

# Whereabouts Clock

Byeonggon Lee, Jaehyun Byeon, Sowon Park, Juhyeok Bae

June 11, 2017

**abstract-** Overall, our real time location clock has two purposes. First, by informing families at home of each member's location, they can be relieved. Second, by automatically showing information at clock, people can be aware of each other's location fast and conveniently than using a phone call or messenger. We provide this service with smart phone application and clock equipped with Raspberry pi. Application server compares initial user's setup data and their real time location using Own tracks application. If the data corresponds to the setup data stored at Database, Application Server sends request to change the hands of clock which indicate family member's location. Our clock provides each locations of member and reduces concerns about safety.

#### Role Assignment

Roles <sup>↗</sup>	Name <sup>↗</sup>	Task description and etc. <sup>↗</sup>
User <sup>↗</sup>	Sowon Park <sup>↗</sup>	Supposing herself as an end-user of this software, listed both predictable inconveniences and desired properties. <sup>↗</sup>
Customer <sup>↗</sup>	Jaehyun Byeon <sup>↗</sup>	Make concrete which functions are necessary. When user purchase this software, customer analysis the benefits of this software. <sup>↗</sup>
Software developer <sup>↗</sup>	Juhyeok Bae <sup>↗</sup>	Plan how to implement this software, consider the algorithm and system within software. <sup>↗</sup>
Developer manager <sup>↗</sup>	Byeonggon Lee <sup>↗</sup>	Design a general idea of this software, managing our team. <sup>↗</sup>

Figure 1: Catpion1

## 0.1 Introduction

Modern society is composed of various social groups. From birth, we belong to a family and get to be part of a peer group as we grow up. The relationship between the group members has been more important since we usually have a strong sense of belonging to it. Among many social groups, a family would be the most crucial social group because we naturally belong to it from birth. Members of a family share a strong bond which is connected emotionally. When children are young, there are usually a lot of opportunities to spend time with their parents. However, as children grow up to be adults and can take care of themselves, it becomes difficult to have enough face-to-face conversations for a family. Both sides often try to keep their sense of family's closeness by phone calls or sending messages. Therefore, the most common question in a family would be asking where the other person is. This is because the most important thing for a family is consistently checking the safety of each other. With this mind, our team has come up with an idea to provide a service that can let the family members know the location of each other easily. There are already many applications in the field that someone can track the exact location of family members. We decided to use that service as our basic idea and come up with something that can present the information in visual forms and sounds. The technology would be applied to the clock because it is easy for us to see in their everyday lives. Whereabouts clock will show the location of each family member instead of only showing the physical time as normal clock does. Our location clock service consists of these things as stated below. The hands of the clock will represent the family members. The clock will use a hardware called the raspberry pi to get information from the application. The application will retrieve information data using the 'Own tracks' application. The information of the users will be stored in the database through the server where their location data would be checked in real-time. The specific fundamentals of this clock's IOT function would be presented more clearly in the specification sector. We would also like to further implement this technology to not only just family members but also to others in need. This clock service could be used in places such as hospitals, or facilities that should take care of young or old peo-

ple. As the number of caretakers of such places are limited and they still need a lot of help, we would like to provide a service that would aid them with informing the location of the people through visual and sound methods using a form of a clock.

## **0.2 Requirement**

### **0.2.1 Requirement for Clock**

#### **0.2.1.1. Clock kit**

- An old beat up clock or old clock shell
- 1 raspberry pi Micro-Computer
- 3 Sail winch servos (special servos that can turn 360 degrees)
- A middle device that connects between raspberry pi and 3 Sail winch servo
- 3 Brass Tubes (one is 1/4 inch, another is 9/32 inch, the other is the smallest one) and 6 gear
- For additional function, 1 display panel and 1 speaker

#### **0.2.1.2. Network**

- To connect raspberry pi with Wi-Fi, we should connect Wi-Fi dongle with raspberry pi

#### **0.2.1.3. Receiving data**

- Connecting the Wi-Fi to receive the GPS data from the Application with clock
- Connecting the Wi-Fi to receive the short message that user send

#### **0.2.1.4. Handling input data**

- To judge where the clock hands should move to

#### **0.2.1.5. Moving clock hands**

- Move clock hands in real time

#### **0.2.1.6. Floating the message**

- Floating the short message that family sends shortly

#### **0.2.1.7. Alarm**

- If the family location changes, clock alarms the user

#### **0.2.1.8. Informing the real time**

- Display panel on the bottom showing the real time
- Ex) 2017.04.01 10 pm

### **0.2.2 Requirement for Application**

#### **0.2.2.1. Install application**

#### **0.2.2.2. Sign in and login**

#### **0.2.2.3. Permitting to give users GPS data.**

#### **0.2.2.4. Making the family group**

- Set the family number
- The main user invites other family member
- Connecting with the clock

#### **0.2.2.5. Setting the default location information ex) school – GPS data**

#### **0.2.2.6. Comparing the real time GPS data with default location data**

- Handling between the real time GPS data and default location information that setting

#### **0.2.2.7. Transmit data that compare between the real time GPS data and default location data to clock**

- If the status of one application is on, the application server sends a GPS data to clock

#### **0.2.2.8. Sending user message and setting states**

- User can send a short message to the clock screen.
- When user click the button which represents location state like (Moving, Studying, Working, and Taking a rest or so on), application changes the clock screen display.

### **0.2.3 Requirement for User**

#### **0.2.3.1. Install Whereabouts clock**

- User is required to get the clock.
- User puts a battery in the clock.

#### **0.2.3.2. Set up family data with hands of clock.**

- User makes each hand of clock indicate family members.
- User puts place to confirm family's location instead of time in clock.

#### **0.2.3.3. Join and sign in smartphone application**

- Family members are required to make an account with his/her email address or phone number and password in the application.
- User can sign in with the setting of email address or phone number and password that he/she registered at the first time.

#### **0.2.3.4. Make a group in the smartphone application**

- One of family member should invite the others to group setting.

#### **0.2.3.5. Connect the setting between members and hands of clock**

- Each member set the application to make watch point themselves.

#### **0.2.3.6. Register user's clock**

- To connect clock with the application, user should register clock and network path. Application considers where and how to send the input data.

#### **0.2.3.7. Create user settings in Database**

- User should register the setup location which they want to represent in Database. If they want to set their home, they can store home address in the application and the server store as they received data in database.
- User also can register other locations they want to confirm. For example, there could be parent's company locations, children's school locations, children's private educational institutes and so on.

#### **0.2.3.8. Register basic application settings**

- User should set the checking time (for 30 seconds of one minute ) when the application retrieves user's real time location.
- User should allow the application to take user's GPS data.

