기초설계
소스코드 분석서

(기초설계 최종 프로젝트 제출물)

제출일 : 2020/12/17(목)
분 반 : 005

학 과 : 전자정보통신공학과
학 번 : 19010836
이 름 : 조 영 우

<목 차>

0. 기능 설명 및 필요 어플, 코드 전문

0.1. 기능 설명

해당 코드는 총 4개의 To-Do List를 블루투스를 통해 한 줄씩 받아 저장하고, 이를 장치 내에서 확인/수정/삭제 할 수 있는 기능을 가진 포스트잇 형태의 장치이다. 그리고 받은 4개의 List를 관리하기 위해 스크롤/내용 삭제/커서 이동기능/화면 온오프/문자열저장관리 등 부가적인 기능들이 들어가 있다.

0.2. 필요 어플 및 초기 설정

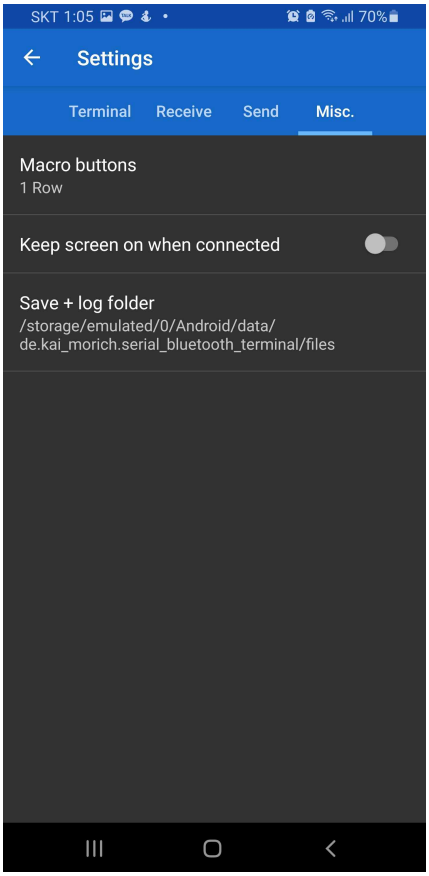
코드 분석과 큰 상관은 없는 파트이지만, 해당 장치는 블루투스로 문자를 주고받을 수 있는 애플리케이션을 사용하는 것이 필수적이기에 이 파트를 넣었다. 안드로이드의 경우 이 어플을 통해 블루투스 송신을 진행하는 것이 편하다.

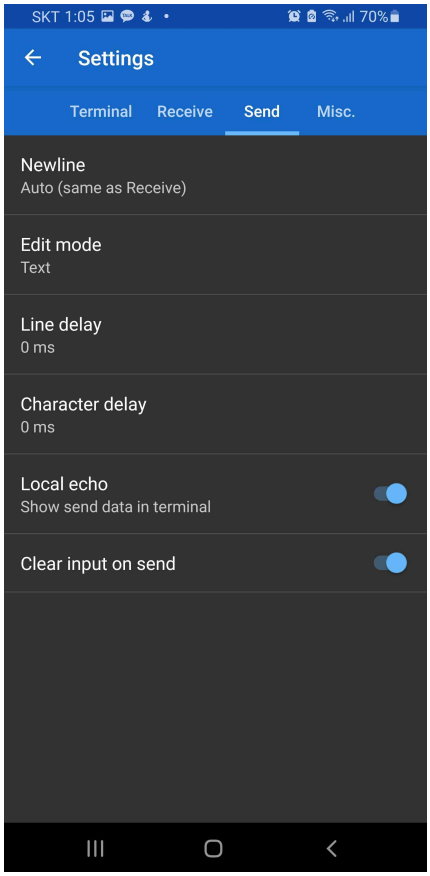
https://play.google.com/store/apps/details?id=de.kai_morich.serial_bluetooth_terminal

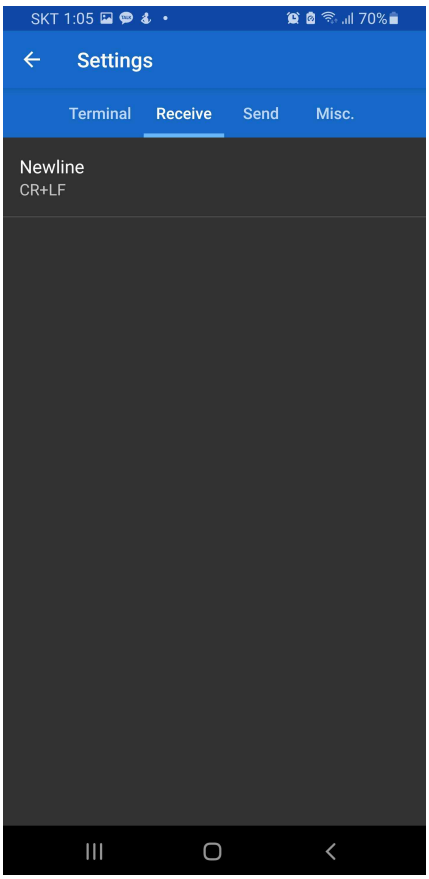
해당 어플을 다운받은 후에 연결 버튼을 눌러 'Kocoafab_BLE'라고 적힌 항목을 클릭하여 연결하면 블루투스 송신 기능을 사용할 수 있다.

