ECSE 426 Microprocessor Systems

**Final Project Report**

IoT - Sensor Data Management from Hardware to Cloud

Group 06

Yan Ren 260580535

Dan Li 260564819

Yingnan Zhao 260563769

Fei Feng 260575129

23rd April, 2018

***Table of Content***

[**Abstract**](#_rjlgm3xlmbrt) **3**

[**Problem Statement**](#_nmgk8xzkrfr) **3**

[**Theory and Hypothesis**](#_cb83chy3bgj4) **4**

[Theory and Hypothesis behind tap detection](#_ptsc0j4r2no7) 4

[Theory behind audio file format transformation](#_bbdm6ikz19j1) 4

[Theory and Hypothesis of Signal processing in cloud](#_vfvsjw5gtp19) 6

[**Implementation**](#_njys0g4dl6w6) **7**

[Block Diagram](#_auzioxolanw) 7

[Discovery Board](#_qiirnxeeddf3) 8

[Audio data](#_3fkvsyocjyw3) 8

[Accelerometer](#_aem8f75s74n0) 8

[Tap Detection](#_raz83heig33) 9

[UART](#_otv1lzegk2vr) 10

[Nucleo Board](#_8gdgtpl9wodu) 11

[Connect to Discovery Board by UART](#_hf8746aytqj5) 11

[Transmit Data to Android Phone](#_8i9mkra3rdzk) 12

[Android Application](#_odvo2w8e5vdj) 12

[User Interface](#_mo2qgce0ey2x) 12

[System Features](#_8w84d3qw15y2) 13

[Firebase Cloud Service](#_565ksh9xfnrp) 14

[Firebase Cloud Storage configuration](#_4botla8wpik0) 14

[Cloud Storage and Cloud functions](#_uwgu12553o8n) 16

[Cloud Signal Processing: Speech API and Plotly](#_j071bo6iqd9d) 16

[Npm, Node.js, and Implementation](#_sicdmcate9nd) 16

[**Testing and Observation**](#_fkd2vyoocznt) **17**

[Testing before the final demonstration](#_n4msnm1l8sdi) 17

[Result during the final demonstration](#_qjvfd2z6l7bf) 20

[**Timeline and Division of Work**](#_37frcgw1cqbe) **24**

[**Conclusion**](#_w29zijclm8kn) **25**

[**References**](#_rjlgm3xlmbrt) **26**

#### Abstract

In this project, we build a system to explore the interaction of embedded peripherals and sensors with chould-enabled services, which is one of the main characteristics of Internet of Things (IoT) designs. The system allows the Discovery board to send audio and accelerometer data over BLE connection to smartphone device. These data then will be saved in a file and be uploaded to Firebase cloud service for processing. STM Discovery board is connected with smartphone by a Bluetooth Low Energy (BLE) connection. The BLE interface is realized through a STM32F401RE Nucleo board along with an IDB04A1 BLE daughter board. Discovery board is connected with Nucleo board by using serial link. BLE module is able to receive data from Discovery board and transmit data to smartphone. Then smartphone and cloud service do uploading and processing data files.

#### Problem Statement

The main goal of this project is to build the communication between Discovery board and Android phone via Bluetooth Low Energy which is supported by Nucleo board. Based on the main goals of the project, the objective is divided into following four major parts: STM Discovery board, Nucleo board, Android phone application, and Firebase Cloud service.

For Discovery board, there are four specific functions need to be accomplished. Firstly, a single tap and double tap detection algorithm needs to be implemented. Then under single tap state, it will use microphone to record audio data for one second. Otherwise under double tap state, it will record accelerometer data for ten seconds. Finally, it will transmit the recorded data to Nucleo board via serial link.

As for Nucleo board, it is not responsible for operating any measurement, but it plays a key role in the communication process. There are two main functions need to be implemented. The first one is to build the serial communication with Discovery board via UART communication. Therefore when data on Discovery board are ready, Nucleo board is able to receive data from Discovery board. The second is to provide BLE functionality to connect with Android phone. So when it finishes receiving data from Discovery board, it can send all data to Android phone.

For Android phone application, it is responsible for receiving and processing two types of data from the Nucleo board, and communicating with firebase. It should be able to differentiate two types of data and process the data accordingly then upload the data to the designated location on firebase storage. If the received data is the sound data, the data should be converted into WAV format and uploaded to Sound/track.wav file on firebase. If the received data is pitch and roll data, the data should be uploaded to pitch.txt and roll.txt files respectively on firebase. It should be able to display the progress of receiving data from Nucleo board and uploading data to firebase. It should also be capable of reading and displaying the output file on the firebase storage which stores the result of voice recognition.

Once the sound file of the recorded audio and two text files of pitch and roll data are uploaded to the Google Firebase Cloud Storage, the actions of speech recognition and plotting of the pitch and roll data versus time should be triggered. The problem is how to trigger the events and how to utilize third party speech recognition and plotting API. After the speech is converted to the text, the result should be saved in a text file located in the cloud storage bucket. For the plotted pitch and roll result, the result jpeg file should be saved back the the bucket. The mechanism should be as simple as possible since the time budget is limited for this project.

#### Theory and Hypothesis

##### Theory and Hypothesis behind tap detection

The accelerometer uses a system of weights and springs to determine acceleration by generating (usually) more voltage on a special crystal when the springs are more stressed. Generally, at steady position, the accelerometer displays a total of 1g downwards in either X, Y, Z axis or a combination of the axis. If subjected to a sudden displacement (tapped), the stress on the springs would change since the weights would want to stay where they are by law of inertia. Thus, tapping on the board would make the accelerometer change its value since there would be less acceleration downwards because the springs are less stressed because the weights wants to stay where they are. The idea behind deciding if a motion was a tap on the board or just a displacement of the board is the difference in acceleration. If tapped, the difference in acceleration would be drastic and if moved (not jerked towards a position), the change in acceleration would be mild. In this project, we are required to detect single tap and double taps by using accelerometer. We conducted the experiment to find the proper threshold for tap and threshold between taps. We will discuss more details in Implementation section.

##### Theory behind audio file format transformation

The audio data is sampled by a 8-bit ADC, at a frequency of 10,000 Hz. Thus the sample rate is 10 K and one byte per sample, and the audio data will be transmitted by the Nucleo board one byte at a time. To transform the raw data from ADC to wav format, we followed fig 3.2.1 below. We need to add several fields in before the data bytes, the value of each field we choose is shown in table 3.2.2 below.