#### **0.2.3.9. Click the state button**

- When user want to represent their current conditions to other family's members, he or she can press the button which represents their present states.

#### **0.2.3.10. Listen to alarm**

- Even if someone is concentrating on his/her working, they can also hear clock alarming.
- If user's location is changed, the hand of clock also moved to point the location on the clock and then clock alarms.

## **0.2.4 Requirement for Developer**

### **0.2.4.1. Raspberry Pi**

- Raspberry Pi is one of important components of Whereabouts Clock. It is core part of the clock and gets the comparing data between initial setup and real time locations. It helps application to send data to clock. And then manipulates stepping motor which can handle hands of clock.
- Developer should set hands of clock about hands' turning angle and its velocity.

### **0.2.4.2.Noobs**

- Noobs help user to install OS they want. According to developer's taste, developer could choose one of OS like Raspbian, Openexel, window 10 IoT and so on. We use Raspbian to develop our clock.

### **0.2.4.3. Connecting Wi-Fi with Raspberry Pi**

- Wi-Fi chip equipped with Raspberry Pi 3 can make Raspberry Pi access wireless router. Through the wireless router, Raspberry Pi connects user's application. Developer makes Wi-Fi chip automatically link wireless router through Noobs.

### **0.2.4.4. Connecting Stepping motor 5V in Raspberry Pi**

- To operate hands of clock correctly, Raspberry Pi uses Stepping motor which spins axis of clock. By turning the axis, each hand of clock indicates location. We use three Stepping motor connecting axis with gears and each axis works respectively.

### **0.2.4.5. Middle equipment, Arduino**

- Between Raspberry Pi 3 and three motors, developer installs Arduino which gets input data from Raspberry Pi and operates each motors.

### **0.2.4.6. Support user to choose one of motors**

- Show Stepping motor's number to see user deciding to which motor is used in application.

### **0.2.4.7. SD Card in Raspberry Pi**

- It is a storage to install Raspbian in SD Card which stores Operating System.

### **0.2.4.8. GPS Tracking API**

- To get real time location and GPS, developer should use tracker which get real time location with GPS tracking open source.

### **0.2.4.9. Use server**

- To manage user's data and store initial setup which user wants, developer makes an EC2 account to use server in AWS.
- To compare between user's real location and initial setup, developer uses server.

### **0.2.4.10. Connect between Application, Server and Database.**

- To interact with each level, developer should make communication environment.

### **0.2.4.11. MYSQL**

- To help server find stored setup data, developer should make a MYSQL in android studio. Developer could also manage all the data stored in the database.

### **0.2.4.12. Android studio**

- Developer uses android studio for writing a code in android application.

## 0.3 DEVELOPMENT ENVIRONMENT

### 0.3.1 Which platform and why?

#### 0.3.1.1. Windows for android application developing

- We will use Windows operating system. Windows OS is the most popular operating system being used worldwide and three of us also use Windows. As a general rule, Linux or Window OS is a better fit for the development environment. Also, we decided to use the Windows, which is the most familiar OS since we don't have much to do with Android.

#### 0.3.1.2. Linux for server developing

- Three of us will use Ubuntu, a type of Linux, to develop a server. Globally, Linux operating system is the most widely used server. Also, it was appropriate because our open source, Traccar, is also offering a Linux version.

#### 0.3.1.3. Raspbian for Raspberry Pi developing

- We will use Raspbian OS because Raspbian OS is convenient and comfortable in Raspberry Pi.

### 0.3.2 Which programming language and why?

#### 0.3.2.1. Java for android application developing

- Java is a general-purpose computer programming language that is concurrent, class-based, object-oriented, and specifically designed to have as few implementation dependencies as possible. As of now, Java is one of the most popular programming languages in use, particularly for client-server web applications, with a reported 9 million developers. Also, because Java can minimize the problem of fragmentation by hardware platform due to existence of JVM, Android chose Java. So

now all Android development is done through Java.

#### 0.3.2.2. Provide a cost estimation for your built

- Python is simple and productive Programming language. This language is made by Netherlands developer. Python language's grammar is simple and looks like human's thinking. Therefore, it has advantages for beginners to learn easily. Python helps people to develop Web Service, Data Analysis, Machine Learning. Python is also used for Raspberry Pi. It is convenient and comfortable for Raspberry Pi beginners.

#### 0.3.2.3. MYSQL for using database

- MYSQL is the world's most popular database management system. It uses SQL Language and RDBMS. It is very fast and flexible. And it is easy for beginners to use Database. We chose MYSQL because it supports our Traccar Application

#### 0.3.2.4. Java and Python for managing Server

- It is composed of Java and Python in Traccar Server side.

### 0.3.3 Provide clear information of your development environment

#### 0.3.3.1. Cost for Server

- Amazon EC2 is free for a year.

#### 0.3.3.2. Cost for hardware

##### 1. Raspberry Pi 3B

- We bought a Raspberry Pi 3 because Raspberry Pi 3 supports Wifi and Bluetooth functions. And it supports the 4 USB ports, Ethernet ports and HDMI port that is needed for using the monitor.

2. Stepping Motors
  - Stepping Motor is the Connector which is used for Whereabouts clock. We use a Raspberry Pi 3B for turning the Stepping Motor.

3. Raspberry Pi case
  - Raspberry Pi case protects the Raspberry Pi from the Outer attack and electromagnetic waves.

4. Bread board
  - Bread board is installed for connecting the Raspberry Pi and Stepping Motor.

5. SD card
  - Raspbian OS is installed in SD card. We can use SD card in a Raspberry Pi 3B.

6. Monitor
  - Monitor is used for coding the program and checking the result. Our team borrow the Monitor from acquaintance.

7. HDMI cable
  - Raspberry Pi 3 supports HDMI for Monitor. So we should use a HDMI cable for Monitor.

8. HDMI to DVI converter
  - Our team's Monitor supports DVI. So we should connect between HDMI and DVI. We bought a HDMI to DVI converter.

9. Keyboard
  - We use a keyboard because Raspberry pi support a USB port for keyboard. Raspberry pi 3B doesn't support a notebook for monitor.

10. Mouse
  - We use a mouse because Raspberry pi support a USB port for mouse.

11. Clock
  - We use a clock that is used for ten years

from one of our team member's home. We will change the clock to Whereabouts clocks that uses Raspberry Pi 3B.

12. 2 Brass Tubes and 4 gear
  - We buy 2 Brass Tubes and 4 gear because we need two axis and power that got from the stepping motor.

