AMI App Documentation

Prepared By: Md. Rakibul Ahasan

Version 1.1

Initial Page

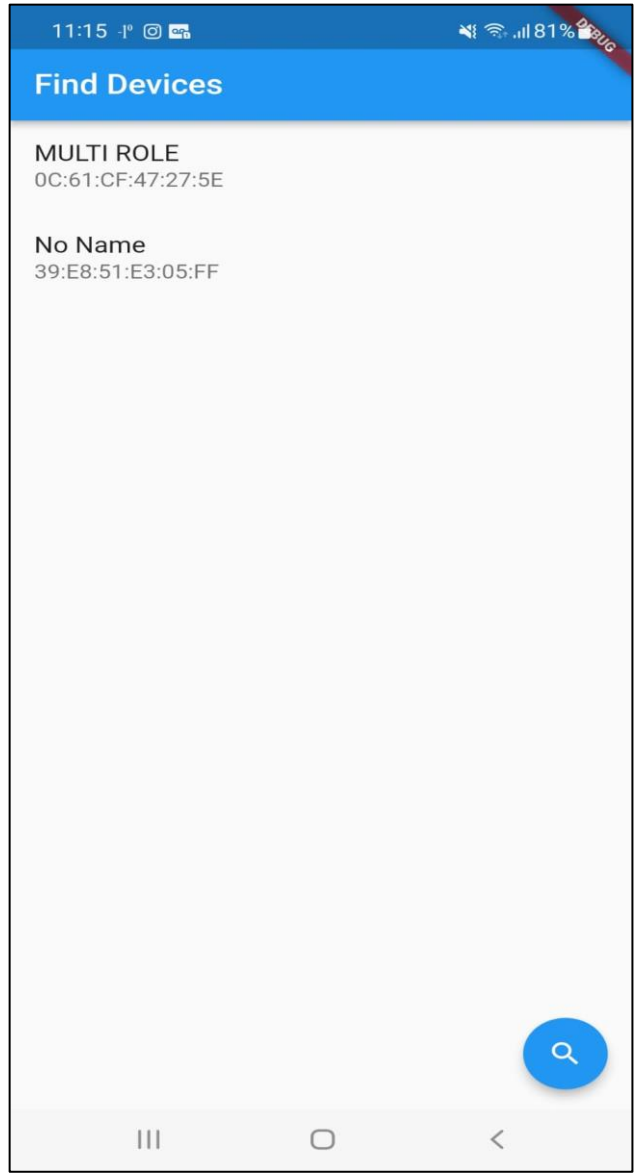
1. The class FlutterBlueApp initiate the app state.
2. Class BluetoothOffScreen sent a massage if Bluetooth adaptor is not “ON” in the mobile phone
3. Class FindDevicesScreen contains the entire 1st page of the app.

✓ In that class there is a floatingActionButton, if pressed it will show the available bluetooth devices.

✓ There is a onTap procedure which allow to connect with desire Bluetooth device and move to the second page DeviceScreen of the app.

New class _FlutterBlueAppState initiate which initialize the local database `initDB()` when app start and delete the data table from the local database when user quit form the app.

```
databaseService.deleteMasterVoltageData();
databaseService.deleteSlaveVoltageData();
```



Database Details

Database Model Class:

- ✓ To create local database two separate database model is created. Those are MasterDBmodel and SlaveDBmodel. This model's will be called and used from the second widget.dart page of the app using json api.

- ✓ These model dart files had to be present in the /../lib/db/model directory. The file name are master_table_model.dart and slave_table_model.dart

Database Service:

- ✓ The class DatabaseService works such a way if any local database exits it return's that database and if not exist it create a new local database.
- ✓ The initDB() method creates the local database `voltageData.db` and tables in the local database for both master and slave.

High level details of the table

Table Name	mastervoltageData				
Column Name	**ID**	**TIME**	**MV**	**MVP**	**MVD**
Derivation			Master Voltage	Master Voltage Percentage	Master Voltage Difference
DataType	Integer	Text	Text	Text	Text

Table Name	slavevoltageData				
Column Name	**ID**	**TIME**	**SV**	**SVP**	**SVD**
Derivation			Slave Voltage	Slave Voltage Percentage	Slave Voltage Difference
DataType	Integer	Text	Text	Text	Text

Database Details

- ✓ The addToMasterDatabase() method is inserting the require data in the master table from the widget.dart page of the app. Query used is “INSERT INTO slavevoltageData(SV,TIME, SVP, SVD) VALUES(?, ?, ?, ?)”

- ✓ The getAllDataFromMasterTable() method returns all the data resides in the master table. Query used is “SELECT * FROM mastervoltageData”

- ✓ The getLatestDataFromMasterTable() method returns the last two iteration value from the master table. Query used is “SELECT * FROM mastervoltageData ORDER BY id DESC LIMIT 2”. This LIMIT “2” is configurable.

- ✓ The deleteMasterVoltageData() method delete the table of master data.