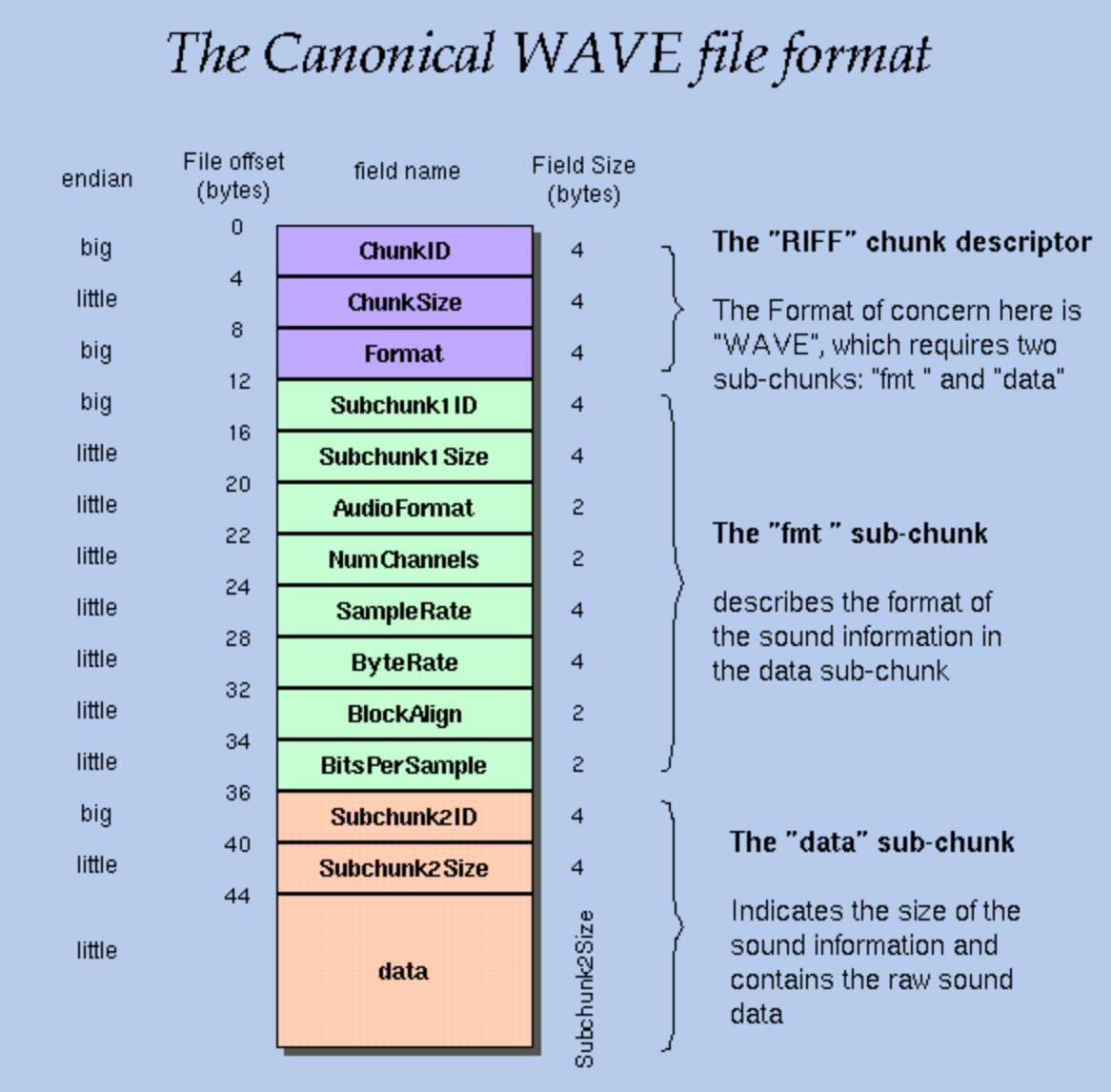


Fig 3.2.1 Standard WAV format [1]

|  |  |  |
| --- | --- | --- |
| offset | Field name | Field value |
| 0 | ChunkID | “RIFF” |
| 4 | ChunkSize | 36+data.length |
| 8 | Format | “WAVE” |
| 12 | Subchunk1ID | “Fmt ” |
| 16 | Subchunk1Size | 16 |
| 20 | AudioFormat | 1 |
| 22 | NumChannels | 1 |
| 24 | SampleRate | 10000 |
| 28 | ByteRate | 10000 |
| 32 | BlockAlign | 1 |
| 34 | BitPerSample | 8 |
| 36 | Subchunk2ID | “data” |
| 40 | Subchunk2Size | data.length |
| 44 | data | data |

Table 3.2.2 Wav Field Value Table

##### Theory and Hypothesis of Signal processing in cloud

The cloud function should be implemented as a Node.js Firebase project that is written in local environment and deployed to the Firebase cloud. To set up the development environment, npm should first to be installed locally, and Node.js, Firebase should then be installed using the npm command line tool. When developing, required npm package can be installed to provide certain functionalities and ease the development process. Testing should be done in the cloud and console printout should be verified in the cloud function console log window.

In the cloud function, Google Speech API and Plot.ly API need be implemented accordingly depending on the file type uploaded to the bucket to achieve speech recognition and plotting of data. Along the process, certain console printout should be implemented to keep track of the cloud process and make easier for the instructor to verify at the time of final demonstration.

The Google Speech API is assumed to be able to convert the speech we recorded to text, and Plot.ly is assumed to plot pitch and roll data versus time correctly.

#### Implementation

##### Block Diagram

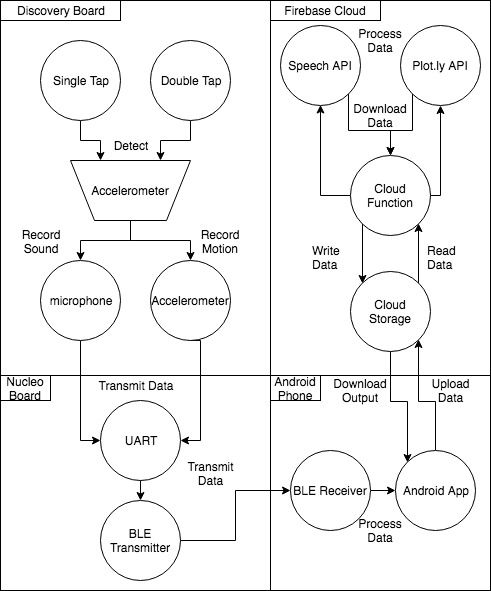


Figure 4.1.1. Block Diagram of the System

As shown in the Block diagram above (fig 4.1.1), the project can be divided into 4 parts. The discovery board is responsible of detecting taps and record audio data and accelerometer data, the Nucleo board serves as the BLE server and will send the data to the Android app, the Android app part will process the received data and upload to firebase server, and the firebase cloud service part will further process the data. Those 4 parts will be discussed in great detail in the sections below.

##### Discovery Board

The implemented features of discovery board contains audio data collection (through ADC), accelerometer, LEDs and UART. The main algorithm is designed as a finite state machine. When user starts the board, all four on board LEDs are on, indicating the board is in tap detection state. When single tap detected, green LED turns on, indicating recording, and after one second, green LED turns off and blue LED turns on, indicating sending audio data to Nucleo Board. After sending data finishes, board goes back to tap detection state. When a double tap detected, orange LED turns on, indicating reading accelerometer data, and after ten seconds, orange LED turns off and blue LED turns on, indicating sending audio data to Nucleo Board.

###### Audio data

Microphone for recording audio data is connected through ADC. We configure ADC as 8-bit resolution timer triggered. The timer2 is used to pull the ADC data and its frequency is 10kHz. Our recording time is 1 second. Therefore, we set a 10000 uint\_8 buffer for storing audio data. The collected data will be filtered by FIR before storing to the buffer. When the buffer is full, the audio data buffer will be sent to Nucleo Board through UART.

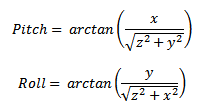
###### Accelerometer

The board uses the provided accelerometer conversion algorithm to determine the x, y, z value. We use following parameter to initialize the accelerometer:

|  |  |  |
| --- | --- | --- |
| Parameter | Value | Description |
| Axes\_Enable | LIS3DSH\_XYZ\_ENABLE | Direction\_XYZ\_selection |
| AA\_Filter\_BW | LIS3DSH\_AA\_BW\_50 | Antialiasing\_Filter\_BW |
| Full\_Scale | LIS3DSH\_FULLSCALE\_2 | Full\_Scale\_selection |
| Power\_Mode\_Output\_DataRate | LIS3DSH\_DATARATE\_100 | Data Rate 100Hz |
| Continous\_Update | LIS3DSH\_ContinousUpdate\_Disabled | Block or update Low/High registers of data until all data is read |

Table 4.2.2.1 Parameters of accelerometer

There are two parts in this project involving using accelerometer. The tap detection and pitch/roll capture. The data rate is set to be 100Hz. In the pitch/roll capture part, we record the data for 10 seconds. Therefore we have 1000 x, y and z values. Three FIR filters are used to filtered the x, y and z value. As part of project requirement, we calculate the pitch and roll by using following equation:



Since arctan returns value in radius, we convert them to degree and also add pitch and roll by 90 degree, so that all pitch and roll value will be positive. This conversion step makes the transmission step much easier. When we collect 1000 pitch and 1000 roll data, we send them to Nucleo Board through UART.

###### Tap Detection

Tap detection algorithm is based on the accelerometer readings. We keep tracking the z value and set the proper threshold as a tap detected. Since this project requires to detect tap and also distinguish single tap and double tap. We find there are two thresholds need to be properly adjusted. First threshold is the tap threshold. When z value exceed this value, a tap is detected. Second threshold is the tap duration threshold. After first tap detected, we delay a certain time to do the tap detection again. If user does not do the second tap in this duration, one tap is found. If second tap detected in this duration, two taps are found. Just as the pitch/roll tracking, we use FIR filter to filter the value before process the tap detection.