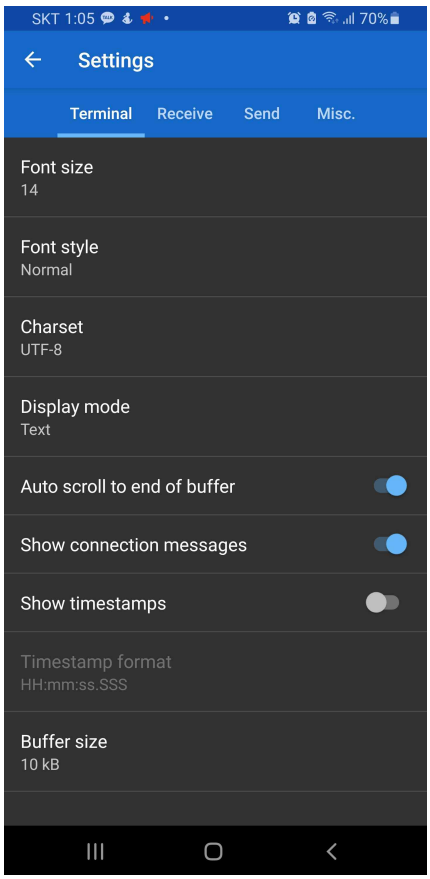
조금 더 쾌적한 사용을 위해 애플리케이션 내 설정을 조율하고 싶다면, 어플 초기 화면에서 [왼쪽으로 슬라이드] - [Settings]에 접속하여, 각 세부사항들을 아래 보이는 [사진]처럼 설정하면 된다.

(사진의 크기가 큰 관계로, 아래 페이지에서 사진을 확인할 수 있습니다.)


SKT 1:05
70%
Settings
Terminal
Receive
Send
Misc.
Macro buttons
1 Row
Keep screen on when connected
Save + log folder
/storage/emulated/0/Android/data/
de.kai_morich.serial_bluetooth_terminal/files


SKT 1:05
70%
Settings
Terminal
Receive
Send
Misc.
Newline
Auto (same as Receive)
Edit mode
Text
Line delay
0 ms
Character delay
0 ms
Local echo
Show send data in terminal
Clear input on send


SKT 1:05
70%
Settings
Terminal
Receive
Send
Misc.
Newline
CR+LF


SKT 1:05
70%
Settings
Terminal
Receive
Send
Misc.
Font size
14
Font style
Normal
Charset
UTF-8
Display mode
Text
Auto scroll to end of buffer
Show connection messages
Show timestamps
Timestamp format
HH:mm:ss.SSS
Buffer size
10 kB

