상세설계서

BEE: Be your Eyes and Ears
시청각장애인 의사소통
보조 시스템

Ver. 1.2

2019. 05. 20

한국외국어대학교

융복합 소프트웨어 공학과

3 팀 (B E E)

문서정보

구 분	소 속	성 명	날 짜	비 고
작성자	한국외국어대학교	허 훈	2019. 05. 07	팀장
	한국외국어대학교	고용규	2019. 05. 07	
	한국외국어대학교	고도현	2019. 05. 07	
	한국외국어대학교	송무경	2019. 05. 07	
	한국외국어대학교	이윤주	2019. 05. 07	
	한국외국어대학교	김서연	2019. 05. 07	
검토자	한국외국어대학교	허 훈	2019. 05. 13	
	한국외국어대학교	김서연	2019. 05. 13	
사용자				
승인자	한국외국어대학교	홍진표		

머리말

본 문서는 점자 입/출력부를 구현한 스마트 디바이스와 음성인식 기능을 탑재한 모바일 어플리케이션을 활용해 장애인과 비장애인의 의사소통을 돕고, 더 나아가 장애인과 장애인 간의 소통까지 도울 수 있는 의사소통 보조 서비스 BEE: Be your Eyes and Ears 을 구축하기 위한 시스템들의 상세 설계기술을 설명한다.

개정이력

버전	작성자	개정일자	개정 내역	승인자
1.0	허 훈 고용규 고도현 송무경 이윤주	2019. 05. 07	초안 작성	
	검토자	김서연		
1.1	허 훈 고용규 고도현 송무경 이윤주	2019. 05.13	초안 수정	
	검토자	김서연		
1.2	송무경 고도현 	2019.05.20	1.1v 수정	
	검토자	허훈, 고용규, 김서연, 이윤주		

목차

1. 개요 9
1.1 목적 9
1.2 범위 10
1.2.1 STB: Speech-To-Braille 10
1.2.2 BTS: Braille-To-Speech 11
1.3 관련 문서 11
1.4 용어 및 약어 12
2. 시스템 구성도 13
2.1 전체 시스템 구성도 13
2.2 하드웨어 구성도 14
2.3 어플리케이션 구성도 15
2.4 서버 구성도 16
2.4.1 BEE API Server 구성도 16
2.4.2 BEE Database Server 구성도 17
2.5 Database 구성도 16
3. 기능 설명 18
3.1 BEE Device 18
3.1.1 시리얼 통신 모듈 18
3.1.2 점자 출력 모듈 19
3.1.3 점자 입력 모듈 21
3.2 BEE Application 22
3.2.1 회원가입 및 로그인 요청 모듈 22
3.2.2 사용자 음성-문자 변환 요청 모듈 24
3.2.3 문자-점자(점자-문자) 변환 요청 모듈 26
3.2.4 문자 정보 음성 출력 모듈 27
3.2.5 블루투스 통신 모듈 28
3.3 BEE Server 29
3.3.1 BEE API Server 29
3.3.1.1 Text-To-Braille 29
3.3.1.2 Braille-To-Text 31
3.3.2 BEE Database Server 35
3.3.2.1 사용자 회원가입 모듈 35
3.3.2.2 중복 사용자 확인 모듈 35
3.3.2.3 사용자 로그인 모듈 36

4. 개발환경 38
4.1 OS (Ubuntu) 38
4.2 Tool & Utility 38
4.2.1 AWS EC2 38
4.2.2 AWS Lambda 38
4.2.3 PyCharm 38
4.2.4 Android Studio 39
4.2.5 Database - AWS RDS 39
4.2.6 Node.js 39
4.3 Programming Language 39
4.3.1 C 39
4.3.2 Java 39
4.3.3 Python 40
5. 기능 동작 40
5.1 Dataflow 40
5.2 Sequence Diagram 41
5.2.1 Sign up & Log-in sequence 41
5.2.2 BEE Application – BEE Device communication sequence 41
6. 자체 테스트 방안 42
6.1 어플리케이션 자체 시험 42
6.2 어플리케이션과 웹 서버 간의 시험 42
6.3 어플리케이션과 디바이스 간의 시험 42
7. 향후 프로젝트 적용 방안 43
8. 기대 효과 43
9. 프로젝트 세부추진계획 및 세부일정 44

표 목 차

[Table 1] 관련문서 12
[Table 2] 용어 및 약어 12
[Table 3] 프로젝트 세부일정 44

그 림 목 차

[Figure 1] 전체 시스템 구성도 13
[Figure 2] BEE 하드웨어 구성도 14
[Figure 3] BEE Application 구성도 15
[Figure 4] BEE API Server 구성도 16
[Figure 5] BEE Database Server 구성도 17
[Figure 6] BEE Database 구성도 17
[Figure 7] Arduino 와 스마트폰의 블루투스 통신 18
[Figure 8] 블루투스 연결 스케치 18
[Figure 9] Arduino 점자부 출력 19
[Figure 10] Arduino 출력부 스케치 20
[Figure 11] Arduino Tactile Button 입력 21
[Figure 12] Arduino 입력부 스케치 22
[Figure 13] 안드로이드 내 회원가입 코드 및 실제 사용 모습 23
[Figure 14] 안드로이드 내 로그인 코드 및 실제 사용 모습 24
[Figure 15] 음성인식 인스턴스 생성 및 인식 결과 반환 코드 25
[Figure 16] 실제 음성인식 결과가 표시된 모습 25
[Figure 17] 문자–점자 변환 요청 코드 26
[Figure 18] 점자 변환 수행 이후, 점자가 노출된 모습 27
[Figure 19] TTS 구현을 위한 코드 27
[Figure 20] 안드로이드 내 블루투스 구현 코드 및 실제 구동 화면 28
[Figure 21] 한글 2 진수 점자 변환 기능 29
[Figure 22] 한글 1 종약자 2 진수 점자 변환 코드 29
[Figure 23] 한글 2 종약자 2 진수 변환 코드 30
[Figure 24] 글자를 분해하는 코드 30
[Figure 25] 텍스트-점자 변환 코드 31
[Figure 26] 점자-텍스트 변환 코드 31
[Figure 27] bee_db_handler 내의 사용자 회원가입 부분 35
[Figure 28] bee_db_handler 내의 사용자 아이디 중복 확인 부분 36
[Figure 29] bee_db_handler 내의 사용자 로그인 부분 37
[Figure 30] BEE 서비스 데이터 플로우 40
[Figure 31] Application 에서 회원가입/로그인 기능을 수행하는 시퀀스 다이어그램 41
[Figure 32] Application – Device 간 소통을 위한 시퀀스 다이어그램 41

1. 개요

본 장에서는 Arduino 와 점자 입/출력 센서로 구성된 스마트 디바이스와 안드로이드 어플리케이션을 이용한 시각장애인 및 청각장애인 의사소통 보조 시스템 BEE (Be your Eyes and Ears)에 대한 목적과 범위, 참고자료 그리고 용어 및 약어 등을 제시한다.

1.1 목적

본 프로젝트는 BEE Device 와 BEE Application 을 활용하여 의사소통에 많은 불편을 겪고 있는 시청각 중복 장애인 및 시각/청각장애인들의 의사소통을 기술적으로 보조해줄 수 있는 End-to-End 의사소통 보조 로직을 구축한다.

이를 통하여 구매력이 낮은 장애인분들이 수백만원을 호가하는 값비싼 의사소통 보조기기를 구매하지 않고도, 자신들이 원하는 의사 표현을 상대에게 원활히 전달할 수 있도록 돕는 보조 의사소통 환경을 구축하는 데에 프로젝트의 목적을 둔다.

본 프로젝트의 단기적 목적은 시청각장애인분들의 삶의 질 개선을 위해 최우선 해결과제인 의사소통 서비스 개선에 있다. 그러나, 장기적으로는 본 프로젝트를 통해 발전시킨 기술을 기존에 장애인분들이 쉽게 사용하지 못했던 스마트 디바이스(e.g. 노트북, 스마트폰, etc.)에 추가 부착함으로써 장애인분들이 보다 더 많은 생활 편의를 누릴 수 있도록 하는 데에 그 목적이 있다.

프로젝트를 진행하기 위해 아래 사항들을 구체적으로 명시하고 구현하도록 한다.

(1) 점자 정보의 입출력을 위한 BEE Device 를 위한 하드웨어 구성

(2) 음성 정보의 입출력을 위한 BEE Application 의 작동 방법과 UI

(3) 점자 정보 변환을 위한 '점자-텍스트' 로직과 해당 로직을 포함하는 API Server 의 구성

1.2 범위

장애인분들의 의사소통을 보조하기 위한 본 'BEE' 프로젝트는 두 가지 범위의 기능을 제공한다. 그 첫 번째 범위는 Application to device 이다. 이는 Application 을 통해 읽어 들인 음성 정보를 Google Speech API 로 전송하고, Speech API 로부터 반환 받은 텍스트 데이터를 점자 정보로 변환해 Arduino Device 로 Bluetooth 를 통해 전송하는 것이다.

두 번째 범위는 Device to application 으로, Device 를 통해 점자 정보를 입력 받고, 입력 받은 점자 정보를 Application 으로 전송해 점자 정보를 텍스트 정보 혹은 음성 정보로 변환해 Application 의 output 으로 사용하는 것이다.

따라서 본 프로젝트 개발 진행에 있어 해당 범위들을 다음과 같이 정의한다.

1.2.1 STB: Speech-To-Braille

- Application 을 통한 음성 입력 기능
  - Application 사용자는 Application 과 Application 이 설치된 단말기를 이용해 자신이 전달하고자 하는 음성을 입력할 수 있다.

- Google Speech API 를 통한 Speech-To-Text
  - 사용자가 Application 을 통해 입력한 음성 데이터를 Google Speech API 로 전송하여 그에 상응하는 텍스트 정보를 반환 받을 수 있도록 기능을 제공한다.

- API server 를 통한 텍스트의 Text-To-Braille
  - Google Speech API 로부터 반환 받은 텍스트 정보를 API Server 에 구현되어 있는 Text to braille 로직을 통해 점자 정보로 변환하는 기능을 제공한다.

- Bluetooth 를 통한 점자 정보 전달 기능
  - API Server 로부터 반환 받은 점자 정보를 Bluetooth 기능을 통해 Arduino device 로 전송하는 기능을 제공한다.

- Device 에 점자 출력

- 음성으로부터 전환된 점자 정보를 Device 의 Solenoid 점자 출력부를 통해 출력하는 기능을 제공한다.

1.2.2 BTS: Braille-To-Speech

- Device 를 통한 점자 입력 기능
  - Arduino 를 이용해 device 사용자가 표현하고자 하는 단어 및 문장을 점자 입력부에 입력한다.
- Bluetooth 를 통한 점자 정보 전달 기능
  - 입력 받은 점자 정보를 Bluetooth module 을 통해 Android application 으로 전송한다.
- API server 를 통한 점자의 Braille-To-Text
  - 전송 받은 점자 정보를 API server 에 구현되어 있는 로직을 통해 텍스트로 전환한다.
- Application display 에 텍스트 출력
  - 점자로부터 전환된 텍스트를 Application display 에 출력하여 점자와 텍스트 간의 의사소통이 가능한 기능을 제공한다.
- Application 에 음성 출력
  - 점자로부터 전환된 텍스트를 Application 이 음성으로 출력하여 점자와 음성 간의 의사소통이 가능한 기능을 제공한다.