To find the proper thresholds, we conduct the experiments: we initialize accelerometer under 100 Hz data rate. We conduct single tap and multiple tap behaviour and record all data points. The following is one of the experiment results:

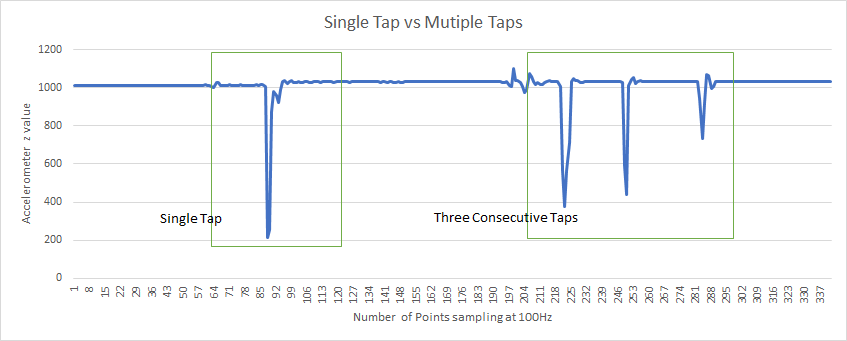


Figure 4.2.3.1 Tap behaviour data record

The diagram is showing a single tap vs consecutive taps. Y-axis is the z value, x-axis is the number of data points. Each data point is required every 0.01 second. After analysing massive experiment data, we find a tap will change the z value more than 200. Therefore in our tap detection, when z changes over 200, we say a tap detected. Also we find it usually takes around 0.4-0.5 second to finish a tap. Therefore, we set our tap detect duration to be 1.5 seconds, which means if user wants a double tap detection, the second need to be finished within 1.5 after first tap.

###### UART

Serial data transmission between discovery board and Nucleo board is achieved by UART. STM32 USART interface is the Universal Synchronous/Asynchronous Receiver/Transmitter Interface, which is widely used for serial communications. In our project, we use Asynchronous UART communication. We use following parameters to configure the UART5 on discovery board:

|  |  |  |
| --- | --- | --- |
| Parameter | Value | Description |
| BaudRate | 115200 | UART communication baud rate |
| WordLength | UART\_WORDLENGTH\_8B | Specifies the number of data bits transmitted or received in a frame |
| StopBits | UART\_STOPBITS\_1 | Specifies the number of stop bits transmitted |
| Mode | UART\_MODE\_TX\_RX | UART Transfer Mode |

Table 4.2.4.1 Parameters of UART5

UART is configured as TX\_RX mode. The sending function used in discovery board is HAL\_UART\_Transmit(UART\_HandleTypeDef \*huart, uint8\_t \*pData, uint16\_t Size, uint32\_t Timeout). Since we have two types of data buffer need to be transmitted: audio data buffer of length 10000; pitch/roll data buffer of length 2000. We define the transmission protocol to Nucleo board that: before sending audio data buffer as size 10000, send a one-byte data “0” to indicate the next sending data is audio data. For pitch/roll data, before sending pitch/roll data buffer as size 2000, send a one-byte data “1” to indicate the next sending data is pitch/roll data. So that the Nucleo Board will choose the proper data buffer for receiving once it reads our one byte signal.

##### Nucleo Board

STM32F401RE Nucleo board and IDB04A1 BLE daughter board are used in this project for building BlueNRG network. They are able to create Bluetooth Low Energy, which is a technology used to build a custom profile for specific application on Android phone. Nucleo board is connected to the Discovery board by serial link. In the connection between Nucleo board and Android phone, the phone plays the role of master which can scan for bluetooth signal. And Nucleo board plays role of slave which is waiting for connect signal and accepting incoming requests. In BLE network, the board can only be connected with one single phone. When the connection is complete, they will transmit data by using GATT (Generic Attribute Profile) Service. Generally, each profile application is a collection of services, and each service is built up by characteristics. Both characteristic and service are represented as attributes. Each attribute is defined by UUID, which is a 128 bits universally unique identifier with specific access permission.

###### Connect to Discovery Board by UART

As mentioned in section 4.1.4 above, serial data transmission between Discovery board and Nucleo board is achieved by UART. We use following parameters to configure the USART6 on Nucleo board:

|  |  |  |
| --- | --- | --- |
| Parameter | Value | Description |
| BaudRate | 115200 | UART communication baud rate |
| WordLength | UART\_WORDLENGTH\_8B | Specifies the number of data bits transmitted or received in a frame |
| StopBits | UART\_STOPBITS\_1 | Specifies the number of stop bits transmitted |
| Mode | UART\_MODE\_TX\_RX | UART Transfer Mode |

Table 4.3.1.1. Parameters of USART6

The receiving function for Nucleo board is HAL\_UART\_Receive(UART\_HandleTypeDef \*huart, uint8\_t \*pData, uint16\_t Size, uint32\_t Timeout). Since we have two types of data need to be received, which are 10000 length audio data or 2000 length pitch and roll data, we used two buffers with different size to receive the different length of data. The protocol for receive is that, when firstly receiving a one-byte data “0”, the program is going to use the 10000 size buffer to receive audio data. If the program firstly receiving a one-byte data “1”, then it will prepare the 2000 size buffer for receiving pitch and roll data. Both audio data and pitch/roll data are sent by an array, hence on Nucleo board, we also receive the data array at one time to avoid losing data.