0.3. 코드 전문

```
#include <SoftwareSerial.h> //블루투스 및 시리얼 통신 라이브러리 호출
#include <LiquidCrystal_I2C.h>//I2C LCD 제어 라이브러리 호출 (1)
#include <Wire.h>//I2C LCD 제어 라이브러리 호출 (2)

#define key1 9//1번 키 할당 핀
#define key2 10//2번 키 할당 핀
#define key3 11//3번 키 할당 핀
#define key4 12//4번 키 할당 핀

LiquidCrystal_I2C lcd(0x27, 20, 4);//초기 I2C LCD 선언, 0x27, 행 20, 열 4

SoftwareSerial BTSerial(4, 5);  //시리얼 통신을 위한 객체선언
String BTString=""; //받는 문자열
String lcdDisplay[4]="";//저장 문자열
String eraser="                    ";//lcd에 떠있는 문자열을 제거하기 위한, 말 그대로
eraser 역할의 문자열
int nowCol = 0;// 현재 선택 중인 열의 인덱스 값
int onToggle = 1; // 화면 on/off를 관리하는 변수

void setup() {
  Serial.begin(9600);    //시리얼모니터
  BTSerial.begin(9600); //블루투스 시리얼 개방

  lcd.begin();//LCD 시작 및 초기화
  lcd.backlight();//LCD 백라이트 켜기

  pinMode(key1, INPUT_PULLUP);//key1 input 설정
  pinMode(key2, INPUT_PULLUP);//key2 input 설정
  pinMode(key3, INPUT_PULLUP);//key3 input 설정
  pinMode(key4, INPUT_PULLUP);//key4 input 설정

  lcd.setCursor(1,1);
  lcd.print("***Initializing***");
  lcd.setCursor(1,2);
  lcd.print("***Please  Wait***");
  delay(500);
  lcd.clear();
```

```
  lcd.cursor();//LCD 커서 활성화
  lcd.setCursor(0, nowCol);//LCD 초기 선택열(0번 열)에 커서 고정
}

void loop() {

  String temp;//임시 저장용 문자열 temp 선언
  int displayLength = 0;//글자 수를 저장하는 데 쓰이는 정수형 변수, 초기값은 0으로 설정

//^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  if(lcdDisplay[nowCol].length() > 20){//만약 선택한 줄의 텍스트가 20자 이상이라면
    for(int i = 0; i < lcdDisplay[nowCol].length() - 19; i++){// (텍스트의 길이 - 19) 한만큼의 아래 반복문을 시행한다.
      temp = lcdDisplay[nowCol].substring(i, i+20);//temp 문자열에 20자씩 저장해서
      lcd.setCursor(0,nowCol);//커서를 정렬하고 나서,
      for(int j = 0; j < 13; j++){//조건에 해당하지만 않는다면, 650만큼을 delay하게 된다.

        if(digitalRead(key1)!=HIGH || digitalRead(key2)!=HIGH || digitalRead(key3)!=HIGH || digitalRead(key4)!=HIGH || BTSerial.available()){//조건 : 만약 어느 키든 눌리거나 블루투스를 통해 전송된 값이 있다면
          temp = lcdDisplay[nowCol].substring(0, 20);//temp 문자열에 도로 초기 글자값을 저장하고,
          lcd.setCursor(0,nowCol);//커서를 정렬한 후에
          lcd.print(temp);//초기화면 출력한다.
          lcd.setCursor(0,nowCol);//커서를 정렬한 후에
          goto keyControl;//그리고 중첩 반복문에서 탈출하여 keyControl로 바로 이동한다.
        }//조건 끝

        lcd.setCursor(0,nowCol);//커서를 정렬한 후에
        delay(50);//단위 delay는 50이다.

      }//delay 반복문 끝

      lcd.print(temp);//임시 저장 문자열을 출력

      if(i == lcdDisplay[nowCol].length() - 19 - 1){//'글자 스크롤 마지막 도달 시' : 만약 원래 문자열의 마지막 글자가 보인다면,
        lcd.setCursor(0,nowCol);//문자열을 재 정렬하고
```

```
        for(int j = 0; j < 30; j++){//조건에 해당하지만 않는다면, 3000만큼을 delay하게 된다.
          if(digitalRead(key1)!=HIGH || digitalRead(key2)!=HIGH || digitalRead(key3)!=HIGH || digitalRead(key4)!=HIGH || BTSerial.available()){//조건 : 만약 어느 키든 눌리거나 블루투스를 통해 전송된 값이 있다면
            temp = lcdDisplay[nowCol].substring(0, 20);//temp 문자열에 도로 초기 글자 값을 저장하고,
            lcd.setCursor(0,nowCol);//커서를 정렬한 후에
            lcd.print(temp);//초기화면 출력한다.
            lcd.setCursor(0,nowCol);//커서를 정렬한 후에
            goto keyControl;//그리고 중첩 반복문에서 탈출하여 keyControl로 바로 이동한다.
          }//조건 끝

          lcd.setCursor(0,nowCol);//커서를 정렬한 후에
          delay(50);//단위 delay는 50이다.

        }//delay 반복문 끝

        //모든 절차가 방해없이 잘 끝났다면,
        temp = lcdDisplay[nowCol].substring(0, 20);//temp 문자열에 초기 글자 값을 저장하여
        lcd.print(temp);//초기화면 출력한다.
        lcd.setCursor(0,nowCol);//재출력을 위하여 커서를 원위치에 둔다.

      }//'글자 스크롤 마지막 도달 시' 끝

    }// '(텍스트의 길이 - 19) 한 만큼의 아래 반복문을 시행한다.' 끝

  }//'20자 이상 글자 스크롤 알고리즘' 끝

//*************************************************************************************

//^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

  keyControl://'20자 이상 글자 스크롤 알고리즘'에서 탈출할 경우 goto 되는 시점
```

```
  //버튼 눌림 여부에 따른 동작 코드 시작
  int key1S = digitalRead(key1);//key1의 pressed 여부를 읽음
  int key2S = digitalRead(key2);//key2의 pressed 여부를 읽음
  int key3S = digitalRead(key3);//key3의 pressed 여부를 읽음
  int key4S = digitalRead(key4);//key4의 pressed 여부를 읽음

  if(!key1S){//1번 키가 인식되면

    if(nowCol != 0 && nowCol > 0){//현재 열이 0이 아니고 0보다 크다면
      nowCol--;//현재열 값 줄이기
      lcd.setCursor(0, nowCol);//줄인 현재열 값에 맞게 커서 설정



    }
    delay(300);//원활한 동작을 위한 딜레이
  }

  else if(!key2S){//2번 키가 인식되면

    if(nowCol != 3 && nowCol < 3){ //현재 열이 3이 아니고 3보다 작다면
      nowCol++;//현재열 값 늘리기
      lcd.setCursor(0, nowCol);//늘린 현재열 값에 맞게 커서 설정
    }
    delay(300);//원활한 동작을 위한 딜레이
  }

  else if(!key3S){//3번 키가 인식되면
    lcdDisplay[nowCol]="                    ";//현재 열에 저장된 문자열의 데이터를 초
기화
    lcd.print(lcdDisplay[nowCol]);//초기화한 값을 출력
    lcd.setCursor(0,nowCol);//현재열 값에 맞게 커서 설정
    lcdDisplay[nowCol]="";//현재 열에 저장된 문자열의 데이터를 재초기화
    delay(300);//원활한 동작을 위한 딜레이
  }

  else if(!key4S){//4번 키가 인식 되면

    if(onToggle == 1){//onToggle 값이 1일 경우
      lcd.noBacklight();//lcd 백라이트 끄고
      lcd.noDisplay();//lcd 디스플레이도 꺼서 '화면 꺼짐' 상태 만듬
```

```
      onToggle = 0;//onToggle 값을 0으로 설정
    }
    else{//onToggle 값이 0일 경우
      lcd.backlight();//lcd 백라이트 켜고
      lcd.display();//lcd 디스플레이도 켜서 '화면 켜짐' 상태 만듬
      onToggle = 1;//onToggle 값을 0으로 설정
    }
    delay(1000);//delay 값은 다른 버튼보다 더 긴 1000으로 설정
  }//버튼 눌림 여부에 따른 동작 코드 끝
//*************************************************************************
************

//^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
^^^^^^^^^^^^
  while(BTSerial.available())  //BTSerial에 전송된 값이 있으면
  {
    char BTChar = (char)BTSerial.read();  //BTSerial int 값을 char 형식으로 변환
    if(BTChar=='\n' || BTChar=='\r'){
      break;
    }
    BTString+=BTChar;     //수신되는 문자를 BTString에 모두 붙임 (1바이트씩 전송되는
것을 연결)
    delay(5);              //수신 문자열 끊김 방지
  }
//*************************************************************************
************

//^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
^^^^^^^^^^^^
  if(!BTString.equals(""))  //BTString 값이 있다면
  {
    lcdDisplay[nowCol] = BTString;//BTstring 값을 lcdDisplay에 대입
    //Serial.println(lcdDisplay[nowCol]); //시리얼모니터에 BTString값 출력
    displayLength = lcdDisplay[nowCol].length();//lcdDisplay의 길이 정수 값을
displayLength에 대입

    if(displayLength > 20){// 만약 받은 문자열의 길이가 20을 넘는다면
```

```
      temp = lcdDisplay[nowCol].substring(0, 20);//temp 문자열을 이용하여 인덱스
0~19까지만 문자열을 끊어서
      lcd.print(temp);//그 temp 값을 LCD에 띄움
      temp = "                    ";//사용한 temp 문자열을 초기화
    }
    else{
      lcd.print(lcdDisplay[nowCol]);//20 이하의 경우 별다른 과정 없이 디스플레이
    }

    BTString="";  //BTString 변수값 초기화
    lcd.setCursor(0, nowCol);//현재열 값에 맞게 커서 설정
  }
//*************************************************************************
************

}//loop문 끝
```

1. 헤더

<코드 원문>
#include <SoftwareSerial.h> //블루투스 및 시리얼 통신 라이브러리 호출
#include <LiquidCrystal_I2C.h>//I2C LCD 제어 라이브러리 호출 (1)
#include <Wire.h>//I2C LCD 제어 라이브러리 호출 (2)

