

Motorbike Statistics

Generated by Doxygen 1.8.13

Contents

1	motorbikestatistics	1
2	Hierarchical Index	3
2.1	Class Hierarchy	3
3	Class Index	5
3.1	Class List	5
4	File Index	7
4.1	File List	7
5	Class Documentation	9
5.1	AndroidApp.BTConnection Class Reference	9
5.1.1	Detailed Description	10
5.1.2	Constructor & Destructor Documentation	10
5.1.2.1	BTConnection()	10
5.1.3	Member Function Documentation	11
5.1.3.1	setRXHandler()	11
5.1.3.2	getRXHandler()	11
5.1.3.3	run()	12
5.1.3.4	isRunning()	13
5.1.3.5	isConnected()	13
5.1.3.6	connect()	14
5.1.4	Member Data Documentation	14
5.1.4.1	txHandler	14

5.2	AndroidApp.BTDeviceItem Class Reference	15
5.2.1	Detailed Description	15
5.2.2	Constructor & Destructor Documentation	15
5.2.2.1	BTDeviceItem()	15
5.2.3	Member Function Documentation	16
5.2.3.1	getConnection()	16
5.2.3.2	setConnection()	16
5.2.3.3	getDevice()	17
5.2.3.4	getStatus()	17
5.2.3.5	setStatus()	17
5.2.3.6	getIconID()	18
5.2.3.7	setIconID()	18
5.3	AndroidApp.BTDeviceListAdapter Class Reference	18
5.3.1	Detailed Description	19
5.3.2	Constructor & Destructor Documentation	19
5.3.2.1	BTDeviceListAdapter()	19
5.3.3	Member Function Documentation	20
5.3.3.1	getView()	20
5.4	AndroidApp.BTDeviceListAdapter.ViewHolder Class Reference	21
5.4.1	Detailed Description	21
5.5	AndroidApp.DataItem< T > Class Template Reference	21
5.5.1	Detailed Description	22
5.5.2	Constructor & Destructor Documentation	22
5.5.2.1	DataItem() [1/2]	22
5.5.2.2	DataItem() [2/2]	23
5.5.3	Member Function Documentation	23
5.5.3.1	getName()	23
5.5.3.2	getEnabledAvgMinMax()	24
5.5.3.3	getCurrent()	24
5.5.3.4	getAverage()	24

5.5.3.5	getMinimum()	25
5.5.3.6	getMaximum()	25
5.5.3.7	setCurrent()	25
5.5.3.8	add()	26
5.5.3.9	divide()	26
5.5.3.10	greaterThan()	27
5.5.3.11	lessThan()	27
5.6	AndroidApp.DataListAdapter Class Reference	28
5.6.1	Detailed Description	28
5.6.2	Constructor & Destructor Documentation	29
5.6.2.1	DataListAdapter()	29
5.6.3	Member Function Documentation	29
5.6.3.1	getView()	29
5.7	AndroidApp.DataListAdapter.ViewHolder Class Reference	30
5.7.1	Detailed Description	31
5.8	AndroidApp.JSONHandlerSingleton Class Reference	31
5.8.1	Detailed Description	31
5.8.2	Member Function Documentation	31
5.8.2.1	getInstance()	32
5.9	AndroidApp.LoadDeviceFragment Class Reference	32
5.9.1	Detailed Description	33
5.9.2	Constructor & Destructor Documentation	33
5.9.2.1	LoadDeviceFragment()	33
5.9.3	Member Function Documentation	33
5.9.3.1	onCreateView()	33
5.9.3.2	setBTConnection()	34
5.9.3.3	addTrip()	34
5.9.4	Member Data Documentation	35
5.9.4.1	RXHandler	35
5.10	AndroidApp.LoadDeviceFragment.TripItemListener Class Reference	36

5.10.1 Detailed Description	36
5.10.2 Member Function Documentation	36
5.10.2.1 onItemClick()	36
5.11 AndroidApp.MainActivity Class Reference	37
5.11.1 Detailed Description	38
5.11.2 Member Function Documentation	38
5.11.2.1 onCreate()	38
5.11.2.2 onNavigationItemSelected()	39
5.12 AndroidApp.MapsActivity Class Reference	40
5.12.1 Detailed Description	41
5.12.2 Member Function Documentation	41
5.12.2.1 onCreate()	41
5.12.2.2 findJSONByLatLng()	42
5.12.2.3 calcDistance()	43
5.12.2.4 onMapReady()	43
5.12.2.5 convertDpToPixel()	44
5.12.2.6 getBitmapDescriptor()	45
5.13 AndroidApp.MapsActivity.StatisticWindowAdapter Class Reference	46
5.13.1 Detailed Description	46
5.13.2 Member Function Documentation	46
5.13.2.1 getInfoContents()	46
5.14 AndroidApp.MapsActivity.ZoomToggleListener Class Reference	47
5.14.1 Detailed Description	48
5.14.2 Member Function Documentation	48
5.14.2.1 setMarkersVisible()	48
5.15 AndroidApp.PairDeviceFragment Class Reference	49
5.15.1 Detailed Description	50
5.15.2 Constructor & Destructor Documentation	50
5.15.2.1 PairDeviceFragment()	50
5.15.3 Member Function Documentation	50

5.15.3.1	onCreateView()	50
5.15.3.2	getBTConnection()	52
5.15.3.3	getNeededPrivileges()	52
5.16	AndroidApp.PairDeviceFragment.DeviceItemListener Class Reference	53
5.16.1	Detailed Description	53
5.16.2	Member Function Documentation	53
5.16.2.1	onItemClick()	53
5.17	AndroidApp.PairDeviceFragment.DiscoverButtonListener Class Reference	54
5.17.1	Detailed Description	55
5.17.2	Member Function Documentation	55
5.17.2.1	onCheckedChanged()	55
5.18	AndroidApp.PairDeviceFragment.DiscoverReceiver Class Reference	55
5.18.1	Detailed Description	56
5.18.2	Member Function Documentation	56
5.18.2.1	onReceive()	56
5.19	AndroidApp.RealtimeFragment Class Reference	57
5.19.1	Detailed Description	58
5.19.2	Constructor & Destructor Documentation	58
5.19.2.1	RealtimeFragment()	58
5.19.3	Member Function Documentation	58
5.19.3.1	onCreateView()	58
5.19.3.2	newData()	59
5.19.4	Member Data Documentation	60
5.19.4.1	RXHandler	60
5.20	AndroidApp.RealtimeFragment.MapButtonListener Class Reference	61
5.20.1	Detailed Description	61
5.20.2	Member Function Documentation	61
5.20.2.1	onClick()	61
5.21	AndroidApp.SetOfDataItems Class Reference	62
5.21.1	Detailed Description	62

5.21.2	Member Function Documentation	62
5.21.2.1	getItemByName()	62
5.22	AndroidApp.TripItem Class Reference	63
5.22.1	Detailed Description	64
5.22.2	Constructor & Destructor Documentation	64
5.22.2.1	TripItem()	64
5.22.3	Member Function Documentation	64
5.22.3.1	getTripName()	64
5.22.3.2	setTripName()	64
5.22.3.3	getFileSize()	65
5.22.3.4	setFileSize()	65
5.23	AndroidApp.TripListAdapter Class Reference	66
5.23.1	Detailed Description	66
5.23.2	Constructor & Destructor Documentation	66
5.23.2.1	TripListAdapter()	66
5.23.3	Member Function Documentation	67
5.23.3.1	getView()	67
5.24	AndroidApp.TripListAdapter.ViewHolder Class Reference	68
5.24.1	Detailed Description	68
5.25	LoggingDevice::Orientation Class Reference	68
5.25.1	Detailed Description	69
5.25.2	Member Function Documentation	69
5.25.2.1	convertRawAccel()	69
5.25.2.2	convertRawGyro()	70
5.25.2.3	init()	70
5.25.2.4	pollIMU()	71
5.25.2.5	getYaw()	71
5.25.2.6	getPitch()	72
5.25.2.7	getRoll()	72
5.26	LoggingDevice::Storage Class Reference	73
5.26.1	Detailed Description	73
5.26.2	Member Function Documentation	73
5.26.2.1	init()	73
5.26.2.2	saveToFile()	74
5.26.2.3	generateFileName()	74
5.26.2.4	loadTripNames()	75
5.26.2.5	loadSavedTrip()	76

6	File Documentation	77
6.1	android-app/app/src/main/java/com/jack/motorbikestatistics/BTConnection.java File Reference . . .	77
6.1.1	Detailed Description	77
6.2	android-app/app/src/main/java/com/jack/motorbikestatistics/BTDeviceItem.java File Reference . . .	77
6.2.1	Detailed Description	78
6.3	android-app/app/src/main/java/com/jack/motorbikestatistics/BTDeviceListAdapter.java File Reference	78
6.3.1	Detailed Description	78
6.4	android-app/app/src/main/java/com/jack/motorbikestatistics/DataItem.java File Reference	79
6.4.1	Detailed Description	79
6.5	android-app/app/src/main/java/com/jack/motorbikestatistics/DataListAdapter.java File Reference . .	79
6.5.1	Detailed Description	79
6.6	android-app/app/src/main/java/com/jack/motorbikestatistics/JSONHandlerSingleton.java File Refer- ence	80
6.6.1	Detailed Description	80
6.7	android-app/app/src/main/java/com/jack/motorbikestatistics/LoadDeviceFragment.java File Reference	80
6.7.1	Detailed Description	80
6.8	android-app/app/src/main/java/com/jack/motorbikestatistics/MainActivity.java File Reference	81
6.8.1	Detailed Description	81
6.9	android-app/app/src/main/java/com/jack/motorbikestatistics/MapsActivity.java File Reference	81
6.9.1	Detailed Description	81
6.10	android-app/app/src/main/java/com/jack/motorbikestatistics/PairDeviceFragment.java File Reference	82
6.10.1	Detailed Description	82
6.11	android-app/app/src/main/java/com/jack/motorbikestatistics/RealtimeFragment.java File Reference	82
6.11.1	Detailed Description	83
6.12	android-app/app/src/main/java/com/jack/motorbikestatistics/SetOfDataItems.java File Reference . .	83
6.12.1	Detailed Description	83
6.13	android-app/app/src/main/java/com/jack/motorbikestatistics/TripItem.java File Reference	83
6.13.1	Detailed Description	84
6.14	android-app/app/src/main/java/com/jack/motorbikestatistics/TripListAdapter.java File Reference . .	84
6.14.1	Detailed Description	84
6.15	logging-device/logging-device.ino File Reference	85

6.15.1 Detailed Description	86
6.15.2 Function Documentation	86
6.15.2.1 setup()	87
6.15.2.2 loop()	87
6.15.2.3 parseNewMode()	88
6.15.2.4 realTimeMode()	89
6.15.2.5 addOrientationToJSON()	89
6.15.2.6 addGPSToJSON()	90
6.15.2.7 addTimeToJSON()	90
6.16 logging-device/Orientation.cpp File Reference	90
6.16.1 Detailed Description	91
6.17 logging-device/Storage.cpp File Reference	91
6.17.1 Detailed Description	92
Index	93

Chapter 1

motorbikestatistics

Motorcycle statistics device for analysing rider performance

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

AndroidApp.BTDeviceItem	15
AndroidApp.BTDeviceListAdapter.ViewHolder	21
AndroidApp.DataItem< T >	21
AndroidApp.DataListAdapter.ViewHolder	30
AndroidApp.JSONHandlerSingleton	31
AndroidApp.TripItem	63
AndroidApp.TripListAdapter.ViewHolder	68
OnClickListener	
AndroidApp.RealtimeFragment.MapButtonListener	61
InfoWindowAdapter	
AndroidApp.MapsActivity.StatisticWindowAdapter	46
OnCameraMoveListener	
AndroidApp.MapsActivity.ZoomToogleListener	47
OnItemClickListener	
AndroidApp.LoadDeviceFragment.TripItemListener	36
AndroidApp.PairDeviceFragment.DeviceItemListener	53
LoggingDevice::Orientation	68
LoggingDevice::Storage	73
OnNavigationItemSelectedListener	
AndroidApp.MainActivity	37
Runnable	
AndroidApp.BTConnection	9
OnCheckedChangeListener	
AndroidApp.PairDeviceFragment.DiscoverButtonListener	54
Fragment	
AndroidApp.LoadDeviceFragment	32
AndroidApp.PairDeviceFragment	49
AndroidApp.RealtimeFragment	57
BroadcastReceiver	
AndroidApp.PairDeviceFragment.DiscoverReceiver	55
FragmentActivity	
AndroidApp.MapsActivity	40
AppCompatActivity	
AndroidApp.MainActivity	37
ArrayAdapter	

AndroidApp.BTDeviceListAdapter	18
AndroidApp.DataListAdapter	28
AndroidApp.TripListAdapter	66
OnMapReadyCallback	
AndroidApp.MapsActivity	40
ArrayList	
AndroidApp.SetOfDataItems	62

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

AndroidApp.BTConnection	
Thread class for a new bluetooth connection to a device	9
AndroidApp.BTDeviceItem	
Class used for holding core UI information of a bluetooth devices	15
AndroidApp.BTDeviceListAdapter	
Adapter class used for displaying bluetooth devices	18
AndroidApp.BTDeviceListAdapter.ViewHolder	
Class that holds all data displayed for each ListItem	21
AndroidApp.DataItem< T >	
Class used for holding and displaying a piece of data within the statistic ListView UI	21
AndroidApp.DataListAdapter	
Adapter class used for displaying statistics	28
AndroidApp.DataListAdapter.ViewHolder	
Class that holds all data displayed for each ListItem	30
AndroidApp.JSONHandlerSingleton	
Singleton class for holding all JSON trip data	31
AndroidApp.LoadDeviceFragment	
UI Class for loading saved trips from device	32
AndroidApp.LoadDeviceFragment.TripItemListener	
Listener used to identify when a trip has been pressed	36
AndroidApp.MainActivity	
Main activity class for fragment navigation	37
AndroidApp.MapsActivity	
Maps activity class for displaying map data	40
AndroidApp.MapsActivity.StatisticWindowAdapter	
Adapter used for displaying statistics at a certain marker that user has clicked on	46
AndroidApp.MapsActivity.ZoomToggleListener	
Listener class for making markers invisible when zoomed out	47
AndroidApp.PairDeviceFragment	
UI Class for discovering, pairing and connecting to the logging device	49
AndroidApp.PairDeviceFragment.DeviceItemListener	
Listener for when a ListView item is pressed (to connect)	53
AndroidApp.PairDeviceFragment.DiscoverButtonListener	
Listener for when discovery button is pressed	54
AndroidApp.PairDeviceFragment.DiscoverReceiver	
Receiver for when a new device is discovered	55

AndroidApp.RealtimeFragment	
UI Class for viewing data sent from the logging device	57
AndroidApp.RealtimeFragment.MapButtonListener	
Listener for starting a map activity when button pressed	61
AndroidApp.SetOfDataItems	
ArrayList extension to allow searching via item name	62
AndroidApp.TripItem	
Class used for holding name and size information relating to a trip	63
AndroidApp.TripListAdapter	
Adapter class used for displaying all trips	66
AndroidApp.TripListAdapter.ViewHolder	
Class that holds all UI data to be displayed for each ListItem	68
LoggingDevice::Orientation	
Class for dealing with Orientation functionality on logging device	68
LoggingDevice::Storage	
Class for storing & retrieving data on the logging device	73

Chapter 4

File Index

4.1 File List

Here is a list of all documented files with brief descriptions:

android-app/app/src/main/java/com/jack/motorbikestatistics/ BTConnection.java	77
Class for holding containing bluetooth connection on app	
android-app/app/src/main/java/com/jack/motorbikestatistics/ BTDeviceItem.java	77
UI class for holding information regarding a bluetooth device	
android-app/app/src/main/java/com/jack/motorbikestatistics/ BTDeviceListAdapter.java	78
UI ListView adapter to display bluetooth devices	
android-app/app/src/main/java/com/jack/motorbikestatistics/ DataItem.java	79
UI class for holding information regarding a specific statistic	
android-app/app/src/main/java/com/jack/motorbikestatistics/ DataListAdapter.java	79
UI ListView adapter to display statistics	
android-app/app/src/main/java/com/jack/motorbikestatistics/ JSONHandlerSingleton.java	80
Singleton class that holds an array of the app's JSON data	
android-app/app/src/main/java/com/jack/motorbikestatistics/ LoadDeviceFragment.java	80
Fragment/Tab for providing UI for loading from device	
android-app/app/src/main/java/com/jack/motorbikestatistics/ MainActivity.java	81
Main activity class responsible for tabbing	
android-app/app/src/main/java/com/jack/motorbikestatistics/ MapsActivity.java	81
Maps activity class responsible for showing data on Google Maps	
android-app/app/src/main/java/com/jack/motorbikestatistics/ PairDeviceFragment.java	82
Fragment/Tab for connecting to the logging device	
android-app/app/src/main/java/com/jack/motorbikestatistics/ RealtimeFragment.java	82
Fragment/Tab for viewing streamed statistics	
android-app/app/src/main/java/com/jack/motorbikestatistics/ SetOfDataItems.java	83
Extension of ArrayList allows for searching via name	
android-app/app/src/main/java/com/jack/motorbikestatistics/ TripItem.java	83
Class for holding information relating to a specific trip	
android-app/app/src/main/java/com/jack/motorbikestatistics/ TripListAdapter.java	84
UI ListView adapter to display all saved trips	
logging-device/ logging-device.ino	85
Arduino sketch for the logging device	
logging-device/ Orientation.cpp	90
Module created to deal with all orientation related functionality	
logging-device/ Orientation.h	??
logging-device/ Storage.cpp	91
Module created to handle all storage related functionality	
logging-device/ Storage.h	??

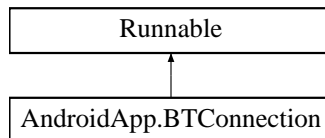
Chapter 5

Class Documentation

5.1 AndroidApp.BTConnection Class Reference

Thread class for a new bluetooth connection to a device.

Inheritance diagram for AndroidApp.BTConnection:



Public Member Functions

- `BTConnection` (BluetoothDevice `btDevice`) throws `IOException`
Constructor for `BTConnection` class.
- void `setRXHandler` (Handler `newHandler`)
Setter function for `RXHandler`.
- Handler `getRXHandler` ()
Getter function for `RXHandler`.
- void `run` ()
Main run procedure for new `Runnable` thread created.
- void `stop` ()
Procedure to stop the bluetooth connection thread from running.
- boolean `isRunning` ()
Function to check whether main connection thread is running.
- boolean `isConnected` ()
Function to check whether BT connection is still valid.

Public Attributes

- final Handler `txHandler`
Handler class for transmission of data.

Private Member Functions

- void `connect` () throws IOException
Procedure to create a connection to logging device.
- void `close` () throws IOException
Closes the BT connection socket, exceptions thrown on failure.

Private Attributes

- BluetoothDevice `btDevice`
Bluetooth Device object, holds information for chosen slave.
- Handler `RXHandler` = null
Handler function where received data is sent to.
- BluetoothSocket `btSocket` = null
Socket created for bluetooth connection, used for TX/RX.
- volatile boolean `running` = false
Indicates whether main run thread is in progress.