###### Transmit Data to Android Phone

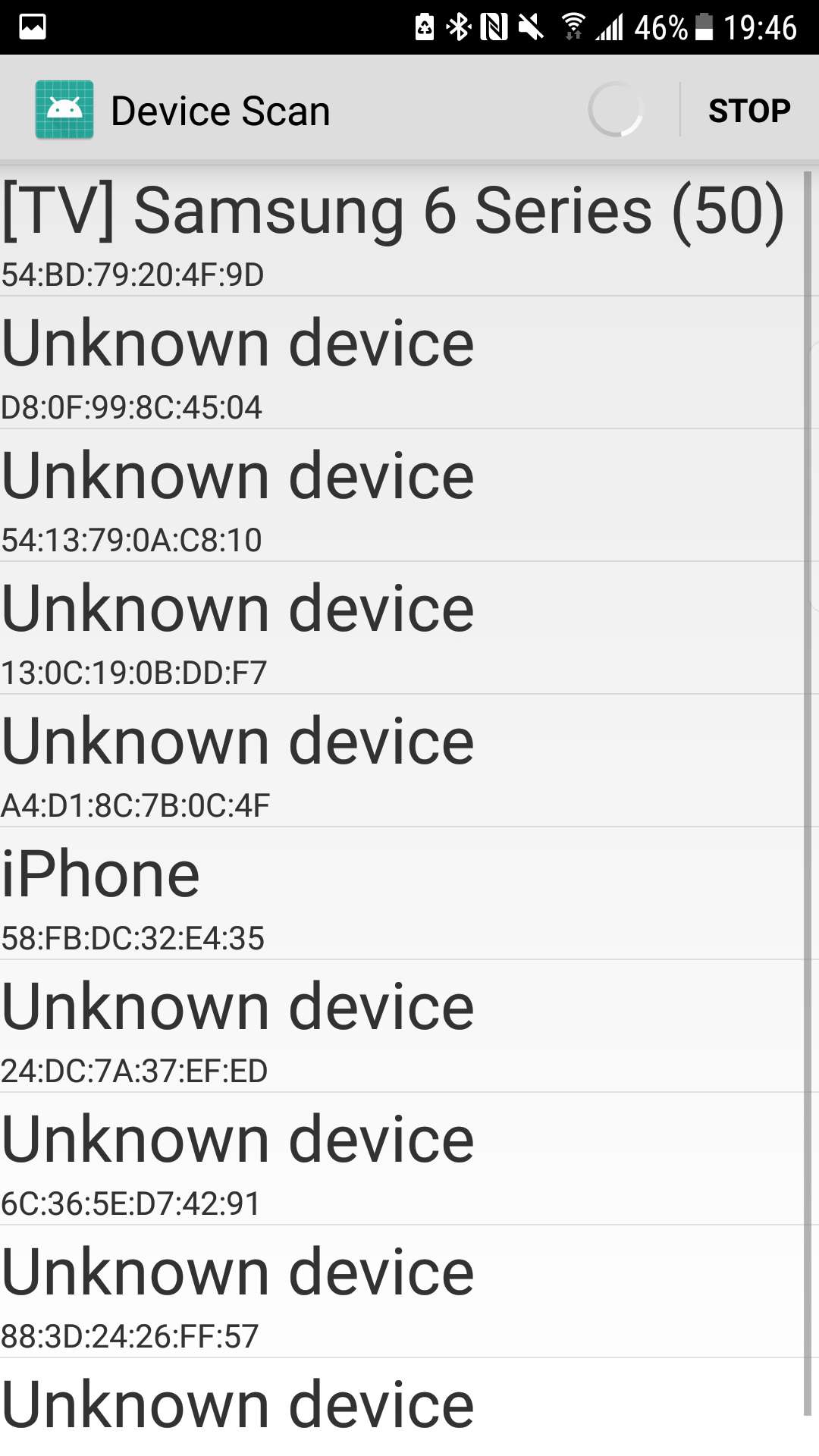
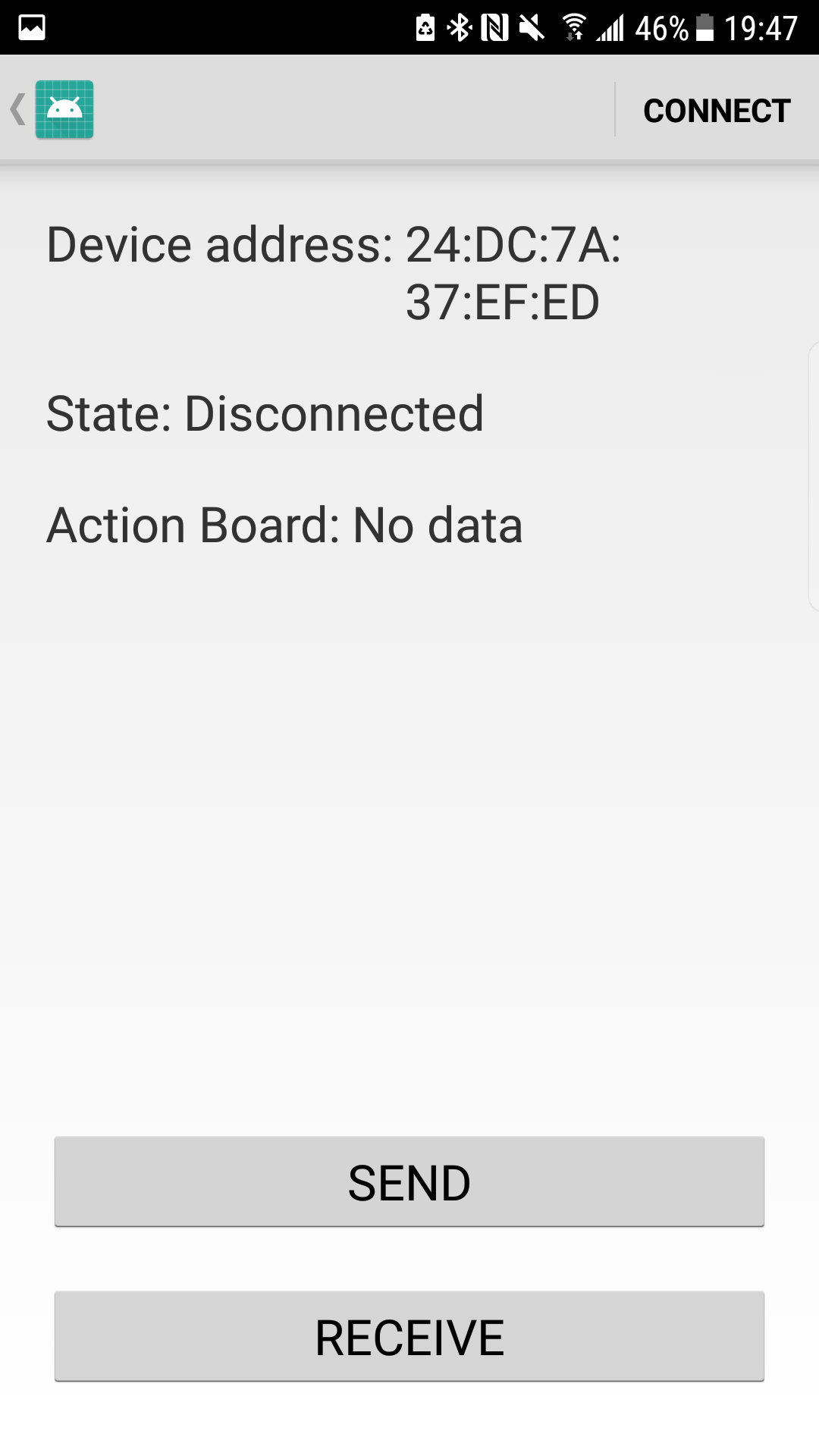
For sending the data to Android phone, we use GATT service function aci\_gatt\_update\_char\_value(Serv\_Handle, Char\_Handle, Var\_Offset, Char\_Value\_Length, Char\_Value). Similar to receiving data, when transmitting data to phone, we firstly send a one-byte data “0” or “1” to phone, which “0” represents 10000 audio data and “1” represents 2000 pitch and roll data. It tells the phone what the coming data are. After that we use a for loop to send all data in buffer array to the Android phone. Meanwhile, each time we send one data, there is a following HAL\_Delay(10), this is used to prevent losing data during the transmitting process.

##### Android Application

The app designed for this project is based on the android-BluetoothLeGatt example project [2] provided by google. To satisfy the project requirements, the android app has to be able to scan BLE device, connect with a BLE device, receive data from connected BLE device, process the received data (byte array to wav) and upload and download from firebase.

###### User Interface

For the user interface of this project, we need 2 different layouts, one for device scanning, the other one for receiving and sending the data. The device scanning layout won the left is directly from google example, it used a linear layout contains listView to display the device names and addresses. Once one of the items in the list has been clicked the name and address of that item will be recorded and the app will switch to the service layout shown on the right. And start scanning and stop scanning activity can be controlled by clicking the text on the top right corner.

Figure 4.4.1.1. Android App User Interface

For the service layout, we used linear layout, and there are 5 elements in total. First two are text views to display the device address and the connection state, once the state is connected, it will look for the changes in characteristics of the connected device, it will display the data change in the action board text area. And the send and receive buttons are for sending data and receiving data from the firebase server. And the result of the send data action and the received data from firebase will be also displayed in the action board. The device can be connected and disconnected by clicking the text on the top right corner.

###### System Features

The control activity class and the bluetooth le service class are responsible to achieve all system features. The UUIDs of all gatt attributes are stored in the GattAttribute class, the UUID stored are exactly the same as the ones in the code for Nucleo board. Device scan activity class is used to scan BLE devices and connect to BLE devices, this class is from the google and it is used alongside with the device scanning layout. It will populate the list in the device scanning layout once new device is found. Since our data is sent one byte at a time from the Nucleo board, so in the Bluetooth LeService class, we override the onCharastericChange function to receive and process the characteristic update. A custom broadcastUpdated function will be called, this function will first check the uuid of the characteristic with the gatt attribute class to identify the data transmitting characteristic. The byte value of that characteristic will be first transformed to a two-bit hex value and transmit this data using intent to the Control Activity class.

In the control activity class, there is a Broadcast receiver method, it is used to receive the data in the intent send by Bluetooth LeService class and process those data. This function will call the displaySoundData function to store the data into a public byte array and display the data in action board. Then there are two onclick listeners for the send and receive button respectively. For the send button, the length of the byte array used to store the data will be checked first. As for pitch and roll data the length is 2000 and for sound data the length is 10000. Then if the array stores the sound data, the wavGet method will be called, this function will transform a byte array stores the sound data into a byte array stores the wav version of that same sound data as discussed in section 3.2. Then a reference to a file in the firebase storage will be created and the putBytes method from firebase library will be used to send the data to the fire base storage. A file named Sound/track.wav will be created on firebase. If the received data is pitch and roll, the byte array will be split in the middle, and the value will be sent using the putBytes method and stored in the fireBase storage, two new files to store the integer value of the 1000 pitch values and roll values will be created named pitch.txt and roll.txt respectively. If the receive button is press, in the onclick handler, functions will be called to check the voice recognition output file on firebase storage, and the result will be displayed on screen. The firebase uploading and downloading functions are achieved by following the instruction from the official documentation of firebase [3].

##### Firebase Cloud Service

###### Firebase Cloud Storage configuration

To use Firebase cloud service, the first is to set up a project that is shared between the Android application and the cloud platform. To establish such link between them, the project ID of the Android project should be selected when establishing the cloud project.

As can be seen in figure 4.5.1.1., when creating the cloud project, the “ECSE426FinalProject”, which is the project ID of the already established Android project using the same google account, should be entered.