1.3 관련 문서

출판사 및 출처	제목
에이콘	안드로이드 음성 인식 어플리케이션 개발
Google Cloud	Cloud Speech-to-Text document
Digital books	Django로 쉽게 배우는 파이썬 웹 프로그래밍
카오스북	꿀잼 아두이노 놀이터
복두출판사	스마트폰 • 블루투스 • 이더넷 • Wifi 그리고 아두이노
앤써북	아두이노로 만드는 사물인터넷

성안당	(모바일로 배우는) 아두이노 따라하기
새국어생활	점자 규격 표준화 사업의 필요성

[Table 1] 관련문서

1.4 용어 및 약어

용어 및 약어	풀이
API	Application Programming Interface: 응용 프로그램에서 사용할 수 있도록 기능을 제어할 수 있게 만든 인터페이스
STT	Speech-To-Text: 사람이 말하는 음성 언어를 컴퓨터가 해석해 그 내용을 문자 데이터로 전환하는 처리
TTS	Text-To-Speech: 말소리의 음파를 기계가 자동으로 만들어 내 텍스트를 기계가 스스로 읽는 기술

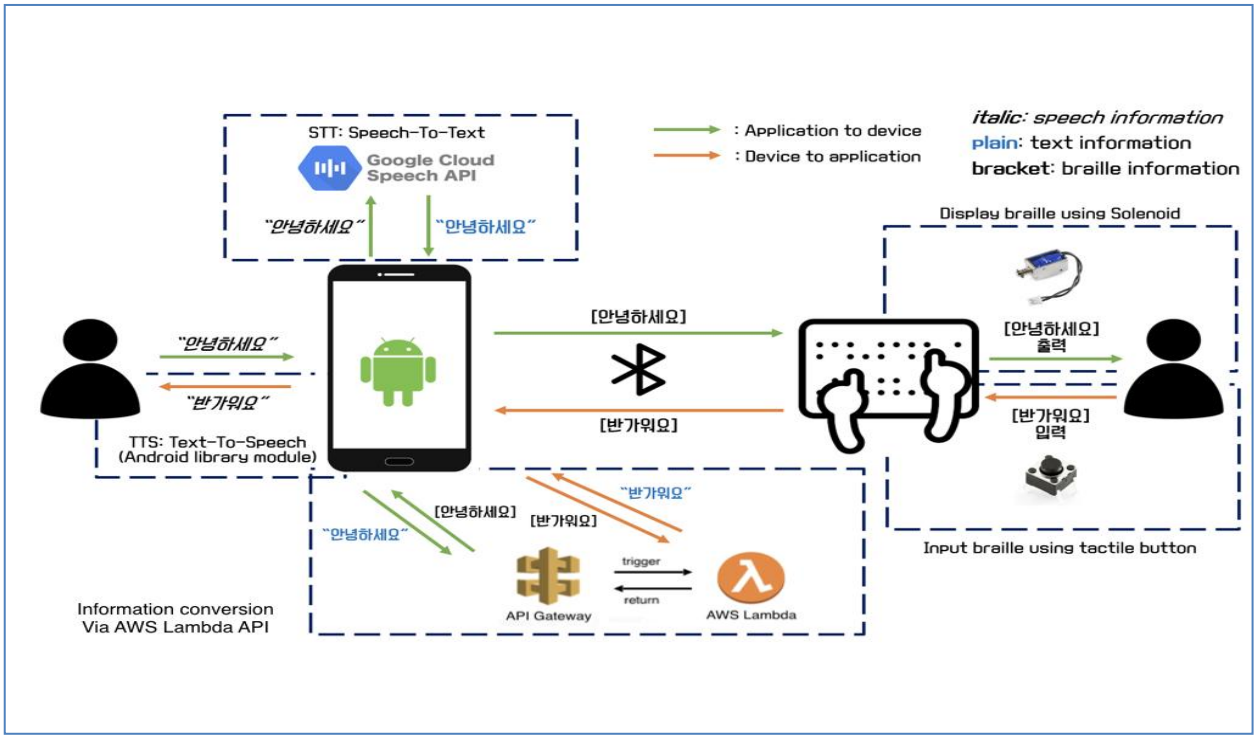
[Table 2] 용어 및 약어

2. 시스템 구성도

본 장에서는 Arduino, Solenoid Actuator, Tactile Button 등을 활용한 스마트 디바이스와 보조 역할을 수행하는 안드로이드 모바일 어플리케이션을 이용하여 시청각중복장애인 그리고 시각/청각장애인과의 의사소통을 보조할 수 있는 BEE 서비스에 대한 전체 시스템 구성과 시스템을 구성하기 위한 개별 소프트웨어, 하드웨어, 서버의 구성도를 기술한다.

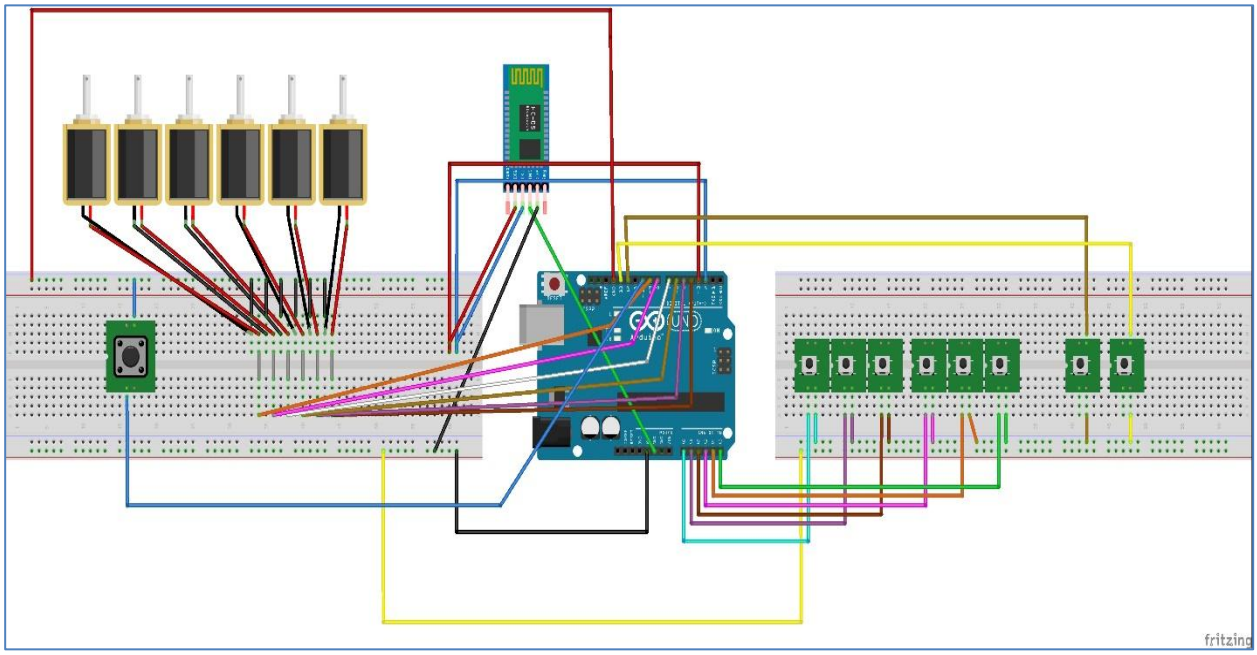
2.1 전체 시스템 구성도

전체 시스템의 구성도는 [Figure 1]과 같이 이루어져 있으며, 시스템을 구성하기 위해 사용되는 개별 하드웨어에 대한 구성도는 2.2 절, 어플리케이션에 대한 구성도는 2.3 절에 다룰 것이다. 또한 모바일 어플리케이션으로부터 발생하는 회원가입 / 로그인 요청을 처리하는 데이터베이스 서버, 점자-문자 간 변환을 담당하는 API 서버의 구성도는 2.4 절에서 다루도록 한다.



[Figure 1] 전체 시스템 구성도

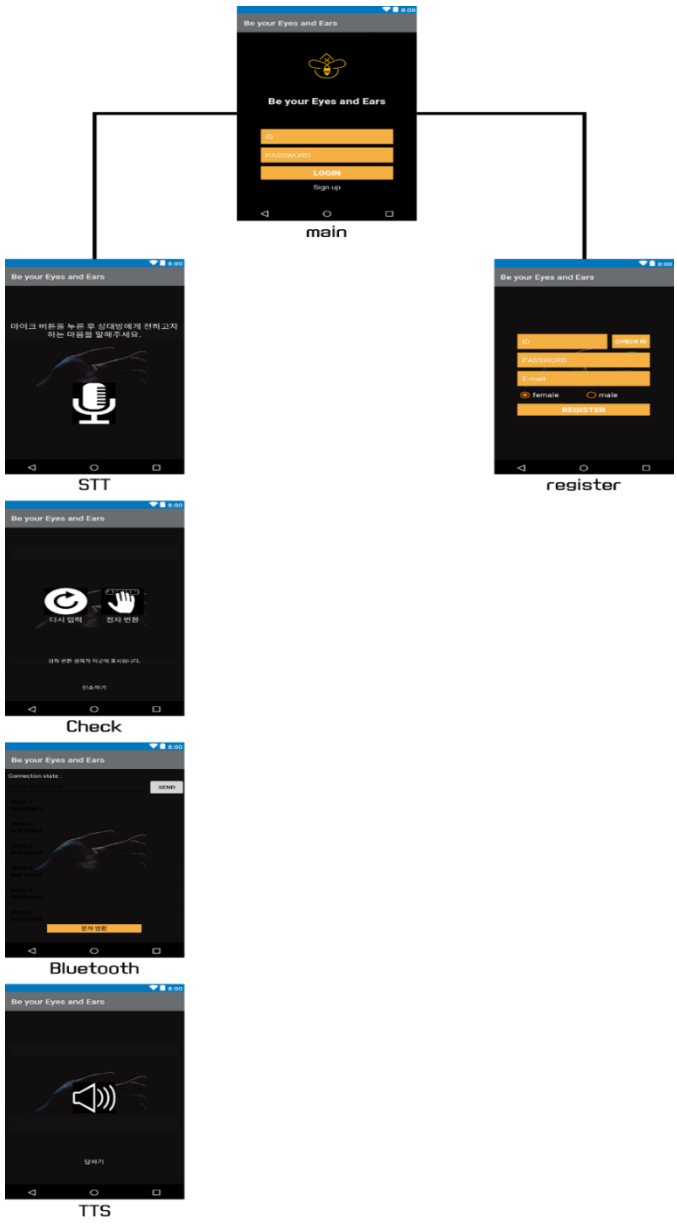
2.2 하드웨어 구성도



[Figure 2] BEE 하드웨어 구성도

BEE Device 의 하드웨어 구성도는 [Figure 2]와 같다. BEE Device 의 메인 보드로는 Arduino Uno 를 활용한다. Device 의 입력부는 Tactile Button 으로, 출력부는 Solenoid Actuator 및 Tactile Button 으로, 블루투스 통신은 HC-06 모듈을 이용하여 제작한다.

