Development Documentation

Transmission and Visualization of Carotid Auscultation signals in an Android Application

Zeynep Ece Ergin, Jahnavi Thimmaiah Cheyyanda

Raspberry Pi Setup

Preparing SD Card:

1. Download a Raspberry Pi operating system. We installed NOOBS which is a easy easy installer for Raspbian. Download the OS here:

https://www.raspberrypi.org/downloads/

2. Unzip the downloaded file.

3. Download and install Etcher software. Etcher is a graphical SD card writing tool that works on Mac OS, Linux and Windows, and is the easiest option.

https://etcher.io/

4. Plug your SD card into your PC.

5. Open Etcher and select from your hard drive the Raspberry Pi image file you wish to write to the SD card.

6. Select the SD card you wish to write your image to.

7. Review your selections and click 'Flash!' to begin writing data to the SD card.

8. After the flashing is done, insert SD card in Raspberry Pi and power up the Raspberry Pi. Raspberry Pi will take some time to boot and prepare the SD card automatically.

The following libraries need to be installed:

- Update the packages on your Raspbian
  - sudo apt-get update

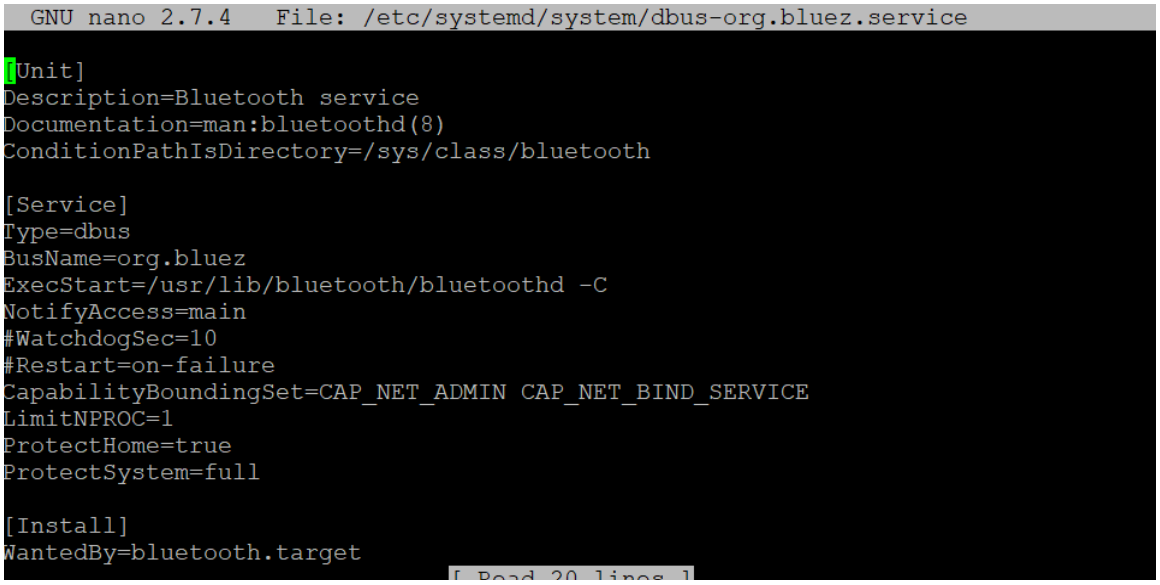
- Install pip
  - sudo apt-get install python-pip

- Pybluez requires two additional software packages. These are the Python Development Library and the Bluetooth Development Library.
  - sudo apt-get install python-dev libbluetooth-dev

- Pybluez:
  - sudo pip install pybluez

In addition, for the Python Bluetooth service to work, we will need to load the Serial Port Profile. This in turn requires the Bluetooth Daemon to run in compatibility mode. Start by editing the service startup parameters in its configuration file:

- sudo nano /etc/systemd/system/dbus-org.bluez.service
- Just add a -C after bluetoothd. The line should look like below:
- ExecStart=/usr/lib/bluetooth/bluetoothd -C



```
  GNU nano 2.7.4   File: /etc/systemd/system/dbus-org.bluez.service

[Unit]
Description=Bluetooth service
Documentation=man:bluetoothd(8)
ConditionPathIsDirectory=/sys/class/bluetooth

[Service]
Type=dbus
BusName=org.bluez
ExecStart=/usr/lib/bluetooth/bluetoothd -C
NotifyAccess=main
#WatchdogSec=10
#Restart=on-failure
CapabilityBoundingSet=CAP_NET_ADMIN CAP_NET_BIND_SERVICE
LimitNPROC=1
ProtectHome=true
ProtectSystem=full

[Install]
WantedBy=bluetooth.target
                              [ Read 20 lines ]
```