<추가 설명>
#include <SoftwareSerial.h> 는 블루투스 및 시리얼 통신 라이브러리 호출을 위해 필요한 헤더이다. 이 작품의 핵심 기능인 블루투스 송수신을 위해 필요하기도 하지만, 종종 오류가 나는 경우 디버깅을 위해 일반 Serial 통신을 사용한다. 디버깅 과정에서는 Serial 함수를 통해 출력 값 등을 시리얼 모니터로 모니터링하여 오류를 잡는 용도로 사용하고, 실사용에서는 BTSerial을 이용해 블루투스 송수신 기능을 구현하고 있다.
#include <LiquidCrystal_I2C.h>는 I2C LCD 제어 라이브러리 호출을 위해 불러온 첫 번째 헤더이다. 작품의 부품 중 I2C LCD를 제어하기 위한 필수적인 함수들을 이 라이브러리에서 모두 불러올 수 있다. 단독으로 사용하는 것은 불가능하고, 후술할 wire.h와 결합하여 다양한 기능들을 구현할 수 있다.
#include <Wire.h>는 I2C LCD 제어 라이브러리 호출을 위해 불러온 두 번째 헤더이다. 이 자체로는 큰 기능을 하지 못하지만, LiquidCrystal_I2C.h를 구동하기 위해 필수적으로 들어가야하는 부가 라이브러리이다.

2. 매크로/초기 설정/전역변수

2.1. 매크로

<코드 원문>
#define key1 9//1번 키 할당 핀
#define key2 10//2번 키 할당 핀
#define key3 11//3번 키 할당 핀
#define key4 12//4번 키 할당 핀

<추가 설명>
전반적인 버튼 설정 및 조작을 위한 매크로 선언 파트이다.
#define key1 9 는 1x4 멤브레인 키보드의 '1번' 이자, '이전' 버튼을 설정하기 위한 매크로이다. define을 붙였기 때문에, 이후 소스코드상의 모든 'key1'을 '9'로 치환해 준다.
#define key2 10 은 1x4 멤브레인 키보드의 '2번' 이자, '다음' 버튼을 설정하기 위한 매크로이다. define을 붙였기 때문에, 이후 소스코드상의 모든 'key2'를 '10'으로 치환해 준다.
#define key3 11 은 1x4 멤브레인 키보드의 '3번' 이자, '삭제' 버튼을 설정하기 위한 매크로이다. define을 붙였기 때문에, 이후 소스코드상의 모든 'key3'을 '11'로 치환해 준다.
#define key4 12 는 1x4 멤브레인 키보드의 '4번' 이자, '화면 전원' 버튼을 설정하기 위한 매크로이다. define을 붙였기 때문에, 이후 소스코드상의 모든 'key4'를 '12'로 치환해 준다.

2.2. 초기 설정

<코드 원문>
LiquidCrystal_I2C lcd(0x27, 20, 4);//초기 I2C LCD 선언, 0x27, 행 20, 열 4

SoftwareSerial BTSerial(4, 5); //시리얼 통신을 위한 객체선언

<추가 설명>
I2C LCD와 Bluetooth 송수신 기능을 사용하기 위한 초기 설정 파트이다.
LiquidCrystal_I2C lcd(0x27, 20, 4);는 초기 I2C LCD 선언해주기 위해 필요한 초기 설정이다. 괄호 안에 해당 LCD에 대한 정보를 담고 있는데, 0x27은 I2C LCD의 주소, 20은 LCD의 행 크기, 4는 LCD의 열 크기를 나타낸다. 이는 이번 설계에서 쓰인 LCD 디스플레이의 정보와 일치한다(주소가 0x27인 20*4 크기의 I2C LCD 디스플레이).
SoftwareSerial BTSerial(4, 5); 블루투스 시리얼 통신을 위한 객체선언이다. TX와 RX 값은 오렌지보드에서 이미 4번 포트와 5번 포트로 할당되어있기에, 부품을 조정하거나 이 구문이 수정될 일은 없다.

2.3. 전역 변수

<코드 원문>
String BTString=""; //받는 문자열
String lcdDisplay[4]="";//저장 문자열
String eraser="   ";//lcd에 떠있는 문자열을 제거하기 위한, 말 그대로 eraser 역할의 문자열
int nowCol = 0;// 현재 선택 중인 열의 인덱스 값
int onToggle = 1; // 화면 on/off를 관리하는 변수

<추가 설명>
본격적인 기능의 시작과 앞서 작품의 기능을 수행하는데 있어 빈번하게 쓰이거나, 많이 쓰이지는 않으나 함수 바깥에서 선언할 필요가 있는 변수들을 전역 변수로 처리한 모습이다
String BTString=""; 은 '블루투스 수신' 때 받아온 데이터들을 String으로 변환할 때 필요한 문자열 변수이다. 이 값이 모두 추려지면, 각 lcdDisplay로 그 값이 옮겨간다.
String lcdDisplay[4]="";은 각 디스플레이 열마다 저장되어있는 값을 의미한다. 화면에 띄우는 글자가 20자가 넘어가도 그 값을 모두 저장하고 있다.
String eraser="   ";은 코드 원문의 주석 설명과 동일하다.
int nowCol = 0;// 현재 선택 중인 열의 인덱스 값으로, lcdDisplay[nowCol]형태로 쓰이며 현재 선택중인 열을 관리 및 조정하는 역할을 하고 있다.
int onToggle = 1; // 화면 on/off를 관리하는 변수로, 4번 키의 기능과 직접적인 연관이 있다.