2.3 어플리케이션 구성도

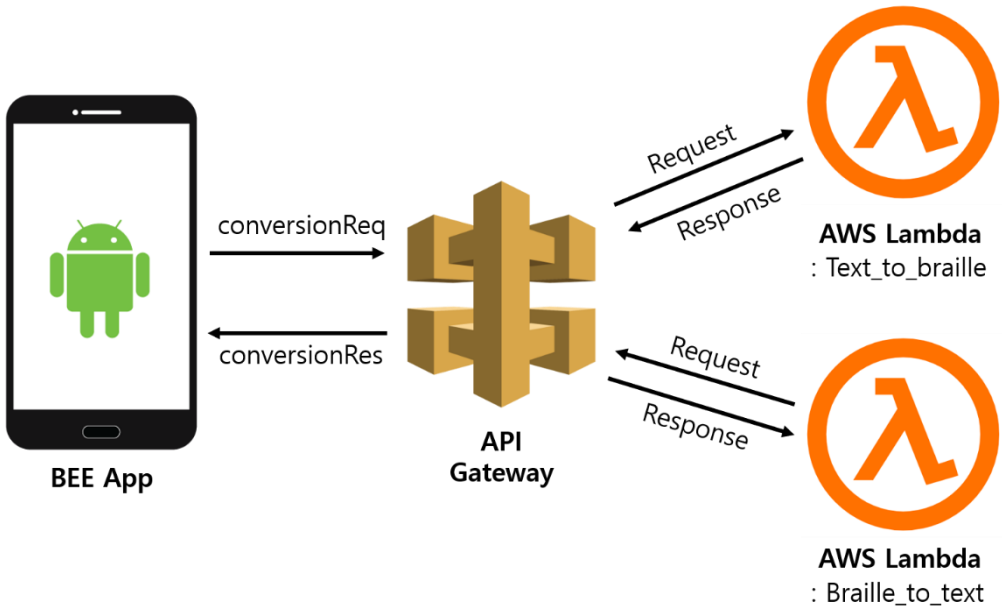


[Figure 3] BEE Application 구성도

BEE 전체 시스템에서 음성인식 기능을 수행하는 어플리케이션의 구성도는 [Figure 3]와 같다. 사용자는 어플리케이션을 사용하기 위해서 회원가입을 수행해야 하며, 회원가입 이후에는 로그인을 거쳐 BEE Application 의 기능을 모두 활용할 수 있게 된다. 사용자의 음성 정보는 Google Speech API 를 통해 문자 정보로 변환되며, 변환된 문자 정보를 BEE API Server 에 점자 정보로 변환 요청을 한 후 BEE Device 에 전송하는 방식으로 정보를 송수신할 수 있게 된다.

2.4 서버 구성도

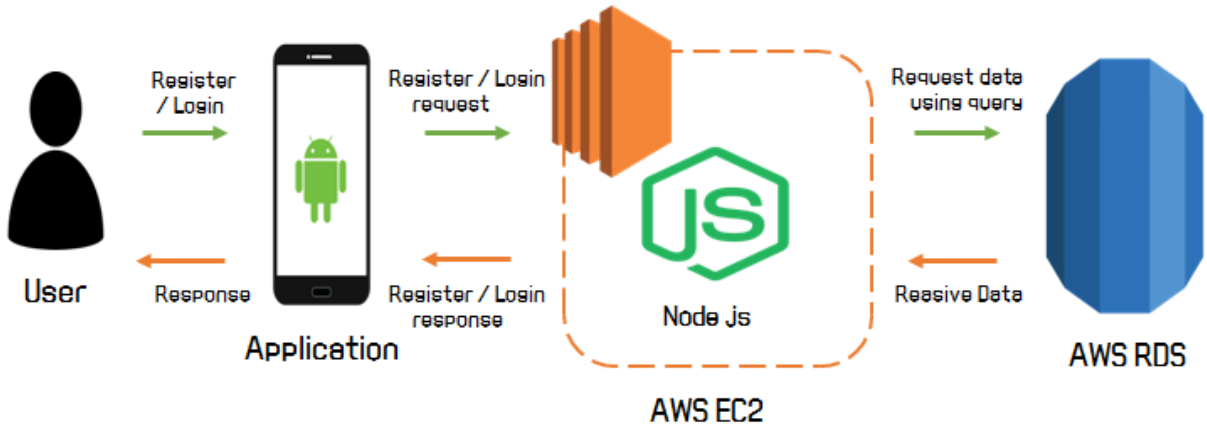
2.4.1 BEE API Server 구성도



[Figure 4] BEE API Server 구성도

문자-점자 간 정보 전환을 수행하는 API Server 는 [Figure 4]와 같이 구성된다. BEE Application 은 점자-문자 변환 혹은 문자-점자 변환 기능을 수행하기 위해 AWS Lambda 에 등록된 함수를 호출해야 한다. 그러나, AWS Lambda 는 IP 나 DNS 를 가진 서버 인스턴스가 아닌 단순 함수이기 때문에 해당 함수를 호출할 수 있는 Trigger 가 존재해야 한다. 그리고 API Gateway 가 그 Trigger 역할을 수행한다. 즉, Application 은 API Gateway 를 통해 Lambda 함수를 호출하고, 입력 값에 해당하는 반환 값을 return 받게 된다.

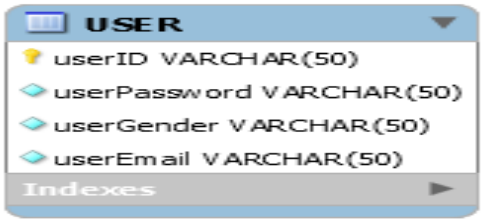
2.4.2 BEE Database Server 구성도



[Figure 5] BEE Database Server 구성도

BEE Application 은 회원제로 사용이 가능하다. 그리고 해당 회원 정보를 관리하기 위한 Database Server 의 구성도는 [Figure 5]와 같다. 어플리케이션 사용자는 AWS EC2 인스턴스에 설치되어 있는 Apache Web server 를 거쳐 데이터베이스(AWS RDS)에 접근할 수 있다. 즉, EC2 인스턴스는 Node.js 코드를 통해 RDS 에 접근해 사용자가 요청한 query 를 수행한 후 결과 값을 반환한다.

2.5 Database 구성도



[Figure 6] BEE Database 구성도

BEE Database 은 한 개의 USER 테이블로 해당 회원 정보를 관리한다. Database 의 USER 테이블 구성도는 [Figure 6]와 같다. Node.js 를 사용하여 query 로 AWS RDS 에 요청할 경우에 해당 USER 테이블에서 요청을 처리하도록 되어있다. BEE 어플리케이션에서는 해당 USER 테이블 이외의 다른 테이블의 필요성이 부족하여, 이외의 테이블을 구현하지 않았기 때문에 Database 정규화 과정을 시도한 결과 또한 [Figure 6]과 같다.

3. 기능 설명

3.1 BEE Device

3.1.1 시리얼 통신 모듈

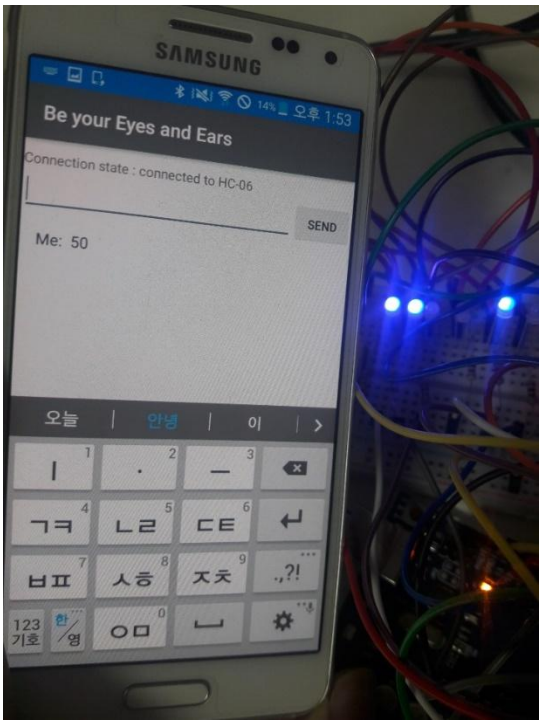
[Figure 7] Arduino 와 스마트폰의 블루투스 통신

통신 모듈의 기능은 어플리케이션과 Arduino 를 블루투스를 통해 연결시켜, 시리얼 통신으로 정보를 주고받는 기능이다. [Figure 7]은 블루투스를 통해 어플리케이션과 디바이스가 연결되어 있음을 보여주는 모습이다.

```
void setup() {
  BEE.begin(9600); // 블루투스 모듈 연결
  Serial.begin(9600);
  pinMode(SOL_1, OUTPUT);
  pinMode(SOL_2, OUTPUT);
  pinMode(SOL_3, OUTPUT);
  pinMode(SOL_4, OUTPUT);
  pinMode(SOL_5, OUTPUT);
  pinMode(SOL_6, OUTPUT);
  pinMode(button1, INPUT_PULLUP);
  pinMode(button2, INPUT_PULLUP);
  pinMode(button3, INPUT_PULLUP);
  pinMode(button4, INPUT_PULLUP);
  pinMode(button5, INPUT_PULLUP);
  pinMode(button6, INPUT_PULLUP);
  pinMode(next, INPUT_PULLUP);
  pinMode(enter, INPUT_PULLUP);
  pinMode(backspace, INPUT_PULLUP);
}
```

