originally developed at the MIT Media Lab, for doing phone-based mobile sensing. Funf provides a reusable set of functionalities enabling the collection and configuration for a broad range of data types. Funf is open sourced under the LGPL license. Figure 1 shows Funf framework can collect many of sensing from smartphone such location, movement, communication and usage, social proximity, and many more. In this document, we do not describe details about Funf architecture but we describe about the data that we have collected and how to extract, visualize and analyze it. More details about Funf architecture we can visit the main site of Funf[[1]](#footnote-1) and also Funf developer site[[2]](#footnote-2).



# Data Description

Our application follows opportunistic sensing because we do not want to bothering user much. To do that we must define the time (interval and duration), when the application will request the data from the smartphone. Interval means how many times in second system will send data request to the smartphone. The example, we set interval 300 seconds means 5 minutes, so application will request and store the data for every 5 minutes. Duration is used in sensor data because without duration is useless to get the sensors data. The example, when we set interval 300 seconds and duration 0.07 s so application will send data request to the smartphone for every 5 minutes and the system will record the data during 0.07 seconds.

Table 1.shows the interval and duration from each probes. Those interval and duration already tested and we thought those setting was optimal but we can change those setting by change the value on the string.xml in android project. Figure 2a shows the string.xml file in the directory of android project and Figure 2b shows inside the string.xml file, we can change value of interval and duration in that file.

|  |  |
| --- | --- |
| C:\Users\rischan\Music\funf.png | D:\Dropbox\thesis\figures\dataviewinsmartphone.png |
| Figure 1. Funf Open Sensing Framework | Figure 3. Personal data in user’s smartphone |

To make easy for remembering, we classify the data to three of data categorization, are:

1. On Request Data (Current Data)
2. Historical Data (Saved in Android db)
3. Continuous Data (Sensor data)

On request data means we try to ask current values from android system such as location, battery, nearby Bluetooth and etc. Historical data means the data that already store in android database so we try to access and collect it, the example of historical data are contact, call log, sms log, and etc. Continuous data means we can get those data continuously such as sensor data (accelerometer, gyroscope, magnetic field, and etc). Another important thing is because we are living in time dimension space so every data has timestamp. Funf already has features to collect time, Funf using UNIX UTC (Coordinated Universal Time) which is ( Unix time or POSIX time or Unix timestamp) is the number of seconds that have elapsed since January 1, 1970. To convert UNIX time to the human readable time, we can use POSIX function in R or another programming language.

|  |  |
| --- | --- |
| D:\Dropbox\thesis\figures\ppt2\pptdata\sstringxml.JPG | D:\Dropbox\thesis\figures\ppt2\pptdata\funfsettingxml.JPG |
| (a) | (b) |
| Figure 2. (a) strings.xml file in project directory, (b) inside the string.xml file | |

Data that we collected using our application will be store in SQLite database format with (\*.db) extension can be seen in Figure 3. To open those database, we can use SQLite browser that can be download in SQLite browser main site[[3]](#footnote-3).

## On Request Data

Table 2.shows the table of On Request Data. The table contain four columns, \_id is automatically generated by database engine, name means the name of probes (sensors), timestamp column is time when system store the data to the phone’s storage, and value is the value that returned from the sensors. On request data has four of probes are location, nearby Wi-Fi, nearby Bluetooth, and battery.

Table 2. On Request Data Table

|  |  |  |  |
| --- | --- | --- | --- |
| **\_id** | **name** | **timestamp** | **value (JSON)** |
|  | SimpleLocationProbe | Unix UTC |  |
|  | WifiProbe |  |  |
|  | BluetoothProbe |  |  |
|  | BatteryProbe |  |  |

### Simple Location Probe

Location is one of the most important information from the user. In this research, we try to get the location information from the users.

{"mAccuracy":1625.0,"mAltitude":0.0,"mBearing":0.0,"mElapsedRealtimeNanos":21989372000000,"mExtras":{"networkLocationSource":"cached","networkLocationType":"cell","noGPSLocation":{"mAccuracy":1625.0,"mAltitude":0.0,"mBearing":0.0,"mElapsedRealtimeNanos":21989372000000,"mHasAccuracy":true,"mHasAltitude":false,"mHasBearing":false,"mHasSpeed":false,"mIsFromMockProvider":false,**"mLatitude":35.1837595,"mLongitude":126.9052379**,"mProvider":"network","mSpeed":0.0,"mTime":1403484137091},"travelState":"stationary"},"mHasAccuracy":true,"mHasAltitude":false,"mHasBearing":false,"mHasSpeed":false,"mIsFromMockProvider":false,"mLatitude":35.1837595,"mLongitude":126.9052379,"mProvider":"network","mSpeed":0.0,"mTime":1403484137091,"timestamp":1403484137.255}

