Energy Consumption Analysis on an Alternative GPS Information Transmission Path for Android Devices

Zhuoheng Li, Tianqin Cai

Project Report (Final)

***Abstract* – Location information is commonly used in Android navigating applications. The most accurate method of obtaining the position for Android mobile devices is using GPS information from an internal receiver. However, there exist a potential problem, after switching on the GPS function and starting the navigating function via some application, the device would start to heat and consume remaining battery power in a significant rate.**

Bluetooth has a wide range of usage including transmitting control information and positioning data. As a result, transmitting GPS information from an external device via Bluetooth is an accessible path of saving energy.

In this project, we introduce BlueGate, a tool to build connections between android devices, and transmitting GPS information. The GPS information is obtained in device A and this information can be utilized by another Android device B. The customized Android framework on device B can receive and utilize the location information globally.

We evaluate the energy consumption using battery-historian. Several groups of energy test under different scenarios are implemented. The results indicate that the new path could save up to 40% of the energy.

I. Introduction and Background

This document proposes the possible functions of an Android Open Source Project (AOSP) implemented alternative GPS information transmission path for Android Devices.

The main idea of implementing the alternative path is transmitting GPS information via Bluetooth connection between Android devices. Through this method, power consumption by a GPS model on an Android device could be burdened by another device.

There are some ideas on applying the external GPS on a single software, BT GPS [1], for example, it can resolve GPS information from an external GPS module via Bluetooth, and apply the location to global level throughout a mock location under Android developer options. However, this path might not be useful in many cases, for example, some applications has anti mock location function. To validate the proposed path, building an alternative on the basic Android system framework is necessary. Android is an open source software stack for a wide range of mobile devices and a corresponding open source project led by Google. This project enabled individual developers and device manufactures to build their own version of Android operating system. And every level of the system can be customized. In this project, our own version of Android framework is established.

There are many papers related to analysis on GPS and Bluetooth power consumption analysis [2][3][5]. However, very few of them mentioned the ratio of energy saving for Bluetooth based location information path. So, a more specific comparison between them will be included in this project.

In this project, the following objects will be achieved

1. To build the project environment, for software development/testing, system level modification, energy consumption evaluations.
2. To build a software to connect two Android devices together and being able to transmit location packets.
3. To modify the existing Android system which enables the received location packet as a system level information that can be utilized by other applications. Also, to save energy, the GPS module should be disabled while the application in step 2 is running.
4. To evaluate the energy consumption in many aspects to observe how much energy is saved in the new path.

The rest of this proposal is organized as follows: Section 2 describes the related work of the proposed project, Section 3 described the technical details of implementing the application and the new Android framework, Section 4 provides the results and evaluation. Section 5 briefly described the workload arrangement and in Section 6 we draw a conclusion, including challenges and limitations of this project. Finally, in Section 7, some possible future work is summarized.