[Figure 8] 블루투스 연결 스케치

3.1.2 점자 출력 모듈



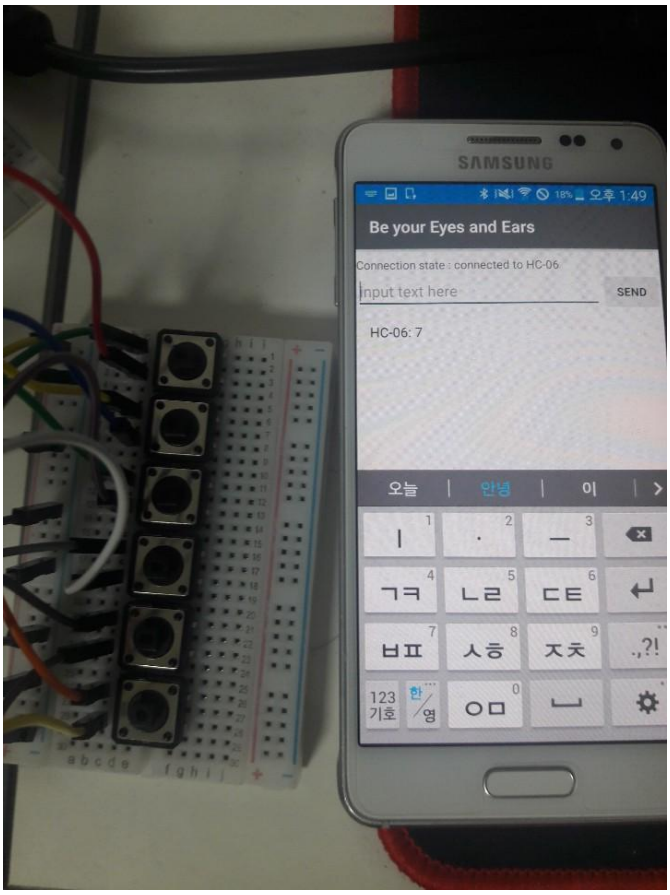
[Figure 9] Arduino 점자부 출력

점자 출력 모듈의 기능은 어플리케이션 사용자로부터 받은 점자정보를 Arduino 의 Solenoid Actuator 를 통해 출력시키는 기능이다. 솔레노이드 6 개와 Tactile Button 1 개로 이루어진 출력부의 기능은 다음과 같이 이루어진다. 어플리케이션으로부터 10 진수의 형태로 정보를 받으면 Arduino 차원에서 2 진수로 변환하여 1 에 해당하는 솔레노이드는 작동시키고, 0 에 해당하는 솔레노이드는 작동을 하지 않는다. 해당 점자를 다 읽으면 Next 버튼을 눌러 대기중인 다음 점자를 출력하도록 한다. [Figure 9]은 어플로부터 받은 점자정보를 Arduino 에서 출력 받은 모습이다.

```
void loop() {
    // 블루투스를 통해 받을 점자정보가 남아있는지 확인, next를 통해 남아있는 정보 수신
    if(BEE.available()&&digitalRead(next)==LOW){
      bufferIndex=0;
      delay(50);
    }
    // 점자정보에서 십단위 숫자를 버퍼에 삽입
    if(BEE.available()&&bufferIndex==0){
      for(int a=0;a<2;a++) {
      buffer[a] = NULL;
      }
      buffer[0]=BEE.read();
      bufferIndex++;
    }
    // 점자정보에서 단단위 숫자를 버퍼에 삽입
    if(BEE.available()&&bufferIndex==1){
      buffer[1]=BEE.read();
      bufferIndex++;
    }

    // 문자로 버퍼에 삽입된 두 자리의 10진수 점자정보를 숫자로 변환
    int bt=atoi(buffer);

  // 2진수로 변환된 점자정보를 출력
  if(cell[0]>0)
    digitalWrite(SOL_1, HIGH);
  else
    digitalWrite(SOL_1, LOW);
  if(cell[1]>0)
    digitalWrite(SOL_2, HIGH);
  else
    digitalWrite(SOL_2, LOW);
  if(cell[2]>0)
    digitalWrite(SOL_3, HIGH);
  else
    digitalWrite(SOL_3, LOW);
  if(cell[3]>0)
    digitalWrite(SOL_4, HIGH);
  else
    digitalWrite(SOL_4, LOW);
  if(cell[4]>0)
    digitalWrite(SOL_5, HIGH);
  else
    digitalWrite(SOL_5, LOW);
  if(cell[5]>0)
    digitalWrite(SOL_6, HIGH);
  else
    digitalWrite(SOL_6, LOW);
```

[Figure 10] Arduino 출력부 스케치

3.1.3 점자 입력 모듈



[Figure 6] Arduino Tactile Button 입력

점자 입력 모듈의 기능은 디바이스 사용자의 점자정보를 Arduino 로 입력하여 어플리케이션에 전달하는 기능이다. BEE 에 장착되어 있는 9 개의 Tactile Button 을 통해 해당 기능을 수행한다. 총 6 개의 점자부와 1 개의 Enter, 1 개의 Backspace, 1 개의 send 버튼으로 구성되어 있는 입력부의 기능은 다음과 같이 이루어진다. [Figure 11]은 BEE 에서 전송 받은 점자정보를 어플리케이션에서 확인하는 모습이다. 각각의 점자 버튼은 2 진수로 인식되어 입력된 점자들의 총합을 10 진수의 형태로 어플리케이션으로 전달한다.

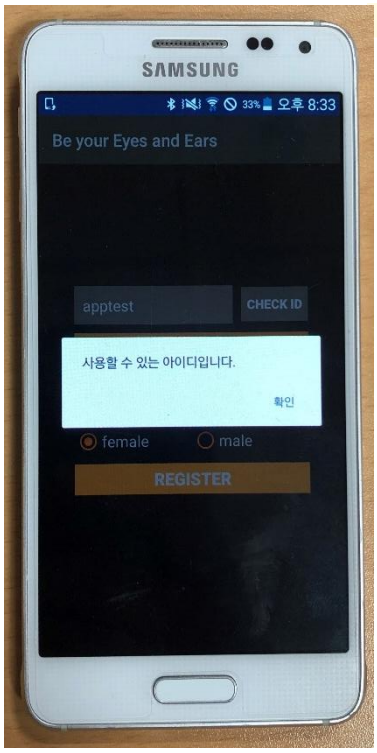
```
    // 점자 버튼 입력시 입력 되었음을 알려주는 트리거
    b1+=!digitalRead(button1);
    b2+=!digitalRead(button2);
    b3+=!digitalRead(button3);
    b4+=!digitalRead(button4);
    b5+=!digitalRead(button5);
    b6+=!digitalRead(button6);

    // enter 버튼 누를 시 입력된 점자정보를 어플로 전송 후 점자정보 초기화
    if(digitalRead(enter)==LOW){
      if(b1>0)
        b1=1;
      if(b2>0)
        b2=1;
      if(b3>0)
        b3=1;
      if(b4>0)
        b4=1;
      if(b5>0)
        b5=1;
      if(b6>0)
        b6=1;
      int sum=b1+(b2*2)+(b3*4)+(b4*8)+(b5*16)+(b6*32);
      BEE.println(sum);
      //Serial.println(sum);
      b1=0;b2=0;b3=0;b4=0;b5=0;b6=0;sum=0;
      delay(500);
    }
     // backspace 버튼 누를 시 입력중인 점자정보 모두 초기화
     if(digitalRead(backspace)==LOW){
       b1=0;b2=0;b3=0;b4=0;b5=0;b6=0;sum=0;
       delay(100);
     }
     delay(100);
}
```

[Figure 7] Arduino 입력부 스케치

3.2 BEE Application

3.2.1 회원가입 및 로그인 요청 모듈

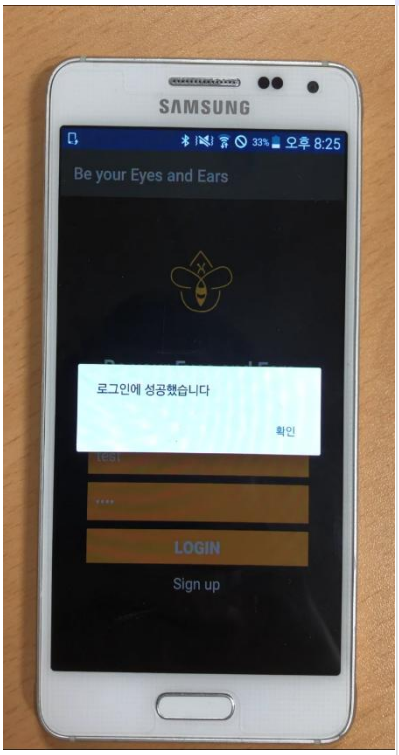
BEE 시스템을 사용하기 위해서는 기본적으로 BEE 서비스에 회원으로 등록이 되어 있어야 한다. 그리고 이러한 회원의 등록은 BEE Application 을 이용해 BEE Database Server 에 회원 가입을 요청함으로써 수행할 수 있다. 회원가입을 위해 사용자가 입력해야 하는 정보로는 "사용자 ID, 사용자 암호, 성별, 이메일"이 있으며, Database Server 로부터 JSON 객체를 통해 "success" 문자열을 반환 받게 될 경우 회원가입을 성공적으로 수행한 것으로 인식한다.




```
// 회원 가입 요청 시작
Response.Listener<String> responseListener = new Response.Listener<String>(){
    @Override
    public void onResponse(String response) {
        try{
            JSONObject jsonResponse = new JSONObject(response);
            boolean success = jsonResponse.getBoolean( name: "success");
            if(success){ // 사용할 수 있는 아이디라면
                AlertDialog.Builder builder = new AlertDialog.Builder( context:
                dialog = builder.setMessage("회원 가입이 완료되었습니다.")
                        .setPositiveButton( text: "확인", listener: null)
                        .create();
                dialog.show();
                finish(); // 액티비티를 종료시킴(회원등록 창을 닫음)
            }else{ // 사용할 수 없는 아이디라면
                AlertDialog.Builder builder = new AlertDialog.Builder( context:
                dialog = builder.setMessage("회원 가입에 실패했습니다.")
                        .setNegativeButton( text: "확인", listener: null)
                        .create();
                dialog.show();
            }
        }
```

[Figure 8] 안드로이드 내 회원가입 코드 및 실제 사용 모습

로그인의 경우, [Figure 13]와 같이 사용자가 입력한 정보 ID 와 암호 정보가 BEE Database 에 등록되어 있는지 확인한 후, 존재하는 정보일 경우 true 값을 반환하게 된다. 즉, JSON 객체를 통해 "success" 문자열을 반환 받는 경우, 로그인이 성공적으로 수행한 것으로 인식해 BEE Application 의 첫 번째 액티비티인 STT(Speech-To-Text) 액티비티로 사용자 화면을 이동시켜준다. 사용자가 입력한 정보가 존재하지 않는 경우, 새로운 시도를 하도록 유도한다.




```
loginButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        String userID = idText.getText().toString();
        String userPassword = passwordText.getText().toString();
        Response.Listener<String> responseLisner = new Response.Listener<String>(){
            @Override
            public void onResponse(String response) {
                try{
                    JSONObject jsonResponse = new JSONObject(response);
                    boolean success = jsonResponse.getBoolean( name: "success");

                    if(success){
                        AlertDialog.Builder builder = new AlertDialog.Builder( context: MainActi
                        dialog = builder.setMessage("로그인에 성공했습니다")
                                .setPositiveButton( text: "확인",  listener: null)
                                .create();
                        dialog.show();
                        Intent intent = new Intent( packageContext: MainActivity.this, SttActivi
                        MainActivity.this.startActivity(intent);
                        finish();
                    }else {
                        AlertDialog.Builder builder = new AlertDialog.Builder( context: MainActi
                        dialog = builder.setMessage("아이디 혹은 비밀번호를 다시 확인해주세요")
                                .setNegativeButton( text: "다시 시도",  listener: null)
                                .create();
                        dialog.show();
                    }
```

[Figure 9] 안드로이드 내 로그인 코드 및 실제 사용 모습

3.2.2 사용자 음성-문자 변환 요청 모듈

사용자는 BEE Application 이 설치된 단말기의 마이크를 통해 입력한 음성 정보를 문자로 변환하도록 Google Speech API 에 요청할 수 있다. 해당 API 는 Google Cloud Platform 에 Google Speech 프로젝트를 생성한 후 Android Project 에 프로젝트 API 키가 포함된 JSON 파일을 추가하여 사용할 수 있으며, 프로젝트의 Gradle 빌더에 Speech 모듈을 추가해주어야 한다.

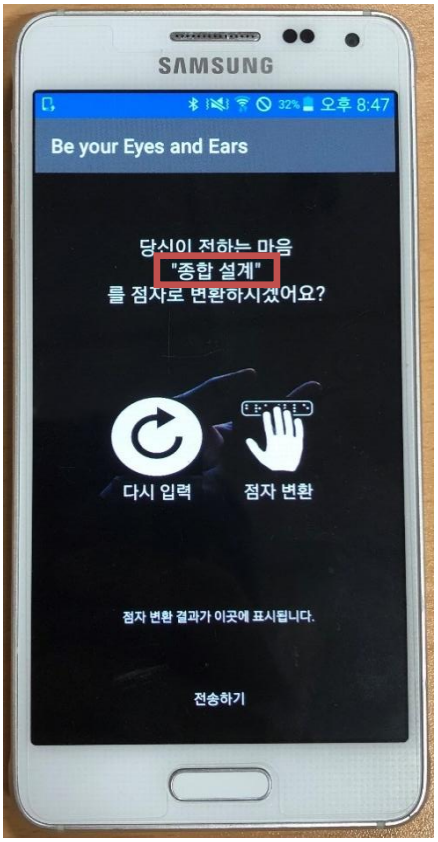
API 키의 추가 이후, 안드로이드 프로젝트에서 [Figure 14]와 같이 Speech Recognizer 클래스를 생성하여 음성인식 기능을 수행할 수 있다. 이 때 해당 음성인식 기능을 서비스 차원에서 프로젝트에 적절하게 사용하기 위한 하나의 주의사항이 있다.

