

BLUETOOTH RANGE EXTENSION THROUGH SINGLE-HOP NETWORKING

A PROJECT REPORT

*Submitted in partial fulfillment of the
requirement for the award of the
Degree of*

BACHELOR OF TECHNOLOGY in ELECTRONICS AND COMMUNICATION ENGINEERING

By

**SAI ANAND N – 13BEC1123
VIGNESHWAR B – 13BEC1167**

Under the Guidance of

Prof. Nagajayanthi B



**SCHOOL OF ELECTRONICS ENGINEERING
VIT University
CHENNAI. (TN) 600127**

(MAY 2017)

BLUETOOTH RANGE EXTENSION THROUGH SINGLE-HOP NETWORKING

A PROJECT REPORT

*Submitted in partial fulfillment of the
requirement for the award of the
Degree of*

BACHELOR OF TECHNOLOGY in ELECTRONICS AND COMMUNICATION ENGINEERING

By

**SAI ANAND N – 13BEC1123
VIGNESHWAR B – 13BEC1167**

Under the Guidance of

Prof. Nagajayanthi B



**SCHOOL OF ELECTRONICS ENGINEERING
VIT University
CHENNAI. (TN) 600127**

(MAY 2017)

CERTIFICATE

This is to certify that the Project work titled “***Bluetooth range extension through single-hop networking***” that is being submitted by “***Sai Anand N***” and “***Vigneshwar B***” is in partial fulfillment of the requirements for the award of **Bachelor of Technology**, is a record of bonafide work done under my guidance. The contents of this Project work, in full or in parts, have neither been taken from any other source nor have been submitted to any other Institute or University for award of any degree or diploma and the same is certified.

Prof. Nagajayanthi B
Guide

The thesis is satisfactory / unsatisfactory

Internal Examiner

External Examiner

Program Chair

Dean (SENSE)

ACKNOWLEDGEMENT

We would like to express our deepest appreciation and gratitude to all those who provided us with this opportunity and the environment to complete our project and subsequently this thesis. We offer our special gratitude to our faculty guide, Prof. Nagajayanthi B, whose constant encouragement, suggestions and feedback enabled us to undertake multiple approaches towards our project objective and the subsequent success in achieving the same. We would also like to express our sincere thanks to the Dean of the School of Electronics Engineering, Dr. SRS Prabakaran, whose constant word of encouragement and extension of the school's facilities towards our work helped us immensely during the course of this Project. Furthermore, we would also like to acknowledge with much appreciation the crucial role of Dr. Manoj Kumar, our Program Chair, whose unstinting academic support helped us plan a proper timeline for the successful completion of this project. We would like to take this opportunity to also express deep gratitude towards our review panel members, for their continuous evaluation, feedback and suggestions which directed us to make constant improvements that led to the completion of the project within stipulated time. Last, but not the least, we would like to thank our parents for their unending financial and emotional support which served as a constant backbone in aiding our progression in the field of engineering and completing this project.

Sai Anand N
13BEC1123

Vigneshwar B
13BEC1167

TABLE OF CONTENTS

LIST OF TABLES	IV
LIST OF FIGURES	V
LIST OF ABBREVIATIONS	VI
Abstract	VII
1 Introduction	1
1.1 Objective	1
1.2 Background	1
2 Project Description	5
2.1 Methodology	5
2.2 Design Approach	9
2.3 Standards	11
2.4 Constraints, Alternatives and Tradeoffs	12
2.5 Technical Specification	14
3 Project Demonstration	15
4 Cost Analysis	25
5 Results and discussions	26
6 Conclusion	28
7 Appendix	29
8 References	35

LIST OF TABLES

Table No.	About	Page No.
1.	Literature Survey	3
2.	Cost Price Tabulation	25

LIST OF FIGURES

Figure No.	About	Page No.
1.	Project Modules	5,6
2.	System Block Diagram	9
3.	Project Flow Diagram	10
4.	Raspberry Pi 3B Schematic	14
5.	Hardware Module	15
6.	Incoming connection	16,17
7.	Directory Monitoring and Device Address Verification	18
8.	Forwarding the file to destination	18.19, 20
9.	RSSI-based Distance Estimation	21,22,23,24

LIST OF ABBREVIATIONS

S. No.	Abbreviations	Full Form
1.	PAN	Personal Area Network
2.	IEEE	Institute of Electrical and Electronics Engineers
3.	SoC	System on a Chip
4.	API	Application Programming Interface
5.	RSSI	Received Signal Strength Indicator
6.	PDA	Personal Digital Assistant
7.	IoT	Internet-of-Things
8.	LAN	Local Area Network
9.	RF	Radio Frequency
10.	R-Pi	Raspberry Pi
11.	IDE	Integrated Development Environment
12.	OBEX	Object Exchange
13.	HTTP	Hyper Text Transfer Protocol
14.	PC	Personal Computer

ABSTRACT

We presently live in an advanced gadget era where cross-platform communication is growing at a rapid pace often leading to the creation of ad hoc wireless personal area networks within households and workspaces where information can flow between connected devices. The most widely used wireless PAN standard is Bluetooth (IEEE 802.15.1) which combines a basic data-rate and a low-energy core configuration for an RF transceiver, baseband and protocol stack. Bluetooth, however, does carry a significant disadvantage going forward as it is accompanied by a severe trade-off in terms of the overall range of operation and the rate of transfer owing to its low-powered nature. In addition to this, the lack of Bluetooth range extension standards for personal networking and the expensive nature of commercial and/or industrial grade Bluetooth extenders/repeaters outlines the need for alternatives. We propose a prototype of a Bluetooth-enabled mediation device that can be used to receive from and send files to other Bluetooth-enabled devices that are out of one another's range of operation but are within the communication range of the proposed prototype. Thus using a Linux-based open source development platform (R-Pi SoC) with good API support, we built a Bluetooth server that can receive files from nearby devices, process them to verify the source and destination addresses and forward them to the intended device as it becomes available. The server has been intelligently designed to function automatically and log all incoming and outgoing transfer metadata. The server in addition also measures the RSSI value of the source and the destination devices and calculates the proximity of these devices to the server, whilst estimating the distance between them to calculate total distance of transmission. Utilising this Bluetooth server, we can build a single-hop network that can effectively double the typical range of communication of Bluetooth devices.

CHAPTER 1

INTRODUCTION

1.1 Objective

The primary objective for the project is to extend the range of Bluetooth communication between Class 2 Bluetooth-enabled devices by designing a server device capable of mediating Bluetooth communication between the two said devices when they are out of their normal range of communication but are within the server's range. The server device shall be used to receive files from the source device, verify the source and destination addresses by processing it and then forward it to the intended destination device as and when it becomes available within its scan area. The server device should also be able to approximate the distance between the source and the destination through RSSI proximity estimation. In order to keep track of all the incoming and outgoing transfers and the associated devices, the server should log all the metadata into an appropriate accessible location on the server itself which can be opened and viewed later by an administrator.

1.2 Background

We live in a technologically advanced era wherein people carry and use a plethora of mobile electronic devices & gadgets such as notebook computers, mobile phones, PDAs, digital cameras, portable video/audio players, position tracking devices, etc. These gadgets aid us in our professions as well as entertain us in our private lives. Earlier, for most parts these devices were used separately and did not often interact with one another. However, as we continue to immerse ourselves in automated cross-platform communication concepts such as the IoT, we have seen networking evolve alongside at a rapid pace. This has given rise to the ubiquitous usage of ad hoc wireless personal area networks across multiple standards in households and workspaces, which

are being preferred over the traditional wired LAN in the current era. These wireless PANs connect multiple devices of varying capabilities using a common standard that facilitates seamless flow of information between these devices. We see PANs becoming increasingly popular as they are made use of even in large laboratories wherein they are built using a hybrid architecture that also includes nodes for sensing ambient data and relaying it back to a central server for processing. IEEE 802.15 is a group that specifies the standards that are widely used in wireless PANs.

1.2.1 Bluetooth

The most commonly used wireless PAN standard is Bluetooth (IEEE 802.15.1) which combines a basic data-rate and a low-energy core configuration for an RF transceiver, baseband and protocol stack. Often used to exchange data over short distances from fixed or mobile devices, its popularity certainly is derived from its ubiquitous presence in 21st century gadgets. Nearly every piece of gadget, including many from the pre-WiFi era comes equipped with Bluetooth and the ease of its usage, requiring nothing but a device's physical address to communicate with it, has driven its popularity through the roof. However, one has to acknowledge that whilst the popularity of Bluetooth in lower power wireless PANs is undeniable, it does certainly carry a significant disadvantages going forward. For starters, Bluetooth's scalability has come into serious questioning after the emergence of other popular standards such as the WiFi (IEEE 802.11) that can seamlessly integrate wired LANs into wireless networks. Bluetooth being a low-powered mechanism comes with severe trade-off in terms of the overall range of operation and the throughput, making rate of transfer a liability when working under time constraints. More than the throughput, it is the range that primarily hinders ones' networking capabilities and this project is intended at solving that issue. Typically, the most commonly found class of Bluetooth-enabled devices i.e. Class 2 devices such mobile phones, laptops, PDAs, etc. can communicate within distances of up to ~10 meters, although this may vary slightly from device to device. From our research we found that patented industrial-grade Bluetooth range extenders/repeaters

are commercially available (e.g. AIRcable Host XR3, Parani SD-1000, SMA Bluetooth Repeater), but their cost (~\$150-300) and application are not suited to the environment we are targeting. Also the implementation of extenders and mimicking becomes overly complex with respect to the physical layer implementation of Bluetooth.

1.2.2 Literature Survey

The literature survey before starting this project is tabulated in the following table:

Paper/Algorithm surveyed	Description	Inferences
Apparatus, Method and System for a Bluetooth Repeater ^[4]	A patented apparatus for a repeater that receives incoming Bluetooth connections from a source device within range and then forwards the same data to an intended recipient that is outside the range of the originating device.	Provides a good understanding on how industrial-grade repeaters are designed in terms of hardware functionality whilst explaining the usage of translation tables to store source and destination addresses to insure proper forwarding.
Enhanced RSSI-based High Accuracy Real-Time User Location Tracking System for Indoor and Outdoor Environments ^[2]	Implements a low-complexity RSSI-tracking algorithm on a non-stationary wireless devices and analyses the results in both indoor and outdoor environments to propose deterministic RSSI coefficients and distance estimation for the device tracked.	Provides a fair idea on how RSSI estimation is used in tracking node positions through triangulation in multi-node networks whilst explaining how the method is not too reliable in an indoor environment owing to signal drops caused by physical barriers such as walls.