3. void setup()

<코드 원문>

```
Serial.begin(9600);    //시리얼모니터
BTSerial.begin(9600); //블루투스 시리얼 개방

lcd.begin();//LCD 시작 및 초기화
lcd.backlight();//LCD 백라이트 켜기

pinMode(key1, INPUT_PULLUP);//key1 input 설정
pinMode(key2, INPUT_PULLUP);//key2 input 설정
pinMode(key3, INPUT_PULLUP);//key3 input 설정
pinMode(key4, INPUT_PULLUP);//key4 input 설정
```

<추가 설명>

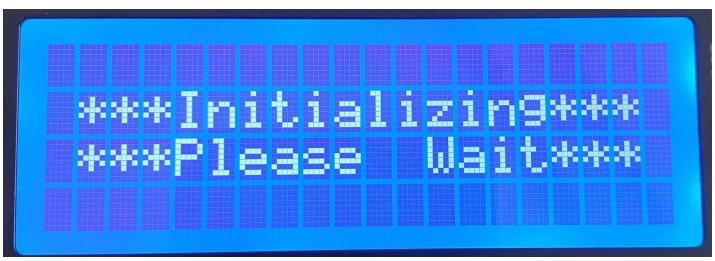
위 <코드 원문>의 주석에 달려있는 설명 그대로의 기능을 수행한다. 시리얼 개방, 블루투스 시리얼 개방, LCD 켜기, 각 버튼의 pinMode 설정 등 초기 1회는 반드시 수행되어야 하는 기능들을 모아놓은 것이다.

3.1. void setup() -초기 화면 설정

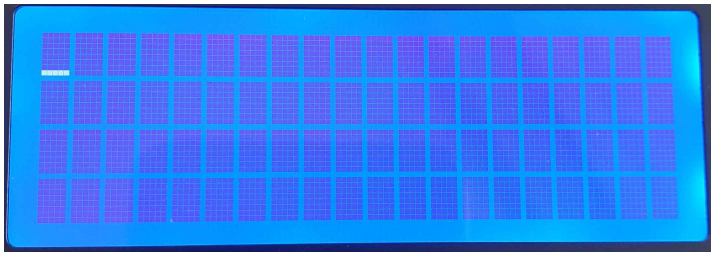
<코드 원문>

```
lcd.setCursor(1,1);
lcd.print("***Initializing***");
lcd.setCursor(1,2);
lcd.print("***Please  Wait***");
delay(500);
lcd.clear();
lcd.cursor();//LCD 커서 활성화
lcd.setCursor(0, nowCol);//LCD 초기 선택열(0번 열)에 커서 고정
```

<추가 설명>



(그림 5) 초기 부팅 화면



(그림 6) 초기 부팅 화면 이후 초기화된 화면

맨 처음 장치에 전원이 들어오거나 RESET된 경우, 가동 가능 상황을 알리고 화면을 초기화하는 코드이다. 우선 (그림 1)의 화면으로 “초기화 중, 기다려 주세요”라는 메시지 창이다. 0.5초(500)동안 띄운 이후 사용자는 장치를 사용할 수 있게 되는데, 이 때 (그림 2)의 화면처럼 디스플레이 초기 세팅까지 마쳐주는 것이 이 코드의 역할이다.

4. void loop()

<코드 원문>
String temp;//임시 저장용 문자열 temp 선언
int displayLength = 0;//글자 수를 저장하는 데 쓰이는 정수형 변수, 초기값은 0으로 설정

<추가 설명>
String temp;는 한 줄에 20자가 넘어가는 글자가 들어올 시, 이를 subString(); 기능을 통해 정확히 20자로 쪼갠 후, 그 값을 저장해주는 역할을 맡고 있다. 20자가 넘어가는 경우 lcdDisplay 배열에 저장되어 있는 원문 값을 올릴 수 없으므로, 쪼개진 data인 temp의 값을 받아 먼저 출력한다.
int displayLength = 0;는 length(); 기능을 통해 가늠한 정수 형태의 글자 수 데이터를 저장하는 데 쓰이는 변수이다. 스크롤이나, temp값 할당 등에 주로 쓰인다. 초기 값은 아무런 글자가 없는 상태와 동일한 0으로 해주었다.

4.1. 20자 이상 스크롤 알고리즘

<코드 원문>
```
if(lcdDisplay[nowCol].length() > 20){//만약 선택한 줄의 텍스트가 20자 이상이라면
    for(int i = 0; i < lcdDisplay[nowCol].length() - 19; i++){// (텍스트의 길이 - 19) 한만큼의 아래 반복문을 시행한다.
      temp = lcdDisplay[nowCol].substring(i, i+20);//temp 문자열에 20자씩 저장해서
      lcd.setCursor(0,nowCol);//커서를 정렬하고 나서,
      for(int j = 0; j < 13; j++){//조건에 해당하지만 않는다면, 650만큼을 delay하게 된다.

        if(digitalRead(key1)!=HIGH || digitalRead(key2)!=HIGH || digitalRead(key3)!=HIGH || digitalRead(key4)!=HIGH || BTSerial.available()){//조건 : 만약 어느 키든 눌리거나 블루투스를 통해 전송된 값이 있다면
          temp = lcdDisplay[nowCol].substring(0, 20);//temp 문자열에 도로 초기 글자 값을 저장하고,
          lcd.setCursor(0,nowCol);//커서를 정렬한 후에
          lcd.print(temp);//초기화면 출력한다.
          lcd.setCursor(0,nowCol);//커서를 정렬한 후에
          goto keyControl;//그리고 중첩 반복문에서 탈출하여 keyControl로 바로 이동한다.
      }//조건 끝

      lcd.setCursor(0,nowCol);//커서를 정렬한 후에
```

```
        delay(50);//단위 delay는 50이다.

      }//delay 반복문 끝

      lcd.print(temp);//임시 저장 문자열을 출력

      if(i == lcdDisplay[nowCol].length() - 19 - 1){//'글자 스크롤 마지막 도달 시' : 만약 원래 문자열의 마지막 글자가 보인다면,
        lcd.setCursor(0,nowCol);//문자열을 재 정렬하고

        for(int j = 0; j < 30; j++){//조건에 해당하지만 않는다면, 3000만큼을 delay하게 된다.
          if(digitalRead(key1)!=HIGH || digitalRead(key2)!=HIGH || digitalRead(key3)!=HIGH || digitalRead(key4)!=HIGH || BTSerial.available()){//조건 : 만약 어느 키든 눌리거나 블루투스를 통해 전송된 값이 있다면
            temp = lcdDisplay[nowCol].substring(0, 20);//temp 문자열에 도로 초기 글자 값을 저장하고,
            lcd.setCursor(0,nowCol);//커서를 정렬한 후에
            lcd.print(temp);//초기화면 출력한다.
            lcd.setCursor(0,nowCol);//커서를 정렬한 후에
            goto keyControl;//그리고 중첩 반복문에서 탈출하여 keyControl로 바로 이동한다.
          }//조건 끝

          lcd.setCursor(0,nowCol);//커서를 정렬한 후에
          delay(50);//단위 delay는 50이다.

        }//delay 반복문 끝

        //모든 절차가 방해없이 잘 끝났다면,
        temp = lcdDisplay[nowCol].substring(0, 20);//temp 문자열에 초기 글자 값을 저장하여
        lcd.print(temp);//초기화면 출력한다.
        lcd.setCursor(0,nowCol);//재출력을 위하여 커서를 원위치에 둔다.

      }//'글자 스크롤 마지막 도달 시' 끝

    }// '(텍스트의 길이 - 19) 한 만큼의 아래 반복문을 시행한다.' 끝

  }//'20자 이상 글자 스크롤 알고리즘' 끝
```