** Similar method is used for slave where method name has slave syntax.

Second Page

✓ The class DeviceScreen holds the entire state of the widget.dart file

✓ Important global variable

double _glassSliderValue = 0; Glass Controller, default is 0

double masterBatterypercentage = 0; Initializing masterbattery percentage value

double slaveBatterypercentage = 0; Initializing slavebattery percentage value

var databaseService = DatabaseService.instance; Database instance initialization

bool isLoadingMaster = true
bool isLoadingSlave = true
| Data loader for master and slave

Timer? mastertimer;
Timer? slavetimer;
| Time initialize for master and slave

List<MasterDBmodel> mastervoldataDateShow = [];
List<SlaveDBmodel> slavevoldataDateShow =[];
| Empty list declaration w.r.t master/slave database model

Second Page

- ✓ The getMasterFromDatabase() method called the getLatestDataFromMasterTable() databased service and store the data in the Database model as a list List<MasterDBmodel>.mastervoldataDateShow[].

- ✓ The getmasterDifferenceValue() method called the getAllDataFromMasterTable() database service to calculate the difference of a voltage from its previous voltage.

- ✓ The getMasterTableValues() method is created to have the list value visualization in the app. This visualization depend on the mastervoldataDateShow.length.

- ✓ The getCurrentDateTime() method derive the current time of the mobile phone. The format is '$hour:$minute:$seconds

- ✓ The getmvcharacter2Value() method is reading the [2] and [3] value of byte array [0, 0, 3, 4] of char2 for master voltage data. These method workflow is below:

```
If (c.properties.read && c.uuid ==
service4Characteristics[1].uuid){}
```

➡ service4ListIntermediate ➡ service4List

```
service4Characteristic1 = service4List.elementAt(2) +(service4List.elementAt(3) * 256);
```

** Similar method is used for slave where method name has slave syntax.

- ✓ The `getMasterVoltage()` method is having the master battery voltage value, calculating the masterBatteryPercentage and using `addToMasterDatabase` database service it inserts the require data in the database table as below

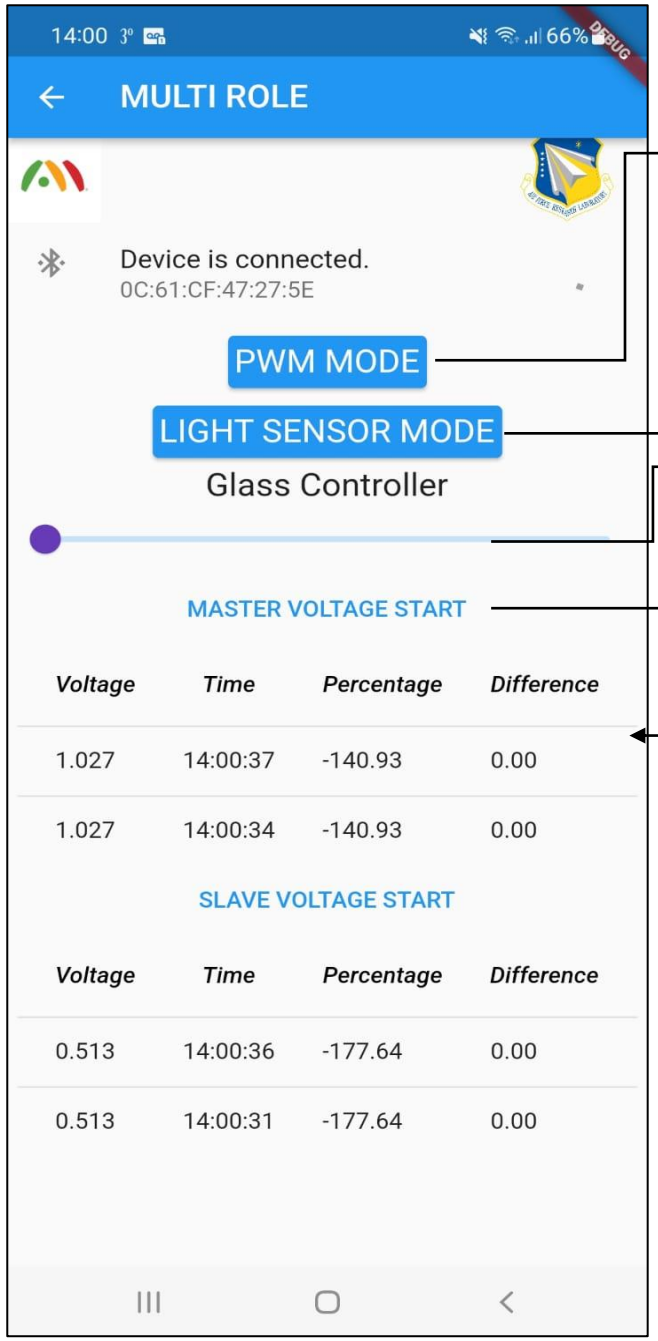
```
mvcv.toString(),
getCurrentDateTime(),
masterBatterypercentage.toString(),
difference.toString(),
```

- ✓ The `sendHexValue` method is created to write hexvalue in the char3 to set the mode of operation of the device.

```
if (services.length >= 4) {
        BluetoothService service4 = services[3];
        if (service4.characteristics.isNotEmpty) {
          service4.characteristics[2].write([hexValue]);
        }
      }
```

** Similar method is used for slave where method name has slave syntax.

How App Work



On pressed sendHexValue is called with value 0x01. This will set the remote device operate as per the value store in the char1

On pressed sendHexValue is called with value 0x02. This will set the remote device operate as per the value perceived from the light sensor.

A Slider determine the glassSlider value to operate. This work on the char1.

Initiate masterTimer.periodic: 3 seconds

getMasterTableValues()
mastervoldataDateShow.length
getMasterFromDatabase()
getLatestDataFromMasterTable()←getMasterVoltage()←getmvcharacter2Value()