A data from probes representing a geographic location. A location can consist of a latitude, longitude, timestamp, and other information such as bearing, altitude and velocity. All locations generated by the *LocationManager* are guaranteed to have a valid latitude, longitude, and timestamp (both UTC time and elapsed real-time since boot) and all other parameters are optional. In general, usually we use latitude and longitude to define the human location, but in this data we have many of data, another data such as accuracy, bearing, altitude, and elapse real time are explained below.

**Accuracy**

Get the estimated accuracy of this location, in meters. We define accuracy as the radius of 68% confidence. In other words, if you draw a circle centered at this location's latitude and longitude, and with a radius equal to the accuracy, then there is a 68% probability that the true location is inside the circle.

**Altitude**

Get the altitude if available, in meters above the WGS 84 (World Geodetic System) reference ellipsoid. If this location does not have an altitude then 0.0 is returned. The coordinate origin of WGS 84 is meant to be located at the Earth's center of mass; the error is believed to be less than 2 cm.

**Bearing**

Get the bearing, in degrees. Bearing is the horizontal direction of travel of this device, and is not related to the device orientation. If this location does not have a bearing then 0.0 is returned.

**Elapsed Real Time**

Note that the UTC time on a device is not monotonic: it can jump forwards or backwards unpredictably. So always use *getElapsedRealtimeNanos()* when calculating time deltas. On the other hand, *getTime()* is useful for presenting a human readable time to the user, or for carefully comparing location fixes across reboot or across devices.

More details about the key and values from the location probes can be seen in Android API documentation through this link.

http://developer.android.com/reference/android/location/Location.html#

### Nearby Wi-Fi Probe

Asss

{"BSSID":"b0:c7:45:7d:0f:7c"**,"SSID":"rischan"**,"capabilities":"[WPA2-PSK-CCMP+TKIP][ESS]","frequency":5180,**"level":-46**,"timestamp":1403476993.05}

**Capabilities**

Describes the authentication, key management, and encryption schemes supported by the access point.

**Frequency**

The frequency in MHz of the channel over which the client is communicating with the access point.

**Level**

The detected signal level in dBm, also known as the RSSI. Use calculateSignalLevel(int, int) to convert this number into an absolute signal level which can be displayed to a user.

### Nearby Bluetooth Probe

Asss

{android.bluetooth.device.extra.DEVICE":{"mAddress":"74:F0:6D:E8:ED:67"},"android.bluetooth.device.extra.NAME":"RRI-ITMS PC","android.bluetooth.device.extra.RSSI":-79,"timestamp":1404128054.397}

**android.bluetooth.device.extra.RSSI**

Used as an optional short extra field in ACTION\_FOUND intents. Contains the RSSI value of the remote device as reported by the Bluetooth hardware. Constant Value: "android.bluetooth.device.extra.RSSI“. More details about Bluetooth documentation can be seen in Android API documentation through this link

http://developer.android.com/reference/android/bluetooth/BluetoothDevice.html

### Battery Probe

Asss

{**"charge\_type":0**,**"health":2**,"icon-small":17303540, "level":89,"online":1, "scale":100,**"status":3**,"technology":"Li-ion","temperature":305,"timestamp":1403476991.281,"voltage":4138}

Charge Type value meaning

* BATTERY\_PLUGED\_AC =1
* BATTERY\_PLUGGED\_USB =2
* BATTERY\_PLUGGED\_WIRELESS=4

Status value meaning

* BATTERY\_STATUS\_CHARGING =2
* BATTERY\_STATUS\_DISCHARGING =3
* BATTERY\_STATUS\_FULL =5
* BATTERY\_STATUS\_NOT\_CHARGING =4
* BATTERY\_STATUS\_UNKNOWN =1

Health values meaning

* BATTERY\_HEALTH\_COLD =7
* BATTERY\_HEALTH\_DEAD =4
* BATTERY\_HEALTH\_GOOD =2
* BATTERY\_HEALTH\_OVERHEAT =3
* BATTERY\_HEALTH\_OVER\_VOLTAGE =5
* BATTERY\_HEALTH\_UNKNOWN =1
* BATTERY\_HEALTH\_UNSPECIFIED\_FAILURE =5

**Voltage**

Integer containing the current battery voltage level. Constant Value: "voltage".