#### 0.3.3.3. Cost Table

Products	Prices	Quantities	Costs
Raspberry PI	45000	1	45000
Stepping Motor	5400	2	10800
Raspberry Pi case	9000	1	9000
Bread board	4460	1	0
SD card	10600	1	10600
Monitor	35000	1	0
HDMI cable	4000	1	4000
HDMI to DVI converter	10000	1	10000
Keyboard	9900	1	0
Mouse	15000	1	0
Clock	30000	1	0
2 Brass Tubes and 4 gears	10000	1	10000
<b>TOTAL</b>	<b>99400</b>		

Figure 2: Caption2

#### 0.3.3.4. Using any commercial cloud platform

1. Android developing
  - Windows: 10
  - Android Studio: 2.3.1
2. Server developing
  - Ubuntu: 14.04.3 LTS

- Java: 1.7.0.121
- Python: 3.4.3

3. Raspberry Pi developing
  - Raspbian: 4.4

#### 0.3.3.5. Which member is responsible for what?

1. Amazon's EC2
2. Software in use

a. 'Life 360' Family location Tracking



Figure 3: Caption3

It is an application that shows each family member's location for other family users. It can identify the family location at the same time.

b. Traccar API

Traccar API is tracing the location and showing the coordinates. This Traccar API is provided for Open Source.

c. Google Maps API

Google Maps API provides a map that User wants to see a coordinates that people is located.

d. RPI.GPIO modules

We use a RPI.GPIO modules for controlling the GPIO. RPI.GPIO modules can control the stepping motor that is used for our Whereabouts clock.

e. Github

Github was the environment we had to be used to. It is a free cooperation development tool for software development. It is a hosting service based on git, gathering all kinds of source.

- Pull request
- Schedule management, report bug.
- Co-working is possible for other team as well as my team.

f. Sublime text

Recently a lot of people use Sublime text. Sublime Text is a sophisticated text editor for code, markup and prose. Also, people can add any kind of functions other people made. Here are the features of Sublime Text include the following:

- Goto Anything
- Command Palette
- Split Editing
- Customize Anything
- Multiple Selections
- Distraction Free Mode



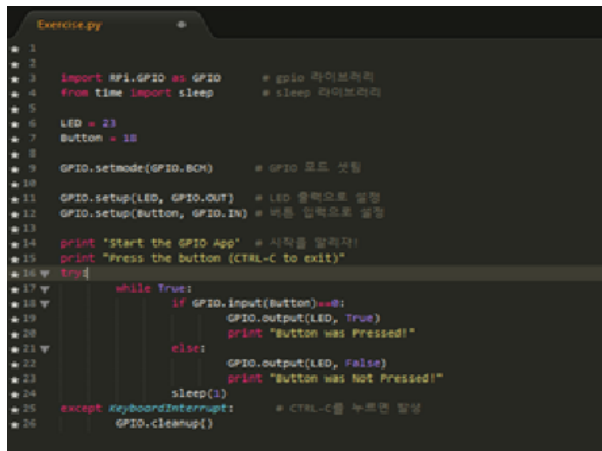


Figure 4: Caption4

g. Android Studio

Android Studio is the official integrated development environment (IDE) for the Android platform. Based on JetBrains' IntelliJ IDEA software, Android Studio is designed specifically for Android development.



Figure 5: Caption5

3. Task distribution (We will provide detail about this table later at the next design phase)

## 0.4 SPECIFICATIONS

### 0.4.1 Modeling for Specifications

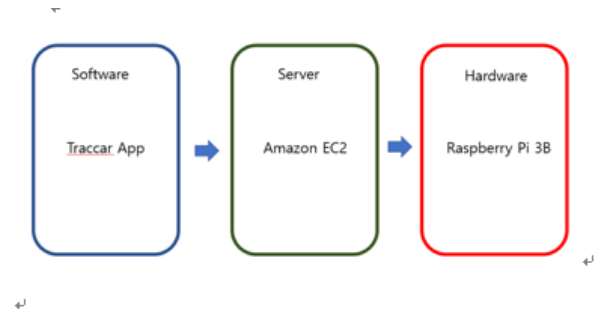


Figure 6: Caption6

After receiving User's location in Traccar App, we provide this information to Amazon EC2. Then Amazon EC2 compares the user's location and default place location coordinates. And then the result is provided to Raspberry Pi 3.

1. find out User's location in Traccar App.
2. deliver User's location to Amazon EC2.
3. Compare the user's location and default place location
4. If the result is changed, Amazon EC2 deliver the result to Raspberry Pi.
5. Raspberry Pi is controlling the stepping motor.
6. Whereabouts clock is turning according to User's location.

### 0.4.2 Specification Application

#### 0.4.2.1. Sign in and login

- It is the first page of the Whereabouts application. After entering the member ID, password, name, name, position of the household, and serial number of the clock. Once the sign-in is signed, the login window appears. We should always place a check box for keeping the log in.

#### 0.4.2.2. Permitting to give users GPS data

- Get permission to receive GPS data from your mobile device directly after you log in. If rejected, close down the app with a message stating that the application cannot proceed.

#### 0.4.2.3. Setting the default location information ex) school – GPS data

- Provide a map API to set the default location such as school, workplace, and home. Also, they enter the radius to be recognized as the same place. Enter this location name, latitude, and longitude to send to the server.

#### 0.4.2.4. Settings and On/Off for location information

- Display the screen where you can set the server address, port number, location transfer time frequency, and location provider as shown in the following figure. Users can also control On/Off on this screen.

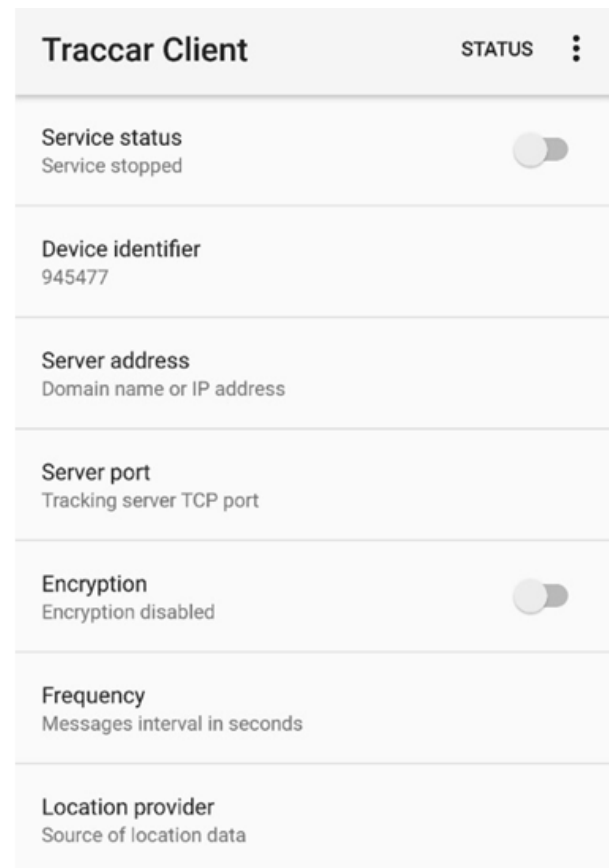


Figure 7: Caption7

#### 0.4.2.5. Transmitting GPS Information to Server

- Sends a location to the server at regular intervals to accommodate the existing server address and frequency. Applications are implemented to operate in the background even if they are not opened on the screen of the mobile device.