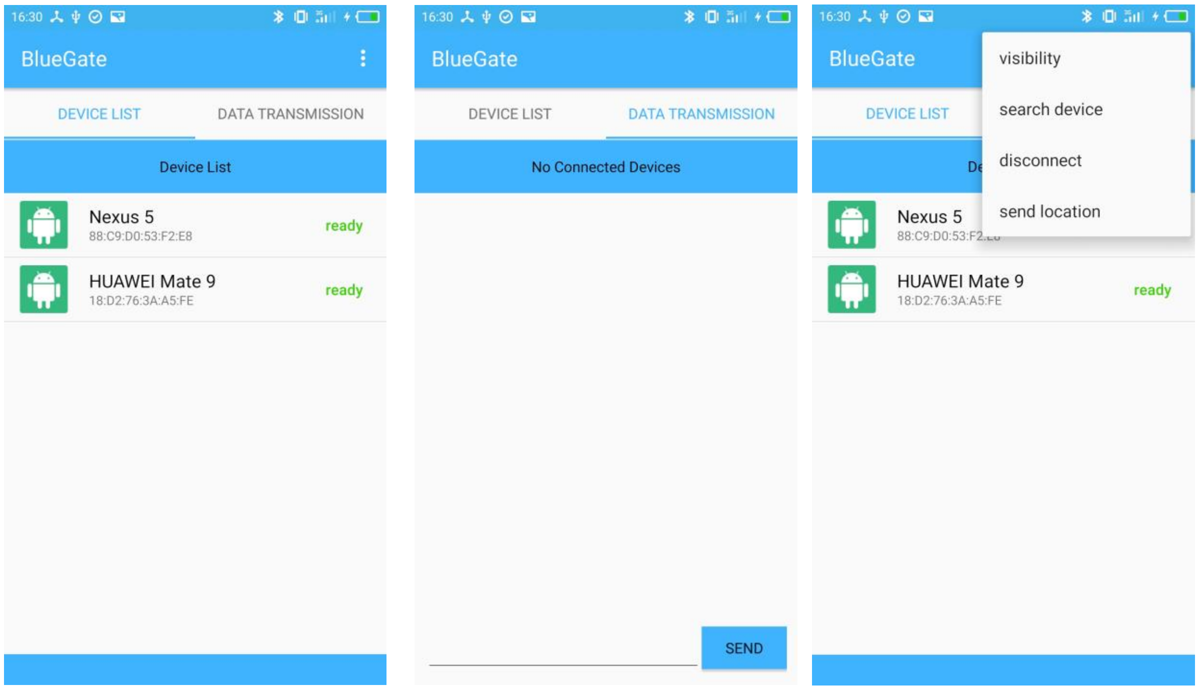


Figure 1 BlueGate Interface

II. Related Work

Current related work connecting two devices via Bluetooth requires some certain conditions. One example is that BTGPS requires an external Bluetooth GPS device. The external device sends a series of standard NEMA packages throughout the Bluetooth port. And if an external tool is not used, then an alternate way requires another software. Our proposed Bluetooth tool connect two Android devices together, with only one type of application installed on both devices. Both devices can act as a host or client of the Bluetooth connection.

One of the popular Android app on Google Play store is Bluetooth GPS Receiver. [1] It is an app to use one of users' Android phone/ external device as location source, and another Android phone as the corresponding receiver. The Bluetooth GPS Receiver needs to obtain the permission to use mock location service, then use this service to provide the received GPS information to other Android devices.

Another app that applies this technique is Tether-GPS Lite published in 2013, it can also available in Google Play store now. [6] Tether-GPS Lite allows users to share the GPS on one Wi-Fi enabled Android device with another Wi-Fi enabled Android Device. It can be launched by using the included 1x1 widgets, through intent broadcasts from other apps, or by using the launch buttons in the Tether-GPS configuration app. The example use of Tether-GPS is as follows: The users can connect their Nook Color and their HTC Evo to the same network, or tether the Nook Color to the HTC Evo with Android-WiFi-Tether. Then run TGPS Server on the HTC Evo and TGPS Client on the Nook color. The Nook Color will receive GPS data from the HTC Evo. Applications on the Nook Color such as Google Maps and Google Navigation will then work using the shared GPS data. Tether-GPS Lite includes the full client and a lite version of the server which will run for 5 minutes at a time. After the time limit is hit the server can simply be restarted via the widget.

After receiving and resolving the GPS signal from external devices, many all of these apps requires using a mock location permission under the developer mode, otherwise the location information cannot be used at system level. Our BlueGate tool is designed to be a system specific application which can interact with the framework level of a designed Android system.

There are also papers relating to the energy analysis on power consumption of many types of related applications: Rahul Bolani from UCLA did the research about average energy consumption of Bluetooth, Wi-Fi and cellular networks for transmitting data produced f bytes per second with different packet type and sniff interval. [1] The result shows that Bluetooth transfer saves much more energy than the latter two. Lei Zhang and Yong Guan performed Sens-Track, which is a location tracking service that leverages the sensor hints on the smartphone to reduce the usage of GPS. The Evaluation on traces from real users demonstrates that Sens-Track can significantly reduce the usage of GPS and still achieve a high tracking accuracy. [3] Nicholas Capurso presented another research by monitoring battery life over time and showing that an indoor localization method triggered by indoor or outdoor context can improve smartphone battery life and, potentially location accuracy. [2] Unlike these previous work, our work is aiming at finding a new path of transmitting location information and analyzing the utilization and energy consumption.