<p>Cordless RF Range Extension for Wireless Piconets ^[5]</p>	<p>A patented design for cordless telephone technology that provides a long range and highly sophisticated wireless extension between a plurality of wireless piconet networks. Proposes that the base unit and the remote handset of single device are made members of different wireless piconets to facilitate message transmission between one network to another outside RF range.</p>	<p>Provides a fair idea on how single-hop or multi-hop communication is established in multi-node networks and how any standard RF device can be reconfigured to behave as a hopping station between two other source and destination devices.</p>
<p>Bluetooth Range Extender for Audio ^[6]</p>	<p>Proposes a range extension mechanism for audio playback over Bluetooth by using an appropriate audio API to access and manipulate the audio stream. The manipulated audio stream is relayed through an intermediate Bluetooth microcontroller to the destination speaker device.</p>	<p>Provides a fair idea on what hardware and architecture can be used to achieve our project objective whilst also specifying open source packages that can be used to control Bluetooth functionalities in a Linux-based open source IDE.</p>
<p>Comparison of the IEEE 802.11, 802.15.1, 802.15.4 and 802.15.6 wireless standards ^[3]</p>	<p>Compares popular wireless standards such as WLAN/WiFi, ZigBee, Bluetooth, etc. outlining their differences and similarities with an emphasis on the physical layer.</p>	<p>Provides a good understanding of their modes of operation whilst specifying appropriate access protocols and theoretical limits on the data rate and range of operation.</p>

Table 1: Background research work

CHAPTER 2

PROJECT DESCRIPTION

2.1. Methodology

The Bluetooth server is designed to have six co-dependent modules:

1. Listening for Incoming Connections.
2. Directory Monitoring
3. Source and Destination Verification
4. Forwarding to the Destination
5. RSSI Distance Estimation
6. Logging of metadata

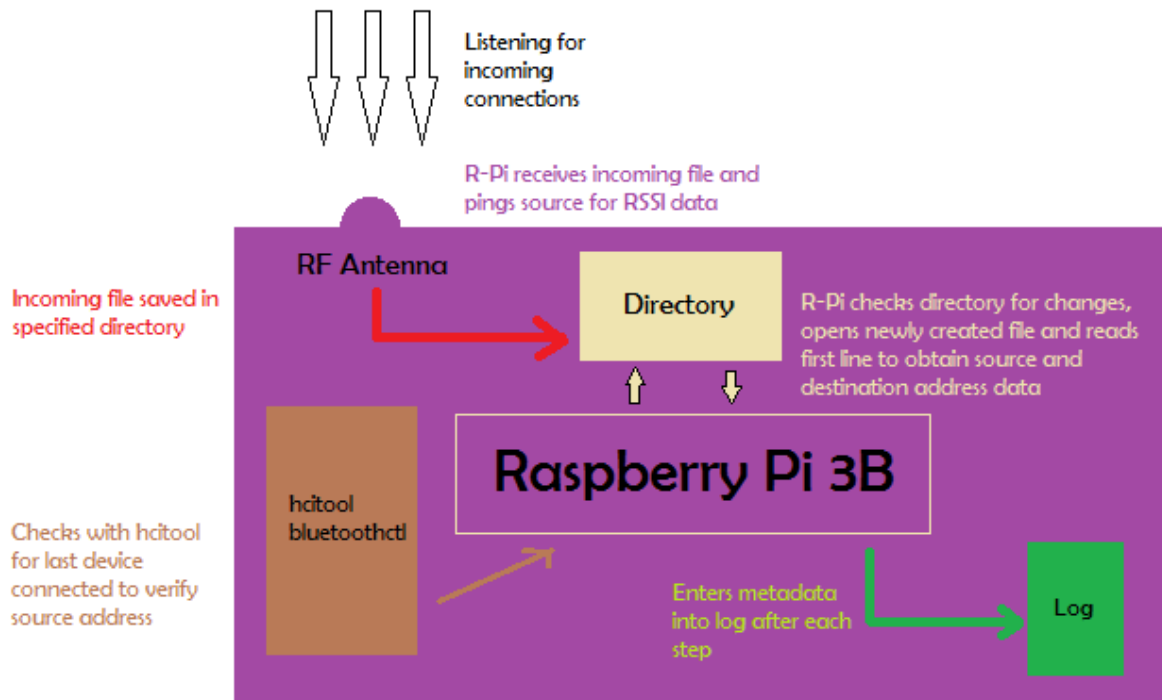


Fig. 1a: Project Modules – Receiving a file

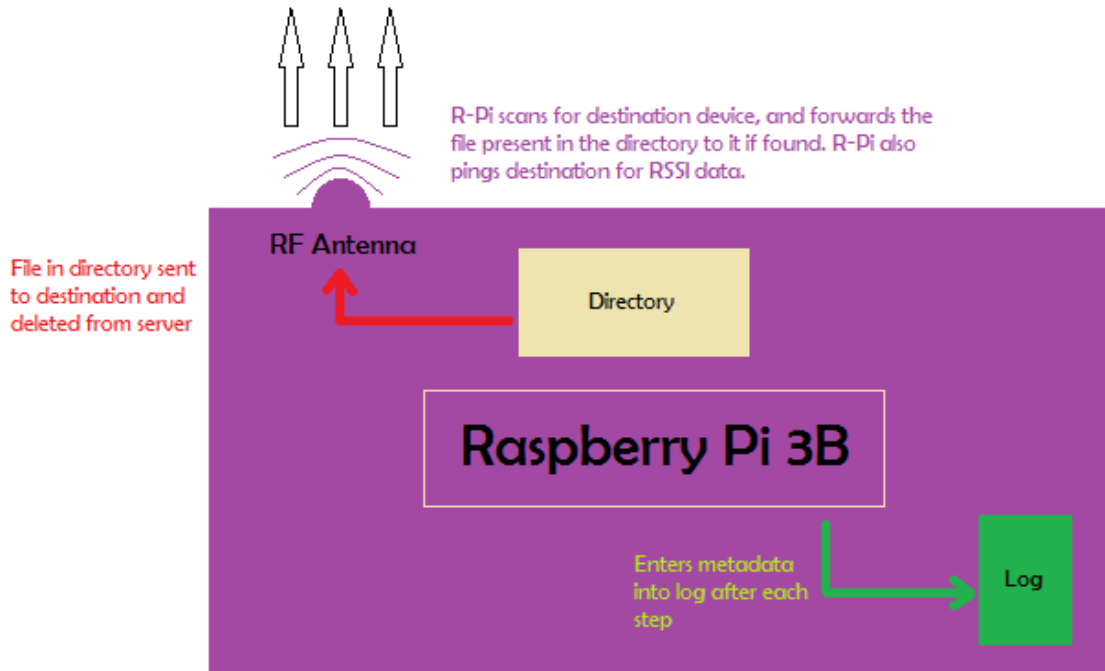


Fig. 1b: Project Modules – Forwarding a file

1. Listening for incoming connections:

For receiving files from other Bluetooth-enabled devices on the Raspberry Pi, we setup an OBEX push server over the command line, which configures the R-Pi by setting up a socket to receive files over the RF channel using the ‘Object Exchange’ communications protocol and save the received files to a designated directory. The socket is bound by default to Port 9 of the R-Pi and is set to listen to for incoming connections whenever the OBEX push server is enabled. The commands for setting up the server is saved as a service script bound to the built-in Bluetooth service and the service is then set to auto-start whenever the R-Pi boots thereby daemonizing the listening process for incoming connections. Whenever a device connects with the R-Pi and it receives a file from the device, the file can be found in the directory designated to the task.

2. Directory Monitoring:

Using the ‘glob’ module for Linux, a python script is set up to constantly monitor the directory designated for incoming connections for any changes/files added. If a

‘.txt’ file is discovered in the directory during this process, the script immediately invokes the address verification module.

3. Source and Destination verification:

This module invokes a python script which opens the newly received ‘.txt’ file for processing to obtain the source and destination address data which is meant to be the first line of the file as required by the design of this system. Upon reading the address data, the script invokes Linux’s in-built Bluetooth communications tool ‘hcitool’ to retrieve the address of the last connected device. If the address matches with the source address specified in the file, it initiates a Bluetooth scan for nearby devices. If the address of any of the discovered devices match that of the destination address specified in the file, it invokes the forwarding module. If either the source address does not match with the address of originating device or a device with the destination address is not found during the Bluetooth scan, the file is immediately deleted from the directory as it poses a security risk to the integrity of the server.

4. Forwarding to the Destination:

This module invokes a python script which requests the found destination device to list all its Bluetooth services. Then amongst the services, it looks for ‘OBEX Object Push’ service which utilizes the OBEX communications protocol to send a file (push an object) to the destination device. If the service is found, it initiates a file transfer with the destination device on the respective port and as the destination accepts the incoming transfer the file is successfully sent from the source to the destination via the server. Whether the destination accepts the file and the server successfully completes the transfer or if the destination rejects it and the server can’t complete the transfer, in either case the file is deleted from the server to avoid any potential security risks to the integrity of the server.

5. RSSI Distance Estimation:

This module runs a python script alongside both the incoming and outgoing modules to ping the source and the destination for RSSI data and estimate their

distances from the server and thus between one another. The RSSI values are measured over a course of 8 seconds with system-introduced delay of 1s between each measurement and the value is averaged for the best result. The relation model of the RSSI value obtained in dBm to the estimated distance as proposed by Texas Instruments for the Chipcon CC2420 radio is given as:

$$RSSI = -(10 \times n) \log_{10}(d) - A$$

where n is the propagation constant (1-3) which is determined by the Class of radio device used, d is the estimated distance in metres and A is the reference received signal strength in dBm, calibrated based on transmission power characteristics of device broadcasted with RSSI or RSSI value measure when the separation distance between received and transmitter is one meter ^[1]. We take the above relation model for this project as no other concrete relation models have been published or widely known yet. Once the distance between the server and the source and destination is estimated using the RSSI values, the server then calculates the distance of transmission between the source and destination by triangulation and invokes the logging module to log the value as metadata.

6. Logging of metadata:

This module runs a python script that redirects the Python's default output `sys.stdout()` to a log file instead of the UI. This script runs alongside other modules and thus all metadata information being published on the screen is written to a log file instead in the absence of an output peripheral. This log file can be opened at any point by a server administrator to check of past activity.

2.2. Design and Approach

2.2.1. System design and description:

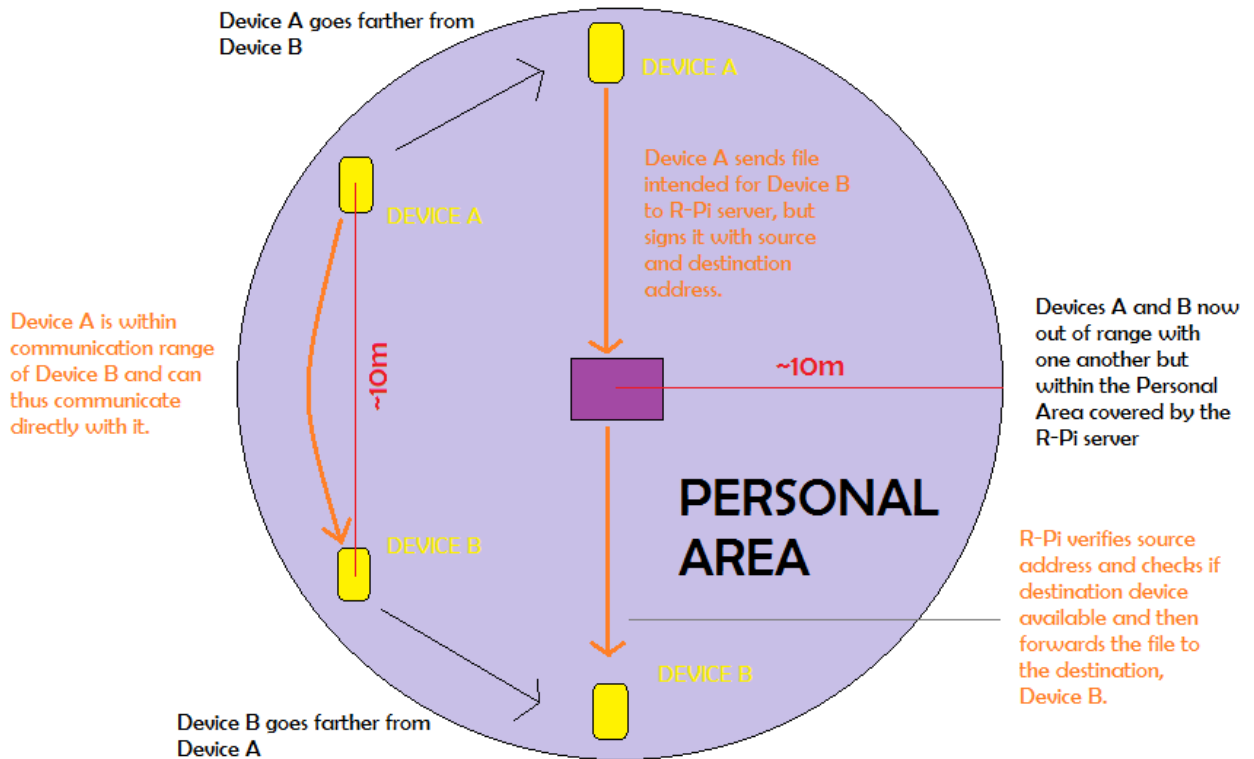


Fig. 2: System Block Diagram

Description:

The system designed is as shown above and it considers the following scenario.