#### 0.4.2.6. Members of Family List

- Display the list of users whose names match with serial number of the clock and the corresponding user name of the clock that the member entered. Here, you can write and share a short state message within 30 characters.

### 0.4.3 Specification Server

#### 0.4.3.1. Service start

- Access the Amazon EC2 Ubuntu environment using Putty. Then, launch the service by running startDaemon.sh in the /opt/traccar/bin folder

#### 0.4.3.2. Sign - up and login processing

- When registering a user's membership, check whether the ID and the position of family are overlapped or not, and if a valid serial number are entered. Also, verify the identity of the ID and password when signing.

#### 0.4.3.3. Receiving GPS data and storing

- Input the ID, latitude, longitude, and serial number from the mobile application and store them on the positions table in the MySQL.

#### 0.4.3.4. User position determination

- Store the default location from the mobile application on the locations table. After comparing it to the location of the current user's latitude, longitude determine the user's place. Send data to the clock if the user changes the location of the user.

#### 0.4.3.5. Transmitting the data to the clock

- Find the clock serial number of a family member whose location changes occurred. Then use a protocol which is set beforehand to notify the change of location to the Raspberry Pi in the corresponding clock

#### 0.4.4.2. Handling input data

- If Raspberry Pi 3 notice the change, Raspberry Pi 3 can control the GPIO modules. We use a RPi.GPIO modules using Python programming language. RPi.GPIO module controls the GPIO that uses to control the stepping motor.

```
Exercise.py
★ 1
★ 2
★ 3 import RPi.GPIO as GPIO
★ 4 from time import sleep
★ 5
★ 6 GPIO.setmode(GPIO.BCM)
★ 7
★ 8 GPIO.setup(Motor, GPIO.OUT)
★ 9 GPIO.setup(Button, GPIO.IN)
★ 10
★ 11 print 'Start the GPIO App'
★ 12 print "Press the button (CTRL-C to exit)"
★ 13 try:
★ 14     while True:
★ 15         if GPIO.input(Button)==0:
```

Figure 9: Caption9

### 0.4.4 Specification Server

#### 0.4.4.1. Receiving data from Amazon EC2 server

- We receive user's location from Amazon EC2 server. Originally, User location is received from Traccar App. Traccar App gives a Real User's location to Amazon EC2 server. And Amazon EC2 compares Real User location and default place location information. If the information compared to using Amazon EC2 is changed, Raspberry Pi 3 can notice the change.

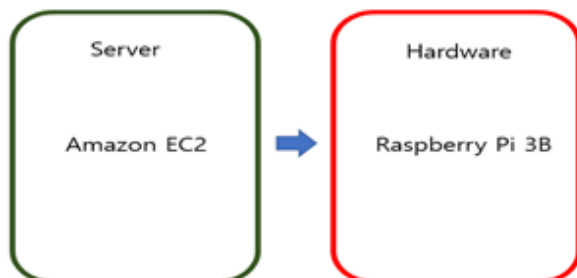


Figure 8: Caption8

#### 0.4.4.3. Moving clock hands

- If the GPIO module controls the stepping motor, stepping motor turns to destination that indicates User Real location. Concretely, Stepping Motor turns a gear that connects from Brass Tubes. The gear makes clock spin to turn heading for the destination.



Figure 10: Caption10

## 0.5 Architecture

### 0.5.1 Overall Architecture

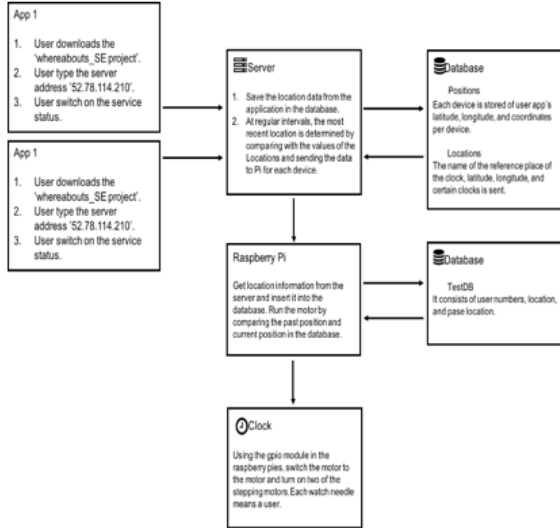


Figure 11: Caption11

#### 0.5.1.1. App 1, App 2

- User downloads the 'whereabouts SE project'
- User type the server address '52.78.114.210'
- User switches on the service status.

#### 0.5.1.2. Server

- Save the location data from the application in the database.
- At regular intervals, the most recent location is determined by comparing with the values of the Locations and sending the data Pi for each device.

#### 0.5.1.3. Server Database

- Positions DB: Each device is stored of user app's latitude, longitude, and coordinates per device.
- Locations DB: the name of the reference place of the clock, latitude, longitude, and certain clocks is sent.

#### 0.5.1.4. Raspberry Pi

- Get location information from the server and insert it into the database. Run the motor by comparing the past location and current position in the database

#### 0.5.1.5. Raspberry Pi Database

- TestDB: It consists of user numbers, location and past location.

### 0.5.2 Directory architecture

Directory ↗	File names ↗	Module names ↗	Language ↗
/traccar ↗	tracker-server.jar ↗	Server ↗	Java ↗
↗	wrapper.jar ↗	Server ↗	Java ↗
↗	wrapperApp.jar ↗	Server ↗	Java ↗
/traccar/bin ↗	startDaemon.sh ↗	Server ↗	Shell Script ↗
↗	stopDaemon.sh ↗	Server ↗	Shell Script ↗
/traccar/conf ↗	traccar.xml ↗	Server ↗	XML ↗
↗	wrapper.conf ↗	Server ↗	↗
/traccar/lib ↗	↗	Server ↗	Java ↗
/traccar/logs ↗	↗	Server ↗	↗
/traccar/schema ↗	↗	Server ↗	XML ↗
/traccar/templates ↗	↗	Server ↗	Java ↗
/traccar/t ↗	↗	Server ↗	↗

Figure 12: Caption12

mp.			
/traccar/ web.		Server.	HTML CSS, JavaScr ipts.
/traccar- client- android.	build.gradle.	APP.	Java.
.	gradlew.	APP.	Java.
.	gradlew.bat.	APP.	Java.
.	settings.gradle.	APP.	Java.
.	translate.py.	APP.	Python.
/traccar- client- android/a pp.		APP.	Java.
/traccar- client- android/a pp/src.		APP.	Java.
/traccar- client- android/a pp/src/hi dden.		APP.	Java.
/traccar- client- android/a pp/src/m ain.		APP.	Java.
/traccar- client- android/a pp/src/re gular/res.		APP.	Java.
/traccar- client- android/a pp/src/re gular/test /java/org/ traccar/cl ient.		APP.	Java.