음성인식 인스턴스는 사용자의 음성을 읽어 들인 후, 해당 음성에 따른 문자열을 하나의 최적 답안 하나가 아닌 여러 개의 경우의 수를 결과로 반환한다. 따라서, 사용자에게 하나의 답을 반환해주기 위해서는 결과 리스트에서 0 번째 인덱스에 해당하는. 즉, 가장 유사도가 높다고 판단된 음성인식 결과를 사용자에게 반환해주도록 설정해야 한다.

```
sttBtn.setOnClickListener(v ->{
    mRecognizer = SpeechRecognizer.createSpeechRecognizer(this);
    mRecognizer.setRecognitionListener(listener);
    mRecognizer.startListening(intent);
});
```

```
@Override
public void onResults(Bundle results) {
    ArrayList<String> matches =
            results.getStringArrayList(SpeechRecognizer.RESULTS_RECOGNITION);
    Intent intent = new Intent(getApplicationContext(), ChkActivity.class);
    intent.putExtra( name: "sentence", matches.get(0)); // 1번 후보 문장 채택
    startActivity(intent);
}
```

[Figure 10] 음성인식 인스턴스 생성 및 인식 결과 반환 코드



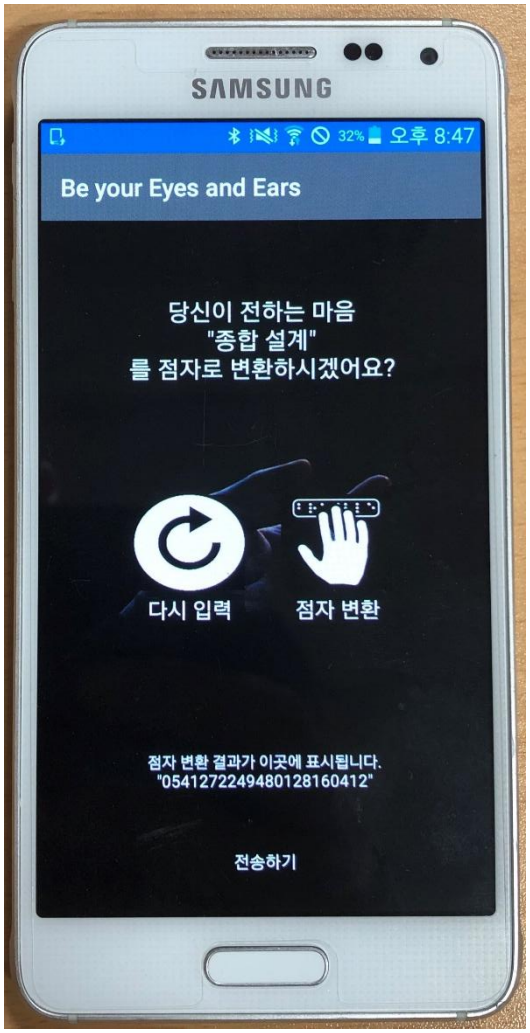
[Figure 11] 실제 음성인식 결과가 표시된 모습

3.2.3 문자-점자(점자-문자) 변환 요청 모듈

사용자는 Google Speech API 로부터 반환 받은 문자열을 BEE API Server 에 점자 정보로 변환하도록 요청할 수 있다. 음성인식 결과 값을 반환 받은 사용자가 '점자 변환' 버튼을 누름에 따라 해당 요청이 전송되며, 어플리케이션은 JSON 객체의 'body' 키로 점자 변환 값을 반환 받을 수 있다. 이후, 사용자는 반환 받은 점자 정보를 확인한 후 BEE Device 에게 전송을 수행할 수 있다.

```
public void getJSON() {
    AsyncTask.execute(new Runnable() {
        @Override
        public void run() {
            try {
                URL url = new URL(REQUEST_URL);
                HttpsURLConnection myConnection = (HttpsURLConnection)url.openConnection();

                if (myConnection.getResponseCode() == 200) {// Success
                    InputStream responseBody = myConnection.getInputStream();
                    InputStreamReader responseBodyReader =
                            new InputStreamReader(responseBody, charsetName: "UTF-8");

                    JsonReader jsonReader = new JsonReader(responseBodyReader);
                    jsonReader.beginObject(); // Start processing the JSON object

                    while (jsonReader.hasNext()) { // Loop through all keys
                        String key = jsonReader.nextName(); // Fetch the next key
                        if (key.equals("body")) { // Check if desired key
                            value = jsonReader.nextString();

                            Bundle bun = new Bundle();
                            bun.putString("value", value);
                            Message msg = handler.obtainMessage();
                            msg.setData(bun);
                            handler.sendMessage(msg);

                            progressDialog.dismiss();
                            break; // Break out of the loop
                        } else { // Skip values of other keys
                            jsonReader.skipValue();
                        }
                    }
                    jsonReader.close();
```

[Figure 12] 문자–점자 변환 요청 코드



[Figure 13] 점자 변환 수행 이후, 점자가 노출된 모습

이후 BEE Device 로부터 전송 받은 점자 정보를 문자 정보로 변환하기 위해서는 점자-문자 변환의 로직을 수행하는 코드를 API Server 에 요청함으로써 수행할 수 있다.

3.2.4 문자 정보 음성 출력 모듈

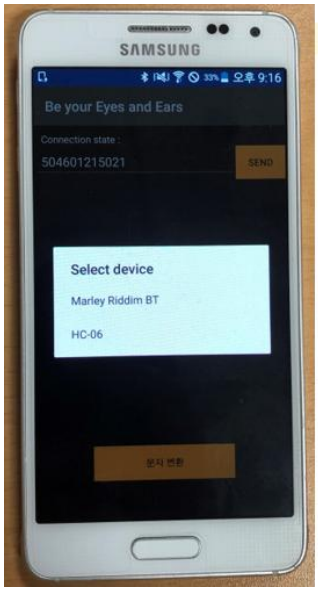
```
tts =new TextToSpeech(getApplicationContext(), new TextToSpeech.OnInitListener() {
    @Override
    public void onInit(int status) {
        if (status != TextToSpeech.ERROR) {
            tts.setLanguage(Locale.KOREAN);
        }
    }
});
```

[Figure 19] TTS 구현을 위한 코드

사용자는 BEE API Server 로부터 반환 받은 문자 정보를 음성으로 들어볼 수 있다. 해당 기능은 BEE Application 을 비장애인만이 아닌 시각장애인도 사용이 가능하게 위해 존재하는 기능이다. 음성 변환 기능은 안드로이드 운영체제가 제공하는 TTS 클래스를 활용하여 구현한다. 어플리케이션은 API Server 로부터 변환 받은 문자 정보를 화면에 노출시켜주며, 사용자는 '음성으로 듣기' 버튼을 눌러 TTS 기능을 활용할 수 있게 된다.

3.2.5 블루투스 통신 모듈

BEE Application 은 Arduino 를 기반으로 구현된 BEE Device 와 블루투스 연결을 통해 정보를 주고 받을 수 있어야 한다. 그리고 어플리케이션 자체에서 이를 구현하기 위한 코드를 [Figure 19]와 같이 작성한다. 블루투스는 어플리케이션을 실행하기 전에 아두이노 디바이스와 연결되어 있어야 하며, API 서버로부터 반환 받은 점자 정보를 전송할 때 연결하고자 하는 기기를 재확인하게 된다.




```
private class ConnectedTask extends AsyncTask<Void, String, Boolean> {
    private InputStream mInputStream = null;
    private OutputStream mOutputStream = null;
    private BluetoothSocket mBluetoothSocket = null;

    ConnectedTask(BluetoothSocket socket){
        mBluetoothSocket = socket;
        try {
            mInputStream = mBluetoothSocket.getInputStream();
            mOutputStream = mBluetoothSocket.getOutputStream();
        } catch (IOException e) {
            Log.e(TAG, msg: "소켓 생성에 실패했습니다.", e );
        }
        Log.d( TAG, msg: "Connected to "+mConnectedDeviceName);
        mConnectionStatus.setText( "Connected to "+mConnectedDeviceName);
    }
```

[Figure 20] 안드로이드 내 블루투스 구현 코드 및 실제 구동 화면

어플리케이션과 연결할 기기를 재확인한 후, 사용자가 전송하고자 하는 점자 정보가 아두이노로 전송되게 되며 동일한 화면에서 아두이노 사용자가 입력한 점자 정보 역시 수신할 수 있다.

3.3 BEE Server

3.3.1 BEE API Server

3.3.1.1 Text-To-Braille

BEE Device 로부터 전달받은 데이터를 2 진수 점자 정보로 바꾸기 위한 기능을 수행한다. 즉, API Server 를 통한 텍스트의 Text-To-Braille 로직은 Speech API 로부터 반환 받은 텍스트 정보를 API Server 에 구현되어 있는 로직을 통해 점자 정보로 변환하는 기능을 수행한다.

```
MATCH_H2B_CHO = {
    u'ㄱ': '04,',
    u'ㄴ': '36,',
    u'ㄷ': '20,',
    u'ㄹ': '02,',
    u'ㅁ': '34,',
    u'ㅂ': '06,',
    u'ㅅ': '01,',
    u'ㅇ': '54,',
    u'ㅈ': '05,',
    u'ㅊ': '03,',
    u'ㅋ': '52,',
    u'ㅌ': '50,',
    u'ㅍ': '38,',
    u'ㅎ': '22,',

    u'ㄲ': ['01,', '04,'],
    u'ㄸ': ['01,', '20,'],
    u'ㅃ': ['01,', '06,'],
    u'ㅆ': ['01,', '01,'],
    u'ㅉ': ['01,', '05,'],
}

MATCH_H2B_JOONG = {
    u'ㅏ': '49,',
    u'ㅑ': '14,',
    u'ㅓ': '28,',
    u'ㅕ': '35,',
    u'ㅗ': '41,',
    u'ㅛ': '13,',
    u'ㅜ': '44,',
    u'ㅠ': '37,',
    u'ㅡ': '21,',
    u'ㅣ': '42,',
    u'ㅐ': '58,',
    u'ㅔ': '46,',
    u'ㅒ': ['14,', '58,'],
    u'ㅖ': '12,',
    u'ㅘ': '57,',
    u'ㅙ': ['57,', '58,'],
    u'ㅚ': '47,',
    u'ㅝ': '60,',
    u'ㅞ': ['60,', '58,'],
    u'ㅟ': ['44,', '58,'],
    u'ㅢ': '23,',
}

MATCH_H2B_JONG = {
    u'ㄱ': '32,',
    u'ㄴ': '18,',
    u'ㄷ': '10,',
    u'ㄹ': '16,',
    u'ㅁ': '17,',
    u'ㅂ': '48,',
    u'ㅅ': '08,',
    u'ㅇ': '27,',
    u'ㅈ': '40,',
    u'ㅊ': '24,',
    u'ㅋ': '26,',
    u'ㅌ': '25,',
    u'ㅍ': '19,',
    u'ㅎ': '11,',
    u'ㄳ': ['32,', '32,'],
    u'ㄵ': ['32,', '08,'],
    u'ㄶ': ['18,', '40,'],
    u'ㄺ': ['18,', '11,'],
    u'ㄻ': ['16,', '32,'],
    u'ㄼ': ['16,', '17,'],
    u'ㄽ': ['16,', '48,'],
    u'ㄾ': ['16,', '08,'],
    u'ㄿ': ['16,', '25,'],
    u'ㅀ': ['16,', '11,'],
    u'ㅄ': ['48,', '08,'],
    u'ㅆ': '12,',
}

MATCH_H2B_ECT = {
    '1': ['15', '32'],
    '2': ['15', '48'],
    '3': ['15', '36'],
    '4': ['15', '38'],
    '5': ['15', '34'],
    '6': ['15', '52'],
    '7': ['15', '54'],
    '8': ['15', '50'],
    '9': ['15', '20'],
    '0': ['15', '22'],

    ',': '16',
    '.': '19',
    '-': '18',
    '?': '23',
    '_': '9',
    '!': '26',
}
```