Static Private Attributes

- static final String `TAG` = "BTConnection"
Tag using for debugging.
- static final UUID `uuid` = UUID.fromString("00001101-0000-1000-8000-00805f9b34fb")
UUID to allow Serial connection via BT.
- static final String `NEW_LINE` = "\r\n"
New line string.

5.1.1 Detailed Description

Thread class for a new bluetooth connection to a device.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 BTConnection()

```
AndroidApp.BTConnection.BTConnection (
    BluetoothDevice btDevice ) throws IOException [inline]
```

Constructor for `BTConnection` class.

Sets the BT device interface used for this class and attempts a connection.

Parameters

<i>btDevice</i>	- Device used for creating connection.
-----------------	--

```

59         {
60             this.btDevice = btDevice;
61         }
62         /* Connect the device so ready to use run */
63         connect();
64     }

```

5.1.3 Member Function Documentation

5.1.3.1 setRXHandler()

```

void AndroidApp.BTConnection.setRXHandler (
    Handler newHandler ) [inline]

```

Setter function for RXHandler.

Parameters

<i>newHandler</i>	- The new Handler where RX'd data will be sent to.
-------------------	--

```

108         {
109             RXHandler = newHandler;
110         }

```

5.1.3.2 getRXHandler()

```

Handler AndroidApp.BTConnection.getRXHandler ( ) [inline]

```

Getter function for RXHandler.

Returns

Handler - The RX handler for BT connection.

```

117         {
118             return RXHandler;
119         }

```

5.1.3.3 run()

```
void AndroidApp.BTConnection.run ( ) [inline]
```

Main run procedure for new Runnable thread created.

If connected procedure waits for data to be received. Parsing this received into lines and then splitting each line into a JSONObject. If a valid JSONObject is found it is then sends to the receive handler in a seperate thread (using messages).

```

130         {
131             InputStream RXStream;
132
133             /* Indicate that we are now running main thread */
134             running = true;
135
136             if (isConnected()) {
137                 /* Get our input stream for receiving bytes */
138                 try {
139                     RXStream = btSocket.getInputStream();
140                 } catch (IOException e) {
141                     Log.e(TAG, "Unable to get RXStream", e);
142                     running = false;
143                     return;
144                 }
145
146                 /*
147                  * While still connected and not signalled to stop we receive data
148                  * and then send it to the handler
149                  */
150                 String recvBuff = "";
151                 while (isRunning() && isConnected()) {
152                     try {
153                         int bytesAvailable = RXStream.available();
154
155                         if (bytesAvailable > 0) {
156                             byte[] packetBytes = new byte[bytesAvailable];
157                             int bytesRead = RXStream.read(packetBytes, 0, bytesAvailable);
158
159                             recvBuff += new String(packetBytes);
160                         }
161
162                         if (RXHandler != null) {
163                             if (recvBuff.indexOf(NEW_LINE) > 0) {
164
165                                 String jsonLine = recvBuff.substring(0, recvBuff.indexOf(
166                                     NEW_LINE));
167
168                                 /*
169                                 * Having to send data to main thread using messages
170                                 * as we are multithreading.
171                                 * If we try and use a standard call to function
172                                 * will cause a crash.
173                                 */
174                                 Bundle dataBundle = new Bundle();
175                                 dataBundle.putString("JSON", jsonLine);
176
177                                 Message message = RXHandler.obtainMessage();
178                                 message.setData(dataBundle);
179                                 message.sendToTarget();
180
181                                 recvBuff = recvBuff.replace(jsonLine + NEW_LINE, "");
182                             }
183                         }
184
185                     } catch (IOException e) {
186                         Log.e(TAG, "Unable to read data", e);
187                         running = false;
188                         return;
189                     }
190                 }
191             }
192         }
193
194         /* Close bluetooth socket */
195         try {
196             this.close();
197         } catch (IOException e) {
198             /* Do nothing */
199         }

```

```
200
201      /* Null BT socket to show needs to reconnect */
202      btSocket = null;
203      running = false;
204  }
```

5.1.3.4 isRunning()

```
boolean AndroidApp.BTConnection.isRunning ( ) [inline]
```

Function to check whether main connection thread is running.

Returns

boolean - Whether thread is running.

```
217      {
218      return running;
219  }
```

5.1.3.5 isConnected()

```
boolean AndroidApp.BTConnection.isConnected ( ) [inline]
```

Function to check whether BT connection is still valid.

Returns

boolean - Whether connection is still available.

```
225      {
226      boolean result = false;
227      if (btSocket != null) {
228          if (btSocket.isConnected())
229              result = true;
230      }
231      return result;
232  }
```

5.1.3.6 connect()

`void AndroidApp.BTConnection.connect () throws IOException [inline], [private]`

Procedure to create a connection to logging device.

Creates a raw Serial socket via UUID and then attempts to connect. Exceptions thrown on failure.

```

241                                     {
242
243         /* Attempt to make connection to remote device, throw exception if not */
244         try {
245             btSocket = btDevice.createRfcommSocketToServiceRecord(
246                 uuid);
247         } catch (IOException e) {
248             Log.e(TAG, "Unable to create RFCOMM", e);
249             throw e;
250         }
251         try {
252             btSocket.connect();
253         } catch (IOException e) {
254             Log.e(TAG, "Unable to connect", e);
255
256             /* Close our socket as unable to connect */
257             try {
258                 this.close();
259             } catch (IOException e2) {
260                 throw e2;
261             }
262             throw e;
263         }
264     }

```

5.1.4 Member Data Documentation

5.1.4.1 txHandler

`final Handler AndroidApp.BTConnection.txHandler`

Initial value:

```

= new Handler(Looper.getMainLooper()) {

    @Override
    public void handleMessage(Message msg) {

        if (isConnected() && isRunning()) {
            OutputStream TXStream;

            try {
                TXStream = btSocket.getOutputStream();

                String txString = (String)msg.obj;
                TXStream.write(txString.getBytes());
            } catch (IOException e) {
                Log.e(TAG, "Unable to use TXStream", e);
                return;
            }
        }
    }
}

```

Handler class for transmission of data.

Messages containing data to be transmitted are sent from main UI thread.

The documentation for this class was generated from the following file:

- [android-app/app/src/main/java/com/jack/motorbikestatistics/BTConnection.java](#)

5.2 AndroidApp.BTDeviceItem Class Reference

Class used for holding core UI information of a bluetooth devices.

Public Member Functions

- [BTConnection getConnection](#) ()
Getter for the bluetooth connection of specified device.
- void [setConnection](#) ([BTConnection](#) newConn)
Setter for setting the DeviceItem object's connection.
- [BluetoothDevice getDevice](#) ()
Getter for BT device object (contains name, HWID etc.).
- String [getStatus](#) ()
Getter for current status of [BTDeviceItem](#).
- void [setStatus](#) (String newStatus)
Setter for current status of [BTDeviceItem](#).
- int [getIconID](#) ()
Getter for icon ID to use in ListView.
- void [setIconID](#) (int newID)
Setter for icon ID to use in ListView.
- [BTDeviceItem](#) ([BluetoothDevice](#) device, String status, int iconID)
Constructor for [BTDeviceItem](#) class.

Private Attributes

- [BTConnection connection](#) = null
Variable for [BTConnection](#) if device is already connected.
- int [iconID](#)
ID of icon to use within the ListView.
- [BluetoothDevice device](#)
Device object that holds info such as name, HWID etc.
- String [status](#)
Status of the device, unpaired, paired, connected.

5.2.1 Detailed Description

Class used for holding core UI information of a bluetooth devices.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 BTDeviceItem()

```
AndroidApp.BTDeviceItem.BTDeviceItem (
    BluetoothDevice device,
    String status,
    int iconID ) [inline]
```

Constructor for [BTDeviceItem](#) class.

Called when new BluetoothDevice is found during discovery, so that it can be added to the device ListView.

Parameters

<i>device</i>	- BluetoothDevice containing HWID, name, etc.
<i>status</i>	- Current status of the discovered device.
<i>iconID</i>	- Icon ID to display within the ListView.

```

96     {
97         this.device = device;
98         this.status = status;
99         this.iconID = iconID;
100    }
```

5.2.3 Member Function Documentation**5.2.3.1 getConnection()**

```
BTConnection AndroidApp.BTDeviceItem.getConnection ( ) [inline]
```

Getter for the bluetooth connection of specified device.

Returns

BTConnection - Connection between app & logging device.

```

33                                     {
34         return connection;
35     }
```

5.2.3.2 setConnection()

```
void AndroidApp.BTDeviceItem.setConnection (
    BTConnection newConn ) [inline]
```

Setter for setting the DeviceItem object's connection.

Parameters

<i>newConn</i>	- New connection between app & logging device.
----------------	--

```

41                                     {
42         connection = newConn;
43     }
```

5.2.3.3 `getDevice()`

```
BluetoothDevice AndroidApp.BTDeviceItem.getDevice ( ) [inline]
```

Getter for BT device object (contains name, HWID etc.).

Returns

BluetoothDevice - The bluetooth device object.

```
49                                     {
50         return device;
51     }
```

5.2.3.4 `getStatus()`

```
String AndroidApp.BTDeviceItem.getStatus ( ) [inline]
```

Getter for current status of [BTDeviceItem](#).

Returns

String - Current status: unpaired, paired or connected.

```
57                                     {
58         return status;
59     }
```

5.2.3.5 `setStatus()`

```
void AndroidApp.BTDeviceItem.setStatus (
    String newStatus ) [inline]
```

Setter for current status of [BTDeviceItem](#).

Parameters

<i>newStatus</i>	- New string for status.
------------------	--------------------------

```
65                                     {
66         status = newStatus;
67     }
```

5.2.3.6 getIconID()

```
int AndroidApp.BTDeviceItem.getIconID ( ) [inline]
```

Getter for icon ID to use in ListView.

Returns

int - Icon ID to use.

```
73     {
74         return iconID;
75     }
```

5.2.3.7 setIconID()

```
void AndroidApp.BTDeviceItem.setIconID (
    int newID ) [inline]
```

Setter for icon ID to use in ListView.

Parameters

<i>newID</i>	- New icon ID to use.
--------------	-----------------------

```
81     {
82         iconID = newID;
83     }
```

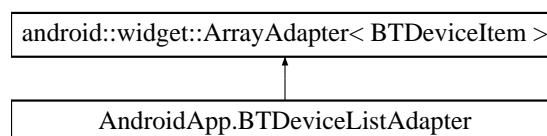
The documentation for this class was generated from the following file:

- android-app/app/src/main/java/com/jack/motorbikestatistics/[BTDeviceItem.java](#)

5.3 AndroidApp.BTDeviceListAdapter Class Reference

Adapter class used for displaying bluetooth devices.

Inheritance diagram for AndroidApp.BTDeviceListAdapter:



Classes

- class [ViewHolder](#)

Class that holds all data displayed for each ListItem.

Public Member Functions

- [BTDeviceListAdapter](#) (Context cnt, int [layoutResourceId](#), ArrayList< [BTDeviceItem](#) > [data](#))
Constructor for the ListView adapter.
- View [getView](#) (int position, View convertView, ViewGroup parent)
Function for returning the view of each list item ([BTDeviceItem](#)).

Private Attributes

- int [layoutResourceId](#)
Resource ID for current layout.
- Context [context](#)
Context that the ListView is operating in.
- ArrayList< [BTDeviceItem](#) > [data](#)
ArrayList of all bluetooth device items to display.

5.3.1 Detailed Description

Adapter class used for displaying bluetooth devices.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 BTDeviceListAdapter()

```
AndroidApp.BTDeviceListAdapter.BTDeviceListAdapter (
    Context cnt,
    int layoutResourceId,
    ArrayList< BTDeviceItem > data ) [inline]
```

Constructor for the ListView adapter.

Calls the constructor of the superclass as well as setting other relevant information needed.

Parameters

<i>cnt</i>	- Context of the adapter to be operating in.
<i>layout↔ ResourceId</i>	- Resource ID for current layout.
<i>data</i>	- ArrayList of devices to display in ListView.

```

51                                     {
52         super(cnt, layoutResourceId, data);
53         this.context = cnt;
54         this.layoutResourceId = layoutResourceId;
55         this.data = data;
56     }

```

5.3.3 Member Function Documentation

5.3.3.1 getView()

```

View AndroidApp.BTDeviceListAdapter.getView (
    int position,
    View convertView,
    ViewGroup parent ) [inline]

```

Function for returning the view of each list item ([BTDeviceItem](#)).

If a view for selected item has not been created inflater initialises it. A holder is then used to hold all the information that will be displayed on the UI to the user.

Parameters

<i>position</i>	- Index of item in array to use/reference to.
<i>convertView</i>	- View to be used for specified item.
<i>parent</i>	- Object where the created view will be placed on.

Returns

View - The result view of item with updated/current information.

References [AndroidApp.BTDeviceItem.getDevice\(\)](#), [AndroidApp.BTDeviceItem.getIconID\(\)](#), and [AndroidApp.BTDeviceItem.getStatus\(\)](#).

```

82                                     {
83
84         ViewHolder holder;
85
86         if (convertView == null)
87         {
88             /* Create new view via inflater as it does not exist. */
89             LayoutInflater inflater = (LayoutInflater)context.getSystemService(Context.
LAYOUT_INFLATER_SERVICE);
90             convertView = inflater.inflate(layoutResourceId, parent, false);
91
92             /* Create holder that will contain information to display. */
93             holder = new ViewHolder();
94             holder.imageStatus = (ImageView)convertView.findViewById(R.id.imageListStatus);
95             holder.name = (TextView)convertView.findViewById(R.id.textListName);
96             holder.address = (TextView)convertView.findViewById(R.id.textListAddress);
97             holder.status = (TextView)convertView.findViewById(R.id.textListStatus);
98             convertView.setTag(holder);
99         }
100         else
101         {
102             /* Get current holder to use instead of creating new one. */
103             holder = (ViewHolder)convertView.getTag();
104         }
105

```

```

106      /* Get BTDeviceItem for specified item and update holder info. */
107      BTDeviceItem btItem = getItem(position);
108      holder.imageStatus.setImageResource(btItem.getIconID());
109      holder.name.setText(btItem.getDevice().getName());
110      holder.address.setText(btItem.getDevice().getAddress());
111      holder.status.setText(btItem.getStatus());
112
113      return convertView;
114  }

```

The documentation for this class was generated from the following file:

- android-app/app/src/main/java/com/jack/motorbikestatistics/[BTDeviceListAdapter.java](#)

5.4 AndroidApp.BTDeviceListAdapter.ViewHolder Class Reference

Class that holds all data displayed for each ListItem.

5.4.1 Detailed Description

Class that holds all data displayed for each ListItem.

The documentation for this class was generated from the following file:

- android-app/app/src/main/java/com/jack/motorbikestatistics/[BTDeviceListAdapter.java](#)

5.5 AndroidApp.DataItem< T > Class Template Reference

Class used for holding and displaying a piece of data within the statistic ListView UI.

Public Member Functions

- [DataItem](#) (String [name](#), boolean avgMinMax)
Constructor for creation of a [DataItem](#).
- [DataItem](#) (String [name](#), boolean avgMinMax, T value)
Constructor for creation of a [DataItem](#).
- String [getName](#) ()
Getter for name of data item.
- boolean [getEnabledAvgMinMax](#) ()
Getter for whether additional functionality enabled.
- T [getCurrent](#) ()
Getter for current reading value.
- Double [getAverage](#) ()
Getter for average of readings.
- T [getMinimum](#) ()
Getter for minimum of readings.
- T [getMaximum](#) ()
Getter for maximum of readings.
- void [setCurrent](#) (T value)
Setter for current reading value.

Private Member Functions

- Double `add` (Number a, Number b)
Function to allow addition of numbers with variable types.
- Double `divide` (Number numerator, Number denominator)
Function to allow division of numbers with variable types.
- boolean `greaterThan` (Number a, Number b)
Function to check whether A is greater than B.
- boolean `lessThan` (Number a, Number b)
Function to check whether A is less than B.

Private Attributes

- String `name`
The name of the statistic.
- boolean `enableAvgMinMax`
Whether averaging, min & max values should be calculated.
- T `current` = null
Current reading value.
- Double `average` = null
Average reading value.
- Double `averageSum` = 0.0
Sum of all readings, used for averaging.
- int `averageCount` = 0
Number of readings, used for averaging.
- T `minimum` = null
Minimum reading value.
- T `maximum` = null
Maximum reading value.

5.5.1 Detailed Description

Class used for holding and displaying a piece of data within the statistic ListView UI.

5.5.2 Constructor & Destructor Documentation

5.5.2.1 DataItem() [1/2]

```
AndroidApp.DataItem< T >.DataItem (
    String name,
    boolean avgMinMax ) [inline]
```

Constructor for creation of a `DataItem`.

Sets up the name of the data item as well as Whether averaging, minimum and maximum readings will be used

Parameters

<i>name</i>	- Name of the data item.
<i>avgMinMax</i>	- Whether additive functionality shall be available.

```

48                                     {
49         this.name = name;
50         this.enableAvgMinMax = avgMinMax;
51     }

```

5.5.2.2 `DataItem()` [2/2]

```

AndroidApp.DataItem< T >.DataItem (
    String name,
    boolean avgMinMax,
    T value ) [inline]

```

Constructor for creation of a `DataItem`.

Similar to other constructor however allows setting of an initial value.

Parameters

<i>name</i>	- Name of the data item.
<i>avgMinMax</i>	- Whether additive functionality shall be available.
<i>value</i>	- Initial reading value.

```

63                                     {
64         this.name = name;
65         this.enableAvgMinMax = avgMinMax;
66         this.current = value;
67
68         if ((avgMinMax) && (current instanceof Number)) {
69
70             Number val = (Number)value;
71             this.average = val.doubleValue();
72             this.averageSum = val.doubleValue();
73             this.averageCount++;
74
75             this.minimum = value;
76             this.maximum = value;
77         }
78     }

```

5.5.3 Member Function Documentation

5.5.3.1 `getName()`

```

String AndroidApp.DataItem< T >.getName ( ) [inline]

```

Getter for name of data item.

Returns

String - [DataItem](#) name.

```
84         {
85         return name;
86     }
```

5.5.3.2 [getEnabledAvgMinMax\(\)](#)

boolean [AndroidApp.DataItem](#)< T >.getEnabledAvgMinMax () [inline]

Getter for whether additional functionality enabled.

Returns

boolean - Averaging, Minimum & Maximum enabled.

```
92         {
93         return enableAvgMinMax;
94     }
```

5.5.3.3 [getCurrent\(\)](#)

T [AndroidApp.DataItem](#)< T >.getCurrent () [inline]

Getter for current reading value.

Returns

T - Current reading value.

```
100         {
101         return current;
102     }
```

5.5.3.4 [getAverage\(\)](#)

Double [AndroidApp.DataItem](#)< T >.getAverage () [inline]

Getter for average of readings.

Returns

Double - Average of all readings.

```
108         {
109         return average;
110     }
```

5.5.3.5 `getMinimum()`

```
T AndroidApp.DataItem< T >.getMinimum ( ) [inline]
```

Getter for minimum of readings.