Figure 13: Caption13

/.	testReturnServer.py.	Server.	Python.
/.	ReceivingTest_Client.py.	Server.	Python.
/.	SocketToReceiverTest_ Client.py.	Server.	Python.
/.	Clock_project4.py.	Raspberry pi.	Python.

Figure 14: Caption14

### 0.5.3 Software Component

#### 0.5.3.1. Software Component 1: Application

1. Purpose
  - Application is useful for delivering user's location in Server. Our Whereabout clock catches user's location using GPS.
2. Functionality
  - Application delivers Id, protocol, deviceid, servertime timestamp, fixtime, valid, latitude, longitude, altitude, speed, course, address, attributes, accuracy, network attributes in Server Database 'Position' Table.
3. Location of Source Code
  - /traccar-client-android
4. Class Components
  - 4-1. Bulid.gradle
    - CompileSdkVersion = 25
    - TargetSdkVersion 25
    - MinSdkVerison = 3

```

apply plugin: 'com.android.application'

android {
    compileSdkVersion 25
    buildToolsVersion '25.0.2'

    defaultConfig {
        minSdkVersion 3
        targetSdkVersion 25
        versionCode 34
        versionName '4.1'
    }

    lintOptions {
        checkReleaseBuilds false
    }

    productFlavors {
        regular {

```

Figure 15: Caption15

#### 4-2. PositionProvider

- 'locationManager = (LocationManager) context.getSystemService (Context.LOCATION\_SERVICE)' is a source code that get actually user's location using Applicatoin

```

PositionProvider | PositionListener | onPositionUpdate()

private long lastUpdateTime;

public PositionProvider(Context context, PositionListener listener) {
    this.context = context;
    this.listener = listener;

    preferences = PreferenceManager.getDefaultSharedPreferences(context);
    locationManager = (LocationManager) context.getSystemService(Context.LOCATION_SERVICE);

    deviceId = preferences.getString(MainActivity.KEY_DEVICE, null);
    period = Long.parseLong(preferences.getString(MainActivity.KEY_INTERVAL, null)) * 1000;

    type = preferences.getString(MainActivity.KEY_PROVIDER, "gps");
}

public abstract void startUpdates();

```

Figure 16: Caption16

#### 5. Where it is taken for?

= <https://github.com/tananev/traccar-client-android.git>

#### 6. How/Why you used it?

- Users can use this app to display location on the clock. Specifically, Users turns on the App and register Server IP address in this App. This app sends user's latitude and longitude to the server. So, the server will then receive

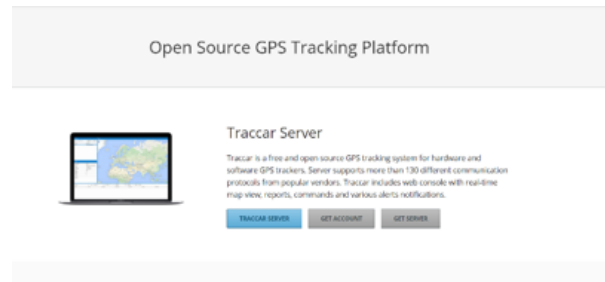


Figure 17: Caption17

the latitude and longitude of the user

#### 0.5.3.2. Software Component 2: Server

##### 1. Purpose

- The server receives the user's location through the app. Then the server stores the user's location in its database. It processes the user's location and data on the server itself. This will allow to send a user data and user location to Raspberry pi 3.

##### 2. Functionality

- Server will do many things. The first thing is to store the user's location in the database through the app. And Server picks up the user's location from the database and handles the value. Processed values will be sent to the raspberry pi 3 Server.

##### 3. Location of Source Code

- /traccar

##### 4. Class Components

##### 4-1. startDaemon.sh

- This is portion that the server operates. This file allows the user to store the location of the users that Android delivers to the server.

```

#!/bin/bash
# start YAJSW daemon script
#
#-----#
# resolve links - $0 may be a softlink
PRG="$0"
while [ -h "$PRG" ] ; do
  ls=`ls -ld "$PRG"`
  link=`expr "$ls" : '.*/\([^/]\*$'`
  if expr "$link" : '/..' > /dev/null; then
    PRG="$link"
  else
    PRG=`dirname "$PRG" "$link"
  fi
done
PRGDIR=`dirname "$PRG"
EXECUTABLE=startDaemonNoPriv.sh

# set java and conf file
source "$PRGDIR"/setenv.sh
export PRGDIR

# Check that target executable exists
if [ ! -x "$PRGDIR"/"$EXECUTABLE" ]; then
  echo "Cannot find $PRGDIR/$EXECUTABLE"
  echo "This file is needed to run this program"
  exit 1
fi

sudo "$PRGDIR"/"$EXECUTABLE"

```

Figure 18: Caption18

#### 4.2. testReturnServer.py

- This is the portion of getting the user's location that is recalled from the Potion database table. The User's location is sent to raspberry pi after processing user's location in an easy manner.

```

db = MySQLdb.connect('localhost','root','0593','traccardb1')
cursorPositions = db.cursor()
cursorLocations = db.cursor()

cursorPositions.execute("select * from (select deviceid,MAX(devicetime) as devicetime
positionRows = cursorPositions.rowcount
conn.send(str(positionRows))

for j in xrange(positionRows):
  lastdata = cursorPositions.fetchone()
  usr_id = lastdata[2]
  usr_pos = (lastdata[4],lastdata[5])

  cursorLocations.execute("select * from locations")

  for i in xrange(cursorLocations.rowcount):
    place,loc_lat,loc_longi,scope,clock = cursorLocations.fetchone()
    loc_pos = (loc_lat,loc_longi)
    distance = vincenty(usr_pos,loc_pos).meters
    if i==0:
      nearest = (place,distance,distance+scope)
    elif(distance<nearest[1]):
      nearest = (place,distance,distance+scope)

  msg = str(usr_id)+" "+nearest[0]+" "+str(nearest[1])+" "+str(nearest[2])
  print(msg)
  conn.send(msg)
  isReceived = conn.recv(1024)
  print(isReceived)