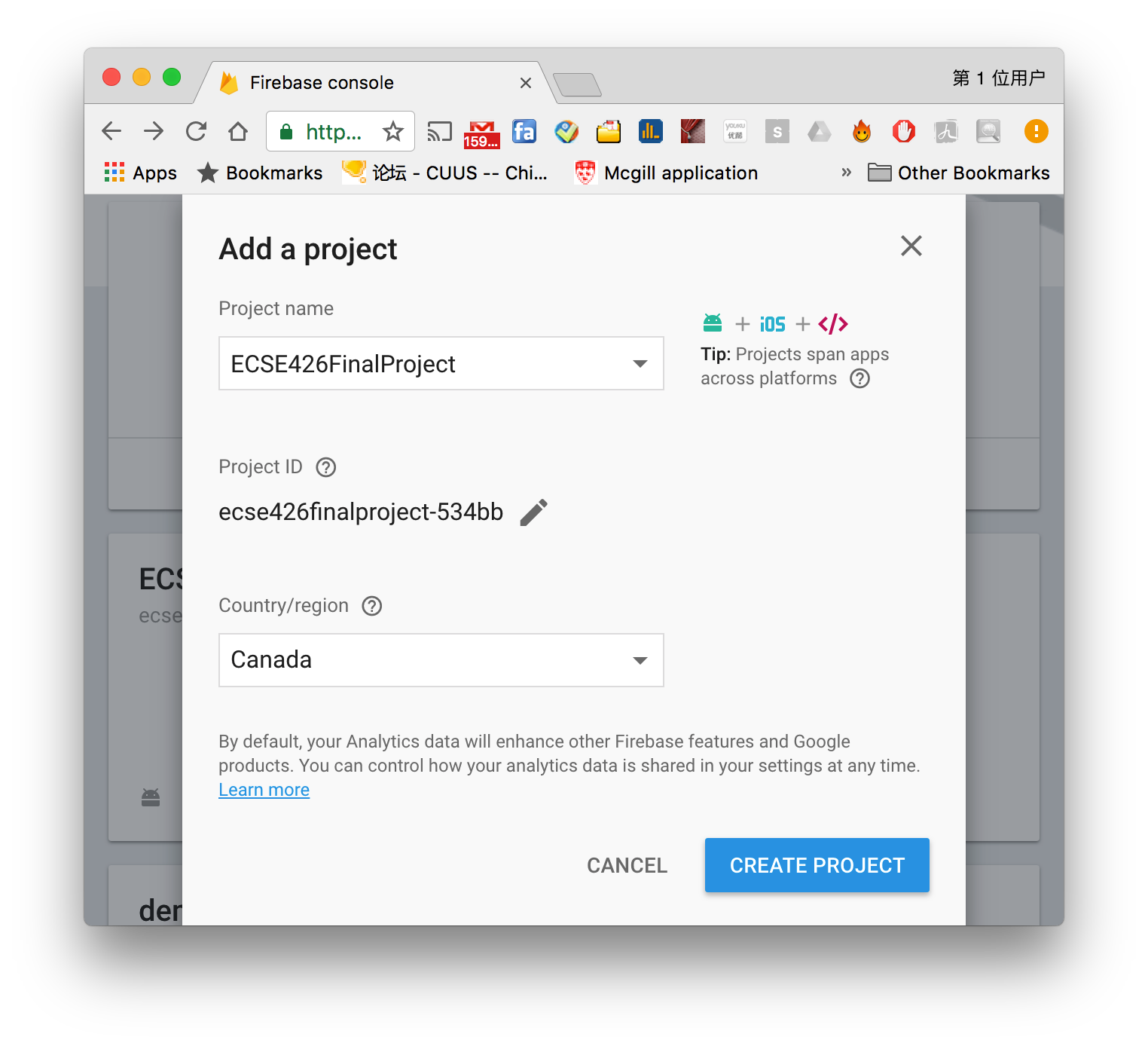


Figure 4.5.1.1. Create a firebase project

Once the cloud project is created, navigate to the cloud storage rules, and delete “if request.auth != null;” shaded in figure 4.5.1.2. to grant access to all the developers and users for faster development cycle.

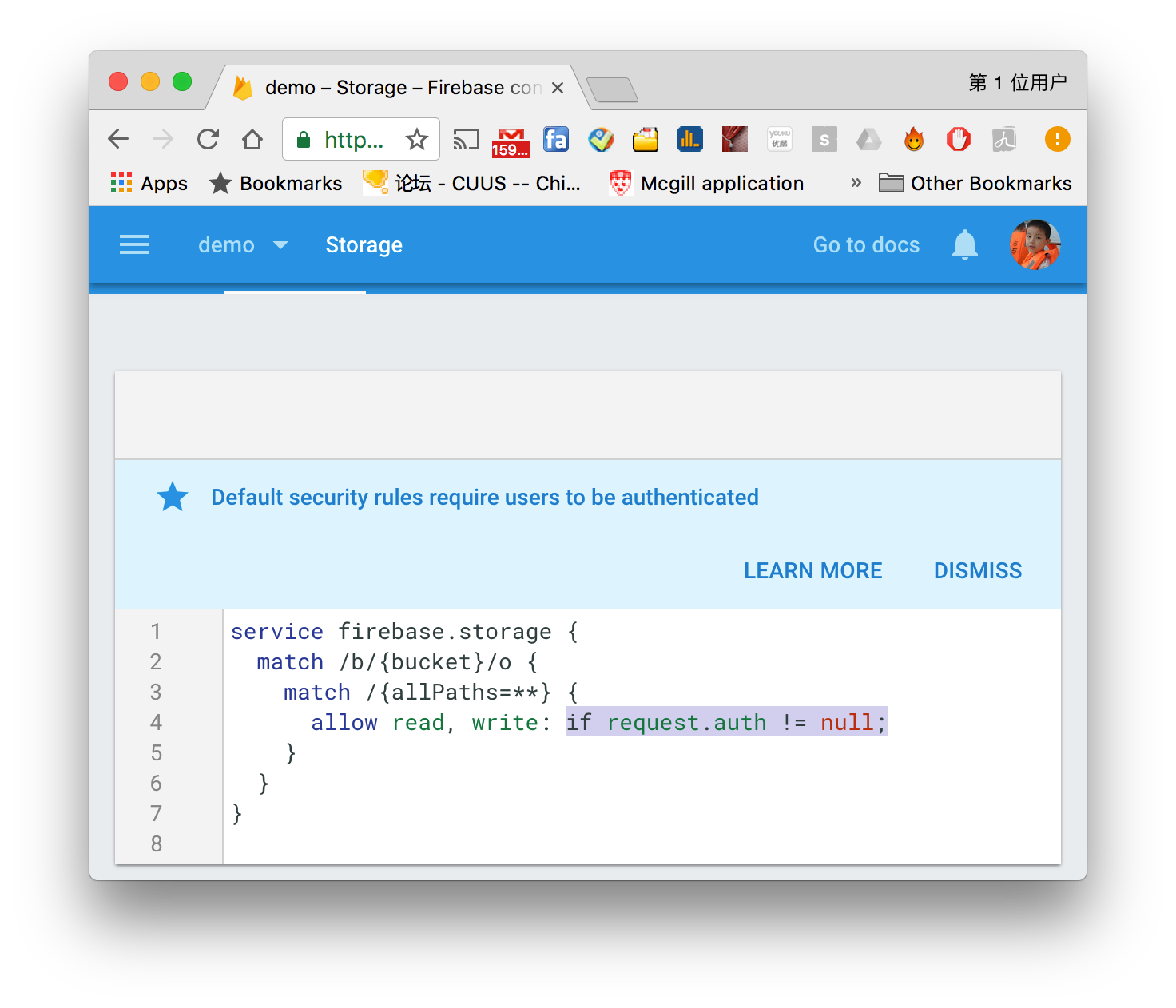


Figure 4.5.1.2. Modify the cloud storage authentication rules

Once the cloud storage authentication is set correctly, the android app should be able to upload files directly to the cloud storage bucket, whose uri should be in the form of “gs://ecse426finalproject.appspot.com”. Since the authentication is set to allow read and write operation for all the users, the android application should not require extra authentication to access the bucket.

###### Cloud Storage and Cloud functions

To detect, access and modify the file uploaded to the storage bucket by the android application, a cloud function residing in the cloud and triggered by new file uploaded to the bucket is needed. To set up and code a cloud function, node.js, a run-time environment for server-side JavaSript [5] should be installed on the development machine locally. The team chose to install npm, a package manager for Node.js [6] to take advantage of its convenience to install packaged modules of JavaScript code to easy the development process of the cloud function. After installing npm and Node.js, Firebase command-line tools was installed using npm by

“$ npm install -g firebase-tools” Once the the android application can upload file in to firebase storage, the processing of the uploaded file should be triggered.