Now reload the service configuration and restart the Bluetooth service with the following two commands

  - sudo systemctl daemon-reload
  - sudo systemctl restart dbus-org.bluez.service

Finally, you can now load the Serial Port profile by issuing:

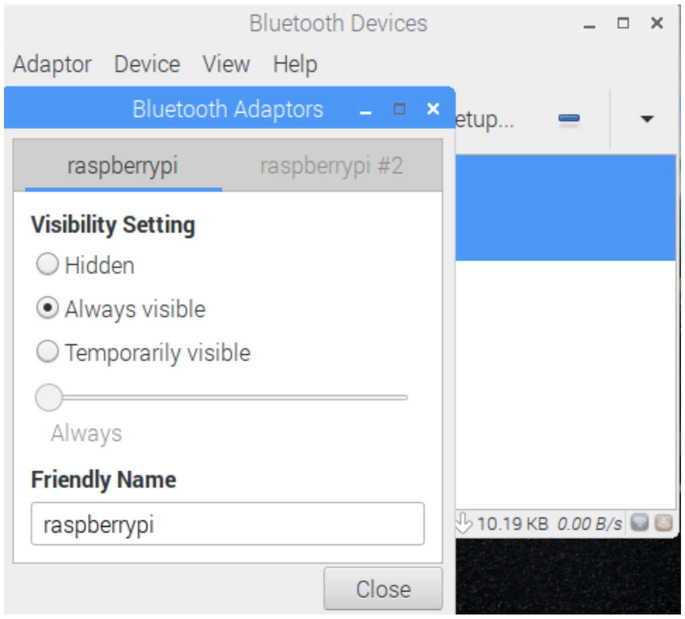
  - sudo sdptool add SP

You should get a Serial Port service registered message as a result, telling you that the operation was successful.

- Pyaudio:
  - sudo apt-get install python-pyaudio
- Blueman: Graphical Interface for Bluetooth
  - sudo apt-get install Blueman
- Reboot
  - sudo reboot

Once the installation is complete, launch the Bluetooth Manager from your applications launcher. Menu> Preferences > Bluetooth Manager. By default, your Bluetooth device is hidden. You need to make it visible so that it can be discovered by other devices.

Go to "Adaptor -> Preferences" and change the visibility setting to "Always visible". Also set a name for your Bluetooth device e.g. raspberrypi.



Visibility Setting
- Hidden
- [✓ Always visible]
- Temporarily visible

Always

Friendly Name: raspberrypi

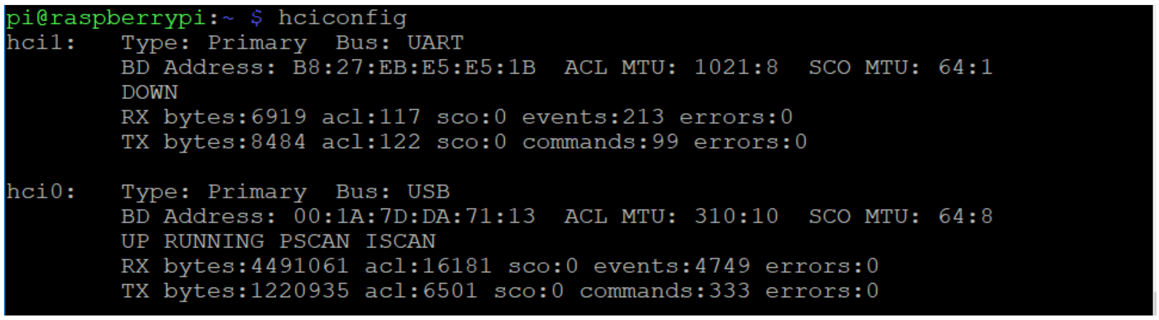
10.19 KB 0.00 B/s

Use the raspi-config utility to change few settings as below:

- Open the utility
  - sudo raspi-config
- Boot options - Desktop/CLI - Desktop Autologin
- Localisation options – Change timezone - Europe – Berlin
- Localisation options – Set Locale – Default (en_GB.UTF-8)
- If the time is still not correct update it manually
  - sudo date -s "01/04/2019 11:00"
- Interfacing options- Enable SSH, VNC, I2C

Configure Bluetooth devices:

- hciconfig



```
pi@raspberrypi:~ $ hciconfig
hci1:   Type: Primary  Bus: UART
        BD Address: B8:27:EB:E5:E5:1B  ACL MTU: 1021:8  SCO MTU: 64:1
        DOWN
        RX bytes:6919 acl:117 sco:0 events:213 errors:0
        TX bytes:8484 acl:122 sco:0 commands:99 errors:0

hci0:   Type: Primary  Bus: USB
        BD Address: 00:1A:7D:DA:71:13  ACL MTU: 310:10  SCO MTU: 64:8
        UP RUNNING PSCAN ISCAN
        RX bytes:4491061 acl:16181 sco:0 events:4749 errors:0
        TX bytes:1220935 acl:6501 sco:0 commands:333 errors:0
```

- Open and initialize HCI device
  - sudo hciconfig hci0 up

- Choose to Disable/Enable simple pairing

  Disable simple pairing and connect to any device (Not secure)

  - sudo hciconfig hci0 sspmode 0

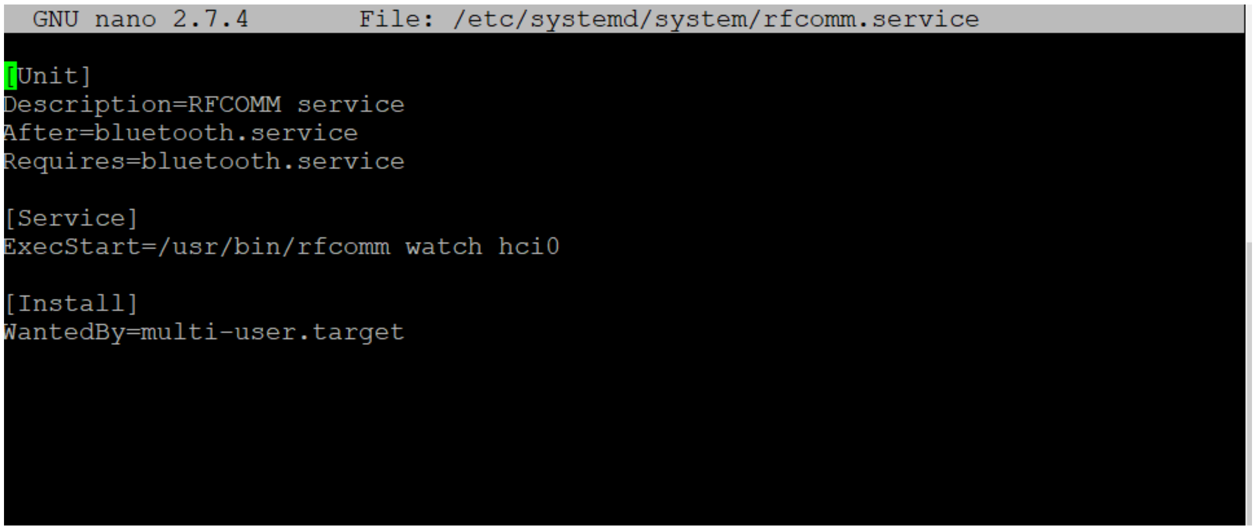
  Enable simple pairing and connect to any device (You have to connect the Pi to a monitor to accept pairing)

  - sudo hciconfig hci0 sspmode 1

- Enable Page and Inquiry scan
    - sudo hciconfig hci0 piscan
- Enable rfcomm during boot
    - sudo nano /etc/systemd/system/rfcomm.service

- Add the following in the file if you want to use hci0

```
1. [Unit]
2. Description=RFCOMM service
3. After=bluetooth.service
4. Requires=bluetooth.service


5. [Service]
6. ExecStart=/usr/bin/rfcomm watch hci0


7. [Install]
8. WantedBy=multi-user.target
```