<추가 설명>
코드가 매우 길기에, 각 열마다 세부적인 기능 설명은 코드 원문에 붙어있는 주석을 참고하는 것을 추천한다.
우선 이 코드의 목적은 '만약 현재 선택 중인 줄의 글자 수가 20자가 넘는다면, 글자의 끝까지 보일 때까지 스크롤하고, 마지막 글자가 보이는 경우 잠시 멈췄다가 다시 초기 화면으로 넘어가 스크롤을 재수행한다. 만약 스크롤 중 버튼 입력이나 블루투스로 새 내용을 수신 받는다면, 해당 작업을 멈추고 곧장 keyControl로 goto하게 된다.'이다. 스크롤 시 글자 당 머무는 시간은 650이며 마지막 글자까지 도달할 시 3000을 머물게 된다.
만약 스크롤 중 버튼 입력이나 블루투스로 새 내용을 수신 받는다면, 해당 작업을 멈추고 곧장 keyControl 포인트인 '버튼 눌림 여부에 따른 동작 코드 알고리즘'으로 즉시 이동하여 눌린 키의 정보를 처리하거나 블루투스 값을 수신할 수 있게 만든다. 알고리즘이 매우 독특하여, 20자 이상 스크롤 알고리즘과는 다른 성격을 지닌 독립된 알고리즘으로 분류하여 다음 파트에서 상세히 설명할 것이다.

4.1.1 버튼 입력 및 새로운 내용 수신 시 스크롤 탈출

<코드 원문 1>
```
for(int j = 0; j < 13; j++){//조건에 해당하지만 않는다면, 650만큼을 delay하게 된다.

        if(digitalRead(key1)!=HIGH || digitalRead(key2)!=HIGH || digitalRead(key3)!=HIGH || digitalRead(key4)!=HIGH || BTSerial.available()){//조건 : 만약 어느 키든 눌리거나 블루투스를 통해 전송된 값이 있다면
          temp = lcdDisplay[nowCol].substring(0, 20);//temp 문자열에 도로 초기 글자 값을 저장하고,
          lcd.setCursor(0,nowCol);//커서를 정렬한 후에
          lcd.print(temp);//초기화면 출력한다.
          lcd.setCursor(0,nowCol);//커서를 정렬한 후에
          goto keyControl;//그리고 중첩 반복문에서 탈출하여 keyControl로 바로 이동한다.
        }//조건 끝

        lcd.setCursor(0,nowCol);//커서를 정렬한 후에
        delay(50);//단위 delay는 50이다.

      }//delay 반복문 끝
```

<코드 원문 2>
```
for(int j = 0; j < 30; j++){//조건에 해당하지만 않는다면, 3000만큼을 delay하게 된다.
          if(digitalRead(key1)!=HIGH || digitalRead(key2)!=HIGH || digitalRead(key3)!=HIGH || digitalRead(key4)!=HIGH || BTSerial.available()){//조건 : 만약
```

어느 키든 눌리거나 블루투스를 통해 전송된 값이 있다면
temp = lcdDisplay[nowCol].substring(0, 20);//temp 문자열에 도로 초기 글자 값을 저장하고,
lcd.setCursor(0,nowCol);//커서를 정렬한 후에
lcd.print(temp);//초기화면 출력한다.
lcd.setCursor(0,nowCol);//커서를 정렬한 후에
goto keyControl;//그리고 중첩 반복문에서 탈출하여 keyControl로 바로 이동한다.
}//조건 끝

lcd.setCursor(0,nowCol);//커서를 정렬한 후에
delay(50);//단위 delay는 50이다.

}//delay 반복문 끝

<추가 설명 2>
위 알고리즘의 기능 중 '만약 스크롤 중 버튼 입력이나 블루투스로 새 내용을 수신 받는다면, 해당 작업을 멈추고 곧장 keyControl로 goto하게 된다.'의 작동 원리를 더 상세히 설명하고자 원래의 알고리즘에서 이를 분리하였다. 이도 마찬가지로 코드문 자체가 길어 미시적인 코드 흐름은 원문의 주석을 참고하고, 거시적 작동 방식은 본 설명을 참고하는 것을 권장한다.
<코드 원문 1>과 <코드 원문 2>의 코드는 크게 다르지 않다. 얼마나 delay하는 가에 따라서 맨 처음 나오는 for 반복문의 반복횟수가 달라질 뿐이다. 우선 단위 delay 시간을 50으로 정하였다. 그리고 <코드 원문 1>은 13번 반복하여 총 650의 delay를, <코드 원문 2>는 60번 반복하여 3000의 delay를 구현하였다. 그리고 매 단위 delay 시간마다, 버튼이 눌리거나 새로운 외부입력이 들어왔는가를 지속적으로 if문을 통해 검사하게 한다.
만약 새 버튼이 눌렸거나 새 입력이 들어온다면, 스크롤을 즉시 멈춘 후 해당 줄에 원래 떠 있던 초기 20자의 글자로 다시 해당 줄의 출력을 돌려 놓고, 커서를 해당 줄의 맨 처음으로 재정렬한뒤, goto문을 통해 해당 알고리즘 및 반복문에서 탈출하여 눌린 버튼에 대응하거나 새로운 bluetooth 값을 받을 수 있는 상태로 다시 돌아오게 만든다. 해당 작업은 매 단위 delay 마다 하게 되며, 반복문에서 요구한 반복횟수를 만족할 때 까지 해당 작업을 반복하게 된다.