```

Figure 19: Caption19

#### 4.3. ReceivingTest.Client.py

- When the ReceivingTest.Client.py runs, it sends the user's location to Raspberry pi database. The user's pin number and the user's location are sent to Raspberry pi database.

```

import SocketToReceiverTest_Client as sock
import time
import MySQLdb
import subprocess

subprocess.Popen(["python", "Clock_project4.py"])

db = MySQLdb.connect('localhost','root','1234567890','TestDB')

cursor = db.cursor()
sql = ""

try:
  while True:
    time.sleep(5)
    rowcount = str(sock.sendingMsg()).split(" ")
    print("sended")
    rowcount = int(rowcount[1])

    for i in range(0,rowcount):
      receiving = str(sock.sendingMsg())
      receiving = receiving[2:-1]
      print(receiving)

      li = receiving.split(" ")
      sql = "UPDATE TestTable SET Location = '%s' WHERE Pin = '%s';" % (li[1],li[0])
      print(sql)
      cursor.execute(sql)

      db.commit()

except:
  db.close()

```

Figure 20: Caption20

#### 4.4. SocketToReceiverTest.Client.py

- Server sends a user's pin number and user's location through a socket.

```

import socket
import time
#from apscheduler.schedulers.background import BackgroundScheduler

HOST = "52.78.114.210"
PORT = 8089
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((HOST,PORT))

#sched = BackgroundScheduler()

def sendingMsg():
    request = bytes('Request','utf-8')
    s.send(request)
    data = s.recv(1024)
    return data

def disconnectingSoc():
    s.close()
#while True:
#    time.sleep(5)
#    sendingMsg()

#sched.add_job(sendingMsg, 'interval', seconds=5)
#sched.start()

```

Figure 21: Caption21

5. Where it is taken for?

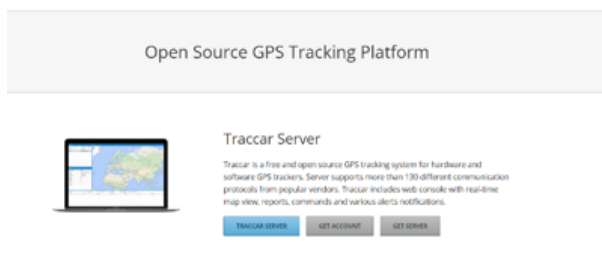


Figure 22: Caption22

6. How/Why you used it?

- The server obtains the user's location from the app. Bring the user's location in the database and process the data easily to send to the raspberry pi 3. Then pass the data to the clock by passing the user onto the raspberry pi 3.

### 0.5.3.3. Software Component 3: Raspberry pi

1. Purpose

- Raspberry pi 3 is essential for rotating our clock. The raspberry pi 3 gives electrical signals to the stepping motor. And the stepping motor which receives electrical signals turns to the direction that describes the location of the user.

2. Functionality

- The raspberry pi 3 calls the user's location from the server. The user's location is stored in the raspberry pi 'TestDB' database. The raspberry pi 3 steer the stepping motor according to the direction of the specified direction. Raspberry pi 3 uses the 'RPi.GPIO' modules. It is useful for sending signal to the stepping motor. 'RPi.GPIO' module is used in Python.

3. Location of Source Code

- /Clock.project4.py

4. Class Components

4-1. Clock.proejct4.py -1

- This Source code is a part that accesses the 'TestDB' database and receives the user's location.

- 'sendSignal()' method is sending electrical signal to the GPIO in Raspberry pi 3.

- 'numberTest()' method is identifying who the user is.

- 'test1()' method is identifying where the user is located



```

import RPi.GPIO as GPIO
import time
import MySQLdb

# Open database Connection
db=MySQLdb.connect("localhost","root","1234567890","TestDB")
# prepare a cursor object using cursor() method
cursor=db.cursor()
# execute SQL query using execute() method
cursor.execute("select * from TestTable WHERE Pin=1 or Pin=2")

try:
    # Fetch a single row using fetchone method
    results = cursor.fetchone()
    # Tuple
    Pin_1 = results[0][0]
    Location_1 = results[0][1]
    Past_Location_1= results[0][2]
    Pin_2 = results[1][0]
    Location_2 = results[1][1]
    Past_Location_2=results[1][2]
except:
    print ("Error: unable to fetch data")
finally:
    db.close()

# Clock signal
def sendSignal(k):
    for k in range(40):
        GPIO.output(pins[k], signal[step][k])

# Motor number
def numbertest(Pin):
    if Pin == 12:
        return 1
    elif Pin == 2:
        return 2

# Location Number
def test1(Location):
    if Location == "ITBT":
        return 1
    elif Location == "Playground":

```

Figure 23: Caption23

#### 4-2. Clock.proejct4.py -2

-‘test2()’ method is identifying where the user is located - ‘Angletest1()’ method compares the user’s past location with the current position and specifies the angle at which the clock is rotated.

```

# Location Number
def test1(Now_Locate):
    if Now_Locate == 1:
        return 'ITBT'
    elif Now_Locate == 2:
        return 'Playground'
    elif Now_Locate == 3:
        return 'Home'
    elif Now_Locate== 4:
        return 'Library'
    else:
        return 0

# Location compare and Angle
def Angletest1(Past_Locate, Now_Locate):
    if Past_Locate < Now_Locate:
        Answer= Now_Locate - Past_Locate
        if Answer ==1:
            return 128
        elif Answer ==2:
            return 256
        elif Answer ==3:
            return 384
    elif Past_Locate > Now_Locate:
        Answer= Past_Locate - Now_Locate
        if Answer== 1:
            return 384
        elif Answer==2:
            return 256
        elif Answer ==3:
            return 128
    elif Past_Locate == Now_Locate:
        return 0

```

Figure 24: Caption24

#### 4-3. Clock.proejct4.py -3

- ‘main()’ method is actually part of the action. It obtains the location of the user from the database. It processes the signals to be sent to the motor thorough the process of testing the user and testing the user’s location.

```

# Main Function
while True:
    # Open database Connection
    db=MySQLdb.connect("localhost","root","1234567890","TestDB")
    # prepare a cursor object using cursor() method
    cursor=db.cursor()
    # execute SQL query using execute() method
    cursor.execute("select * from TestTable WHERE Pin=1 or Pin=2")

    try:
        # Fetch a single row using fetchone method
        results = cursor.fetchone()
        # Tuple
        Pin_1 = results[0][0]
        Location_1 = results[0][1]
        Past_Location_1= results[0][2]
        Pin_2 = results[1][0]
        Location_2 = results[1][1]
        Past_Location_2=results[1][2]
    except:
        print ("Error: unable to fetch data")
    finally:
        db.close()

    Pin_1 = results[0][0]
    Location_1 = results[0][1]
    Past_Location_1= results[0][2]
    Pin_2 = results[1][0]
    Location_2 = results[1][1]
    Past_Location_2=results[1][2]

    Pin number 1= numbertest(Pin 1)
    Pin number 2= numbertest(Pin 2)
    Now Locate 1= test1(Location 1)
    Now Locate 2= test1(Location 2)
    Past Locate 1= test1(Past_Location 1)
    Past Locate 2= test1(Past_Location 2)

    Answer_range_1= Angletest1(Past_Locate_1, Now_Locate_1)
    Answer_range_2= Angletest1(Past_Locate_1, Now_Locate_1)
    time.sleep(10)