Returns

T - Minimum value.

```
116         {  
117             return minimum;  
118         }
```

5.5.3.6 `getMaximum()`

```
T AndroidApp.DataItem< T >.getMaximum ( ) [inline]
```

Getter for maximum of readings.

Returns

T - Maximum value.

```
124         {  
125             return maximum;  
126         }
```

5.5.3.7 `setCurrent()`

```
void AndroidApp.DataItem< T >.setCurrent (   
    T value ) [inline]
```

Setter for current reading value.

If additive functionality enabled and the reading is of types number then we go ahead and update our min, max & average values as well will the passed in new reading.

Parameters

<i>value</i>	- New reading.
--------------	----------------

```
137         {  
138             this.current = value;  
139             if ((enableAvgMinMax) && (current instanceof Number)) {  
140                   
141             }
```

```

142         /* Sets the average */
143         averageCount++;
144         averageSum = add(averageSum, (Number)value);
145         average = divide(averageSum, averageCount);
146
147         /* Sets the new minimum and maximums if true */
148         if ((minimum == null) || lessThan((Number)current, (Number)
minimum)) {
149             minimum = current;
150         }
151         if ((maximum == null) || greaterThan((Number)current, (Number)
maximum)) {
152             maximum = current;
153         }
154     }
155 }

```

5.5.3.8 add()

```

Double AndroidApp.DataItem< T >.add (
    Number a,
    Number b ) [inline], [private]

```

Function to allow addition of numbers with variable types.

Parameters

<i>a</i>	- First operand.
<i>b</i>	- Second operand.

Returns

Double - Sum.

```

163         {
164         return new Double(a.doubleValue() + b.doubleValue());
165     }

```

5.5.3.9 divide()

```

Double AndroidApp.DataItem< T >.divide (
    Number numerator,
    Number denominator ) [inline], [private]

```

Function to allow division of numbers with variable types.

Parameters

<i>numerator</i>	- Numerator of divisor.
<i>denominator</i>	- Denominator of divisor.

Returns

Double - Result of division.

```

173                                     {
174         return new Double(numerator.doubleValue() / denominator.doubleValue());
175     }

```

5.5.3.10 greaterThan()

```

boolean AndroidApp.DataItem< T >.greaterThan (
    Number a,
    Number b ) [inline], [private]

```

Function to check whether A is greater than B.

Parameters

<i>a</i>	- First operand.
<i>b</i>	- Second operand.

Returns

boolean - Whether A is greater than B.

```

183                                     {
184         return a.doubleValue() > b.doubleValue();
185     }

```

5.5.3.11 lessThan()

```

boolean AndroidApp.DataItem< T >.lessThan (
    Number a,
    Number b ) [inline], [private]

```

Function to check whether A is less than B.

Parameters

<i>a</i>	- First operand.
<i>b</i>	- Second operand.

Returns

boolean - Whether A is less than B.

```

193                                     {

```

```

194         return a.doubleValue() < b.doubleValue();
195     }

```

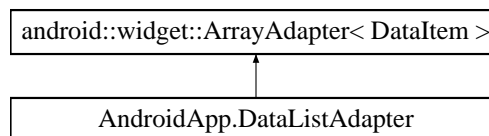
The documentation for this class was generated from the following file:

- android-app/app/src/main/java/com/jack/motorbikestatistics/[DataItem.java](#)

5.6 AndroidApp.DataListAdapter Class Reference

Adapter class used for displaying statistics.

Inheritance diagram for AndroidApp.DataListAdapter:



Classes

- class [ViewHolder](#)
Class that holds all data displayed for each ListItem.

Public Member Functions

- [DataListAdapter](#) (Context cnt, int [layoutResourceId](#), ArrayList< [DataItem](#) > [data](#))
Constructor for the ListView adapter.
- View [getView](#) (int position, View convertView, ViewGroup parent)
Function for returning the view of each list item ([DataItem](#)).

Private Attributes

- Context [context](#)
Context that the ListView is operating in.
- int [layoutResourceId](#)
Resource ID for current layout.
- ArrayList< [DataItem](#) > [data](#)
ArrayList of all statistic items to display.

5.6.1 Detailed Description

Adapter class used for displaying statistics.

5.6.2 Constructor & Destructor Documentation

5.6.2.1 DataListAdapter()

```
AndroidApp.DataListAdapter.DataListAdapter (
    Context cnt,
    int layoutResourceId,
    ArrayList< DataItem > data ) [inline]
```

Constructor for the ListView adapter.

Calls the constructor of the superclass as well as setting other relevant information needed.

Parameters

<i>cnt</i>	- Context of the adapter to be operating in.
<i>layoutResourceId</i>	- Resource ID for current layout.
<i>data</i>	- ArrayList of statistics to display in ListView.

```
47                                     {
48         super(cnt, layoutResourceId, data);
49
50         this.context = cnt;
51         this.layoutResourceId = layoutResourceId;
52         this.data = data;
53     }
```

5.6.3 Member Function Documentation

5.6.3.1 getView()

```
View AndroidApp.DataListAdapter.getView (
    int position,
    View convertView,
    ViewGroup parent ) [inline]
```

Function for returning the view of each list item ([DataItem](#)).

If a view for selected item has not been created inflater initialises it. A holder is then used to hold all the information that will be displayed on the UI to the user.

Parameters

<i>position</i>	- Index of item in array to use/reference to.
<i>convertView</i>	- View to be used for specified item.
<i>parent</i>	- Object where the created view will be placed on.

Returns

View - The result view of item with updated/current information.

References `AndroidApp.DataItem< T >.getAverage()`, `AndroidApp.DataItem< T >.getCurrent()`, `AndroidApp.DataItem< T >.getEnabledAvgMinMax()`, `AndroidApp.DataItem< T >.getMaximum()`, `AndroidApp.DataItem< T >.getMinimum()`, and `AndroidApp.DataItem< T >.getName()`.

```

80                                     {
81
82         ViewHolder holder;
83
84         if (convertView == null)
85         {
86             /* If view does not already exist. */
87             LayoutInflater inflater = (LayoutInflater) context.getSystemService(Context.
LAYOUT_INFLATER_SERVICE);
88             convertView = inflater.inflate(layoutResourceId, parent, false);
89
90             holder = new ViewHolder();
91             holder.name = (TextView) convertView.findViewById(R.id.datalist_name);
92             holder.current = (TextView) convertView.findViewById(R.id.datalist_current);
93             holder.average = (TextView) convertView.findViewById(R.id.datalist_average);
94             holder.minimum = (TextView) convertView.findViewById(R.id.datalist_minimum);
95             holder.maximum = (TextView) convertView.findViewById(R.id.datalist_maximum);
96             convertView.setTag(holder);
97         }
98         else
99         {
100             /* If view already exists. */
101             holder = (ViewHolder) convertView.getTag();
102         }
103
104         DataItem dataItem = getItem(position);
105
106         /* Set our holder with current data of item */
107         holder.name.setText(dataItem.getName());
108
109         Object current = dataItem.getCurrent();
110         if (current != null) {
111             DecimalFormat df = new DecimalFormat("#.####");
112             df.setRoundingMode(RoundingMode.CEILING);
113
114             /* To aid aesthetics rounding is used. */
115             if (current instanceof Double) {
116                 holder.current.setText(df.format(current));
117             } else {
118                 holder.current.setText(current.toString());
119             }
120
121             /*
122              * Displays added functionality if available.
123              * Not all statistics need it, for example averaging of LAT/LNG.
124              */
125             if (dataItem.getEnabledAvgMinMax()) {
126                 holder.average.setText(df.format(dataItem.getAverage()));
127                 holder.minimum.setText(df.format(dataItem.getMinimum()));
128                 holder.maximum.setText(df.format(dataItem.getMaximum()));
129             } else {
130                 holder.average.setText("N/A");
131                 holder.minimum.setText("N/A");
132                 holder.maximum.setText("N/A");
133             }
134         }
135
136         return convertView;
137     }

```

The documentation for this class was generated from the following file:

- `android-app/app/src/main/java/com/jack/motorbikestatistics/DataListAdapter.java`

5.7 AndroidApp.DataListAdapter.ViewHolder Class Reference

Class that holds all data displayed for each ListItem.

5.7.1 Detailed Description

Class that holds all data displayed for each ListItem.

The documentation for this class was generated from the following file:

- android-app/app/src/main/java/com/jack/motorbikestatistics/[DataListAdapter.java](#)

5.8 AndroidApp.JSONHandlerSingleton Class Reference

Singleton class for holding all JSON trip data.

Static Public Member Functions

- static [JSONHandlerSingleton](#) [getInstance](#) ()
Returns the instance of this singleton class.

Public Attributes

- `ArrayList< JSONObject > tripData`

Private Member Functions

- [JSONHandlerSingleton](#) ()
Constructor for instance. Ensures ArrayList is initialised.

Static Private Attributes

- static [JSONHandlerSingleton](#) **ourInstance** = new [JSONHandlerSingleton](#)()

5.8.1 Detailed Description

Singleton class for holding all JSON trip data.

5.8.2 Member Function Documentation

5.8.2.1 getInstance()

```
static JSONHandlerSingleton AndroidApp.JSONHandlerSingleton.getInstance ( ) [inline], [static]
```

Returns the instance of this singleton class.

Returns

[JSONHandleSingleton](#) - Instance of this class.

```
31                                     {
32     return ourInstance;
33 }
```

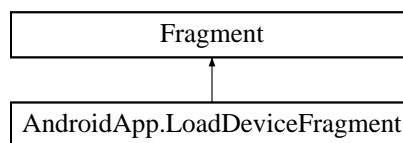
The documentation for this class was generated from the following file:

- [android-app/app/src/main/java/com/jack/motorbikestatistics/JSONHandlerSingleton.java](#)

5.9 [AndroidApp.LoadDeviceFragment](#) Class Reference

UI Class for loading saved trips from device.

Inheritance diagram for [AndroidApp.LoadDeviceFragment](#):



Classes

- class [TripltemListener](#)
Listener used to identify when a trip has been pressed.

Public Member Functions

- [LoadDeviceFragment](#) ()
Constructor for UI fragment.
- View [onCreateView](#) (LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState)
Function called when fragment is shown on UI.
- void [setBTConnection](#) ([BTConnection](#) btConnection)
Setter for current BT connection.

Public Attributes

- final Handler [RXHandler](#)
Handler used for receiving trip names.

Private Member Functions

- final void [addTrip](#) (JSONObject jsonData)
Adds a trip to the ListView specifying name and filesize.

Private Attributes

- [BTConnection](#) [btConnection](#) = null
Current connectected logging device (via bluetooth).
- ArrayList< [Tripltem](#) > [tripList](#)
List of all trips saved on the logging device.
- ArrayAdapter< [Tripltem](#) > [lvAdapter](#)
Array adapter for displaying trips in ListView.

Static Private Attributes

- static final String [NEW_LINE](#) = "\r\n"
New line string.
- static final String [LOAD_TRIP_CHAR](#) = "3"
Command string to be sent to device to load a specific trip.

5.9.1 Detailed Description

UI Class for loading saved trips from device.

5.9.2 Constructor & Destructor Documentation

5.9.2.1 LoadDeviceFragment()

```
AndroidApp.LoadDeviceFragment.LoadDeviceFragment ( ) [inline]
```

Constructor for UI fragment.

Creates a new arraylist of trips that is empty and ready to be filled from the logging device.

```
55         {
56         tripList = new ArrayList<TripItem>( );
57     }
```

5.9.3 Member Function Documentation

5.9.3.1 onCreateView()

```
View AndroidApp.LoadDeviceFragment.onCreateView (
    LayoutInflater inflater,
    ViewGroup container,
    Bundle savedInstanceState ) [inline]
```

Function called when fragment is shown on UI.

Sets up the ListView on the screen using our custom ArrayAdapter specified.

Parameters

<i>inflater</i>	- Inflater used to load fragment on UI.
<i>container</i>	- Container where fragment will be shown.
<i>savedInstanceState</i>	- Information holding past state.

Returns

View - Modified view to display on the UI.

```

72                                     {
73         View myView = inflater.inflate(R.layout.loaddevice_layout, container, false);
74
75         /* Get our ListView via ID, set headers and create our ArrayAdapter for it */
76         ListView lvTripList = (ListView)myView.findViewById(R.id.loaddevice_triplist);
77         lvTripList.setOnItemClickListener(new TripItemListener());
78
79         ViewGroup headerView = (ViewGroup)inflater.inflate(R.layout.trip_list_header, lvTripList, false);
80         lvTripList.addHeaderView(headerView);
81
82         lvAdapter = new TripListAdapter(getActivity(), R.layout.trip_list_item,
tripList);
83         lvTripList.setAdapter(lvAdapter);
84
85         tripList.clear();
86         lvAdapter.notifyDataSetChanged();
87
88         return myView;
89     }

```

5.9.3.2 setBTConnection()

```

void AndroidApp.LoadDeviceFragment.setBTConnection (
    BTConnection btConnection ) [inline]

```

Setter for current BT connection.

Set from main UI activity, allows cross tab communication with the logging device.

Parameters

<i>btConnection</i>	- Logging device bluetooth connection.
---------------------	--

```

99                                     {
100         this.btConnection = btConnection;
101     }

```

5.9.3.3 addTrip()

```

final void AndroidApp.LoadDeviceFragment.addTrip (
    JSONObject jsonData ) [inline], [private]

```

Adds a trip to the ListView specifying name and filesize.

Parameters

<i>jsonData</i>	- JSON object holding trip name and size.
-----------------	---

```

108                                     {
109         try {
110
111             /* Get name and size from json object */
112             String tripName = jsonData.getString("name");
113             int fileSize = jsonData.getInt("size");
114
115             /* Add new trip to our list & notify list view */
116             TripItem newTrip = new TripItem(tripName, fileSize);
117             tripList.add(newTrip);
118             lvAdapter.notifyDataSetChanged();
119
120         } catch (JSONException e) {
121             /* Do nothing */
122         }
123     }

```

5.9.4 Member Data Documentation

5.9.4.1 RXHandler

```
final Handler AndroidApp.LoadDeviceFragment.RXHandler
```

Initial value:

```

= new Handler(Looper.getMainLooper()) {

    @Override
    public void handleMessage(Message msg) {

        Bundle msgData = msg.getData();
        String jsonString = msgData.getString("JSON");

        if (jsonString != null) {

            try {
                JSONObject tmpJSON = new JSONObject(jsonString);
                addTrip(tmpJSON);

            } catch (JSONException e) {

            }

        }

    }

}

```

Handler used for receiving trip names.

Receives trip information from the bluetooth connection thread. Handler has to be used as system is multithreaded.

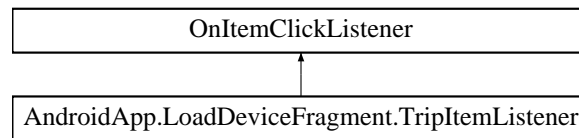
The documentation for this class was generated from the following file:

- android-app/app/src/main/java/com/jack/motorbikestatistics/[LoadDeviceFragment.java](#)

5.10 AndroidApp.LoadDeviceFragment.TripItemListener Class Reference

Listener used to identify when a trip has been pressed.

Inheritance diagram for AndroidApp.LoadDeviceFragment.TripItemListener:



Public Member Functions

- void [onItemClick](#) (AdapterView<?> parent, View view, int position, long id)
Loads a trip the user has specified.

5.10.1 Detailed Description

Listener used to identify when a trip has been pressed.

5.10.2 Member Function Documentation

5.10.2.1 onItemClick()

```

void AndroidApp.LoadDeviceFragment.TripItemListener.onItemClick (
    AdapterView<?> parent,
    View view,
    int position,
    long id ) [inline]
  
```

Loads a trip the user has specified.

User has selected a trip via the ListView, method switches to the statistic fragment and sends a message to logging device to load the specified trip (via name).

References [AndroidApp.TripItem.getTripName\(\)](#), [AndroidApp.BTConnection.isConnected\(\)](#), [AndroidApp.RealtimeFragment.RXHandler](#), [AndroidApp.BTConnection.setRXHandler\(\)](#), and [AndroidApp.BTConnection.txHandler](#).

```

138
139
140         if (btConnection != null && btConnection.
isConnected()) {
141             TripItem tripItem = (TripItem) parent.getItemAtPosition(position);
142
143             /*
144              * Create a new statistics fragment.
145              * This will receive the stored data from the logging device.
146              */
147             RealtimeFragment statFragment = new RealtimeFragment();
148             btConnection.setRXHandler(statFragment.RXHandler);
149
150             /* Transmit over the name of the trip we want to load */
151             Message message = new Message();
152             message.obj = (String) LOAD_TRIP_CHAR + tripItem.getTripName();
153             message.setTarget(btConnection.txHandler);
154             message.sendToTarget();
155
156             FragmentManager fragmentManager = getFragmentManager();
157             fragmentManager.beginTransaction()
158                 .replace(R.id.content_frame, statFragment)
159                 .commit();
160         }
161     }

```

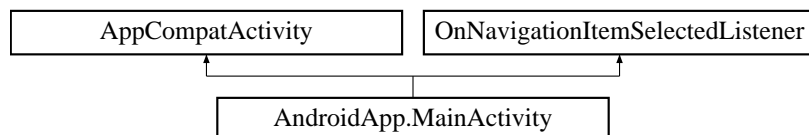
The documentation for this class was generated from the following file:

- android-app/app/src/main/java/com/jack/motorbikestatistics/[LoadDeviceFragment.java](#)

5.11 AndroidApp.MainActivity Class Reference

Main activity class for fragment navigation.

Inheritance diagram for AndroidApp.MainActivity:



Public Member Functions

- void [onBackPressed](#) ()
Responsible for closing navigation drawer when back button pressed.
- boolean [onNavigationItemSelectedListener](#) (MenuItem item)
Changes active fragment when a tab has been pressed.

Protected Member Functions

- void [onCreate](#) (Bundle savedInstanceState)
Function called when main activity is loaded.

Private Attributes

- `RealtimeFragment rtFragment` = null
UI fragment for realtime statistic display.
- `LoadDeviceFragment ldFragment` = null
UI fragment for loading previous trips.
- `PairDeviceFragment pdFragment` = null
UI fragment for pairing to a logging device.

Static Private Attributes

- static final String `REALTIME_CHAR` = "1"
Command for switching to realtime logging.
- static final String `LIST_SAVED_CHAR` = "2"
Command for loading all saved trip details.

5.11.1 Detailed Description

Main activity class for fragment navigation.

5.11.2 Member Function Documentation

5.11.2.1 onCreate()

```
void AndroidApp.MainActivity.onCreate (
    Bundle savedInstanceState ) [inline], [protected]
```

Function called when main activity is loaded.

Procedure is called when application is first started, sets up UI and creates relevant fragments.