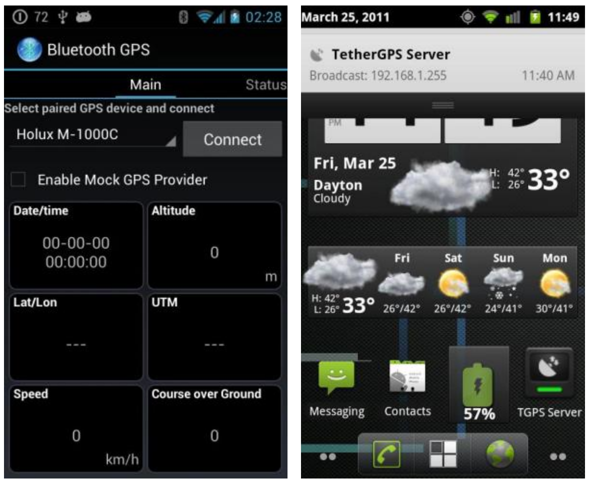


Figure 2 Bluetooth GPS Receiver and Tether-GPS Lite

III. Technical Approach

The main idea of implementing the alternative path is transmitting GPS information via Bluetooth function of two Android devices. Figure 3 shows the structure of the project. There are two phones marked as Phone 1 (Sender) and Phone 2 (Receiver), where we both installed the BlueGate application. The location information will be transferred through Bluetooth from Phone 1 to Phone 2.
After receiving the GPS information, BlueGate will send broadcast to the system. And in the system, we added a new broadcast receiver, which on receiving the broadcast from BlueGate will block the native library and send the transferred GPS information to *LocationManager*.
The process mainly contains seven parts, including Application design, Bluetooth Connection, obtaining GPS information, pack location information into packets, receive and resolve packets, utilize GPS information and battery historian test evaluation. The detailed methodology is presented below.

3.1 BlueGate Application Design

Our first step is designing a user-friendly application. Android Studio is selected as our IDE, with which we created the application shown in Figure 1. It contains two layouts. The first page contains a device list to display the devices that can be connected through Bluetooth. After the device is selected, it will lead the users to the second page, where they can choose how to use the GPS information. Users can use the navigation applications on their phone and provide the received location information to those apps searching for nearby resources, or the they can obtain the GPS information presented by String displayed on BlueGate receiver end.

3.2 Bluetooth Connection

The Bluetooth testing interface is used to testify the utility and capability of sending any kind of information using existing Bluetooth API.

At this stage, a basic framework which uses system Bluetooth APIs has been built. The testing app has the function to get the permission of using Bluetooth on the cellphone, to let it be detectable to other cellphones in 300 seconds, and search for all the nearby devices for connecting. The nearby devices will also be displayed as a list, and can be connected by clicking on the list item. By doing this step, we ensured that the Bluetooth connection is working well.