```

Figure 25: Caption25

#### 4-4. Clock.proejct4.py -4

- If the user current location and past location differ, this part moves. And if the pin number is one, this part moves. Raspberry pi 3 sends electrical signals to pin 12, 14, 20, 21 At this point, it is moved by a predetermined angle.

```

#Pin number 1
if Now Locate_1 != Past Locate_1:
    GPIO.setmode(GPIO.BCM)
    pins = [12,16,20,21]
    for p in pins:
        GPIO.setup(p, GPIO.OUT)
        GPIO.output(p, GPIO.LOW)
    FULL_STEP = 4
    HALF_STEP = 8
    signal_full = [
        [GPIO.HIGH, GPIO.LOW, GPIO.LOW, GPIO.LOW],
        [GPIO.LOW, GPIO.HIGH, GPIO.LOW, GPIO.LOW],
        [GPIO.LOW, GPIO.LOW, GPIO.HIGH, GPIO.LOW],
        [GPIO.LOW, GPIO.LOW, GPIO.LOW, GPIO.HIGH],
    ]
    signal_half = [
        [GPIO.HIGH, GPIO.LOW, GPIO.LOW, GPIO.HIGH],
        [GPIO.HIGH, GPIO.LOW, GPIO.LOW, GPIO.LOW],
        [GPIO.HIGH, GPIO.HIGH, GPIO.LOW, GPIO.LOW],
        [GPIO.LOW, GPIO.HIGH, GPIO.LOW, GPIO.LOW],
        [GPIO.LOW, GPIO.HIGH, GPIO.HIGH, GPIO.LOW],
        [GPIO.LOW, GPIO.LOW, GPIO.HIGH, GPIO.LOW],
        [GPIO.LOW, GPIO.LOW, GPIO.HIGH, GPIO.HIGH],
        [GPIO.LOW, GPIO.LOW, GPIO.LOW, GPIO.HIGH],
    ]
    steps = FULL_STEP
    signal = signal_full
    clockwise = True
    try:
        for i in range(0,Answer_range_1):
            if clockwise :
                for step in range(steps):
                    sendSignal()
                    time.sleep(0.01)
            else :
                for step in reversed(range(steps)):
                    sendSignal()
                    time.sleep(0.01)
    except KeyboardInterrupt:
        print("Interrupted!")
    finally:
        GPIO.cleanup()
        Real Past Locate_1 = test2(Now Locate_1)
        db=MySqlDB.connect('localhost','root','1234567890','TestDB')

```

Figure 26: Caption26

#### 4-5. Clock.proejct4.py -5

- If the user current location and past location differ, this part moves. And if the pin number is two, this part moves. Raspberry pi 3 sends electrical signals to pin 6, 13, 19, 26 At this point, it is moved by a predetermined angle.
- Finally, after turning the motor, save the current user's location in the 'Past.Location' column.

```

# Pin number 2
if Now Locate_2 != Past Locate_2:
    GPIO.setmode(GPIO.BCM)
    pins = [6,13,19,26]
    for p in pins:
        GPIO.setup(p, GPIO.OUT)
        GPIO.output(p, GPIO.LOW)
    FULL_STEP = 4
    HALF_STEP = 8
    signal_full = [
        [GPIO.HIGH, GPIO.LOW, GPIO.LOW, GPIO.LOW],
        [GPIO.LOW, GPIO.HIGH, GPIO.LOW, GPIO.LOW],
        [GPIO.LOW, GPIO.LOW, GPIO.HIGH, GPIO.LOW],
        [GPIO.LOW, GPIO.LOW, GPIO.LOW, GPIO.HIGH],
    ]
    signal_half = [
        [GPIO.HIGH, GPIO.LOW, GPIO.LOW, GPIO.HIGH],
        [GPIO.HIGH, GPIO.LOW, GPIO.LOW, GPIO.LOW],
        [GPIO.HIGH, GPIO.HIGH, GPIO.LOW, GPIO.LOW],
        [GPIO.LOW, GPIO.HIGH, GPIO.LOW, GPIO.LOW],
        [GPIO.LOW, GPIO.HIGH, GPIO.HIGH, GPIO.LOW],
        [GPIO.LOW, GPIO.LOW, GPIO.HIGH, GPIO.LOW],
        [GPIO.LOW, GPIO.LOW, GPIO.HIGH, GPIO.HIGH],
        [GPIO.LOW, GPIO.LOW, GPIO.LOW, GPIO.HIGH],
    ]
    steps = FULL_STEP

```

Figure 27: Caption27

#### 5. Where it is taken for?

- Blog Address: <https://codefooo.gitbooks.io/raspberry-experiments/content>.
- We refer to this blog to turn around the stepping motor.

```

import RPi.GPIO as GPIO
import time

def sendSignal() :
    for k in range(4):
        GPIO.output(pins[k], signal[step][k])
        print "GPIO X1" %(pins[k])

GPIO.setmode(GPIO.BCM)
pins = [12,16,20,21] #IN1, IN2, IN3, IN4
for p in pins:
    GPIO.setup(p, GPIO.OUT)
    GPIO.output(p, GPIO.LOW)

FULL_STEP = 4
HALF_STEP = 8

signal_full = [
    [GPIO.HIGH, GPIO.LOW, GPIO.LOW, GPIO.LOW],
    [GPIO.LOW, GPIO.HIGH, GPIO.LOW, GPIO.LOW],
    [GPIO.LOW, GPIO.LOW, GPIO.HIGH, GPIO.LOW],
    [GPIO.LOW, GPIO.LOW, GPIO.LOW, GPIO.HIGH]
]

signal_half = [
    [GPIO.HIGH, GPIO.LOW, GPIO.LOW, GPIO.HIGH],
    [GPIO.HIGH, GPIO.LOW, GPIO.LOW, GPIO.LOW],
    [GPIO.HIGH, GPIO.HIGH, GPIO.LOW, GPIO.LOW],
    [GPIO.LOW, GPIO.HIGH, GPIO.LOW, GPIO.LOW],
    [GPIO.LOW, GPIO.HIGH, GPIO.HIGH, GPIO.LOW],
    [GPIO.LOW, GPIO.LOW, GPIO.HIGH, GPIO.LOW],
    [GPIO.LOW, GPIO.LOW, GPIO.HIGH, GPIO.HIGH],
    [GPIO.LOW, GPIO.LOW, GPIO.LOW, GPIO.HIGH]
]

#stepping mode and direction
steps = FULL_STEP
signal = signal_full
clockwise = True

try:
    # 1 cycle = 4 step for FULL
    # 1 cycle = 8 step for HALF
    # 1 rev = 512 cycle
    for i in range(512):
        if clockwise :
            for step in range(steps):
                sendSignal()
                time.sleep(0.01)
        else :
            for step in reversed(range(steps)):
                sendSignal()
                time.sleep(0.01)