Parameters

<code>savedInstanceState</code>	- Information holding last previous state.
---------------------------------	--

```
55                                     {
56     super.onCreate(savedInstanceState);
57     setContentView(R.layout.activity_main);
58
59     Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
60     setSupportActionBar(toolbar);
61
62     DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
63     ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(
64         this, drawer, toolbar, R.string.navigation_drawer_open, R.string.navigation_drawer_close);
65     drawer.setDrawerListener(toggle);
66     toggle.syncState();
67
68     NavigationView navigationView = (NavigationView) findViewById(R.id.nav_view);
69     navigationView.setNavigationItemSelectedListener(this);
```



```

70
71     /* Create our fragments for different sections of UI */
72     rtFragment = new RealtimeFragment();
73     ldFragment = new LoadDeviceFragment();
74     pdFragment = new PairDeviceFragment();
75 }

```

5.11.2.2 onNavigationItemSelected()

```

boolean AndroidApp.MainActivity.onNavigationItemSelected (
    MenuItem item ) [inline]

```

Changes active fragment when a tab has been pressed.

Responsible for changing to the new chosen fragment on the UI. Opening of realtime and loaddevice fragments not possible when not connected to the logging device.

Method also responsible for change system state machine on the logging device, this is done by transmitting command code.

Parameters

<i>item</i>	- New selected fragment/tab to display.
-------------	---

References `AndroidApp.PairDeviceFragment.getBTConnection()`, `AndroidApp.BTConnection.isConnected()`, `AndroidApp.LoadDeviceFragment.RXHandler`, `AndroidApp.RealtimeFragment.RXHandler`, `AndroidApp.LoadDeviceFragment.setBTConnection()`, `AndroidApp.BTConnection.setRXHandler()`, and `AndroidApp.BTConnection.txHandler`.

```

105                                     {
106
107         Fragment activeFragment = null;
108
109         /* Handle navigation view clicks here */
110         FragmentManager fragmentManager = getFragmentManager();
111         int id = item.getItemId();
112
113         switch (id) {
114             case R.id.nav_realtime: {
115                 /* Get our bluetooth connection from pairing fragment */
116                 BTConnection btConn = pdFragment.getBTConnection();
117
118                 if (btConn != null && btConn.isConnected()) {
119                     /* We set our RX handler and also send our command to indicate mode change */
120                     btConn.setRXHandler(rtFragment.
RXHandler);
121                     Message message = new Message();
122                     message.obj = (String) REALTIME_CHAR;
123                     message.setTarget(btConn.txHandler);
124                     message.sendToTarget();
125
126                     /* Change to our new active fragment */
127                     activeFragment = rtFragment;
128                 } else {
129                     /* Indicate that we are not connected to device */
130                     View rootView = findViewById(R.id.content_main);
131                     Snackbar.make(rootView, "Please connect to a device first.", Snackbar.LENGTH_LONG)
132                         .setAction("Action", null).show();
133                 }
134                 break;
135             }
136
137             case R.id.nav_loaddevice: {
138                 /* Get our bluetooth connection from pairing fragment */
139                 BTConnection btConn = pdFragment.getBTConnection();

```

```

140
141         if (btConn != null && btConn.isConnected()) {
142             /* We set our RX handler and also send our command to indicate mode change */
143             ldFragment.setBTConnection(btConn);
144
145             btConn.setRXHandler(ldFragment.RXHandler);
146             Message message = new Message();
147             message.obj = (String) LIST_SAVED_CHAR;
148             message.setTarget(btConn.txHandler);
149             message.sendToTarget();
150
151             /* Change to our new active fragment */
152             activeFragment = ldFragment;
153         } else {
154             /* Indicate that we are not connected to device */
155             View rootView = findViewById(R.id.content_main);
156             Snackbar.make(rootView, "Please connect to a device first.", Snackbar.LENGTH_LONG)
157                 .setAction("Action", null).show();
158         }
159         break;
160     }
161
162     case R.id.nav_pairdevice: {
163         activeFragment = pdFragment;
164     }
165
166 }
167
168 if (activeFragment != null) {
169     /* Replaces content frame with newly selected one */
170     fragmentManager.beginTransaction()
171         .replace(R.id.content_frame, activeFragment)
172         .commit();
173 }
174
175 DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
176 drawer.closeDrawer(GravityCompat.START);
177 return (activeFragment != null);
178 }

```

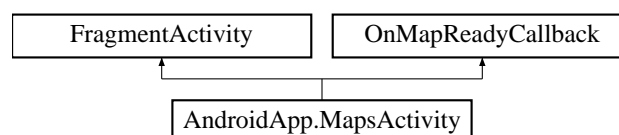
The documentation for this class was generated from the following file:

- [android-app/app/src/main/java/com/jack/motorbikestatistics/MainActivity.java](#)

5.12 AndroidApp.MapsActivity Class Reference

Maps activity class for displaying map data.

Inheritance diagram for AndroidApp.MapsActivity:



Classes

- class [StatisticWindowAdapter](#)
Adapter used for displaying statistics at a certain marker that user has clicked on.
- class [ZoomToggleListener](#)
Listener class for making markers invisible when zoomed out.

Public Member Functions

- void [onMapReady](#) (GoogleMap googleMap)
Manipulates the map once available.

Protected Member Functions

- void [onCreate](#) (Bundle savedInstanceState)
Fills our maps array with points to plot on the map.

Private Member Functions

- JSONObject [findJSONByLatLng](#) (LatLng position)
Finds JSONObject from ArrayList via LAT/LNG coordinates.
- float [calcDistance](#) (LatLng start, LatLng end)
Calculates the absolute distance between two points.
- float [convertDpToPixel](#) (float dp, Context context)
Function for converting raw DP value to pixels.
- BitmapDescriptor [getBitmapDescriptor](#) (int id)
Created a valid bitmap descriptor from a drawable resource ID.

Private Attributes

- GoogleMap [mMap](#)
Google maps object for plotting.
- ArrayList< JSONObject > [tripData](#)
ArrayList holding all trip data.
- ArrayList< Marker > **markerList**

5.12.1 Detailed Description

Maps activity class for displaying map data.

5.12.2 Member Function Documentation

5.12.2.1 onCreate()

```
void AndroidApp.MapsActivity.onCreate (
    Bundle savedInstanceState ) [inline], [protected]
```

Fills our maps array with points to plot on the map.

Called when maps activity is first started. Responsible for making sure we have points to plot.

Parameters

<i>savedInstanceState</i>	- Information holding last previous state.
---------------------------	--

References `AndroidApp.JSONHandlerSingleton.getInstance()`.

```

67                                     {
68         super.onCreate(savedInstanceState);
69         setContentView(R.layout.activity_maps);
70         // Obtain the SupportMapFragment and get notified when the map is ready to be used.
71         SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()
72             .findFragmentById(R.id.map);
73
74         /* Clear our marker list for new instances. */
75         markerList = new ArrayList<Marker>();
76
77         /* Get the tripData from our JSON handler singleton. */
78         tripData = JSONHandlerSingleton.getInstance().tripData;
79
80         mapFragment.getMapAsync(this);
81     }

```

5.12.2.2 findJSONByLatLng()

```

JSONObject AndroidApp.MapsActivity.findJSONByLatLng (
    LatLng position ) [inline], [private]

```

Finds `JSONObject` from `ArrayList` via LAT/LNG coordinates.

Parameters

<i>position</i>	- Latitude and Longitude position.
-----------------	------------------------------------

Returns

`JSONObject` - The found JSON object.

References `gpsJSON`.

```

89                                     {
90         JSONObject result = null;
91
92         for (int i = 0; i < tripData.size(); i++) {
93             JSONObject tmpJSON = tripData.get(i);
94
95             try {
96                 JSONObject gpsJSON = tmpJSON.getJSONObject("gps");
97
98                 Double latitude = gpsJSON.getDouble("lat");
99                 Double longitude = gpsJSON.getDouble("lng");
100
101                 /* Check to see if latitude and longitudes match */
102                 if ((latitude == position.latitude) && (longitude == position.longitude)) {
103                     result = tmpJSON;
104                     break;
105                 }
106             } catch (JSONException e) {
107                 /* Do nothing */
108             }
109         }
110
111         return result;
112     }
113 }

```

5.12.2.3 calcDistance()

```
float AndroidApp.MapActivity.calcDistance (
    LatLng start,
    LatLng end ) [inline], [private]
```

Calculates the absolute distance between two points.

Distance is as the crow flies and not via streets etc.

Parameters

<i>start</i>	- Start position.
<i>end</i>	- End position.

Returns

float - Distance between points in metres.

```
124 {
125     float[] results = new float[1];
126
127     Location.distanceBetween(start.latitude, start.longitude, end.latitude, end.longitude, results);
128     return results[0];
129 }
```

5.12.2.4 onMapReady()

```
void AndroidApp.MapActivity.onMapReady (
    GoogleMap googleMap ) [inline]
```

Manipulates the map once available.

This callback is triggered when the map is ready to be used. This is where we can add markers or lines.

If Google Play services is not installed on the device, the user will be prompted to install it inside the SupportMapFragment. This method will only be triggered once the user has installed Google Play services and returned to the app.

Parameters

<i>googleMap</i>	- Our map object ready to manipulate.
------------------	---------------------------------------

References gpsJSON.

```
144 {
145     mMap = googleMap;
```

```

146
147     mMap.setMapType(GoogleMap.MAP_TYPE_HYBRID);
148
149     /* Set our info window adapter class that is shown when marker clicked */
150     mMap.setInfoWindowAdapter(new StatisticWindowAdapter());
151
152     /* Set our listener class for adjusting visibility when zoomed. */
153     mMap.setOnCameraMoveListener(new ZoomToogleListener());
154
155     /* If we have no data don't bother plotting points */
156     if (tripData.size() != 0)
157     {
158         /* lineOpts will store our route */
159         PolylineOptions lineOpts = new PolylineOptions();
160         lineOpts.color(Color.parseColor( "#CCF44242"));
161         lineOpts.width(18);
162         lineOpts.visible(true);
163
164         try
165         {
166             LatLng lastMarker = null;
167
168             /* Plot every point in the our JSONObject array */
169             for (int i = 0; i < tripData.size(); i++)
170             {
171                 JSONObject rootJSON = tripData.get(i);
172                 JSONObject gpsJSON = rootJSON.getJSONObject("gps");
173
174                 Double lat = gpsJSON.getDouble("lat");
175                 Double lng = gpsJSON.getDouble("lng");
176                 LatLng location = new LatLng(lat, lng);
177
178                 /* Don't add location with invalid lat & lng. */
179                 if (location.latitude != 0.00 && location.longitude != 0.00) {
180
181                     /* Add this location to our trip line */
182                     lineOpts.add(location);
183
184                     /*
185                      * Check if distance between this point and
186                      * last marker is greater than 5m otherwise don't add marker.
187                      * Adding markers every 5 metres prevents the map being spammed with
188                      * thousands of readings.
189                      */
190                     if ((lastMarker == null) || (calcDistance(location, lastMarker) > 5)) {
191                         /* Only add a marker if the gps data is valid */
192                         if (gpsJSON.getBoolean("gps_valid") == true) {
193                             MarkerOptions markerOptions = new MarkerOptions();
194                             markerOptions.position(location);
195                             markerOptions.title("Reading: " + i);
196                             markerOptions.icon(getBitmapDescriptor(R.drawable.
ic_expand_more_white_24px));
197                             markerOptions.visible(false);
198
199                             /* Add our marker to map and to list. */
200                             markerList.add(mMap.addMarker(markerOptions));
201
202                             lastMarker = location;
203
204                             /* Changes camera to point to newest marker */
205                             mMap.animateCamera(CameraUpdateFactory.newLatLngZoom(location, 16));
206                         }
207                     }
208                 }
209             }
210             mMap.addPolyline(lineOpts);
211         }
212         catch (JSONException e)
213         {
214             /* Do nothing */
215         }
216     }
217 }
218

```

5.12.2.5 convertDpToPixel()

```

float AndroidApp.MapsActivity.convertDpToPixel (
    float dp,
    Context context ) [inline], [private]

```

Function for converting raw DP value to pixels.

Parameters

<i>dp</i>	- DP value.
<i>context</i>	- Context of application.

Returns

float - Value in pixels.

```

227                                     {
228         Resources resources = context.getResources();
229         DisplayMetrics metrics = resources.getDisplayMetrics();
230         float px = dp * ((float)metrics.densityDpi / DisplayMetrics.DENSITY_DEFAULT);
231         return px;
232     }

```

5.12.2.6 getBitmapDescriptor()

```

BitmapDescriptor AndroidApp.MainActivity.getBitmapDescriptor (
    int id ) [inline], [private]

```

Created a valid bitmap descriptor from a drawable resource ID.

Parameters

<i>id</i>	- Drawable resource ID.
-----------	-------------------------

Returns

BitmapDescriptor - Bitmap image converted from VectorAsset.

```

240                                     {
241
242         Context context = getApplicationContext();
243         Drawable vectorDrawable = ContextCompat.getDrawable(context, id);
244         int h = ((int) convertDpToPixel(42, context));
245         int w = ((int) convertDpToPixel(25, context));
246         vectorDrawable.setBounds(0, 0, w, h);
247         Bitmap bm = Bitmap.createBitmap(w, h, Bitmap.Config.ARGB_8888);
248         Canvas canvas = new Canvas(bm);
249         vectorDrawable.draw(canvas);
250         return BitmapDescriptorFactory.fromBitmap(bm);
251     }

```

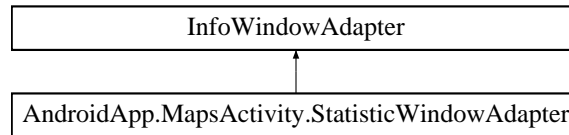
The documentation for this class was generated from the following file:

- android-app/app/src/main/java/com/jack/motorbikestatistics/[MapsActivity.java](#)

5.13 AndroidApp.MapsActivity.StatisticWindowAdapter Class Reference

Adapter used for displaying statistics at a certain marker that user has clicked on.

Inheritance diagram for AndroidApp.MapsActivity.StatisticWindowAdapter:



Public Member Functions

- View [getInfoWindow](#) (Marker marker)
We don't want to use default information window.
- View [getInfoContents](#) (Marker marker)
Displays statistics at a marker that the user has clicked on.

5.13.1 Detailed Description

Adapter used for displaying statistics at a certain marker that user has clicked on.

5.13.2 Member Function Documentation

5.13.2.1 getInfoContents()

```
View AndroidApp.MapsActivity.StatisticWindowAdapter.getInfoContents (
    Marker marker ) [inline]
```

Displays statistics at a marker that the user has clicked on.

Parameters

<i>marker</i>	- The marker the user has clicked on.
---------------	---------------------------------------

Returns

View - Updated view showing information.

References [gpsJSON](#), [orientJSON](#), and [timeJSON](#).

```
319                                     {
320
```



```

321         View v = getLayoutInflater().inflate(R.layout.map_marker_info, null);
322
323         /* Get latitude and longitude from marker */
324         LatLng latLng = marker.getPosition();
325
326         /* Find the JSONObject relating to this location */
327         JSONObject rootJSON = findJSONByLatLng(latLng);
328         if (rootJSON != null) {
329             try {
330                 JSONObject gpsJSON = rootJSON.getJSONObject("gps");
331                 JSONObject orientJSON = rootJSON.getJSONObject("orientation");
332                 JSONObject timeJSON = rootJSON.getJSONObject("time");
333
334                 /* Set latitude and longitude in info window */
335                 TextView tvLatLng = (TextView)v.findViewById(R.id.map_latlng);
336                 tvLatLng.setText("Lat/Lng: " + Double.toString(latLng.latitude) + "/"
337                     + Double.toString(latLng.longitude));
338
339                 /* Set time */
340                 TextView tvTime = (TextView)v.findViewById(R.id.map_time);
341                 Calendar cal = Calendar.getInstance();
342                 cal.clear();
343                 cal.set(Calendar.YEAR, timeJSON.getInt("year"));
344                 cal.set(Calendar.MONTH, timeJSON.getInt("month"));
345                 cal.set(Calendar.DATE, timeJSON.getInt("day"));
346
347                 cal.set(Calendar.HOUR, timeJSON.getInt("hour"));
348                 cal.set(Calendar.MINUTE, timeJSON.getInt("minute"));
349                 cal.set(Calendar.SECOND, timeJSON.getInt("second"));
350                 cal.set(Calendar.MILLISECOND, timeJSON.getInt("centiseconds") * 10);
351
352                 /* Create format for date and times then add to view */
353                 DateFormat dateFormat = new SimpleDateFormat("dd/MM/yy HH:mm:ss.SS");
354                 tvTime.setText("Time: " + dateFormat.format(cal.getTime()));
355
356                 /* Velocity & Altitude */
357                 TextView tvVelocity = (TextView)v.findViewById(R.id.map_velocity);
358                 tvVelocity.setText("Velocity: " + gpsJSON.getDouble("vel_mph") + "mph");
359
360                 TextView tvAltitude = (TextView)v.findViewById(R.id.map_altitude);
361                 tvAltitude.setText("Altitude: " + gpsJSON.getDouble("alt_ft") + "ft");
362
363                 /* Orientation */
364                 TextView tvPitch = (TextView)v.findViewById(R.id.map_pitch);
365                 tvPitch.setText("Pitch Angle: " + orientJSON.getDouble("pitch") + "\u00b0");
366
367                 TextView tvRoll = (TextView)v.findViewById(R.id.map_roll);
368                 tvRoll.setText("Roll/Lean Angle: " + orientJSON.getDouble("roll") + "\u00b0");
369
370             } catch (JSONException e) {
371                 marker.hideInfoWindow();
372             }
373         } else {
374             /* If unable to find relating we hide the info window */
375             marker.hideInfoWindow();
376         }
377     }
378     return v;
379 }

```

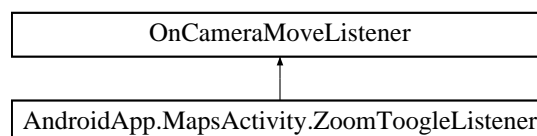
The documentation for this class was generated from the following file:

- [android-app/app/src/main/java/com/jack/motorbikestatistics/MapsActivity.java](#)

5.14 AndroidApp.MapsActivity.ZoomToggleListener Class Reference

Listener class for making markers invisible when zoomed out.

Inheritance diagram for AndroidApp.MapsActivity.ZoomToggleListener:



Public Member Functions

- void [onCameraMove](#) ()
Listener function that tooggles all markers visibility.

Private Member Functions

- void [setMarkersVisible](#) (boolean visible)
Sets all markers either visible or invisible.

Private Attributes

- boolean **currentVisible** = false

5.14.1 Detailed Description

Listener class for making markers invisible when zoomed out.

5.14.2 Member Function Documentation

5.14.2.1 [setMarkersVisible\(\)](#)

```
void AndroidApp.MapsActivity.ZoomToogleListener.setMarkersVisible (
    boolean visible ) [inline], [private]
```

Sets all markers either visible or invisible.

Parameters

<i>visible</i>	- Value to set to.
----------------	--------------------

```
285                                     {
286         if (currentVisible != visible) {
287             for (int i = 0; i < markerList.size(); i++) {
288
289                 Marker tmp = markerList.get(i);
290                 tmp.setVisible(visible);
291                 currentVisible = visible;
292             }
293         }
294     }
```

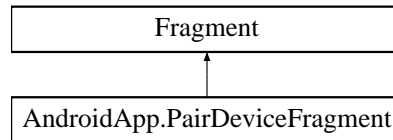
The documentation for this class was generated from the following file:

- android-app/app/src/main/java/com/jack/motorbikestatistics/[MapsActivity.java](#)

5.15 AndroidApp.PairDeviceFragment Class Reference

UI Class for discovering, pairing and connecting to the logging device.