We conducted our experiment on the Nexus 5 Android

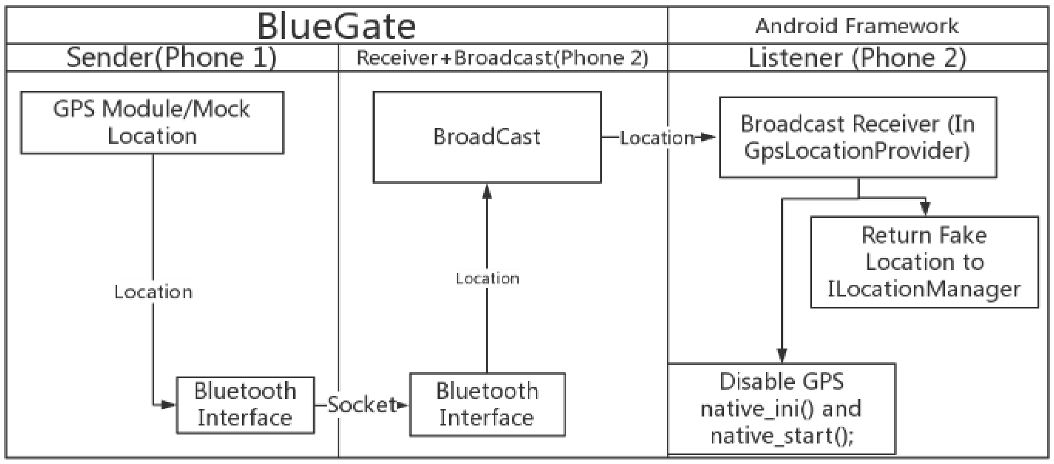


Figure 3. Project Structure

Phone with Android 6.0.1. And the testing application working on the Nexus phone is shown in Figure 4. We applied Bluetooth-Adapter APIs to send out intent to trigger corresponding behaviors, and used Broadcast-Receiver class to capture the Scan mode of the Nexus phone.

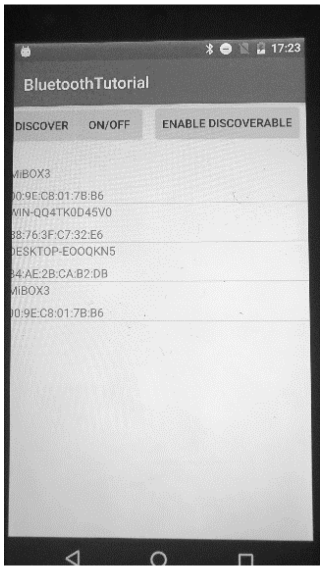


Figure 4 Bluetooth Connection test app on Nexus Phone

3.3 Obtaining GPS information

Another problem to solve before we starte to build BlueGate is to obtain the GPS information from the cellphone.

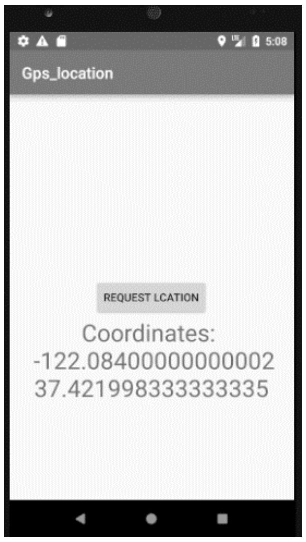
For this task, we developed another function, which is shown in Figure 4. The key part in this step is to set up *LocationManager* and *LocationListener* class in the source code. The first one will set up the system services for obtaining GPS information, while the second test application will provide *onStatusChanged* and *onLocationChanged* method, which helps us update the GPS information when necessary. To obtaining the latest location on the sender end, we set the minTime and minDistance term to 0 to maximize the accuracy of the location.

One important issue worth to mention is that, after Android 6.0.0, Google changed the location authentication rules for apps using both location permissions. Thus, adding the dynamic permission request term is necessary.

After completing these two steps, we combined them to test GPS information transfer through Bluetooth. In this third test, we successfully transferred the GPS information in text format through Bluetooth to the receiver end. The result is shown in Figure 5. After requesting location on the receiver end, we successfully received the GPS information sending from the source end through Bluetooth.

3.4 Pack location information into packets

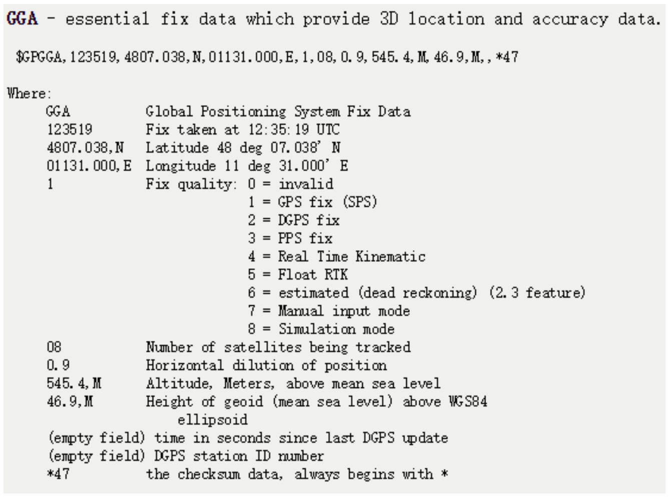
One key problem we need to solve is how to stream the location information from the location manager to the Bluetooth manager.



```
Coordinates:
-122.08400000000002
37.421998333333335
```