[Figure 21] 한글 2 진수 점자 변환 코드

점자의 약자 표기는 특히 많이 사용되는 문자의 점자 표기를 단순화한 것으로, 정자로 점자 표기하면 2-3 칸(Cell)이 소요되는 것을 점자 1 칸(Cell)으로 약자화하여 대응시킨다. 한글 1 종약자에 대응되는 점자 기호는 [Figure 22]과 같으며, 한글 2 종약자에 대응되는 점자 기호는 [Figure 23]과 같다.

```
def second_handler(second_text):
    abbreviation = {'가': '53', '나': '36', '다': '20', '마': '34',
                    '바': '06', '사': '56', '자': '05', '카': '52',
                    '타': '50', '파': '38', '하': '22',

                    '억': '39', '언': '31', '얼': '30', '연': '33',
                    '열': '51', '영': '55', '옥': '45', '온': '59',
                    '옹': '63', '운': '54', '울': '61', '은': '43',
                    '을': '29', '인': '62',
                    }

    if second_text in abbreviation:
        return abbreviation[second_text]
    else:
        return False
```

[Figure 22] 한글 1 종약자 2 진수 점자 변환 코드

```
def fastest_handler(fast_text):
    abbreviation = {'그래서': '3228',
                    '그러나': '3236',
                    '그러면': '3218',
                    '그러므로': '3217',
                    '그런데': '3246',
                    '그리고': '3241',
                    '그리하여': '3235',
                    '것': '0728'}

    if fast_text in abbreviation:
        return abbreviation[fast_text]
    else:
        return False
```

[Figure 23] 한글 2 종약자 2 진수 변환 기능

한글 점자는 초성, 중성, 종성을 풀어쓰기로 표현하기 때문에 이를 구분해주기 위한 기능이 필요하다. [Figure 24]는 이를 위해 2 진수로 변환된 초성, 종성, 중성 자모음 변수를 활용하여 글자를 분해하는 코드이다.

```
def letter(hangul_letter):
    result = ""

    hangul_decomposed = hgtk.text.decompose(hangul_letter[0])
    hangul_decomposed = 
        hangul_decomposed.replace(hgtk.text.DEFAULT_COMPOSE_CODE, '')
    cho=''
    jung=''
    jong=''
    for i in range(len(hangul_decomposed)):
        hangul = hangul_decomposed[i]
        if i == 0 and hangul in MATCH_H2B_CHO:
            cho = MATCH_H2B_CHO[hangul]
        if i == 0 and hangul in MATCH_H2B_ECT:
            cho = MATCH_H2B_ECT[hangul]
        if i == 1 and hangul in MATCH_H2B_JOONG:
            jung = MATCH_H2B_JOONG[hangul]
        #종성이 있는 경우
        if i == 2 and hangul in MATCH_H2B_JONG:
            jong = MATCH_H2B_JONG[hangul]
            #초성+중성 1종약자
            if (second_handler(hgtk.letter.compose(hangul_decomposed[0], hangul_decomposed[1]))):
                result = second_handler(hgtk.letter.compose(hangul_decomposed[0], hangul_decomposed[1])) + jong
            #중성+종성 1종약자
            elif (second_handler(hgtk.letter.compose('ㅇ', hangul_decomposed[1], hangul_decomposed[2]))):
                result = cho + second_handler(hgtk.letter.compose('ㅇ', hangul_decomposed[1], hangul_decomposed[2]))
            #1종약자 미포함
            else:
                result = cho + jung + jong
        #종성이 없는 경우
    #종성이 없는 경우
    else:
        jong=''
        #초성+중성 1종약자
        if (second_handler(hgtk.letter.compose(hangul_decomposed[0], hangul_decomposed[1]))):
            result = second_handler(hgtk.letter.compose(hangul_decomposed[0], hangul_decomposed[1]))
        #1종약자 미포함
        else:
            result = cho + jung
if result == []:
    result += 'OO'
return result
```

[Figure 24] 글자를 분해하는 코드

[Figure 25]는 한글을 점자로 변환시켜주는 코드이다. 입력 받은 텍스트를 2 종약자 규칙에 의거하여 먼저 실행시킨 후 점자로 변환한다. 문자 단위로 다시 분석하여 글자를 분해한 후 1 종약자 규칙을 적용시키며, 어떠한 규칙에도 해당되지 않는 글자를 2 진수의 점자 형태로 표현하도록 한다.

```
def main(text):
    result = ""
    text = text.split(" ")

    # 이종약자 규칙 먼저 걸러줌
    for elem in text:
        # 이종약자 규칙 함수 호출
        fast_result = fastest_handler(elem)

        # 이종약자 규칙에 걸렸으면 해당 반환값 전체 result에 추가
        if fast_result != False:
            result += fast_result

        # 이종약자 규칙에 안걸렸으면 문자 단위 분석 수행
        else:
            for hangul_letter in elem:
                result+= letter(hangul_letter)

    return result


print(main("양"))
```

[Figure 25] 텍스트-점자 변환 코드

3.3.1.2 Braille-To-Text

[Figure 26]은 BEE Device 에서 BEE Application 으로 입력한 점자 정보를 문자로 변환하기 위한 로직의 구현 부이다. Text-To-Braille 코드를 기본 골자로 사용하고, 점자 입력 시 적용되는 추가적인 규칙은 별도로 추가해주는 방식으로 구현이 가능하다.

BEE Device 에서 BEE Application 으로 점자 정보를 전송할 시, 10 진수 형태의 2 자리수 문자열로 전송이 된다. 2 자리의 문자열이 하나의 철자 단위의 점자정보를 의미하고, '!'를 Seperator 로 사용하여 단어 단위의 점자정보를 구분한다. 이 단어 단위의 점자정보가 모여서 전체 점자정보가 된다.

점자정보를 text 로 바꿀 시에는 2 종약자의 유무를 판단한 후, 초성의 존재 여부와 단자음, 쌍자음, 1 종약자의 순서로 확인한다. 모음 또한 단모음과 이중모음을 구분하여 확인하고, 종성이 존재 여부를 확인한다. 모든 확인 작업이 끝나면 글자단위로 합치고, 이를 반복하여 모든 글자를 문장 단위로 합친 후 출력한다.

```
import hgtk

MATCH_H2B_CHO = {
    '04':'ㄱ',   '36':'ㄴ',   '20':'ㄷ',   '02':'ㄹ',
    '34':'ㅁ',   '06':'ㅂ',   '01':'ㅅ',   #'':'ㅇ',
    '05':'ㅈ',   '03':'ㅊ',   '52':'ㅋ',   '50':'ㅌ',
    '38':'ㅍ',   '22':'ㅎ',

    '0104':'ㄲ', '0120':'ㄸ', '0106':'ㅃ', '0101':'ㅆ',
    '0105':'ㅉ',
}

MATCH_H2B_JOONG = {
    '49':'ㅏ',   '14':'ㅑ',   '28':'ㅓ',   '35':'ㅕ',
    '41':'ㅗ',   '13':'ㅛ',   '44':'ㅜ',   '37':'ㅠ',
    '21':'ㅡ',   '42':'ㅣ',   '58':'ㅐ',   '46':'ㅔ',
    '1458':'ㅒ', '12':'ㅖ',   '57':'ㅘ',   '5758':'ㅙ',
    '47':'ㅚ',   '60':'ㅝ',   '6058':'ㅞ', '4458':'ㅟ',
    '23':'ㅢ',
}

MATCH_H2B_JONG = {
    '32':'ㄱ',   '18':'ㄴ',   '10':'ㄷ',   '16':'ㄹ',
    '17':'ㅁ',   '48':'ㅂ',   '08':'ㅅ',   '27':'ㅇ',
    '40':'ㅈ',   '24':'ㅊ',   '26':'ㅋ',   '25':'ㅌ',
    '19':'ㅍ',   '11':'ㅎ',   '3232':'ㄲ', '3208':'ㄳ',
    '1840':'ㄵ', '1811':'ㄶ', '1632':'ㄺ', '1617':'ㄻ',
    '1648':'ㄼ', '1608':'ㄽ', '1625':'ㄾ', '1611':'ㄿ',
    '4808':'ㅀ', '12':'ㅆ',
}
--- ... ---
fastest_handler= {'3228':'그래서',
                  '3236':'그러나',
                  '3218':'그러면',
                  '3217':'그러므로',
                  '3246':'그런데',
                  '3241':'그리고',
                  '3235':'그리하여',
                  '0728':'것'}


second_handler= {
    '53':'가',   '36':'나',   '20':'다',   '34':'마',
    '06':'바',   '56':'사',   '05':'자',   '52':'카',
    '50':'타',   '38':'파',   '22':'하',   '39':'억',
    '31':'언',   '30':'얼',   '33':'연',   '51':'열',
    '55':'영',   '45':'옥',   '59':'온',   '63':'옹',
    '54':'운',   '61':'울',   '43':'은',   '29':'을',
    '62':'인',
}
```

```
def compose(braille):
    result=''
    # 맨 마지막에 들어온 '!' 제거
    if(braille[-1]=='!'):
        braille=braille[:-1]
    # !를 기준으로 점자정보 쪼개기
    braille=braille.split('!')

    for character in braille:
        text=''
        letter = []
        # 점자 정보가 2종약자
        if(character in fastest_handler):
            text=fastest_handler[character]
        # 점자 정보에 초성이 포함됨 (초성이 'ㅇ'이 아님)
        elif(character[0:2] in MATCH_H2B_CHO):
            # 초성이 쌍자음
            if(character[0:4] in MATCH_H2B_CHO):
                letter.append(MATCH_H2B_CHO[character[0:4]])
                character=character[4:]
            # 초성이 단자음
            else:
                letter.append(MATCH_H2B_CHO[character[0:2]])
                character=character[2:]
            # 중성과 종성이 1종 약자
            if(character[0:2] in second_handler):
                tmp=hgtk.text.decompose(second_handler[character])
                tmp = ㅟ
                    tmp.replace(hgtk.text.DEFAULT_COMPOSE_CODE, '')
                letter.append(tmp[1])
                letter.append(tmp[2])
            # 중성이 따로 있는 경우(초성+종성 1종약자가 아닌 경우)
            elif(character[0:2] in MATCH_H2B_JOONG):
                # 중성이 겹모음인 경우
                if(character[0:4] in MATCH_H2B_JOONG):
                    letter.append(MATCH_H2B_JOONG[character[0:4]])
                    character=character[4:]

                # 중성이 단모음인 경우
                else:
                    letter.append(MATCH_H2B_JOONG[character[0:2]])
                    character=character[2:]
                # 종성이 있는 경우
                if(character in MATCH_H2B_JONG):
                    letter.append(MATCH_H2B_JONG[character])
            # 중성이 따로 없음 (초성 + 중성 1종약자)
            else:
                letter.append('ㅏ')
                # 종성이 있는 경우
                if(character in MATCH_H2B_JONG):
                    letter.append(MATCH_H2B_JONG[character])

        # 1종약자로 시작되는 경우
        elif(character[0:2] in second_handler):
            tmp = hgtk.text.decompose(second_handler[character[0:2]])
            tmp = ㅟ
                tmp.replace(hgtk.text.DEFAULT_COMPOSE_CODE, '')
```

```
            # 1종약자가 '가'와 '사'인 경우
            if((character[0:2] == '53')or(character[0:2] == '56')):
                letter.append(tmp[0])
                letter.append(tmp[1])
                character=character[2:]
            # 그 외의 1종약자인 경우
            else:
                letter.append(tmp[0])
                letter.append(tmp[1])
                letter.append(tmp[2])

            if(character in MATCH_H2B_JONG):
                letter.append(MATCH_H2B_JONG[character])
        # 점자의 초성이 'ㅇ'인 경우
        else:
            if(character[0:2] in second_handler):
                text=second_handler[character]
            elif(character[0:2] in MATCH_H2B_JOONG):
                letter.append('ㅇ')

                if(character[0:4] in MATCH_H2B_JOONG):
                    letter.append(MATCH_H2B_JOONG[character[0:4]])
                    character=character[4:]
                else:
                    letter.append(MATCH_H2B_JOONG[character[0:2]])
                    character=character[2:]

                if(character in MATCH_H2B_JONG):
                    letter.append(MATCH_H2B_JONG[character])
        #letter=(",".join(repr(e) for e in letter))

        if(character=='00'):
            text=' '
        if(text==''):
            if(int(len(letter))==3):
                text=hgtk.letter.compose(letter[0],letter[1],letter[2])
            else:
                text=hgtk.letter.compose(letter[0],letter[1])

        result+=text

    return result
```