Inheritance diagram for AndroidApp.PairDeviceFragment:



Classes

- class [DeviceItemListener](#)
Listener for when a ListView item is pressed (to connect).
- class [DiscoverButtonListener](#)
Listener for when discovery button is pressed.
- class [DiscoverReceiver](#)
Receiver for when a new device is discovered.

Public Member Functions

- [PairDeviceFragment](#) ()
Constructor for UI fragment.
- View [onCreateView](#) (LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState)
Function called when fragment is shown on UI.
- [BTConnection](#) [getBTConnection](#) ()
Getter for getting current connected device.

Private Member Functions

- void [getNeededPrivileges](#) ()
Prompts user for needed permissions of this application.

Private Attributes

- boolean [firstRun](#) = true
Check variable used to stop ListView from being re-populated.
- ToggleButton [btnScan](#)
Scan button, used for toggling discovery.
- BluetoothAdapter [btAdapter](#) = null
Mobile's bluetooth adapter.
- ArrayList< [BTDeviceItem](#) > [btDeviceList](#)
List of all devices, unpaired, paired & connected.
- ArrayList< [BTDeviceItem](#) > [btPairedList](#)
List of only paired devices.
- ArrayAdapter< [BTDeviceItem](#) > [lvAdapter](#)
UI adapter for ListView that displays bluetooth devices.
- [BTDeviceItem](#) [btConnectedDevice](#) = null
Applications connected logging device.
- [DiscoverReceiver](#) [btReceiver](#) = null
Receiver class for when new device discovered.

Static Private Attributes

- static final int `REQUEST_BLUETOOTH` = 1
Request code for activating bluetooth.
- static final String `CONNECTED_STATUS` = "connected"
Status to change `BTDeviceItem` to when connected.
- static final int `BT_DISABLED_ICON` = R.drawable.ic_bluetooth_disabled_black_24px
Icon ID to use when device is not connected.

5.15.1 Detailed Description

UI Class for discovering, pairing and connecting to the logging device.

5.15.2 Constructor & Destructor Documentation

5.15.2.1 PairDeviceFragment()

```
AndroidApp.PairDeviceFragment.PairDeviceFragment ( ) [inline]
```

Constructor for UI fragment.

Get's the mobile's bluetooth adapter and sets up our lists of used for holding devices.

```
85     {
86         /* Get bluetooth adapter for device & create device arrays */
87         btAdapter = BluetoothAdapter.getDefaultAdapter();
88         btDeviceList = new ArrayList<BTDeviceItem>();
89         btPairedList = new ArrayList<BTDeviceItem>();
90         btReceiver = new DiscoverReceiver();
91     }
```

5.15.3 Member Function Documentation

5.15.3.1 onCreateView()

```
View AndroidApp.PairDeviceFragment.onCreateView (
    LayoutInflater inflater,
    ViewGroup container,
    Bundle savedInstanceState ) [inline]
```

Function called when fragment is shown on UI.

Sets up the UI ListView and Buttons. Add all paired devices for the bluetooth adapter to the ListView.

Parameters

<i>inflater</i>	- Inflater used for displaying view.
<i>container</i>	- Container that the view will be displayed on.
<i>savedInstanceState</i>	- Last known state of this fragment.

Returns

View - The UI view of this fragment.

References `AndroidApp.BTDeviceItem.getConnection()`, `AndroidApp.BTConnection.isConnected()`, and `AndroidApp.BTConnection.isRunning()`.

```

106
107
108     View myView = inflater.inflate(R.layout.pairdevice_layout, container, false);
109
110     /* Request needed privileges for bluetooth to work */
111     getNeededPrivileges();
112
113     /* Set our variables for UI buttons */
114     btnScan = (ToggleButton)myView.findViewById(R.id.pairdevice_search);
115     btnScan.setOnCheckedChangeListener(new DiscoverButtonListener());
116
117     ListView lvDevices = (ListView)myView.findViewById(R.id.pairdevice_deviceList);
118     lvDevices.setOnItemClickListener(new DeviceItemListener());
119
120     lvAdapter = new BTDeviceListAdapter(getActivity(), R.layout.device_list_item,
121     btDeviceList);
122     lvDevices.setAdapter(lvAdapter);
123
124     /* Check and set up bluetooth adapter */
125     if (btAdapter == null)
126     {
127         Toast.makeText(getActivity().getApplicationContext(),
128             "This device has no bluetooth adapter", Toast.LENGTH_LONG).show();
129     }
130     else
131     {
132         /* Check to see if connected device still is connected */
133         if (btConnectedDevice != null)
134         {
135             if (!btConnectedDevice.getConnection().
136             isConnected() ||
137             !btConnectedDevice.getConnection().
138             isRunning())
139             {
140                 btConnectedDevice = null;
141             }
142         }
143
144         /* firstRun check to list from being re-populated */
145         if (firstRun)
146         {
147             firstRun = false;
148
149             /* Enable bluetooth adapter if disabled */
150             if (!btAdapter.isEnabled())
151             {
152                 Intent enableBT = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
153                 startActivityForResult(enableBT, REQUEST_BLUETOOTH);
154             }
155
156             while (!btAdapter.isEnabled())
157             {
158                 /* Wait for BT to be enabled */
159             }
160
161             /* Add all paired devices to list */
162             Set<BluetoothDevice> pairedDevices = btAdapter.getBondedDevices();
163             if (pairedDevices.size() > 0)
164             {
165                 for (BluetoothDevice device : pairedDevices)
166                 {
167                     BTDeviceItem newDevice =
168                         new BTDeviceItem(device, "paired", BT_DISABLED_ICON);
169                     btPairedList.add(newDevice);
170                 }
171             }
172         }
173     }
174 }

```

```

167         }
168     }
169     btDeviceList.addAll(btPairedList);
170 }
171
172 }
173
174 return myView;
175 }

```

5.15.3.2 getBTConnection()

`BTConnection` `AndroidApp.PairDeviceFragment.getBTConnection ()` [inline]

Getter for getting current connected device.

Returns

`BTConnection` - Bluetooth device (logging device).

References `AndroidApp.BTDeviceItem.getConnection()`.

```

182 {
183     if (btConnectedDevice != null) {
184         return btConnectedDevice.getConnection();
185     } else {
186         return null;
187     }
188 }

```

5.15.3.3 getNeededPrivileges()

`void` `AndroidApp.PairDeviceFragment.getNeededPrivileges ()` [inline], [private]

Prompts user for needed permissions of this application.

Due to android using a permissions/access method this method parses through each permission needed and prompts the user to accept.

```

197 {
198     final int REQUEST_CODE = 5;
199
200     boolean permsGranted = true;
201     String[] permsToRequest =
202     {
203         Manifest.permission.BLUETOOTH_ADMIN,
204         Manifest.permission.BLUETOOTH,
205         Manifest.permission.ACCESS_FINE_LOCATION,
206         Manifest.permission.ACCESS_COARSE_LOCATION
207     };
208
209     for (String permission: permsToRequest)
210     {
211         permsGranted &= (ContextCompat.checkSelfPermission(getActivity(), permission) == PackageManager
212             .PERMISSION_GRANTED);
213     }
214     if (!permsGranted)
215     {
216         ActivityCompat.requestPermissions(getActivity(), permsToRequest, REQUEST_CODE);
217     }
218 }

```

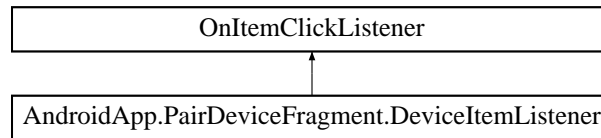
The documentation for this class was generated from the following file:

- `android-app/app/src/main/java/com/jack/motorbikestatistics/PairDeviceFragment.java`

5.16 AndroidApp.PairDeviceFragment.DeviceItemClickListener Class Reference

Listener for when a ListView item is pressed (to connect).

Inheritance diagram for AndroidApp.PairDeviceFragment.DeviceItemClickListener:



Public Member Functions

- void [onItemClick](#) (AdapterView<?> parent, View view, int position, long id)
Function called when user wants to connect to a device.

5.16.1 Detailed Description

Listener for when a ListView item is pressed (to connect).

5.16.2 Member Function Documentation

5.16.2.1 onItemClick()

```

void AndroidApp.PairDeviceFragment.DeviceItemClickListener.onItemClick (
    AdapterView<?> parent,
    View view,
    int position,
    long id ) [inline]
  
```

Function called when user wants to connect to a device.

Discovery is turned off to stop power wastage. A new connection thread is then created which is responsible for parsing receive, and transmission requests from other fragments.

Parameters

<i>parent</i>	- The parent ListView.
<i>view</i>	- Current view of the ListItem.
<i>position</i>	- Index of item pressed in ListView.
<i>id</i>	- ID of the ListItem.

References [AndroidApp.BTDeviceItem.getConnection\(\)](#), [AndroidApp.BTDeviceItem.getDevice\(\)](#), [AndroidApp.BTDeviceItem.isConnected\(\)](#), [AndroidApp.BTDeviceItem.setConnection\(\)](#), [AndroidApp.BTDeviceItem.setIconID\(\)](#),

and `AndroidApp.BTDeviceItem.setStatus()`.

```

305
306
307         BTDeviceItem deviceItem = (BTDeviceItem)parent.getItemAtPosition(position);
308
309         /* Check if there is already a connection between devices */
310         if ((deviceItem.getConnection() == null) ||
311             (!deviceItem.getConnection().isConnected()))
312         {
313             if (btAdapter.isDiscovering())
314             {
315                 /* Cancel discovery is still enabled */
316                 btnScan.setChecked(false);
317                 btAdapter.cancelDiscovery();
318             }
319
320             try
321             {
322                 Toast.makeText(parent.getContext(), "Connecting to: " +
323                     deviceItem.getDevice().getName(), Toast.LENGTH_SHORT).show();
324
325                 /* Create a new BTConnection item with no RX handler */
326                 BTConnection newConn = new BTConnection(deviceItem.getDevice());
327
328                 /* Execute the 'run' procedure in object in new thread */
329                 Thread tmpThread = new Thread(newConn);
330                 tmpThread.start();
331
332                 /* Add set connection and add item to listview */
333                 deviceItem.setConnection(newConn);
334                 btConnectedDevice = deviceItem;
335
336                 /* Update status and icon in list view */
337                 deviceItem.setIconID(R.drawable.ic_bluetooth_connected_black_24px);
338                 deviceItem.setStatus(CONNECTED_STATUS);
339                 lvAdapter.notifyDataSetChanged();
340             }
341             catch (IOException e)
342             {
343                 Toast.makeText(parent.getContext(), "Unable to connect: " +
344                     e.toString(), Toast.LENGTH_SHORT).show();
345             }
346         }
347     }

```

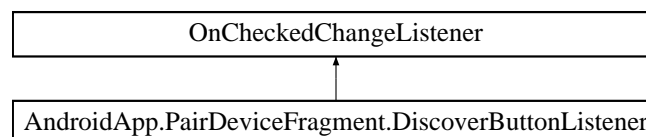
The documentation for this class was generated from the following file:

- [android-app/app/src/main/java/com/jack/motorbikestatistics/PairDeviceFragment.java](#)

5.17 AndroidApp.PairDeviceFragment.DiscoverButtonListener Class Reference

Listener for when discovery button is pressed.

Inheritance diagram for `AndroidApp.PairDeviceFragment.DiscoverButtonListener`:



Public Member Functions

- void [onCheckedChanged](#) (CompoundButton buttonView, boolean isChecked)
Function for handling when discover toggle button pressed.

5.17.1 Detailed Description

Listener for when discovery button is pressed.

5.17.2 Member Function Documentation

5.17.2.1 onCheckedChanged()

```
void AndroidApp.PairDeviceFragment.DiscoverButtonListener.onCheckedChanged (
    CompoundButton buttonView,
    boolean isChecked ) [inline]
```

Function for handling when discover toggle button pressed.

If toggled on it bluetooth adapter is turned to discover mode. If toggled off bluetooth adapter is turn off of discover mode.

Parameters

<i>buttonView</i>	- Current view of the toggle button.
<i>isChecked</i>	- The new state of the toggle button.

```
262                                     {
263
264     IntentFilter filter = new IntentFilter(BluetoothDevice.ACTION_FOUND);
265     if (isChecked)
266     {
267         /* Clear listview, add previous paired items, start discovery */
268         lvAdapter.clear();
269         lvAdapter.addAll(btPairedList);
270
271         if (btConnectedDevice != null)
272             lvAdapter.add(btConnectedDevice);
273
274         getActivity().registerReceiver(btReceiver, filter);
275         btAdapter.startDiscovery();
276     }
277     else
278     {
279         /* Stop searching for new devices */
280         getActivity().unregisterReceiver(btReceiver);
281         btAdapter.cancelDiscovery();
282     }
283 }
```

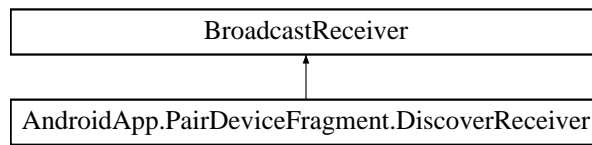
The documentation for this class was generated from the following file:

- android-app/app/src/main/java/com/jack/motorbikestatistics/[PairDeviceFragment.java](#)

5.18 AndroidApp.PairDeviceFragment.DiscoverReceiver Class Reference

Receiver for when a new device is discovered.

Inheritance diagram for `AndroidApp.PairDeviceFragment.DiscoverReceiver`:



Public Member Functions

- void [onReceive](#) (Context context, Intent intent)
When a BT device is found, adds the device to the ListView.

5.18.1 Detailed Description

Receiver for when a new device is discovered.

5.18.2 Member Function Documentation

5.18.2.1 onReceive()

```
void AndroidApp.PairDeviceFragment.DiscoverReceiver.onReceive (
    Context context,
    Intent intent ) [inline]
```

When a BT device is found, adds the device to the ListView.

Parameters

<i>context</i>	- Context that the application is running in.
<i>intent</i>	- Intent holding the device object.

```

231                                     {
232         String action = intent.getAction();
233
234         /* Check to see if found device */
235         if (BluetoothDevice.ACTION_FOUND.equals(action))
236         {
237             BluetoothDevice device = intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
238
239             /* Create new device item and add to list */
240             BTDeviceItem newDevice = new BTDeviceItem(device, "unpaired",
241                 BT_DISABLED_ICON);
241             lvAdapter.add(newDevice);
242             lvAdapter.notifyDataSetChanged();
243         }
244     }
```

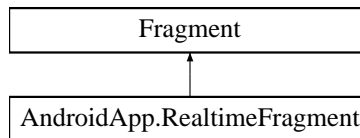
The documentation for this class was generated from the following file:

- [android-app/app/src/main/java/com/jack/motorbikestatistics/PairDeviceFragment.java](#)

5.19 AndroidApp.RealtimeFragment Class Reference

UI Class for viewing data sent from the logging device.

Inheritance diagram for AndroidApp.RealtimeFragment:



Classes

- class [MapButtonListener](#)
Listener for starting a map activity when button pressed.

Public Member Functions

- [RealtimeFragment](#) ()
Constructor for UI fragment.
- View [onCreateView](#) (LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState)
Function called when fragment is shown on UI.

Public Attributes

- final Handler [RXHandler](#)
Handler used for receiving statistics via bluetooth.

Private Member Functions

- final void [newData](#) (JSONObject jsonData)
Function for adding new statistics when received via bluetooth.

Private Attributes

- TextView [textStatus](#)
TextView to show amount of logs received.
- [SetOfDataItems](#) [dataList](#)
Array holding each statistic that device is measuring.
- ArrayAdapter< [DataItem](#) > [lvAdapter](#)
Adapter used for displaying statistics in the ListView.

Static Private Attributes

- static final String [NEW_LINE](#) = "\r\n"
String for new line parsing.

5.19.1 Detailed Description

UI Class for viewing data sent from the logging device.

5.19.2 Constructor & Destructor Documentation

5.19.2.1 RealtimeFragment()

AndroidApp.RealtimeFragment.RealtimeFragment () [inline]

Constructor for UI fragment.

Creates our initial data items that we are going to log. Setting whether extended functionality is needed for each data item.

References AndroidApp.JSONHandlerSingleton.getInstance().

```

59         {
60
61         dataList = new SetOfDataItems();
62
63         /* Set up our data items that we will want to log & show on ListView */
64         dataList.add(new DataItem<Double>("Yaw", true));
65         dataList.add(new DataItem<Double>("Pitch", true));
66         dataList.add(new DataItem<Double>("Roll", true));
67         dataList.add(new DataItem<Boolean>("GPS Valid", false));
68         dataList.add(new DataItem<Integer>("Satellites", false));
69         dataList.add(new DataItem<Double>("Latitude", false));
70         dataList.add(new DataItem<Double>("Longitude", false));
71         dataList.add(new DataItem<Double>("Velocity (MPH)", true));
72         dataList.add(new DataItem<Double>("Altitude (FT)", true));
73         dataList.add(new DataItem<Boolean>("Date Valid", false));
74         dataList.add(new DataItem<String>("Date", false));
75         dataList.add(new DataItem<String>("Time", false));
76
77         /* Clear our tripData handler in singleton class */
78         ArrayList<JSONObject> tripData = JSONHandlerSingleton.getInstance().tripData;
79         tripData.clear();
80     }

```

5.19.3 Member Function Documentation

5.19.3.1 onCreateView()

View AndroidApp.RealtimeFragment.onCreateView (
 LayoutInflater inflater,
 ViewGroup container,
 Bundle savedInstanceState) [inline]

Function called when fragment is shown on UI.

Sets up the UI ListView and Buttons.

Parameters

<i>inflater</i>	- Inflater used for displaying view.
<i>container</i>	- Container that the view will be displayed on.
<i>savedInstanceState</i>	- Last known state of this fragment.

Returns

View - The UI view of this fragment.

```

94                                     {
95         View myView = inflater.inflate(R.layout.realtime_layout, container, false);
96
97         textStatus = (TextView)myView.findViewById(R.id.realtime_status);
98
99         /* Get the ListView via ID */
100        ListView lvDataItems = (ListView) myView.findViewById(R.id.realtime_data_list);
101
102        /* Inflate the header view for ListView */
103        ViewGroup headerView = (ViewGroup) inflater.inflate(R.layout.data_list_header, lvDataItems, false);
104        lvDataItems.addHeaderView(headerView);
105
106        /* Create our new list adapter for our data list view */
107        lvAdapter = new DataListAdapter(getActivity(), R.layout.data_list_item,
dataList);
108        lvDataItems.setAdapter(lvAdapter);
109
110        /* Set our listeners for buttons */
111        FloatingActionButton mapButton = (FloatingActionButton) myView.findViewById(R.id.realtime_show_map)
;
112        mapButton.setOnClickListener(new MapButtonListener());
113
114        return myView;
115    }

```

5.19.3.2 newData()

```

final void AndroidApp.RealtimeFragment.newData (
    JSONObject jsonData ) [inline], [private]

```

Function for adding new statistics when received via bluetooth.