More details about Battery documentation can be seen in Android API documentation in this link http://developer.android.com/reference/android/os/BatteryManager.html

## Historical Data

Table 3. Historical Data Table

|  |  |  |  |
| --- | --- | --- | --- |
| **\_id** | **name** | **timestamp** | **value (JSON)** |
|  | CallLogProbe | Unix UTC |  |
|  | SmsProbe |  |  |
|  | ApplicationsProbe |  |  |
|  | HardwareInfoProbe |  |  |
|  | BrowserBookmarksProbe |  |  |
|  | BrowserSearchesProbe |  |  |
|  | ContactProbe |  |  |

### Call Log Probe

### Sms Log Probe

### Installed Application probe

### Hardware Info Probe

### Browser Bookmark Probe

### Browser Search Probe

### Contact Probe

## Continuous Data

Table 4. Continuous Data Table

|  |  |  |  |
| --- | --- | --- | --- |
| **\_id** | **name** | **timestamp** | **value (JSON)** |
|  | LightSensorProbe | Unix UTC |  |
|  | ProximitySensorProbe |  |  |
|  | TemperatureSensorProbe |  |  |
|  | MagneticFieldSensorProbe |  |  |
|  | PressureSensorProbe |  |  |
|  | ScreenProbe |  |  |
|  | RunningApplicationsProbe |  |  |
|  | ActivityProbe |  |  |

### Light Sensor Probe

### Proximity Sensor Probe

### Temperature Sensor Probe

### Magnetic Field Sensor Probe

### Pressure Sensor Probe

### Screen Probe

### Running Application Probe

### Activity Probe

|  |  |  |  |
| --- | --- | --- | --- |
| Sensor | Type | Description | Common Uses |
| TYPE\_ACCELEROMETER | Hardware | Measures the acceleration force in m/s2 that is applied to a device on all three physical axes (x, y, and z), including the force of gravity. | Motion detection (shake, tilt, etc.). |
| TYPE\_LIGHT | Hardware | Measures the ambient light level (illumination) in lx. | Controlling screen brightness. |
| TYPE\_MAGNETIC\_FIELD | Hardware | Measures the ambient geomagnetic field for all three physical axes (x, y, z) in μT. | Creating a compass. |
| TYPE\_PRESSURE | Hardware | Measures the ambient air pressure in hPa or mbar. | Monitoring air pressure changes. |
| TYPE\_PROXIMITY | Hardware | Measures the proximity of an object in cm relative to the view screen of a device. This sensor is typically used to determine whether a handset is being held up to a person's ear. | Phone position during a call. |
| TYPE\_TEMPERATURE | Hardware | Measures the temperature of the device in degrees Celsius (°C). This sensor implementation varies across devices and this sensor was replaced with theTYPE\_AMBIENT\_TEMPERATURE sensor in API Level 14 | Monitoring temperatures. |