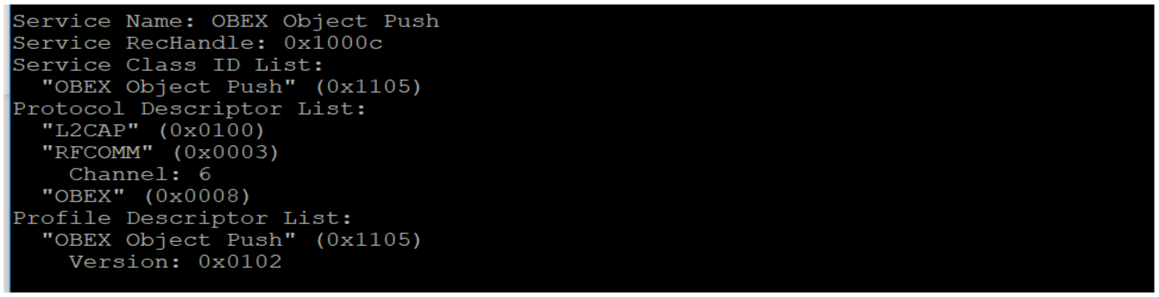
```
  GNU nano 2.7.4        File: /etc/systemd/system/rfcomm.service

[Unit]
Description=RFCOMM service
After=bluetooth.service
Requires=bluetooth.service

[Service]
ExecStart=/usr/bin/rfcomm watch hci0

[Install]
WantedBy=multi-user.target
```

    - sudo systemctl enable rfcomm
    - sudo systemctl start rfcomm

- As for saving the rfcomm0 data to a file
    - cat /dev/rfcomm0 > mytext.txt

- sudo cp /lib/systemd/system/bluetooth.service /etc/systemd/system/

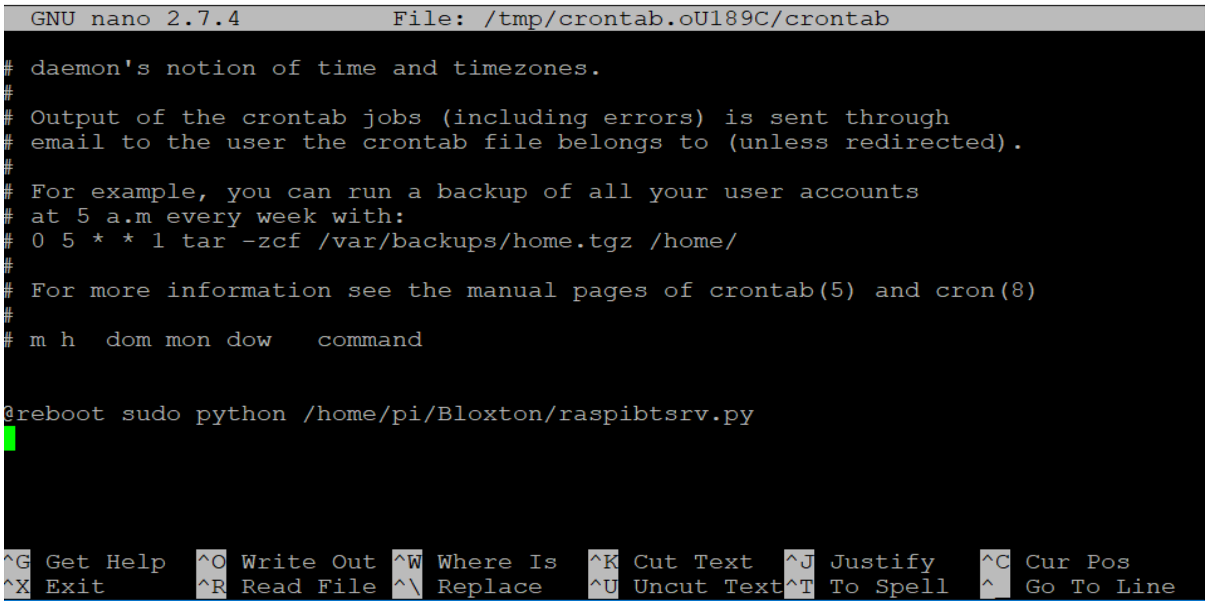
- To check if Obex Object Push service is available on your phone
  - sudo sdptool browse 40:4E:36:AC:9C:0E (MAC address of your phone)



```
Service Name: OBEX Object Push
Service RecHandle: 0x1000c
Service Class ID List:
  "OBEX Object Push" (0x1105)
Protocol Descriptor List:
  "L2CAP" (0x0100)
  "RFCOMM" (0x0003)
    Channel: 6
  "OBEX" (0x0008)
Profile Descriptor List:
  "OBEX Object Push" (0x1105)
    Version: 0x0102
```