[Figure 26] 텍스트-점자 변환 코드

3.3.2 BEE Database Server

3.3.2.1 사용자 회원가입 모듈

모바일 App 화면 내에서 회원가입 기능을 통해 입력된 사용자 정보(User ID, Password, Gender, Email)를 Database 에 저장하는 기능을 수행한다.

먼저 AWS RDS 주소와 User, Password, Database name 정보의 입력을 통해 데이터베이스 접속을 요청한다. 다음으로 어플리케이션이 Post 메서드를 사용하여 얻은 정보를 읽어와 해당 정보를 바탕으로 query 를 활용한다. query 문에서 회원 가입에 대한 결과 값을 반환하는데 이 때 회원가입에 성공했을 경우, json 객체 내 "success" 키에 true 값을 반환해주게 된다.

```
// 회원 가입을 위한 function
app.post('/user/register', function (req, res) {
    // 모바일 App에서 넘어온 Request body에서 정보 추출
    var id = req.body.userID;
    var pw = req.body.userPassword;
    var gender = req.body.userGender;
    var email = req.body.userEmail;

    // SQL 생성
    var sql = 'INSERT INTO USER (userID, userPassword, userGender, UserEmail) VALUES (?, ?, ?, ?)';
    // SQL의 '?' 부에 들어갈 변수 리스트 생성
    var params = [id, pw, gender, email];
    console.log(sql);

    // SQL 수행
    connection.query(sql, params, function (err, result) {
        if (err){
            console.log(err);
        }
        // 성공적으로 회원가입에 성공하는 경우
        else {
            res.json({
                success: true
            })
        }
    });
});
```

[Figure 27] bee_db_handler 내의 사용자 회원가입 부분

3.3.2.2 중복 사용자 확인 모듈

App 화면 내에서 회원가입 기능을 통해 입력된 사용자 정보(User ID)와 중복된 정보가 Database 에 존재할 시, 회원 가입 불가를 판단하고 false 값을 반환해주는 기능을 수행한다.

앞선 회원가입과 같은 방법으로 먼저 AWS RDS 주소와 User, Password, Database name 정보로 데이터베이스에 접속을 요청한다. 어플리케이션이 POST 메서드를 사용하여 얻은 사용자 ID 값을 가져와 해당 ID 값으로 query 를 활용한다. SELECT 문을 생성해 데이터베이스에 사용자가 입력한 ID 와 중복된 row 가 존재하는지 확인한다. 결과 값이 데이터베이스에 존재 시에 json 객체 내

"success" 키에 false 를 반환하여 해당 User ID 가 사용불가라는 것을 어플리케이션 사용자에게 알린다.

```
// 아이디 중복 확인을 위한 function
app.post('/user/validate', function (req, res) {
    // 모바일 App에서 넘어온 Request body에서 정보 추출
    var id = req.body.userID;

    // SQL 생성
    var sql = 'SELECT userID FROM USER WHERE userID = ?';
    console.log(sql);

    // SQL 수행
    connection.query(sql, id, function (err, result) {
        if (err)
            console.log(err);
        else {
            // 사용자가 입력한 ID가 데이터베이스에 이미 존재하는 경우
            if (result.length === 1) {
                res.json({
                    success: false
                });
            }
            // 사용자가 입력한 ID가 데이터베이스에 존재하지 않는 경우
            else {
                res.json({
                    success: true
                })
            }
        }
    });
});
```

[Figure 28] bee_db_handler 내의 아이디 중복 확인 부분

3.3.2.3 사용자 로그인 모듈

App 화면 내에서 로그인 기능을 통해 입력된 사용자 정보(User ID, Password) 가 Database 에 존재하는지 확인하는 로그인 기능을 수행한다.

앞선 방법과 같이, 먼저 AWS RDS 주소와 User, Password, Database name 정보로 데이터베이스에 접속을 요청한다. 어플리케이션이 POST 메서드를 사용하여 얻은 사용자 ID 와 Password 값을 가져와 해당 ID 값으로 query 를 활용한다. Database 에 해당하는 row 가 없는 경우, json 객체 내 "success" 키에 false 를 반환하고 "해당 사용자가 존재하지 않습니다."의 메시지를 출력한다. 해당 ID 의 Password 가 다른 경우, 위의 방식과 같이 false 를 반환하고 "사용자 암호를 다시 확인해주세요."의 메시지를 출력한다. 마지막으로 해당 ID 와 Password 가 같은 경우, json 객체 내 "success" 키에 true 를 반환하고 "로그인에 성공하였습니다."의 메시지를 출력하여 bee 어플리케이션의 기능을 사용할 수 있도록 한다.

```
// 로그인을 위한 function
app.post('/user/login', function (req, res) {
    // 모바일 App에서 넘어온 Request body에서 정보 추출
    var id = req.body.userID;
    var pw = req.body.userPassword;

    // SQL 생성
    var sql = 'SELECT * FROM USER WHERE userID = ?'
    console.log(sql);

    // SQL 수행
    connection.query(sql, id, function (err, result) {
        if (err)
            console.log(err)
        else {
            // 사용자가 입력한 계정에 해당하는 row가 없을 경우
            if (result.length === 0) {
                res.json({
                    success: false,
                    msg: '해당 사용자가 존재하지 않습니다.'
                });
            }
            // 사용자가 입력한 계정과 암호가 불일치 하는 경우
            else if (pw !== result[0].userPassword) {
                res.json({
                    success: false,
                    msg: '사용자 암호를 다시 확인해주세요.'
                });
            }
            // 정상적으로 로그인이 수행된 경우
            else {
                res.json({
                    success: true,
                    msg: '로그인에 성공하였습니다.'
                });
            }
        }
    })
});
```

[Figure 29] bee_db_handler 내의 사용자 로그인 부분

4. 개발환경

시청각장애인 의사소통 보조 시스템 BEE 의 개발에 있어 OS 로는 Ubuntu 가 사용되고, Tool 로는 AWS EC2, AWS Lambda, PyCharm, Android Studio 를 사용한다. 그리고 회원정보를 담기 위한 데이터베이스로는 AWS RDS, Node.js 를 사용한다. 마지막으로 개발 언어는 C, Java, Python 를 사용한다.

4.1 OS (Ubuntu)

우분투는 데비안 GNU/리눅스를 기반으로 만들어졌으며 고유한 데스크탑 환경을 사용하는 리눅스 배포판이다. 개인용, 데스크탑 환경에 최적화되도록 개발되고 있고, 리눅스의 특징을 그대로 물려 받아 자유 소프트웨어에 기반하기 때문에 누구나 무료로 사용이 가능하다.

4.2 Tool & Utility

4.2.1 AWS EC2

Amazon Elastic Computer Cloud (Amazon EC2)는 AWS 클라우드에서 확장식 컴퓨팅을 제공한다. 개발자가 더 쉽게 웹 규모의 클라우드 컴퓨팅 작업을 할 수 있게 설계됐으며, 간편하게 필요한 용량을 얻고 구성할 수 있다. 또한 새로운 서버 인스턴스를 획득하고 부팅하는데 필요한 시간이 몇 분 걸리지 않으므로 요구 사항 변화에 신속하게 대처가 가능하다.

4.2.2 AWS Lambda

AWS Lambda 는 서버에 대한 걱정 없이 코드 실행이 가능하고 사용한 컴퓨팅 시간에 대해서만 비용을 지불하는 서버리스 코드이다. Lambda 로 작성한 코드는 다른 AWS 서비스에서 코드를 자동으로 Trigger 하도록 설정하거나 웹 또는 모바일 앱에서 직접 코드를 호출할 수 있다.

4.2.3 PyCharm

PyCharm 은 JetBrains 에서 제작한 Python 용 IDE 이다. 코드를 작성하고 수정할 수 있는 Editor 이며, IntelliJ IDEA 에 기반을 두고 있다. 현용 Python 개발 툴 중에서는 가장 높은 완성도를 지니고 있어 널리 사용되고 있다.

4.2.4 Android Studio

Android Studio 는 Android 개발을 위한 공식 IDE 이다. 때문에 풍부한 코드 편집, 디버깅, 테스트 및 프로파일링 도구를 비롯한 맞춤형 도구를 Android 개발자에게 제공한다. 해당 IDE 는 IntelliJ IDEA 기반으로, 코딩 및 실행 워크플로에서 가장 빠른 소요 시간을 제공한다. 본 서비스에서는 안드로이드 어플리케이션을 개발하기 위한 언어로 Java 를 사용한다.

4.2.5 Database - AWS RDS

AWS RDS 는 클라우드에서 관계형 데이터베이스를 쉽게 설치, 운영 및 확장할 수 있는 웹 서비스이다. RDS 는 표준 관계형 데이터베이스를 위한 경제적이고 크기 조절이 가능한 용량을 제공하고, 공통 데이터베이스 관리 작업을 한다. 또한 여러 데이터베이스 인스턴스 유형을 제공하며, 본 프로젝트에서는 MySQL 을 사용한다.

4.2.6 Node.js

Node.js 는 확장성 있는 네트워크 애플리케이션(특히 서버 사이드) 개발에 사용되는 소프트웨어 플랫폼이다. 작성 언어로 자바스크립트를 활용하며 Non-blocking I/O 와 단일 스레드 이벤트 루프를 통한 높은 처리 성능을 가지고 있다.

4.3 Programming Language

4.3.1 C

본 프로젝트의 아두이노 동작을 위해서는 C 언어 (C++) 기반을 사용한다. C 컴파일러는 avr-gcc 를 사용하기 때문에 avr-gcc 가 제공하는 많은 C 언어의 표준라이브러리 함수를 사용할 수 있다.