Attempts to break the initial JSON object into it's child objects and then retrieve the data from these child nodes.

Parameters

<i>jsonData</i>	- Received JSONObject from receive handler.
-----------------	---

References AndroidApp.JSONHandlerSingleton.getInstance(), AndroidApp.SetOfDataItems.getItemByName(), and AndroidApp.DataItem< T >.setCurrent().

```

125                                     {
126
127        try {
128            /*
129             * Add parent object to our singleton trip data list.
130             * This is so that other activities/fragments can access it.
131             */
132            ArrayList<JSONObject> tripData = JSONHandlerSingleton.getInstance().tripData;

```

```

133         tripData.add(jsonData);
134
135         /* Get the child JSON objects from parents. */
136         JSONObject orientObject = jsonData.getJSONObject("orientation");
137         JSONObject gpsObject = jsonData.getJSONObject("gps");
138         JSONObject timeObject = jsonData.getJSONObject("time");
139
140         /* Update our data items */
141         dataList.getItemByName("Yaw").setCurrent(orientObject.getDouble(
"yaw"));
142         dataList.getItemByName("Pitch").setCurrent(orientObject.
getDouble("pitch"));
143         dataList.getItemByName("Roll").setCurrent(orientObject.getDouble
("roll"));
144
145         /* Add GPS based data to */
146         dataList.getItemByName("GPS Valid").setCurrent(gpsObject.
getBoolean("gps_valid"));
147         dataList.getItemByName("Satellites").setCurrent(gpsObject.getInt
("available"));
148         dataList.getItemByName("Latitude").setCurrent(gpsObject.
getDouble("lat"));
149         dataList.getItemByName("Longitude").setCurrent(gpsObject.
getDouble("lng"));
150         dataList.getItemByName("Velocity (MPH)").
setCurrent(gpsObject.getDouble("vel_mph"));
151         dataList.getItemByName("Altitude (FT)").
setCurrent(gpsObject.getDouble("alt_ft"));
152
153         /* DateTime based data */
154         dataList.getItemByName("Date Valid").setCurrent(timeObject.
getBoolean("time_valid"));
155
156         Calendar cal = Calendar.getInstance();
157         cal.clear();
158         cal.set(Calendar.YEAR, timeObject.getInt("year"));
159         cal.set(Calendar.MONTH, timeObject.getInt("month"));
160         cal.set(Calendar.DATE, timeObject.getInt("day"));
161
162         cal.set(Calendar.HOUR, timeObject.getInt("hour"));
163         cal.set(Calendar.MINUTE, timeObject.getInt("minute"));
164         cal.set(Calendar.SECOND, timeObject.getInt("second"));
165         cal.set(Calendar.MILLISECOND, timeObject.getInt("centiseconds") * 10);
166
167         /* Create format for date and times then add to list */
168         DateFormat dateFormat = new SimpleDateFormat("dd/MM/yy");
169         dataList.getItemByName("Date").setCurrent(dateFormat.format(cal.
getTime()));
170
171         DateFormat timeFormat = new SimpleDateFormat("HH:mm:ss.SS");
172         dataList.getItemByName("Time").setCurrent(timeFormat.format(cal.
getTime()));
173
174         lvAdapter.notifyDataSetChanged();
175
176         textStatus.setText("Reading count: " + Integer.toString(tripData.size()));
177     } catch (JSONException e) {
178         /* Do nothing */
179     }
180 }

```

5.19.4 Member Data Documentation

5.19.4.1 RXHandler

```
final Handler AndroidApp.RealtimeFragment.RXHandler
```

Initial value:

```

= new Handler(Looper.getMainLooper()) {

    @Override

```

```

public void handleMessage(Message msg) {

    Bundle msgData = msg.getData();
    String jsonString = msgData.getString("JSON");

    if (jsonString != null) {

        try {
            JSONObject tmpJSON = new JSONObject(jsonString);
            newData(tmpJSON);
        } catch (JSONException e) {

        }

    }

}
}

```

Handler used for receiving statistics via bluetooth.

Receives data in a bundle passed from the bluetooth connection thread. This is due to multithreading as safe data exchange between threads has to be done via messages. Attempts to parse the data into a JSON object, if successful this data is then passed to our JSON adding procedure.

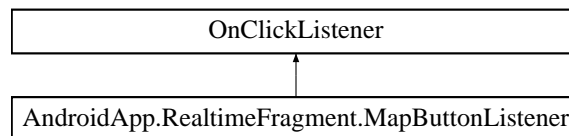
The documentation for this class was generated from the following file:

- android-app/app/src/main/java/com/jack/motorbikestatistics/[RealtimeFragment.java](#)

5.20 AndroidApp.RealtimeFragment.MapButtonListener Class Reference

Listener for starting a map activity when button pressed.

Inheritance diagram for AndroidApp.RealtimeFragment.MapButtonListener:



Public Member Functions

- void [onClick](#) (View v)
Function for handling when map button pressed.

5.20.1 Detailed Description

Listener for starting a map activity when button pressed.

5.20.2 Member Function Documentation

5.20.2.1 onClick()

```

void AndroidApp.RealtimeFragment.MapButtonListener.onClick (
    View v ) [inline]

```

Function for handling when map button pressed.

Created a new intent to start our map activity. Serialised statistics are then added as a bundle in the intent.

Parameters

v	- Current view of the button.
---	-------------------------------

```

196                                     {
197         Intent intent = new Intent(getActivity(), MapsActivity.class);
198         startActivity(intent);
199     }

```

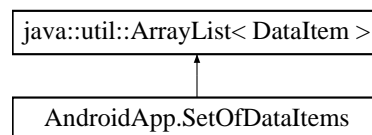
The documentation for this class was generated from the following file:

- android-app/app/src/main/java/com/jack/motorbikestatistics/[RealtimeFragment.java](#)

5.21 AndroidApp.SetOfDataItems Class Reference

ArrayList extension to allow searching via item name.

Inheritance diagram for AndroidApp.SetOfDataItems:

**Public Member Functions**

- [SetOfDataItems](#) ()
Constructor, just calls inherited constructor method.
- [DataItem getItemByName](#) (String name)
Function to allow searching of ArrayList<DataItem> via name.

5.21.1 Detailed Description

ArrayList extension to allow searching via item name.

5.21.2 Member Function Documentation

5.21.2.1 getItemByName()

```

DataItem AndroidApp.SetOfDataItems.getItemByName (
    String name ) [inline]

```

Function to allow searching of ArrayList<DataItem> via name.

Loops through all items in array until one item with matching name is found. This is then returned by the function.

Parameters

<i>name</i>	- Name to match.
-------------	------------------

Returns

[DataItem](#) - The item with matching name.

References `AndroidApp.DataItem< T >.getName()`.

```

36                                     {
37         DataItem result = null;
38
39         for (DataItem item: this) {
40             if (item.getName().equals(name)) {
41                 result = item;
42                 break;
43             }
44         }
45
46         return result;
47     }

```

The documentation for this class was generated from the following file:

- `android-app/app/src/main/java/com/jack/motorbikestatistics/SetOfDataItems.java`

5.22 AndroidApp.TripItem Class Reference

Class used for holding name and size information relating to a trip.

Public Member Functions

- [TripItem](#) (String name, int size)
Constructor for creating of a [TripItem](#).
- String [getTripName](#) ()
Getter for trip name.
- void [setTripName](#) (String [tripName](#))
Setter for trip name.
- int [getFileSize](#) ()
Getter for trip filesize.
- void [setFileSize](#) (int [fileSize](#))
Setter for trip filesize.

Private Attributes

- String [tripName](#) = null
The trips name on the uSD card.
- int [fileSize](#) = 0
The trips file size on the uSD card.

5.22.1 Detailed Description

Class used for holding name and size information relating to a trip.

5.22.2 Constructor & Destructor Documentation

5.22.2.1 TriplItem()

```
AndroidApp.TriplItem.TriplItem (
    String name,
    int size ) [inline]
```

Constructor for creating of a [TriplItem](#).

Sets the original file name and size.

Parameters

<i>name</i>	- Trip name.
<i>size</i>	- Size of the file.

```
31
32     this.tripName = name;
33     this.fileSize = size;
34 }
```

5.22.3 Member Function Documentation

5.22.3.1 getTripName()

```
String AndroidApp.TriplItem.getTripName ( ) [inline]
```

Getter for trip name.

Returns

String - Trip name.

```
40
41     return tripName;
42 }
```

5.22.3.2 setTripName()

```
void AndroidApp.TriplItem.setTripName (
    String tripName ) [inline]
```

Setter for trip name.

Parameters

<i>tripName</i>	- New trip name.
-----------------	------------------

```
48                                     {
49         this.tripName = tripName;
50     }
```

5.22.3.3 getFileSize()

```
int AndroidApp.TripItem.getFileSize ( ) [inline]
```

Getter for trip filesize.

Returns

int - Filesize in bytes.

```
56                                     {
57         return fileSize;
58     }
```

5.22.3.4 setFileSize()

```
void AndroidApp.TripItem.setFileSize (
    int fileSize ) [inline]
```

Setter for trip filesize.

Parameters

<i>fileSize</i>	- New trip filesize.
-----------------	----------------------

```
64                                     {
65         if (fileSize >= 0) {
66             this.fileSize = fileSize;
67         }
68     }
```

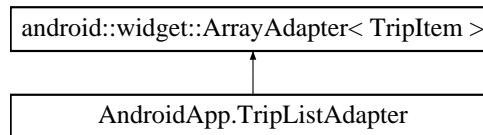
The documentation for this class was generated from the following file:

- android-app/app/src/main/java/com/jack/motorbikestatistics/TripItem.java

5.23 AndroidApp.TripListAdapter Class Reference

Adapter class used for displaying all trips.

Inheritance diagram for AndroidApp.TripListAdapter:



Classes

- class [ViewHolder](#)
Class that holds all UI data to be displayed for each ListItem.

Public Member Functions

- [TripListAdapter](#) (Context cnt, int [layoutResourceId](#), ArrayList< [TripItem](#) > [data](#))
Constructor for the ListView adapter.
- View [getView](#) (int position, View convertView, ViewGroup parent)
Function for returning the view of each list item ([TripItem](#)).

Private Attributes

- Context [context](#)
Context that the ListView is operating in.
- int [layoutResourceId](#)
Resource ID for current layout.
- ArrayList< [TripItem](#) > [data](#)
ArrayList of all trip items to display.

5.23.1 Detailed Description

Adapter class used for displaying all trips.

5.23.2 Constructor & Destructor Documentation

5.23.2.1 TripListAdapter()

```

AndroidApp.TripListAdapter.TripListAdapter (
    Context cnt,
    int layoutResourceId,
    ArrayList< TripItem > data ) [inline]
  
```

Constructor for the ListView adapter.

Calls the constructor of the superclass as well as setting other relevant information needed.

Parameters

<i>cnt</i>	- Context of the adapter to be operating in.
<i>layoutResourceId</i>	- Resource ID for current layout.
<i>data</i>	- ArrayList of statistics to display in ListView.

```

47                                     {
48         super(cnt, layoutResourceId, data);
49
50         this.context = cnt;
51         this.layoutResourceId = layoutResourceId;
52         this.data = data;
53     }

```

5.23.3 Member Function Documentation

5.23.3.1 getView()

```

View AndroidApp.TripListAdapter.getView (
    int position,
    View convertView,
    ViewGroup parent ) [inline]

```

Function for returning the view of each list item ([Tripltem](#)).

If a view for selected item has not been created inflater initialises it. A holder is then used to hold all the information that will be displayed on the UI to the user.

Parameters

<i>position</i>	- Index of item in array to use/reference to.
<i>convertView</i>	- View to be used for specified item.
<i>parent</i>	- Object where the created view will be placed on.

Returns

View - The result view of item with updated/current information.

References [AndroidApp.TripItem.getFileSize\(\)](#), and [AndroidApp.TripItem.getTripName\(\)](#).

```

77                                     {
78         ViewHolder holder;
79
80         if (convertView == null)
81         {
82             /* If view does not already exist. */
83             LayoutInflater inflater = (LayoutInflater)context.getSystemService(Context.
LAYOUT_INFLATER_SERVICE);
84             convertView = inflater.inflate(layoutResourceId, parent, false);
85
86             holder = new ViewHolder();
87             holder.name = (TextView)convertView.findViewById(R.id.triplist_name);

```

```

88         holder.fileSize = (TextView)convertView.findViewById(R.id.triplist_size);
89         convertView.setTag(holder);
90     }
91     else
92     {
93         /* If view already exists. */
94         holder = (ViewHolder)convertView.getTag();
95     }
96
97     TripItem tripItem = getItem(position);
98
99     /* Set our holder with current data of item */
100    holder.name.setText(tripItem.getTripName());
101    holder.fileSize.setText(Integer.toString(tripItem.getFileSize()));
102
103    return convertView;
104 }

```

The documentation for this class was generated from the following file:

- [android-app/app/src/main/java/com/jack/motorbikestatistics/TripListAdapter.java](#)

5.24 AndroidApp.TripListAdapter.ViewHolder Class Reference

Class that holds all UI data to be displayed for each ListItem.

5.24.1 Detailed Description

Class that holds all UI data to be displayed for each ListItem.

The documentation for this class was generated from the following file:

- [android-app/app/src/main/java/com/jack/motorbikestatistics/TripListAdapter.java](#)

5.25 LoggingDevice::Orientation Class Reference

Class for dealing with [Orientation](#) functionality on logging device.

```
#include <Orientation.h>
```

Public Member Functions

- void [init](#) ()
Initialisation function for orientation module.
- bool [pollIMU](#) ()
Updates the IMU with newest values at 25Hz frequency.
- float [getYaw](#) ()
Returns the Yaw orientation of the device.
- float [getPitch](#) ()
Returns the Pitch orientation of the device.
- float [getRoll](#) ()
Returns the Roll orientation of the device.

Private Member Functions

- float [convertRawAccel](#) (int aRaw)
Converts a raw reading from accelerometer to a value in G.
- float [convertRawGyro](#) (int aRaw)
Converts a raw reading from gyro to a value in deg/sec.

Private Attributes

- Madgwick [IMUfilter](#)
Madgwick filter object uses to steady orientation readings.

5.25.1 Detailed Description

Class for dealing with [Orientation](#) functionality on logging device.

5.25.2 Member Function Documentation

5.25.2.1 convertRawAccel()

```
float Orientation::convertRawAccel (
    int aRaw ) [private]
```

Converts a raw reading from accelerometer to a value in G.

Parameters

<i>aRaw</i>	- Raw accelerometer axis value.
-------------	---------------------------------

Returns

float - Processed acceleration axis in G.

References [ACCEL_RANGE](#).

```
118 {
119     /*
120      * Since using 2G range.
121      * -2G maps to raw value of -32768
122      * +2G maps to raw value of +32767
123      */
124     float a = (aRaw * (float)ACCEL_RANGE) / 32768.0;
125     return a;
126 }
```

5.25.2.2 convertRawGyro()

```
float Orientation::convertRawGyro (
    int gRaw ) [private]
```

Converts a raw reading from gyro to a value in deg/sec.

Parameters

<i>gRaw</i>	- Raw gyroscope axis value.
-------------	-----------------------------

Returns

float - Processed rotation axis in deg/sec.

References `GYRO_RANGE`.

```
135 {
136     /*
137      * since we are using 250 degrees/seconds range
138      * -250 maps to a raw value of -32768
139      * +250 maps to a raw value of 32767
140      */
141     float g = (gRaw * (float)GYRO_RANGE) / 32768.0;
142     return g;
143 }
```

5.25.2.3 init()

```
void Orientation::init ( )
```

Initialisation function for orientation module.

Initialises the CurieIMU module with set ranges and rates, our Madgwick filter is also initialised with this information.

References `ACCEL_RANGE`, `GYRO_RANGE`, `IMU_FREQUENCY`, and `IMUfilter`.

```
46 {
47     /* Set up the Gyroscope + Accelerometer */
48     CurieIMU.begin();
49     CurieIMU.setGyroRate(IMU_FREQUENCY);
50     CurieIMU.setAccelerometerRate(IMU_FREQUENCY);
51     CurieIMU.setAccelerometerRange(ACCEL_RANGE);
52     CurieIMU.setGyroRange(GYRO_RANGE);
53
54     IMUfilter.begin(IMU_FREQUENCY);
55 }
```


5.25.2.4 pollIMU()

```
bool Orientation::pollIMU ( )
```

Updates the IMU with newest values at 25Hz frequency.

Function reads raw values from accelerometer and gyroscope and sends them to our Madgwick filter (IMUfilter).

This function needs to be called by the system as often as possible.

To ensure correct frequency of 25Hz if kept to a micros counter is in place.

Function will return true or false as of whether that call actually updated the IMU (depending on micros count check).

Returns

bool - Whether the IMU was actually updated.

References AXIS_X, AXIS_Y, AXIS_Z, convertRawAccel(), convertRawGyro(), IMU_FREQUENCY, IMUfilter, and NUMBER_AXIS.

```
71 {
72     static const unsigned long US_PER_READING = 1000000 / IMU_FREQUENCY;
73     static unsigned long usPrevious = micros();
74
75     bool result = false;
76     int accel_raw[NUMBER_AXIS];
77     int gyro_raw[NUMBER_AXIS];
78     float accel_g[NUMBER_AXIS];
79     float gyro_ds[NUMBER_AXIS];
80     unsigned long usNow;
81
82     /* Ensures we stick to the sample rate (by not sampling too early) */
83     usNow = micros();
84     if ((usNow - usPrevious) >= US_PER_READING)
85     {
86         /* Read raw data from the IMU */
87         CurieIMU.readMotionSensor(accel_raw[AXIS_X], accel_raw[AXIS_Y], accel_raw[
88             AXIS_Z],
89                                     gyro_raw[AXIS_X], gyro_raw[AXIS_Y], gyro_raw[AXIS_Z]);
90
91         /* Convert raw readings from IMU to accel (G) and rotation vel (deg/s) */
92         accel_g[AXIS_X] = convertRawAccel(accel_raw[AXIS_X]);
93         accel_g[AXIS_Y] = convertRawAccel(accel_raw[AXIS_Y]);
94         accel_g[AXIS_Z] = convertRawAccel(accel_raw[AXIS_Z]);
95         gyro_ds[AXIS_X] = convertRawGyro(gyro_raw[AXIS_X]);
96         gyro_ds[AXIS_Y] = convertRawGyro(gyro_raw[AXIS_Y]);
97         gyro_ds[AXIS_Z] = convertRawGyro(gyro_raw[AXIS_Z]);
98
99         /* Update the filter. Orientation is calculated here */
100         IMUfilter.updateIMU(gyro_ds[AXIS_X], gyro_ds[AXIS_Y], gyro_ds[AXIS_Z],
101                             accel_g[AXIS_X], accel_g[AXIS_Y], accel_g[AXIS_Z]);
102
103         /* Increment previous counter */
104         usPrevious += US_PER_READING;
105
106         result = true;
107     }
108     return result;
109 }
```

5.25.2.5 getYaw()

```
float Orientation::getYaw ( )
```

Returns the Yaw orientation of the device.