## Data Summarization

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Data ID** | **Size (MB)** | **Starting Point** | **Ending Point** |
| 1. | ENFP\_0719 | 628 | 6/30/2014 8:26 | 8/20/2014 0:18 |
| 2. | ENFP\_0773 | 664 | 6/26/2014 12:34 | 8/18/2014 4:58 |
| 3. | ENFP\_2012 | 661 | 6/27/2014 6:11 | 9/2/2014 3:57 |
| 4. | ENTJ\_5868 | 6890 | 6/27/2014 5:31 | 8/13/2014 0:00 |
| 5. | ENTJ\_6454 | 121 | 6/26/2014 5:32 | 8/6/2014 18:53 |
| 6. | ENTJ\_6966 | 272 | 7/2/2014 7:24 | 8/19/2014 11:22 |
| 7. | ENTP\_5623 | 455 | 6/30/2014 4:49 | 8/19/2014 20:57 |
| 8. | ESFJ\_2301 | 145 | 6/27/2014 5:31 | 8/20/2014 2:58 |
| 9. | ESFJ\_9284 | 158 | 6/26/2014 12:34 | 8/18/2014 4:58 |
| 10. | ESFP\_0912 | 278 | 6/26/2014 5:28 | 8/18/2014 8:53 |
| 11. | ESFP\_3295 | - |  |  |
| 12. | ESFP\_4634 | 486 | 6/27/2014 5:25 | 8/20/2014 4:10 |
| 13. | ESFP\_7467 | 607 | 6/26/2014 5:27 | 8/19/2014 7:18 |
| 14. | ESTJ\_0371 | 2390 | 7/3/2014 16:21 | 8/16/2014 21:03 |
| 15. | ESTJ\_3022 | 183 | 6/26/2014 5:28 | 8/18/2014 23:22 |
| 16. | ESTJ\_5071 | 1920 | 7/2/2014 2:34 | 9/11/2014 1:49 |
| 17. | ESTJ\_5190 | 258 | 7/30/2014 6:04 | 8/24/2014 1:43 |
| 18. | ESTJ\_5824 | 173 | 6/26/2014 5:29 | 8/18/2014 3:51 |
| 19. | ESTJ\_6510 | 756 | 6/27/2014 5:30 | 8/20/2014 8:09 |
| 20. | ESTP\_4301 | 232 | 6/26/2014 5:29 | 8/20/2014 4:39 |
| 21. | ESTP\_5154 | 990 | 6/27/2014 5:31 | 8/13/2014 0:00 |
| 22. | INFP\_1993 | 432 | 6/26/2014 5:31 | 8/20/2014 0:31 |
| 23. | INTJ\_5498 | 342 | 6/26/2014 5:28 | 8/20/2014 2:49 |
| 24. | INTJ\_7906 | 312 | 6/14/2014 11:00 | 8/16/2014 23:01 |
| 25. | INTP\_3739 | 1030 | 6/27/2014 5:28 | 8/18/2014 5:58 |
| 26. | INTP\_6399 | 199 | 6/26/2014 5:29 | 8/12/2014 8:32 |
| 27. | INTP\_9712 | 180 | 6/26/2014 5:37 | 8/16/2014 18:05 |
| 28. | ISFJ\_2057 | 183 | 6/27/2014 5:32 | 8/14/2014 23:19 |
| 29. | ISFJ\_2711 | 767 | 7/31/2014 0:51 | 8/20/2014 6:59 |
| 30. | ISFJ\_7328 | 133 | 6/30/2014 7:09 | 8/19/2014 23:37 |
| 31. | ISFP\_4030 | 2380 | 6/27/2014 6:11 | 9/2/2014 3:57 |
| 32. | ISFP\_4282 | 613 | 6/27/2014 5:27 | 8/20/2014 2:46 |
| 33. | ISTJ\_0178 | 158 | 6/26/2014 5:28 | 8/19/2014 5:05 |
| 34. | ISTJ\_0386 | 284 | 6/26/2014 5:27 | 8/19/2014 7:18 |
| 35. | ISTJ\_2068 | 339 | 6/26/2014 5:29 | 8/18/2014 5:30 |
| 36. | ISTJ\_2837 | 186 | 6/27/2014 5:27 | 8/22/2014 5:41 |
| 37. | ISTJ\_3052 | 131 | 6/27/2014 5:27 | 8/20/2014 3:41 |
| 38. | ISTJ\_4659 | 325 | 7/2/2014 2:34 | 9/11/2014 1:49 |
| 39. | ISTJ\_4667 | 156 | 6/26/2014 5:29 | 8/15/2014 10:44 |
| 40. | ISTJ\_4700 | 170 | 7/3/2014 6:50 | 8/25/2014 13:08 |
| 41. | ISTJ\_4753 | 363 | 6/26/2014 5:29 | 8/18/2014 23:48 |
| 42. | ISTJ\_4968 | 95 | 7/3/2014 16:21 | 8/16/2014 21:03 |
| 43. | ISTJ\_9139 | 473 | 7/3/2014 16:21 | 8/20/2014 5:57 |
| 44. | ISTJ\_9576 | 198 | 7/4/2014 1:00 | 8/18/2014 7:12 |
| 45. | ISTP\_3948 | 500 | 6/26/2014 5:29 | 8/20/2014 1:28 |
| 46. | ISTP\_7676 | 365 | 6/27/2014 5:31 | 8/19/2014 22:11 |
| 47. | XXXX\_XXXX | 434 | 6/27/2014 5:31 | 8/21/2014 6:02 |

# Data Extraction

Our application still need many of improvements, we have some of limitation in this application as follows:

# Data Visualization

Our application still need many of improvements, we have some of limitation in this application as follows:

# Limitation

Our application still need many of improvements, we have some of limitation in this application as follows:

This application does not have function that can send automatically the data from user to the server. Current version still need USB to copy the data from user’s phone to the PC server.

This

1. http://www.funf.org/ [↑](#footnote-ref-1)
2. https://code.google.com/p/funf-open-sensing-framework/wiki/FunfArchitecture [↑](#footnote-ref-2)
3. http://sqlitebrowser.org/ [↑](#footnote-ref-3)