4.3.2 Java

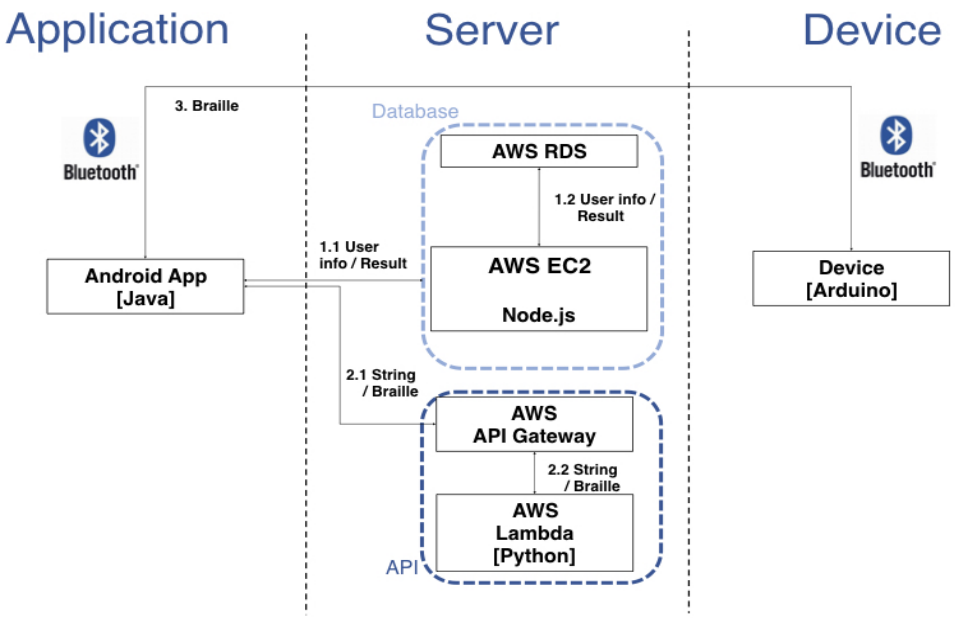
본 프로젝트의 안드로이드 개발에 있어서는 프로그래밍 언어로 Java 를 사용한다. 자바는 객체 지향 프로그래밍 언어이고, 웹 어플리케이션 및 모바일 기기용 소프트웨어 개발에 널리 사용된다.

4.3.3 Python

본 프로젝트에서 점자-문자 간 변환 로직을 수행하는 코드는 AWS Lambda 에 Python 언어로 등록된다. Python 은 간단하고 쉬운 문법과 빠른 실행 속도로 개발 기간을 단축시킬 수 있다.

5. 기능 동작

5.1 Dataflow

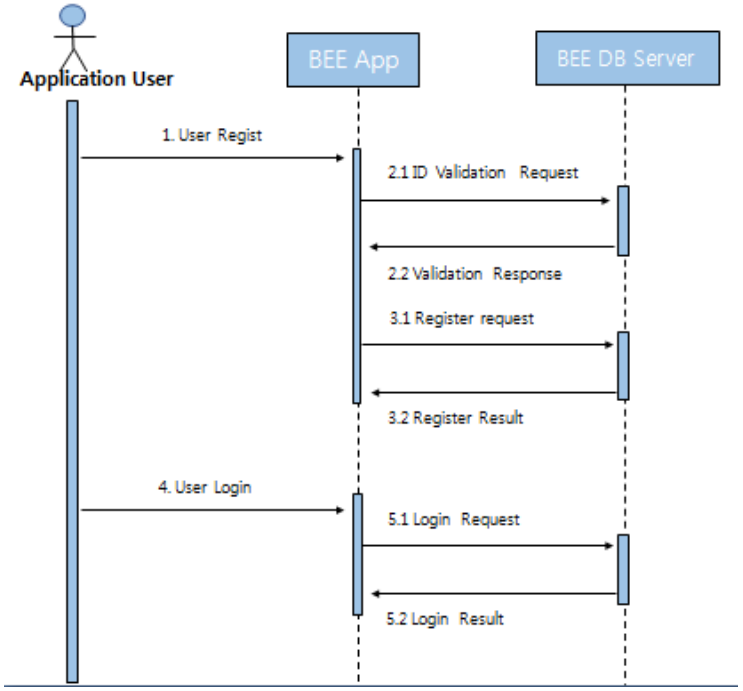


[Figure 30] BEE 서비스 데이터 플로우

시청각장애인 의사소통 보조시스템 BEE 의 전체적인 Data Flow 는 [Figure 30]와 같다. End-to-End 방식의 서비스이기 때문에 양 끝 단에는 각각 Application 과 Device 가 위치하고, 둘의 통신을 돕는 API Server 와 Database Server 가 그 사이에 위치해 데이터 정보의 전송을 수행한다.

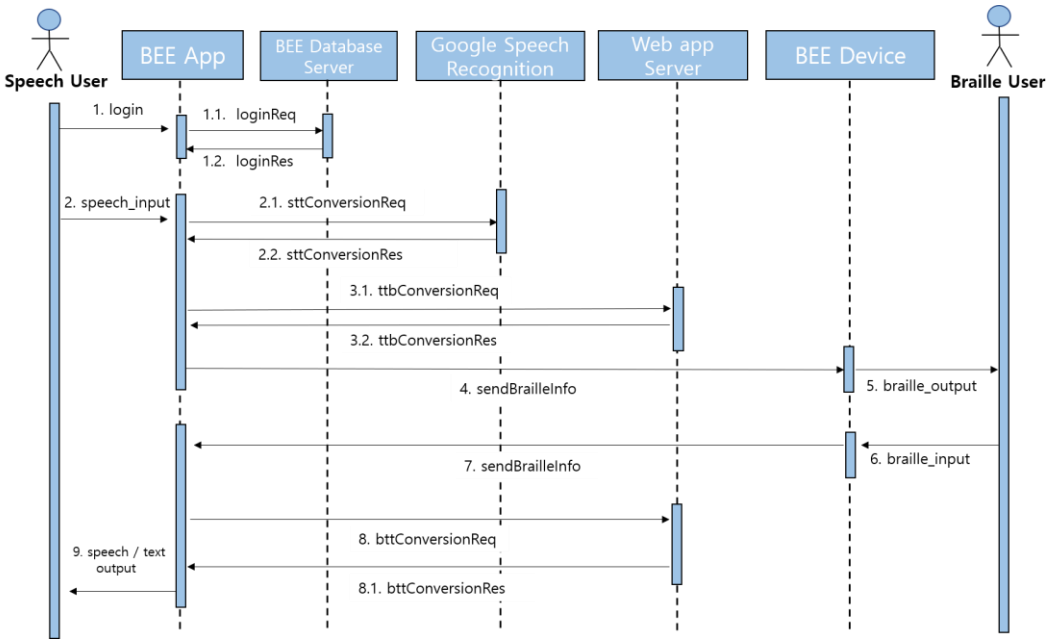
5.2 Sequence Diagram

5.2.1 Sign up & Log-in sequence



[Figure31] 사용자가 Application 에서 회원가입/로그인 기능을 수행하는 시퀀스 다이어그램

5.2.2 BEE Application – BEE Device communication sequence



[Figure 32] Application – Device 간 소통을 위한 시퀀스 다이어그램

6. 자체 테스트 방안

BEE 서비스의 자체 테스트는 다음과 같은 사항을 토대로 수행될 것이다. 각 단계별 시험 방법 및 절차는 아래와 같다.

6.1 어플리케이션 자체 시험

- Android 어플리케이션의 다운로드가 잘 수행되는지 확인
- 지정한 버전 이상의 Android 운영체제에서 동작하는지 확인
- 어플리케이션의 UI 화면이 깨지지 않는지 확인

6.2 어플리케이션과 웹 서버 간의 시험

- 회원 가입을 위해 사용자가 어플리케이션 폼을 이용해 입력한 정보를 웹 서버가 연동된 데이터베이스에 저장할 수 있는지 확인
- 데이터베이스에 이미 저장되어 있는 정보를 사용자가 입력하여 로그인을 요청했을 때, 서버가 데이터를 조회하여 해당 데이터가 없다면 로그인을 거절하고, 조회한 정보와 같은 데이터가 있으면 로그인을 처리하는지 확인
- 데이터베이스에 이미 저장되어 있는 ID 를 사용자가 입력하여 회원 가입을 요청했을 때, 서버가 데이터를 조회하여 해당 데이터가 있다면 가입을 거절하고, 없다면 가입을 처리하는지 확인
- 로그인 상태가 계속 유지될 수 있는 지 확인

6.3 어플리케이션과 디바이스 간의 시험

- 블루투스 통신을 통해 어플리케이션과 디바이스를 서로 연동될 수 있는지 확인
- 블루투스 통신의 연결이 디바이스 점자 입력 시에 끊김이 생기지 않는 지 확인
- 어플리케이션에서 송신한 정보가 디바이스에서 정확히 출력이 되는지 확인
- 디바이스에서 송신한 정보가 어플리케이션에서 정확히 출력이 되는지 확인

7. 향후 프로젝트 적용 방안

- 기존 보조기기보다 가격을 대폭 낮춰서 구매력이 낮은 시 · 청각장애인들에게도 접근성을 높여 근본적인 의사소통 문제를 해결하고자 한다.
- 음성, 점자, 텍스트 모두 변환 가능해 장애인과 장애인간의 의사소통 뿐만 아니라 장애인과 비장애인과의 소통에도 활용이 가능하다.
- 기존에 사용이 어려웠던 스마트 디바이스(e.g. 노트북, 스마트폰, etc.)에 본 프로젝트의 기술을 추가적으로 탑재하여 장애인들이 편의를 누릴 수 있게 한다.

8. 기대 효과

시청각복합장애인은 촉수화와 점화를 통해 서로 소통한다. 그러나 촉수화와 점화는 한국어로 정해진 체계가 없고, 한국에서 구사할 수 있는 사람도 많지 않아 의사소통에 차질이 있다.

비장애인과의 의사소통의 경우 보조기기 없이는 불가능해 보조기기의 사용이 불가피한데, 기존의 시청각장애 보조기기의 경우 가격대가 고가에 형성 되어있고 수리 비용도 만만치 않다. 본 프로젝트에서는 시중에 판매되는 시청각장애 보조기기보다 비교적 저렴한 가격에 공급할 수 있는 기기를 개발 예정이다.

기존에 판매되는 기기는 음성을 직접 텍스트화 해야 하므로 시 · 청각장애인과 비장애인간의 즉각적인 소통에 한계가 있다. 본 프로젝트는 이와 같은 문제점을 개선하여 Application 을 통해 음성 데이터를 입력 받고 즉시 출력하므로 실시간 소통이 가능하다.

궁극적으로 실시간 소통이 가능한 저가형 양방향 의사소통 보조 시스템을 구축하여 시 · 청각장애인들의 근본적인 불편 해소와 원활한 의사소통 환경을 구축하는데 기여하는 데 목적이 있다.

더 나아가 본 프로젝트의 프로토타입은 향후 소형화를 통해 스마트 device(e.g.노트북, 스마트폰, etc.)에 기술을 추가적으로 탑재하여 장애인들이 더 많은 생활 편의를 누릴 수 있을 것으로 기대한다.

9. 프로젝트 세부추진계획 및 세부일정

주차 / 내용	4 월				5 월				6 월		
	1	2	3	4	1	2	3	4	1	2	3
아이디어 회의 및 기술 동향 조사											
사업계획서 작성											
개발장비 조사											
요구사항 정의서 작성											
서버 생성											
안드로이드 앱 UI 설계											
안드로이드 앱 UI Activity 설계											
안드로이드와 Google Speech API 연동											
서버와 안드로이드 앱 연동											
Arduino 스케치 작성											
Arduino 점자 입출력부 제작											
Arduino 와 안드로이드 앱 Bluetooth 연동											
웹 서버 내 점자-텍스트 변환 로직 구현											
안드로이드 앱 세부 기능 구현											
통합 테스트 및 오류 수정											
최종 보고서 작성											

[Table 3] 프로젝트 세부일정