Returns

float - Yaw orientation.

References IMUfilter.

```
150 {  
151     return IMUfilter.getYaw();  
152 }
```

5.25.2.6 getPitch()

```
float Orientation::getPitch ( )
```

Returns the Pitch orientation of the device.

Returns

float - Pitch orientation.

References IMUfilter.

```
159 {  
160     return IMUfilter.getPitch();  
161 }
```

5.25.2.7 getRoll()

```
float Orientation::getRoll ( )
```

Returns the Roll orientation of the device.

Returns

float - Roll orientation.

References IMUfilter.

```
168 {  
169     return IMUfilter.getRoll();  
170 }
```

The documentation for this class was generated from the following files:

- logging-device/Orientation.h
- logging-device/[Orientation.cpp](#)

5.26 LoggingDevice::Storage Class Reference

Class for storing & retrieving data on the logging device.

```
#include <Storage.h>
```

Public Member Functions

- void `init` ()
Initialisation function for storage module.
- bool `saveToFile` (char data[], bool newLine)
Saves a single line of data to a file.
- bool `generateFileName` ()
Generates a new filename to use for saving.
- void `loadTripNames` ()
Loads the information of all trips and sends them over bluetooth.
- void `loadSavedTrip` ()
Loads a saved trip and sends data to client via Serial.

Private Attributes

- char `fileName` [30]
File name to use when saving data.
- StaticJsonBuffer< 200 > `jsonFileBuffer`
Allocated space for holding JSON objects within.
- JsonObject & `fileJSON` = jsonFileBuffer.createObject()
JSON object that holds file information (size + name)

5.26.1 Detailed Description

Class for storing & retrieving data on the logging device.

5.26.2 Member Function Documentation

5.26.2.1 `init()`

```
void Storage::init ( )
```

Initialisation function for storage module.

Responsible for starting the uSD library.

References `USD_CS`.

```
40 {
41     SD.begin(USD_CS);
42 }
```

5.26.2.2 saveToFile()

```
bool Storage::saveToFile (
    char data[],
    bool newLine )
```

Saves a single line of data to a file.

Opens a handle to the current fileName. If the file exists data is appended, if not the file is created first.

Parameters

<i>data</i>	- Character array of data to save.
<i>newLine</i>	- Whether to add new line character at end of line.

Returns

bool - Whether saving was a success.

References fileName.

```
55 {
56     bool result = false;
57
58     /* Create handle to log file */
59     File logHandle = SD.open(fileName, FILE_WRITE);
60
61     /* If handle exists print line to file */
62     if (logHandle)
63     {
64
65         /* Print line, option to add newline characters */
66         logHandle.print(data);
67         if (newLine)
68         {
69             logHandle.println();
70         }
71
72         logHandle.close();
73         result = true;
74     }
75     return result;
76 }
```

5.26.2.3 generateFileName()

```
bool Storage::generateFileName ( )
```

Generates a new filename to use for saving.

Searches through existing files using pattern PREFIX_ID.SUFFIX
Existing files are skipped, once non-existent is found that is used.

Returns

bool - Whether a valid file name was able to be found.

References fileName, LOG_EXTENSION, LOG_NAME, and MAX_LOG_FILES.

```

87 {
88     bool result = false;
89     int i = 0;
90
91     for (i = 0; i < MAX_LOG_FILES; i++)
92     {
93         /* Clear name of log file */
94         memset(fileName, 0, strlen(fileName));
95
96         /* Set the new log file name to: trip_XXXXX.json */
97         sprintf(fileName, "%s%d.%s", LOG_NAME, i, LOG_EXTENSION);
98
99         if (!SD.exists(fileName))
100         {
101             /* If a file doesn't exist */
102             result = true;
103             break;
104         }
105     }
106
107     return result;
108 }
```

5.26.2.4 loadTripNames()

```
void Storage::loadTripNames ( )
```

Loads the information of all trips and sends them over bluetooth.

Searches directory for trips, then sends trip's name & size over serial.

References BT_SERIAL, and fileJSON.

```

116 {
117     bool filesRemaining = true;
118
119     File root = SD.open("/");
120
121     /* Try to open directory for logs */
122     if (root)
123     {
124         /* Ensure starting from start of directory */
125         root.rewindDirectory();
126
127         while (filesRemaining == true)
128         {
129             /* Try open handle for next file */
130             File entry = root.openNextFile();
131             if (entry)
132             {
133                 if (entry.isDirectory() == false)
134                 {
135                     /* Print out file name & size */
136                     fileJSON["name"] = entry.name();
137                     fileJSON["size"] = entry.size();
138
139                     fileJSON.printTo(BT_SERIAL);
140                     BT_SERIAL.println();
141                 }
142                 entry.close();
143             }
144             else
145             {
146                 /* No more files remaining in directory */
147                 filesRemaining = false;
148             }
149         }
150
151         root.close();
152     }
153 }
```

5.26.2.5 loadSavedTrip()

```
void Storage::loadSavedTrip ( )
```

Loads a saved trip and sends data to client via Serial.

Waits for the filename to be received via serial. Once file name is received, procedure attempts to open the file. If the file exists it then sends all bytes in the file via Serial.

References BT_SERIAL, and LOG_EXTENSION.

```
163 {
164     bool nameComplete = false;
165     String fileToOpen = "";
166
167     while (nameComplete == false)
168     {
169         /* Keep reading input in serial until file name is found */
170         if (BT_SERIAL.available() > 0)
171         {
172             char recvByte = BT_SERIAL.read();
173             fileToOpen += recvByte;
174
175             /* Wait until extension is found, then we know full file name */
176             if (fileToOpen.endsWith(LOG_EXTENSION))
177             {
178                 nameComplete = true;
179             }
180         }
181     }
182
183     /* Check if file exists */
184     if (SD.exists(fileToOpen))
185     {
186         /* Open file, then read out data byte by byte */
187         File handle = SD.open(fileToOpen);
188         if (handle)
189         {
190
191             while (handle.available())
192             {
193                 char readByte = handle.read();
194
195                 BT_SERIAL.write(readByte);
196             }
197
198             handle.close();
199         }
200     }
201 }
```

The documentation for this class was generated from the following files:

- logging-device/Storage.h
- logging-device/[Storage.cpp](#)

Chapter 6

File Documentation

6.1 `android-app/app/src/main/java/com/jack/motorbikestatistics/BTConnection.java` File Reference

Class for holding containing bluetooth connection on app.

Classes

- class [AndroidApp.BTConnection](#)
Thread class for a new bluetooth connection to a device.

6.1.1 Detailed Description

Class for holding containing bluetooth connection on app.

Class runs in a separate thread to main UI allowing for concurrent transmission and receiving of data to/from the logging device.

Author

Jack Allister - 23042098

Date

2016-2017

6.2 `android-app/app/src/main/java/com/jack/motorbikestatistics/BTDeviceItem.java` File Reference

UI class for holding information regarding a bluetooth device.

Classes

- class [AndroidApp.BTDeviceItem](#)
Class used for holding core UI information of a bluetooth devices.

6.2.1 Detailed Description

UI class for holding information regarding a bluetooth device.

Implemented for the ListView that shows unpaired/paired & connected bluetooth devices.

Author

Jack Allister - 23042098

Date

2016-2017

6.3 android-app/app/src/main/java/com/jack/motorbikestatistics/BTDeviceListAdapter.java File Reference

UI ListView adapter to display bluetooth devices.

Classes

- class [AndroidApp.BTDeviceListAdapter](#)
Adapter class used for displaying bluetooth devices.
- class [AndroidApp.BTDeviceListAdapter.ViewHolder](#)
Class that holds all data displayed for each ListItem.

6.3.1 Detailed Description

UI ListView adapter to display bluetooth devices.

Implemented so that the device ListView can display relevant information relating to the BluetoothDevice's that are available to pair, connect.

Author

Jack Allister - 23042098

Date

2016-2017

6.4 android-app/app/src/main/java/com/jack/motorbikestatistics/DataItem.java File Reference

UI class for holding information regarding a specific statistic.

Classes

- class [AndroidApp.DataItem< T >](#)

Class used for holding and displaying a piece of data within the statistic ListView UI.

6.4.1 Detailed Description

UI class for holding information regarding a specific statistic.

Implementation of generic class to allow multiple data types android added functionality such as averaging, minimum and maximum.

Author

Jack Allister - 23042098

Date

2016-2017

6.5 android-app/app/src/main/java/com/jack/motorbikestatistics/DataListAdapter.java File Reference

UI ListView adapter to display statistics.

Classes

- class [AndroidApp.DataListAdapter](#)
Adapter class used for displaying statistics.
- class [AndroidApp.DataListAdapter.ViewHolder](#)
Class that holds all data displayed for each ListItem.

6.5.1 Detailed Description

UI ListView adapter to display statistics.

Implemented so that the statistics ListView can display relevant information relating to the statistic such as name, value, average, min & max.

Author

Jack Allister - 23042098

Date

2016-2017

6.6 android-app/app/src/main/java/com/jack/motorbikestatistics/JSONHandlerSingleton.java File Reference

Singleton class that holds an array of the app's JSON data.

Classes

- class [AndroidApp.JSONHandlerSingleton](#)
Singleton class for holding all JSON trip data.

6.6.1 Detailed Description

Singleton class that holds an array of the app's JSON data.

Implementation of Singleton design pattern to allow cross activity/fragment data sharing of JSON trip data.

Author

Jack Allister - 23042098

Date

2016-2017

6.7 android-app/app/src/main/java/com/jack/motorbikestatistics/LoadDeviceFragment.java File Reference

Fragment/Tab for providing UI for loading from device.

Classes

- class [AndroidApp.LoadDeviceFragment](#)
UI Class for loading saved trips from device.
- class [AndroidApp.LoadDeviceFragment.TripItemListener](#)
Listener used to identify when a trip has been pressed.

6.7.1 Detailed Description

Fragment/Tab for providing UI for loading from device.

UI to allow the user to load saved trips stored on the uSD of the logging device.

Author

Jack Allister - 23042098

Date

2016-2017

6.8 android-app/app/src/main/java/com/jack/motorbikestatistics/MainActivity.java File Reference

Main activity class responsible for tabbing.

Classes

- class [AndroidApp.MainActivity](#)
Main activity class for fragment navigation.

6.8.1 Detailed Description

Main activity class responsible for tabbing.

Responsible for navigation between each fragment/tab. Sends relevant commands to switch system modes on the logging device as well.

Author

Jack Allister - 23042098

Date

2016-2017

6.9 android-app/app/src/main/java/com/jack/motorbikestatistics/MapsActivity.java File Reference

Maps activity class responsible for showing data on Google Maps.

Classes

- class [AndroidApp.MapsActivity](#)
Maps activity class for displaying map data.
- class [AndroidApp.MapsActivity.ZoomToggleListener](#)
Listener class for making markers invisible when zoomed out.
- class [AndroidApp.MapsActivity.StatisticWindowAdapter](#)
Adapter used for displaying statistics at a certain marker that user has clicked on.

6.9.1 Detailed Description

Maps activity class responsible for showing data on Google Maps.

Responsible for showing trip data on google maps. Places clickable points 5m away from each other showing stats at that point.

Author

Jack Allister - 23042098

Date

2016-2017

6.10 android-app/app/src/main/java/com/jack/motorbikestatistics/PairDeviceFragment.java File Reference

Fragment/Tab for connecting to the logging device.

Classes

- class [AndroidApp.PairDeviceFragment](#)
UI Class for discovering, pairing and connecting to the logging device.
- class [AndroidApp.PairDeviceFragment.DiscoverReceiver](#)
Receiver for when a new device is discovered.
- class [AndroidApp.PairDeviceFragment.DiscoverButtonListener](#)
Listener for when discovery button is pressed.
- class [AndroidApp.PairDeviceFragment.DeviceItemListener](#)
Listener for when a ListView item is pressed (to connect).

6.10.1 Detailed Description

Fragment/Tab for connecting to the logging device.

Implements Android's bluetooth API to discover, pair and connecting to the logging device.

Communication to the logging device is done via using Serial data mode.

Author

Jack Allister - 23042098

Date

2016-2017

6.11 android-app/app/src/main/java/com/jack/motorbikestatistics/RealtimeFragment.java File Reference

Fragment/Tab for viewing streamed statistics.

Classes

- class [AndroidApp.RealtimeFragment](#)
UI Class for viewing data sent from the logging device.
- class [AndroidApp.RealtimeFragment.MapButtonListener](#)
Listener for starting a map activity when button pressed.

6.11.1 Detailed Description

Fragment/Tab for viewing streamed statistics.

Implements RXHandler from bluetooth device to receive statistics. Data is then displayed in a ListView as well as option to view via Google Maps.

Author

Jack Allister - 23042098

Date

2016-2017

6.12 android-app/app/src/main/java/com/jack/motorbikestatistics/SetOfDataItems.java File Reference

Extension of ArrayList allows for searching via name.

Classes

- class [AndroidApp.SetOfDataItems](#)
ArrayList extension to allow searching via item name.

6.12.1 Detailed Description

Extension of ArrayList allows for searching via name.

This class is created to allow RealtimeFragment to search items by name. Simple searches through all items for a matching name.

Author

Jack Allister - 23042098

Date

2016-2017

6.13 android-app/app/src/main/java/com/jack/motorbikestatistics/TripItem.java File Reference

Class for holding information relating to a specific trip.

Classes

- class [AndroidApp.TripItem](#)

Class used for holding name and size information relating to a trip.

6.13.1 Detailed Description

Class for holding information relating to a specific trip.

Holds the trips name and file size. This information is used when loading a previous trip.

Author

Jack Allister - 23042098

Date

2016-2017

6.14 android-app/app/src/main/java/com/jack/motorbikestatistics/TripListAdapter.java File Reference

UI ListView adapter to display all saved trips.

Classes

- class [AndroidApp.TripListAdapter](#)
Adapter class used for displaying all trips.
- class [AndroidApp.TripListAdapter.ViewHolder](#)

Class that holds all UI data to be displayed for each ListItem.

6.14.1 Detailed Description

UI ListView adapter to display all saved trips.

Implemented so that the trip list ListView can display relevant information relating to the statistic such as name and file size.

Author

Jack Allister - 23042098

Date

2016-2017

6.15 logging-device/logging-device.ino File Reference

Arduino sketch for the logging device.

```
#include <SoftwareSerial.h>
#include <TinyGPS++.h>
#include <ArduinoJson.h>
#include "Orientation.h"
#include "Storage.h"
```

Macros

- `#define IDLE_CHAR '0'`
Command to set system to idle mode.
- `#define REALTIME_CHAR '1'`
Command to set system to realtime logging mode.
- `#define LIST_SAVED_CHAR '2'`
Command to list all saved trip names from uSD.
- `#define LOAD_TRIP_CHAR '3'`
Command to load a trip stored on uSD.
- `#define BT_SERIAL Serial1`
Mapping for which HW-Serial port BT module is on.
- `#define BT_BAUD 115200`
BAUD rate of BT device.
- `#define GPS_TX_PIN 9`
GPS serial transmit pin.
- `#define GPS_RX_PIN 8`
GPS serial receive pin.
- `#define GPS_BAUD 9600`
GPS serial baud rate.
- `#define LED_PIN 13`
LED pin to indicate read.

Enumerations

- `enum OPERATING_MODE { IDLE, REALTIME }`
Typedef holding two possible states for device.

Functions

- `SoftwareSerial serGPS (GPS_RX_PIN, GPS_TX_PIN)`
Serial object for communicating with GPS module.
- `void setup ()`
Runs once at boot of arduino.
- `void loop ()`
Main system loop for arduino.
- `bool parseNewMode (char modeChar, OPERATING_MODE &newMode)`
Returns whether system should change operating mode.

- void `realTimeMode ()`
Responsible for completing work needed in relitime mode.
- void `addOrientationToJSON ()`
Responsible for updating orientation JSON object with newest information.
- void `addGPSToJSON ()`
Responsible for updating GPS JSON object with newest information.
- void `addTimeToJSON ()`
Responsible for updating time JSON object with newest information.

Variables

- `OPERATING_MODE systemMode = IDLE`
State machine for system state of device.
- `Orientation orientation`
Orientation object, used for receiving device orientation.
- `Storage storage`
Storage object, responsible for saving & loading from uSD.
- `TinyGPSPlus gps`
Our GPS object, responsible for parsing NMEA codes.
- `StaticJsonBuffer< 500 > jsonBuffer`
Allocated space for holding all JSON objects within.
- `JsonObject & mainJSON = jsonBuffer.createObject()`
Parent JSON object, holds orientation, time & gps children.
- `JsonObject & orientJSON = mainJSON.createNestedObject("orientation")`
Holds all orientation related information.
- `JsonObject & gpsJSON = mainJSON.createNestedObject("gps")`
Holds all location related information.
- `JsonObject & timeJSON = mainJSON.createNestedObject("time")`
Holds all time related inforamtion.

6.15.1 Detailed Description

Arduino sketch for the logging device.

Author

Jack Allister - 23042098

Date

2016-2017

- Arduino 101
- Sparkfun GPS Logger shield
- Onboard gyroscope + accelerometer
- HC-06 Serial Bluetooth Module

6.15.2 Function Documentation

6.15.2.1 setup()

```
void setup ( )
```

Runs once at boot of arduino.

Responsible for setting up the peripherals.

Initialises modules such as storage, bluetooth & gps.

References BT_BAUD, BT_SERIAL, GPS_BAUD, LoggingDevice::Storage::init(), LoggingDevice::Orientation::init(), LED_PIN, and serGPS().

```
100 {
101   pinMode(LED_PIN, OUTPUT);
102
103   /* Initialise our created modules */
104   storage.init();
105   orientation.init();
106
107   /* Set up serial for wireless data transmission */
108   BT_SERIAL.begin(BT_BAUD);
109
110   /* Set up serial for GPS module */
111   serGPS.begin(GPS_BAUD);
112 }
```

6.15.2.2 loop()

```
void loop ( )
```

Main system loop for arduino.

Checks serial to see if any commands are available.

If available reads the byte and changes system mode relating to it.

System state machine is also iterated through each loop.

Relevant procedure depending on system state is then called.

References BT_SERIAL, parseNewMode(), and systemMode.

```
123 {
124
125   /* Check if mode change character received from front-end */
126   if (BT_SERIAL.available() > 0)
127   {
128     char modeChar = BT_SERIAL.read();
129
130     OPERATING_MODE newMode;
131
132     /* If valid new mode character found change system state */
133     if (parseNewMode(modeChar, newMode) == true)
134     {
135       systemMode = newMode;
136     }
137   }
138
139   /* State machine for choosing what option takes place */
140   switch (systemMode)
141   {
142     case IDLE:
143     {
144       /*
145        * In IDLE mode MCU does nothing.
146        * System waits and still parses incoming commands.
147        */
148       break;
149     }
150
151     case REALTIME:
152     {
153       realTimeMode();
154       break;
155     }
156   }
157 }
```

6.15.2.3 parseNewMode()

```
bool parseNewMode (
    char modeChar,
    OPERATING_MODE & newMode )
```

Returns whether system should change operating mode.

Parameters

<i>modeChar</i>	- The received command byte
<i>&newMode</i>	- Reference to new operating mode calculated via command.