“Two Bluetooth-enabled devices are able to wirelessly communicate and transfer data between each other. However, there may be circumstances where these devices will go out of the Bluetooth operation range (~10m) but still remain within the same work area. In such a case, data transfer between these two devices should still be possible using only Bluetooth.”

Thus, to facilitate such an environment, we place our designed Bluetooth server in the center that serves a mediation device forming a momentary single-hop network to aid the file transfer from one device to another. Although the server does mimick

the functionality of a Bluetooth repeater or a range extender, the outcome is that the Device A & B are able to communicate with one another despite being out of one another's range of communication. One primary concern in achieving proper end to end transmission in the server side was identification of the source and destination. Since we are using the OBEX communication protocol, it only allows us to push objects and does not give us packet or frame-level access for us to include overheads that could be used to define the physical addresses of the source and destination devices. Hence, for the sake of demonstration, we limited our transmissions to '.txt' files only as it allows for easier data processing and manipulation. Hence, when a device wishes to send another device any data, it can create a '.txt' with the data to be sent but has to include the source and destination addresses in its first line which behaves like an overhead that the server can process upon receiving the file.

2.2.2. Project flow description:

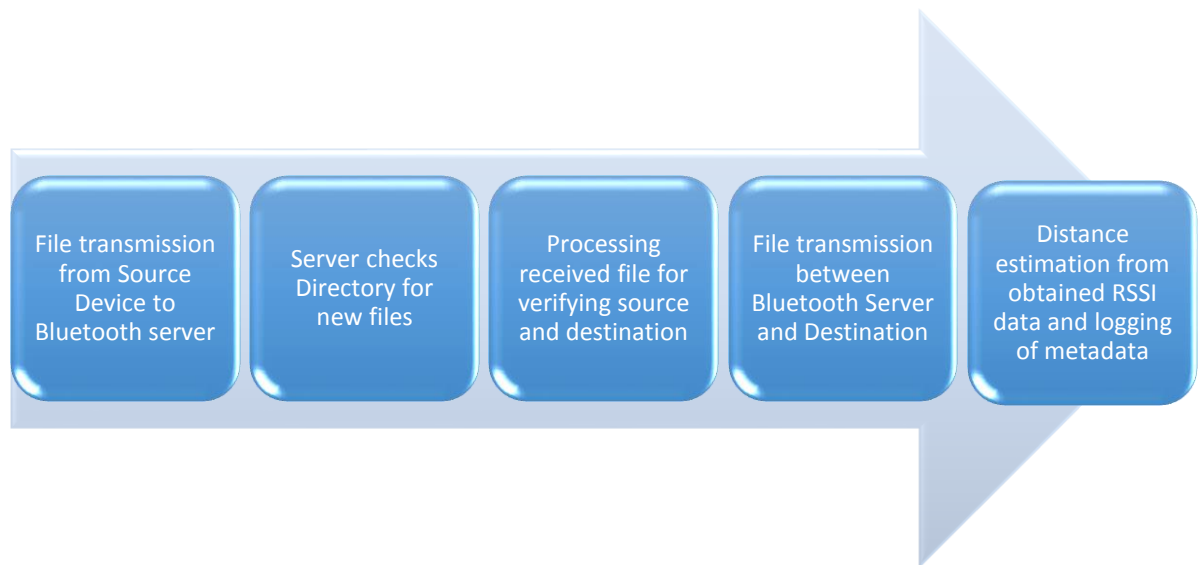


Fig. 3: Project Flow Diagram

Step 1: The source device sends a file to the Bluetooth server which is listening for incoming connections on Port 9.

Step 2: The server receives the file, stores it in the designated directory, obtains RSSI data from the source, disconnects and invokes the ‘glob’ module to check the directory for changes.

Step 3: As a new file is detected in the directory, it is opened and its first line is read to obtain the physical addresses of the source and destination as specified by the source. The address obtained are verified against the last connection for source and scanned devices for destination.

Step 4: The server connects to the destination device on its respective OBEX Port, forwards the file, obtains RSSI data from the destination and disconnects.

Step 5: The server estimates the distance between the source and the destination using the logged RSSI data and then logs all other metadata into the log file and waits for the next transmission.

2.3. Standards, Protocols, Packages:

- IEEE 802.15.1: Bluetooth
- OBEX Communication Protocol
- Raspbian 4.4 Packages:
 - BLUEZ
 - OBEXPUSHD
 - OPENOBEX-APPS
 - BLUETOOTH-PROXIMITY
- Python 2.7 Packages:
 - PYTHON-BLUEZ
 - PYTHON-DEV
 - LIGHTBLUE
 - PYOBEX
- Software platforms used:
 - Apache 2.4.23 HTTP Server for Windows
 - PHP 5.6.25
 - Bennett 1.15 Bluetooth RSSI Monitoring for Windows

2.4. Constraints, Alternatives and Tradeoffs

Constraints:

- We initially planned to build this entire project on a Windows platform using a web-server and local cloud for automatic synchronization support as we thought designing an application for the most commonly used commercial platform would allow us to effectively transform any Bluetooth-enabled PC/laptop into a server that serves our purpose. But the Microsoft Bluetooth stack offers limited API access and does not support remote toggling or daemonization of server processes.
- RSSI distance estimation based on the relation given by Texas Instruments does not account for signal drops in indoor setting due to physical obstructions such as walls, large objects in the line-of-sight of communication, etc. and thus this leads to bloated values when the server or the devices are not positioned optimally with respect to one another.
- Typically used Bluetooth communication protocols such as L2CAP or RFCOMM could not be used for the file transfer as setting them up as a daemonized process bound to the built-in R-Pi Bluetooth process was impossible due to these channels communicating on different ports for receiving and transmitting.
- Using the OBEX communication protocol allows us to transfer any file between the devices and servers, but it does not support packet level access and thus we cannot specify source or destination address in the form of an overhead like we can in other protocols that grant packet level access.
- Bluetooth visibility uptime of devices is commonly very short and thus in some cases, a file sent to the server may not have its destination discovered in the first scan by the server and hence maybe deleted without any forwarding to avoid security risks.
- All devices that utilize the server will have to go through a one-time manual pairing process as automated pairing can often timeout due to low uptime on device discovery. However, this process also increases the security of the

prototype as automatic pairing could be vulnerable to exploits by malicious intrusions.

Alternatives:

- We moved to an open source Linux-based platform which allowed for greater API support and better remote and code-level access to Bluetooth functionalities to overcome our difficulties working with the Microsoft Bluetooth Stack.
- RSSI values taken over a period of time and averaged produced a better result of estimating distances even in an indoor setting despite hindrances in the line-of-sight communication.
- Using the OBEX protocol in the place of RFCOMM/L2CAP allowed us greater flexibility with setting up a daemonized process that created a socket which could listen to incoming Bluetooth connections at all times.
- Despite the lack of packet-level access in the OBEX protocol, one could still ensure some level of authentication through verification of source and destination addresses if they are added to the files being sent in a readable format such using the first line of '.txt' files as we have done in our demonstration.

Trade Offs:

- Use of an industrial quality Bluetooth dongle instead of the inbuilt Raspberry Pi RF chip can increase the accuracy of the RSSI measurements and also yield better transmission results by reducing connection drops, but that would have significantly increased the cost of the project.
- Since Bluetooth is a standard that permits only one connection at a time, the server is also limited to the same wherein it cannot handle multiple incoming connection requests, in which case, one device or the other will be randomly prioritized and the other device will have to wait it out, thus denting the speed of operation in some cases.

2.5. Technical Specifications

1. Raspberry Pi 3 Model B:

- SoC: Broadcom BCM2837
- CPU: 1.2 GHZ quad-core ARM Cortex A53 (ARMv8 Instruction Set)
- GPU: Broadcom VideoCore IV @ 400 MHz
- Memory: 1 GB LPDDR2-900 SDRAM
- USB ports: 4
- Network: 10/100 MBPS Ethernet, 802.11n Wireless LAN, Bluetooth 4.0
- GPIO Pins: 40
- HDMI port: 1
- Camera Interface: 1
- Audio I/O: 3.5mm Composite Jack
- Storage: micro-SD

Schematic:

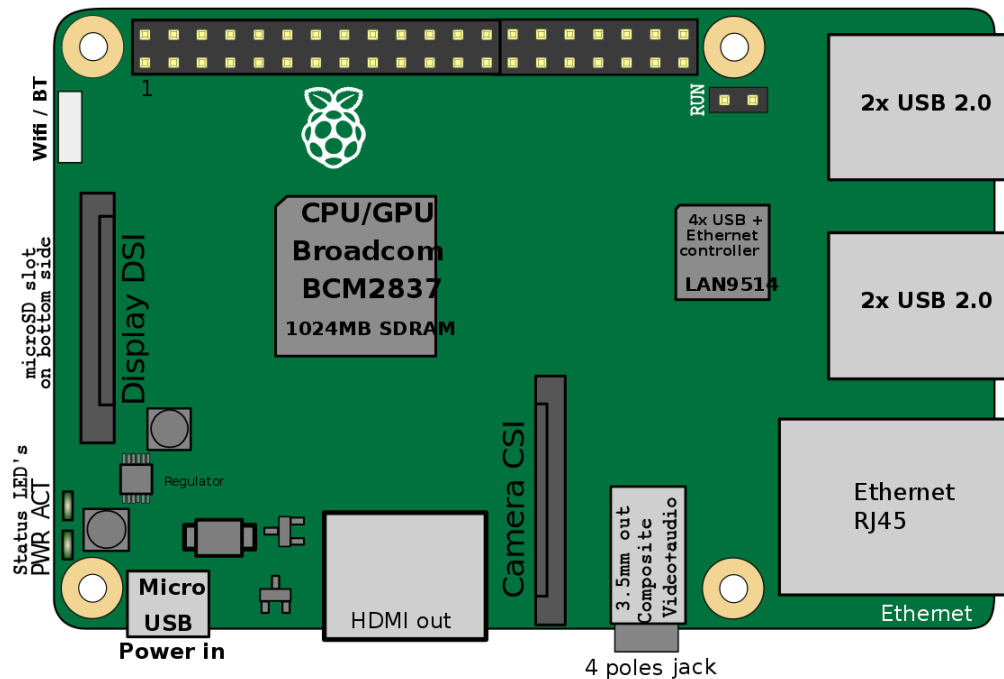


Fig. 4: Raspberry Pi 3B Schematic

CHAPTER 3

PROJECT DEMONSTRATION

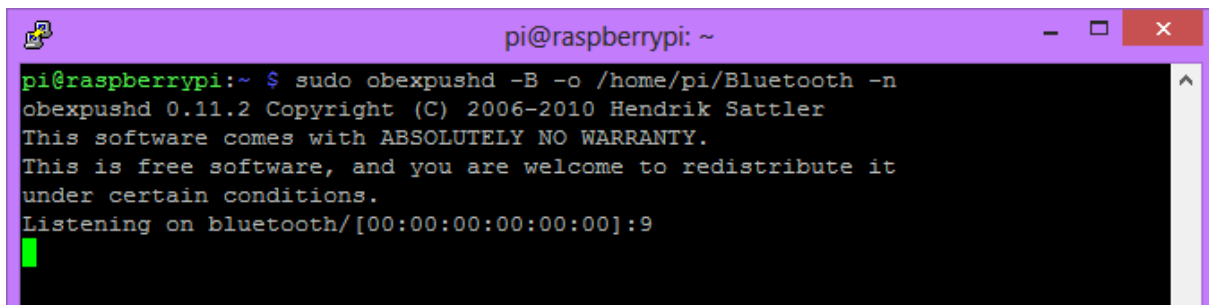
3.1 Hardware Module:

The hardware module for this project consists of a Raspberry Pi 3 Model B loaded with respective Raspbian and Python packages mentioned in the earlier sections. No other peripherals are required to set up the Bluetooth server as the Python scripts and the automated process scripts are daemonized and start functioning as soon as the R-Pi is connected to a power source and is booted up. The idea is to have the R-Pi powered up and sitting in a central mantle such that it is within the communication range of all the corners of a work area much like the WiFi routers that are setup for establishing a WLAN within a workspace.