4.2. 버튼 눌림 여부에 따른 동작 코드 알고리즘

<코드 원문>
keyControl://'20자 이상 글자 스크롤 알고리즘'에서 탈출할 경우 goto 되는 시점

//버튼 눌림 여부에 따른 동작 코드 시작
int key1S = digitalRead(key1);//key1의 pressed 여부를 읽음
int key2S = digitalRead(key2);//key2의 pressed 여부를 읽음

```
  int key3S = digitalRead(key3);//key3의 pressed 여부를 읽음
  int key4S = digitalRead(key4);//key4의 pressed 여부를 읽음

  if(!key1S){//1번 키가 인식되면

    if(nowCol != 0 && nowCol > 0){//현재 열이 0이 아니고 0보다 크다면
      nowCol--;//현재열 값 줄이기
      lcd.setCursor(0, nowCol);//줄인 현재열 값에 맞게 커서 설정



    }
    delay(300);//원활한 동작을 위한 딜레이
  }

  else if(!key2S){//2번 키가 인식되면

    if(nowCol != 3 && nowCol < 3){ //현재 열이 3이 아니고 3보다 작다면
      nowCol++;//현재열 값 늘리기
      lcd.setCursor(0, nowCol);//늘린 현재열 값에 맞게 커서 설정
    }
    delay(300);//원활한 동작을 위한 딜레이
  }

  else if(!key3S){//3번 키가 인식되면
    lcdDisplay[nowCol]="                              ";//현재 열에 저장된 문자열의 데이터를 초
기화
    lcd.print(lcdDisplay[nowCol]);//초기화한 값을 출력
    lcd.setCursor(0,nowCol);//현재열 값에 맞게 커서 설정
    lcdDisplay[nowCol]="";//현재 열에 저장된 문자열의 데이터를 재초기화
    delay(300);//원활한 동작을 위한 딜레이
  }

  else if(!key4S){//4번 키가 인식 되면

    if(onToggle == 1){//onToggle 값이 1일 경우
      lcd.noBacklight();//lcd 백라이트 끄고
      lcd.noDisplay();//lcd 디스플레이도 꺼서 '화면 꺼짐' 상태 만듬
      onToggle = 0;//onToggle 값을 0으로 설정
    }
    else{//onToggle 값이 0일 경우
      lcd.backlight();//lcd 백라이트 켜고
```

```
      lcd.display();//lcd 디스플레이도 켜서 '화면 켜짐' 상태 만듬
      onToggle = 1;//onToggle 값을 0으로 설정
    }
    delay(1000);//delay 값은 다른 버튼보다 더 긴 1000으로 설정
  }//버튼 눌림 여부에 따른 동작 코드 끝
```

<추가 설명>
key1 ~ key4의 INPUT_PULLUP 값을 key1S ~ key4S에 저장하고, 해당 입력 값에 따른 기능을 수행하는 코드이다. 각 버튼마다의 상세 기능은 따로 분류해 설명한다.

4.2.1. key1 : 이전 줄로 이동 (=이전)

<원문 코드>

```
if(!key1S){//1번 키가 인식되면

    if(nowCol != 0 && nowCol > 0){//현재 열이 0이 아니고 0보다 크다면
      nowCol--;//현재열 값 줄이기
      lcd.setCursor(0, nowCol);//줄인 현재열 값에 맞게 커서 설정



    }
    delay(300);//원활한 동작을 위한 딜레이
  }
```

<추가 설명>
key1은 nowCol의 값을 조정하여, 현재 선택하고 있는 줄에서 이전 줄로 (예를 들어, 위에서 3번째 줄에서 -> 위에서 2번째 줄로 이동) 커서를 옮기는 역할을 한다. 당연히 맨 첫 번째 줄에 도달했을 때, 즉 nowCol == 0일 때는 if 문을 통해 nowCol이 음수 값을 갖는 것을 방지하여 더 이상 위로 올라가지 않도록 설정하였고, 당연히 이러한 상황에서 key1을 누르면 아무런 기능을 하지 않는다.

4.2.2. key2 : 다음 줄로 이동 (=다음)

<원문 코드>

```
else if(!key2S){//2번 키가 인식되면

    if(nowCol != 3 && nowCol < 3){ //현재 열이 3이 아니고 3보다 작다면
      nowCol++;//현재열 값 늘리기
```

```
      lcd.setCursor(0, nowCol);//늘린 현재열 값에 맞게 커서 설정
    }
    delay(300);//원활한 동작을 위한 딜레이
  }
```

<추가 설명>
key2은 nowCol의 값을 조정하여, 현재 선택하고 있는 줄에서 다음 줄로 (예를 들어, 위에서 3번째 줄에서 -> 위에서 4번째 줄로 이동) 커서를 옮기는 역할을 한다. 당연히 맨 마지막 줄에 도달했을 때, 즉 nowCol == 3일 때는 if 문을 통해 nowCol이 4 이상의 값을 갖는 것을 방지하여 더 이상 아래로 내려가거나 엉뚱한 값을 갖지 않도록 설정하였고, 당연히 이러한 상황에서 key2을 누르면 아무런 기능을 하지 않는다.