```

Figure 28: Caption28

#### 6. How/Why you used it?

- Raspberry pi 3 is currently used in many places. Raspberry pi 3 is a small computer that you use to turn a stepping motor. You can create a program by using a python and sending electrical signals to the GPIO. We used RPI.GPIO modules for sending signals

to the GPIO. We can rotate the two stepping motors depending on the location of the user.

## 0.6 Use Cases

### 0.6.1 Main Use Cases

#### 0.6.1.1. Download Our Application

- User downloads our Whereabouts-SE Project APP in google store.

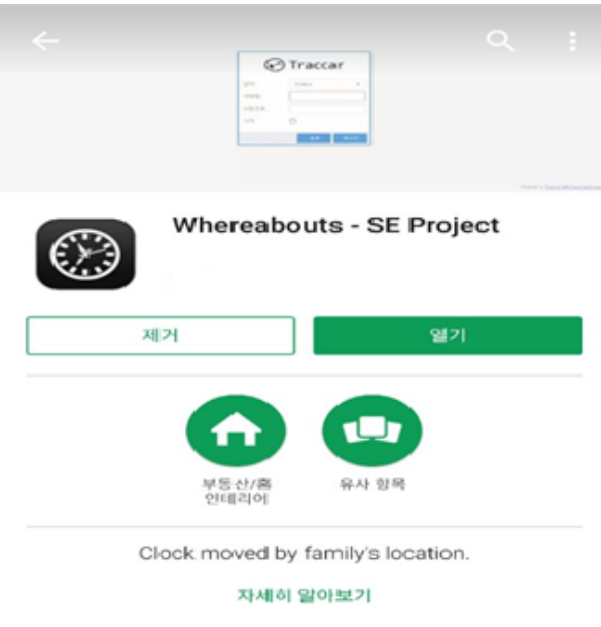


Figure 29: Caption29

#### 0.6.1.2. Permit to use GPS

- Second, if User installs Whereabouts – SE Project Application, User permits to the use GPS in the App. This function allows App to use User’s GPS data.

#### 0.6.1.3. Put the Amazon EC2 Server Address

- User puts the Amazon EC2 Server Address in the Whereabouts – SE Project App. So, User’s Location is sent to Server.

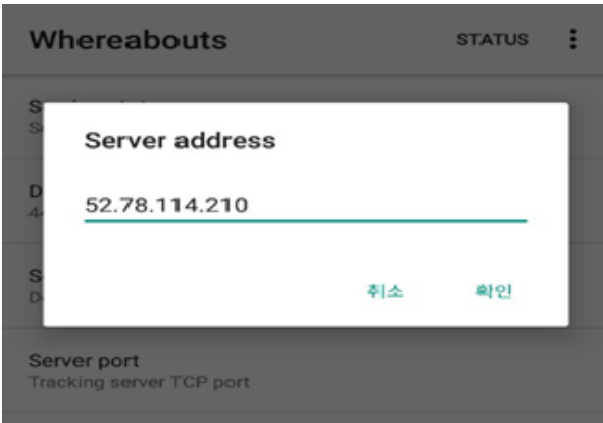


Figure 30: Caption30

#### 0.6.1.4. Set the Frequency

- User set the Frequency that request to the Amazon EC2 Server.

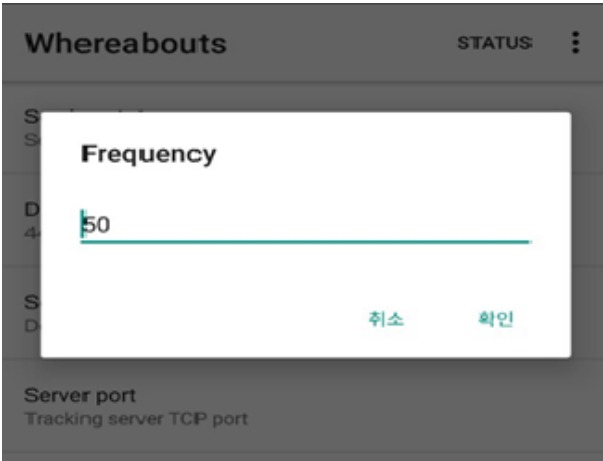


Figure 31: Caption31

#### 0.6.1.5. Select location provider between GPS mode and Network mode

- User selects location provider between GPS mode and Network mode.

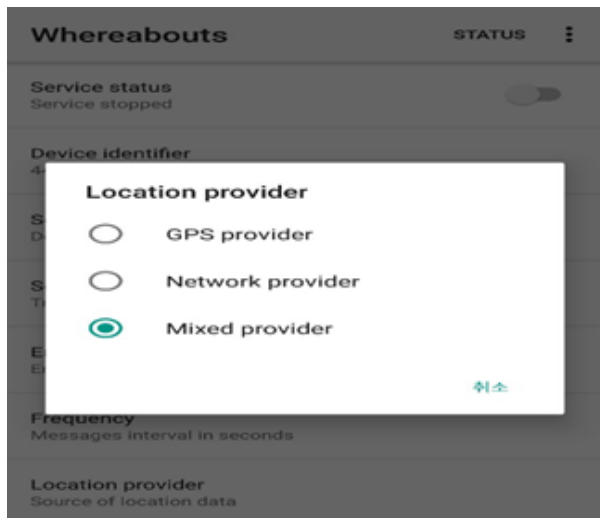


Figure 32: Caption32

#### 0.6.1.6. Service Start

- After User finished app setting, User can use the Clock Service. User's location is sent to Server. And Server knows User's real location. So, Server sends a user's location to Raspberry pi.

#### 0.6.1.7. Connect the Clock power

- User should connect the Raspberry pi power in the socket. So, Clock can be rotated thorough User's real location.