Fig. 5: Hardware module

3.2 Incoming Connection Socket:

A terminal window titled 'pi@raspberrypi: ~' with a purple title bar. The terminal output shows the command 'sudo obexpushd -B -o /home/pi/Bluetooth -n' being executed. The output includes the version '0.11.2', copyright information for Hendrik Sattler (2006-2010), a disclaimer about warranty, and a status message: 'Listening on bluetooth/[00:00:00:00:00:00]:9'. A green cursor is visible on the line following the status message.

```
pi@raspberrypi:~ $ sudo obexpushd -B -o /home/pi/Bluetooth -n
obexpushd 0.11.2 Copyright (C) 2006-2010 Hendrik Sattler
This software comes with ABSOLUTELY NO WARRANTY.
This is free software, and you are welcome to redistribute it
under certain conditions.
Listening on bluetooth/[00:00:00:00:00:00]:9
█
```

Fig. 6a: Setting up OBEX Push Server on Command Line

Here we set up the OBEX Push server using the ‘obexpushd’ package and set it to allow incoming Bluetooth connections on the default port. By default, it uses Port 9 and it displays the status message that the server is listening on Port 9. We can also designate a directory for the incoming files as done above in the command line execution.

Following this we create a text file on a nearby Bluetooth-enabled device intended for another device, specify both their addresses in the file in the first line followed by the information to be exchanged and send the file to the Raspberry Pi server.

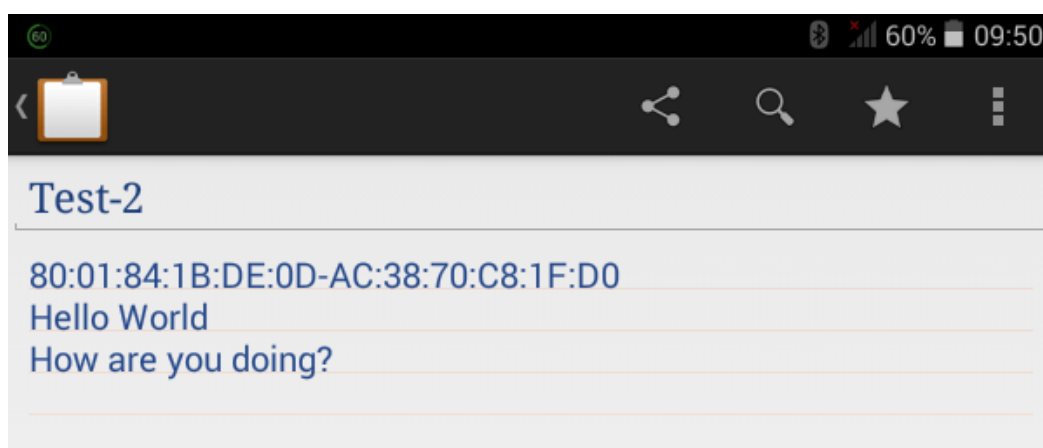


Fig. 6b: Creating a Test ‘.txt’ file on Device A

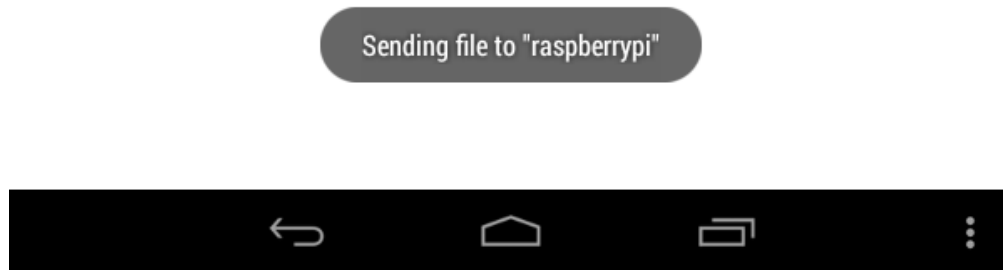


Fig. 6c: Sending Test file from Device A to R-Pi

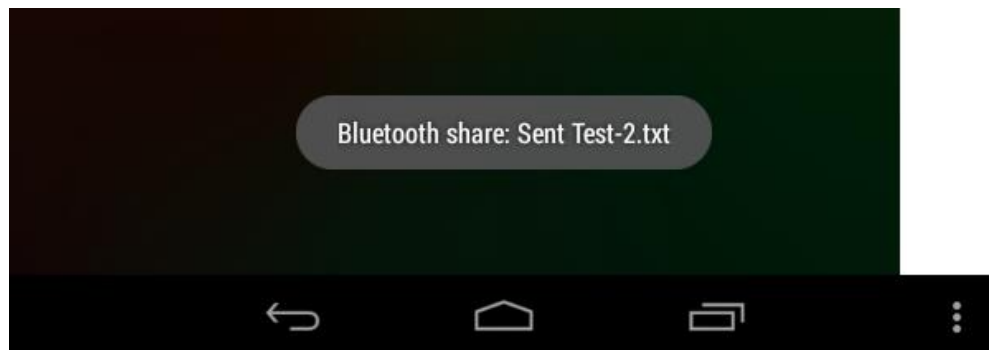
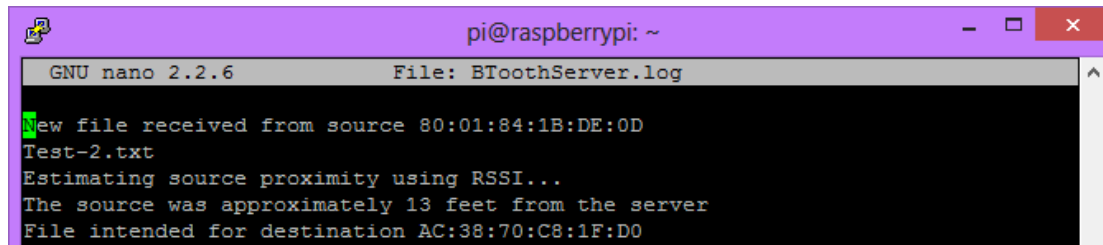


Fig. 6d: Test file sent from Device A to R-Pi

```
pi@raspberrypi: ~  
pi@raspberrypi:~ $ sudo obexpushd -B -o /home/pi/Bluetooth -n  
obexpushd 0.11.2 Copyright (C) 2006-2010 Hendrik Sattler  
This software comes with ABSOLUTELY NO WARRANTY.  
This is free software, and you are welcome to redistribute it  
under certain conditions.  
Listening on bluetooth/[00:00:00:00:00:00]:9  
Creating file "/home/pi/Bluetooth/Test-2.txt"
```

Fig. 6e: Test file received in the designated directory in R-Pi

3.3 Directory Monitoring and Verification of Source & Destination addresses:



```
pi@raspberrypi: ~  
GNU nano 2.2.6 File: BToothServer.log  
New file received from source 80:01:84:1B:DE:0D  
Test-2.txt  
Estimating source proximity using RSSI...  
The source was approximately 13 feet from the server  
File intended for destination AC:38:70:C8:1F:D0
```

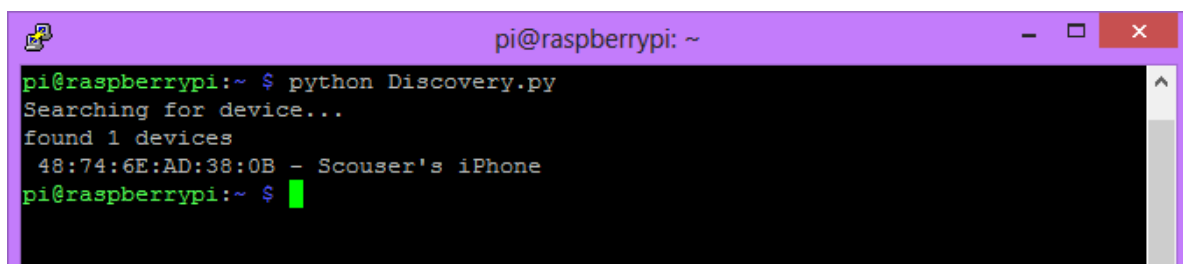
Fig. 7: Designated directory scanned for new files

Here we have the python script that uses the ‘glob’ module to scan the designated incoming files directory to check for new files. Upon encountering a new file, it opens the file and reads the first line to identify the source and destination physical addresses. Upon obtaining these addresses it verifies the same by checking ‘hcitool’ and undertaking a scan of visible devices. Meanwhile, the RSSI estimation module pings the source device over a period of 8 seconds and returns an average distance estimate in feet.

3.4 Forwarding to the destination

3.4.1 Scanning for devices in vicinity

Upon identification of the destination physical address, the server carries out a simple Bluetooth scan that returns the addresses of all devices in communication range. The server verifies the presence of destination by comparing all discovered devices’ address with the specified destination address.

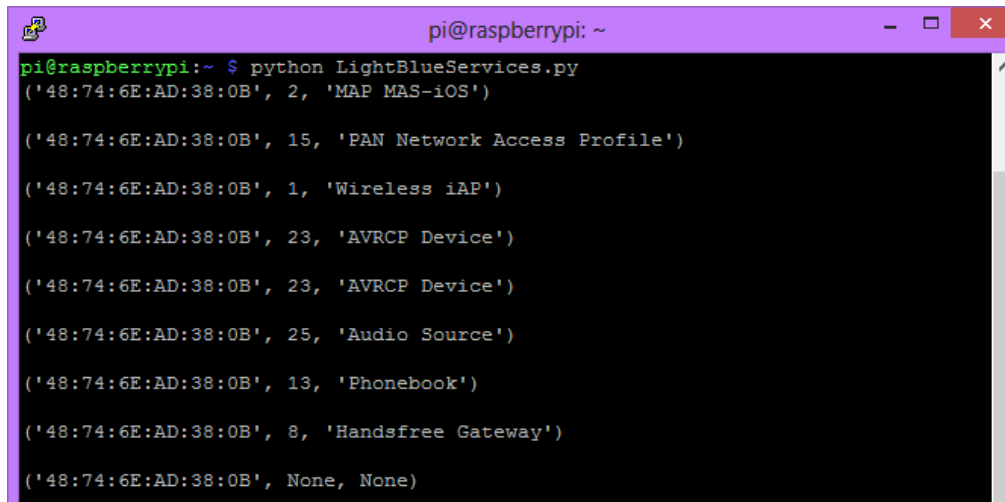


```
pi@raspberrypi: ~  
pi@raspberrypi:~ $ python Discovery.py  
Searching for device...  
found 1 devices  
48:74:6E:AD:38:0B - Scouser's iPhone  
pi@raspberrypi:~ $
```

Fig. 8a: Devices found in scan

3.4.2 Scanning for services

Once a device matching the destination physical address is found, the server carries out a search of the Bluetooth services offered by the destination device using the Service Discovery Protocol and lists them.