Figure 5 GPS information query test app

Many of the critical information from the GPS module is to be transmitted, including height, longitude, etc. All the information generated by GPS module is defined in NMEA-0183 protocol.



```
GGA - essential fix data which provide 3D location and accuracy data.

 $GPGGA,123519,4807.038,N,01131.000,E,1,08,0.9,545.4,M,46.9,M,,*47

Where:
     GGA          Global Positioning System Fix Data
     123519       Fix taken at 12:35:19 UTC
     4807.038,N   Latitude 48 deg 07.038' N
     01131.000,E  Longitude 11 deg 31.000' E
     1            Fix quality: 0 = invalid
                               1 = GPS fix (SPS)
                               2 = DGPS fix
                               3 = PPS fix
                               4 = Real Time Kinematic
                               5 = Float RTK
                               6 = estimated (dead reckoning) (2.3 feature)
                               7 = Manual input mode
                               8 = Simulation mode
     08           Number of satellites being tracked
     0.9          Horizontal dilution of position
     545.4,M      Altitude, Meters, above mean sea level
     46.9,M       Height of geoid (mean sea level) above WGS84
                      ellipsoid
     (empty field) time in seconds since last DGPS update
     (empty field) DGPS station ID number
     *47          the checksum data, always begins with *
```

Figure 6 GGA information sample

As shown in Figure 6, typical GGA information is a simple String contains various of information including all that is essential. So, we simply form the information obtained from Location Manager in to a String format. The location information in Location object can be extracted by functions such as *getlongitude()*.

Alternatively, transfer the Location object to String directly is possible. However, in some cases, the format of the Location information might change, which add difficulty to the next resolving step.

Another path of packing the *Location* Object is using serializable method provided by Android development toolkit. The receiver end can resolve the *Location* directly with proper settings.

3.5 Receive and Utilize Packets

After resolving packet from step 3.4 accordingly, with String separation or reversing the serialize process, Location information from the sender can be displayed on the screen. And to utilize the location information, a Location buffer is set to handle all the information from the received packets. Information such as latitude and longitude are stored in Location object in double format, and can be modified by using functions such as *setlatitude (double latitude).* Additionally, the *isFromMock* boolean flag is included in the *Location* object as well. Such location information in *Location* format can be transmitted to *LocationManagerService* directly in the following steps. Similarly, resolving the string using key word identification. Since the format of the data is fixed, then the *Location* in String can be separated and reformed as a new *Location* object.

3.6 Utilize the location information

To implement the utilization of received location, we managed to change the path of the original location provider.

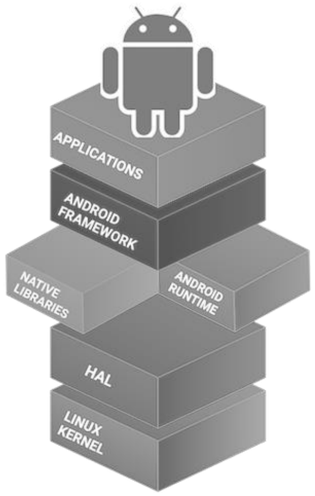


Figure 7 Android Framework

The original path of obtaining location information from GPS module is 1. *getLastKnownLocation()* for the GPS module cannot be initialized instantly. 2. Give *Location Request* to *LocationManager*. 3. The *LocationManager* then call some functions in *GPSLocationProvider* to call some native functions in the native libraries. Then the native functions will start GPS module through the HAL and Linux Kerner level (we don’t care in this case, for we focus the changes on framework level). 4. After receiving location information from satellite, some callback functions in framework level will be triggered, such as *reportlocation()*. 5. The *LocationManagerService* will determine whether this information should be broadcasted to the system. 6. The Location will be returned to *OnLocationChanged()* function in the applications.

Our thought is not establishing a new transmission path of GPS location information, but 1. Stop the initialization of GPS module by interrupting the native calls. 2. Return *Location* object properly back to *LocationManger*.

3.6.1 Broadcasting

In the receiver end of our BlueGate application. The app will constantly listen to the transmitted information. As long as the *Location* is received, the key information can be packed in to a *Broadcast intent* with *extras.* The sending rate is 1 packet per second, thus such broadcast will be provided at same frequency.