###### Cloud Signal Processing: Speech API and Plotly

There are two different kinds of files to be uploaded to the cloud bucket, one is a wave (.wav) sound file named as “track.wav” which was recorded using the microphone and uploaded by the Android application, the other one is pitch and roll, for which two text (.txt) file named as “pitch.txt” and “roll.txt”, will be uploaded to the cloud bucket.

The uploaded sound file should be converted to text. To achieve such goal, the team chose to take advantage of Google cloud speech API since the Google Firebase cloud storage and Google cloud function are already in use. By using Google speech recognition API, for example, if the user record “one” in the sound file, a “1” should be saved into a blank text file in the cloud bucket.

The uploaded pitch and roll file should be converted to a 2-D plot of time versus pitch/roll. The team decided to use Plotly, an online plotting website with Node.js API support. Pitch and roll numbers from two text files will be passed to Plotly , and the finished plot will be sent back to the cloud bucket.

###### Npm, Node.js, and Implementation

A set of behavior discussed above need to be implemented in Node.js. The cloud file detection, reading text file, passing data to API and retrieving the results are implemented in a “index.js”.

First, whenever a file is uploaded onto the bucket, the function will use two if statement to check if it is an audio file or text file by reading its header. If it is an audio file, 'This is an audio file' will be printed in the console log, and readAudio function taking the uri, encoding 'LINEAR16' and the sample rate ‘10000’ hertz as inputs will be triggered. Then the Google speech recognition will read the audio file located at the uri and translate the speech to text. If the uploaded file is a text file, a readArray function taking 'microp-d55c0.appspot.com', 'pitch.txt','roll.txt', plot, as parameters, to read the content in those two files as two arrays, where the plot indicates calling the plot functions when the readArray function is done. When the 2000 numbers in the two text files are read as arrays, they will be then passed to the plot function, where the arrays are sent to plot.ly API and plotted accordingly. Once the plot function is done, the downloadImg function will be triggered to download the jpeg format of the plot from plot.ly to the cloud bucket.

#### Testing and Observation

##### Testing before the final demonstration

The audio recorded “one two three four” can be recognized by Google Speech Recognition API, and the result “1 2 3 4” can be printed on console and saved to the bucket correctly.



Figure 5.1.1. Google Speech Recognition testing

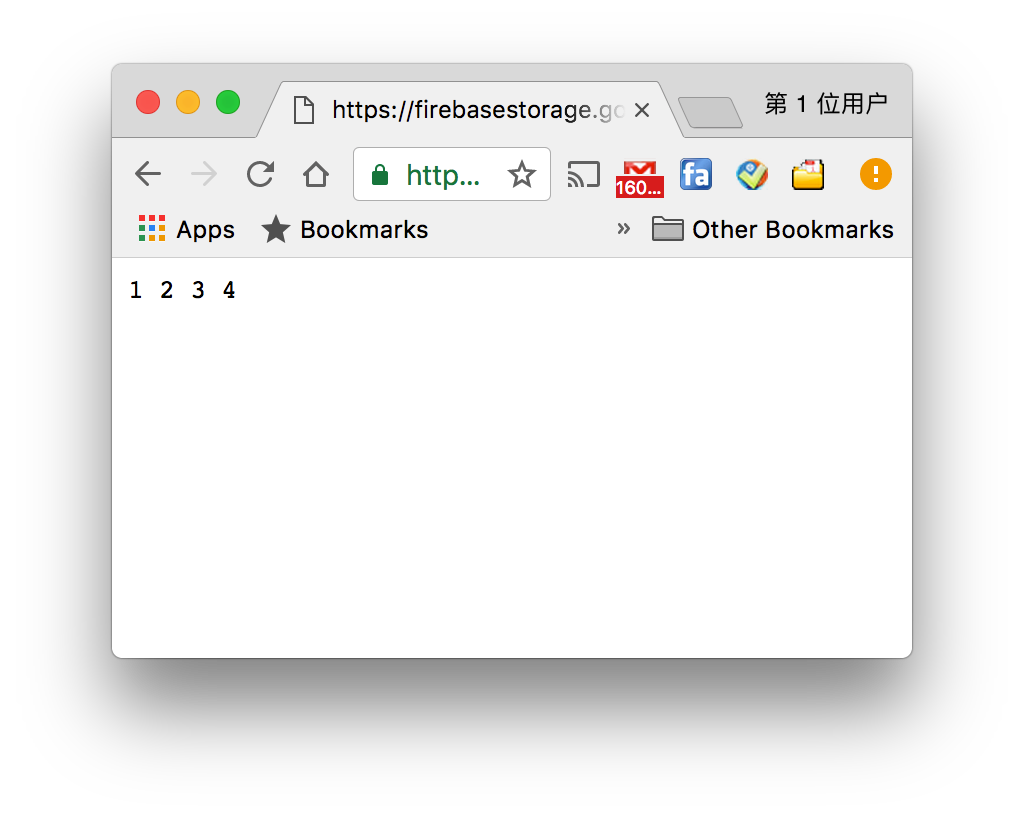


Figure 5.1.2. Google Speech Recognition testing result saved to text file on cloud bucket.

The pitch and roll data are recorded correctly as can be seen in figure 5.1.3. and 5.1.4.,