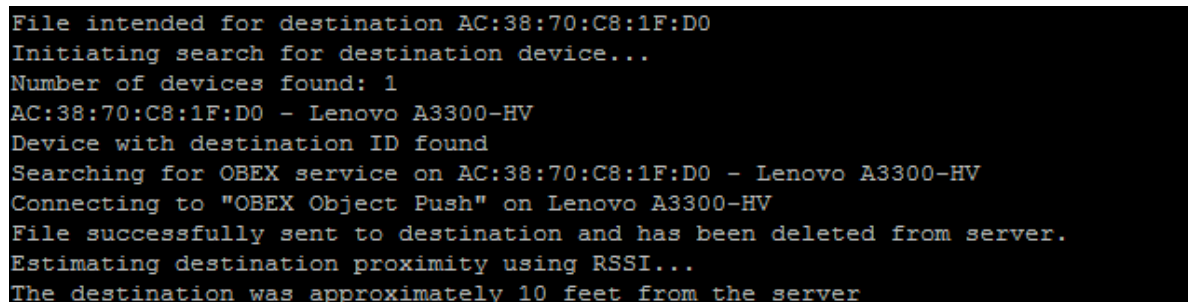


```
pi@raspberrypi: ~  
pi@raspberrypi:~ $ python LightBlueServices.py  
(('48:74:6E:AD:38:0B', 2, 'MAP MAS-iOS'))  
  
(('48:74:6E:AD:38:0B', 15, 'PAN Network Access Profile'))  
  
(('48:74:6E:AD:38:0B', 1, 'Wireless iAP'))  
  
(('48:74:6E:AD:38:0B', 23, 'AVRCP Device'))  
(('48:74:6E:AD:38:0B', 23, 'AVRCP Device'))  
  
(('48:74:6E:AD:38:0B', 25, 'Audio Source'))  
  
(('48:74:6E:AD:38:0B', 13, 'Phonebook'))  
  
(('48:74:6E:AD:38:0B', 8, 'Handsfree Gateway'))  
  
(('48:74:6E:AD:38:0B', None, None))
```

Fig. 8b: Services offered by the device

3.4.3 Using OBEX service to forward the file

Once the server retrieves the list of services offered by the destination device, it looks for the ‘OBEX Object Push’ service to initiate the file transfer to the device on the associated port. Meanwhile, the RSSI estimation module pings the destination device over a period of 8 seconds and returns an average distance estimate in feet.



```
File intended for destination AC:38:70:C8:1F:D0  
Initiating search for destination device...  
Number of devices found: 1  
AC:38:70:C8:1F:D0 - Lenovo A3300-HV  
Device with destination ID found  
Searching for OBEX service on AC:38:70:C8:1F:D0 - Lenovo A3300-HV  
Connecting to "OBEX Object Push" on Lenovo A3300-HV  
File successfully sent to destination and has been deleted from server.  
Estimating destination proximity using RSSI...  
The destination was approximately 10 feet from the server
```

Fig. 8c: The file is forwarded to the specified destination

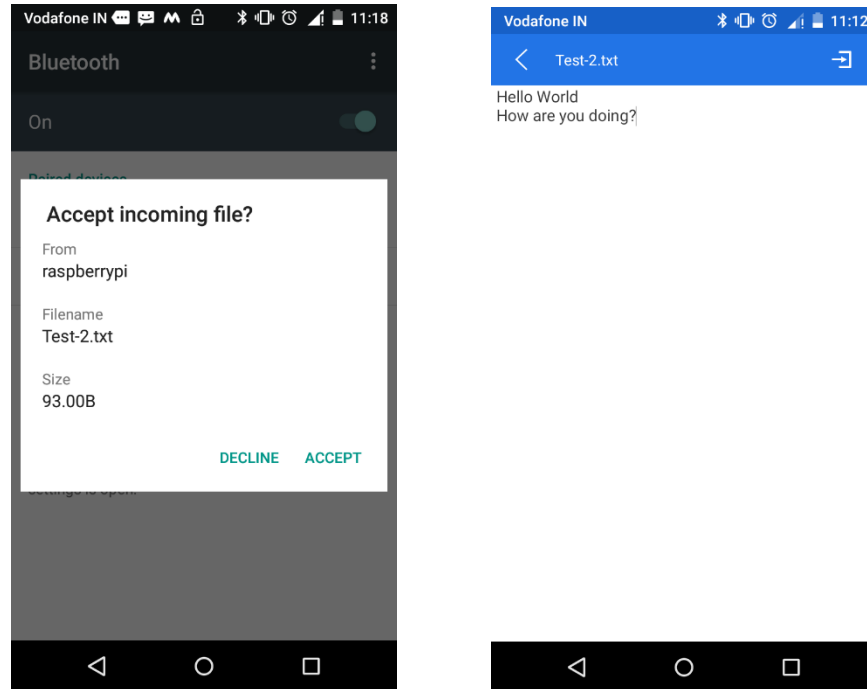


Fig. 8d: Device B receives the file successfully

3.5 RSSI-based distance estimation

Initially, to establish the normal range of operation between two Class 2 Bluetooth-enabled devices, we carried out real-time tracking of RSSI values as two paired devices were moved away from one another until they were out of communication range and then moved back in. This was done on a Windows platform using an application called Bennett which makes use of the BroadComm Bluetooth API to measure RSSI data for Bluetooth Classic devices. The software was made to automatically log all stable values received in a log file which was then processed using a PHP script to estimate the distance. In addition to this, we also ran our RSSI-tracking python script in a loop on the R-Pi server to estimate its maximum range of Bluetooth communication as we tracked another device that was made to slowly move out of range and back in.

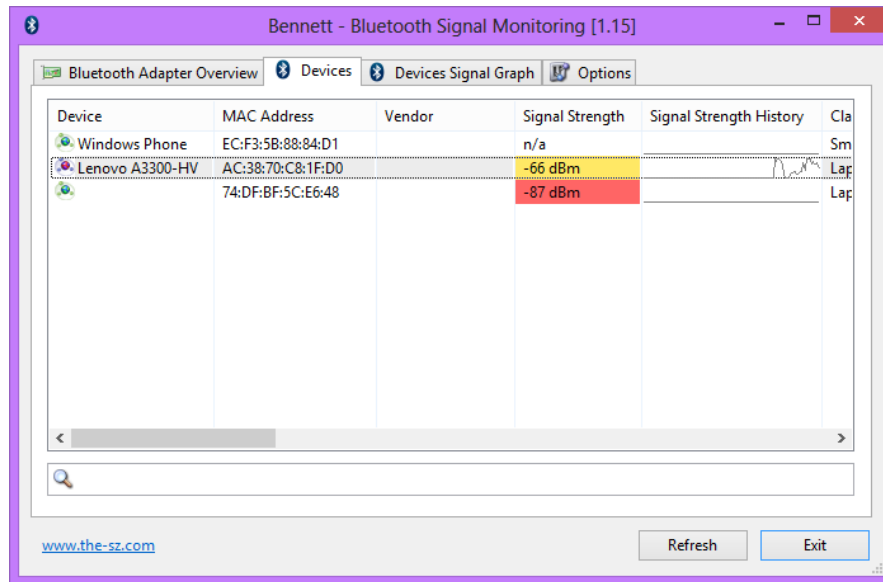
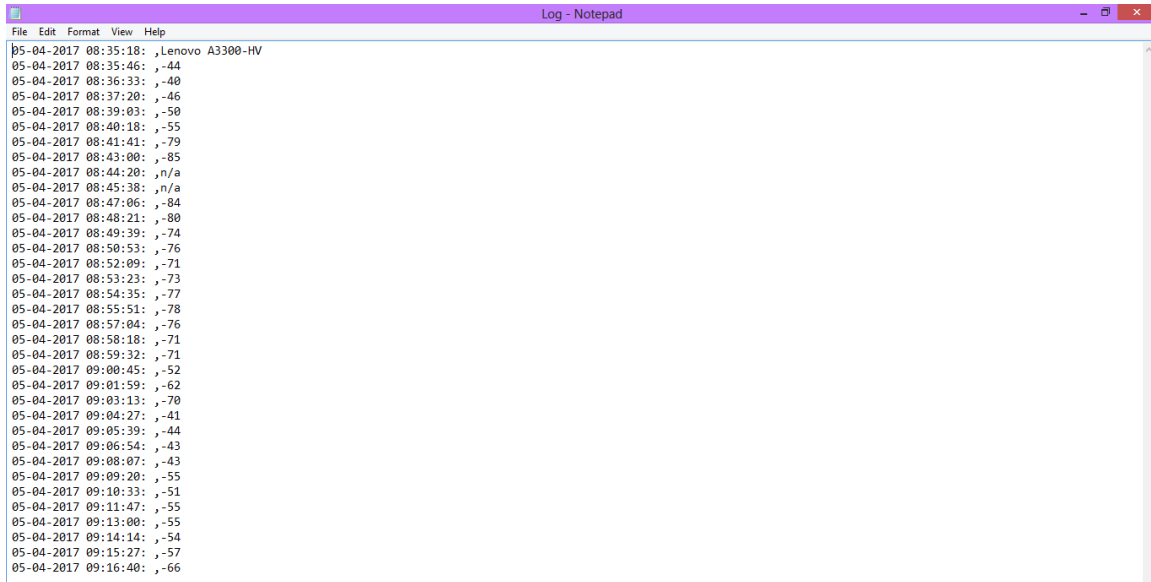


Fig. 9a: Bennett monitors RSSI values of paired device ‘Lenovo A3300-HV’