3.6.2 GPS on/off switch

Next, the original native calls should be disabled. For one of our main goal is saving energy of the receiver. There are several native calls in the *GPSLocationProvider* class. Once the constructor of *GPSLocationProvider* is called, it will automatically initialize the GPS module and wait for location information to be returned. So we added an if else before those critical native calls, for example *native_initialize()* and *natiave_navigation_start()*. We can either disable those functions by listening to the application broad cast in 3.6.1 or by adding a system level switch in the system Settings app.

3.6.3 Report the location if requested

Since we have disabled the normal initialization of GPS module in part 3.6.2, the GPS module will not report the location information as usual. So, alternatively, we build our own location returning function.

The location is stored in a buffer of *GPSLocation Provider*. Once the broadcast is received by the listener added in *GPSLocationProvider*, then the location stored in the buffer will be send to *LocationService* via *reportLocation (Location location)* associate with *ILocationmanager_aidl*. This step is an imitation of the original location transmission path.

3.6.5 Mock flag removal

There exists a mock flag adding step in the *Location* generation process if the *Location* comes from the mock location source. Some of the *Location* from our sources may generated from a mock location source, thus if the BlueGate application or the system switch is enabled, then the mock flag of the *Location* will be removed.

3.7 Battery Historian test

To test the energy saving of this alternative path, a battery usage monitoring tool is required. Currently, our project group doesn’t have any available tools to monitor the battery usage of the GPS module directly, thus we monitor the overall battery level change against time.

Battery Historian is a popular tool to inspect battery related information and events on an Android device running Android 5.0 Lollipop and later. It allows application developers to visualize system and application level events on a timeline with panning and zooming functionality, so that users can easily see various aggregated statistics since the device was last fully charged, and select an application to inspect the metrics that impact battery specific to the chosen application.

The bug report generated by the Android system is the source of Battery Historian report. And the Battery Historian online [10] tool can analysis this bug report without installing the Battery Historian tool locally.

IV. Results and Evaluation

There are two key objectives of this project. 1. To explore whether this transmission path is doable. 2. To explore the potential of energy saving when using an external location source.

For a typical navigation application, the energy consuming consists of three parts: screen, GPS module, and CPU.

BlueGate disables GPS module, it saves energy up to 40% with screen on.

4.1 Effectiveness of the project

In our project, 2 path of transmitting Location object and 2 system-level GPS module switch is verified as effective.
We verified the location from both GPS module and a mock location application in the sender side. And the google map installed on the receiver can receive this information and utilize them in the navigation.

4.2. Evaluation of energy consumptions

For the first 3 group of testing, we managed to estimate the energy saving utilization of using the BlueGate application in a standardized condition, the following tests are implemented and the results are showed in table 1.

4.2.1. Test the basic energy consumption of one Android device when transmitting Bluetooth location information/receiving GPS information from GPS module, etc. To give a standardized scale to the following evaluation.

4.2.2. Test and compare the energy consumption of a customized location information, basically a series of continuous dynamic location information. The energy consuming history will be logged and compared to the results from GPS model.
4.2.3. Energy consumption test on the proposed transmission path uses real navigation tools. Tester uses the navigation function of the testing applications and walk through same route. And then the BlueGate location providing is tested. Energy usage history is recorded.