Returns

bool - Whether a valid command was found.

References IDLE_CHAR.

```
167 {
168     bool result = true;
169     switch (modeChar)
170     {
171     case IDLE_CHAR:
172     {
173         newMode = IDLE;
174         break;
175     }
176     case REALTIME_CHAR:
177     {
178         /* Change mode and then generate new file name for new log */
179         if (systemMode != REALTIME)
180         {
181             /* Generate new name if not already in this mode */
182             storage.generateFileName();
183         }
184         newMode = REALTIME;
185         break;
186     }
187     case LIST_SAVED_CHAR:
188     {
189         /*
190          * Load all trips and send to application.
191          * Once we have finished sending trips we can go back to idle mode.
192          */
193         storage.loadTripNames();
194         newMode = IDLE;
195         break;
196     }
197     case LOAD_TRIP_CHAR:
198     {
199         /* Load a specific trip by file name */
200         storage.loadSavedTrip();
201         newMode = IDLE;
202         break;
203     }
204     default:
205     {
206         /*
207          * If not a valid operating mode character
208          * then return that parsing failed.
209          */
210         result = false;
211     }
212     }
213     return result;
214 }
```

6.15.2.4 realTimeMode()

```
void realTimeMode ( )
```

Responsible for completing work needed in reltime mode.

Every time called this procedure will poll the IMU to update our orientation class with newest information.

If available NMEA sentences received from GPS serial are sent to our GPS parsing object.

Every 1000ms all current information is transmitted via bluetooth, this information is also stored to the uSD so it can be retrieved at a later point.

References `addGPSToJSON()`, `addOrientationToJSON()`, `addTimeToJSON()`, `BT_SERIAL`, `gps`, `LED_PIN`, `mainJSON`, `LoggingDevice::Orientation::pollIMU()`, `LoggingDevice::Storage::saveToFile()`, and `serGPS()`.

```
236 {
237     static const unsigned int MAX_STRING_SIZE = 512;
238     static const unsigned long PRINT_DELAY = 1000;
239     static unsigned long lastMillis = 0;
240     char jsonString[MAX_STRING_SIZE];
241
242     /* Poll our IMU to update XYZ */
243     orientation.pollIMU();
244
245     /* Parse NMEA codes into GPS object */
246     while (serGPS.available() > 0)
247     {
248         gps.encode(serGPS.read());
249     }
250
251     /* Print orientation and location information */
252     if ((millis() - lastMillis) > PRINT_DELAY)
253     {
254         digitalWrite(LED_PIN, HIGH);
255
256         addOrientationToJSON();
257         addGPSToJSON();
258         addTimeToJSON();
259
260         /* Print our json object into a string */
261         mainJSON.printTo(jsonString, MAX_STRING_SIZE);
262
263         /* Log JSON to the microSD */
264         storage.saveToFile(jsonString, true);
265
266         /* Print to our bluetooth module */
267         BT_SERIAL.println(jsonString);
268
269         lastMillis = millis();
270         digitalWrite(LED_PIN, LOW);
271     }
272 }
```

6.15.2.5 addOrientationToJSON()

```
void addOrientationToJSON ( )
```

Responsible for updating orientation JSON object with newest information.

Interacts with devices Orientation object to get Yaw, Pitch & Roll.

References `LoggingDevice::Orientation::getPitch()`, `LoggingDevice::Orientation::getRoll()`, and `LoggingDevice::Orientation::getYaw()`.

```
282 {
283     orientJSON["yaw"] = orientation.getYaw();
284     orientJSON["pitch"] = orientation.getPitch();
285     orientJSON["roll"] = orientation.getRoll();
286 }
```

6.15.2.6 addGPSToJSON()

```
void addGPSToJSON ( )
```

Responsible for updating GPS JSON object with newest information.

Interacts with devices TinyGPSPlus object to get all locational/gps related information. Floats are cat'd to 6 digits max.

References gps.

```
297 {
298     /* Add location information */
299     gpsJSON["gps_valid"] = gps.location.isUpdated();
300     gpsJSON["lat"] = double_with_n_digits(gps.location.lat(), 6);
301     gpsJSON["lng"] = double_with_n_digits(gps.location.lng(), 6);
302
303     /* Other crucial GPS information */
304     gpsJSON["available"] = gps.satellites.value();
305     gpsJSON["vel_mph"] = gps.speed.mph();
306     gpsJSON["alt_ft"] = gps.altitude.feet();
307 }
```

6.15.2.7 addTimeToJSON()

```
void addTimeToJSON ( )
```

Responsible for updating time JSON object with newest information.

Interacts with devices TinyGPSPlus object to get time related information. This is because GPS module has a RTC (Realtime-Clock) kept via NMEA sentences.

References gps.

```
319 {
320     /* Add time information to JSON */
321     timeJSON["time_valid"] = gps.date.isValid() && gps.time.isValid();
322     timeJSON["day"] = gps.date.day();
323     timeJSON["month"] = gps.date.month();
324     timeJSON["year"] = gps.date.year();
325
326     timeJSON["hour"] = gps.time.hour();
327     timeJSON["minute"] = gps.time.minute();
328     timeJSON["second"] = gps.time.second();
329     timeJSON["centiseconds"] = gps.time.centisecond();
330 }
```

6.16 logging-device/Orientation.cpp File Reference

Module created to deal with all orientation related functionality.

```
#include <BMI160.h>
#include <CurieIMU.h>
#include "Orientation.h"
```

Macros

- `#define IMU_FREQUENCY 25`
Frequency of update rate for IMU (25Hz)
- `#define ACCEL_RANGE 2`
Range of accelerometer +-2G.
- `#define GYRO_RANGE 250`
Range of gyroscope +-250 deg/sec.
- `#define NUMBER_AXIS 3`
Number of axis for our IMU.
- `#define AXIS_X 0`
Reference to X axis in array.
- `#define AXIS_Y 1`
Reference to Y axis in array.
- `#define AXIS_Z 2`
Reference to Z axis in array.

6.16.1 Detailed Description

Module created to deal with all orientation related functionality.

Author

Jack Allister - 23042098

Date

2016-2017 Uses the built in Gyroscope & Accelerometer of the Arduino 101 to create an Inertial Measurement Unit (IMU).

6.17 logging-device/Storage.cpp File Reference

Module created to handle all storage related functionality.

```
#include <SD.h>
#include <ArduinoJson.h>
#include "Storage.h"
```

Macros

- `#define BT_SERIAL Serial1`
Mapping for which HW-Serial port BT module is on.
- `#define USD_CS 10`
Chip select pin for MicroSD card (SPI)
- `#define MAX_LOG_FILES 5000`
Maximum amount of log files that can be stored on the device.
- `#define LOG_NAME "TRIP_"`
The prefix of the name for logs.
- `#define LOG_EXTENSION "TXT"`
The suffix of the name for logs (file extension)

6.17.1 Detailed Description

Module created to handle all storage related functionality.

Author

Jack Allister - 23042098

Date

2016-2017 Handles saving, listing & loading of trips. Uses MicroSD available on the Sparkfun GPS logging shield.

Index

add
 AndroidApp::DataItem, 26
addGPSToJSON
 logging-device.ino, 89
addOrientationToJSON
 logging-device.ino, 89
addTimeToJSON
 logging-device.ino, 90
addTrip
 AndroidApp::LoadDeviceFragment, 34
android-app/app/src/main/java/com/jack/motorbikestatistics/
 BTConnection.java, 77
android-app/app/src/main/java/com/jack/motorbikestatistics/
 BTDeviceItem.java, 77
android-app/app/src/main/java/com/jack/motorbikestatistics/
 BTDeviceListAdapter.java, 78
android-app/app/src/main/java/com/jack/motorbikestatistics/
 DataItem.java, 79
android-app/app/src/main/java/com/jack/motorbikestatistics/
 DataListAdapter.java, 79
android-app/app/src/main/java/com/jack/motorbikestatistics/
 JSONHandlerSingleton.java, 80
android-app/app/src/main/java/com/jack/motorbikestatistics/
 LoadDeviceFragment.java, 80
android-app/app/src/main/java/com/jack/motorbikestatistics/
 MainActivity.java, 81
android-app/app/src/main/java/com/jack/motorbikestatistics/
 MapsActivity.java, 81
android-app/app/src/main/java/com/jack/motorbikestatistics/
 PairDeviceFragment.java, 82
android-app/app/src/main/java/com/jack/motorbikestatistics/
 RealtimeFragment.java, 82
android-app/app/src/main/java/com/jack/motorbikestatistics/
 SetOfDataItems.java, 83
android-app/app/src/main/java/com/jack/motorbikestatistics/
 TripItem.java, 83
android-app/app/src/main/java/com/jack/motorbikestatistics/
 TripListAdapter.java, 84
AndroidApp.BTConnection, 9
AndroidApp.BTDeviceItem, 15
AndroidApp.BTDeviceListAdapter, 18
AndroidApp.BTDeviceListAdapter.ViewHolder, 21
AndroidApp.DataItem< T >, 21
AndroidApp.DataListAdapter, 28
AndroidApp.DataListAdapter.ViewHolder, 30
AndroidApp.JSONHandlerSingleton, 31
AndroidApp.LoadDeviceFragment, 32
AndroidApp.LoadDeviceFragment.TripItemListener, 36
AndroidApp.MainActivity, 37
AndroidApp.MapsActivity, 40
AndroidApp.MapsActivity.StatisticWindowAdapter, 46
AndroidApp.MapsActivity.ZoomToogleListener, 47
AndroidApp.PairDeviceFragment, 49
AndroidApp.PairDeviceFragment.DeviceItemListener, 53
AndroidApp.PairDeviceFragment.DiscoverButton↔
 Listener, 54
AndroidApp.PairDeviceFragment.DiscoverReceiver, 55
AndroidApp.RealtimeFragment, 57
AndroidApp.RealtimeFragment.MapButtonListener, 61
AndroidApp.SetOfDataItems, 62
AndroidApp.TripItem, 63
AndroidApp.TripListAdapter, 66
AndroidApp.TripListAdapter.ViewHolder, 68
AndroidApp::BTConnection
 BTConnection, 10
 connect, 13
 getRXHandler, 11
 isConnected, 13
 isRunning, 13
 run, 11
 setRXHandler, 11
 txHandler, 14
AndroidApp::BTDeviceItem
 BTDeviceItem, 15
 getConnection, 16
 getDevice, 16
 getIconID, 17
 getStatus, 17
 setConnection, 16
 setIconID, 18
 setStatus, 17
AndroidApp::BTDeviceListAdapter
 BTDeviceListAdapter, 19
 getView, 20
AndroidApp::DataItem
 add, 26
 DataItem, 22, 23
 divide, 26
 getAverage, 24
 getCurrent, 24
 getEnabledAvgMinMax, 24
 getMaximum, 25
 getMinimum, 24
 getName, 23
 greaterThan, 27
 lessThan, 27
 setCurrent, 25

- AndroidApp::DataListAdapter
 - DataListAdapter, 29
 - getView, 29
- AndroidApp::JSONHandlerSingleton
 - getInstance, 31
- AndroidApp::LoadDeviceFragment
 - addTrip, 34
 - LoadDeviceFragment, 33
 - onCreateView, 33
 - RXHandler, 35
 - setBTConnection, 34
- AndroidApp::LoadDeviceFragment::TripltemListener
 - onItemClick, 36
- AndroidApp::MainActivity
 - onCreate, 38
 - onNavigationItemSelected, 39
- AndroidApp::MapsActivity
 - calcDistance, 43
 - convertDpToPixel, 44
 - findJSONByLatLng, 42
 - getBitmapDescriptor, 45
 - onCreate, 41
 - onMapReady, 43
- AndroidApp::MapsActivity::StatisticWindowAdapter
 - getInfoContents, 46
- AndroidApp::MapsActivity::ZoomToggleListener
 - setMarkersVisible, 48
- AndroidApp::PairDeviceFragment
 - getBTConnection, 52
 - getNeededPrivileges, 52
 - onCreateView, 50
 - PairDeviceFragment, 50
- AndroidApp::PairDeviceFragment::DeviceItemListener
 - onItemClick, 53
- AndroidApp::PairDeviceFragment::DiscoverButton↔
 - Listener
 - onCheckedChanged, 55
- AndroidApp::PairDeviceFragment::DiscoverReceiver
 - onReceive, 56
- AndroidApp::RealtimeFragment
 - newData, 59
 - onCreateView, 58
 - RXHandler, 60
 - RealtimeFragment, 58
- AndroidApp::RealtimeFragment::MapButtonListener
 - onClick, 61
- AndroidApp::SetOfDataItems
 - getItemByName, 62
- AndroidApp::Tripltem
 - getFileSize, 65
 - getTripName, 64
 - setFileSize, 65
 - setTripName, 64
 - Tripltem, 64
- AndroidApp::TripListAdapter
 - getView, 67
 - TripListAdapter, 66
- BTConnection
 - AndroidApp::BTConnection, 10
- BTDeviceItem
 - AndroidApp::BTDeviceItem, 15
- BTDeviceListAdapter
 - AndroidApp::BTDeviceListAdapter, 19
- calcDistance
 - AndroidApp::MapsActivity, 43
- connect
 - AndroidApp::BTConnection, 13
- convertDpToPixel
 - AndroidApp::MapsActivity, 44
- convertRawAccel
 - LoggingDevice::Orientation, 69
- convertRawGyro
 - LoggingDevice::Orientation, 69
- Dataltem
 - AndroidApp::Dataltem, 22, 23
- DataListAdapter
 - AndroidApp::DataListAdapter, 29
- divide
 - AndroidApp::Dataltem, 26
- findJSONByLatLng
 - AndroidApp::MapsActivity, 42
- generateFileName
 - LoggingDevice::Storage, 74
- getAverage
 - AndroidApp::Dataltem, 24
- getBTConnection
 - AndroidApp::PairDeviceFragment, 52
- getBitmapDescriptor
 - AndroidApp::MapsActivity, 45
- getConnection
 - AndroidApp::BTDeviceItem, 16
- getCurrent
 - AndroidApp::Dataltem, 24
- getDevice
 - AndroidApp::BTDeviceItem, 16
- getEnabledAvgMinMax
 - AndroidApp::Dataltem, 24
- getFileSize
 - AndroidApp::Tripltem, 65
- getIconID
 - AndroidApp::BTDeviceItem, 17
- getInfoContents
 - AndroidApp::MapsActivity::StatisticWindow↔
 - Adapter, 46
- getInstance
 - AndroidApp::JSONHandlerSingleton, 31
- getItemByName
 - AndroidApp::SetOfDataItems, 62
- getMaximum
 - AndroidApp::Dataltem, 25
- getMinimum
 - AndroidApp::Dataltem, 24
- getName
 - AndroidApp::Dataltem, 24

- AndroidApp::DataItem, 23
- getNeededPrivileges
 - AndroidApp::PairDeviceFragment, 52
- getPitch
 - LoggingDevice::Orientation, 72
- getRXHandler
 - AndroidApp::BTConnection, 11
- getRoll
 - LoggingDevice::Orientation, 72
- getStatus
 - AndroidApp::BTDeviceItem, 17
- getTripName
 - AndroidApp::TripItem, 64
- getView
 - AndroidApp::BTDeviceListAdapter, 20
 - AndroidApp::DataListAdapter, 29
 - AndroidApp::TripListAdapter, 67
- getYaw
 - LoggingDevice::Orientation, 71
- greaterThan
 - AndroidApp::DataItem, 27
- init
 - LoggingDevice::Orientation, 70
 - LoggingDevice::Storage, 73
- isConnected
 - AndroidApp::BTConnection, 13
- isRunning
 - AndroidApp::BTConnection, 13
- lessThan
 - AndroidApp::DataItem, 27
- LoadDeviceFragment
 - AndroidApp::LoadDeviceFragment, 33
- loadSavedTrip
 - LoggingDevice::Storage, 75
- loadTripNames
 - LoggingDevice::Storage, 75
- logging-device.ino
 - addGPSToJSON, 89
 - addOrientationToJSON, 89
 - addTimeToJSON, 90
 - loop, 87
 - parseNewMode, 87
 - realTimeMode, 88
 - setup, 86
- logging-device/Orientation.cpp, 90
- logging-device/Storage.cpp, 91
- logging-device/logging-device.ino, 85
- LoggingDevice::Orientation, 68
 - convertRawAccel, 69
 - convertRawGyro, 69
 - getPitch, 72
 - getRoll, 72
 - getYaw, 71
 - init, 70
 - pollIMU, 70
- LoggingDevice::Storage, 73
 - generateFileName, 74
 - init, 73
 - loadSavedTrip, 75
 - loadTripNames, 75
 - saveToFile, 73
- loop
 - logging-device.ino, 87
- newData
 - AndroidApp::RealtimeFragment, 59
- onCheckedChanged
 - AndroidApp::PairDeviceFragment::Discover↔
 - ButtonListener, 55
- onClick
 - AndroidApp::RealtimeFragment::MapButton↔
 - Listener, 61
- onCreate
 - AndroidApp::MainActivity, 38
 - AndroidApp::MapsActivity, 41
- onCreateView
 - AndroidApp::LoadDeviceFragment, 33
 - AndroidApp::PairDeviceFragment, 50
 - AndroidApp::RealtimeFragment, 58
- onItemClick
 - AndroidApp::LoadDeviceFragment::TripItem↔
 - Listener, 36
 - AndroidApp::PairDeviceFragment::DeviceItem↔
 - Listener, 53
- onMapReady
 - AndroidApp::MapsActivity, 43
- onNavigationItemSelected
 - AndroidApp::MainActivity, 39
- onReceive
 - AndroidApp::PairDeviceFragment::Discover↔
 - Receiver, 56
- PairDeviceFragment
 - AndroidApp::PairDeviceFragment, 50
- parseNewMode
 - logging-device.ino, 87
- pollIMU
 - LoggingDevice::Orientation, 70
- RXHandler
 - AndroidApp::LoadDeviceFragment, 35
 - AndroidApp::RealtimeFragment, 60
- realTimeMode
 - logging-device.ino, 88
- RealtimeFragment
 - AndroidApp::RealtimeFragment, 58
- run
 - AndroidApp::BTConnection, 11
- saveToFile
 - LoggingDevice::Storage, 73
- setBTConnection
 - AndroidApp::LoadDeviceFragment, 34
- setConnection
 - AndroidApp::BTDeviceItem, 16

- setCurrent
 - AndroidApp::DataItem, [25](#)
- setFileSize
 - AndroidApp::TriplItem, [65](#)
- setIconID
 - AndroidApp::BTDeviceItem, [18](#)
- setMarkersVisible
 - AndroidApp::MapsActivity::ZoomToggleListener, [48](#)
- setRXHandler
 - AndroidApp::BTConnection, [11](#)
- setStatus
 - AndroidApp::BTDeviceItem, [17](#)
- setTripName
 - AndroidApp::TriplItem, [64](#)
- setup
 - logging-device.ino, [86](#)
- TriplItem
 - AndroidApp::TriplItem, [64](#)
- TripListAdapter
 - AndroidApp::TripListAdapter, [66](#)
- txHandler
 - AndroidApp::BTConnection, [14](#)