Fig. 9b: Graph of the RSSI values obtained as generated by Bennett



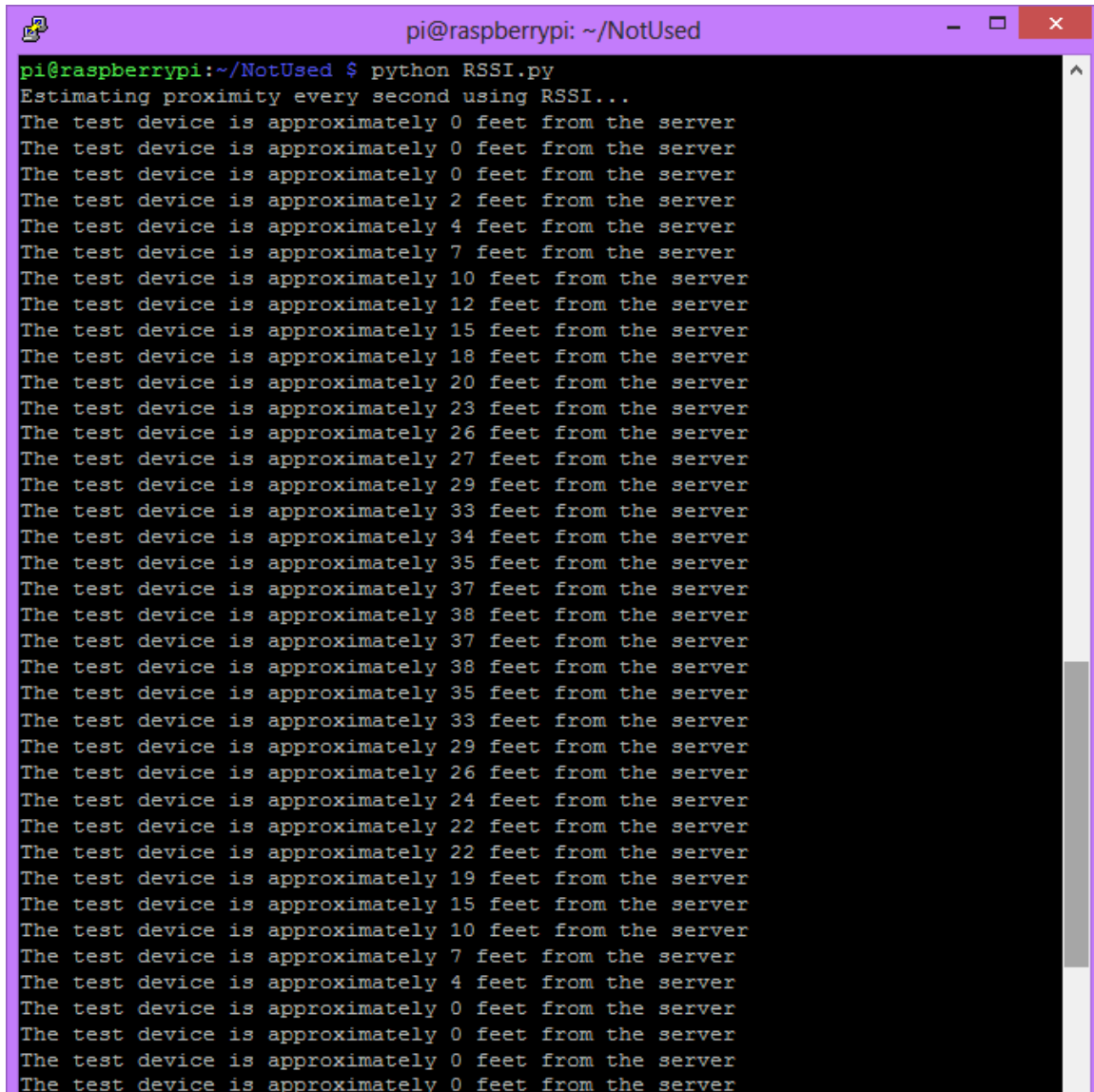
```
File Edit Format View Help
05-04-2017 08:35:18: ,Lenovo A3300-HV
05-04-2017 08:35:46: ,-44
05-04-2017 08:36:33: ,-40
05-04-2017 08:37:20: ,-46
05-04-2017 08:39:03: ,-50
05-04-2017 08:40:18: ,-55
05-04-2017 08:41:41: ,-79
05-04-2017 08:43:00: ,-85
05-04-2017 08:44:20: ,n/a
05-04-2017 08:45:38: ,n/a
05-04-2017 08:47:06: ,-84
05-04-2017 08:48:21: ,-80
05-04-2017 08:49:39: ,-74
05-04-2017 08:50:53: ,-76
05-04-2017 08:52:09: ,-71
05-04-2017 08:53:23: ,-73
05-04-2017 08:54:35: ,-77
05-04-2017 08:55:51: ,-78
05-04-2017 08:57:04: ,-76
05-04-2017 08:58:18: ,-71
05-04-2017 08:59:32: ,-71
05-04-2017 09:00:45: ,-52
05-04-2017 09:01:59: ,-62
05-04-2017 09:03:13: ,-70
05-04-2017 09:04:27: ,-41
05-04-2017 09:05:39: ,-44
05-04-2017 09:06:54: ,-43
05-04-2017 09:08:07: ,-43
05-04-2017 09:09:20: ,-55
05-04-2017 09:10:33: ,-51
05-04-2017 09:11:47: ,-55
05-04-2017 09:13:00: ,-55
05-04-2017 09:14:14: ,-54
05-04-2017 09:15:27: ,-57
05-04-2017 09:16:40: ,-66
```

Fig. 9c: Log of the RSSI values obtained as generated by Bennett

BLUETOOTH DISTANCE ESTIMATION

```
05-04-2017 08:35:18: Lenovo A3300-HV
05-04-2017 08:35:46: Calibrating...
05-04-2017 08:36:33: Calibrating...
05-04-2017 08:37:20: Calibrating...
Calibration complete, Transmission power (Tx) calculated to be roughly -43dB.
05-04-2017 08:39:03: Received Signal Strength: -50dB Distance: 0-1 meters approximately.
05-04-2017 08:40:18: Received Signal Strength: -55dB Distance: 0-1 meters approximately.
05-04-2017 08:41:41: Received Signal Strength: -79dB Distance: 8-12 meters approximately.
05-04-2017 08:43:00: Received Signal Strength is -85dB or lower. Device has gone out of communication range.
05-04-2017 08:44:20: Received Signal Strength is -85dB or lower. Device has gone out of communication range.
05-04-2017 08:45:38: Received Signal Strength is -85dB or lower. Device has gone out of communication range.
05-04-2017 08:47:06: Received Signal Strength: -84dB Distance: 15-19 meters approximately.
05-04-2017 08:48:21: Received Signal Strength: -80dB Distance: 9-13 meters approximately.
05-04-2017 08:49:39: Received Signal Strength: -74dB Distance: 4-6 meters approximately.
05-04-2017 08:50:53: Received Signal Strength: -76dB Distance: 6-8 meters approximately.
05-04-2017 08:52:09: Received Signal Strength: -71dB Distance: 2-4 meters approximately.
05-04-2017 08:53:23: Received Signal Strength: -73dB Distance: 4-6 meters approximately.
05-04-2017 08:54:35: Received Signal Strength: -77dB Distance: 6-8 meters approximately.
05-04-2017 08:55:51: Received Signal Strength: -78dB Distance: 7-9 meters approximately.
05-04-2017 08:57:04: Received Signal Strength: -76dB Distance: 6-8 meters approximately.
05-04-2017 08:58:18: Received Signal Strength: -71dB Distance: 2-4 meters approximately.
05-04-2017 08:59:32: Received Signal Strength: -71dB Distance: 2-4 meters approximately.
05-04-2017 09:00:45: Received Signal Strength: -52dB Distance: 0-1 meters approximately.
05-04-2017 09:01:59: Received Signal Strength: -62dB Distance: 0-2 meters approximately.
05-04-2017 09:03:13: Received Signal Strength: -70dB Distance: 2-4 meters approximately.
05-04-2017 09:04:27: Received Signal Strength: -41dB Distance: 0-1 meters approximately.
05-04-2017 09:05:39: Received Signal Strength: -44dB Distance: 0-1 meters approximately.
05-04-2017 09:06:54: Received Signal Strength: -43dB Distance: 0-1 meters approximately.
05-04-2017 09:08:07: Received Signal Strength: -43dB Distance: 0-1 meters approximately.
05-04-2017 09:09:20: Received Signal Strength: -55dB Distance: 0-1 meters approximately.
05-04-2017 09:10:33: Received Signal Strength: -51dB Distance: 0-1 meters approximately.
05-04-2017 09:11:47: Received Signal Strength: -55dB Distance: 0-1 meters approximately.
05-04-2017 09:13:00: Received Signal Strength: -55dB Distance: 0-1 meters approximately.
05-04-2017 09:14:14: Received Signal Strength: -54dB Distance: 0-1 meters approximately.
05-04-2017 09:15:27: Received Signal Strength: -57dB Distance: 0-1 meters approximately.
05-04-2017 09:16:40: Received Signal Strength: -66dB Distance: 1-3 meters approximately.
```

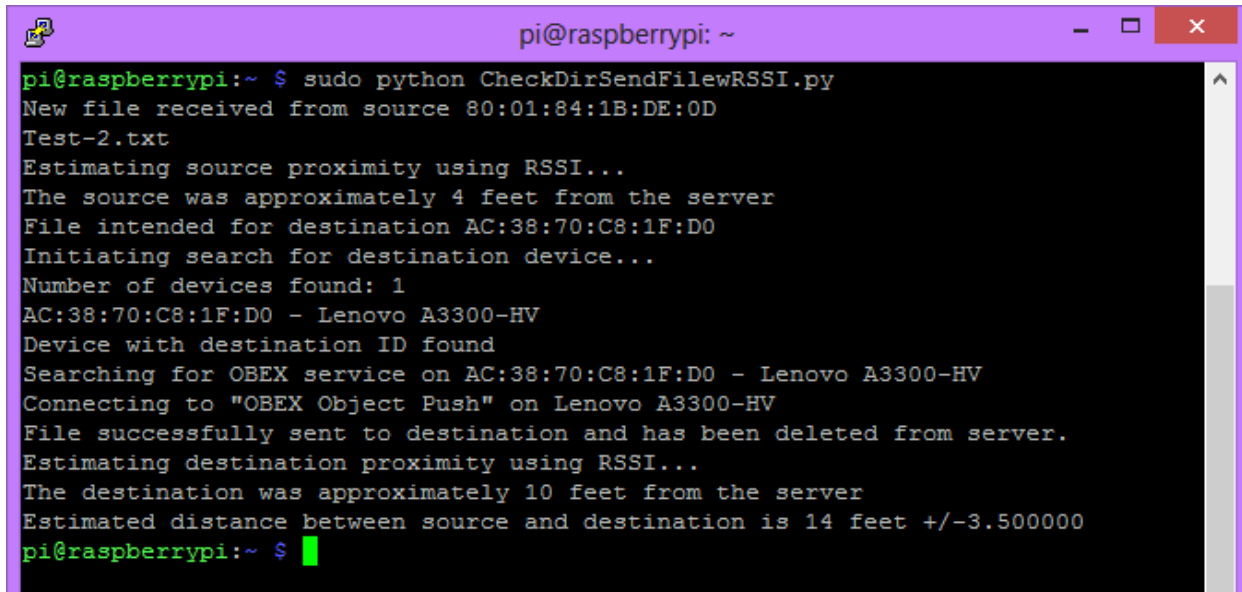
Fig. 9d: The distance estimation page created by the webserver PHP script that processes the Bennett log



```
pi@raspberrypi: ~/NotUsed
pi@raspberrypi:~/NotUsed $ python RSSI.py
Estimating proximity every second using RSSI...
The test device is approximately 0 feet from the server
The test device is approximately 0 feet from the server
The test device is approximately 0 feet from the server
The test device is approximately 2 feet from the server
The test device is approximately 4 feet from the server
The test device is approximately 7 feet from the server
The test device is approximately 10 feet from the server
The test device is approximately 12 feet from the server
The test device is approximately 15 feet from the server
The test device is approximately 18 feet from the server
The test device is approximately 20 feet from the server
The test device is approximately 23 feet from the server
The test device is approximately 26 feet from the server
The test device is approximately 27 feet from the server
The test device is approximately 29 feet from the server
The test device is approximately 33 feet from the server
The test device is approximately 34 feet from the server
The test device is approximately 35 feet from the server
The test device is approximately 37 feet from the server
The test device is approximately 38 feet from the server
The test device is approximately 37 feet from the server
The test device is approximately 38 feet from the server
The test device is approximately 35 feet from the server
The test device is approximately 33 feet from the server
The test device is approximately 29 feet from the server
The test device is approximately 26 feet from the server
The test device is approximately 24 feet from the server
The test device is approximately 22 feet from the server
The test device is approximately 22 feet from the server
The test device is approximately 19 feet from the server
The test device is approximately 15 feet from the server
The test device is approximately 10 feet from the server
The test device is approximately 7 feet from the server
The test device is approximately 4 feet from the server
The test device is approximately 0 feet from the server
The test device is approximately 0 feet from the server
The test device is approximately 0 feet from the server
The test device is approximately 0 feet from the server
```