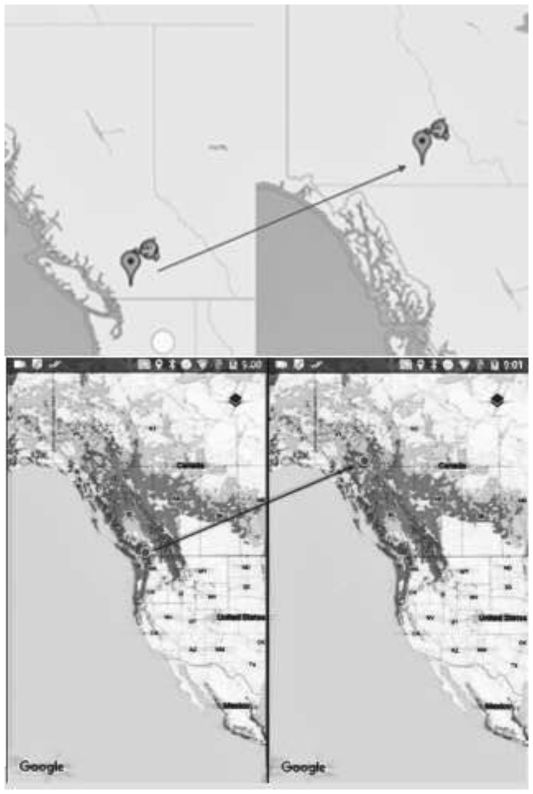


Figure 8 BlueGate Operation Result

Number of times	Planned Testing Time	Method	AVG. Battery Usage (Start From 100%)
4	30 min	Screen On, BlueGate App only	5.75%
4	30 min	Screen On, GPS test App only	17.00 %
4	30 min	Screen On, GPS test APP with BlueGate (GPS Disabled, Source: Mock Location)	10.00 %
2	15 min	Google Map Outdoor Test, Walk Guide Mode, without BlueGate	7.50%
2	15 min	Google Map Outdoor Test, Walk Guide Mode, with BlueGate	6.50%

Table 1 Battery Historian test result

4.3 Packet sending frequency optimization

One problem worth consideration is that, the frequency of packet sending is not determined yet, which is related to energy consumption. The result of frequency test indicates that the frequency change on packet sending/broadcasting has insignificant influence on battery consuming.

Number of times	Planned Testing Time	Method	AVG. Battery Usage (Start From 100%)
4	30 min	Screen On, BlueGate App only (1 packet/ second)	5.75%
4	30 min	Screen On, BlueGate App only (2 packets/ second)	6.75%

Table 2 Battery consumption result

V. Workload Arrangement

4.5 Division of workload

Zhuoheng: Relevant Document Analysis, Stabilize the system level switch, Changing the packet sending frequency, UI-Optimization, Implementation and Evaluation part in Final Project Report

Tianqin Cai: Utilize the location information, Relevant Document Analysis, UI-Optimization, Software testing, Energy Consumption Testing, The Background, Related work, and Evaluation part in the Final Project Report

VI. Conclusion & Limitations

6.1 Conclusion of the project

The object of developing an Android-based system level APP which can connect two Android devices via Bluetooth and enable one of them to receive and utilize information from the other one has achieved. And an adjustable interface is developed later. Then, an energy analysis on this application is implemented to estimate the possible energy saving, the result shows that saving energy is saved up to 40%.

6.2 Key challenges in the project

6.2.1 Code review and framework building

Tianqin has no experience on Android programming before taking this course, and Zhuoheng has half a year's Android experience. Reading the system source code without guidance is hard for this level's Android programmers. Apart from the complexity of Android source code, the environment of compiling is hard for us establish. From Linux to fastboot the system image to the Nexus phone, every step takes time and energy to implement. Situation is not optimistic when the source code occupies more than 30GB of memory. The significant size of code leads to an inflexible direction changing. A single change on the source code usually takes 20 minutes until we could observe the change on the Nexus phone.

The code compiling process is time consuming, the shortest 20-min path is: update source list, lunch to choose version, mmm the specific module, make snod to save changes to existing system image, adb into fastboot mode, then overlay the original system image, finally, rebooting. Not to mention Jack server failure, 6-hour make file process failure, Syntax error which cannot be checked by IDE, etc. More than 30 types of problems in every level of the environment are solved until we could operate our own system.

6.2.2 Version control/data saving

There are hundreds of versions release of Android source code. The 6.0.1 for hammerhead is selected for this project. In source code reviewing step, some useful suggestions are version limited, 4.4 or even 2.1. As a result, there is barely useful resource for such specific problem.

Version control is important, for recovering the source code at this size requires time and patience. In some cases, making a backup on hard drive or on cloud is necessary for preventing any possible disaster like the portable disk disorder problem which we've met.