- Obex FTP
  - sudo apt-get install python-obexftp

- AutoStart:
- sudo crontab –e
- @reboot sudo python /home/pi/Bloxton/raspibtsrv.py (path-to-your-python-script)



```
  GNU nano 2.7.4            File: /tmp/crontab.oU189C/crontab

# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command


@reboot sudo python /home/pi/Bloxton/raspibtsrv.py




^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit       ^R Read File  ^\ Replace    ^U Uncut Text^T To Spell    ^_ Go To Line
```

Server Application on Raspberrypi:

To accept the connections coming from android phone we are creating a bluetooth server application on the raspberrypi. The python code can be found here

https://github.com/zecergin/HealthTracker/blob/master/raspibtsrv.py

PyBluez library currently supports two types of BluetoothSocket objects: RFCOMM and L2CAP. We create a Bluetooth socket using the RFCOM protocol. The RFCOMM socket is created by passing RFCOMM as an argument to the BluetoothSocket constructor. An UUID is advertised so any client knowing the UUID may connect. When the socket receives a command from the client, the audio file is transferred using Obexftp Service.

The Android Application

The android app was developed in Android studio using Java programming language.

Pre-requisites and versions we used:

- Android Studio Latest version: 3.4.1 for Windows 64-bit
- Android SDK: 26.1.1
- Android versions: Android 5.0 (Lollipop)/API level 21 - Android 9 (Pie)/API level 28

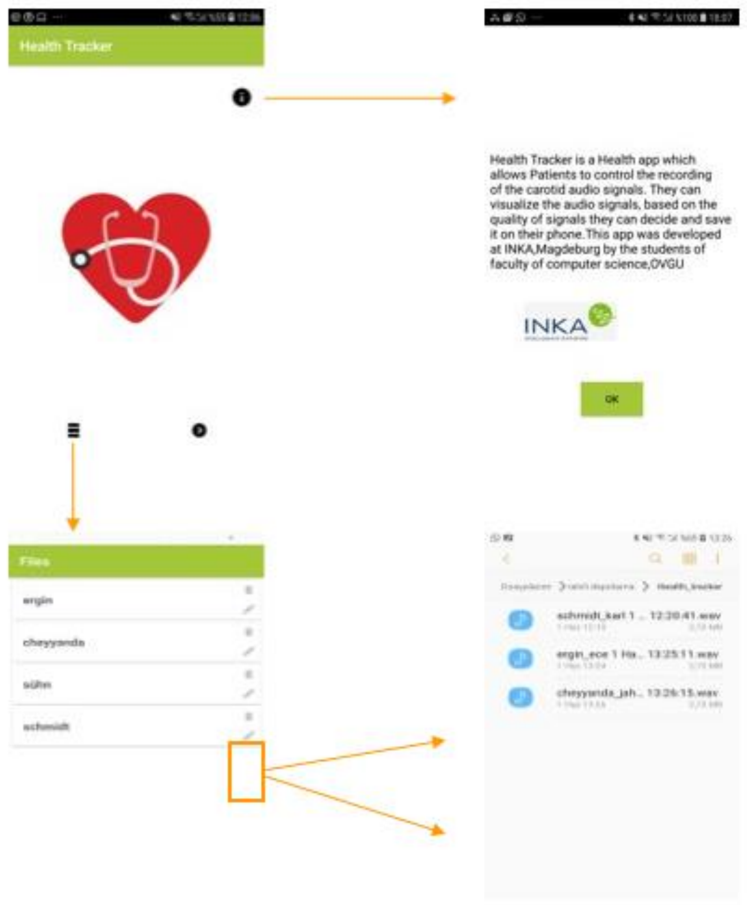
Implementation details of the App can be found in the below link:
https://github.com/zecergin/HealthTracker/tree/master

Bluetooth Communication Interface:

BroadcastReceiver is used to turn on the bluetooth and discover the nearby bluetooth devices.