Fig. 9e: Range estimation of the R-Pi server

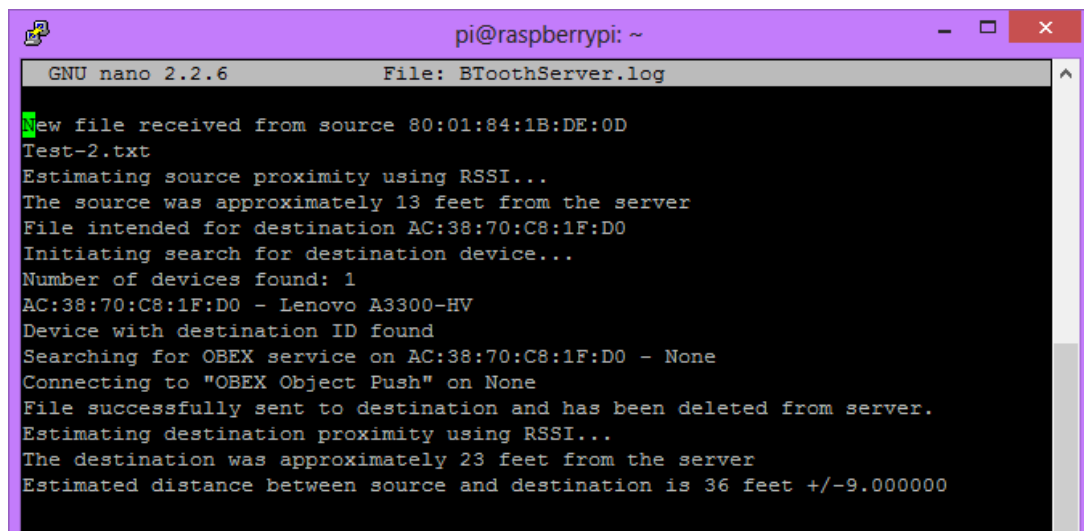
From the above experiment, we were able to establish that the range of operation for Class 2 Bluetooth-enabled devices is ~10-12m as seen from the RSSI (dBm) values >-80 . Carrying through this relation and algorithm, we coded a Python script that does the same for the devices connecting to our Bluetooth R-Pi server.



```
pi@raspberrypi: ~  
pi@raspberrypi:~ $ sudo python CheckDirSendFilewRSSI.py  
New file received from source 80:01:84:1B:DE:0D  
Test-2.txt  
Estimating source proximity using RSSI...  
The source was approximately 4 feet from the server  
File intended for destination AC:38:70:C8:1F:D0  
Initiating search for destination device...  
Number of devices found: 1  
AC:38:70:C8:1F:D0 - Lenovo A3300-HV  
Device with destination ID found  
Searching for OBEX service on AC:38:70:C8:1F:D0 - Lenovo A3300-HV  
Connecting to "OBEX Object Push" on Lenovo A3300-HV  
File successfully sent to destination and has been deleted from server.  
Estimating destination proximity using RSSI...  
The destination was approximately 10 feet from the server  
Estimated distance between source and destination is 14 feet +/-3.500000  
pi@raspberrypi:~ $
```

Fig. 9f: RSSI distance estimation between the source and destination

3.6 Logging of metadata:



```
GNU nano 2.2.6 File: BluetoothServer.log  
New file received from source 80:01:84:1B:DE:0D  
Test-2.txt  
Estimating source proximity using RSSI...  
The source was approximately 13 feet from the server  
File intended for destination AC:38:70:C8:1F:D0  
Initiating search for destination device...  
Number of devices found: 1  
AC:38:70:C8:1F:D0 - Lenovo A3300-HV  
Device with destination ID found  
Searching for OBEX service on AC:38:70:C8:1F:D0 - None  
Connecting to "OBEX Object Push" on None  
File successfully sent to destination and has been deleted from server.  
Estimating destination proximity using RSSI...  
The destination was approximately 23 feet from the server  
Estimated distance between source and destination is 36 feet +/-9.000000
```

Fig. 10: The log file generated in server during one entire single-hop transfer

CHAPTER 4

COST ANALYSIS

Material	Quantity	Cost Price (Rounded value in Rs)
Raspberry Pi 3B	1	3000
microSD Card	1	500
Power Cord	1	300
Ethernet Cable	1	300
Miscellaneous	-	300
TOTAL		4400

Table 2: Cost Price Tabulation

If produced commercially, this prototype can be converted into a compact unit costing not more than Rs.4000 (\$65). Even if it is sold at roughly Rs.5000 (\$80), keeping a narrow margin of profit, it is still capable of achieving the same job as industrial-grade repeaters which cost roughly about twice as much on average and occupy a lot of space in the work area in comparison to the credit-card sized R-Pi. However, it cannot be denied that building a makeshift Bluetooth server for out of an R-Pi for single-hop communication lacks longevity and scalability in comparison to the patented industrial-grade counterparts.

CHAPTER 5

RESULT AND DISCUSSIONS

1.1 INFERENCES

1.1.1 Bluetooth Server Inferences

1. Bluetooth despite being a low-powered, low data-rate, short-range communication standard, can be reliably worked with to scale under open-source IDE with good API support.
2. A considerable delay is introduced in the end-to-end transmission through the server, despite the single-hop nature owing to intermediate device and service discovery processes coupled with reduced uptime on device discovery.
3. The OBEX communication protocol can be used with ease to push files from one device to another and thus can be used to possibly convert any Bluetooth device into a mediatory server provided the API support is there.
4. As devices go out of communication range, an extension is possible without the need for a vertical handover to other standards as popularly believed and this can be achieved at a lower cost than industrial alternatives.
5. The Bluetooth protocol has a physical layer restriction on the data rate, especially with increase in distance, thus making the standard suitable only for the exchange of smaller files in situations such as this where an extension of range is targeted though single-hop networking.
6. While the OBEX communication protocol allows for easy file transfer, it does not have a provision for adding a layer of security or authentication to the transmissions at a packet-level and rather it can only be done by altering the contents of the files being transmitted which requires to be the files of a readable format.
7. The Bluetooth server can be easily throttled or choked when multiple incoming connections are requested.

1.1.2 RSSI Distance Estimation Inferences

1. RSSI distance estimation is a theoretical approach based on a proposed relation that is not too reliable when used indoors for mobile low-powered networks as the estimated values tend to vary from the actual due to physical obtrusions.
2. RSSI distance estimation method is highly influenced by external factors affecting low powered RF communications such as Bluetooth as the distance estimated is based largely values obtained during Line-of-sight communication under ideal EM environment. When devices are not moved but objects are placed disrupting Line-of-Sight communication, the RSSI values exhibit large fluctuations yielding garbage and undesirable values.
3. RSSI distance estimation yields more accurate results when averaged over multiple measurements.
4. RSSI-estimated distance values exhibit lesser deflection when the reference RSSI value is calibrated by taking into account the initial readings between devices at fixed distances.

CHAPTER 6

CONCLUSION

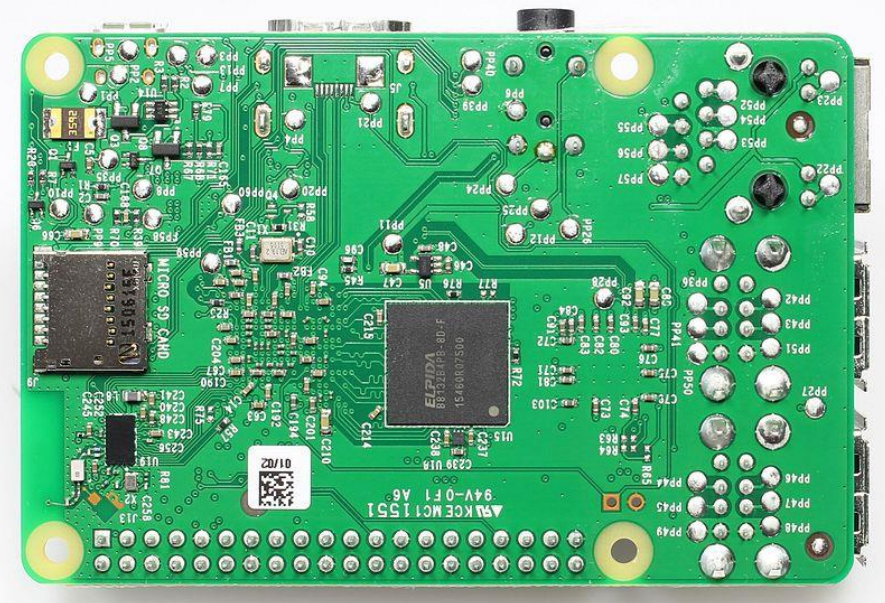
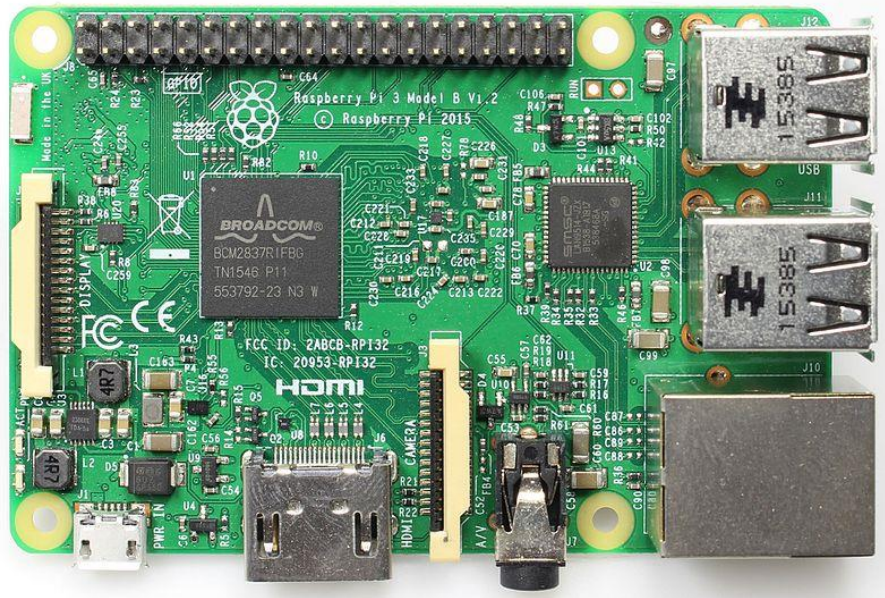
6.1. Project Summary