6.3 Limitations of this project

6.3.1 Stability of the modified framework

The framework we built changed the original path of transmitting the location information. There is the potential risk of low stability of the new framework. The existing path has been developed and optimized by many experts in this area, changing its operation trace without enough software testing is at high risk of instable.

6.3.2 Insufficient Energy test

The energy test we preformed is limited to the global battery usage and the module usage from every single application. Since we are not attempting to establish a completely new path of transmitting location which deprecating the original classes, so, the recording of GPS start/disable is remained in the log file.

Additionally, the energy test is not being able to eliminate other applications' interference or the system level interruption such as cell network connected/disconnected.

Our effort on minimizing the error is increasing the testing attempts and close all unnecessary application/services which is far from a solid prove.

6.3.3 Non-mainstream Modification on AOSP

The method we proposed in this report is not likely to be used in the Android systems running on other mobile phones, for many important functions of application on the markets requires a real position to perform correctly. Such modification is against the original thoughts of the system developers, also, the broadcasted location information might lead to privacy leakage, thus, the proposed path is not likely to be used by common users.

VII. Future work

1. Increase the net energy utilization rate

At this stage, the BlueGate application can connect one slave device only, if the number of connections allowed increased, then the net energy saving ratio can be increased. Except for that, the WIFI hotspot or even NFC connection can be possible ways to make connections.

2. More energy tests

At this stage, due to the limitation of experiment resource, the only phone we could perform energy test is a Nexus 5. If possible, more energy test with more precise criteria and more phones involved can be conducted in the future work.

In the packet sending frequency testing part, only 2 groups of tests are implemented, in the future, the extreme frequencies such as 100 packets per second or 1 packet per minute should be tested.

3. Feedback term

In this stage, we are not caring the energy consumption on the sender. Thus, a negative feedback term is not in the application design. In next version of BlueGate application, a stop receiving order can be designed to stop the sender consuming energy on obtaining location information from the GPS module.

APPENDIX

Working environment

Build Environment	Linux Ubuntu 16.04
Source Code Version	6.0.1 r32 for hammerhead
Testing Phone 1	LG Nexus5 16 GB
Testing Phone 2	MEIZU Note 3 16 GB
Battery Analysis Tool	Battery Historian V2

References

[1] BlueTooth GPS: http://www.gpscentral.ca/ bluetooth-gps-receivers.htm
[2] Balani, R. (2007). Energy consumption analysis for bluetooth, wifi and cellular networks. Online Httpnesl Ee Ucla Edufwdocumentsreports2007PowerAnalysis Pdf.
[3] Capurso, N., Song, T., Cheng, W., Yu, J., & Cheng, X. (2017). An android-based mechanism for energy efficient localization depending on indoor/outdoor context. IEEE Internet of Things Journal, 4(2), 299-307.
[4] Kim, K. R., Jeong, S. H., Kim, W. Y., Jeon, Y., Kim, K. S., & Hong, J. H. (2017, June). Design of small mobile robot remotely controlled by an android operating system via bluetooth and NFC communication. In Ubiquitous Robots and Ambient Intelligence (URAI), 2017 14th International Conference on (pp. 913-915). IEEE.
[5] Lin, K., Kansal, A., Lymberopoulos, D., & Zhao, F. (2010, June). Energy-accuracy trade-off for continuous mobile device location. In Proceedings of the 8th international conference on Mobile systems, applications, and services (pp. 285-298). ACM.
[6] TetherGPS Lite: https://apkpure.com/tether-gps-lite/com.comptonsoft.tgps_lite
[7] Noertjahyana, A., & Andjarwirawan, J. (2012). Development of Mobile Indoor Positioning System Application Using Android and Bluetooth Low Energy with Trilateration Method (Doctoral dissertation, Petra Christian University).
[8] Zhang, L., Liu, J., Jiang, H., & Guan, Y. (2013). SensTrack: Energy-efficient location tracking with smartphone sensors. IEEE Sensors Journal, 13(10), 3775-3784.
[9] Li, T-HS, Shih-Jie Chang, and Wei Tong. "Fuzzy target tracking control of autonomous mobile robots by using infrared sensors." IEEE Transactions on Fuzzy Systems 12.4 (2004): 491-501
[10] BatteryHistorian online: https://bathist.ef.lc/