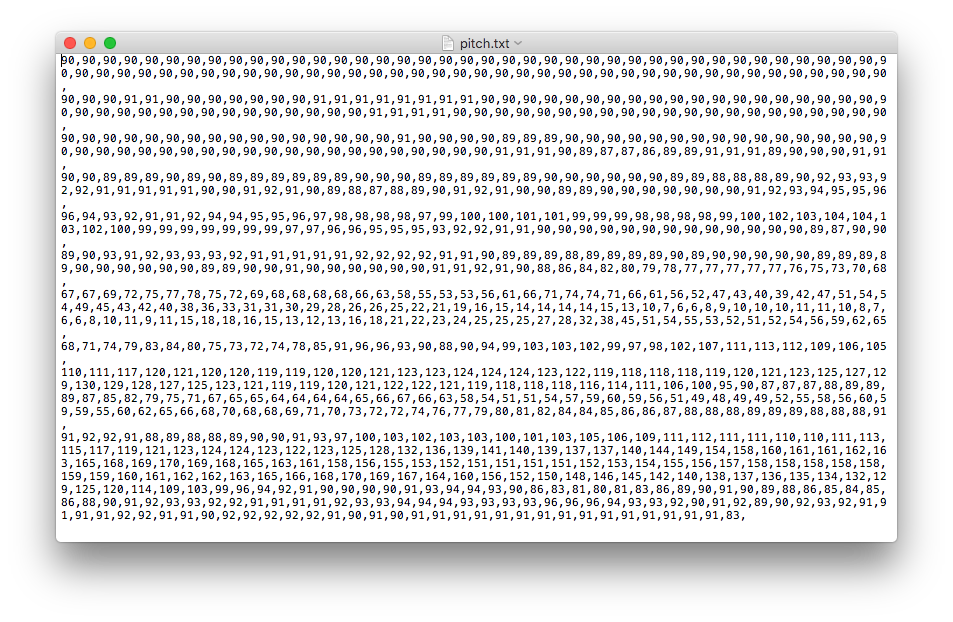


Figure 5.1.3. pitch data recorded and uploaded to the cloud

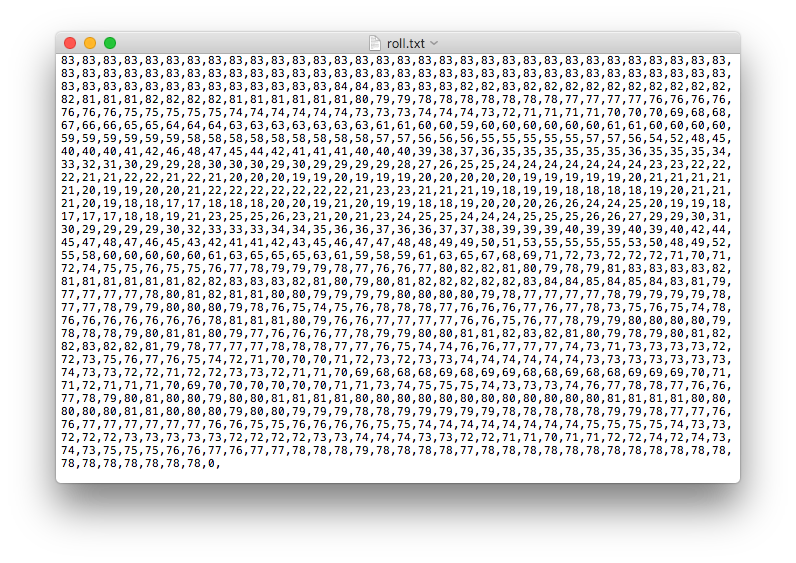


Figure 5.1.4. roll data recorded and uploaded to the cloud

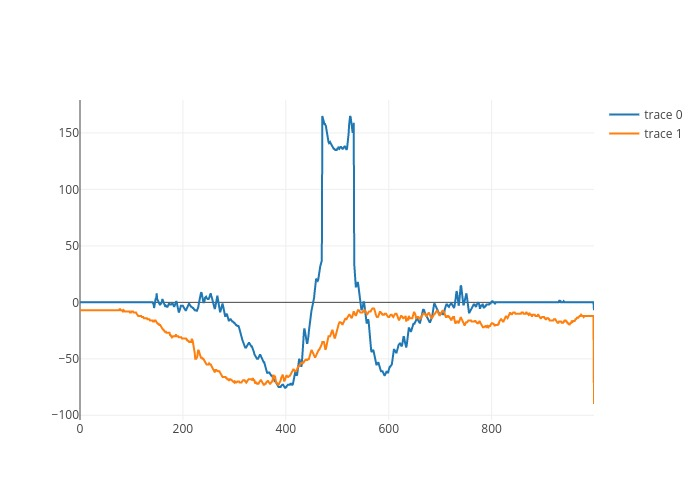
The recorded data are plotted as in figure 5.1.5., where the trace 0 is pitch and trace 1 is roll.

Figure 5.1.5. pitch / roll versus time plot

##### Result during the final demonstration

During the final presentation, pitch and roll data are also recorded and plotted correctly. The set of data can be seen in figure 5.2.1. and 5.2.2., and plot is in figure 5.2.3.

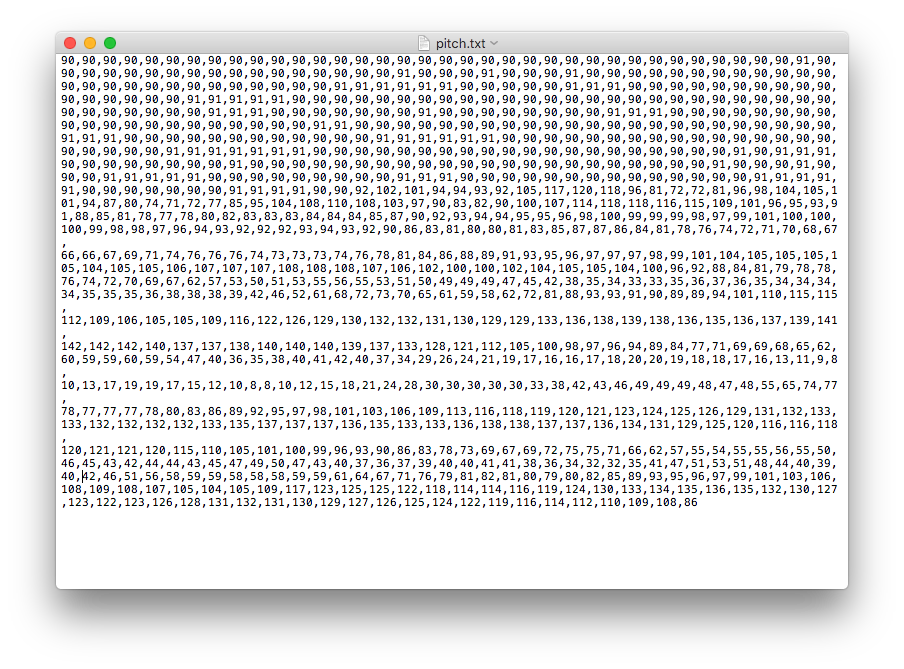


Figure 5.2.1. roll data recorded and uploaded to the cloud during the final demonstration

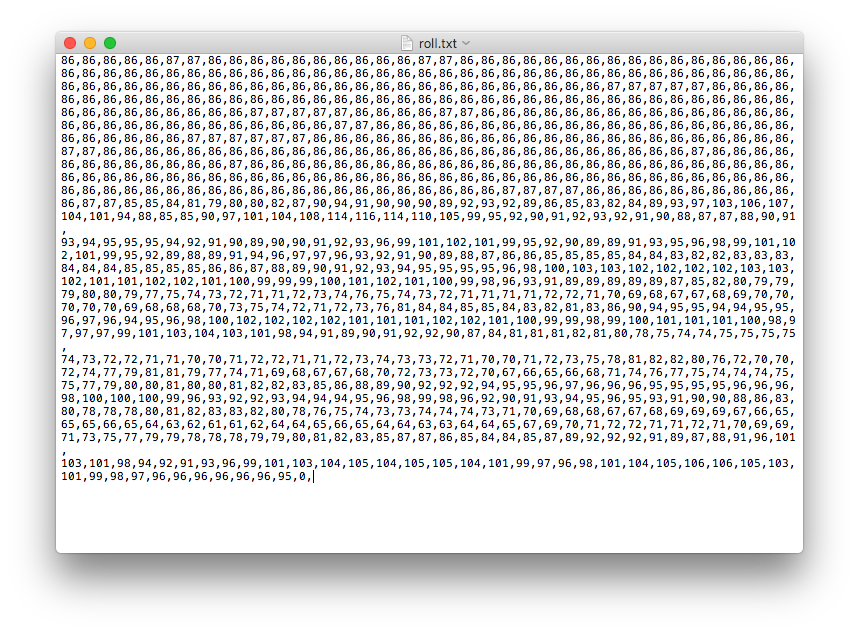


Figure 5.2.2. roll data recorded and uploaded to the cloud during the final demonstration

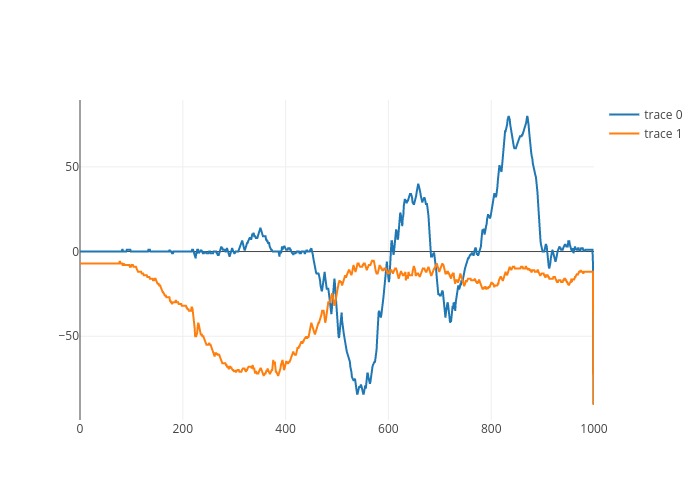


Figure 5.2.3. pitch / roll versus time plot during the final demonstration

During the final presentation, the audio file uploaded is correctly recognized, and the sound recorded is clear, but there was an error calling the Google Speech API, so the speech recognition was not successful. The other parts are all executed as expected, as can be seen in the figure 5.2.4. The text files are correctly recognized, and saved to a temporary file location in memory. Then the 1000 numbers in each of the text file are read into two arrays correctly. After loading the data as two arrays, the plot function is triggered and the result is plotted by Plot.ly and downloaded back to the cloud bucket.