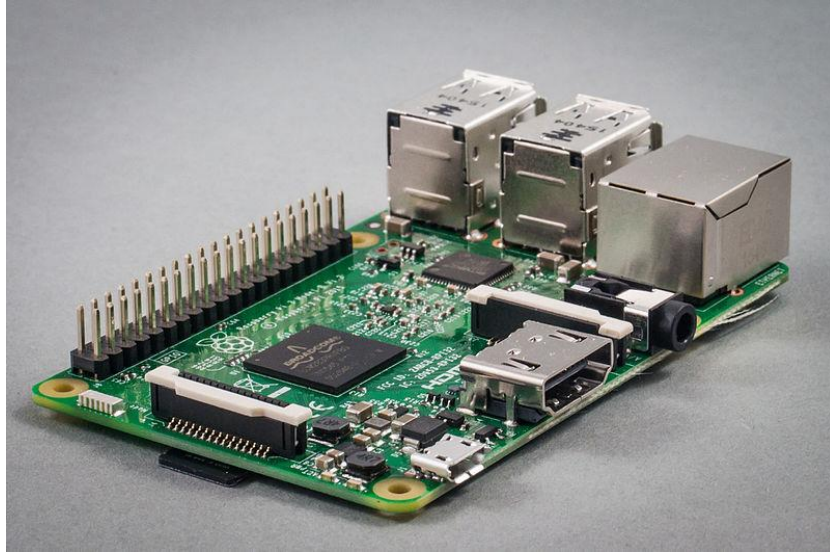
The aim of this project was to extend the range of Bluetooth communication between Class 2 Bluetooth-enabled devices by designing a server device capable of mediating Bluetooth communication between the two said devices when they are out of their normal range of communication but are within the server's range. The server was designed to accept incoming files from paired devices and then process them to identify an intended target device, verify their identities by reading and matching their addresses from the file and then forward the file to the destination when it becomes available. The server was also designed to be capable of estimating the distance between the originating device and the destination through RSSI-tracking. The range extension was successfully achieved through the implementation of the designed Bluetooth server and the end to end flow of data has been well documented. This approach of aiming to replace industrial-grade repeaters with a lower-cost prototype developed using an open source hardware IDE can be replicated for other IoT standards as well with similar pitfalls as Bluetooth, albeit with limited success. Our prototype's capability to estimate the distance however was greatly hindered by the inherent drawbacks of RSSI localization given our work was done largely indoors in a lab environment. Although, the main objective of our project was fulfilled in the sense that we were able to achieve end to end communication entirely using only the Bluetooth standard, our inability to perfect the RSSI algorithm to achieve perfect results was a bit disappointing. Nevertheless, this project can also be considered as a proof of concept for further approaches towards range extension in wireless personal area networking.

CHAPTER 7

APPENDIX

Appendix 1: Raspberry Pi 3 B Layout & Reduced Schematics





Appendix 2: OBEX Push Socket Script for Incoming Connections

```
pi@raspberrypi: ~  
GNU nano 2.2.6 File: /etc/systemd/system/obexpush.service  
[Unit]  
Description=OBEX Push service  
After=bluetooth.service  
Requires=bluetooth.service  
  
[Service]  
ExecStart=/usr/bin/obexpusd -B -o /home/pi/Bluetooth -n  
  
[Install]  
WantedBy=multi-user.target
```

```
Service Name: OBEX Object Push  
Service Description: a free OBEX server  
Service Provider: obexpusd  
Service RecHandle: 0x10005  
Service Class ID List:  
  "OBEX Object Push" (0x1105)  
Protocol Descriptor List:  
  "L2CAP" (0x0100)  
  "RFCOMM" (0x0003)  
    Channel: 9  
  "OBEX" (0x0008)  
Profile Descriptor List:  
  "OBEX Object Push" (0x1105)  
    Version: 0x0100
```


Appendix 3: Python Script used to undertake Directory Monitoring, Source-Destination Identification, Forwarding file to Destination, RSSI Distance Estimation & Logging of metadata

```
import glob
import os
import bluetooth
import sys
import time
from bluetooth import *
from PyOBEX.client import Client
from bt_proximity import BluetoothRSSI

log_file=open("BToothServer.log","a+")
sys.stdout=log_file
distance=0
direct=glob.glob('./Bluetooth/*.txt')
if len(direct)!=0:
    for f in direct:
        file=open(f,"r")
        source=file.readline(17)
        file.readline(1)
        print("New file received from source %s " % (source))
        print os.path.split(f)[1]
        BT_ADDR = source
        num=8
        btrssi = BluetoothRSSI(addr=BT_ADDR)
        print("Estimating source proximity using RSSI...")
        rssival=0
        for i in range(0, num):
            rssival+=btrssi.get_rssi()
            time.sleep(1)
        distance+=rssival/8
        print ("The source was approximately %d feet from the server" %
abs(rssival/8))
        destination=file.readline(17)
        file.readline(1)
        print("File intended for destination %s " % (destination))
        contents=file.read()
        print ("Initiating search for destination device...")

nearby_devices=bluetooth.discover_devices(flush_cache=True,lookup_names
=True)
print("Number of devices found: %d " % len(nearby_devices))
for addr, name in nearby_devices:
    print("%s - %s " % (addr, name))
flag=0
for addr, name in nearby_devices:
    if destination==addr:
        device_name=bluetooth.lookup_name(addr)
        print("Device with destination ID found")
        print("Searching for OBEX service on %s - %s" % (addr,
device_name))
```



```

        service_matches = find_service(name="OBEX Object Push",
address=addr)
        if len(service_matches)==0:
            print("Couldn't find the service. Device
doesn't support file transfer. File will be deleted to avoid security
risks.")
            file.close()
            os.remove(f)
            break
        first_match=service_matches[0]
        port=first_match["port"]
        name=first_match["name"]
        host=first_match["host"]
        print("Connecting to \"%s\" on %s" % (name,
device_name))
        client=Client(host,port)
        client.connect()
        client.put(os.path.split(f)[1], contents)
        client.disconnect()
        print("File successfully sent to destination and has
been deleted from server.")
        file.close()
        os.remove(f)
        BT_ADDR = destination
        num=8
        btrssi = BluetoothRSSI(addr=BT_ADDR)
        print("Estimating destination proximity using RSSI...")
        rssival=0
        for i in range(0, num):
            rssival+=btrssi.get_rssi()
            time.sleep(1)
        distance+=rssival/8
        print ("The destination was approximately %d feet from
the server" % abs(rssival/8))
        print ("Estimated distance between source and
destination is %d feet +/-%f" % (abs(distance), abs(distance*0.25)))
        flag=0
        break
    else:
        flag=1
    if flag!=0:
        if len(nearby_devices)!=0:
            print("None of the found devices match specified
destination ID")
        if len(nearby_devices)==0:
            print("Device with destination ID not found")

print("\n\n\n\n\n")
log_file.close()
sys.stdout=sys.__stdout__

```

Appendix 4: PHP Script used to establish range of communication in Class 2 Bluetooth devices through RSSI estimation

```
<html>
<head><title>BLUETOOTH DISTANCE</title><meta http-equiv="refresh"
content="10"></head>
<body>
<?php
echo "<h2><b><center>BLUETOOTH DISTANCE
ESTIMATION</center></b></h2><br/><br/>";
$filename = "log.txt";
$name = file_get_contents($filename);
$scal=0; //calibration counter1 <60
$scal1=0; //calibration counter2 >60
$scalib=0; //calibration variable
$tx=0; //calibration flag
for ($i=0;$i<strlen($name);$i++)
{
    $dist = 0; //RSSI variable
    if($name[$i]=="-" && $name[$i-1]=="")
    {
        $distcalc=0;
        $i++;
        for($j=1;$j<=2;$j++)
        {
            $dist = ($dist*10) + $name[$i];
            $i++;
        }
        if($scal>=3||$scal1>=3)
        {

            $filename = "Tx.txt";
            $scalib2=file_get_contents($filename);
            if($scalib2<=55)
                $distcalc = pow(10,($dist-59)/20);
            else
                $distcalc = pow(10,($dist-$scalib2)/20);
            $distcalcf = (int)$distcalc;
            if($dist<85)
            {
                if($distcalcf==0)
                    echo "Received Signal Strength: -",$dist,"dB
Distance: ",$distcalcf,"-",$distcalcf+1," meters approximately.";
                elseif($distcalcf<10)
                    echo "Received Signal Strength: -",$dist,"dB
Distance: ",$distcalcf-1,"-",$distcalcf+1," meters approximately.";
                elseif($distcalcf>=10 && $distcalcf<20)
                    echo "Received Signal Strength: -",$dist,"dB
Distance: ",$distcalcf-2,"-",$distcalcf+2," meters approximately.";
                elseif($distcalcf>=20)
                    echo "Received Signal Strength: -",$dist,"dB
Distance: ",$distcalcf-5,"-",$distcalcf+5," meters approximately. Device
going out of communication range. ";
            }
            else
                echo "Received Signal Strength is -85dB or lower.
Device has gone out of communication range.";
        }
    }
}
```

```

    }
    elseif($dist<=60 && $cal<3)
    {
        $scalib=$scalib+$dist;
        $cal++;
        echo "Calibrating...";
    }
    elseif($dist>60 && $call<3)
    {
        $call++;
        echo "Calibrating...";
    }
    if($cal==3 || $call==3)
    {
        if($tx==0)
        {
            if($cal==3)
            {
                $scalib1=($scalib/3);
                $scalib2=(int)$scalib1;
                $filename="Tx.txt";
                file_put_contents($filename, $scalib2);
                echo "<br/>Calibration complete,
Transmission power (Tx) calculated to be roughly -",$scalib2,"dB. ";
            }
            else
            {
                $filename="Tx.txt";
                file_put_contents($filename, "60");
                echo "<br/>Calibration complete,
Transmission power (Tx) calculated to be roughly -60dB. ";
            }
            $tx++;
        }
    }
}

elseif($name[$i]=="n" && $name[$i+1]=="/" && $name [$i+2]=="a" &&
$name[$i-1]==",")
{
    $i=$i+3;
    echo "Received Signal Strength is -85dB or lower. Device has
gone out of communication range.";
}
else
{
    if ($name[$i]=="\n")
        echo "<br/>";
    elseif ($name[$i]==",")
        continue;
    else
        echo "$name[$i]";
}
}
?>

</body>
</html>

```

CHAPTER 8

REFERENCES

1. Qian Dong, Waltenegus Dargie, “Evaluation of the Reliability of RSSI for Indoor Localization”, Technical University of Dresden, 2012.
2. Erin-Ee-Lin Lau, Boon-Giin Lee, Seung-Chul Lee, Wan-Young Chung, “Enhanced RSSI-based High Accuracy Real-Time User Location Tracking System for Indoor and Outdoor Environments”, International Journal on Smart Sensing and Intelligent Systems, Vol. 1 Issue 2, June 2008.
3. Jan Magne Tjensvold, “Comparison of the IEEE 802.11, 802.15.1, 802.15.4, 802.15.6 wireless standards”, September 2007.
4. Tomi Heinonen, Timo M. Laitien, Jarkko Lempio, “Apparatus, Method and System for a Bluetooth Repeater”, U.S. Patent 6 968 153, November 22, 2005.
5. Joseph M. Cannon, James A. Johanson, Philip D. Mooney, “Cordless RF Range Extension for Wireless Piconets”, U.S. Patent 6 650 871, November 18, 2003.
6. Yeab Wondimu, “Bluetooth Range Extender for Audio”,
<https://scholarship.tricolib.brynmawr.edu/handle/10066/14265>
7. Alexandros Schillings, “Bluetooth LE Library”, <https://github.com/alt236/Bluetooth-LE-Library---Android>
8. Albert Huang, “Bluetooth Programming with PyBluez”,
<https://people.csail.mit.edu/albert/bluez-intro/c212.html>