In order to initiate a connection with the remote device a client socket is created in MainActivity.java using createRfcommSocketToServiceRecord(UUID). The control signal is sent by calling the thread's write() method and passing in the bytes to be sent. This method calls write(byte[]) to send the data to the remote device. The remote device on receiving the control signal sends the recorded audio file.

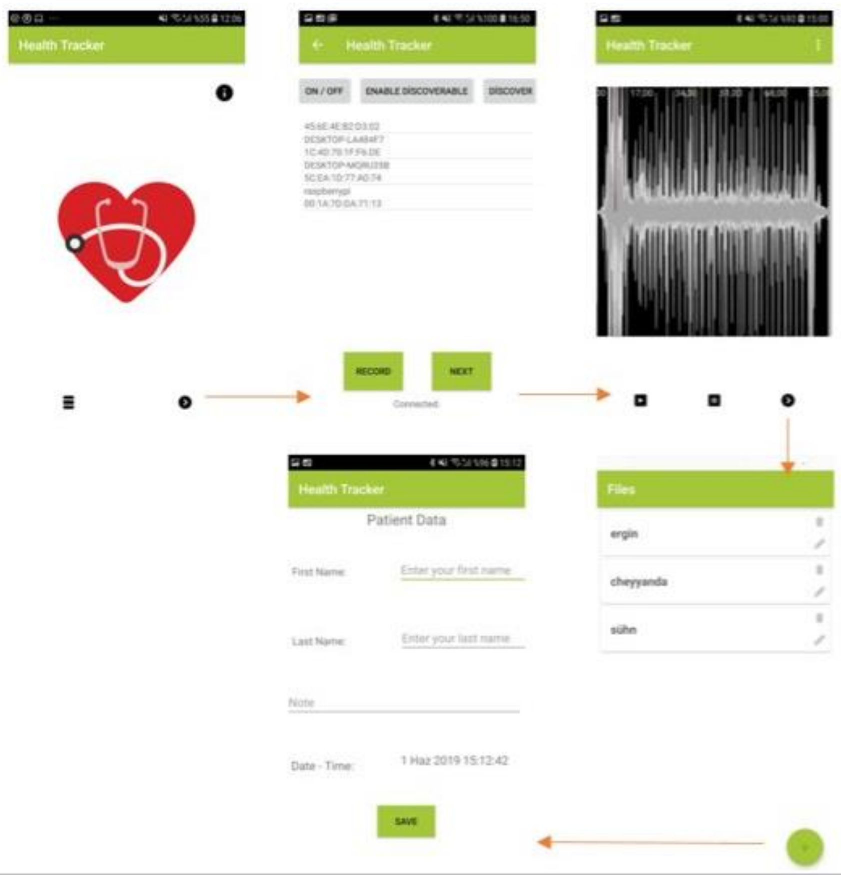
Workflow:



Health Tracker is a Health app which allows Patients to control the recording of the carotid audio signals. They can visualize the audio signals, based on the quality of signals they can decide and save it on their phone.This app was developed at INKA,Magdeburg by the students of faculty of computer science,OVGU

schmidt_karl 1 .. 12:20:41.wav
ergin_ece 1 Ha... 13:25:11.wav
cheyyanda_jah... 13:26:15.wav

The Main Page of the Android app has an information tab and a files tab where all the previous file information can be viewed.



45:6E:4E:82:D3:02
DESKTOP-LA484F7
1C:40:70:1F:F6:DE
DESKTOP-MQRU238
5C:EA:10:77:A0:74
raspberrypi
00:1A:7D:DA:71:13

Patient Data

First Name: Enter your first name

Last Name: Enter your last name

Note

Date - Time: 1 Haz 2019 15:12:42

ergin

cheyyanda

sühn

The Raspberrypi is discovered and a connection request is sent. The received audio file is visualized and saved with corresponding Patient Data.

Important links:

(1) https://projects.raspberrypi.org/en/projects/raspberry-pi-setting-up/3
(2) https://developer.android.com/guide/topics/connectivity/bluetooth
(3) https://developer.android.com/guide/topics/media/media-formats
(4) https://people.csail.mit.edu/rudolph/Teaching/Articles/BTBook-march.pdf
(5) https://github.com/newventuresoftware/WaveformControl
(6) https://developer.android.com/reference/android/arch/persistence/room/package-summary