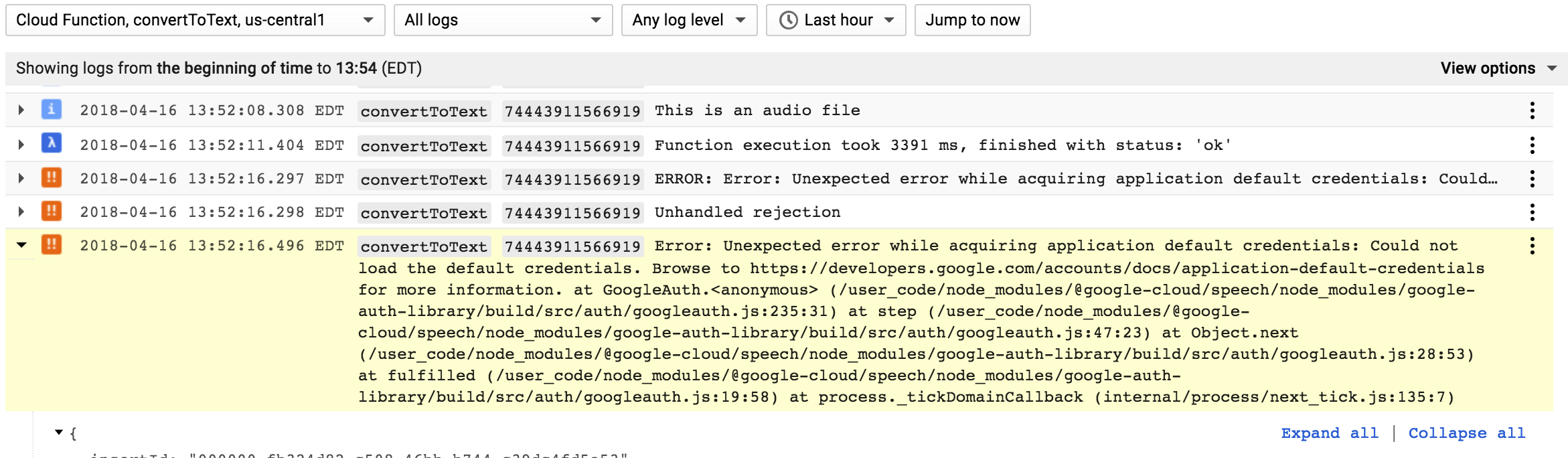


Figure 5.2.4. Google Speech API error during final presentation

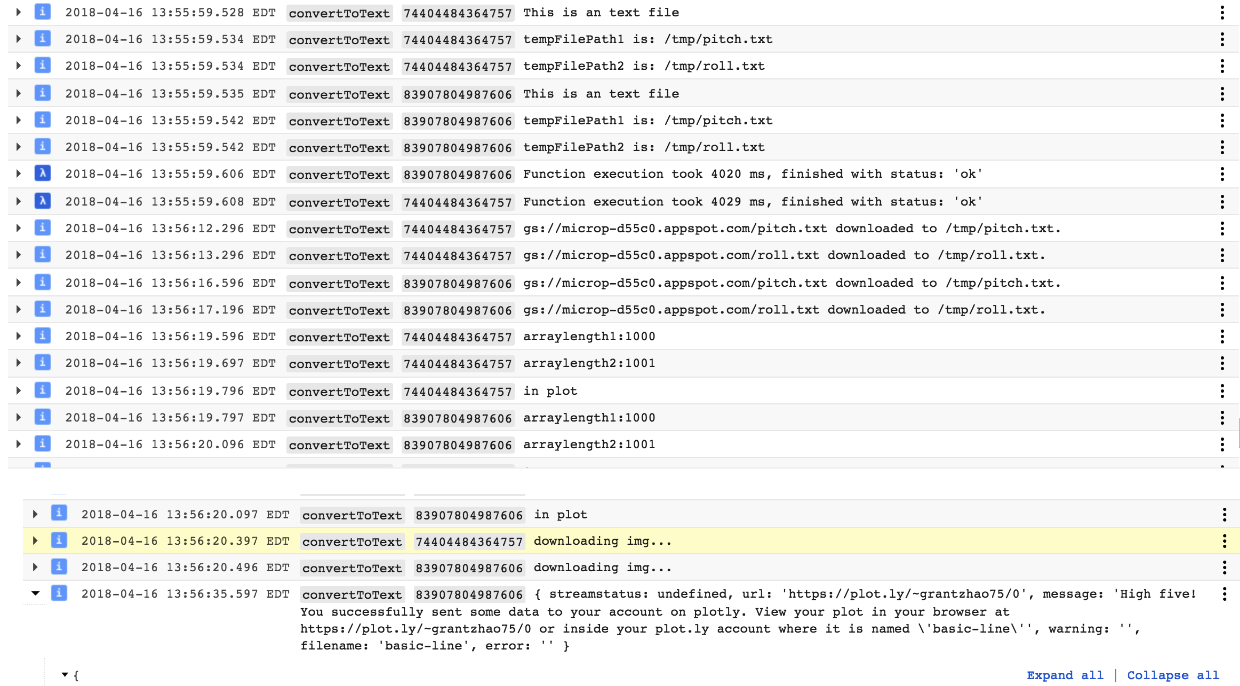


Figure 5.2.5. Console log printout during final presentation

#### Timeline and Division of Work

We divide the project into four parts. Yan works on STM Discovery board, Fei works on Nucleo board, Yingnan works on Android phone application, and Dan works on Firebase cloud service.

***Week 0:***

|  |  |
| --- | --- |
| Members | Tasks |
| Yan | Had group meetings to discuss the project and split the work.  Studied the concepts and theories required for this project.  Made plans for the project. |
| Fei |
| Yingnan |
| Dan |

***Week 1:***

|  |  |
| --- | --- |
| Members | Tasks |
| Yan | Set up STM Discovery board, initialize peripherals. Designed code structure as a finite state machine. Implemented and tested serial connection with Nucleo board via UART. |
| Fei | Set up BLE connection on Nucleo board.  Implemented and tested serial connection with Discovery board via UART. |
| Yingnan | Set up Android Studio. |
| Dan | Set up Firebase cloud service.  Installation of npm, node.js, firebase command line tool locally. |

Progress demo is done at the end of the second week.

***Week 2:***

|  |  |
| --- | --- |
| Members | Tasks |
| Yan | Defined and implemented data transmission protocol for different data buffer  Work on improving tap detection algorithm. |
| Fei | Done receiving different data content, based on defined transmission protocol.  Work on sending data functionality. |
| Yingnan | Built the android app to receive data from Nucleo board, and upload raw data to firebase. |
| Dan | Testing of Google Speech Recognition API using Rest API provided by Google.  Set up cloud storage, and implemented speech recognition functionality in cloud function. |

During the last week, we integrate four parts together, and prepare for the final project demo.

***Week 3:***

|  |  |
| --- | --- |
| Members | Tasks |
| Yan | Improved tap detection algorithm, system testing and debugging. |
| Fei | Done sending data to Android function. |
| Yingnan | Done developing the audio processing part of the android, fine tuning the Android app to fit our needs, and added the function to read and display the content of a output file on the firebase storage. |
| Dan | Implemented plotting functionality and testing the integration of all the cloud services. |

#### Conclusion

In this Project, the required functionalities in the project handout are mostly achieved. A system is developed to detect single and double taps to choose record audio data or accelerometer data accordingly using the STM discovery board, and then the data can be transmitted to the smartphone using the Nucleo board via bluetooth low energy. Then the android app is capable of processing the data and sending the data to firebase for further processing. The sound data on firebase can be recognized using the speech API and the pitch and roll data can be plotted using plotly. We have demonstrated all the functionalities discussed above successfully during the demo.

Some optimization can be done on the data transmitting speed and the APP UI in the future. Currently the data is sent one byte at a time and we are send large chunk of data (10000 bytes for audio and 2000 bytes for pitch and roll). The BLE technology is capable of having a packet size of 20 bytes[4], so if more time is given we should be able to speed up the BLE transmission by 20 times.

#### References

[1]"Microsoft WAVE sound file format", Soundfile.sapp.org, 2018. [Online]. Available: http://soundfile.sapp.org/doc/WaveFormat/. [Accessed: 23- Apr- 2018].

[2]"googlesamples/android-BluetoothLeGatt", GitHub, 2018. [Online]. Available: https://github.com/googlesamples/android-BluetoothLeGatt. [Accessed: 23- Apr- 2018].

[3]"Upload Files on Android | Firebase", Firebase, 2018. [Online]. Available: https://firebase.google.com/docs/storage/android/upload-files. [Accessed: 23- Apr- 2018].

[4]"Maximizing BLE Throughput on iOS and Android - Punch Through", Punchthrough.com, 2018. [Online]. Available: https://punchthrough.com/blog/posts/maximizing-ble-throughput-on-ios-and-android. [Accessed: 23- Apr- 2018].

[5] "JavaScript". *Collins English Dictionary – Complete & Unabridged 2012 Digital Edition*. William Collins Sons & Co. 2012. Retrieved 21 August 2015.

[6]"Why You Really Need (A) node.js development services". codebrahma. January 10, 2018. Retrieved January 10, 2018.