4.2.3. key3 : 해당 줄의 내용 삭제 (=삭제)

<원문 코드>

```
else if(!key3S){//3번 키가 인식되면
    lcdDisplay[nowCol="                    ";//현재 열에 저장된 문자열의 데이터를 초기화
    lcd.print(lcdDisplay[nowCol]);//초기화한 값을 출력
    lcd.setCursor(0,nowCol);//현재열 값에 맞게 커서 설정
    lcdDisplay[nowCol]="";//현재 열에 저장된 문자열의 데이터를 재초기화
    delay(300);//원활한 동작을 위한 딜레이
  }
```

<추가 설명>
key3은 nowCol의 값에서의 lcdDisplay값을 삭제하고, 화면 상에 떠 있는 해당 줄의 정보를 모두 지워버리는 기능을 수행한다. 우선 현재 열에 저장된 문자열의 데이터를 공백만 20개가 있는 글자로 만들어 화면을 모두 지운 후, 커서를 가다듬고 다시 해당 nowCol의 값에서의 lcdDisplay의 모든 문자 데이터를 삭제하는 과정을 거친다.

4.2.4. key4 : 화면 끄기/켜기 기능 (=화면 전원)

<원문 코드>

```
  else if(!key4S){//4번 키가 인식 되면

    if(onToggle == 1){//onToggle 값이 1일 경우
      lcd.noBacklight();//lcd 백라이트 끄고
      lcd.noDisplay();//lcd 디스플레이도 꺼서 '화면 꺼짐' 상태 만듬
```

```
    onToggle = 0;//onToggle 값을 0으로 설정
  }
  else{//onToggle 값이 0일 경우
    lcd.backlight();//lcd 백라이트 켜고
    lcd.display();//lcd 디스플레이도 켜서 '화면 켜짐' 상태 만듬
    onToggle = 1;//onToggle 값을 0으로 설정
  }
  delay(1000);//delay 값은 다른 버튼보다 더 긴 1000으로 설정
}//버튼 눌림 여부에 따른 동작 코드 끝
```

<추가 설명>
key4는 화면을 끄고 켜는 작업을 수행한다. 화면에 있는 정보를 일시적으로 받고 싶지 않거나, 사용하지 않을 경우 배터리 소모량을 최소화하기 위해 화면을 끄는 경우가 있는데, 이 때 전원을 완전히 차단할 필요 없이 이 버튼만 눌러주면 된다. 필요하면 다시 버튼을 눌러 켤 수 있는 토글 방식을 채택하였다. 화면을 끈 상태에서도 수신 및 버튼 조작은 모두 가능하나 일반적으로 화면을 끈 상태에서 기능을 조작하는 경우는 거의 없기 때문에, 화면 꺼짐 시 조작을 제한하는 기능은 따로 첨부하지 않았다.

4.3. 블루투스 수신 및 String화 알고리즘

<원문 코드>

```
while(BTSerial.available())  //BTSerial에 전송된 값이 있으면
  {
    char BTChar = (char)BTSerial.read();  //BTSerial int 값을 char 형식으로 변환
    if(BTChar=='\n' || BTChar=='\r'){
      break;
    }
    BTString+=BTChar;     //수신되는 문자를 BTString에 모두 붙임 (1바이트씩 전송되는 것을 연결)
    delay(5);             //수신 문자열 끊김 방지
  }
```

<추가 설명>
‘블루투스 수신 및 String화 알고리즘’은 BTSerial을 통해 아두이노로 수신된 문자가 있다면, \n이나 \r 값이 나오기 전까지의 값을 char형식으로 바꾼 후, BTChar라는 임시 char 저장소에 저장하고, 이를 BTString+=BTChar를 통해 BTChar에 들어왔던 데이터 1글자씩 string에 이어붙여 BTSting이라는 임시 String 저장소에 저장하는 과정을 거친다. BTSting 구성 단계에서 수신되었던 데이터들은 모두 String화를 거치게 되어, 향후 대입이나 문자열 처리를 용이하게 만들어주는 과정이다.

4.4. String 저장 및 디스플레이 제어 알고리즘

<추가 설명>

```
 if(!BTString.equals(""))  //BTString 값이 있다면
  {
    lcdDisplay[nowCol] = BTString;//BTstring 값을 lcdDisplay에 대입
    //Serial.println(lcdDisplay[nowCol]); //시리얼모니터에 BTString값 출력
    displayLength = lcdDisplay[nowCol].length();//lcdDisplay의 길이 정수 값을 displayLength에 대입

    if(displayLength > 20){// 만약 받은 문자열의 길이가 20을 넘는다면
      temp = lcdDisplay[nowCol].substring(0, 20);//temp 문자열을 이용하여 인덱스 0~19까지만 문자열을 끊어서
      lcd.print(temp);//그 temp 값을 LCD에 띄움
      temp = "                    ";//사용한 temp 문자열을 초기화
    }
    else{
      lcd.print(lcdDisplay[nowCol]);//20 이하의 경우 별다른 과정 없이 디스플레이
    }

    BTString="";  //BTString 변수값 초기화
    lcd.setCursor(0, nowCol);//현재열 값에 맞게 커서 설정
  }
```

<추가 설명>

4.3.에서 String 처리한 문자들을 현재 nowCol 값에 따라 해당 lcdDisplay[nowCol] 문자열에 이동시키는 과정이다. 이 과정에서 BTString의 값은 초기화되어 새로운 문자를 받을 수 있는 상태로 변한다. BTString의 값을 할당 받은 'lcdDisplay[nowCol]'의 문자 길이가 '\0'을 제외하고 20자가 넘지 않는다면, LCD에 lcdDisplay[nowCol]'의 값을 그대로 송출시킨다.
만약 20자가 넘는다면, substring 기능을 통해 자른 글자 20자를 temp를 이용하여 저장하고, 이를 디스플레이로 송출함으로써 해당 값의 초기 모형을 잡는 역할을 수행한다. substring을 통해 글자가 20자까지만 잘려 송출되어도, 해당 값의 원본은 lcdDisplay 문자열에 그대로 저장되어있기 때문에 원본 손실의 우려는 없고, 향후 원본 값을 통해 다른 작업을 수행하기 용이한 상태로 만들어준다.